QGIS Desktop 3.34 User Guide

QGIS Project

mai 20, 2025

Cuprins

1 Preambul **3**
1.1 Ce este nou în QGIS 3.34 . . . 4

2 Cuvânt înainte **5**

3 Convenții **7**
3.1 Convențiile GUI . . . 7
3.2 Convenții pentru Text și Tastatură . . . 8
3.3 Instrucțiuni specifice platformelor . . . 8

4 Funcțiuni **9**
4.1 Vizualizarea datelor . . . 9
4.2 Explorarea datelor și compunerea hărților . . . 10
4.3 Crearea, editarea, gestionarea și exportul datelor . . . 10
4.4 Analiza datelor . . . 11
4.5 Publicarea hărților de pe Internet . . . 11
4.6 Extinderea funcționalității QGIS prin intermediul plugin-urilor . . . 11
4.6.1 Plugin-uri de bază . . . 11
4.6.2 Plugin-urile Externe Python . . . 12
4.7 Consola Python . . . 12
4.8 Probleme Cunoscute . . . 12
4.8.1 Limitarea numărului de fișiere deschise . . . 12

5 Noțiuni de bază **13**
5.1 Instalarea QGIS . . . 13
5.1.1 Instalarea pornind de la pachetele binare . . . 13
5.1.2 Instalarea pornind de la sursă . . . 13
5.1.3 Instalare pe suport extern . . . 14
5.1.4 Descărcarea datelor demonstrative . . . 14
5.2 Pornirea și oprirea QGIS . . . 14
5.3 Sesiune de Lucru: Încărcarea straturilor raster și a celor vectoriale . . . 15

6 Lucrul cu Fișierele Proiectelor **23**
6.1 Prezentarea proiectelor QGIS . . . 23
6.2 Gestionarea căilor de fișiere întrerupte . . . 25
6.3 Obținerea rezultatelor . . . 26

7 QGIS GUI **27**
7.1 Bara de Meniuri . . . 28
7.1.1 Proiect . . . 28
7.1.2 Editare . . . 29

7.1.3 Vizualizare . . . 33
7.1.4 Strat . . . 35
7.1.5 Setări . . . 38
7.1.6 Plugin-uri . . . 39
7.1.7 Vector . . . 39
7.1.8 Raster . . . 41
7.1.9 Bază de date . . . 42
7.1.10 Web . . . 43
7.1.11 Suprafață poliedrică . . . 43
7.1.12 Procesare . . . 43
7.1.13 Ajutor . . . 44
7.1.14 QGIS . . . 44
7.2 Panouri și Bare de Instrumente . . . 45
7.2.1 Bare de instrumente . . . 45
7.2.2 Panouri . . . 46
7.3 Bara de Stare . . . 46
7.3.1 Bara locatorului . . . 46
7.3.2 Raportarea acțiunilor . . . 47
7.3.3 Controlul canevasului hărții . . . 47
7.3.4 Jurnalizarea . . . 48

8 Panoul Navigatorului **49**
8.1 Resources that can be opened / run from the Browser . . . 52
8.2 Browser panel top-level entries . . . 52
8.2.1 Favorites . . . 52
8.2.2 Semne de Carte Spațiale . . . 52
8.2.3 Project Home . . . 52
8.2.4 Drives and file system . . . 53
8.2.5 Database entries . . . 53
8.2.6 Tiles and Web Services . . . 56
8.3 Resources . . . 56

9 Configurare QGIS **59**
9.1 Opţiuni . . . 59
9.1.1 Setări Generale . . . 60
9.1.2 System Settings . . . 62
9.1.3 User Profiles Settings . . . 63
9.1.4 CRS and Transforms Settings . . . 63
9.1.5 Data Sources settings . . . 67
9.1.6 Rendering Settings . . . 71
9.1.7 Canvas and Legend Settings . . . 75
9.1.8 Map tools Settings . . . 77
9.1.9 Elevation settings . . . 81
9.1.10 3D settings . . . 81
9.1.11 Colors settings . . . 82
9.1.12 Setări Fonturi . . . 83
9.1.13 Layouts settings . . . 84
9.1.14 Variables settings . . . 85
9.1.15 Authentication settings . . . 85
9.1.16 Network settings . . . 86
9.1.17 Setări GPS . . . 88
9.1.18 Locator settings . . . 90
9.1.19 Acceleration settings . . . 91
9.1.20 IDE settings . . . 92
9.1.21 Processing settings . . . 95
9.1.22 Advanced settings . . . 96
9.2 Working with User Profiles . . . 97
9.2.1 The concept . . . 97

9.2.2 Setting user profile . . . 98
9.3 Proprietăţi Proiect . . . 99
9.3.1 General Properties . . . 99
9.3.2 Proprietățile Metadatelor . . . 101
9.3.3 View Settings . . . 102
9.3.4 Proprietăți CRS . . . 102
9.3.5 Transformations Properties . . . 103
9.3.6 Styles Properties . . . 103
9.3.7 Data Sources Properties . . . 104
9.3.8 Relations Properties . . . 106
9.3.9 Proprietățile Variabilelor . . . 107
9.3.10 Macros Properties . . . 107
9.3.11 Proprietățile Serverului QGIS . . . 108
9.3.12 Proprietăți Temporale . . . 109
9.3.13 Terrain Properties . . . 109
9.3.14 Sensors Properties . . . 111
9.4 Personalizare . . . 112
9.5 Combinații de taste . . . 114
9.6 Running QGIS with advanced settings . . . 115
9.6.1 Command line and environment variables . . . 115
9.6.2 Deploying QGIS within an organization . . . 120

10 Lucrul cu Proiecții **123**
10.1 Privire de ansamblu asupra Suportului Proiecției . . . 123
10.2 Layer Coordinate Reference Systems . . . 123
10.3 Project Coordinate Reference Systems . . . 125
10.4 Coordinate Reference System Selector . . . 127
10.5 Sistem de Referință și Coordonate Personalizat . . . 128
10.5.1 Integrate an NTv2-transformation in QGIS . . . 129
10.6 Datum Transformations . . . 130

11 Vizualizarea Hărţilor **133**
11.1 2D Map View . . . 133
11.1.1 Exploring the map view . . . 134
11.1.2 Controlling map rendering . . . 135
11.1.3 Time-based control on the map canvas . . . 136
11.1.4 Bookmarking extents on the map . . . 138
11.1.5 Decorating the map . . . 140
11.1.6 Instrumente de Adnotare . . . 147
11.1.7 Măsurarea . . . 153
11.1.8 Setting additional map views . . . 155
11.1.9 Exportarea vizualizării hărții . . . 156
11.2 3D Map View . . . 159
11.2.1 Scene Configuration . . . 160
11.2.2 Navigation options . . . 167
11.2.3 Creating an animation . . . 167
11.2.4 3D vector layers . . . 168
11.3 Elevation Profile View . . . 168
11.3.1 The interface . . . 169
11.3.2 Creating an elevation profile . . . 170
11.3.3 Interacting with the profile Views . . . 171

12 Instrumente generale **173**
12.1 Ajutor contextual . . . 173
12.2 Panouri . . . 173
12.2.1 Panoul Straturilor . . . 173
12.2.2 Layer Styling Panel . . . 180
12.2.3 Layer Order Panel . . . 183
12.2.4 Overview Panel . . . 183

12.2.5 Log Messages Panel . . . 184
12.2.6 Undo/Redo Panel . . . 184
12.2.7 Statistical Summary Panel . . . 184
12.2.8 Debugging/Development Tools Panel . . . 185
12.3 Embedding layers from external projects . . . 188
12.4 Interacting with features . . . 189
12.4.1 Selectare entități . . . 189
12.4.2 Identifying Features . . . 192
12.5 Salvarea și Partajarea Proprietăților Stratului . . . 197
12.5.1 Managing Custom Styles . . . 197
12.5.2 Storing Styles in a File or a Database . . . 198
12.5.3 Layer definition file . . . 200
12.6 Documenting your data . . . 200
12.6.1 Metadata . . . 200
12.6.2 Notițele stratului . . . 201
12.7 Storing values in Variables . . . 202
12.8 Autentificarea . . . 203
12.9 Common widgets . . . 204
12.9.1 Selectorul de Culoare . . . 204
12.9.2 Symbol Widget . . . 208
12.9.3 Remote or embedded file selector . . . 208
12.9.4 Visibility Scale Selector . . . 209
12.9.5 Spatial Extent Selector . . . 209
12.9.6 Font Selector . . . 210
12.9.7 Unit Selector . . . 211
12.9.8 Number Formatting . . . 212
12.9.9 Modurile de Fuziune . . . 213
12.9.10 Configurarea suprascrierii definită de date . . . 215

13 Lucrul cu expresii **219**
13.1 Expresii . . . 219
13.1.1 Constructorul expresiilor de tip șir . . . 219
13.1.2 Editorul de Funcții . . . 224
13.2 Lista funcțiilor . . . 226
13.2.1 Funcții de Agregare . . . 226
13.2.2 Array Functions . . . 236
13.2.3 Funcțiile Culorilor . . . 247
13.2.4 Funcții Condiționale . . . 254
13.2.5 Conversions Functions . . . 256
13.2.6 Funcții Personalizate . . . 261
13.2.7 Funcții pentru Dată și Oră . . . 261
13.2.8 Fields and Values . . . 272
13.2.9 Files and Paths Functions . . . 272
13.2.10 Form Functions . . . 275
13.2.11 Funcții pentru Potrivirea Fuzzy . . . 276
13.2.12 Funcții Generale . . . 277
13.2.13 Funcții Geometrice . . . 279
13.2.14 Layout Functions . . . 341
13.2.15 Map Layers . . . 342
13.2.16 Maps Functions . . . 344
13.2.17 Funcțiile Matematice . . . 348
13.2.18 Meshes Functions . . . 356
13.2.19 Operatori . . . 358
13.2.20 Processing Functions . . . 367
13.2.21 Rasters Functions . . . 367
13.2.22 Record and Attributes Functions . . . 369
13.2.23 Relations . . . 378
13.2.24 Sensors Functions . . . 378

13.2.25 Funcții pentru Șiruri . . . 378
13.2.26 User Expressions . . . 387
13.2.27 Variabile . . . 388
13.2.28 Funcții Recente . . . 391

14 Biblioteca de Stiluri **393**
14.1 The Style Manager . . . 393
14.1.1 The Style Manager dialog . . . 393
14.1.2 Setting a Color Ramp . . . 398
14.1.3 Creating a Legend Patch Shape . . . 400
14.2 The Symbol Selector . . . 402
14.2.1 The symbol layer tree . . . 403
14.2.2 Configuring a symbol . . . 404
14.3 Setting a label . . . 421
14.3.1 Formatting the label text . . . 423
14.3.2 Configuring interaction with labels . . . 431
14.4 Creating 3D Symbols . . . 440
14.4.1 Point Layers . . . 440
14.4.2 Line layers . . . 441
14.4.3 Polygon Layers . . . 442
14.4.4 Shading the texture . . . 444
14.4.5 Application example . . . 444

15 Gestiunea Surselor de Date **445**
15.1 Deschiderea datelor . . . 445
15.1.1 Panoul Navigatorului . . . 447
15.1.2 The DB Manager . . . 450
15.1.3 Provider-based loading tools . . . 451
15.1.4 The Layer Metadata Search Panel . . . 472
15.1.5 QGIS Custom formats . . . 473
15.1.6 QLR - QGIS Layer Definition File . . . 473
15.1.7 Connecting to web services . . . 473
15.2 Crearea Straturilor . . . 479
15.2.1 Creating new vector layers . . . 480
15.2.2 Creating new layers from an existing layer . . . 487
15.2.3 Crearea fișierelor DXF noi . . . 491
15.2.4 Creating new layers from the clipboard . . . 492
15.2.5 Creating virtual layers . . . 492
15.3 Explorarea Formatelor de Date și a Câmpurilor . . . 495
15.3.1 Datele raster . . . 495
15.3.2 Datele Vectoriale . . . 495
15.3.3 Layers crossing 180° longitude . . . 503

16 Lucrul cu Datele Vectoriale **505**
16.1 Dialogul Proprietăților Vectoriale . . . 505
16.1.1 Information Properties . . . 506
16.1.2 Proprietățile Sursei . . . 506
16.1.3 Symbology Properties . . . 509
16.1.4 Proprietățile Etichetelor . . . 531
16.1.5 Proprietățile Diagramelor . . . 542
16.1.6 Masks Properties . . . 546
16.1.7 3D View Properties . . . 547
16.1.8 Fields Properties . . . 549
16.1.9 Attributes Form Properties . . . 550
16.1.10 Joins Properties . . . 558
16.1.11 Auxiliary Storage Properties . . . 558
16.1.12 Proprietățile Acțiunilor . . . 567
16.1.13 Display Properties . . . 572
16.1.14 Rendering Properties . . . 575

16.1.15 Proprietăți Temporale . . . 577
16.1.16 Proprietățile Variabilelor . . . 578
16.1.17 Elevation Properties . . . 578
16.1.18 Proprietăți Metadate . . . 580
16.1.19 Dependencies Properties . . . 580
16.1.20 Proprietățile Legendei . . . 580
16.1.21 Proprietățile Serverului QGIS . . . 581
16.1.22 Digitizing Properties . . . 581
16.2 Lucrul cu Tabela de Atribute . . . 584
16.2.1 Foreword: Spatial and non-spatial tables . . . 584
16.2.2 Introducing the attribute table interface . . . 585
16.2.3 Interacting with features in an attribute table . . . 590
16.2.4 Using action on features . . . 592
16.2.5 Editarea valorilor atributului . . . 594
16.2.6 Exploring features attributes through the Identify Tool . . . 597
16.2.7 Storing and fetching an external resource . . . 598
16.3 Editarea . . . 600
16.3.1 Setting the snapping tolerance and search radius . . . 601
16.3.2 Opțiuni de Digitizare și Acroșare . . . 601
16.3.3 Editarea topologică . . . 604
16.3.4 Digitizarea unui strat vectorial existent . . . 605
16.3.5 Digitizare avansată . . . 615
16.3.6 Shape digitizing . . . 623
16.3.7 Panoul de Digitizare Avansată . . . 625
16.3.8 The Processing in-place layer modifier . . . 633
16.4 Connecting and Editing Data Across Layers . . . 635
16.4.1 Joining features between two layers . . . 635
16.4.2 Setting relations between multiple layers . . . 637

17 Lucrul cu Datele Raster **651**
17.1 Dialogul Proprietăților Rasterului . . . 651
17.1.1 Information Properties . . . 652
17.1.2 Proprietățile Sursei . . . 652
17.1.3 Symbology Properties . . . 653
17.1.4 Transparency Properties . . . 663
17.1.5 Histogram Properties . . . 664
17.1.6 Rendering Properties . . . 665
17.1.7 Proprietăți Temporale . . . 666
17.1.8 Proprietățile Piramidelor . . . 667
17.1.9 Elevation Properties . . . 669
17.1.10 Proprietăți Metadate . . . 670
17.1.11 Proprietățile Legendei . . . 670
17.1.12 Display Properties . . . 671
17.1.13 Proprietățile Serverului QGIS . . . 672
17.1.14 Identify raster cells . . . 673
17.2 Analiza Raster . . . 674
17.2.1 Calculatorul Raster . . . 674
17.3 Georeferencer . . . 677
17.3.1 Usual procedure . . . 677

18 Working with Mesh Data **683**
18.1 What's a mesh? . . . 683
18.2 Supported formats . . . 685
18.3 Mesh Dataset Properties . . . 686
18.3.1 Information Properties . . . 686
18.3.2 Source Properties . . . 687
18.3.3 Symbology Properties . . . 688
18.3.4 3D View Properties . . . 693

18.3.5 Rendering Properties . . . 695
18.3.6 Temporal Properties . . . 695
18.3.7 Elevation Properties . . . 696
18.3.8 Metadata Properties . . . 697
18.4 Editing a mesh layer . . . 697
18.4.1 Overview of the mesh digitizing tools . . . 698
18.4.2 Exploring the Z value assignment logic . . . 698
18.4.3 Selecting mesh elements . . . 699
18.4.4 Modifying mesh elements . . . 700
18.4.5 Reindexing meshes . . . 704
18.5 Mesh Calculator . . . 705

19 Working with Vector Tiles **707**
19.1 What are Vector Tiles? . . . 707
19.2 Supported Formats . . . 708
19.3 Vector Tiles Dataset Properties . . . 709
19.3.1 Information Properties . . . 709
19.3.2 Source Properties . . . 709
19.3.3 Symbology and Labels Properties . . . 710
19.3.4 Rendering Properties . . . 711
19.3.5 Metadata Properties . . . 712

20 Working with Point Clouds **713**
20.1 Introduction to Point Clouds . . . 713
20.2 Point Clouds Properties . . . 714
20.2.1 Information Properties . . . 714
20.2.2 Proprietățile Sursei . . . 715
20.2.3 Symbology Properties . . . 718
20.2.4 3D View Properties . . . 723
20.2.5 Rendering Properties . . . 725
20.2.6 Elevation Properties . . . 725
20.2.7 Proprietăți Metadate . . . 726
20.2.8 Statistics Properties . . . 726
20.3 Virtual point cloud . . . 727

21 Working with 3D Tiles **731**
21.1 What are 3D Tiles? . . . 731
21.2 Data Sources . . . 732
21.2.1 Using data from Cesium ion . . . 732
21.2.2 Using Google 3D data . . . 733
21.3 3D Tiles Properties . . . 733
21.3.1 Information Properties . . . 733
21.3.2 Source Properties . . . 734
21.3.3 Symbology Properties . . . 734
21.3.4 3D View Properties . . . 735
21.3.5 Rendering Properties . . . 736
21.3.6 Elevation Properties . . . 736
21.3.7 Metadata Properties . . . 736

22 Așezarea în pagină a hărților **737**
22.1 Generalități despre Compozitorul de Hărți . . . 737
22.1.1 Sample Session for beginners . . . 737
22.1.2 Managerul de Compoziții . . . 738
22.1.3 Menus, tools and panels of the print layout . . . 739
22.2 Elementele Compoziției . . . 753
22.2.1 Layout Items Common Options . . . 753
22.2.2 Elementul Hartă . . . 758
22.2.3 The 3D Map Item . . . 768
22.2.4 Elementul Etichetă . . . 769

22.2.5 Elementul Legendă . . . 773
22.2.6 Elementul Scării Grafice . . . 781
22.2.7 The Table Items . . . 784
22.2.8 The Marker, Picture and North Arrow Items . . . 793
22.2.9 The Elevation Profile Item . . . 797
22.2.10 Elementul HTML de tip cadru . . . 798
22.2.11 The Shape Items . . . 801
22.3 Generarea unui Rezultat . . . 804
22.3.1 Setări de export . . . 804
22.3.2 Exportare ca imagine . . . 805
22.3.3 Exportare ca SVG . . . 806
22.3.4 Exportare ca PDF . . . 807
22.3.5 Generarea unui Atlas . . . 808
22.4 Generarea unui Raport . . . 814
22.4.1 What is it? . . . 814
22.4.2 Get started . . . 814
22.4.3 Layout Report Workspace . . . 815
22.4.4 Setări de export . . . 830

23 Working with OGC / ISO protocols **831**
23.1 Client WMS/WMTS . . . 832
23.1.1 Privire de ansamblu asupra suportului WMS . . . 832
23.1.2 Privire de ansamblu asupra suportului WMTS . . . 832
23.1.3 Selectarea serverelor WMS/WMTS . . . 833
23.1.4 Încărcarea Straturilor WMS/WMTS . . . 835
23.1.5 Seturi de plăcuțe . . . 837
23.1.6 Folosirea instrumentului de identificare . . . 837
23.1.7 Viewing Properties . . . 838
23.1.8 Show WMS legend graphic in table of contents and layout . . . 842
23.2 Client WCS . . . 842
23.3 Client WFS și WFS-T . . . 842

24 Lucrul cu datele GPS **847**
24.1 Introducing GNSS/GPS Data . . . 847
24.1.1 Ce este GPS? . . . 847
24.1.2 Transferring or loading GPS data . . . 847
24.2 Urmărirea live a GPS-ului . . . 850
24.2.1 GPS toolbar . . . 850
24.2.2 GPS Information Panel . . . 851
24.2.3 Conectaare la un GPS Bluetooth pentru urmărirea în direct a poziției . . . 853
24.2.4 Examples of GPS devices connection in QGIS . . . 854

25 Sistem de Autentificare **857**
25.1 Prezentare Generală a Sistemului de Autentificare . . . 857
25.1.1 Baza de date de autentificare . . . 858
25.1.2 Parola master . . . 858
25.1.3 Configurații de Autentificare . . . 859
25.1.4 Metoda de Autentificare . . . 862
25.1.5 Utilitare pentru Parola Master și Configurările de Autentificare . . . 866
25.1.6 Folosirea configurațiilor de autentificare . . . 867
25.1.7 Legături Python . . . 867
25.2 Fluxul de Autentificare a Utilizatorului . . . 868
25.2.1 Autentificarea HTTP(S) . . . 868
25.2.2 Autentificarea bazei de date . . . 869
25.2.3 Autentificarea PKI . . . 870
25.2.4 Gestionarea straturilor eronate . . . 877
25.2.5 Schimbarea ID-ului de configurare a autentificării . . . 878
25.2.6 Suport pentru serverul QGIS . . . 879
25.2.7 Excepții de server SSL . . . 879

25.3 Avertismente de Securitate . . . 882
25.3.1 Restricții . . . 883

26 Integrarea GRASS GIS **885**
26.1 Setul de date demonstrative . . . 885
26.2 Încărcarea straturilor raster și vectoriale GRASS . . . 885
26.3 Importing data into a GRASS LOCATION via drag and drop . . . 886
26.4 Managing GRASS data in QGIS Browser . . . 886
26.5 Opțiuni GRASS . . . 886
26.6 Startarea plugin-ului GRASS . . . 886
26.7 Deschiderea Setului de hărți GRASS . . . 887
26.8 GRASS LOCATION și MAPSET . . . 887
26.9 Importați datele într-o LOCAȚIE GRASS . . . 887
26.9.1 Crearea unei noi LOCAȚII GRASS . . . 888
26.9.2 Adăugarea unui nou MAPSET . . . 889
26.10 Modelul de date vectoriale GRASS . . . 890
26.11 Crearea unui nou strat vectorial GRASS . . . 890
26.12 Digitizarea și editarea unui strat vectorial GRASS . . . 891
26.13 Regiunea instrumentelor GRASS . . . 893
26.14 Bara de instrumente GRASS . . . 893
26.14.1 Lucrul cu modulele GRASS . . . 894
26.14.2 Exemple de module GRASS . . . 897
26.14.3 Personalizarea Barei de Instrumente GRASS . . . 900

27 Cadrul de procesare al QGIS **903**
27.1 Introducere . . . 903
27.2 Configuring the Processing Framework . . . 906
27.2.1 General . . . 906
27.2.2 Menus . . . 908
27.2.3 Models and Scripts . . . 908
27.2.4 Providers . . . 908
27.3 The Toolbox . . . 908
27.3.1 Dialogul algoritmului . . . 910
27.3.2 Obiecte de date generate de algoritmi . . . 917
27.4 Managerul istoricului . . . 918
27.4.1 Procesarea istoricului . . . 918
27.4.2 Jurnalul de Procesare . . . 919
27.5 The model designer . . . 919
27.5.1 The model designer interface . . . 920
27.5.2 Creating a model . . . 922
27.5.3 Salvarea și încărcarea modelelor . . . 932
27.5.4 Editarea unui model . . . 932
27.6 Interfața de prelucrare în serie . . . 934
27.6.1 Introducere . . . 934
27.6.2 Tabela parametrilor . . . 935
27.6.3 Completarea tabelei cu parametri . . . 936
27.6.4 Executare ca proces în serie . . . 937
27.7 Utilizarea algoritmilor de procesare din consolă . . . 937
27.7.1 Apelarea algoritmilor din consola Python . . . 938
27.7.2 Crearea scripturilor și execuția lor din caseta de instrumente . . . 942
27.7.3 Script de interceptare a pre- și post-execuției . . . 945
27.8 Using processing from the command line . . . 945
27.9 Writing new Processing algorithms as Python scripts . . . 947
27.9.1 Extending QgsProcessingAlgorithm . . . 947
27.9.2 The @alg decorator . . . 952
27.9.3 Input and output types for Processing Algorithms . . . 953
27.9.4 Handing algorithm output . . . 956
27.9.5 Comunicarea cu utilizatorul . . . 956

27.9.6 Documentarea scripturilor . . . 956
27.9.7 Flags . . . 956
27.9.8 Cele mai bune practici pentru scrierea algoritmilor pentru script . . . 957
27.10 Configurarea Aplicațiilor Externe . . . 957
27.10.1 O notă pentru utilizatorii de Windows . . . 957
27.10.2 O notă privind formatele de fișiere . . . 957
27.10.3 O notă privind selecțiile stratului vectorial . . . 958
27.10.4 Using third-party Providers . . . 958

28 Furnizorii de prelucrare și algoritmi 969
28.1 Furnizorul algoritmului QGIS . . . 969
28.1.1 3D Tiles . . . 969
28.1.2 Cartography . . . 971
28.1.3 Database . . . 992
28.1.4 File tools . . . 1000
28.1.5 GPS . . . 1001
28.1.6 Interpolarea . . . 1004
28.1.7 Layer tools . . . 1016
28.1.8 Suprafață poliedrică . . . 1019
28.1.9 Instrumentele modelatorului . . . 1031
28.1.10 Network analysis . . . 1041
28.1.11 Plots . . . 1052
28.1.12 Point Cloud Conversion . . . 1059
28.1.13 Point Cloud Data Management . . . 1065
28.1.14 Point Cloud Extraction . . . 1078
28.1.15 Analizarea rasterelor . . . 1083
28.1.16 Crearea Rasterelor . . . 1138
28.1.17 Raster terrain analysis . . . 1156
28.1.18 Raster tools . . . 1167
28.1.19 Vector analysis . . . 1178
28.1.20 Vector creation . . . 1203
28.1.21 Vectori, în general . . . 1228
28.1.22 Vector geometry . . . 1266
28.1.23 Vector overlay . . . 1380
28.1.24 Vector selection . . . 1398
28.1.25 Tabela Vectorială . . . 1415
28.1.26 Plăcuțe Vectoriale . . . 1431
28.2 Furnizorul algoritmului GDAL . . . 1435
28.2.1 Analizarea rasterelor . . . 1435
28.2.2 Raster conversion . . . 1465
28.2.3 Raster extraction . . . 1473
28.2.4 Raster miscellaneous . . . 1482
28.2.5 Proiecții raster . . . 1503
28.2.6 Vector conversion . . . 1508
28.2.7 Vector geoprocessing . . . 1514
28.2.8 Vector miscellaneous . . . 1523
28.3 Furnizorul aplicațiilor OTB . . . 1533

29 Plugin-uri 1535
29.1 Plugin-uri QGIS . . . 1535
29.1.1 Plugin-uri de bază și externe . . . 1535
29.1.2 Dialogul Plugin-urilor . . . 1536
29.2 Using QGIS Core Plugins . . . 1539
29.2.1 DB Manager Plugin . . . 1539
29.2.2 Geometry Checker Plugin . . . 1542
29.2.3 MetaSearch Catalog Client . . . 1546
29.2.4 Offline Editing Plugin . . . 1553
29.2.5 Topology Checker Plugin . . . 1555

29.3 Consola Python din QGIS . . . 1557
29.3.1 Consola interactivă . . . 1558
29.3.2 Editorul de Cod . . . 1560

30 Asistență și Ajutor **1563**
30.1 Liste de discuții . . . 1563
30.1.1 Utilizatori QGIS . . . 1563
30.1.2 Dezvoltatori QGIS . . . 1563
30.1.3 Echipa Comunității QGIS . . . 1563
30.1.4 Traduceri QGIS . . . 1564
30.1.5 Comitetul de Dirijare a Proiectului QGIS (PSC) . . . 1564
30.1.6 Grupuri de utilizatori QGIS . . . 1564
30.2 Matrix / IRC . . . 1564
30.3 Suport comercial . . . 1564
30.4 BugTracker . . . 1565
30.5 Blog . . . 1565
30.6 Plugin-uri . . . 1565
30.7 Wiki . . . 1565

31 Contribuții **1567**
31.1 Autori . . . 1567
31.2 Traducători . . . 1568
31.3 Statistica traducerilor . . . 1569

32 Documente adiționale **1571**
32.1 Anexa A: Licența Publică Generală GNU . . . 1571
32.2 Anexa B: Licența GNU pentru Documentația Liberă . . . 1575
32.3 Anexa C: Formatele Fișierelor QGIS . . . 1580
32.3.1 QGS/QGZ - Formatul Fișierelor Proiect . . . 1580
32.3.2 QLR - Fișier cu Definiții ale Straturilor . . . 1582
32.3.3 QML - Formatul Fișierelor de Stilizare . . . 1583
32.4 Anexa D: Sintaxa scrípturilor R din QGIS . . . 1584
32.4.1 Intrări . . . 1585
32.4.2 Rezultat . . . 1585
32.4.3 Syntax Summary for QGIS R scripts . . . 1585
32.4.4 Exemple . . . 1587
32.5 Anexa E: Conexiuni de Rețea ale Aplicației QGIS . . . 1589

33 Literatură și Referințe Web **1591**

CAPITOLUL 1

Preambul

Acesta este ghidul de utilizare pentru softul QGIS, specializat în sistemele de informații geografice (GIS). QGIS este distribuit sub licența publică generală GNU. Mai multe informații sunt disponibile pe pagina de pornire a QGIS, https://www.qgis.org.

Conținutul acestui document a fost scris și verificat de către autori și editori. Cu toate acestea, greșelile sunt posibile.

Prin urmare, atât autorii cât și editorii nu își asumă nicio responsabilitate pentru erorile din acest document și posibilele consecințe ale acestora. Vă încurajăm să raportați eventualele greșeli.

Acest document a fost tipărit cu ajutorul reStructuredText. Este disponibil sub formă de cod sursă reST pe **`github<https://github.com/qgis/QGIS-Documentation>`_**, și online la https://www.qgis.org/en/docs/, în formatele HTML și PDF. Versiunile traduse ale acestui document pot fi răsfoite și descărcate și din zona de documentare a proiectului QGIS.

Pentru mai multe informații despre contribuția la acest document și despre traducere, vă rugăm să vizitați https://qgis.org/en/site/getinvolved/index.html.

Link-urile din acest document

Acest document conține legături interne și externe. Un clic pe un link intern produce deplasarea în cadrul unui document, în timp ce un clic pe un link extern deschide o adresă de internet.

Autorii și Editorii Documentației

Lista persoanelor care au contribuit la redactarea, revizuirea și traducerea următoarei documentații este disponibilă la adresa *Contribuții*.

Drepturi de autor (c) 2004 - 2020 Echipa de dezvoltare QGIS

Internet: https://www.qgis.org

Licența acestui document

Se acordă permisiunea de copiere, distribuire și/sau de modificare a acestui document sub termenii GNU Free Documentation License, versiunea 1.3 sau orice versiune ulterioară, publicată de către Free Software Foundation; fără Secțiunile Fixe și fără Textele Coperților. O copie a acestei licențe este inclusă în Anexa *Anexa B: Licența GNU pentru Documentația Liberă*.

1.1 Ce este nou în QGIS 3.34

Această versiune de QGIS include sute de remedieri de erori și multe funcții și îmbunătățiri, comparativ cu QGIS 3.28. Pentru o listă a noilor funcționalități, parcurgeți jurnalele modificărilor de la https://qgis.org/en/site/forusers/visualchangelogs.html.

CAPITOLUL 2

Cuvânt înainte

Bine ați venit în lumea minunată a Sistemelor de Informații Geografice (GIS)!

QGIS is an Open Source Geographic Information System. The project was born in May 2002 and was established as a project on SourceForge in June the same year. We have worked hard to make GIS software (which is traditionally expensive proprietary software) available to anyone with access to a personal computer.

QGIS currently runs on most Unix platforms, Windows, and macOS. QGIS is developed using the Qt toolkit (https://www.qt.io) and C++. This means that QGIS feels snappy and has a pleasing, easy-to-use graphical user interface (GUI). There are also independently created applications that allow you to take QGIS into the field. These applications can run on Android and iOS.

QGIS își propune să fie un GIS ușor de utilizat, oferind funcții și caracteristici comune. Scopul inițial al proiectului a fost de a crea un vizualizator de date GIS. La acest moment QGIS a atins punctul de evoluție în care poate folosit pentru nevoile zilnice de vizualizare a datelor GIS, pentru captura datelor, analiza GIS avansată și pentru prezentări sub formă de hărți, atlase și rapoarte sofisticate. QGIS acceptă o multitudine de formate de date raster și vectoriale, având o arhitectură pentru pluginuri care permite adăugarea facilă a noilor formate.

QGIS este lansat sub licența publică generală GNU (GPL). Dezvoltarea QGIS sub această licență vă dă dreptul de a inspecta și modifica codul sursă și vă garantează că dvs., utilizatorul nostru, veți avea întotdeauna acces la un program GIS care este gratuit și care poate fi modificat în mod liber. Ar fi trebuit să primiți o copie completă a licenței în cadrul copiei dvs. de QGIS, care se poate găsi și în Anexă *Anexa A: Licența Publică Generală GNU*.

Sfat: Documentație la-zi

Cea mai recentă versiune a acestui document se găsește mereu în zona documentației a site-ului web QGIS, la https://www.qgis.org/en/docs/.

CAPITOLUL 3

Convenții

Această secțiune descrie stilurile uniforme care vor fi folosite pe parcursul acestui manual.

3.1 Convențiile GUI

Stilurile convenite pentru GUI sunt destinate imitării aspectului unui GUI. În general, un stil va reflecta aspectul non-hover, astfel încât un utilizator poate scana vizual GUI-ul, pentru a găsi ceva care arată similar instrucțiunilor din manual.

- Opțiuni de Meniu: *Strat* ► *Adăugare Strat Raster* sau *Setări* ► *Bare de instrumente* ► *Digitizare*
- Instrument: Adaugă un Strat Raster
- Buton : *Salvează ca Implicit*
- Titlul Casetei de Dialog: *Proprietățile Stratului*
- Filă: *Generalități*
- Casetă de bifare: *Rendare*
- Buton Radio: *Postgis SRID* *EPSG ID*
- Selectare număr:
- Selectare șir:
- Alegere fișier: …
- Selectare culoare:
- Cursor:
- Introducere Text:

O umbră indică o componentă GUI pe care se poate face clic.

3.2 Convenții pentru Text și Tastatură

Acest manual include, de asemenea, stiluri pentru text, comenzi pentru taste și codificări pentru indicarea diferitelor entități, cum ar fi clasele sau metodele. Aceste stiluri nu corespund aspectului efectiv al unui text sau al unei codificări din QGIS.

- Hiperlegături: https://qgis.org
- Combinații de Taste: Apăsarea combinației `Ctrl+B` constă în ținerea apăsată a tastei Ctrl, urmată de apăsarea tastei B.
- Numele unui Fișier: `lakes.shp`
- Numele unei Clase: **NewLayer**
- Metodă: *classFactory*
- Server: *myhost.de*
- Text Utilizator: `qgis --help`

Liniile de cod sunt indicate printr-un font cu lățime fixă:

```
PROJCS["NAD_1927_Albers",
  GEOGCS["GCS_North_American_1927",
```

3.3 Instrucțiuni specifice platformelor

Secvențele GUI și textele scurte pot fi formatate în modul următor: Clic pe *Fișier* ► *QGIS* ► *Închidere pentru a ieși din QGIS*. Rețineți că pe platformele Linux, Unix și Windows ar trebui să faceți clic mai întâi pe meniul File, apoi pe Quit, în timp ce pe platformele Macintosh OS X ar trebui să faceți clic mai întâi pe meniul QGIS și ulterior pe Quit.

Cantități mai mari de text pot fi formatate sub formă de listă:

- Faceți asta
- Procedați așa
- Sau așa

sau ca paragrafe:

Faceți acest lucru și asta și asta. Apoi faceți acest lucru și asta și asta, și asta și asta și asta, și asta și asta și asta.

Faceți asta. Apoi asta și asta și asta, și asta și asta, și asta și asta, și asta și asta.

Capturile de ecran care apar în ghidul utilizatorului au fost create pe diverse platforme.

CAPITOLUL 4

Funcțiuni

QGIS oferă o multitudine de funcții GIS, prin intermediul funcțiilor de bază și al pluginurilor. Bara de localizare facilitează căutarea de funcții, seturi de date și multe altele.

Un scurt rezumat pentru șase categorii generale de funcții și pluginuri este prezentat mai jos, urmat de informații despre consola integrată Python.

4.1 Vizualizarea datelor

Puteți vizualiza combinații de date vectoriale și raster (în 2D sau 3D) în diferite formate și proiecții, fără a conversia într-un format intern sau comun. Formatele acceptate sunt:

- Tabelele și vederile activate spațial cu ajutorul PostGIS, SpatiaLite și MS SQL Spatial, Oracle Spatial, formatele vectoriale acceptate de biblioteca OGR instalată, inclusiv GeoPackage, ESRI Shapefile, MapInfo, SDTS, GML și multe altele. Consultați secțiunea *Lucrul cu Datele Vectoriale*.
- Formatele raster și imaginile acceptate de biblioteca GDAL (Geospatial Data Abstraction Library), cum ar fi GeoTIFF, ERDAS IMG, ArcInfo ASCII GRID, JPEG, PNG și multe altele. Parcurgeți secțiunea *Lucrul cu Datele Raster*.
- Suprafețele poliedrice (sunt acceptate TIN-urile și grilele obișnuite). Consultați *Working with Mesh Data*.
- Plăcuțe vectoriale
- Date raster și vectoriale GRASS din bazele de date GRASS (locație/set de hărți). Parcurgeți secțiunea *Integrarea GRASS GIS*.
- Datele spațiale online au servit ca Servicii Web OGC, inclusiv WMS, WMTS, WCS, WFS și WFS-T. Mai multe informații în *Working with OGC / ISO protocols*.

  Infrastructura de autentificare QGIS vă ajută să gestionați utilizatorul/parola, certificatele și cheile serviciilor web și a altor resurse.
- Foi de calcul (ODS / XLSX)

Datele temporale sunt acceptate.

4.2 Explorarea datelor și compunerea hărților

Puteți compune hărți, apoi să explorați în mod interactiv datele spațiale prin intermediul interfeței grafice prietenoase. Printre cele mai multe instrumente utile disponibile în GUI se includ:

- Navigatorul QGIS
- Reproiectarea Din-Zbor
- Generarea de hărți 2D și 3D
- DB Manager
- Aspect Pagină Imprimată
- Raport
- Panoul de Vizualizare
- Semne de carte Spațiale
- Instrumente de adnotare
- Identificarea/selectarea entităților
- Editarea/vizualizarea/căutarea atributelor
- Etichetarea entităților definite de date
- Instrumente de simbologie definită cu ajutorul datelor, pentru rastere și vectori
- Compoziția hărților pentru atlas, folosind straturile cu graticule
- Săgeata nordului, scara grafică și eticheta drepturilor de autor pentru hărți
- Suport pentru salvarea și restaurarea proiectelor

4.3 Crearea, editarea, gestionarea și exportul datelor

Puteți crea, edita, gestiona și exporta în diverse formate straturile raster sau vectoriale. QGIS oferă următoarele:

- Instrumente de digitalizare vectorială
- Abilitatea de a crea și edita mai multe formate de fișiere și straturi vectoriale GRASS
- Instrument de georeferențiere dedicat geocodării vectorilor și a imaginilor
- Instrumente GPS dedicate importării și exportării formatului GPX și pentru a converti alte formate GPS în GPX, sau pentru a le descărca/încărca direct pe o unitate GPS (în Linux, usb: a fost adăugat listei de dispozitive GPS)
- Suport pentru salvarea și editarea datelor OpenStreetMap
- Abilitatea de a crea tabele de baze de date spațiale din fișiere, folosind pluginul DB Manager
- Gestionarea îmbunătățită a tabelelor bazei de date spațiale
- Instrumente pentru gestionarea tabelelor cu atribute vectoriale
- Opțiunea de a salva capturile de ecran ca imagini georeferențiate
- Instrumentul DXF-Export cu capacități sporite pentru exportul stilurilor și a plugin-urilor pentru a efectua funcții similare CAD

4.4 Analiza datelor

Puteți efectua analize de date spațiale pe baze de date spațiale și pe alte formate acceptate de OGR. QGIS oferă în prezent analiză vectorială, analiză raster, eșantionare, geoprocesare, instrumente de gestionare a geometriei și a bazelor de date. De asemenea, puteți utiliza instrumentele GRASS integrate, ceea ce constă în funcționalitatea completă a peste 400 de module GRASS (vezi secțiunea *Integrarea GRASS GIS*). De asemenea, se poate folosi pluginul Processing, care oferă un cadru puternic de analiză geospațială, fie făcând apel la algoritmi QGIS nativi, fie la cei terți precum GDAL, SAGA, GRASS, OTB, R și alții (consultați secțiunea :ref:`sec_processing_intro `). Toate funcțiile de analiză rulează în fundal, permițându-vă să vă continuați munca.

Modelatorul grafic vă permite să combinați / înlănțuiți funcții într-un flux de lucru complet, într-un mediu grafic intuitiv.

4.5 Publicarea hărților de pe Internet

QGIS poate fi folosit ca un client WMS, WMTS, WMS-C, WFS, OAPIF și WFS-T (v. secțiunea *Working with OGC / ISO protocols*), iar QGIS Server (vezi QGIS-Server-manual) vă permite să publicați pe Internet propriile date, prin protocoalele WMS, WCS, WFS și OAPIF, folosind un server web.

4.6 Extinderea funcționalității QGIS prin intermediul plugin-urilor

QGIS poate fi adaptat nevoilor dumneavoastră speciale cu arhitectură de plugin-uri extensibile și biblioteci care pot fi folosite pentru a crea plugin-uri. Puteți crea chiar noi aplicații cu C++ sau Python!

4.6.1 Plugin-uri de bază

Plugin-urile de bază includ:

1. DB Manager (editarea și vizualizarea straturilor și a tabelelor din/în bazele de date; executarea interogărilor SQL)
2. Geometry Checker (identificarea erorilor din geometrii)
3. Georeferencer GDAL (adăugarea în rastere a informațiilor despre proiecție cu ajutorul GDAL)
4. GPS Tools (încărcarea și importarea datelor GPS)
5. GRASS (integrare cu GRASS GIS)
6. MetaSearch Catalog Client (interacționează cu serviciile de catalogare a metadatelor care acceptă standardul OGC Catalog Service for the Web (CSW))
7. Offline Editing (permite editarea offline și sincronizarea cu bazele de date)
8. Processing (cadrul de procesare a datelor spațiale pentru QGIS)
9. Topology Checker (face posibilă identificarea erorilor topologice în straturile vectoriale)

4.6.2 Plugin-urile Externe Python

QGIS oferă un număr tot mai mare de plugin-uri externe, scrise în Python, care sunt furnizate de către comunitate. Aceste plugin-uri rezidă în depozitul oficial de plugin-uri, și pot fi instalate cu ușurință cu ajutorul Instalatorului de Plugin-uri Python. Parcurgeți secțiunea *Dialogul Plugin-urilor*.

4.7 Consola Python

Pentru scripting puteți utiliza consola Python integrată accesând: *Plugins* ► *Python Console*. Consola se deschide ca o fereastră utilitară non-modală. Pentru interacțiunea cu mediul QGIS, există variabila `qgis.utils.iface`, care este o instanță a `QgisInterface`. Această interfață oferă acces la canevasul hărții, meniuri, bare de instrumente și alte părți ale aplicației QGIS. Puteți crea un script, apoi să-l plasați în fereastra QGIS, după care va fi executat în mod automat.

Pentru mai multe informații despre lucrul cu consola Python și despre programarea pluginurilor și a aplicațiilor QGIS, vă rugăm să consultați *Consola Python din QGIS* și PyQGIS-Developer-Cookbook.

4.8 Probleme Cunoscute

4.8.1 Limitarea numărului de fișiere deschise

Dacă deschideți un proiect QGIS mare și sunteți sigur că toate straturile sunt valide, dar unele straturi sunt marcate ca fiind eronate, puteți gestiona, probabil, această problemă. Linux (și alte sisteme de operare, de asemenea) are o limită pentru fișierele deschise de către un proces. Limitele resurselor sunt per-proces și sunt moștenite. Comanda internă `ulimit`, modifică limitele numai pentru procesul curent; noua limită va fi moștenită de către orice proces copil.

Puteți vedea toate informațiile ulimit curente, tastând:

```
$ ulimit -aS
```

Puteți vedea numărul permis de fișiere deschise per proces, în mod curent, folosind următoarea comandă într-o consolă:

```
$ ulimit -Sn
```

Pentru a modifica limitele unei **sesiuni existente** puteți folosi ceva de genul:

```
$ ulimit -Sn #number_of_allowed_open_files
$ ulimit -Sn
$ qgis
```

Alternativ, puteți utiliza utilitarul mai nou `prlimit`. Mai multe informații la: https://manpages.ubuntu.com/manpages/latest/man1/prlimit.1.html

Pentru a remedia definitiv problema

Pe majoritatea sistemelor Linux, limitele resurselor sunt stabilite de modulul `pam_limits`, în conformitate cu setările conținute în `/etc/security/limits.conf` sau `/etc/security/limits.d/*.conf`. Ar trebui să puteți edita acele fișiere dacă aveți privilegiul de root (de asemenea, prin intermediul sudo), dar va trebui să vă conectați din nou pentru ca schimbările să aibă efect.

Mai multe informații:

https://www.cyberciti.biz/faq/linux-increase-the-maximum-number-of-open-files/ https://linuxaria.com/article/open-files-in-linux

CAPITOLUL 5

Noțiuni de bază

Acest capitol oferă o privire de ansamblu rapidă despre instalarea QGIS, descărcarea datelor QGIS oferite ca exemplu și rularea unei prime sesiuni simple, de vizualizare a datelor raster și vectoriale.

5.1 Instalarea QGIS

Proiectul QGIS oferă diferite moduri de instalare a aplicației QGIS, în funcție de platforma dvs.

5.1.1 Instalarea pornind de la pachetele binare

Pentru MS Windows și macOS sunt disponibile programele de instalare standard. De asemenea, sunt furnizate pachetele binare (rpm și deb), precum și depozitele software pentru multe variante de GNU/Linux.

Pentru mai multe informații și instrucțiuni pentru sistemul dvs. de operare accesați https://download.qgis.org.

5.1.2 Instalarea pornind de la sursă

Dacă trebuie să compilați QGIS de la sursă, vă rugăm să consultați instrucțiunile de instalare. Acestea sunt distribuite alături de codul sursă QGIS într-un fișier numit `INSTALL`. Le puteți găsi și online la https://github.com/qgis/QGIS/blob/release-3_34/INSTALL.md.

Dacă doriți să compilați o anumită versiune și nu versiunea în dezvoltare, ar trebui să înlocuiți `master` cu ramura de lansare (de obicei în formularul `release-X_Y`) în linkul menționat mai sus (instrucțiunile de instalare pot diferă).

5.1.3 Instalare pe suport extern

Este posibilă instalarea QGIS (împreună cu toate pluginurile și setările) pe o unitate flash. Acest lucru se realizează prin definirea unei opțiuni *–profiles-path* care înlocuiește calea pentru *profilul utilizatorului* implicit și forțează **QSettings** să folosească și acest director. Consultați secțiunea *System Settings* pentru informații suplimentare.

5.1.4 Descărcarea datelor demonstrative

Acest ghid de utilizare conține exemple bazate pe setul de date demonstrativ din QGIS (denumit și `Setul de date Alaska`). Descărcați datele respective de la adresa https://github.com/qgis/QGIS-Sample-Data/archive/master.zip și dezarhivați arhiva în orice locație convenabilă de pe sistemul dvs.

Setul de date Alaska cuprinde toate datele GIS care stau la baza exemplelor și a capturilor de ecran din ghidul utilizatorului. Este inclusă, de asemenea, o mică bază de date GRASS. Proiecția datelor eșantion pentru setul de date QGIS este Alaska Albers cu Suprafețe Egale, unitățile de măsura fiind picioarele. Codul EPSG este 2964.

```
PROJCS["Albers Equal Area",
GEOGCS["NAD27",
DATUM["North_American_Datum_1927",
SPHEROID["Clarke 1866",6378206.4,294.978698213898,
AUTHORITY["EPSG","7008"]],
TOWGS84[-3,142,183,0,0,0,0],
AUTHORITY["EPSG","6267"]],
PRIMEM["Greenwich",0,
AUTHORITY["EPSG","8901"]],
UNIT["degree",0.0174532925199433,
AUTHORITY["EPSG","9108"]],
AUTHORITY["EPSG","4267"]],
PROJECTION["Albers_Conic_Equal_Area"],
PARAMETER["standard_parallel_1",55],
PARAMETER["standard_parallel_2",65],
PARAMETER["latitude_of_center",50],
PARAMETER["longitude_of_center",-154],
PARAMETER["false_easting",0],
PARAMETER["false_northing",0],
UNIT["us_survey_feet",0.3048006096012192]]
```

Dacă intenționați să utilizați QGIS ca interfață grafică pentru GRASS, puteți găsi o selecție de locații cu date demonstrative (de ex.: Spearfish sau South Dakota) pe site-ul oficial GRASS GIS, https://grass.osgeo.org/download/data/.

5.2 Pornirea și oprirea QGIS

QGIS se poate lansa în execuție ca oricare altă aplicație de pe computer. Aceasta înseamnă că puteți porni QGIS prin:

- accesarea meniului Aplicații, a meniului Start sau a Dock-ului
- dublu clic pe pictograma din folderul Aplicațiilor sau pe o scurtătură de pe ecran.
- dublu clic pe un fișier de proiect QGIS existent (cu extensia `.qgz` sau `.qgs`). Rețineți că acest lucru va deschide și proiectul.
- introducerea textului `qgis` într-o linie de comandă (presupunând că QGIS este adăugat la PATH sau că vă aflați în folderul de instalare)

Pentru a ieși din QGIS, utilizați:

- opțiunea meniului *Proiect* ► *Închidere QGIS*, sau folosiți combinația de taste `Ctrl+Q`.

- *QGIS* ► *Închidere QGIS*, sau folosirea combinației de taste `Cmd+Q`.
- sau prin utilizarea butonului din colțul din dreapta sus al interfeței principale a aplicației.

5.3 Sesiune de Lucru: Încărcarea straturilor raster și a celor vectoriale

Având *QGIS instalat* și un *set amplu cu date de test* la îndemână, vom demara o primă sesiune de lucru. În acest exemplu, vom vizualiza un strat raster și un strat vectorial. Vom folosi:

- stratul raster `landcover` (`qgis_sample_data/raster/landcover.img`)
- și stratul vectorial `lakes` (`qgis_sample_data/gml/lakes.gml`)

`qgis_sample_data` reprezintă calea către setul de date dezarhivat.

1. Porniți QGIS așa cum se recomandă în *Pornirea și oprirea QGIS*.
2. Datele de lucru sunt în proiecția `Albers Equal Area`, așa că vom seta CRS-ul proiectului în consecință:
    1. Faceți clic pe butonul de Selectare a proiecției din colțul din dreapta jos al interfeței QGIS. Dialogul proprietăților proiectului se deschide cu fila *CRS* activă.
    2. Tastați `2964` în câmpul de text *Filtrare*.
    3. Selectați rândul cu numele CRS `NAD27 / Alaska Albers`.

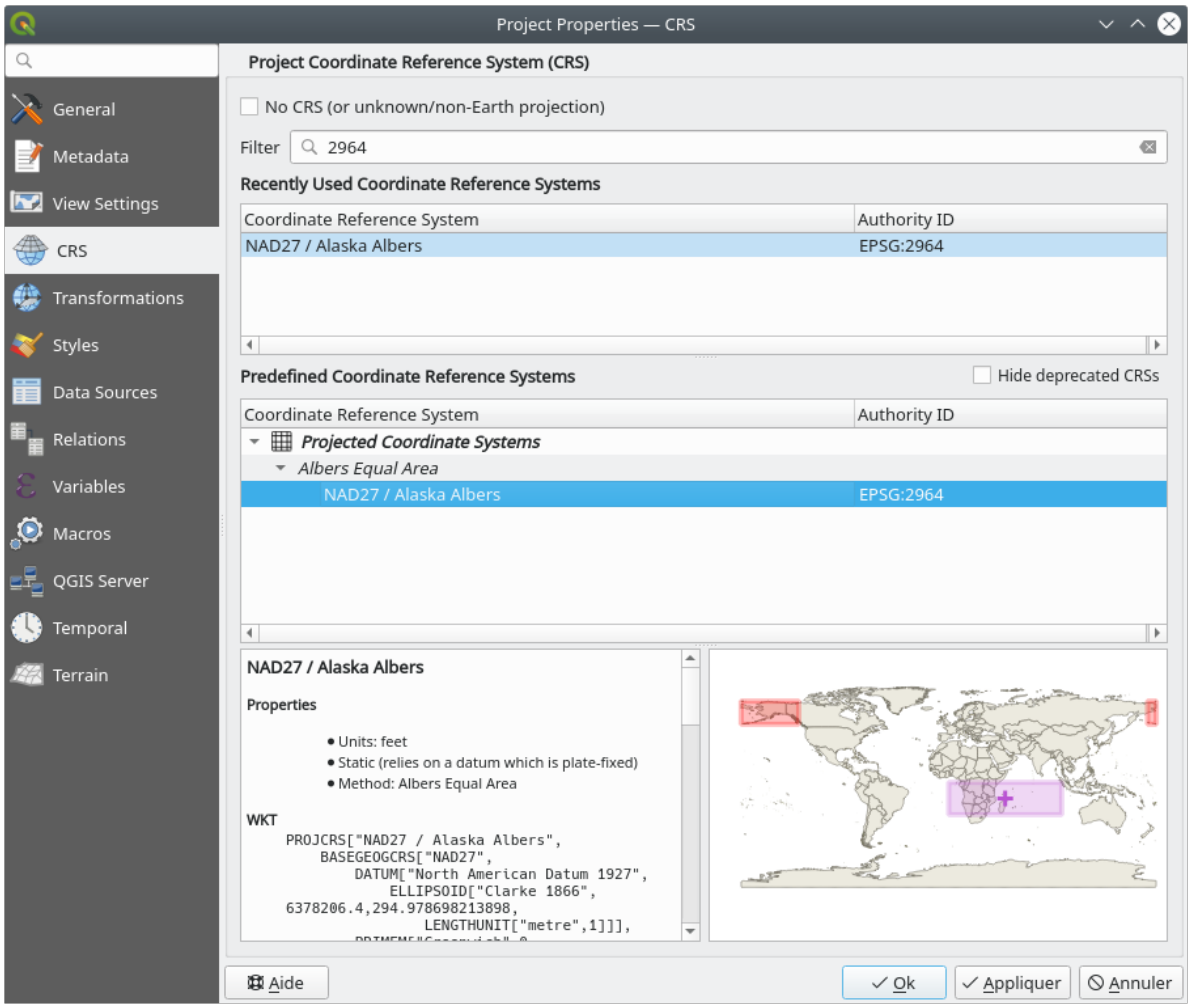


Fig. 5.1: Selectați Sistemul de Referință și Coordonate al datelor

4. Apăsați pe *OK*

Notă: Puteți ignora/închide deocamdată mesajul de „transformare a terenului” care ar putea apărea.

---

3. Încărcați fișierele în QGIS:

   1. Clic pe pictograma de Deschidere a Managerului Sursei de Date. Managerul sursei de date ar trebui să se deschidă în modul de Navigare.

   2. Intrați în folderul `qgis_sample_data/raster/`

   3. Selectați fișierul ERDAS IMG *landcover.img* și faceți dublu clic pe el. Stratul de acoperire a terenului este adăugat în fundal, în timp ce fereastra Managerului Sursei de Date rămâne deschisă.

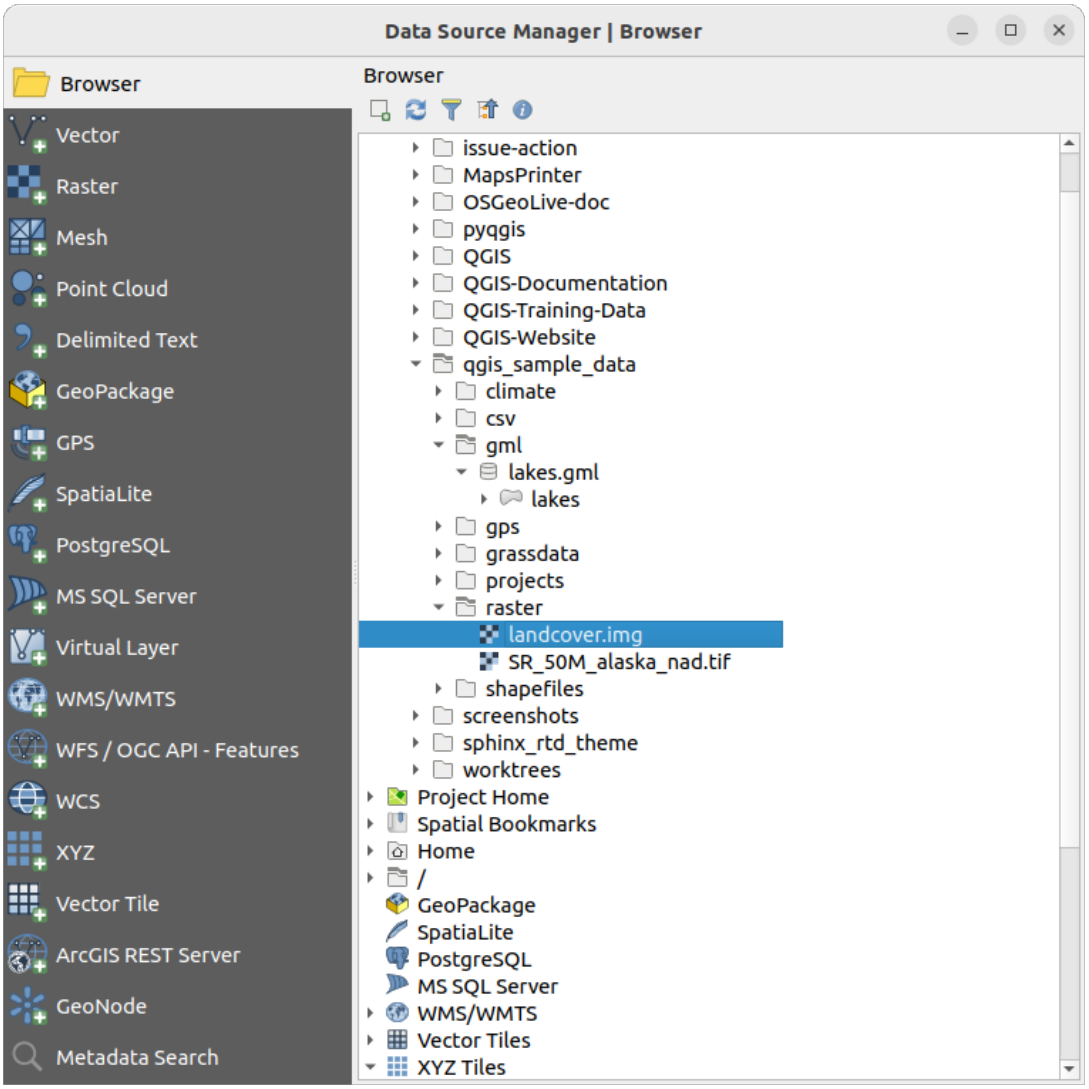


Fig. 5.2: Adăugarea de date unui nou proiect din QGIS

4. Pentru a încărca datele lacurilor, navigați la folderul `qgis_sample_data/gml/`, apoi trageți și plasați fișierul *lakes.gml* în ferestra principală din QGIS. (Sau faceți dublu clic, așa cum s-a menționat mai sus.)
5. Se va deschide fereastra de *Selectare a Elementelor de Adăugat*, care va scana fișierul. Acest lucru are loc datorită faptului că formatul de fișier `.gml` poate stoca mai mult de un singur strat.

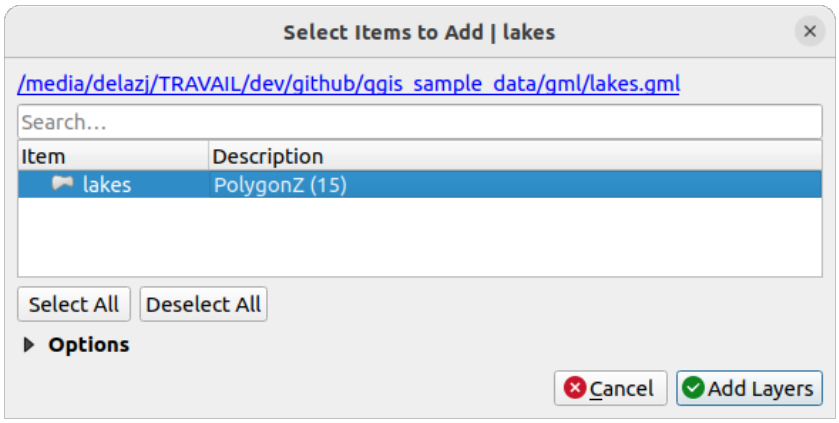


Fig. 5.3: Selectarea straturilor dintr-un fișier

6. În cazul nostru există un singur strat |poligonLayer| *lakes*. Selectați-l și apăsați *Adăugare Straturi*.
7. Stratul este adăugat în panoul *Straturilor*

4. Închideți fereastra Data Source Manager

În panoul *Sraturilor*, *lakes* ne informează că Stratul nu are setat un sistem de referință și coordonate. Haideți să remediem acest lucru.

1. Efectuați clic pepictorgrama . Se va deschide fereastra de *Selectare a Sistemului de Referință și Coordonate*.
2. Așa cum am făcut mai devreme, găsiți și selectați CRS-ul *NAD27 / Alaska Albers*.
3. Clic pe *OK*

Acum, cele două straturi sunt marcate în proiectul dvs. în culori aleatorii. Haideți să personalizăm un pic stratul lacurilor.

1. Selectați instrumentul Zoom In din bara instrumentelor de *Navigare*
2. Alegeți o zonă cu câteva lacuri
3. Efectuați dublu clic pe stratul `lakes` din legenda hărții, pentru a deschide fereastra *Proprietăților*
4. Pentru a schimba culoarea lacurilor:
    1. Clic pe fila |simbologie| *Simbologiei*
    2. Pentru umplere, alegeți culoarea albastră.

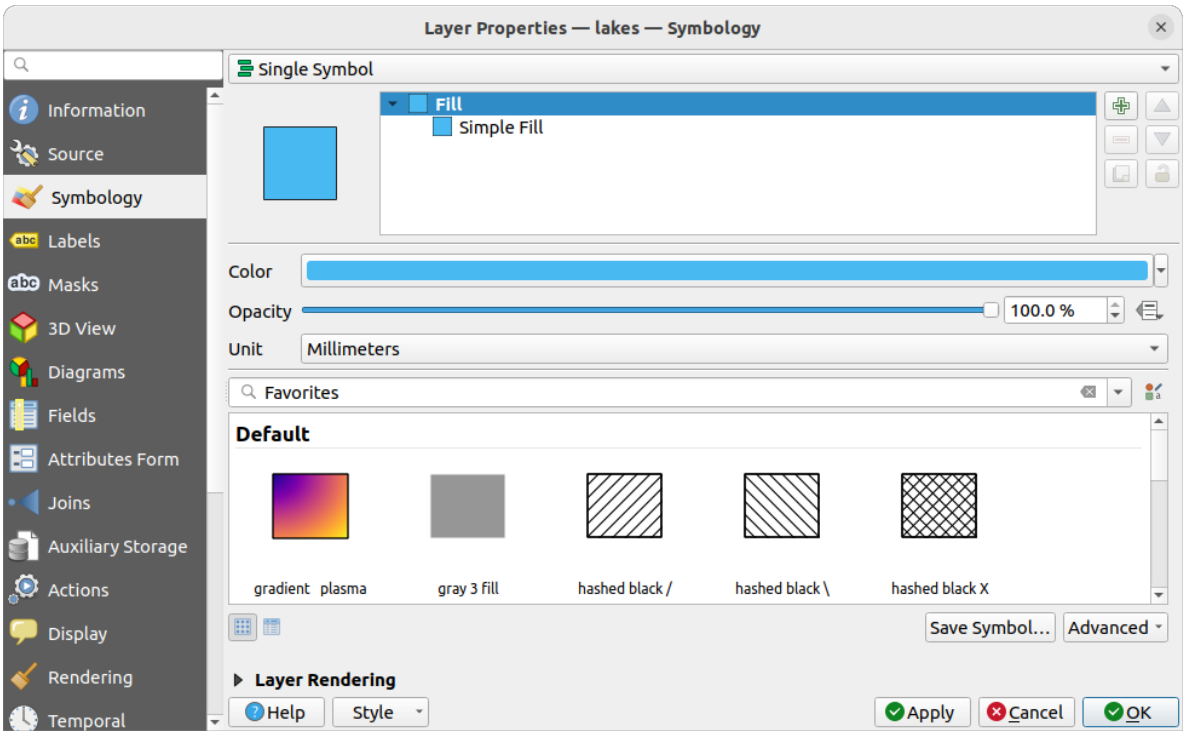


Fig. 5.4: Selectarea culorii pentru Lakes

3. Apăsați *OK*. Lacurile sunt acum prezentate în albastru pe canevasul hărții.

5. Pentru a afișa numele lacurilor:

   1. Deschideți din nou fereastra de *Proprietăți* a stratului `lakes`
   2. Faceți clic pe fila *Etichetelor*.
   3. Pentru a activa etichetarea alegeți *Etichete unice* în caseta de selectare.
   4. În lista de *Etichetare cu*, alegeți câmpul `NAMES`.

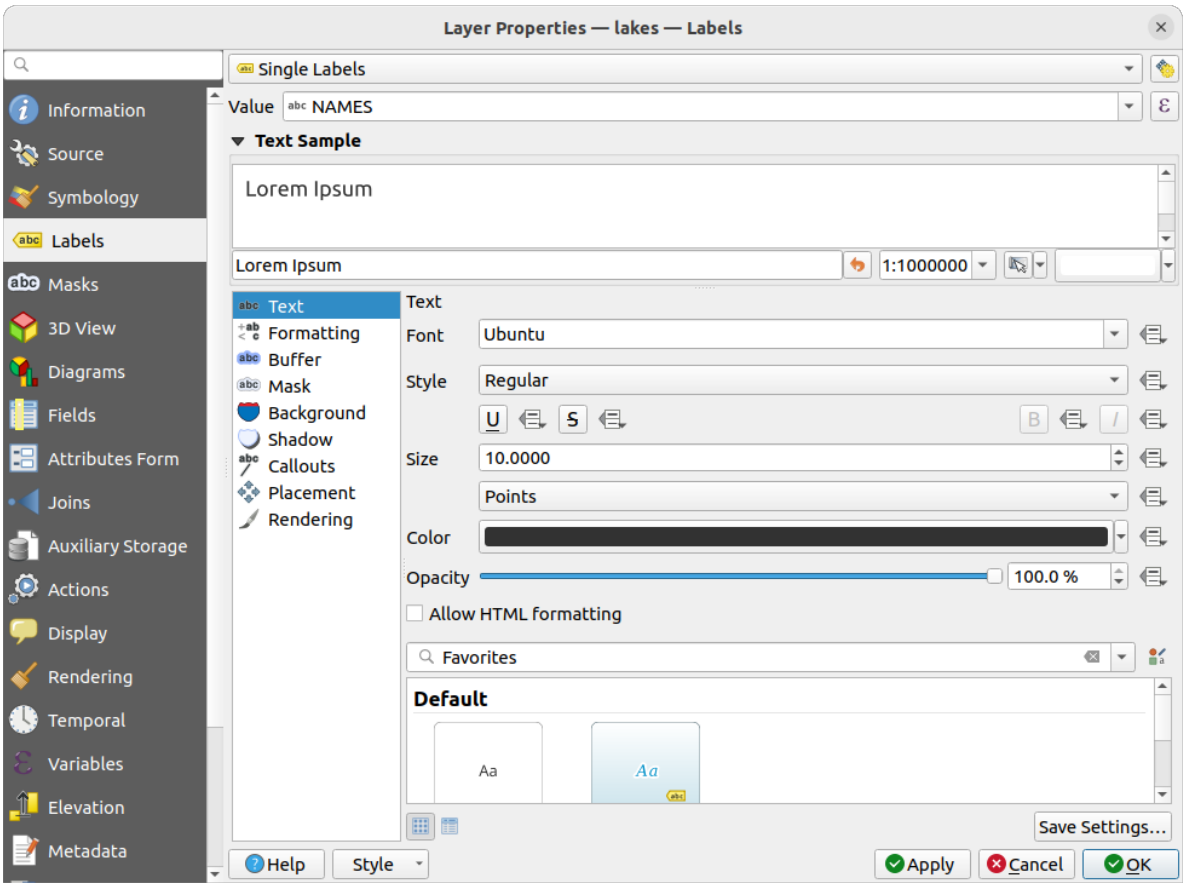


Fig. 5.5: Afișarea numele lacurilor

5. Clic pe *Aplicare*. Numele se vor încărca deasupra limitelor.

6. Puteți îmbunătăți lizibilitatea etichetelor adăugând un tampon alb în jurul lor:

   1. Faceți clic pe fila *Buffer* din lista din partea stângă
   2. Bifați *Desenare zonă tampon în jurul textului*
   3. Alegeți `3` pentru dimensiunea zonei tampon
   4. Clic pe *Aplicare*.
   5. Verificați dacă rezultatul arată bine și actualizați valoarea dacă este cazul.
   6. În cele din urmă, faceți clic pe *OK* pentru a închide fereastra *Proprietăților stratului* și pentru a aplica modificările.

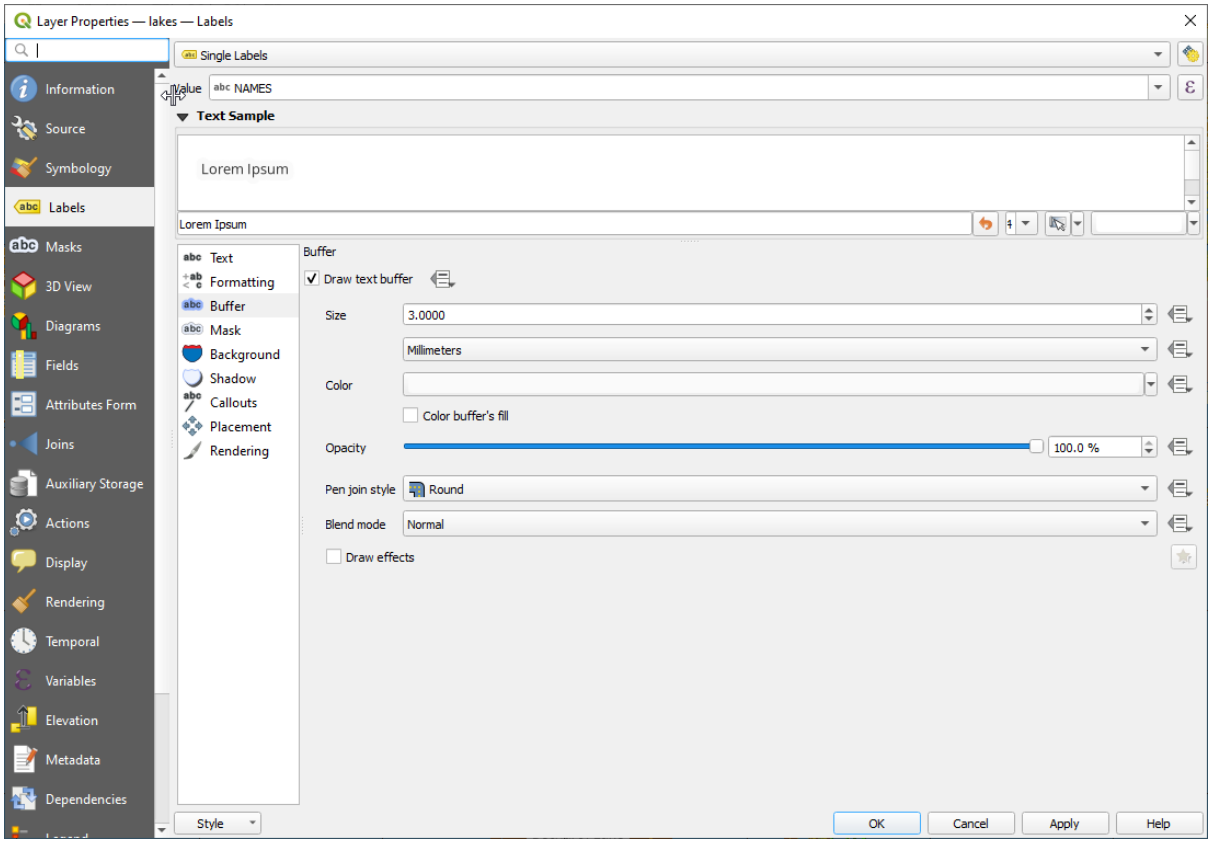


Fig. 5.6: Showing Buffers around Labels

Să adăugăm acum câteva decorațiuni pentru a întregi harta și pentru a o exporta din QGIS:

1. Selectați *Vizualizare* ► *Decorațiuni* ► *Scară grafică*
2. În fereastra de dialog care se deschide, bifați opțiunea de *Activare a Scării Grafice*
3. Personalizați după dorință opțiunile ferestrei de dialog
4. Apăsați pe *Aplicare*
5. În mod similar, din meniul decorațiunilor, adăugați mai multe elemente (săgeată nord, drepturi de autor…) pe canevasul hărții.
6. Faceți clic pe *Proiect* ► *Importare/Exportare* ► *Exportare Hartă într-o Imagine…*
7. Apăsați *Salvare* în fereastra care se deschide
8. Selectați o locație de fișier, un format și confirmați apăsând din nou pe *Salvare*.
9. Alegeți *Proiect* ► *Salvare…* pentru a stoca modificările dvs. într-un fișier de proiect `.qgz`.

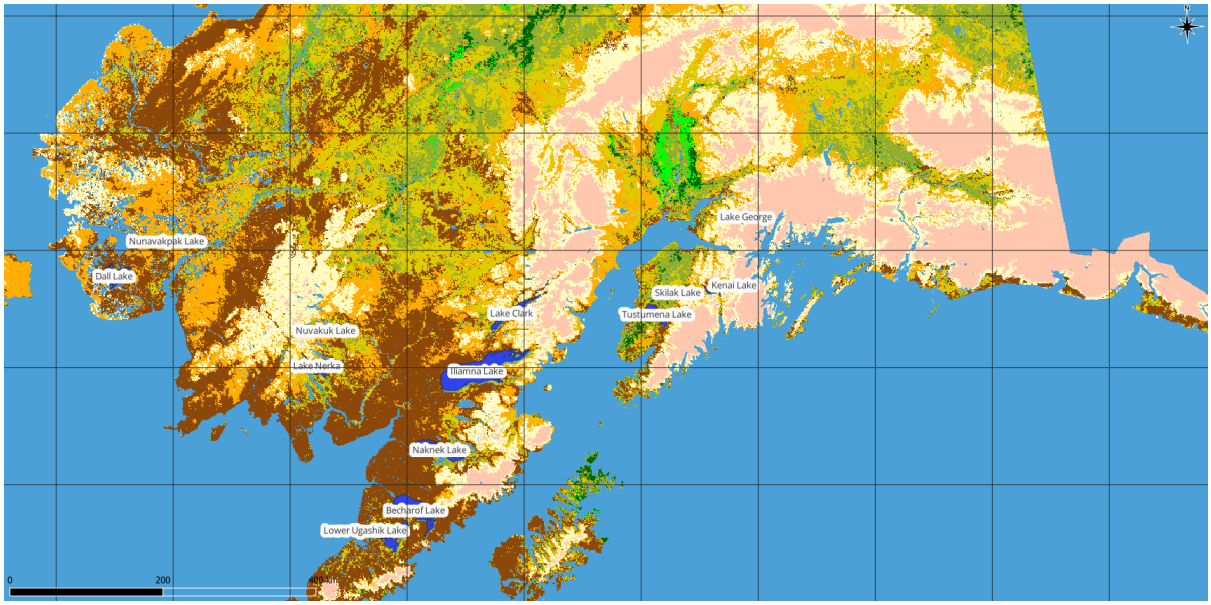


Fig. 5.7: Showing Exported Map with Decorations

Asta este! Ați văzut cât de ușor este să vizualizați straturi raster și vectoriale în QGIS, să le configurați și să generați harta într-un format de imagine utilizabil în alte softuri. Să mergem mai departe, pentru a afla mai multe despre funcționalitățile, caracteristicile și setările disponibile și despre cum se pot utiliza.

Notă: Pentru a continua să învățați QGIS prin exerciții pas cu pas, apelați la Manualul de instruire.

CAPITOLUL 6

Lucrul cu Fișierele Proiectelor

6.1 Prezentarea proiectelor QGIS

Starea unei sesiuni de QGIS poartă numele de proiect. În orice moment, QGIS permite lucrul la un singur proiect. O setare poate fi specifică doar unui proiect sau poate fi valabilă, la nivel de aplicație, pentru toate proiectele noi (v. secțiunea *Opţiuni*). QGIS poate salva starea spațiului dvs. de lucru într-un *fișier proiect QGIS* folosind opțiunile de meniu *Proiect* ► *Salvare* sau *Proiect* ► *Salvare Ca…*.

Notă: Dacă proiectul a fost modificat, simbolul * va apărea în bara de titlu iar QGIS vă va întreba, în mod implicit, dacă doriți să salvați modificările. Acest comportament este controlat de opțiunea de *Solicitare a salvării modificărilor proiectului și sursei de date atunci când este necesar* setarea din *Setări* ► *Opțiuni* ► *Generalități*.

Puteți încărca proiectele existente în QGIS folosind panoul Navigatorului sau accesând *Proiect* ► *Deschide…*, *Proiect* ► *Nou din șablon* sau *Proiect* ► *Deschide Recent* ►.

La pornire, sunt afișate listele *Șabloanelor proiect* și a *Proiectelor recente*, inclusiv capturi de ecran, nume și căi de fișiere (pentru până la zece proiecte). Lista *Proiectelor recente* permite accesarea proiectelor utilizate recent. Faceți dublu clic pe o poziție pentru a deschide proiectul sau șablonul de proiect. Efectuați clic dreapta pe o poziție pentru a o *Reține în Listă*, pentru a-i *Deschide Directorul…* sau pentru a o *Elimina din Listă*. De asemenea, puteți adăuga un strat pentru a crea automat un nou proiect. Listele vor dispărea apoi, făcând loc canevasului hărții.

Dacă doriți să ștergeți sesiunea și să reîncepeți, accesați *Proiect* ► *Nou*. Vi se va cere să salvați proiectul existent, dacă s-au efectuat modificări de când a fost deschis sau de la ultima salvare.

Când deschideți un proiect nou, bara de titlu va afișa `Proiect fără titlu` până când îl veți salva.

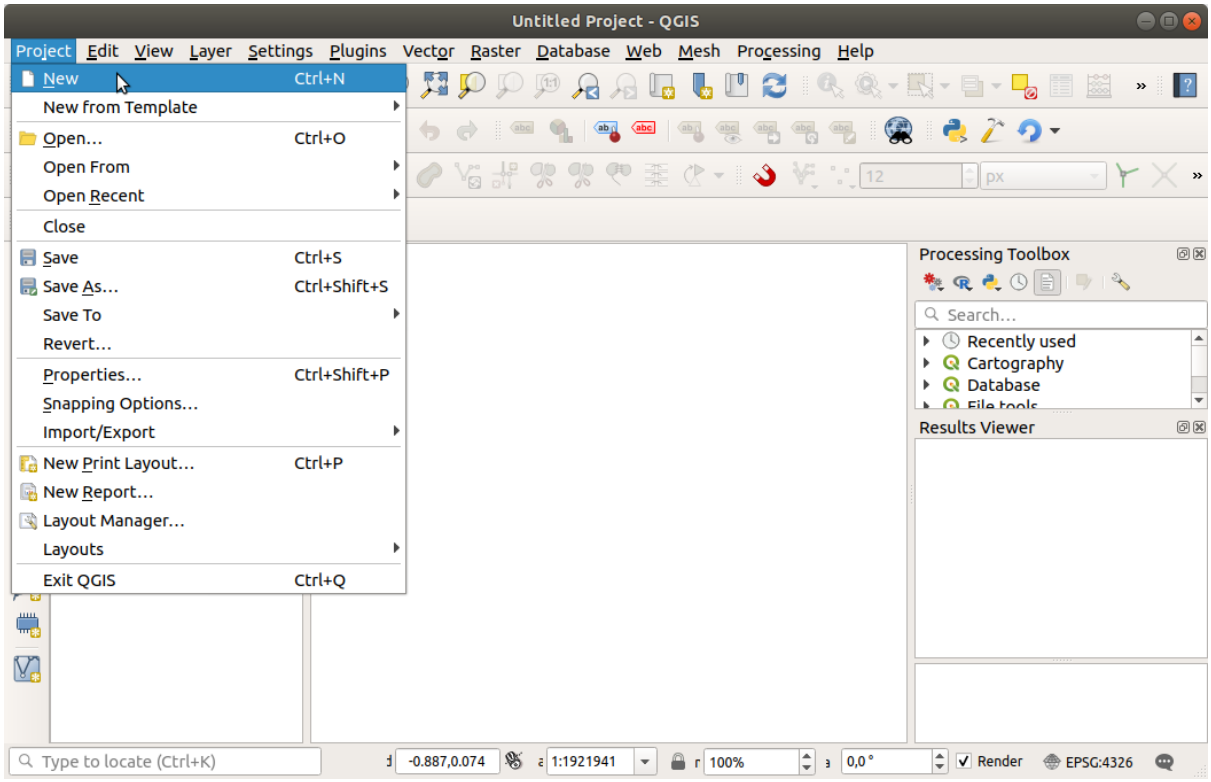


Fig. 6.1: Începerea unui nou proiect în QGIS

Informațiile salvate într-un fișier de proiect includ:

- Straturile adăugate
- Straturile care pot fi interogate
- Proprietățile stratului, inclusiv simbolizarea și stilurile
- Notițele straturilor
- Vizualizări 2D și 3D ale hărților
- Proiecția pentru fiecare vizualizare de hartă
- Ultima extindere vizualizată pentru fiecare hartă
- Compozițiile imprimabile
- Setări pentru elementele compozițiilor imprimabile
- Setări pentru atlasurile compozițiilor imprimabile
- Setări de digitizare
- Relațiile dintre Tabele
- Comenzi Macro pentru Proiecte
- Stilurile implicite ale proiectului
- Setările pkuginurilor
- Setările serverului QGIS, din fila Setărilor OWS a Proprietăților proiectului
- Interogările stocate în DB Manager

Fișierul proiect este salvat în format XML (v. *QGISQGISQGS/QGZ - Formatul Fișierelor Proiect QGIS*). Aceasta înseamnă că este posibil să editați fișierul în afara QGIS, dacă știți ce faceți. Formatul fișierului de proiect a fost

actualizat de mai multe ori. Este posibil ca fișierele de proiect din versiunile QGIS mai vechi să nu mai funcționeze corect.

Notă: În mod implicit, QGIS vă va avertiza cu privire la diferențele de versiune. Acest comportament este controlat în fila *Generalități* din *Setări* ► *Opțiuni* (☑ *Avertizare la deschiderea unui fișier de proiect salvat cu o versiune mai veche de QGIS*).

Ori de câte ori salvați un fișier de proiect `.qgs` în QGIS, o copie de rezervă a fișierului este creată în același director cu fișierul de proiect care are extensia `.qgs~`.

The extension for QGIS projects is `.qgs` but when saving from QGIS, the default is to save using a compressed format with the `.qgz` extension. The `.qgs` file is embedded in the `.qgz` file (a zip archive), together with its associated SQLite database (`.qgd`) for *auxiliary data*. You can get to these files by unzipping the `.qgz` file.

Notă: Mecanismul *Auxiliary Storage Properties* face un proiect zip deosebit de util, deoarece acesta înglobează date auxiliare.

Projects can also be saved/loaded to/from a PostgreSQL, GeoPackage or Oracle database using the following Project menu items:

- *Proiect* ► *Deschidere din*
- *Proiect* ► *Salvare în*

Both menu items have a sub-menu with a list of extra project storage implementations (PostgreSQL, GeoPackage and Oracle). Clicking the action will open a dialog to pick a GeoPackage connection and project, a PostgreSQL connection, schema and project or Oracle connection, owner and project.

Projects stored in GeoPackage, PostgreSQL or Oracle can also be loaded through the QGIS browser panel, either by double-clicking them or by dragging them to the map canvas.

6.2 Gestionarea căilor de fișiere întrerupte

La deschiderea unui proiect, QGIS poate să nu găsească unele surse de date, din cauza indisponibilității serviciului/bazei de date sau a redenumirii sau mutării unui fișier. QGIS va deschide fereastra de *Gestionare a straturilor indisponibile*, care face referire la straturile negăsite. Veți putea:

- Efectua dublu clic în câmpul *Sursei de date*, să ajustați calea fiecărui strat apoi să faceți clic pe *Aplicare modificări*;
- Selecta un rând, să apăsați pe *Răsfoire* pentru a indica locația corectă apoi să faceți clic pe *Aplicare modificări*;
- Apăsa pe *Auto-Identificare* pentru a răsfoi dosarele și a încerca remedierea automată a tuturor căi(lor) incorecte sau doar a celor selectate. Rețineți că răsfoirea poate dura ceva timp. Apoi faceți clic pe *Aplicare modificări*.
- Ignora mesajul și să deschideți proiectul cu căi(le) întrerupte făcând clic pe *Păstrare straturi indisponibile*. Stratul dvs. este apoi afișat în panoul *Straturilor*, dar fără date, până la remedierea căii cu ajutorul pictogramei ⚠ Strat indisponibil! afișată lângă acesta în panoul *Straturilor* sau alegând *Reparare sursă de date…* din meniul contextual al stratului.

  Când se folosește instrumentul de *Reparare sursă de date…*, o dată ce o cale a stratului a fost remediată, QGIS scanează toate celelalte căi întrerupte și încearcă să le repare automat pe cele care au întreruptă aceeași cale de fișier.
- 🗑 *Elimină straturile indisponibile* din proiect.

Lansarea QGIS din linia de comandă folosind opțiunea *--skipbadlayers* vă poate ajuta să săriți peste fereastra de *Gestionare a straturilor indisponibile* de la pornire.

6.3 Obținerea rezultatelor

Există mai multe moduri de a genera rezultate din sesiunea dvs. de QGIS. Am discutat deja despre salvarea sub formă de fișier proiect în *Prezentarea proiectelor QGIS*. Alte modalități de a produce fișiere de ieșire sunt:

- Creating images: *Project* ► *Import/Export* ► *Export Map to Image…* outputs the map canvas rendering to an image format (PNG, JPG, TIFF…) at custom scale, resolution and size. Including georeference information in the exported image is possible, simply enable *Append georeference information (embedded or via world file)*. See *Exportarea vizualizării hărții* for more details.
- Exportarea în fișiere PDF: *Proiect* ► *Importare/Exportare* ► *Exportare Hartă într-un PDF…* redă canevasul hărții în format PDF la scară, rezoluție personalizată și cu unele setări avansate (simplificare, georeferențiere, …). Consultați *Exportarea vizualizării hărții* pentru mai multe detalii.
- Exportarea în fișiere DXF: *Proiect* ► *Importare/Exportare* ► *Exportare Proiect într-un DXF…* deschide o fereastră în care se poate defini «Modul simbologiei», «Scara simbologiei» și straturile vectoriale care trebuie să fie exportate în DXF. Prin intermediul «Modului simbologiei», simbolurile originale din QGIS pot fi exportate cu fidelitate ridicată (v. secțiunea *Crearea fișierelor DXF noi*).
- Proiectarea hărților: *Proiect* ► *Compoziție Imprimabilă Nouă …* deschide o fereastră de dialog în care puteți aranja și tipări canevasul hărții curente (v. secțiunea *Așezarea în pagină a hărților*).

CAPITOLUL 7

QGIS GUI

Interfața grafică cu utilizatorul (GUI) QGIS este prezentată în figura de mai jos (numerele de la 1 la 5 încercuite cu galben indică elemente importante ale GUI QGIS și sunt detaliate în continuare).

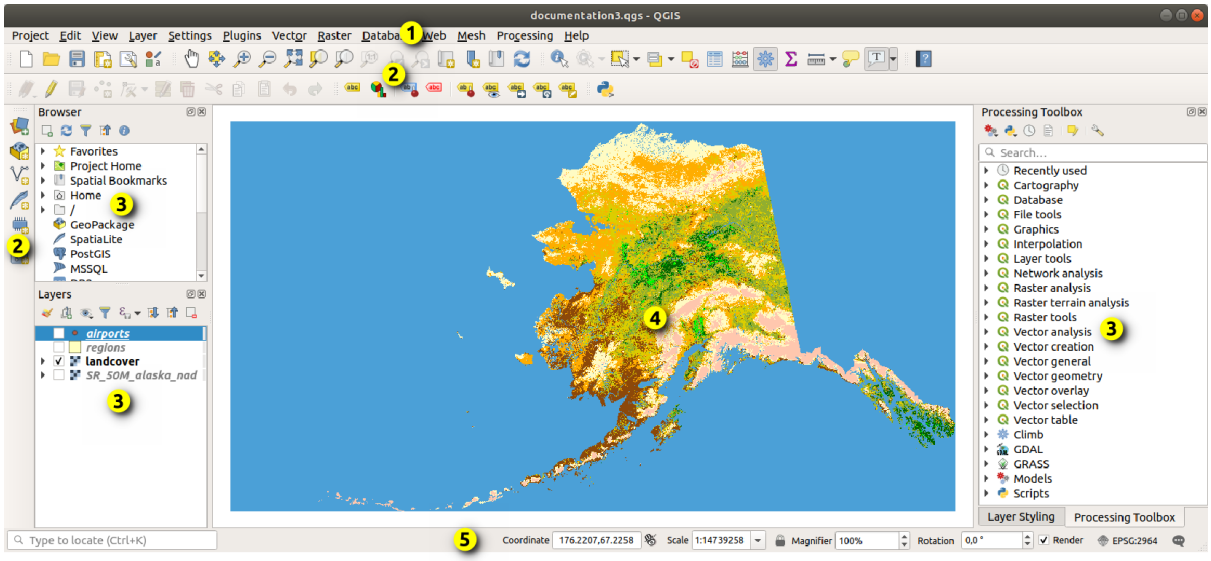


Fig. 7.1: Interfața QGIS cu datele exemplificative pentru Alaska

Notă: Aspectul ferestrelor (bara de titlu, etc) poate fi diferit, în funcție de sistemul de operare și de managerul de ferestre.

Interfața principală QGIS (Fig. 7.1) constă în cinci componente/tipuri de componente:

1. *Bara de Meniuri*
2. *Bare cu instrumente*
3. *Panouri*
4. *Vizualizare Hartă*

5. *Bara de Stare*

Derulați în jos pentru explicații detaliate ale acestora.

7.1 Bara de Meniuri

Bara de meniu oferă acces la funcțiile QGIS folosind meniuri standard, ierarhizate. Meniurile, opțiunile acestora, pictogramele asociate și combinațiile de taste sunt descrise mai jos. Combinațiile de taste pot fi reconfigurate (*Setări* ► *Combinații de taste*).

Pentru majoritatea comenzilor dintr-un meniu există câte un instrument corespunzător și invers. Cu toate acestea, meniurile nu sunt organizate întocmai barelor de instrumente. În tabelul de mai jos este prezentată corespondența dintre comenzile meniurilor și barele de instrumente. Pluginurile pot introduce noi opțiuni în Meniuri. Pentru mai multe informații despre instrumente și barele acestora, consultați *Bare de instrumente*.

Notă: QGIS este o aplicație multiplatformă. Instrumentele sunt în general disponibile pe toate platformele, dar pot fi plasate în meniuri diferite, în funcție de sistemele de operare. Listele de mai jos arată cele mai comune locații, inclusiv variante cunoscute.

7.1.1 Proiect

Meniul *Proiect* oferă puncte de acces și ieșire pentru *fișierele de proiect*. Dispune de instrumente pentru:

- Crearea unui fișier de proiect *Nou* de la zero sau prin utilizarea unui alt fișier de proiect ca șablon (consultați *Opțiunile fișierelor de proiect* pentru configurarea șablonului)
- *Open…* a project from a file, a GeoPackage, PostgreSQL or Oracle database
- *Închiderea* unui proiect sau revenirea la ultima stare salvată
- *Save* a project in `.qgs` or `.qgz` file format, either as a file or within a GeoPackage, PostgreSQL or Oracle database
- Exportarea canevasului hărții în diverse formate sau utilizarea unei *compoziții imprimabile* pentru rezultate mai complexe
- Setarea proprietăților proiectului și a opțiunilor de acroșare în scopul editării geometriei.

Tabelul 7.1: Comenzile meniului Proiect

Comanda din Meniu	Scurtătură	Bara de Instrume	Referință
Nou	`Ctrl+N`	*Proiect*	*Prezentarea proiectelor QGIS*
Nou după șablon ►			*Prezentarea proiectelor QGIS*
Deschidere…	`Ctrl+O`	*Proiect*	*Prezentarea proiectelor QGIS*
Deschidere din ►			*Prezentarea proiectelor QGIS*
► *GeoPackage…*			*Prezentarea proiectelor QGIS*
► *PostgreSQL…*			*Prezentarea proiectelor QGIS*
► *Oracle…*			*Prezentarea proiectelor QGIS*
Deschidere Proiect Recent ►	`Alt+J + R`		*Prezentarea proiectelor QGIS*
Închidere			*Prezentarea proiectelor QGIS*
Salvare	`Ctrl+S`	*Proiect*	*Prezentarea proiectelor QGIS*
Salvare Ca…	`Ctrl+Shift+S`	*Proiect*	*Prezentarea proiectelor QGIS*
Salvare în ►			*Prezentarea proiectelor QGIS*
► *Șabloane…*			*Prezentarea proiectelor QGIS*
► *GeoPackage…*			*Prezentarea proiectelor QGIS*
► *PostgreSQL…*			*Prezentarea proiectelor QGIS*
Revenire…			
Proprietăți…	`Ctrl+Shift+P`		*Proprietăți Proiect*
Opțiuni de Acroșare…			*Opțiuni de Digitizare și Acroșare*
Import/Export ►			
► *Exportare Hartă în Imagine…*			*Exportarea vizualizării hărții*
► *Exportare Hartă în PDF…*			*Exportarea vizualizării hărții*
► *Exportare Proiect în DXF…*			*Crearea fișierelor DXF noi*
► *Importare Straturi din DWG/DXF…*			*Importarea unui fișier DXF sau DWG*
Compoziție Imprimabilă Nouă…	`Ctrl+P`	*Proiect*	*Așezarea în pagină a hărților*
Raport Nou…			*Generarea unui Raport*
Managerul de Compoziții…		*Proiect*	*Așezarea în pagină a hărților*
Compoziții ►			*Așezarea în pagină a hărților*
Modele ►			*The model designer*
Ieșire din QGIS	`Ctrl+Q`		

În macOS, comanda corespunzătoare *Ieșirii din QGIS* este *QGIS* ► *Închidere QGIS* (`Cmd+Q`).

7.1.2 Editare

The *Edit* menu provides most of the native tools needed to edit layer attributes or geometry. To enable *Edit* menu options you need to switch to editing mode by clicking on Toggle editing (see *Editarea* for details).

Tabelul 7.2: Comenzile meniului Editare

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Anulare	`Ctrl+Z`	*Digitizare*	*Anulare/Restabilire*
Refacere	`Ctrl+Shift+Z`	*Digitizare*	*Anulare/Restabilire*
Decupare entități	`Ctrl+X`	*Digitizare*	*Tăiere, Copiere și Lipire Entități*
Copiere entități	`Ctrl+C`	*Digitizare*	*Tăiere, Copiere și Lipire Entități*
Lipire Entități	`Ctrl+V`	*Digitizare*	*Tăiere, Copiere și Lipire Entități*
Lipire Entități într-un ►			*Lucrul cu Tabela de Atribute*
► *Strat Vectorial Nou……*			*Lucrul cu Tabela de Atribute*
► *Strat Temporar…*	`Ctrl+Alt+V`		*Lucrul cu Tabela de Atribute*
Ștergere Selecție		*Digitizare*	*Ștergerea Entităților Selectate*
Selectare ►			*Selectare entități*
► *Selectare Entit(ăți)*		*Selecție*	*Selectare entități*
► *Selectare Entități după un Poligon*		*Selecție*	*Selectare entități*
► *Selectare Entități în Mod Liber*		*Selecție*	*Selectare entități*
► *Selectare Entități după o Rază*		*Selecție*	*Selectare entități*
► *Selectare Entități După o Valoare…*	`F3`	*Selecție*	*Selectare entități*
► *Selectare Entități După o Expresie…*	`Ctrl+F3`	*Selecție*	*Selectare entități*
► *Deselectează Entitățile din Toate Straturile*	`Ctrl+Alt+A`	*Selecție*	*Selectare entități*
► *Deselectează Entitățile din Stratul Activ*	`Ctrl+Shift+A`	*Selecție*	*Selectare entități*
► *Reselectare Entități*			*Selectare entități*
► *Selectează Toate Entitățile*	`Ctrl+A`	*Selecție*	*Selectare entități*
► *Inversează Selecția Entităților*		*Selecție*	*Selectare entități*
Adăugare Înregistrare	`Ctrl+.`	*Digitizare*	
Adăugare Entitate de tip Punct	`Ctrl+.`	*Digitizare*	*Adăugare Entități*
Adăugare Entitate de tip Linie	`Ctrl+.`	*Digitizare*	*Adăugare Entități*
Adăugare Entitate de tip Poligon	`Ctrl+.`	*Digitizare*	*Adăugare Entități*
Add Circular String		*Shape Digitizing*	*Șir circular în funcție de rază*
Add Circular String by Radius		*Shape Digitizing*	*Șir circular în funcție de rază*
Add Circle ►		*Shape Digitizing*	*Desenare Cercuri*
► *Add Circle from 2 Points*		*Shape Digitizing*	*Desenare Cercuri*

continues on next page

Tabelul 7.2 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
► *Add Circle from 3 Points*		*Shape Digitizing*	*Desenare Cercuri*
► *Add Circle from 3 Tangents*		*Shape Digitizing*	*Desenare Cercuri*
► *Add Circle from 2 Tangents and a Point*		*Shape Digitizing*	*Desenare Cercuri*
► *Add Circle by a Center Point and Another Point*		*Shape Digitizing*	*Desenare Cercuri*
Add Rectangle ►		*Shape Digitizing*	*Desenare Dreptunghiuri*
► *Add Rectangle from Extent*		*Shape Digitizing*	*Desenare Dreptunghiuri*
► *Add Rectangle from Center and a Point*		*Shape Digitizing*	*Desenare Dreptunghiuri*
► *Add Rectangle from 3 Points (Distance from 2nd and 3rd point)*		*Shape Digitizing*	*Desenare Dreptunghiuri*
► *Add Rectangle from 3 Points (Distance from projected point on segment p1 and p2)*		*Shape Digitizing*	*Desenare Dreptunghiuri*
Add Regular Polygon ►		*Shape Digitizing*	*Desenare Poligoane Regulate*
► *Add Regular Polygon from Center and a Point*		*Shape Digitizing*	*Desenare Poligoane Regulate*
► *Add Regular Polygon from Center and a Corner*		*Shape Digitizing*	*Desenare Poligoane Regulate*
► *Add Regular Polygon from 2 Points*		*Shape Digitizing*	*Desenare Poligoane Regulate*
Add Ellipse ►		*Shape Digitizing*	*Desenare Elipse*
► *Add Ellipse from Center and 2 Points*		*Shape Digitizing*	*Desenare Elipse*
► *Add Ellipse from Center and a Point*		*Shape Digitizing*	*Desenare Elipse*
► *Add Ellipse from Extent*		*Shape Digitizing*	*Desenare Elipse*
► *Add Ellipse from Foci*		*Shape Digitizing*	*Desenare Elipse*
Adăugare Adnotare ►			*Instrumente de Adnotare*
► *Adnotare de tip Text*		*Adnotări*	*Instrumente de Adnotare*
► *Adnotare de tip Formular*		*Adnotări*	*Instrumente de Adnotare*
► *Adnotare de tip HTML*		*Adnotări*	*Instrumente de Adnotare*
► *Adnotare de tip SVG*		*Adnotări*	*Instrumente de Adnotare*
Editare Atribute ►			

continues on next page

Tabelul 7.2 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
► *Modifică Atributele Entităților Selectate*		*Digitizare*	*Editarea valorilor atributului*
► *Îmbină Atributele Entităților Selectate*		*Digitizare Avansată*	*Unificați atributele pentru entitățile selectate*
Editare Geometrie ►			
► *Deplasare Entit(ăți)*		*Digitizare Avansată*	*Deplasare Enti(tăți)*
► *Copiere și Deplasare Entit(ăți)*		*Digitizare Avansată*	*Deplasare Enti(tăți)*
► *Rotire Entit(ăți)*		*Digitizare Avansată*	*Rotire Enti(tăți)*
► *Scalare Entit(ăți)*		*Digitizare Avansată*	*Scalare Entități*
► *Simplificare Entitate*		*Digitizare Avansată*	*Simplificare Entitate*
► *Adăugare Inel*		*Digitizare Avansată*	*Adăugare Inel*
► *Adăugare Parte*		*Digitizare Avansată*	*Adăugare Parte*
► *Umplere Inel*		*Digitizare Avansată*	*Umplere Inel*
► *Ștergere Inel*		*Digitizare Avansată*	*Ştergere Inel*
► *Ștergere Parte*		*Digitizare Avansată*	*Ştergere Parte*
► *Remodelare Entități*		*Digitizare Avansată*	*Remodelare Entități*
► *Decalare Curbă*		*Digitizare Avansată*	*Curbe de Compensare*
► *Divizare Entități*		*Digitizare Avansată*	*Divizare Entități*
► *Divizare Părți*		*Digitizare Avansată*	*Divizare părți*
► *Îmbinare Entități Selectate*		*Digitizare Avansată*	*Unificați entitățile selectate*
► *Instrumentul Vertex (Toate Straturile)*		*Digitizare*	*Instrumentul vertex*
► *Instrumentul Vertex (Stratul Curent)*		*Digitizare*	*Instrumentul vertex*
► *Inversare Linie*		*Digitizare Avansată*	*Inversare Linie*
► *Diminuare/majorare Entitate*		*Digitizare Avansată*	*Reducere/Extindere Entitate*

continues on next page

Tabelul 7.2 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Rotirea Simbolurilor Punctelor		*Digitizare Avansată*	*Rotiți Simbolurile Punctelor*
Decalarea Simbolurilor Punctelor		*Digitizare Avansată*	*Decalarea Simbolurilor Punctelor*

Instrumentele care depind de tipul de geometrie al stratului selectat, adică punct, polilinie sau poligon, sunt activate în mod corespunzător:

Tabelul 7.3: Pictogramele „Deplasare entitate” bazate pe geometrie.

Comanda din Meniu	Punct	Polilinie	Poligon
Deplasare Entit(ăți)			
Copiere și Deplasare Entit(ăți)			

7.1.3 Vizualizare

Harta este prezentată în ferestre de vizualizare. Puteți interacționa cu aceste vizualizări folosind instrumentele *Vizualizare*. De exemplu, puteți:

- Crea vizualizări 2D sau 3D ale hărților, dispuse lângă canevasul principal al hărții
- *Mări sau muta* oriunde
- Query the attributes or geometry of the displayed features
- Îmbunătăți vizualizarea hărții cu diferite moduri de previzualizare, adnotări sau decorațiuni
- Accesa orice panou sau bară de instrumente

De asemenea, meniul vă permite să reorganizați interfața QGIS folosind acțiuni precum:

- *Trecere în Modul Ecran Întreg*: acoperă întregul ecran în timp ce este ascunsă bara de titlu
- *Toggle Panel Visibility*: shows or hides enabled *panels* - useful when digitizing features (for maximum canvas visibility) as well as for (projected/recorded) presentations using QGIS main canvas
- *Trecere în modul Hartă Exclusivă*: ascunde panourile, barele de instrumente, meniurile și bara de stare și arată doar canevasul hărții. În combinație cu opțiunea de ecran întreg, face ca ecranul dvs. să afișeze doar harta

Tabelul 7.4: Comenzile meniului Vizualizare

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Nouă Vizualizare de Hartă	`Ctrl+M`	*Navigare pe Hartă*	*2D Map View*
Vizualizare Hărți în 3D ►			*3D Map View*
► *Nouă Vizualizare de Hartă în 3D*	`Ctrl+Alt+M`	*Navigare pe Hartă*	*3D Map View*
► *Gestionarea Vizualizărilor Hărților în 3D*			*3D Map View*
Deplasare Hartă		*Navigare pe Hartă*	*Exploring the map view*

continues on next page

Tabelul 7.4 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Deplasează Harta spre Selecție		*Navigare pe Hartă*	*Exploring the map view*
Mărire	`Ctrl+Alt++`	*Navigare pe Hartă*	*Exploring the map view*
Micșorare	`Ctrl+Alt+-`	*Navigare pe Hartă*	*Exploring the map view*
Identificare Entități	`Ctrl+Shift+I`	*Atribute*	*Identifying Features*
Măsurare ►		*Atribute*	*Măsurarea*
► *Măsurare Linie*	`Ctrl+Shift+M`	*Atribute*	*Măsurarea*
► *Măsurare Arie*	`Ctrl+Shift+J`	*Atribute*	*Măsurarea*
► *Measure Bearing*		*Atribute*	*Măsurarea*
► *Măsurare Unghi*		*Atribute*	*Măsurarea*
Sumar Statistic		*Atribute*	*Statistical Summary Panel*
Elevation Profile			*Elevation Profile View*
Transfocare la nivelul Hărții	`Ctrl+Shift+F`	*Navigare pe Hartă*	*Exploring the map view*
Transfocare la nivelul Selecției	`Ctrl+J`	*Navigare pe Hartă*	*Exploring the map view*
Zoom Pe Strat(uri)		*Navigare pe Hartă*	*Exploring the map view*
Zoom La Rezoluția Nativă (100%)		*Navigare pe Hartă*	*Exploring the map view*
Nivelul de Mărire Anterior		*Navigare pe Hartă*	*Exploring the map view*
Nivelul de Mărire Următor		*Navigare pe Hartă*	*Exploring the map view*
Decorațiuni ►	`Alt+V + D`		*Decorating the map*
► *Grilă…*			*Grilă*
► *Scară Grafică…*			*Scara Grafică*
► *Imagine…*			*Image Decoration*
► *Săgeata Nordului…*			*Săgeata Nordului*
► *Eticheta Titlului…*			*Title Label*
► *Eticheta Dreptului de Autor…*			*Copyright Label*
► *Extinderile Compoziției…*			*Layout Extents*
Mod de previzualizare ►			
► *Normal*			
► *Simulare Monocromie*			
► *Simulare Acromatopsie (Vedere în Nuanțe de Gri)*			

continues on next page

Tabelul 7.4 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
► *Simulare Protanopie (Neperceperea Culorii Roșii)*			
► *Simulare Deuteranopie (Neperceperea Culorii Verzi)*			
► *Simulare Tritanopie (Neperceperea Culorii Albastre)*			
Afișare Baloane Informative		*Atribute*	*Display Properties*
Reper Spațial Nou…	`Ctrl+B`	*Navigare pe Hartă*	*Bookmarking extents on the map*
Afișare Repere Spațiale	`Ctrl+Shift+B`	*Navigare pe Hartă*	*Bookmarking extents on the map*
Arată Managerul Reperelor Spațiale			*Bookmarking extents on the map*
Actualizare	`F5`	*Navigare pe Hartă*	
Vizibilitate Straturi ►			*Panoul Straturilor*
► *Arată Toate Straturile*	`Ctrl+Shift+U`		*Panoul Straturilor*
► *Arată Toate Straturile*	`Ctrl+Shift+H`		*Panoul Straturilor*
► *Arată Straturile Selectate*			*Panoul Straturilor*
► *Ascunde Straturile Selectate*			*Panoul Straturilor*
► *Inversează Selecția Straturilor*			*Panoul Straturilor*
► *Inversează Vizibilitatea Straturilor Selectate*			*Panoul Straturilor*
► *Ascundere Straturi Selectate*			*Panoul Straturilor*
Panouri ►			*Panouri și Bare de Instrumente*
Bare de Instrumente ►			*Panouri și Bare de Instrumente*
Comută în Modul Ecran Complet	`F11`		
Inversează Vizibilitatea Panoului	`Ctrl+Tab`		
Comută în modul Hartă Exclusivă	`Ctrl+Shift+Tab`		

În Linux KDE, *Panouri* ►, *Bare de instrumente* ► și *Comutare în Modul Ecran Întreg* se află în meniul *Setări*.

7.1.4 Strat

Meniul *Straturi* oferă un set mare de instrumente pentru *crearea* noilor surse de date, *adăugarea* lor la un proiect sau *salvarea modificărilor* din ele. Folosind aceleași surse de date, puteți, de asemenea:

- Să *Duplicați* un strat pentru a genera o copie unde se pot modifica numele, stilul (simbologie, etichete, …), îmbinările, … Copia folosește aceeași sursă de date ca și originalul.
- Să *Copiați* și să *Lipiți* straturi sau grupuri dintr-un proiect în altul, într-o nouă instanță ale cărei proprietăți pot fi modificate independent. În ceea ce privește *Duplicarea*, straturile se bazează în continuare pe aceeași sursă de date.
- sau să *Încorporați Straturi și Grupuri…* dintr-un alt proiect, sub formă de copii nemodificabile, care pot fi doar citite (vezi *Embedding layers from external projects*)

Meniul *Straturilor* conține, de asemenea, instrumente pentru configurarea, copierea sau lipirea proprietăților stratului (stilul, scara, CRS-ul…).

Tabelul 7.5: Comenzile meniului Straturi

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Managerul Surselor de Date	`Ctrl+L`	*Managerul Surselor de Date*	*Deschidere Date*
Creează Strat ►			*Creating new vector layers*
► *Strat GeoPackage Nou…*	`Ctrl+Shift+N`	*Managerul Surselor de Date*	*Creating a new GeoPackage layer*
► *Strat Shapefile Nou…*		*Managerul Surselor de Date*	*Crearea unui nou strat de tip Fișier Shape*
► *Strat SpatiaLite Nou…*		*Managerul Surselor de Date*	*Creating a new SpatiaLite layer*
► *Strat Temporar Nou…*		*Managerul Surselor de Date*	*Creating a new Temporary Scratch Layer*
► *Strat Nou de Suprafață Poliedrică…*		*Managerul Surselor de Date*	*Creating a new Mesh layer*
► *New GPX Layer…*		*Managerul Surselor de Date*	*Creating a new GPX layer*
► *Strat Virtual Nou…*		*Managerul Surselor de Date*	*Creating virtual layers*
Adaugă Strat ►			*Deschiderea datelor*
► *Adăugare Strat Vectorial……*	`Ctrl+Shift+V`	*Gestionare Straturi*	*Încărcarea unui strat dintr-un fișier*
► *Adăugare Strat Raster…*	`Ctrl+Shift+R`	*Gestionare Straturi*	*Încărcarea unui strat dintr-un fișier*
► *Adăugare Strat Nou de Suprafață Poliedrică…*		*Gestionare Straturi*	*Loading a mesh layer*
► *Adăugare Strat cu Text Delimitat…*	`Ctrl+Shift+T`	*Gestionare Straturi*	*Importarea unui fișier cu texte delimitate*
► *Adăugare Strat PostGIS…*	`Ctrl+Shift+D`	*Gestionare Straturi*	*Database related tools*
► *Adăugare Strat SpatiaLite…*	`Ctrl+Shift+L`	*Gestionare Straturi*	*Straturile SpatiaLite*
► *Adăugare Strat MS SQL Server…*		*Gestionare Straturi*	*Database related tools*
► *Adăugare Strat Oracle Spatial…*		*Gestionare Straturi*	*Database related tools*
► *Add SAP HANA Spatial Layer…*		*Gestionare Straturi*	*Database related tools*
► *Adăugare/Editare Strat Virtual…*		*Gestionare Straturi*	*Creating virtual layers*
► *Adăugare Strat WMS/WMTS…*	`Ctrl+Shift+W`	*Gestionare Straturi*	*Încărcarea Straturilor WMS/WMTS*

continues on next page

Tabelul 7.5 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
► *Adăugare Strat XYZ…*			*Using XYZ Tile services*
► *Adăugare Strat WCS…*		*Gestionare Straturi*	*Client WCS*
► *Add WFS / OGC API - Features Layer…*		*Gestionare Straturi*	*Client WFS și WFS-T*
► *Add ArcGIS REST Server Layer…*		*Gestionare Straturi*	*Using ArcGIS REST Servers*
► *Adăugare Strat de Plăcuțe Vectoriale…*			*Using Vector Tiles services*
► *Adăugare Strat cu Nor de Puncte…*			*Working with Point Clouds*
► *Adăugare Strat GPX…*			*Introducing GNSS/GPS Data*
Încapsulare Straturi și Grupuri…			*Embedding layers from external projects*
Adăugare din Fișierul cu Definițiile Stratului…			*Layer definition file*
Georeferențiator…			*Georeferencer*
Copiere Stil			*Salvarea și Partajarea Proprietăților Stratului*
Lipire Stil			*Salvarea și Partajarea Proprietăților Stratului*
Copiere Strat			
Lipire Strat/Grup			
Deschide Tabela de Atribute	`F6`	*Atribute*	*Lucrul cu Tabela de Atribute*
Filtrare Tabelă de Atribute ►			*Lucrul cu Tabela de Atribute*
► *Deschidere Tabelă de Atribute (Entități Selectate)*	`Shift+F6`	*Atribute*	*Lucrul cu Tabela de Atribute*
► *Deschidere Tabelă de Atribute (Entități Vizibile)*	`Ctrl+F6`	*Atribute*	*Lucrul cu Tabela de Atribute*
► *Deschidere Tabelă de Atribute (Entități Noi și Editate)*		*Atribute*	*Lucrul cu Tabela de Atribute*
Comută Editarea		*Digitizare*	*Digitizarea unui strat vectorial existent*
Salvează Modificările aduse Stratului		*Digitizare*	*Salvarea Straturilor Modificate*
Editări Curente ►		*Digitizare*	*Salvarea Straturilor Modificate*
► *Salvare Strat(uri) Selectate*		*Digitizare*	*Salvarea Straturilor Modificate*
► *Refacere Strat(uri) Selectate*		*Digitizare*	*Salvarea Straturilor Modificate*
► *Renunțare pentru Strat(urile) Selectate*		*Digitizare*	*Salvarea Straturilor Modificate*

continues on next page

Tabelul 7.5 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
► *Salvează toate Straturile*		*Digitizare*	*Salvarea Straturilor Modificate*
► *Reface toate Straturile*		*Digitizare*	*Salvarea Straturilor Modificate*
► *Renunțare pentru toate Straturile*		*Digitizare*	*Salvarea Straturilor Modificate*
Salvare Ca…			*Creating new layers from an existing layer*
Salvare ca Fișier cu Definiții pentru Strat…			*Layer definition file*
Elimină Strat/Grup	`Ctrl+D`		
Strat(uri) Duplicate			
Setează Vizibilitatea Scării Strat(urilor)			*Visibility Scale Selector*
Setează CRS-ul statului(-urilor)	`Ctrl+Shift+C`		*Layer Coordinate Reference Systems*
Setează CRS-ul Proiectului din Strat			*Project Coordinate Reference Systems*
Proprietățile Stratului…			• *vector* • *raster* • *mesh* • *point cloud* • *vector tiles*
Filtrare…	`Ctrl+F`		*Constructorul de Interogări*
Etichetare			*Proprietățile Etichetelor*
Arată în Vizualizare			*Overview Panel*
Arată-le pe Toate în Vizualizare			*Overview Panel*
Ascunde-le pe Toate din Vizualizare			*Overview Panel*

7.1.5 Setări

Tabelul 7.6: Comenzile meniului Setări

Comanda din Meniu	Referință
Profilurile Utilizatorilor ►	*Working with User Profiles*
► *implicit*	*Working with User Profiles*
► *Deschide Dosarul Profilului Activ*	*Working with User Profiles*
► *Profill Nou…*	*Working with User Profiles*
Managerul de Stiluri…	*The Style Manager*
Proiecții Personalizate…	*Sistem de Referință și Coordonate Personalizat*
Combinații de Taste…	*Combinații de taste*
Personalizare Interfață…	*Personalizare*
Opțiuni…	*Opțiuni*

Under Linux KDE, you'll find more tools in the *Settings* menu such as *Panels* ►, *Toolbars* ► and *Toggle Full Screen Mode*.

7.1.6 Plugin-uri

Tabelul 7.7: Comenzile meniului Plugins

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Gestionaree și Instalare Pluginuri…			*Dialogul Plugin-urilor*
„ *Consola Python*	`Ctrl+Alt+P`	*Pluginuri*	*Consola Python din QGIS*

La prima startare a QGIS, nu vor fi încărcate toate plugin-urile de bază.

7.1.7 Vector

Așa arată meniul :guilabel:`Vector`atunci când toate pluginurile de bază sunt activate.

Tabelul 7.8: Comenzile meniului Vector

Comanda din Meniu	Scurtătură	Bara de Instrumer	Referință
Verificare Geometrii…			*Geometry Checker Plugin*
Verificatorul de Topologie		*Vector*	*Topology Checker Plugin*
Instrumente de Geoprocesare ►	`Alt+O + G`		
► *Zonă tampon…*			*Buffer*
► *Decupare…*			*Clip*
► *Carcasă Convexă…*			*Convex hull*
► *Diferență…*			*Difference*
► *Dizolvare…*			*Dizolvare*
► *Intersectare…*			*Intersection*
► *Diferență Simetrică…*			*Symmetrical difference*
► *Îmbinare…*			*Union*
► *Eliminare Poligoane Selectate…*			*Eliminate selected polygons*
Instrumente de Geometrie ►	`Alt+O + E`		
► *Centroizi…*			*Centroids*
► *Collectare Geometrii…*			*Collect geometries*
► *Extragere Vertecși…*			*Extract vertices*
► *Din părți Multiple în părți Singulare…*			*Multipart to singleparts*
► *Din Poligoane în Linii…*			*Polygons to lines*
► *Simplificare…*			*Simplify*
► *Verificare Validitate…*			*Check validity*
► *Triangulație Delaunay…*			*Delaunay triangulation*

continues on next page

Tabelul 7.8 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrume	Referință
► *Densificare după Număr…*			*Densify by count*
► *Adăugare Atribute Geometrice…*			*Add geometry attributes*
► *Din Linii în Poligoane…*			*Lines to polygons*
► *Poligoane Voronoi…*			*Voronoi polygons*
Instrumente de Analiză ►	`Alt+O + A`		
► *Intersectare Linii…*			*Line intersections*
► *Coordonat(e) Medii…*			*Mean coordinate(s)*
► *Statistici de Bază pentru Câmpuri…*			*Basic statistics for fields*
► *Numără Punctele din Poligon…*			*Count points in polygon*
► *Matricea Distanțelor…*			*Distance matrix*
► *Lista Valorilor Unice…*			*List unique values*
► *Analiza Celui Mai Apropiat Vecin…*			*Nearest neighbour analysis*
► *Suma Lungimilor Liniilor…*			*Sum line lengths*
Instrumente de Gestionare a Datelor ►	`Alt+O + D`		
► *Îmbinare Straturi Vectoriale…*			*Merge vector layers*
► *Reproiectare Strat…*			*Reproject layer*
► *Creare Index Spațial…*			*Create spatial index*
► *Îmbinare Atribute după Locație…*			*Join attributes by location*
► *Divizare Strat Vectorial…*			*Split vector layer*
Instrumente de Cercetare ►	`Alt+O + R`		
► *Creare Grilă…*			*Create grid*
► *Extrage Extinderea Stratului…*			*Extragere extindere din strat*
► *Puncte Aleatorii în Extindere…*			*Random points in extent*
► *Puncte Aleatorii în Poligoane…*			*Random points in polygons*
► *Puncte Aleatorii pe Linii…*			*Random points on lines*
► *Selectare după Locații…*			*Selectare după locație*
► *Select Within Distance…*			*Select within distance*
► *Puncte Aleatorii în Limitele Stratului…*			*Random points in layer bounds*
► *Puncte Aleatorii în Interiorul Poligoanelor…*			*Random points inside polygons*
► *Selecție Aleatorie…*			*Random selection*
► *Selecție Aleatorie în Interiorul Subseturilor…*			*Random selection within subsets*
► *Puncte Regulate…*			*Regular points*

În mod implicit, QGIS adaugă algoritmii de *Procesare* meniului *Vector*, grupați în submeniuri. Sunt disponibile

comenzi rapide pentru multe activități GIS vectoriale de la diferiți furnizori. Dacă toate aceste submeniuri lipsesc, activați pluginul de Procesare în *Pluginuri* ► *Gestiune și Instalare Pluginuri…*.

Rețineți că lista algoritmilor și meniul lor pot fi modificate/extinse cu oricare alt algoritm de Procesare (v. *Configuring the Processing Framework*) sau cu unele *pluginuri* externe.

7.1.8 Raster

Așa arată meniul *Raster* atunci când toate pluginurile de bază sunt activate.

Tabelul 7.9: Comenzile meniului Raster

Comanda din Meniu	Scurtătură	Bara de Instrun	Referință
Calculator raster…			*Calculatorul Raster*
Align Rasters…			*Align rasters*
Analiză ►			
► *Aspect…*			*Aspectul*
► *Umplere fărădate…*			*Fill NoData*
► *Grilă (Deplasare Medie)…*			*Grid (Moving average)*
► *Grilă (Metricile Datelor)…*			*Grid (Data metrics)*
► *Grilă (Distanța Inversă ridicată la o Putere)…*			*Grilă (Distanța inversă ridicată la o putere)*
► *Grilă (Cel Mai Apropiat Vecin)…*			*Grid (IDW with nearest neighbor searching)*
► *Aproape Negru…*			*Near black*
► *Umbrire Versanți…*			*Hillshade*
► *Proximitate (Distanța Raster)…*			*Proximity (raster distance)*
► *Rugozitate…*			*Roughness*
► *Cernere…*			*Sieve*
► *Pantă…*			*Panta*
► *Indicele Poziției Topografice (TPI)…*			*Topographic Position Index (TPI)*
► *Indicele de Rugozitate a Terenului (TRI)…*			*Terrain Ruggedness Index (TRI)*
Proiecții ►			
► *Asignare Proiecție…*			*Asignare proiecție*
► *Extragere Proiecție…*			*Extragere proiecție*
► *Warp (Reproiectare)…*			*Warp (reproiectare)*
Diverse ►			
► *Construire Raster Virtual…*			*Build virtual raster*
► *Informații Raster…*			*Raster information*
► *Îmbinare…*			*Merge*
► *Construire Vederi de Ansamblu (Piramide)…*			*Build overviews (pyramids)*

continues on next page

Tabelul 7.9 – continuare din pagina precedentă

Comanda din Meniu	Scurtătură	Bara de Instrun	Referință
► *Indexul Plăcuței…*			*Tile index*
Extragere ►			
► *Decupare Raster după o Extindere…*			*Clip raster by extent*
► *Decupare Raster după un Strat Mască…*			*Clip raster by mask layer*
► *Curbe de Nivel…*			*Curbe de nivel*
Conversie ►			
► *Din PCT în RGB…*			*PCT to RGB*
► *Poligonizare (din Raster în Vector)…*			*Polygonize (raster to vector)*
► *Rasterizare (din Vector în Raster)*			*Rasterize (vector to raster)*
► *Din RGB în PCT…*			*RGB to PCT*
► *Transformare (Conversie Format)…*			*Translate (convert format)*

În mod implicit, QGIS adaugă algoritmii de *Procesare* meniului *Raster*, grupați în submeniuri. Sunt disponibile comenzi rapide pentru multe activități GIS raster de la diferiți furnizori. Dacă toate aceste submeniuri lipsesc, activați pluginul de Procesare în *Pluginuri* ► *Gestiune și Instalare Pluginuri…*.

Rețineți că lista algoritmilor și meniul lor pot fi modificate/extinse cu oricare alt algoritm de Procesare (v. *Configuring the Processing Framework*) sau cu unele *pluginuri* externe.

7.1.9 Bază de date

Așa arată meniul *Bază de Date* atunci când toate pluginurile de bază sunt activate. Dacă nu sunt activate pluginurile pentru baze de date, nu va exista niciun meniu *Bază de Date*.

Tabelul 7.10: Comenzile meniului Bază de Date

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Editare offline…	`Alt+D` + `O`		*Offline Editing Plugin*
► *Conversie în Proiect Offline…*		*Baza de Date*	*Offline Editing Plugin*
► *Sincronizare*		*Baza de Date*	*Offline Editing Plugin*
DB Manager…		*Baza de Date*	*DB Manager Plugin*

La prima startare a QGIS, nu vor fi încărcate toate plugin-urile de bază.

7.1.10 Web

Așa arată meniul *Web* atunci când toate pluginurile de bază sunt activate. Dacă pluginurile web nu sunt activate, nu va exista niciun meniu *Web*.

Tabelul 7.11: Comenzile meniului Web

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
MetaCăutare ►	`Alt+W + M`		*MetaSearch Catalog Client*
► *Metacăutare*		*Web*	*MetaSearch Catalog Client*
► *Ajutor*			*MetaSearch Catalog Client*

La prima startare a QGIS, nu vor fi încărcate toate plugin-urile de bază.

7.1.11 Suprafață poliedrică

Meniul *Mesh* oferă instrumentele necesare pentru a manipula *straturile cu suprafețe poliedrice*. Pluginurile terțe pot adăuga elemente în acest meniu.

Tabelul 7.12: Comenzile meniului Suprafață Poliedrică

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
Calculator Suprafață Poliedrică…			*Mesh Calculator*
Reindexare Fațete și Vertecși			*Reindexing meshes*

7.1.12 Procesare

Tabelul 7.13: Comenzile meniului Procesare

Comanda din Meniu	Scurtătură	Bara de Instrumei	Referință
Bara de Instrumente	`Ctrl+Alt+T`		*The Toolbox*
Model Designer…	`Ctrl+Alt+G`		*The model designer*
Istoric…	`Ctrl+Alt+H`		*Managerul istoricului*
Vizualizare Rezultate	`Ctrl+Alt+R`		*Configurarea Aplicațiilor Externe*
Editare Pe-Poziție a Entităților			*The Processing in-place layer modifier*

La prima startare a QGIS, nu vor fi încărcate toate plugin-urile de bază.

7.1.13 Ajutor

Tabelul 7.14: Comenzile meniului Ajutor

Comanda din Meniu	Scurtătură	Bara de Instrumente	Referință
|mActionHelpContents| *Conținutul fișierului de Ajutor*	`F1`	*Ajutor*	
Documentație API			
Pluginuri ►			
Raportați o problemă			
Aveți nevoie de suport comercial?			
Pagina de Casă QGIS	`Ctrl+H`		
Verificare Versiune QGIS			
Despre			
Membri Susținători QGIS			

7.1.14 QGIS

Acest meniu este disponibil numai sub macOS și conține unele comenzi legate de sistemul de operare.

Tabelul 7.15: Comenzile meniului QGIS

Comanda din Meniu	Scurtătură
Preferințe	
Despre QGIS	
Ascundere QGIS	
Arată Tot	
Ascunde Restul	
Închidere QGIS	`Cmd+Q`

Preferințe corespunde cu *Setări* ► *Opțiuni*, *Despre QGIS* corespunde cu *Ajutor* ► *Despre* și *Închide QGIS* corespunde cu *Proiect* ► *Ieșire din QGIS* pentru alte platforme.

7.2 Panouri și Bare de Instrumente

Din meniul *Vizualizare* (sau *Setări*), puteți activa/dezactiva controalele grafice ale QGIS (*Panouri* ►) și barele de instrumente (*Toolbars* ►). Pentru a le (dez)activa pe oricare dintre ele, faceți clic dreapta pe bara de meniu sau pe bara de instrumente, apoi alegeți elementul dorit. Panourile și barele de instrumente pot fi plasate oriunde în interfața QGIS. Lista poate fi extinsă și cu activarea pluginurilor *De bază sau externe*.

7.2.1 Bare de instrumente

Barele de instrumente oferă acces la majoritatea funcțiilor din meniuri, plus instrumente suplimentare pentru interacțiunea cu harta. Fiecare element din bara de instrumente are ajutor pop-up disponibil. Treceți mouse-ul peste element pentru a citi o scurtă descriere a scopului instrumentului.

Barele de instrumente disponibile sunt:

Tabelul 7.16: Barele de instrumente QGIS

Numele barei de instrumente	Referință principală pentru instrumente
Digitizare Avansată	*Digitizare avansată*
Adnotări	*Instrumente de Adnotare*
Atribute	*Lucrul cu Tabela de Atribute*, *Instrumente generale*
Managerul Surselor de Date	*Gestiunea Surselor de Date*
Baza de Date	*DB Manager Plugin*
Digitizare	*Digitizarea unui strat vectorial existent*
GPS	*Urmărirea live a GPS-ului*
GRASS	*Integrarea GRASS GIS*
Ajutor	
Etichetă	*The Label Toolbar*
Gestionare Straturi	*Deschiderea datelor*
Navigare pe Hartă	*Exploring the map view*
Digitizare Suprafață Poiledrică	*Editing a mesh layer*
Pluginuri	*Plugin-uri*
Proiect	*Lucrul cu Fișierele Proiectelor*, *Așezarea în pagină a hărților*, *Biblioteca de Stiluri*
Algoritmi de Procesare	*Configuring the Processing Framework*
Raster	*Plugin-uri*
Selecție	*Selectare entități*
Digitizare formă	*Shape digitizing*
Acroșare	*Setting the snapping tolerance and search radius*
Vector	*Plugin-uri*
Web	*Plugin-uri*, *MetaSearch Catalog Client*

Notă: Pluginurile terțe pot extinde bara de instrumente implicită cu propriile lor instrumente sau pot avea o bară de instrumente distinctă.

Sfat: Restaurare bare de instrumente

Dacă ați ascuns accidental o bară de instrumente, o puteți readuce folosind *Vizualizare* ► *Bare de instrumente* ► (sau *Setări* ► *Bare de instrumente* ►). Dacă, dintr-un motiv oarecare, o bară de instrumente (sau orice alt control grafic) dispare complet din interfață, veți găsi sfaturi pentru a o recupera prin *restaurarea interfeței inițiale*.

7.2.2 Panouri

QGIS oferă multe panouri. Panourile sunt controale grafice speciale cu care puteți interacționa (selectarea opțiunilor, bifarea casetelor, completarea cu valori…) pentru a efectua sarcini mai complexe.

Mai jos este o listă a panourilor implicite furnizate de QGIS:

Tabelul 7.17: Panouri QGIS

Numele panoului	Scurtătură	Referință
Digitizare Avansată	`Ctrl+4`	*Panoul de Digitizare Avansată*
Navigator	`Ctrl+2`	*Panoul Navigatorului*
Navigator (2)		*Panoul Navigatorului*
Instrumente de Depanare/Dezvoltare	`F12`	*Debugging/Development Tools Panel*
Profilul Elevației		
Validarea Geometriei		*Digitizing Properties*
Informații GPS	`Ctrl+0`	*Urmărirea live a GPS-ului*
Instrumente GRASS		*Integrarea GRASS GIS*
Ordinea Straturilor	`Ctrl+9`	*Layer Order Panel*
Stilizarea Straturilor	`Ctrl+3`	*Layer Styling Panel*
Straturi	`Ctrl+1`	*Panoul Straturilor*
Jurnalul Mesajelor		*Log Messages Panel*
Vizualizare	`Ctrl+8`	*Overview Panel*
Bara Instrumentelor de Procesare		*The Toolbox*
Vizualizare Rezultate		*The Toolbox*
Opțiuni de Digitizare și Acroșare		*Setting the snapping tolerance and search radius*
Managerul Reperelor Spațiale	`Ctrl+7`	*Bookmarking extents on the map*
Statistici	`Ctrl+6`	*Statistical Summary Panel*
Controlerul Temporal		*The temporal controller panel*
Scara Plăcuței		*Seturi de plăcuțe*
Revenire/Refacere	`Ctrl+5`	*Undo/Redo Panel*
Editorul de Vertecși		*The Vertex Editor Panel*

7.3 Bara de Stare

Bara de stare oferă informații generale despre acțiunile procesate sau disponibile, deținând și instrumente pentru a gestiona vizualizarea hărții.

7.3.1 Bara locatorului

În partea stângă a barei de stare, bara locatorului, un control grafic de căutare rapidă, vă ajută să găsiți și să rulați orice funcție sau opțiune din QGIS:

1. Faceți clic în controlul grafic de tip text pentru a activa bara de căutare a locatorului sau apăsați `Ctrl+K`.
2. Introduceți un text asociat articolului pe care îl căutați (nume, etichetă, cuvânt cheie, …). În mod implicit, rezultatele sunt returnate pentru filtrele de localizare activate, dar puteți limita căutarea la un anumit domeniu prefixând textul cu prefixul *filtrelor locatorului* , de exemplu, tastând `l cad` se vor va returna doar straturile al căror nume conține `cad`.

   Filtrul poate fi selectat și printr-un dublu clic în meniul care apare la accesarea controlul grafic al locatorului.
3. Faceți clic pe un rezultat pentru a executa acțiunea corespunzătoare, în funcție de tipul articolului.

Sfat: Limitarea căutării la anumite câmp(uri) ale stratului activ

În mod implicit, o căutare cu filtrul „entităților stratului activ” (`f`) parcurge întregul tabel de atribute al stratului. Puteți limita căutarea la un anumit câmp folosind prefixul `@`. De exemplu, `f @name sal` sau `@name sal` returnează numai entitățile al căror atribut „nume” conține «sal». Completarea automată a textului este activă la scriere, iar sugestia poate fi aplicată folosind tasta `Tab`.

Un control mai avansat asupra câmpurilor interogate este posibil din fila stratului *Câmpurilor*. Citiți *Fields Properties* pentru detalii.

Căutarea este gestionată folosind fire, astfel încât rezultatele să devină întotdeauna disponibile cât mai repede posibil, chiar dacă sunt instalate filtre de căutare lente. Ele apar, de asemenea, imediat ce sunt întâlnite de un filtru, ceea ce înseamnă că, de exemplu, un filtru de căutare de fișiere va afișa rezultatele unul câte unul, pe măsură ce arborele de fișiere este scanat. Acest lucru asigură că interfața de utilizare este întotdeauna receptivă, chiar dacă este prezent un filtru de căutare foarte lent (de exemplu, unul care utilizează un serviciu online).

Notă: Instrumentul de localizare Nominatim se poate comporta diferit (fără căutare de completare automată, întârziere de preluare a rezultatelor, …) conform *politicii de utilizare a OpenStreetMap Nominatim <https://operations.osmfoundation.org/policies/nominatim/>*

Sfat: Acces rapid la configurațiile locatorului

Faceți clic pe pictograma din interiorul controlului grafic de localizare de pe bara de stare, pentru a afișa lista de filtre care se pot utiliza și o intrare de *Configurare*, care deschide fila *Locator* a meniului *Setări* ► *Opțiuni…*.

7.3.2 Raportarea acțiunilor

În zona de lângă bara locatorului va fi afișat, atunci când este necesar, un rezumat al acțiunilor pe care le-ați efectuat (cum ar fi selectarea entităților dintr-un strat, eliminarea stratului, distanța de deplasare și direcția), sau va fi prezentată o descriere lungă a instrumentului peste care treceți (nu este disponibil pentru toate instrumentele).

În cazul operațiunilor de durată, cum ar fi culegerea de statistici din straturile raster, executarea algoritmilor de procesare sau randarea mai multor straturi în vizualizarea hărții, se afișează o bară de progres în bara de stare.

7.3.3 Controlul canevasului hărții

|urmărirea| Opțiunea *Coordonatelor* arată poziția curentă a mouse-ului, urmărindu-l în timp ce se deplasează pe hartă. Puteți seta unitățile (și precizia) în fila *Proiectului* ► *Proprietăți…* ► *Generalități*. Faceți clic pe butonul mic din stânga casetei de text pentru a comuta între opțiunile Coordonate și *Extinderi*, care afișează, în unități de hartă, coordonatele curente ale colțurilor din stânga-jos și din dreapta-sus ale vizualizării hărții.

Lângă coordonate veți găsi afișată și *Scara*. Acesta prezintă scara vizualizării hărții. Există un selector de scară, care vă permite să alegeți între *scări predefinite și personalizate*.

În partea dreaptă a scării, apăsați butonul pentru a bloca scara și pentru a utiliza lupa pentru a mări sau micșora. Lupa vă permite să măriți o hartă fără a modifica scara hărții, ceea ce face mai ușoară ajustarea cu precizie a pozițiilor etichetelor și simbolurilor. Nivelul de mărire este exprimat ca procentaj. Dacă *Lupă* are un nivel de 100%, atunci harta curentă nu este mărită, adică este redată la o scară precisă în raport cu rezoluția monitorului (DPI). O valoare implicită de mărire poate fi definită în *Setări* ► *Opțiuni* ► *Randare* ► *Comportamentul randării*, care este foarte utilă ecranelor de înaltă rezoluție pentru a mări simbolurile mici. În plus, o setare din *Setări* ► *Opțiuni* ► *Canevas & Legendă* ► *DPI* stabilește dacă QGIS respectă DPI-ul fizic al fiecărui monitor sau folosește DPI-ul logic al sistemului general.

În partea dreaptă a lupei, puteți defini o rotație curentă în sensul acelor de ceasornic, pentru vizualizarea hărții dvs. în grade.

În partea dreaptă a bării de stare, caseta *Randării* poate fi folosită pentru a suspenda temporar redesenarea hărții (v. secțiunea *Controlling map rendering*).

În dreapta funcției *Render*, veți găsi butonul *EPSG:code* care arată CRS-ul proiectului curent. Făcând clic pe acesta se deschide fereastra *Proprietăților proiectului* și vă permite să reproiectați vizualizarea hărții sau să ajustați oricare altă proprietate a proiectului.

Sfat: Calculul Scării Corecte a Canevasului Hărții

Când porniți QGIS, CRS-ul implicit este `WGS 84 (EPSG 4326)`, iar unitățile sunt în grade. Aceasta înseamnă că aplicația QGIS va interpreta orice coordonată din stratul dvs. în grade. Pentru a obține valori corecte de scară, puteți fie să modificați manual această setare în fila *Generalități* din *Proiect* ► *Proprietăți...* (de exemplu, în metri), sau puteți utiliza Pictograma EPSG:code de mai sus. În ultimul caz, unitățile sunt setate la ceea ce specifică proiecția proiectului (de exemplu, `+units=us-ft`).

Rețineți că alegerea CRS la pornire poate fi setată în *Setări* ► *Opțiuni* ► *Gestionare CRS*.

7.3.4 Jurnalizarea

Butonul Mesaje de lângă acesta deschide *Panoul Mesajelor de Jurnalizare*, care conține informații despre procesele subiacente (pornire QGIS, încărcare pluginuri, instrumente de procesare…)

CAPITOLUL 8

Panoul Navigatorului

- *Resources that can be opened / run from the Browser*
- *Browser panel top-level entries*
  - *Favorites*
  - *Semne de Carte Spațiale*
  - *Project Home*
  - *Drives and file system*
  - *Database entries*
  - *Tiles and Web Services*
- *Resources*

Panoul Navigatorului QGIS este un instrument excelent pentru navigarea, căutarea, inspectarea, copierea și încărcarea resurselor QGIS. Doar resursele pe care QGIS știe să le gestioneze sunt afișate în browser.

Folosind panoul Navigatorului puteți localiza, inspecta și adăuga date, așa cum este descris în *Panoul Navigatorului*. În plus, panoul Navigatorului acceptă glisarea și plasarea multor resurse QGIS, cum ar fi fișierele de proiect, scripturile Python, scripturile de procesare și modelele de procesare.

Scripturile Python, scripturile și modelele Processing pot fi, de asemenea, deschise pentru editare într-un editor extern și în modelatorul grafic.

Puteți trage și plasa straturi din panoul *Straturi* în panoul *Navigatorului*, de exemplu într-un GeoPackage sau o bază de date PostGIS.

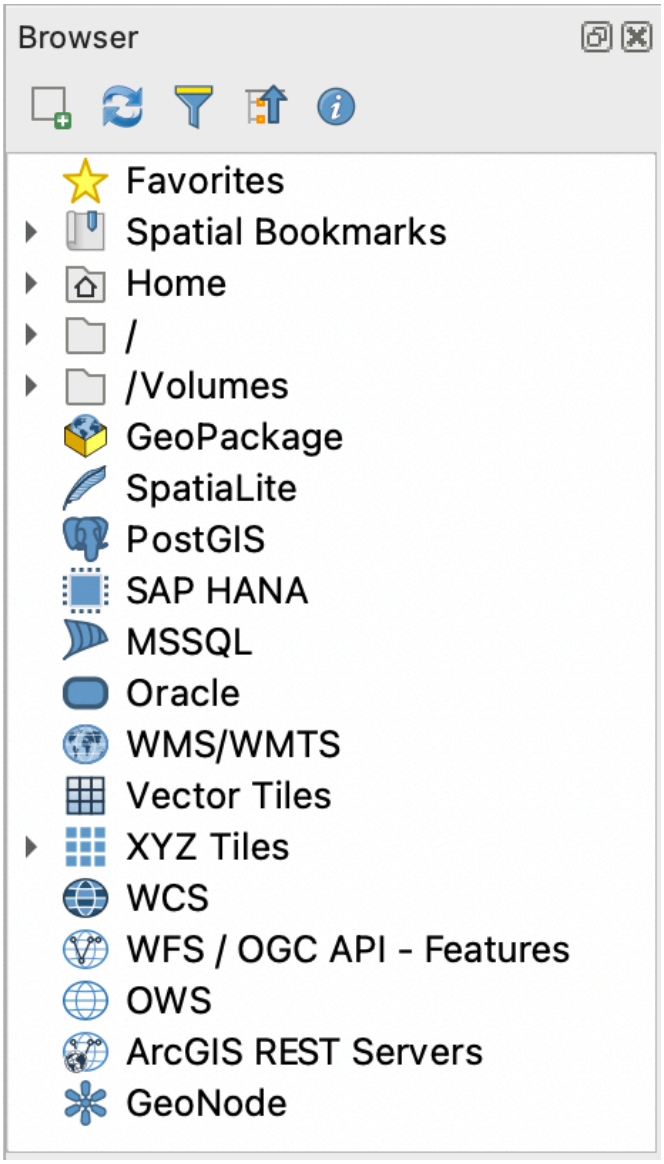


Fig. 8.1: Panoul Navigatorului

Panoul navigatorului (Fig. 8.1) are o ierarhie extensibilă, cu câteva intrări fixe de nivel superior care organizează resursele gestionate. Intrările nodurilor sunt extinse făcând clic pe din stânga numelui intrării. Orice ramură este restrânsă făcând clic pe . Butonul |colapseTree| Restrânge Tot restrânge toate intrările de nivel superior.

În *Setări* ► *Interfață Personalizabilă* este posibilă dezactivarea resurselor. Dacă, de exemplu, nu doriți să afișați scripturi Python în browser, puteți debifa intrarea *Navigator* ► *py*, iar dacă doriți să scăpați de folderul de casă din navigator, puteți debifați intrarea *Navigator* ► *special:Home*.

(Filtrarea Navigării) poate fi folosită pentru căutarea pe baza numelor intrărilor (atât intrările subordonate, cât și intrările nodurilor din ierarhie). Folosind meniul derulant Opțiuni de lângă câmpul de text al filtrului, puteți

- efectua *Căutare Senzitivă*
- seta *Sintaxa modelului de filtrare* la
  - *Normal*
  - *Metacaracter(e)*

– *Expresii Regulate*

Controlul grafic al Proprietăților, care arată informații utile despre unele intrări/resurse, poate fi activat/dezactivat folosind butonul |metadate| Activează/dezactivează controlul grafic al proprietăților. Când este activat, se deschide în partea de jos a panoului navigatorului, așa cum se arată în Fig. 8.2.

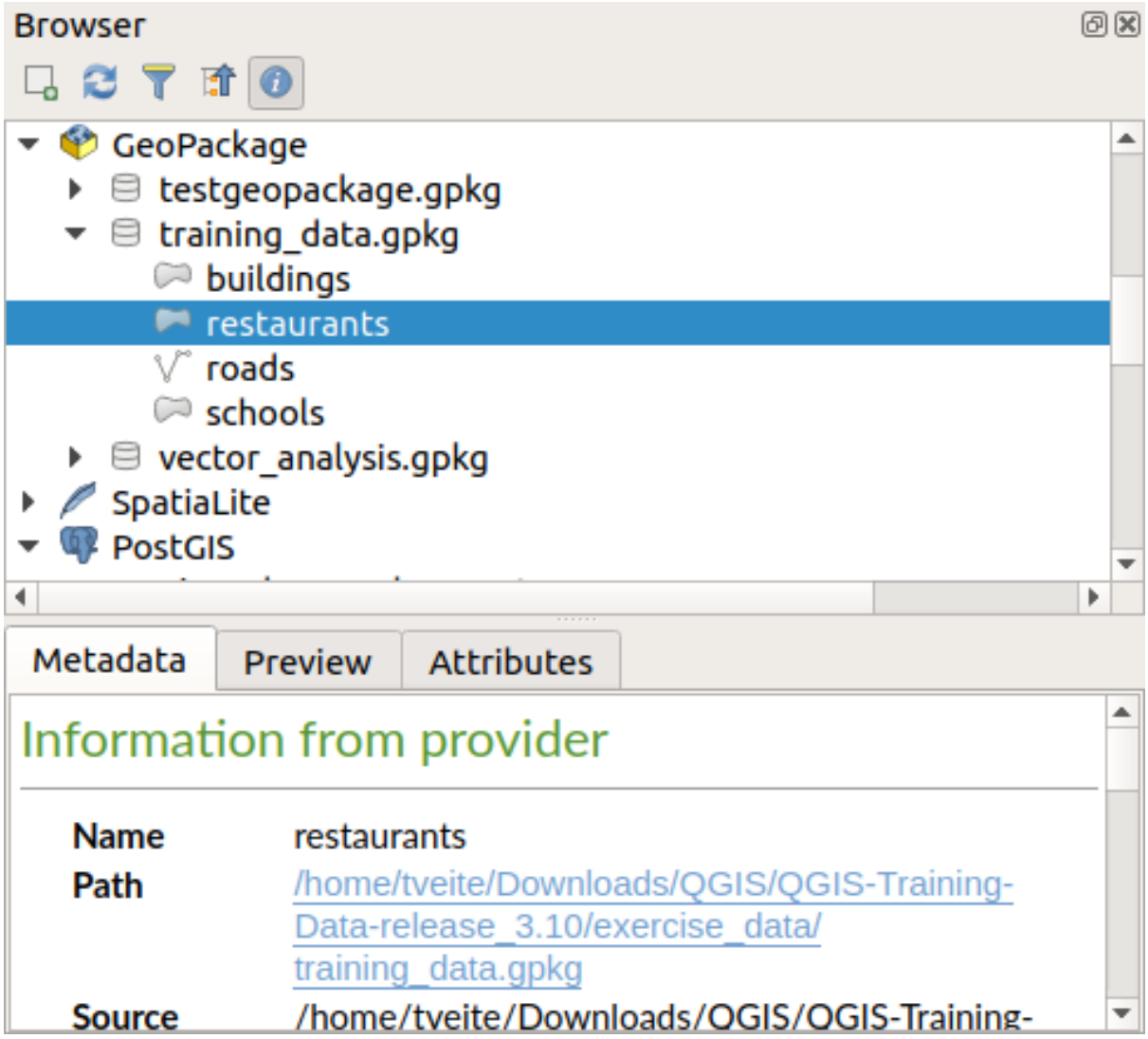


Fig. 8.2: Controlul grafic al proprietăților

A second browser panel can be opened by activating the *Browser (2)* panel in *View* ► *Panels*. Having two browser panels can be useful when copying layers between resources that are locationed deep down in different branches of the browser hierarchy.

8.1 Resources that can be opened / run from the Browser

A lot can be accomplished in the Browser panel

- Add vector, raster and mesh layers to your map by double-clicking, dragging onto the map canvas or clicking the Add Selected Layers button (after selecting layers)
- Run Python scripts (including Processing algorithms) by double-clicking or dragging onto the map canvas
- Run models by double-clicking or dragging onto the map canvas
- *Extract Symbols…* from QGIS Project files using the context menu
- Open files with their default applications (*Open <file type> Externally…* in the context menu). Examples: HTML files, spreadsheets, images, PDFs, text files, …
- Copy entries
- Rename and delete (multiple) layers (context menu: *Manage* ►)
- Open a file explorer window and directly select the file *Show in Files*

Resource specific actions are listed for the different resource groups sorted under the top-level entries listed below.

8.2 Browser panel top-level entries

8.2.1 Favorites

Often used file system locations can be tagged as favorites. The ones you have tagged will appear here.

In addition to the operations described under *Home*, the context menu allows you to *Rename Favorite…* and *Remove Favorite*.

8.2.2 Semne de Carte Spațiale

This is where you will find your spatial bookmarks, organised into *Project Bookmarks* and *User Bookmarks*.

From the top level context menu, you can create a bookmark (*New Spatial Bookmark…*), *Show the Spatial Bookmark Manager*, *Import Spatial Bookmarks…* and *Export Spatial Bookmarks…*.

For bookmark group entries you can *Export Spatial Bookmarks…*, create a bookmark (*New Spatial Bookmark…*), *Rename Bookmark Group*, and *Delete Bookmark Group*.

For bookmark entries you can *Zoom to Bookmark*, *Edit Spatial Bookmark…* and *Delete Spatial Bookmark*

8.2.3 Project Home

Available once the project file has been saved, the *Project home* entry is a folder containing data and other contents (scripts, models, text, …) that may be used within the current project. Displayed in the *Browser* panel, it allows you to quickly access data and other files of the project.

It defaults to the project file folder but can be changed through the *Project* ► *Properties…* ► *General* ► *Project home* option, or by right-clicking on the *Project Home* item of the Browser panel and selecting *Set project home…*. Customizing that folder is especially useful in contexts where QGIS projects are not stored in the root folder of an organisational «project», along with datasets.

8.2.4 Drives and file system

The next items of the *Browser* panel depend on the OS in use and concern the top level entries of its file system.

They are mainly:

- The *Home* folder, pointing to the current user home folder
- on Unix-based machines, the root `/` folder
- the connected drives, either local or network. Depending on the OS, they are directly listed (eg, `C:\`, `D:\`) or through the `/Volumes` entry.

From the contextual menu of each of these folders or drives, you can:

- refresh the contents
- create a *New* ► subitem that is a *Directory*, *GeoPackage* or ESRI *Shapefile* format dataset
- hide the directory (*Hide from Browser*)
- *Set color*: customize the folder icon color, aiding in rapid browser navigation of complex folder structures
- enable *Scanning*:
  - *Monitor for changes*: allows to manually control whether a particular directory should be monitored and automatically updated. This setting applies to the selected directory and all subdirectories. This means that you can manually opt-in to monitoring of network drives if you know there's no issue, or manually opt-out of monitoring of large directories which you don't want monitored for other reasons. By default, remote or network drives are not automatically monitored.
  - *Fast scan this directory*
- open the directory in your file manager (*Open Directory…*)
- open the directory in a terminal window (*Open in Terminal…*)
- inspect the *Properties…* or the parent *Directory Properties…*

8.2.5 Database entries

Depending on your OS and installed drivers, you might have access to different database types to use in QGIS. Below are listed the different entries of contextual menu at each level of the dataset tree.

Level	Context menu	Type of database					
		GeoPackage ([1])	SpatiaLite	PostGIS	SAP HANA	MS SQL Server	Oracle
Top menu	Create a *New Connection…* to an existing database	☑	☑	☑	☑	☑	☑
	Create Database…	☑	☑				
	Save Connections… details to a file			☑	☑	☑	

continues on next page

Tabelul 8.1 – continuare din pagina precedentă

	Load Connections.			✓	✓	✓	
Connection / Database	*Refresh* a connection			✓	✓	✓	
	Edit Connection… settings			✓	✓	✓	
	Remove Connection	✓	✓	✓	✓	✓	
	Delete <database_n	✓	✓				
	Compact Database (VACUUM)	✓					
	Create a *New Schema…*			✓	✓	✓	
	Create a *New Table…*	✓	✓	✓	✓		
	Execute SQL… query	✓	✓	✓	✓		
Schema	*Refresh* a schema			✓	✓	✓	
	Schema Operations ► Rename Schema…			✓	✓	✓	
	Schema Operations ► Delete Schema…			✓	✓	✓	
	Create a *New Table…*			✓	✓		
	Execute SQL… query			✓	✓		
Table / Layer	*Table Operations ► Rename Table…*			✓	✓	✓	
	Table Operations ► Truncate Table…			✓		✓	
	Execute SQL… query	✓	✓	✓			

continues on next page

Tabelul 8.1 – continuare din pagina precedentă

	Export Layer ► To file…	☑	☑	☑	☑	☑
	Manage ► Rename Layer <layer_name	☑	☑			
	Manage ► Delete Layer <layer_name	☑	☑	☑	☑	☑
	Manage ► Delete Selected Layers	☑	☑	☑	☑	☑
	Manage ► Add Layer to Project	☑	☑	☑	☑	☑
	Manage ► Add Selected Layers to Project	☑	☑	☑	☑	☑
	Open *Layer Properties…* dialog	☑	☑	☑	☑	☑
	Open *File Properties…* dialog	☑				
Fields	*Add New Field…*	☑	☑	☑	☑	
Field	*Set Alias…*	☑				
	Set Comment…	☑		☑		
	Delete Field…	☑	☑	☑	☑	

[1] The different entries may also be available for some GDAL supported vector file formats such as ESRI File Geodatabase, FlatGeobuf, GeoParquet, NetCDF, … when compatible.

8.2.6 Tiles and Web Services

Level	Context menu	Type of services						
		WMS / WMTS	Vector Tiles	XYZ Tiles	WCS	WFS / OGC API - Features	ArcGIS REST Servers	Scene
Top menu	Create a *New Connection…*	☑		☑	☑	☑	☑	☑
	Create a *New Generic Connection…*		☑					
	Create a *New ArcGIS Vector Tile Service Connection…*		☑					
	Save Connections… details to a file	☑	☑	☑	☑	☑	☑	☑
	Load Connections…	☑	☑	☑	☑	☑	☑	☑
Connec	*Refresh* connection	☑		☑	☑	☑	☑	
	Edit… connection settings	☑	☑	☑	☑	☑	☑	☑
	Delete connection	☑	☑	☑	☑	☑	☑	☑
	View Service Info in Web browser						☑	
Table / Layer	*Export Layer ► To File…*	☑		☑	☑	☑	☑	
	Add layer to Project	☑	☑	☑	☑	☑	☑	☑
	Open *Layer properties…* dialog	☑	☑	☑	☑	☑	☑	☑
	View Service Info in Web browser						☑	

8.3 Resources

- Project files. The context menu for QGIS project files allows you to:
  - open it (*Open Project*)
  - extract symbols (*Extract Symbols…*) - opens the style manager that allows you to export symbols to an XML file, add symbols to the default style or export as PNG or SVG.
  - inspect properties (*File Properties…*)

  You can expand the project file to see its layers. The context menu of a layer offers the same actions as elsewhere in the browser.
- QGIS Layer Definition files (QLR). The following actions are available from the context menu:
  - export it (*Export Layer ► To file*)
  - add it to the project (*Add Layer to Project*)
  - inspect properties (*Layer Properties…*)
- Processing models (.model3). The following actions are available from the context menu:

  - *Run Model…*)
  - *Edit Model…*)
- QGIS print composer templates (QPT). The following action is available from the context menu:
  - (*New Layout from Template*)
- Python scripts (.py). The following actions are available from the context menu:
  - (*Run script…*)
  - (*Open in External Editor*)
- Recognized raster formats. The following actions are available from the context menu:
  - delete it (*Delete File <dataset name>*)
  - export it (*Export Layer ► To file*)
  - add it to the project (*Add Layer to Project*)
  - inspect properties (*Layer Properties…*, *File Properties…*)

  For some formats you can also *Open <file type> Externally…*
- Recognized vector formats. The following actions are available from the context menu:
  - delete it (*Delete File <dataset name>*)
  - export it (*Export Layer ► To file*)
  - add it to the project (*Add Layer to Project*)
  - inspect properties (*Layer Properties…*, *File Properties…*)

  For some formats you can also *Open <file type> Externally…*

CAPITOLUL 9

Configurare QGIS

QGIS dispune de numeroase configurări Prin meniul *Settings*, oferă diferite instrumente pentru:

- *Style Manager…*: create and manage *symbols, styles and color ramps*.
- *Custom Projections…*: create your own *coordinate reference systems*.
- *Keyboard Shortcuts…*: define your own set of *keyboard shortcuts*. Also, they can be overridden during each QGIS session by the *project properties* (accessible under *Project* menu).
- *Interface Customization…*: configure the *application interface*, hiding dialogs or tools you may not need.
- *Options…*: set global *options* to apply in different areas of the software. These preferences are saved in the active *User profile* settings and applied by default whenever you open a new project with this profile.

9.1 Opţiuni

Some basic options for QGIS can be selected using the *Options* dialog. Select the menu option *Settings* ► *Options*. You can modify the options according to your needs. Some of the changes may require a restart of QGIS before they will be effective.

The tabs where you can customize your options are described below.

Notă: Plugins can embed their settings within the Options dialog

While only Core settings are presented below, note that this list can be extended by *installed plugins* implementing their own options into the standard Options dialog. This avoids each plugin having their own config dialog with extra menu items just for them…

9.1.1 Setări Generale

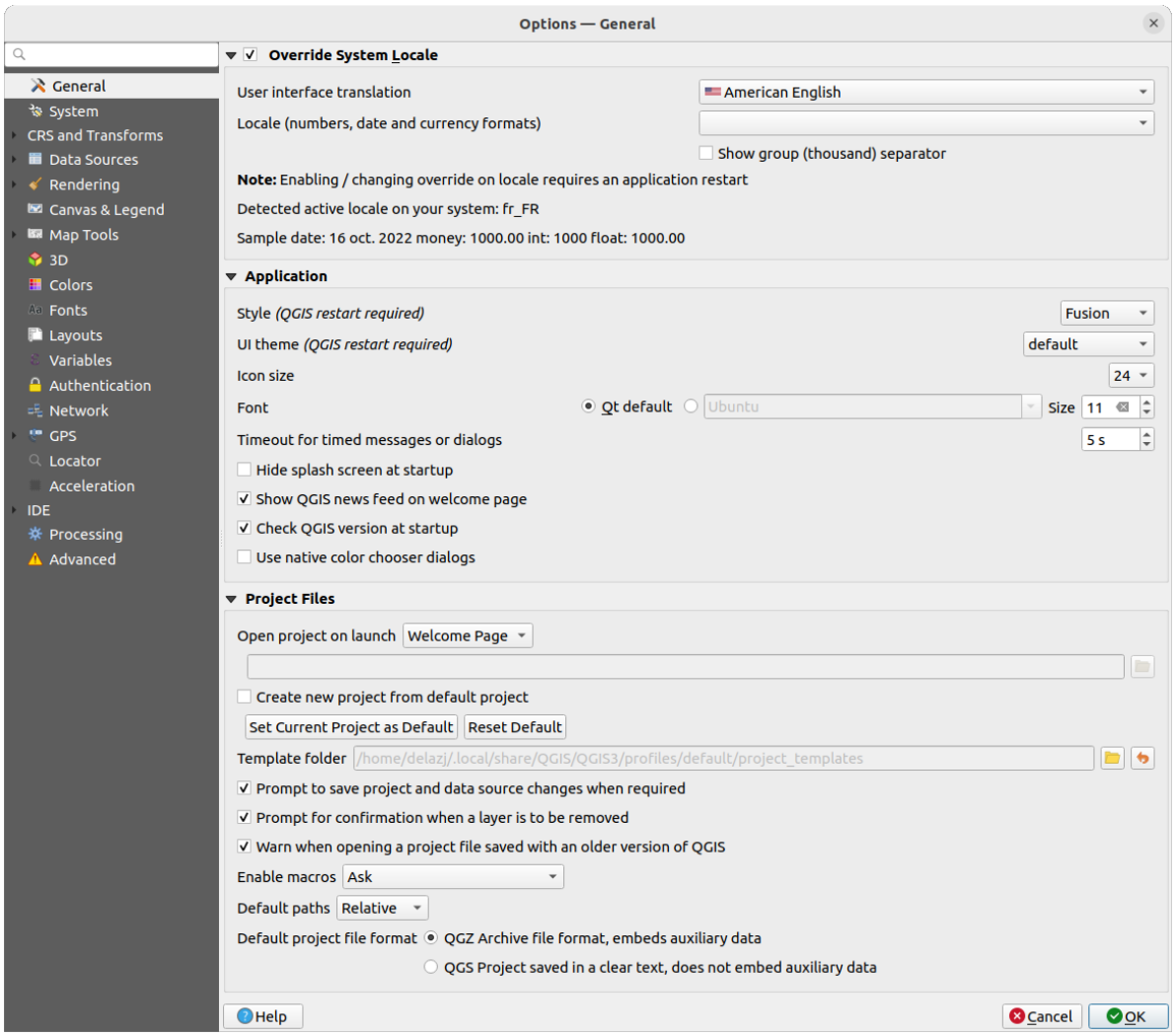


Fig. 9.1: Setări Generale

Suprascrie Limba Sistemului

By default, QGIS relies on your Operating System configuration to set language and manipulate numerical values. Enabling this group allows you to customize the behavior.

- Select from *User interface translation* the language to apply to the GUI
- Select in *Locale (number, date and currency formats)* the system on which date and numeric values should be input and rendered
- *Show group (thousand) separator*

A summary of the selected settings and how they would be interpreted is displayed at the bottom of the frame.

Aplicație

- Select the *Style (QGIS restart required)* ie, the widgets look and placement in dialogs. Possible values depend on your Operating System.
- Define the *UI theme (QGIS restart required)* . It can be «default», «Night Mapping», or «Blend of Gray»
- Define the *Icon size*

- Define the *Font* and its *Size*. The font can be *Qt default* or a user-defined one
- Change the *Timeout for timed messages or dialogs*
- *Hide splash screen at startup*
- *Show QGIS news feed on welcome page*: displays a curated QGIS news feed on the welcome page, giving you a direct way to be aware of project news (user/developer meetings date and summary, community surveys, releases announcements, various tips…)
- *Check QGIS version at startup* to keep you informed if a newer version is released
- *Use native color chooser dialogs* (see *Selectorul de Culoare*)

Fișierele proiectului

- *Deschidere proiect la pornire*
  - «Welcome Page» (default): can display the „News" feed, the project template(s) and the most recent projects (with thumbnails) of the *user profile*. No project is opened by default.
  - «New»: opens a new project, based on the default template
  - «Most recent»: reopens the last saved project
  - and «Specific»: opens a particular project. Use the … button to define the project to use by default.
- *Create new project from default project*. You have the possibility to press on *Set current project as default* or on *Reset default*. You can browse through your files and define a directory where you find your user-defined project templates. This will be added to *Project* ► *New From Template*. If you first activate *Create new project from default project* and then save a project in the project templates folder.
- *Prompt to save project and data source changes when required* to avoid losing changes you made.
- *Se solicită confirmarea atunci când un strat va fi eliminat*
- *Warn when opening a project file saved with an older version of QGIS*. You can always open projects created with older version of QGIS but once the project is saved, trying to open with older release may fail because of features not available in that version.
- *Enable macros* . This option was created to handle macros that are written to perform an action on project events. You can choose between «Never», «Ask», «For this session only» and «Always (not recommended)».
- *Default paths*: defines whether paths to files and layers used in new projects are stored as «Absolute» or «Relative» to the project file. This setting can be overwritten at the project level.
- *Default project file format*
  - *QGZ Archive file format, embeds auxiliary data* (see *auxiliary data*)
  - *QGS Project saved in a clear text, does not embed auxiliary data*: the auxiliary data is stored in a separate `.qgd` file along with the project file.

9.1.2 System Settings

SVG paths

Add or Remove *Path(s) to search for Scalable Vector Graphic (SVG) symbols*. These SVG files are then available to symbolize or label the features or decorate your map composition.

Also read *Remote or embedded file selector* for different ways to refer to svg files in a QGIS path.

Căile plugin-urilor

Add or Remove *Path(s) to search for additional C++ plugin libraries*.

Documentation paths

Add or Remove *Documentation Path(s)* to use for QGIS help. By default, a link to the official online User Manual corresponding to the version being used is added. You can however add other links and prioritize them from top to bottom: each time you click on a *Help* button in a dialog, the topmost link is checked and if no corresponding page is found, the next one is tried, and so on.

Notă: Documentation is versioned and translated only for QGIS Long Term Releases (LTR), meaning that if you are running a regular release (eg, QGIS 3.0), the help button will by default open the next LTR manual page (ie. 3.4 LTR), which may contain description of features in newer releases (3.2 and 3.4). If no LTR documentation is available then the *testing* doc, with features from newer and development versions, is used.

Setări

It helps you *Reset user interface to default settings (restart required)* if you made any *customization*.

Mediu

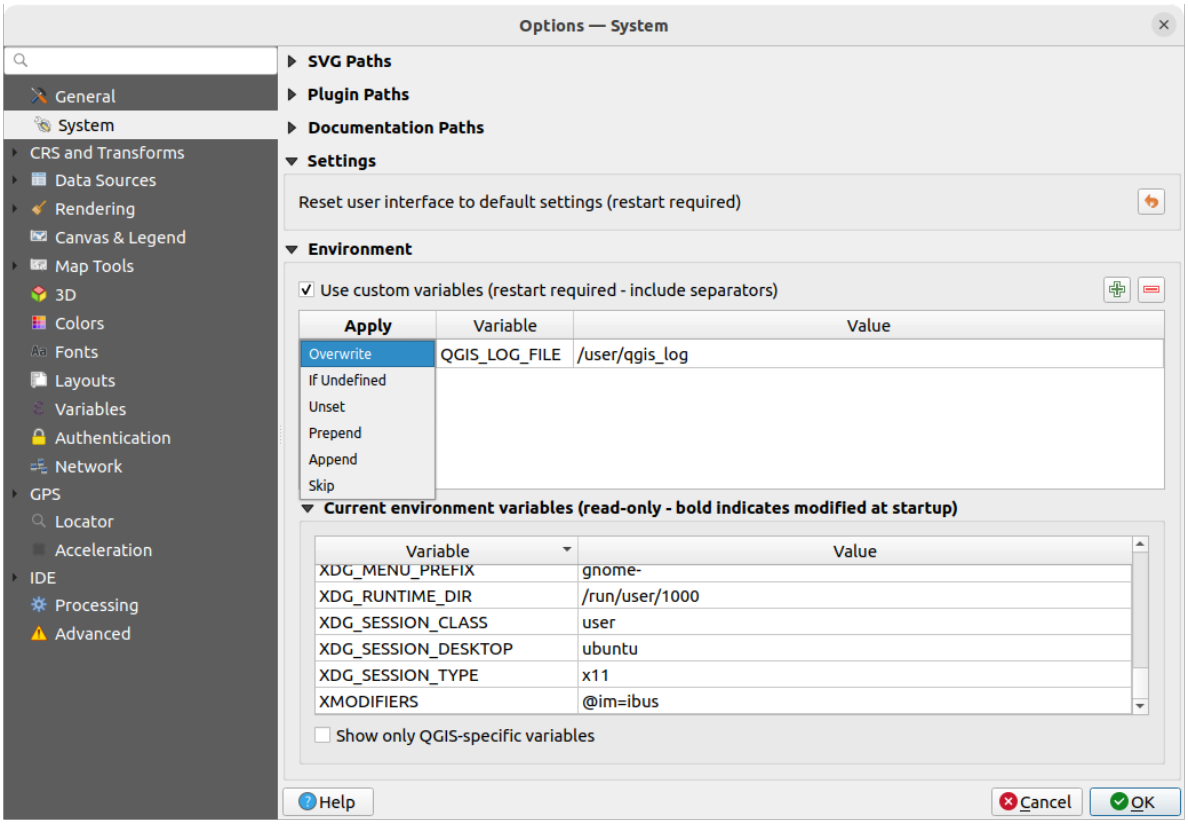


Fig. 9.2: System environment variables

System environment variables can be viewed, and many configured, in the **Environment** group. This is useful for

platforms, such as Mac, where a GUI application does not necessarily inherit the user's shell environment. It's also useful for setting and viewing environment variables for the external tool sets controlled by the Processing toolbox (e.g., SAGA, GRASS), and for turning on debugging output for specific sections of the source code.

Check *Use custom variables (restart required - include separators)* and you can Add and Remove environment variables. For each new item, you can configure a *Variable* name, its *Value* and the *Apply* method to use, among which:

- *Overwrite*: replace any preexisting value of the variable
- *If undefined*: use this value for the variable if not already defined at a higher level (e.g. OS or application levels)
- *Unset*: remove the variable from the environment (the *Value* parameter is not used)
- *Prepend*: prepend the value to the preexisting value of the variable
- *Append*: append the value to the preexisting value of the variable
- *Skip*: the item is kept in the list for future reference but unused

Already defined environment variables are displayed in *Current environment variables*, and it's possible to filter them by activating *Show only QGIS-specific variables*.

9.1.3 User Profiles Settings

Notă: For more information on how to manage user profiles, please read the dedicated section at *Working with User Profiles*.

9.1.4 CRS and Transforms Settings

Notă: For more information on how QGIS handles layer projection, please read the dedicated section at *Lucrul cu Proiecții*.

CRS Handling

In the *CRS Handling* tab you can configure which CRS will be used for a new project or layer.

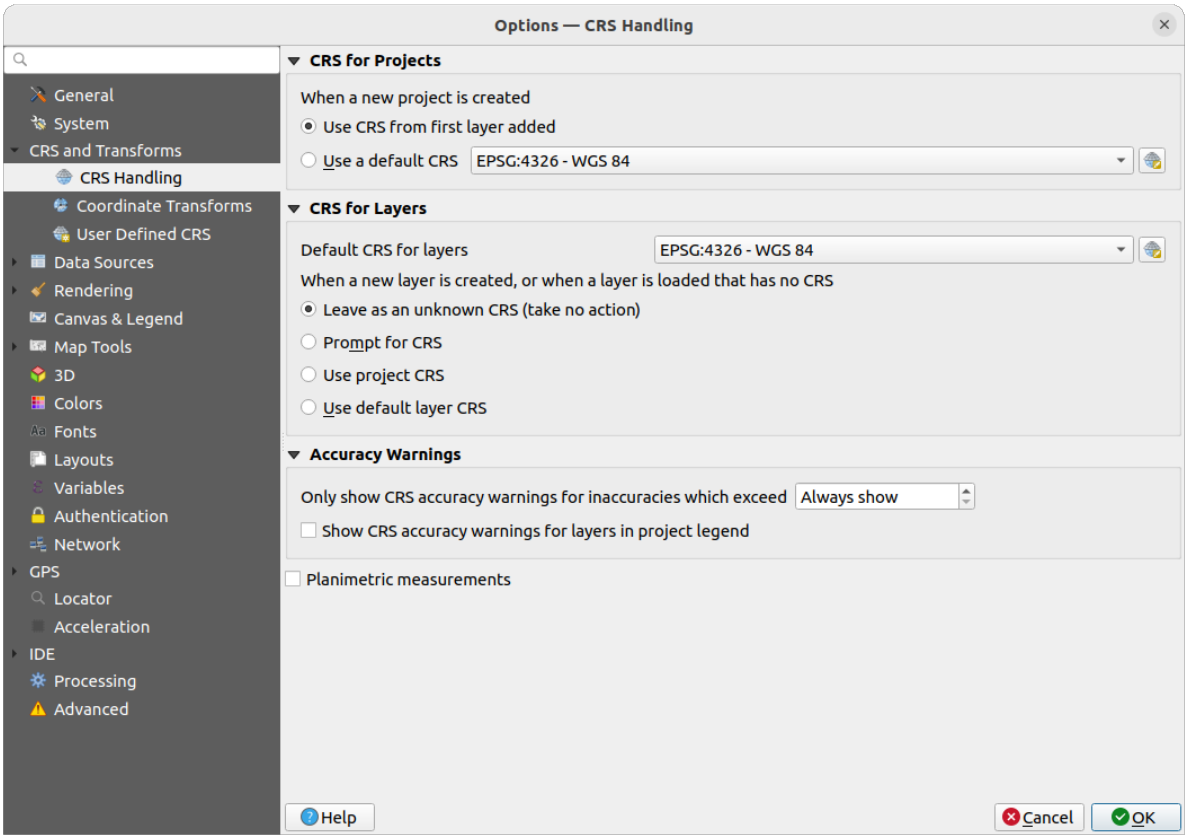


Fig. 9.3: CRS Settings

CRS for Projects

There is an option to automatically set new project's CRS:

- *Use CRS from first layer added*: the CRS of the project will be set to the CRS of the first layer loaded into it
- *Use a default CRS*: a preselected CRS is applied by default to any new project and is left unchanged when adding layers to the project.

The choice will be saved for use in subsequent QGIS sessions. The Coordinate Reference System of the project can still be overridden from the *Project* ► *Properties…* ► *CRS* tab.

CRS for Layers

Default CRS for layers: select a default CRS to use when you create a layer

You can also define the action to take when a new layer is created, or when a layer without a CRS is loaded.

- *Leave as unknown CRS (take no action)*
- *Prompt for CRS*
- *Utilizează CRS-ul proiectului*
- *Use default layer CRS*

Accuracy Warnings

Only show CRS accuracy warnings for inaccuracies which exceed a given distance: occurs when you are explicitly creating or modifying a dataset and select a CRS based on a datum ensemble with lower accuracy. The default is to `Always show` the warning if any inaccuracy. Requires a QGIS version using at least PROJ 8.0.

☐ *Show CRS accuracy warning for layers in project legend*: If checked, any layer with a CRS with accuracy issues (i.e. a dynamic crs with no coordinate epoch available, or a CRS based on a datum ensemble with inherent inaccuracy exceeding the user-set limit) will have the warning icon in the *Layers* panel reflecting that it is a low-accuracy layer.

This is designed for use in engineering, BIM, asset management, and other fields where inaccuracies of meter/submeter level are potentially very dangerous or expensive!

☐ *Planimetric measurements*: sets the default for the *planimetric measurements* property for newly created projects.

Coordinate Transforms

The *Coordinate Transforms* tab helps you set coordinate transformations and operations to apply when loading a layer to a project or reprojecting a layer.

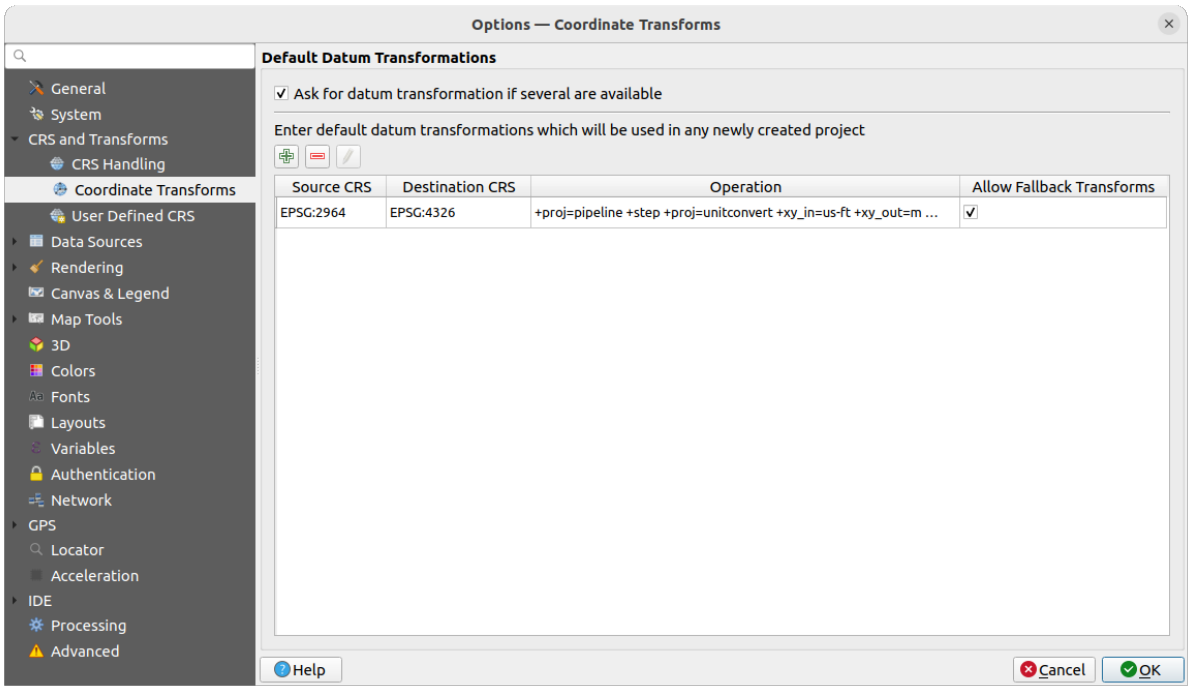


Fig. 9.4: Transformations settings

Default Datum Transformations

Here you can control whether reprojecting layers to another CRS should be:

- automatically processed using QGIS default transformations settings;
- and/or more controlled by you with custom preferences such as:
  - ☑ *Ask for datum transformation if several are available*
  - a predefined list of datum transformations to apply by default. See *Datum Transformations* for more details.

You can Add, Remove or Edit transformations, which will be used in any newly created project.

User Defined CRS

The *User Defined CRS* tab helps you to define a custom CRS which must conform to a WKT or Proj string format.

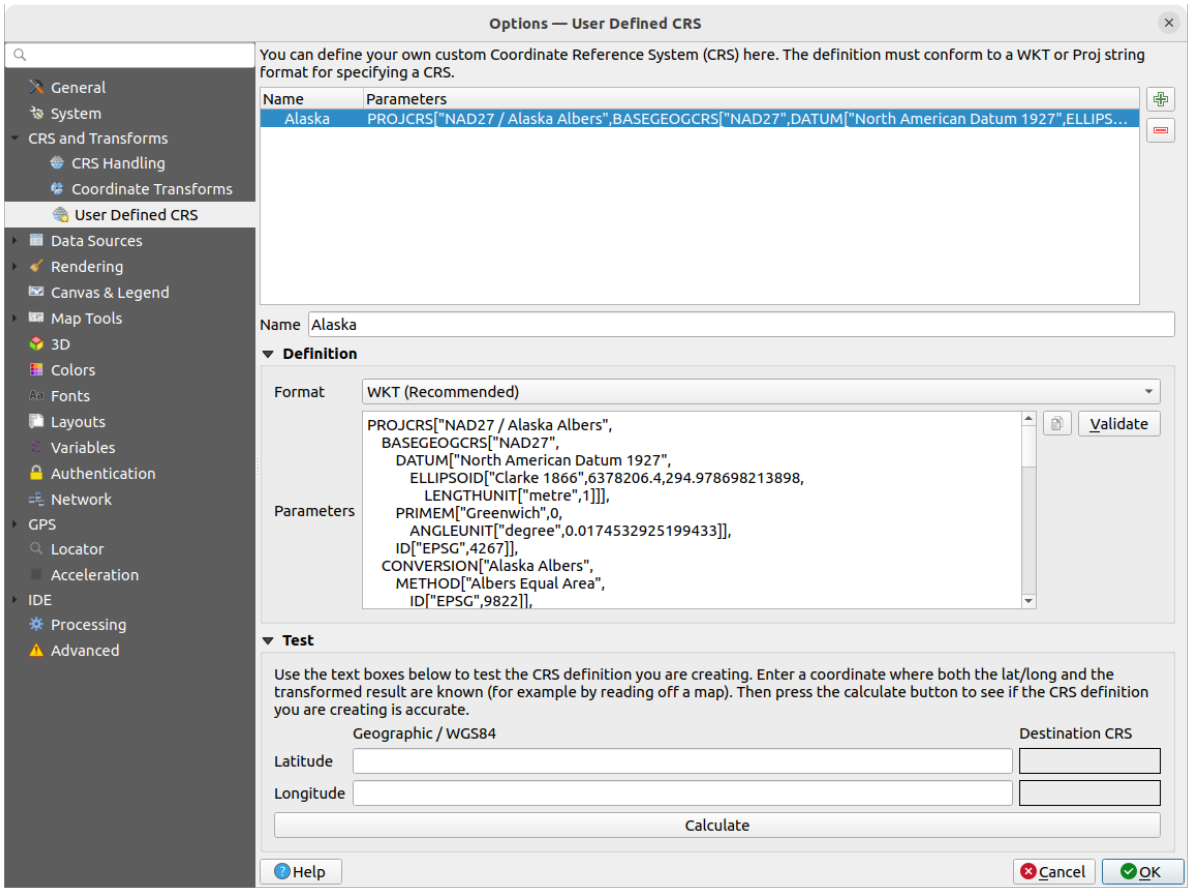


Fig. 9.5: User Defined CRS

Set a *Name* and use Add new CRS. If you want to delete an existing one you can use Remove CRS.

Definition

- ***Format***
  - WKT (Recomandat)
  - Șir Proj (Învechit - Nerecomandat)
- ***Parametri***
  - Copy parameters from an existing CRS.
  - *Validate* tests if your expression is correct.

Test

Here you can test your created CRS definition by Latitude and Longitude. Use a known coordinate to control if your definition is accurate.

9.1.5 Data Sources settings

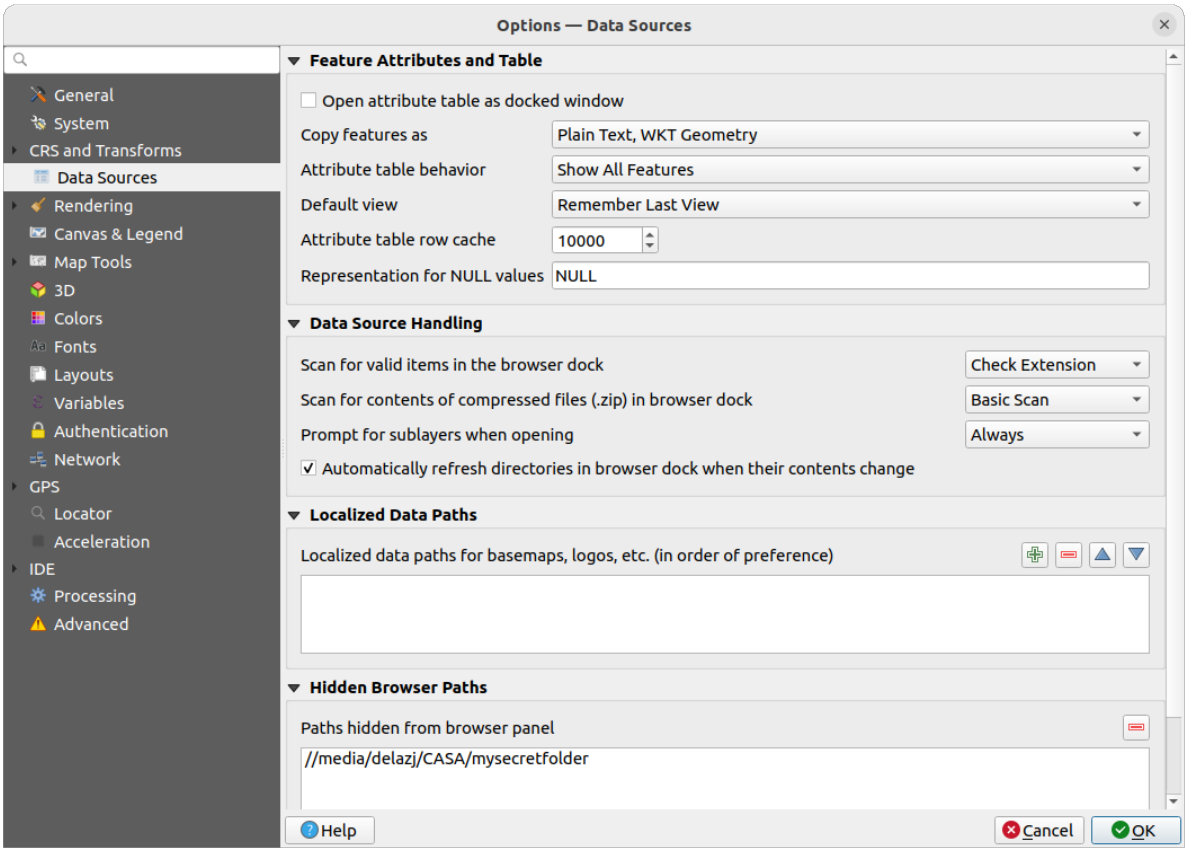


Fig. 9.6: Data Sources settings

Atributele şi tabela entităţii

- *Open attribute table as docked window*
- *Copy features as* «Plain text, no geometry», «Plain text, WKT geometry», or «GeoJSON» when pasting features in other applications.
- *Attribute table behavior* : set filter on the attribute table at the opening. There are three possibilities: «Show all features», «Show selected features» and «Show features visible on map».
- *Default view*: define the view mode of the attribute table at every opening. It can be «Remember last view», «Table view» or «Form view».
- *Attribute table row cache* 1,00 . This row cache makes it possible to save the last loaded N attribute rows so that working with the attribute table will be quicker. The cache will be deleted when closing the attribute table.
- *Representation for NULL values*. Here, you can define a value for data fields containing a NULL value.

Sfat: Improve opening of big data attribute table

When working with layers with big amount of records, opening the attribute table may be slow as the dialog request all the rows in the layer. Setting the *Attribute table behavior* to **Show features visible on map** will make QGIS request only the features in the current map canvas when opening the table, allowing a quick data loading.

Note that data in this attribute table instance will be always tied to the canvas extent it was opened with, meaning that selecting **Show All Features** within such a table will not display new features. You can however update the set

of displayed features by changing the canvas extent and selecting **Show Features Visible On Map** option in the attribute table.

Gestiunea surselor de date

- *Scanează elementele valide din navigator* . Se poate alege «Verificare extensie» sau «Verificare conținut fișier».
- *Scan for contents of compressed files (.zip) in browser dock* defines how detailed is the widget information at the bottom of the Browser panel when querying such files. «No», «Basic scan» and «Full scan» are possible options.
- *Prompt for sublayers when opening*. Some rasters support sublayers — they are called subdatasets in GDAL. An example is netCDF files — if there are many netCDF variables, GDAL sees every variable as a subdataset. The option allows you to control how to deal with sublayers when a file with sublayers is opened. You have the following choices:
  - 'Always': Always ask (if there are existing sublayers)
  - 'If needed': Ask if layer has no bands, but has sublayers
  - 'Never': Never prompt, will not load anything
  - 'Load all': Never prompt, but load all sublayers
- *Automatically refresh directories in browser dock when their contents change*: Allows you to manually opt-out of monitoring directories in the *Browser* panel by default (eg, to avoid potential slow down due to network latency).

Localized data paths

It is possible to use localized paths for any kind of file based data source. They are a list of paths which are used to abstract the data source location. For instance, if `C:\my_maps` is listed in the localized paths, a layer having `C:\my_maps\my_country\ortho.tif` as data source will be saved in the project using `localized:my_country\ortho.tif`.

The paths are listed by order of preference, in other words QGIS will first look for the file in the first path, then in the second one, etc.

Hidden browser paths

This widget lists all the folders you chose to hide from the *Browser panel*. Removing a folder from the list will make it available in the *Browser* panel.

Setările GDAL

GDAL is a data exchange library for geospatial data that supports a large number of vector and raster formats. It provides drivers to read and (often) write data in these formats. The *GDAL* tab exposes the drivers for raster and vector formats with their capabilities.

GDAL raster and vector drivers

The *Raster Drivers* and *Vector Drivers* tabs allow you to define which GDAL driver is enabled to read and/or write files, as in some cases more than one GDAL driver is available.

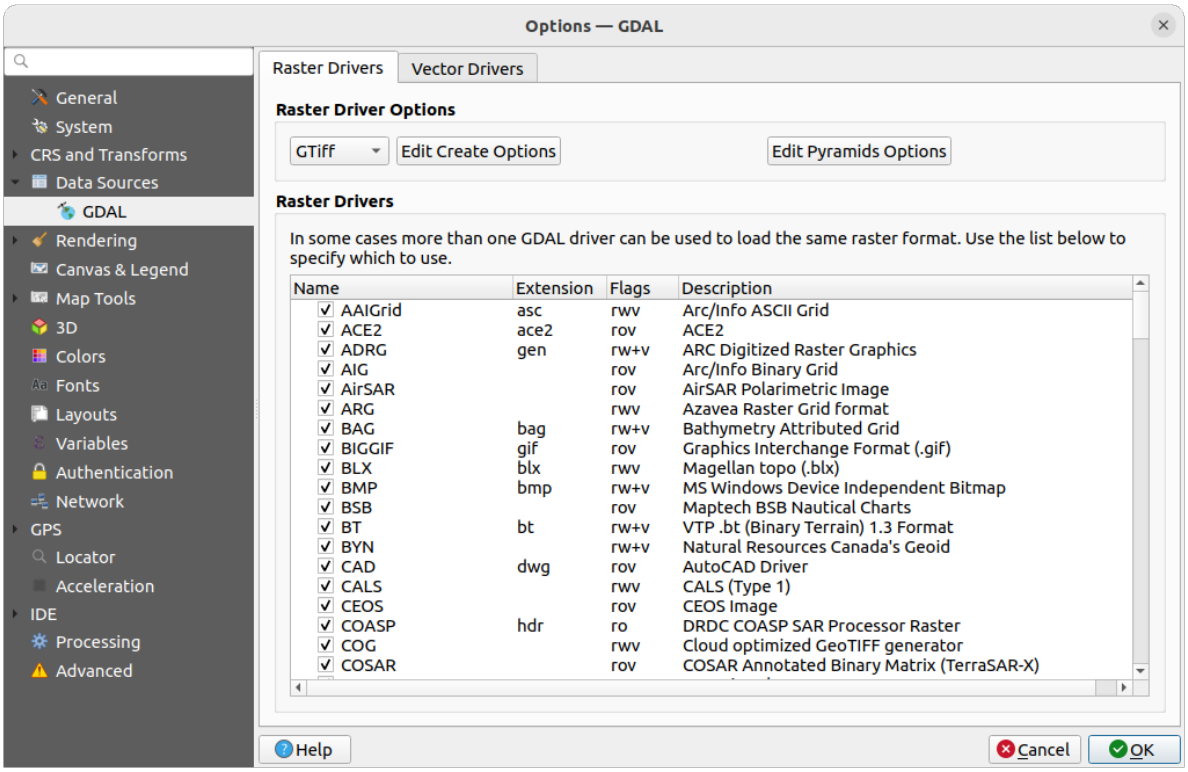


Fig. 9.7: GDAL Settings - Raster drivers

Sfat: Double-click a raster driver that allows read and write access (`rw+ (v)`) opens the *Edit Create options* dialog for customization.

Raster driver options

This frame provides ways to customize the behavior of raster drivers that support read and write access:

- *Edit create options*: allows you to edit or add different profiles of file transformation, i.e. a set of predefined combinations of parameters (type and level of compression, blocks size, overview, colorimetry, alpha…) to use when outputting raster files. The parameters depend on the driver.

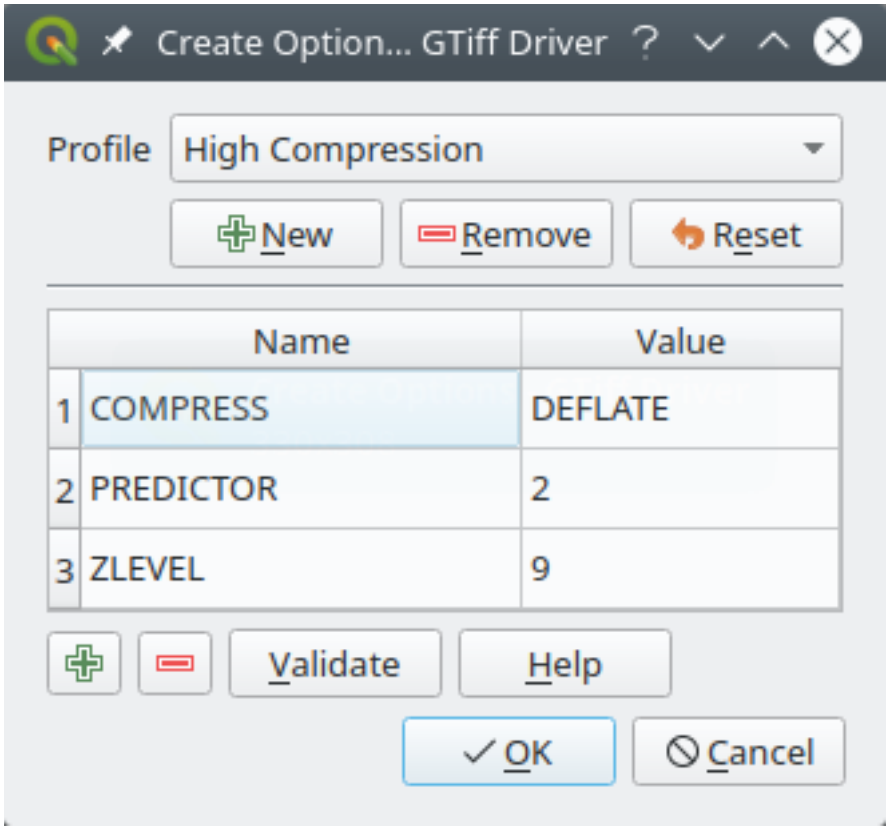


Fig. 9.8: Sample of create options profile (for GeoTiff)

The upper part of the dialog lists the current profile(s) and allows you to add new ones or remove any of them. You can also reset the profile to its default parameters if you have changed them. Some drivers (eg, GeoTiff) have some sample of profiles you can work with.

At the bottom of the dialog:

- The button lets you add rows to fill with the parameter name and value
- The button deletes the selected parameter
- Click the *Validate* button to check that the creation options entered for the given format are valid
- Use the *Help* button to find the parameters to use, or refer to the GDAL raster drivers documentation.

- *Edit Pyramids Options*

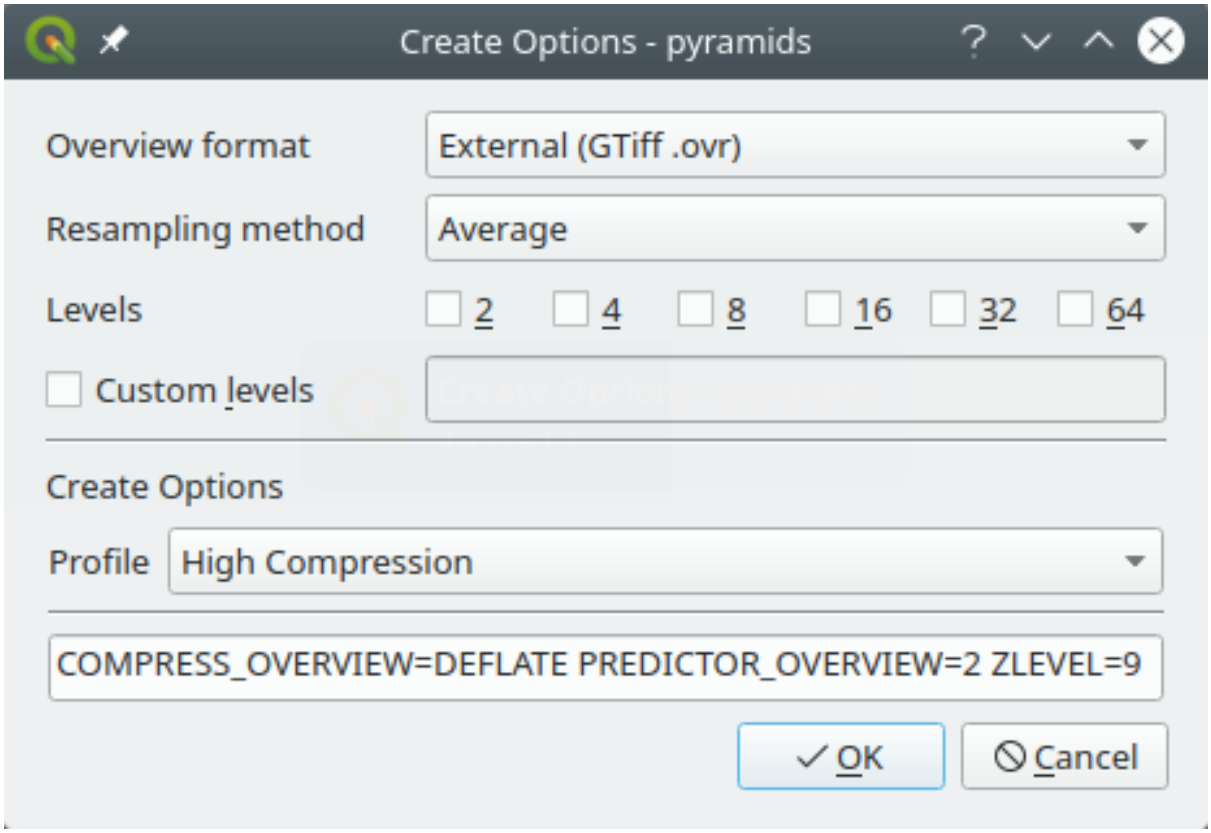


Fig. 9.9: Sample of pyramids profile

9.1.6 Rendering Settings

The *Rendering* tab provides settings for controlling layers rendering in the map canvas.

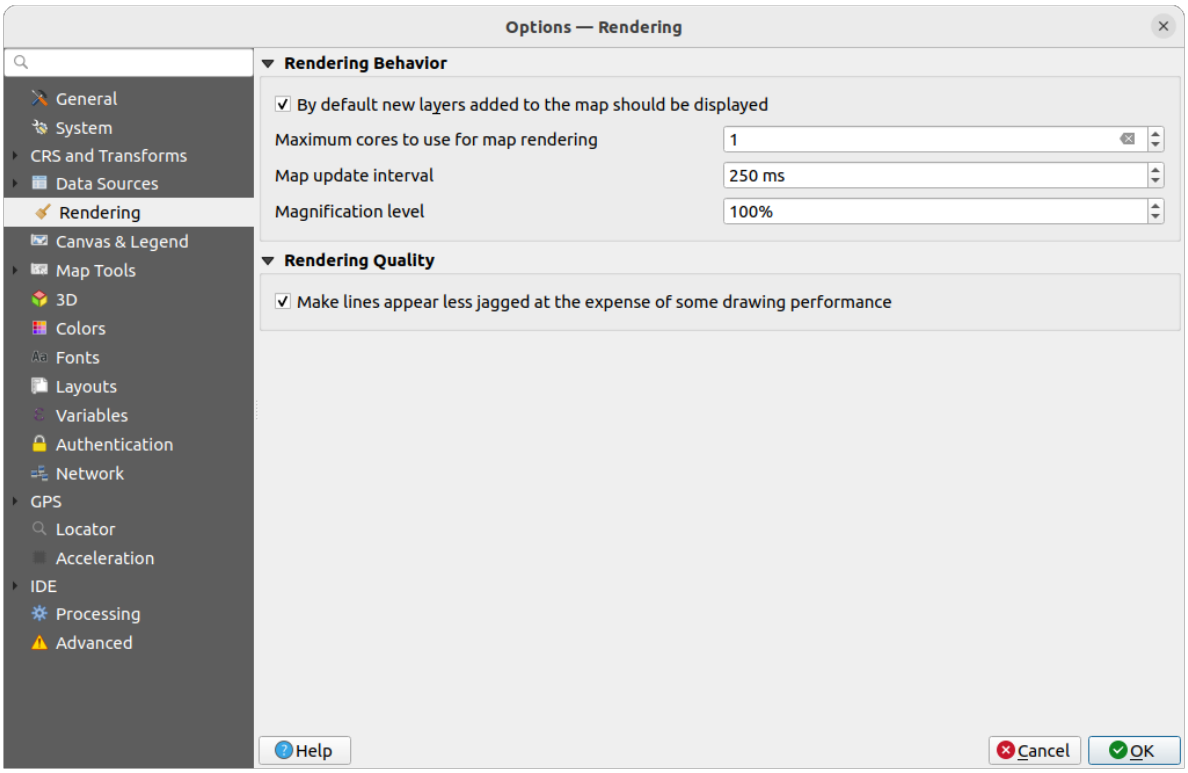


Fig. 9.10: Rendering settings

Rendering Behavior

- *By default new layers added to the map should be displayed*: unchecking this option can be handy when loading multiple layers to avoid each new layer being rendered in the canvas and slow down the process
- Set the *Maximum cores to use for map rendering*
- The map canvas renders in the background onto a separate image and at each *Map update interval* (defaults to 250 ms), the content from this (off-screen) image will be taken to update the visible screen representation. However, if rendering finishes faster than this duration, it will be shown instantaneously.
- *Magnification level* (see the *magnifier*)

Rendering Quality

- *Liniile vor apărea mai puțin zimțate, cu prețul unei pierderi de performanță la desenare*

Vector rendering settings

The *Vector* tab contains specific settings for rendering vector layers.

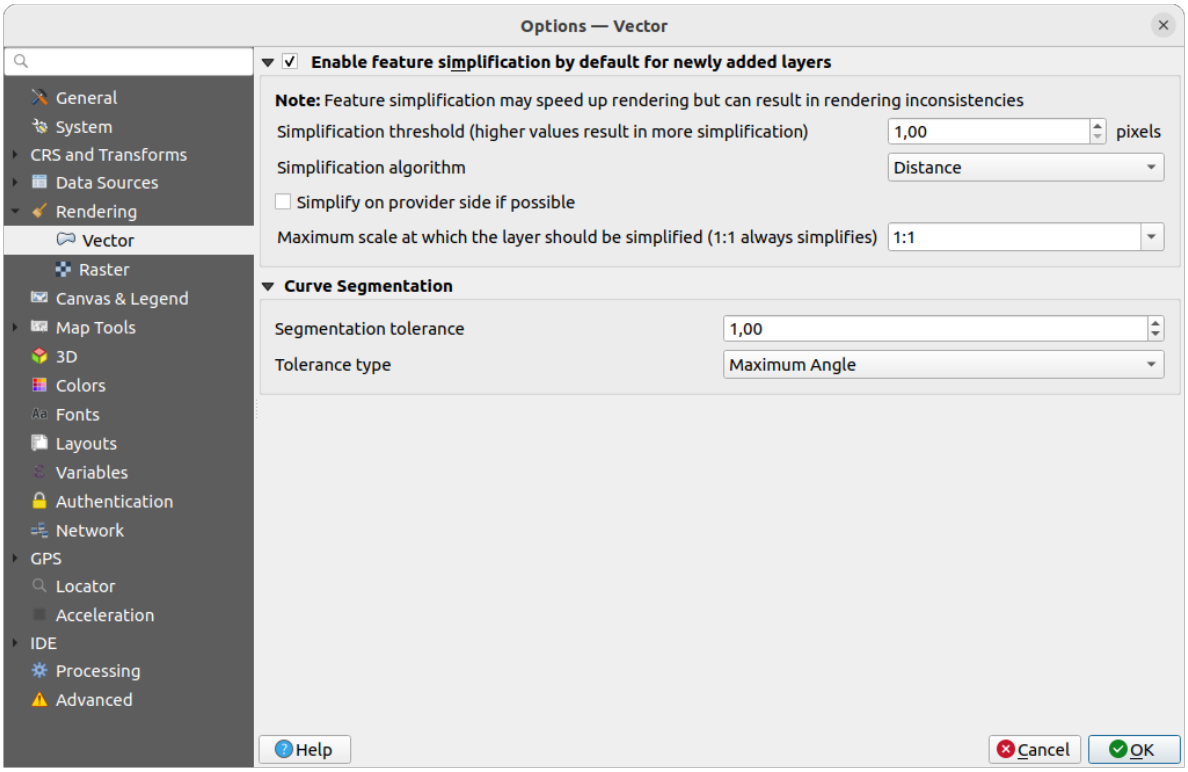


Fig. 9.11: Vector rendering settings

- *Enable Feature Simplification by Default for Newly Added Layers*: you simplify features» geometry (fewer nodes) and as a result, they display more quickly. Be aware that this can cause rendering inconsistencies. Available settings are:
  - *Simplification threshold (higher values result in more simplification)*
  - *Simplification algorithm*: This option performs a local „on-the-fly" simplification on feature's and speeds up geometry rendering. It doesn't change the geometry fetched from the data providers. This is important when you have expressions that use the feature geometry (e.g. calculation of area) - it ensures that these calculations are done on the original geometry, not on the simplified one. For this purpose, QGIS provides three algorithms: «Distance» (default), «SnapToGrid» and «Visvalingam».
  - *Simplify on provider side if possible*: the geometries are simplified by the provider (PostGIS, Oracle…) and unlike the local-side simplification, geometry-based calculations may be affected
  - *Maximum scale at which the layer should be simplified (1:1 always simplifies)*

Notă: Besides the global setting, feature simplification can be set for any specific layer from its *Layer properties* ► *Rendering* menu.

- *Curve Segmentation*
  - *Segmentation tolerance*: this setting controls the way circular arcs are rendered. **The smaller** maximum angle (between the two consecutive vertices and the curve center, in degrees) or maximum difference (distance between the segment of the two vertices and the curve line, in map units), the **more straight line** segments will be used during rendering.

– *Tolerance type*: it can be *Maximum angle* or *Maximum difference* between approximation and curve.

Raster rendering settings

The *Raster* tab contains specific settings for rendering raster layers.

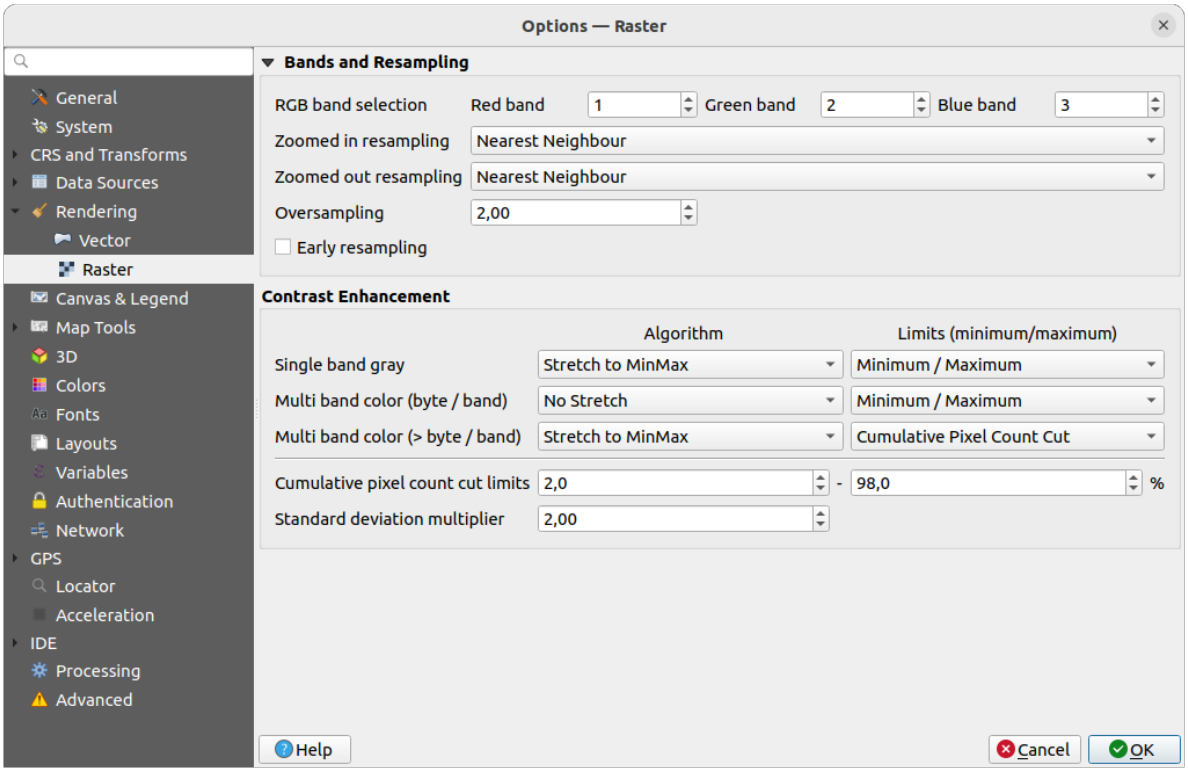


Fig. 9.12: Raster rendering settings

Under *Bands and Resampling*:

- With *RGB band selection*, you can define the number for the Red, Green and Blue band.
- The *Zoomed in resampling* and the *Zoomed out resampling* methods can be defined. For *Zoomed in resampling* you can choose between three resampling methods: «Nearest neighbour», «Bilinear (2x2 kernel)» and «Cubic (4x4 kernel)». For *Zoomed out resampling* you can choose between «Nearest Neighbour» and «Average». You can also set the *Oversampling* value (between 0.0 and 99.99 - a large value means more work for QGIS - the default value is 2.0).
- *Early resampling*: allows to calculate the raster rendering at the provider level where the resolution of the source is known, and ensures a better zoom in rendering with QGIS custom styling. Really convenient for tile rasters loaded using an *interpretation method*. The option can also be set at the layer level (*Symbology* properties)

Contrast Enhancement options can be applied to *Single band gray*, *Multi band color (byte/band)* or *Multi band color (>byte/band)*. For each, you can set:

- the *Algorithm* to use, whose values can be «No stretch», «Stretch to MinMax», «Stretch and Clip to MinMax» or «Clip to MinMax»
- the *Limits (minimum/maximum)* to apply, with values such as «Cumulative pixel count cut», «Minimum/Maximum», «Mean +/- standard deviation».

The *Contrast Enhancement* options also include:

- *Limitele reducerii numărului cumulat de pixeli*
- *Multiplicator pentru Abaterea Standard*

9.1.7 Canvas and Legend Settings

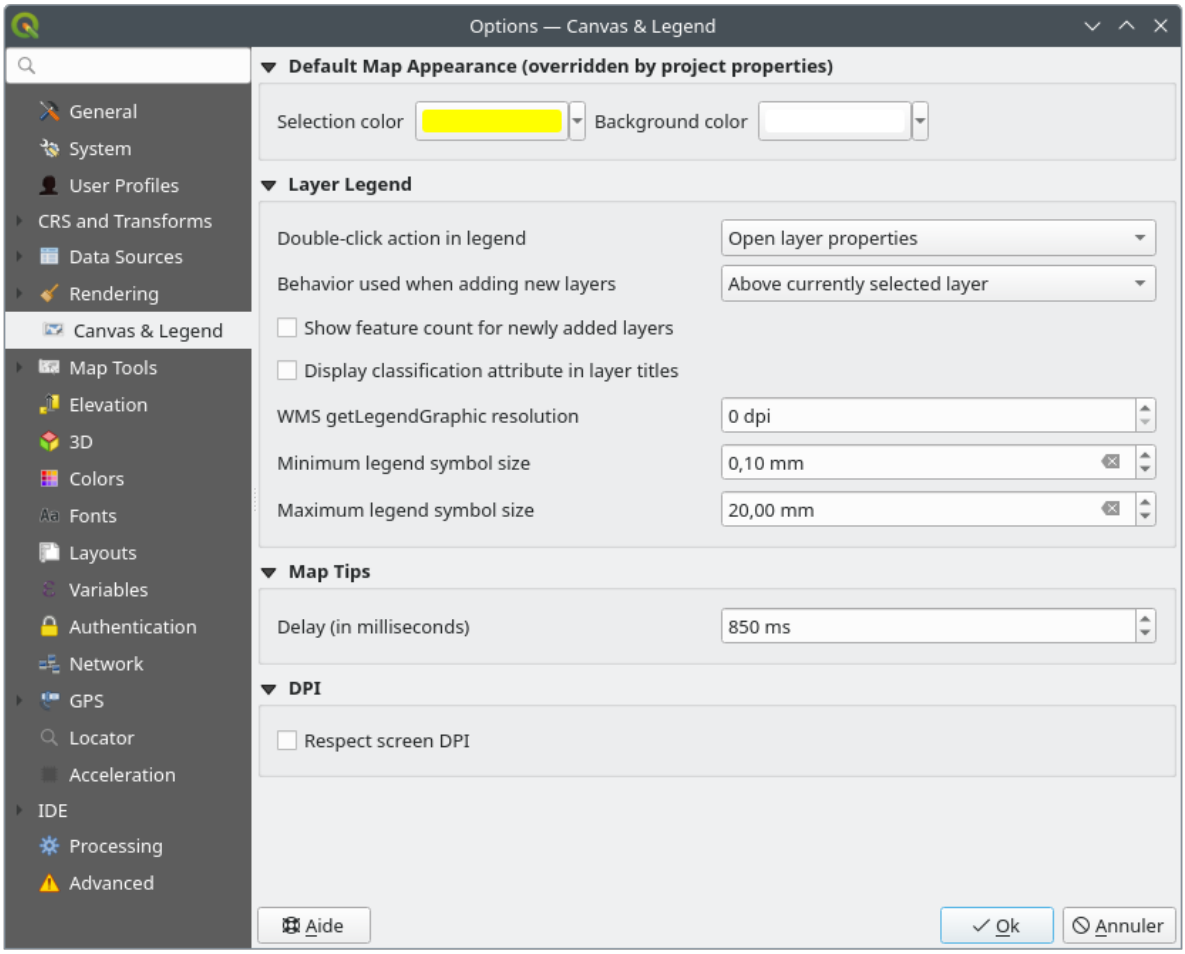


Fig. 9.13: Canvas & Legend settings

These properties let you set:

- the **Default map appearance (overridden by project properties)**: the *Selection color* and *Background color*.
- **Layer legend** interaction:
  - *Double click action in legend*: whether a double-click on a layer should either *Open layer properties* (default), *Open attribute table* or *Open layer styling dock*.
  - *Behavior used when adding new layers*: determines where layers are placed in the *Layers* panel when loaded into QGIS. It can be:
    - *Above currently selected layer*
    - *Always on top of the layer tree*
    - *Optimal index within current layer tree group*: Unlike the other options that sort the new layers among them and place them as a stack at the desired location, this option extents the sorting logic to the whole layer tree (or the active group) and inserts new layers in an „optimal" fashion by insuring that point layers sit on top of point layers, followed by line layers on top of lines, followed by polygon layers, etc.

  - *Show feature count for newly added layers*: displays in the *Layers* panel the number of features next to the layer name. Feature count of classes, if any, is as well displayed. You can right-click on a layer to turn on/off its feature count.
  - *Display classification attribute names* in the Layers panel, e.g. when applying a categorized or rule-based renderer (see *Symbology Properties* for more information).
  - the *WMS getLegendGraphic Resolution*
  - *Minimum* and *Maximum legend symbol size* to control symbol size display in the *Layers* panel
- the *Delay* in milliseconds of layers *map tips* display
- Whether QGIS should *Respect screen DPI*: If enabled, QGIS will attempt to display the canvas with physically accurate scale on screen, depending on the monitor's physical DPI. Symbology with specified display size will also be rendered accurately, e.g. a 10mm symbol will show as 10mm on screen. However, label font sizes on canvas may differ from those in QGIS» UI or other applications. If this setting is turned off, QGIS will use the operating system's logical DPI, which will be consistent with other applications on the system. However, canvas scale and symbology size may be physically inaccurate on screen. In particular, on high-dpi screens, symbology is likely to appear too small.

  For best experience, it is recommended to enable *Respect screen DPI*, especially when using multiple or different monitors and preparing visually high-quality maps. Disabling *Respect screen DPI* will generate output that may be more suitable for mapping intended for on-screen use only, especially where font sizes should match other applications.

Notă: Rendering in layouts is not affected by the *Respect screen DPI* setting; it always respects the specified DPI for the target output device. Also note that this setting uses the physical screen DPI as reported by the operating system, which may not be accurate for all displays.

9.1.8 Map tools Settings

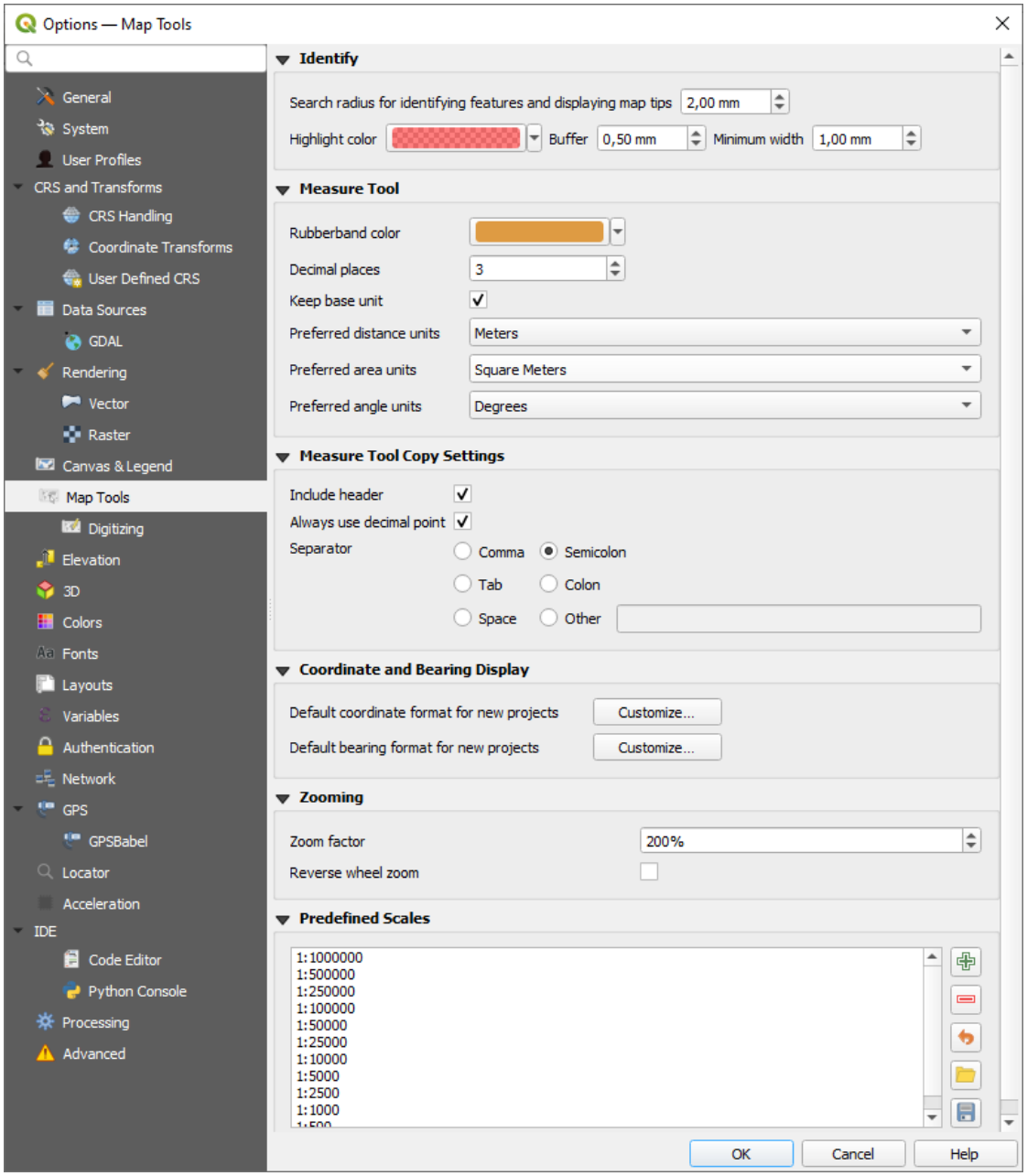


Fig. 9.14: Map tools settings

This tab offers some options regarding the behavior of the *Identify tool*.

- *Search radius for identifying features and displaying map tips* is a tolerance distance within which the identify tool will depict results as long as you click within this tolerance.
- *Highlight color* allows you to choose with which color features being identified should be highlighted.
- *Buffer* determines a buffer distance to be rendered from the outline of the identify highlight.
- *Minimum width* determines how thick should the outline of a highlighted object be.

Instrumentul de măsură

- Definește *Culoarea benzii elastice* a instrumentelor de măsurare

- Definire *Număr de zecimale*
- *Keep base unit* to not automatically convert large numbers (e.g., meters to kilometers)
- *Preferred distance units*: options are «Meters», «Kilometers», «Feet», «Yards», «Miles», «Nautical Miles», «Centimeters», «Millimeters», «Inches», «Degrees» or «Map Units»
- *Preferred area units*: options are «Square meters», «Square kilometers», «Square feet», «Square yards», «Square miles», «Hectares», «Acres», «Square nautical miles», «Square centimeters», «Square millimeters», «Square inches», «Square degrees» or «Map Units»
- *Preferred angle units*: options are «Degrees», «Radians», «Gon/gradians», «Minutes of arc», «Seconds of arc», «Turns/revolutions», milliradians (SI definition) or mil (NATO/military definition)

Some of these options can be overridden at the *project level*.

Measure Tool Copy Settings

These settings allow you to control the behavior of the *Copy* button in the Measure Line window. You can choose to *Include header* to keep columns names, and you can also select *Separator* of your choice. You can also choose *Always use decimal point* to keep your coordinates copied to the clipboard with a dot as a decimal separator, even if your language settings in QGIS options are set up to use a comma. In this case, if you don't have the *Always use decimal point* option checked, you will be unable to select comma as a field separator.

Coordinate and Bearing Display

This section provides ways to *Configure*:

- *Default coordinate format for new projects*, as displayed in the *Coordinates* box on QGIS status bar and in the *Derived* section of the Identify features tool's results
- *Default bearing format for new projects*, as displayed in the status bar for the map canvas panning direction and by the Measure bearing tool.

These options can be overridden at the *project level*.

Deplasare și transfocare

- Define a *Zoom factor* for zoom tools or wheel mouse
- *Reverse wheel zoom* allows you to adjust mouse wheel scrolling direction.

Scări predefinite

Here, you find a list of predefined scales to display by default in the scale-related drop-down widgets, such as the status bar *Scale*, the visibility scales selector or secondary 2D map view settings,… With the and buttons you can add or remove your personal scales. You can also import or export scales from/to a `.XML` file. Note that you still have the possibility to remove your changes and reset to the predefined list.

From the project properties dialog, you can also set your own list of scales, overriding this global one in the widgets.

Setări de digitizare

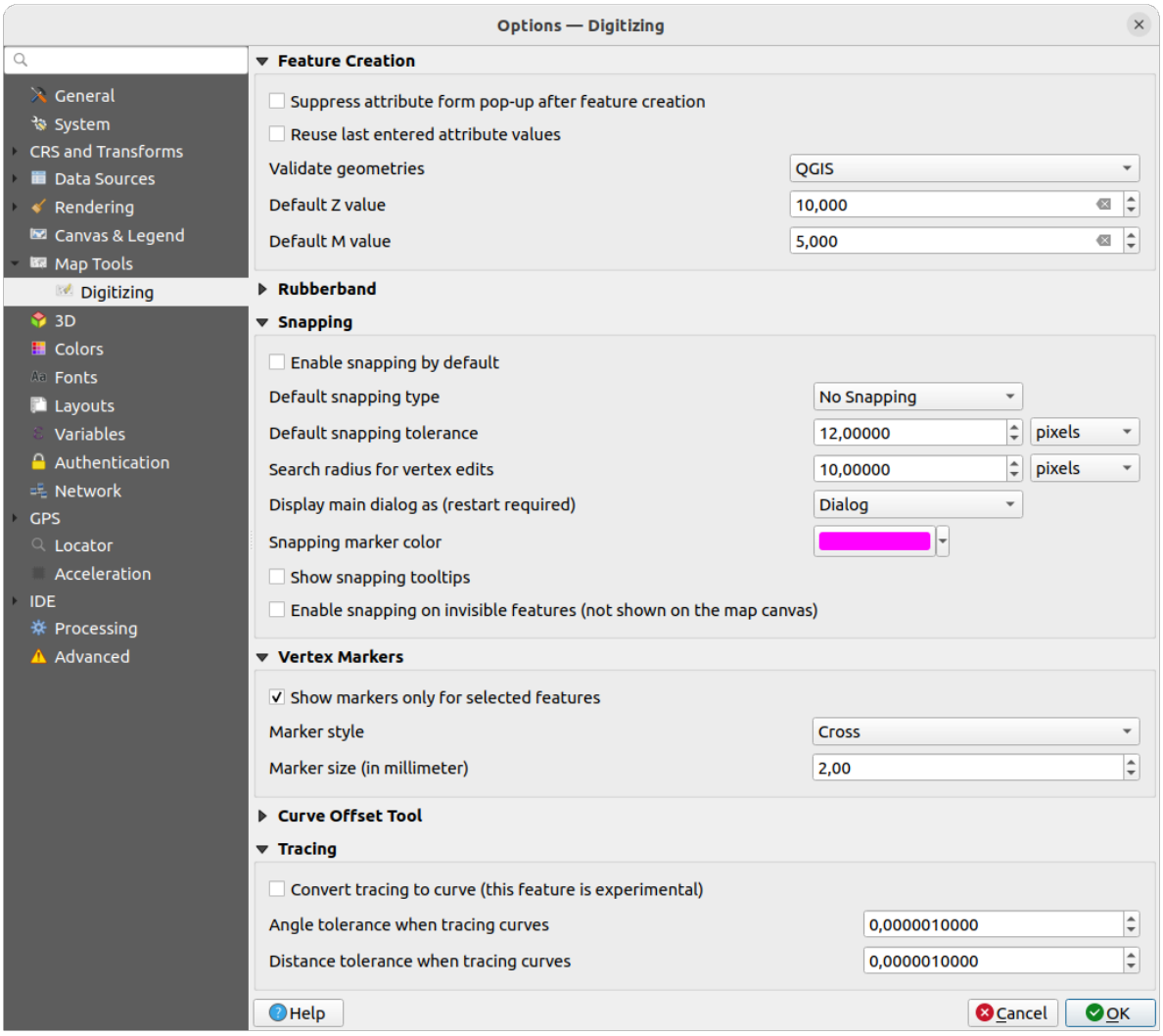


Fig. 9.15: Setări de digitizare

This tab helps you configure general settings when *editing vector layer* (attributes and geometry).

Creare entitate

- *Suppress attribute form pop-up after feature creation*: this choice can be overridden in each layer properties dialog.
- *Reuse last entered attribute values*: remember the last used value of every attribute and use it as default for the next feature being digitized. Works per layer. This behavior can also be controled on a per-field basis (see *Configure the field behavior*).
- *Validate geometries*. Editing complex lines and polygons with many nodes can result in very slow rendering. This is because the default validation procedures in QGIS can take a lot of time. To speed up rendering, it is possible to select GEOS geometry validation (starting from GEOS 3.3) or to switch it off. GEOS geometry validation is much faster, but the disadvantage is that only the first geometry problem will be reported.

  Note that depending on the selection, reports of geometry errors may differ (see *Types of error messages and their meanings*)
- *Default Z value* to use when creating new 3D features.

Bandă elastică

- Define Rubberband *Line width*, *Line color* and *Fill color*.
- *Don't update rubberband during vertex editing*.

Acroșare

- *Enable snapping by default* activates snapping when a project is opened
- Define *Default snap mode* («Vertex», «Segment», «Centroid», «Middle of segments», «Line endpoints», «Area»)
- Definește *Toleranța de acroșare implicită*, în unități de hartă sau în pixeli
- Definește *Raza de căutare pentru editarea vertecșilor*, în unități de hartă sau în pixeli
- *Display main dialog as (restart required)*: set whether the Advanced Snapping dialog should be shown as «Dialog» or «Dock».
- *Snapping marker color*
- *Show snapping tooltips* such as name of the layer whose feature you are about to snap. Helpful when multiple features overlap.
- *Enable snapping on invisible features (not shown on the map canvas)*

Simbolurile vertexului

- *Arată simbolurile numai pentru entitățile selectate*
- Define vertex *Marker style* («Cross» (default), «Semi transparent circle» or «None»)
- Define vertex *Marker size (in millimeter)*

Instrument pentru curba de decalare

The next 3 options refer to the Offset Curve tool in *Digitizare avansată*. Through the various settings, it is possible to influence the shape of the line offset. These options are possible starting from GEOS 3.3.

- *Join style*: «Round», «Mitre» or «Bevel»
- *Segmentele cvadrantului*
- *Miter limit*

Tracing

By activating the *Convert tracing to curve* you can create curve segments while digitizing. Keep in mind that your data provider must support this feature.

9.1.9 Elevation settings

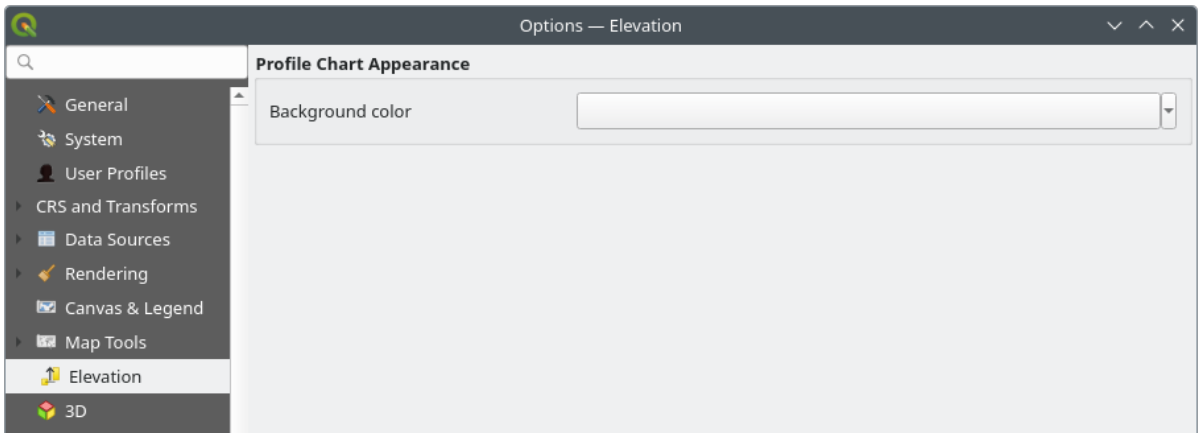


Fig. 9.16: Elevation settings

In *Elevation* menu, you can set a specific color to use as the *Background color* for *elevation profiles*. This can make the chart more readable for certain datasets, e.g. point clouds with RGB coloring, where the default background color is too similar to point colors to be easily discernable. If unchanged, the elevation profiles will continue to display using the standard system background color.

9.1.10 3D settings

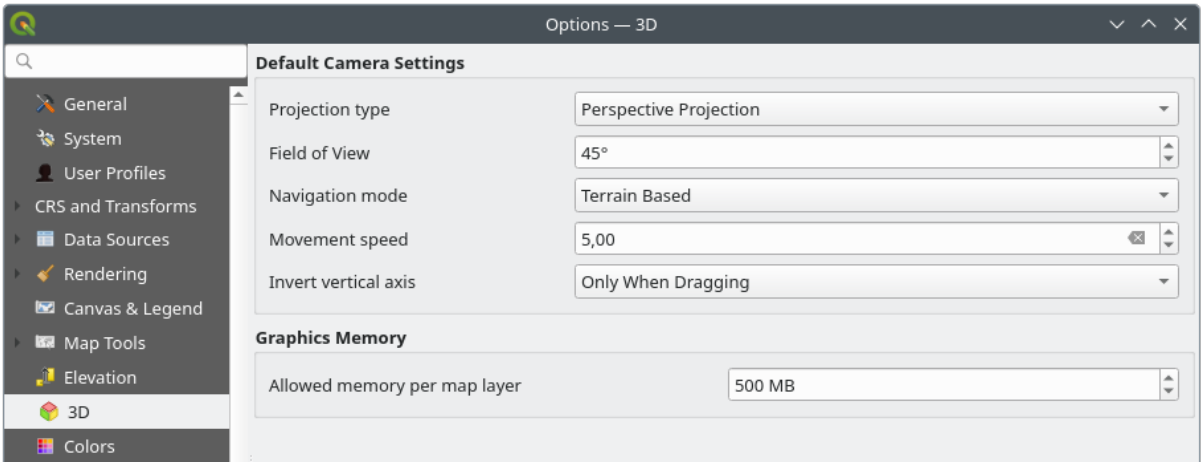


Fig. 9.17: 3D settings

The *3D* menu helps you configure some default settings to use for any *3D Map view*. These can refer to:

- *Default Camera Settings*:
  - *Projection type*: allowing to view the 3D scene in a:
    - *Perspective projection* (default): Parallel lines appear to meet in the distance. Objects appear to shrink the farther they are from the camera.
    - or an *Orthogonal projection*: Parallel lines appear parallel. Objects appear the same size regardless of distance.

  - Camera's *Field of view*: only relevant in perspective projection, specifies the current vertical field of view in degrees and determines how much of the scene is visible to the camera. Default value is 45°.
  - *Navigation mode*: provides different means to interact with the 3D scene. Available modes are:
    * *Terrain based*: the camera follows around a fixed position on the surface of the terrain as the scene is navigated.
    * *Walk mode (first person)*

    Depending on the selected mode, *navigation commands* differ.
  - *Movement speed*
  - *Invert vertical axis*: Controls whether vertical axis movements should be inverted from their normal behaviour. Only affects movement in the *Walk mode*. It can be set to:
    * *Never*
    * *Only when dragging*: causes the vertical motion to inverted only when performing a click-and-drag camera rotation
    * and *Always*: causes the motions to be inverted when both click-and-dragging and when the camera movement is locked to the cursor (via a ~ key press)

- Under *Graphics memory*, the *Allowed memory per layer* option lets you set the GPU memory limit configuration on each layer. This is useful for users utilizing large 3D scenes which exhaust the available GPU memory resources. When a limit is hit, a warning is also displayed, which should assist in troubleshooting large scenes.

9.1.11 Colors settings

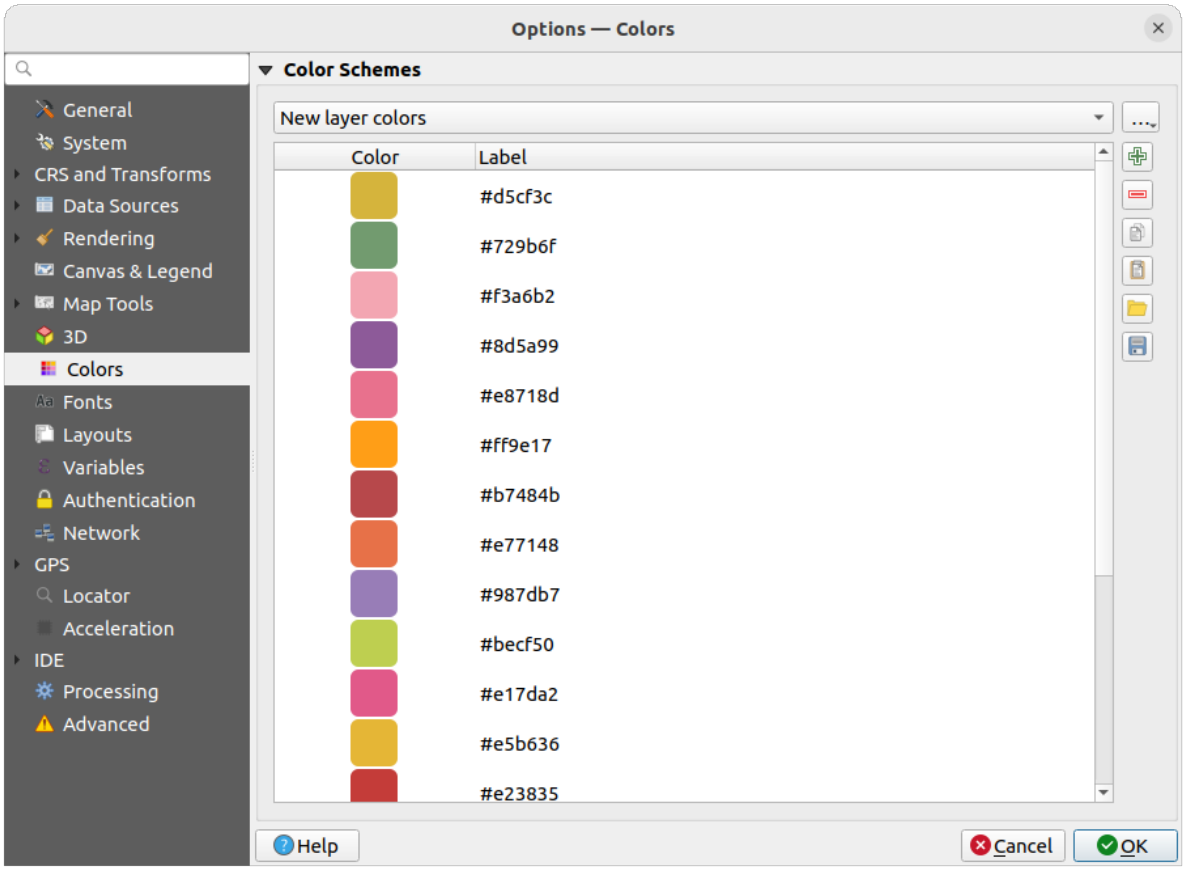


Fig. 9.18: Colors settings

This menu allows you to create or update palettes of colors used throughout the application in the *color selector widget*. You can choose from:

- *Recent colors* showing recently used colors
- *Standard colors*, the default palette of colors
- *Project colors*, a set of colors specific to the current project (see *Styles Properties* for more details)
- *New layer colors*, a set of colors to use by default when new layers are added to QGIS
- or custom palette(s) you can create or import using the … button next to the palette combobox.

By default, *Recent colors*, *Standard colors* and *Project colors* palettes can not be removed and are set to appear in the color button drop-down. Custom palettes can also be added to this widget thanks to the *Show in Color Buttons* option.

For any of the palettes, you can manage the list of colors using the set of tools next to the frame, ie:

- *Add* or *Remove* color
- *Copy* or *Paste* color
- *Import* or *Export* the set of colors from/to `.gpl` file.

Double-click a color in the list to tweak or replace it in the *Color Selector* dialog. You can also rename it by double-clicking in the *Label* column.

9.1.12 Setări Fonturi

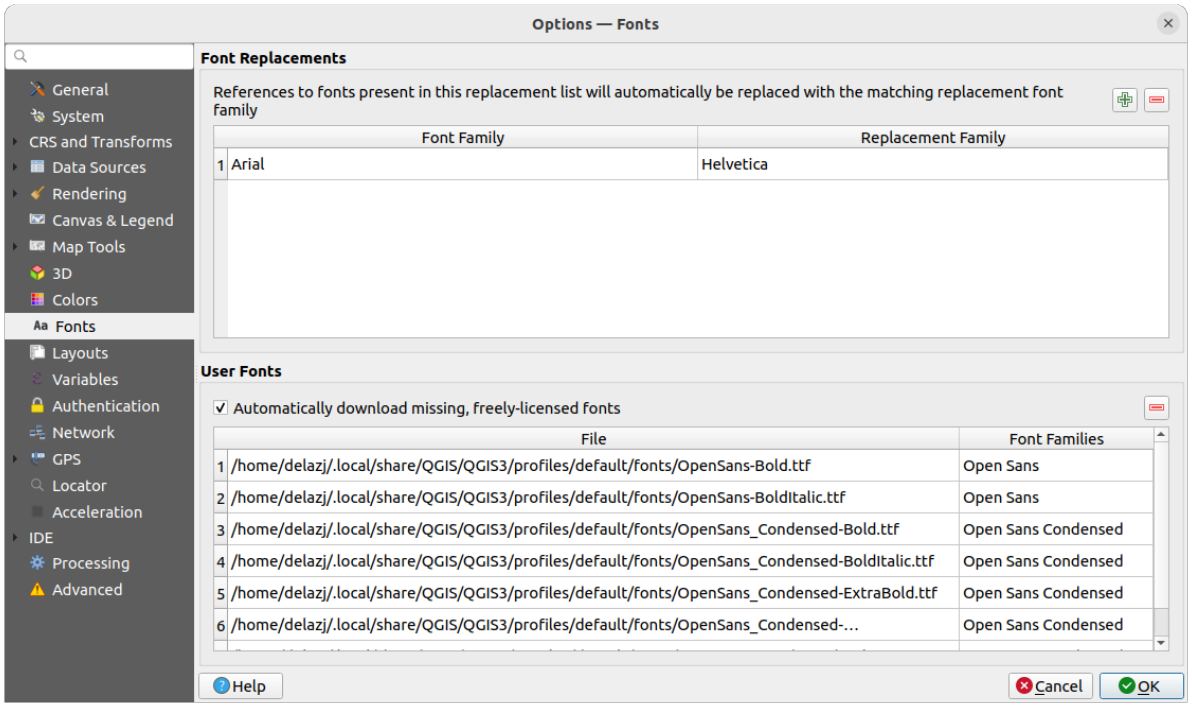


Fig. 9.19: Setările fonturilor

The *Fonts* tab provides support to manage fonts used across the projects:

- *Font Replacements*: gives ability to populate a list of automatic font replacements to apply when loading projects or styles, allowing better support for projects and styles to be used across different operating systems (e.g. replace „Arial” with „Helvetica”).

- *User Fonts*: Allows you to place TTF or OTF fonts in the `fonts` sub-folder of the *user profile*. These fonts can be automatically loaded at QGIS startup time. This provides a way to use fonts without requiring them to be installed on an operating system level, which is often blocked in enterprise environments. The panel lists all installed user fonts and allows you to manage (i.e. remove) previously installed user fonts.

  It is also possible to *Automatically download missing, freely-licensed fonts*: E.g. if you open a project or style, or try to load a vector tile layer that references fonts that aren't currently available, then a hard-coded list of freely licensed fonts to download via URL is consulted to determine whether it's possible to automatically download the font to the user profile font directory (with notification of the font license).

9.1.13 Layouts settings

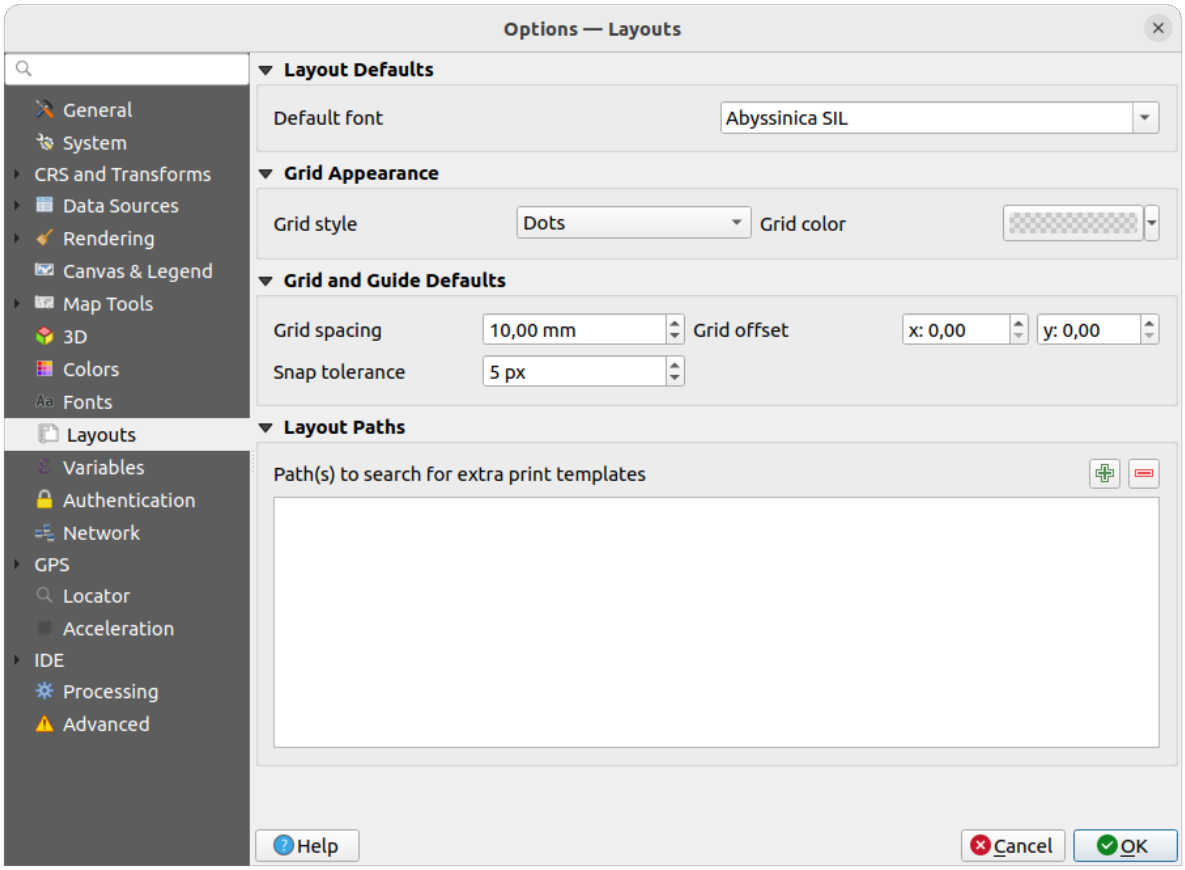


Fig. 9.20: Layouts settings

Valori implicite pentru compoziție

You can define the *Default font* used within the *print layout*.

Aspectul grilei

- Define the *Grid style* («Solid», «Dots», «Crosses»)
- Definiți *Culoarea grilei*

Grilă și de ghidaje implicite

- Define the *Grid spacing*
- Define the *Grid offset* for X and Y
- Define the *Snap tolerance*

Layout Paths

- Define *Path(s) to search for extra print templates*: a list of folders with custom layout templates to use while creating new one.

9.1.14 Variables settings

The *Variables* tab lists all the variables available at the global-level.

It also allows the user to manage global-level variables. Click the button to add a new custom global-level variable. Likewise, select a custom global-level variable from the list and click the button to remove it.

More information about variables in the *Storing values in Variables* section.

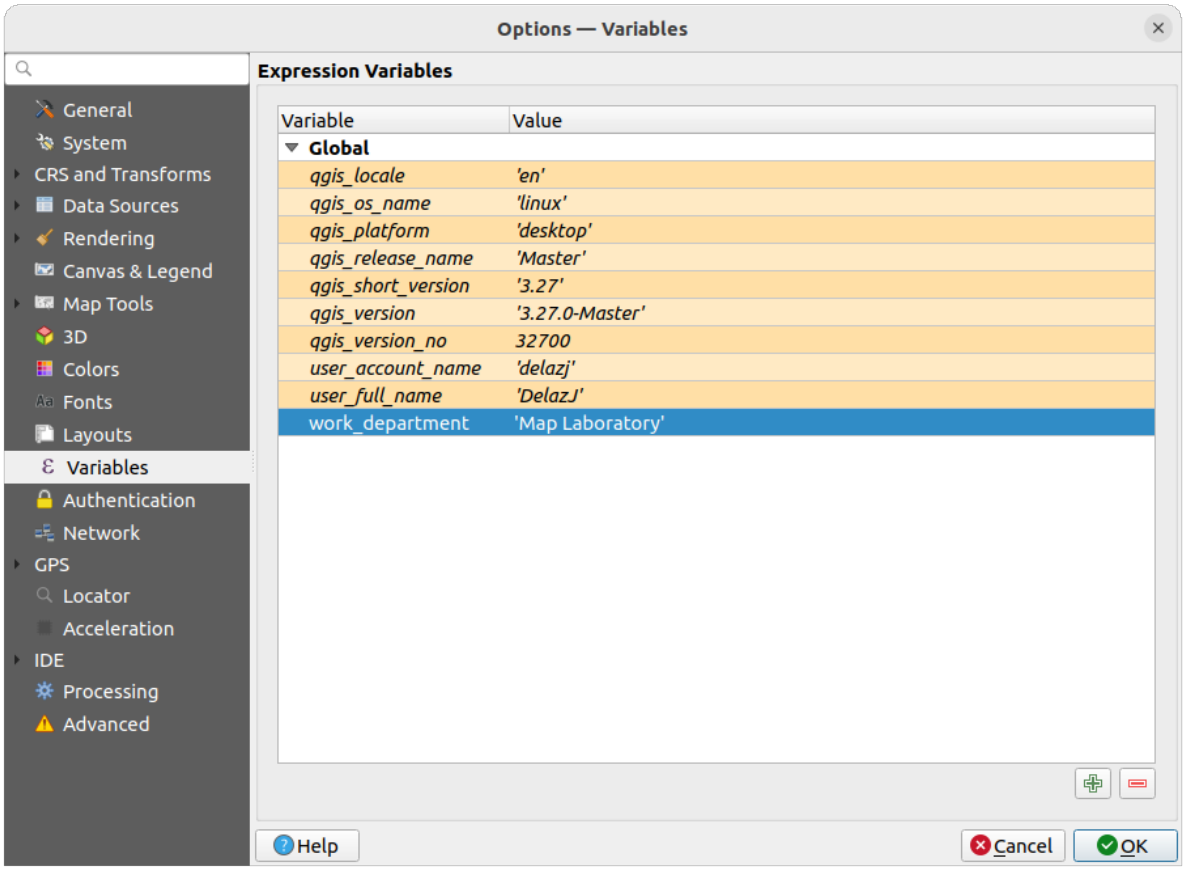


Fig. 9.21: Variables settings

9.1.15 Authentication settings

In the *Authentication* tab you can set authentication configurations and manage PKI certificates. See *Sistem de Autentificare* for more details.

To manage authentications, you can use the list of tools next to the frame, ie:

- Add new authentication configuration
- Remove selected authentication configuration
- Edit selected authentication configuration

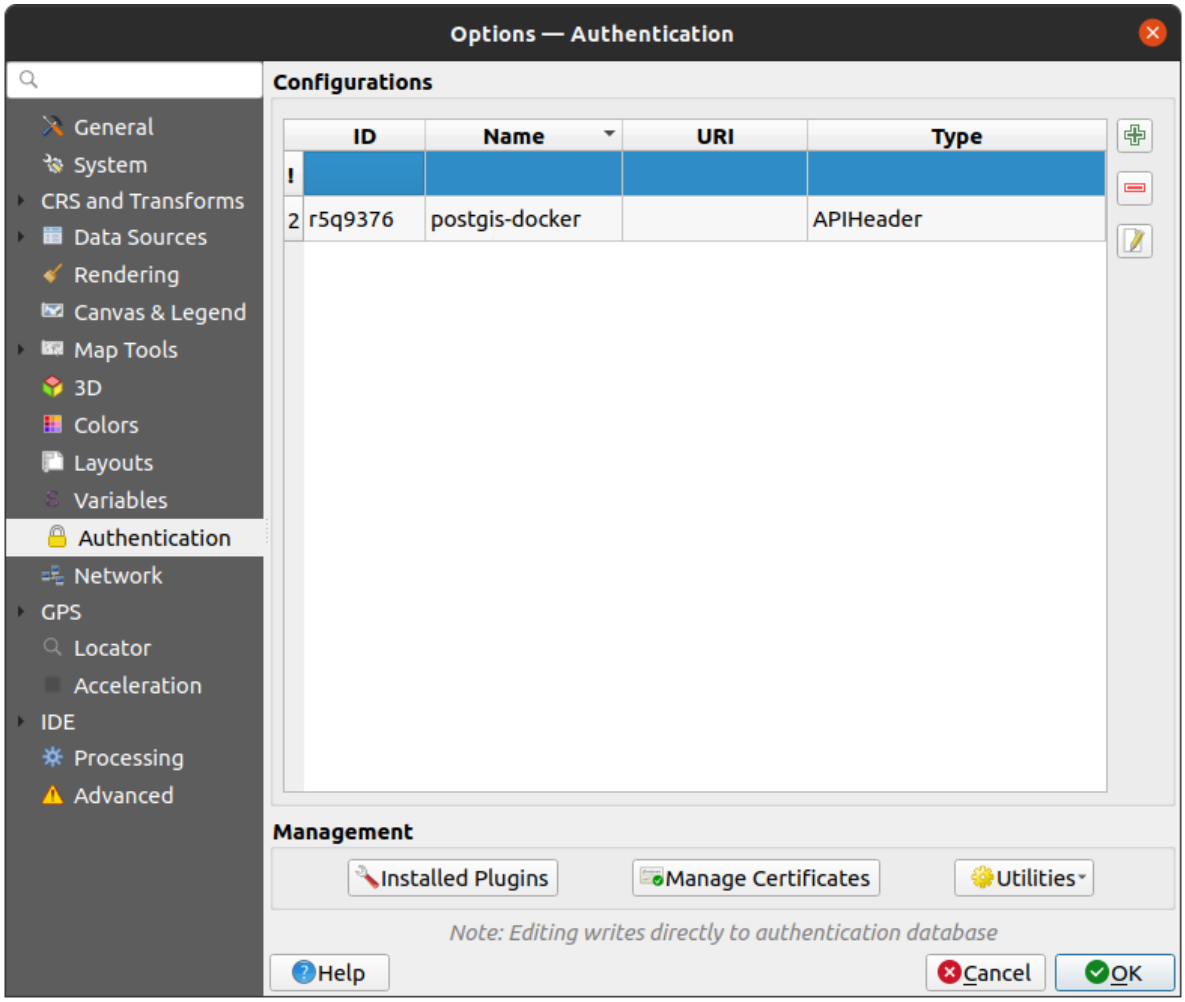


Fig. 9.22: Authentication settings

9.1.16 Network settings

Generalități

- Definire *Timpul de așteptare pentru cererile de rețea (ms)* - implicit este 60000
- Define *Default expiration period for WMS Capabilities (hours)* - default is 24
- Define *Default expiration period for WMS-C/WMTS tiles (hours)* - default is 24
- Define *Max retry in case of tile or feature request errors*
- Definire *User-Agent*

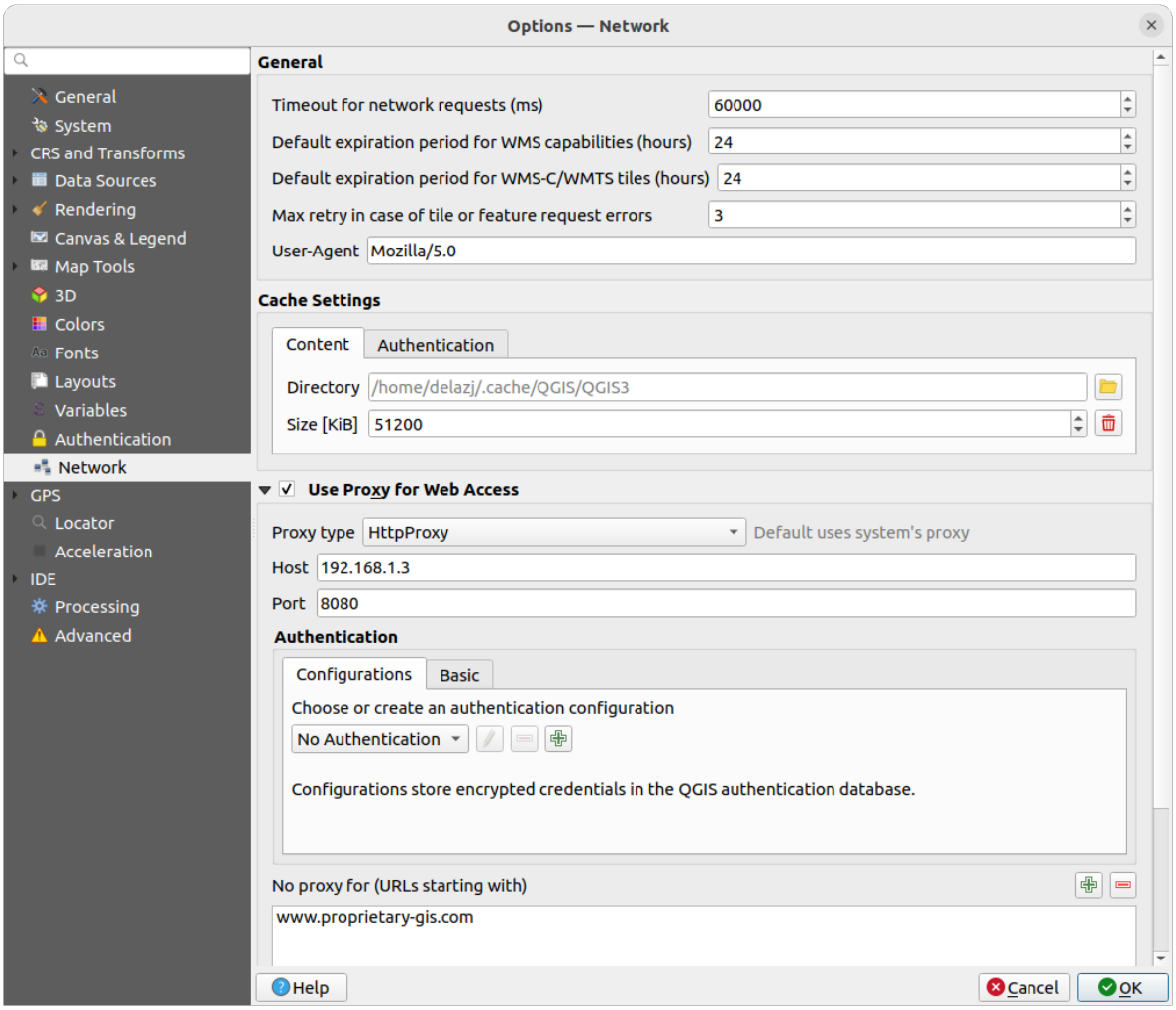


Fig. 9.23: Network and proxy settings

Setările memoriei tampon

Defines the *Directory* and a *Size* for the cache. Also offers tools to *automatically clear the connection authentication cache on SSL errors (recommended)*.

Proxy for web access

- *Use proxy for web access*
- Set the *Proxy type* according to your needs and define «Host» and «Port». Available proxy types are:
  - *Default Proxy*: Proxy is determined based on system's proxy
  - *Socks5Proxy*: Generic proxy for any kind of connection. Supports TCP, UDP, binding to a port (incoming connections) and authentication.
  - *HttpProxy*: Implemented using the „CONNECT" command, supports only outgoing TCP connections; supports authentication.
  - *HttpCachingProxy*: Implemented using normal HTTP commands, it is useful only in the context of HTTP requests.
  - *FtpCachingProxy*: Implemented using an FTP proxy, it is useful only in the context of FTP requests.

Credentials of proxy are set using the *authentication widget*.

Excluding some URLs can be added to the text box below the proxy settings (see Fig. 9.23). No proxy will be used if the target url starts with one of the string listed in this text box.

If you need more detailed information about the different proxy settings, please refer to the manual of the underlying QT library documentation at https://doc.qt.io/archives/qt-5.9/qnetworkproxy.html#ProxyType-enum

Sfat: Folosirea Proxi-urilor

Using proxies can sometimes be tricky. It is useful to proceed by «trial and error» with the above proxy types, to check if they succeed in your case.

9.1.17 Setări GPS

The *GPS* dialog helps you configure GPS devices connections and properties in QGIS. It also provides settings for GPS tracking and data digitizing.

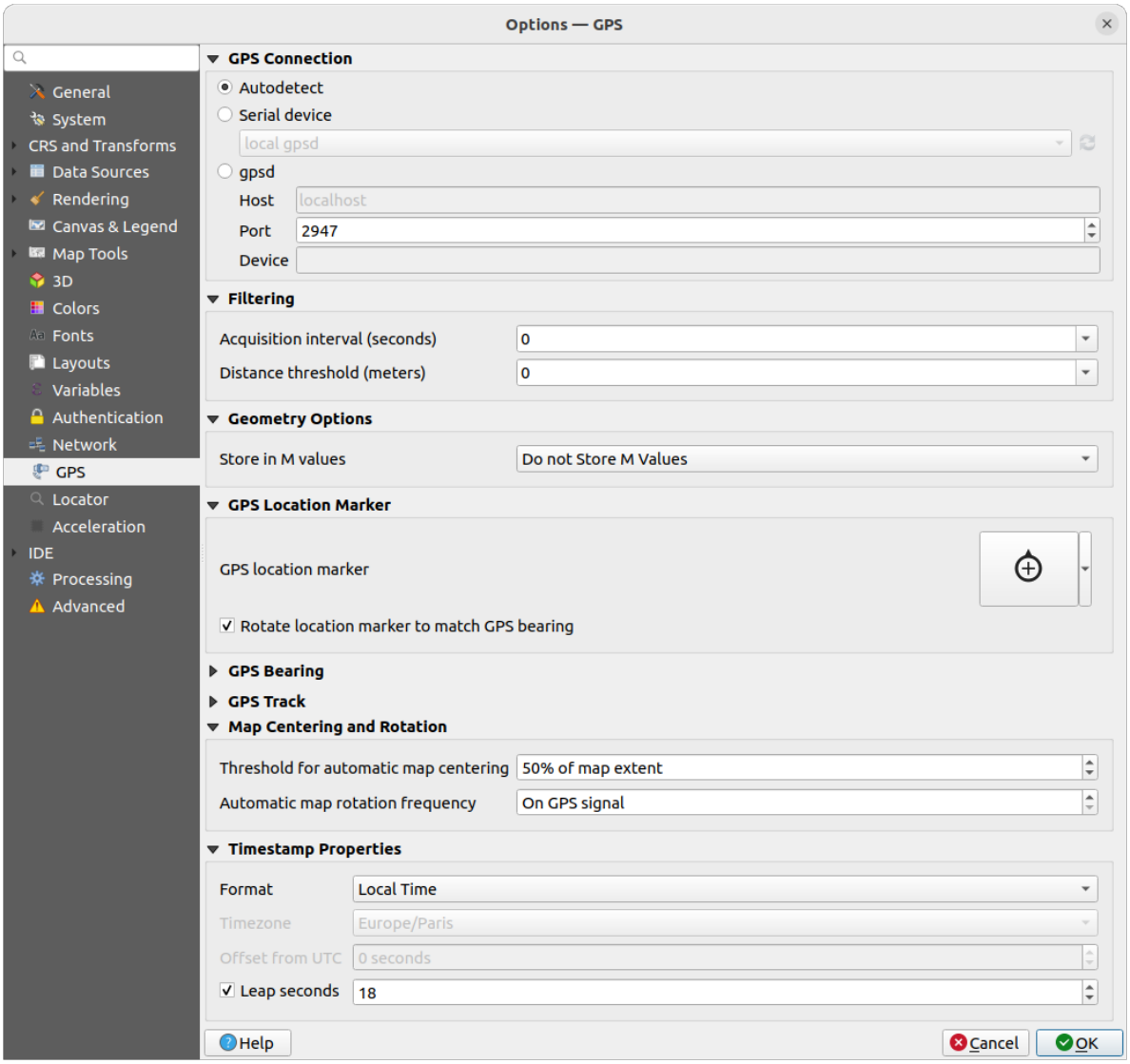


Fig. 9.24: Setări GPS

Puteți specifica:

- *GPS Connection*: provides different means to connect the device to QGIS
  - *Autodetect*
  - *Serial device* (reload required if a new GPS Device is connected)
  - *gpsd* (selecting the *Host* and *Port* your GPS is connected to, and providing a *Device* name)

  In case of connection problems, you can try to switch from one to another.
- *Filtering*: You can set an *Acquisition interval (seconds)* and/or a *Distance threshold (meters)* parameters to keep the cursor still active when the receiver is in static conditions.
- Under *Geometry Options*, the *Store in M values* option allows to create geometries with M values from the inbuilt GPS logging tools. This applies to both features digitized from GPS logs and from the *Log to Geopackage/SpatiaLite* functionality… Options include storing timestamps (as ms since epoch), ground speed, altitudes, bearings, and accuracy components as m values.
- *GPS Location Marker*
  - Configure a symbol for the *GPS location marker*, indicating the current GPS position
  - *Rotate location marker to match GPS bearing*: whether the marker symbol should be rotated to match the GPS direction
- *GPS Bearing*:
  - configure a *Bearing line style* using QGIS line symbol properties
  - set whether to *Calculate Bearings from travel direction*: If checked, the bearing reported by the GPS device will be ignored and the bearing will instead be calculated by the angle between the previous two GPS locations.
- *GPS Track*: set symbol to use for the *Track line style*
- *Map Centering and Rotation*: defines when the map canvas is updated according to the user displacement on the field:
  - *Threshold for automatic map centering*: defines the minimal offset of the GPS position from the map canvas center to trigger an automatic *map recentering* if enabled.
  - *Automatic map rotation frequency*: defines how often the map rotation to match the GPS bearing could happen; it can be on a custom duration or *On GPS signal*.
- *Timestamp Properties* to configure how time values are displayed and stored in the data. Parameters include the *Format* which can be:
  - *Local time*
  - *UTC*
  - *UTC with offset*, to account for daylight savings offsets or other complex time zone issues
  - a specific *Time zone*

  Moreover, *Leap seconds* correction can be applied, by adding the seconds to GPS timestamp.

GPSBabel

GPSBabel converts waypoints, tracks, and routes between popular GPS receivers such as Garmin or Magellan and mapping programs like Google Earth or Basecamp. Literally hundreds of GPS receivers and programs are supported. QGIS relies on GPSBabel to interact with these devices and *manipulate their data*.

For details on how-to, please refer to *Loading to or from a device*.

9.1.18 Locator settings

The *Locator* tab lets you configure the *Locator bar*, a quick search widget available on the status bar to help you perform searches in the application. It provides some default filters (with prefix) to use:

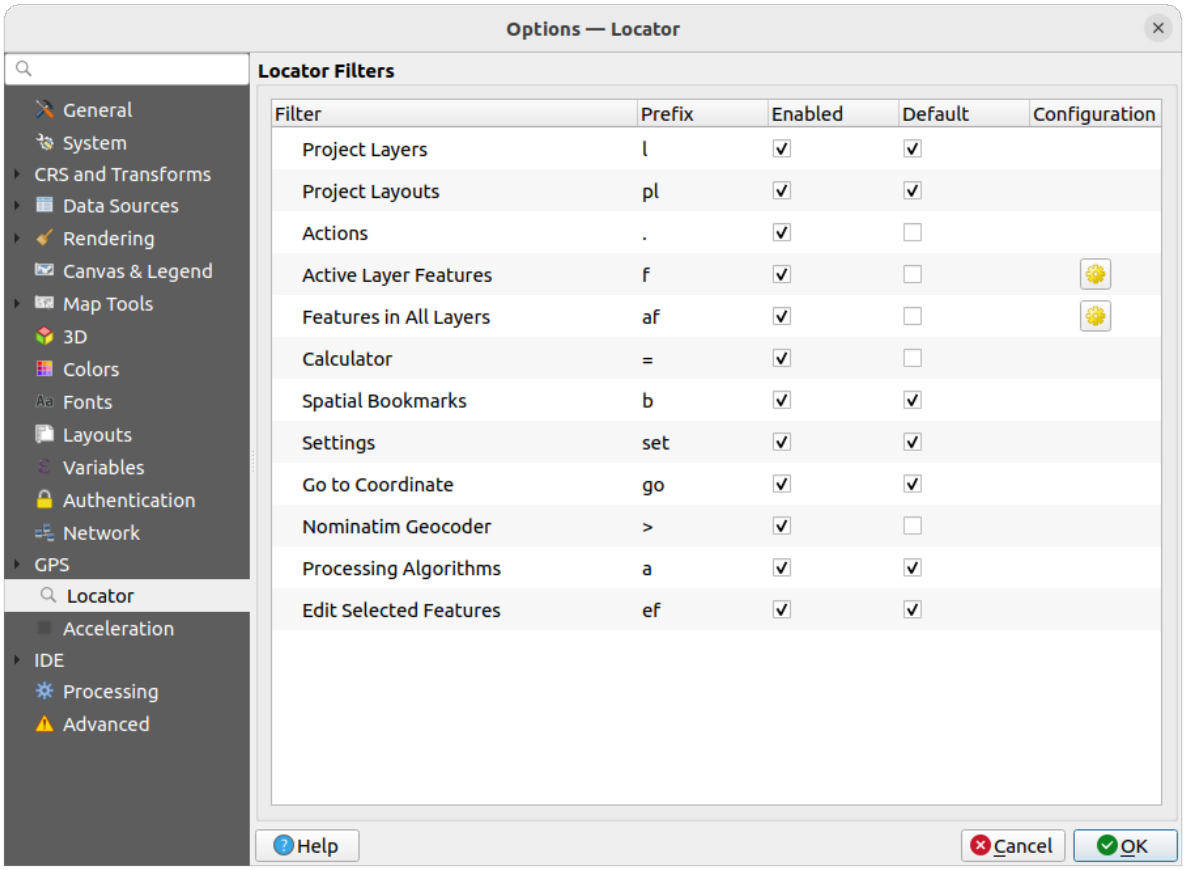


Fig. 9.25: Locator settings

- *Project Layers* (`l`): finds and selects a layer in the *Layers* panel.
- *Project Layouts* (`pl`): finds and opens a print layout.
- Actions (`.`): finds and executes a QGIS action; actions can be any tool or menu in QGIS, opening a panel…
- *Active Layer Features* (`f`): searches for matching attributes in any field from the current active layer and zooms to the selected feature. Press to configure the maximum number of results.
- *Features in All Layers* (`af`): searches for matching attributes in the *display name* of each *searchable layers* and zooms to the selected feature. Press to configure the maximum number of results and the maximum number of results per layer.
- *Calculator* (=): allows evaluation of any QGIS expression and, if valid, gives an option to copy the result to the clipboard.

- *Spatial Bookmarks* (`b`): finds and zooms to the bookmark extent.
- *Settings* (`set`): browses and opens project and application-wide properties dialogs.
- *Go to Coordinate* (`go`): pans the map canvas to a location defined by a comma or space separated pair of x and y coordinates or a formatted URL (e.g., OpenStreetMap, Leaflet, OpenLayer, Google Maps, …). The coordinate is expected in WGS 84 (`epsg:4326`) and/or map canvas CRS.
- *Nominatim Geocoder* (`>`): geocodes using the Nominatim geocoding service of the OpenStreetMap Foundation.
- Processing Algorithms (`a`): searches and opens a Processing algorithm dialog.
- *Edit Selected Features* (`ef`): gives quick access and runs a compatible *modify-in-place* Processing algorithm on the active layer.

In the dialog, you can:

- customize the filter *Prefix*, i.e. the keyword to use to trigger the filter
- set whether the filter is *Enabled*: the filter can be used in the searches and a shortcut is available in the locator bar menu
- set whether the filter is *Default*: a search not using a filter returns results from only the default filters categories.
- Some filters provide a way to configure the number of results in a search.

The set of default locator filters can be extended by plugins, eg for OSM nominatim searches, direct database searching, layer catalog searches, …

9.1.19 Acceleration settings

OpenCL acceleration settings.

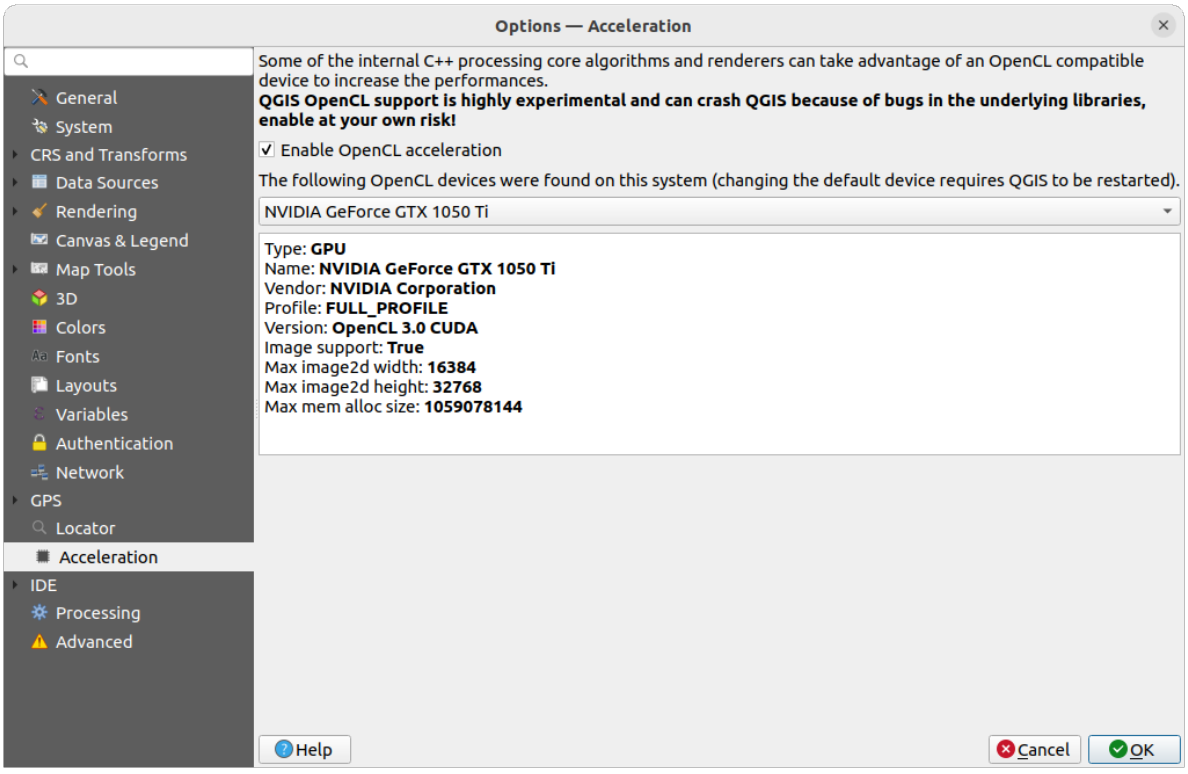


Fig. 9.26: Acceleration settings

Depending on your hardware and software, you may have to install additional libraries to enable OpenCL acceleration.

9.1.20 IDE settings

Code Editor settings

In the *Code Editor* tab, you can control the appearance and behaviour of code editor widgets (Python interactive console and editor, expression widget and function editor, …).

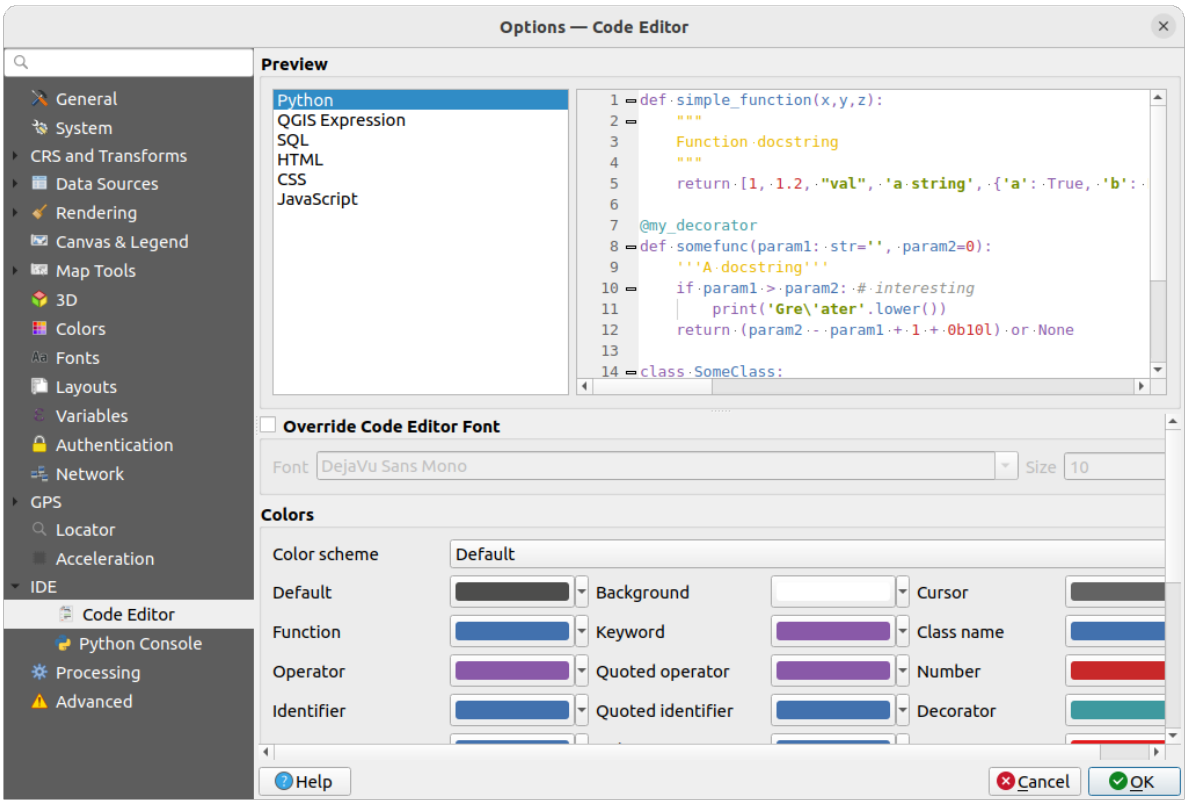


Fig. 9.27: Code Editor settings

At the top of the dialog, a widget provides a live preview of the current settings, in various coding languages (Python, QGIS expression, HTML, SQL, JavaScript). A convenient way to adjust settings.

- Check *Override Code Editor Font* to modify the default *Font* family and *Size*.
- Under the *Colors* group, you can:
  - select a *Color scheme*: predefined settings are `Default`, `Solarized Dark` and `Solarized Light`. A `Custom` scheme is triggered as soon as you modify a color and can be reset with selecting a predefined scheme.
  - change the *color* of each element in code writing, such as the colors to use for comments, quotes, functions, background, …

Python Console settings

The *Python Console* settings help you manage and control the behavior of the Python editors (*interactive console*, *code editor*, *project macros*, *custom expressions*, …). It can also be accessed using the Options… button from:

- the *Python console* toolbar
- the contextual menu of the *Python console* widget
- and the contextual menu of the code editor.

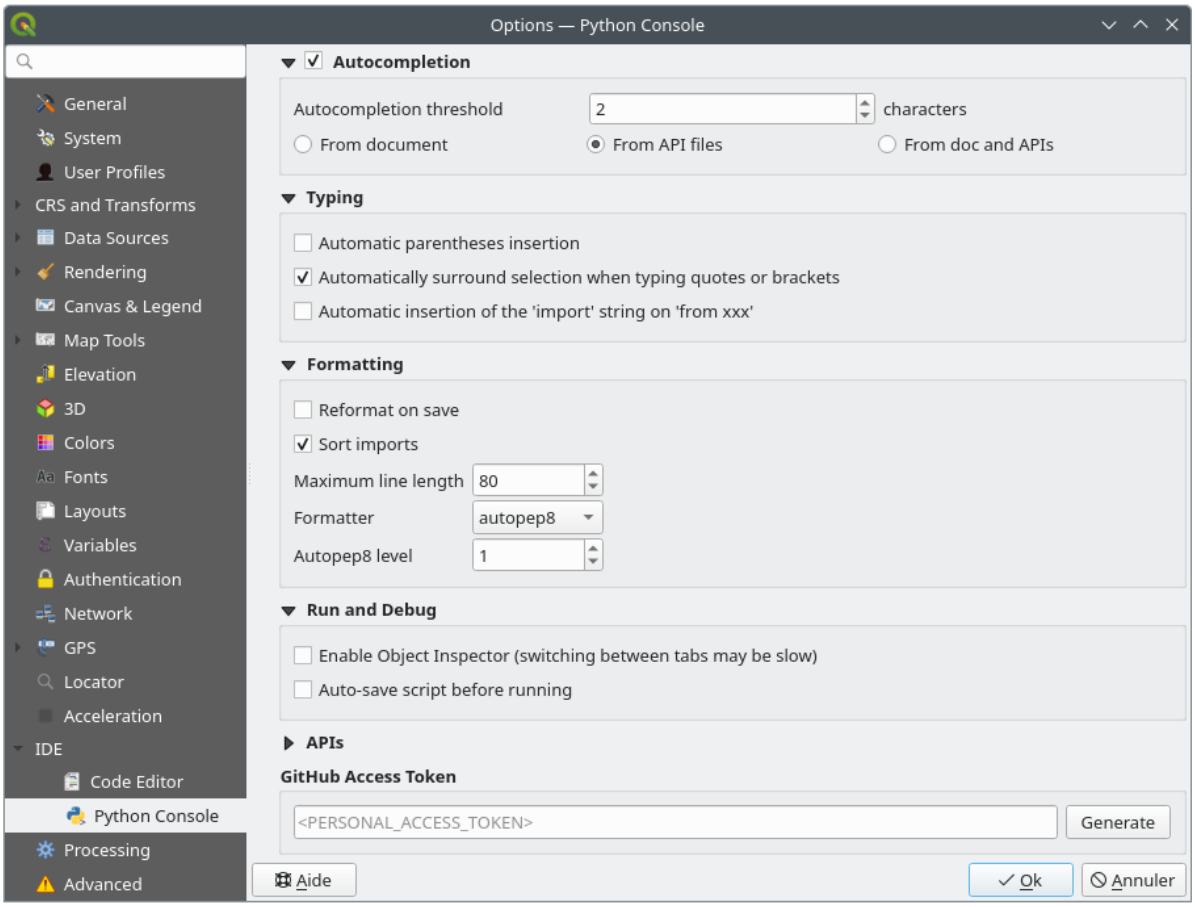


Fig. 9.28: Python Console settings

Puteți specifica:

- *Autocompletion*: Enables code completion. You can get autocompletion from the current document, the installed API files or both.
  - *Autocompletion threshold*: Sets the threshold for displaying the autocompletion list (in characters)
- under *Typing*
  - *Automatic parentheses insertion*: When no text is selected, if an opening character (parentheses, quotes, brackets, …) is entered, inserts the matching closing character just after the cursor. Note that this behavior is disabled if the current cursor is inside a string or comment.
  - *Automatically surround selection when typing quotes or brackets*: When an opening character is entered, the selected text is enclosed in the opening/closing pair. Selection remains the same, so it is possible to quote a selected word and enclose it in parentheses just by typing " then (.

Special case for multiline selection with quotes and double quotes: selection is enclosed in triple single/double quotes.

  - ☐ *Automatic insertion of the «import» string on «from xxx»*: Enables insertion of «import» when specifying imports

- under *Formatting*, you can add automated tools to reformat the code you are writing:
  - ☐ *Reformat on save*: formatting is applied just before saving the script
  - ☑ *Sort imports*: sorts «import» statements using the isort library
  - *Maximum line length*: controls how the formatter will wrap the lines, and controls the editor ruler
  - *Formatter* - supported tools are *autopep8* and *black*, with dedicated option:
    - *Autopep8 level* - more details at autopep8 advanced usage
    - ☐ *Normalize quotes*: replaces all single quotes with double quotes if possible
- under *Run and Debug*
  - ☐ *Enable Object Inspector (switching between tabs may be slow)*
  - ☐ *Auto-save script before running*: Saves the script automatically when executed. This action will store a temporary file (in the temporary system directory) that will be deleted automatically after running.

For *APIs* you can specify:

- ☑ *Using preloaded APIs file*: You can choose if you would like to use the preloaded API files. If this is not checked you can add API files and you can also choose if you would like to use prepared API files (see next option).
- ☐ *Using prepared APIs file*: If checked, the chosen `*.pap` file will be used for code completion. To generate a prepared API file you have to load at least one `*.api` file and then compile it by clicking the *Compile APIs…* button.

Under *GitHub access token*, you can generate a personal token allowing you to share code snippets from within the Python code editor. More details on GitHub authentication

9.1.21 Processing settings

The *Processing* tab provides you with general settings of tools and data providers that are used in the QGIS Processing framework. More information at *Cadrul de procesare al QGIS*.

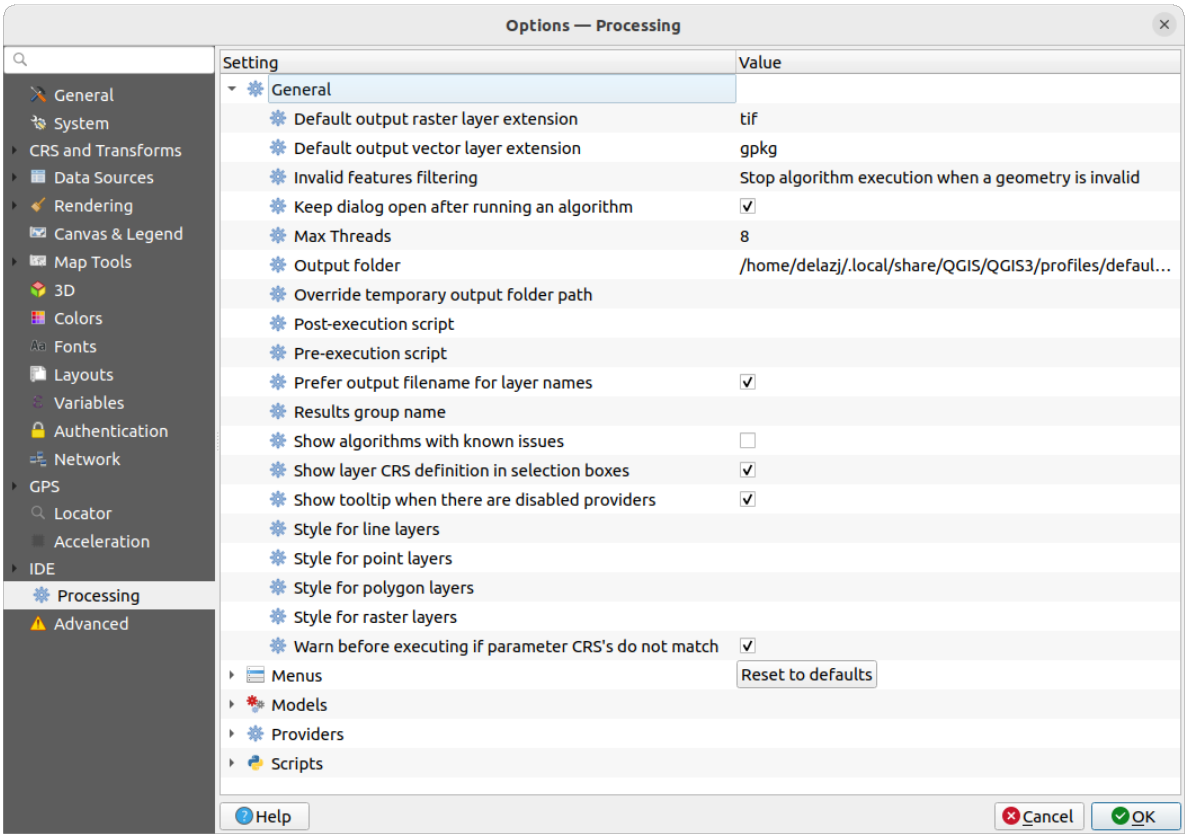


Fig. 9.29: Processing settings

9.1.22 Advanced settings

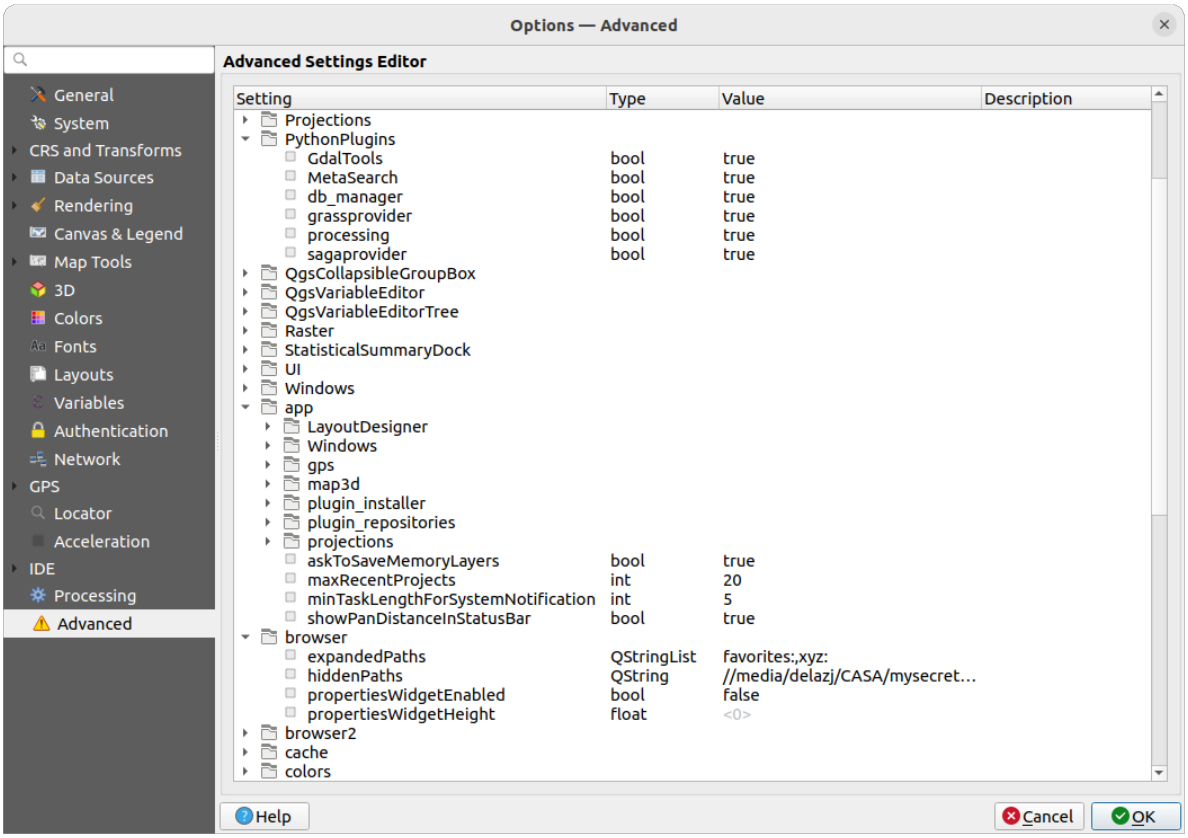


Fig. 9.30: Advanced settings

All the settings related to QGIS (UI, tools, data providers, Processing configurations, default values and paths, plugins options, expressions, geometry checks…) are saved in a `QGIS/QGIS3.ini` file under the active *user profile* directory. Configurations can be shared by copying this file to other installations.

From within QGIS, the *Advanced* tab offers a way to manage these settings through the *Advanced Settings Editor*. After you promise to be careful, the widget is populated with a tree of all the existing settings, and you can edit their value. Right-click over a setting or a group and you can delete it (to add a setting or group, you have to edit the `QGIS3.ini` file). Changes are automatically saved in the `QGIS3.ini` file.

Atenționare: Avoid using the Advanced tab settings blindly

Be careful while modifying items in this dialog given that changes are automatically applied. Doing changes without knowledge can break your QGIS installation in various ways.

9.2 Working with User Profiles

9.2.1 The concept

The *Settings* ► *User Profiles* menu provides functions to set and access user profiles. A user profile is a unified application configuration that allows to store in a single folder:

- all the *global settings*, including locale, projections, authentication settings, color palettes, shortcuts…
- GUI configurations and *customization*
- grid files and other proj helper files installed for datum transformation
- installed *plugins* and their configurations
- project templates and history of saved project with their image preview
- *processing settings*, logs, scripts, models.

By default, a QGIS installation contains a single user profile named `default`. But you can create as many user profiles as you want:

1. Click the *New profile…* entry.
2. You'll be prompted to provide a profile name, creating a folder of the same name under `~/<UserProfiles>/` where:
   - `~` represents the **HOME** directory, which on Windows is usually something like `C:\Users\<username>`.
   - and `<UserProfiles>` represents the main profiles folder, i.e.:
     - `.local/share/QGIS/QGIS3/profiles/`
     - `%AppData%\Roaming\QGIS\QGIS3\profiles\`
     - `Library/Application Support/QGIS/QGIS3/profiles/`

   The user profile folder can be opened from within QGIS using the *Open Active Profile Folder*.
3. A new instance of QGIS is started, using a clean configuration. You can then set your custom configurations.

If you have more than one profile in your QGIS installation, the name of the active profile is shown in the application title bar between square brackets.

As each user profile contains isolated settings, plugins and history they can be great for different workflows, demos, users of the same machine, or testing settings, etc. And you can switch from one to the other by selecting them in the *Settings* ► *User Profiles* menu. You can also run QGIS with a specific user profile from the *command line*.

Sfat: Run QGIS under a new user profile to check for bug persistence

When you encounter weird behavior with some functions in QGIS, create a new user profile and run the commands again. Sometimes, bugs are related to some leftovers in the current user profile and creating a new one may fix them as it restarts QGIS with the new (clean) profile.

9.2.2 Setting user profile

By default, QGIS opens a new session with the profile of the last closed session. This, among other settings, can be customized in *Settings* ► *Options* ► *User Profiles* tab:

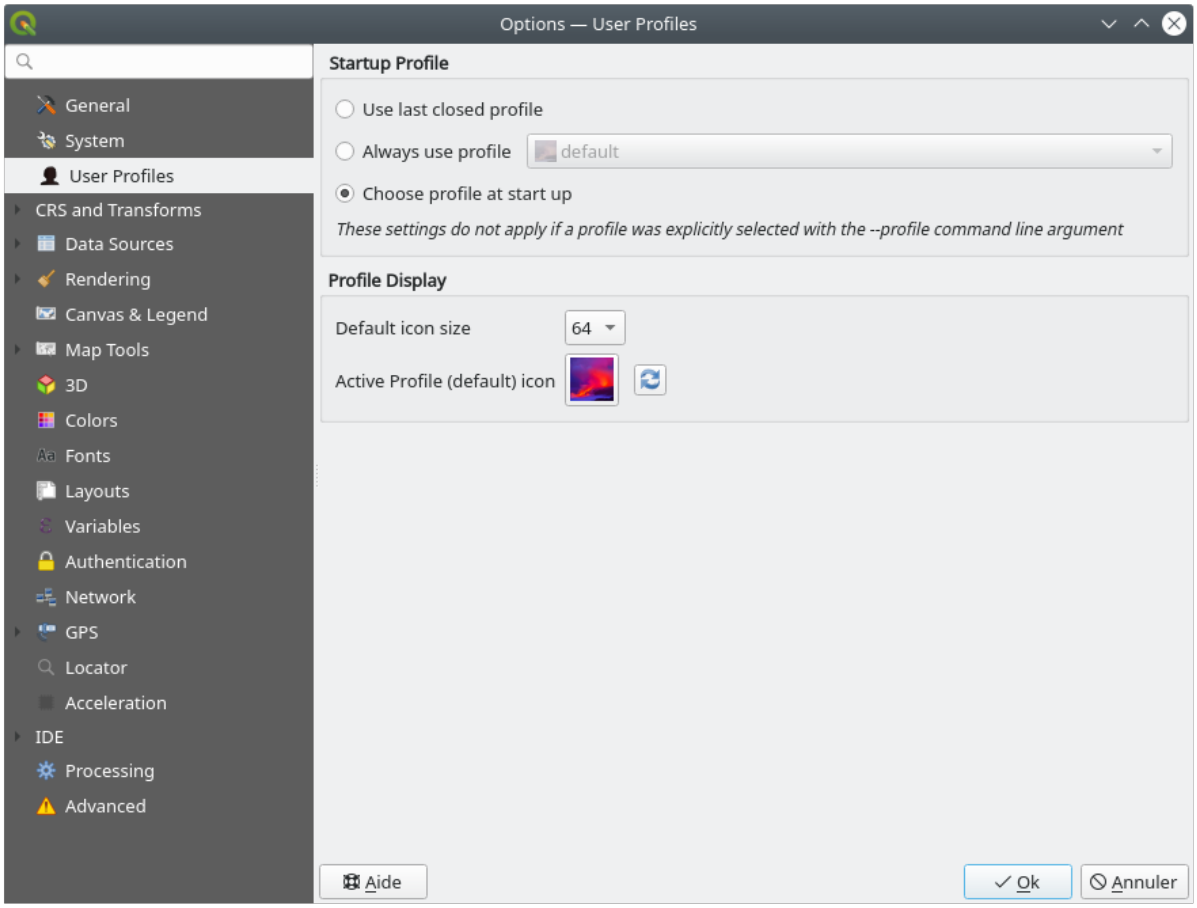


Fig. 9.31: User profiles settings

- *Startup profile*: indicates the user profile to use when starting a QGIS session. It can be:
  - *Use last closed profile*
  - a specific user profile to select from *Always use profile* drop-down menu
  - *Choose profile at start up*: Opens the *User Profile Selector* dialog listing available user profiles. Double-click an entry or select a profile and press *OK* to use that user profile for the starting session. It is also possible to *Add new profile* to the list: a profile is created from scratch and automatically run with the opening session.
- Under *Profile display*, you can set:
  - The size of icons to use when selecting a profile from the *User Profile Selector* dialog.
  - A specific icon to display next to the current profile in the *Settings* ► *User profiles* menu or the *User Profile Selector* dialog. Press Reset profile icon to remove custom modifications.

9.3 Proprietăţi Proiect

In the properties window for the project under *Project* ► *Project Properties*, you can set project-specific options. The project-specific options overwrite their equivalent in the *Options* dialog described above.

9.3.1 General Properties

In the *General* tab, the *General settings* let you:

- see the location of the project file
- set the folder for the project home (available in the *Project home* item of the *Browser* panel). The path can be relative to the folder of the project file (type it in) or absolute. The project home can be used for storing data and other content that is useful for the project. Convenient when dataset and project files are not stored at the same place. If not filled, the *Project home* defaults to the project file folder.
- give a title to the project beside the project file path
- choose the color to use for features when they are selected
- choose the background color: the color to use for the map canvas
- set whether the path to layers in the project should be saved as absolute (full) or as relative to the project file location. You may prefer relative path when both layers and project files can be moved or shared or if the project is accessed from computers on different platforms.
- choose to avoid artifacts when project is rendered as map tiles. Note that checking this option can lead to performance degradation.
- *Remember attribute tables windows and docks between sessions*: If checked for a project, then any opened attribute tables will be saved into the project and immediately restored when loading that project. This can improve workflows when you constructed a project with a particular set of attribute table configurations for your requirements, and re-setting up these attribute tables is a hassle.

Calculating areas and distances is a common need in GIS. However, these values are really tied to the underlying projection settings. The *Measurements* frame lets you control these parameters. You can indeed choose:

- the *Ellipsoid*, on which distance and area calculations are entirely based; it can be:
  - **None/Planimetric**: returned values are in this case cartesian measurements. This option can be set as default for new projects from the *Settings* ► *Options* ► *CRS Handling* menu
  - a **Custom** one: you'll need to set values of the semi-major and semi-minor axes.
  - or an existing one from a predefined list (Clarke 1866, Clarke 1880 IGN, New International 1967, WGS 84…).
- the *Units for distance measurements* for length and perimeter, and the *Units for area measurements*. These settings which default to their corresponding *global options* override them in the current project. They are used in:
  - Bara de actualizare a câmpului din tabela de atribute
  - Operațiunile din calculatorul de câmpuri
  - *Identify tool* derived length, perimeter and area values
  - *measure dialog*
  - *scale bar decoration*

The *Coordinate and Bearing display* allows you to customize the display of:

- the coordinates shown in the *Coordinates* box on QGIS status bar and in the *Derived* section of the tool's results

- the bearing value displayed in the status bar for the map canvas panning direction and by the Measure bearing tool.

Parametrii disponibili sunt:

- *Display coordinates using* either:
  - `Map Units`, based on the project CRS
  - `Map Geographic (degrees)`: based on the project CRS if it is of geographic type, otherwise uses its associated geographic CRS. This is helpful e.g. for non-earth celestial bodies.
  - or `Custom Projection Units`: allows to rely on any CRS you desire for coordinates display
- In the *Coordinate CRS* option, you can view or define the CRS to use depending on your display mode.
- *Coordinate format*: you can configure it as `Decimal Degrees`, `Degrees, Minutes` or `Degrees, Minutes, Seconds`, and whether it should display:
  - ☐ *Show directional suffix*
  - ☐ *Show leading zeros for minutes and seconds*
  - ☐ *Show leading zeros for degrees*
  - ☐ *Show trailing zeros*
- *Coordinate precision*: the number of decimal places can be automatic (derived from the type of CRS) or set manually
- *Coordinate order*: you can opt to display the coordinates in the native order of the CRS (`Default`) or switch it to either `Easting, Northing (Longitude, Latitude)` or `Northing, Easting (Latitude, Longitude)` order
- *Bearing format* possible values are `0 to 180°, with E/W suffix`, `-180 to +180°` or `0 to 360°`. The number of *Decimal places* as well as whether to *Show trailing zeros* can be set.

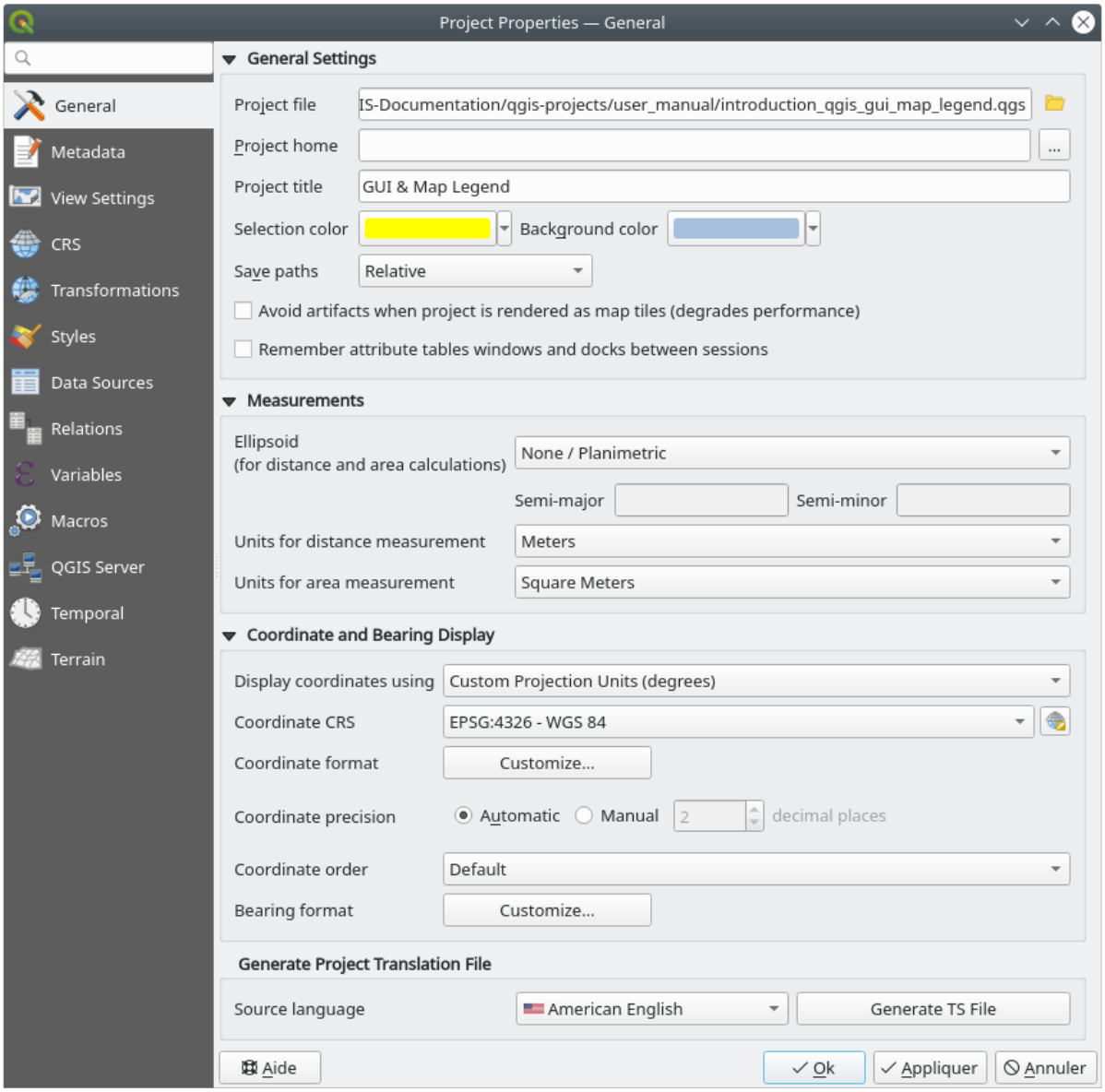


Fig. 9.32: General tab of the Project Properties dialog

9.3.2 Proprietățile Metadatelor

The *Metadata* tab allows detailed metadata to be defined, including (among the others): author, creation date, language, abstracts, categories, keywords, contact details, links, history. There is also a validation functionality that checks if specific fields were filled, anyway this is not enforced. See *vector layer metadata properties* for some details.

9.3.3 View Settings

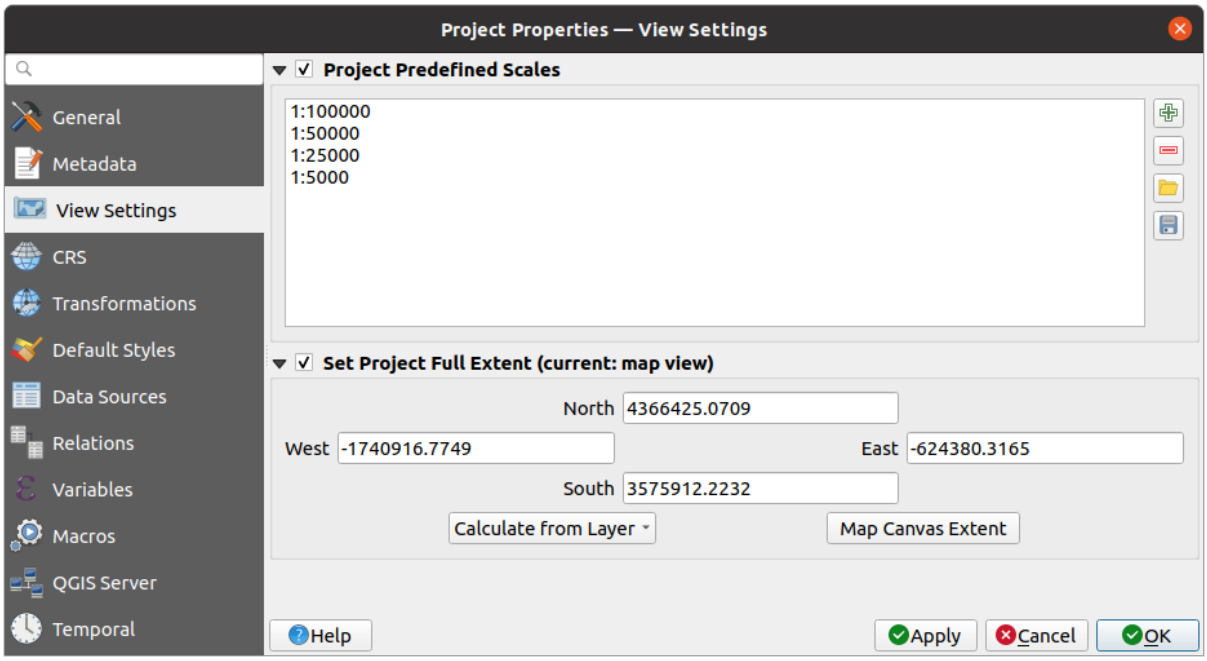


Fig. 9.33: View Settings tab of the Project Properties dialog

The *View Settings* tab provides means to control the project map canvas. You can:

- set *Project predefined scales*: the list of scales to display in scale-related drop-down widgets, such as the status bar *Scale*, the visibility scales selector or secondary 2D map view settings,… in replacement of the global *predefined scales*.
- *Set Project full Extent*: this extent will be used instead of the extent of all layers when zooming to full map extent ( ). It's useful when a project contains web layers/national layers/global layers yet the actual area of interest for the project is a smaller geographic area. The project full extent coordinates can be set with the *extent selector* widget.

9.3.4 Proprietăți CRS

Notă: For more information on how QGIS handles project projection, please read the dedicated section at *Lucrul cu Proiecții*.

The *CRS* tab helps you set the coordinate reference system to use in this project. It can be:

- *No CRS (or unknown/non-Earth projection)*: layers are drawn based on their raw coordinates
- or an existing coordinate reference system that can be *geographic*, *projected* or *user-defined*. Layers added to the project are translated on-the-fly to this CRS in order to overlay them regardless their original CRS.

9.3.5 Transformations Properties

The *Transformations* tab helps you control the layers reprojection settings by configuring the datum transformation preferences to apply in the current project. As usual, these override any corresponding global settings. See *Datum Transformations* for more details.

9.3.6 Styles Properties

Under *Styles* tab, you can configure symbols and colors inherent to the project, allowing to safely share the project among different machines.

The *Default Symbols* group lets you control how new layers will be drawn in the project when they do not have an existing `.qml` style defined. You can set *Marker*, *Line*, *Fill* to apply depending on the layer geometry type as well as default *Color Ramp* and *Text Format* (e.g. when enabling labeling). Any of these items can be reset using the *Clear* entry from the corresponding drop-down widget.

In the *Options* group, you can:

- Apply a default *Opacity* to new layers
- *Assign random colors to symbols*, modifying the symbols fill colors, hence avoiding same rendering for all layers.

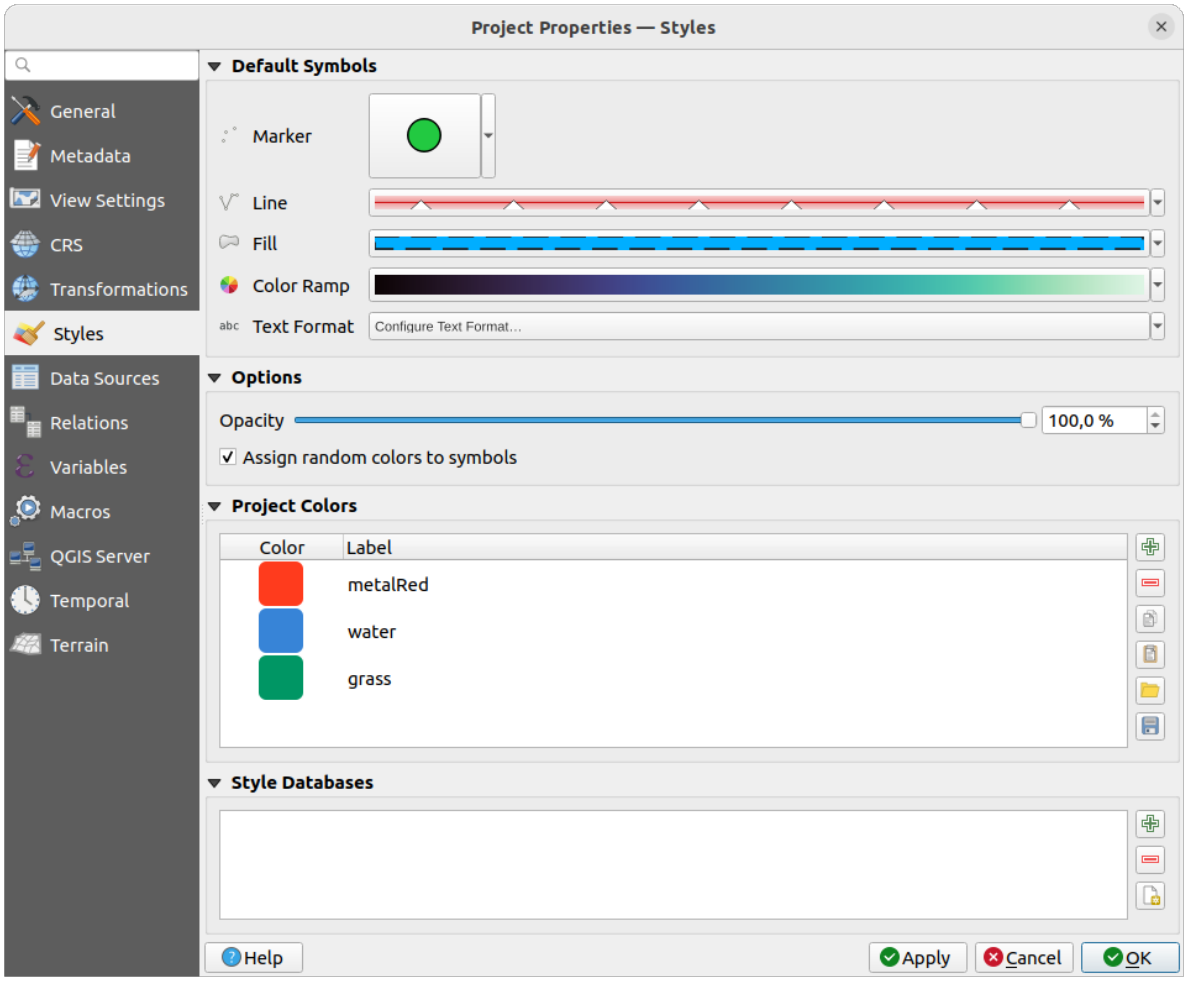


Fig. 9.34: Styles tab

There is also an additional section where you can define specific colors for the running project. Like the *global colors*, you can:

- *Add* or *Remove* color
- *Copy* or *Paste* color
- *Import* or *Export* the set of colors from/to `.gpl` file.

Double-click a color in the list to tweak or replace it in the *Color Selector* dialog. You can also rename it by double-clicking in the *Label* column.

These colors are identified as *Project colors* and listed as part of *color widgets*.

Sfat: Use project colors to quickly assign and update color widgets

Project colors can be refered to using their label and the color widgets they are used in are bound to them. This means that instead of repeatedly setting the same color for many properties and, to avoid a cumbersome update you can:

1. Define the color as a project color
2. Click the *data defined override widget* next to the color property you want to set
3. Hover over the *Color* menu and select the project color. The property is then assigned the expression `project_color('color_label')` and the color widget reflects that color.
4. Repeat steps 2 and 3 as much as needed
5. Update the project color once and the change is reflected EVERYWHERE it's in use.

9.3.7 Data Sources Properties

In the *Data Sources* tab, you can:

- *Transaction mode*, defines how edits are sent to the data provider:
  - *Local Edit Buffer*: edits are buffered locally and sent to the provider when toggling layer editing mode or clicking *Save edits*.
  - *Automatic Transaction Groups*: on supported datasources (postgres and geopackage databases) the edit state of all tables that originate from the same database are synchronized and executed in a server side transaction. Also, instead of buffering edit changes locally, they are directly sent to a transaction in the database which gets committed when toggling layer editing mode or clicking *Save edits*.
  - *Buffered Transaction Groups*: all editable layers, disregarding from which provider, are toggled synchronously and all edits are saved in a local edit buffer. Saving changes is executed within a single transaction on all layers (per provider).

  Note that you can change this option only if no layer is being edited in the project.
- *Evaluate default values on provider side*: When adding new features in a PostgreSQL table, fields with default value constraint are evaluated and populated at the form opening, and not at the commit moment. This means that instead of an expression like `nextval('serial')`, the field in the *Add Feature* form will display expected value (e.g., `25`).
- *Remember editable layer status between sessions*: makes sure that all layers that are editable in a project will be remembered as such when saving the project, as well as making sure that those layers are immediately made editable whenever the project is restored.
- Configure the *Layers Capabilities*, i.e.:
  - Set (or disable) which layers are `identifiable`, i.e. will respond to the *identify tool*. By default, layers are set queryable.

  - Set whether a layer should appear as `read-only`, meaning that it can not be edited by the user, regardless of the data provider's capabilities. Although this is a weak protection, it remains a quick and handy configuration to avoid end-users modifying data when working with file-based layers.
  - Define which layers are `searchable`, i.e. could be queried using the *locator widget*. By default, layers are set searchable.
  - Define which layers are defined as `required`. Checked layers in this list are protected from inadvertent removal from the project.
  - Define which layers are `private`, i.e. hidden from the *Layers* panel. This is meant for accessory layers (basemap, join, lookups for value-relations, most probably aspatial layers, …) that you still need in a project but you don't want them to pollute the legend tree and other layer selection tools. If set visible, they are still displayed in the map canvas and rendered in the print layout legend. Use the *Filter legend* ► *Show private layers* option in the *Layers* panel top toolbar to temporarily turned them on for any interaction.

  The *Layers Capabilities* table provides some convenient tools to:

  - Select multiple cells and press *Toggle Selection* to have them change their checkbox state;
  - *Show spatial layers only*, filtering out non-spatial layers from the layers list;
  - *Filter layers…* and quickly find a particular layer to configure.

- Under the *Advanced Settings* group, you can select *Trust project when data source has no metadata*: To speed up project loading by skipping data checks. Useful in QGIS Server context or in projects with huge database views/materialized views. The extent of layers will be read from the QGIS project file (instead of data sources) and when using the PostgreSQL provider the primary key unicity will not be checked for views and materialized views.

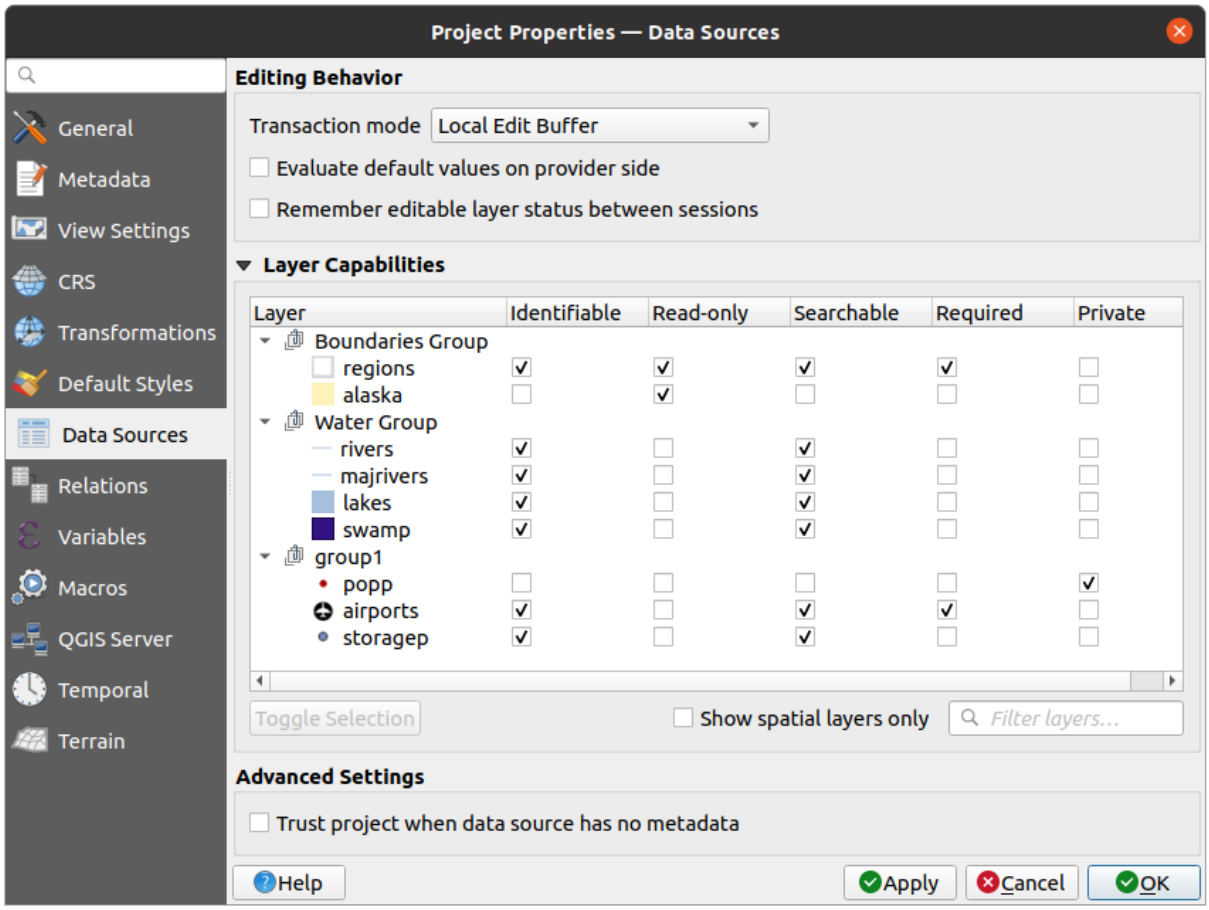


Fig. 9.35: Data Sources tab

9.3.8 Relations Properties

The *Relations* tab is used to define relations between layers. The relations can be of one to one, many to many or polymorphic type. They are defined in the project properties dialog. Once relations exist for a layer, a new user interface element in the form view (e.g. when identifying a feature and opening its form) will list the related entities. This provides a powerful way to express e.g. the inspection history on a length of pipeline or road segment. You can find out more about relations support in section *Setting relations between multiple layers*.

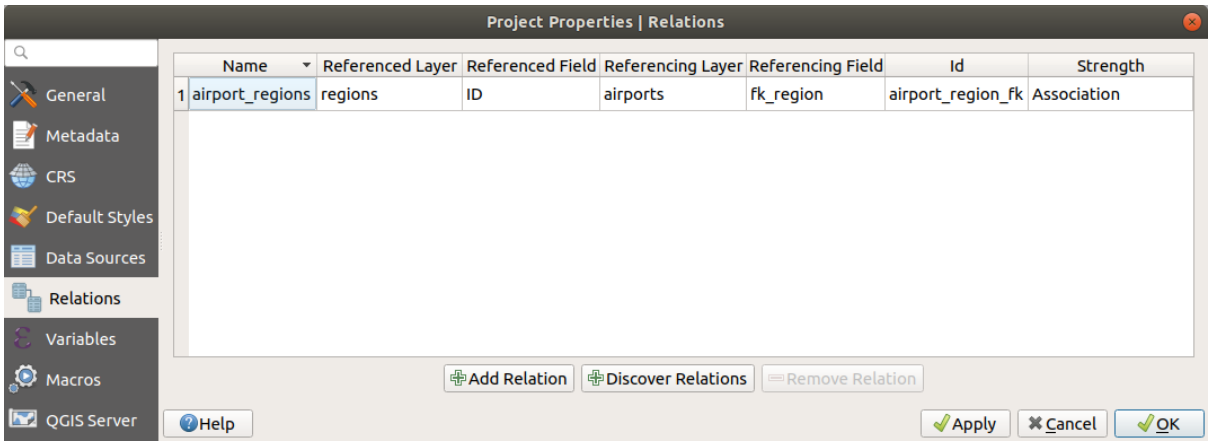


Fig. 9.36: Relations tab

9.3.9 Proprietățile Variabilelor

The *Variables* tab lists all the variables available at the project's level (which includes all global variables). Besides, it also allows the user to manage project-level variables. Click the button to add a new custom project-level variable. Likewise, select a custom project-level variable from the list and click the button to remove it. More information on variables usage in the General Tools *Storing values in Variables* section.

9.3.10 Macros Properties

The *Macros* tab is used to edit Python macros for projects. Currently, only three macros are available: `openProject()`, `saveProject()` and `closeProject()`.

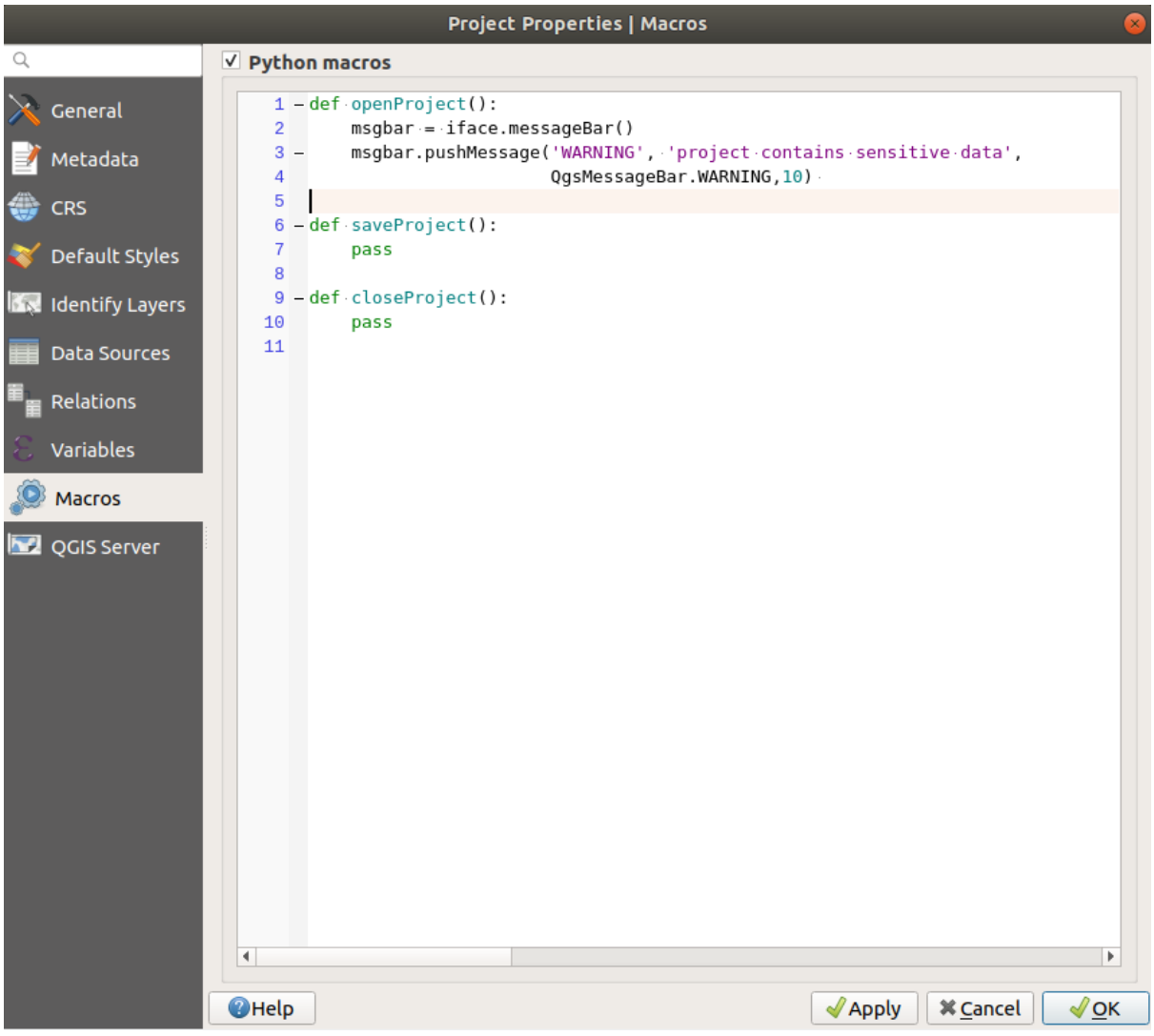


Fig. 9.37: Setări macro

9.3.11 Proprietățile Serverului QGIS

The *QGIS Server* tab allows you to configure your project in order to publish it online. Here you can define information about the QGIS Server WMS and WFS capabilities, extent and CRS restrictions. More information available in section Creatingwmsfromproject and subsequent.

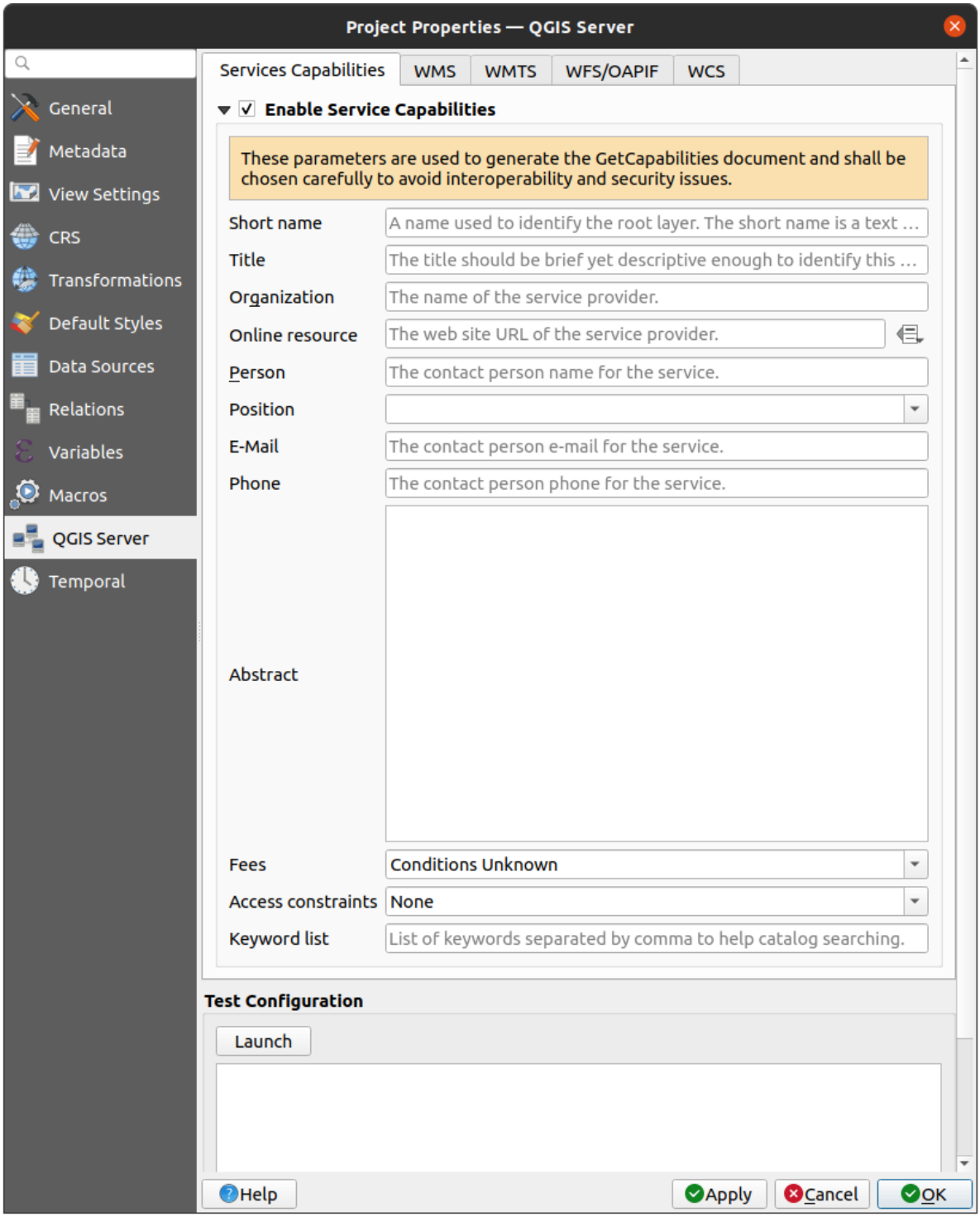


Fig. 9.38: QGIS Server settings

9.3.12 Proprietăți Temporale

The *Temporal* tab is used to set the temporal range of your project, either by using manual *Start date* and *End date* inputs or by calculating it from the current project temporal layers. The project time range can then be used in the *Temporal controller panel* to manage the map canvas *temporal navigation*.

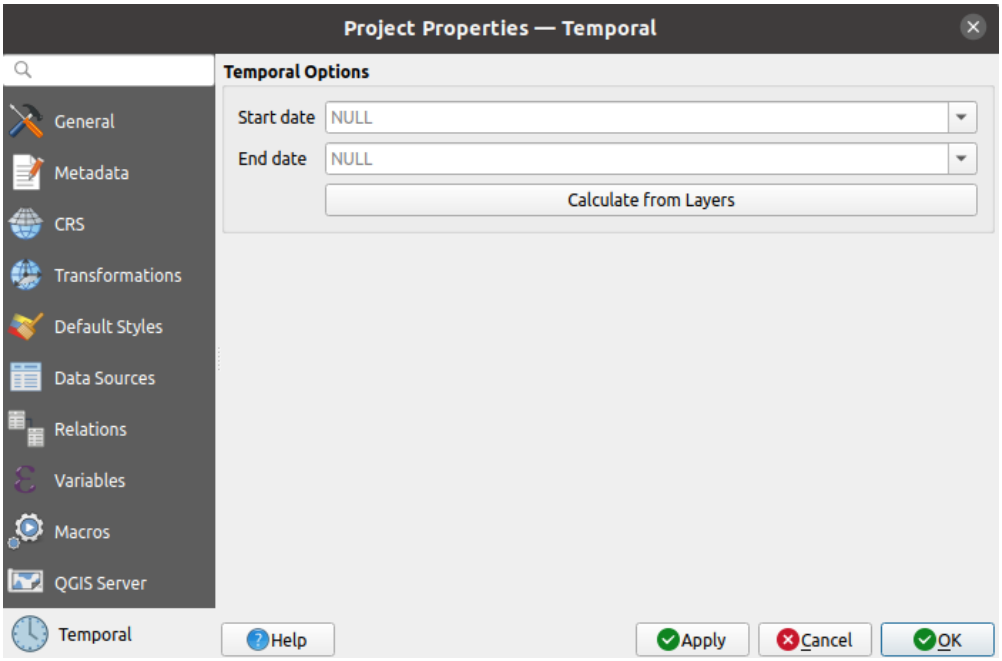


Fig. 9.39: Project Temporal tab

9.3.13 Terrain Properties

The *Terrain* tab helps you configure default settings for the terrain and elevation. When any new *3d map* is created in the project, the map will default to using the same terrain settings as are defined for the project. The project elevation settings will also be respected by the *elevation profile* tool.

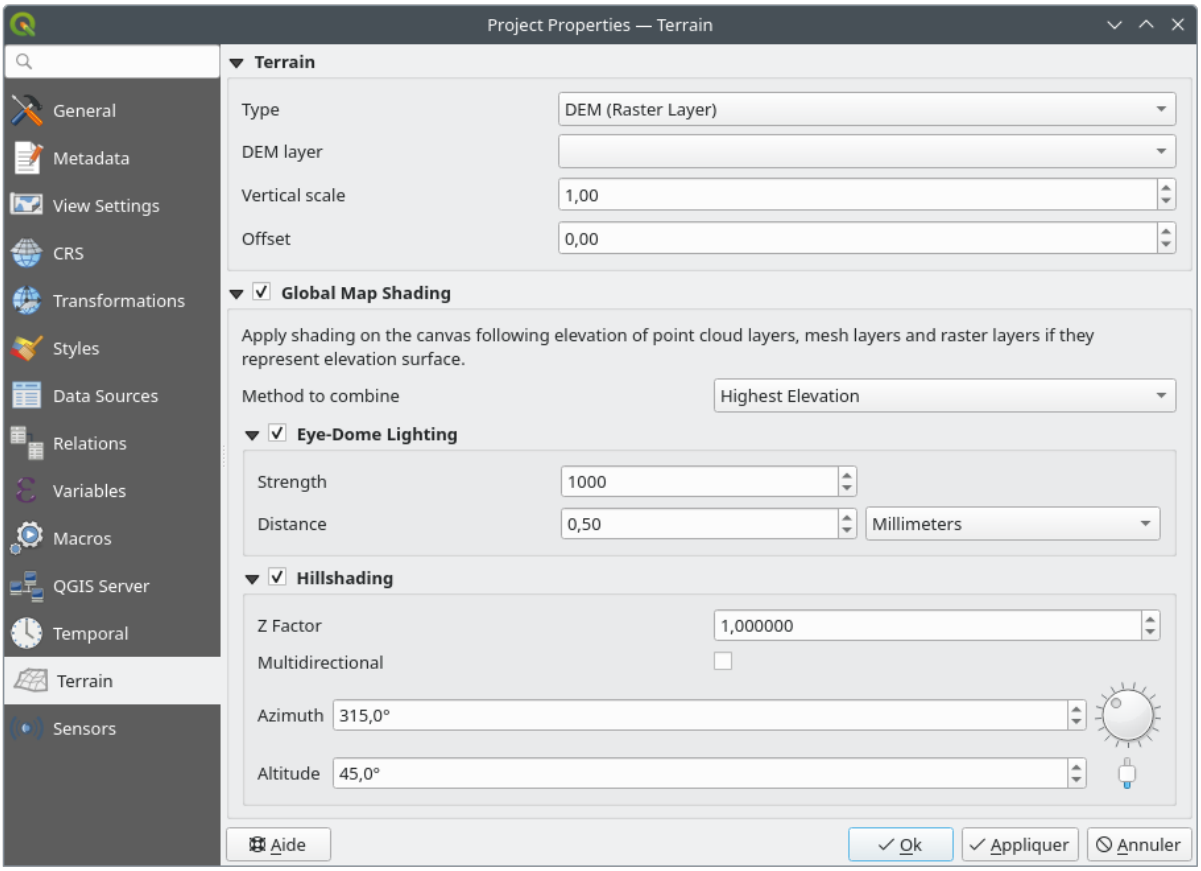


Fig. 9.40: Project Terrain tab

- *Terrain* and elevation options are available for:
  - *Flat terrain* with *Terrain height* setting
  - *DEM (Raster Layer)*: with setting for defining the *Raster layer*, a *Vertical scale* factor to apply to band values and a vertical *Offset*
  - *Mesh*: with setting for defining the *Mesh layer*, a *Vertical scale* factor to apply to vertices Z value and a vertical *Offset*

  These settings can be overwritten from the 3D map *configuration dialog*.
- With *Global map shading* settings, you apply a global shading effect to the map, based on the elevation of all the active layers that have elevation properties enabled, including:
  - *raster layers*
  - *mesh layers*
  - *point cloud layers*

  Method to combine: the elevation value at any particular position for all the relevant layers are combined, and the considered elevation is chosen depending on one of these methods:
  - *Highest elevation* value
  - *Based on layer's order*: the elevation on the topmost layer in the layer tree is considered.

  Supported shading options are:
  - *Eye-dome lighting*: applies shading effects to the map canvas for a better depth rendering. Following parameters can be controlled:
    - *Strength*: increases the contrast, allowing for better depth perception

    - *Distance*: represents the distance of the used pixels off the center pixel and has the effect of making edges thicker.
  - *Hillshading*, shaping some reliefs on the map using shading (levels of gray):
    - *Z Factor*: Scaling factor for the elevation value
    - *Multidirectional*: Specify if multidirectional hillshading is to be used
    - *Azimuth*: The azimuth of the light source
    - *Altitude*: The elevation angle of the light source

Notă: A shortcut to *Global map shading* properties is available through the *Layer Styling* panel.

9.3.14 Sensors Properties

The *Sensors* tab is used to configure sensors and toggle their connection status. When active, sensors will passively collect data in the background and make available their latest data to expressions and python scripts.

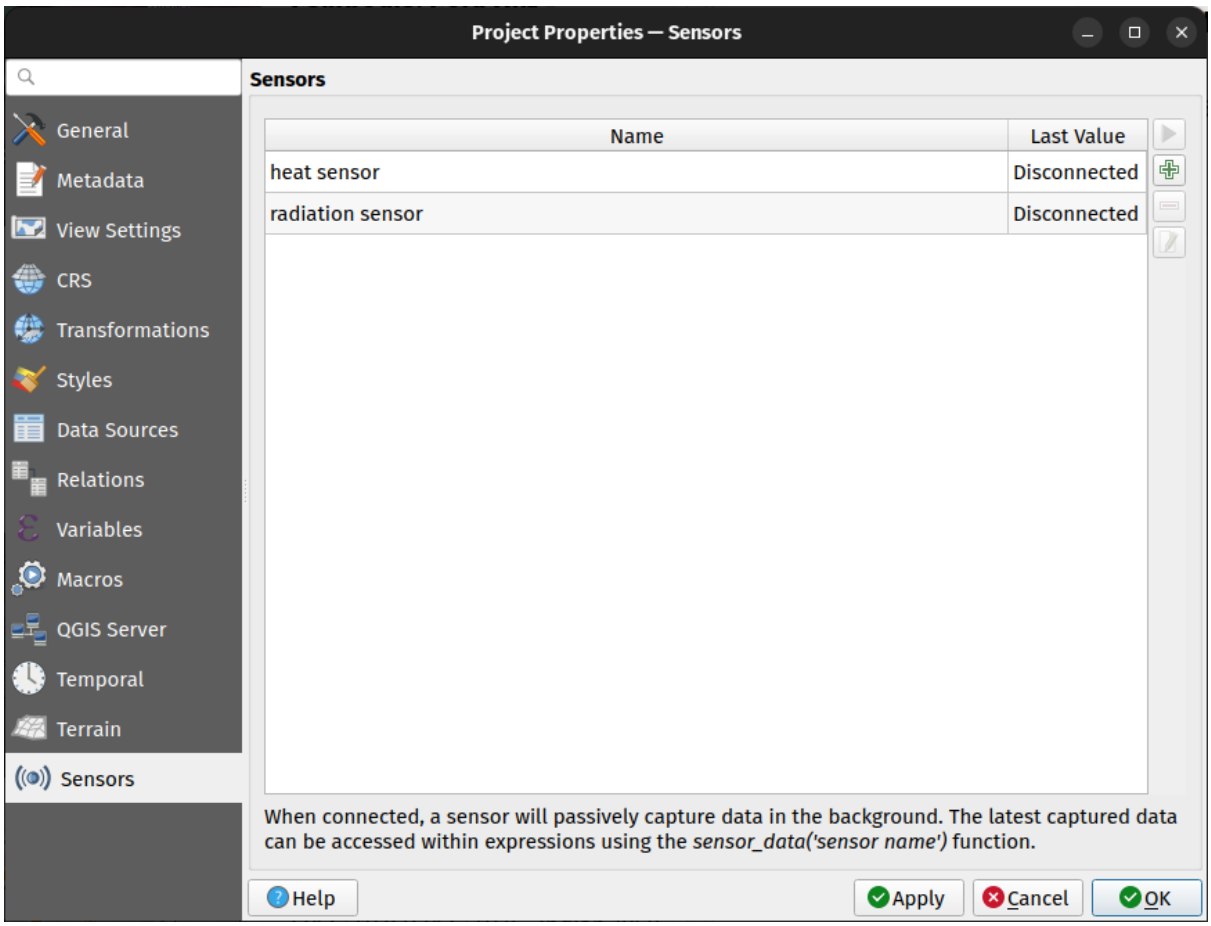


Fig. 9.41: Project Sensors tab

To add a new sensor, click on the button. A settings sub-panel will open and allow you to configure:

- the *Sensor name*: used to retrieve sensor values in expressions and python scripts;
- the *Sensor type*: TCP, UDP, serial port, etc.; and

- additional type-specific details (e.g. host name and port)

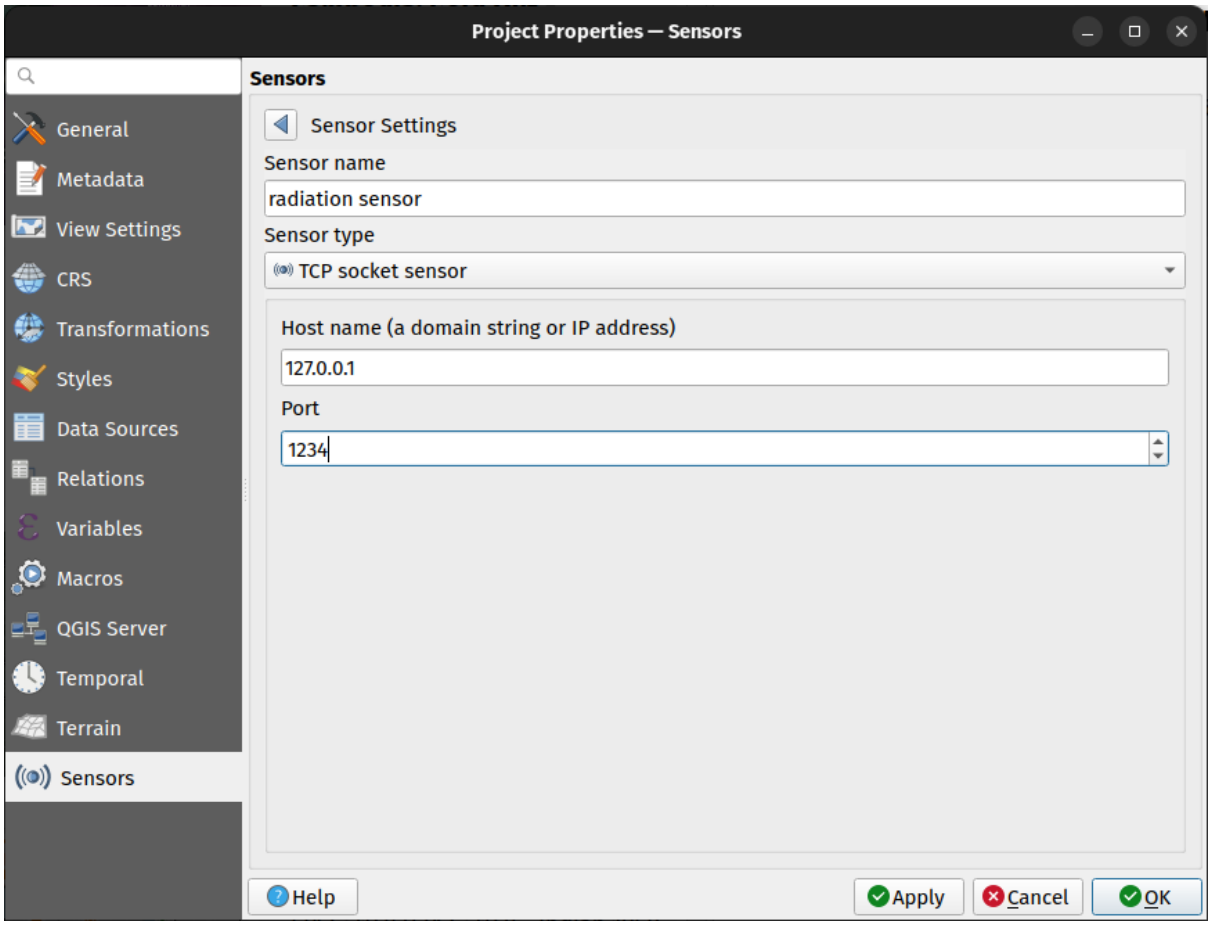


Fig. 9.42: Sensor Settings sub-panel

Once a sensor is configured, you can use the Start button to connect the sensor. Once active, the last collected data will be displayed in the sensors table's *Last value* column.

9.4 Personalizare

The *Customization* dialog lets you (de)activate almost every element in the QGIS user interface. This can be very useful if you want to provide your end-users with a «light» version of QGIS, containing only the icons, menus or panels they need.

Notă: Pentru aplicarea modificărilor, trebuie să reporniți QGIS.

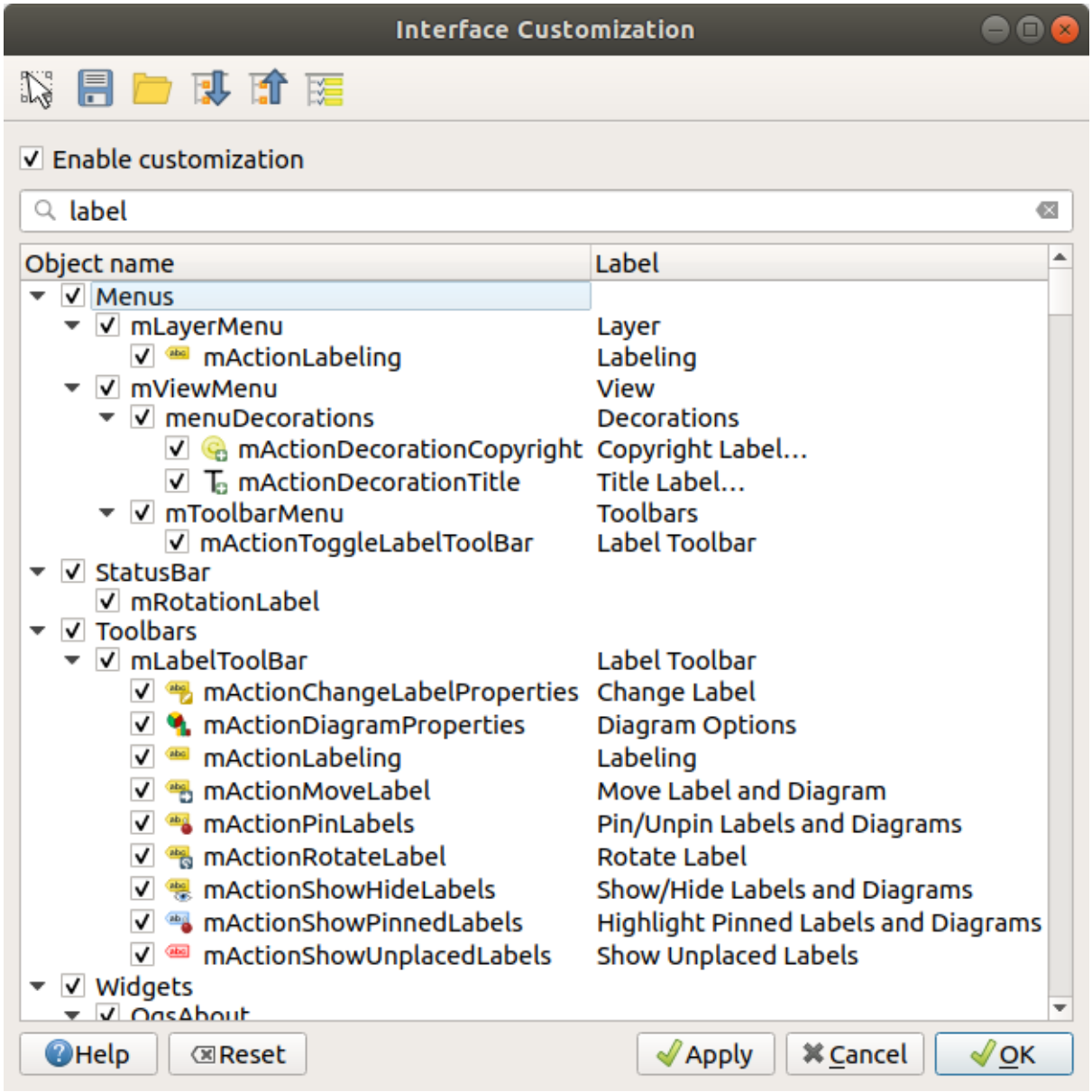


Fig. 9.43: Dialogul de Personalizare

Ticking the *Enable customization* checkbox is the first step on the way to QGIS customization. This enables the toolbar and the widget panel from which you can uncheck and thus disable some GUI items.

Elementul configurabil poate fi:

- a **Menu** or some of its sub-menus from the *Bara de Meniuri*
- un întreg **Panou** (see *Panouri și Bare de Instrumente*)
- the **Status bar** described in *Bara de Stare* or some of its items
- a **Toolbar**: the whole bar or some of its icons
- or any **widget** from any dialog in QGIS: label, button, combobox…

With Switch to catching widgets in main application, you can click on an item in QGIS interface that you want to be hidden and QGIS automatically unchecks the corresponding entry in the Customization dialog. You can also use the *Search* box to find items by their name or label.

Once you setup your configuration, click *Apply* or *OK* to validate your changes. This configuration becomes the one used by default by QGIS at the next startup.

The modifications can also be saved in a `.ini` file using Save To File button. This is a handy way to share a common QGIS interface among multiple users. Just click on Load from File from the destination computer in order to import the `.ini` file. You can also run *command line tools* and save various setups for different use cases as well.

Sfat: Restaurează cu ușurință setările QGIS predefinite

The initial QGIS GUI configuration can be restored by one of the methods below:

- unchecking *Enable customization* option in the Customization dialog or click the Check All button
- pressing the *Reset* button in the **Settings** frame under *Settings* ► *Options* menu, *System* tab
- launching QGIS at a command prompt with the following command line `qgis --nocustomization`
- setting to `false` the value of *UI* ► *Customization* ► *Enabled* variable under *Settings* ► *Options* menu, *Advanced* tab (see the *warning*).

In most cases, you need to restart QGIS in order to have the change applied.

9.5 Combinații de taste

QGIS provides default keyboard shortcuts for many features. You can find them in section *Bara de Meniuri*. Additionally, the menu option *Settings* ► *Keyboard Shortcuts...* allows you to change the default keyboard shortcuts and add new ones to QGIS features.

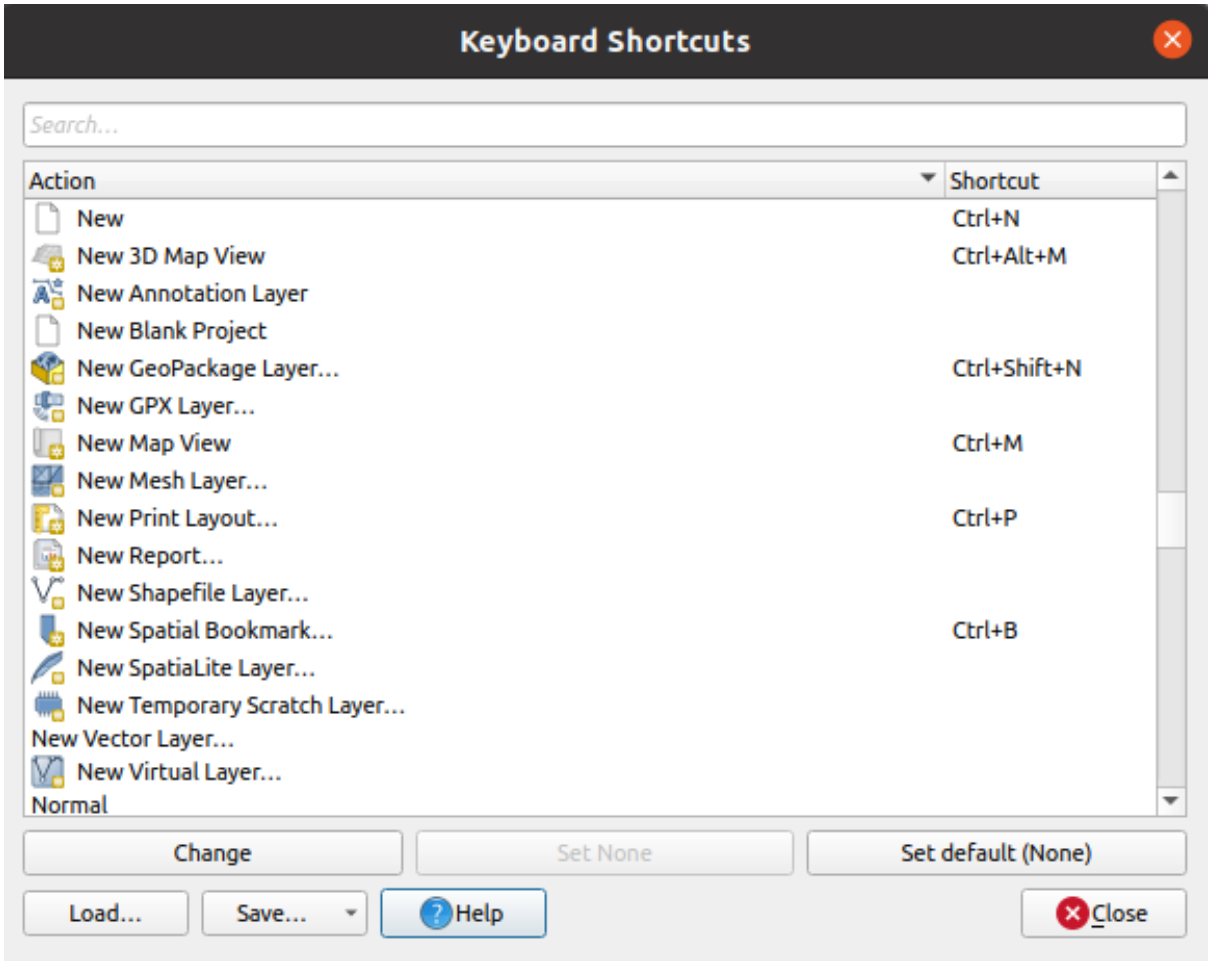


Fig. 9.44: Define shortcut options

Configuration is very simple. Use the search box at the top of the dialog to find a particular action, select it from the list and click on :

- *Change* and press the new combination you want to assign as new shortcut
- *Set None* to clear any assigned shortcut
- or *Set Default* to backup the shortcut to its original and default value.

Proceed as above for any other tools you wish to customize. Once you have finished your configuration, simply *Close* the dialog to have your changes applied. You can also *Save* the changes either as an `.XML` file with only the User Shortcuts or with all Shortcuts or as an `.PDF` file with all Shortcuts and *Load* them into another QGIS installation.

9.6 Running QGIS with advanced settings

9.6.1 Command line and environment variables

We've seen that *launching QGIS* is done as for any application on your OS. QGIS provides command line options for more advanced use cases (in some cases you can use an environment variable instead of the command line option). To get a list of the options, enter `qgis --help` on the command line, which returns:

```
QGIS is a user friendly Open Source Geographic Information System.
Usage: /usr/bin/qgis.bin [OPTION] [FILE]
  OPTION:
```

(continues on next page)

(continuare din pagina precedentă)

```
        [-v, --version]     display version information and exit
        [-s, --snapshot filename]   emit snapshot of loaded datasets to given file
        [-w, --width width] width of snapshot to emit
        [-h, --height height]       height of snapshot to emit
        [-l, --lang language]       use language for interface text (changes␣
↪existing override)
        [-p, --project projectfile] load the given QGIS project
        [-e, --extent xmin,ymin,xmax,ymax]  set initial map extent
        [-n, --nologo]      hide splash screen
        [-V, --noversioncheck]      don't check for new version of QGIS at startup
        [-P, --noplugins]   don't restore plugins on startup
        [-B, --skipbadlayers]     don't prompt for missing layers
        [-C, --nocustomization]     don't apply GUI customization
        [-z, --customizationfile path]      use the given ini file as GUI␣
↪customization
        [-g, --globalsettingsfile path]     use the given ini file as Global␣
↪Settings (defaults)
        [-a, --authdbdirectory path] use the given directory for authentication␣
↪database
        [-f, --code path]   run the given python file on load
        [-d, --defaultui]   start by resetting user ui settings to default
        [--hide-browser]        hide the browser widget
        [--dxf-export filename.dxf]     emit dxf output of loaded datasets to␣
↪given file
        [--dxf-extent xmin,ymin,xmax,ymax]      set extent to export to dxf
        [--dxf-symbology-mode none|symbollayer|feature] symbology mode for dxf␣
↪output
        [--dxf-scale-denom scale]       scale for dxf output
        [--dxf-encoding encoding]       encoding to use for dxf output
        [--dxf-map-theme maptheme]      map theme to use for dxf output
        [--take-screenshots output_path]        take screen shots for the user␣
↪documentation
        [--screenshots-categories categories]   specify the categories of␣
↪screenshot to be used (see QgsAppScreenShots::Categories).
        [--profile name]        load a named profile from the user's profiles␣
↪folder.
        [-S, --profiles-path path]  path to store user profile folders. Will␣
↪create profiles inside a {path}\profiles folder
        [--version-migration]   force the settings migration from older version if␣
↪found
        [--openclprogramfolder]         path to the folder containing the sources␣
↪for OpenCL programs.
        [--help]                this text
        [--]            treat all following arguments as FILEs

 FILE:
   Files specified on the command line can include rasters,
   vectors, and QGIS project files (.qgs and .qgz):
    1. Rasters - supported formats include GeoTiff, DEM
       and others supported by GDAL
    2. Vectors - supported formats include ESRI Shapefiles
       and others supported by OGR and PostgreSQL layers using
       the PostGIS extension
```

Sfat: Example Using command line arguments

You can start QGIS by specifying one or more data files on the command line. For example, assuming you are in the `qgis_sample_data` directory, you could start QGIS with a vector layer and a raster file set to load on startup using the following command: `qgis ./raster/landcover.img ./gml/lakes.gml`

--version

This option returns QGIS version information.

--snapshot

This option allows you to create a snapshot in PNG format from the current view. This comes in handy when you have many projects and want to generate snapshots from your data, or when you need to create snapshots of the same project with updated data.

Currently, it generates a PNG file with 800x600 pixels. The size can be adjusted using the `--width` and `--height` arguments. The filename can be added after `--snapshot`. For example:

```
qgis --snapshot my_image.png --width 1000 --height 600 --project my_project.qgs
```

--width

This option returns the width of the snapshot to be emitted (used with `--snapshot`).

--height

This option returns the height of the snapshot to be emitted (used with `--snapshot`).

--lang

Based on your locale, QGIS selects the correct localization. If you would like to change your language, you can specify a language code. For example, `qgis --lang it` starts QGIS in Italian localization.

--project

Starting QGIS with an existing project file is also possible. Just add the command line option `--project` followed by your project name and QGIS will open with all layers in the given file loaded.

--extent

To start with a specific map extent use this option. You need to add the bounding box of your extent in the following order separated by a comma:

```
--extent xmin,ymin,xmax,ymax
```

This option probably makes more sense when paired with the `--project` option to open a specific project at the desired extent.

--nologo

This option hides the splash screen when you start QGIS.

--noversioncheck

Skip searching for a new version of QGIS at startup.

--noplugins

If you have trouble at start-up with plugins, you can avoid loading them at start-up with this option. They will still be available from the Plugins Manager afterwards.

--nocustomization

Using this option, any existing *GUI customization* will not be applied at startup. This means that any hidden buttons, menu items, toolbars, and so on, will show up on QGIS start up. This is not a permanent change. The customization will be applied again if QGIS is launched without this option.

This option is useful for temporarily allowing access to tools that have been removed by customization.

--skipbadlayers

Using this option, you can avoid QGIS prompting the *Handle Unavailable Layers* dialog at startup. The project file is loaded, with missing layers kept as unavailable. More details on the topic at *Gestionarea căilor de fișiere întrerupte*.

--customizationfile

Using this option, you can define a UI customization file, that will be used at startup.

--globalsettingsfile

The equivalent environment variable is `QGIS_GLOBAL_SETTINGS_FILE`.

Using this option, you can specify the path for a Global Settings file (`.ini`), also known as the Default Settings. The settings in the specified file replace the original inline default ones, but the user profiles» settings will be set on top of those.

QGIS looks for the default global settings file in the following order and only the first found file will be used:

- path specified by the commandline parameter
- path defined by the environment variable
- the AppDataLocation folder, where persistent application data can be stored; it is managed by the user or system administrator and is not touched by installer and does not require any additional setup like passing commandline parameters or settings environment variable. Depending on the OS, it is:
  - `$HOME/.local/share/QGIS/QGIS3/`
  - `C:\Users\<username>\%AppData%\Roaming\QGIS\QGIS3\`
  - `$HOME/Library/Application Support/QGIS/QGIS3/`
- the installation directory, i.e. `your_QGIS_package_path/resources/qgis_global_settings.ini`.

Presently, there's no way to specify a file to write settings to; therefore, you can create a copy of an original settings file, rename, and adapt it.

Setting the `qgis_global_setting.ini` file path to a network shared folder, allows a system administrator to change global settings and defaults in several machines by only editing one file.

--authdbdirectory

This option is similar to `--globalsettingsfile`, but defines the path to the directory where the authentication database will be stored and loaded.

--code

This option can be used to run a given python file directly after QGIS has started.

For example, when you have a python file named `load_alaska.py` with following content:

```
from qgis.utils import iface
raster_file = "/home/gisadmin/Documents/qgis_sample_data/raster/landcover.img"
layer_name = "Alaska"
iface.addRasterLayer(raster_file, layer_name)
```

Assuming you are in the directory where the file `load_alaska.py` is located, you can start QGIS, load the raster file `landcover.img` and give the layer the name «Alaska» using the following command:

```
qgis --code load_alaska.py
```

--defaultui

On load, **permanently resets** the user interface (UI) to the default settings. This option will restore the panels and toolbars visibility, position, and size. Unless it's changed again, the default UI settings will be used in the following sessions.

Notice that this option doesn't have any effect on *GUI customization*. Items hidden by GUI customization (e.g. the status bar) will remain hidden even using the `--defaultui` option. See also the `--nocustomization` option.

--hide-browser

On load, hides the *Browser* panel from the user interface. The panel can be enabled by right-clicking a space in the toolbars or using the *View* ► *Panels* (*Settings* ► *Panels* in Linux KDE).

Unless it's enabled again, the Browser panel will remain hidden in the following sessions.

--dxf-*

These options can be used to export a QGIS project into a DXF file. Several options are available:

- *–dxf-export*: the DXF filename into which to export the layers;
- *–dxf-extent*: the extent of the final DXF file;
- *–dxf-symbology-mode*: several values can be used here: `none` (no symbology), `symbollayer` (Symbol layer symbology), `feature` (feature symbology);
- *–dxf-scale-denom*: the scale denominator of the symbology;
- *–dxf-encoding*: the file encoding;
- *–dxf-map-theme*: choose a *map theme* from the layer tree configuration.

--take-screenshots

Takes screenshots for the user documentation. Can be used together with `--screenshots-categories` to filter which categories/sections of the documentation screenshots should be created (see QgsAppScreenShots::Categories).

--profile

Loads QGIS using a specific profile from the user's profile folder. This option takes precedence over *user profile startup setting*.

--profiles-path

With this option, you can choose a path to load and save the profiles (user settings). It creates profiles inside a `{path}\profiles` folder, which includes settings, installed plugins, processing models and scripts, and so on.

This option allows you to, for instance, carry all your plugins and settings in a flash drive, or, for example, share the settings between different computers using a file sharing service.

The equivalent environment variable is `QGIS_CUSTOM_CONFIG_PATH`.

--version-migration

If settings from an older version are found (*e.g.*, the `.qgis2` folder from QGIS 2.18), this option will import them into the default QGIS profile.

--openclprogramfolder

Using this option, you can specify an alternative path for your OpenCL programs. This is useful for developers while testing new versions of the programs without needing to replace the existing ones.

The equivalent environment variable is `QGIS_OPENCL_PROGRAM_FOLDER`.

9.6.2 Deploying QGIS within an organization

If you need to deploy QGIS within an organization with a custom configuration file, first you need to copy/paste the content of the default settings file located in `your_QGIS_package_path/resources/qgis_global_settings.ini`. This file already contains some default sections identified by a block starting with `[]`. We recommend that you keep these defaults values and add your own sections at the bottom of the file. If a section is duplicated in the file, QGIS will take the last one from top to bottom.

You can change `allowVersionCheck=false` to disable the QGIS version check.

If you do not want to display the migration window after a fresh install, you need the following section:

```
[migration]
fileVersion=2
settings=true
```

If you want to add a custom variable in the global scope:

```
[variables]
organisation="Your organization"
```

A lot of settings can be set using the `.INI` file such as WMS/WMTS, PostGIS connections, proxy settings, maptips, default values for algorithms parameters… For example, if you want to configure organization wide default values for some algorithms it would be handy if you could configure it in the prepared ini-file for all users. If the users then open the toolbox to execute an algorithm, the default values will be there. To set the default value e.g. for `GRID_SIZE` parameter for the `native:intersection` algorithm, you could write the following into the `.INI` file:

```
[Processing]
DefaultGuiParam\native%3Aintersection\GRID_SIZE=0.01
```

Not all settings can be prepared from QGIS GUI, but to discover the possibilities of the settings `.INI` file, we suggest that you set the config you would like in QGIS Desktop and then search for it in your `.INI` file located in your profile using a text editor.

Finally, you need to set the environment variable `QGIS_GLOBAL_SETTINGS_FILE` to the path of your customized file.

In addition, you can also deploy files such as Python macros, color palettes, layout templates, project templates… either in the QGIS system directory or in the QGIS user profile.

- Layout templates must be deployed in the `composer_templates` directory.
- Project templates must be deployed in the `project_templates` directory.
- Custom Python macros must be deployed in the `python` directory.

CAPITOLUL 10

Lucrul cu Proiecții

A Coordinate Reference System, or CRS, is a method of associating numerical coordinates with a position on the surface of the Earth. QGIS has support for approximately 7,000 standard CRSs, each with different use cases, pros and cons! Choosing an appropriate reference system for your QGIS projects and data can be a complex task, but fortunately QGIS helps guide you through this choice, and makes working with different CRSs as transparent and accurate as possible.

10.1 Privire de ansamblu asupra Suportului Proiecției

QGIS has support for approximately 7,000 known CRSs. These standard CRSs are based on those defined by the European Petroleum Search Group (EPSG) and the Institut Geographique National de France (IGNF), and are made available in QGIS through the underlying „Proj" projection library. Commonly, these standard projections are identified through use of an authority:code combination, where the authority is an organisation name such as „EPSG" or „IGNF", and the code is a unique number associated with a specific CRS. For instance, the common WGS 84 latitude/longitude CRS is known by the identifier `EPSG:4326`, and the web mapping standard CRS is `EPSG:3857`.

Custom, user-created CRSs are stored in a user CRS database. See section *Sistem de Referință și Coordonate Personalizat* for information on managing your custom coordinate reference systems.

10.2 Layer Coordinate Reference Systems

In order to correctly project data into a specific target CRS, either your data must contain information about its coordinate reference system or you will need to manually assign the correct CRS to the layer. For PostGIS layers, QGIS uses the spatial reference identifier that was specified when that PostGIS layer was created. For data supported by GDAL, QGIS relies on the presence of a recognized means of specifying the CRS. For instance, for the Shapefile format this is a file containing an ESRI Well-Known Text (WKT) representation of the layer's CRS. This projection file has the same base name as the `.shp` file and a `.prj` extension. For example, `alaska.shp` would have a corresponding projection file named `alaska.prj`.

Whenever a layer is loaded into QGIS, QGIS attempts to automatically determine the correct CRS for that layer. In some cases this is not possible, e.g. when a layer has been provided without retaining this information. You can configure QGIS behavior whenever it cannot automatically determine the correct CRS for a layer:

1. Open *Settings* ► *Options...* ► *CRS*

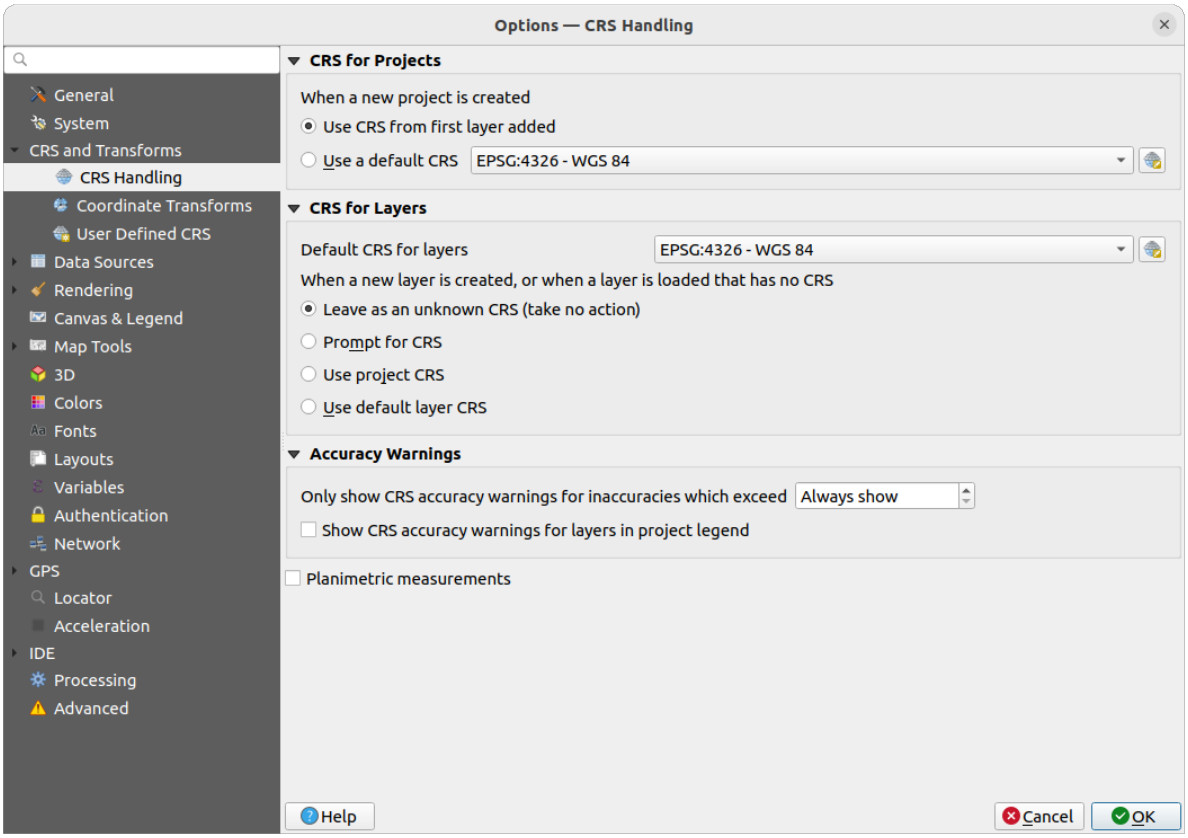


Fig. 10.1: The CRS tab in the QGIS Options Dialog

2. Under the *CRS for layers* group, set the action to do *when a new layer is created, or when a layer is loaded that has no CRS*. One of:

   - *Leave as unknown CRS (take no action)*: there will be no prompt to select a CRS when a layer without CRS is loaded, defering CRS choice to a later time. Convenient when loading a lot of layers at once. Such layers will be identifiable in the *Layers* panel by the icon next to them. They'll also be un-referenced, with coordinates from the layer treated as purely numerical, non-earth values, i.e. the same behavior as all layers get when *a project is set to have no CRS*.
   - *Prompt for CRS*: it will prompt you to manually select the CRS. Selecting the correct choice is crucial, as a wrong choice will place your layer in the wrong position on the Earth's surface! Sometimes, accompanying metadata will describe the correct CRS for a layer, in other cases you will need to contact the original author of the data to determine the correct CRS to use.
   - *Utilizează CRS-ul proiectului*
   - *Use default layer CRS*, as set in the *Default CRS for layers* combobox above.

Sfat: To assign the same CRS to multiple layers that have no crs or have a wrong one in one operation:

1. Select the layers in the *Layers* panel
2. Press `Ctrl+Shift+C`. You could also right-click over one of the selected layers or go to *Layer* ► *Set CRS of layer(s)*
3. Find and select the right CRS to use
4. And press *OK*. You can confirm that it has been set correctly in the *Source* tab of the layers» properties dialog.

Note that changing the CRS in this setting does not alter the underlying data source in any way, rather it just changes how QGIS interprets the raw coordinates from the layer in the current QGIS project.

10.3 Project Coordinate Reference Systems

Every project in QGIS also has an associated Coordinate Reference System. The project CRS determines how data is projected from its underlying raw coordinates to the flat map rendered within your QGIS map canvas.

QGIS supports „on the fly" CRS transformation for both raster and vector data. This means that regardless of the underlying CRS of particular map layers in your project, they will always be automatically transformed into the common CRS defined for your project. Behind the scenes, QGIS transparently reprojects all layers contained within your project into the project's CRS, so that they will all be rendered in the correct position with respect to each other!

It is important to make an appropriate choice of CRS for your QGIS projects. Choosing an inappropriate CRS can cause your maps to look distorted, and poorly reflect the real-world relative sizes and positions of features. Usually, while working in smaller geographic areas, there will be a number of standard CRSs used within a particular country or administrative area. It's important to research which CRSs are appropriate or standard choices for the area you are mapping, and ensure that your QGIS project follows these standards.

By default, QGIS starts each new project using a global default projection. This default CRS is `EPSG:4326` (also known as „WGS 84"), and it is a global latitude/longitude based reference system. This default CRS can be changed via the *CRS for New Projects* setting in the *CRS* tab under *Settings* ► *Options…* (see Fig. 10.1). There is an option to automatically set the project's CRS to match the CRS of the first layer loaded into a new project, or alternatively you can select a different default CRS to use for all newly created projects. This choice will be saved for use in subsequent QGIS sessions.

The project CRS can also be set through the *CRS* tab of the *Project* ► *Properties…* dialog. It will also be shown in the lower-right of the QGIS status bar.

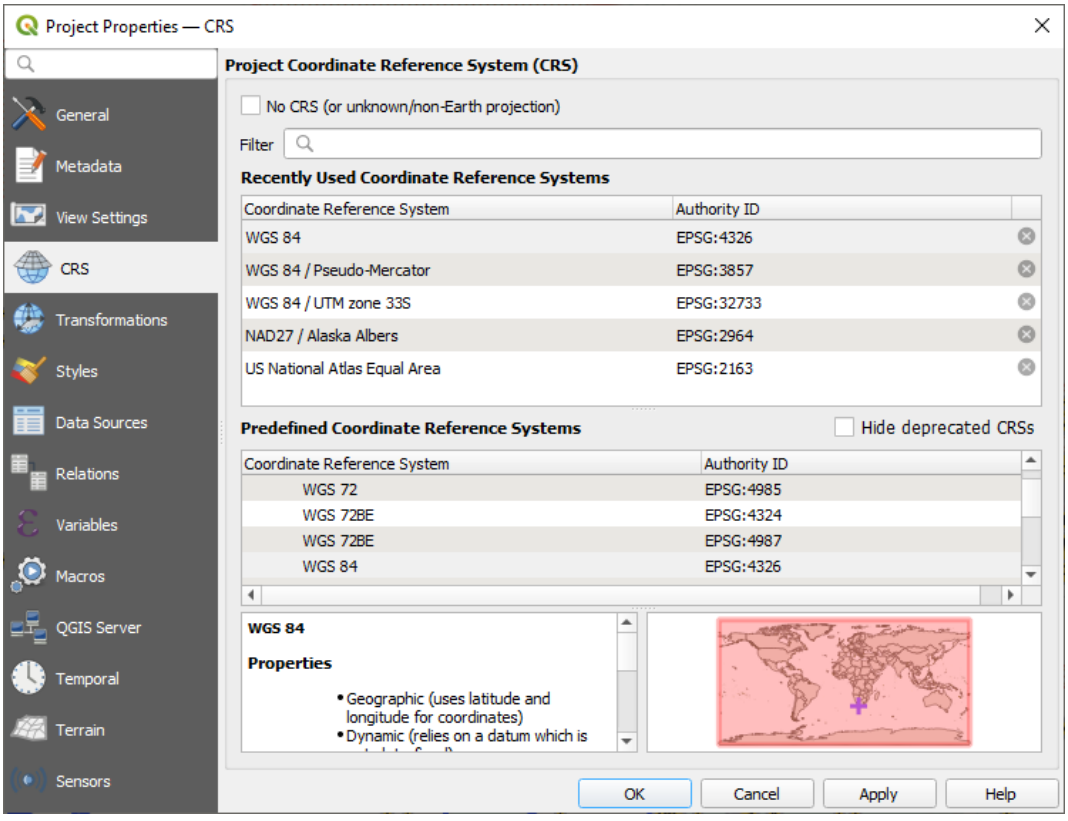


Fig. 10.2: Dialogul Proprietăților de Proiecție

Available options are:

- *No CRS (or unknown/non-Earth projection)*: Checking this setting will disable ALL projection handling within the QGIS project, causing all layers and map coordinates to be treated as simple 2D Cartesian coordinates, with no relation to positions on the Earth's surface. It can be used to guess a layer CRS (based on its raw coordinates or when using QGIS for non earth uses like role-playing game maps, building mapping or microscopic stuff. In this case:
  - No reprojection is done while rendering the layers: features are just drawn using their raw coordinates.
  - The ellipsoid is locked out and forced to `None/Planimetric`.
  - The distance and area units, and the coordinate display are locked out and forced to „unknown units"; all measurements are done in unknown map units, and no conversion is possible.
- or an existing coordinate reference system that can be *geographic*, *projected* or *user-defined*. A preview of the CRS extent on earth is displayed to help you select the appropriate one. Layers added to the project are translated on-the-fly to this CRS in order to overlay them regardless of their original CRS. Use of units and ellipsoid setting are available and make sense and you can perform calculations accordingly.

Whenever you select a new CRS for your QGIS project, the measurement units will automatically be changed in the *General* tab of the *Project properties* dialog (*Project* ► *Properties…*) to match the selected CRS. For instance, some CRSs define their coordinates in feet instead of meters, so setting your QGIS project to one of these CRSs will also set your project to measure using feet by default.

Sfat: Setting the project CRS from a layer

You can assign a CRS to the project using a layer CRS:

1. In the *Layers* panel, right-click on the layer you want to pick the CRS
2. Select *Set project CRS from Layer*.

The project's CRS is redefined using the layer's CRS. Map canvas extent, coordinates display are updated accordingly and all the layers in the project are on-the-fly translated to the new project CRS.

10.4 Coordinate Reference System Selector

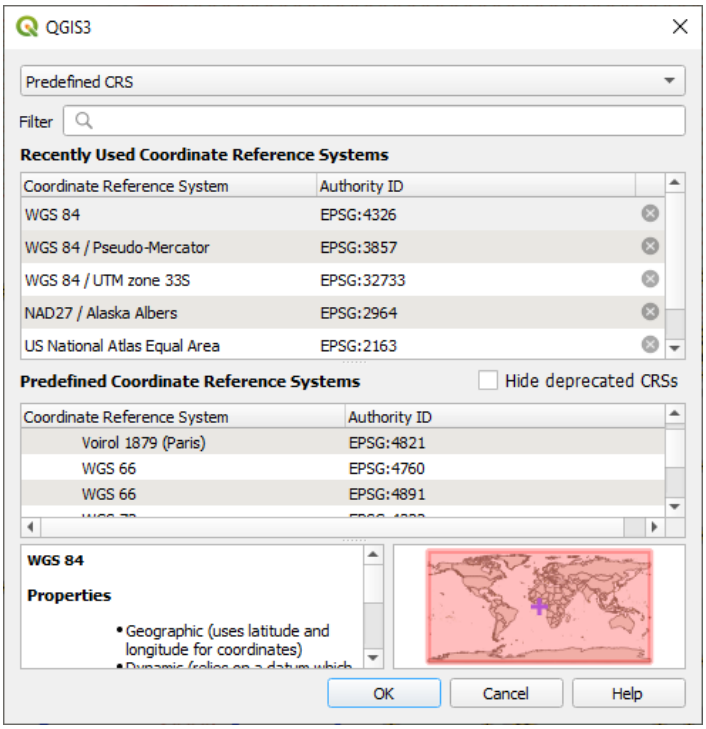


Fig. 10.3: CRS Selector

This dialog helps you assign a Coordinate Reference System to a project or a layer, provided a set of projection databases. Items in the dialog are:

- **Filter**: If you know the EPSG code, the identifier, or the name for a Coordinate Reference System, you can use the search feature to find it. Enter the EPSG code, the identifier or the name.
- **Recently used coordinate reference systems**: If you have certain CRSs that you frequently use in your everyday GIS work, these will be displayed in this list. Click on one of these items to select the associated CRS.
- **Coordinate reference systems of the world**: This is a list of all CRSs supported by QGIS, including Geographic, Projected and Custom coordinate reference systems. To define a CRS, select it from the list by expanding the appropriate node and selecting the CRS. The active CRS is preselected.
- **PROJ text**: This is the CRS string used by the PROJ projection engine. This text is read-only and provided for informational purposes.

If you want to remove a recently used CRS from the list that can be done in several ways:

- Press Remove from recently used CRS button on the right side.
- Press `Del` after selecting CRS that you want to remove.
- Right-click on used CRS and choose *Remove selected CRS from recently used CRS* to remove selected CRS or *Clear all recently used CRS* to remove all used CRS.

The CRS selector also shows a rough preview of the geographic area for which a selected CRS is valid for use. Many CRSs are designed only for use in small geographic areas, and you should not use these outside of the area they were designed for. The preview map shades an approximate area of use whenever a CRS is selected from the list. In addition, this preview map also shows an indicator of the current main canvas map extent.

10.5 Sistem de Referință și Coordonate Personalizat

If QGIS does not provide the coordinate reference system you need, you can define a custom CRS. To define a CRS, select *Custom CRS...* from the *Settings* menu. Custom CRSs are stored in your QGIS user database. In addition to your custom CRSs, this database also contains your spatial bookmarks and other custom data.

Defining a custom CRS in QGIS requires a good understanding of the PROJ projection library. To begin, refer to „Cartographic Projection Procedures for the UNIX Environment - A User's Manual" by Gerald I. Evenden, U.S. Geological Survey Open-File Report 90-284, 1990 (available at https://pubs.usgs.gov/of/1990/of90-284/ofr90-284.pdf).

This manual describes the use of `proj` and related command line utilities. The cartographic parameters used with `proj` are described in the user manual and are the same as those used by QGIS.

Pentru a defini un CRS personalizat, fereastra de *Definire a Sistemului de Referință și Coordonate* necesită doar doi parametri:

1. Un nume descriptiv
2. The cartographic parameters in PROJ or WKT format

To create a new CRS:

1. Click the Add new CRS button
2. Enter a descriptive name
3. Select the format: it can be *Proj String* or *WKT*
4. Add the CRS *Parameters*.

   Notă: Prefer storing the CRS definition in WKT format

   Although both `Proj String` and `WKT` formats are supported, it's highly recommended to store projection definitions in the WKT format. Therefore, if the available definition is in the proj format, select that format, enter the parameters and then switch to WKT format. QGIS will convert the definition to the WKT format that you can later save.

5. Click *Validate* to test whether the CRS definition is an acceptable projection definition.

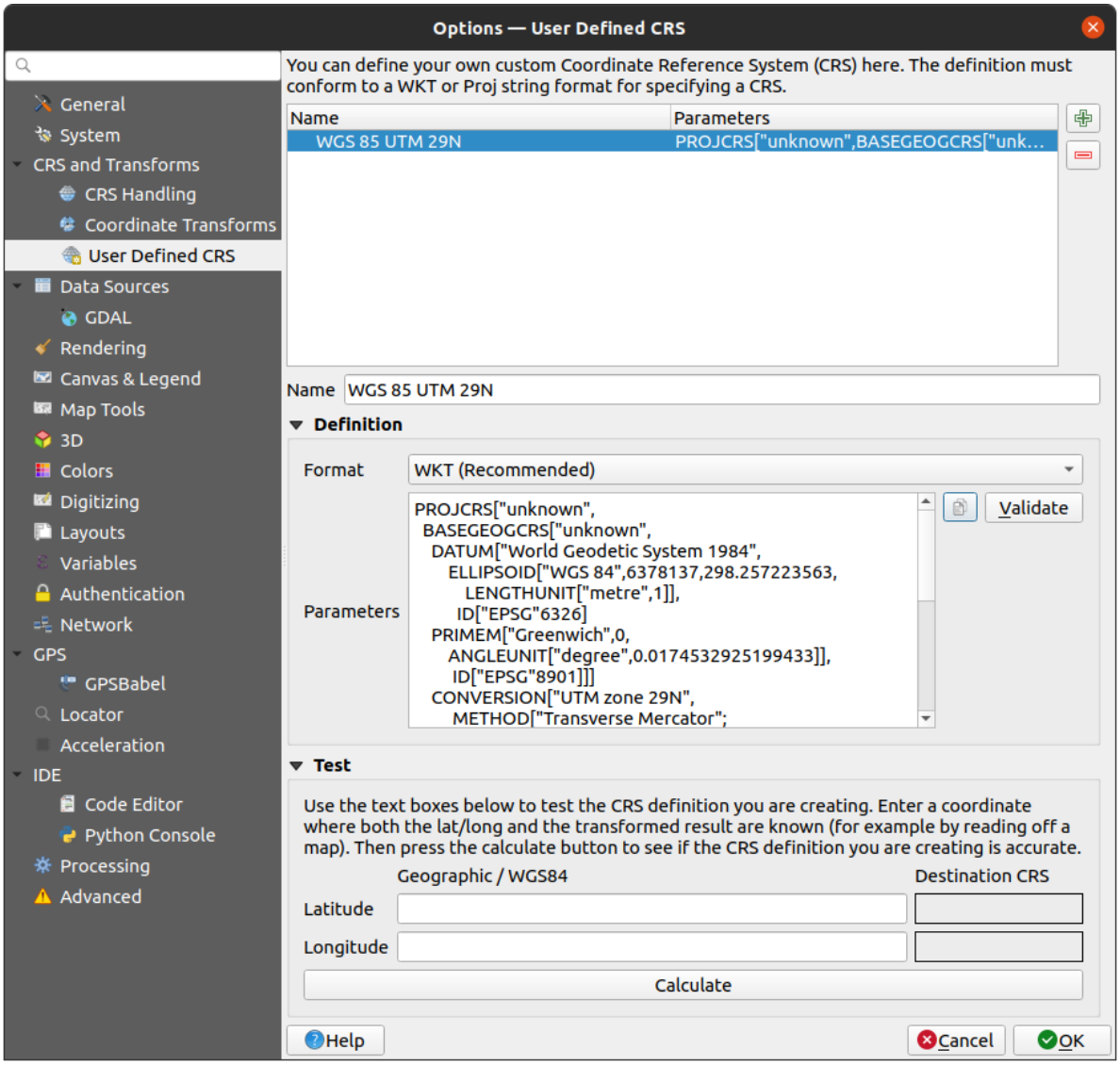


Fig. 10.4: Dialogul CRS-urilor Personalizate

You can test your CRS parameters to see if they give sane results. To do this, enter known WGS 84 latitude and longitude values in *North* and *East* fields, respectively. Click on *Calculate*, and compare the results with the known values in your coordinate reference system.

10.5.1 Integrate an NTv2-transformation in QGIS

To integrate an NTv2 transformation file in QGIS you need one more step:

1. Place the NTv2 file (.gsb) in the CRS/Proj folder that QGIS uses (e.g. `C:\OSGeo4W64\share\proj` for windows users)
2. Add **nadgrids** (`+nadgrids=nameofthefile.gsb`) to the Proj definition in the *Parameters* field of the *Custom Coordinate Reference System Definition* (*Settings* ► *Custom Projections…*).

Fig. 10.5: Setting an NTv2 transformation

10.6 Datum Transformations

In QGIS, «on-the-fly» CRS transformation is enabled by default, meaning that whenever you use layers with different coordinate systems QGIS transparently reprojects them to the project CRS. For some CRS, there are a number of possible transforms available to reproject to the project's CRS!

By default, QGIS will attempt to use the most accurate transformation available. However, in some cases this may not be possible, e.g. whenever additional support files are required to use a transformation. Whenever a more accurate transformation is available, but is not currently usable, QGIS will show an informative warning message advising you of the more accurate transformation and how to enable it on your system. Usually, this requires download of an external package of transformation support files, and extracting these to the `proj` folder under your QGIS *user profile* folder.

If desired, QGIS can also prompt you whenever multiple possible transformations can be made between two CRSs, and allow you to make an informed selection of which is the most appropriate transformation to use for your data.

This customization is done in the *Settings* ► *Options* ► *Transformations* tab menu under the *Default datum transformations* group:

- using *Ask for datum transformation if several are available*: when more than one appropriate datum transformation exist for a source/destination CRS combination, a dialog will automatically be opened prompting users to choose which of these datum transformations to use for the project. If the *Make default* checkbox is ticked when selecting a transformation from this dialog, then the choice is remembered and automatically applied to any newly created QGIS projects.
- or defining a list of appropriate datum transformations to use as defaults when loading a layer to a project or reprojecting a layer.

  Use the button to open the *Select Datum Transformations* dialog. Then:

  1. Choose the *Source CRS* of the layer, using the drop-down menu or the Select CRS widget.
  2. Provide the *Destination CRS* in the same way.
  3. A list of available transformations from source to destination will be shown in the table. Clicking a row shows details on the settings applied and the corresponding accuracy and area of use of the transformation.

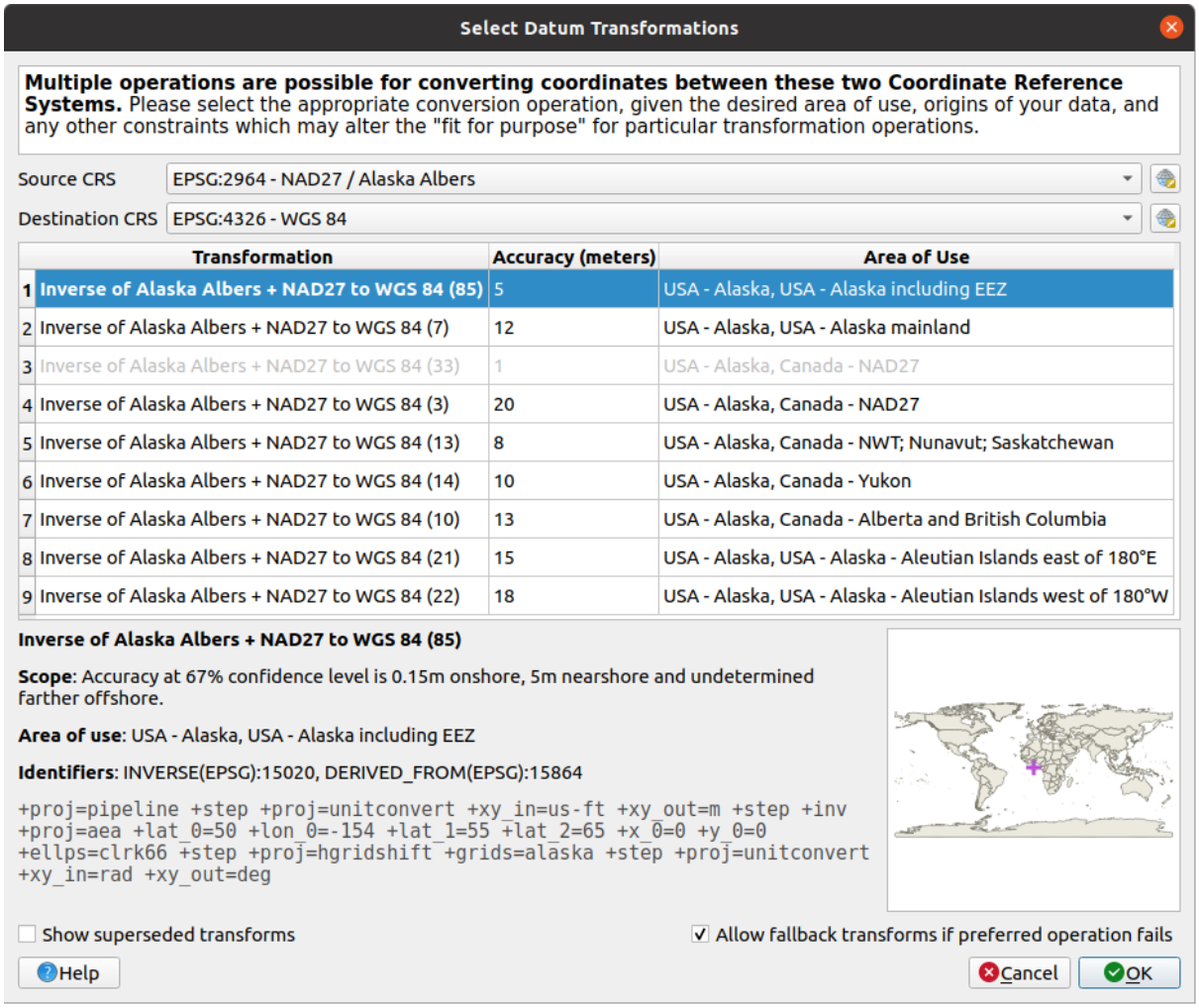


Fig. 10.6: Selecting a preferred default datum transformation

In some cases a transformation may not be available for use on your system. In this case, the transformation will still be shown (greyed) in this list but can not be picked until you install the required package of transformation support. Usually, a button is provided to download and install the corresponding grid, which is then stored under the `proj` folder in the active *user profile* directory.

4. Find your preferred transformation and select it
5. Set whether you *Allow fallback transforms if preferred operation fails*
6. Clic pe *OK*

   A row is added to the table under *Default Datum Transformations* with information about the *Source CRS*, the *Destination CRS*, the *Operation* applied for the transformation and whether *Allow fallback Transforms* is enabled.

From now, QGIS automatically uses the selected datum transformations for further transformation between these two CRSs until you remove it ( ) from the list or change the entry ( ) in the list.

Datum transformations set in the *Settings* ► *Options* ► *Transformations* tab will be inherited by all new QGIS projects created on the system. Additionally, a particular project may have its own specific set of transformations specified via the *CRS* tab of the *Project properties* dialog (*Project* ► *Properties…*). These settings apply to the current project only.

CAPITOLUL 11

Vizualizarea Hărților

Creating maps is one of the „business ends” of QGIS. Map views are canvases to visualize spatial layers loaded in the project, with the various rendering settings applied to them. They can be of *2D* or *3D* type or plotted along a *profile line*, show different scale or extent, or display a different set of the loaded layers thanks to *map themes*.

11.1 2D Map View

The 2D map view (also called **Map canvas**) is the central place where maps are displayed. QGIS opens by default with a single map view (called `main map`), showing layers in 2D, and tightly bound to the *Layers* panel. That window reflects the rendering (symbology, labeling, visibilities…) you applied to the loaded layers.

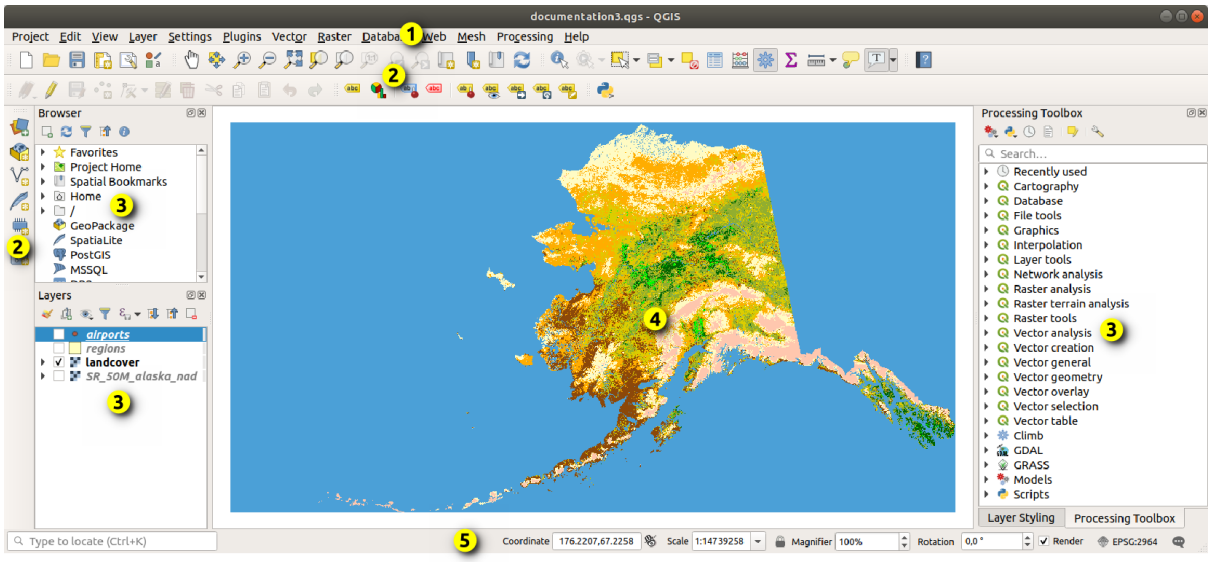


Fig. 11.1: QGIS GUI

11.1.1 Exploring the map view

When you add a layer (see e.g. *Deschiderea datelor*), QGIS automatically looks for its CRS. If a different CRS is set by default for the project (see *Project Coordinate Reference Systems*) then the layer extent is „on-the-fly" translated to that CRS, and the map view is zoomed to that extent if you start with a blank QGIS project. If there are already layers in the project, no map canvas resize is performed, so only features falling within the current map canvas extent will be visible.

Click on the map view and you should be able to interact with it, panning or zooming to different areas of the map. Dedicated tools are provided in the *Navigation Toolbar* and in the *View* menu, with handful shortcuts from the keyboard or the mouse buttons.

Tabelul 11.1: Map canvas navigation tools

Instrument	Folosire
Pan Map	• Single left click: the map is centered on the clicked point, at the same scale • Hold down the left mouse button and drag the map canvas.
Zoom In	• Single left click: the map is centered on the clicked point, while the scale gets doubled • Drag a rectangle on the map canvas with the left mouse button to zoom in to an area. • Hold the `Alt` key to switch to the Zoom Out tool.
Zoom Out	• Single left click: the map is centered on the clicked point, while the scale gets halved • Drag a rectangle on the map canvas with the left mouse button to zoom out from an area. • Hold the `Alt` key to switch to the Zoom In tool.
Pan Map to Selection	Pan the map to the selected features of all the selected layers in the *Layers* panel.
Zoom To Selection	Zoom to the selected features of all the selected layers in the *Layers* panel. *Also available in the layer contextual menu*
Zoom To Layer(s)	Zoom to the extent of all the selected layers in the *Layers* panel. *Also available in the layer contextual menu*
Zoom Full	Zoom to the extent of all the layers in the project or to the *project full extent*.
Zoom Last	Zoom the map to the previous extent in history.
Zoom Next	Zoom the map to the next extent in history.
Zoom to Native Resolution	Zoom the map to a level where one pixel of the active raster layer covers one screen pixel. *Also available in the layer contextual menu*
Mouse wheel	• Pan map: Hold and drag the mouse wheel. • Zoom: Roll the mouse wheel to zoom in or zoom out. With `Ctrl` key pressed while rolling the mouse wheel results in a finer zoom. • Press the back or forward button to browse the map canvas zoom history.

continues on next page

Tabelul 11.1 – continuare din pagina precedentă

Instrument	Folosire
Keyboard	• Pan map: Hold down the `Space` key and move the mouse. Press the arrow keys to pan up, down, left and right. • Zoom in: Press `PgUp` or `Ctrl++` • Zoom out: Press `PgDown` or `Ctrl+-` • Zoom to area: When certain map tools are active (Identify, Measure…), hold down `Shift` and drag a rectangle on the map to zoom to that area. Not compatible with active selection or edit tools.

Right-click over the map and you should be able to *Copy coordinates* of the clicked point in the map CRS, in WGS84 or in a custom CRS. The copied information can then be pasted in an expression, a script, text editor or spreadsheet…

11.1.2 Controlling map rendering

În mod implicit, QGIS face toate straturile vizibile de fiecare dată când canevasul hărții este actualizat. Între evenimentele care declanșează o actualizare de canevas sunt incluse:

- changing the visibility of a layer
- modifying symbology of a visible layer
- adding a layer
- panning or zooming
- resizing the QGIS window

QGIS permite controlarea procesului de randare în mai multe moduri.

- at the *global level*
- per layer, using e.g. the *scale dependent rendering*
- or with dedicated tools in the GUI.

To stop the map drawing, press the `Esc` key. This will halt the refresh of the map canvas and leave the map partially drawn. It may however take a bit of time after pressing `Esc` for the map drawing to halt.

To suspend rendering, click the *Render* checkbox in the bottom-right corner of the status bar. When *Render* is unchecked, QGIS does not redraw the canvas in response to any of the usual triggers mentioned earlier. Examples of when you might want to suspend rendering include:

- adding many layers and symbolizing them prior to drawing
- adding one or more large layers and setting scale dependency before drawing
- adding one or more large layers and zooming to a specific view before drawing
- any combination of the above

Bifarea casetei *Randare* activează randarea și provoacă o reîmprospătare imediată a canevasului hărții.

11.1.3 Time-based control on the map canvas

QGIS can handle temporal control on loaded layers, i.e. modify the map canvas rendering based on a time variation. To achieve this, you need:

1. Layers that have dynamic temporal properties set. QGIS supports temporal control for different data providers, with custom settings. It's mainly about setting the time range in which the layer would display:
   - *raster layers*: controls whether to display or not the layer.
   - *WMTS layers*: controls whether the data should be rendered based on a static time range or following a dynamic temporal range
   - *vector layers*: features are filtered based on time values associated to their attributes
   - *mesh layers*: displays dynamically the active dataset groups values

   When dynamic temporal options are enabled for a layer, an icon is displayed next to the layer in the *Layers* panel to remind you that the layer is temporally controlled. Click the icon to update the temporal settings.
2. Enable the temporal navigation of the map canvas using the *Temporal controller panel*. The panel is activated:
   - using the Temporal controller panel icon located in the *Map Navigation* toolbar
   - or from the *View* ► *Panels* ► *Temporal controller panel* menu

The temporal controller panel

The *Temporal controller* panel has the following modes:

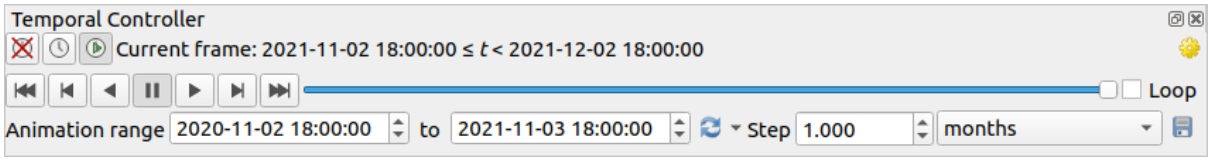


Fig. 11.2: Temporal Controller Panel in navigation mode

- Turn off temporal navigation: all the temporal settings are disabled and visible layers are rendered as usual
- Fixed range temporal navigation: a time range is set and only layers (or features) whose temporal range overlaps with this range are displayed on the map.
- Animated temporal navigation: a time range is set, split into steps, and only layers (or features) whose temporal range overlaps with each frame are displayed on the map
- Settings for general control of the animation
  - *Frames rate*: number of steps that are shown per second
  - *Cumulative range*: all animation frames will have the same start date-time but different end dates and times. This is useful if you wish to accumulate data in your temporal visualization instead of showing a 'moving time window' across your data.

Animating a temporal navigation

An animation is based on a varying set of visible layers at particular times within a time range. To create a temporal animation:

1. Toggle on the Animated temporal navigation, displaying the animation player widget
2. Enter the *Time range* to consider. Using the button, this can be defined as:
   - *Set to full range* of all the time enabled layers
   - *Set to preset project range* as defined in the *project properties*
   - *Set to single layer's range* taken from a time-enabled layer
3. Fill in the time *Step* to split the time range. Different units are supported, from `seconds` to `centuries`. A `source timestamps` option is also available as step: when selected, this causes the temporal navigation to step between all available time ranges from layers in the project. It's useful when a project contains layers with non-contiguous available times, such as a WMS-T service which provides images that are available at irregular dates. This option will allow you to only step between time ranges where the next available image is shown.
4. Click the button to preview the animation. QGIS will generate scenes using the layers rendering at the set times. Layers display depends on whether they overlap any individual time frame.

   The animation can also be previewed by moving the time slider. Checking the *Loop* checkbox will repeatedly run the animation while clicking stops a running animation. A full set of video player buttons is available.

   Horizontal scrolling using the mouse wheel (where supported) with the cursor on the map canvas will also allow you to navigate, or "scrub", the temporal navigation slider backwards and forwards.
5. Click the Export animation button if you want to generate a series of images representing the scene. They can be later combined in a video editor software:

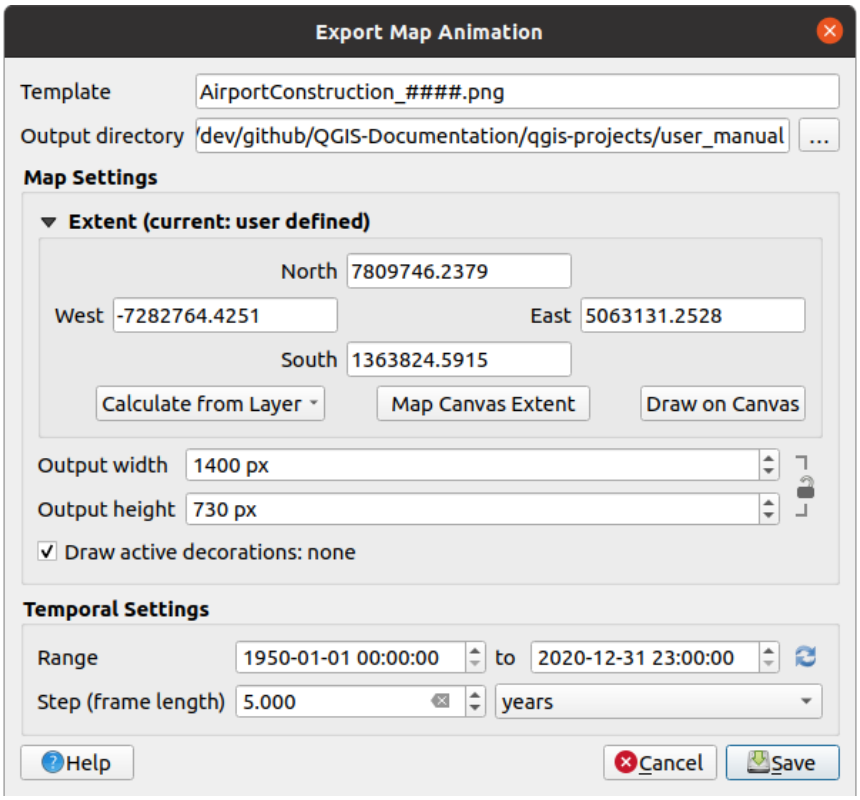


Fig. 11.3: Exporting map canvas animation scenes to images

- The filename *Template*: the `####` are replaced with frame sequence number
- The *Output directory*
- Under *Map settings*, you can:
  - redefine the *spatial extent* to use
  - control the *Resolution* of the image (*Output width* and *Output height*)
  - *Draw active decorations*: whether active *decorations* should be kept in the output
- Under *Temporal settings*, you can redefine:
  - the time *Range* for the animation
  - the *Step (frame length)* in the unit of your choice

11.1.4 Bookmarking extents on the map

Spatial Bookmarks allow you to „bookmark" a geographic location and return to it later. By default, bookmarks are saved in the user's profile (as *User Bookmarks*), meaning that they are available from any project the user opens. They can also be saved for a single project (named *Project Bookmarks*) and stored within the project file, which can be helpful if the project is to be shared with other users.

Crearea unui Semn de Carte

Pentru a crea un semn de carte:

1. Zoom and pan to the area of interest.
2. Select the menu option *View* ► *New Spatial Bookmark…*, press `Ctrl+B` or right-click the *Spatial Bookmarks* entry in the *Browser* panel and select *New Spatial Bookmark*. The *Bookmark Editor* dialog opens.

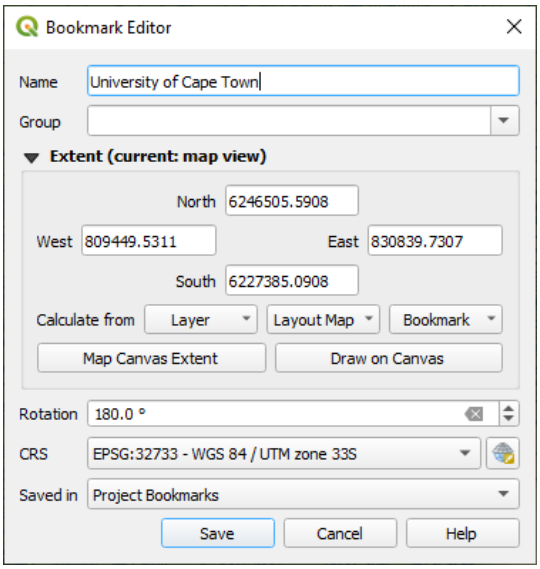


Fig. 11.4: The Bookmark Editor Dialog

3. Enter a descriptive name for the bookmark
4. Enter or select a group name in which to store related bookmarks
5. Select the extent of the area you wish to save, using the *extent selector* widget
6. Change the map *Rotation*

7. Indicate the *CRS* to use for the extent
8. Select whether the bookmark will be *Saved in User Bookmarks* or *Project Bookmarks* (by default, this drop-down list is set to *User Bookmarks*)
9. Press *Save* to add the bookmark to the list

Rețineți că puteți avea mai multe marcaje cu același nume.

Lucrul cu Marcaje

To use and manage bookmarks, you can either use the *Spatial Bookmarks* panel or *Browser*.

Select *View* ► *Show Spatial Bookmark Manager* or press `Ctrl+7` to open the *Spatial Bookmarks Manager* panel.
Select *View* ► *Show Bookmarks* or `Ctrl+Shift+B` to show the *Spatial Bookmarks* entry in the *Browser* panel.

You can perform the following tasks:

Tabelul 11.2: Managing bookmark actions

Task	Spatial Bookmark Manager	Browser
Zoom to a Bookmark	Double-click on it, or select the bookmark and press the *Zoom to bookmark* button.	Double-click on it, drag and drop it to the map canvas, or right-click the bookmark and select *Zoom to Bookmark*.
Delete a bookmark	Select the bookmark and click the *Delete bookmark* button. Confirm your choice.	Right-click the bookmark and select *Delete Spatial Bookmark*. Confirm your choice.
Export bookmarks to XML	Click the *Import/Export Bookmarks* button and select *Export*. All the bookmarks (user or project) are saved in an xml file.	Select one or more folders (user or project) or subfolders (groups), then right-click and select *Export Spatial Bookmarks…*. The selected bookmark subset is saved.
Import bookmarks from XML	Click the *Import/Export Bookmarks* button and select *Import*. All bookmarks in the XML file are imported as user bookmarks.	Right-click the *Spatial Bookmarks* entry or one of its folders (user or project) or subfolders (groups) to determine where to import the bookmarks, then select *Import Spatial Bookmarks*. If performed on the *Spatial Bookmarks* entry, the bookmarks are added to *User Bookmarks*.
Edit bookmark	You can change a bookmark by changing the values in the table. You can edit the name, the group, the extent and if it is stored in the project or not.	Right-click the desired bookmark and select *Edit Spatial Bookmark…*. The *Bookmark Editor* will open, allowing you to redefine every aspect of the bookmark as if you were creating it for the first time. You can also drag and drop the bookmark between folders (user and project) and subfolders (groups).

You can manage bookmark actions by right-click on the desired bookmark in the *Spatial Bookmarks Manager*. You can also zoom to bookmarks by typing the bookmark name in the *locator*.

11.1.5 Decorating the map

Decorations include Grid, Title Label, Copyright Label, Image, North Arrow, Scale Bar and Layout Extents. They are used to «decorate» the map by adding cartographic elements.

Grilă

Grid allows you to add a coordinate grid and coordinate annotations to the map canvas.

1. Select menu option *View* ► *Decorations* ► *Grid…* to open the dialog.

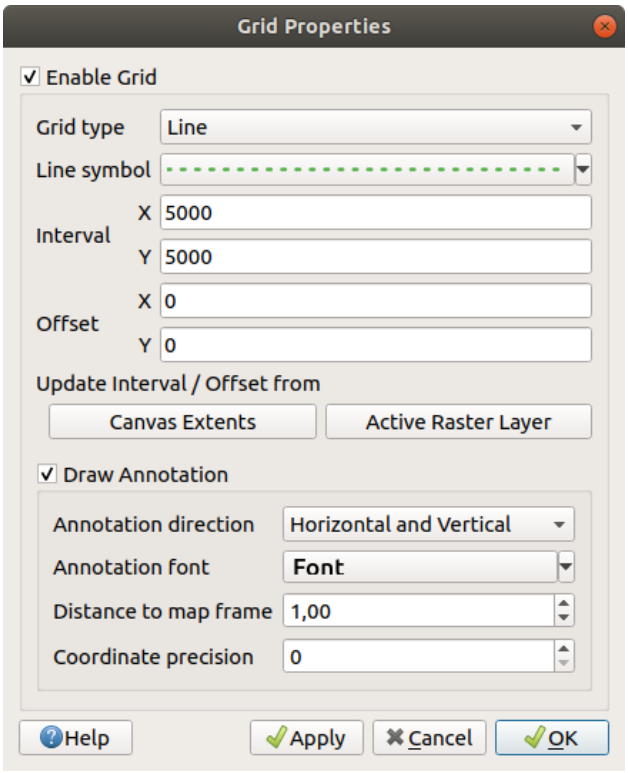


Fig. 11.5: Dialogul Grilei

2. Tick *Enable grid* and set grid definitions according to the layers loaded in the map canvas:
   - The *Grid type*: it can be *Line* or *Marker*
   - The associated *Line symbol* or *marker symbol* used to represent the grid marks
   - The *Interval X* and *Interval Y* between the grid marks, in map units
   - An *Offset X* and *Offset Y* distance of the grid marks from the bottom left corner of the map canvas, in map units
   - The interval and offset parameters can be set based on the:
     - *Canvas Extents*: generates a grid with an interval that is approximatively 1/5 of the canvas width
     - *Active Raster Layer* resolution
3. Tick *Draw annotations* to display the coordinates of the grid marks and set:
   - The *Annotation direction*, ie how the labels would be placed relative to their grid line. It can be:
     - *Horizontal* or *Vertical* for all the labels

  - *Horizontal and Vertical*, ie each label is parallel to the grid mark it refers to
  - *Boundary direction*, ie each label follows the canvas boundary, and is perpendicular to the grid mark it refers to
- The *Annotation font* (text formatting, buffer, shadow…) using the *font selector widget*
- The *Distance to map frame*, margin between annotations and map canvas limits. Convenient when *exporting the map canvas* eg to an image format or PDF, and avoid annotations to be on the „paper" limits.
- The *Coordinate precision*

4. Click *Apply* to verify that it looks as expected or *OK* if you're satisfied.

Title Label

Title Label allows you to decorate your map with a **Title**.

To add a Title Label decoration:

1. Select menu option *View* ► *Decorations* ► *Title Label…* to open the dialog.

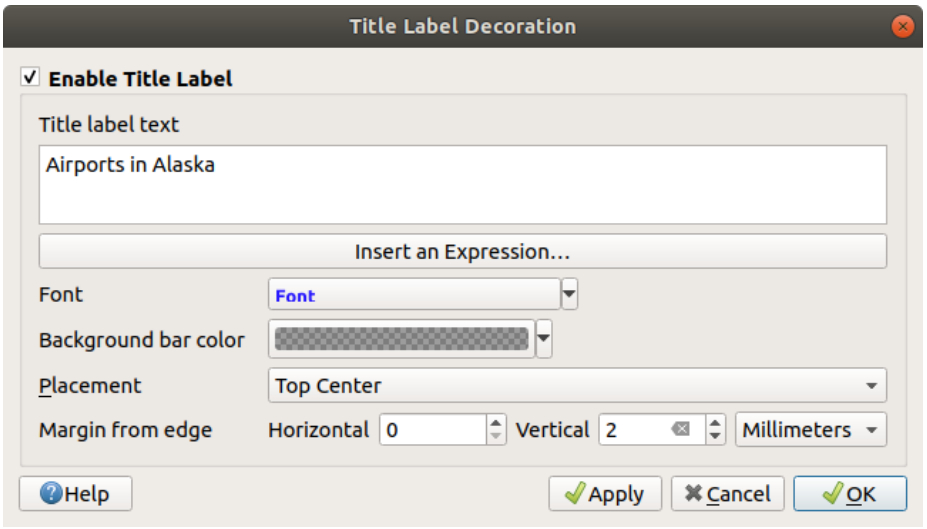


Fig. 11.6: The Title Decoration Dialog

2. Make sure *Enable Title Label* is checked
3. Enter the title text you want to place on the map. You can make it dynamic using the *Insert or Edit an Expression…* button.
4. Choose the *Font* for the label using the *font selector widget* with full access to QGIS *text formatting* options. Quickly set the font color and opacity by clicking the black arrow to the right of the font combo box.
5. Select the *color* to apply to the title's *Background bar color*.
6. Choose the *Placement* of the label in the canvas: options are *Top left*, *Top Center* (default), *Top Right*, *Bottom left*, *Bottom Center* and *Bottom Right*.
7. Refine the placement of the item by setting a horizontal and/or vertical *Margin from Edge*. These values can be in **Millimeters** or **Pixels** or set as a **Percentage** of the width or height of the map canvas.
8. Click *Apply* to verify that it looks as expected or *OK* if you're satisfied.

Copyright Label

Copyright Label can be used to decorate your map with a **Copyright** label.

To add this decoration:

1. Select menu option *View* ► *Decorations* ► *Copyright Label…* to open the dialog.

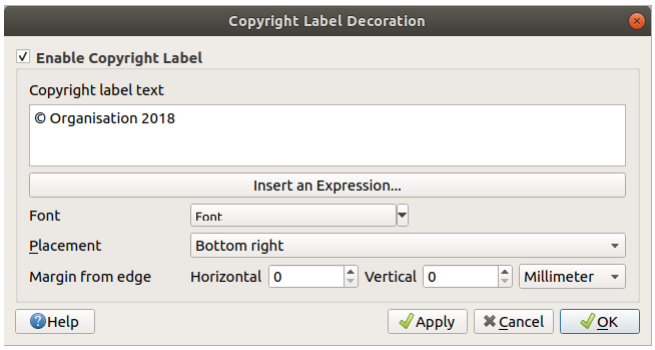


Fig. 11.7: The Copyright Decoration Dialog

2. Make sure *Enable Copyright Label* is checked
3. Enter the copyright text you want to place on the map. You can make it dynamic using the *Insert or Edit an Expression…* button.
4. Choose the *Font* for the label using the *font selector widget* with full access to QGIS *text formatting* options. Quickly set the font color and opacity by clicking the black arrow to the right of the font combo box.
5. Choose the *Placement* of the label in the canvas: options are *Top left*, *Top Center*, *Top Right*, *Bottom left*, *Bottom Center*, and *Bottom Right* (default for Copyright decoration)
6. Refine the placement of the item by setting a horizontal and/or vertical *Margin from Edge*. These values can be in **Millimeters** or **Pixels** or set as a **Percentage** of the width or height of the map canvas.
7. Click *Apply* to verify that it looks as expected or *OK* if you're satisfied.

Image Decoration

Image allows you to add an image (logo, legend, ..) on the map canvas.

To add an image:

1. Select menu option *View* ► *Decorations* ► *Image…* to open the dialog.

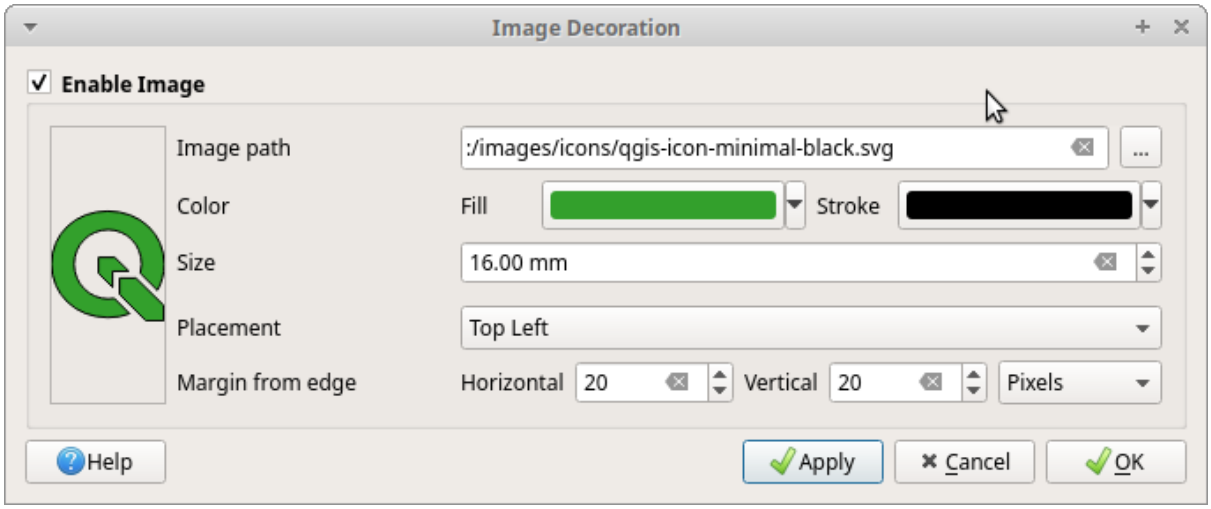


Fig. 11.8: The Image Decoration Dialog

2. Make sure *Enable Image* is checked
3. Select a bitmap (e.g. png or jpg) or SVG image using the … Browse button
4. If you have chosen a parameter enabled SVG then you can also set a *Fill* or *Stroke* (outline) color. For bitmap images, the color settings are disabled.
5. Set a *Size* of the image in mm. The width of selected image is used to resize it to given *Size*.
6. Choose where you want to place the image on the map canvas with the *Placement* combo box. The default position is *Top Left*.
7. Set the *Horizontal* and *Vertical Margin from (Canvas) Edge*. These values can be set in **Millimeters**, **Pixels** or as a **Percentage** of the width or height of the map canvas.
8. Click *Apply* to verify that it looks as expected and *OK* if you're satisfied.

Săgeata Nordului

North Arrow allows you to add a north arrow on the map canvas.

To add a north arrow:

1. Select menu option *View* ► *Decorations* ► *North Arrow*… to open the dialog.

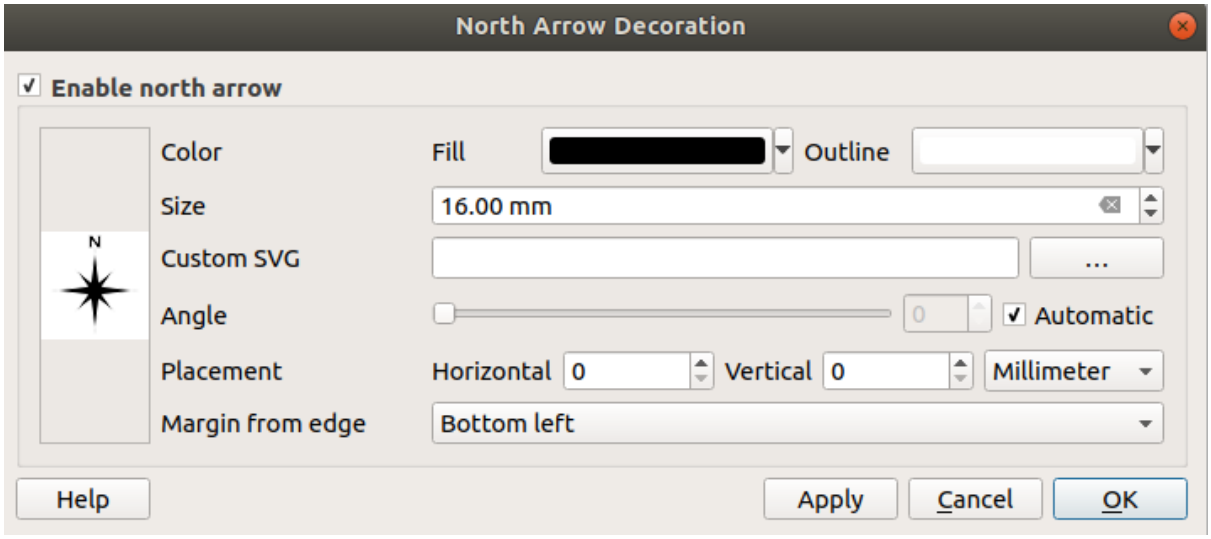


Fig. 11.9: Dialogul Săgeții Nordului

2. Make sure *Enable north arrow* is checked
3. Optionally change the color and size, or choose a custom SVG
4. Optionally change the angle or choose **Automatic** to let QGIS determine the direction
5. Optionally choose the placement from the Placement combo box
6. Optionally refine the placement of the arrow by setting a horizontal and/or vertical *Margin from (Canvas) Edge*. These values can be in **Millimeters** or **Pixels** or set as a **Percentage** of the width or height of the map canvas.
7. Click *Apply* to verify that it looks as expected and *OK* if you're satisfied.

Scara Grafică

Scale Bar adds a simple scale bar to the map canvas. You can control the style and placement, as well as the labelling of the bar. The scale bar respects the active *project's distance unit* as defined in *Project properties* ► *General* ► *Units for distance measurement*.

Pentru a adăuga o scară grafică:

1. Select menu option *View* ► *Decorations* ► *Scale Bar*... to open the dialog

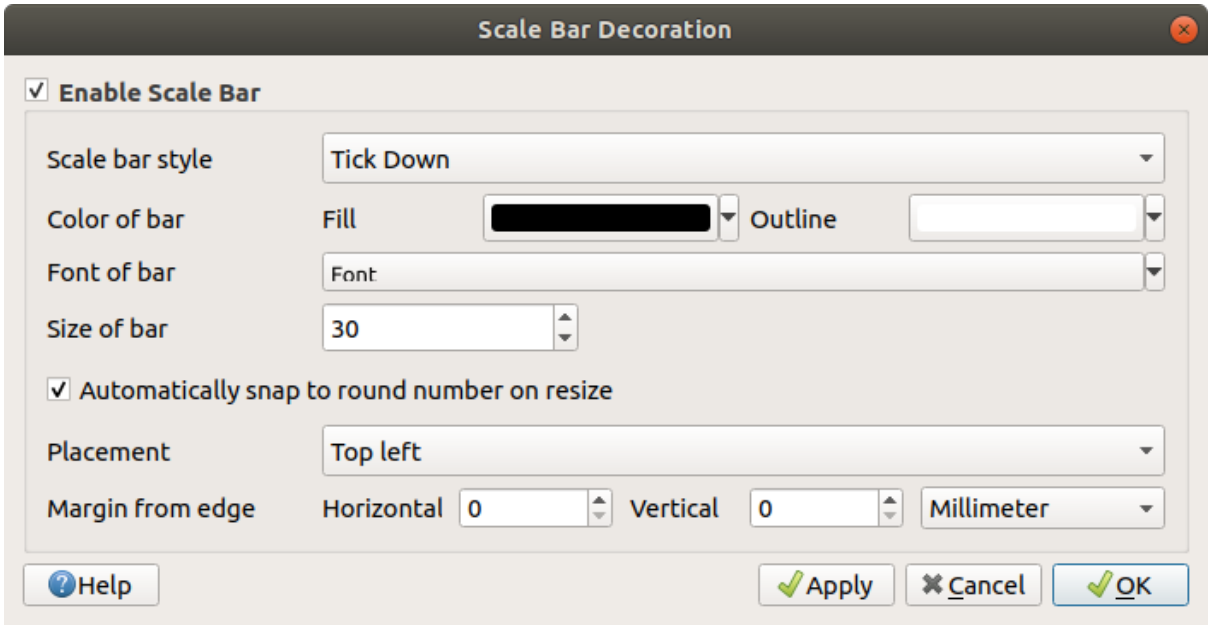


Fig. 11.10: Dialogul Scării Grafice

2. Make sure *Enable scale bar* is checked
3. Choose a style from the *Scale bar style* combo box
4. Select the *Color of bar* by choosing a fill color (default: black) and an outline color (default: white). The scale bar fill and outline can be made opaque by clicking on the down arrow to the right of the color input.
5. Select the font for the scale bar from the *Font of bar* combo box
6. Set the *Size of bar* in the active unit
7. Optionally check *Automatically snap to round number on resize* to display easy-to-read values
8. Choose the placement from the *Placement* combo box
9. You can refine the placement of the item by setting a horizontal and/or vertical *Margin from (Canvas) Edge*. These values can be in **Millimeters** or **Pixels** or set as a **Percentage** of the width or height of the map canvas.
10. Click *Apply* to verify that it looks as expected or *OK* if you're satisfied.

Layout Extents

Layout Extents adds the extents of *map item(s)* in print layout(s) to the canvas. When enabled, the extents of all map items within all print layouts are shown using a lightly dotted border labeled with the name of the print layout and map item. You can control the style and labeling of the displayed layout extents. This decoration is useful when you are tweaking the positioning of map elements such as labels, and need to know the actual visible region of print layouts.

Fig. 11.11: Example of layout extents displayed in a QGIS project with two print layouts. The print layout named «Sights» contains two map items, while the other print layout contains one map item.

To add layout extent(s):

1. Select *View* ► *Decorations* ► *Layout Extents* to open the dialog

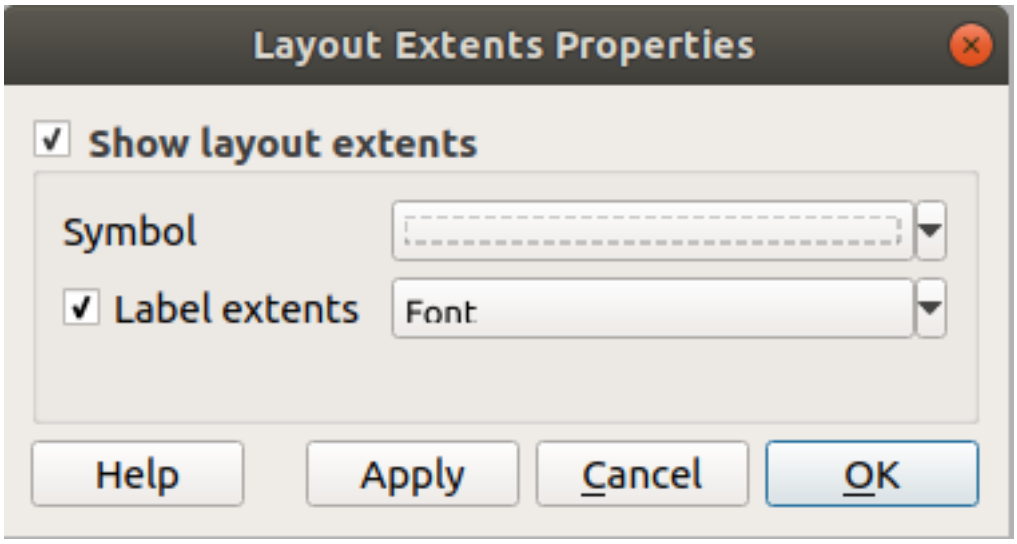


Fig. 11.12: The Layout Extents Dialog

2. Make sure *Show layout extents* is checked.

3. Optionally change the symbol and labeling of the extents.
4. Click *Apply* to verify that it looks as expected and *OK* if you're satisfied.

Sfat: Decorations Settings

When you save a QGIS project file, any changes you have made to Grid, North Arrow, Scale Bar, Copyright and Layout Extents will be saved in the project and restored the next time you load the project.

11.1.6 Instrumente de Adnotare

Annotations are another type of elements added onto the map canvas to provide additional information that can not be depicted by the rendered layers. Unlike *labels* that rely on attribute values stored in vector layers, annotations are independent details, stored within the project file itself.

Two families of annotations are available in QGIS:

- **Feature annotations**: they are actual georeferenced features of text, marker, line or polygon type stored within a special layer type called „annotation layer". They are tied to a particular geographic location, meaning that moving your map, changing the scale or changing projection won't cause your annotations to jump around the map. Rather, they'll be locked in place to the location you've drawn them.
- **Balloon annotations**: these are individuals annotations of text, form or image type placed inside a bubble. They can be associated to any layer for their visibility, are drawn on top of the map canvas. The size is dependent from the map canvas scale, and its position can be anchored.

Sfat: Layout the map with annotations

You can print or export annotations with your map to various formats using:

- map canvas export tools available in the *Project* menu
- *print layout*, in which case you need to check *Draw map canvas items* in the corresponding map item properties

---

The *Annotations Toolbar* provides a set of tools to create and interact with both families of annotations.

Tabelul 11.3: The Annotations Toolbar actions

Instrument	Folosire	Scope
New Annotation Layer	Create a new layer to store annotations	Feature annotations
Main Annotation Layer Properties	Control settings of the Main Annotation Layer	
Modify Annotations	Select, move, resize and modify symbology properties of annotations	
Create Polygon Annotation	Create an annotation as a polygon feature	
Create Line Annotation	Create an annotation as a polyline feature	
Create Marker Annotation	Create an annotation as a point feature	
Create Text Annotation at Point	Create an annotation as a text label	
Create Text Annotation along Line	Create an annotation as a curved text along a linestring	
Text Annotation	Select and create a text formatted annotation	Balloon annotations
HTML Annotation	Select and create annotation with an `HTML` file's content	
SVG Annotation	Select and create annotation showing an `SVG` file	
Form Annotation	Select and create annotation showing attributes of a vector layer in a custom form file	

Feature Annotations

Feature annotations are stored in **annotation layers**. Unlike conventional layers, an annotation layer is available only in the current project and can contain features of different types (text, marker, line, polygon). The layer has no attributes and no symbology associated, but instead each feature can be symbolized on an item-by-item basis, through *Layer Styling* panel.

Two types of annotation layer are available in QGIS:

- A common *Annotation Layer*: you can create one using the New Annotation Layer tool. It is listed in the *Layers* panel, allowing you to control its features» visibility, move it to show above or below particular layers in your map, like any common layer. Double-click the layer and you can access its properties.
- The *Main Annotation Layer*: By default, this is where annotations are stored when no annotation layer is available in the project or is selected at creation time. This layer is always drawn on the very top of your map and you won't see it listed in the *Layers* panel alongside the other layers in your project, meaning that its features are always visible. The *Main Annotation Layer Properties* entry on the *Annotations* toolbar helps you open its properties dialog.

Layer Properties

The properties dialog of an annotation layer provides the following tabs:

- *Information*: a read-only dialog representing an interesting place to quickly grab summarized information and metadata on the current layer. This may include the layer extent, count of items per annotation type and total count, CRS details, …
- *Source*: defines general settings for the annotation layer. You can:
  - Set a *Layer name* that will be used to identify the layer in the project (in the *Layers Panel*, with expressions, …)

  - Display the layer's *Assigned Coordinate Reference System (CRS)*: you can change the layer's CRS, selecting a recently used one in the drop-down list or clicking on Select CRS button (see *Coordinate Reference System Selector*). Use this process only if the CRS applied to the layer is a wrong one or if none was applied.

- *Rendering*:
  - You can set the *Maximum (inclusive)* and *Minimum (exclusive)* scale, defining a range of scale in which features will be visible. Out of this range, they are hidden. The Set to current canvas scale button helps you use the current map canvas scale as boundary of the range visibility. See *Visibility Scale Selector* for more information.
  - *Opacity*: You can make the underlying layer in the map canvas visible with this tool. Use the slider to adapt the visibility of your vector layer to your needs. You can also make a precise definition of the percentage of visibility in the menu beside the slider.
  - *Blending mode* at the *Layer* level: You can achieve special rendering effects with these tools that you may previously only know from graphics programs. The pixels of your overlaying and underlying layers are mixed through the settings described in *Modurile de Fuziune*.
  - Apply *paint effects* on all the layer features with the *Draw Effects* button.

  Some of these options are accessible from the feature annotation *Symbology* properties.

Interacting with features

The Feature annotations have dedicated tools for creation depending on their type:

- Create Polygon Annotation
- Create Line Annotation
- Create Marker Annotation
- Create Text Annotation at Point
- Create Text Annotation along Line

All the usual QGIS shortcuts for creating features apply when creating annotation items. A line or polygon annotation is drawn by left-clicking once for each vertex, with a final right mouse click to complete the shape. Snapping can be enabled while you draw, you can use the *Advanced Digitizing Tools* to precisely place vertices, and even switch the *drawing tools* to the streaming mode for completely free-form shapes.

Unlike common layers, an annotation layer does not need to be active before you select its features. Simply grab the Modify Annotations tool and you will be able to interact with any feature annotation:

- **Selection**: left-click on the annotation. By default, annotations are rendered in the order of creation, meaning that recent annotations will be placed on top of older. You may need to play with the Z-index property of features in order to select ones they would sit above.
- **Moving**: Left click on a selected annotation item to start moving it. A right-click or pressing `Esc` key cancels the move, while a second left click will confirm the move. The displacement can also be controlled pressing the cursor keys:
  - `Shift+key` pentru o deplasare mai amplă
  - `Alt+key` pentru o deplasare de `1 px`
- **Geometry modification**: for line or polygon annotations, left-click on a vertex of the geometry, move and click again. Double-click a segment to add a new vertex.
- **Delete**: Pressing the `Del` or `Backspace` key while an annotation is selected will delete that annotation

- *Change feature symbology*

Feature symbology

A selected annotation will display its *Symbology* properties in the *Layer styling* panel. You can:

- Modify the appearance using full capabilities of:
  - the *symbol properties* for polygon, polyline and marker annotations
  - the *text format properties* for text-based annotations. A text area allows you to construct the string to display using QGIS expression functions. It is also possible to set the *Alignment* for text annotation at point (left, center or right of the text point).
- For text annotation at point, also configure whether it should *Ignore map rotation* or *Rotate with map*. In both cases, a custom *Angle* can be set for the feature orientation.
- For text annotation along a line, configure an *Offset from line* in the unit of your choice
- Configure a *Reference scale*: indicates the map scale at which symbol or text sizes which uses paper-based units (such as millimeters or points) relate to. The sizes will be scaled accordingly whenever the map is viewed at a different scale. For instance, a line feature wide of 2mm at 1:2000 *Reference scale* will be rendered using 4mm when the map is viewed at 1:1000.
- Set a *Z-index*: a feature with a higher index is placed on top of features with lower index. A convenient setting for both feature display and selection.
- Modify some of the *Layer rendering* settings

Balloon annotations

You can add balloon annotations through *Edit* ► *Add Annotation* ► menu or from the *Annotations Toolbar*:

- Text Annotation for custom formatted text
- HTML Annotation to place the content of an `html` file
- SVG Annotation to add an `SVG` symbol
- Form Annotation: useful to display attributes of a vector layer in a customized `ui` file (see Fig. 11.13). This is similar to the *custom attribute forms*, but displayed in an annotation item. Also see this video https://www.youtube.com/watch?v=0pDBuSbQ02o&feature=youtu.be&t=2m25s from Tim Sutton for more information.

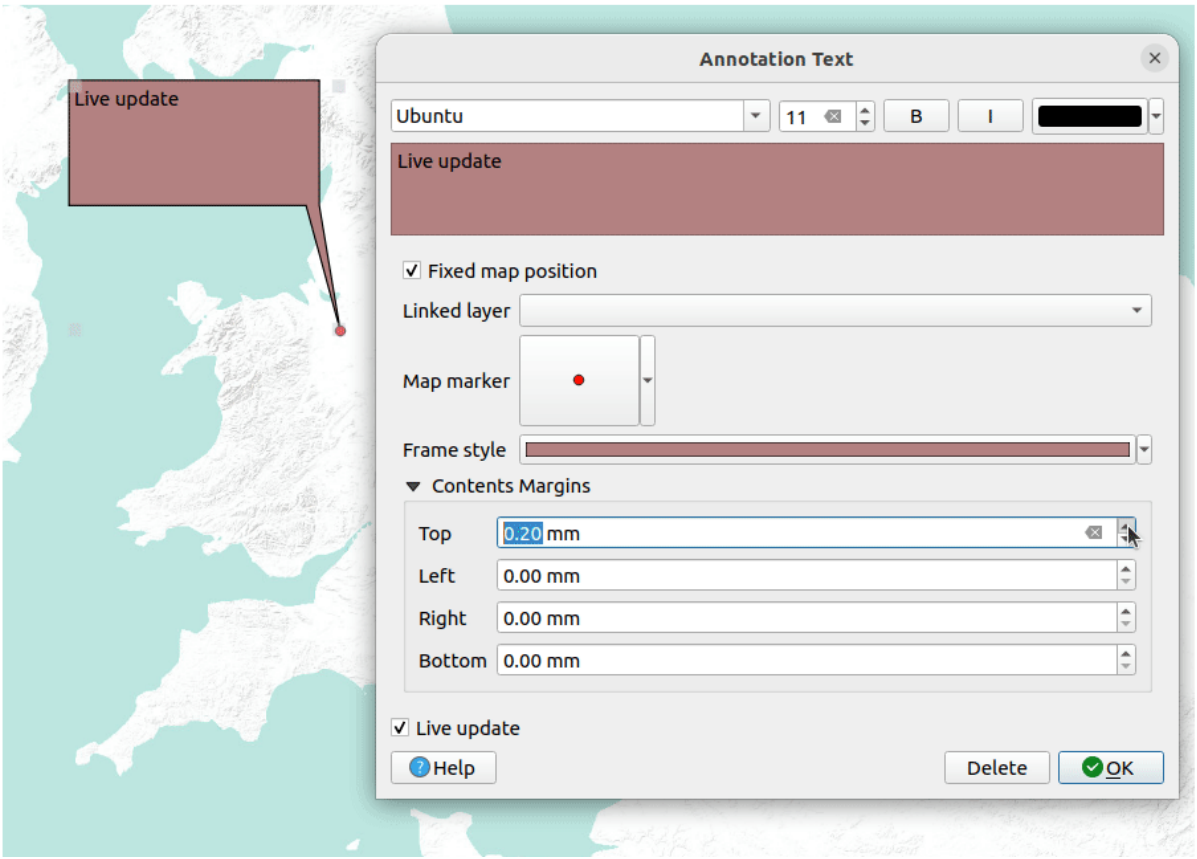


Fig. 11.13: Examples of balloon annotations

To add a balloon annotation, select the corresponding tool and click on the map canvas. An empty balloon is added. Double-click on it and a dialog opens with various options. This dialog is almost the same for all the annotation types:

- At the top, a file selector to fill with the path to an `html`, `svg` or `ui` file depending on the type of annotation. For text annotation, you can enter your message in a text box and set its rendering with the normal font tools.
- *Fixed map position*: when unchecked, the balloon placement is based on a screen position (instead of the map), meaning that it's always shown regardless the map canvas extent.
- *Linked layer*: associates the annotation with a map layer, making it visible only when that layer is visible.
- *Map marker*: using *QGIS symbols*, sets the symbol to display at the balloon anchor position (shown only when *Fixed map position* is checked).
- *Frame style*: sets the frame background color, transparency, stroke color or width of the balloon using QGIS symbols.
- *Contents margins*: sets interior margins of the annotation frame.
- *Live update* allows you to live preview your changes.

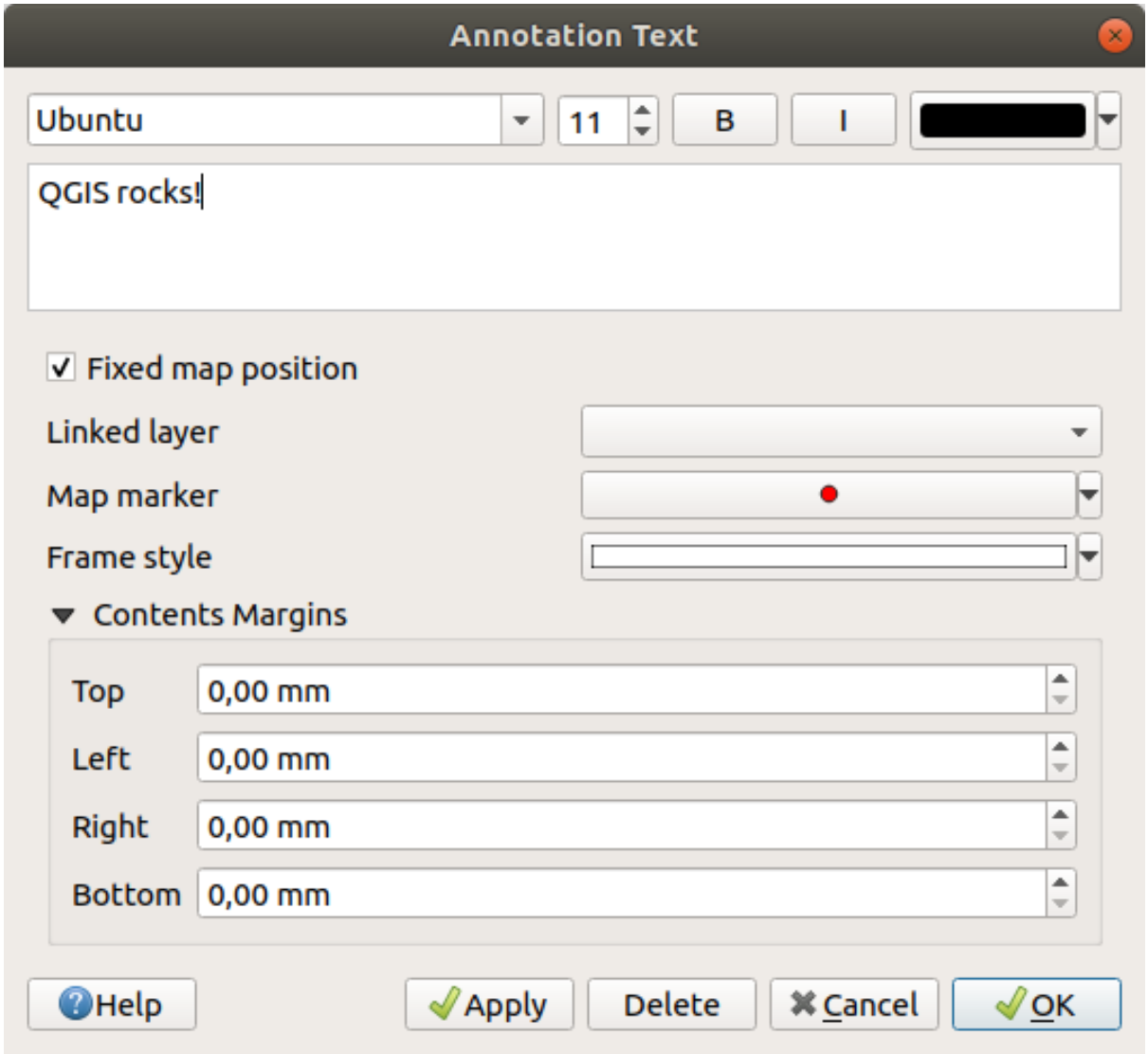


Fig. 11.14: A ballon annotation text dialog

When a balloon annotation tool is active, you can also:

- Select an annotation
- Resize an annotation
- Move an annotation by map position (by dragging the map marker) or by moving only the balloon.
- Delete an annotation: select it and either press the `Del` or `Backspace` button, or double-click it and press the *Delete* button in the properties dialog.
- Right-click and in the contextual menu:
  - *Copy coordinate* of the annotation's map marker in various CRS
  - *Edit* properties of the annotation. Same as double-clicking the annotation.
  - *Delete* the annotation

11.1.7 Măsurarea

Informații generale

QGIS provides four means of measuring geometries:

- interactive measurement tools
- measuring in the Field Calculator
- derived measurements in the *Identifying Features* tool
- the vector analysis tool: *Vector* ► *Geometry Tools* ► *Export/Add Geometry Columns*

Measuring works within projected coordinate systems (e.g., UTM) and unprojected data. The first three measuring tools behave equally to global project settings:

- Unlike most other GIS, the default measurement metric is ellipsoidal, using the ellipsoid defined in *Project* ► *Properties…* ► *General*. This is true both when geographic and projected coordinate systems are defined for the project.
- If you want to calculate the projected/planimetric area or distance using cartesian maths, the measurement ellipsoid has to be set to „None/Planimetric" (*Project* ► *Properties…* ► *General*). However, with a geographic (ie unprojected) CRS defined for the data and project, area and distance measurement will be ellipsoidal.

However, neither the identify tool nor the field calculator will transform your data to the project CRS before measuring. If you want to achieve this, you have to use the vector analysis tool: *Vector* ► *Geometry Tools* ► *Add Geometry Attributes…*. Here, measurement is planimetric, unless you choose the ellipsoidal measurement.

Measure length, areas, bearings and angles interactively

Click the icon in the Attribute toolbar to begin measurements. The down arrow near the icon switches between length, area, bearing or angle. The default unit used in the dialog is the one set in *Project* ► *Properties…* ► *General* menu.

For the *Measure Line* and the *Measure Area* the measurements can be done in *Cartesian* or *Ellipsoidal* measure.

Notă: Configuring the measure tool

While measuring length or area, clicking the *Configuration* button at the bottom of the widget opens the *Settings* ► *Options* ► *Map Tools* menu, where you can select the rubberband color, the precision of the measurements and the unit behavior. You can also choose your preferred measurement or angle units, but keep in mind that those values are overridden in the current project by the selection made in the *Project* ► *Properties…* ► *General* menu, and by the selection made in the measurement widget.

All measuring modules use the snapping settings from the digitizing module (see section *Setting the snapping tolerance and search radius*). So, if you want to measure exactly along a line feature, or around a polygon feature, first set its layer snapping tolerance. Now, when using the measuring tools, each mouse click (within the tolerance setting) will snap to that layer.

The Measure Line measures distances between given points. The tool then allows you to click points on the map. Each segment length, as well as the total, shows up in the measure window. In the measure window, you will see coordinates for all your points and distances. Keep in mind that the first row will contain only coordinates, as it represents your starting point. Now it is possible to copy all your line measurements at once to the clipboard using the *Copy* button. Clicking the *Configuration* button you will access to *Measure Tool Copy Settings* where you can set up *copy options*. To stop measuring, click the right mouse button.

Note that you can use the drop-down list near the total to change the *measurement units* interactively while working with the measure tool. This unit is retained for the widget until a new project is created or another project is opened.

The *Info* section in the dialog explains how calculations are made according to the CRS settings available.

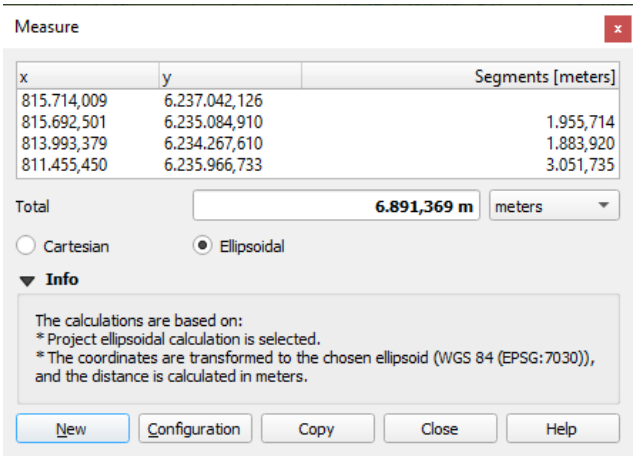


Fig. 11.15: Măsurarea Distanței

Measure Area: Areas can also be measured. In the measure window, the accumulated area size appears. Right-click to stop drawing. The Info section is also available as well as the ability to switch between different *area units*.

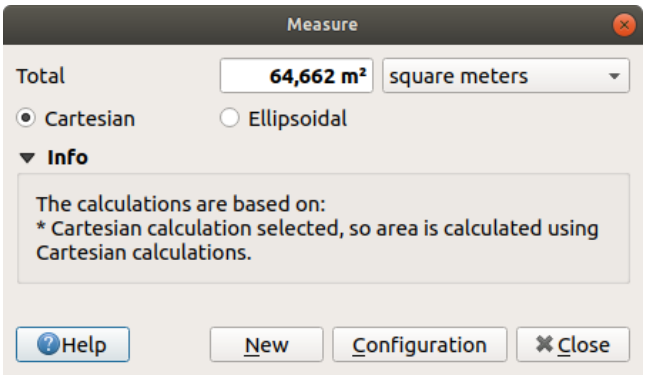


Fig. 11.16: Măsurarea Ariei

Measure Bearing: You can also measure bearings. The cursor becomes cross-shaped. Click to draw the first point of the bearing, then move the cursor to draw the second point. The measurement is displayed in a pop-up dialog.

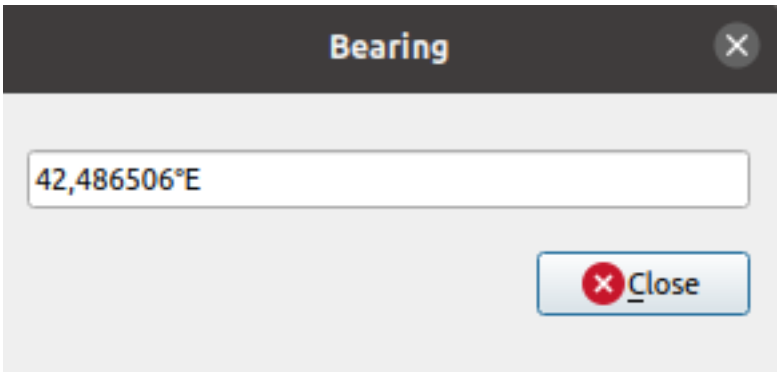


Fig. 11.17: Measure Bearing

Measure Angle: You can also measure angles. The cursor becomes cross-shaped. Click to draw the first segment of the angle you wish to measure, then move the cursor to draw the desired angle. The measurement is displayed in a pop-up dialog.

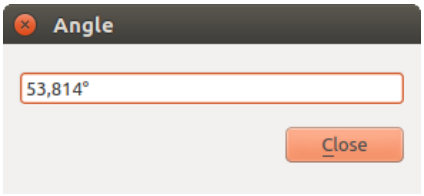


Fig. 11.18: Măsurarea Unghiului

11.1.8 Setting additional map views

It is also possible to open additional map views whose content could diverge from the *Layers* panel current state. To add a new map view, go to *View* ► *New Map View*. A new floating widget, mimicking the main map view's rendering, is added to QGIS. You can add as many map views as you need. They can be kept floating, placed side by side or stacked on top of each other.

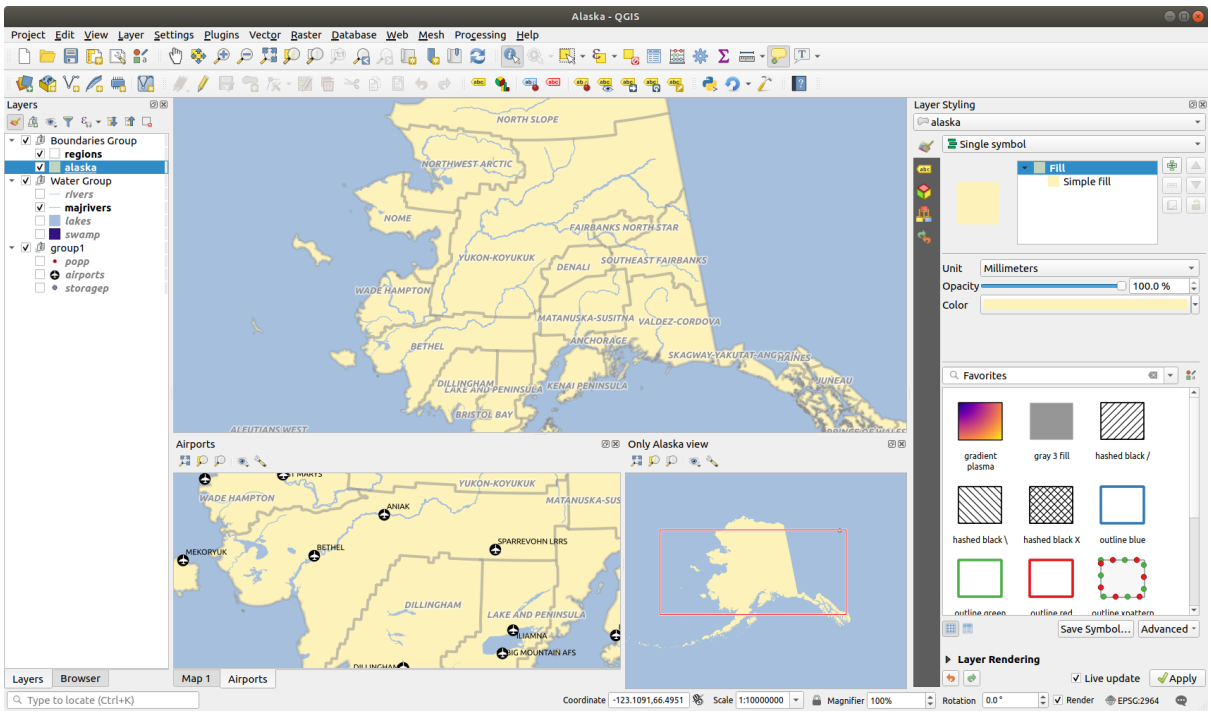


Fig. 11.19: Multiple map views with different settings

At the top of an additional map canvas, there's a toolbar with the following capabilities:

- Zoom Full, Zoom to Selection and Zoom to Layer(s) to navigate within the view
- Set View Theme to select the *map theme* to display in the map view. If set to `(none)`, the view will follow the *Layers* panel changes.
- View settings to configure the map view:
  - *Synchronize view center with main map*: syncs the center of the map views without changing the scale. This allows you to have an overview style or magnified map which follows the main canvas center.

- *Synchronize view to selection*: same as zoom to selection
- *Scale*
- *Rotation*
- *Magnification*
- *Synchronize scale* with the main map scale. A *Scale factor* can then be applied, allowing you to have a view which is e.g. always 2x the scale of the main canvas.
- *Show annotations*
- *Show cursor position*
- *Show main canvas extent*
- *Show labels*: allows to hide labels regardless they are set in the displayed layers» properties
- *Change map CRS…*
- *Rename view…*

11.1.9 Exportarea vizualizării hărții

Maps you make can be layout and exported to various formats using the advanced capabilities of the *print layout or report*. It's also possible to directly export the current rendering, without a layout. This quick „screenshot" of the map view has some convenient features.

To export the map canvas with the current rendering:

1. Go to *Project* ► *Import/Export*
2. Depending on your output format, select either
   - *Export Map to Image…*
   - or *Export Map to PDF…*

The two tools provide you with a common set of options. In the dialog that opens:

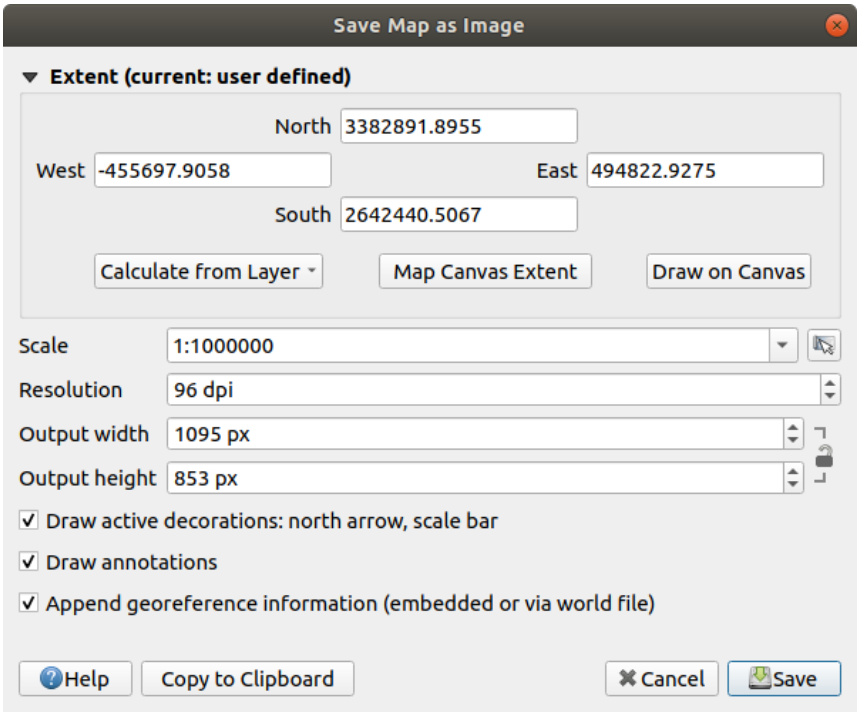


Fig. 11.20: The Save Map as Image dialog

1. Choose the *Extent* to export: it can be the current view extent (the default), the extent of a layer or a custom extent drawn over the map canvas. Coordinates of the selected area are displayed and manually editable.
2. Enter the *Scale* of the map or select it from the *predefined scales*: changing the scale will resize the extent to export (from the center).
3. Set the *Resolution* of the output
4. Control the *Output width* and *Output height* in pixels of the image: based by default on the current resolution and extent, they can be customized and will resize the map extent (from the center). The size ratio can be locked, which may be particularly convenient when drawing the extent on the canvas.
5. *Draw active decorations*: in use *decorations* (scale bar, title, grid, north arrow…) are exported with the map
6. *Draw annotations* to export any *annotation*
7. *Append georeference information (embedded or via world file)*: depending on the output format, a world file of the same name (with extension `PNGW` for `PNG` images, `JPGW` for `JPG`, …) is saved in the same folder as your image. The `PDF` format embeds the information in the PDF file.
8. When exporting to PDF, more options are available in the *Save map as PDF…* dialog:

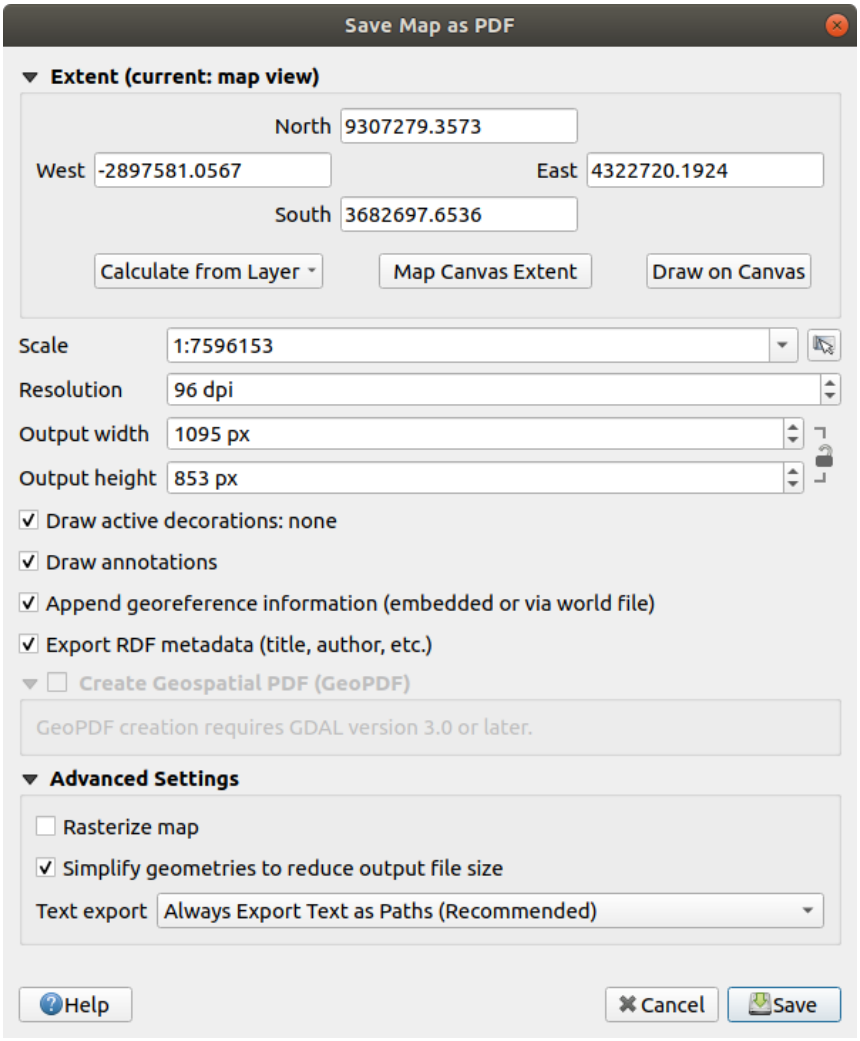


Fig. 11.21: The Save Map as PDF dialog

- *Export RDF metadata* of the document such as the title, author, date, description…
- *Create Geospatial PDF (GeoPDF)*: Generate a georeferenced PDF file. You can:
  - Choose the GeoPDF *Format*
  - *Include vector feature information* in the GeoPDF file: will include all the geometry and attribute information from features visible within the map in the output GeoPDF file.

Notă: A GeoPDF file can also be used as a data source. For more on GeoPDF support in QGIS, see https://north-road.com/2019/09/03/qgis-3-10-loves-geopdf/.

- *Rasterize map*
- *Simplify geometries to reduce output file size*: Geometries will be simplified while exporting the map by removing vertices that are not discernibly different at the export resolution (e.g. if the export resolution is `300 dpi`, vertices that are less than `1/600 inch` apart will be removed). This can reduce the size and complexity of the export file (very large files can fail to load in other applications).
- Set the *Text export*: controls whether text labels are exported as proper text objects (*Always export texts as text objects*) or as paths only (*Always export texts as paths*). If they are exported as text objects then they can be edited in external applications (e.g. Inkscape) as normal text. BUT the side effect is that

the rendering quality is decreased, AND there are issues with rendering when certain text settings like buffers are in place. That's why exporting as paths is recommended.

9. Click *Save* to select file location, name and format.

   When exporting to image, it's also possible to *Copy to clipboard* the expected result of the above settings and paste the map in another application such as LibreOffice, GIMP…

11.2 3D Map View

3D visualization support is offered through the 3D map view. You can create, manage and open 3D map views via *View* ► *3D Map Views* ► menu:

1. By clicking on *New 3D Map View* you can create a new 3D map view. A floating and dockable QGIS panel will appear (see *The 3D Map View dialog*). It has the same extent and view as the 2D main map canvas and provides a set of navigation tools to turn the view into 3D.
2. By clicking on *Manage 3D Map Views* you get in the 3D Map Views Manager. Here you get the ability to open, duplicate, remove and rename 3D map views.
3. If you created one or more 3D map views, you see them listed in *3D Map Views*. You can turn them on and off by clicking on. They will be saved by saving the project, even if they are turned off.

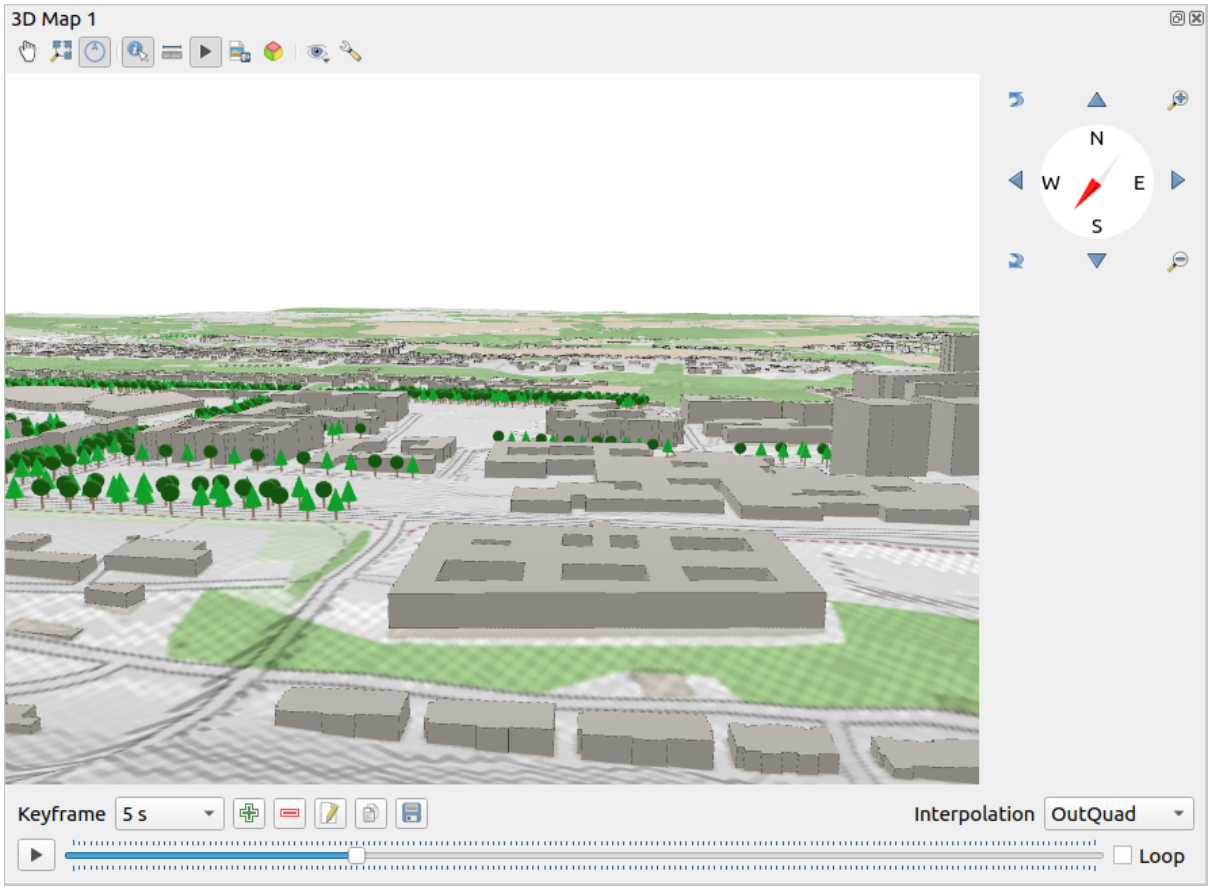


Fig. 11.22: The 3D Map View dialog

The following tools are provided at the top of the 3D map view panel:

- Camera Control: moves the view, keeping the same angle and direction of the camera

- Zoom Full: resizes the view to the whole layers» extent, or the *reference extent* if set
- Toggle On-Screen Notification: shows/hides the navigation widget (that is meant to ease controlling of the map view)
- Identify: returns information on the clicked point of the terrain or the clicked 3D feature(s) – More details at *Identifying Features*
- Measurement Line: measures the horizontal distance between points
- Animations: shows/hides the *animation player* widget
- Save as Image…: exports the current view to an image file format
- Export 3D Scene: exports the current view as a 3D scene (`.obj` file), allowing post-processing in applications like Blender… The terrain and vector features are exported as 3D objects. The export settings, overriding the layers *properties* or map view *configuration*, include:
  - *Scene name* and destination *Folder*
  - *Terrain resolution*
  - *Terrain texture resolution*
  - *Model scale*
  - *Smooth edges*
  - *Export normals*
  - *Export textures*
- Set View Theme: Allows you to select the set of layers to display in the map view from predefined *map themes*.
- The Options menu provides shortcuts to:
  - Add visual effects to the 3D rendering, such as showing *shadows*, *eye dome lighting* or *ambient occlusion*
  - Synchronize the views (*2D map view follows 3D camera* and/or *3D camera follows 2D Map view*)
  - *Show visible camera area in 2D map view*
  - Configure the 3D map view *settings*.
- Dock 3D Map View: switch from docked widget to top level window

11.2.1 Scene Configuration

The 3D map view opens with some default settings you can customize. To do so, expand the Options menu at the top of the 3D canvas panel and press the *Configure* button to open the *3D configuration* window.

In the 3D Configuration window there are various options to fine-tune the 3D scene:

General

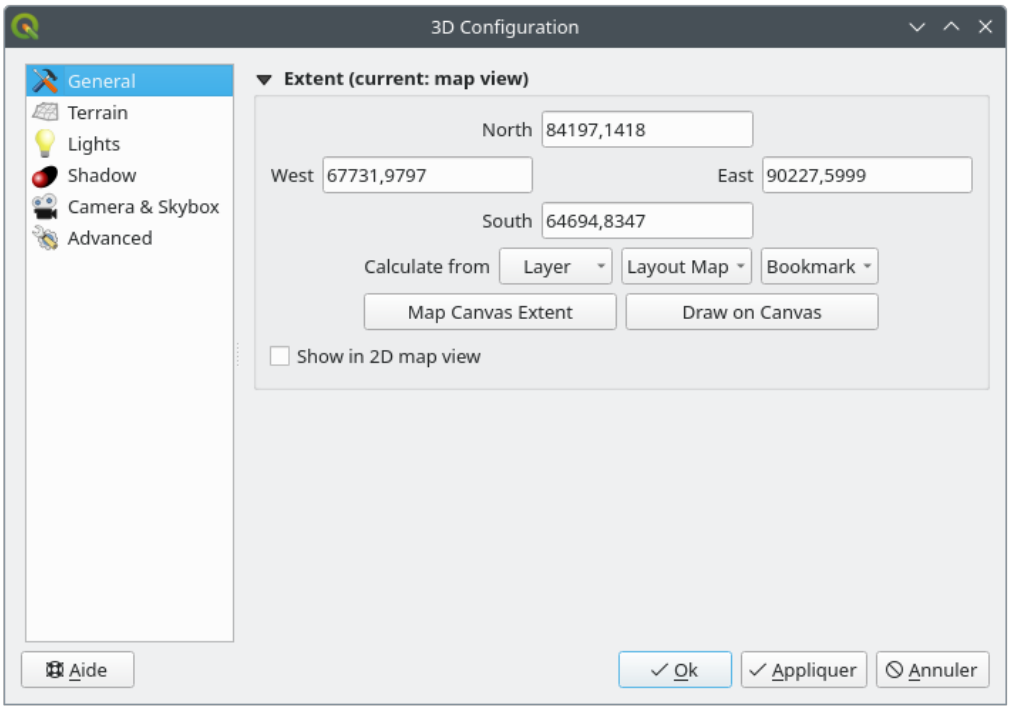


Fig. 11.23: The 3D Map General Configuration dialog

Under the *General* tab, you can:

- Limit the 3D scene's contents to a 2D map extent, using the *spatial extent selector*: terrain and features are clipped at the specified extent and only the parts falling within the extent are loaded in the scene. Elevation range of the terrain, mesh and pointcloud layers is also taken into account so that the camera is not positioned below the scene's contents, particularly when using the terrain's vertical scale setting to exaggerate the elevation differences.

  The set extent is also used as reference extent when pressing the Zoom full button in 3D map view.

- Press *Show in 2D map view* to display in the main map canvas a rubberband corresponding to the current extent of the 3D scene.

Terrain

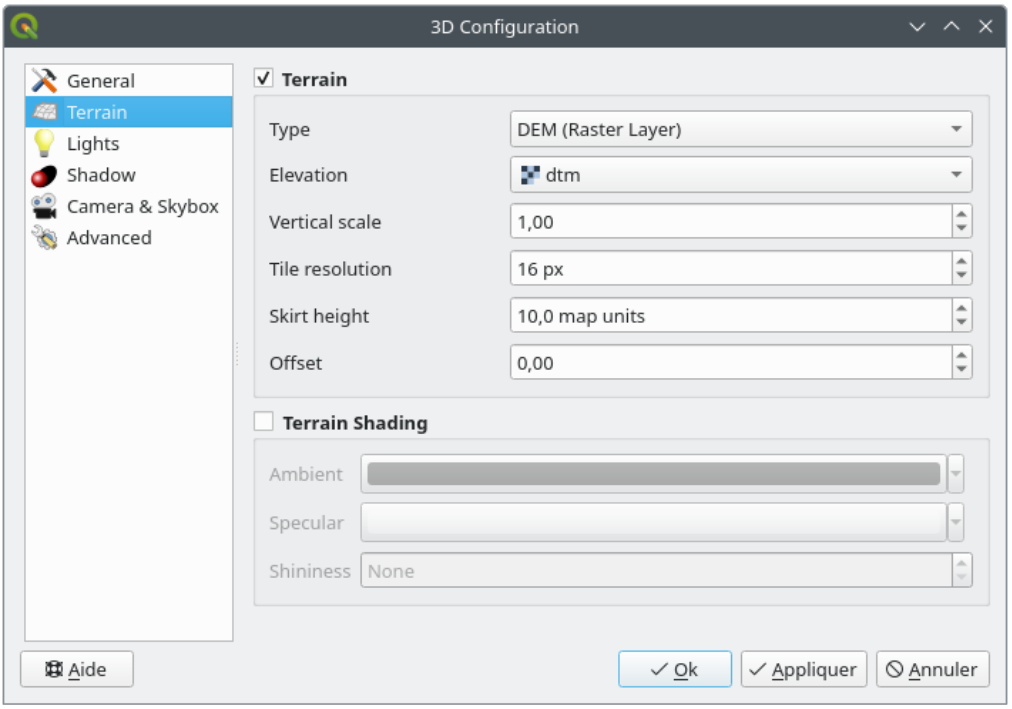


Fig. 11.24: The 3D Map Terrain Configuration dialog

- *Terrain*: Before diving into the details, it is worth noting that the terrain in a 3D view is represented by a hierarchy of terrain tiles and as the camera moves closer to the terrain, existing tiles that do not have sufficient details are replaced by smaller tiles with more details. Each tile has mesh geometry derived from the elevation raster layer and texture from 2D map layers.
  - The elevation terrain *Type* can be:
    - a *Teren plat*
    - a loaded *DEM (Raster Layer)*
    - an *Online* service, loading elevation tiles produced by Mapzen tools – more details at https://registry.opendata.aws/terrain-tiles/
    - a loaded *Mesh* dataset
  - *Elevation*: Raster or mesh layer to be used for generation of the terrain. The raster layer must contain a band that represents elevation. For a mesh layer, the Z values of the vertices are used.
  - *Vertical scale*: Scale factor for vertical axis. Increasing the scale will exaggerate the height of the landforms.
  - *Tile resolution*: How many samples from the terrain raster layer to use for each tile. A value of 16px means that the geometry of each tile will consist of 16x16 elevation samples. Higher numbers create more detailed terrain tiles at the expense of increased rendering complexity.
  - *Skirt height*: Sometimes it is possible to see small cracks between tiles of the terrain. Raising this value will add vertical walls („skirts”) around terrain tiles to hide the cracks.
  - *Offset*: moves the terrain up or down, e.g. to adjust its elevation with respect to the ground level of other objects in the scene.

    This can be useful when there is a discrepancy between the height of the terrain and the height of layers in your scene (e.g. point clouds which use a relative vertical height only). In this case adjusting the terrain

elevation manually to coincide with the elevation of objects in your scene can improve the navigation experience.

- When a mesh layer is used as terrain, you can configure the *Triangles settings* (wireframe display, smooth triangles, level of detail) and the *Rendering colors settings* (as a uniform color or *color ramp based*). More details in the *Mesh layer 3D properties* section.
- *Terrain shading*: Allows you to choose how the terrain should be rendered:
  - Shading disabled - terrain color is determined only from map texture
  - Shading enabled - terrain color is determined using Phong's shading model, taking into account map texture, the terrain normal vector, scene light(s) and the terrain material's *Ambient* and *Specular* colors and *Shininess*

Lights

From the *Lights* tab, press the menu to add

- up to eight *Point lights*: emits light in all directions, like a sphere of light filling an area. Objects closer to the light will be brighter, and objects further away will be darker. A point light has a set position (*X*, *Y* and *Z*), a *Color*, an *Intensity* and an *Attenuation*
- up to four *Directional lights*: mimics the lighting that you would get from a giant flash light very far away from your objects, always centered and that never dies off (e.g. the sun). It emits parallel light rays in a single direction but the light reaches out into infinity. A directional light can be rotated given an *Azimuth*, have an *Altitude*, a *Color* and an *Intensity*.

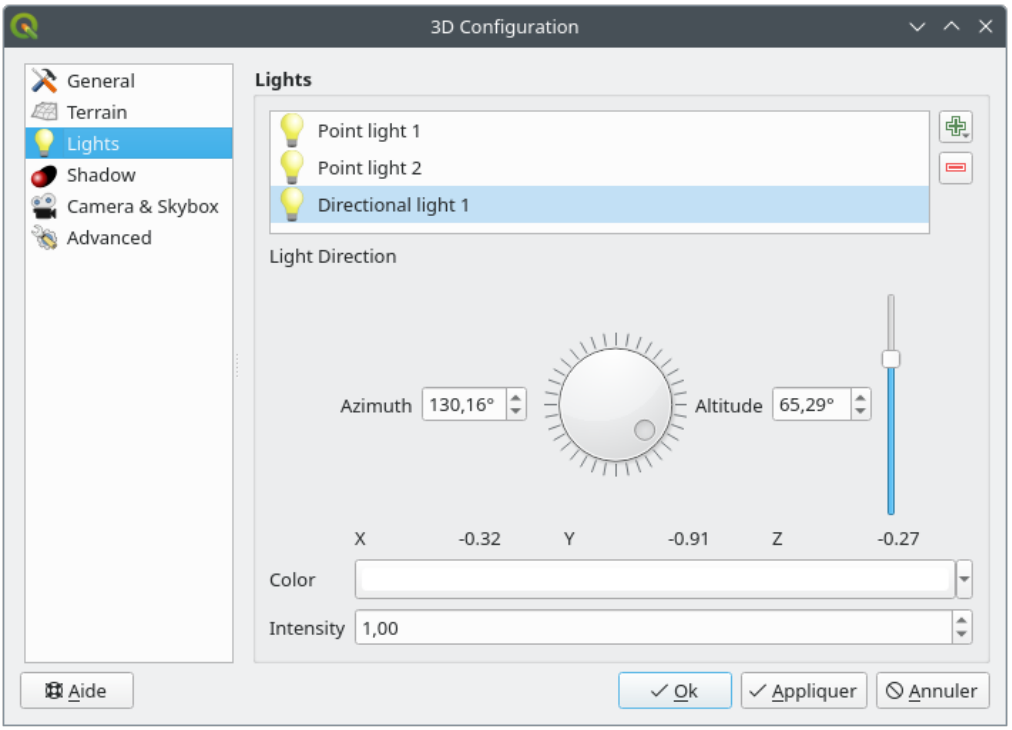


Fig. 11.25: The 3D Map Lights Configuration dialog

Shadow

Check *Show shadows* to display shadows within your scene, given:

- a *Directional light*
- a *Shadow rendering maximum distance*: to avoid rendering shadow of too distant objects, particularly when the camera looks up along the horizon
- a *Shadow bias*: to avoid self-shadowing effects that could make some areas darker than others, due to differences between map sizes. The lower the better
- a *Shadow map resolution*: to make shadows look sharper. It may result in less performance if the resolution parameter is too high.

Camera & Skybox

In this tab, you can control different parameters like camera, 3D axis, navigation synchronization and skybox.

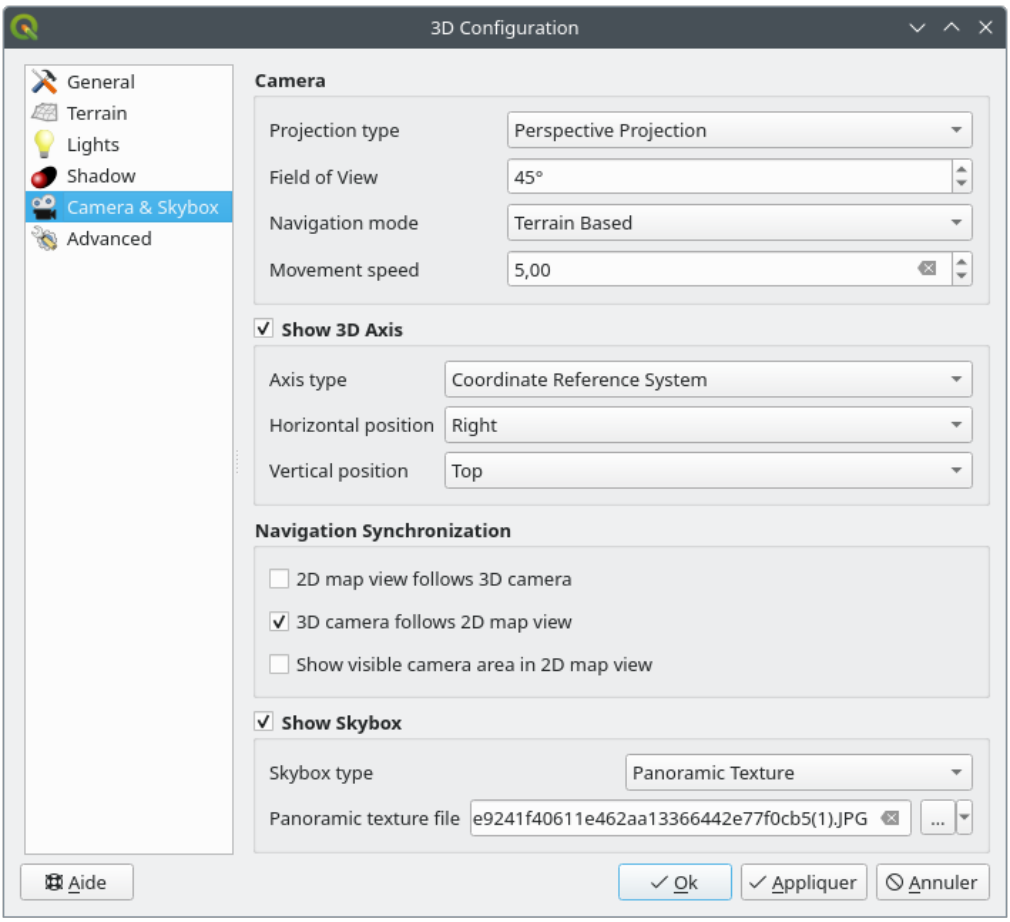


Fig. 11.26: The 3D Map Camera Configuration dialog

- The *Camera* parameter group overrides some *default camera settings* made in the *Settings* ► *Options* ► *3D* dialog.
- Check *Show 3D Axis* to enable 3D axis tool. This parameter group allows to set the axis type and its position.
  - With the *Coordinate Reference System* type an orthogonal axis will be represented.

  - With the *Cube* type, a 3D cube will be represented. The cube faces can be used to change the camera view: for example, click on the *north* face to set the camera to see from the north.

Sfat: Right-click the 3D axis to quickly set its position and type, and the camera view.

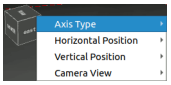


Fig. 11.27: The 3D Axis context menu

- The *Navigation Synchronization* parameter group adds options to synchronize 2D view with 3D camera position or 3D camera position with 2D view or bi directional synchronization. The last option displays the extent visible from the 3D camera over the 2D map view.
- Check ☐ *Show skybox* to enable skybox rendering in the scene. The skybox type can be:
  - *Panoramic texture*, with a single file providing sight on 360°
  - *Distinct faces*, with a texture file for each of the six sides of a box containing the scene

  Texture image files of the skybox can be files on the disk, remote URLs or embedded in the project (*more details*).

Advanced

- *Map tile resolution*: Width and height of the 2D map images used as textures for the terrain tiles. 256px means that each tile will be rendered into an image of 256x256 pixels. Higher numbers create more detailed terrain tiles at the expense of increased rendering complexity.
- *Max. screen error*: Determines the threshold for swapping terrain tiles with more detailed ones (and vice versa) - i.e. how soon the 3D view will use higher quality tiles. Lower numbers mean more details in the scene at the expense of increased rendering complexity.
- *Max. ground error*: The resolution of the terrain tiles at which dividing tiles into more detailed ones will stop (splitting them would not introduce any extra detail anyway). This value limits the depth of the hierarchy of tiles: lower values make the hierarchy deep, increasing rendering complexity.
- *Zoom levels*: Shows the number of zoom levels (depends on the map tile resolution and max. ground error).
- ☐ *Show labels*: Toggles map labels on/off
- ☐ *Show map tile info*: Include border and tile numbers for the terrain tiles (useful for troubleshooting terrain issues)
- ☐ *Show bounding boxes*: Show 3D bounding boxes of the terrain tiles (useful for troubleshooting terrain issues)
- ☐ *Show camera's view center*
- ☐ *Show camera's rotation center*
- ☐ *Show light sources*: shows a sphere at light source origins, allowing easier repositioning and placement of light sources relative to the scene contents
- ☐ *Show frames per second (FPS)*
- ☐ *Show debug overlay*: visual overlay which displays some useful debugging and profiling information. This allows in particular to quickly see the frame graph and the scene graph.

- *Show Eye Dome Lighting* (EDL): a post processing effect which enhances depth perception. Each pixel's depth (distance off the camera) is compared to its neighboring pixels» depth and gets highlighted according to that depth difference, making the edges stand out. Affects the whole scene and can be combined with *Screen Space Ambient Occlusion*. Following parameters can be controlled:
  - *Lighting strength*: increases the contrast, allowing for better depth perception
  - *Lighting distance*: represents the distance of the used pixels off the center pixel and has the effect of making edges thicker.
- Add screen-space *Ambient Occlusion* (SSAO): a post processing effect which also enhances depth perception by applying a darker shading to areas which are less exposed to ambient lighting. Affects the whole scene and can be combined with *Eye dome Lighting*. Following parameters can be controlled:
  - *Radius*: how far we will reach to calculate ambient occlusion
  - *Intensity*: how strong the effect should be (higher values make things darker)
  - *Occlusion threshold*: how many neighboring points need to be occluded for the effect to appear (lower values than 50% will make the output darker, but possibly providing greater range of occlusion)

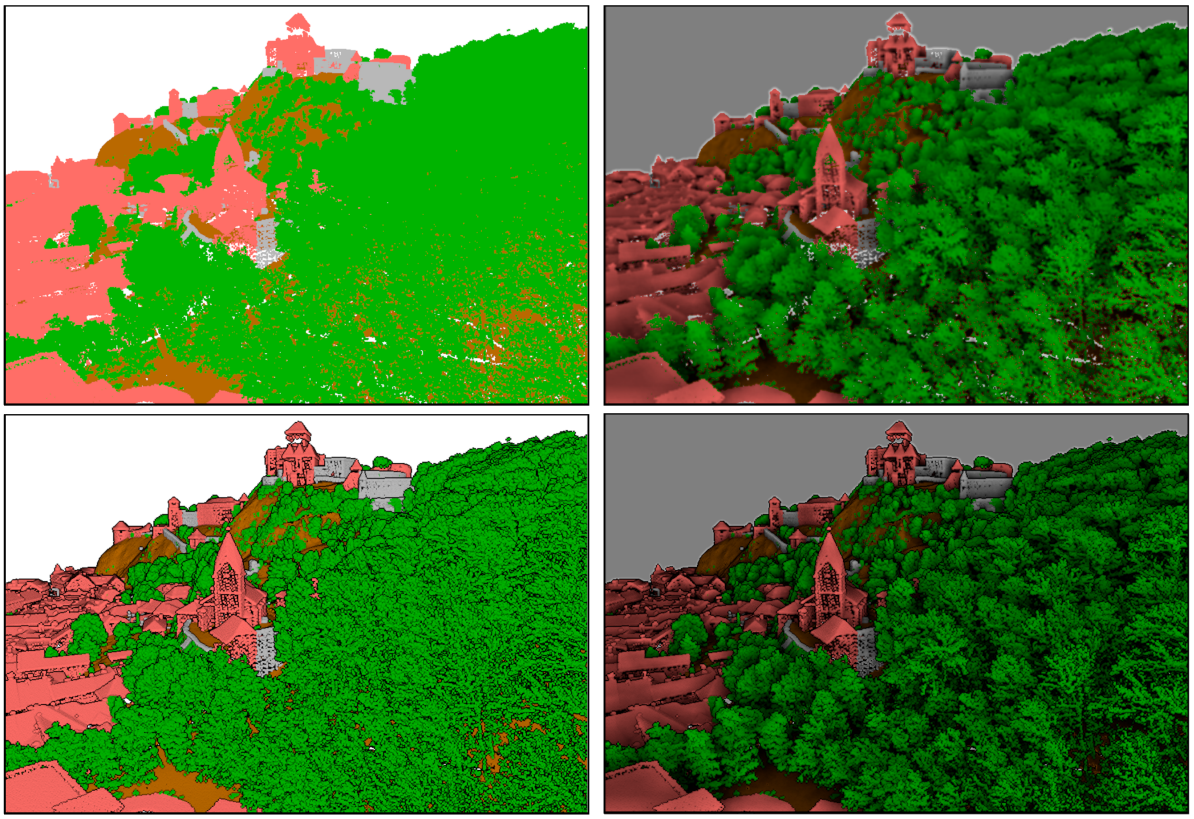


Fig. 11.28: Rendering Point clouds in 3D map using Eye Dome Lighting (EDL) and/or Screen-Space Ambient Occlusion (SSAO)

From top, left to right: No effect – SSAO only – EDL only – SSAO and EDL

- *Debug Shadow Map*: renders the scene as a red-black image from the point of view of the light used for shadows (for troubleshooting). The widget is set with a proportional *Size* to the 3D map view's, and docked in a *Corner*.
- *Debug Depth Map*: renders the scene's depth map as an image with nearer pixels being darker (for troubleshooting). The widget is set with a proportional *Size* to the 3D map view's, and docked in a *Corner*.

11.2.2 Navigation options

To explore the map view in 3D:

- Tilt the terrain (rotating it around a horizontal axis that goes through the center of the window)
  - Press the Tilt up and Tilt down tools
  - Press `Shift` and use the up/down keys
  - Drag the mouse forward/backward with the middle mouse button pressed
  - Press `Shift` and drag the mouse forward/backward with the left mouse button pressed
- Rotate the terrain (around a vertical axis that goes through the center of the window)
  - Turn the compass of the navigation widget to the watching direction
  - Press `Shift` and use the left/right keys
  - Drag the mouse right/left with the middle mouse button pressed
  - Press `Shift` and drag the mouse right/left with the left mouse button pressed
- Change the camera position (and the view center), moving it around in a horizontal plan
  - Drag the mouse with the left mouse button pressed, and the Camera control button enabled
  - Press the directional arrows of the navigation widget
  - Use the up/down/left/right keys to move the camera forward, backward, right and left, respectively
- Change the camera altitude: press the `Page Up`/`Page Down` keys
- Change the camera orientation (the camera is kept at its position but the view center point moves)
  - Press `Ctrl` and use the arrow keys to turn the camera up, down, left and right
  - Press `Ctrl` and drag the mouse with the left mouse button pressed
- Zoom in and out
  - Press the corresponding Zoom In and Zoom Out tools of the navigation widget
  - Scroll the mouse wheel (keep `Ctrl` pressed results in finer zooms)
  - Drag the mouse with the right mouse button pressed to zoom in (drag down) and out (drag up)

To reset the camera view, click the Zoom Full button on the top of the 3D canvas panel.

11.2.3 Creating an animation

An animation is based on a set of keyframes - camera positions at particular times. To create an animation:

1. Toggle on the Animations tool, displaying the animation player widget
2. Click the Add keyframe button and enter a *Keyframe time* in seconds. The *Keyframe* combo box now displays the time set.
3. Using the navigation tools, move the camera to the position to associate with the current keyframe time.
4. Repeat the previous steps to add as many keyframes (with time and position) as necessary.
5. Click the button to preview the animation. QGIS will generate scenes using the camera positions/rotations at set times, and interpolating them in between these keyframes. Various *Interpolation* modes for animations are available (eg, linear, inQuad, outQuad, inCirc… – more details at https://doc.qt.io/qt-5/qeasingcurve.html#EasingFunction-typedef).

The animation can also be previewed by moving the time slider. Keeping the *Loop* box checked will repeatedly run the animation while clicking ▶ stops a running animation.

Click Export animation frames to generate a series of images representing the scene. Other than the filename *Template* and the *Output directory*, you can set the number of *Frames per second*, the *Output width* and *Output height*.

11.2.4 3D vector layers

A vector layer with elevation values can be shown in the 3D map view by checking *Enable 3D Renderer* in the *3D View* section of the vector layer properties. A number of options are available for controlling the rendering of the 3D vector layer.

11.3 Elevation Profile View

The *Elevation Profile* panel is a plotting tool for side view, for visualizing elevation data along a line. It supports vector, raster, mesh and point cloud layers. Data can be of 2D or 3D type.

To add an elevation profile view, go to *View* ► *Elevation Profile* menu. You can add as many profile views as you want, and they can be docked, piled on top of each other, or floating.

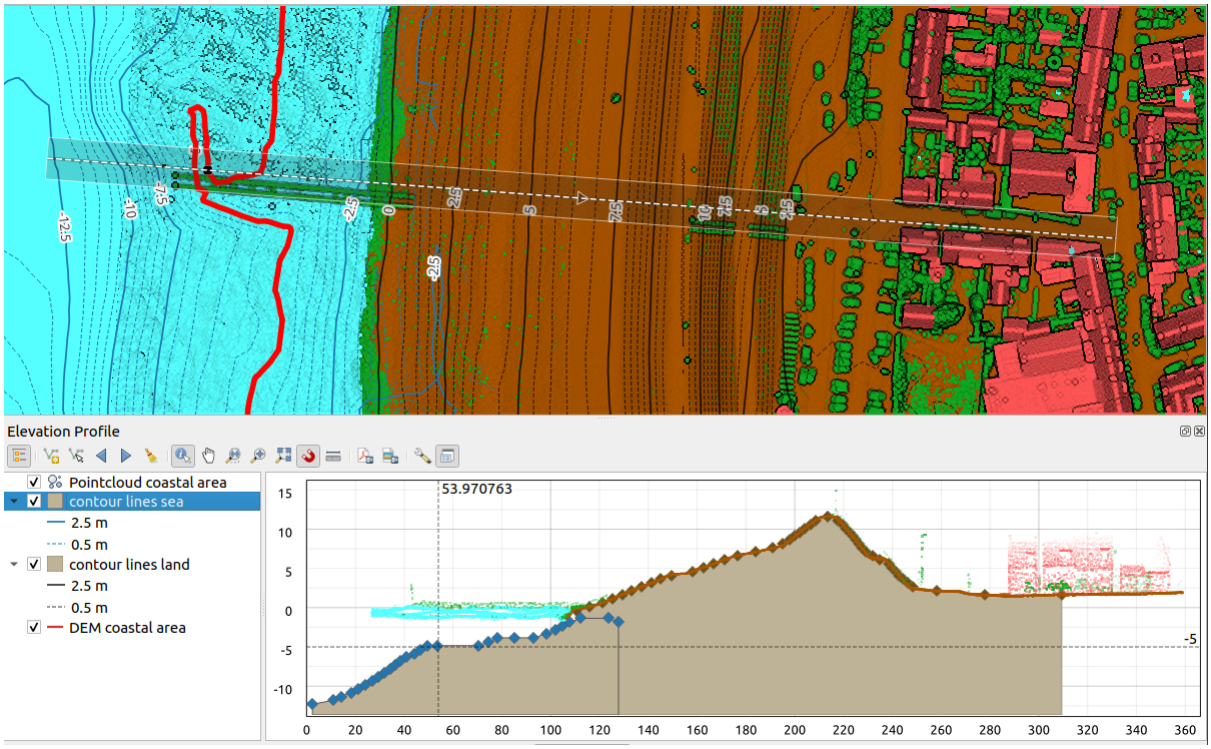


Fig. 11.29: Elevation Profile dialog embedded below main map canvas
Figure shows a coastline with a town behind a sea-dike. A larger tolerance applied to the elevation profile line returns more point cloud features.

11.3.1 The interface

At the top of the *Elevation Profile* panel, a toolbar provides you with the following tools:

Tabelul 11.4: Elevation Profile View toolbar

Tool	Shorcut	Description
Add Layers	hold `Ctrl` while drag-and-drop	Allows you to add new layers to a plot.
Show Layer Tree		Shows or hides a list of project layers to configure rendering in the profile view.
Capture Curve		Draws interactively a line over the map canvas to represent the profile curve.
Capture Curve From Feature		Picks an existing line feature on the map canvas and generate a profile curve along that line.
Nudge Left	`Ctrl+Al`	Allows you to slowly move the capture line across the map to the left (e.g. to find the optimal profile line based on the elevation).
Nudge Right	`Ctrl+Al`	Allows you to slowly move the capture line across the map to the right (e.g. to find the optimal profile line based on the elevation).
Clear		Removes the profile line and any plot displayed in the *Elevation Profile* view
Identify Features		Identifies features in the plot canvas via either a single click, or click-and-drag rectangle. Results are shown in the standard *Identify Results* dock.
Pan	`Space`	Click and drag to pan the plot canvas. Can also be done with the middle mouse button.
Zoom X Axis		Zooms in/out along the horizontal axis, keeping the vertical ratio
zoom	`Ctrl+Sp`	Click or click-and-drag a rectangle over the plot to zoom in. Press `Alt` and click to instead zoom out.
Zoom Full		Zooms the *Elevation Profile* view to the extent of the capture line
Enable Snapping		Allows to snap to the edge or vertex of the plot features in the profile view. Convenient for accurate retrieval of coordinates or distance measurements.
Measure Distances		Measures horizontal and vertical distances
Export As PDF		Exports plots to PDF (as high quality vector objects)
Export As Image		Exports plots to several image formats
Export Results		Exports plots as 3D features, 2D Profile and Distance/Elevation Table. (The results can be saved as DXF files, CSV files or any of the standard writable vector spatial formats)
Options ►		Provides access to configuration settings of the profile elevation line.
► *Lock distance/elevation scales*		Ensures that the horizontal and vertical scales are always kept equal (so that e.g. a 45° slope will appear as a 45° slope in the profile view).
► *Distance units*		Allows to render distances in the profile chart with units other than the map canvas units.
► *Tolerance*		Sets how far from the actual profile line a point can reside within to be included in the results. Uses the map units and is ignored by other layer or geometry types.

continues on next page

Tabelul 11.4 – continuare din pagina precedentă

Tool	Shorcut	Description
Dock Elevation Profile View		Switch between docked and floating status of the view

In the bottom left, a copy of the *Layers* panel can be displayed pushing the *Show Layer Tree* button. This is however an independent widget, with its own set of visible layers, in a custom stack order. It allows you to control layers rendering and behavior within the plot canvas:

- Tick the box next to the layer name to set whether it should be rendered in the plot canvas
- Drag-and-drop layers up or down to change the order of the layers
- style rendering of layers in the profile view: double-click a layer or right-click and select *Properties…* to open the layer's *Elevation* properties tab for configuration. A summary of elevation settings is displayed as tooltip when hovering over the layer.

On the right of the layer tree, the plot canvas is the main place you can preview the elevation profile of the enabled layers. It is based on a graduated grid in which the horizontal axis displays the length of the profile line and the vertical axis displays the Z elevation of the observed features. It also allows a set of interactions such as zooming, panning, measuring, identifying features, … using the tools at the top.

11.3.2 Creating an elevation profile

To create a profile view, you can:

1. Go to *View* ► *Elevation Profile* menu. The *Elevation profile* panel opens.
2. Create the profile line along which the terrain and the features will be rendered. Select a drawing tool:
   - Capture Curve: click left over the main map canvas to add vertices and click right to finalize a line that will be used as profile line
   - or Capture Curve From Feature: click a line feature on the map canvas to select it as the profile line. If multiple features are present at the clicked point then a popup menu will appear allowing you to select among them.

   All the line digitizing capabilities such as the *snapping options*, *tracing*, *digitizing techniques* or the advanced *digitizing panel* are available for use.

   The plot canvas may start rendering some features.
3. The next step is to configure the elevation properties of the layers you want to visualize.
   1. Push on the Show Layer Tree button to display the list of layers. By default, not all the layers of the project are loaded and referenced in the profile tool; e.g., raster layers which are not marked as having elevation data in their properties are ignored. Clicking the Add layers button will show a filtered list of possible layers which can be added to the plot, but which currently aren't in the plot. Applying the dialog with selected layers will automatically mark them as having elevation data and immediately add them to the plot.

      Sfat: Holding `Ctrl` key, it is possible to drag and drop additional layers from the main layer tree to one in the elevation profile tool.

   2. Toggle visibility of the layers you are interested in. These are the only ones rendered in the profile view and selected layers can be different from the main *Layers* panel's.

3. Double-click a layer name or right-click and select *Properties*. The *Elevation* properties tab of the layer opens. This is the place you configure how each feature or terrain should render on the profile view. Available properties depend on the layer type:

   - *Raster Elevation Properties*
   - *Vector Elevation Properties*
   - *Point Cloud Elevation Properties*
   - *Mesh Elevation Properties*

   The profile view starts rendering terrain or features of active layers crossing the given profile curve, as soon as they have elevation properties configured.

4. Under Options drop-down menu, you can set the *Tolerance* value. This value is used to create a flat buffer around the elevation profile line, visible in the main map canvas. Any visible point feature overlapping that buffer will be captured in the plot canvas.

11.3.3 Interacting with the profile Views

When an elevation profile line is created, the plot canvas zooms to its full extent. On the X-axis you can see the length of the profile and, on the Y-axis the height range between minimum and maximum height captured, both in map units.

When you move the mouse pointer in the elevation profile view, you will see two crossing dot lines:

- the vertical line shows the height information
- the horizontal line shows the distance from the beginning of the elevation profile Line

When you move the mouse pointer in the elevation profile canvas, you can also see a black dot move along the elevation profile line on the main map canvas. At the middle of the profile line, an arrow indicates its direction.

As for the main map canvas, QGIS provides means to navigate on the plot canvas:

- Pan is used to move the elevation profile extent in any direction you want. Holding `Space` key while moving the mouse also shifts the plot canvas extent.
- Zoom X Axis is used to zoom in along the horizontal axis, keeping the ratio of the vertical axis (the elevation) the same. Left click to stretch out the plot along the X axis, with the clicked point at the center of the axis, or drag a rectangle to stretch out the plot along the X axis to the rectangle width. Hold `Alt` while using Zoom X Axis to zoom out along that axis.
- Zoom is used to zoom in on a certain point (using left click), or to a certain extent (dragging a rectangle on the area). Hold `Alt` while using Zoom to instead zoom out. In combination with the `Ctrl` key you can zoom in or out more smoothly.
- Zoom Full is the default zoom level used at the beginning and shows the full extent of the profile line, with all returned features. Use it to reset the zoom level.

It is also possible to interact with the elements displayed in the plot canvas:

- Press Enable snapping button to accurately catch points, vertices or edges of the features, for an accurate measurement or coordinates report.
- Identify Features is used to identify features of the visible layers in the layer tree. You can drag a rectangle across several features in the profile view to query all of them. When compatible with the format (e.g. vector, point cloud), these features will be highlighted in the main map canvas.
- Measurement Distances: click or select two points in the plot canvas to report the horizontal *Distance*, the *Elevation* and the *Total distance* between them, in map units.

- Nudge Left and Nudge right are used to shift the position of the elevation profile line in the map canvas to its left or right. The plot canvas will be redrawn, showing features and terrain overlapping the profile line buffer. The line is moved sideways using the *Tolerance* value in Options menu.

Atenționare: Closing an elevation profile view or the project currently removes the view from the project.

For more details, give a look to QGIS elevation profile/cross section tool – a deep dive!, a presentation done by Nyall Dawson.

CAPITOLUL 12

Instrumente generale

12.1 Ajutor contextual

Whenever you need help on a specific topic, you can access the corresponding page in the current User Manual via the *Help* button available in most dialogs — please note that third-party plugins can point to dedicated web pages.

12.2 Panouri

By default, QGIS provides many panels to work with. Some of these panels are described below while others may be found in different parts of the document. A complete list of default panels provided by QGIS is available via the *View* ► *Panels* ► menu and mentioned at *Panouri*.

12.2.1 Panoul Straturilor

The *Layers* panel (also called the `map legend`) lists all the layers in the project and helps you manage their visibility and shape the map. You can show or hide the panel by pressing `Ctrl+1`.

QGIS provides a variety of ways to add layers to a project:

- using the *Add* button from the dedicated data provider tab in the *Data source manager* dialog
- from QGIS *Browser panel* or *DB Manager*: double-click, drag-and-drop files and layers onto QGIS or use the contextual menu
- drag-and-drop files from the Operating System files explorer onto QGIS

In all these scenarios, you can open one or many layers at a time. New layers are added to the *Layers* panel:

1. if dropped over the *Layers* panel, at the exact location they are released
2. in the other cases, at a location that respects the global *behavior used when adding new layers* setting
3. and in case of multiple layers, they are sorted in a way that increases the chance of their stacking being logical and features being visible as most as possible, using the following logic (top to bottom):
    - vector point layers
    - vector line layers

- vector polygon layers
- point cloud layers
- mesh layers
- raster layers

At the top of the *Layers* panel, a toolbar allows you to:

- Open the layer styling dock (F7): toggle the *Layer Styling* panel on and off.
- Add new group: see *Interact with groups and layers*
- Manage Map Themes: control visibility of layers and arrange them in different *map themes*.
- filter layers in the legend tree:
  - *Filter Legend by Map Content*: only the layers that are set visible and whose features intersect the current map canvas have their style rendered in the layers panel. Otherwise, a generic NULL symbol is applied to the layer. Based on the layer symbology, this is a convenient way to identify which kind of features from which layers cover your area of interest.
  - *Show Private Layers*: a convenient shortcut to display and interact with *private layers* in the *Layers* panel without modifying the project settings.
- Filter Legend by Expression: apply an expression to remove styles from the selected layer tree that have no feature satisfying the condition. This can be used to highlight features that are within a given area/feature of another layer. From the drop-down list, you can edit and clear the expression currently applied.
- Expand All or Collapse All layers and groups in the layers panel.
- Remove Layer/Group currently selected.

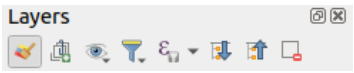


Fig. 12.1: Layer Toolbar in Layers Panel

Notă: Tools to manage the layers panel are also available for map and legend items in print layouts

Configuring map themes

The Manage Map Themes drop-down button provides access to convenient shortcuts to manipulate visibility of the layers in the *Layers* panel:

- *Arată Toate Straturile*
- *Ascunde Toate Straturile*
- *Arată Straturile Selectate*
- *Ascunde Straturile Selectate*
- *Toggle Selected Layers*: changes the visibility of the first selected layer in the panel, and applies that state to the other selected layers. Also accesible through `Space` shortcut.
- *Toggle Selected Layers Independently*: changes the visibility status of each selected layer

- *Hide Deselected Layers*

Beyond the simple control of layer visibility, the Manage Map Themes menu allows you to configure **Map Themes** in the legend and switch from one map theme to another. A map theme is a **snapshot** of the current map legend that records:

- the layers set as visible in the *Layers* panel
- **and** for each visible layer:
  - the reference to the *style* applied to the layer
  - the visible classes of the style, ie the layer checked node items in the *Layers panel*. This applies to *symbologies* other than the single symbol rendering
  - the collapsed/expanded state of the layer node(s) and the group(s) it's placed inside

To create a map theme:

1. Check a layer you want to show
2. Configure the layer properties (symbology, diagram, labels…) as usual
3. Expand the *Style* ► menu at the bottom and click on *Add…* to store the settings as *a new style embedded in the project*

   Notă: A map theme does not remember the current details of the properties: only a reference to the style name is saved, so whenever you apply modifications to the layer while this style is enabled (eg change the symbology rendering), the map theme is updated with new information.

   ---

4. Repeat the previous steps as necessary for the other layers
5. If applicable, expand or collapse groups or visible layer nodes in the *Layers* panel
6. Click on the Manage Map Themes button on top of the panel, and *Add Theme…*
7. Enter the map theme's name and click *OK*

The new theme is listed in the lower part of the drop-down menu.

You can create as many map themes as you need: whenever the current combination in the map legend (visible layers, their active style, the map legend nodes) does not match any existing map theme contents as defined above, click on *Add Theme…* to create a new map theme, or use *Replace Theme* ► to update a map theme. You can rename the active map theme with *Rename Current Theme…* or use the *Remove Current Theme* button to delete it.

Map themes are helpful to switch quickly between different preconfigured combinations: select a map theme in the list to restore its combination. All configured themes are also accessible in the print layout, allowing you to create different map items based on specific themes and independent of the current main canvas rendering (see *Map item layers*).

Overview of the context menu of the Layers panel

At the bottom of the toolbar, the main component of the Layers panel is the frame listing all the layers added to the project, optionally organized in groups. A layer with a checked box next to it displays its contents overlapping the map canvas extent, unless a *scale-based visibility* is set. A layer can be selected and dragged up or down in the legend to change the Z-ordering. Z-ordering means that layers listed nearer the top of the legend are drawn over layers listed lower down in the legend. Also a layer or a group of layers can be dragged across several QGIS instances.

Notă: The Z-ordering behavior can be overridden by the *Layer Order* panel.

Depending on the item selected in the panel, a right-click shows a dedicated set of options presented below.

Tabelul 12.1: Contextual menus from *Layers* panel items

Opţiune	Grup	Strat Vectorial	Strat Raster	Mesh Layer	Point Cloud Layer	3D Lay
Zoom to Layer(s)/Group	✓	✓	✓	✓	✓	✓
Zoom to Selection		✓				
Arată în Vizualizare		✓	✓	✓	✓	✓
Arată Numărul de Entităţi		✓				
Show Label		✓				
Copiere Strat/Grup	✓	✓	✓	✓	✓	✓
Redenumire Strat/Grup	✓	✓	✓	✓	✓	✓
Zoom to Native Resolution (100%)			✓			
Stretch Using Current Extent			✓			
Update SQL Layer…		✓				
Edit Virtual Layer…		✓				
Adăugare Grup	✓					
Duplicate Layer		✓	✓	✓	✓	✓
Eliminare Strat/Grup…	✓	✓	✓	✓	✓	✓
Move Out of Group		✓	✓	✓	✓	
Move to Top	✓	✓	✓	✓	✓	
Move to Bottom	✓	✓	✓	✓	✓	
Check and all its Parents		✓	✓	✓	✓	
Group Selected		✓	✓	✓	✓	
Deschide Tabela de Atribute		✓				
Comută Editarea		✓		✓		
Editări Curente ►		✓		✓		
Filtrare…		✓	✓		✓	
Change Data Source…		✓	✓	✓	✓	
Repair Data Source…		✓	✓	✓	✓	
Actions on selections ► (in edit mode)		✓				
► *Duplicate Feature*		✓				
► *Duplicate Feature and Digitize*		✓				
Set Layer Scale Visibility…		✓	✓	✓	✓	✓
Zoom to Visible Scale		✓	✓	✓	✓	
Layer CRS ►		✓	✓	✓	✓	✓
► *Set Project CRS from Layer*		✓	✓	✓	✓	✓
► *Set to..* (recent CRSs)				✓	✓	✓
► *Set Layer CRS…*		✓	✓	✓	✓	✓
Set Group CRS…	✓					
Set Group WMS Data…	✓					
Mutually Exclusive Group	✓					

continues on next pag

Tabelul 12.1 – continuare din pagina precedentă

Opţiune	Grup	Strat Vectorial	Strat Raster	Mesh Layer	Point Cloud Layer	3D Lay
Check and all its children (Ctrl-click)	☑					
Uncheck and all its children (Ctrl-click)	☑					
Make Permanent		☑				
Export ►	☑	☑	☑	☑	☑	☑
► *Save As…*			☑			
► *Save Features As…*		☑				
► *Save Selected Features As…*		☑				
► *Save As Layer Definition File…*	☑	☑	☑	☑	☑	☑
► *Save As QGIS Layer Style File…*		☑	☑	☑	☑	☑
Stiluri ►		☑	☑	☑	☑	☑
► *Copy Style*		☑	☑	☑	☑	☑
► *Paste Style*	☑	☑	☑	☑	☑	
► *Add…*		☑	☑	☑	☑	☑
► *Rename Current…*		☑	☑	☑	☑	☑
► *Edit symbol…*		☑				
► *Copy Symbol*		☑				
► *Paste Symbol*		☑				
Add Layer Notes…		☑	☑	☑	☑	☑
Edit Layer Notes…		☑	☑	☑	☑	
Remove Layer Notes		☑	☑	☑	☑	
Proprietăţi…		☑	☑	☑	☑	☑

For GRASS vector layers, Toggle editing is not available. See section *Digitizarea și editarea unui strat vectorial GRASS* for information on editing GRASS vector layers.

Interact with groups and layers

Layers in the legend window can be organized into groups. There are different ways to do this:

1. Press the icon to add a new group. Type in a name for the group and press `Enter`. Now click on an existing layer and drag it onto the group.
2. Select more than one layer, then press the icon. The selected layers are automatically inserted into the new group.
3. Select some layers, right-click in the legend window and choose *Group Selected*. The selected layers will automatically be placed in a new group.

To move a layer out of a group, drag it out, or right-click on it and choose *Move Out of Group*: the layer is moved from the group and placed above it. Groups can also be nested inside other groups. If a layer is placed in a nested group, *Move Out of Group* will move the layer out of all nested groups.

To move a group or layer to the top of the layer panel, either drag it to the top, or choose *Move to Top*. If you use this option on a layer nested in a group, the layer is moved to the top in its current group. The *Move to Bottom* option follows the same logic to move layers and groups down.

The checkbox for a group will show or hide the checked layers in the group with one click. With `Ctrl` pressed, the checkbox will also turn on or off all the layers in the group and its sub-groups.

`Ctrl`-click on a checked / unchecked layer will uncheck / check the layer and all its parents.

Enabling the **Mutually Exclusive Group** option means you can make a group have only one layer visible at the same time. Whenever a layer within the group is set visible the others will be toggled not visible.

It is possible to select more than one layer or group at the same time by holding down the `Ctrl` key while clicking additional layers. You can then move all selected layers to a new group at the same time.

You may also delete more than one layer or group at once by selecting several items with the `Ctrl` key and then pressing `Ctrl+D`: all selected layers or groups will be removed from the layers list.

More information on layers and groups using indicator icon

In some circumstances, formatting or icons next to the layer or group in the *Layers* panel change to give more information about the layer/group. These elements are:

- to indicate that the layer is in edit mode and you can modify the data
- to indicate that the layer being edited has some unsaved changes
- to indicate *a filter* applied to the layer. Hover over the icon to see the filter expression and double-click to update the query
- to identify layers that are *required* in the project, hence non removable
- to identify an *embedded group or layer* and the path to their original project file
- to identify a layer whose data source was not available at the project file opening (see *Gestionarea căilor de fișiere întrerupte*). Click the icon to update the source path or select *Repair Data Source…* entry from the layer contextual menu.
- to remind you that the layer is a *temporary scratch layer* and its content will be discarded when you close this project. To avoid data loss and make the layer permanent, click the icon to store the layer in any of the GDAL vector formats supported by QGIS.
- to identify a layer used in *offline editing mode*.
- to identify a layer that has no/unknown CRS
- for layers with coordinates stored in a coordinate reference system which is inherently low accuracy (requires the *corresponding setting* to be enabled)
- to identify a temporal layer controlled by canvas animation
- to identify a layer that has *notes* associated
- A grayed name, when the map canvas current scale is outside the layer's visibility scale range (as set in its *Rendering* properties). Select the contextual menu *Zoom to Visible Scale* option to zoom the map to the layer's nearest visibility scale bound.

Control layers rendering through grouping

Groups are a means of structuring layers within a tree in the project but they can also impact how their component layers are rendered, namely as a single flattened object during map renders.

The option for such a rendering is available within the *Layer Styling* panel whenever a group is selected. Under the Symbology tab, check *Render Layers as a Group* to enable a set of options to control the appearance of the child layers as a whole, instead of individual layers:

- *Opacity*: Features from child layers which are obscured by other child layers remain obscured, and the opacity applies to the „whole of group" only.

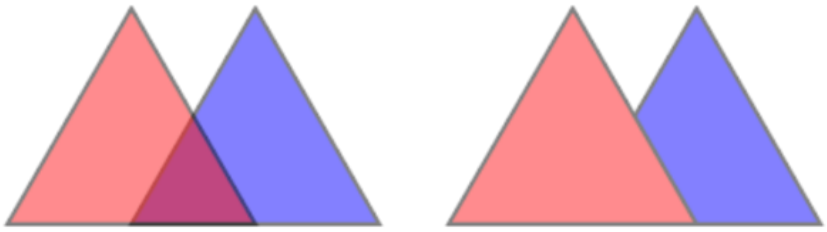


Fig. 12.2: Setting opacity on layers vs on a group
The image on the left shows two layers being rendered at 50% opacity (underlying features are visible, but semi-masked by the 50% red feature on top). The second image shows the result of setting the opacity on the group (parts of the blue underlying child layer is completely obscured by the red layer on top and then the result is rendered at 50% opacity).

- *Blend modes*: Just like opacity, setting a *blend mode* (like multiply, overlay, …) for an entire group results first in flattening features of child layers, with upper ones obscuring lower. The rendering is then obtained by blending the flat group and the layers sitting below the group.
  - When the child layers have blend modes assigned, it is applied before flattening but the scope is restricted to only affecting other child layers from that group, and not other layers sitting below the whole group.
  - Some more *blending modes* options are available for child layers in groups, in their *Symbology* tab which perform „clipping" style operations on other child layers during the render. You can e.g. clip the render of one layer's content by the content in a second „mask" layer.
- *Layer effects*: applies *effects* only to the flattened render of the child layers; So e.g. a drop shadow effect applied to the group would not be visible for obscured child layers.

When a group is set to *Render layers as a group*, then only the group will be shown in the *Layer Order* panel list. Group children will not be visible in this order list, as their ordering is determined by the placement of the group layer.

Editing layer style

From the *Layers* panel, you have shortcuts to change the layer rendering quickly and easily.

Right-click on a layer and select *Styles* ► in the list in order to:

- see the *styles* currently available for the layer. If you defined many styles for the layer, you can switch from one to another and your layer rendering will automatically be updated on the map canvas.
- copy part or all of the current style, and when applicable, paste a copied style from another layer
- *Rename current…* style
- *Add* a new style (which is actually a copy of the current one)
- or *Remove current* style (only when multiple styles are available).

Sfat: Quickly share a layer style

From the context menu, copy the style of a layer and paste it to a group or a selection of layers: the style is applied to all the layers that are of the same type (vector, raster, mesh, point cloud, …) as the original layer and, for vector layers, have the same geometry type (point, line or polygon).

When using a symbology based on features classification (e.g. *categorized*, *graduated* or *rule-based* for vector layers, or *classification* for point clouds), right-clicking a class entry in the *Layers* panels makes it possible to edit the visibility of the classes (and their features) and avoid (un)checking them one by one:

- *Toggle Items*
- *Show All Items*
- *Hide All Items*

With vector layer, the contextual menu of a class leaf entry also gives access to:

- *Select features*: selects in the layer all the features matching that class
- *Show in attribute table*: opens an attribute table filtered to only the features matching that class
- update the *symbol color* using a **Color Wheel**. For convenience, the recently used colors are also available at the bottom of the color wheel.
- *Edit Symbol…*: opens the *Symbol Selector* dialog to change feature symbol (symbol, size, color…).
- *Copy Symbol*
- *Paste Symbol*

Sfat: Double-clicking a class leaf entry also opens the *Symbol Selector* dialog.

12.2.2 Layer Styling Panel

The *Layer Styling* panel (also enabled with `Ctrl+3`) is a shortcut to some of the functionalities of the *Layer Properties* dialog. It provides a quick and easy way to define the rendering and the behavior of a layer, and to visualize its effects without having to open the layer properties dialog.

In addition to avoiding the blocking (or „modal”) layer properties dialog, the layer styling panel also avoids cluttering the screen with dialogs, and contains most style functions (color selector, effects properties, rule edit, label substitution…): e.g., clicking color buttons inside the layer style panel causes the color selector dialog to be opened inside the layer style panel itself rather than as a separate dialog.

From a drop-down list of current layers in the layer panel, select an item and:

- Depending on the active item, set:
  - *Symbology* for groups (see *Control layers rendering through grouping*)
  - *Symbology*, *Transparency*, and *Histogram* properties for raster layer. These options are the same as in the *Dialogul Proprietăților Rasterului*.
  - *Symbology*, *Labels*, *Mask* and *3D View* properties for vector layer. These options are the same as in the *Dialogul Proprietăților Vectoriale* and can be extended by custom properties introduced by third-party plugins.

  - *Symbology* and *3D View* properties for mesh layer. These options are the same as in the *Mesh Dataset Properties*.
  - *Symbology*, *3D View* and *Elevation* properties for point cloud layer. These options are the same as in the *Point Clouds Properties*.
- Enable and configure *global map shading* properties
- Manage the associated style(s) in the *Style Manager* (more details at *Managing Custom Styles*).
- See the *History* of changes you applied to the layer style in the current project: you can therefore cancel or restore to any state by selecting it in the list and clicking *Apply*.

Another powerful feature of this panel is the *Live update* checkbox. Tick it to render your changes immediately on the map canvas: you no longer need to click the *Apply* button.

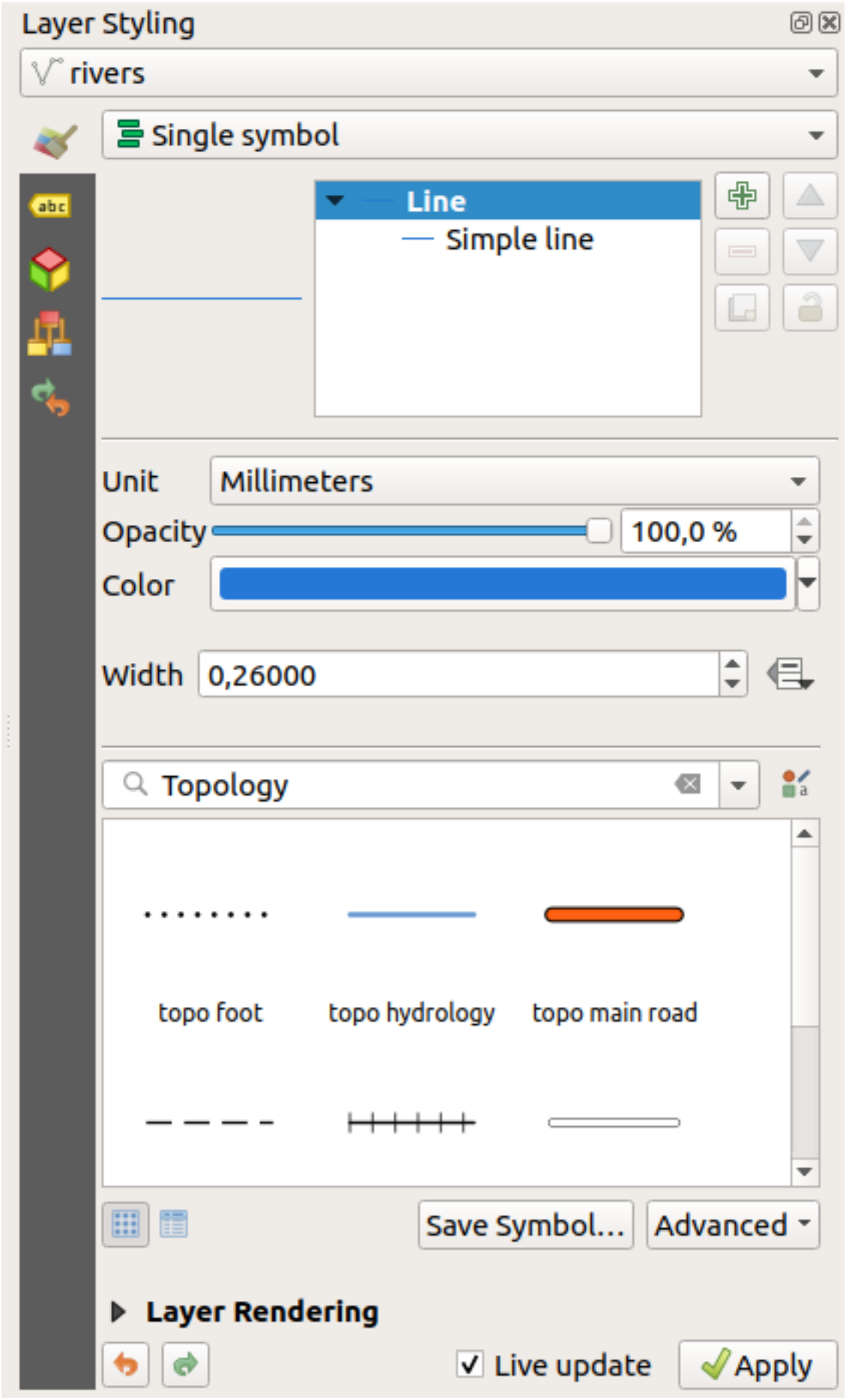


Fig. 12.3: Defining a layer's symbology from the layer styling panel

12.2.3 Layer Order Panel

By default, layers shown on the QGIS map canvas are drawn following their order in the *Layers* panel: the higher a layer is in the panel, the higher (hence, more visible) it'll be in the map view.

You can define a drawing order for the layers independent of the order in the layers panel with the *Layer Order* panel enabled in *View* ► *Panels* ► menu or with `Ctrl+9`. Check *Control rendering order* underneath the list of layers and reorganize the layers in the panel as you want. This order becomes the one applied to the map canvas. For example, in Fig. 12.4, you can see that the `airports` features are displayed over the `alaska` polygon despite those layers» respective placement in the Layers panel.

Unchecking *Control rendering order* will revert to default behavior.

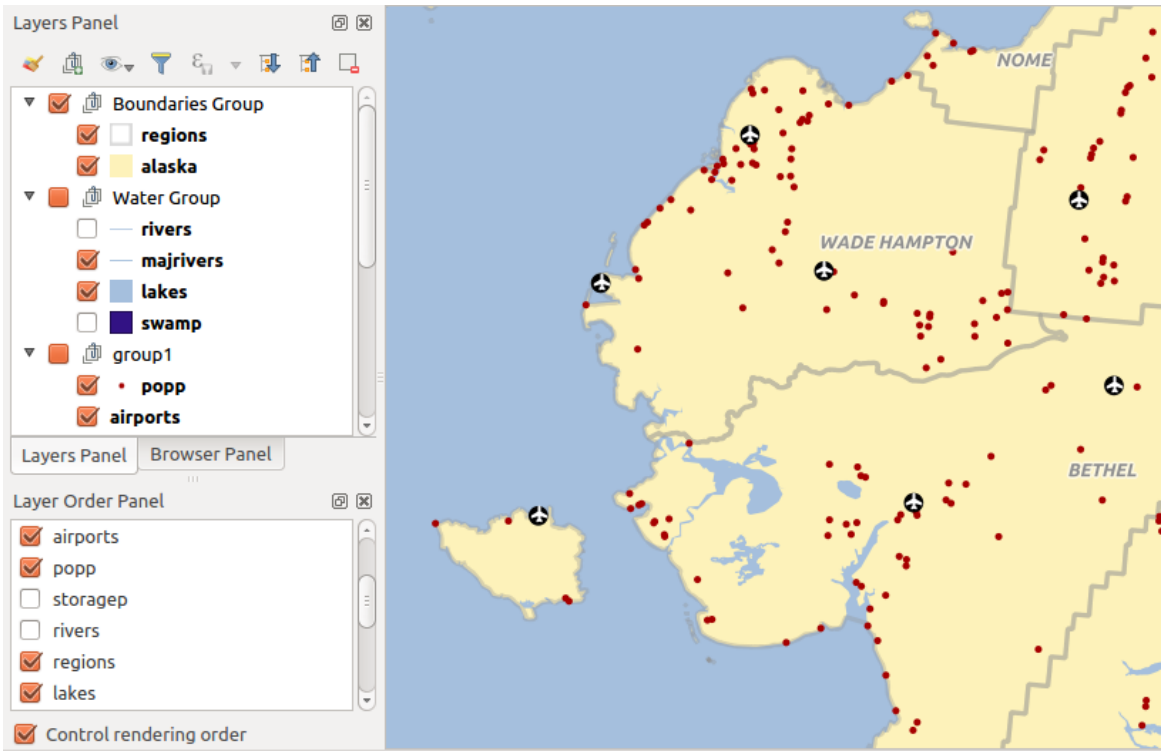


Fig. 12.4: Define a layer order independent of the legend

12.2.4 Overview Panel

The *Overview* panel (`Ctrl+8`) displays a map with a full extent view of some of the layers. The Overview map is filled with layers using the *Show in Overview* option from the *Layer* menu or in the layer contextual menu. Within the view, a red rectangle shows the current map canvas extent, helping you quickly to determine which area of the whole map you are currently viewing. If you click-and-drag the red rectangle in the overview frame, the main map view extent will update accordingly.

Note that labels are not rendered to the map overview even if the layers used in the map overview have been set up for labeling.

12.2.5 Log Messages Panel

When loading or processing some operations, you can track and follow messages that appear in different tabs using the *Log Messages Panel*. It can be activated using the most right icon in the bottom status bar.

12.2.6 Undo/Redo Panel

For each layer being edited, the *Undo/Redo* (`Ctrl+5`) panel shows the list of actions carried out, allowing you quickly to undo a set of actions by selecting the action listed above. More details at *Undo and Redo edits*.

12.2.7 Statistical Summary Panel

The *Statistics* panel (`Ctrl+6`) provides summarized information on any vector layer. This panel allows you to select:

- the vector layer to compute the statistics on: it can be selected from the top drop-down menu, or synced with the active layer in the *Layers* panel using the *Follow selected layer* checkbox at the bottom of the statistics drop-down list
- the field or *expression* to use: for each layer, the last entry is remembered and automatically computed upon layer reselection.
- the statistics to return using the drop-down button at the bottom-right of the dialog. Depending on the field's (or expression's values) type, available statistics are:

Tabelul 12.2: Statistics available for each field type

Statistici	Șirul	Întreg	Zecimal	Data
Număr	☑	☑	☑	☑
Count Distinct Value	☑			☑
Count Missing value	☑	☑	☑	☑
Sum		☑	☑	
Media		☑	☑	☑
Abaterea Standard		☑	☑	
Standard Deviation on Sample		☑	☑	
Minimal value	☑	☑	☑	☑
Maximal value	☑	☑	☑	☑
Intervalul		☑	☑	☑
Minority	☑	☑	☑	
Majority	☑	☑	☑	
Variety		☑	☑	
Prima Quartilă		☑	☑	
A Treia Quartilă		☑	☑	
Inter Quartile Range		☑	☑	
Lungimea Minimă	☑			
Lungimea Maximă	☑			
Mean Length	☑			

The statistical summary can be:

- returned for the whole layer or *Selected features only*
- copied to the clipboard and pasted as a table in another application
- recalculated using the button when the underlying data source changes (eg, new or removed features/fields, attribute modification)

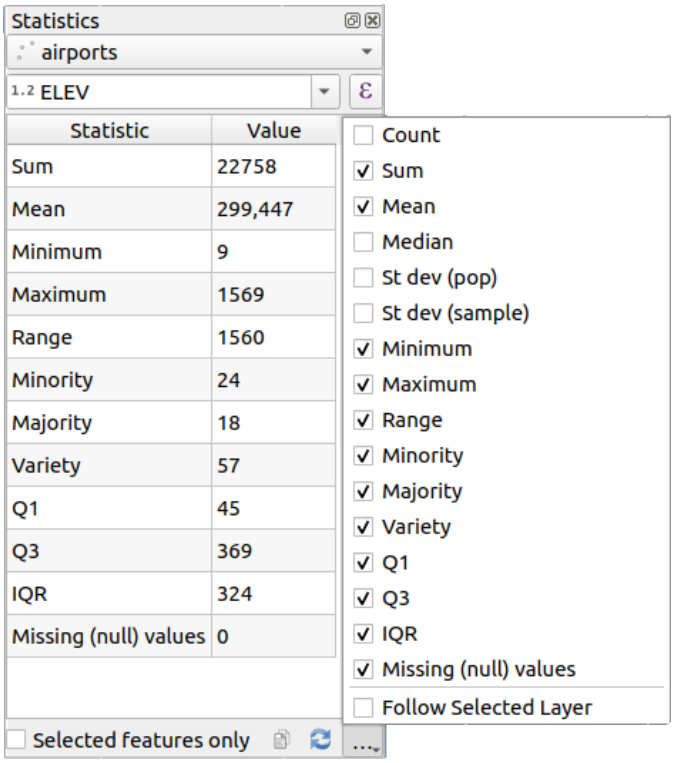


Fig. 12.5: Show statistics on a field

12.2.8 Debugging/Development Tools Panel

The *Debugging/Development Tools* panel (`F12`) provides a unified place for handling and debugging actions within QGIS. Available tools are organized under following tabs:

- *Network Logger*
- *Query Logger*
- *Profiler* - under this tab you can choose one out of these categories:
  - *Startup* helps you diagnose (and fix) occasional long startup times of QGIS.
  - *Project Load* allows you to get a breakdown of the various stages of project load, in order to identify the causes of slow project load times.
  - *Map Render* tool allows you to identify pain points in your map rendering and track down exactly which layers are causing long map redraws.

Notă: Plugin authors can extend the panel with custom tabs for debugging and developping their own plugins. This is done using `registerDevToolWidgetFactory` method.

Network Logger

The *Network Logger* tab helps you record and display network requests, along with a whole load of useful details like request and reply status, header, errors, SSL configuration errors, timeouts, cache status, etc.

From its top toolbar, you can:

- *Record Log*: which will start or stop the logging.
- *Clear Log*: will clear the log history.
- *Save Log…*: will first show a big warning that the log is sensitive and should be treated as confidential and then allow you to save the log.
- Press the *Settings* drop-down menu to select whether to *Show Successful Requests*, *Show Timeouts* and *Show Replies Served from Cache*.
- *Disable cache*: will disable the cache so that every request has to be performed.
- *Filter requests* based on URL string subsets or request status

By right clicking on a request you can:

- *Open URL* which will open the URL in your default browser.
- *Copy URL*
- *Copy As cURL* to use it in the terminal.
- *Copy as JSON*: copies the tree values as a json string to the clipboard, for easy pasting in bug reports or for remote assistance.

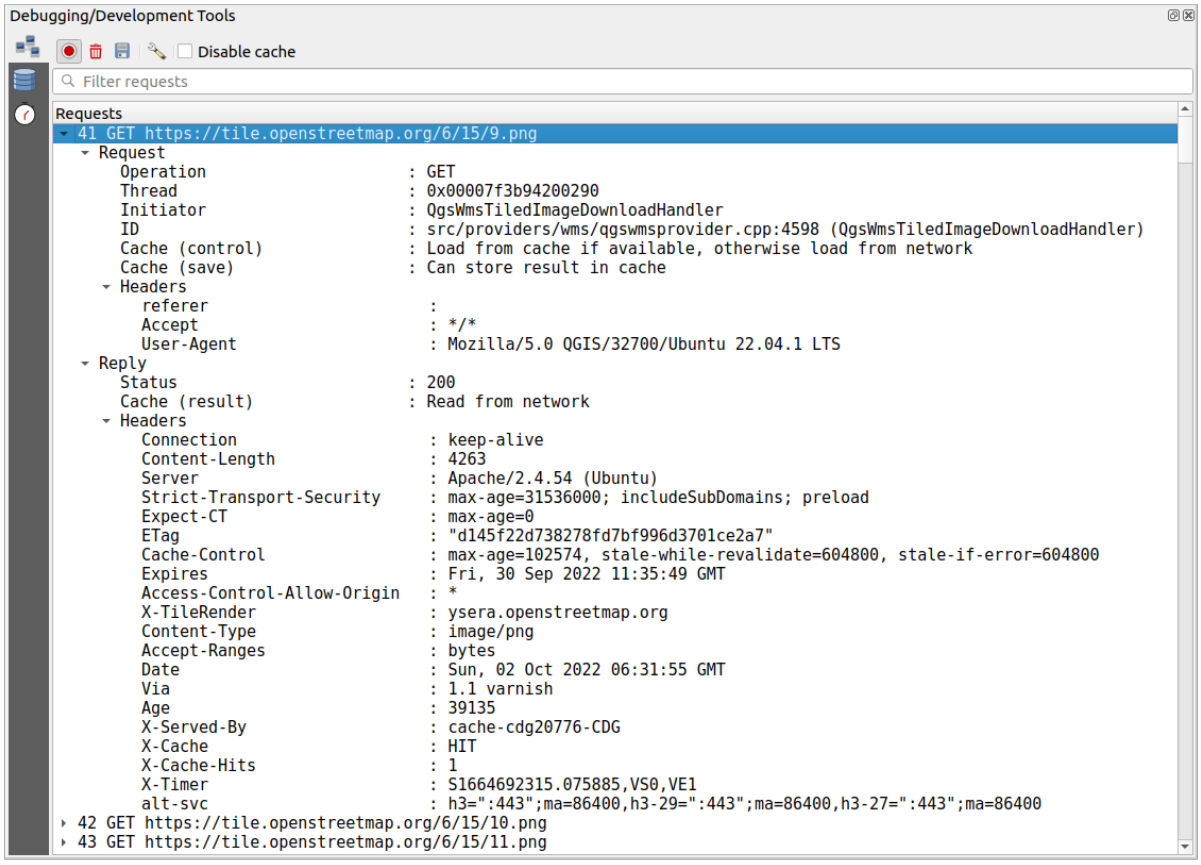


Fig. 12.6: Network Logger output for GET Request

Query Logger

The *Query Logger* is the place to log the SQL commands sent by the data provider and the connections API to the backend database, together with their execution time as measured by QGIS (i.e. in the client that sent the commands). This can be useful when investigating performances of a particular layer during debugging or development of a QGIS algorithm or plugin.

From its top toolbar, you can:

- *Record Log*: which will start or stop the logging.
- *Clear Log*: will clear the log history.
- *Save Log…*: will first show a big warning that the log is sensitive and should be treated as confidential and then allow you to save the log.
- *Filter queries* based on the query string subsets or details such as provider type, start time, initiator, …

Right-click on a reported query, and you can:

- *Copy SQL* command called by QGIS on the database
- *Copy as JSON*: copies the tree values as a json string to the clipboard, for easy pasting in bug reports or for remote assistance.

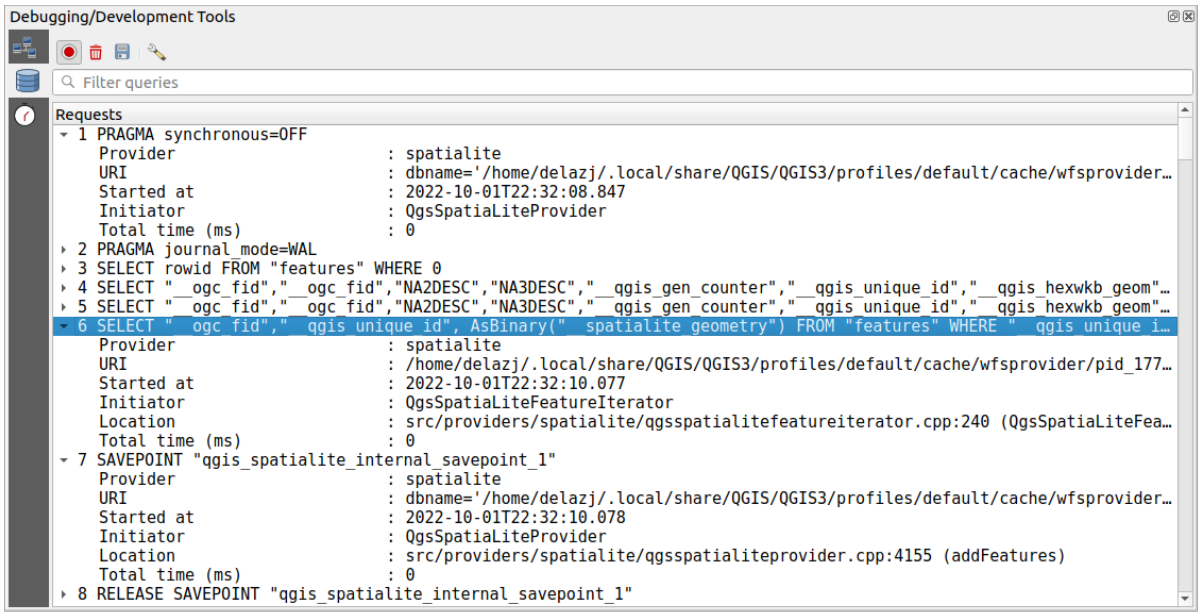


Fig. 12.7: Query Logger output

Profiler

The *Profiler* tab allows to get load times for every single operation involved in actions requested by the user. Depending on the context, these operations can be settings reading, menu, map canvas or 3D views creation, resolving map layers reference, bookmark or layout loading, … This helps identify causes of slow load times.

Default supported actions can be selected from the *Categories* drop-down menu:

- QGIS *Startup*
- *Project Load*

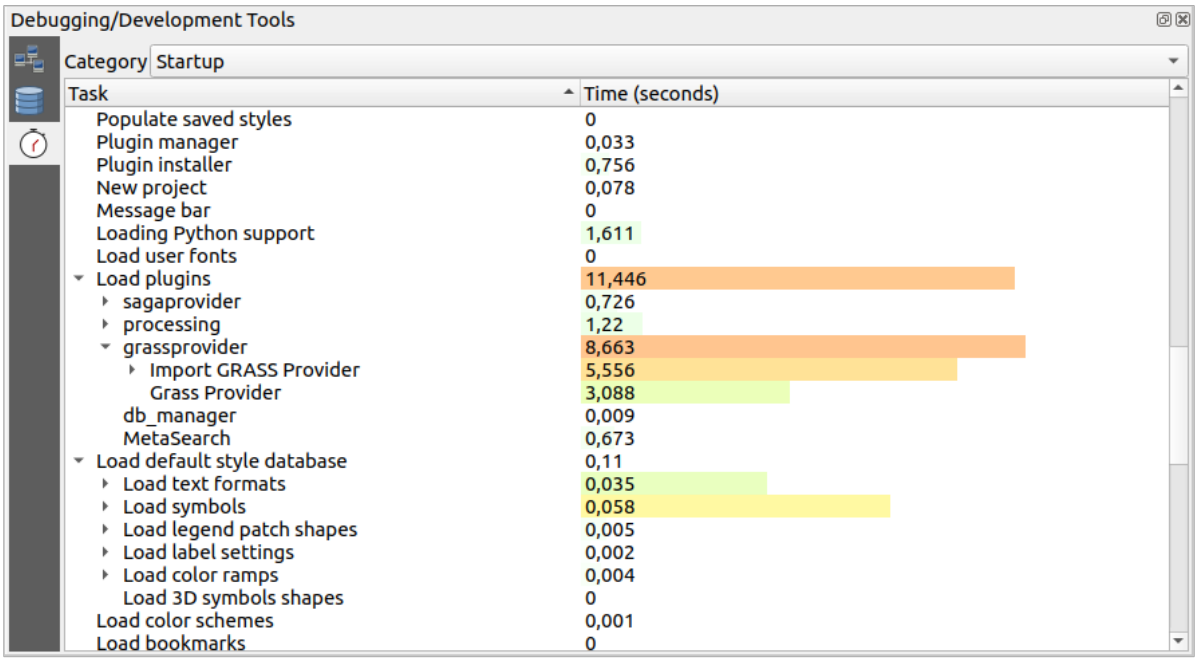


Fig. 12.8: Profiler for QGIS Startup

12.3 Embedding layers from external projects

Sometimes, you'd like to keep some layers in different projects, but with the same style. You can either create a *default style* for these layers or embed them from another project to save time and effort.

Embed layers and groups from an existing project has some advantages over styling:

- All types of layers (vector or raster, local or online…) can be added
- Fetching groups and layers, you can keep the same tree structure of the „background" layers in your different projects
- While the embedded layers are editable, you can't change their properties such as symbology, labels, forms, default values and actions, ensuring consistency across projects
- Modify the items in the original project and changes are propagated to all the other projects

If you want to embed content from other project files into your project, select *Layer* ► *Embed Layers and Groups*:

1. Click the … button to look for a project: you can see the content of the project (see Fig. 12.9)
2. Hold down `Ctrl` (or `Cmd`) and click on the layers and groups you wish to retrieve
3. Clic pe *OK*

The selected layers and groups are embedded in the *Layers* panel and displayed on the map canvas. An icon is added next to their name for recognition and hovering over displays a tooltip with the original project file path.

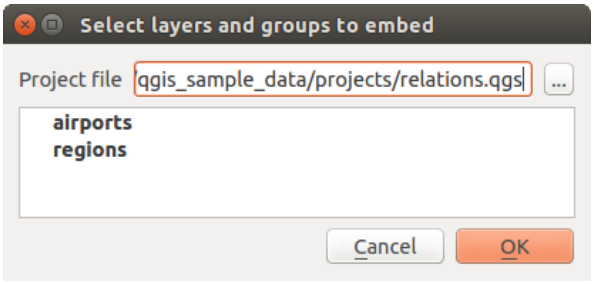


Fig. 12.9: Selectați straturile și grupurile pentru încapsulare

Like any other layer, an embedded layer can be removed from the project by right-clicking on the layer and clicking Remove.

Sfat: Change rendering of an embedded layer

It's not possible to change the rendering of an embedded layer, unless you make the changes in the original project file. However, right-clicking on a layer and selecting *Duplicate* creates a layer which is fully-featured and not dependent on the original project. You can then safely remove the linked layer.

12.4 Interacting with features

12.4.1 Selectare entități

QGIS provides several tools to select features on the map canvas. Selection tools are available in the *Edit* ► *Select* menu or in the *Selection Toolbar*.

Notă: Selection tools work with the currently active layer.

Selecting manually on the map canvas

To select one or more features with the mouse, you can use one of the following tools:

- Select Features by area or single click
- Selectare Entități după un Poligon
- Selectare Entități în Mod Liber
- Selectare Entități după o Rază

Notă: Other than Select Features by Polygon, these manual selection tools allow you to select feature(s) on the map canvas with a single click.

Notă: Use the Select Features by Polygon tool to use an existing polygon feature (from any layer) to select overlapping features in the active layer. Right-click in the polygon and choose it from the context menu that shows a list of all the polygons that contain the clicked point. All the overlapping features from the active layer are selected.

Sfat: Use the *Edit* ► *Select* ► *Reselect Features* tool to redo your latest selection. Very useful when you have painstakingly made a selection, and then click somewhere else accidentally and clear your selection.

În timp ce utilizați *Selectare Entit(ăți)*, ținând apăsat `Shift` sau `Ctrl` se stabilește dacă o entitate este selectată (adică, fie o adaugă la selecția curentă, fie o elimină din respectiva selecție).

For the other tools, different behaviors can be performed by holding down:

- `Shift`: add features to the current selection
- `Ctrl`: substract features from the current selection
- `Ctrl+Shift`: intersect with current selection, ie only keep overlapping features from the current selection
- `Alt`: select features that are totally within the selection shape. Combined with `Shift` or `Ctrl` keys, you can add or substract features to/from the current selection.

Automatic selection

The other selection tools, most of them available from the *Attribute table*, perform a selection based on a feature's attribute or its selection state (note that attribute table and map canvas show the same information, so if you select one feature in the attribute table, it will be selected on the map canvas too):

- Select By Expression… select features using expression dialog
- Select Features By Value… or press `F3`
- Deselect Features from All Layers or press `Ctrl+Alt+A` to deselect all selected features in all layers
- Deselect Features from the Current Active Layer or press `Ctrl+Shift+A`
- Select All Features or press `Ctrl+A` to select all features in the current layer
- Invert Feature Selection to invert the selection in the current layer
- Select by Location to select the features based on their spatial relationship with other features (in the same or another layer - see *Selectare după locație*)
- Select within distance to select features wherever they are within the specified maximum distance from referenced features - see *Select within distance*)

For example, if you want to find regions that are boroughs from `regions.shp` of the QGIS sample data, you can:

1. Use the Select features using an Expression icon
2. Expand the *Fields and Values* group
3. Double-click the field that you want to query („TYPE_2")
4. Click *All Unique* in the panel that shows up on the right
5. From the list, double-click «Borough». In the *Expression* editor field, write the following query:

```
"TYPE_2"  =  'Borough'
```

6. Click *Select Features*

From the expression builder dialog, you can also use *Function list* ► *Recent (Selection)* to make a selection that you have used before. The dialog remembers the last 20 expressions used. See *Expresii* for more information and examples.

Sfat: Save your selection into a new file

Users can save selected features into a **New Temporary Scratch Layer** or a **New Vector Layer** using *Edit* ► *Copy Features* and *Edit* ► *Paste Features as* in the desired format.

Select Features By Value

This selection tool opens the layer's feature form allowing the user to choose which value to look for for each field, whether the search should be case-sensitive, and the operation that should be used. The tool has also autocompletes, automatically filling the search box with existing values.

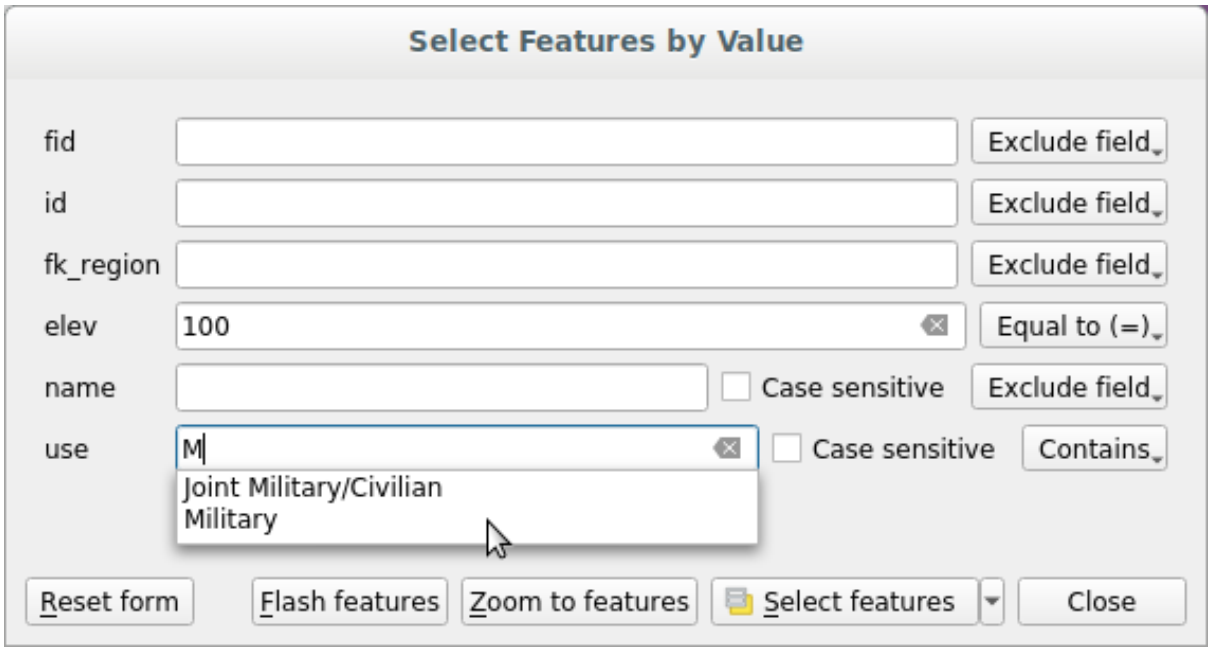


Fig. 12.10: Filter/Select features using form dialog

Alongside each field, there is a drop-down list with options to control the search behaviour:

Tabelul 12.3: Query operators per data type

Field search option	Șirul	Numeric	Data
Exclude Field from the search	☑	☑	☑
Egal cu (=)	☑	☑	☑
Nu este egal cu (≠)	☑	☑	☑
Mai mare decât (>)		☑	☑
Mai mic decât (<)		☑	☑
Mai mare sau egal cu (≥)		☑	☑
Mai mic sau egal cu (≤)		☑	☑
Between (inclusive)		☑	☑
Not between (inclusive)		☑	☑
Contains	☑		
Does not contain	☑		
Is missing (null)	☑	☑	☑
Is not missing (not null)	☑	☑	☑
Starts with	☑		
Ends with	☑		

For string comparisons, it is also possible to use the ☑ *Case sensitive* option.

After setting all search options, click *Select features* to select the matching features. The drop-down options are:

- *Select features*
- *Add to current selection*
- *Remove from current selection*
- *Filter current selection*

You can also clear all search options using the *Reset form* button.

Once the conditions are set, you can also either:

- *Zoom to features* on the map canvas without the need of a preselection
- *Flash features*, highlighting the matching features. This is a handy way to identify a feature without selection or using the Identify tool. Note that the flash does not alter the map canvas extent and would be visible only if the feature is within the bounds of the current map canvas.

12.4.2 Identifying Features

The Identify Features tool allows you to interact with the map canvas and get information on features or pixels in a pop-up window. It can be used to query most of the layer types supported by QGIS (vector, raster, mesh, point cloud, wms, wfs, …). To identify an element, use either:

- *View* ► *Identify Features*
- `Ctrl+Shift+I` (sau `Cmd+Shift+I`),
- Identify Features button on the *Attributes* toolbar

Then click on a feature or pixel of the active layer. The identified item gets highlighted in the map canvas while the *Identify Results* dialog opens with detailed information on it. The dialog also shows a set of buttons for advanced configuration.

The Identify Results dialog

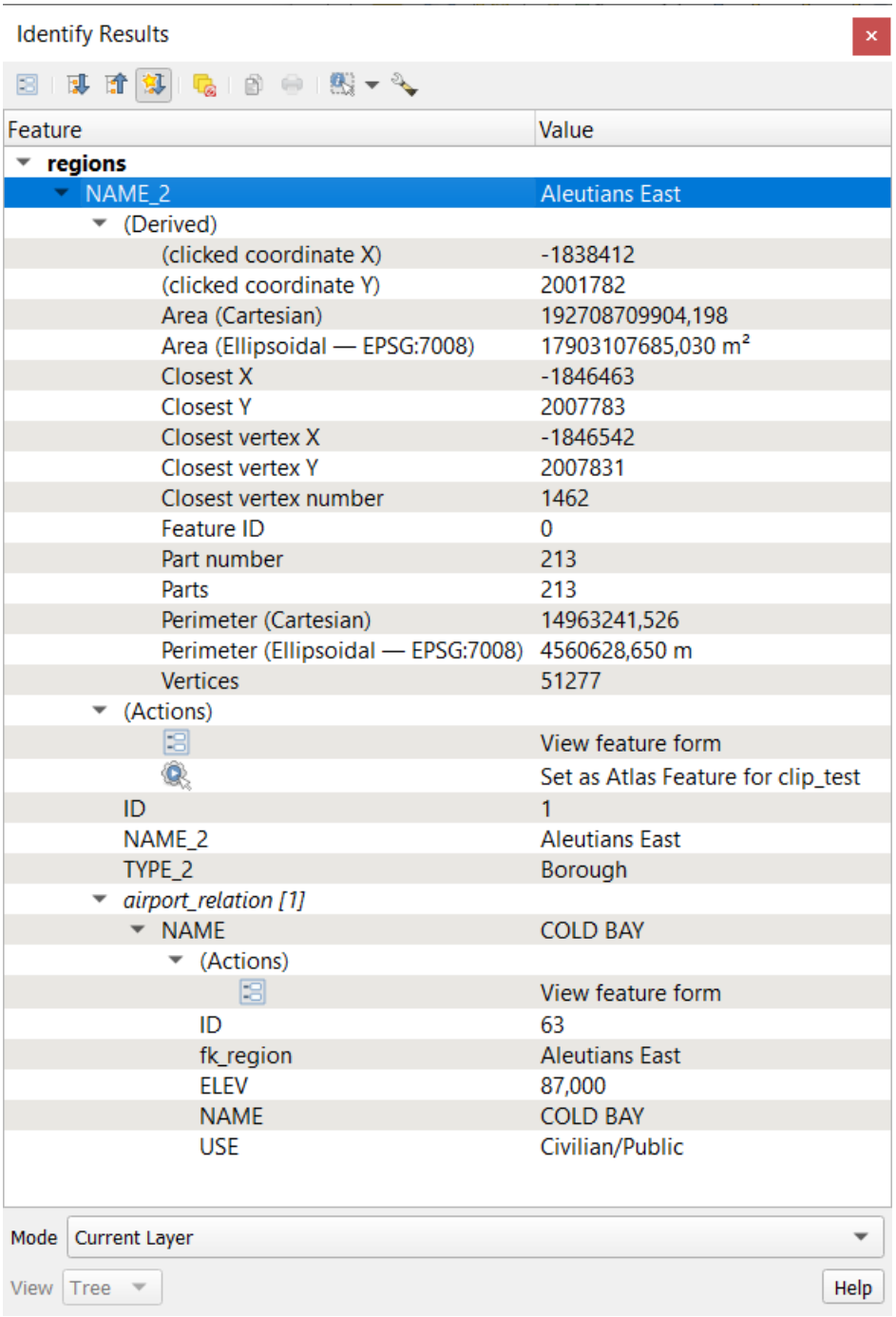


Fig. 12.11: Identify Results dialog

From bottom to top:

- The *View* controls the general aspect of the dialog and the formatting of the results; it can be set as:
  - **Tree**: this is the default view, and returns the results in a tree-structure

  - **Table**: available only for raster-based layers, it allows to display the results as a table whose columns are `Layer`, `FID`, `Attribute` and `Value`
  - or **Graph**: available only for raster-based layers
- The *Mode* helps you select the layers from which results could be returned. These layers should be set visible, displaying data in the map canvas, and set *identifiable* from the *Project properties* ► *Data Sources* ► *Layers capabilities*. Available modes are:
  - **Current layer**: only the layer(s) selected in the *Layers* panel return results. If a group is selected, then results are picked from its leaf layers.
  - **Top down, stop at first**: results are from the layer of the top most feature or pixel under the mouse.
  - **Top down**: results are from the layers with feature or pixel under the mouse.
  - **Layer selection**: opens a contextual menu where the user selects the layer to identify features from. If only a single feature is under the mouse, then the results are automatically displayed.
- In the upper part of the *Identify Results* dialog, a frame shows the *information* returned by features as a table, a graph or a tree, depending on the *selected view*. When in a tree view, you have a handful of tools above the results:
  - Open Form of the current feature
  - Expandare arbore
  - Restrângere arbore
  - Expand New Results by Default to define whether the next identified feature's information should be collapsed or expanded
  - Clear Results
  - Copy the identified feature to clipboard, suitable for pasting in a spreadsheet.
  - Print selected HTML response: a text-based formatting of the results to print on paper or save as a `.PDF` file
  - the *interactive identifying tools*: a drop-down menu with tools for selecting on the map canvas features or pixels to identify
  - Under Identify Settings, you can activate whether to:
    - *Auto open form for single feature results*: If checked, each time a single feature is identified, a form opens showing its attributes. This is a handy way to quickly edit a feature's attributes.
    - *Hide derived attributes from results* to only show fields actually defined in the layer
    - *Hide NULL values from results*
  - Help to access the current documentation

Using the Identify Features tool

In its default display (*View: Tree*), the *Identify Results* panel offers several tools to interact with the layers to query. A smart combination of these tools with the *target layers selector* may greatly improve identification operations:

- Identify Feature(s) by single click or click-and-drag
  - single click or click-and-drag: overlaying features in the target layers are returned
  - right-click: overlaying features from target layers are listed in the contextual menu, grouped by layers. You can then choose to:
    - display the result for a specific feature,

    * display the result for all the features of a specific layer,
    * for vector layers, it is also possible to open its attribute table filtered to the returned features
    * or show all of the returned features.

- Identify Features on Mouse over: move over the map canvas and hovered items in the target layers get highlighted and returned in the results panel.
- Identify Features by Polygon: returns items overlapping a drawn or selected polygon.
    - Draw a polygon (left click to add point, right click to close the polygon) and all the overlaying features from target layers are highlighted and returned in the results panel.
    - Right-click and you get the list of all visible polygon features in the project under the click. Pick an entry and QGIS will return all the features from the target layers that overlap the selected polygon.
- Identify Features by Freehand: returns items overlapping a polygon drawn by freehand. Draw a polygon (left-click to start, move the pointer to shape the area and right-click to close the polygon). All the overlaying features from target layers are highlighted and returned in the results panel.
- Identify Features by Radius returns items overlapping a drawn circle. Draw a cercle (left-click to indicate the center point, move the pointer to shape the area or enter the radius in the pop-up text box and left-click or press `Enter` to validate the circle). All the overlaying features from target layers are highlighted and returned in the results panel.

Sfat: Filter the layers to query with the Identify Features tool

Under *Project* ► *Properties…* ► *Data Sources* ► *Layer Capabilities*, uncheck the *Identifiable* column next to a layer to avoid it being queried when using the Identify Features tool. This is a handy way to return features from only layers that are of interest to you.

Feature information

When you identify a data in the map canvas, the *Identify Results* dialog will list information about the items clicked (or hovered over, depending on the tool in use). The default view is a tree view in which the first item is the name of the layer and its children are its identified feature(s). Each feature is described by the name of a field along with its value. This field is the one set in *Layer Properties* ► *Display*. All the other information about the feature follows.

The feature information displayed by the identify tool will depend on the type of layer you have selected, whether it is a vector layer (including vector tiles or point cloud data) or raster layer. If your layer is raster, clicking on a location on the map canvas with identify tool will highlight the identified raster pixel. The Identify Results dialog can be customized to display custom fields, but by default it will display the following information:

- The feature *display name*;
- **Actions**: Actions can be added to the identify feature windows. The action is run by clicking on the action label. By default, only one action is added, namely `View feature form` for editing. You can define more actions in the layer's properties dialog (see *Proprietățile Acțiunilor*).
- **Derived**: This information is calculated or derived from other information. It includes:
    - general information about the feature's geometry:
        * depending on the geometry type, the cartesian measurements of length, perimeter or area in the layer's CRS units. For 3D line vectors the cartesian line length is available.
        * depending on the geometry type and if an ellipsoid is set in the project properties dialog for *Measurements*, the ellipsoidal values of length, perimeter or area using the specified units
        * the count of geometry parts in the feature and the number of the part clicked

        * the count of vertices in the feature
    - coordinate information, using the project properties *Coordinates display* settings:
        * `X` and `Y` coordinate values of the point clicked
        * the number of the closest vertex to the point clicked
        * `X` and `Y` coordinate values of the closest vertex (and `Z`/`M` if applicable)
        * if you click on a curved segment, the radius of that section is also displayed.
- **Atributele datelor**: Aceasta este lista câmpurilor și valorilor atributelor, pentru entitatea pe care s-a efectuat click.
- information about the related child feature if you defined a *relation*:
    - the name of the relation
    - the entry in reference field, e.g. the name of the related child feature
    - **Actions**: lists actions defined in the layer's properties dialog (see *Proprietățile Acțiunilor*) and the default action is `View feature form`.
    - **Data attributes**: This is the list of attributes fields and values of the related child feature.

Notă: Links in the feature's attributes are clickable from the *Identify Results* panel and will open in your default web browser.

Results contextual menu

Alte funcții pot fi găsite în meniul contextual al elementului identificat. De exemplu, din meniul contextual puteți:

- Vizualiza formularul entității
- Transfocare pe entitate
- Copia entități: Copierea tuturor entităților geometrice și a atributelor
- Toggle feature selection: Add identified feature to selection
- Copia valoarea atributului: Copie doar valoarea atributului pe care faceți clic
- Copiere atribute entitate: Copiază atributele entității
- Select features by attribute value: Select all features in the layer that match the selected attribute
- Șterge rezultatele: Elimină rezultatele din fereastră
- Elimina evidențierea: Anulează evidențierea entităților de pe hartă
- Evidențiază tot
- Evidențiere strat
- Activa stratul: Alegeți un strat pentru a fi activat
- Afișa proprietățile straturilor: Deschide fereastra de proprietăți a unui strat
- Expandează tot
- Restrânge tot

12.5 Salvarea și Partajarea Proprietăților Stratului

12.5.1 Managing Custom Styles

When a vector layer is added to the map canvas, QGIS by default uses a random symbol/color to render its features. However, you can set a default symbol in *Project* ► *Properties…* ► *Default styles* that will be applied to each newly added layer according to its geometry type.

Most of the time, though, you'd rather have a custom and more complex style that can be applied automatically or manually to the layers (with less effort). You can achieve this by using the *Style* menu at the bottom of the Layer Properties dialog. This menu provides you with functions to create, load and manage styles.

A style stores any information set in the layer properties dialog to render or interact with the layer (including symbology, labeling, fields and form definitions, actions, diagrams…) for vector layers, or the pixels (band or color rendering, transparency, pyramids, histogram …) for raster.

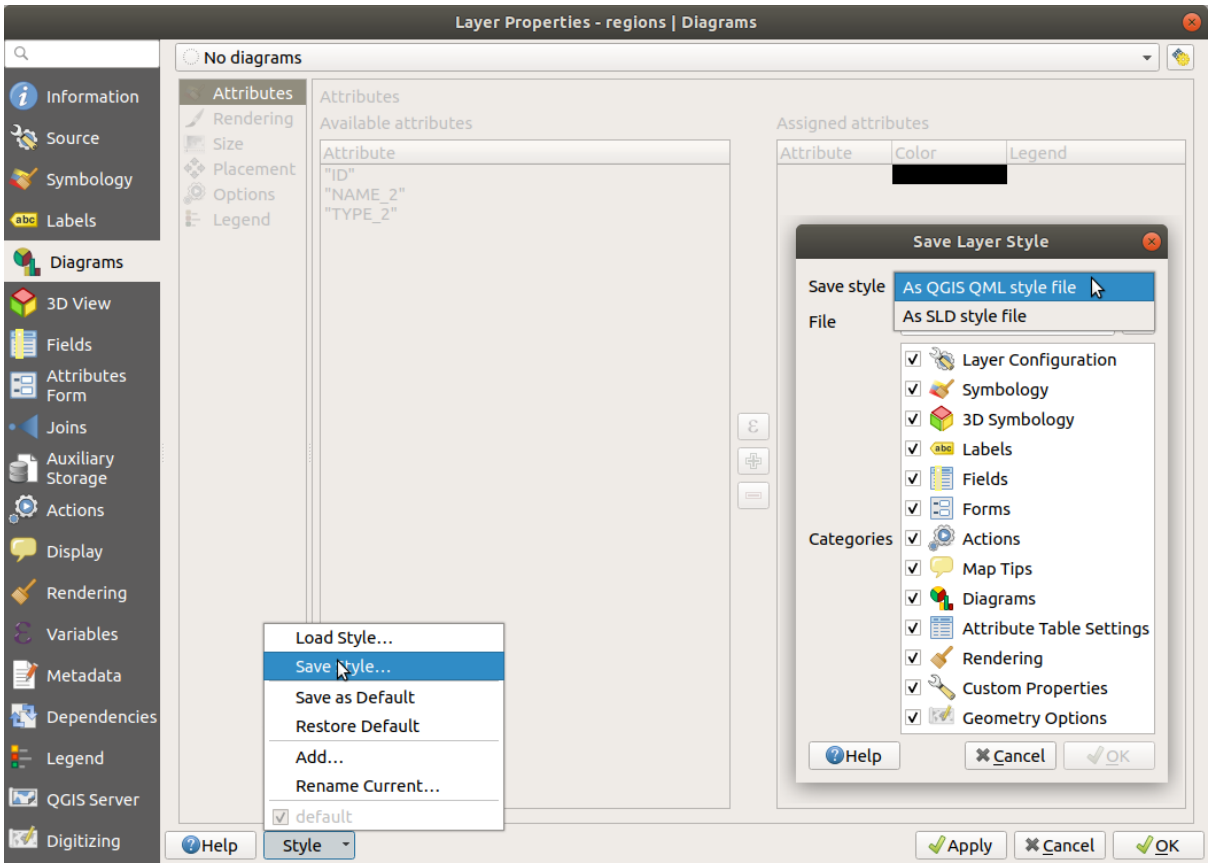


Fig. 12.12: Vector layer style combo box options

By default, the style applied to a loaded layer is named `default`. Once you have got the ideal and appropriate rendering for your layer, you can save it by clicking the *Style* combo box and choosing:

- **Rename Current**: The active style is renamed and updated with the current options
- **Add**: A new style is created using the current options. By default, it will be saved in the QGIS project file. See below to save the style in another file or a database
- **Remove**: Delete unwanted style, in case you have more than one style defined for the layer.

At the bottom of the Style drop-down list, you can see the styles set for the layer with the active one checked.

Note that each time you validate the layer properties dialog, the active style is updated with the changes you've made.

You can create as many styles as you wish for a layer but only one can be active at a time. In combination with *Map Themes*, this offers a quick and powerful way to manage complex projects without the need to duplicate any layer in the map legend.

Notă: Given that whenever you apply modifications to the layer properties, changes are stored in the active style, always ensure you are editing the right style to avoid mistakenly altering a style used in a *map theme*.

Sfat: Manage styles from layer context menu

Right-click on the layer in the *Layers* panel to copy, paste, add or rename layer styles.

12.5.2 Storing Styles in a File or a Database

While styles created from the *Style* combo box are by default saved inside the project and can be copied and pasted from layer to layer in the project, it's also possible to save them outside the project so that they can be loaded in another project.

Save as text file

Clicking the *Style* ► *Save Style*, you can save the style as a:

- QGIS layer style file (`.qml`)
- SLD file (`.sld`), only available for vector layers

Used on file-based format layers (`.shp`, `.tab`...), *Save as Default* generates a `.qml` file for the layer (with the same name). SLDs can be exported from any type of renderer – single symbol, categorized, graduated or rule-based – but when importing an SLD, either a single symbol or rule-based renderer is created. This means that categorized or graduated styles are converted to rule-based. If you want to preserve those renderers, you have to use the QML format. On the other hand, it can be very handy sometimes to have this easy way of converting styles to rule-based.

Save in database

Vector layer styles can also be stored in a database if the layer datasource is a database provider. Supported formats are PostGIS, GeoPackage, SpatiaLite, MS SQL Server and Oracle. The layer style is saved inside a table (named `layer_styles`) in the database. Click on *Save Style...* ► *Save in database* then fill in the dialog to define a style name, add a description, a `.ui` file if applicable and to check if the style should be the default style.

You can save several styles for a single table in the database. However, each table can have only one default style. Default styles can be saved in the layer database or in `qgis.db`, a local SQLite database in the active *user profile* directory.

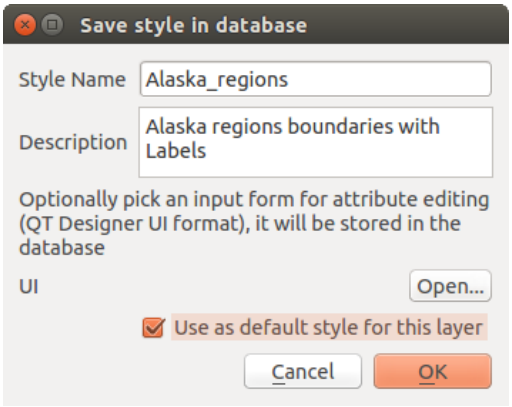


Fig. 12.13: Save Style in database Dialog

Sfat: Sharing style files between databases

You can only save your style in a database if the layer comes from such a database. You can't mix databases (layer in Oracle and style in MS SQL Server for instance). Use instead a plain text file if you want the style to be shared among databases.

Notă: You may encounter issues restoring the `layer_styles` table from a PostgreSQL database backup. Follow *QGIS layer_style table and database backup* to fix that.

Load style

When loading a layer in QGIS, if a default style already exists for this layer, QGIS loads the layer with this style. Also *Style* ► *Restore Default* looks for and loads that file, replacing the layer's current style.

Style ► *Load Style* helps you apply any saved style to a layer. While text-file styles (`.sld` or `.qml`) can be applied to any layer whatever its format, loading styles stored in a database is only possible if the layer is from the same database or the style is stored in the QGIS local database.

The *Database Styles Manager* dialog displays a list of styles related to the layer found in the database and all the other styles saved in it, with name and description.

Sfat: Quickly share a layer style within the project

You can also share layer styles within a project without importing a file or database style: right-click on the layer in the *Layers Panel* and, from the *Styles* combo box , copy the style of a layer and paste it to a group or a selection of layers: the style is applied to all the layers that are of the same type (vector vs raster) as the original layer and, in the case of vector layers, have the same geometry type (point, line or polygon).

12.5.3 Layer definition file

Layer definitions can be saved as a `Layer Definition File (.qlr)` using *Export* ► *Save As Layer Definition File…* in the active layers» context menu. A layer definition file (`.qlr`) includes references to the data source of the layers and their styles. `.qlr` files are shown in the Browser Panel and can be used to add the layers (with the saved style) to the Layers Panel. You can also drag and drop `.qlr` files from the system file manager into the map canvas.

12.6 Documenting your data

In addition to displaying and symbolizing the data in the layers, QGIS allows you to fill:

- **metadata**: information to help people find and understand the dataset, how they can access and use it… these are properties of the datasource and can live out of the QGIS project.
- **notes**: instructions and comments regarding the layer in the current project

12.6.1 Metadata

In the layer properties dialog, the *Metadata* tab provides you with options to create and edit a metadata report on your layer.

Information to fill concern:

- the data *Identification*: basic attribution of the dataset (parent, identifier, title, abstract, language…);
- the *Categories* the data belongs to. Alongside the **ISO** categories, you can add custom ones;
- the *Keywords* to retrieve the data and associated concepts following a standard based vocabulary;
- the *Access* to the dataset (licenses, rights, fees, and constraints);
- the *Extent* of the dataset, either spatial one (CRS, map extent, altitudes) or temporal;
- the *Contact* of the owner(s) of the dataset;
- the *Links* to ancillary resources and related information;
- the *History* of the dataset.

A summary of the filled information is provided in the *Validation* tab and helps you identify potential issues related to the form. You can then either fix them or ignore them.

Metadata are saved in the project file by default, the *Metadata* drop-down offers options for loading/saving metadata from `.qmd` file and for loading/saving metadata in the „Default” location.

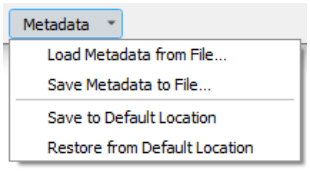


Fig. 12.14: Metadata load/save options

The „Default” location used by *Save to Default Location* and *Restore from Default Location* changes depending on the underlying data source and on its configuration:

- For PostgreSQL data sources if the configuration option *Allow saving/loading QGIS layer metadata in the database* is checked the metadata are stored inside a dedicated table in the database.

- For GeoPackage data sources *Save to Default Location* always saves the metadata in the internal metadata tables of the GeoPackage.

  When metadata are saved into the internal tables of PostgreSQL or GeoPackage they become available for search and filtering in the browser and in the *layer metadata search panel*.

- For all other file based data sources *Save to Default Location* saves the metadata in a `.qmd` file alongside the file.
- In all other cases *Save to Default Location* saves the metadata in a local `.sqlite` database.

12.6.2 Notițele stratului

Layer notes allow you to document the layer within the current project. They can be place to store important messages for users of the project like to do lists, instructions, warnings, …

From the layer's contextual menu in *Layers* panel, select *Add layer notes…* and fill the open dialog with necessary texts.

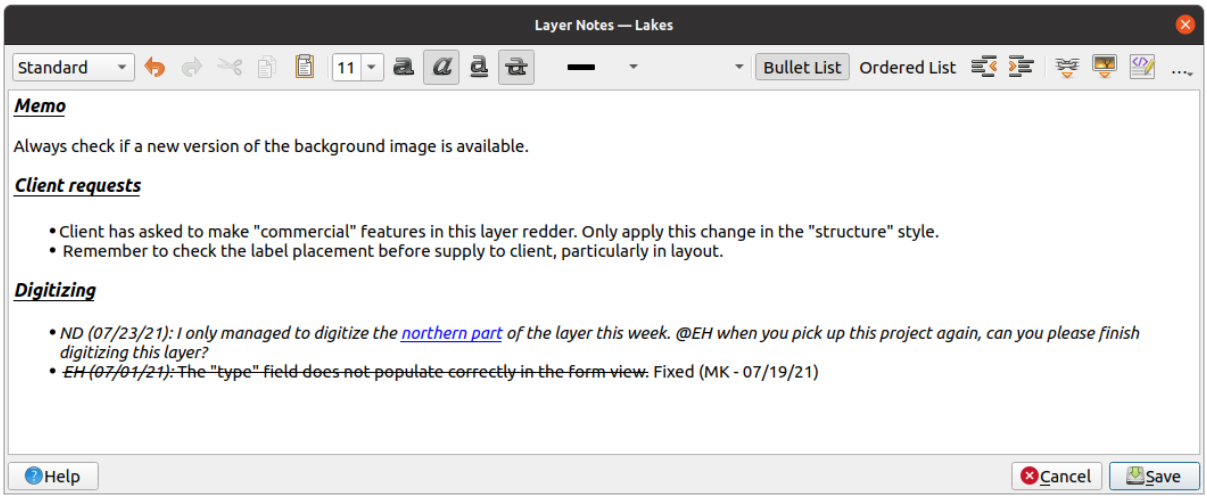


Fig. 12.15: Adding notes to a layer

The *Add layer notes* dialog provides a html-based multiline text box with a complete set of tools for:

- text manipulation: cut, copy, paste, undo, redo
- characters formatting, applied to all or parts of the contents: font size and color, bold, italic, underline, strikethrough, background color, URL highlighting
- paragraph structuring: bullet and numbered lists, indentation, predefined headings
- file insertion, even with drag-and-drop
- editing with HTML coding

From the … drop-down at the far right of the toolbar, you can:

- *Remove all formatting*
- *Remove character formatting*
- *Clear all content*

In the *Layers* panel, a layer with a note is assigned the icon which, upon hover, displays the note. Click the icon to edit the note. You can as well right-click the layer and *Edit layer note…* or *Remove layer note*.

Notă: Notes are part of the *layer style* and can be saved in the `.qml` or `.qlr` file. They can also be transferred from one layer to another while copy-pasting the layer style.

12.7 Storing values in Variables

In QGIS, you can use variables to store useful recurrent values (e.g. the project's title, or the user's full name) that can be used in expressions. Variables can be defined at the application's global level, project level, layer level, processing modeler level, layout level, and layout item's level. Just like CSS cascading rules, variables can be overwritten - e.g., a project level variable will overwrite any application global level variables set with the same name. You can use these variables to build text strings or other custom expressions using the `@` character before the variable name. For example in print layout creating a label with this content:

```
This map was made using QGIS [% @qgis_version %]. The project file for this
map is: [% @project_path %]
```

Va produce o etichetă care arată în felul următor:

```
This map was made using QGIS 3.4.4-Madeira. The project file for this map is:
/gis/qgis-user-conference-2019.qgs
```

Besides the *preset read-only variables*, you can define your own custom variables for any of the levels mentioned above. You can manage:

- **global variables** from the *Settings* ► *Options* menu
- **project variables** from the *Project Properties* dialog (see *Proprietăţi Proiect*)
- **vector layer variables** from the *Layer Properties* dialog (see *Dialogul Proprietăților Vectoriale*);
- **modeler variables** from the *Model Designer* dialog (see *The model designer*);
- **layout variables** from the *Layout* panel in the Print layout (see *The Layout Panel*);
- and **layout item variables** from the *Item Properties* panel in the Print layout (see *Layout Items Common Options*).

To differentiate from editable variables, read-only variable names and values are displayed in italic. On the other hand, higher level variables overwritten by lower level ones are strike through.

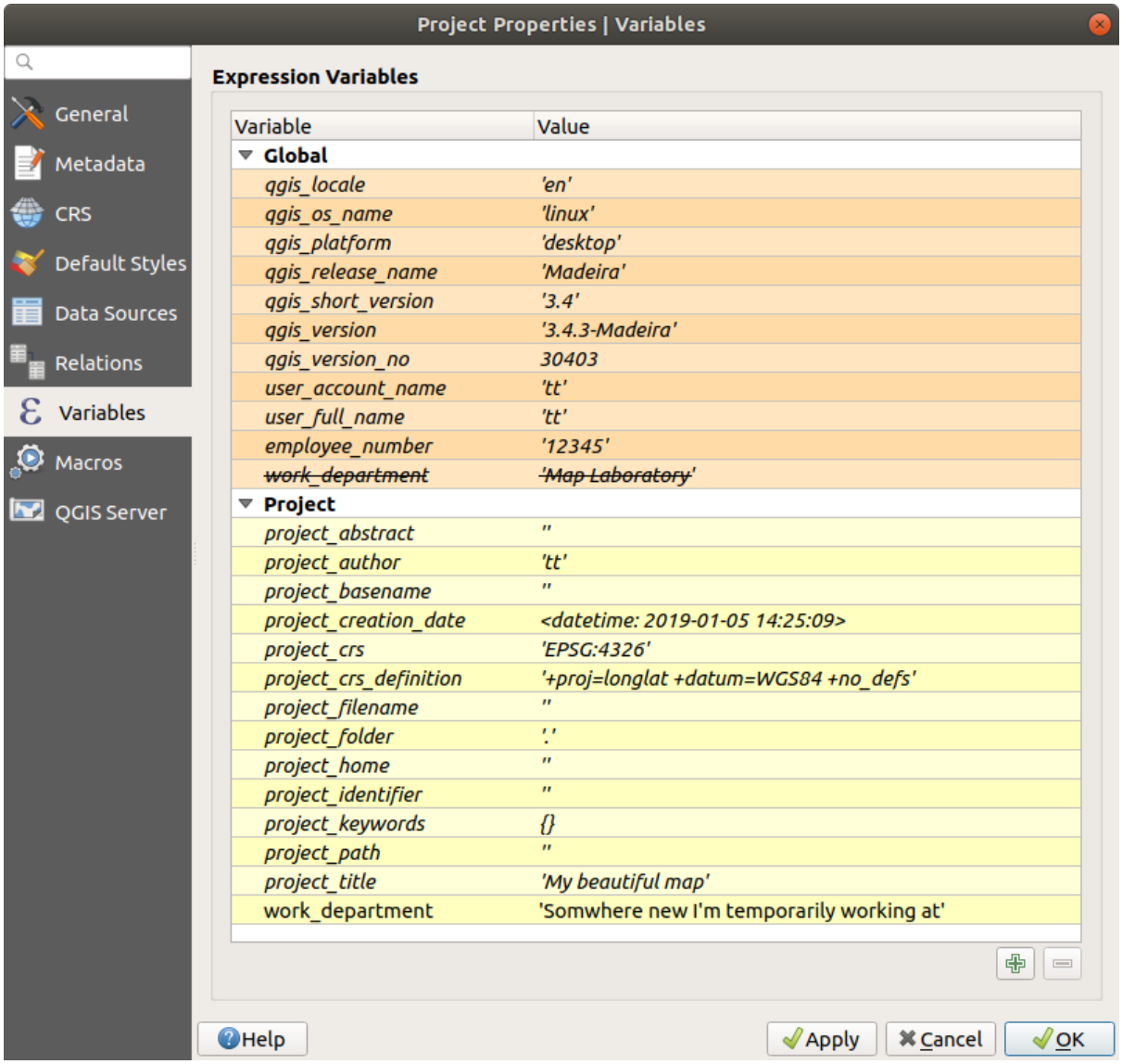


Fig. 12.16: Variables editor at the project level

Notă: You can read more about variables and find some examples in Nyall Dawson's Exploring variables in QGIS 2.12, part 1, part 2 and part 3 blog posts.

12.8 Autentificarea

QGIS has the facility to store/retrieve authentication credentials in a secure manner. Users can securely save credentials into authentication configurations, which are stored in a portable database, can be applied to server or database connections, and are safely referenced by their ID tokens in project or settings files. For more information see *Sistem de Autentificare*.

A master password needs to be set up when initializing the authentication system and its portable database.

12.9 Common widgets

In QGIS, there are some options you'll often have to work with. For convenience, QGIS provides you with special widgets that are presented below.

12.9.1 Selectorul de Culoare

The color dialog

The *Select Color* dialog will appear whenever you click the icon to choose a color. The features of this dialog depend on the state of the *Use native color chooser dialogs* parameter checkbox in *Settings* ► *Options…* ► *General*. When checked, the color dialog used is the native one of the OS on which QGIS is running. Otherwise, the QGIS custom color chooser is used.

The custom color chooser dialog has four different tabs which allow you to select colors by Color ramp, Color wheel, Color swatches or Color picker. With the first two tabs, you can browse to all possible color combinations and apply your choice to the item.

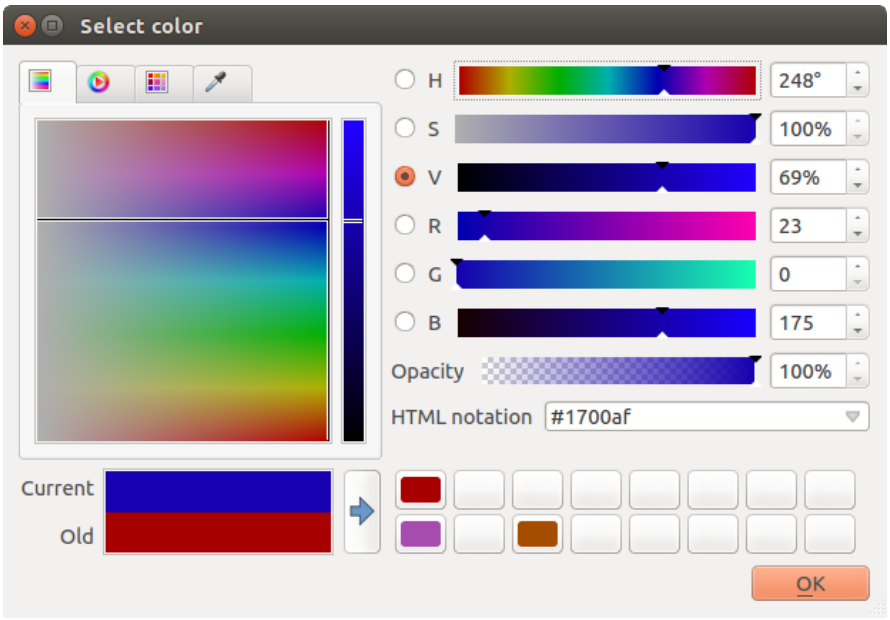


Fig. 12.17: Fila de selectare a gamei de culori

In the Color swatches tab, you can choose from a list of color palettes (see *Colors settings* for details). All but the *Recent colors* palette can be modified with the Add current color and Remove selected color buttons at the bottom of the frame.

The … button next to the palette combo box also offers several options to:

- copy, paste, import or export colors
- create, import or remove color palettes
- add the custom palette to the color selector widget with the *Show in Color Buttons* item (see Fig. 12.19)

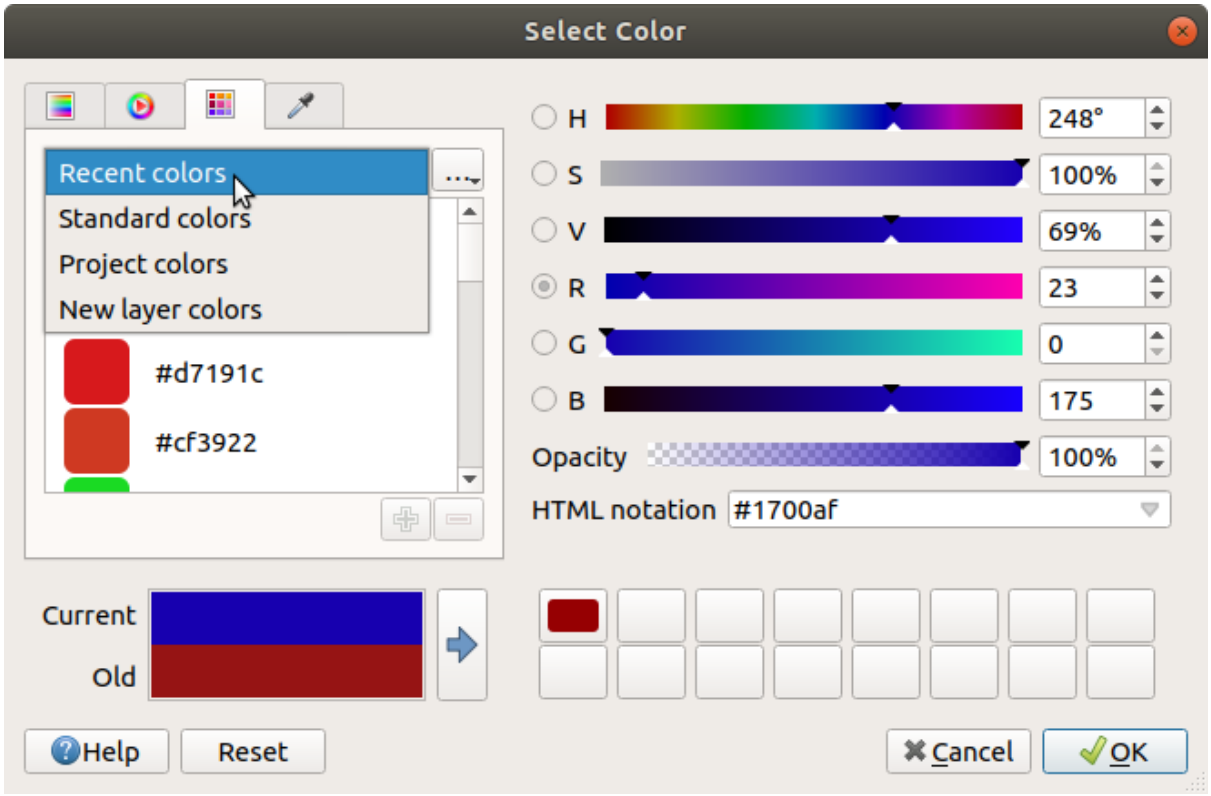


Fig. 12.18: Color selector swatches tab

Another option is to use the Color picker which allows you to sample a color from under your mouse cursor at any part of the QGIS UI or even from another application: press the space bar while the tab is active, move the mouse over the desired color and click on it or press the space bar again. You can also click the *Sample Color* button to activate the picker.

Whatever method you use, the selected color is always described through color sliders for `HSV` (Hue, Saturation, Value) and `RGB` (Red, Green, Blue) values. The color is also identifiable in *HTML notation*.

Modifying a color is as simple as clicking on the color wheel or ramp or on any of the color parameters sliders. You can adjust such parameters with the spinbox beside or by scrolling the mouse wheel over the corresponding slider. You can also type the color in HTML notation. Finally, there is an *Opacity* slider to set transparency level.

The dialog also provides a visual comparison between the *Old* color (applied to object) and the *Current* one (being selected). Using drag-and-drop or pressing the Add color to swatch button, any of these colors can be saved in a slot for easy access.

Sfat: Quick color modification

Drag-and-drop a color selector widget onto another one to apply its color.

The color drop-down shortcut

Click the drop-down arrow to the right of the color button to display a widget for quick color selection. This shortcut provides access to:

- a color wheel to pick a color from
- an alpha slider to change color opacity
- the color palettes previously set to *Show in Color Buttons*
- copy the current color and paste it into another widget
- pick a color from anywhere on your computer display
- choose a color from the color selector dialog
- drag-and-drop the color from one widget to another for quick modification

---

Sfat: Scroll the mouse wheel over a color selector widget to quickly modify the opacity of the associated color.

Notă: When the color widget is set to a *project color* through the data-defined override properties, the above functions for changing the color are unavailable. You'd first need to *Unlink color* or *Clear* the definition.

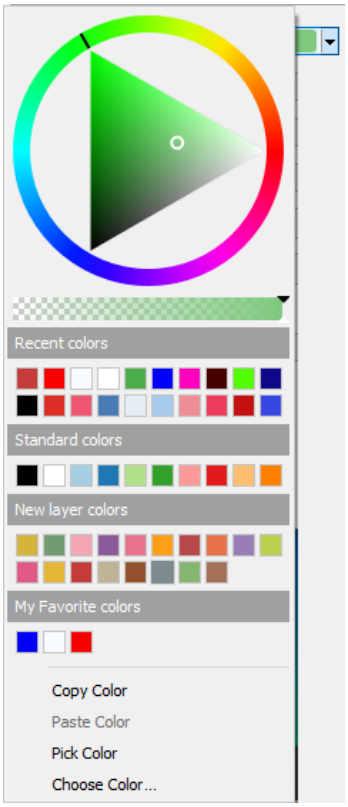


Fig. 12.19: Meniul rapid al selectorului de culoare

The color ramp drop-down shortcut

Color ramps are a practical way to apply a set of colors to one or many features. Their creation is described in the *Setting a Color Ramp* section. As for the colors, pressing the color ramp button opens the corresponding color ramp type dialog allowing you to change its properties.

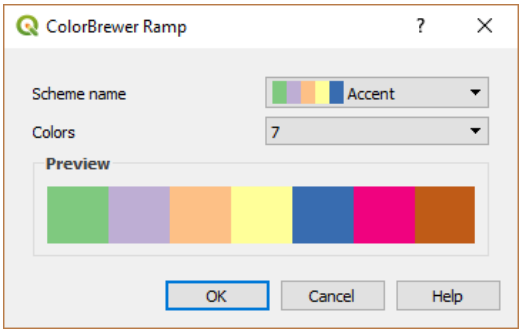


Fig. 12.20: Customizing a colorbrewer ramp

The drop-down menu to the right of the button gives quick access to a wider set of color ramps and options:

- *Invert Color Ramp*
- *Clear Current Ramp* to unset any assigned color ramp to the widget (available only in some contexts)
- *Random Colors*: available only in some contexts (e.g., when a color ramp is being used for a layer symbology), checking this entry creates and applies a color ramp with random colors. It also enables a *Shuffle random colors* entry to regenerate a new random color ramp if the current one is not satisfactory.
- a preview of the `gradient` or `catalog: cpt-city` color ramps flagged as **Favorites** in the *Style Manager* dialog
- *All Color Ramps* to access the compatible color ramps database
- *Create New Color Ramp…* of any supported type that could be used in the current widget (note that this color ramp will not be available elsewhere unless you save it in the library)
- *Edit Color Ramp…*, the same as clicking the whole color ramp button
- *Save Color Ramp…*, to save the current color ramp with its customizations in the style library

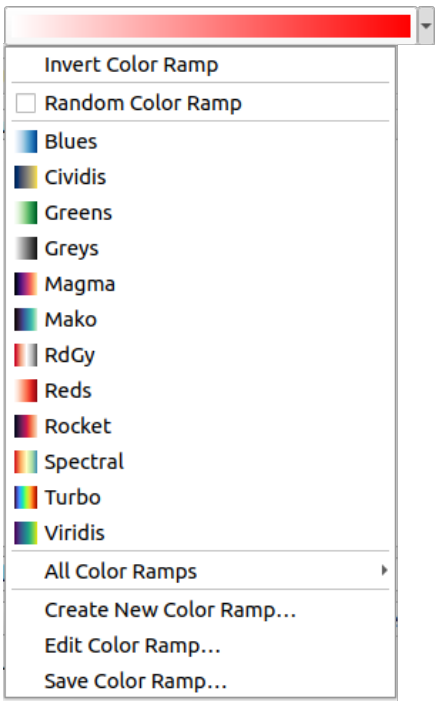


Fig. 12.21: Quick color ramp selection widget

12.9.2 Symbol Widget

The *Symbol* selector widget is a convenient shortcut when you want to set symbol properties of a feature. Clicking the drop-down arrow shows the following symbol options, together with the features of the *color drop-down widget*:

- *Configure Symbol...*: the same as pressing the symbol selector widget. It opens a dialog to set the *symbol parameters*.
- *Copy Symbol* from the current item
- *Paste Symbol* to the current item, speeding configuration
- *Clear Current Symbol* to unset any assigned symbol to the widget (available only in some contexts)

Sfat: Scroll the mouse wheel over a marker or line symbol widget to quickly modify the size of the associated symbol.

12.9.3 Remote or embedded file selector

Along with the file selector widget, the ... button will sometimes show a drop-down arrow. This is usually available when using:

- an SVG file in a symbol or a label
- a raster image to customize symbols, labels, textures or decorations

Pressing the arrow will provide you with a menu to:

- *Select File...*, to load the file from the file system. The file is identified through the file path and QGIS needs to resolve the path in order to display the corresponding image
- *From URL...*, to load the file from a remote URL. As above, the image will only be loaded on successful retrieval of the remote resource

- *Embed File…*, to embed the file inside the current project, style database, or print layout template. The file is then always rendered as part of the item. This is a convenient way to create self-contained projects with custom symbols which can be easily shared amongst different users and installations of QGIS.
- *Extract Embedded File…*, to extract the embedded file from the widget and save it on disk.

12.9.4 Visibility Scale Selector

The visibility scale selector provides options to control the scales at which an element will be made visible in the map canvas. Out of the specified range of scales, the elements are not displayed. It can be applied e.g. to layers, labels or diagrams, from their *Rendering* properties tab.

1. Tick the *Scale dependent visibility* box
2. Fill the *Minimum (exclusive)* box with the most zoomed out desired scale, typing the value or selecting it from the *predefined scales*
3. and/or fill the *Maximum (inclusive)* box with the most zoomed in desired scale

   The Set to current canvas scale button next to the scale boxes sets the current map canvas scale as boundary of the range visibility. Press the arrow next to the button to access scales from layouts» maps and reuse them to fill the box.

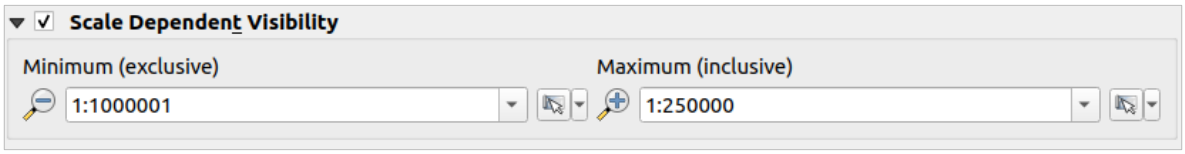


Fig. 12.22: Visibility scale selector widget

12.9.5 Spatial Extent Selector

The *Extent* selector widget is a convenient shortcut when you want to select a spatial extent to assign to a layer or to limit the actions to run on. Depending on the context, it offers selection between:

- *Current Layer Extent*: e.g. when exporting a layer
- *Calculate from Layer* ►: uses extent of a layer loaded in the current project
- Use current *Map Canvas Extent*
- *Draw on Canvas*: a rectangle whose coordinates are then used
- *Calculate from Bookmark*: uses extent of a saved *bookmark*
- *Calculate from Layout Map*: uses extent of a *layout map*
- Enter or edit the coordinates as `xmin, xmax, ymin, ymax`

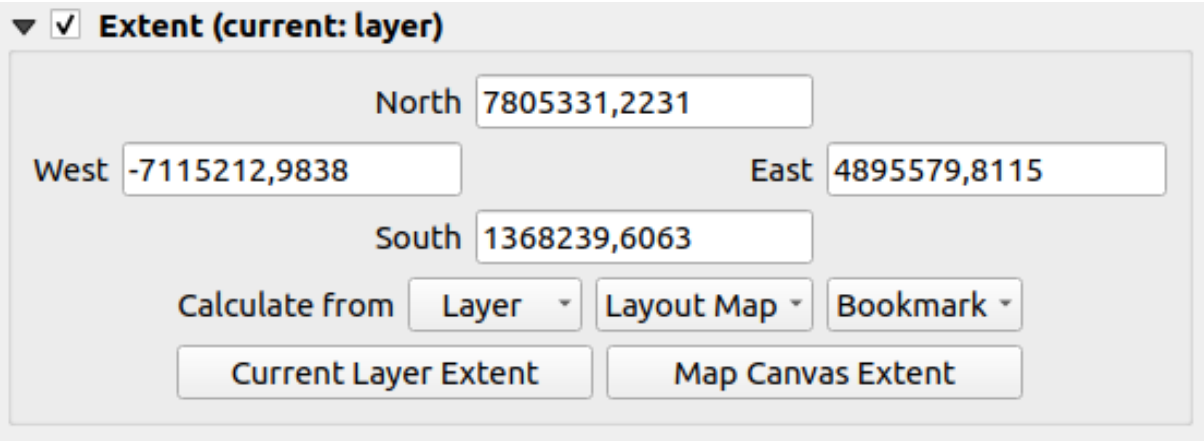


Fig. 12.23: Extent selector widget

12.9.6 Font Selector

The *Font* selector widget is a convenient shortcut when you want to set font properties for textual information (feature labels, decoration labels, map legend text, …). Clicking the drop-down arrow shows some or all of the following options:

Fig. 12.24: Font selector drop-down menu

- *Clear Current Text Format* to unset any assigned text format to the widget (available only in some contexts)
- *Font Size* in the associated unit
- *Recent Fonts* ► menu with the active font checked (at the top)
- *Configure Format...*: same as pressing the font selector widget. It opens the *Text format* dialog, providing advanced formatting options such as color, opacity, orientation, HTML notation, buffer, background, shadow, …

- *Copy Format* of the text
- *Paste Format* to the text, speeding configuration
- the *color widget* for quick color setting

Sfat: Scroll the mouse wheel over a font selector widget to quickly modify the font size of the associated text.

12.9.7 Unit Selector

Size properties of the items (labels, symbols, layout elements, …) in QGIS are not necessarily bound to either the project units or the units of a particular layer. For a large set of properties, the *Unit* selector drop-down menu allows you to tweak their values according to the rendering you want (based on screen resolution, paper size, or the terrain). Available units are:

- *Millimeters*
- *Points*
- *Pixels*
- *Inches*
- *Percentage*: allows you to set some properties as a percent of another one. For example, this is useful for creation of text formats where the components (buffer size, shadow radius…) nicely scale as the text size is changed, instead of having constant buffer/shadow sizes. So you don't need to adjust those sizes, when the text size changes.
- *Meters at Scale*: This allows you to always set the size in meters, regardless of what the underlying map units are (e.g. they can be in inches, feet, geographic degrees, …). The size in meters is calculated based on the current project ellipsoid setting and a projection of the distances in meters at the center of the current map extent. For maps in a projected coordinate system this is calculated using projected units. For maps in a geographic (latitude/longitude) based system the size is approximated by calculating meter sizes using ellipsoidal calculations for the vertical scale of the map.
- and *Map Units*: The size is scaled according to the map view scale. Because this can lead to too big or too small values, use the button next to the entry to constrain the size to a range of values based on:
  - The *Minimum scale* and the *Maximum scale*: The value is scaled based on the map view scale until you reach any of these scale limits. Out of the range of scale, the value at the nearest scale limit is kept.
  - and/or The *Minimum size* and the *Maximum size* in `mm`: The value is scaled based on the map view scale until it reaches any of these limits; Then the limit size is kept.

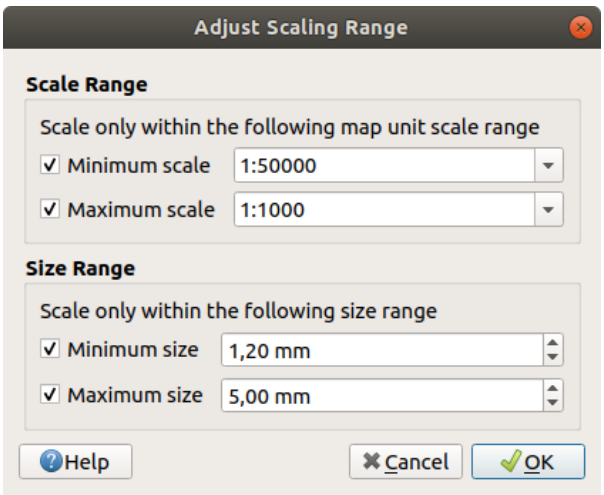


Fig. 12.25: Adjust scaling range dialog

12.9.8 Number Formatting

Numeric formatters allow formatting of numeric values for display, using a variety of different formatting techniques (for instance scientific notation, currency values, percentage values, etc). One use of this is to set text in a layout scale bar or fixed table.

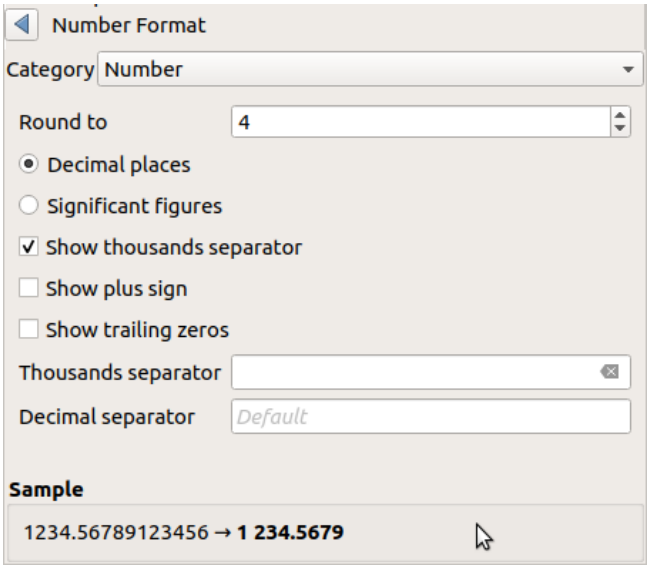


Fig. 12.26: Formatting numeric value

Different categories of formats are supported. For most of them, you can set part or all of the following numeric options:

- *Show thousands separator*
- *Show plus sign*
- *Show trailing zeros*

But they can also have their custom settings. Provided categories are:

- *General*, the default category: has no setting and displays values as set in the parent widget properties or using the global settings.
- *Number*
  - The value can be *Round to* a self defined number of *Decimal places* or their *Significant figures*
  - customize the *Thousands separator* and *Decimal separator*
- *Bearing* for a text representation of a direction/bearing using:
  - *Format*: possible ranges of values are `0 to 180°, with E/W suffix`, `-180 to +180°` and `0 to 360°`
  - number of *Decimal places*
- *Currency* for a text representation of a currency value.
  - *Prefix*
  - *Suffix*
  - number of *Decimal places*
- *Fraction* for a vulgar fractional representation of a decimal value (e.g. *1/2* instead of *0.5*)
  - *Use unicode super/subscript* to show. For example $^{1/2}$ instead of 1/2

  - ☐ *Use dedicated Unicode characters*
  - customize the *Thousands separator*
- *Percentage* - appends `%` to the values, with setting of:
  - number of *Decimal places*
  - *Scaling* to indicate whether the actual values already represent percentages (then they will be kept as is) or fractions (then they are converted)
- *Scientific* notation in the form `2.56e+03`. The number of *Decimal places* can be set.

A live preview of the settings is displayed under the *Sample* section.

12.9.9 Modurile de Fuziune

QGIS offers different options for special rendering effects with these tools that you may previously only know from graphics programs. Blending modes can be applied on layers and features, and also on print layout items:

- *Normal*: This is the standard blend mode, which uses the alpha channel of the top pixel to blend with the pixel beneath it. The colors aren't mixed.
- *Lighten*: This selects the maximum of each component from the foreground and background pixels. Be aware that the results tend to be jagged and harsh.
- *Screen*: Light pixels from the source are painted over the destination, while dark pixels are not. This mode is most useful for mixing the texture of one item with another item (such as using a hillshade to texture another layer).
- *Dodge*: Brighten and saturate underlying pixels based on the lightness of the top pixel. Brighter top pixels cause the saturation and brightness of the underlying pixels to increase. This works best if the top pixels aren't too bright. Otherwise the effect is too extreme.
- *Addition*: Adds pixel values of one item to the other. In case of values above the maximum value (in the case of RGB), white is displayed. This mode is suitable for highlighting features.
- *Darken*: Retains the lowest values of each component of the foreground and background pixels. Like lighten, the results tend to be jagged and harsh.
- *Multiply*: Pixel values of the top item are multiplied with the corresponding values for the bottom item. The results are darker.
- *Burn*: Darker colors in the top item cause the underlying items to darken. Burn can be used to tweak and colorize underlying layers.
- *Overlay*: Combines multiply and screen blending modes. Light parts become lighter and dark parts become darker.
- *Soft light*: Very similar to overlay, but instead of using multiply/screen it uses color burn/dodge. This is supposed to emulate shining a soft light onto an image.
- *Hard light*: Hard light is also very similar to the overlay mode. It's supposed to emulate projecting a very intense light onto an image.
- *Difference*: Subtracts the top pixel from the bottom pixel, or the other way around, in order always to get a positive value. Blending with black produces no change, as the difference with all colors is zero.
- *Subtract*: Subtracts pixel values of one item from the other. In the case of negative values, black is displayed.

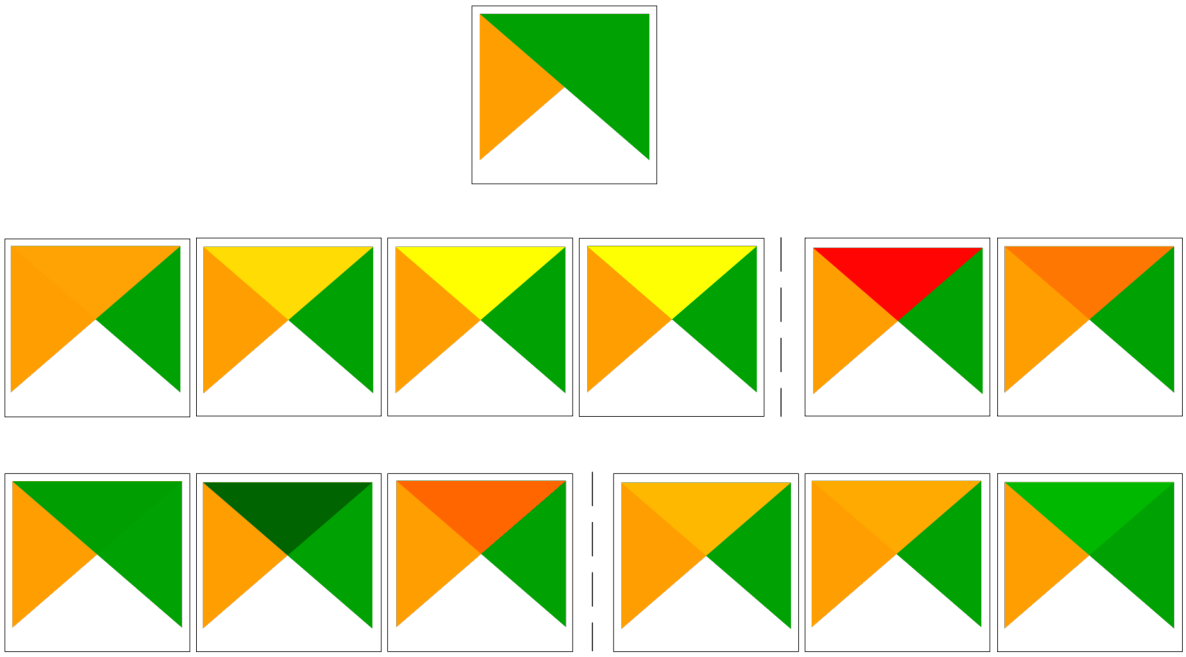


Fig. 12.27: Examples of blend modes applied to a green feature over an orange
From top to bottom, left to right: Normal – Lighten, Screen, Dodge, Addition – Difference, Subtract – Darken, Multiply, Burn – Overlay, Soft light, Hard light

When a layer is part of a group that *renders layers as a group*, additional blending modes are available for the rendering. They provide methods to clip the render of one layer's content by the content in a second „mask" layer.

- *Masked By Below*: The output is the top pixel, where the opacity is reduced by that of the bottom pixel.
- *Mask Below*: The output is the bottom pixel, where the opacity is reduced by that of the top pixel.
- *Inverse Masked By Below*: The output is the top pixel, where the opacity is reduced by the inverse of the bottom pixel.
- *Inverse Mask Below*: The output is the bottom pixel, where the opacity is reduced by the inverse of the top pixel.
- *Paint Inside Below*: The top pixel is blended on top of the bottom pixel, with the opacity of the top pixel reduced by the opacity of the bottom pixel.
- *Paint Below Inside*: The bottom pixel is blended on top of the top pixel, with the opacity of the bottom pixel reduced by the opacity of the top pixel.

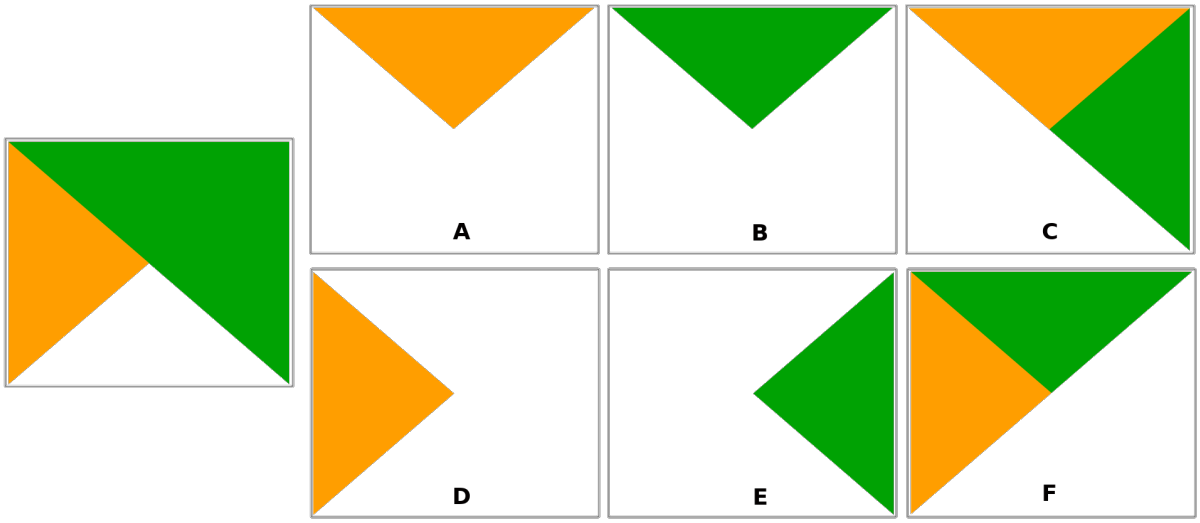


Fig. 12.28: Examples of blend clipping mode applied to top green layer in a group
A: Mask Below **B**: Masked By Below **C**: Paint Below Inside **D**: Inverse Mask Below **E**: Inverse Masked By Below **F**: Paint Inside Below

12.9.10 Configurarea suprascrierii definită de date

Next to many options in the vector layer properties dialog or settings in the print layout, you will find a Data defined override icon. Using *expressions* based on layer attributes or item settings, prebuilt or custom functions and *variables*, this tool allows you to set dynamic values for parameters. When enabled, the value returned by this widget is applied to the parameter regardless of its normal value (checkbox, textbox, slider…).

The data defined override widget

Clicking the Data defined override icon shows the following entries:

- *Description…* that indicates if the option is enabled, which input is expected, the valid input type and the current definition. Hovering over the widget also pops up this information.
- *Store data in the project*: a button allowing the property to be stored using the *Auxiliary Storage Properties* mechanism.
- *Field type*: an entry to select from the layer's fields that match the valid input type.
- *Color*: when the widget is linked to a color property, this menu gives access to the colors defined as part of the current *project's colors* scheme.
- *Variable*: a menu to access the available user-defined *variables*
- *Edit…* button to create or edit the expression to apply, using the *Expression String Builder* dialog. To help you correctly fill in the expression, a reminder of the expected output's format is provided in the dialog.
- *Paste* and *Copy* buttons.
- *Clear* button to remove the setup.
- For numeric and color properties, *Assistant…* to rescale how the feature data is applied to the property (more details *below*)

Sfat: Use right-click to (de)activate the data override

When the data-defined override option is set up correctly the icon is yellow or . If it is broken, the icon is red or .

You can enable or disable a configured Data-defined override button by simply clicking the widget with the right mouse button.

Using the data-defined assistant interface

When the Data-defined override button is associated with a size, a rotation, an opacity or a color property, it has an *Assistant...* option that helps you change how the data is applied to the parameter for each feature. The assistant allows you to:

- Define the *Input* data, ie:
  - *Source*: the attribute to represent, using a field or an *expression*
  - the range of values to represent: you can manually enter the values or use the Fetch value range from layer button to fill these fields automatically with the minimum and maximum values returned by the *Source* expression applied to your data
- *Apply transform curve*: by default, output values (see below for setting) are applied to input features following a linear scale. You can override this logic: enable the transform option, click on the graphic to add break point(s) and drag the point(s) to apply a custom distribution.
- Define the *Output* values: the options vary according to the parameter to define. You can globally set:
  - for a color setting, the *color ramp* to apply to values and the single color to use for NULL values
  - for the others, the minimum and maximum values to apply to the selected property as well as the size/angle/opacity value for ignored or NULL source features
  - for size properties, the *Scale method* of representation which can be **Flannery**, **Exponential**, **Surface**, **Radius** or **Linear**
  - the *Exponent* to use for data scaling when the *Scale method* is of exponential type or when tweaking the opacity

When compatible with the property, a live-update preview is displayed in the right-hand side of the dialog to help you control the value scaling.

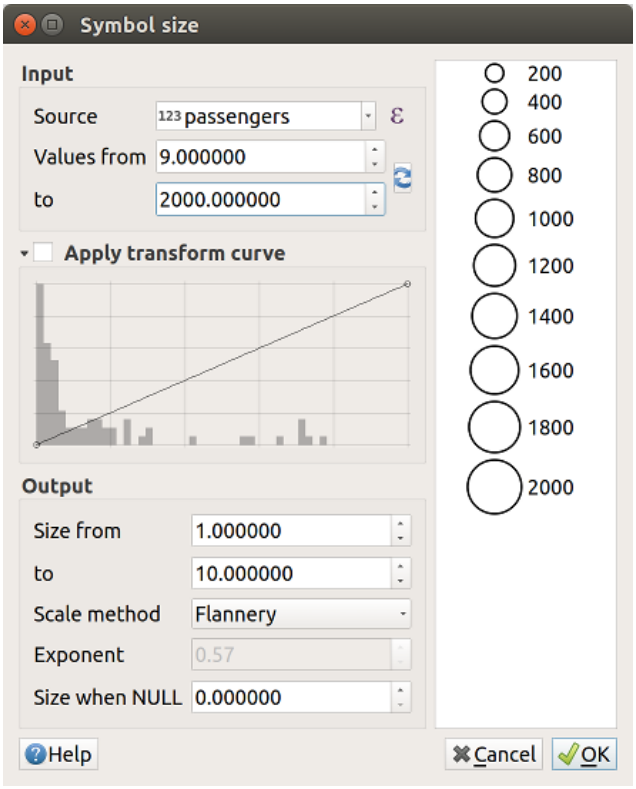


Fig. 12.29: Scaling feature size based on passengers field's value

The values presented in the varying size assistant above will set the size «Data-defined override» with:

```
coalesce(scale_exp("passengers", 9, 2000, 1, 10, 0.57), 0)
```

CAPITOLUL 13

Lucrul cu expresii

13.1 Expresii

Based on layer data and prebuilt or user defined functions, **Expressions** offer a powerful way to manipulate attribute value, geometry and variables in order to dynamically change the geometry style, the content or position of the label, the value for diagram, the height of a layout item, select some features, create virtual field, …

Notă: A list of the default functions and variables for writing expressions can be found at *Lista funcțiilor*, with detailed information and examples.

13.1.1 Constructorul expresiilor de tip șir

Main dialog to build expressions, the *Expression string builder* is available from many parts in QGIS and, can particularly be accessed when:

- făcând clic pe butonul ;
- *selecting features* with the Select By Expression… tool;
- *editing attributes* with e.g. the Field calculator tool;
- manipulating symbology, label or layout item parameters with the Data defined override tool (see *Configurarea suprascrierii definită de date*);
- building a *geometry generator* symbol layer;
- doing some *geoprocessing*.

Dialogul Constructorului de Expresii oferă acces la:

- *Expression* tab which, thanks to a list of *predefined functions*, helps to write and check the expression to use;
- *Function Editor tab* which helps to extend the list of functions by creating custom ones.

The Interface

The *Expression* tab provides the main interface to write expressions using functions, layer fields and values. It contains the following widgets:

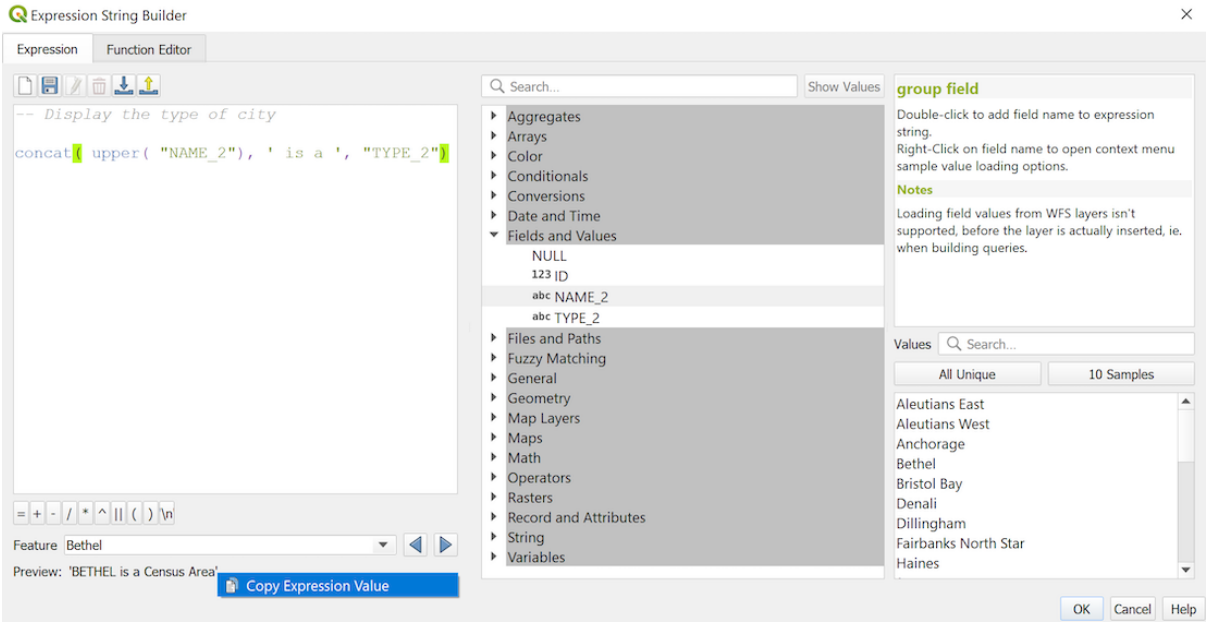


Fig. 13.1: Fila Expresiilor

- An expression editor area for typing or pasting expressions. Autocompletion is available to speed expression writing:
  - Corresponding variables, function names and field names to the input text are shown below: use the `Up` and `Down` arrows to browse the items and press `Tab` to insert in the expression or simply click on the wished item.
  - Function parameters are shown while filling them.

  QGIS also checks the expression rightness and highlights all the errors using:
  - *Underline*: for unknown functions, wrong or invalid arguments;
  - *Marker*: for every other error (eg, missing parenthesis, unexpected character) at a single location.

Sfat: Document your expression with comments

When using complex expression, it is good practice to add text either as a multiline comment or inline comments to help you remember.

```
/*
Labels each region with its highest (in altitude) airport(s)
and altitude, eg 'AMBLER : 264m' for the 'Northwest Artic' region
*/
with_variable(
  'airport_alti', -- stores the highest altitude of the region
  aggregate(
    'airports',
    'max',
    "ELEV", -- the field containing the altitude
    -- and limit the airports to the region they are within
    filter := within( $geometry, geometry( @parent ) )
  ),
```

(continues on next page)

(continuare din pagina precedentă)

```
    aggregate( -- finds airports at the same altitude in the region
      'airports',
      'concatenate',
      "NAME",
      filter := within( $geometry, geometry( @parent ) )
        and "ELEV" = @airport_alti
    )
    || ' : ' || @airport_alti || 'm'
    -- using || allows regions without airports to be skipped
)
```

- Above the expression editor, a set of tools helps you:
  - Clear the expression editor
  - Create and manage *user expressions*
- Under the expression editor, you find:
  - A set of basic operators to help you build the expression
  - An indication of the expected format of output when you are data-defining feature properties
  - A live *Output preview* of the expression (up to 60 characters), evaluated on the first feature of the Layer by default. To view output preview text exceeding 60 characters, you can hover your cursor over the text to display a tooltip pop-up containing the entire output preview. To copy the output preview text onto your clipboard, right-click on the output preview text and select *Copy Expression Value*.

    You can browse and evaluate other features of the layer using the *Feature* combobox (the values are taken from the *display name* property of the layer).

    In case of error, it indicates it and you can access the details with the provided hyperlink.
- A function selector displays the list of functions, variables, fields… organized in groups. A search box is available to filter the list and quickly find a particular function or field. Double-clicking an item adds it to the expression editor.
- A help panel displays help for each selected item in the function selector.

Sfat: Press `Ctrl+Click` when hovering a function name in an expression to automatically display its help in the dialog.

A field's values widget shown when a field is selected in the function selector helps to fetch features attributes:

- Look for a particular field value
- Display the list of *All Unique* or *10 Samples* values. Also available from right-click.

  When the field is mapped with another layer or a set of values, i.e. if the *field widget* is of *RelationReference*, *ValueRelation* or *ValueMap* type, it's possible to list all the values of the mapped field (from the referenced layer, table or list). Moreover, you can filter this list to *Only show values in use* in the current field.

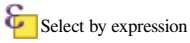
Double-clicking a field value in the widget adds it to the expression editor.

Sfat: The right panel, showing functions help or field values, can be collapsed (invisible) in the dialog. Press the *Show Values* or *Show Help* button to get it back.

Writing an expression

QGIS expressions are used to select features or set values. Writing an expression in QGIS follows some rules:

1. **The dialog defines the context**: if you are used to SQL, you probably know queries of the type *select features from layer where condition* or *update layer set field = new_value where condition*. A QGIS expression also needs all these information but the tool you use to open the expression builder dialog provides parts of them. For example, giving a layer (`buildings`) with a field (`height`):

   - pressing the  tool means that you want to „select features from buildings". The **condition** is the only information you need to provide in the expression text widget, e.g. type `"height" > 20` to select buildings that are higher than 20.
   - with this selection made, pressing the button and choosing „height" as *Update existing field*, you already provide the command „update buildings set height = ??? where height > 20". The only remaining bits you have to provide in this case is the **new value**, e.g. just enter `50` in the expression editor textbox to set the height of the previously selected buildings.

2. **Pay attention to quotes**: single quotes return a literal, so a text placed between single quotes (`'145'`) is interpreted as a string. Double quotes will give you the value of that text so use them for fields (`"myfield"`). Fields can also be used without quotes (`myfield`). No quotes for numbers (`3.16`).

Notă: Functions normally take as argument a string for field name. Do:

```
attribute( @atlas_feature, 'height' ) -- returns the value stored in the
↪"height" attribute of the current atlas feature
```

And not:

```
attribute( @atlas_feature, "height" ) -- fetches the value of the attribute␣
↪named "height" (e.g. 100), and use that value as a field
                                    -- from which to return the atlas␣
↪feature value. Probably wrong as a field named "100" may not exist.
```

Sfat: Use named parameters to ease expression reading

Some functions require many parameters to be set. The expression engine supports the use of named parameters. This means that instead of writing the cryptic expression `clamp( 1, 2, 9)`, you can use `clamp( min:=1, value:=2, max:=9)`. This also allows arguments to be switched, e.g. `clamp( value:=2, max:=9, min:=1)`. Using named parameters helps clarify what the arguments for an expression function refer to, which is helpful when you are trying to interpret an expression later!

Some use cases of expressions

- From the Field Calculator, calculate a „pop_density" field using the existing „total_pop" and „area_km2" fields:

```
"total_pop" / "area_km2"
```

- Label or categorize features based on their area:

```
CASE WHEN $area > 10 000 THEN 'Larger' ELSE 'Smaller' END
```

- Actualizarea câmpului „density_level" cu categorii, în funcție de valorile „pop_density":

```
CASE WHEN "pop_density" < 50 THEN 'Low population density'
     WHEN "pop_density" >= 50 and "pop_density" < 150 THEN 'Medium population␣
```

(continues on next page)

(continuare din pagina precedentă)

```
↪density'
     WHEN "pop_density" >= 150 THEN 'High population density'
END
```

- Aplicarea unui stil categorisit pentru toate entitățile, în funcție de prețul mediu al casei, dacă este mai mică sau mai mare de 10000€ pe metru pătrat:

```
"price_m2" > 10000
```

- Folosind instrumentul de „Selectare După Expresie…”, selectați toate entitățile care reprezintă zonele cu „Densitate ridicată a populației” și în care prețul mediu pentru o casă este mai mare de 10000€ pe metru pătrat:

```
"density_level" = 'High population density' and "price_m2" > 10000
```

  The previous expression could also be used to define which features to label or show on the map.

- Create a different symbol (type) for the layer, using the geometry generator:

```
point_on_surface( $geometry )
```

- Given a point feature, generate a closed line (using `make_line`) around its geometry:

```
make_line(
  -- using an array of points placed around the original
  array_foreach(
    -- list of angles for placing the projected points (every 90°)
    array:=generate_series( 0, 360, 90 ),
    -- translate the point 20 units in the given direction (angle)
    expression:=project( $geometry, distance:=20, azimuth:=radians( @element )␣
↪)
  )
)
```

- In a print layout label, display the name of the „airports” features that are within the layout „Map 1” item:

```
with_variable( 'extent',
               map_get( item_variables( 'Map 1' ), 'map_extent' ),
               aggregate( 'airports', 'concatenate', "NAME",
                          intersects( $geometry, @extent ), ' ,'
                        )
             )
```

Saving Expressions

Using the Add current expression to user expressions button above the expression editor frame, you can save important expressions you want to have quick access to. These are available from the **User expressions** group in the middle panel. They are saved under the *user profile* (`<userprofile>/QGIS/QGIS3.ini` file) and available in all expression dialogs inside all projects of the current user profile.

A set of tools available above the expression editor frame helps you manage the user expressions:

- Add the current expression to user expressions: store the expression in the user profile. A label and a help text can be added for easy identification.
- Edit selected expression from user expressions, as well as their help and label
- Remove selected expression from user expressions

- Import user expressions from a `.json` file into the active user profile folder
- Export user expressions as a `.json` file; all the user expressions in the user profile `QGIS3.ini` file are shared

13.1.2 Editorul de Funcții

With the *Function Editor* tab, you are able to write your own functions in Python language. This provides a handy and comfortable way to address particular needs that would not be covered by the predefined functions.

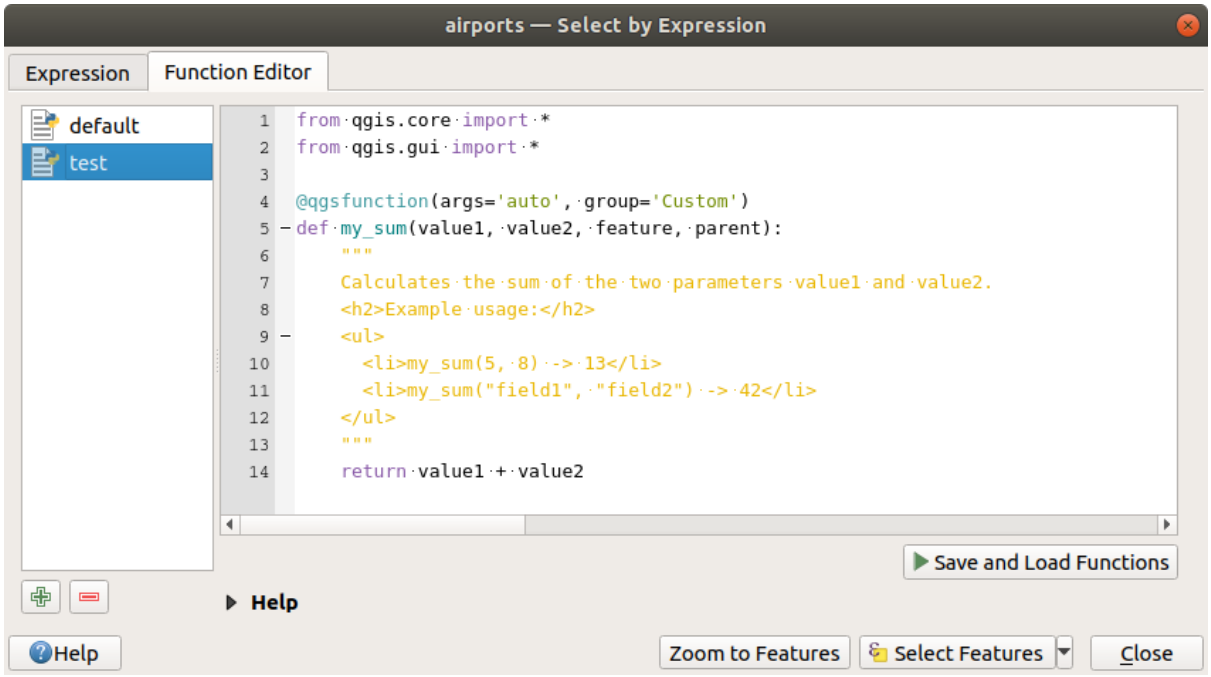


Fig. 13.2: Fila Editorului de Funcții

To create a new function:

1. Press the New File button.
2. Enter a name to use in the form that pops up and press *OK*.

   A new item of the name you provide is added in the left panel of the *Function Editor* tab; this is a Python `.py` file based on QGIS template file and stored in the `/python/expressions` folder under the active *user profile* directory.
3. The right panel displays the content of the file: a python script template. Update the code and its help according to your needs.
4. Press the *Save and Load Functions* button. The function you wrote is added to the functions tree in the *Expression* tab, by default under the `Custom` group.
5. Enjoy your new function.
6. If the function requires improvements, enable the *Function Editor* tab, do the changes and press again the *Save and Load Functions* button to make them available in the file, hence in any expression tab.

Custom Python functions are stored under the user profile directory, meaning that at each QGIS startup, it will auto load all the functions defined with the current user profile. Be aware that new functions are only saved in the `/python/expressions` folder and not in the project file. If you share a project that uses one of your custom functions you will need to also share the `.py` file in the `/python/expressions` folder.

To delete a custom function:

1. Enable the *Function Editor* tab
2. Select the function in the list
3. Press the Remove selected function. The function is removed from the list and the corresponding `.py` file deleted from the user profile folder.

Example

Here's a short example on how to create your own `my_sum` function that will operate with two values.

```
from qgis.core import *
from qgis.gui import *

@qgsfunction(args='auto', group='Custom')
def my_sum(value1, value2, feature, parent):
    """
    Calculates the sum of the two parameters value1 and value2.
    <h2>Example usage:</h2>
    <ul>
      <li>my_sum(5, 8) -> 13</li>
      <li>my_sum("field1", "field2") -> 42</li>
    </ul>
    """
    return value1 + value2
```

The `@qgsfunction` decorator accepts the following arguments:

- `args`: the number of arguments. When using the `args='auto'` argument the number of function arguments required will be calculated by the number of arguments the function has been defined with in Python (minus 2 - `feature`, and `parent`). With `args = -1`, any number of arguments are accepted.
- The `group` argument indicates the group in which the function should be listed in the Expression dialog.
- `usesgeometry=True` if the expression requires access to the features geometry. By default `False`.
- `handlesnull=True` if the expression has custom handling for NULL values. If `False` (default), the result will always be NULL as soon as any parameter is NULL.
- `referenced_columns=[list]`: An array of attribute names that are required to the function. Defaults to `[QgsFeatureRequest.ALL_ATTRIBUTES]`.

The function itself allows following arguments:

- any number and type of parameters you want to pass to your function, set before the following arguments.
- `feature`: the current feature
- `parent`: the `QgsExpression` object
- `context`: If there is an argument called `context` found at the last position, this variable will contain a `QgsExpressionContext` object, that gives access to various additional information like expression variables. E.g. `context.variable( 'layer_id' )`

The previous example function can then be used in expressions:

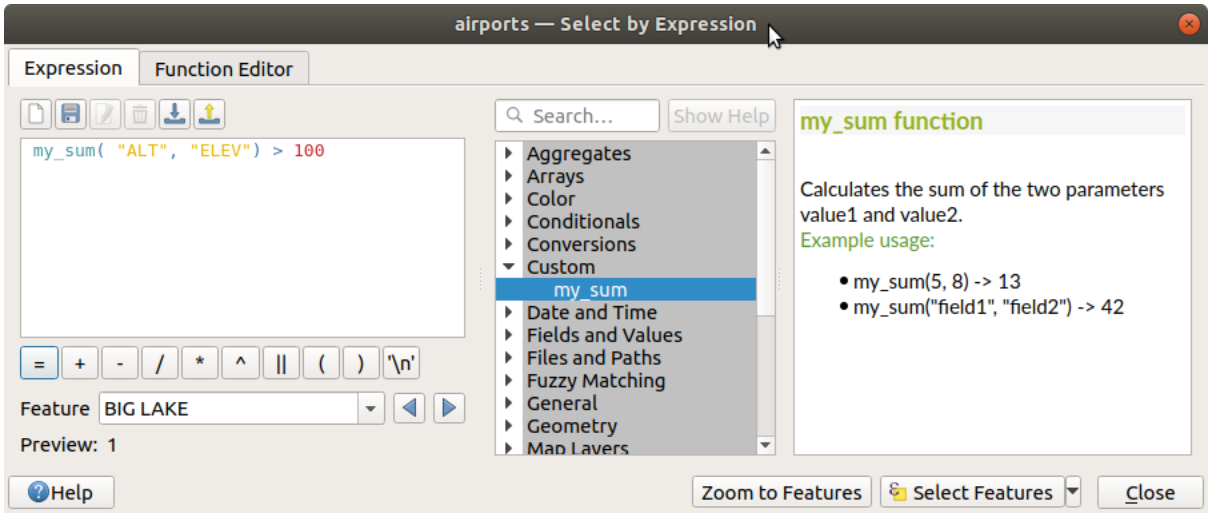


Fig. 13.3: Custom Function added to the Expression tab

Mai multe informații despre crearea de cod Python pot fi găsite în PyQGIS-Developer-Cookbook.

13.2 Lista funcțiilor

The functions, operators and variables available in QGIS are listed below, grouped by categories.

13.2.1 Funcții de Agregare

This group contains functions which aggregate values over layers and fields.

aggregate

Returns an aggregate value calculated using features from another layer.

Sintaxă	aggregate(layer, aggregate, expression, [filter], [concatenator=»»], [order_by]) [] marks optional arguments
Argumente	• **layer** - a string, representing either a layer name or layer ID • **aggregate** - a string corresponding to the aggregate to calculate. Valid options are: – count – count_distinct – count_missing – min – max – sum – mean – median – stdev – stdevsample – range – minority – majority – q1: first quartile – q3: third quartile – iqr: inter quartile range – min_length: minimum string length – max_length: maximum string length – concatenate: join strings with a concatenator – concatenate_unique: join unique strings with a concatenator – collect: create an aggregated multipart geometry – array_agg: create an array of aggregated values • **expression** - sub expression or field name to aggregate • **filter** - optional filter expression to limit the features used for calculating the aggregate. Fields and geometry are from the features on the joined layer. The source feature can be accessed with the variable @parent. • **concatenator** - optional string to use to join values for «concatenate» and «concatenate_unique» aggregates • **order_by** - optional filter expression to order the features used for calculating the aggregate. Fields and geometry are from the features on the joined layer. By default, the features will be returned in an unspecified order.
Exemple	• `aggregate(layer:='rail_stations',aggregate:='sum', expression:="passengers")` → sum of all values from the passengers field in the rail_stations layer • `aggregate('rail_stations','sum', "passengers"/7)` → calculates a daily average of „passengers” by dividing the „passengers” field by 7 before summing the values • `aggregate(layer:='rail_stations',aggregate:='sum', expression:="passengers",filter:="class">3)` → sums up all values from the „passengers” field from features where the „class” attribute is greater than 3 only • `aggregate(layer:='rail_stations',aggregate:='concatenate', expression:="name", concatenator:=',')` → comma separated list of the name field for all features in the rail_stations layer • `aggregate(layer:='countries', aggregate:='max', expression:="code", filter:=intersects( @geometry, geometry(@parent) ) )` → The country code of an intersecting country on the layer «countries» • `aggregate(layer:='rail_stations',aggregate:='sum', expression:="passengers",filter:=contains( @atlas_geometry, @geometry ) )` → sum of all values from the passengers field in the rail_stations within the current atlas feature • `aggregate(layer:='rail_stations', aggregate:='collect', expression:=centroid(@geometry), filter:="region_name" attribute(@parent,'name') )` → aggregates centroid geometries of the rail_stations of the same region as current feature

array_agg

Returns an array of aggregated values from a field or expression.

Sintaxă	array_agg(expression, [group_by], [filter], [order_by]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate • **order_by** - optional expression to use to order features used to calculate aggregate. By default, the features will be returned in an unspecified order.
Exemple	• `array_agg("name",group_by:="state")` → list of name values, grouped by state field

collect

Returns the multipart geometry of aggregated geometries from an expression

Sintaxă	collect(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - geometry expression to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `collect( @geometry )` → multipart geometry of aggregated geometries • `collect( centroid(@geometry), group_by:="region", filter:= "use" = 'civilian' )` → aggregated centroids of the civilian features based on their region value

concatenate

Returns all aggregated strings from a field or expression joined by a delimiter.

Sintaxă	concatenate(expression, [group_by], [filter], [concatenator], [order_by]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate • **concatenator** - optional string to use to join values. Empty by default. • **order_by** - optional expression to use to order features used to calculate aggregate. By default, the features will be returned in an unspecified order.
Exemple	• `concatenate("town_name",group_by:="state",concatenator:=', ')` → comma separated list of town_names, grouped by state field

concatenate_unique

Returns all unique strings from a field or expression joined by a delimiter.

Sintaxă	concatenate_unique(expression, [group_by], [filter], [concatenator], [order_by]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate • **concatenator** - optional string to use to join values. Empty by default. • **order_by** - optional expression to use to order features used to calculate aggregate. By default, the features will be returned in an unspecified order.
Exemple	• `concatenate_unique("town_name",group_by:="state", concatenator:=',')` → comma separated list of unique town_names, grouped by state field

count

Returns the count of matching features.

Sintaxă	count(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `count("stations",group_by:="state")` → count of stations, grouped by state field

count_distinct

Returns the count of distinct values.

Sintaxă	count_distinct(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `count_distinct("stations",group_by:="state")` → count of distinct stations values, grouped by state field

count_missing

Returns the count of missing (NULL) values.

Sintaxă	count_missing(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `count_missing("stations",group_by:="state")` → count of missing (NULL) station values, grouped by state field

iqr

Returns the calculated inter quartile range from a field or expression.

Sintaxă	iqr(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `iqr("population",group_by:="state")` → inter quartile range of population value, grouped by state field

majority

Returns the aggregate majority of values (most commonly occurring value) from a field or expression.

Sintaxă	majority(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `majority("class",group_by:="state")` → most commonly occurring class value, grouped by state field

max_length

Returns the maximum length of strings from a field or expression.

Sintaxă	max_length(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `max_length("town_name",group_by:="state")` → maximum length of town_name, grouped by state field

maximum

Returns the aggregate maximum value from a field or expression.

Sintaxă	maximum(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `maximum("population",group_by:="state")` → maximum population value, grouped by state field

mean

Returns the aggregate mean value from a field or expression.

Sintaxă	mean(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `mean("population",group_by:="state")` → mean population value, grouped by state field

median

Returns the aggregate median value from a field or expression.

Sintaxă	median(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `median("population",group_by:="state")` → median population value, grouped by state field

min_length

Returns the minimum length of strings from a field or expression.

Sintaxă	min_length(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `min_length("town_name",group_by:="state")` → minimum length of town_name, grouped by state field

minimum

Returns the aggregate minimum value from a field or expression.

Sintaxă	minimum(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `minimum("population",group_by:="state")` → minimum population value, grouped by state field

minority

Returns the aggregate minority of values (least occurring value) from a field or expression.

Sintaxă	minority(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `minority("class",group_by:="state")` → least occurring class value, grouped by state field

q1

Returns the calculated first quartile from a field or expression.

Sintaxă	q1(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `q1("population",group_by:="state")` → first quartile of population value, grouped by state field

q3

Returns the calculated third quartile from a field or expression.

Sintaxă	q3(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `q3("population",group_by:="state")` → third quartile of population value, grouped by state field

range

Returns the aggregate range of values (maximum - minimum) from a field or expression.

Sintaxă	range(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `range("population",group_by:="state")` → range of population values, grouped by state field

relation_aggregate

Returns an aggregate value calculated using all matching child features from a layer relation.

Sintaxă	relation_aggregate(relation, aggregate, expression, [concatenator=»»], [order_by]) [] marks optional arguments
Argumente	• **relation** - a string, representing a relation ID • **aggregate** - a string corresponding to the aggregate to calculate. Valid options are: – count – count_distinct – count_missing – min – max – sum – mean – median – stdev – stdevsample – range – minority – majority – q1: first quartile – q3: third quartile – iqr: inter quartile range – min_length: minimum string length – max_length: maximum string length – concatenate: join strings with a concatenator – concatenate_unique: join unique strings with a concatenator – collect: create an aggregated multipart geometry – array_agg: create an array of aggregated values • **expression** - sub expression or field name to aggregate • **concatenator** - optional string to use to join values for «concatenate» aggregate • **order_by** - optional expression to order the features used for calculating the aggregate. Fields and geometry are from the features on the joined layer. By default, the features will be returned in an unspecified order.
Exemple	• `relation_aggregate(relation:='my_relation', aggregate:='mean',expression:="passengers")` → mean value of all matching child features using the «my_relation» relation • `relation_aggregate('my_relation','sum', "passengers"/7)` → sum of the passengers field divided by 7 for all matching child features using the «my_relation» relation • `relation_aggregate('my_relation','concatenate', "towns", concatenator:=',')` → comma separated list of the towns field for all matching child features using the «my_relation» relation • `relation_aggregate('my_relation','array_agg', "id")` → array of the id field from all matching child features using the «my_relation» relation

Mai multe detalii: *Setting relations between multiple layers*

stdev

Returns the aggregate standard deviation value from a field or expression.

Sintaxă	stdev(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `stdev("population",group_by:="state")` → standard deviation of population value, grouped by state field

sum

Returns the aggregate summed value from a field or expression.

Sintaxă	sum(expression, [group_by], [filter]) [] marks optional arguments
Argumente	• **expression** - sub expression of field to aggregate • **group_by** - optional expression to use to group aggregate calculations • **filter** - optional expression to use to filter features used to calculate aggregate
Exemple	• `sum("population",group_by:="state")` → summed population value, grouped by state field

13.2.2 Array Functions

This group contains functions to create and manipulate arrays (also known as list data structures). The order of values within the array matters, unlike the *«map» data structure*, where the order of key-value pairs is irrelevant and values are identified by their keys.

array

Returns an array containing all the values passed as parameter.

Sintaxă	array(value1, value2, ...)
Argumente	• **value** - o valoare
Exemple	• `array(2,10)` → [2, 10] • `array(2,10)[0]` → 2

array_all

Returns TRUE if an array contains all the values of a given array.

Sintaxă	array_all(array_a, array_b)
Argumente	• **array_a** - an array • **array_b** - the array of values to search
Exemple	• `array_all(array(1,2,3),array(2,3))` → TRUE • `array_all(array(1,2,3),array(1,2,4))` → FALSE

array_append

Returns an array with the given value added at the end.

Sintaxă	array_append(array, value)
Argumente	• **array** - an array • **value** - valoarea de adăugat
Exemple	• `array_append(array(1,2,3),4)` → [1, 2, 3, 4]

array_cat

Returns an array containing all the given arrays concatenated.

Sintaxă	array_cat(array1, array2, …)
Argumente	• **array** - an array
Exemple	• `array_cat(array(1,2),array(2,3))` → [1, 2, 2, 3]

array_contains

Returns TRUE if an array contains the given value.

Sintaxă	array_contains(array, value)
Argumente	• **array** - an array • **value** - the value to search
Exemple	• `array_contains(array(1,2,3),2)` → TRUE

array_count

Counts the number of occurrences of a given value in an array.

Sintaxă	array_count(array, value)
Argumente	• **array** - an array • **value** - the value to count
Exemple	• `array_count(array('a', 'b', 'c', 'b'), 'b')` → 2

array_distinct

Returns an array containing distinct values of the given array.

Sintaxă	array_distinct(array)
Argumente	• **array** - an array
Exemple	• `array_distinct(array(1,2,3,2,1))` → [1, 2, 3]

array_filter

Returns an array with only the items for which the expression evaluates to true.

Sintaxă	array_filter(array, expression, [limit=0]) [] marks optional arguments
Argumente	• **array** - an array • **expression** - an expression to evaluate on each item. The variable *@element* will be replaced by the current value. • **limit** - maximum number of elements to be returned. Use 0 to return all values.
Exemple	• `array_filter(array(1,2,3),@element < 3)` → [1, 2] • `array_filter(array(1,2,3),@element < 3, 1)` → [1]

array_find

Returns the lowest index (0 for the first one) of a value within an array. Returns -1 if the value is not found.

Sintaxă	array_find(array, value)
Argumente	• **array** - an array • **value** - the value to search
Exemple	• `array_find(array('a', 'b', 'c'), 'b')` → 1 • `array_find(array('a', 'b', 'c', 'b'), 'b')` → 1

array_first

Returns the first value of an array.

Sintaxă	array_first(array)
Argumente	• **array** - an array
Exemple	• `array_first(array('a','b','c'))` → «a»

array_foreach

Returns an array with the given expression evaluated on each item.

Sintaxă	array_foreach(array, expression)
Argumente	• **array** - an array • **expression** - an expression to evaluate on each item. The variable *@element* will be replaced by the current value.
Exemple	• `array_foreach(array('a','b','c'),upper(@element))` → [«A», «B», «C»] • `array_foreach(array(1,2,3),@element + 10)` → [11, 12, 13]

array_get

Returns the Nth value (0 for the first one) or the last -Nth value (-1 for the last one) of an array.

Sintaxă	array_get(array, pos)
Argumente	• **array** - an array • **pos** - the index to get (0 based)
Exemple	• `array_get(array('a','b','c'),1)` → «b» • `array_get(array('a','b','c'),-1)` → «c»

Sugestie: You can also use the *index operator ([])* to get a value from an array.

array_insert

Returns an array with the given value added at the given position.

Sintaxă	array_insert(array, pos, value)
Argumente	• **array** - an array • **pos** - the position where to add (0 based) • **value** - valoarea de adăugat
Exemple	• `array_insert(array(1,2,3),1,100)` → [1, 100, 2, 3]

array_intersect

Returns TRUE if at least one element of array1 exists in array2.

Sintaxă	array_intersect(array1, array2)
Argumente	• **array1** - an array • **array2** - another array
Exemple	• `array_intersect(array(1,2,3,4),array(4,0,2,5))` → TRUE

array_last

Returns the last value of an array.

Sintaxă	array_last(array)
Argumente	• **array** - an array
Exemple	• `array_last(array('a','b','c'))` → «c»

array_length

Returns the number of elements of an array.

Sintaxă	array_length(array)
Argumente	• **array** - an array
Exemple	• `array_length(array(1,2,3))` → 3

array_majority

Returns the most common values in an array.

Sintaxă	array_majority(array, [option=»all»]) [] marks optional arguments
Argumente	• **array** - an array • **option=»all»** - a string specifying the return values handling. Valid options are: – all: Default, all most common values are returned in an array. – any: Returns one of the most common values. – median: Returns the median of the most common values. Non arithmetic values are ignored. – real_majority: Returns the value which occurs more than half the size of the array.
Exemple	• `array_majority(array(0,1,42,42,43), 'all')` → [42] • `array_majority(array(0,1,42,42,43,1), 'all')` → [42, 1] • `array_majority(array(0,1,42,42,43,1), 'any')` → 1 or 42 • `array_majority(array(0,1,1,2,2), 'median')` → 1.5 • `array_majority(array(0,1,42,42,43), 'real_majority')` → NULL • `array_majority(array(0,1,42,42,43,42), 'real_majority')` → NULL • `array_majority(array(0,1,42,42,43,42,42), 'real_majority')` → 42

array_max

Returns the maximum value of an array.

Sintaxă	array_max(array)
Argumente	• **array** - an array
Exemple	• `array_max(array(0,42,4,2))` → 42

array_mean

Returns the mean of arithmetic values in an array. Non numeric values in the array are ignored.

Sintaxă	array_mean(array)
Argumente	• **array** - an array
Exemple	• `array_mean(array(0,1,7,66.6,135.4))` → 42 • `array_mean(array(0,84,'a','b','c'))` → 42

array_median

Returns the median of arithmetic values in an array. Non arithmetic values in the array are ignored.

Sintaxă	array_median(array)
Argumente	• **array** - an array
Exemple	• `array_median(array(0,1,42,42,43))` → 42 • `array_median(array(0,1,2,42,'a','b'))` → 1.5

array_min

Returns the minimum value of an array.

Sintaxă	array_min(array)
Argumente	• **array** - an array
Exemple	• `array_min(array(43,42,54))` → 42

array_minority

Returns the less common values in an array.

Sintaxă	array_minority(array, [option=»all»]) [] marks optional arguments
Argumente	• **array** - an array • **option=»all»** - a string specifying the return values handling. Valid options are: – all: Default, all less common values are returned in an array. – any: Returns one of the less common values. – median: Returns the median of the less common values. Non arithmetic values are ignored. – real_minority: Returns values which occur less than half the size of the array.
Exemple	• `array_minority(array(0,42,42), 'all')` → [0] • `array_minority(array(0,1,42,42), 'all')` → [0, 1] • `array_minority(array(0,1,42,42,43,1), 'any')` → 0 or 43 • `array_minority(array(1,2,3,3), 'median')` → 1.5 • `array_minority(array(0,1,42,42,43), 'real_minority')` → [42, 43, 0, 1] • `array_minority(array(0,1,42,42,43,42), 'real_minority')` → [42, 43, 0, 1] • `array_minority(array(0,1,42,42,43,42,42), 'real_minority')` → [43, 0, 1]

array_prepend

Returns an array with the given value added at the beginning.

Sintaxă	array_prepend(array, value)
Argumente	• **array** - an array • **value** - valoarea de adăugat
Exemple	• `array_prepend(array(1,2,3),0)` → [0, 1, 2, 3]

array_prioritize

Returns an array sorted using the ordering specified in another array. Values which are present in the first array but are missing from the second array will be added to the end of the result.

Sintaxă	array_prioritize(array, array_prioritize)
Argumente	• **array** - an array • **array_prioritize** - an array with values ordered by priority
Exemple	• `array_prioritize(array(1, 8, 2, 5), array(5, 4, 2, 1, 3, 8))` → [5, 2, 1, 8] • `array_prioritize(array(5, 4, 2, 1, 3, 8), array(1, 8, 6, 5))` → [1, 8, 5, 4, 2, 3]

array_remove_all

Returns an array with all the entries of the given value removed.

Sintaxă	array_remove_all(array, value)
Argumente	• **array** - an array • **value** - the values to remove
Exemple	• `array_remove_all(array('a','b','c','b'),'b')` → [«a», «c»]

array_remove_at

Returns an array with the item at the given index removed. Supports positive (0 for the first element) and negative (the last -Nth value, -1 for the last element) index.

Sintaxă	array_remove_at(array, pos)
Argumente	• **array** - an array • **pos** - the position to remove (0 based)
Exemple	• `array_remove_at(array(1, 2, 3), 1)` → [1, 3] • `array_remove_at(array(1, 2, 3), -1)` → [1, 2]

array_replace

Returns an array with the supplied value, array, or map of values replaced.

Value & array variant

Returns an array with the supplied value or array of values replaced by another value or an array of values.

Sintaxă	array_replace(array, before, after)
Argumente	• **array** - the input array • **before** - the value or array of values to replace • **after** - the value or array of values to use as a replacement
Exemple	• `array_replace(array('QGIS','SHOULD','ROCK'),'SHOULD', 'DOES')` → [«QGIS», «DOES», «ROCK»] • `array_replace(array(3,2,1),array(1,2,3),array(7,8,9))` → [9, 8, 7] • `array_replace(array('Q','G','I','S'),array('Q','S'),'-')` → [«-», «G», «I», «-»]

Map variant

Returns an array with the supplied map keys replaced by their paired values.

Sintaxă	array_replace(array, map)
Argumente	• **array** - the input array • **map** - the map containing keys and values
Exemple	• `array_replace(array('APP', 'SHOULD', 'ROCK'),map('APP', 'QGIS','SHOULD','DOES'))` → [«QGIS», «DOES», «ROCK»]

array_reverse

Returns the given array with array values in reversed order.

Sintaxă	array_reverse(array)
Argumente	• **array** - an array
Exemple	• `array_reverse(array(2,4,0,10))` → [10, 0, 4, 2]

array_slice

Returns a portion of the array. The slice is defined by the start_pos and end_pos arguments.

Sintaxă	array_slice(array, start_pos, end_pos)
Argumente	• **array** - an array • **start_pos** - the index of the start position of the slice (0 based). The start_pos index is included in the slice. If you use a negative start_pos, the index is counted from the end of the list (-1 based). • **end_pos** - the index of the end position of the slice (0 based). The end_pos index is included in the slice. If you use a negative end_pos, the index is counted from the end of the list (-1 based).
Exemple	• `array_slice(array(1,2,3,4,5),0,3)` → [1, 2, 3, 4] • `array_slice(array(1,2,3,4,5),0,-1)` → [1, 2, 3, 4, 5] • `array_slice(array(1,2,3,4,5),-5,-1)` → [1, 2, 3, 4, 5] • `array_slice(array(1,2,3,4,5),0,0)` → [1] • `array_slice(array(1,2,3,4,5),-2,-1)` → [4, 5] • `array_slice(array(1,2,3,4,5),-1,-1)` → [5] • `array_slice(array('Dufour','Valmiera','Chugiak', 'Brighton'),1,2)` → [«Valmiera», «Chugiak»] • `array_slice(array('Dufour','Valmiera','Chugiak', 'Brighton'),-2,-1)` → [«Chugiak», «Brighton»]

array_sort

Returns the provided array with its elements sorted.

Sintaxă	array_sort(array, [ascending=true]) [] marks optional arguments
Argumente	• **array** - an array • **ascending** - set this parameter to false to sort the array in descending order
Exemple	• `array_sort(array(3,2,1))` → [1, 2, 3]

array_sum

Returns the sum of arithmetic values in an array. Non numeric values in the array are ignored.

Sintaxă	array_sum(array)
Argumente	• **array** - an array
Exemple	• `array_sum(array(0,1,39.4,1.6,'a'))` → 42.0

array_to_string

Concatenates array elements into a string separated by a delimiter and using optional string for empty values.

Sintaxă	array_to_string(array, [delimiter=»,»], [empty_value=»»]) [] marks optional arguments
Argumente	• **array** - the input array • **delimiter** - the string delimiter used to separate concatenated array elements • **empty_value** - the optional string to use as replacement for empty (zero length) matches
Exemple	• `array_to_string(array('1','2','3'))` → «1,2,3» • `array_to_string(array(1,2,3),'-')` → «1-2-3» • `array_to_string(array('1','','3'),',','0')` → «1,0,3»

generate_series

Creează o matrice care conține o secvență de numere.

Sintaxă	generate_series(start, stop, [step=1]) [] marks optional arguments
Argumente	• **start** - prima valoare a secvenței • **stop** - valoare care încheie secvența o dată atinsă • **pas** - valoare folosită ca increment între valori
Exemple	• `generate_series(1,5)` → [1, 2, 3, 4, 5] • `generate_series(5,1,-1)` → [5, 4, 3, 2, 1]

geometries_to_array

Împarte o geometrie într-o matrice cu geometrii mai simple.

Sintaxă	geometries_to_array(geometry)
Argumente	• **geometrie** - geometria de intrare
Exemple	• `geometries_to_array(geom_from_wkt('GeometryCollection (Poligon ((5 8, 4 1, 3 2, 5 8)),LineString (3 2, 4 2))'))` → o matrice de geometrii poligonale și liniare • `geom_to_wkt(geometries_to_array(geom_from_wkt('GeometryCollection (Polygon ((5 8, 4 1, 3 2, 5 8)),LineString (3 2, 4 2))'))[0])` → «Polygon ((5 8, 4 1, 3 2, 5 8))» • `geometries_to_array(geom_from_wkt('MULTIPOLYGON(((5 5,0 0, 0 10,5 5)),((5 5,10 10,10 0,5 5))'))` → o matrice cu două geometrii poligonale

regexp_matches

Returnează o matrice având toate șirurile capturate de grupuri, în ordinea în care grupurile însele apar în expresia regulată furnizată față de un șir.

Sintaxă	regexp_matches(string, regex, [empty_value=»»]) [] marks optional arguments
Argumente	• **șir** - șirul din care expresia regulată va captura grupurile • **regex** - expresia regulată folosită pentru a captura grupuri • **empty_value** - the optional string to use as replacement for empty (zero length) matches
Exemple	• `regexp_matches('QGIS=>rocks','(.*)=>(.*)')` → [«QGIS», «rocks»] • `regexp_matches('key=>','(.*)=>(.*)','empty value')` → [«key», «empty value»]

string_to_array

Împarte șirul într-o matrice, folosind delimitatorul furnizat și șirul opțional pentru valorile goale.

Sintaxă	string_to_array(string, [delimiter=»,»], [empty_value=»»]) [] marks optional arguments
Argumente	• **string** - șirul de intrare • **delimitator** - delimitatorul folosit pentru a diviza șirul de intrare • **empty_value** - the optional string to use as replacement for empty (zero length) matches
Exemple	• `string_to_array('1,2,3',',')` → [«1», «2», «3»] • `string_to_array('1,,3',',','0')` → [«1», «0», «3»]

13.2.3 Funcțiile Culorilor

Acest grup conține funcții pentru manipularea culorilor.

color_cmyk

Returnează o reprezentare de tip șir a unei culori, în funcție de componentele ei cyan, magenta, galben și negru

Sintaxă	color_cmyk(cyan, magenta, yellow, black)
Argumente	• **cyan** - componenta cyan a culorii, ca valoare procentuală întreagă, de la 0 la 100 • **magenta** - componenta magenta a culorii, ca valoare procentuală întreagă, de la 0 la 100 • **galben** - componenta galbenă a culorii, ca valoare întreagă procentuală, de la 0 la 100 • **negru** - componenta neagră a culorii, ca valoare întreagă procentuală, de la 0 la 100
Exemple	• `color_cmyk(100,50,0,10)` → «0,115,230»

color_cmyka

Returnează o reprezentare de tip șir a unei culori, în funcție de componentele ei cyan, magenta, galben, negru și alpha (transparență)

Sintaxă	color_cmyka(cyan, magenta, yellow, black, alpha)
Argumente	• **cyan** - componenta cyan a culorii, ca valoare procentuală întreagă, de la 0 la 100 • **magenta** - componenta magenta a culorii, ca valoare procentuală întreagă, de la 0 la 100 • **galben** - componenta galbenă a culorii, ca valoare întreagă procentuală, de la 0 la 100 • **negru** - componenta neagră a culorii, ca valoare întreagă procentuală, de la 0 la 100 • **alfa** - componentă alfa, ca valoare întreagă de la 0 (transparență completă) la 255 (opacitate completă).
Exemple	• `color_cmyka(100,50,0,10,200)` → «0,115,230,200»

color_grayscale_average

Aplică un filtru în tonuri de gri și returnează o reprezentare sub formă de șir dintr-o culoare furnizată.

Sintaxă	color_grayscale_average(color)
Argumente	• **culoare** - culoarea sub formă de șir
Exemple	• `color_grayscale_average('255,100,50')` → «135,135,135,255»

color_hsl

Returnează o reprezentare sub formă de șir a unei culori, pe baza atributelor sale de nuanță, saturație și luminozitate.

Sintaxă	color_hsl(hue, saturation, lightness)
Argumente	• **hue** - nuanța culorii, ca valoare întreagă, de la 0 la 360 • **saturation** - procentajul saturației culorii, ca valoare întreagă, de la 0 la 100 • **lightness** - procentul de luminozitate al culorii, ca valoare întreagă, de la 0 la 100
Exemple	• `color_hsl(100,50,70)` → «166,217,140»

color_hsla

Returnează o reprezentare de tip șir a culorii, pe baza atributelor nuanței, saturației, luminozității și alpha (transparență)

Sintaxă	color_hsla(hue, saturation, lightness, alpha)
Argumente	• **hue** - nuanța culorii, ca valoare întreagă, de la 0 la 360 • **saturation** - procentajul saturației culorii, ca valoare întreagă, de la 0 la 100 • **lightness** - procentul de luminozitate al culorii, ca valoare întreagă, de la 0 la 100 • **alfa** - componentă alfa, ca valoare întreagă de la 0 (transparență completă) la 255 (opacitate completă).
Exemple	• `color_hsla(100,50,70,200)` → «166,217,140,200»

color_hsv

Returnează o reprezentare sub formă de șir a unei culori, pe baza atributelor sale de nuanță, saturație și valoare.

Sintaxă	color_hsv(hue, saturation, value)
Argumente	• **hue** - nuanța culorii, ca valoare întreagă, de la 0 la 360 • **saturation** - procentajul saturației culorii, ca valoare întreagă, de la 0 la 100 • **value** - valoare procentuală a culorii, ca număr întreg, de la 0 la 100
Exemple	• `color_hsv(40,100,100)` → «255,170,0»

color_hsva

Returnează o reprezentare de tip șir a unei culori, bazată pe atributele nuanța, saturația, valoarea și alfa (transparență).

Sintaxă	color_hsva(hue, saturation, value, alpha)
Argumente	• **hue** - nuanța culorii, ca valoare întreagă, de la 0 la 360 • **saturation** - procentajul saturației culorii, ca valoare întreagă, de la 0 la 100 • **value** - valoare procentuală a culorii, ca număr întreg, de la 0 la 100 • **alpha** - componenta alfa, ca valoare întreagă de la 0 (transparență completă) la 255 (opacitate completă).
Exemple	• `color_hsva(40,100,100,200)` → «255,170,0,200»

color_mix_rgb

Returnează un șir reprezentând o culoare dată de amestecul valorilor roșu, verde, albastru și alfa furnizate pentru două culori și în baza unei rații date.

Sintaxă	color_mix_rgb(color1, color2, ratio)
Argumente	• **color1** - culoarea 1 sub formă de șir • **color2** - culoarea 2 sub formă de șir • **ratio** - rezultatul raportului dintre două valori
Exemple	• `color_mix_rgb('0,0,0','255,255,255',0.5)` → «127,127,127,255»

color_part

Returnează o anumită componentă din șirul unei culori; de ex.: componenta roșie sau componenta alfa.

Sintaxă	color_part(color, component)
Argumente	• **culoare** - culoarea sub formă de șir • **component** - un șir corespunzător componentei de culoare de returnat. Opțiunile valide sunt: – red: codul RGB al componentei roșii (0-255) – green: codul RGB al componentei verzi (0-255) – blue: codul RGB al componentei albastre (0-255) – alpha: valoarea alfa (transparența) (0-255) – hue: codul HSV al nuanței (0-360) – saturation: codul HSV al saturației (0-100) – value: codul HSV al valorii (0-100) – hsl_hue: codul HSL al nuanței (0-360) – hsl_saturation: codul HSL al saturației (0-100) – lightness: codul HSL al luminozității (0-100) – cyan: codul CMYK al componentei cyan (0-100) – magenta: codul CMYK al componentei magenta (0-100) – yellow: codul CMYK al componentei galbene (0-100) – black: codul CMYK al componentei negre (0-100)
Exemple	• `color_part('200,10,30','green')` → 10

color_rgb

Returnează o reprezentare de tip șir a unei culori, în funcție de componentele sale roșie, verde și albastră.

Sintaxă	color_rgb(red, green, blue)
Argumente	• **red** - componenta roșie, ca valoare întreagă de la 0 la 255 • **green** - componenta verde, ca valoare întreagă de la 0 la 255 • **blue** - componenta albastră, ca valoare întreagă de la 0 la 255
Exemple	• `color_rgb(255,127,0)` → «255,127,0»

color_rgba

Returnează o reprezentare de tip șir a unei culori, în funcție de componentele sale roșie, verde, albastră și alfa (transparență).

Sintaxă	color_rgba(red, green, blue, alpha)
Argumente	• **red** - componenta roșie, ca valoare întreagă de la 0 la 255 • **green** - componenta verde, ca valoare întreagă de la 0 la 255 • **blue** - componenta albastră, ca valoare întreagă de la 0 la 255 • **alfa** - componentă alfa, ca valoare întreagă de la 0 (transparență completă) la 255 (opacitate completă).
Exemple	• `color_rgba(255,127,0,200)` → «255,127,0,200»

create_ramp

Returnează o rampă gradientală pornind de la o hartă cu șiruri de culori și pași.

Sintaxă	create_ramp(map, [discrete=false]) [] marks optional arguments
Argumente	• **map** - o hartă cu șiruri de culori și pași • **discrete** - setați acest parametru la true pentru a crea o rampă de culori discretă
Exemple	• `ramp_color(create_ramp(map(0,'0,0,0',1,'255,0,0')),1)` → «255,0,0,255»

darker

Returnează o culoare, de tip șir, mai închisă (sau mai deschisă)

Sintaxă	darker(color, factor)
Argumente	• **culoare** - culoarea sub formă de șir • **factor** - un număr întreg corespunzător factorului de întunecare: – dacă factorul este mai mare de 100, această funcție returnează o culoare mai închisă (de ex.: setarea factorului la 200 returnează o culoare care are o jumătate din luminozitate); – dacă factorul este mai mic de 100, culoarea returnată este mai deschisă; totuși, se recomandă utilizarea funcției lighter() în acest scop; – dacă factorul este 0 sau negativ, valoarea returnată este nespecificată.
Exemple	• `darker('200,10,30', 200)` → «100,5,15,255»

Mai multe detalii: *lighter*

lighter

Returnează o culoare, de tip șir, mai deschisă (sau mai închisă)

Sintaxă	lighter(color, factor)
Argumente	• **culoare** - culoarea sub formă de șir • **factor** - un număr întreg corespunzător factorului de iluminare: – dacă factorul este mai mare de 100, această funcție returnează o culoare mai deschisă (de ex.: setarea factorului la 150 returnează o culoare cu 50% mai strălucitoare); – dacă factorul este mai mic de 100, culoarea returnată este mai închisă; totuși, se recomandă utilizarea funcției darker() în acest scop; – dacă factorul este 0 sau negativ, valoarea returnată este nespecificată.
Exemple	• `lighter('200,10,30', 200)` → «255,158,168,255»

Mai multe detalii: *darker*

project_color

Returnează o culoare din schema de culori a proiectului.

Sintaxă	project_color(name)
Argumente	• **nume** - numele unei culori
Exemple	• `project_color('Logo color')` → «20,140,50»

Mai multe detalii: *setarea culorilor proiectului*

ramp_color

Returnează un șir reprezentând o culoare dintr-o rampă de culoare.

Varianta salvată a rampei

Returnează un șir reprezentând o culoare dintr-o rampă salvată

Sintaxă	ramp_color(ramp_name, value)
Argumente	• **ramp_name** - numele rampei de culoare, sub formă de șir, de ex.: „Spectral” • **value** - poziția de pe rampă, de unde se va selecta culoarea, ca număr real între 0 și 1
Exemple	• `ramp_color('Spectral',0.3)` → «253,190,115,255»

Notă: Rampele de culoare disponibile variază în diversele versiuni de QGIS. Este posibil ca această funcție să nu dea rezultatele așteptate, dacă deschideți proiectul în alte versiuni de QGIS.

Variantă de rampă creată folosind expresii

Returnează un șir care reprezintă o culoare dintr-o rampă creată printr-o expresie

Sintaxă	ramp_color(ramp, value)
Argumente	• **ramp** - rampa de culoare • **value** - poziția de pe rampă, de unde se va selecta culoarea, ca număr real între 0 și 1
Exemple	• `ramp_color(create_ramp(map(0,'0,0,0',1,'255,0,0')),1)` → «255,0,0,255»

Mai multe detalii: *Setting a Color Ramp*, *The color ramp drop-down shortcut*

set_color_part

Setează o componentă de culoare specifică pentru șirul unei culori, de ex.: componenta roșie sau componenta alfa.

Sintaxă	set_color_part(color, component, value)
Argumente	• **culoare** - culoarea sub formă de șir • **component** - un șir corespunzător componentei culorii de setat. Opțiunile valide sunt: – red: codul RGB al componentei roșii (0-255) – green: codul RGB al componentei verzi (0-255) – blue: codul RGB al componentei albastre (0-255) – alpha: valoarea alfa (transparența) (0-255) – hue: codul HSV al nuanței (0-360) – saturation: codul HSV al saturației (0-100) – value: codul HSV al valorii (0-100) – hsl_hue: codul HSL al nuanței (0-360) – hsl_saturation: codul HSL al saturației (0-100) – lightness: codul HSL al luminozității (0-100) – cyan: codul CMYK al componentei cyan (0-100) – magenta: codul CMYK al componentei magenta (0-100) – yellow: codul CMYK al componentei galbene (0-100) – black: codul CMYK al componentei negre (0-100) • **value** - noua valoare pentru componenta de culoare, respectând intervalele enumerate mai sus
Exemple	• `set_color_part('200,10,30','green',50)` → «200,50,30,255»

13.2.4 Funcții Condiționale

Acest grup conține funcții care gestionează verificările condiționale din expresii.

CASE

CASE se folosește pentru a evalua o serie de condiții și pentru a returna un rezultat pentru prima condiție îndeplinită. Condițiile sunt evaluate secvențial, iar dacă o condiție este adevărată, evaluarea se oprește iar rezultatul corespunzător este returnat. Dacă niciuna dintre condiții nu este adevărată, se returnează valoarea din clauza ELSE. În plus, dacă nu este setată nicio clauză ELSE și nici una dintre condiții nu este îndeplinită, se returnează valoarea NULL.

CASE

WHEN *condition* THEN *result*

[…n]

[ELSE *result*]

END

[ ] marchează componentele opționale

Argumente	• **WHEN condition** - Expresia condiției de evaluat • **THEN result** - Dacă *condiția* este evaluată la True, atunci este evaluat și returnat *rezultatul*. • **ELSE result** - Dacă niciuna dintre condițiile de mai sus nu este evaluată la True, atunci este evaluat și returnat *rezultatul*.
Exemple	• `CASE WHEN 'name' IS NULL THEN 'None' END` → Returnează șirul «None» dacă câmpul „nume” este NULL • `CASE WHEN $area > 10000 THEN 'Big property' WHEN $area > 5000 THEN 'Medium property' ELSE 'Small property' END` → Returnează șirul «Big property» dacă aria este mai mare de 10000, «Medium property» dacă aria este între 5000 și 10000 și «Small property» în rest

coalesce

Returnează prima valoare non-NULL din lista de expresii.

Această funcție poate prelua orice număr de argumente.

Sintaxă	coalesce(expression1, expression2, …)
Argumente	• **expresie** - orice expresie sau valoare validă, indiferent de tip.
Exemple	• `coalesce(NULL, 2)` → 2 • `coalesce(NULL, 2, 3)` → 2 • `coalesce(7, NULL, 3*2)` → 7 • `coalesce('fieldA', 'fallbackField', 'ERROR')` → valoarea câmpului A dacă nu este NULL, în caz contrar valoarea lui «fallbackField» sau șirul «ERROR» dacă ambele sunt NULL

if

Testează o condiție și returnează un rezultat diferit, în funcție de verificarea condiționată.

Sintaxă	if(condition, result_when_true, result_when_false)
Argumente	• **condition** - condiția care ar trebui verificată • **result_when_true** - rezultatul care va fi returnat atunci când condiția este adevărată sau o altă valoare care nu se poate converti în fals. • **result_when_false** - rezultatul care va fi returnat atunci când condiția este falsă sau o altă valoare care se transformă în fals, cum ar fi 0 sau «». NULL se va converti, de asemenea, în fals.
Exemple	• `if( 1+1=2, 'Yes', 'No' )` → «Yes» • `if( 1+1=3, 'Yes', 'No' )` → «No» • `if( 5 > 3, 1, 0)` → 1 • `if( '', 'It is true (not empty)', 'It is false (empty)' )` → «It is false (empty)» • `if( ' ', 'It is true (not empty)', 'It is false (empty)' )` → «It is true (not empty)» • `if( 0, 'One', 'Zero' )` → «Zero» • `if( 10, 'One', 'Zero' )` → «One»

nullif

Returnează o valoare NULL dacă valoarea1 este egală cu valoarea2; în caz contrar, returnează valoarea1. Se poate folosi pentru a înlocui condiționat valorile cu NULL.

Sintaxă	nullif(value1, value2)
Argumente	• **value1** - Valoarea care ar trebui fie folosită, fie înlocuită cu NULL. • **value2** - Valoarea de control care va declanșa substituirea cu NULL.
Exemple	• `nullif('(none)', '(none)')` → NULL • `nullif('text', '(none)')` → «text» • `nullif("name", '')` → NULL, dacă numele este un șir vid (sau deja NULL), iar numele în restul cazurilor.

regexp_match

Return the first matching position matching a regular expression within an unicode string, or 0 if the substring is not found.

Sintaxă	regexp_match(input_string, regex)
Argumente	• **input_string** - the string to test against the regular expression • **regex** - The regular expression to test against. Backslash characters must be double escaped (e.g., „\\s” to match a white space character or „\\b” to match a word boundary).
Exemple	• `regexp_match('QGIS ROCKS','\\sROCKS')` → 5 • `regexp_match('Budač','udač\\b')` → 2

try

Tries an expression and returns its value if error-free. If the expression returns an error, an alternative value will be returned when provided otherwise the function will return NULL.

Sintaxă	try(expression, [alternative]) [] marks optional arguments
Argumente	• **expression** - the expression which should be run • **alternative** - the result which will be returned if the expression returns an error.
Exemple	• `try( to_int( '1' ), 0 )` → 1 • `try( to_int( 'a' ), 0 )` → 0 • `try( to_date( 'invalid_date' ) )` → NULL

13.2.5 Conversions Functions

This group contains functions to convert one data type to another (e.g., string from/to integer, binary from/to string, string to date, …).

from_base64

Decodes a string in the Base64 encoding into a binary value.

Sintaxă	from_base64(string)
Argumente	• **string** - the string to decode
Exemple	• `from_base64('UUdJUw==')` → «QGIS»

hash

Creates a hash from a string with a given method. One byte (8 bits) is represented with two hex «»digits»», so «md4» (16 bytes) produces a 16 * 2 = 32 character long hex string and «keccak_512» (64 bytes) produces a 64 * 2 = 128 character long hex string.

Sintaxă	hash(string, method)
Argumente	• **string** - the string to hash • **method** - Metoda de codificare dintre «md4», «md5», «sha1», «sha224», «sha384», «sha512», «sha3_224», «sha3_256», «sha3_384», «sha3_512», «keccak_224», «keccak_256», «keccak_384», «keccak_512»
Exemple	• `hash('QGIS', 'md4')` → «c0fc71c241cdebb6e888cbac0e2b68eb» • `hash('QGIS', 'md5')` → «57470aaa9e22adaefac7f5f342f1c6da» • `hash('QGIS', 'sha1')` → «f87cfb2b74cdd5867db913237024e7001e62b114» • `hash('QGIS', 'sha224')` → «4093a619ada631c770f44bc643ead18fb393b93d6a6af1861fcfece0» • `hash('QGIS', 'sha256')` → «eb045cba7a797aaa06ac58830846e40c8e8c780bc0676d3393605fae50 • `hash('QGIS', 'sha384')` → «91c1de038cc3d09fdd512e99f9dd9922efadc39ed21d3922e69a4305cc2 • `hash('QGIS', 'sha512')` → «c2c092f2ab743bf8edbeb6d028a745f30fc720408465ed369421f0a4e20 • `hash('QGIS', 'sha3_224')` → «467f49a5039e7280d5d42fd433e80d203439e338eaabd701f0d6c17 • `hash('QGIS', 'sha3_256')` → «540f7354b6b8a6e735f2845250f15f4f3ba4f666c55574d9e935457 • `hash('QGIS', 'sha3_384')` → «96052da1e77679e9a65f60d7ead961b2879778231447863 86eb4364 • `hash('QGIS', 'sha3_512')` → «900d079dc69761da113980253aa8ac0414a8bd6d09879a916228f87 • `hash('QGIS', 'keccak_224')` → «5b0ce6acef8b0a121d4ac4f3eaa8503c799ad4e26a3392d1fb2014 • `hash('QGIS', 'keccak_256')` → «991c520aa6815392de24087f61b2ae0fd56abbfeee4a8ca019c101 • `hash('QGIS', 'keccak_384')` → «c57a3aed9d856fa04e5eeee9b62b6e027cca81ba574116d3cc1f0d

md5

Creează o codificare md5 dintr-un șir.

Sintaxă	md5(string)
Argumente	• **string** - the string to hash
Exemple	• `md5('QGIS')` → «57470aaa9e22adaefac7f5f342f1c6da»

sha256

Creează o codificare sha256 dintr-un șir.

Sintaxă	sha256(string)
Argumente	• **string** - the string to hash
Exemple	• `sha256('QGIS')` → «eb045cba7a797aaa06ac58830846e40c8e8c780bc0676d3393605fae50c05309»

to_base64

Codifică o valoare binară într-un șir, folosind Base64.

Sintaxă	to_base64(value)
Argumente	• **value** - valoarea binară de codificat
Exemple	• `to_base64('QGIS')` → «UUdJUw==»

to_date

Converts a string into a date object. An optional format string can be provided to parse the string; see QDate::fromString or the documentation of the format_date function for additional documentation on the format. By default the current QGIS user locale is used.

Sintaxă	to_date(string, [format], [language]) [] marks optional arguments
Argumente	• **string** - string representing a date value • **format** - format used to convert the string into a date • **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to convert the string into a date. By default the current QGIS user locale is used.
Exemple	• `to_date('2012-05-04')` → 2012-05-04 • `to_date('June 29, 2019','MMMM d, yyyy')` → 2019-06-29, if the current locale uses the name «June» for the sixth month, otherwise an error occurs • `to_date('29 juin, 2019','d MMMM, yyyy','fr')` → 2019-06-29

to_datetime

Converts a string into a datetime object. An optional format string can be provided to parse the string; see QDate::fromString, QTime::fromString or the documentation of the format_date function for additional documentation on the format. By default the current QGIS user locale is used.

Sintaxă	to_datetime(string, [format], [language]) [] marks optional arguments
Argumente	• **string** - string representing a datetime value • **format** - format used to convert the string into a datetime • **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to convert the string into a datetime. By default the current QGIS user locale is used.
Exemple	• `to_datetime('2012-05-04 12:50:00')` → 2012-05-04T12:50:00 • `to_datetime('June 29, 2019 @ 12:34','MMMM d, yyyy @ HH:mm')` → 2019-06-29T12:34, if the current locale uses the name «June» for the sixth month, otherwise an error occurs • `to_datetime('29 juin, 2019 @ 12:34','d MMMM, yyyy @ HH:mm', 'fr')` → 2019-06-29T12:34

to_decimal

Converts a degree, minute, second coordinate to its decimal equivalent.

Sintaxă	to_decimal(value)
Argumente	• **value** - A degree, minute, second string.
Exemple	• `to_decimal('6°21\'16.445')` → 6.3545680555

to_dm

Converts a coordinate to degree, minute.

Sintaxă	to_dm(coordinate, axis, precision, [formatting=]) [] marks optional arguments
Argumente	• **coordinate** - A latitude or longitude value. • **axis** - The axis of the coordinate. Either «x» or «y». • **precision** - Numărul de zecimale. • **formatting** - Designates the formatting type. Acceptable values are NULL (default), «aligned» or «suffix».
Exemple	• `to_dm(6.1545681, 'x', 3)` → 6°9.274′ • `to_dm(6.1545681, 'y', 4, 'aligned')` → 6°09.2741′N • `to_dm(6.1545681, 'y', 4, 'suffix')` → 6°9.2741′N

to_dms

Converts a coordinate to degree, minute, second.

Sintaxă	to_dms(coordinate, axis, precision, [formatting=]) [] marks optional arguments
Argumente	• **coordinate** - A latitude or longitude value. • **axis** - The axis of the coordinate. Either «x» or «y». • **precision** - Numărul de zecimale. • **formatting** - Designates the formatting type. Acceptable values are NULL (default), «aligned» or «suffix».
Exemple	• `to_dms(6.1545681, 'x', 3)` → 6°9′16.445″ • `to_dms(6.1545681, 'y', 4, 'aligned')` → 6°09′16.4452″N • `to_dms(6.1545681, 'y', 4, 'suffix')` → 6°9′16.4452″N

to_int

Converts a string to integer number. Nothing is returned if a value cannot be converted to integer (e.g «123asd» is invalid).

Sintaxă	to_int(string)
Argumente	• **string** - string to convert to integer number
Exemple	• `to_int('123')` → 123

to_interval

Converts a string to an interval type. Can be used to take days, hours, month, etc of a date.

Sintaxă	to_interval(string)
Argumente	• **string** - a string representing an interval. Allowable formats include {n} days {n} hours {n} months.
Exemple	• `to_interval('1 day 2 hours')` → interval: 1.08333 days • `to_interval( '0.5 hours' )` → interval: 30 minutes • `to_datetime('2012-05-05 12:00:00') - to_interval('1 day 2 hours')` → 2012-05-04T10:00:00

to_real

Converts a string to a real number. Nothing is returned if a value cannot be converted to real (e.g «123.56asd» is invalid). Numbers are rounded after saving changes if the precision is smaller than the result of the conversion.

Sintaxă	to_real(string)
Argumente	• **string** - string to convert to real number
Exemple	• `to_real('123.45')` → 123.45

to_string

Convertește un număr într-un șir.

Sintaxă	to_string(number)
Argumente	• **number** - Număr întreg sau valoare reală. Numărul de convertit în șir.
Exemple	• `to_string(123)` → «123»

to_time

Converts a string into a time object. An optional format string can be provided to parse the string; see QTime::fromString for additional documentation on the format.

Sintaxă	to_time(string, [format], [language]) [] marks optional arguments
Argumente	• **string** - string representing a time value • **format** - format used to convert the string into a time • **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to convert the string into a time
Exemple	• `to_time('12:30:01')` → 12:30:01 • `to_time('12:34','HH:mm')` → 12:34:00 • `to_time('12:34','HH:mm','fr')` → 12:34:00

13.2.6 Funcții Personalizate

This group contains functions created by the user. See *Editorul de Funcții* for more details.

13.2.7 Funcții pentru Dată și Oră

This group contains functions for handling date and time data. This group shares several functions with the *Conversions Functions* (to_date, to_time, to_datetime, to_interval) and *Funcții pentru Șiruri* (format_date) groups.

Notă: Storing date, datetime and intervals on fields

The ability to store *date*, *time* and *datetime* values directly on fields depends on the data source's provider (e.g., Shapefile accepts *date* format, but not *datetime* or *time* format). The following are some suggestions to overcome this limitation:

- *date*, *datetime* and *time* can be converted and stored in text type fields using the *format_date()* function.
- *Intervals* can be stored in integer or decimal type fields after using one of the date extraction functions (e.g., *day()* to get the interval expressed in days)

age

Returns the difference between two dates or datetimes.

The difference is returned as an `Interval` and needs to be used with one of the following functions in order to extract useful information:

- `year`
- `month`
- `week`
- `day`
- `hour`
- `minute`
- `second`

Sintaxă	age(datetime1, datetime2)
Argumente	• **datetime1** - un șir, dată sau datetime care reprezintă o dată ulterioară • **datetime2** - un șir, dată sau datetime care reprezintă data anterioară
Exemple	• `day(age('2012-05-12','2012-05-02'))` → 10 • `hour(age('2012-05-12','2012-05-02'))` → 240

datetime_from_epoch

Returns a datetime whose date and time are the number of milliseconds, msecs, that have passed since 1970-01-01T00:00:00.000, Coordinated Universal Time (Qt.UTC), and converted to Qt.LocalTime.

Sintaxă	datetime_from_epoch(int)
Argumente	• **int** - număr (milisecunde)
Exemple	• `datetime_from_epoch(1483225200000)` → 2017-01-01T00:00:00

day

Extrage ziua dintr-o dată sau numărul de zile dintr-un interval.

Date variant

Extracts the day from a date or datetime.

Sintaxă	day(date)
Argumente	• **date** - a date or datetime value
Exemple	• `day('2012-05-12')` → 12

Interval variant

Calculates the length in days of an interval.

Sintaxă	day(interval)
Argumente	• **interval** - interval value to return number of days from
Exemple	• `day(to_interval('3 days'))` → 3 • `day(to_interval('3 weeks 2 days'))` → 23 • `day(age('2012-01-01','2010-01-01'))` → 730

day_of_week

Returns the day of the week for a specified date or datetime. The returned value ranges from 0 to 6, where 0 corresponds to a Sunday and 6 to a Saturday.

Sintaxă	day_of_week(date)
Argumente	• **date** - date or datetime value
Exemple	• `day_of_week(to_date('2015-09-21'))` → 1

epoch

Returns the interval in milliseconds between the unix epoch and a given date value.

Sintaxă	epoch(date)
Argumente	• **date** - a date or datetime value
Exemple	• `epoch(to_date('2017-01-01'))` → 1483203600000

format_date

Formats a date type or string into a custom string format. Uses Qt date/time format strings. See QDateTime::toString.

Sintaxă	format_date(datetime, format, [language]) [] marks optional arguments
Argumente	• **datetime** - date, time or datetime value • **format** - String template used to format the string.

Expresie	Rezultat
d	ziua, un număr fără zero la început (de la 1 la 31)
dd	ziua, un număr cu zero la început (de la 01 la 31)
ddd	the abbreviated localized day name (e.g. «Mon» to «Sun»)
dddd	the long localized day name (e.g. «Monday» to «Sunday»)
M	the month as number without a leading zero (1-12)
MM	the month as number with a leading zero (01-12)
MMM	the abbreviated localized month name (e.g. «Jan» to «Dec»)
MMMM	the long localized month name (e.g. «January» to «December»)
yy	the year as two digit number (00-99)
yyyy	the year as four digit number

These expressions may be used for the time part of the format string:

Expresie	Rezultat
h	the hour without a leading zero (0 to 23 or 1 to 12 if AM/PM display)
hh	the hour with a leading zero (00 to 23 or 01 to 12 if AM/PM display)
H	the hour without a leading zero (0 to 23, even with AM/PM display)
HH	the hour with a leading zero (00 to 23, even with AM/PM display)
m	minutul, fără zero la început (de la 0 la 59)
mm	minutul, cu zero la început (de la 00 la 59)
s	secunda, fără zero la început (de la 0 la 59)
ss	secunda, cu un zero la început (de la 00 la 59)
z	the milliseconds without trailing zeroes (0 to 999)
zzz	the milliseconds with trailing zeroes (000 to 999)
AP or A	interpret as an AM/PM time. *AP* must be either «AM» or «PM».
ap or a	Interpret as an AM/PM time. *ap* must be either «am» or «pm».

- **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to format the date into a custom string. By default the current QGIS user locale is used.

Exemple

- `format_date('2012-05-15','dd.MM.yyyy')` → «15.05.2012»
- `format_date('2012-05-15','d MMMM yyyy','fr')` → «15 mai 2012»
- `format_date('2012-05-15','dddd')` → «Tuesday», if the current locale is an English variant
- `format_date('2012-05-15 13:54:20','dd.MM.yy')` → «15.05.12»
- `format_date('13:54:20','hh:mm AP')` → «01:54 PM»

hour

Extracts the hour part from a datetime or time, or the number of hours from an interval.

Time variant

Extracts the hour part from a time or datetime.

Sintaxă	hour(datetime)
Argumente	• **datetime** - a time or datetime value
Exemple	• `hour( to_datetime('2012-07-22 13:24:57') )` → 13

Interval variant

Calculates the length in hours of an interval.

Sintaxă	hour(interval)
Argumente	• **interval** - interval value to return number of hours from
Exemple	• `hour(to_interval('3 hours'))` → 3 • `hour(age('2012-07-22T13:00:00','2012-07-22T10:00:00'))` → 3 • `hour(age('2012-01-01','2010-01-01'))` → 17520

make_date

Creates a date value from year, month and day numbers.

Sintaxă	make_date(year, month, day)
Argumente	• **year** - Year number. Years 1 to 99 are interpreted as is. Year 0 is invalid. • **month** - Month number, where 1=January • **day** - Day number, beginning with 1 for the first day in the month
Exemple	• `make_date(2020,5,4)` → date value 2020-05-04

make_datetime

Creates a datetime value from year, month, day, hour, minute and second numbers.

Sintaxă	make_datetime(year, month, day, hour, minute, second)
Argumente	• **year** - Year number. Years 1 to 99 are interpreted as is. Year 0 is invalid. • **month** - Month number, where 1=January • **day** - Day number, beginning with 1 for the first day in the month • **hour** - Ora • **minute** - Minute • **second** - Secunde (valorile fracționale includ milisecunde)
Exemple	• `make_datetime(2020,5,4,13,45,30.5)` → datetime value 2020-05-04 13:45:30.500

make_interval

Creează o valoare de tip interval din valorile an, lună, săptămâni, zile, ore, minute și secunde.

Sintaxă	make_interval([years=0], [months=0], [weeks=0], [days=0], [hours=0], [minutes=0], [seconds=0]) [] marks optional arguments
Argumente	• **years** - Numărul de ani (presupunând o lungime a anului de 365,25 zile). • **months** - Number of months (assumes a 30 day month length) • **weeks** - Numărul de săptămâni • **days** - Numărul de zile • **hours** - Numărul de ore • **minutes** - Numărul de minute • **seconds** - Numărul de secunde
Exemple	• `make_interval(hours:=3)` → interval: 3 ore • `make_interval(days:=2, hours:=3)` → interval: 2.125 zile • `make_interval(minutes:=0.5, seconds:=5)` → interval: 35 secounde

make_time

Creates a time value from hour, minute and second numbers.

Sintaxă	make_time(hour, minute, second)
Argumente	• **hour** - Ora • **minute** - Minute • **second** - Secunde (valorile fracționale includ milisecunde)
Exemple	• `make_time(13,45,30.5)` → time value 13:45:30.500

minute

Extracts the minutes part from a datetime or time, or the number of minutes from an interval.

Time variant

Extracts the minutes part from a time or datetime.

Sintaxă	minute(datetime)
Argumente	• **datetime** - a time or datetime value
Exemple	• `minute( to_datetime('2012-07-22 13:24:57') )` → 24

Interval variant

Calculates the length in minutes of an interval.

Sintaxă	minute(interval)
Argumente	• **interval** - interval value to return number of minutes from
Exemple	• `minute(to_interval('3 minutes'))` → 3 • `minute(age('2012-07-22T00:20:00','2012-07-22T00:00:00'))` → 20 • `minute(age('2012-01-01','2010-01-01'))` → 1051200

month

Extracts the month part from a date, or the number of months from an interval.

Date variant

Extracts the month part from a date or datetime.

Sintaxă	month(date)
Argumente	• **date** - a date or datetime value
Exemple	• `month('2012-05-12')` → 05

Interval variant

Calculates the length in months of an interval.

Sintaxă	month(interval)
Argumente	• **interval** - interval value to return number of months from
Exemple	• `month(to_interval('3 months'))` → 3 • `month(age('2012-01-01','2010-01-01'))` → 4.03333

now

Returns the current date and time. The function is static and will return consistent results while evaluating. The time returned is the time when the expression is prepared.

Sintaxă	now()
Exemple	• `now()` → 2012-07-22T13:24:57

second

Extracts the seconds part from a datetime or time, or the number of seconds from an interval.

Time variant

Extracts the seconds part from a time or datetime.

Sintaxă	second(datetime)
Argumente	• **datetime** - a time or datetime value
Exemple	• `second( to_datetime('2012-07-22 13:24:57') )` → 57

Interval variant

Calculates the length in seconds of an interval.

Sintaxă	second(interval)
Argumente	• **interval** - interval value to return number of seconds from
Exemple	• `second(to_interval('3 minutes'))` → 180 • `second(age('2012-07-22T00:20:00','2012-07-22T00:00:00'))` → 1200 • `second(age('2012-01-01','2010-01-01'))` → 63072000

to_date

Converts a string into a date object. An optional format string can be provided to parse the string; see QDate::fromString or the documentation of the format_date function for additional documentation on the format. By default the current QGIS user locale is used.

Sintaxă	to_date(string, [format], [language]) [] marks optional arguments
Argumente	• **string** - string representing a date value • **format** - format used to convert the string into a date • **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to convert the string into a date. By default the current QGIS user locale is used.
Exemple	• `to_date('2012-05-04')` → 2012-05-04 • `to_date('June 29, 2019','MMMM d, yyyy')` → 2019-06-29, if the current locale uses the name «June» for the sixth month, otherwise an error occurs • `to_date('29 juin, 2019','d MMMM, yyyy','fr')` → 2019-06-29

to_datetime

Converts a string into a datetime object. An optional format string can be provided to parse the string; see QDate::fromString, QTime::fromString or the documentation of the format_date function for additional documentation on the format. By default the current QGIS user locale is used.

Sintaxă	to_datetime(string, [format], [language]) [] marks optional arguments
Argumente	• **string** - string representing a datetime value • **format** - format used to convert the string into a datetime • **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to convert the string into a datetime. By default the current QGIS user locale is used.
Exemple	• `to_datetime('2012-05-04 12:50:00')` → 2012-05-04T12:50:00 • `to_datetime('June 29, 2019 @ 12:34','MMMM d, yyyy @ HH:mm')` → 2019-06-29T12:34, if the current locale uses the name «June» for the sixth month, otherwise an error occurs • `to_datetime('29 juin, 2019 @ 12:34','d MMMM, yyyy @ HH:mm', 'fr')` → 2019-06-29T12:34

to_interval

Converts a string to an interval type. Can be used to take days, hours, month, etc of a date.

Sintaxă	to_interval(string)
Argumente	• **string** - a string representing an interval. Allowable formats include {n} days {n} hours {n} months.
Exemple	• `to_interval('1 day 2 hours')` → interval: 1.08333 days • `to_interval( '0.5 hours' )` → interval: 30 minutes • `to_datetime('2012-05-05 12:00:00') - to_interval('1 day 2 hours')` → 2012-05-04T10:00:00

to_time

Converts a string into a time object. An optional format string can be provided to parse the string; see QTime::fromString for additional documentation on the format.

Sintaxă	to_time(string, [format], [language]) [] marks optional arguments
Argumente	• **string** - string representing a time value • **format** - format used to convert the string into a time • **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to convert the string into a time
Exemple	• `to_time('12:30:01')` → 12:30:01 • `to_time('12:34','HH:mm')` → 12:34:00 • `to_time('12:34','HH:mm','fr')` → 12:34:00

week

Extracts the week number from a date, or the number of weeks from an interval.

Date variant

Extracts the week number from a date or datetime.

Sintaxă	week(date)
Argumente	• **date** - a date or datetime value
Exemple	• `week('2012-05-12')` → 19

Interval variant

Calculates the length in weeks of an interval.

Sintaxă	week(interval)
Argumente	• **interval** - interval value to return number of months from
Exemple	• `week(to_interval('3 weeks'))` → 3 • `week(age('2012-01-01','2010-01-01'))` → 104.285

year

Extracts the year part from a date, or the number of years from an interval.

Date variant

Extracts the year part from a date or datetime.

Sintaxă	year(date)
Argumente	• **date** - a date or datetime value
Exemple	• `year('2012-05-12')` → 2012

Interval variant

Calculates the length in years of an interval.

Sintaxă	year(interval)
Argumente	• **interval** - interval value to return number of years from
Exemple	• `year(to_interval('3 years'))` → 3 • `year(age('2012-01-01','2010-01-01'))` → 1.9986

Câteva exemple:

Besides these functions, subtracting dates, datetimes or times using the − (minus) operator will return an interval.

Adding or subtracting an interval to dates, datetimes or times, using the + (plus) and − (minus) operators, will return a datetime.

- Get the number of days until QGIS 3.0 release:

```
to_date('2017-09-29') - to_date(now())
-- Returns <interval: 203 days>
```

- The same with time:

```
to_datetime('2017-09-29 12:00:00') - now()
-- Returns <interval: 202.49 days>
```

- Get the datetime of 100 days from now:

```
now() + to_interval('100 days')
-- Returns <datetime: 2017-06-18 01:00:00>
```

13.2.8 Fields and Values

Contains a list of fields from the active layer, and special values. Fields list includes the ones stored in the dataset, *virtual* and *auxiliary* ones as well as from *joins*.

Double-click a field name to have it added to your expression. You can also type the field name (preferably inside double quotes) or its *alias*.

To retrieve fields values to use in an expression, select the appropriate field and, in the shown widget, choose between *10 Samples* and *All Unique*. Requested values are then displayed and you can use the *Search* box at the top of the list to filter the result. Sample values can also be accessed via right-clicking on a field.

To add a value to the expression you are writing, double-click on it in the list. If the value is of a string type, it should be simple quoted, otherwise no quote is needed.

NULL

Echivalează cu o valoare NULL.

Sintaxă	NULL
Exemple	• `NULL` → o valoare NULL

Notă: To test for NULL use an *IS NULL* or *IS NOT NULL* expression.

13.2.9 Files and Paths Functions

Acest grup conține funcții care manipulează numele fișierelor și căilor.

base_file_name

Returnează numele de bază al fișierului, fără directorul sau sufixul fișierului.

Sintaxă	base_file_name(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `base_file_name('/home/qgis/data/country_boundaries.shp')` → «country_boundaries»

exif

Preia valorile etichetelor exif dintr-un fișier imagine.

Sintaxă	exif(path, [tag]) [] marks optional arguments
Argumente	• **cale** - O cale a unui fișier de tip imagine sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci se va folosi sursa fișierului stratului. • **tag** - Eticheta de returnat. Dacă se lasă necompletat, atunci se va returna o hartă cu toate valorile etichetelor exif.
Exemple	• `exif('/my/photo.jpg','Exif.Image.Orientation')` → 0

file_exists

Returnează TRUE dacă există o cale de fișier.

Sintaxă	file_exists(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `file_exists('/home/qgis/data/country_boundaries.shp')` → TRUE

file_name

Returnează numele unui fișier (inclusiv extensia fișierului), excluzând directorul.

Sintaxă	file_name(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `file_name('/home/qgis/data/country_boundaries.shp')` → «country_boundaries.shp»

file_path

Returnează directorul, din compunerea unei căi de fișier. Nu se include și numele fișierului.

Sintaxă	file_path(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `file_path('/home/qgis/data/country_boundaries.shp')` → «/home/qgis/data»

file_size

Returnează dimensiunea (în octeți) a unui fișier.

Sintaxă	file_size(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `file_size('/home/qgis/data/country_boundaries.geojson')` → 5674

file_suffix

Returnează sufixul fișierului (extensia) dintr-o cale de fișier.

Sintaxă	file_suffix(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `file_suffix('/home/qgis/data/country_boundaries.shp')` → «shp»

is_directory

Returnează TRUE dacă o cale corespunde unui director.

Sintaxă	is_directory(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `is_directory('/home/qgis/data/country_boundaries.shp')` → FALSE • `is_directory('/home/qgis/data/')` → TRUE

is_file

Returns TRUE if a path corresponds to a file.

Sintaxă	is_file(path)
Argumente	• **path** - o cale de fișier sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci va fi folosită sursa fișierului stratului.
Exemple	• `is_file('/home/qgis/data/country_boundaries.shp')` → TRUE • `is_file('/home/qgis/data/')` → FALSE

13.2.10 Form Functions

This group contains functions that operate exclusively under the attribute form context. For example, in *field's widgets* settings.

current_parent_value

Only usable in an embedded form context, this function returns the current, unsaved value of a field in the parent form currently being edited. This will differ from the parent feature's actual attribute values for features which are currently being edited or have not yet been added to a parent layer. When used in a value-relation widget filter expression, this function should be wrapped into a «coalesce()» that can retrieve the actual parent feature from the layer when the form is not used in an embedded context.

Sintaxă	current_parent_value(field_name)
Argumente	• **field_name** - a field name in the current parent form
Exemple	• `current_parent_value( 'FIELD_NAME' )` → The current value of a field «FIELD_NAME» in the parent form.

current_value

Returns the current, unsaved value of a field in the form or table row currently being edited. This will differ from the feature's actual attribute values for features which are currently being edited or have not yet been added to a layer.

Sintaxă	current_value(field_name)
Argumente	• **field_name** - a field name in the current form or table row
Exemple	• `current_value( 'FIELD_NAME' )` → The current value of field «FIELD_NAME».

13.2.11 Funcții pentru Potrivirea Fuzzy

Acest grup conține funcții pentru comparații fuzzy între valori.

hamming_distance

Returns the Hamming distance between two strings. This equates to the number of characters at corresponding positions within the input strings where the characters are different. The input strings must be the same length, and the comparison is case-sensitive.

Sintaxă	hamming_distance(string1, string2)
Argumente	• **string1** - a string • **string2** - a string
Exemple	• `hamming_distance('abc','xec')` → 2 • `hamming_distance('abc','ABc')` → 2 • `hamming_distance(upper('abc'),upper('ABC'))` → 0 • `hamming_distance('abc','abcd')` → NULL

levenshtein

Returns the Levenshtein edit distance between two strings. This equates to the minimum number of character edits (insertions, deletions or substitutions) required to change one string to another.

The Levenshtein distance is a measure of the similarity between two strings. Smaller distances mean the strings are more similar, and larger distances indicate more different strings. The distance is case sensitive.

Sintaxă	levenshtein(string1, string2)
Argumente	• **string1** - a string • **string2** - a string
Exemple	• `levenshtein('kittens','mitten')` → 2 • `levenshtein('Kitten','kitten')` → 1 • `levenshtein(upper('Kitten'),upper('kitten'))` → 0

longest_common_substring

Returns the longest common substring between two strings. This substring is the longest string that is a substring of the two input strings. For example, the longest common substring of „ABABC” and „BABCA” is „BABC”. The substring is case sensitive.

Sintaxă	longest_common_substring(string1, string2)
Argumente	• **string1** - a string • **string2** - a string
Exemple	• `longest_common_substring('ABABC','BABCA')` → «BABC» • `longest_common_substring('abcDeF','abcdef')` → «abc» • `longest_common_substring(upper('abcDeF'),upper('abcdex'))` → «ABCDE»

soundex

Returns the Soundex representation of a string. Soundex is a phonetic matching algorithm, so strings with similar sounds should be represented by the same Soundex code.

Sintaxă	soundex(string)
Argumente	• **string** - a string
Exemple	• `soundex('robert')` → «R163» • `soundex('rupert')` → «R163» • `soundex('rubin')` → «R150»

13.2.12 Funcții Generale

This group contains general assorted functions.

env

Gets an environment variable and returns its content as a string. If the variable is not found, NULL will be returned. This is handy to inject system specific configuration like drive letters or path prefixes. Definition of environment variables depends on the operating system, please check with your system administrator or the operating system documentation how this can be set.

Sintaxă	env(name)
Argumente	• **name** - The name of the environment variable which should be retrieved.
Exemple	• `env( 'LANG' )` → «en_US.UTF-8» • `env( 'MY_OWN_PREFIX_VAR' )` → «Z:» • `env( 'I_DO_NOT_EXIST' )` → NULL

eval

Evaluates an expression which is passed in a string. Useful to expand dynamic parameters passed as context variables or fields.

Sintaxă	eval(expression)
Argumente	• **expression** - an expression string
Exemple	• `eval('\'nice\'')` → «nice» • `eval(@expression_var)` → [whatever the result of evaluating @expression_var might be…]

eval_template

Evaluates a template which is passed in a string. Useful to expand dynamic parameters passed as context variables or fields.

Sintaxă	eval_template(template)
Argumente	• **template** - a template string
Exemple	• `eval_template('QGIS [% upper(\'rocks\') %]')` → QGIS ROCKS

is_layer_visible

Returns TRUE if a specified layer is visible.

Sintaxă	is_layer_visible(layer)
Argumente	• **layer** - a string, representing either a layer name or layer ID
Exemple	• `is_layer_visible('baseraster')` → TRUE

mime_type

Returns the mime type of the binary data.

Sintaxă	mime_type(bytes)
Argumente	• **bytes** - the binary data
Exemple	• `mime_type('<html><body></body></html>')` → text/html • `mime_type(from_base64('R0lGODlhAQABAAAAACH5BAEKAAEALAAAAAABAAEAAAIAOw==` → image/gif

var

Returns the value stored within a specified variable.

Sintaxă	var(name)
Argumente	• **name** - a variable name
Exemple	• `var('qgis_version')` → «2.12»

Mai multe detalii: Lista variabilelor implicite :ref:` <expression_variables>`

with_variable

This function sets a variable for any expression code that will be provided as 3rd argument. This is only useful for complicated expressions, where the same calculated value needs to be used in different places.

Sintaxă	with_variable(name, value, expression)
Argumente	• **name** - the name of the variable to set • **value** - the value to set • **expression** - the expression for which the variable will be available
Exemple	• `with_variable('my_sum', 1 + 2 + 3, @my_sum * 2 + @my_sum * 5)` → 42

13.2.13 Funcții Geometrice

This group contains functions that operate on geometry objects (e.g. buffer, transform, $area).

affine_transform

Returns the geometry after an affine transformation. Calculations are in the Spatial Reference System of this geometry. The operations are performed in a scale, rotation, translation order. If there is a Z or M offset but the coordinate is not present in the geometry, it will be added.

Sintaxă	affine_transform(geometry, delta_x, delta_y, rotation_z, scale_x, scale_y, [delta_z=0], [delta_m=0], [scale_z=1], [scale_m=1]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **delta_x** - x-axis translation • **delta_y** - y-axis translation • **rotation_z** - rotation around z-axis in degrees counter-clockwise • **scale_x** - x-axis scale factor • **scale_y** - y-axis scale factor • **delta_z** - z-axis translation • **delta_m** - m-axis translation • **scale_z** - z-axis scale factor • **scale_m** - m-axis scale factor
Exemple	• `geom_to_wkt(affine_transform(geom_from_wkt('LINESTRING(1 1, 2 2)'), 2, 2, 0, 1, 1))` → «LineString (3 3, 4 4)» • `geom_to_wkt(affine_transform(geom_from_wkt('POLYGON((0 0, 0 3, 2 2, 0 0))'), 0, 0, -90, 1, 2))` → «Polygon ((0 0, 6 0, 4 -2, 0 0))» • `geom_to_wkt(affine_transform(geom_from_wkt('POINT(3 1)'), 0, 0, 0, 1, 1, 5, 0))` → «PointZ (3 1 5)»

angle_at_vertex

Returns the bisector angle (average angle) to the geometry for a specified vertex on a linestring geometry. Angles are in degrees clockwise from north.

Sintaxă	angle_at_vertex(geometry, vertex)
Argumente	• **geometry** - a linestring geometry • **vertex** - vertex index, starting from 0; if the value is negative, the selected vertex index will be its total count minus the absolute value
Exemple	• `angle_at_vertex(geometry:=geom_from_wkt('LineString(0 0, 10 0, 10 10)'),vertex:=1)` → 45.0

apply_dash_pattern

Applies a dash pattern to a geometry, returning a MultiLineString geometry which is the input geometry stroked along each line/ring with the specified pattern.

Sintaxă	apply_dash_pattern(geometry, pattern, [start_rule=no_rule], [end_rule=no_rule], [adjustment=both], [pattern_offset=0]) [] marks optional arguments
Argumente	• **geometry** - a geometry (accepts (multi)linestrings or (multi)polygons). • **pattern** - dash pattern, as an array of numbers representing dash and gap lengths. Must contain an even number of elements. • **start_rule** - optional rule for constraining the start of the pattern. Valid values are «no_rule», «full_dash», «half_dash», «full_gap», «half_gap». • **end_rule** - optional rule for constraining the end of the pattern. Valid values are «no_rule», «full_dash», «half_dash», «full_gap», «half_gap». • **adjustment** - optional rule for specifying which part of patterns are adjusted to fit the desired pattern rules. Valid values are «both», «dash», «gap». • **pattern_offset** - Optional distance specifying a specific distance along the pattern to commence at.
Exemple	• `geom_to_wkt(apply_dash_pattern(geom_from_wkt('LINESTRING(1 1, 10 1)'), array(3, 1)))` → MultiLineString ((1 1, 4 1),(5 1, 8 1),(9 1, 10 1, 10 1)) • `geom_to_wkt(apply_dash_pattern(geom_from_wkt('LINESTRING(1 1, 10 1)'), array(3, 1), start_rule:='half_dash'))` → MultiLineString ((1 1, 2.5 1),(3.5 1, 6.5 1),(7.5 1, 10 1, 10 1))

$area

Returns the area of the current feature. The area calculated by this function respects both the current project's ellipsoid setting and area unit settings. For example, if an ellipsoid has been set for the project then the calculated area will be ellipsoidal, and if no ellipsoid is set then the calculated area will be planimetric.

Sintaxă	$area
Exemple	• `$area` → 42

area

Returns the area of a geometry polygon object. Calculations are always planimetric in the Spatial Reference System (SRS) of this geometry, and the units of the returned area will match the units for the SRS. This differs from the calculations performed by the $area function, which will perform ellipsoidal calculations based on the project's ellipsoid and area unit settings.

Sintaxă	area(geometry)
Argumente	• **geometry** - polygon geometry object
Exemple	• `area(geom_from_wkt('POLYGON((0 0, 4 0, 4 2, 0 2, 0 0))'))` → 8.0

azimuth

Returns the north-based azimuth as the angle in radians measured clockwise from the vertical on point_a to point_b.

Sintaxă	azimuth(point_a, point_b)
Argumente	• **point_a** - point geometry • **point_b** - point geometry
Exemple	• `degrees( azimuth( make_point(25, 45), make_point(75, 100) ) )` → 42.273689 • `degrees( azimuth( make_point(75, 100), make_point(25,45) ) )` → 222.273689

bearing

Returns the north-based bearing as the angle in radians measured clockwise on the ellipsoid from the vertical on point_a to point_b.

Sintaxă	bearing(point_a, point_b, [source_crs], [ellipsoid]) [] marks optional arguments
Argumente	• **point_a** - point geometry • **point_b** - point geometry • **source_crs** - an optional string representing the source CRS of the points. By default the current layer's CRS is used. • **ellipsoid** - an optional string representing the acronym or the authority:ID (eg «EPSG:7030») of the ellipsoid on which the bearing should be measured. By default the current project's ellipsoid setting is used.
Exemple	• `degrees( bearing( make_point(16198544, -4534850), make_point(18736872, -1877769), 'EPSG:3857', 'EPSG:7030') )` → 49.980071 • `degrees( bearing( make_point(18736872, -1877769), make_point(16198544, -4534850), 'EPSG:3857', 'WGS84') )` → 219.282386

boundary

Returns the closure of the combinatorial boundary of the geometry (ie the topological boundary of the geometry). For instance, a polygon geometry will have a boundary consisting of the linestrings for each ring in the polygon. Some geometry types do not have a defined boundary, e.g., points or geometry collections, and will return NULL.

Sintaxă	boundary(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `geom_to_wkt(boundary(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')))` → «LineString(1 1,0 0,-1 1,1 1)» • `geom_to_wkt(boundary(geom_from_wkt('LineString(1 1,0 0,-1 1)')))` → «MultiPoint ((1 1),(-1 1))»

Mai multe detalii: *Boundary* algorithm

bounds

Returns a geometry which represents the bounding box of an input geometry. Calculations are in the Spatial Reference System of this geometry.

Sintaxă	bounds(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `bounds(@geometry)` → bounding box of the current feature's geometry • `geom_to_wkt(bounds(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')))` → «Polygon ((-1 0, 1 0, 1 1, -1 1, -1 0))»

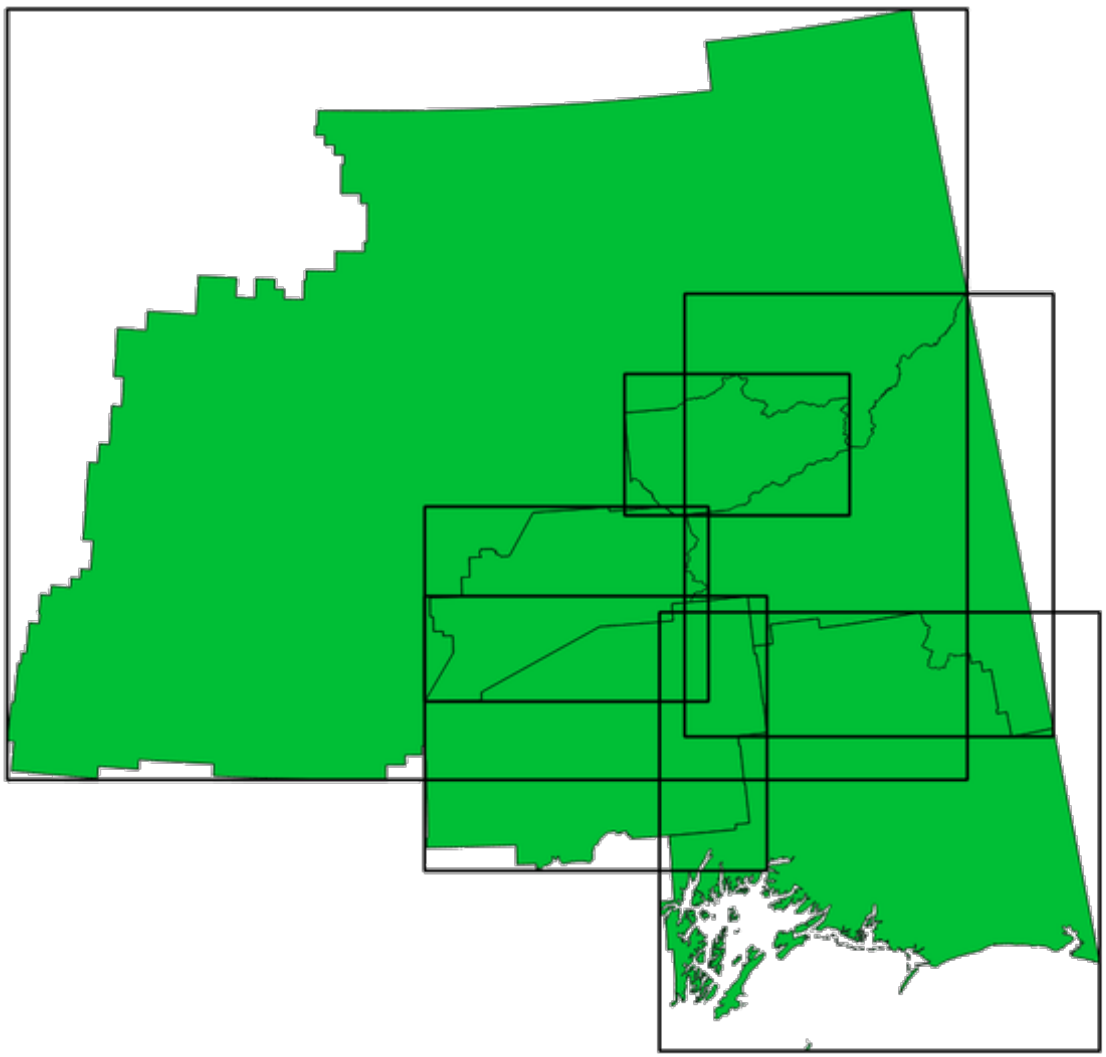


Fig. 13.4: Black lines represent the bounding boxes of each polygon feature

Mai multe detalii: *Bounding boxes* algorithm

bounds_height

Returns the height of the bounding box of a geometry. Calculations are in the Spatial Reference System of this geometry.

Sintaxă	bounds_height(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `bounds_height(@geometry)` → height of bounding box of the current feature's geometry • `bounds_height(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))'))` → 1

bounds_width

Returns the width of the bounding box of a geometry. Calculations are in the Spatial Reference System of this geometry.

Sintaxă	bounds_width(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `bounds_width(@geometry)` → width of bounding box of the current feature's geometry • `bounds_width(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))'))` → 2

buffer

Returns a geometry that represents all points whose distance from this geometry is less than or equal to distance. Calculations are in the Spatial Reference System of this geometry.

Sintaxă	buffer(geometry, distance, [segments=8], [cap=»round»], [join=»round»], [miter_limit=2]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **distance** - buffer distance in layer units • **segments** - number of segments to use to represent a quarter circle when a round join style is used. A larger number results in a smoother buffer with more nodes. • **cap** - end cap style for buffer. Valid values are «round», «flat» or «square» • **join** - join style for buffer. Valid values are «round», «bevel» or «miter». • **miter_limit** - miter distance limit, for use when the join style is set to «miter»
Exemple	• `buffer(@geometry, 10.5)` → polygon of the current feature's geometry buffered by 10.5 units

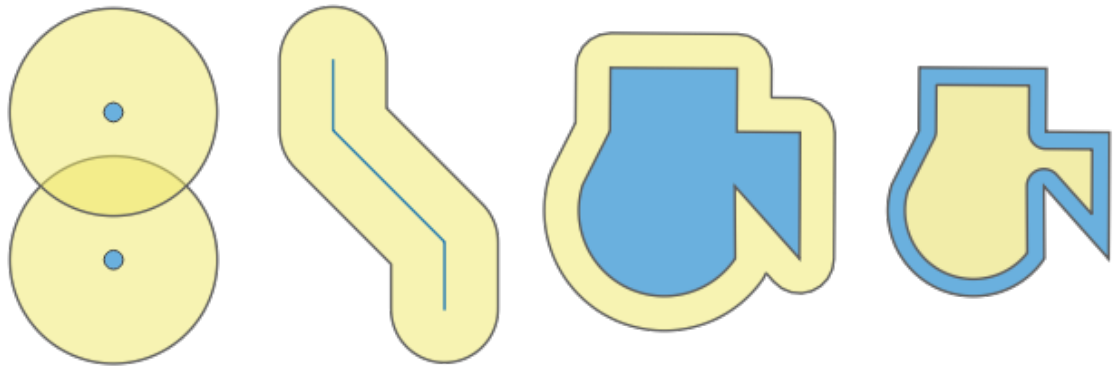


Fig. 13.5: Buffer (in yellow) of points, line, polygon with positive buffer, and polygon with negative buffer

Mai multe detalii: *Buffer* algorithm

buffer_by_m

Creates a buffer along a line geometry where the buffer diameter varies according to the m-values at the line vertices.

Sintaxă	buffer_by_m(geometry, [segments=8]) [] marks optional arguments
Argumente	• **geometry** - input geometry. Must be a (multi)line geometry with m values. • **segments** - number of segments to approximate quarter-circle curves in the buffer.
Exemple	• `buffer_by_m(geometry:=geom_from_wkt('LINESTRINGM(1 2 0.5, 4 2 0.2)'),segments:=8)` → A variable width buffer starting with a diameter of 0.5 and ending with a diameter of 0.2 along the linestring geometry.

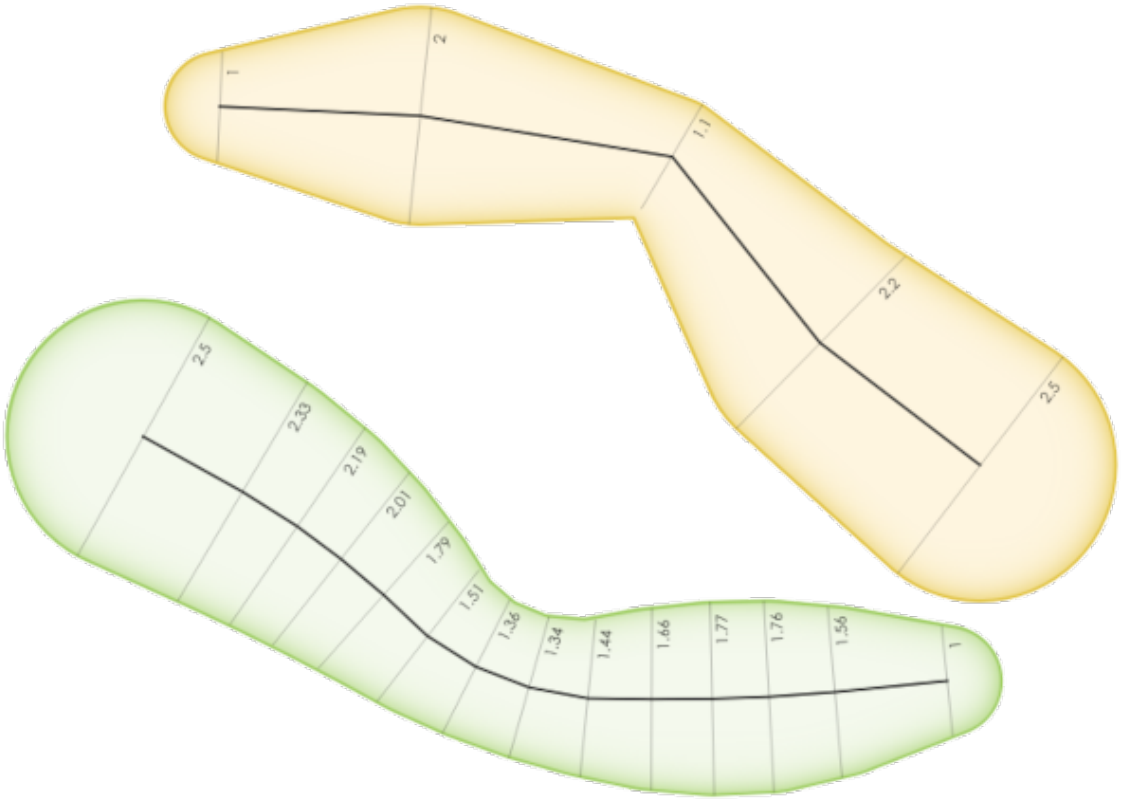


Fig. 13.6: Buffering line features using the m value on the vertices

Mai multe detalii: Algoritmul *Variable width buffer (by M value)*

centroid

Returns the geometric center of a geometry.

Sintaxă	centroid(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `centroid(@geometry)` → a point geometry

Mai multe detalii: Algoritmul *Centroids*

close_line

Returns a closed line string of the input line string by appending the first point to the end of the line, if it is not already closed. If the geometry is not a line string or multi line string then the result will be NULL.

Sintaxă	close_line(geometry)
Argumente	• **geometry** - a line string geometry
Exemple	• `geom_to_wkt(close_line(geom_from_wkt('LINESTRING(0 0, 1 0, 1 1)')))` → «LineString (0 0, 1 0, 1 1, 0 0)» • `geom_to_wkt(close_line(geom_from_wkt('LINESTRING(0 0, 1 0, 1 1, 0 0)')))` → «LineString (0 0, 1 0, 1 1, 0 0)»

closest_point

Returns the point on geometry1 that is closest to geometry2.

Sintaxă	closest_point(geometry1, geometry2)
Argumente	• **geometry1** - geometry to find closest point on • **geometry2** - geometry to find closest point to
Exemple	• `geom_to_wkt(closest_point(geom_from_wkt('LINESTRING (20 80, 98 190, 110 180, 50 75 )'),geom_from_wkt('POINT(100 100)')))` → «Point(73.0769 115.384)»

collect_geometries

Collects a set of geometries into a multi-part geometry object.

List of arguments variant

Geometry parts are specified as separate arguments to the function.

Sintaxă	collect_geometries(geometry1, geometry2, …)
Argumente	• **geometry** - a geometry
Exemple	• `geom_to_wkt(collect_geometries(make_point(1,2), make_point(3,4), make_point(5,6)))` → «MultiPoint ((1 2),(3 4),(5 6))»

Array variant

Geometry parts are specified as an array of geometry parts.

Sintaxă	collect_geometries(array)
Argumente	• **array** - array of geometry objects
Exemple	• `geom_to_wkt(collect_geometries(array(make_point(1,2), make_point(3,4), make_point(5,6))))` → «MultiPoint ((1 2),(3 4),(5 6))»

Mai multe detalii: Algoritmul *Collect geometries*

combine

Returns the combination of two geometries.

Sintaxă	combine(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `geom_to_wkt( combine( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 2 1)' ) ) )` → «MULTILINESTRING((4 4, 2 1), (3 3, 4 4), (4 4, 5 5))» • `geom_to_wkt( combine( geom_from_wkt( 'LINESTRING(3 3, 4 4)' ), geom_from_wkt( 'LINESTRING(3 3, 6 6, 2 1)' ) ) )` → «LINESTRING(3 3, 4 4, 6 6, 2 1)»

concave_hull

Returns a possibly concave polygon that contains all the points in the geometry

Sintaxă	concave_hull(geometry, target_percent, [allow_holes=False]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **target_percent** - the percentage of area of the convex hull the solution tries to approach. A target_percent of 1 gives the same result as the convex hull. A target_percent between 0 and 0.99 produces a result that should have a smaller area than the convex hull. • **allow_holes** - optional argument specifying whether to allow holes within the output geometry. Defaults to FALSE, set to TRUE to avoid including holes in the output geometry.
Exemple	• `geom_to_wkt(concave_hull(geom_from_wkt('MULTILINESTRING((106 164,30 112,74 70,82 112,130 94,130 62,122 40,156 32,162 76,172 88),(132 178,134 148,128 136,96 128,132 108,150 130,170 142,174 110,156 96,158 90,158 88),(22 64,66 28,94 38,94 68,114 76,112 30,132 10,168 18,178 34,186 52,184 74,190 100,190 122,182 148,178 170,176 184,156 164,146 178,132 186,92 182,56 158,36 150,62 150,76 128,88 118))'), 0.99))` → «Polygon ((30 112, 36 150, 92 182, 132 186, 176 184, 190 122, 190 100, 186 52, 178 34, 168 18, 132 10, 112 30, 66 28, 22 64, 30 112))»

Mai multe detalii: *convex_hull*

contains

Tests whether a geometry contains another. Returns TRUE if and only if no points of geometry2 lie in the exterior of geometry1, and at least one point of the interior of geometry2 lies in the interior of geometry1.

Sintaxă	contains(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `contains( geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))' ), geom_from_wkt( 'POINT(0.5 0.5 )' ) )` → TRUE • `contains( geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → FALSE

Mai multe detalii: *overlay_contains*

convex_hull

Returns the convex hull of a geometry. It represents the minimum convex geometry that encloses all geometries within the set.

Sintaxă	convex_hull(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `geom_to_wkt( convex_hull( geom_from_wkt( 'LINESTRING(3 3, 4 4, 4 10)' ) ) )` → «POLYGON((3 3, 4 10, 4 4, 3 3))»

Mai multe detalii: *concave_hull*, *Convex hull* algorithm

crosses

Tests whether a geometry crosses another. Returns TRUE if the supplied geometries have some, but not all, interior points in common.

Sintaxă	crosses(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `crosses( geom_from_wkt( 'LINESTRING(3 5, 4 4, 5 3)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `crosses( geom_from_wkt( 'POINT(4 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → FALSE

Mai multe detalii: *overlay_crosses*

densify_by_count

Takes a polygon or line layer geometry and generates a new one in which the geometries have a larger number of vertices than the original one.

Sintaxă	densify_by_count(geometry, vertices)
Argumente	• **geometry** - a geometry (accepts (multi)linestrings or (multi)polygons). • **vertices** - number of vertices to add (per segment)
Exemple	• `geom_to_wkt(densify_by_count(geom_from_wkt('LINESTRING(1 1, 10 1)'), 3))` → LineString (1 1, 3.25 1, 5.5 1, 7.75 1, 10 1)

Mai multe detalii: *Densify by count* algorithm

densify_by_distance

Takes a polygon or line layer geometry and generates a new one in which the geometries are densified by adding additional vertices on edges that have a maximum distance of the specified interval distance.

Sintaxă	densify_by_distance(geometry, distance)
Argumente	• **geometry** - a geometry (accepts (multi)linestrings or (multi)polygons). • **distance** - maximum interval distance between vertices in output geometry
Exemple	• `geom_to_wkt(densify_by_distance(geom_from_wkt('LINESTRING(1 1, 10 1)'), 4))` → LineString (1 1, 4 1, 7 1, 10 1)

Mai multe detalii: *Densify by interval* algorithm

difference

Returns a geometry that represents that part of geometry1 that does not intersect with geometry2.

Sintaxă	difference(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `geom_to_wkt( difference( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4)' ) ) )` → «LINESTRING(4 4, 5 5)»

Mai multe detalii: Algoritmul *Difference*

disjoint

Tests whether geometries do not spatially intersect. Returns TRUE if the geometries do not share any space together.

Sintaxă	disjoint(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `disjoint( geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0 ))' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `disjoint( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'POINT(4 4)' ))` → FALSE

Mai multe detalii: *overlay_disjoint*

distance

Returns the minimum distance (based on spatial reference) between two geometries in projected units.

Sintaxă	distance(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `distance( geom_from_wkt( 'POINT(4 4)' ), geom_from_wkt( 'POINT(4 8)' ) )` → 4

distance_to_vertex

Returns the distance along the geometry to a specified vertex.

Sintaxă	distance_to_vertex(geometry, vertex)
Argumente	• **geometry** - a linestring geometry • **vertex** - vertex index, starting from 0; if the value is negative, the selected vertex index will be its total count minus the absolute value
Exemple	• `distance_to_vertex(geometry:=geom_from_wkt('LineString(0 0, 10 0, 10 10)'),vertex:=1)` → 10.0

end_point

Returns the last node from a geometry.

Sintaxă	end_point(geometry)
Argumente	• **geometry** - geometry object
Exemple	• `geom_to_wkt(end_point(geom_from_wkt('LINESTRING(4 0, 4 2, 0 2)')))` → «Point (0 2)»

Mai multe detalii: Algoritmul *Extract specific vertices*

exif_geotag

Creates a point geometry from the exif geotags of an image file.

Sintaxă	exif_geotag(path)
Argumente	• **cale** - O cale a unui fișier de tip imagine sau o valoare a unui strat de hartă. Dacă este specificată o valoare a unui strat de hartă, atunci se va folosi sursa fișierului stratului.
Exemple	• `geom_to_wkt(exif_geotag('/my/photo.jpg'))` → «Point (2 4)»

extend

Extends the start and end of a linestring geometry by a specified amount. Lines are extended using the bearing of the first and last segment in the line. For a multilinestring, all the parts are extended. Distances are in the Spatial Reference System of this geometry.

Sintaxă	extend(geometry, start_distance, end_distance)
Argumente	• **geometry** - a (multi)linestring geometry • **start_distance** - distance to extend the start of the line • **end_distance** - distance to extend the end of the line.
Exemple	• `geom_to_wkt(extend(geom_from_wkt('LineString(0 0, 1 0, 1 1)'),1,2))` → «LineString (-1 0, 1 0, 1 3)» • `geom_to_wkt(extend(geom_from_wkt('MultiLineString((0 0, 1 0, 1 1), (2 2, 0 2, 0 5))'),1,2))` → «MultiLineString ((-1 0, 1 0, 1 3),(3 2, 0 2, 0 7))»

Mai multe detalii: Algoritmul *Extend lines*

exterior_ring

Returns a line string representing the exterior ring of a polygon geometry. If the geometry is not a polygon then the result will be NULL.

Sintaxă	exterior_ring(geometry)
Argumente	• **geometry** - a polygon geometry
Exemple	• `geom_to_wkt(exterior_ring(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1),( 0.1 0.1, 0.1 0.2, 0.2 0.2, 0.2, 0.1, 0.1 0.1))')))` → «LineString (-1 -1, 4 0, 4 2, 0 2, -1 -1)»

extrude

Returns an extruded version of the input (Multi-)Curve or (Multi-)Linestring geometry with an extension specified by x and y.

Sintaxă	extrude(geometry, x, y)
Argumente	• **geometry** - a curve or linestring geometry • **x** - x extension, numeric value • **y** - y extension, numeric value
Exemple	• `geom_to_wkt(extrude(geom_from_wkt('LineString(1 2, 3 2, 4 3)'), 1, 2))` → «Polygon ((1 2, 3 2, 4 3, 5 5, 4 4, 2 4, 1 2))» • `geom_to_wkt(extrude(geom_from_wkt('MultiLineString((1 2, 3 2), (4 3, 8 3))'), 1, 2))` → «MultiPolygon (((1 2, 3 2, 4 4, 2 4, 1 2)),((4 3, 8 3, 9 5, 5 5, 4 3)))»

flip_coordinates

Returns a copy of the geometry with the x and y coordinates swapped. Useful for repairing geometries which have had their latitude and longitude values reversed.

Sintaxă	flip_coordinates(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `geom_to_wkt(flip_coordinates(make_point(1, 2)))` → «Point (2 1)» • `geom_to_wkt(flip_coordinates(geom_from_wkt('LineString(0 2, 1 0, 1 6)')))` → «LineString (2 0, 0 1, 6 1)»

Mai multe detalii: Algoritmul *Swap X and Y coordinates*

force_polygon_ccw

Forces a geometry to respect the convention where exterior rings are counter-clockwise, interior rings are clockwise.

Sintaxă	force_polygon_ccw(geometry)
Argumente	• **geometry** - a geometry. Any non-polygon geometries are returned unchanged.
Exemple	• `geom_to_wkt(force_polygon_ccw(geometry:=geom_from_wkt('Polygon ((-1 -1, 0 2, 4 2, 4 0, -1 -1))')))` → «Polygon ((-1 -1, 4 0, 4 2, 0 2, -1 -1))»

Mai multe detalii: *force_polygon_cw*, *force_rhr*

force_polygon_cw

Forces a geometry to respect the convention where exterior rings are clockwise, interior rings are counter-clockwise.

Sintaxă	force_polygon_cw(geometry)
Argumente	• **geometry** - a geometry. Any non-polygon geometries are returned unchanged.
Exemple	• `geom_to_wkt(force_polygon_cw(geometry:=geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))')))` → «Polygon ((-1 -1, 0 2, 4 2, 4 0, -1 -1))»

Mai multe detalii: *force_polygon_ccw*, *force_rhr*

force_rhr

Forces a geometry to respect the Right-Hand-Rule, in which the area that is bounded by a polygon is to the right of the boundary. In particular, the exterior ring is oriented in a clockwise direction and the interior rings in a counter-clockwise direction. Due to the inconsistency in the definition of the Right-Hand-Rule in some contexts it is recommended to use the explicit force_polygon_cw function instead.

Sintaxă	force_rhr(geometry)
Argumente	• **geometry** - a geometry. Any non-polygon geometries are returned unchanged.
Exemple	• `geom_to_wkt(force_rhr(geometry:=geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))')))` → «Polygon ((-1 -1, 0 2, 4 2, 4 0, -1 -1))»

Mai multe detalii: *Force right-hand-rule* algorithm, *force_polygon_ccw*, *force_polygon_cw*

geom_from_gml

Returns a geometry from a GML representation of geometry.

Sintaxă	geom_from_gml(gml)
Argumente	• **gml** - GML representation of a geometry as a string
Exemple	• `geom_from_gml('<gml:LineString srsName="EPSG:4326"><gml:coordinates>4,4 5,5 6,6</gml:coordinates></gml:LineString>')` → a line geometry object

geom_from_wkb

Returns a geometry created from a Well-Known Binary (WKB) representation.

Sintaxă	geom_from_wkb(binary)
Argumente	• **binary** - Well-Known Binary (WKB) representation of a geometry (as a binary blob)
Exemple	• `geom_from_wkb( geom_to_wkb( make_point(4,5) ) )` → a point geometry object

geom_from_wkt

Returns a geometry created from a Well-Known Text (WKT) representation.

Sintaxă	geom_from_wkt(text)
Argumente	• **text** - Well-Known Text (WKT) representation of a geometry
Exemple	• `geom_from_wkt( 'POINT(4 5)' )` → a geometry object

geom_to_wkb

Returns the Well-Known Binary (WKB) representation of a geometry

Sintaxă	geom_to_wkb(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `geom_to_wkb( @geometry )` → binary blob containing a geometry object

geom_to_wkt

Returns the Well-Known Text (WKT) representation of the geometry without SRID metadata.

Sintaxă	geom_to_wkt(geometry, [precision=8]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **precision** - numeric precision
Exemple	• `geom_to_wkt( make_point(6, 50) )` → «POINT(6 50)» • `geom_to_wkt(centroid(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')))` → «POINT(0 0.66666667)» • `geom_to_wkt(centroid(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')), 2)` → «POINT(0 0.67)»

$geometry

Returns the geometry of the current feature. Can be used for processing with other functions. **WARNING: This function is deprecated. It is recommended to use the replacement @geometry variable instead.**

Sintaxă	$geometry
Exemple	• `geom_to_wkt( $geometry )` → «POINT(6 50)»

geometry

Returns a feature's geometry.

Sintaxă	geometry(feature)
Argumente	• **feature** - a feature object
Exemple	• `geometry( @feature )` → the geometry of the current feature. Prefer using @geometry. • `geom_to_wkt( geometry( get_feature_by_id( 'streets', 1 ) ) )` → the geometry in WKT of the feature with the id 1 on the layer „streets", e.g. «POINT(6 50)» • `intersects( @geometry, geometry( get_feature( 'streets', 'name', 'Main St.' ) ) )` → TRUE if the current feature spatially intersects the «Main St.» named feature in the „streets" layer

geometry_n

Returns a specific geometry from a geometry collection, or NULL if the input geometry is not a collection. Also returns a part from a multipart geometry.

Sintaxă	geometry_n(geometry, index)
Argumente	• **geometry** - geometry collection • **index** - index of geometry to return, where 1 is the first geometry in the collection
Exemple	• `geom_to_wkt(geometry_n(geom_from_wkt('GEOMETRYCOLLECTION(POINT(0 1), POINT(0 0), POINT(1 0), POINT(1 1))'),3))` → «Point (1 0)»

geometry_type

Returns a string value describing the type of a geometry (Point, Line or Polygon)

Sintaxă	geometry_type(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `geometry_type( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → «Line» • `geometry_type( geom_from_wkt( 'MULTILINESTRING((2 5, 3 6, 4 8), (1 1, 0 0))') )` → «Line» • `geometry_type( geom_from_wkt( 'POINT(2 5)') )` → «Point» • `geometry_type( geom_from_wkt( 'POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))') )` → «Polygon»

hausdorff_distance

Returns the Hausdorff distance between two geometries. This is basically a measure of how similar or dissimilar 2 geometries are, with a lower distance indicating more similar geometries.

The function can be executed with an optional densify fraction argument. If not specified, an approximation to the standard Hausdorff distance is used. This approximation is exact or close enough for a large subset of useful cases. Examples of these are:

- computing distance between Linestrings that are roughly parallel to each other, and roughly equal in length. This occurs in matching linear networks.
- Testing similarity of geometries.

If the default approximate provided by this method is insufficient, specify the optional densify fraction argument. Specifying this argument performs a segment densification before computing the discrete Hausdorff distance. The parameter sets the fraction by which to densify each segment. Each segment will be split into a number of equal-length subsegments, whose fraction of the total length is closest to the given fraction. Decreasing the densify fraction parameter will make the distance returned approach the true Hausdorff distance for the geometries.

Sintaxă	hausdorff_distance(geometry1, geometry2, [densify_fraction]) [] marks optional arguments
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry • **densify_fraction** - densify fraction amount
Exemple	• `hausdorff_distance( geometry1:= geom_from_wkt('LINESTRING (0 0, 2 1)'),geometry2:=geom_from_wkt('LINESTRING (0 0, 2 0)'))` → 2 • `hausdorff_distance( geom_from_wkt('LINESTRING (130 0, 0 0, 0 150)'),geom_from_wkt('LINESTRING (10 10, 10 150, 130 10)'))` → 14.142135623 • `hausdorff_distance( geom_from_wkt('LINESTRING (130 0, 0 0, 0 150)'),geom_from_wkt('LINESTRING (10 10, 10 150, 130 10)'), 0.5)` → 70.0

inclination

Returns the inclination measured from the zenith (0) to the nadir (180) on point_a to point_b.

Sintaxă	inclination(point_a, point_b)
Argumente	• **point_a** - point geometry • **point_b** - point geometry
Exemple	• `inclination( make_point( 5, 10, 0 ), make_point( 5, 10, 5 ) )` → 0.0 • `inclination( make_point( 5, 10, 0 ), make_point( 5, 10, 0 ) )` → 90.0 • `inclination( make_point( 5, 10, 0 ), make_point( 50, 100, 0 ) )` → 90.0 • `inclination( make_point( 5, 10, 0 ), make_point( 5, 10, -5 ) )` → 180.0

interior_ring_n

Returns a specific interior ring from a polygon geometry, or NULL if the geometry is not a polygon.

Sintaxă	interior_ring_n(geometry, index)
Argumente	• **geometry** - polygon geometry • **index** - index of interior to return, where 1 is the first interior ring
Exemple	• `geom_to_wkt(interior_ring_n(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1),(-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1),(-1 -1, 4 0, 4 2, 0 2, -1 -1))'),1))` → «LineString (-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1))»

intersection

Returns a geometry that represents the shared portion of two geometries.

Sintaxă	intersection(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `geom_to_wkt( intersection( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4)' ) ) )` → «LINESTRING(3 3, 4 4)» • `geom_to_wkt( intersection( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'MULTIPOINT(3.5 3.5, 4 5)' ) ) )` → «POINT(3.5 3.5)»

Mai multe detalii: Algoritmul *Intersection*

intersects

Tests whether a geometry intersects another. Returns TRUE if the geometries spatially intersect (share any portion of space) and false if they do not.

Sintaxă	intersects(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `intersects( geom_from_wkt( 'POINT(4 4)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `intersects( geom_from_wkt( 'POINT(4 5)' ), geom_from_wkt( 'POINT(5 5)' ) )` → FALSE

Mai multe detalii: *overlay_intersects*

intersects_bbox

Tests whether a geometry's bounding box overlaps another geometry's bounding box. Returns TRUE if the geometries spatially intersect the bounding box defined and false if they do not.

Sintaxă	intersects_bbox(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `intersects_bbox( geom_from_wkt( 'POINT(4 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `intersects_bbox( geom_from_wkt( 'POINT(6 5)' ), geom_from_wkt( 'POLYGON((3 3, 4 4, 5 5, 3 3))' ) )` → FALSE

is_closed

Returns TRUE if a line string is closed (start and end points are coincident), or false if a line string is not closed. If the geometry is not a line string then the result will be NULL.

Sintaxă	is_closed(geometry)
Argumente	• **geometry** - a line string geometry
Exemple	• `is_closed(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)'))` → FALSE • `is_closed(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2, 0 0)'))` → TRUE

is_empty

Returns TRUE if a geometry is empty (without coordinates), false if the geometry is not empty and NULL if there is no geometry. See also is_empty_or_null.

Sintaxă	is_empty(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `is_empty(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)'))` → FALSE • `is_empty(geom_from_wkt('LINESTRING EMPTY'))` → TRUE • `is_empty(geom_from_wkt('POINT(7 4)'))` → FALSE • `is_empty(geom_from_wkt('POINT EMPTY'))` → TRUE

Mai multe detalii: *is_empty_or_null*

is_empty_or_null

Returns TRUE if a geometry is NULL or empty (without coordinates) or false otherwise. This function is like the expression «@geometry IS NULL or is_empty(@geometry)»

Sintaxă	is_empty_or_null(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `is_empty_or_null(NULL)` → TRUE • `is_empty_or_null(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)'))` → FALSE • `is_empty_or_null(geom_from_wkt('LINESTRING EMPTY'))` → TRUE • `is_empty_or_null(geom_from_wkt('POINT(7 4)'))` → FALSE • `is_empty_or_null(geom_from_wkt('POINT EMPTY'))` → TRUE

Mai multe detalii: *is_empty*, *NULL*

is_multipart

Returns TRUE if the geometry is of Multi type.

Sintaxă	is_multipart(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `is_multipart(geom_from_wkt('MULTIPOINT ((0 0),(1 1),(2 2))'))` → TRUE • `is_multipart(geom_from_wkt('POINT (0 0)'))` → FALSE

is_valid

Returns TRUE if a geometry is valid; if it is well-formed in 2D according to the OGC rules.

Sintaxă	is_valid(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `is_valid(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2, 0 0)'))` → TRUE • `is_valid(geom_from_wkt('LINESTRING(0 0)'))` → FALSE

Mai multe detalii: *make_valid*, *Check validity* algorithm

$length

Returns the length of a linestring. If you need the length of a border of a polygon, use $perimeter instead. The length calculated by this function respects both the current project's ellipsoid setting and distance unit settings. For example, if an ellipsoid has been set for the project then the calculated length will be ellipsoidal, and if no ellipsoid is set then the calculated length will be planimetric.

Sintaxă	$length
Exemple	• `$length` → 42.4711

length

Returns the number of characters in a string or the length of a geometry linestring.

String variant

Returns the number of characters in a string.

Sintaxă	length(string)
Argumente	• **string** - string to count length of
Exemple	• `length('hello')` → 5

Geometry variant

Calculate the length of a geometry line object. Calculations are always planimetric in the Spatial Reference System (SRS) of this geometry, and the units of the returned length will match the units for the SRS. This differs from the calculations performed by the $length function, which will perform ellipsoidal calculations based on the project's ellipsoid and distance unit settings.

Sintaxă	length(geometry)
Argumente	• **geometry** - line geometry object
Exemple	• `length(geom_from_wkt('LINESTRING(0 0, 4 0)'))` → 4.0

Mai multe detalii: *straight_distance_2d*

length3D

Calculates the 3D length of a geometry line object. If the geometry is not a 3D line object, it returns its 2D length. Calculations are always planimetric in the Spatial Reference System (SRS) of this geometry, and the units of the returned length will match the units for the SRS. This differs from the calculations performed by the $length function, which will perform ellipsoidal calculations based on the project's ellipsoid and distance unit settings.

Sintaxă	length3D(geometry)
Argumente	• **geometry** - line geometry object
Exemple	• `length3D(geom_from_wkt('LINESTRINGZ(0 0 0, 3 0 4)'))` → 5.0

line_interpolate_angle

Returns the angle parallel to the geometry at a specified distance along a linestring geometry. Angles are in degrees clockwise from north.

Sintaxă	line_interpolate_angle(geometry, distance)
Argumente	• **geometry** - a linestring geometry • **distance** - distance along line to interpolate angle at
Exemple	• `line_interpolate_angle(geometry:=geom_from_wkt('LineString(0 0, 10 0)'),distance:=5)` → 90.0

line_interpolate_point

Returns the point interpolated by a specified distance along a linestring geometry.

Sintaxă	line_interpolate_point(geometry, distance)
Argumente	• **geometry** - a linestring geometry • **distance** - distance along line to interpolate
Exemple	• `geom_to_wkt(line_interpolate_point(geometry:=geom_from_wkt('LineString(0, 8 0)'), distance:=5))` → «Point (5 0)» • `geom_to_wkt(line_interpolate_point(geometry:=geom_from_wkt('LineString(0, 1 1, 2 0)'), distance:=2.1))` → «Point (1.48492424 0.51507576)» • `geom_to_wkt(line_interpolate_point(geometry:=geom_from_wkt('LineString(0, 1 0)'), distance:=2))` → NULL

Mai multe detalii: Algoritmul *Interpolate point on line*

line_locate_point

Returns the distance along a linestring corresponding to the closest position the linestring comes to a specified point geometry.

Sintaxă	line_locate_point(geometry, point)
Argumente	• **geometry** - a linestring geometry • **point** - point geometry to locate closest position on linestring to
Exemple	• `line_locate_point(geometry:=geom_from_wkt('LineString(0 0, 10 0)'),point:=geom_from_wkt('Point(5 0)'))` → 5.0

line_merge

Returns a LineString or MultiLineString geometry, where any connected LineStrings from the input geometry have been merged into a single linestring. This function will return NULL if passed a geometry which is not a LineString/MultiLineString.

Sintaxă	line_merge(geometry)
Argumente	• **geometry** - a LineString/MultiLineString geometry
Exemple	• `geom_to_wkt(line_merge(geom_from_wkt('MULTILINESTRING((0 0, 1 1),(1 1, 2 2))')))` → «LineString(0 0,1 1,2 2)» • `geom_to_wkt(line_merge(geom_from_wkt('MULTILINESTRING((0 0, 1 1),(11 1, 21 2))')))` → «MultiLineString((0 0, 1 1),(11 1, 21 2)»

line_substring

Returns the portion of a line (or curve) geometry which falls between the specified start and end distances (measured from the beginning of the line). Z and M values are linearly interpolated from existing values.

Sintaxă	line_substring(geometry, start_distance, end_distance)
Argumente	• **geometry** - a linestring or curve geometry • **start_distance** - distance to start of substring • **end_distance** - distance to end of substring
Exemple	• `geom_to_wkt(line_substring(geometry:=geom_from_wkt('LineString(0 0, 10 0)'),start_distance:=2,end_distance:=6))` → «LineString (2 0,6 0)»

Mai multe detalii: Algoritmul *Line substring*

m

Returns the m (measure) value of a point geometry.

Sintaxă	m(geometry)
Argumente	• **geometry** - a point geometry
Exemple	• `m( geom_from_wkt( 'POINTM(2 5 4)' ) )` → 4

m_at

Retrieves a m coordinate of the geometry, or NULL if the geometry has no m value.

Sintaxă	m_at(geometry, vertex)
Argumente	• **geometry** - geometry object • **vertex** - index of the vertex of the geometry (indices start at 0; negative values apply from the last index, starting at -1)
Exemple	• `m_at(geom_from_wkt('LineStringZM(0 0 0 0, 10 10 0 5, 10 10 0 0)'), 1)` → 5

m_max

Returns the maximum m (measure) value of a geometry.

Sintaxă	m_max(geometry)
Argumente	• **geometry** - a geometry containing m values
Exemple	• `m_max( make_point_m( 0,0,1 ) )` → 1 • `m_max(make_line( make_point_m( 0,0,1 ), make_point_m( -1, -1,2 ), make_point_m( -2,-2,0 ) ) )` → 2

m_min

Returns the minimum m (measure) value of a geometry.

Sintaxă	m_min(geometry)
Argumente	• **geometry** - a geometry containing m values
Exemple	• `m_min( make_point_m( 0,0,1 ) )` → 1 • `m_min(make_line( make_point_m( 0,0,1 ), make_point_m( -1, -1,2 ), make_point_m( -2,-2,0 ) ) )` → 0

main_angle

Returns the angle of the long axis (clockwise, in degrees from North) of the oriented minimal bounding rectangle, which completely covers the geometry.

Sintaxă	main_angle(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `main_angle(geom_from_wkt('Polygon ((321577 129614, 321581 129618, 321585 129615, 321581 129610, 321577 129614))'))` → 38.66

make_circle

Creates a circular polygon.

Sintaxă	make_circle(center, radius, [segments=36]) [] marks optional arguments
Argumente	• **center** - center point of the circle • **radius** - radius of the circle • **segments** - optional argument for polygon segmentation. By default this value is 36
Exemple	• `geom_to_wkt(make_circle(make_point(10,10), 5, 4))` → «Polygon ((10 15, 15 10, 10 5, 5 10, 10 15))» • `geom_to_wkt(make_circle(make_point(10,10,5), 5, 4))` → «PolygonZ ((10 15 5, 15 10 5, 10 5 5, 5 10 5, 10 15 5))» • `geom_to_wkt(make_circle(make_point(10,10,5,30), 5, 4))` → «PolygonZM ((10 15 5 30, 15 10 5 30, 10 5 5 30, 5 10 5 30, 10 15 5 30))»

make_ellipse

Creates an elliptical polygon.

Sintaxă	make_ellipse(center, semi_major_axis, semi_minor_axis, azimuth, [segments=36]) [] marks optional arguments
Argumente	• **center** - center point of the ellipse • **semi_major_axis** - semi-major axis of the ellipse • **semi_minor_axis** - semi-minor axis of the ellipse • **azimuth** - orientation of the ellipse • **segments** - optional argument for polygon segmentation. By default this value is 36
Exemple	• `geom_to_wkt(make_ellipse(make_point(10,10), 5, 2, 90, 4))` → «Polygon ((15 10, 10 8, 5 10, 10 12, 15 10))» • `geom_to_wkt(make_ellipse(make_point(10,10,5), 5, 2, 90, 4))` → «PolygonZ ((15 10 5, 10 8 5, 5 10 5, 10 12 5, 15 10 5))» • `geom_to_wkt(make_ellipse(make_point(10,10,5,30), 5, 2, 90, 4))` → «PolygonZM ((15 10 5 30, 10 8 5 30, 5 10 5 30, 10 12 5 30, 15 10 5 30))»

make_line

Creates a line geometry from a series of point geometries.

List of arguments variant

Line vertices are specified as separate arguments to the function.

Sintaxă	make_line(point1, point2, …)
Argumente	• **point** - a point geometry (or array of points)
Exemple	• `geom_to_wkt(make_line(make_point(2,4),make_point(3,5)))` → «LineString (2 4, 3 5)» • `geom_to_wkt(make_line(make_point(2,4),make_point(3,5), make_point(9,7)))` → «LineString (2 4, 3 5, 9 7)»

Array variant

Line vertices are specified as an array of points.

Sintaxă	make_line(array)
Argumente	• **array** - array of points
Exemple	• `geom_to_wkt(make_line(array(make_point(2,4),make_point(3,5),make_point(9,7))))` → «LineString (2 4, 3 5, 9 7)»

make_point

Creates a point geometry from an x and y (and optional z and m) value.

Sintaxă	make_point(x, y, [z], [m]) [] marks optional arguments
Argumente	• **x** - x coordinate of point • **y** - y coordinate of point • **z** - optional z coordinate of point • **m** - optional m value of point
Exemple	• `geom_to_wkt(make_point(2,4))` → «Point (2 4)» • `geom_to_wkt(make_point(2,4,6))` → «PointZ (2 4 6)» • `geom_to_wkt(make_point(2,4,6,8))` → «PointZM (2 4 6 8)»

make_point_m

Creates a point geometry from an x, y coordinate and m value.

Sintaxă	make_point_m(x, y, m)
Argumente	• **x** - x coordinate of point • **y** - y coordinate of point • **m** - m value of point
Exemple	• `geom_to_wkt(make_point_m(2,4,6))` → «PointM (2 4 6)»

make_polygon

Creates a polygon geometry from an outer ring and optional series of inner ring geometries.

Sintaxă	make_polygon(outerRing, [innerRing1], [innerRing2], …) [] marks optional arguments
Argumente	• **outerRing** - closed line geometry for polygon's outer ring • **innerRing** - optional closed line geometry for inner ring
Exemple	• `geom_to_wkt(make_polygon(geom_from_wkt('LINESTRING( 0 0, 0 1, 1 1, 1 0, 0 0 )')))` → «Polygon ((0 0, 0 1, 1 1, 1 0, 0 0))» • `geom_to_wkt(make_polygon(geom_from_wkt('LINESTRING( 0 0, 0 1, 1 1, 1 0, 0 0 )'),geom_from_wkt('LINESTRING( 0.1 0.1, 0.1 0.2, 0.2 0.2, 0.2 0.1, 0.1 0.1 )'), geom_from_wkt('LINESTRING( 0.8 0.8, 0.8 0.9, 0.9 0.9, 0.9 0.8, 0.8 0.8 )')))` → «Polygon ((0 0, 0 1, 1 1, 1 0, 0 0),(0.1 0.1, 0.1 0.2, 0.2 0.2, 0.2 0.1, 0.1 0.1),(0.8 0.8, 0.8 0.9, 0.9 0.9, 0.9 0.8, 0.8 0.8))»

make_rectangle_3points

Creates a rectangle from 3 points.

Sintaxă	make_rectangle_3points(point1, point2, point3, [option=0]) [] marks optional arguments
Argumente	• **point1** - First point. • **point2** - Second point. • **point3** - Third point. • **option** - An optional argument to construct the rectangle. By default this value is 0. Value can be 0 (distance) or 1 (projected). Option distance: Second distance is equal to the distance between 2nd and 3rd point. Option projected: Second distance is equal to the distance of the perpendicular projection of the 3rd point on the segment or its extension.
Exemple	• `geom_to_wkt(make_rectangle_3points(make_point(0, 0), make_point(0,5), make_point(5, 5), 0))` → «Polygon ((0 0, 0 5, 5 5, 5 0, 0 0))» • `geom_to_wkt(make_rectangle_3points(make_point(0, 0), make_point(0,5), make_point(5, 3), 1))` → «Polygon ((0 0, 0 5, 5 5, 5 0, 0 0))»

make_regular_polygon

Creates a regular polygon.

Sintaxă	make_regular_polygon(center, radius, number_sides, [circle=0]) [] marks optional arguments
Argumente	• **center** - center of the regular polygon • **radius** - second point. The first if the regular polygon is inscribed. The midpoint of the first side if the regular polygon is circumscribed. • **number_sides** - Number of sides/edges of the regular polygon • **circle** - Optional argument to construct the regular polygon. By default this value is 0. Value can be 0 (inscribed) or 1 (circumscribed)
Exemple	• `geom_to_wkt(make_regular_polygon(make_point(0,0), make_point(0,5), 5))` → «Polygon ((0 5, 4.76 1.55, 2.94 -4.05, -2.94 -4.05, -4.76 1.55, 0 5))» • `geom_to_wkt(make_regular_polygon(make_point(0,0), project(make_point(0,0), 4.0451, radians(36)), 5))` → «Polygon ((0 5, 4.76 1.55, 2.94 -4.05, -2.94 -4.05, -4.76 1.55, 0 5))»

make_square

Creates a square from a diagonal.

Sintaxă	make_square(point1, point2)
Argumente	• **point1** - First point of the diagonal • **point2** - Last point of the diagonal
Exemple	• `geom_to_wkt(make_square( make_point(0,0), make_point(5, 5)))` → «Polygon ((0 0, -0 5, 5 5, 5 0, 0 0))» • `geom_to_wkt(make_square( make_point(5,0), make_point(5, 5)))` → «Polygon ((5 0, 2.5 2.5, 5 5, 7.5 2.5, 5 0))»

make_triangle

Creates a triangle polygon.

Sintaxă	make_triangle(point1, point2, point3)
Argumente	• **point1** - first point of the triangle • **point2** - second point of the triangle • **point3** - third point of the triangle
Exemple	• `geom_to_wkt(make_triangle(make_point(0,0), make_point(5, 5), make_point(0,10)))` → «Triangle ((0 0, 5 5, 0 10, 0 0))» • `geom_to_wkt(boundary(make_triangle(make_point(0,0), make_point(5,5), make_point(0,10))))` → «LineString (0 0, 5 5, 0 10, 0 0)»

make_valid

Returns a valid geometry or an empty geometry if the geometry could not be made valid.

Sintaxă	make_valid(geometry, [method=structure], [keep_collapsed=false]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **method** - repair algorithm. May be either «structure» or «linework». The «linework» option combines all rings into a set of noded lines and then extracts valid polygons from that linework. The «structure» method first makes all rings valid and then merges shells and subtracts holes from shells to generate valid result. Assumes that holes and shells are correctly categorized. • **keep_collapsed** - if set to true, then components that have collapsed into a lower dimensionality will be kept. For example, a ring collapsing to a line, or a line collapsing to a point.
Exemple	• `geom_to_wkt(make_valid(geom_from_wkt('POLYGON((3 2, 4 1, 5 8, 3 2, 4 2))')))` → «Polygon ((3 2, 5 8, 4 1, 3 2))» • `geom_to_wkt(make_valid(geom_from_wkt('POLYGON((3 2, 4 1, 5 8, 3 2, 4 2))'), 'linework'))` → «GeometryCollection (Polygon ((5 8, 4 1, 3 2, 5 8)),LineString (3 2, 4 2))» • `geom_to_wkt(make_valid(geom_from_wkt('POLYGON((3 2, 4 1, 5 8))'), method:='linework'))` → «Polygon ((3 2, 4 1, 5 8, 3 2))» • `make_valid(geom_from_wkt('LINESTRING(0 0)'))` → An empty geometry

Mai multe detalii: *is_valid*, *Fix geometries* algorithm

minimal_circle

Returns the minimal enclosing circle of a geometry. It represents the minimum circle that encloses all geometries within the set.

Sintaxă	minimal_circle(geometry, [segments=36]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **segments** - optional argument for polygon segmentation. By default this value is 36
Exemple	• `geom_to_wkt( minimal_circle( geom_from_wkt( 'LINESTRING(0 5, 0 -5, 2 1)' ), 4 ) )` → «Polygon ((0 5, 5 -0, -0 -5, -5 0, 0 5))» • `geom_to_wkt( minimal_circle( geom_from_wkt( 'MULTIPOINT(1 2, 3 4, 3 2)' ), 4 ) )` → «Polygon ((3 4, 3 2, 1 2, 1 4, 3 4))»

Mai multe detalii: Algoritmul *Minimum enclosing circles*

nodes_to_points

Returns a multipoint geometry consisting of every node in the input geometry.

Sintaxă	nodes_to_points(geometry, [ignore_closing_nodes=false]) [] marks optional arguments
Argumente	• **geometry** - geometry object • **ignore_closing_nodes** - optional argument specifying whether to include duplicate nodes which close lines or polygons rings. Defaults to false, set to true to avoid including these duplicate nodes in the output collection.
Exemple	• `geom_to_wkt(nodes_to_points(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)')))` → «MultiPoint ((0 0),(1 1),(2 2))» • `geom_to_wkt(nodes_to_points(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))'),true))` → «MultiPoint ((-1 -1),(4 0),(4 2),(0 2))»

Mai multe detalii: Algoritmul *Extract vertices*

num_geometries

Returns the number of geometries in a geometry collection, or the number of parts in a multi-part geometry. The function returns NULL if the input geometry is not a collection.

Sintaxă	num_geometries(geometry)
Argumente	• **geometry** - geometry collection or multi-part geometry
Exemple	• `num_geometries(geom_from_wkt('GEOMETRYCOLLECTION(POINT(0 1), POINT(0 0), POINT(1 0), POINT(1 1))'))` → 4 • `num_geometries(geom_from_wkt('MULTIPOINT((0 1), (0 0), (1 0))'))` → 3

num_interior_rings

Returns the number of interior rings in a polygon or geometry collection, or NULL if the input geometry is not a polygon or collection.

Sintaxă	num_interior_rings(geometry)
Argumente	• **geometry** - input geometry
Exemple	• `num_interior_rings(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1),(-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1))'))` → 1

num_points

Returns the number of vertices in a geometry.

Sintaxă	num_points(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `num_points(@geometry)` → number of vertices in the current feature's geometry

num_rings

Returns the number of rings (including exterior rings) in a polygon or geometry collection, or NULL if the input geometry is not a polygon or collection.

Sintaxă	num_rings(geometry)
Argumente	• **geometry** - input geometry
Exemple	• `num_rings(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1),(-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1))'))` → 2

offset_curve

Returns a geometry formed by offsetting a linestring geometry to the side. Distances are in the Spatial Reference System of this geometry.

Sintaxă	offset_curve(geometry, distance, [segments=8], [join=1], [miter_limit=2.0]) [] marks optional arguments
Argumente	• **geometry** - a (multi)linestring geometry • **distance** - offset distance. Positive values will be buffered to the left of lines, negative values to the right • **segments** - number of segments to use to represent a quarter circle when a round join style is used. A larger number results in a smoother line with more nodes. • **join** - join style for corners, where 1 = round, 2 = miter and 3 = bevel • **miter_limit** - limit on the miter ratio used for very sharp corners (when using miter joins only)
Exemple	• `offset_curve(@geometry, 10.5)` → line offset to the left by 10.5 units • `offset_curve(@geometry, -10.5)` → line offset to the right by 10.5 units • `offset_curve(@geometry, 10.5, segments:=16, join:=1)` → line offset to the left by 10.5 units, using more segments to result in a smoother curve • `offset_curve(@geometry, 10.5, join:=3)` → line offset to the left by 10.5 units, using a beveled join

Mai multe detalii: Algoritmul *Offset lines*

order_parts

Ordonează părțile unei MultiGeometrii după anumite criterii

Sintaxă	order_parts(geometry, orderby, [ascending=true]) [] marks optional arguments
Argumente	• **geometry** - a multi-type geometry • **orderby** - an expression string defining the order criteria • **ascending** - boolean, True for ascending, False for descending
Exemple	• `geom_to_wkt(order_parts(geom_from_wkt('MultiPolygon (((1 1, 5 1, 5 5, 1 5, 1 1)),((1 1, 9 1, 9 9, 1 9, 1 1)))'), 'area(@geometry)', False))` → «MultiPolygon (((1 1, 9 1, 9 9, 1 9, 1 1)),((1 1, 5 1, 5 5, 1 5, 1 1)))» • `geom_to_wkt(order_parts(geom_from_wkt('LineString(1 2, 3 2, 4 3)'), '1', True))` → «LineString(1 2, 3 2, 4 3)»

oriented_bbox

Returns a geometry which represents the minimal oriented bounding box of an input geometry.

Sintaxă	oriented_bbox(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `geom_to_wkt( oriented_bbox( geom_from_wkt( 'MULTIPOINT(1 2, 3 4, 3 2)' ) ) )` → «Polygon ((3 2, 3 4, 1 4, 1 2, 3 2))»

Mai multe detalii: Algoritmul *Oriented minimum bounding box*

overlaps

Tests whether a geometry overlaps another. Returns TRUE if the geometries share space, are of the same dimension, but are not completely contained by each other.

Sintaxă	overlaps(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `overlaps( geom_from_wkt( 'LINESTRING(3 5, 4 4, 5 5, 5 3)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `overlaps( geom_from_wkt( 'LINESTRING(0 0, 1 1)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → FALSE

overlay_contains

Returns whether the current feature spatially contains at least one feature from a target layer, or an array of expression-based results for the features in the target layer contained in the current feature.

Read more on the underlying GEOS „Contains" predicate, as described in PostGIS ST_Contains function.

Sintaxă	overlay_contains(layer, [expression], [filter], [limit], [cache=false]) [] marks optional arguments
Argumente	• **layer** - the layer whose overlay is checked • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features will be checked. • **limit** - an optional integer to limit the number of matching features. If not set, all the matching features will be returned. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider)
Exemple	• `overlay_contains('regions')` → TRUE if the current feature spatially contains a region • `overlay_contains('regions', filter:= population > 10000)` → TRUE if the current feature spatially contains a region with a population greater than 10000 • `overlay_contains('regions', name)` → an array of names, for the regions contained in the current feature • `array_to_string(overlay_contains('regions', name))` → a string as a comma separated list of names, for the regions contained in the current feature • `array_sort(overlay_contains(layer:='regions', expression:='name', filter:=population > 10000))` → o matrice ordonată după numele regiunilor conținute în entitatea curentă, și cu o populație de peste 10000 de locuitori • `overlay_contains(layer:='regions', expression:= geom_to_wkt(@geometry), limit:=2)` → an array of geometries (in WKT), for up to two regions contained in the current feature

Mai multe detalii: *contains*, *manipularea matricilor*, algoritmul *Selectare după locație*

overlay_crosses

Returns whether the current feature spatially crosses at least one feature from a target layer, or an array of expression-based results for the features in the target layer crossed by the current feature.

Read more on the underlying GEOS „Crosses" predicate, as described in PostGIS ST_Crosses function.

Sintaxă	overlay_crosses(layer, [expression], [filter], [limit], [cache=false]) [] marks optional arguments
Argumente	• **layer** - the layer whose overlay is checked • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features will be checked. • **limit** - an optional integer to limit the number of matching features. If not set, all the matching features will be returned. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider)
Exemple	• `overlay_crosses('regions')` → TRUE if the current feature spatially crosses a region • `overlay_crosses('regions', filter:= population > 10000)` → TRUE if the current feature spatially crosses a region with a population greater than 10000 • `overlay_crosses('regions', name)` → an array of names, for the regions crossed by the current feature • `array_to_string(overlay_crosses('regions', name))` → a string as a comma separated list of names, for the regions crossed by the current feature • `array_sort(overlay_crosses(layer:='regions', expression:='name', filter:= population > 10000))` → o matrice ordonată după numele regiunilor traversate de entitatea curentă, și cu o populație de peste 10000 de locuitori • `overlay_crosses(layer:='regions', expression:= geom_to_wkt(@geometry), limit:=2)` → an array of geometries (in WKT), for up to two regions crossed by the current feature

Mai multe detalii: *crosses*, *Gestionarea matricilor*, Algoritmul *Selectare după locație*

overlay_disjoint

Returns whether the current feature is spatially disjoint from all the features of a target layer, or an array of expression-based results for the features in the target layer that are disjoint from the current feature.

Read more on the underlying GEOS „Disjoint” predicate, as described in PostGIS ST_Disjoint function.

Sintaxă	overlay_disjoint(layer, [expression], [filter], [limit], [cache=false]) [] marks optional arguments
Argumente	• **layer** - the layer whose overlay is checked • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features will be checked. • **limit** - an optional integer to limit the number of matching features. If not set, all the matching features will be returned. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider)
Exemple	• `overlay_disjoint('regions')` → TRUE if the current feature is spatially disjoint from all the regions • `overlay_disjoint('regions', filter:= population > 10000)` → TRUE if the current feature is spatially disjoint from all the regions with a population greater than 10000 • `overlay_disjoint('regions', name)` → an array of names, for the regions spatially disjoint from the current feature • `array_to_string(overlay_disjoint('regions', name))` → a string as a comma separated list of names, for the regions spatially disjoint from the current feature • `array_sort(overlay_disjoint(layer:='regions', expression:='name', filter:=population > 10000))` → o matrice ordonată după numele regiunilor disjunse spațial de entitatea curentă, și cu o populație de peste 10000 de locuitori • `overlay_disjoint(layer:='regions', expression:= geom_to_wkt(@geometry), limit:=2)` → an array of geometries (in WKT), for up to two regions spatially disjoint from the current feature

Mai multe detalii: *disjoint*, *Gestionarea matricilor*, Algoritmul *Selectare după locație*

overlay_equals

Returns whether the current feature spatially equals to at least one feature from a target layer, or an array of expression-based results for the features in the target layer that are spatially equal to the current feature.

Read more on the underlying GEOS „Equals” predicate, as described in PostGIS ST_Equals function.

Sintaxă	overlay_equals(layer, [expression], [filter], [limit], [cache=false]) [] marks optional arguments
Argumente	• **layer** - the layer whose overlay is checked • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features will be checked. • **limit** - an optional integer to limit the number of matching features. If not set, all the matching features will be returned. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider)
Exemple	• `overlay_equals('regions')` → TRUE if the current feature is spatially equal to a region • `overlay_equals('regions', filter:= population > 10000)` → TRUE if the current feature is spatially equal to a region with a population greater than 10000 • `overlay_equals('regions', name)` → an array of names, for the regions spatially equal to the current feature • `array_to_string(overlay_equals('regions', name))` → a string as a comma separated list of names, for the regions spatially equal to the current feature • `array_sort(overlay_equals(layer:='regions', expression:='name', filter:=population > 10000))` → o matrice ordonată după numele regiunilor egale spațial cu entitatea curentă, și cu o populație de peste 10000 de locuitori • `overlay_equals(layer:='regions', expression:= geom_to_wkt(@geometry), limit:=2)` → an array of geometries (in WKT), for up to two regions spatially equal to the current feature

Mai multe detalii: *Gestionarea matricilor*, Algoritmul *Selectare după locație*

overlay_intersects

Returns whether the current feature spatially intersects at least one feature from a target layer, or an array of expression-based results for the features in the target layer intersected by the current feature.

Read more on the underlying GEOS „Intersects” predicate, as described in PostGIS ST_Intersects function.

Sintaxă	overlay_intersects(layer, [expression], [filter], [limit], [cache=false], [min_overlap], [min_inscribed_circle_radius], [return_details], [sort_by_intersection_size]) [] marks optional arguments
Argumente	• **layer** - the layer whose overlay is checked • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features will be checked. • **limit** - an optional integer to limit the number of matching features. If not set, all the matching features will be returned. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider) • **min_overlap** - defines an optional exclusion filter: – for polygons, a minimum area in current feature squared units for the intersection. If the intersection results in multiple polygons the intersection will be returned if at least one of the polygons has an area greater or equal to the value – for lines, a minimum length in current feature units. If the intersection results in multiple lines the intersection will be returned if at least one of the lines has a length greater or equal to the value. • **min_inscribed_circle_radius** - defines an optional exclusion filter (for polygons only): minimum radius in current feature units for the maximum inscribed circle of the intersection. If the intersection results in multiple polygons the intersection will be returned if at least one of the polygons has a radius for the maximum inscribed circle greater or equal to the value. Read more on the underlying GEOS predicate, as described in PostGIS ST_MaximumInscribedCircle function. This argument requires GEOS >= 3.9. • **return_details** - Set this to true to return a list of maps containing (key names in quotes) the feature «id», the expression «result» and the «overlap» value. The «radius» of the maximum inscribed circle is also returned when the target layer is a polygon. Only valid when used with the expression parameter • **sort_by_intersection_size** - only valid when used with an expression, set this to «des» to return the results ordered by the overlap value in descending order or set this to «asc» for ascending order.
Exemple	• `overlay_intersects('regions')` → TRUE if the current feature spatially intersects a region • `overlay_intersects('regions', filter:= population > 10000)` → TRUE if the current feature spatially intersects a region with a population greater than 10000 • `overlay_intersects('regions', name)` → an array of names, for the regions intersected by the current feature • `array_to_string(overlay_intersects('regions', name))` → a string as a comma separated list of names, for the regions intersected by the current feature • `array_sort(overlay_intersects(layer:='regions', expression:='name', filter:=population > 10000))` → o matrice ordonată după numele regiunilor intersectate de entitatea curentă, și cu o populație de peste 10000 de locuitori • `overlay_intersects(layer:='regions', expression:= geom_to_wkt(@geometry), limit:=2)` → an array of geometries (in WKT), for up to two regions intersected by the current feature • `overlay_intersects(layer:='regions', min_overlap:=0.54)` → TRUE if the current feature spatially intersects a region and the intersection area (of at least one of the parts in case of multipolygons) is greater or equal to 0.54 • `overlay_intersects(layer:='regions', min_inscribed_circle_radius:=0.54)` → TRUE if the current feature spatially intersects a region and the intersection area maximum inscribed circle's radius (of at least one of the parts in case of multipart) is greater or equal to 0.54 • `overlay_intersects(layer:='regions', expression:= geom_to_wkt(@geometry), return_details:=true)` → an array of maps containing «id», «result», «overlap» and «radius» • `overlay_intersects(layer:='regions', expression:=`

Mai multe detalii: *intersects*, *manipularea matricilor*, algoritmul *Selectare după locație*

overlay_nearest

Returns whether the current feature has feature(s) from a target layer within a given distance, or an array of expression-based results for the features in the target layer within a distance from the current feature.

Note: This function can be slow and consume a lot of memory for large layers.

Sintaxă	overlay_nearest(layer, [expression], [filter], [limit=1], [max_distance], [cache=false]) [] marks optional arguments
Argumente	• **layer** - the target layer • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features in the target layer will be used. • **limit** - an optional integer to limit the number of matching features. If not set, only the nearest feature will be returned. If set to -1, returns all the matching features. • **max_distance** - an optional distance to limit the search of matching features. If not set, all the features in the target layer will be used. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider)
Exemple	• `overlay_nearest('airports')` → TRUE if the „airports” layer has at least one feature • `overlay_nearest('airports', max_distance:= 5000)` → TRUE if there is an airport within a distance of 5000 map units from the current feature • `overlay_nearest('airports', name)` → the name of the closest airport to the current feature, as an array • `array_to_string(overlay_nearest('airports', name))` → the name of the closest airport to the current feature, as a string • `overlay_nearest(layer:='airports', expression:= name, max_distance:= 5000)` → numele celui mai apropiat aeroport, situat la o distanță de 5000 de unități de hartă față de entitatea curentă, sub formă de matrice • `overlay_nearest(layer:='airports', expression:='name', filter:= 'Use'='Civil', limit:=3)` → o matrice cu nume, pentru până la cele mai apropiate trei aeroporturi civile, ordonată după distanță • `overlay_nearest(layer:='airports', expression:="name", limit:= -1, max_distance:= 5000)` → an array of names, for all the airports within a distance of 5000 map units from the current feature, ordered by distance

Mai multe detalii: *Gestionarea matricilor*, Algoritmul *Join attributes by nearest*

overlay_touches

Returns whether the current feature spatially touches at least one feature from a target layer, or an array of expression-based results for the features in the target layer touched by the current feature.

Read more on the underlying GEOS „Touches” predicate, as described in PostGIS ST_Touches function.

Sintaxă	overlay_touches(layer, [expression], [filter], [limit], [cache=false]) [] marks optional arguments
Argumente	• **layer** - the layer whose overlay is checked • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features will be checked. • **limit** - an optional integer to limit the number of matching features. If not set, all the matching features will be returned. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider)
Exemple	• `overlay_touches('regions')` → TRUE if the current feature spatially touches a region • `overlay_touches('regions', filter:= population > 10000)` → TRUE if the current feature spatially touches a region with a population greater than 10000 • `overlay_touches('regions', name)` → an array of names, for the regions touched by the current feature • `string_to_array(overlay_touches('regions', name))` → a string as a comma separated list of names, for the regions touched by the current feature • `array_sort(overlay_touches(layer:='regions', expression:='name', filter:=population > 10000))` → o matrice ordonată după numele regiunilor atinse de entitatea curentă, și cu o populație de peste 10000 de locuitori • `overlay_touches(layer:='regions', expression:= geom_to_wkt(@geometry), limit:=2)` → an array of geometries (in WKT), for up to two regions touched by the current feature

Mai multe detalii: *touches*, *Gestionarea matricilor*, Algoritmul *Selectare după locație*

overlay_within

Returns whether the current feature is spatially within at least one feature from a target layer, or an array of expression-based results for the features in the target layer that contain the current feature.

Read more on the underlying GEOS „Within” predicate, as described in PostGIS ST_Within function.

Sintaxă	overlay_within(layer, [expression], [filter], [limit], [cache=false]) [] marks optional arguments
Argumente	• **layer** - the layer whose overlay is checked • **expression** - an optional expression to evaluate on the features from the target layer. If not set, the function will just return a boolean indicating whether there is at least one match. • **filter** - an optional expression to filter the target features to check. If not set, all the features will be checked. • **limit** - an optional integer to limit the number of matching features. If not set, all the matching features will be returned. • **cache** - set this to true to build a local spatial index (most of the time, this is unwanted, unless you are working with a particularly slow data provider)
Exemple	• `overlay_within('regions')` → TRUE if the current feature is spatially within a region • `overlay_within('regions', filter:= population > 10000)` → TRUE if the current feature is spatially within a region with a population greater than 10000 • `overlay_within('regions', name)` → an array of names, for the regions containing the current feature • `array_to_string(overlay_within('regions', name))` → a string as a comma separated list of names, for the regions containing the current feature • `array_sort(overlay_within(layer:='regions', expression:='name', filter:=population > 10000))` → o matrice ordonată după numele regiunilor care conțin entitatea curentă, și au o populație de peste 10000 de locuitori • `overlay_within(layer:='regions', expression:= geom_to_wkt(@geometry), limit:=2)` → an array of geometries (in WKT), for up to two regions containing the current feature

Mai multe detalii: *within*, *Gestionarea matricilor*, Algoritmul *Selectare după locație*

$perimeter

Returns the perimeter length of the current feature. The perimeter calculated by this function respects both the current project's ellipsoid setting and distance unit settings. For example, if an ellipsoid has been set for the project then the calculated perimeter will be ellipsoidal, and if no ellipsoid is set then the calculated perimeter will be planimetric.

Sintaxă	$perimeter
Exemple	• `$perimeter` → 42

perimeter

Returns the perimeter of a geometry polygon object. Calculations are always planimetric in the Spatial Reference System (SRS) of this geometry, and the units of the returned perimeter will match the units for the SRS. This differs from the calculations performed by the $perimeter function, which will perform ellipsoidal calculations based on the project's ellipsoid and distance unit settings.

Sintaxă	perimeter(geometry)
Argumente	• **geometry** - polygon geometry object
Exemple	• `perimeter(geom_from_wkt('POLYGON((0 0, 4 0, 4 2, 0 2, 0 0))'))` → 12.0

point_n

Returns a specific node from a geometry.

Sintaxă	point_n(geometry, index)
Argumente	• **geometry** - geometry object • **index** - index of node to return, where 1 is the first node; if the value is negative, the selected vertex index will be its total count minus the absolute value
Exemple	• `geom_to_wkt(point_n(geom_from_wkt('POLYGON((0 0, 4 0, 4 2, 0 2, 0 0))'),2))` → «Point (4 0)»

Mai multe detalii: Algoritmul *Extract specific vertices*

point_on_surface

Returns a point guaranteed to lie on the surface of a geometry.

Sintaxă	point_on_surface(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `point_on_surface(@geometry)` → a point geometry

Mai multe detalii: Algoritmul *Point on Surface*

pole_of_inaccessibility

Calculates the approximate pole of inaccessibility for a surface, which is the most distant internal point from the boundary of the surface. This function uses the «polylabel» algorithm (Vladimir Agafonkin, 2016), which is an iterative approach guaranteed to find the true pole of inaccessibility within a specified tolerance. More precise tolerances require more iterations and will take longer to calculate.

Sintaxă	pole_of_inaccessibility(geometry, tolerance)
Argumente	• **geometry** - a geometry • **tolerance** - maximum distance between the returned point and the true pole location
Exemple	• `geom_to_wkt(pole_of_inaccessibility( geom_from_wkt('POLYGON((0 1, 0 9, 3 10, 3 3, 10 3, 10 1, 0 1))'), 0.1))` → «Point(1.546875 2.546875)»

Mai multe detalii: Algoritmul *Pole of inaccessibility*

project

Returns a point projected from a start point using a distance, a bearing (azimuth) and an elevation in radians.

Sintaxă	project(point, distance, azimuth, [elevation]) [] marks optional arguments
Argumente	• **point** - start point • **distance** - distance to project • **azimuth** - azimuth in radians clockwise, where 0 corresponds to north • **elevation** - angle of inclination in radians
Exemple	• `geom_to_wkt(project(make_point(1, 2), 3, radians(270)))` → «Point(-2, 2)»

Mai multe detalii: Algoritmul *Project points (Cartesian)*

relate

Tests the Dimensional Extended 9 Intersection Model (DE-9IM) representation of the relationship between two geometries.

Relationship variant

Returns the Dimensional Extended 9 Intersection Model (DE-9IM) representation of the relationship between two geometries.

Sintaxă	relate(geometry, geometry)
Argumente	• **geometry** - a geometry • **geometry** - a geometry
Exemple	• `relate( geom_from_wkt( 'LINESTRING(40 40,120 120)' ), geom_from_wkt( 'LINESTRING(40 40,60 120)' ) )` → «FF1F00102»

Pattern match variant

Tests whether the DE-9IM relationship between two geometries matches a specified pattern.

Sintaxă	relate(geometry, geometry, pattern)
Argumente	• **geometry** - a geometry • **geometry** - a geometry • **pattern** - DE-9IM pattern to match
Exemple	• `relate( geom_from_wkt( 'LINESTRING(40 40,120 120)' ), geom_from_wkt( 'LINESTRING(40 40,60 120)' ), '**1F001**' )` → TRUE

reverse

Reverses the direction of a line string by reversing the order of its vertices.

Sintaxă	reverse(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `geom_to_wkt(reverse(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)')))` → «LINESTRING(2 2, 1 1, 0 0)»

Mai multe detalii: Algoritmul *Inversează sensul liniei*

rotate

Returns a rotated version of a geometry. Calculations are in the Spatial Reference System of this geometry.

Sintaxă	rotate(geometry, rotation, [center=NULL], [per_part=false]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **rotation** - clockwise rotation in degrees • **center** - rotation center point. If not specified, the center of the geometry's bounding box is used. • **per_part** - apply rotation per part. If true, then rotation will apply around the center of each part's bounding box when the input geometry is multipart and an explicit rotation center point is not specified.
Exemple	• `rotate(@geometry, 45, make_point(4, 5))` → geometry rotated 45 degrees clockwise around the (4, 5) point • `rotate(@geometry, 45)` → geometry rotated 45 degrees clockwise around the center of its bounding box

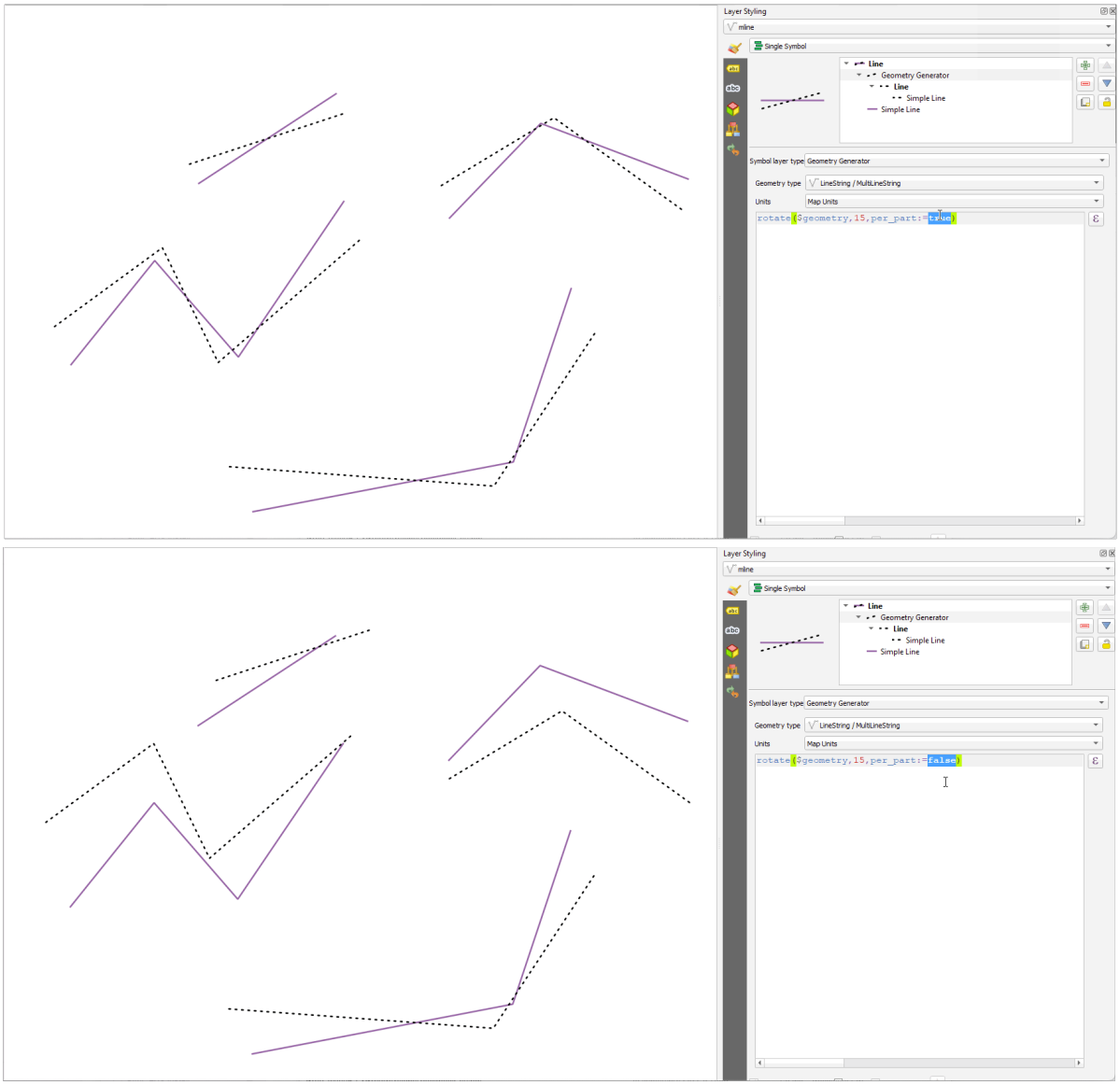


Fig. 13.7: Rotating features

roundness

Calculates how close a polygon shape is to a circle. The function Returns TRUE when the polygon shape is a perfect circle and 0 when it is completely flat.

Sintaxă	roundness(geometry)
Argumente	• **geometry** - a polygon
Exemple	• `round(roundness(geom_from_wkt('POLYGON(( 0 0, 0 1, 1 1, 1 0, 0 0))')), 3)` → 0.785 • `round(roundness(geom_from_wkt('POLYGON(( 0 0, 0 0.1, 1 0.1, 1 0, 0 0))')), 3)` → 0.260

Mai multe detalii: *Roundness* algorithm

scale

Returns a scaled version of a geometry. Calculations are in the Spatial Reference System of this geometry.

Sintaxă	scale(geometry, x_scale, y_scale, [center]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **x_scale** - x-axis scaling factor • **y_scale** - y-axis scaling factor • **center** - scaling center point. If not specified, the center of the geometry's bounding box is used.
Exemple	• `scale(@geometry, 2, 0.5, make_point(4, 5))` → geometry scaled twice horizontally and halved vertically, around the (4, 5) point • `scale(@geometry, 2, 0.5)` → geometry twice horizontally and halved vertically, around the center of its bounding box

segments_to_lines

Returns a multi line geometry consisting of a line for every segment in the input geometry.

Sintaxă	segments_to_lines(geometry)
Argumente	• **geometry** - geometry object
Exemple	• `geom_to_wkt(segments_to_lines(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)')))` → «MultiLineString ((0 0, 1 1),(1 1, 2 2))»

Mai multe detalii: Algoritmul *Explode lines*

shared_paths

Returns a collection containing paths shared by the two input geometries. Those going in the same direction are in the first element of the collection, those going in the opposite direction are in the second element. The paths themselves are given in the direction of the first geometry.

Sintaxă	shared_paths(geometry1, geometry2)
Argumente	• **geometry1** - a LineString/MultiLineString geometry • **geometry2** - a LineString/MultiLineString geometry
Exemple	• `geom_to_wkt(shared_paths(geom_from_wkt('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),(51 150,101 150,76 175,51 150))'),geom_from_wkt('LINESTRING(151 100,126 156.25,126 125,90 161, 76 175)')))` → «GeometryCollection (MultiLineString ((126 156.25, 126 125),(101 150, 90 161),(90 161, 76 175)),MultiLineString EMPTY)» • `geom_to_wkt(shared_paths(geom_from_wkt('LINESTRING(76 175,90 161,126 125,126 156.25,151 100)'), geom_from_wkt('MULTILINESTRING((26 125,26 200,126 200, 126 125,26 125),(51 150,101 150,76 175,51 150))')))` → «GeometryCollection (MultiLineString EMPTY,MultiLineString ((76 175, 90 161),(90 161, 101 150),(126 125, 126 156.25)))»

shortest_line

Returns the shortest line joining geometry1 to geometry2. The resultant line will start at geometry1 and end at geometry2.

Sintaxă	shortest_line(geometry1, geometry2)
Argumente	• **geometry1** - geometry to find shortest line from • **geometry2** - geometry to find shortest line to
Exemple	• `geom_to_wkt(shortest_line(geom_from_wkt('LINESTRING (20 80, 98 190, 110 180, 50 75 )'),geom_from_wkt('POINT(100 100)')))` → «LineString(73.0769 115.384, 100 100)»

simplify

Simplifies a geometry by removing nodes using a distance based threshold (ie, the Douglas Peucker algorithm). The algorithm preserves large deviations in geometries and reduces the number of vertices in nearly straight segments.

Sintaxă	simplify(geometry, tolerance)
Argumente	• **geometry** - a geometry • **tolerance** - maximum deviation from straight segments for points to be removed
Exemple	• `geom_to_wkt(simplify(geometry:=geom_from_wkt('LineString(0 0, 5 0.1, 10 0)'),tolerance:=5))` → «LineString(0 0, 10 0)»

Mai multe detalii: Algoritmul *Simplify*

simplify_vw

Simplifies a geometry by removing nodes using an area based threshold (ie, the Visvalingam-Whyatt algorithm). The algorithm removes vertices which create small areas in geometries, e.g., narrow spikes or nearly straight segments.

Sintaxă	simplify_vw(geometry, tolerance)
Argumente	• **geometry** - a geometry • **tolerance** - a measure of the maximum area created by a node for the node to be removed
Exemple	• `geom_to_wkt(simplify_vw(geometry:=geom_from_wkt('LineString(0 0, 5 0, 5.01 10, 5.02 0, 10 0)'),tolerance:=5))` → «LineString(0 0, 10 0)»

Mai multe detalii: Algoritmul *Simplify*

single_sided_buffer

Returns a geometry formed by buffering out just one side of a linestring geometry. Distances are in the Spatial Reference System of this geometry.

Sintaxă	single_sided_buffer(geometry, distance, [segments=8], [join=1], [miter_limit=2.0]) [] marks optional arguments
Argumente	• **geometry** - a (multi)linestring geometry • **distance** - buffer distance. Positive values will be buffered to the left of lines, negative values to the right • **segments** - number of segments to use to represent a quarter circle when a round join style is used. A larger number results in a smoother buffer with more nodes. • **join** - join style for corners, where 1 = round, 2 = miter and 3 = bevel • **miter_limit** - limit on the miter ratio used for very sharp corners (when using miter joins only)
Exemple	• `single_sided_buffer(@geometry, 10.5)` → line buffered to the left by 10.5 units • `single_sided_buffer(@geometry, -10.5)` → line buffered to the right by 10.5 units • `single_sided_buffer(@geometry, 10.5, segments:=16, join:=1)` → line buffered to the left by 10.5 units, using more segments to result in a smoother buffer • `single_sided_buffer(@geometry, 10.5, join:=3)` → line buffered to the left by 10.5 units, using a beveled join

Mai multe detalii: Algoritmul *Single sided buffer*

sinuosity

Returns the sinuosity of a curve, which is the ratio of the curve length to the straight (2D) distance between its endpoints.

Sintaxă	sinuosity(geometry)
Argumente	• **geometry** - Input curve (circularstring, linestring)
Exemple	• `round(sinuosity(geom_from_wkt('LINESTRING(2 0, 2 2, 3 2, 3 3)')), 3)` → 1.265 • `sinuosity(geom_from_wkt('LINESTRING( 3 1, 5 1)'))` → 1.0

smooth

Smooths a geometry by adding extra nodes which round off corners in the geometry. If input geometries contain Z or M values, these will also be smoothed and the output geometry will retain the same dimensionality as the input geometry.

Sintaxă	smooth(geometry, [iterations=1], [offset=0.25], [min_length=-1], [max_angle=180]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **iterations** - number of smoothing iterations to apply. Larger numbers result in smoother but more complex geometries. • **offset** - value between 0 and 0.5 which controls how tightly the smoothed geometry follow the original geometry. Smaller values result in a tighter smoothing, larger values result in looser smoothing. • **min_length** - minimum length of segments to apply smoothing to. This parameter can be used to avoid placing excessive additional nodes in shorter segments of the geometry. • **max_angle** - maximum angle at node for smoothing to be applied (0-180). By lowering the maximum angle intentionally sharp corners in the geometry can be preserved. For instance, a value of 80 degrees will retain right angles in the geometry.
Exemple	• `geom_to_wkt(smooth(geometry:=geom_from_wkt('LineString(0 0, 5 0, 5 5)'),iterations:=1,offset:=0.2,min_length:=-1, max_angle:=180))` → «LineString (0 0, 4 0, 5 1, 5 5)»

Mai multe detalii: Algoritmul *Smooth*

square_wave

Constructs square/rectangular waves along the boundary of a geometry.

Sintaxă	square_wave(geometry, wavelength, amplitude, [strict=False]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **wavelength** - wavelength of square waveform • **amplitude** - amplitude of square waveform • **strict** - By default the wavelength argument is treated as a „maximum wavelength", where the actual wavelength will be dynamically adjusted so that an exact number of square waves are created along the boundaries of the geometry. If the strict argument is set to true then the wavelength will be used exactly and an incomplete pattern may be used for the final waveform.
Exemple	• `square_wave(geom_from_wkt('LineString(0 0, 10 0)'), 3, 1)` → Square waves with wavelength 3 and amplitude 1 along the linestring

square_wave_randomized

Constructs randomized square/rectangular waves along the boundary of a geometry.

Sintaxă	square_wave_randomized(geometry, min_wavelength, max_wavelength, min_amplitude, max_amplitude, [seed=0]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **min_wavelength** - minimum wavelength of waves • **max_wavelength** - maximum wavelength of waves • **min_amplitude** - minimum amplitude of waves • **max_amplitude** - maximum amplitude of waves • **seed** - specifies a random seed for generating waves. If the seed is 0, then a completely random set of waves will be generated.
Exemple	• `square_wave_randomized(geom_from_wkt('LineString(0 0, 10 0)'), 2, 3, 0.1, 0.2)` → Randomly sized square waves with wavelengths between 2 and 3 and amplitudes between 0.1 and 0.2 along the linestring

start_point

Returns the first node from a geometry.

Sintaxă	start_point(geometry)
Argumente	• **geometry** - geometry object
Exemple	• `geom_to_wkt(start_point(geom_from_wkt('LINESTRING(4 0, 4 2, 0 2)')))` → «Point (4 0)»

Mai multe detalii: Algoritmul *Extract specific vertices*

straight_distance_2d

Returns the direct/euclidean distance between the first and last vertex of a geometry. The geometry must be a curve (circularstring, linestring).

Sintaxă	straight_distance_2d(geometry)
Argumente	• **geometry** - The geometry.
Exemple	• `straight_distance_2d(geom_from_wkt('LINESTRING(1 0, 1 1)'))` → 1 • `round(straight_distance_2d(geom_from_wkt('LINESTRING(1 4, 3 5, 5 0)')), 3)` → 5.657

Mai multe detalii: *length*

sym_difference

Returns a geometry that represents the portions of two geometries that do not intersect.

Sintaxă	sym_difference(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `geom_to_wkt( sym_difference( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 8 8)' ) ) )` → «LINESTRING(5 5, 8 8)»

Mai multe detalii: Algoritmul *Symmetrical difference*

tapered_buffer

Creates a buffer along a line geometry where the buffer diameter varies evenly over the length of the line.

Sintaxă	tapered_buffer(geometry, start_width, end_width, [segments=8]) [] marks optional arguments
Argumente	• **geometry** - input geometry. Must be a (multi)line geometry. • **start_width** - width of buffer at start of line, • **end_width** - width of buffer at end of line. • **segments** - number of segments to approximate quarter-circle curves in the buffer.
Exemple	• `tapered_buffer(geometry:=geom_from_wkt('LINESTRING(1 2, 4 2)'),start_width:=1,end_width:=2,segments:=8)` → A tapered buffer starting with a diameter of 1 and ending with a diameter of 2 along the linestring geometry.

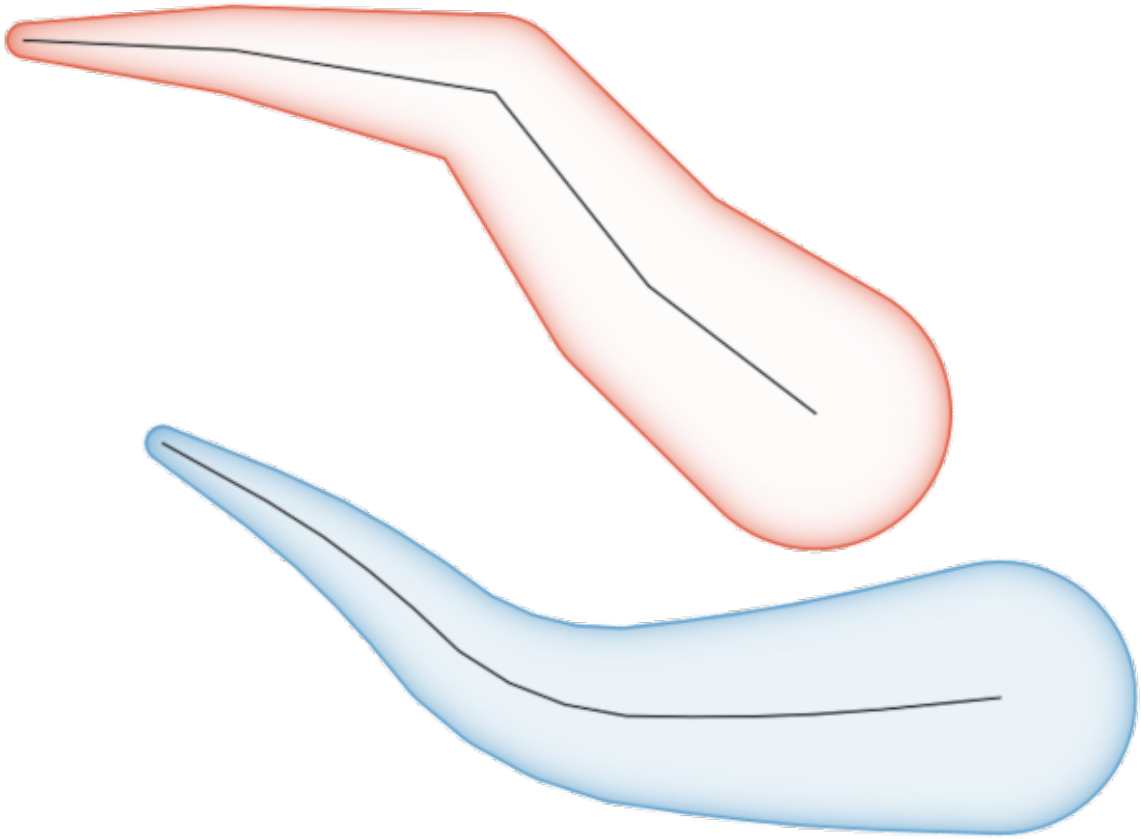


Fig. 13.8: Tapered buffer on line features

Mai multe detalii: Algoritmul *Tapered buffers*

touches

Tests whether a geometry touches another. Returns TRUE if the geometries have at least one point in common, but their interiors do not intersect.

Sintaxă	touches(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `touches( geom_from_wkt( 'LINESTRING(5 3, 4 4)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `touches( geom_from_wkt( 'POINT(4 4)' ), geom_from_wkt( 'POINT(5 5)' ) )` → FALSE

Mai multe detalii: *overlay_touches*

transform

Returns the geometry transformed from a source CRS to a destination CRS.

Sintaxă	transform(geometry, source_auth_id, dest_auth_id)
Argumente	• **geometry** - a geometry • **source_auth_id** - the source auth CRS ID • **dest_auth_id** - the destination auth CRS ID
Exemple	• `geom_to_wkt( transform( make_point(488995.53240249, 7104473.38600835), 'EPSG:2154', 'EPSG:4326' ) )` → «POINT(0 51)»

Mai multe detalii: Algoritmul *Reproject layer*

translate

Returns a translated version of a geometry. Calculations are in the Spatial Reference System of this geometry.

Sintaxă	translate(geometry, dx, dy)
Argumente	• **geometry** - a geometry • **dx** - delta x • **dy** - delta y
Exemple	• `translate(@geometry, 5, 10)` → a geometry of the same type like the original one

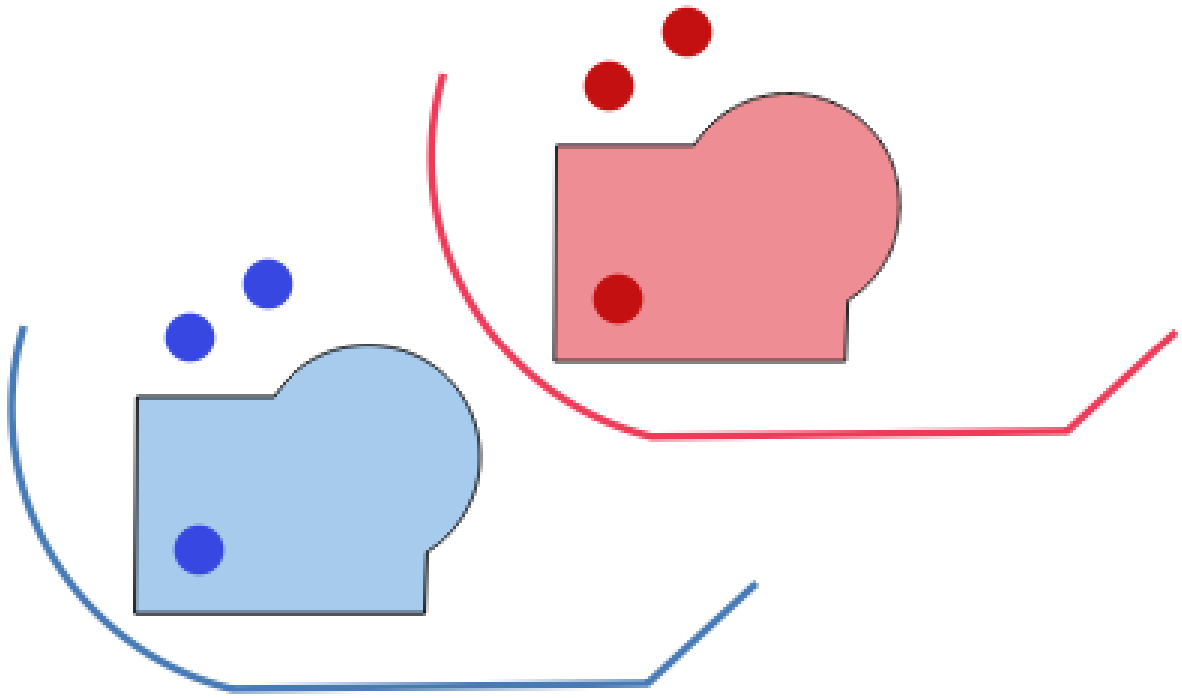


Fig. 13.9: Translating features

Mai multe detalii: Algoritmul *Translate*

triangular_wave

Constructs triangular waves along the boundary of a geometry.

Sintaxă	triangular_wave(geometry, wavelength, amplitude, [strict=False]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **wavelength** - wavelength of triangular waveform • **amplitude** - amplitude of triangular waveform • **strict** - By default the wavelength argument is treated as a „maximum wavelength", where the actual wavelength will be dynamically adjusted so that an exact number of triangular waves are created along the boundaries of the geometry. If the strict argument is set to true then the wavelength will be used exactly and an incomplete pattern may be used for the final waveform.
Exemple	• `triangular_wave(geom_from_wkt('LineString(0 0, 10 0)'), 3, 1)` → Triangular waves with wavelength 3 and amplitude 1 along the linestring

triangular_wave_randomized

Constructs randomized triangular waves along the boundary of a geometry.

Sintaxă	triangular_wave_randomized(geometry, min_wavelength, max_wavelength, min_amplitude, max_amplitude, [seed=0]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **min_wavelength** - minimum wavelength of waves • **max_wavelength** - maximum wavelength of waves • **min_amplitude** - minimum amplitude of waves • **max_amplitude** - maximum amplitude of waves • **seed** - specifies a random seed for generating waves. If the seed is 0, then a completely random set of waves will be generated.
Exemple	• `triangular_wave_randomized(geom_from_wkt('LineString(0 0, 10 0)'), 2, 3, 0.1, 0.2)` → Randomly sized triangular waves with wavelengths between 2 and 3 and amplitudes between 0.1 and 0.2 along the linestring

union

Returns a geometry that represents the point set union of the geometries.

Sintaxă	union(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `geom_to_wkt( union( make_point(4, 4), make_point(5, 5) ) )` → «MULTIPOINT(4 4, 5 5)»

wave

Constructs rounded (sine-like) waves along the boundary of a geometry.

Sintaxă	wave(geometry, wavelength, amplitude, [strict=False]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **wavelength** - wavelength of sine-like waveform • **amplitude** - amplitude of sine-like waveform • **strict** - By default the wavelength argument is treated as a „maximum wavelength”, where the actual wavelength will be dynamically adjusted so that an exact number of waves are created along the boundaries of the geometry. If the strict argument is set to true then the wavelength will be used exactly and an incomplete pattern may be used for the final waveform.
Exemple	• `wave(geom_from_wkt('LineString(0 0, 10 0)'), 3, 1)` → Sine-like waves with wavelength 3 and amplitude 1 along the linestring

wave_randomized

Constructs randomized curved (sine-like) waves along the boundary of a geometry.

Sintaxă	wave_randomized(geometry, min_wavelength, max_wavelength, min_amplitude, max_amplitude, [seed=0]) [] marks optional arguments
Argumente	• **geometry** - a geometry • **min_wavelength** - minimum wavelength of waves • **max_wavelength** - maximum wavelength of waves • **min_amplitude** - minimum amplitude of waves • **max_amplitude** - maximum amplitude of waves • **seed** - specifies a random seed for generating waves. If the seed is 0, then a completely random set of waves will be generated.
Exemple	• `wave_randomized(geom_from_wkt('LineString(0 0, 10 0)'), 2, 3, 0.1, 0.2)` → Randomly sized curved waves with wavelengths between 2 and 3 and amplitudes between 0.1 and 0.2 along the linestring

wedge_buffer

Returns a wedge shaped buffer originating from a point geometry.

Sintaxă	wedge_buffer(center, azimuth, width, outer_radius, [inner_radius=0.0]) [] marks optional arguments
Argumente	• **center** - center point (origin) of buffer. Must be a point geometry. • **azimuth** - angle (in degrees) for the middle of the wedge to point. • **width** - buffer width (in degrees). Note that the wedge will extend to half of the angular width either side of the azimuth direction. • **outer_radius** - outer radius for buffers • **inner_radius** - optional inner radius for buffers
Exemple	• `wedge_buffer(center:=geom_from_wkt('POINT(1 2)'), azimuth:=90,width:=180,outer_radius:=1)` → A wedge shaped buffer centered on the point (1,2), facing to the East, with a width of 180 degrees and outer radius of 1.

Mai multe detalii: Algoritmul *Create wedge buffers*

within

Tests whether a geometry is within another. Returns TRUE if the geometry1 is completely within geometry2.

Sintaxă	within(geometry1, geometry2)
Argumente	• **geometry1** - a geometry • **geometry2** - a geometry
Exemple	• `within( geom_from_wkt( 'POINT( 0.5 0.5)' ), geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))' ) )` → TRUE • `within( geom_from_wkt( 'POINT( 5 5 )' ), geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0 ))' ) )` → FALSE

Mai multe detalii: *overlay_within*

$x

Returns the x coordinate of the current point feature. If the feature is a multipoint feature, then the x-coordinate of the first point will be returned. **WARNING: This function is deprecated. It is recommended to use the replacement x() function with @geometry variable instead.**

Sintaxă	$x
Exemple	• `$x` → 42

Further reading: *x*

x

Returns the x coordinate of a point geometry, or the x coordinate of the centroid for a non-point geometry.

Sintaxă	x(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `x( geom_from_wkt( 'POINT(2 5)' ) )` → 2 • `x( @geometry )` → x coordinate of the current feature's centroid

$x_at

Retrieves a x coordinate of the current feature's geometry. **WARNING: This function is deprecated. It is recommended to use the replacement x_at function with @geometry variable instead.**

Sintaxă	$x_at(vertex)
Argumente	• **vertex** - index of the vertex of the current geometry (indices start at 0; negative values apply from the last index, starting at -1)
Exemple	• `$x_at(1)` → 5

Further reading: *x_at*

x_at

Retrieves a x coordinate of the geometry.

Sintaxă	x_at(geometry, vertex)
Argumente	• **geometry** - geometry object • **vertex** - index of the vertex of the geometry (indices start at 0; negative values apply from the last index, starting at -1)
Exemple	• `x_at( geom_from_wkt( 'POINT(4 5)' ), 0 )` → 4

x_max

Returns the maximum x coordinate of a geometry. Calculations are in the spatial reference system of this geometry.

Sintaxă	x_max(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `x_max( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 4

x_min

Returns the minimum x coordinate of a geometry. Calculations are in the spatial reference system of this geometry.

Sintaxă	x_min(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `x_min( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 2

$y

Returns the y coordinate of the current point feature. If the feature is a multipoint feature, then the y-coordinate of the first point will be returned. **WARNING: This function is deprecated. It is recommended to use the replacement y() function with @geometry variable instead.**

Sintaxă	$y
Exemple	• `$y` → 42

Further reading: *y*

y

Returns the y coordinate of a point geometry, or the y coordinate of the centroid for a non-point geometry.

Sintaxă	y(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `y( geom_from_wkt( 'POINT(2 5)' ) )` → 5 • `y( @geometry )` → y coordinate of the current feature's centroid

$y_at

Retrieves a y coordinate of the current feature's geometry. **WARNING: This function is deprecated. It is recommended to use the replacement y_at function with @geometry variable instead.**

Sintaxă	$y_at(vertex)
Argumente	• **vertex** - index of the vertex of the current geometry (indices start at 0; negative values apply from the last index, starting at -1)
Exemple	• `$y_at(1)` → 2

Further reading: *y_at*

y_at

Retrieves a y coordinate of the geometry.

Sintaxă	y_at(geometry, vertex)
Argumente	• **geometry** - geometry object • **vertex** - index of the vertex of the geometry (indices start at 0; negative values apply from the last index, starting at -1)
Exemple	• `y_at( geom_from_wkt( 'POINT(4 5)' ), 0 )` → 5

y_max

Returns the maximum y coordinate of a geometry. Calculations are in the spatial reference system of this geometry.

Sintaxă	y_max(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `y_max( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 8

y_min

Returns the minimum y coordinate of a geometry. Calculations are in the spatial reference system of this geometry.

Sintaxă	y_min(geometry)
Argumente	• **geometry** - a geometry
Exemple	• `y_min( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 5

$z

Returns the z value of the current point feature if it is 3D. If the feature is a multipoint feature, then the z value of the first point will be returned. **WARNING: This function is deprecated. It is recommended to use the replacement z() function with @geometry variable instead.**

Sintaxă	$z
Exemple	• `$z` → 123

z

Returns the z coordinate of a point geometry, or NULL if the geometry has no z value.

Sintaxă	z(geometry)
Argumente	• **geometry** - a point geometry
Exemple	• `z( geom_from_wkt( 'POINTZ(2 5 7)' ) )` → 7

z_at

Retrieves a z coordinate of the geometry, or NULL if the geometry has no z value.

Sintaxă	z_at(geometry, vertex)
Argumente	• **geometry** - geometry object • **vertex** - index of the vertex of the geometry (indices start at 0; negative values apply from the last index, starting at -1)
Exemple	• `z_at(geom_from_wkt('LineStringZ(0 0 0, 10 10 5, 10 10 0)'), 1)` → 5

z_max

Returns the maximum z coordinate of a geometry, or NULL if the geometry has no z value.

Sintaxă	z_max(geometry)
Argumente	• **geometry** - a geometry with z coordinate
Exemple	• `z_max( geom_from_wkt( 'POINT ( 0 0 1 )' ) )` → 1 • `z_max( geom_from_wkt( 'MULTIPOINT ( 0 0 1 , 1 1 3 )' ) )` → 3 • `z_max( make_line( make_point( 0,0,0 ), make_point( -1,-1,-2 ) ) )` → 0 • `z_max( geom_from_wkt( 'LINESTRING( 0 0 0, 1 0 2, 1 1 -1 )' ) )` → 2 • `z_max( geom_from_wkt( 'POINT ( 0 0 )' ) )` → NULL

z_min

Returns the minimum z coordinate of a geometry, or NULL if the geometry has no z value.

Sintaxă	z_min(geometry)
Argumente	• **geometry** - a geometry with z coordinate
Exemple	• `z_min( geom_from_wkt( 'POINT ( 0 0 1 )' ) )` → 1 • `z_min( geom_from_wkt( 'MULTIPOINT ( 0 0 1 , 1 1 3 )' ) )` → 1 • `z_min( make_line( make_point( 0,0,0 ), make_point( -1,-1,-2 ) ) )` → -2 • `z_min( geom_from_wkt( 'LINESTRING( 0 0 0, 1 0 2, 1 1 -1 )' ) )` → -1 • `z_min( geom_from_wkt( 'POINT ( 0 0 )' ) )` → NULL

13.2.14 Layout Functions

This group contains functions to manipulate print layout items properties.

item_variables

Returns a map of variables from a layout item inside this print layout.

Sintaxă	item_variables(id)
Argumente	• **id** - layout item ID
Exemple	• `map_get( item_variables('Map 0'), 'map_scale')` → scale of the item «Map 0» in the current print layout

Mai multe detalii: Lista variabilelor implicite :ref:` <expression_variables>`

map_credits

Returns a list of credit (usage rights) strings for the layers shown in a layout, or specific layout map item.

Sintaxă	map_credits([id], [include_layer_names=false], [layer_name_separator=»: «]) [] marks optional arguments
Argumente	• **id** - Map item ID. If not specified, the layers from all maps in the layout will be used. • **include_layer_names** - Set to true to include layer names before their credit strings • **layer_name_separator** - String to insert between layer names and their credit strings, if include_layer_names is true
Exemple	• `array_to_string( map_credits() )` → comma separated list of layer credits for all layers shown in all map items in the layout, e.g «CC-BY-NC, CC-BY-SA» • `array_to_string( map_credits( 'Main Map' ) )` → comma separated list of layer credits for layers shown in the «Main Map» layout item, e.g «CC-BY-NC, CC-BY-SA» • `array_to_string( map_credits( 'Main Map', include_layer_names := true, layer_name_separator := ': ' ) )` → comma separated list of layer names and their credits for layers shown in the «Main Map» layout item, e.g. «Railway lines: CC-BY-NC, Basemap: CC-BY-SA»

This function requires the *Access metadata properties* of the layers to have been filled.

13.2.15 Map Layers

This group contains a list of the available layers in the current project and, for each layer, their fields (stored in the dataset, virtual or auxiliary ones as well as from joins). The fields can be interacted the same way as mentioned in *Fields and Values*, except that a double-click will add the name as a string (single quoted) to the expression instead of as a field reference given that they do not belong to the active layer. This offers a convenient way to write expressions referring to different layers, such as when performing *aggregates*, *attribute* or *spatial* queries.

It also provides some convenient functions to manipulate layers.

decode_uri

Takes a layer and decodes the uri of the underlying data provider. It depends on the dataprovider, which data is available.

Sintaxă	decode_uri(layer, [part]) [] marks optional arguments
Argumente	• **layer** - The layer for which the uri should be decoded. • **part** - The part of the uri to return. If unspecified, a map with all uri parts will be returned.
Exemple	• `decode_uri(@layer)` → {«layerId»: «0», «layerName»: «», «path»: «/home/qgis/shapefile.shp»} • `decode_uri(@layer)` → {«layerId»: NULL, «layerName»: «layer», «path»: «/home/qgis/geopackage.gpkg»} • `decode_uri(@layer, 'path')` → «C:\my_data\qgis\shape.shp»

layer_property

Returns a matching layer property or metadata value.

Sintaxă	layer_property(layer, property)
Argumente	• **layer** - a string, representing either a layer name or layer ID • **property** - a string corresponding to the property to return. Valid options are: – name: layer name – id: layer ID – title: metadata title string – abstract: metadata abstract string – keywords: metadata keywords – data_url: metadata URL – attribution: metadata attribution string – attribution_url: metadata attribution URL – source: layer source – min_scale: minimum display scale for layer – max_scale: maximum display scale for layer – is_editable: if layer is in edit mode – crs: layer CRS – crs_definition: layer CRS full definition – crs_description: layer CRS description – crs_ellipsoid: acronym of the layer CRS ellipsoid – extent: layer extent (as a geometry object) – distance_units: layer distance units – type: layer type, e.g., Vector or Raster – storage_type: storage format (vector layers only) – geometry_type: geometry type, e.g., Point (vector layers only) – feature_count: approximate feature count for layer (vector layers only) – path: File path to the layer data source. Only available for file based layers.
Exemple	• `layer_property('streets','title')` → «Basemap Streets» • `layer_property('airports','feature_count')` → 120 • `layer_property('landsat','crs')` → «EPSG:4326»

Mai multe detalii: *vector*, *raster* and *mesh* layer properties

load_layer

Loads a layer by source URI and provider name.

Sintaxă	load_layer(uri, provider)
Argumente	• **uri** - layer source URI string • **provider** - layer data provider name
Exemple	• `layer_property(load_layer('c:/data/roads.shp', 'ogr'), 'feature_count')` → count of features from the c:/data/roads.shp vector layer

13.2.16 Maps Functions

This group contains functions to create or manipulate keys and values of map data structures (also known as dictionary objects, key-value pairs, or associative arrays). Unlike the *list data structure* where values order matters, the order of the key-value pairs in the map object is not relevant and values are identified by their keys.

from_json

Încarcă un șir formatat JSON.

Sintaxă	from_json(string)
Argumente	• **string** - șir JSON
Exemple	• `from_json('{"qgis":"rocks"}')` → { «qgis»: «rocks» } • `from_json('[1,2,3]')` → [1,2,3]

hstore_to_map

Creează o hartă dintr-un șir formatat în hstore.

Sintaxă	hstore_to_map(string)
Argumente	• **string** - șirul de intrare
Exemple	• `hstore_to_map('qgis=>rocks')` → { «qgis»: «rocks» }

map

Returnează o hartă care conține toate cheile și valorile transmise ca pereche de parametri.

Sintaxă	map(key1, value1, key2, value2, …)
Argumente	• **key** - o cheie (șir) • **value** - o valoare
Exemple	• `map('1','one','2', 'two')` → { «1»: «one», «2»: «two» } • `map('1','one','2', 'two')['1']` → «one»

map_akeys

Returnează toate cheile unei hărți într-un tablou.

Sintaxă	map_akeys(map)
Argumente	• **map** - o hartă
Exemple	• `map_akeys(map('1','one','2','two'))` → [«1», «2»]

map_avals

Returnează toate valorile unei hărți într-o matrice.

Sintaxă	map_avals(map)
Argumente	• **map** - o hartă
Exemple	• `map_avals(map('1','one','2','two'))` → [«one», «two»]

map_concat

Returnează o hartă care conține toate intrările hărților date. Dacă două hărți conțin aceeași cheie, se ia valoarea celei de-a doua hărți.

Sintaxă	map_concat(map1, map2, …)
Argumente	• **map** - o hartă
Exemple	• `map_concat(map('1','one', '2','overridden'),map('2','two', '3','three'))` → { «1»: «one», «2»: «two», «3»: «three» }

map_delete

Returnează o hartă cu cheia dată și valoarea ștearsă corespunzătoare ei.

Sintaxă	map_delete(map, key)
Argumente	• **map** - o hartă • **key** - cheia de șters
Exemple	• `map_delete(map('1','one','2','two'),'2')` → { «1»: «one» }

map_exist

Returns TRUE if the given key exists in the map.

Sintaxă	map_exist(map, key)
Argumente	• **map** - o hartă • **key** - cheia de căutare
Exemple	• `map_exist(map('1','one','2','two'),'3')` → FALSE

map_get

Returnează valoarea unei hărți, având în vedere cheia acesteia. Returnează NULL atunci când cheia nu există.

Sintaxă	map_get(map, key)
Argumente	• **map** - o hartă • **key** - cheia de căutare
Exemple	• `map_get(map('1','one','2','two'),'2')` → «two» • `map_get( item_variables('Map 0'), 'map_scale')` → scara elementului «Map 0» (dacă există) în compoziția curentă

Sugestie: You can also use the *index operator ([])* to get a value from a map.

map_insert

Returnează o hartă cu o cheie/o valoare adăugată. Atunci când cheia există deja, valoarea acesteia este suprascrisă.

Sintaxă	map_insert(map, key, value)
Argumente	• **map** - o hartă • **key** - cheia de adăugat • **value** - valoarea de adăugat
Exemple	• `map_insert(map('1','one'),'3','three')` → { «1»: «one», «3»: «three» } • `map_insert(map('1','one','2','overridden'),'2','two')` → { «1»: «one», «2»: «two» }

map_prefix_keys

Returns a map with all keys prefixed by a given string.

Sintaxă	map_prefix_keys(map, prefix)
Argumente	• **map** - o hartă • **prefix** - a string
Exemple	• `map_prefix_keys(map('1','one','2','two'), 'prefix-')` → { «prefix-1»: «one», «prefix-2»: «two» }

map_to_hstore

Îmbină elementele hărții într-un șir formatat hstore.

Sintaxă	map_to_hstore(map)
Argumente	• **map** - harta de intrare
Exemple	• `map_to_hstore(map('qgis','rocks'))` → «„qgis”=>”rocks”»

map_to_html_dl

Merge map elements into a HTML definition list string.

Sintaxă	map_to_html_dl(map)
Argumente	• **map** - harta de intrare
Exemple	• `map_to_html_dl(map('qgis','rocks'))` → <dl><dt>qgis</dt><dd>rocks</dd></dl>

map_to_html_table

Merge map elements into a HTML table string.

Sintaxă	map_to_html_table(map)
Argumente	• **map** - harta de intrare
Exemple	• `map_to_html_table(map('qgis','rocks'))` → <table><thead><th>qgis</th></thead><tbody><tr><td>rocks</td></tr></tbody></table>

to_json

Creează un șir formatat JSON dintr-o hartă, matrice sau altă valoare.

Sintaxă	to_json(value)
Argumente	• **value** - valoarea de intrare
Exemple	• `to_json(map('qgis','rocks'))` → {„qgis”:”rocks”} • `to_json(array(1,2,3))` → [1,2,3]

url_encode

Returns an URL encoded string from a map. Transforms all characters in their properly-encoded form producing a fully-compliant query string.

Note that the plus sign «+» is not converted.

Sintaxă	url_encode(map)
Argumente	• **map** - a map.
Exemple	• `url_encode(map('a&+b', 'a and plus b', 'a=b', 'a equals b'))` → «a%26+b=a%20and%20plus%20b&a%3Db=a%20equals%20b»

13.2.17 Funcțiile Matematice

Acest grup conține funcții matematice (ex.: rădăcina pătrată, sin și cos).

abs

Returns the absolute value of a number.

Sintaxă	abs(value)
Argumente	• **value** - a number
Exemple	• `abs(-2)` → 2

acos

Returns the inverse cosine of a value in radians.

Sintaxă	acos(value)
Argumente	• **value** - cosine of an angle in radians
Exemple	• `acos(0.5)` → 1.0471975511966

asin

Returns the inverse sine of a value in radians.

Sintaxă	asin(value)
Argumente	• **value** - sine of an angle in radians
Exemple	• `asin(1.0)` → 1.5707963267949

atan

Returns the inverse tangent of a value in radians.

Sintaxă	atan(value)
Argumente	• **value** - tan of an angle in radians
Exemple	• `atan(0.5)` → 0.463647609000806

atan2

Returns the inverse tangent of dy/dx by using the signs of the two arguments to determine the quadrant of the result.

Sintaxă	atan2(dy, dx)
Argumente	• **dy** - y coordinate difference • **dx** - x coordinate difference
Exemple	• `atan2(1.0, 1.732)` → 0.523611477769969

ceil

Rounds a number upwards.

Sintaxă	ceil(value)
Argumente	• **value** - a number
Exemple	• `ceil(4.9)` → 5 • `ceil(-4.9)` → -4

clamp

Restricts an input value to a specified range.

Sintaxă	clamp(minimum, input, maximum)
Argumente	• **minimum** - the smallest value *input* is allowed to take. • **input** - a value which will be restricted to the range specified by *minimum* and *maximum* • **maximum** - the largest value *input* is allowed to take
Exemple	• `clamp(1,5,10)` → 5 *input* is between 1 and 10 so is returned unchanged • `clamp(1,0,10)` → 1 *input* is less than minimum value of 1, so function returns 1 • `clamp(1,11,10)` → 10 *input* is greater than maximum value of 10, so function returns 10

cos

Returns cosine of an angle.

Sintaxă	cos(angle)
Argumente	• **angle** - angle in radians
Exemple	• `cos(1.571)` → 0.000796326710733263

degrees

Converts from radians to degrees.

Sintaxă	degrees(radians)
Argumente	• **radians** - numeric value
Exemple	• `degrees(3.14159)` → 180 • `degrees(1)` → 57.2958

exp

Returns exponential of an value.

Sintaxă	exp(value)
Argumente	• **value** - number to return exponent of
Exemple	• `exp(1.0)` → 2.71828182845905

floor

Rounds a number downwards.

Sintaxă	floor(value)
Argumente	• **value** - a number
Exemple	• `floor(4.9)` → 4 • `floor(-4.9)` → -5

ln

Returns the natural logarithm of a value.

Sintaxă	ln(value)
Argumente	• **value** - numeric value
Exemple	• `ln(1)` → 0 • `ln(2.7182818284590452354)` → 1

log

Returns the value of the logarithm of the passed value and base.

Sintaxă	log(base, value)
Argumente	• **base** - any positive number • **value** - any positive number
Exemple	• `log(2, 32)` → 5 • `log(0.5, 32)` → -5

log10

Returns the value of the base 10 logarithm of the passed expression.

Sintaxă	log10(value)
Argumente	• **value** - any positive number
Exemple	• `log10(1)` → 0 • `log10(100)` → 2

max

Returns the largest value in a set of values.

Sintaxă	max(value1, value2, …)
Argumente	• **value** - a number
Exemple	• `max(2,10.2,5.5)` → 10.2 • `max(20.5,NULL,6.2)` → 20.5

min

Returns the smallest value in a set of values.

Sintaxă	min(value1, value2, …)
Argumente	• **value** - a number
Exemple	• `min(20.5,10,6.2)` → 6.2 • `min(2,-10.3,NULL)` → -10.3

pi

Returns value of pi for calculations.

Sintaxă	pi()
Exemple	• `pi()` → 3.14159265358979

radians

Converts from degrees to radians.

Sintaxă	radians(degrees)
Argumente	• **degrees** - numeric value
Exemple	• `radians(180)` → 3.14159 • `radians(57.2958)` → 1

rand

Returns a random integer within the range specified by the minimum and maximum argument (inclusive). If a seed is provided, the returned will always be the same, depending on the seed.

Sintaxă	rand(min, max, [seed=NULL]) [] marks optional arguments
Argumente	• **min** - an integer representing the smallest possible random number desired • **max** - an integer representing the largest possible random number desired • **seed** - any value to use as seed
Exemple	• `rand(1, 10)` → 8

randf

Returns a random float within the range specified by the minimum and maximum argument (inclusive). If a seed is provided, the returned will always be the same, depending on the seed.

Sintaxă	randf([min=0.0], [max=1.0], [seed=NULL]) [] marks optional arguments
Argumente	• **min** - an float representing the smallest possible random number desired • **max** - an float representing the largest possible random number desired • **seed** - any value to use as seed
Exemple	• `randf(1, 10)` → 4.59258286403147

round

Rounds a number to number of decimal places.

Sintaxă	round(value, [places=0]) [] marks optional arguments
Argumente	• **value** - decimal number to be rounded • **places** - Optional integer representing number of places to round decimals to. Can be negative.
Exemple	• `round(1234.567, 2)` → 1234.57 • `round(1234.567)` → 1235 • `round(1234.567, -1)` → 1230

scale_exponential

Transforms a given value from an input domain to an output range using an exponential curve. This function can be used to ease values in or out of the specified output range.

Sintaxă	scale_exponential(value, domain_min, domain_max, range_min, range_max, exponent)
Argumente	• **value** - A value in the input domain. The function will return a corresponding scaled value in the output range. • **domain_min** - Specifies the minimum value in the input domain, the smallest value the input value should take. • **domain_max** - Specifies the maximum value in the input domain, the largest value the input value should take. • **range_min** - Specifies the minimum value in the output range, the smallest value which should be output by the function. • **range_max** - Specifies the maximum value in the output range, the largest value which should be output by the function. • **exponent** - A positive value (greater than 0), which dictates the way input values are mapped to the output range. Large exponents will cause the output values to «ease in», starting slowly before accelerating as the input values approach the domain maximum. Smaller exponents (less than 1) will cause output values to «ease out», where the mapping starts quickly but slows as it approaches the domain maximum.
Exemple	• `scale_exp(5,0,10,0,100,2)` → 25 easing in, using an exponent of 2 • `scale_exp(3,0,10,0,100,0.5)` → 54.772 easing out, using an exponent of 0.5

scale_linear

Transforms a given value from an input domain to an output range using linear interpolation.

Sintaxă	scale_linear(value, domain_min, domain_max, range_min, range_max)
Argumente	• **value** - A value in the input domain. The function will return a corresponding scaled value in the output range. • **domain_min** - Specifies the minimum value in the input domain, the smallest value the input value should take. • **domain_max** - Specifies the maximum value in the input domain, the largest value the input value should take. • **range_min** - Specifies the minimum value in the output range, the smallest value which should be output by the function. • **range_max** - Specifies the maximum value in the output range, the largest value which should be output by the function.
Exemple	• `scale_linear(5,0,10,0,100)` → 50 • `scale_linear(0.2,0,1,0,360)` → 72 scaling a value between 0 and 1 to an angle between 0 and 360 • `scale_linear(1500,1000,10000,9,20)` → 9.6111111 scaling a population which varies between 1000 and 10000 to a font size between 9 and 20

scale_polynomial

Transforms a given value from an input domain to an output range using a polynomial curve. This function can be used to ease values in or out of the specified output range.

Sintaxă	scale_polynomial(value, domain_min, domain_max, range_min, range_max, exponent)
Argumente	• **value** - A value in the input domain. The function will return a corresponding scaled value in the output range. • **domain_min** - Specifies the minimum value in the input domain, the smallest value the input value should take. • **domain_max** - Specifies the maximum value in the input domain, the largest value the input value should take. • **range_min** - Specifies the minimum value in the output range, the smallest value which should be output by the function. • **range_max** - Specifies the maximum value in the output range, the largest value which should be output by the function. • **exponent** - A positive value (greater than 0), which dictates the way input values are mapped to the output range. Large exponents will cause the output values to «ease in», starting slowly before accelerating as the input values approach the domain maximum. Smaller exponents (less than 1) will cause output values to «ease out», where the mapping starts quickly but slows as it approaches the domain maximum.
Exemple	• `scale_polynomial(5,0,10,0,100,2)` → 25 easing in, using an exponent of 2 • `scale_polynomial(3,0,10,0,100,0.5)` → 54.772 easing out, using an exponent of 0.5

sin

Returns the sine of an angle.

Sintaxă	sin(angle)
Argumente	• **angle** - angle in radians
Exemple	• `sin(1.571)` → 0.999999682931835

sqrt

Returns square root of a value.

Sintaxă	sqrt(value)
Argumente	• **value** - a number
Exemple	• `sqrt(9)` → 3

tan

Returns the tangent of an angle.

Sintaxă	tan(angle)
Argumente	• **angle** - angle in radians
Exemple	• `tan(1.0)` → 1.5574077246549

13.2.18 Meshes Functions

This group contains functions which calculate or return mesh related values.

$face_area

Returns the area of the current mesh face. The area calculated by this function respects both the current project's ellipsoid setting and area unit settings. For example, if an ellipsoid has been set for the project then the calculated area will be ellipsoidal, and if no ellipsoid is set then the calculated area will be planimetric.

Sintaxă	$face_area
Exemple	• `$face_area` → 42

$face_index

Returns the index of the current mesh face.

Sintaxă	$face_index
Exemple	• `$face_index` → 4581

$vertex_as_point

Returns the current vertex as a point geometry.

Sintaxă	$vertex_as_point
Exemple	• `geom_to_wkt( $vertex_as_point )` → «POINT(800 1500 41)»

$vertex_index

Returns the index of the current mesh vertex.

Sintaxă	$vertex_index
Exemple	• `$vertex_index` → 9874

$vertex_x

Returns the X coordinate of the current mesh vertex.

Sintaxă	$vertex_x
Exemple	• `$vertex_x` → 42.12

$vertex_y

Returns the Y coordinate of the current mesh vertex.

Sintaxă	$vertex_y
Exemple	• `$vertex_y` → 12.24

$vertex_z

Returns the Z value of the current mesh vertex.

Sintaxă	$vertex_z
Exemple	• `$vertex_z` → 42

13.2.19 Operatori

Acest grup cuprinde operatorii (ex.: +, -, *). De notat că pentru majoritatea funcțiilor matematice de mai jos, în cazul în care una dintre intrări are valoarea NULL, atunci rezultatul este NULL.

%

Remainder of division. Takes the sign of the dividend.

Sintaxă	a % b
Argumente	• **a** - value • **b** - value
Exemple	• `9 % 2` → 1 • `9 % -2` → 1 • `-9 % 2` → -1 • `5 % NULL` → NULL

*

Multiplication of two values

Sintaxă	a * b
Argumente	• **a** - value • **b** - value
Exemple	• `5 * 4` → 20 • `5 * NULL` → NULL

+

Addition of two values. If one of the values is NULL the result will be NULL.

Sintaxă	a + b
Argumente	• **a** - value • **b** - value
Exemple	• `5 + 4` → 9 • `5 + NULL` → NULL • `'QGIS ' + 'ROCKS'` → «QGIS ROCKS» • `to_datetime('2020-08-01 12:00:00') + '1 day 2 hours'` → 2020-08-02T14:00:00

Mai multe detalii: *concat*, ||

-

Subtraction of two values. If one of the values is NULL the result will be NULL.

Sintaxă	a - b
Argumente	• **a** - value • **b** - value
Exemple	• `5 - 4` → 1 • `5 - NULL` → NULL • `to_datetime('2012-05-05 12:00:00') - to_interval('1 day 2 hours')` → 2012-05-04T10:00:00

/

Division of two values

Sintaxă	a / b
Argumente	• **a** - value • **b** - value
Exemple	• `5 / 4` → 1.25 • `5 / NULL` → NULL

//

Floor division of two values

Sintaxă	a // b
Argumente	• **a** - value • **b** - value
Exemple	• `9 // 2` → 4

<

Compares two values and evaluates to 1 if the left value is less than the right value.

Sintaxă	a < b
Argumente	• **a** - value • **b** - value
Exemple	• `5 < 4` → FALSE • `5 < 5` → FALSE • `4 < 5` → TRUE

<=

Compares two values and evaluates to 1 if the left value is less or equal than the right value.

Sintaxă	a <= b
Argumente	• **a** - value • **b** - value
Exemple	• `5 <= 4` → FALSE • `5 <= 5` → TRUE • `4 <= 5` → TRUE

<>

Compares two values and evaluates to 1 if they are not equal.

Sintaxă	a <> b
Argumente	• **a** - value • **b** - value
Exemple	• `5 <> 4` → TRUE • `4 <> 4` → FALSE • `5 <> NULL` → NULL • `NULL <> NULL` → NULL

=

Compares two values and evaluates to 1 if they are equal.

Sintaxă	a = b
Argumente	• **a** - value • **b** - value
Exemple	• `5 = 4` → FALSE • `4 = 4` → TRUE • `5 = NULL` → NULL • `NULL = NULL` → NULL

>

Compares two values and evaluates to 1 if the left value is greater than the right value.

Sintaxă	a > b
Argumente	• **a** - value • **b** - value
Exemple	• `5 > 4` → TRUE • `5 > 5` → FALSE • `4 > 5` → FALSE

>=

Compares two values and evaluates to 1 if the left value is greater or equal than the right value.

Sintaxă	a >= b
Argumente	• **a** - value • **b** - value
Exemple	• `5 >= 4` → TRUE • `5 >= 5` → TRUE • `4 >= 5` → FALSE

AND

Returns TRUE when conditions a and b are true.

Sintaxă	a AND b
Argumente	• **a** - condition • **b** - condition
Exemple	• `TRUE AND TRUE` → TRUE • `TRUE AND FALSE` → FALSE • `4 = 2+2 AND 1 = 1` → TRUE • `4 = 2+2 AND 1 = 2` → FALSE

BETWEEN

Returns TRUE if value is within the specified range. The range is considered inclusive of the bounds. To test for exclusion NOT BETWEEN can be used.

Sintaxă	value BETWEEN lower_bound AND higher_bound
Argumente	• **value** - the value to compare with a range. It can be a string, a number or a date. • **lower_bound AND higher_bound** - range bounds
Exemple	• `'B' BETWEEN 'A' AND 'C'` → TRUE • `2 BETWEEN 1 AND 3` → TRUE • `2 BETWEEN 2 AND 3` → TRUE • `'B' BETWEEN 'a' AND 'c'` → FALSE • `lower('B') BETWEEN 'a' AND 'b'` → TRUE

Notă: *value BETWEEN lower_bound AND higher_bound* is the same as „*value* >= *lower_bound AND value* <= *higher_bound*”.

Mai multe detalii: *NOT BETWEEN*

ILIKE

Returns TRUE if the first parameter matches case-insensitive the supplied pattern. LIKE can be used instead of ILIKE to make the match case-sensitive. Works with numbers also.

Sintaxă	string/number ILIKE pattern
Argumente	• **string/number** - string to search • **pattern** - pattern to find, you can use «%» as a wildcard, «_» as a single char and «\\» to escape these special characters.
Exemple	• `'A' ILIKE 'A'` → TRUE • `'A' ILIKE 'a'` → TRUE • `'A' ILIKE 'B'` → FALSE • `'ABC' ILIKE 'b'` → FALSE • `'ABC' ILIKE 'B'` → FALSE • `'ABC' ILIKE '_b_'` → TRUE • `'ABC' ILIKE '_B_'` → TRUE • `'ABCD' ILIKE '_b_'` → FALSE • `'ABCD' ILIKE '_B_'` → FALSE • `'ABCD' ILIKE '_b%'` → TRUE • `'ABCD' ILIKE '_B%'` → TRUE • `'ABCD' ILIKE '%b%'` → TRUE • `'ABCD' ILIKE '%B%'` → TRUE • `'ABCD%' ILIKE 'abcd\\%'` → TRUE • `'ABCD' ILIKE '%B\\%'` → FALSE

IN

Returns TRUE if value is found within a list of values.

Sintaxă	a IN b
Argumente	• **a** - value • **b** - list of values
Exemple	• `'A' IN ('A','B')` → TRUE • `'A' IN ('C','B')` → FALSE

IS

Returns TRUE if a is the same as b.

Sintaxă	a IS b
Argumente	• **a** - any value • **b** - any value
Exemple	• `'A' IS 'A'` → TRUE • `'A' IS 'a'` → FALSE • `4 IS 4` → TRUE • `4 IS 2+2` → TRUE • `4 IS 2` → FALSE • `@geometry IS NULL` → 0, if your geometry is not NULL

IS NOT

Returns TRUE if a is not the same as b.

Sintaxă	a IS NOT b
Argumente	• **a** - value • **b** - value
Exemple	• `'a' IS NOT 'b'` → TRUE • `'a' IS NOT 'a'` → FALSE • `4 IS NOT 2+2` → FALSE

LIKE

Returns TRUE if the first parameter matches the supplied pattern. Works with numbers also.

Sintaxă	string/number LIKE pattern
Argumente	• **string/number** - value • **pattern** - pattern to compare value with, you can use «%» as a wildcard, «_» as a single char and «\\» to escape these special characters.
Exemple	• `'A' LIKE 'A'` → TRUE • `'A' LIKE 'a'` → FALSE • `'A' LIKE 'B'` → FALSE • `'ABC' LIKE 'B'` → FALSE • `'ABC' LIKE '_B_'` → TRUE • `'ABCD' LIKE '_B_'` → FALSE • `'ABCD' LIKE '_B%'` → TRUE • `'ABCD' LIKE '%B%'` → TRUE • `'1%' LIKE '1\\%'` → TRUE • `'1_' LIKE '1\\%'` → FALSE

NOT

Negates a condition.

Sintaxă	NOT a
Argumente	• **a** - condition
Exemple	• `NOT 1` → FALSE • `NOT 0` → TRUE

NOT BETWEEN

Returns TRUE if value is not within the specified range. The range is considered inclusive of the bounds.

Sintaxă	value NOT BETWEEN lower_bound AND higher_bound
Argumente	• **value** - the value to compare with a range. It can be a string, a number or a date. • **lower_bound AND higher_bound** - range bounds
Exemple	• `'B' NOT BETWEEN 'A' AND 'C'` → FALSE • `1.0 NOT BETWEEN 1.1 AND 1.2` → TRUE • `2 NOT BETWEEN 2 AND 3` → FALSE • `'B' NOT BETWEEN 'a' AND 'c'` → TRUE • `lower('B') NOT BETWEEN 'a' AND 'b'` → FALSE

Notă: *value NOT BETWEEN lower_bound AND higher_bound* is the same as „*value* < *lower_bound OR value* > *higher_bound*”.

Mai multe detalii: *BETWEEN*

OR

Returns TRUE when condition a or b is true.

Sintaxă	a OR b
Argumente	• **a** - condition • **b** - condition
Exemple	• `4 = 2+2 OR 1 = 1` → TRUE • `4 = 2+2 OR 1 = 2` → TRUE • `4 = 2 OR 1 = 2` → FALSE

[]

Index operator. Returns an element from an array or map value.

Sintaxă	[index]
Argumente	• **index** - array index or map key value
Exemple	• `array(1,2,3)[0]` → 1 • `array(1,2,3)[2]` → 3 • `array(1,2,3)[-1]` → 3 • `map('a',1,'b',2)['a']` → 1 • `map('a',1,'b',2)['b']` → 2

Mai multe detalii: *array_get*, *map_get*

^

Power of two values.

Sintaxă	a ^ b
Argumente	• **a** - value • **b** - value
Exemple	• `5 ^ 4` → 625 • `5 ^ NULL` → NULL

||

Joins two values together into a string.

If one of the values is NULL the result will be NULL. See the CONCAT function for a different behavior.

Sintaxă	a || b
Argumente	• **a** - value • **b** - value
Exemple	• `'Here' || ' and ' || 'there'` → «Here and there» • `'Nothing' || NULL` → NULL • `'Dia: ' || "Diameter"` → «Dia: 25» • `1 || 2` → «12»

Mai multe detalii: *concat*, *+*

~

Performs a regular expression match on a string value. Backslash characters must be double escaped (e.g., „\\s” to match a white space character).

Sintaxă	string ~ regex
Argumente	• **string** - A string value • **regex** - A regular expression. Slashes must be escaped, eg \\d.
Exemple	• `'hello' ~ 'll'` → TRUE • `'hello' ~ '^ll'` → FALSE • `'hello' ~ 'llo$'` → TRUE • `'abc123' ~ '\\d+'` → TRUE

Mai multe detalii: *regexp_match*

13.2.20 Processing Functions

This group contains functions that operate on processing algorithms.

- *parameter*

parameter

Returns the value of a processing algorithm input parameter.

Sintaxă	parameter(name)
Argumente	• **name** - name of the corresponding input parameter
Exemple	• `parameter('BUFFER_SIZE')` → 5.6

13.2.21 Rasters Functions

This group contains functions to operate on raster layer.

- *raster_attributes*
- *raster_statistic*
- *raster_value*

raster_attributes

Returns a map with the fields names as keys and the raster attribute table values as values from the attribute table entry that matches the given raster value.

Sintaxă	raster_attributes(layer, band, value)
Argumente	• **layer** - the name or id of a raster layer • **band** - the band number for the associated attribute table lookup. • **value** - raster value
Exemple	• `raster_attributes('vegetation', 1, raster_value('vegetation', 1, make_point(1,1)))` → {«class»: «Vegetated», «subclass»: «Trees»}

raster_statistic

Returns statistics from a raster layer.

Sintaxă	raster_statistic(layer, band, property)
Argumente	• **layer** - a string, representing either a raster layer name or layer ID • **band** - integer representing the band number from the raster layer, starting at 1 • **property** - a string corresponding to the property to return. Valid options are: – min: minimum value – max: maximum value – avg: average (mean) value – stdev: standard deviation of values – range: range of values (max - min) – sum: sum of all values from raster
Exemple	• `raster_statistic('lc',1,'avg')` → Average value from band 1 from «lc» raster layer • `raster_statistic('ac2010',3,'min')` → Minimum value from band 3 from «ac2010» raster layer

raster_value

Returns the raster value found at the provided point.

Sintaxă	raster_value(layer, band, point)
Argumente	• **layer** - the name or id of a raster layer • **band** - the band number to sample the value from. • **point** - point geometry (for multipart geometries having more than one part, a NULL value will be returned)
Exemple	• `raster_value('dem', 1, make_point(1,1))` → 25

13.2.22 Record and Attributes Functions

Acest grup conține funcții care operează asupra identificatorilor de înregistrare.

attribute

Returns an attribute from a feature.

Variant 1

Returns the value of an attribute from the current feature.

Sintaxă	attribute(attribute_name)
Argumente	• **attribute_name** - name of attribute to be returned
Exemple	• `attribute( 'name' )` → value stored in «name» attribute for the current feature

Variant 2

Allows the target feature and attribute name to be specified.

Sintaxă	attribute(feature, attribute_name)
Argumente	• **feature** - a feature • **attribute_name** - name of attribute to be returned
Exemple	• `attribute( @atlas_feature, 'name' )` → value stored in «name» attribute for the current atlas feature

attributes

Returns a map containing all attributes from a feature, with field names as map keys.

Variant 1

Returns a map of all attributes from the current feature.

Sintaxă	attributes()
Exemple	• `attributes()['name']` → value stored in «name» attribute for the current feature

Variant 2

Allows the target feature to be specified.

Sintaxă	attributes(feature)
Argumente	• **feature** - a feature
Exemple	• `attributes( @atlas_feature )['name']` → value stored in «name» attribute for the current atlas feature

Mai multe detalii: *Maps Functions*

$currentfeature

Returns the current feature being evaluated. This can be used with the «attribute» function to evaluate attribute values from the current feature. **WARNING: This function is deprecated. It is recommended to use the replacement @feature variable instead.**

Sintaxă	$currentfeature
Exemple	• `attribute( $currentfeature, 'name' )` → value stored in «name» attribute for the current feature

display_expression

Returns the display expression for a given feature in a layer. The expression is evaluated by default. Can be used with zero, one or more arguments, see below for details.

No parameters

If called with no parameters, the function will evaluate the display expression of the current feature in the current layer.

Sintaxă	display_expression()
Exemple	• `display_expression()` → The display expression of the current feature in the current layer.

One «feature» parameter

If called with a «feature» parameter only, the function will evaluate the specified feature from the current layer.

Sintaxă	display_expression(feature)
Argumente	• **feature** - The feature which should be evaluated.
Exemple	• `display_expression(@atlas_feature)` → The display expression of the current atlas feature.

Layer and feature parameters

If the function is called with both a layer and a feature, it will evaluate the specified feature from the specified layer.

Sintaxă	display_expression(layer, feature, [evaluate=true]) [] marks optional arguments
Argumente	• **layer** - The layer (or its ID or name) • **feature** - The feature which should be evaluated. • **evaluate** - If the expression must be evaluated. If false, the expression will be returned as a string literal only (which could potentially be later evaluated using the «eval» function).
Exemple	• `display_expression( 'streets', get_feature_by_id('streets', 1))` → The display expression of the feature with the ID 1 on the layer «streets». • `display_expression('a_layer_id', @feature, 'False')` → The display expression of the given feature not evaluated.

feature_id

Returns a feature's unique ID, or NULL if the feature is not valid.

Sintaxă	feature_id(feature)
Argumente	• **feature** - a feature object
Exemple	• `feature_id( @feature )` → the ID of the current feature

Further reading: *get_feature_by_id*

get_feature

Returns the first feature of a layer matching a given attribute value.

Single value variant

Along with the layer ID, a single column and value are specified.

Sintaxă	get_feature(layer, attribute, value)
Argumente	• **layer** - layer name or ID • **attribute** - attribute name to use for the match • **value** - attribute value to match
Exemple	• `get_feature('streets','name','main st')` → first feature found in „streets" layer with „main st" value in the „name" field

Map variant

Along with the layer ID, a map containing the columns (key) and their respective value to be used.

Sintaxă	get_feature(layer, attribute)
Argumente	• **layer** - layer name or ID • **attribute** - Map containing the column and value pairs to use
Exemple	• `get_feature('streets',map('name','main st','lane_num','4'))` → first feature found in „streets” layer with „main st” value in the „name” field and „4” value in the „lane_num” field

get_feature_by_id

Returns the feature with an id on a layer.

Sintaxă	get_feature_by_id(layer, feature_id)
Argumente	• **layer** - layer, layer name or layer id • **feature_id** - the id of the feature which should be returned
Exemple	• `get_feature_by_id('streets', 1)` → the feature with the id 1 on the layer „streets”

Further reading: *feature_id*

$id

Returns the feature id of the current row. **WARNING: This function is deprecated. It is recommended to use the replacement @id variable instead.**

Sintaxă	$id
Exemple	• `$id` → 42

Further reading: *feature_id*, *get_feature_by_id*

is_attribute_valid

Returns TRUE if a specific feature attribute meets all constraints.

Sintaxă	is_attribute_valid(attribute, [feature], [layer], [strength]) [] marks optional arguments
Argumente	• **attribute** - an attribute name • **feature** - A feature. If not set, the current feature will be used. • **layer** - A vector layer. If not set, the current layer will be used. • **strength** - Set to «hard» or «soft» to narrow down to a specific constraint type. If not set, the function will return FALSE if either a hard or a soft constraint fails.
Exemple	• `is_attribute_valid('HECTARES')` → TRUE if the current feature's value in the „HECTARES" field meets all constraints. • `is_attribute_valid('HOUSES',get_feature('my_layer', 'FID', 10), 'my_layer')` → FALSE if the value in the „HOUSES" field from the feature with „FID"=10 in «my_layer» fails to meet all constraints.

Further reading: *Constraints*

is_feature_valid

Returns TRUE if a feature meets all field constraints.

Sintaxă	is_feature_valid([feature], [layer], [strength]) [] marks optional arguments
Argumente	• **feature** - A feature. If not set, the current feature will be used. • **layer** - A vector layer. If not set, the current layer will be used. • **strength** - Set to «hard» or «soft» to narrow down to a specific constraint type. If not set, the function will return FALSE if either a hard or a soft constraint fails.
Exemple	• `is_feature_valid(strength:='hard')` → TRUE if all fields from the current feature meet their hard constraints. • `is_feature_valid(get_feature('my_layer', 'FID', 10), 'my_layer')` → FALSE if all fields from feature with „FID"=10 in «my_layer» fails to meet all constraints.

Further reading: *Constraints*

is_selected

Returns TRUE if a feature is selected. Can be used with zero, one or two arguments, see below for details.

No parameters

If called with no parameters, the function will return TRUE if the current feature in the current layer is selected.

Sintaxă	is_selected()
Exemple	• `is_selected()` → TRUE if the current feature in the current layer is selected.

One «feature» parameter

If called with a «feature» parameter only, the function returns TRUE if the specified feature from the current layer is selected.

Sintaxă	is_selected(feature)
Argumente	• **feature** - The feature which should be checked for selection.
Exemple	• `is_selected(@atlas_feature)` → TRUE if the current atlas feature is selected. • `is_selected(get_feature('streets', 'name', 'Main St.'))` → TRUE if the unique named „Main St." feature on the active „streets" layer is selected. • `is_selected(get_feature_by_id('streets', 1))` → TRUE if the feature with the id 1 on the active „streets" layer is selected.

Two parameters

If the function is called with both a layer and a feature, it will return TRUE if the specified feature from the specified layer is selected.

Sintaxă	is_selected(layer, feature)
Argumente	• **layer** - The layer (its ID or name) on which the selection will be checked. • **feature** - The feature which should be checked for selection.
Exemple	• `is_selected( 'streets', get_feature('streets', 'name', "street_name"))` → TRUE if the current building's street is selected (assuming the building layer has a field named «street_name» and the «streets» layer has a field called «name» with unique values). • `is_selected( 'streets', get_feature_by_id('streets', 1))` → TRUE if the feature with the id 1 on the „streets" layer is selected.

maptip

Returns the maptip for a given feature in a layer. The expression is evaluated by default. Can be used with zero, one or more arguments, see below for details.

No parameters

If called with no parameters, the function will evaluate the maptip of the current feature in the current layer.

Sintaxă	maptip()
Exemple	• `maptip()` → The maptip of the current feature in the current layer.

One «feature» parameter

If called with a «feature» parameter only, the function will evaluate the specified feature from the current layer.

Sintaxă	maptip(feature)
Argumente	• **feature** - The feature which should be evaluated.
Exemple	• `maptip(@atlas_feature)` → The maptip of the current atlas feature.

Layer and feature parameters

If the function is called with both a layer and a feature, it will evaluate the specified feature from the specified layer.

Sintaxă	maptip(layer, feature, [evaluate=true]) [] marks optional arguments
Argumente	• **layer** - The layer (or its ID or name) • **feature** - The feature which should be evaluated. • **evaluate** - If the expression must be evaluated. If false, the expression will be returned as a string literal only (which could potentially be later evaluated using the «eval_template» function).
Exemple	• `maptip('streets', get_feature_by_id('streets', 1))` → The maptip of the feature with the ID 1 on the layer «streets». • `maptip('a_layer_id', @feature, 'False')` → The maptip of the given feature not evaluated.

num_selected

Returns the number of selected features on a given layer. By default works on the layer on which the expression is evaluated.

Sintaxă	num_selected([layer=current layer]) [] marks optional arguments
Argumente	• **layer** - The layer (or its id or name) on which the selection will be checked.
Exemple	• `num_selected()` → The number of selected features on the current layer. • `num_selected('streets')` → The number of selected features on the layer streets

represent_attributes

Returns a map with the attribute names as keys and the configured representation values as values. The representation value for the attributes depends on the configured widget type for each attribute. Can be used with zero, one or more arguments, see below for details.

No parameters

If called with no parameters, the function will return the representation of the attributes of the current feature in the current layer.

Sintaxă	represent_attributes()
Exemple	• `represent_attributes()` → The representation of the attributes for the current feature.

One «feature» parameter

If called with a «feature» parameter only, the function will return the representation of the attributes of the specified feature from the current layer.

Sintaxă	represent_attributes(feature)
Argumente	• **feature** - The feature which should be evaluated.
Exemple	• `represent_attributes(@atlas_feature)` → The representation of the attributes for the specified feature from the current layer.

Layer and feature parameters

If called with a «layer» and a «feature» parameter, the function will return the representation of the attributes of the specified feature from the specified layer.

Sintaxă	represent_attributes(layer, feature)
Argumente	• **layer** - The layer (or its ID or name). • **feature** - The feature which should be evaluated.
Exemple	• `represent_attributes('atlas_layer', @atlas_feature)` → The representation of the attributes for the specified feature from the specified layer.

Mai multe detalii: *represent_value*

represent_value

Returns the configured representation value for a field value. It depends on the configured widget type. Often, this is useful for «Value Map» widgets.

Sintaxă	represent_value(value, [fieldName]) [] marks optional arguments
Argumente	• **value** - The value which should be resolved. Most likely a field. • **fieldName** - The field name for which the widget configuration should be loaded.
Exemple	• `represent_value("field_with_value_map")` → Description for value • `represent_value('static value', 'field_name')` → Description for static value

Mai multe detalii: *widget types*, *represent_attributes*

sqlite_fetch_and_increment

Manage autoincrementing values in sqlite databases.

SQlite default values can only be applied on insert and not prefetched.

This makes it impossible to acquire an incremented primary key via AUTO_INCREMENT before creating the row in the database. Sidenote: with postgres, this works via the option *evaluate default values*.

When adding new features with relations, it is really nice to be able to already add children for a parent, while the parents form is still open and hence the parent feature uncommitted.

To get around this limitation, this function can be used to manage sequence values in a separate table on sqlite based formats like gpkg.

The sequence table will be filtered for a sequence id (filter_attribute and filter_value) and the current value of the id_field will be incremented by 1 and the incremented value returned.

If additional columns require values to be specified, the default_values map can be used for this purpose.

Note

This function modifies the target sqlite table. It is intended for usage with default value configurations for attributes.

When the database parameter is a layer and the layer is in transaction mode, the value will only be retrieved once during the lifetime of a transaction and cached and incremented. This makes it unsafe to work on the same database from several processes in parallel.

Sintaxă	sqlite_fetch_and_increment(database, table, id_field, filter_attribute, filter_value, [default_values]) [] marks optional arguments
Argumente	• **database** - Path to the sqlite file or geopackage layer • **table** - Name of the table that manages the sequences • **id_field** - Name of the field that contains the current value • **filter_attribute** - Name the field that contains a unique identifier for this sequence. Must have a UNIQUE index. • **filter_value** - Name of the sequence to use. • **default_values** - Map with default values for additional columns on the table. The values need to be fully quoted. Functions are allowed.
Exemple	• `sqlite_fetch_and_increment(@layer, 'sequence_table', 'last_unique_id', 'sequence_id', 'global', map('last_change', 'date(''now'')', 'user', '''' || @user_account_name || '''')) → 0` • `sqlite_fetch_and_increment(layer_property(@layer, 'path'), 'sequence_table', 'last_unique_id', 'sequence_id', 'global', map('last_change', 'date(''now'')', 'user', '''' || @user_account_name || '''')) → 0`

Mai multe detalii: *Data Sources Properties*, *Setting relations between multiple layers*

uuid

Generates a Universally Unique Identifier (UUID) for each row using the Qt QUuid::createUuid method.

Sintaxă	uuid([format=»WithBraces»]) [] marks optional arguments
Argumente	• **format** - The format, as the UUID will be formatted. «WithBraces», «WithoutBraces» or «Id128».
Exemple	• `uuid()` → «{0bd2f60f-f157-4a6d-96af-d4ba4cb366a1}» • `uuid('WithoutBraces')` → «0bd2f60f-f157-4a6d-96af-d4ba4cb366a1» • `uuid('Id128')` → «0bd2f60ff1574a6d96afd4ba4cb366a1»

13.2.23 Relations

This group contains the list of the *relations* available in the current project, with their description. It provides a quick access to the relation ID for writing an expression (with e.g. the *relation_aggregate* function) or customizing a form.

13.2.24 Sensors Functions

This group contains functions to interact with sensors.

sensor_data

Returns the last captured value (or values as a map for sensors which report multiple values) from a registered sensor.

Sintaxă	sensor_data(name, [expiration]) [] marks optional arguments
Argumente	• **name** - the sensor name • **expiration** - maximum millisecond since last captured value allowed
Exemple	• `sensor_data('geiger_1')` → «2000»

13.2.25 Funcții pentru Șiruri

Acest grup conține funcții care operează asupra șirurilor, (de ex: înlocuirea, conversia în majuscule).

ascii

Returns the unicode code associated with the first character of a string.

Sintaxă	ascii(string)
Argumente	• **string** - the string to convert to unicode code
Exemple	• `ascii('Q')` → 81

char

Returns the character associated with a unicode code.

Sintaxă	char(code)
Argumente	• **code** - a unicode code number
Exemple	• `char(81)` → «Q»

concat

Concatenates several strings to one. NULL values are converted to empty strings. Other values (like numbers) are converted to strings.

Sintaxă	concat(string1, string2, …)
Argumente	• **string** - a string value
Exemple	• `concat('sun', 'set')` → «sunset» • `concat('a','b','c','d','e')` → «abcde» • `concat('Anno ', 1984)` → «Anno 1984» • `concat('The Wall', NULL)` → «The Wall»

About fields concatenation

You can also concatenate strings or field values using either `||` or + operators, with some special characteristics:

- The + operator also means sum up expression, so if you have an integer (field or numeric value) operand, this can be error prone and you better use the others:

```
'My feature id is: ' + "gid" => triggers an error as gid returns an integer
```

- When any of the arguments is a NULL value, either `||` or + will return a NULL value. To return the other arguments regardless the NULL value, you may want to use the `concat` function:

```
'My feature id is: ' + NULL ==> NULL
'My feature id is: ' || NULL => NULL
concat('My feature id is: ', NULL) => 'My feature id is: '
```

mai multe detalii: ||, +

format

Format a string using supplied arguments.

Sintaxă	format(string, arg1, arg2, …)
Argumente	• **string** - A string with placeholders %1, %2, etc., for the arguments. Placeholders can be repeated. The lowest numbered placeholder is replaced by arg1, the next by arg2, etc. • **arg** - any type. Any number of arguments.
Exemple	• `format('This %1 a %2','is', 'test')` → «This is a test» • `format('This is %2','a bit unexpected but 2 is lowest number in string','normal')` → «This is a bit unexpected but 2 is lowest number in string»

format_date

Formats a date type or string into a custom string format. Uses Qt date/time format strings. See QDateTime::toString.

Sintaxă	format_date(datetime, format, [language]) [] marks optional arguments
Argumente	• **datetime** - date, time or datetime value • **format** - String template used to format the string.

Expresie	Rezultat
d	ziua, un număr fără zero la început (de la 1 la 31)
dd	ziua, un număr cu zero la început (de la 01 la 31)
ddd	the abbreviated localized day name (e.g. «Mon» to «Sun»)
dddd	the long localized day name (e.g. «Monday» to «Sunday»)
M	the month as number without a leading zero (1-12)
MM	the month as number with a leading zero (01-12)
MMM	the abbreviated localized month name (e.g. «Jan» to «Dec»)
MMMM	the long localized month name (e.g. «January» to «December»)
yy	the year as two digit number (00-99)
yyyy	the year as four digit number

These expressions may be used for the time part of the format string:

Expresie	Rezultat
h	the hour without a leading zero (0 to 23 or 1 to 12 if AM/PM display)
hh	the hour with a leading zero (00 to 23 or 01 to 12 if AM/PM display)
H	the hour without a leading zero (0 to 23, even with AM/PM display)
HH	the hour with a leading zero (00 to 23, even with AM/PM display)
m	minutul, fără zero la început (de la 0 la 59)
mm	minutul, cu zero la început (de la 00 la 59)
s	secunda, fără zero la început (de la 0 la 59)
ss	secunda, cu un zero la început (de la 00 la 59)
z	the milliseconds without trailing zeroes (0 to 999)
zzz	the milliseconds with trailing zeroes (000 to 999)
AP or A	interpret as an AM/PM time. *AP* must be either «AM» or «PM».
ap or a	Interpret as an AM/PM time. *ap* must be either «am» or «pm».

- **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to format the date into a custom string. By default the current QGIS user locale is used.

Exemple

- `format_date('2012-05-15','dd.MM.yyyy')` → «15.05.2012»
- `format_date('2012-05-15','d MMMM yyyy','fr')` → «15 mai 2012»
- `format_date('2012-05-15','dddd')` → «Tuesday», if the current locale is an English variant
- `format_date('2012-05-15 13:54:20','dd.MM.yy')` → «15.05.12»
- `format_date('13:54:20','hh:mm AP')` → «01:54 PM»

format_number

Returns a number formatted with the locale separator for thousands. By default the current QGIS user locale is used. Also truncates the decimal places to the number of supplied places.

Sintaxă	format_number(number, [places=0], [language], [omit_group_separators=false], [trim_trailing_zeroes=false]) [] marks optional arguments
Argumente	• **number** - number to be formatted • **places** - integer representing the number of decimal places to truncate the string to. • **language** - language (lowercase, two- or three-letter, ISO 639 language code) used to format the number into a string. By default the current QGIS user locale is used. • **omit_group_separators** - if set to true then group separators will not be included in the string • **trim_trailing_zeroes** - if set to true then trailing zeros following the decimal point will be trimmed from the string
Exemple	• `format_number(10000000.332,2)` → «10,000,000.33» if e.g. the current locale is an English variant • `format_number(10000000.332,2,'fr')` → «10 000 000,33»

left

Returns a substring that contains the *n* leftmost characters of the string.

Sintaxă	left(string, length)
Argumente	• **string** - a string • **length** - integer. The number of characters from the left of the string to return.
Exemple	• `left('Hello World',5)` → «Hello»

length

Returns the number of characters in a string or the length of a geometry linestring.

String variant

Returns the number of characters in a string.

Sintaxă	length(string)
Argumente	• **string** - string to count length of
Exemple	• `length('hello')` → 5

Geometry variant

Calculate the length of a geometry line object. Calculations are always planimetric in the Spatial Reference System (SRS) of this geometry, and the units of the returned length will match the units for the SRS. This differs from the

calculations performed by the $length function, which will perform ellipsoidal calculations based on the project's ellipsoid and distance unit settings.

Sintaxă	length(geometry)
Argumente	• **geometry** - line geometry object
Exemple	• `length(geom_from_wkt('LINESTRING(0 0, 4 0)'))` → 4.0

lower

Converts a string to lower case letters.

Sintaxă	lower(string)
Argumente	• **string** - the string to convert to lower case
Exemple	• `lower('HELLO World')` → «hello world»

lpad

Returns a string padded on the left to the specified width, using a fill character. If the target width is smaller than the string's length, the string is truncated.

Sintaxă	lpad(string, width, fill)
Argumente	• **string** - string to pad • **width** - length of new string • **fill** - character to pad the remaining space with
Exemple	• `lpad('Hello', 10, 'x')` → «xxxxxHello» • `lpad('Hello', 3, 'x')` → «Hel»

ltrim

Removes the longest string containing only the specified characters (a space by default) from the start of string.

Sintaxă	ltrim(string, [characters=» «]) [] marks optional arguments
Argumente	• **string** - string to trim • **characters** - characters to trim
Exemple	• `ltrim(' hello world ')` → «hello world « • `ltrim('zzzytest', 'xyz')` → «test»

Further reading: *rtrim*, *trim*

regexp_match

Return the first matching position matching a regular expression within an unicode string, or 0 if the substring is not found.

Sintaxă	regexp_match(input_string, regex)
Argumente	• **input_string** - the string to test against the regular expression • **regex** - The regular expression to test against. Backslash characters must be double escaped (e.g., „\\s” to match a white space character or „\\b” to match a word boundary).
Exemple	• `regexp_match('QGIS ROCKS','\\sROCKS')` → 5 • `regexp_match('Budač','udač\\b')` → 2

regexp_replace

Returns a string with the supplied regular expression replaced.

Sintaxă	regexp_replace(input_string, regex, replacement)
Argumente	• **input_string** - the string to replace matches in • **regex** - The regular expression to replace. Backslash characters must be double escaped (e.g., „\\s” to match a white space character). • **replacement** - The string that will replace any matching occurrences of the supplied regular expression. Captured groups can be inserted into the replacement string using \\1, \\2, etc.
Exemple	• `regexp_replace('QGIS SHOULD ROCK','\\sSHOULD\\s',' DOES ')` → «QGIS DOES ROCK» • `regexp_replace('ABC123','\\d+','')` → «ABC» • `regexp_replace('my name is John','(.*) is (.*)','\\2 is \\1')` → «John is my name»

regexp_substr

Returns the portion of a string which matches a supplied regular expression.

Sintaxă	regexp_substr(input_string, regex)
Argumente	• **input_string** - the string to find matches in • **regex** - The regular expression to match against. Backslash characters must be double escaped (e.g., „\\s” to match a white space character).
Exemple	• `regexp_substr('abc123','(\\d+)')` → «123»

replace

Returns a string with the supplied string, array, or map of strings replaced.

String & array variant

Returns a string with the supplied string or array of strings replaced by a string or an array of strings.

Sintaxă	replace(string, before, after)
Argumente	• **string** - șirul de intrare • **before** - the string or array of strings to replace • **after** - the string or array of strings to use as a replacement
Exemple	• `replace('QGIS SHOULD ROCK','SHOULD','DOES')` → «QGIS DOES ROCK» • `replace('QGIS ABC',array('A','B','C'),array('X','Y','Z'))` → «QGIS XYZ» • `replace('QGIS',array('Q','S'),'')` → «GI»

Map variant

Returns a string with the supplied map keys replaced by paired values. Longer map keys are evaluated first.

Sintaxă	replace(string, map)
Argumente	• **string** - șirul de intrare • **map** - the map containing keys and values
Exemple	• `replace('APP SHOULD ROCK',map('APP','QGIS','SHOULD','DOES'))` → «QGIS DOES ROCK» • `replace('forty two',map('for','4','two','2','forty two','42'))` → «42»

right

Returns a substring that contains the *n* rightmost characters of the string.

Sintaxă	right(string, length)
Argumente	• **string** - a string • **length** - integer. The number of characters from the right of the string to return.
Exemple	• `right('Hello World',5)` → «World»

rpad

Returns a string padded on the right to the specified width, using a fill character. If the target width is smaller than the string's length, the string is truncated.

Sintaxă	rpad(string, width, fill)
Argumente	• **string** - string to pad • **width** - length of new string • **fill** - character to pad the remaining space with
Exemple	• `rpad('Hello', 10, 'x')` → «Helloxxxxx» • `rpad('Hello', 3, 'x')` → «Hel»

rtrim

Removes the longest string containing only the specified characters (a space by default) from the end of string.

Sintaxă	rtrim(string, [characters=» «]) [] marks optional arguments
Argumente	• **string** - string to trim • **characters** - characters to trim
Exemple	• `rtrim(' hello world ')` → « hello world» • `rtrim('testxxzx', 'xyz')` → «test»

Further reading: *ltrim*, *trim*

strpos

Return the first matching position of a substring within another string, or 0 if the substring is not found.

Sintaxă	strpos(haystack, needle)
Argumente	• **haystack** - string that is to be searched • **needle** - string to search for
Exemple	• `strpos('HELLO WORLD','WORLD')` → 7 • `strpos('HELLO WORLD','GOODBYE')` → 0

substr

Returns a part of a string.

Sintaxă	substr(string, start, [length]) [] marks optional arguments
Argumente	• **string** - the full input string • **start** - integer representing start position to extract beginning with 1; if start is negative, the return string will begin at the end of the string minus the start value • **length** - integer representing length of string to extract; if length is negative, the return string will omit the given length of characters from the end of the string
Exemple	• `substr('HELLO WORLD',3,5)` → «LLO W» • `substr('HELLO WORLD',6)` → « WORLD» • `substr('HELLO WORLD',-5)` → «WORLD» • `substr('HELLO',3,-1)` → «LL» • `substr('HELLO WORLD',-5,2)` → «WO» • `substr('HELLO WORLD',-5,-1)` → «WORL»

title

Converts all words of a string to title case (all words lower case with leading capital letter).

Sintaxă	title(string)
Argumente	• **string** - the string to convert to title case
Exemple	• `title('hello WOrld')` → «Hello World»

to_string

Convertește un număr într-un șir.

Sintaxă	to_string(number)
Argumente	• **number** - Număr întreg sau valoare reală. Numărul de convertit în șir.
Exemple	• `to_string(123)` → «123»

trim

Removes all leading and trailing whitespace (spaces, tabs, etc) from a string.

Sintaxă	trim(string)
Argumente	• **string** - string to trim
Exemple	• `trim(' hello world ')` → «hello world»

Further reading: *ltrim*, *rtrim*

upper

Converts a string to upper case letters.

Sintaxă	upper(string)
Argumente	• **string** - the string to convert to upper case
Exemple	• `upper('hello WOrld')` → «HELLO WORLD»

wordwrap

Returns a string wrapped to a maximum/minimum number of characters.

Sintaxă	wordwrap(string, wrap_length, [delimiter_string]) [] marks optional arguments
Argumente	• **string** - the string to be wrapped • **wrap_length** - an integer. If wrap_length is positive the number represents the ideal maximum number of characters to wrap; if negative, the number represents the minimum number of characters to wrap. • **delimiter_string** - Optional delimiter string to wrap to a new line.
Exemple	• `wordwrap('UNIVERSITY OF QGIS',13)` → «UNIVERSITY OF QGIS» • `wordwrap('UNIVERSITY OF QGIS',-3)` → «UNIVERSITY OF QGIS»

13.2.26 User Expressions

This group contains the expressions saved as *user expressions*.

13.2.27 Variabile

This group contains dynamic variables related to the application, the project file and other settings. The availability of variables depends on the context:

- din dialogul Select by expression
- din dialogul Field calculator
- din dialogul cu proprietățile stratului
- from the print layout

To use these variables in an expression, they should be preceded by the `@` character (e.g, `@row_number`).

Variable	Descriere
algorithm_id	The unique ID of an algorithm
animation_end_time	End of the animation's overall temporal time range (as a datetime value)
animation_interval	Duration of the animation's overall temporal time range (as an interval value)
animation_start_time	Start of the animation's overall temporal time range (as a datetime value)
atlas_feature	The current atlas feature (as feature object)
atlas_featureid	The current atlas feature ID
atlas_featurenumber	The current atlas feature number in the layout
atlas_filename	The current atlas file name
atlas_geometry	The current atlas feature geometry
atlas_layerid	The current atlas coverage layer ID
atlas_layername	The current atlas coverage layer name
atlas_pagename	The current atlas page name
atlas_totalfeatures	The total number of features in atlas
canvas_cursor_point	The last cursor position on the canvas in the project's geographical coordinates
cluster_color	The color of symbols within a cluster, or NULL if symbols have mixed colors
cluster_size	The number of symbols contained within a cluster
current_feature	The feature currently being edited in the attribute form or table row
current_geometry	The geometry of the feature currently being edited in the form or the table row
current_parent_feature	represents the feature currently being edited in the parent form. Only usable in an embedded form context.
current_parent_geometry	represents the geometry of the feature currently being edited in the parent form. Only usable in an embedded form context.
form_mode	What the form is used for, like AddFeatureMode, SingleEditMode, MultiEditMode, SearchMode, AggregateSearchMode or IdentifyMode as string.
feature	The current feature being evaluated. This can be used with the «attribute» function to evaluate attribute values from the current feature.
frame_duration	Temporal duration of each animation frame (as an interval value)
frame_number	Current frame number during animation playback
frame_rate	Number of frames per second during animation playback
fullextent_maxx	Maximum x value from full canvas extent (including all layers)
fullextent_maxy	Maximum y value from full canvas extent (including all layers)
fullextent_minx	Minimum x value from full canvas extent (including all layers)
fullextent_miny	Minimum y value from full canvas extent (including all layers)
geometry	The geometry of the current feature being evaluated
geometry_part_count	The number of parts in rendered feature's geometry
geometry_part_num	The current geometry part number for feature being rendered
geometry_point_count	The number of points in the rendered geometry's part
geometry_point_num	The current point number in the rendered geometry's part
geometry_ring_num	Current geometry ring number for feature being rendered (for polygon features only). The exterior ring has a value of 0.
grid_axis	The current grid annotation axis (eg, «x» for longitude, «y» for latitude)

continues on next page

Tabelul 13.1 – continuare din pagina precedentă

Variable	Descriere
grid_number	The current grid annotation value
id	The ID of the current feature being evaluated
item_id	The layout item user ID (not necessarily unique)
item_uuid	The layout item unique ID
layer	The current layer
layer_crs	The Coordinate Reference System Authority ID of the current layer
layer_crs_ellipsoid	The ellipsoid Authority ID of the current layer CRS
layer_cursor_point	Point geometry under the mouse position in map canvas, in active layer's CRS
layer_id	The ID of current layer
layer_ids	The IDs of all the map layers in the current project as a list
layer_name	The name of current layer
layers	All the map layers in the current project as a list
layout_dpi	The composition resolution (DPI)
layout_name	The layout name
layout_numpages	The number of pages in the layout
layout_page	The page number of the current item in the layout
layout_pageheight	The active page height in the layout (in mm for standard paper sizes, or whatever unit was used for custom paper size)
layout_pageoffsets	Array of Y coordinate of the top of each page. Allows to dynamically position items on pages in a context where page sizes may change
layout_pagewidth	The active page width in the layout (in mm for standard paper sizes, or whatever unit was used for custom paper size)
legend_column_count	The number of columns in the legend
legend_filter_by_map	Indicates if the content of the legend is filtered by the map
legend_filter_out_atlas	Indicates if the atlas is filtered out of the legend
legend_split_layers	Indicates if layers can be split in the legend
legend_title	The title of the legend
legend_wrap_string	The character(s) used to wrap the legend text
map_crs	The Coordinate reference system of the current map
map_crs_acronym	The acronym of the Coordinate reference system of the current map
map_crs_definition	The full definition of the Coordinate reference system of the current map
map_crs_description	The name of the Coordinate reference system of the current map
map_crs_ellipsoid	The acronym of the ellipsoid of the Coordinate reference system of the current map
map_crs_proj4	The Proj4 definition of the Coordinate reference system of the current map
map_crs_projection	The descriptive name of the projection method used by the Coordinate reference system of the map (e.g. «Albers Equal Area»)
map_crs_wkt	The WKT definition of the Coordinate reference system of the current map
map_end_time	The end of the map's temporal time range (as a datetime value)
map_extent	The geometry representing the current extent of the map
map_extent_center	The point feature at the center of the map
map_extent_height	The current height of the map
map_extent_width	The current width of the map
map_id	The ID of current map destination. This will be «canvas» for canvas renders, and the item ID for layout map renders
map_interval	The duration of the map's temporal time range (as an interval value)
map_layer_ids	The list of map layer IDs visible in the map
map_layers	The list of map layers visible in the map
map_rotation	The current rotation of the map
map_scale	The current scale of the map
map_start_time	The start of the map's temporal time range (as a datetime value)
map_units	The units of map measurements
model_path	Full path (including file name) of current model (or project path if model is embedded in a project).
model_folder	Folder containing current model (or project folder if model is embedded in a project).

continues on next page

Tabelul 13.1 – continuare din pagina precedentă

Variable	Descriere
model_name	Name of current model
model_group	Group for current model
notification_message	Content of the notification message sent by the provider (available only for actions triggered by provider notifications).
parent	Refers to the current feature in the parent layer, providing access to its attributes and geometry when filtering an *aggregate* function
project_abstract	The project abstract, taken from project metadata
project_area_units	The area unit for the current project, used when calculating areas of geometries
project_author	The project author, taken from project metadata
project_basename	The basename of current project's filename (without path and extension)
project_creation_date	The project creation date, taken from project metadata
project_crs	The Coordinate reference system of the project
project_crs_arconym	The acronym of the Coordinate reference system of the project
project_crs_definition	The full definition of the Coordinate reference system of the project
project_crs_description	The description of the Coordinate reference system of the project
project_crs_ellipsoid	The ellipsoid of the Coordinate reference system of the project
project_crs_proj4	The Proj4 representation of the Coordinate reference system of the project
project_crs_wkt	The WKT (well known text) representation of the coordinate reference system of the project
project_distance_units	The distance unit for the current project, used when calculating lengths of geometries and distances
project_ellipsoid	The name of the ellipsoid of the current project, used when calculating geodetic areas or lengths of geometries
project_filename	The filename of the current project
project_folder	The folder of the current project
project_home	The home path of the current project
project_identifier	The project identifier, taken from the project's metadata
project_keywords	The project keywords, taken from the project's metadata
project_last_saved	Date/time when project was last saved.
project_path	The full path (including file name) of the current project
project_title	The title of current project
project_units	The units of the project's CRS
qgis_locale	The current language of QGIS
qgis_os_name	The current Operating system name, eg «windows», «linux» or «osx»
qgis_platform	The QGIS platform, eg «desktop» or «server»
qgis_release_name	The current QGIS release name
qgis_short_version	The current QGIS version short string
qgis_version	The current QGIS version string
qgis_version_no	The current QGIS version number
row_number	Stochează numărul rândului curent
snapping_results	Gives access to snapping results while digitizing a feature (only available in add feature)
scale_value	The current scale bar distance value
selected_file_path	Selected file path from file widget selector when uploading a file with an external storage system
symbol_angle	The angle of the symbol used to render the feature (valid for marker symbols only)
symbol_color	The color of the symbol used to render the feature
symbol_count	The number of features represented by the symbol (in the layout legend)
symbol_frame	The frame number (for animated symbols only)
symbol_id	The Internal ID of the symbol (in the layout legend)
symbol_label	The label for the symbol (either a user defined label or the default autogenerated label - in the layout legend)
symbol_layer_count	Total number of symbol layers in the symbol
symbol_layer_index	Current symbol layer index
symbol_marker_column	Column number for marker (valid for point pattern fills only).

continues on next page

Tabelul 13.1 – continuare din pagina precedentă

Variable	Descriere
symbol_marker_row	Row number for marker (valid for point pattern fills only).
user_account_name	The current user's operating system account name
user_full_name	The current user's operating system user name
value	The current value
vector_tile_zoom	Exact vector tile zoom level of the map that is being rendered (derived from the current map scale). Normally in interval [0, 20]. Unlike @zoom_level, this variable is a floating point value which can be used to interpolate values between two integer zoom levels.
with_variable	Allows setting a variable for usage within an expression and avoid recalculating the same value repeatedly
zoom_level	Vector tile zoom level of the map that is being rendered (derived from the current map scale). Normally in interval [0, 20].

Câteva exemple:

- Return the X coordinate of a map item center in layout:

```
x( map_get( item_variables( 'map1'), 'map_extent_center' ) )
```

- Return, for each feature in the current layer, the number of overlapping airport features:

```
aggregate( layer:='airport', aggregate:='count', expression:="code",
               filter:=intersects( $geometry, geometry( @parent ) ) )
```

- Get the object_id of the first snapped point of a line:

```
with_variable(
  'first_snapped_point',
  array_first( @snapping_results ),
  attribute(
    get_feature_by_id(
      map_get( @first_snapped_point, 'layer' ),
      map_get( @first_snapped_point, 'feature_id' )
    ),
    'object_id'
  )
)
```

13.2.28 Funcții Recente

This group contains recently used functions. Depending on the context of its usage (feature selection, field calculator, generic), recently applied expressions are added to the corresponding list (up to ten expressions), sorted from more to less recent. This makes it easy to quickly retrieve and reapply previously used expressions.

CAPITOLUL 14

Biblioteca de Stiluri

14.1 The Style Manager

14.1.1 The Style Manager dialog

The *Style Manager* is the place where you can manage and create generic style items. These are symbols, color ramps, text formats or label settings that can be used to symbolize features, layers or print layouts. They are stored in the `symbology-style.db` database under the active *user profile* and shared with all the project files opened with that profile. Style items can also be shared with others thanks to the export/import capabilities of the *Style Manager* dialog.

You can open that modeless dialog either:

- from the *Settings* ► *Style Manager…* menu
- with the Style Manager button from the Project toolbar
- or with the Style Manager button from a vector *Layer Properties* ► menu (while *configuring a symbol* or *formatting a text*).

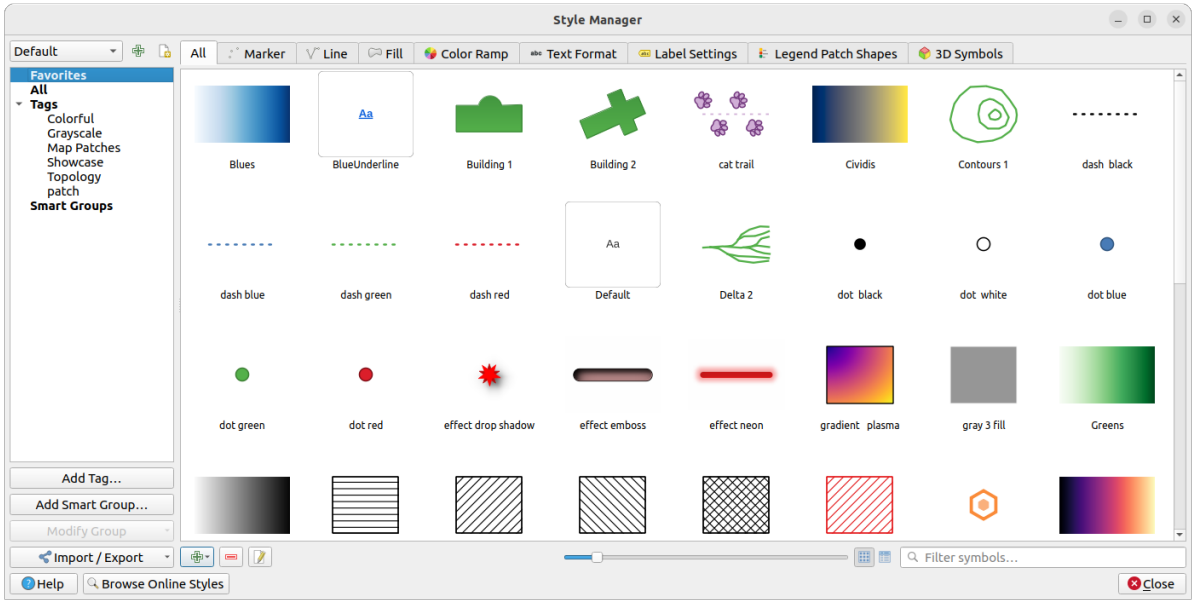


Fig. 14.1: The Style Manager

Organizing style items

The *Style Manager* dialog displays in its center a frame with previewed items organized into tabs:

- *All* for a complete collection of point, linear and surface symbols and label settings as well as predefined color ramps and text formats;
- *Marker* for point symbols only;
- *Line* for linear symbols only;
- *Fill* for surface symbols only;
- *Color ramp*;
- abc *Text format* to manage *text formats*, which store the font, color, buffers, shadows, and backgrounds of texts (i.e. all the formatting parts of the label settings, which for instance can be used in layouts);
- *Label settings* to manage *label settings*, which include the text formats and some layer-type specific settings such as label placement, priority, callouts, rendering…
- *Legend Patch Shapes* to manage custom legend patch shapes, which include *Marker*, *Line* and *Fill* geometries.
- *3D Symbols* to configure symbols with *3D properties* (extrusion, shading, altitude, …) for the features to render in a *3D Map view*

You can arrange the Styles in *Icon View* or in *List View* on the bottom right side. In both views the tooltip shows a larger instance of the style. The thumbnail size slider at the left of the icons helps you adjust the actual thumbnail sizes in the dialog, for a better preview of the symbols!

For each family of items, you can organize the elements into different categories, listed in the panel on the left:

- **Favorites**: displayed by default when configuring an item, it shows an extensible set of items;
- **All**: lists all the available items for the active type;

- **Tags**: shows a list of labels you can use to identify the items. An item can be tagged more than once. Select a tag in the list and the tabs are updated to show only their items that belong to it. To create a new tag you could later attach to a set of items, use the *Add Tag…* button or select the *Add Tag…* from any tag contextual menu;
- **Smart Group**: a smart group dynamically fetches its symbols according to conditions set (see eg, Fig. 14.2). Click the *Add Smart Group…* button to create smart groups. The dialog box allows you to enter an expression to filter the items to select (has a particular tag, have a string in its name, etc.). Any symbol, color ramp, text format or label setting that satisfies the entered condition(s) is automatically added to the smart group.

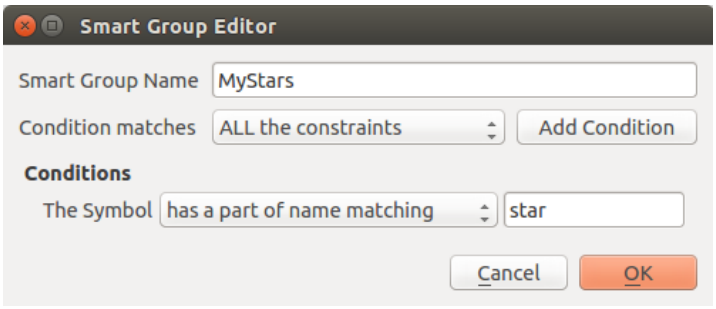


Fig. 14.2: Creating a Smart Group

Tags and smart groups are not mutually exclusive: they are simply two different ways to organize your style elements. Unlike the smart groups that automatically fetch their belonged items based on the input constraints, tags are filled by the user. To edit any of those categories, you can either:

- select the items, right-click and choose *Add to Tag* ► and then select the tag name or create a new tag;
- select the tag and press *Modify group…* ► *Attach Selected Tag to Symbols*. A checkbox appears next to each item to help you select or deselect it. When selection is finished, press *Modify group…* ► *Finish Tagging*.
- select the smart group, press *Modify group…* ► *Edit smart group…* and configure a new set of constraints in the *Smart Group Editor* dialog. This option is also available in the contextual menu of the smart group.

To remove a tag or a smart group, right-click on it and select the *Remove* button. Note that this does not delete the items grouped in the category.

Adding, editing or removing an item

As seen earlier, style elements are listed under different tabs whose contents depend on the active category (tag, smart group, favorites…). When a tab is enabled, you can:

- Add new items: press the Add item button and configure the item following *symbols*, *color ramps* or *text format and label* builder description.
- Modify an existing item: select an item and press Edit item button and configure as mentioned above.
- Delete existing items: to delete an element you no longer need, select it and click Remove item (also available through right-click). The item will be deleted from the local database.

Note that the *All* tab provides access to these options for every type of item.

Right-clicking over a selection of items also allows you to:

- *Add to Favorites*;
- *Remove from Favorites*;
- *Add to Tag* ► and select the appropriate tag or create a new one to use; the currently assigned tags are checked;
- *Clear Tags*: detaching the symbols from any tag;

- *Remove Item(s)*;
- *Edit Item*: applies to the item you right-click over;
- *Copy Item*;
- *Paste Item* …: pasting to one of the categories of the style manager or elsewhere in QGIS (symbol or color buttons)
- *Export Selected Symbol(s) as PNG…* (only available with symbols);
- *Export Selected Symbol(s) as SVG…* (only available with symbols);

Sharing style items

The *Import/Export* tool, at the left bottom of the Style Manager dialog, offers options to easily share symbols, color ramps, text formats and label settings with others. These options are also available through right-click over the items.

Exporting items

You can export a set of items to an `.XML` file:

1. Expand the *Import/Export* drop-down menu and select *Export Item(s)…*
2. Choose the items you'd like to integrate. Selection can be done with the mouse or using a tag or a group previously set.
3. Press *Export* when ready. You'll be prompted to indicate the destination of the saved file. The XML format generates a single file containing all the selected items. This file can then be imported in another user's style library.

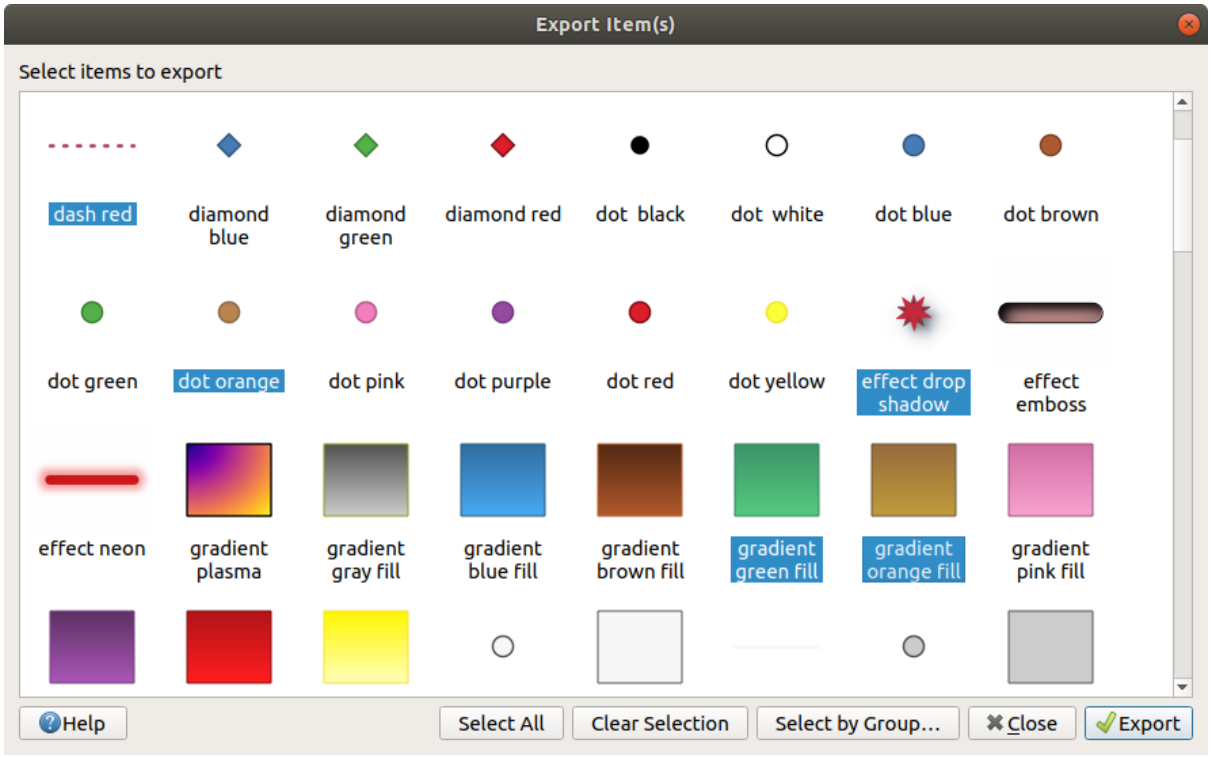


Fig. 14.3: Exporting style items

When symbols are selected, you can also export them to `.PNG` or `.SVG`. Exporting to `.PNG` or `.SVG` (both not available for other style item types) creates a file for each selected symbol in a given folder. The SVG folder can be added to the *SVG paths* in *Settings* ► *Options* ► *System* menu of another user, allowing him direct access to all these symbols.

Importing items

You can extend your style library by importing new items:

1. Expand the *Import/Export* drop-down menu and select *Import Item(s)* at the left bottom of the dialog.
2. In the new dialog, indicate the source of the style items (it can be an `.xml` file on the disk or a url).
3. Set whether to *Add to favorites* the items to import.
4. Check *Do not import embedded tags* to avoid the import of tags associated to the items being imported.
5. Give the name of any *Additional tag(s)* to apply to the new items.
6. Select from the preview the symbols you want to add to your library.
7. And press *Import*.

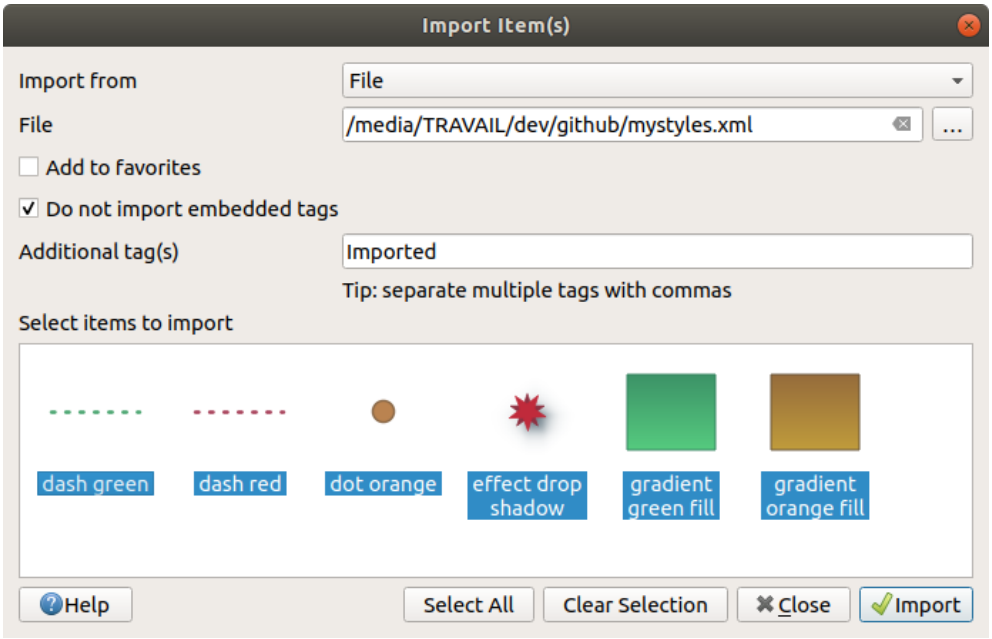


Fig. 14.4: Importing style items

Using the Browser panel

It's also possible to import style items into the active user profile style database directly from the *Browser* panel:

1. Select the style `.xml` file in the browser
2. Drag-and-drop it over the map canvas or right-click and select *Import Style…*
3. Fill the *Import Items* dialog following *Importing items*
4. Press *Import* and the selected style items are added to the style database

Double-clicking the style file in the browser opens the *Style Manager* dialog showing the items in the file. You can select them and press *Copy to Default Style…* to import them into the active style database. Tags can be assigned to items. Also available through right-click, *Open Style…* command.

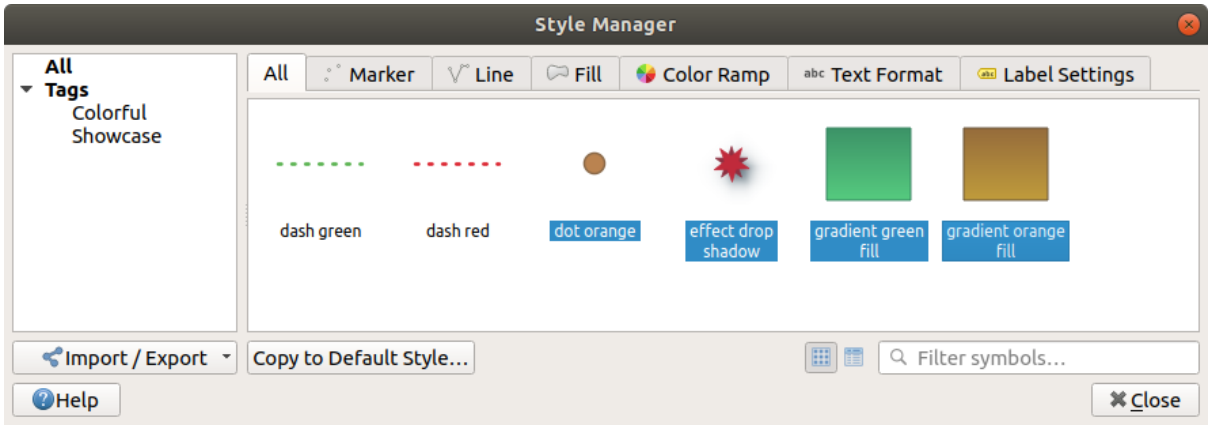


Fig. 14.5: Opening a style items file

The dialog also allows to export single symbols as `.PNG` or `.SVG` files.

Using the online repository

The QGIS project maintains a repository with a collection of styles shared by QGIS users. This is available at https://plugins.qgis.org/styles and can be accessed from the *Style Manager* dialog, pressing the *Browse Online Styles* button at the bottom.

From that repository, you can:

1. Browse and search for any style items, based on their type or name
2. Download the style file and unzip it
3. Load the `.xml` based file into your style database in QGIS, using any of the aforementioned import methods.

14.1.2 Setting a Color Ramp

The Color ramp tab in the *Style Manager* dialog helps you preview different color ramps based on the category selected in the left panel.

To create a custom color ramp, activate the Color ramp tab and click the Add item button. The button reveals a drop-down list to choose the ramp type:

- *Gradient*: given a start and end colors, generates a color ramp which can be **continuous** or **discrete**. With double-clicking the ramp preview, you can add as many intermediate color stops as you want. Click on the color stop indicator and under *Gradient stop* you can:
  - adjust its *Relative position* from the color ramp start. Also possible dragging the indicator with the mouse, or pressing the arrow keys (combine with `Shift` key for a larger move)
  - specify the color model to use when interpolating between colors: it can be *RGB*, *HSL* or *HSV*. In some circumstances, this option can help avoid desaturated mid tones, resulting in more visually pleasing gradients.
  - set the direction which the interpolation should follow for the **Hue** component of a *HSL* or *HSV* color specification. It can be *Clockwise* or *Counterclockwise*.
  - set the *color properties*

– remove the color stop pressing *Delete stop* or `DEL`

The *Plots* group provides another graphical way to design the color ramp, changing the position or the opacity and HSL components of the color stops.

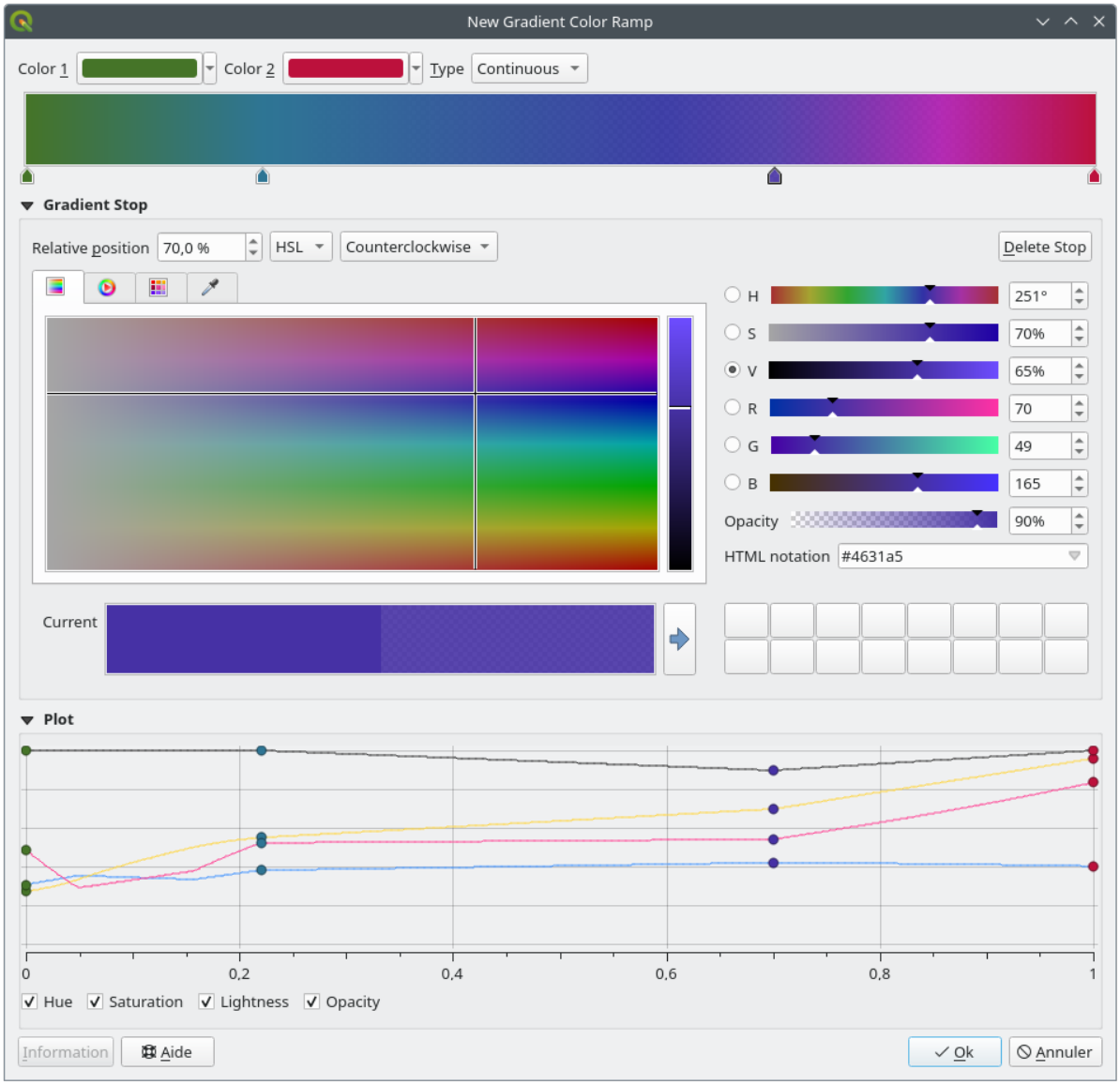


Fig. 14.6: Example of custom gradient color ramp with multiple stops

Sugestie: Drag-and-drop a color from a color spot onto the gradient ramp preview adds a new color stop.

- *Color presets*: allows to create a color ramp consisting of a list of colors selected by the user;
- *Random*: creates a random set of colors based on range of values for *Hue*, *Saturation*, *Value* and *Opacity* and a number of colors (*Classes*);
- *Catalog: ColorBrewer*: a set of predefined discrete color gradients you can customize the number of colors in the ramp;
- or *Catalog: cpt-city*: an access to a whole catalog of color gradients to locally *save as standard gradient*. The cpt-city option opens a new dialog with hundreds of themes included «out of the box».

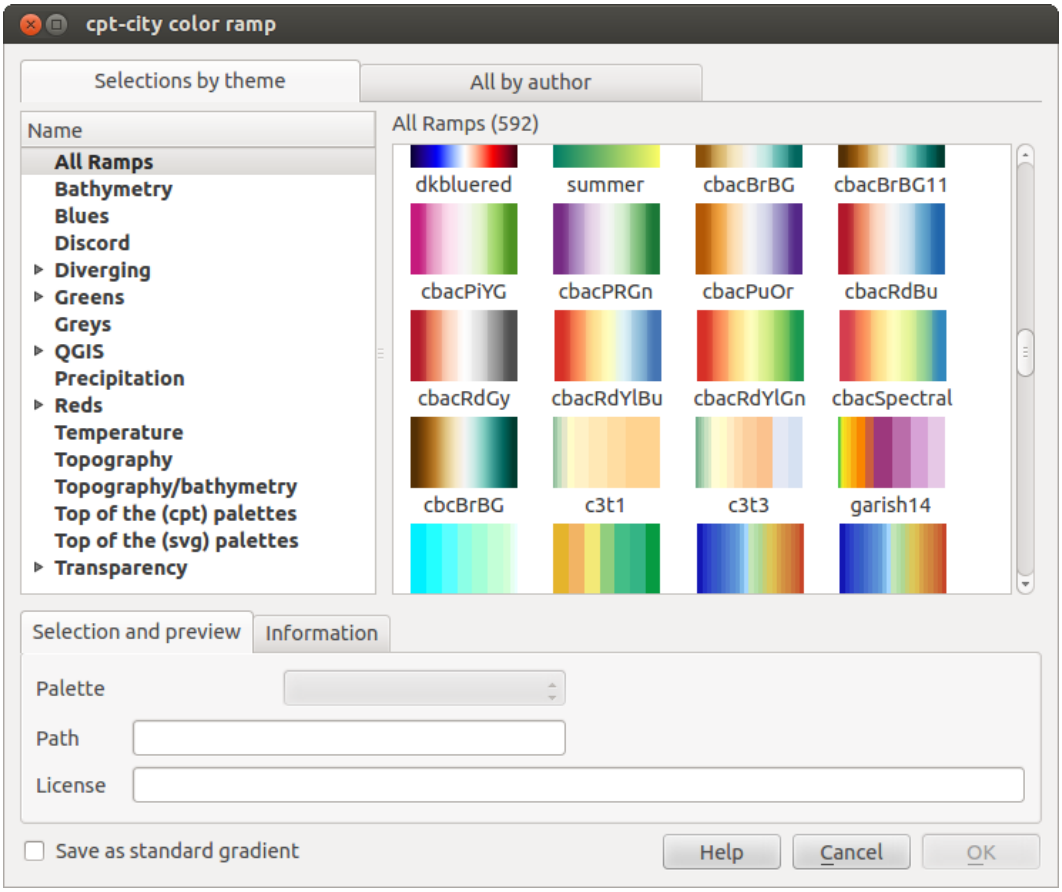


Fig. 14.7: cpt-city dialog with hundreds of color ramps

14.1.3 Creating a Legend Patch Shape

To create a new Legend Patch Shape, activate the *Legend Patch Shapes* tab and click the Add item button. The button reveals a drop-down list to choose the geometry type:

- *Marker Legend Patch Shape…*: to use with point geometries.
- *Line Legend Patch Shape…*: to use with line geometries.
- *Fill Legend Patch Shape…*: to use with polygon geometries.

All three options will show the same dialog.

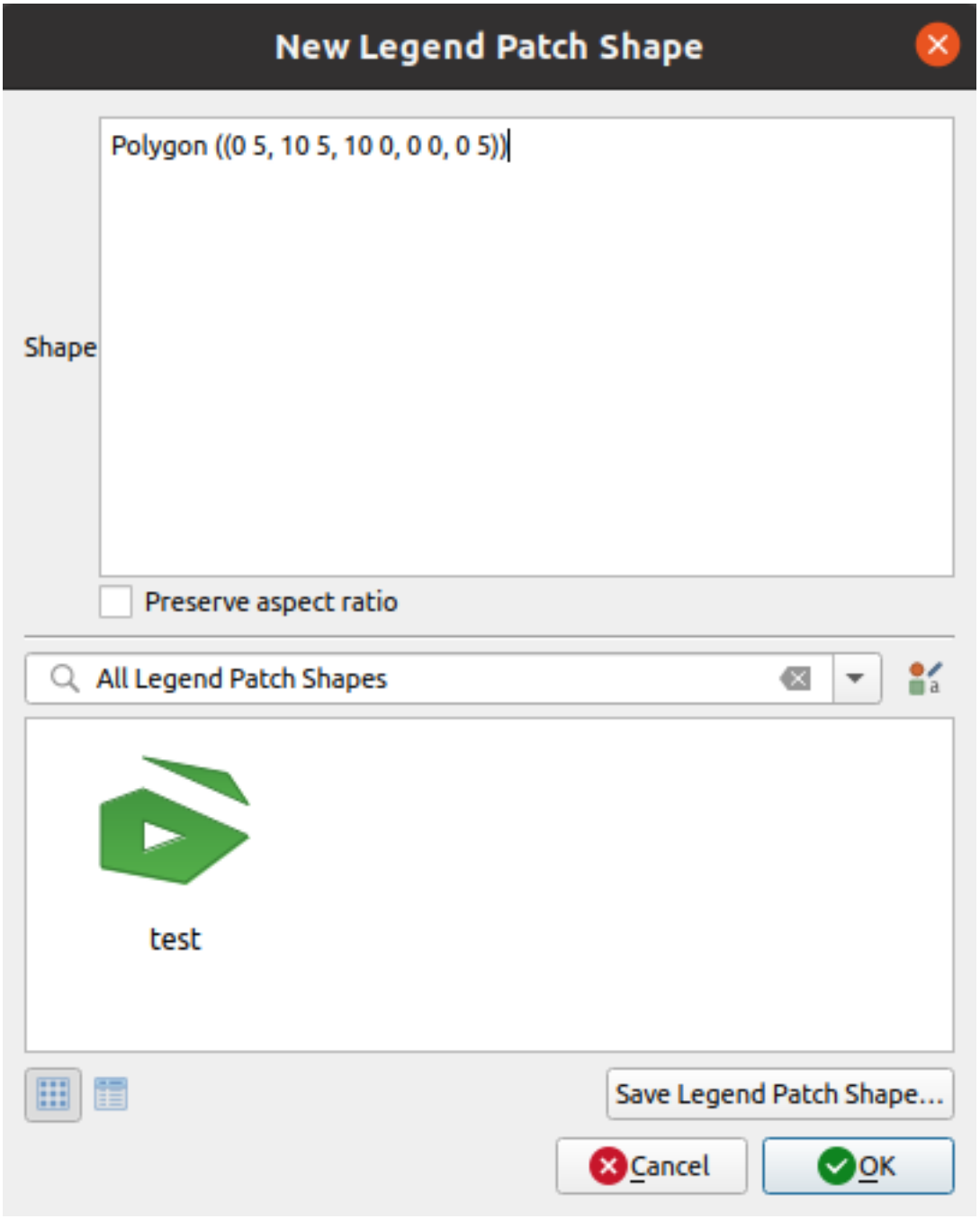


Fig. 14.8: Create a new Legend Patch Shape

Only the shape type and displayed legend patch shapes will differ regarding to the chosen geometry type. The following options will be available:

- *Shape*: define the shape of the legend patch shape as a WKT string. Single and multipart geometries may be used, but no GeometryCollection.
- *Preserve aspect ratio*
- *Icon View* or *List View* of available legend patch shapes, filtered by tags.

When the new Shape is defined you can *Save Legend Patch Shape…* or press *OK*, which will both lead to the same dialog.

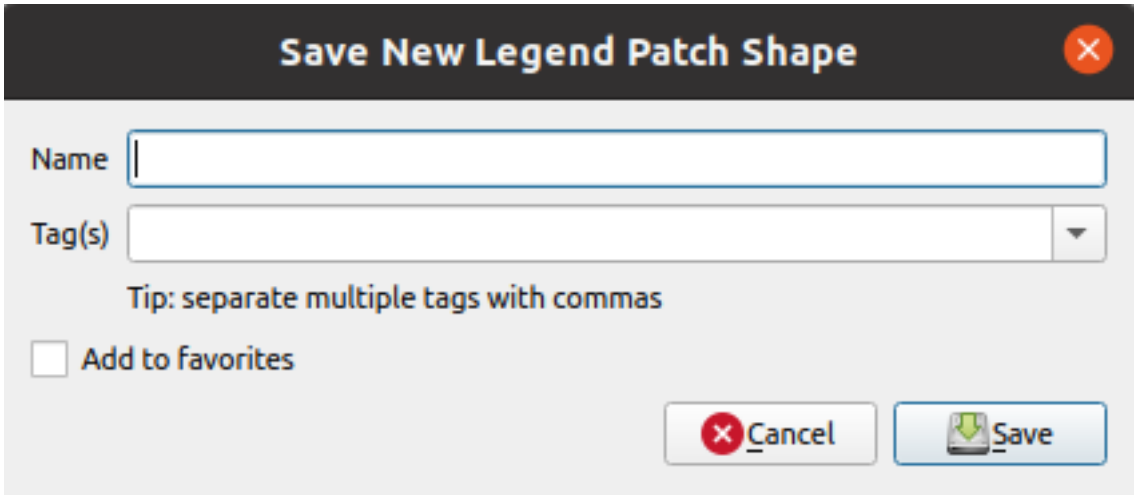


Fig. 14.9: Save a new Legend Patch Shape

Here you have to choose a name, tags to describe the shape and if it should be added to favorites.

If you press *Save…*, the shape is added to the list and you are directed back to the *New Legend Patch Shape* dialog to keep creating new shapes.

14.2 The Symbol Selector

The Symbol selector is the main dialog to design a symbol. You can create or edit Marker, Line or Fill Symbols.

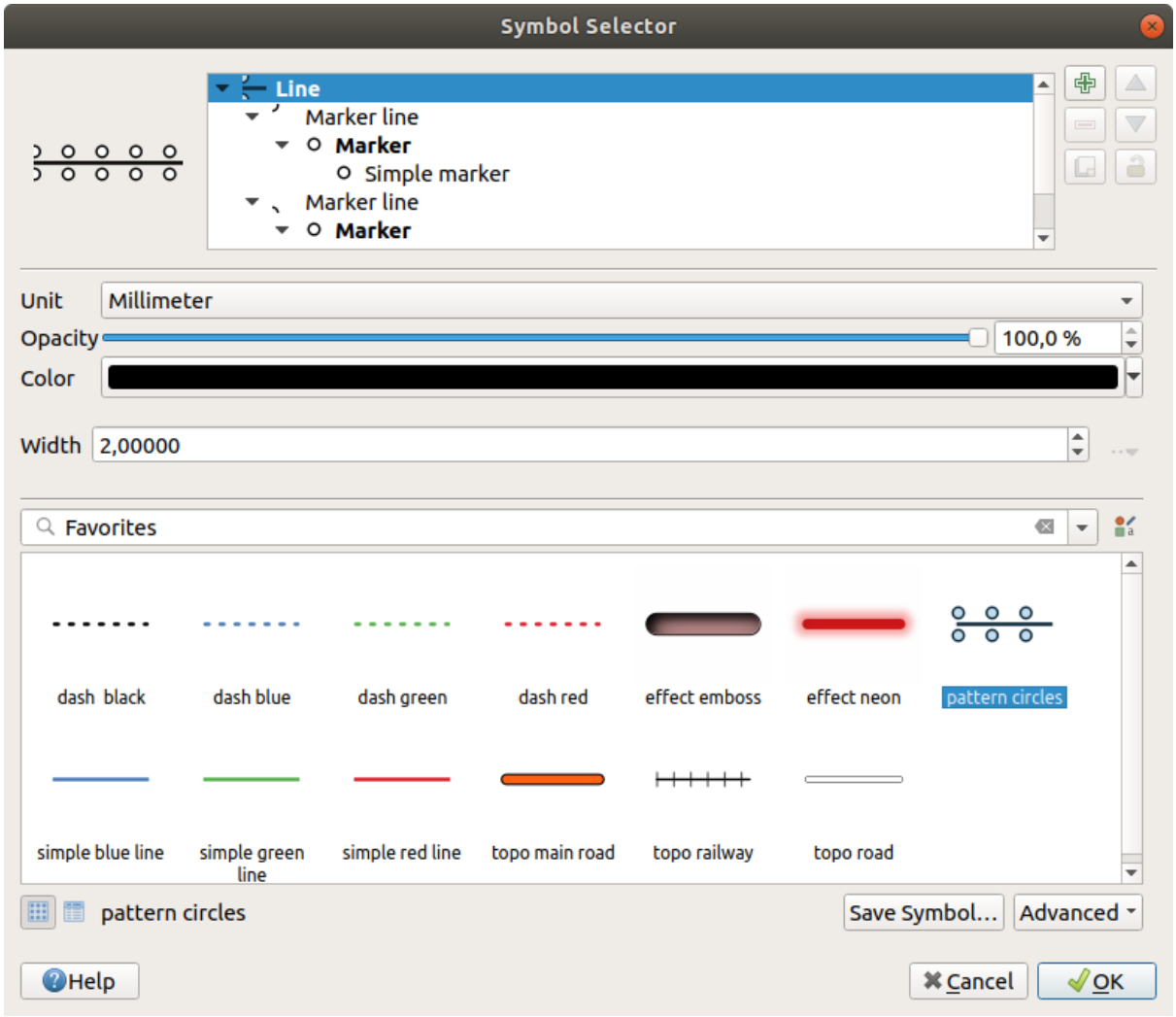


Fig. 14.10: Designing a Line symbol

Two main components structure the symbol selector dialog:

- the symbol tree, showing symbol layers that are combined afterwards to shape a new global symbol
- and settings to configure the selected symbol layer in the tree.

14.2.1 The symbol layer tree

A symbol can consist of several *Symbol layers*. The symbol tree shows the overlay of these symbol layers that are combined afterwards to shape a new global symbol. Besides, a dynamic symbol representation is updated as soon as symbol properties change.

Depending on the level selected in the symbol tree items, various tools are made available to help you manage the tree:

- add new symbol layer: you can stack as many symbols as you want
- remove the selected symbol layer
- lock colors of symbol layer: a locked color stays unchanged when user changes the color at the global (or upper) symbol level
- duplicate a (group of) symbol layer(s)

- move up or down the symbol layer

14.2.2 Configuring a symbol

In QGIS, configuring a symbol is done in two steps: the symbol and then the symbol layer.

The symbol

At the top level of the tree, it depends on the layer geometry and can be of **Marker**, **Line** or **Fill** type. Each symbol can embed one or more symbols (including, of any other type) or symbol layers.

You can setup some parameters that apply to the global symbol:

- *Unit*: it can be **Millimeters**, **Points**, **Pixels**, **Meters at Scale**, **Map units** or **Inches** (see *Unit Selector* for more details)
- *Opacity*
- *Color*: when this parameter is changed by the user, its value is echoed to all unlocked sub-symbols color
- *Size* and *Rotation* for marker symbols
- *Width* for line symbols

Sfat: Use the *Size* (for marker symbols) or the *Width* (for line symbols) properties at the symbol level to proportionally resize all of its embedded *symbol layers* dimensions.

Notă: The *Data-defined override* button next to the width, size or rotation parameters is inactive when setting the symbol from the Style manager dialog. When the symbol is connected to a map layer, this button helps you create *proportional or multivariate analysis* rendering.

- A preview of the *symbols library*: Symbols of the same type are shown and, through the editable drop-down list just above, can be filtered by free-form text or by *categories*. You can also update the list of symbols using the Style Manager button and open the eponym dialog. There, you can use any capabilities as exposed in *The Style Manager* section.

  The symbols are displayed either:

  - in an icon list (with thumbnail, name and associated tags) using the List View button below the frame;
  - or as icon preview using the Icon View button.
- Press the *Save Symbol* button to add the symbol being edited to the symbols library.
- With the *Advanced* option, you can:
  - for line and fill symbols, *Clip features to canvas extent*.
  - for fill symbols, *Force right-hand rule orientation*: allows forcing rendered fill symbols to follow the standard „right hand rule” for ring orientation (i.e, polygons where the exterior ring is clockwise, and the interior rings are all counter-clockwise).

    The orientation fix is applied while rendering only, and the original feature geometry is unchanged. This allows for creation of fill symbols with consistent appearance, regardless of the dataset being rendered and the ring orientation of individual features.
  - Depending on the *symbology* of the layer a symbol is being applied to, additional settings are available in the *Advanced* menu:
    - *Symbol levels…* to define the order of symbols rendering

- *Data-defined Size Legend*
- *Match to Saved Symbols…* and *Match to Symbols from File…* to automatically *assign symbols to classes*
- *Animation settings*

The symbol layer

At a lower level of the tree, you can customize the symbol layers. The available symbol layer types depend on the upper symbol type. You can apply on the symbol layer *paint effects* to enhance its rendering.

Because describing all the options of all the symbol layer types would not be possible, only particular and significant ones are mentioned below.

Parametri comuni

Some common options and widgets are available to build a symbol layer, regardless it's of marker, line or fill sub-type:

- the *color selector* widget to ease color manipulation
- *Units*: it can be **Millimeters**, **Points**, **Pixels**, **Meters at Scale**, **Map units** or **Inches** (see *Unit Selector* for more details)
- the Data-defined override widget near almost all options, extending capabilities of customizing each symbol (see *Configurarea suprascrierii definită de date* for more information)
- the *Enable symbol layer* option controls the symbol layer's visibility. Disabled symbol layers are not drawn when rendering the symbol but are saved in the symbol. Being able to hide symbol layers is convenient when looking for the best design of your symbol as you don't need to remove any for the testing. The data-defined override then makes it possible to hide or display different symbol layers based on expressions (using, for instance, feature attributes).
- the *Draw effects* button for *effects rendering*.

Notă: While the description below assumes that the symbol layer type is bound to the feature geometry, keep in mind that you can embed symbol layers in each others. In that case, the lower level symbol layer parameter (placement, offset…) might be bound to the upper-level symbol, and not to the feature geometry itself.

Marker Symbols

Appropriate for point geometry features, marker symbols have several *Symbol layer types*:

- **Simple marker** (default)

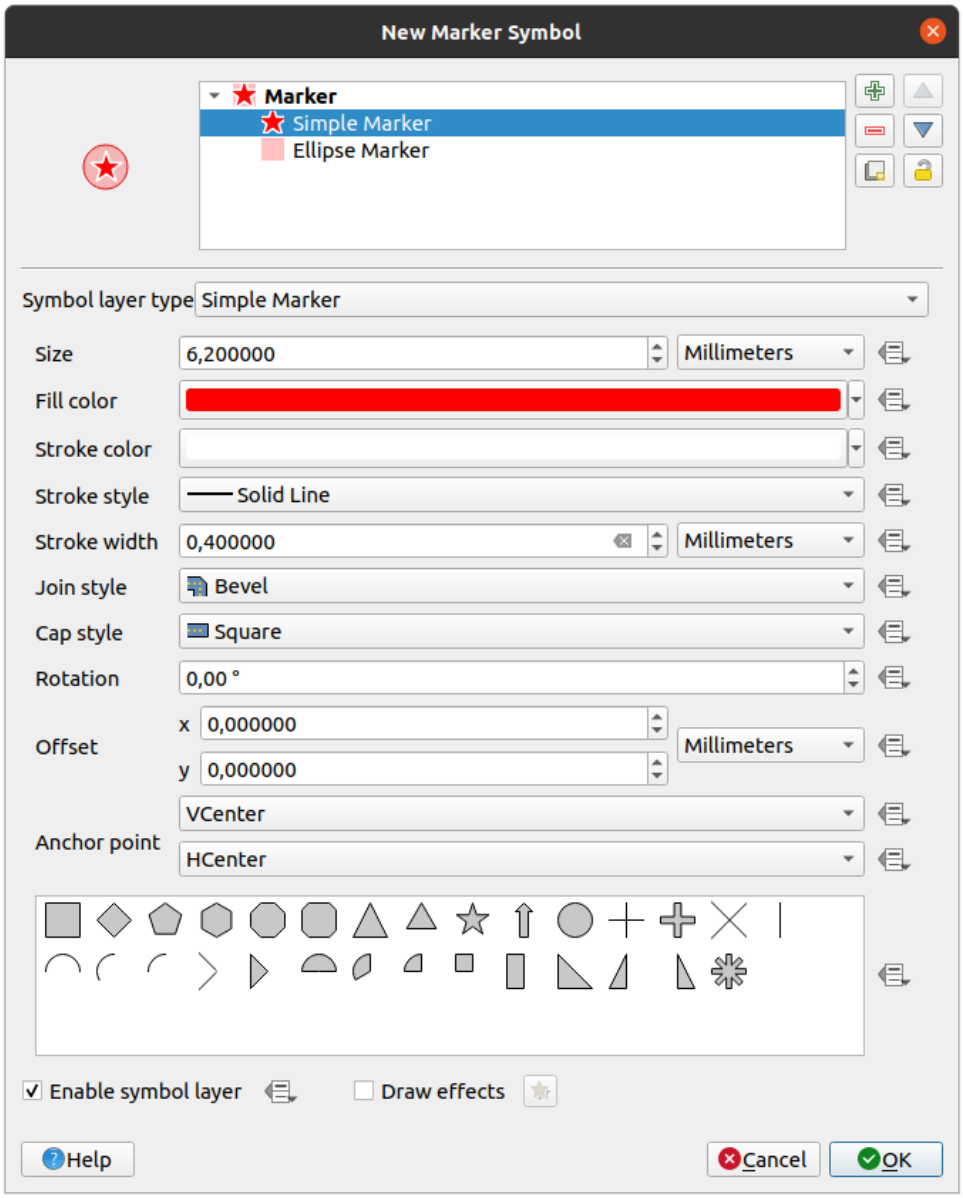


Fig. 14.11: Designing a Simple Marker Symbol

The simple marker symbol layer type has the following properties:

- *Size* in various supported units
- *Fill color*
- *Stroke color*, *Stroke style* from a predefined list and *Stroke size*
- *Join style*: it can be **Bevel**, **Miter** or **Round**
- *Cap style*: it can be **Square**, **Flat** or **Round**
- *Rotation*
- *Offset* in *X* and *Y* directions from the feature
- *Anchor point*: defining the quadrant point on the symbol to settle as placement origin. This is the point the *Offset* is applied on.

- **Animated marker** (see *Animated marker*)
- **Ellipse marker**: a simple marker symbol layer, with customizable width and height

- **Filled marker**: similar to the simple marker symbol layer, except that it uses a *fill sub symbol* to render the marker. This allows use of all the existing QGIS fill (and stroke) styles for rendering markers, e.g. gradient or shapeburst fills.
- **Font marker**: similar to the simple marker symbol layer, except that it uses installed fonts to render the marker. Its additional properties are:
  - *Familia fontului*
  - *Stilul fontului*
  - *Character(s)*, representing the text to display as symbol. They can be typed in or selected from the font characters collection widget and you can live *Preview* them with the selected settings.
- **Geometry generator** (see *Generatorul de Geometrie*)
- **Mask**: its sub-symbol defines a mask shape whose color property will be ignored and only the opacity will be used. This is convenient when the marker symbol overlaps with labels or other symbols whose colors are close, making it hard to decipher. More details at *Masks Properties*.
- **Raster image marker**: use an image (`PNG`, `JPG`, `BMP` ...) as marker symbol. The image can be a file on the disk, a remote URL, embedded in the style database (*more details*) or it can be encoded as a base64 string. Width and height of the image can be set independently or using the Lock aspect ratio. The size can be set using any of the *common units* or as a percentage of the image's original size (scaled by the width).
- **Vector Field marker** (see *The Vector Field Marker*)
- **SVG marker**: provides you with images from your SVG paths (set in *Settings* ► *Options...* ► *System* menu) to render as marker symbol. Width and height of the symbol can be set independently or using the Lock aspect ratio. Each SVG file colors and stroke can also be adapted. The image can be a file on the disk, a remote URL, embedded in the style database (*more details*) or it can be encoded as a base64 string.

  The symbol can also be set with *Dynamic SVG parameters*. See *SVG parametrizabil* section to parametrize an SVG symbol.

Notă: Cerințele versiunii SVG

QGIS renders SVG files that follow the SVG Tiny 1.2 profile, intended for implementation on a range of devices, from cellphones and PDAs to laptop and desktop computers, and thus includes a subset of the features included in SVG 1.1 Full, along with new features to extend the capabilities of SVG.

Some features not included in these specifications might not be rendered correctly in QGIS.

Line Symbols

Appropriate for line geometry features, line symbols have the following symbol layer types:

- **Linie simplă** (implicit)

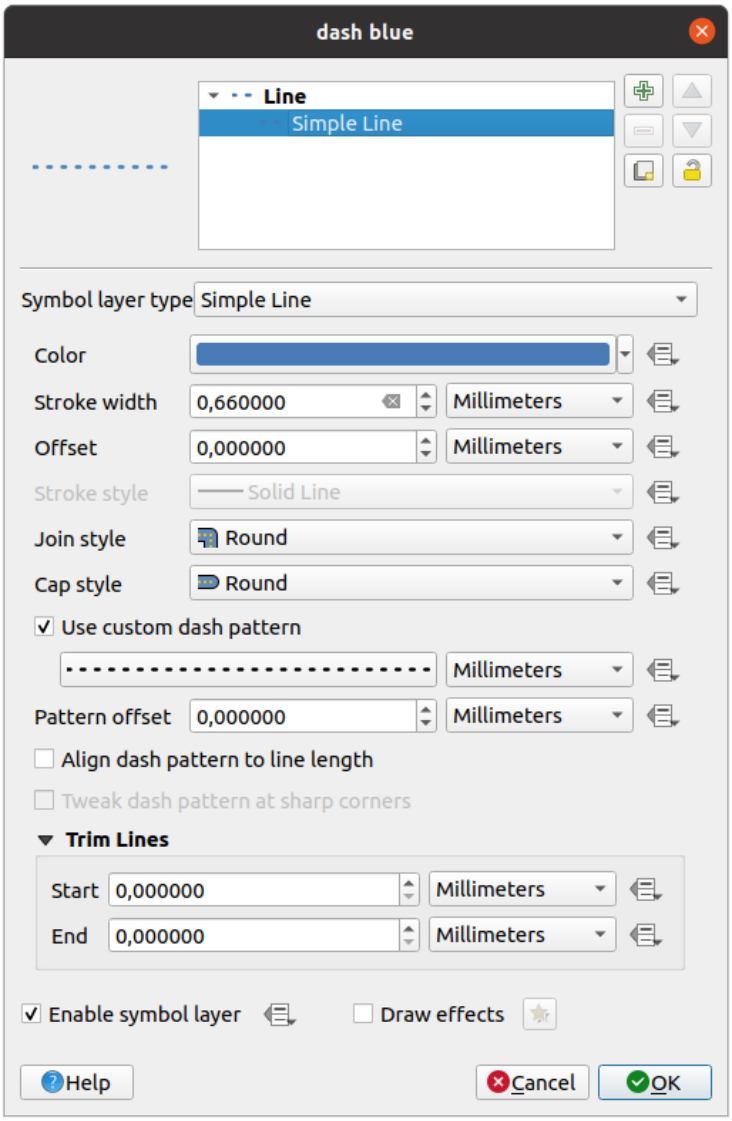


Fig. 14.12: Designing a Simple Line Symbol

The simple line symbol layer type has many of the same properties as the *simple marker symbol*, and in addition:

- *Use custom dash pattern*: overrides the *Stroke style* setting with a custom dash. You would need to define length of consecutive dashes and spaces shaping the model, in the chosen unit. The total length of the pattern is displayed at the bottom of the dialog.
- *Pattern offset*: the positioning of the dashes/spaces in the line can be tweaked, so that they can be placed at nicer positions to account for corners in the line (also can be used potentially to „align" adjacent dash pattern borders)
- *Align dash pattern to line length*: the dash pattern length will be adjusted so that the line will end with a complete dash element, instead of a gap.
- *Tweak dash pattern at sharp corners*: dynamically adjusts the dash pattern placement so that sharp

corners are represented by a full dash element coming into and out of the sharp corner. Dependent on *Align dash pattern to line length*.

  - *Trim lines* from *Start* and/or *End*: allows for the line rendering to trim off the first x mm and last y mm from the actual line string when drawing the line. It supports a range of *units*, including percentage of the overall line length, and can be data defined for extra control. The start/end trim distance can be used e.g. when creating complex symbols where a line layer should not overlap marker symbol layers placed at the start and end of the line.

- **Arrow**: draws lines as curved (or not) arrows with a single or a double head with configurable (and data-defined):
  - *Head type*
  - *Tipul săgeții*
  - *Lățimea săgeții*
  - *Arrow width at start*
  - *Head length*
  - *Head thickness*
  - *Offset*

  It is possible to create *Curved arrows* (the line feature must have at least three vertices) and *Repeat arrow on each segment*. It also uses a *fill symbol* such as gradients or shapeburst to render the arrow body. Combined with the geometry generator, this type of layer symbol helps you representing flow maps.

- **Geometry generator** (see *Generatorul de Geometrie*)

- **Interpolated line**: allows to render a line whose *Stroke width* and/or *Color* may be constant (given a *Fixed width* and *Single color* parameters) or vary along the geometry. When varying, necessary inputs are:
  - *Start value* and *End value*: Values that will be used for interpolation at the extremities of the features geometry. They can be fixed values, feature's attributes or based on an expression.
  - *Min. value* and *Max. value*: Values between which the interpolation is performed. Press the Load button to automatically fill them based on the minimum and maximum start/end values applied to the layer.
  - Only available for the stroke option:
    * *Min. width* and *Max. width*: define the range of the varying width. *Min. width* is assigned to the *Min. value* and *Max. width* to the *Max. value*. A *unit* can be associated.
    * *Use absolute value*: only consider absolute value for interpolation (negative values are used as positive).
    * *Ignore out of range*: by default, when the `[start value - end value]` range of a feature is not included in the `[min. value - max. value]` range, the out-of-bounds parts of the feature's geometry are rendered with the min or max width. Check this option to not render them at all.
  - For varying color, you can use any of the interpolation methods of *color ramp classification*

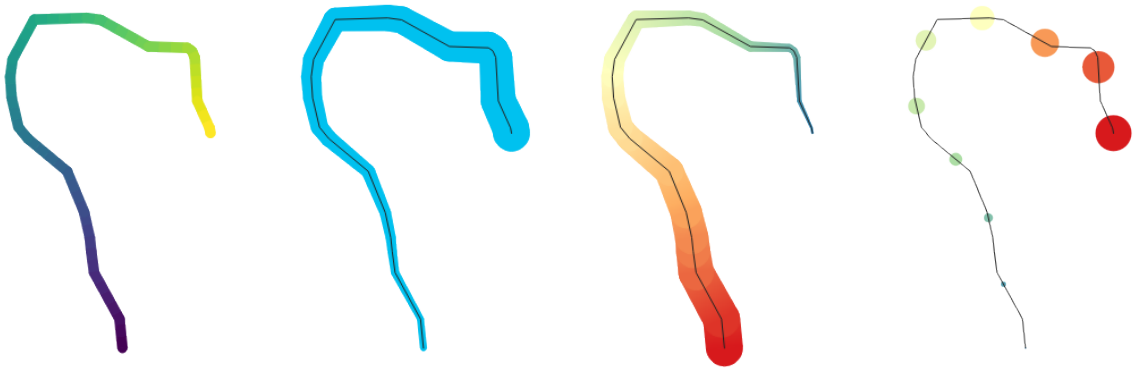


Fig. 14.13: Examples of interpolated lines

- **Marker line**: repeats a *marker symbol* over the length of a line.
  - The *Marker placement* can be set using a regular interval setting (starting from the first vertex) and/or the line geometry property (on first or last vertex, inner vertices, the central point of the line or of each segment, or on every curve point).
  - When first or last vertex placement is enabled, the *Place on every part extremity* option will make the markers render also at the first or last vertex for every part of multipart geometries.
  - *Offset along line*: the markers placement can also be given an offset along the line, in the ref:*unit <unit_selector>* of your choice (millimeters, points, map unit, meters at scale, percentage, …):
    - A positive value offsets the markers symbols in the line direction (with *On first vertex* and *With interval* placements) and backwards (with *On last vertex* placement).
    - A negative value on a not closed line will result in no offset (for *On first vertex* and *On last vertex* placements) or backwards offset of the symbols (from the last vertex).
    - With a closed ring, QGIS treats the offset as continuing to loop around the ring (forward or backward). E.g. setting the offset to 150% (resp. -10% or -110%) results in the offset being treated as 50% (resp. 90%) of the length of the closed ring.
  - The *Rotate marker to follow line direction* option sets whether each marker symbol should be oriented relative to the line direction or not.

    Because a line is often a succession of segments of different directions, the rotation of the marker is calculated by averaging over a specified distance along the line. For example, setting the *Average angle over* property to `4mm` means that the two points along the line that are `2mm` before and after the symbol placement are used to calculate the line angle for that marker symbol. This has the effect of smoothing (or removing) any tiny local deviations from the overall line direction, resulting in much nicer visual orientations of the marker line symbols.
  - *Line offset*: the marker symbols can also be offset from the line feature.
- **Hashed line**: repeats a line segment (a hash) over the length of a line symbol, with a line sub-symbol used to render each individual segment. In other words, a hashed line is like a marker line in which marker symbols are replaced with segments. As such, the hashed lines have the *same properties* as marker line symbols, along with:
  - *Hash length*
  - *Hash rotation*

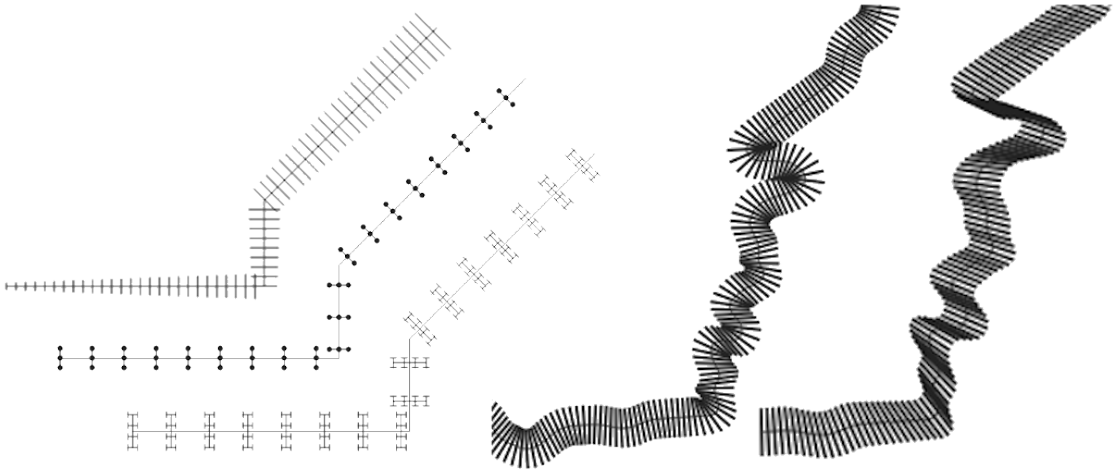


Fig. 14.14: Examples of hashed lines

- **Raster line**: renders and repeats a raster image following the length of a line feature shape. The *Stroke width*, *Offset*, *Join style*, *Cap style* and *Opacity* can be adjusted.

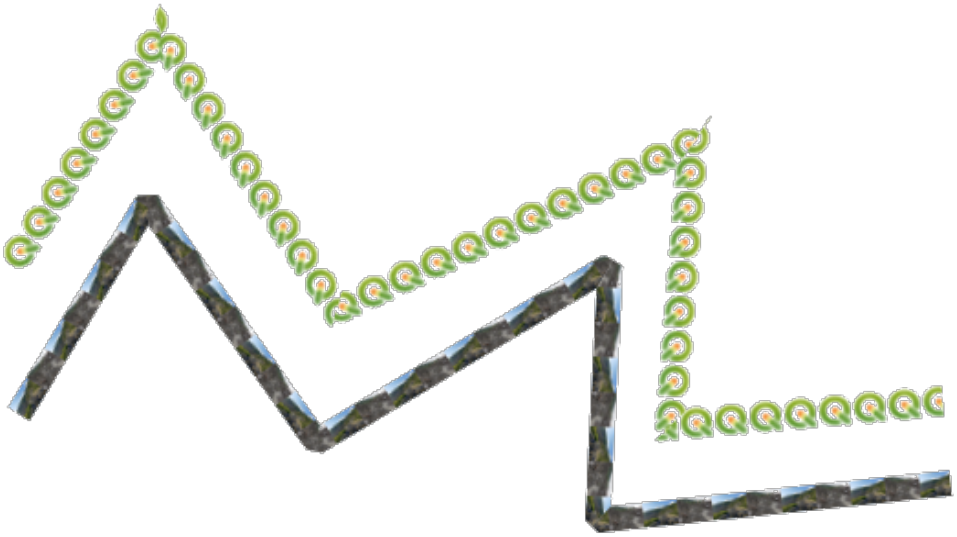


Fig. 14.15: Examples of raster lines

- **Lineburst**: renders a gradient along the width of a line. You can choose between *Two color* or *Color ramp* and the *Stroke width*, *Offset*, *Join style*, *Cap style* can be adjusted.

Fig. 14.16: Examples of lineburst lines

Fill Symbols

Appropriate for polygon geometry features, fill symbols have also several symbol layer types:

- **Simple fill** (default): fills a polygon with a uniform color

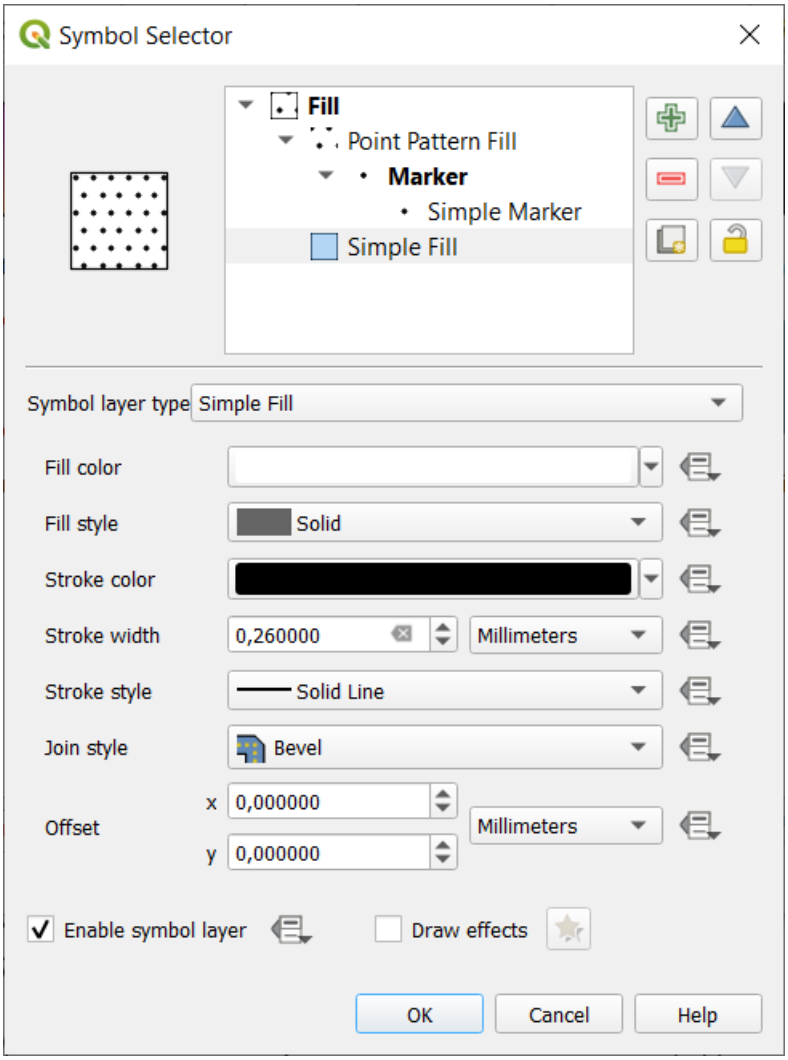


Fig. 14.17: Designing a Simple Fill Symbol

- **Centroid fill**: places a *marker symbol* at the centroid of the visible feature. The position of the marker may not be the real centroid of the feature, because calculation takes into account the polygon(s) clipped to area visible in map canvas for rendering and ignores holes. Use the *geometry generator symbol* if you want the exact centroid.

  Puteți:

  - *Force placement of markers inside polygons*
  - *Draw markers on every part of multi-part features* or place the point only on its biggest part
  - display the marker symbol(s) in whole or in part, keeping parts overlapping the current feature geometry (*Clip markers to polygon boundary*) or the geometry part the symbol belongs to (*Clip markers to current part boundary only*)

- **Geometry generator** (see *Generatorul de Geometrie*)
- **Gradient fill**: uses a radial, linear or conical gradient, based on either simple two color gradients or a predefined *gradient color ramp* to fill polygons. The gradient can be rotated and applied on a single feature basis or across the whole map extent. Also start and end points can be set via coordinates or using the centroid (of feature or map). A data-defined offset can be defined.
- **Line pattern fill**: fills the polygon with a hatching pattern of *line symbol layer*. You can set:

  - *Alignment*: defines how the pattern is positioned relative to the feature(s):

        - *Align pattern to feature*: lines are rendered within each feature
        - *Align pattern to map extent*: a pattern is rendered over the whole map extent, allowing lines to align nicely across features
    - *Rotation* of the lines, counter-clockwise
    - *Spacing*: distance between consecutive lines
    - *Offset* distance of the lines from the feature boundary
    - *Clipping*: allows to control how lines in the fill should be clipped to the polygon shape. Options are:
        - *Clip During Render Only*: lines are created covering the whole bounding box of the feature and then clipped while drawing. Line extremities (beginning and end) will not be visible.
        - *Clip Lines Before Render*: lines are clipped to the exact shape of the polygon prior to rendering. Line extremities (including cap styles, start/end marker line objects, …) will be visible, and may sometimes extend outside of the polygon (depending on the line symbol settings).
        - *No Clipping*: no clipping at all is done - lines will cover the whole bounding box of the feature
- **Point pattern fill**: fills the polygon with a grid pattern of *marker symbol*. You can set:
    - *Alignment*: defines how the pattern is positioned relative to the feature(s):
        - *Align pattern to feature*: marker lines are rendered within each feature
        - *Align pattern to map extent*: a pattern is rendered over the whole map extent, allowing markers to align nicely across features

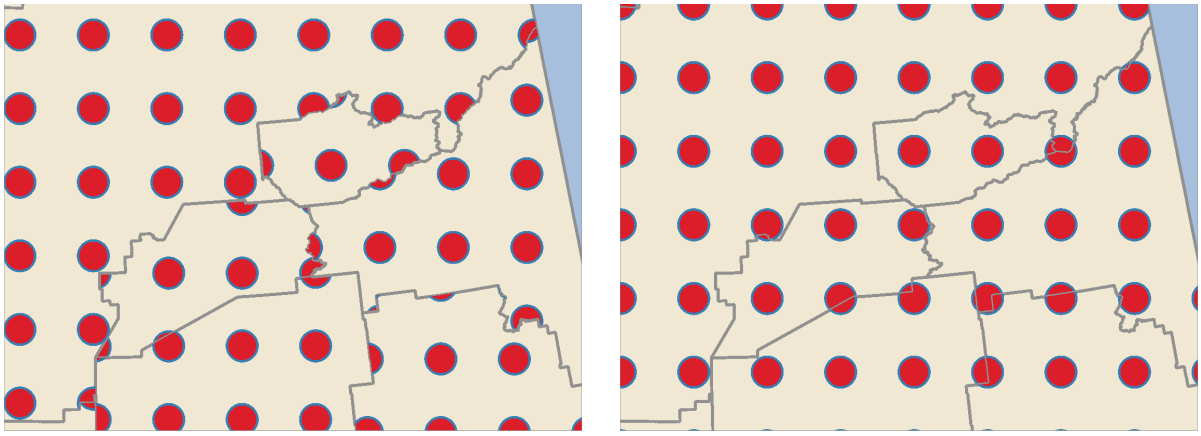


Fig. 14.18: Aligning point pattern to feature (left) and to map extent (right)

    - *Distance*: *Horizontal* and *Vertical* distances between consecutive markers
    - *Displacement*: a *Horizontal* (resp. *Vertical*) offset of alignment between consecutive markers in a column (resp. in a row)
    - *Offset*: *Horizontal* and *Vertical* distances from the feature boundary
    - *Clipping*: allows to control how markers in the fill should be clipped to the polygon shape. Options are:
        - *Clip to shape*: markers are clipped so that only the portions inside the polygon are visible
        - *Marker centroid within shape*: only markers where the center of the marker falls inside the polygon are drawn, but these markers won't be clipped to the outside of the polygon
        - *Marker completely within shape*: only markers which fall completely within the polygon are shown
        - *No clipping*: any marker which intersects at all with the polygon will be completely rendered (strictly speaking its the „intersects with the bounding box of the marker")

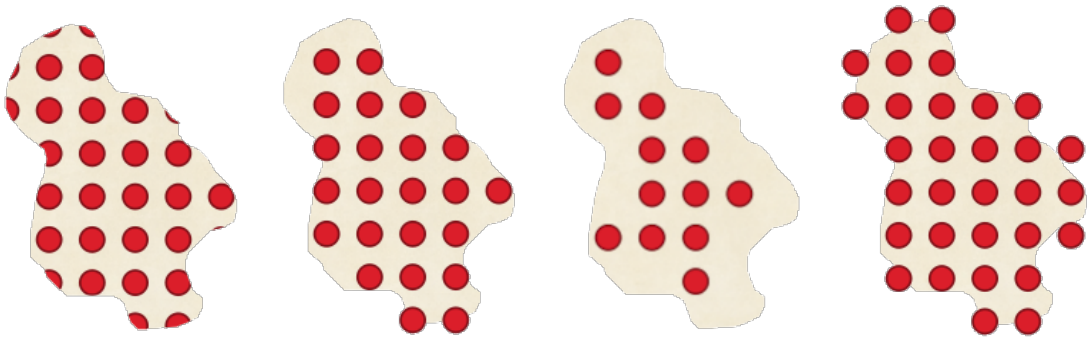


Fig. 14.19: Clipping markers in fill - From left to right: Clip to shape, Marker centroid within shape, Marker completely within shape, No clipping

  - *Rotation* of the whole pattern, clockwise
  - The *Randomize pattern* group setting allows each point in a point pattern fill to be randomly shifted up to the specified maximum distance *Horizontally* or *Vertically*. You can specify the maximum offset in any supported units, such as millimeters, points, map units, or even „percentage" (where percentage is relative to the pattern width or height).

    You can set an optional random number seed to avoid the symbol patterns „jumping" around between map refreshes. Data defined overrides are also supported.

    Notă: The main difference between the *Randomize pattern* and the *random marker fill* symbol type is that the random offset with a point pattern allows for quasi-„regular" placement of markers – because the points in the pattern are effectively constrained to a grid, this allows creation of semi-random fills which don't have empty areas or overlapping markers. (As opposed to the random marker fill, which will always place points completely randomly… sometimes resulting in visual clusters of points or unwanted empty areas).

- **Random marker fill**: fills the polygon with a *marker symbol* placed at random locations within the polygon boundary. You can set:
  - *Count method*: whether the number of marker symbols to render is considered as an absolute count or density-based
  - *Point count*: the number of marker symbols to render,
  - an optional random number *seed*, to give consistent placement
  - *Density area*: in case of density-based count method, ensures the fill density of markers remains the same on different scale / zoom levels of markers whenever maps are refreshed (also allows random placement to play nice with QGIS server and tile-based rendering)
  - *Clip markers to polygon boundary*: whether markers rendered near the edges of polygons should be clipped to the polygon boundary or not
- **Raster image fill**: fills the polygon with tiles from a raster image (`PNG` `JPG`, `BMP` …). The image can be a file on the disk, a remote URL or an embedded file encoded as a string (*more details*). Options include (data defined) opacity, image width, coordinate mode (object or viewport), rotation and offset. The image width can be set using any of the *common units* or as a percentage of the original size.
- **SVG fill**: fills the polygon using *SVG markers* of a given size (*Texture width*).
- **Shapeburst fill**: buffers a gradient fill, where a gradient is drawn from the boundary of a polygon towards the polygon's centre. Configurable parameters include distance from the boundary to shade, use of color ramps or simple two color gradients, optional blurring of the fill and offsets.
- **Outline: Arrow**: uses a line *arrow symbol* layer to represent the polygon boundary. The settings for the outline arrow are the same as for arrow line symbols.

- **Outline: Hashed line**: uses a *hash line symbol* layer to represent the polygon boundary (*Rings*) which can be the interior rings only, the exterior ring only or all the rings). The other settings for the outline hashed line are the same as for hashed line symbols.
- **Outline: Marker line**: uses a *marker line symbol* layer to represent the polygon boundary (*Rings*) which can be the interior rings only, the exterior ring only or all the rings). The other settings for the outline marker line are same as for marker line symbols.
- **Outline: simple line**: uses a *simple line symbol* layer to represent the polygon boundary (*Rings*) which can be the interior rings only, the exterior ring only or all the rings). The *Draw line only inside polygon* option displays the polygon borders inside the polygon and can be useful to clearly represent adjacent polygon boundaries. The other settings for the outline simple line are the same as for simple line symbols.

Notă: When geometry type is polygon, you can choose to disable the automatic clipping of lines/polygons to the canvas extent. In some cases this clipping results in unfavourable symbology (e.g. centroid fills where the centroid must always be the actual feature's centroid).

SVG parametrizabil

You have the possibility to change the colors of a *SVG marker*. You have to add the placeholders `param(fill)` for fill color, `param(fill-opacity)` for fill opacity, `param(outline)` and `param(outline-opacity)` for stroke color and opacity respectively, and `param(outline-width)` for stroke width. These placeholders can optionally be followed by a default value, e.g.:

```
<svg width="100%" height="100%">
<rect fill="param(fill) #ff0000" fill-opacity="param(fill-opacity) 1" stroke=
↪"param(outline) #00ff00" stroke-opacity="param(outline-opacity) 1" stroke-width=
↪"param(outline-width) 10" width="100" height="100">
</rect>
</svg>
```

More generally, SVG can be freely parametrized using `param(param_name)`. This param can either be used as an attribute value or a node text:

```
<g stroke-width=".265" text-anchor="middle" alignment-baseline="param(align)">
  <text x="98" y="147.5" font-size="6px">param(text1)</text>
  <text x="98" y="156.3" font-size="4.5px">param(text2)</text>
</g>
```

The parameters can then be defined as expressions in the *Dynamic SVG parameters* table.

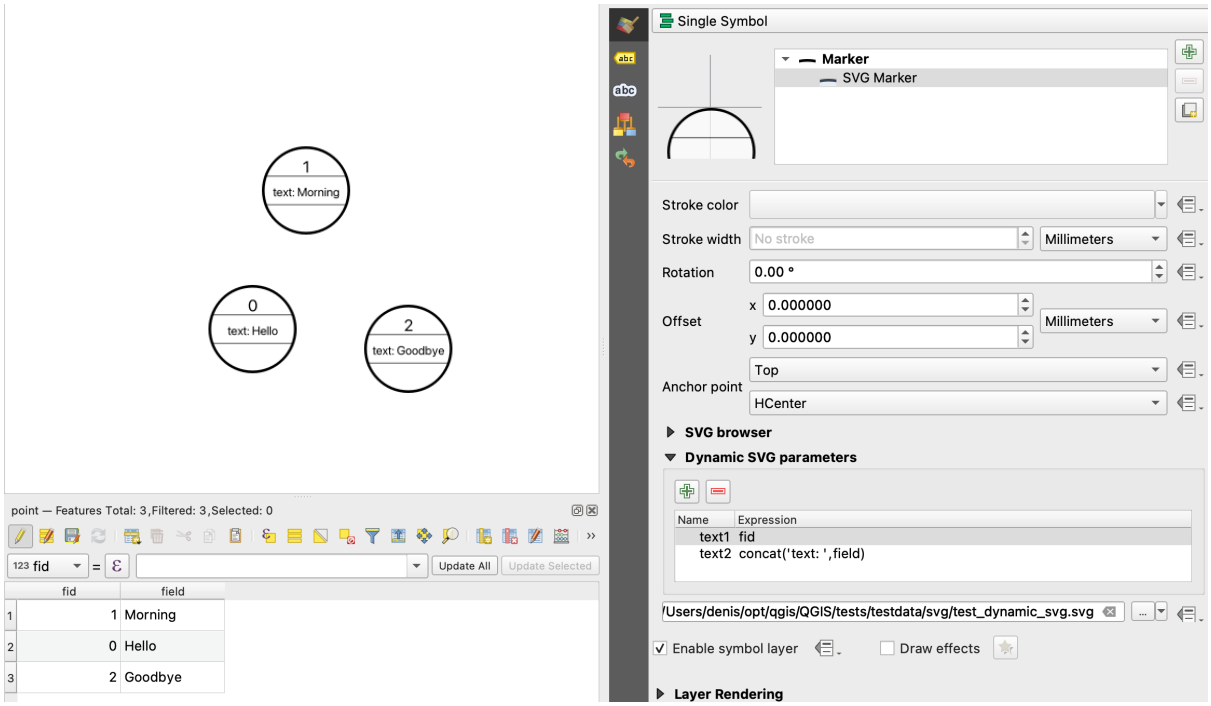


Fig. 14.20: Dynamic SVG parameters table

Notă: QGIS is looking for a complete SVG node. So if your parameter is within a more complex node, you need to inject the complete node with the expression. For instance, `transform="rotate(param(angle)"` will not work. Instead, you need to do `transform="param(rotation)"` and `rotation` parameter will be defined with the expression `'rotate(' || coalesce(my_field, 0) || ')'`.

Generatorul de Geometrie

Available with all types of symbols, the *geometry generator* symbol layer allows to use *expression syntax* to generate a geometry on the fly during the rendering process. The resulting geometry does not have to match with the original *Geometry type* and you can add several differently modified symbol layers on top of each other.

A *Units* property can be set: when the geometry generator symbol is not applied to a layer (e.g., it is used on a layout item), this allows more control over the generated output.

Câteva exemple:

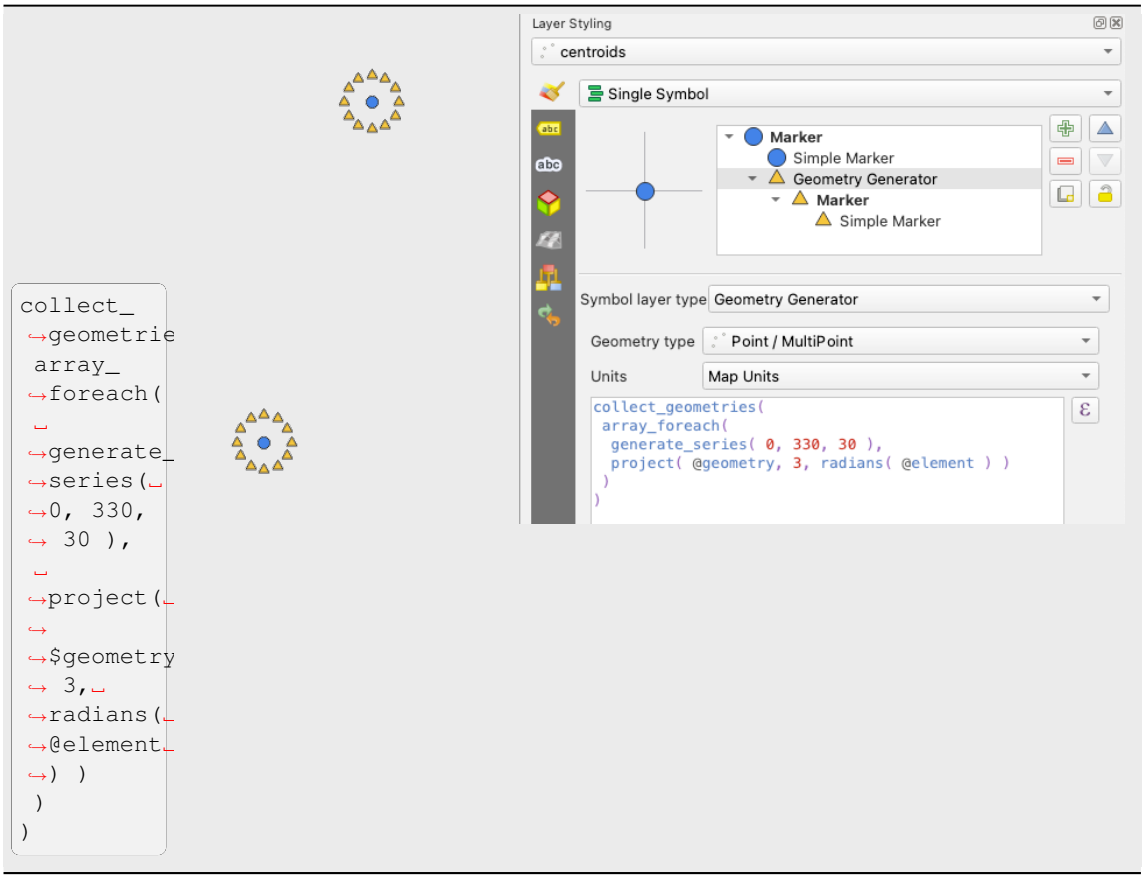
- Render symbol as the centroid of a feature

```
centroid( $geometry )
```

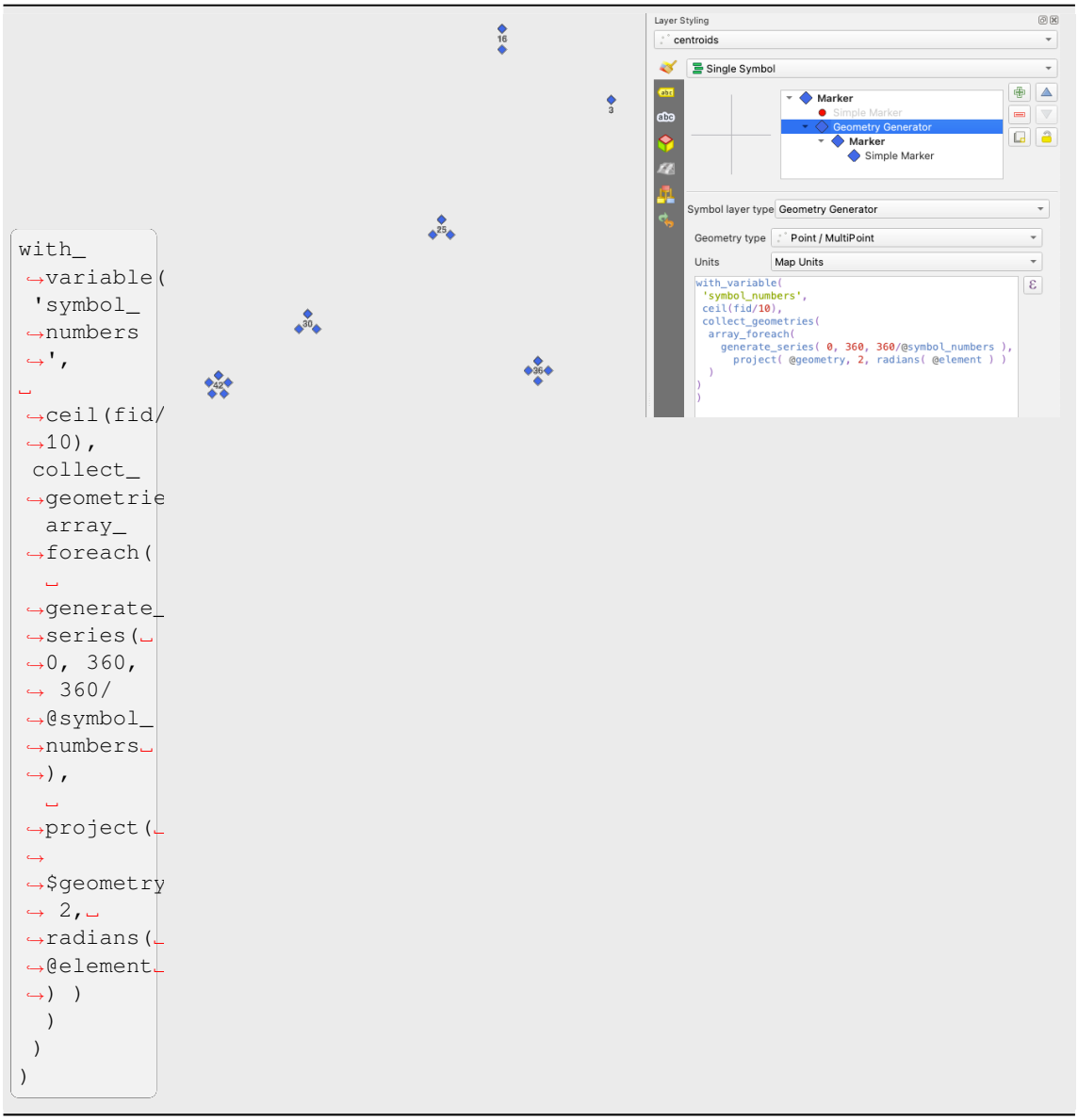
- Visually overlap features within a 100 map units distance from a point feature, i.e generate a 100m buffer around the point

```
buffer( $geometry, 100 )
```

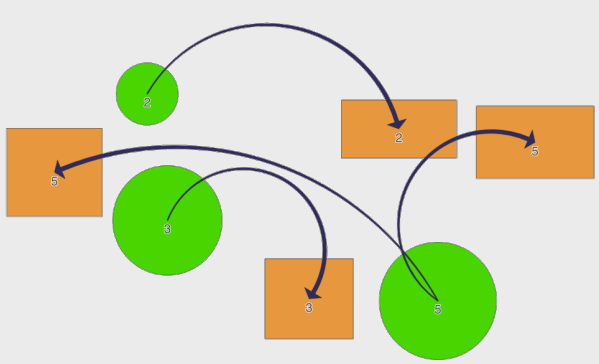
- Create a radial effect of points surrounding the central feature point when used as a MultiPoint geometry generator

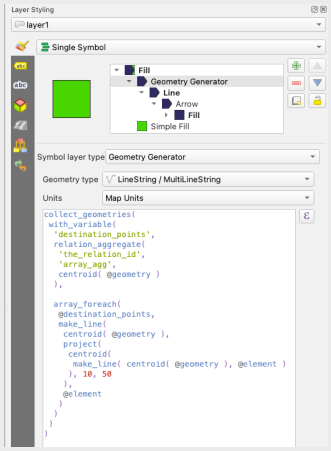


- Create a radial effect of points surrounding the central feature point. The number of points varies based on a field.



- Create a curved arrow line connecting features of two layers based on their *relation*





```
collect_
↪geometrie
 with_
↪variable(

↪'destinat
↪points
↪',
 ␣
↪relation_
↪aggregate
   'the_
↪relation_
↪id',

↪'array_
↪agg',
  ␣
↪centroid(
↪
↪$geometry
↪)
  ),
  array_
↪foreach(
  ␣
↪@destinat
↪points,
   make_
↪line(
   ␣
↪centroid(
↪@geometry
↪),
   ␣
↪project(
    ␣
↪centroid(
     ␣
↪make_
↪line(␣
↪centroid(
↪@geometry
↪),␣
↪@element␣
↪)
     ),
     10,␣
↪50
    ),
   ␣
↪@element
   )
  )
 )
)
```

The Vector Field Marker

The vector field marker is used to display vector field data such as earth deformation, tidal flows, and the like. It displays the vectors as lines (preferably arrows) that are scaled and oriented according to selected attributes of data points. It can only be used to render point data; line and polygon layers are not drawn by this symbology.

The vector field is defined by attributes in the data, which can represent the field either by:

- **cartesian** components (`x` and `y` components of the field)
- or **polar** coordinates: in this case, attributes define `Length` and `Angle`. The angle may be measured either clockwise from north, or Counterclockwise from east, and may be either in degrees or radians.
- or as **height only** data, which displays a vertical arrow scaled using an attribute of the data. This is appropriate for displaying the vertical component of deformation, for example.

The magnitude of field can be scaled up or down to an appropriate size for viewing the field.

Animated marker

Animated marker symbol type allows you to use a `.GIF`, `.WebP`, `.MNG`, etc. animation file to represent points on your map. You can specify:

- *File* path,
- *Frame rate*: number of steps that are shown per second, indicating how fast the animation is played,
- *Size* in any *supported unit*,
- *Opacity*,
- *Rotation*,
- *Offset* in *x* and *y* directions from the marker position,
- *Anchor point*

There are two ways to handle animated symbols:

- **When your map is not configured as an animation** (i.e. it's a standard QGIS project without animations), the frame for the animated markers will be determined solely by the current timestamp.
- **When your map is** *configured as an animation*, the animated markers will sync with the animation's timeline. This means that animated markers will pause when the animation is paused, progress with the animation, and so forth. The map will also be redrawn according to the frame rate established for temporal animation. This mode is also applied when exporting an animation using the temporal controller.

14.3 Setting a label

Labels are textual information you can display on vector features or maps. They add details you could not necessarily represent using symbols. Two types of text-related items are available in QGIS:

- *Text Format*: defines the appearance of the text, including *font, size, colors*, *shadow*, *background*, *buffer*, …

  They can be used to render texts over the map (layout/map title, decorations, scale bar, …), usually through the *font* widget.

  To create a *Text Format* item:

  1. Open the *Style Manager* dialog

2. Activate the *Text format* tab

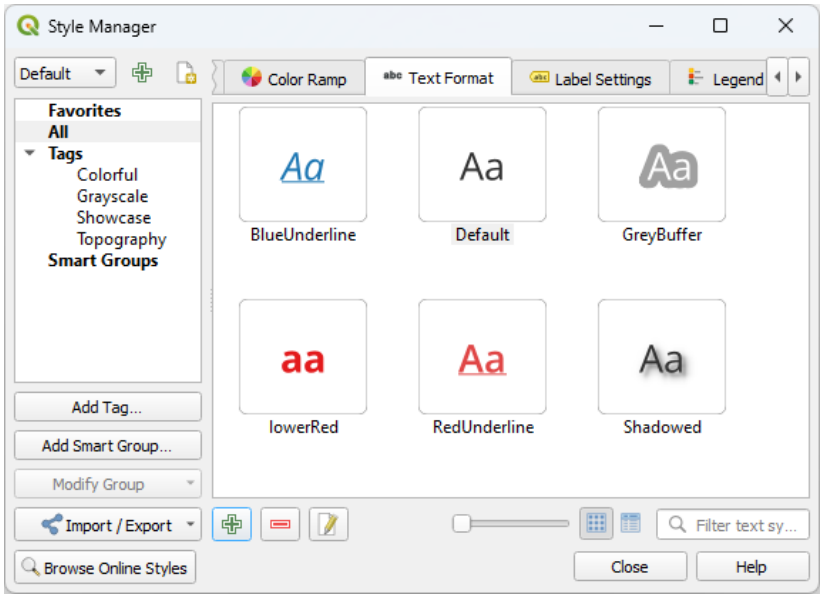


Fig. 14.21: Text formats in Style Manager dialog

3. Press the Add item button. The *Text Format* dialog opens for *configuration*. As usual, these properties are *data-definable*.

- *Label Settings*: extend the text format settings with properties related to the location or the interaction with other texts or features (*callouts*, *placement*, *overlay, scale visibility*, mask …).

  They are used to configure smart labelling for vector layers through the *Labels* tab of the vector *Layer Properties* dialog or *Layer Styling* panel or using the Layer Labeling Options button of the *Label toolbar*.

  To create a *Label Settings* item:

  1. Open the *Style Manager* dialog
  2. Activate the *Label Settings* tab

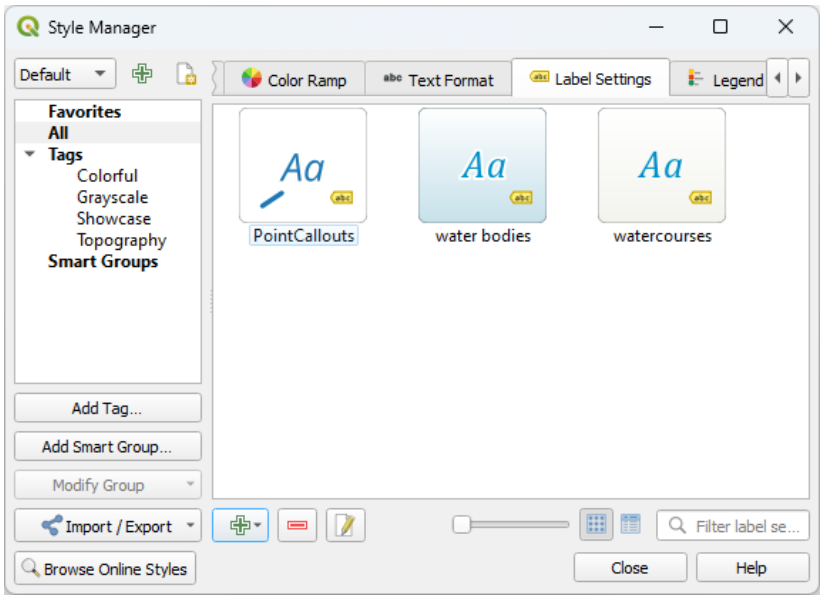


Fig. 14.22: Label Settings in Style Manager dialog

3. Press the Add item menu and select the entry corresponding to the geometry type of the features you want to label.

The *Label Settings* dialog opens with the following properties. As usual, these properties are *data-definable*.

14.3.1 Formatting the label text

Whether you are configuring a *Text Format* or *Label Settings* item, you will be given the following options:

Properties tab	Text format	Label settings
Text	☑	☑
Formatting	☑	☑
Buffer	☑	☑
Mask		☑
Background	☑	☑
Shadow	☑	☑
Callout		☑
Placement		☑
Rendering		☑

Text tab

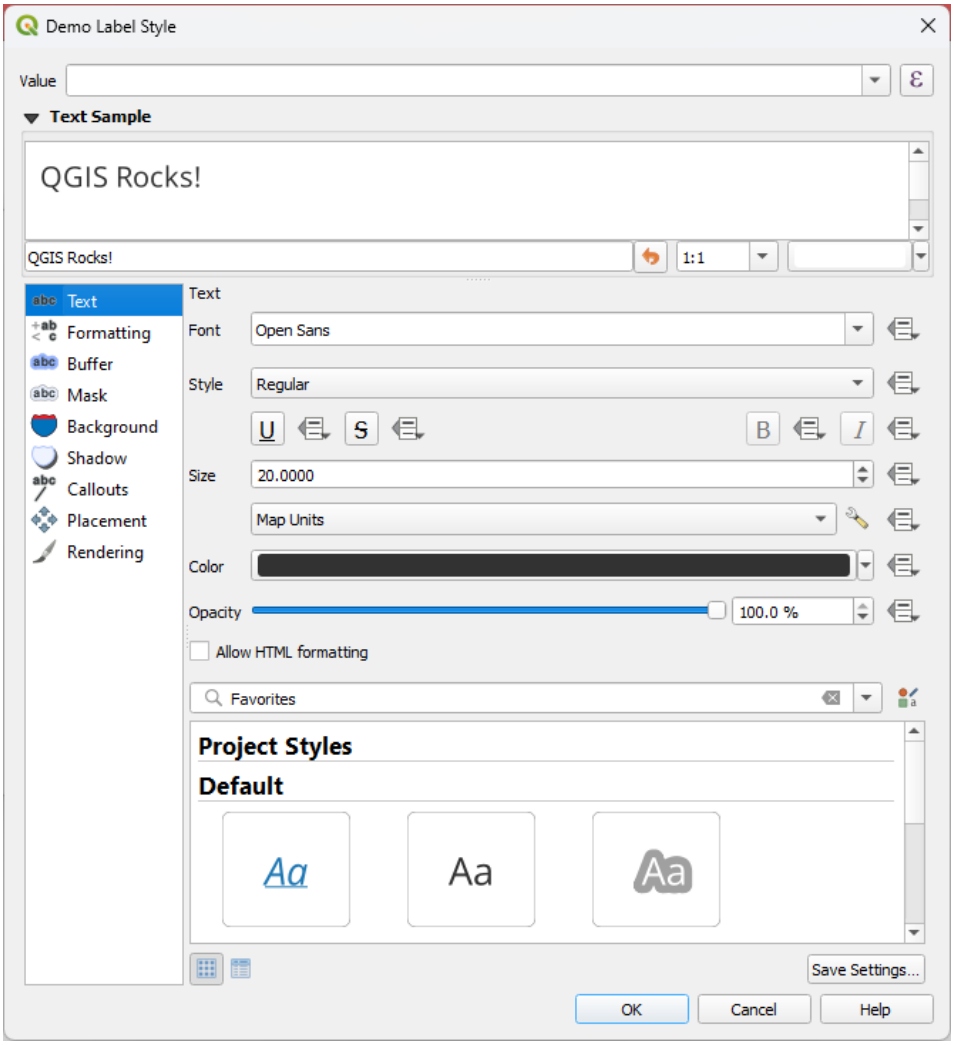


Fig. 14.23: Labels settings - Text tab

In the abc *Text* tab, you can set:

- the *Font*, from the ones available on your machine
- the *Style*: along with the common styles of the font, you can set whether the text should be underlined or striked through
- the *Size* in any *supported unit*
- the *Color*
- the *Opacity*
- and *Allow HTML Formatting*: The HTML formatting option enables the proper rendering of some HTML tags to customize the label. The supported HTML tags are:
  - Color, applicable to text, underline, strikethrough, and overline
  - Font properties (font family, font size, bold and italic)
  - Superscript and subscript components in text, where the text will be vertically $^{\text{super}}$ or $_{\text{sub}}$ aligned and automatically sized to 2/3 of the parent font size. You can also set a fixed font size for the superscript/subscript by including css rules, e.g.:

```
<sup style="font-size:33pt">my superscript text</sup>
```

The CSS formatting rules `vertical-align: super` or `vertical-align: sub` are also available in any other HTML element (annotation, layout label or HTML items, …).

In order to use the HTML formatting, you need to provide the HTML code in the *Value* field. The expression is parsed and any supported HTML tag overrides its corresponding setting in the labels properties. They also combine well with other background, shadow, buffer… properties of labels.

Below an example of a HTML-based expression and rendering (applies different colors and underline to the same label):

```
format(
  '<span style="color:blue">%1</span> ( <span style="color:red"><u>%2 ft</u></
↪span> )',
  title( lower( "Name" ) ),
  round($length)
)
```

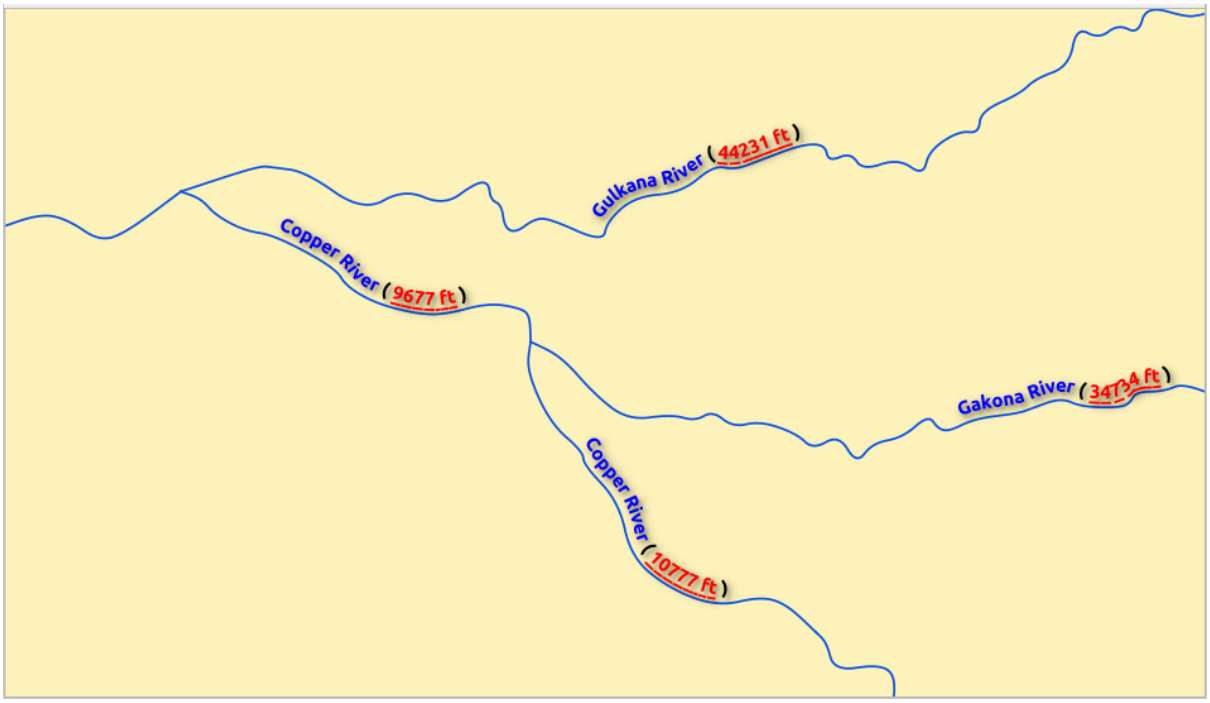


Fig. 14.24: Labeling with HTML formatting enabled

At the bottom of the tab, a widget shows a filterable list of compatible items stored in your *style manager database*. This allows you to easily configure the current text format or label setting based on an existing one, and also save a new item to the style database: Press the *Save format…* or *Save settings…* button and provide a name and tag(s).

Notă: When configuring a *Label Settings* item, text format items are also available in this widget. Select one to quickly overwrite the current *textual properties* of the label. Likewise, you can create/overwrite a text format from there.

Formatting tab

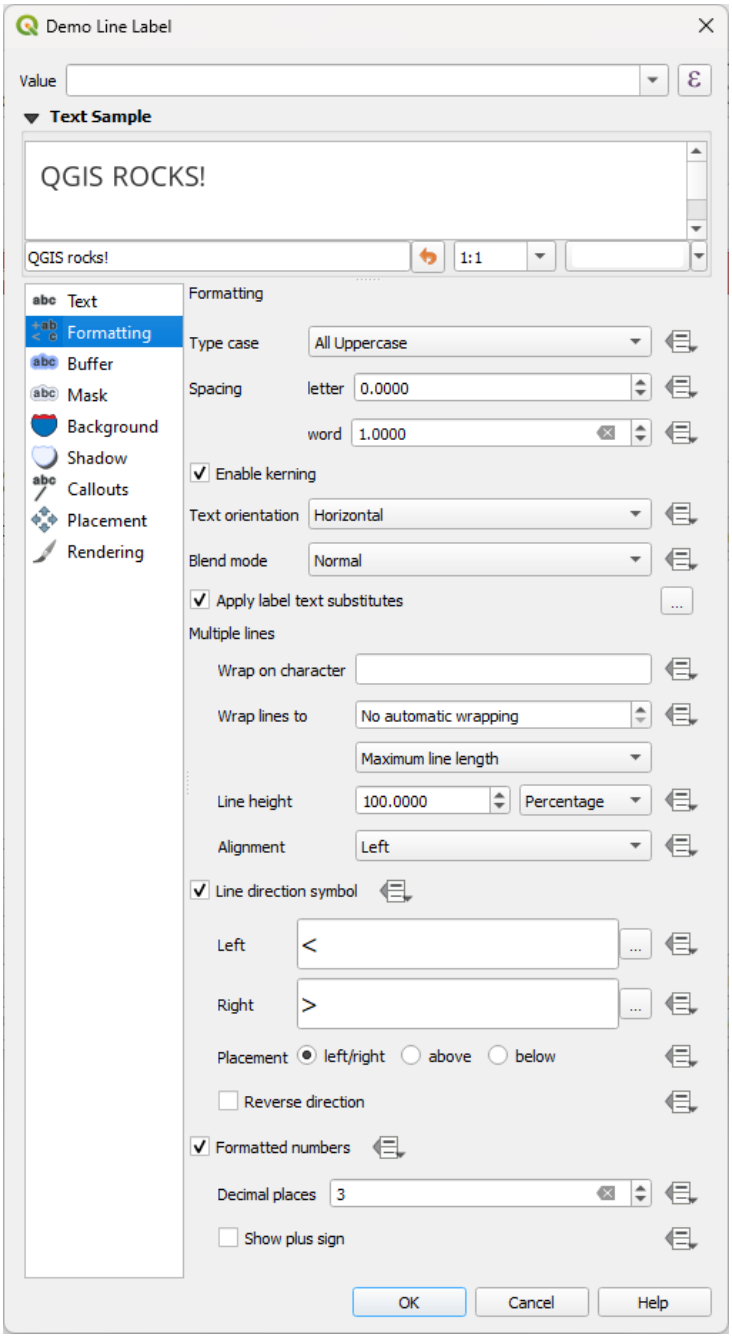


Fig. 14.25: Label settings - Formatting tab

In the *Formatting* tab, you can:

- Use the *Type case* option to change the capitalization style of the text. You have the possibility to render the text as:
  - *No change*
  - *All uppercase*
  - *All lowercase*
  - *Title case*: modifies the first letter of each word into capital, and turns the other letters into lower case if

the original text is using a single type case. In case of mixed type cases in the text, the other letters are left untouched.

  - *Force first letter to capital*: modifies the first letter of each word into capital and leaves the other letters in the text untouched.

- Under *Spacing*, change the space between words and between individual letters.
- *Stretch* ratio: allows text to be horizontally stretched or condensed by a factor. Handy for tweaking the widths of fonts to fit a bit of extra text into labels.
- *Enable kerning* of the text font
- Set the *Text orientation* which can be *Horizontal* or *Vertical*. It can also be *Rotation-based* when setting a label (e.g., to properly label line features in *parallel* placement mode).
- Use the *Blend mode* option to determine how your labels will mix with the map features below them (more details at *Modurile de Fuziune*).
- The *Apply label text substitutes* option allows you to specify a list of texts to substitute to texts in feature labels (e.g., abbreviating street types). Replacement texts are used when displaying labels on the map. Users can also export and import lists of substitutes to make reuse and sharing easier.
- Configure *Multiple lines*:
  - Set a character that will force a line break in the text with the *Wrap on character* option
  - Set an ideal line size for auto-wrapping using the *Wrap lines to* option. The size can represent either the *Maximum line length* or the *Minimum line length*.
  - Decide the *Line Height*: values can be set to be in *Millimeters*, *Points*, *Pixels*, *Percentage*, or *Inches*. When line height is set to percentage it is the percentage of the default text line spacing of that font family. Typically 1.2 to 1.5 times the text size.
  - Format the *Alignment*: typical values available are *Left*, *Right*, *Justify* and *Center*.

    When setting point labels properties, the text alignment can also be *Follow label placement*. In that case, the alignment will depend on the final placement of the label relative to the point. E.g., if the label is placed to the left of the point, then the label will be right aligned, while if it is placed to the right, it will be left aligned.

Notă: The *Multiple lines* formatting is not yet supported by curve based *label placement*. The options will then be deactivated.

- For line labels you can include *Line direction symbol* to help determine the line directions, with symbols to use to indicate the *Left* or *Right*. They work particularly well when used with the *curved* or *Parallel* placement options from the *Placement* tab. There are options to set the symbols position, and to *Reverse direction*.
- Use the *Formatted numbers* option to format numeric texts. You can set the number of *Decimal places*. By default, 3 decimal places will be used. Use the *Show plus sign* if you want to show the plus sign for positive numbers.

Buffer tab

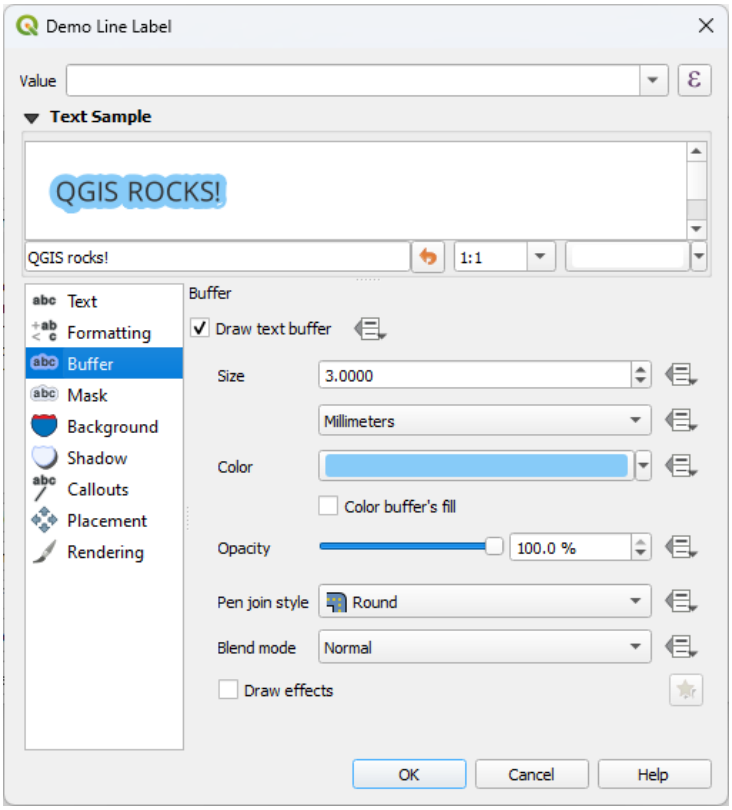


Fig. 14.26: Label settings - Buffer tab

To create a buffer around the label, activate the *Draw text buffer* checkbox in the *Buffer* tab. Then you can:

- Set the buffer's *Size* in any *supported unit*
- Select the buffer's *Color*
- *Color buffer's fill*: The buffer expands from the label's outline, so, if the option is activated, the label's interior is filled. This may be relevant when using partially transparent labels or with non-normal blending modes, which will allow seeing behind the label's text. Unchecking the option (while using totally transparent labels) will allow you to create outlined text labels.
- Define the buffer's *Opacity*
- Apply a *Pen join style*: it can be *Round*, *Miter* or *Bevel*
- Use the *Blend mode* option to determine how your label's buffer will mix with the map components below them (more details at *Modurile de Fuziune*).
- Check *Draw effects* to add advanced *paint effects* for improving text readability, eg through outer glows and blurs.

Background tab

The *Background* tab allows you to configure a shape that stays below each label. To add a background, activate the *Draw Background* checkbox and select the *Shape* type. It can be:

- a regular shape such as *Rectangle*, *Square*, *Circle* or *Ellipse* using full properties of a *fill symbol*
- an *SVG* symbol from a file, a URL or embedded in the project or style database (*more details*)
- or a *Marker Symbol* you can create or select from the *symbol library*.

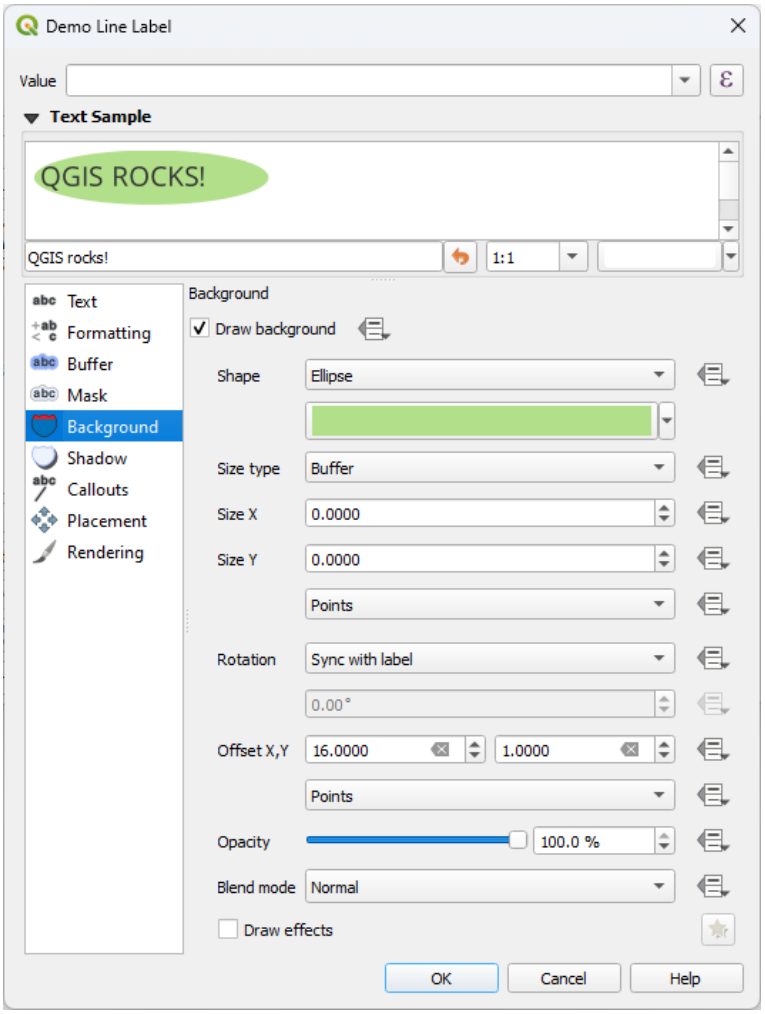


Fig. 14.27: Label settings - Background tab

Depending on the selected shape, you need to configure some of the following properties:

- The *Size type* of the frame, which can be:
  - *Fixed*: using the same size for all the labels, regardless the size of the text
  - or a *Buffer* over the text's bounding box
- The *Size* of the frame in X and Y directions, using any *supported units*
- A *Rotation* of the background, between *Sync with label*, *Offset of label* and *Fixed*. The last two require an angle in degrees.
- An *Offset X,Y* to shift the background item in the X and/or Y directions
- A *Radius X,Y* to round the corners of the background shape (applies to rectangle and square shapes only)

- An *Opacity* of the background
- A *Blend mode* to mix the background with the other items in the rendering (see *Modurile de Fuziune*).
- For SVG symbol, you can use its default properties (*Load symbol parameters*) or set a custom *Fill color*, *Stroke color* and *Stroke width*.
- *Draw effects* to add advanced *paint effects* for improving text readability, eg through outer glows and blurs.

Shadow tab

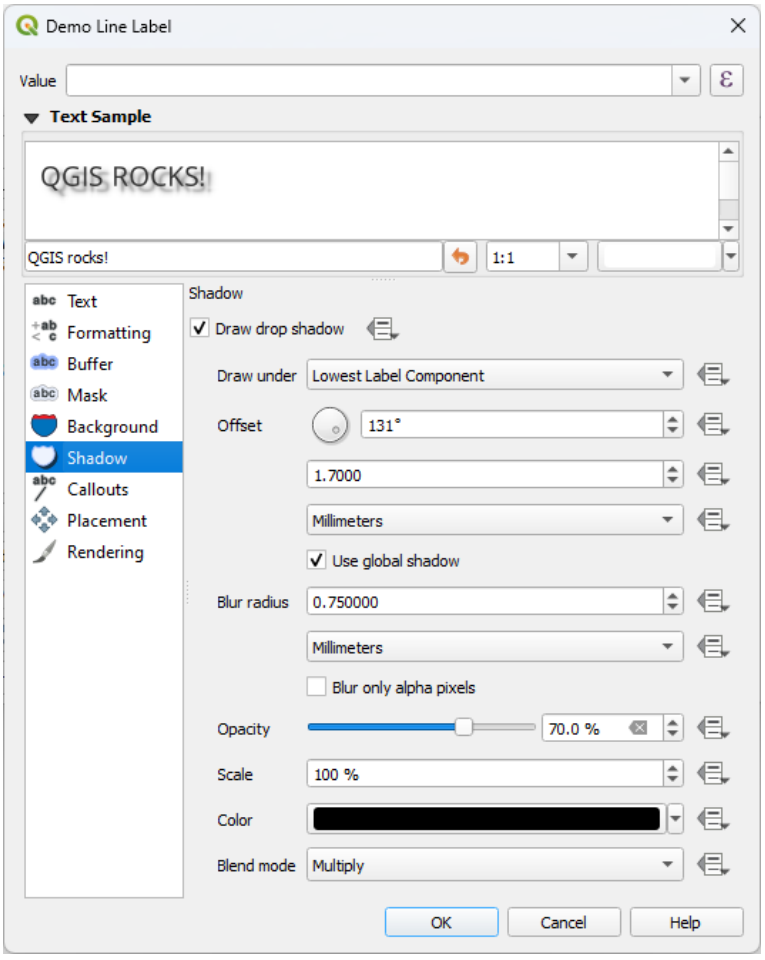


Fig. 14.28: Label settings - Shadow tab

To add a shadow to the text, enable the *Shadow* tab and activate the *Draw drop shadow*. Then you can:

- Indicate the item used to generate the shadow with *Draw under*. It can be the *Lowest label component* or a particular component such as the *Text* itself, the *Buffer* or the *Background*.
- Set the shadow's *Offset* from the item being shadowded, ie:
  - The angle: clockwise, it depends on the underlying item orientation
  - The distance of offset from the item being shadowded
  - The units of the offset

  If you tick the *Use global shadow* checkbox, then the zero point of the angle is always oriented to the north and doesn't depend on the orientation of the label's item.

- Influence the appearance of the shadow with the *Blur radius*. The higher the number, the softer the shadows, in the units of your choice.
- Define the shadow's *Opacity*
- Rescale the shadow's size using the *Scale* factor
- Choose the shadow's *Color*
- Use the *Blend mode* option to determine how your label's shadow will mix with the map components below them (more details at *Modurile de Fuziune*).

14.3.2 Configuring interaction with labels

Other than the text formatting settings exposed above, you can also set how labels interact with each others or with the features.

Mask tab

The *Mask* tab allows you to define a mask area around the labels. This feature is very useful when you have overlapping symbols and labels with similar colors, and you want to make the labels visible. A label mask prevents specified features from drawing within the boundary set for the mask. For example, you could set a label mask so that a specified layer does not draw within 2mm of the label, but allow features from another layer to still show. Label masks are similar to label buffers in that they allow control of the legibility of labels that cover other features. The label buffer draws on top of any underlying features, while the label mask selectively stops other layers from drawing.

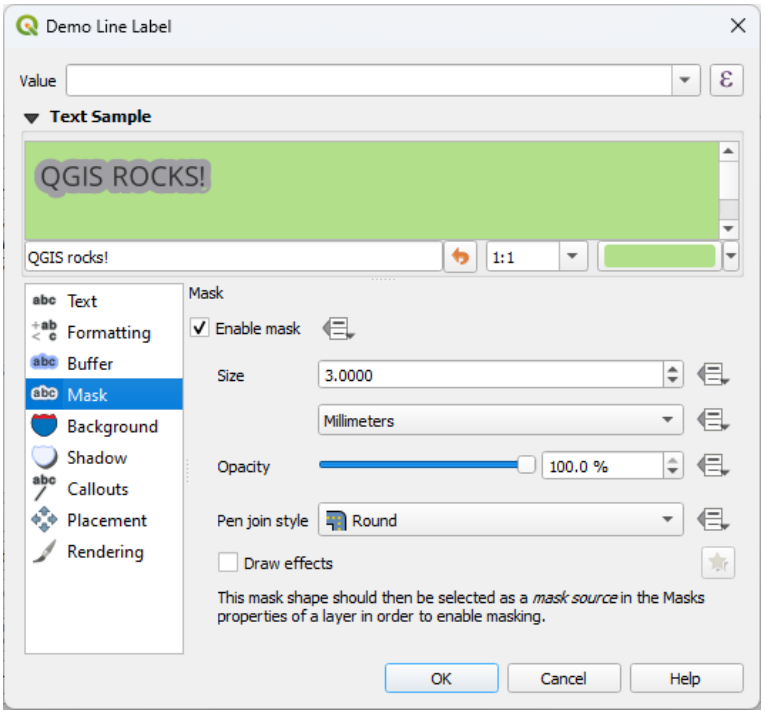


Fig. 14.29: Labels settings - Mask tab (with the text sample showing a green background representing another layer being excluded)

To create masking effects on labels:

1. Activate the *Enable mask* checkbox in the tab.
2. Then you can set:

- the mask's *Size* in the *supported units*
- the *Opacity* of the mask area around the label
- a *Pen Join Style*
- *paint effects* through the *Draw effects* checkbox.

3. Select this mask shape as a mask source in the overlapping layer properties *Mask* tab (see *Masks Properties*).

Callouts tab

A common practice when placing labels on a crowded map is to use **callouts** - labels which are placed outside (or displaced from) their associated feature are identified with a dynamic line connecting the label and the feature. If one of the two endings (either the label or the feature) is moved, the shape of the connector is recomputed.

Fig. 14.30: Labels with various callouts settings

To add a callout to a label, enable the *Callouts* tab and activate the *Draw callouts*. Then you can:

1. Select the *Style* of connector, one of:
   - *Simple lines*: a straight line, the shortest path
   - *Manhattan style*: a 90° broken line
   - *Curved lines*: a curved line
   - *Balloons*: a speech bubble surrounding the label and pointing to the feature. It can have rounded corners.
2. For a line-based callout:
   1. Select the *Line style* with full capabilities of a *line symbol* including layer effects, and data-defined settings
   2. If curved, you also define:
      - the percentage of *Curvature* of the connection line
      - and its *Orientation*: starting from the label to the feature, it can be *Clockwise* or *Counter-clockwise*, or *Automatic* (determining an optimal orientation for each label)
   3. Set the *Minimum length* of callout lines

4. Check whether to *Draw lines to all feature parts* from the feature's label
5. Set the *Label anchor point*: controls where the connector line should join to the label text. Available options:
   - *Closest point*
   - *Centroid*
   - Fixed position at the edge (*Top left*, *Top center*, *Top right*, *Left middle*, *Right middle*, *Bottom left*, *Bottom center* and *Bottom right*).
6. Set the *Offset from label area* option: controls the distance from the label anchor point (where the callout line ends). This avoids drawing lines right up against the text.

3. For a balloon callout, you'd need to set:
   - the *Fill style* with full capabilities of a *fill symbol* including layer effects, and data-defined settings
   - the *Corner radius* of the speech bubble
   - the *Wedge width*: how large the bubble speech connection with feature's pointer should be
   - the *Margins* around the label's text
4. Set the *Offset from feature* option: controls the distance from the feature (or its anchor point if a polygon) where callout lines end. Eg, this avoids drawing lines right up against the edges of the features.
5. Set the *Feature anchor point* for the (polygon) feature (the end point of the connector line). Available options:
   - *Pole of inaccessibility*
   - *Point on exterior*
   - *Point on surface*
   - *Centroid*
6. Set the *Blend mode*: controls the *blending* of the callout.

Under the *Data defined placement* group, coordinates of the *Origin* (on the label side) and/or *Destination* (on the feature side) points of the callout can be controlled. Callouts can also be controlled manually by using the

Move Label, Diagram or Callout tool in the *Labeling Toolbar*. The start and end points of each callout can be moved this way. The nodes should be highlighted when the mouse pointer is nearby. If needed the `Shift` Key can be held during the movement. This will snap the point in a way that the angle between the two callout points increments by 15 degrees.

Placement tab

Choose the *Placement* tab for configuring label placement and labeling priority. Note that the placement options differ according to the type of vector layer, namely point, line or polygon, and are affected by the global *PAL setting*.

Placement for point layers

Point labels placement modes available are:

- *Cartographic*: point labels are generated with a better visual relationship with the point feature, following ideal cartographic placement rules. Labels can be placed:
  - at a set *Distance* in *supported units*, either from the point feature itself or from the bounds of the symbol used to represent the feature (set in *Distance offset from*). The latter option is especially useful when the symbol size isn't fixed, e.g. if it's set by a data defined size or when using different symbols in a *categorized* renderer.

  - following a *Position priority* that can be customized or set for an individual feature using a data defined list of prioritised positions. This also allows only certain placements to be used, so e.g. for coastal features you can prevent labels being placed over the land.

    By default, cartographic mode placements are prioritised in the following order (respecting the guidelines from Krygier and Wood (2011) and other cartographic textbooks):

    1. dreapta sus
    2. stânga sus
    3. dreapta jos
    4. stânga jos
    5. dreapta mijloc
    6. stânga mijloc
    7. sus, ușor dreapta
    8. jos, ușor stânga.

- *Around Point*: labels are placed in a circle around the feature. equal radius (set in *Distance*) circle around the feature. The placement priority is clockwise from the „top right". The position can be constrained using the data-defined *Quadrant* option.
- *Offset from Point*: labels are placed at an *Offset X,Y* distance from the point feature, in various units, or preferably over the feature. You can use a data-defined *Quadrant* to constrain the placement and can assign a *Rotation* to the label.

Placement for line layers

Label modes for line layers include:

- *Parallel*: draws the label parallel to a generalised line representing the feature, with preference for placement over straighter portions of the line. You can define:
  - *Allowed positions*: *Above line*, *On line*, *Below line* and *Line orientation dependent position* (placing the label at the left or the right of the line). It's possible to select several options at once. In that case, QGIS will look for the optimal label position.
  - *Distance* between the label and the line
- *Curved*: draws the label following the curvature of the line feature. In addition to the parameters available with the *Parallel* mode, you can set the *Maximum angle between curved characters*, either inside or outside.
- *Horizontal*: draws labels horizontally along the length of the line feature.

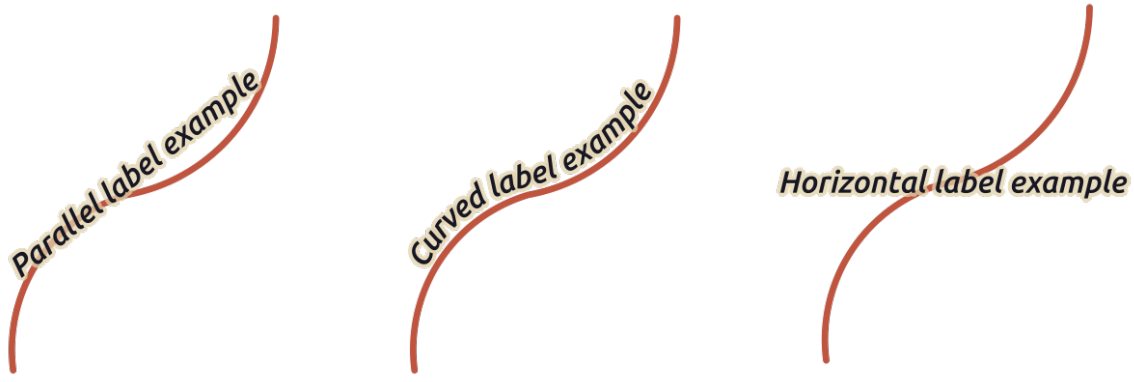


Fig. 14.31: Exemple de poziționare a etichetelor pentru linii

Next to placement modes, you can set:

- *Repeating Labels Distance* to display multiple times the label over the length of the feature. The distance can be in `Millimeters`, `Points`, `Pixels`, `Meters at scale`, `Map Units` and `Inches`.
- A *Label Overrun Distance* (not available for horizontal mode): specifies the maximal allowable distance a label may run past the end (or start) of line features. Increasing this value can allow for labels to be shown for shorter line features.
- *Label Anchoring*: controls the placement of the labels along the line feature they refer to. Click on *Settings …* to choose:
  - the position along the line (as a ratio) which labels will be placed close to. It can be data-defined and possible values are:
    - *Centrul Liniei*
    - *Începutul Liniei*
    - *Finalul Liniei*
    - sau *Personalizare…*
  - *Clipping*: Determines how the label placement on a line is calculated. By default only the visible extent of the line is used but the whole extent can be used to have more consistent results.
  - *Anchor text*: controls which part of the text (start, center or end) will line up with the anchor point. Using *Automatic* anchoring means that:
    - For labels anchored near the start of the line (0-25%), the anchor placement will be the **start** of the label text
    - For labels anchored near the end of the line (75-100%), the anchor placement will be the **end** of the label text
    - For labels anchored near the center of the line (25-75%), the anchor placement will be the **center** of the label text
  - *Placement Behavior*: use *Preferred Placement Hint* to treat the label anchor only as a hint for the label placement. By choosing *Strict*, labels are placed exactly on the label anchor.

Placement for polygon layers

You can choose one of the following modes for placing labels of polygons:

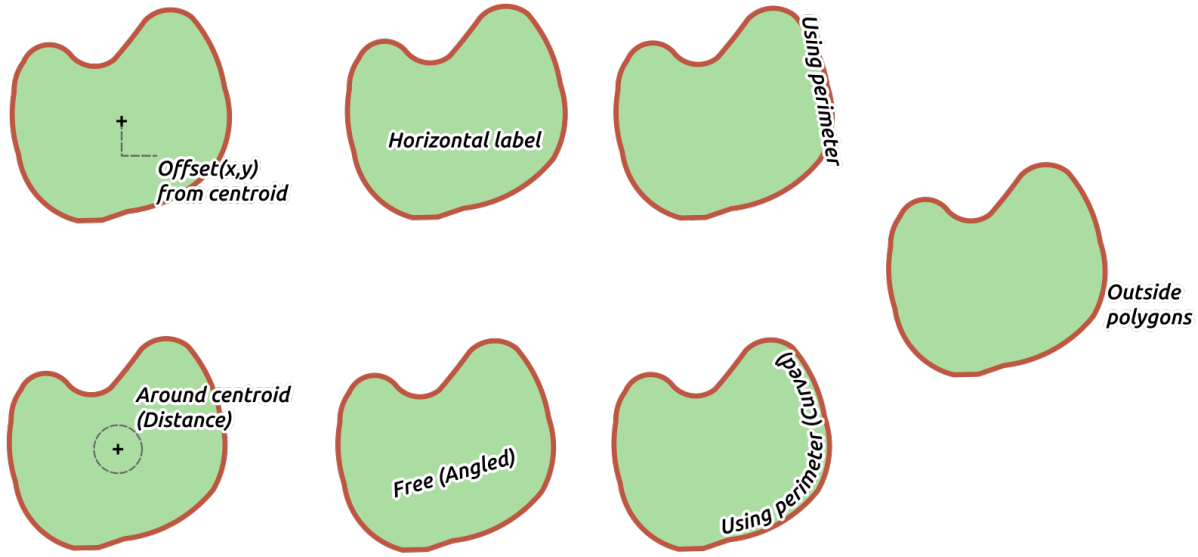


Fig. 14.32: Exemple de poziționare a etichetelor pentru poligoane

- *Offset from Centroid*: labels are placed over the feature centroid or at a fixed *Offset X,Y* distance (in *supported units*) from the centroid. The reference centroid can be determined based on the part of the polygon rendered in the map canvas (*visible polygon*) or the *whole polygon*, no matter if you can see it. You can also:
  - force the centroid point to lay inside their polygon
  - place the label within a specific quadrant
  - assign a rotation
  - *Allow placing labels outside of polygons* when it is not possible to place them inside the polygon. Thanks to data-defined properties, this makes possible to either allow outside labels, prevent outside labels, or force outside labels on a feature-by-feature basis.
- *Around Centroid*: places the label within a preset distance around the centroid, with a preference for the placement directly over the centroid. Again, you can define whether the centroid is the one of the *visible polygon* or the *whole polygon*, and whether to force the centroid point inside the polygon.
- *Horizontal*: places at the best position a horizontal label inside the polygon. The preferred placement is further from the edges of the polygon. It's possible to *Allow placing labels outside of polygons*.
- *Free (Angled)*: places at the best position a rotated label inside the polygon. The rotation respects the polygon's orientation and the preferred placement is further from the edges of the polygon. It's possible to *Allow placing labels outside of polygons*.
- *Using Perimeter*: draws the label parallel to a generalised line representing the polygon boundary, with preference for straighter portions of the perimeter. You can define:
  - *Allowed positions*: *Above line*, *On line*, *Below line* and *Line orientation dependent position* (placing the label at the left or the right of the polygon's boundary). It's possible to select several options at once. In that case, QGIS will look for the optimal label position.
  - *Distance* between the label and the polygon's outline
  - the *Repeating Labels Distance* to display multiple times the label over the length of the perimeter.
- *Using Perimeter (Curved)*: draws the label following the curvature of the polygon's boundary. In addition to the parameters available with the *Using Perimeter* mode, you can set the *Maximum angle between curved characters* *polygon*, either inside or outside.

- *Outside Polygons*: always places labels outside the polygons, at a set *Distance*

Common placement settings

Some label placement settings are available for all layer geometry types:

Geometry Generator

The *Geometry Generator* section allows a user to alter the underlying geometry used to place and render the label, by using *expressions*. This can be useful to perform displacement of the geometry dynamically or to convert it to another geometry (type).

In order to use the geometry generator:

1. Check the *Geometry generator* option
2. Enter the expression generating the geometry to rely on
3. If relevant, select the geometry type of the expression output: the label geometry-based settings such as placement or rendering are updated to match the new geometry type capabilities.

Some use cases include:

- Use a geometry which is saved in another field „label_position”
- Use the *generated geometry* from the symbology also for labeling
- Use the @map_scale variable to calculate distances / sizes be zoom level independent.
- Combined with the curved placement mode, creates a circular label around a point feature:

```
exterior_ring(make_circle($geometry, 20))
```

- Add a label at the start and the end of a line feature:

```
collect_geometries( start_point($geometry), end_point($geometry) )
```

- Rely on a smoothed line of a river to get more room for label placement:

```
smooth( $geometry, iterations:=30, offset:=0.25, min_length:=10 )
```

Data Defined

The *Data Defined* group provides direct control on labels placement, on a feature-by-feature basis. It relies on their attributes or an expression to set:

- the *X* and *Y* coordinate
- the text alignment over the custom position set above:
  - *Horizontal*: it can be **Left**, **Center** or **Right**
  - the text *Vertical*: it can be **Bottom**, **Base**, **Half**, **Cap** or **Top**
- the text *Rotation*. Different units can be defined for the labeling rotation (e.g. `degrees`, `minutes of arc`, `turns`). Check the *Preserve data rotation values* entry if you want to keep the rotation value in the associated field and apply it to the label, whether the label is pinned or not. If unchecked, unpinning the label rotation is reset and its value cleared from the attribute table.

Notă: Data-defined rotation with polygon features is currently supported only with the *Around centroid* placement mode.

Notă: Expressions can not be used in combination with the labels map tools (ie the *Rotate label* and *Move label* tools) to *data-define* labels placement. The widget will be reset to the corresponding *auxiliary storage field*.

Prioritate

In the *Priority* section you can define the placement priority rank of each label, ie if there are different diagrams or labels candidates for the same location, the item with the higher priority will be displayed and the others could be left out.

The priority rank is also used to evaluate whether a label could be omitted due to a greater weighted *obstacle feature*.

Obstacole

In some contexts (eg, high density labels, overlapping features…), the labels placement can result in labels being placed over unrelated features.

An obstacle is a feature over which QGIS avoids placing other features» labels or diagrams. This can be controlled from the *Obstacles* section:

1. Activate the *Features act as obstacles* option to decide that features of the layer should act as obstacles for any label and diagram (including items from other features in the same layer).

   Instead of the whole layer, you can select a subset of features to use as obstacles, using the Data-defined override control next to the option.

2. Use the *Settings* button to tweak the obstacle's weighting.

   - For every potential obstacle feature you can assign an *Obstacle weight*: any *label* or *diagram* whose placement priority rank is greater than this value can be placed over. Labels or diagrams with lower rank will be omitted if no other placement is possible.

     This weighting can also be data-defined, so that within the same layer, certain features are more likely to be covered than others.

   - For polygon layers, you can choose the kind of obstacle the feature is:

     – **over the feature's interior**: avoids placing labels over the interior of the polygon (prefers placing labels totally outside or just slightly inside the polygon)

     – or **over the feature's boundary**: avoids placing labels over the boundary of the polygon (prefers placing labels outside or completely inside the polygon). This can be useful for layers where the features cover the whole area (administrative units, categorical coverages, …). In this case, it is impossible to avoid placing labels within these features, and it looks much better when placing them over the boundaries between features is avoided.

Rendering tab

In the *Rendering* tab, you can tune when the labels can be rendered and their interaction with other labels and features.

Label options

Under *Label options*:

- You find the *scale-based* and the *Pixel size-based* visibility settings.
- The *Label z-index* determines the order in which labels are rendered, as well in relation with other feature labels in the layer (using data-defined override expression), as with labels from other layers. Labels with a higher z-index are rendered on top of labels (from any layer) with lower z-index.

  Additionally, the logic has been tweaked so that if two labels have matching z-indexes, then:

  – if they are from the same layer, the smaller label will be drawn above the larger label

  – if they are from different layers, the labels will be drawn in the same order as their layers themselves (ie respecting the order set in the map legend).

  Notă: This setting doesn't make labels to be drawn below the features from other layers, it just controls the order in which labels are drawn on top of all the layers» features.

- *Allow inferior fallback placements*: By default QGIS tries to render labels at their best placement, following your settings. Check this mode to allow features to fallback to worse placement options when there's no other choice, e.g. when a line is too short to fit a curved label text then the label may be placed horizontally just over the feature's center point.
- With data-defined expressions in *Show label* and *Always Show* you can fine tune which labels should be rendered.
- Allow to *Show upside-down labels*: alternatives are **Never**, **when rotation defined** or **always**.
- The *Overlapping labels* group allows you to control whether overlapping labels are permitted for features in the layer and how each of them should be handled:

  – *Never overlap*: never ever place overlapping labels for the layer, even if it means some labels will be missing

  – *Allow overlaps if required*: if the label can't otherwise be placed, draw an overlapping label. This mode will cause the label to be moved to a less ideal placement if possible, e.g. moving the label further from the center of a line or polygon, IF doing so will avoid overlapping labels. But if there's no other positions possible, then draw the label overlapping.

  – *Allow overlaps without penalty*: It doesn't matter at all if the label overlaps other labels or obstacles, that's fine to do and the best placement (e.g most central placement) should always be used even if an alternate further placement is possible which avoids overlaps entirely.

  Allowing both overlapping labels and fallback placements options will guarantee that all features in the layer are labeled… not necessarily at their best rendering!

Feature options

Under *Feature options*:

- You can choose to *Label every part of a multi-part features* and *Limit number of features to be labeled to*.
- Both line and polygon layers offer the option to set a minimum size for the features to be labeled, using *Suppress labeling of features smaller than*.
- For polygon features, you can also filter the labels to show according to whether they completely fit within their feature or not.
- For line features, you can choose to *Merge connected lines to avoid duplicate labels*, rendering a quite airy map in conjunction with the *Distance* or *Repeat* options in the *Placement* tab.

14.4 Creating 3D Symbols

The *Style Manager* helps you create and store 3D symbols for every geometry type to render in the *3D map view*.

As of the other items, enable the *3D Symbols* tab and expand the button menu to create:

- *3D point symbols*
- *3D line symbols*
- *3D polygon symbols*

14.4.1 Point Layers

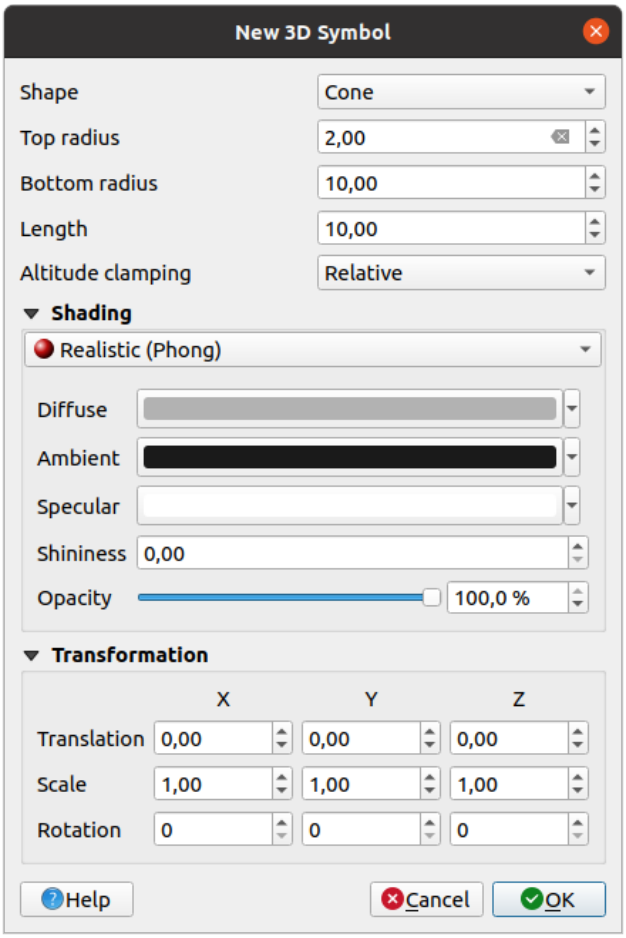


Fig. 14.33: Properties of a 3D point symbol

- You can define different types of 3D *Shape* to use for point symbols. They are mainly defined by their dimensions whose unit refers to the CRS of the project. Available types are:
  - *Sphere* defined by a *Radius*
  - *Cylinder* defined by a *Radius* and *Length*
  - *Cube* defined by a *Size*
  - *Cone* defined by a *Top radius*, a *Bottom radius* and a *Length*
  - *Plane* defined by a *Size*

  - *Torus* defined by a *Radius* and a *Minor radius*
  - *3D Model*, using a 3D model file: supported formats include wavefront `.obj`, `.glTF` and `.fbx`. Models can be a file on disk, a remote URL or *embedded in the project*. Community-created models are shared on the QGIS Hub at https://plugins.qgis.org/wavefronts.
  - *Billboard*, defined by the *Billboard height* and the *Billboard symbol* (usually based on a *marker symbol*). The symbol will have a stable size. Convenient for visualizing 3D point clouds Shapes.
- The *Altitude clamping* can be set to *Absolute*, *Relative* or *Terrain*. The *Absolute* setting can be used when height values of the 3d vectors are provided as absolute measures from 0. *Relative* and *Terrain* add given elevation values to the underlying terrain elevation.
- The *shading* properties can be defined.
- Under the *Transformations* frame, you can apply affine transformation to the symbol:
  - *Translation* to move objects in x, y and z axis.
  - *Scale* to resize the 3D shapes
  - *Rotation* around the x-, y- and z-axis.

14.4.2 Line layers

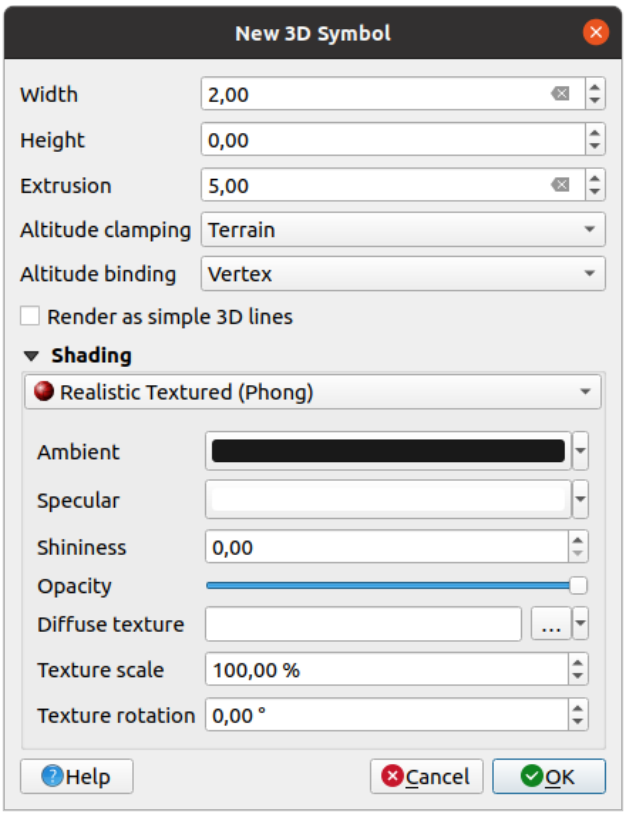


Fig. 14.34: Properties of a 3D line symbol

- Beneath the *Width* and *Height* settings you can define the *Extrusion* of the vector lines. If the lines do not have z-values, you can define the 3d volumes with this setting.
- With the *Altitude clamping* you define the position of the 3D lines relative to the underlying terrain surface, if you have included raster elevation data or other 3D vectors.

- The *Altitude binding* defines how the feature is clamped to the terrain. Either every *Vertex* of the feature will be clamped to the terrain or this will be done by the *Centroid*.
- It is possible to *Render as simple 3D lines*.
- The *shading* properties can be defined.

14.4.3 Polygon Layers

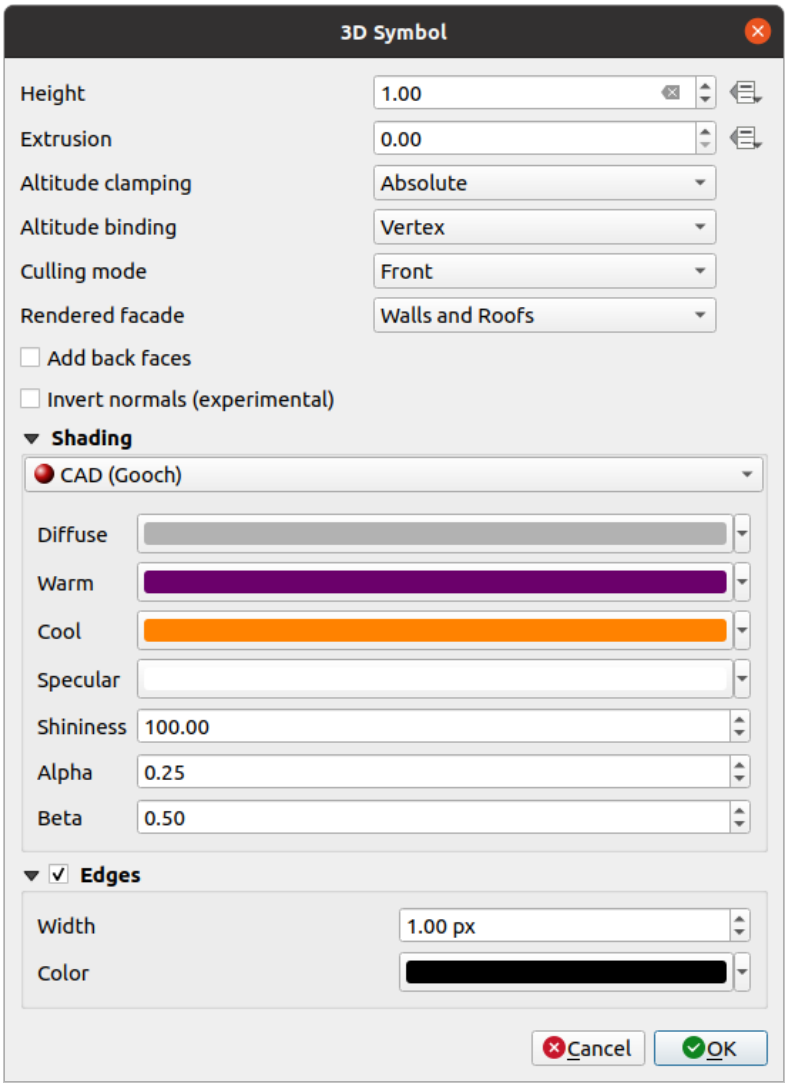


Fig. 14.35: Properties of a 3D polygon symbol

- As for the other ones, *Height* can be defined in CRS units. You can also use the button to overwrite the value with a custom expression, a variable or an entry of the attribute table
- Again, *Extrusion* is possible for missing z-values. Also for the extrusion you can use the button in order to use the values of the vector layer and have different results for each polygon:

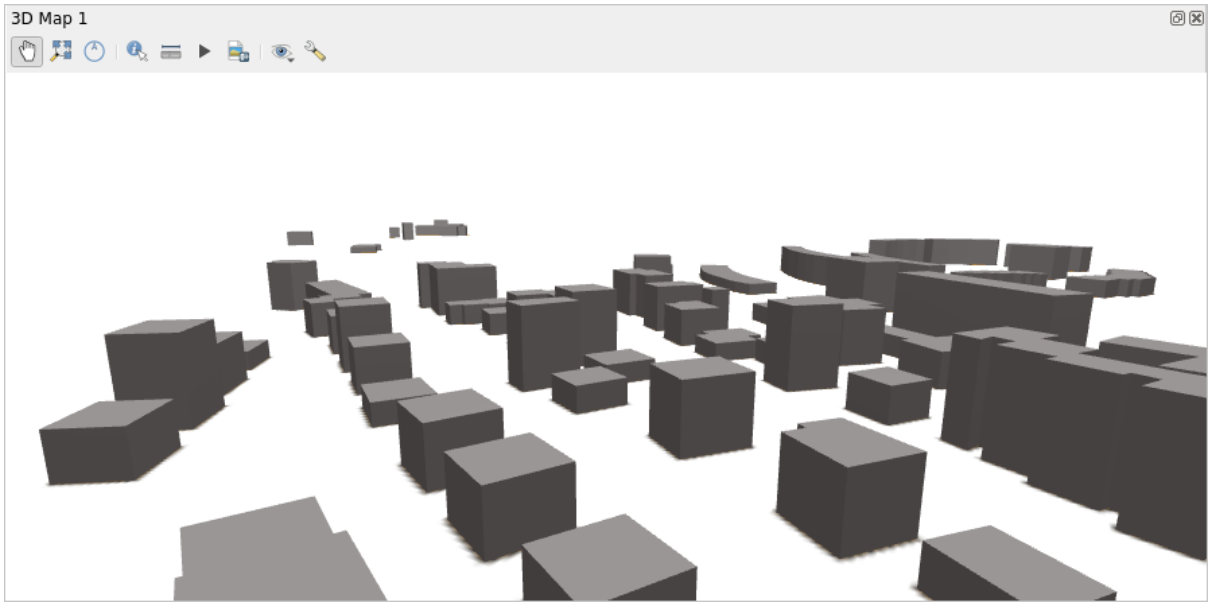


Fig. 14.36: Data Defined Extrusion

- The *Altitude clamping*, *Altitude binding* can be defined as explained above.
- The *Culling mode* to apply to the symbol; it can be:
  - *No Culling*: this can help to avoid seemingly missing surfaces when polygonZ/multipatch data do not have consistent ordering of vertices (e.g. all clock-wise or counter clock-wise)
  - *Front*
  - or *Back*
- The *Rendered facade* determines the faces to display. Possible values are *No facades*, *Walls*, *Roofs*, or *Walls and roofs*
- *Add back faces*: for each triangle, creates both front and back face with correct normals - at the expense of increased number of vertex data. This option can be used to fix shading issues (e.g., due to data with inconsistent order of vertices).
- *Invert normals (experimental)*: can be useful for fixing clockwise/counter-clockwise face vertex orders
- The *shading* properties can be defined.
- Display of the *Edges* of the symbols can be enabled and assigned a *Width* and *Color*.

Sugestie: Combination for best rendering of 3D data

Culling mode, *Add back faces* and *Invert normals* are all meant to fix the look of 3D data if it does not look right. Typically when loading some data, it is best to first try `culling mode=back` and `add back faces=disabled` - it is the most efficient. If the rendering does not look correct, try `add back faces=enabled` and keep `culling mode=no culling`. Other combinations are more advanced and useful only in some scenarios based on how mixed up is the input dataset.

14.4.4 Shading the texture

Shading helps you reveal 3d details of objects which may otherwise be hidden due to the scene's lighting. Ultimately, it's an easier material to work with as you don't need to worry about setting up appropriate scene lighting in order to visualise features.

Various techniques of shading are used in QGIS and their availability depends on the geometry type of the symbol:

- *Realistic (Phong)*: describes the way a surface reflects light as a combination of the *Diffuse* reflection of rough surfaces with the *Specular* reflection of shiny surfaces (*Shininess*). It also includes an *Ambient* option to account for the small amount of light that is scattered about the entire scene. Use the *Opacity* slider to render semi-transparent objects in 3D. Read more at https://en.wikipedia.org/wiki/Phong_reflection_model#Description
- *Realistic Textured (Phong)*: same as the *Realistic (Phong)* except that an image is used as *Diffuse Texture*. The image can be a file on disk, a remote URL or *embedded in the project*. The *Texture scale* and *Texture rotation* are required. Use the *Opacity* slider to render semi-transparent objects in 3D.
- *CAD (Gooch)*: this technique allows shading to occur only in mid-tones so that edge lines and highlights remain visually prominent. Along with the *Diffuse*, *Specular*, *Shininess* options, you need to provide a *Warm* color (for surface facing toward the light) and a *Cool* color (for the ones facing away). Also, the relative contributions to the cool and warm colors by the diffuse color are controlled by *Alpha* and *Beta* properties respectively. See also https://en.wikipedia.org/wiki/Gooch_shading
- *Embedded Textures* with 3D models shape

14.4.5 Application example

To go through the settings explained above you can have a look at https://app.merginmaps.com/projects/saber/luxembourg/tree.

CAPITOLUL 15

Gestiunea Surselor de Date

15.1 Deschiderea datelor

Ca parte a unui ecosistem Software cu Sursă Deschisă, QGIS este construit pe diferite biblioteci care, combinate cu furnizorii proprii, oferă posibilități de a citi și de a scrie într-o mulțime de formate:

- Vector data formats include GeoPackage, GML, GeoJSON, GPX, KML, Comma Separated Values, ESRI formats (Shapefile, Geodatabase…), MapInfo and MicroStation file formats, AutoCAD DWG/DXF, GRASS and many more… Read the complete list of supported vector formats.
- Raster data formats include GeoTIFF, JPEG, ASCII Gridded XYZ, MBTiles, R or Idrisi rasters, GDAL Virtual, SRTM, Sentinel Data, ERDAS IMAGINE, ArcInfo Binary Grid, ArcInfo ASCII Grid, and many more… Read the complete list of supported raster formats.
- Database formats include PostgreSQL/PostGIS, SQLite/SpatiaLite, Oracle, MS SQL Server, SAP HANA, MySQL…
- Web map and data services (WM(T)S, WFS, WCS, CSW, XYZ tiles, ArcGIS services, …) are also handled by QGIS providers. See *Working with OGC / ISO protocols* for more information about some of these.
- You can read supported files from archived folders and use QGIS native formats such as QML files (*QML - Formatul Fișierelor de Stilizare*) and virtual and memory layers.

More than 80 vector and 140 raster formats are supported by GDAL and QGIS native providers.

Notă: Not all of the listed formats may work in QGIS for various reasons. For example, some require external proprietary libraries, or the GDAL/OGR installation of your OS may not have been built to support the format you want to use. To see the list of available formats, run the command line `ogrinfo --formats` (for vector) and `gdalinfo --formats` (for raster), or check the *Settings* ► *Options* ► *GDAL* menu in QGIS.

In QGIS, depending on the data format, there are different tools to open a dataset, mainly available in the *Layer* ► *Add Layer* ► menu or from the *Manage Layers* toolbar (enabled through *View* ► *Toolbars* menu). However, all these tools point to a unique dialog, the *Data Source Manager* dialog, that you can open with the Open Data Source Manager button, available on the *Data Source Manager Toolbar*, or by pressing `Ctrl+L`. The *Data Source Manager* dialog (Fig. 15.1) offers a unified interface to open file-based data as well as databases or web services supported by QGIS.

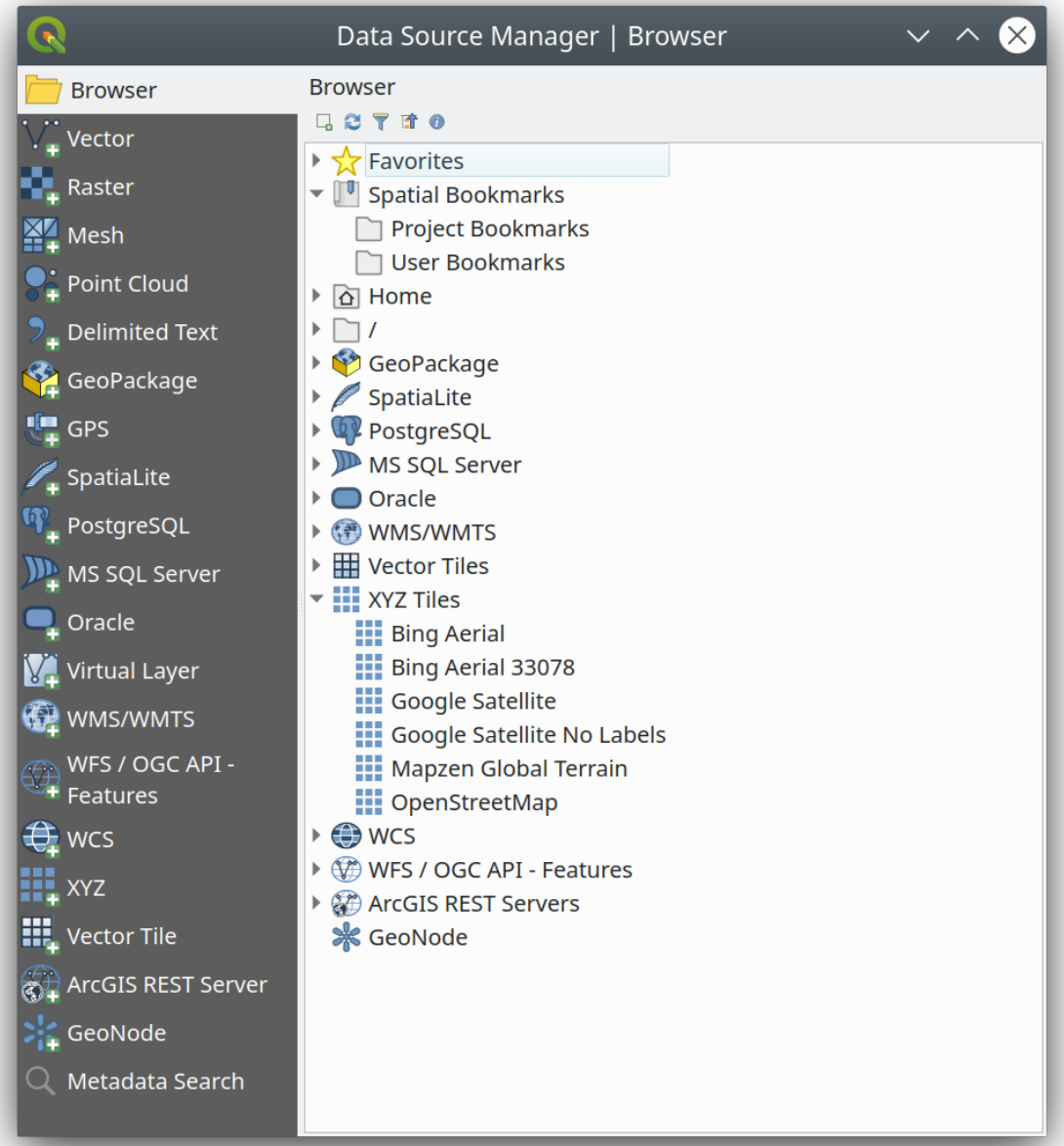


Fig. 15.1: QGIS Data Source Manager dialog

Beside this main entry point, you also have the *DB Manager* plugin that offers advanced capabilities to analyze and manipulate connected databases. More information on DB Manager capabilities can be found in *DB Manager Plugin*.

There are many other tools, native or third-party plugins, that help you open various data formats.

This chapter will describe only the tools provided by default in QGIS for loading data. It will mainly focus on the *Data Source Manager* dialog but more than describing each tab, it will also explore the tools based on the data provider or format specificities.

15.1.1 Panoul Navigatorului

The *Browser* is one of the main ways to quickly and easily add your data to projects. It's available as:

- a *Data Source Manager* tab, enabled pressing the Open Data Source Manager button (`Ctrl+L`);
- as a QGIS panel you can open from the menu *View* ► *Panels* (or *Settings* ► *Panels*) or by pressing `Ctrl+2`.

In both cases, the *Browser* helps you navigate in your file system and manage geodata, regardless the type of layer (raster, vector, table), or the datasource format (plain or compressed files, databases, web services).

Exploring the Interface

At the top of the Browser panel, you find some buttons that help you to:

- Add Selected Layers: you can also add data to the map canvas by selecting **Add selected layer(s)** from the layer's context menu;
- Refresh the browser tree;
- Filter Browser to search for specific data. Enter a search word or wildcard and the browser will filter the tree to only show paths to matching DB tables, filenames or folders – other data or folders won't be displayed. See the Browser Panel(2) example in Fig. 15.2. The comparison can be case-sensitive or not. It can also be set to:
  - *Normal*: show items containing the search text
  - *Wildcard(s)*: fine tune the search using the `?` and/or `*` characters to specify the position of the search text
  - *Regular expression*
- Collapse All the whole tree;
- Enable/disable properties widget: when toggled on, a new widget is added at the bottom of the panel showing, if applicable, metadata for the selected item.

The entries in the *Browser* panel are organised hierarchically, and there are several top level entries:

1. *Favorites* where you can place shortcuts to often used locations
2. *Spatial Bookmarks* where you can store often used map extents (see *Bookmarking extents on the map*)
3. *Project Home*: for a quick access to the folder in which (most of) the data related to your project are stored. The default value is the directory where your project file resides.
4. *Home* directory in the file system and the filesystem root directory.
5. Connected local or network drives
6. Then comes a number of container / database types and service protocols, depending on your platform and underlying libraries:
   - *GeoPackage*
   - *SpatiaLite*
   - *PostgreSQL*
   - *SAP HANA*
   - *MS SQL Server*
   - *Oracle*

- *WMS/WMTS*
- *Vector Tiles*
- *XYZ Tiles*
- *WCS*
- *WFS/OGC API-Features*
- *ArcGIS REST Server*

Interacting with the Browser items

The browser supports drag and drop within the browser, from the browser to the canvas and *Layers* panel, and from the *Layers* panel to layer containers (e.g. GeoPackage) in the browser.

Project file items inside the browser can be expanded, showing the full layer tree (including groups) contained within that project. Project items are treated the same way as any other item in the browser, so they can be dragged and dropped within the browser (for example to copy a layer item to a geopackage file) or added to the current project through drag and drop or double click.

The context menu for an element in the *Browser* panel is opened by right-clicking on it.

For file system directory entries, the context menu offers the following:

- *New* ► to create in the selected entry a:
  - *Directory…*
  - *GeoPackage…*
  - *ShapeFile…*
- *Add as a Favorite*: favorite folders can be renamed (*Rename favorite…*) or removed (*Remove favorite*) any time.
- *Hide from Browser*: hidden folders can be toggled to visible from the *Settings* ► *Options* ► *Data Sources* ► *Hidden browser paths* setting
- *Fast Scan this Directory*
- *Open Directory*
- *Open in Terminal*
- *Proprietăţi…*
- *Directory Properties…*

For leaf entries that can act as layers in the project, the context menu will have supporting entries. For example, for non-database, non-service-based vector, raster and mesh data sources:

- *Export Layer* ► *To File…*
- *Add Layer to Project*
- *Layer Properties*
- *Manage* ► *Rename „<name of file>”…* or *Delete „<name of file>”…*
- *Show in Files*
- *File Properties*

In the *Layer properties* entry, you will find (similar to what you will find in the *vector* and *raster* layer properties once the layers have been added to the project):

- *Metadata* for the layer. Metadata groups: *Information from provider* (if possible, *Path* will be a hyperlink to the source), *Identification*, *Extent*, *Access*, *Fields* (for vector layers), *Bands* (for raster layers), *Contacts*, *Links* (for vector layers), *References* (for raster layers), *History*.
- A *Preview* panel
- The attribute table for vector sources (in the *Attributes* panel).

To add a layer to the project using the *Browser*:

1. Enable the *Browser* as described above. A browser tree with your file system, databases and web services is displayed. You may need to connect databases and web services before they appear (see dedicated sections).
2. Find the layer in the list.
3. Use the context menu, double-click its name, or drag-and-drop it into the *map canvas*. Your layer is now added to the *Layers panel* and can be viewed on the map canvas.

Sfat: Open a QGIS project directly from the browser

You can also open a QGIS project directly from the Browser panel by double-clicking its name or by drag-and-drop into the map canvas.

Once a file is loaded, you can zoom around it using the map navigation tools. To change the style of a layer, open the *Layer Properties* dialog by double-clicking on the layer name or by right-clicking on the name in the legend and choosing *Properties* from the context menu. See section *Symbology Properties* for more information on setting symbology for vector layers.

Right-clicking an item in the browser tree helps you to:

- for a file or a table, display its metadata or open it in your project. Tables can even be renamed, deleted or truncated.
- for a folder, bookmark it into your favourites or hide it from the browser tree. Hidden folders can be managed from the *Settings* ► *Options* ► *Data Sources* tab.
- manage your *spatial bookmarks*: bookmarks can be created, exported and imported as `XML` files.
- create a connection to a database or a web service.
- refresh, rename or delete a schema.

You can also import files into databases or copy tables from one schema/database to another with a simple drag-and-drop. There is a second browser panel available to avoid long scrolling while dragging. Just select the file and drag-and-drop from one panel to the other.

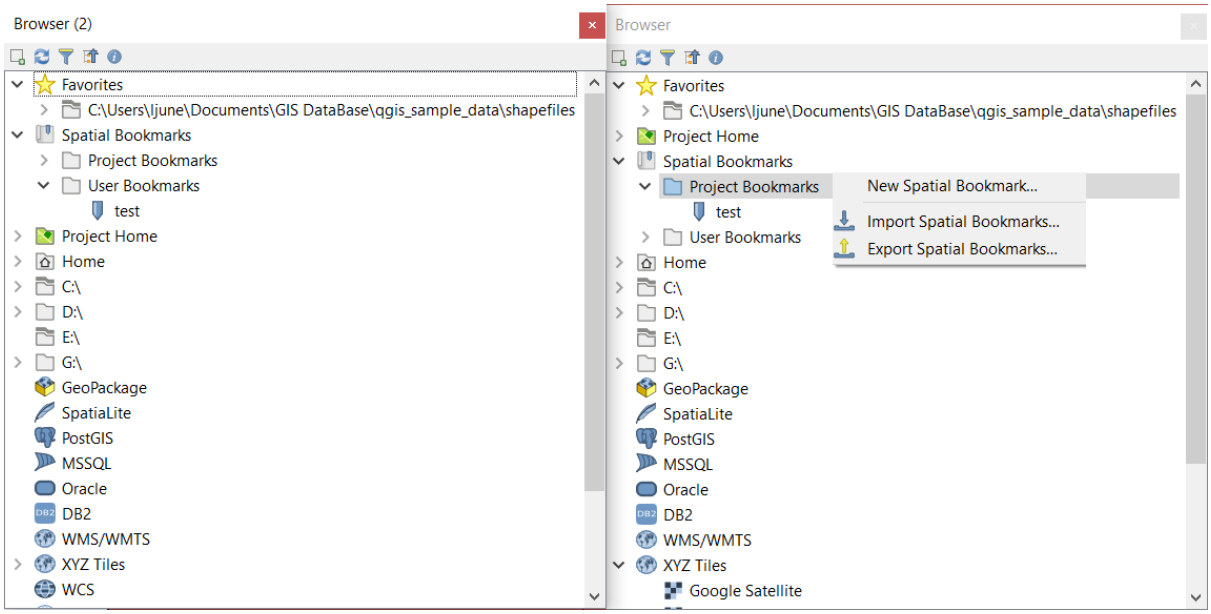


Fig. 15.2: Panourile Navigatorului QGIS, unul lângă altul

Sfat: Add layers to QGIS by simple drag-and-drop from your OS file browser

You can also add file(s) to the project by drag-and-dropping them from your operating system file browser to the *Layers Panel* or the map canvas.

15.1.2 The DB Manager

The *DB Manager* Plugin is another tool for integrating and managing spatial database formats supported by QGIS (PostGIS, SpatiaLite, GeoPackage, Oracle Spatial, MS SQL Server, Virtual layers). It can be activated from the *Plugins* ► *Manage and Install Plugins...* menu.

The DB Manager Plugin provides several features:

- connect to databases and display their structure and contents
- preview tables of databases
- add layers to the map canvas, either by double-clicking or drag-and-drop.
- add layers to a database from the QGIS Browser or from another database
- create SQL queries and add their output to the map canvas
- create *virtual layers*

More information on DB Manager capabilities is found in *DB Manager Plugin*.

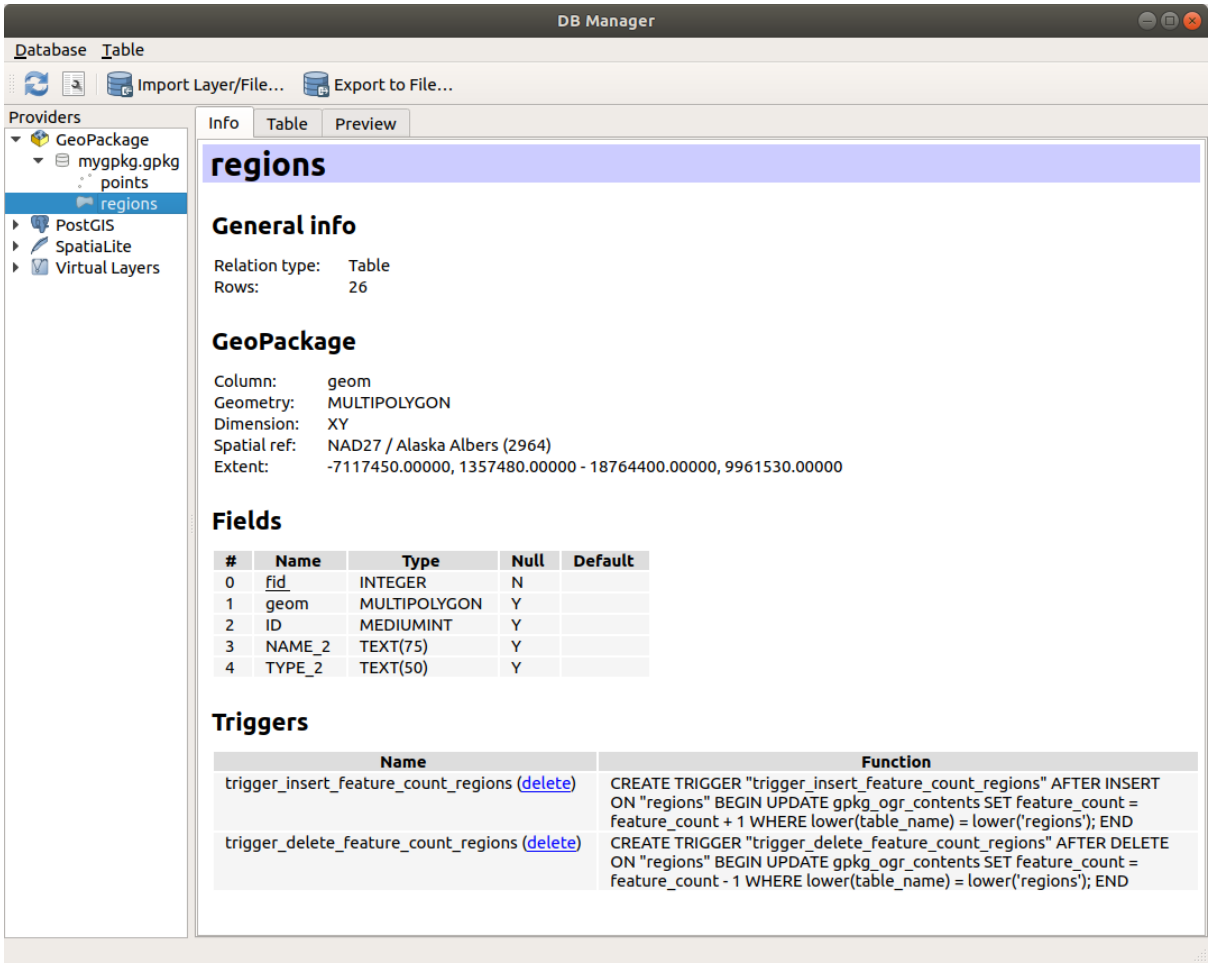


Fig. 15.3: DB Manager dialog

15.1.3 Provider-based loading tools

Beside the Browser Panel and the DB Manager, the main tools provided by QGIS to add layers, you'll also find tools that are specific to data providers.

Notă: Some *external plugins* also provide tools to open specific format files in QGIS.

Încărcarea unui strat dintr-un fișier

To load a layer from a file:

1. Open the layer type tab in the *Data Source Manager* dialog, ie click the Open Data Source Manager button (or press `Ctrl+L`) and enable the target tab or:
   - for vector data (like GML, ESRI Shapefile, Mapinfo and DXF layers): press `Ctrl+Shift+V`, select the *Layer* ► *Add Layer* ► *Add Vector Layer* menu option or click on the Add Vector Layer toolbar button.

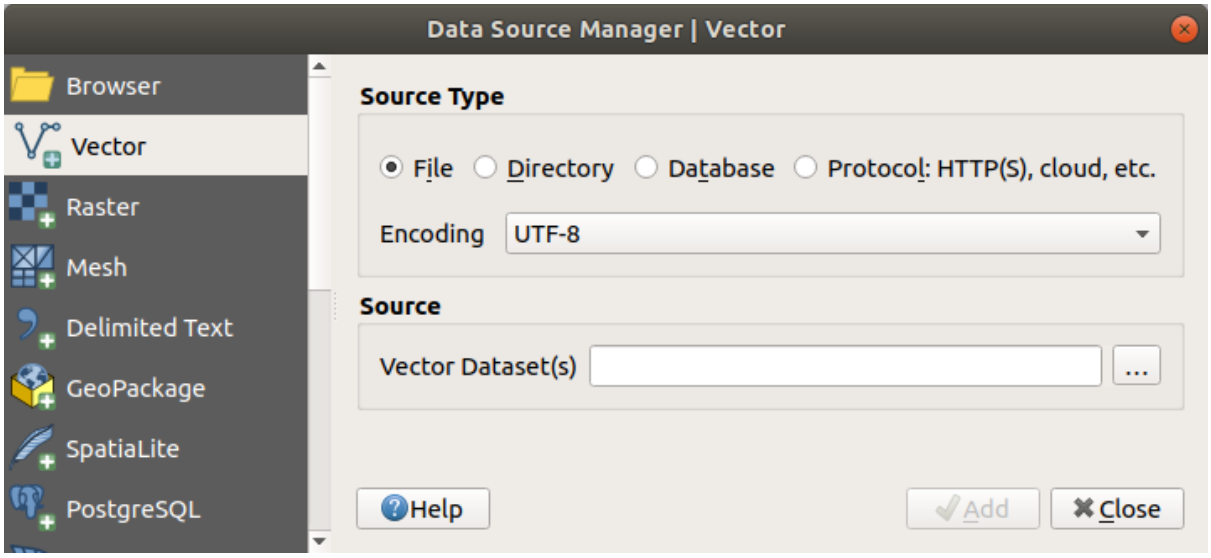


Fig. 15.4: Add Vector Layer Dialog

- for raster data (like GeoTiff, MBTiles, GRIdded Binary and DWG layers): press `Ctrl+Shift+R`, select the *Layer* ► *Add Layer* ► *Add Raster Layer* menu option or click on the Add Raster Layer toolbar button.

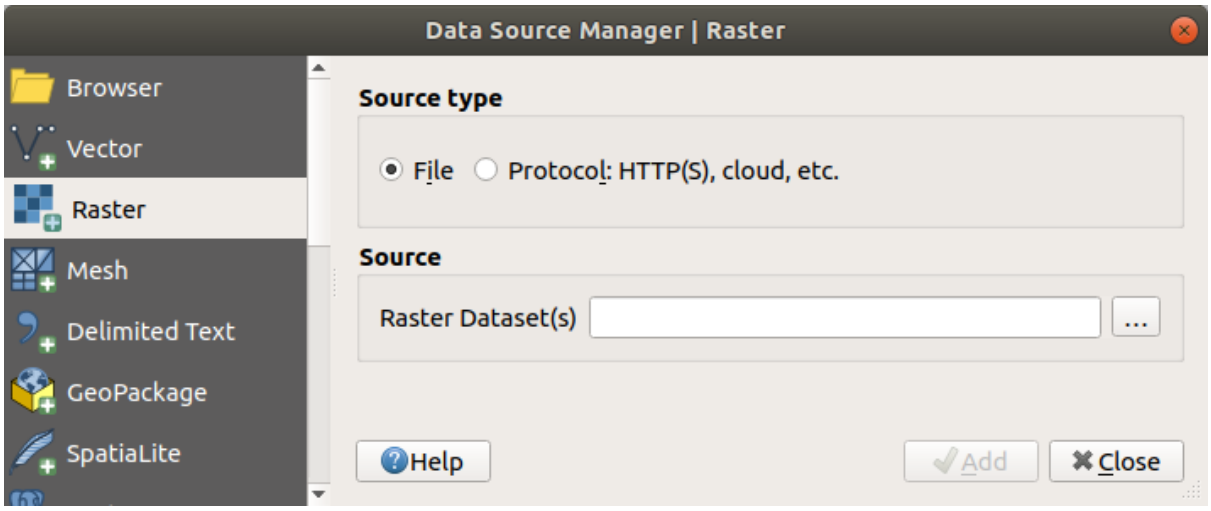


Fig. 15.5: Add Raster Layer Dialog

2. Check *File* source type
3. Click on the ... Browse button
4. Navigate the file system and load a supported data source. More than one layer can be loaded at the same time by holding down the `Ctrl` key and clicking on multiple items in the dialog or holding down the `Shift` key to select a range of items by clicking on the first and last items in the range. Only formats that have been well tested appear in the formats filter. Other formats can be loaded by selecting `All files` (the top item in the pull-down menu).
5. Press *Open* to load the selected file into *Data Source Manager* dialog

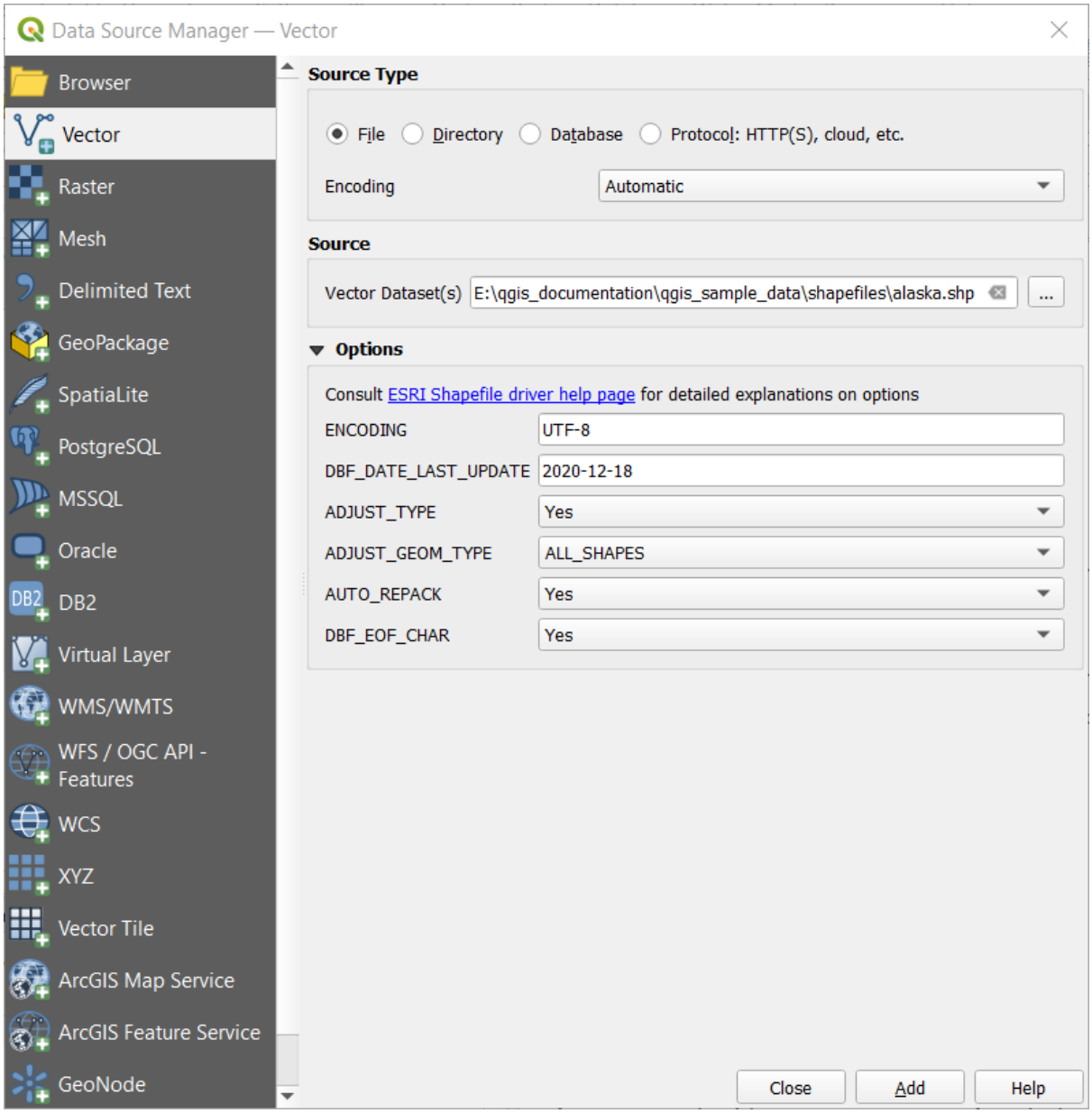


Fig. 15.6: Loading a Shapefile with open options

6. Press *Add* to load the file in QGIS and display them in the map view. When adding vector datasets containing multiple layers, the *Select Items to Add* dialog will appear. In this dialog, you can choose the specific layers from your dataset that you want to add. Also, under *Options* you can choose to:

   - ☑ *Add layers to a group*
   - ☑ *Show system and internal tables*
   - ☑ *Show empty vector layers.*

   Fig. 15.7 shows QGIS after loading the `alaska.shp` file.

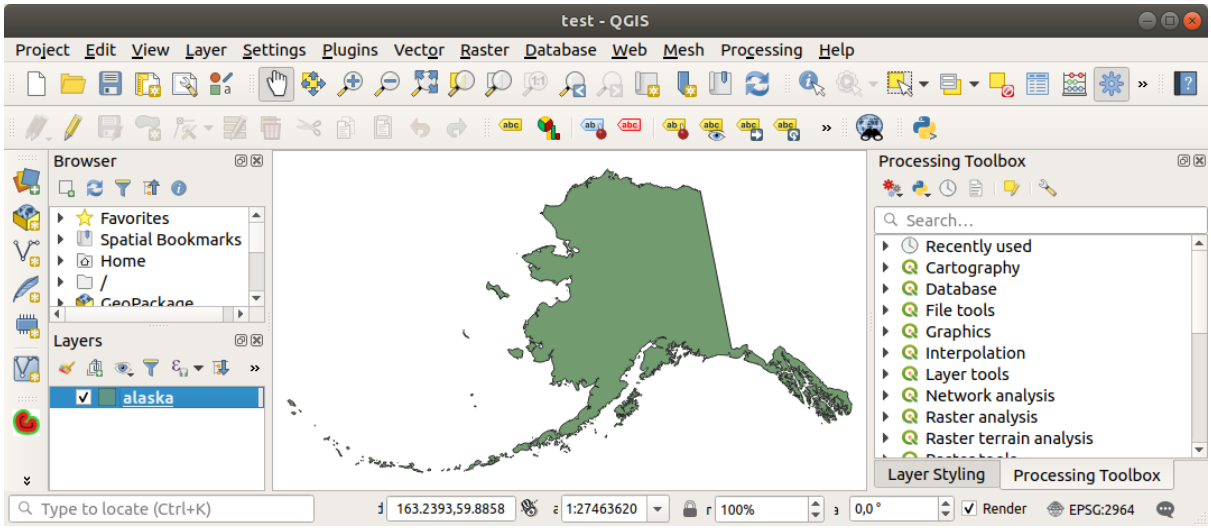


Fig. 15.7: QGIS with Shapefile of Alaska loaded

Notă: For loading vector and raster files the GDAL driver offers to define open actions. These will be shown when a file is selected. Options are described in detail on https://gdal.org/drivers/vector/, https://gdal.org/drivers/raster and if a file is selected in QGIS, a text with hyperlink will directly lead to the documentation of the selected file type.

Notă: Because some formats like MapInfo (e.g., `.tab`) or Autocad (`.dxf`) allow mixing different types of geometry in a single file, loading such datasets opens a dialog to select geometries to use in order to have one geometry per layer.

The Add Vector Layer and Add Raster Layer tabs allow loading of layers from source types other than *File*:

- You can load specific vector formats like `ArcInfo Binary Coverage`, `UK. National Transfer Format`, as well as the raw TIGER format of the `US Census Bureau` or `OpenfileGDB`. To do that, you select *Directory* as *Source type*. In this case, a directory can be selected in the dialog after pressing ... Browse.
- With the *Database* source type you can select an existing database connection or create one to the selected database type. Some possible database types are `ODBC`, `Esri Personal Geodatabase`, `MS SQL Server` as well as `PostgreSQL` or `MySQL` .

  Pressing the *New* button opens the *Create a New OGR Database Connection* dialog whose parameters are among the ones you can find in *Creating a stored Connection*. Pressing *Open* lets you select from the available tables, for example of PostGIS enabled databases.
- The *Protocol: HTTP(S), cloud, etc.* source type opens data stored locally or on the network, either publicly accessible, or in private buckets of commercial cloud storage services. Supported protocol types are:
  - `HTTP/HTTPS/FTP`, with a *URI* and, if required, an *authentication*.
  - Cloud storage such as `AWS S3`, `Google Cloud Storage`, `Microsoft Azure Blob`, `Alibaba OSS Cloud`, `Open Stack Swift Storage`. You need to fill in the *Bucket or container* and the *Object key*.
  - service supporting OGC `WFS 3` (still experimental), using `GeoJSON` or `GEOJSON - Newline Delimited` format or based on `CouchDB` database. A *URI* is required, with optional *authentication*.
  - For all vector source types it is possible to define the *Encoding* or to use the *Automatic* ► setting.

Loading a mesh layer

A mesh is an unstructured grid usually with temporal and other components. The spatial component contains a collection of vertices, edges and faces in 2D or 3D space. More information on mesh layers at *Working with Mesh Data*.

To add a mesh layer to QGIS:

1. Open the *Data Source Manager* dialog, either by selecting it from the *Layer* ► menu or clicking the Open Data Source Manager button.
2. Enable the *Mesh* tab on the left panel
3. Press the … Browse button to select the file. *Various formats* are supported.
4. Select the file and press *Add*. The layer will be added using the native mesh rendering.
5. If the selected file contains many mesh layers, then you'll be prompted with a dialog to choose the sublayers to load. Do your selection and press *OK* and the layers are loaded with the native mesh rendering. It's also possible to load them within a group.

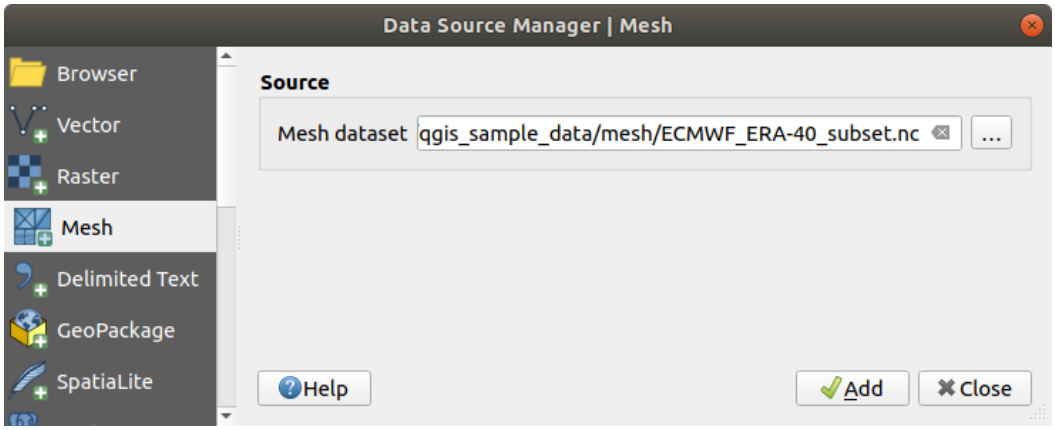


Fig. 15.8: Mesh tab in Data Source Manager

Importarea unui fișier cu texte delimitate

Delimited text files (e.g. `.txt`, `.csv`, `.dat`, `.wkt`) can be loaded using the tools described above. This way, they will show up as simple tables. Sometimes, delimited text files can contain coordinates / geometries that you could want to visualize. This is what *Add Delimited Text Layer* is designed for.

1. Click the Open Data Source Manager icon to open the *Data Source Manager* dialog
2. Enable the *Delimited Text* tab
3. Select the delimited text file to import (e.g., `qgis_sample_data/csv/elevp.csv`) by clicking on the … Browse button.
4. In the *Layer name* field, provide the name to use for the layer in the project (e.g. `Elevation`).
5. Configure the settings to meet your dataset and needs, as explained below.

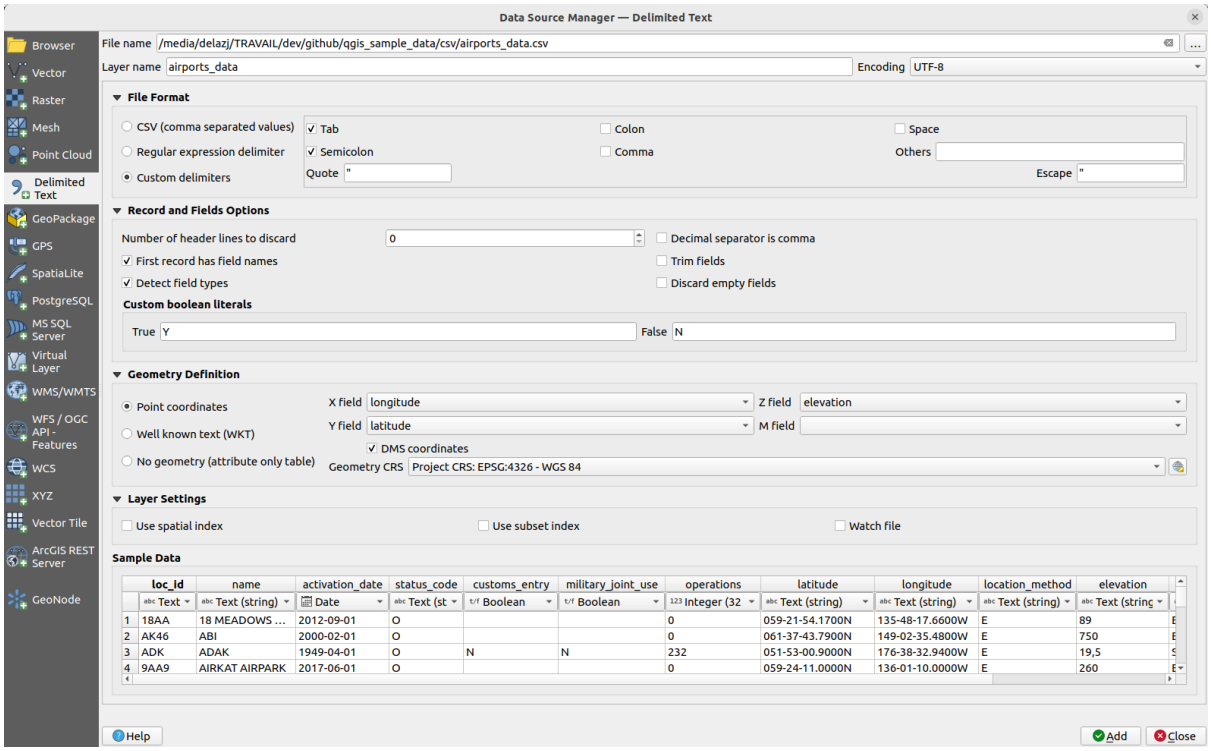


Fig. 15.9: Delimited Text Dialog

File format

Once the file is selected, QGIS attempts to parse the file with the most recently used delimiter, identifying fields and rows. To enable QGIS to correctly parse the file, it is important to select the right delimiter. You can specify a delimiter by choosing between:

- *CSV (comma separated values)* to use the comma character.
- *Regular expression delimiter* and enter text into the *Expression* field. For example, to change the delimiter to tab, use `\t` (this is used in regular expressions for the tab character).
- *Custom delimiters*, choosing among some predefined delimiters like `comma`, `space`, `tab`, `semicolon`, ...

Records and fields

Some other convenient options can be used for data recognition:

- *Number of header lines to discard*: convenient when you want to avoid the first lines in the file in the import, either because those are blank lines or with another formatting.
- *First record has field names*: values in the first line are used as field names, otherwise QGIS uses the field names `field_1`, `field_2`...
- *Detect field types*: automatically recognizes the field type. If unchecked then all attributes are treated as text fields.
- *Decimal separator is comma*: you can force decimal separator to be a comma.
- *Trim fields*: allows you to trim leading and trailing spaces from fields.

- *Discard empty fields*.
- *Custom boolean literals*: allows you to add a custom couple of string that will be detected as boolean values.

Field type detection

QGIS tries to detect the field types automatically (unless *Detect field types* is not checked) by examining the content of an optional sidecar CSVT file (see: GeoCSV specification) and by scanning the whole file to make sure that all values can actually be converted without errors, the fall-back field type is text.

The detected field type appears under the field name in sample data preview table and can be manually changed if necessary.

The following field types are supported:

- `Boolean` case-insensitive literal couples that are interpreted as boolean values are `1`/`0`, `true`/`false`, `t`/`f`, `yes`/`no`
- `Whole Number (integer)`
- `Whole Number (integer - 64 bit)`
- `Decimal Number`: double precision floating point number
- `Date`
- `Time`
- `Date and Time`
- `Text`

Geometry definition

Once the file is parsed, set *Geometry definition* to

- *Point coordinates* and provide the *X field*, *Y field*, *Z field* (for 3-dimensional data) and *M field* (for the measurement dimension) if the layer is of point geometry type and contains such fields. If the coordinates are defined as degrees/minutes/seconds, activate the *DMS coordinates* checkbox. Provide the appropriate *Geometry CRS* using the Select CRS widget.
- *Well known text (WKT)* option if the spatial information is represented as WKT: select the *Geometry field* containing the WKT geometry and choose the approriate *Geometry field* or let QGIS auto-detect it. Provide the appropriate *Geometry CRS* using the Select CRS widget.
- If the file contains non-spatial data, activate *No geometry (attribute only table)* and it will be loaded as an ordinary table.

Layer settings

Additionally, you can enable:

- *Use spatial index* to improve the performance of displaying and spatially selecting features.
- *Use subset index* to improve performance of *subset filters* (when defined in the layer properties).
- *Watch file* to watch for changes to the file by other applications while QGIS is running.

At the end, click *Add* to add the layer to the map. In our example, a point layer named `Elevation` is added to the project and behaves like any other map layer in QGIS. This layer is the result of a query on the `.csv` source file (hence, linked to it) and would require *to be saved* in order to get a spatial layer on disk.

Sample Data

As you set the parser properties, the sample data preview updates regarding to the applied settings.

Also in the Sample Data Table it is possible to override the automatically determined column types.

Importarea unui fișier DXF sau DWG

`DXF` and `DWG` files can be added to QGIS by simple drag-and-drop from the Browser Panel. You will be prompted to select the sublayers you would like to add to the project. Layers are added with random style properties.

Notă: For DXF files containing several geometry types (point, line and/or polygon), the name of the layers will be generated as *<filename.dxf> entities <geometry type>*.

To keep the dxf/dwg file structure and its symbology in QGIS, you may want to use the dedicated *Project* ► *Import/Export* ► *Import Layers from DWG/DXF…* tool which allows you to:

1. import elements from the drawing file into a GeoPackage database.
2. add imported elements to the project.

In the *DWG/DXF Import* dialog, to import the drawing file contents:

1. Input the location of the *Source drawing*, i.e. the DWG/DXF drawing file to import.
2. Specify the coordinate reference system of the data in the drawing file.
3. Input the location of the *Target package*, i.e. the GeoPackage file that will store the data. If an existing file is provided, then it will be overwritten.
4. Choose how to import `blocks` with the dedicated combobox:
   - *Expand Block Geometries*: imports the blocks in the drawing file as normal elements.
   - *Expand Block Geometries and Add Insert Points*: imports the blocks in the drawing file as normal elements and adds the insertion point as a point layer.
   - *Add Only Insert Points*: adds the blocks insertion point as a point layer.
5. Check *Use curves* to promote the imported layers to a `curved` geometry type.
6. Use the *Import* button to import the drawing into the destination GeoPackage file. The GeoPackage database will be automatically populated with the drawing file content. Depending on the size of the file, this can take some time.

After the `.dwg` or `.dxf` data has been imported into the GeoPackage database, the frame in the lower half of the dialog is populated with the list of layers from the imported file. There you can select which layers to add to the QGIS project:

1. At the top, set a *Group name* to group the drawing files in the project. By default this is set to the filename of the source drawing file.
2. Check layers to show: Each selected layer is added to an ad hoc group which contains vector layers for the point, line, label and area features of the drawing layer. The style of the layers will resemble the look they originally had in *CAD.
3. Choose if the layer should be visible at opening.
4. Checking the *Merge layers* option places all layers in a single group.
5. Press *OK* to open the layers in QGIS.

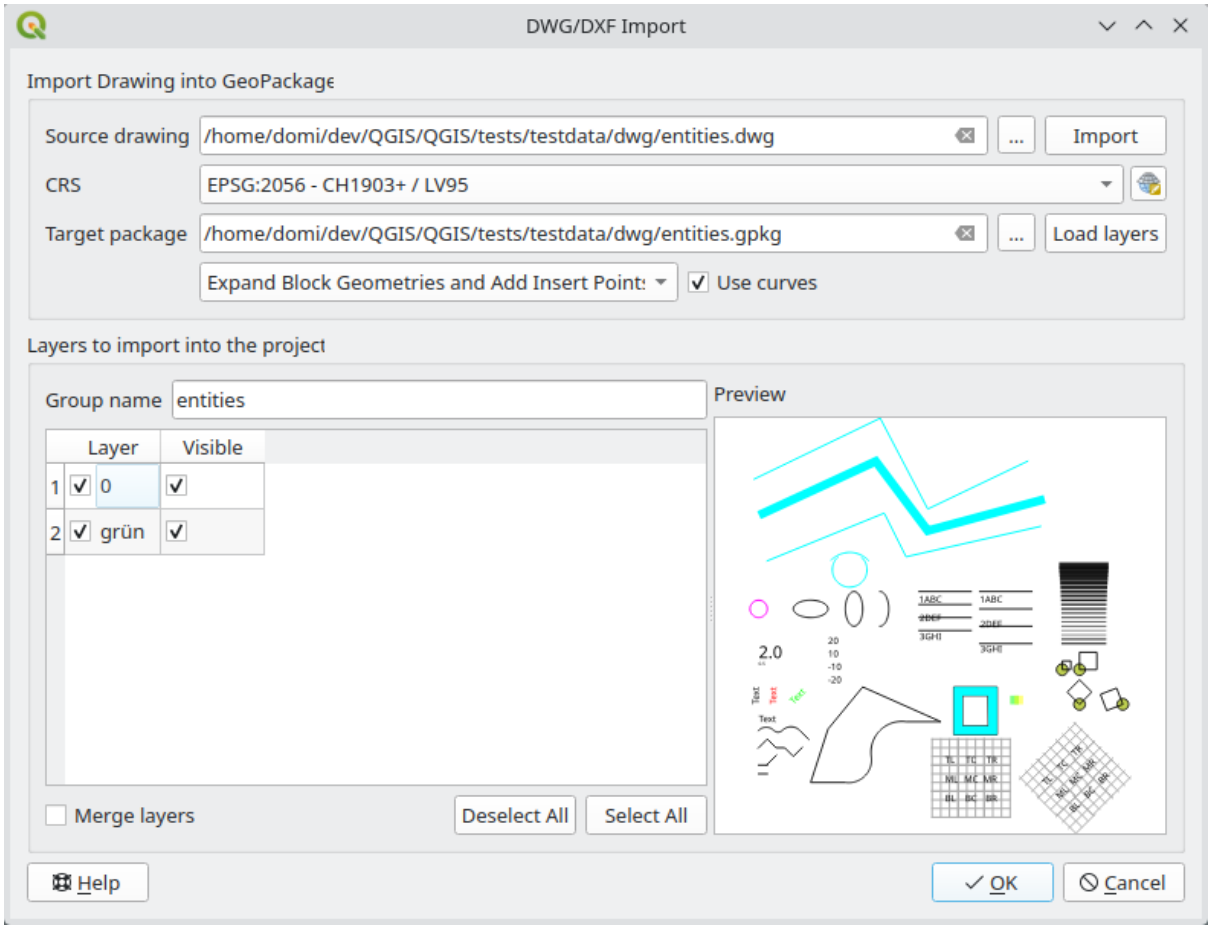


Fig. 15.10: Import dialog for DWG/DXF files

Importarea Vectorilor OpenStreetMap

The OpenStreetMap project is popular because in many countries no free geodata such as digital road maps are available. The objective of the OSM project is to create a free editable map of the world from GPS data, aerial photography and local knowledge. To support this objective, QGIS provides support for OSM data.

Using the *Browser Panel*, you can load an `.osm` file to the map canvas, in which case you'll get a dialog to select sublayers based on the geometry type. The loaded layers will contain all the data of that geometry type in the `.osm` file, and keep the `osm` file data structure.

Straturile SpatiaLite

The first time you load data from a SpatiaLite database, begin by:

- clicking on the Add SpatiaLite Layer toolbar button
- selecting the *Add SpatiaLite Layer…* option from the *Layer* ► *Add Layer* menu
- or by typing `Ctrl+Shift+L`

This will bring up a window that will allow you either to connect to a SpatiaLite database already known to QGIS (which you choose from the drop-down menu) or to define a new connection to a new database. To define a new connection, click on *New* and use the file browser to point to your SpatiaLite database, which is a file with a `.sqlite` extension.

QGIS also supports editable views in SpatiaLite.

GPS

There are dozens of different file formats for storing GPS data. The format that QGIS uses is called GPX (GPS eXchange format), which is a standard interchange format that can contain any number of waypoints, routes and tracks in the same file.

Use the … Browse button to select the GPX file, then use the check boxes to select the feature types you want to load from that GPX file. Each feature type will be loaded in a separate layer.

More on GPS data manipulation at *Lucrul cu datele GPS*.

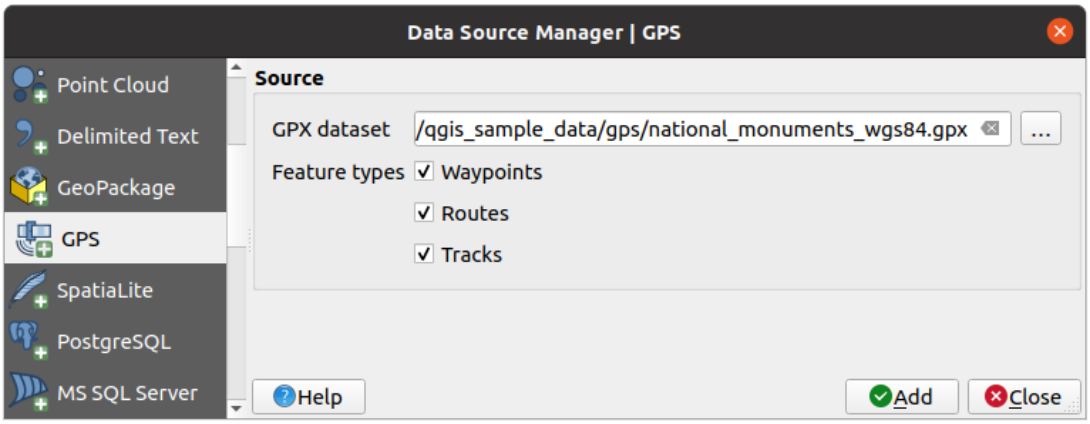


Fig. 15.11: Loading GPS Data dialog

GRASS

Working with GRASS vector data is described in section *Integrarea GRASS GIS*.

Database related tools

Creating a stored Connection

In order to read and write tables from a database format QGIS supports you have to create a connection to that database. While *QGIS Browser Panel* is the simplest and recommanded way to connect to and use databases, QGIS provides other tools to connect to each of them and load their tables:

- *Add PostGIS Layer…* or by typing `Ctrl+Shift+D`
- *Add MS SQL Server Layer*
- *Add Oracle Spatial Layer…* or by typing `Ctrl+Shift+O`
- *Add SAP HANA Spatial Layer…* or by typing `Ctrl+Shift+G`

These tools are accessible either from the *Manage Layers Toolbar* and the *Layer* ► *Add Layer* ► menu. Connecting to SpatiaLite database is described at *Straturile SpatiaLite*.

Sfat: Create connection to database from the QGIS Browser Panel

Selecting the corresponding database format in the Browser tree, right-clicking and choosing connect will provide you with the database connection dialog.

Most of the connection dialogs follow a common basis that will be described below using the PostgreSQL database tool as an example. For additional settings specific to other providers, you can find corresponding descriptions at:

- *Connecting to MS SQL Server*;
- *Connecting to Oracle Spatial*;
- *Connecting to SAP HANA*.

The first time you use a PostGIS data source, you must create a connection to a database that contains the data. Begin by clicking the appropriate button as exposed above, opening an *Add PostGIS Table(s)* dialog (see Fig. 15.15). To access the connection manager, click on the *New* button to display the *Create a New PostGIS Connection* dialog.

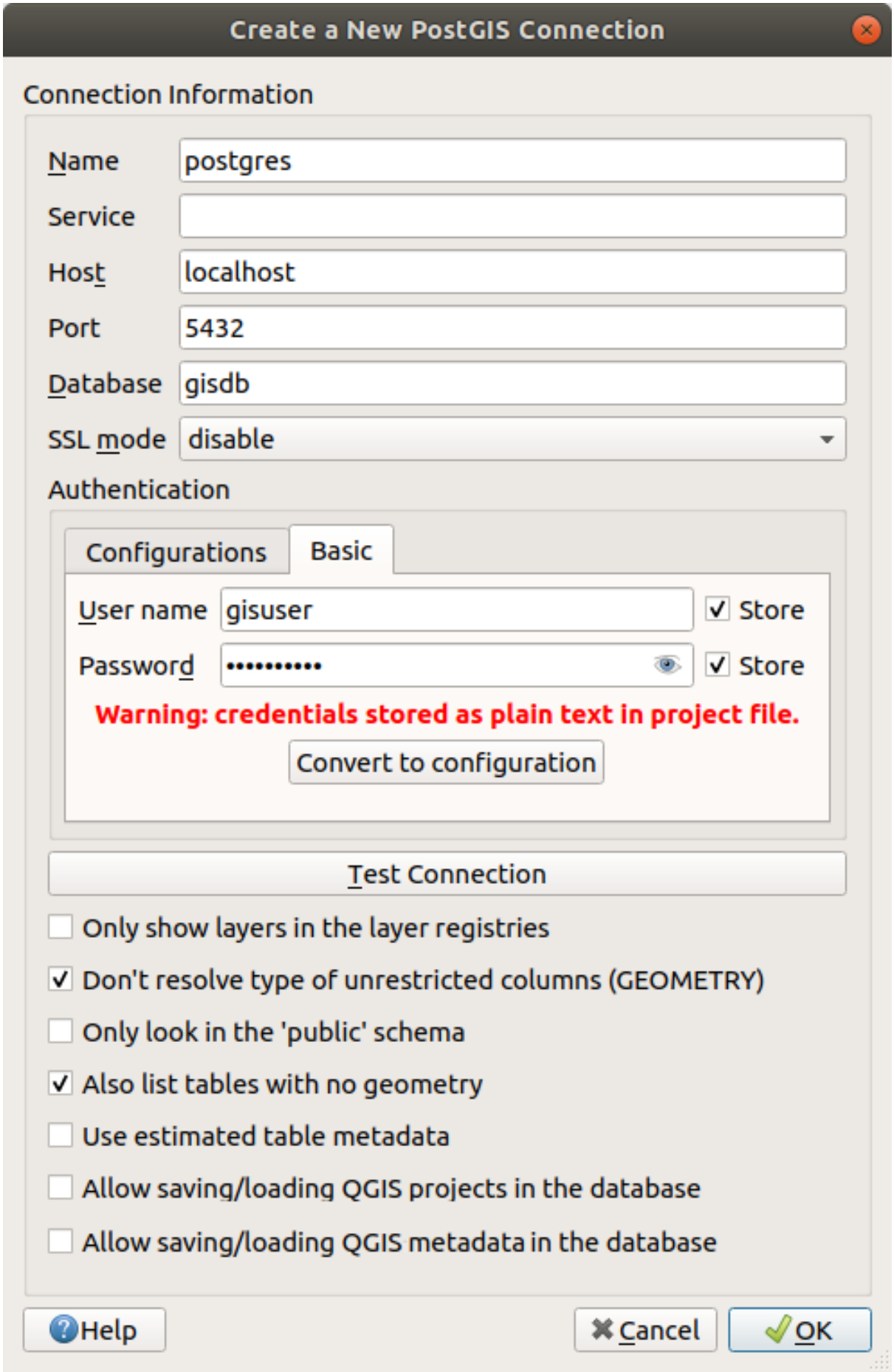


Fig. 15.12: Create a New PostGIS Connection Dialog

The parameters required for a PostGIS connection are explained below. For the other database types, see their differences at *Particular Connection requirements*.

- *Name*: A name for this connection. It can be the same as *Database*.
- *Service*: Service parameter to be used alternatively to hostname/port (and potentially database). This can be defined in `pg_service.conf`. Check the *PostgreSQL Service connection file* section for more details.
- *Host*: Name of the database host. This must be a resolvable host name such as would be used to open a TCP/IP connection or ping the host. If the database is on the same computer as QGIS, simply enter *localhost* here.
- *Port*: Port number the PostgreSQL database server listens on. The default port for PostGIS is `5432`.
- *Database*: Name of the database.
- *SSL mode*: SSL encryption setup The following options are available:
  - *Prefer* (the default): I don't care about encryption, but I wish to pay the overhead of encryption if the server supports it.
  - *Require*: I want my data to be encrypted, and I accept the overhead. I trust that the network will make sure I always connect to the server I want.
  - *Verify CA*: I want my data encrypted, and I accept the overhead. I want to be sure that I connect to a server that I trust.
  - *Verify Full*: I want my data encrypted, and I accept the overhead. I want to be sure that I connect to a server I trust, and that it's the one I specify.
  - *Allow*: I don't care about security, but I will pay the overhead of encryption if the server insists on it.
  - *Disable*: I don't care about security, and I don't want to pay the overhead of encryption.
- *Authentication*, basic.
  - *User name*: User name used to log in to the database.
  - *Password*: Password used with *Username* to connect to the database.

  You can save any or both of the `User name` and `Password` parameters, in which case they will be used by default each time you need to connect to this database. If not saved, you'll be prompted to supply the credentials to connect to the database in next QGIS sessions. The connection parameters you entered are stored in a temporary internal cache and returned whenever a username/password for the same database is requested, until you end the current QGIS session.

  Atenționare: QGIS User Settings and Security

  In the *Authentication* tab, saving **username** and **password** will keep unprotected credentials in the connection configuration. Those **credentials will be visible** if, for instance, you share the project file with someone. Therefore, it is advisable to save your credentials in an *Authentication configuration* instead (*Configurations* tab - See *Sistem de Autentificare* for more details) or in a service connection file (see *PostgreSQL Service connection file* for example).

- *Authentication*, configurations. Choose an authentication configuration. You can add configurations using the button. Choices are:
  - Basic authentication
  - PKI PKCS#12 authentication
  - PKI paths authentication
  - PKI stored identity certificate

Optionally, depending on the type of database, you can activate the following checkboxes:

- *Only show layers in the layer registries*

- *Don't resolve type of unrestricted columns (GEOMETRY)*
- *Only look in the «public» schema*
- *Also list tables with no geometry*
- *Use estimated table metadata*
- *Allow saving/loading QGIS projects in the database* - more details *here*
- *Allow saving/loading QGIS layer metadata in the database* - more details *here*

Sfat: Use estimated table metadata to speed up operations

When initializing layers, various queries may be needed to establish the characteristics of the geometries stored in the database table. When the *Use estimated table metadata* option is checked, these queries examine only a sample of the rows and use the table statistics, rather than the entire table. This can drastically speed up operations on large datasets, but may result in incorrect characterization of layers (e.g. the feature count of filtered layers will not be accurately determined) and may even cause strange behaviour if columns that are supposed to be unique actually are not.

Once all parameters and options are set, you can test the connection by clicking the *Test Connection* button or apply it by clicking the *OK* button. From *Add PostGIS Table(s)*, click now on *Connect*, and the dialog is filled with tables from the selected database (as shown in Fig. 15.15).

Particular Connection requirements

Because of database type particularities, provided options are not the same. Database specific options are described below.

PostgreSQL Service connection file

The service connection file allows PostgreSQL connection parameters to be associated with a single service name. That service name can then be specified by a client and the associated settings will be used.

It's called `.pg_service.conf` under *nix systems (GNU/Linux, macOS etc.) and `pg_service.conf` on Windows.

The service file can look like this:

```
[water_service]
host=192.168.0.45
port=5433
dbname=gisdb
user=paul
password=paulspass

[wastewater_service]
host=dbserver.com
dbname=water
user=waterpass
```

Notă: There are two services in the above example: `water_service` and `wastewater_service`. You can use these to connect from QGIS, pgAdmin, etc. by specifying only the name of the service you want to connect to (without the enclosing brackets). If you want to use the service with `psql`, you can do `psql service=water_service`.

You can find all the PostgreSQL parameters here

Notă: If you don't want to save the passwords in the service file you can use the .pg_pass option.

Notă: QGIS Server and service

When using a service file and QGIS Server, you must configure the service on the server side as well. You can follow the QGIS Server documentation.

On *nix operating systems (GNU/Linux, macOS etc.) you can save the `.pg_service.conf` file in the user's home directory and PostgreSQL clients will automatically be aware of it. For example, if the logged user is `web`, `.pg_service.conf` should be saved in the `/home/web/` directory in order to directly work (without specifying any other environment variables).

You can specify the location of the service file by creating a `PGSERVICEFILE` environment variable (e.g. run the `export PGSERVICEFILE=/home/web/.pg_service.conf` command under your *nix OS to temporarily set the `PGSERVICEFILE` variable)

You can also make the service file available system-wide (all users) either by placing the `.pg_service.conf` file in `pg_config --sysconfdir` or by adding the `PGSYSCONFDIR` environment variable to specify the directory containing the service file. If service definitions with the same name exist in the user and the system file, the user file takes precedence.

Atenționare: There are some caveats under Windows:

- The service file should be saved as `pg_service.conf` and not as `.pg_service.conf`.
- The service file should be saved in Unix format in order to work. One way to do it is to open it with Notepad++ and *Edit* ► *EOL Conversion* ► *UNIX Format* ► *File save*.
- You can add environmental variables in various ways; a tested one, known to work reliably, is *Control Panel* ► *System and Security* ► *System* ► *Advanced system settings* ► *Environment Variables* adding `PGSERVICEFILE` with the path - e.g. `C:\Users\John\pg_service.conf`
- After adding an environment variable you may also need to restart the computer.

Connecting to Oracle Spatial

The spatial features in Oracle Spatial aid users in managing geographic and location data in a native type within an Oracle database. In addition to some of the options in *Creating a stored Connection*, the connection dialog proposes:

- **Database**: SID or SERVICE_NAME of the Oracle instance;
- **Port**: Port number the Oracle database server listens on. The default port is `1521`;
- **Options**: Oracle connection specific options (e.g. OCI_ATTR_PREFETCH_ROWS, OCI_ATTR_PREFETCH_MEMORY). The format of the options string is a semicolon separated list of option names or option=value pairs;
- **Workspace**: Workspace to switch to;
- **Schema**: Schema in which the data are stored

Optionally, you can activate the following checkboxes:

- ☑ *Only look in metadata table*: restricts the displayed tables to those that are in the `all_sdo_geom_metadata` view. This can speed up the initial display of spatial tables.

- *Only look for user's tables*: when searching for spatial tables, restricts the search to tables that are owned by the user.
- *Also list tables with no geometry*: indicates that tables without geometry should also be listed by default.
- *Use estimated table statistics for the layer metadata*: when the layer is set up, various metadata are required for the Oracle table. This includes information such as the table row count, geometry type and spatial extents of the data in the geometry column. If the table contains a large number of rows, determining this metadata can be time-consuming. By activating this option, the following fast table metadata operations are done: Row count is determined from `all_tables.num_rows`. Table extents are always determined with the SDO_TUNE.EXTENTS_OF function, even if a layer filter is applied. Table geometry is determined from the first 100 non-null geometry rows in the table.
- *Only existing geometry types*: only lists the existing geometry types and don't offer to add others.
- *Include additional geometry attributes*.

Sfat: Oracle Spatial Layers

Normally, an Oracle Spatial layer is defined by an entry in the **USER_SDO_METADATA** table.

To ensure that selection tools work correctly, it is recommended that your tables have a **primary key**.

Connecting to MS SQL Server

As mentioned in *Creating a stored Connection* QGIS allows you to create MS SQL Server connection through *Data Source Manager*.

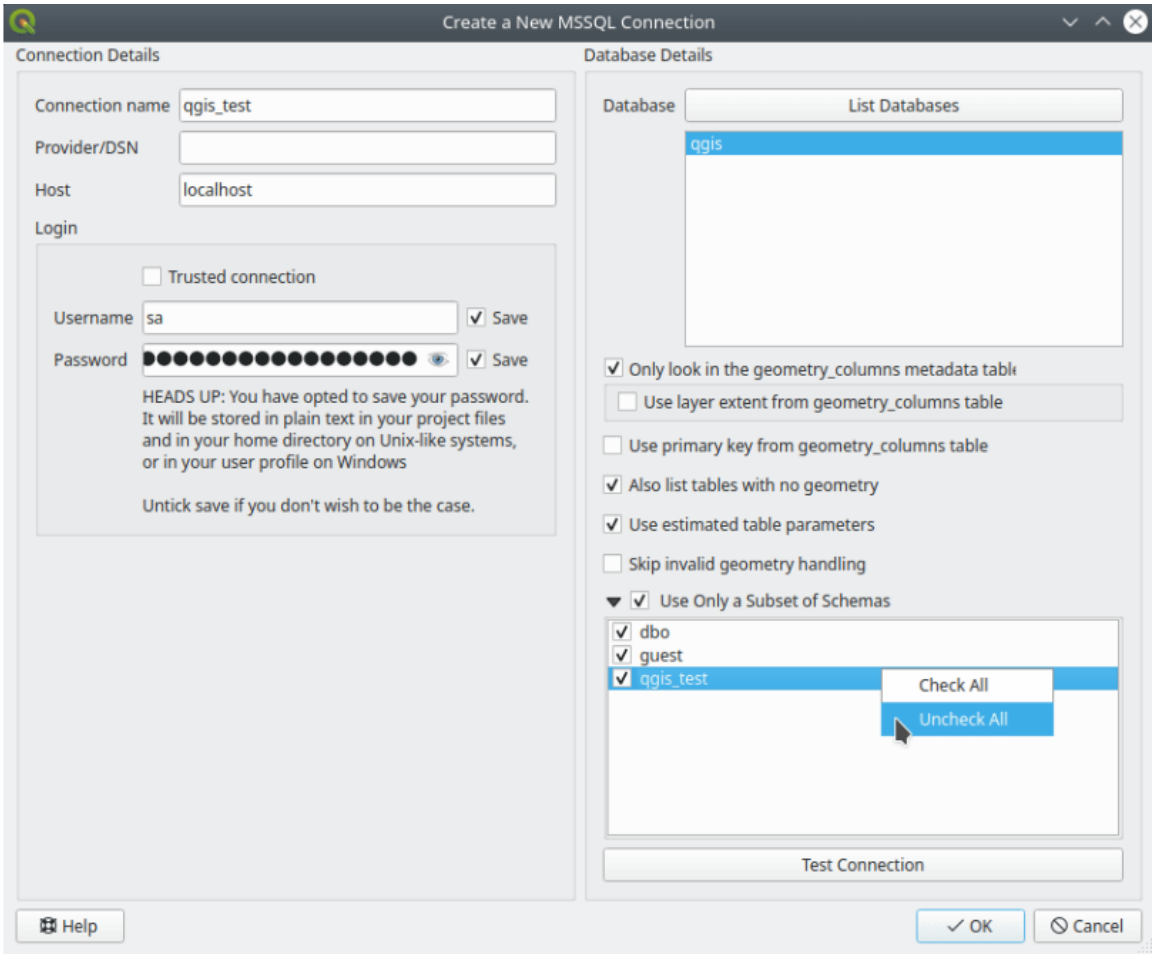


Fig. 15.13: MS SQL Server Connection

To create a new MS SQL Server connection, you need to provide some of the following information in the *Connection Details* dialog:

- *Connection name*
- *Provider/DNS*
- *Host*
- *Login* information. You can choose to *Save* your credentials.

Navigate to the *Database Details* section and click the *List Databases* button to view the available datasets. Select datasets that you want, then press *OK*. Optionally, you can also perform a *Test Connection*. Once you click *OK* the *Create a New MS SQL Server Connection* dialog will close and in the *Data Source Manager* press *Connect*, select a layer and then click *Add*.

Optionally, you can activate the following options:

- *Only look in the geometry_columns metadata table*.
- *Use layer extent from geometry_columns table*, this checkbox is dependent on the first one; it remains disabled unless the first option is checked.
- *Use primary key from geometry_columns table*
- *Also list table with no geometry*: tables without a geometry column attached will also be shown in the available table list.

- *Use estimated table parameters*: only estimated table metadata will be used. This avoids a slow table scan, but may result in incorrect layer properties such as layer extent.
- *Skip invalid geometry handling*: all handling of records with invalid geometry will be disabled. This speeds up the provider, however, if any invalid geometries are present in a table then the result is unpredictable and may include missing records. Only check this option if you are certain that all geometries present in the database are valid, and any newly added geometries or tables will also be valid.
- *Use only a Subset of Schemas* will alow you to filter schemas for MS SQL connection. If enabled, only checked schemas will be displayed. You can right-click to *Check* or *Uncheck* any schema in the list.

Connecting to SAP HANA

Notă: You require the SAP HANA Client to connect to a SAP HANA database. You can download the SAP HANA Client for your platform at the SAP Development Tools website.

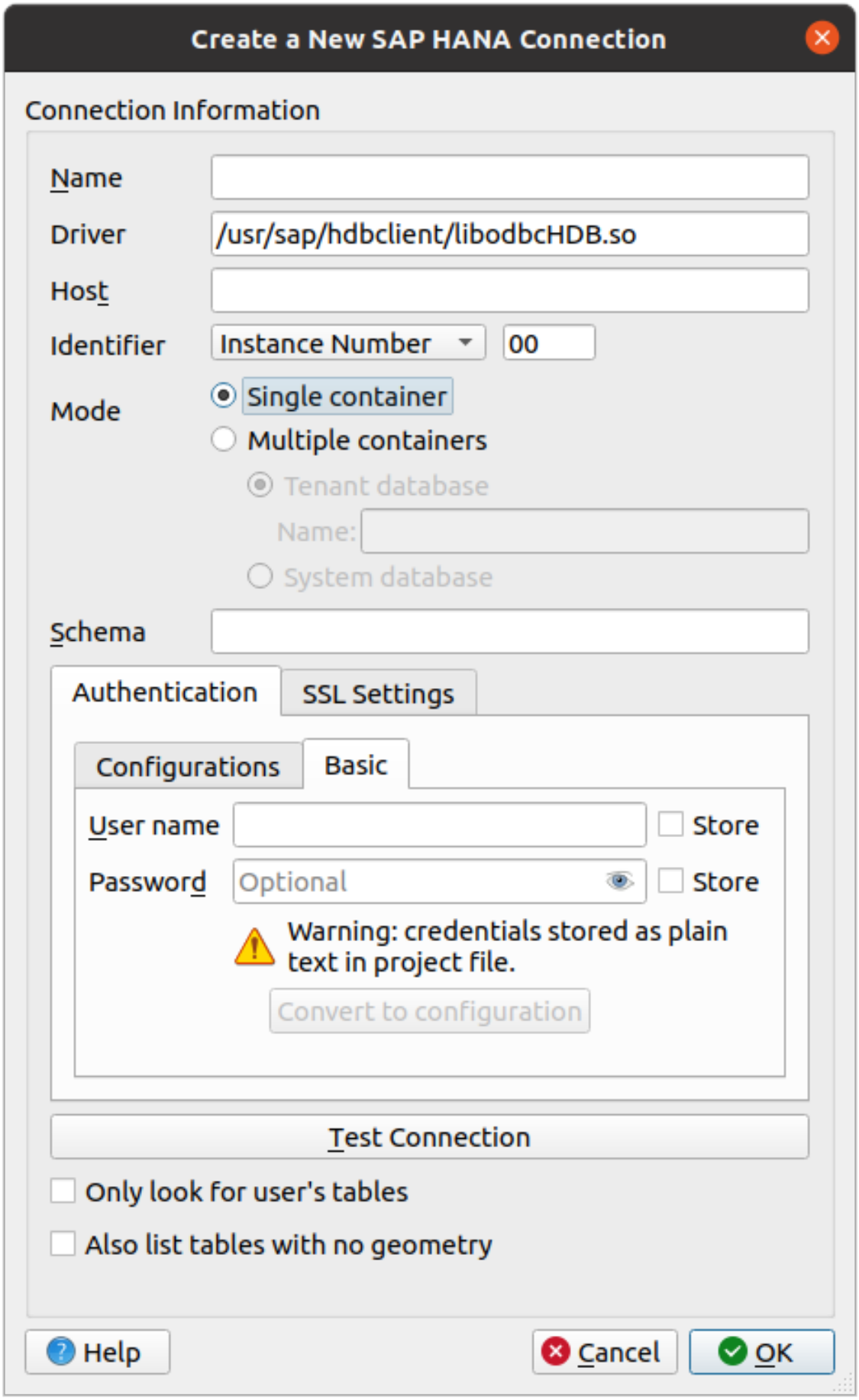


Fig. 15.14: Create a New SAP HANA Connection Dialog

The following parameters can be entered:

- *Name*: A name for this connection.
- *Driver* : The name of the HANA ODBC driver. It is `HDBODBC` if you are using 64-bit QGIS, `HDBODBC32` if you are using 32-bit QGIS. The appropriate driver name is entered automatically.
- *Driver* : Either the name under which the SAP HANA ODBC driver has been registered in `/etc/odbcinst.ini` or the full path to the SAP HANA ODBC driver. The SAP HANA Client installer will install the ODBC driver to `/usr/sap/hdbclient/libodbcHDB.so` by default.
- *Host*: The name of the database host.
- *Identifier*: Identifies the instance to connect to on the host. This can be either *Instance Number* or *Port Number*. Instance numbers consist of two digits, port numbers are in the range from 1 to 65,535.
- *Mode*: Specifies the mode in which the SAP HANA instance runs. This setting is only taken into account if *Identifier* is set to *Instance Number*. If the database hosts multiple containers, you can either connect to a tenant with the name given at *Tenant database* or you can connect to the system database.
- *Schema*: This parameter is optional. If a schema name is given, QGIS will only search for data in that schema. If this field is left blank, QGIS will search for data in all schemas.
- *Authentication* ► *Basic*.
  - *User name*: User name used to connect to the database.
  - *Password*: Password used to connect to the database.
- *SSL Settings*
  - *Enable TLS/SSL encryption*: Enables TLS 1.1 - TLS1.2 encryption. The server will choose the highest available.
  - *Provider*: Specifies the cryptographic library provider used for SSL communication. *sapcrypto* should work on all platforms, *openssl* should work on , *mscrypto* should work on and *commoncrypto* requires CommonCryptoLib to be installed.
  - *Validate SSL certificate*: If checked, the SSL certificate will be validated using the truststore given in *Trust store file with public key*.
  - *Override hostname in certificate*: Specifies the host name used to verify server's identity. The host name specified here verifies the identity of the server instead of the host name with which the connection was established. If you specify `*` as the host name, then the server's host name is not validated. Other wildcards are not permitted.
  - *Keystore file with private key*: Currently ignored. This parameter might allow to authenticate via certificate instead via user and password in future.
  - *Trust store file with public key*: Specifies the path to a trust store file that contains the server's public certificates if using OpenSSL. Typically, the trust store contains the root certificate or the certificate of the certification authority that signed the server's public certificates. If you are using the cryptographic library CommonCryptoLib or msCrypto, then leave this property empty.
- *Only look for user's tables*: If checked, QGIS searches only for tables and views that are owned by the user that connects to the database.
- *Also list tables with no geometries*: If checked, QGIS searches also for tables and views that do not contain a spatial column.

Sfat: Connecting to SAP HANA Cloud

If you'd like to connect to an SAP HANA Cloud instance, you usually must set *Port Number* to `443` and check *Enable TLS/SSL encryption*.

Încărcarea unui Strat al unei Baze de date

Once you have one or more connections defined to a database (see section *Creating a stored Connection*), you can load layers from it. Of course, this requires that data are available. See section *Importing Data into PostgreSQL* for a discussion on importing data into a PostGIS database.

To load a layer from a database, you can perform the following steps:

1. Open the „Add <database> table(s)" dialog (see *Creating a stored Connection*).
2. Choose the connection from the drop-down list and click *Connect*.
3. Select or unselect *Also list tables with no geometry*.
4. Optionally, use some *Search Options* to reduce the list of tables to those matching your search. You can also set this option before you hit the *Connect* button, speeding up the database fetching.
5. Find the layer(s) you wish to add in the list of available layers.
6. Select it by clicking on it. You can select multiple layers by holding down the `Shift` or `Ctrl` key while clicking.
7. If applicable, use the *Set Filter* button (or double-click the layer) to start the *Query Builder* dialog (see section *Constructorul de Interogări*) and define which features to load from the selected layer. The filter expression appears in the `sql` column. This restriction can be removed or edited in the *Layer Properties* ► *General* ► *Provider Feature Filter* frame.
8. The checkbox in the `Select at id` column that is activated by default gets the feature ids without the attributes and generally speeds up the data loading.
9. Click on the *Add* button to add the layer to the map.

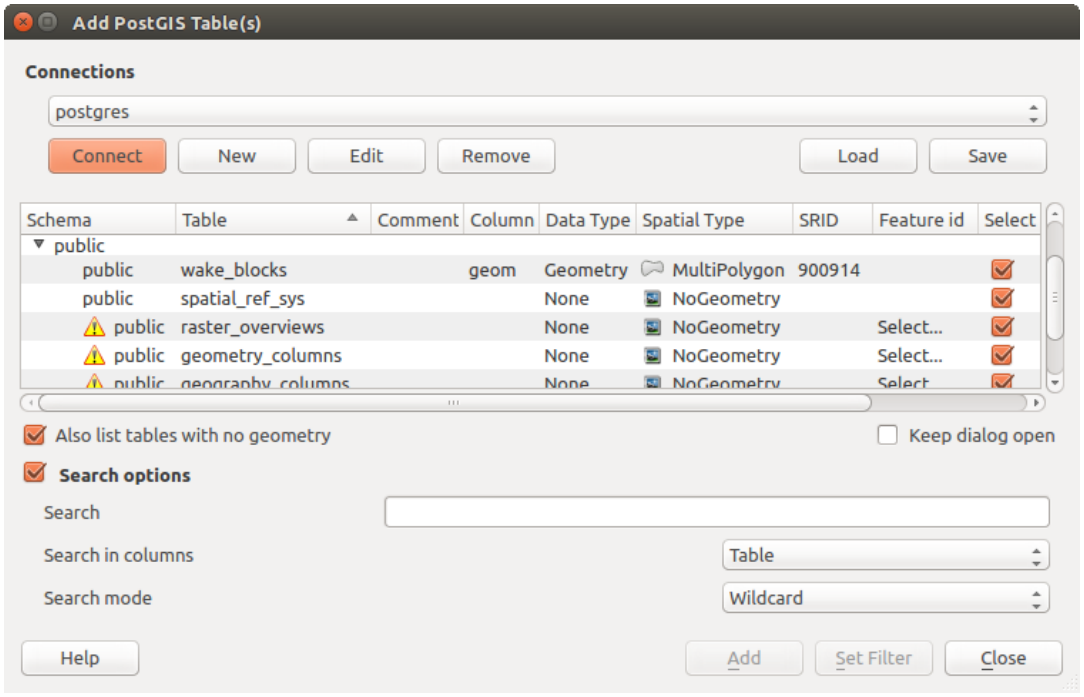


Fig. 15.15: Add PostGIS Table(s) Dialog

Sfat: Use the Browser Panel to speed up loading of database table(s)

Adding DB tables from the *Data Source Manager* may sometimes be time consuming as QGIS fetches statistics and properties (e.g. geometry type and field, CRS, number of features) for each table beforehand. To avoid this, once *the*

connection is set, it is better to use the *Browser Panel* or the *DB Manager* to drag and drop the database tables into the map canvas.

15.1.4 The Layer Metadata Search Panel

By default, QGIS can retrieve layers metadata from the connections or data providers that allow metadata storage (more details on *saving metadata to the database*). The *Metadata search* panel allows to browse the layers by their metadata and add them to the project (either with a double-click or the *Add* button). The list can be filtered:

- by text, watching a set of metadata properties (identifier, title, abstract)
- by spatial extent, using the current *project extent* or the map canvas extent
- by the layer (geometry) type

Notă: The sources of metadata are implemented through a layer metadata provider system that can be extended by plugins.

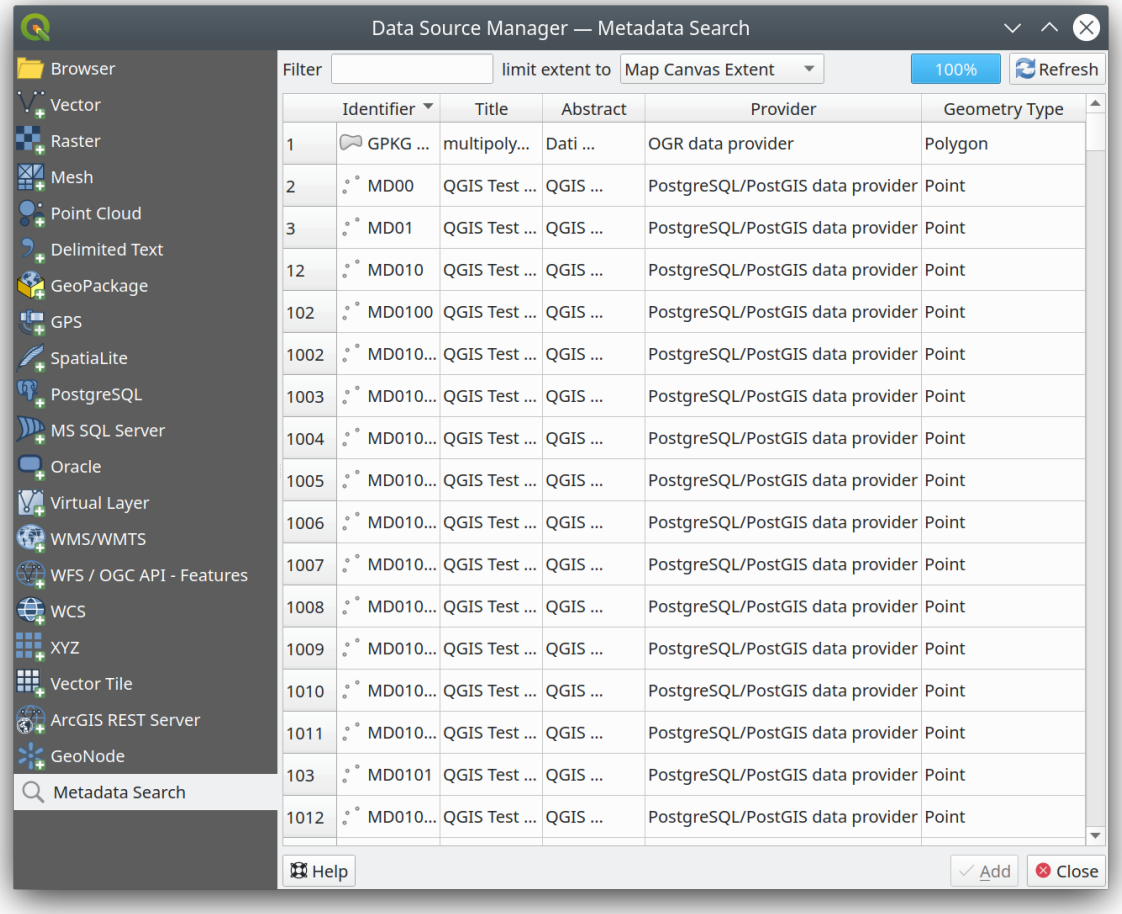


Fig. 15.16: Layer Metadata Search Panel

15.1.5 QGIS Custom formats

QGIS proposes two custom formats:

- Temporary Scratch Layer: a memory layer that is bound to the project (see *Creating a new Temporary Scratch Layer* for more information)
- Virtual Layers: a layer resulting from a query on other layer(s) (see *Creating virtual layers* for more information)

15.1.6 QLR - QGIS Layer Definition File

Layer definitions can be saved as a *Layer Definition File* (QLR - `.qlr`) using *Export* ► *Save As Layer Definition File…* in the layer context menu.

The QLR format makes it possible to share „complete" QGIS layers with other QGIS users. QLR files contain links to the data sources and all the QGIS style information necessary to style the layer.

QLR files are shown in the Browser Panel and can be used to add layers (with their saved styles) to the Layers Panel. You can also drag and drop QLR files from the system file manager into the map canvas.

15.1.7 Connecting to web services

With QGIS you can get access to different types of OGC web services (WM(T)S, WFS(-T), WCS, CSW, …). Thanks to QGIS Server, you can also publish such services. QGIS-Server-manual contains descriptions of these capabilities.

Using Vector Tiles services

Vector Tile services can be added via the *Vector Tiles* tab of the *Data Source Manager* dialog or the contextual menu of the *Vector Tiles* entry in the *Browser* panel. Services can be either a *New Generic Connection…* or a *New ArcGIS Vector Tile Service Connection…*.

You set up a service by adding:

- a *Name*
- the *URL*: of the type `http://example.com/{z}/{x}/{y}.pbf` for generic services and `http://example.com/arcgis/rest/services/Layer/VectorTileServer` for ArcGIS based services. The service must provide tiles in `.pbf` format.
- the *Min. Zoom Level* and the *Max. Zoom Level*. Vector Tiles have a pyramid structure. By using these options you have the opportunity to individually generate layers from the tile pyramid. These layers will then be used to render the Vector Tile in QGIS.

  For Mercator projection (used by OpenStreetMap Vector Tiles) Zoom Level 0 represents the whole world at a scale of 1:500.000.000. Zoom Level 14 represents the scale 1:35.000.
- a *Style URL*: a URL to a MapBox GL JSON style configuration. If provided, then that style will be applied whenever the layers from the connection are added to QGIS. In the case of Arcgis vector tile service connections, the URL overrides the default style configuration specified in the server configuration.
- the *authentication* configuration if necessary
- a *Referer*

Fig. 15.17 shows the dialog with the MapTiler planet Vector Tiles service configuration.

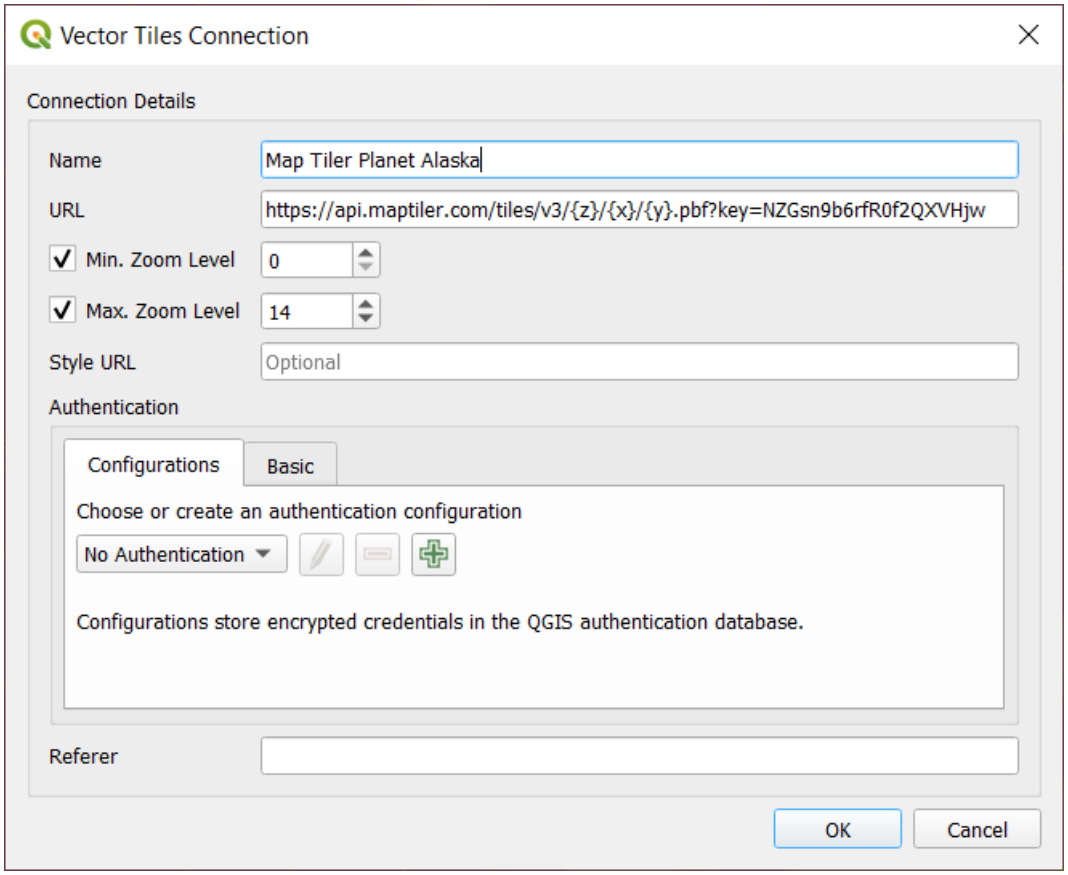


Fig. 15.17: Vector Tiles - Maptiler Planet configuration

Configurations can be saved to `.XML` file (*Save Connections*) through the *Vector Tiles* entry in *Data Source Manager* dialog or its context menu in the *Browser* panel. Likewise, they can be added from a file (*Load Connections*).

Once a connection to a vector tile service is set, it's possible to:

- *Edit* the vector tile connection settings
- *Remove* the connection
- From the *Browser* panel, right-click over the entry and you can also:
  - *Add layer to project*: a double-click also adds the layer
  - View the *Layer Properties...* and get access to metadata and a preview of the data provided by the service. More settings are available when the layer has been loaded into the project.

Using XYZ Tile services

XYZ Tile services can be added via the *XYZ* tab of the *Data Source Manager* dialog or the contextual menu of the *XYZ Tiles* entry in the *Browser* panel. By default, QGIS provides some default and ready-to-use XYZ Tiles services:

- *Mapzen Global Terrain*, allowing an immediate access to global DEM source for the projects. More details and resources at https://registry.opendata.aws/terrain-tiles/
- *OpenStreetMap* to access the world 2D map. Fig. 15.18 shows the dialog with the OpenStreetMap XYZ Tile service configuration.

To add a new service, press *New* (respectively *New Connection* from the Browser panel) and provide:

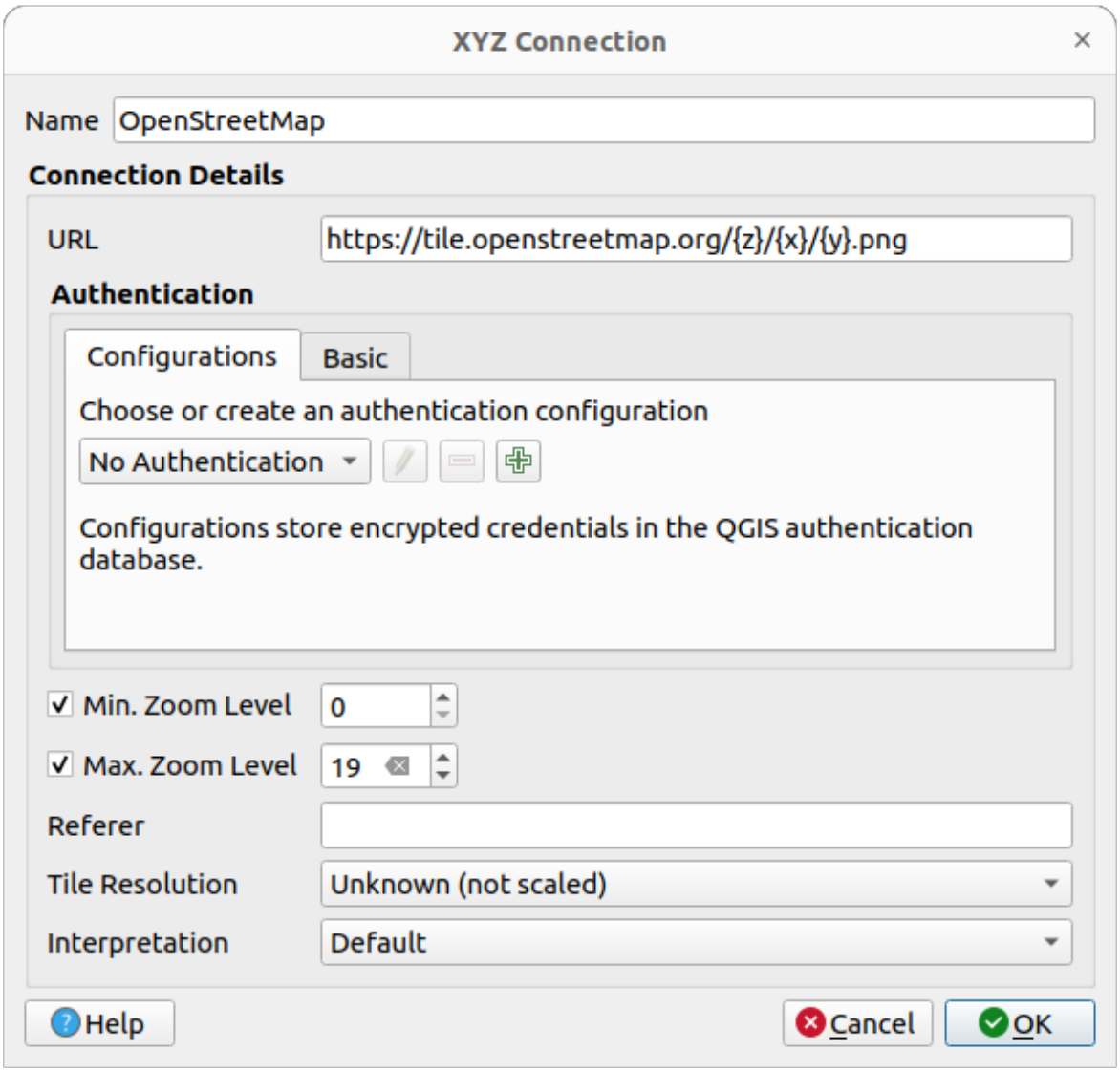


Fig. 15.18: XYZ Tiles - OpenStreetMap configuration

- a *Name*
- the *URL*, you can add `http://example.com/{z}/{x}/{y}.png` or `file:///local_path/{z}/{x}/{y}.png`
- the *authentication* configuration if necessary
- the *Min. Zoom level* and *Max. Zoom level*
- a *Referer*
- the *Tile Resolution*: possible values are *Unknown (not scaled)*, *Standard (256x256 / 96DPI)* and *High (512x512 / 192DPI)*
- *Interpretation*: converts WMTS/XYZ raster datasets to a raster layer of single band float type following a predefined encoding scheme. Supported schemes are *Default* (no conversion is done), *MapTiler Terrain RGB* and *Terrarium Terrain RGB*. The selected converter will translate the RGB source values to float values for each pixel. Once loaded, the layer will be presented as a single band floating point raster layer, ready for styling using QGIS usual *raster renderers*.

Press *OK* to establish the connection. It will then be possible to:

- *Add* the new layer to the project; it is loaded with the name given in the settings.
- *Edit* the XYZ connection settings
- *Remove* the connection
- From the *Browser* panel, right-click over the entry and you can also:
  - *Export layer…* ► *To File*, *saving it as a raster*
  - *Add layer to project*: a double-click also adds the layer
  - View the *Layer Properties…* and get access to metadata and a preview of the data provided by the service. More settings are available when the layer has been loaded into the project.

Configurations can be saved to `.XML` file (*Save Connections*) through the *XYZ* entry in *Data Source Manager* dialog or its contextual menu in the *Browser* panel. Likewise, they can be added from a file (*Load Connections*).

The XML file for OpenStreetMap looks like this:

```
<!DOCTYPE connections>
<qgsXYZTilesConnections version="1.0">
  <xyztiles url="https://tile.openstreetmap.org/{z}/{x}/{y}.png"
   zmin="0" zmax="19" tilePixelRatio="0" password="" name="OpenStreetMap"
   username="" authcfg="" referer=""/>
</qgsXYZTilesConnections>
```

Sfat: Loading XYZ tiles without creating a connection

It is also possible to add XYZ tiles to a project without necessarily storing its connection settings in you user profile (e.g. for a dataset you may need once). In the *Data Source Manager* ► *XYZ* tab, edit any properties in the *Connection Details* group. The *Name* field above should turn into `Custom`. Press *Add* to load the layer in the project. It will be named by default `XYZ Layer`.

Exemple de servicii pentru Plăcuțe XYZ:

- OpenStreetMap Monochrome: *URL*: `http://tiles.wmflabs.org/bw-mapnik/{z}/{x}/{y}.png`, *Min. Zoom Level*: 0, *Max. Zoom Level*: 19.
- Google Maps: *URL*: `https://mt1.google.com/vt/lyrs=m&x={x}&y={y}&z={z}`, *Min. Zoom Level*: 0, *Max. Zoom Level*: 19.
- Open Weather Map Temperature: *URL*: `http://tile.openweathermap.org/map/temp_new/{z}/{x}/{y}.png?appid={api_key}` *Min. Zoom Level*: 0, *Max. Zoom Level*: 19.

Using ArcGIS REST Servers

ArcGIS REST Servers can be added via the *ArcGIS REST Server* tab of the *Data Source Manager* dialog or the contextual menu of the *ArcGIS REST Servers* entry in the *Browser* panel. Press *New* (respectively *New Connection*) and provide:

- a *Name*
- the *URL*
- a *Community endpoint URL*
- a *Content endpoint URL*
- the *authentication* configuration if necessary
- a *Referer*

Notă: ArcGIS Feature Service connections which have their corresponding Portal endpoint URLS set can be explored by content groups in the browser panel.

If a connection has the Portal endpoints set, then expanding out the connection in the browser will show a "Groups" and "Services" folder, instead of the full list of services usually shown. Expanding out the groups folder will show a list of all content groups that the user is a member of, each of which can be expanded to show the service items belonging to that group.

Configurations can be saved to `.XML` file (*Save Connections*) through the *ArcGIS REST Server* entry in *Data Source Manager* dialog. Likewise, they can be added from a file (*Load Connections*).

Once a connection to an ArcGIS REST Server is set, it's possible to:

- *Edit* the ArcGIS REST Server connection settings
- *Remove* the connection
- *Refresh* the connection
- use a filter for the available layers
- choose from a list of available layers with the option to *Only request features overlapping the current view extent*
- From the *Browser* panel, right-click over the connection entry and you can:
  - *Refresh*
  - *Edit connection…*
  - *Remove connection…*
  - *View Service Info* which will open the default web browser and display the Service Info.
- Right-click over the layer entry and you can also:
  - *View Service Info* which will open the default web browser and display the Service Info.
  - *Export layer…* ► *To File*
  - *Add layer to project*: a double-click also adds the layer
  - View the *Layer Properties…* and get access to metadata and a preview of the data provided by the service. More settings are available when the layer has been loaded into the project.

Using 3D tiles services

To load a 3D tiles into QGIS, use the *Scene* tab in the *Data Source Manager* dialog.

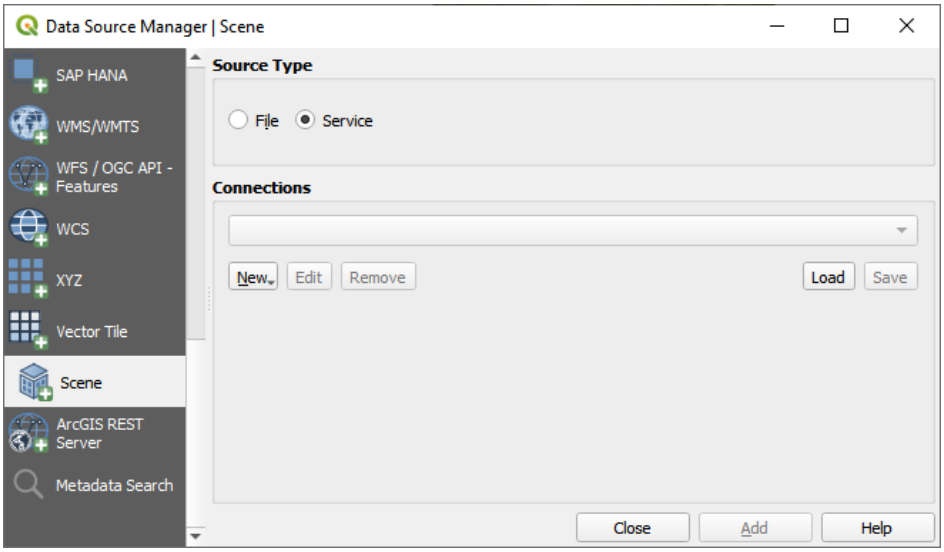


Fig. 15.19: Data Source Manager - Scene

Create a *New Cesium 3D Tiles Connection* by clicking on *New*. Add *Name* and *URL* or add local tileset file.

Support for 3D tiles:

- Remote source - `http://example.com/tileset.json`
- Local files - `file:///path/to/tiles/tileset.json`

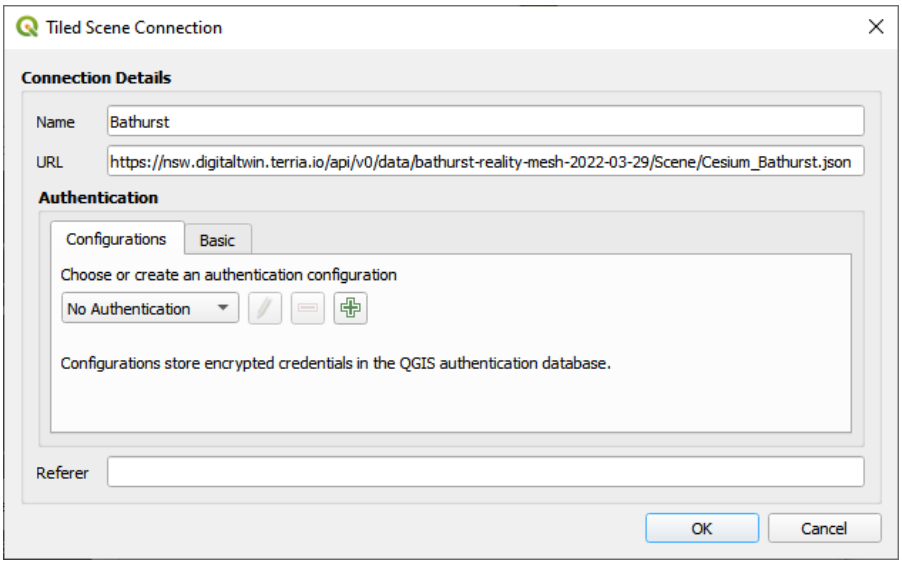


Fig. 15.20: Tiled Scene Connection

You can also add the service from *Browser Panel*.

After creating new connection you are able to *Add* the new layer to your map.

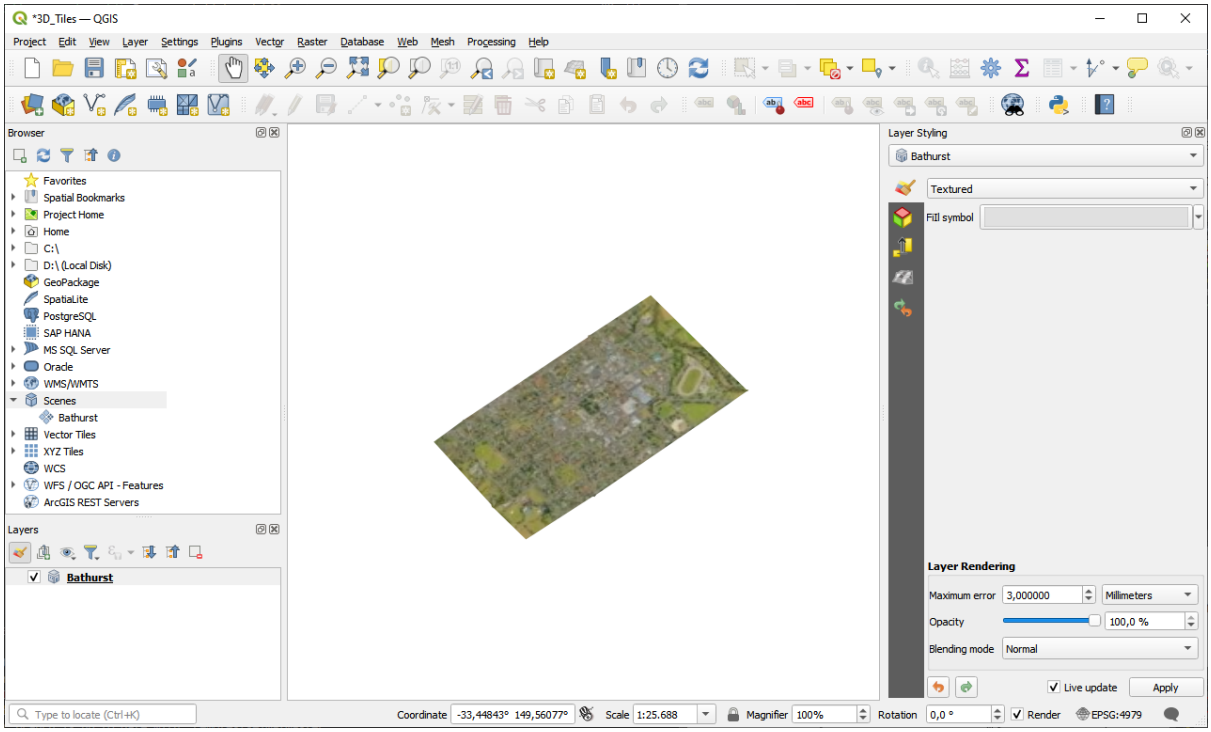


Fig. 15.21: 3D Tiles Layer - Textured

15.2 Crearea Straturilor

Layers can be created in many ways, including:

- empty layers from scratch
- layers from existing layers
- layers from the clipboard
- layers as a result of an SQL-like query based on one or many layers (*virtual layers*)

QGIS also provides tools to import/export from/to different formats.

15.2.1 Creating new vector layers

QGIS allows you to create new layers in different formats. It provides tools for creating GeoPackage, Shapefile, SpatiaLite, GPX format and Temporary Scratch layers (aka memory layers). Creation of a *new GRASS layer* is supported within the GRASS plugin.

Creating a new GeoPackage layer

To create a new GeoPackage layer, press the *New GeoPackage Layer…* button in the *Layer* ► *Create Layer* ► menu or from the *Data Source Manager* toolbar. The *New GeoPackage Layer* dialog will be displayed as shown in Fig. 15.22.

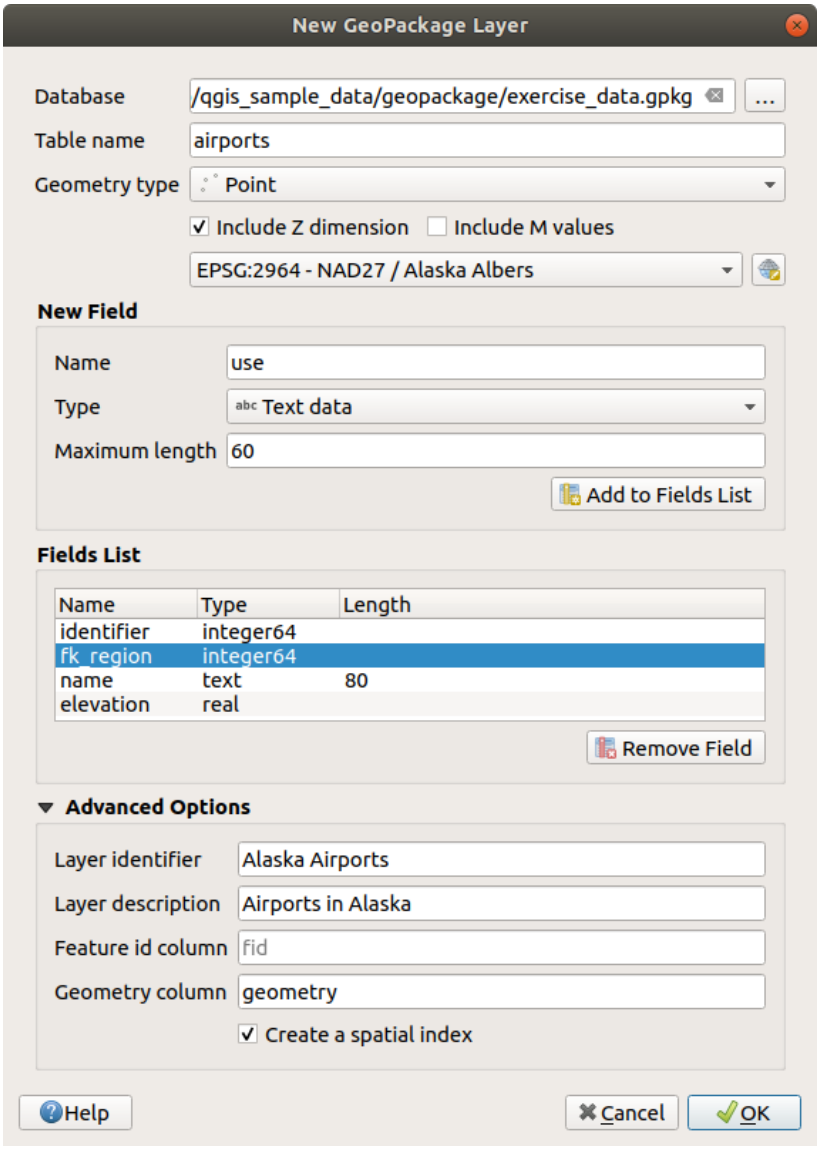


Fig. 15.22: Creating a New GeoPackage layer dialog

1. The first step is to indicate the database file location. This can be done by pressing the … button to the right of the *Database* field and select an existing GeoPackage file or create a new one. QGIS will automatically add the right extension to the name you provide.
2. Give the new layer / table a name (*Table name*)

3. Define the *Geometry type*. If not a geometryless layer, you can specify whether it should *Include Z dimension* and/or *Include M values*.
4. Specify the coordinate reference system using the button

To add fields to the layer you are creating:

1. Enter the *Name* of the field
2. Select the data *Type*. Supported types are *Text data*, *Whole number* (both integer and integer64), *Decimal number*, *Date* and *Date and time*, *Binary (BLOB)* and *Boolean*.
3. Depending on the selected data format, enter the *Maximum length* of values.
4. Click on the *Add to Fields List* button
5. Reproduce the steps above for each field you need to add
6. Once you are happy with the attributes, click *OK*. QGIS will add the new layer to the legend, and you can edit it as described in section *Digitizarea unui strat vectorial existent*.

By default, when creating a GeoPackage layer, QGIS generates a *Feature id column* called `fid` which acts as the primary key of the layer. The name can be changed. The geometry field, if availabe, is named `geometry`, and you can choose to *Create a spatial index* on it. These options can be found under the *Advanced Options* together with the *Layer identifier* (short human readable name of the layer) and the *Layer description*.

Further management of GeoPackage layers can be done with the *DB Manager*.

Crearea unui nou strat de tip Fișier Shape

To create a new ESRI Shapefile format layer, press the *New Shapefile Layer…* button in the *Layer* ► *Create Layer* ► menu or from the *Data Source Manager* toolbar. The *New Shapefile Layer* dialog will be displayed as shown in Fig. 15.23.

1. Provide a path and file name using the … button next to *File name*. QGIS will automatically add the right extension to the name you provide.
2. Next, indicate the *File encoding* of the data
3. Choose the *Geometry type* of the layer: No Geometry (resulting in a `.DBF` format file), point, multipoint, line or polygon
4. Specify whether the geometry should have additional dimensions: *None*, *Z (+ M values)* or *M values*
5. Specify the coordinate reference system using the button

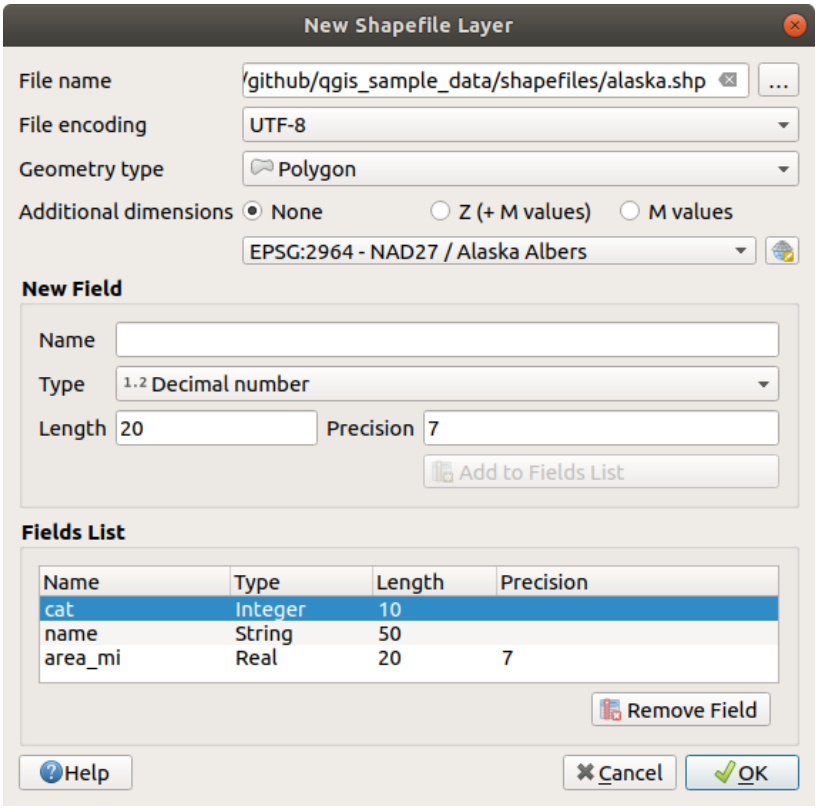


Fig. 15.23: Crearea unui dialog pentru un nou Fișier Shape

To add fields to the layer you are creating:

1. Enter the *Name* of the field
2. Select the data *Type*. Only *Decimal number*, *Whole number*, *Text data* and *Date* attributes are supported.
3. Depending on the selected data format, enter the *Length* and *Precision*.
4. Click on the *Add to Fields List* button
5. Reproduce the steps above for each field you need to add
6. Once you are happy with the attributes, click *OK*. QGIS will add the new layer to the legend, and you can edit it as described in section *Digitizarea unui strat vectorial existent*.

By default, a first integer `id` column is added but can be removed.

Creating a new SpatiaLite layer

To create a new SpatiaLite layer, press the *New SpatiaLite Layer…* button in the *Layer* ► *Create Layer* ► menu or from the *Data Source Manager* toolbar. The *New SpatiaLite Layer* dialog will be displayed as shown in Fig. 15.24.

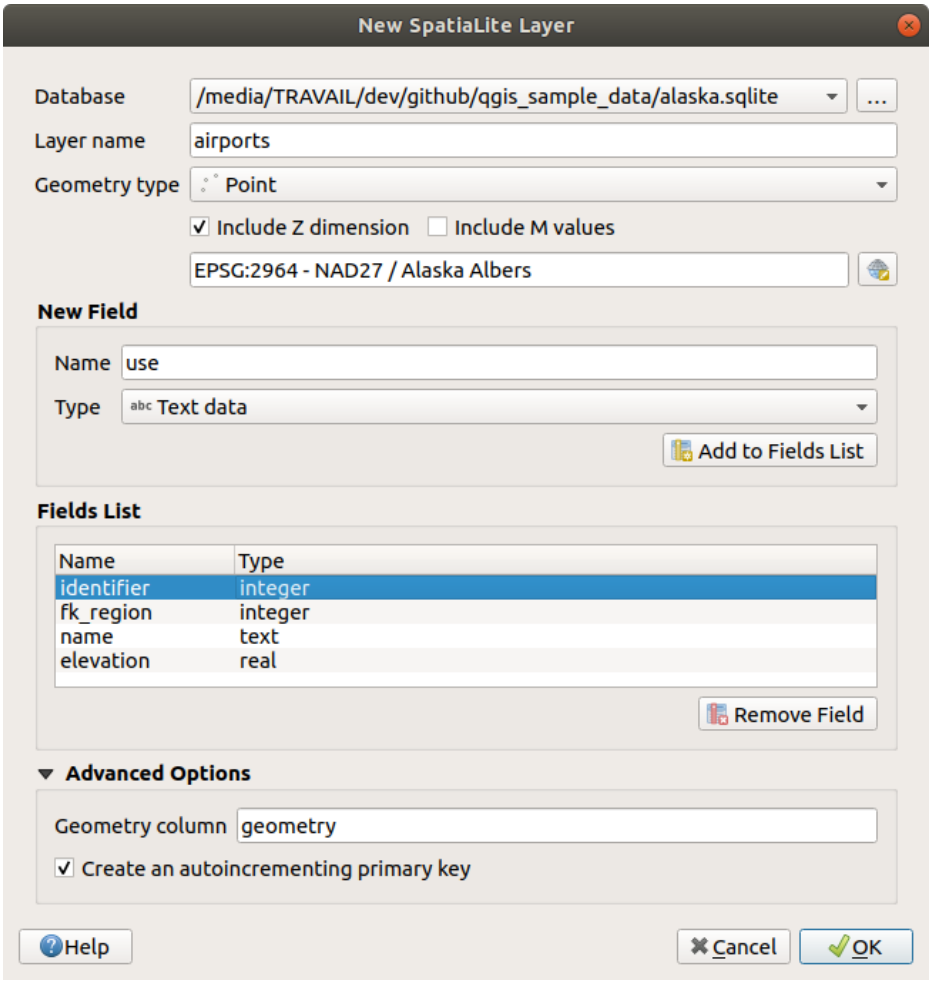


Fig. 15.24: Creating a New SpatiaLite layer dialog

1. The first step is to indicate the database file location. This can be done by pressing the … button to the right of the *Database* field and select an existing SpatiaLite file or create a new one. QGIS will automatically add the right extension to the name you provide.
2. Provide a name (*Layer name*) for the new layer
3. Define the *Geometry type*. If not a geometryless layer, you can specify whether it should *Include Z dimension* and/or *Include M values*.
4. Specify the coordinate reference system using the button.

To add fields to the layer you are creating:

1. Enter the *Name* of the field
2. Select the data *Type*. Supported types are *Text data*, *Whole number*, *Decimal number*, *Date* and *Date time*.
3. Click on the *Add to Fields List* button
4. Reproduce the steps above for each field you need to add

5. Once you are happy with the attributes, click *OK*. QGIS will add the new layer to the legend, and you can edit it as described in section *Digitizarea unui strat vectorial existent*.

If desired, you can select *Create an autoincrementing primary key* under the guilabel:*Advanced Options* section. You can also rename the *Geometry column* (`geometry` by default).

Further management of SpatiaLite layers can be done with *DB Manager*.

Creating a new Mesh layer

To create a new Mesh layer, press the *New Mesh Layer…* button in the *Layer* ► *Create Layer* ► menu or from the *Data Source Manager* toolbar. The *New Mesh Layer* dialog will be displayed as shown in Fig. 15.25.

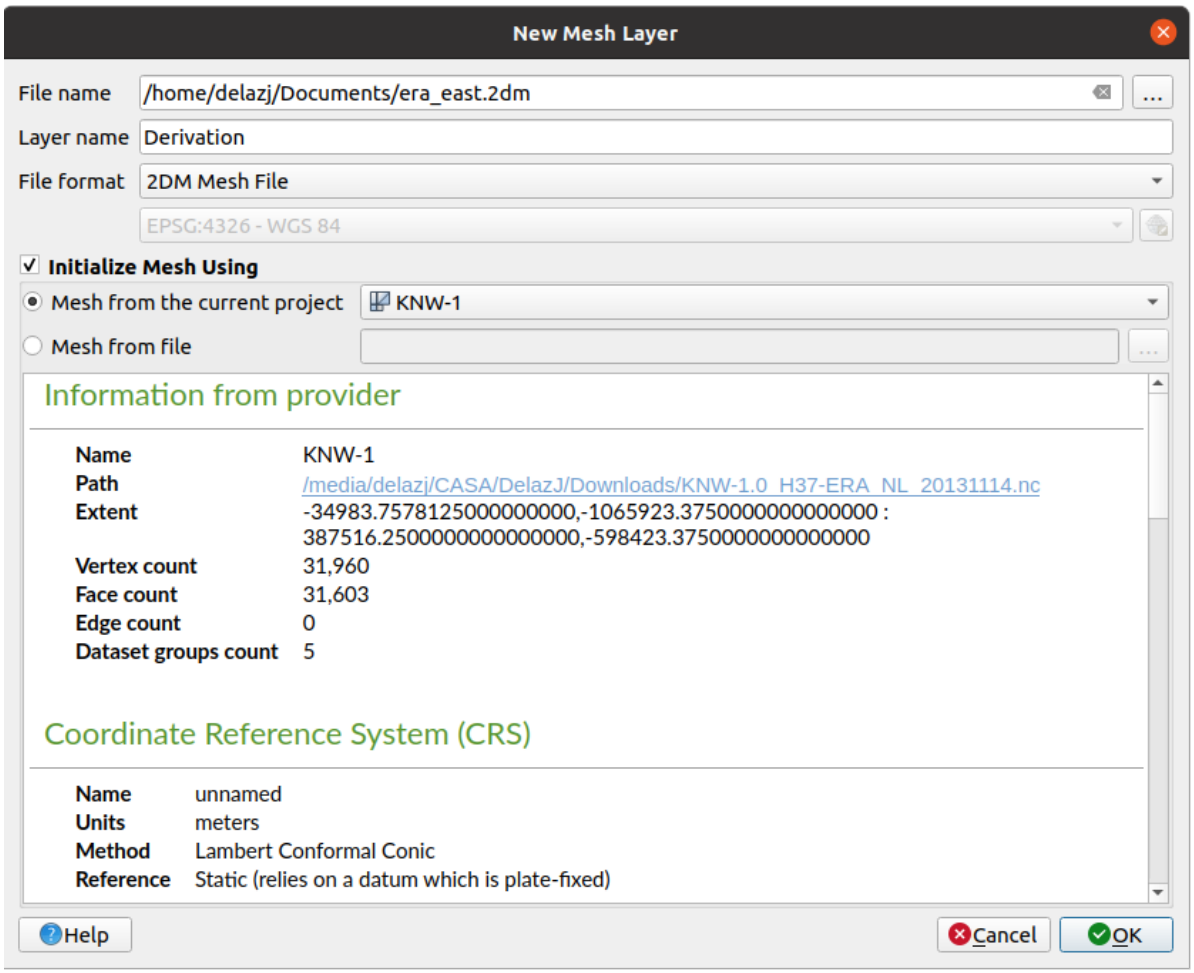


Fig. 15.25: Creating a New Mesh layer dialog

1. The first step is to indicate the mesh file location. This can be done by pressing the … button to the right of the *File name* field and select an existing mesh file or create a new one.
2. Provide a name (*Layer name*), i.e. the name the layer is displayed with in the *Layers* panel
3. Select the *File format*: currently supported mesh file formats are `2DM Mesh File (*.2dm)`, `Selafin File (*.slf)` and `UGRID (*.nc)`.
4. Indicate the *Coordinate Reference System* to assign to the dataset
5. The above steps will generate an empty layer that you can afterwards digitize vertices and add dataset groups to. It's however also possible to initialize the layer with an existing mesh layer, i.e. populate the new layer with vertices or faces from the other. To do so:

1. Check *Initialize Mesh using*
2. and select either a *Mesh from the current project* or *Mesh from a file*. Informations on the selected mesh file are displayed for checkup.

Note that only the frame of the mesh layer is transferred to the new layer; their datasets are not copied.

Creating a new GPX layer

To create a new GPX file:

1. Select *Create Layer* ► *New GPX Layer…* from the *Layer* menu.
2. In the dialog, choose where to save the new file, name it and press *Save*.
3. Three new layers are added to the *Layers Panel*:
   - a point layer to digitize locations (`waypoints`) with fields storing the name, elevation, comment, description, source, url and url name
   - a line layer to digitize sequences of locations that make up a planned route (`routes`) with fields storing the name, symbol, number, comment, description, source, url, url name
   - and a line layer to track the receiver's movement over time (`tracks`) with fields storing the name, symbol, number, comment, description, source, url, url name.
4. You can now edit any of them as described in section *Digitizarea unui strat vectorial existent*.

Creating a new Temporary Scratch Layer

Temporary Scratch Layers are in-memory layers, meaning that they are not saved on disk and will be discarded when QGIS is closed. They can be handy for storing features you temporarily need or as intermediate layers during geoprocessing operations.

To create a new Temporary Scratch layer, choose the *New Temporary Scratch Layer…* entry in the *Layer* ► *Create Layer* ► menu or in the *Data Source Manager* toolbar. The *New Temporary Scratch Layer* dialog will be displayed as shown in Fig. 15.26. Then:

1. Provide the *Layer name*
2. Select the *Geometry type*. Here you can create a:
   - `No geometry` type layer, served as simple table,
   - `Point` or `MultiPoint` layer,
   - `LineString/CompoundCurve` or `MultiLineString/MultiCurve` layer,
   - `Polygon/CurvePolygon` or `MultiPolygon/MultiSurface` layer.
3. For geometric types, specify the dimensions of the dataset: check whether it should *Include Z dimension* and/or *Include M values*
4. Specify the coordinate reference system using the button.
5. Add fields to the layer. Note that unlike many formats, temporary layers can be created without any fields. This step is thus optional.
   1. Enter the *Name* of the field
   2. Select the data *Type*: *Text*, *Whole number*, *Decimal number*, *Boolean*, *Date*, *Time*, *Date & Time* and *Binary (BLOB)* are supported.
   3. Depending on the selected data format, enter the *Length* and *Precision*
   4. Click on the *Add to Fields List* button

5. Repeat the steps above for each field you need to add

6. Once you are happy with the settings, click *OK*. QGIS will add the new layer to the *Layers* panel, and you can edit it as described in section *Digitizarea unui strat vectorial existent*.

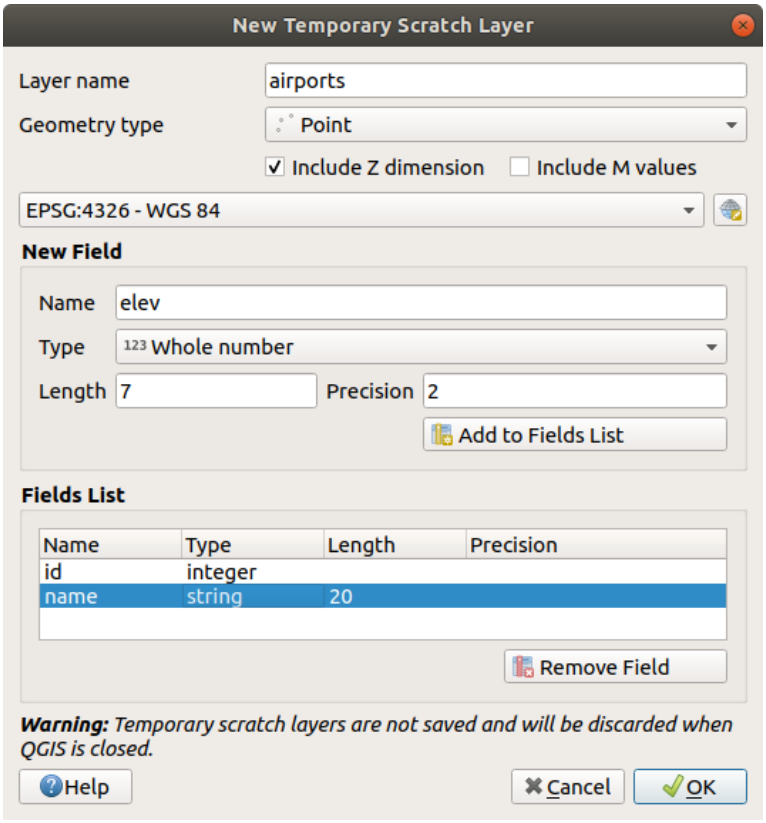


Fig. 15.26: Creating a new Temporary Scratch layer dialog

You can also create prepopulated temporary scratch layers using e.g. the clipboard (see *Creating new layers from the clipboard*) or as a result of a *Processing algorithm*.

Sfat: Permanently store a memory layer on disk

To avoid data loss when closing a project with temporary scratch layers, you can save these layers to any vector format supported by QGIS:

- clicking the indicator icon next to the layer;
- selecting the *Make permanent* entry in the layer contextual menu;
- using the *Export* ► entry from the contextual menu or the *Layer* ► *Save As…* menu.

Each of these commands opens the *Save Vector Layer as* dialog described in the *Creating new layers from an existing layer* section and the saved file replaces the temporary one in the *Layers* panel.

15.2.2 Creating new layers from an existing layer

Both raster and vector layers can be saved in a different format and/or reprojected to a different coordinate reference system (CRS) using the *Layer* ► *Save As…* menu or right-clicking on the layer in the *Layers panel* and selecting:

- *Export* ► *Save As…* for raster layers
- *Export* ► *Save Features As…* or *Export* ► *Save Selected Features As…* for vector layers.
- Drag and drop the layer from the layer tree to the PostGIS entry in the *Browser Panel*. Note that you must have a PostGIS connection in the *Browser Panel*.

Parametri comuni

The *Save Layer as…* dialog shows several parameters to change the behavior when saving the layer. Among the common parameters for raster and vector are:

- *File name*: the location of the file on the disk. It can refer to the output layer or to a container that stores the layer (for example database-like formats such as GeoPackage, SpatiaLite or Open Document Spreadsheets).
- *CRS*: can be changed to reproject the data
- *Extent*: restricts the extent of the input that is to be exported using the *extent_selector* widget
- *Add saved file to map*: to add the new layer to the canvas

However, some parameters are specific to raster and vector formats:

Raster specific parameters

Depending on the format of export, some of these options may not be available:

- *Output mode* (it can be **raw data** or **rendered image**)
- *Format*: exports to any raster format GDAL can write to, such as GeoTiff, GeoPackage, MBTiles, Geospatial PDF, SAGA GIS Binary Grid, Intergraph Raster, ESRI .hdr Labelled…
- *Rezoluție*
- *Create Options*: use advanced options (file compression, block sizes, colorimetry…) when generating files, either from the *predefined create profiles* related to the output format or by setting each parameter.
- *Pyramids* creation
- *VRT Tiles* in case you opted to *Create VRT*
- *No data values*

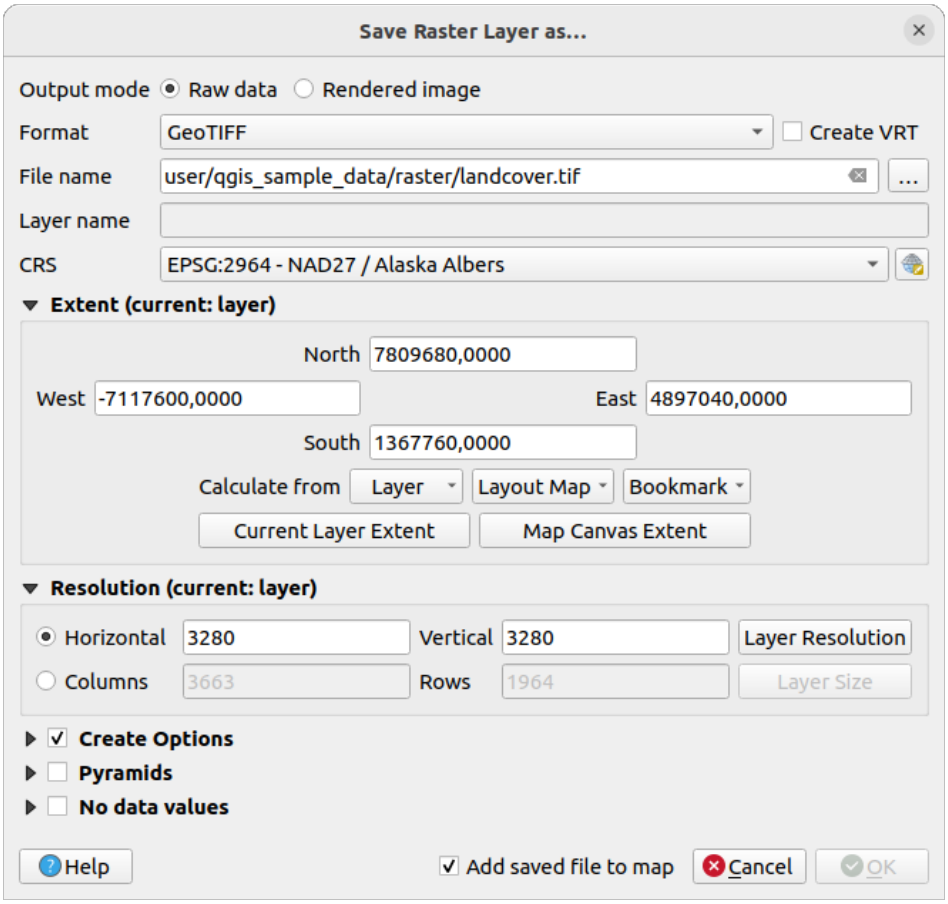


Fig. 15.27: Saving as a new raster layer

Vector specific parameters

Depending on the format of export, some of these options may be available:

- *Format*: exports to any vector format GDAL can write to, such as GeoPackage, GML, ESRI Shapefile, AutoCAD DXF, ESRI FileGDB, Mapinfo TAB or MIF, SpatiaLite, CSV, KML, ODS, …
- *Layer name*: available when the *File name* refers to a container-like format, this entry represents the output layer.
- *Encoding*
- *Save only selected features*
- *Select fields to export and their export options*: provides means to export fields with custom names and *form widget* settings:
  - Check rows under the *Name* column to choose fields to keep in the output layer, or press *Select All* or *Deselect All* buttons
  - Toggle the *Use aliases for exported name* checkbox to populate the *Export name* column with corresponding field aliases or reset to the original field name. Double-clicking a cell will also edit the name.
  - Depending on whether attribute form custom widgets are in use, you can *Replace all selected raw field values by displayed values*. E.g. if a `value map` widget is applied to a field, the output layer will contain the description values instead of the original values. The replacement can also be done on a field by field basis, in the *Replace with displayed values* column.
- *Persist layer metadata*: ensures that any layer *metadata* present in the source layer will be copied and stored:

  - in the newly created layer, if the output is of GeoPackage format
  - as a `.qmd` file along with the output layer, for other formats. Note that file-based formats supporting more than one dataset (e.g. SpatiaLite, DXF,…) may have unintended behavior.
- *Symbology export*: can be used mainly for DXF export and for all file formats who manage OGR feature styles (see note below) as DXF, KML, tab file formats:
  - **No symbology**: default style of the application that reads the data
  - **Feature symbology**: save style with OGR Feature Styles (see note below)
  - **Symbol Layer symbology**: save with OGR Feature Styles (see note below) but export the same geometry multiple times if there are multiple symbology symbol layers used
  - A **Scale** value can be applied to the latest options

Notă: *OGR Feature Styles* are a way to store style directly in the data as a hidden attribute. Only some formats can handle this kind of information. KML, DXF and TAB file formats are such formats. For advanced details, you can read the OGR Feature Styles specification document.

- *Geometry*: you can configure the geometry capabilities of the output layer
  - *geometry type*: keeps the original geometry of the features when set to **Automatic**, otherwise removes or overrides it with any type. You can add an empty geometry column to an attribute table and remove the geometry column of a spatial layer.
  - *Force multi-type*: forces creation of multi-geometry features in the layer.
  - *Include z-dimension* to geometries.

Sfat: Overriding layer geometry type makes it possible to do things like save a geometryless table (e.g. `.csv` file) into a shapefile WITH any type of geometry (point, line, polygon), so that geometries can then be manually added to rows with the Add Part tool.

- *Datasource Options*, *Layer Options* or *Custom Options* which allow you to configure advanced parameters depending on the output format. Some are described in *Explorarea Formatelor de Date și a Câmpurilor* but for full details, see the GDAL driver documentation. Each file format has its own custom parameters, e.g. for the `GeoJSON` format have a look at the GDAL GeoJSON documentation.

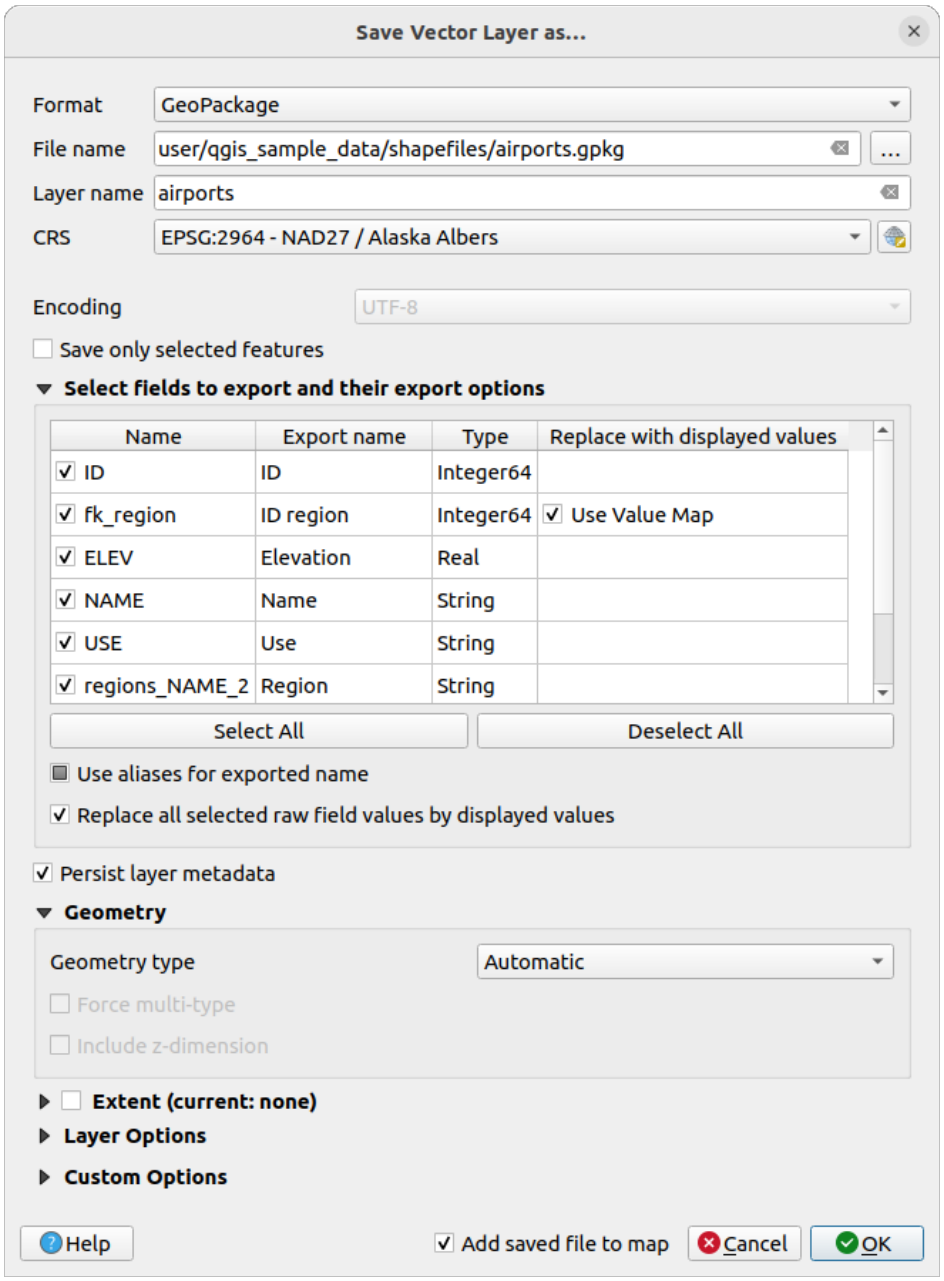


Fig. 15.28: Saving as a new vector layer

When saving a vector layer into an existing file, depending on the capabilities of the output format (Geopackage, SpatiaLite, FileGDB…), the user can decide whether to:

- overwrite the whole file
- overwrite only the target layer (the layer name is configurable)
- append features to the existing target layer
- append features, add new fields if there are any.

For formats like ESRI Shapefile, MapInfo .tab, feature append is also available.

15.2.3 Crearea fișierelor DXF noi

Besides the *Save As...* dialog which provides options to export a single layer to another format, including `*.DXF`, QGIS provides another tool to export multiple layers as a single DXF layer. It's accessible in the *Project* ► *Import/Export* ► *Export Project to DXF...* menu.

In the *DXF Export* dialog:

1. Provide the destination file.
2. Choose the symbology mode and scale (see the *OGR Feature Styles* note), if applicable.
3. Select the data *Encoding*.
4. Select the *CRS* to apply: the selected layers will be reprojected to the given CRS.
5. Select the layers to include in the DXF files either by checking them in the table widget or automatically picking them from an existing *map theme*. The *Select All* and *Deselect All* buttons can help to quickly set the data to export.

   For each layer, you can choose whether to export all the features in a single DXF layer or rely on a field whose values are used to split the features into layers in the DXF output.

Optionally, you can also choose to:

- *Use the layer title as name if set* instead of the layer name itself;
- *Export features intersecting the current map extent*;
- *Force 2d output (eg. to support polyline width)*;
- *Export label as MTEXT elements* or TEXT elements.

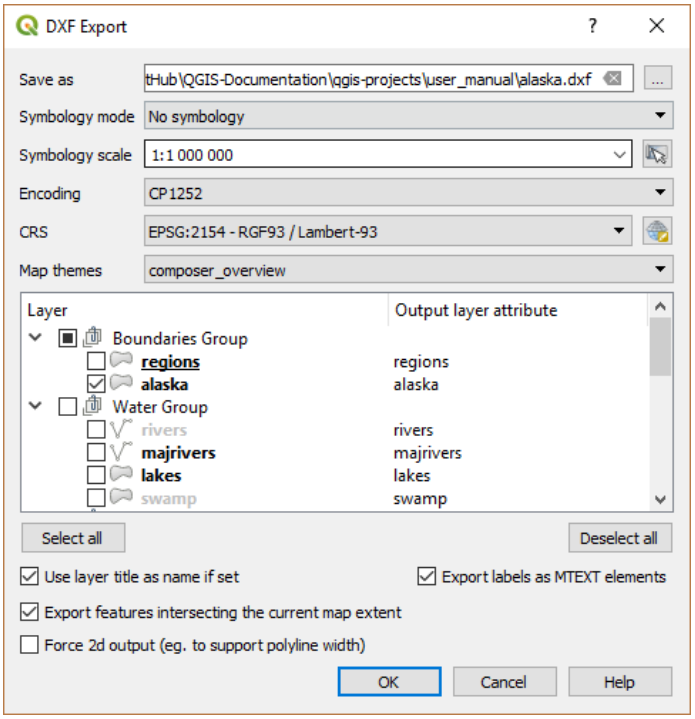


Fig. 15.29: Exporting a project to DXF dialog

15.2.4 Creating new layers from the clipboard

Features that are on the clipboard can be pasted into a new layer. To do this, Select some features, copy them to the clipboard, and then paste them into a new layer using *Edit* ► *Paste Features as* ► and choosing:

- *New Vector Layer…*: the *Save vector layer as…* dialog appears (see *Creating new layers from an existing layer* for parameters)
- or *Temporary Scratch Layer…*: you need to provide a name for the layer

A new layer, filled with selected features and their attributes is created (and added to map canvas).

Notă: Creating layers from the clipboard is possible with features selected and copied within QGIS as well as features from another application, as long as their geometries are defined using well-known text (WKT).

15.2.5 Creating virtual layers

A virtual layer is a special kind of vector layer. It allows you to define a layer as the result of an SQL query involving any number of other vector layers that QGIS is able to open. Virtual layers do not carry data by themselves and can be seen as views.

To create a virtual layer, open the virtual layer creation dialog by:

- choosing the *Add/Edit Virtual Layer* entry in the *Layer* ► *Add Layer* ► menu;
- enabling the *Add Virtual Layer* tab in the *Data Source Manager* dialog;
- or using the *DB Manager* dialog tree.

The dialog allows you to specify a *Layer name* and an SQL *Query*. The query can use the name (or id) of loaded vector layers as tables, as well as their field names as columns.

For example, if you have a layer called `airports`, you can create a new virtual layer called `public_airports` with an SQL query like:

```
SELECT *
FROM airports
WHERE USE = "Civilian/Public"
```

The SQL query will be executed, regardless of the underlying provider of the `airports` layer, even if this provider does not directly support SQL queries.

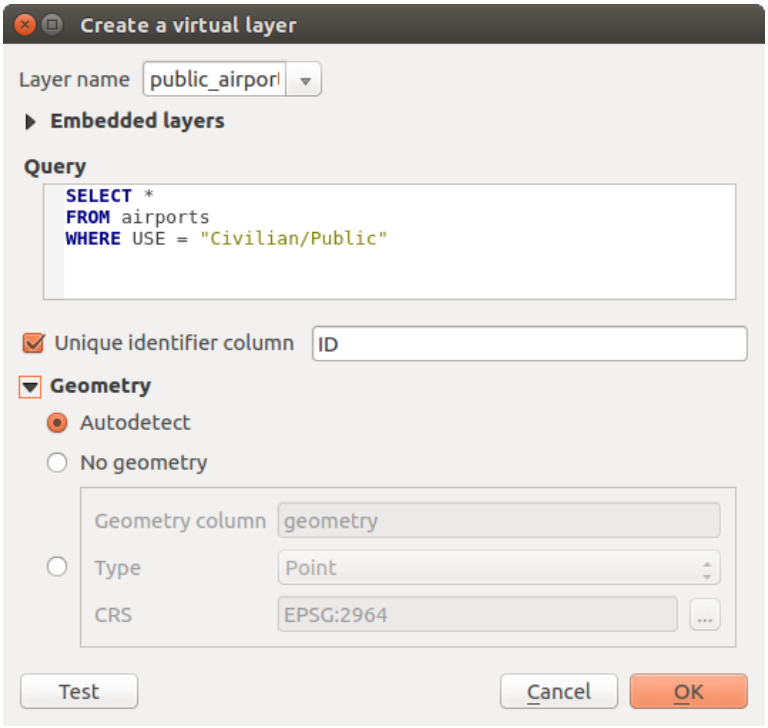


Fig. 15.30: Create virtual layers dialog

Joins and complex queries can also be created, for example, to join airports and country information:

```
SELECT airports.*, country.population
FROM airports
JOIN country
ON airports.country = country.name
```

Notă: It's also possible to create virtual layers using the SQL window of *DB Manager Plugin*.

Embedding layers for use in queries

Besides the vector layers available in the map canvas, the user can add layers to the *Embedded layers* list, which can be used in queries without the need to have them showing in the map canvas or Layers panel.

To embed a layer, click *Add* and provide the *Local name*, *Provider*, *Encoding* and the path to the *Source*.

The *Import* button allows adding layers in the map canvas into the Embedded layers list. Those layers can then be removed from the Layers panel without breaking existent queries.

Supported query language

The underlying engine uses SQLite and SpatiaLite to operate.

It means you can use all of the SQL your local installation of SQLite understands.

Functions from SQLite and spatial functions from SpatiaLite can also be used in a virtual layer query. For instance, creating a point layer out of an attribute-only layer can be done with a query similar to:

```
SELECT id, MakePoint(x, y, 4326) as geometry
FROM coordinates
```

Functions of QGIS expressions can also be used in a virtual layer query.

To refer the geometry column of a layer, use the name `geometry`.

Contrary to a pure SQL query, all the fields of a virtual layer query must be named. Don't forget to use the `as` keyword to name your columns if they are the result of a computation or a function call.

Probleme legate de performanță

With default parameters, the virtual layer engine will try its best to detect the type of the different columns of the query, including the type of the geometry column if one is present.

This is done by introspecting the query when possible or by fetching the first row of the query (LIMIT 1) as a last resort. Fetching the first row of the result just to create the layer may be undesirable for performance reasons.

The creation dialog parameters:

- *Unique identifier column*: specifies a field of the query that represents unique integer values that QGIS can use as row identifiers. By default, an autoincrementing integer value is used. Defining a unique identifier column speeds up the selection of rows by id.
- *No geometry*: forces the virtual layer to ignore any geometry field. The resulting layer is an attribute-only layer.
- Geometry *Column*: specifies the name of the geometry column.
- Geometry *Type*: specifies the type of the geometry.
- Geometry *CRS*: specifies the coordinate reference system of the virtual layer.

Special comments

The virtual layer engine tries to determine the type of each column of the query. If it fails, the first row of the query is fetched to determine column types.

The type of a particular column can be specified directly in the query by using some special comments.

The syntax is the following: `/*:type*/`. It has to be placed just after the name of a column. `type` can be either `int` for integers, `real` for floating point numbers or `text`.

For instance:

```
SELECT id+1 as nid /*:int*/
FROM table
```

The type and coordinate reference system of the geometry column can also be set thanks to special comments with the following syntax `/*:gtype:srid*/` where `gtype` is the geometry type (`point`, `linestring`, `polygon`, `multipoint`, `multilinestring` or `multipolygon`) and `srid` an integer representing the EPSG code of a coordinate reference system.

Use of indexes

When requesting a layer through a virtual layer, the source layer indices will be used in the following ways:

- if an = predicate is used on the primary key column of the layer, the underlying data provider will be asked for a particular id (FilterFid)
- for any other predicates (>, <=, !=, etc.) or on a column without a primary key, a request built from an expression will be used to request the underlying vector data provider. It means indexes may be used on database providers if they exist.

A specific syntax exists to handle spatial predicates in requests and triggers the use of a spatial index: a hidden column named `_search_frame_` exists for each virtual layer. This column can be compared for equality to a bounding box. Example:

```
SELECT *
FROM vtab
WHERE _search_frame_=BuildMbr(-2.10,49.38,-1.3,49.99,4326)
```

Spatial binary predicates like `ST_Intersects` are sped up significantly when used in conjunction with this spatial index syntax.

15.3 Explorarea Formatelor de Date și a Câmpurilor

15.3.1 Datele raster

GIS raster data are matrices of discrete cells that represent features / phenomena on, above or below the earth's surface. Each cell in the raster grid has the same size, and cells are usually rectangular (in QGIS they will always be rectangular). Typical raster datasets include remote sensing data, such as aerial photography, or satellite imagery and modelled data, such as elevation or temperature.

Unlike vector data, raster data typically do not have an associated database record for each cell. They are geocoded by pixel resolution and the X/Y coordinate of a corner pixel of the raster layer. This allows QGIS to position the data correctly on the map canvas.

The GeoPackage format is convenient for storing raster data when working with QGIS. The popular and powerful GeoTiff format is a good alternative.

QGIS makes use of georeference information inside the raster layer (e.g., GeoTiff) or an associated *world file* to properly display the data.

15.3.2 Datele Vectoriale

Many of the features and tools available in QGIS work the same, regardless the vector data source. However, because of the differences in format specifications (GeoPackage, ESRI Shapefile, MapInfo and MicroStation file formats, AutoCAD DXF, PostGIS, SpatiaLite, Oracle Spatial, MS SQL Server, SAP HANA Spatial databases and many more), QGIS may handle some of their properties differently. Support is provided by the GDAL vector drivers. This section describes how to work with these specifics.

Notă: QGIS supports (multi)point, (multi)line, (multi)polygon, CircularString, CompoundCurve, CurvePolygon, MultiCurve, MultiSurface feature types, all optionally with Z and/or M values.

You should also note that some drivers don't support some of these feature types, like CircularString, CompoundCurve, CurvePolygon, MultiCurve, MultiSurface feature type. QGIS will convert them.

GeoPackage

The GeoPackage (GPKG) format is platform-independent, and is implemented as a SQLite database container, and can be used to store both vector and raster data. The format was defined by the Open Geospatial Consortium (OGC), and was published in 2014.

GeoPackage can be used to store the following in a SQLite database:

- **vector** features
- **tile matrix sets of imagery** and **raster** maps
- attributes (non-spatial data)
- extensions

Since QGIS version 3.8, GeoPackage can also store QGIS projects. GeoPackage layers can have JSON fields.

GeoPackage is the default format for vector data in QGIS.

ESRI Shapefile format

The ESRI Shapefile format is still one of the most used vector file formats, even if it has some limitations compared to for instance GeoPackage and SpatiaLite.

An ESRI Shapefile format dataset consists of several files. The following three are required:

1. Fișierul `.shp` care conține geometriile entității
2. `.dbf` file containing the attributes in dBase format
3. `.shx` index file

An ESRI Shapefile format dataset can also include a file with a `.prj` suffix, which contains projection information. While it is very useful to have a projection file, it is not mandatory. A Shapefile format dataset can contain additional files. For further details, see the the ESRI technical specification.

GDAL has read-write support for compressed ESRI Shapefile format (`shz` and `shp.zip`).

Improving Performance for ESRI Shapefile format datasets

To improve the drawing performance for an ESRI Shapefile format dataset, you can create a spatial index. A spatial index will improve the speed of both zooming and panning. Spatial indexes used by QGIS have a `.qix` extension.

Use these steps to create the index:

1. Load an ESRI Shapefile format dataset (see *Panoul Navigatorului*)
2. Open the *Layer Properties* dialog by double-clicking on the layer name in the legend or by right-clicking and choosing *Properties...* from the context menu
3. In the *Source* tab, click the *Create Spatial Index* button

Problem loading a .prj file

If you load an ESRI Shapefile format dataset with a `.prj` file and QGIS is not able to read the coordinate reference system from that file, you will need to define the proper projection manually in the *Layer Properties* ► *Source* tab of the layer by clicking the Select CRS button. This is due to the fact that `.prj` files often do not provide the complete projection parameters as used in QGIS and listed in the *CRS* dialog.

For the same reason, if you create a new ESRI Shapefile format dataset with QGIS, two different projection files are created: a `.prj` file with limited projection parameters, compatible with ESRI software, and a `.qpj` file, providing all the parameters of the CRS. Whenever QGIS finds a `.qpj` file, it will be used instead of the `.prj`.

Fișiere cu Text Delimitat

Delimited text files are very common and widely used because of their simplicity and readability – data can be viewed and edited in a plain text editor. A delimited text file is tabular data with columns separated by a defined character and rows separated by line breaks. The first row usually contains the column names. A common type of delimited text file is a CSV (Comma Separated Values), with columns separated by commas. Delimited text files can also contain positional information (see *Storing geometry information in delimited text files*).

QGIS allows you to load a delimited text file as a layer or an ordinary table (see *Panoul Navigatorului* or *Importarea unui fișier cu texte delimitate*). First check that the file meets the following requirements:

1. The file must have a delimited header row of field names. This must be the first line of the data (ideally the first row in the text file).
2. If geometry should be enabled, the file must contain field(s) that define the geometry. These field(s) can have any name.

3. The X and Y coordinates fields (if geometry is defined by coordinates) must be specified as numbers. The coordinate system is not important.
4. If you have a CSV file with non-string columns, you can have an accompanying CSVT file (see section *Using CSVT file to control field formatting*).

The elevation point data file `elevp.csv` in the QGIS sample dataset (see section *Descărcarea datelor demonstrative*) is an example of a valid text file:

```
X;Y;ELEV
-300120;7689960;13
-654360;7562040;52
1640;7512840;3
[...]
```

Some things to note about the text file:

1. The example text file uses `;` (semicolon) as delimiter (any character can be used to delimit the fields).
2. The first row is the header row. It contains the fields `X`, `Y` and `ELEV`.
3. No quotes (`"`) are used to delimit text fields
4. The X coordinates are contained in the `X` field
5. The Y coordinates are contained in the `Y` field

Storing geometry information in delimited text files

Delimited text files can contain geometry information in two main forms:

- As coordinates in separate columns (eg. `Xcol`, `Ycol`...), for point geometry data;
- As well-known text (WKT) representation of geometry in a single column, for any geometry type.

Features with curved geometries (CircularString, CurvePolygon and CompoundCurve) are supported. Here are some examples of geometry types in a delimited text file with geometries coded as WKT:

```
Label;WKT_geom
LineString;LINESTRING(10.0 20.0, 11.0 21.0, 13.0 25.5)
CircularString;CIRCULARSTRING(268 415,227 505,227 406)
CurvePolygon;CURVEPOLYGON(CIRCULARSTRING(1 3, 3 5, 4 7, 7 3, 1 3))
CompoundCurve;COMPOUNDCURVE((5 3, 5 13), CIRCULARSTRING(5 13, 7 15,
  9 13), (9 13, 9 3), CIRCULARSTRING(9 3, 7 1, 5 3))
```

Delimited text files also support Z and M coordinates in geometries:

```
LINESTRINGZ(10.0 20.0 30.0, 11.0 21.0 31.0, 11.0 22.0 30.0)
```

Using CSVT file to control field formatting

When loading CSV files, the GDAL driver assumes all fields are strings (i.e. text) unless it is told otherwise. You can create a CSVT file to tell GDAL (and QGIS) the data type of the different columns:

Tipul	Nume	Exemplu
Număr întreg	Întreg	4
Boolean	Integer(Boolean)	true
Număr zecimal	Real	3.456
Data	Date (YYYY-MM-DD)	2016-07-28
Time	Time (HH:MM:SS+nn)	18:33:12+00
Date & Time	DateTime (YYYY-MM-DD HH:MM:SS+nn)	2016-07-28 18:33:12+00
CoordX	CoordX	8.8249
CoordY	CoordY	47.2274
Point(X)	Point(X)	8.8249
Point(Y)	Point(Y)	47.2274
WKT	WKT	POINT(15 20)

The CSVT file is a **ONE line** plain text file with the data types in quotes and separated by commas, e.g.:

```
"Integer","Real","String"
```

You can even specify width and precision of each column, e.g.:

```
"Integer(6)","Real(5.5)","String(22)"
```

This file is saved in the same folder as the `.csv` file, with the same name, but `.csvt` as the extension.

You can find more information at GDAL CSV Driver.

Sfat: Detect Field Types

Instead of using a CSVT file to tell the data types, QGIS provides the possibility to automatically detect the field types and to change the assumed field types.

Straturi PostGIS

PostGIS layers are stored in a PostgreSQL database. The advantages of PostGIS are spatial indexing, filtering and querying capabilities. Using PostGIS, vector functions such as select and identify work more accurately than they do with GDAL layers in QGIS.

Sfat: Straturi PostGIS

Normally, a PostGIS layer is identified by an entry in the geometry_columns table. QGIS can load layers that do not have an entry in the geometry_columns table. This includes both tables and views. Refer to your PostgreSQL manual for information on creating views.

This section contains some details on how QGIS accesses PostgreSQL layers. Most of the time, QGIS should simply provide you with a list of database tables that can be loaded, and it will load them on request. However, if you have trouble loading a PostgreSQL table into QGIS, the information below may help you understand QGIS messages and give you directions for modifying the PostgreSQL table or view definition to allow QGIS to load it.

Notă: A PostgreSQL database can also store QGIS projects.

Cheie primară

QGIS requires that PostgreSQL layers contain a column that can be used as a unique key for the layer. For tables, this usually means that the table needs a primary key, or a column with a unique constraint on it. In QGIS, this column needs to be of type int4 (an integer of size 4 bytes). Alternatively, the ctid column can be used as primary key. If a table lacks these items, the oid column will be used instead. Performance will be improved if the column is indexed (note that primary keys are automatically indexed in PostgreSQL).

QGIS offers a checkbox **Select at id** that is activated by default. This option gets the ids without the attributes, which is faster in most cases.

Vizualizare

If the PostgreSQL layer is a view, the same requirement exists, but views do not always have primary keys or columns with unique constraints on them. You have to define a primary key field (has to be integer) in the QGIS dialog before you can load the view. If a suitable column does not exist in the view, QGIS will not load the layer. If this occurs, the solution is to alter the view so that it does include a suitable column (a type of integer and either a primary key or with a unique constraint, preferably indexed).

As for table, a checkbox **Select at id** is activated by default (see above for the meaning of the checkbox). It can make sense to disable this option when you use expensive views.

Notă: PostgreSQL foreign table

PostgreSQL foreign tables are not explicitely supported by the PostgreSQL provider and will be handled like a view.

QGIS layer_style table and database backup

If you want to make a backup of your PostGIS database using the `pg_dump` and `pg_restore` commands, and the default layer styles as saved by QGIS fail to restore afterwards, you need to set the XML option to `DOCUMENT` before the restore command:

1. Make a PLAIN backup of the `layer_style` table
2. Open the file within a text editor
3. Change the line `SET xmloption = content;` into `SET XML OPTION DOCUMENT;`
4. Save the file
5. Use psql to restore the table in the new database

Filter database side

QGIS allows to filter features already on server side. Check *Settings* ► *Options* ► *Data Sources* ► *Execute expressions on server-side if possible* to do so. Only supported expressions will be sent to the database. Expressions using unsupported operators or functions will gracefully fallback to local evaluation.

Support of PostgreSQL data types

Data types supported by the PostgreSQL provider include: integer, float, boolean, binary object, varchar, geometry, timestamp, array, hstore and json.

Importing Data into PostgreSQL

Data can be imported into PostgreSQL/PostGIS using several tools, including the DB Manager plugin and the command line tools shp2pgsql and ogr2ogr.

DB Manager

QGIS comes with a core plugin named DB Manager. It can be used to load data, and it includes support for schemas. See section *DB Manager Plugin* for more information.

shp2pgsql

PostGIS includes a utility called **shp2pgsql**, that can be used to import Shapefile format datasets into a PostGIS-enabled database. For example, to import a Shapefile format dataset named `lakes.shp` into a PostgreSQL database named `gis_data`, use the following command:

```
shp2pgsql -s 2964 lakes.shp lakes_new | psql gis_data
```

This creates a new layer named `lakes_new` in the `gis_data` database. The new layer will have a spatial reference identifier (SRID) of 2964. See section *Lucrul cu Proiecții* for more information about spatial reference systems and projections.

Sfat: Exporting datasets from PostGIS

There is also a tool for exporting PostGIS datasets to Shapefile format: **pgsql2shp**. It is shipped within your PostGIS distribution.

---

ogr2ogr

In addition to **shp2pgsql** and **DB Manager**, there is another tool for feeding geographical data in PostGIS: **ogr2ogr**. It is part of your GDAL installation.

To import a Shapefile format dataset into PostGIS, do the following:

```
ogr2ogr -f "PostgreSQL" PG:"dbname=postgis host=myhost.de user=postgres
password=topsecret" alaska.shp
```

This will import the Shapefile format dataset `alaska.shp` into the PostGIS database *postgis* using the user *postgres* with the password *topsecret* on the host server *myhost.de*.

Note that GDAL must be built with PostgreSQL to support PostGIS. You can verify this by typing (in):

```
ogrinfo --formats | grep -i post
```

If you prefer to use the PostgreSQL's **COPY** command instead of the default **INSERT INTO** method, you can export the following environment variable (at least available on and):

```
export PG_USE_COPY=YES
```

ogr2ogr does not create spatial indexes like **shp2pgsl** does. You need to create them manually, using the normal SQL command **CREATE INDEX** afterwards, as an extra step (as described in the next section *Îmbunătățirea Performanței*).

Îmbunătățirea Performanței

Retrieving features from a PostgreSQL database can be time-consuming, especially over a network. You can improve the drawing performance of PostgreSQL layers by ensuring that a PostGIS spatial index exists on each layer in the database. PostGIS supports creation of a GiST (Generalized Search Tree) index to speed up spatial searching (GiST index information is taken from the PostGIS documentation available at https://postgis.net).

Sfat: You can use the DBManager to create an index for your layer. You should first select the layer and click on *Table* ► *Edit table*, go to *Indexes* tab and click on *Add Spatial Index*.

The syntax for creating a GiST index is:

```
CREATE INDEX [indexname] ON [tablename]
  USING GIST ( [geometryfield] GIST_GEOMETRY_OPS );
```

Note that for large tables, creating the index can take a long time. Once the index is created, you should perform a `VACUUM ANALYZE`. See the PostGIS documentation (POSTGIS-PROJECT in *Literatură și Referințe Web*) for more information.

The following example creates a GiST index:

```
gsherman@madison:~/current$ psql gis_data
Welcome to psql 8.3.0, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

gis_data=# CREATE INDEX sidx_alaska_lakes ON alaska_lakes
gis_data-# USING GIST (the_geom GIST_GEOMETRY_OPS);
CREATE INDEX
gis_data=# VACUUM ANALYZE alaska_lakes;
VACUUM
gis_data=# \q
gsherman@madison:~/current$
```

Straturile SpatiaLite

If you want to save a vector layer using the SpatiaLite format, you can do this by following instructions at *Creating new layers from an existing layer*. You select `SpatiaLite` as *Format* and enter both *File name* and *Layer name*.

Also, you can select `SQLite` as format and then add `SPATIALITE=YES` in the *Custom Options* ► *Data source* field. This tells GDAL to create a SpatiaLite database. See also https://gdal.org/drivers/vector/sqlite.html.

QGIS also supports editable views in SpatiaLite. For SpatiaLite data management, you can also use the core plugin *DB Manager*.

If you want to create a new SpatiaLite layer, please refer to section *Creating a new SpatiaLite layer*.

GeoJSON specific parameters

When *exporting layers* to GeoJSON, there are some specific *Layer Options* available. These options come from GDAL which is responsible for the writing of the file:

- *COORDINATE_PRECISION* the maximum number of digits after the decimal separator to write in coordinates. Defaults to 15 (note: for Lat Lon coordinates 6 is considered enough). Truncation will occur to remove trailing zeros.
- *RFC7946* by default GeoJSON 2008 will be used. If set to YES, the updated RFC 7946 standard will be used. Default is NO (thus GeoJSON 2008). See https://gdal.org/drivers/vector/geojson.html#rfc-7946-write-support for the main differences, in short: only EPSG:4326 is allowed, other crs's will be transformed, polygons will be written such as to follow the right-hand rule for orientation, values of a „bbox" array are [west, south, east, north], not [minx, miny, maxx, maxy]. Some extension member names are forbidden in FeatureCollection, Feature and Geometry objects, the default coordinate precision is 7 decimal digits
- *WRITE_BBOX* set to YES to include the bounding box of the geometries at the feature and feature collection level

Besides GeoJSON there is also an option to export to „GeoJSON - Newline Delimited" (see https://gdal.org/drivers/vector/geojsonseq.html). Instead of a FeatureCollection with Features, you can stream one type (probably only Features) sequentially separated with newlines.

GeoJSON - Newline Delimited has some specific Layer options availabe too:

- *COORDINATE_PRECISION* see above (same as for GeoJSON)
- *RS* whether to start records with the RS=0x1E character. The difference is how the features are separated: only by a newline (LF) character (Newline Delimited JSON, geojsonl) or by also prepending a record-separator (RS) character (giving GeoJSON Text Sequences, geojsons). Default to NO. Files are given the `.json` extension if extension is not provided.

SAP HANA Spatial Layers

This section contains some details on how QGIS accesses SAP HANA layers. Most of the time, QGIS should simply provide you with a list of database tables and views that can be loaded, and it will load them on request. However, if you have trouble loading an SAP HANA table or view into QGIS, the information below may help you understand the root cause and assist in resolving the issue.

Feature Identification

If you'd like to use all of QGIS» feature editing capabilities, QGIS must be able to unambiguously identify each feature in a layer. Internally, QGIS uses a 64-bit signed integer to identify features, whereas the negative range is reserved for special purposes.

Therefore, the SAP HANA provider requires a unique key that can be mapped to a positive 64-bit integer to fully support QGIS» feature editing capabilities. If it is not possible to create such a mapping, you might still view the features, but editing might not work.

Adding tables

When adding a table as a layer, the SAP HANA provider uses the table's primary key to map it to a unique feature id. Therefore, to have full feature editing support, you need to have a primary key to your table definition.

The SAP HANA provider supports multi-column primary keys, but if you'd like to get the best performance, your primary key should be a single column of type `INTEGER`.

Adding views

When adding a view as a layer, the SAP HANA provider cannot automatically identify columns that unambiguously identify a feature. Furthermore, some views are read-only and cannot be edited.

To have full feature editing support, the view must be updatable (check column `IS_READ_ONLY` in system view `SYS.VIEWS` for the view in question) and you must manually provide QGIS with one or more columns that identify a feature. The columns can be given by using *Layer* ► *Add Layer* ► *Add SAP HANA Spatial Layer* and then selecting the columns in the *Feature id* column. For best performance, the *Feature id* value should be a single `INTEGER` column.

15.3.3 Layers crossing 180° longitude

Many GIS packages don't wrap layers with a geographic reference system (lat/lon) crossing the 180 degrees longitude line. As result, if we open such a layer in QGIS, we could see two widely separated locations, that should appear near each other. In Fig. 15.31, the tiny point on the far left of the map canvas (Chatham Islands) should be within the grid, to the right of the New Zealand main islands.

Fig. 15.31: Map in lat/lon crossing the 180° longitude line

Solving in PostGIS

A work-around is to transform the longitude values using PostGIS and the ST_ShiftLongitude function. This function reads every point/vertex in every component of every feature in a geometry, and shifts its longitude coordinate from -180..0° to 180..360° and vice versa if between these ranges. This function is symmetrical so the result is a 0..360° representation of a -180..180° data and a -180..180° representation of a 0..360° data.

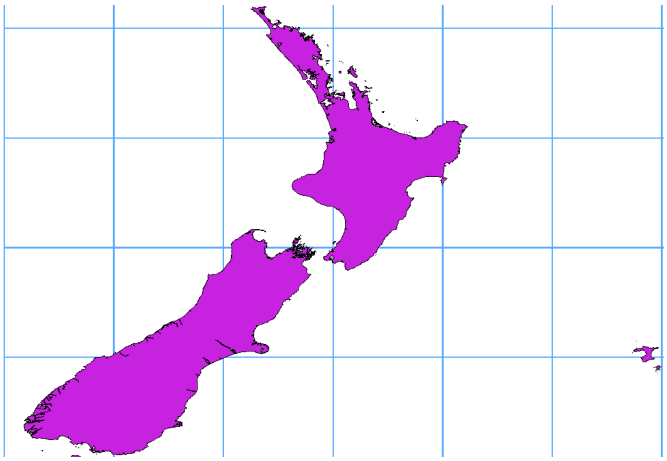


Fig. 15.32: Crossing 180° longitude applying the **ST_ShiftLongitude** function

1. Import data into PostGIS (*Importing Data into PostgreSQL*) using, for example, the DB Manager plugin.
2. Use the PostGIS command line interface to issue the following command:

```
-- In this example, "TABLE" is the actual name of your PostGIS table
update TABLE set geom=ST_ShiftLongitude(geom);
```

3. If everything went well, you should receive a confirmation about the number of features that were updated. Then you'll be able to load the map and see the difference (*Figure_vector_crossing_map*).

CAPITOLUL 16

Lucrul cu Datele Vectoriale

16.1 Dialogul Proprietăților Vectoriale

The *Layer Properties* dialog for a vector layer provides general settings to manage appearance of layer features in the map (symbology, labeling, diagrams), interaction with the mouse (actions, map tips, form design). It also provides information about the layer.

To access the *Layer Properties* dialog:

- In the *Layers* panel, double-click the layer or right-click and select *Properties…* from the pop-up menu;
- Go to *Layer* ► *Layer Properties…* menu when the layer is selected.

The vector *Layer Properties* dialog provides the following sections:

Information	*Source*	*Symbology*[1]
Labels[1]	*Mask*[1]	*3D View*[1]
Diagrams	*Fields*	*Attributes Form*
Joins	*Auxiliary Storage*	*Actions*
Display	*Rendering*	*Temporal*
Variabile	*Elevation*	*Metadata*
Dependencies	*Legend*	*QGIS Server*
Digitizing	*External plugins*[2] tabs	

[1] Also available in the *Layer styling panel*

[2] *External plugins* you install can optionally add tabs to this dialog. Those are not presented in this document. Refer to their documentation.

Sfat: Share full or partial properties of the layer styles

The *Style* menu at the bottom of the dialog allows you to import or export these or part of these properties from/to several destination (file, clipboard, database). See *Managing Custom Styles*.

Notă: Because properties (symbology, label, actions, default values, forms…) of embedded layers (see *Embedding layers from external projects*) are pulled from the original project file and to avoid changes that may break this behavior, the layer properties dialog is made unavailable for these layers.

16.1.1 Information Properties

The *Information* tab is read-only and represents an interesting place to quickly grab summarized information and metadata on the current layer. Provided information are:

- general such as name in the project, source path, list of auxiliary files, last save time and size, the used provider
- based on the provider of the layer: format of storage, geometry type, data source encoding, extent, feature count…
- the Coordinate Reference System: name, units, method, accuracy, reference (i.e. whether it's static or dynamic)
- picked from the *filled metadata*: access, extents, links, contacts, history…
- and related to its geometry (spatial extent, CRS…) or its attributes (number of fields, characteristics of each…).

16.1.2 Proprietățile Sursei

Use this tab to define general settings for the vector layer.

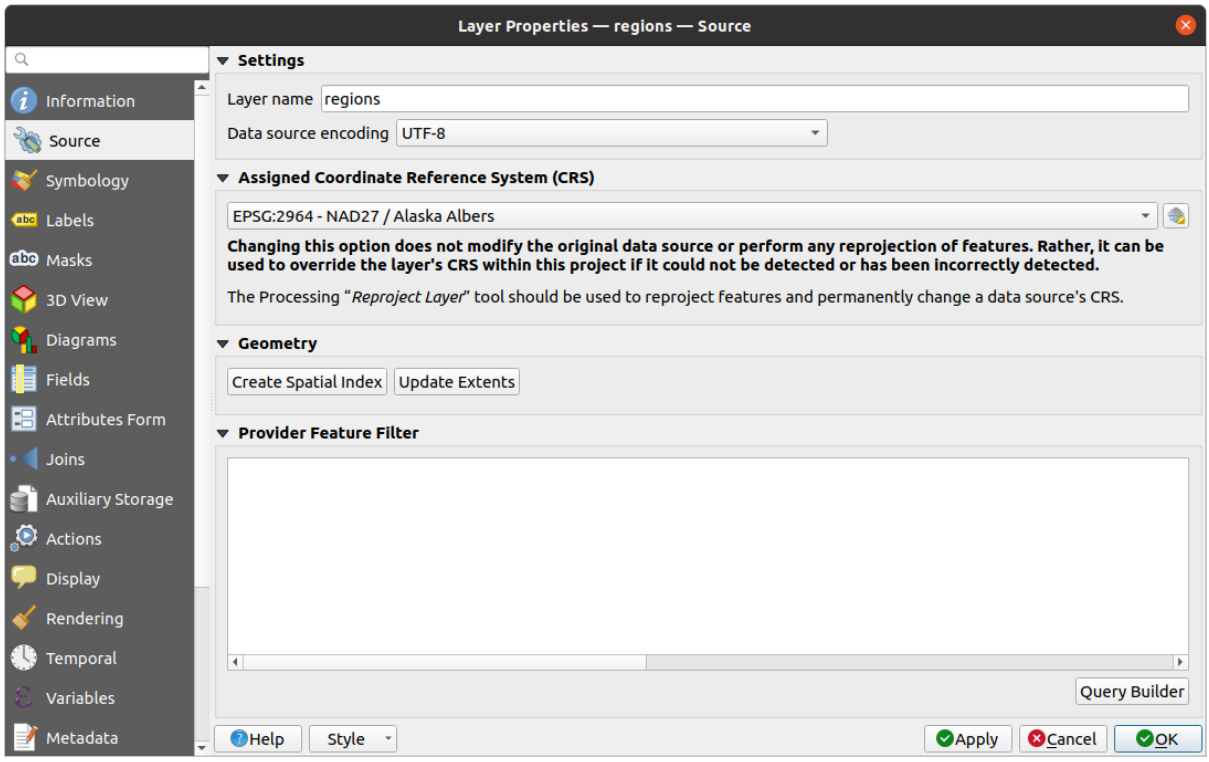


Fig. 16.1: Source tab in vector Layer Properties dialog

Setări

- Set a *Layer name* different from the layer filename that will be used to identify the layer in the project (in the *Layers Panel*, with expressions, in print layout legend, …)
- Depending on the data format, select the *Data source encoding* if not correctly detected by QGIS.

Coordinate Reference System and Geometry

- Displays the layer's *Assigned Coordinate Reference System (CRS)*. You can change the layer's CRS, selecting a recently used one in the drop-down list or clicking on Select CRS button (see *Coordinate Reference System Selector*). Use this process only if the CRS applied to the layer is a wrong one or if none was applied. If you wish to reproject your data into another CRS, rather use layer reprojection algorithms from Processing or *Save it into another layer*.
- *Create spatial index* (only for OGR-supported formats).
- *Update extents* information for a layer.

Constructorul de Interogări

The Query Builder dialog is accessible through the *Query Builder* button at the bottom of the *Source* tab in the Layer Properties dialog, under the *Provider feature filter* group.

The Query Builder provides an interface that allows you to define a subset of the features in the layer using a SQL-like WHERE clause and to display the result in the main window. As long as the query is active, only the features corresponding to its result are available in the project.

You can use one or more layer attributes to define the filter in the `Query Builder`. The use of more than one attribute is shown in Fig. 16.2. In the example, the filter combines the attributes

- `toa` (`DateTime` field: `cast("toa" as character) > '2017-05-17'` and `cast("toa" as character) < '2019-12-24T18:00:00'`),
- `name` (`String` field: `"name" > 'S'`) and
- `FID` (`Integer` field: `FID > 10`)

using the AND, OR and NOT operators and parenthesis. This syntax (including the DateTime format for the `toa` field) works for GeoPackage datasets.

The filter is made at the data provider (OGR, PostgreSQL, MS SQL Server…) level. So the syntax depends on the data provider (DateTime is for instance not supported for the ESRI Shapefile format). The complete expression:

```
cast("toa" as character) > '2017-05-17' AND
cast("toa" as character) < '2019-12-24T18:00:00' AND
NOT ("name" > 'S' OR FID > 10)
```

You can also open the *Query Builder* dialog using the *Filter…* option from the *Layer* menu or the layer contextual menu. The *Fields*, *Values* and *Operators* sections in the dialog help you to construct the SQL-like query exposed in the *Provider specific filter expression* box.

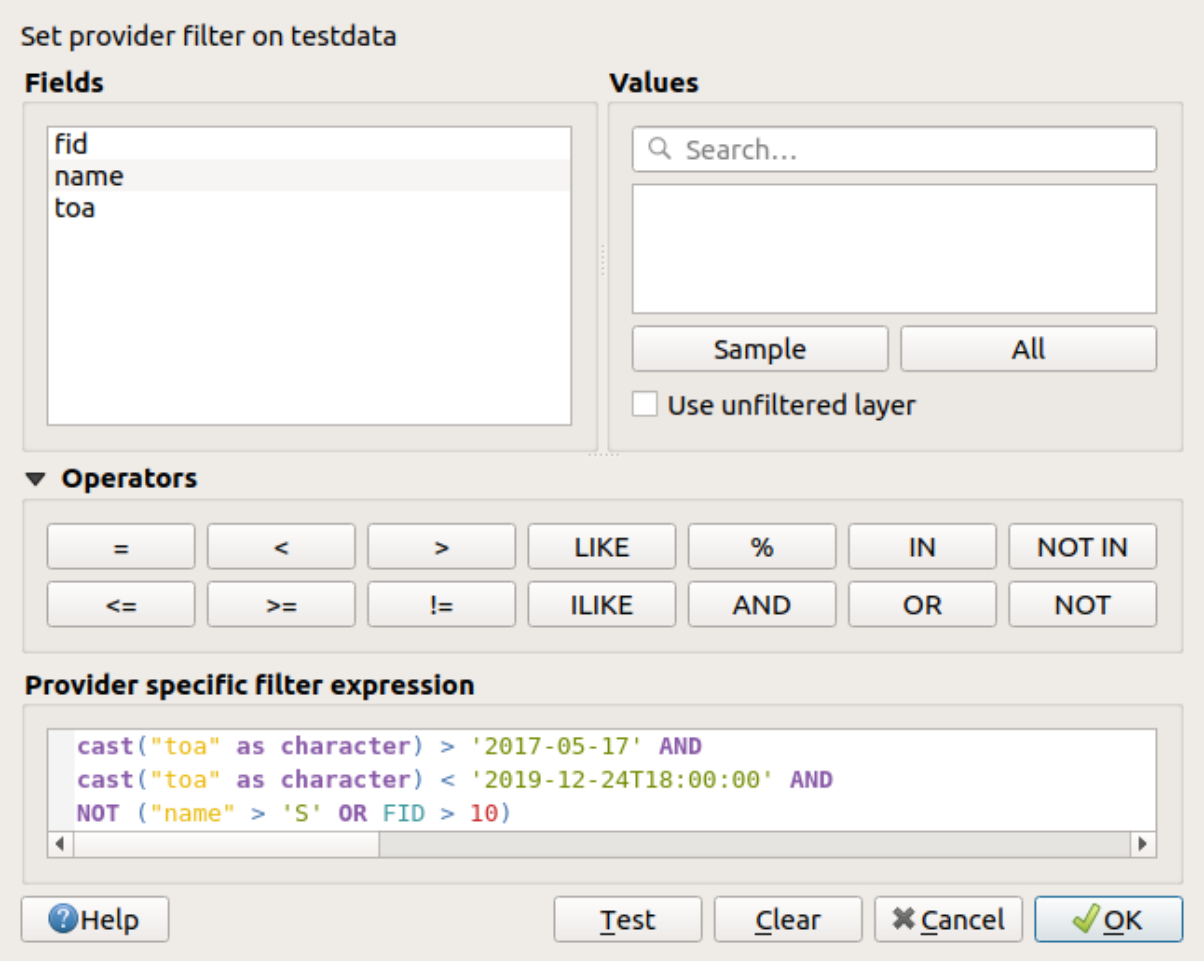


Fig. 16.2: Constructorul de Interogări

The **Fields** list contains all the fields of the layer. To add an attribute column to the SQL WHERE clause field, double-click its name or just type it into the SQL box.

The **Values** frame lists the values of the currently selected field. To list all unique values of a field, click the *All* button. To instead list the first 25 unique values of the column, click the *Sample* button. To add a value to the SQL WHERE clause field, double click its name in the Values list. You can use the search box at the top of the Values frame to easily browse and find attribute values in the list.

The **Operators** section contains all usable operators. To add an operator to the SQL WHERE clause field, click the appropriate button. Relational operators (= , > , ...), string comparison operator (`LIKE`), and logical operators (`AND`, `OR`, ...) are available.

The *Test* button helps you check your query and displays a message box with the number of features satisfying the current query. Use the *Clear* button to wipe the SQL query and revert the layer to its original state (ie, fully load all the features). It is possible to *Save...* the query as a `.QQF` file, or *Load...* the query from a file into the dialog.

When a filter is applied, QGIS treats the resulting subset acts as if it were the entire layer. For example if you applied the filter above for «Borough» (`"TYPE_2" = 'Borough'`), you can not display, query, save or edit `Anchorage`, because that is a «Municipality» and therefore not part of the subset.

Sfat: Filtered layers are indicated in the Layers Panel

In the *Layers* panel, filtered layer is listed with a Filter icon next to it indicating the query used when the mouse hovers over the button. Double-click the icon opens the *Query Builder* dialog for edit. This can also be achieved through the *Layer* ► *Filter...* menu.

16.1.3 Symbology Properties

The *Symbology* tab provides you with a comprehensive tool for rendering and symbolizing your vector data. You can use tools that are common to all vector data, as well as special symbolizing tools that were designed for the different kinds of vector data. However all types share the following dialog structure: in the upper part, you have a widget that helps you prepare the classification and the symbol to use for features and at the bottom the *Randarea stratului* widget.

Sfat: Comutarea rapidă între diferite reprezentări ale straturilor

Using the *Styles* ► *Add* menu at the bottom of the *Layer Properties* dialog, you can save as many styles as needed. A style is the combination of all properties of a layer (such as symbology, labeling, diagram, fields form, actions…) as you want. Then, simply switch between styles from the context menu of the layer in *Layers Panel* to automatically get different representations of your data.

Sfat: Exportare simbologie vectorială

You have the option to export vector symbology from QGIS into Google *.kml, *.dxf and MapInfo *.tab files. Just open the right mouse menu of the layer and click on *Save As…* to specify the name of the output file and its format. In the dialog, use the *Symbology export* menu to save the symbology either as *Feature symbology* ► or as *Symbol layer symbology* ►. If you have used symbol layers, it is recommended to use the second setting.

Randarea entităților

The renderer is responsible for drawing a feature together with the correct symbol. Regardless layer geometry type, there are four common types of renderers: single symbol, categorized, graduated and rule-based. For point layers, there are point displacement, point cluster and heatmap renderers available while polygon layers can also be rendered with the merged features, inverted polygons and 2.5 D renderers.

There is no continuous color renderer, because it is in fact only a special case of the graduated renderer. The categorized and graduated renderers can be created by specifying a symbol and a color ramp - they will set the colors for symbols appropriately. For each data type (points, lines and polygons), vector symbol layer types are available. Depending on the chosen renderer, the dialog provides different additional sections.

Notă: If you change the renderer type when setting the style of a vector layer the settings you made for the symbol will be maintained. Be aware that this procedure only works for one change. If you repeat changing the renderer type the settings for the symbol will get lost.

Render cu Simbol Unic

The *Single Symbol* renderer is used to render all features of the layer using a single user-defined symbol. See *The Symbol Selector* for further information about symbol representation.

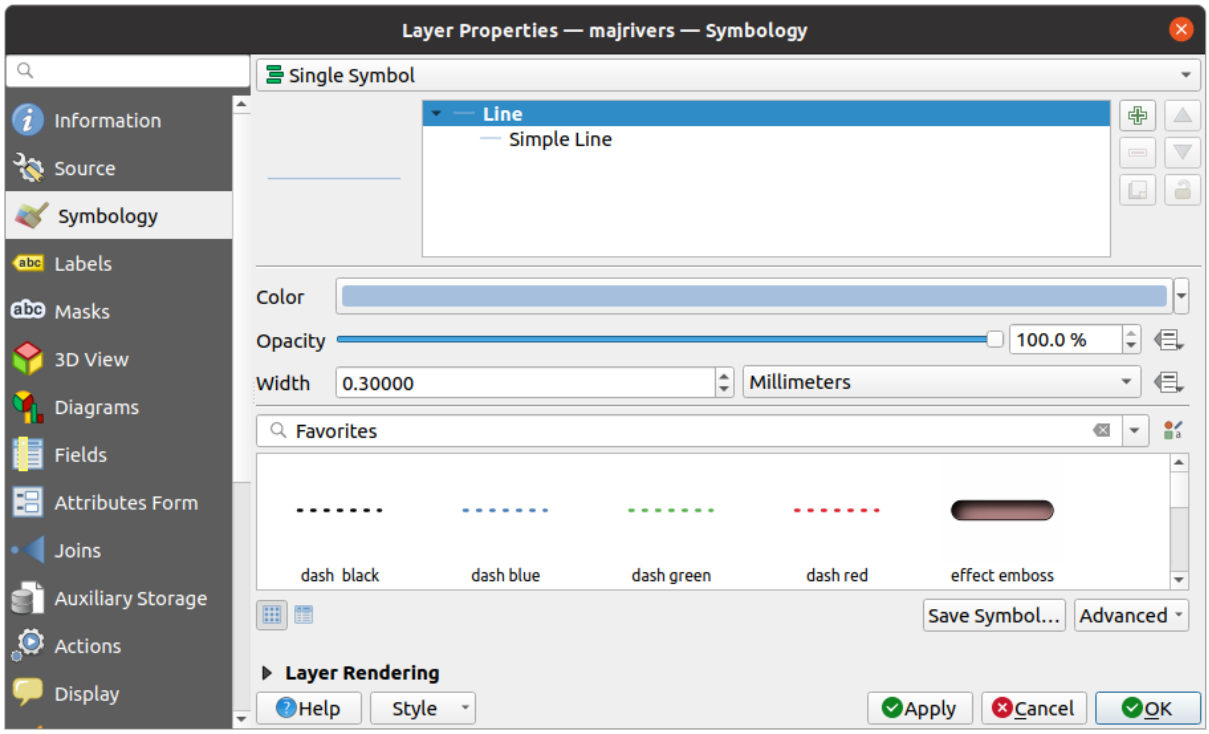


Fig. 16.3: Proprietățile liniei cu simbol unic

No Symbols Renderer

The *No Symbols* renderer is a special use case of the Single Symbol renderer as it applies the same rendering to all features. Using this renderer, no symbol will be drawn for features, but labeling, diagrams and other non-symbol parts will still be shown.

Selections can still be made on the layer in the canvas and selected features will be rendered with a default symbol. Features being edited will also be shown.

This is intended as a handy shortcut for layers which you only want to show labels or diagrams for, and avoids the need to render symbols with totally transparent fill/border to achieve this.

Renderul Categorisit

The *Categorized* renderer is used to render the features of a layer, using a user-defined symbol whose aspect reflects the discrete values of a field or an expression.

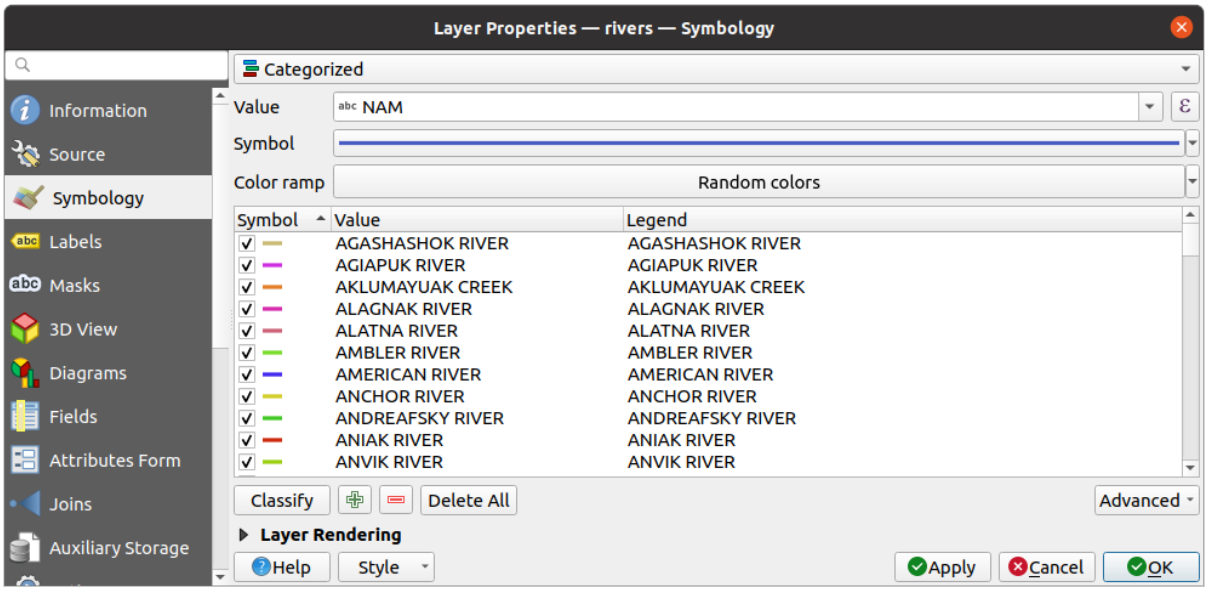


Fig. 16.4: Categorized Symbolizing options

To use categorized symbology for a layer:

1. Select the *Value* of classification: it can be an existing field or an *expression* you can type in the box or build using the associated ε button. Using expressions for categorizing avoids the need to create a field for symbology purposes only (eg, if your classification criteria are derived from one or more attributes).

   The expression used to classify features can be of any type; eg, it can:

   - be a comparison. In this case, QGIS returns values `1` (**True**) and `0` (**False**). Some examples:

     ```
     myfield >= 100
     $id = @atlas_featureid
     myfield % 2 = 0
     within( $geometry, @atlas_geometry )
     ```

   - combine different fields:

     ```
     concat( field_1, ' ', field_2 )
     ```

   - be a calculation on fields:

     ```
     myfield % 2
     year( myfield )
     field_1 + field_2
     substr( field_1, -3 )
     ```

   - be used to transform linear values to discrete classes, e.g.:

     ```
     CASE WHEN x > 1000 THEN 'Big' ELSE 'Small' END
     ```

   - combine several discrete values into a single category, e.g.:

```
CASE
WHEN building IN ('residence', 'mobile home') THEN 'residential'
WHEN building IN ('commercial', 'industrial') THEN 'Commercial and␣
↪Industrial'
END
```

Sfat: While you can use any kind of expression to categorize features, for some complex expressions it might be simpler to use *rule-based rendering*.

2. Configure the *Symbol*, which will be used as base symbol for all the classes;
3. Indicate the *Color ramp*, ie the range of colors from which the color applied to each symbol is selected.

   Besides the common options of the *color ramp widget*, you can apply a *Random Color Ramp* to the categories. You can click the *Shuffle Random Colors* entry to regenerate a new set of random colors if you are not satisfied.
4. Then click on the *Classify* button to create classes from the distinct values of the provided field or expression.
5. *Apply* the changes if the *live update* is not in use and each feature on the map canvas will be rendered with the symbol of its class.

   By default, QGIS appends an *all other values* class to the list. While empty at the beginning, this class is used as a default class for any feature not falling into the other classes (eg, when you create features with new values for the classification field / expression).

Further tweaks can be done to the default classification:

- You can Add new categories, Remove selected categories or *Delete All* of them.
- A class can be disabled by unchecking the checkbox to the left of the class name; the corresponding features are hidden on the map.
- Drag-and-drop the rows to reorder the classes
- To change the symbol, the value or the legend of a class, double click the item.

Right-clicking over selected item(s) shows a contextual menu to:

- *Copy Symbol* and *Paste Symbol*, a convenient way to apply the item's representation to others
- *Change Color…* of the selected symbol(s)
- *Change Opacity…* of the selected symbol(s)
- *Change Output Unit…* of the selected symbol(s)
- *Change Width…* of the selected line symbol(s)
- *Change Size…* of the selected point symbol(s)
- *Change Angle…* of the selected point symbol(s)
- *Merge Categories*: Groups multiple selected categories into a single one. This allows simpler styling of layers with a large number of categories, where it may be possible to group numerous distinct categories into a smaller and more manageable set of categories which apply to multiple values.

  Sfat: Since the symbol kept for the merged categories is the one of the topmost selected category in the list, you may want to move the category whose symbol you wish to reuse to the top before merging.

- *Unmerge Categories* that were previously merged

The created classes also appear in a tree hierarchy in the *Layers* panel. Double-click an entry in the map legend to edit the assigned symbol. Right-click and you will get some *more options*.

The *Advanced* menu gives access to options to speed classification or fine-tune the symbols rendering:

- *Match to saved symbols*: Using the *symbols library*, assigns to each category a symbol whose name represents the classification value of the category
- *Match to symbols from file…*: Provided a file with symbols, assigns to each category a symbol whose name represents the classification value of the category
- *Symbol levels…* to define the order of symbols rendering.

Renderul Gradat

The *Graduated* renderer is used to render all the features from a layer, using an user-defined symbol whose color or size reflects the assignment of a selected feature's attribute to a class.

Like the Categorized Renderer, the Graduated Renderer allows you to define rotation and size scale from specified columns.

Also, analogous to the Categorized Renderer, it allows you to select:

- The *Value* of classification: it can be an existing field or an *expression* you can type in the box or build using the associated button. Using expressions for graduating avoids the need to create a field for symbology purposes only (eg, if your classification criteria are derived from one or more attributes).
- The symbol (using the Symbol selector dialog)
- Formatul legendei și precizia
- Metoda utilizată pentru a schimba culoarea sau dimensiunea simbolului
- The colors (using the color Ramp list) if the color method is selected
- The size (using the size domain and its unit)

Then you can use the Histogram tab which shows an interactive histogram of the values from the assigned field or expression. Class breaks can be moved or added using the histogram widget.

Notă: You can use Statistical Summary panel to get more information on your vector layer. See *Statistical Summary Panel*.

Back to the Classes tab, you can specify the number of classes and also the mode for classifying features within the classes (using the Mode list). The available modes are:

- Equal Count (Quantile): each class will have the same number of elements (the idea of a boxplot).
- Equal Interval: each class will have the same size (e.g. with the values from 1 to 16 and four classes, each class will have a size of four).
- Logarithmic scale: suitable for data with a wide range of values. Narrow classes for low values and wide classes for large values (e.g. for decimal numbers with range [0..100] and two classes, the first class will be from 0 to 10 and the second class from 10 to 100).
- Natural Breaks (Jenks): the variance within each class is minimized while the variance between classes is maximized.
- Pretty Breaks: computes a sequence of about n+1 equally spaced nice values which cover the range of the values in x. The values are chosen so that they are 1, 2 or 5 times a power of 10. (based on pretty from the R statistical environment https://www.rdocumentation.org/packages/base/topics/pretty).
- Standard Deviation: classes are built depending on the standard deviation of the values.

The listbox in the center part of the *Symbology* tab lists the classes together with their ranges, labels and symbols that will be rendered.

Click on **Classify** button to create classes using the chosen mode. Each classes can be disabled unchecking the checkbox at the left of the class name.

To change symbol, value and/or label of the class, just double click on the item you want to change.

Right-clicking over selected item(s) shows a contextual menu to:

- *Copy Symbol* and *Paste Symbol*, a convenient way to apply the item's representation to others
- *Change Color...* of the selected symbol(s)
- *Change Opacity...* of the selected symbol(s)
- *Change Output Unit...* of the selected symbol(s)
- *Change Width...* of the selected line symbol(s)
- *Change Size...* of the selected point symbol(s)
- *Change Angle...* of the selected point symbol(s)

The example in Fig. 16.5 shows the graduated rendering dialog for the major_rivers layer of the QGIS sample dataset.

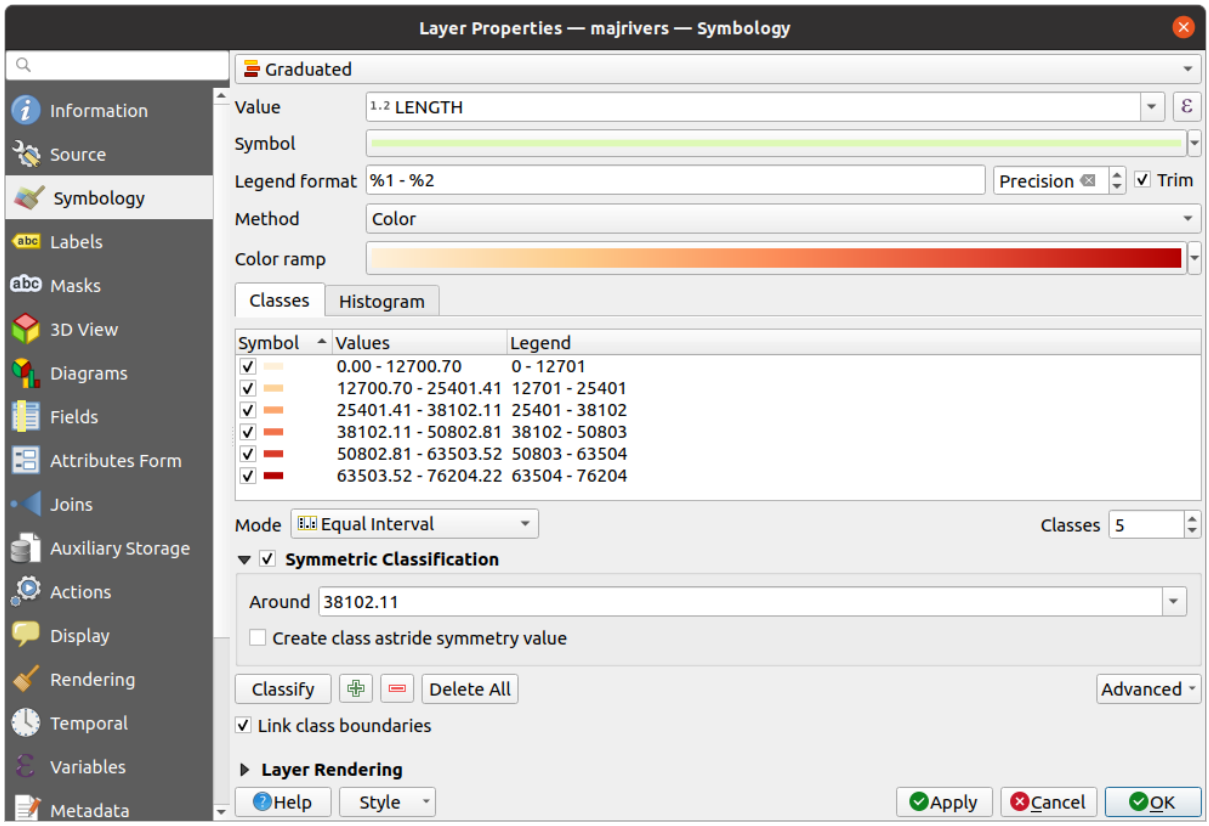


Fig. 16.5: Opțiunile de Simbolizare Gradată

The created classes also appear in a tree hierarchy in the *Layers* panel. Double-click an entry in the map legend to edit the assigned symbol. Right-click and you will get some *more options*.

Simbolul Proporțional și Analiză Multivariată

Proportional Symbol and Multivariate Analysis are not rendering types available from the Symbology rendering drop-down list. However with the *data-defined override* options applied over any of the previous rendering options, QGIS allows you to display your point and line data with such representation.

Creating proportional symbol

To apply a proportional rendering:

1. First apply to the layer the *single symbol renderer*.
2. Then set the symbol to apply to the features.
3. Select the item at the upper level of the symbol tree, and use the Data-defined override *button* next to the *Size* (for point layer) or *Width* (for line layer) option.
4. Select a field or enter an expression, and for each feature, QGIS will apply the output value to the property and proportionally resize the symbol in the map canvas.

   If need be, use the *Size assistant...* option of the menu to apply some transformation (exponential, flannery…) to the symbol size rescaling (see *Using the data-defined assistant interface* for more details).

You can choose to display the proportional symbols in the *Layers panel* and the *print layout legend item*: unfold the *Advanced* drop-down list at the bottom of the main dialog of the *Symbology* tab and select **Data-defined size legend…** to configure the legend items (see *Data-defined size legend* for details).

Crearea analiză multivariate

O randare a unei analize multivariată vă ajută să evaluați relația dintre două sau mai multe variabile, astfel, una poate fi reprezentată printr-o gamă de culoare, pe când cealaltă poate fi reprezentată printr-o mărime.

The simplest way to create multivariate analysis in QGIS is to:

1. First apply a categorized or graduated rendering on a layer, using the same type of symbol for all the classes.
2. Then, apply a proportional symbology on the classes:
   1. Click on the *Change* button above the classification frame: you get the *The Symbol Selector* dialog.
   2. Rescale the size or width of the symbol layer using the *data defined override* widget as seen above.

Like the proportional symbol, the scaled symbology can be added to the layer tree, on top of the categorized or graduated classes symbols using the *data defined size legend* feature. And both representation are also available in the print layout legend item.

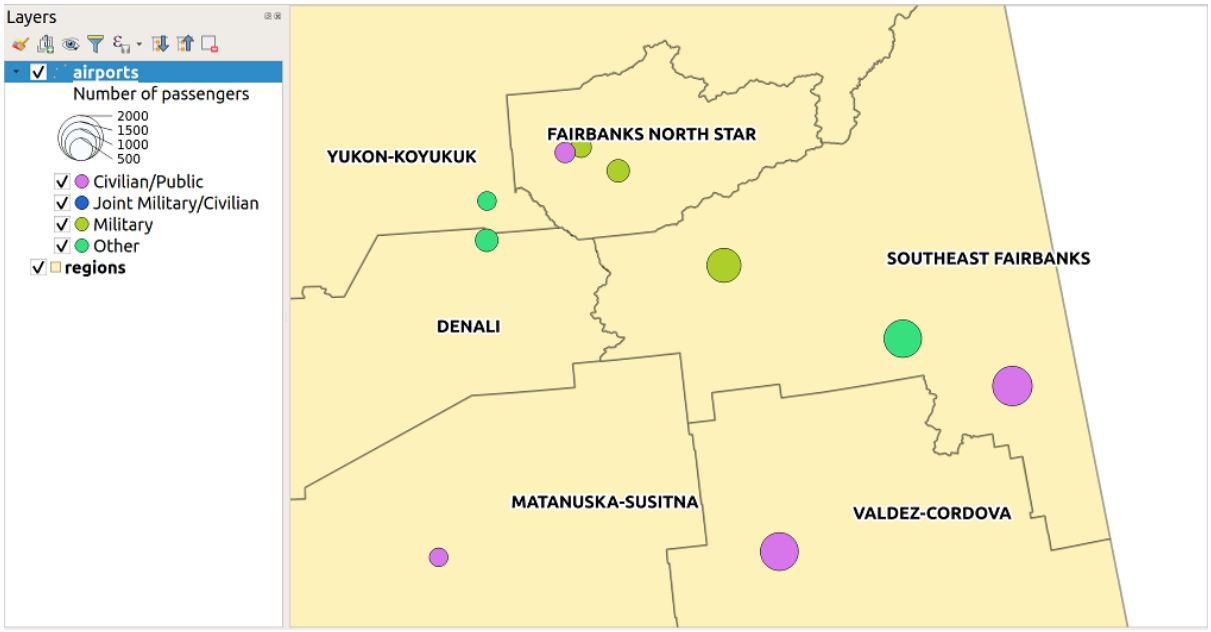


Fig. 16.6: Multivariate example with scaled size legend

Rule-based Renderer

Rules are QGIS *expressions* used to discriminate features according to their attributes or properties in order to apply specific rendering settings to them. Rules can be nested, and features belong to a class if they belong to all the upper nesting level(s).

The *Rule-based* renderer is thus designed to render all the features from a layer, using symbols whose aspect reflects the assignment of a selected feature to a fine-grained class.

Pentru a crea o regulă:

1. Activate an existing row by double-clicking it (by default, QGIS adds a symbol without a rule when the rendering mode is enabled) or click the Edit rule or Add rule button.
2. In the *Edit Rule* dialog that opens, you can define a label to help you identify each rule. This is the label that will be displayed in the *Layers Panel* and also in the print composer legend.
3. Manually enter an expression in the text box next to the *Filter* option or press the button next to it to open the expression string builder dialog.
4. Use the provided functions and the layer attributes to build an *expression* to filter the features you'd like to retrieve. Press the *Test* button to check the result of the query.
5. You can enter a longer label to complete the rule description.
6. You can use the *Scale Range* option to set scales at which the rule should be applied.
7. Finally, configure the *symbol to use* for these features.
8. Apoi apăsați pe *OK*

A new row summarizing the rule is added to the Layer Properties dialog. You can create as many rules as necessary following the steps above or copy pasting an existing rule. Drag-and-drop the rules on top of each other to nest them and refine the upper rule features in subclasses.

The rule-based renderer can be combined with categorized or graduated renderers. Selecting a rule, you can organize its features in subclasses using the *Refine selected rules* drop-down menu. Refined classes appear like sub-items of the

rule, in a tree hierarchy and like their parent, you can set the symbology and the rule of each class. Automated rule refinement can be based on:

- **scales**: given a list of scales, this option creates a set of classes to which the different user-defined scale ranges apply. Each new scale-based class can have its own symbology and expression of definition. This can e.g. be a convenient way to display the same features with various symbols at different scales, or display only a set of features depending on the scale (e.g. local airports at large scale vs international airports at small scale).
- **categories**: applies a *categorized renderer* to the features falling in the selected rule.
- or **ranges**: applies a *graduated renderer* to the features falling in the selected rule.

Refined classes appear like sub-items of the rule, in a tree hierarchy and like above, you can set symbology of each class. Symbols of the nested rules are stacked on top of each other so be careful in choosing them. It is also possible to uncheck *Symbols* in the *Edit rule* dialog to avoid rendering a particular symbol in the stack.

In the *Edit rule* dialog, you can avoid writing all the rules and make use of the *Else* option to catch all the features that do not match any of the other rules, at the same level. This can also be achieved by writing `Else` in the *Rule* column of the *Layer Properties* ► *Symbology* ► *Rule-based* dialog.

Right-clicking over selected item(s) shows a contextual menu to:

- *Copy* and *Paste*, a convenient way to create new item(s) based on existing item(s)
- *Copy Symbol* and *Paste Symbol*, a convenient way to apply the item's representation to others
- *Change Color…* of the selected symbol(s)
- *Change Opacity…* of the selected symbol(s)
- *Change Output Unit…* of the selected symbol(s)
- *Change Width…* of the selected line symbol(s)
- *Change Size…* of the selected point symbol(s)
- *Change Angle…* of the selected point symbol(s)
- *Refine Current Rule*: open a submenu that allows to refine the current rule with **scales**, **categories** or **Ranges**. Same as selecting the *corresponding menu* at the bottom of the dialog.

Unchecking a row in the rule-based renderer dialog hides in the map canvas the features of the specific rule and the nested ones.

The created rules also appear in a tree hierarchy in the map legend. Double-click an entry in the map legend to edit the assigned symbol. Right-click and you will get some *more options*.

The example in Fig. 16.7 shows the rule-based rendering dialog for the rivers layer of the QGIS sample dataset.

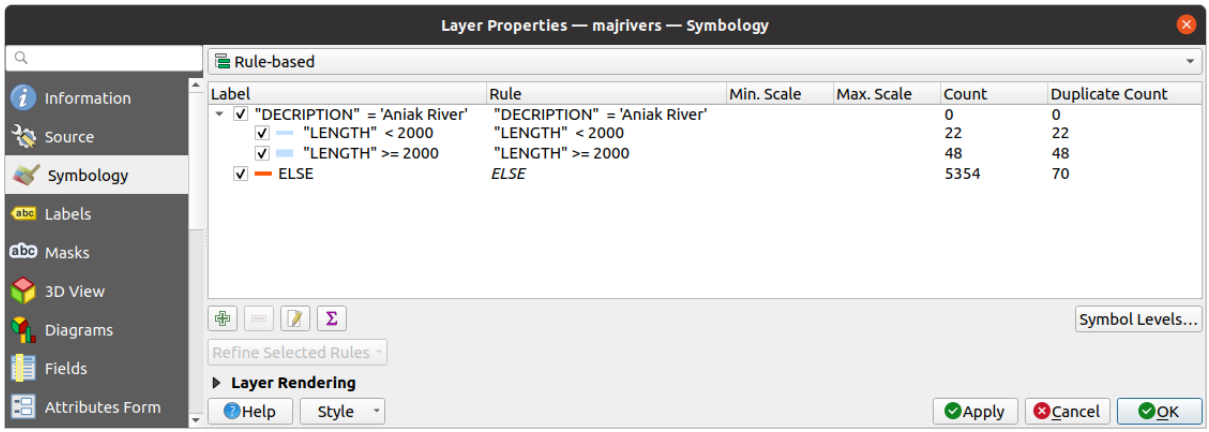


Fig. 16.7: Opțiunile de Simbolizare bazată pe reguli

Point displacement Renderer

The *Point Displacement* renderer takes the point features falling in a given distance tolerance from each other and places their symbols around their barycenter, following different placement methods. This can be a convenient way to visualize all the features of a point layer, even if they have the same location (e.g. amenities in a building).

To configure a point displacement renderer, you have to:

1. Set the *Center symbol*: how the virtual point at the center will look like
2. Select the *Renderer* type: how you want to classify features in the layer (single, categorized, rule-based…)
3. Press the *Renderer Settings…* button to configure features» symbology according to the selected renderer
4. Indicate the *Distance* tolerance in which close features are considered overlapping and then displaced over the same virtual point. Supports common symbol units.
5. Configure the *Placement methods*:
   - **Ring**: places all the features on a circle whose radius depends on the number of features to display.
   - **Concentric rings**: uses a set of concentric circles to show the features.
   - **Grid**: generates a regular grid with a point symbol at each intersection.
6. Displaced symbols are placed on the *Displacement lines*. While the minimal spacing of the displacement lines depends on the point symbols renderer, you can still customize some of their settings such as the *Stroke width*, *Stroke color* and *Size adjustment* (e.g., to add more spacing between the rendered points).
7. Use the *Labels* group options to perform points labeling: the labels are placed near the displaced symbol, and not at the feature real position.
   1. Select the *Label attribute*: a field of the layer to use for labeling
   2. Indicate the *Label font* properties and size
   3. Pick a *Label color*
   4. Set a *Label distance factor*: for each point feature, offsets the label from the symbol center proportionally to the symbol's diagonal size.
   5. Turn on *Use scale dependent labeling* if you want to display labels only on scales larger than a given *Minimum map scale*.

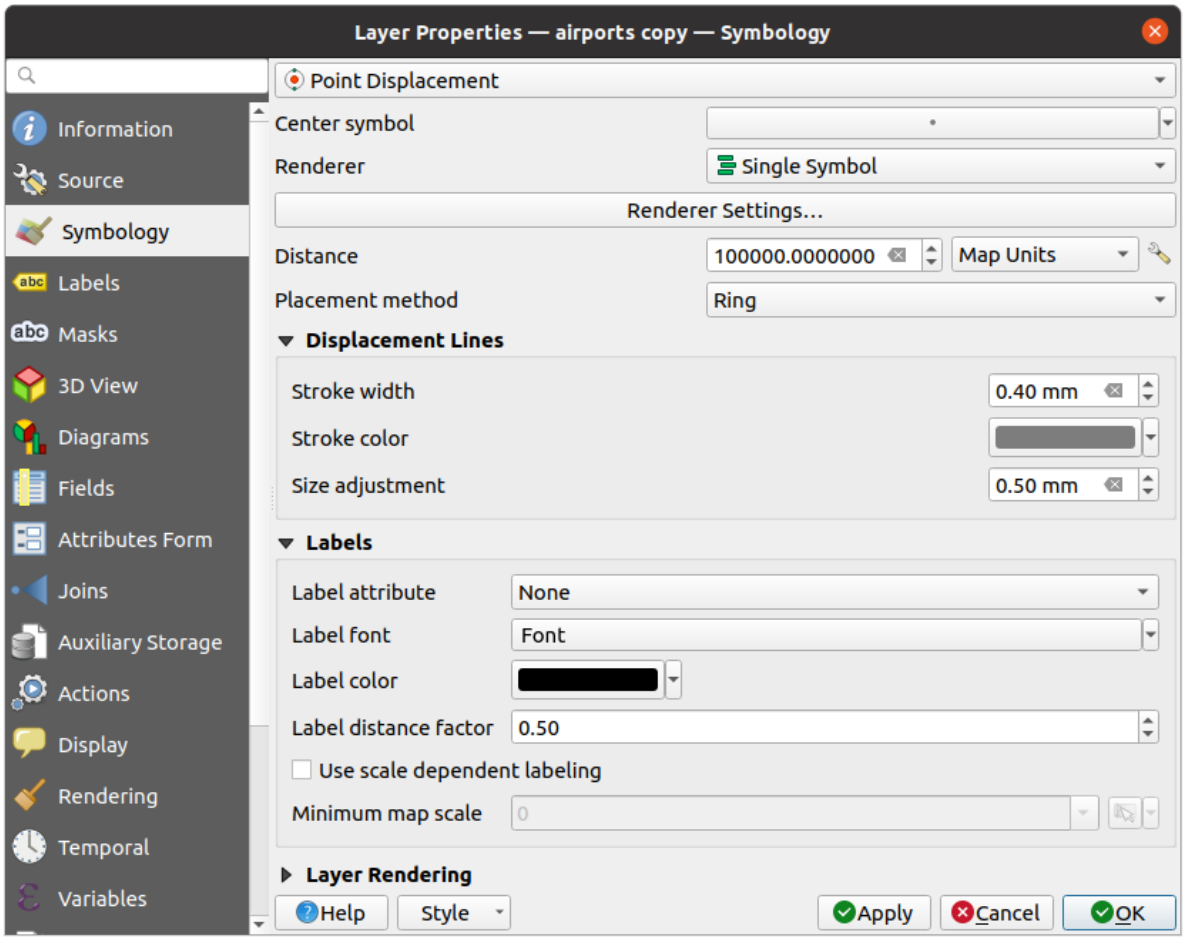


Fig. 16.8: Dialogul de deplasare a punctelor

Notă: Point Displacement renderer does not alter feature geometry, meaning that points are not moved from their position. They are still located at their initial place. Changes are only visual, for rendering purpose. Use instead the Processing *Points displacement* algorithm if you want to create displaced features.

Point Cluster Renderer

Unlike the *Point Displacement* renderer which blows up nearest or overlaid point features placement, the *Point Cluster* renderer groups nearby points into a single rendered marker symbol. Points that fall within a specified distance from each others are merged into a single symbol. Points aggregation is made based on the closest group being formed, rather than just assigning them the first group within the search distance.

În fereastra de dialog principală, puteți:

1. Set the symbol to represent the point cluster in the *Cluster symbol*; the default rendering displays the number of aggregated features thanks to the `@cluster_size` *variable* on Font marker symbol layer.
2. Select the *Renderer* type, i.e. how you want to classify features in the layer (single, categorized, rule-based…)
3. Press the *Renderer Settings…* button to configure features» symbology as usual. Note that this symbology is only visible on features that are not clustered, the *Cluster symbol* being applied otherwise. Also, when all the point features in a cluster belong to the same rendering class, and thus would be applied the same color, that color represents the `@cluster_color` variable of the cluster.
4. Indicate the maximal *Distance* to consider for clustering features. Supports common symbol units.

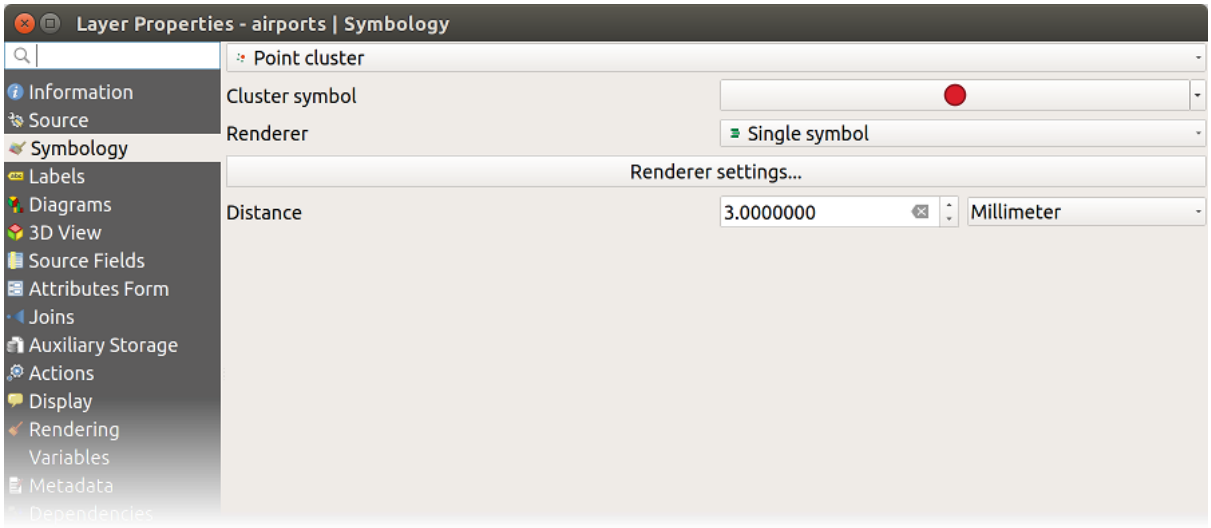


Fig. 16.9: Point Cluster dialog

Notă: Point Cluster renderer does not alter feature geometry, meaning that points are not moved from their position. They are still located at their initial place. Changes are only visual, for rendering purpose. Use instead the Processing *K-means clustering* or *DBSCAN clustering* algorithm if you want to create cluster-based features.

Merged Features Renderer

The *Merged Features* renderer allows area and line features to be "dissolved" into a single object prior to rendering to ensure that complex symbols or overlapping features are represented by a uniform and contiguous cartographic symbol.

Inverted Polygon Renderer

The *Inverted Polygon* renderer allows user to define a symbol to fill in outside of the layer's polygons. As above you can select subrenderers, namely Single symbol, Graduated, Categorized, Rule-Based or 2.5D renderer.

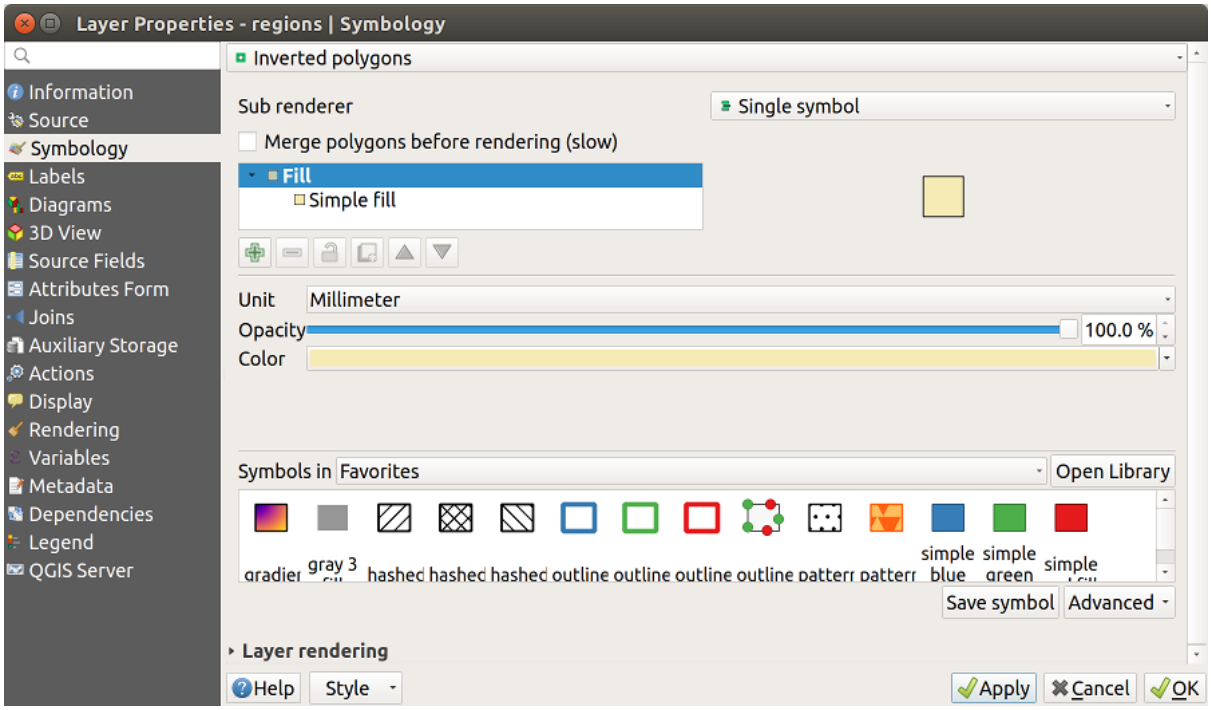


Fig. 16.10: Dialogul Poligonului Invers

Heatmap Renderer

With the *Heatmap* renderer you can create live dynamic heatmaps for (multi)point layers. You can specify the heatmap radius in millimeters, points, pixels, map units or inches, choose and edit a color ramp for the heatmap style and use a slider for selecting a trade-off between render speed and quality. You can also define a maximum value limit and give a weight to points using a field or an expression. When adding or removing a feature the heatmap renderer updates the heatmap style automatically.

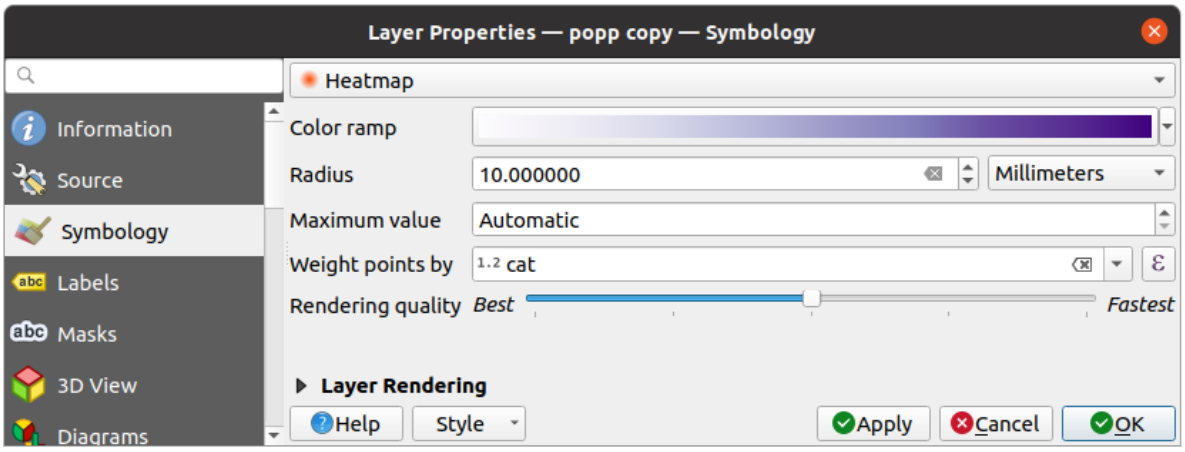


Fig. 16.11: Fereasta plugin-ului Heatmap

2.5D Renderer

Using the *2.5D* renderer it's possible to create a 2.5D effect on your layer's features. You start by choosing a *Height* value (in map units). For that you can use a fixed value, one of your layer's fields, or an expression. You also need to choose an *Angle* (in degrees) to recreate the viewer position (0° means west, growing in counter clock wise). Use advanced configuration options to set the *Roof Color* and *Wall Color*. If you would like to simulate solar radiation on the features walls, make sure to check the *Shade walls based on aspect* option. You can also simulate a shadow by setting a *Color* and *Size* (in map units).

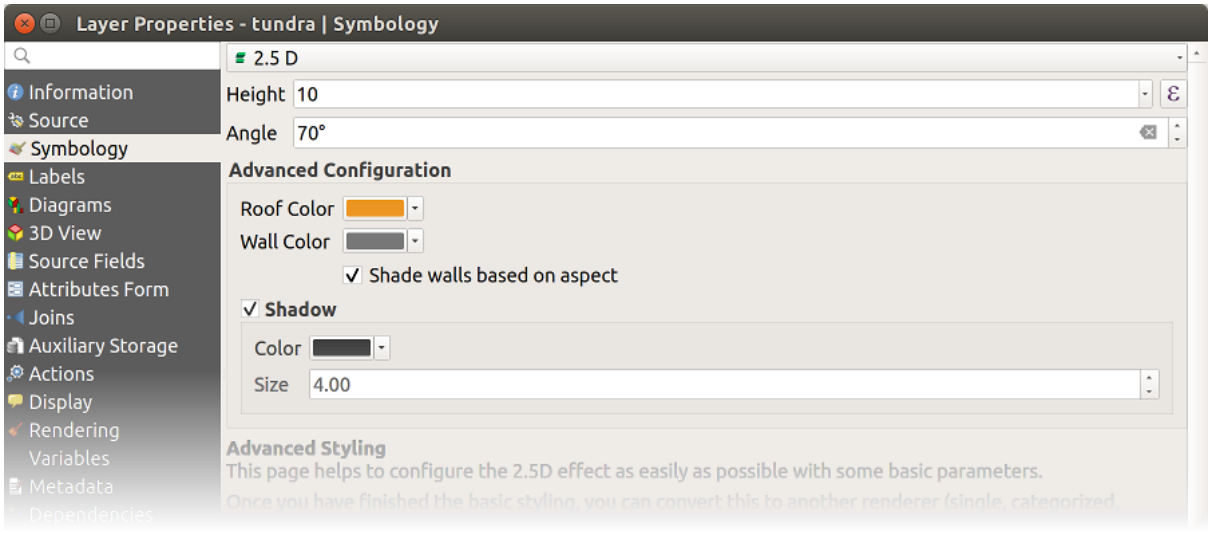


Fig. 16.12: 2.5D dialog

Sfat: Using 2.5D effect with other renderers

Once you have finished setting the basic style on the 2.5D renderer, you can convert this to another renderer (single, categorized, graduated). The 2.5D effects will be kept and all other renderer specific options will be available for you to fine tune them (this way you can have for example categorized symbols with a nice 2.5D representation or add some extra styling to your 2.5D symbols). To make sure that the shadow and the „building" itself do not interfere with other nearby features, you may need to enable Symbols Levels (*Advanced* ► *Symbol levels…*). The 2.5D height and angle values are saved in the layer's variables, so you can edit it afterwards in the variables tab of the layer's properties dialog.

Embedded Renderer

The *Embedded Symbols* renderer allows to display the «native» symbology of a provided datasource. This is mostly the case with `KML` and `TAB` datasets that have predefined symbology.

Randarea stratului

From the Symbology tab, you can also set some options that invariably act on all features of the layer:

- *Opacity* : You can make the underlying layer in the map canvas visible with this tool. Use the slider to adapt the visibility of your vector layer to your needs. You can also make a precise definition of the percentage of visibility in the menu beside the slider.
- *Blending mode* at the *Layer* and *Feature* levels: You can achieve special rendering effects with these tools that you may previously only know from graphics programs. The pixels of your overlaying and underlaying layers are mixed through the settings described in *Modurile de Fuziune*.
- Apply *paint effects* on all the layer features with the *Draw Effects* button.
- *Control feature rendering order* allows you, using features attributes, to define the z-order in which they shall be rendered. Activate the checkbox and click on the button beside. You then get the *Define Order* dialog in which you:

  1. Choose a field or build an expression to apply to the layer features.
  2. Set in which order the fetched features should be sorted, i.e. if you choose **Ascending** order, the features with lower value are rendered under those with higher value.
  3. Define when features returning NULL value should be rendered: **first** (bottom) or **last** (top).
  4. Repeat the above steps as many times as rules you wish to use.

  The first rule is applied to all the features in the layer, z-ordering them according to their returned value. Then, within each group of features with the same value (including those with NULL value) and thus the same z-level, the next rule is applied to sort them. And so on…

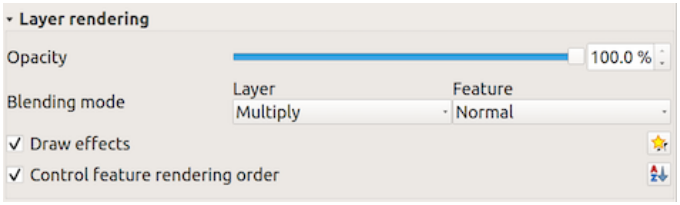


Fig. 16.13: Opţiuni de randare a stratului

Alte Setări

Nivelurile simbolului

For renderers that allow stacked symbol layers (only heatmap doesn't) there is an option to control the rendering order of each symbol's levels.

For most of the renderers, you can access the Symbols levels option by clicking the *Advanced* button below the saved symbols list and choosing *Symbol levels*. For the *Rule-based Renderer* the option is directly available through *Symbols Levels…* button, while for *Point displacement Renderer* renderer the same button is inside the *Rendering settings* dialog.

To activate symbols levels, select the *Enable symbol levels*. Each row will show up a small sample of the combined symbol, its label and the individual symbols layer divided into columns with a number next to it. The numbers represent the rendering order level in which the symbol layer will be drawn. Lower values levels are drawn first, staying at the bottom, while higher values are drawn last, on top of the others.

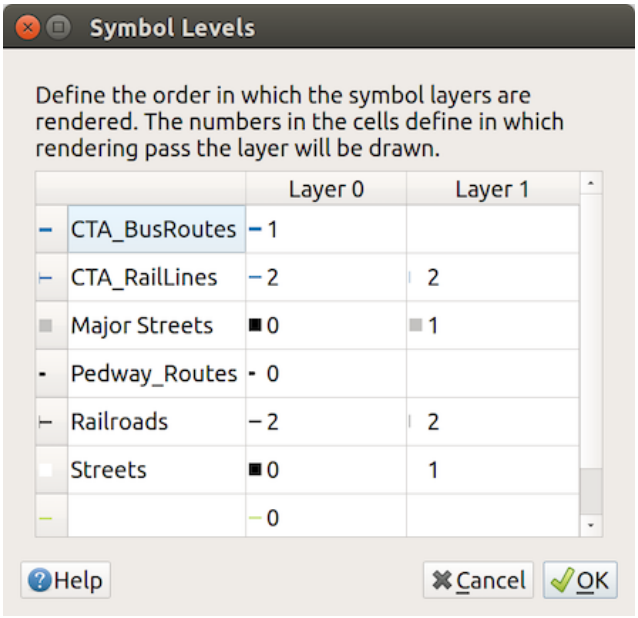


Fig. 16.14: Dialogul Nivelurilor simbolului

Notă: If symbols levels are deactivated, the complete symbols will be drawn according to their respective features order. Overlapping symbols will simply obfuscate to other below. Besides, similar symbols won't „merge" with each other.

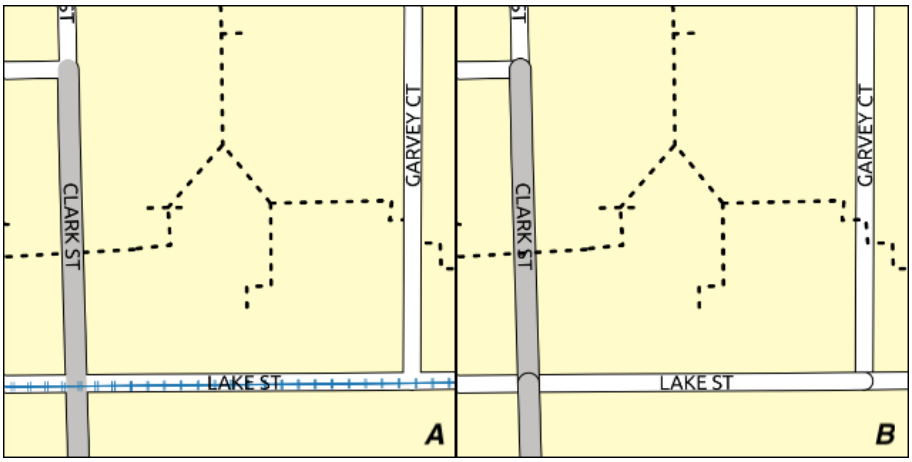


Fig. 16.15: Symbol levels activated (A) and deactivated (B) difference

Data-defined size legend

When a layer is rendered with the *proportional symbol or the multivariate rendering* or when a *scaled size diagram* is applied to the layer, you can allow the display of the scaled symbols in both the *Layers panel* and the *print layout legend*.

To enable the *Data-defined Size Legend* dialog to render symbology, select the eponym option in the *Advanced* button below the saved symbols list. For diagrams, the option is available under the *Legend* tab. The dialog provides the following options to:

- select the type of legend: *Legend not enabled*, *Separated legend items* and *Collapsed legend*. For the latter option, you can select whether the legend items are aligned at the **Bottom** or at the **Center**;

- preview the *symbol to use* for legend representation;
- insert the title in the legend;
- resize the classes to use: by default, QGIS provides you with a legend of five classes (based on natural pretty breaks) but you can apply your own classification using the *Manual size classes* option. Use the and buttons to set your custom classes values and labels.
- For collapsed legend, it's possible to:
  - *Align symbols* in the center or the bottom
  - configure the horizontal leader *Line symbol* from the symbol to the corresponding legend text.

A preview of the legend is displayed in the right panel of the dialog and updated as you set the parameters.

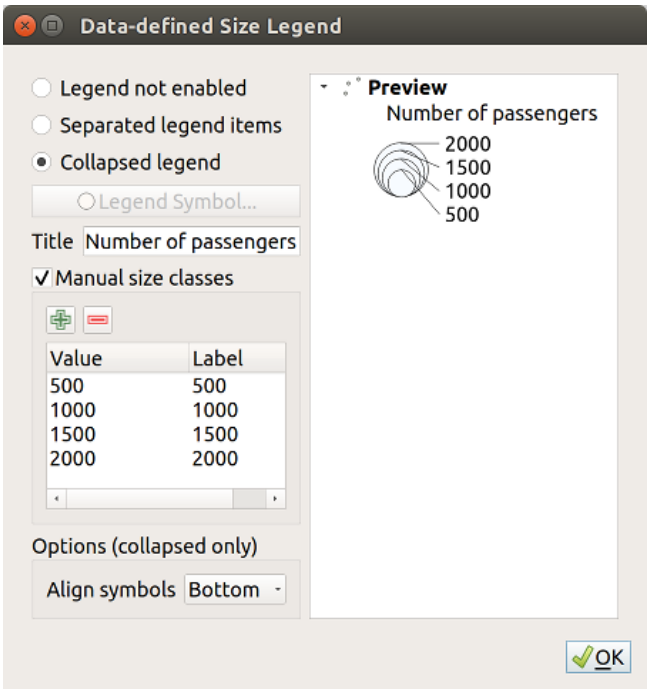


Fig. 16.16: Setting size scaled legend

Notă: Currently, data-defined size legend for layer symbology can only be applied to point layer using single, categorized or graduated symbology.

Animation settings

To allow any symbol to become an *animated symbol*, you can utilize *Animation settings* panel. In this panel, you can enable animation for the symbol and set a specific frame rate for the symbol's redrawing.

1. Start by going to the top symbol level and select *Advanced* menu in the bottom right of the dialog
2. Find *Animation settings* option
3. Check *Is Animated* to enable animation for the symbol
4. Configure the *Frame rate*, i.e. how fast the animation would be played
5. You can now use `@symbol_frame` variable in any sub-symbol data defined property in order to animate that property.

Fow example, setting the symbol's rotation to data defined expression `@symbol_frame % 360` will cause the symbol to rotate over time, with rotation speed dictated by the symbol's frame rate:

Efecte de Desenare

In order to improve layer rendering and avoid (or at least reduce) the resort to other software for final rendering of maps, QGIS provides another powerful functionality: the *Draw Effects* options, which adds paint effects for customizing the visualization of vector layers.

The option is available in the *Layer Properties* ► *Symbology* dialog, under the *Layer rendering* group (applying to the whole layer) or in *symbol layer properties* (applying to corresponding features). You can combine both usage.

Paint effects can be activated by checking the *Draw effects* option and clicking the Customize effects button. That will open the *Effect Properties* Dialog (see Fig. 16.17). The following effect types, with custom options are available:

- **Source**: Draws the feature's original style according to the configuration of the layer's properties. The *Opacity* of its style can be adjusted as well as the *Blend mode* and *Draw mode*. These are common properties for all types of effects.

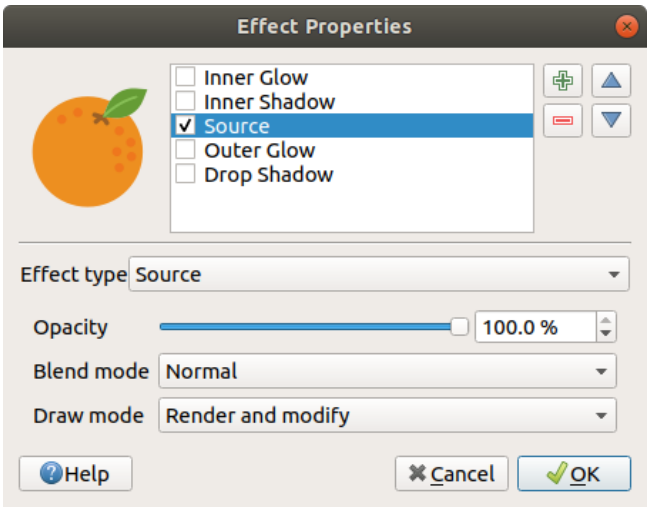


Fig. 16.17: Efecte de Desenare: Dialogul sursă

- **Blur**: Adds a blur effect on the vector layer. The custom options that you can change are the *Blur type* (*Stack blur (fast)* or *Gaussian blur (quality)*) and the *Blur strength*.

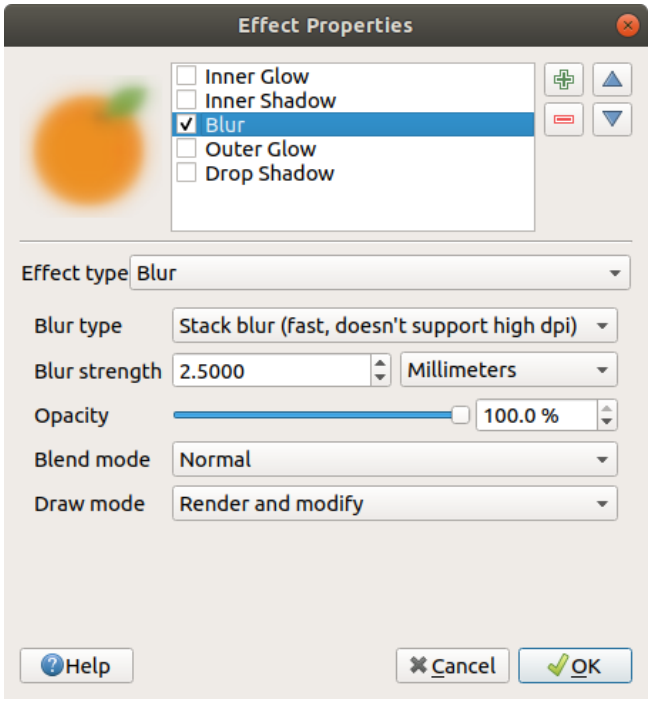


Fig. 16.18: Draw Effects: Blur dialog

- **Colorise**: This effect can be used to make a version of the style using one single hue. The base will always be a grayscale version of the symbol and you can:
  - Use the *Grayscale* to select how to create it: options are «By lightness», «By luminosity», «By average» and «Off».
  - If *Colorise* is selected, it will be possible to mix another color and choose how strong it should be.
  - Control the *Brightness*, *Contrast* and *Saturation* levels of the resulting symbol.

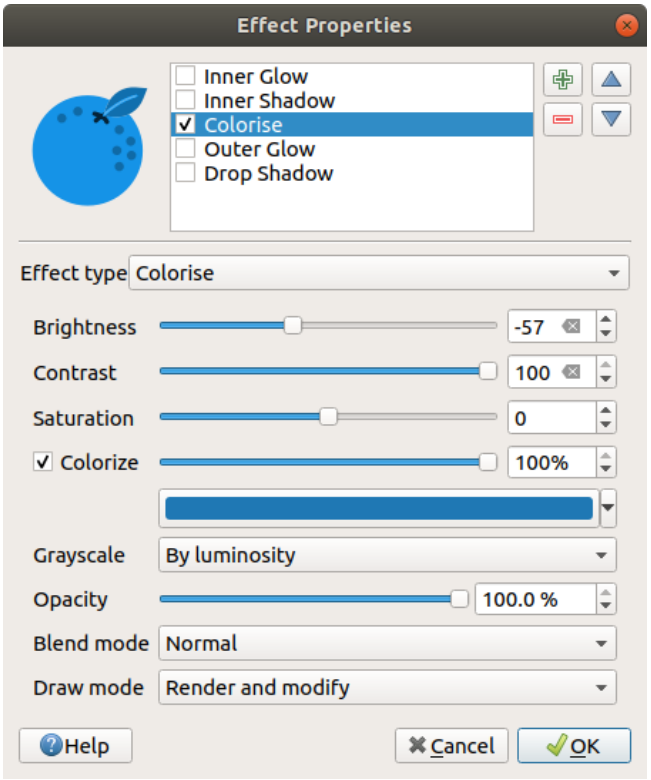


Fig. 16.19: Efecte de Desenare: Dialogul de colorizare

- **Drop Shadow**: Using this effect adds a shadow on the feature, which looks like adding an extra dimension. This effect can be customized by changing the *Offset* angle and distance, determining where the shadow shifts towards to and the proximity to the source object. *Drop Shadow* also has the option to change the *Blur radius* and the *Color* of the effect.

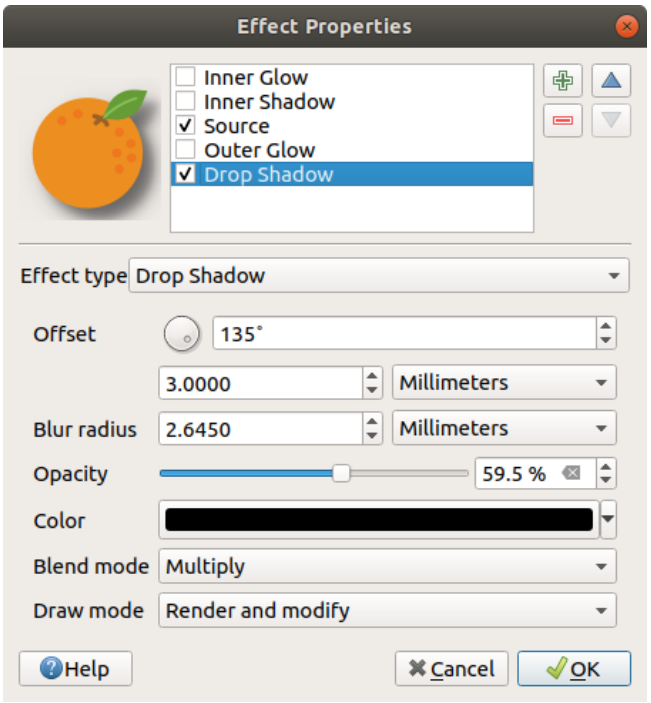


Fig. 16.20: Draw Effects: Drop Shadow dialog

- **Inner Shadow**: This effect is similar to the *Drop Shadow* effect, but it adds the shadow effect on the inside of the edges of the feature. The available options for customization are the same as the *Drop Shadow* effect.

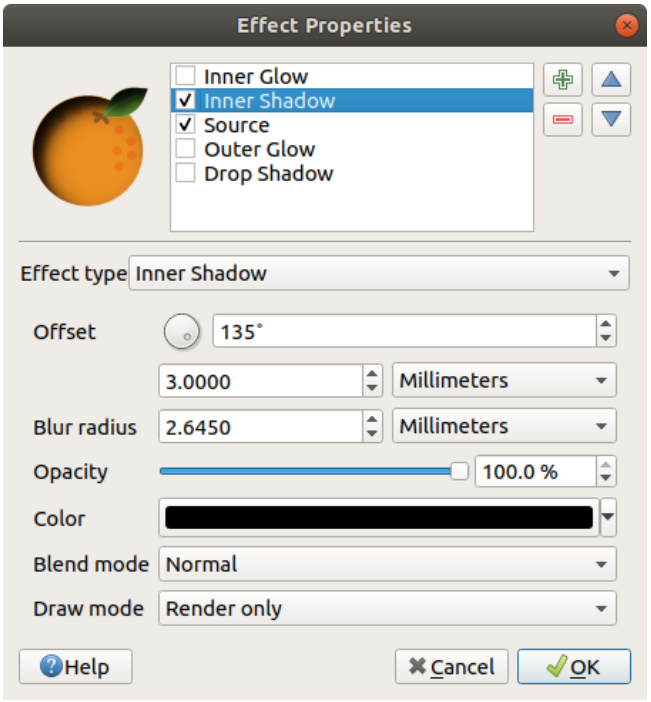


Fig. 16.21: Efecte de Desenare: Dialogul Umbrei Interioare

- **Inner Glow**: Adds a glow effect inside the feature. This effect can be customized by adjusting the *Spread* (width) of the glow, or the *Blur radius*. The latter specifies the proximity from the edge of the feature where you want any blurring to happen. Additionally, there are options to customize the color of the glow using a *Single color* or a *Color ramp*.

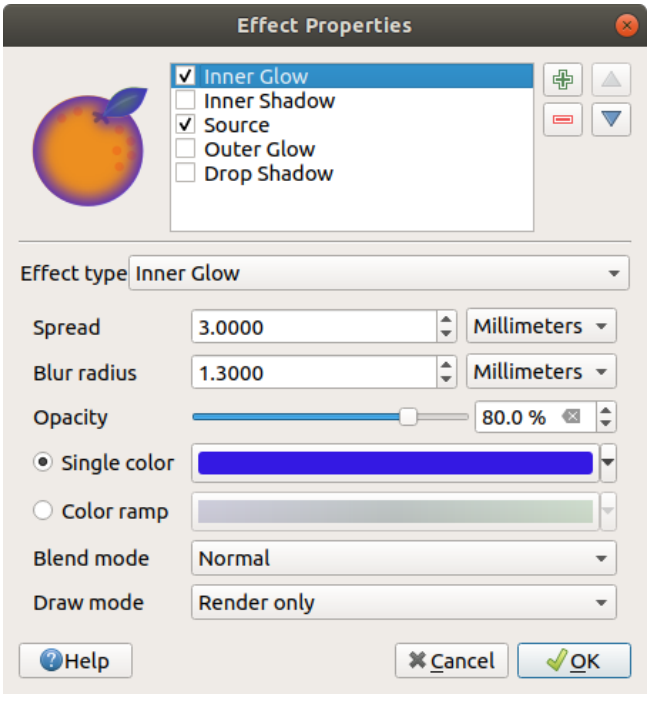


Fig. 16.22: Efecte de Desenare: Dialogul Strălucirii Interioare

- **Outer Glow**: This effect is similar to the *Inner Glow* effect, but it adds the glow effect on the outside of the edges of the feature. The available options for customization are the same as the *Inner Glow* effect.

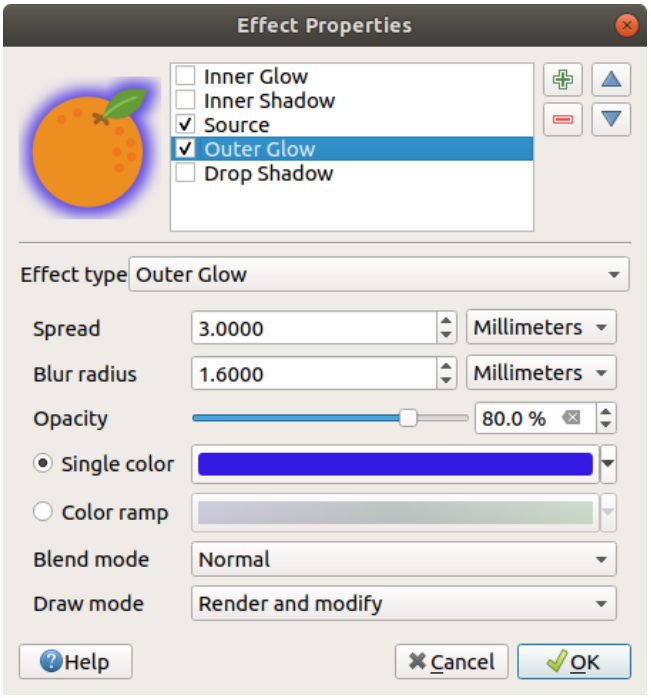


Fig. 16.23: Efecte de Desenare: Dialogul Strălucirii Exterioare

- **Transform**: Adds the possibility of transforming the shape of the symbol. The first options available for customization are the *Reflect horizontal* and *Reflect vertical*, which actually create a reflection on the horizontal and/or vertical axes. The other options are:
  - *Shear X,Y*: Slants the feature along the X and/or Y axis.
  - *Scale X,Y*: Enlarges or minimizes the feature along the X and/or Y axis by the given percentage.
  - *Rotation*: Turns the feature around its center point.
  - and *Translate X,Y* changes the position of the item based on a distance given on the X and/or Y axis.

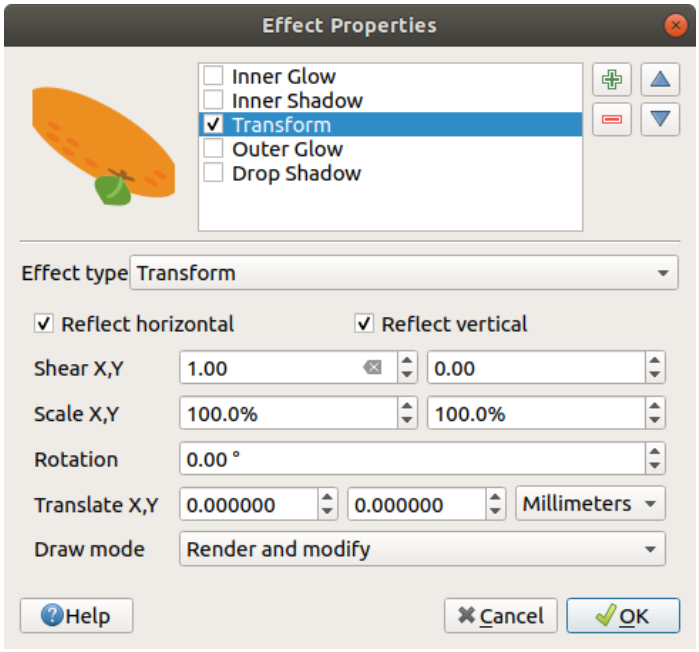


Fig. 16.24: Efecte de Desenare: Dialogul de transformare

One or more effect types can be used at the same time. You (de)activate an effect using its checkbox in the effects list. You can change the selected effect type by using the *Effect type* option. You can reorder the effects using Move up and Move down buttons, and also add/remove effects using the Add new effect and Remove effect buttons.

There are some common options available for all draw effect types. *Opacity* and *Blend mode* options work similar to the ones described in *Randarea stratului* and can be used in all draw effects except for the transform one.

There is also a *Draw mode* option available for every effect, and you can choose whether to render and/or modify the symbol, following some rules:

- Effects render from top to bottom.
- *Render only* mode means that the effect will be visible.
- *Modifier only* mode means that the effect will not be visible but the changes that it applies will be passed to the next effect (the one immediately below).
- The *Render and Modify* mode will make the effect visible and pass any changes to the next effect. If the effect is at the top of the effects list or if the immediately above effect is not in modify mode, then it will use the original source symbol from the layers properties (similar to source).

16.1.4 Proprietățile Etichetelor

The *Labels* properties provides you with all the needed and appropriate capabilities to configure smart labeling on vector layers. This dialog can also be accessed from the *Layer Styling* panel, or using the Layer Labeling Options button of the **Labels toolbar**.

The first step is to choose the labeling method from the drop-down list. Available methods are:

- *No labels*: the default value, showing no labels from the layer
- *Single labels*: Show labels on the map using a single attribute or an expression
- *Rule-based labeling*

- and *Blocking*: allows to set a layer as just an obstacle for other layer's labels without rendering any labels of its own.

The next steps assume you select the *Single labels* option, opening the following dialog.

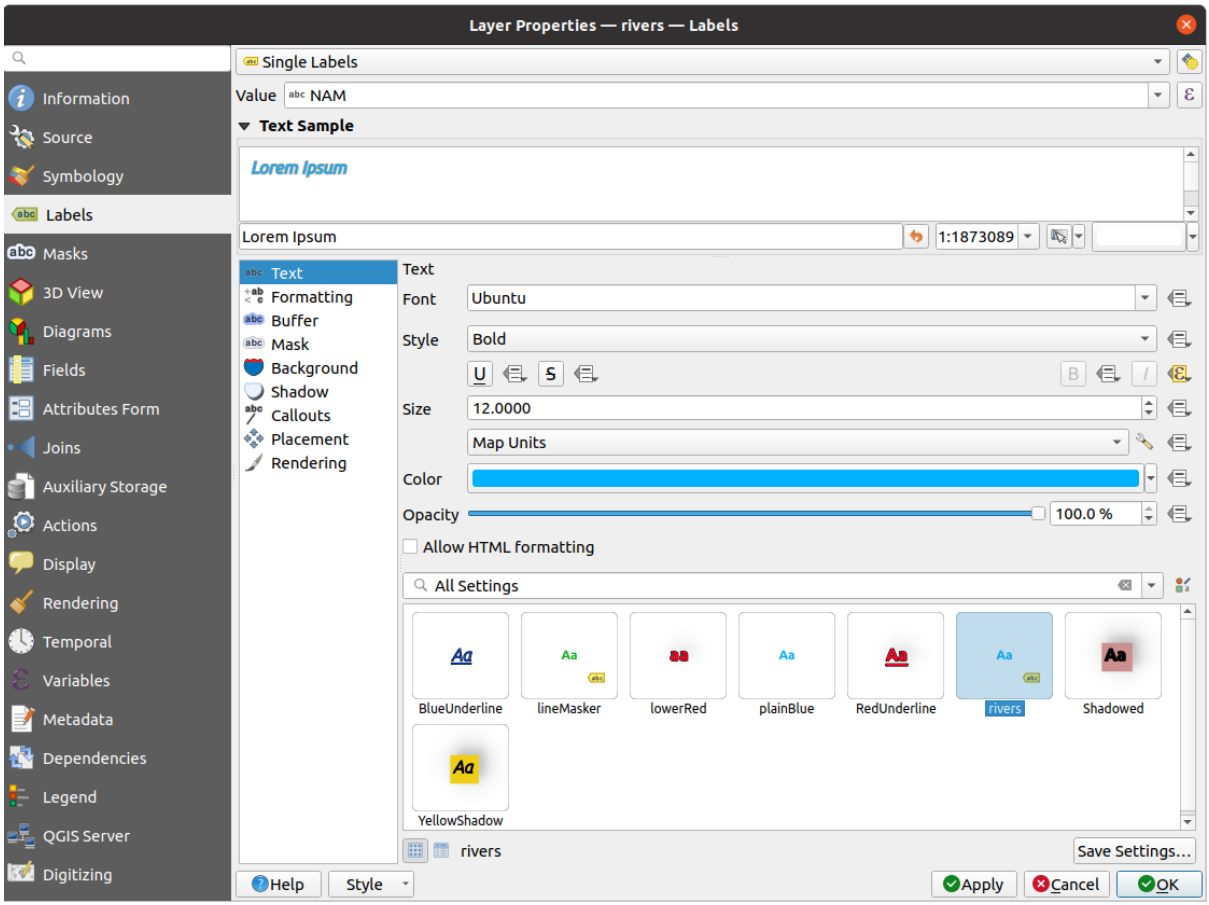


Fig. 16.25: Layer labeling settings - Single labels

At the top of the dialog, a *Value* drop-down list is enabled. You can select an attribute column to use for labeling. By default, the *display field* is used. Click ε if you want to define labels based on expressions - See *Definirea etichetelor pe baza expresiilor*.

Notă: Labels with their formatting can be displayed as entries in the legends, if enabled in the *Legend* tab.

Below are displayed options to customize the labels, under various tabs:

- *Text*
- *Formatting*
- *Buffer*
- *Mask*
- *Background*
- *Shadow*
- *Callouts*

- *Placement*
- *Rendering*

Description of how to set each property is exposed at *Setting a label*.

Setting the automated placement engine

You can use the automated placement settings to configure a project-level automated behavior of the labels. In the top right corner of the *Labels* tab, click the Automated placement settings (applies to all layers) button, opening a dialog with the following options:

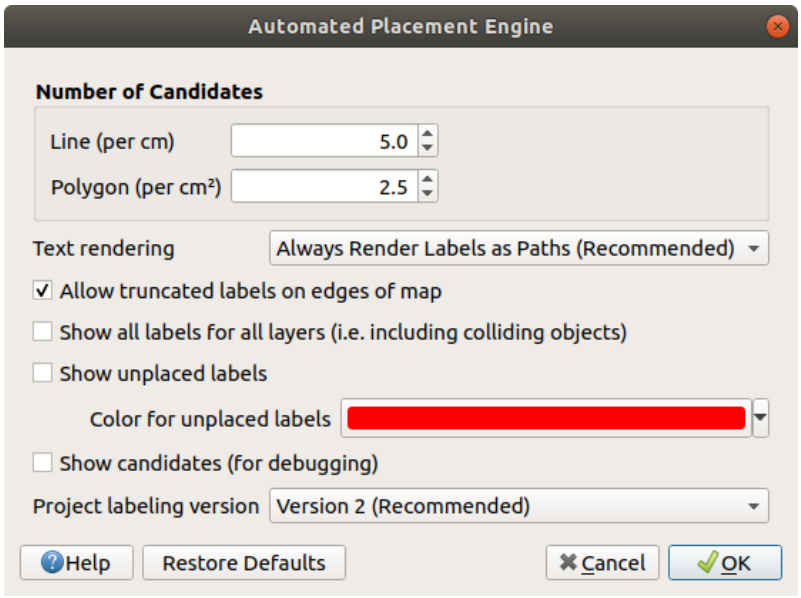


Fig. 16.26: The labels automated placement engine

- *Number of candidates*: calculates and assigns to line and polygon features the number of possible labels placement based on their size. The longer or wider a feature is, the more candidates it has, and its labels can be better placed with less risk of collision.
- *Text rendering*: sets the default value for label rendering widgets when *exporting a map canvas* or *a layout* to PDF or SVG. If *Always render labels as text* is selected then labels can be edited in external applications (e.g. Inkscape) as normal text. BUT the side effect is that the rendering quality is decreased, and there are issues with rendering when certain text settings like buffers are in place. That's why *Always render labels as paths (recommended)* which exports labels as outlines, is recommended.
- *Allow truncated labels on edges of map*: controls whether labels which fall partially outside of the map extent should be rendered. If checked, these labels will be shown (when there's no way to place them fully within the visible area). If unchecked then partially visible labels will be skipped. Note that this setting has no effects on labels» display in the *layout map item*.
- *Show all labels for all layers (i.e. including colliding objects)*. Note that this option can be also set per layer (see *Rendering tab*)
- *Show unplaced labels*: allows to determine whether any important labels are missing from the maps (e.g. due to overlaps or other constraints). They are displayed using a customizable color.
- *Show candidates (for debugging)*: controls whether boxes should be drawn on the map showing all the candidates generated for label placement. Like the label says, it's useful only for debugging and testing the effect different labeling settings have. This could be handy for a better manual placement with tools from the *label toolbar*.

- *Project labeling version*: QGIS supports two different versions of label automatic placement:
  - *Version 1*: the old system (used by QGIS versions 3.10 and earlier, and when opening projects created in these versions in QGIS 3.12 or later). Version 1 treats label and obstacle priorities as „rough guides" only, and it's possible that a low-priority label will be placed over a high-priority obstacle in this version. Accordingly, it can be difficult to obtain the desired labeling results when using this version and it is thus recommended only for compatibility with older projects.
  - *Version 2 (recommended)*: this is the default system in new projects created in QGIS 3.12 or later. In version 2, the logic dictating when labels are allowed to overlap *obstacles* has been reworked. The newer logic forbids any labels from overlapping any obstacles with a greater obstacle weight compared to the label's priority. As a result, this version results in much more predictable and easier to understand labeling results.

Etichetarea bazată pe reguli

With rule-based labeling multiple label configurations can be defined and applied selectively on the base of expression filters and scale range, as in *Rule-based rendering*.

Pentru a crea o regulă:

1. Select the **Rule-based labeling** option in the main drop-down list from the *Labels* tab
2. Click the Add rule button at the bottom of the dialog.
3. Fill the new dialog with:
   - *Description*: a text used to identify the rule in the *Labels* tab and as a *label legend entry* in the print layout legend
   - *Filter*: an expression to select the features to apply the label settings to
   - If there are rules already set, the *Else* option can be used to select all the features not matching any filter of the rules in the same group.
4. You can set a *scale range* in which the label rule should be applied.
5. The options available under the *Labels* group box are the usual *label settings*. Configure them and press *OK*.

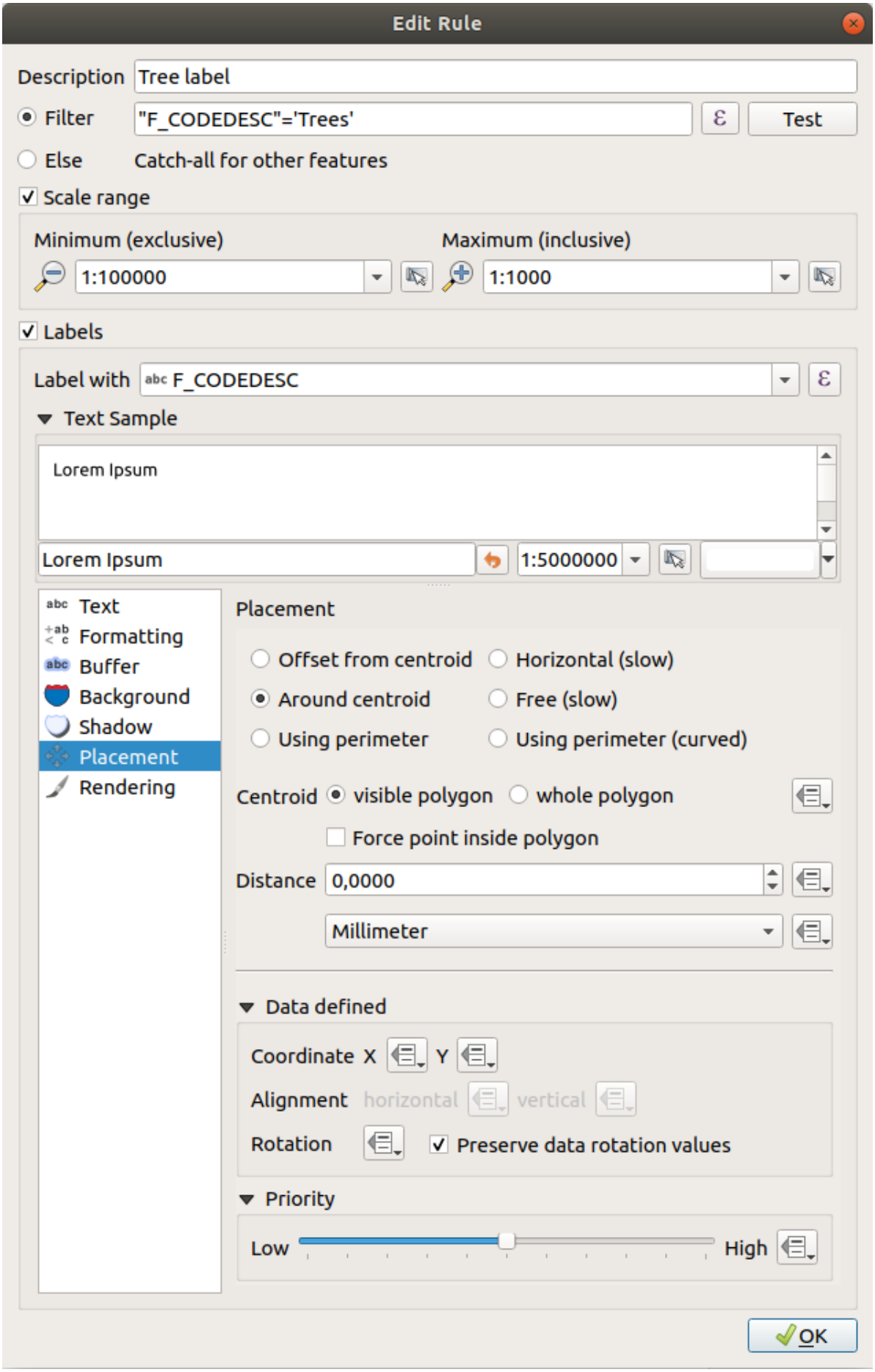


Fig. 16.27: Setările unei reguli

A summary of existing rules is shown in the main dialog (see Fig. 16.28). You can add multiple rules, reorder or imbricate them with a drag-and-drop. You can as well remove them with the button or edit them with button or a double-click.

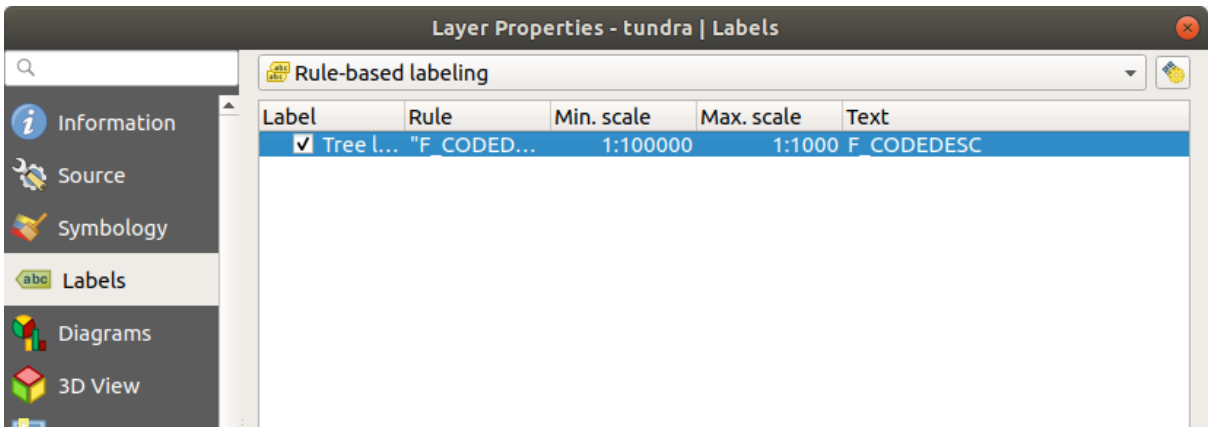


Fig. 16.28: Panoul de etichetare bazată pe reguli

Definirea etichetelor pe baza expresiilor

Whether you choose single or rule-based labeling type, QGIS allows using expressions to label features.

Assuming you are using the *Single labels* method, click the button near the *Value* drop-down list in the *Labels* tab of the properties dialog.

In Fig. 16.29, you see a sample expression to label the alaska trees layer with tree type and area, based on the field «VEGDESC», some descriptive text, and the function `$area` in combination with `format_number()` to make it look nicer.

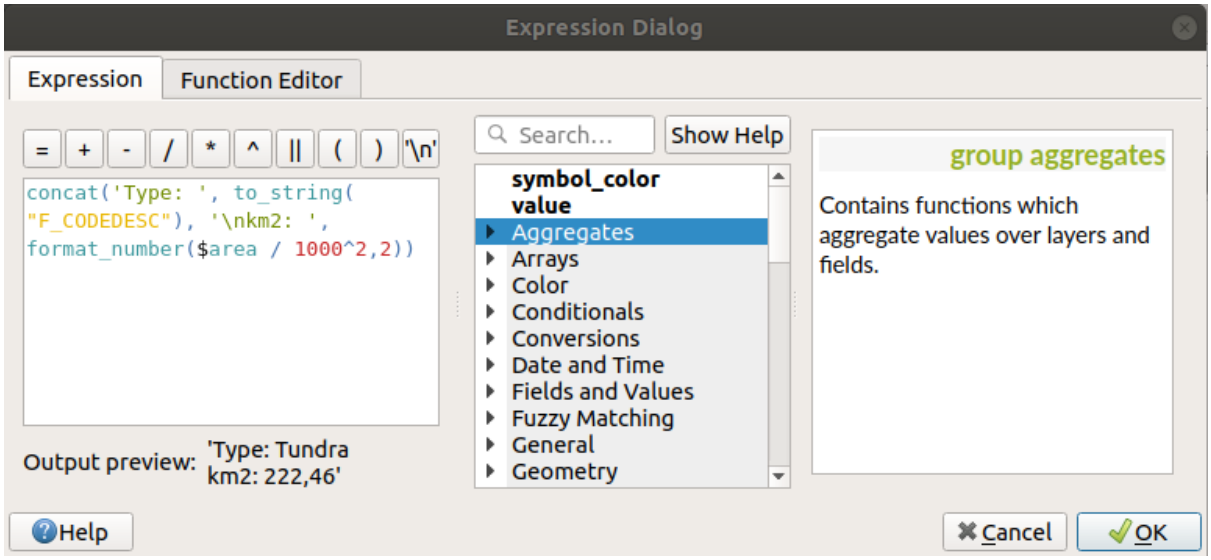


Fig. 16.29: Folosirea expresiilor pentru etichetare

Expression based labeling is easy to work with. All you have to take care of is that:

- You may need to combine all elements (strings, fields, and functions) with a string concatenation function such as `concat`, `+` or `||`. Be aware that in some situations (when null or numeric value are involved) not all of these tools will fit your need.

- Strings are written in «single quotes».
- Fields are written in „double quotes” or without any quote.

Să aruncăm o privire asupra câtorva exemple:

1. Label based on two fields «name» and «place» with a comma as separator:

```
"name" || ', ' || "place"
```

   Returnează:

```
John Smith, Paris
```

2. Label based on two fields «name» and «place» with other texts:

```
'My name is ' + "name" + 'and I live in ' + "place"
'My name is ' || "name" || 'and I live in ' || "place"
concat('My name is ', name, ' and I live in ', "place")
```

   Returnează:

```
My name is John Smith and I live in Paris
```

3. Label based on two fields «name» and «place» with other texts combining different concatenation functions:

```
concat('My name is ', name, ' and I live in ' || place)
```

   Returnează:

```
My name is John Smith and I live in Paris
```

   Or, if the field «place» is NULL, returns:

```
My name is John Smith
```

4. Multi-line label based on two fields «name» and «place» with a descriptive text:

```
concat('My name is ', "name", '\n' , 'I live in ' , "place")
```

   Returnează:

```
My name is John Smith
I live in Paris
```

5. Label based on a field and the $area function to show the place’s name and its rounded area size in a converted unit:

```
'The area of ' || "place" || ' has a size of '
|| round($area/10000) || ' ha'
```

   Returnează:

```
The area of Paris has a size of 10500 ha
```

6. Create a CASE ELSE condition. If the population value in field *population* is <= 50000 it is a town, otherwise it is a city:

```
concat('This place is a ',
CASE WHEN "population" <= 50000 THEN 'town' ELSE 'city' END)
```

   Returnează:

```
This place is a town
```

7. Display name for the cities and no label for the other features (for the „city” context, see example above):

```
CASE WHEN "population" > 50000 THEN "NAME" END
```

Returnează:

```
Paris
```

As you can see in the expression builder, you have hundreds of functions available to create simple and very complex expressions to label your data in QGIS. See *Expresii* chapter for more information and examples on expressions.

Folosirea suprascrierii definițiilor de date pentru etichetare

With the Data defined override function, the settings for the labeling are overridden by entries in the attribute table or expressions based on them. This feature can be used to set values for most of the labeling options described above.

For example, using the Alaska QGIS sample dataset, let's label the `airports` layer with their name, based on their militarian `USE`, i.e. whether the airport is accessible to :

- military people, then display it in gray color, size 8;
- others, then show in blue color, size 10.

To do this, after you enabled the labeling on the `NAME` field of the layer (see *Setting a label*):

1. Activate the *Text* tab.
2. Click on the icon next to the *Size* property.
3. Select *Edit...* and type:

```
CASE
  WHEN "USE" like '%Military%' THEN 8 -- because compatible values are
↪'Military'
                                      -- and 'Joint Military/Civilian'
  ELSE 10
END
```

4. Press *OK* to validate. The dialog closes and the button becomes meaning that an rule is being run.
5. Then click the button next to the color property, type the expression below and validate:

```
CASE
  WHEN "USE" like '%Military%' THEN '150, 150, 150'
  ELSE '0, 0, 255'
END
```

Likewise, you can customize any other property of the label, the way you want. See more details on the Data-define override widget's description and manipulation in *Configurarea suprascrierii definită de date* section.

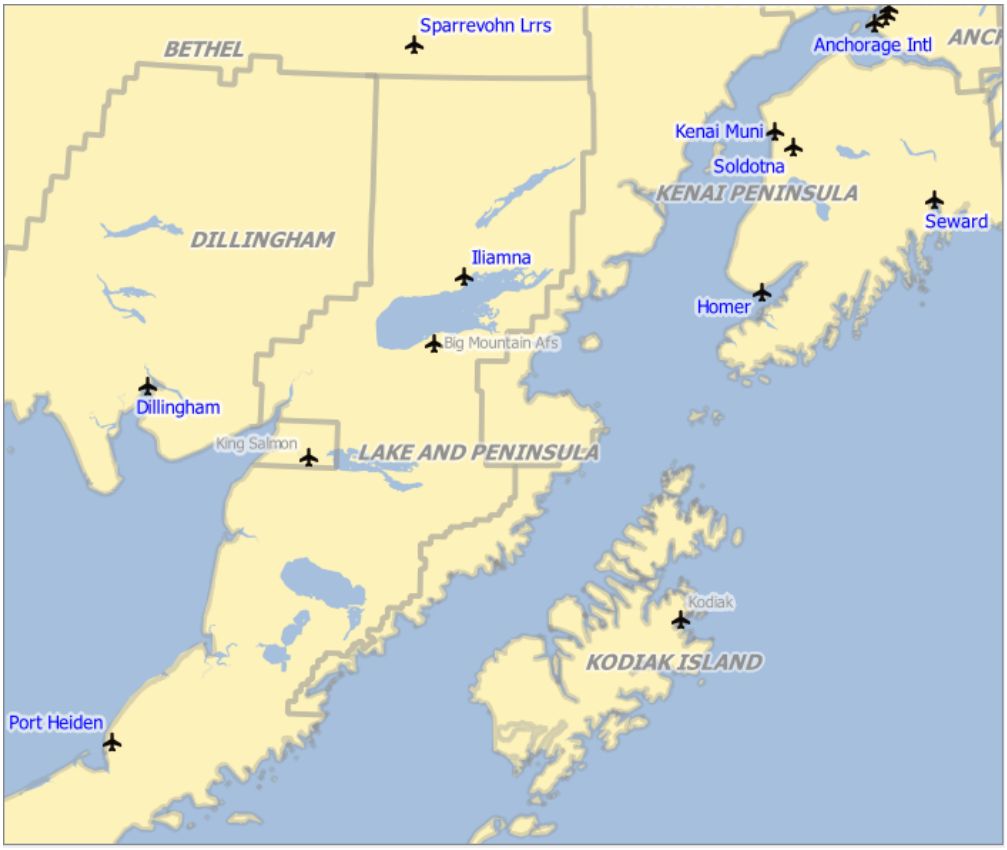


Fig. 16.30: Airports labels are formatted based on their attributes

Sfat: Use the data-defined override to label every part of multi-part features

There is an option to set the labeling for multi-part features independently from your label properties. Choose the *Rendering*, `Feature options`, go to the Data-define override button next to the checkbox *Label every part of multipart-features* and define the labels as described in *Configurarea suprascrierii definită de date*.

The Label Toolbar

The *Label Toolbar* provides some tools to manipulate *label* (including their *callout*) or *diagram* properties:

Fig. 16.31: The Label toolbar

- Highlight Pinned Labels, Diagrams and Callouts. If the vector layer of the item is editable, then the highlighting is green, otherwise it's blue.
- Toggle Display of Unplaced Labels: Allows to determine whether any important labels are missing from the maps (e.g. due to overlaps or other constraints). They are displayed with a customizable color (see *Setting the automated placement engine*).
- Pin/Unpin Labels and Diagrams. By clicking or draging an area, you pin overlaid items. If you click or drag an area holding `Shift`, the items are unpinned. Finally, you can also click or drag an area holding `Ctrl` to toggle

their pin status.

- Show/Hide Labels and Diagrams. If you click on the items, or click and drag an area holding `Shift`, they are hidden. When an item is hidden, you just have to click on the feature to restore its visibility. If you drag an area, all the items in the area will be restored.
- Move a Label, Diagram or Callout: click to select the item and click to move it to the desired place. The new coordinates are stored in *auxiliary fields*. Selecting the item with this tool and hitting the `Delete` key will delete the stored position value.
- Rotate a Label. Click to select the label and click again to apply the desired rotation. Likewise, the new angle is stored in an auxiliary field. Selecting a label with this tool and hitting the `Delete` key will delete the rotation value of this label.
- Change Label Properties. It opens a dialog to change the clicked label properties; it can be the label itself, its coordinates, angle, font, size, multiline alignment … as long as this property has been mapped to a field. Here you can set the option to *Label every part of a feature*.

Atenționare: Label tools overwrite current field values

Using the *Label toolbar* to customize the labeling actually writes the new value of the property in the mapped field. Hence, be careful to not inadvertently replace data you may need later!

Notă: The *Auxiliary Storage Properties* mechanism may be used to customize labeling (position, and so on) without modifying the underlying data source.

Customize the labels from the map canvas

Combined with the *Label Toolbar*, the data defined override setting helps you manipulate labels in the map canvas (move, edit, rotate). We now describe an example using the data-defined override function for the Move Label, Diagram or Callout function (see Fig. 16.32).

1. Importă `lakes.shp` din setul de date eșantion al QGIS.
2. Double-click the layer to open the Layer Properties. Click on *Labels* and *Placement*. Select *Offset from centroid*.
3. Look for the *Data defined* entries. Click the icon to define the field type for the *Coordinate*. Choose `xlabel` for X and `ylabel` for Y. The icons are now highlighted in yellow.

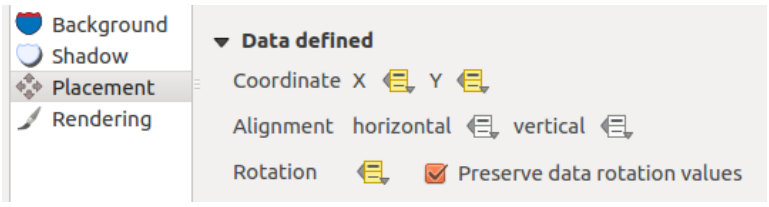


Fig. 16.32: Labeling of vector polygon layers with data-defined override

4. Transfocare către un lac.
5. Set editable the layer using the Toggle Editing button.

6. Go to the Label toolbar and click the icon. Now you can shift the label manually to another position (see Fig. 16.33). The new position of the label is saved in the `xlabel` and `ylabel` columns of the attribute table.
7. It's also possible to add a line connecting each lake to its moved label using:
   - the label's *callout property*
   - or the *geometry generator symbol layer* with the expression below:

```
make_line( centroid( $geometry ), make_point( "xlabel", "ylabel" ) )
```

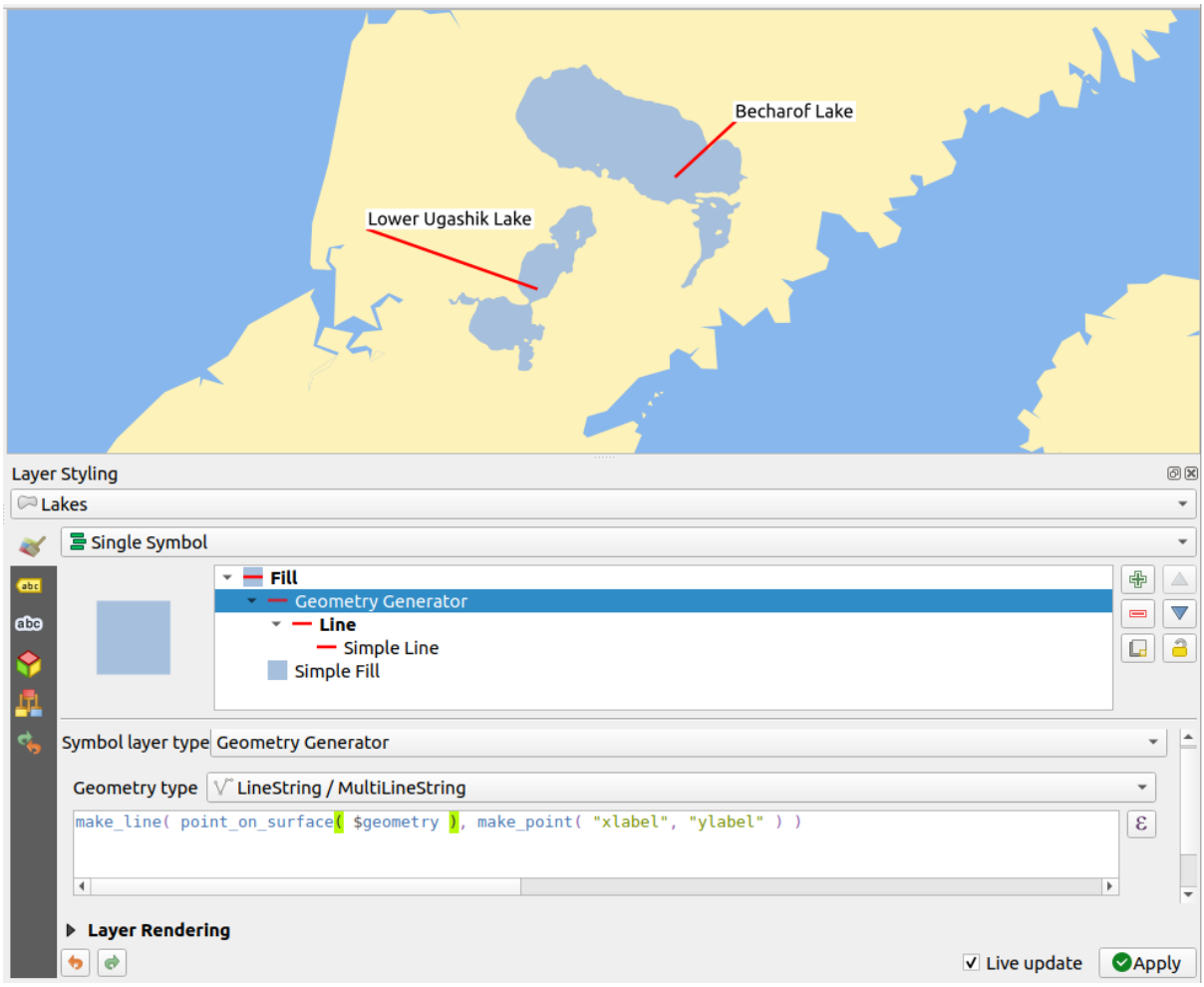


Fig. 16.33: Moved labels

Notă: The *Auxiliary Storage Properties* mechanism may be used with data-defined properties without having an editable data source.

16.1.5 Proprietățile Diagramelor

The *Diagrams* tab allows you to add a graphic overlay to a vector layer (see Fig. 16.34).

The current core implementation of diagrams provides support for:

- *No diagrams*: the default value with no diagram displayed over the features;
- *Pie chart*, a circular statistical graphic divided into slices to illustrate numerical proportion. The arc length of each slice is proportional to the quantity it represents;
- abc *Text diagram*, a horizontaly divided circle showing statistics values inside;
- *Histogram*, bars of varying colors for each attribute aligned next to each other
- *Stacked bars*, Stacks bars of varying colors for each attribute on top of each other vertically or horizontally

In the top right corner of the *Diagrams* tab, the Automated placement settings (applies to all layers) button provides means to control diagram *labels placement* on the map canvas.

Sfat: Switch quickly between types of diagrams

Given that the settings are almost common to the different types of diagram, when designing your diagram, you can easily change the diagram type and check which one is more appropriate to your data without any loss.

For each type of diagram, the properties are divided into several tabs:

- *Atribute*
- *Rendering*
- *Dimensiune*
- *Plasament*
- *Opțiuni*
- *Legendă*

Atribute

Attributes defines which variables to display in the diagram. Use add item button to select the desired fields into the «Assigned Attributes» panel. Generated attributes with *Expresii* can also be used.

You can move up and down any row with click and drag, sorting how attributes are displayed. You can also change the label in the «Legend» column or the attribute color by double-clicking the item.

This label is the default text displayed in the legend of the print layout or of the layer tree.

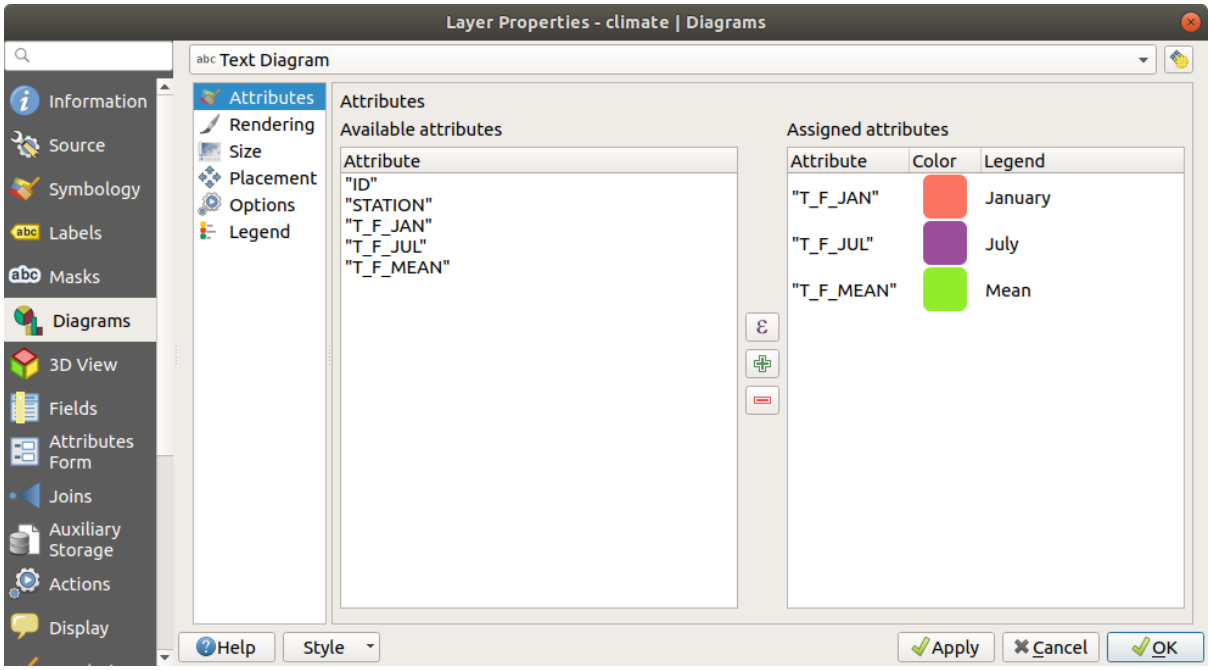


Fig. 16.34: Diagram properties - Attributes tab

Randare

Rendering defines how the diagram looks like. It provides general settings that do not interfere with the statistic values such as:

- opacitatea graficului, lățimea și culoarea conturului;
- depending on the type of diagram:
  - for histogram and stacked bars, the width of the bar and the spacing between the bars. You may want to set the spacing to `0` for stacked bars. Moreover, the *Axis line symbol* can be made visible on the map canvas and customized using *line symbol properties*.
  - for text diagram, the circle background color and the *font* used for texts;
  - for pie charts, the *Start angle* of the first slice and their *Direction* (clockwise or not).
- the use of *paint effects* on the graphics.

In this tab, you can also manage and fine tune the diagram visibility with different options:

- *Diagram z-index*: controls how diagrams are drawn on top of each other and on top of labels. A diagram with a high index is drawn over diagrams and labels;
- *Show all diagrams*: shows all the diagrams even if they overlap each other;
- *Show diagram*: allows only specific diagrams to be rendered;
- *Always Show*: selects specific diagrams to always render, even when they overlap other diagrams or map labels;
- setting the *Scale dependent visibility*;

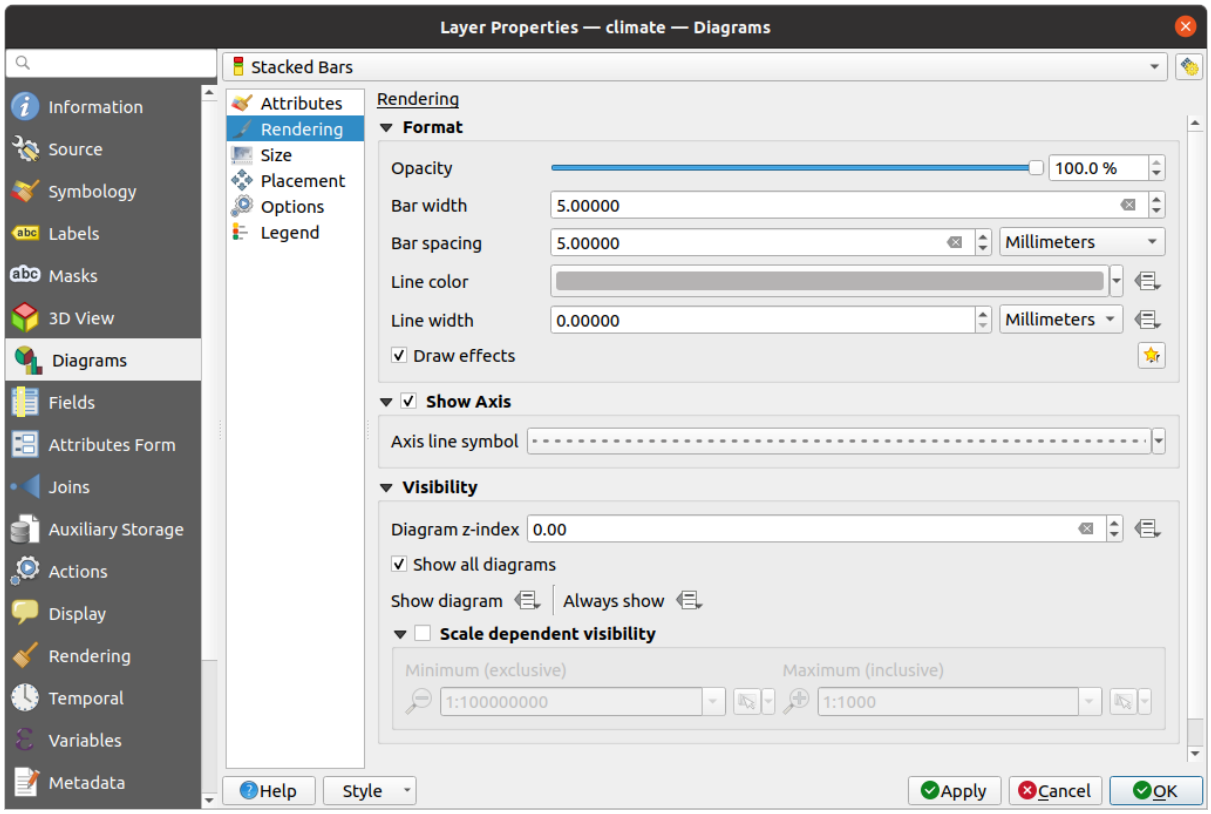


Fig. 16.35: Proprietățile Diagramei - Fila Randării

Dimensiuni

Size is the main tab to set how the selected statistics are represented. The diagram size *units* can be «Millimeters», «Points», «Pixels», «Map Units» or «Inches». You can use:

- *Fixed size*, a unique size to represent the graphic of all the features (not available for histograms)
- or *Scaled size*, based on an expression using layer attributes:
  1. In *Attribute*, select a field or build an expression
  2. Press *Find* to return the *Maximum value* of the attribute or enter a custom value in the widget.
  3. For histogram and stacked bars, enter a *Bar length* value, used to represent the *Maximum value* of the attributes. For each feature, the bar length will then be scaled linearly to keep this matching.
  4. For pie chart and text diagram, enter a *Size* value, used to represent the *Maximum value* of the attributes. For each feature, the circle area or diameter will then be scaled linearly to keep this matching (from `0`). A *Minimum size* can however be set for small diagrams.

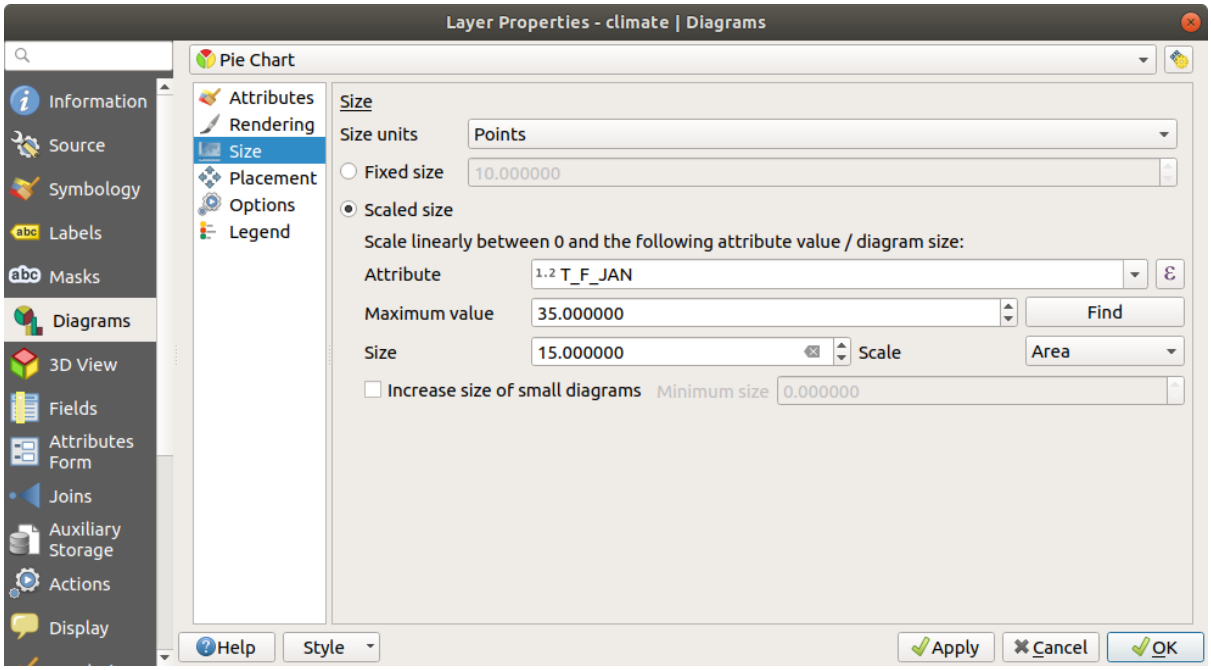


Fig. 16.36: Diagram properties - Size tab

Poziționare

Placement defines the diagram position. Depending on the layer geometry type, it offers different options for the placement (more details at *Placement*):

- *Around point* or *Over point* for point geometry. The former variable requires a radius to follow.
- *Around line* or *Over line* for line geometry. Like point feature, the first variable requires a distance to respect and you can specify the diagram placement relative to the feature («above», «on» and/or «below» the line) It's possible to select several options at once. In that case, QGIS will look for the optimal position of the diagram. Remember that you can also use the line orientation for the position of the diagram.
- *Around centroid* (at a set *Distance*), *Over centroid*, *Using perimeter* and *Inside polygon* are the options for polygon features.

The *Coordinate* group provides direct control on diagram placement, on a feature-by-feature basis, using their attributes or an expression to set the *X* and *Y* coordinate. The information can also be filled using the *Move labels and diagrams* tool.

In the *Priority* section, you can define the placement priority rank of each diagram, ie if there are different diagrams or labels candidates for the same location, the item with the higher priority will be displayed and the others could be left out.

Discourage diagrams and labels from covering features defines features to use as *obstacles*, ie QGIS will try to not place diagrams nor labels over these features. The priority rank is then used to evaluate whether a diagram could be omitted due to a greater weighted obstacle feature.

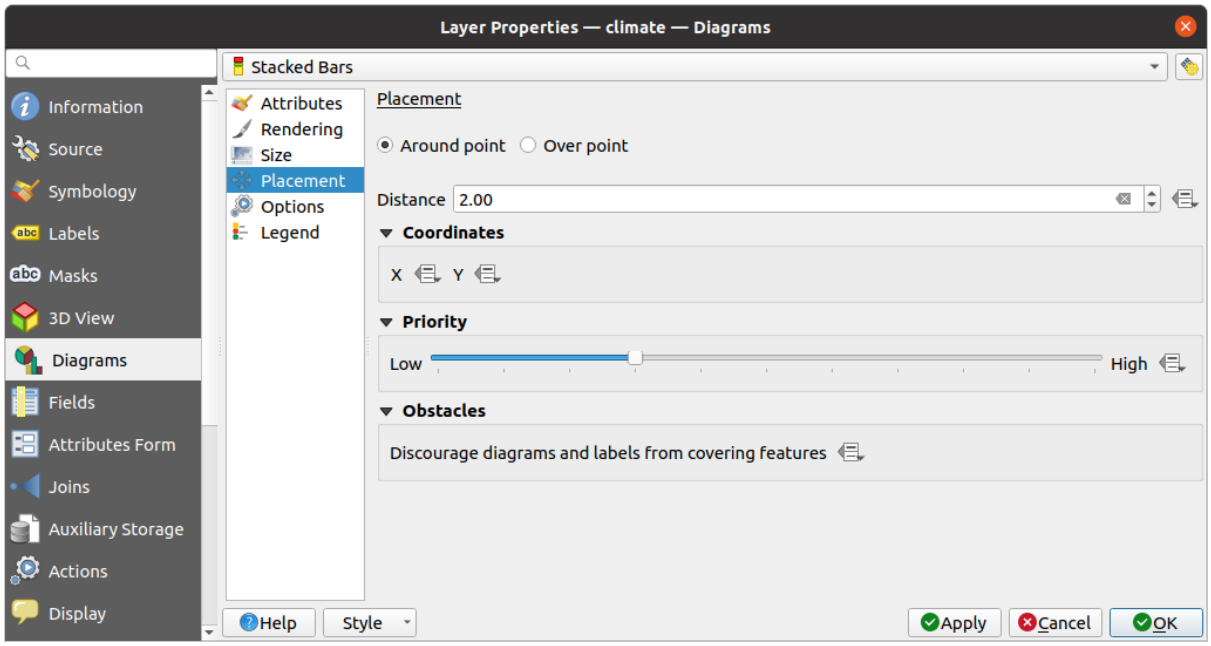


Fig. 16.37: Vector properties dialog with diagram properties, Placement tab

Opţiuni

The *Options* tab has settings for histograms and stacked bars. You can choose whether the *Bar orientation* should be *Up*, *Down*, *Right* or *Left*, for horizontal and vertical diagrams.

Legend

From the *Legend* tab, you can choose to display items of the diagram in the *Layers panel*, and in the *print layout legend*, next to the layer symbology:

- check *Show legend entries for diagram attributes* to display in the legends the `Color` and `Legend` properties, as previously assigned in the *Attributes* tab;
- and, when a *scaled size* is being used for the diagrams, push the *Legend Entries for Diagram Size…* button to configure the diagram symbol aspect in the legends. This opens the *Data-defined Size Legend* dialog whose options are described in *Data-defined size legend*.

When set, the diagram legend items (attributes with color and diagram size) are also displayed in the print layout legend, next to the layer symbology.

16.1.6 Masks Properties

The *Masks* tab helps you configure the current layer symbols overlay with other symbol layers or labels, from any layer. This is meant to improve the readability of symbols and labels whose colors are close and can be hard to decipher when overlapping; it adds a custom and transparent mask around the items to „hide” parts of the symbol layers of the current layer.

To apply masks on the active layer, you first need to enable in the project either *mask symbol layers* or *mask labels*. Then, from the *Masks* tab, check:

- the *Masked symbol layers*: lists in a tree structure all the symbol layers of the current layer. There you can select the symbol layer item you would like to transparently „cut out” when they overlap the selected mask sources
- the *Mask sources* tab: list all the mask labels and mask symbol layers defined in the project. Select the items that would generate the mask over the selected masked symbol layers

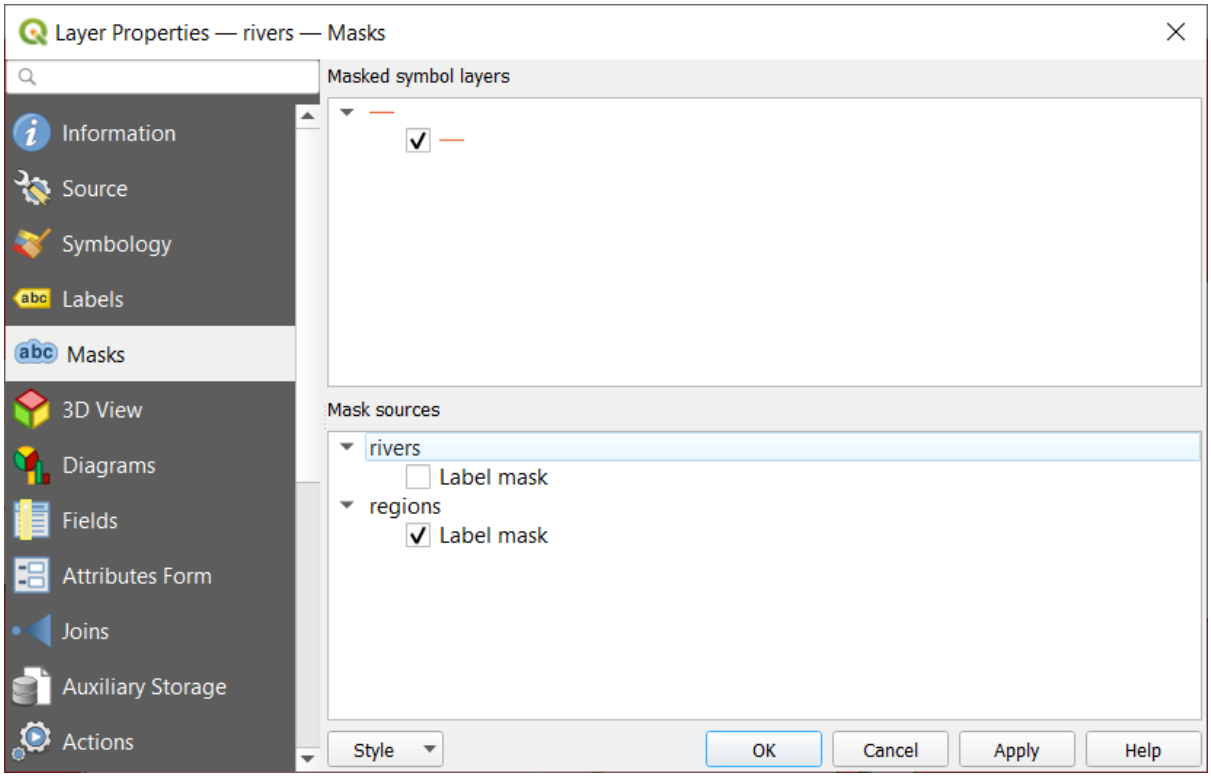


Fig. 16.38: Layer properties - Masks tab

16.1.7 3D View Properties

The *3D View* tab provides settings for vector layers that should be depicted in the *3D Map view* tool.

To display a layer in 3D, select from the combobox at the top of the tab, either:

- *Single symbol*: features are rendered using a common 3D symbol whose properties can be *data-defined* or not. Read details on *setting a 3D symbol* for each layer geometry type.
- *Rule-based*: multiple symbol configurations can be defined and applied selectively based on expression filters and scale range. More details on how-to at *Rule-based rendering*.

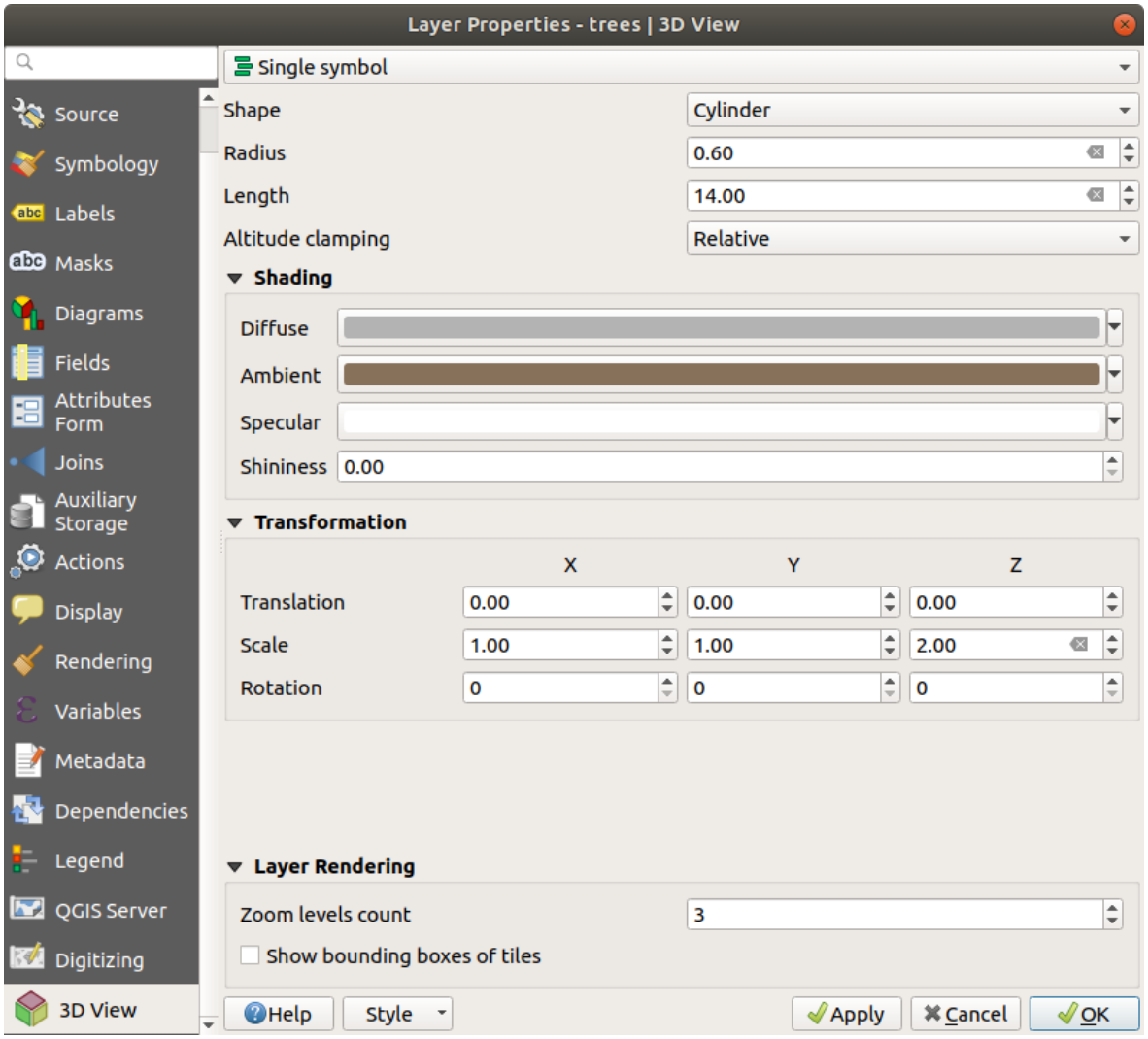


Fig. 16.39: 3D properties of a point layer

Atenție: Prefer the *Elevation* tab for symbol elevation and terrain settings

Features» elevation and altitude related properties (*Altitude clamping*, *Altitude binding*, *Extrusion* or *Height*) in the *3D View* tab inherit their default values from the layer's *Elevation properties* and should preferably be set from within the *Elevation* tab.

For better performance, data from vector layers are loaded in the background, using multithreading, and rendered in tiles whose size can be controlled from the *Layer rendering* section of the tab:

- *Zoom levels count*: determines how deep the quadtree will be. For example, one zoom level means there will be a single tile for the whole layer. Three zoom levels means there will be 16 tiles at the leaf level (every extra zoom level multiplies that by 4). The default is `3` and the maximum is `8`.
- *Show bounding boxes of tiles*: especially useful if there are issues with tiles not showing up when they should.

16.1.8 Fields Properties

The *Fields* tab provides information on fields related to the layer and helps you organize them.

The layer can be made *editable* using the Toggle editing mode. At this moment, you can modify its structure using the New field and Delete field buttons.

When creating New field, the *Comment* option is available only for data sources that allow editing comments (See *Database entries* for more details). You can also set aliases within *Add Field* dialog, for supported OGR formats (GeoPackage and ESRI File Geodatabase).

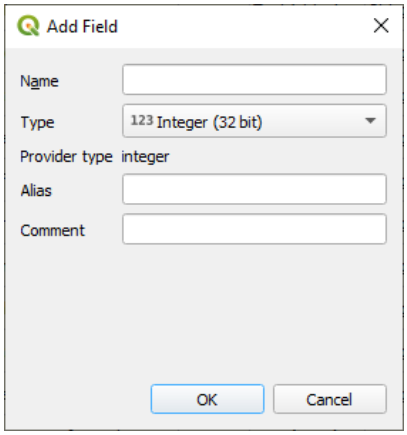


Fig. 16.40: Add Field Dialog

You can also rename fields by double-clicking its name. This is only supported for data providers like PostgreSQL, Oracle, Memory layer and some GDAL layers depending on the GDAL version.

If set in the underlying data source or in the *forms properties*, the field's alias is also displayed. An alias is a human readable field name you can use in the feature form or the attribute table. Aliases are saved in the project file.

Other than the fields contained in the dataset, *virtual fields* and *Auxiliary Storage* included, the *Fields* tab also lists fields from any *joined layers*. Depending on the origin of the field, a different background color is applied to it.

For each listed field, the dialog also lists read-only characteristics such as its *Type*, *Type name*, *Length* and *Precision*`.

Depending on the data provider, you can associate a comment with a field, for example at its creation. This information is retrieved and shown in the *Comment* column and is later displayed when hovering over the field label in a feature form.

Under the *Configuration* column, you can set how the field should behave in certain circumstances:

- `Not searchable`: check this option if you do not want this field to be queried by the *search locator bar*
- `Do not expose via WMS`: check this option if you do not want to display this field if the layer is served as WMS from QGIS server
- `Do not expose via WFS`: check this option if you do not want to display this field if the layer is served as WFS from QGIS server

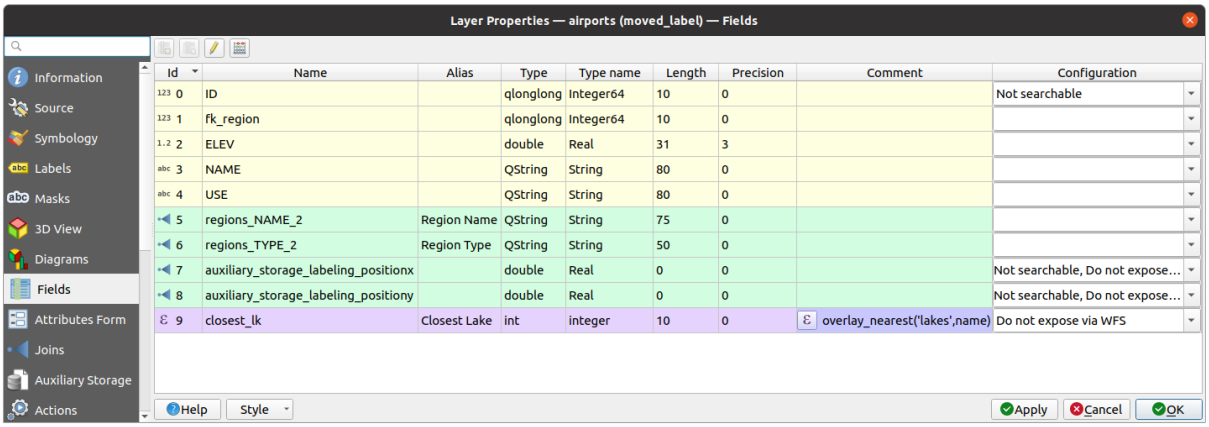


Fig. 16.41: Fields properties tab

16.1.9 Attributes Form Properties

The *Attributes Form* tab helps you set up the form to display when creating new features or querying existing one. You can define:

- the look and the behavior of each field in the feature form or the attribute table (label, widget, constraints…);
- the form's structure (custom or autogenerated):
- extra logic in Python to handle interaction with the form or field widgets.

At the top right of the dialog, you can set whether the form is opened by default when creating new features. This can be configured per layer or globally with the *Suppress attribute form pop-up after feature creation* option in the *Settings* ► *Options* ► *Digitizing* menu.

Customizing a form for your data

By default, when you click on a feature with the Identify Features tool or switch the attribute table to the *form view* mode, QGIS displays a basic form with predefined widgets (generally spinboxes and textboxes — each field is represented on a dedicated row by its label next to the widget). If *relations* are set on the layer, fields from the referencing layers are shown in an embedded frame at the bottom of the form, following the same basic structure.

This rendering is the result of the default `Autogenerate` value of the *Attribute editor layout* setting in the *Layer properties* ► *Attributes Form* tab. This property holds three different values:

- `Autogenerate`: keeps the basic structure of „one row - one field" for the form but allows to customize each corresponding widget.
- `Drag-and-drop designer`: other than widget customization, the form structure can be made more complex eg, with widgets embedded in groups and tabs.
- `Provide ui file`: allows to use a Qt designer file, hence a potentially more complex and fully featured template, as feature form.

Formularul autogenerat

When the `Autogenerate` option is on, the *Available widgets* panel shows lists of fields (from the layer and its relations) that would be shown in the form. Select a field and you can configure its appearance and behavior in the right panel:

- adding *custom label and automated checks* to the field;
- setting a *particular widget* to use.

The drag and drop designer

The drag and drop designer allows you to create a form with several containers (tabs or groups) to present the attribute fields or other widgets that are not directly linked to a particular field (like the HTML/QML widgets or the *actions* defined for the layer), as shown for example in Fig. 16.42.

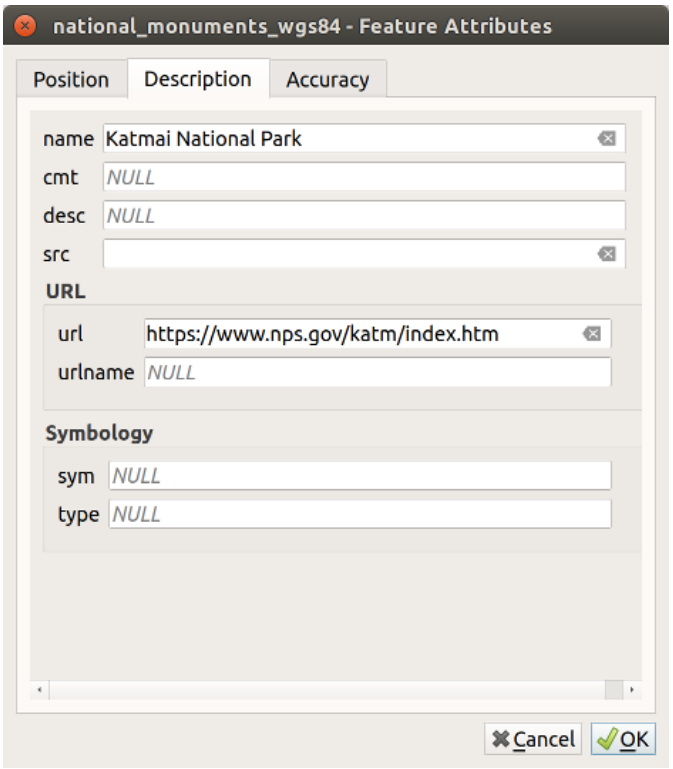


Fig. 16.42: Resulting built-in form with tabs and named groups

1. Choose `Drag and drop designer` from the *Select attribute layout editor* combobox. This enables the *Form Layout* panel next to the *Available widgets* panel, filled with existing fields. The selected field displays its *properties* in a third panel.

2. Select fields you do not want to use in your *Form Layout* panel and hit the button to remove them. You can also toggle the selection with the Invert selection button.

3. Drag and drop fields from the first panel to the :guilabel`Form Layout` one to re-add them. The same field can be added multiple times.

4. Drag and drop fields within the *Form Layout* panel to reorder their position.

5. Add containers to associate fields that belong to the same category and better structure the form.

    1. The first step is to use the Add a new tab or group to the form layout icon. Fields and other groups will be displayed in it.

2. Then set the properties of the container, ie:
   - the *Label*: the title that will be used for the container
   - the *Container Type*: it can be a *Tab*, *Group box in container* (a collapsible group box inside a tab or another group) or a *Row* (a container type that allows you to arrange your widgets in a horizontal row, automatically determining the number of columns based on the number of widgets),
   - the *Within*: this optional feature allows you to select an existing container in which the new container (*Group box in container* or *Row*) will be embedded.
   - and the *Number of columns* the embedded fields should be distributed over

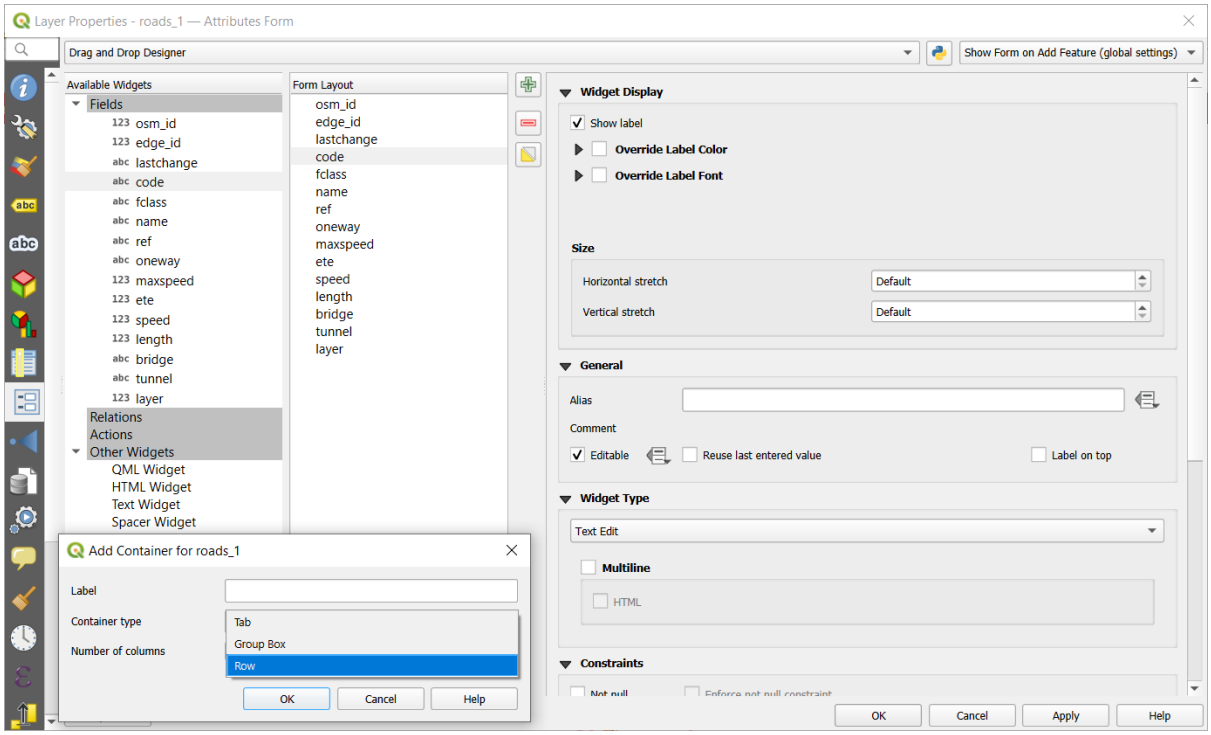


Fig. 16.43: Dialog to create containers with the **Attribute editor layout**

These, and other properties can later be updated by selecting the item and, from the third panel:

- hide or show the container's label
- rename the container
- set the number of columns
- enter an expression to control the container's visibility. The expression will be re-evaluated every time values in the form change, and the tab or group box shown/hidden accordingly
- *Show as Group Box*: converts a tab into a top-level group box and vice versa
- in case of a group box, set whether at the form opening, it should show as *Collapsed* for every features, or only for those matching an expression (*Control collapsed by expression*).
- configure the *Style* look of the container, with custom *Background color*, label color and font properties

3. You can create and embed as many containers as you want by pressing again the Add a new tab or group to the form layout icon.

6. The next step is to assign the relevant fields to each container, by simple drag and drop. Group boxes and tabs can also be moved the same way.

7. *Customize the widget* of the fields in use
8. In case the layer is involved in a *one or many to many relation*, drag-and-drop the relation name from the *Available Widgets* panel to the *Form Layout* panel. The associated layer attribute form will be embedded at the chosen place in the current layer's form. As for the other items, select the relation label to configure some properties:
   - hide or show the relation label
   - show the link button
   - show the unlink button
9. In case the layer has one or more *actions* enabled for `Layer` or `Feature` scope, the actions will be listed under *Actions* and you can drag and drop them as with the other fields. The associated action will be embedded at the chosen place in the current layer's form.
10. Further customize the form by adding one or more widgets from *Other Widgets* (see *Other Widgets*)
11. Apply the layer's properties dialog
12. Open a feature attribute form (eg, using the Identify features tool) and it should display the new form.

Other Widgets

The drag and drop designer offers a number of widgets that are not connected to a particular field. They can be used to enhance the appearance of the form or to display dynamically calculated values.

- *HTML Widget*: embeds an HTML page, the HTML source may contain the result of dynamically calculated expressions.

  HTML widgets can be used for example to display images stored as BLOB in a field (let's call it `photo`):

  1. In the `Drag-and-drop designer` mode, add a *HTML Widget* to your *Form Layout*.
  2. Double-click on the *HTML Widget* to configure it.
  3. Change the default *Title* or hide it.
  4. Press the button and enter the following QGIS expression:

     ```
     '<img src= "' || to_base64("photo") || '">'
     ```

     Ensure that you replace *photo* with your own BLOB field name. The above expression creates a string with HTML image tag in which the BLOB file is encoded.
  5. Apply the dialog and then press the button.
  6. QGIS automatically applies HTML formatting and functions to evaluate your expression, resulting in following code:

     ```
     <script>document.write(expression.evaluate("'<img src=\"data:image/png;
     ↪base64,' || to_base64(\"photo\") || '\">'"));</script>
     ```

     A preview of your image is displayed on the right.
- *QML Widget*: embeds a QML page, the QML source may contain the result of dynamically calculated expressions.
- *Text Widget*: displays a text widget which supports basic HTML markup and may contain the result of dynamically calculated expressions.
- *Spacer Widget*: inserts an empty transparent rectangle, increasing the vertical distance between two widgets.

Sfat: Display Dynamic Content

The widgets mentioned above (except the *Spacer Widget*) support expressions that can be used to display content that dynamically changes whenever another field in the form changes. This can be achieved using the `current_value('field_name')` function in the expression to examine the value of another field.

Folosirea fișierului UI personalizat

The `Provide ui-file` option allows you to use complex dialogs made with Qt-Designer. Using a UI-file allows a great deal of freedom in creating a dialog. Note that, in order to link the graphical objects (textbox, combobox…) to the layer's fields, you need to give them the same name.

Use the *Edit UI* to define the path to the file to use.

UI-files can also be hosted on a remote server. In this case, you provide the URL of the form instead of the file path in *Edit UI*.

You'll find some example in the Creating a new form lesson of the QGIS-training-manual-index-reference. For more advanced information, see https://woostuff.wordpress.com/2011/09/05/qgis-tips-custom-feature-forms-with-python-logic/.

Enhance your form with custom functions

QGIS forms can have a Python function that is called when the dialog is opened. Use this function to add extra logic to your dialogs. The form code can be specified in three different ways:

- `load from the environment`: use a function, for example in `startup.py` or from an installed plugin
- `load from an external file`: a file chooser will let you select a Python file from your filesystem or enter a URL for a remote file.
- `provide code in this dialog`: a Python editor will appear where you can directly type the function to use.

In all cases you must enter the name of the function that will be called (`open` in the example below).

Un exemplu este (în modulul MyForms.py):

```
def open(dialog,layer,feature):
    geom = feature.geometry()
    control = dialog.findChild(QWidget,"My line edit")
```

Reference in Python Init Function like so: `open`

Configure the field behavior

The main part of the *Attributes Form* tab helps you set the type of widget used to fill or display values of the field, in the attribute table or the feature form: you can define how user interacts with each field and the values or range of values that are allowed to be added to each.

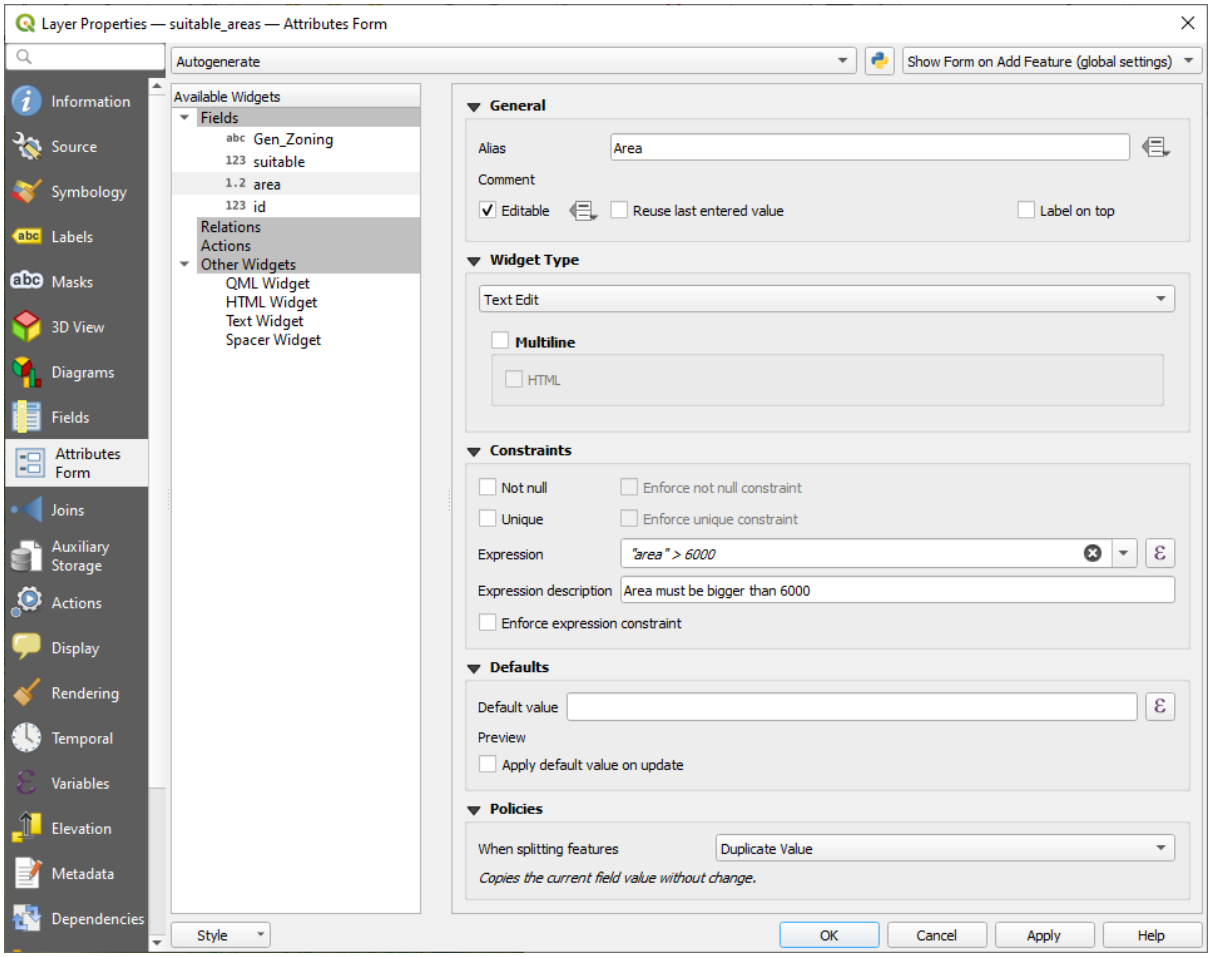


Fig. 16.44: Dialog to select an edit widget for an attribute column

Common settings

Regardless the type of widget applied to the field, there are some common properties you can set to control whether and how a field can be edited.

Widget display

Only available for the *Drag and drop* designer mode, this group helps you configure the look of the widget assigned to the field:

- *Show label*: indicates whether the field name should be displayed in the form
- *Override label color*: applies specific color to the field's label
- *Override label font*: applies specific font properties (bold, italic, underline, strikethrough, or font family) to the field's label

Opțiuni generale

- *Alias*: a human readable name to use for fields. The alias will be displayed in the feature form, the attribute table, or in the *Identify results* panel. It can also be used as field name replacement in the *expression builder*, easing expressions understanding and reviews. Aliases are saved in project file.
- *Comment*: displays the field's comment as shown in the *Fields* tab, in a read-only state. This information is shown as tooltip when hovering over the field label in a feature form.
- *Editable*: uncheck this option to set the field read-only (not manually modifiable) even when the layer is in edit mode. Note that checking this setting doesn't override any edit limitation from the provider. This option can be controlled by a data-defined property thanks to the Data-defined override button.
- *Reuse last entered value*: remembers the last value entered in this field and uses it as default for the next feature being edited in the layer.
- *Label on top*: places the field name above or beside the widget in the feature form.

Constraints

You can constrain the value to insert in the field. This constraint can be:

- *Not null*: requires the user to provide a value;
- *Unique*: guarantee the inserted value to be unique throughout the field;
- based on a custom *expression*: e.g. `not regexp_match(col0,'[^A-Za-z]')` will ensure that the value of the field *col0* has only alphabet letters. A short description can be added to help you remember the constraint.

Whenever a value is added or edited in a field, it's submitted to the existing constraints and:

- if it meets all the requirements, a green check is shown beside the field in the form;
- if it does not meet all the requirements, then the field is colored in yellow or orange and a corresponding cross is displayed next to the widget. You can hover over the cross to remind which constraints are applied to the field and fix the value:
  - A yellow cross appears when the unmet constraint is an unenforced one (`soft constraint`) and it does not prevent you to save the changes with the „wrong" values;
  - An orange cross can not be ignored and does not allow you to save your modifications until they meet the constraints. It appears when the *Enforce constraint* option is checked (`hard constraint`).

Default values

- *Default value*: for new features, automatically populates by default the field with a predefined value or an *expression-based one*. For example, you can:
  - use `$x`, `$length`, `$area` to automatically populate a field with the feature's X coordinate, length, area or any geometric information at its creation;
  - increment a field by 1 for each new feature using `maximum("field")+1`;
  - save the feature creation datetime using `now()`;
  - use *variables* in expressions, making it easier to e.g. insert the operator name (`@user_full_name`), the project file path (`@project_path`), …

  A preview of the resulting default value is displayed at the bottom of the widget.

Notă: The `Default value` option is not aware of the values in any other field of the feature being created so it won't be possible to use an expression combining any of those values i.e using an expression like `concat(field1, field2)` may not work.

- *Apply default value on update*: whenever the feature attribute or geometry is changed, the default value is recalculated. This could be handy to save values like last user that modifies data, last time it was changed…

Policies

Policies allows you to determine how values are assigned to the field when *Splitting features*:

- *Duplicate Values*: Keeps the existing value of the field for the resulting split features.
- *Use Default Value*: Resets the field by recalculating its *default value*. If no default value clause exists, the existing value is kept for the resulting split features.
- *Remove Value*: Clears the field to an unset state.
- *Use Ratio Geometries*: Recalculates the field value for all split portions by multiplying the existing value by ratio of the split parts lengths or areas.

Edit widgets

Based on the field type, QGIS automatically determines and assigns a default widget type to it. You can then replace the widget with any other compatible with the field type. The available widgets are:

- **Binary (BLOB)**: Available only for binary fields, it displays by default a label with the size of the embedded data, if not empty. A drop-down button next to the label allows to:
  - *Embed file*, replacing or filling the field
  - *Clear contents*, removing any data in the field
  - *Save contents to file*, exporting the data as a file on disk

  It is also possible to preview the embedded binary file in the field, if combined in a drag-and-drop form with e.g. a *QML or HTML widget*.
- **Checkbox**: Displays a checkbox whose state defines the value to insert.
- **Classification**: Only available when a *categorized symbology* is applied to the layer, displays a combo box with the values of the classes.
- **Color**: Displays a *color widget* allowing to select a color; the color value is stored as a html notation in the attribute table.
- **Date/Time**: Displays a line field which can open a calendar widget to enter a date, a time or both. Column type must be text. You can select a custom format, pop-up a calendar, etc.
- **Enumeration**: Opens a combo box with predefined values fetched from the database. This is currently only supported by the PostgreSQL provider, for fields of `enum` type.
- **Attachment**: Uses a „Open file" dialog to store file path in a relative or absolute mode. It can be used to display a hyperlink (to document path), a picture or a web page. User can also configure an *external storage system* to fetch/store resources.

Sfat: Relative Path in Attachment widget

If the path which is selected with the file browser is located in the same directory as the `.qgs` project file or below, paths are converted to relative paths. This increases portability of a `.qgs` project with multimedia information attached.

- **Hidden**: A hidden attribute column is invisible. The user is not able to see its contents.
- **Key/Value**: Displays a two-columns table to store sets of key/value pairs within a single field. This is currently supported by the PostgreSQL provider, for fields of `hstore` type.
- **JSON View**: Displays JSON data in a syntax highlighted text edit or in tree view. This widget is currently read only. Several options are available to change how the data is displayed. «Default view» specify if the widget should appear in Text or Tree mode. «Format JSON» has three options which are related to the tree view only:
  - Indented: Display data in a human readable form with newlines and four space characters for indentation.
  - Compact: Display data in a one-line size optimized string without newlines or spaces.
  - Disabled: Display data as it comes from the provider.
- **List**: Displays a single column table to add different values within a single field. This is currently supported by the PostgreSQL provider, for fields of `array` type.
- **Range**: Allows you to set numeric values from a specific range. The edit widget can be either a slider or a spin box.
- **Relation Reference**: This is the default widget assigned to the referencing field (i.e., the foreign key in the child layer) when a *relation* is set. It provides direct access to the parent feature's form which in turn embeds the list and form of its children.
- **Text Edit** (default): This opens a text edit field that allows simple text or multiple lines to be used. If you choose multiple lines you can also choose html content.
- **Unique Values**: You can select one of the values already used in the attribute table. If «Editable» is activated, a line edit is shown with autocompletion support, otherwise a combo box is used.
- **Uuid Generator**: Generates a read-only UUID (Universally Unique Identifiers) field, if empty.
- **Value Map**: A combo box with predefined items. The value is stored in the attribute, the description is shown in the combo box. You can define values manually or load them from a layer or a CSV file.
- **Value Relation**: Offers values from a related table in a combobox. You can select layer, key column and value column. Several options are available to change the standard behaviors: allow null value, order by value, allow multiple selections and use of auto-completer. The forms will display either a drop-down list or a line edit field when completer checkbox is enabled.

  If a layer that is stored in PostgreSQL, GeoPackage or SpatiaLite is configured to use a value relation widget, but the required layer is not already loaded into the project, QGIS will automatically search for the layer in the same database/connection.

16.1.10 Joins Properties

The *Joins* tab allows you to associate features of the current layer to features from another loaded vector layer (or table). The join is based on an attribute that is shared by the layers, in a one-to-one relationship. For more details on joins, please read *Unirea entităților din două straturi*.

16.1.11 Auxiliary Storage Properties

The regular way to customize styling and labeling is to use data-defined properties as described in *Configurarea suprascrierii definită de date*. However, it may not be possible if the underlying data is read only. Moreover, configuring these data-defined properties may be very time consuming or not desirable! For example, if you want to fully use map tools coming with *The Label Toolbar*, then you need to add and configure more than 20 fields in your original data source (X and Y positions, rotation angle, font style, color and so on).

The Auxiliary Storage mechanism provides the solution to these limitations and awkward configurations. Auxiliary fields are a roundabout way to automatically manage and store these data-defined properties (labels, diagram, symbology…) in a SQLite database thanks to editable joins. This allows you to store properties for layers that aren't editable.

A tab is available in vector layer properties dialog to manage auxiliary storage:

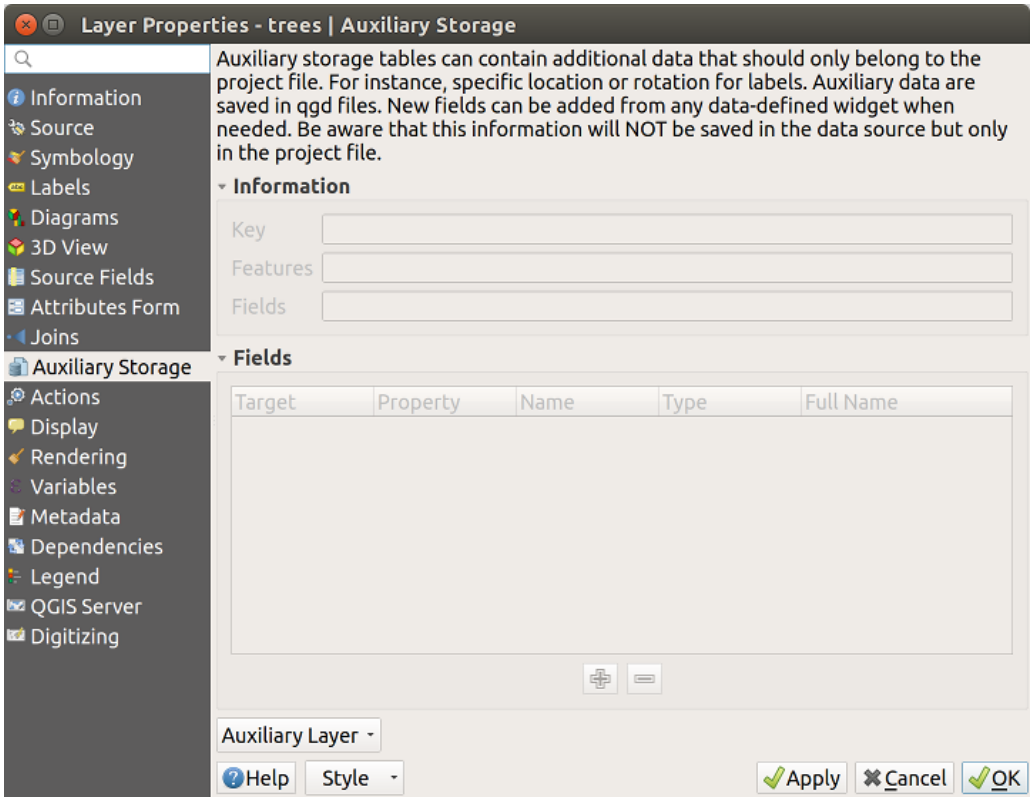


Fig. 16.45: Auxiliary Storage tab

Etichetare

Considering that the data source may be customized thanks to data-defined properties without being editable, labeling map tools described in *The Label Toolbar* are always available as soon as labeling is activated.

Actually, the auxiliary storage system needs an auxiliary layer to store these properties in a SQLite database (see *Auxiliary storage database*). Its creation process is run the first time you click on the map while a labeling map tool is currently activated. Then, a window is displayed, allowing you to select the primary key to use for joining (to ensure that features are uniquely identified):

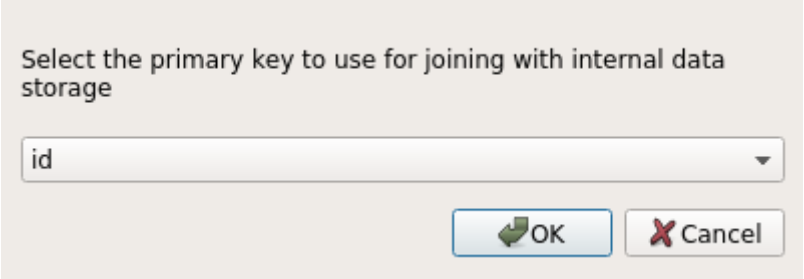


Fig. 16.46: Auxiliary Layer creation dialog

As soon as an auxiliary layer is configured for the current data source, you can retrieve its information in the tab:

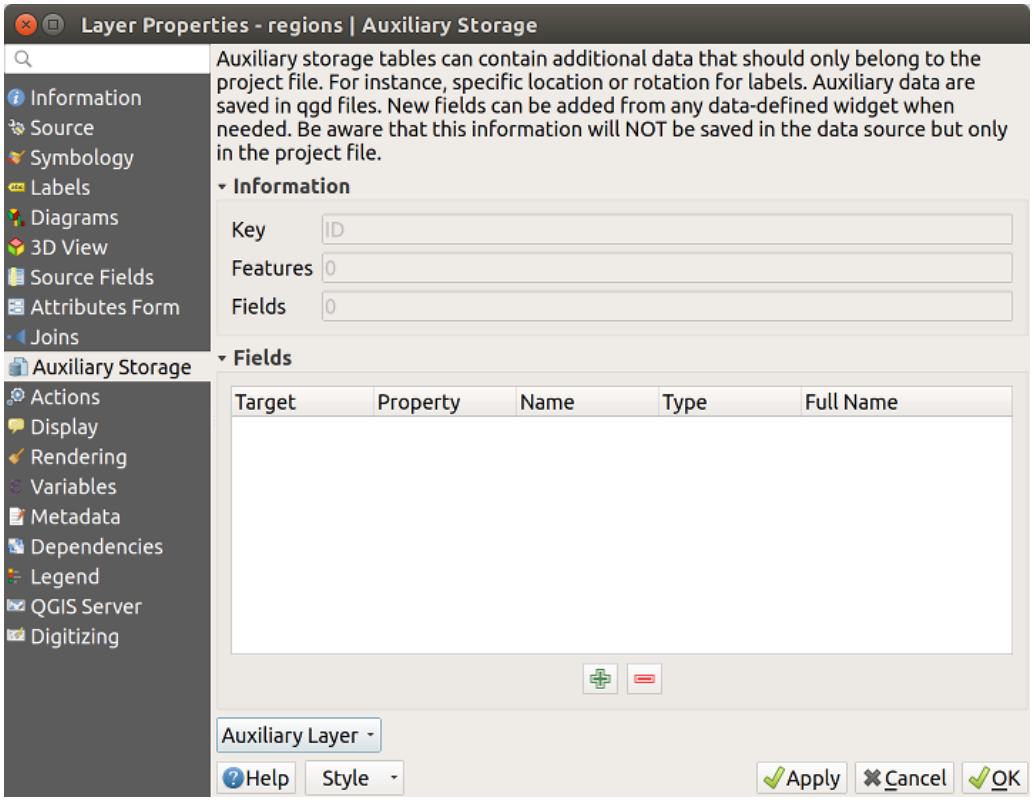


Fig. 16.47: Cheia Stratului Auxiliar

The auxiliary layer now has these characteristics:

- the primary key is `ID`,
- there are `0` features using an auxiliary field,
- there are `0` auxiliary fields.

Now that the auxiliary layer is created, you can edit the layer labels. Click on a label while the Change Label map tool is activated, then you can update styling properties like sizes, colors, and so on. The corresponding data-defined properties are created and can be retrieved:

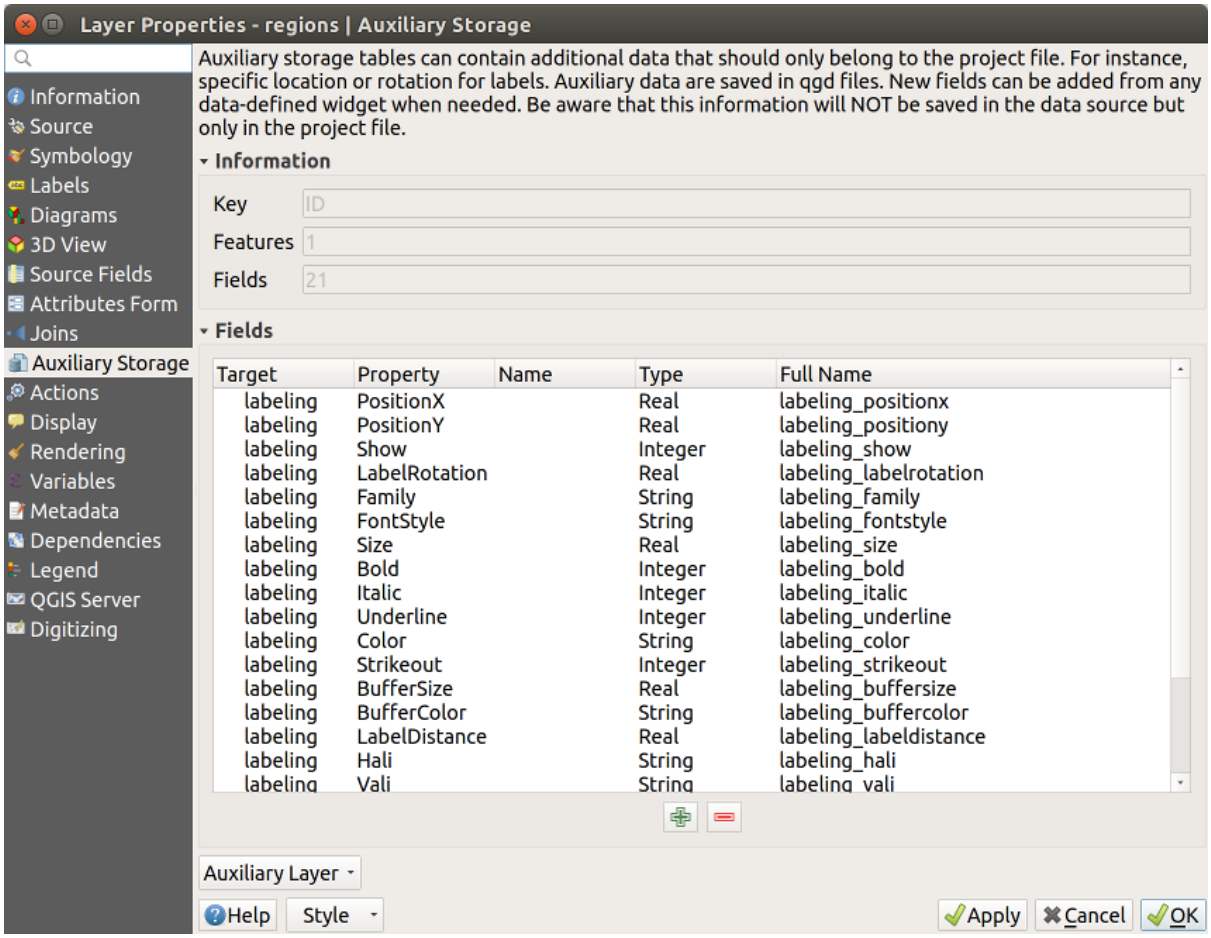


Fig. 16.48: Auxiliary Fields

As you can see in the figure above, `21` fields are automatically created and configured for labeling. For example, the `FontStyle` auxiliary field type is a `String` and is named `labeling_fontstyle` in the underlying SQLite database. There is also `1` feature which is currently using these auxiliary fields.

Notice that the icon is displayed in the *Labels* properties tab indicating that the data-defined override options are set correctly:

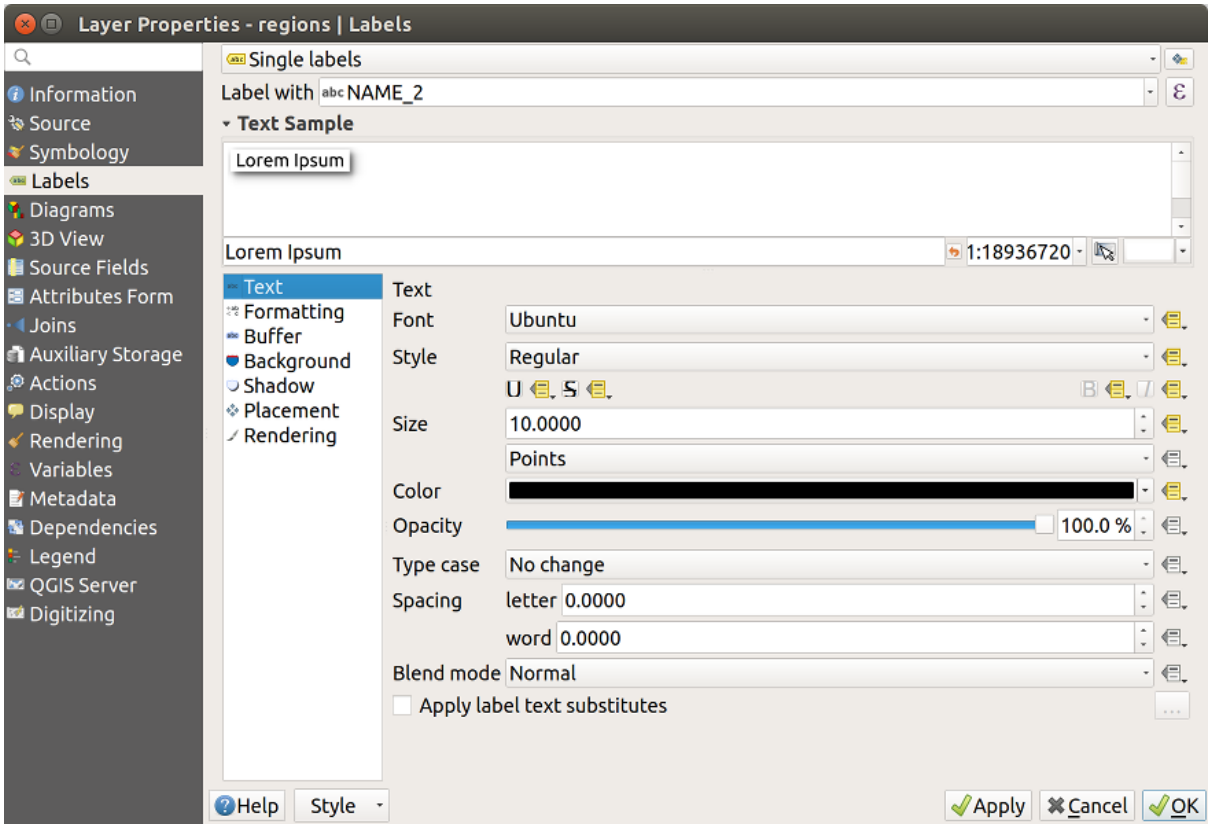


Fig. 16.49: Data-defined properties automatically created

Otherwise, there's another way to create an auxiliary field for a specific property thanks to the Data-defined override button. By clicking on *Store data in the project*, an auxiliary field is automatically created for the *Opacity* field. If you click on this button and the auxiliary layer is not created yet, a window (Fig. 16.46) is first displayed to select the primary key to use for joining.

Simbologie

Like the method described above for customizing labels, auxiliary fields can also be used to stylize symbols and diagrams. To do this, click on Data-defined override and select *Store data in the project* for a specific property. For example, the *Fill color* field:

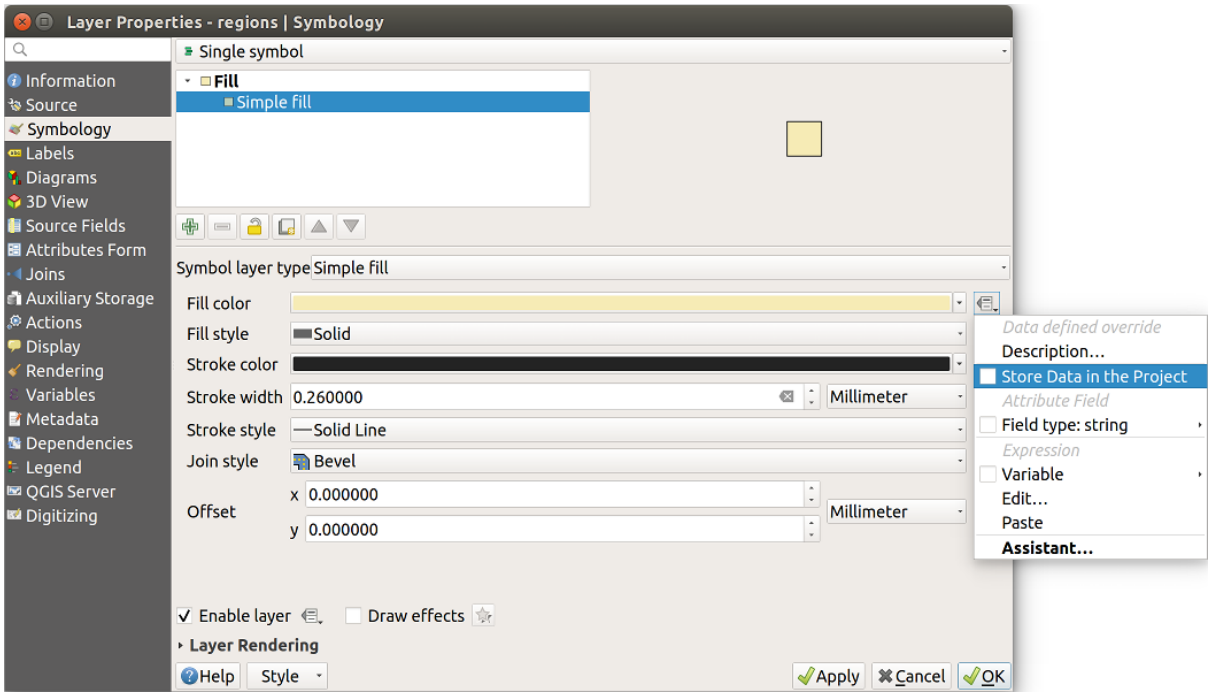


Fig. 16.50: Data-defined property menu for symbol

There are different attributes for each symbol (e.g. fill style, fill color, stroke color, etc…), so each auxiliary field representing an attribute requires a unique name to avoid conflicts. After selecting *Store data in the project*, a window opens and displays the *Type* of the field and prompts you to enter a unique name for the auxiliary field. For example, when creating a *Fill color* auxiliary field the following window opens:

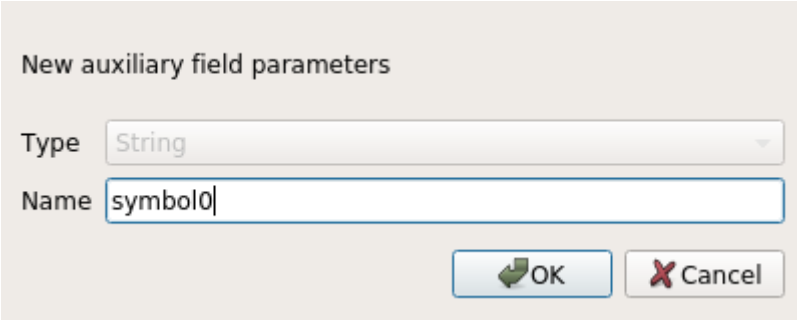


Fig. 16.51: Numele câmpului auxiliar pentru un simbol

Once created, the auxiliary field can be retrieved in the auxiliary storage tab:

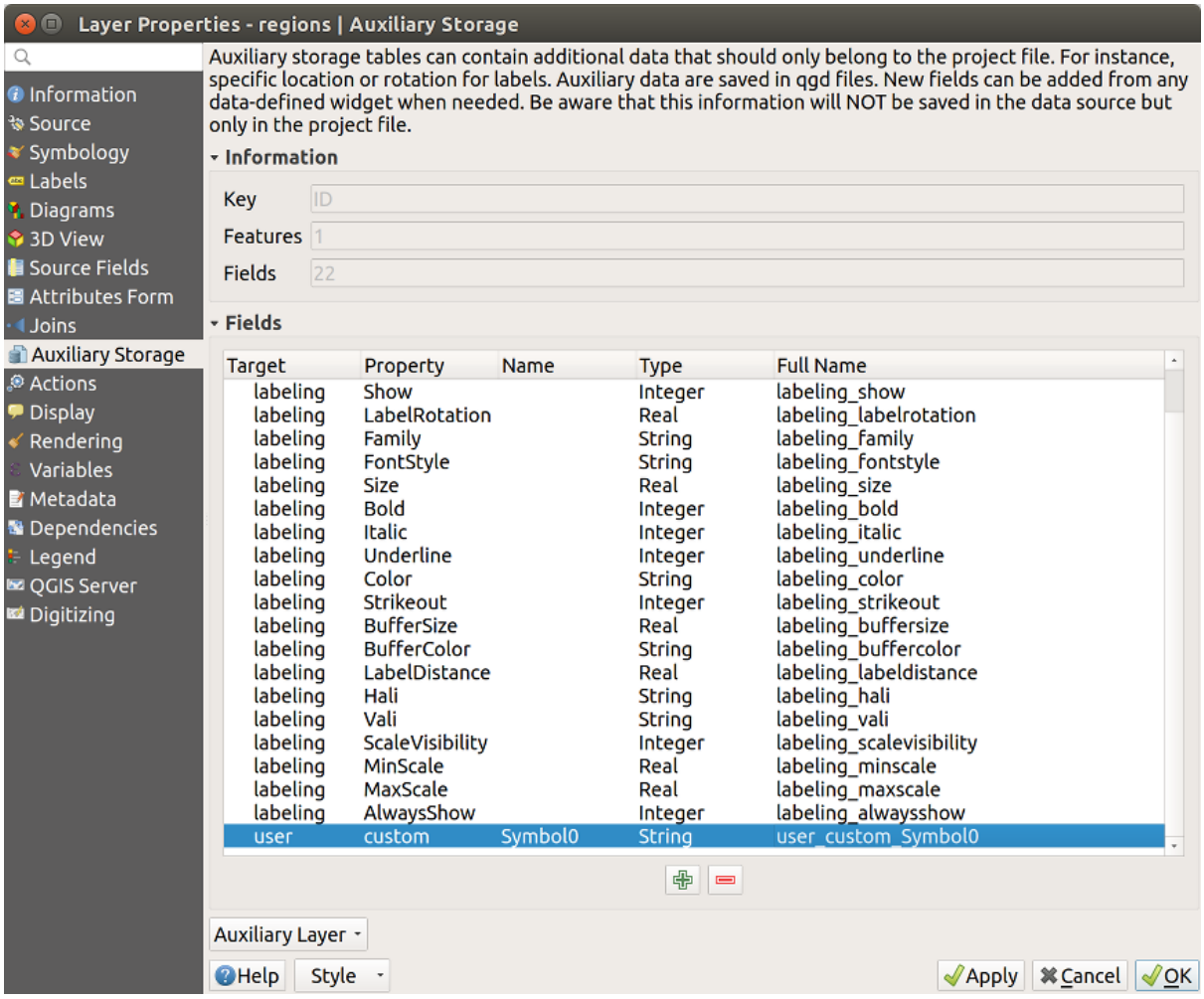


Fig. 16.52: Auxiliary field symbol

Tabela de atribute și controale grafice

Auxiliary fields can be edited using the *attribute table*. However, not all auxiliary fields are initially visible in the attribute table.

Auxiliary fields representing attributes of a layer's symbology, labeling, appearance, or diagrams will appear automatically in the attribute table. The exception are attributes that can be modified using the *Label Toolbar* which are hidden by default. Auxiliary fields representing a `Color` have a widget **Color** set by default, otherwise auxiliary fields default to the **Text Edit** widget.

Auxiliary fields that represent attributes that can be modified using the *Label toolbar* are **Hidden** in the attribute table by default. To make a field visible, open the *Attribute Form properties tab* and change the value of an auxiliary field *Widget Type* from **Hidden** to another relevant value. For example, change the **auxiliary_storage_labeling_size** to **Text Edit** or change **auxiliary_storage_labeling_color** to the **Color** widget. Those fields will now be visible in the attribute table.

Auxiliary fields in the attribute table will appear like the following image:

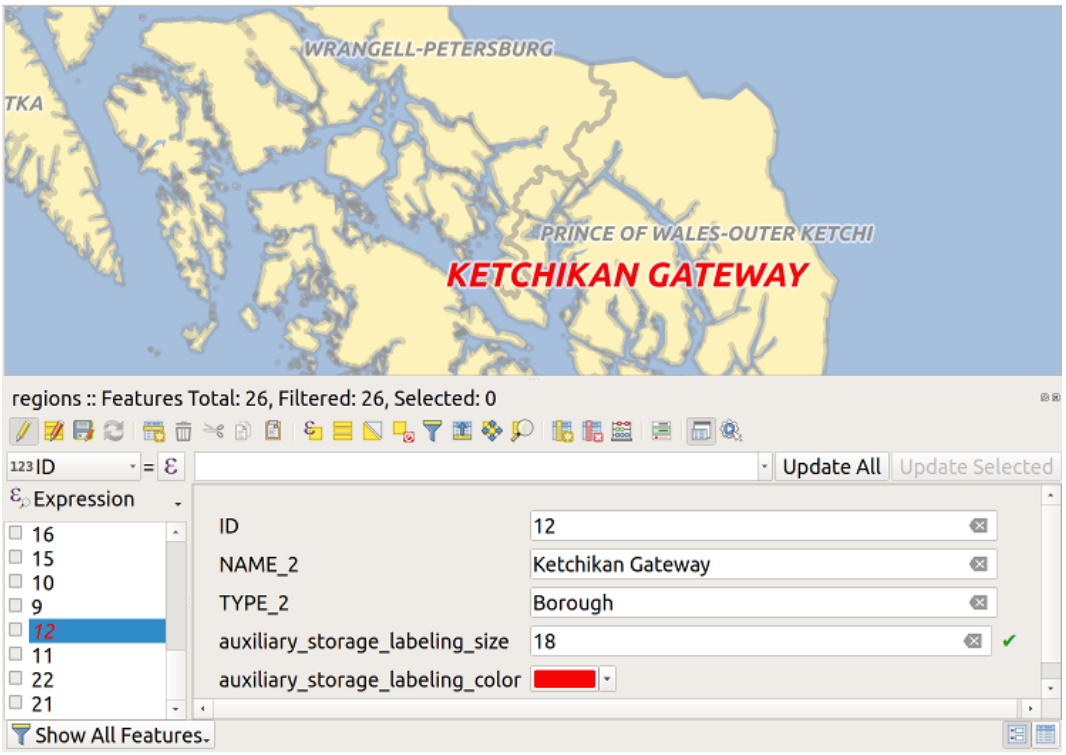


Fig. 16.53: Form with auxiliary fields

Management

The *Auxiliary Layer* menu allows you to manage the auxiliary fields:

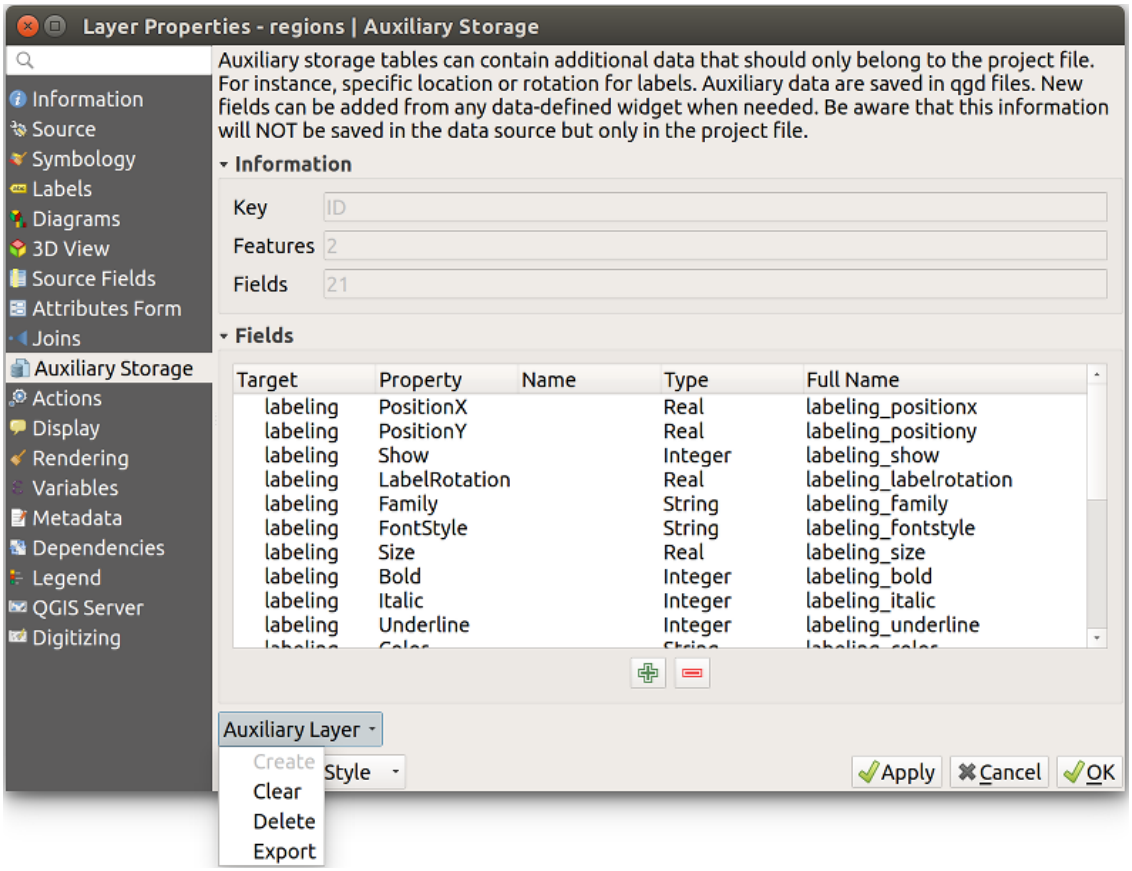


Fig. 16.54: Auxiliary layer management

The first item *Create* is disabled in this case because the auxiliary layer is already created. But in case of a fresh work, you can use this action to create an auxiliary layer. As explained in *Etichetare*, a primary key will be needed then.

The *Clear* action allows to keep all auxiliary fields, but remove their contents. This way, the number of features using these fields will fall to `0`.

The *Delete* action completely removes the auxiliary layer. In other words, the corresponding table is deleted from the underlying SQLite database and properties customization are lost.

Finally, the *Export* action allows to save the auxiliary layer as a *new vector layer*. Note that geometries are not stored in auxiliary storage. However, in this case, geometries are exported from the original data source too.

Auxiliary storage database

When you save your project with the `.qgs` format, the SQLite database used for auxiliary storage is saved at the same place but with the extension `.qgd`.

For convenience, an archive may be used instead thanks to the `.qgz` format. In this case, `.qgd` and `.qgs` files are both embedded in the archive.

16.1.12 Proprietățile Acțiunilor

The *Actions* tab provides the ability to perform an action based on the attributes of a feature. This can be used to perform any number of actions, for example, running a program with arguments built from the attributes of a feature or passing parameters to a web reporting tool.

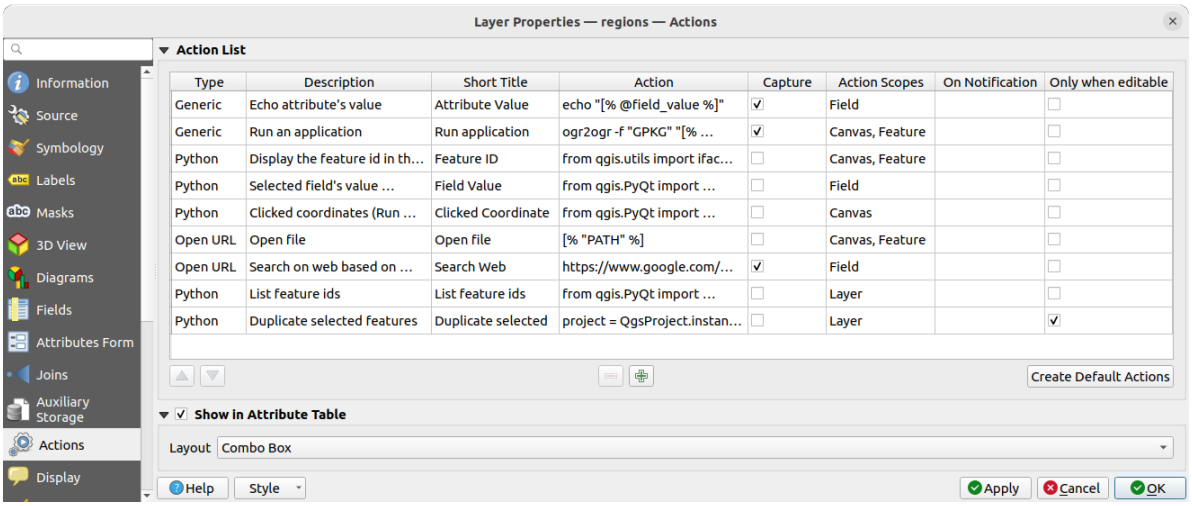


Fig. 16.55: Dialogul de ansamblu al acțiunilor, cu unele acțiuni de probă

Actions are useful when you frequently want to run an external application or view a web page based on one or more values in your vector layer. They are of different types and can be used like this:

- *Generic*, *macOS*, *Windows* and *Unix* actions start an external process.
- *Python* actions execute a Python expression.
- *Generic* and *Python* actions are visible everywhere.
- *macOS*, *Windows* and *Unix* actions are visible only on the respective platform (i.e., you can define three „Edit” actions to open an editor and the users can only see and execute the one „Edit” action for their platform to run the editor).
- *Open URL*: Uses a HTTP GET request to open a provided URL.
- *Submit URL (urlencoded or JSON)*: Same as the *Open URL* action but using a HTTP POST request. Data are posted to a URL, using „application/x-www-form-urlencoded” or „application/json” if the body is a valid JSON.

  An example of action call could be:

  ```
  http://localhost:8000?/[% url_encode(map('file', 'index.php')) %]
  ```

- *Submit URL (multipart)*: Same as the *Open URL* action but using a HTTP POST request. Data are posted to a URL, using „multipart/form-data”.

There are several examples included in the dialog. You can load them by clicking on *Create Default Actions*. To edit any of the examples, double-click its row. One example is performing a search based on an attribute value. This concept is used in the following discussion.

The *Show in Attribute Table* allows you to display in the attribute table dialog the checked feature-scoped actions, either as *Combo Box* or as *Separate Buttons* (see *Configuring the columns*).

Definirea Acțiunilor

To define an attribute action, open the vector *Layer Properties* dialog and click on the *Actions* tab. In the *Actions* tab, click the Add a new action to open the *Edit Action* dialog.

Select the action *Type* and provide a descriptive name for the action. The action itself must contain the name of the application that will be executed when the action is invoked. You can add one or more attribute field values as arguments to the application. When the action is invoked, any set of characters that start with a `%` followed by the name of a field will be replaced by the value of that field. The special characters `%%` will be replaced by the value of the field that was selected from the identify results or attribute table (see *Folosirea Acțiunilor*). Double quote marks can be used to group text into a single argument to the program, script or command. Double quotes will be ignored if preceded by a backslash.

The *Action Scopes* allows you to define where the action should be available. You have following choices:

1. *Field*: action is available when right click in the cell within the attribute table, in the feature form and in the default action button of the main toolbar.
2. *Feature*: action is available when right click in the cell within the attribute table.
3. *Canvas*: action is available in the main action button in the toolbar.
4. *Form*: action is available only in a feature form designed using the *drag-and-drop* mode.
5. *Layer*: action is available in the action button in the attribute table toolbar. Be aware that this type of action involves the entire layer and not the single features.

If you have field names that are substrings of other field names (e.g., `col1` and `col10`), you should indicate that by surrounding the field name (and the % character) with square brackets (e.g., `[%col10]`). This will prevent the `%col10` field name from being mistaken for the `%col1` field name with a `0` on the end. The brackets will be removed by QGIS when it substitutes in the value of the field. If you want the substituted field to be surrounded by square brackets, use a second set like this: `[[%col10]]`.

Using the *Identify Features* tool, you can open the *Identify Results* dialog. It includes a *(Derived)* item that contains information relevant to the layer type. The values in this item can be accessed in a similar way to the other fields by proceeding the derived field name with `(Derived).`. For example, a point layer has an `X` and `Y` field, and the values of these fields can be used in the action with `%(Derived).X` and `%(Derived).Y`. The derived attributes are only available from the *Identify Results* dialog box, not the *Attribute Table* dialog box.

Two example actions are shown below:

- `konqueror https://www.google.com/search?q=%nam`
- `konqueror https://www.google.com/search?q=%%`

In the first example, the web browser konqueror is invoked and passed a URL to open. The URL performs a Google search on the value of the `nam` field from our vector layer. Note that the application or script called by the action must be in the path, or you must provide the full path. To be certain, we could rewrite the first example as: `/opt/kde3/bin/konqueror https://www.google.com/search?q=%nam`. This will ensure that the konqueror application will be executed when the action is invoked.

The second example uses the %% notation, which does not rely on a particular field for its value. When the action is invoked, the %% will be replaced by the value of the selected field in the identify results or attribute table.

Folosirea Acțiunilor

QGIS offers many ways to execute actions you enabled on a layer. Depending on their settings, they can be available:

- in the drop-down menu of Run Feature Action button from the *Attributes toolbar* or *Attribute table* dialog;
- when right-clicking a feature with the Identify Features tool (see *Identifying Features* for more information);
- from the *Identify Results* panel, under the *Actions* section;
- as items of an *Actions* column in the *Attribute Table* dialog.

Dacă invocați o acțiune care utilizează notația `%%`, faceți clic dreapta pe valoarea câmpului din dialogul de *Identificare a Rezultatelor* sau pe dialogul *Tabelei de atribute*, pe care doriți să le transmiteți aplicației sau script-ului.

Here is another example that pulls data out of a vector layer and inserts it into a file using bash and the `echo` command (so it will only work on or perhaps). The layer in question has fields for a species name `taxon_name`, latitude `lat` and longitude `long`. We would like to be able to make a spatial selection of localities and export these field values to a text file for the selected record (shown in yellow in the QGIS map area). Here is the action to achieve this:

```
bash -c "echo \"%taxon_name %lat %long\" >> /tmp/species_localities.txt"
```

După selectarea câtorva localități și desfășurarea acțiunii pentru fiecare dintre ele, deschiderea fișierului de ieșire va prezenta ceva de genul:

```
Acacia mearnsii -34.0800000000 150.0800000000
Acacia mearnsii -34.9000000000 150.1200000000
Acacia mearnsii -35.2200000000 149.9300000000
Acacia mearnsii -32.2700000000 150.4100000000
```

As an exercise, we can create an action that does a Google search on the `lakes` layer. First, we need to determine the URL required to perform a search on a keyword. This is easily done by just going to Google and doing a simple search, then grabbing the URL from the address bar in your browser. From this little effort, we see that the format is https://www.google.com/search?q=QGIS, where `QGIS` is the search term. Armed with this information, we can proceed:

1. Asigurați-vă că stratul `lakes` este încărcat.
2. Open the *Layer Properties* dialog by double-clicking on the layer in the legend, or right-click and choose *Properties* from the pop-up menu.
3. Click on the *Actions* tab.
4. Click Add a new action.
5. Choose the *Open URL* action type,
6. Introduceți un nume pentru acțiune, cum ar fi `Google Search`.
7. Additionally you can add a *Short Name* or even an *Icon*.
8. Choose the *Action Scopes*. See *Definirea Acțiunilor* for further information. Leave the default settings for this example.
9. For the action, add the URL used for doing a Google search, up to but not including the search term: `https://www.google.com/search?q=`
10. The text in the *Action* field should now look like this:

    ```
    https://www.google.com/search?q=
    ```

11. Click on the drop-down box containing the field names for the `lakes` layer. It's located just to the left of the *Insert* button.
12. From the drop-down box, select *NAMES* and click *Insert*.

13. Your action text now looks like this:

```
https://www.google.com/search?q=[%NAMES%]
```

14. Pentru a finaliza acțiunea, faceți clic pe butonul *OK*.

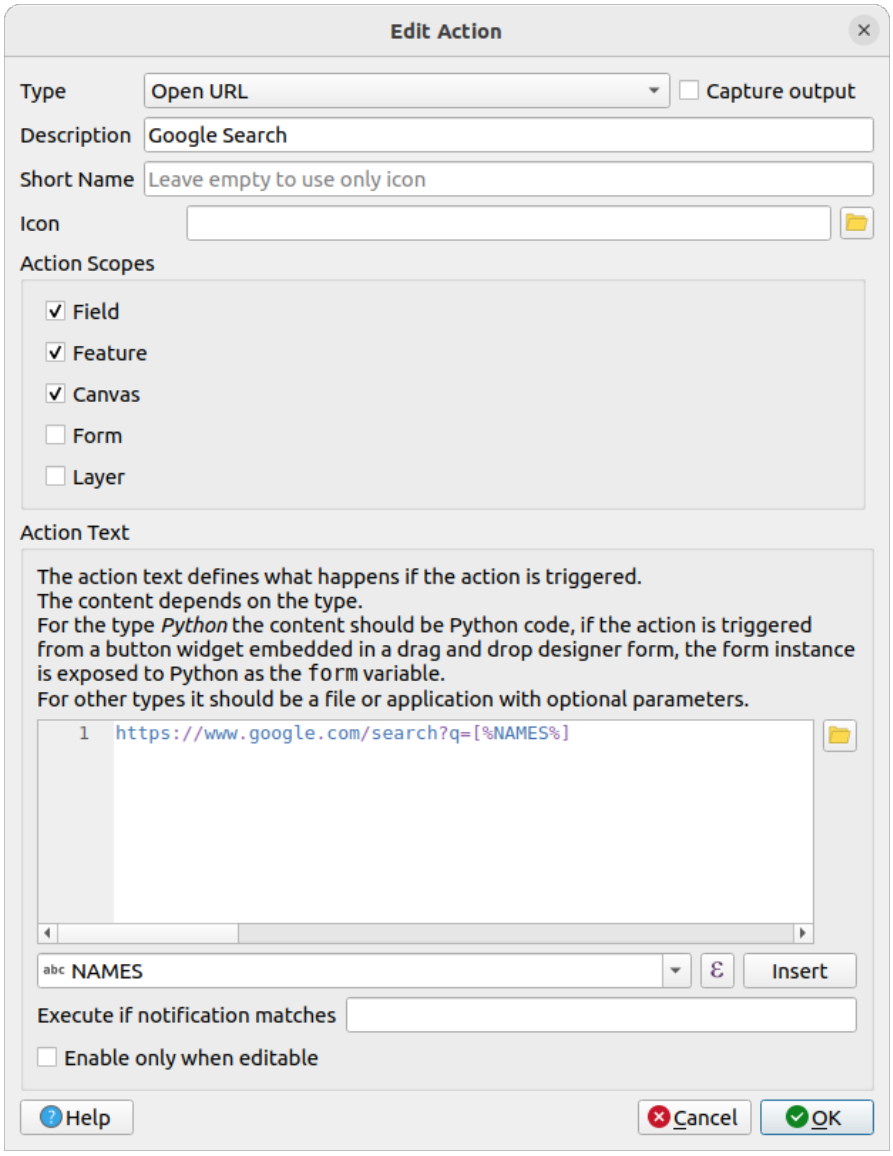


Fig. 16.56: Edit action dialog configured with the example

This completes the action, and it is ready to use.

Close the *Layer Properties* dialog and zoom in to an area of interest. Make sure the `lakes` layer is active and identify a lake. In the result box you'll now see that our action is visible:

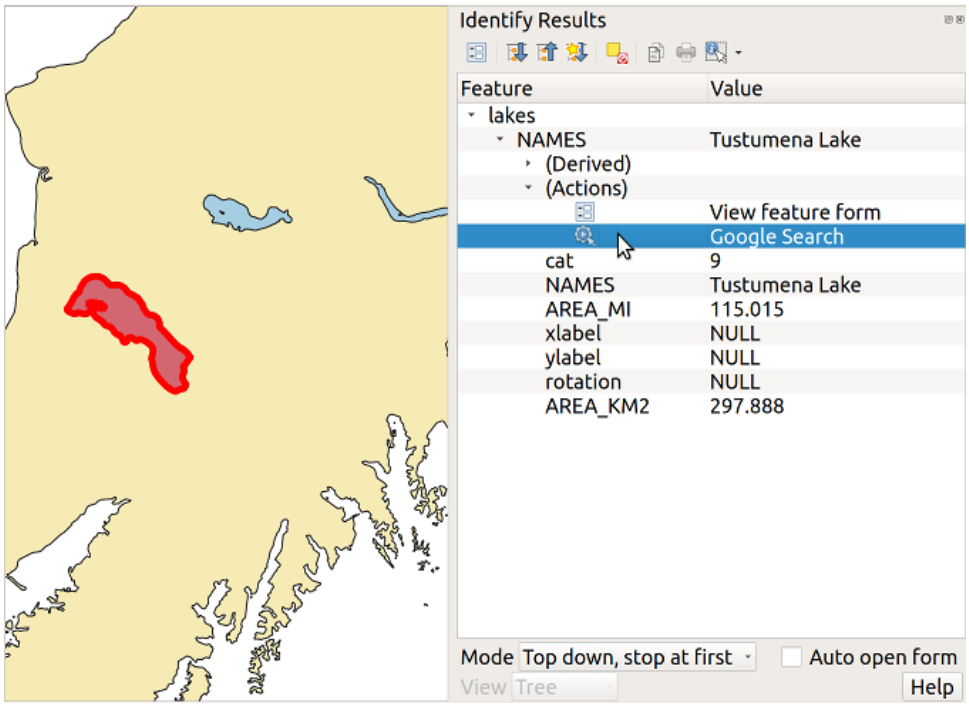


Fig. 16.57: Selectarea entitatății și alegerea acțiunii

When we click on the action, it brings up our default browser and navigates to the URL https://www.google.com/search?q=Tustumena. It is also possible to add further attribute fields to the action. Therefore, you can add a + to the end of the action text, select another field and click on *Insert Field*. In this example, there is just no other field available that would make sense to search for.

You can define multiple actions for a layer, and each will show up in the *Identify Results* dialog.

You can also invoke actions from the attribute table by selecting a row and right-clicking, then choosing the action from the pop-up menu.

There are all kinds of uses for actions. For example, if you have a point layer containing locations of images or photos along with a file name, you could create an action to launch a viewer to display the image. You could also use actions to launch web-based reports for an attribute field or combination of fields, specifying them in the same way we did in our Google search example.

We can also make more complex examples, for instance, using **Python** actions.

Usually, when we create an action to open a file with an external application, we can use absolute paths, or eventually relative paths. In the second case, the path is relative to the location of the external program executable file. But what about if we need to use relative paths, relative to the selected layer (a file-based one, like Shapefile or SpatiaLite)? The following code will do the trick:

```
command = "firefox"
imagerelpath = "images_test/test_image.jpg"
layer = qgis.utils.iface.activeLayer()
import os.path
layerpath = layer.source() if layer.providerType() == 'ogr'
  else (qgis.core.QgsDataSourceURI(layer.source()).database()
    if layer.providerType() == 'spatialite' else None)
path = os.path.dirname(str(layerpath))
image = os.path.join(path,imagerelpath)
import subprocess
subprocess.Popen( [command, image ] )
```

We just have to remember that the action is one of type *Python* and the *command* and *imagerelpath* variables must be changed to fit our needs.

But what about if the relative path needs to be relative to the (saved) project file? The code of the Python action would be:

```
command = "firefox"
imagerelpath = "images_test/test_image.jpg"
projectpath = qgis.core.QgsProject.instance().fileName()
import os.path
path = os.path.dirname(str(projectpath)) if projectpath != '' else None
image = os.path.join(path, imagerelpath)
import subprocess
subprocess.Popen( [command, image ] )
```

Another Python action example is the one that allows us to add new layers to the project. For instance, the following examples will add to the project respectively a vector and a raster. The names of the files to be added to the project and the names to be given to the layers are data driven (*filename* and *layername* are column names of the table of attributes of the vector where the action was created):

```
qgis.utils.iface.addVectorLayer('/yourpath/[% "filename" %].shp',
  '[% "layername" %]', 'ogr')
```

Pentru a adăuga un raster (o imagine TIF în acest exemplu), devine:

```
qgis.utils.iface.addRasterLayer('/yourpath/[% "filename" %].tif',
  '[% "layername" %]')
```

16.1.13 Display Properties

The *Display* tab helps you configure fields to use for feature identification:

- The *Display name*: based on a field or an *expression*. By default, this is set to the first field in the layer if no field with `<name>` component exists. This is used as:
  - the label shown on top of the feature information in the *Identify tool* results
  - the field used in the *locator bar* when looking for features in all layers
  - the feature identifier in the attribute table *form view*
  - the feature identifier when the map or layout is exported to a layered output format such as GeoPDF
  - the map tip information, i.e. the message displayed in the map canvas when hovering over a feature of the active layer with the Show Map Tips icon pressed. Applicable when *Enable Map Tips* is active and no *HTML Map Tip* is set.
- *Enable Map Tips* controls whether to display map tips for the layer
- The *HTML Map Tip* provides a complex and full HTML text editor for map tips, mixing QGIS expressions and html styles and tags (multiline, fonts, images, hyperlink, tables, …). You can check the result of your code sample in the *Preview* frame (also convenient for previewing the *Display name* output).

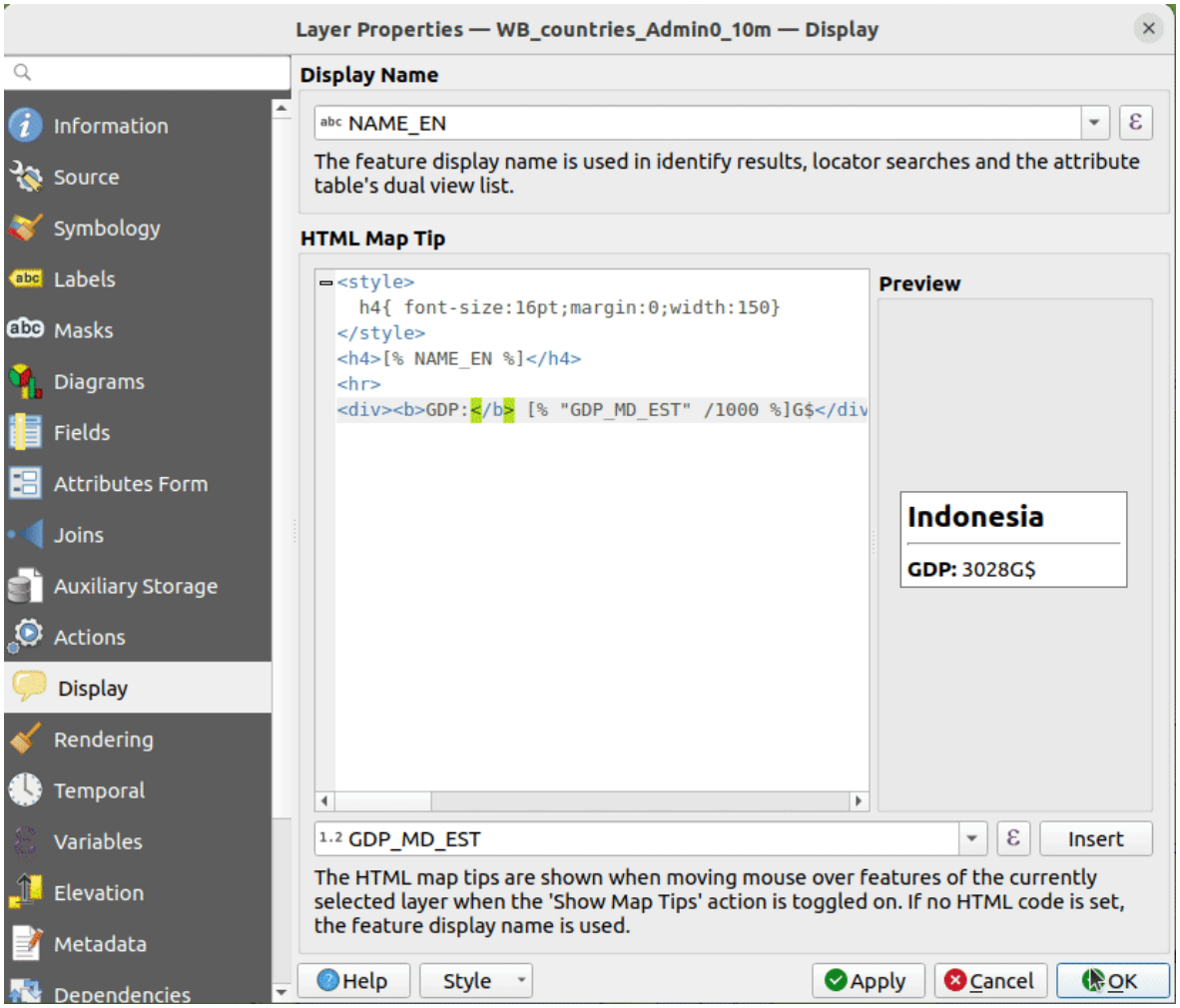


Fig. 16.58: HTML code for map tip

To display map tips:

1. Select the menu option *View* ► *Show Map Tips* or click on the Show Map Tips icon of the *Attributes Toolbar*.
2. Make sure that the layer you target is active and has the *Enable Map Tips* property checked.
3. Move over a feature, and the corresponding information will be displayed over.

Map tip is a cross-layer feature meaning that once activated, it stays on and applies to any map tip enabled layer in the project until it's toggled off.

Fig. 16.59: Map tip made with HTML code

16.1.14 Rendering Properties

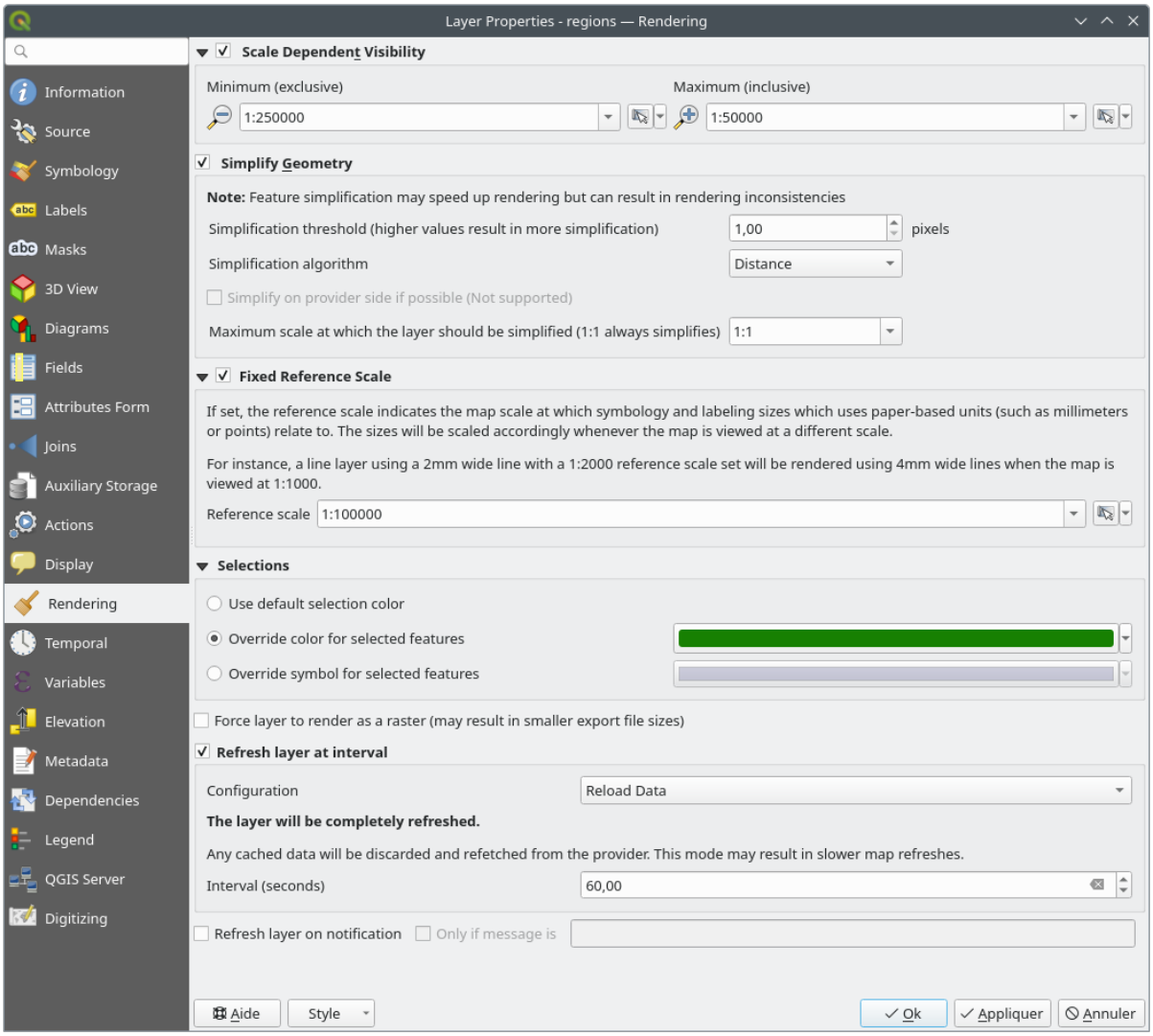


Fig. 16.60: Layer Rendering Properties dialog

The *Rendering* tab offers following properties:

- Under *Scale dependent visibility*, you can set the *Maximum (inclusive)* and *Minimum (exclusive)* scales, defining a range of scales in which features will be visible. Out of this range, they are hidden. The Set to current canvas scale button helps you use the current map canvas scale as boundary of the range visibility. See *Visibility Scale Selector* for more information.

  Notă: You can also activate scale dependent visibility on a layer from within the *Layers* panel: right-click on the layer and in the contextual menu, select *Set Layer Scale Visibility*.

- QGIS offers support for on-the-fly feature generalisation. This can improve rendering times when drawing many complex features at small scales. This feature can be enabled or disabled in the layer settings using the *Simplify geometry* option. There is also a global setting that enables generalisation by default for newly added layers (see *global simplification* for more information).

Notă: Feature generalisation may introduce artefacts into your rendered output in some cases. These may include slivers between polygons and inaccurate rendering when using offset-based symbol layers.

- The *Fixed reference scale* indicates the map scale at which symbology and labeling sizes which uses paper-based units (such as millimeters or points) relate to. The sizes will be scaled accordingly whenever the map is viewed at a different scale.

  For instance, a line layer using a 2mm wide line with a 1:2000 *Reference scale* set will be rendered using 4mm wide lines when the map is viewed at 1:1000.

- The *Selections* group allows you to control whether a specific color or symbol should be used in place of the defaults (*Project properties* ► *General* ► *Selection color*) for a particular layer. This is useful to improve the visibility of selected features with certain symbology:
  - *Use default selection color*
  - *Override color for selected features*: e.g. if the layer uses a yellow color by default and the standard yellow selection is not visible.
  - *Override symbol for selected features*: e.g. if a line layer uses a thin symbol, and coloring the lines might not make them visible enough, overriding the symbol with a thicker line can help. Also, if the layer uses raster symbology or gradient fills/lines/shapeburst with color ramp symbology, then the default selection color is not applied at all; being able to set a specific simpler symbol to use for selected features in the layer can help.

- Rendering extremely detailed layers (e.g. polygon layers with a huge number of nodes), can cause layout exports in PDF/SVG format to be huge as all nodes are included in the exported file. This can also make the resultant file very slow to work with/open in other programs.

  Checking *Force layer to render as raster* forces these layers to be rasterised so that the exported files won't have to include all the nodes contained in these layers and the rendering is therefore sped up.

  You can also do this by forcing the layout to export as a raster, but that is an all-or-nothing solution, given that the rasterisation is applied to all layers. Alternatively, you can rely on geometry simplification in *layout export settings*.

- *Refresh layer at interval*: controls whether and how regular a layer can be refreshed. Available *Configuration* options are:
  - *Reload data*: the layer will be completely refreshed. Any cached data will be discarded and refetched from the provider. This mode may result in slower map refreshes.
  - *Redraw layer only*: this mode is useful for animation or when the layer's style will be updated at regular intervals. Canvas updates are deferred in order to avoid refreshing multiple times if more than one layer has an auto update interval set.

  It is also possible to set the *Interval (seconds)* between consecutive refreshments.

- Depending on the data provider (e.g. PostgreSQL), notifications can be sent to QGIS when changes are applied to the data source, out of QGIS. Use the *Refresh layer on notification* option to trigger an update. You can also limit the layer refresh to a specific message set in the *Only if message is* text box.

16.1.15 Proprietăți Temporale

The *Temporal* tab provides options to control the rendering of the layer over time. Such dynamic rendering requires the *temporal navigation* to be enabled over the map canvas.

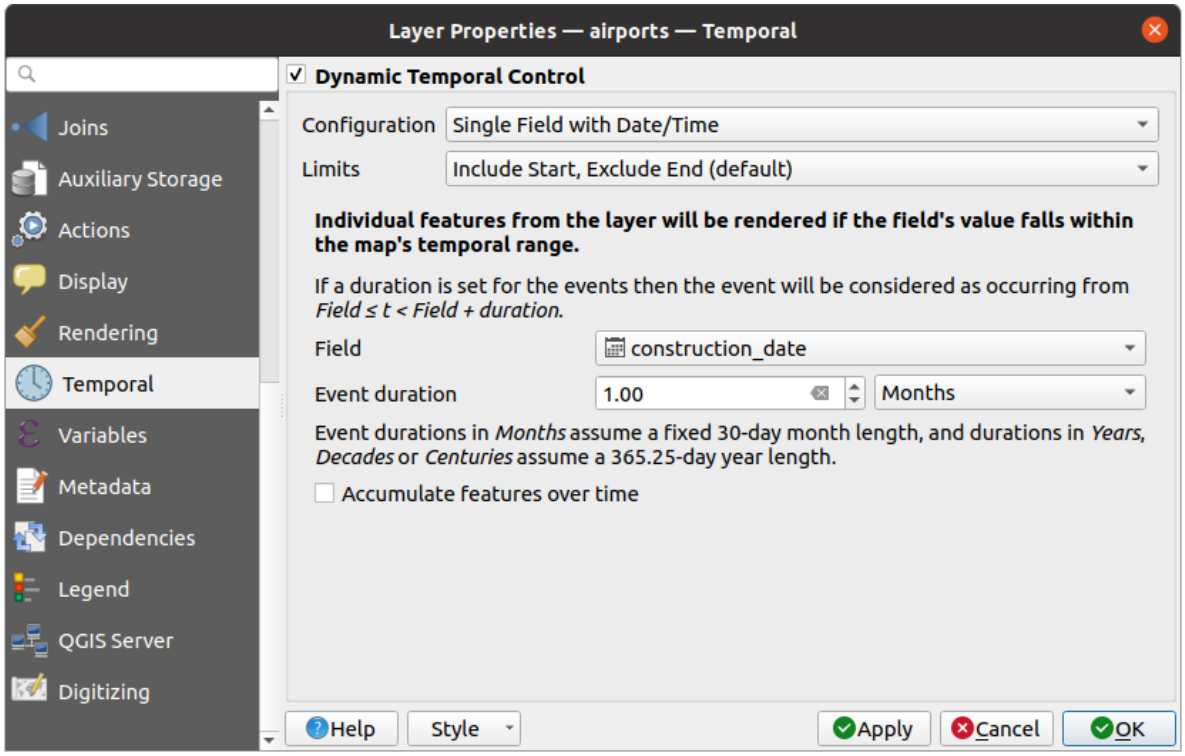


Fig. 16.61: Vector layer temporal properties dialog

Check the *Dynamic Temporal Control* option to configure the vector layer temporal rendering. Depending on the structure of your dataset, you may want to use one of the provided *Configuration* options:

- *Fixed time range*: all the features are rendered if the map canvas temporal frame overlaps the given *Start date* and *End date* range.
- *Single field with date/time*: features are rendered if their *Field*'s value falls within the map canvas temporal frame. An *Event duration* can be set. With checking the *Accumulate features over time* option, all features which occur before or within the map's temporal range will continue to be rendered. The event duration is thus ignored.
- *Separate fields for start and end date/time*: features are rendered if the range specified by their *Start field* and *End field* values overlaps the map canvas temporal.
- *Separate fields for start and event duration*: features are rendered if the range defined by their *Start field* and *Event duration field* values overlaps the map canvas temporal.
- *Start and end date/time from expressions*: features are rendered if the time range specified by the fields *Start expression* and *End expression* overlaps the map canvas temporal.
- *Redraw layer only*: the layer is redrawn at each new animation frame but no time-based filtering is applied to the features. It's useful when the layer uses time-based expression values for renderer settings (e.g. data-defined symbology).

It is also possible to set the *Limits* of the features time range as:

- *Include start, exclude end*
- *Include start, include end*

16.1.16 Proprietățile Variabilelor

The *Variables* tab lists all the variables available at the layer's level (which includes all global and project's variables).

It also allows the user to manage layer-level variables. Click the button to add a new custom layer-level variable. Likewise, select a custom layer-level variable from the list and click the button to remove it.

More information on variables usage in the General Tools *Storing values in Variables* section.

16.1.17 Elevation Properties

The *Elevation* tab provides options to control the layer elevation properties within a *3D map view* and its appearance in the *profile tool charts*. Specifically, you can set:

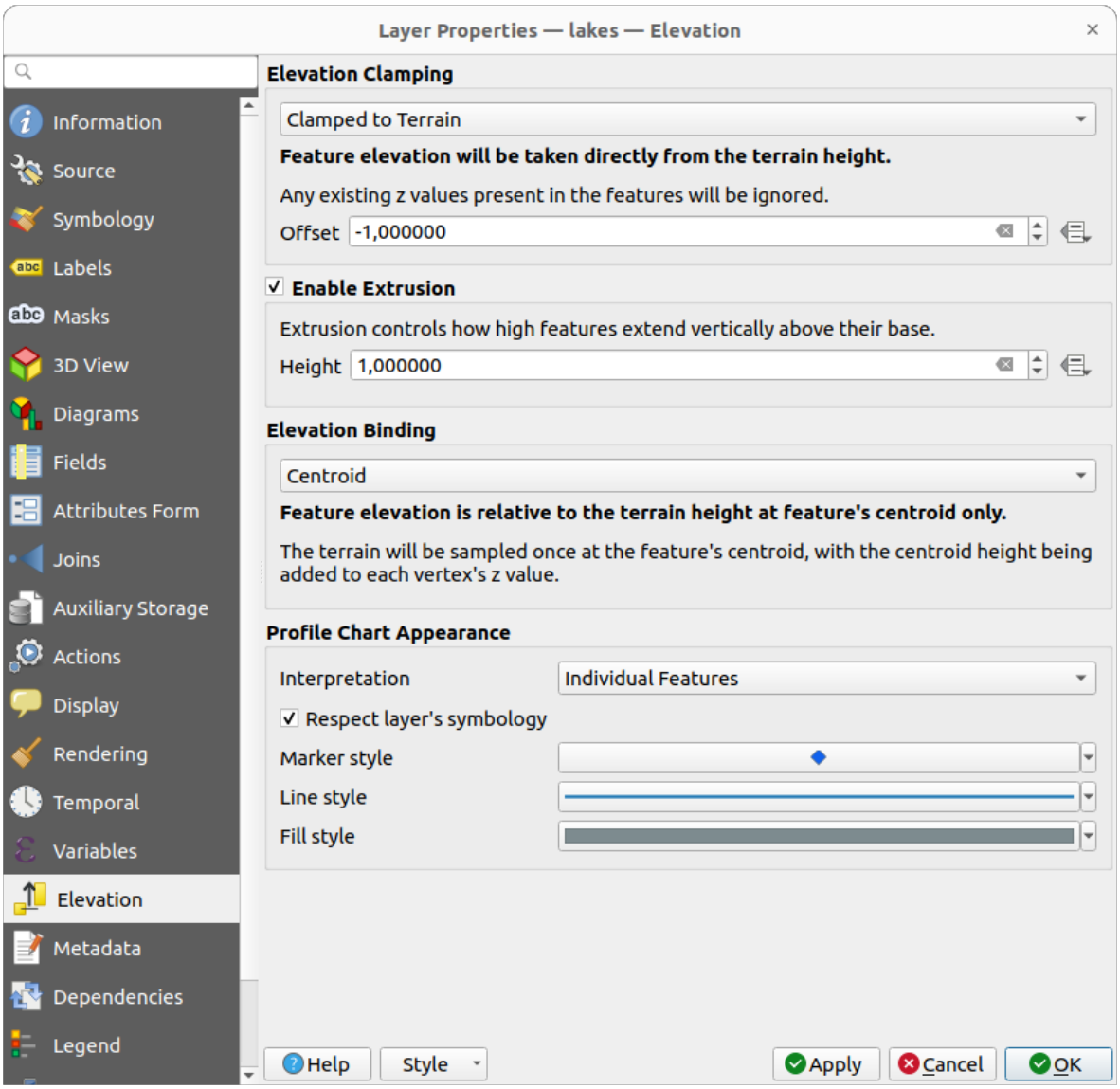


Fig. 16.62: Vector layer elevation properties dialog

- *Elevation Clamping*: defines how and whether the features altitude should be:

  - *Clamped to terrain*: takes elevation directly from the terrain height and ignores any existing Z values in the features. A data-defined *Offset* value from the terrain can also be filled.
  - *Relative to terrain*: any existing Z values in the features are added to the terrain height. A *Scale* factor followed by a data-defined *Offset* can be used to adjust the elevation. This option is not available for 2D geometry layers.
  - *Absolute*: ignores the terrain height and directly takes Z values from the features for the elevation. A *Scale* factor followed by a data-defined *Offset* can be used to adjust the elevation. For 2D geometry layers (with no Z values), a data-defined *Base height* can instead be set.

- *Enable extrusion*: you can set a *Height* to control how high features vertically extend above their base. This is convenient to indicate that a 2D geometry layers, e.g. a polygon building footprints layer, actually represents 3D objects.
- *Elevation Binding*: only relevant when combining an *Elevation clamping* relying on the terrain with a line or polygon layer, this option controls how feature elevation is set relative to the terrain height. The terrain can be sampled:
  - at the feature's *Centroid*, with the centroid height being added to each vertex's z value
  - at every individual *Vertex* before being added to the vertex's z value
- *Profile Chart Appearance*: controls how features are rendered when drawing a profile chart. Two main *Interpretation* modes are available:
  - as *Individual features*: samples discrete positions where the cross section profile line intersects the vector features. That intersection can be represented as point, line or surface depending on the layer type and whether an extrusion is applied.

    With checking *Respect layer symbology*, features will be rendered on the profile chart with their corresponding *layer styling* (allowing e.g. categorized classes to be visible on the profile chart). If the profile symbol type does not match the layer's renderer symbol types, only the symbol color from the renderer is applied to the profile symbol.

    Depending on the layer settings, profile symbols can be represented with a custom style, using:

    * *Marker style*: for non-extruded point and line features, and for non-extruded polygon features touched by the profile line
    * *Line style*: for extruded point and line features, and for non-extruded polygon features intersected by the profile line
    * *Fill style*: for extruded polygon features
  - as *Continuous Surface (e.g. contours)*: the elevation chart will be rendered as a surface instead of separate features by joining the sampled elevation results into a continuous line. This can enhance visualisation and is designed for vector layers which represent a continuous elevation surface, e.g. contour lines or surveyed elevation points. The profile *Style* can be set as:

    * a *Line* with a specific *Line style*
    * an elevation surface rendered using a fill symbol either above (*Fill above*) or below (*Fill below*) the elevation curve line. The surface symbology is represented using:
      · a *Fill style*
      · and a *Limit*: the maximum (respectively minimum) altitude determining how high the fill surface will be

    Moreover, you can check *Show markers at sampled points* to make them visible over the interpretation line and assign them a *Marker style*.

16.1.18 Proprietăți Metadate

The *Metadata* tab provides you with options to create and edit a metadata report on your layer. See *Metadata* for more information.

16.1.19 Dependencies Properties

The *Dependencies* tab allows to declare data dependencies between layers. A data dependency occurs when a data modification in a layer, not by direct user manipulation, may modify data of other layers. This is the case for instance when geometry of a layer is updated by a database trigger or custom PyQGIS scripting after modification of another layer's geometry.

In the *Dependencies* tab, you can select any layers which may externally alter the data in the current layer. Correctly specifying dependent layers allows QGIS to invalidate caches for this layer when the dependent layers are altered.

16.1.20 Proprietățile Legendei

The *Legend* properties tab provides you with advanced settings for the *Layers panel* and/or the *print layout legend*. These options include:

- Depending on the symbology applied to the layer, you may end up with several entries in the legend, not necessarily readable/useful to display. The *Legend placeholder image* helps you *select an image* for replacement, displayed both in the *Layers* panel and the print layout legend.
- *Show label legend*: Displays overviews of the different label settings as entries in the legends. The *label style* is previewed along with the description.
- *Text on symbols*: In some cases it can be useful to add extra information to the symbols in the legend. With this frame, you can affect to any of the symbols used in the layer symbology a text that is displayed over the symbol, in both *Layers* panel and print layout legend. This mapping is done by typing each text next to the symbol in the table widget or filling the table using the *Set Labels from Expression* button. Text appearance is handled through the font and color selector widgets of the *Text Format* button.

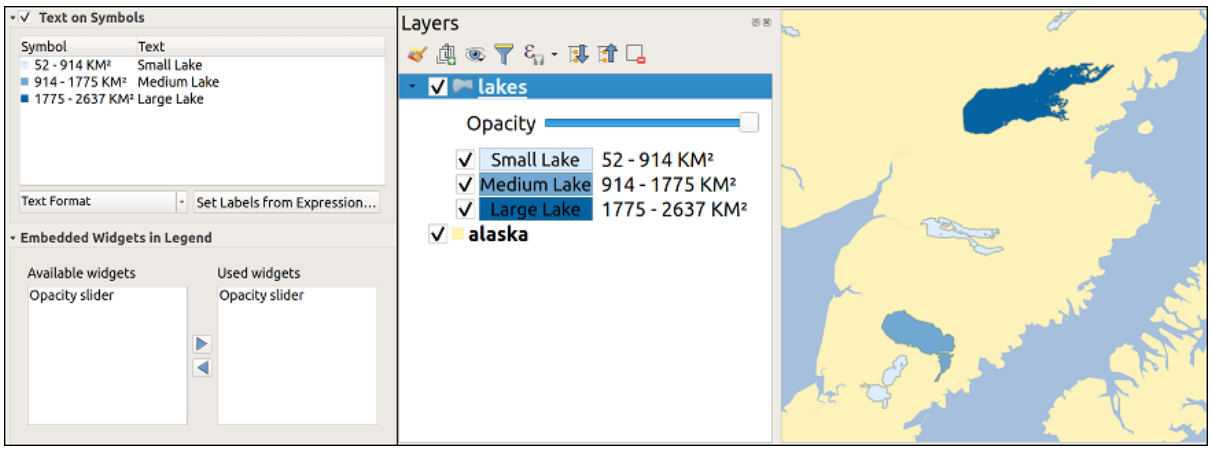


Fig. 16.63: Setting text on symbols (left) and its rendering in the *Layers* panel (right)

- a list of widgets you can embed within the layer tree in the Layers panel. The idea is to have a way to quickly access some actions that are often used with the layer (setup transparency, filtering, selection, style or other stuff…).

  By default, QGIS provides transparency widget but this can be extended by plugins registering their own widgets and assign custom actions to layers they manage.

16.1.21 Proprietățile Serverului QGIS

The *QGIS Server* tab consists of *Description*, *Attribution*, *Metadata URL*, and *Legend URL* sections.

From the *Description* section, you can change the *Short name* used to reference the layer in requests (to learn more about short names, read services_basics_short_name). You can also add or edit a *Title* and *Abstract* for the layer, or define a *Keyword list* here. These keyword lists can be used in a metadata catalog. If you want to use a title from an XML metadata file, you have to fill in a link in the *Data URL* field.

Use *Attribution* to get attribute data from an XML metadata catalog.

In *Metadata URL*, you can add the general paths to the XML metadata catalog. This information will be saved in the QGIS project file for subsequent sessions and will be used for QGIS Server.

In the *Legend URL* section, you can provide the url of a legend image in the url field. You can use the Format drop-down option to apply the appropriate format of the image. Currently png, jpg and jpeg image formats are supported.

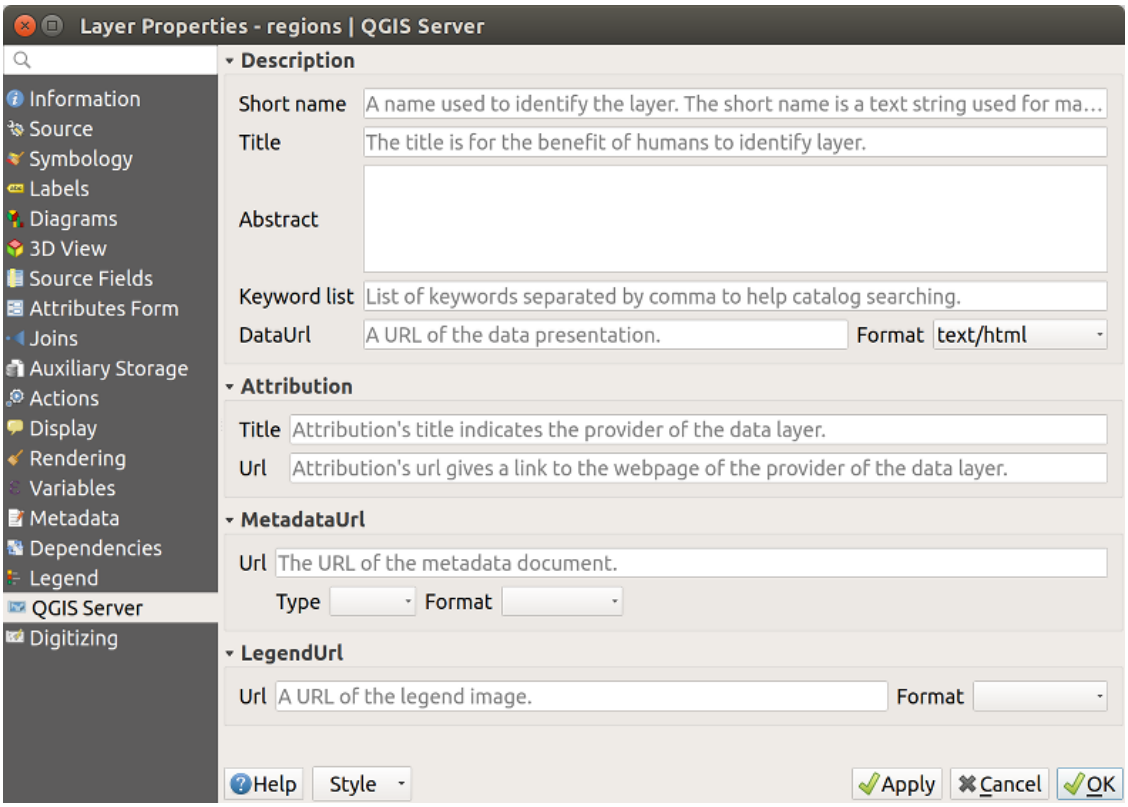


Fig. 16.64: QGIS Server tab in vector layers properties dialog

Pentru a afla mai multe despre QGIS Server, citiți QGIS-Server-manual.

16.1.22 Digitizing Properties

The *Digitizing* tab gives access to options that help to ensure the quality of digitized geometries.

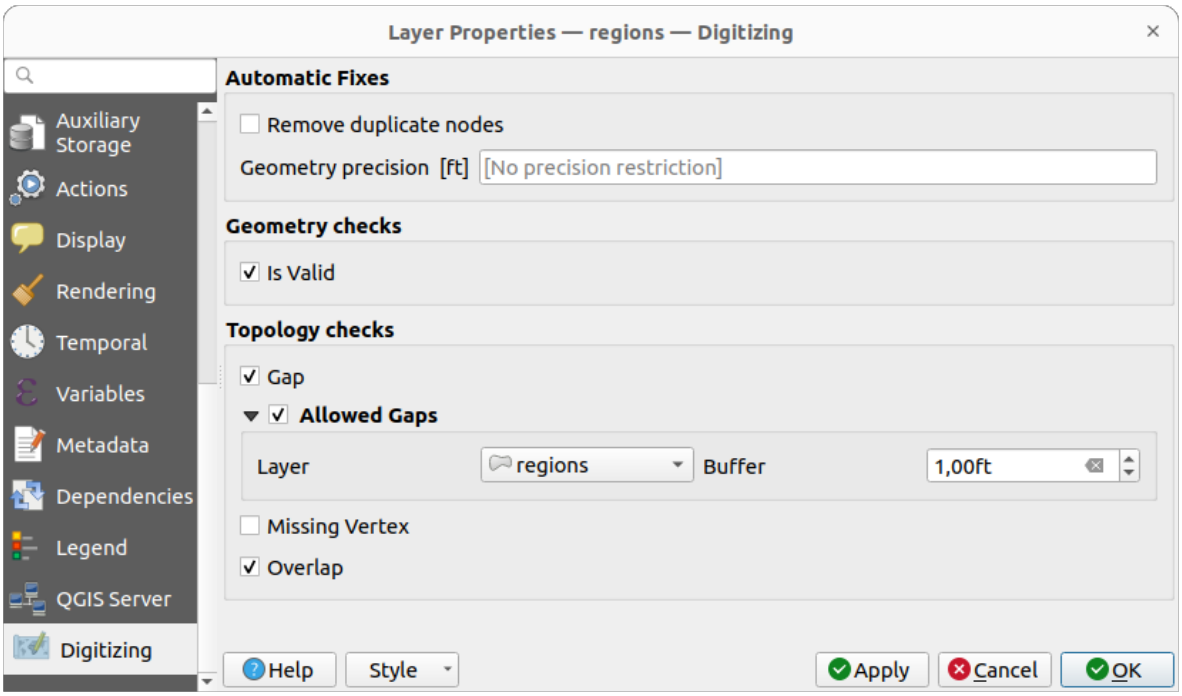


Fig. 16.65: The QGIS Digitizing tab in the vector layers properties dialog

Automatic Fixes

Options in the *Automatic Fixes* section will directly affect the vertices of any geometry which is added or modified. If the *Remove duplicate nodes* option is checked, any two subsequent vertices with exactly the same coordinates will be removed. If the *Geometry precision* is set, all vertices will be rounded to the closest multiple of the configured geometry precision. The rounding will happen in the layer coordinate reference system. Z and M values are not rounded. With many map tools, a grid is shown on the canvas while digitizing.

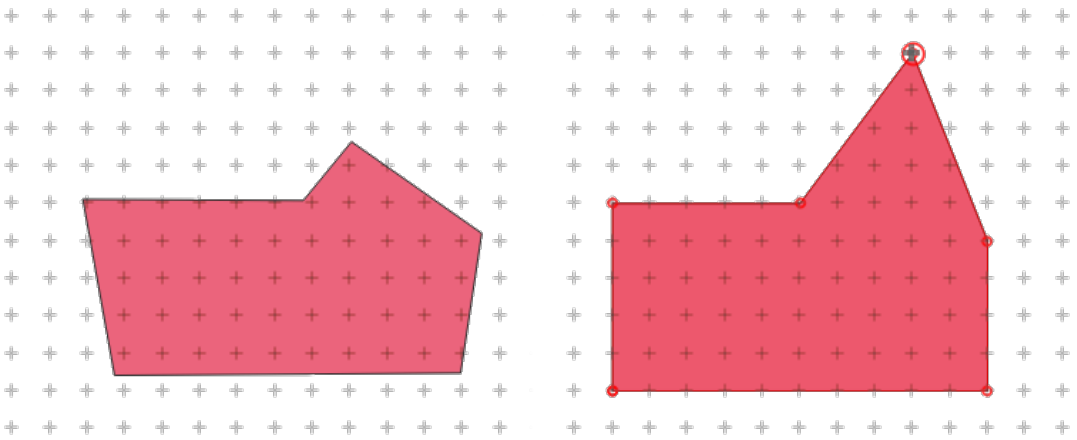


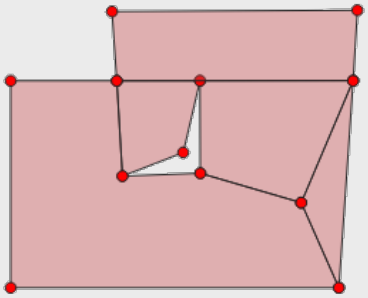
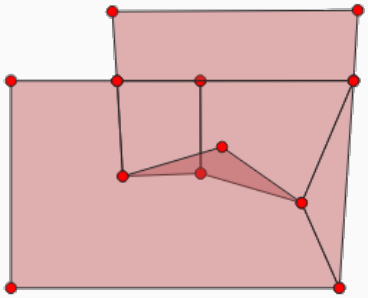
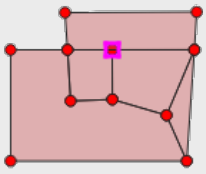
Fig. 16.66: Moving the top vertex snaps all the vertices to the grid

Verificări de Geometrie

In the *Geometry checks* section, additional validations on a per geometry basis can be activated. Immediately after any geometry modification, failures in these checks are reported to the user in the *Geometry validation* panel. As long as a check is failing, it is not possible to save the layer. The *Is valid* check will run basic validity checks like self intersection on geometries.

Topology Checks

In the *Topology checks* section, additional topology validation checks can be activated. Topology checks will be executed when the user saves the layer. Check errors will be reported in the *Geometry validation* panel. As long as validation errors are present, the layer can not be saved. Topology checks are executed in the area of the bounding box of the modified features. Since other features may be present in the same area, topological errors concerning these features are reported as well as errors introduced in the current edit session.

Topology check option	Ilustrație
The *Gap* check will check for gaps between neighbouring polygons.	
The *Overlap* check will check for overlaps between neighbouring polygons.	
The *Missing vertex* check will check for shared boundaries of neighbouring polygons where one border misses a vertex which is present on the other one.	

Gap check exceptions

Sometimes it is desirable to keep gaps inside an area in a polygon layer that otherwise is fully covered by polygons. For example, a land use layer may have acceptable holes for lakes. It is possible to define areas that are ignored in the gap check. Since gaps inside these areas are allowed, we will refer to them as *Allowed Gaps* areas.

In the options for the gap checks under *Allowed Gaps*, an *Allowed Gaps layer* can be configured.

Whenever the gap check is executed, gaps which are covered by one or more polygons in the *Allowed Gaps Layer* are not reported as topology errors.

It is also possible to configure an additional *Buffer*. This buffer is applied to each polygon on the *Allowed Gaps Layer*. This makes it possible to make the tests less susceptible to small changes in the outlines at the borders of gaps.

When *Allowed Gaps* are enabled, an additional button (*Add Allowed Gap*) for detected gap errors is available in the geometry validation dock, where gaps are reported during digitizing. If the *Add Allowed Gap* button is pushed, a new polygon with the geometry of the detected gap is inserted into the *Allowed Gaps Layer*. This makes it possible to quickly flag gaps as allowed.

Geometry validation panel

The *Geometry Validation* panel is triggered when any of the abovementioned digitizing checks finds an error. The dialog provides you with the list of errors and their description, and you can to browse the list using the keyboard arrows or dedicated arrows.

You'll need to address all the issues before you can save edits to the layer. To do so:

1. Select an error, and it's possible to:
   - *Zoom to Feature(s)*
   - *Zoom to problem*
2. Pick the usual *digitizing tools* to fix the issue.

16.2 Lucrul cu Tabela de Atribute

The attribute table displays information on features of a selected layer. Each row in the table represents a feature (with or without geometry), and each column contains a particular piece of information about the feature. Features in the table can be searched, selected, moved or even edited.

16.2.1 Foreword: Spatial and non-spatial tables

QGIS allows you to load spatial and non-spatial layers. This currently includes tables supported by GDAL and delimited text, as well as the PostgreSQL, MS SQL Server, SpatiaLite and Oracle providers. All loaded layers are listed in the *Layers* panel. Whether a layer is spatially enabled or not determines whether you can interact with it on the map.

Non-spatial tables can be browsed and edited using the attribute table view. Furthermore, they can be used for field lookups. For example, you can use columns of a non-spatial table to define attribute values, or a range of values that are allowed, to be added to a specific vector layer during digitizing. Have a closer look at the edit widget in section *Attributes Form Properties* to find out more.

16.2.2 Introducing the attribute table interface

To open the attribute table for a vector layer, activate the layer by clicking on it in the *Panoul Straturilor*. Then, from the main *Layer* menu, choose *Open Attribute Table*. It is also possible to right-click on the layer and choose *Open Attribute Table* from the drop-down menu, or to click on the *Open Attribute Table* button in the Attributes toolbar. If you prefer shortcuts, `F6` will open the attribute table. `Shift+F6` will open the attribute table filtered to selected features and `Ctrl+F6` will open the attribute table filtered to visible features.

This will open a new window that displays the feature attributes for the layer (*figure_attributes_table*). According to the setting in *Settings* ► *Options* ► *Data sources* menu, the attribute table will open in a docked window or a regular window. The total number of features in the layer and the number of currently selected/filtered features are shown in the attribute table title, as well as if the layer is spatially limited.

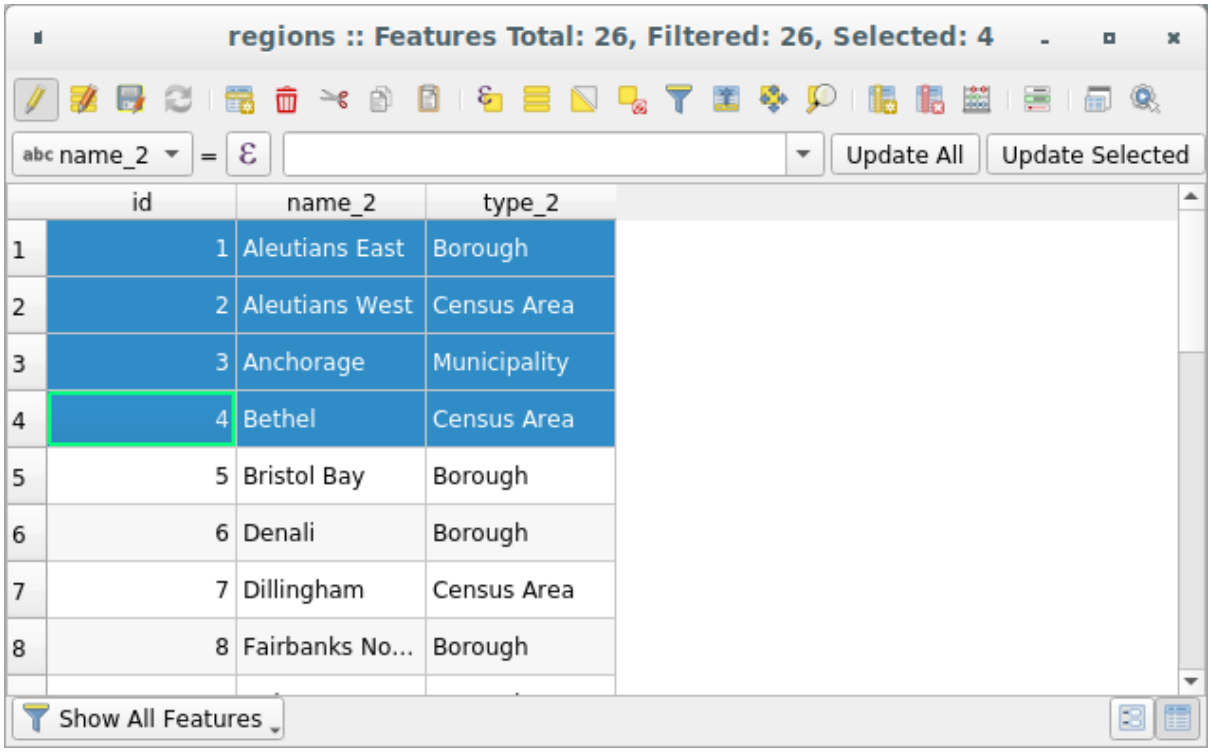


Fig. 16.67: Tabela de Atribute pentru stratul regiunilor

Butoanele din partea de sus a ferestrei tabelei de atribute oferă următoarele funcționalități:

Tabelul 16.1: Available Tools

Pictogramă	Etichetă	Scop	Default Shortcut
	Comutare mod de editare	Enable editing functionalities	`Ctrl+E`
	Toggle multi edit mode	Update multiple fields of many features	
	Save Edits	Save current modifications	
	Reload the table		
	Add feature	Add new geometryless feature	
	Șterge entitățile selectate	Remove selected features from the layer	
	Cut selected features to clipboard		`Ctrl+X`

continues on next page

Tabelul 16.1 – continuare din pagina precedentă

Pictogramă	Etichetă	Scop	Default Shortcut
	Copy selected features to clipboard		Ctrl+C
	Paste features from clipboard	Insert new features from copied ones	Ctrl+V
	Select features using an Expression		
	Selectează Tot	Select all features in the layer	Ctrl+A
	Invert selection	Invert the current selection in the layer	Ctrl+R
	Deselect all	Deselect all features in the current layer	Ctrl+Shift+A
	Filter/Select features using form		Ctrl+F
	Move selected to top	Move selected rows to the top of the table	
	Pan map to the selected rows		Ctrl+P
	Zoom map to the selected rows		Ctrl+J
	New field	Add a new field to the data source	Ctrl+W
	Delete field	Remove a field from the data source	
	Organize columns	Show/hide fields from the attribute table	
	Open field calculator	Update field for many features in a row	Ctrl+I
	Formatare condiționată	Enable table formatting	
	Dock attribute table	Allows to dock/undock the attribute table	
	Actions	Lists the actions related to the layer	

Notă: Depending on the format of the data and the GDAL library built with your QGIS version, some tools may not be available.

Below these buttons is the Quick Field Calculation bar (enabled only in *edit mode*), which allows to quickly apply calculations to all or part of the features in the layer. This bar uses the same *expressions* as the Field Calculator (see *Editarea valorilor atributului*).

Table view vs Form view

QGIS provides two view modes to easily manipulate data in the attribute table:

- The Table view, displays values of multiple features in a tabular mode, each row representing a feature and each column a field. A right-click on the column header allows you to *configure the table display* while a right-click on a cell provides *interaction with the feature*.

  The attribute table supports `Shift+Mouse Wheel` scrolling in table view mode to switch between vertical and horizontal scrolling movements. This can also be achieved replacing the mouse with the trackpad on macOS.

- The Form view shows *feature identifiers* in a first panel and displays only the attributes of the clicked identifier in the second one. There is a pull-down menu at the top of the first panel where the „identifier” can be specified using an attribute (*Column preview*) or an *Expression*. The pull-down also includes the last 10 expressions for re-use. Form view uses the layer fields configuration (see *Attributes Form Properties*).

  You can browse through the feature identifiers with the arrows on the bottom of the first panel. The features attributes update in the second panel as you go. It's also possible to identify or move to the active feature in

the map canvas with pushing down any of the button at the bottom:

- Highlight current feature if visible in the map canvas
- Automatically pan to current feature
- Zoom to current feature

You can switch from one mode to the other by clicking the corresponding icon at the bottom right of the dialog.

You can also specify the *Default view* mode at the opening of the attribute table in *Settings* ► *Options* ► *Data Sources* menu. It can be «Remember last view», «Table view» or «Form view».

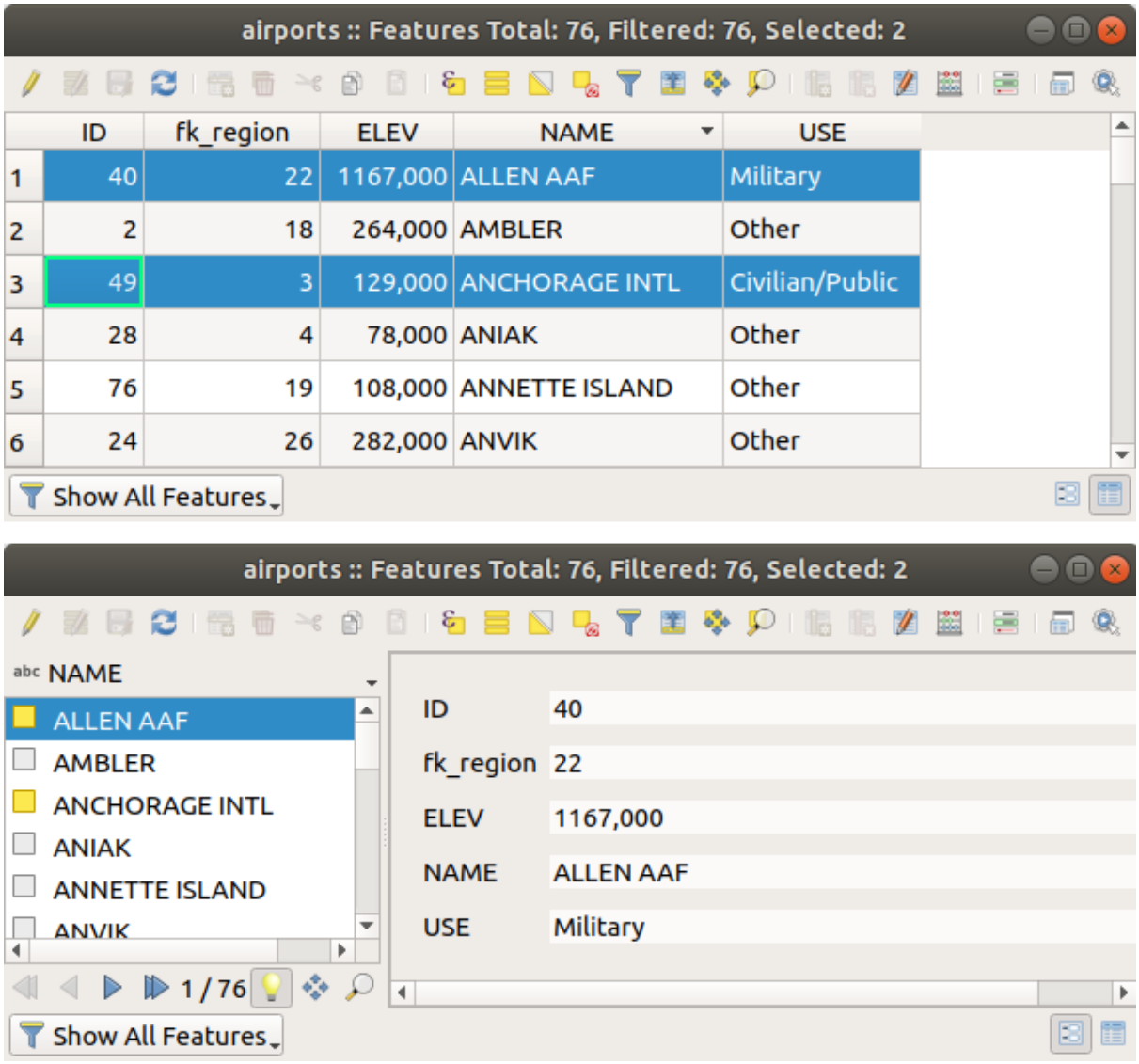


Fig. 16.68: Attribute table in table view (top) vs form view (bottom)

Configuring the columns

Right-click in a column header when in table view to have access to tools that help you control:

- the *column(s) size*
- the *column(s) visibility and order*
- the *sort order of the data*

Resizing columns widths

Columns width can be set through a right-click on the column header and select either:

- *Set width…* to enter the desired value. By default, the current value is displayed in the widget
- *Set all column widths…* to the same value
- *Autosize* to resize at the best fit the column.
- *Autosize all columns*

A column size can also be changed by dragging the boundary on the right of its heading. The new size of the column is maintained for the layer, and restored at the next opening of the attribute table.

Hiding and organizing columns and enabling actions

By right-clicking in a column header, you can choose to *Hide column* from the attribute table (in „table view" mode). For more advanced controls, press the Organize columns… button from the dialog toolbar or choose *Organize columns…* in a column header contextual menu. In the new dialog, you can:

- check/uncheck columns you want to show or hide: a hidden column will disappear from every instances of the attribute table dialog until it is actively restored.
- drag-and-drop items to reorder the columns in the attribute table. Note that this change is for the table rendering and does not alter the fields order in the layer datasource
- add a new virtual *Actions* column that displays in each row a drop-down box or a button list of enabled actions. See *Proprietățile Acțiunilor* for more information about actions.

Sorting columns

The table can be sorted by any column, by clicking on the column header. A small arrow indicates the sort order (downward pointing means descending values from the top row down, upward pointing means ascending values from the top row down). You can also choose to sort the rows with the *Sort…* option of the column header context menu and write an expression. E.g. to sort the rows using multiple columns you can write `concat(col0, col1)`.

In form view, features identifier can be sorted using the *Sort by preview expression* option.

Sfat: Sorting based on columns of different types

Trying to sort an attribute table based on columns of string and numeric types may lead to unexpected result because of the `concat("USE", "ID")` expression returning string values (ie, `'Borough105' < 'Borough6'`). You can workaround this by using eg `concat("USE", lpad("ID", 3, 0))` which returns `'Borough105' > 'Borough006'`.

Formatting of table cells using conditions

Conditional formatting settings can be used to highlight in the attribute table features you may want to put a particular focus on, using custom conditions on feature's:

- geometry (e.g., identifying multi-parts features, small area ones or in a defined map extent…);
- or field value (e.g., comparing values to a threshold, identifying empty cells, duplicates, …).

You can enable the conditional formatting panel clicking on Conditional formatting button at the top right of the attributes window in table view (not triggered in form view).

The new panel allows user to add new rules to format rendering of *Field* or *Full row*. Adding new rule opens a form to define:

- the name of the rule;
- a condition using any of the *expression builder* functions;
- the formatting: it can be choosen from a list of predefined formats or created based on properties like:
  - background and text colors;
  - use of icon;
  - bold, italic, underline, or strikeout;
  - fontul.

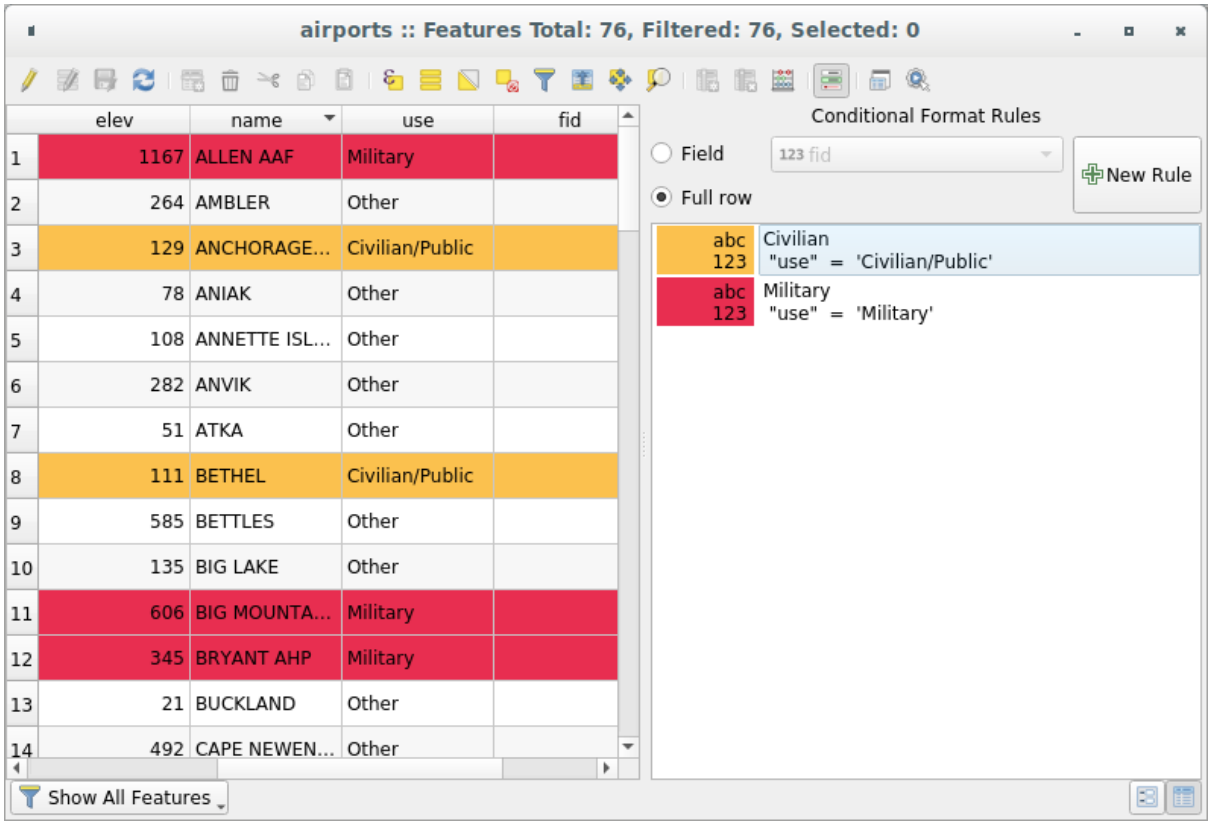


Fig. 16.69: Formatarea condițională a unei tabelelor cu atribute

16.2.3 Interacting with features in an attribute table

Selectare entități

In table view, each row in the attribute table displays the attributes of a unique feature in the layer. Selecting a row selects the feature and likewise, selecting a feature in the map canvas (in case of geometry enabled layer) selects the row in the attribute table. If the set of features selected in the map canvas (or attribute table) is changed, then the selection is also updated in the attribute table (or map canvas) accordingly.

Rows can be selected by clicking on the row number on the left side of the row. **Multiple rows** can be marked by holding the `Ctrl` key. A **continuous selection** can be made by holding the `Shift` key and clicking on several row headers on the left side of the rows. All rows between the current cursor position and the clicked row are selected. Moving the cursor position in the attribute table, by clicking a cell in the table, does not change the row selection. Changing the selection in the main canvas does not move the cursor position in the attribute table.

In form view of the attribute table, features are by default identified in the left panel by the value of their displayed field (see *Display Properties*). This identifier can be replaced using the drop-down list at the top of the panel, either by selecting an existing field or using a custom expression. You can also choose to sort the list of features from the drop-down menu.

Click a value in the left panel to display the feature's attributes in the right one. To select a feature, you need to click inside the square symbol at the left of the identifier. By default, the symbol turns into yellow. Like in the table view, you can perform multiple feature selection using the keyboard combinations previously exposed.

Beyond selecting features with the mouse, you can perform automatic selection based on feature's attribute using tools available in the attribute table toolbar, such as (see section *Automatic selection* and following one for more information and use case):

- *Selectează După Expresie…*
- *Selectează Entitățile După Valoare…*
- *Deselectează Toate Entitățile dintr-un Strat*
- *Selectează Toate Entitățile*
- *Invert Feature Selection*.

It is also possible to select features using the *Filtering and selecting features using forms*.

Filtrarea entităților

Once you have selected features in the attribute table, you may want to display only these records in the table. This can be easily done using the *Show Selected Features* item from the drop-down list at the bottom left of the attribute table dialog. This list offers the following filters:

- *Show All Features*
- *Show Selected Features* - same as using *Open Attribute Table (Selected Features)* from the *Layer* menu or the *Attributes Toolbar* or pressing `Shift+F6`
- *Show Features visible on map* - same as using *Open Attribute Table (Visible Features)* from the *Layer* menu or the *Attributes Toolbar* or pressing `Ctrl+F6`
- *Show Features with Failing Constraints* - features will be filtered to only show the ones which have failing *constraints*. Depending on whether the unmet constraint is hard or soft, failing field values are displayed in respectively dark or light orange cells.
- *Show Edited and New Features* - same as using *Open Attribute Table (Edited and New Features)* from the *Layer* menu or the *Attributes Toolbar*

- *Field Filter* - allows the user to filter based on value of a field: choose a column from a list, type or select a value and press `Enter` to filter. Then, only the features matching `num_field = value` or `string_field ilike '%value%'` expression are shown in the attribute table. You can check *Case sensitive* to be less permissive with strings.
- *Advanced filter (Expression)* - Opens the expression builder dialog. Within it, you can create *complex expressions* to match table rows. For example, you can filter the table using more than one field. When applied, the filter expression will show up at the bottom of the form.
- *Stored filter expressions* ►: a shortcut to *saved expressions* frequently used for filtering your attribute table.

It is also possible to *filter features using forms*.

Notă: Filtering records out of the attribute table does not filter features out of the layer; they are simply momentaneously hidden from the table and can be accessed from the map canvas or by removing the filter. For filters that do hide features from the layer, use the *Query Builder*.

Sfat: Update datasource filtering with `Show Features Visible on Map`

When for performance reasons, features shown in attribute table are spatially limited to the canvas extent at its opening (see *Data Source Options* for a how-to), selecting *Show Features Visible on Map* on a new canvas extent updates the spatial restriction.

Storing filter expressions

Expressions you use for attribute table filtering can be saved for further calls. When using *Field Filter* or *Advanced Filter (expression)* entries, the expression used is displayed in a text widget in the bottom of the attribute table dialog. Press the Save expression with text as name next to the box to save the expression in the project. Pressing the drop-down menu next to the button allows to save the expression with a custom name (*Save expression as...*). Once a saved expression is displayed, the button is triggered and its drop-down menu allows you to *Edit the expression* and name if any, or *Delete stored expression*.

Saved filter expressions are saved in the project and available through the *Stored filter expressions* menu of the attribute table. They are different from the *user expressions*, shared by all projects of the active user profile.

Filtering and selecting features using forms

Clicking the Filter/Select features using form or pressing `Ctrl+F` will make the attribute table dialog switch to form view and replace each widget with its search variant.

From this point onwards, this tool functionality is similar to the one described in *Select Features By Value*, where you can find descriptions of all operators and selecting modes.

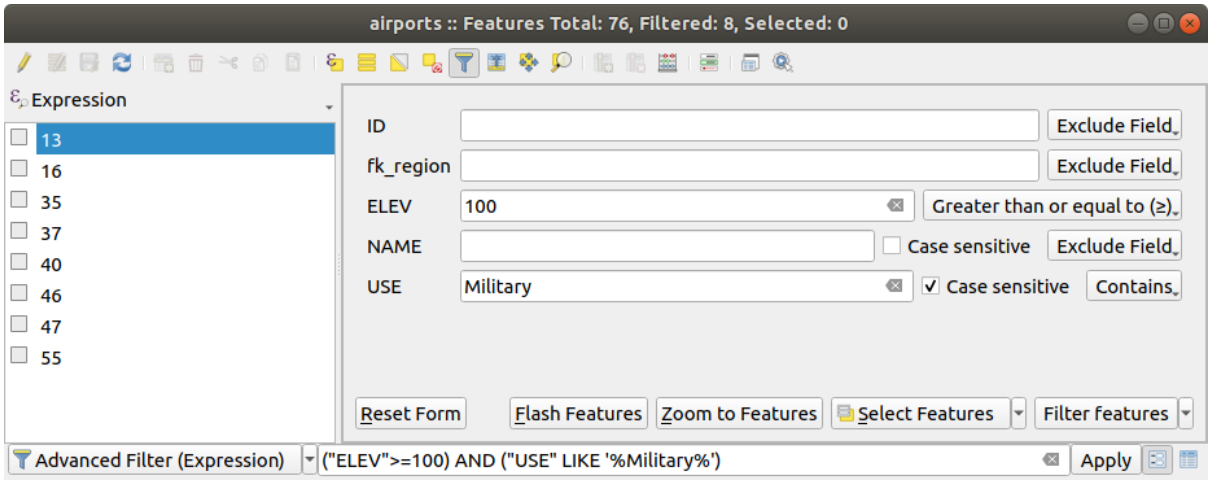


Fig. 16.70: Attribute table filtered by the filter form

When selecting / filtering features from the attribute table, there is a *Filter features* button that allows defining and refining filters. Its use triggers the *Advanced filter (Expression)* option and displays the corresponding filter expression in an editable text widget at the bottom of the form.

If there are already filtered features, you can refine the filter using the drop-down list next to the *Filter features* button. The options are:

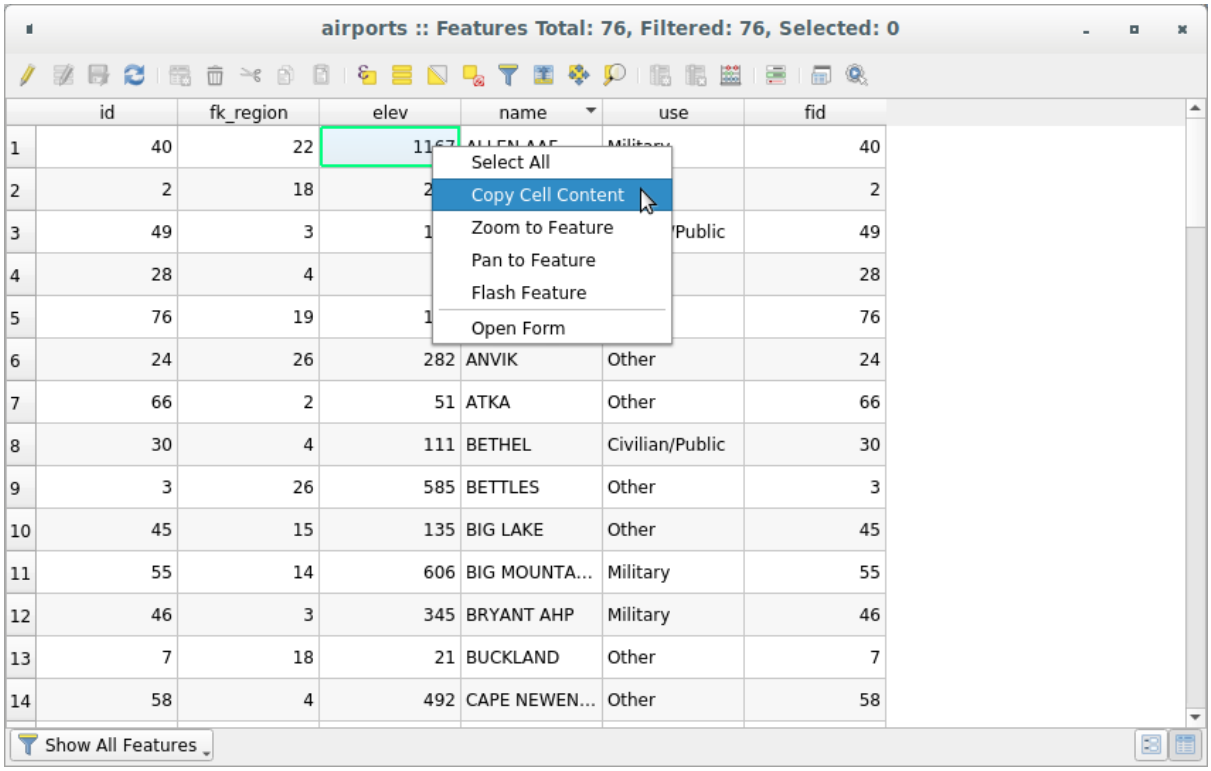
- *Filter within („AND”)*
- *Extend filter („OR”)*

To clear the filter, either select the *Show all features* option from the bottom left pull-down menu, or clear the expression and click *Apply* or press `Enter`.

16.2.4 Using action on features

Users have several possibilities to manipulate feature with the contextual menu like:

- *Selectați toate* (`Ctrl+A`) entitățile;
- Copy the content of a cell in the clipboard with *Copy cell content*;
- *Zoom to feature* without having to select it beforehand;
- *Pan to feature* without having to select it beforehand;
- *Flash feature*, to highlight it in the map canvas;
- *Open form*: it toggles attribute table into form view with a focus on the clicked feature.



	id	fk_region	elev	name	use	fid
1	40	22	11			40
2	2	18	2			2
3	49	3	1		Public	49
4	28	4				28
5	76	19	1			76
6	24	26	282	ANVIK	Other	24
7	66	2	51	ATKA	Other	66
8	30	4	111	BETHEL	Civilian/Public	30
9	3	26	585	BETTLES	Other	3
10	45	15	135	BIG LAKE	Other	45
11	55	14	606	BIG MOUNTA...	Military	55
12	46	3	345	BRYANT AHP	Military	46
13	7	18	21	BUCKLAND	Other	7
14	58	4	492	CAPE NEWEN...	Other	58

Fig. 16.71: Copy cell content button

If you want to use attribute data in external programs (such as Excel, LibreOffice, QGIS or a custom web application), select one or more row(s) and use the Copy selected rows to clipboard button or press `Ctrl+C`.

In *Settings* ► *Options* ► *Data Sources* menu you can define the format to paste to with *Copy features as* dropdown list:

- Plain text, no geometry,
- Plain text, WKT geometry,
- GeoJSON

You can also display a list of actions in this contextual menu. This is enabled in the *Layer properties* ► *Actions* tab. See *Proprietățile Acțiunilor* for more information on actions.

Saving selected features as new layer

The selected features can be saved as any OGR-supported vector format and also transformed into another coordinate reference system (CRS). In the contextual menu of the layer, from the *Layers* panel, click on *Export* ► *Save selected features as...* to define the name of the output dataset, its format and CRS (see section *Creating new layers from an existing layer*). You'll notice that *Save only selected features* is checked. It is also possible to specify GDAL creation options within the dialog.

16.2.5 Editarea valorilor atributului

Editing attribute values can be done by:

- typing the new value directly in the cell, whether the attribute table is in table or form view. Changes are hence done cell by cell, feature by feature;
- using the *field calculator*: update in a row a field that may already exist or to be created but for multiple features. It can be used to create virtual fields;
- using the quick field *calculation bar*: same as above but for only existing field;
- or using the *multi edit* mode: update in a row multiple fields for multiple features.

Using the Field Calculator

The Field Calculator button in the attribute table allows you to perform calculations on the basis of existing attribute values or defined functions, for instance, to calculate length or area of geometry features. The results can be used to update an existing field, or written to a new field (that can be a *virtual* one).

The field calculator is available on any layer that supports edit. When you click on the field calculator icon the dialog opens (see Fig. 16.72). If the layer is not in edit mode, a warning is displayed and using the field calculator will cause the layer to be put in edit mode before the calculation is made.

Based on the *Expression Builder* dialog, the field calculator dialog offers a complete interface to define an expression and apply it to an existing or a newly created field. To use the field calculator dialog, you must select whether you want to:

1. apply calculation on the whole layer or on selected features only
2. create a new field for the calculation or update an existing one.

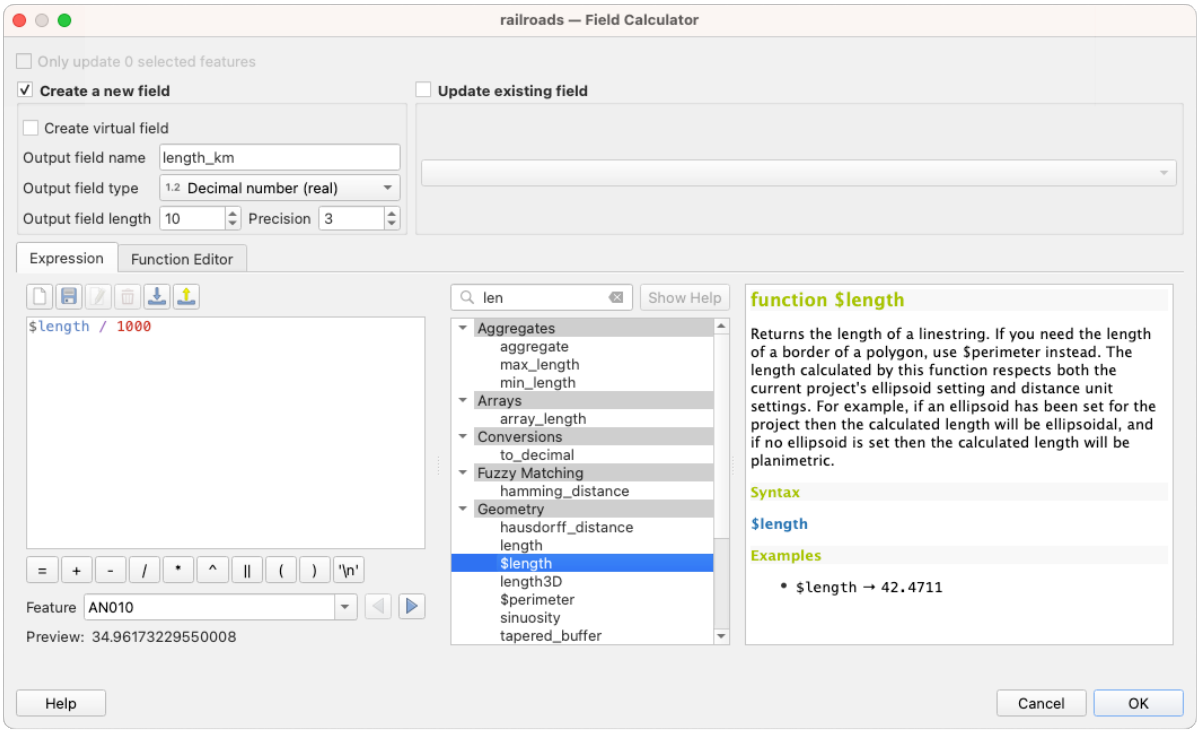


Fig. 16.72: Calculatorul de Câmpuri

If you choose to add a new field, you need to enter a field name, a field type (integer, real, date or string) and if needed, the total field length and the field precision. For example, if you choose a field length of 10 and a field precision of 3,

it means you have 7 digits before the dot, and 3 digits for the decimal part.

A short example illustrates how field calculator works when using the *Expression* tab. We want to calculate the length in km of the `railroads` layer from the QGIS sample dataset:

1. Încărcați fișierul shape `railroads.shp` în QGIS și efectuați pe Deschiderea Tabelei de Atribute.
2. Clic pe Comutare în modul de editiare, apoi deschideți dialogul Calculatorului de Câmpuri.
3. Selectați caseta de bifare pentru *Crearea unui nou câmp*, în scopul salvării calculelor într-un nou câmp.
4. Set *Output field name* to `length_km`
5. Select `Decimal number (real)` as *Output field type*
6. Set the *Output field length* to `10` and the *Precision* to `3`
7. Double click on `$length` in the *Geometry* group to add the length of the geometry into the Field calculator expression box (you will begin to see a preview of the output, up to 60 characters, below the expression box updating in real-time as the expression is assembled).
8. Complete the expression by typing `/ 1000` in the Field calculator expression box and click *OK*.
9. You can now find a new *length_km* field in the attribute table.

Creating a Virtual Field

A virtual field is a field based on an expression calculated on the fly, meaning that its value is automatically updated as soon as an underlying parameter changes. The expression is set once; you no longer need to recalculate the field each time underlying values change. For example, you may want to use a virtual field if you need area to be evaluated as you digitize features or to automatically calculate a duration between dates that may change (e.g., using `now()` function).

Notă: Utilizarea Câmpurilor Virtuale

- Virtual fields are not permanent in the layer attributes, meaning that they're only saved and available in the project file they've been created.
- A field can be set virtual only at its creation. Virtual fields are marked with a purple background in the fields tab of the layer properties dialog to distinguish them from regular physical or joined fields. Their expression can be edited later by pressing the expression button in the Comment column. An expression editor window will be opened to adjust the expression of the virtual field.

Using the Quick Field Calculation Bar

While Field calculator is always available, the quick field calculation bar on top of the attribute table is only visible if the layer is in edit mode. Thanks to the expression engine, it offers a quicker access to edit an already existing field:

1. Select the field to update in the drop-down list.
2. Fill the textbox with a value, an expression you directly write or build using the expression button.
3. Click on *Update All*, *Update Selected* or *Update Filtered* button according to your need.

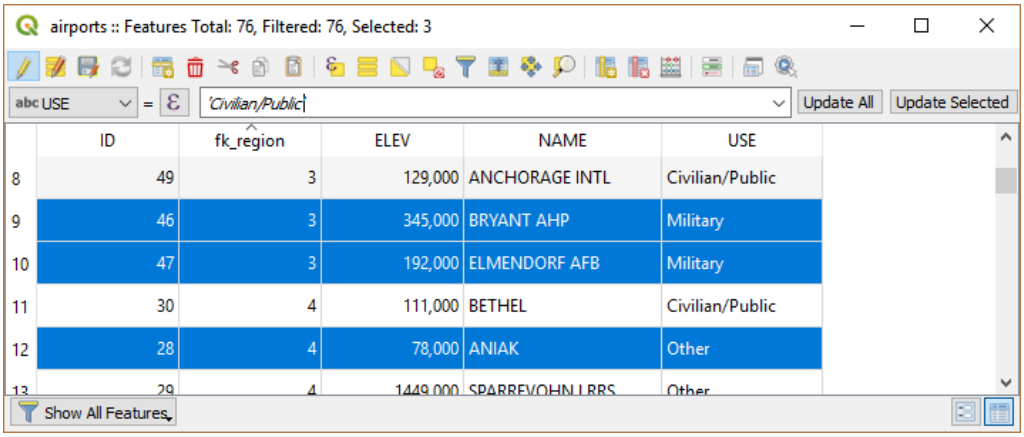


Fig. 16.73: Quick Field Calculation Bar

Editing multiple fields

Unlike the previous tools, multi edit mode allows multiple attributes of different features to be edited simultaneously. When the layer is toggled to edit, multi edit capabilities are accessible:

- using the Toggle multi edit mode button from the toolbar inside the attribute table dialog;
- or selecting *Edit* ► *Modify attributes of selected features* menu.

Notă: Unlike the tool from the attribute table, hitting the *Edit* ► *Modify Attributes of Selected Features* option provides you with a modal dialog to fill attributes changes. Hence, features selection is required before execution.

In order to edit multiple fields in a row:

1. Select the features you want to edit.
2. From the attribute table toolbar, click the button. This will toggle the dialog to its form view. Feature selection could also be made at this step.
3. At the right side of the attribute table, fields (and values) of selected features are shown. New widgets appear next to each field allowing for display of the current multi edit state:

   - The field contains different values for selected features. It's shown empty and each feature will keep its original value. You can reset the value of the field from the drop-down list of the widget.
   - All selected features have the same value for this field and the value displayed in the form will be kept.
   - The field has been edited and the entered value will be applied to all the selected features. A message appears at the top of the dialog, inviting you to either apply or reset your modification.

   Clicking any of these widgets allows you to either set the current value for the field or reset to original value, meaning that you can roll back changes on a field-by-field basis.

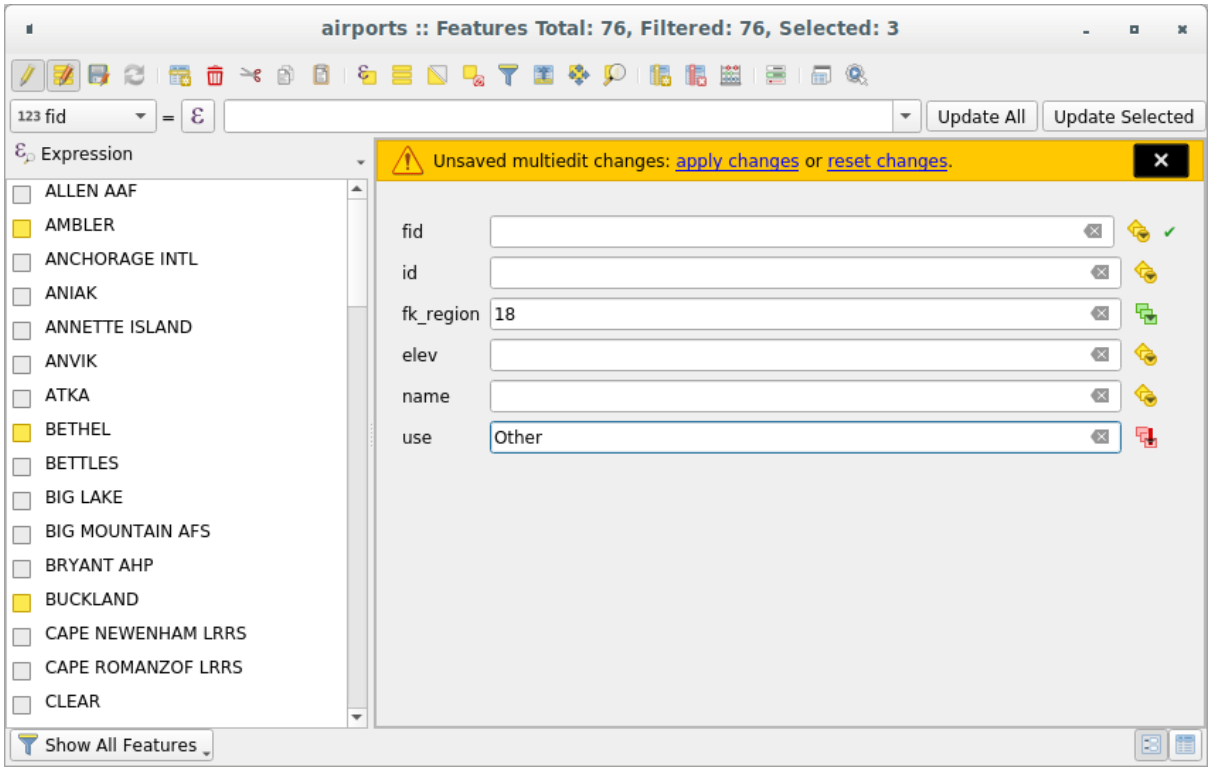


Fig. 16.74: Editing fields of multiple features

4. Make the changes to the fields you want.
5. Click on **Apply changes** in the upper message text or any other feature in the left panel.

Changes will apply to **all selected features**. If no feature is selected, the whole table is updated with your changes. Modifications are made as a single edit command. So pressing Undo will rollback the attribute changes for all selected features at once.

Notă: Multi edit mode is only available for auto generated and drag and drop forms (see *Customizing a form for your data*); it is not supported by custom ui forms.

16.2.6 Exploring features attributes through the Identify Tool

The *Identify features* tool can be used to display all attributes of a feature in the map canvas. It is a quick way to view and verify all data without having to search for it in the attribute table.

To use the *Identify features* tool for vector layers, follow these steps:

1. Select the vector layer in the Layers panel.
2. Click on the *Identify features* tool in the toolbar or press `Ctrl+Shift+I`.
3. Click on a feature in the map view.

The *Identify results* panel will display different features information depending on the layer type. There are two columns in the panel, on the left side you can see *Feature* and on the right side *Value*. Under the *Feature* column, panel will display following information:

- **Derived** section - those are the information calculated or derived from other information in the layer. For example, the area of a polygon or the length of a line. General information that can be found in this section:

  - Depending on the geometry type, cartesian measurements of length, perimeter, or area in the layer's CRS units. For 3D line vectors, the cartesian line length is available.
  - Depending on the geometry type and if an ellipsoid is set in the *Project Properties* dialog (*General –> Measurements*), ellipsoidal values of length, perimeter, or area using the specified units.
  - The count of geometry parts in the feature and the number of the part clicked.
  - The count of vertices in the feature.

  Coordinate information that can be found in this section:

  - X and Y coordinate values of the clicked point.
  - The number of the closest vertex to the clicked point.
  - X and Y coordinate values of the closest vertex.
  - If you click on a curved segment, the radius of that section is also displayed.
- **Data attributes**: This is the list of attribute fields and values for the feature that has been clicked.
- information about the related child feature if you defined a *relation*:
  - the name of the relation
  - the entry in reference field, e.g. the name of the related child feature
  - **Actions**: lists actions defined in the layer's properties dialog (see *Proprietățile Acțiunilor*) and the default action is `View feature form`.
  - **Data attributes**: This is the list of attributes fields and values of the related child feature.

16.2.7 Storing and fetching an external resource

A field may target a resource stored on an external storage system. Attribute forms can be configured so they act as a client to an external storage system in order to store and fetch those resources, on users demand, directly from the forms.

Configuring an external storage

In order to setup an external storage, you have to first configure it from the vector *attribute form properties* and select the *Attachment* widget.

Fig. 16.75: Editing a WebDAV external storage for a given field

From the *Attachment* widget, you have to first select the *Storage type*:

- *Select Existing File*: The target URL already exists. When you select a resource, no store operation is achieved, the attribute is simply updated with the URL.
- *Simple Copy*: Stores a copy of the resource on a file disk destination (which could be a local or network shared file system) and the attribute is updated with the path to the copy.
- *WebDAV Storage*: The resource is pushed to a HTTP server supporting the WebDAV protocol and the attribute is updated with its URL. Nextcloud, Pydio or other file hosting software support this protocol.
- *AWS S3*: The resource is pushed to a server supporting AWS Simple Storage Service protocol and the attribute is updated with its URL. Amazon Web Service and MinIO hosting software support this protocol.

Then, you have to set up the *Store URL* parameter, which provides the URL to be used when a new resource needs to be stored. It's possible to set up an expression using the *data defined override widget* in order to have specific values according to feature attributes.

The variable **@selected_file_path** could be used in that expression and represent the absolute file path of the user selected file (using the file selector or drag'n drop).

Notă: Using the **WebDAV** or **AWS S3** external storage, if the URL ends with a „/", it is considered as a folder and the selected file name will be appended to get the final URL.

If the external storage system needs to, it's possible to configure an *authentication*.

Notă: To use the **AWS S3** external storage, you must use an **AWS S3** authentication type.

Using an external storage

Once configured, you can select a local file using the button … when editing a feature's attribute. Depending on the configured *storage type*, the file will be stored on the external storage system (except if *Select existing file* has been selected) and the field will be updated with the new resource URL.

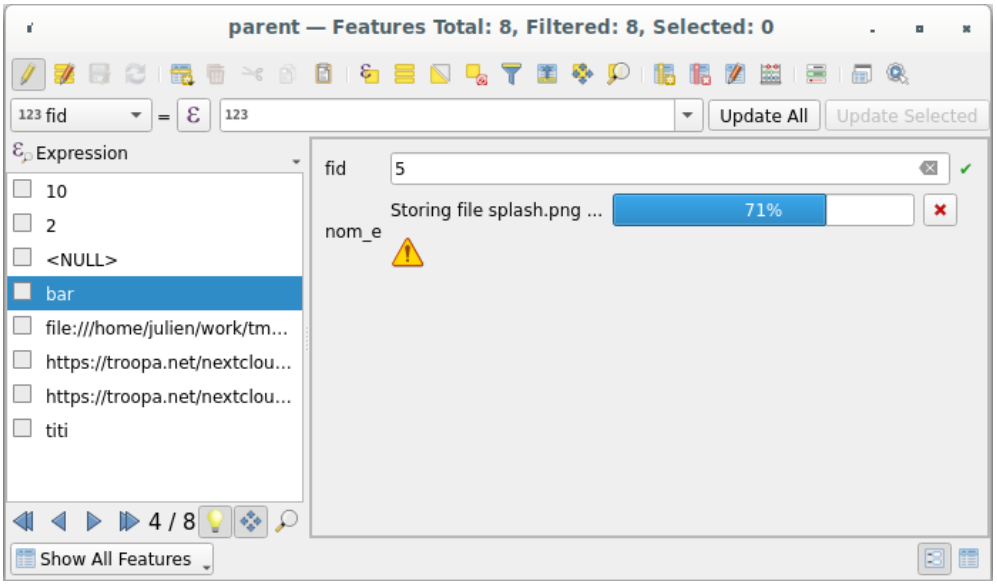


Fig. 16.76: Storing a file to a WebDAV external storage

Notă: User can also achieve the same result if he drags and drops a file on the whole attachment widget.

Use the Cancel button to abort the storing process. It's possible to configure a viewer using the *Integrated document viewer* so the resource will be automatically fetched from the external storage system and displayed directly below the URL. The above icon indicates that the resource cannot be fetched from the external storage system. In that case, more details might appear in the *Log Messages Panel*.

16.3 Editarea

QGIS has various capabilities for editing OGR, SpatiaLite, PostGIS, MS SQL Server and Oracle Spatial vector layers and tables. They can be of 2D or 3D geometry type.

Notă: Procedura pentru editarea straturilor GRASS este diferită - a se vedea secțiunea *Digitizarea și editarea unui strat vectorial GRASS* pentru detalii.

Atenție: Editări Concurente

QGIS does not track if somebody else is editing the same feature at the same time as you are. The last person to save the edits wins.

Sfat: Validating Edits

Continuous validation can be activated on a layer basis in the *Layer Properties* ► *Digitizing* tab. More at *Digitizing Properties*.

16.3.1 Setting the snapping tolerance and search radius

Under the *Settings* ► *Options...* ► *Digitizing* menu, QGIS provides a number of parameters to configure default behaviour of editing tools. More information at *Setări de digitizare*.

For optimal and accurate editing of vector layer geometries, we need to set an appropriate value of snapping tolerance and search radius for features vertices. The *Snapping* group provides related options, namely handling of the snapping tolerance and the search radius.

- Snapping tolerance: When you add a new vertex or move an existing one, the snapping tolerance is the distance QGIS uses to search for the closest vertex or segment you are trying to connect to. If you are not within the snapping tolerance, QGIS will leave the vertex where you release the mouse button, instead of snapping it to an existing vertex or segment.

  The tolerance setting affects all tools that work with snapping and applies by default to new layers and projets. It can however be overridden at layer level (see *Opțiuni de Digitizare și Acroșare*).

- Search radius: *Search radius for vertex edits* is the distance QGIS uses to `search` for the vertex to select when you click on the map. If you are not within the search radius, QGIS will not find and select any vertex for editing.

Snap tolerance and search radius are set in `map units` or `pixels`. You may need to experiment to get them right. If you specify a too big tolerance, QGIS may snap to the wrong vertex, especially if you are dealing with a large number of vertices in close proximity. The smaller the search radius, the more difficult it will be to hit what you want to move.

16.3.2 Opțiuni de Digitizare și Acroșare

Global *snapping and digitizing settings* (snapping mode, tolerance value, and units…) can be overridden in the project from the *Project* ► *Snapping Options...* menu. In the *Snapping and Digitizing Options*, you can also configure some other properties (snapping layers, scale limit, topology…) The *Snapping Toolbar* gives access to most of these features.

By default, snapping is disabled in a project until you press the Enable snapping button or press `S`. The snapping mode, tolerance value, and units can also be configured in this toolbar.

Snapping properties

There are three options to select the layer(s) to snap to:

- *All layers*: quick setting for all visible layers in the project so that the pointer snaps to all vertices and/or segments. In most cases, it is sufficient to use this snapping mode, but beware when using it for projects with many vector layers, as it may affect performance.
- *Current layer*: only the active layer is used, a convenient way to ensure topological consistency within the layer being edited.
- *Advanced Configuration*: allows you to enable and adjust snapping mode, tolerance and units, overlaps and scales of snapping on a layer basis (see Fig. 16.77). If you need to edit a layer and snap its vertices to another, make sure that the target layer is checked and increase the snapping tolerance to a higher value. Snapping will not occur to a layer that is not checked in the snapping options dialog.

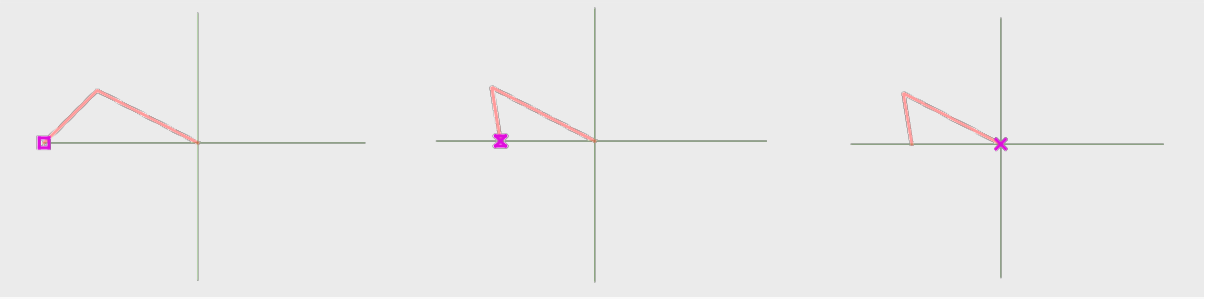
When moving or creating vertex, you can opt for the following snapping modes:

- *Vertex*

- *Segment*: snaps along a line or a polygon perimeter. If topological editing is enabled, then a new vertex is added at the snapping location.
- *Area*: guarantees that the snap point lies anywhere on a polygon's area, not necessarily on its boundary
- *Centroid*: snaps to the centroid of the geometry of a feature. In case of a multipart geometry, the target point may be distinct from the existing parts.
- *Middle of Segments* on line or polygon feature
- *Line Endpoints*: snaps to the first or last vertex of every part of a line or polygon feature.

QGIS will show different *snap* icons depending on the kind of *snap*:

Tabelul 16.2: Snapping icons



Snapping to a vertex: box icon	Snapping to a segment: hourglass icon	Snapping to an intersection: cross icon

Note that it is possible to change the color of these icons in the *Digitizing* part of the global settings.

The tolerance values can be set either in the project's `map units` or in `pixels`. The advantage of choosing `pixels` is that it keeps the snapping constant at different map scales. 10 to 12 pixels is normally a good value, but it depends on the DPI of your screen. Using map units allows the tolerance to be related to real ground distances. For example, if you have a minimum distance between elements, this option can be useful to ensure that you don't add vertices too close to each other.

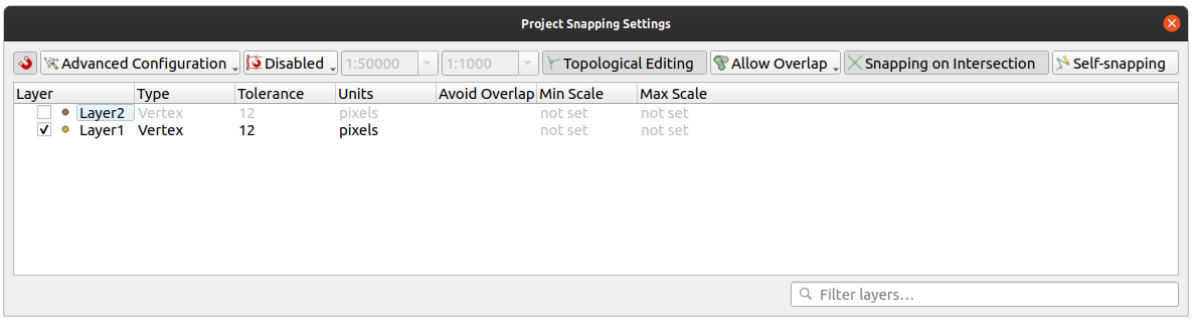


Fig. 16.77: Snapping options (Advanced Configuration mode)

Notă: By default, only visible features (the features whose style is displayed, except for layers where the symbology is „No symbols") can be snapped. You can enable the snapping on invisible features by checking *Enable snapping on invisible features* under the *Settings* ► *Options* ► *Digitizing* tab.

Sfat: Enable snapping by default

You can set snapping to be enabled by default on all new projects in the *Settings* ► *Options* ► *Digitizing* tab. You can also set the default snapping mode, tolerance value, and units, which will populate the *Snapping Options* dialog.

Activarea acroșării la intersecții

Another available option is to use *snapping on intersection*, which allows you to snap to geometry intersections of snapping enabled layers, even if there are no vertices at the intersections.

Limit snapping to a scale range

In some cases snapping can become very slow. This is often caused by the amount of features in some layers that require a heavy index to compute and maintain. Some parameters exist to enable snapping only when the map view is inside a relevant scale range. This allows to only do the costly index computation related to snapping at a scale where drawing is relevant.

Scale limit to snapping is configured in *Project* ► *Snapping Options…*. Limiting snapping to scale is only available in *Advanced Configuration* mode.

To limit snapping to a scale range you have three modes available:

- *Disabled*: Snapping is enabled whatever the current map scale is. This is the default mode.
- *Global*: Snapping is limited and only enabled when the current scale of the map is between a global minimum and a global maximum value. When selecting this mode two widgets become available to configure the range of scales in which snapping is enabled.
- *Per layer*: The snapping scale range limit is defined for each layer. When selecting this mode two columns become available to configure the minimum and maximum scales for each layer.

Please note that the minimum and maximum scales follow the QGIS convention: minimum scale is the most „zoomed out” scale while maximum scale is the most „zoomed in”. A minimum or maximum scale that is set to „0” or „not set” is considered not limiting.

Self-snapping

The Self-snapping option allows you to snap to the geometry that is being edited. Combined with the *advanced digitizing panel*, this provides a handy way to digitize new edges relative to the previous edges or vertices. Self-snapping can cause invalid geometries, use with caution.

Snapping on custom grid

A snapping distance can also be customized on a layer basis in the *Digitizing* tab of the layer properties dialog. With setting the *Geometry precision* distance, you enable a dotted grid visible when the map canvas is at a coherent scale for display. Snapping can then be performed on the dots of the grid: an added or modified geometry will have all of its vertices snapped automatically to the closest node of the grid. More information at *Digitizing Properties*.

16.3.3 Editarea topologică

In addition to these snapping options, the *Snapping options…* dialog (*Project* ► *Snapping options*) and the *Snapping* toolbar allow you to enable / disable some other topological functionalities.

Editarea editării topologice

The Topological editing button helps when editing and maintaining features with common boundaries. With this option enabled, QGIS «detects» shared boundaries. When you move common vertices/segments, QGIS will also move them in the geometries of the neighboring features.

Topological editing works with features from different layers, as long as the layers are visible and in editing mode.

In layer with Z or M values, topological editing will interpolate the Z or M value of the vertex based on the value of the edge used for the connection.

Overlapping control

Overlapping prevents you from drawing new features that overlap existing ones in the selected layer, speeding up digitizing of adjacent polygons. It can be controlled by the overlap tool. Three modes are available:

1. *Allow Overlap* (default)
2. *Avoid Overlap on Active Layer*: prevents any overlap with other features from the layer being edited. Digitize the new geometries so that they overlap their neighbours and QGIS will cut the overlapping part(s) of the new geometries and snap them to the boundary of the existing features. The advantage is that you don't have to digitize the common vertices on boundary.
3. *Follow Advanced Configuration*: allows the overlapping setting to be set on a layer basis in the *Advanced configuration* view mode.

Notă: If the new geometry is totally covered by existing ones, it gets cleared, and QGIS will show an error message.

Atenționare: Use cautiously the *Avoid overlap* **option**

Since this option will cut new overlapping geometries of any polygon layer, you can get unexpected geometries if you forget to uncheck it when no longer needed.

Urmărirea Automată

Usually, when using capturing map tools (add feature, add part, add ring, reshape and split), you need to click each vertex of the feature. With the automatic tracing mode, you can speed up the digitization process as you no longer need to manually place all the vertices during digitization:

1. Enable the Tracing tool (in the *Snapping* toolbar) by pushing the icon or pressing `T` key.
2. *Snap to* a vertex or segment of a feature you want to trace along.
3. Move the mouse over another vertex or segment you'd like to snap and, instead of the usual straight line, the digitizing rubber band represents a path from the last point you snapped to the current position. The tool also works with curved geometries.

   QGIS actually uses the underlying features topology to build the shortest path between the two points. Tracing requires snapping to be activated in traceable layers to build the path. You should also snap to an existing

vertex or segment while digitizing and ensure that the two nodes are topologically connectable through existing features edges, otherwise QGIS is unable to connect them and thus traces a single straight line.

4. Click and QGIS places the intermediate vertices following the displayed path.

Unfold the Enable Tracing icon and set the *Offset* option to digitize a path parallel to the features instead of tracing along them. A positive value shifts the new drawing to the left side of the tracing direction and a negative value does the opposite.

Notă: Adjust map scale or snapping settings for an optimal tracing

If there are too many features in map display, tracing is disabled to avoid potentially long tracing structure preparation and large memory overhead. After zooming in or disabling some layers the tracing is enabled again.

Notă: Does not add topological points

This tool does not add points to existing polygon geometries even if *Topological editing* is enabled. If geometry precision is activated on the edited layer, the resulting geometry might not exactly follow an existing geometry.

Sfat: Quickly enable or disable automatic tracing by pressing the `T` key

By pressing the `T` key, tracing can be enabled/disabled anytime (even while digitizing a feature), so it is possible to digitize parts of the feature with tracing enabled and other parts with tracing disabled. Tools behave as usual when tracing is disabled.

Sfat: Convert tracing to curved geometries

By using *Settings* ► *Options* ► *Digitizing* ► *Tracing* you can create curved geometries while digitizing. See *digitizing options*.

16.3.4 Digitizarea unui strat vectorial existent

By default, QGIS loads layers read-only. This is a safeguard to avoid accidentally editing a layer if there is a slip of the mouse. However, you can choose to edit any layer as long as the data provider supports it (see *Explorarea Formatelor de Date și a Câmpurilor*), and the underlying data source is writable (i.e., its files are not read-only).

Sfat: Restrict edit permission on layers within a project

From the *Project* ► *Properties…* ► *Data Sources* ► *Layers Capabilities* table, you can choose to set any layer read-only regardless the provider permission. This can be a handy way, in a multi-users environment to avoid unauthorized users to mistakenly edit layers (e.g., Shapefile), hence potentially corrupt data. Note that this setting only applies inside the current project.

In general, tools for editing vector layers are divided into a digitizing and an advanced digitizing toolbar, described in section *Digitizare avansată*. You can select and unselect both under *View* ► *Toolbars* ►.

Using the basic digitizing tools, you can perform the following functions:

Tabelul 16.3: Vector layer basic editing toolbar

Instrument	Scop	Instrum	Scop
	Access to save, rollback or cancel changes in all or selected layers simultaneously		Turn on or off edit status of selected layer(s) based on the active layer status
	Save edits to the active layer		
	Digitize using straight segments		Digitize using curve lines
	Enable freehand digitizing		Digitize polygon of regular shape
	Add new record		Adăugare Entități: Captura unui Punct
	Adăugare Entități: Captura unei Linii		Adăugare Entități: Captura unui Poligon
	Instrumentul Vertex (Toate Straturile)		Instrumentul Vertex (Stratul Curent)
Show Vertex Editor	Set whether the vertex editor panel should auto-open		Modify the attributes of all selected features simultaneously
	Delete Selected features from the active layer		Cut Features from the active layer
	Copy selected Features from the active layer		Paste Features into the active layer
	Undo changes in the active layer		Redo changes in active layer

Note that while using any of the digitizing tools, you can still *zoom or pan* in the map canvas without losing the focus on the tool.

All editing sessions start by choosing the Toggle editing option found in the context menu of a given layer, from the attribute table dialog, the digitizing toolbar or the *Layer* menu.

Once the layer is in edit mode, additional tool buttons on the editing toolbar will become available and markers will appear at the vertices of all features unless *Show markers only for selected features* option under *Settings* ► *Options…* ► *Digitizing* menu is checked.

Sfat: Salvați cu Regularitate

Amintiți-vă să Salvați Modificările Stratului cu regularitate. Aceasta va confirma, de asemenea, că sursa de date poate accepta modificările.

Geometry editing techniques

When a geometry drawing tool (mainly the ones that add, split, reshape features) is enabled for a line or polygon based layer, you can select the technique for adding new vertices:

- The Digitize with Segment: draws straight segment whose start and end points are defined by left clicks.
- The Digitize with Curve: draws curve line based on three consecutive nodes defined by left clicks (start, point along the arc, end). If the geometry type does not support curves, then consecutive smaller segments are used to approximate the curvature.
- The Stream Digitizing: draws lines in freehand mode, i.e. nodes are added following cursor movement in the map canvas and a *Streaming Tolerance*. The streaming tolerance defines the spacing between consecutive vertices. Currently, the only supported unit is pixels (`px`). Only the starting left click and the ending right click are necessary in this mode.

- Digitize Shape: acționează instrumentele de pe *Bara de Digitizare a Formelor* în scopul desenării unui poligon cu o formă regulată.

The selected technique remains while switching among the digitizing tools. You can combine any of the first three methods while drawing the same geometry.

Adăugare Entități

Depending on the layer type, you can use the Add Record, Add Point Feature, Add Line Feature or Add Polygon Feature icons on the toolbar to add new features into the current layer.

To add a geometryless feature, click on the Add Record button and you can enter attributes in the feature form that opens.

To create features with the spatially enabled tools, you first digitize the geometry then enter its attributes. To digitize the geometry:

1. (Optional as it is the default) Select the Digitize With Segment geometry drawing method
2. Left-click on the map area to create the first point of your new feature. For point features, this should be enough and trigger, if required, the feature form to fill in their attributes.
3. For line or polygon geometries, keep on left-clicking for each additional point you wish to capture. You can rely on the *snapping to features* options, the *snap-to-grid* or the *advanced digitizing* panel to accurately position each vertex.

   Along with drawing straight segments between nodes you click one by one, lines and polygons can be:

   - *traced automatically*, accelerating the digitization. This will create consecutive straight lines between the vertices you place, following existing features.
   - free-hand digitized, pressing `R` or activating Stream Digitizing.
   - drawn as curve, pressing `Ctrl+Shift+G` or activating Digitize with Curve.

   Notă: While digitizing line or polygon geometries, you can switch back and forth between the geometry drawing methods, allowing you to create features mixing straight segments, free-hand ones and curved parts.

   ---

4. Press `Delete` or `Backspace` key to revert the last node(s) you may wrongly add.
5. When you have finished adding points, right-click anywhere on the map area to confirm you have finished entering the geometry of that feature.

   Sfat: Customize the digitizing rubber band

   While capturing polygon, the by-default red rubber band can hide underlying features or places you'd like to capture a point. This can be fixed by setting a lower opacity (or alpha channel) to the rubber band's *Fill Color* in *Settings* ► *Options* ► *Digitizing* menu. You can also avoid the use of the rubber band by checking *Don't update rubber band during node editing*.

   ---

6. For line feature pressing `Shift` + right-click will close the line automatically.
7. The attribute window will appear, allowing you to enter the information for the new feature. Fig. 16.78 shows setting attributes for a fictitious new river. However, in the *Digitizing* menu under the *Settings* ► *Options* menu, you can also:

   - *Suppress attributes pop-up windows after each created feature* to avoid the form opening;

- or *Reuse last entered attribute values* to have fields automatically filled at the opening of the form and just have to type changing values.

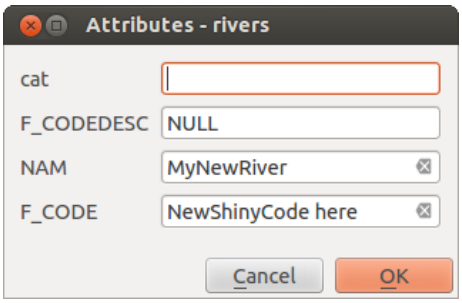


Fig. 16.78: Completați Dialogul Valorilor pentru Atribute, după digitizarea unei noi entități vectoriale

Instrumentul vertex

QGIS provides two tools to interact with vector features vertices:

- Vertex Tool (Current Layer): only overlaid features in the active layer (in the *Layers* panel) are affected
- Vertex Tool (All Layers): any overlaid features in all editable layers are affected. This allows you to edit features without switching the active layer or edit multiple layers at once (e.g., country and their regions boundaries)

For any editable vector layer, the vertex tools provide manipulation capabilities of feature vertices similar to CAD programs. It is possible to select multiple vertices at once and to move, add or delete them altogether. The vertex tools also support the topological editing feature. They are selection persistent, so when some operation is done, selection stays active for this feature and tool.

It is important to set the property *Settings* ► *Options* ► *Digitizing* ► *Search Radius:* 1,00 to a number greater than zero. Otherwise, QGIS will not be able to tell which vertex is being edited and will display a warning.

Sfat: Simbolurile Vertexului

QGIS supports different kinds of vertex markers: «Semi-transparent circle», «Cross» and «None». To change the marker style, choose *Options* from the *Settings* menu, click on the *Digitizing* tab and select the appropriate entry.

Operaţiuni de bază

Given a layer in edit mode, start by activating the vertex tool. Red circles will appear when hovering vertices.

- **Selecting vertices**: You can select vertices by:
  - Clicking on them one at a time holding `Shift` key pressed
  - Click-and-dragging a rectangle surrounding the target vertices
  - Drawing a polygon surrounding the target vertices: Hold `Alt` and click using the vertex tool to start digitizing a polygon. Each subsequent click adds a new vertex to the rubberband polygon. `Backspace` or `Delete` removes last added rubberband vertex. `Esc` cancels the polygon selection mode, as also does backspacing/deleting all of the rubberband's vertices. Right click finalizes the polygon digitizing and selects all vertices within the rubberband polygon.

  When a vertex is selected, its color changes to blue. To add more vertices to the current selection, hold down the `Shift` key while proceeding as above. To remove vertices from the selection, hold down `Ctrl`.

Sfat: Feature selection bounds vertex tool

Vertices can be selected accross different features (or layers). If you are looking for vertices of a specific feature in a crowded place, first select that feature. Then draw the rectangle or polygon selector with the vertex tool around the vertices: only the selected feature's vertices are selected.

This is also the case if you display the feature in the *vertex editor* panel.

- **Batch vertex selection mode**: The batch selection mode can be activated by pressing `Shift+R`. Select a first node with one single click, and then hover **without clicking** another vertex. This will dynamically select all the nodes in between using the shortest path (for polygons).

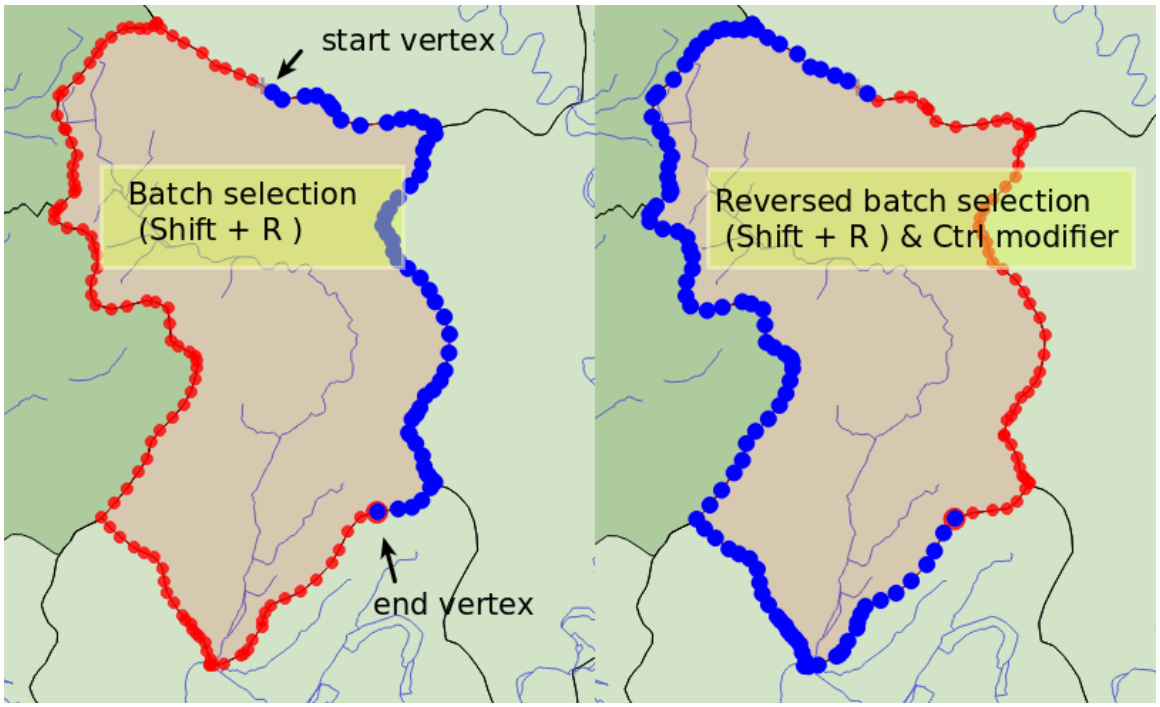


Fig. 16.79: Batch vertex selection using `Shift+R`

Press `Ctrl` will invert the selection, selecting the longest path along the feature boundary. Ending your node selection with a second click, or pressing `Esc` will escape the batch mode.

- **Adding vertices**: To add a vertex to a line or polygon geometry, hold `Shift` and double-click the place on the segment.

  When hovering a segment, a virtual new node appears on the center. Click on it, move the cursor to a new location and click again to add a new vertex. For lines, a virtual node is also proposed at both extremities: click on it, do subsequent clicks and finish with a right-click; this allows to easily extend an existing line.

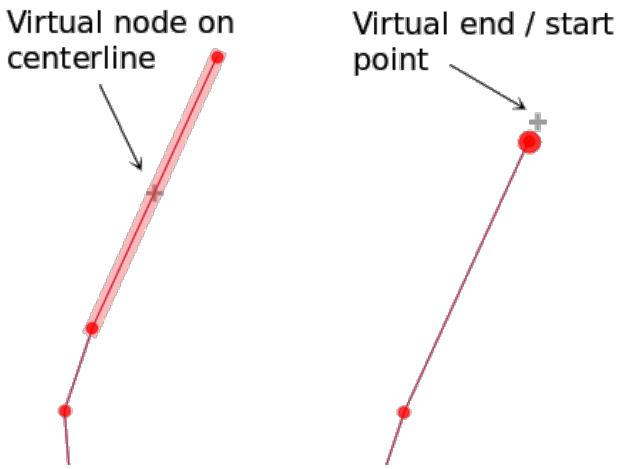


Fig. 16.80: Virtual nodes for adding vertices

- **Deleting vertices**: Select the vertices and click the `Delete` key. Deleting all the vertices of a feature generates, if compatible with the datasource, a geometryless feature. Note that this doesn't delete the complete feature, just the geometry part. To delete a complete feature use the Delete Selected tool.
- **Moving vertices**: Select all the vertices you want to move, click on a selected vertex or edge, and click on the desired new location. You can use the *snapping to feature capabilities* and the *Advanced Digitizing Panel* constraints for distance, angles, exact X and Y location before the second click. All the selected vertices will be translated.

  However, if the *snap-to-grid* option is enabled, selected vertices are snapped to the closest grid intersection to their translated position. Unselected vertices are also moved to their closest grid intersection. There is no simple translation.

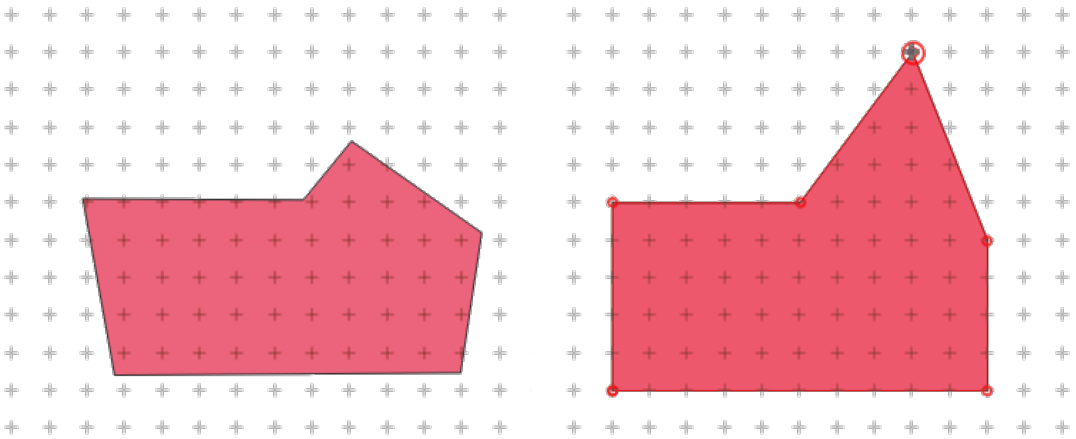


Fig. 16.81: Moving the top vertex snaps all the vertices to the grid

- **Converting adjacent segments to/from curve**: Select the center vertex of the segment you want to convert, hit the `O` letter key. If the vertex was in a curve, the curve is converted into straight lines. If the vertex was between two straight lines, they are converted into a curve. A first or a last vertex of a line can't be converted to a center vertex curve. The layer must be compatible with curve geometry type.

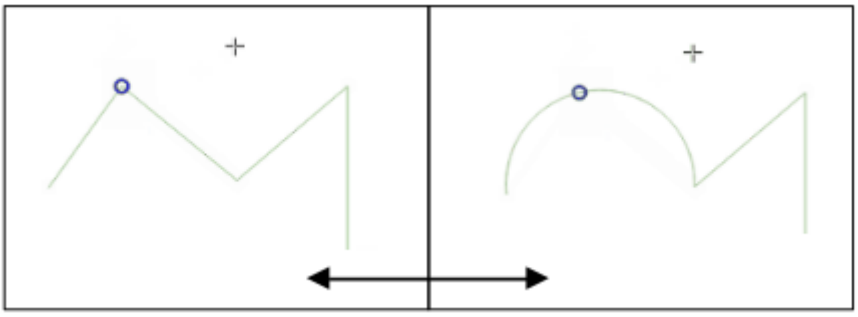


Fig. 16.82: Switch from curve to straight lines with O letter

Each change made with the vertex tool is stored as a separate entry in the *Undo* dialog. Remember that all operations support topological editing when this is turned on. On-the-fly projection is also supported.

The Vertex Editor Panel

With enabling a vertex tool, you also open the *Vertex Editor* panel. Right-clicking over a feature fills the panel with the list of all the vertices of the feature with their *x*, *y* (*z*, *m* if applicable) coordinates and *r* (for the radius, in case of circular geometry). The feature is also made exclusive for editing, meaning that the edit of any other features is disabled:

- Selecting a row in the table does select the corresponding vertex in the map canvas, and vice versa.
- Clicking or dragging over the map canvas will only select or move vertices and segments of that feature
- Change a coordinate in the table and the vertex position is updated. This is a convenient way to edit Z coordinate or M value on vertices.
- You can also select multiple rows and delete them altogether.
- New vertices can only be added to the bound feature

If you do not want the *Vertex Editor* panel to immediately show each time you interact with vertex tools (and potentially hide other panels or disturb panels placement), uncheck the *Auto-open table* entry in the ☰ Options menu at the top of the panel. You can then also close the panel. To reopen the panel, you would need to right-click over a panel or toolbar and select it in the list or tick the *Show vertex editor* entry in the *Digitizing toolbar*.

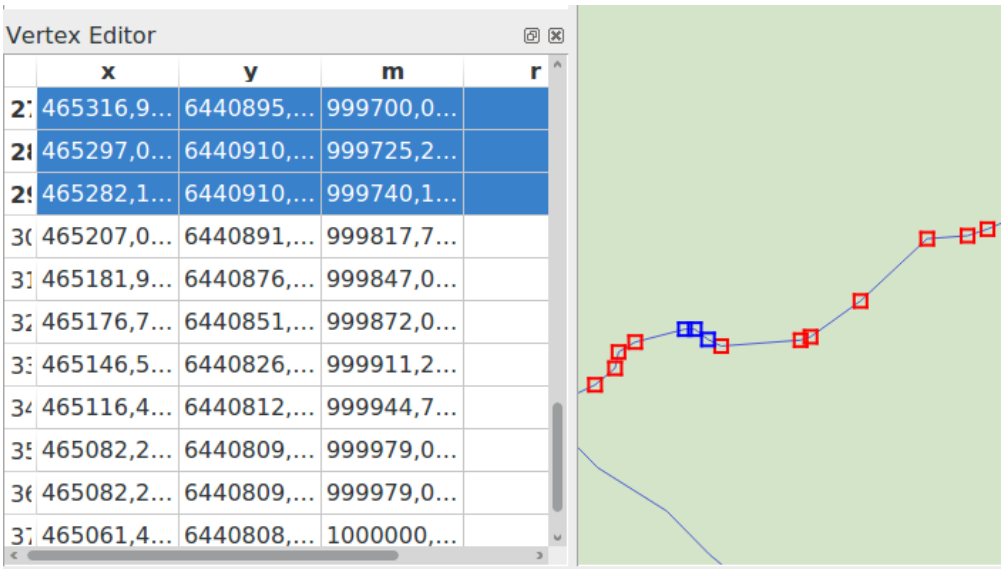


Fig. 16.83: Vertex editor panel showing selected nodes

Rules of Z coordinate or M value assignment

Digitizing 3D vector features or features with M value is not that different from (X,Y) 2D layers». Tools and options described in this chapter are still available and help you place the vertex or point in a planar environment. Then you may need to handle the Z coordinate (or M value) assignment:

- By default, QGIS will assign to new vertices the *Default Z value* (respectively *Default M value*) set in the *Settings* ► *Options* ► *Digitizing* tab. If the *Advanced Digitizing Panel* is in use, then the value is taken from its *z* (respectively *m*) widget.
- When snapping to a vertex, the new or moved vertex takes the snapped one's Z or M value.
- When snapping to a segment while the topological editing is on, then the new vertex Z or M value is interpolated along the segment.
- If the *z* (respectively *m*) widget of the *Advanced Digitizing Panel* is locked, then its value is applied to the vertex, taking precedence over any snapped vertex or segment Z or M value.

To edit Z or M values of an existing feature, you can use the *Vertex editor panel*. To create features with custom Z or M values you may want to rely on the *Advanced Digitizing Panel*.

Tăiere, Copiere și Lipire Entități

Selected features can be cut, copied and pasted between layers in the same QGIS project, as long as destination layers are set to Toggle editing beforehand.

Sfat: Transform polygon into line and vice-versa using copy/paste

Copy a line feature and paste it in a polygon layer: QGIS pastes in the target layer a polygon whose boundary corresponds to the closed geometry of the line feature. This is a quick way to generate different geometries of the same data.

Features can also be pasted to external applications as text. That is, the features are represented in CSV format, with the geometry data appearing in the OGC Well-Known Text (WKT) format. WKT and GeoJSON features from outside QGIS can also be pasted to a layer within QGIS.

When would the copy and paste function come in handy? Well, it turns out that you can edit more than one layer at a time and copy/paste features between layers. Why would we want to do this? Say we need to do some work on a new layer but only need one or two lakes, not the 5,000 on our `big_lakes` layer. We can create a new layer and use copy/paste to plop the needed lakes into it.

Ca un exemplu, vom copia unele lacuri într-un nou strat:

1. Încărcați stratul din care doriți să copiați (stratul sursă)
2. Încărcați sau creați stratul în care doriți să copiați (stratul țintă)
3. Începeți editarea stratului țintă
4. Activați stratul sursă, făcând clic pe el în legendă
5. Folosiți instrumentul de Selectare a Entităților după arie sau cu un singur clic pentru a selecta entit(ățile) din stratul sursă
6. Clic pe instrumentul de Copiere Entități
7. Activați stratul destinație, făcând clic pe el în legendă
8. Clic pe instrumentul de Lipire Entități
9. Dezactivați editarea și salvați modificările.

What happens if the source and target layers have different schemas (field names and types are not the same)? QGIS populates what matches and ignores the rest. If you don't care about the attributes being copied to the target layer, it doesn't matter how you design the fields and data types. If you want to make sure everything - the feature and its attributes - gets copied, make sure the schemas match.

Notă: Congruența Entităților Inserate

If your source and destination layers use the same projection, then the pasted features will have geometry identical to the source layer. However, if the destination layer is a different projection, then QGIS cannot guarantee the geometry is identical. This is simply because there are small rounding-off errors involved when converting between projections.

Sfat: Copie un atribut de tip șir în altul**

If you have created a new column in your attribute table with type «string» and want to paste values from another attribute column that has a greater length the length of the column size will be extended to the same amount. This is because the GDAL Shapefile driver knows to auto-extend string and integer fields to dynamically accommodate for the length of the data to be inserted.

ţtergerea Entităților Selectate

If we want to delete an entire feature (attribute and geometry), we can do that by first selecting the geometry using the regular Select Features by area or single click tool. Selection can also be done from the attribute table. Once you have the selection set, press `Delete` or `Backspace` key or use the Delete Selected tool to delete the features. Multiple selected features can be deleted at once.

The Cut Features tool on the digitizing toolbar can also be used to delete features. This effectively deletes the feature but also places it on a „spatial clipboard". So, we cut the feature to delete. We could then use the Paste Features tool to put it back, giving us a one-level undo capability. Cut, copy, and paste work on the currently selected features, meaning we can operate on more than one at a time.

Anulare/Restabilire

The Undo and Redo tools allows you to undo or redo vector editing operations. There is also a dockable widget, which shows all operations in the undo/redo history (see Fig. 16.84). This widget is not displayed by default; it can be displayed by right-clicking on the toolbar and activating the *Undo/Redo Panel* checkbox. The Undo/Redo capability is however active, even if the widget is not displayed.

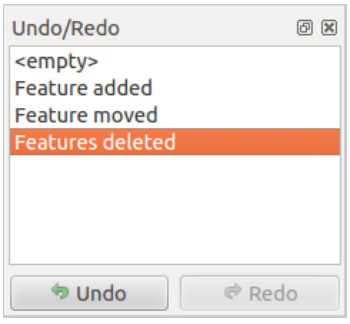


Fig. 16.84: Pașii de Anulare/Restabilire a Digitizării

When Undo is hit or `Ctrl+Z` (or `Cmd+Z`) pressed, the state of all features and attributes are reverted to the state before the reverted operation happened. Changes other than normal vector editing operations (for example, changes

done by a plugin) may or may not be reverted, depending on how the changes were performed.

To use the undo/redo history widget, simply click to select an operation in the history list. All features will be reverted to the state they were in after the selected operation.

Salvarea Straturilor Modificate

When a layer is in editing mode, any changes remain in the memory of QGIS. Therefore, they are not committed/saved immediately to the data source or disk. If you want to save edits to the current layer but want to continue editing without leaving the editing mode, you can click the Save Layer Edits button. When you turn editing mode off with Toggle editing (or quit QGIS for that matter), you are also asked if you want to save your changes or discard them.

If the changes cannot be saved (e.g., disk full, or the attributes have values that are out of range), the QGIS in-memory state is preserved. This allows you to adjust your edits and try again.

Sfat: Integritatea Datelor

It is always a good idea to back up your data source before you start editing. While the authors of QGIS have made every effort to preserve the integrity of your data, we offer no warranty in this regard.

Saving multiple layers at once

This feature allows the digitization of multiple layers. Choose *Save for Selected Layers* to save all changes you made in multiple layers. You also have the opportunity to *Rollback for Selected Layers*, so that the digitization may be withdrawn for all selected layers. If you want to stop editing the selected layers, *Cancel for Selected Layer(s)* is an easy way.

Aceleași funcții sunt disponibile pentru editarea tuturor straturilor proiectului.

Sfat: Use transaction group to edit, save or rollback multiple layers changes at once

When working with layers from the same PostGreSQL database, activate the *Automatically create transaction groups where possible* option in *Project* ► *Properties…* ► *Data Sources* to sync their behavior (enter or exit the edit mode, save or rollback changes at the same time).

16.3.5 Digitizare avansată

Tabelul 16.4: Vector layer advanced editing toolbar

Pictogramă	Scop	Pictogramă	Scop
	Activarea Instrumentelor de Digitizare Avansată		
	Deplasare Enti(tăți)		Copiere și Deplasare Entit(ăți)
	Rotire Enti(tăți)		Simplificare Entitate
	Scalare Entități		
	Adăugare Inel		Adăugare Parte
	Umplere Inel		Swap direction
	Ştergere Inel		Ştergere Parte
	Curba de Compensare		Remodelare Entități
	Divizare Părți		Divizare Entități
	Unificare Atribute pentru Entitățile Selectate		Unificare Entități Selectate
	Rotiți Simbolurile Punctelor		Decalarea Simbolurilor Punctelor
	Trim or Extend Feature		

Deplasare Enti(tăți)

Instrumentul Deplasare Entit(ăți) permite deplasarea entităților existente:

1. Selectați entit(ățile) de ´deplasat.
2. Click on the map canvas to indicate the origin point of the displacement; you can rely on snapping capabilities to select an accurate point.

   You can also take advantages of the *advanced digitizing constraints* to accurately set the origin point coordinates. In that case:

   1. First click on the button to enable the panel.
   2. Type `x` and enter the corresponding value for the origin point you'd like to use. Then press the button next to the option to lock the value.
   3. Do the same for the `y` coordinate.
   4. Click on the map canvas and your origin point is placed at the indicated coordinates.
3. Move over the map canvas to indicate the destination point of the displacement, still using snapping mode or, as above, use the advanced digitizing panel which would provide complementary `distance` and `angle` placement constraints to place the end point of the translation.
4. Click on the map canvas: the whole features are moved to new location.

De asemenea, puteți crea o copie deplasată a entit(ăților) utilizând Copiere și Mutare Entit(ăți).

Notă: Dacă nu este selectată nicio entitate, atunci când faceți clic pentru prima dată pe canevasul hărții cu oricare dintre instrumentele de *Deplasare Entit(ăți)* sau de *Copiere și Deplasare Entit(ăți)*, doar entitatea de sub mouse este afectate de acțiune. Deci, dacă doriți să deplasați mai multe entități, acestea ar trebui să fie, mai întâi, selectate .

Rotire Enti(tăți)

Use the Rotate Feature(s) tool to rotate one or multiple features in the map canvas:

1. Press the Rotate Feature(s) icon
2. Then click on the feature to rotate. The feature's centroid is referenced as rotation center, a preview of the rotated feature is displayed and a widget opens showing the current *Rotation* angle.
3. Click on the map canvas when you are satisfied with the new placement or manually enter the rotation angle in the text box. You can also use the *Snap to* ° box to constrain the rotation values.
4. If you want to rotate several features at once, they shall be selected first, and the rotation is by default around the centroid of their combined geometries.

You can also use an anchor point different from the default feature centroid: press the `Ctrl` button, click on the map canvas and that point will be used as the new rotation center.

If you hold `Shift` before clicking on the map, the rotation will be done in 45 degree steps, which can be modified afterwards in the user input widget.

To abort feature rotation, press the `ESC` button or click on the Rotate Feature(s) icon.

Scalare Entități

The Scale Feature tool is similar to the Rotate feature. Though instead of performing a rotation of selected features, it rescales their geometry. The change is performed in relation to the anchor point and the scale ratio can be manually specified in the widget that appears in the upper corner of the canvas.

Simplificare Entitate

The Simplify Feature tool allows you to interactively reshape a line or polygon geometry by reducing or densifying the number of vertices, as long as the geometry remains valid:

1. Select the Simplify Feature tool.
2. Click on the feature or drag a rectangle over the features.
3. A dialog pops up allowing you to define the *Method* to apply, ie whether you would like to:
   - *simplify the geometry*, meaning less vertices than the original. Available methods are `Simplify by distance`, `Simplify by snapping to grid` or `simplify by area (Visvalingam)`. You'd then need to indicate the value of *Tolerance* in `Layer units`, `Pixels` or `map units` to use for simplification. The higher the tolerance is the more vertices can be deleted.
   - or *densify the geometries* with new vertices thanks to the `Smooth` option: for each existing vertex, two vertices are placed on each of the segments originated from it, at an *Offset* distance representing the percentage of the segment length. You can also set the number of *Iterations* the placement would be processed: the more iterations, the more vertices and smoother is the feature.

   Settings that you used will be saved when leaving a project or an edit session. So you can go back to the same parameters the next time you simplify a feature.

4. A summary of the modifications that would apply is shown at the bottom of the dialog, listing number of features and number of vertices (before and after the operation and the ratio the change represents). Also, in the map canvas, the expected geometry is displayed over the existing one, using the rubberband color.
5. When the expected geometry fits your needs, click *OK* to apply the modification. Otherwise, to abort the operation, you can either press *Cancel* or right-click in the map canvas.

Notă: Unlike the feature simplification option in *Settings* ► *Options* ► *Rendering* menu which simplifies the geometry just for rendering, the Simplify Feature tool permanently modifies feature's geometry in data source.

Adăugare Parte

You can Add Part to a selected feature generating a multipoint, multiline or multipolygon feature. The new part must be digitized outside the existing one which should be selected beforehand.

The Add Part can also be used to add a geometry to a geometryless feature. First, select the feature in the attribute table and digitize the new geometry with the Add Part tool.

Ştergere Parte

The Delete Part tool allows you to delete parts from multifeatures (e.g., to delete polygons from a multi-polygon feature). This tool works with all multi-part geometries: point, line and polygon. Furthermore, it can be used to totally remove the geometric component of a feature. To delete a part, simply click within the target part.

Adăugare Inel

You can create ring polygons using the Add Ring icon in the toolbar. This means that inside an existing area, it is possible to digitize further polygons that will occur as a «hole», so only the area between the boundaries of the outer and inner polygons remains as a ring polygon.

Umplere Inel

The Fill Ring tool helps you create polygon feature that totally falls within another one without any overlapping area; that is the new feature covers a hole within the existing one. To create such a feature:

1. Select the Fill Ring tool.
2. Draw a new polygon over the existing feature: QGIS adds a ring to its geometry (like if you used the Add Ring tool) and creates a new feature whose geometry matches the ring (like if you *traced* over the interior boundaries with the Add polygon feature tool).
3. Or alternatively, if the ring already exists on the feature, place the mouse over the ring and left-click while pressing `Shift`: a new feature filling the hole is drawn at that place.

   The *Feature Attributes* form of the new feature opens, pre-filled with values of the „parent" feature and/or *fields constraints*.

Ştergere Inel

The Delete Ring tool allows you to delete rings within an existing polygon, by clicking inside the hole. This tool only works with polygon and multi-polygon features. It doesn't change anything when it is used on the outer ring of the polygon.

Remodelare Entități

You can reshape line and polygon features using the Reshape Features tool on the toolbar. For lines, it replaces the line part from the first to the last intersection with the original line.

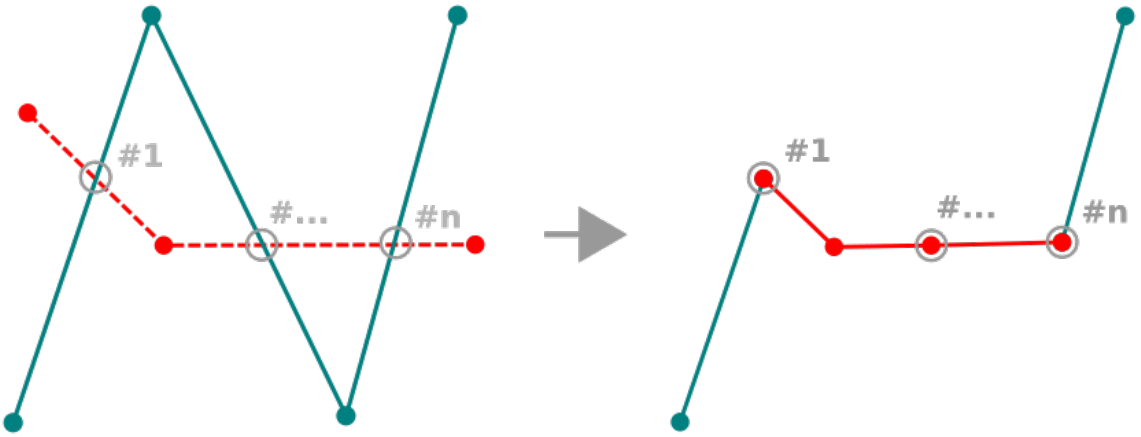


Fig. 16.85: Reshape line

Sfat: Extend linestring geometries with reshape tool

Use the Reshape Features tool to extend existing linestring geometries: snap to the first or last vertex of the line and draw a new one. Validate and the feature's geometry becomes the combination of the two lines.

For polygons, it will reshape the polygon's boundary. For it to work, the reshape tool's line must cross the polygon's boundary at least twice. To draw the line, click on the map canvas to add vertexes. To finish it, just right-click. Like with the lines, only the segment between the first and the last intersections is considered. The reshape line's segments that are inside the polygon will result in cropping it, where the ones outside the polygon will extend it.

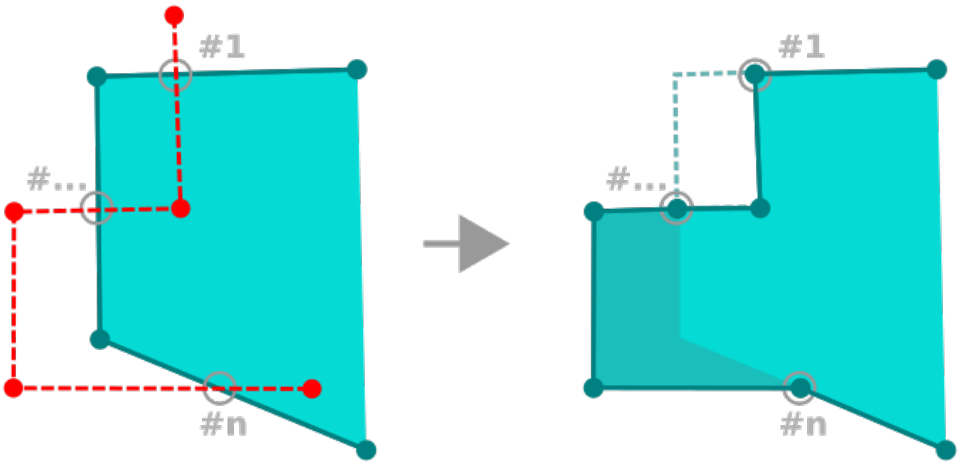


Fig. 16.86: Reshape polygon

With polygons, reshaping can sometimes lead to unintended results. It is mainly useful to replace smaller parts of a polygon, not for major overhauls, and the reshape line is not allowed to cross several polygon rings, as this would generate an invalid polygon.

Notă: The reshape tool may alter the starting position of a polygon ring or a closed line. So, the point that is represented «twice» will not be the same any more. This may not be a problem for most applications, but it is something to consider.

Curbe de Compensare

The Offset Curve tool creates parallel shifts of line layers. The tool can be applied to the edited layer (the geometries are modified) or also to background layers (in which case it creates copies of the lines / rings and adds them to the edited layer). It is thus ideally suited for the creation of distance line layers. The *User Input* dialog pops-up, showing the displacement distance.

To create a shift of a line layer, you must first go into editing mode and activate the Offset Curve tool. Then click on a feature to shift it. Move the mouse and click where wanted or enter the desired distance in the user input widget. Holding `Ctrl` during the 2nd click will make an offset copy. Your changes may then be saved with the Save Layer Edits tool.

QGIS options dialog (Digitizing tab then **Curve offset tools** section) allows you to configure some parameters like **Join style**, **Quadrant segments**, **Miter limit**.

Inversare Linie

Changing the direction of a line geometry can be useful for cartographical purposes or when preparing for network analysis.

To change a line direction:

1. Activate the reverse line tool by clicking Reverse line.
2. Click on the line. The direction of the line is reversed.

Divizare Entități

Use the Split Features tool to split a feature into two or more new and independent features, ie. each geometry corresponding to a new row in the attribute table.

To split line or polygon features:

1. Select the Split Features tool.
2. Draw a line across the feature(s) you want to split. If a selection is active, only selected features are split. Fields of resulting features are filled according to their *splitting policy*.
3. You can then as usual modify any of the attributes of any resulting feature.

Sfat: Split a polyline into new features in one-click

Using the Split Features tool, snap and click on an existing vertex of a polyline feature to split that feature into two new features.

Divizare părți

In QGIS it is possible to split the parts of a multi part feature so that the number of parts is increased. Just draw a line across the part you want to split using the Split Parts icon.

Sfat: Split a polyline into new parts in one-click

Using the Split Parts tool, snap and click on an existing vertex of a polyline feature to split the feature into two new polylines belonging to the same feature.

Unificați entitățile selectate

The Merge Selected Features tool allows you to create a new feature by merging existing ones: their geometries are merged to generate a new one. If features don't have common boundaries, a multipolygon/multipolyline/multipoint feature is created.

1. First, select the features you'd like to combine.
2. Then press the Merge Selected Features button.
3. In the new dialog, the *Merge* line at the bottom of the table shows the attributes of the resulting feature. You can alter any of these values either by:
   - manually replacing the value in the corresponding cell;
   - selecting a row in the table and pressing *Take attributes from selected feature* to use the values of this initial feature;
   - pressing the *Take attributes from the largest geometry* to use the attributes from the longest line feature, the largest polygon, or the multipoints with the most parts;
   - pressing *Skip all fields* to use empty attributes;
   - expanding the drop down menu at the top of the table, select any of the above options to apply to the corresponding field only. There, you can also choose to aggregate the initial features attributes (Minimum, Maximum, Median, Sum, Count, Concatenation… depending on the type of the field. see *Statistical Summary Panel* for the full list of functions).

Notă: If the layer has default values or clauses present on fields, these are used as the initial value for the merged feature.

4. Press *OK* to apply the modifications. A single (multi)feature is created in the layer, replacing the previously selected ones.

Unificați atributele pentru entitățile selectate

The Merge Attributes of Selected Features tool allows you to apply same attributes to features without merging their boundaries. The dialog is the same as the `Merge Selected Features` tool's except that unlike that tool, selected objects are kept with their geometry while some of their attributes are made identical.

Rotiți Simbolurile Punctelor

The Rotate Point Symbols allows you to individually change the rotation of point symbols in the map canvas.

1. First, you need to indicate the field to store the rotation value in. This is made by assigning a field to the symbol *data-defined* rotation property:
   1. In the *Layer Properties* ► *Symbology* dialog, browse to the symbol editor dialog.
   2. Click the *Data-defined override* widget near the *Rotation* option of the top *Marker* level (preferably) of the symbol layers.
   3. Choose a field in the *Field Type* combobox. Values of this field are hence used to rotate each feature's symbol accordingly.

      You can also check the Store data in project entry to generate an *auxiliary data storage* field to control the rotation value.

Notă: Make sure that the same field is assigned to all the symbol layers

Setting the data-defined rotation field at the topmost level of the symbol tree automatically propagates it to all the symbol layers, a prerequisite to perform graphical symbol rotation with the *Rotate Point Symbols* tool. Indeed, if a symbol layer does not have the same field attached to its rotation property, the tool will not work.

Fig. 16.87: Rotating a point symbol

2. Then click on a point symbol in the map canvas with the Rotate Point Symbols tool
3. Move the mouse around. A red arrow with the rotation value will be visualized (see Fig. 16.87). If you hold the `Ctrl` key while moving, the rotation will be done in 15 degree steps.

4. When you get the expected angle value, click again. The symbol is rendered with this new rotation and the associated field is updated accordingly.

   You can right-click to abort symbol rotation.

Decalarea Simbolurilor Punctelor

The Offset Point Symbols allows you to interactively change the rendered position of point symbols in the map canvas. This tool behaves like the Rotate Point Symbols tool except that it requires you to connect a field to the data-defined *Offset (X,Y)* property of each layer of the symbol. The field will then be populated with the offset coordinates for the features whose symbol is moved in the map canvas.

1. Associate a field to the data-defined widget of the *Offset (X,Y)* property of the symbol. If the symbol is made with many layers, you may want to assign the field to each of them
2. Selectați instrumentul de Decalare a Simbolurilor Punctelor
3. Click a point symbol
4. Move to a new location
5. Click again. The symbol is moved to the new place. Offset values from the original position are stored in the linked field.

   You can right-click to abort symbol offset.

Notă: The Offset Point Symbols tool doesn't move the point feature itself; you should use the Vertex Tool (Current Layer) or Move Feature tool for this purpose.

Reducere/Extindere Entitate

The Trim/Extend tool allows you to shorten or lengthen segments of a (multi)line or (multi)polygon geometry to converge with a selected segment (the cutting line). This results in a modified geometry with a vertex snapped to the target segment or in its prolongation. Depending on how the selected geometries are placed in relation to each other, the tool will either:

- **Trim**: removes parts of the line segment or polygon boundary, beyond the cutting line
- **Extend**: extends polygon boundaries or line segments so that they can snap to the cutting line.

In order to trim or extend existing geometries:

1. Enable appropriate *snapping settings* on segment for the involved layer(s)
2. Select the Trim/Extend tool
3. Click the target limit segment, i.e. the segment with respect to which you want to extend or trim another segment. It appears highlighted.
4. Move to the segment you want to trim or extend. It does not need to be the last segment of the geometry, but has to be on the active layer.
5. Hover over the segment, and QGIS displays a preview of what the feature's geometry would be. If OK, click the segment. In the case of a trim, you must select the part that should be shortened.
6. When both segments are in 3D, the tool performs an interpolation on the limit segment to get the Z value.

Atenție: Pay attention to the modified geometry while using the Trim/Extend tool. Depending on the inputs, it can create invalid geometries, potentially resulting in failure at layer saving.

16.3.6 Shape digitizing

The *Shape Digitizing* toolbar offers a set of tools to draw lines or polygons features of regular shape. It is synchronized with the Digitize Shape *geometry drawing method* you can select on the *Digitizing Toolbar*. To use it:

1. Display the toolbar: *View* ► *Toolbars* ► *Shape Digitizing*
2. Select a tool that creates or modifies the shape of a geometry, e.g. Add line feature, Add polygon feature, Add part, Add ring, Reshape Features, …
3. The Digitize with segment button on the *Digitizing Toolbar* is enabled. The first time, you may need to switch it to the Digitize Shape in order to enable tools on the *Shape Digitizing* toolbar.
4. Pick a shape digitizing tool and draw.

ir circular în funcție de rază

Butonul Șir circular în funcție de rază permite adăugarea de entități liniare sau poligonale cu o geometrie circulară, având date două noduri pe curbă și o rază:

1. Left click twice to place the two points on the geometry.
2. A *Radius* widget in the top right corner of the map canvas displays current radius (corresponding to distance between the points). Edit that field to the value you want.
3. An overview of the arcs matching these constraints is displayed while moving around the cursor. Right-click to validate when the expected arc is shown.
4. Add a new point to start shaping another arc.

Notă: Curved geometries are stored as such only in compatible data provider

Although QGIS allows to digitize curved geometries within any editable data format, you need to be using a data provider (e.g. PostGIS, memory layer, GML or WFS) that supports curves to have features stored as curved, otherwise QGIS segmentizes the circular arcs.

Desenare Cercuri

Există un set de instrumente dedicate desenării cercurilor. Instrumentele sunt descrise mai jos.

Circles are converted into circular strings. Therefore, as explained in *Șir circular în funcție de rază*, if allowed by the data provider, it will be saved as a curved geometry, if not, QGIS will segmentize the circular arcs.

- Circle from 2 points: The two points define the diameter and the orientation of the circle. (Left-click, right-click)
- Circle from 3 points: Draws a circle from three known points on the circle. (Left-click, left-click, right-click)
- Circle by a center point and another point: Draws a circle with a given center and a point on the circle (Left-click, right-click). When used with the *Panoul de Digitizare Avansată* this tool can become a „Add circle from center and radius" tool by setting and locking the distance value after first click.

- Circle from 3 tangents: Draws a circle that is tangential to three segments. **Note that you must activate snapping to segments** (See *Setting the snapping tolerance and search radius*). Click on a segment to add a tangent. If two tangents are parallel, the coordinates of the click on the first parallel tangent are used to determine the positioning of the circle. If three tangents are parallel, an error message appears and the input is cleared. (Left-click, left-click, right-click)
- Circle from 2 tangents and a point: Similar to circle from 3 tangents, except that you have to select two tangents, enter a radius and select the desired center.

Desenare Elipse

Există un set de instrumente dedicate desenării elipselor. Instrumentele sunt descrise mai jos.

Ellipses cannot be converted as circular strings, so they will always be segmented.

- Ellipse from center and two points: Draws an ellipse with a given center, major axis and minor axis. (Left-click, left-click, right-click)
- Ellipse from center and a point: Draws an ellipse into a bounding box with the center and a corner. (Left-click, right-click)
- Ellipse from extent: Draws an ellipse into a bounding box with two opposite corners. (Left-click, right-click)
- Ellipse from foci: Draws an ellipse by 2 points for foci and a point on the ellipse. (Left-click, left-click, right-click)

Desenare Dreptunghiuri

Există un set de instrumente dedicate desenării dreptunghiurilor. Instrumentele sunt descrise mai jos.

- Rectangle from center and a point: Draws a rectangle from the center and a corner. (Left-click, right-click)
- Rectangle from extent: Draws a rectangle from two opposite corners. (Left-click, right-click)
- Rectangle from 3 points (distance): Draws an oriented rectangle from three points. The first and second points determine the length and angle of the first edge. The third point determines the length of the other edge. One can use *Panoul de Digitizare Avansată* to set the length of the edges. (Left-click, left-click, right-click)
- Rectangle from 3 points (projected): Same as the preceding tool, but the length of the second edge is computed from the projection of the third point on the first edge. (Left-click, left-click, right-click)

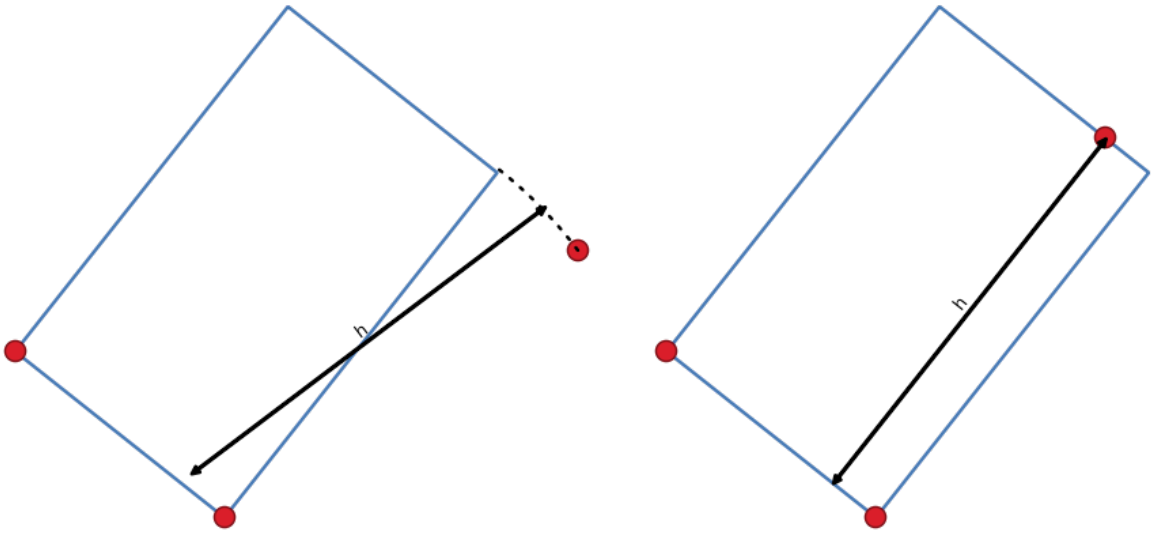


Fig. 16.88: Draw rectangle from 3 points using distance (right) and projected (left)

Desenare Poligoane Regulate

Există un set de instrumente dedicate desenării poligoanelor regulate. Instrumentele sunt descrise mai jos. Faceți clic stânga pentru a plasa primul punct. Va apărea o fereastră de dialog, unde veți puteți seta numărul de margini ale poligonului. Faceți clic dreapta pentru a finaliza poligonul regulat.

- Poligon regulat pornind de la două puncte: Desenează un poligon regulat în care cele două puncte determină lungimea și unghiul primei muchii.
- Poligon regulat pornind de la centru și un punct: Desenează un poligon regulat plecând de la punctul central. Al doilea punct determină unghiul și distanța până la mijlocul muchiei.
- Regular polygon from center and a corner: Same as the preceding tool, but the second point determines the angle and distante to a vertex.

16.3.7 Panoul de Digitizare Avansată

When capturing, reshaping, splitting new or existing geometries you also have the possibility to use the Advanced Digitizing panel. You can digitize lines exactly parallel or perpendicular to a particular angle or lock lines to specific angles. Furthermore, you can make a precise definition of your new geometry by entering X and Y coordinates as well as Z for 3D features, or M values.

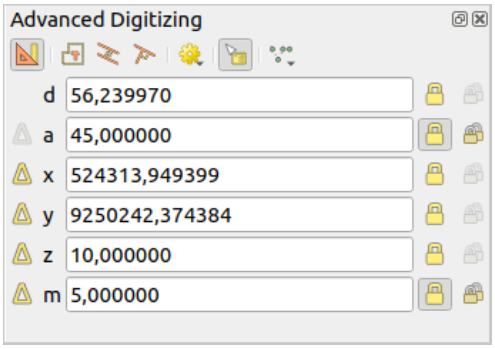


Fig. 16.89: Panoul de Digitizare Avansată

The *Advanced Digitizing* panel can be opened either with a right-click on the toolbar, from *View* ► *Panels* ► menu or pressing `Ctrl+4`. Once the panel is visible, click the Enable advanced digitizing tools button to activate the set of tools.

Notă: Instrumentele nu sunt activate dacă vizualizarea hărții este în coordonate geografice.

The aim of the Advanced Digitizing tool is to lock coordinates, lengths, and angles when moving the mouse during the digitalizing in the map canvas.

You can also create constraints with relative or absolute reference. Relative reference means that the next vertex constraints» values will be relative to the previous vertex or segment.

The toolbar

At the top of the *Digitizing panel*, you find the following buttons:

- Enable advanced digitizing tools
- Construction mode: allows to capture the clicks» positions to reuse as reference points to lock distance, angle, X, Y, Z or M relative values. More details at *Construction mode*.
- Parallel to draw a line parallel to an existing one (more at *Parallel and perpendicular lines*)
- Perpendicular to draw a line perpendicular to an existing one (more at *Parallel and perpendicular lines*)
- Snap to common angles: when moving the cursor, displays a virtual line that you can snap to to add the next vertex. The snapping line is defined by the last added vertex and an (absolute or relative to previous segment) angle from a preset list (following steps of 0.1°, 0.5°, 1°, 5°, 10°, 15°, 18°, 22.5°, 30°, 45° or 90°). Choose *Do not snap to common angles* to disable this feature.

  Snapping to features can be used along with snapping to common angles for accurate digitizing. For a fine-grained control on how the target element to snap to is retained, you can indicate whether to prioritize snapping to features over common angles, and vice-versa under the *Snapping priority* entry. You can switch from one method to the other during the digitizing operation, and this avoids disabling any of the snapping options in the meantime. Press `N` (or `Shift+N`) during a digitizing operation to cycle through the angles list.
- Floater settings: if the *Show floater* item is checked, a contextual menu with digitizing information follows the cursor during digitizing. The values can be accessed using the *panel's shortcuts*, edited and Locked after validation (pressing `Enter`). The type of information to display can be selected in the bottom part of the menu:
  - *Show distance*
  - *Show angle*

  - *Show XY coordinates*
  - *Show Z value*
  - *Show M value*
  - *Show bearing/azimuth*
  - *Show common snapping angle*

- Construction Tools provides a couple of options that constrain the vertices placement based on extrapolated coordinates of existing elements:
  - *Line Extension*: hover over a segment and you get a purple dotted line extending the segment across the map canvas. You can snap the vertex anywhere on this virtual line.
  - *X/Y Point*: hover over a vertex and you get a purple dotted line along its X or Y coordinate, across the map canvas. You can snap the vertex anywhere on this virtual line. It is even possible to hover over two different vertices, generating virtual coordinate lines for both, and snap to their intersection.

Below the toolbar, you will find a number of text boxes whose value reflects by default the position or movement of the cursor in the map canvas. Editing these values helps you constrain the position of the items you edit:

- *d* for the distance from a reference position, usually the last edited vertex
- *a* for the angle (absolute or relative) from a reference position, usually the last edited segment
- *x* for the X coordinate of the pointer
- *y* for the Y coordinate of the pointer
- *z* for the default Z value or the Z coordinate of the vertex or segment under the pointer
- *m* for the default M value or the M value of the vertex or segment under the pointer

Combinații de taste

To speed up the use of Advanced Digitizing Panel, there are a couple of keyboard shortcuts available:

Tabelul 16.5: Keyboard shortcuts of the Advanced Digitizing Panel tools

Cheie	Simplu	`Ctrl+` or `Alt+`	`Shift+`
`D`	Set distance	Lock distance	
`A`	Set angle	Lock angle	Toggle relative angle to last segment
`X`	Set X coordinate	Lock X coordinate	Toggle relative X to last vertex
`Y`	Set Y coordinate	Lock Y coordinate	Toggle relative Y to last vertex
`Z`	Set Z coordinate	Lock Z coordinate	Toggle relative Z to last vertex
`M`	Set M value	Lock M value	Toggle relative M to last vertex
`C`	Toggle construction mode		
`P`	Toggle perpendicular and parallel modes		

Notă: Z coordinate and M value options are available only if compatible with the layer geometry dimension.

Absolute reference digitizing

When drawing a new geometry from scratch, it is very useful to have the possibility to start digitizing vertexes at given coordinates.

For example, to add a new feature to a polygonal layer, click the button. You can enter the exact coordinates where you want to start editing the feature, i.e.:

1. Click the *x* text box (or use the `X` keyboard shortcut).
2. Type the X coordinate value you want and press `Enter` or click the button to their right to lock the mouse to the X axis on the map canvas.
3. Click the *y* text box (or use the `Y` keyboard shortcut).
4. Type the Y coordinate value you want and press `Enter` or click the button to their right to lock the mouse to the Y axis on the map canvas.
5. If the layer has Z coordinate or M values, the corresponding *z* or *m* widget is enabled and displays its default value, as set in *Settings* ► *Options* ► *Digitizing* tab.
   1. Click the *z* or *m* text box (or use respectively the `Z` or `M` keyboard shortcut).
   2. Type the coordinate value you want and press `Enter` or click the button to their right to lock the value in the widget.

   Notă: Read *Rules of Z coordinate or M value assignment* for details on how Z coordinate and M values are automatically determined from existing features.

   ---

6. Two blue dotted lines and a green cross identify the exact coordinates you entered. Click on the map canvas to add a vertex at the green cross position.

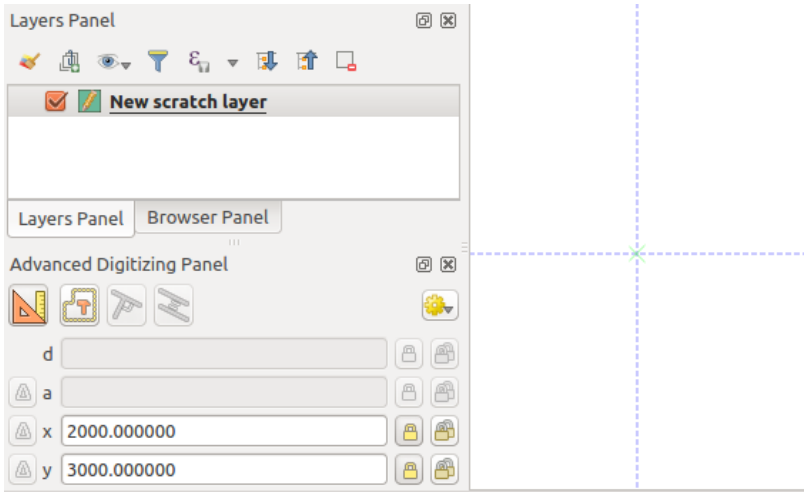


Fig. 16.90: Start drawing at given coordinates

7. You can proceed as above, adding a new set of coordinates for the next vertex, or switch to another *mode of digitizing* (e.g. segment, curve or stream).
8. If you want to draw a segment of a given length:
   1. Click the *d (distance)* text box (keyboard shortcut `D`)
   2. Type the distance value (in map units)

3. Press `Enter` or click the button on the right to lock the mouse in the map canvas to the length of the segment. In the map canvas, the latest vertex is surrounded by a circle whose radius is the value entered in the distance text box. A cross on the circle shows the position of the next vertex if you click.

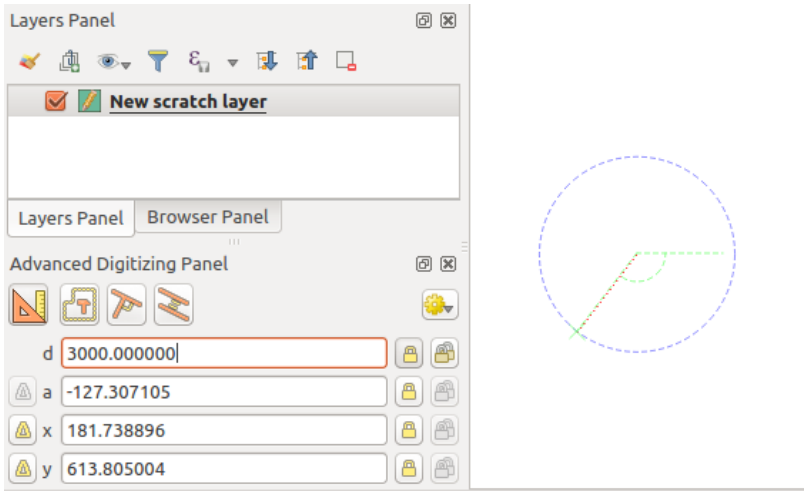


Fig. 16.91: Fixed length segment

9. You can also constrain the vertex position, setting the angle of the segment. As described before:
   1. Click the *a (angle)* text box (keyboard shortcut `A`)
   2. Type the angle value (in degrees)
   3. Press `Enter` or click the button on the right to lock it. A line going through the latest vertex and rotated based on the set angle appears in the map canvas and a cross on it shows the next vertex position if you click.

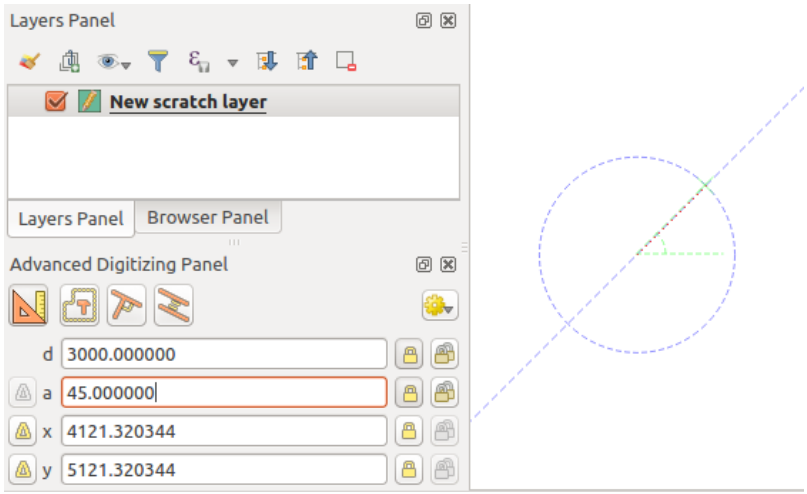


Fig. 16.92: Fixed angle segment

Sugestie: Pressing `Ctrl+<key>` or `Alt+<key>` automatically locks the target property and puts its value into edit. Modify, press `Enter` and you are done. Combined with the Toggle floater, this can be a real time saver, with keyboard digitizing.

Relative reference digitizing

Instead of using absolute values of angles or coordinates, you can also use values relative to the last digitized vertex or segment.

For angles, you can click the button on the left of the *a* text box (or press `Shift+A`) to toggle relative angles to the previous segment. With that option on, angles are measured between the last segment and the mouse pointer.

For coordinates, click the buttons to the left of the *x*, *y*, *z* or *m* text boxes (or press `Shift+<key>`) to toggle relative coordinates to the previous vertex. With these options on, coordinates measurement will consider the last vertex to be the origin of the set coordinates.

Continuous lock

Both in absolute or relative reference digitizing, angle, distance, X, Y, Z and M constraints can be locked continuously by clicking the *Continuous lock* buttons. Using continuous lock allows you to digitize several points or vertexes using the same constraints.

Parallel and perpendicular lines

All the tools described above can be combined with the Perpendicular and Parallel tools. These two tools allow drawing segments perfectly perpendicular or parallel to another segment. The target segment can be on another layer, another feature within the layer or the feature being digitized (requires *self-snapping option*).

To draw a *perpendicular* segment:

1. First add one of the segment vertices.
2. Click the Perpendicular icon (keyboard shortcut `P`) to activate it.
3. Click on the segment that you want to be perpendicular to.
4. A virtual dotted line perpendicular to the segment through the previous vertex appears. The angle property is locked, constraining the next vertex on that line and, a cross indicates the projected position of the cursor on the line. Click to place the new vertex.

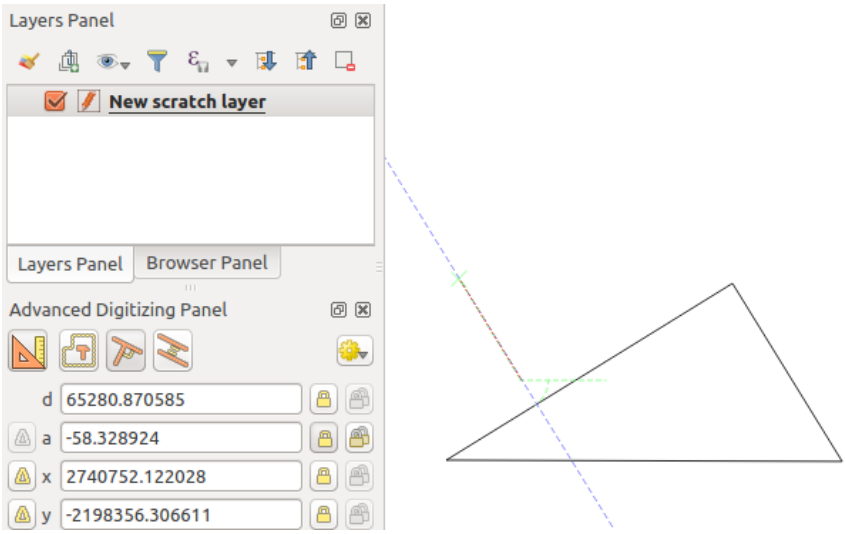


Fig. 16.93: Perpendicular digitizing

To draw a *parallel* segment, the steps are the same except that you need to click on the Parallel icon (keyboard shortcut `P` twice).

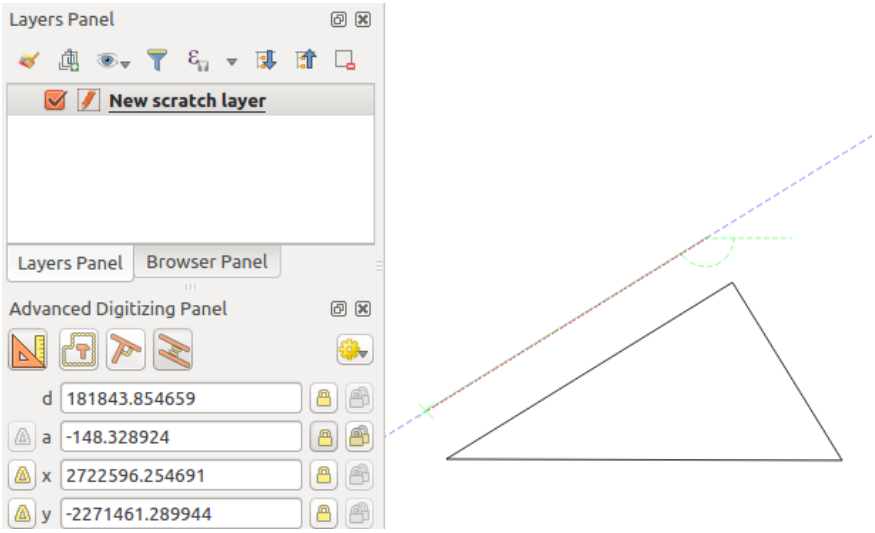


Fig. 16.94: Parallel digitizing

These two tools just find the right angle of the perpendicular and parallel angle and lock this parameter during your editing. Unlock the angle parameter to cancel their use in the middle of the process.

Construction mode

You can enable and disable *construction mode* by clicking on the Construction mode icon or with the `C` keyboard shortcut. While in construction mode, clicking the map canvas won't add new vertexes, but will capture the clicks» positions so that you can use them as reference points to then lock distance, angle or X, Y, Z, M relative values.

As an example, the construction mode can be used to draw some point at an exact distance from an existing point.

With an existing point in the map canvas and the snapping mode correctly activated, you can easily draw other points at given distances and angles from it. In addition to the button, you have to activate also the *construction mode* by clicking the Construction mode icon or with the `C` keyboard shortcut.

Click next to the point from which you want to calculate the distance and click on the *d* box (`D` shortcut) type the desired distance and press `Enter` to lock the mouse position in the map canvas:

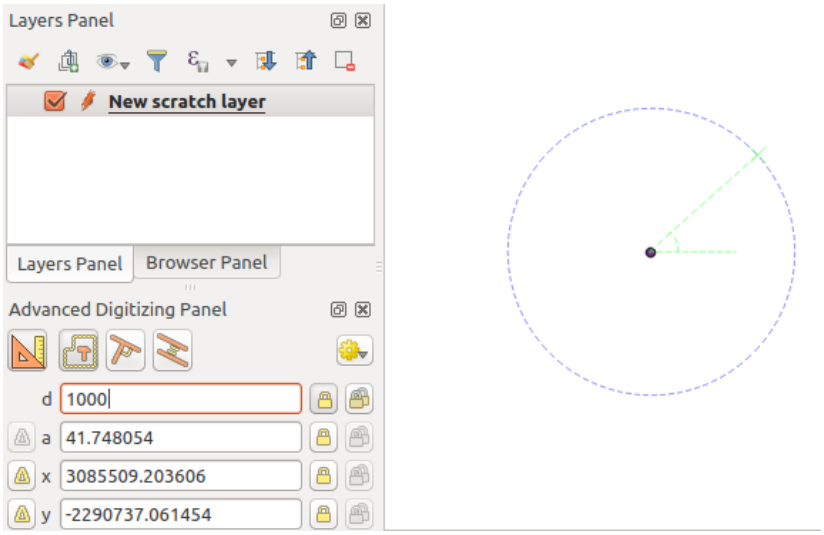


Fig. 16.95: Distance from point

Before adding the new point, press `C` to exit the construction mode. Now, you can click on the map canvas, and the point will be placed at the distance entered.

You can also use the angle constraint to, for example, create another point at the same distance of the original one, but at a particular angle from the newly added point. Click the Construction mode icon or with the `C` keyboard shortcut to enter construction mode. Click the recently added point, and then the other one to set a direction segment. Then, click on the *d* text box (`D` shortcut) type the desired distance and press `Enter`. Click the *a* text box (`A` shortcut) type the angle you want and press `Enter`. The mouse position will be locked both in distance and angle.

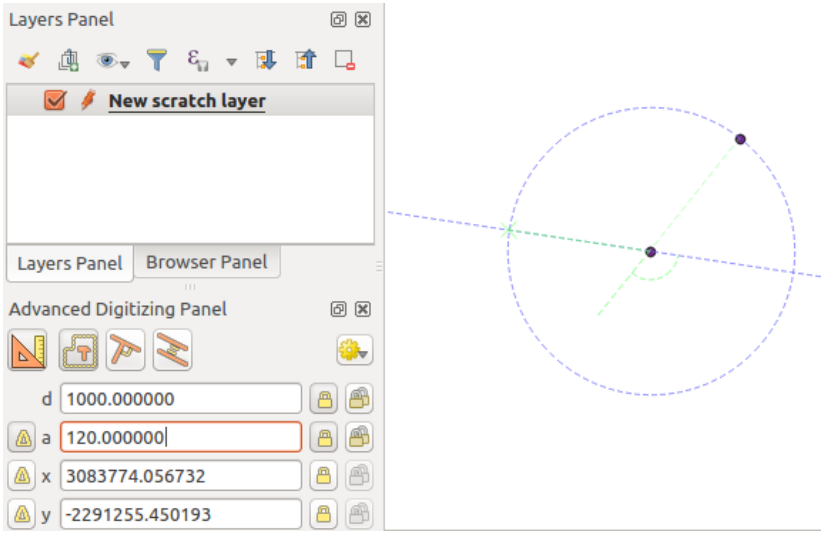


Fig. 16.96: Distance and angle from points

Before adding the new point, press `C` to exit the construction mode. Now, you can click on the map canvas, and the point will be placed at the distance and angle entered. Repeating the process, several points can be added.

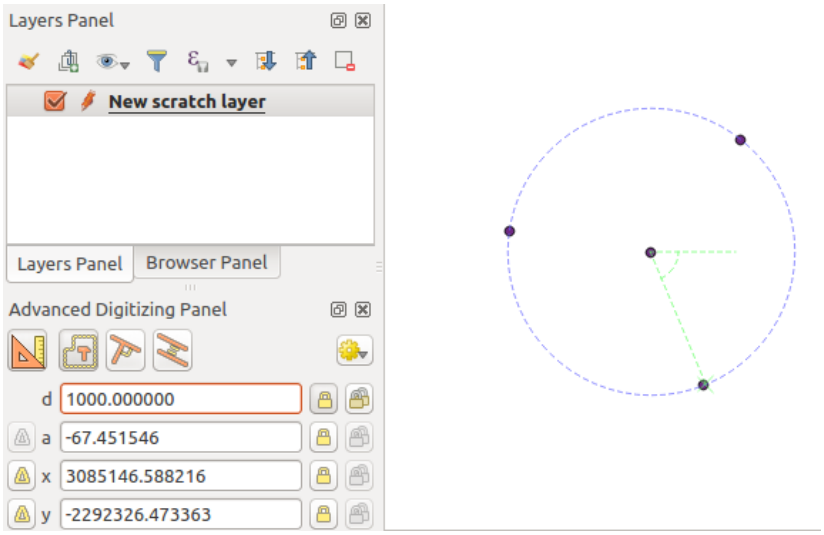


Fig. 16.97: Points at given distance and angle

16.3.8 The Processing in-place layer modifier

The *Processing menu* provides access to a large set of tools to analyze and create new features based on the properties of the input features or their relations with other features (within the same layer or not). While the common behavior is to create new layers as outputs, some algorithms also allow modifications to the input layer. This is a handy way to automate multiple features modification using advanced and complex operations.

To edit features in-place:

1. Select the layer to edit in the *Layers* panel.
2. Select the concerned features. You can skip this step, in which case the modification will apply to the whole layer.
3. Press the Edit Features In-Place button at the top of the *Processing toolbox*. The list of algorithms is filtered, showing only those compatible with in-place modifications, i.e.:
   - They work at the feature source and not at the layer level.
   - They do not change the layer structure, e.g. adding or removing fields.
   - They do not change the geometry type, e.g. from line to point layer.

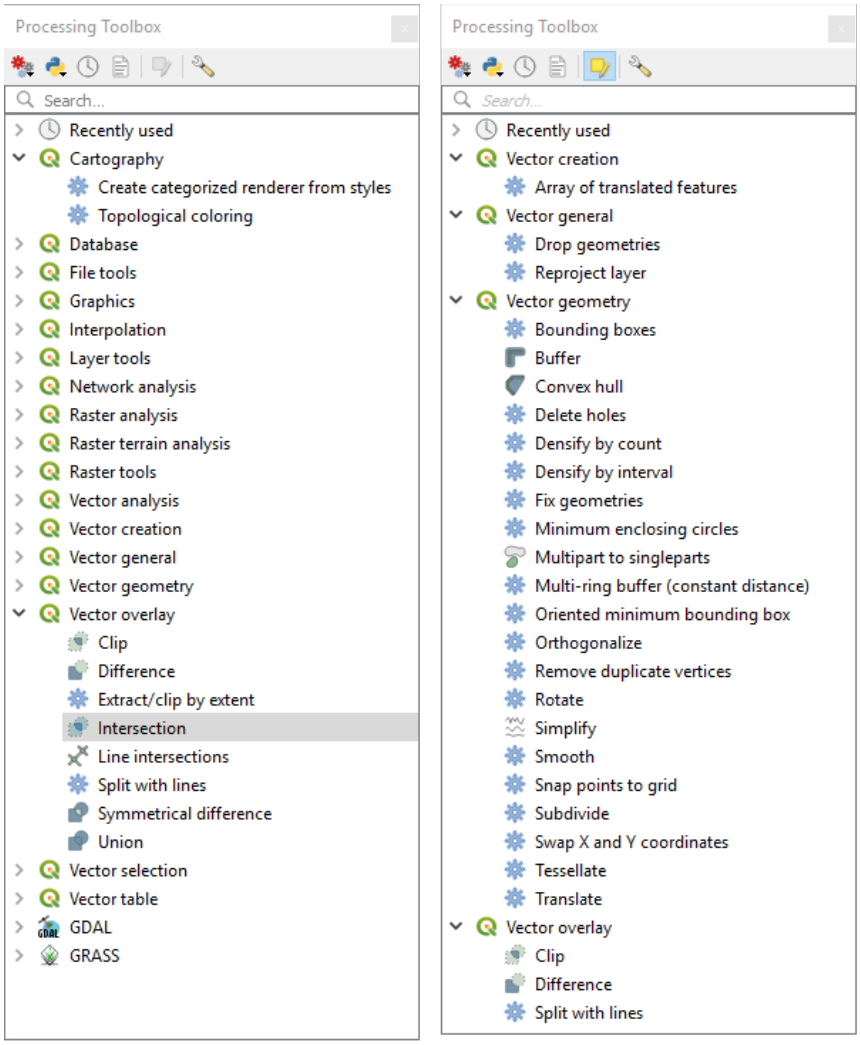


Fig. 16.98: Processing algorithms: all (left) vs polygon in-place editors (right)

4. Find the algorithm you'd like to run and double-click it.

Notă: If the algorithm does not need any additional user-set parameters (excluding the usual input and output layer parameters), then the algorithm is run immediately without any dialog popup.

---

   1. If parameters other than the usual input or output layers are needed, the algorithm dialog pops up. Fill in the required information.
   2. Click *Modify Selected Features* or *Modify All Features* depending on whether there's an active selection.

   Changes are applied to the layer and placed in the edit buffer: the layer is indeed toggled to editing mode with unsaved modification as indicated by the icon next to the layer name.

5. As usual, press Save layer edits to commit the changes in the layer. You can also press Undo to rollback the whole modification.

16.4 Connecting and Editing Data Across Layers

Ability to connect data from different layers is one of the duties of a GIS software. Such a connection can be based on the spatial relationship between the features, or on their shared attributes. QGIS provides tools to handle any of these associations, such as:

- Processing algorithms that can create a new layer as a result of the connection, namely *Join attributes by location*, *Join attributes by nearest*, *Join attributes by field value*, ...
- SQL queries to create a new layer from the *DB Manager* or as a *virtual layer*
- *Joins properties* or *relations settings* that temporarily extend attributes of features in a given layer, with those of features in another layer based on some matching attribute(s).

  Joins and relations are technical concepts borrowed from databases to get the most out of data stored in tables by combining their contents. The idea is that features (rows) of different layers (tables) can be associated to each other. The number of rows which are matching each other can be of any value (zero, one, many).

16.4.1 Joining features between two layers

Joins in QGIS allow you to associate features of the current layer to features from another loaded vector layer. Whether they are spatially enabled and the type of geometry do not matter. The join is based on an attribute that is shared by the layers, in a one-to-one relationship.

To create a join on a layer (identified below as `target layer`):

1. Go to the layer *Properties* ► *Joins* tab
2. Click the Add new join button. The *Add vector join* dialog appears.
3. Select the *Join layer* you want to connect with the target vector layer
4. Specify the *Join field* (from the `join layer`) and the *Target field* (from the `target layer`). These are the fields that are used to find matching feature in both layers hence they should have values in common.
5. Press *OK* and a summary of selected parameters is added to the *Join* panel.

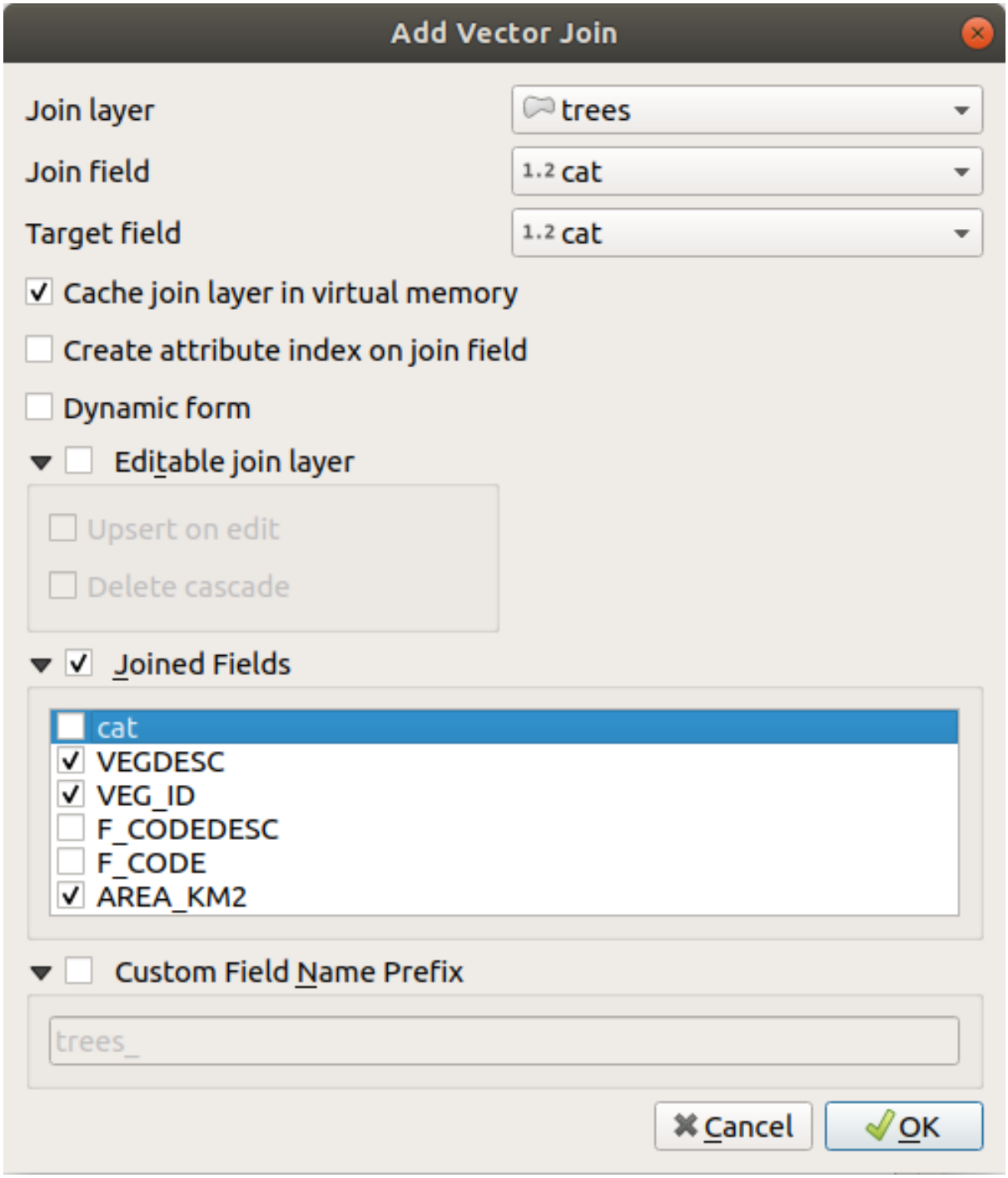


Fig. 16.99: Join an attribute table to an existing vector layer

The steps above will create a join, where **ALL** the attributes of the **first matching feature** in the join layer is added to the target layer's feature. The following logic is used to pair features during a join process:

- All the features in the target layer are returned, regardless they have a match
- If the target field contains duplicate values, these features are assigned the same feature from the join layer.
- If the join field contains duplicate matching values, only the first fetched feature is picked.

Notă: Joins in QGIS are based on a single field matching so most of the times, you would want to make sure that

values in the matchable fields are unique.

QGIS provides some more options to tweak the join:

- *Cache join layer in virtual memory*: allows you to cache values in memory (without geometries) from the joined layer in order to speed up lookups.
- *Create attribute index on the join field* to speed up lookups
- *Dynamic form*: helps to synchronize join fields on the fly, according to the *Target field*. This way, constraints for join fields are also correctly updated. Note that it's deactivated by default because it may be very time consuming if you have a lot of features or a myriad of joins.
- If the target layer is editable, then some icons will be displayed in the attribute table next to fields, in order to inform about their status:
  - : the join layer is not configured to be editable. If you want to be able to edit join features from the target attribute table, then you have to check the option *Editable join layer*.
  - : the join layer is well configured to be editable, but its current status is read only.
  - : the join layer is editable, but synchronization mechanisms are not activated. If you want to automatically add a feature in the join layer when a feature is created in the target layer, then you have to check the option *Upsert on edit*. Symmetrically, the option *Delete cascade* may be activated if you want to automatically delete join features.
- *Joined fields*: instead of adding all the fields from the joined layer, you can specify a subset.
- *Custom field name prefix* for joined fields, in order to avoid name collision

16.4.2 Setting relations between multiple layers

Unlike joins that define a one-to-one link between features across two layers, relations help you build interconnections between multiple features across two or more layers. As such, relations are project level settings and are set in *Project* ► *Properties* ► *Relations* tab. From there, you can:

- *Add relation* whose type can be:
  - *one to many relation*
  - *many to many relation*
  - *polymorphic relation* that you can add or edit with the dedicated tools in the action drop-down menu.

  Notă: There is no simple way yet to edit a non-polymorphic relation once it has been created. Only the name can be edited with a double-click. For any other parameters of such a relation you will have to remove and recreate it.
- *Discover relations*: QGIS is able to discover existing relations from supported database formats (PostgreSQL, GeoPackage, ESRI File Geodatabase, …). This can be a good way to ease the relations definition.
- *Remove relation*

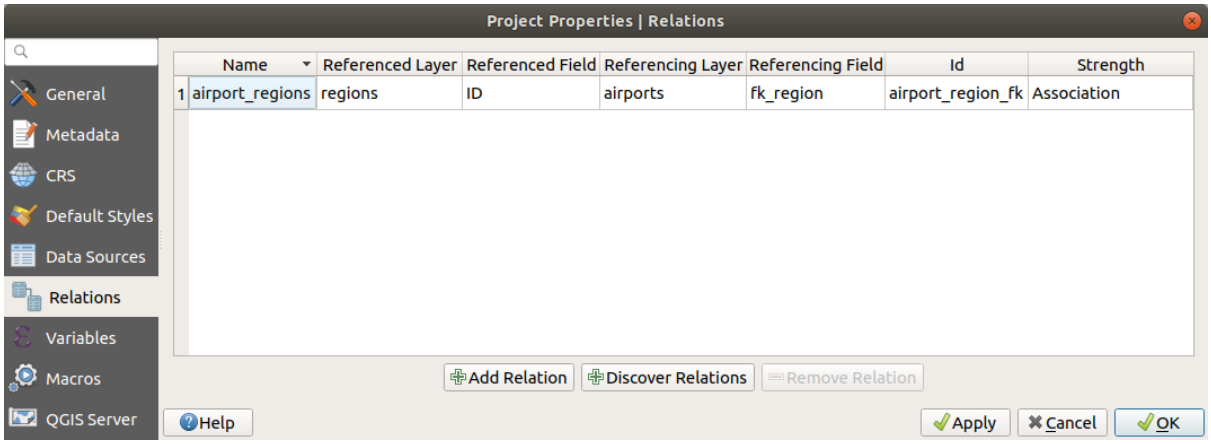


Fig. 16.100: Relations tab

One to many (1-N) relations

As an example you have a layer with all regions of Alaska (polygon) which provides some attributes about its name and region type and a unique id (which acts as primary key).

Then you get another point layer or table with information about airports that are located in the regions and you also want to keep track of these. If you want to add them to the regions layer, you need to create a one to many relation using foreign keys, because there are several airports in most regions.

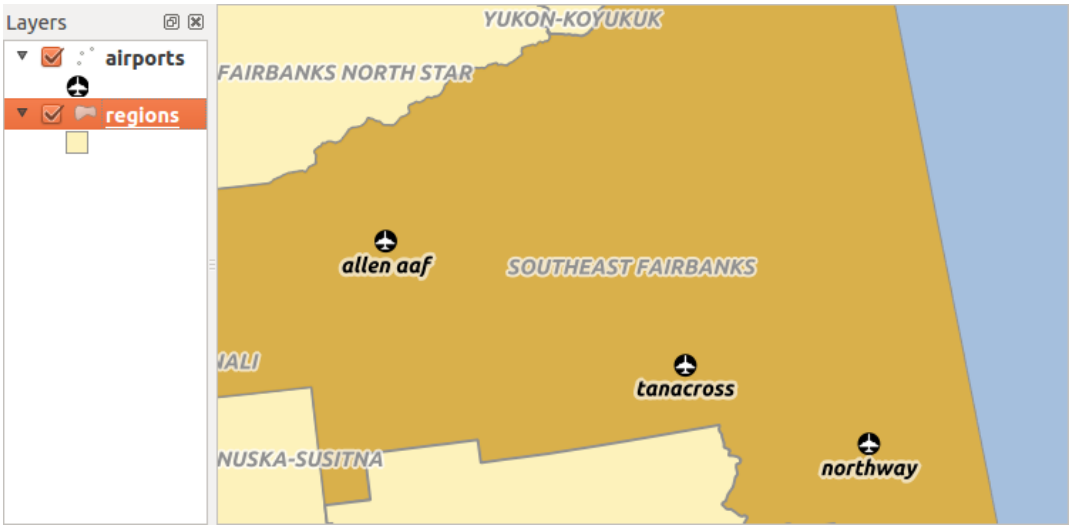


Fig. 16.101: Alaska region with airports

Layers and keys

QGIS makes no difference between a table and a vector layer. Basically, a vector layer is a table with a geometry. So you can add your table as a vector layer. To demonstrate the 1-n relation, you can load the `regions` and `airports` layers in the sample dataset. In practice, each airport belongs to exactly one region while each region can have any number of airports (a typical one to many relation).

which has a foreign key field (`fk_region`) to the layer regions.

In addition to the attributes describing the airports, the aiports layer has another field `fk_region` which acts as a foreign key (if you have a database, you will probably want to define a constraint on it). This `fk_region` field will always contain an id of a region. It can be seen like a pointer to the region it belongs to.

All you have to do is to tell QGIS the relation between the layers so that you can design a custom edit form for editing and QGIS takes care of the setup. It works with different providers (so you can also use it with shape and csv files).

Defining 1-N relations

The first thing we are going to do is to let QGIS know about the relations between the layers. This is done in *Project* ► *Properties…*. Open the *Relations* tab and click on *Add Relation*.

- **Name** is going to be used as a title. It should be a human readable string describing what the relation is used for. We will just call say **airport_relation** in this case.
- **Referenced Layer (Parent)** also considered as parent layer, is the one with the primary key, pointed to, so here it is the `regions` layer. You need to define the primary key of the referenced layer, so it is `ID`.
- **Referencing Layer (Child)** also considered as child layer, is the one with the foreign key field on it. In our case, this is the `airports` layer. For this layer you need to add a referencing field which points to the other layer, so this is `fk_region`.

Notă: Sometimes, you need more than a single field to uniquely identify features in a layer. Creating a relation with such a layer requires a **composite key**, i.e. more than a single pair of matching fields. Use the Add new field pair as part of a composite foreign key button to add as many pairs as necessary.

- **Id** will be used for internal purposes and has to be unique. You may need it to build *custom forms*. If you leave it empty, one will be generated for you but you can assign one yourself to get one that is easier to handle
- **Relationship strength** sets the strength of the relation between the parent and the child layer. The default *Association* type means that the parent layer is *simply* linked to the child one while the *Composition* type allows you to duplicate also the child features when duplicating the parent ones and on deleting a feature the children are deleted as well, resulting in cascade over all levels (means children of children of… are deleted as well).

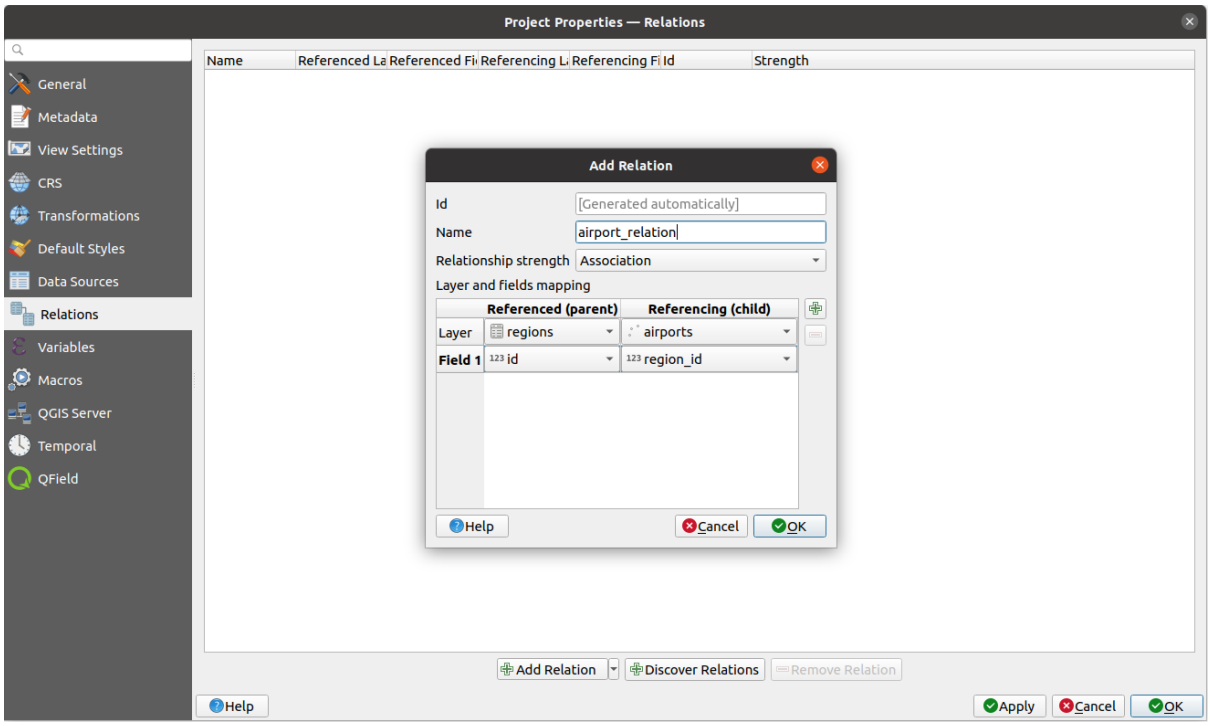


Fig. 16.102: Adding a relation between regions and airports layers

From the *Relations* tab, you can also press the *Discover Relation* button to fetch the relations available from the

providers of the loaded layers. This is possible for layers stored in data providers like PostgreSQL or SpatiaLite.

Forms for 1-N relations

Now that QGIS knows about the relation, it will be used to improve the forms it generates. As we did not change the default form method (autogenerated), it will just add a new widget in our form. So let's select the layer region in the legend and use the identify tool. Depending on your settings, the form might open directly or you will have to choose to open it in the identification dialog under actions.

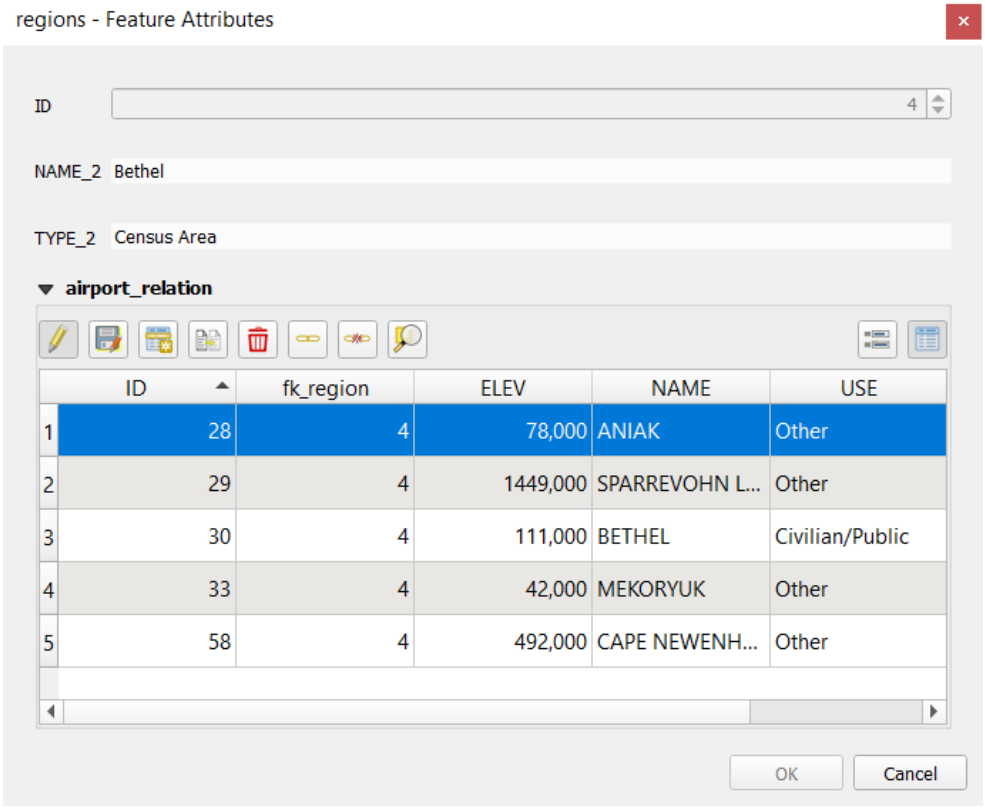


Fig. 16.103: Identification dialog regions with relation to airports

As you can see, the airports assigned to this particular region are all shown in a table. And there are also some buttons available. Let's review them shortly:

- The button is for toggling the edit mode. Be aware that it toggles the edit mode of the airport layer, although we are in the feature form of a feature from the region layer. But the table is representing features of the airport layer.
- The button is for saving all the edits in the child layer (airport).
- The button lets you digitize the airport geometry in the map canvas and assigns the new feature to the current region by default. Note that the icon will change according to the geometry type.
- The button adds a new record to the airport layer attribute table and assigns the new feature to the current region by default. The geometry can be drawn later with the *Add part* digitizing tool.
- The button allows you to copy and paste one or more child features within the child layer. They can later be assigned to a different parent feature or have their attributes modified.
- The button deletes the selected airport(s) permanently.

- The symbol opens a new dialog where you can select any existing airport which will then be assigned to the current region. This may be handy if you created the airport on the wrong region by accident.
- The symbol unlinks the selected airport(s) from the current region, leaving them unassigned (the foreign key is set to NULL) effectively.
- With the button you can zoom the map to the selected child features.
- The two buttons and to the right switch between the *table view and form view* of the related child features.

If you use the *Drag and Drop Designer* for the regions feature, you can select which tools are available. You can even decide whether to open a new form when a new feature is added using *Force hide form on add feature* option. Be aware that this option implies that not null attributes must take a valid default value to work correctly.

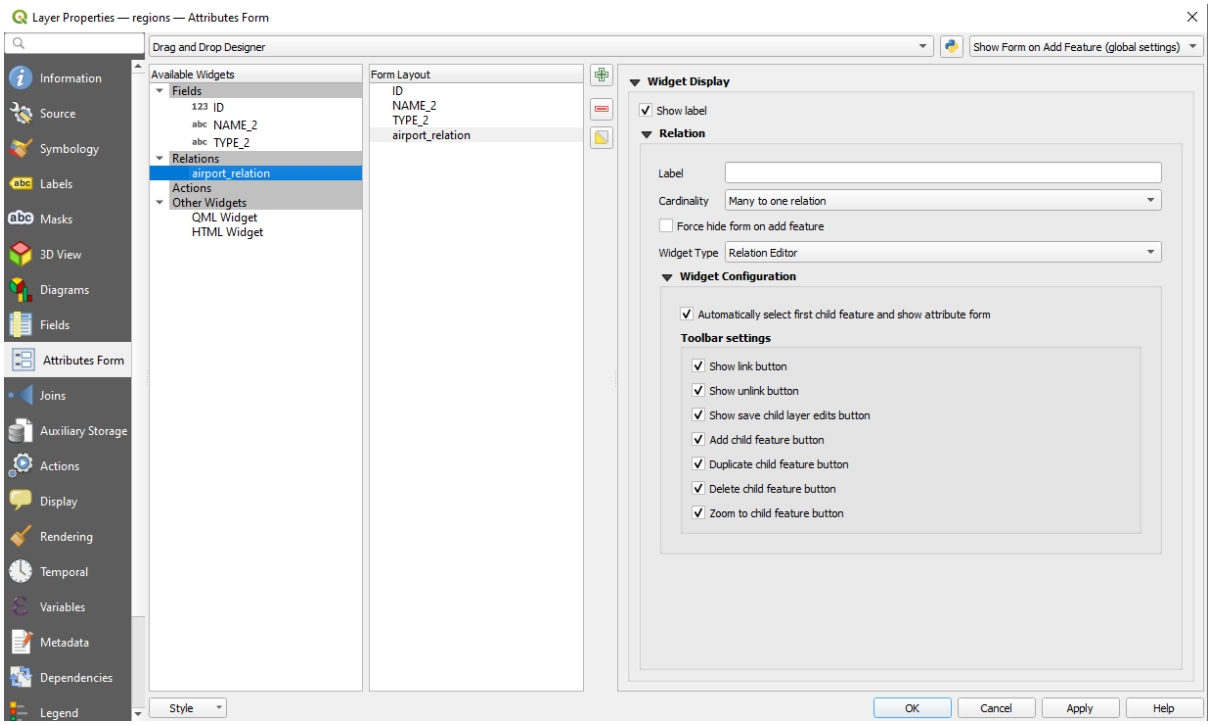


Fig. 16.104: Drag and Drop Designer for configure regions-airports relation tools

In the above example the referencing layer has geometries (so it isn't just an alphanumeric table) so the above steps will create an entry in the layer attribute table that has no corresponding geometric feature. To add the geometry:

1. Choose *Open Attribute Table* for the referencing layer.
2. Select the record that has been added previously within the feature form of the referenced layer.
3. Use the Add Part digitizing tool to attach a geometry to the selected attributes table record.

If you work on the airport table, the widget Relation Reference is automatically set up for the `fk_region` field (the one used to create the relation), see *Relation Reference widget*.

In the airport form you will see the button at the right side of the `fk_region` field: if you click on the button the form of the region layer will be opened. This widget allows you to easily and quickly open the forms of the linked parent features.

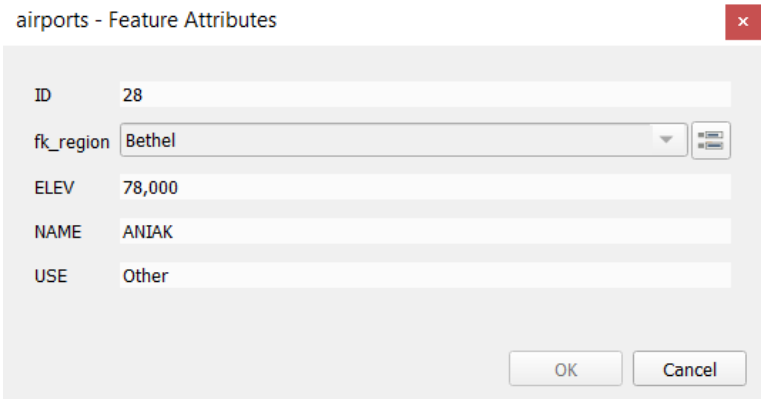
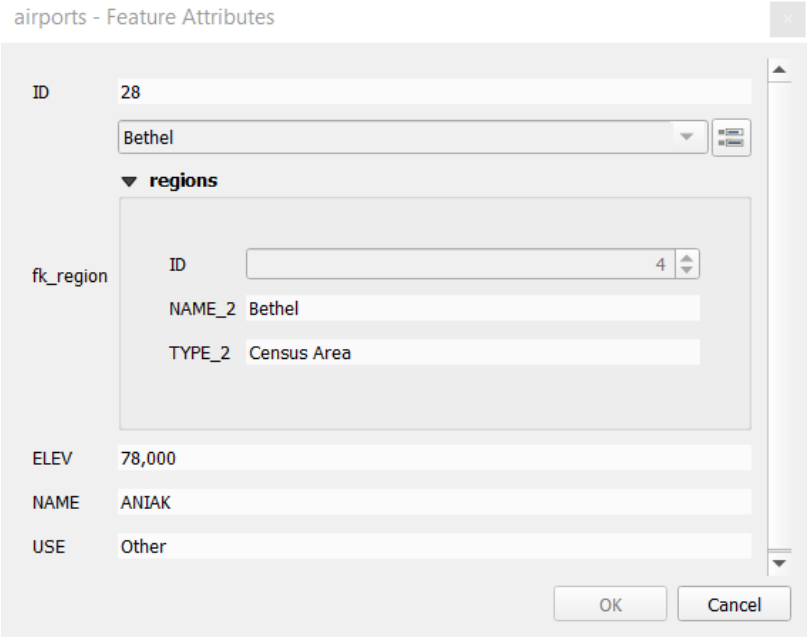


Fig. 16.105: Identification dialog airport with relation to regions

The Relation Reference widget has also an option to embed the form of the parent layer within the child one. It is available in the *Properties* ► *Attributes Form* menu of the airport layer: select the `fk_region` field and check the `Show embedded form` option.

If you look at the feature dialog now, you will see that the form of the region is embedded inside the airports form and will even have a combobox, which allows you to assign the current airport to another region.



Moreover if you toggle the editing mode of the airport layer, the `fk_region` field has also an autocompleter function: while typing you will see all the values of the `id` field of the region layer. Here it is possible to digitize a polygon for the region layer using the button if you chose the option `Allow adding new features` in the *Properties* ► *Attributes Form* menu of the airport layer.

The child layer can also be used in the *Select Features By Value* tool in order to select features of the parent layer based on attributes of their children.

In Fig. 16.106, all the regions where the mean altitude of the airports is greater than 500 meters above sea level are selected.

You will find that many different aggregation functions are available in the form.

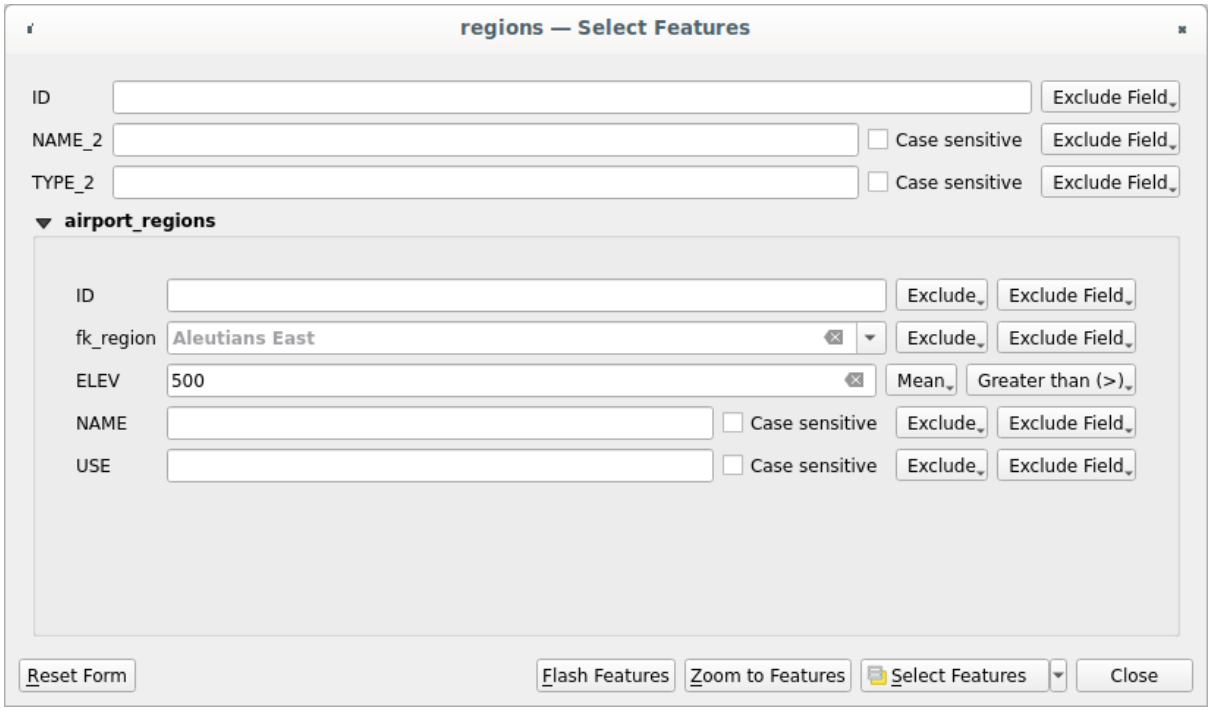


Fig. 16.106: Select parent features with child values

Many-to-many (N-M) relations

N-M relations are many-to-many relations between two tables. For instance, the `airports` and `airlines` layers: an airport receives several airline companies and an airline company flies to several airports.

This SQL code creates the three tables we need for an N-M relationship in a PostgreSQL/PostGIS schema named *locations*. You can run the code using the *Database* ► *DB Manager*… for PostGIS or external tools such as pgAdmin. The airports table stores the `airports` layer and the airline table stores the `airlines` layer. In both tables few fields are used for clarity. The *tricky* part is the `airports_airlines` table. We need it to list all airlines for all airports (or vice versa). This kind of table is known as a *pivot table*. The *constraints* in this table force that an airport can be associated with an airline only if both already exist in their layers.

```
CREATE SCHEMA locations;

CREATE TABLE locations.airports
(
   id serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
   airport_name text NOT NULL,
   CONSTRAINT airports_pkey PRIMARY KEY (id)
);

CREATE INDEX airports_geom_idx ON locations.airports USING gist (geom);

CREATE TABLE locations.airlines
(
   id serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
   airline_name text NOT NULL,
   CONSTRAINT airlines_pkey PRIMARY KEY (id)
);

CREATE INDEX airlines_geom_idx ON locations.airlines USING gist (geom);
```

(continues on next page)

(continuare din pagina precedentă)

```
CREATE TABLE locations.airports_airlines
(
   id serial NOT NULL,
   airport_fk integer NOT NULL,
   airline_fk integer NOT NULL,
   CONSTRAINT airports_airlines_pkey PRIMARY KEY (id),
   CONSTRAINT airports_airlines_airport_fk_fkey FOREIGN KEY (airport_fk)
      REFERENCES locations.airports (id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
      DEFERRABLE INITIALLY DEFERRED,
   CONSTRAINT airports_airlines_airline_fk_fkey FOREIGN KEY (airline_fk)
      REFERENCES locations.airlines (id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
      DEFERRABLE INITIALLY DEFERRED
 );
```

Instead of PostgreSQL you can also use GeoPackage. In this case, the three tables can be created manually using the *Database* ► *DB Manager*.... In GeoPackage there are no schemas so the *locations* prefix is not needed.

Foreign key constraints in `airports_airlines` table can´t be created using *Table* ► *Create Table*... or *Table* ► *Edit Table*... so they should be created using *Database* ► *SQL Window*.... GeoPackage doesn't support *ADD CONSTRAINT* statements so the `airports_airlines` table should be created in two steps:

1. Set up the table only with the `id` field using *Table* ► *Create Table*...
2. Using *Database* ► *SQL Window*..., type and execute this SQL code:

```
ALTER TABLE airports_airlines
   ADD COLUMN airport_fk INTEGER
   REFERENCES airports (id)
   ON DELETE CASCADE
   ON UPDATE CASCADE
   DEFERRABLE INITIALLY DEFERRED;

ALTER TABLE airports_airlines
   ADD COLUMN airline_fk INTEGER
   REFERENCES airlines (id)
   ON DELETE CASCADE
   ON UPDATE CASCADE
   DEFERRABLE INITIALLY DEFERRED;
```

Then in QGIS, you should set up two *one-to-many relations* as explained above:

- a relation between `airlines` table and the pivot table;
- and a second one between `airports` table and the pivot table.

An easier way to do it (only for PostgreSQL) is using the *Discover Relations* in *Project* ► *Properties* ► *Relations*. QGIS will automatically read all relations in your database and you only have to select the two you need. Remember to load the three tables in the QGIS project first.

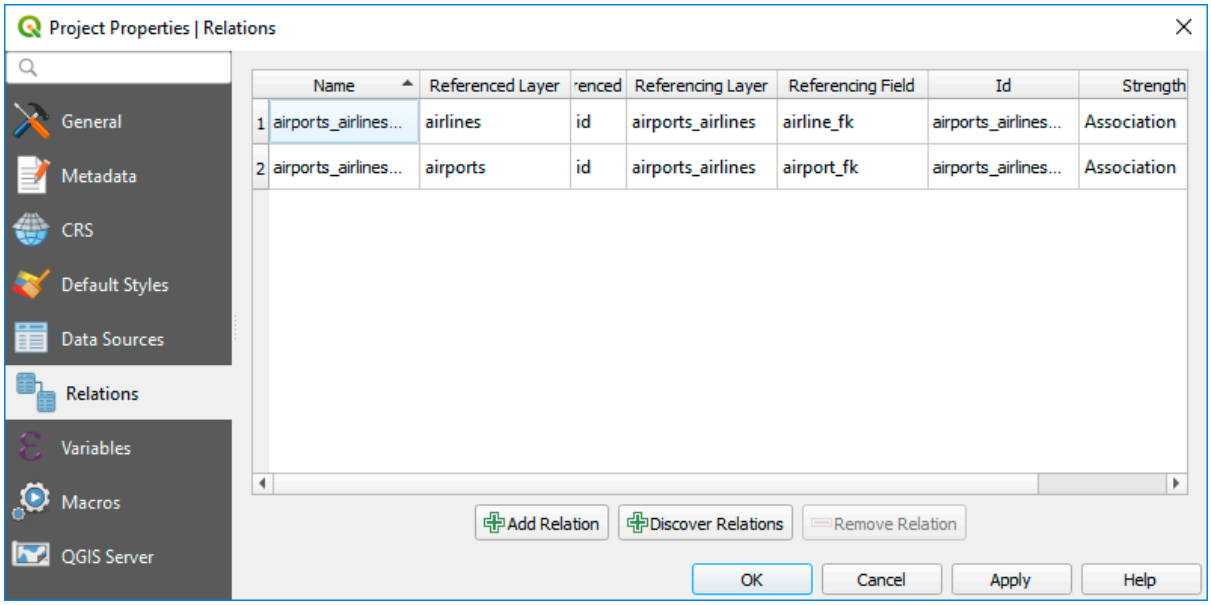


Fig. 16.107: Relations and autodiscover

In case you want to remove an `airport` or an `airline`, QGIS won't remove the associated record(s) in `airports_airlines` table. This task will be made by the database if we specify the right *constraints* in the pivot table creation as in the current example.

Notă: Combining N-M relation with automatic transaction group

You should enable the transaction mode in *Project Properties* ► *Data Sources* ► when working on such context. QGIS should be able to add or update row(s) in all tables (airlines, airports and the pivot tables).

Finally we have to select the right cardinality in the *Layer Properties* ► *Attributes Form* for the `airports` and `airlines` layers. For the first one we should choose the **airlines (id)** option and for the second one the **airports (id)** option.

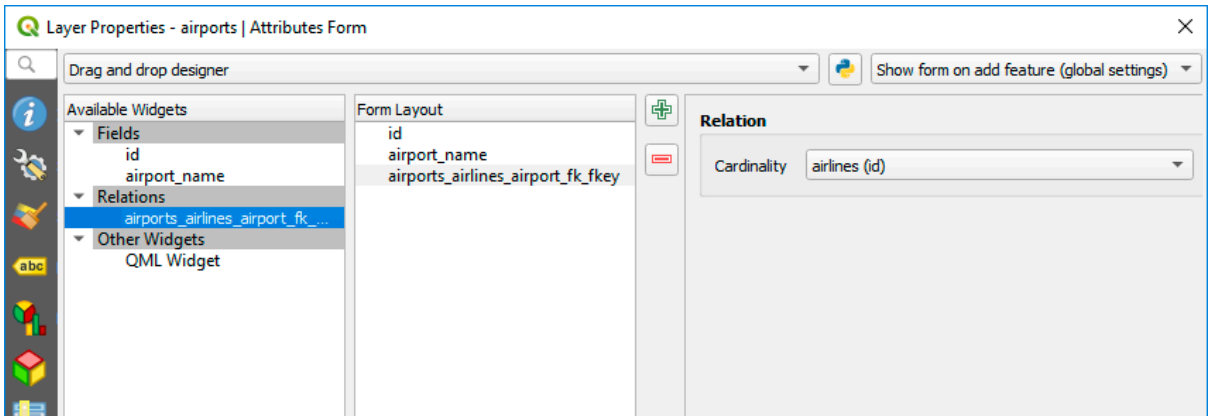


Fig. 16.108: Set relationship cardinality

Now you can associate an airport with an airline (or an airline with an airport) using *Add child feature* or *Link existing child feature* in the subforms. A record will automatically be inserted in the `airports_airlines` table.

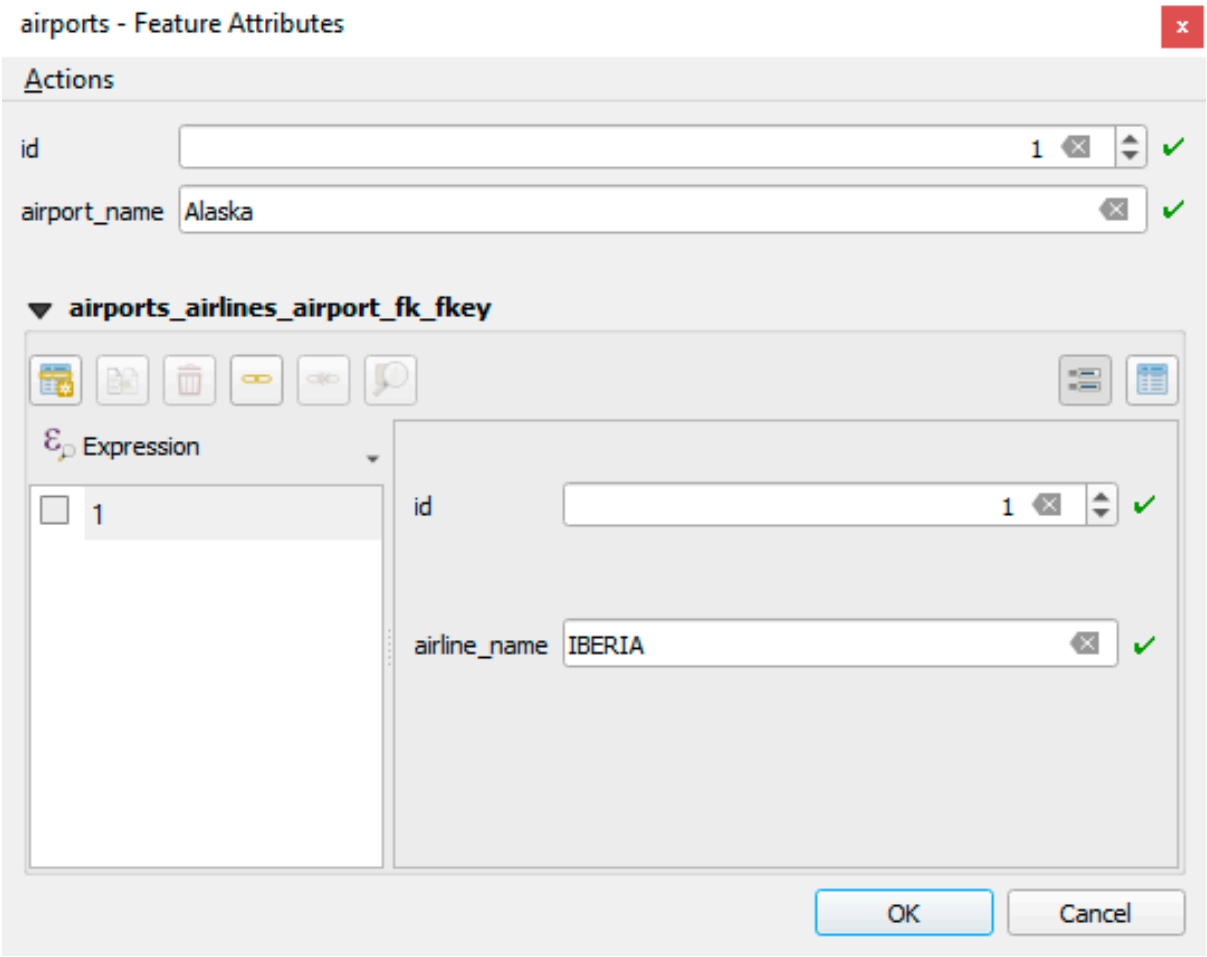


Fig. 16.109: N-M relationship between airports and airlines

Notă: Using **Many to one relation** cardinality

Sometimes hiding the pivot table in an N-M relationship is not desirable. Mainly because there are attributes in the relationship that can only have values when a relationship is established. If your tables have a geometry field, it could be interesting to activate the *On map identification* option (*Layer Properties* ► *Attributes Form* ► *Available widgets* ► *Fields*) for the foreign key fields in the pivot table.

Notă: **Pivot table primary key**

Avoid using multiple fields in the primary key in a pivot table. QGIS assumes a single primary key so a constraint like `constraint airports_airlines_pkey primary key (airport_fk, airline_fk)` will not work.

Polymorphic relations

The purpose

Polymorphic relations are special case of 1-N relations, where a single referencing (document) layer contains the features for multiple referenced layers. This differs from normal relations which require different referencing layer for each referenced layer. A single referencing (document) layer is achieved by adding an adiditonal `layer_field` column in the referencing (document) layer that stores information to identify the referenced layer. In its most simple form, the referencing (document) layer will just insert the layer name of the referenced layer into this field.

To be more precise, a polymorphic relation is a set of normal relations having the same referencing layer but having the referenced layer dynamically defined. The polymorphic setting of the layer is solved by using an expression which has to match some properties of the referenced layer like the table name, layer id, layer name.

Imagine we are going to the park and want to take pictures of different species of `plants` and `animals` we see there. Each plant or animal has multiple pictures associated with it, so if we use the normal 1:N relations to store pictures, we would need two separate tables, `animal_images` and `plant_images`. This might not be a problem for 2 tables, but imagine if we want to take separate pictures for mushrooms, birds etc.

Polymorphic relations solve this problem as all the referencing features are stored in the same table `documents`. For each feature the referenced layer is stored in the `referenced_layer` field and the referenced feature id in the `referenced_fk` field.

Defining polymorphic relations

First, let QGIS know about the polymorphic relations between the layers. This is done in *Project* ► *Properties…*. Open the *Relations* tab and click on the little down arrow next to the *Add Relation* button, so you can select the *Add Polymorphic Relation* option from the newly appeared dropdown.

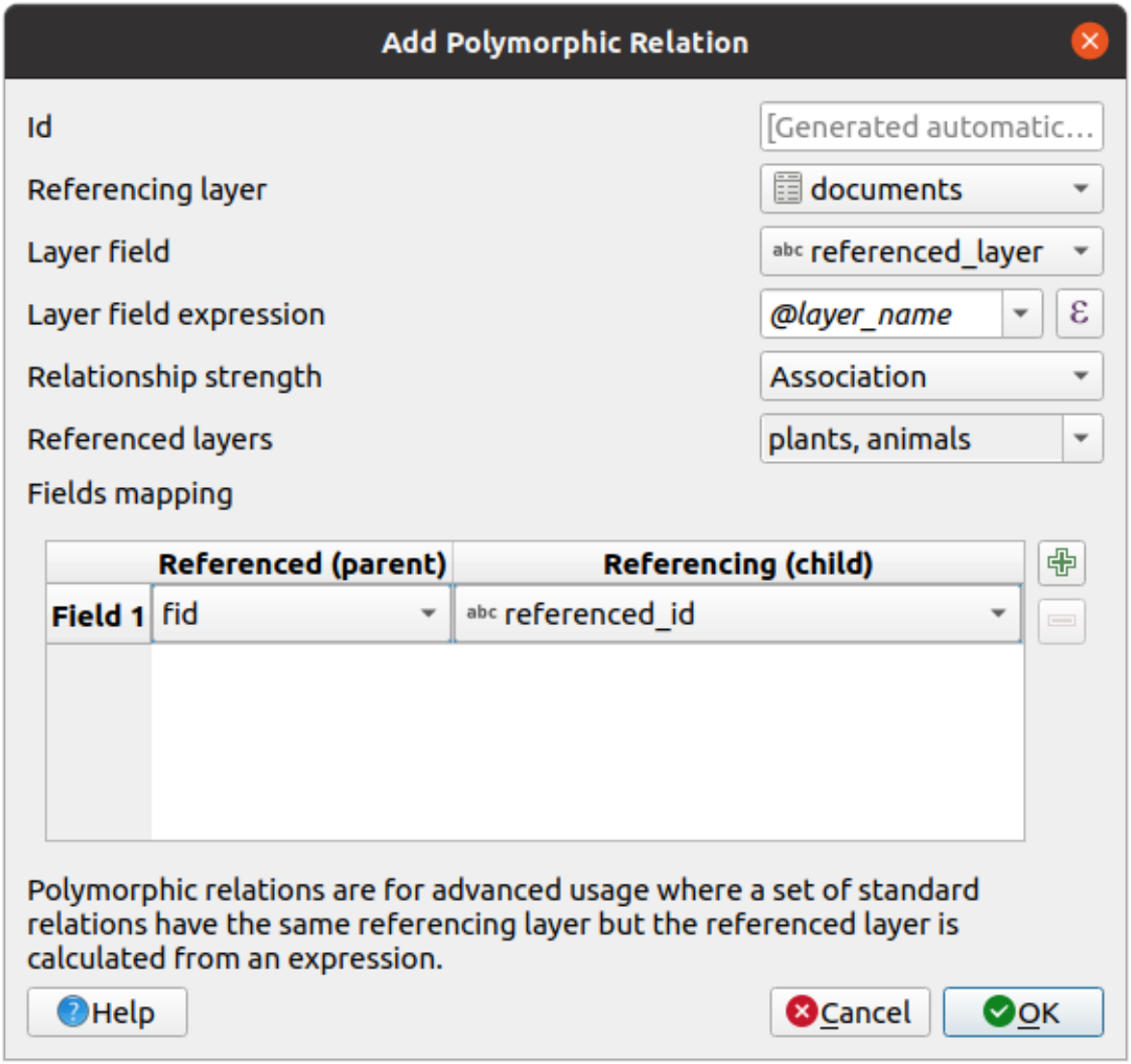


Fig. 16.110: Adding a polymorphic relation using `documents` layer as referencing and `animals` and `plants` as referenced layers.

- **Id** will be used for internal purposes and has to be unique. You may need it to build *custom forms*. If you leave it empty, one will be generated for you but you can assign one yourself to get one that is easier to handle
- **Referencing Layer (Child)** also considered as child layer, is the one with the foreign key field on it. In our case, this is the `documents` layer. For this layer you need to add a referencing field which points to the other layer, so this is `referenced_fk`.

Notă: Sometimes, you need more than a single field to uniquely identify features in a layer. Creating a relation with such a layer requires a **composite key**, i.e. more than a single pair of matching fields. Use the Add new field pair as part of a composite foreign key button to add as many pairs as necessary.

- **Layer Field** is the field in the referencing table that stores the result of the evaluated layer expression which is the referencing table that this feature belongs to. In our example, this would be the `referenced_layer` field.
- **Layer expression** evaluates to a unique identifier of the layer. This can be the layer name `@layer_name`, the layer id `@layer_id`, the layer's table name `decode_uri(@layer, 'table')` or anything that can uniquely identifies a layer.

- **Relationship strength** sets the strength of the generated relations between the parent and the child layer. The default *Association* type means that the parent layer is *simply* linked to the child one while the *Composition* type allows you to duplicate also the child features when duplicating the parent ones and on deleting a feature the children are deleted as well, resulting in cascade over all levels (means children of children of… are deleted as well).
- **Referenced Layers** also considered as parent layers, are those with the primary key, pointed to, so here they would be `plants` and `animals` layers. You need to define the primary key of the referenced layers from the dropdown, so it is `fid`. Note that the definition of a valid primary key requires all the referenced layers to have a field with that name. If there is no such field you cannot save a polymorphic relation.

Once added, the polymorphic relation can be edited via the *Edit Polymorphic Relation* menu entry.

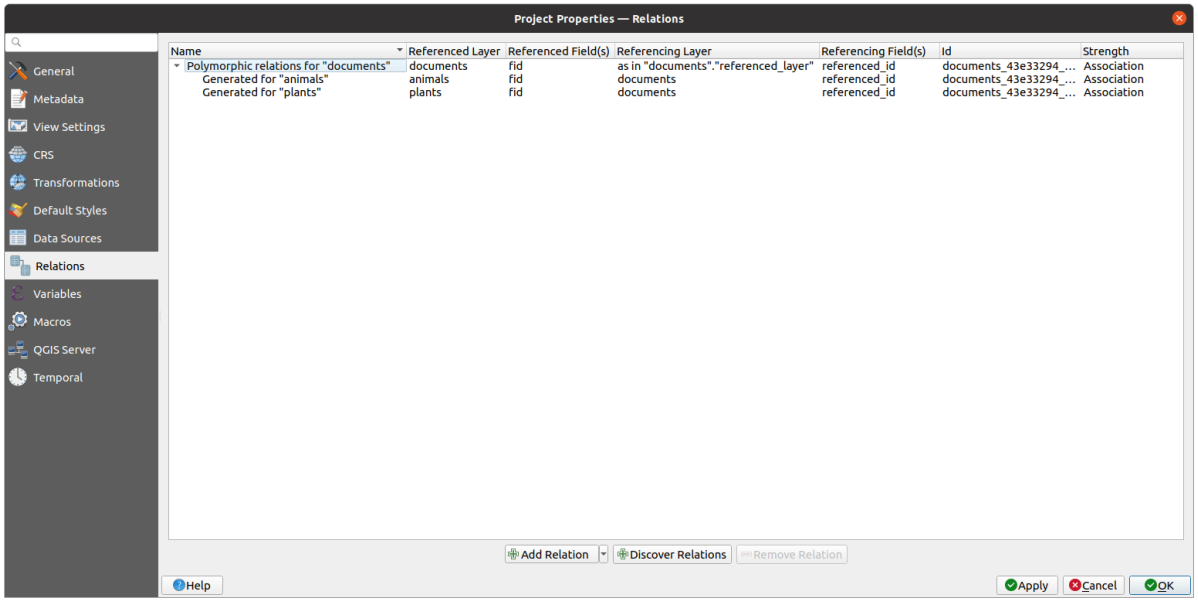


Fig. 16.111: Preview of the newly created polymorphic relation and its child relations for animals and plants.

The example above uses the following database schema:

```
CREATE SCHEMA park;

CREATE TABLE park.animals
(
   fid serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
   animal_species text NOT NULL,
   CONSTRAINT animals_pkey PRIMARY KEY (fid)
);

CREATE INDEX animals_geom_idx ON park.animals USING gist (geom);

CREATE TABLE park.plants
(
   fid serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
   plant_species text NOT NULL,
   CONSTRAINT plants_pkey PRIMARY KEY (fid)
);

CREATE INDEX plants_geom_idx ON park.plants USING gist (geom);

CREATE TABLE park.documents
```

(continues on next page)

(continuare din pagina precedentă)

```
(
   fid serial NOT NULL,
   referenced_layer text NOT NULL,
   referenced_fk integer NOT NULL,
   image_filename text NOT NULL,
   CONSTRAINT documents_pkey PRIMARY KEY (fid)
);
```

CAPITOLUL 17

Lucrul cu Datele Raster

17.1 Dialogul Proprietăților Rasterului

Raster data is made up of pixels (or cells), and each pixel has a value. It is commonly used to store various types of data, including:

- Imagery, such as satellite images, digital aerial photographs, scanned maps
- Elevation data, such as digital elevation models (DEMs), digital terrain models (DTMs)
- Other types of data, such as land cover, soil types, rainfall and many others.

Raster data can be stored in several supported formats, including GeoTIFF, ERDAS Imagine, ArcInfo ASCII GRID, PostGIS Raster and others. See more at *Deschiderea datelor*.

To view and set the properties for a raster layer, double click on the layer name in the map legend, or right click on the layer name and choose *Properties* from the context menu. This will open the *Raster Layer Properties* dialog.

There are several tabs in the dialog:

Information	*Source*	*Symbology*[1]
Transparency[1]	*Histogram*[1]	*Rendering*
Temporal	*Pyramids*	*Elevation*
Metadata	*Legend*	*Display*
QGIS Server	*External plugins*[2] tabs	

[1] Also available in the *Layer styling panel*

[2] *External plugins* you install can optionally add tabs to this dialog. Those are not presented in this document. Refer to their documentation.

Sfat: Live update rendering

The *Layer Styling Panel* provides you with some of the common features of the Layer properties dialog and is a good modeless widget that you can use to speed up the configuration of the layer styles and view your changes on the map canvas.

Notă: Because properties (symbology, label, actions, default values, forms…) of embedded layers (see *Embedding layers from external projects*) are pulled from the original project file, and to avoid changes that may break this behavior, the layer properties dialog is made unavailable for these layers.

17.1.1 Information Properties

The *Information* tab is read-only and represents an interesting place to quickly grab summarized information and metadata for the current layer. Provided information are:

- general such as name in the project, source path, list of auxiliary files, last save time and size, the used provider
- based on the provider of the layer: extent, width and height, data type, GDAL driver, bands statistics
- the Coordinate Reference System: name, units, method, accuracy, reference (i.e. whether it's static or dynamic)
- read from layer properties: data type, extent, width/height, compression, pixel size, statistics on bands, number of columns, rows and no-data values of the raster…
- picked from the *filled metadata*: access, extents, links, contacts, history…

17.1.2 Proprietățile Sursei

The *Source* tab displays basic information about the selected raster, including:

- the *Layer name* to display in the *Layers Panel*;
- the *Coordinate Reference System*: Displays the layer's *Coordinate Reference System (CRS)*. You can change the layer's CRS, by selecting a recently used one in the drop-down list or clicking on the Select CRS button (see *Coordinate Reference System Selector*). Use this process only if the layer CRS is a wrong or not specified. If you wish to reproject your data, use a reprojection algorithm from Processing or *Save it as new dataset*.

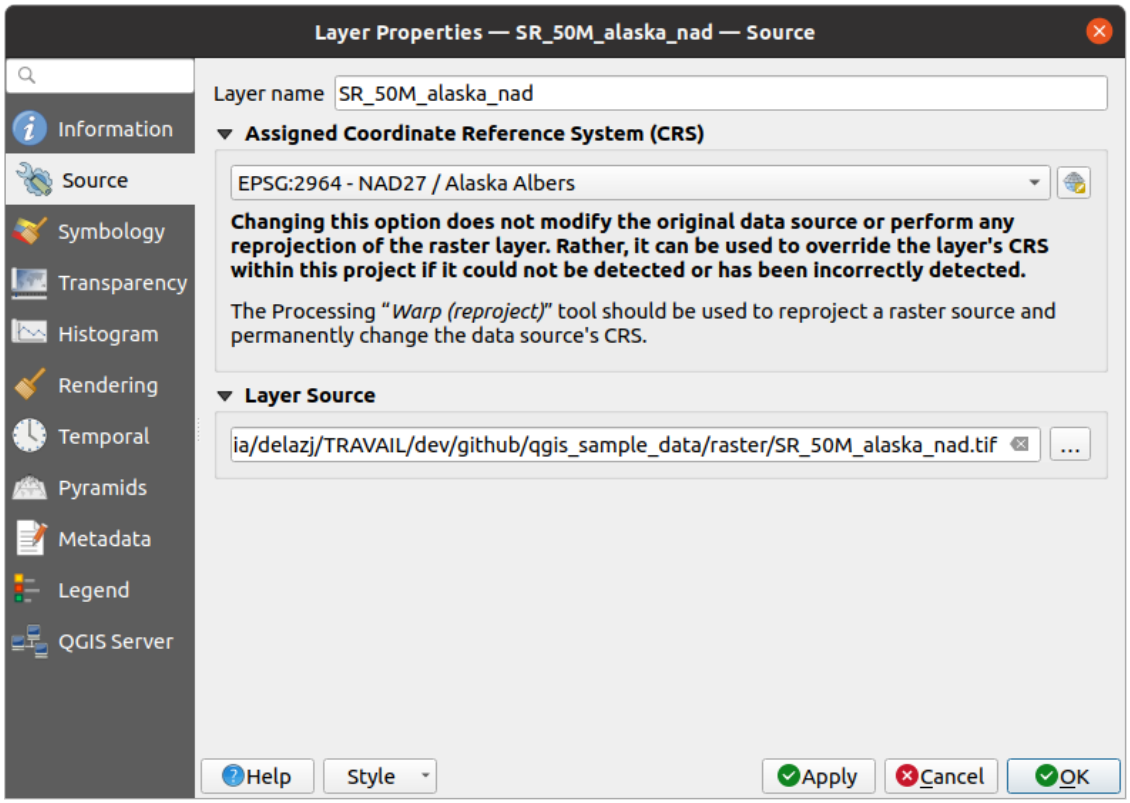


Fig. 17.1: Raster Layer Properties - Source Dialog

17.1.3 Symbology Properties

The raster layer symbology tab is made of three different sections:

- The *Band rendering* where you can control the renderer type to use
- The *Layer rendering* to apply effects on rendered data
- The *Resampling* methods to optimize rendering on map

Randare bandă

QGIS offers many different *Render types*. The choice of renderer depends on the data type and the information you'd like to highlight.

1. *Multiband color* - if the file comes with several bands (e.g. a satellite image with several bands).
2. *Paletted/Unique values* - for single band files that come with an indexed palette (e.g. a digital topographic map) or for general use of palettes for rendering raster layers.
3. *Singleband gray* - (one band of) the image will be rendered as gray. QGIS will choose this renderer if the file is neither multiband nor paletted (e.g. a shaded relief map).
4. *Singleband pseudocolor* - this renderer can be used for files with a continuous palette or color map (e.g. an elevation map).
5. *Hillshade* - Creates hillshade from a band.
6. *Contours* - Generates contours on the fly for a source raster band.

Multiband color

With the multiband color renderer, three selected bands from the image will be used as the red, green or blue component of the color image. QGIS automatically fetches *Min* and *Max* values for each band of the raster and scales the coloring accordingly. You can control the value ranges in the *Min/Max Value Settings* section.

A *Contrast enhancement* method can be applied to the values: «No enhancement», «Stretch to MinMax», «Stretch and clip to MinMax» and «Clip to min max».

Notă: Îmbunătăţirea contrastului

When adding GRASS rasters, the option *Contrast enhancement* will always be set automatically to *stretch to min max*, even if this is set to another value in the QGIS general options.

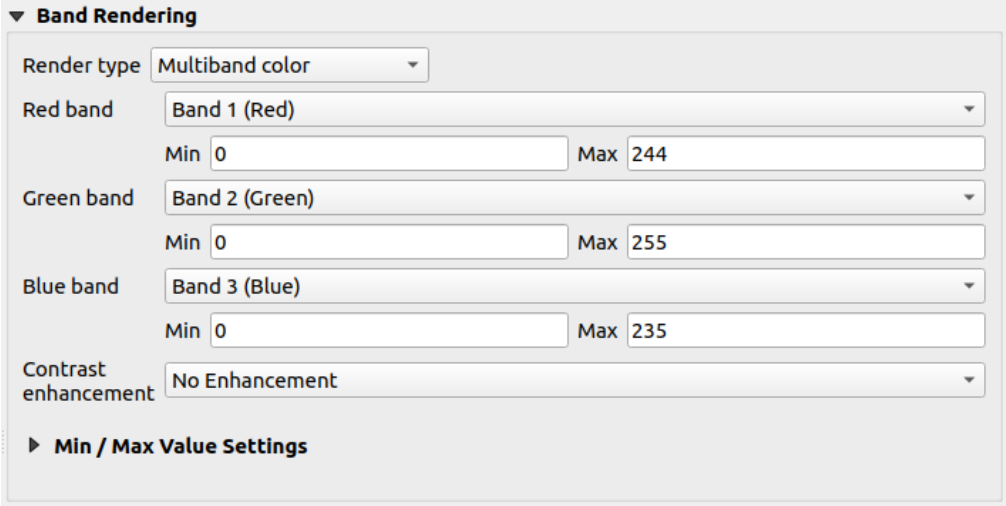


Fig. 17.2: Raster Symbology - Multiband color rendering

Sfat: Vizualizarea unei singure benzi dintr-un Raster Multibandă

If you want to view a single band of a multiband image (for example, Red), you might think you would set the Green and Blue bands to *Not Set*. But the preferred way of doing this is to set the image type to *Singleband gray*, and then select Red as the *Gray band* to use.

Paletted/Unique values

This is the standard render option for singleband files that include a color table, where a certain color is assigned to each pixel value. In that case, the palette is rendered automatically.

It can be used for all kinds of raster bands, assigning a color to each unique raster value.

If you want to change a color, just double-click on the color and the *Select color* dialog appears.

It is also possible to assign labels to the colors. The label will then appear in the legend of the raster layer.

Right-clicking over selected rows in the color table shows a contextual menu to:

- *Change Color…* for the selection
- *Change Opacity…* for the selection
- *Change Label…* for the selection

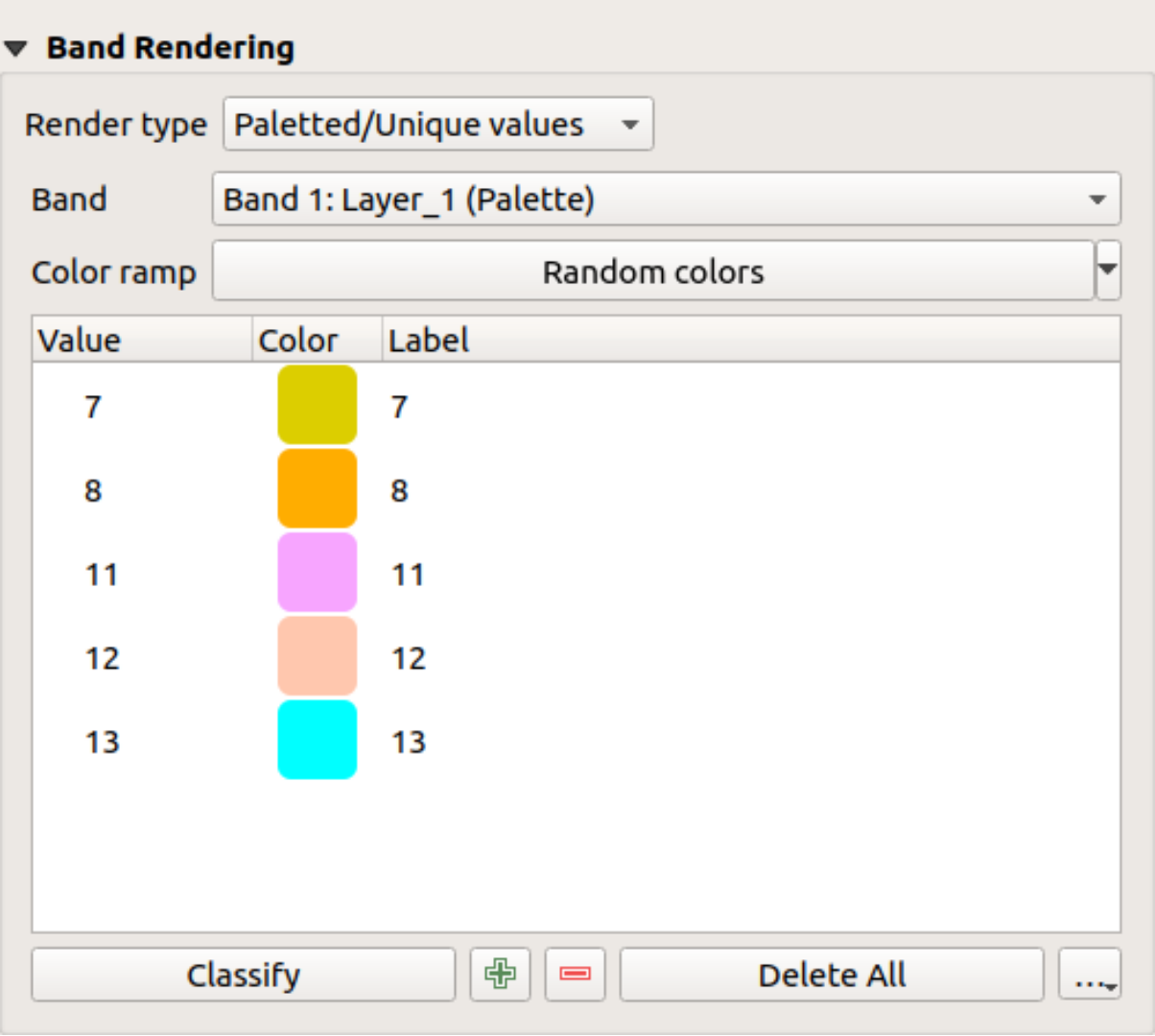


Fig. 17.3: Raster Symbology - Paletted unique value rendering

The pulldown menu, that opens when clicking the ... (Advanced options) button below the color map to the right, offers color map loading (*Load Color Map from File...*) and exporting (*Export Color Map to File...*), and loading of classes (*Load Classes from Layer*).

Singleband gray

This renderer allows you to render a layer using only one band with a *Color gradient*: «Black to white» or «White to black». You can change the range of values to color (*Min* and *Max*) in the *Min/Max Value Settings*.

A *Contrast enhancement* method can be applied to the values: «No enhancement», «Stretch to MinMax», «Stretch and clip to MinMax» and «Clip to min max».

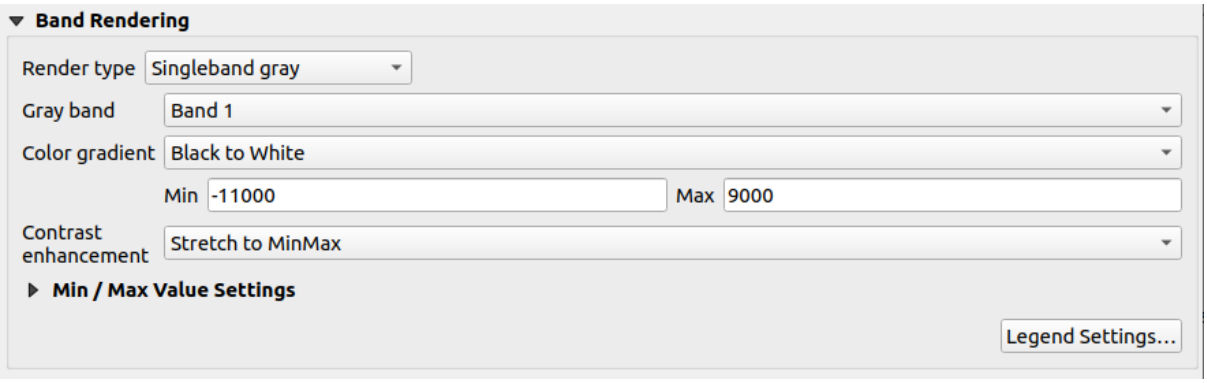


Fig. 17.4: Raster Symbology - Singleband gray rendering

Pixels are assigned a color based on the selected color gradient and the layer's legend (in the *Layers* panel and the layout *legend item*) is displayed using a continuous color ramp. Press *Legend settings...* if you wish to tweak the settings. More details at *Customize raster legend*.

Singleband pseudocolor

This is a render option for single-band files that include a continuous palette. You can also create color maps for a band of a multiband raster.

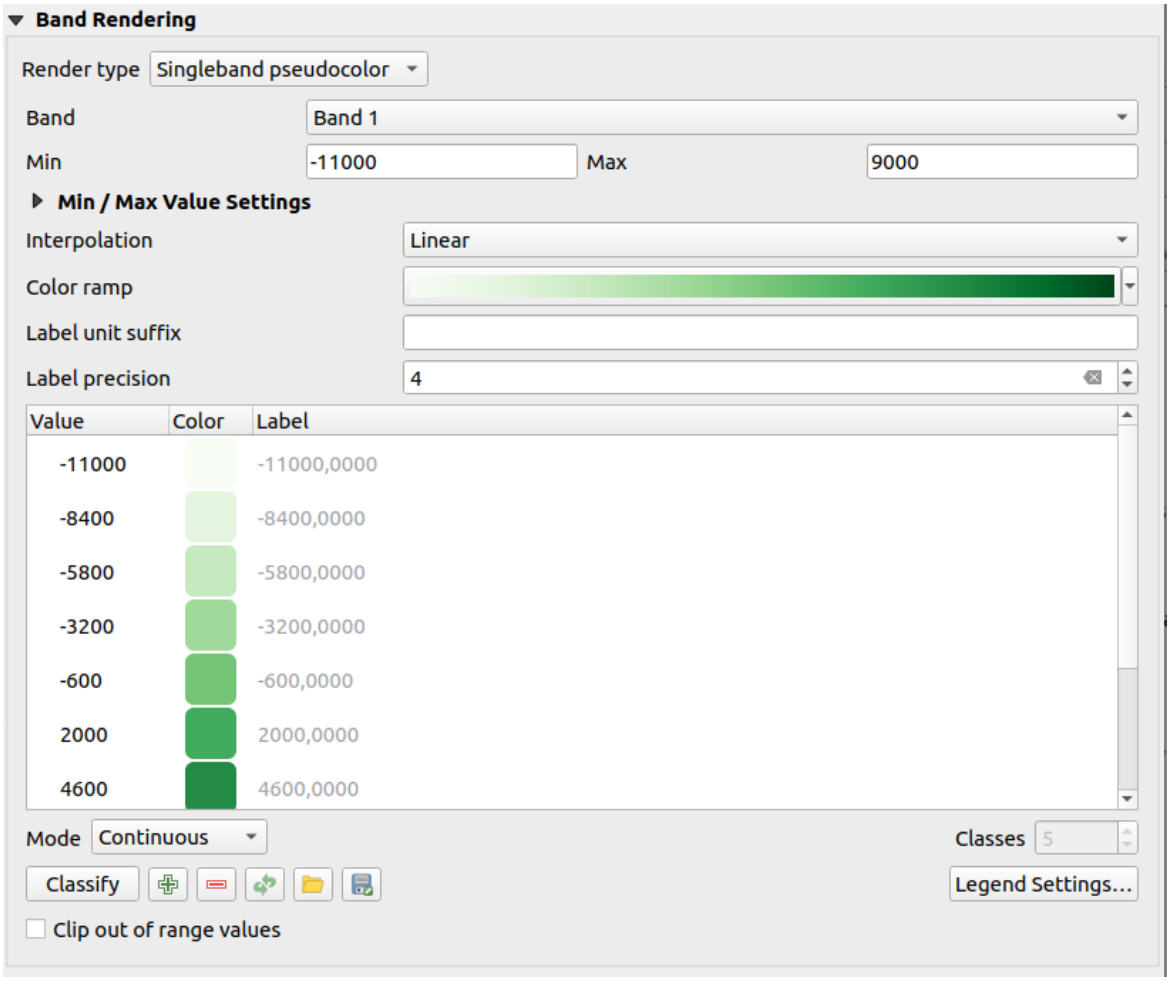


Fig. 17.5: Raster Symbology - Singleband pseudocolor rendering

Using a *Band* of the layer and a *values range*, you can now interpolate and assign representation color to pixels within classes. More at *Color ramp shader classification*.

Pixels are assigned a color based on the selected color ramp and the layer's legend (in the *Layers* panel and the layout *legend item*) is displayed using a continuous color ramp. Press *Legend settings...* if you wish to tweak the settings or instead use a legend with separated classes (and colors). More details at *Customize raster legend*.

Hillshade

Render a band of the raster layer using hillshading.

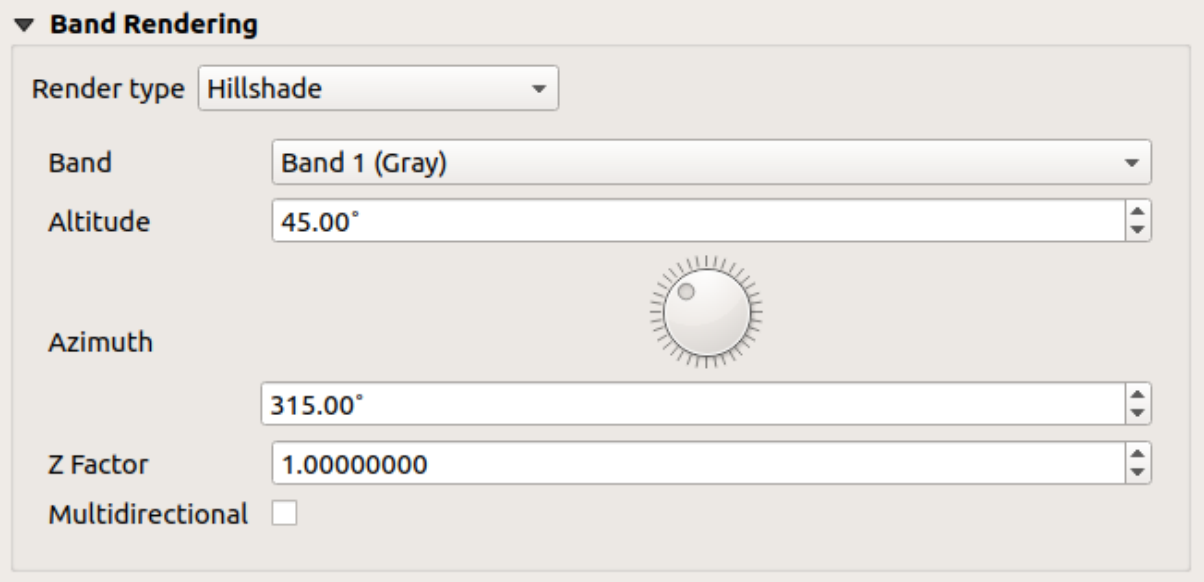


Fig. 17.6: Raster Symbology - Hillshade rendering

Opţiuni:

- *Band*: The raster band to use.
- *Altitude*: The elevation angle of the light source (default is `45°`).
- *Azimuth*: The azimuth of the light source (default is `315°`).
- *Z Factor*: Scaling factor for the values of the raster band (default is `1`).
- ☑ *Multidirectional*: Specify if multidirectional hillshading is to be used (default is `off`).

Contours

This renderer draws contour lines that are calculated on the fly from the source raster band.

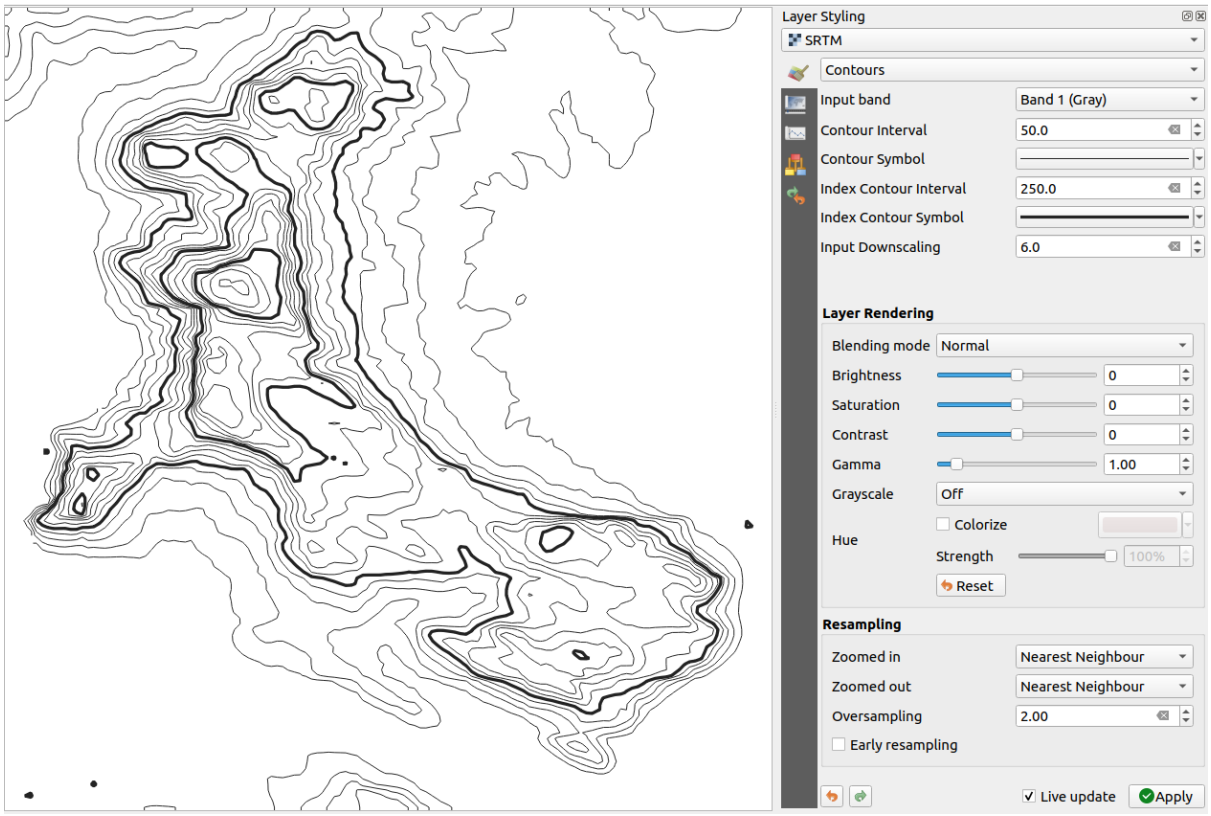


Fig. 17.7: Raster Symbology - Contours rendering

Opţiuni:

- *Input band*: the raster band to use.
- *Contour interval*: the distance between two consecutive contour lines
- *Contour symbol*: the *symbol* to apply to the common contour lines.
- *Index contour interval*: the distance between two consecutive **index contours**, that is the lines shown in a distinctive manner for ease of identification, being commonly printed more heavily than other contour lines and generally labeled with a value along its course.
- *Index contour symbol*: the symbol to apply to the index contour lines
- *Input downscaling*: Indicates by how much the renderer will scale down the request to the data provider (default is `4.0`).

  For example, if you generate contour lines on input raster block with the same size as the output raster block, the generated lines would contain too much detail. This detail can be reduced by the „downscale” factor, requesting lower resolution of the source raster. For a raster block 1000x500 with downscale 10, the renderer will request raster 100x50 from provider. Higher downscale makes contour lines more simplified (at the expense of losing some detail).

Setting the min and max values

By default, QGIS reports the *Min* and *Max* values of the band(s) of the raster. A few very low and/or high values can have a negative impact on the rendering of the raster. The *Min/Max Value Settings* frame helps you control the rendering.

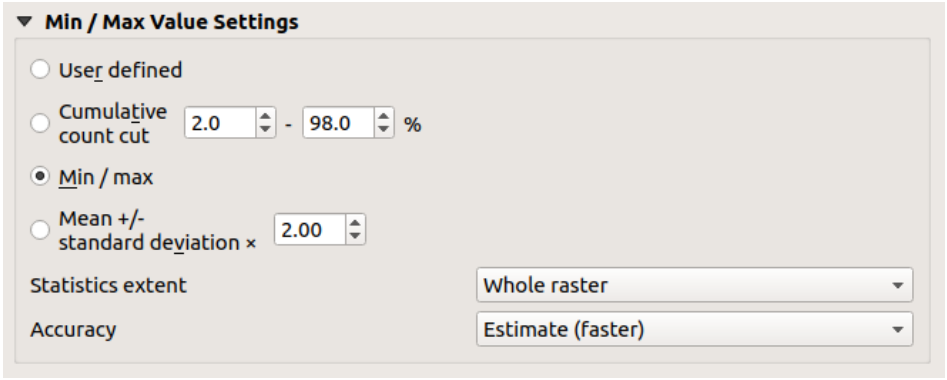


Fig. 17.8: Raster Symbology - Min and Max Value Settings

Available options are:

- *User defined*: The default *Min* and *Max* values of the band(s) can be overridden
- *Cumulative count cut*: Removes outliers. The standard range of values is `2%` to `98%`, but it can be adapted manually.
- *Min / max*: Uses the whole range of values in the image band.
- *Mean +/- standard deviation x*: Creates a color table that only considers values within the standard deviation or within multiple standard deviations. This is useful when you have one or two cells with abnormally high values in a raster layer that impact the rendering of the raster negatively.

Calculations of the min and max values of the bands are made based on the:

- *Statistics extent*: it can be *Whole raster*, *Current canvas* or *Updated canvas*. *Updated canvas* means that min/max values used for the rendering will change with the canvas extent (dynamic stretching).
- *Accuracy*, which can be either *Estimate (faster)* or *Actual (slower)*.

Notă: For some settings, you may need to press the *Apply* button of the layer properties dialog in order to display the actual min and max values in the widgets.

Color ramp shader classification

This method can be used to classify and represent scalar dataset (raster or mesh contour) based on their values. Given a *color ramp* and a number of classes, it generates intermediate color map entries for class limits. Each color is mapped with a value interpolated from a range of values and according to a classification mode. The scalar dataset elements are then assigned their color based on their class.

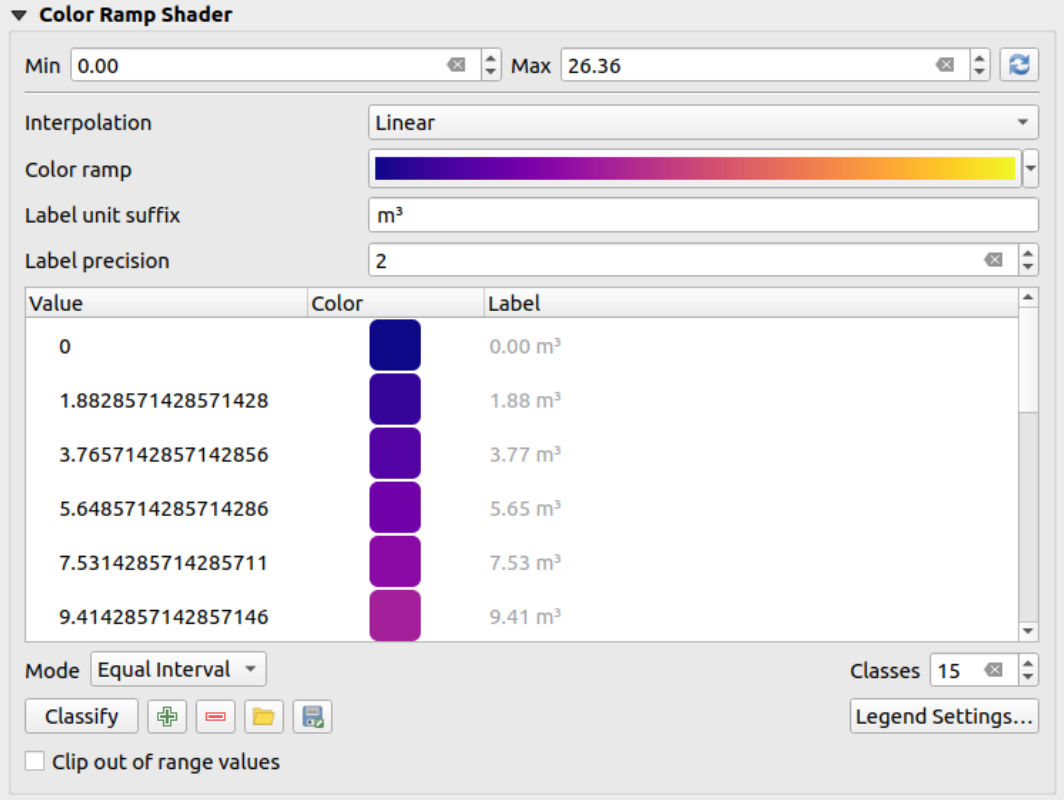


Fig. 17.9: Classifying a dataset with a color ramp shader

1. A *Min* and *Max* values must be defined and used to interpolate classes bounds. By default QGIS detects them from the dataset but they can be modified.
2. The *Interpolation* entry defines how scalar elements are assigned their color :
   - *Discrete* (a <= symbol appears in the header of the *Value* column): The color is taken from the closest color map entry with equal or higher value
   - *Linear*: The color is linearly interpolated from the color map entries above and below the pixel value, meaning that to each dataset value corresponds a unique color
   - *Exact* (a = symbol appears in the header of the *Value* column): Only pixels with value equal to a color map entry are applied a color; others are not rendered.
3. The *Color ramp* widget helps you select the color ramp to assign to the dataset. As usual with *this widget*, you can create a new one and edit or save the currently selected one. The name of the color ramp will be saved in the configuration.
4. The *Label unit suffix* adds a label after the value in the legend, and the *Label precision* controls the number of decimals to display.
5. The classification *Mode* helps you define how values are distributed across the classes:
   - *Equal interval*: Provided the *Number of classes*, limits values are defined so that the classes all have the same magnitude.
   - *Continuous*: Classes number and color are fetched from the color ramp stops; limits values are set following stops distribution in the color ramp.
   - *Quantile*: Provided the *Number of classes*, limits values are defined so that the classes have the same number of elements. Not available with *mesh layers*.
6. You can then *Classify* or tweak the classes:
   - The button Add values manually adds a value to the table.

- The button Remove selected row deletes selected values from the table.
- Double clicking in the *Value* column lets you modify the class value.
- Double clicking in the *Color* column opens the dialog *Change color*, where you can select a color to apply for that value.
- Double clicking in the *Label* column to modify the label of the class, but this value won't be displayed when you use the identify feature tool.
- Right-clicking over selected rows in the color table shows a contextual menu to *Change Color…* and *Change Opacity…* for the selection.

You can use the buttons Load color map from file or Export color map to file to load an existing color table or to save the color table for later use.

7. With linear *Interpolation*, you can also configure:

   - *Clip out of range values*: By default, the linear method assigns the first class (respectively the last class) color to values in the dataset that are lower than the set *Min* (respectively greater than the set *Max*) value. Check this setting if you do not want to render those values.
   - *Legend settings*, for display in the *Layers* panel and the layout *legend item*. More details at *Customize raster legend*.

Customize raster legend

When applying a color ramp to a raster or a mesh layer, you may want to display a legend showing the classification. By default, QGIS displays a continuous color ramp with min and max values in the *Layers* panel and the layout *legend item*. This can be customized using the *Legend settings* button in the classification widget.

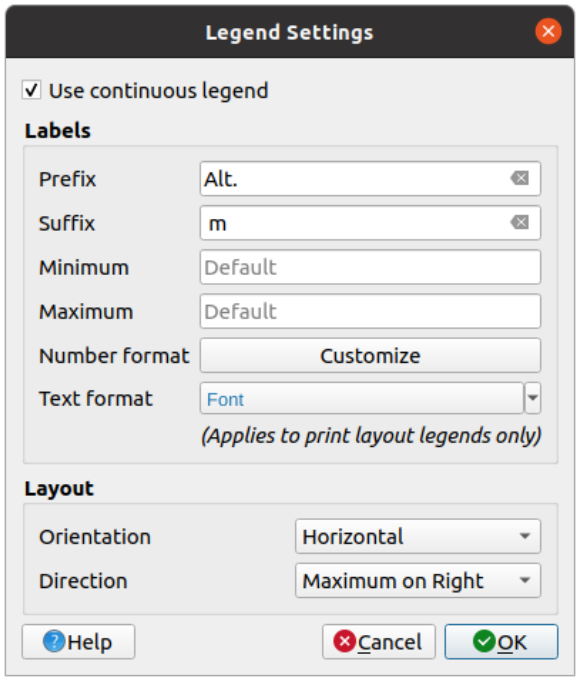


Fig. 17.10: Modifying a raster legend

In this dialog, you can set whether to *Use continuous legend*: if unchecked, the legend displays separated colors corresponding to the different classes applied. This option is not available for raster *singleband gray* symbology.

Checking the *Use continuous legend* allows you to configure both the labels and layout properties of the legend.

Labels

- Add a *Prefix* and a *Suffix* to the labels
- Modify the *Minimum* and a *Maximum* values to show in the legend
- *Customize* the *Number format*
- *Customize* the *Text format* to use in the print layout legend.

Layout

- Control the *Orientation* of the legend color ramp; it can be **Vertical** or **Horizontal**
- Control the *Direction* of the values depending on the orientation:
  - If vertical, you can display the **Maximum on top** or the **Minimum on top**
  - If horizontal, you can display the **Maximum on right** or the **Minimum on right**

Randarea stratului

Over the symbology type applied to the layer band(s), you can achieve special rendering effects for the whole raster file(s):

- Use one of the blending modes (see *Modurile de Fuziune*)
- Set custom *Brightness*, *Saturation*, *Gamma* and *Contrast* to colors.
- With the *Invert colors*, the layer is rendered with opposite colors. Handy, for example, to switch out-of-the box OpenStreetMap tiles to dark mode.
- Turn the layer to *Grayscale* option either «By lightness», «By luminosity» or «By average».
- *Colorize* and adjust the *Strength* of *Hue* in the color table

Press *Reset* to remove any custom changes to the layer rendering.

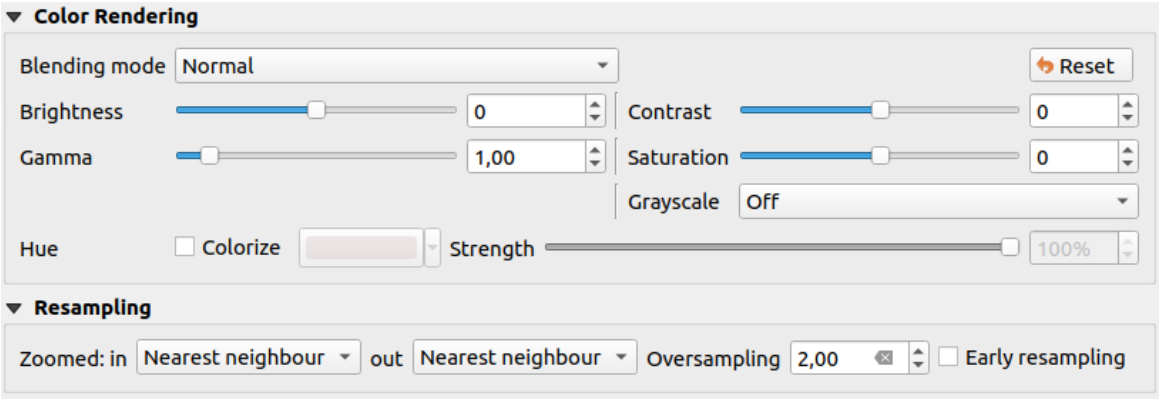


Fig. 17.11: Raster Symbology - Layer rendering and Resampling settings

Reeşantionare

The *Resampling* option has effect when you zoom in and out of an image. Resampling modes can optimize the appearance of the map. They calculate a new gray value matrix through a geometric transformation.

When applying the «Nearest neighbour» method, the map can get a pixelated structure when zooming in. This appearance can be improved by using the «Bilinear (2x2 kernel)» or «Cubic (4x4 kernel)» method, which cause sharp edges to be blurred. The effect is a smoother image. This method can be applied to for instance digital topographic raster maps.

Early resampling: allows to calculate the raster rendering at the provider level where the resolution of the source is known, and ensures a better zoom in rendering with QGIS custom styling. Really convenient for tile rasters loaded using an *interpretation method*.

17.1.4 Transparency Properties

QGIS provides capabilities to set the *Transparency* level of a raster layer.

Use the *Global opacity* slider to set to what extent the underlying layers (if any) should be visible through the current raster layer. This is very useful if you overlay raster layers (e.g., a shaded relief map overlayed by a classified raster map). This will make the look of the map more three dimensional. The opacity of the raster can be data-defined, and vary e.g. depending on the visibility of another layer, by temporal variables, on different pages of an atlas, …

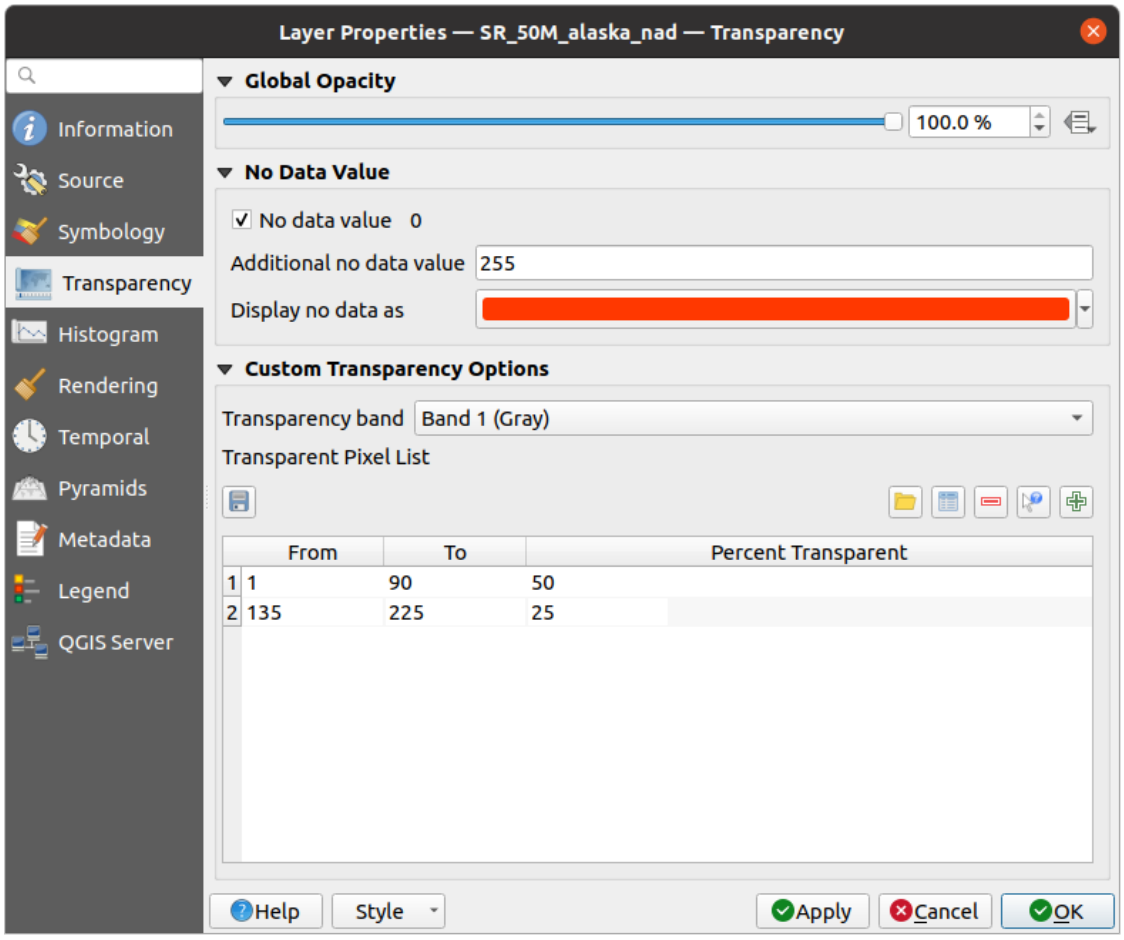


Fig. 17.12: Raster Transparency

With *No data value* QGIS reports the original source no data value (if defined) which you can consider as is in the rendering. Additionally, you can enter a raster value that should be treated as an *Additional no data value*. The

Display no data as color selector allows you to apply a custom color to no data pixels, instead of the default transparent rendering.

An even more flexible way to customize the transparency is available in the *Custom transparency options* section:

- Use *Transparency band* to apply transparency for an entire band.
- Provide a list of pixels to make transparent with corresponding levels of transparency:

  1. Clic pe butonul Add values manually. Un nou rând va apărea în lista pixelilor.
  2. Enter the **Red**, **Green** and **Blue** values of the pixel and adjust the **Percent Transparent** to apply.
  3. Alternatively, you can fetch the pixel values directly from the raster using the Add values from display button. Then enter the transparency value.
  4. Repeat the steps to adjust more values with custom transparency.
  5. Press the *Apply* button and have a look at the map.

  După cum se poate vedea, deși este destul de ușor de stabilit transparența după dorință, trebuie parcurși o mulțime de pași. Prin urmare, puteți utiliza butonul Export în fișier pentru a salva setările de transparență într-un fișier. Butonul de Import din fișier încarcă setările de transparență și le aplică stratului raster curent.

17.1.5 Histogram Properties

The *Histogram* tab allows you to view the distribution of the values in your raster. The histogram is generated when you press the *Compute Histogram* button. All existing bands will be displayed together. You can save the histogram as an image with the button.

At the bottom of the histogram, you can select a raster band in the drop-down menu and *Set min/max style for* it. The *Prefs/Actions* drop-down menu gives you advanced options to customize the histogram:

- With the *Visibility* option, you can display histograms for individual bands. You will need to select the option *Show selected band*.
- The *Min/max options* allow you to «Always show min/max markers», to «Zoom to min/max» and to «Update style to min/max».
- The *Actions* option allows you to «Reset» or «Recompute histogram» after you have changed the min or max values of the band(s).

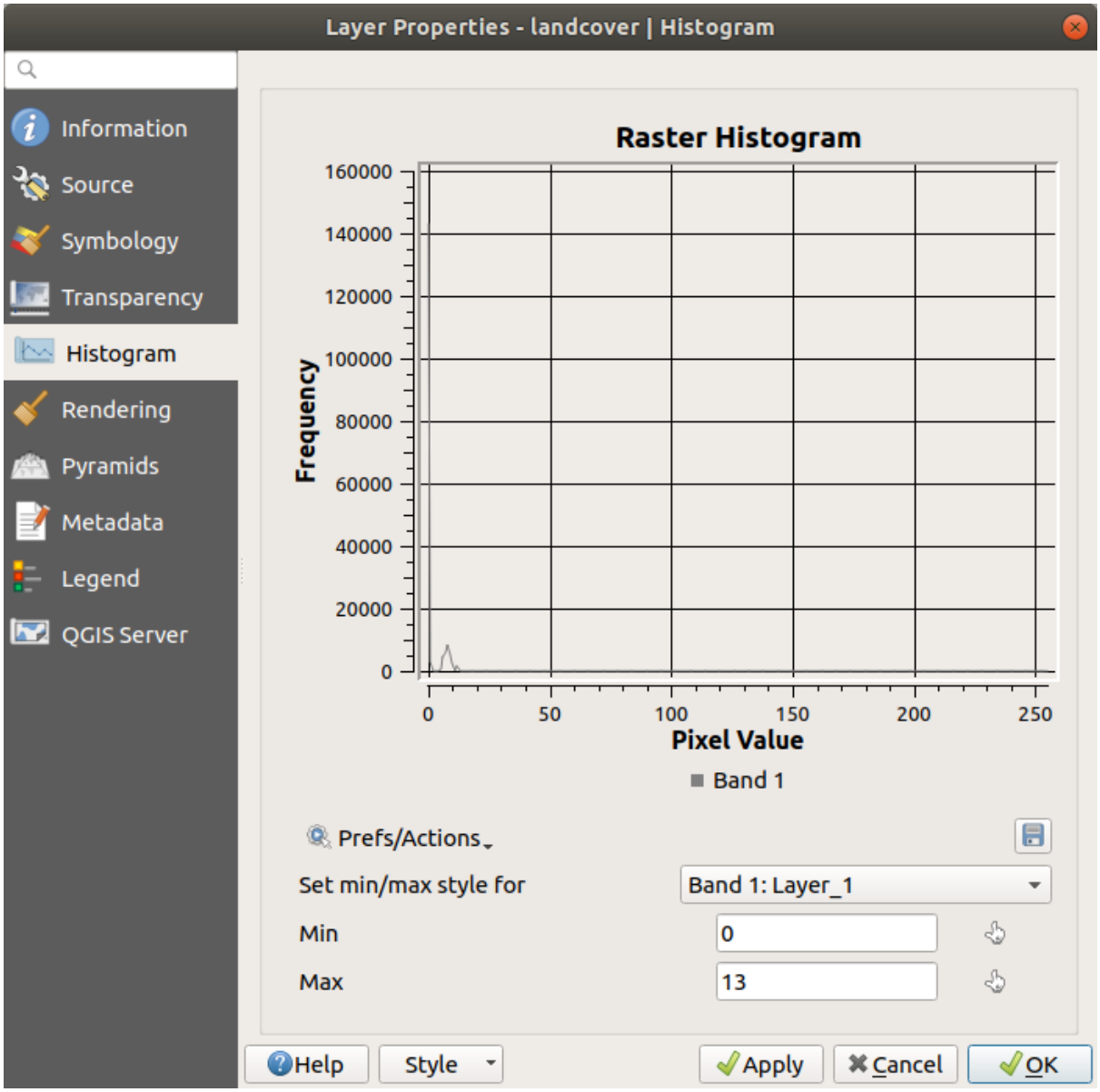


Fig. 17.13: Histograma Raster

17.1.6 Rendering Properties

In the *Rendering* tab, it's possible to:

- set *Scale dependent visibility* for the layer: You can set the *Maximum (inclusive)* and *Minimum (exclusive)* scales, defining a range of scales in which the layer will be visible. It will be hidden outside this range. The Set to current canvas scale button helps you use the current map canvas scale as a boundary. See *Visibility Scale Selector* for more information.

  Notă: You can also activate scale dependent visibility on a layer from within the *Layers* panel: right-click on the layer and in the contextual menu, select *Set Layer Scale Visibility*.

- *Refresh layer at interval*: controls whether and how regular a layer can be refreshed. Available *Configuration* options are:

- *Reload data*: the layer will be completely refreshed. Any cached data will be discarded and refetched from the provider. This mode may result in slower map refreshes.
- *Redraw layer only*: this mode is useful for animation or when the layer's style will be updated at regular intervals. Canvas updates are deferred in order to avoid refreshing multiple times if more than one layer has an auto update interval set.

It is also possible to set the *Interval (seconds)* between consecutive refreshments.

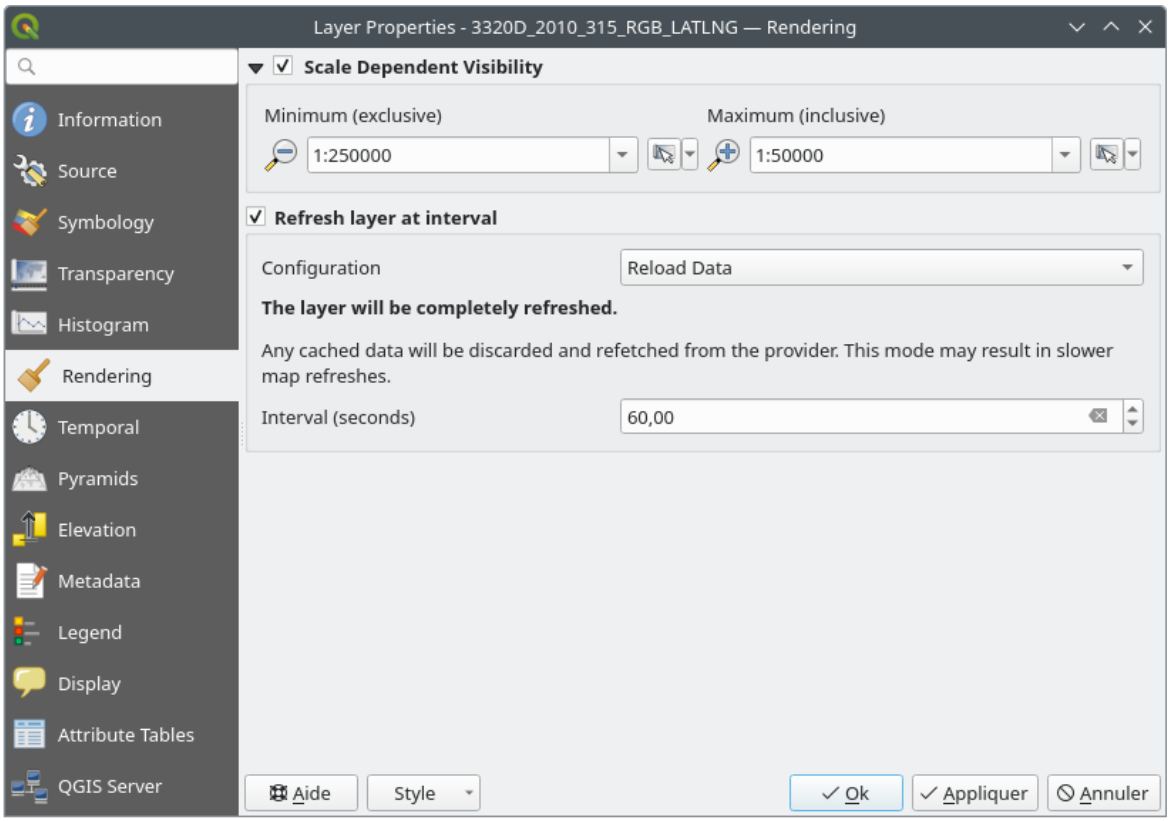


Fig. 17.14: Raster Rendering Properties

17.1.7 Proprietăți Temporale

The *Temporal* tab provides options to control the rendering of the layer over time. Such dynamic rendering requires the *temporal navigation* to be enabled over the map canvas.

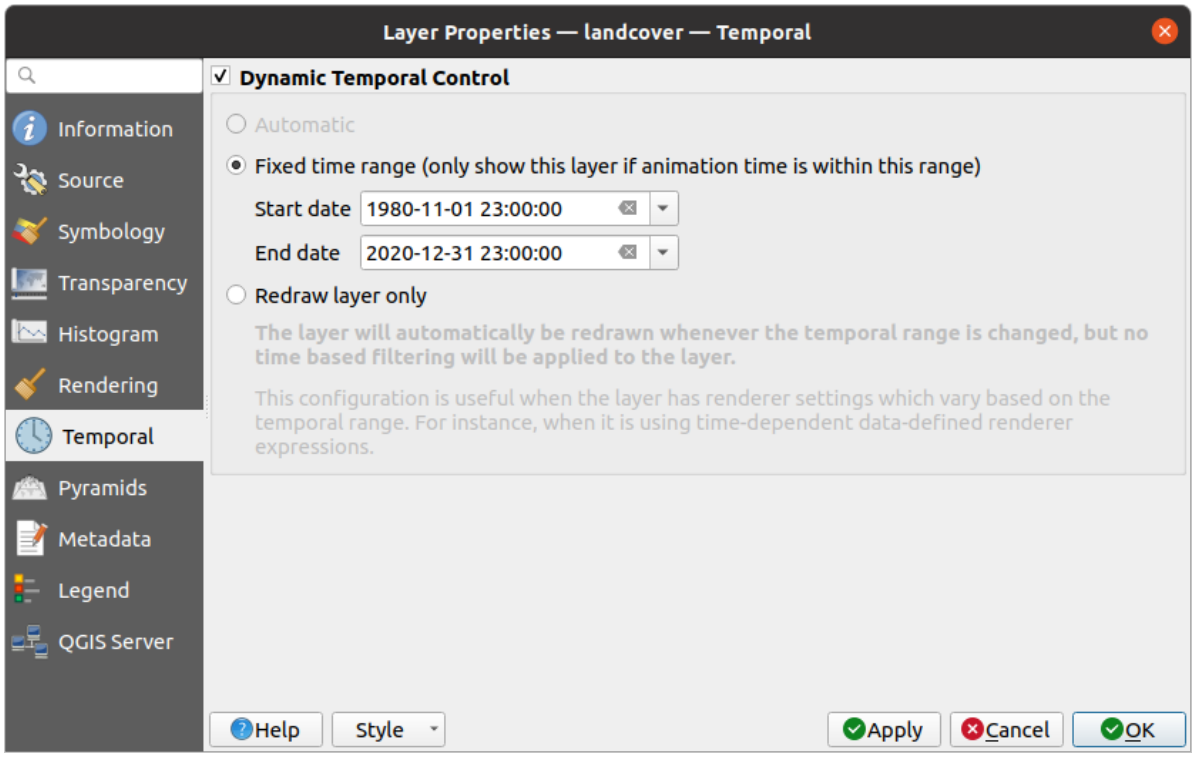


Fig. 17.15: Raster Temporal Properties

Check the *Dynamic Temporal Control* option and set whether the layer redraw should be:

- *Automatic*: the rendering is controlled by the underlying data provider if it suppports temporal data handling. E.g. this can be used with WMS-T layers or PostGIS rasters.
- *Fixed time range*: only show the raster layer if the animation time is within a *Start date* and *End date* range
- *Redraw layer only*: the layer is redrawn at each new animation frame. It's useful when the layer uses time-based expression values for renderer settings (e.g. data-defined renderer opacity, to fade in/out a raster layer).

17.1.8 Proprietățile Piramidelor

High resolution raster layers can slow navigation in QGIS. By creating lower resolution copies of the data (pyramids), performance can be considerably improved, as QGIS selects the most suitable resolution to use depending on the zoom level.

Trebuie să aveți acces de scriere în directorul în care sunt stocate datele originale, pentru a construi piramide.

From the *Resolutions* list, select resolutions at which you want to create pyramid levels by clicking on them.

If you choose **Internal (if possible)** from the *Overview format* drop-down menu, QGIS tries to build pyramids internally.

Notă: Please note that building pyramids may alter the original data file, and once created they cannot be removed. If you wish to preserve a «non-pyramided» version of your raster, make a backup copy prior to pyramid building.

If you choose **External** and **External (Erdas Imagine)** the pyramids will be created in a file next to the original raster with the same name and a `.ovr` extension.

Several *Resampling methods* can be used for pyramid calculation:

- cel mai apropiat vecin

- Medie
- Gauss
- Cubic
- Cubic Spline
- Laczos
- Mod
- Niciuna

Finally, click *Build Pyramids* to start the process.

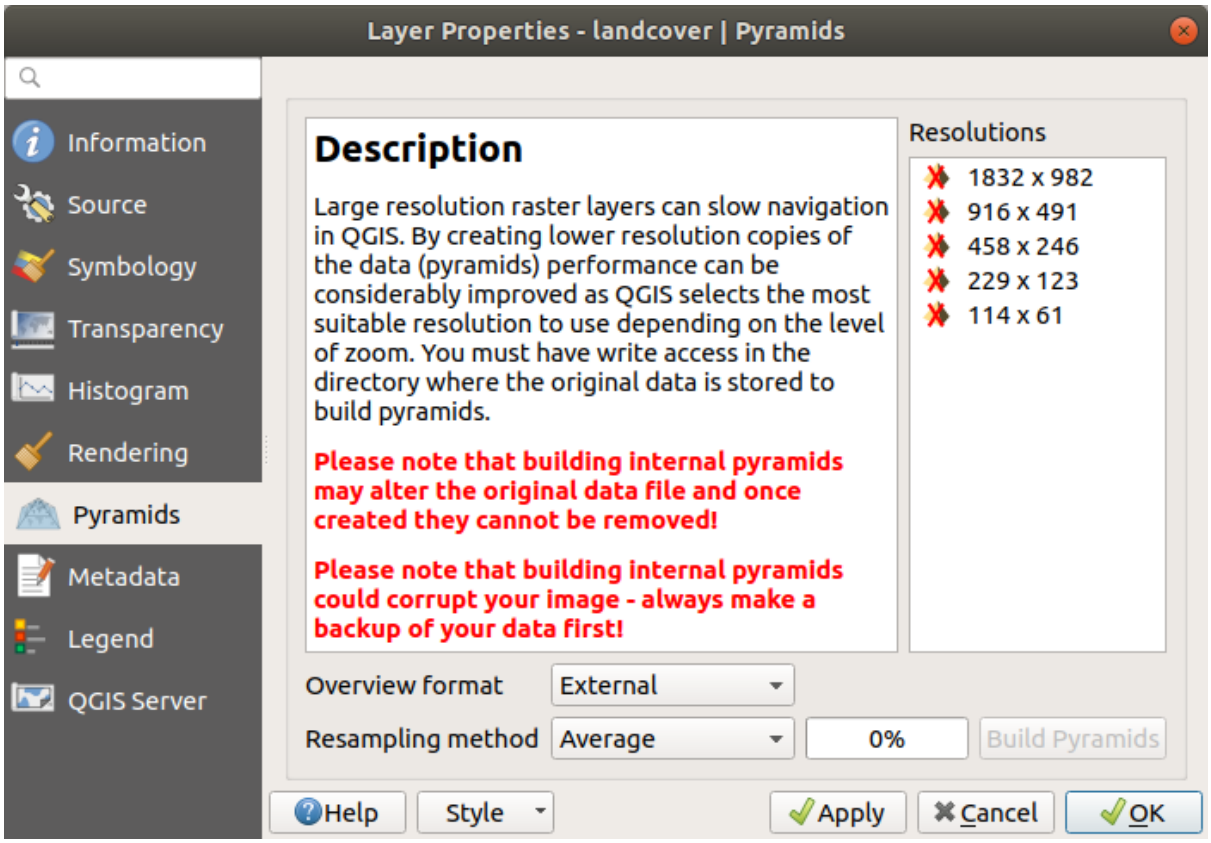


Fig. 17.16: Raster Pyramids

17.1.9 Elevation Properties

The *Elevation* tab provides options to control the layer elevation properties within a *3D map view* and its appearance in the *profile tool charts*. Specifically, you can set:

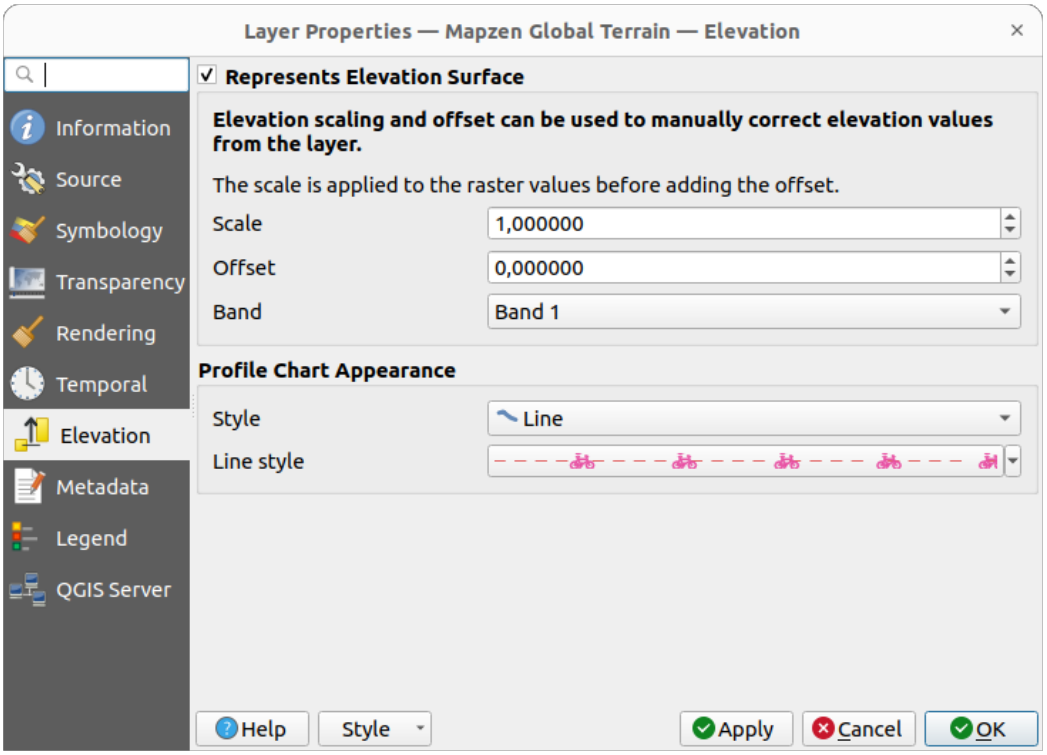


Fig. 17.17: Raster Elevation Properties

- *Represents Elevation Surface*: whether the raster layer represents a height surface (e.g DEM) and the pixel values should be interpreted as elevations. Check this option if you want to display a raster in an *elevation profile view*. You will also need to fill in the *Band* to pick values from and can apply a *Scale* factor and an *Offset*.
- *Profile Chart Appearance*: controls the rendering of the raster elevation data in the profile chart. The profile *Style* can be set as:
  - a *Line* with a specific *Line style*
  - an elevation surface rendered using a fill symbol either above (*Fill above*) or below (*Fill below*) the elevation curve line. The surface symbology is represented using:
    - a *Fill style*
    - and a *Limit*: the maximum (respectively minimum) altitude determining how high the fill surface will be

17.1.10 Proprietăți Metadate

The *Metadata* tab provides you with options to create and edit a metadata report on your layer. See *Metadata* for more information.

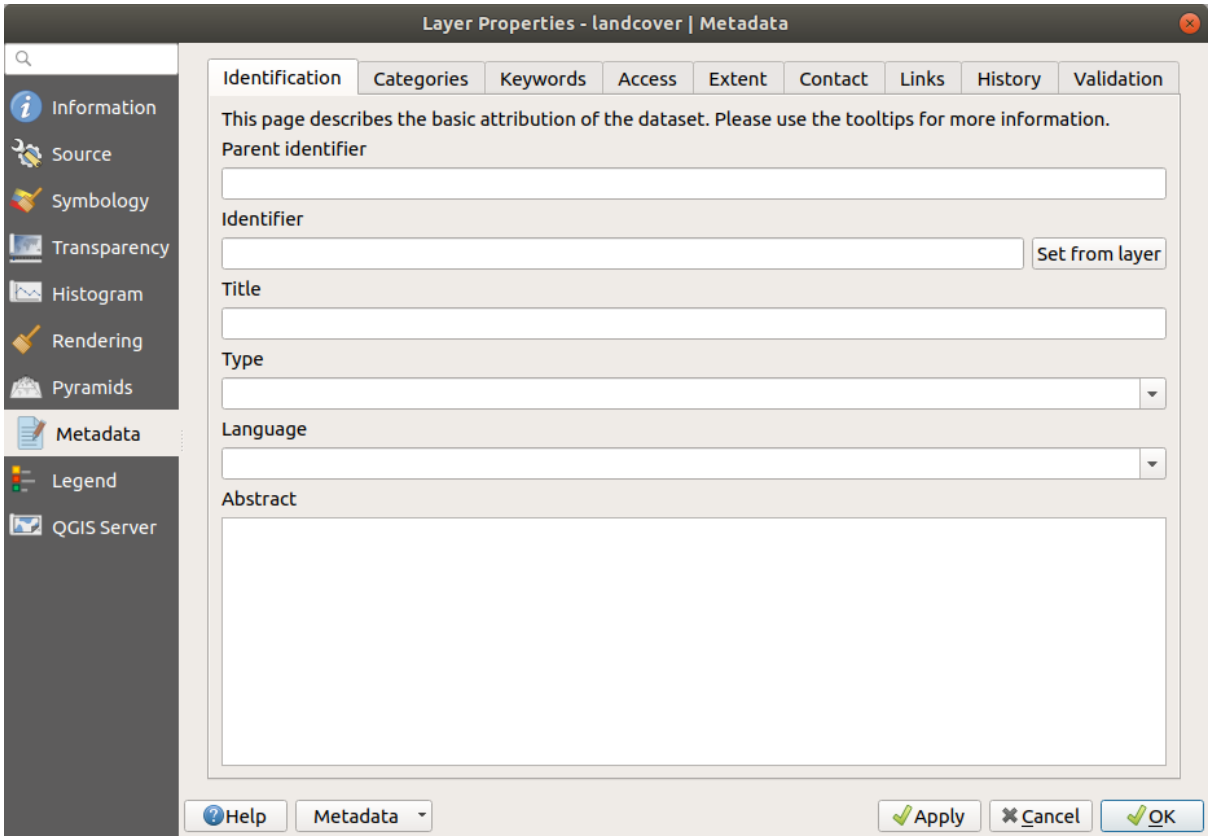


Fig. 17.18: Raster Metadata

17.1.11 Proprietățile Legendei

The *Legend* tab provides you with advanced settings for the *Layers panel* and/or the *print layout legend*. These options include:

- Depending on the symbology applied to the layer, you may end up with several entries in the legend, not necessarily readable/useful to display. The *Legend placeholder image* helps you *select an image* for replacement, displayed both in the *Layers* panel and the print layout legend.
- The *Embedded widgets in Legend* provides you with a list of widgets you can embed within the layer tree in the Layers panel. The idea is to have a way to quickly access some actions that are often used with the layer (setup transparency, filtering, selection, style or other stuff…).

  By default, QGIS provides a transparency widget but this can be extended by plugins that register their own widgets and assign custom actions to layers they manage.

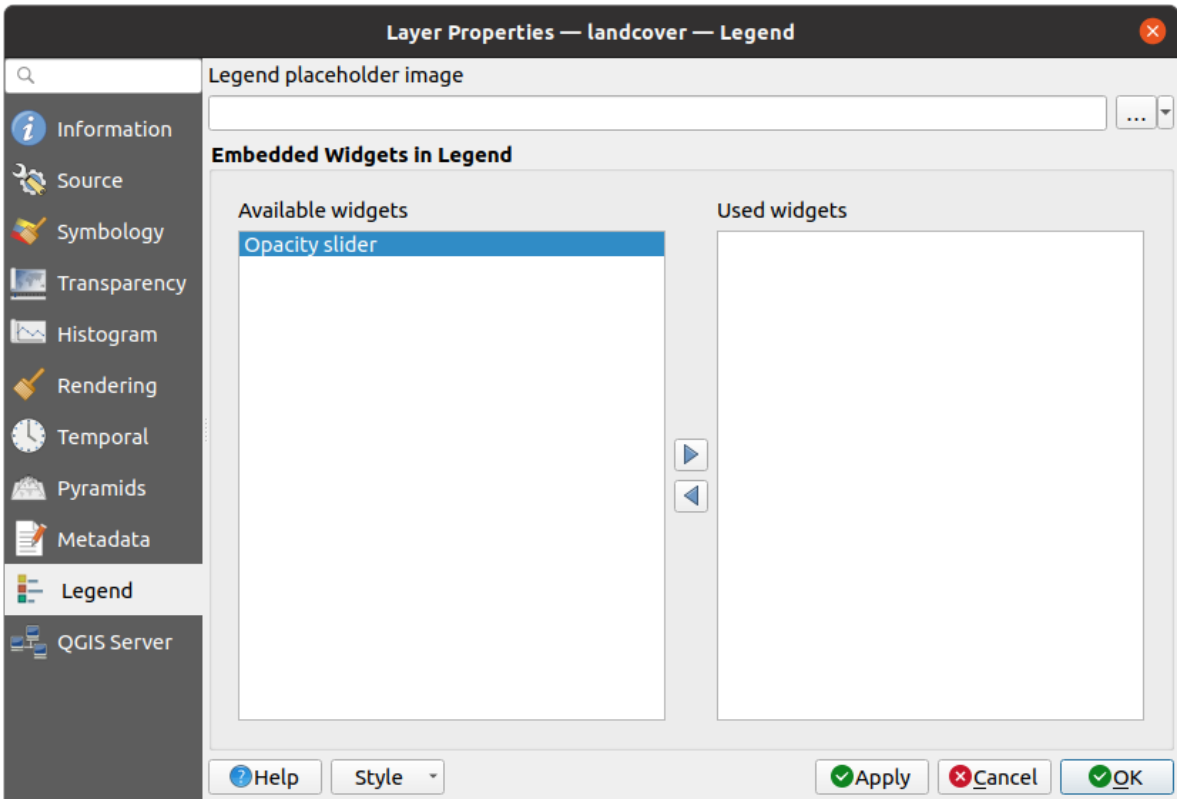


Fig. 17.19: Raster Legend

17.1.12 Display Properties

The *Display* tab helps you configure HTML map tips to use for pixels identification:

- *Enable Map Tips* controls whether to display map tips for the layer
- The *HTML Map Tip* provides a complex and full HTML text editor for map tips, mixing QGIS expressions and html styles and tags (multiline, fonts, images, hyperlink, tables, …). You can check the result of your code sample in the *Preview* frame.

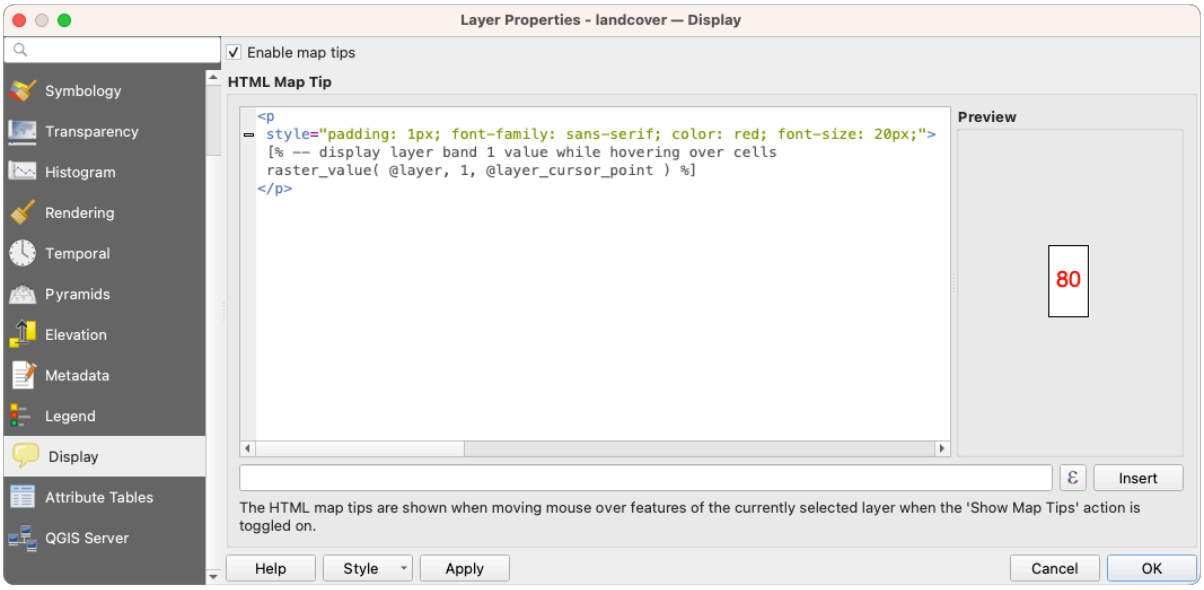


Fig. 17.20: Map tips with raster layer

To display map tips:

1. Select the menu option *View* ► *Show Map Tips* or click on the Show Map Tips icon of the *Attributes Toolbar*.
2. Make sure that the layer you target is active and has the *Enable Map Tips* property checked.
3. Move over a pixel, and the corresponding information will be displayed over.

Map tip is a cross-layer feature meaning that once activated, it stays on and applies to any map tip enabled layer in the project until it is toggled off.

17.1.13 Proprietățile Serverului QGIS

The *QGIS Server* tab helps you configure settings of the data when published by QGIS Server. The configuration concerns:

- *Description*: provides information to describe the data, such as *Short name*, *Title*, *Summary*, a *List of Keywords*, and a *Data URL* whose *Type* can be in `text/html`, `text/plain` or `application/pdf`.
- *Attribution*: a *Title* and *URL* to identify who provides the data
- *Metadata URL*: a list of *URL* for the metadata that can be of `FGDC` or `TC211` *Type*, and in `text/plain` or `text/xml` *Format*
- *Legend URL*: a *URL* for the legend, in either `image/png` or `image/jpeg` *Format*

Notă: When the raster layer you want to publish is already provided by a web service, further *properties* are available for setting.

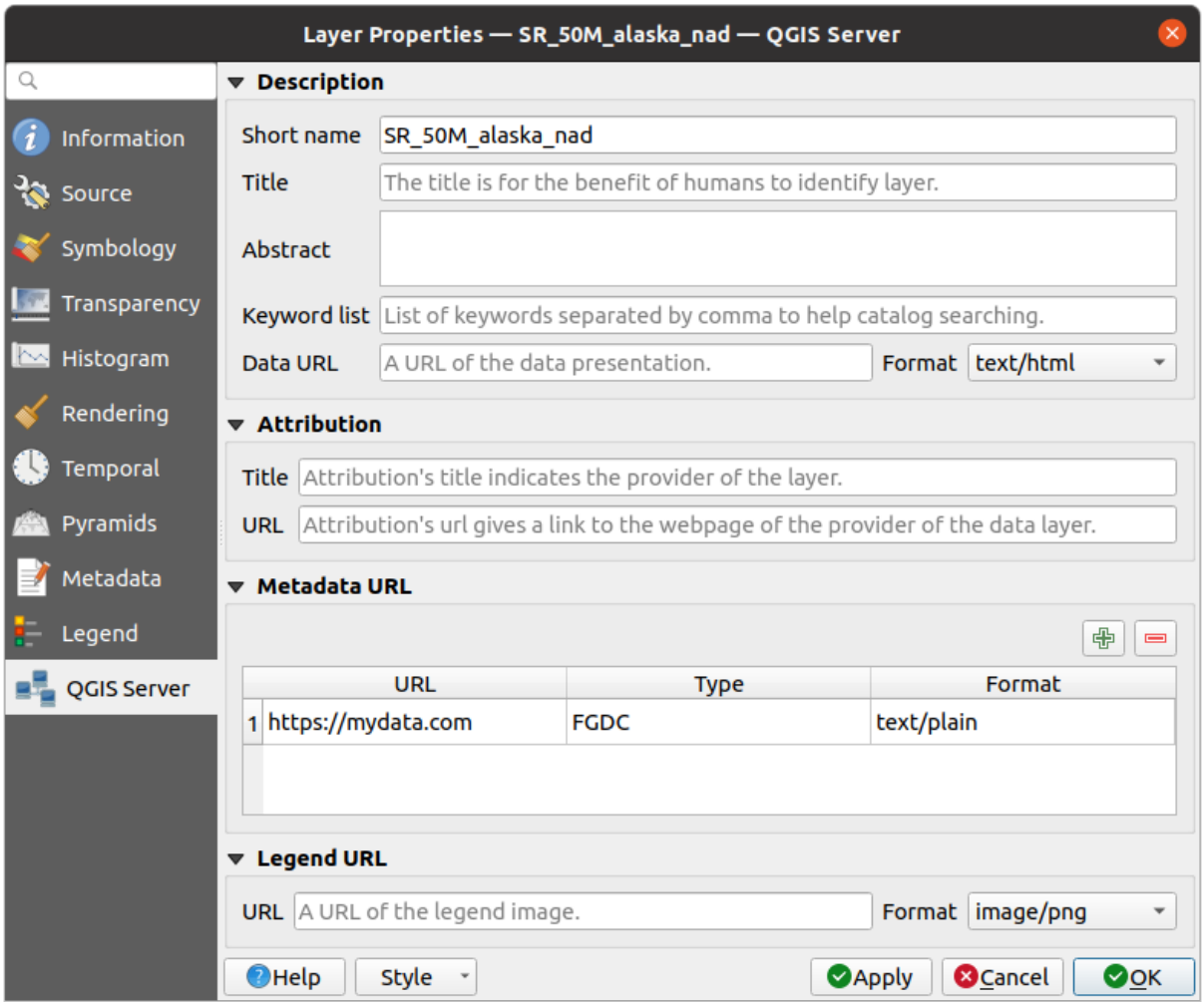


Fig. 17.21: QGIS Server in Raster Properties

17.1.14 Identify raster cells

The *identify features* tool allows you to get information about specific points in a raster layer.

To use the *Identify features* tool:

1. Select the raster layer in the Layers panel.
2. Click on the *Identify features* tool in the toolbar or press `Ctrl+Shift+I`.
3. Click on the point in the raster layer that you want to identify.

The Identify Results panel will open in its default `Tree` view and display information about the clicked point. Below the name of the raster layer, you have on the left the band(s) of the clicked pixel, and on the right their respective value. These values can also be rendered (from the *View* menu located at the bottom of the panel) in:

- a `Table` view - organizes the information about the identified features and their values in a table.
- a `Graph` view - organizes the information about the identified features and their values in a graph.

Under the pixel attributes, you will find the *Derived* information, such as:

- `X` and `Y` coordinate values of the point clicked
- Column and row of the point clicked (pixel)

17.2 Analiza Raster

17.2.1 Calculatorul Raster

The *Raster Calculator* in the *Raster* menu allows you to perform calculations on the basis of existing raster pixel values (see Fig. 17.22). The results are written to a new raster layer in a GDAL-supported format.

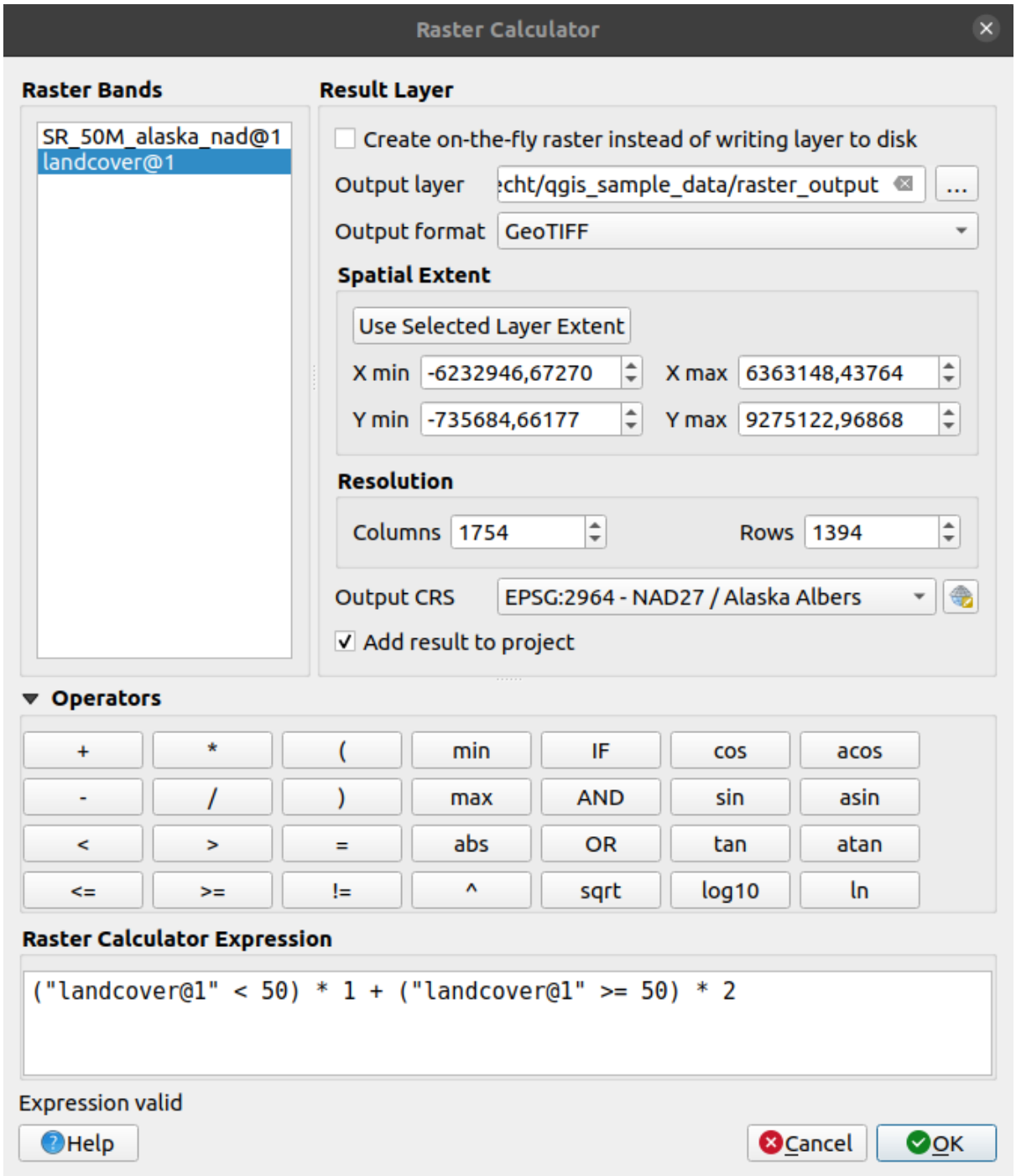


Fig. 17.22: Calculatorul Raster

The *Raster bands* list contains all loaded raster layers that can be used. To add a raster to the raster calculator expression field, double click its name in the Fields list. You can then use the operators to construct calculation expressions, or you can just type them into the box.

In the *Result layer* section, you will need to define an output layer. You can:

- *Create on-the-fly raster instead of writing layer to disk*:
  - If unchecked, the output is stored on the disk as a new plain file. An *Output layer* path and an *Output format* are required.
  - If checked, a virtual raster layer, i.e. a raster layer defined by its URI and whose pixels are calculated on-the-fly, is created. It's not a new file on disk; the virtual layer is still connected to the rasters used in the calculation meaning that deleting or moving these rasters would break it. A *Layer name* can be provided, otherwise the calculation expression is used as such. Removing the virtual layer from the project deletes it, and it can be made persistent in file using the layer *Export* ► *Save as…* contextual menu.
- Define the *Spatial extent* of the calculation based on an input raster layer extent, or on custom X,Y coordinates
- Set the *Resolution* of the layer using columns and rows number. If the input layer has a different resolution, the values will be resampled with the nearest neighbor algorithm.
- With the *Add result to project* checkbox, the result layer will automatically be added to the legend area and can be visualized. Checked by default for virtual rasters.

The *Operators* section contains all available operators. To add an operator to the raster calculator expression box, click the appropriate button. Mathematical calculations (+, −, *, …) and trigonometric functions (`sin`, `cos`, `tan`, …) are available. Conditional expressions (=, !=, <, >=, …) return either 0 for false or 1 for true, and therefore can be used with other operators and functions.

Vezi și:

Calculator raster and *Raster calculator (virtual)* algorithms

Raster calculator expression

The dialog

The *Raster calculator expression* dialog provides means to write expressions for pixels calculations between a set of raster layers.

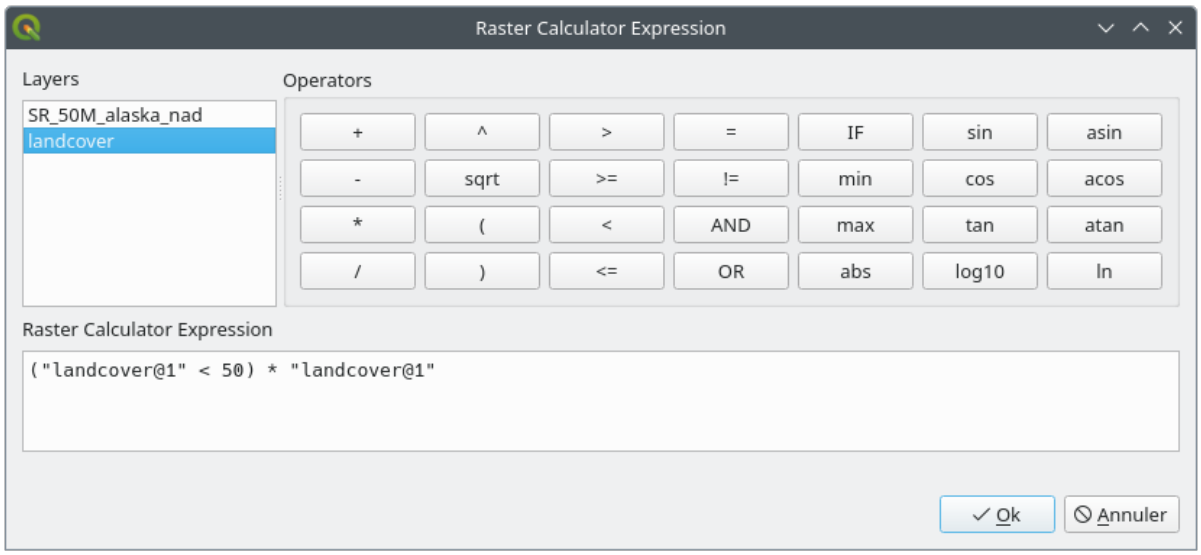


Fig. 17.23: Raster Expression Calculator

- *Layers*: Shows the list of all raster layers loaded in the legend. These can be used to fill the expression box (double click to add). Raster layers are referred by their name and the number of the band:

`layer_name@band_number`. For instance, the first band from a layer named `DEM` will be referred as `DEM@1`.

- *Operators*: contains a number of calculation operators for pixels manipulation:
  - Arithmetical: `+`, `-`, `*`, `sqrt`, `abs`, `ln`, ...
  - Trigonometric: `sin`, `cos`, `tan`, ...
  - Comparison: `=`, `!=`, `<`, `>=`, ...
  - Logical: `IF`, `AND`, `OR`, `(`, `)`
  - Statistical: `min`, `max`

  To add an operator to the raster calculator expression box, click the appropriate button.
- *Raster calculator expression* is the area in which the expression is composed

Exemple

Conversia valorilor de elevație de la metri la picioare

Pentru crearea unui raster de elevație în feet dintr-un raster în metri, trebuie să utilizați factorul de conversie de la metri la picioare: 3.28. Expresia este:

```
"elevation@1" * 3.28
```

Folosirea unei măști

Dacă doriți să mascați unele părți dintr-un raster - să zicem, de exemplu, pentru că vă interesează doar altitudinile de peste 0 metri - puteți utiliza următoarea expresie pentru a crea o mască și pentru a aplica rezultatul unui raster, într-un singur pas.

```
("elevation@1" >= 0) * "elevation@1"
```

In other words, for every cell greater than or equal to 0 the conditional expression evaluates to 1, which keeps the original value by multiplying it by 1. Otherwise the conditional expression evaluates to 0, which sets the raster value to 0. This creates the mask on the fly.

Classify a Raster

În cazul în care doriți să clasificați un raster - să zicem, de exemplu, în două clase de altitudine, puteți utiliza următoarea expresie pentru a crea un raster cu două valori, 1 și 2, într-un singur pas.

```
("elevation@1" < 50) * 1 + ("elevation@1" >= 50) * 2
```

Cu alte cuvinte, pentru fiecare celulă mai mică de 50, valoarea sa va fi setată la 1. Pentru fiecare celulă mai mare sau egală cu 50 valoarea sa va fi setată la 2.

Or you can use the `IF` operator.

```
if ( elevation@1 < 50 , 1 , 2 )
```

17.3 Georeferencer

The Georeferencer is a tool for generating world files for layers. It allows you to reference rasters or vectors to geographic or projected coordinate systems by creating a new GeoTiff or by adding a world file to the existing image. The basic approach to georeferencing a layer is to locate points on it for which you can accurately determine coordinates.

Features

Pictogramă	Scop	Pictogramă	Scop
	Open raster		Open vector
	Start georeferencing		
	Generate GDAL Script		Load GCP Points
	Save GCP Points As		Transformation settings
	Add Point		Delete Point
	Move GCP Point		Pan
	Zoom In		Zoom Out
	Zoom To Layer		Zoom Last
	Zoom Next		Link Georeferencer to QGIS
	Link QGIS to Georeferencer		Full histogram stretch
	Local histogram stretch		

Table Georeferencer: Georeferencer Tools

17.3.1 Usual procedure

As X and Y coordinates (DMS (dd mm ss.ss), DD (dd.dd) or projected coordinates (mmmm.mm)), which correspond with the selected point on the image, two alternative procedures can be used:

- The raster itself sometimes provides crosses with coordinates „written" on the image. In this case, you can enter the coordinates manually.
- Using already georeferenced layers. This can be either vector or raster data that contain the same objects/features that you have on the image that you want to georeference and with the projection that you want for your image. In this case, you can enter the coordinates by clicking on the reference dataset loaded in the QGIS map canvas.

The usual procedure for georeferencing an image involves selecting multiple points on the raster, specifying their coordinates, and choosing a relevant transformation type. Based on the input parameters and data, the Georeferencer will compute the world file parameters. The more coordinates you provide, the better the result will be.

The first step is to start QGIS and click on *Layer* ► *Georeferencer*, which appears in the QGIS menu bar. The Georeferencer dialog appears as shown in Fig. 17.24.

For this example, we are using a topo sheet of South Dakota from SDGS. It can later be visualized together with the data from the GRASS `spearfish60` location. You can download the topo sheet here: https://grass.osgeo.org/sampledata/spearfish_toposheet.tar.gz.

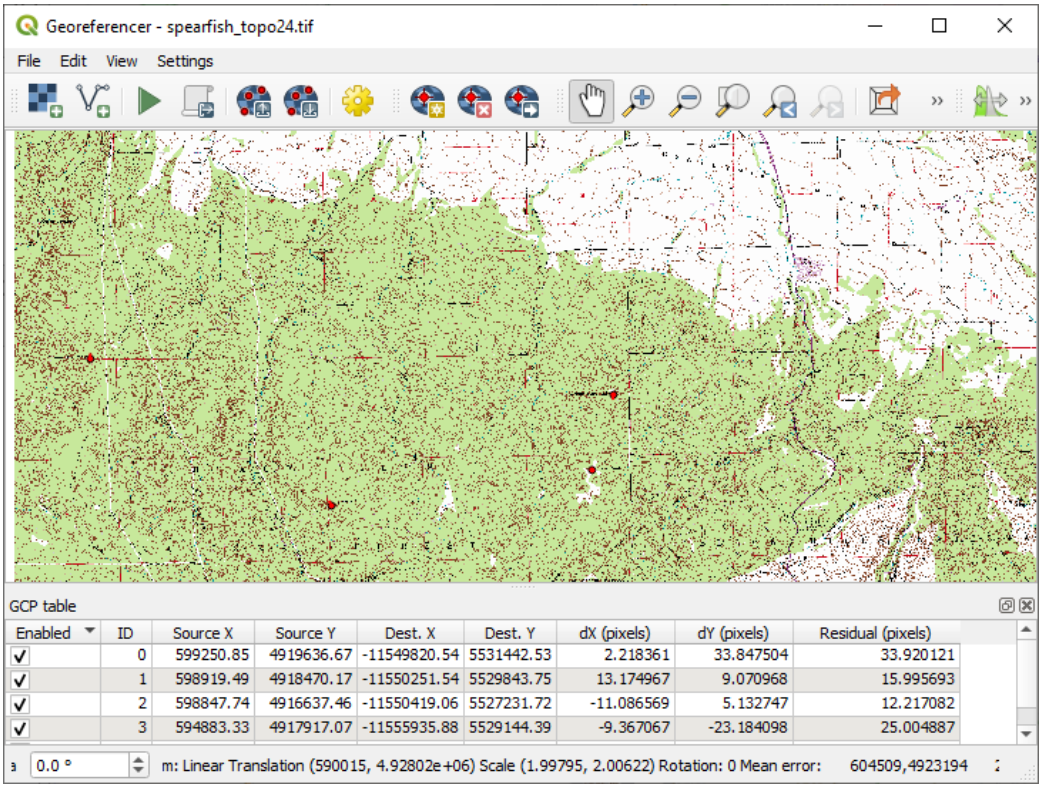


Fig. 17.24: Georeferencer Dialog

Entering ground control points (GCPs)

1. To start georeferencing an unreferenced raster, we must load it using the button. The raster will show up in the main working area of the dialog. Once the raster is loaded, we can start to enter reference points.

2. Using the Add Point button, add points to the main working area and enter their coordinates (see Figure Fig. 17.25). For this procedure you have the following options:

   - Click on a point in the raster image and enter the X and Y coordinates manually, along with the CRS of the point.
   - Click on a point in the raster image and choose the From map canvas button to add the X and Y coordinates with the help of a georeferenced map already loaded in the QGIS map canvas. The CRS will be set automatically.
   - When entering GCPs from the main map canvas, you have the option to hide the georeferencer window while selecting points from the main canvas. If the *Automatically hide georeferencer window* checkbox is ticked, after clicking *From Map Canvas*, the main georeferencer window will be hidden until a point is added on the map canvas. The *Enter Map Coordinates* dialog will remain open. If the box is unchecked, both windows will remain open while selecting a point on the map canvas. This option only takes effect when the georeferencer window is not docked in the main interface.

3. Continue entering points. You should have at least four points, and the more coordinates you can provide, the better the result will be. There are additional tools for zooming and panning the working area in order to locate a relevant set of GCP points.

4. With the tool, you can move the GCPs in both the canvas and the georeferencing window, if you need to correct them.

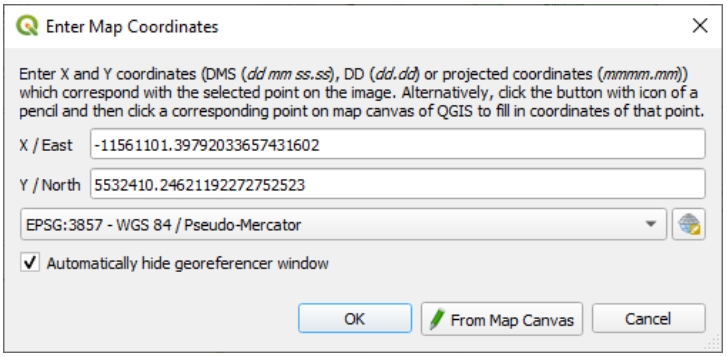


Fig. 17.25: Add points to the raster image

The points that are added to the map will be stored in a separate text file (`[filename].points`) usually together with the raster image. This allows us to reopen the Georeferencer at a later date and add new points or delete existing ones to optimize the result. The points file contains values of the form: `mapX, mapY, pixelX, pixelY`. You can use the Load GCP points and Save GCP points as buttons to manage the files.

Defining the transformation settings

After you have added your GCPs to the raster image, you need to define the transformation settings for the georeferencing process.

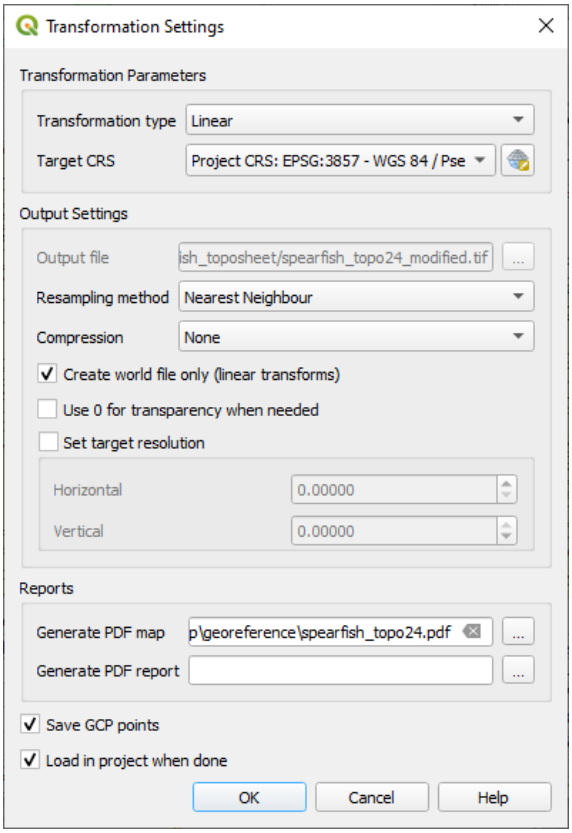


Fig. 17.26: Defining the georeferencer transformation settings

Available Transformation algorithms

A number of transformation algorithms are available, dependent on the type and quality of input data, the nature and amount of geometric distortion that you are willing to introduce to the final result, and the number of ground control points (GCPs).

Currently, the following *Transformation types* are available:

- The **Linear** algorithm is used to create a world file and is different from the other algorithms, as it does not actually transform the raster pixels. It allows positioning (translating) the image and uniform scaling, but no rotation or other transformations. It is the most suitable if your image is a good quality raster map, in a known CRS, but is just missing georeferencing information. At least 2 GCPs are needed.
- The **Helmert** transformation also allows rotation. It is particularly useful if your raster is a good quality local map or orthorectified aerial image, but not aligned with the grid bearing in your CRS. At least 2 GCPs are needed.
- The **Polynomial 1** algorithm allows a more general affine transformation, in particular also a uniform shear. Straight lines remain straight (i.e., collinear points stay collinear) and parallel lines remain parallel. This is particularly useful for georeferencing data cartograms, which may have been plotted (or data collected) with different ground pixel sizes in different directions. At least 3 GCP's are required.
- The **Polynomial** algorithms 2-3 use more general 2nd or 3rd degree polynomials instead of just affine transformation. This allows them to account for curvature or other systematic warping of the image, for instance photographed maps with curving edges. At least 6 (respectively 10) GCP's are required. Angles and local scale are not preserved or treated uniformly across the image. In particular, straight lines may become curved, and there may be significant distortion introduced at the edges or far from any GCPs arising from extrapolating the data-fitted polynomials too far.
- The **Projective** algorithm generalizes Polynomial 1 in a different way, allowing transformations representing a central projection between 2 non-parallel planes, the image and the map canvas. Straight lines stay straight, but parallelism is not preserved and scale across the image varies consistently with the change in perspective. This transformation type is most useful for georeferencing angled photographs (rather than flat scans) of good quality maps, or oblique aerial images. A minimum of 4 GCPs is required.
- Finally, the **Thin Plate Spline** (TPS) algorithm „rubber sheets" the raster using multiple local polynomials to match the GCPs specified, with overall surface curvature minimized. Areas away from GCPs will be moved around in the output to accommodate the GCP matching, but will otherwise be minimally locally deformed. TPS is most useful for georeferencing damaged, deformed, or otherwise slightly inaccurate maps, or poorly orthorectified aerials. It is also useful for approximately georeferencing and implicitly reprojecting maps with unknown projection type or parameters, but where a regular grid or dense set of ad-hoc GCPs can be matched with a reference map layer. It technically requires a minimum of 10 GCPs, but usually more to be successful.

In all of the algorithms except TPS, if more than the minimum GCPs are specified, parameters will be fitted so that the overall residual error is minimized. This is helpful to minimize the impact of registration errors, i.e. slight imprecisions in pointer clicks or typed coordinates, or other small local image deformations. Absent other GCPs to compensate, such errors or deformations could translate into significant distortions, especially near the edges of the georeferenced image. However, if more than the minimum GCPs are specified, they will match only approximately in the output. In contrast, TPS will precisely match all specified GCPs, but may introduce significant deformations between nearby GCPs with registration errors.

Define the Resampling method

The type of resampling you choose will likely depend on your input data and the ultimate objective of the exercise. If you don't want to change statistics of the raster (other than as implied by nonuniform geometric scaling if using other than the Linear, Helmert, or Polynomial 1 transformations), you might want to choose «Nearest neighbour». In contrast, «cubic resampling», for instance, will usually generate a visually smoother result.

It is possible to choose between five different resampling methods:

1. Nearest neighbour
2. Bilinear (2x2 kernel)
3. Cubic (4x4 kernel)
4. Cubic B-Spline (4x4 kernel)
5. Lanczos (6x6 kernel)

Define the transformation settings

There are several options that need to be defined for the georeferenced output raster.

- The *Create world file* checkbox is only available if you decide to use the linear transformation type, because this means that the raster image actually won't be transformed. In this case, the *Output raster* field is not activated, because only a new world file will be created.
- For all other transformation types, you have to define an *Output raster*. As default, a new file ([filename]_modified) will be created in the same folder together with the original raster image.
- As a next step, you have to define the *Target CRS* (Coordinate Reference System) for the georeferenced raster (see *Lucrul cu Proiecții*).
- If you like, you can **generate a pdf map** and also **a pdf report**. The report includes information about the used transformation parameters, an image of the residuals and a list with all GCPs and their RMS errors.
- Furthermore, you can activate the *Set Target Resolution* checkbox and define the pixel resolution of the output raster. Default horizontal and vertical resolution is 1.
- The *Use 0 for transparency when needed* can be activated, if pixels with the value 0 shall be visualized transparent. In our example toposheet, all white areas would be transparent.
- The *Save GCP Points* will store GCP Points in a file next to the output raster.
- Finally, *Load in project when done* loads the output raster automatically into the QGIS map canvas when the transformation is done.

Show and adapt raster properties

Clicking on the *Raster properties* option in the *Settings* menu opens the *Layer properties* dialog of the raster file that you want to georeference.

Configure the georeferencer

You can customize the behavior of the georeferencer in *Settings* ► *Configure Georeferencer* (or use keyboard shortcut `Ctrl+P`).

- Under *Point Tip* you can use the checkboxes to toggle displaying GCP IDs and X/Y coordinates in both the Georeferencer window and the main map canvas.
- *Residual Units* controls whether residual units are given in pixels or map units
- *PDF Report* allows you to set margin size in mm for the report export
- *PDF Map* allows you to choose a paper size for the map export
- Finally, you can activate to *Show Georeferencer window docked*. This will dock the Georeferencer window in the main QGIS window rather than showing it as a separate window that can be minimized.

Running the transformation

After all GCPs have been collected and all transformation settings are defined, just press the Start georeferencing button to create the new georeferenced raster.

CAPITOLUL 18

Working with Mesh Data

18.1 What's a mesh?

A mesh is an unstructured grid usually with temporal and other components. The spatial component contains a collection of vertices, edges and/or faces, in 2D or 3D space:

- **vertices** - XY(Z) points (in the layer's coordinate reference system)
- **edges** - connect pairs of vertices
- **faces** - a face is a set of edges forming a closed shape - typically a triangle or a quadrilateral (quad), rarely polygons with more vertices

Relying on the above, mesh layers can thus have different types of structure:

- 1D Meshes: consist of vertices and edges. An edge connects two vertices and can have assigned data (scalars or vectors) on it. The 1D mesh network can be for example used for modelling of an urban drainage system.
- 2D meshes: consist of faces with triangles, regular or unstructured quads.
- 3D layered meshes: consist of multiple stacked 2D unstructured meshes each extruded in the vertical direction (levels) by means of a vertical coordinate. The vertices and faces have the same topology in each vertical level. The mesh definition (vertical level extrusion) could in general change in time. The data is usually defined in volume centres or by some parametric function.

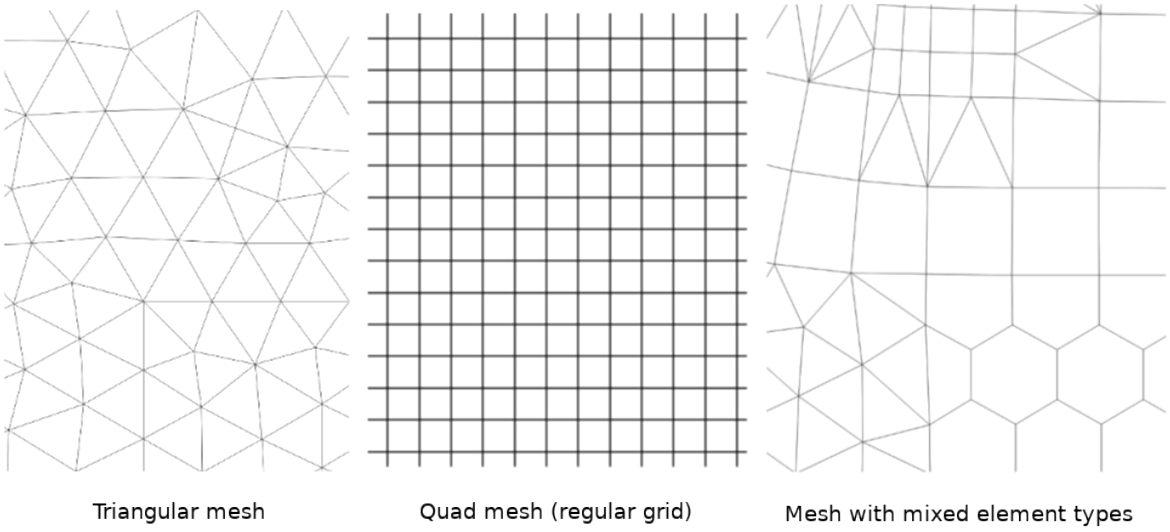


Fig. 18.1: Different mesh types

Mesh provides information about the spatial structure. In addition, the mesh can have datasets (groups) that assign a value to every vertex. For example, having a triangular mesh with numbered vertices as shown in the image below:

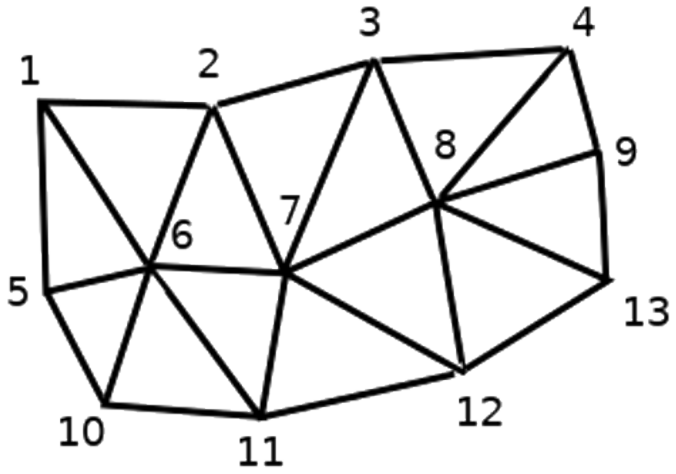


Fig. 18.2: Triangular grid with numbered vertices

Each vertex can store different datasets (typically multiple quantities), and those datasets can also have a temporal dimension. Thus, a single file may contain multiple datasets.

The following table gives an idea about the information that can be stored in mesh datasets. Table columns represent indices of mesh vertices, each row represents one dataset. Datasets can have different datatypes. In this case, it stores wind velocity at 10m at a particular moments in time (t1, t2, t3).

In a similar way, the mesh dataset can also store vector values for each vertex. For example, wind direction vector at the given time stamps:

Tabelul 18.1: Example of mesh dataset

10 metre wind	1	2	3	...
10 metre speed at time=t1	17251	24918	32858	…
10 metre speed at time=t2	19168	23001	36418	…
10 metre speed at time=t3	21085	30668	17251	…
…	…	…	…	…
10m wind direction time=t1	[20,2]	[20,3]	[20,4.5]	…
10m wind direction time=t2	[21,3]	[21,4]	[21,5.5]	…
10m wind direction time=t3	[22,4]	[22,5]	[22,6.5]	…
…	…	…	…	…

We can visualize the data by assigning colors to values (similarly to how it is done with *Singleband pseudocolor* raster rendering) and interpolating data between vertices according to the mesh topology. It is common that some quantities are 2D vectors rather than being simple scalar values (e.g. wind direction). For such quantities it is desirable to display arrows indicating the directions.

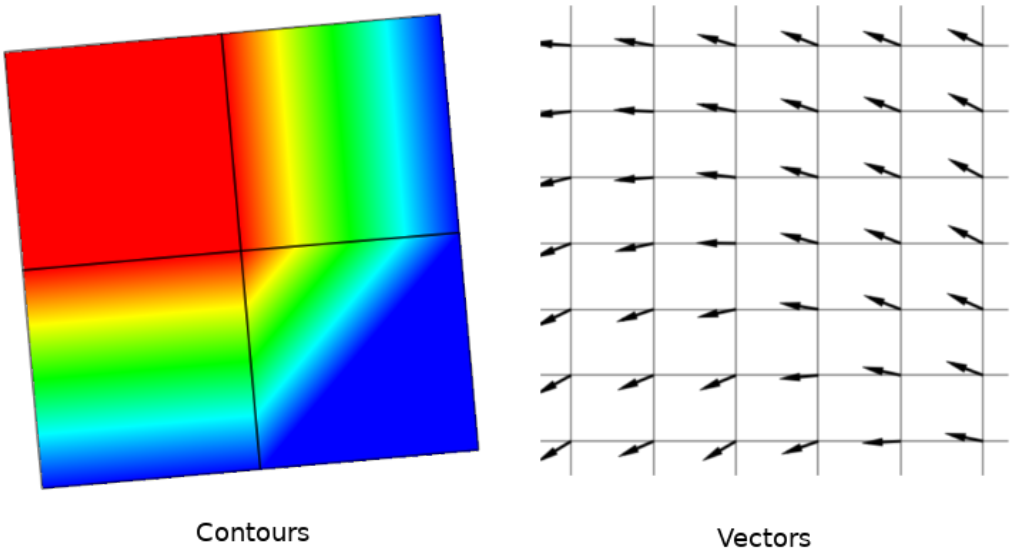


Fig. 18.3: Possible visualisation of mesh data

18.2 Supported formats

QGIS accesses mesh data using the MDAL drivers, and natively supports a variety of formats. Whether QGIS can edit a mesh layer depends on the format and the mesh structure type.

To load a mesh dataset into QGIS, use the *Mesh* tab in the *Data Source Manager* dialog. Read *Loading a mesh layer* for more details.

18.3 Mesh Dataset Properties

The *Layer Properties* dialog for a mesh layer provides general settings to manage dataset groups of the layer and their rendering (active dataset groups, symbology, 2D and 3D rendering). It also provides information about the layer.

To access the *Layer Properties* dialog:

- In the *Layers* panel, double-click the layer or right-click and select *Properties…* from the pop-up menu;
- Go to *Layer* ► *Layer Properties…* menu when the layer is selected.

The mesh *Layer Properties* dialog provides the following sections:

Tabelul 18.2: Tabs of the Mesh Layer Properties

Information	*Source*	*Symbology*[1]
3D View[1]	*Temporal*	*Elevation*
Rendering	*Metadata*	

[1] Also available in the *Layer styling panel*

Notă: Most of the properties of a mesh layer can be saved to or loaded from a `.qml` using the *Style* menu at the bottom of the dialog. More details at *Managing Custom Styles*.

18.3.1 Information Properties

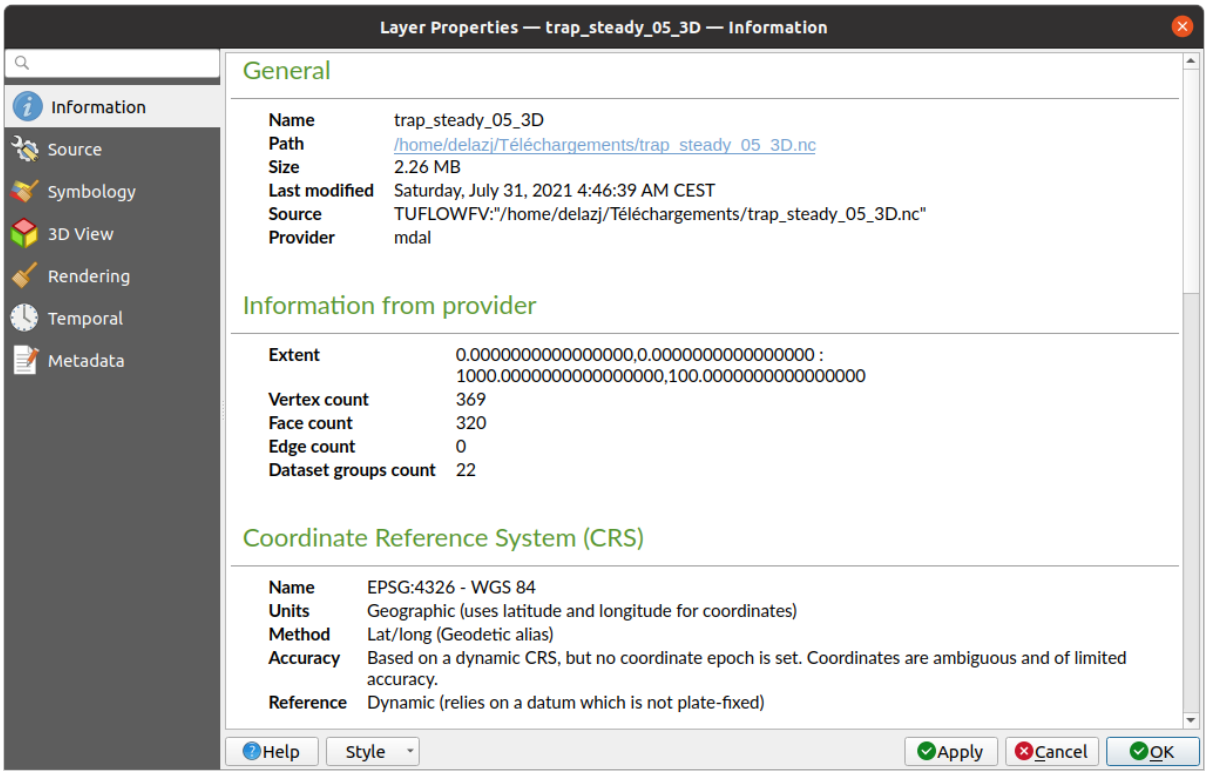


Fig. 18.4: Mesh Layer Information Properties

The *Information* tab is read-only and represents an interesting place to quickly grab summarized information and metadata on the current layer. Provided information are:

- general such as name in the project, source path, list of auxiliary files, last save time and size, the used provider
- based on the provider of the layer: extent, vertex, face, edges and/or dataset groups count
- the Coordinate Reference System: name, units, method, accuracy, reference (i.e. whether it's static or dynamic)
- extracted from filled *metadata*: access, extents, links, contacts, history…

18.3.2 Source Properties

The *Source* tab displays basic information about the selected mesh, including:

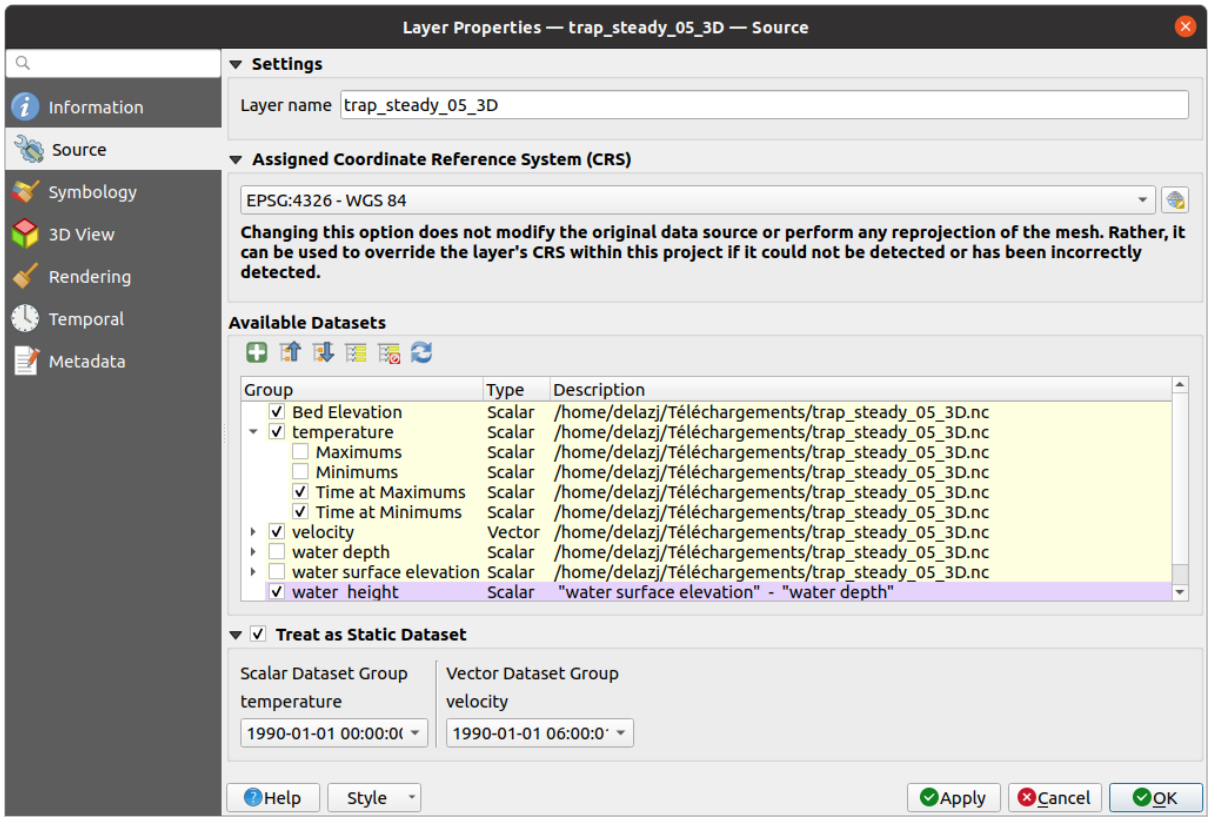


Fig. 18.5: Mesh Layer Source Properties

- the layer name to display in the *Layers* panel
- setting the Coordinate Reference System: Displays the layer's *Assigned Coordinate Reference System (CRS)*.

  You can change the layer's CRS by selecting a recently used one in the drop-down list or clicking on *Select CRS* button (see *Coordinate Reference System Selector*). Use this process only if the CRS applied to the layer is wrong or if none was applied.
- The *Available datasets* frame lists all the dataset groups (and subgroups) in the mesh layer, with their type and description in a tree view. Both regular datasets (i.e. their data is stored in the file) and virtual datasets (which are *calculated on the fly*) are listed.
  - Use the *Assign extra dataset to mesh* button to add more groups to the current mesh layer.
  - *Collapse all* and *Expand all* the dataset tree, in case of embedded groups

  - If you are interested in few datasets, you can uncheck the others and make them unavailable in the project
  - Double-click over a name and you can rename the dataset.
  - *Reset to defaults*: checks all the groups and renames them back to their original name in the provider.
  - Right-click over a virtual dataset group and you can:
    * *Remove dataset group* from the project
    * *Save dataset group as…* a file on disk, to any supported format. The new file is kept assigned to the current mesh layer in the project.
- Checking the *Treat as static dataset* group allows to ignore the *map temporal navigation* properties while rendering the mesh layer. For each active dataset group (as selected in *Symbology* ► *Datasets* tab), you can:
  - set to *None*: the dataset group is not displayed at all
  - *Display dataset*: e.g., for the „bed elevation” dataset which is not time aware
  - extract a particular date time: the dataset matching the provided time is rendered and stay fixed during map navigation.

18.3.3 Symbology Properties

Click the *Symbology* button to activate the dialog. Symbology properties are divided into several tabs:

- *Datasets*
- *Contours*
- *Vectors*
- *Rendering*
- *Stacked mesh averaging method*

Datasets

The tab Datasets is the main place to control and set which datasets will be used for the layer. It presents the following items:

- *Groups* available in the mesh dataset, with whether they provide:
  - scalar dataset
  - or vector dataset: by default, each vector dataset has a scalar dataset representing its magnitude automatically generated.

  Click on the icon next to the dataset name to select the group and type of data to represent.
- *Selected dataset group(s) metadata*, with details on:
  - the mesh type: edges or faces
  - the data type: vertices, edges, faces or volume
  - whether it's of vector type or not
  - the original name in the mesh layer
  - the unit, if applicable
- *blending mode* available for the selected datasets.

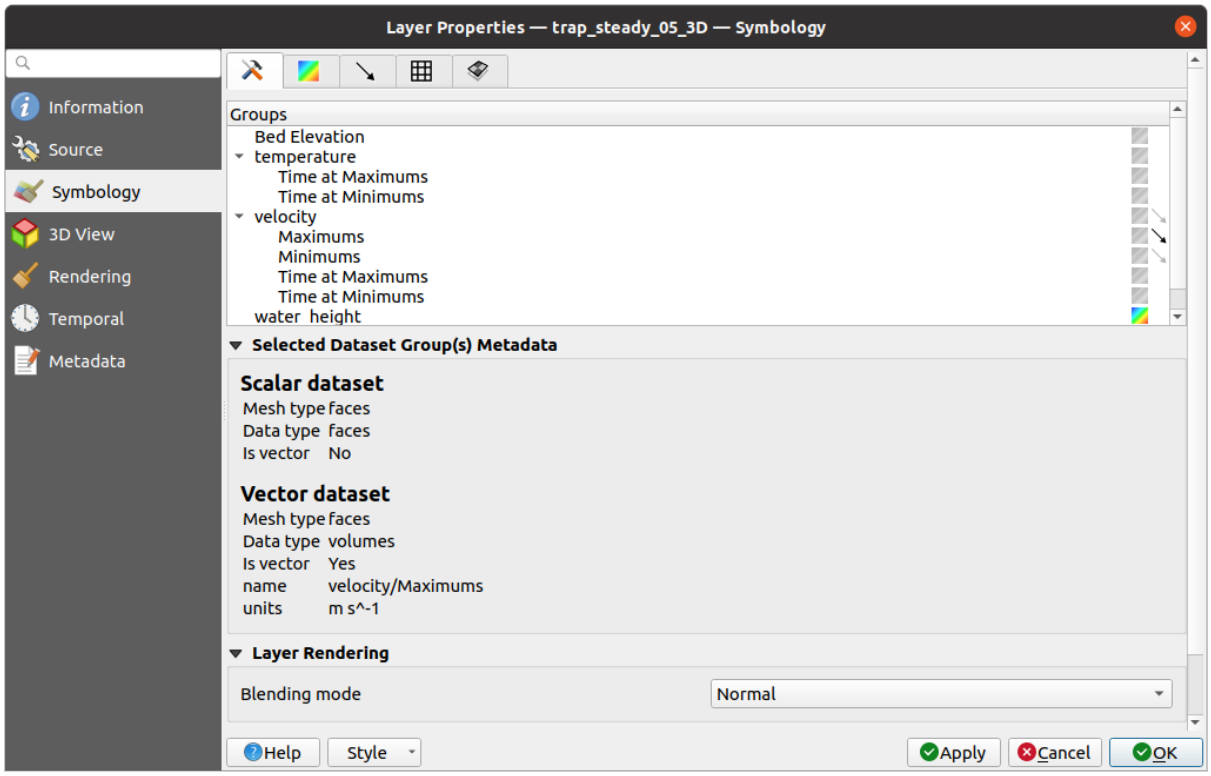


Fig. 18.6: Mesh Layer Datasets

You can apply symbology to the selected vector and/or scalar group using the next tabs.

Contours Symbology

Notă: The Contours tab can be activated only if a scalar dataset has been selected in the *Datasets* tab.

In the Contours tab you can see and change the current visualization options of contours for the selected group, as shown in Fig. 18.7 below:

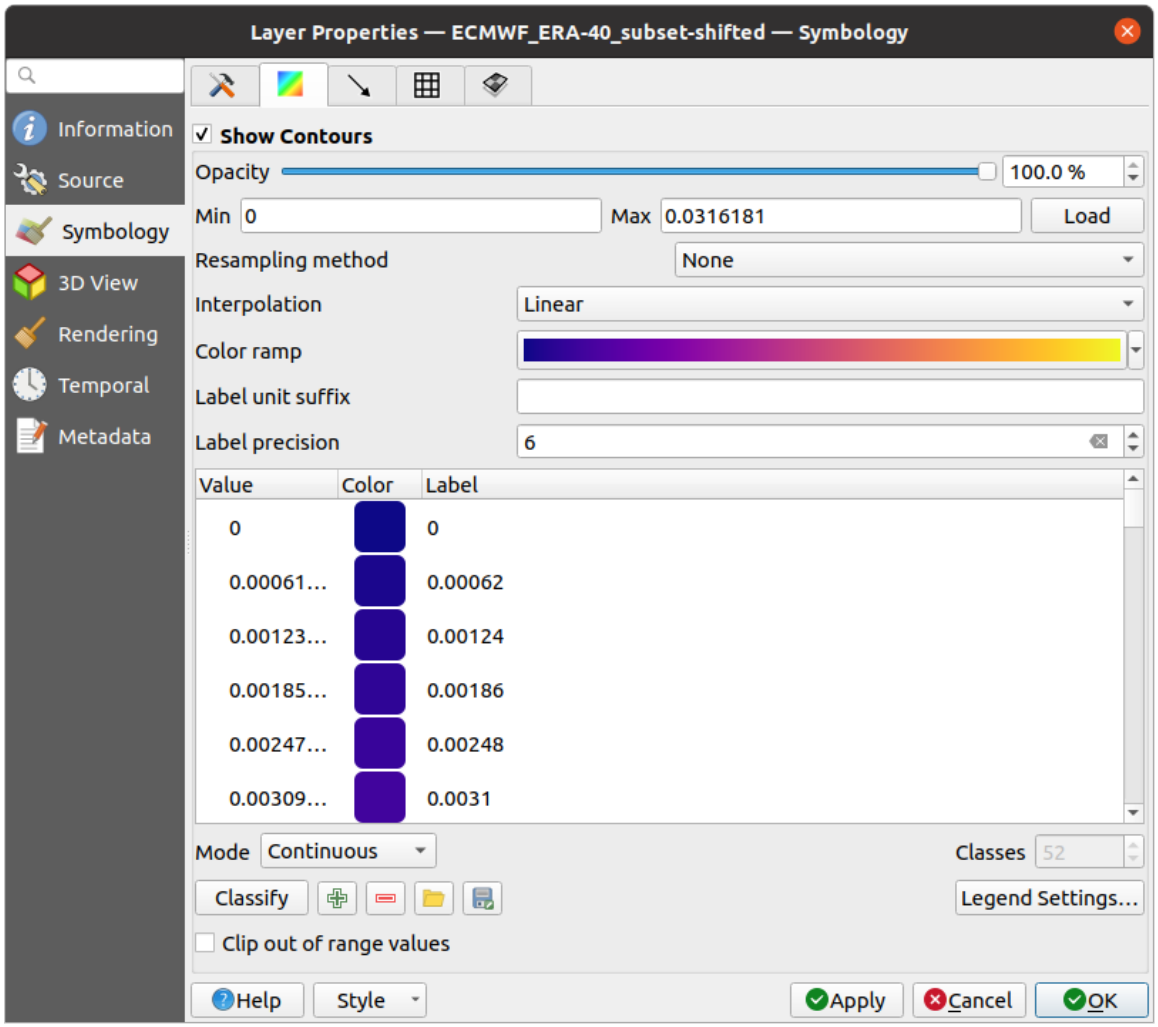


Fig. 18.7: Styling Contours in a Mesh Layer

- For 1D mesh, set the *Stroke width* of the edges. This can be a fixed size for the whole dataset, or vary along the geometry (more details with the *interpolated line renderer*)
- Use the slider or the spinbox to set the *Opacity* of the current group, if of a 2D mesh type.
- Enter the range of values you want to represent on the current group: use Load to fetch the min and max values of the current group or enter custom values if you want to exclude some.
- For 2D/3D meshes, select the *Resampling method* to interpolate the values on the surrounding vertices to the faces (or from the surrounding faces to the vertices) using the *Neighbour average* method. Depending on whether the dataset is defined on the vertices (respectively on the faces), QGIS defaults this setting to *None* (respectively *Neighbour average*) method in order to use values on vertices and keep the default rendering smooth.
- Classify the dataset using the *color ramp shader* classification.

Vectors Symbology

Notă: The Vectors tab can be activated only if a vector dataset has been selected in the *Datasets* tab.

In the Vectors tab you can see and change the current visualization options of vectors for the selected group, as shown in Fig. 18.8:

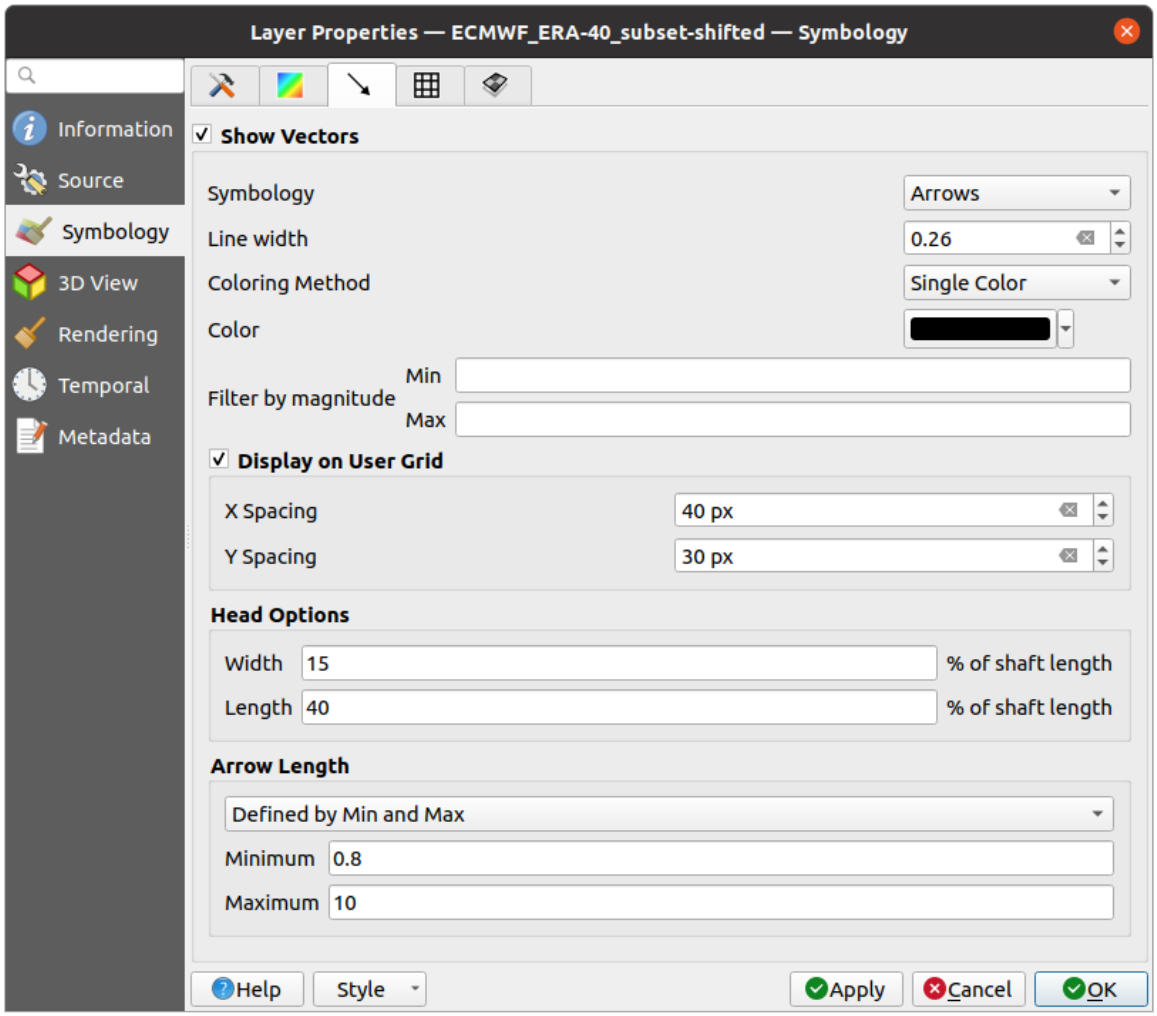


Fig. 18.8: Styling Vectors in a Mesh Layer with arrows

Mesh vector dataset can be styled using various types of *Symbology*:

- **Arrows**: vectors are represented with arrows at the same place as they are defined in the raw dataset (i.e. on the nodes or center of elements) or on a user-defined grid (hence, they are evenly distributed). The arrow length is proportional to the magnitude of the arrow as defined in the raw data but can be scaled by various methods.
- **Streamlines**: vectors are represented with streamlines seeded from start points. The seeding points can start from the vertices of the mesh, from a user grid or randomly.
- **Traces**: a nicer animation of the streamlines, the kind of effect you get when you randomly throws sand in the water and see where the sand items flows.

Available properties depend on the selected symbology as shown in the following table.

Tabelul 18.3: Availability and meaning of the vectors symbology properties

Label	Description and Properties	Arrow	Streamlines	Traces
Line width	Width of the vector representation	☑	☑	☑
Coloring method	• a *Single color* assigned to all vectors • or a variable color based on vectors magnitude, using a *Color ramp shader*	☑	☑	☑
Filter by magnitude	Only vectors whose length for the selected dataset falls between a *Min* and *Max* range are displayed	☑	☑	
Display on user grid	Places the vector on a grid with custom *X spacing* and *Y spacing* and interpolates their length based on neighbours	☑	☑	
Head options	*Length* and *Width* of the arrow head, as a percentage of its shaft length	☑		
Arrow length	• **Defined by Min and Max**: You specify the minimum and maximum length for the arrows, QGIS will interpolate their size based on the underlying vector's magnitude • **Scale to magnitude**: arrow length is proportional to their vector's magnitude • **Fixed**: all the vectors are shown with	☑		

Rendering

In the tab Rendering tab, QGIS offers possibilities to display and customize the mesh structure. *Line width* and *Line color* can be set to represent:

- the edges for 1D meshes
- For 2D meshes:
  - *Native mesh rendering*: shows original faces and edges from the layer
  - *Triangular mesh rendering*: adds more edges and displays the faces as triangles

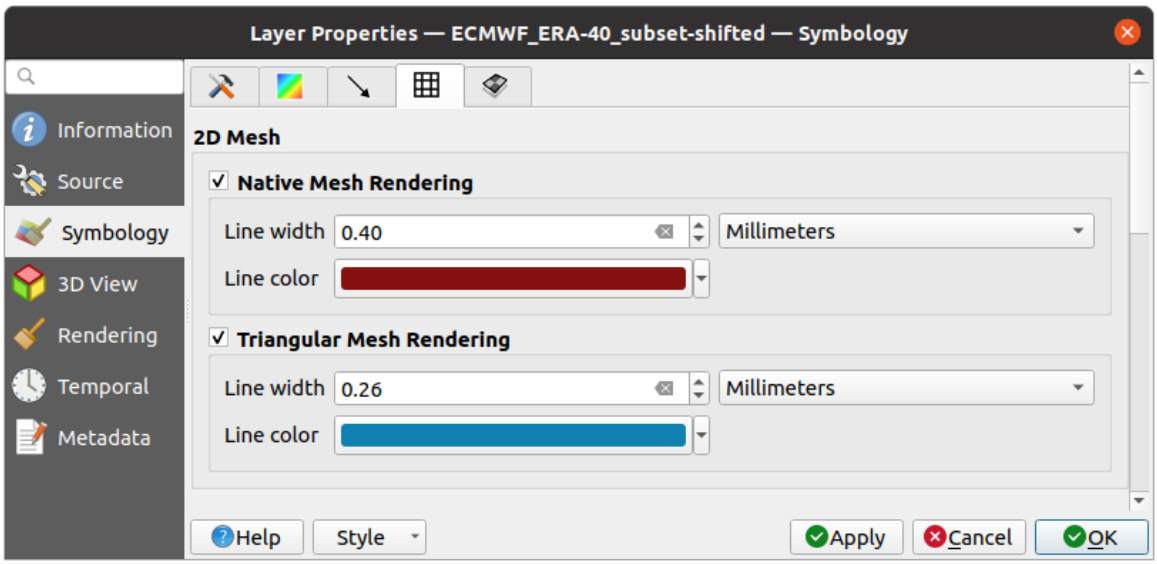


Fig. 18.9: 2D Mesh Rendering

Stacked mesh averaging method

3D layered meshes consist of multiple stacked 2D unstructured meshes each extruded in the vertical direction (`levels`) by means of a vertical coordinate. The vertices and faces have the same topology in each vertical level. Values are usually stored on the volumes that are regularly stacked over base 2d mesh. In order to visualise them on 2D canvas, you need to convert values on volumes (3d) to values on faces (2d) that can be shown in mesh layer. The Stacked mesh averaging method provides different averaging/interpolation methods to handle this.

You can select the method to derive the 2D datasets and corresponding parameters (level index, depth or height values). For each method, an example of application is shown in the dialog but you can read more on the methods at https://fvwiki.tuflow.com/index.php?title=Depth_Averaging_Results.

18.3.4 3D View Properties

Mesh layers can be used as *terrain in a 3D map view* based on their vertices Z values. From the *3D View* properties tab, it's also possible to render the mesh layer's dataset in the same 3D view. Therefore, the vertical component of the vertices can be set equal to dataset values (for example, level of water surface) and the texture of the mesh can be set to render other dataset values with color shading (for example velocity).

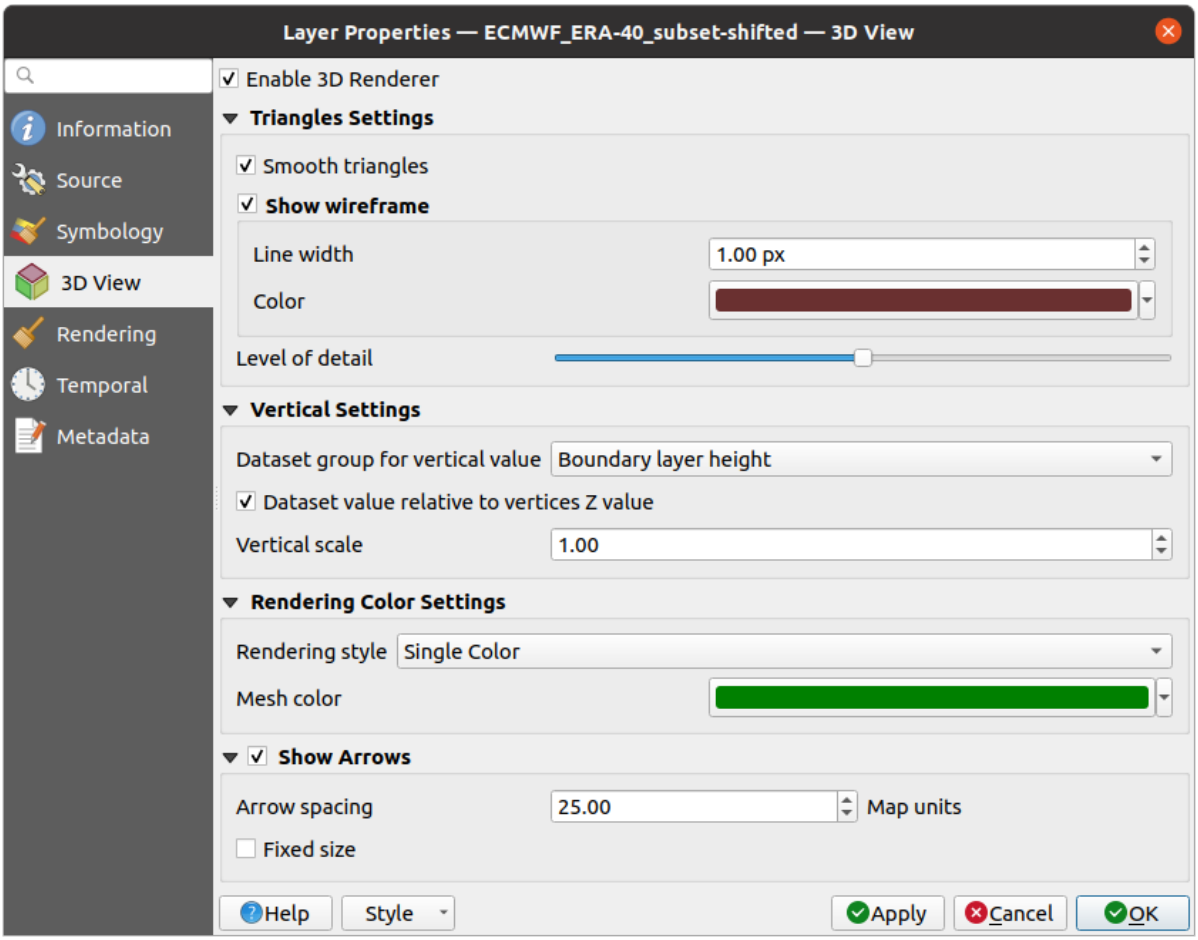


Fig. 18.10: Mesh dataset 3D properties

Check *Enable 3D Renderer* and you can edit following properties:

- Under *Triangle settings*
  - *Smooth triangles*: Angles between consecutive triangles are smoothed for a better 3D rendering
  - *Show wireframe* whose you can set the *Line width* and *Color*
  - *Level of detail*: Controls how *simplified* the mesh layer to render should be. On the far right, it is the base mesh, and the more you go left, the more the layer is simplified and is rendered with less details. This option is only available if the *Simplify mesh* option under the *Rendering* tab is activated.
- ***Vertical settings* to control behavior of the vertical component**
  of vertices of rendered triangles
  - *Dataset group for vertical value*: the dataset group that will be used for the vertical component of the mesh
  - *Dataset value relative to vertices Z value*: whether to consider the dataset values as absolute Z coordinate or relative to the vertices native Z value
  - *Vertical scale*: the scale factor to apply to the dataset Z values
- *Rendering color settings* with a *Rendering style* that can be based on the color ramp shader set in *Contours Symbology* (*2D contour color ramp shader*) or as a *Single color* with an associated *Mesh color*
- *Show arrows*: displays arrows on mesh layer dataset 3D entity, based on the same vector dataset group used in the *vector 2D rendering*. They are displayed using the 2D color setting. It's also possible to define the *Arrow spacing* and, if it's of a *Fixed size* or scaled on magnitude. This spacing setting defines also the max size of arrows because arrows can't overlap.

18.3.5 Rendering Properties

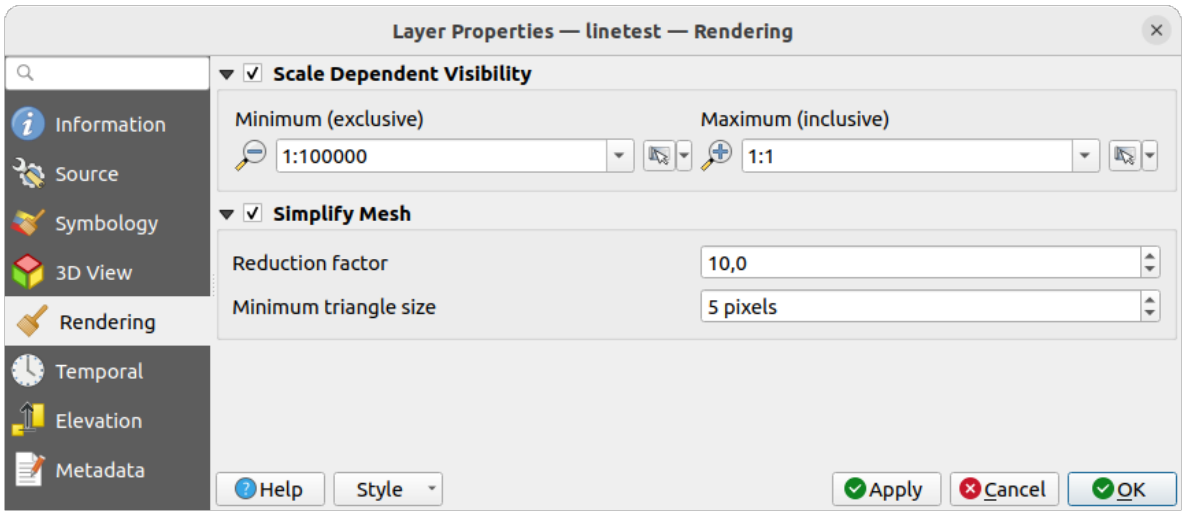


Fig. 18.11: Mesh rendering properties

Under the *Scale dependent visibility* group box, you can set the *Maximum (inclusive)* and *Minimum (exclusive)* scale, defining a range of scale in which mesh elements will be visible. Out of this range, they are hidden. The Set to current canvas scale button helps you use the current map canvas scale as boundary of the range visibility. See *Visibility Scale Selector* for more information.

Notă: You can also activate scale dependent visibility on a layer from within the *Layers* panel: right-click on the layer and in the contextual menu, select *Set Layer Scale Visibility*.

As mesh layers can have millions of faces, their rendering can sometimes be very slow, especially when all the faces are displayed in the view whereas they are too small to be viewed. To speed up the rendering, you can simplify the mesh layer, resulting in one or more meshes representing different *levels of detail* and select at which level of detail you would like QGIS to render the mesh layer. Note that the simplify mesh contains only triangular faces.

From the *Rendering* tab, check *Simplify mesh* and set:

- a *Reduction factor*: Controls generation of successive levels of simplified meshes. For example, if the base mesh has 5M faces, and the reduction factor is 10, the first simplified mesh will have approximately 500 000 faces, the second 50 000 faces, the third 5000,… If a higher reduction factor leads quickly to simpler meshes (i.e. with triangles of bigger size), it produces also fewer levels of detail.
- *Minimum triangle size*: the average size (in pixels) of the triangles that is permitted to display. If the average size of the mesh is lesser than this value, the rendering of a lower level of details mesh is triggered.

18.3.6 Temporal Properties

The *Temporal* tab provides options to control the rendering of the layer over time. It allows to dynamically display temporal values of the enabled dataset groups. Such a dynamic rendering requires the *temporal navigation* to be enabled over the map canvas.

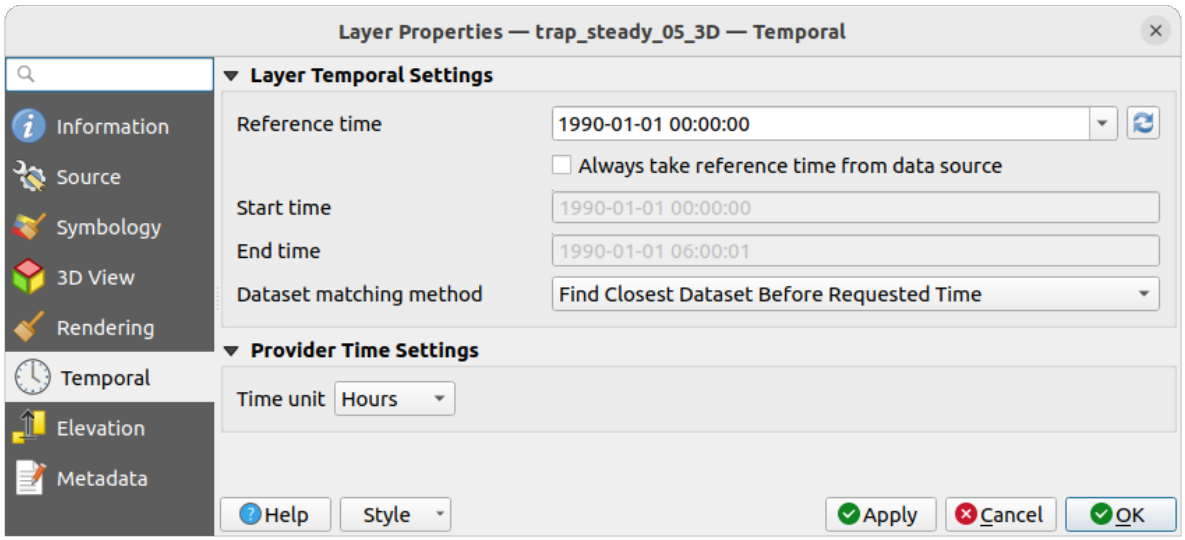


Fig. 18.12: Mesh Temporal properties

Layer temporal settings

- *Reference time* of the dataset group, as an absolute date time. By default, QGIS parses the source layer and returns the first valid reference time in the layer's dataset group. If unavailable, the value will be set by the project time range or fall back to the current date. The *Start time* and *End time* to consider are then calculated based on the internal timestamp step of the dataset.

  It is possible to set a custom *Reference time* (and then the time range), and revert the changes using the Reload from provider button. With *Always take reference time from data source* checked, you ensure that the time properties are updated from the file each time the layer is reloaded or the project reopened.

- *Dataset matching method*: determines the dataset to display at the given time. Options are *Find closest dataset before requested time* or *Find closest dataset from requested time (after or before)*.

Provider time settings

- *Time unit* extracted from the raw data, or user defined. This can be used to align the speed of the mesh layer with other layers in the project during map time navigation. Supported units are *Seconds*, *Minutes*, *Hours* and *Days*.

18.3.7 Elevation Properties

The *Elevation* tab provides options to control the layer elevation properties within a *3D map view* and its appearance in the *profile tool charts*. Specifically, you can set:

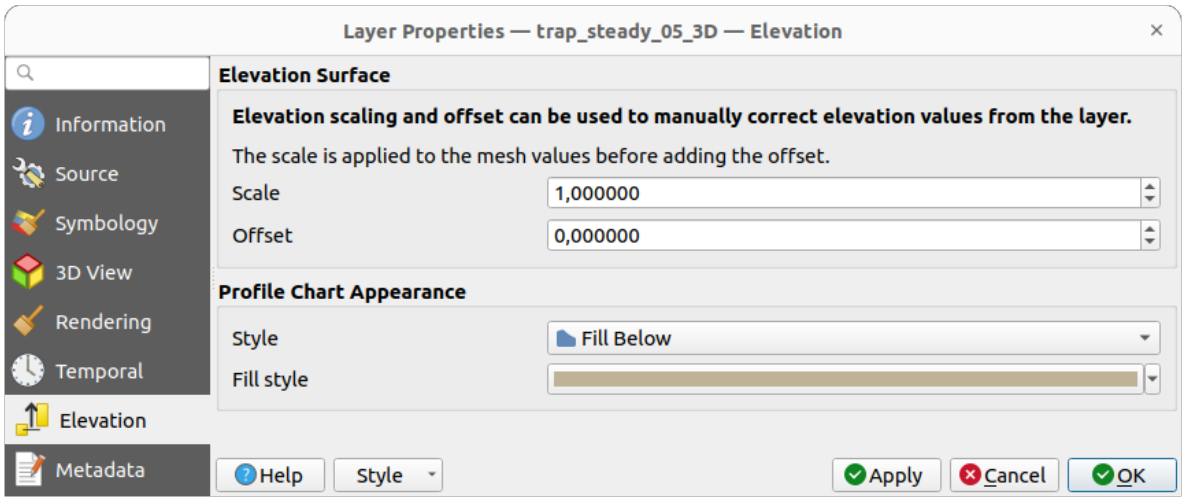


Fig. 18.13: Mesh Elevation properties

- *Elevation Surface*: how the mesh layer vertices Z values should be interpreted as terrain elevation. You can apply a *Scale* factor and an *Offset*.
- *Profile Chart Appearance*: controls the rendering of the mesh elements elevation in the profile chart. The profile *Style* can be set as:
  - a *Line* with a specific *Line style*
  - an elevation surface rendered using a fill symbol either above (*Fill above*) or below (*Fill below*) the elevation curve line. The surface symbology is represented using:
    - a *Fill style*
    - and a *Limit*: the maximum (respectively minimum) altitude determining how high the fill surface will be

18.3.8 Metadata Properties

The *Metadata* tab provides you with options to create and edit a metadata report on your layer. See *Metadata* for more information.

18.4 Editing a mesh layer

QGIS allows to *create a mesh layer* from scratch or based on an existing one. You can create/modify the geometries of the new layer whom you can assign datasets afterwards. It's also possible to edit an existing mesh layer. Because the editing operation requires a frames-only layer, you will be asked to either remove any associated datasets first (make sure you have them available if they still are necessary) or create a copy (only geometries) of the layer.

Notă: QGIS does not allow to digitize edges on mesh layers. Only vertices and faces are mesh elements that can be created. Also not all supported mesh formats can be edited in QGIS (see permissions).

18.4.1 Overview of the mesh digitizing tools

To interact with or edit a base mesh layer element, following tools are available.

Tabelul 18.4: Tools for mesh digitizing

Label	Purpose	Location
Current Edits	Access to save, rollback or cancel changes in all or selected layers simultaneously	*Digitizing* toolbar
Toggle to Edit	Turn on/off the layer in edit mode	*Digitizing* toolbar
Save Edits	Save changes done to the layer	*Digitizing* toolbar
Undo	Undo the last change(s) - `Ctrl+Z`	*Digitizing* toolbar
Redo	Redo the last undone action(s) - `Ctrl+Shift+Z`	*Digitizing* toolbar
Enable Advanced Digitizing tools	Turn on/off the *Advanced Digitizing Panel*	*Advanced Digitizing* toolbar
Reindex Faces and Vertices	Recreate index and renumber the mesh elements for optimization	*Mesh* menu
Digitize Mesh Elements	Select/Create vertices and faces	*Mesh* *Digitizing* toolbar
Select Mesh Elements by Polygon	Select vertices and faces overlapped by a drawn polygon	*Mesh* *Digitizing* toolbar
Select Mesh Elements by Expressio	Select vertices and faces using an expression	*Mesh* *Digitizing* toolbar
Transform Vertices Coordinates	Modify coordinates of a selection of vertices	*Mesh* *Digitizing* toolbar
Force by Selected Geometries	Split faces and constrain Z value using a linear geometry	*Mesh* *Digitizing* toolbar

18.4.2 Exploring the Z value assignment logic

When a mesh layer is turned into edit mode, a *Vertex Z value* widget opens at the top right of the map canvas. By default, its value corresponds to the *Default Z value* set in *Settings* ► *Options* ► *Digitizing* tab. When there are selected vertices, the widget displays the average Z value of the selected vertices.

During editing, the *Vertex Z value* is assigned to new vertices. It is also possible to set a custom value: edit the widget, press `Enter` and you will override the default value and make use of this new value in the digitizing process. Click the icon in the widget to reset its value to the Options default value.

Rules of assignment

When **creating** a new vertex, its Z value definition may vary depending on the active selection in the mesh layer and its location. The following table displays the various combinations.

Tabelul 18.5: Matrix of Z value assignment to new vertex

Vertex creation	Are there selected vertices in mesh layer?	Source of assigned value	Assigned Z Value
„Free" vertex, not connected to any face or edge of a face	No	*Vertex Z value*	Default or user defined
		Advanced Digitizing Panel (if *z* widget is in Locked state)	*z* widget if in Locked state
	Yes	*Vertex Z value*	Average of the selected vertices
Vertex on an edge	—	Mesh layer	Interpolated from the edge's vertices
Vertex on a face	—	Mesh layer	Interpolated from the face's vertices
Vertex snapped to a 2D vector feature	—	*Vertex Z value*	Default or user defined
Vertex snapped to a 3D vector vertex	—	Vector layer	Vertex
Vertex snapped to a 3D vector segment	—	Vector layer	Interpolated along the vector segment

Notă: The *Vertex Z value* widget is deactivated if the *Advanced Digitizing Panel* is enabled and no mesh element is selected. The latter's *z* widget then rules the Z value assignment.

Modifying Z value of existing vertices

To modify the Z value of vertices, the most straightforward way is:

1. Select one or many vertices. The *Vertex Z value* widget will display the average height of the selection.
2. Change the value in the widget.
3. Press `Enter`. The entered value is assigned to the vertices and becomes the default value of next vertices.

Another way to change the Z value of a vertex is to move and snap it on a vector layer feature with the Z value capability. If more than one vertex are selected, the Z value can't be changed in this way.

The *Transform mesh vertices* dialog also provides means to modify the Z value of a selection of vertices (along with their X or Y coordinates).

18.4.3 Selecting mesh elements

Using *Digitize Mesh Elements*

Activate the Digitize Mesh Elements tool. Hover over an element and it gets highlighted, allowing you to select it.

- Click on a vertex, and it is selected.
- Click on the small square at the center of a face or an edge, and it gets selected. Connected vertices are also selected. Conversely, selecting all the vertices of an edge or a face also selects that element.
- Drag a rectangle to select overlapping elements (a selected face comes with all their vertices). Press `Alt` key if you want to select only completely contained elements.
- To add elements to a selection, press `Shift` while selecting them.

- To remove an element from the selection, press `Ctrl` and reselect it. A deselected face will also deselect all their vertices.

Using *Select Mesh Elements by Polygon*

Activate the Select Mesh Elements by Polygon tool and:

- Draw a polygon (left-click to add vertex, `Backspace` to undo last vertex, `Esc` to abort the polygon and right-click to validate it) over the mesh geometries. Any partially overlapping vertices and faces will get selected. Press `Alt` key while drawing if you want to select only completely contained elements.
- Right-click over the geometry of a vector layer's feature, select it in the list that pops up and any partially overlapping vertices and faces of the mesh layer will get selected. Use `Alt` while drawing to select only completely contained elements.
- To add elements to a selection, press `Shift` while selecting them.
- To remove an element from the selection, press `Ctrl` while drawing over the selection polygon.

Using *Select Mesh Elements by Expression*

Another tool for mesh elements selection is Select Mesh Elements by Expression. When pressed, the tool opens the mesh *expression selector dialog* from which you can:

1. Select the method of selection:
   - *Select by vertices*: applies the entered expression to vertices, and returns matching ones and their eventually associated edges/faces
   - *Select by faces*: applies the entered expression to faces, and returns matching ones and their associated edges/vertices
2. Write the expression of selection. Depending on the selected method, available functions in the *Meshes group* will be filtered accordingly.
3. Run the query by setting how the selection should behave and pressing:
   - *Select*: replaces any existing selection in the layer
   - *Add to current selection*
   - *Remove from current selection*

18.4.4 Modifying mesh elements

Adding vertices

To add vertices to a mesh layer:

1. Press the Digitize mesh elements button
2. A *Vertex Z value* widget appears on the top right corner of the map canvas. Set this value to the Z coordinate you would like to assign to the subsequent vertices
3. Then double-click:
   - outside a face: adds a „free vertex", that is a vertex not linked to any face. This vertex is represented by a red dot when the layer is in editing mode.
   - on the edge of existing face(s): adds a vertex on the edge, splits the touching face(s) into triangles connected to the new vertex.

- inside a face: splits the face into triangles whose edges connect the surrounding vertices to the new vertex.

Adding faces

To add faces to a mesh layer:

1. Press the Digitize mesh elements button
2. A *Vertex Z value* widget appears on the top right corner of the map canvas. Set this value to the Z coordinate you would like to assign to the subsequent vertices.
3. Hover over a vertex and click the small triangle that appears next it.
4. Move the cursor to the next vertex position; you can snap to existing vertex or left-click to add a new one.
5. Proceed as above to add as many vertices you wish for the face. Press `Backspace` button to undo the last vertex.
6. While moving the mouse, a rubberband showing the shape of the face is displayed. If it is shown in green, then the expected face is valid and you can right-click to add it to the mesh. If it is red, the face is not valid (e.g. because it self-intersects, overlaps an existing face or vertex, creates a hole, …) and can't be added. You'd need to undo some vertices and fix the geometry.
7. Press `Esc` to abort the face digitizing.
8. Right-click to validate the face.

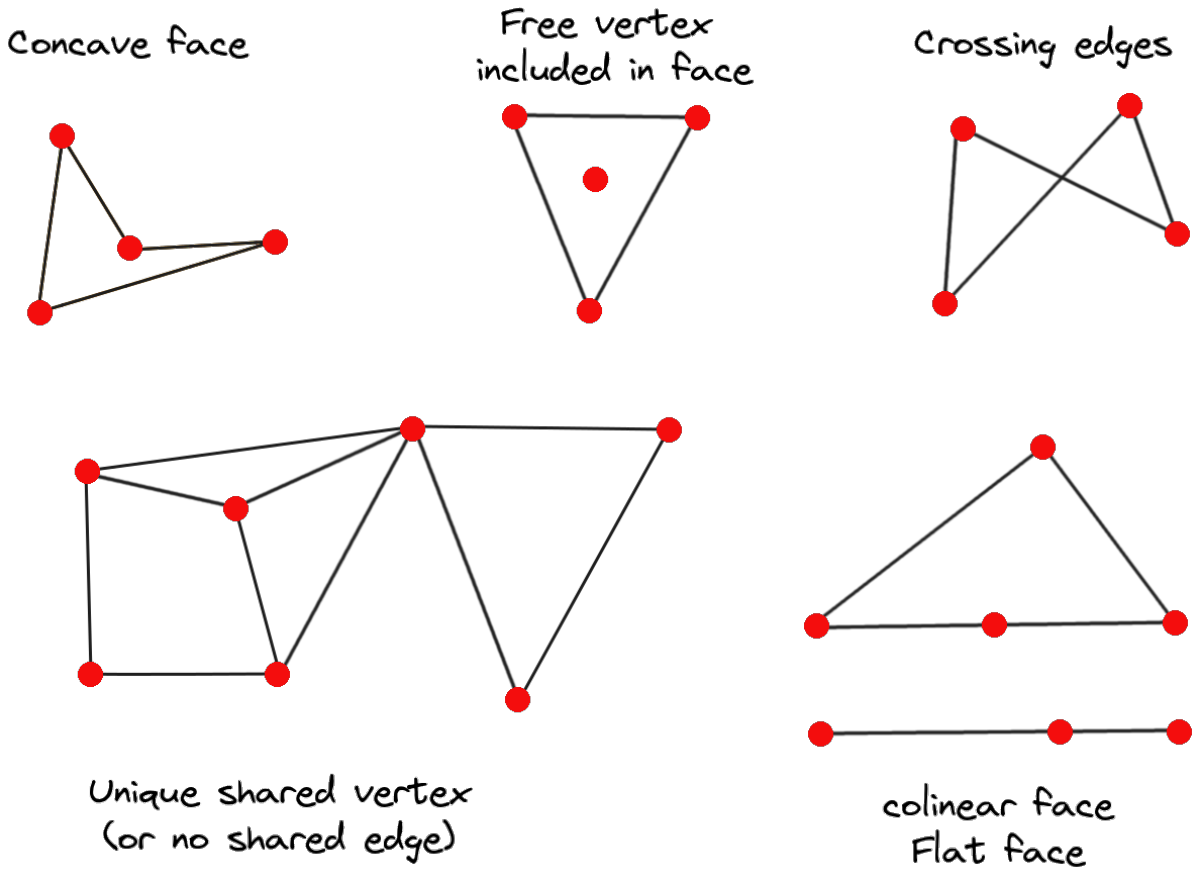


Fig. 18.14: Examples of invalid mesh

Removing mesh elements

1. *Select the target elements*
2. Enable the Digitize mesh elements tool
3. Right-click and select:
   - *Remove Selected Vertices and Fill Hole(s)* or press `Ctrl+Del`: removes vertices and linked faces and fills the hole(s) by triangulating from the neighbor vertices
   - *Remove Selected Vertices Without Filling Hole(s)* or press `Ctrl+Shift+Del`: removes vertices and linked faces and do not fill hole(s)
   - *Remove Selected Face(s)* or press `Shift+Del`: removes faces but keeps the vertices

These options are also accessible from the contextual menu when hovering over a single item without selecting.

Moving mesh elements

To move vertices and faces of a mesh layer:

1. *Select the target elements*
2. Enable the Digitize mesh elements tool
3. To start moving the element, click on a vertex or the centroid of a face/edge
4. Move the cursor to the target location (snapping to vector features is supported).
5. If the new location does not generate an *invalid mesh*, the moved elements appear in green. Click again to release them at this location. Faces whose vertices are all selected are translated, their neighbors are reshaped accordingly.

Transforming mesh vertices

The Transform Vertices Coordinates tool gives a more advanced way to move vertices, by editing their X, Y and/or Z coordinates thanks to expressions.

1. Select the vertices you want to edit the coordinates
2. Press Transform Vertices Coordinates. A dialog opens with a mention of the number of selected vertices. You can still add or remove vertices from the selection.
3. Depending on the properties you want to modify, you need to check the *X coordinate*, *Y coordinate* and/or *Z value*.
4. Then enter the target position in the box, either as a numeric value or an expression (using the Expression dialog)
5. With the Import Coordinates of the Selected Vertex pressed, the X, Y and Z boxes are automatically filled with its coordinates whenever a single vertex is selected. A convenient and quick way to adjust vertices individually.
6. Press *Preview Transform* to simulate the vertices new location and preview the mesh with transformation.
   - If the preview is green, transformed mesh is valid and you can apply the transformation.
   - If the preview is red, the transformed mesh is invalid and you can not apply the transformation until it is corrected.
7. Press *Apply Transform* to modify the selected coordinates for the set of vertices.

Reshaping mesh geometry

The contextual menu

1. Enable the Digitize mesh elements
2. Select mesh item(s), or not
3. Hover over a mesh element, it gets highlighted.
4. Right-click and you can:
   - *remove the item(s)*
   - *Split Selected Face(s)* (*Split Current Face*): splits the face you are hovering over or each selected quad mesh faces into two triangles
   - *Delaunay Triangulation with Selected vertices*: builds triangular faces using selected free vertices.
   - *Refine Selected Face(s)* (*Refine Current Face*): splits the face into four faces, based on vertices added at the middle of each edge (a triangle results into triangles, a quad into quads). Also triangulates adjacent faces connected to the new vertices.

The edge markers

When the Digitize mesh elements is active and you hover over an edge, the edge is highlighted and it is possible to interact with it. Depending on the context, following markers may be available:

- a **square**, at the center of the edge: click on it to select extremity vertices.
- a **cross** if the two faces on either side can be merged: click on it to delete the edge and merge the faces.
- a **circle** if the edge is between two triangles: Click on it to flip the edge, i.e. connect it instead to the two other „free" vertices of the faces

The *Force by Selected Geometries* tool

The Force by Selected Geometries tool provides advanced ways to apply break lines using lines geometry. A break line will force the mesh to have edges along the line. Note that the break line will not be considered persistent once the operation is done; resulting edges will not act as constraints anymore and can be modified like any other edge. This can be used for example to locally modify a mesh layer with accurate lines, as river banks or border of road embankments.

1. Enable the Force by Selected Geometries tool
2. Indicate the geometry to use as „forcing line"; it can be:
   - picked from a line or polygon feature in the map canvas: right-click over the vector feature and select it from the list in the contextual menu.
   - a virtual line drawn over the mesh frame: left-click to add vertices, right-click for validation. Vertices Z value is set through the *Vertex Z value* widget or the *z* widget if the *Advanced Digitizing Panel* is on. If the line is snapped to a mesh vertex or a 3D vector feature's vertex or segment, the new vertex takes the snapped element Z value.

Mesh faces that overlap the line geometry or the polygon's boundary will be affected in a way that depends on options you can set from the Force by Selected Geometries tool drop-down menu:

- *Add new vertex on intersecting edges*: with this option, a new vertex is added each time the forcing line intersect an edge. This option leads to split along the line each encountered faces.

Without this option, encountered faces are removed and replaced by faces coming from a triangulation with only the existing vertices plus the vertices of the forcing lines (new vertices are also added on the boundary edge intersecting the forcing lines).

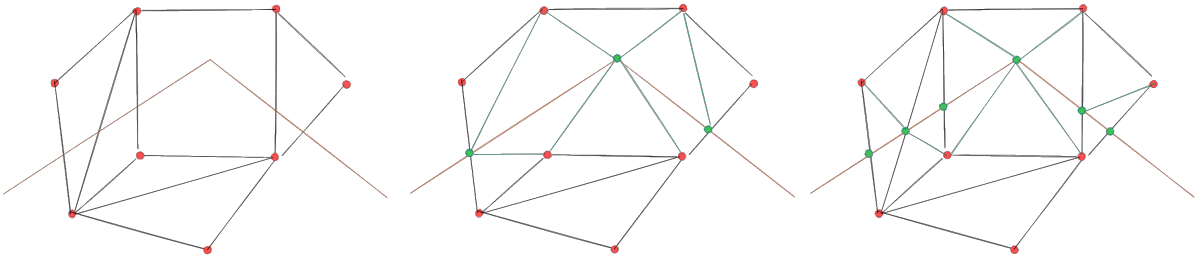


Fig. 18.15: Force Mesh using a line geometry - Results without (middle) and with (right) new vertex on edges intersection

- *Interpolate Z value from*: set how the new vertices Z value is calculated. It can be from:
  - the *Mesh* itself: the new vertices Z value is interpolated from vertices of the face they fall within
  - or the *Forcing line*: if the line is defined by a 3D vector feature or a drawn line then the new vertices Z value is derived from its geometry. In case of 2D line feature, the new vertices Z value is the *Vertex Z value*.
- *Tolerance*: when an existing mesh vertex is closer to the line than the tolerance value, do not create new vertex on the line but use the existing vertex instead. The value can be set in *Meters at Scale* or in *Map Units* (more details at *Unit Selector*).

18.4.5 Reindexing meshes

During edit, and in order to allow quick undo/redo operations, QGIS keeps empty places for deleted elements, which may lead to growing memory use and inefficient mesh structuring. The *Mesh* ► *Reindex Faces and Vertices* tool is designed to remove these holes and renumber the indices of faces and vertices so that they are continuous and somewhat reasonably ordered. This optimizes relation between faces and vertices and increases the efficiency of calculation.

Notă: The *Reindex Faces and Vertices* tool saves the layer and clear the undo/redo stacks, disabling any rollback.

18.5 Mesh Calculator

The *Mesh Calculator* tool from the top *Mesh* menu allows you to perform arithmetic and logical calculations on existing dataset groups to generate a new dataset group (see Fig. 18.16).

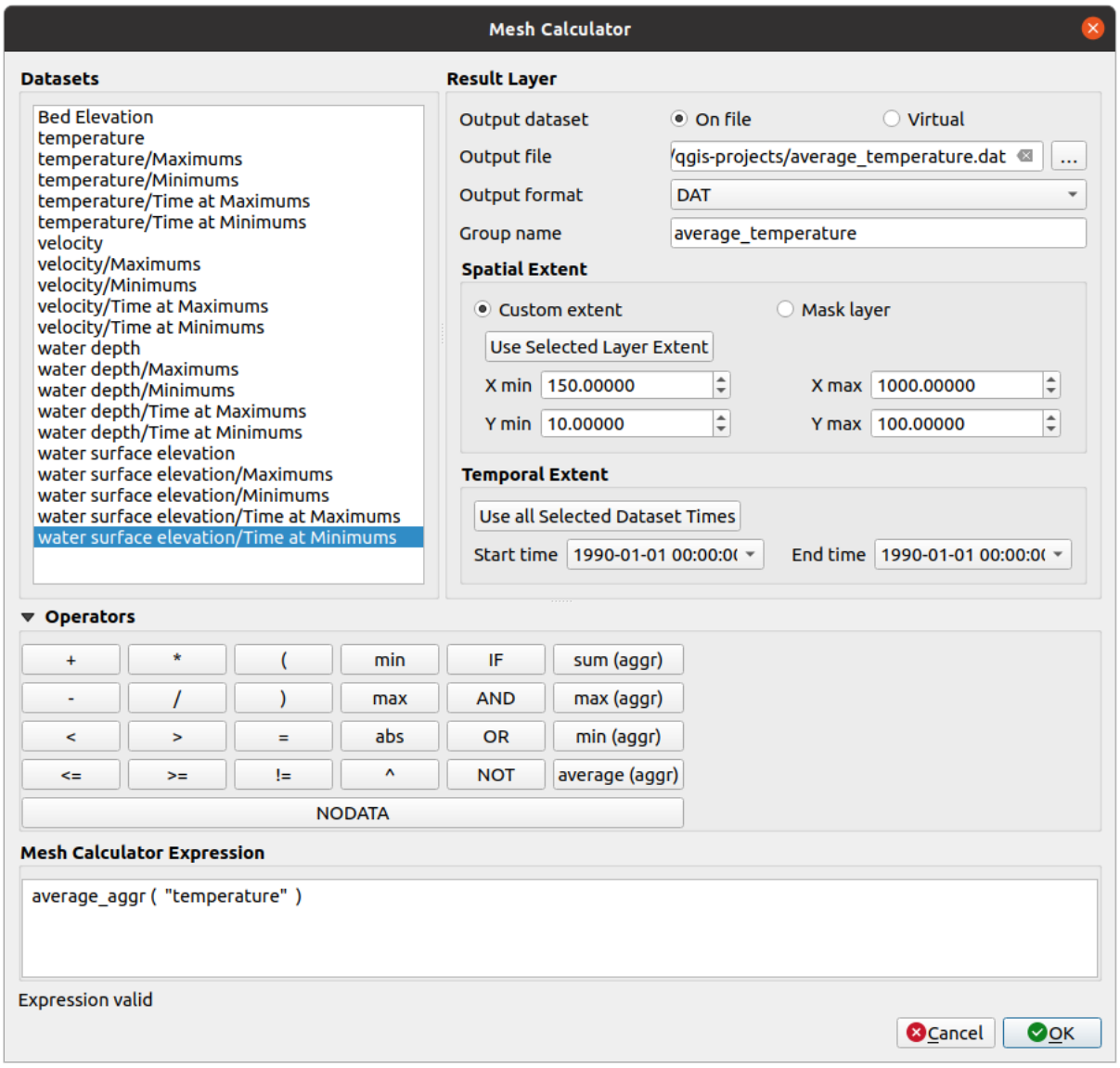


Fig. 18.16: Mesh Calculator

The *Datasets* list contains all dataset groups in the active mesh layer. To use a dataset group in an expression, double click its name in the list and it will be added to the *Mesh calculator expression* field. You can then use the operators to construct calculation expressions, or you can just type them into the box.

The *Result Layer* helps you configure properties of the output layer:

- ☑ *Create on-the-fly dataset group instead of writing layer to disk*:
  - If unchecked, the output is stored on disk as a new plain file. An *Output File* path and an *Output Format* are required.
  - If checked, a new dataset group will be added to the mesh layer. Values of the dataset group are not stored in memory but each dataset is calculated when needed with the formula entered in the mesh calculator. That virtual dataset group is saved with the project, and if needed, it can be removed or made persistent in file from the layer *Source* properties tab.

In either case, you should provide a *Group Name* for the output dataset group.

- The *Spatial extent* to consider for calculation can be:
  - a *Custom extent*, manually filled with the *X min*, *X max*, *Y min* and *Y max* coordinate, or extracted from an existing dataset group (select it in the list and press *Use selected layer extent* to fill the abovementioned coordinate fields)
  - defined by a polygon layer (*Mask layer*) of the project: the polygon features geometry are used to clip the mesh layer datasets
- The *Temporal extent* to take into account for datasets can be set with the *Start time* and *End time* options, selected from the existing dataset groups timesteps. They can also be filled using the *Use all selected dataset times* button to take the whole range.

The *Operators* section contains all available operators. To add an operator to the mesh calculator expression box, click the appropriate button. Mathematical calculations (`+`, `-`, `*`, ...) and statistical functions (`min`, `max`, `sum (aggr)`, `average (aggr)`, ...) are available. Conditional expressions (`=`, `!=`, `<`, `>=`, `IF`, `AND`, `NOT`, ...) return either 0 for false and 1 for true, and therefore can be used with other operators and functions. The `NODATA` value can also be used in the expressions.

The *Mesh Calculator Expression* widget shows and lets you edit the expression to execute.

CAPITOLUL 19

Working with Vector Tiles

19.1 What are Vector Tiles?

Vector tiles are packets of geographic data, packaged into pre-defined roughly-square shaped „tiles” for transfer over the web. They combine pre-rendered raster map tiles and vector map tiles. The vector tile server returns vector map data, which has been clipped to the boundaries of each tile, instead of a pre-rendered map image. The clipped tiles represent the zoom-levels of the vector tile service, derived from a pyramid approach. Using this structure, the data-transfer is reduced in comparison to un-tiled vector maps. Only data within the current map view, and at the current zoom level need to be transferred. Also, compared to a tiled raster map, data transfer is also greatly reduced, as vector data is typically much smaller than a rendered bitmap. Vector tiles do not have any styling information assigned so QGIS needs to apply a cartographic style in order to display the data.

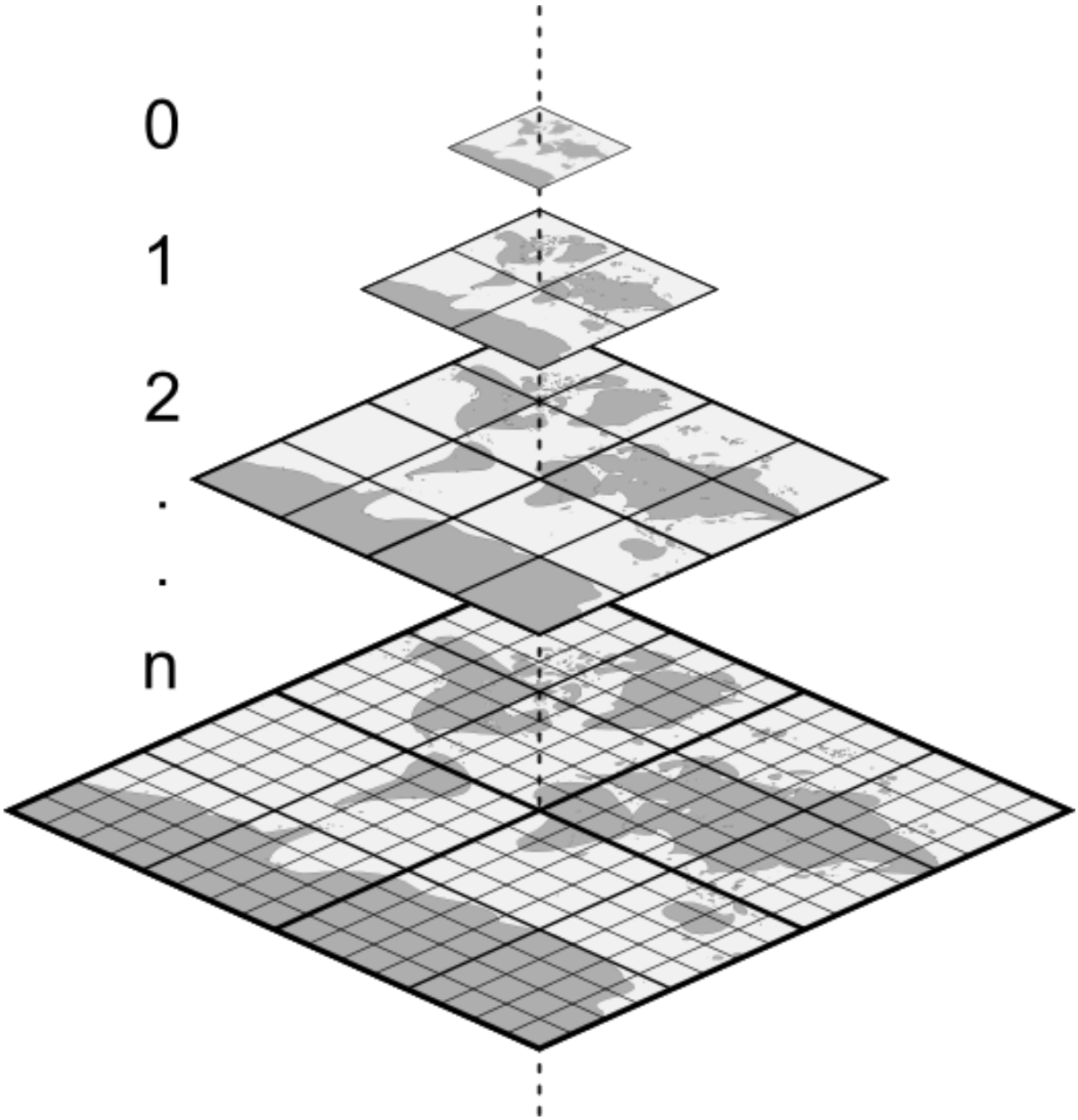


Fig. 19.1: Pyramid structure of vector tiles with zoom-levels

19.2 Supported Formats

There is support for vector tiles through:

- remote sources (HTTP/S) - with XYZ template - `type=xyz&url=http://example.com/{z}/{x}/{y}.pbf`
- local files - with XYZ template - e.g. `type=xyz&url=file:///path/to/tiles/{z}/{x}/{y}.pbf`
- local MBTiles database - e.g. `type=mbtiles&url=file:///path/to/file.mbtiles`

To load a vector tiles dataset into QGIS, use the *Vector Tile* tab in the *Data Source Manager* dialog. Read *Using Vector Tiles services* for more details.

19.3 Vector Tiles Dataset Properties

The vector tiles *Layer Properties* dialog provides the following sections:

[1] Also available in the *Layer styling panel*

19.3.1 Information Properties

The *Information* tab is read-only and represents an interesting place to quickly grab summarized information and metadata on the current layer. Provided information are:

- based on the provider of the layer: name, URI, source type and path, number of zoom levels
- the Coordinate Reference System: name, units, method, accuracy, reference (i.e. whether it's static or dynamic)
- picked from the *filled metadata*: access, extents, links, contacts, history…

19.3.2 Source Properties

The *Source* tab displays basic information about the selected vector tile, including:

- the *Layer name* to display in the *Layers Panel*;
- the *Coordinate Reference System*: Displays the layer's *Coordinate Reference System (CRS)*. You can change the layer's CRS, by selecting a recently used one in the drop-down list or clicking on the Select CRS button (see *Coordinate Reference System Selector*). Use this process only if the layer CRS is wrong or not specified.

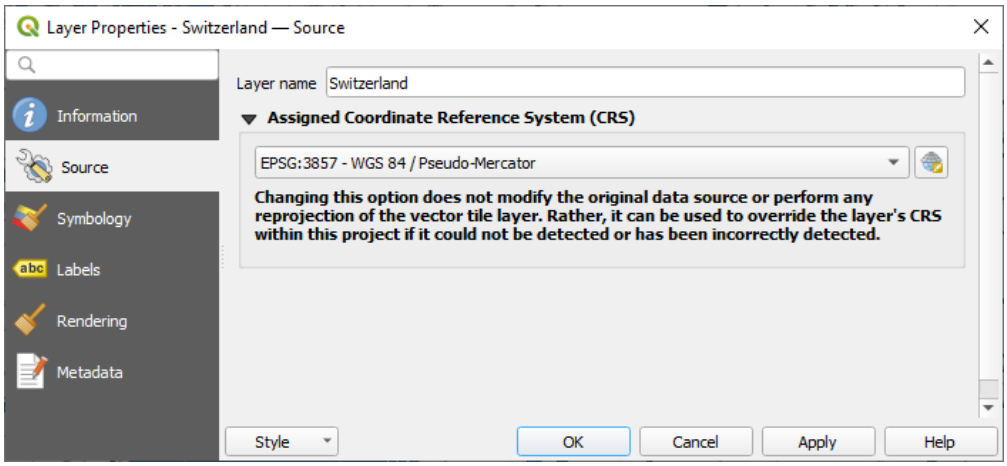


Fig. 19.2: Vector Tiles Properties - Source Dialog

19.3.3 Symbology and Labels Properties

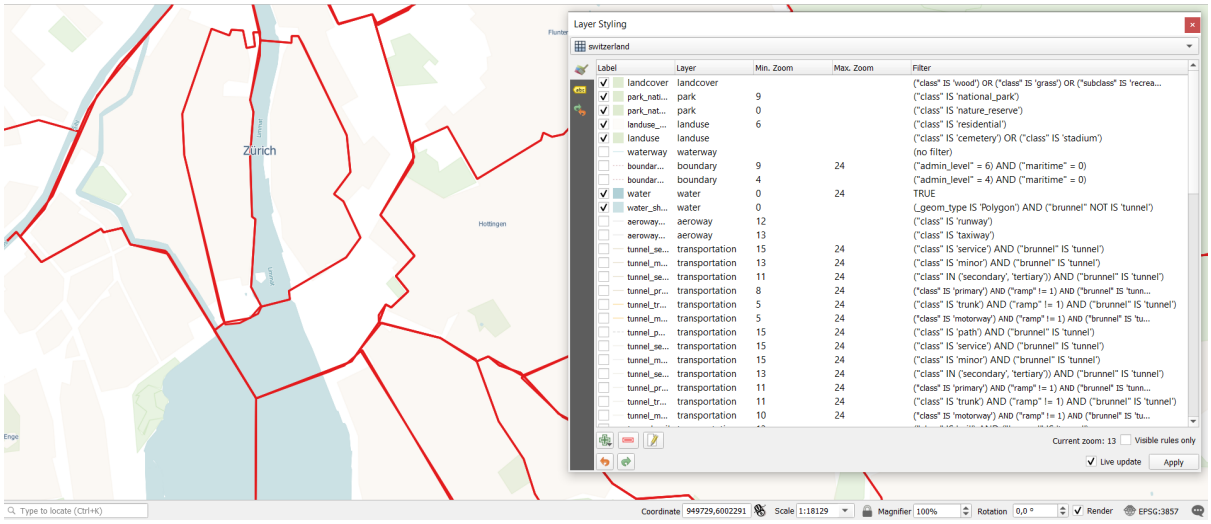


Fig. 19.3: Vector Tile Layer Symbology

Setting rules

As vector tiles consist of point, line and polygon geometries, the respective symbols are available. To apply a cartographic style (with symbology and/or labels), you can either:

- Use a *Style URL* when creating the *Vector Tiles Connection*. The symbology will be shown immediately in the *Symbology* tab after the layer is loaded in QGIS.
- Or build your own symbology and labeling in the corresponding tabs of the layer properties. By default, QGIS assigns an identical symbol to the features based on their geometry type.

In both cases, setting a style for a vector tile relies on a set of *rules* applied to the features, indicating:

- a vector *symbol* or *label*
- a *Label*, a title for comprehensive identification of the rule
- the name of a particular *Layer* the rule should apply to, if not applied to `(all layers)`
- a *Min. Zoom* and a *Max. Zoom*, for the range of display. symbology and labelling can be dependent on the zoom level.
- a *Filter*, a QGIS expression to identify the features to apply the style to

Each rule is added pressing the Add rule button and selecting the type of symbols (*Marker*, *Line*, *Fill*) corresponding to the features geometry type. You can as well Remove selected rules or Edit current rule.

At the bottom the *Current Zoom* is shown. Check the *Visible rules only* option at the top of the dialog to filter the list of rules to only those that are visible at the current zoom level. This makes it easier to work with complex vector styling and to locate troublesome rules. The *Filter rules* text box also helps you easily find a rule, by searching the *Label*, *Layer* and *Filter* fields.

In Fig. 19.3 we set up style for the OpenStreetMap `landuse` layer. For better visibility most of the rules are deselected.

Layer rendering

From the *Symbology* tab, you can also set some options that invariably act on all features of the layer:

- *Opacity*: You can make the underlying layer in the map canvas visible with this tool. Use the slider to adapt the visibility of your vector layer to your needs. You can also make a precise definition of the percentage of visibility in the menu beside the slider.
- *Blending mode*: You can achieve special rendering effects with these tools that you may previously only know from graphics programs. The pixels of your overlaying and underlaying layers are mixed through the settings described in *Modurile de Fuziune*.

Styles

Available at the bottom of most of the tabs, the *Styles* ► menu provides shortcuts to save, load, create, switch styles to apply to the vector tiles. Vector tiles can have their style saved from QGIS as *QML* files and they can be imported as:

- *QML* files (*QML - Formatul Fișierelor de Stilizare*)
- *MapBox GL Json* style configuration files

More details at *Salvarea și Partajarea Proprietăților Stratului*.

19.3.4 Rendering Properties

Under *Scale dependent visibility*, you can set the *Maximum (inclusive)* and *Minimum (exclusive)* scales, defining a range of scales in which features will be visible. Out of this range, they are hidden. The Set to current canvas scale button helps you use the current map canvas scale as boundary of the range visibility. See *Visibility Scale Selector* for more information.

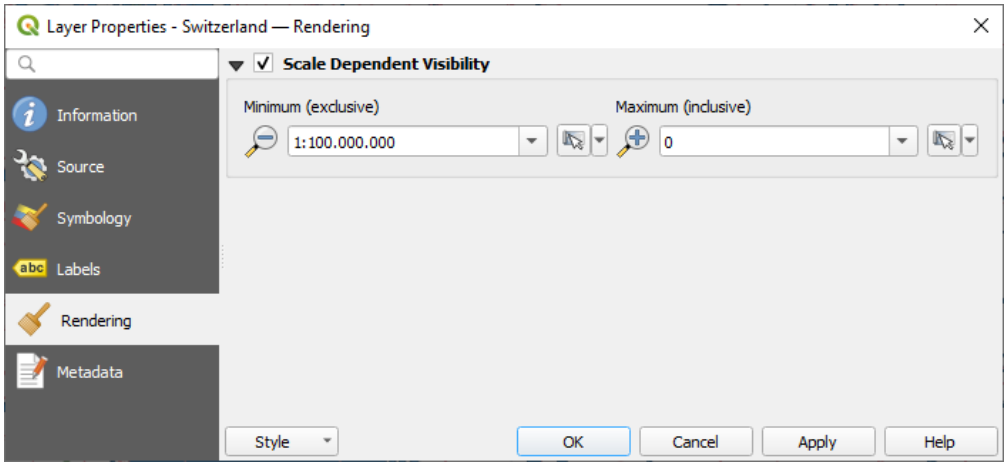


Fig. 19.4: Vector Tiles Properties - Rendering Dialog

19.3.5 Metadata Properties

The *Metadata* tab provides you with options to create and edit a metadata report on your layer. See *Metadata* for more information.

CAPITOLUL 20

Working with Point Clouds

20.1 Introduction to Point Clouds

What is A Point Cloud?

A point cloud is a three-dimensional image of a space made up of many individual of data points (up to billions, even trillions). Each of the points has an x, y and z coordinate. Depending on the capture method, point clouds usually also have additional attributes that came from the capture, such as color values or intensity. These attributes can be used, for instance, to display point clouds in different colors. In QGIS, a point cloud can be used to generate a three-dimensional image of the landscape (or of another space).

Supported Formats

QGIS supports the data formats Entwine Point Tile (EPT) and LAS/LAZ. To work with point clouds, QGIS always saves the data in EPT. EPT is a storage format that consists of several files stored in a common folder. To allow quick access to the data, EPT uses indexing. For more information on the EPT format, see entwine homepage

If the data is in LAS or LAZ format, QGIS will convert it to EPT when it is loaded for the first time. Depending on the size of the file, this may take some time. In this process, a subfolder is created in the folder in which the LAS/LAZ file is located according to the scheme `ept_` + `name_LAS/LAZ_file`. If such a subfolder already exists, QGIS loads the EPT immediately (which leads to a reduced loading time).

Worth Knowing

In QGIS it is not (yet) possible to edit point clouds. If you want to manipulate your point cloud, you can use CloudCompare, an open source point cloud processing tool. Also the Point Data Abstraction Library (PDAL - similar to GDAL) offers you options to edit point clouds (PDAL is command line only).

Due to the large number of data points, it is not possible to display an attribute table of point clouds in QGIS. However, the *Identify tool* supports point clouds, so you can display all attributes, even of a single data point.

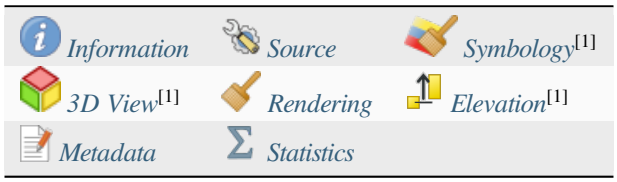
20.2 Point Clouds Properties

The *Layer Properties* dialog for a point cloud layer offers general settings for the layer and its rendering. It also provides information about the layer.

To access the *Layer Properties* dialog:

- In the *Layers* panel, double-click the layer or right-click and select *Properties…* from the context menu;
- Go to *Layer* ► *Layer Properties…* menu when the layer is selected.

The point cloud *Layer Properties* dialog provides the following sections:



[1] Also available in the *Layer styling panel*

Notă: Most of the properties of a point cloud layer can be saved to or loaded from a `.qml` file using the *Style* menu at the bottom of the properties dialog. More details at *Salvarea și Partajarea Proprietăților Stratului*

20.2.1 Information Properties

The *Information* tab is read-only and represents an interesting place to quickly grab summarized information and metadata on the current layer. Provided information are:

- General such as name in the project, source path, last save time and size, the used provider
- Based on the provider of the layer: extent and number of points
- The Coordinate Reference System: name, units, method, accuracy, reference (i.e. whether it's static or dynamic)
- Metadata delivered by the provider: creation date, version, data format, scale X/Y/Z, …
- Picked from the *Metadata* tab (where they can be edited): access, extents, links, contacts, history…

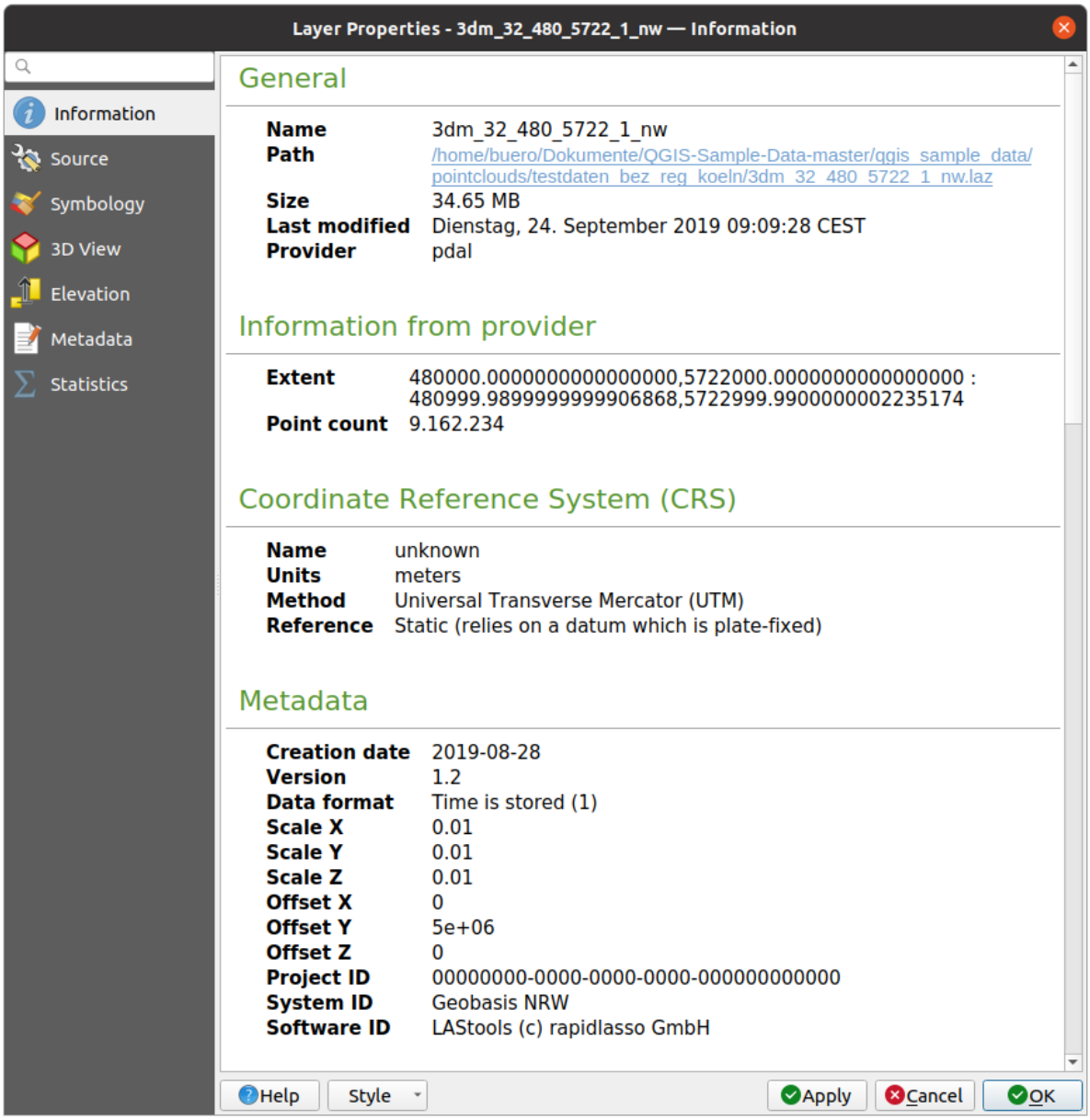


Fig. 20.1: Point cloud information tab

20.2.2 Proprietățile Sursei

In the *Source* tab you can see and edit basic information about the point cloud layer:

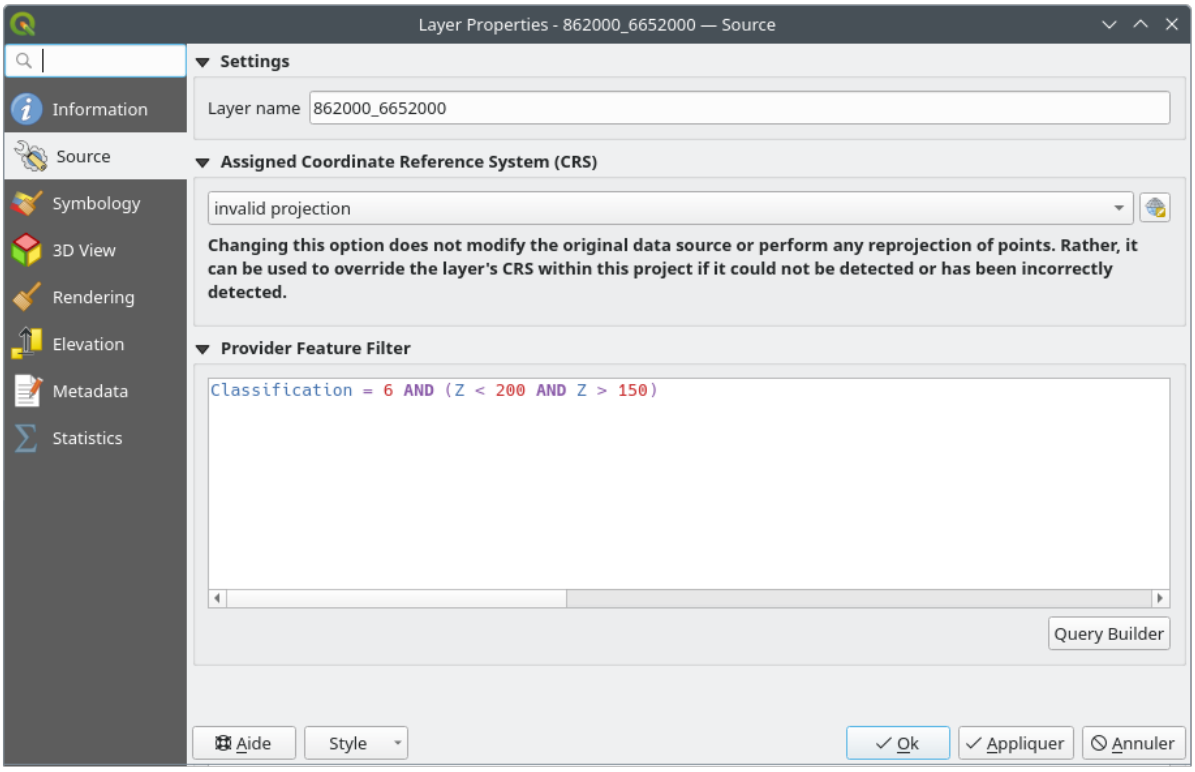


Fig. 20.2: Point cloud source tab

- *Settings*: Set a Layer name different from the layer filename that will be used to identify the layer in the project (in the Layers Panel, with expressions, in print layout legend,…)
- *Assigned Coordinate Reference System (CRS)*: Here you can change the layer's assigned *Coordinate Reference System*, selecting a recently used one in the drop-down list or clicking on set Projection Select CRS button (see *Coordinate Reference System Selector*). Use this process only if the CRS applied to the layer is a wrong one or if none was applied.
- *Provider Feature Filter*: allows to restrict data to use in the current project to a subset, using functions and expression at the PDAL data provider level. Press the *Query Builder* button at the bottom to start setting the filter.

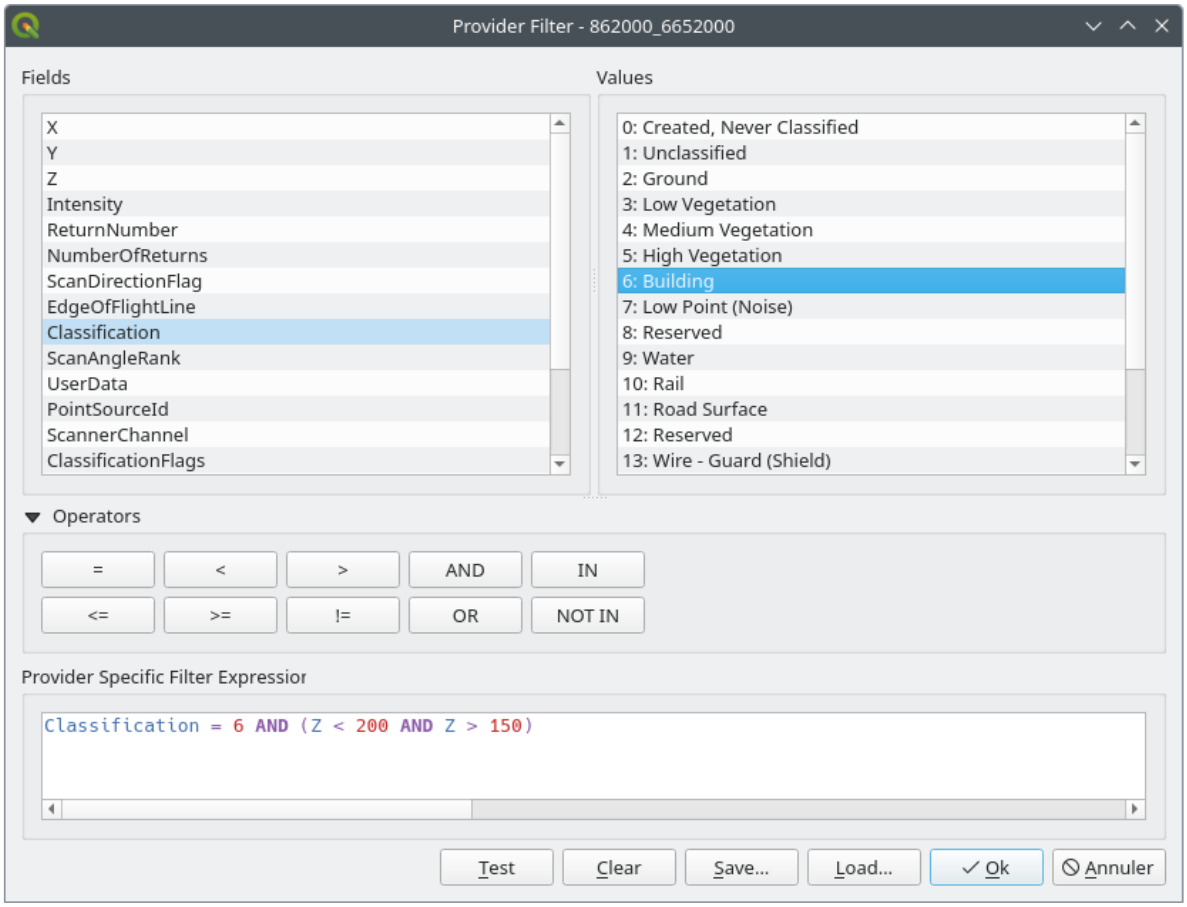


Fig. 20.3: Filtering point cloud layer to buildings at specific altitude

In the bottom part of the dialog, you can type a *Provider specific filter expression*. Such an expression can be built thanks to:

- *Fields*: the list contains all the attributes of the point cloud layer. To add an attribute to the expression, double-click its name or just type it into the text box.
- *Values*: frame lists values or statistics of the currently selected attribute, as a `key:value` pair. To add a value to the expression field, double click its name in the list: depending on the type of attribute, either the key or the value will be added to the expression. Or type the value in the expression text box.
- *Operators*: this toolbar contains all usable operators. To add an operator to the expression field, click the appropriate button. Relational operators (= , > , …) and logical operators (`AND`, `OR`, …) are available.

The *Test* button helps you check the syntax of your query. Use the *Clear* button to wipe the query and revert the layer to its original state (i.e., fully load all the points in the layer). It is possible to *Save…* the query as a `.QQF` file, or *Load…* the query from a file into the dialog.

When a filter is applied, QGIS treats the resulting subset as if it were the entire layer. For example if you applied the *filter above* for filtering buildings, you can not e.g. display, query, save or edit points that are of vegetation classification because they are not part of the subset.

Sfat: Filtered layers are indicated in the Layers Panel

In the *Layers* panel, a filtered layer is listed with a Filter icon next to it indicating the query used when the mouse hovers over the icon. Double-click the icon to open the *Query Builder* dialog for edit. This can also be achieved through the *Layer* ► *Filter…* menu.

20.2.3 Symbology Properties

In the *Symbology* tab the settings for the rendering of the point cloud are made. In the upper part, the settings of the different feature renderers can be found. In the lower part, there are sections with which general settings for the entire layer can be made and which apply over feature renderers.

Feature Rendering types

There are different options for rendering point clouds that can be selected using the drop-down menu at the top of the *Symbology* tab (see Fig. 20.4):

- *Extent Only*: Only a bounding box of the extent of the data is displayed; convenient for overviewing the data extent. As usual, the *Symbol widget* helps you configure any properties (color, stroke, opacity, sub-layers, …) you'd like for the box.
- *Attribute by Ramp*: The data is drawn over a color gradient. See *Attribute by Ramp Renderer*
- *RGB*: Draw the data using red, green and blue color values. See *RGB Renderer*
- *Classification*: The data is drawn using different colors for different classes. See *Classification Renderer*

When a point cloud is loaded, QGIS follows a logic to select the best renderer:

- if the dataset contains color information (red, green, blue attributes), the RGB renderer will be used
- else if the dataset contains a `Classification` attribute, the classified renderer will be used
- else it will fall back to rendering based on Z attribute

If you do not know the attributes of the point cloud, the *Statistics tab* provides a good overview of which attributes are contained in the point cloud and in which ranges the values are located.

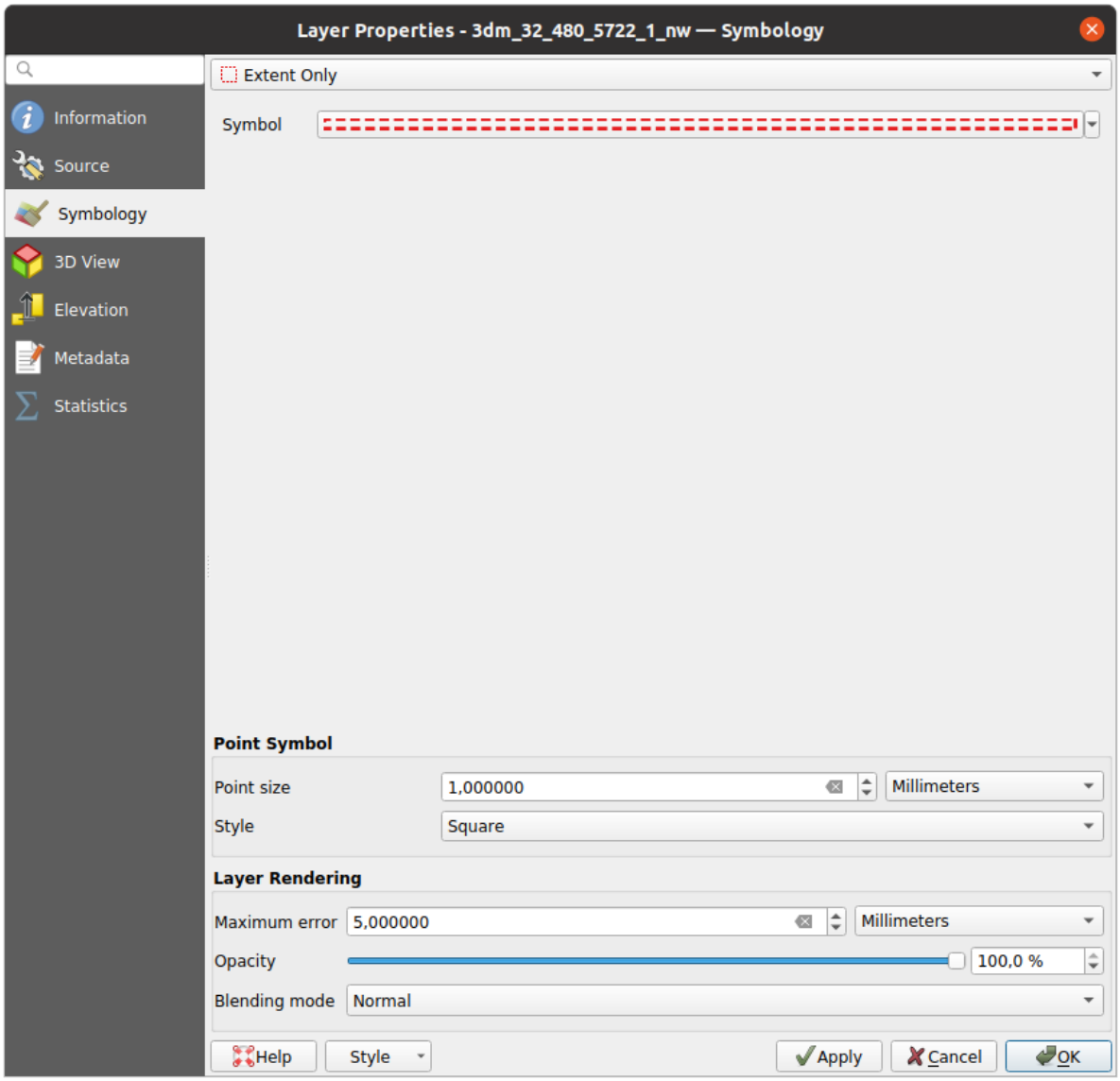


Fig. 20.4: Point cloud symbology tab

Attribute by Ramp Renderer

With *Attribute by Ramp*, the data can be displayed by numerical values over a color gradient. Such numerical values can be, for example, an existing intensity attribute or the Z-value. Depending on a minimum and a maximum value, the other values are spread to the color gradient via interpolation. The distinct values and their assignment to a certain color are called „color map" and are shown in the table. There are various setting options, which are described below the figure.

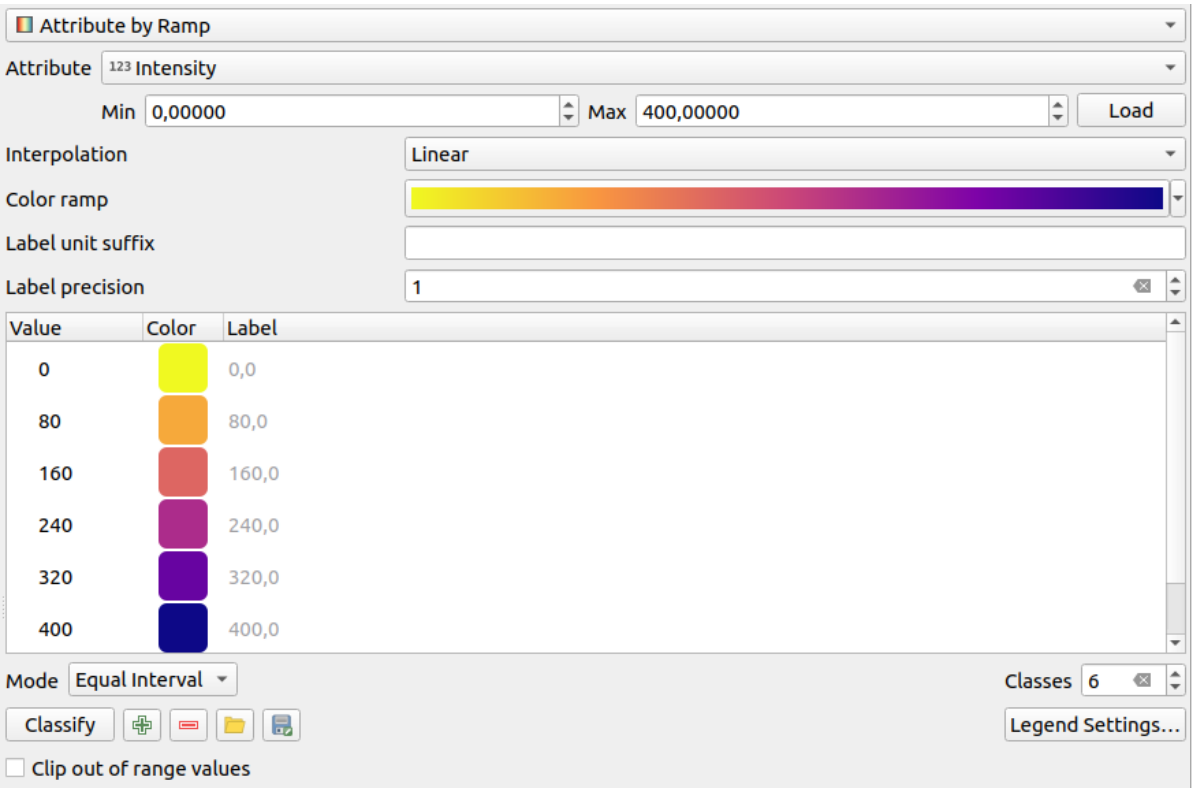


Fig. 20.5: Point cloud symbology tab: Attribute by Ramp

- *Min* and *Max* define the range that is applied to the color ramp: the *Min* value represents the left, the *Max* value the right end of the color ramp, the values in between are interpolated. By default QGIS detects the minimum and the maximum from the selected attribute but they can be modified. Once you have changed the values, you can restore the defaults by clicking on the *Load* button.
- The *Interpolation* entry defines how values are assigned their color:
  - *Discrete* (a <= symbol appears in the header of the *Value* column): The color is taken from the closest color map entry with equal or higher value
  - *Linear* The color is linearly interpolated from the color map entries above and below the pixel value, meaning that to each dataset value corresponds a unique color
  - *Exact* (a = symbol appears in the header of the *Value* column): Only pixels with value equal to a color map entry are applied a color; others are not rendered.
- The *Color ramp* widget helps you select the color ramp to assign to the dataset. As usual with *this widget*, you can create a new one and edit or save the currently selected one.
- The *Label unit suffix* adds a label after the value in the legend, and the *Label precision* controls the number of decimals to display.

The classification *Mode* helps you define how values are distributed across the classes:

- *Continuous*: Classes number and color are fetched from the color ramp stops; limits values are set following stops distribution in the color ramp (you can find more information on stops in *Setting a Color Ramp*).
- *Equal interval*: The number of classes is set by the *Classes* field at the end of the line; limits values are defined so that the classes all have the same magnitude.

The classes are determined automatically and shown in the color map table. But you can also edit these classes manually:

- Double clicking in a *Value* in the table lets you modify the class value

- Double clicking in the *Color* column opens the *Selectorul de Culoare* widget, where you can select a color to apply for that value
- Double clicking in the *Label* column to modify the label of the class
- Right-clicking over selected rows in the color table shows a contextual menu to *Change Color…* and *Change Opacity…* for the selection

Below the table there are the options to restore the default classes with *Classify* or to manually Add values or Delete selected values from the table.

Since a customized color map can be very complex, there is also the option to Load an existing color map or to Save it for use in other layers (as a `txt` file).

If you have selected *Linear* for *Interpolation*, you can also configure:

- *Clip out of range values* By default, the linear method assigns the first class (respectively the last class) color to values in the dataset that are lower than the set *Min* (respectively greater than the set *Max*) value. Check this setting if you do not want to render those values.
- *Legend settings*, for display in the *Layers* panel and in the *layout legend*. Customization works the same way as with a raster layer (find more details at *Customize raster legend*).

RGB Renderer

With the *RGB* renderer, three selected attributes from the point cloud will be used as the red, green and blue component. If the attributes are named accordingly, QGIS selects them automatically and fetches *Min* and *Max* values for each band and scales the coloring accordingly. However, it is also possible to modify the values manually.

A *Contrast enhancement* method can be applied to the values: *No Enhancement*, *Stretch to MinMax*, *Stretch and Clip to MinMax* and *Clip to MinMax*

Notă: The *Contrast enhancement* tool is still under development. If you have problems with it, you should use the default setting *Stretch to MinMax*.

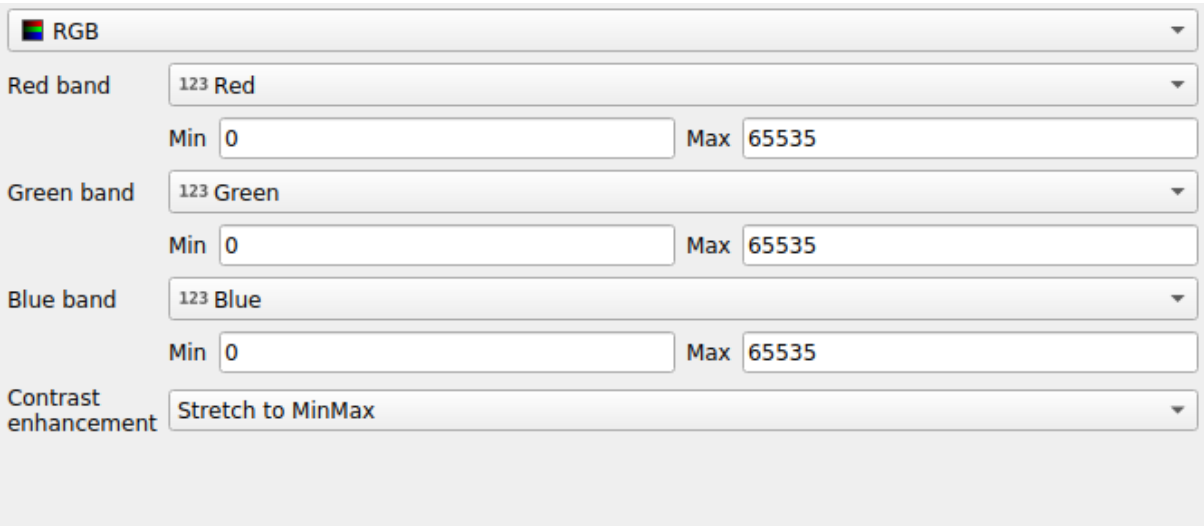


Fig. 20.6: The point cloud RGB renderer

Classification Renderer

In the *Classification* rendering, the point cloud is shown differentiated by color on the basis of an attribute. Any type of attribute can be used (numeric, string, …). Point cloud data often includes a field called `Classification`. This usually contains data determined automatically by post-processing, e.g. about vegetation. With *Attribute* you can select the field from the attribute table that will be used for the classification. By default, QGIS uses the definitions of the LAS specification (see table «ASPRS Standard Point Classes» in the PDF on ASPRS home page). However, the data may deviate from this schema; in case of doubt, you have to ask the person or institution from which you received the data for the definitions.

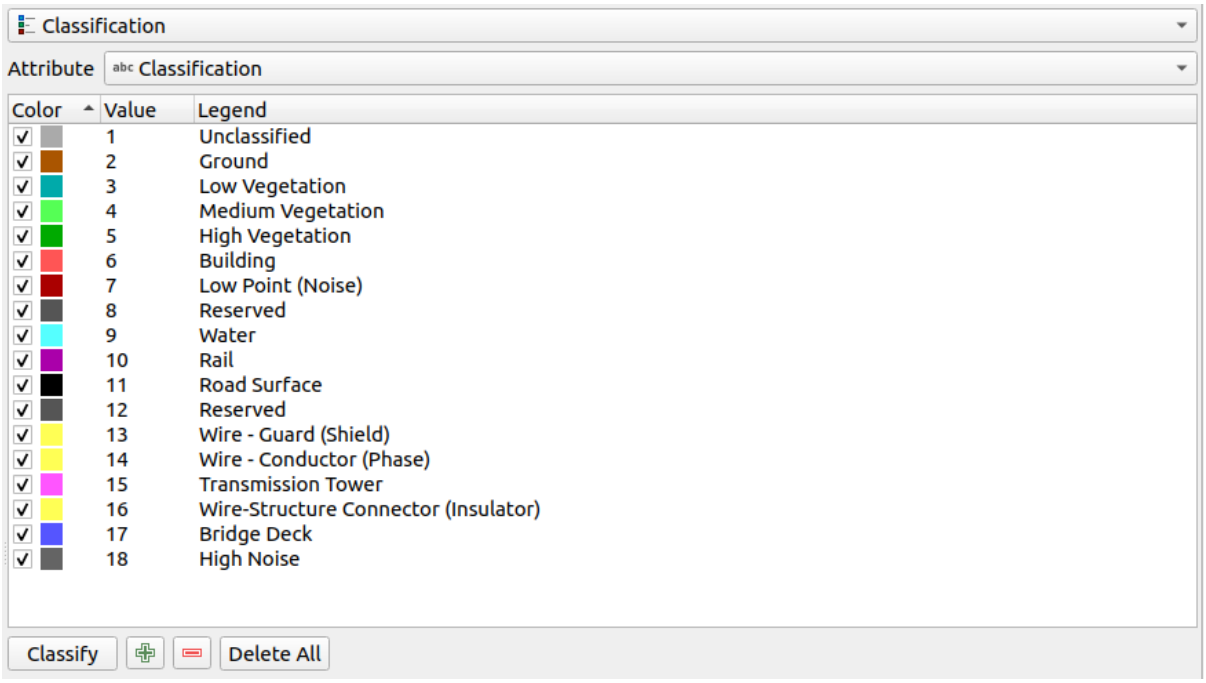


Fig. 20.7: The point cloud classification renderer

In the table all used values are displayed with the corresponding color and legend. At the beginning of each row there is a check box; if it is unchecked, this value is no longer shown on the map. With double click in the table, the *Color*, the *Value* and the *Legend* can be modified (for the color, the *Selectorul de Culoare* widget opens).

Below the table there are buttons with which you can change the default classes generated by QGIS:

- With the *Classify* button the data can be classified automatically: all values that occur in the attributes and are not yet present in the table are added
- With Add and Delete, values can be added or removed manually
- *Delete All* removes all values from the table

Sugestie: In the *Layers* panel, you can right-click over a class leaf entry of a layer to quickly configure visibility of the corresponding features.

Point Symbol

Under *Point Symbol*, the size and the unit (e.g. millimeters, pixels, inches) with which each data point is displayed can be set. Either *Circle* or *Square* can be selected as the style for the points.

Layer Rendering

In the *Layer Rendering* section you have the following options to modify the rendering of the layer:

- *Draw order*: allows to control whether point clouds rendering order on 2d map canvas should rely on their Z value. It is possible to render :
  - with the *Default* order in which the points are stored in the layer,
  - from *Bottom to top* (points with larger Z values cover lower points giving the looks of a true ortho photo),
  - or from *Top to bottom* where the scene appears as viewed from below.
- *Maximum error*: Point clouds usually contains more points than are needed for the display. By this option you set how dense or sparse the display of the point cloud will be (this can also be understood as «maximum allowed gap between points»). If you set a large number (e.g. 5 mm), there will be visible gaps between points. Low value (e.g. 0.1 mm) could force rendering of unnecessary amount of points, making rendering slower (different units can be selected).
- *Opacity*: You can make the underlying layer in the map canvas visible with this tool. Use the slider to adapt the visibility of your layer to your needs. You can also make a precise definition of the percentage of visibility in the menu beside the slider.
- *Blending mode*: You can achieve special rendering effects with this tool. The pixels of your overlaying and underlying layers are mixed through the settings described in *Modurile de Fuziune*.
- *Eye dome lighting*: this applies shading effects to the map canvas for a better depth rendering. Rendering quality depends on the *draw order* property; the *Default* draw order may give sub-optimal results. Following parameters can be controlled:
  - *Strength*: increases the contrast, allowing for better depth perception
  - *Distance*: represents the distance of the used pixels off the center pixel and has the effect of making edges thicker.

20.2.4 3D View Properties

In the *3D View* tab you can make the settings for the rendering of the point cloud in 3D maps.

3D Rendering modes

Following options can be selected from the drop down menu at the top of the tab:

- *No Rendering*: Data are not displayed
- *Follow 2D Symbology*: Syncs features rendering in 3D with *symbology assigned in 2D*
- *Single Color*: All points are displayed in the same *color* regardless of attributes
- *Attribute by Ramp*: Interpolates a given attribute over a color ramp and assigns to features their matching color. See *Attribute by Ramp Renderer*.
- *RGB*: Use different attributes of the features to set the Red, Green and Blue color components to assign to them. See *RGB Renderer*.
- *Classification*: differentiates points by color on the basis of an attribute. See *Classification Renderer*.

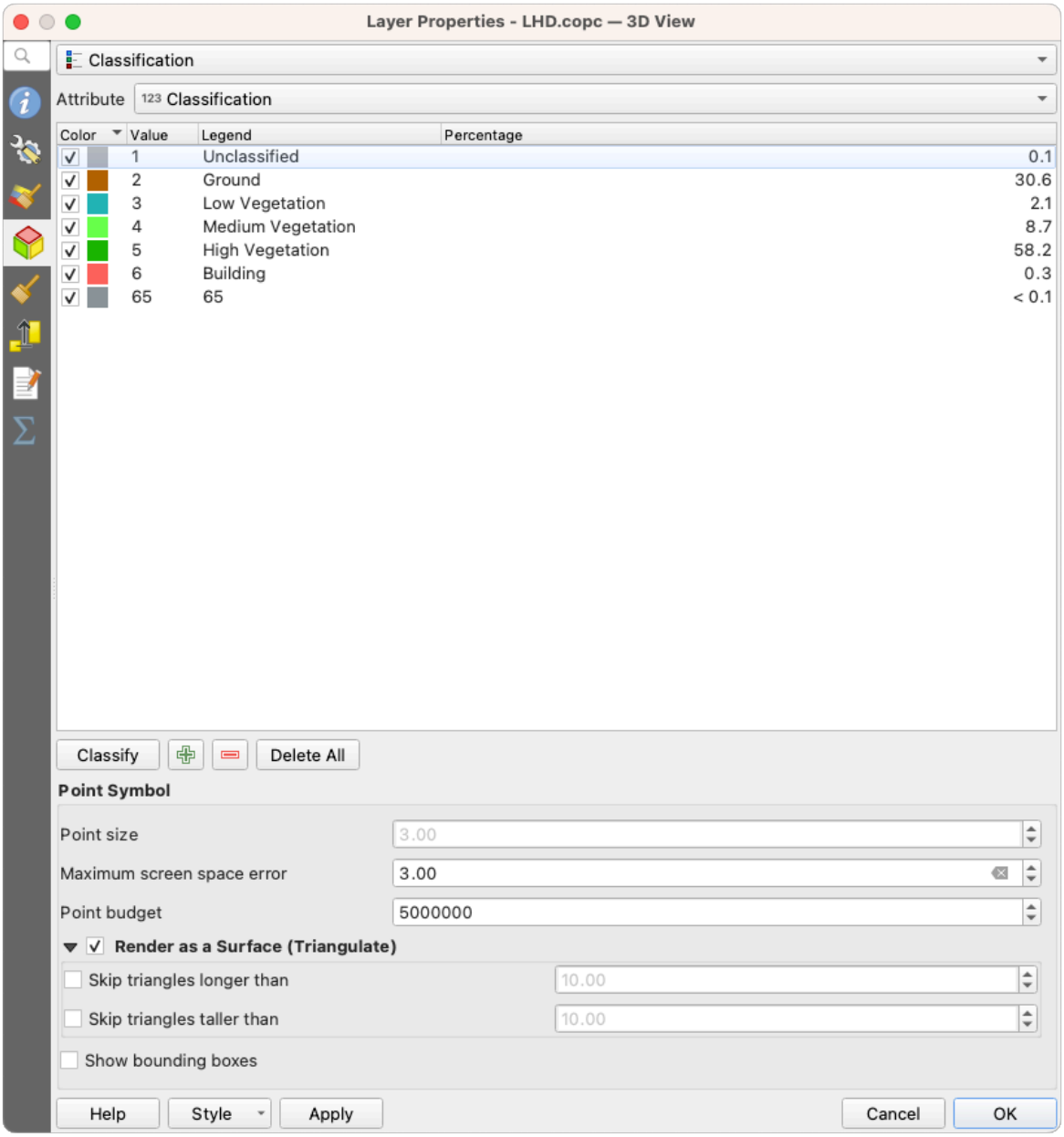


Fig. 20.8: The point cloud 3D view tab with the classification renderer

3D Point Symbol

In the lower part of the *3D View* tab you can find the *Point Symbol* section. Here you can make general settings for the entire layer which are the same for all renderers. There are the following options:

- *Point size*: The size (in pixels) with which each data point is displayed can be set
- *Maximum screen space error*: By this option you set how dense or sparse the display of the point cloud will be (in pixels). If you set a large number (e.g. 10), there will be visible gaps between points; low value (e.g. 0) could force rendering of unnecessary amount of points, making rendering slower (you can find more details at *Symbology Maximum error*).
- *Point budget*: To avoid long rendering, you can set the maximum number of points that will be rendered
- Check *Render as surface (Triangulate)* to render the point cloud layer in the 3D view with a solid surface obtained by triangulation. You can control dimensions of the computed triangles:

  - *Skip triangles longer than* a threshold value: sets in the horizontal plan, the maximum length of a side of the triangles to consider
  - *Skip triangles taller than* a threshold value: sets in the vertical plan, the maximum height of a side of the triangles to consider
- *Show bounding boxes*: Especially useful for debugging, shows bounding boxes of nodes in hierarchy

20.2.5 Rendering Properties

Under the *Scale dependent visibility* group box, you can set the *Maximum (inclusive)* and *Minimum (exclusive)* scale, defining a range of scale in which features will be visible. Out of this range, they are hidden. The Set to current canvas scale button helps you use the current map canvas scale as boundary of the range visibility. See *Visibility Scale Selector* for more information.

Notă: You can also activate scale dependent visibility on a layer from within the *Layers* panel: right-click on the layer and in the contextual menu, select *Set Layer Scale Visibility*.

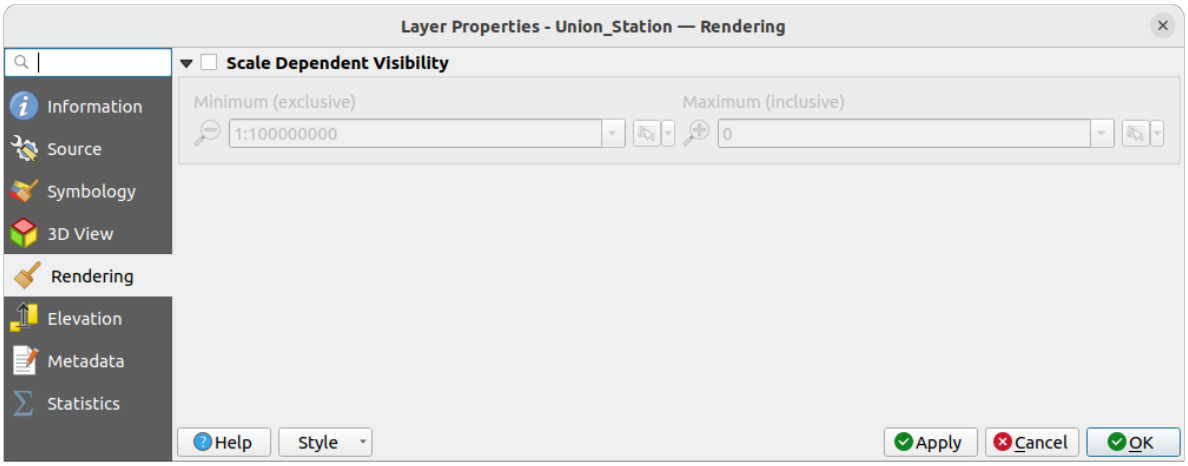


Fig. 20.9: The point cloud rendering tab

20.2.6 Elevation Properties

In the *Elevation* tab, you can set corrections for the Z-values of the data. This may be necessary to adjust the elevation of the data in 3D maps and its appearance in the *profile tool charts*. There are following options:

- Under *Elevation* group:
  - You can set a *Scale*: If `10` is entered here, a point that has a value Z = `5` is displayed at a height of `50`.
  - An *offset* to the z-level can be entered. This is useful to match different data sources in its height to each other. By default, the lowest z-value contained in the data is used as this value. This value can also be restored with the Refresh button at the end of the line.
- Under *Profile Chart Accuracy*, the *Maximum error* helps you control how dense or sparse the points will be rendered in the elevation profile. Larger values result in a faster generation with less points included.
- Under *Profile Chart Appearance*, you can control the point display:
  - *Point size*: the size to render the points with, in supported units (millimeters, map units, pixels, …)
  - *Style*: whether to render the points as *Circle* or *Square*

- Apply a single *Color* to all the points visible in the profile view
- Check *Respect layer's coloring* to instead show the points with the color assigned via their *2D symbology*
- *Apply opacity by distance from curve effect*, reducing the opacity of points which are further from the profile curve

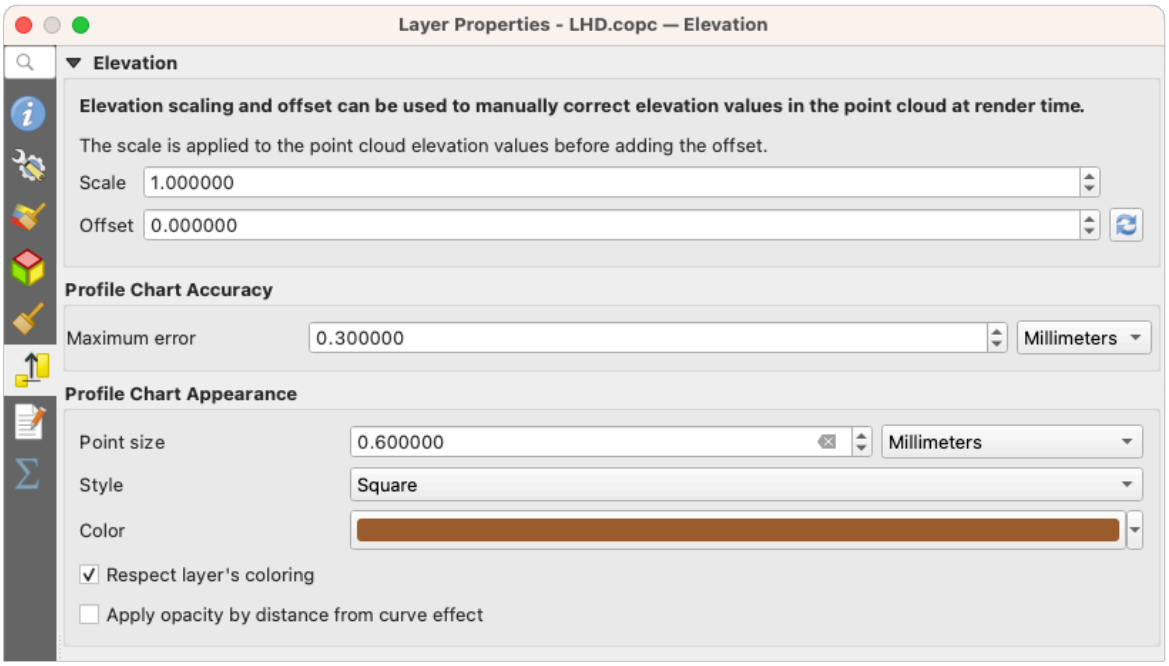


Fig. 20.10: The point cloud elevation tab

20.2.7 Proprietăți Metadate

The *Metadata* tab provides you with options to create and edit a metadata report on your layer. See *Metadata* for more information.

20.2.8 Statistics Properties

In the *Statistics* tab you can get an overview of the attributes of your point cloud and their distribution.

At the top you will find the section *Attribute Statistics*. Here all attributes contained in the point cloud are listed, as well as some of their statistical values: *Minimum*, *Maximum*, *Mean*, *Standard Deviation*

If there is an attribute *Classification*, then there is another table in the lower section. Here all values contained in the attribute are listed, as well as their absolute *Count* and relative % abundance.

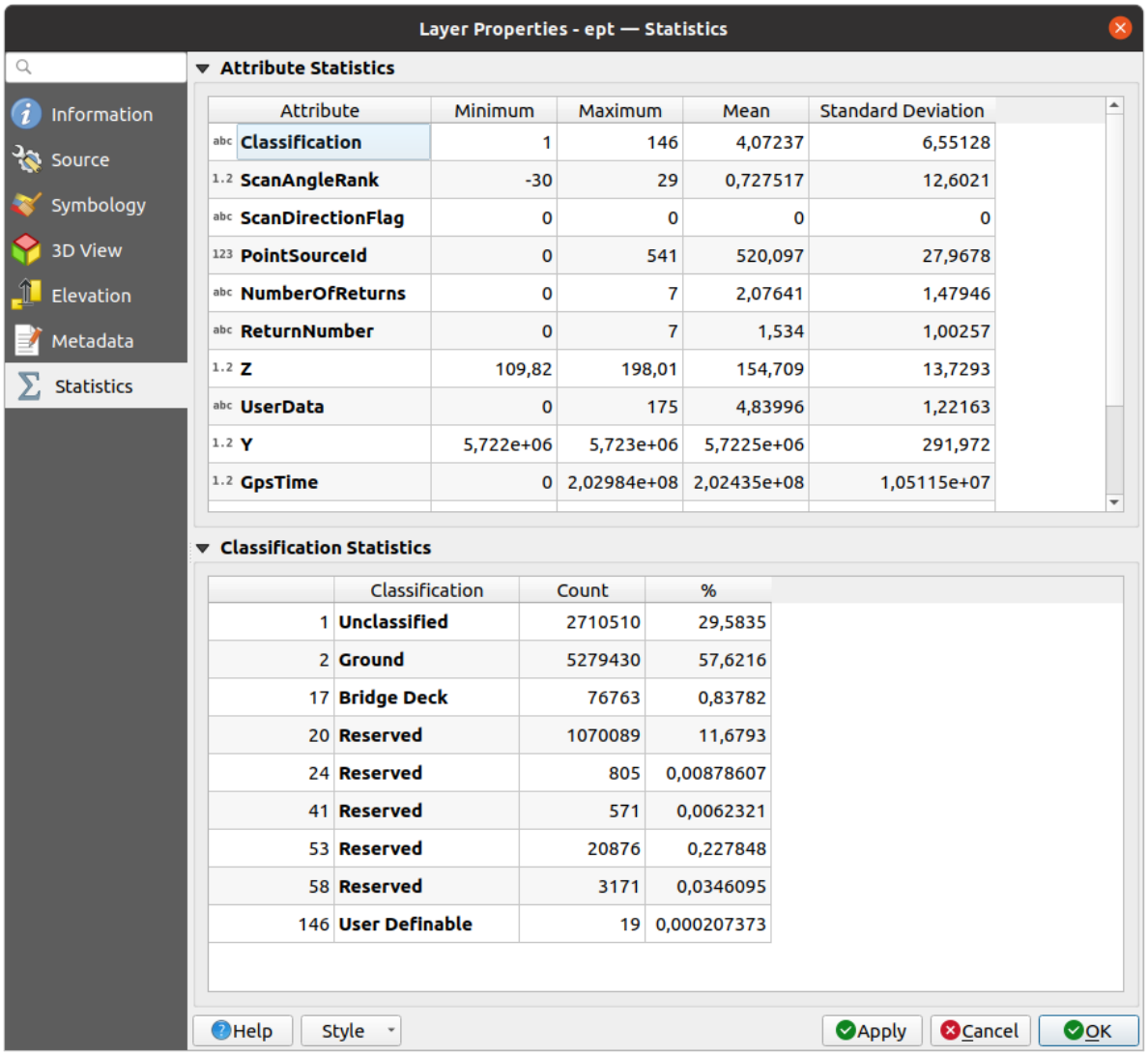


Fig. 20.11: The point cloud statistics tab

20.3 Virtual point cloud

Lidar surveys of larger areas are often multi-terabyte datasets with many billions of points. Having such large datasets represented as a single point cloud file is not practical due to the difficulties of storage, transfer, display and analysis. Point cloud data are therefore typically stored and distributed split into square tiles (e.g. `1km x 1km`), each tile having a more manageable file size (e.g. ~200 MB when compressed).

Tiling of data solves the problems with size of data, but it introduces issues when processing or viewing an area of interest that does not fit entirely into a single tile. Users need to develop workflows that take into account multiple tiles and special care needs to be taken to deal with data near edges of tiles to avoid unwanted artefacts in outputs. Similarly, when viewing point cloud data, it becomes cumbersome to load many individual files and apply the same symbology.

Here is an example of several point cloud tiles loaded in QGIS. Each tile is styled based on min/max Z values of the tile, creating visible artefacts on tile edges. The styling has to be adjusted for each layer separately:

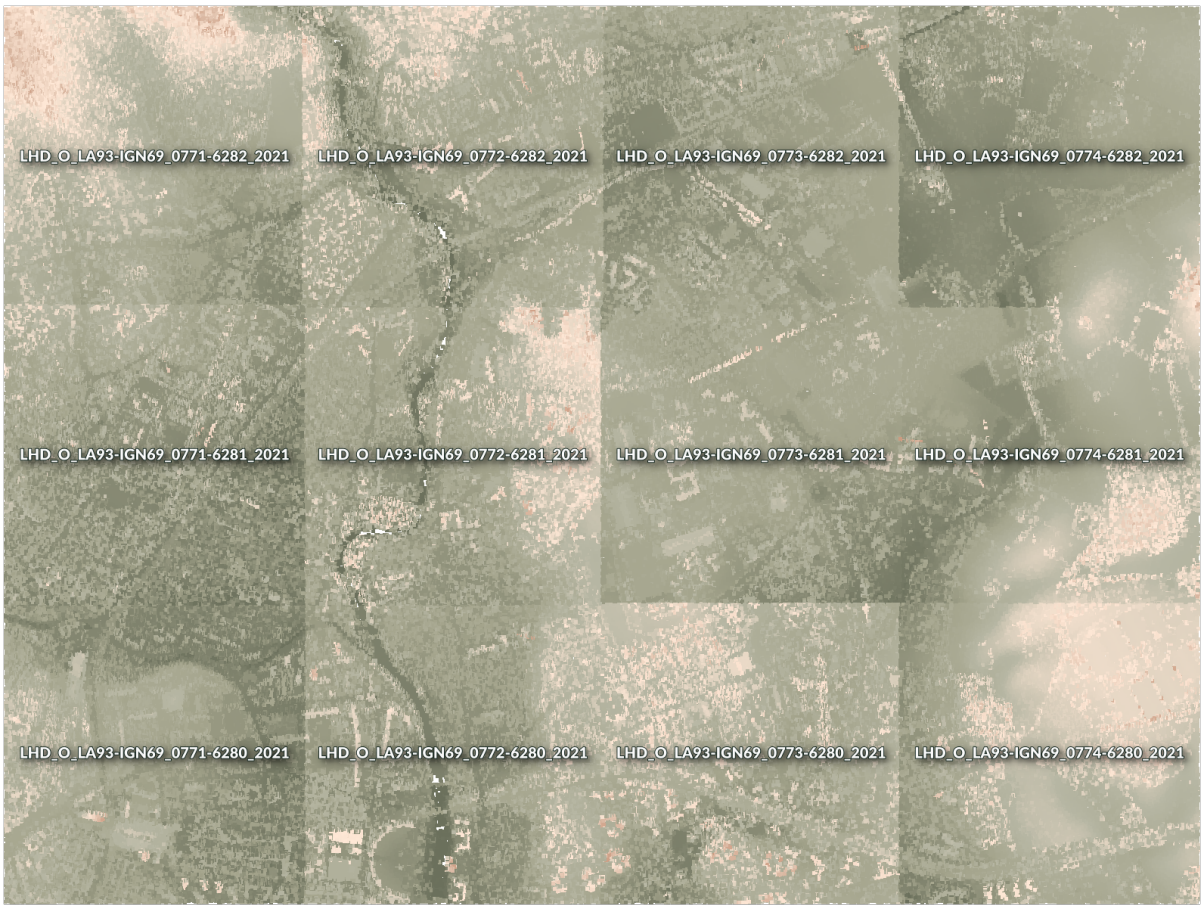


Fig. 20.12: Individual point cloud tiles loaded, with artefacts on edges

In the GIS world, many users are familiar with the concept of virtual rasters. A virtual raster is a file that simply references other raster files with actual data. In this way, GIS software then treats the whole dataset comprising many files as a single raster layer, making the display and analysis of all the rasters listed in the virtual file much easier.

Borrowing the concept of virtual rasters from GDAL, **virtual point cloud (VPC)** is a file format that references other point cloud files. Software supporting virtual point clouds handles the whole tiled dataset as a single data source.

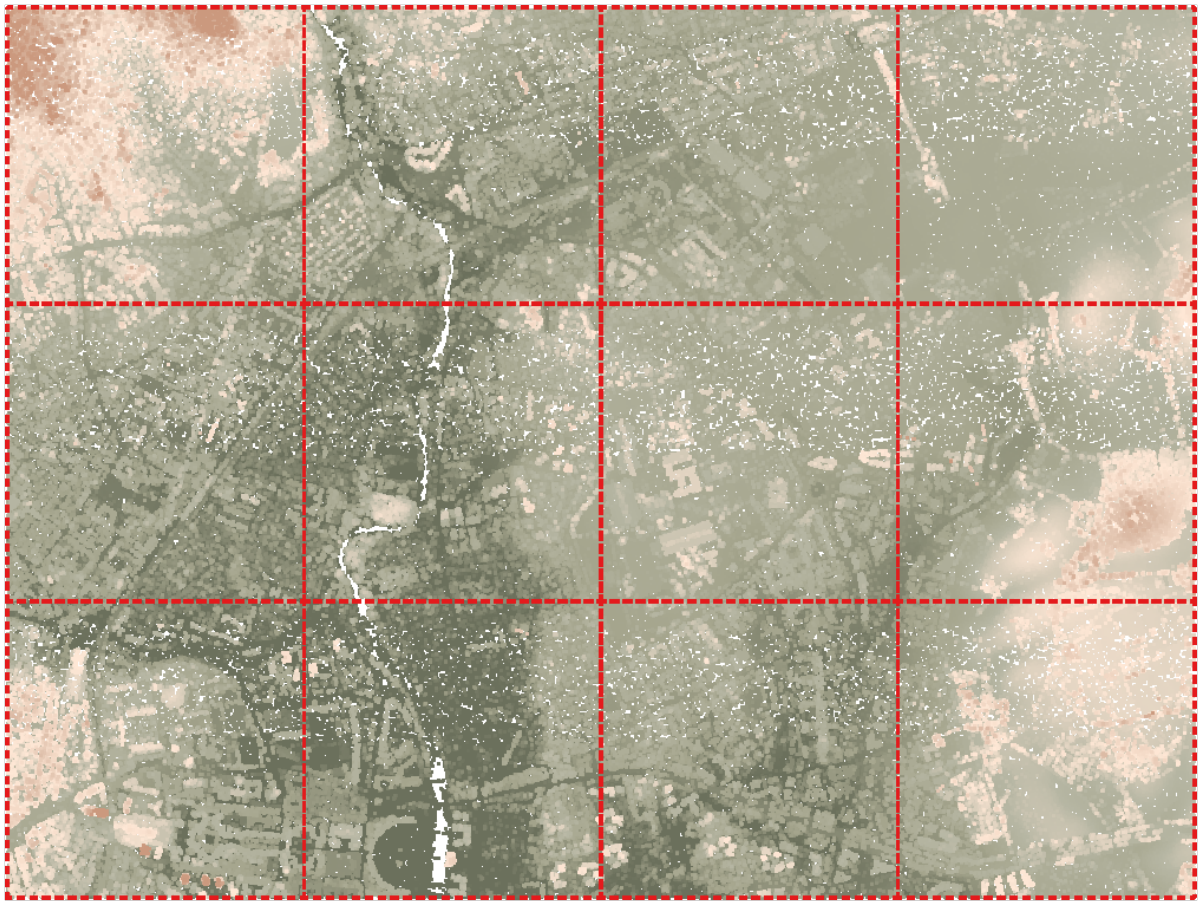


Fig. 20.13: The virtual point cloud

Displaying and manipulating virtual point cloud is much more fluent and easy.

At the core, a virtual point cloud file is a simple JSON file with `.vpc` extension, containing references to actual data files (e.g. `.LAS`, `.LAZ` or `.COPC` files) and additional metadata extracted from the files. Even though it is possible to write VPC files by hand, it is strongly recommended to create them using an automated tool, such as:

- The Processing *Build virtual point cloud (VPC)* algorithm
- The `build_vpc` command of PDAL wrench tool

For more details, please refer to the VPC specification that also contains best practices and optional extensions (such as overviews).

CAPITOLUL 21

Working with 3D Tiles

21.1 What are 3D Tiles?

3D tiles are specification for streaming and rendering large-scale 3D geospatial datasets. They use a hierarchical structure to efficiently manage and display 3D content, optimising performance by dynamically loading appropriate levels of detail. This technology is widely used in urban planning, architecture, simulation, gaming, and virtual reality, providing a standardised and interoperable solution for visualising complex geographical data. Read *Using 3D tiles services* for more details.

Fig. 21.1: Example of 3D tiles

21.2 Data Sources

Support for 3D tiles:

- Remote source - `http://example.com/tileset.json`
- Local files - `file:///path/to/tiles/tileset.json`

21.2.1 Using data from Cesium ion

Cesium ion is a cloud-based platform for managing and streaming 3D geospatial data. It simplifies data management, visualisation and sharing. To add 3D tiles from Cesium ion, you need to first sign up to their service here: https://ion.cesium.com.

You can also use Cesium ion plugin from the QGIS repository to add the data to QGIS.

21.2.2 Using Google 3D data

To use Google Photorealistic 3D tiles in QGIS, get an API key for 3D tiles. Once you have the key, simply add Google tiles using the same connection process.

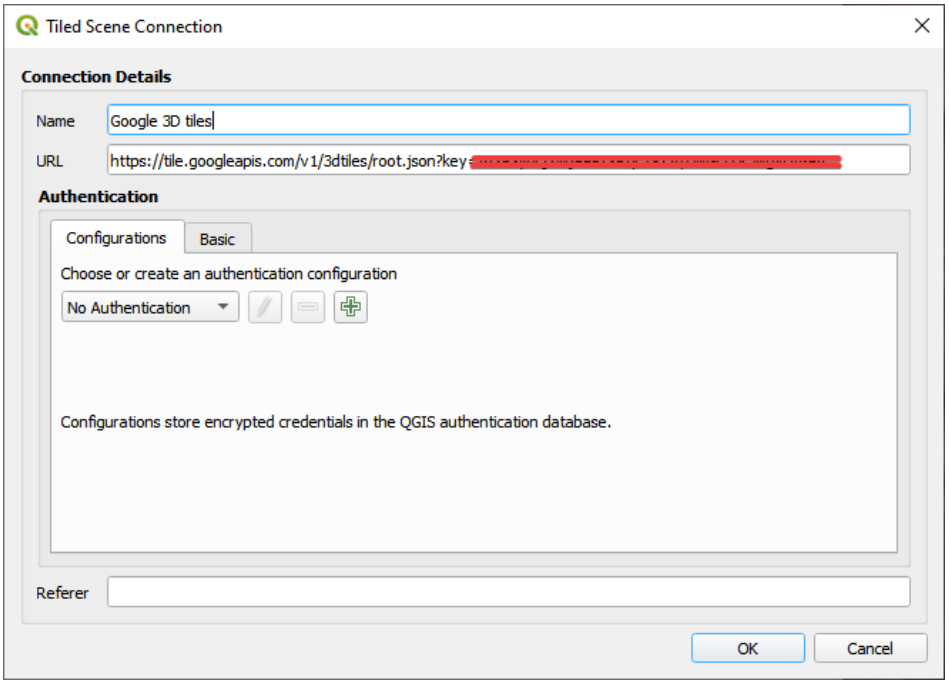
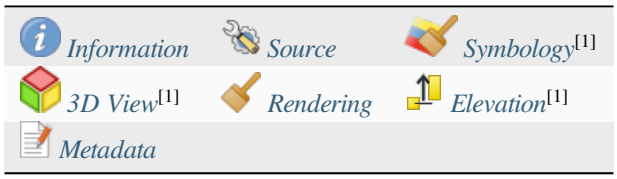


Fig. 21.2: Google 3D Tiles connection

21.3 3D Tiles Properties

The 3D tiles *Layer Properties* dialog provides the following sections:



[1] Also available in the *Layer styling panel*

21.3.1 Information Properties

The *Information* tab is read-only and represents an interesting place to quickly grab summarized information and metadata on the current layer. Provided information are:

- based on the provider of the layer: name, URL, source type and path, number of zoom levels
- the Coordinate Reference System: name, units, method, accuracy, reference (i.e. whether it's static or dynamic)
- picked from the *filled metadata*: access, extents, links, contacts, history…

21.3.2 Source Properties

The *Source* tab displays basic information about the selected 3D tile, including:

- the *Layer name* to display in the *Layers Panel*;
- the *Coordinate Reference System*: Displays the layer's *Coordinate Reference System (CRS)*. You can change the layer's CRS, by selecting a recently used one in the drop-down list or clicking on the Select CRS button (see *Coordinate Reference System Selector*). Use this process only if the layer CRS is wrong or not specified.

21.3.3 Symbology Properties

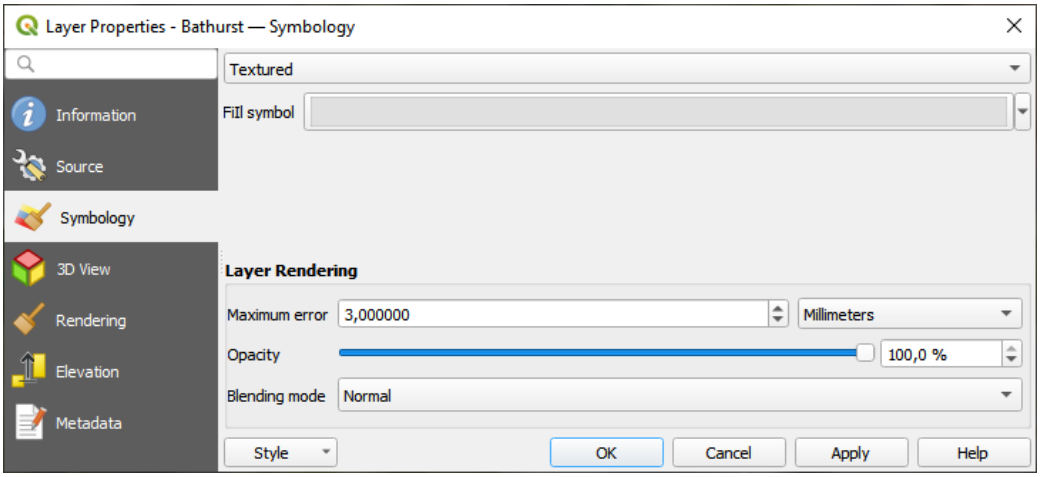


Fig. 21.3: 3D Tile Layer Symbology

By default, the layer is styled using texture, but you can change it to see the wireframe mesh behind the scene by choosing *Wireframe* in the drop-down menu. You can also, change the mesh fill and line symbols similar to the vector polygons. Checking *Use texture colors* will render each mesh element with the average value of the full texture. This is a good option to try when dealing with a large dataset and want to get a quick overview of the data.

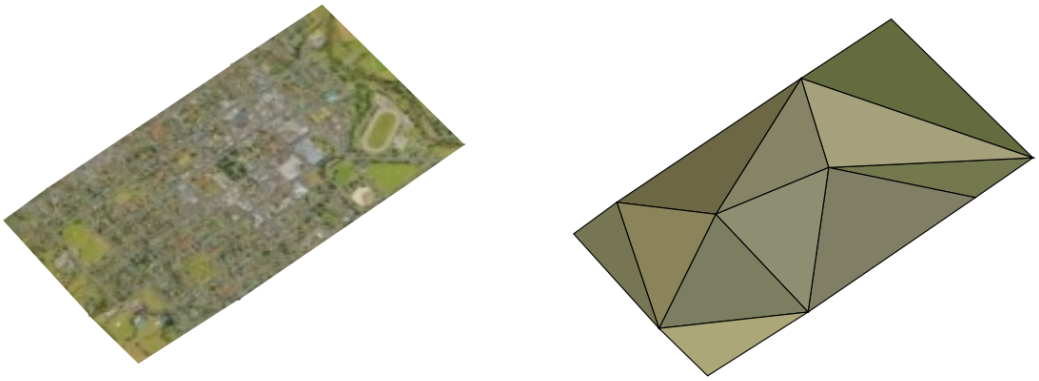


Fig. 21.4: 3D Tiles - textured and wireframe

To view the data you can open *New 3D map view*.

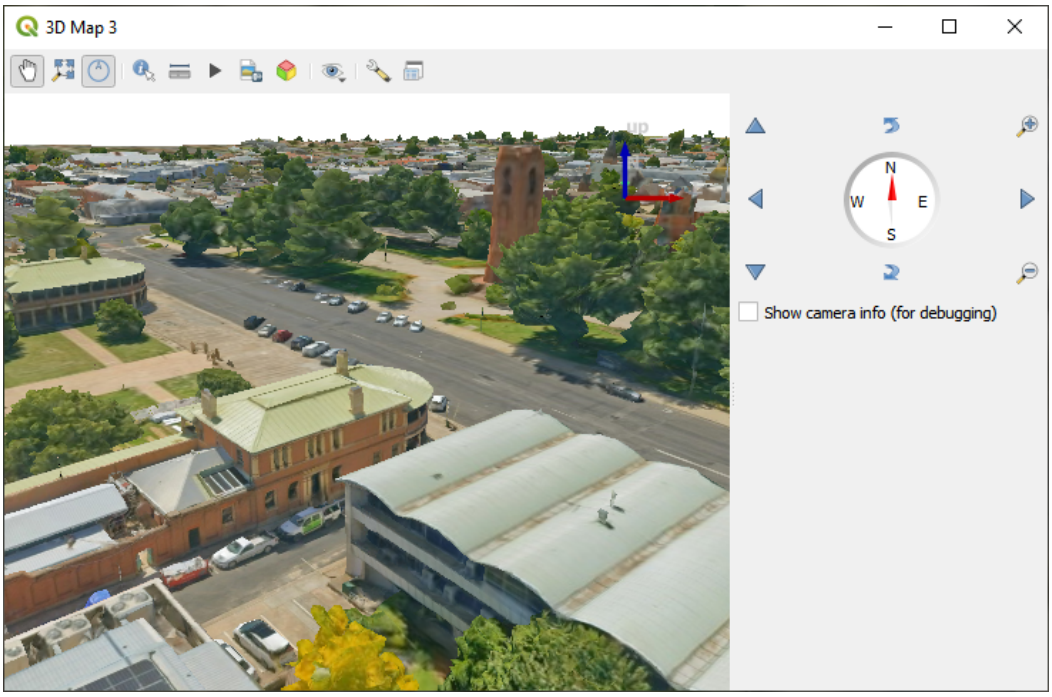


Fig. 21.5: 3D Map View

From the *Symbology* tab, you can also set some options that invariably act on all features of the layer:

- *Maximum error*: This parameter determines the level of detail displayed in the 3D model. Similar to point clouds, 3D tiles often contain more information than necessary for visual representation. By adjusting this setting, you control the balance between display density and rendering speed. A larger value (e.g., 5 mm) may introduce noticeable gaps between elements, while a smaller value (e.g., 0.1 mm) could lead to the rendering of an excessive number of details, potentially slowing down the rendering process. Different units can be selected to tailor the setting to your specific needs.
- *Opacity*: Adjust the visibility of the underlying layer on the map canvas using this tool. Use slider to tailor the visibility of your scene layer according to your preferences. Alternatively, specify the exact percentage of visibility through the menu next to the slider.
- *Blending mode*: You can achieve special rendering effects with these tools that you may previously only know from graphics programs. The pixels of your overlaying and underlaying layers are mixed through the settings described in *Modurile de Fuziune*.

21.3.4 3D View Properties

- *Maximum screen space error* determines the threshold for swapping terrain tiles with more detailed ones (and vice versa) - i.e. how soon the 3D view will use higher quality tiles. Lower numbers mean more details in the scene at the expenses of increased rendering complexity.
- *Show bounding boxes* Show 3D bounding boxes of the terrain tiles (useful for troubleshooting terrain issues).

21.3.5 Rendering Properties

Under *Scale dependent visibility*, you can set the *Maximum (inclusive)* and *Minimum (exclusive)* scales, defining a range of scales in which features will be visible. Out of this range, they are hidden. The Set to current canvas scale button helps you use the current map canvas scale as boundary of the range visibility. See *Visibility Scale Selector* for more information.

21.3.6 Elevation Properties

The *Elevation* tab provides options to control the layer elevation properties within a *3D map view*. Specifically, you can set:

- *Elevation Surface*: how the 3D layer vertices Z values should be interpreted as terrain elevation. You can apply a *Scale* factor and an *Offset*.

21.3.7 Metadata Properties

The *Metadata* tab provides you with options to create and edit a metadata report on your layer. See *Metadata* for more information.

CAPITOLUL 22

Așezarea în pagină a hărților

Folosind Compozițiile și Rapoartele Imprimabile puteți crea hărți și atlase pe care le puteți imprima sau salva în fișiere de tip imagine, PDF sau SVG.

22.1 Generalități despre Compozitorul de Hărți

The print layout provides growing layout and printing capabilities. It allows you to add elements such as the QGIS 2D or 3D map canvas, text labels, images, legends, scale bars, basic shapes, arrows, attribute and simple tables, elevation profiles and HTML frames… You can size, group, align, position and rotate each element and adjust their properties to create your layout. The layout can be printed or exported to image formats, PostScript, PDF or to SVG. You can save the layout as a template and load it again in another session. Finally, generating several maps based on a template can be done through the atlas generator.

22.1.1 Sample Session for beginners

Before you start to work with the print layout, you need to load some raster or vector layers in the QGIS map canvas and adapt their properties to suit your own convenience. After everything is rendered and symbolized to your liking, click the New Print Layout icon in the *Project* toolbar or choose *Project* ► *New Print Layout*. You will be prompted to choose a title for the new layout.

Pentru a demonstra cum se poate crea o hartă folosiți următoarele instrucțiuni.

1. On the left side, select the Add map toolbar button and draw a rectangle on the canvas holding down the left mouse button. Inside the drawn rectangle the QGIS map view to the canvas.

2. Select the Add scalebar toolbar button and click with the left mouse button on the print layout canvas. A scalebar will be added to the canvas.

3. Select the Add legend toolbar button and draw a rectangle on the canvas holding down the left mouse button. Inside the drawn rectangle the legend will be drawn.

4. Selectați pictograma Select/Move item pentru a selecta harta de pe canevas și pentru a o muta un pic.

5. While the map item is still selected you can also change the size of the map item. Click while holding down the left mouse button, in a white little rectangle in one of the corners of the map item and drag it to a new location to change its size.
6. Click the *Item Properties* panel on the left down side and find the setting for the orientation. Change the value of the setting *Map orientation* to «15.00°». You should see the orientation of the map item change.
7. Now, you can print or export your print layout to image formats, PDF or to SVG with the export tools in *Layout* menu.
8. Finally, you can save your print layout within the project file with the Save Project button.

You can add multiple elements to the print layout. It is also possible to have more than one map view or legend or scale bar in the print layout canvas, on one or several pages. Each element has its own properties and, in the case of the map, its own extent. If you want to remove any elements from the layout canvas, you can do that with the `Delete` or the `Backspace` key.

22.1.2 Managerul de Compoziții

The *Layout Manager* is the main window to manage print layouts in the project. It gives you an overview of existing print layouts and reports in the project and offers tools to:

- search for a layout;
- add new print layout or new report from scratch, template or duplicating an existing one;
- rename or delete any of them;
- open them in the project.

To open the layout manager dialog:

- from the main QGIS dialog, select *Project* ► *Layout Manager…* menu or click on the Layout Manager button in the *Project Toolbar*;
- from a print layout or report dialog, select *Layout* ► *Layout Manager…* menu or click on the Layout Manager button in the *Layout Toolbar*.

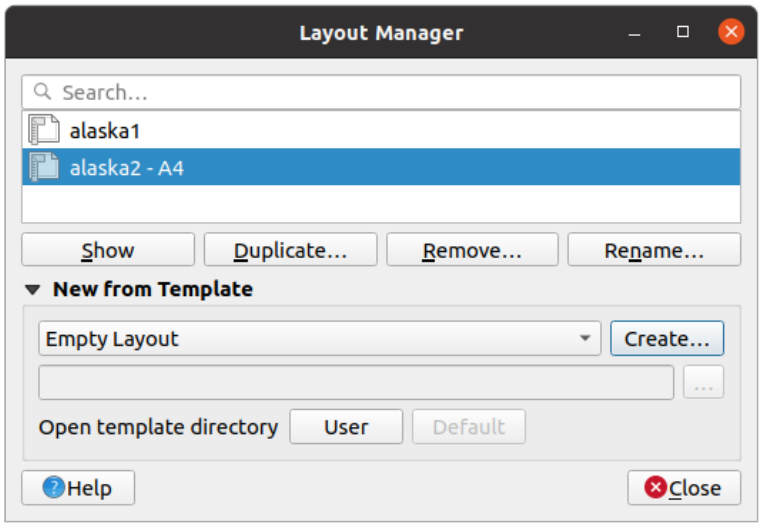


Fig. 22.1: The Print Layout Manager

The layout manager lists in its upper part all the available print layouts or reports in the project with tools to:

- show the selection: you can select multiple reports and/or print layout(s) and open them in one-click. Double-click a name also opens it;

- duplicate the selected print layout or report (available only if one item is selected): it creates a new dialog using the selected one as template. You'll be prompted to choose a new title for the new layout;
- rename the report or layout (available only if one item is selected): you'll be prompted to choose a new title for the layout;
- remove the layout: the selected print layout(s) will be deleted from the project.

In the lower part, it's possible to create new print layouts or reports from scratch or a template. By default, QGIS will look for templates in the *user profile* folder and the application template directories (accessible with the two buttons at the bottom of the frame) but also in any folder declared as *Path(s) to search for extra print templates* in *Settings* ► *Options* ► *Layouts*. Found templates are listed in the combobox. Select an item and press the *Create* button to generate a new report or print layout.

You can also use layout templates from a custom folder; in that case, select *specific* in the templates drop-down list, browse to the template and press *Create*.

Sfat: Creating template-based print layouts from Browser panel

Drag-and-drop a print layout template `.qpt` file from any file browser onto the map canvas or double-click it in the *Browser panel* generates a new print layout from the template.

22.1.3 Menus, tools and panels of the print layout

Opening the print layout provides you with a blank canvas that represents the paper surface when using the print option. Initially you find buttons on the left beside the canvas to add print layout items: 2D or 3D map canvases, text labels, images, legends, scale bars, basic shapes, arrows, attribute or simple tables, HTML frames, elevation profiles,… In this toolbar you also find buttons to navigate, zoom in on an area and pan the view on the layout as well as buttons to select any layout item and to move the extents of the map items.

Fig. 22.2 shows the initial view of the print layout before any elements are added.

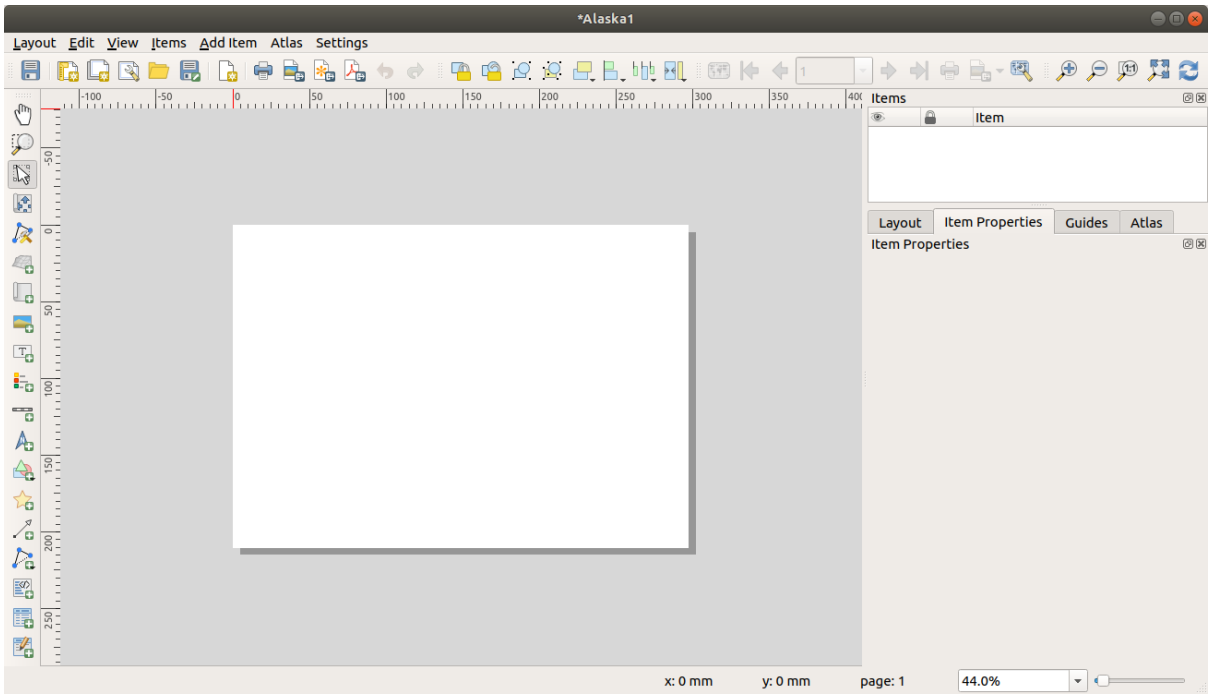


Fig. 22.2: Aspect Pagină Imprimată

On the right beside the canvas you find two set of panels. The upper one holds the panels *Items* and *Undo History* and the lower holds the panels *Layout*, *Item properties* and *Atlas generation*.

- The *Items* panel provides a list of all the print layout items added to the canvas and ways to globally interact with them (see *Panoul Elementelor* for more information).
- The *Undo History* panel displays the history of all changes applied to the layout. With a mouse click, it is possible to undo and redo layout steps back and forth to a certain status.
- The *Layout* panel allows you to set general parameters to apply to the layout when exporting or working within (see *The Layout Panel* for more details);
- The *Item Properties* panel displays the properties for the selected item. Click the Select/Move item icon to select an item (e.g., legend, scale bar or label) on the canvas. Then click the *Item Properties* panel and customize the settings for the selected item (see *Elementele Compoziției* for detailed information on each item settings).
- The *Atlas* panel allows you to enable the generation of an atlas for the current layout and gives access to its parameters (see *Generarea unui Atlas* for detailed information on atlas generation usage).

In the bottom part of the print layout window, you can find a status bar with mouse position, current page number, a combo box to set the zoom level, the number of selected items if applicable and, in the case of atlas generation, the number of features.

In the upper part of the print layout window, you can find menus and other toolbars. All print layout tools are available in menus and as icons in a toolbar.

The toolbars and the panels can be switched off and on using the right mouse button over any toolbar, or through *View* ► *Toolbars* ► or *View* ► *Panels* ►.

Menus and Tools

Layout menu

The *Layout* provides action to manage the layout:

- Save the project file directly from the print layout window.
- Create a new and blank print layout with *New Layout…*.
- *Duplicate Layout…* : Create a new print layout by duplicating the current one.
- Remove the current layout with *Delete Layout…*.
- Open the *Layout Manager…*.
- *Layouts* ► : Open an existing print layout.

Once the layout is designed, with *Save as Template* and *Add Items from Template* icons, you can save the current state of a print layout session as a `.qpt` template file and load its items again in another session/print layout.

In the *Layout* menu, there are also powerful ways to share geographical information produced with QGIS that can be included in reports or published. These tools are *Export as Image…*, *Export as PDF…*, *Export as SVG…* and *Print…*.

Below is a list of all the available tools in this menu with some convenient information.

Instrument	Scurtătură	Bara de instrumente	Referință
Save Project	`Ctrl+S`	*Layout*	*Prezentarea proiectelor QGIS*
New Layout	`Ctrl+N`	*Layout*	*Managerul de Compoziții*
Duplicate Layout		*Layout*	*Managerul de Compoziții*
Delete Layout			
Managerul de Compoziții…		*Layout*	*Managerul de Compoziții*
Compoziții ►			
Layout Properties…			*The Layout Panel*
Rename Layout…			
Adăugare Pagini…		*Layout*	*Working with the page properties*
Add Items from Template		*Layout*	*Creating a layout item*
Save as Template…		*Layout*	*Managerul de Compoziții*
Export as Image…		*Layout*	*Exportare ca imagine*
Export as SVG…		*Layout*	*Exportare ca SVG*
Export as PDF…		*Layout*	*Exportare ca PDF*
Setare Pagină…	`Ctrl+Shift+P`		
Print…	`Ctrl+P`	*Layout*	*Generarea unui Rezultat*
Închidere	`Ctrl+Q`		

Edit menu

The *Edit* menu offers tools to manipulate print layout items. It includes common actions like selection tools, Copy/Cut/Paste and undo/redo (see *The Undo History Panel: Revert and Restore actions*) functionality for the items in the layout.

When using the Paste action, the elements will be pasted according to the current mouse position. Using the *Edit* ► *Paste in Place* action or pressing `Ctrl+Shift+V` will paste the items into the current page, at the same position they were in their initial page. It ensures to copy/paste items at the same place, from page to page.

Below is a list of all the available tools in this menu with some convenient information.

Tabelul 22.1: Available Tools

Instrument	Scurtătură	Bara de instrument	Referință
Undo (last change)	`Ctrl+Z`	*Layout*	*The Undo History Panel: Revert and Restore actions*
Redo (last reverted change)	`Ctrl+Y`	*Layout*	*The Undo History Panel: Revert and Restore actions*
Delete	`Del`		
Cut	`Ctrl+X`		
Copy	`Ctrl+C`		
Lipire	`Ctrl+V`		
Paste in place	`Ctrl+Shift+V`		
Select All	`Ctrl+A`		
Deselect all	`Ctrl+Shift+A`		
Invert Selection			
Select Next Item Below	`Ctrl+Alt+[`		
Select Next Item above	`Ctrl+Alt+]`		
Pan Layout	`P`	*Instrumenta*	
Zoom	`Z`	*Instrumenta*	
Select/Move Item	`V`	*Instrumenta*	*Interacting with layout items*
Move Content	`C`	*Instrumenta*	*Elementul Hartă*
Edit Nodes Item		*Instrumenta*	*Formele Articulate*

View menu

The *View* menu gives access to navigation tools and helps to configure general behavior of the print layout. Beside the common zoom tools, you have means to:

- Refresh view (if you find the view in an inconsistent state);
- enable a *grid* you could snap items to when moving or creating them. Grids setting is done in *Settings* ► *Layout Options…* or in the *Layout Panel*;
- enable *guides* you could snap items to when moving or creating them. Guides are red lines that you can create by clicking in the ruler (above or at the left side of the layout) and drag and drop to the desired location;
- *Smart Guides*: uses other layout items as guides to dynamically snap to as you move or reshape an item;
- *Clear Guides* to remove all current guides;
- *Show Bounding box* around the items to better identify your selection;
- *Show Rules* around the layout;
- *Show Pages* or set up pages to transparent. Often layout is used to create non-print layouts, e.g. for inclusion in presentations or other documents, and it's desirable to export the composition using a totally transparent background. It's sometimes referred to as „infinite canvas" in other editing packages.

In the print layout, you can change the zoom level using the mouse wheel or the slider and combo box in the status bar. If you need to switch to pan mode while working in the layout area, you can hold the `Spacebar` or the mouse

wheel. With `Ctrl+Spacebar`, you can temporarily switch to Zoom In mode, and with `Ctrl+Alt+Spacebar`, to Zoom Out mode.

Panels and toolbars can be enabled from the *View* ► menu. To maximise the space available to interact with a composition you can check the *View* ► *Toggle Panel Visibility* option or press `Ctrl+Tab`; all panels are hidden and only previously visible panels are restored when unchecked.

It's also possible to switch to a full screen mode to have more space to interact with by pressing `F11` or using *View* ► *Toggle Full Screen*.

Instrument	Scurtătură	Bara de instrumente	Referință
Actualizare	`F5`	*Navigation*	
Previzualizare ►			
Mărire	`Ctrl++`	*Navigation*	
Micșorare	`Ctrl+-`	*Navigation*	
Zoom la 100%	`Ctrl+1`	*Navigation*	
Transfocare la nivelul Hărții	`Ctrl+0`	*Navigation*	
Zoom to Width			
Show Grid	`Ctrl+'`		*Guides and Grid*
Snap to Grid	`Ctrl+Shift+'`		*Guides and Grid*
Show Guides	`Ctrl+;`		*Guides and Grid*
Snap to Guides	`Ctrl+Shift+;`		*Guides and Grid*
Smart Guides	`Ctrl+Alt+;`		
Manage Guides…			*Panoul Ghidajelor*
Ștergere Ghidaje			*Panoul Ghidajelor*
Show Rulers	`Ctrl+R`		
Show Bounding Boxes	`Ctrl+Shift+B`		
Afișare Pagini			
Bare de Instrumente ►			*Panouri și Bare de Instrumente*
Panouri ►			*Panouri și Bare de Instrumente*
Toggle Full Screen	`F11`		*Vizualizare*
Toggle Panel Visibility	`Ctrl+Tab`		*Vizualizare*

Items menu

The *Items* helps you configure items» position in the layout and the relations between them (see *Interacting with layout items*).

Instrument	Scurtătură	Bara de instrumente	Referință
Grup	`Ctrl+G`	*Actions*	*Grouping items*
Ungroup	`Ctrl+Shift+G`	*Actions*	*Grouping items*
Raise	`Ctrl+]`	*Actions*	*Alignment and distribution*
Lower	`Ctrl+[`	*Actions*	*Alignment and distribution*
Bring to Front	`Ctrl+Shift+]`	*Actions*	*Alignment and distribution*
Send to Back	`Ctrl+Shift+[`	*Actions*	*Alignment and distribution*
Blochează Elementele Selectate	`Ctrl+L`	*Actions*	*Locking items*
Unlock All	`Ctrl+Shift+L`	*Actions*	*Locking items*
Align Items ►		*Actions*	*Alignment and distribution*
Distribute Items ►		*Actions*	*Moving and resizing items*
Redimensionare ►		*Actions*	*Moving and resizing items*

Adăugare element de meniu

These are tools to create layout items. Each of them is deeply described in *Elementele Compoziției* chapter.

Instrument	Bara de instrumente	Referință
Adăugare Hartă	*Instrumentar*	*Elementul Hartă*
Add 3D Map	*Instrumentar*	*The 3D Map Item*
Add Picture	*Instrumentar*	*The Picture Item*
Adăugare Etichetă	*Instrumentar*	*Elementul Etichetă*
Add Dynamic Text ►		*The Label Item*
Adăugare Legendă	*Instrumentar*	*Elementul Legendă*
Add Scale Bar	*Instrumentar*	*Elementul Scării Grafice*
Add North Arrow	*Instrumentar*	*The North Arrow Item*
Add Shape ►	*Instrumentar*	*The Regular Shape Item*
► Add Rectangle	*Instrumentar*	*The Regular Shape Item*
► Add Ellipse	*Instrumentar*	*The Regular Shape Item*
► Add Triangle	*Instrumentar*	*The Regular Shape Item*
Add Marker	*Instrumentar*	*The Marker Item*
Add Arrow	*Instrumentar*	*Elementul Săgeată*
Add Node Item ►	*Instrumentar*	*Formele Articulate*
► Add Polygon	*Instrumentar*	*Formele Articulate*
► Add Polyline	*Instrumentar*	*Formele Articulate*
Add HTML	*Instrumentar*	*Elementul HTML de tip cadru*
Add Attribute Table	*Instrumentar*	*The attribute table item*
Add Fixed Table	*Instrumentar*	*The fixed table item*
Add Elevation Profile	*Instrumentar*	*The Elevation Profile Item*

Atlas menu

Instrument	Scurtătură	Bara de instrumente	Referință
Previzualizare Atlas	`Ctrl+ALt+/`	*Atlas*	*Preview and generate an atlas*
First Feature	`Ctrl+<`	*Atlas*	*Preview and generate an atlas*
Previous Feature	`Ctrl+,`	*Atlas*	*Preview and generate an atlas*
Next Feature	`Ctrl+.`	*Atlas*	*Preview and generate an atlas*
Last feature	`Ctrl+>`	*Atlas*	*Preview and generate an atlas*
Print Atlas…		*Atlas*	*Preview and generate an atlas*
Exportare Atlas sub formă de Imagini…		*Atlas*	*Preview and generate an atlas*
Export Atlas as SVG…		*Atlas*	*Preview and generate an atlas*
Export Atlas as PDF…		*Atlas*	*Preview and generate an atlas*
Atlas Settings		*Atlas*	*Generarea unui Atlas*

Meniul Setărilor

The *Settings* ► *Layout Options…* menu is a shortcut to *Settings* ► *Options* ► *Layouts* menu of QGIS main canvas. Here, you can set some options that will be used as default on any new print layout:

- *Layout defaults* let you specify the default font to use;
- With *Grid appearance*, you can set the grid style and its color. There are three types of grid: **Dots**, **Solid** lines and **Crosses**;
- *Grid and guide defaults* defines spacing, offset and tolerance of the grid (see *Guides and Grid* for more details);
- *Layout Paths*: to manage list of custom paths to search print templates.

The *Settings* ► *Keyboard Shortcuts…* menu allows you to use the *shortcuts manager* in the print layout interface.

Contextual menus

Depending on where you right-click in the print layout dialog, you open a contextual menu with various features:

- Right-click on the menu bar or any toolbar and you get the list of layout panels and toolbars you can enable or disable in one-click.
- Right-click over a ruler and you can *Show Guides*, *Snap to Guides*, *Manage Guides…* opening the *Guides panel* or *Clear Guides*. It's also possible to hide the rulers.
- Right-click in the print layout canvas and:

  - You'll be able to *Undo* and *Redo* recent changes, or *Paste* any copied item (only available if no item is selected).
  - If you click over a page, you can additionally access the current *Page Properties* panel or *Remove Page*.
  - If you click on a selected item then you can cut or copy it as well as open the *Item Properties* panel.
  - If more than one item are selected, then you can either group them and/or ungroup if at least one group is already in the selection.
- Right-click inside a text box or spinbox widget of any layout panel provides edit options to manipulate its content.

The Layout Panel

In the *Layout* panel, you can define the global settings of your print layout.

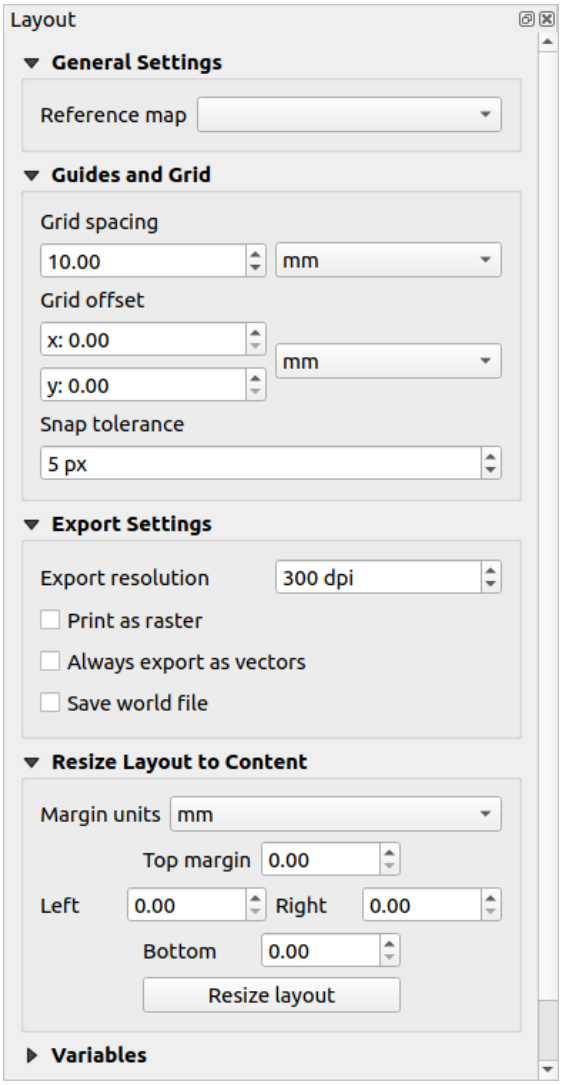


Fig. 22.3: Layout Settings in the Print Layout

General settings

In a print layout, you can use more than one map item. The *Reference map* represents the map item to use as the layout's master map. It's assigned as long as there's a map item in the layout. The layout will use this map in any of their properties and variables calculating units or scale. This includes exporting the print layout to georeferenced formats.

Moreover, new layout items such as scale bar, legend or north arrow have by default their settings (orientation, displayed layers, scale, …) bound to the map item they are drawn over, and fall back to the reference map if there is no overlapping map.

Guides and Grid

You can put some reference marks on your paper sheet to help you accurately place some items. These marks can be:

- simple horizontal or vertical lines (called **Guides**) put at the position you want (see *Panoul Ghidajelor* for guides creation).
- or regular **Grid**: a network of horizontal and vertical lines superimposed over the layout.

Settings like *Grid spacing* or *Grid offset* can be adjusted in this group as well as the *Snap tolerance* to use for items. The tolerance is the maximum distance below which the mouse cursor is snapped to a grid or a guide, while moving, resizing or creating an item.

Whether grid or guides should be shown is set in *View* menu. There, you can also decide if they might be used to snap layout items. When both a grid line and a guide line are within tolerance of a point, guides will always take precedence - since they have been manually set (hence, assumption that they have been explicitly placed at highly desirable snapping locations, and should be selected over the general grid).

Notă: In the *Settings* ► *Layout Options* menu, you can also set the grid and guides parameters exposed above. However, these options will only apply as defaults to new print layouts.

Setări de export

You can define a resolution to use for all exported maps in *Export resolution*. This setting can then be overridden each time you export a map.

Because of some advanced rendering options (*blending mode*, *effects*…), a layout item may need rasterization in order to be exported correctly. QGIS will individually rasterize it without forcing every other item to also be rasterized. This allows printing or saving as PostScript or PDF to keep items as much as possible as vectors, e.g. a map item with layer opacity won't force labels, scale bars, etc to be rasterized too. You can however:

- force all the items to be rasterized checking the *Print as raster* box;
- or use the opposite option, i.e. *Always export as vectors*, to force the export to keep items as vectors when exported to a compatible format. Note that in some cases, this could cause the output to look different to layout.

Where the format makes it possible (e.g., `.TIF`, `.PDF`) exporting a print layout results by default in a georeferenced file (based on the *Reference map* item in the *General settings* group). For other formats, georeferenced output requires you to generate a world file by checking *Save world file*. The world file is created beside the exported map(s), has the name of the page output with the reference map item and contains information to georeference it easily.

Resize layout to content

Using the *Resize page* tool in this group, you create a unique page composition whose extent covers the current contents of the print layout (with some optional *margins* around the cropped bounds).

Note that this behavior is different from the *crop to content* option in that all the items are placed on a real and unique page in replacement of all the existing pages.

Variabile

The *Variables* lists all the variables available at the layout's level (which includes all global and project's variables).

It also allows the user to manage layout-level variables. Click the button to add a new custom layout-level variable. Likewise, select a custom layout-level variable from the list and click the button to remove it.

More information on variables usage in the *General Tools* section.

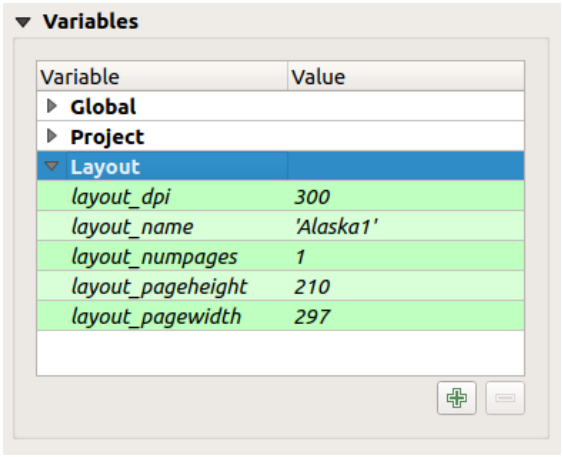


Fig. 22.4: Variables Editor in the Print Layout

Working with the page properties

A layout can be composed of several pages. For instance, a first page can show a map canvas, and a second page can show the attribute table associated with a layer, while a third one shows an HTML frame linking to your organization website. Or you can add many types of items on each page.

Adăugarea unei pagini noi

Futhermore, a layout can be made using different size and/or orientation of pages. To add a page, select the *Add Pages...* tool from the *Layout* menu or *Layout Toolbar*. The *Insert Pages* dialog opens and you are asked to fill:

- numărul paginilor de inserat;
- the position of the page(s): before or after a given page or at the end of the print layout;
- The *Page size*: it could be of a preset format page (`A4`, `B0`, `Legal`, `Letter`, `ANSI A`, `Arch A` and their derivatives as well as a resolution type, such as `1920x1080` or `1024x768`) with associated *Orientation* (Portrait or Landscape).

  The page size can also be of a `custom` format; In that case, you'd need to enter its *Width* and *Height* (with locked size ratio if needed) and select the unit to use among `mm`, `cm`, `px`, `pt`, `in`, `ft`... Conversion of entered values is automatically applied when switching from one unit to another.

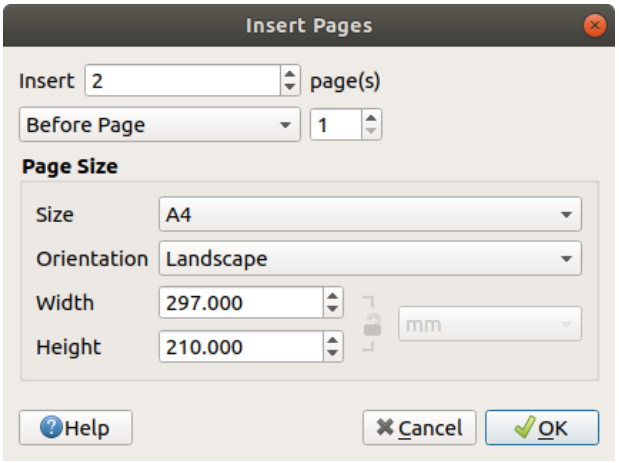


Fig. 22.5: Creating a new page in the Print Layout

Updating page properties

Any page can be later customized through the Page *Item Properties* panel. To access a page's properties, left-click on an empty section of the page or right-click on a page and select *Page Properties…*. The *Item Properties* panel opens with settings such as:

- for layouts with several pages, it displays the active page number and the count of available pages
- the *Page size* frame described above. You can modify each property using the data defined override options (see *Explore Data-defined override buttons with atlas* for a use case);
- the *Exclude page from exports* to control whether the current page with its content should be included in the *layout output*;
- the *Background* of the current page using the *color* or *symbol* you want.

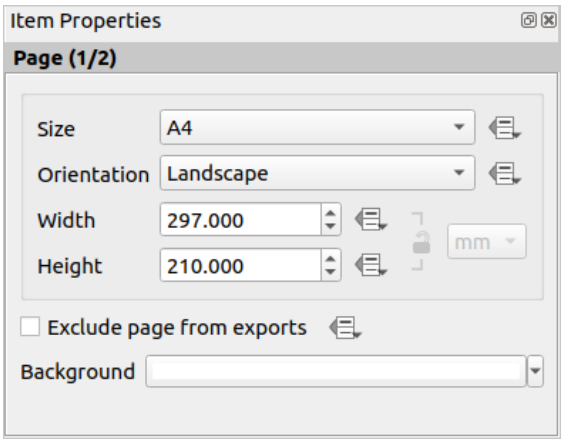


Fig. 22.6: Fereastra proprietăților paginii

Panoul Ghidajelor

Guides are vertical or horizontal line references you can place on a layout page to assist you on items placement, when creating, moving or resizing them. To be active, guides require the *View* ► *Show Guides* and *View* ► *Snap to Guides* options to be checked. To create a guide, there are two different methods:

- if the *View* ► *Show Rulers* option is set, drag out a ruler and release the mouse button within the page area, at the desired position.
- for more precision, use the *Guides* panel from the *View* ► *Toolbox* ► or by selecting *Manage guides for page…* from the page's contextual menu.

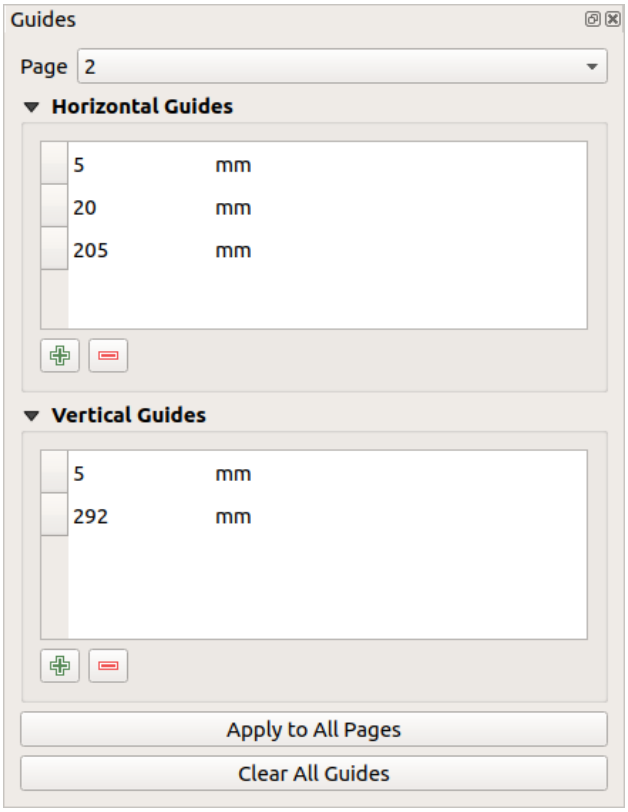


Fig. 22.7: The Guides panel

The *Guides* panel allows creation of snap lines at specific locations:

1. Select the *Page* you'd like to add the guides to
2. Click the Add new guide button and enter the coordinates of the horizontal or vertical line. The origin is at the top left corner. Different units are available for this.

   The panel also allows adjusting the position of existing guides to exact coordinates: double-click and replace the value.
3. The *Guides* panel lists only the items for the current page. It allows creation or removal of guides only in the current page. However, you can use the *Apply to All Pages* button to replicate the guide configuration of the current page to the other pages in the layout.
4. To delete a guide, select it and press the Remove selected guide button. Use *Clear All Guides* to remove all the guides in the current page.

Sfat: Snapping to existing layout items

Other than guides and grids, you can use existing items as snapping references when moving, resizing or creating new items; these are called **smart guides** and require *View* ► *Smart Guides* option to be checked. Anytime the mouse pointer is close to an item's bound, a snapping cross appears.

Panoul Elementelor

The *Items* panel offers some options to manage selection and visibility of items. All the items added to the print layout canvas (including *items group*) are shown in a list and selecting an item makes the corresponding row selected in the list as well as selecting a row does select the corresponding item in the print layout canvas. This is thus a handy way to select an item placed behind another one. Note that a selected row is shown as bold. Multiple items selection is also possible holding `Shift` for contiguous items, and `Ctrl` key for non-contiguous ones.

For any available item, you can:

- set it visible or not;
- lock or unlock its position;
- sort its Z position. You can move up and down each item in the list with a click and drag. The upper item in the list will be brought to the foreground in the print layout canvas. By default, a newly created item is placed in the foreground.
- change the item ID by double-clicking the text;
- right-click an item and select whether to copy or delete it or open its *properties panel*.

Once you have found the correct position for an item, you can lock it by ticking the box in column. Locked items are **not** selectable on the canvas. Locked items can be unlocked by selecting the item in the *Items* panel and unchecking the tickbox or you can use the icons on the toolbar.

The Undo History Panel: Revert and Restore actions

During the layout process, it is possible to revert and restore changes. This can be done with the revert and restore tools available in the *Edit* menu, the *Layout* toolbar or the contextual menu any time you right-click in the print layout area:

- Anulează ultima modificare
- Restaurează ultima modificare

This can also be done by mouse click within the *Undo history* panel (see Fig. 22.8). The History panel lists the last actions done within the print layout. Select the point you want to revert to and once you do a new action all the actions done after the selected one will be removed.

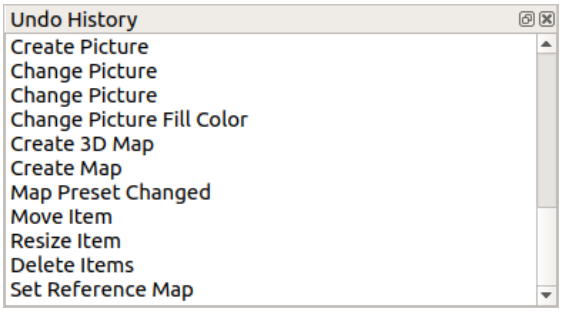


Fig. 22.8: Undo History in the Print Layout

22.2 Elementele Compoziției

22.2.1 Layout Items Common Options

QGIS provides a large set of items to layout a map. They can be of map, legend, scale bar, picture, table, north arrow, image type… They however share some common options and behavior that are exposed below.

Creating a layout item

Items can be created using different tools, either from scratch or based on existing items.

To create a layout item from scratch:

1. Select the corresponding tool either from the *Add Item* menu or the *Toolbox* bar.
2. Then:
   - Click on the page and fill the size and placement information requested in the *New Item Properties* dialog that pops up (for details, see *Position and Size*);

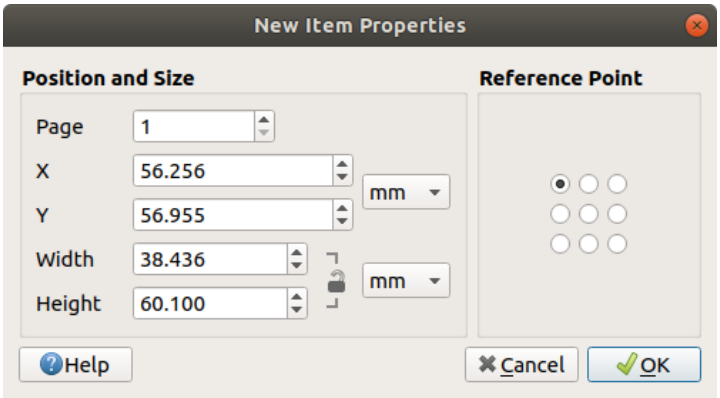


Fig. 22.9: New Item properties dialog

   - Or click-and-drag to define the initial size and placement of the item. You can rely on *grids and guides* snapping for a better position.

Notă: Because they can have particular shapes, drawing node or arrow items does not work with one-click nor click-and-drag methods; you need to click and place each node of the item. See *Formele Articulate* for more details.

You can also:

1. Select an existing item with the Select/Move item button from the *Toolbox* toolbar
2. Use the contextual menu or the *Edit* menu tools to copy/cut the item and paste it at the mouse position as a new item.

   You can also use the *Paste in Place* (`Ctrl+Shift+V`) command to duplicate an item from one page to another and place it in the new page at the same coordinates as the original.

Moreover, you can create items using a print layout template (for details, see *Managerul de Compoziții*) through the *Layout* ► *Add Items from Template…* command.

Sfat: Add layout items using the file browser

From your file browser or using the *Browser* panel, drag-and-drop a print layout template (`.qpt` file) onto a print layout dialog and QGIS automatically adds all items from that template to the layout.

Interacting with layout items

Each item inside the print layout can be moved and resized to create a perfect layout. For both operations the first step is to activate the Select/Move item tool and click on the item.

You can select multiple items with the Select/Move item button: click and drag over the items or hold the `Shift` button and click on each of the items you want. To deselect an item, click on it holding the `Shift` button.

Each time there's a selection, count of selected items is displayed on the status bar. Inside the *Edit* menu, you can find actions to select all the items, clear all selections, invert the current selection and more…

Moving and resizing items

Unless *View* ► *Show Bounding Boxes* option is unchecked, a selected item will show squares on its boundaries; moving one of them with the mouse will resize the item in the corresponding direction. While resizing, holding `Shift` will maintain the aspect ratio. Holding `Alt` will resize from the item center.

To move a layout item, select it with the mouse and move while holding the left button. If you need to constrain the movements to the horizontal or vertical axis, hold the `Shift` button on the keyboard while moving the mouse. You can also move a selected item using the `Arrow keys` on the keyboard; if the movement is too slow, you can speed it up by holding `Shift`. If you need better precision, use the *Position and size* properties, or grid/guides snapping as explained above for item's creation.

Resizing or moving several items at once is made the same way as for a single item. QGIS however provides some advanced tools to automatically resize a selection of items following different rules:

- each item height matches the tallest or the shortest selected item;
- each item width matches the widest or the narrowest selected item;
- resizes items to squares: each item is enlarged to shape a square.

Likewise, there are *tools* available to organize multiple items position by distributing equidistantly:

- edges (left, right, top or bottom) of items;
- centers of items horizontally or vertically;
- spacing between items horizontally or vertically.

Grouping items

Grouping items allows you to manipulate a set of items like a single one: you can easily resize, move, delete, copy the items as a whole.

To create a group of items, select more than one and press the . *Group* button on the *Items* menu or the *Actions* toolbar, or from the right-click menu. A row named `Group` is added to the *Items* panel and can be locked or hidden like any other *Items panel's object*. Grouped items are **not individually** selectable on the canvas; use the Items panel for direct selection and access the item's properties panel.

Locking items

Once you have found the correct position for an item, you can lock it by using the *Lock selected items* button in the *Items* menu or the *Actions* toolbar, or ticking the box next to the item in the *Items* panel. Locked items are **not** selectable on the canvas.

Locked items can be unlocked by selecting the item in the *Items* panel and unchecking the tickbox or you can use the icons on the toolbar.

Alignment and distribution

Tools for raising or lowering the Z position of items in the layout are inside the Raise selected items pull-down menu. Choose an element on the print layout canvas and select the matching functionality to raise or lower the selected element over the other elements. This order is shown in the *Items* panel. You can also raise or lower objects in the *Items* panel by clicking and dragging an object's label in this list.

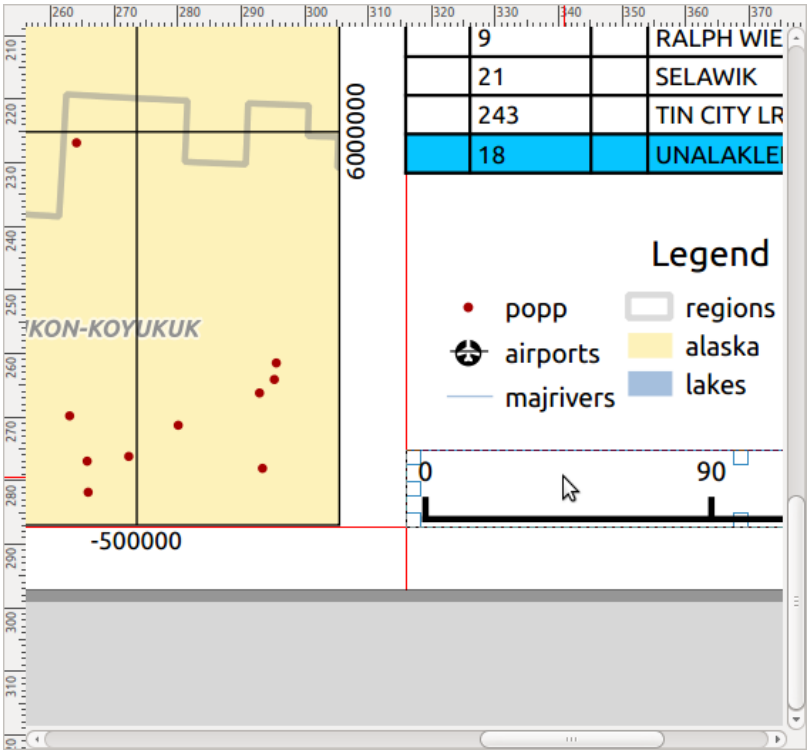


Fig. 22.10: Alignment helper lines in the print layout

There are several alignment options available within the Align selected items pull-down menu (see Fig. 22.10). To use an alignment function, you first select the elements and then click on one of the alignment icons:

- *Align Left* or *Align Right*;
- *Align Top* or *Align Bottom*;
- *Align Center* horizontally or *Align Center Vertical*.

All selected elements will then be aligned to their common bounding box. When moving items on the layout canvas, alignment helper lines appear when borders, centers or corners are aligned.

Another way to improve layout items placement is by adjusting the spacing between them, over the layout page. This can be done by selecting the items and press the *Distribute Left Edges* drop-down menu to:

- *Distribute Left Edges* or *Distribute Right Edges* of items equidistantly
- *Distribute Top edges* or *Distribute Bottom Edges* of items equidistantly
- *Distribute Horizontal Centers* or *Distribute Vertical Centers* of items equidistantly
- Add equal space between items: *Distribute Horizontal Spacing Equally* or *Distribute Vertical Spacing Equally*

Items Common Properties

Layout items have a set of common properties you will find at the bottom of the *Item Properties* panel: Position and size, Rotation, Frame, Background, Item ID, Variables and Rendering (see Fig. 22.11).

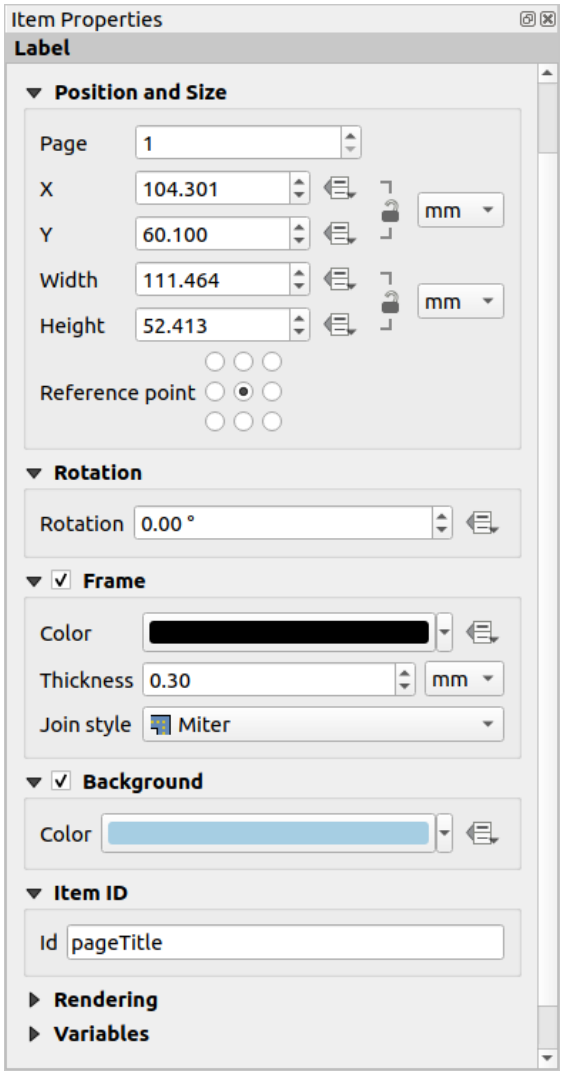


Fig. 22.11: Common Item Properties groups

Notă: The Data defined override icon next to most of the options means that you can associate that property with a

layer, features attributes, geometry or with any other layout item's property, using *expressions* or *variables*. For more information see *Configurarea suprascrierii definită de date*.

- The *Position and size* group lets you define the size and position of the frame which contains the item (see *Position and Size* for more information).
- *Rotirea* stabilește rotația elementului (în grade).
- The *Frame* shows or hides the frame around the item. Use the *Color*, *Thickness* and *Join style* widgets to adjust those properties.
- In the *Background* group you can create or pick a background *Color*. Transparency can be adjusted through altering the alpha field settings.
- Use the *Item ID* to create a relationship to other print layout items. This is used with QGIS server and other potential web clients. You can set an ID on an item (for example, a map or a label), and then the web client can send data to set a property (e.g., label text) for that specific item. The GetProjectSettings command will list the items and IDs which are available in a layout.
- *Rendering* mode helps you set whether and how the item can be displayed: you can, for instance, apply *blending mode*, adjust the opacity of the item or *Exclude item from exports*.

Position and Size

Extending the features of the *New Item Properties* dialog with data-defined capabilities, this group allows you to place the items accurately.

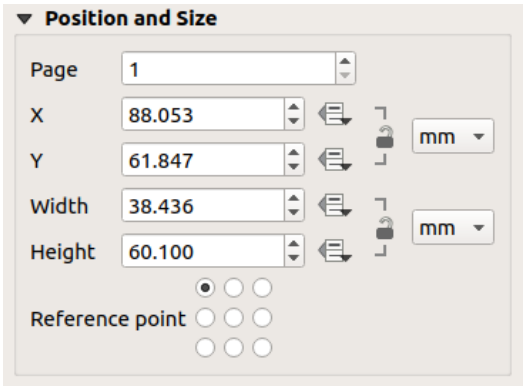


Fig. 22.12: Position and size

- the actual number of the page to place the item on;
- the reference point of the item;
- the *X* and *Y* coordinates of the *Reference point* of the item on the chosen page. The ratio between these values can be locked by clicking on the button. Changes made to a value using the widget or the Select/Move item tool will be reflected in both of them;
- the *Width* and *Height* of the item bounding box. As for coordinates, the ratio between width and height can be locked.

Modul de randare

QGIS allows advanced rendering for layout items just like vector and raster layers.

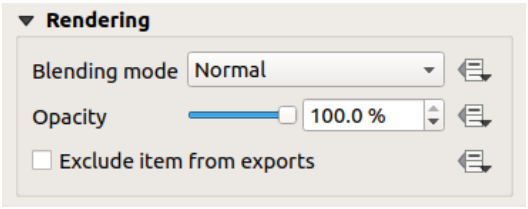


Fig. 22.13: Modul de randare

- *Blending mode*: With this tool you can achieve effects which would otherwise only be achieved using graphic rendering software. The pixels of your overlaying and underlaying items can be mixed according to the mode set (see *Modurile de Fuziune* for description of each effect).
- *Opacity*: You can make the underlying item in the layout visible with this tool. Use the slider to adapt the visibility of your item to your needs. You can also make a precise definition of the percentage of visibility in the menu beside the slider.
- *Exclude item from exports*: Combined with data-defined expressions, you can decide to make an item invisible in some exports. After activating this checkbox, the item will not be included in export to PDF, print etc..

Variabile

The *Variables* lists all the variables available at the layout item's level (which includes all global, project and layout's variables). Layout map items alhave an additional *Map settings* section for variables that provide easy access to values like the map's scale, extent, and so on.

In *Variables*, it's also possible to manage layout item level variables. Click the button to add a new custom variable. Likewise, select any custom item-level variable from the list and click the button to remove it.

More information on variables usage in the *Storing values in Variables* section.

22.2.2 Elementul Hartă

The map item is the main frame that displays the map you've designed in the map canvas. Use the *Add Map* tool following *items creation instructions* to add a new map item that you can later manipulate the same way as exposed in *Interacting with layout items*.

By default, a new map item shows the current status of the *map canvas* with its extent and visible layers. You can customize it thanks to the *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities:

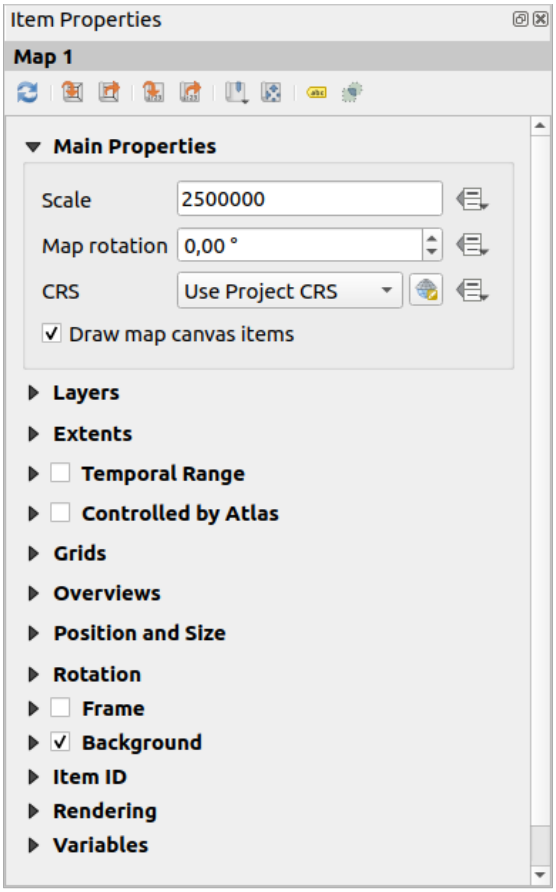


Fig. 22.14: Panoul Proprietăților pentru Elementul de tip Hartă

The Toolbar

The Map *Item Properties* panel embeds a toolbar with the following functionalities:

- Update map preview
- Set map canvas to match main canvas extent
- View current map extent in main canvas
- Set map scale to match main canvas scale
- Set main canvas to match current map scale
- Bookmarks: set the map item extent to match an existing spatial bookmark
- Interactively edit map extent: pan and zoom interactively within the map item
- Labeling settings: control feature label behaviour (placement, visibility…) in the layout map item extent:
  - set a *Margin from map edges*, a data definable distance from the map item's limits inside which no label should be displayed
  - *Allow truncated labels on edges of map*: controls whether labels which fall partially outside of the map item allowed extent should be rendered. If checked, these labels will be shown (when there's no way to place them fully within the visible area). If unchecked then partially visible labels will be skipped.

  - *Label blocking items*: allows other layout items (such as scalebars, north arrows, inset maps, etc) to be marked as a blockers for the map labels in the **active** map item. This prevents any map labels from being placed under those items - causing the labeling engine to either try alternative placement for these labels or discard them altogether.

    If a *Margin from map edges* is set, the map labels are not placed closer than the specified distance from the checked layout items.

  - *Show unplaced labels*: can be used to determine whether labels are missing from the layout map (e.g. due to conflicts with other map labels or due to insufficient space to place the label) by highlighting them in a *predefined color*.

- Clipping settings: allows to clip the map item to the atlas feature and to shape and polygon items:

  - *Clip to atlas feature*: you can determine that the layout map item will be clipped automatically to the current *atlas feature*.

    There are different clipping modes available:

    * *Clip During Render Only*: applies a painter based clip, so that portions of vector features which sit outside the atlas feature become invisible
    * *Clip Feature Before Render*: applies the clip before rendering features, so borders of features which fall partially outside the atlas feature will still be visible on the boundary of the atlas feature
    * *Render Intersecting Features Unchanged*: renders all features which intersect the current atlas feature, but without clipping their their geometry.

    You can *Force labels inside atlas feature*. If you don't want to *Clip all layers* to the atlas feature you can use the *Clip selected layers* option.

  - *Clip to item*: it is possible to change the shape of the map item by using a *shape* or *polygon* item from the print layout. When you enable this option the map will be automatically clipped to the selected shape in the combobox. Again, the above mentioned clipping modes are available and labels can be forced to display only inside the clipping shape.

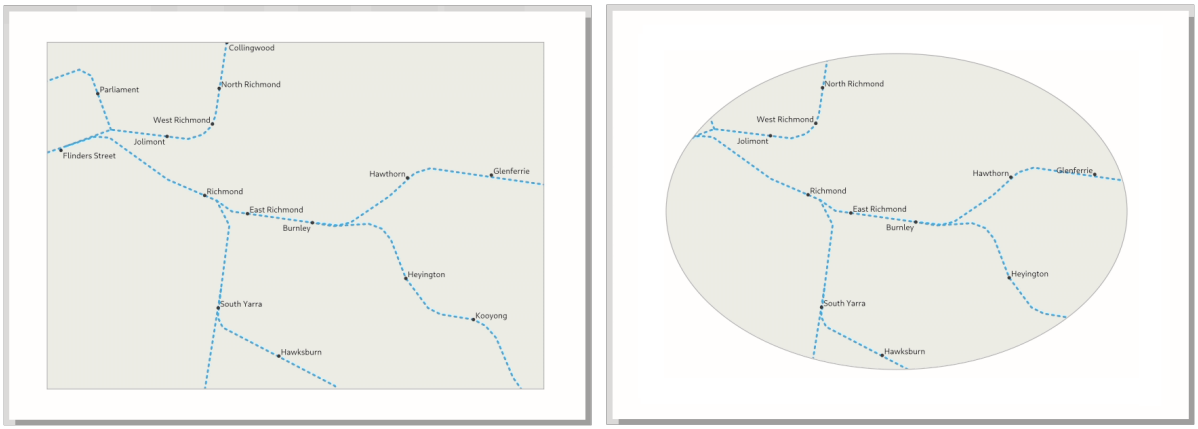


Fig. 22.15: Clipping a layout map item to shapes

Proprietăți principale

In the *Main properties* group (see Fig. 22.14) of the map *Item Properties* panel, available options are:

- The *Update Preview* button to refresh the map item rendering if the view in map canvas has been modified. Note that most of the time, the map item refresh is automatically triggered by the changes;
- The *Scale* to manually set the map item scale;
- The *Map rotation* allows you to rotate the map item content clockwise in degrees. The rotation of the map canvas can be imitated here;
- The *CRS* allows you to display the map item content in any *CRS*. It defaults to `Use project CRS`;
- *Draw map canvas items* lets you show in the print layout *annotations* that are placed on the main map canvas.

Straturi

By default, map item appearance is synced with the map canvas rendering meaning that toggling visibility of the layers or modifying their style in the *Layers Panel* is automatically applied to the map item. Because, like any other item, you may want to add multiple map items to a print layout, there's a need to break this synchronization in order to allow showing different areas, layer combinations, at different scales… The *Layers* properties group (see Fig. 22.16) helps you do that.

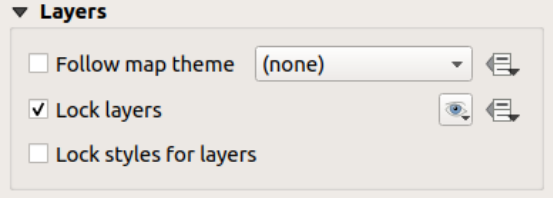


Fig. 22.16: Map Layers group

If you want to keep the map item consistent with an existing *map theme*, check *Follow map theme* and select the desired theme in the drop-down list. Any changes applied to the theme in QGIS» main window (using the replace theme function) will automatically affect the map item. If a map theme is selected, the *Lock styles for layers* option is disabled because *Follow map theme* also updates the style (symbology, labels, diagrams) of the layers.

To lock the layers shown in a map item to the current map canvas visibility, check *Lock layers*. When this option is enabled, any changes on the layers» visibility in QGIS» main window will not affect the layout's map item. Nevertheless, style and labels of locked layers are still refreshed according to QGIS» main window. You can prevent this by using *Lock styles for layers*.

Instead of using the current map canvas, you can also lock the layers of the map item to those of an existing map theme: select a map theme from the Set layer list from a map theme drop-down button, and the *Lock layers* is activated. The set of visible layers in the map theme is from now on used for the map item until you select another map theme or uncheck the *Lock layers* option. You then may need to refresh the view using the Refresh view button of the *Navigation* toolbar or the *Update Preview* button seen above.

Note that, unlike the *Follow map theme* option, if the *Lock layers* option is enabled and set to a map theme, the layers in the map item will not be refreshed even if the map theme is updated (using the replace theme function) in QGIS» main window.

Locked layers in the map item can also be *data-defined*, using the icon beside the option. When used, this overrides the selection set in the drop-down list. You need to pass a list of layers separated by `|` character. The following example locks the map item to use only layers `layer 1` and `layer 2`:

```
concat ('layer 1', '|', 'layer 2')
```

Extinderi

The *Extents* group of the map item properties panel provides the following functionalities (see Fig. 22.17):

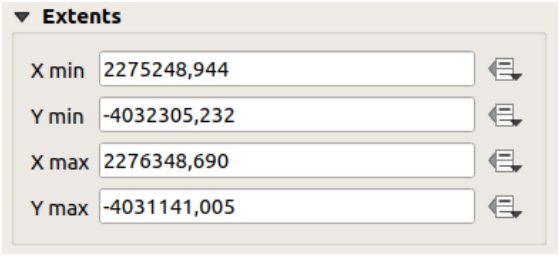


Fig. 22.17: Map Extents group

The **Extents** area displays `X` and `Y` coordinates of the area shown in the map item. Each of these values can be manually replaced, modifying the map canvas area displayed and/or map item size. The extent can also be modified using tools at the top of the map item panel such as:

- Set map canvas to match main canvas extent
- Set map scale to match main canvas scale

You can also alter a map item extent using the Move item content tool: click-and-drag within the map item to modify its current view, keeping the same scale. With the tool enabled, use the mouse wheel to zoom in or out, modifying the scale of the shown map. Combine the movement with `Ctrl` key pressed to have a smaller zoom.

Temporal range

The *Temporal range* group of the map item properties panel provides the options to control layers rendering in the map item based on a temporal range. Only layers whose temporal properties overlap with the time range set by the *Start* and *End* dates are displayed in the map item.

The associated data-defined widgets help make the time range dynamic, and allow outputting temporal *atlases*, i.e. automated maps with fixed spatial extent and whose contents vary based on time. For example, using as coverage layer a csv file with a start and end pair of fields and a number of rows representing date ranges, enable both the temporal range and control by atlas in the map item properties and hit atlas export.

Controlled by atlas

The *Controlled by atlas* group properties is available only if an *atlas* is active in the print layout. Check this option if you want the map item being ruled by the atlas; when iterating over the coverage layer, the map item extent is panned/zoomed to the atlas feature following:

- *Margin around features*: zooms to the feature at the best scale, keeping around each a margin representing a percentage of the map item width or height. The margin can be the same for all features or *set variable*, e.g., depending on map scale;
- *Predefined scale (best fit)*: zooms to the feature at the project *predefined scale* where the atlas feature best fits;

- *Fixed scale*: atlas features are panned from one to another, keeping the same scale of the map item. Ideal when working with features of same size (e.g., a grid) or willing to highlight size differences among atlas features.

Grile

With grids, you can add, over your map, information relative to its extent or coordinates, either in the map item projection or a different one. The *Grids* group provides the possibility to add several grids to a map item.

- With the and buttons you can add or remove a selected grid;
- With the and buttons you can move up and down a grid in the list, hence move it on top or bottom of another one, over the map item.

Double-click the added grid to rename it.

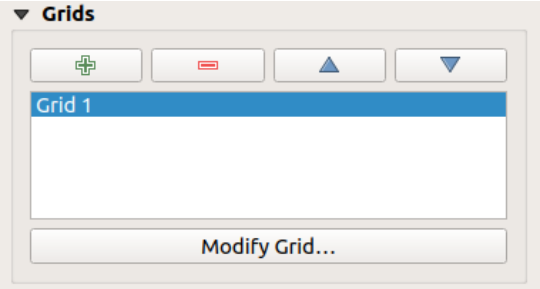


Fig. 22.18: Dialogul Grilelor Hărții

To modify a grid, select it and press the *Modify Grid…* button to open the *Map Grid Properties* panel and access its configuration options.

Grid Appearance

In the *Map Grid Properties* panel, check *Grid enabled* to show the grid on the map item.

As grid type, you can specify to use a:

- **Solid**: shows a line across the grid frame. The *Line style* can be customized using *color* and *symbol* selector widget;
- **Cross**: displays segment at the grid lines intersection for which you can set the *Line style* and the *Cross width*;
- **Markers**: only displays customizable markers symbol at grid lines intersection;
- or **Frame and annotations only**.

Other than the grid type, you can define:

- the *CRS* of the grid: by default, it will follow the map item CRS. Press Select CRS button to set it to a different CRS.
- the *Interval* type to use for the grid references:
  - **Map Units**: you set a distance within the map (in the unit of the grid CRS) between consecutive grid references in the *X* and *Y* directions. The number of grid ticks will vary depending on the map scale.
  - choosing **Fit Segment Width** will dynamically select the grid interval based on the map extent to a „pretty” interval. That optimal interval is calculated within a range of distances whose *Minimum* and *Maximum* values can be customized.

  - With **Millimeters** or **Centimeters**, you set a distance on the paper between consecutive grid references in the *X* and *Y* directions. The number of grid ticks will be the same whatever the map scale.
- the *Offset* from the map item edges, in the *X* and/or the *Y* direction
- and the *Blend mode* of the grid (see *Modurile de Fuziune*) when compatible.

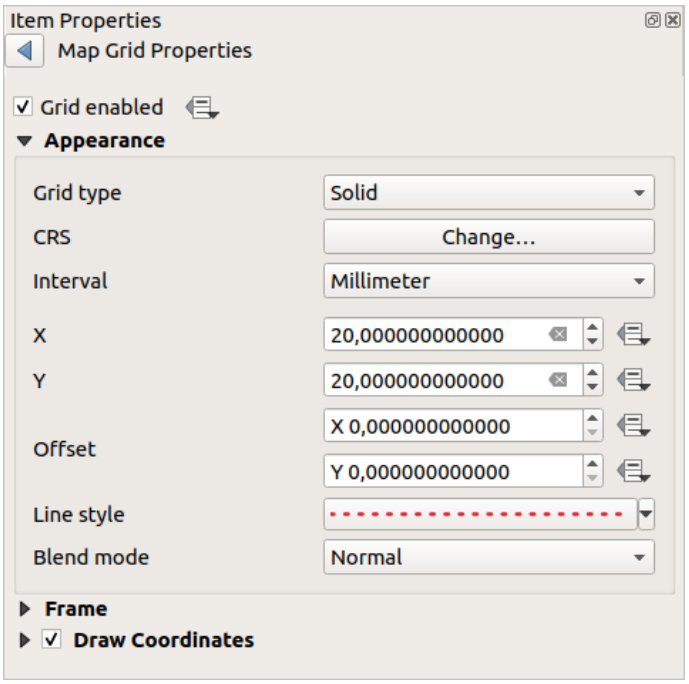


Fig. 22.19: Grid Appearance Dialog

Grid Frame

There are different options to style the frame that holds the map. The following options are available: `No Frame`, `Zebra`, `Zebra (nautical)`, `Interior ticks`, `Exterior ticks`, `Interior and Exterior ticks`, `Line border` and `Line border (nautical)`.

When compatible, it's possible to set the *Frame size*, a *Frame margin*, the *Frame line thickness* with associated color and the *Frame fill colors*.

Using `Latitude/Y only` and `Longitude/X only` values in the divisions section you can prevent a mix of latitude/Y and longitude/X coordinates showing on each side when working with rotated maps or reprojected grids. Also you can choose to set visible or not each side of the grid frame.

When the map item extent is rotated (from the *Main properties* group) or the grid has a different CRS applied, grid lines may not be orthogonal to the map item sides. This can result in bad looking of the grid when styled with interior and/or exterior ticks. Checking *Follow grid rotation* will align the ticks with grid lines. Moreover, you can adjust some more properties:

- *Ticks alignment*: The interior and/or exterior ticks will be parallel to their corresponding grid line. Their alignment can be:
  - **Orthogonal**: ticks on the same side end at one line, parallel to the side. This can result e.g. in some ticks getting longer when with a low angle to the frame.
  - **Fixed length**: all ticks have the same length, so they may not align
- *Skip below angle*: prevents displaying ticks for grid lines intersecting the frame border below a specified threshold

- *Margin from map corner*: prevents displaying ticks too close to the map corners, because they could overlap and/or be out of bounds.

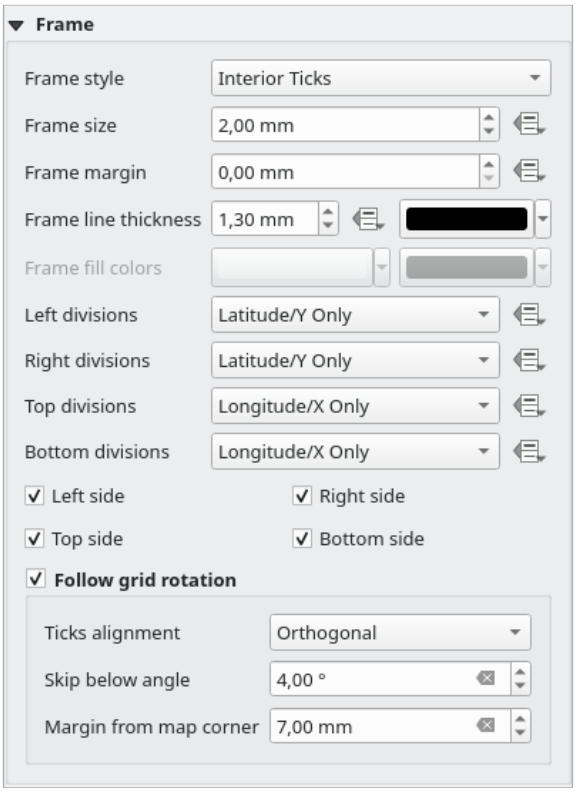


Fig. 22.20: Dialogul Cadrului Grilei

Coordinates

The *Draw coordinates* checkbox allows you to add coordinates to the map frame. Displayed values relate to the chosen *grid interval* unit. You can choose the annotation numeric format, the options range from decimal to degrees, minute and seconds, with or without suffix, aligned or not and a custom format using the expression dialog.

For each of the *Left*, *Right*, *Top* and *Bottom* sides of the grid frame, you can indicate:

- whether to render the coordinates: **Show all**, **Show latitude/Y only**, **Show longitude/X only**, **Disabled**. Showing only Latitude/Y or Longitude/X values in the divisions helps prevent a mix of latitude/Y and longitude/X coordinates showing on each side when working with rotated maps or reprojected grids.
- the relative position of the text to the grid frame: **Outside frame** or **Inside frame**
- the placement and orientation of the annotation:
  - **Horizontal**
  - **Vertical ascending**, **Vertical descending**
  - **Boundary direction**
  - **Above tick**, **On tick**, **Under tick** when a tick-based frame is used

You can also define the *Font properties* (font, size, color, buffer,…) the *Distance to the map frame* and the *Coordinate precision* (number of decimals) for the drawn annotations.

Follow grid rotation: available when the map extent is rotated or the grid is reprojected, it helps you adjust the annotations placement. Depending on the selected placement mode, the annotations are also rotated:

- *Annotations alignment*: it can be **Orthogonal** or of **Fixed length**

- *Skip below angle*: prevents displaying annotations for grid lines intersecting the frame border below a specified threshold
- *Margin from map corner*: prevents displaying annotations too close to the map corners, because they could overlap and/or be out of bounds.

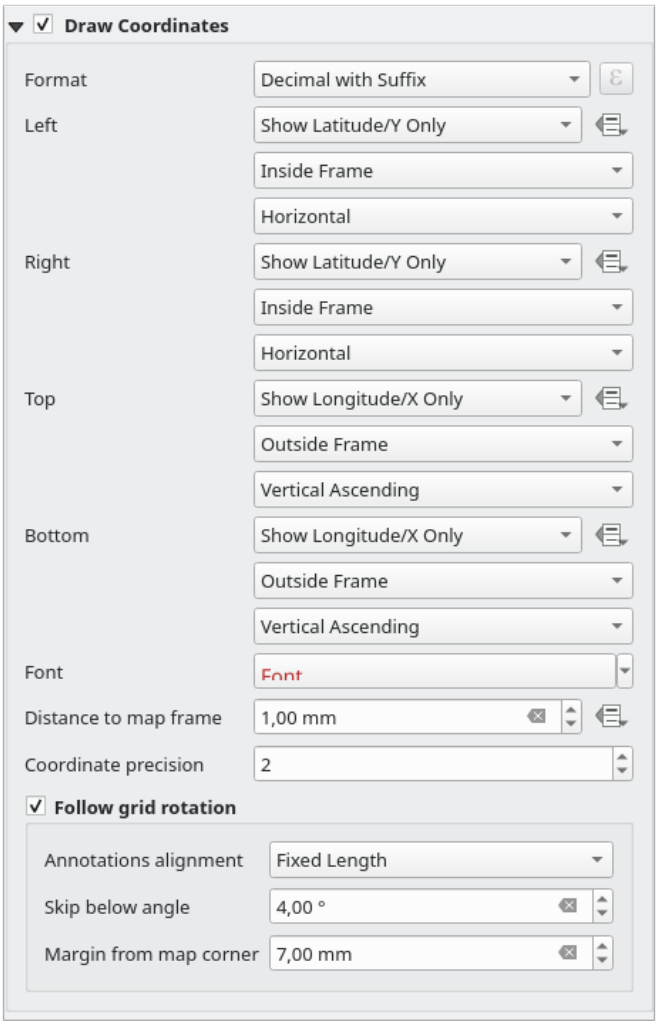


Fig. 22.21: Dialogul de Desenare a Coordonatelor Grilei

Prezentare generală

Sometimes you may have more than one map in the print layout and would like to locate the study area of one map item on another one. This could be for example to help map readers identify the area in relation with its larger geographic context shown in the second map.

The *Overviews* group of the map panel helps you create the link between two different maps extent and provides the following functionalities:

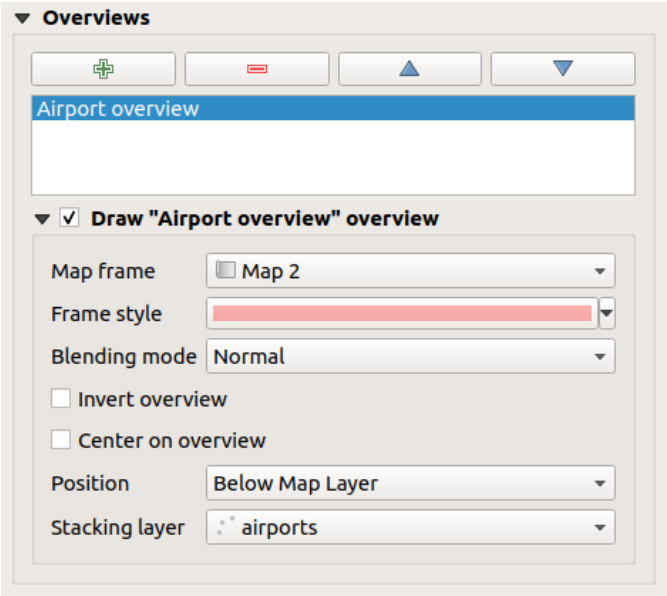


Fig. 22.22: Map Overviews group

To create an overview, select the map item on which you want to show the other map item's extent and expand the *Overviews* option in the *Item Properties* panel. Then press the button to add an overview.

Initially this overview is named «Overview 1» (see Fig. 22.22). You can:

- Rename it with a double-click
- With the and buttons, add or remove overviews
- With the and buttons, move an overview up and down in the list, placing it above or below other overviews in the map item (when they are at the same *stack position*).

Then select the overview item in the list and check the *Draw „<name_overview>" overview* to enable the drawing of the overview on the selected map frame. You can customize it with:

- The *Map frame* selects the map item whose extents will be shown on the present map item.
- The *Frame Style* uses the *symbol properties* to render the overview frame.
- The *Blending mode* allows you to set different transparency blend modes.
- The *Invert overview* creates a mask around the extents when activated: the referenced map extents are shown clearly, whereas the rest of the map item is blended with the frame fill color (if a fill color is used).
- The *Center on overview* pans the map item content so that the overview frame is displayed at the center of the map. You can only use one overview item to center, when you have several overviews.
- The *Position* controls exactly where in the map item's layer stack the overview will be placed, e.g. allowing an overview extent to be drawn below some feature layers such as roads whilst drawing it above other background layers. Available options are:
  - *Below map*
  - *Below map layer* and *Above map layer*: place the overview frame below and above the geometries of a layer, respectively. The layer is selected in the *Stacking layer* option.
  - *Below map labels*: given that labels are always rendered above all the feature geometries in a map item, places the overview frame above all the geometries and below any label.
  - *Above map labels*: places the overview frame above all the geometries and labels in the map item.

22.2.3 The 3D Map Item

The 3D Map item is used to display a *3D map view*. Use the *Add 3D Map* button, and follow *items creation instructions* to add a new 3D Map item that you can later manipulate the same way as demonstrated in *Interacting with layout items*.

By default, a new 3D Map item is empty. You can set the properties of the 3D view and customize it in the *Item Properties* panel. In addition to the *common properties*, this feature has the following functionalities (Fig. 22.23):

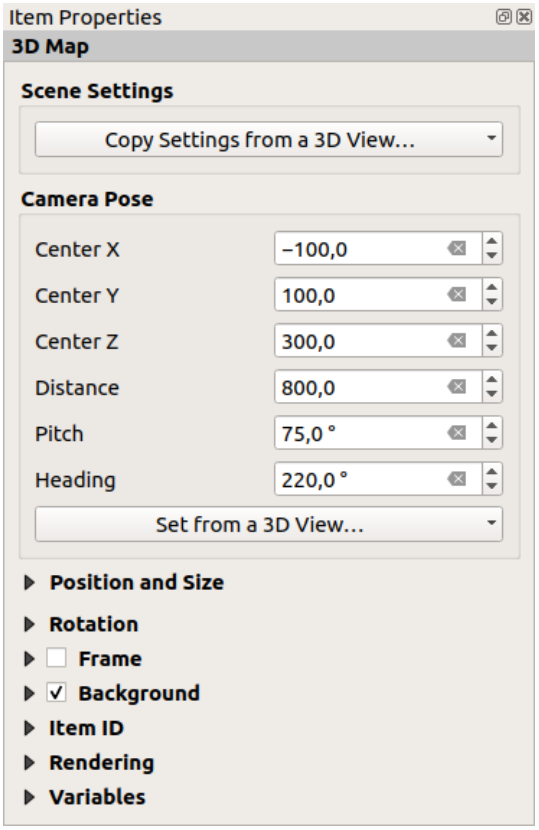


Fig. 22.23: 3D Map Item Properties

Scene settings

Press *Copy Settings from a 3D View…* to choose the 3D map view to display.

The 3D map view is rendered with its current configuration (layers, terrain, lights, camera position and angle…).

Camera pose

- *Center X* sets the X coordinate of the point the camera is pointing at
- *Center Y* sets the Y coordinate of the point the camera is pointing at
- *Center Z* sets the Z coordinate of the point the camera is pointing at
- *Distance* sets the distance from the camera center to the point the camera is pointing at
- *Pitch* sets the rotation of the camera around the X-axis (vertical rotation). Values from 0 to 360 (degrees). 0°: terrain seen straight from above; 90°: horizontal (from the side); 180°: straight from below; 270°: horizontal, upside down; 360°: straight from above.

- *Heading* sets the rotation of the camera around the Y-axis (horizontal rotation - 0 to 360 degrees). 0°/360°: north; 90°: west; 180°: south; 270°: east.

The *Set from a 3D View…* pull-down menu lets you populate the items with the parameters of a 3D View.

22.2.4 Elementul Etichetă

The *Label* item is a tool that helps decorate your map with texts that would help to understand it; it can be the title, author, data sources or any other information… You can add a label with the *Add Label* tool following *items creation instructions* and manipulate it the same way as exposed in *Interacting with layout items*.

By default, the label item provides a default text that you can customize using its *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities (see Fig. 22.24):

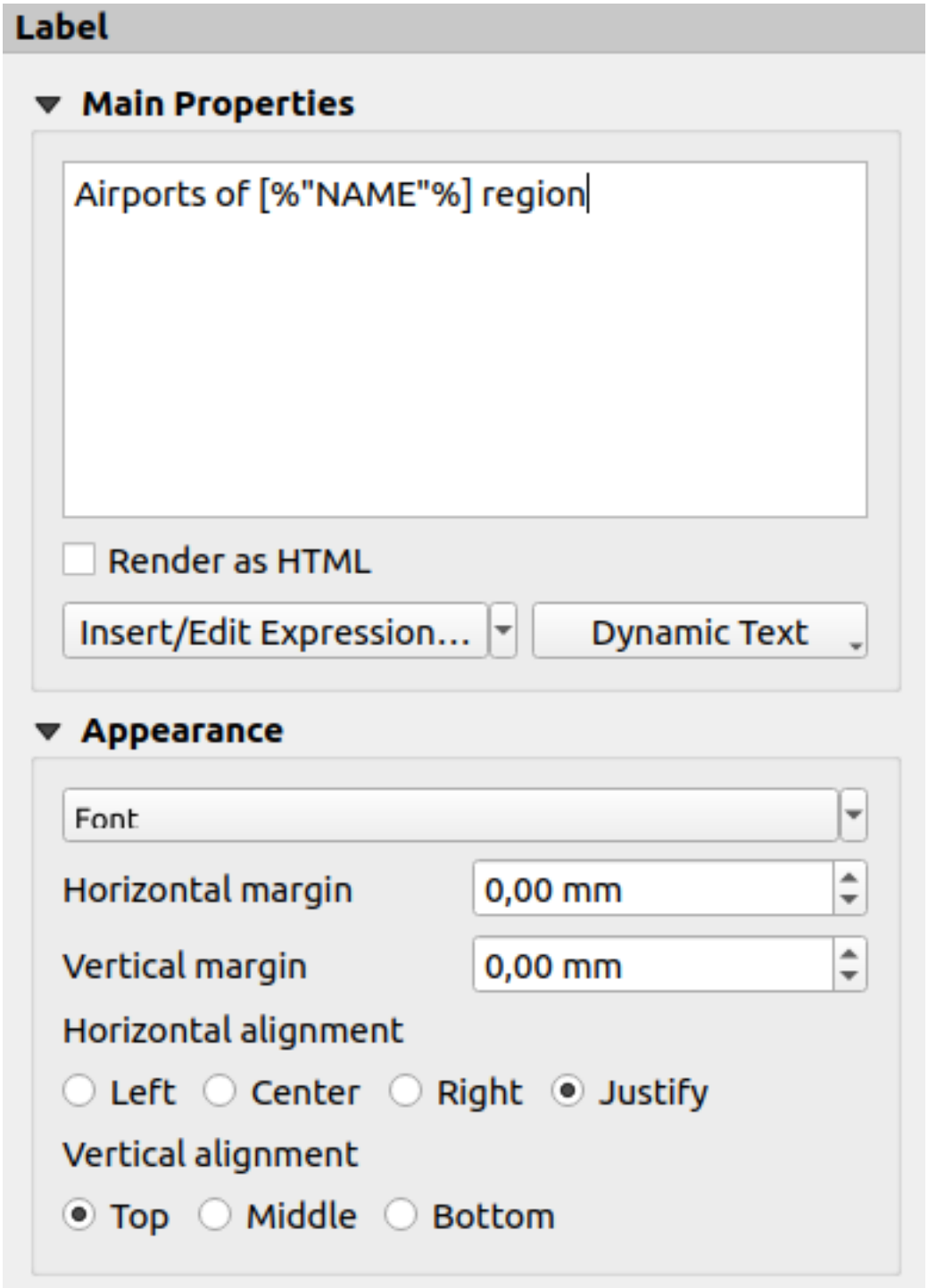


Fig. 22.24: Panoul Proprietăților Elementului Etichetă

Proprietăți principale

The *Main properties* group is the place to provide the text of the label. The text can be static, dynamic with *expression* functions and variables, and/or formatted with HTML. Dynamic parts of a label need to be surrounded by `[%` and `%]` in order to be interpreted and evaluated as such.

- To use expressions in labels, you can click on *Insert/Edit Expression…* button, write your formula as usual and when the dialog is applied, QGIS automatically adds the surrounding characters.

  Sugestie: Clicking the *Insert/Edit Expression…* button when no selection is made in the textbox will append the new expression to the existing text. If you want to modify an existing expression, you need to first select the part of interest.

  Because maps are usually filled with some common textual information (date, author, title, page number, …), QGIS provides a direct access to the corresponding expressions or variables: press the *Dynamic text* button to select and insert them into your label.

  Sfat: The top menu *Add Item* ► *Add Dynamic Text* ► can be used to create a new label item filled with the selected predefined expression.

  It's possible to turn a dynamic label into static: press the drop-down arrow next to the *Insert/Edit Expression…* button and select *Convert to Static*. Any dynamic parts of the label's contents will be evaluated and replaced with their current values. You can then manually tweak the resulting text when needed.

- Labels can be interpreted as HTML code: check *Render as HTML*. You can now insert HTML tags or styles, URL, a clickable image that links to a web page, or something more complex…

The following code combines HTML rendering with expressions, for an advanced labeling and will output Fig. 22.25:

```
<html>
 <head>
   <style>
      /* Define some custom styles, with attribute-based size */
      name {color:red; font-size: [% ID %]px; font-family: Verdana; text-shadow:␣
↪grey 1px 0 10px;}
      use {color:blue;}
   </style>
 </head>

 <body>
   <!-- Information to display -->
   <u>Feature Information</u>
   <ul style="list-style-type:disc">
     <li>Feature Id: [% ID %]</li>
     <li>Airport: <name>[% NAME %]</name></li>
     <li>Main use: <use>[% USE %]</use></li>
   </ul>
   Last check: [% concat( format_date( "control_date", 'yyyy-MM-dd'), ' by <b><i>',
↪ @user_full_name, '</i></b>' ) %]

   <!-- Insert an image -->
   <p align=center><img src="path/to/logos/qgis-logo-made-with-color.svg" alt=
↪"QGIS icon" style="width:80px;height:50px;"</p>
 </body>
</html>
```

Fig. 22.25: Leveraging a label with HTML styling

Aspectul

- Define font and style of the text by clicking on the *Font* button. In the *Label Font* menu you can use some of the options for *Formatting the label text*.
- You can specify different horizontal and vertical margins in `mm`. This is the margin from the edge of the layout item. The label can be positioned outside the bounds of the label e.g. to align label items with other items. In this case you have to use negative values for the margin.
- Using the text alignment is another way to position your label. It can be:
  - *Left*, *Center*, *Right* or *Justify* for *Horizontal alignment*
  - and *Top*, *Middle*, *Bottom* for *Vertical alignment*.

Exploring expressions in a label item

Below some examples of expressions you can use to populate the label item with interesting information - remember that the code, or at least the calculated part, should be surrounded by `[%` and `%]` in the *Main properties* frame:

- Display a title with the current atlas feature value in „field1”:

```
'This is the map for ' || "field1"
```

  or, written in the *Main properties* section:

```
This is the map for [% "field1" %]
```

- Add a pagination for processed atlas features (eg, `Page 1/10`):

```
concat( 'Page ', @atlas_featurenumber, '/', @atlas_totalfeatures )
```

- Return the name of the airports of the current atlas region feature, based on their common attributes:

```
aggregate( layer := 'airports',
           aggregate := 'concatenate',
           expression := "NAME",
           filter := fk_regionId = attribute( @atlas_feature, 'ID' ),
           concatenator := ', '
         )
```

  Or, if an *attributes relation* is set:

```
relation_aggregate( relation := 'airports_in_region_relation',
                    aggregate := 'concatenate',
                    expression := "NAME",
                    concatenator := ', '
                  )
```

- Return the name of the airports contained in the current atlas region feature, based on their spatial relationship:

```
aggregate( layer := 'airports',
           aggregate := 'concatenate',
           expression := "NAME",
           filter := contains( geometry( @parent ), $geometry ),
           concatenator := ', '
         )
```

OR:

```
array_to_string( array:= overlay_contains( layer := 'airports',
                                           expression := "NAME" ),
                 delimiter:= ', '
               )
```

- Return the lower X coordinate of the `Map 1` item's extent:

```
x_min( map_get( item_variables( 'Map 1' ), 'map_extent' ) )
```

- Retrieve the name of the layers in the current layout `Map 1` item, and formats in one name by line:

```
array_to_string(
 array_foreach(
  map_get( item_variables( 'Map 1' ), 'map_layers' ), -- retrieve the layers␣
↪list
  layer_property( @element, 'name' ) -- retrieve each layer name
 ),
 '\n' -- converts the list to string separated by breaklines
)
```

- Display the list of layers with their license strings (usage rights) in a layout `Map 1` item. You need to fill the layers» *Access metadata* properties first.

```
array_to_string( map_credits( 'Map 1', true ) )
```

22.2.5 Elementul Legendă

The *Legend* item is a box or a table that explains the meanings of the symbols used on the map. A legend is then bound to a map item. You can add a legend item with the *Add Legend* tool following *items creation instructions* and manipulate it the same way as exposed in *Interacting with layout items*.

By default, the legend item displays all available layers and can be refined using its *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities (see Fig. 22.26):

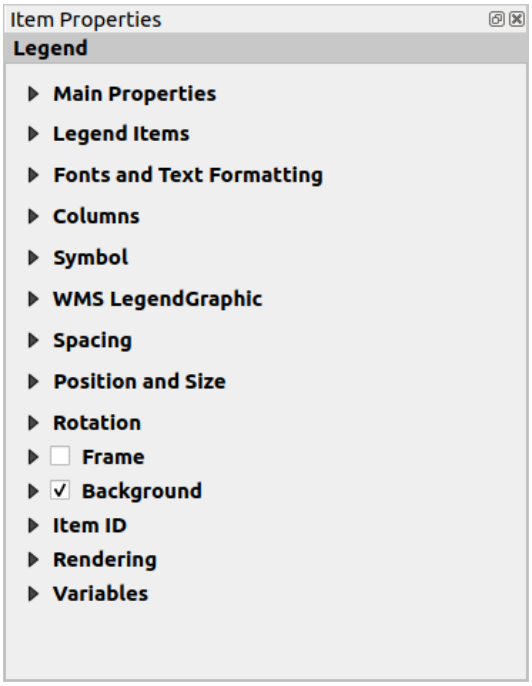


Fig. 22.26: Panoul Proprietăților Elementului Legendă

Proprietăți principale

The *Main properties* group of the legend *Item Properties* panel provides the following functionalities (see Fig. 22.27):

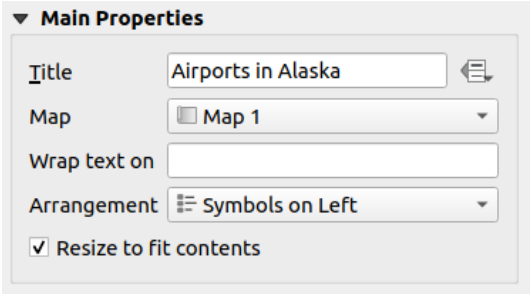


Fig. 22.27: Legend Main properties group

În proprietățile principale puteți:

- Change the *Title* of the legend. It can be made dynamic using the *data-defined override* setting, useful for example when generating an atlas;
- Choose which *Map* item the current legend will refer to. By default, the map over which the legend item is drawn is picked. If none, then it falls back to the *reference map*.

Notă: *Variables* of the linked map item (@map_id, @map_scale, @map_extent…) are also accessible from data-defined properties of the legend.

- Wrap the text of the legend on a given character: each time the character appears, it's replaced with a line break;
- Set the symbols and text placement in the legend: the *Arrangement* can be *Symbols on left* or *Symbols on right*. The default value depends on the locale in use (right-to-left based or not).

- Use *Resize to fit contents* to control whether or not a legend should be automatically resized to fit its contents. If unchecked, then the legend will never resize and instead just stick to whatever size the user has set. Any content which doesn't fit the size is cropped out.

Elementele legendei

The *Legend items* group of the legend *Item Properties* panel provides the following functionalities (see Fig. 22.28):

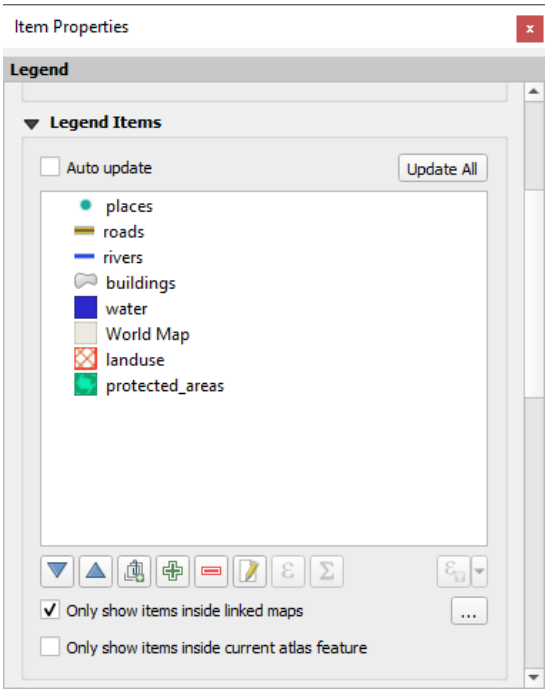


Fig. 22.28: Legend Items group

- The legend will be updated automatically if *Auto update* is checked. When *Auto update* is unchecked this will give you more control over the legend items. All the icons below the legend items list will be activated.
- The legend items window lists all legend items and allows you to change item order, group layers, remove and restore items in the list, edit layer names and symbology and add a filter.
  - The item order can be changed using the and buttons or with «drag-and-drop» functionality. The order can not be changed for WMS legend graphics.
  - Use the button to add a legend group.
  - Use the button to add layers and button to remove groups, layers or symbol classes.
  - The button is used to edit the layer, group name or title. First you need to select the legend item. Double-clicking the item also opens the text box to rename it.
  - The button uses expressions to customize each symbol label of the selected layer (see *Data-define the legend labels*)
  - The button adds a feature count for each class of vector layer.
  - The Filter legend by expression helps you filter which of the legend items of a layer will be displayed, i.e. using a layer that has different legend items (e.g., from a rule-based or categorized symbology), you can specify a boolean expression to remove from the legend tree, styles that have no feature satisfying a condition. Note that the features are nevertheless kept and shown in the layout map item.

While the default behavior of the legend item is to mimic the *Layers* panel tree, displaying the same groups, layers and classes of symbology, right-click any item offers you options to hide layer's name or raise it as a group or subgroup. In case you have made some changes to a layer, you can revert them by choosing *Reset to defaults* from the contextual menu of the legend entry.

After changing the symbology in the QGIS main window, you can click on *Update All* to adapt the changes in the legend element of the print layout.

- With the *Only show items inside linked maps*, only the legend items visible in the linked map will be listed in the legend. If you have more than one map you can click on … and select other maps from your layout. This tool remains available when *Auto-update* is active.
- While generating an atlas with polygon features, you can filter out legend items that lie outside the current atlas feature. To do that, check the *Only show items inside current atlas feature* option.

Data-define the legend labels

allows you to add *expressions* to each symbol label of a given layer. New variables (`@symbol_label`, `@symbol_id` and `@symbol_count`) help you interact with the legend entry.

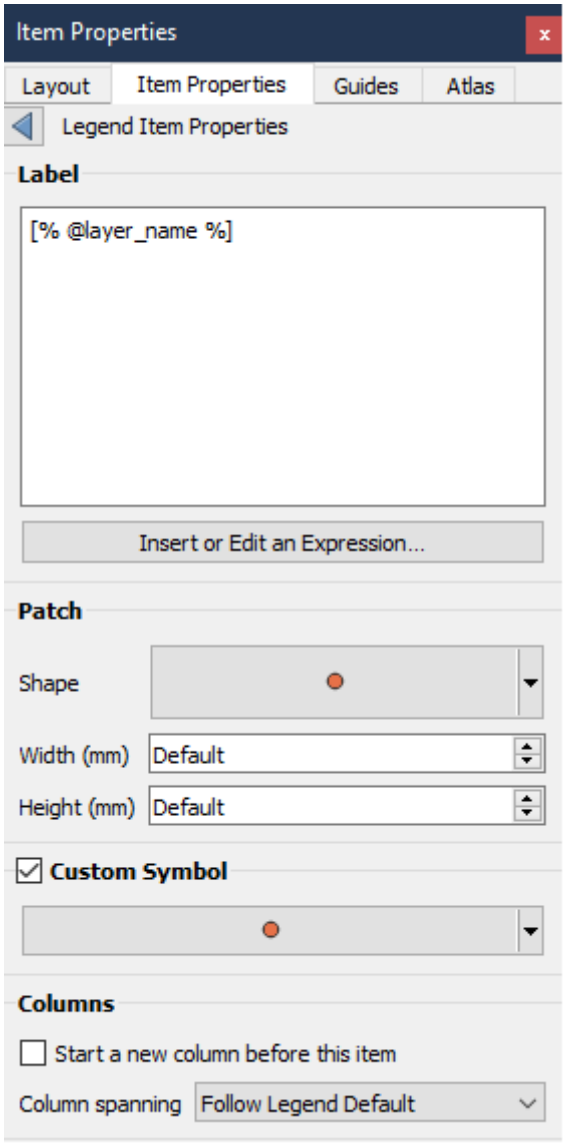
For example, given a `regions` layer categorized by its `type` field, you can append to each class in the legend their number of features and total area, e.g. `Borough (3) - 850ha`:

1. Select the layer entry in the legend tree
2. Press the button, opening the *Expression String Builder* dialog
3. Enter the following expression (*assuming symbol labels have not been edited*):

```
format( '%1 (%2) - %3ha',
        @symbol_label,
        @symbol_count,
        round( aggregate(@layer, 'sum', $area, filter:= "type"=@symbol_label)/
↪10000 )
      )
```

4. Apăsați pe *OK*

Customizing legend items



Legend items can also be customized individually in the *Legend Items Properties*. But these customization can only be done with *Auto update* disabled.

Double-clicking on an item or pressing Edit selected item properties allows for futher customization.

Etichetă

For all item types it allows to modify the label text by typing in or by inserting expressions using the *Insert or Edit an Expression*. Expressions can also be added directly anywhere in the item's label by using the [% expression %] notation.

Columns

The Legend Item Property also allows you to control the column splitting behaviour by forcing the column split to occur after a specific item or all symbols of a layer. Automatic splitting of a layer and its child can also be allowed or blocked on a layer-basis in this widget.

Patch

For items with a symbol, the Legend Item Property allows you to specify the maximum height and width that a symbol can occupy.

For vector symbols, you can specify a custom shape for the symbol. The shapes are usually defined by an expression to represent the geometry in a simple plane, but those symbols can also be saved in the style manager and imported later. The default symbol for each geometry type can also be controlled via the style manager.

Custom Symbol

A custom symbol can also be specified for vector symbols. This can be useful to tweak the render of a specific symbol, to enhance it in the legend or have a symbol independent from its true symbol preview. This custom symbol will override the legend symbol, but will take into account the symbol *Patch* specified.

Fonts and text formatting

The *Fonts and text formatting* group of the legend *Item Properties* panel provides the following functionalities:

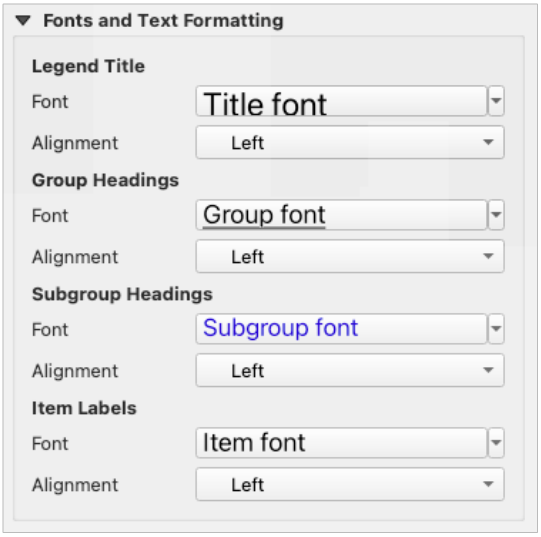


Fig. 22.29: Legend Fonts properties

- You can change the font of the legend title, group, subgroup and item (feature) in the legend item using the *font selector* widget, providing all the capabilities of *text formatting* (font spacing control, mixed HTML formatting, coloring, blending, background, text buffer, shadow, …)
- For each of these levels you can set the text *Alignment*: it can be *Left* (default for left-to-right based locales), *Center* or *Right* (default for right-to-left based locales).

Columns

Under the *Columns* group of the legend *Item Properties* panel, legend items can be arranged over several columns:

- Set the number of columns in the *Count* 1,00 field. This value can be made dynamic e.g., following atlas features, legend contents, the frame size…
- *Equal column widths* stabilește modul în care ar trebui să fie ajustate coloanele legendei.
- The *Split layers* option allows a categorized or a graduated layer legend to be divided between columns.

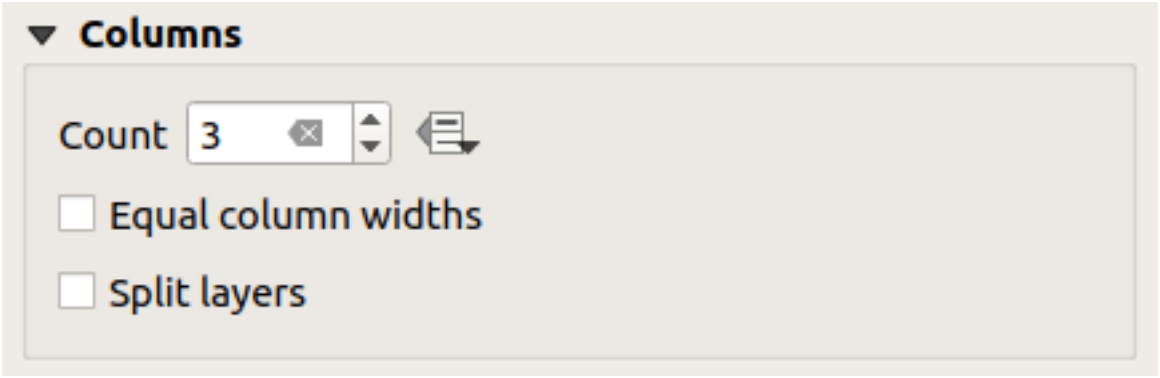


Fig. 22.30: Legend Columns settings

Symbol

The *Symbol* group of the legend *Item Properties* panel configures the size of symbols displayed next to the legend labels. You can:

- Set the *Symbol width* and *Symbol height*
- Set the markers» *Min symbol size* and *Max symbol size*: `0.00mm` means there is no value set.
- *Draw stroke for raster symbols*: this adds an outline to the symbol representing the band color of the raster layer; you can set both the *Stroke color* and *Tickness*.

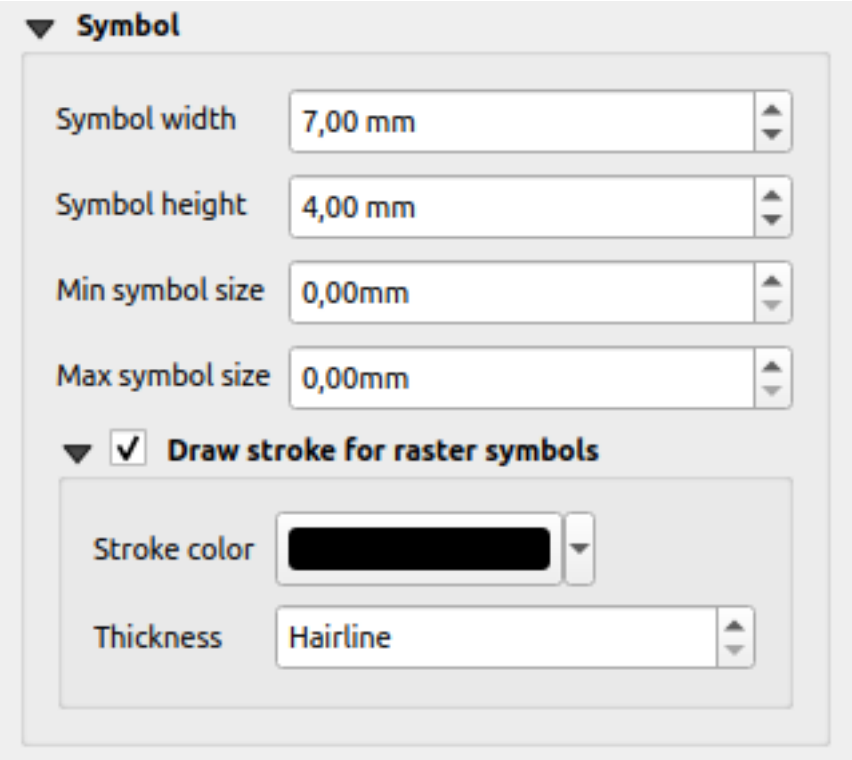


Fig. 22.31: Legend Symbol configuration

WMS LegendGraphic

The *WMS LegendGraphic* section of the legend *Item Properties* panel provide the following functionalities (see Fig. 22.32):

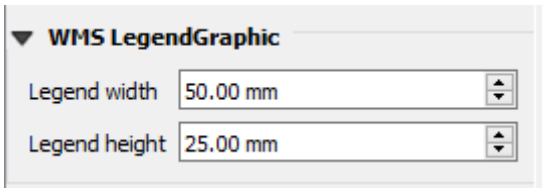
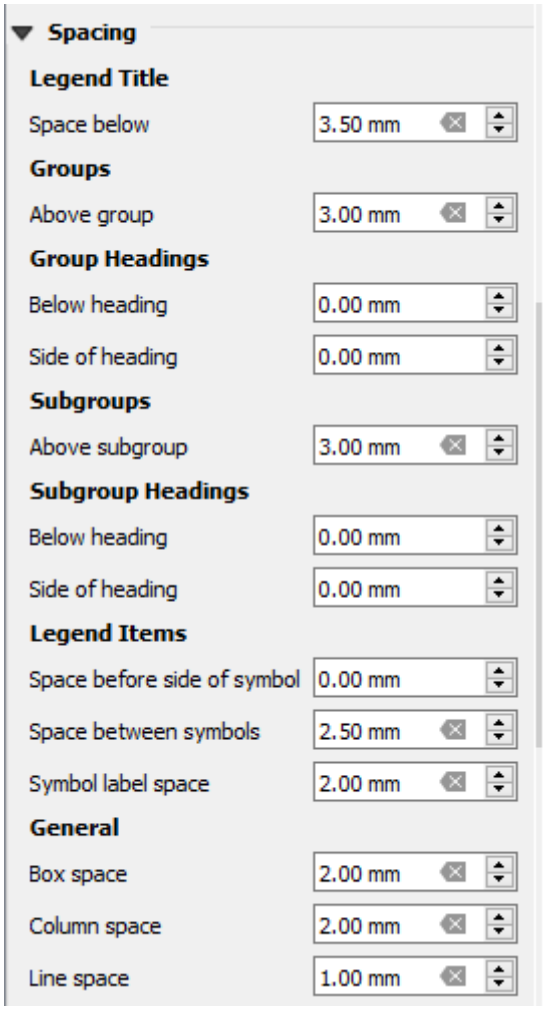


Fig. 22.32: WMS LegendGraphic

When you have added a WMS layer and you insert a legend item, a request will be sent to the WMS server to provide a WMS legend. This Legend will only be shown if the WMS server provides the GetLegendGraphic capability. The WMS legend content will be provided as a raster image.

WMS LegendGraphic is used to be able to adjust the *Legend width* and the *Legend height* of the WMS legend raster image.

Spacing



The *Spacing* section allows you to customize the spacing within the legend. Spacing can greatly help denote the groupement of items in the legend and their relation.

Spacing around and before title, groups, subgroups, symbols, labels, boxes, columns and lines can be customized through this dialog.

22.2.6 Elementul Scării Grafice

Scale bars provide a visual indication of the size of features, and distance between features, on the map item. A scale bar item requires a map item. Use the *Add Scale Bar* tool following *items creation instructions* to add a new scale bar item that you can later manipulate the same way as exposed in *Interacting with layout items*.

By default, a new scale bar item shows the scale of the map item over which it is drawn. If there is no map item below, the *reference map* is used. You can customize it in the *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities (see Fig. 22.33):

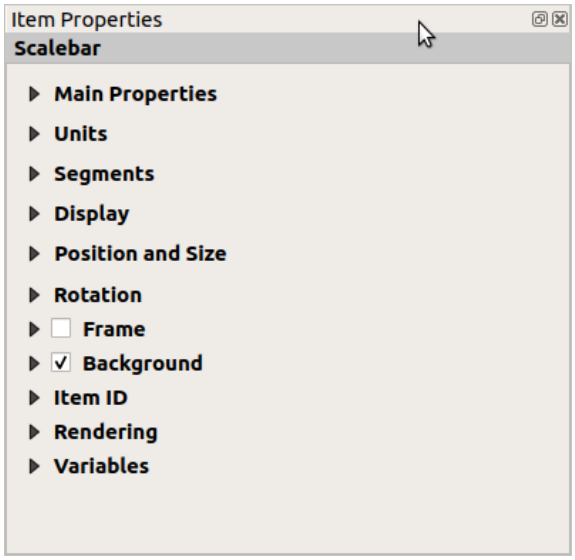


Fig. 22.33: Panoul Proprietăților pentru Elementul de tip Scară Grafică

Proprietăți principale

The *Main properties* group of the scale bar *Item Properties* panel provides the following functionalities (see Fig. 22.34):

Fig. 22.34: Scale Bar Main properties group

1. First, choose the map the scale bar will be attached to
2. Then, choose the style of the scale bar. Available styles are:
   - **Single box** and **Double box** styles, which contain one or two lines of boxes alternating colors;
   - **Middle**, **Up** or **Down** line ticks;
   - **Stepped line** style that draws a stepped line representation of a scalebar

- **Hollow** style that draws a single box with alternating color for the segments, with horizontal lines through alternating segments
- **Numeric**, where the scale ratio is printed (e.g., `1:50000`).

3. Set properties as appropriate

Units

The *Units* group of the scale bar *Item Properties* panel provides the functionalities to set the units of display and some text formatting (see Fig. 22.35):

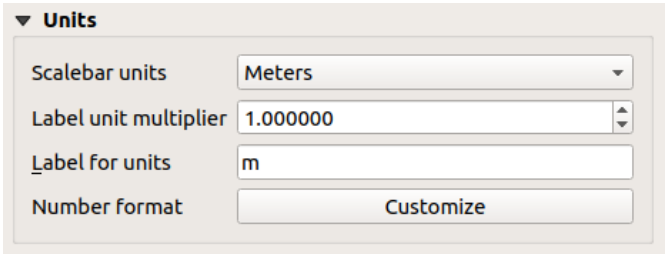


Fig. 22.35: Scale Bar Units group

- Select the units you want to use with *Scalebar units*. There are many possible choices: **Map Units** (the default one), **Meters**, **Feet**, **Miles** or **Nautical Miles**… and some derivatives. Units conversion is handled automatically.
- The *Label unit multiplier* specifies how many scale bar units per labeled unit. Eg, if your scale bar units are set to „meters", a multiplier of 1000 will result in the scale bar labels in „kilometers".
- The *Label for units* field defines the text used to describe the units of the scale bar, eg `m` or `km`. This should be matched to reflect the multiplier above.
- Press *Customize* next to *Number format* to have control over all the formatting properties for the numbers in the scale bar, including thousand separators, decimal places, scientific notation, etc. (see *Number Formatting* for more details). Very useful in the case of making maps for audiences outside of the current QGIS locale, or when you would like to vary the style from the locale defaults (e.g. adding thousands separators when the locale default is to hide them).

Segments

The *Segments* group of the scale bar *Item Properties* panel provides the functionalities to configure the number and size of segments and subdivisions (see Fig. 22.36):

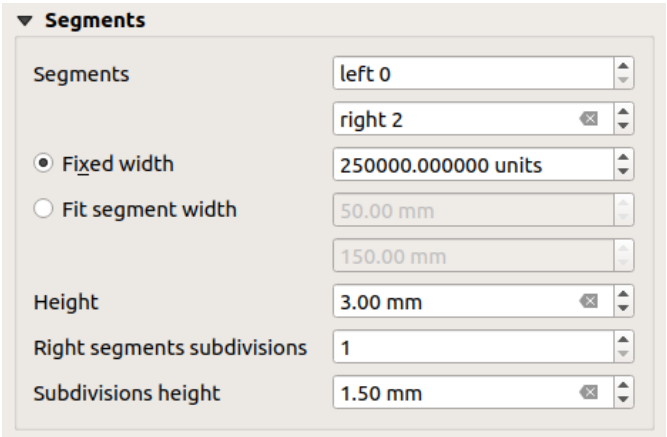


Fig. 22.36: Scale Bar Segments group

- You can define the number of *Segments* that will be drawn at the left and right sides of the `0` of the scale bar:
  - number of subdivisions of a unique segment on the *Left* side
  - number of segments on the *Right* side
- You can set the width of a segment or a range for the total length of the scale bar:
  - Set how long a segment will be in your scale bar units (*Fixed width*)
  - Or limit the total scale bar size in `mm` with *Fit segment width* option. In the latter case, each time the map scale changes, the scale bar is resized (and its label updated) to fit between the upper and lower range set.
- *Height* is used to define the height of the bar.
- *Right segment subdivisions* is used to define the number of sections the right-side segments of the scale bar can have (for *Line Ticks Down*, *Line Ticks Middle* and *Line Ticks Up* scale bar styles) .
- *Subdivision height* is used to define the height of the subdivision segment.

Afișarea

The *Display* group of the scale bar *Item Properties* panel provides the following functionalities:

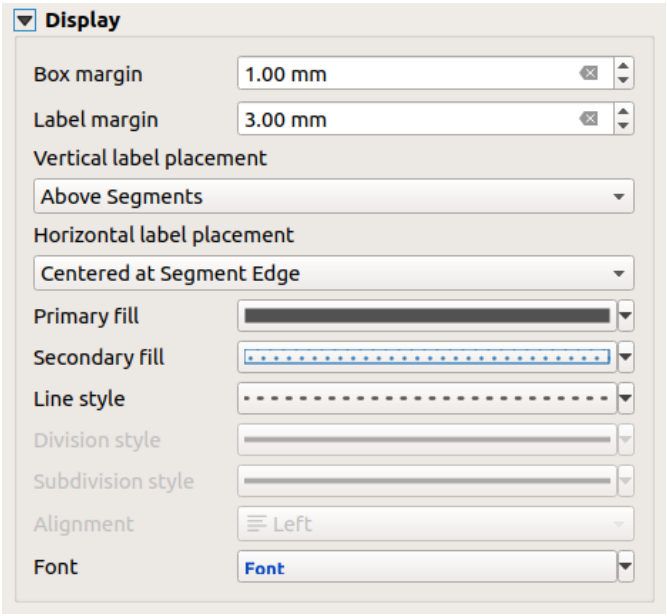


Fig. 22.37: Scale Bar Display group

Puteți defini modul în care va fi afișată scara grafică în cadrul său.

- *Box margin* : spațiul dintre bordurile textului și ale cadrului
- *Label margin* : space between text and scale bar drawing
- *Vertical label placement*: it can be above or below the scale bar segment
- *Horizontal label placement*: which would be centered at the scale bar segment's edge or center
- *Primary fill* and *Secondary fill* of the scale bar drawing using *fill symbols properties* (color, opacity, patterns, effects…) — for *Single Box*, *Double Box* and *Hollow* styles
- *Line style* of the scale bar drawing using *line symbols properties* (color, stroke, join, cap style, patterns, effects…) — for all but *Numeric* style

- *Division style* and *Subdivision style* respectively for division and subdivision segments in *Line Ticks Up*, *Line Ticks Middle* and *Line Ticks Down* scale bar styles using *line symbols properties* (color, stroke, join, cap style, patterns, effects…)
- *Alignment* puts text on the left, center or right side of the frame (only for *Numeric* scale bar style)
- *Font* to set the *properties* (size, font, color, letter spacing, shadow, background…) of the scale bar label.

Since most of the display properties of the scale bar rely on symbols whose properties can be data-defined, it's possible to render data-defined scale bars.

Example: The following code applied to the bold property of the scale labels will display numbers in bold when they are a multiple of 500:

```
-- returns True (or 1) if the value displayed on the bar
-- is a multiple of 500

@scale_value % 500 = 0
```

22.2.7 The Table Items

You can use table items to decorate and explain your map:

- *Attribute table*: shows a subset of the attributes of a layer, based on predefined rules
- *Fixed table*: inserts a manual text table where information can be independent from the layers.

The attribute table item

Any layer in the project can have its attributes shown in the print layout. Use the *Add Attribute Table* tool following *items creation instructions* to add a new table item that you can later manipulate the same way as exposed in *Interacting with layout items*.

By default, a new attribute table item loads first rows of the first (alphabetically sorted) layer, with all the fields. You can however customize the table thanks to its *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities (see Fig. 22.38):

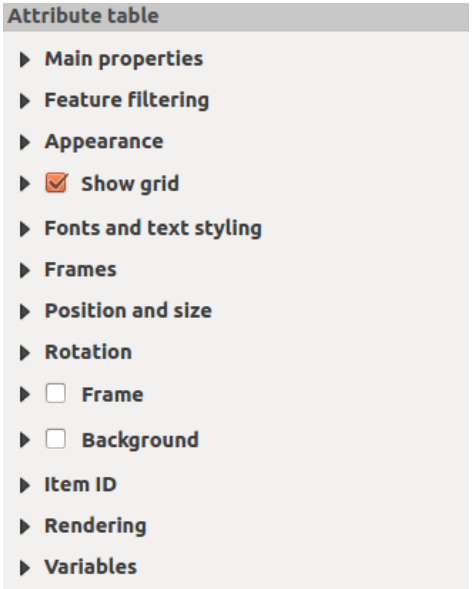


Fig. 22.38: Panoul Proprietăților pentru Elementul de tip Tabelă de Atribute

Proprietăți principale

The *Main properties* group of the attribute table provides the following functionalities (see Fig. 22.39):

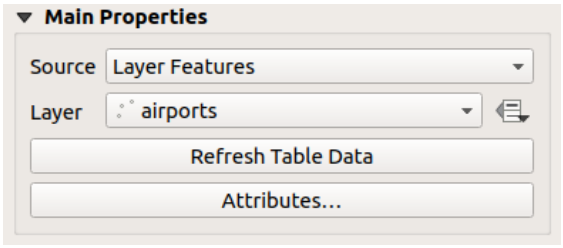


Fig. 22.39: Attribute table Main properties Group

- For *Source* you can by default only select **Layer features** allowing you to select a *Layer* from the vector layers loaded in the project.

  The Data-defined override button near the layer list allows you to dynamically change the layer which is used to populate the table, e.g. you could fill the attribute table with different layer attributes per atlas page. Note that the table structure used (Fig. 22.42) is the one of the layer shown in the *Layer* drop-down list and it is left intact, meaning that setting a data defined table to a layer with different field(s) will result in empty column(s) in the table.

  In case you activate the *Generate an atlas* option in the *Atlas* panel (see *Generarea unui Atlas*), there are two additional *Source* possible:

  - **Current atlas feature** (see Fig. 22.40): you won't see any option to choose the layer, and the table item will only show a row with the attributes from the current feature of the atlas coverage layer.
  - and **Relation children** (see Fig. 22.41): an option with the relation names will show up. This feature can only be used if you have defined a *relation* using your atlas coverage layer as parent, and the table will show the children rows of the atlas coverage layer's current feature.
- The button *Refresh Table Data* can be used to refresh the table when the actual contents of the table has changed.

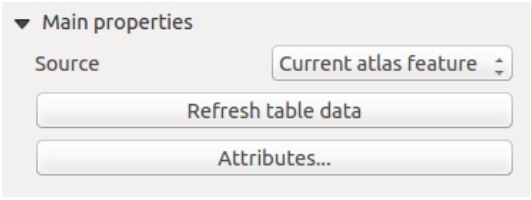


Fig. 22.40: Attribute table Main properties for «Current atlas feature»

Fig. 22.41: Attribute table Main properties for «Relation children»

- The button *Attributes…* starts the *Select Attributes* dialog, (see Fig. 22.42) that can be used to change the visible contents of the table. The upper part of the window shows the list of the attributes to display and the lower part helps you sort the data.

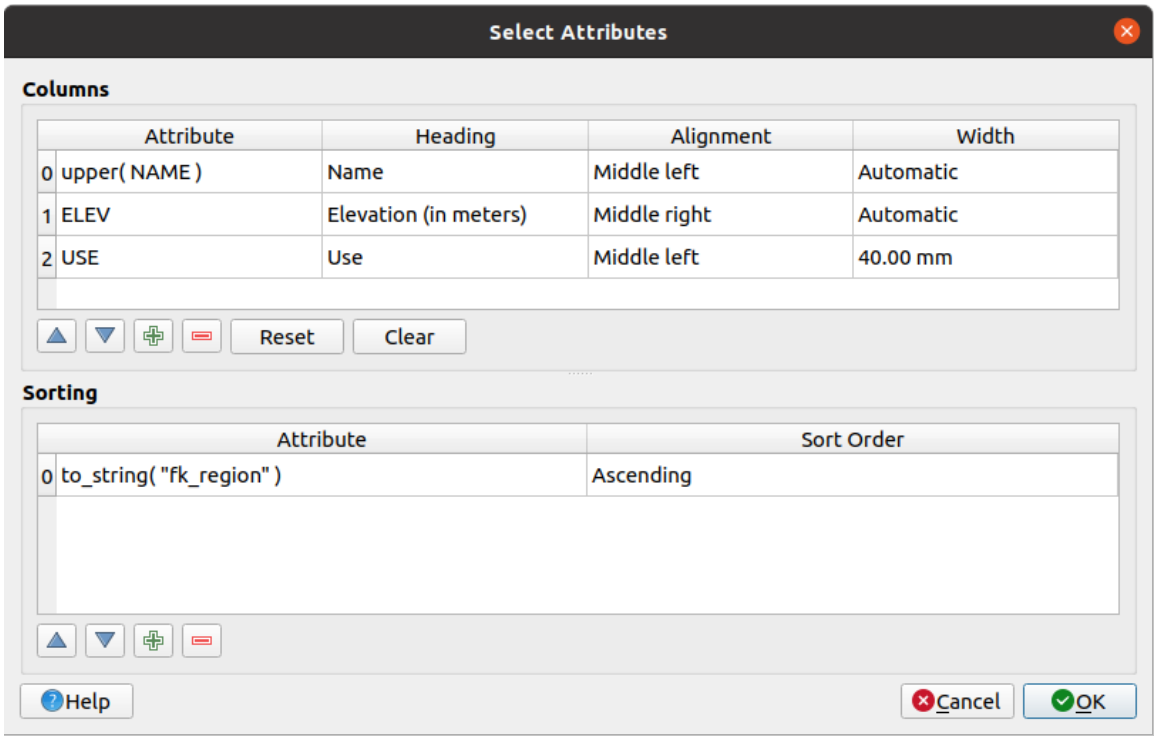


Fig. 22.42: Attribute table Select attributes Dialog

În secțiunea *Coloanelor* puteți:

- Move attributes up or down the list by selecting the rows and then using the and buttons to shift the rows. Multiple rows can be selected and moved at any one time.
- Add an attribute with the button. This will add an empty row at the bottom of the table where you can select a field to be the attribute value or create an attribute via a regular expression.
- Remove an attribute with the button. Multiple rows can be selected and removed at any one time.
- Reset the attribute table back to its default state with the *Reset* button.
- Clear the table using the *Clear* button. This is useful when you have a large table but only want to show a small number of attributes. Instead of manually removing each row, it may be quicker to clear the table and add the rows needed.
- Cell headings can be altered by adding the custom text in the *Heading* column.
- Cell alignment can be managed with the *Alignment* column which will dictate the texts position within the table cell.
- Cell width can be manually managed by adding custom values to the *width* column.

În secțiunea de *Sortare* puteți:

- Add an attribute to sort the table with: press the button and a new empty row is added. Insert a field or an expression in the *Attribute* column and set the *Sort order* to **Ascending** or **Descending**.
- Select a row in the list and use the and buttons to change the sort priority on attribute level. Selecting a cell in the *Sort Order* column helps you change the sorting order of the attribute field.
- Use the button to remove an attribute from the sorting list.

Filtrarea entităților

The *Feature filtering* group of the attribute table provides the following functionalities (see Fig. 22.43):

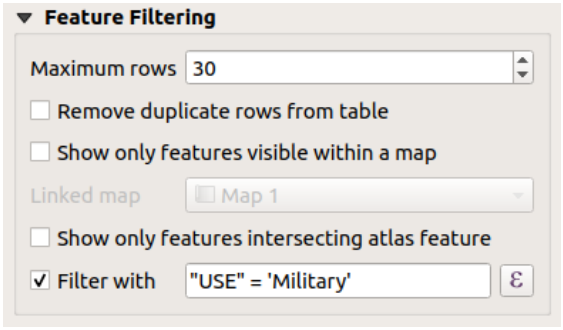


Fig. 22.43: Attribute table Feature filtering Group

Puteți:

- Defini *Numărul maxim de rânduri* care vor fi afișate.
- Activate *Remove duplicate rows from table* to show unique records only.
- Activate *Show only visible features within a map* and select the corresponding *Linked map* whose visible features attributes will be displayed.
- Activate *Show only features intersecting Atlas feature* is only available when *Generate an atlas* is activated. When activated it will show a table with only the features which intersect the current atlas feature.
- Activate *Filter with* and provide a filter by typing in the input line or insert a regular expression using the given expression button. A few examples of filtering statements you can use when you have loaded the airports layer from the Sample dataset:
  - `ELEV > 500`
  - `NAME = 'ANIAK'`
  - `NAME NOT LIKE 'AN%'`
  - `regexp_match( attribute( $currentfeature, 'USE' ) , '[i]')`

  The last regular expression will include only the airports that have a letter «i» in the attribute field «USE».

Aspectul

The *Appearance* group of the attribute table provides the following functionalities (see Fig. 22.44):

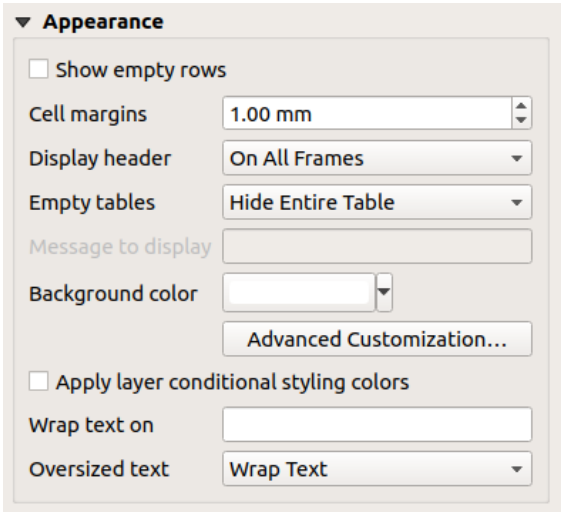


Fig. 22.44: Attribute table appearance Group

- Click *Show empty rows* to fill the attribute table with empty cells. This option can also be used to provide additional empty cells when you have a result to show!
- With *Cell margins* you can define the margin around text in each cell of the table.
- With *Display header* you can select from a list one of «On first frame», «On all frames» default option, or «No header».
- The option *Empty table* controls what will be displayed when the result selection is empty.
  - **Draw headers only**, will only draw the header except if you have chosen «No header» for *Display header*.
  - **Hide entire table**, will only draw the background of the table. You can activate *Don't draw background if frame is empty* in *Frames* to completely hide the table.
  - **Show set message**, will draw the header and adds a cell spanning all columns and display a message like «No result» that can be provided in the option *Message to display*
- The option *Message to display* is only activated when you have selected **Show set message** for *Empty table*. The message provided will be shown in the table in the first row, when the result is an empty table.
- With *Background color* you can set the background color of the table using the *color selector* widget. The *Advanced customization* option helps you define different background colors for each cell (see Fig. 22.45)

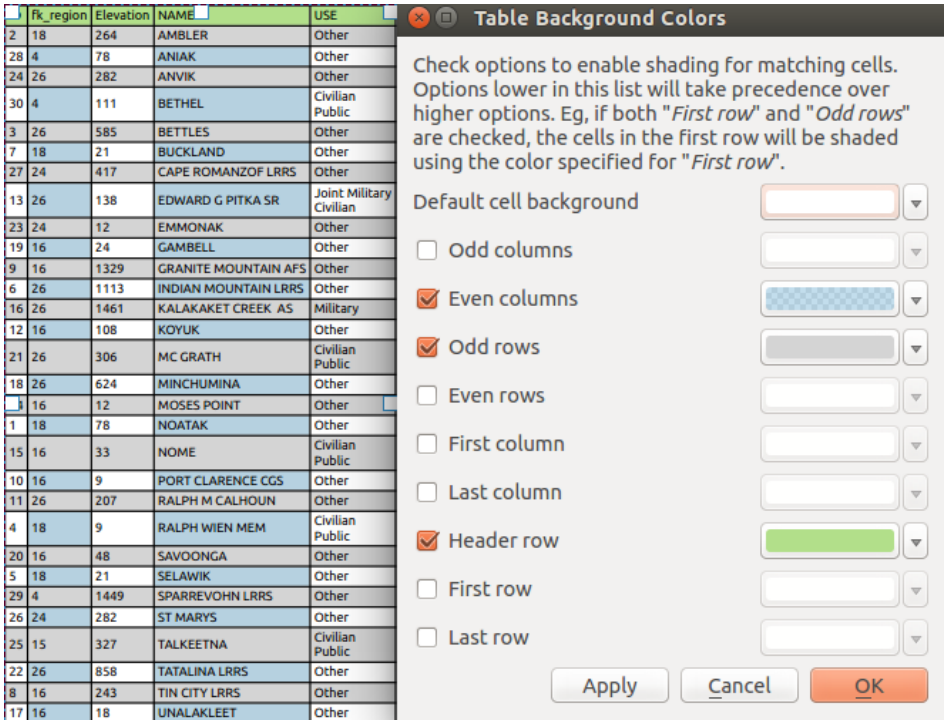


Fig. 22.45: Attribute table Advanced Background Dialog

- *Apply layer conditional styling colors*: the *conditional table formatting* present in the layer is applied inside the layout attribute table (background color, font family and properties such as bold, italic, strikeout, underline, color, …). Conditional formatting rules take precedence over other layout table formatting settings, e.g. they will override other cell background color settings such as alternating row colors.
- With the *Wrap text on* option, you can define a character on which the cell content will be wraped each time it is met
- With *Oversized text* you define the behavior when the width set for a column is smaller than its content's length. It can be **Wrap text** or **Truncate text**.

Notă: More properties of the attribute table item are described in the *Tables common functionalities* section.

The fixed table item

Additional information about the map can be inserted manually into a table by choosing *Add Fixed Table* and by following *items creation instructions* to add a new table item that you can later manipulate the same way as exposed in *Interacting with layout items*.

By default, an empty table with two minimized columns and rows appears in the map layout. You have to customize the table in the *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities:

Proprietăți principale

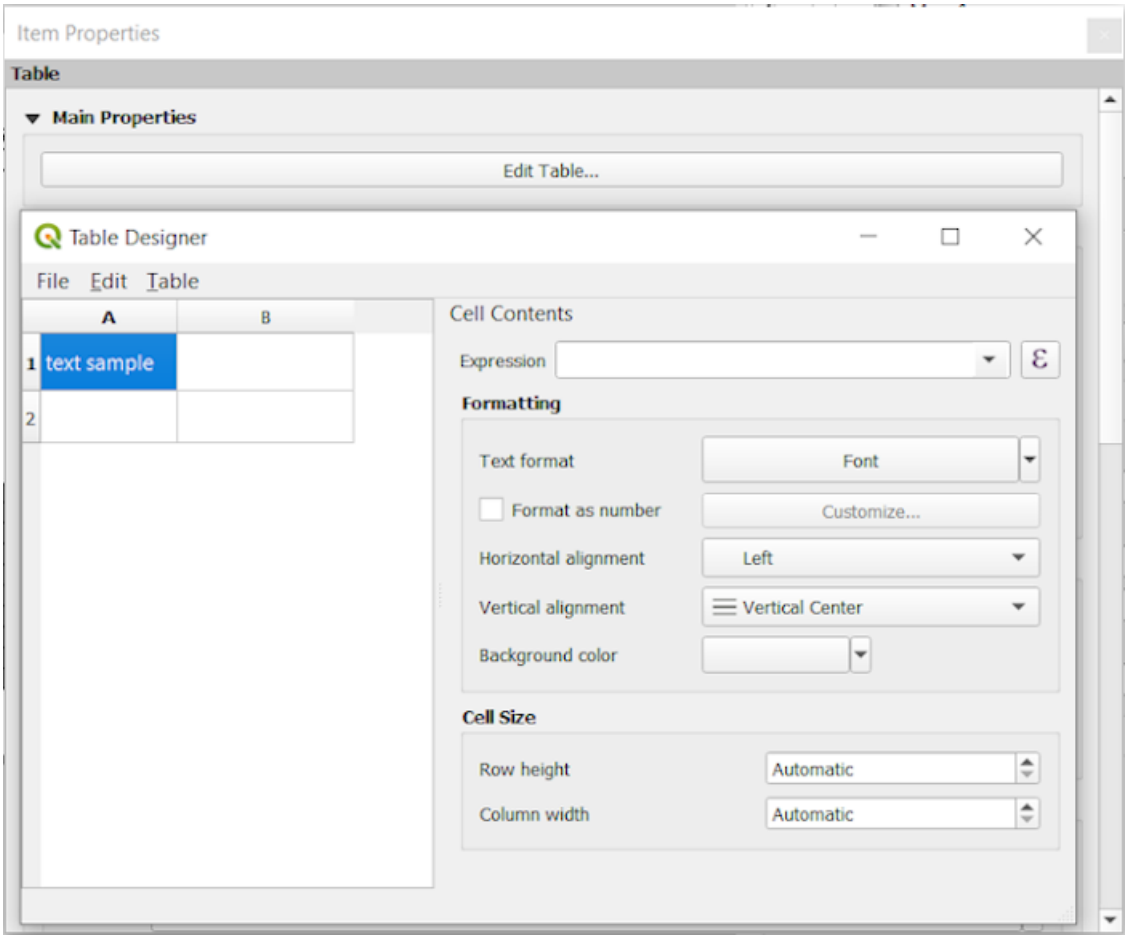


Fig. 22.46: Fixed table Item Properties Panel with Table designer

In *Main properties* you can work with the *Table designer* when clicking the *Edit table …*:

- You can click into the table and insert texts manually.
- Through the menus on top it is possible to:
  - *Import Content From Clipboard* by going to *File* (it overrides given inputs).
  - work with selection functionalities for rows and columns by going to *Edit*.
  - *Insert rows*, *Insert columns*, *Delete Rows*, *Delete Columns* as well as using the option to *Include Header Row*.
- You can work with the *Cell Contents* section on the right and:
  - Define the text format of selected cells in *Formatting*
    - by clicking on the given expression button and using a regular expression for the input of the cell
    - by choosing the *Text format*
    - by *Format as number* (several formats are available)
    - by defining the *Horizontal alignment* and the *Vertical alignment*
    - by choosing a *Background color*

– Define the *Cell Size* with *Row height* and *Column width*.

Aspectul

The *Appearance* group of the fixed table provides the following functionalities:

- Click *Show empty rows* to fill the attribute table with empty cells.
- With *Cell margins* you can define the margin around text in each cell of the table.
- With *Display header* you can select from a list one of «On first frame», «On all frames» default option, or «No header».
- With *Background color* you can set the background color of the table using the *color selector* widget. The *Advanced customization* option helps you define different background colors for each cell.
- With *Oversized text* you define the behavior when the width set for a column is smaller than its content's length. It can be **Wrap text** or **Truncate text**.

Notă: More properties of the fixed table item are described in the *Tables common functionalities* section.

Tables common functionalities

Afișarea grilei

The *Show grid* group of the table items provides the following functionalities (see Fig. 22.47):

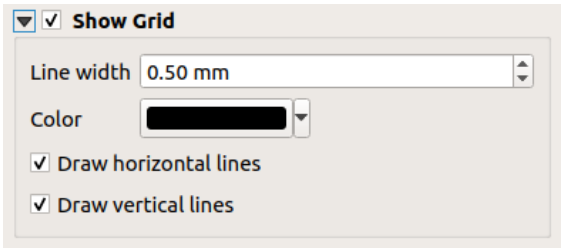


Fig. 22.47: Attribute table Show grid Group

- Activate *Show grid* when you want to display the grid, the outlines of the table cells. You can also select to either *Draw horizontal lines* or *Draw vertical lines* or both.
- With *Line width* you can set the thickness of the lines used in the grid.
- The *Color* of the grid can be set using the color selection widget.

Stilizarea fonturilor și a textelor

The *Fonts and text styling* group of the table items provides the following functionalities (see Fig. 22.48):

Fig. 22.48: Attribute table Fonts and text styling Group

- You can define *Font* properties for *Table heading* and *Table contents*, using the advanced *text settings* widget (with buffer, shadow, paint effects, transparence, background, coloring, …). Note that these changes do not affect the cells that have custom font assigned, either from the *Appearance* section or the *Table Designer* dialog. Only cells with the default rendering are overwritten.
- For *Table heading* you can additionally set the *Alignment* to `Follow column alignment` or override this setting by choosing `Left`, `Center` or `Right`. The column alignment is set using the *Select Attributes* dialog (see Fig. 22.42).

Cadrele

The *Frames* group of the table item properties provides the following functionalities (see Fig. 22.49):

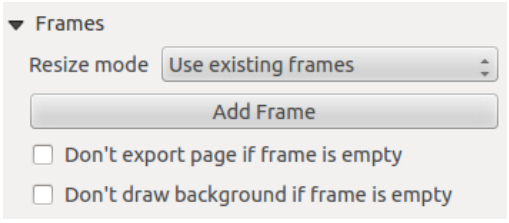


Fig. 22.49: Attribute table Frames Group

- With *Resize mode* you can select how to render the attribute table contents:
  - `Use existing frames` displays the result in the first frame and added frames only.
  - `Extend to next page` will create as many frames (and corresponding pages) as necessary to display the full selection of attribute table. Each frame can be moved around on the layout. If you resize a frame, the resulting table will be divided up between the other frames. The last frame will be trimmed to fit the table.
  - `Repeat until finished` will also create as many frames as the *Extend to next page* option, except all frames will have the same size.
- Use the *Add Frame* button to add another frame with the same size as selected frame. The result of the table that will not fit in the first frame will continue in the next frame when you use the Resize mode `Use existing frames`.
- Activate *Don't export page if frame is empty* prevents the page to be exported when the table frame has no contents. This means all other layout items, maps, scalebars, legends etc. will not be visible in the result.
- Activate *Don't draw background if frame is empty* prevents the background to be drawn when the table frame has no contents.

22.2.8 The Marker, Picture and North Arrow Items

- *The Picture Item*
  - *Proprietăți principale*
  - *Size and placement*
  - *Image rotation*
- *The North Arrow Item*
- *The Marker Item*

Along with the map or legend items in the print layout, you may want to decorate your realization with images or annotations. QGIS provides different tools to achieve this:

- the *picture item*: decorates the layout with an image raster or SVG file (e.g. logos, pictures, north arrows, …)
- the *north arrow item*: a picture item predefined with a north arrow image
- the *marker item*: decorates the layout with QGIS vector *symbols*. It can be used to place markers over a map item or for creation of advanced custom legends.

The Picture Item

You can add a picture by dragging it from your file manager onto the canvas, pasting it directly into the layout by using `Ctrl+V` or *Edit* ► *Paste* and by using the Add Picture, following *items creation instructions*. Then you can manipulate it, as explained in *Interacting with layout items*.

When using Add Picture, the picture item will be a blank frame that you can customize using its *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities:

Proprietăți principale

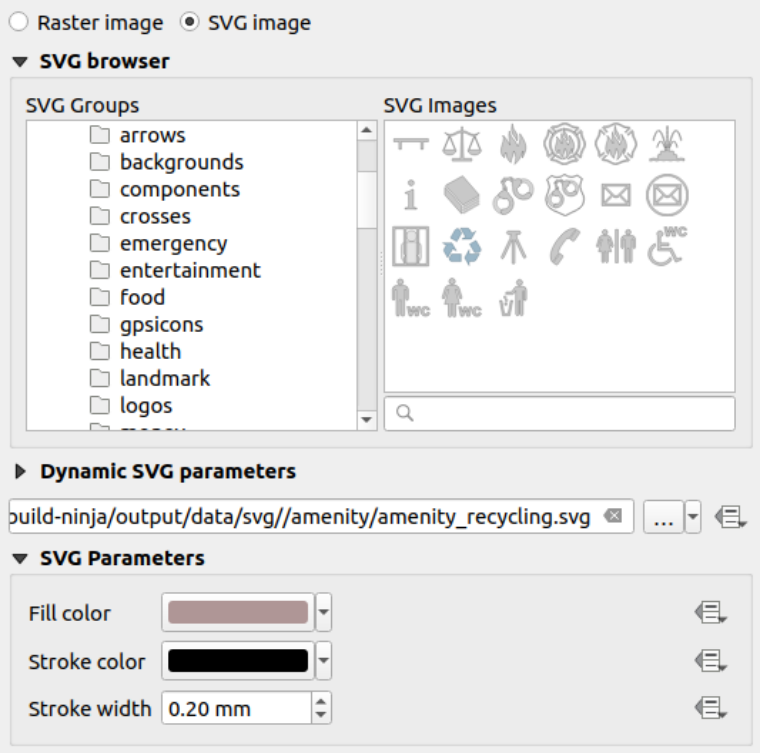


Fig. 22.50: Picture Item Properties panel

The picture item supports two types of images:

- *Raster Image*: a file selector widget can be used to fetch the data. Use the … Browse button to select a file on your computer or enter the path directly in the text field. You can even provide a remote URL that points to a picture. The associated image can also be *embedded* in the layout.

  Use the data defined override button to set the image source from a feature attribute or using a regular expression.

- *SVG Image*: using by default the SVG libraries provided in *Settings* ► *Options* ► *System* ► *SVG Paths*. You can however use any other file, and the file selection follows the same rules as for the raster image. The SVG parameters can as well be set dynamic.

  The QGIS provided (default) `.SVG` files are customizable, meaning that you can easily apply other *Fill color*, *Stroke color* (including opacity) and *Stroke width* than the original, using their corresponding feature in the *SVG Parameters* group. These properties can also be *data-defined*.

  If you add an `.SVG` file that does not enable these properties, you may need to add the following tags to the file in order to add support e.g. for transparency:

  – *fill-opacity="param(fill-opacity)"*

  – *stroke-opacity="param(outline-opacity)"*

  More details at *SVG parametrizabil*.

Notă: Drag-and-drop an image file (raster or SVG) into the layout page will create a layout picture item with corresponding settings.

Size and placement

Fig. 22.51: Layout pictures size and placement properties

With the *Resize mode* option, you can set how the image is displayed when the frame is resized:

- `Zoom`: enlarges/reduces the image to the frame while maintaining the aspect ratio of picture
- `Stretch`: stretches the image to fit inside the frame
- `Clip`: use this mode for raster images only, it sets the size of the image to the original image size without scaling, and the frame is used to clip the image. So only the part of the image that is inside the frame will be visible.
- `Zoom and resize frame`: enlarges the image to fit the frame, and then resizes frame to fit the resulting image dimensions
- `Resize frame to image size`: sets the size of the frame to match the original size of the image (no scaling)

Depending on the selected *Resize mode*, the *Placement* and *Image rotation* options may be disabled. *Placement* lets you select the position of the image inside its frame (top/middle/bottom and left/center/right).

Image rotation

Images can be rotated with the *Image rotation* field. Activating the *Sync with map* checkbox synchronizes the rotation of the image with the rotation applied to the selected map item. This is a convenient feature to make any picture behave as a north arrow. The *North alignment* can be:

- **Grid north**: the direction of a grid line which is parallel to the central meridian of the national/local grid
- **True north**: direction of a meridian of longitude.

You can also apply a declination *Offset* to the picture rotation.

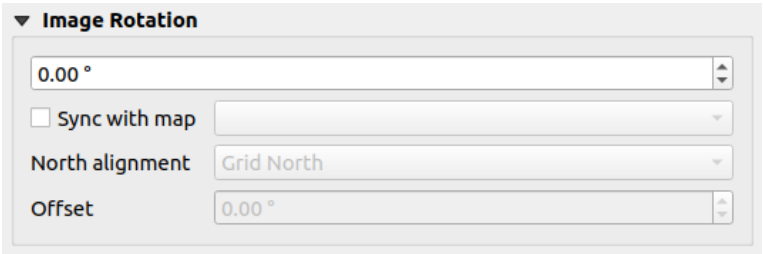


Fig. 22.52: Layout pictures image rotation properties

The North Arrow Item

You can add a north arrow with the Add North Arrow button, following *items creation instructions* and manipulate it the same way as exposed in *Interacting with layout items*.

Since north arrows are images, the *North Arrow* item has the same properties as the *picture item*. The main differences are:

- A default north arrow is used when adding the item, instead of a blank frame
- The north arrow item is synced with a map item by default: the *Sync with map* property is the map over which the north arrow item is drawn. If none, it falls back to the *reference map*.

Notă: Many of the north arrows do not have an «N» added in the north arrow. This is done on purpose, since there are languages that do not use an «N» for North.

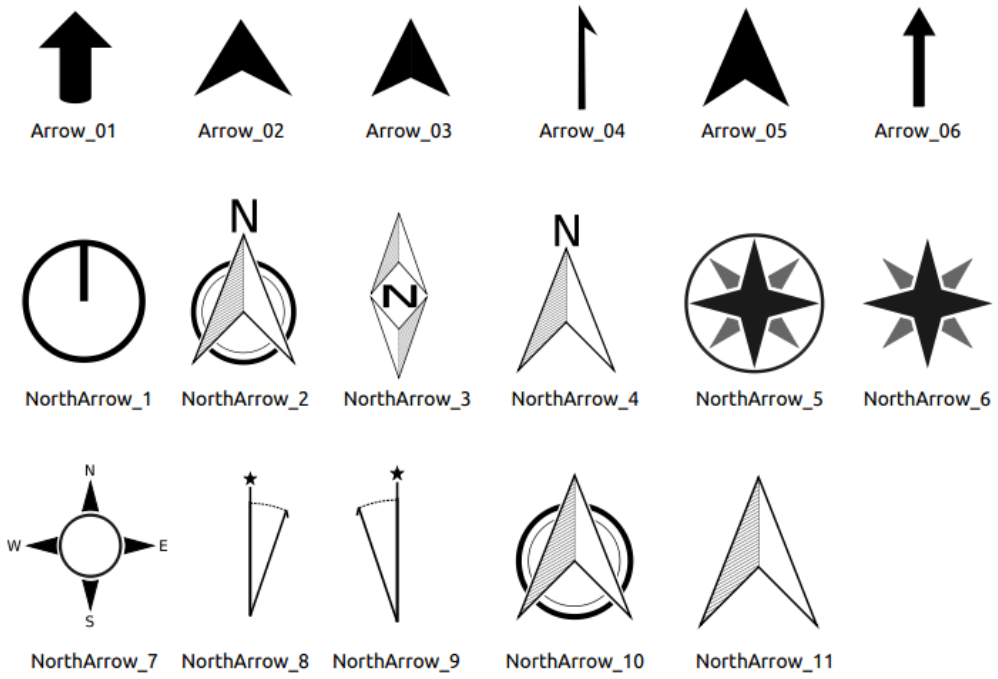


Fig. 22.53: Săgețile Nordului disponibile pentru selecție în Biblioteca SVG

The Marker Item

To add a marker item, select the Add Marker button, and click on the page. A default point marker symbol is added. Then you can manipulate it, as explained in *Interacting with layout items*. But note that unlike most of the other items, you resize the item given that its size is controlled by the embedded symbols properties.

The marker item can be customized from the *Item Properties* panel. Other than the *items common properties*, you can also:

- modify the *Symbol*, relying on all the symbol *widget capabilities*
- sync the marker item rotation with the map's (see *Image rotation*), acting as a north arrow. The map rotation is added to any existing marker symbol level rotation (so .e.g if you have to rotate the triangle marker 90° to get it pointing straight up, it will still work nicely in north arrow mode!)

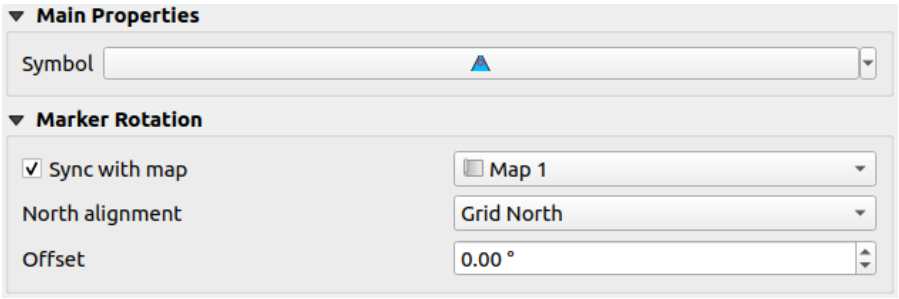


Fig. 22.54: The marker item custom properties

22.2.9 The Elevation Profile Item

The Elevation Profile item is used to display an *elevation profile view* in a layout. Use the *Add Elevation Profile* button, and follow *items creation instructions* to add a new elevation profile item that you can later manipulate the same way as demonstrated in *Interacting with layout items*.

A new elevation profile item has default settings that make it render an empty chart. You can customize its properties in the *Item Properties* panel. In addition to the *common properties*, this feature has the following functionalities:

The Elevation profile *Item Properties* panel embeds a top toolbar with the following functionalities:

- Update elevation profile to refresh the item rendering
- Copy from elevation profile: a drop-down menu to select an elevation profile view from. The view settings are applied to the layout elevation profile item, and can be later modified.

Layers

Under the *Layers* group, check in the tree view the layers you would like to render in the profile item. Remember to properly configure the *Elevation* properties of the selected layers.

Profile curve

- *Controlled by atlas*: the *profile curve* will be taken from the current atlas feature and the elevation profile view updated as you walk through the atlas features. This is currently supported for an active layout atlas or report, using a line geometry type coverage layer.
- The *Tolerance* distance, which can be data-defined, helps you control how far a feature of the visible layers should be from the profile curve in order to display in the layout elevation view. Only point features are currently returned.

Chart ranges

A layout elevation profile item does not necessarily display the full extent of the elevation profile view it is based on. You can limit the area to render, providing:

- on the X axis, the *Minimum distance* and *Maximum distance* from the profile curve starting point
- On the Y axis, the *Minimum elevation* and *Maximum elevation*

Distance and elevation axes

The *Distance axis* and *Elevation axis* groups give options to tweak the grid over the elevation profile item, respectively on the X and Y axes:

- the graduation on the axis with both a *Major interval* and *Minor interval*
- the line symbols to apply to the corresponding *Major grid lines* and *Minor grid lines*
- how regular the graduation items should be labeled (*Label interval*) as well as their *Label format* and *Label font*

Chart area

Under *Chart area*, you can configure the rendering of the area in which the elevation profile plot is actually displayed:

- a *Background* fill symbol
- a *Border* line symbol
- the margins from the elevation profile item border

22.2.10 Elementul HTML de tip cadru

It is possible to add a frame that displays the contents of a website or even create and style your own HTML page and display it! You can add a picture with the *Add HTML* following *items creation instructions* and manipulate it the same way as exposed in *Interacting with layout items*. Note that the HTML scale is controlled by the layout export resolution at the time the HTML frame is created.

The HTML item can be customized using its *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities (see Fig. 22.55):

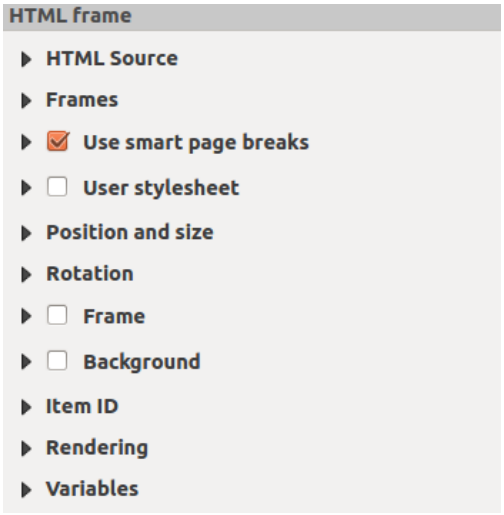


Fig. 22.55: Cadrul HTML, Panoul Proprietăților Elementului

Sursa HTML

The *HTML Source* group of the HTML frame *Item Properties* panel provides the following functionalities (see Fig. 22.56):

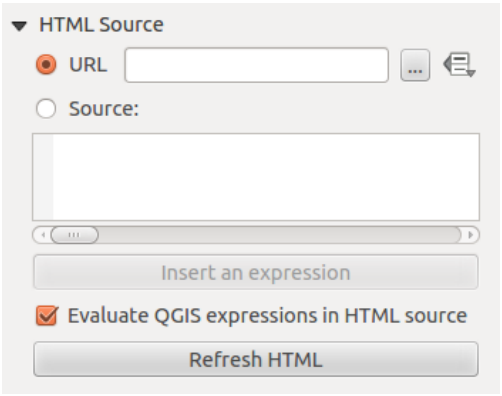


Fig. 22.56: Cadrul HTML, Proprietățile Sursei HTML

- In *URL* you can enter the URL of a webpage you copied from your Internet browser or select an HTML file using the … Browse button. There is also the option to use the Data-defined override button, to provide a URL from the contents of an attribute field of a table or using a regular expression.
- În cadrul casetei de text *Sursă* se poate introduce un text , însoțit de etichete HTML, sau să furnizați o pagină HTML completă.
- The *Insert or Edit an Expression…* button can be used to add an expression like `[%Year($now)%]` in the Source textbox to display the current year. This button is only activated when radiobutton *Source* is selected. After inserting the expression click somewhere in the textbox before refreshing the HTML frame, otherwise you will lose the expression.
- Activați *Evaluarea expresiilor QGIS din codul HTML* pentru a vedea rezultatul expresiei pe care ați inclus-o, în caz contrar, veți vedea expresia originală.
- Use the *Refresh HTML* button to refresh the HTML frame(s) and see the result of changes.

Cadrele

The *Frames* group of the HTML frame *Item Properties* panel provides the following functionalities (see Fig. 22.57):

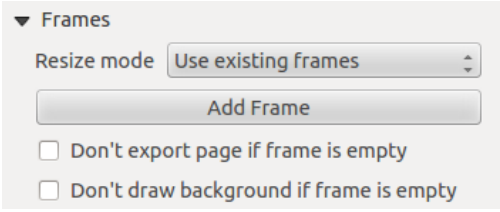


Fig. 22.57: HTML frame, the Frames properties

- Cu ajutorul *Modului de redimensionare* puteți selecta randarea conținutului HTML:
  - `Use existing frames` displays the result in the first frame and added frames only.
  - `Extend to next page` will create as many frames (and corresponding pages) as necessary to render the height of the web page. Each frame can be moved around on the layout. If you resize a frame, the webpage will be divided up between the other frames. The last frame will be trimmed to fit the web page.
  - `Repeat on every page` will repeat the upper left of the web page on every page in frames of the same size.
  - `Repeat until finished` will also create as many frames as the `Extend to next page` option, except all frames will have the same size.
- Use the *Add Frame* button to add another frame with the same size as selected frame. If the HTML page does not fit in the first frame it will continue in the next frame when you use *Resize mode* or *Use existing frames*.
- Activate *Don't export page if frame is empty* prevents the page from being exported when the frame has no HTML contents. This means all other layout items, maps, scale bars, legends etc. will not be visible in the result.
- Activarea opțiunii de *Nedesenare a fundalului în cazul unui cadru vid* previne desenarea cadrului HTML atunci când cadrul nu conține nimic.

Folosirea salturilor de pagină inteligente și a paginii de stil a Utilizatorului

The *Use smart page breaks* dialog and *User style sheet* dialog of the HTML frame *Item Properties* panel provides the following functionalities (see Fig. 22.58):

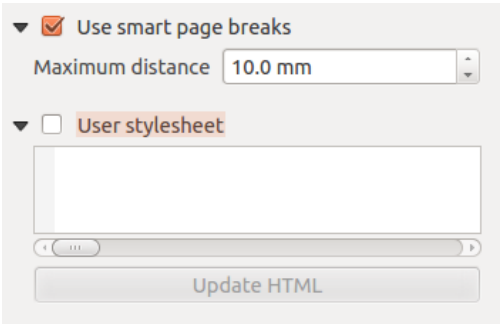


Fig. 22.58: HTML frame, Use smart page breaks and User style sheet properties

- Activați *Folosirea întreruperilor de pagină inteligente* pentru a preveni ruperea liniilor de text la jumătate din cadrele cu conținut html, astfel încât ele să continue frumos și neted în cadrul următor.
- Set the *Maximum distance* allowed when calculating where to place page breaks in the html. This distance is the maximum amount of empty space allowed at the bottom of a frame after calculating the optimum break

location. Setting a larger value will result in better choice of page break location, but more wasted space at the bottom of frames. This is only used when *Use smart page breaks* is activated.

- Activate *User style sheet* to apply HTML styles that often is provided in cascading style sheets. An example of style code is provided below to set the color of `<h1>` header tag to green and set the font and font size of text included in paragraph tags `<p>`.

```
h1 {color: #00ff00;
}
p {font-family: "Times New Roman", Times, serif;
   font-size: 20px;
}
```

- Use the *Update HTML* button to see the result of the style sheet settings.

22.2.11 The Shape Items

QGIS oferă câteva instrumente pentru a desena forme simple sau complexe în cadrul compozițiilor.

Notă: Unlike other print layout items, you can not style the frame nor the background color of the shapes bounding frame (set to transparent by default).

The Regular Shape Item

The *Shape* item is a tool that helps to decorate your map with regular shapes like triangle, rectangle, ellipse… You can add a regular shape using the Add Shape tool which gives access to particular tools like Add Rectangle, Add Ellipse and Add Triangle. Once you have selected the appropriate tool, you can draw the item following *items creation instructions*. Like other layout items, a regular shape can be manipulated the same way as exposed in *Interacting with layout items*.

Notă: Holding down the `Shift` key while drawing the basic shape with the click and drag method helps you create a perfect square, circle or triangle.

The default shape item can be customized using its *Item Properties* panel. Other than the *items common properties*, this feature has the following functionalities (see Fig. 22.59):

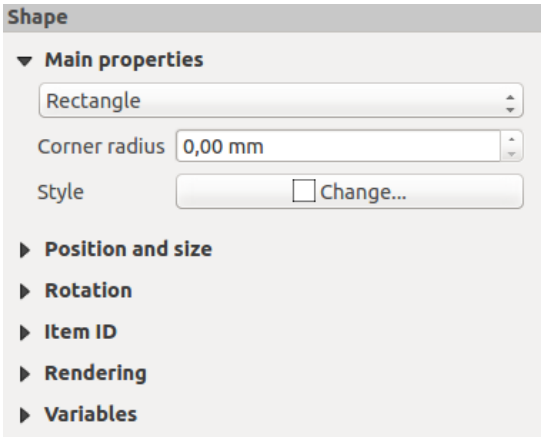


Fig. 22.59: Panoul Proprietăților pentru Elementul de tip Formă

The *Main properties* group shows and allows you to switch the type of the shape item (**Ellipse**, **Rectangle** or **Triangle**) inside the given frame.

You can set the style of the shape using the advanced *symbol* and *color* selector widget…

For the rectangle shape, you can set in different units the value of the *Corner radius* to round of the corners.

Formele Articulate

While the *Add Shape* tool provides way to create simple and predefined geometric item, the *Add Node Item* tool helps you create a custom and more advanced geometric item. For polylines or polygons, you can draw as many lines or sides as you want and vertices of the items can be independently and directly manipulated using the *Edit Nodes Item*. The item itself can be manipulated as exposed in *Interacting with layout items*.

Pentru a adăuga o formă articulată:

1. Click the Add Node Item icon
2. Select either Add Polygon or Add Polyline tool
3. Perform consecutive left clicks to add nodes of your item. If you hold down the `Shift` key while drawing a segment, it is constrained to follow an orientation multiple of 45°.
4. When you're done, right-click to terminate the shape.

You can customize the appearance of the shape in the *Item Properties* panel.

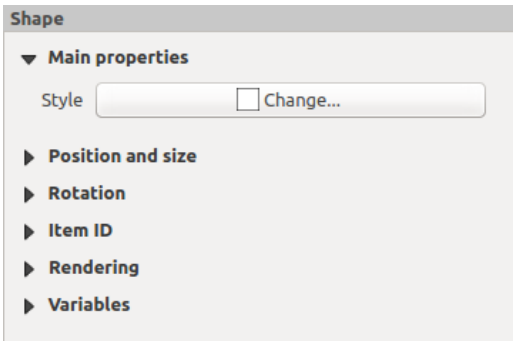


Fig. 22.60: Polygon Node Shape Item Properties Panel

In the *Main properties*, you can set the style of the shape using the advanced *symbol* and *color* selector widget…

For polyline node items, you can also parameterize the *Line markers* i.e. add:

- start and/or end markers with options:
  - *None*: draws a simple polyline.
  - *Arrow*: adds a regular triangular arrow head that you can customize.
  - *SVG* marker: uses an `SVG` file as arrow head of the item.
- customize the arrow head:
  - *Arrow stroke color*: sets the stroke color of the arrow head.
  - *Arrow fill color*: sets the fill color of the arrow head.
  - *Arrow stroke width*: sets the stroke width of the arrow head.
  - *Arrow head width*: sets the size of the arrow head.

SVG images are automatically rotated with the line. Stroke and fill colors of QGIS predefined SVG images can be changed using the corresponding options. Custom SVG may require some tags following this *instruction*.

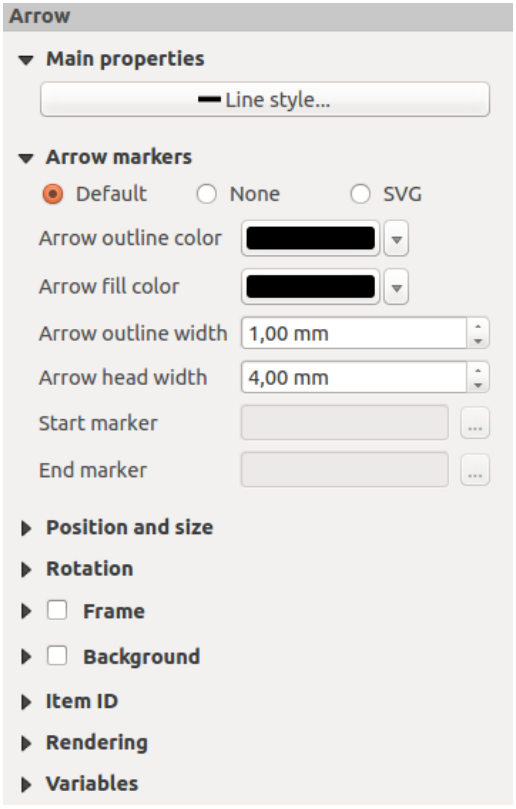


Fig. 22.61: Polyline Node Shape Item Properties Panel

Elementul Săgeată

The Add Arrow tool is a shortcut to create an arrow-enabled polyline by default and thus has the same properties and behavior as a *polyline node item*.

Actually, the arrow item can be used to add a simple arrow, for example, to show the relation between two different print layout items. However, to create a north arrow, the *image item* should be considered first as it gives access to a set of north arrows in `.SVG` format that you can sync with a map item so that it rotates automatically with it.

Editing a node item geometry

Editarea formelor articulate se efectuează cu instrumentul specific de Editare a Formelor Articulate. Selectarea unui nod se efectuează printr-un simplu clic, cu ajutorul acestui instrument (nodul selectat fiind pus în evidență). Un nod selectat poate fi mutat fie prin glisarea lui, fie prin acționarea tastelor cu săgeți. Mai mult, se pot adăuga noduri la o formă existentă: un dublu-clic pe un segment va adăuga un nod în locul în care s-a efectuat clicul. În cele din urmă, nodul selectat poate fi eliminat prin apăsarea tastei `Del`.

22.3 Generarea unui Rezultat

Fig. 22.62 shows an example print layout including all the types of layout items described in the previous section.

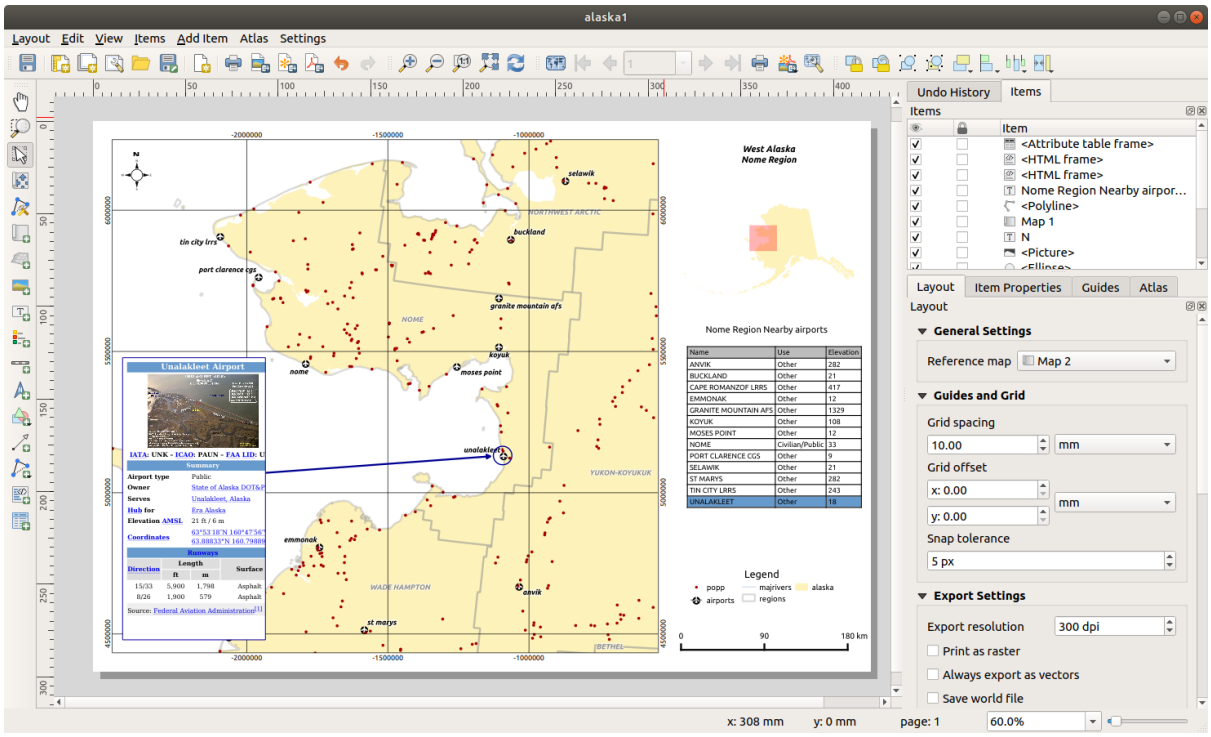


Fig. 22.62: Print Layout with map view, legend, image, scale bar, coordinates, text and HTML frame added

From the *Layout* menu or toolbar, you can output the print layout to different file formats, and it is possible to modify the resolution (print quality) and paper size:

- The Print icon allows you to print the layout to a connected printer or a PostScript file, depending on the installed printer drivers.
- The Export as image icon exports the print layout image formats such as `PNG`, `BMP`, `TIF`, `JPG`, and many others…
- The Export as SVG icon saves the print layout as an `SVG` (Scalable Vector Graphic).
- The Export as PDF icon saves the defined print layout directly as a `PDF` (Portable Document Format) file.

22.3.1 Setări de export

Whenever you export a print layout, there are a selection of export settings QGIS needs to check in order to produce the most appropriate output. These configurations are:

- The *Export settings* of the *Layout* panel, such as *Export resolution*, *Print as raster* *Always export as vectors* or *Save world file*
- *Exclude page from exports* in the *page item properties* panel
- *Exclude item from exports* in the *item properties* panel

Moreover, a number of predefined checks are automatically applied to the layout. Currently these checks include testing that scalebars are correctly linked to map items, and that map overview items are also correctly linked to a map. If the checks fail, you are shown a nice warning advising you of the issue.

22.3.2 Exportare ca imagine

To export a layout as an image:

1. Click the Export as image icon
2. Select the image format, the folder and filename (e.g. `myill.png`) to use. If the layout contains more than one page, each page will be exported to a file with the given filename with the page number appended (e.g. `myill_2.png`).
3. In the next (*Image Export Options*) dialog:
   - You can override the print layout *Export resolution* and the exported page dimensions (as set in *Layout* panel).
   - Image rendering can also be improved with the *Enable antialiasing* option.
   - If you want to export your layout as a **georeferenced image** (e.g., to share with other projects), check the *Generate world file* option, and an *ESRI World File* with the same name as the exported image, but a different extension (`.tfw` for TIFF, `.pnw` for PNG, `jgw` for JPEG, ...) will be created when exporting. This option can also be checked by default in the *layout panel*.

     Notă: For multi-page output, only the page that contains the *reference map* will get a world file (assuming that the *Generate world file* option is checked).

   - By checking *Crop to content* option, the image output by the layout will include the minimal area enclosing all the items (map, legend, scale bar, shapes, label, image...) of each page of the composition:
     - If the composition includes a single page, then the output is resized to include EVERYTHING on the composition. The page can then be reduced or extended to all items depending on their position (on, above, below, left or right of the page).
     - In case of a multi-page layout, each page will be resized to include items in its area (left and right sides for all pages, plus top for the first page and bottom for the last page). Each resized page is exported to a separate file.

     The *Crop to content* dialog also lets you add margins around the cropped bounds.
   - By checking *Open file after exporting* the exported file will automatically open in the default image viewer.

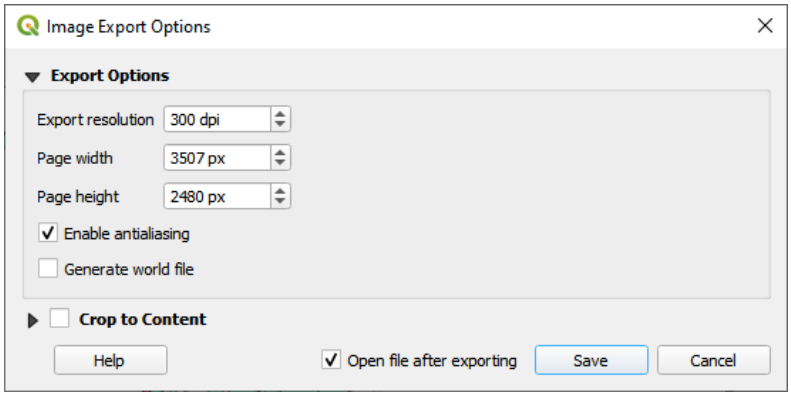


Fig. 22.63: Image Export Options

Sfat: Use image formats that support transparency when items extend beyond the paper extent

Layout items may be placed outside the paper extent. When exporting with the *Crop to content* option, the resulting image may therefore extend beyond the paper extent. Since the background outside of the paper extent will be

transparent, for image formats that do not support transparency (e.g. `BMP` and `JPG`) the transparent background will be rendered as full black, „corrupting" the image. Use transparency-compatible formats (e.g. `TIFF` and `PNG`) in such cases.

Notă: When supported by the format (e.g. `PNG`) and the underlying Qt library, the exported image may include *project metadata* (author, title, date, description…)

22.3.3 Exportare ca SVG

To export a layout as SVG:

1. Click the Export as SVG icon
2. Fill in the path and filename (used as a base name for all the files in case of multi-page composition, as for image export)
3. In the next *SVG Export Options* dialog, you can override the layout default *export settings* or configure new ones:
   - *Export map layers as SVG groups*: exported items are grouped within layers whose name matches the layer names from QGIS, making it much easier to understand the contents of the document.
   - *Always export as vectors*: some rendering options require items to be rasterized for a better rendering. Check this option to keep the objects as vectors with the risk that the appearance of the output file may not match the print layout preview (for more details, see *Setări de export*).
   - *Export RDF metadata* of the document such as the title, author, date, description…
   - *Simplify geometries to reduce output file size*: this avoids exporting ALL geometry vertices, which can result in a ridiculously complex and large export file size that could fail to load in other applications. Geometries will be simplified while exporting the layout in order to remove any redundant vertices which are not discernably different at the export resolution (e.g. if the export resolution is `300 dpi`, vertices that are less than `1/600 inch` apart will be removed).
   - Set the *Text export*: controls whether text labels are exported as proper text objects (*Always export texts as text objects*) or as paths only (*Always export texts as paths*). If they are exported as text objects, they can be edited in external applications (e.g. Inkscape) as normal text. BUT the side effect is that the rendering quality is reduced, AND there are issues with rendering when certain text settings like buffers are in place. That's why exporting as paths is recommended.
   - Apply *Crop to content* *option*
   - *Disable tiled raster layer exports*: When exporting files, QGIS uses a built-in raster layer tiled rendering that saves memory. Sometimes, this can cause visible „seams" in the rasters for generated files. Checking this option would fix that, at the cost of a higher memory usage during exports.
   - By checking *Open file after exporting* the exported file will automatically open in the default SVG viewer.

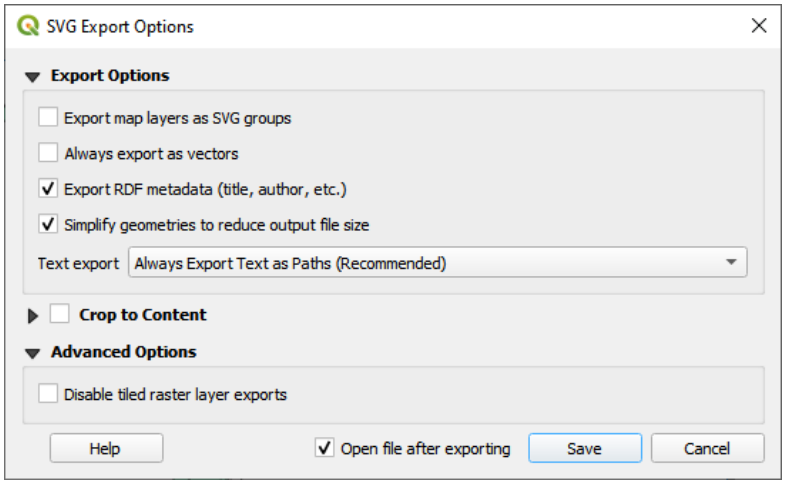


Fig. 22.64: Opțiuni de Export SVG

Notă: În prezent, producerea SVG-ului este foarte simplă. Aceasta nu ține de QGIS, ci de biblioteca Qt de la bază. Acest lucru se va schimba, sperăm, în versiunile viitoare.

22.3.4 Exportare ca PDF

To export a layout as PDF:

1. Click the Export as PDF icon
2. Fill in the path and filename: unlike for image and SVG export, all the pages in the layout are exported to a single PDF file.
3. In the next *PDF Export Options* dialog, you can override the layout default *export settings* or configure new ones:
   - *Always export as vectors*: some rendering options require items to be rasterized for a better rendering. Check this option to keep the objects as vectors with the risk that the appearance of the output file may not match the print layout preview (for more details, see *Setări de export*).
   - *Append georeference information*: available only if the *reference map*, from which the information is taken, is on the first page.
   - *Export RDF metadata* of the document such as the title, author, date, description…
   - Set the *Text export*: controls whether text labels are exported as proper text objects (*Always export texts as text objects*) or as paths only (*Always export texts as paths*). If they are exported as text objects then they can be edited in external applications (e.g. Inkscape) as normal text. BUT the side effect is that the rendering quality is decreased, AND there are issues with rendering when certain text settings like buffers are in place. That's why exporting as paths is recommended.
   - Control the PDF *Image compression* using:
     - *Lossy (JPEG)*, which is the default compression mode
     - or *Lossless*, which creates bigger files in most cases, but is much more suitable for printing outputs or for post-production in external applications (requires Qt 5.13 or later).
   - *Create Geospatial PDF (GeoPDF)*: Generate a georeferenced PDF file.
   - *Disable tiled raster layer exports*: When exporting files, QGIS uses tiled based rendering that saves memory. Sometimes, this can cause visible „seams" in the rasters for generated files. Checking this option would fix that, at the cost of a higher memory usage during exports.

- *Simplify geometries to reduce output file size*: Geometries will be simplified while exporting the layout by removing vertices that are not discernably different at the export resolution (e.g. if the export resolution is `300 dpi`, vertices that are less than `1/600 inch` apart will be removed). This can reduce the size and complexity of the export file (very large files can fail to load in other applications).
- By checking *Open file after exporting* the exported file will automatically open in the default PDF viewer.

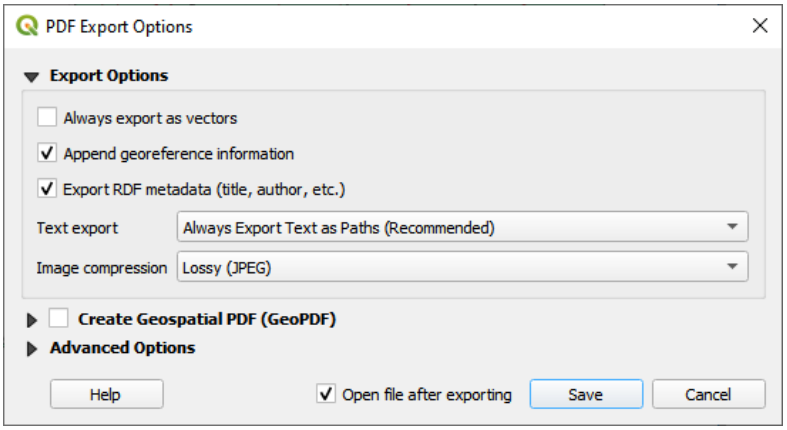


Fig. 22.65: PDF Export Options

Notă: GeoPDF export is supported, and a number of GeoPDF specific options are available:

- *Format* (GeoPDF format - there are some GeoPDF variations),
- *Include multiple map themes* (specify map themes to include),
- *Include vector feature information* (choose the layers and group them into logical PDF groups).

Notă: Exporting a print layout to formats that supports georeferencing (e.g. `PDF` and `TIFF`) creates a georeferenced output by default.

22.3.5 Generarea unui Atlas

Atlas functions allow you to create map books in an automated way. Atlas uses the features of a table or vector layer (*Coverage layer*) to create an output for each feature (**atlas feature**) in the table / layer. The most common usage is to zoom a map item to the current atlas feature. Further use cases include:

- a map item showing, for another layer, only features that share the same attribute as the atlas feature or are within its geometry.
- a label or HTML item whose text is replaced as features are iterated over
- a table item showing attributes of associated *parent or children* features of the current atlas feature…

For each feature, the output is processed for all pages and items according to their exports settings.

Sfat: Use variables for more flexibility

QGIS provides a large panel of functions and *variables*, including atlas related ones, that you can use to manipulate the layout items, but also the symbology of the layers, according to atlas status. Combining these features gives you a lot of flexibility and helps you easily produce advanced maps.

To enable the generation of an atlas and access atlas parameters, refer to the *Atlas* panel. This panel contains the following (see Fig. 22.66):

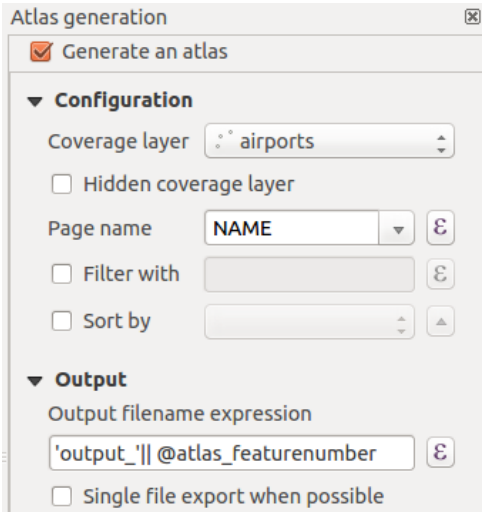


Fig. 22.66: Atlas Panel

- *Generate an atlas* enables or disables atlas generation.
- *Configuration*
  - A *Coverage layer* combo box that allows you to choose the table or vector layer containing the features to iterate over.
  - An optional *Hidden coverage layer* that, if checked, will hide the coverage layer (but not the other layers) during the generation.
  - An optional *Page name* combo box to specify the name for the feature page(s). You can select a field of the coverage layer or set an *expression*. If this option is empty, QGIS will use an internal ID, according to the filter and/or the sort order applied to the layer.
  - An optional *Filter with* text area that allows you to specify an expression for filtering features from the coverage layer. If the expression is not empty, only features that evaluate to `True` will be processed.
  - An optional *Sort by* that allows you to sort features of the coverage layer (and the output), using a field of the coverage layer or an expression. The sort order (either ascending or descending) is set by the two-state *Sort direction* button that displays an up or a down arrow.
- *Output* - this is where the output of the atlas can be configured:
  - An *Output filename expression* textbox that is used to generate a filename for each atlas feature. It is based on expressions. is meaningful only for rendering to multiple files.
  - A *Single file export when possible* that allows you to force the generation of a single file if this is possible with the chosen output format (`PDF`, for instance). If this field is checked, the value of the *Output filename expression* field is meaningless.
  - An *Image export format* drop-down list to select the output format when using the Export atlas as Images… button.

Control map by atlas

The most common usage of atlas is with the map item, zooming to the current atlas feature, as iteration goes over the coverage layer. This behavior is set in the *Controlled by atlas* group properties of the map item. See *Controlled by atlas* for different settings you can apply on the map item.

Customize labels with expression

In order to adapt labels to the feature the atlas iterates over, you can include expressions. Make sure that you place the expression part (including functions, fields or variables) between `[%` and `%]` (see *Elementul Etichetă* for more details).

For example, for a city layer with fields `CITY_NAME` and `ZIPCODE`, you could insert this:

```
The area of [% concat( upper(CITY_NAME), ',', ZIPCODE, ' is ',
format_number($area/1000000, 2) ) %] km2
```

sau, o altă combinație:

```
The area of [% upper(CITY_NAME)%],[%ZIPCODE%] is
[%format_number($area/1000000,2) %] km2
```

The information `[% concat( upper(CITY_NAME), ',', ZIPCODE, ' is ', format_number($area/1000000, 2) ) %]` is an expression used inside the label. Both expressions would result in the following type of label in the generated atlas:

```
The area of PARIS,75001 is 1.94 km2
```

Explore Data-defined override buttons with atlas

There are several places where you can use a Data defined override button to override the selected setting. This is particularly useful with atlas generation. See *Configurarea suprascrierii definită de date* for more details on this widget.

For the following examples the `Regions` layer of the QGIS sample dataset is used and selected as *Coverage layer* for the atlas generation. We assume that it is a single page layout containing a map item and a label item.

When the height (north-south) of a region extent is greater than its width (east-west), you should use *Portrait* instead of *Landscape* orientation to optimize the use of paper. With a Data Defined Override button you can dynamically set the paper orientation.

Right-click on the page and select *Page Properties* to open the panel. We want to set the orientation dynamically, using an expression depending on the region geometry, so press the button of field *Orientation*, select *Edit...* to open the *Expression string builder* dialog and enter the following expression:

```
CASE WHEN bounds_width(@atlas_geometry) > bounds_height(@atlas_geometry)
THEN 'Landscape' ELSE 'Portrait' END
```

Now if you *preview the atlas*, the paper orients itself automatically, but item placements may not be ideal. For each Region you need to reposition the location of the layout items as well. For the map item you can use the button of its *Width* property to set it dynamic using the following expression:

```
@layout_pagewidth - 20
```

Likewise, use the button of the *Height* property to provide the following expression to constrain map item size:

```
@layout_pageheight - 20
```

To ensure the map item is centered in the page, set its *Reference point* to the upper left radio button and enter `10` for its *X* and *Y* positions.

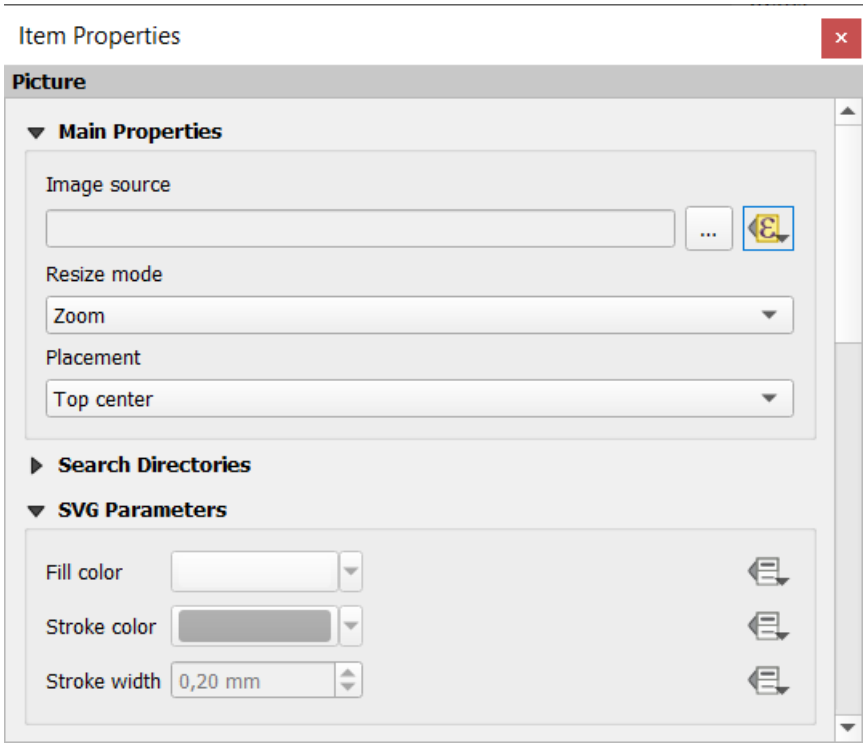
Let's add a title above the map in the center of the page. Select the label item and set the horizontal alignment to *Center*. Next move the label to the right position, choose the middle button for the *Reference point*, and provide the following expression for field *X*:

```
@layout_pagewidth / 2
```

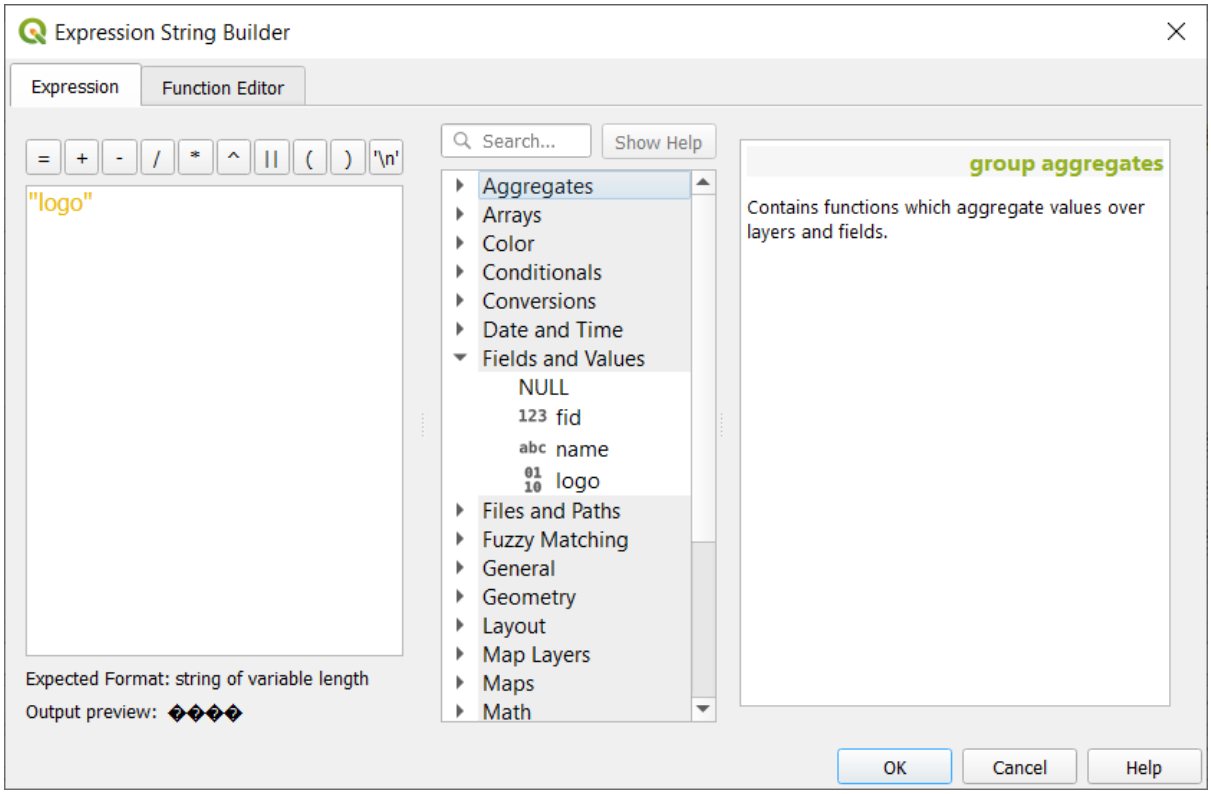
For all other layout items you can set the position in a similar way so they are correctly positioned both for portrait and landscape. You can also do more tweaks such as customizing the title with feature attributes (see *Customize labels with expression* example), changing images, resizing the number of legend columns number according to page orientation, …

The information provided here is an update of the excellent blog (in English and Portuguese) on the Data Defined Override options Multiple_format_map_series_using_QGIS_2.6 .

Another example for using data-defined override buttons is the usage of a dynamic picture. For the following examples we use a geopackage layer containing a BLOB field called `logo` with the field type binary (see *Creating a new GeoPackage layer*). For every feature there is defined a different picture so that the atlas can iterate over as described in *Preview and generate an atlas*. All you need to do is add a picture in the print layout and go to its *Item properties* in the atlas context. There you can find a data-defined override button in the *Image source* section of the *Main Properties*.



In the following window choose *Edit* so that the *Expression String Builder* opens. From the *Fields and values* section you can find the BLOB field that was defined in the geopackage layer. Double-click the field name `logo` and click *OK*.



The atlas iterates over the entries in the BLOB field provided that you choose the geopackage layer as *Coverage layer* (further instructions you can find in *Preview and generate an atlas*).

These are just two examples of how you can use some advanced settings with atlas.

Preview and generate an atlas

Fig. 22.67: Bara instrumentelor de previzualizare a atlasului

Once the atlas settings have been configured, and layout items (map, table, image…) linked to it, you can create a preview of all the pages by choosing *Atlas* ► *Preview Atlas* or clicking the Preview Atlas icon. You can then use the arrows to navigate through all the features:

- Prima entitate
- Entitatea anterioară
- Următoarea entitate
- Ultima entitate

You can also use the combo box to select and preview a specific feature. The combo box shows atlas feature names according to the expression set in the atlas *Page name* option.

As for simple compositions, an atlas can be generated in different ways (see *Generarea unui Rezultat* for more information - just use tools from the *Atlas* menu or toolbar instead of the *Layout* menu.

This means that you can directly print your compositions with *Atlas* ► *Print Atlas*. You can also create a PDF using *Atlas* ► *Export Atlas as PDF…*: You will be asked for a directory to save all the generated PDF files, except if the *Single file export when possible* has been selected. In that case, you'll be prompted to give a filename.

With *Atlas* ► *Export Atlas as Images…* or *Atlas* ► *Export Atlas as SVG…* tool, you're also prompted to select a folder. Each page of each atlas feature composition is exported to the image file format set in *Atlas* panel or to SVG.

Notă: With multi-page output, an atlas behaves like a layout in that only the page that contains the *General settings* will get a world file (for each feature output).

Sfat: Imprimă o entitate specifică a atlasului

If you want to print or export the composition of only one feature of the atlas, simply start the preview, select the desired feature in the drop-down list and click on *Layout* ► *Print* (or *Export…* to any supported file format).

Use project defined relations for atlas creation

For users with HTML and Javascript knowledge it is possible to operate on GeoJSON objects and use project defined relations from the QGIS project. The difference between this approach and using expressions directly inserted into the HTML is that it gives you a full, unstructured GeoJSON feature to work with. This means that you can use existing Javascript libraries and functions that operate on GeoJSON feature representations.

The following code includes all related child features from the defined relation. Using the JavaScript `setFeature` function it allows you to make flexible HTML which represents relations in whatever format you like (lists, tables, etc). In the code sample, we create a dynamic bullet list of the related child features.

```
// Declare the two HTML div elements we will use for the parent feature id
// and information about the children
<div id="parent"></div>
<div id="my_children"></div>

<script type="text/javascript">
   function setFeature(feature)
   {
     // Show the parent feature's identifier (using its "ID" field)
     document.getElementById('parent').innerHTML = feature.properties.ID;
     //clear the existing relation contents
     document.getElementById('my_children').innerHTML = '';
     feature.properties.my_relation.forEach(function(child_feature) {
     // for each related child feature, create a list element
     // with the feature's name (using its "NAME" field)
       var node = document.createElement("li");
       node.appendChild(document.createTextNode(child_feature.NAME));
       document.getElementById('my_children').appendChild(node);
     });
   }
</script>
```

During atlas creation there will be an iteration over the coverage layer containing the parent features. On each page, you will see a bullet list of the related child features following the parent's identifier.

22.4 Generarea unui Raport

This section will help you set up a report in QGIS.

22.4.1 What is it?

By definition, a GIS report is a document containing information organized in a narrative way, containing maps, text, graphics, tables, etc. A report can be prepared ad hoc, periodic, recurring, regular, or as required. Reports may refer to specific periods, events, occurrences, subjects or locations.

In QGIS, a *Report* is an extension of a *Layouts*.

Reports allow users to output their GIS projects in a simple, quick and structured way.

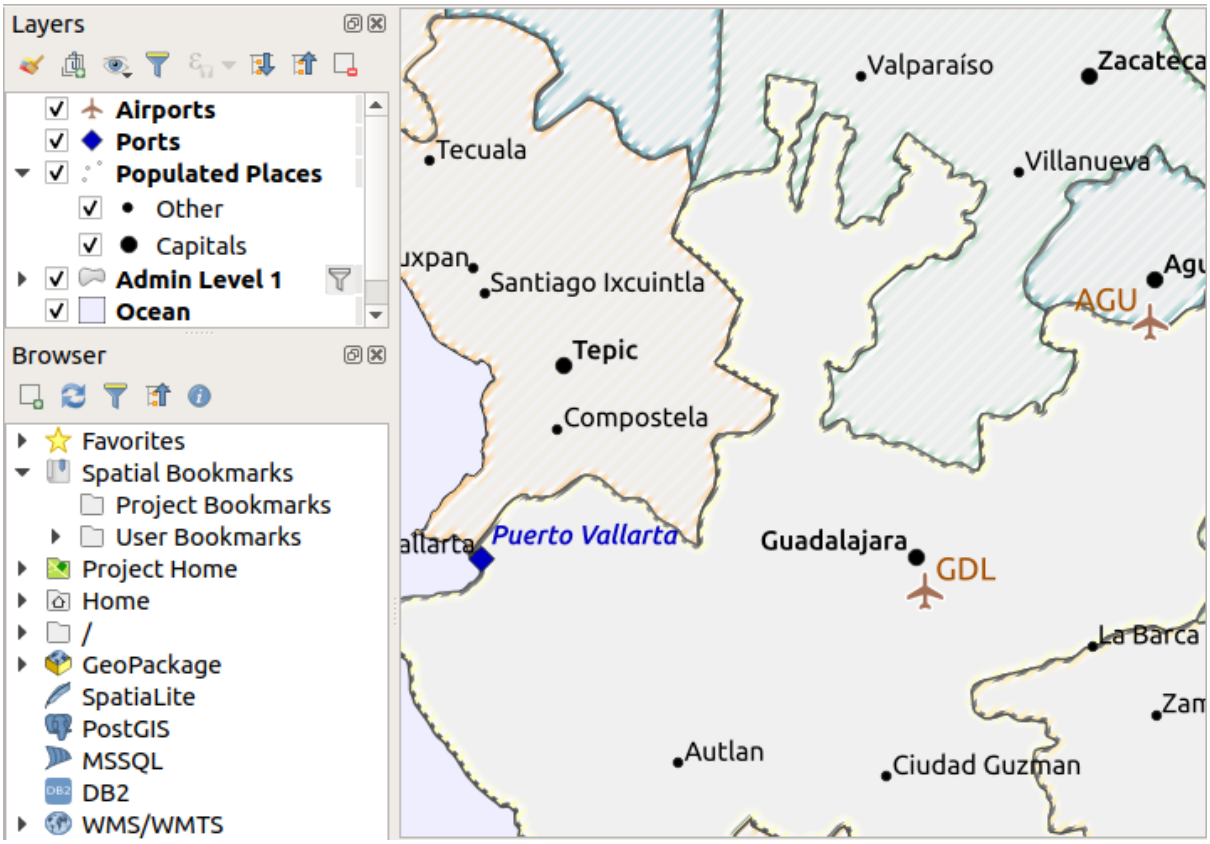
A report can be created with *Project* ► *New Report* or inside the *Project* ► *Layout Manager*.

Notă: The maps in QGIS reports behave in the same way as maps in print layouts and atlases. We will concentrate on the specifics of QGIS reports. For details on map handling, see the sections on *print layouts* and *atlases*.

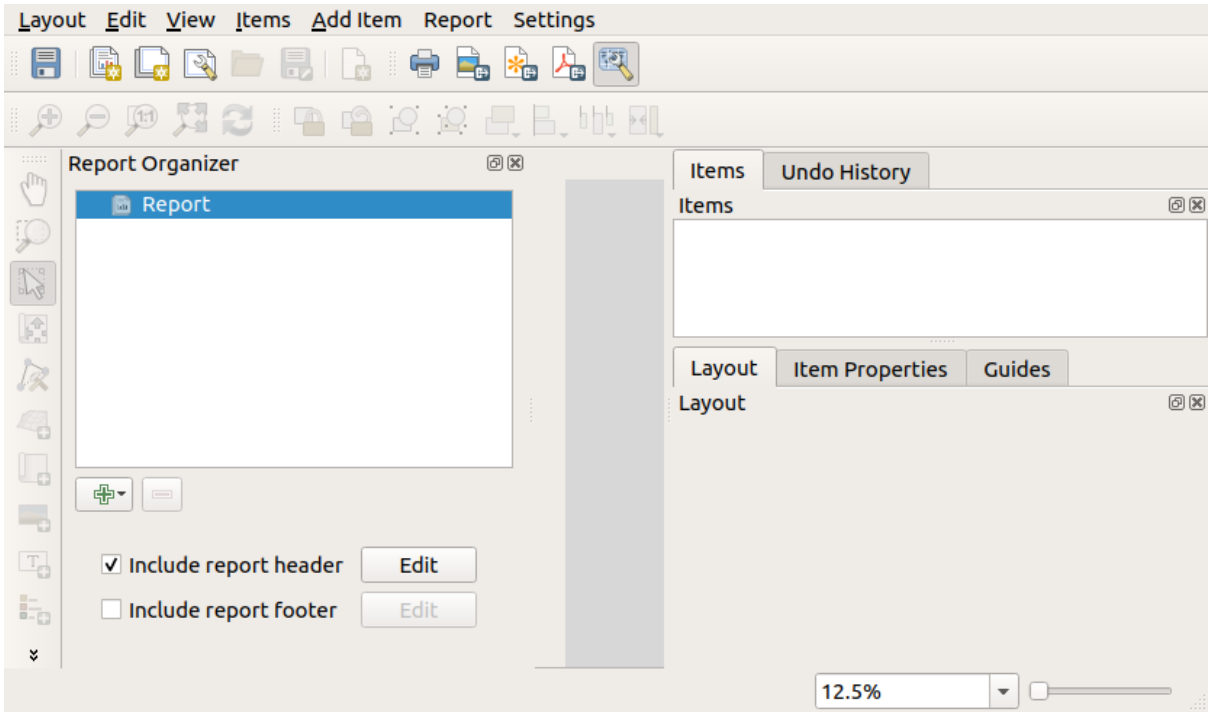
22.4.2 Get started

In the *Layout Manager* dialog a report can be created through *New from template* by selecting the dropdown option *Empty Report* and hitting the *Create…* button.

For this example, we use some administrative boundaries, populated places, ports and airports from the Natural Earth dataset (1:10M).

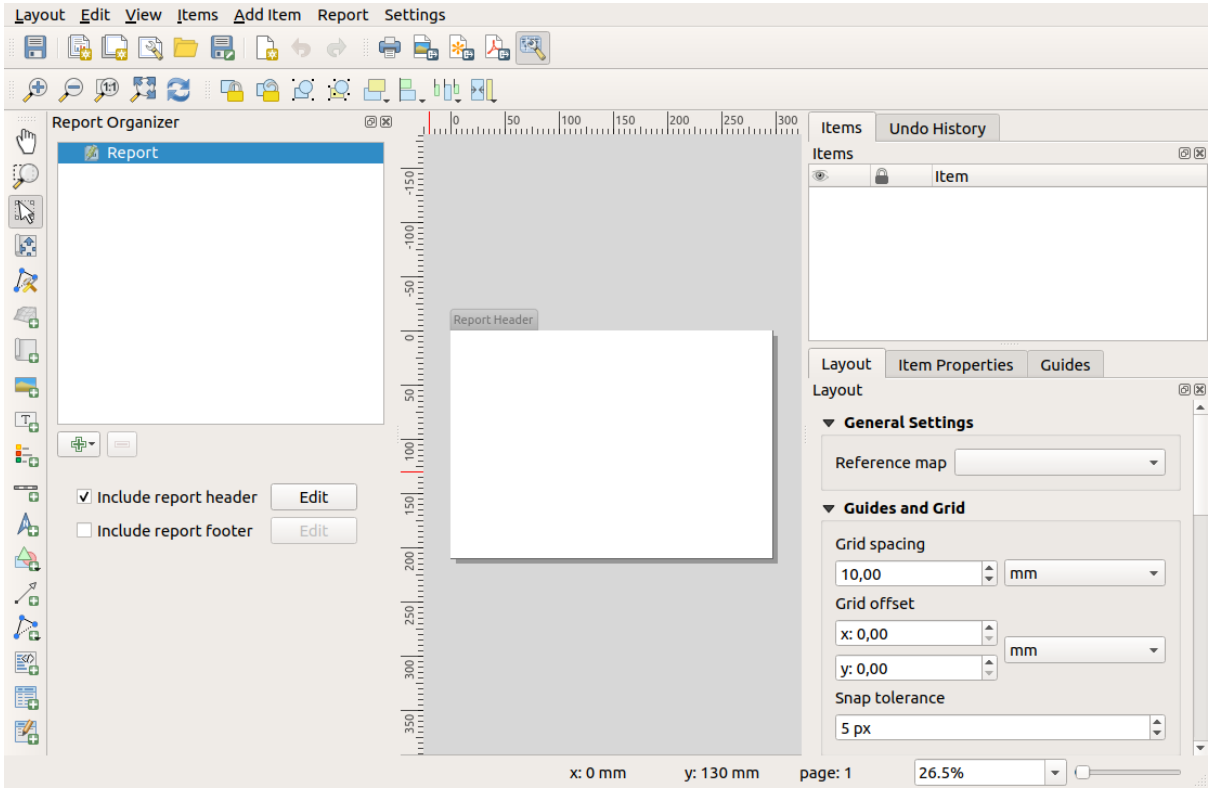


Using the *Project* ► *New Report* command, we create a blank report. Initially, there is not much to look at – the dialog which is displayed looks much like the print layout designer, except for the *Report Organizer* panel to the left:



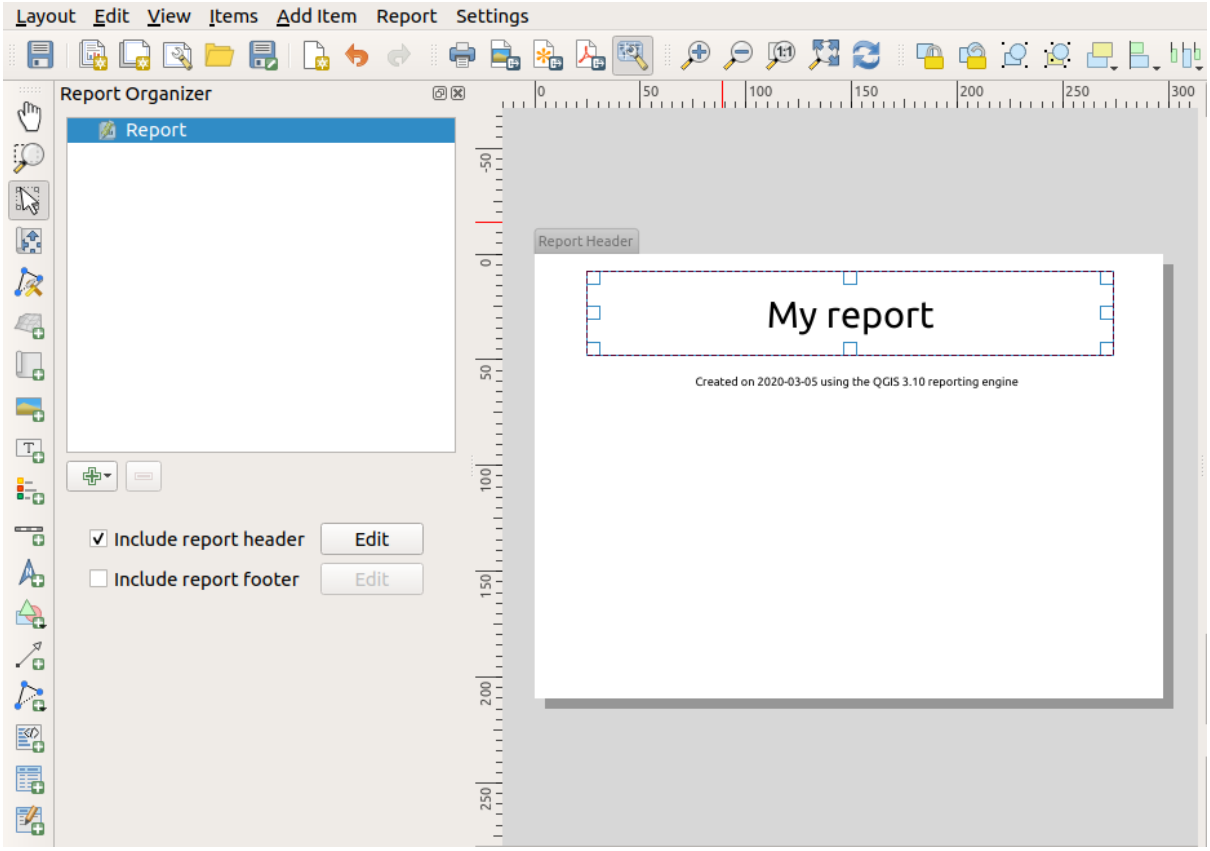
22.4.3 Layout Report Workspace

QGIS reports can consist of multiple, nested sections. In our new blank report we initially only have the main report section. The only options for this report section is *Include report header* and *Include report footer*. If we enable these options, a header will be included as the first page(s) (individual parts of reports can be multi-page if desired) in the report, and a footer will constitute the last page(s). Enable the header (*Include report header*), and hit the *Edit* button next to it:

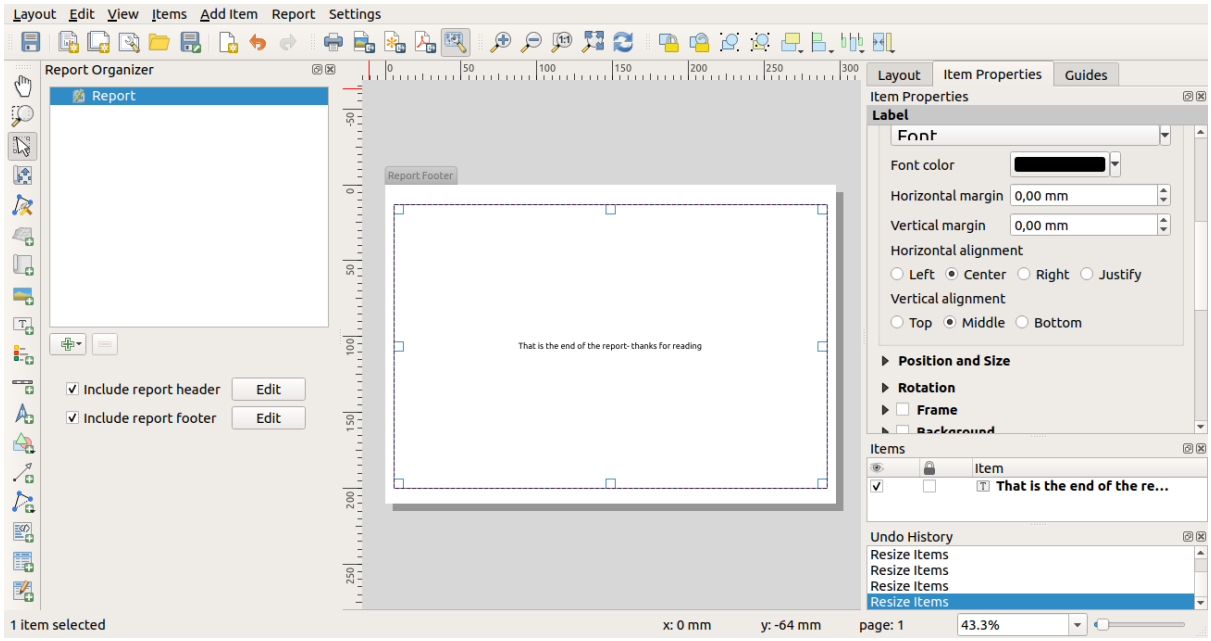


A few things happen as a result. Firstly, an edit pencil is shown next to *Report* in the *Report Organizer*, indicating that the report section is currently being edited in the designer. We also see a new page with a small *Report Header* title. The page has *landscape* orientation by default, but this (and other properties of the page) can be changed by right-clicking on the page and choosing *Page properties*. This will bring up the *Item properties* tab for the page, and page *Size*, *Width*, *Height*, and more can be specified.

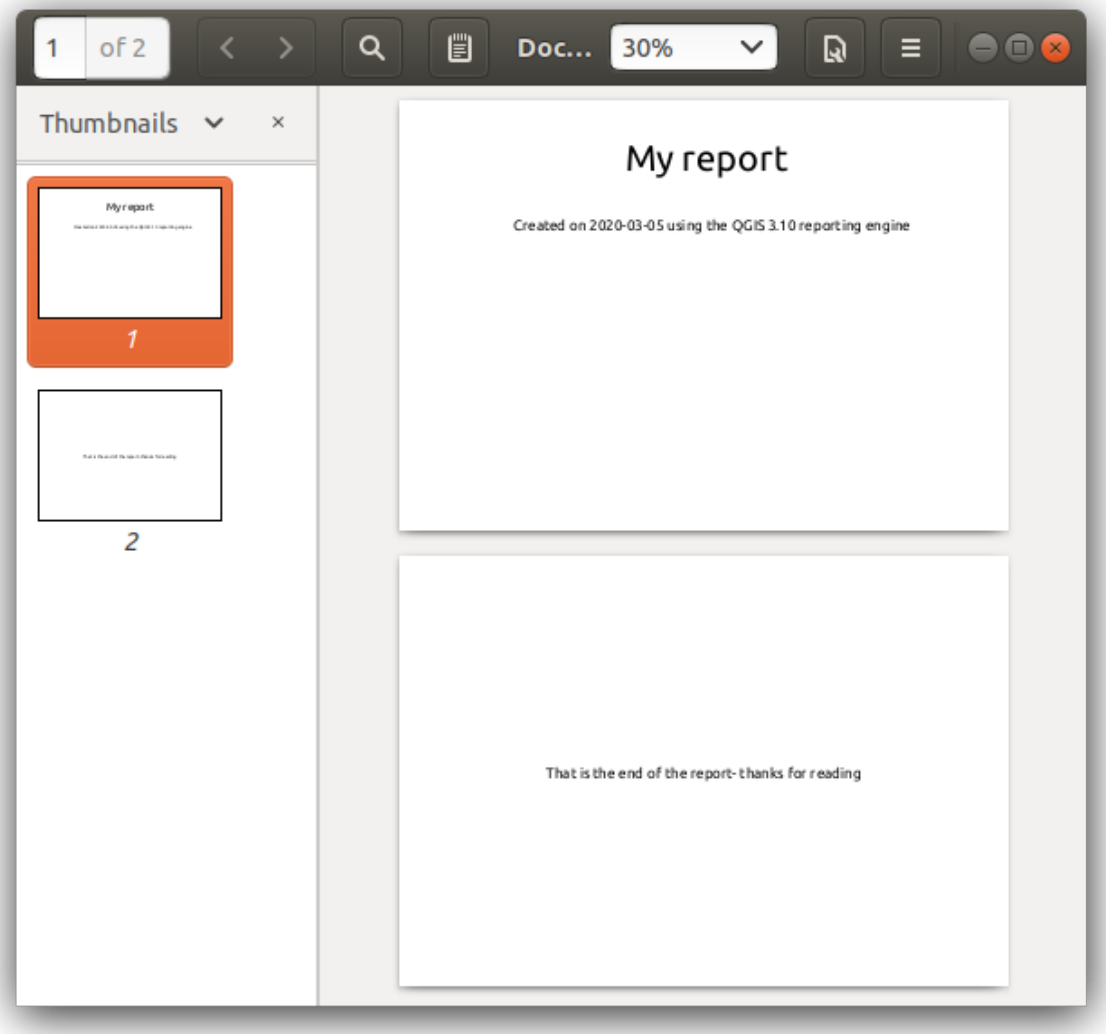
In QGIS reports, every component of the report is made up of individual layouts. They can be created and modified using the same tools as for standard print layouts – so you can use any desired combination of labels, pictures, maps, tables, etc. Let us add some items to our report header to demonstrate:



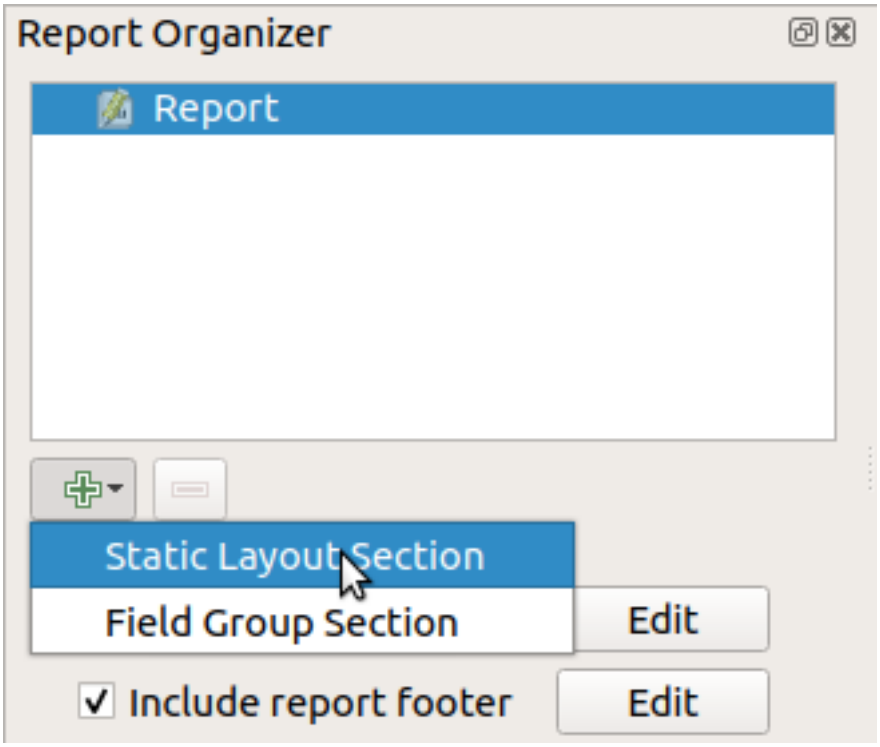
We will also create a simple footer for the report by checking the *Include report footer* option and hitting *Edit*.



Before proceeding further, let us export this report and see what we get. Exporting is done from the *Report* menu – in this case we select *Export Report as PDF…* to render the whole report to a PDF file. Here is the not-very-impressive result – a two page PDF consisting of our header and footer:



Let us make things more interesting. By hitting the Add Section button in the *Report Organizer*, we are given a choice of new sections to add to our report.

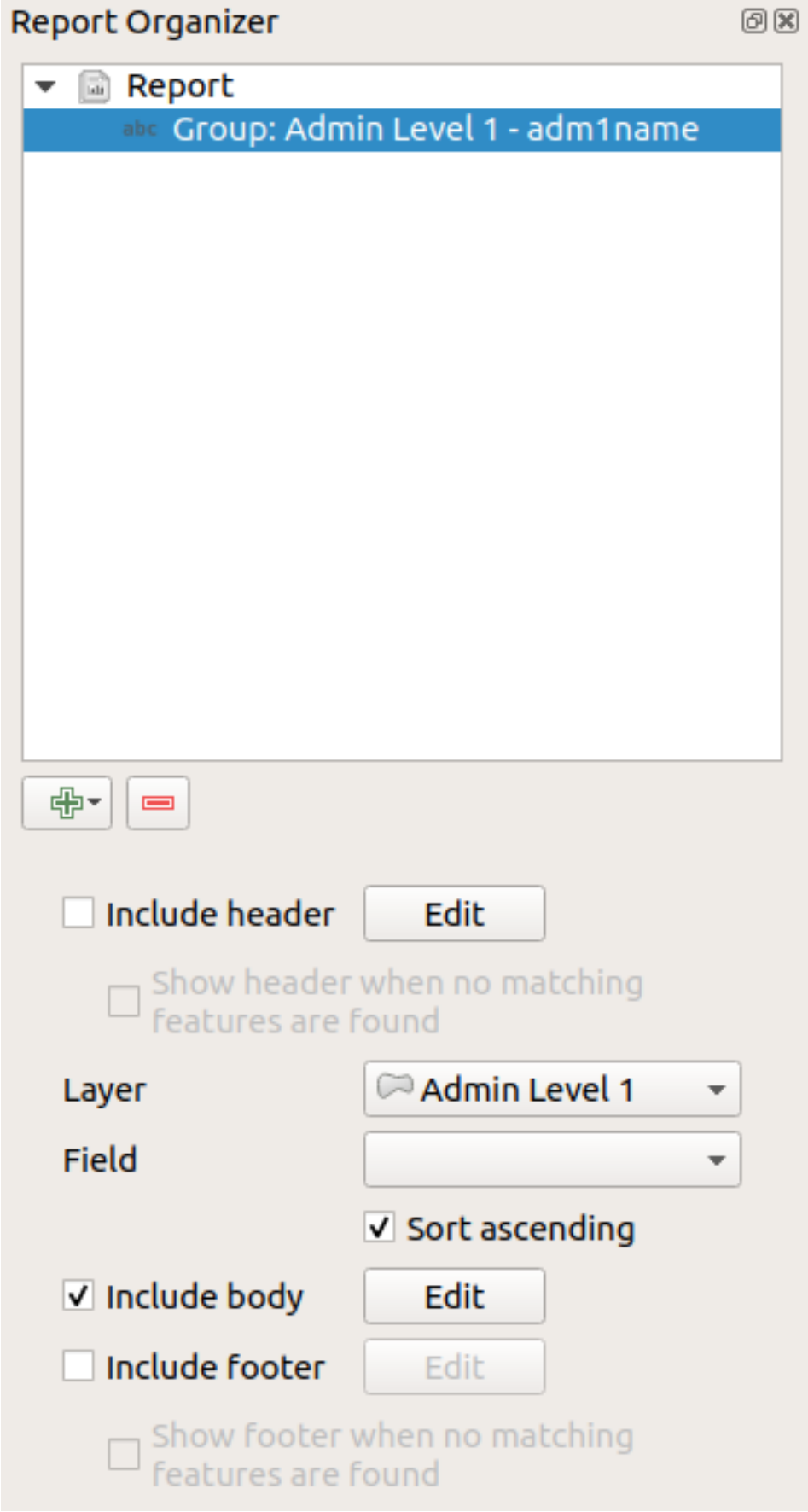


There are two options: *Static Layout Section* and *Field Group Section*.

The *Add Static Layout Section* is a single, static body layout. This can be used to embed static layouts mid-way through a report.

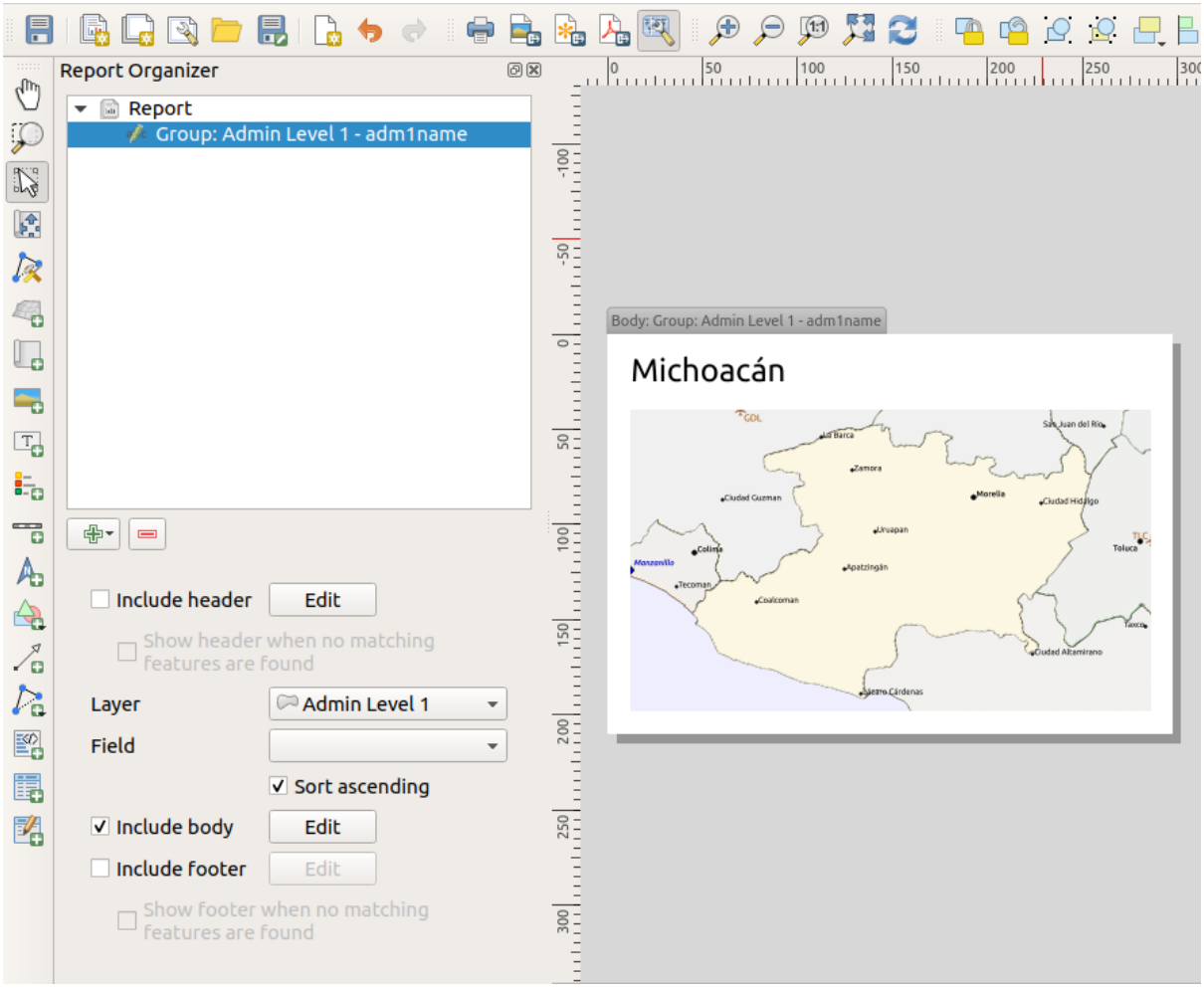
The *Field Group Section* repeats its body layout for every feature of a layer. The features are sorted by the selected grouping feature (with an option for ascending/descending sort). If a field group section has child sections (e.g. another field group section with a different field), then only features with unique values for the group feature are iterated over. This allows nested reports.

For now we will add a *Field Group Section* to our report. At its most basic level, you can think of a *Field Group Section* as the equivalent of a *print atlas*: you select a layer to iterate over, and the report will insert a section for each feature found. Selecting the new *Field Group Section* reveals a number of new related settings:



In this case we've setup our Field Group so that we iterate over all the states from the *Admin Level 1* layer, using the

values from the *adm1name* field. The same options to include header and footer are present, together with a new option to include a *body* for this section. We'll do that, and edit the body:

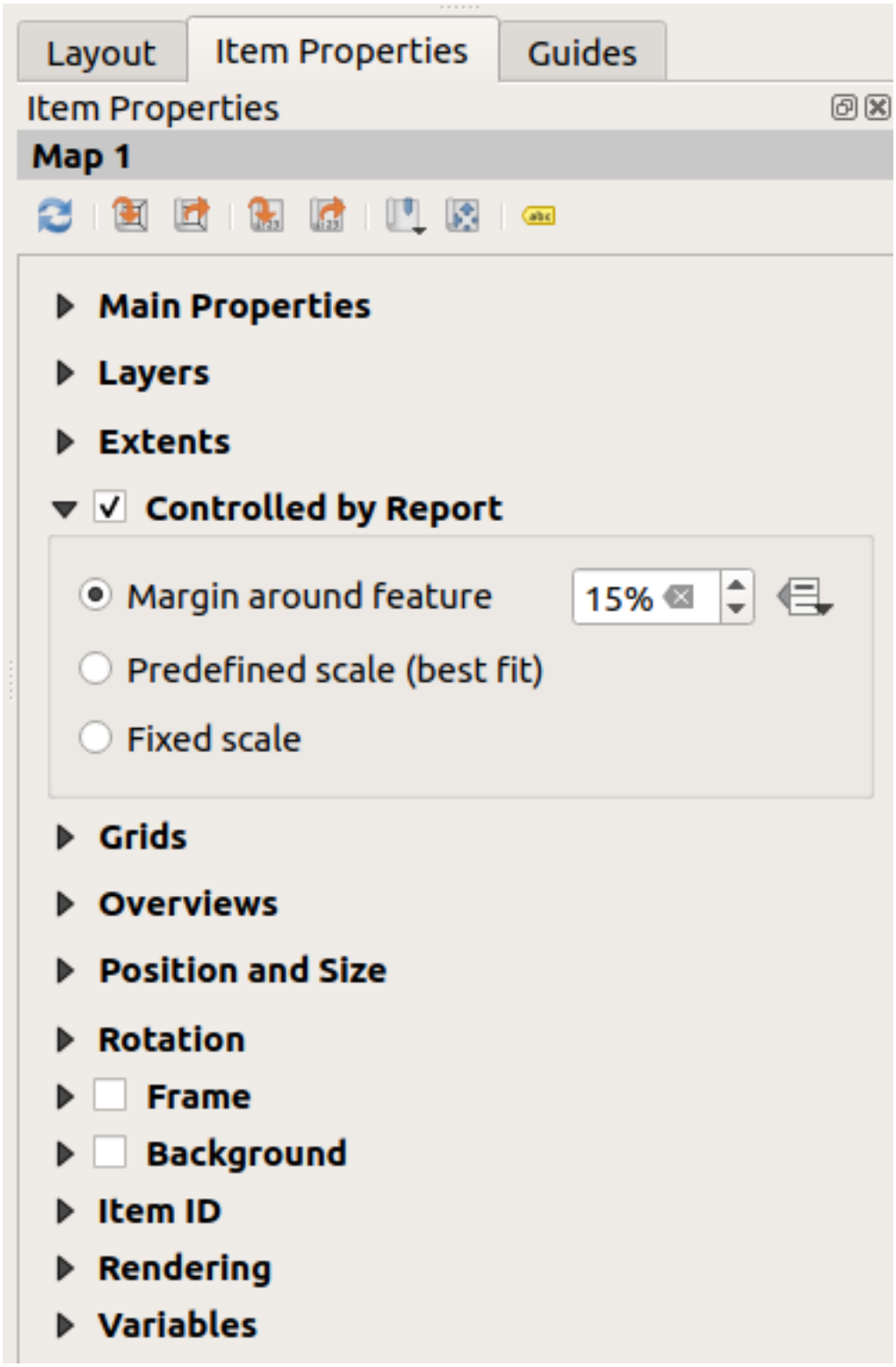


Our body now consists of a map and a label showing the name of the state. To include the name of the state, we selected *Add Item* ► *Add Label* and data defined the text under *Main Properties* with the help of *Insert or Edit an Expression*….

The result was the following expression (*name* is the name of the attribute in the *Admin Level 1* layer that contains the name of the state):

```
[% "name" %]
```

The map is set to follow the current report feature (enabled by checking *Controlled by Report* – just like a map item in an atlas will follow the current atlas feature when *Controlled by Atlas* is checked):



If we went ahead and exported our report now, we'd get something like this:

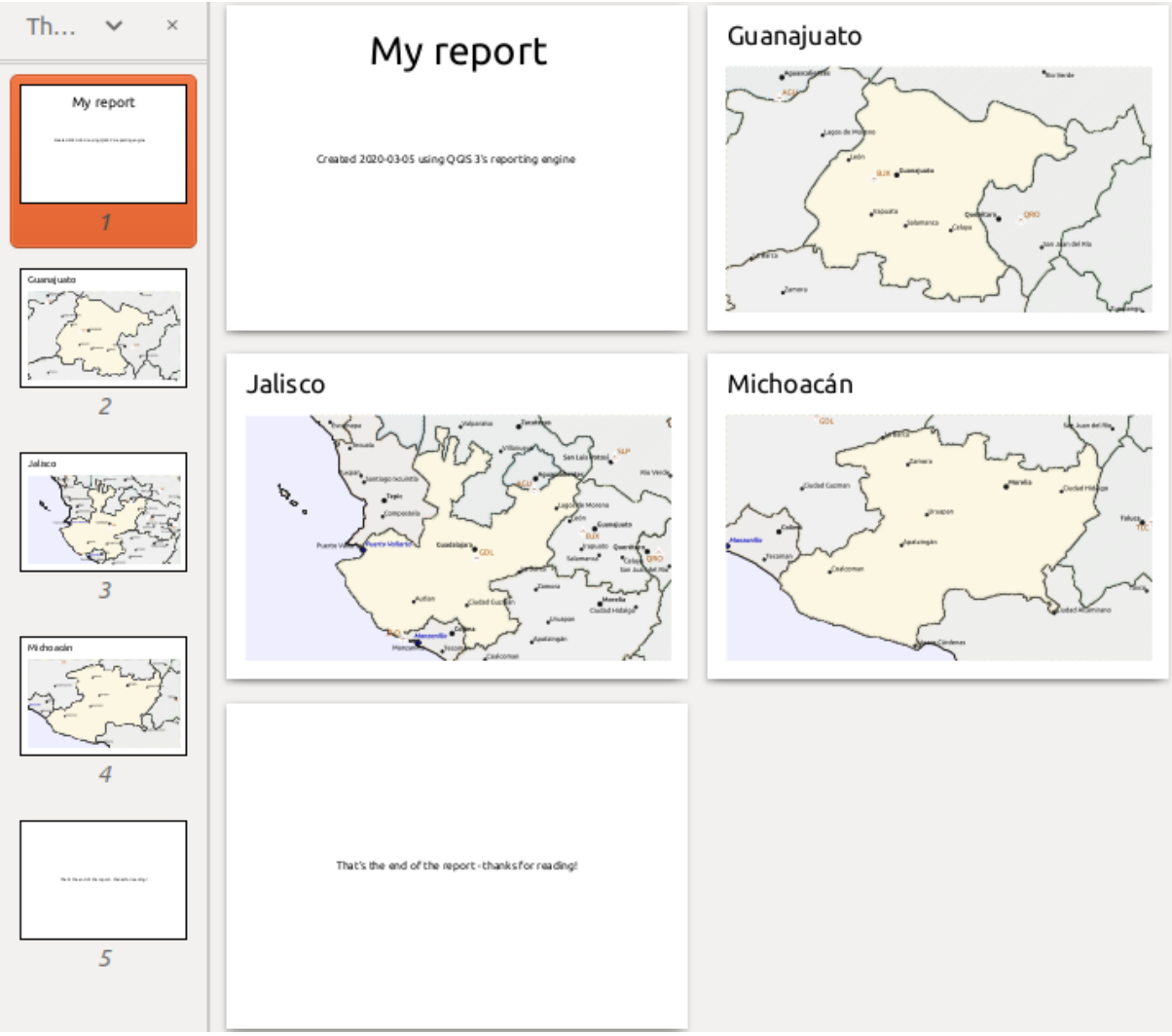
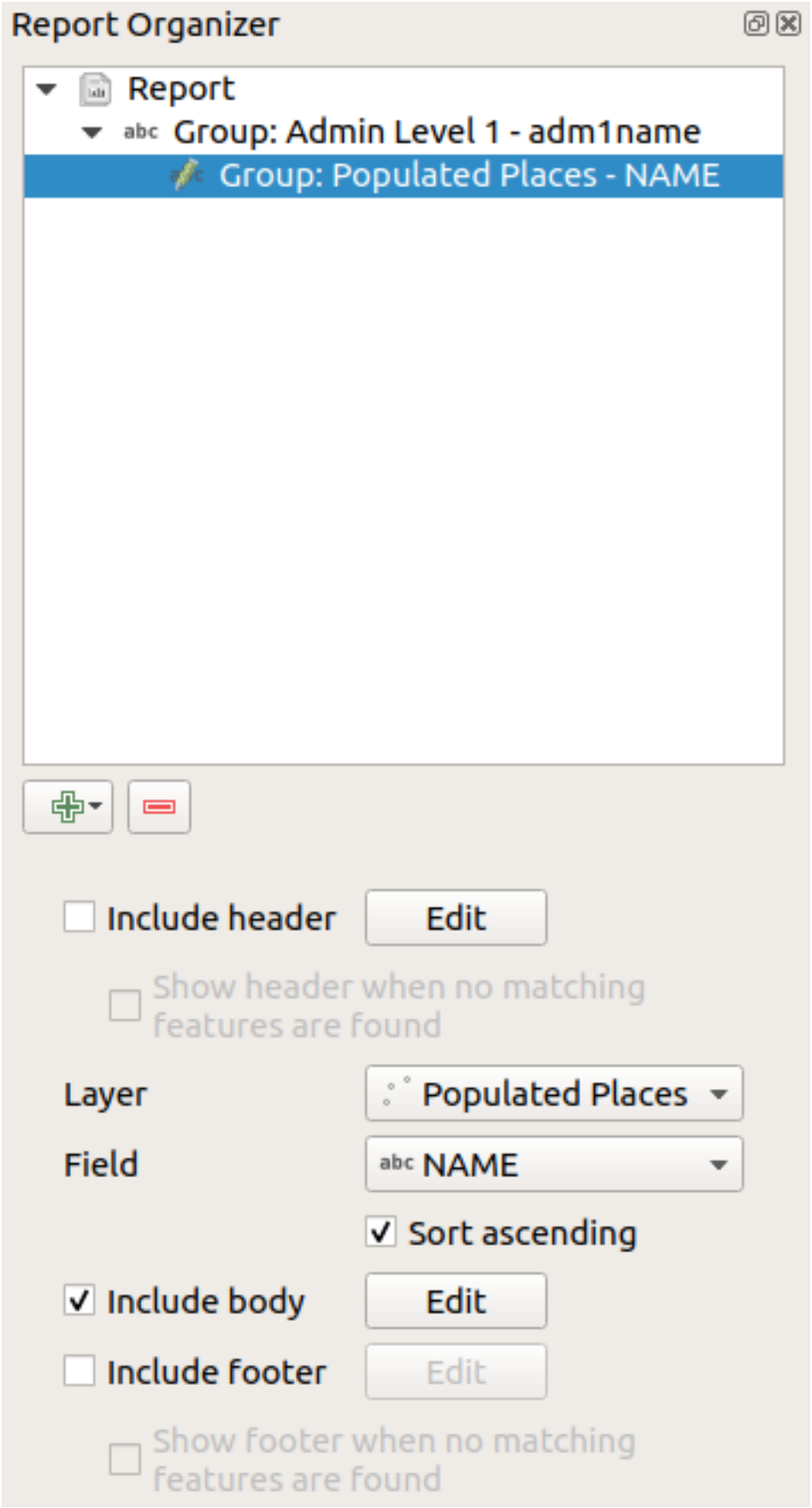


Fig. 22.68: The report header, a page for each state, and the report footer.

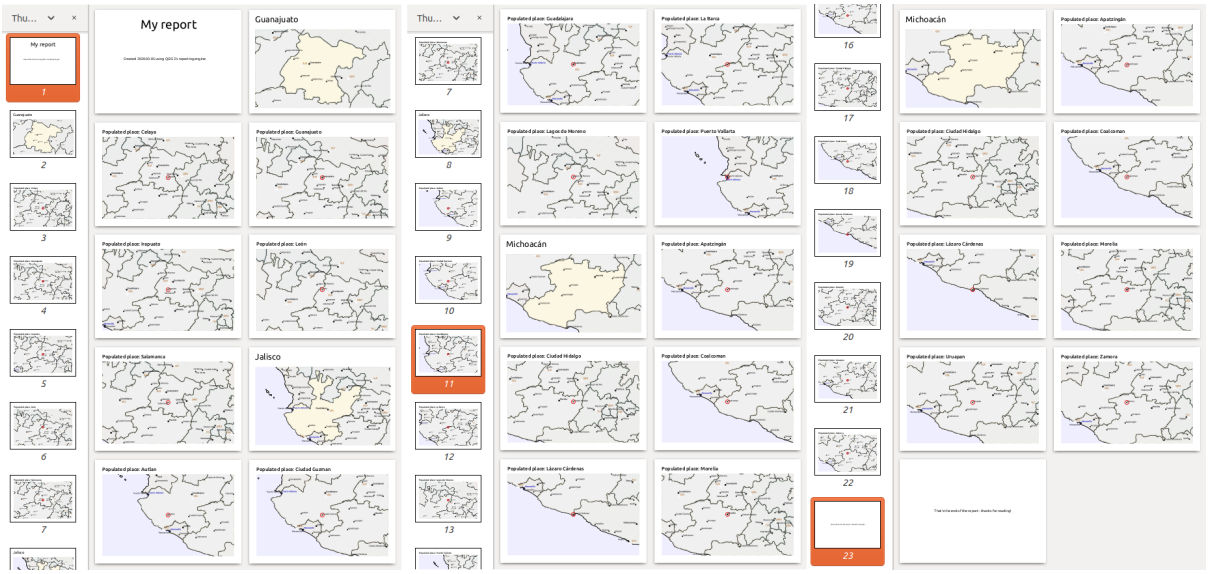
So more or less an atlas, but with a header and footer page.

Let us make things more interesting by adding a subsection to our state group. We do this by first selecting the *Admin Level 1* field group in the organizer, then hitting the Add Field button and adding a new *Field Group Section*:



When iterating over the features of a *Field Group Section*, the features will be filtered to match the defining field

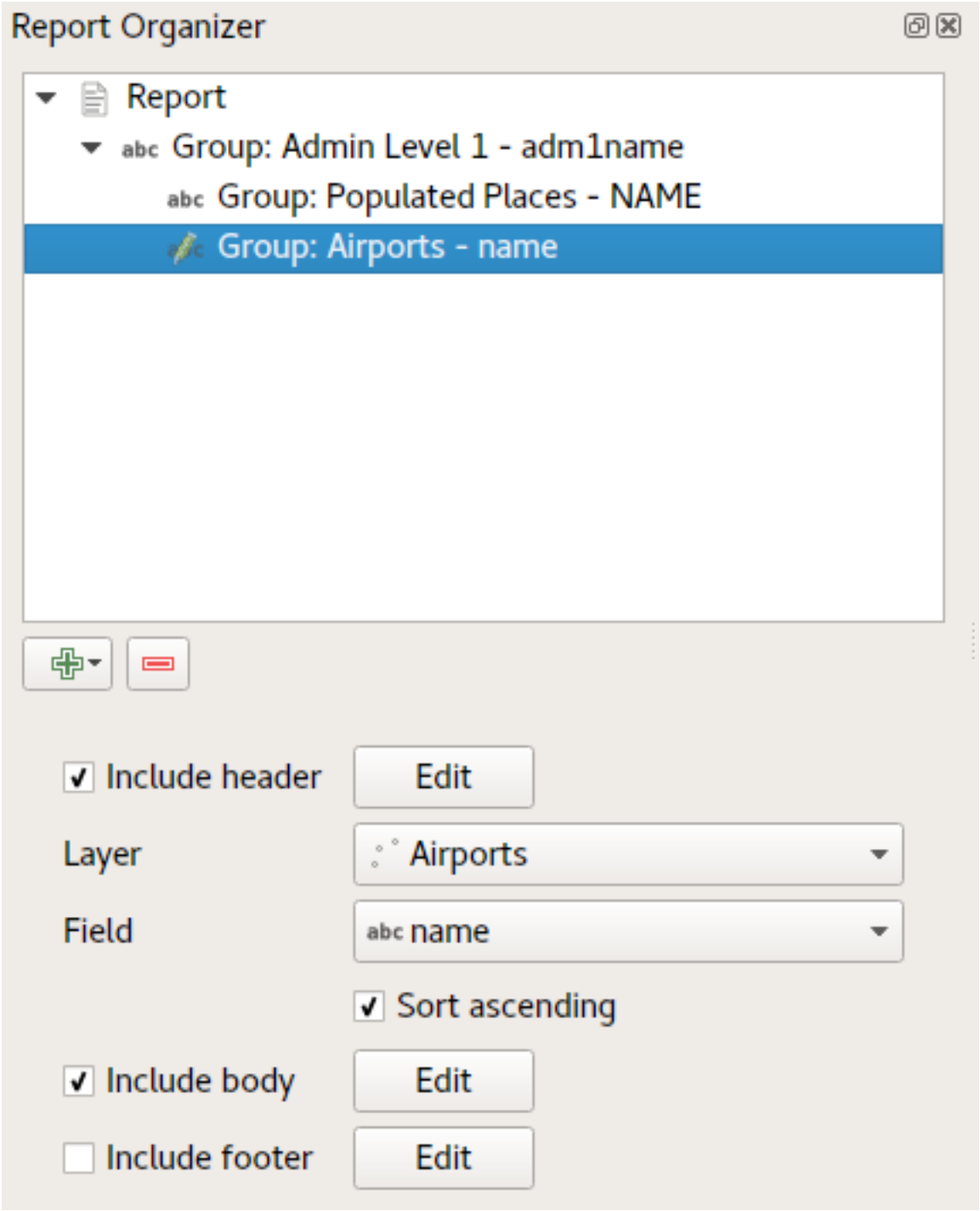
of its parent group (`adm1name` in this case). Here, the subsection we added will iterate over a *Populated Places* layer, including a body section for each place encountered. The magic here is that the *Populated Places* layer has an attribute with the same name as the defining field in the parent layer, *adm1name*, tagging each place with the state it is contained within (if you're lucky your data will already be structured like this – if not, run the *Join Attributes by Location* Processing algorithm and create your own field). When we export this report, QGIS will grab the first state from the *Admin Level 1* layer, and then iterate over all the *Populated Places* with a matching *adm1name* value. Here's what we get:



Here we created a basic body for the Populated Places group, including a map of the place and a table of some place attributes. So our report is now a report header, a page for the first state, followed by a page for every populated place within that state, then the rest of the states with their populated places, and finally the report footer. If we were to add a header for the Populated Places group, it would be included just before listing the populated places for each state, as shown in the illustration below.

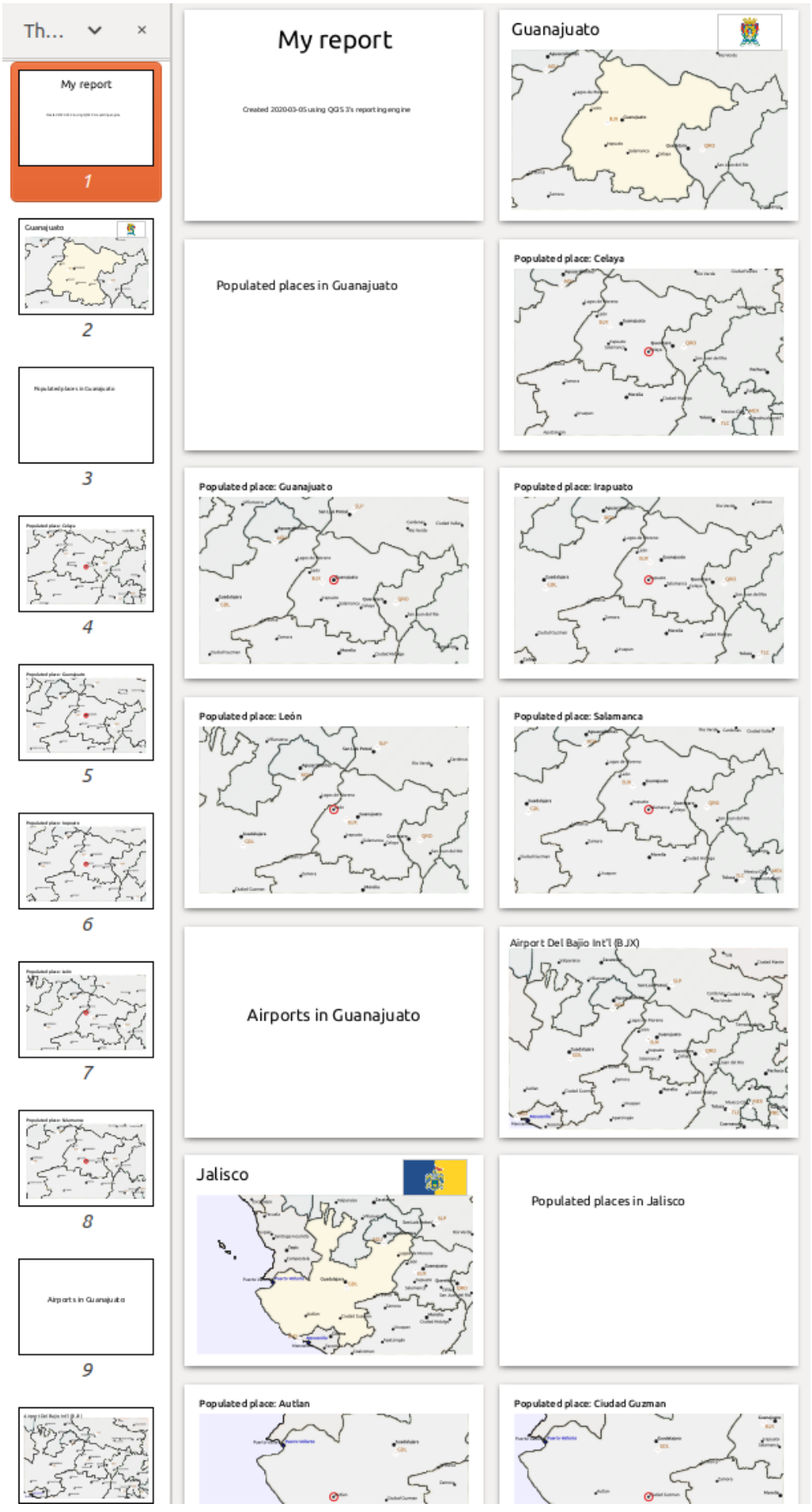
Similarly, a footer for the Populated Places group would be inserted after the final place for each state is included.

In addition to nested subsections, subsections in a report can also be included consecutively. If we add a second subsection to the *Admin Level 1 group* for *Airports*, then (if the *Airports* layer has an attribute *adm1name* that can link it to the parent group) our report will first list ALL the populated places for each state, followed by all the airports within that state, before proceeding to the next state.



The key point here is that our *Airports group* is a subsection of the *Admin Level 1 group* – not the *Populated Places group*.

In this case our report would be structured like this (note that state flags have also been included - the procedure for adding feature specific pictures in this way is described below):


Th...
My report
1
2
3
4
5
6
7
8
9
My report
Created 2020-03-05 using QGIS 3's reporting engine
Guanajuato
Populated places in Guanajuato
Populated place: Celaya
Populated place: Guanajuato
Populated place: Irapuato
Populated place: León
Populated place: Salamanca
Airports in Guanajuato
Airport Del Bajio Int'l (BJX)
Jalisco
Populated places in Jalisco
Populated place: Autlan
Populated place: Ciudad Guzman

Including pictures in a report

Pictures can be quite useful in reports, and QGIS allows pictures in both the static and dynamic parts of a report. Pictures are added in the same way as for standard print layouts, and for the static report parts (and static pictures in dynamic parts) there is not more to it.

But if you want illustrations that are tailored to the report features, your layer must have an attribute that can be used to define the picture to include.

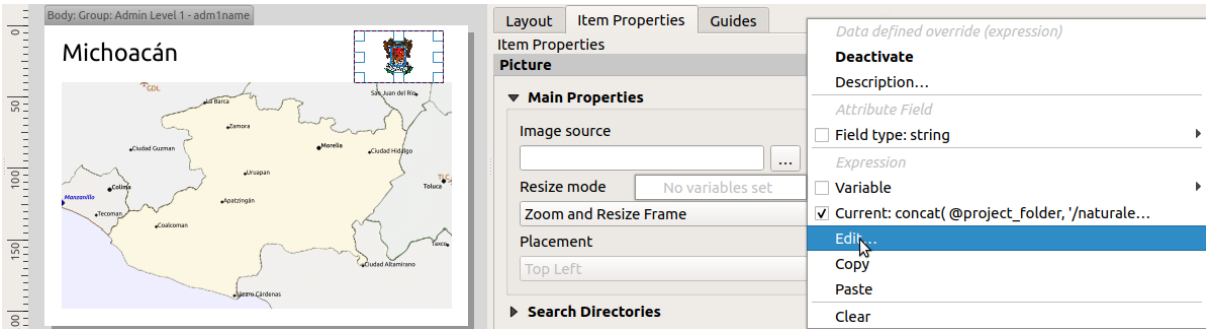
QGIS depends on absolute file names for images in reports.

For dynamic pictures, you first add a picture to the body part of the group, as usual. In the *Item properties* of the picture, you set the *Image Source* using the Data defined override button, and either select an attribute that contains the absolute path of the images or *Edit...* (to enter an expression that generates the absolute image path).

Below is an example expression that uses string concatenation to specify the absolute path to the pictures, using the directory where the project file is located (`@project_path`) and an attribute (`adm1name`) from which the file name is generated (in this case by transforming the string in the `adm1name` attribute to uppercase, and appending «_flag.png»):

```
concat(@project_folder, '/naturalearth/pictures/' ,
       upper("adm1name"), '_flag.png')
```

This means that the pictures are located in the `naturalearth/pictures` subdirectory of the project file directory.



Highlighting the current report feature in a map

In the above report, the report features are emphasized in the maps using highlighting (state) and circles (populated places). To emphasize the report features in the maps (apart from placing them at the centre of the maps), you must data define the style using a comparison between its `@id` and the `@atlas_featureid`, as for atlases.

For instance, if you would like to use a thicker line / border for the report feature than the other features you can data define the line width:

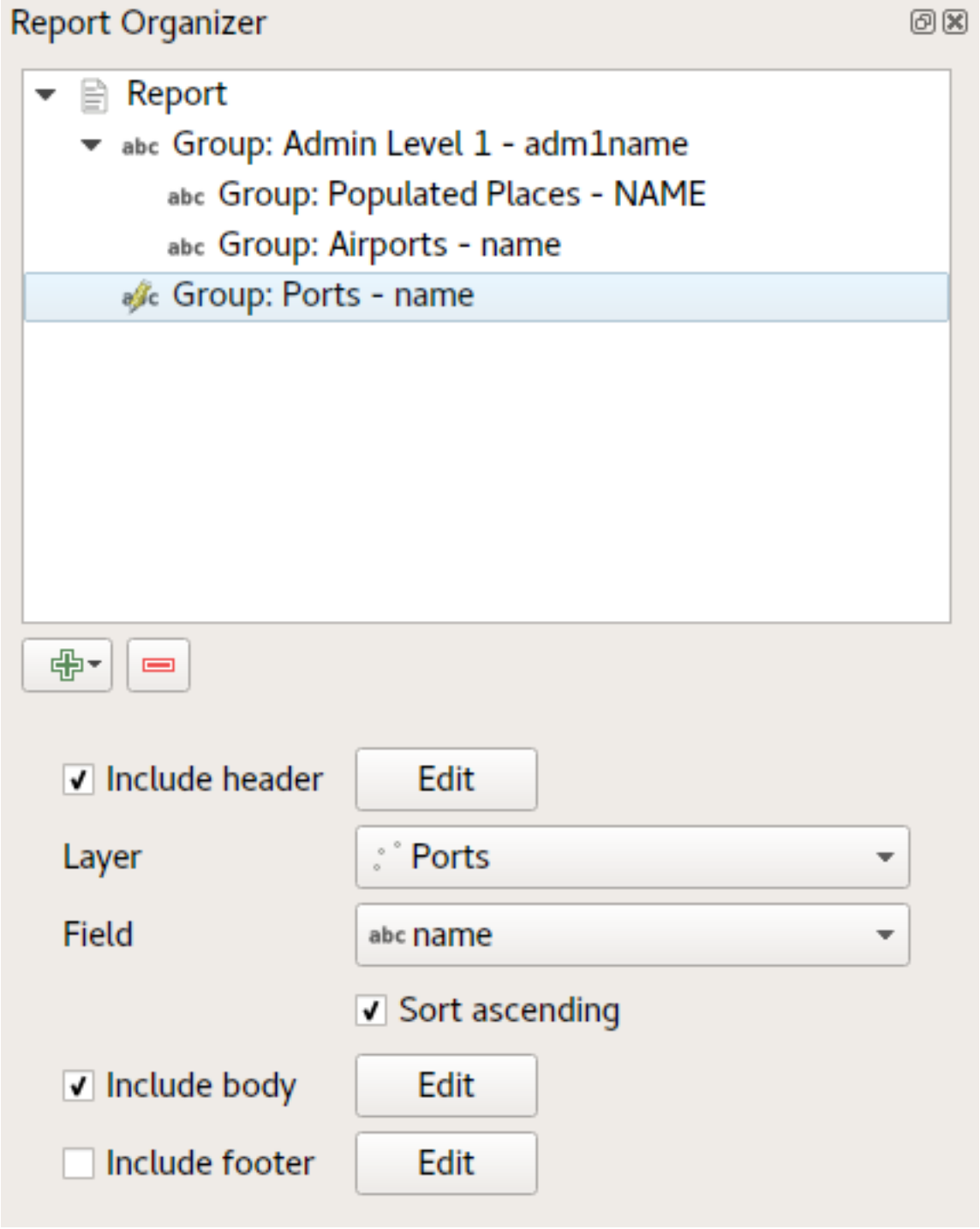
```
if($id=@atlas_featureid, 2.0, 0.1)
```

The report feature will get a 2 units wide polygon outline, while all other features will get a 0.1 units wide line. It is also possible to data define the colour (non-transparent dark magenta for the report feature and semi-transparent light gray for the other features):

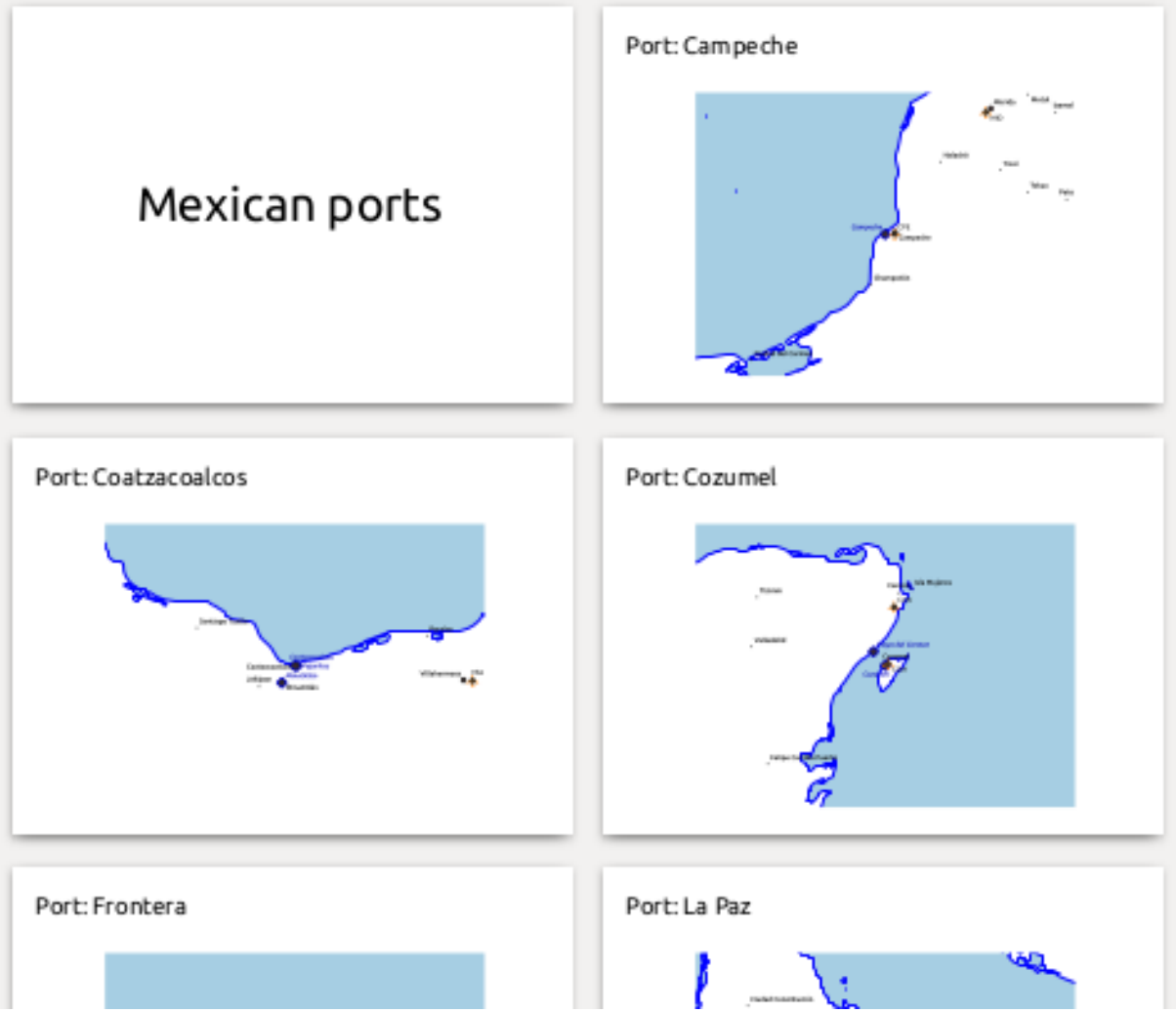
```
if($id=@atlas_featureid, '#FF880088', '#88CCCCCC')
```

More level 1 groups

Combining nested and consecutive sections, together with section headers and footers allows for tons of flexibility. For instance, in the below report we add another field group as a child of the main report for the :guilabel`Ports` layer. Now, after listing the states together with their populated places and airports, we'll get a summary list of all the ports in the region:



This results in the last part of our report exporting as:



22.4.4 Setări de export

When you export a report (*Report* ► *Export Report as Images… / SVG… / PDF…*), you will be asked for a file name, and then you get the opportunity to tune the export settings to get the most appropriate output.

As you see, reports in QGIS are extremely powerful and flexible!

Notă: The current information was adapted from a North Road blog, Exploring Reports in QGIS 3.0 - the Ultimate Guide!

CAPITOLUL 23

Working with OGC / ISO protocols

Open Geospatial Consortium (OGC) este o organizatie internațională, având ca membri mai mult de 300 de organizații comerciale, guvernamentale, non-profit și de cercetare din întreaga lume. Membrii săi dezvoltă și implementează standarde și servicii pentru conținut geospațial, de prelucrare și de schimb a datelor GIS.

Describing a basic data model for geographic features, an increasing number of specifications are developed by OGC to serve specific needs for interoperable location and geospatial technology, including GIS. Further information can be found at https://www.ogc.org/.

Specificațiile OGC importante, acceptate de către QGIS sunt:

- **WMS** — Web Map Service (*Client WMS/WMTS*)
- **WMTS** — Web Map Tile Service (*Client WMS/WMTS*)
- **WFS** — Web Feature Service (*Client WFS și WFS-T*)
- **WFS-T** — Web Feature Service - Transactional (*Client WFS și WFS-T*)
- **WCS** — Web Coverage Service (*Client WCS*)
- **WPS** — Web Processing Service
- **CSW** — Serviciu de Catalogare pentru Web
- **SFS** — Simple Features for SQL (*Straturi PostGIS*)
- **GML** — Limbaj cu Marcaje Geografice

Serviciile OGC sunt tot mai mult utilizate pentru a interschimba datele geospațiale între diferitele implementări GIS, sau între depozitele de date. QGIS are de-a face cu specificațiile de mai sus din punctul de vedere al unui client, fiind **SFS** (prin sprijinirea furnizorului de date PostgreSQL / PostGIS, v. secțiunea *Straturi PostGIS*).

You can also share your maps and data through the WMS, WMTS, WFS, WFS-T and WCS protocols using a webserver with QGIS Server, UMN MapServer or GeoServer installed.

23.1 Client WMS/WMTS

23.1.1 Privire de ansamblu asupra suportului WMS

În prezent, QGIS poate acționa ca un client WMS care înțelege serverele WMS 1.1, 1.1.1 și 1.3. În special, acesta a fost testat cu servere accesibile publicului, cum ar fi DEMIS.

Un server WMS acționează asupra cererilor din partea clientului (cum ar fi QGIS) pentru o hartă raster cu o anumită extindere, set de straturi, stil de simbolizare, și transparență. Serverul WMS își consultă apoi sursele de date locale, rasterizează harta, și o trimite înapoi la client, într-un format raster. Pentru QGIS, acest format va fi, de obicei, JPEG sau PNG.

WMS este mai degrabă un serviciu generic REST (Transfer de Stare de Reprezentare), decât un serviciu Web matur. De fapt, puteți lua ca atare adresele URL generate de QGIS și să le utilizați într-un browser web, pentru a obține aceleași imagini pe care QGIS le folosește intern. Acest lucru poate fi util pentru depanare, deoarece există mai multe mărci de servere WMS pe piață, și toate au propria interpretare a standardului WMS.

Straturile WMS pot fi adăugate pur și simplu, atât timp cât: știți URL-ul de acces la serverul WMS, ați stabilit o conexiune validă la acel server, iar serverul înțelege ptotocolul HTTP, ca mecanism de transport al datelor.

Additionally, QGIS will cache your WMS responses (i.e. images) for 24h as long as the GetCapabilities request is not triggered. The GetCapabilities request is triggered every time the *Connect* button in the *WMS/WMTS* dialog is used to retrieve the WMS server capabilities. This is an automatic feature meant to optimize project loading time. If a project is saved with a WMS layer, the corresponding WMS tiles will be loaded from the cache the next time the project is opened as long as they are not older than 24h.

23.1.2 Privire de ansamblu asupra suportului WMTS

QGIS poate acționa, de asemenea, ca un client WMTS. WMTS reprezintă un standard OGC pentru distribuirea de seturi de plăcuțe ale datelor geospațiale. Acesta constituie un mod mai rapid și mai eficient de a distribui date, comparativ cu WMS, deoarece în cazul WMTS seturile de plăcuțe sunt pre-generate, iar clientul solicită numai transmiterea lor, nu și producerea acestora. O cerere WMS implică, de regulă, atât generarea cât și transmiterea datelor. Un exemplu bine-cunoscut al unui standard de bază non-OGC, pentru vizualizarea de date geospațiale cu ajutorul plăcuțelor, este Google Maps.

Pentru a afișa datele la o varietate de scări, apropiate de ceea ce și-ar putea dori utilizatorul, seturile de plăcuțe WMTS sunt produse la mai multe niveluri de scări diferite, și sunt puse la dispoziția clientului GIS pentru a le apela.

Această diagramă ilustrează conceptul seturilor de plăcuțe:

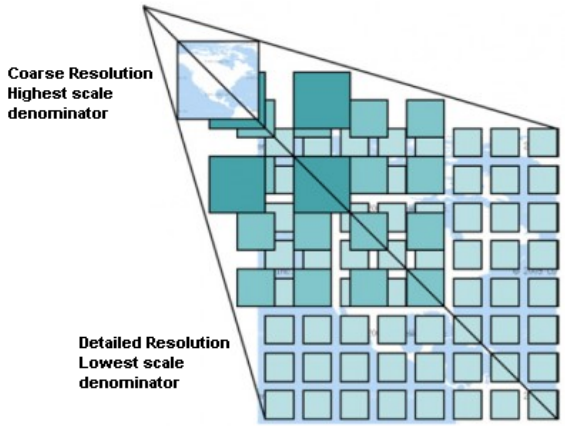


Fig. 23.1: Conceptul seturilor de plăcuțe WMTS:

Cele două tipuri de interfețe WMTS, pe care QGIS le acceptă, sunt Key-Value-Pairs (KVP) și REST. Aceste două interfețe sunt diferite, și trebuie să le specificați în QGIS în mod diferit.

1. Pentru a avea acces la un serviciu **WMTS KVP**, un utilizator QGIS trebuie să deschidă interfața WMS/WMTS și să adăuge următorul șir la adresa URL a serviciului de plăcuțe WMTS:

   ```
   "?SERVICE=WMTS&REQUEST=GetCapabilities"
   ```

   An example of this type of address is:

   ```
   https://opencache.statkart.no/gatekeeper/gk/gk.open_wmts?service=WMTS&
   ↪request=GetCapabilities
   ```

   Pentru testare, stratul topo2 din acest WMTS funcționează bine. Adăugarea acestui șir indică faptul că se va utiliza un serviciu web WMTS în locul unui serviciu WMS.

2. Serviciul **RESTful WMTS** are o formă diferită de adresă URL, simplă. Formatul recomandat de OGC este:

   ```
   {WMTSBaseURL}/1.0.0/WMTSCapabilities.xml
   ```

   This format helps you to recognize that it is a RESTful address. A RESTful WMTS is accessed in QGIS by simply adding its address in the WMS setup in the URL field of the form. An example of this type of address for the case of an Austrian basemap is:

   ```
   https://maps.wien.gv.at/basemap/1.0.0/WMTSCapabilities.xml
   ```

Notă: You can still find some old services called WMS-C. These services are quite similar to WMTS (i.e., same purpose but working a little bit differently). You can manage them the same as you do WMTS services. Just add `?tiled=true` at the end of the url. See https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification for more information about this specification.

Când citiți WMTS, vă puteți gândi adesea și la WMS-C.

23.1.3 Selectarea serverelor WMS/WMTS

The first time you use the WMS/WMTS feature in QGIS, there are no servers defined.

You then need to create connections to the server you are targeting:

1. Go to the *WMS/WMTS* tab of the *Data Source Manager* dialog, either by:
   - clicking the Open Data Source Manager button (or pressing `Ctrl+L`) and enabling the tab
   - clicking the Add WMS layer button on the *Manage Layers* toolbar
   - or selecting *Layer* ► *Add Layer* ► *Add WMS/WMTS Layer…* menu
2. Press *New* from the *Layers* tab. The *Create a New WMS/WMTS Connection…* dialog appears.

   Sfat: Right-click the *WMS/WMTS* entry from within the *Browser panel* and select *New Connection…* also opens the *Create a New WMS/WMTS Connection…* dialog.

   ---

3. Then enter the parameters to connect to your desired WMS server, as listed below:

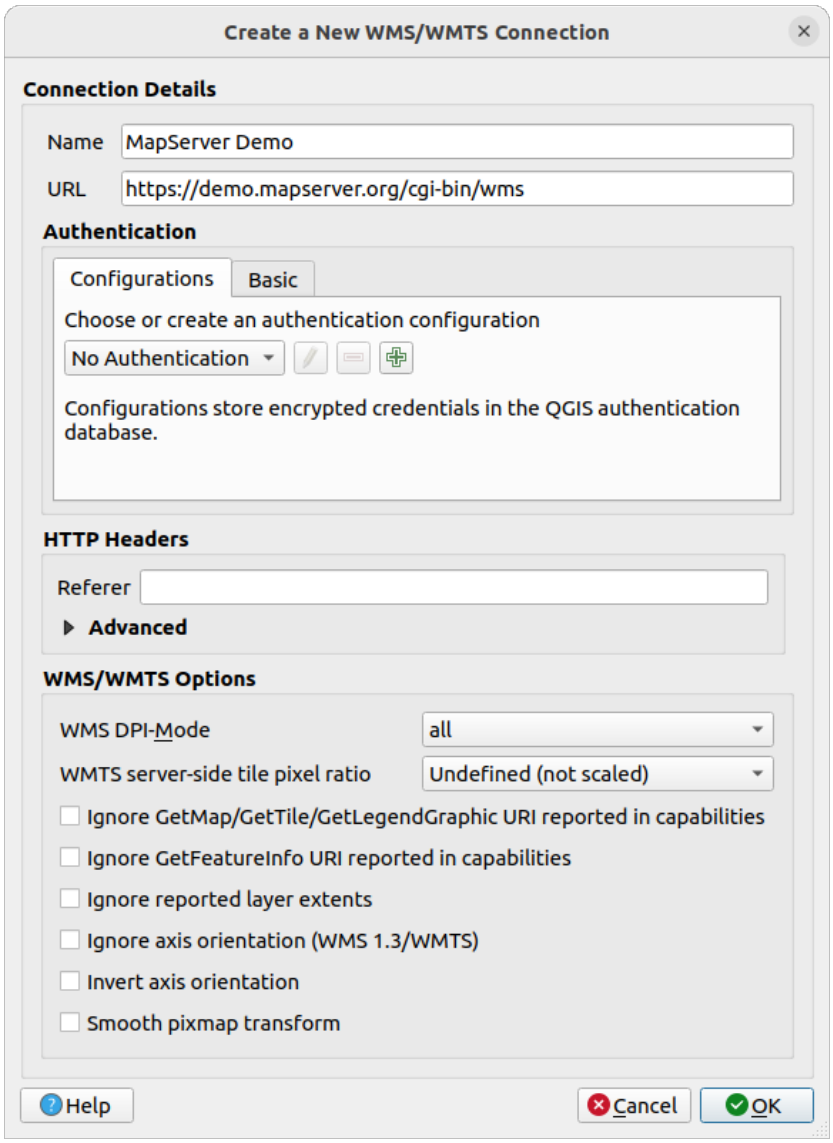


Fig. 23.2: Creating a connection to a WMS server

- *Name*: A name for the connection. This name will be used in the Server Connections drop-down box so that you can distinguish it from other WMS servers.
- *URL*: URL of the server providing the data. This must be a resolvable host name – the same format as you would use to open a telnet connection or ping a host, i.e. the base URL only. For example, you shouldn't have fragments such as `request=GetCapabilities` or `version=1.0.0` in your URL.
- *Authentication* (optional): using a *stored configuration* or a basic authentication with *Username* and *Password*.

> **Atenționare:** Entering **username** and **password** in the *Authentication* tab will keep unprotected credentials in the connection configuration. Those **credentials will be visible** if, for instance, you shared the project file with someone. Therefore, it's advisable to save your credentials in an *Authentication configuration* instead (*Configurations* tab). See *Sistem de Autentificare* for more details.

- HTTP *Referer*
- *WMS DPI-Mode*: Available options are **all**, **off**, **QGIS**, **UMN** and **GeoServer**
- *WMTS server-side tile pixel ratio*: When rendering WMTS layers, allows to scale up or down the tiles

based on the device screen DPI. Available options are *Undefined (not scaled)*, *Standard (96 DPI)* and *High (192 DPI)*.

- *Ignore GetMap/GetTile/GetLegendGraphic URI reported in capabilities*: if checked, use given URI from the *URL* field above.
- *Ignore GetFeatureInfo URI reported in capabilities*: if checked, use given URI from the *URL* field above.
- *Ignore reported layer extents*: because the extent reported by raster layers may be smaller than the actual area which can be rendered (notably for WMS servers with symbology which takes more space than the data extent), check this option to avoid cropping raster layers to their reported extents, resulting in truncated symbols on the borders of these layers.
- *Ignore axis orientation (WMS 1.3/WMTS)*
- *Invert axis orientation*
- *Smooth pixmap transformation*

4. Apăsați pe *OK*

Once the new WMS/WMTS server connection has been created, it will be preserved for future QGIS sessions. Note that it is also possible to *Load* the connection parameters from a `.XML` file or *Save* them to a `.XML` file.

If you need to set up a proxy server to be able to receive WMS services from the internet, you can add your proxy server in the options. Choose *Settings* ► *Options* and click on the *Network* tab. There, you can add your proxy settings and enable them by setting *Use proxy for web access*. Make sure that you select the correct proxy type from the *Proxy type* drop-down menu.

23.1.4 Încărcarea Straturilor WMS/WMTS

Once you have successfully filled in your parameters, you can use the *Connect* button to retrieve the capabilities of the selected server. This includes the image encoding, layers, layer styles and projections. Since this is a network operation, the speed of the response depends on the quality of your network connection to the WMS server. While downloading data from the WMS server, the download progress is visualized in the lower left corner of the main QGIS dialog.

Your screen should now look a bit like Fig. 23.3, which shows the response provided by a WMS server.

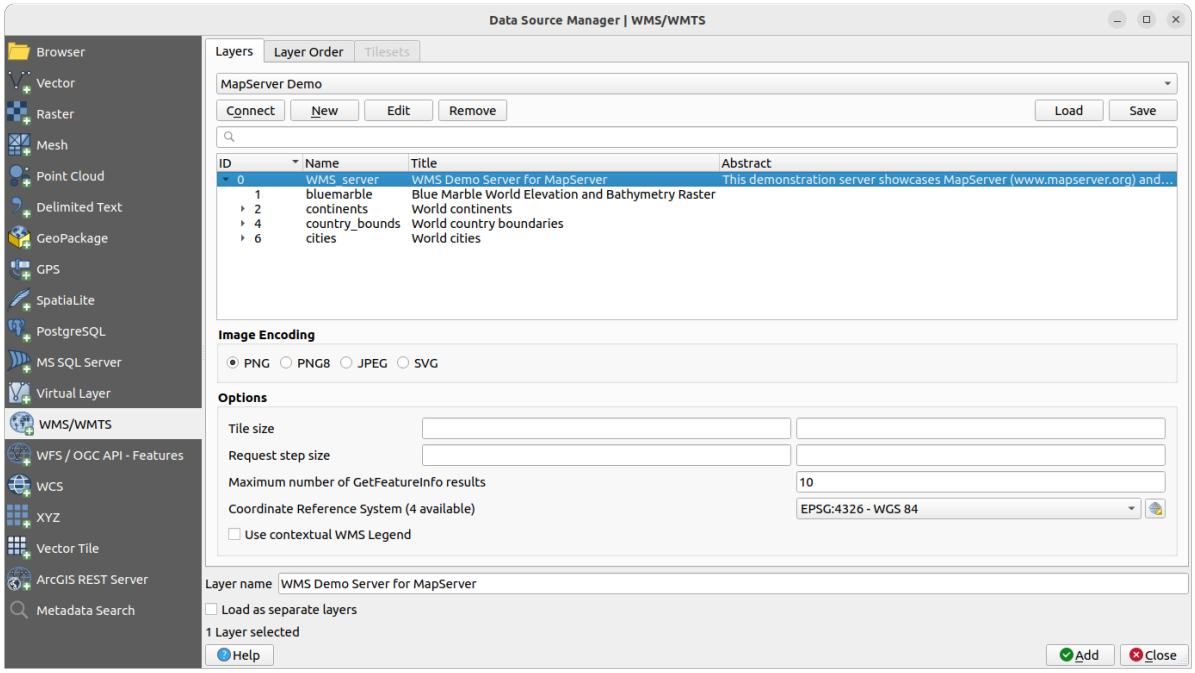


Fig. 23.3: Dialog for adding a WMS server, with filter on available layers

The upper part of the *Layers* tab of the dialog shows a tree structure that can include layer groups embedding layers with their associated image style(s) served by the server. Each item can be identified by:

- an *ID*
- a *Name*
- a *Title*
- and an *Abstract*.

The list can be filtered using the widget in the top right corner.

Codificare Imagine

The *Image encoding* section lists the formats that are supported by both the client and server. Choose one depending on your image accuracy requirements.

Sfat: Codificare Imagine

Veți descoperi că, de obicei, un server WMS vă oferă posibilitatea de a alege codificarea imaginii sub formă de JPEG sau PNG. JPEG este un format de compresie cu pierderi, în timp ce PNG reproduce fidel datele raster originale.

Use JPEG if you expect the WMS data to be photographic in nature and/or you don't mind some loss in picture quality. This trade-off typically reduces by five times the data transfer requirement compared with PNG.

Utilizați PNG dacă doriți reprezentări precise ale datelor originale, și nu vă deranjează cerințele crescute de transfer de date.

Opțiuni

The Options area of the dialog provides means to configure the WMS requests. You can define:

- *Tile size* if you want to set tile sizes (e.g., 256x256) to split up the WMS request into multiple requests.
- *Request step size*: if you want to reduce the effect of cut labels at tile borders, increasing the step size creates larger requests, fewer tiles and fewer borders. The default value is 2000.
- The *Maximum number of GetFeatureInfo results* from the server

- Each WMS layer can be presented in multiple CRSs, depending on the capability of the WMS server. If you select a WMS from the list, a field with the default projection provided by the web server appears. Press the Select CRS widget to replace the default projection of the WMS with another CRS supported by the WMS server.

  A dialog similar to the one shown in Fig. 10.4 will appear. The main difference with the WMS version of the dialog is that only those CRSs supported by the WMS server will be shown.

- Finally you can activate *Use contextual WMS Legend* if the WMS Server supports this feature. Then only the relevant legend for your current map view extent will be shown and thus will not include legend items for items you can't see in the current map.

At the bottom of the dialog, a *Layer name* text field displays the selected item's *Title*. You can change the name at your will. This name will appear in the *Layers* panel after you pressed the *Add* button and loaded the layer(s) in QGIS.

You can select several layers at once, but only one image style per layer. When several layers are selected, they will be combined at the WMS server and transmitted to QGIS in one go, as a single layer. The default name is a slash (/) separated list of their original title. You can however opt to *Load as separate layers*.

Ordinea straturilor

The *Layer Order* tab lists the selected layers available from the current connected WMS server.

WMS layers rendered by a server are overlaid in the order listed in the *Layers* tab, from top to bottom of the list. If you want to change the overlay order, you can use the *Up* and *Down* buttons of the *Layer Order* tab.

Transparenţă

The *Global transparency* setting from the *Layer Properties* is hard coded to be always on, where available.

23.1.5 Seturi de plăcuțe

When using WMTS (Cached WMS) services you are able to browse through the *Tilesets* tab given by the server. Additional information like tile size, formats and supported CRS are listed in this table.

```
# example of WMTS service

https://opencache.statkart.no/gatekeeper/gk/gk.open_wmts?service=WMTS&
↪request=GetCapabilities
```

Selecting a layer to load, it is also possible to apply an *Interpretation method*, converting it into a raster layer of single band float type, ready for styling using QGIS usual *raster renderers*.

In combination with this feature, you can use the tile scale slider by selecting *View* ► *Panels* (or *Settings* ► *Panels*), then choosing *Tile Scale Panel*. This gives you the available scales from the tile server with a nice slider docked in.

23.1.6 Folosirea instrumentului de identificare

După ce ați adăugat un server WMS, în cazul în care orice strat primit de la serverul WMS este interogabil, puteți folosi instrumentul Identificare pentru a selecta un pixel de pe canevasul hărții. O interogare a serverului WMS are loc pentru fiecare selecție efectuată. Rezultatele interogării sunt returnate sub formă de text simplu. Formatarea acestui text depinde de particularitățile serverului WMS folosit.

Selecţia Formatului

If multiple output formats are supported by the server, a combo box with supported formats is automatically added to the identify results dialog and the selected format may be stored in the project for the layer.

Suport pentru formatul GML

The Identify tool supports WMS server response (GetFeatureInfo) in GML format (it is called Feature in the QGIS GUI in this context). If „Feature" format is supported by the server and selected, results of the Identify tool are vector features, as from a regular vector layer. When a single feature is selected in the tree, it is highlighted in the map and it can be copied to the clipboard and pasted to another vector layer. See the example setup of the UMN Mapserver below to support GetFeatureInfo in GML format.

```
# in layer METADATA add which fields should be included and define geometry␣
↪(example):

"gml_include_items"   "all"
"ows_geometries"      "mygeom"
"ows_mygeom_type"     "polygon"

# Then there are two possibilities/formats available, see a) and b):

# a) basic (output is generated by Mapserver and does not contain XSD)
# in WEB METADATA define formats (example):
"wms_getfeatureinfo_formatlist" "application/vnd.ogc.gml,text/html"

# b) using OGR (output is generated by OGR, it is sent as multipart and contains␣
↪XSD)
# in MAP define OUTPUTFORMAT (example):
OUTPUTFORMAT
    NAME "OGRGML"
    MIMETYPE "ogr/gml"
    DRIVER "OGR/GML"
    FORMATOPTION "FORM=multipart"
END

# in WEB METADATA define formats (example):
"wms_getfeatureinfo_formatlist" "OGRGML,text/html"
```

23.1.7 Viewing Properties

Once you have added a WMS server, you can view its properties by right-clicking on it in the legend and selecting *Properties*. The WMS/WMTS layer properties is much like the raster layer properties so you will find detailed description at *Dialogul Proprietăților Rasterului*. However, there are some differences, which will be explained below.

Information properties

Fila Metadatelor

The tab *Metadata* displays a wealth of information about the WMS server, generally collected from the capabilities statement returned from that server. Many definitions can be gleaned by reading the WMS standards (see OPEN-GEOSPATIAL-CONSORTIUM in *Literatură și Referințe Web*), but here are a few handy definitions:

- **Proprietăţile serverului**
  - **Versiunea WMS** — Versiunea WMS acceptată de server.
  - **Formatele de Imagine** — Lista tipurilor MIME pentru care serverul poate răspunde la momentul desenării hărții. QGIS acceptă oricare dintre formatele recunoscute de bibliotecile Qt, acestea fiind de obicei `image/png` și `image/jpeg`.
  - **Identity Formats** — The list of MIME-types the server can respond with when you use the Identify tool. Currently, QGIS supports the `text-plain` type.
- **Proprietăţile stratului**
  - **Selectat** — Indiferent dacă acest strat a fost sau nu selectat, atunci când serverul său a fost adăugat în acest proiect.

- **Vizibil** — Indiferent dacă acest strat este selectat ca vizibil în legendă (nu este utilizat încă în această versiune de QGIS).
- **Poate Identifica** — Dacă acest strat va returna un rezultat, sau nu, atunci când este folosit instrumentul Identificare asupra lui.
- **Poate fi Transparent** — Dacă sau nu, un strat poate fi randat cu transparență. Această versiune de QGIS va folosi întotdeauna transparența dacă această setare este `Yes`, și dacă codificarea imaginii acceptă transparență.
- **Se Poate Mări** — Dacă acest strat poate fi, sau nu, mărit de către server. Această versiune de QGIS presupune că toate straturile WMS au această opțiune setată pe `Yes`. Straturile cu deficiențe pot fi randate în mod eronat.
- **Numărare în Cascadă** — Serverele WMS pot acționa ca un proxy pentru alte servere WMS, pentru a obține datele raster ale unui strat. Această intrare arată de câte ori este transmisă cererea pentru acest strat către alte servere WMS, pentru obținerea unui rezultat.
- **Lățime fixă, Înălțime Fixă** — Dacă acest strat are, sau nu, dimensiuni fixe. Această versiune de QGIS presupune că toate straturile WMS nu au această opțiune setată. Straturile cu deficiențe pot fi randate în mod eronat.
- **WGS 84 Bounding Box** — The bounding box of the layer, in WGS 84 coordinates. Some WMS servers do not set this correctly (e.g., UTM coordinates are used instead). If this is the case, then the initial view of this layer may be rendered with a very «zoomed-out» appearance by QGIS. The WMS webmaster should be informed of this error, which they may know as the WMS XML elements `LatLonBoundingBox`, `EX_GeographicBoundingBox` or the CRS:84 `BoundingBox`.
- **Disponibil în CRS** — Proiecțiile în care poate fi randat acest strat de către serverul WMS. Acestea sunt prezentate în format nativ WMS.
- **Disponibil în stil** — Stilurile de imagine în care poate fi randat acest strat de către serverul WMS.

Temporal properties

Raster *temporal properties* (namely *Dynamic Temporal Control*) can be set for WMS and WMTS layers. By default, when a time-dimension enabled WMS or WMTS layer is added to the project, it is indicated in the *Layers* panel with the Temporal Layer icon next to it. Its *Temporal* properties default to the *Automatic* temporal mode, meaning that the layer will follow the temporal controller's current time range by default.

You can then opt to show a specific static time value for the layer by unchecking *Dynamic Temporal Control* and picking an option under *Static WMS-T Temporal Range*:

- *Server default*
- *Predefined date* with a server exposing data for non-contiguous temporal ranges or *Predefined range* with a server exposing a range of available dates. A *Start date* and *End date* are necessary in the latter case. Their expected formatting can be deduced from the reference time option (see below). depending on whether the provider has data for contiguous period or not
- *Follow project's temporal range* as defined in the project's properties dialog

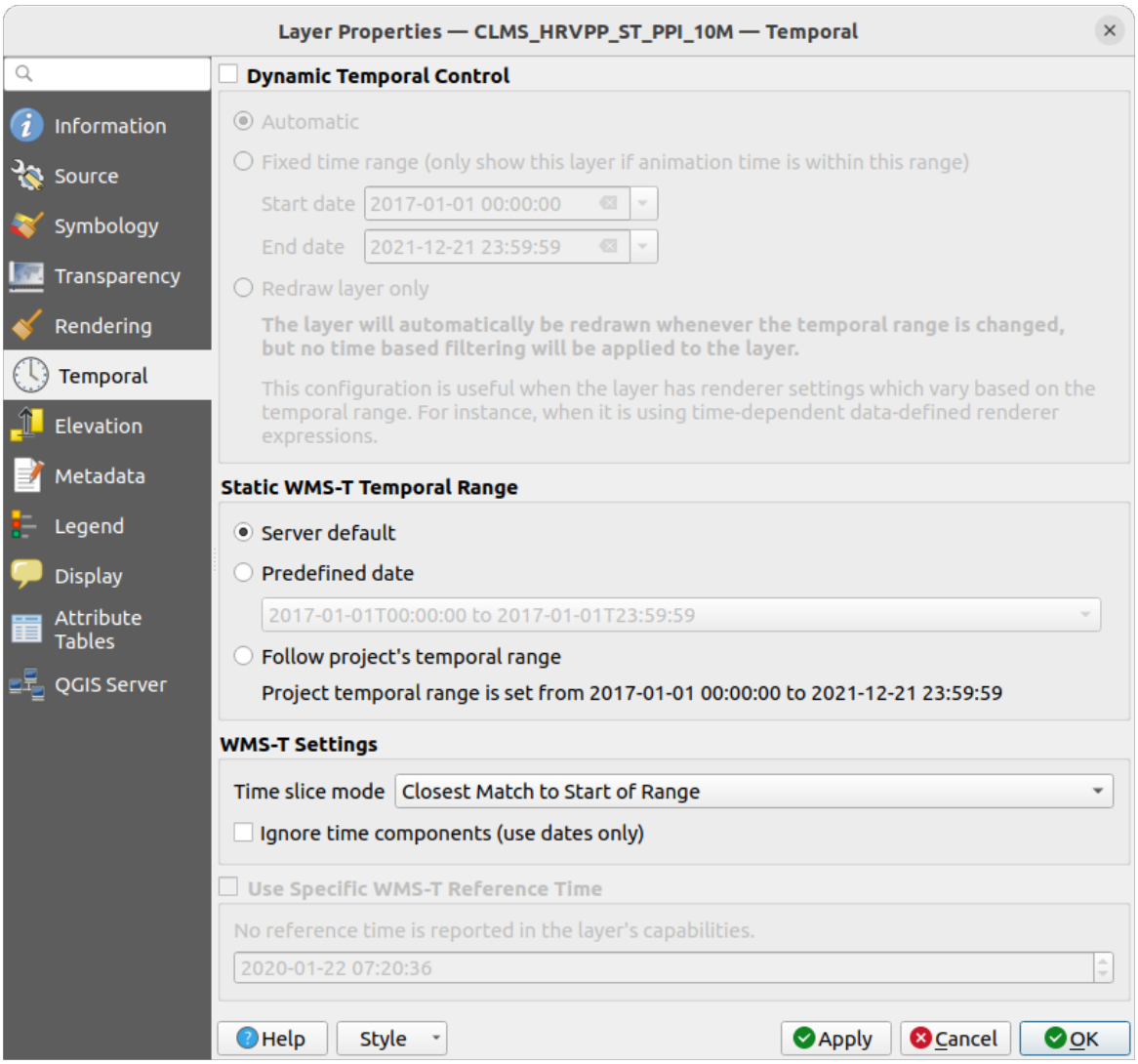


Fig. 23.4: Temporal properties of a WMTS layer

Whatever temporal data control is in use, there are some *WMS-T Settings* to help display the correct temporal data:

- *Time slice mode* which can be:
  - *Use whole temporal range*
  - *Match to start of range*
  - *Match to end of range*
  - *Closest match to start of range*
  - *Closest match to end of range*
- *Ignore time components (use dates only)*: If checked, the time component of temporal queries will be discarded and only the date component will be used in server requests.

You can also *Use Specific WMS-T Reference Time* picked from times reported in the layer's capabilities. Convenient for servers which expose a non-contiguous set of date time instances (instead of a range of dates).

QGIS Server properties

In addition to the raster layer *properties*, publishing a WMS/WMTS layer with QGIS Server will display following options:

- *WMS Print layer*: Allows to set alternative WMS layer that should be used for printing (in the GetProjectSettings reply). Convenient for WMTS layers that are generally not suitable for printing.
- ☑ *Publish WMS/WMTS data source uri*: Allows a web client to fetch the WMS/WMTS data directly
- ☑ *Advertise as background layer*

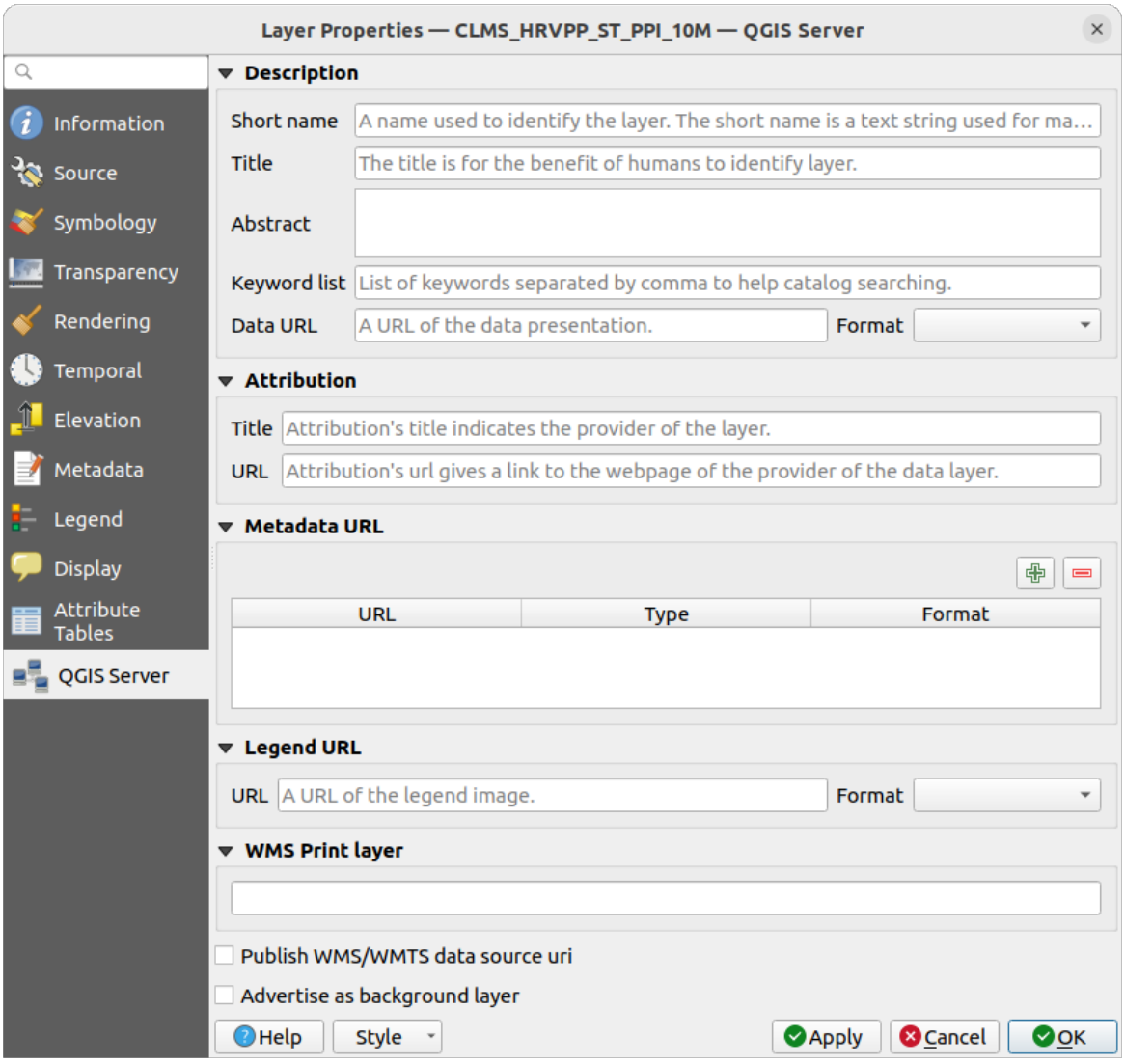


Fig. 23.5: QGIS Server properties of a WMS/WMTS layer

23.1.8 Show WMS legend graphic in table of contents and layout

The QGIS WMS data provider is able to display a legend graphic in the table of contents» layer list and in the print layout. The WMS legend will be shown only if the WMS server has GetLegendGraphic capability and the layer has getCapability url specified, so you additionally have to select a styling for the layer.

If a legendGraphic is available, it is shown below the layer. It is little and you have to click on it to open it in real dimension (due to QgsLegendInterface architectural limitation). Clicking on the layer's legend will open a frame with the legend at full resolution.

In the print layout, the legend will be integrated at it's original (downloaded) dimension. Resolution of the legend graphic can be set in the item properties under *Legend* ► *WMS LegendGraphic* to match your printing requirements.

The legend will display contextual information based on your current scale. The WMS legend will be shown only if the WMS server has GetLegendGraphic capability and the layer has getCapability url specified, so you have to select a styling.

23.2 Client WCS

Un Serviciu de Acoperire Web (WCS) oferă acces la datele raster, în forme care sunt utile pentru randarea pe partea clientului, ca date de intrare în modelele științifice, precum și pentru alți clienți. WCS poate fi comparat cu WFS și WMS. La fel ca și instanțele serviciilor WMS și WFS, un WCS permite clienților să aleagă porțiuni din informațiile serverelor, bazate pe constrângeri spațiale și pe alte criterii de interogare.

QGIS has a native WCS provider and supports both version 1.0 and 1.1 (which are significantly different), but currently it prefers 1.0, because 1.1 has many issues (i.e., each server implements it in a different way with various particularities).

Furnizorul nativ WCS se ocupă de toate cererile de rețea, utilizând toate setările de rețea standard ale QGIS (în special proxy). Este, de asemenea, posibil de selectat modul cache («întotdeauna cache», «se preferă cache», «se preferă rețeaua», «mereu rețea»), furnizorul acceptând, de asemenea, selecția poziției în timp, dacă domeniul temporal este oferit de server.

Atenționare: Entering **username** and **password** in the *Authentication* tab will keep unprotected credentials in the connection configuration. Those **credentials will be visible** if, for instance, you shared the project file with someone. Therefore, it's advisable to save your credentials in a *Authentication configuration* instead (*configurations* tab). See *Sistem de Autentificare* for more details.

23.3 Client WFS și WFS-T

In QGIS, a WFS layer behaves pretty much like any other vector layer. You can identify and select features, and view the attribute table. QGIS supports WFS 1.0.0, 1.1.0, 2.0 and OGC API - Features (OAPIF), including editing (through WFS-T). QGIS also supports background download and progressive rendering, on-disk caching of downloaded features and version autodetection.

Layers of servers implementing OGC API - Features - Part 4: Create, Replace, Update and Delete can be turned into editing mode to allow creating, modifying and deleting features. Note that each created/modified/deleted feature requires a dedicated network request, so performance might suffer in case of simultaneous modification of hundreds or more features at a time.

In general, adding a WFS layer is very similar to the procedure used with WMS. There are no default servers defined, so you have to add your own. You can find WFS servers by using the *MetaSearch plugin* or your favourite web search engine. There are a number of lists with public URLs, some of them maintained and some not.

Încărcarea unui strat WFS

As an example, we use the Gateway Geomatics WFS server and display a layer.

```
https://demo.gatewaygeomatics.com/cgi-bin/wfs_gateway?REQUEST=GetCapabilities&
↪VERSION=1.0.0&SERVICE=WFS
```

To be able to load a WFS Layer, first create a connection to the WFS server:

1. Open the *Data Source Manager* dialog by pressing the Open Data Source Manager button
2. Enable the *WFS / OGC API - Features* tab
3. Click on *New…* to open the *Create a New WFS Connection* dialog
4. Enter `Gateway Geomatics` as name
5. Enter the URL (see above)

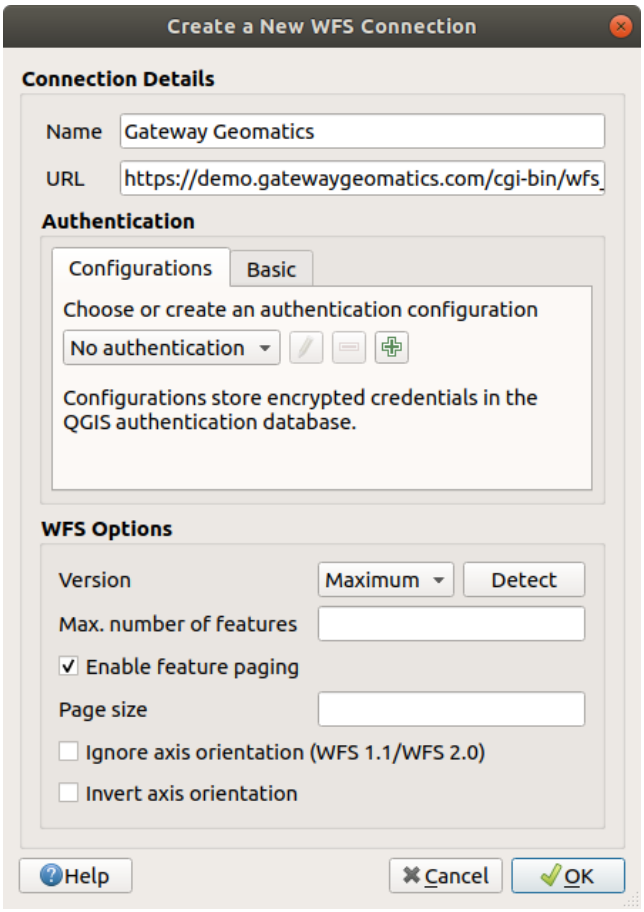


Fig. 23.6: Creating a connection to a WFS server

Notă: In case of an OGC API - Features (OAPIF), the URL to provide should be the landing page, ie the main page from which it is possible to navigate to all the available service endpoints.

6. In the WFS settings dialog, you can:
   - Indicate the WFS version of the server. If unknown, press the *Detect* button to automatically retrieve it.
   - Define the *maximum number of features* retrieved in a single GetFetFeature request. If empty, no limit is set.
   - And depending on the WFS version, indicate whether to:

- *Enable feature paging* and specify the maximum number of features to retrieve with *Page size*. If no limit is defined, then the server default is applied.
- Force to *Ignore axis orientation (WFS 1.1/WFS 2.0)*
- *Invert axis orientation*.
- *Use GML2 encoding for transactions*.

> **Atenționare:** Entering **username** and **password** in the *Authentication* tab will keep unprotected credentials in the connection configuration. Those **credentials will be visible** if, for instance, you shared the project file with someone. Therefore, it's advisable to save your credentials in an *Authentication configuration* instead (*Configurations* tab). See *Sistem de Autentificare* for more details.

7. Press *OK* to create the connection.

Note that any proxy settings you may have set in your preferences are also recognized. Also note that it is possible to *Load* the connection parameters from a `.XML` file or *Save* them to a `.XML` file.

Now we are ready to load WFS layers from the above connection.

1. Choose «Gateway Geomatics» from the *Server Connections* drop-down list.
2. Click *Connect*
3. Select the *Parks* layer in the list
4. You can also choose whether to:
   - *Use title for layer name*, showing the layer's title as defined on the server in the *Layers* panel instead of its *Name*
   - *Only request features overlapping the view extent*
   - *Change…* the layer's CRS to any other supported by the service
   - or build a query to specify particular features to retrieve from the service: double-click on the row of the layer to open the *SQL Query Composer* dialog. That dialog provides widgets to write an advanced SQL query relying on available tables and columns of the service, with sorting and filtering and a bunch of SQL functions, spatial predicates and operators.

     The query you build will appear after validation in the *SQL* column within the *WFS / OGC API - Features* table, and the filtered layer will display the icon next to it in the *Layers* panel. It is thus possibe to adjust the query at any moment.
5. Click *Add* to add the layer to the map.

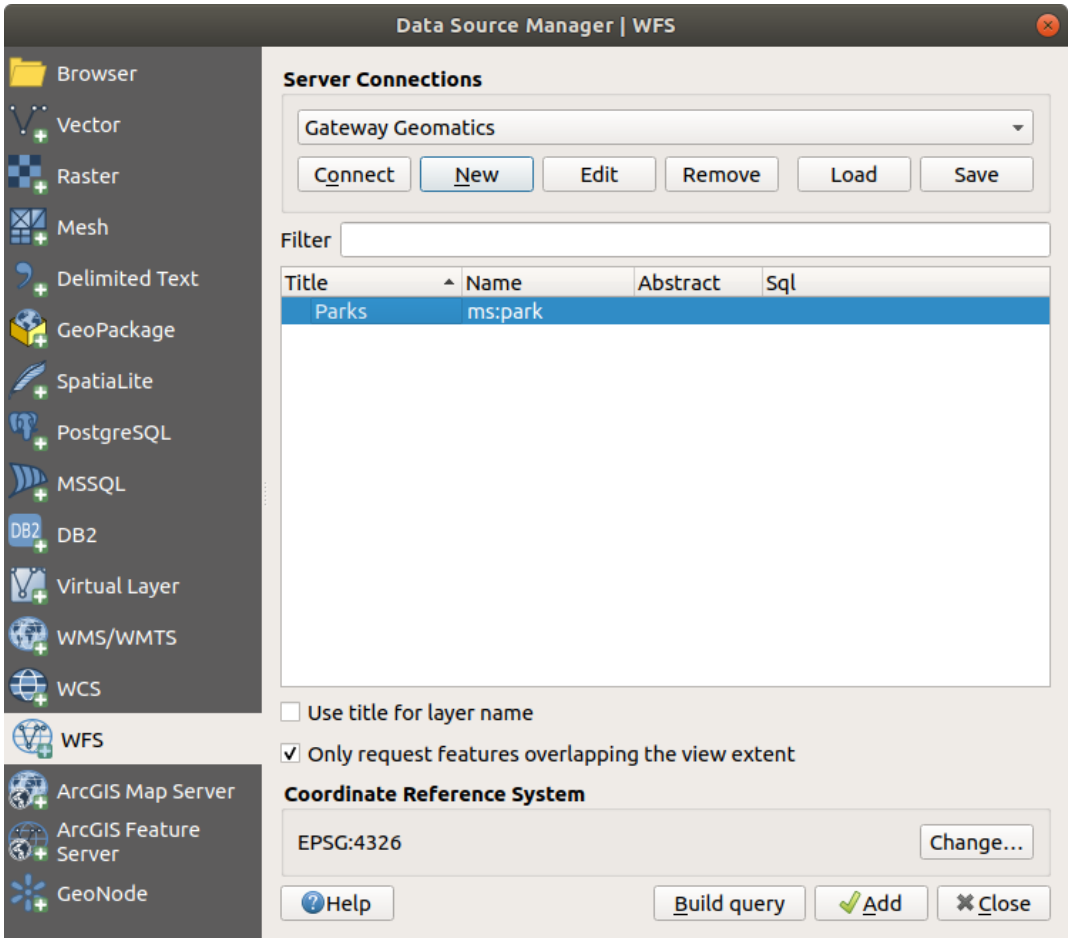


Fig. 23.7: Adăugarea unui strat WFS

You'll notice the download progress is visualized in the lower left of the QGIS main window. Once the layer is loaded, you can identify and select a couple of features and view the attribute table.

CAPITOLUL 24

Lucrul cu datele GPS

24.1 Introducing GNSS/GPS Data

24.1.1 Ce este GPS?

GPS, the **Global Positioning System**, is a satellite-based system that allows anyone with a GPS receiver to find their exact position anywhere in the world. GPS is used as an aid in navigation, for example in airplanes, in boats and by hikers. The GPS receiver uses the signals from the satellites to calculate its latitude, longitude and (sometimes) elevation. Most receivers also have the capability to store:

- locations (known as **waypoints**)
- sequences of locations that make up a planned **route**
- and a **track** log of the receiver's movement over time.

Waypoints, routes and tracks are the three basic feature types in GPS data. QGIS displays waypoints in point layers, while routes and tracks are displayed in linestring layers.

Notă: QGIS supports also GNSS receivers. But we keep using the term GPS in this documentation.

24.1.2 Transferring or loading GPS data

Loading a GPX file

There are dozens of different file formats for storing GPS data. The format that QGIS uses is called GPX (GPS eXchange format), which is a standard interchange format that can contain any number of waypoints, routes and tracks in the same file.

To load a `GPX` file:

1. Open the *GPS* tab in the *Data Source Manager* dialog, i.e.:
   - Click the Open Data Source Manager button on the toolbar (or press `Ctrl+L`) and enable the target tab

- Or select *Layer* ► *Add Layer* ► *Add GPX Layer…*

2. Use the … Browse button next to the *GPX dataset* option to select the GPX file
3. Use the check boxes to select the *Feature types* you want to load from the file. Each feature type (*Waypoints*, *Tracks* or *Routes*) will be loaded in a separate layer.

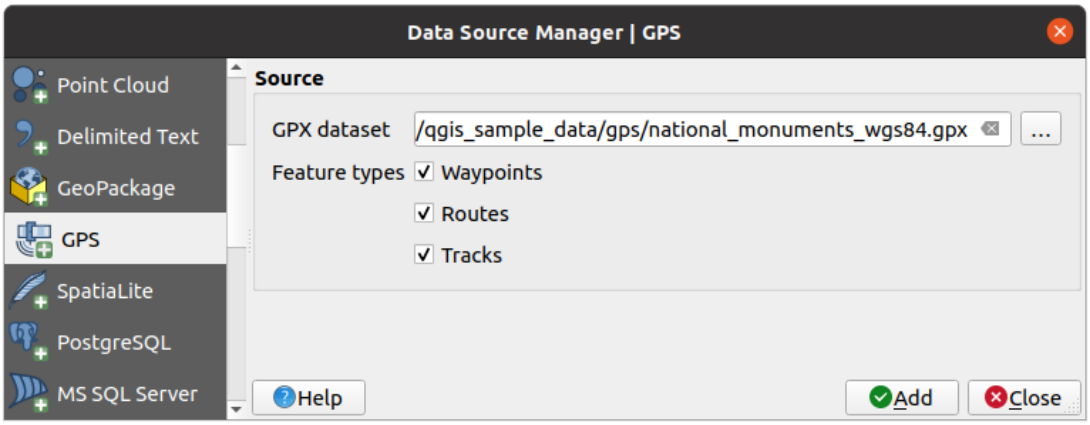


Fig. 24.1: Loading GPS Data dialog

Loading to or from a device

There are lots of different types of GPS devices and formats. Since QGIS uses GPX files, you need a way to convert other GPS file formats to GPX. QGIS can do that using the free program GPSBabel. GPSBabel can help you convert waypoints, tracks, and routes between popular GPS receivers such as Garmin or Magellan and mapping programs like Google Earth or Basecamp. Literally hundreds of GPS receivers and programs are supported. It can also transfer GPS data between your computer and a GPS device.

Under *Settings* ► *Options* ► *GPS* ► *GPSBabel*, QGIS allows you to define your own device type and set parameters of conversion that could later be used by the *Processing GPS algorithms*.

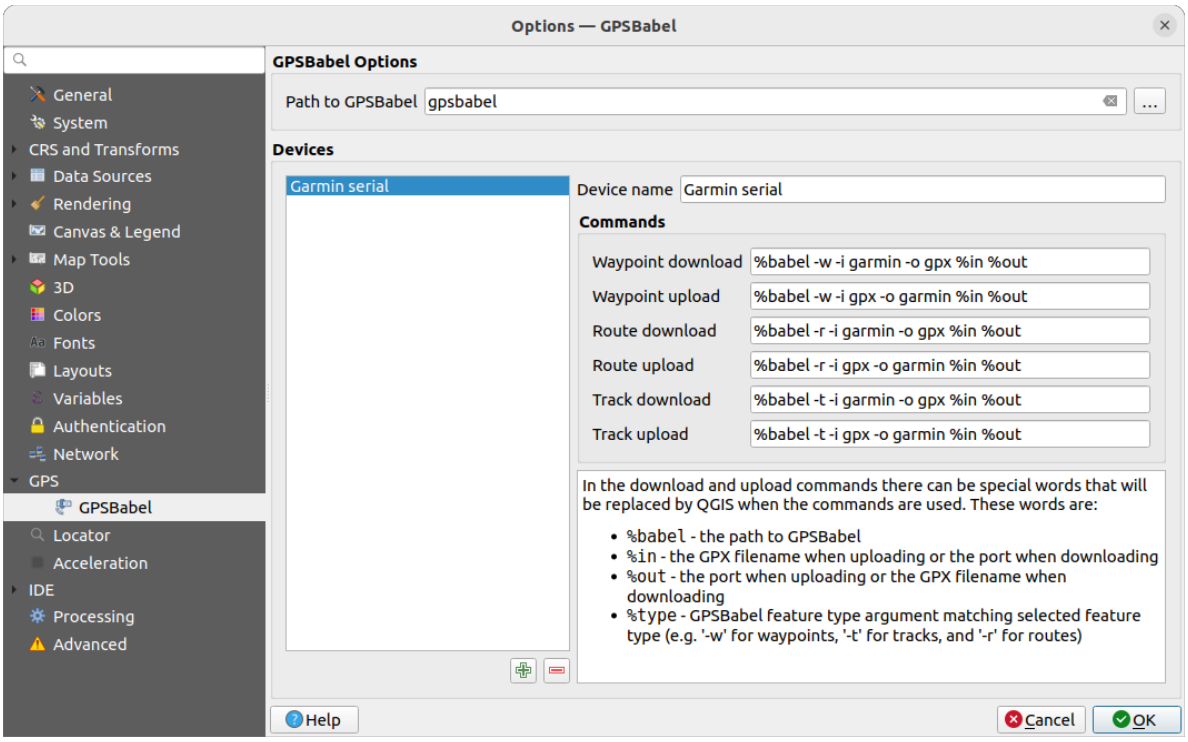


Fig. 24.2: GPS Babel settings

1. First you have to define the *Path to GPSBabel* binaries.
2. Then you may want to add your device. You can update devices list using Add new device or Remove device button.
3. For each device:
   - you provide a *Device name*
   - you configure different *Commands* QGIS will use while interacting with it, such as:
     - *Waypoint download* from the device
     - *Waypoint upload* to the device
     - *Route download* from the device
     - *Route upload* to the device
     - *Track download* from the device
     - *Track upload* to the device

   While the commands are usually GPSBabel commands, you can also use any other command line program that can create a GPX file. QGIS will replace the keywords `%type`, `%in`, and `%out` when it runs the command.

   As an example, if you create a device type with the download command `gpsbabel %type -i garmin -o gpx %in %out` and then use it to download waypoints from port `/dev/ttyS0` to the file `output.gpx`, QGIS will replace the keywords and run the command `gpsbabel -w -i garmin -o gpx /dev/ttyS0 output.gpx`.

   Read the GPSBabel manual for the command line options that may be specific to your use case.

Once you have created a new device type, it will appear in the device lists for the GPS download and upload algorithms.

Notă: GPS units allow you to store data in different coordinate systems. When downloading a GPX file (from your GPS unit or a web site) and then loading it in QGIS, be sure that the data stored in the GPX file uses WGS 84

(latitude/longitude). QGIS expects this, and it is the official GPX specification. See GPX 1.1 Schema Documentation.

24.2 Urmărirea live a GPS-ului

QGIS can help you do field mapping with a GPS receiver. Such a live tracking operation is done using the *GPS Toolbar*. Some *device configuration* may be needed before connecting QGIS and the GPS receiver.

24.2.1 GPS toolbar

The *GPS Toolbar* provides the main tools to control a live tracking session. It can be activated from *View* ► *Toolbars* ► *GPS*. It follows the state of the project, GPS, and current GPS track, and enables actions only when they make sense. Once a device is detected by QGIS, you will be able to interact with it:

- Connect to GPS
- Disconnect from GPS
- Recenter map on GPS location: the map will immediately recenter on the current GPS position regardless of the *automatic recentering parameter*.
- Set destination layer for GPS digitized features: By default, QGIS will *Follow active layer* for feature digitizing, meaning that GPS digitizing tools will adapt to the layer selected in *Layers* panel and created features will be stored in it accordingly. While this may be useful in some contexts, it also implies careful interaction with other layers to avoid storing inadvertently features in undesired layers. This option allows you to explicitly indicate a layer for the data storage during the live tracking session, and you could switch depending on your needs. Exposed tools will adapt as well to the selected layer type.

  The GPS destination layer is made automatically editable when a feature is being created, and the user is notified accordingly.
- actions for feature creation:
  - Add vertex to GPS track using GPS current location
  - Create feature from track
  - Create a new point feature at current GPS location
  - Reset track
- Show GPS information: opens the *GPS Information* panel
- a display box for quick access to some GPS information components you may need:
  - Show location
  - Show altitude (Geoid)
  - Show altitude (WGS 84 ellipsoid)
  - Show ground speed
  - Show bearing
  - Show total track length
  - Show distance from start of track

- The Settings button opens a drop-down menu with common settings which are expected to be modified mid-session:
  - *Show location marker*
  - *Show bearing line*
  - *Rotate map to match GPS direction*
  - Options to control map recentering:
    - *Always recenter map* when the GPS position is offset from the map canvas center by a certain distance (as a ratio of the map canvas extent)
    - *Recenter map when leaving extent*
    - *Never recenter*
  - *Automatically add track vertices* whenever new locations are received from the GPS device.
  - *Automatically save added features*: features created from GPS locations are immediately committed to their target layers (skipping the usual layer edit buffer)
  - *Time stamp destination* to adjust the field for time fix storage
  - *Log to GeoPackage/SpatiaLite…*: When activated, the user will be prompted to select an existing GeoPackage/SpatiaLite file or enter a new file name. A `gps_points` and `gps_tracks` table will be created in the file with a predefined structure.

    All incoming GPS messages will be logged to the `gps_points` layer, along with speed, bearing, altitude and accuracy information from the GPS.

    When the GPS is disconnected (or QGIS closed), the entire recorded GPS track will be added to the `gps_tracks` table (along with some calculated information like track length, start and end times).
  - *Log NMEA sentences…*: enables logging all raw NMEA strings from the device to a text file
  - *GPS settings…* to access the GPS *global options* dialog

---

Sfat: Live status bar information on position

When a GPS device is connected and the user moves the cursor over the map canvas, a live status bar message displays the distance and bearing from the cursor to the GPS position. Project distance and bearing settings are respected in this display. On a touch screen device, use a tap-and-hold event to trigger the live status bar message.

24.2.2 GPS Information Panel

For full monitoring of a live GPS tracking in QGIS, you may need to enable the *GPS Information Panel* (*View* ► *Panels* ► or press `Ctrl+0`).

In the top right corner of the *GPS Information Panel*, you press *Connect* to initiate connection between QGIS and a plugged-in GPS receiver, or *Disconnect* them.

In the top left of the panel, following buttons are accessible:

- Position: live details on the GPS position and sensors
- Signal: signal strength of satellite connections
- Settings: drop-down menu to *live tracking options* you may need to modify during a session

Poziția și atributele suplimentare

In the Position tab, if the GPS is receiving signals from satellites, you will see your position in latitude, longitude and altitude together with additional attributes.

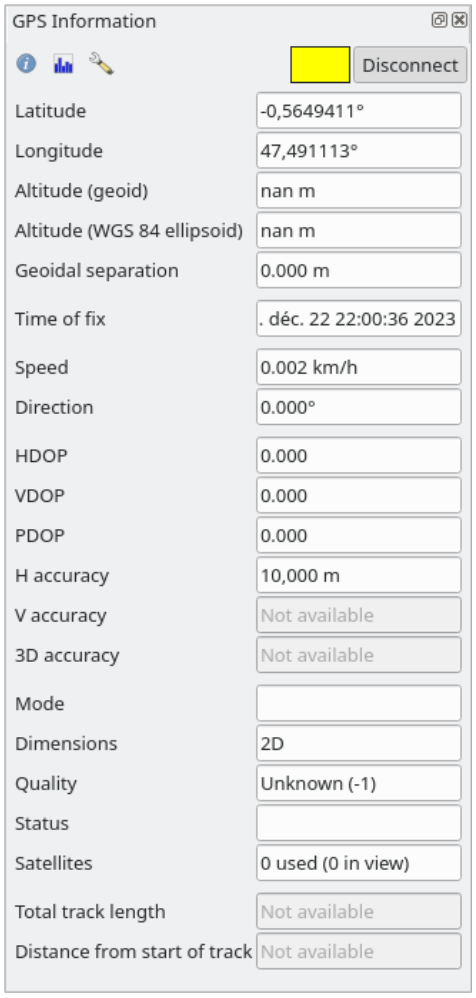


Fig. 24.3: Urmărirea poziției GPS și atributele suplimentare

- *Latitude*
- *Longitude*
- *Altitude (Geoid)*: Altitude/elevation above or below the mean sea level
- *Altitude (WGS 84 ellipsoid)*: Altitude/elevation above or below the WGS-84 Earth ellipsoid
- *Geoidal separation*: Difference between the WGS-84 Earth ellipsoid and mean sea level (geoid), − means mean sea level is below ellipsoid
- *Time of fix*
- *Speed*: Ground speed
- *Direction*: Bearing measured in degrees clockwise from true north to the direction of travel
- *HDOP*: Horizontal dilution of precision
- *VDOP*: Vertical dilution of precision
- *PDOP*: Dilution of precision
- *H accuracy*: Horizontal accuracy in meters

- *V accuracy*: Vertical accuracy in meters
- *3D accuracy*: 3D Root Mean Square (RMS) in meters
- *Mode*: GPS receiver configuration 2D/3D mode, can be `automatic` or `manual`
- *Dimensions*: Position fix dimension, can be `2D`, `3D` or `No fix`
- *Quality*: Positioning quality indicator
- *Status*: Position fix status, can be `Valid` or `Invalid`
- *Satellites*: Count of satellites used in obtaining the fix
- *Total track length*: Total distance of current GPS track
- *Distance from start of track*: Direct distance from first vertex in current GPS track to latest vertex

Signal

With Signal tab, you can see the signal strength of the satellites you are receiving signals from.

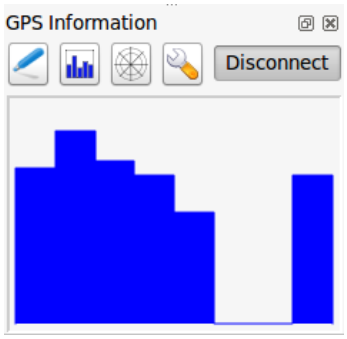


Fig. 24.4: Puterea de urmărire a semnalului GPS

24.2.3 Conectaare la un GPS Bluetooth pentru urmărirea în direct a poziției

With QGIS you can connect a Bluetooth GPS for field data collection. To perform this task you need a GPS Bluetooth device and a Bluetooth receiver on your computer.

At first you must let your GPS device be recognized and paired to the computer. Turn on the GPS, go to the Bluetooth icon on your notification area and search for a New Device.

On the right side of the Device selection mask make sure that all devices are selected so your GPS unit will probably appear among those available. In the next step a serial connection service should be available, select it and click on *Configure* button.

Rețineți numărul portului COM atribuit conexiunii GPS, după cum rezultă din proprietățile Bluetooth.

După ce GPS-ul a fost recunoscut, creați asocierea pentru conexiune. De obicei, codul de autorizare este `0000`.

Now open *GPS information* panel and switch to GPS options screen. Select the COM port assigned to the GPS connection and click the *Connect*. After a while a cursor indicating your position should appear.

If QGIS can't receive GPS data, then you should restart your GPS device, wait 5-10 seconds then try to connect again. Usually this solution work. If you receive again a connection error make sure you don't have another Bluetooth receiver near you, paired with the same GPS unit.

24.2.4 Examples of GPS devices connection in QGIS

Folosirea GPSMAP 60cs

MS Windows

Easiest way to make it work is to use a middleware (freeware, not open) called GPSGate.

Launch the program, make it scan for GPS devices (works for both USB and BT ones) and then in QGIS:

1. In *Settings* ► *Options* ► *GPS*, detect the plugged-in device. You can use the *Autodetect* mode.
2. Press *Connect* in the *GPS Information* panel

Ubuntu/Mint GNU/Linux

As for Windows the easiest way is to use a server in the middle, in this case GPSD, so

1. Install the program

```
sudo apt install gpsd
```

2. Apoi, încărcați nucleul `garmin_gps`

```
sudo modprobe garmin_gps
```

#. And then connect the unit. Then check with `dmesg` the actual device being used by the unit, for example `/dev/ttyUSB0`.

1. Now you can launch gpsd

```
gpsd /dev/ttyUSB0
```

2. Și, la final, conectați-va la instrumentul de urmărire în direct din QGIS.

Folosirea datalogger-ului BTGP-38KM (doar Bluetooth)

Folosirea GPSD (în Linux) sau GPSGate (în Windows) se face fără efort.

Folosirea datalogger-ului BlueMax GPS-4044 (atât BT cât și USB)

MS Windows

The live tracking works for both USB and BT modes, by using GPSGate or even without it, just use the *Autodetect* mode, or point the tool to the right port.

Ubuntu/Mint GNU/Linux

Pentru USB

The live tracking works both with GPSD

```
gpsd /dev/ttyACM3
```

or without it, by connecting the QGIS live tracking tool directly to the device (for example `/dev/ttyACM3`).

Pentru Bluetooth

The live tracking works both with GPSD

```
gpsd /dev/rfcomm0
```

or without it, by connecting the QGIS live tracking tool directly to the device (for example `/dev/rfcomm0`).

CAPITOLUL 25

Sistem de Autentificare

25.1 Prezentare Generală a Sistemului de Autentificare

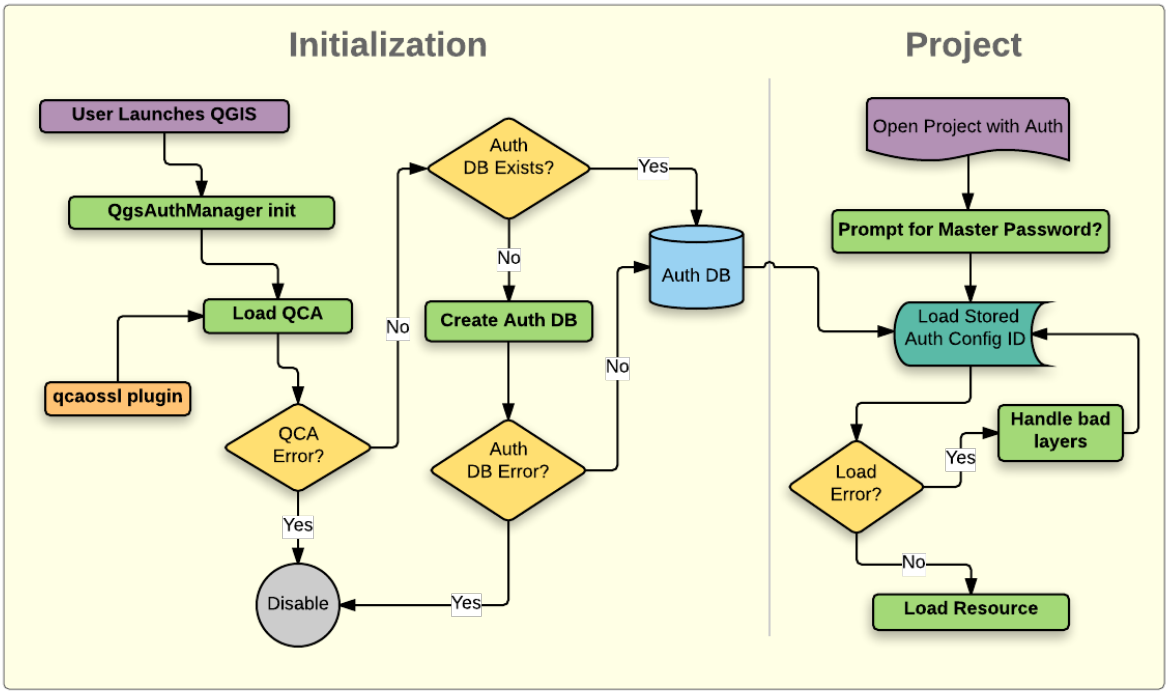


Fig. 25.1: Anatomia Sistemului de Autentificare

25.1.1 Baza de date de autentificare

The new authentication system stores authentication configurations in an SQLite database file located, by default, at `<profile directory>/qgis-auth.db`.

Această bază de date de autentificare poate fi mutată între instalările QGIS, fără afectarea altor preferințe ale utilizatorului QGIS curent, deoarece este complet separată de setările normale ale QGIS. Atunci când stocați inițial o configurație a unei baze de date, este generat un ID de configurare (un șir aleatoriu de 7 caractere alfanumerice). Acesta reprezintă configurația, fiind posibilă stocarea în clar a ID-ului de către componentele aplicației, (cum ar fi proiectele, plugin-urile, sau fișierele de setare) fără divulgarea credențialelor asociate.

Notă: Directorul părinte al *qgis-auth.db* poate fi setat folosind variabila de mediu `QGIS_AUTH_DB_DIR_PATH`, sau din linia de comandă, pe durata lansării cu opțiunea `--authdbdirectory`.

25.1.2 Parola master

To store or access sensitive information within the database, a user must define a *master password*. A new master password is requested and verified when initially storing any encrypted data to the database. When sensitive information is accessed, the user is prompted for the master password. The password is then cached for the remainder of the session (until application is quit), unless the user manually chooses an action to clear its cached value. Some instances of using the authentication system do not require input of the master password, such as when selecting an existing authentication configuration, or applying a configuration to a server configuration (such as when adding a WMS layer).

You can choose to save the password in the `Wallet/Keyring` of your computer.

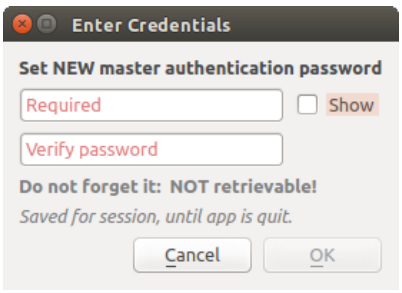


Fig. 25.2: Introducerea noii parole master

Notă: A path to a file containing the master password can be set using the following environment variable, `QGIS_AUTH_PASSWORD_FILE`.

Gestiunea parolei master

Odată stabilită, parola principală poate fi resetată; parola master curentă va fi necesară înainte de resetare. Pe durata acestui proces, există opțiunea de a genera o copie de rezervă completă a bazei de date curente.

Fig. 25.3: Resetarea parolei master

În cazul în care utilizatorul uită parola principală, nu există nici o modalitate de a o recupera sau de a o suprascrie. De asemenea, nu există nici un mijloc de recuperare a informațiilor criptate, fără a cunoaște parola de master.

Dacă un utilizator introduce incorect, de trei ori, parola, fereastra de dialog se va oferi să șteargă baza de date.

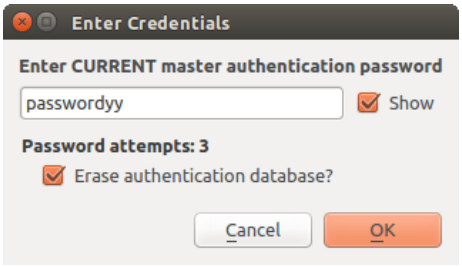


Fig. 25.4: Se va cere parola după trei încercări eronate

25.1.3 Configurații de Autentificare

You can manage authentication configurations from *Configurations* in the *Authentication* tab of the QGIS Options dialog (*Settings* ► *Options*).

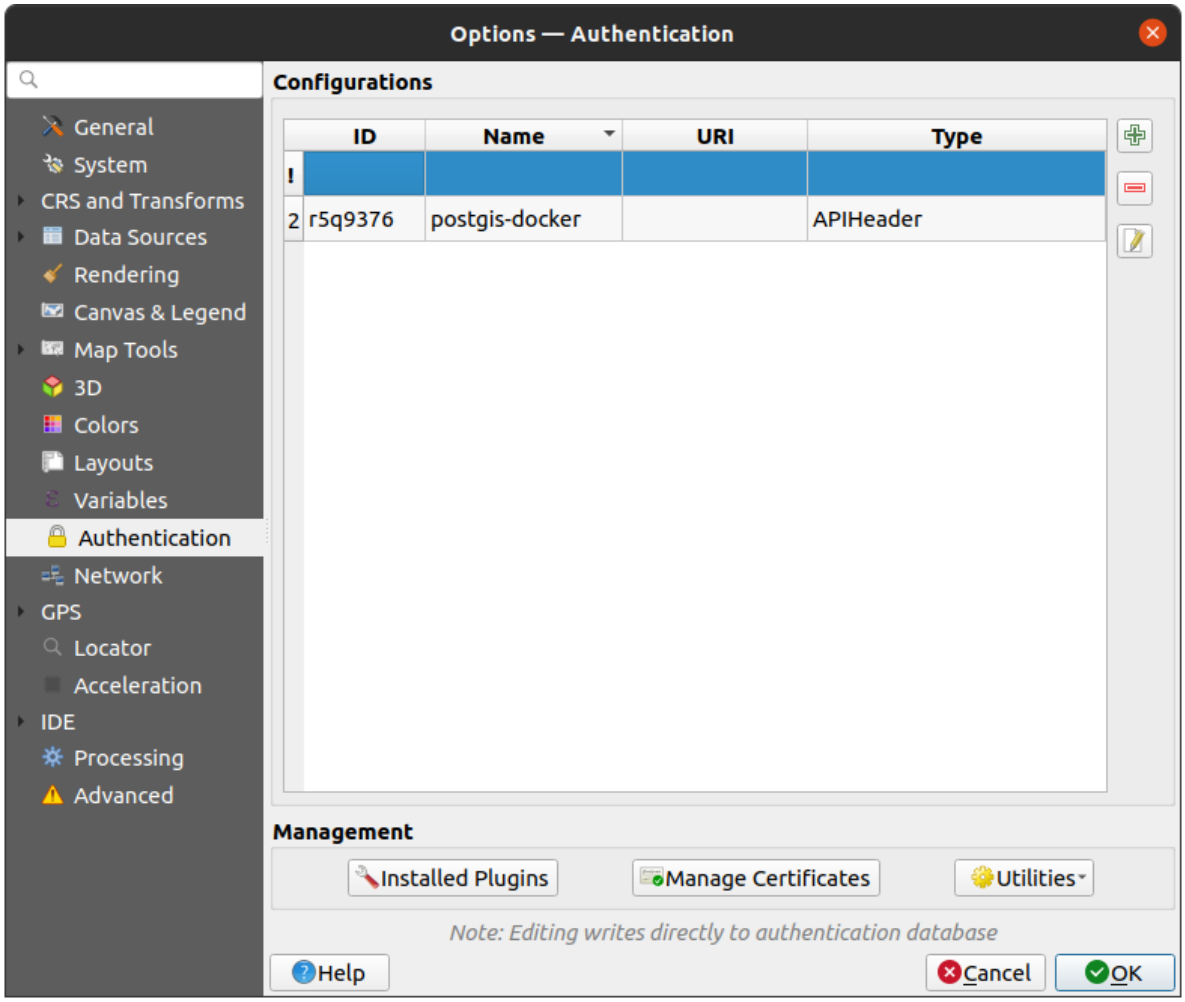


Fig. 25.5: Editorul de configurații

Use the button to add a new configuration, the button to remove configurations, and the button to modify existing ones.

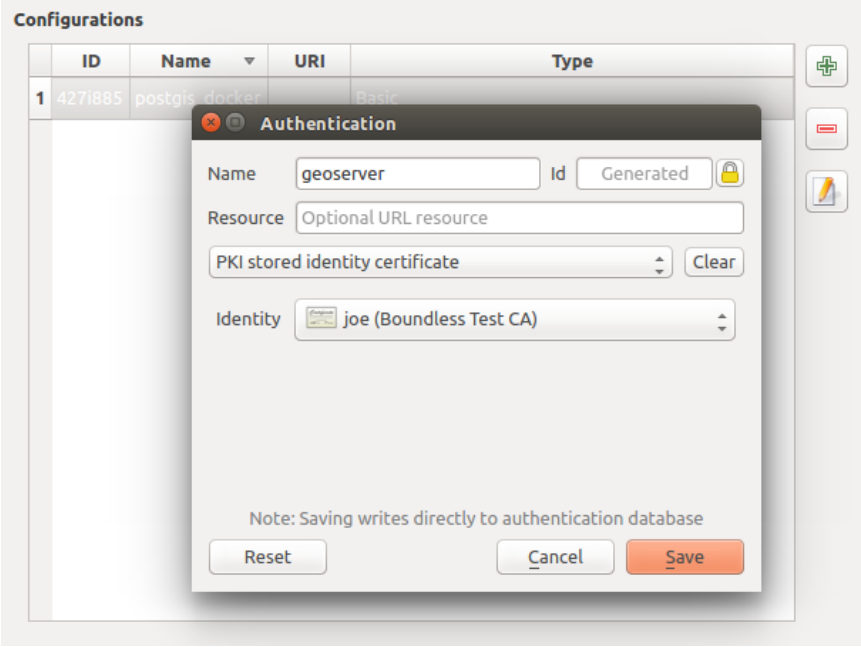


Fig. 25.6: Adăugarea configurărilor din cadrul Editorului de Configurații

The same type of operations for authentication configuration management (Add, Edit and Remove) can be done when configuring a given service connection, such as configuring an OWS service connection. For that, there are action buttons within the configuration selector for fully managing configurations found within the authentication database. In this case, there is no need to go to the *configurations* in *Authentication* tab of QGIS options unless you need to do more comprehensive configuration management.

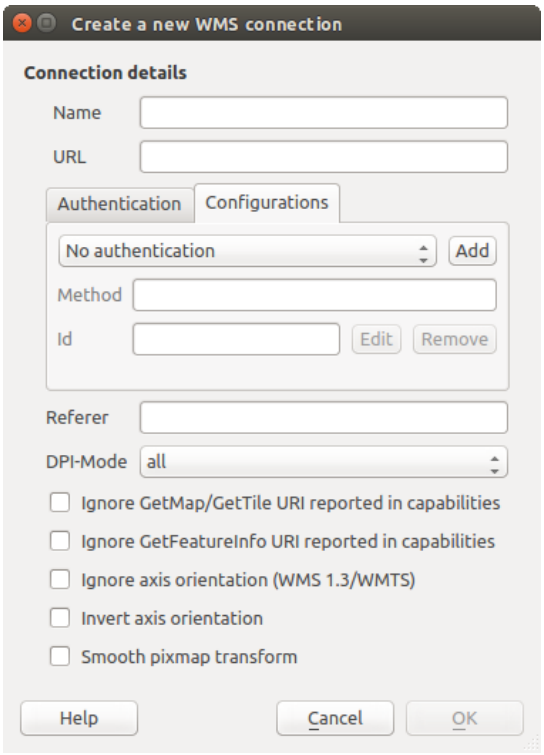


Fig. 25.7: WMS connection dialog showing *Add*, *Edit*, and *Remove* authentication configuration buttons

When creating or editing an authentication configuration, the info required is a name, an authentication method and

any other info that the authentication method requires (see more about the available authentication types in *Metoda de Autentificare*).

25.1.4 Metoda de Autentificare

Available authentications are provided by C++ plugins much in the same way data provider plugins are supported by QGIS. The method of authentication that can be selected is relative to the access needed for the resource/provider, e.g. HTTP(S) or database, and whether there is support in both QGIS code and a plugin. As such, some authentication method plugins may not be applicable everywhere an authentication configuration selector is shown. A list of available authentication method plugins and their compatible resource/providers can be accessed going to *Settings* ► *Options* and, in the *Authentication* tab, click the *Installed Plugins* button.

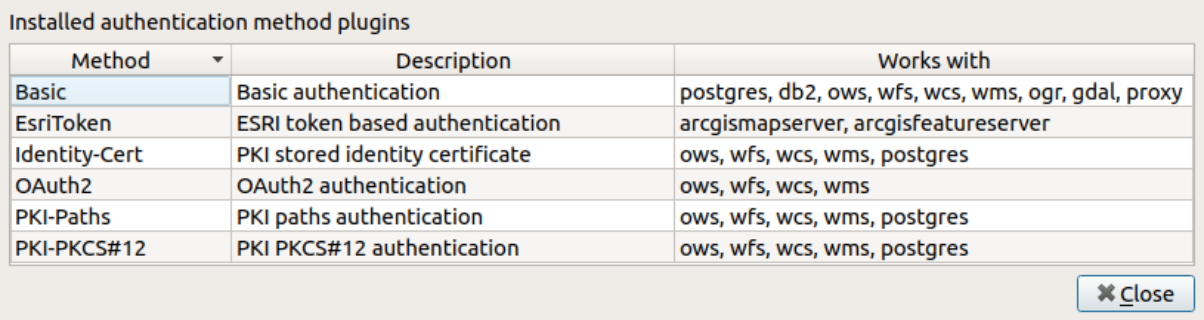


Fig. 25.8: Available method plugins list

Plugins can be created for new authentication methods that do not require QGIS to be recompiled. Since the support for plugins is currently C++-only, QGIS will need to be restarted for the new dropped-in plugin to become available to the user. Ensure your plugin is compiled against the same target version of QGIS if you intend to add it to an existing target install.

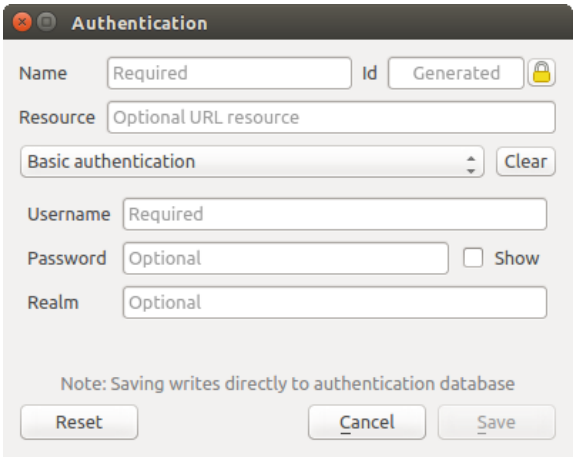


Fig. 25.9: Configurări de autentificare HTTP de bază

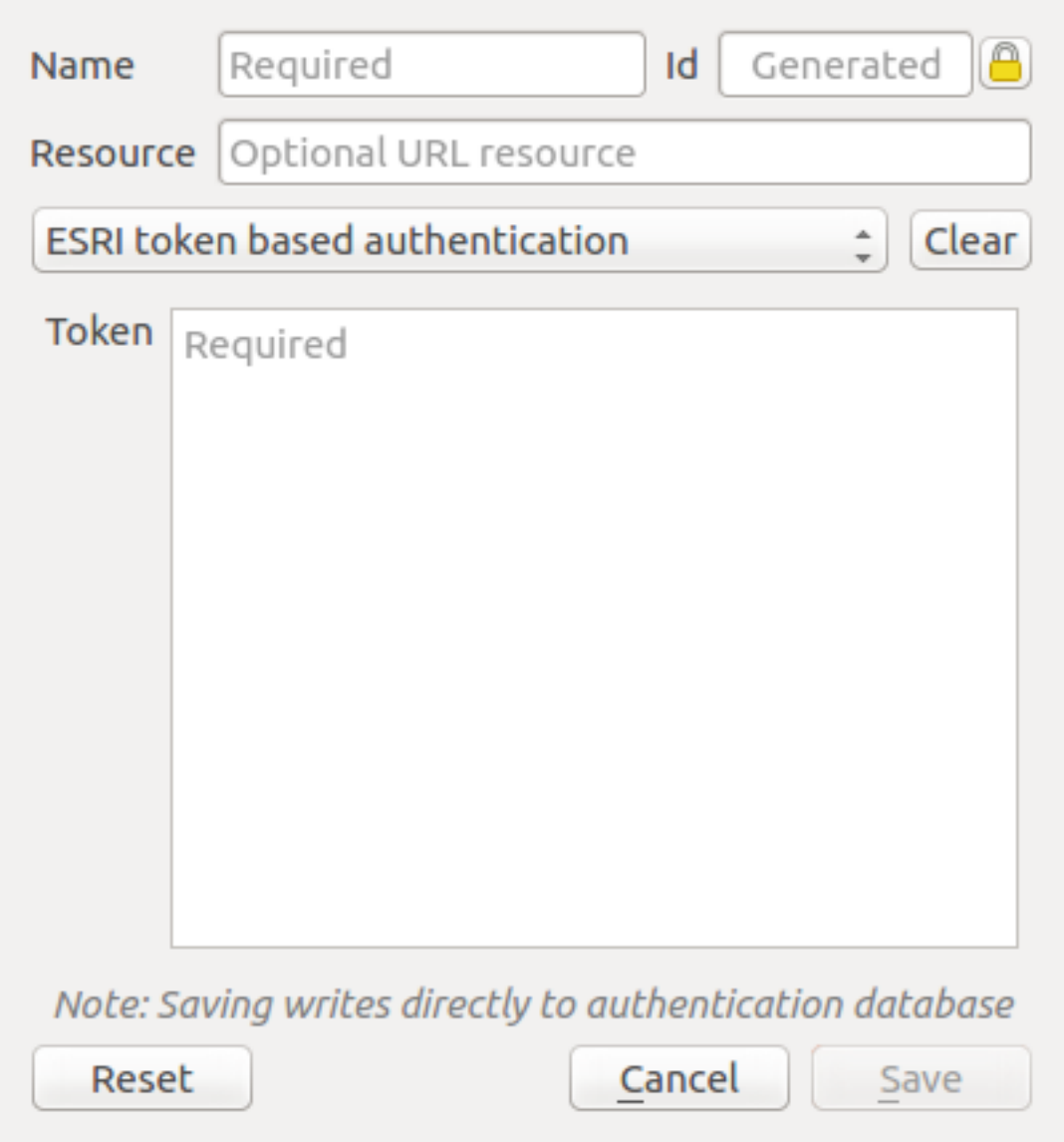


Fig. 25.10: ESRI Token authentication configs

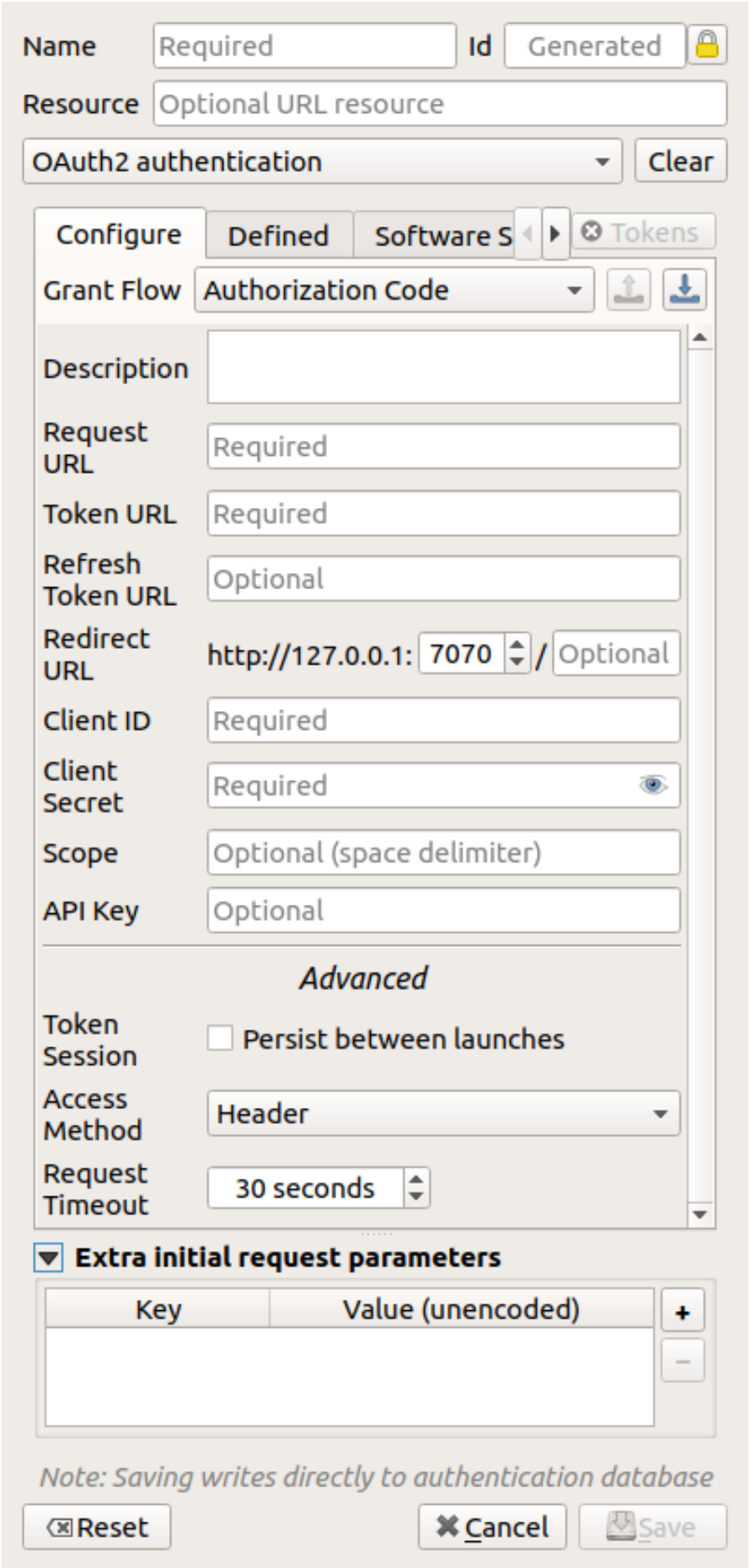


Fig. 25.11: OAuth2 authentication configs

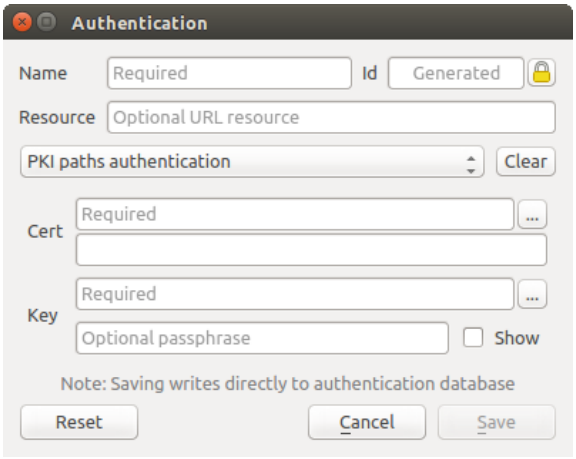


Fig. 25.12: Configurări de autentificare ale căilor PKI

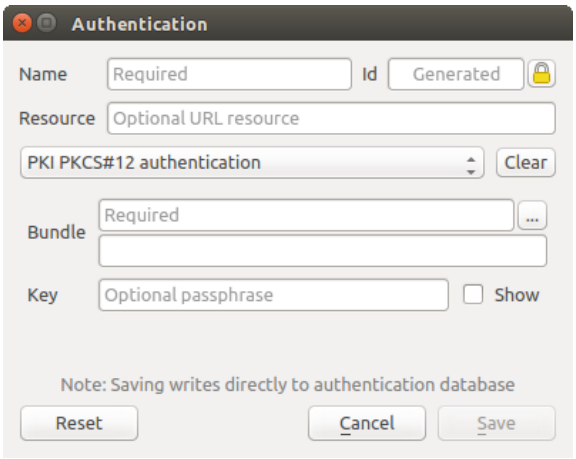


Fig. 25.13: Configurări de autentificare ale căilor către fișierele PKI PKCS#12

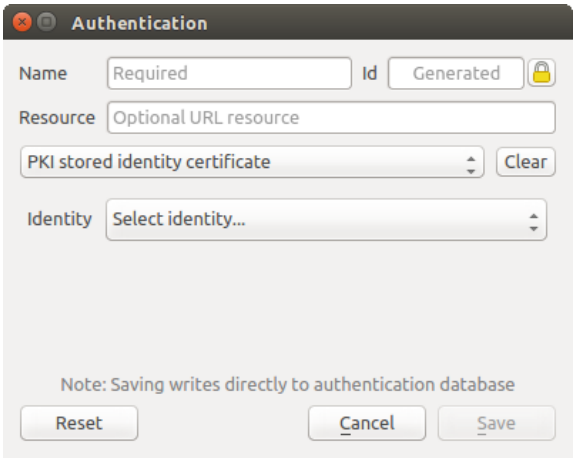


Fig. 25.14: Configurări de autentificare ale Identităților Stocate

Notă: The Resource URL is currently an *unimplemented* feature that will eventually allow a particular configuration to be auto-chosen when connecting to resources at a given URL.

25.1.5 Utilitare pentru Parola Master și Configurările de Autentificare

Under the Options menu (*Settings* ► *Options*) in the *Authentication* tab, there are several utility actions to manage the authentication database and configurations:

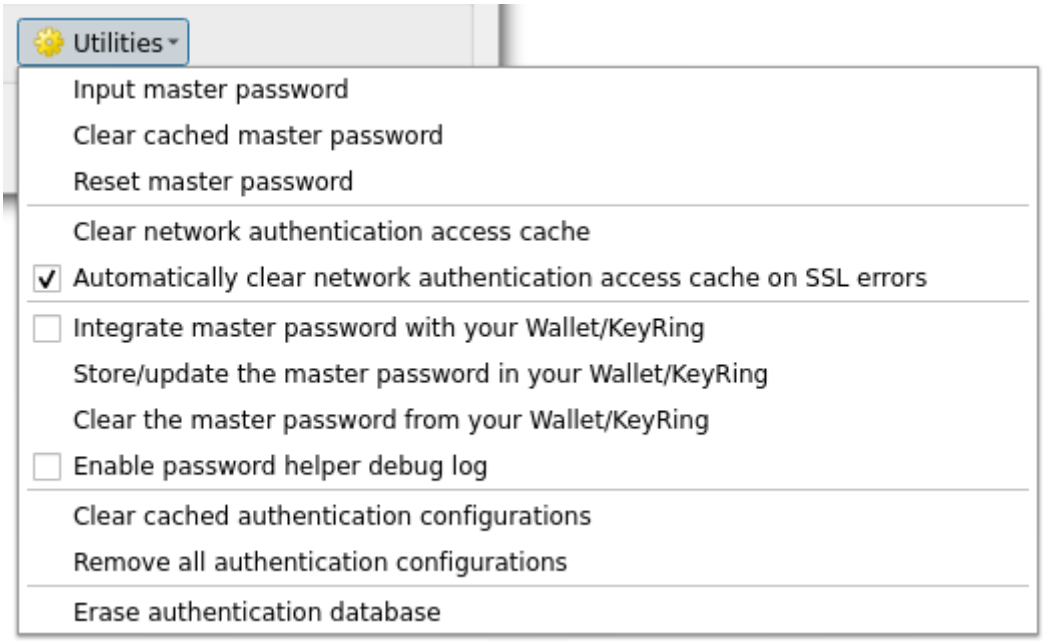


Fig. 25.15: Meniul Utilitarelor

- **Input master password**: opens the master password input dialog, independent of performing any authentication database command
- **Clear cached master password**: unsets the master password if it has been set
- **Reset master password**: opens a dialog to change the master password (the current password must be known) and optionally back up the current database
- **Clear network authentication access cache**: clears the authentication cache of all connections
- **Automatically clear network authentication access cache on SSL errors**: the connection cache stores all authentication data for connections, also when the connection fails. If you change authentication configurations or certification authorities, you should clear the authentication cache or restart QGIS. When this option is checked, the authentication cache will be automatically cleared every time an SSL error occurs and you choose to abort the connection
- **Integrate master password with your Wallet/Keyring**: adds the master password to your personal Wallet/Keyring
- **Store/update the master password in your Wallet/Keyring**: updates the changed master password in your Wallet/Keyring
- **Clear the master password from your Wallet/Keyring**: deletes the master password from your Wallet/Keyring
- **Enable password helper debug log**: enables a debug tool that will contain all the log information of the authentication methods
- **Clear cached authentication configurations**: clears the internal lookup cache for configurations, used to speed up network connections. This does not clear QGIS's core network access manager's cache, which requires a relaunch of QGIS.
- **Remove all authentication configurations**: clears the database of all configuration records, without removing other stored records.

- **Erase authentication database**: schedules a backup of the current database and complete rebuild of the database table structure. The actions are scheduled for a later time, to ensure that other operations, like project loading, do not interrupt the operation or cause errors due to a temporarily missing database.

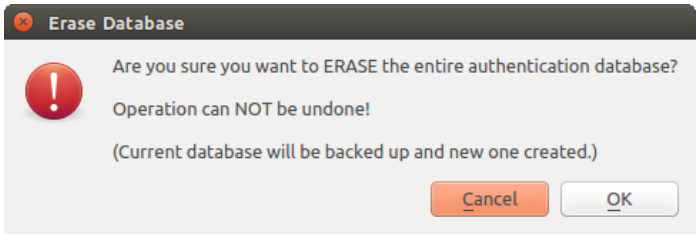


Fig. 25.16: Meniul de verificare a ștergerii DB

25.1.6 Folosirea configurațiilor de autentificare

În mod tipic, o configurație de autentificare este selectată într-un dialog de configurare, pentru serviciile de rețea (cum ar fi WMS). Cu toate acestea, controlul grafic de selectare poate fi încorporat oriunde este necesară autentificarea, sau într-o funcționalitate care nu este de esențială, cum ar fi plugin-urile terțe PyQGIS sau C++.

When using the selector, *No authentication* is displayed in the pop-up menu control when nothing is selected, when there are no configurations to choose from, or when a previously assigned configuration can no longer be found in the database. The *Type* and *Id* fields are read-only and provide a description of the authentication method and the config's ID respectively.

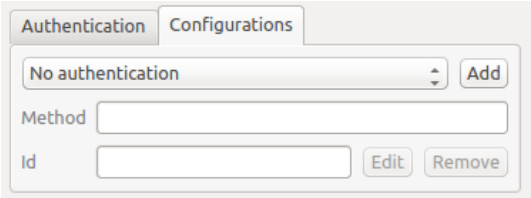


Fig. 25.17: Authentication configuration selector with no authentication

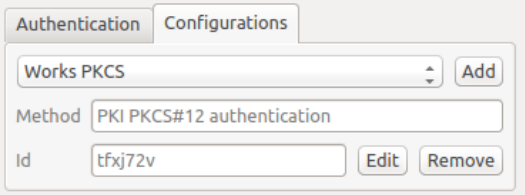


Fig. 25.18: Authentication configuration selector with selected config

25.1.7 Legături Python

All classes and public functions have sip bindings, except `QgsAuthCrypto`, since management of the master password hashing and auth database encryption should be handled by the main app, and not via Python. See authentication_security_considerations concerning Python access.

25.2 Fluxul de Autentificare a Utilizatorului

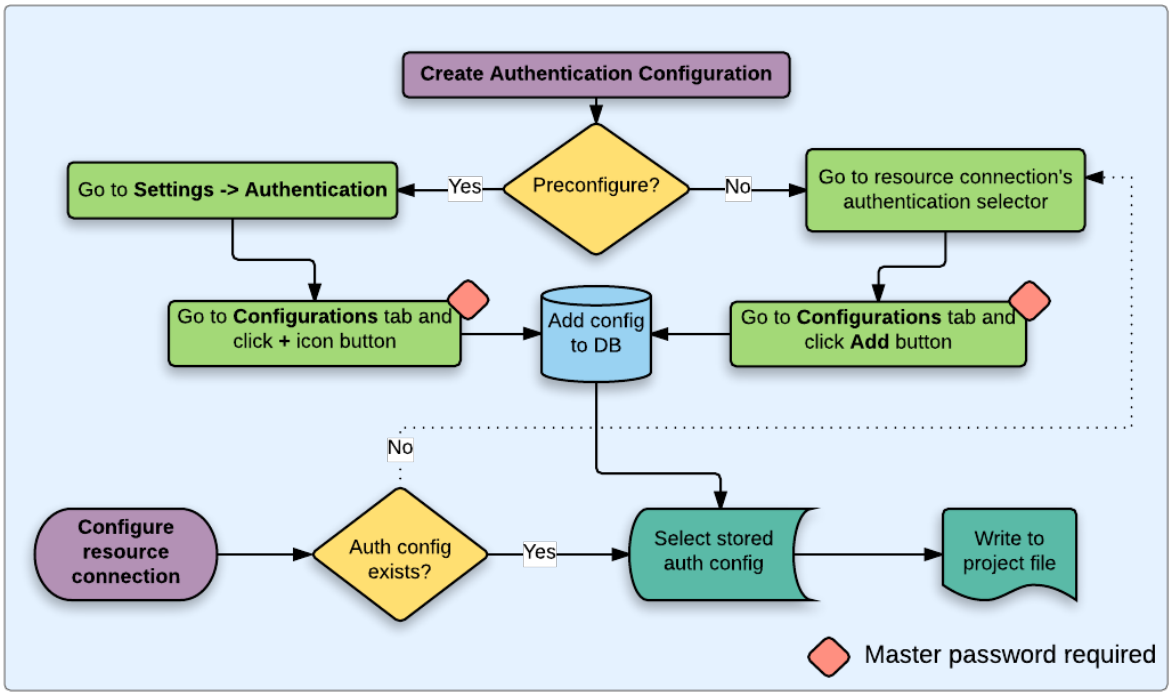


Fig. 25.19: Fluxul de lucru al utilizatorului generic

25.2.1 Autentificarea HTTP(S)

Una dintre cele mai comune conectări la resurse are loc prin prin HTTP(S), cum ar fi serverele de cartografiere web și plugin-urile cu metode de autentificare, care adesea lucrează cu aceste tipuri de conexiuni. Plugin-urile cu metode au acces la obiectul cerere HTTP și pot manipula atât cererea cât și antetele sale. Acest lucru permite mai multe forme de autentificare prin internet. Atunci când conectarea are loc prin HTTP(S) cu ajutorul metodei standard de autentificare nume de utilizator/ parolă, se va încerca autentificarea HTTP de bază a conexiunii.

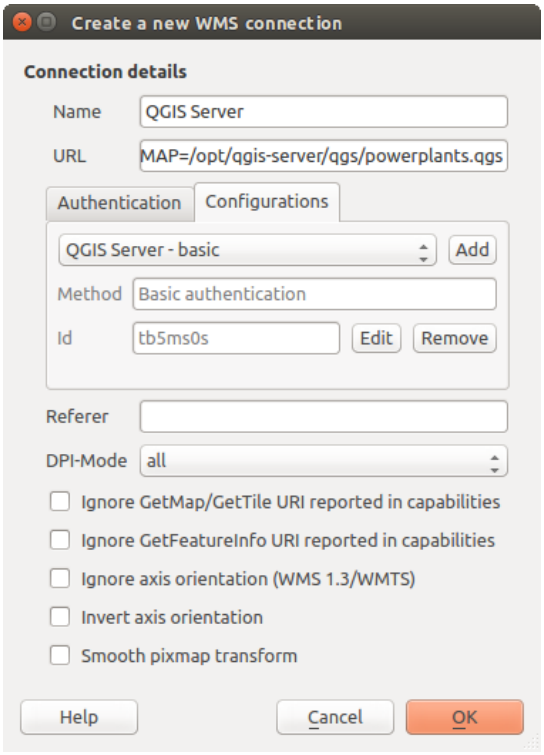


Fig. 25.20: Configurarea unei conexiuni WMS pentru HTTP BASIC

25.2.2 Autentificarea bazei de date

Connections to database resources are generally stored as `key=value` pairs, which will expose usernames and (optionally) passwords, if *not* using an authentication configuration. When configuring with the auth system, the `key=value` will be an abstracted representation of the credentials, e.g. `authfg=81t21b9`.

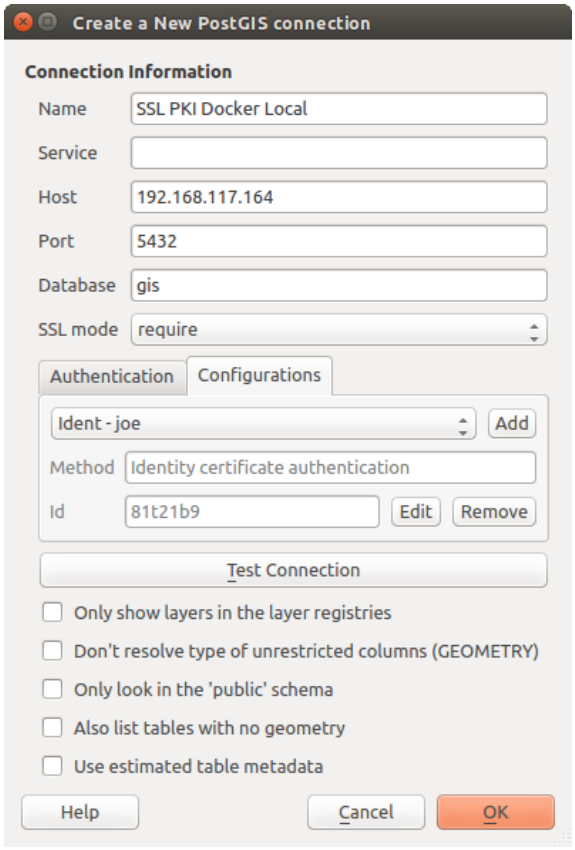


Fig. 25.21: Configurarea unei conexiuni Postgres SSL-with-PKI

25.2.3 Autentificarea PKI

When configuring PKI components within the authentication system, you have the option of importing components into the database or referencing component files stored on your filesystem. The latter may be useful if such components change frequently, or where the components will be replaced by a system administrator. In either instance you will need to store any passphrase needed to access private keys within the database.

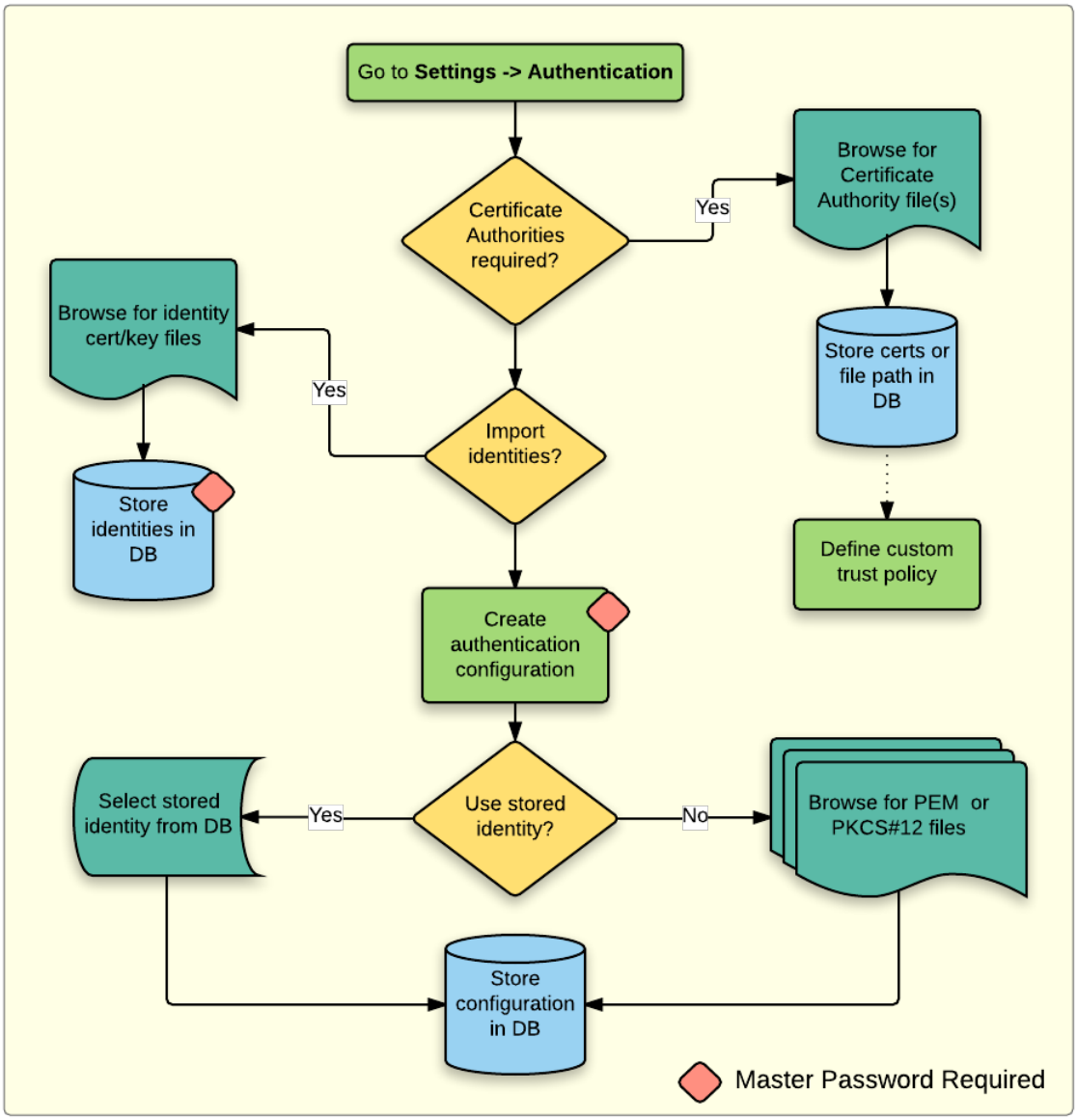


Fig. 25.22: Fluxul de lucru pentru configurarea PKI

All PKI components can be managed in separate editors within the **Certificate Manager**, which can be accessed in the *Authentication* tab in QGIS *Options* dialog (*Settings* ► *Options*) by clicking the *Manage Certificates* button.

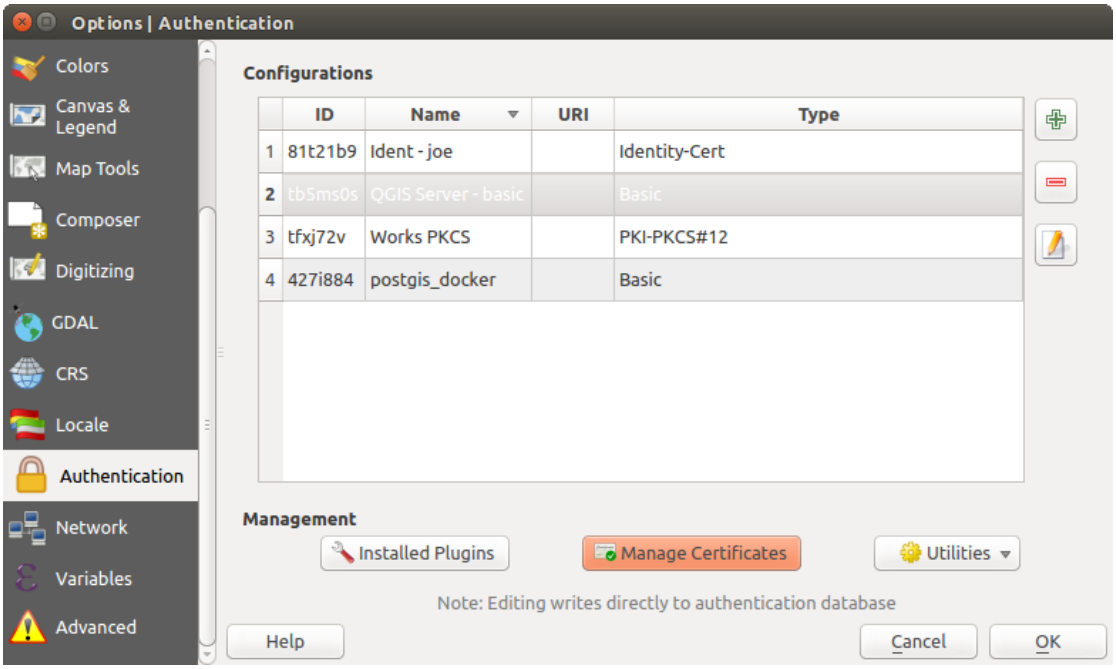


Fig. 25.23: Deschiderea Managerului de Certificate

In the *Certificate Manager*, there are editors for **Identities**, **Servers** and **Authorities**. Each of these are contained in their own tabs, and are described below in the order they are encountered in the workflow chart above. The tab order is relative to frequently accessed editors once you are accustomed to the workflow.

Notă: Because all authentication system edits write immediately to the authentication database, there is no need to click the *Options* dialog *OK* button for any changes to be saved. This is unlike other settings in the Options dialog.

Autorități

You can manage available Certificate Authorities (CAs) from the **Authorities** tab in the **Certificate manager** from the **Authentication** tab of the QGIS **Options** dialog.

As referenced in the workflow chart above, the first step is to import or reference a file of CAs. This step is optional, and may be unnecessary if your PKI trust chain originates from root CAs already installed in your operating system (OS), such as a certificate from a commercial certificate vendor. If your authenticating root CA is not in the OS's trusted root CAs, it will need to be imported or have its file system path referenced. (Contact your system administrator if unsure.)

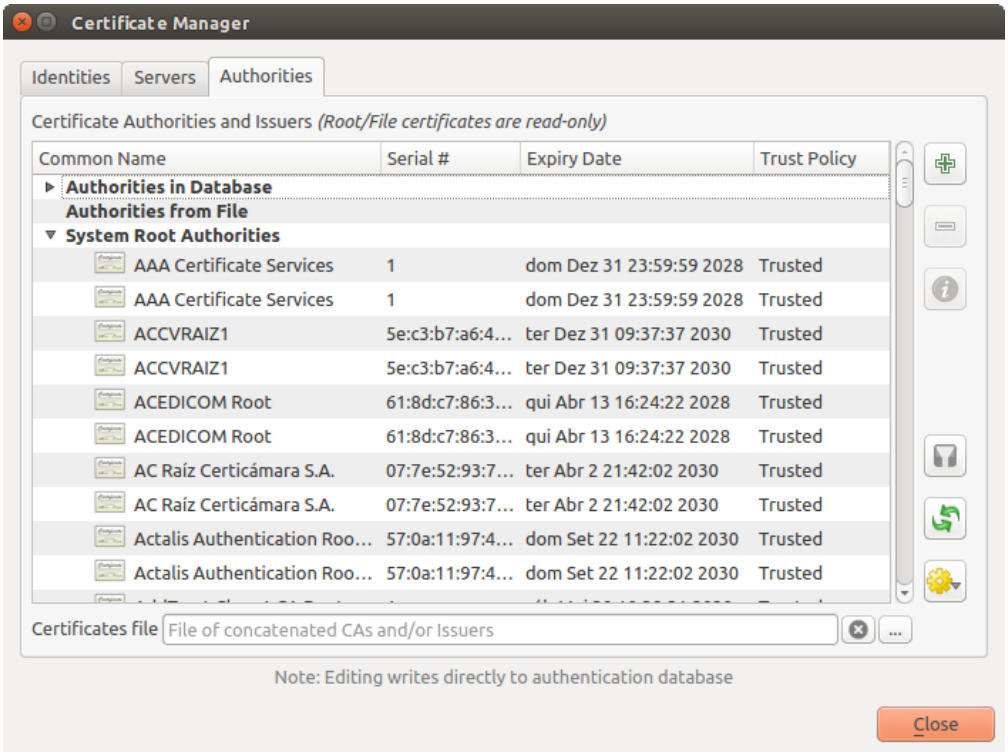


Fig. 25.24: Editorul de autorități

By default, the root CAs from your OS are available; however, their trust settings are not inherited. You should review the certificate trust policy settings, especially if your OS root CAs have had their policies adjusted. Any certificate that is expired will be set to untrusted and will not be used in secure server connections, unless you specifically override its trust policy. To see the QGIS-discoverable trust chain for any certificate, select it and click the Show information for certificate.

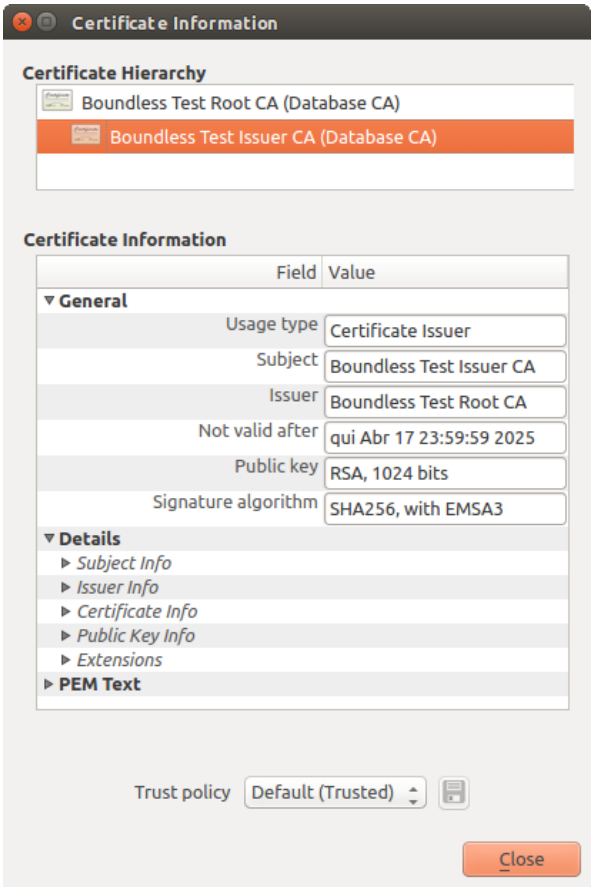


Fig. 25.25: Dialogul nformațiilor despre Certificat

You can edit the *Trust policy* for any selected certificate within the chain. Any change in trust policy to a selected certificate will not be saved to the database unless the Save certificate trust policy change to database button is clicked *per* selected certification. Closing the dialog will **not** apply the policy changes.

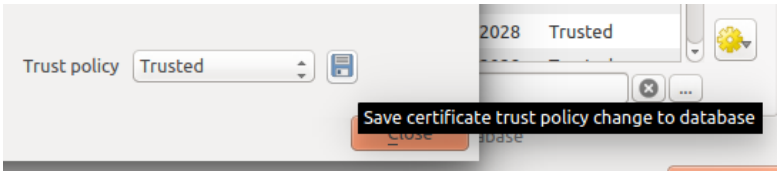


Fig. 25.26: Salvarea modificărilor aduse politicii de încredere

You can review the filtered CAs, both intermediate and root certificates, that will be trusted for secure connections or change the default trust policy by clicking the **Options** button.

Atenționare: Schimbarea politicii de încredere implicită, poate genera probleme conexiunilor securizate.

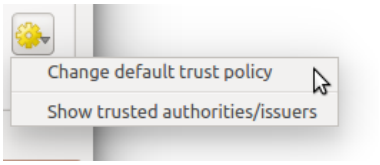


Fig. 25.27: Meniul opțiunilor pentru autorități

You can import CAs or save a file system path from a file that contains multiple CAs, or import individual CAs. The standard PEM format for files that contain multiple CA chain certifications has the root cert at the bottom of the file and all subsequently signed child certificates above, towards the beginning of the file.

The CA certificate import dialog will find all CA certificates within the file, regardless of order, and also offers the option to import certificates that are considered invalid (in case you want to override their trust policy). You can override the trust policy upon import, or do so later within the **Authorities** editor.

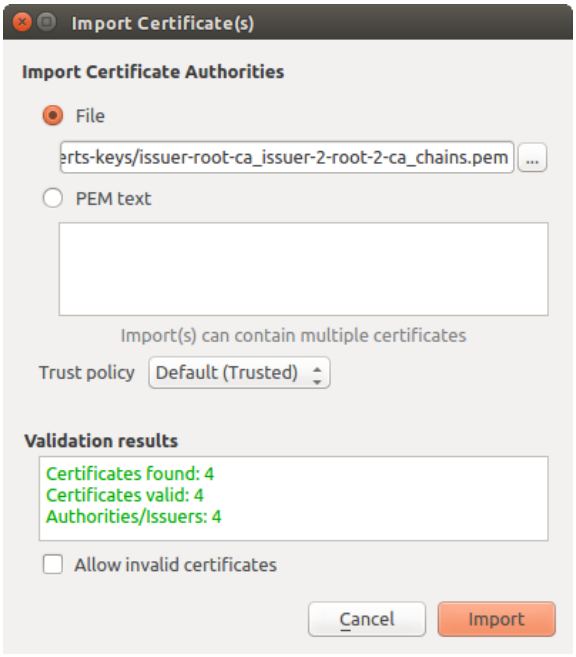


Fig. 25.28: Dialogul de importare a certificatelor

Notă: If you are pasting certificate information into the *PEM text* field, note that encrypted certificates are not supported.

Identități

You can manage available client identity bundles from the *Identities* tab in the *Certificate manager* from the **Authentication** tab of the QGIS **Options** dialog. An identity is what authenticates you against a PKI-enabled service and usually consists of a client certificate and private key, either as separate files or combined into a single „bundled" file. The bundle or private key is often passphrase-protected.

Once you have any Certificate Authorities (CAs) imported you can optionally import any identity bundles into the authentication database. If you do not wish to store the identities, you can reference their component file system paths within an individual authentication configuration.

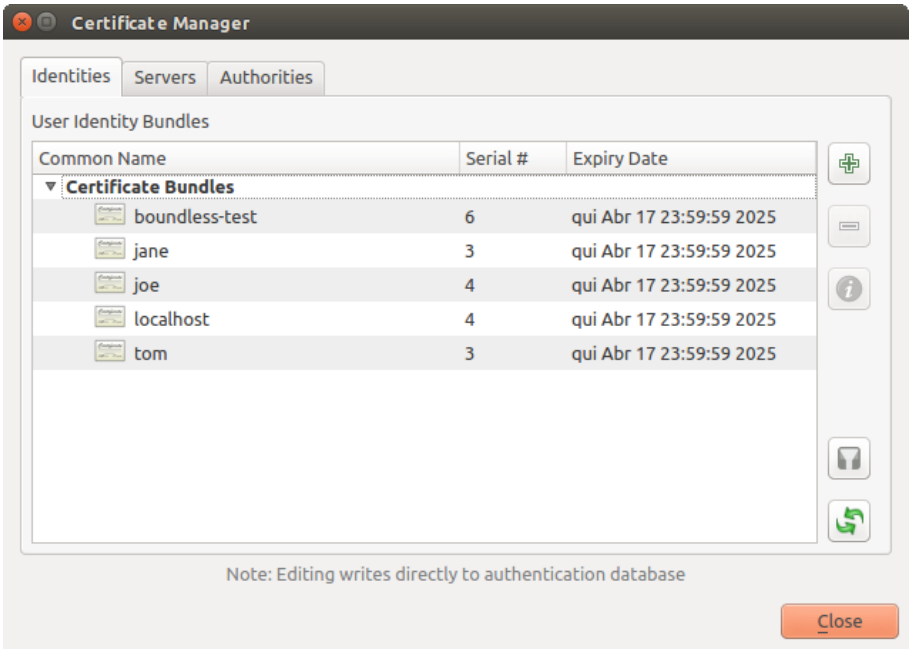


Fig. 25.29: Editorul de Identități

When importing an identity bundle, it can be passphrase-protected or unprotected, and can contain CA certificates forming a trust chain. Trust chain certifications will not be imported here; they can be added separately under the *Authorities* tab.

Upon import the bundle's certificate and private key will be stored in the database, with the key's storage encrypted using the QGIS master password. Subsequent usage of the stored bundle from the database will only require input of the master password.

Personal identity bundles consisting of PEM/DER (.pem/.der) and PKCS#12 (.p12/.pfx) components are supported. If a key or bundle is passphrase-protected, the password will be required to validate the component prior to import. Likewise, if the client certificate in the bundle is invalid (for example, its effective date has not yet started or has elapsed) the bundle can not be imported.

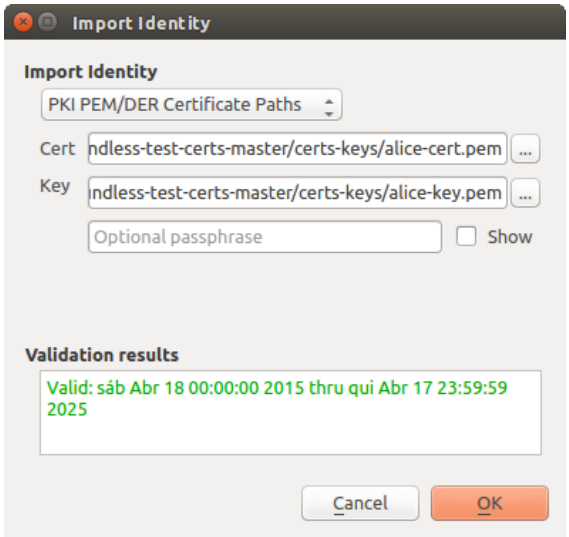


Fig. 25.30: Import de identitate PEM/DER

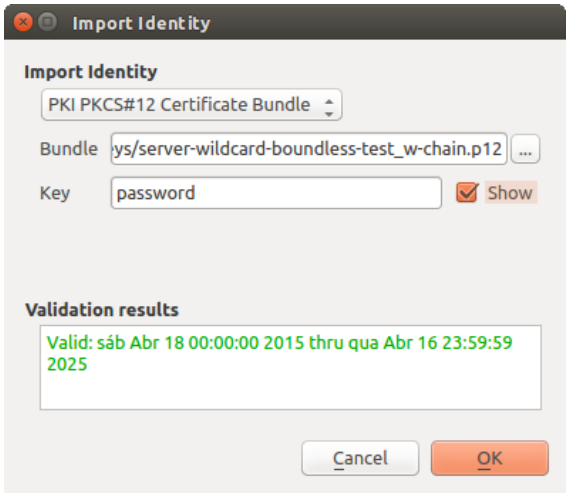


Fig. 25.31: Import de identitate PKCS#12

25.2.4 Gestionarea straturilor eronate

Occasionally, the authentication configuration ID that is saved with a project file is no longer valid, possibly because the current authentication database is different than when the project was last saved, or due to a credentials mismatch. In such cases the *Handle bad layers* dialog will be presented upon QGIS launch.

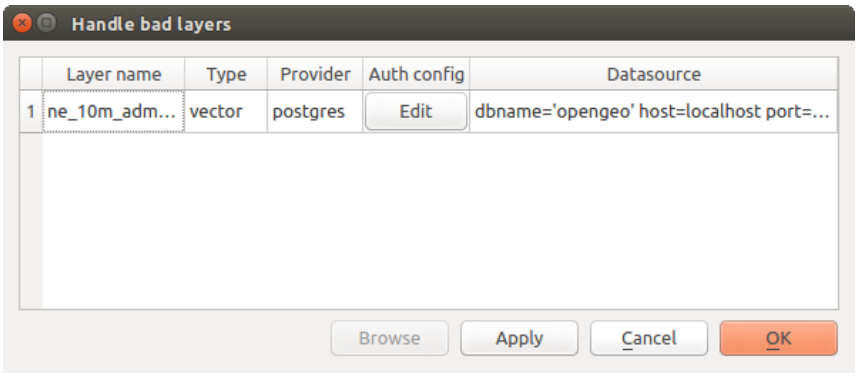


Fig. 25.32: Gestionarea straturilor eronate și a autentificării

If a data source is found to have an authentication configuration ID associated with it, you will be able to edit it. Doing so will automatically edit the data source string, much in the same way as opening the project file in a text editor and editing the string.

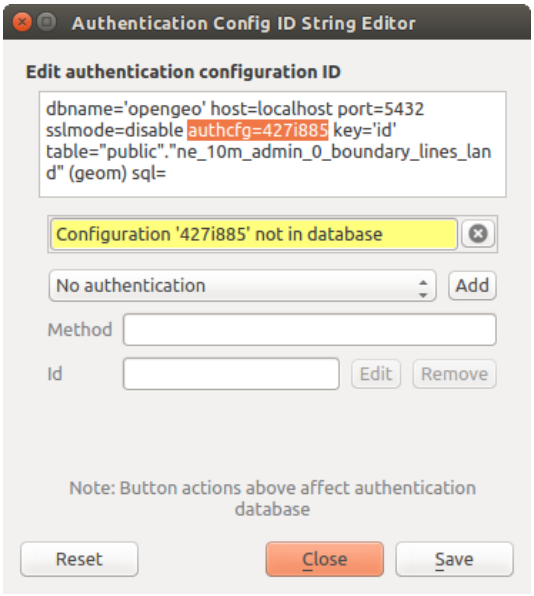


Fig. 25.33: Editarea ID-ului de configurare a autentificării, în cazul straturilor eronate

25.2.5 Schimbarea ID-ului de configurare a autentificării

Occasionally, you will need to change the authentication configuration ID that is associated with accessing a resource. There are instances where this is useful:

- **Resource auth config ID is no longer valid**: This can occur when you have switched auth databases add need to *align* a new configuration to the ID already associated with a resource.
- **Shared project files**: If you intended to share projects between users, e.g. via a shared file server, you can *predefine* a 7-character (containing **a-z** and/or **0-9**) that is associated with the resource. Then, individual users change the ID of an authentication configuration that is specific to their credentials of the resource. When the project is opened, the ID is found in the authentication database, but the credentials are different per user.

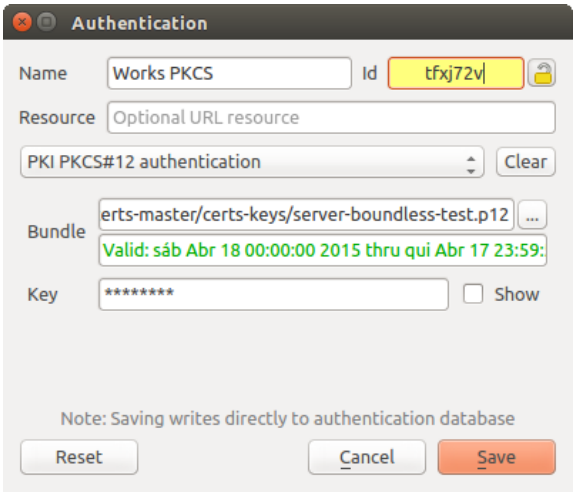


Fig. 25.34: Schimbarea ID-ului de configurare pentru autentificarea stratului (câmp de text galben, deblocat)

Atenționare: Changing the auth config ID is considered an advanced operation and should only be done with full knowledge as to why it is necessary. This is why there is a lock button that needs clicked, to unlock the ID's text field prior to editing the ID.

25.2.6 Suport pentru serverul QGIS

When using a project file, with layers that have authentication configurations, as a basis for a map in QGIS Server, there are a couple of additional setup steps necessary for QGIS to load the resources:

- Baza de date pentru autentificare trebuie să fie disponibilă
- Parola master a bazei de date pentru autentificare trebuie să fie disponibilă

When instantiating the authentication system, Server will create or use `qgis-auth.db` file in the active *user profile*, or the directory defined by the `QGIS_AUTH_DB_DIR_PATH` environment variable. It may be that the Server's user has no HOME directory, in which case, use the environment variable to define a directory that the Server's user has read/write permissions and is not located within the web-accessible directories.

To pass the master password to Server, write it to the first line of file at a path on the file system readable by the Server processes user and defined using the `QGIS_AUTH_PASSWORD_FILE` environment variable. Ensure to limit the file as only readable by the Server's process user and to not store the file within web-accessible directories.

Notă: `QGIS_AUTH_PASSWORD_FILE` variable will be removed from the Server environment immediately after accessing.

25.2.7 Excepții de server SSL

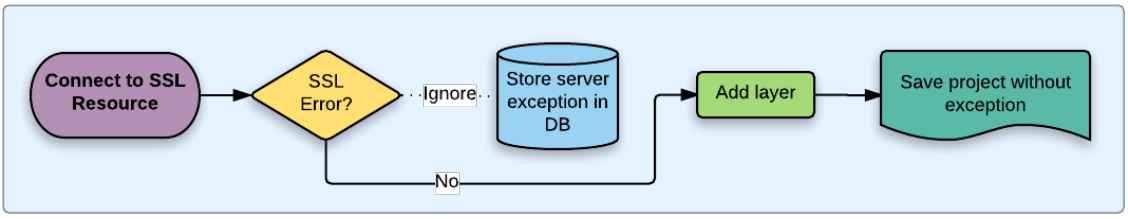


Fig. 25.35: Excepție de server SSL

You can manage SSL server configurations and exceptions from the **Servers** tab in the **Authentication** section of the QGIS **Options** dialog.

Sometimes, when connecting to an SSL server, there are errors with the SSL „handshake" or the server's certificate. You can ignore those errors or create an SSL server configuration as an exception. This is similar to how web browsers allow you to override SSL errors, but with more granular control.

Atenționare: You should not create an SSL server configuration unless you have complete knowledge of the entire SSL setup between the server and client. Instead, report the issue to the server administrator.

Notă: Some PKI setups use a completely different CA trust chain to validate client identities than the chain used to validate the SSL server certificate. In such circumstances, any configuration created for the connecting server will not necessarily fix an issue with the validation of your client identity, and only your client identity's issuer or server administrator can fix the issue.

You can pre-configure an SSL server configuration by clicking the button. Alternatively, you can add a configuration when an SSL error occurs during a connection and you are presented with an **SSL Error** dialog (where the error can be ignored temporarily or saved to the database and ignored):

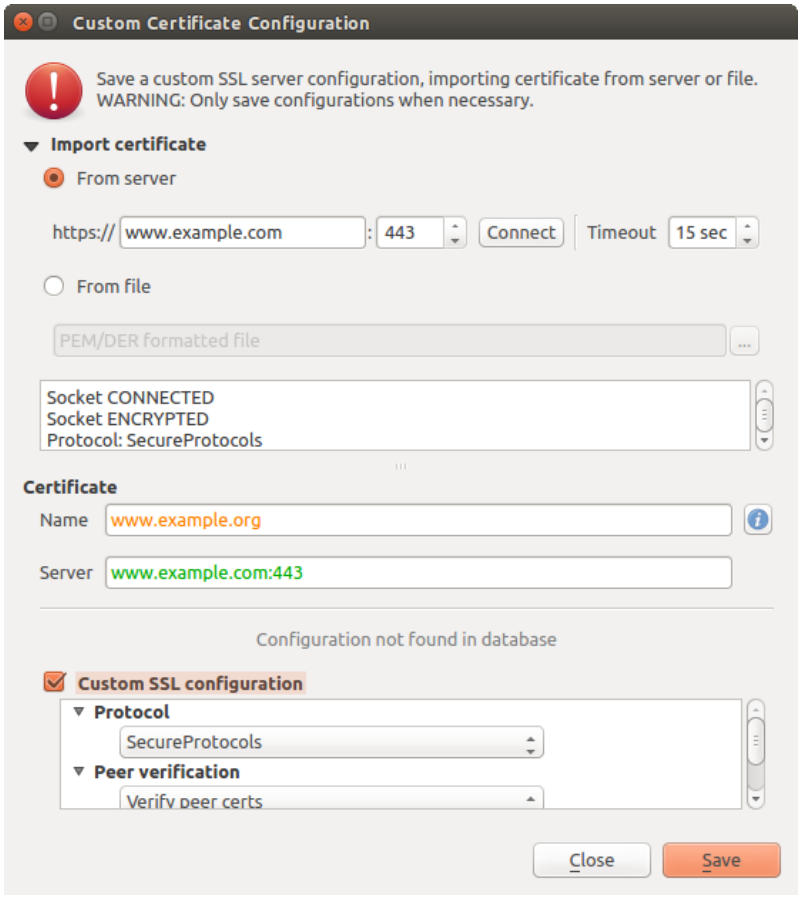


Fig. 25.36: Adăugarea manuală a configurărilor

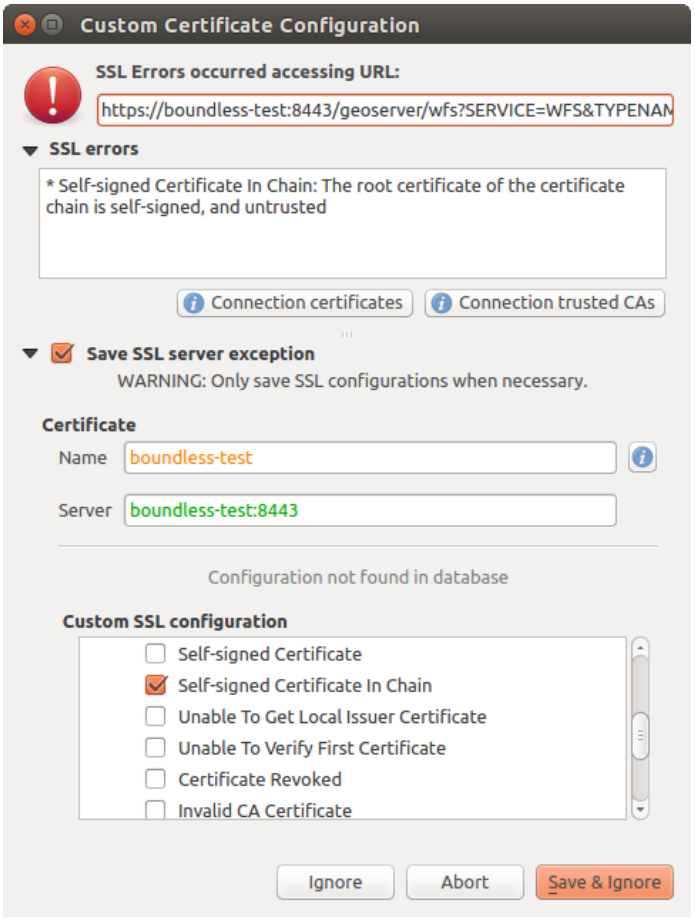


Fig. 25.37: Adăugarea configurărilor pe durata erorii SSL

O dată ce o configurație SSL este salvată în baza de date, aceasta poate fi editată sau ștearsă.

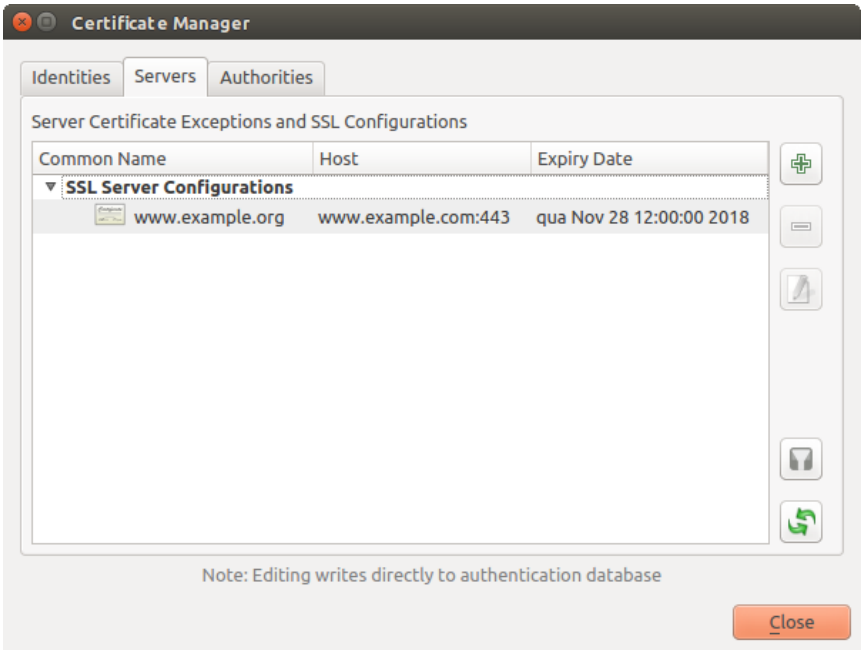


Fig. 25.38: Configurațiile SSL existente

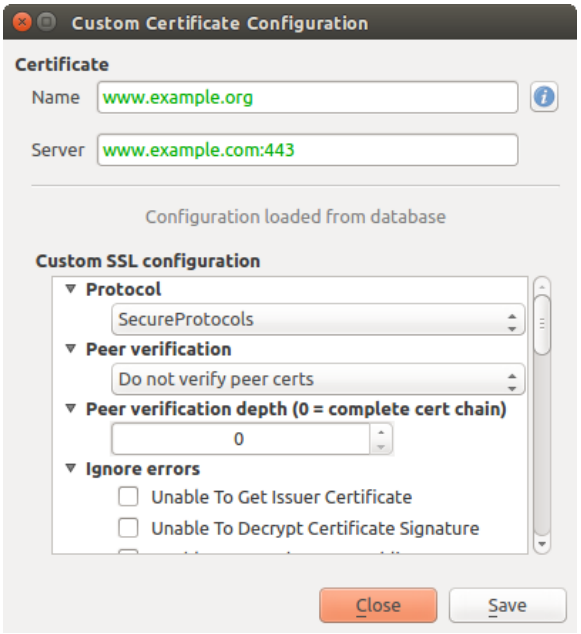


Fig. 25.39: Editarea unei configurații SSL existente

If you want to pre-configure an SSL configuration and the import dialog is not working for your server's connection, you can manually trigger a connection via the **Python Console** by running the following code (replace `https://bugreports.qt-project.org` with the URL of your server):

```
from qgis.PyQt.QtNetwork import QNetworkRequest
from qgis.PyQt.QtCore import QUrl
from qgis.core import QgsNetworkAccessManager

req = QNetworkRequest(QUrl('https://bugreports.qt-project.org'))
reply = QgsNetworkAccessManager.instance().get(req)
```

This will open an SSL error dialog if any errors occur, where you can choose to save the configuration to the database.

25.3 Avertismente de Securitate

O dată ce este introdusă parola de master, API-ul este deschis pentru a vă oferi acces la baza datelor de autentificare, similar modului în care funcționează Firefox. Cu toate acestea, la implementarea inițială, nu sunt definit piedici împotriva accesului PyQGIS. Acest lucru poate conduce la probleme, în cazul în care un utilizator descarcă/instalează un plugin PyQGIS rău intenționat, sau o aplicație de sine stătătoare, care obține acces la prerogativele de autentificare.

Soluția rapidă pentru versiunea inițială a caracteristicii, este de a nu include doar cele mai multe legături PyQGIS pentru sistemul de autentificare.

O altă modalitate simplă, deși nu robustă, este de a adăuga un ComboBox prin *Setări* ► *Opțiuni* ► *Autentificare* (implicit „niciodată”):

```
"Allow Python access to authentication system"
Choices: [ confirm once per session | always confirm | always allow | never]
```

Setarea unei asemenea opțiuni ar trebui să fie salvată într-o locație de bază inaccesibilă pentru Python, cum ar fi baza de date de autentificare, și să fie criptată cu parola principală.

- O altă opțiune ar putea consta în urmărirea plugin-urilor specifice utilizatorului

- allowed to access the authentication system, though it may be tricky to deduce which plugin is actually making the call.
- Sandboxing plugins, possibly in their own virtual environments, would reduce «cross-plugin» hacking of authentication configs from another plugin that is authorized. This might mean limiting cross-plugin communication as well, but maybe only between third-party plugins.
- Another good solution is to issue code-signing certificates to vetted plugin authors. Then validate the plugin's certificate upon loading. If need be the user can also directly set an untrusted policy for the certificate associated with the plugin using existing certificate management dialogs.
- Alternativ, accesați din Python datele sensibile de autentificare la sistem
- could never be allowed, and only the use of QGIS core widgets, or duplicating authentication system integrations, would allow the plugin to work with resources that have an authentication configuration, while keeping master password and authentication config loading in the realm of the main app.

The same security concerns apply to C++ plugins, though it will be harder to restrict access, since there is no function binding to simply be removed as with Python.

25.3.1 Restricții

The confusing licensing and exporting issues associated with OpenSSL apply. In order for Qt to work with SSL certificates, it needs access to the OpenSSL libraries. Depending upon how Qt was compiled, the default is to dynamically link to the OpenSSL libs at run-time (to avoid the export limitations).

QCA follows a similar tactic, whereby linking to QCA incurs no restrictions, because the qca-ossl (OpenSSL) plugin is loaded at run-time. The qca-ossl plugin is directly linked to the OpenSSL libs. Packagers would be the ones needing to ensure any OpenSSL-linking restrictions are met, if they ship the plugin. Maybe. I don't really know. I'm not a lawyer.

The authentication system safely disables itself when `qca-ossl` is not found at run-time.

CAPITOLUL 26

Integrarea GRASS GIS

GRASS integration provides access to GRASS GIS databases and functionalities (see GRASS-PROJECT in *Literatură și Referințe Web*). The integration consists of two parts: provider and plugin. The provider allows to browse, manage and visualize GRASS raster and vector layers. The plugin can be used to create new GRASS locations and mapsets, change GRASS region, create and edit vector layers and analyze GRASS 2-D and 3-D data with more than 400 GRASS modules. In this section, we'll introduce the provider and plugin functionalities and give some examples of managing and working with GRASS data.

The provider supports GRASS version 6 and 7, the plugin supports GRASS 6 and 7 (starting from QGIS 2.12). QGIS distribution may contain provider/plugin for either GRASS 6 or GRASS 7 or for both versions at the same time (binaries have different file names). Only one version of the provider/plugin may be loaded on runtime however.

26.1 Setul de date demonstrative

As an example, we will use the QGIS Alaska dataset (see section *Descărcarea datelor demonstrative*). It includes a small sample GRASS `LOCATION` with three vector layers and one raster elevation map. Create a new folder called `grassdata`, download the QGIS «Alaska» dataset `qgis_sample_data.zip` from https://qgis.org/downloads/data/ and unzip the file into `grassdata`.

More sample GRASS `LOCATIONs` are available at the GRASS website at https://grass.osgeo.org/download/data/.

26.2 Încărcarea straturilor raster și vectoriale GRASS

If the provider is loaded in QGIS, the location item with GRASS icon is added in the browser tree under each folder item which contains GRASS location. Go to the folder `grassdata` and expand location `alaska` and mapset `demo`.

You can load GRASS raster and vector layers like any other layer from the browser either by double click on layer item or by dragging and dropping to map canvas or legend.

Sfat: Încărcarea Datelor GRASS

If you don't see GRASS location item, verify in *Help* ► *About* ► *Providers* if GRASS vector provider is loaded.

26.3 Importing data into a GRASS LOCATION via drag and drop

This section gives an example of how to import raster and vector data into a GRASS mapset.

1. In QGIS browser navigate to the mapset you want to import data into.
2. In QGIS browser find a layer you want to import to GRASS, note that you can open another instance of the browser (*Browser Panel (2)*) if source data are too far from the mapset in the tree.
3. Drag a layer and drop it on the target mapset. The import may take some time for larger layers, you will see animated icon in front of new layer item until the import finishes.

When raster data are in different CRS, they can be reprojected using an *Approximate* (fast) or *Exact* (precise) transformation. If a link to the source raster is created (using `r.external`), the source data are in the same CRS and the format is known to GDAL, the source data CRS will be used. You can set these options in the *Browser* tab in *Opțiuni GRASS*.

If a source raster has more bands, a new GRASS map is created for each layer with **.<band number>** suffix and group of all maps with icon is created. External rasters have a different icon .

26.4 Managing GRASS data in QGIS Browser

- Copying maps: GRASS maps may be copied between mapsets within the same location using drag and drop.
- Deleting maps: Right click on a GRASS map and select *Delete* from context menu.
- Renaming maps: Right click on a GRASS map and select *Rename* from context menu.

26.5 Opțiuni GRASS

GRASS options may be set in *GRASS Options* dialog, which can be opened by right clicking on the location or mapset item in the browser and then choosing *GRASS Options*.

26.6 Startarea plugin-ului GRASS

To use GRASS functionalities in QGIS, you must select and load the GRASS plugin using the Plugin Manager. To do this, go to the menu *Plugins* ► *Manage and Install Plugins…*, select *GRASS* and click *OK*.

The following main features are provided with the GRASS menu (*Plugins* ► *GRASS*) when you start the GRASS plugin:

- Open Mapset
- New Mapset
- Închidere set de hărți
- Deschidere Instrumente GRASS
- Afișarea regiunii curente GRASS
- Opțiuni GRASS

26.7 Deschiderea Setului de hărți GRASS

A GRASS mapset must be opened to get access to GRASS Tools in the plugin (the tools are disabled if no mapset is open). You can open a mapset from the browser: right click on mapset item and then choose *Open mapset* from context menu.

26.8 GRASS LOCATION și MAPSET

GRASS data are stored in a directory referred to as GISDBASE. This directory, often called `grassdata`, must be created before you start working with the GRASS plugin in QGIS. Within this directory, the GRASS GIS data are organized by projects stored in subdirectories called `LOCATIONs`. Each `LOCATION` is defined by its coordinate system, map projection and geographical boundaries. Each `LOCATION` can have several `MAPSETs` (subdirectories of the `LOCATION`) that are used to subdivide the project into different topics or sub-regions, or as workspaces for individual team members (see Neteler & Mitasova 2008 in *Literatură și Referințe Web*). In order to analyze vector and raster layers with GRASS modules, you generally have to import them into a GRASS `LOCATION`. (This is not strictly true – with the GRASS modules `r.external` and `v.external` you can create read-only links to external GDAL-supported datasets without importing them. This is not the usual way for beginners to work with GRASS, therefore this functionality will not be described here.)

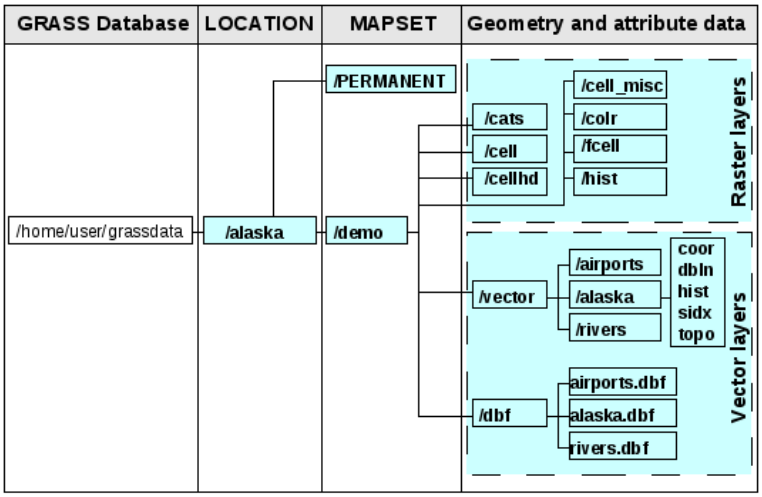


Fig. 26.1: Datele GRASS din LOCAȚIA alaska

26.9 Importați datele într-o LOCAȚIE GRASS

See section *Importing data into a GRASS LOCATION via drag and drop* to find how data can be easily imported by dragging and dropping in the browser.

This section gives an example of how to import raster and vector data into the «alaska» GRASS `LOCATION` provided by the QGIS «Alaska» dataset in traditional way, using standard GRASS modules. Therefore, we use the landcover raster map `landcover.img` and the vector GML file `lakes.gml` from the QGIS «Alaska» dataset (see *Descărcarea datelor demonstrative*).

1. Start QGIS, apoi asigurați-vă că plugin-ul GRASS este încărcat.
2. In the GRASS toolbar, click the Open MAPSET icon to bring up the *MAPSET* wizard.
3. Select as GRASS database the folder `grassdata` in the QGIS Alaska dataset, as `LOCATION` «alaska», as `MAPSET` «demo» and click *OK*.

4. Acum faceți clic pe pictograma Open GRASS tools. Va apărea bara de instrumente GRASS (v. secțiunea *Bara de instrumente GRASS*).
5. To import the raster map `landcover.img`, click the module `r.in.gdal` in the *Modules Tree* tab. This GRASS module allows you to import GDAL-supported raster files into a GRASS `LOCATION`. The module dialog for `r.in.gdal` appears.
6. Răsfoiți folderul `raster` din setul de date «Alaska» din QGIS, apoi selectați fișierul `landcover.img`.
7. As raster output name, define `landcover_grass` and click *Run*. In the *Output* tab, you see the currently running GRASS command `r.in.gdal -o input=/path/to/landcover.img output=landcover_grass`.
8. When it says **Successfully finished**, click *View Output*. The `landcover_grass` raster layer is now imported into GRASS and will be visualized in the QGIS canvas.
9. To import the vector GML file `lakes.gml`, click the module `v.in.ogr` in the *Modules Tree* tab. This GRASS module allows you to import OGR-supported vector files into a GRASS `LOCATION`. The module dialog for `v.in.ogr` appears.
10. Răsfoiți folderul `gml` din setul de date «Alaska» din QGIS, apoi selectați fișierul `lakes.gml` ca fișier OGR.
11. As vector output name, define `lakes_grass` and click *Run*. You don't have to care about the other options in this example. In the *Output* tab you see the currently running GRASS command `v.in.ogr -o dsn=/path/to/lakes.gml output=lakes\_grass`.
12. When it says **Succesfully finished**, click *View Output*. The `lakes_grass` vector layer is now imported into GRASS and will be visualized in the QGIS canvas.

26.9.1 Crearea unei noi LOCAȚII GRASS

Ca exemplu, este prezentat eșantionul GRASS `LOCATION alaska`, care este proiectat în proiecția Albers cu Suprafețe Egale și având feet ca unitate de măsură. Acest eșantion GRASS `LOCATION alaska` va fi folosit pentru toate exemplele și exercițiile din următoarele secțiuni legate de GRASS. Este util să descărcați setul de date pe computerul dvs, apoi să-l instalați (v. *Descărcarea datelor demonstrative*).

1. Start QGIS, apoi asigurați-vă că plugin-ul GRASS este încărcat.
2. Visualize the `alaska.shp` shapefile (see section *Încărcarea unui strat dintr-un fișier*) from the QGIS Alaska dataset (see *Descărcarea datelor demonstrative*).
3. In the GRASS toolbar, click on the New mapset icon to bring up the *MAPSET* wizard.
4. Select an existing GRASS database (GISDBASE) folder `grassdata`, or create one for the new `LOCATION` using a file manager on your computer. Then click *Next*.
5. We can use this wizard to create a new `MAPSET` within an existing `LOCATION` (see section *Adăugarea unui nou MAPSET*) or to create a new `LOCATION` altogether. Select *Create new location* (see Fig. 26.2).
6. Enter a name for the `LOCATION` – we used «alaska» – and click *Next*.
7. Define the projection by clicking on the radio button *Projection* to enable the projection list.
8. We are using Albers Equal Area Alaska (feet) projection. Since we happen to know that it is represented by the EPSG ID 2964, we enter it in the search box. (Note: If you want to repeat this process for another `LOCATION` and projection and haven't memorized the EPSG ID, click on the CRS Status icon in the lower right-hand corner of the status bar (see section *Lucrul cu Proiecții*)).
9. În *Filtrul*, inserați 2964 pentru a selecta proiecția.
10. Clic pe *Next*.
11. To define the default region, we have to enter the `LOCATION` bounds in the north, south, east, and west directions. Here, we simply click on the button *Set Current QGIS Extent*, to apply the extent of the loaded layer `alaska.shp` as the GRASS default region extent.

12. Clic pe *Next*.
13. We also need to define a MAPSET within our new LOCATION (this is necessary when creating a new LOCATION). You can name it whatever you like - we used «demo». GRASS automatically creates a special MAPSET called PERMANENT, designed to store the core data for the project, its default spatial extent and coordinate system definitions (see Neteler & Mitasova 2008 in *Literatură și Referințe Web*).
14. Check out the summary to make sure it's correct and click *Finish*.
15. Sunt create noua LOCATION, «alaska», și două MAPSETs, «demo» și «PERMANENT». Setul deschis în mod curent este «demo», așa cum l-ați definit.
16. Observați că unele instrumente din bara de instrumente GRASS, dezactivate anterior, sunt acum activate.

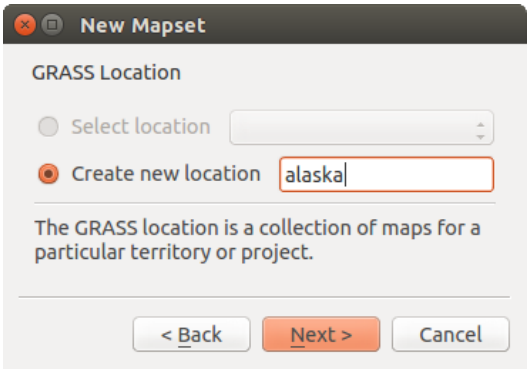


Fig. 26.2: Crearea unei LOCAȚII GRASS noi, sau a unui nou SET DE HĂRȚI în QGIS

If that seemed like a lot of steps, it's really not all that bad and a very quick way to create a LOCATION. The LOCATION «alaska» is now ready for data import (see section *Importați datele într-o LOCAȚIE GRASS*). You can also use the already-existing vector and raster data in the sample GRASS LOCATION «alaska», included in the QGIS «Alaska» dataset *Descărcarea datelor demonstrative*, and move on to section *Modelul de date vectoriale GRASS*.

26.9.2 Adăugarea unui nou MAPSET

A user has write access only to a GRASS MAPSET which he or she created. This means that besides access to your own MAPSET, you can read maps in other users» MAPSETs (and they can read yours), but you can modify or remove only the maps in your own MAPSET.

All MAPSETs include a WIND file that stores the current boundary coordinate values and the currently selected raster resolution (see Neteler & Mitasova 2008 in *Literatură și Referințe Web*, and section *Regiunea instrumentelor GRASS*).

1. Start QGIS, apoi asigurați-vă că plugin-ul GRASS este încărcat.
2. In the GRASS toolbar, click on the New mapset icon to bring up the *MAPSET* wizard.
3. Selectați folderul grassdata al bazei de date GRASS (GISDBASE) cu locația LOCATION «alaska», în care dorim să adăugăm un nou MAPSET denumit «test».
4. Clic pe *Next*.
5. We can use this wizard to create a new MAPSET within an existing LOCATION or to create a new LOCATION altogether. Click on the radio button *Select location* (see Fig. 26.2) and click *Next*.
6. Enter the name test for the new MAPSET. Below in the wizard, you see a list of existing MAPSETs and corresponding owners.
7. Click *Next*, check out the summary to make sure it's all correct and click *Finish*.

26.10 Modelul de date vectoriale GRASS

It is important to understand the GRASS vector data model prior to digitizing. In general, GRASS uses a topological vector model. This means that areas are not represented as closed polygons, but by one or more boundaries. A boundary between two adjacent areas is digitized only once, and it is shared by both areas. Boundaries must be connected and closed without gaps. An area is identified (and labelled) by the **centroid** of the area.

Besides boundaries and centroids, a vector map can also contain points and lines. All these geometry elements can be mixed in one vector and will be represented in different so-called «layers» inside one GRASS vector map. So in GRASS, a layer is not a vector or raster map but a level inside a vector layer. This is important to distinguish carefully. (Although it is possible to mix geometry elements, it is unusual and, even in GRASS, only used in special cases such as vector network analysis. Normally, you should prefer to store different geometry elements in different layers.)

It is possible to store several «layers» in one vector dataset. For example, fields, forests and lakes can be stored in one vector. An adjacent forest and lake can share the same boundary, but they have separate attribute tables. It is also possible to attach attributes to boundaries. An example might be the case where the boundary between a lake and a forest is a road, so it can have a different attribute table.

The «layer» of the feature is defined by the «layer» inside GRASS. «Layer» is the number which defines if there is more than one layer inside the dataset (e.g., if the geometry is forest or lake). For now, it can be only a number. In the future, GRASS will also support names as fields in the user interface.

Atributele pot fi stocate în interiorul `LOCATION` GRASS în format dBase sau SQLite3, sau în tabelele bazei de date externe, cum ar fi PostgreSQL, MySQL, Oracle, etc.

Atributele din tabelele bazei de date sunt legate de elementele geometrice printr-o valoare de «categorie».

«Categoria» (key, ID) este un număr întreg atașat primitivelor geometrice, fiind folosită ca legătură către o coloană cheie, din tabelul bazei de date.

Sfat: Înțelegerea modelului de date vectoriale GRASS

The best way to learn the GRASS vector model and its capabilities is to download one of the many GRASS tutorials where the vector model is described more deeply. See https://grass.osgeo.org/learn/manuals/ for more information, books and tutorials in several languages.

26.11 Crearea unui nou strat vectorial GRASS

To create a new GRASS vector layer, select one of following items from mapset context menu in the browser:

- Strat Nou, de tip Punct
- Strat Nou, de tip Linie
- Strat Nou, de tip Poligon

and enter a name in the dialog. A new vector map will be created and layer will be added to canvas and editing started. Selecting type of the layer does not restrict geometry types which can be digitized in the vector map. In GRASS, it is possible to organize all sorts of geometry types (point, line and polygon) in one vector map. The type is only used to add the layer to the canvas, because QGIS requires a layer to have a specific type.

It is also possible to add layers to existing vector maps selecting one of the items described above from context menu of existing vector map.

In GRASS, it is possible to organize all sorts of geometry types (point, line and area) in one layer, because GRASS uses a topological vector model, so you don't need to select the geometry type when creating a new GRASS vector. This is different from shapefile creation with QGIS, because shapefiles use the Simple Feature vector model (see section *Creating new vector layers*).

26.12 Digitizarea și editarea unui strat vectorial GRASS

GRASS vector layers can be digitized using the standard QGIS digitizing tools. There are however some particularities, which you should know about, due to

- GRASS topological model versus QGIS simple feature
- complexity of GRASS model
  - multiple layers in single maps
  - multiple geometry types in single map
  - geometry sharing by multiple features from multiple layers

The particularities are discussed in the following sections.

Save, discard changes, undo, redo

> **Atenționare:** All the changes done during editing are immediately written to vector map and related attribute tables.

Changes are written after each operation, it is however, possible to do undo/redo or discard all changes when closing editing. If undo or discard changes is used, original state is rewritten in vector map and attribute tables.

There are two main reasons for this behaviour:

- It is the nature of GRASS vectors coming from conviction that user wants to do what he is doing and it is better to have data saved when the work is suddenly interrupted (for example, blackout)
- Necessity for effective editing of topological data is visualized information about topological correctness, such information can only be acquired from GRASS vector map if changes are written to the map.

Bara de Instrumente

The «Digitizing Toolbar» has some specific tools when a GRASS layer is edited:

Pictogramă	Instrument	Scop
	Punct Nou	Digitizare punct nou
	Linie nouă	Digitizare linie nouă
	Limită Nouă	Digitize new boundary
	Centroid Nou	Digitizarea unui nou centroid (etichetarea zonei existente)
	New Closed Boundary	Digitize new closed boundary

Table GRASS Digitizing: GRASS Digitizing Tools

Sfat: Digitizarea poligoanelor în GRASS

If you want to create a polygon in GRASS, you first digitize the boundary of the polygon. Then you add a centroid (label point) into the closed boundary. The reason for this is that a topological vector model links the attribute information of a polygon always to the centroid and not to the boundary.

Category

Category, often called cat, is sort of ID. The name comes from times when GRASS vectors had only singly attribute „category”. Category is used as a link between geometry and attributes. A single geometry may have multiple categories and thus represent multiple features in different layers. Currently it is possible to assign only one category per layer using QGIS editing tools. New features have automatically assigned new unique category, except boundaries.

Boundaries usually only form areas and do not represent linear features, it is however possible to define attributes for a boundary later, for example in different layer.

New categories are always created only in currently being edited layer.

It is not possible to assign more categories to geometry using QGIS editing, such data are properly represented as multiple features, and individual features, even from different layers, may be deleted.

Atribute

Attributes of currently edited layer can only be modified. If the vector map contains more layers, features of other layers will have all attributes set to «<not editable (layer #)>» to warn you that such attribute is not editable. The reason is, that other layers may have and usually have different set of fields while QGIS only supports one fixed set of fields per layer.

If a geometry primitive does not have a category assigned, a new unique category is automatically assigned and new record in attribute table is created when an attribute of that geometry is changed.

Sfat: If you want to do bulk update of attributes in table, for example using «Field Calculator» (*Using the Field Calculator*), and there are features without category which you don't want to update (typically boundaries), you can filter them out by setting «Advanced Filter» to `cat is not null`.

Editing style

The topological symbology is essential for effective editing of topological data. When editing starts, a specialized «GRASS Edit» renderer is set on the layer automatically and original renderer is restored when editing is closed. The style may be customized in layer properties «Style» tab. The style can also be stored in project file or in separate file as any other style. If you customize the style, do not change its name, because it is used to reset the style when editing is started again.

Sfat: Do not save project file when the layer is edited, the layer would be stored with «Edit Style» which has no meaning if layer is not edited.

The style is based on topological information which is temporarily added to attribute table as field «topo_symbol». The field is automatically removed when editing is closed.

Sfat: Do not remove «topo_symbol» field from attribute table, that would make features invisible because the renderer is based on that column.

Acroşarea

To form an area, vertices of connected boundaries must have **exactly** the same coordinates. This can be achieved using snapping tool only if canvas and vector map have the same CRS. Otherwise, due conversion from map coordinates to canvas and back, the coordinate may become slightly different due to representation error and CRS transformations.

Sfat: Canvasul folosește CRS-ul stratului și la momentul editării.

Limitări

Simultaneous editing of multiple layers within the same vector at the same time is not supported. This is mainly due to the impossibility of handling multiple undo stacks for a single data source.

On Linux and macOS only one GRASS layer can be edited at time. This is due to a bug in GRASS which does not allow to close database drivers in random order. This is being solved with GRASS developers.

Sfat: Permisiuni de Editare GRASS

Trebuie să fiți proprietarul MAPSET GRASS, pentru a-l putea edita. Este imposibilă editarea datelor din straturile MAPSET care nu vă aparține, chiar dacă aveți permisiunea de scriere.

26.13 Regiunea instrumentelor GRASS

The region definition (setting a spatial working window) in GRASS is important for working with raster layers. Vector analysis is by default not limited to any defined region definitions. But all newly created rasters will have the spatial extension and resolution of the currently defined GRASS region, regardless of their original extension and resolution. The current GRASS region is stored in the $LOCATION/$MAPSET/WIND file, and it defines north, south, east and west bounds, number of columns and rows, horizontal and vertical spatial resolution.

It is possible to switch on and off the visualization of the GRASS region in the QGIS canvas using the
Display current GRASS region button.

The region can be modified in «Region» tab in «GRASS Tolls» dock widget. Type in the new region bounds and resolution, and click *Apply*. If you click on *Select the extent by dragging on canvas* you can select a new region interactively with your mouse on the QGIS canvas dragging a rectangle.

The GRASS module `g.region` provides a lot more parameters to define an appropriate region extent and resolution for your raster analysis. You can use these parameters with the GRASS Toolbox, described in section *Bara de instrumente GRASS*.

26.14 Bara de instrumente GRASS

The
Open GRASS Tools box provides GRASS module functionalities to work with data inside a selected GRASS LOCATION and MAPSET. To use the GRASS Toolbox you need to open a LOCATION and MAPSET that you have write permission for (usually granted, if you created the MAPSET). This is necessary, because new raster or vector layers created during analysis need to be written to the currently selected LOCATION and MAPSET.

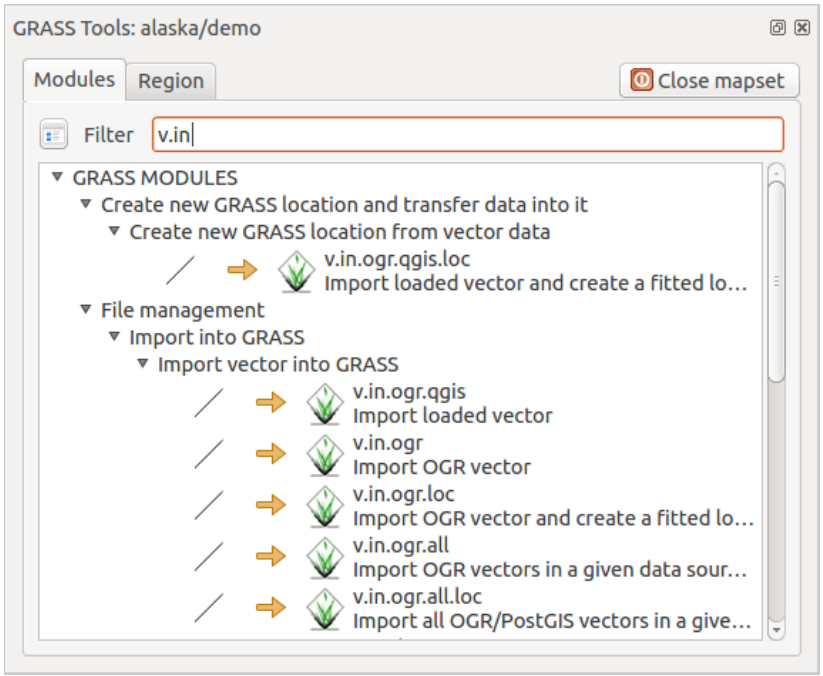


Fig. 26.3: GRASS Toolbox and Module Tree

26.14.1 Lucrul cu modulele GRASS

The GRASS shell inside the GRASS Toolbox provides access to almost all (more than 300) GRASS modules in a command line interface. To offer a more user-friendly working environment, about 200 of the available GRASS modules and functionalities are also provided by graphical dialogs within the GRASS plugin Toolbox.

A complete list of GRASS modules available in the graphical Toolbox in QGIS version 3.34 is available in the GRASS wiki at https://grasswiki.osgeo.org/wiki/GRASS-QGIS_relevant_module_list.

De asemenea, este posibilă personalizarea conținutul Instrumentarului GRASS. Această procedură este descrisă în secțiunea *Personalizarea Barei de Instrumente GRASS*.

As shown in Fig. 26.3, you can look for the appropriate GRASS module using the thematically grouped *Modules Tree* or the searchable *Modules List* tab.

By clicking on a graphical module icon, a new tab will be added to the Toolbox dialog, providing three new sub-tabs: *Options*, *Output* and *Manual*.

Opțiuni

The *Options* tab provides a simplified module dialog where you can usually select a raster or vector layer visualized in the QGIS canvas and enter further module-specific parameters to run the module.

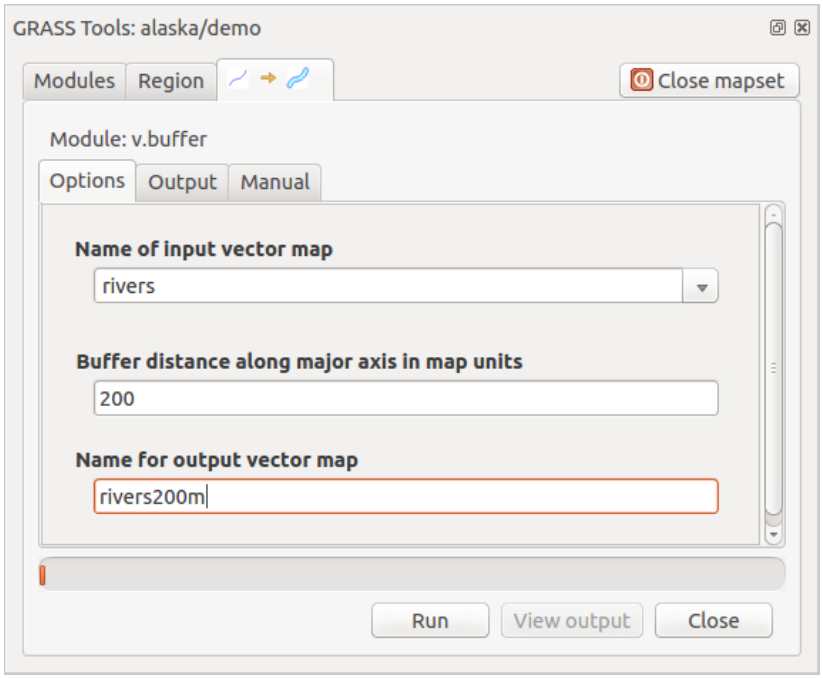


Fig. 26.4: GRASS Toolbox Module Options

The provided module parameters are often not complete to keep the dialog simple. If you want to use further module parameters and flags, you need to start the GRASS shell and run the module in the command line.

A new feature since QGIS 1.8 is the support for a *Show Advanced Options* button below the simplified module dialog in the *Options* tab. At the moment, it is only added to the module `v.in.ascii` as an example of use, but it will probably be part of more or all modules in the GRASS Toolbox in future versions of QGIS. This allows you to use the complete GRASS module options without the need to switch to the GRASS shell.

Rezultat

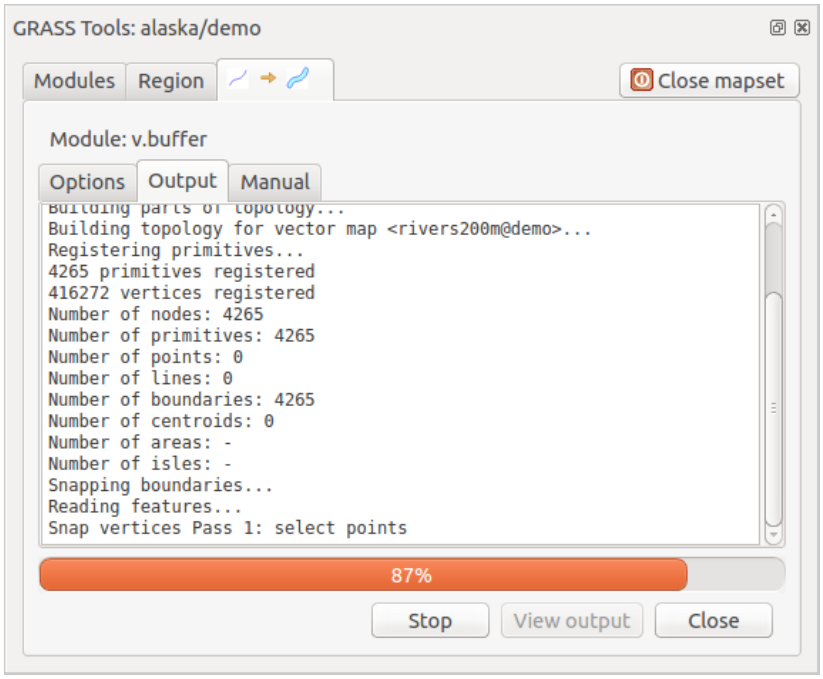


Fig. 26.5: GRASS Toolbox Module Output

The *Output* tab provides information about the output status of the module. When you click the *Run* button, the module switches to the *Output* tab and you see information about the analysis process. If all works well, you will finally see a `Successfully finished` message.

Manual

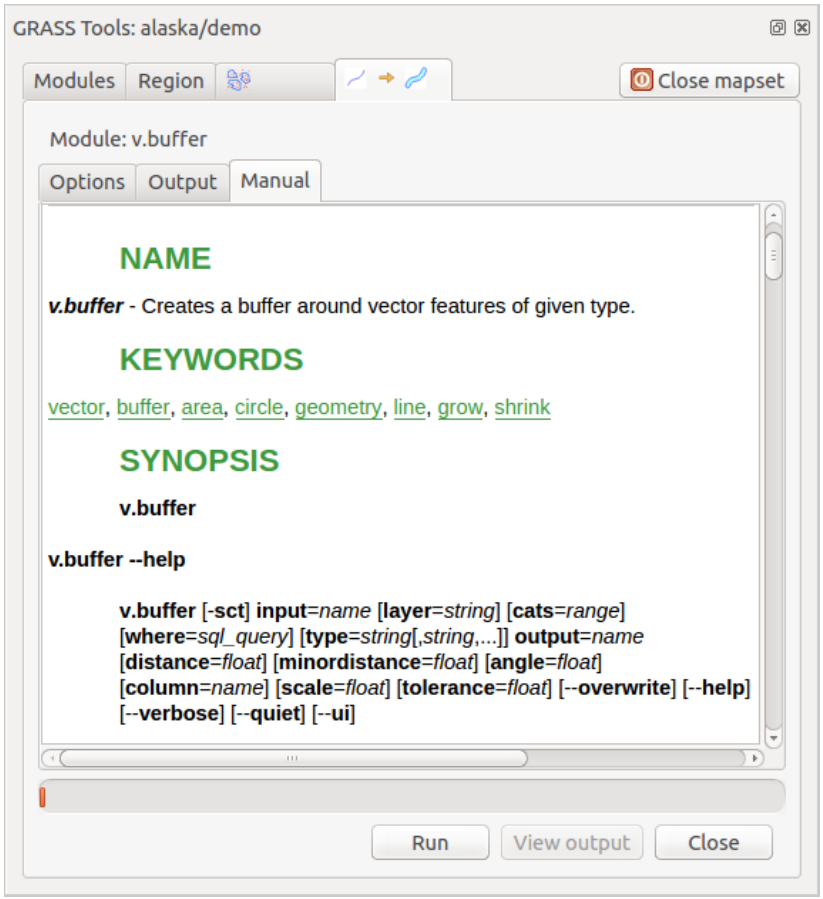


Fig. 26.6: GRASS Toolbox Module Manual

The *Manual* tab shows the HTML help page of the GRASS module. You can use it to check further module parameters and flags or to get a deeper knowledge about the purpose of the module. At the end of each module manual page, you see further links to the `Main Help index`, the `Thematic index` and the `Full index`. These links provide the same information as the module `g.manual`.

Sfat: Afişează imediat rezultatele

Dacă doriți să afișați imediat rezultatele calculelor dvs în canevasul hărții, puteți folosi butonul «Vizualizare Output», din partea de jos a filei modulului.

26.14.2 Exemple de module GRASS

Următoarele exemple vor demonstra puterea unora dintre modulele GRASS.

Crearea curbelor de nivel

The first example creates a vector contour map from an elevation raster (DEM). Here, it is assumed that you have the Alaska `LOCATION` set up as explained in section *Importați datele într-o LOCAȚIE GRASS*.

- First, open the location by clicking the Open mapset button and choosing the Alaska location.
- Now open the Toolbox with the Open GRASS tools button.
- În lista de de unelte pentru categorii, faceți dublu-clic pe *Raster* ► *Surface Management* ► *Generate vector contour lines*.
- Now a single click on the tool **r.contour** will open the tool dialog as explained above (see *Lucrul cu modulele GRASS*).
- In the *Name of input raster map* enter `gtopo30`.
- Type into the *Increment between Contour levels* the value 100. (This will create contour lines at intervals of 100 meters.)
- Introduceți în *Name for output vector map `numele ``ctour_100`*.
- Click *Run* to start the process. Wait for several moments until the message `Successfully finished` appears in the output window. Then click *View Output* and *Close*.

Deoarece aceasta este o regiune mare, va dura ceva timp până la afișare. După ce se termină randarea, puteți deschide fereastra cu proprietățile stratului, pentru a schimba culoarea liniei astfel încât conturul să apară clar pe rasterul de elevație, la fel ca în *Dialogul Proprietăților Vectoriale*.

Next, zoom in to a small, mountainous area in the center of Alaska. Zooming in close, you will notice that the contours have sharp corners. GRASS offers the **v.generalize** tool to slightly alter vector maps while keeping their overall shape. The tool uses several different algorithms with different purposes. Some of the algorithms (i.e., Douglas Peuker and Vertex Reduction) simplify the line by removing some of the vertices. The resulting vector will load faster. This process is useful when you have a highly detailed vector, but you are creating a very small-scale map, so the detail is unnecessary.

Sfat: Instrumentul de simplificare

Note that QGIS has a *Vector* ► *Geometry Tools* ► *Simplify geometries* tool that works just like the GRASS **v.generalize** Douglas-Peuker algorithm.

However, the purpose of this example is different. The contour lines created by `r.contour` have sharp angles that should be smoothed. Among the **v.generalize** algorithms, there is Chaiken's, which does just that (also Hermite splines). Be aware that these algorithms can **add** additional vertices to the vector, causing it to load even more slowly.

- Open the GRASS Toolbox and double-click the categories *Vector* ► *Develop map* ► *Generalization*, then click on the **v.generalize** module to open its options window.
- Verificați dacă «ctour_100» apare ca *Nume pentru vectorul de intrare*.
- From the list of algorithms, choose Chaiken's. Leave all other options at their default, and scroll down to the last row to enter in the field *Name for output vector map* «ctour_100_smooth», and click *Run*.
- The process takes several moments. Once `Successfully finished` appears in the output windows, click *View Output* and then *Close*.
- Puteți schimba culoarea vectorului pentru a-l afișa în mod clar pe fundalul raster, și pentru a contrasta față de curbele de nivel originale. Veți observa că noile curbe de nivel au colțuri mai fine decât originalul, în timp ce urmează fidel forma originală.

Fig. 26.7: GRASS module v.generalize to smooth a vector map

Sfat: Alte utilizări pentru r.contour

The procedure described above can be used in other equivalent situations. If you have a raster map of precipitation data, for example, then the same method will be used to create a vector map of isohyetal (constant rainfall) lines.

Crearea unui efect 3-D de umbrire

Several methods are used to display elevation layers and give a 3-D effect to maps. The use of contour lines, as shown above, is one popular method often chosen to produce topographic maps. Another way to display a 3-D effect is by hillshading. The hillshade effect is created from a DEM (elevation) raster by first calculating the slope and aspect of each cell, then simulating the sun's position in the sky and giving a reflectance value to each cell. Thus, you get sun-facing slopes lighted; the slopes facing away from the sun (in shadow) are darkened.

- Begin this example by loading the `gtopo30` elevation raster. Start the GRASS Toolbox, and under the Raster category, double-click to open *Spatial analysis* ► *Terrain analysis*.
- Apoi faceți clic pe **r.shaded.relief** pentru a deschide modulul.
- Change the *azimuth angle* 1,00 270 to 315.
- Enter `gtopo30_shade` for the new hillshade raster, and click *Run*.
- Când procesul se încheie, adăugați hărții rasterul reliefat. Ar trebui să-l vedeți afișat în tonuri de gri.
- To view both the hillshading and the colors of the `gtopo30` together, move the hillshade map below the `gtopo30` map in the table of contents, then open the *Properties* window of `gtopo30`, switch to the *Transparency* tab and set its transparency level to about 25%.

Ar trebui să aveți acum elevația `gtopo30` cu harta de culori și transparența setate **deasupra** hărții reliefului, în tonuri de gri. Pentru a observa mai bine efectele vizuale ale reliefării, desetați vizualizarea hărții `gtopo30_shade`, apoi resetați-o.

Folosirea consolei GRASS

The GRASS plugin in QGIS is designed for users who are new to GRASS and not familiar with all the modules and options. As such, some modules in the Toolbox do not show all the options available, and some modules do not appear at all. The GRASS shell (or console) gives the user access to those additional GRASS modules that do not appear in the Toolbox tree, and also to some additional options to the modules that are in the Toolbox with the simplest default parameters. This example demonstrates the use of an additional option in the **r.shaded.relief** module that was shown above.

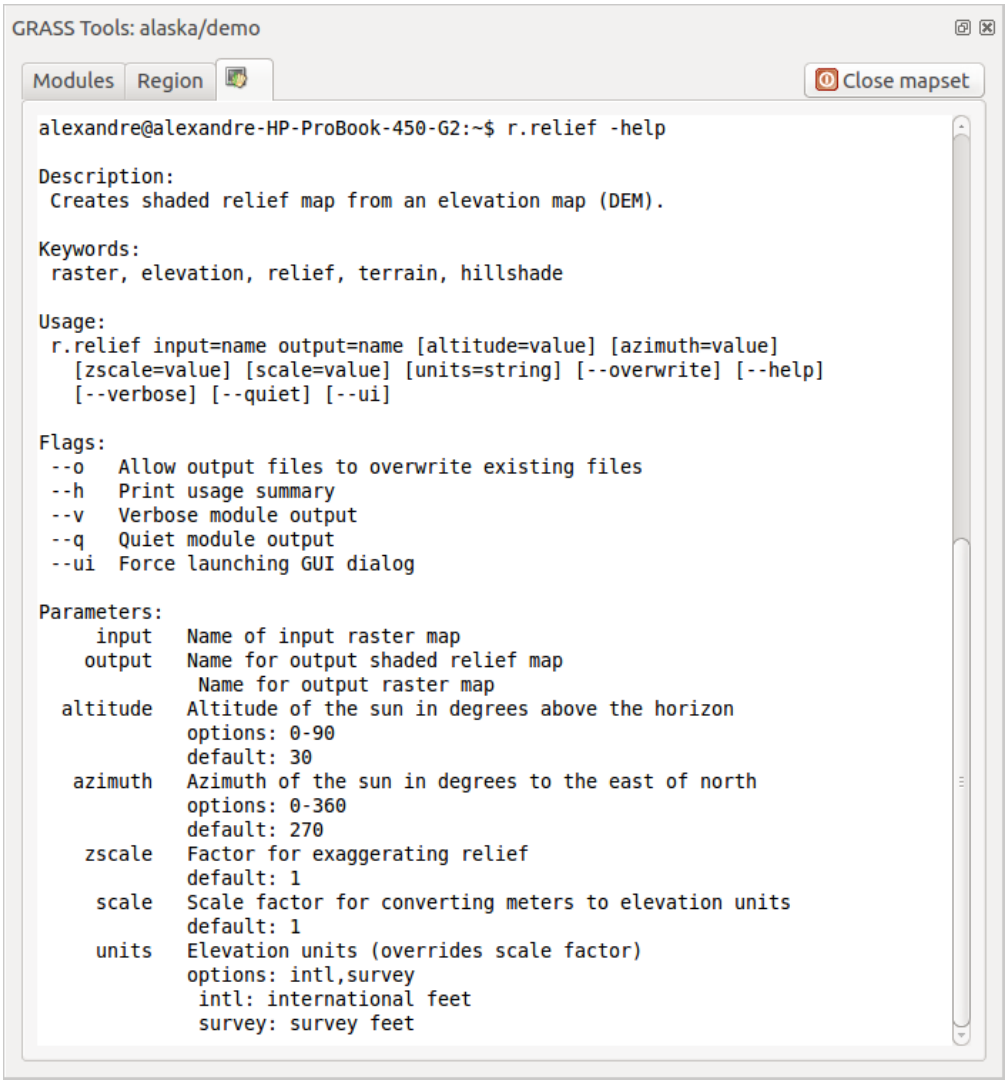


Fig. 26.8: The GRASS shell, r.shaded.relief module

The module **r.shaded.relief** can take a parameter `zmult`, which multiplies the elevation values relative to the X-Y coordinate units so that the hillshade effect is even more pronounced.

- Load the `gtopo30` elevation raster as above, then start the GRASS Toolbox and click on the GRASS shell. In the shell window, type the command `r.shaded.relief map=gtopo30 shade=gtopo30_shade2 azimuth=315 zmult=3` and press `Enter`.
- After the process finishes, shift to the *Browse* tab and double-click on the new `gtopo30_shade2` raster to display it in QGIS.
- As explained above, move the shaded relief raster below the `gtopo30` raster in the table of contents, then check the transparency of the colored `gtopo30` layer. You should see that the 3-D effect stands out more strongly compared with the first shaded relief map.

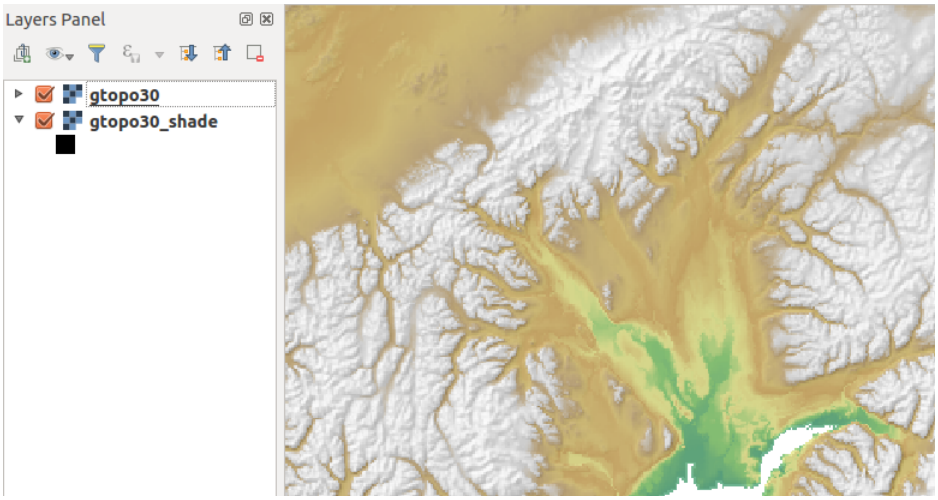


Fig. 26.9: Displaying shaded relief created with the GRASS module r.shaded.relief

Statistici raster pentru o hartă vectorială

Următorul exemplu arată modul în care un modul din GRASS poate agrega datele rastere, apoi să adauge coloanele de statistici pentru fiecare poligon din harta vectorială.

- Again using the Alaska data, refer to *Importați datele într-o LOCAȚIE GRASS* to import the `shapefiles/trees.shp` file into GRASS.
- Now an intermediate step is required: centroids must be added to the imported trees map to make it a complete GRASS area vector (including both boundaries and centroids).
- Din Bara de instrumente alegeți *Vector* ► *Manage features*, apoi deschideți modulul **v.centroids**.
- Introduceți «forest_areas» pentru *output vector map*, apoi rulați modulul.
- Now load the `forest_areas` vector and display the types of forests - deciduous, evergreen, mixed - in different colors: In the layer *Properties* window, *Symbology* tab, choose from *Legend type* «Unique value» and set the *Classification field* to «VEGDESC». (Refer to the explanation of the symbology tab in *Symbology Properties* of the vector section.)
- Mai departe, redeschideți Bara de instrumente GRASS, apoi deschideți *Vector* ► *Vector update* din alte hărți.
- Clic pe modulul **v.rast.stats**. Introduceți `gtopo30` și `forest_areas`.
- Only one additional parameter is needed: Enter *column prefix* `elev`, and click *Run*. This is a computationally heavy operation, which will run for a long time (probably up to two hours).
- Finally, open the `forest_areas` attribute table, and verify that several new columns have been added, including `elev_min`, `elev_max`, `elev_mean`, etc., for each forest polygon.

26.14.3 Personalizarea Barei de Instrumente GRASS

Nearly all GRASS modules can be added to the GRASS Toolbox. An XML interface is provided to parse the pretty simple XML files that configure the modules» appearance and parameters inside the Toolbox.

Un fișier XML eșantion, pentru generarea modulului `v.buffer` (v.buffer.qgm) arată în felul următor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE qgisgrassmodule SYSTEM "http://mrcc.com/qgisgrassmodule.dtd">

<qgisgrassmodule label="Vector buffer" module="v.buffer">
        <option key="input" typeoption="type" layeroption="layer" />
```

(continues on next page)

(continuare din pagina precedentă)

```
        <option key="buffer"/>
        <option key="output" />
</qgisgrassmodule>
```

The parser reads this definition and creates a new tab inside the Toolbox when you select the module. A more detailed description for adding new modules, changing a module's group, etc., can be found at https://qgis.org/en/site/getinvolved/development/addinggrasstools.html.

CAPITOLUL 27

Cadrul de procesare al QGIS

27.1 Introducere

Acest capitol introduce cadrul de prelucrare al QGIS, un mediu de geoprocesare care poate fi folosit pentru a apela algoritmi nativi sau ai unei terțe părți din QGIS, făcând sarcinile de analiză spațială mai productive și mai ușor de realizat.

As a *Core plugin*, Processing is installed by default but you need to activate it:

1. Go to *Plugins* ► *Manage and install plugins…*
2. Click on the *Installed* tab at the left
3. Check the box next to the *Processing* entry
4. Închideți dialogul.

   A *Processing* menu is now available in the top menu bar. From there you can reach the main components of this framework.

În următoarele secțiuni, vom examina modul de a folosi elementele grafice ale acestui cadru și de a obține mai mult de la fiecare.

There are four basic elements in the framework GUI, which are used to run algorithms for different purposes. Choosing one tool or another will depend on the kind of analysis that is to be performed and the particular characteristics of each user and project. All of them (except for the batch processing interface, which is called from the toolbox or the algorithm execution dialog, as we will see) can be accessed from the *Processing* menu item (you will see more entries; the remaining ones are not used to execute algorithms and will be explained later in this chapter).

- The *Toolbox*: The main element of the GUI, it is used to execute a single algorithm or run a batch process based on that algorithm.

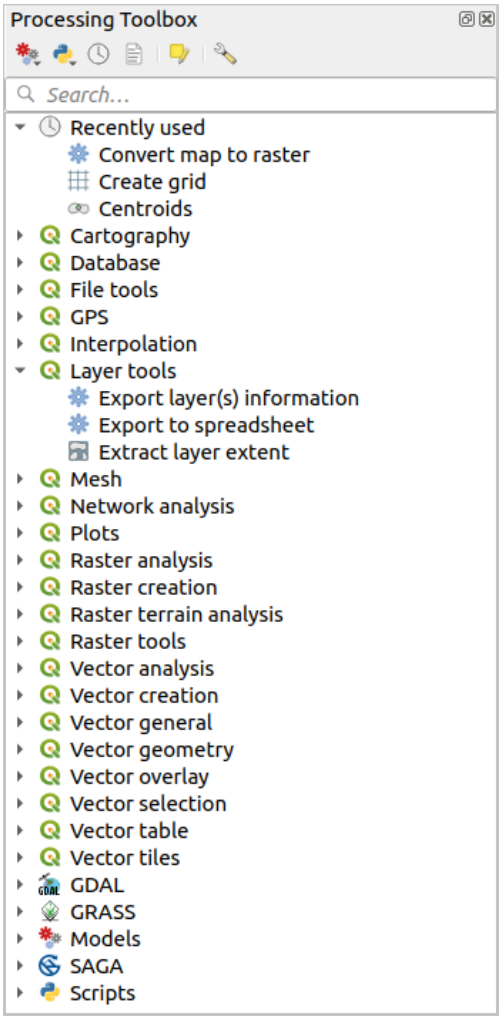


Fig. 27.1: Bara Instrumentelor Processing

- The *Model Designer*: Several algorithms can be combined graphically using the modeler to define a workflow, creating a single process that involves several subprocesses.

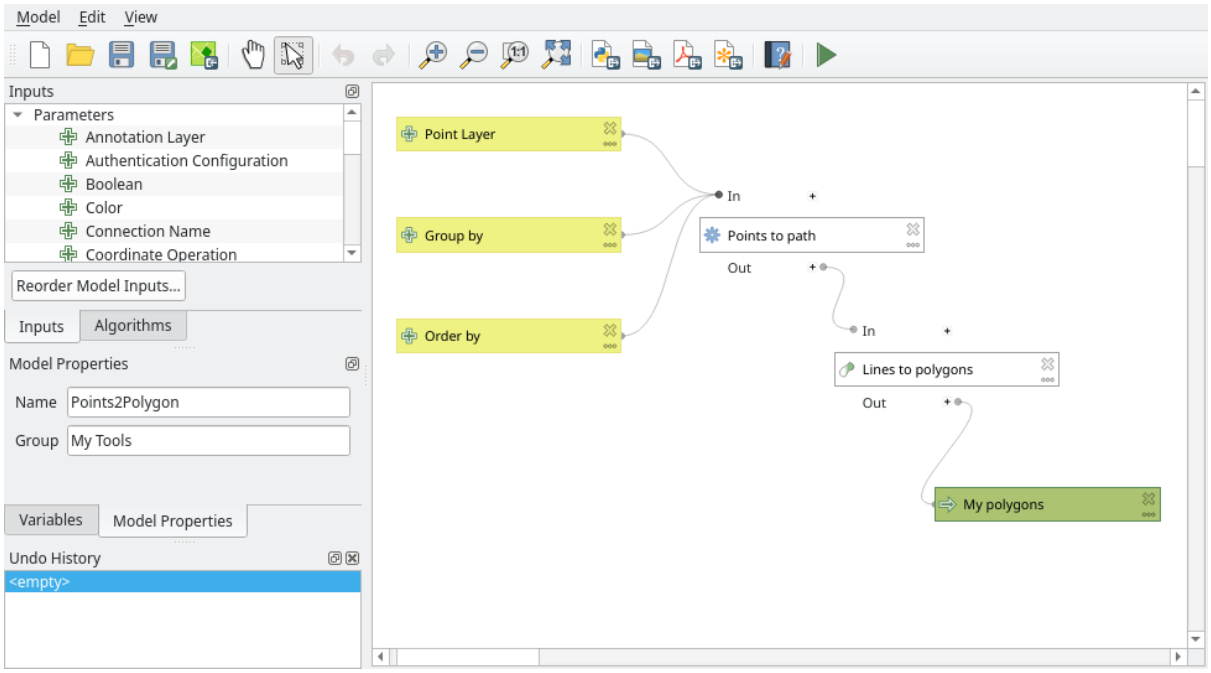


Fig. 27.2: Modelatorul Processing

- The *History* manager: All actions performed using any of the aforementioned elements are stored in a history file and can be later easily reproduced using the history manager.

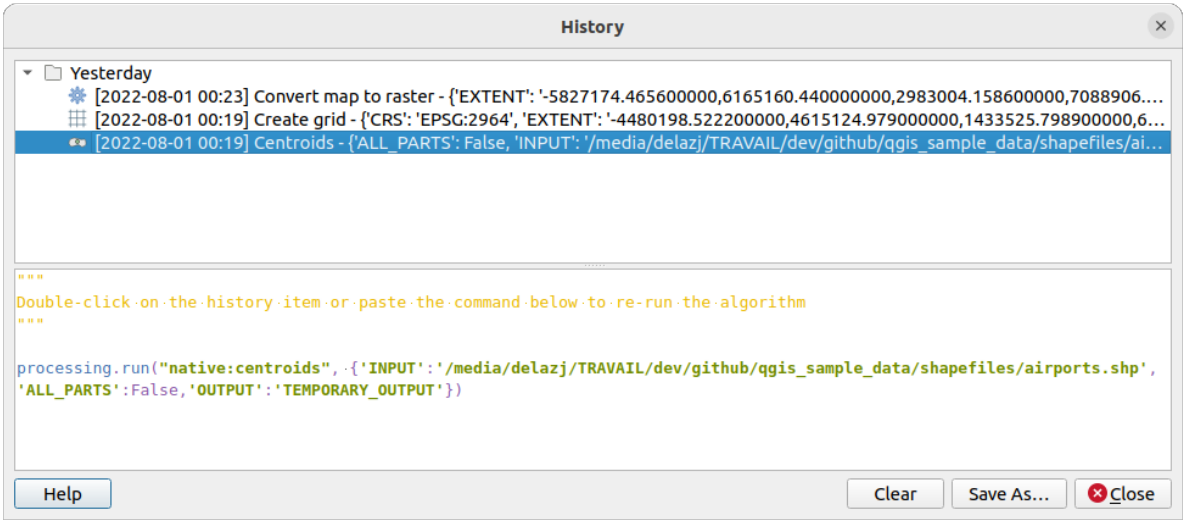


Fig. 27.3: Istoricul Processing

- The *Batch Processing* interface: This interface allows you to execute batch processes and automate the execution of a single algorithm on multiple datasets.

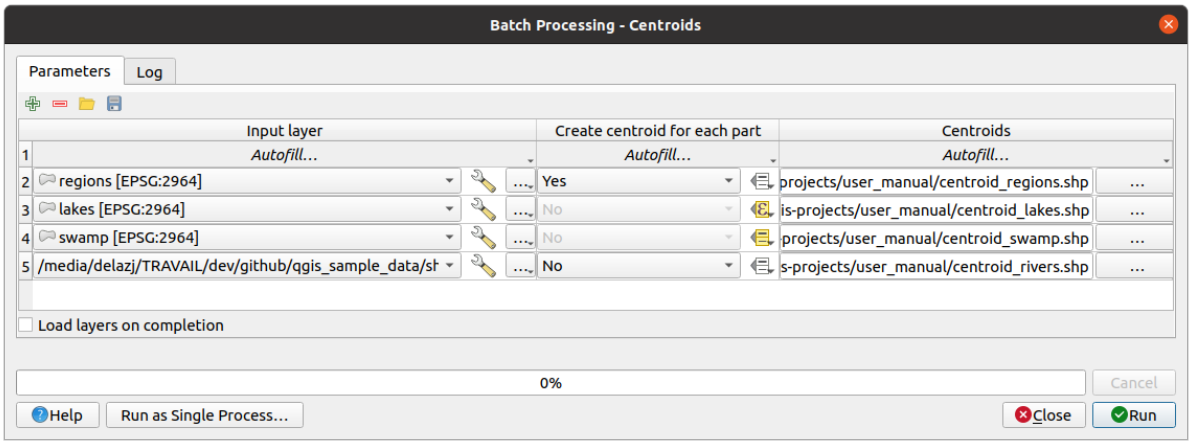


Fig. 27.4: Interfața de Procesare în Serie

În următoarele secțiuni, vom examina în detaliu fiecare dintre aceste elemente.

27.2 Configuring the Processing Framework

The Processing Options menu (*Settings*► *Options* ► *Processing* tab) allows you to configure how algorithms work. Configuration parameters are structured in separate blocks that you can select on the left-hand side of the dialog.

27.2.1 General

The *General* block contains the default settings to control how the algorithm dialog as well as input or output parameters should behave. Some of the settings can however be overridden at the algorithm level, per *algorithm run* or for *individual parameters*.

- *Default output raster layer extension* is by default `tif`
- *Default output vector layer extension* is by default `gpkg`
- *Invalid features filtering* when executing algorithm:
  - *Do not filter (better performance)*: all the features (with valid and invalid geometries) are processed, but the result may be erroneous depending on how the geometry invalidity affects the operations
  - *Skip (ignore) features with invalid geometries*, meaning that only a subset of your dataset (the valid geometry features) will be processed
  - *Stop algorithm execution when a geometry is invalid*: you'll need to track and fix the invalid geometries if you want the algorithm to process the whole layer. Algorithms like *Check validity* or *Fix geometries* can help you achieve this.

  The *Invalid features filtering* setting can be overridden on a per-input basis, at algorithm runtime.
- *Keep dialog open after running algorithm*. Once an algorithm has finished execution and its output layers are loaded into the QGIS project, the algorithm dialog is closed. If you want to keep it open (to run the algorithm again with different parameters, or to better check the output that is written to the log tab), check this option.
- *Max Threads*
- *Output folder* for non temporary outputs: If no folder path is provided for the Processing execution outputs, this is the folder in which they will be saved. Default is `processing/outputs` under the active *user profile* directory.

- *Override temporary output folder path*: Temporary outputs are saved by default in the `tmp` folder on the machine. This option helps you set a different place for storage.
- *Pre-execution script* and *Post-execution script*. These parameters point to files that contain scripts written using the processing scripting functionality, explained in the section covering scripting and the console.
- *Prefer output filename for layer names*. The name of each resulting layer created by an algorithm is defined by the algorithm itself. In some cases, a fixed name might be used, meaning that the same output name will be used, no matter which input layer is used. In other cases, the name might depend on the name of the input layer or some of the parameters used to run the algorithm. If this checkbox is checked, the name will be taken from the output filename instead. Notice that, if the output is saved to a temporary file, the filename of this temporary file is usually a long and meaningless one intended to avoid collision with other already existing filenames.
- *Results group name*. If you want to obtain all processing result layers in a group in the *Layers* panel, set a group name for this parameter. The group may exist already or not. QGIS will add all output layers to such a group. By default, this parameter is empty, so all output layers are added to different places in the *Layers* panel, depending on the item that is active when running an algorithm. Note that output layers will be loaded to the *Layers* panel only if *Open output file after running algorithm* is checked in the algorithm dialog.
- *Show algorithms with known issues*: By default, QGIS avoids display of broken algorithms (generally from third-party providers). If checked, they will be available in the Processing toolbox, with a warning icon and a tooltip explaining they have issues. Use at your own risks.
- *Show feature count for output vector layers*. Since calculating the feature count might take some time in certain data formats, this option is off by default.
- *Show layer CRS definition in selection boxes*
- *Show tooltip when there are disabled providers*
- *Style for line layers*, *Style for point layers*, *Style for polygons layers* and *Style for raster layers* are used for setting the default rendering style for output layers (that is, layers generated by processing algorithms). Just create the style you want using QGIS, save it to a file, and then enter the path to that file in the settings so the algorithms can use it. Whenever a layer is loaded by Processing and added to the QGIS canvas, it will be rendered with that style.

  Rendering styles can be configured individually for each algorithm and each one of its outputs. Just right-click on the name of the algorithm in the toolbox and select *Edit rendering styles for outputs*. You will see a dialog like the one shown next.

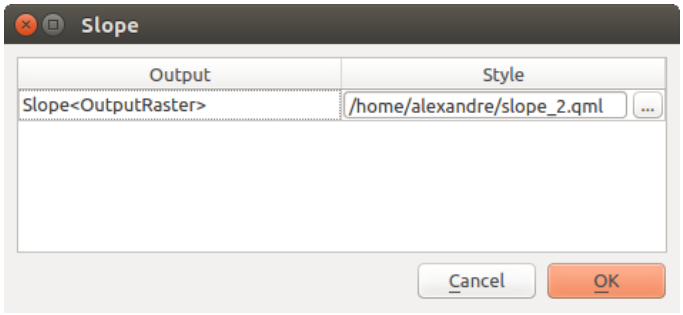


Fig. 27.5: Rendering Styles

  Select the style file (`.qml`) that you want for each output and press *OK*.
- *Warn before executing if parameter CRS's do not match*: By default, QGIS native algorithms (i.e. the ones listed under the *Menus* ► *QGIS (native C++)* group) transparently reproject the input layers to the first one's CRS before execution. Check this option to get a notification from the other tools that do not suppport reprojection, when the inputs CRS are not identical. Third-party providers are not concerned.

27.2.2 Menus

The *Menus* block controls whether an algorithm, script or model (built-in or provided by plugins) should be made available through a dedicated menu or toolbar (along with the Processing Toolbox). For each item of each provider, you can:

- *Add button in toolbar*, making it available in the *Processing Algorithms* toolbar
- assign an *Icon* to the algorithm
- set a *Menu path*: the algorithm will then be available through an existing or a custom menu, e.g. `Vect&or/MyTopAlgorithms`

Restart QGIS to apply the settings. At any time, your changes can be *Reset to defaults*.

27.2.3 Models and Scripts

In the *Models* and *Scripts* blocks, you can set a default folder to store, and look for models and scripts respectively.

27.2.4 Providers

You will also find a block for algorithm *Providers*. This is the place where installed providers expose their settings. For example, built-in providers contain an *Activate* item that you can use to make their algorithms appear or not in the toolbox. Some algorithm providers have their own configuration items, which will be explained when covering particular algorithm providers.

27.3 The Toolbox

The *Processing Toolbox* is the main element of the processing GUI, and the one that you are more likely to use in your daily work. It shows the list of all available **algorithms** grouped in different blocks called *Providers*, and custom **models** and **scripts** you can add to extend the set of tools. Hence the toolbox is the access point to run them, whether as a single process or as a batch process involving several executions of the same algorithm on different sets of inputs.

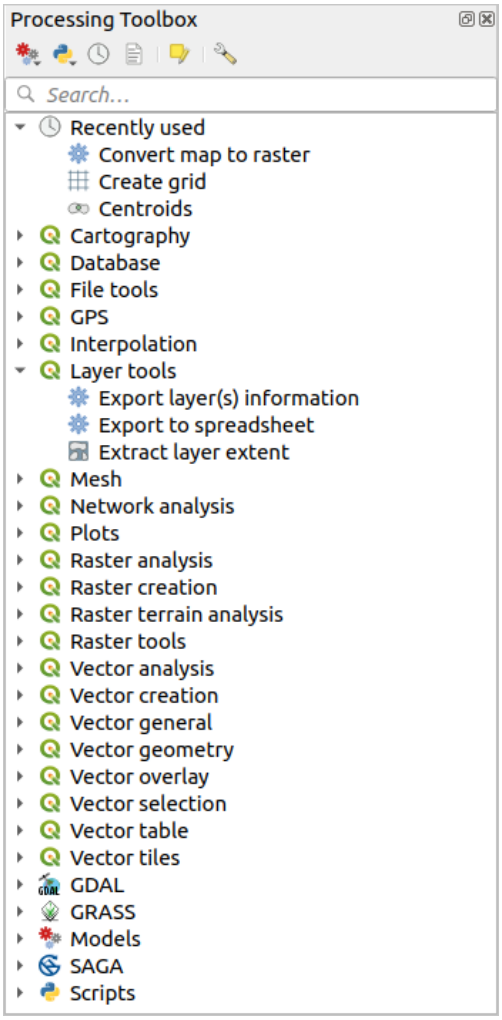


Fig. 27.6: Bara instrumentelor de procesare

Providers can be (de)activated in the *Processing settings dialog*. By default, only providers that do not rely on third-party applications (that is, those that only require QGIS elements to be run) are active. Algorithms requiring external applications might need additional configuration. Configuring providers is explained in a *later chapter* in this manual.

În partea de sus a casetei de dialog, veți găsi un set de instrumente pentru:

- work with Models: *Create New Model…*, *Open Existing Model…* and *Add Model to Toolbox…*;
- work with Scripts: *Create New Script…*, *Create New Script from Template…*, *Open Existing Script…* and *Add Script to Toolbox…*;
- open the History panel;
- open the Results Viewer panel;
- toggle the toolbox to the *in-place modification mode* using the Edit Features In-Place button: only the algorithms that are suitable to be executed on the active layer without outputting a new layer are displayed;
- open the Options dialog.

Below this toolbar is a *Search…* box to help you easily find the tools you need. You can enter any word or phrase on the text box. Notice that, as you type, the number of algorithms, models or scripts in the toolbox is reduced to just those that contain the text you have entered in their names or keywords.

Notă: At the top of the list of algorithms are displayed the most recent used tools; handy if you want to reexecute any.

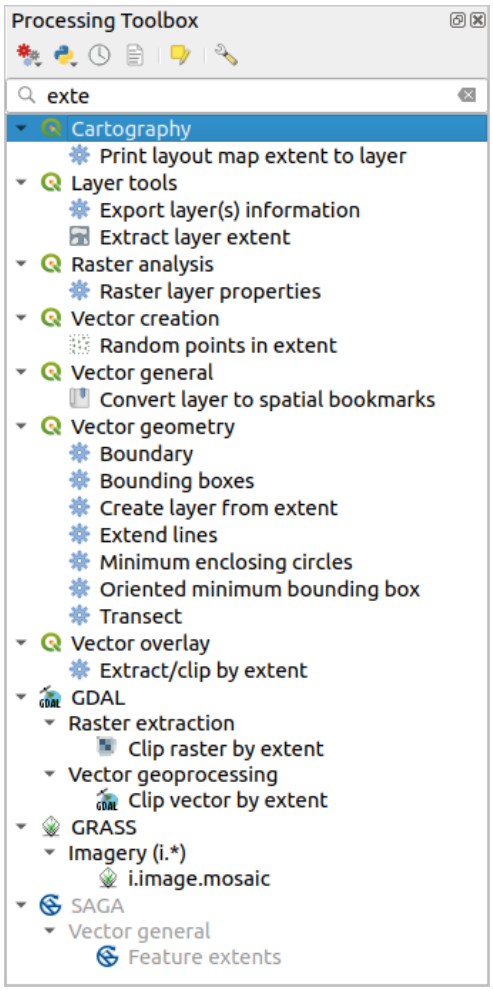


Fig. 27.7: Processing Toolbox showing search results

To execute a tool, just double-click on its name in the toolbox.

27.3.1 Dialogul algoritmului

Once you double-click on the name of the algorithm that you want to execute, a dialog similar to that in the Fig. 27.8 below is shown (in this case, the dialog corresponds to the `Centroids` algorithm).

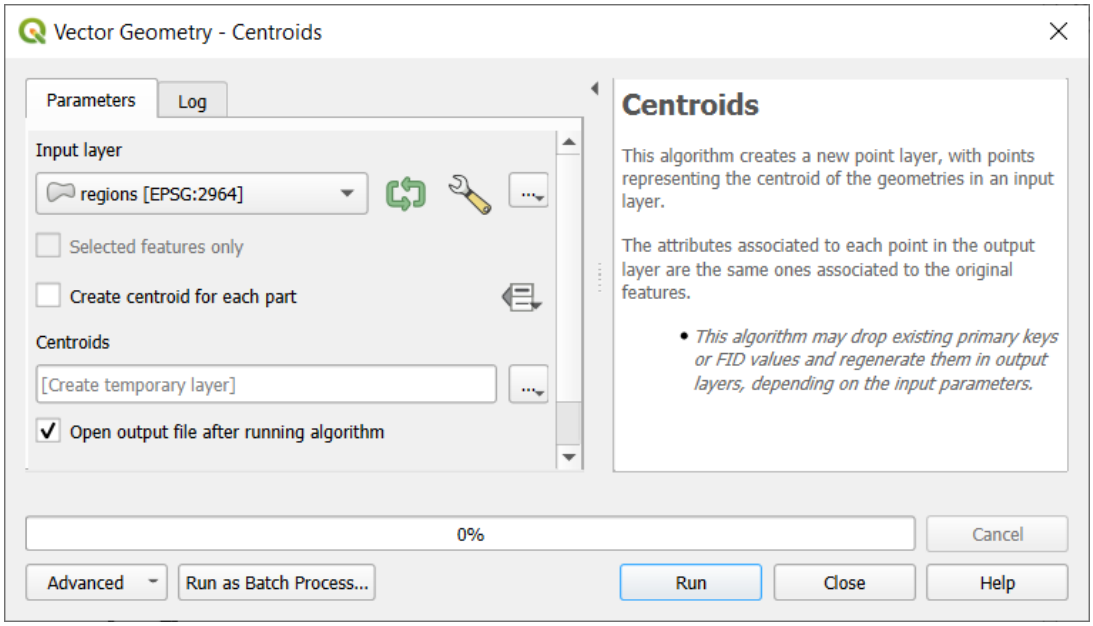


Fig. 27.8: Algorithm Dialog - Parameters

The dialog shows two tabs (*Parameters* and *Log*) on the left part, the algorithm description on the right, and a set of buttons at the bottom.

Parameter types

The *Parameters* tab is used to set the input values that the algorithm needs to be executed. It shows a list of input values and configuration parameters to be set. It of course has a different content, depending on the requirements of the algorithm to be executed, and is created automatically based on those requirements.

Sfat: Setting your own default values for algorithm parameters

Algorithm dialogs open with some parameters prefilled with values from QGIS installation. It is however possible to set *your own default values* for specific algorithm parameters so that they are used at algorithm startup.

Deși numărul și tipul de parametri depind de caracteristicile algoritmului, structura este similară pentru toate. Parametrii din tabel pot avea unul din tipurile de mai jos.

- A **vector layer**, to select from a list of all vector layers available (currently opened) in QGIS. You can also use unloaded layers: press the … button on the widget right-hand side, and select:
  - *Select file…*: selects file on disk using the Operating System file explorer
  - *Browse for layer…*: opens the *Browser panel*, allowing to take the layers directly from database sources (PostgreSQL, SQL Server, Oracle, …), web services (WFS, AFS, …) or files on disk.

Fig. 27.9: Vector input widget

Notă: By default, the layer widget shows the CRS of the layer along with its name. If you do not want to see this additional information, you can disable this functionality in the Processing Settings dialog, unchecking the

General ► *Show layer CRS definition in selection boxes* option.

The vector input widget also has following features:

- an iterator button: If toggled, the algorithm will be executed iteratively on each one of its features, instead of just once for the whole layer, producing as many outputs as times the algorithm is executed. This allows for automating the process when all features in a layer have to be processed separately. If the algorithm contains several input vectors you can iterate over, the iteration will be processed only on the first toggled parameter, in the order parameters are declared in the algorithm.
- Advanced options button to adjust settings to use for that specific parameter. These settings concern:
  - *Invalid feature filtering*: allows the *default method* for handling features with invalid geometries to be overridden
  - *Limit features processed*: optional limit on number of features processed from the source
  - *Feature filter*: allows to enter an expression to subset the layer dynamically when running the tool, avoiding the need for separate steps to set layer filters or create layer subsets.

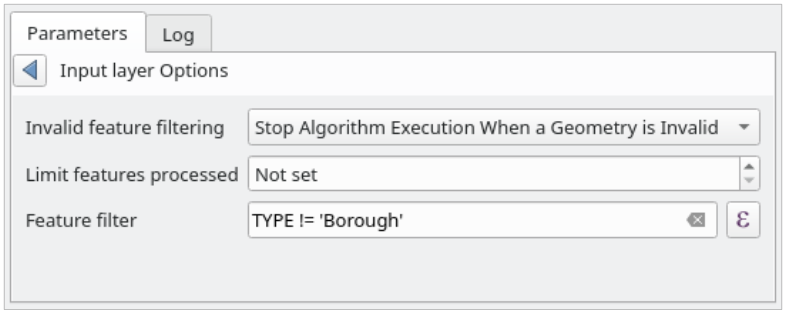


Fig. 27.10: Advanced options for vector input widget

- It is also possible to limit the algorithm execution on the vector layer to its *Selected features only*.

- A **table**, to select from a list of all available in QGIS. Non-spatial tables are loaded into QGIS like vector layers, and use the *same widget*.
- A **raster layer**, to select from a list of all raster layers available in QGIS. The selector contains as well a … button on its right-hand side, to let you select filenames that represent layers currently not loaded in QGIS.

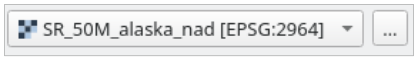


Fig. 27.11: Raster input widget

- An **option**, to choose from a selection list of possible options.
- A **numerical value**, to be introduced in a spin box. In some contexts (when the parameter applies at the feature level and not at the layer's), you will find a Data-defined override button by its side, allowing you to open the *expression builder* and enter a mathematical expression to generate variable values for the parameter. Some useful variables related to data loaded into QGIS can be added to your expression, so you can select a value derived from any of these variables, such as the cell size of a layer or the northernmost coordinate of another one.

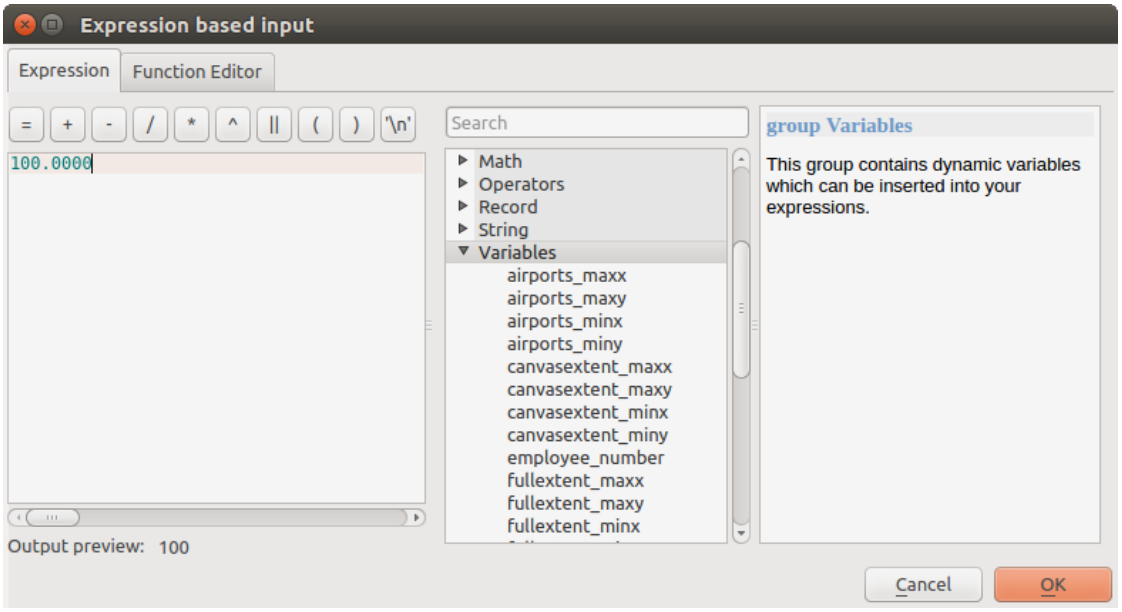


Fig. 27.12: Intrare bazată pe o expresie

- A **range**, with min and max values to be introduced in two text boxes.
- A **text string**, to be introduced in a text box.
- A **field**, to choose from the attributes table of a vector layer or a single table selected in another parameter.
- A **coordinate reference system**. You can select it among the recently used ones from the drop-down list or from the *CRS selection* dialog that appears when you click on the button on the right-hand side.
- An **extent**, a text box defining a rectangle through its corners coordinate in the format `xmin, xmax, ymin, ymax`. Press the Set to current map canvas extent button to use the map canvas extent. Clicking the arrow on the right-hand side of the value selector, a pop-up menu will appear, giving you options to:
  - *Calculate from layer* ►: fills the text box with the coordinates of the bounding box of a layer to select among the loaded ones
  - *Calculate from layout map* ►: fills the text box with the coordinates of a map item selected from a layout in the current project
  - *Calculate from bookmark* ►: fills the text box with the coordinates of a saved bookmark
  - *Use current map canvas extent*
  - *Draw on canvas*: the parameters window will hide itself, so you can click and drag onto the canvas. Once you have defined the extent rectangle, the dialog will reappear, containing the values in the extent text box.

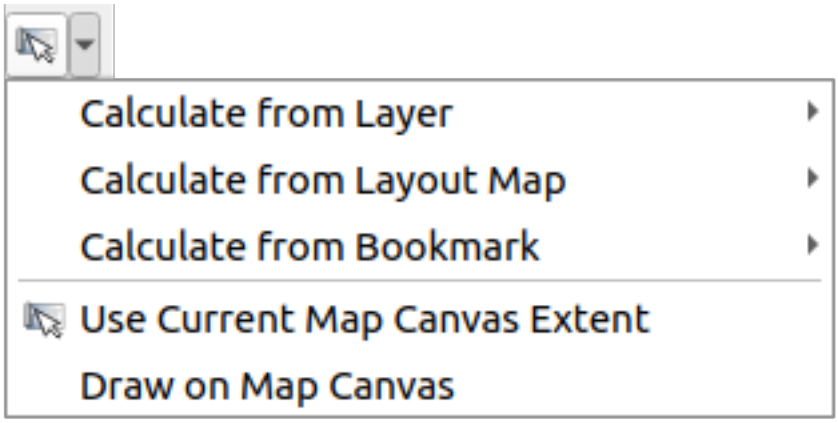


Fig. 27.13: Selectorul Extinderii

- A **list of elements** (whether raster or vector layers, tables, fields) to select from. Click on the … button at the left of the option to see a dialog like the following one. Multiple selection is allowed and when the dialog is closed, number of selected items is displayed in the parameter text box widget.

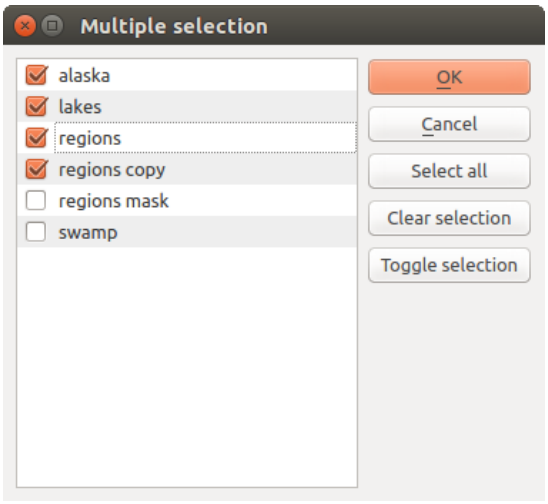


Fig. 27.14: Selecție Multiplă

- A **small table** to be edited by the user. These are used to define parameters like lookup tables or convolution kernels, among others.

  Faceți clic pe butonul din partea dreapta pentru a vedea tabelul și pentru a-i edita valorile.

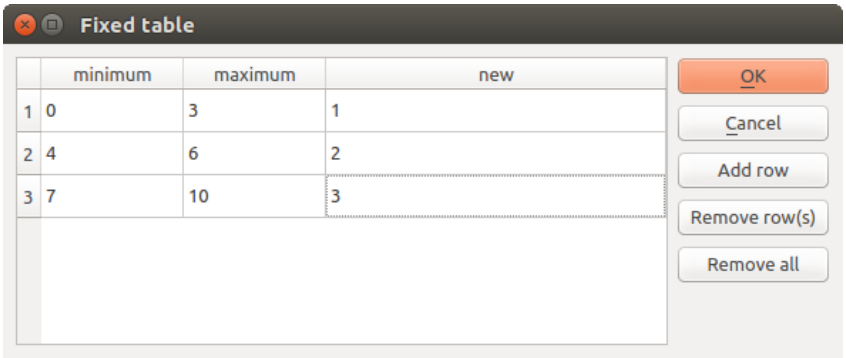


Fig. 27.15: Tabelă Reparată

În funcție de algoritm, numărul de rânduri poate fi modificat sau nu, cu ajutorul butoanelor din pe partea dreaptă a ferestrei.

Notă: Some algorithms require many parameters to run, e.g. in the *Calculator raster* you have to specify manually the cell size, the extent and the CRS. You can avoid to choose all the parameters manually when the algorithm has the `Reference layers` parameter. With this parameter you can choose the reference layer and all its properties (cell size, extent, CRS) will be used.

Logging the execution

Along with the *Parameters* tab, there is another tab named *Log* (see Fig. 27.16 below). Information provided by the algorithm during its execution is written in this tab, allowing you to track the execution as well as being aware and having more details about the algorithm as it runs. Information on algorithm execution is also output in the *View* ► *Panels* ► *Log Messages Panel*.

Notice that not all algorithms write information to the *Log* tab, and many of them might run silently without producing any output other than the final files. Check the *Log Messages Panel* in that case.

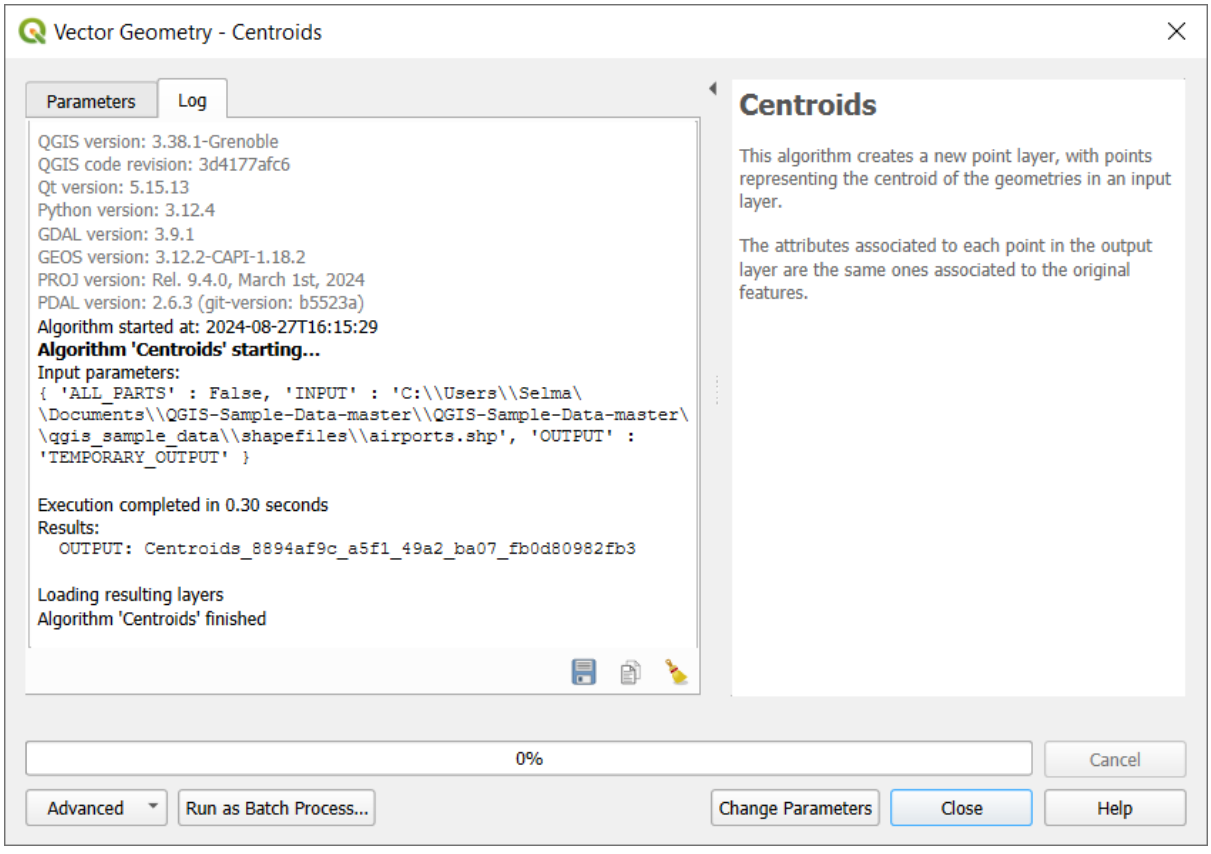


Fig. 27.16: Algorithm Dialog - Log

At the bottom of the *Log* tab you will find buttons to *Save Log to File*, *Copy Log to Clipboard* and *Clear Log*. These are particularly handy when you have checked the *Keep dialog open after running algorithm* in the *General* part of the Processing options.

Other tools

On the right hand side of the dialog you will find a short description of the algorithm, which will help you understand its purpose and its basic ideas. If such a description is not available, the description panel will not be shown.

For a more detailed help file, which might include description of every parameter it uses, or examples, you will find a *Help* button at the bottom of the dialog bringing you to the *Processing algorithms documentation* or to the provider documentation (for some third-party providers).

The *Advanced* ► menu provides functions to reuse the configuration defined in the dialog without running the algorithm:

- *Algorithm Settings…*: allows to override processing settings for the current algorithm execution. More details at *Override algorithm settings*.
- *Copy as Python Command*: allows for easy copying of the equivalent *PyQGIS command* to run the tool using the parameters defined in the dialog
- *Copy as qgis_process Command*: allows for easy generation of *qgis_process command*, including its environment settings like the distance units, area units, ellipsoid, and any tricky parameter values like GeoPackage outputs with specific layers
- *Copy as JSON*: all the settings of the command are copied in a `JSON` format, ready to be consumed by qgis_process. This is a convenient way to see the expected format of the commands, even for complex parameters (like TIN interpolation parameters). You can store these easily and then restore them later by pasting the values.
- *Paste Settings* in a `JSON` format

The *Run as Batch Process…* button triggers the *batch processing mode* allowing to configure and run multiple instances of the algorithm with a variety of parameters. A *Run as Single Process…* helps you switch back from the batch mode.

When an algorithm execution finishes (either successfully or not), a new button *Change Parameters* is shown as long as the *Log* tab is active.

Override algorithm settings

Triggered from within the *Advanced* drop-down menu at the bottom of an algorithm dialog, the *Algorithm Settings…* shows a panel allowing users to control general processing settings which apply to that algorithm execution only. It is intended to be a place where a user can override their *global processing settings* on an ad-hoc basis without having to change their usual default settings.

Settings that can be overridden are:

- *Invalid feature filtering*: unlike the existing per-parameter setting override for this, setting the handling method here will apply to **ALL inputs** for the algorithm
- *Calculation settings*, such as *Distance units* and *Area units* to use for distance/area measurements
- *Environment settings*, such as *Temporary folder* and *Number of threads to use*

O notă privind proiecțiile

Processing algorithm execution are always performed in the input layer coordinate reference system (CRS). Due to QGIS's on-the-fly reprojecting capabilities, although two layers might seem to overlap and match, that might not be true if their original coordinates are used without reprojecting them onto a common coordinate system. Whenever you use more than one layer as input to a *QGIS native algorithm*, whether vector or raster, the layers will all be reprojected to match the coordinate reference system of the first input layer.

This is however less true for most of the external applications whose algorithms are exposed through the processing framework as they assume that all of the layers are already in a common coordinate system and ready to be analyzed.

By default, the parameters dialog will show a description of the CRS of each layer along with its name, making it easy to select layers that share the same CRS to be used as input layers. If you do not want to see this additional information, you can disable this functionality in the Processing settings dialog, unchecking the *Show layer CRS definition in selection boxes* option.

If you try to execute an algorithm using as input two or more layers with unmatching CRSs, a warning dialog will be shown. This occurs thanks to the *Warn before executing if layer CRS's do not match* option.

Puteți încă să executați algoritmul, dar fiți conștienți de faptul că, în cele mai multe cazuri se vor produce rezultate greșite, cum ar fi straturile goale datorate straturilor de intrare care nu se suprapun.

Sfat: Use Processing algorithms to do intermediate reprojection

When an algorithm can not successfully perform on multiple input layers due to unmatching CRSs, use QGIS internal algorithm such as *Reproject layer* to perform layers» reprojection to the same CRS before executing the algorithm using these outputs.

27.3.2 Obiecte de date generate de algoritmi

Obiectele de date generate de un algoritm pot fi oricare din următoarele tipuri:

- Un strat raster
- Un strat vectorial
- O tabelă
- Un fișier HTML (folosit pentru ieșiri de text și grafice)

These are all saved to disk, and the parameters table will contain a text box corresponding to each one of these outputs, where you can type the output channel to use for saving it. An output channel contains the information needed to save the resulting object somewhere. In the most usual case, you will save it to a file, but in the case of vector layers, and when they are generated by native algorithms (algorithms not using external applications) you can also save to a PostGIS, GeoPackage or SpatiaLite database, or a memory layer.

To select an output channel, just click on the button on the right side of the text box, and you will see a small context menu with the available options.

In the most usual case, you will select saving to a file. If you select that option, you will be prompted with a save file dialog, where you can select the desired file path. Supported file extensions are shown in the file format selector of the dialog, depending on the kind of output and the algorithm.

The format of the output is defined by the filename extension. The supported formats depend on what is supported by the algorithm itself. To select a format, just select the corresponding file extension (or add it, if you are directly typing the file path instead). If the extension of the file path you entered does not match any of the supported formats, a default extension will be appended to the file path, and the file format corresponding to that extension will be used to save the layer or table. Default extensions are `.dbf` for tables, `.tif` for raster layers and `.gpkg` for vector layers. These can be modified in the setting dialog, selecting any other of the formats supported by QGIS.

If you do not enter any filename in the output text box (or select the corresponding option in the context menu), the result will be saved as a *temporary file* in the corresponding default file format, and it will be deleted once you exit QGIS (take care with that, in case you save your project and it contains temporary layers).

You can set a default folder for output data objects. Go to the settings dialog (you can open it from the *Settings* ► *Options* ► *Processing* menu), and in the *General* group, you will find a parameter named *Output folder*. This output folder is used as the default path in case you type just a filename with no path (i.e., `myfile.shp`) when executing an algorithm.

When running an algorithm that uses a vector layer in iterative mode, the entered file path is used as the base path for all generated files, which are named using the base name and appending a number representing the index of the iteration. The file extension (and format) is used for all such generated files.

Apart from raster layers and tables, algorithms also generate graphics and text as HTML files. These results are shown at the end of the algorithm execution in a new dialog. This dialog will keep the results produced by any algorithm during the current session, and can be shown at any time by selecting *Processing* ► *Results Viewer* from the QGIS main menu.

Some external applications might have files (with no particular extension restrictions) as output, but they do not belong to any of the categories above. Those output files will not be processed by QGIS (opened or included into the current QGIS project), since most of the time they correspond to file formats or elements not supported by QGIS. This is, for instance, the case with LAS files used for LiDAR data. The files get created, but you won't see anything new in your QGIS working session.

Pentru toate celelalte tipuri de rezultate, veți găsi o casetă de selectare, pe care o puteți folosi pentru a indica algoritmului dacă să încarce fișierul după ce este generat de către algoritm, sau nu. În mod implicit, toate fișierele vor fi deschise.

Optional outputs are not supported. That is, all outputs are created. However, you can uncheck the corresponding checkbox if you are not interested in a given output, which essentially makes it behave like an optional output (in other words, the layer is created anyway, but if you leave the text box empty, it will be saved to a temporary file and deleted once you exit QGIS).

27.4 Managerul istoricului

27.4.1 Procesarea istoricului

Every time you execute an algorithm, information about the process is stored in the history manager. The date and time of the execution are saved, along with the parameters used, making it is easy to track and control all the work that has been developed using the Processing framework, and to reproduce it.

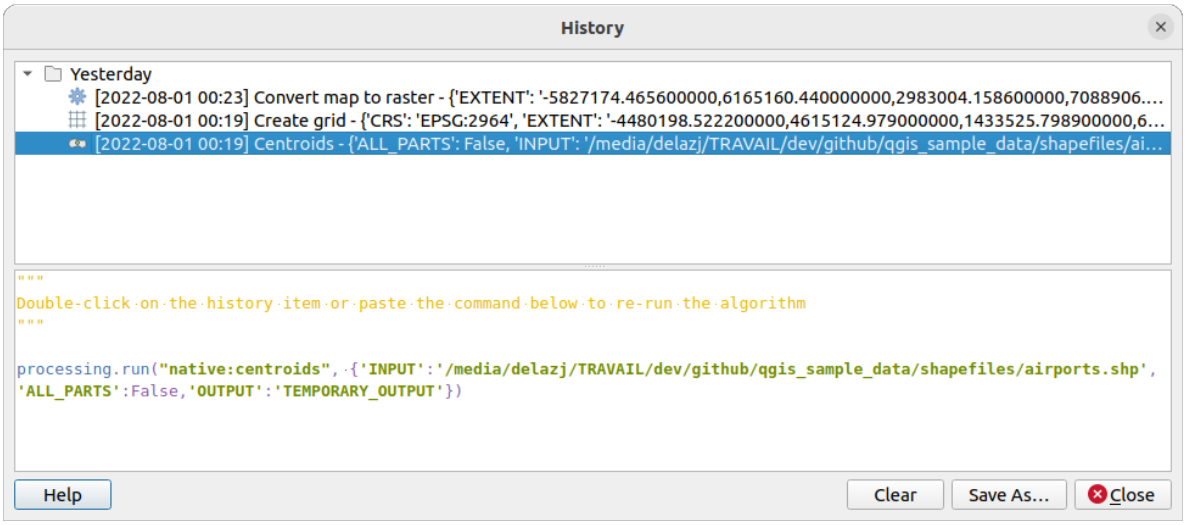


Fig. 27.17: Istoric

Process information is kept as a command-line expression, even if the algorithm was launched from the toolbox. This makes it useful for those learning how to use the command-line interface, since they can call an algorithm using the toolbox and then check the history manager to see how it could be called from the command line.

Right-click on a row and you should be able to:

- *Copy as Python Command*: allows for easy copying of the equivalent *PyQGIS command* run from the dialog. Same as the code displayed below the commands list.
- *Copy as qgis_process Command*: allows for easy generation of *qgis_process command*, including its environment settings like the distance units, area units, ellipsoid, and any tricky parameter values like GeoPackage outputs with specific layers
- *Copy as JSON*: all the settings of the command are copied in a `JSON` format, ready to be consumed by qgis_process. This is a convenient way to see the expected format of the commands, even for complex parameters (like TIN interpolation parameters). You can store these easily and then restore them later by pasting the values over an algorithm dialog.
- *Create Test...* using the concerned algorithm and parameters, following instructions at Processing README file.

Apart from browsing the entries in the registry, you can also re-execute processes by simply double-clicking on the entry. The algorithm dialog then opens with parameters already set, and you can change any of them to fit your needs and re-run the algorithm.

The *History* dialog also provides a convenient way to contribute to the consolidation of the testing infrastructure of QGIS Processing algorithms and scripts.

27.4.2 Jurnalul de Procesare

The history dialog only contains the execution calls, but not the information produced by the algorithm when executed. That information is written to the QGIS log (*View* ► *Panels* ► *Log Messages Panel*).

Third-party algorithms are usually executed by using their command-line interfaces, which communicate with the user via the console. Although that console is not shown, usually a full dump of it is written to the log each time you run one of those algorithms. To avoid cluttering the log with that information, you can disable it for each provider in the settings dialog.

Some algorithms, even if they can produce a result with the given input data, output comments or additional information to log when they detect potential problems with the data, in order to warn you. Make sure you check those messages in the log if you get unexpected results.

27.5 The model designer

The *model designer* allows you to create complex models using a simple and easy-to-use interface. When working with a GIS, most analysis operations are not isolated, rather part of a chain of operations. Using the model designer, that chain of operations can be wrapped into a single process, making it convenient to execute later with a different set of inputs. No matter how many steps and different algorithms it involves, a model is executed as a single algorithm, saving time and effort.

The model designer can be opened from the Processing menu (*Processing* ► *Model Designer*).

27.5.1 The model designer interface

Fig. 27.18: Model designer

In its main part, the modeler has a working canvas where the structure of the model and the workflow it represents can be constructed.

At the top of the dialog, different menus and the *Navigation* toolbar give access to a variety of tools.

Model menu

Etichetă	Scurtătură	Navigation Toolbar	Descriere
Validate Model			Checks whether the algorithms and inputs used in the model exist. Convenient before releasing a model.
Run Model…	`F5`	☑	Executes the model
Reorder Model Inputs…			Sets the order in which inputs are presented to the user in the algorithm dialog.
Reorder Output Layers…			Sets a specific order which the outputs from the model must use when loading the results into a project.
Open Model…	`Ctrl+O`	☑	Opens a `.model3` file for edit or execution
Save Model	`Ctrl+S`	☑	Saves the model to disk as a `.model3` file
Save Model as…	`Ctrl+Shi`	☑	Saves the model to disk as a new `.model3` file
Save Model in project		☑	Embeds the model file in the project file, making it available when sharing the project file.
Edit Model Help…		☑	An interface to document the model, the algorithms, the parameters and outputs, as well as the author and versioning
Export ►			
► *Export as Image…*		☑	Saves the model's graphical design to an image file format (for illustration purpose)
► *Export as PDF…*			Saves the model's graphical design to a `PDF` file format (for illustration purpose)
► *Export as SVG…*			Saves the model's graphical design to an `SVG` file format (for illustration purpose)
► *Export as Script Algorithm…*		☑	Generates a python script file including the model's instructions

Edit menu

Etichetă	Scurtătură	Navigation Toolbar	Descriere
Select All	`Ctrl+A`		Selects all the model components in the designer
Snap selected components to Grid			snaps and aligns the elements into a grid
Refacere	`Ctrl+Y`	☑	Rollback the latest canceled action. See also the *Undo/Redo* panel.
Anulare	`Ctrl+Z`	☑	Cancel the previous change. See also the *Undo/Redo* panel.
Cut	`Ctrl+X`		Cuts a selection of components from the model.
Copy	`Ctrl+C`		Copies a selection of components from the model.
Lipire	`Ctrl+V`		Pastes a cut or copied selection of components from a model to another or within the same model. The selected components keep their original properties and comments.
Delete selected components	`Del`		Removes a component from the model.
Add Group Box			Adds a box at the background of related components in order to visually group them. Particularly useful in big models to keep the workflow clean.

View menu

Etichetă	Scurtătură	Navigation Toolbar	Descriere
Zoom To ►			Zooms to the selected group box extent
Mărire	`Ctrl++`	☑	
Micșorare	`Ctrl+-`	☑	
Zoom la 100%	`Ctrl+1`	☑	
Transfocare la nivelul Hărții	`Ctrl+0`	☑	Displays all the components in the designer current canvas
Show Comments			Displays comments associated to every algorithm or input in the model designer
Enable Snapping			
Toggle Panel Visibility	`Ctrl+Tab`		Switches ON or OFF the *panels* in the designer

Panouri

The left part of the window is a section with five panels that can be used to add new elements to the model:

1. *Model Properties*: specify the name (required) of the model and the group in which it will be displayed in the *Processing Toolbox*
2. *Inputs*: all the *input parameters* that could shape your model
3. *Algorithms*: the available *Processing algorithms*
4. *Variables*: Models can contain dedicated *variables* that are unique and only available to them. These variables can be accessed by any expression used within the model. They are useful to control algorithms within a model and control multiple aspects of the model by changing a single variable. The variables can be viewed and modified in the *Variables* panel.
5. *Undo History*: this panel will register everything that happens in the modeler, making it easy to cancel things you did wrong.

Despre algoritmii disponibili

Some algorithms that can be executed from the toolbox do not appear in the list of available algorithms when you are designing a model. To be included in a model, an algorithm must have the correct semantic. If an algorithm does not have such a well-defined semantic (for instance, if the number of output layers cannot be known in advance), then it is not possible to use it within a model, and it will not appear in the list of algorithms that you can find in the modeler dialog. On the other hand some algorithms are specific to the modeler. Those algorithms are located within the group «Modeler Tools».

27.5.2 Creating a model

Creating a model involves two basic steps:

1. *Definition of necessary inputs*. These inputs will be added to the parameters window, so the user can set their values when executing the model. The model itself is an algorithm, so the parameters window is generated automatically as for all algorithms available in the Processing framework.
2. *Definition of the workflow*. Using the input data of the model, the workflow is defined by adding algorithms and selecting how they use the defined inputs or the outputs generated by other algorithms in the model.

Definiția intrărilor

The first step is to define the inputs for the model. The following elements are found in the *Inputs* panel on the left side of the modeler window:

Tabelul 27.1: List of parameter types for model building

Annotation Layer	Authentication Configuration	Boolean	Color
Coordinate Operation	CRS	Database Schema	Database
Distance	Duration	DXF Layers	Enum
Extent	Field Aggregates	Fields Mapper	File/Fol
Map Layer	Map Theme	Matrix	Mesh Dat
Mesh Layer	Multiple Input	Number	Point
Point Cloud Layer	Print Layout	Print Layout Item	Range
Raster Layer	Scale	String	TIN Crea
Vector Field	Vector Layer	Vector Tile Writer Layers	

Notă: Hovering with the mouse over the inputs will show a tooltip with additional information.

When double-clicking on an element, a dialog is shown that lets you define its characteristics. Depending on the parameter, the dialog will contain at least one element (the description, which is what the user will see when executing the model). For example, when adding a numerical value, as can be seen in the next figure, in addition to the description of the parameter, you have to set a default value and the range of valid values.

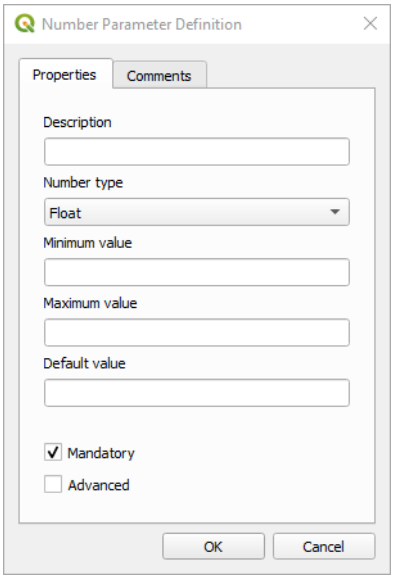


Fig. 27.19: Definirea Parametrilor Modelului

You can define your input as mandatory for your model by checking the `Mandatory` option and by checking the `Advanced` checkbox you can set the input to be within the `Advanced` section. This is particularly useful when the model has many parameters and some of them are not trivial, but you still want to choose them.

For each added input, a new element is added to the modeler canvas.

Fig. 27.20: Model Parameters

You can also add inputs by dragging the input type from the list and dropping it at the position where you want it in the modeler canvas. If you want to change a parameter of an existing input, just double click on it, and the same dialog will pop up.

When using a model within another model, the inputs and outputs necessary will be displayed in the canvas.

Definiția fluxului de lucru

In the following example we will add two inputs and two algorithms. The aim of the model is to copy the elevation values from a DEM raster layer to a line layer using the `Drape` algorithm, and then calculate the total ascent of the line layer using the `Climb Along Line` algorithm.

In the *Inputs* tab, choose the two inputs as `Vector Layer` for the line and `Raster Layer` for the DEM. We are now ready to add the algorithms to the workflow.

Algorithms can be found in the *Algorithms* panel, grouped much in the same way as they are in the Processing toolbox.

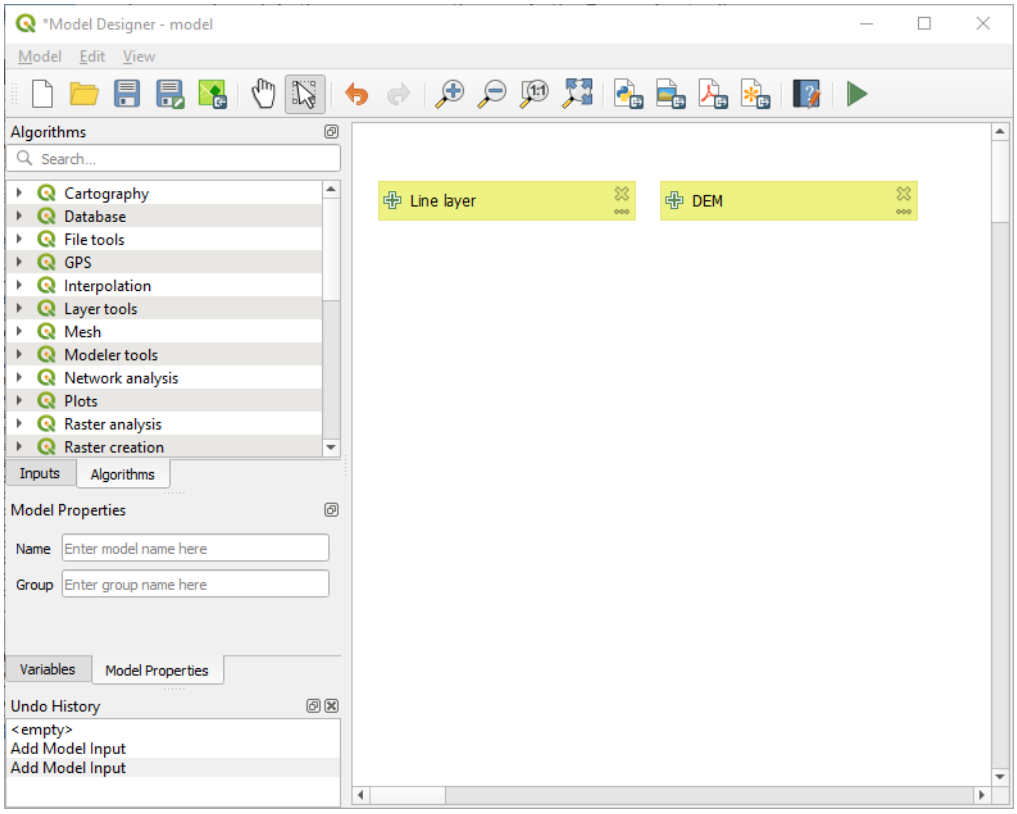


Fig. 27.21: Model Inputs

To add an algorithm to a model, double-click on its name or drag and drop it, just like for inputs. As for the inputs you can change the description of the algorithm and add a comment. When adding an algorithm, an execution dialog will appear, with a content similar to the one found in the execution panel that is shown when executing the algorithm from the toolbox. The following picture shows both the `Drape (set Z value from raster)` and the `Climb along line` algorithm dialogs.

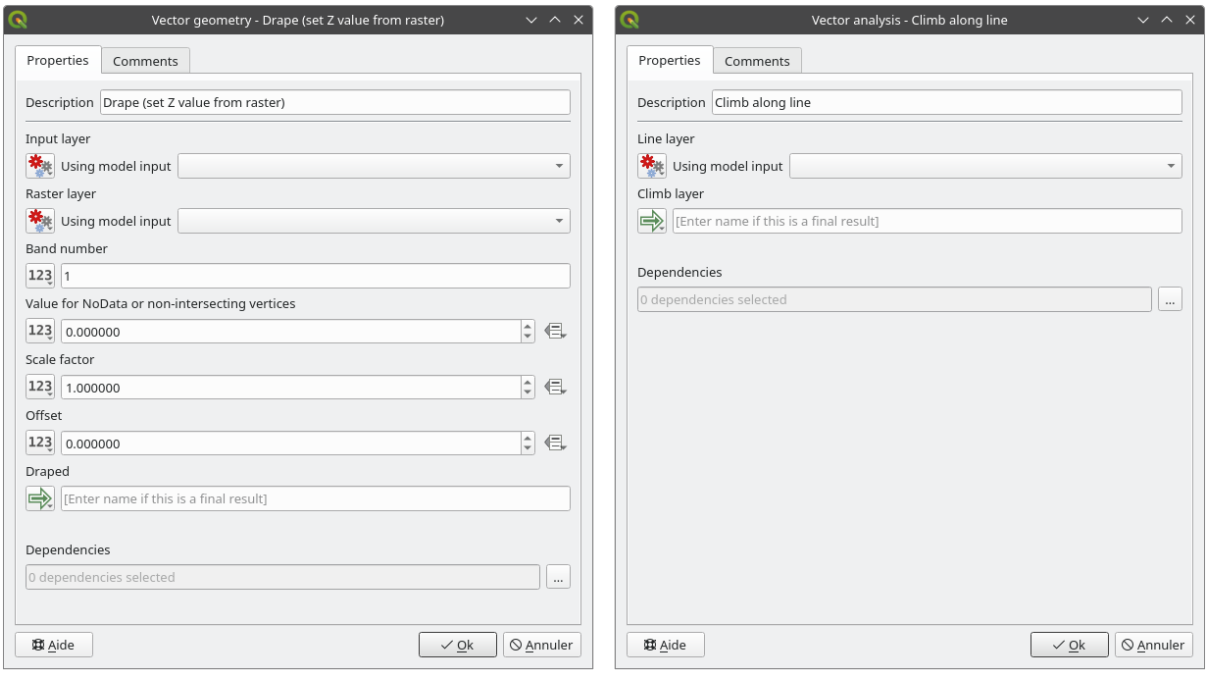


Fig. 27.22: Model Algorithm parameters

As you can see, there are however some differences. Each parameter has a drop-down menu next to it allowing to control how it will be served during the workflow:

- Value: allows you to assign a static value to the parameter. Depending on the parameter type, the widget will let you enter a number (`5.0`), a string (`mytext`), select layer(s) loaded in the QGIS project or from a folder, pick items from a list, …
- Pre-calculated Value: opens the *Expression Builder* dialog and lets you define an expression to fill the parameter. Model inputs together with some other layer statistics are available as **variables** and are listed at the top of the Search dialog of the Expression Builder. The expression is evaluated once before the child algorithm is executed and used during the execution of that algorithm.
- Model Input: allows to use an input added to the model as a parameter. Once clicked, this option will list all the suitable inputs for the parameter.
- Algorithm Output: allows to use the output of another algorithm as an input of the current algorithm. As of model inputs, this option will list all the suitable inputs for the parameter.
- The **output parameter** also has the above options in its drop-down menu:
  - add static outputs for child algorithms, e.g. always saving a child algorithm's output to a predefined geopackage or postgres layer
  - use an expression based output values for child algorithms, e.g. generating an automatic file name based on today's date and saving outputs to that file
  - use a model input, e.g. the *File/Folder* model input to specify an output file or folder
  - use another algorithm output, e.g. the output of the *Create directory* algorithm (from *Modeler tools*)
  - an addditional Model Output option makes the output of the algorithm available in the model. If a layer generated by the algorithm is only to be used as input to another algorithm, don't edit that text box.

  In the following picture you can see the two input parameters defined as `Model Input` and the temporary output layer:

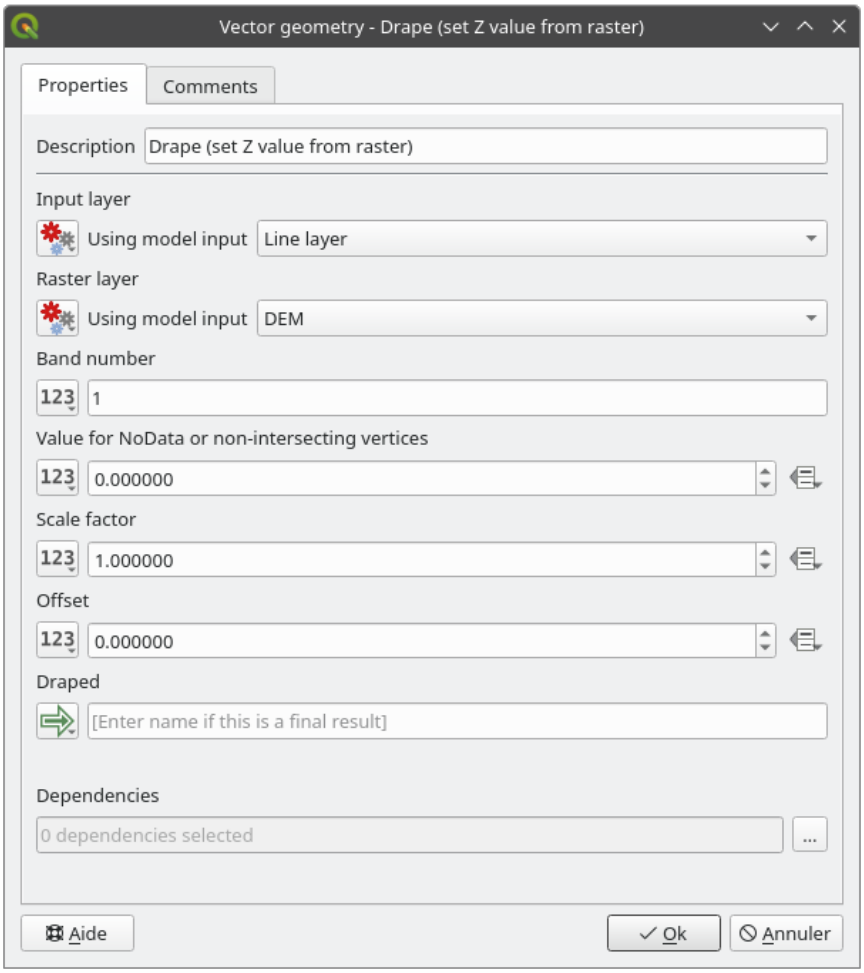


Fig. 27.23: Algorithm Input and Output parameters

You will also find an additional parameter named *Dependencies* that is not available when calling the algorithm from the toolbox. This parameter allows you to define the order in which algorithms are executed, by explicitly defining one algorithm as a *parent* of the current one. This will force the *parent* algorithm to be executed before the current one.

When you use the output of a previous algorithm as the input of your algorithm, that implicitly sets the previous algorithm as parent of the current one (and places the corresponding arrow in the modeler canvas). However, in some cases an algorithm might depend on another one even if it does not use any output object from it (for instance, an algorithm that executes a SQL sentence on a PostGIS database and another one that imports a layer into that same database). In that case, just select the previous algorithm in the *Dependencies* parameter and they will be executed in the correct order.

Once all the parameters have been assigned valid values, click on *OK* and the algorithm will be added to the canvas. It will be linked to the elements in the canvas (algorithms or inputs) that provide objects that are used as inputs for the algorithm.

Elements can be dragged to a different position on the canvas using the Select/Move Item tool. This is useful to make the structure of the model clearer and more intuitive. You can also resize the elements, grasping their border. This is particularly useful if the description of the input or algorithm is long. With *View* ► *Enable snapping* option checked, items resizing or displacement can be bound to a virtual grid, for a more visually structured algorithm design.

Links between elements are updated automatically and you can see a + button at the top and at the bottom of each algorithm. Clicking the button will list all the inputs and outputs of the algorithm so you can have a quick overview.

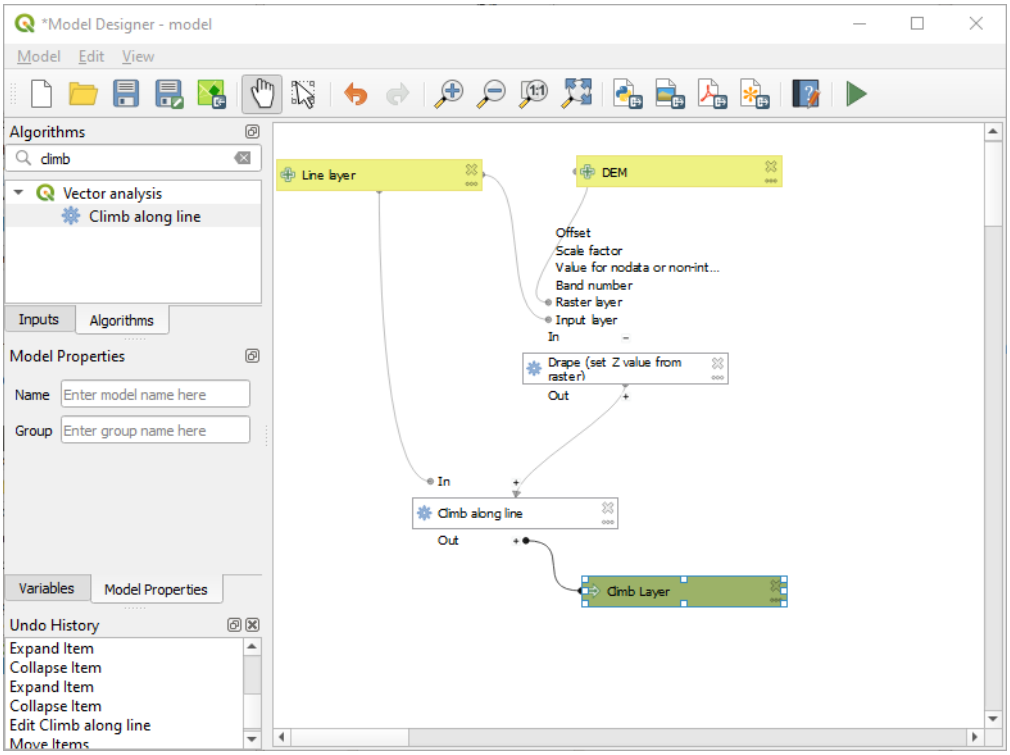


Fig. 27.24: A complete model

With the *Edit* ► *Add Group Box* tool, you can add a draggable *box* to the canvas. This feature is very useful in big models to group related elements in the modeler canvas and to keep the workflow clean. For example we might group together all the inputs of the example:

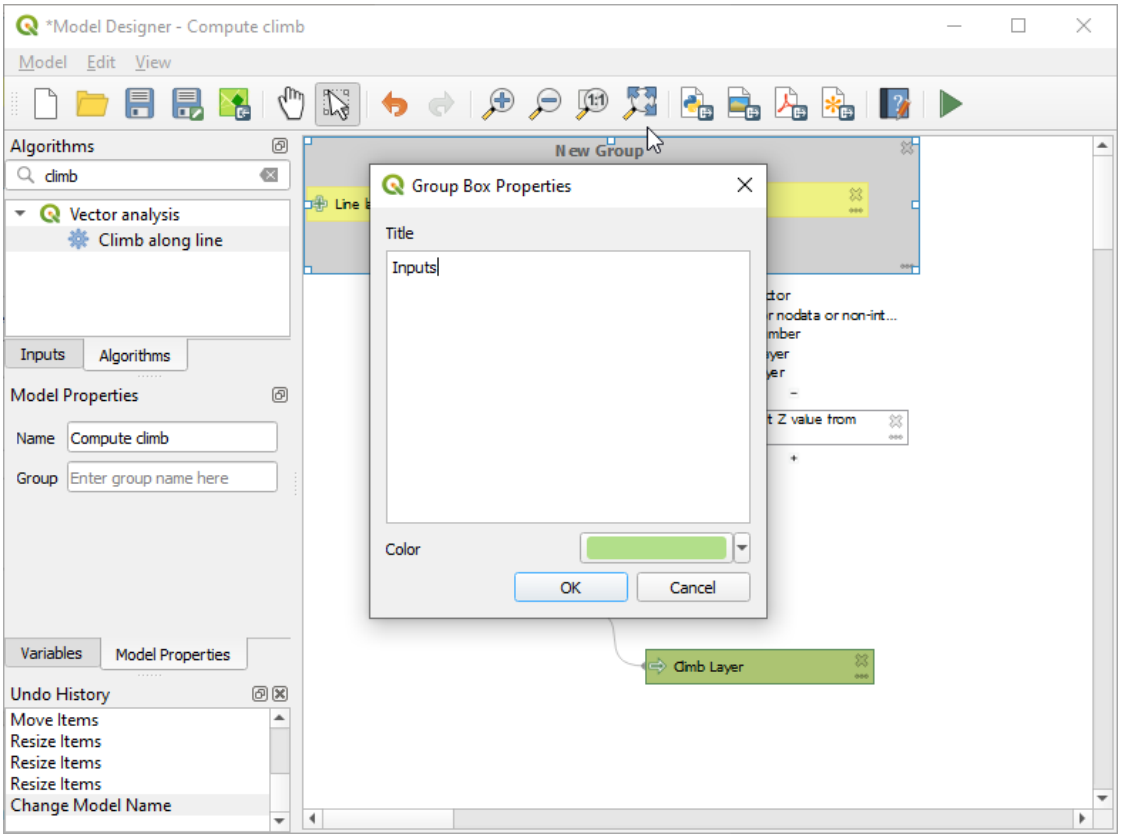


Fig. 27.25: Model Group Box

You can change the name and the color of the boxes. Group boxes are very useful when used together with *View* ► *Zoom To* ► tool, allowing you to zoom to a specific part of the model. You can also zoom in and out by using the mouse wheel.

You might want to change the order of the inputs and how they are listed in the main model dialog. At the bottom of the `Input` panel you will find the `Reorder Model Inputs...` button and by clicking on it a new dialog pops up allowing you to change the order of the inputs:

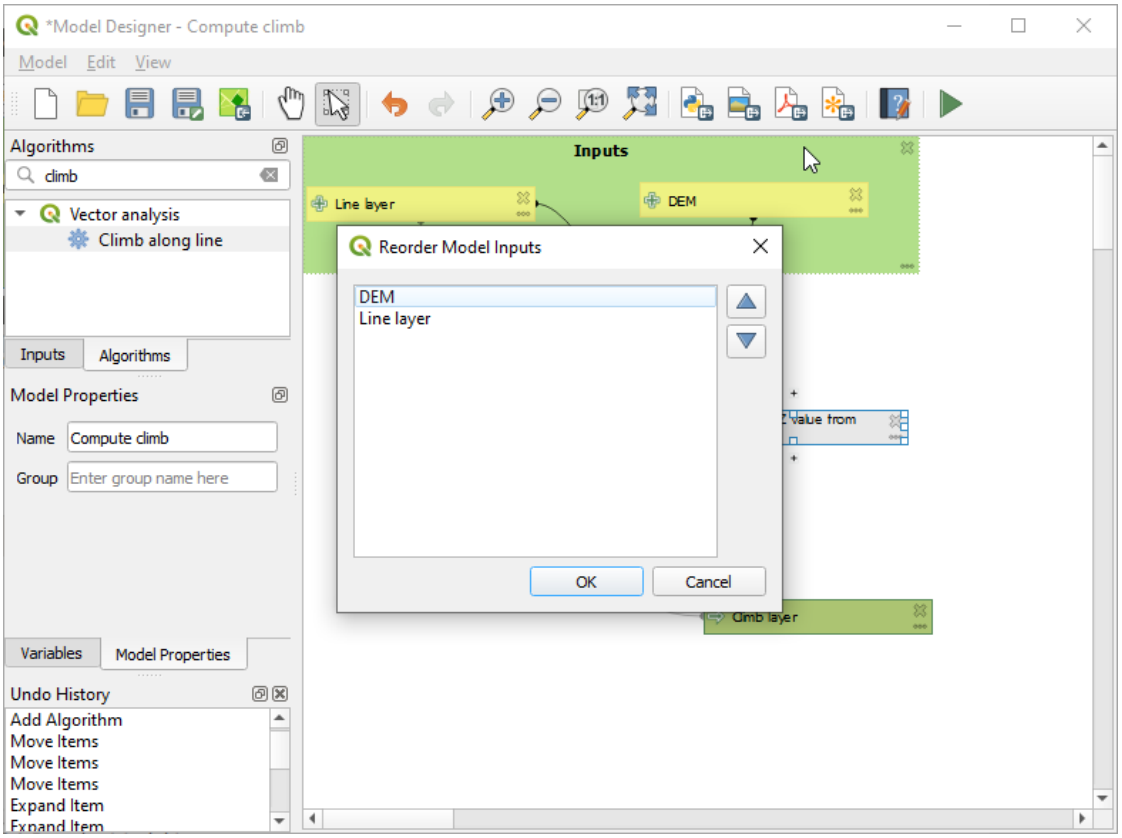


Fig. 27.26: Reorder Model Inputs

There is also the possibility to set a specific order which the outputs from the model must use when loading the results into a project. This gives the model creator a means of ensuring that layers are logically ordered on the canvas when running a model, such as placing a vector layer output over a raster layer output, or a point layer over a polygon layer. The model creator can also set an optional „Group name" for the outputs for automatically grouping outputs within the layer tree using a new group name or by adding them to an existing group. In the `Model` menu you will find the `Reorder Output Layers...` entry and by clicking on it a new dialog pops up allowing you to change the order of the output layers:

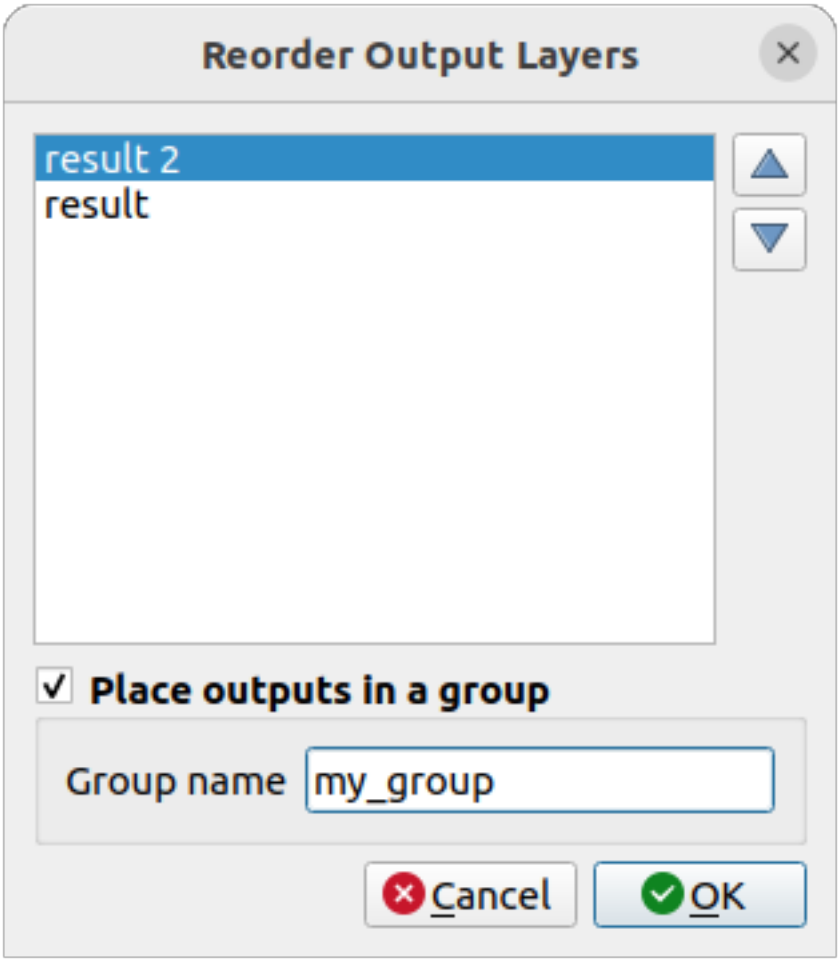


Fig. 27.27: Reorder Output Layers

Comments can also be added to inputs or algorithms present in the modeler. This can be done by going in the *Comment* tab of the item or with a right-click. In the same tab a color can be set manual for individual model comments. Comments are visible only in the modeler canvas and not in the final algorithm dialog; they can be hidden by deactivating *View* ► *Show Comments*.

You can run your algorithm any time by clicking on the Run model button. When using the editor to execute a model, any non-default values will be saved in the inputs. This means that executing the model at a later time from the editor will have the dialog prefilled with those values on any subsequent run.

In order to use the algorithm from the toolbox, it has to be saved and the modeler dialog closed, to allow the toolbox to refresh its contents.

Documenting your model

You need to document your model, and this can be done from the modeler itself. Click on the Edit model help button, and a dialog like the one shown next will appear.

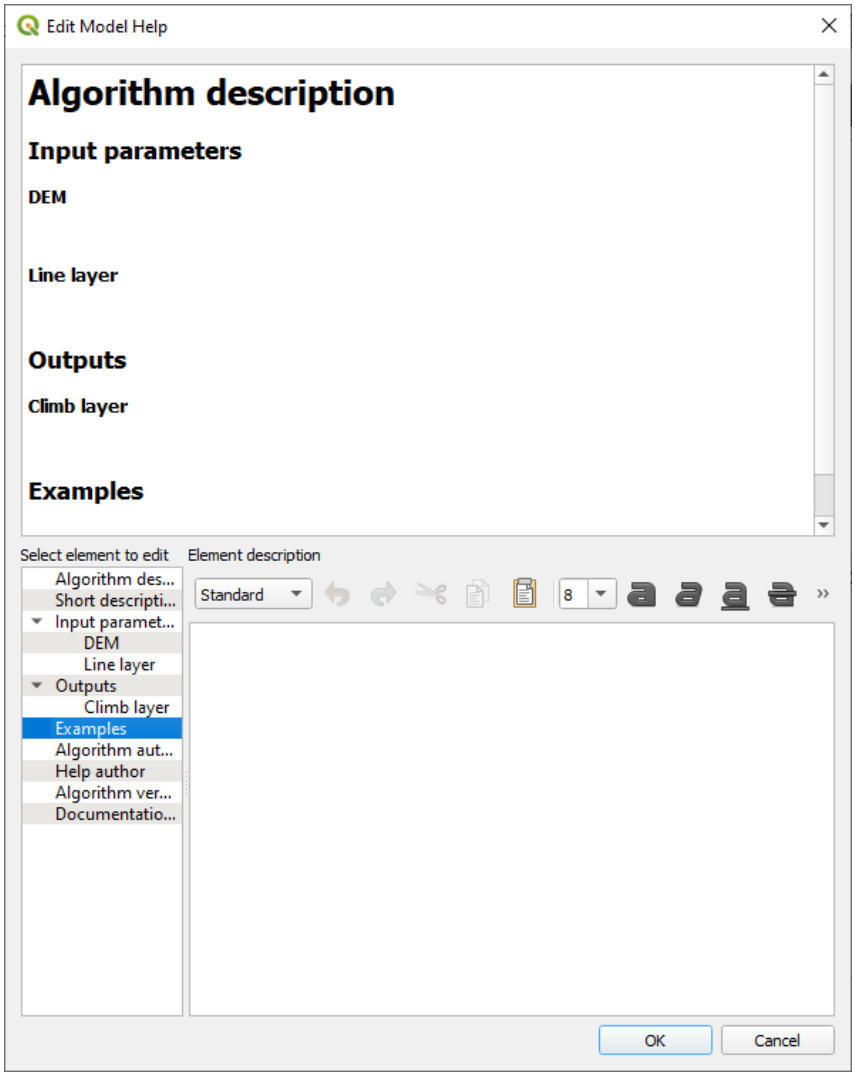


Fig. 27.28: Editing Help

On the right-hand side, you will see a simple HTML page, created using the description of the input parameters and outputs of the algorithm, along with some additional items like a general description of the model or its author. Also, there is an Example section where you can input your own custom examples to help explain the usage of the model. The first time you open the help editor, all these descriptions are empty, but you can edit them using the elements on the left-hand side of the dialog. Select an element on the upper part and then write its description in the text box below.

Modelul ajutorului este salvat ca parte a modelului în sine.

27.5.3 Salvarea și încărcarea modelelor

Saving models

Use the Save model button to save the current model and the Open Model button to open a previously saved model. Models are saved with the `.model3` extension. If the model has already been saved from the modeler window, you will not be prompted for a filename. Since there is already a file associated with the model, that file will be used for subsequent saves.

Before saving a model, you have to enter a name and a group for it in the text boxes in the upper part of the window.

Models saved in the `models` folder (the default folder when you are prompted for a filename to save the model) will appear in the toolbox in the corresponding branch. When the toolbox is invoked, it searches the `models` folder for files with the `.model3` extension and loads the models they contain. Since a model is itself an algorithm, it can be added to the toolbox just like any other algorithm.

Models can also be saved within the project file using the Save model in project button. Models saved using this method won't be written as `.model3` files on the disk but will be embedded in the project file.

Project models are available in the *Project models* menu of the toolbox and in the *Project* ► *Models* menu item.

Dosarul modelelor poate fi setat din dialogul de configurare pentru Processing, conform grupului *Modeler*.

Models loaded from the `models` folder appear not only in the toolbox, but also in the algorithms tree in the *Algorithms* tab of the modeler window. That means that you can incorporate a model as a part of a bigger model, just like other algorithms.

Models will show up in the *Browser* panel and can be run from there.

Exporting a model as a Python script

As we will see in a later chapter, Processing algorithms can be called from the QGIS Python console, and new Processing algorithms can be created using Python. A quick way to create such a Python script is to create a model and then export it as a Python file.

To do so, click on the Export as Script Algorithm… in the modeler canvas or right click on the name of the model in the Processing Toolbox and choose Export Model as Python Algorithm….

Exporting a model as an image, PDF or SVG

A model can also be exported as an image, SVG or PDF (for illustration purposes) by clicking Export as image, Export as PDF or Export as SVG.

27.5.4 Editarea unui model

You can edit the model you are currently creating, redefining the workflow and the relationships between the algorithms and inputs that define the model.

If you right-click on an algorithm in the canvas, you will see a context menu like the one shown next:

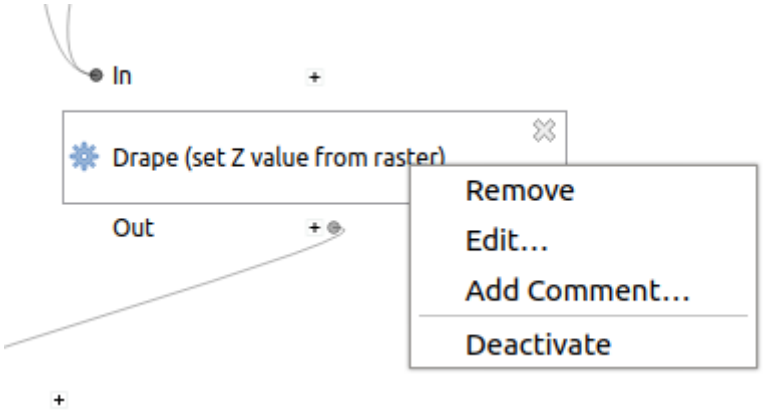


Fig. 27.29: Clic-Dreapta pe Modelator

Selecting the *Remove* option will cause the selected algorithm to be removed. An algorithm can be removed only if there are no other algorithms depending on it. That is, if no output from the algorithm is used in a different one as input. If you try to remove an algorithm that has others depending on it, a warning message like the one you can see below will be shown:

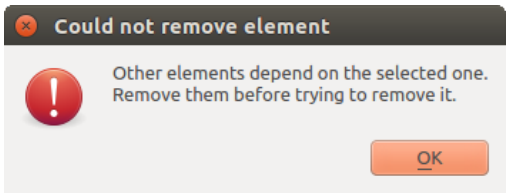


Fig. 27.30: Nu se Poate Șterge Algoritmul

Selecting the *Edit...* option will show the parameter dialog of the algorithm, so you can change the inputs and parameter values. Not all input elements available in the model will appear as available inputs. Layers or values generated at a more advanced step in the workflow defined by the model will not be available if they cause circular dependencies.

Select the new values and click on the *OK* button as usual. The connections between the model elements will change in the modeler canvas accordingly.

The *Add comment...* allows you to add a comment to the algorithm to better describe the behavior.

A model can be run partially by deactivating some of its algorithms. To do it, select the *Deactivate* option in the context menu that appears when right-clicking on an algorithm element. The selected algorithm, and all the ones in the model that depend on it will be displayed in grey and will not be executed as part of the model.

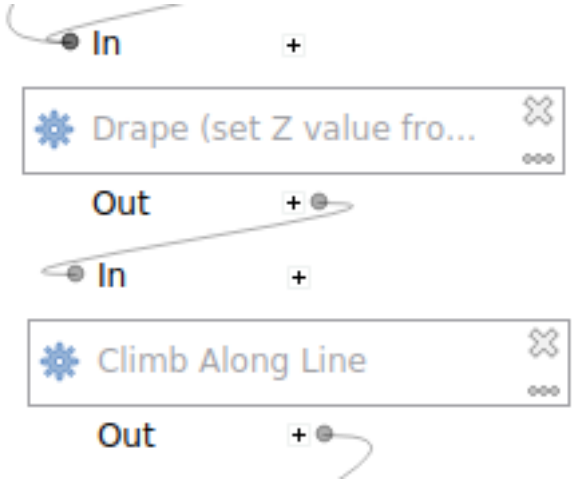


Fig. 27.31: Modelul cu Algoritmi Deactivați

When right-clicking on an algorithm that is not active, you will see a *Activate* menu option that you can use to reactivate it.

27.6 Interfața de prelucrare în serie

27.6.1 Introducere

All algorithms (including models) can be executed as a batch process. That is, they can be executed using not just a single set of inputs, but several of them, executing the algorithm as many times as needed. This is useful when processing large amounts of data, since it is not necessary to launch the algorithm many times from the toolbox.

Pentru a executa un algoritm sub formă de procedeu în serie, faceți clic-dreapta pe numele său din caseta de instrumente, apoi selectați opțiunea *Execute as batch process* din meniul pop-up care va apărea.

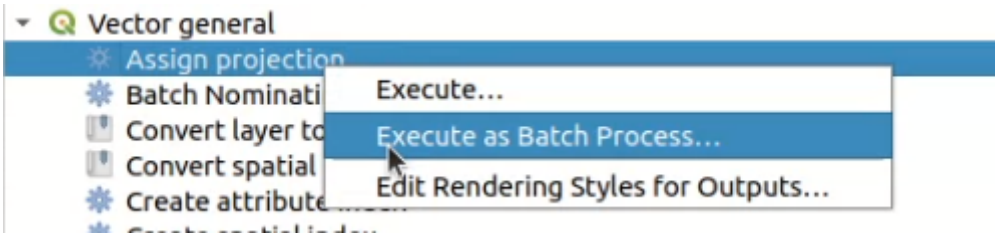


Fig. 27.32: Batch Processing from right-click

Dacă aveți deschis dialogul de execuție a algoritmului, tot de acolo puteți lansa interfața de prelucrare în serie, făcând clic pe butonul de *Executare ca proces în serie…*.

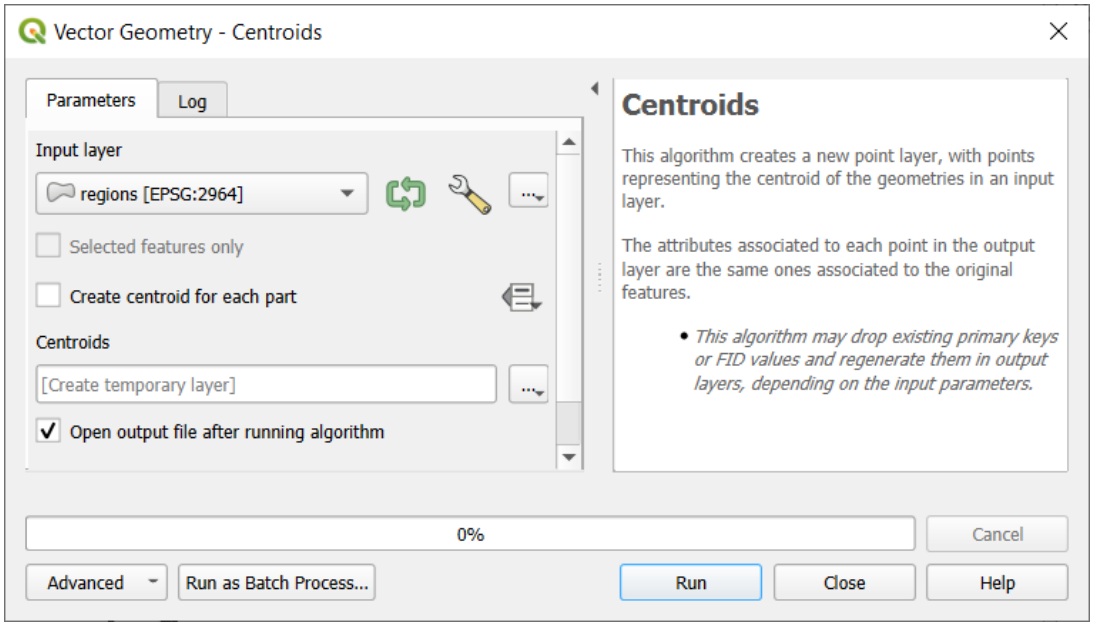


Fig. 27.33: Procesarea în Serie din Dialogul Algoritmului

27.6.2 Tabela parametrilor

Executing a batch process is similar to performing a single execution of an algorithm. Parameter values have to be defined, but in this case we need not just a single value for each parameter, but a set of them instead, one for each time the algorithm has to be executed. Values are introduced using a table like the one shown next, where each row is an iteration and columns are the parameters of the algorithm.

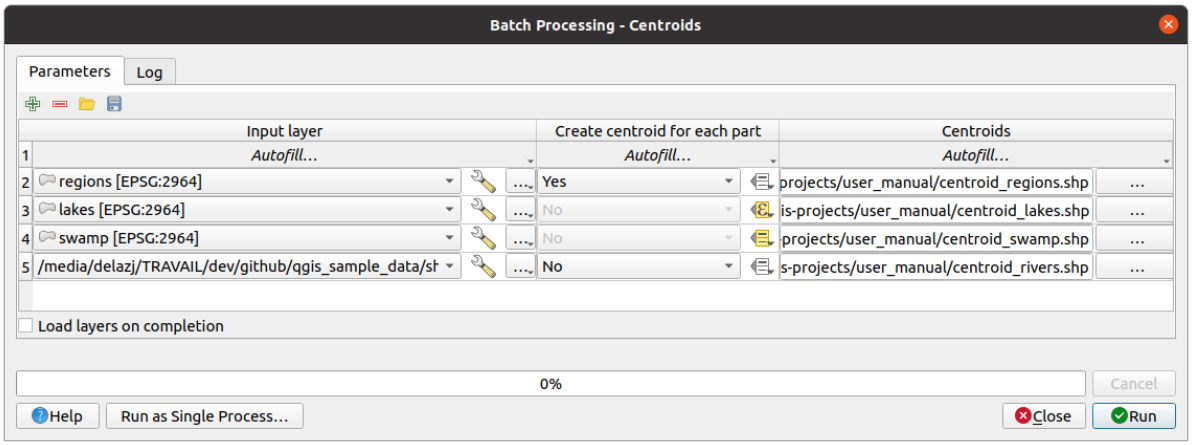


Fig. 27.34: Procesarea în Serie

From the top toolbar, you can:

- Add row: adds a new processing entry for configuration
- Remove row(s): remove selected rows from the table. Row selection is done by clicking the number at the left and allows *keyboard combination* for multi selection.
- Open a batch processing configuration file
- Save the batch processing configuration to a `.JSON` file that can be run afterwards

By default, the table contains just two rows:

- The first row displays in each cell an *Autofill…* ► drop-down menu with *options* to quickly fill the cells below. Available options depend on the parameter type.
- The second row (as well as each subsequent one) represents a single execution of the algorithm, and each cell contains the value of one of the parameters. It is similar to the parameters dialog that you see when executing an algorithm from the toolbox, but with a different arrangement.

At the bottom of the table, you can set whether to *Load layers on completion*.

O dată ce dimensiunea tabelei a fost stabilită, acesta trebuie să fie umplută cu valorile dorite.

27.6.3 Completarea tabelei cu parametri

For most parameters, setting the value is trivial. The appropriate widget, same as in the *single process dialog*, is provided, allowing to just type the value, or select it from a list of possible values, depending on the parameter type. This also includes data-define widget, when compatible.

To automate the batch process definition and avoid filling the table cell by cell, you may want to press down the *Autofill…* menu of a parameter and select any of the following options to replace values in the column:

- *Fill Down* will take the input for the first process and enter it for all other processes.
- *Calculate by Expression…* will allow you to create a new QGIS expression to use to update all existing values within that column. Existing parameter values (including those from other columns) are available for use inside the expression via *variables*. E.g. setting the number of segments based on the buffer distance of each layer:

```
CASE WHEN @DISTANCE > 20 THEN 12 ELSE 8 END
```

- *Add Values by Expression…* will add new rows using the values from an expression which returns an array (as opposed to *Calculate by Expression…*, which works only on existing rows). The intended use case is to allow populating the batch dialog using complex numeric series. For example adding rows for a batch buffer using the expression `generate_series(100, 1000, 50)` for distance parameter results in new rows with values 100, 150, 200, …. 1000.
- When setting a file or layer parameter, more options are provided:
  - *Add Files by Pattern…*: adds new rows to the table for files matching a *File pattern* in a folder to *Look in*. E.g. `*.shp` will add to the list all the `SHP` files in the folder. Check *Search recursively* to also browse sub-folders.
  - *Select Files…* individually on disk
  - *Add All Files from a Directory…*
  - *Select from Open Layers…* in the active project

Output data parameter exposes the same capabilities as when executing the algorithm as a single process. Depending on the algorithm, the output can be:

- skipped, if the cell is left empty
- saved as a temporary layer: fill the cell with `TEMPORARY_OUTPUT` and remember to tick the *Load layers on completion* checkbox.
- saved as a plain file (`.SHP`, `.GPKG`, `.XML`, `.PDF`, `.JPG`,…) whose path could be set with the *Autofill* options exposed beforehand. E.g. use *Calculate by Expression…* to set output file names to complex expressions like:

```
'/home/me/stuff/buffer_' || left(@INPUT, 30) || '_' || @DISTANCE || '.shp'
```

You can also type the file path directly or use the file chooser dialog that appears when clicking on the accompanying ... button. Once you select the file, a new dialog is shown to allow for auto-completion of other cells in the same column (same parameter).

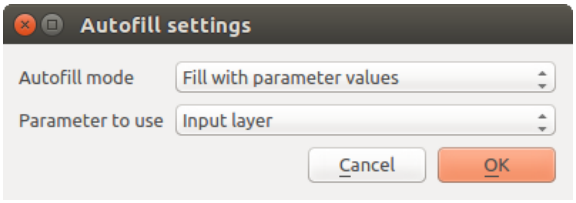


Fig. 27.35: Salvarea Procesării în Serie

If the default value (*Do not autofill*) is selected, it will just put the selected filename in the selected cell from the parameters table. If any of the other options is selected, all the cells **below** the selected one will be automatically filled based on a defined criteria:

- *Fill with numbers*: incrementally appends a number to the file name
- *Fill with parameter values*: you can select a parameter whose value in the same row is appended to the file name. This is particularly useful for naming output data objects according to input ones.

- saved as a layer within a database container:

```
# Indicate a layer within a GeoPackage file
ogr:dbname='C:/Path/To/Geopackage.gpkg' table="New_Table" (geom)

# Use the "Calculate By Expression" to output to different layers in a␣
↪GeoPackage
'ogr:dbname=\'' || @project_folder || '/Buffers.gpkg\' table="' || @INPUT || '_
↪' || @DISTANCE || '" (geom)'
```

27.6.4 Executare ca proces în serie

To execute the batch process once you have introduced all the necessary values, just click on *Run*. The *Log* panel is activated and displays details and steps of the execution process. Progress of the global batch task will be shown in the progress bar in the lower part of the dialog.

27.7 Utilizarea algoritmilor de procesare din consolă

The console allows advanced users to increase their productivity and perform complex operations that cannot be performed using any of the other GUI elements of the processing framework. Models involving several algorithms can be defined using the command-line interface, and additional operations such as loops and conditional sentences can be added to create more flexible and powerful workflows.

There is not a processing console in QGIS, but all processing commands are available instead from the QGIS built-in *Python console*. That means that you can incorporate those commands into your console work and connect processing algorithms to all the other features (including methods from the QGIS API) available from there.

Codul pe care îl puteți executa din consola Python, chiar dacă nu apelează nici o metodă de procesare specifică, poate fi transformat într-un nou algoritm pe care să îl puteți apela mai târziu din caseta de instrumente, din modelatorul grafic sau dintr-o componentă, similar oricărui alt algoritm. De fapt, unii algoritmi pe care îi puteți găsi în caseta de instrumente sunt simple scripturi.

În această secțiune, vom vedea cum se utilizează algoritmii de procesare din Consola Python a QGIS, și, de asemenea, cum să scrieți algoritmi folosind Python.

27.7.1 Apelarea algoritmilor din consola Python

Primul lucru pe care trebuie să-l faceți, este de a importa funcțiile de prelucrare cu ajutorul următoarei linii:

```
>>> from qgis import processing
```

Now, there is basically just one (interesting) thing you can do with that from the console: execute an algorithm. That is done using the `run()` method, which takes the name of the algorithm to execute as its first parameter, and then a variable number of additional parameters depending on the requirements of the algorithm. So the first thing you need to know is the name of the algorithm to execute. That is not the name you see in the toolbox, but rather a unique command–line name. To find the right name for your algorithm, you can use the `processingRegistry`. Type the following line in your console:

```
>>> for alg in QgsApplication.processingRegistry().algorithms():
        print(alg.id(), "->", alg.displayName())
```

You will see something like this (with some extra dashes added to improve readability).

```
3d:tessellate --------------> Tessellate
gdal:aspect ----------------> Aspect
gdal:assignprojection ------> Assign projection
gdal:buffervectors ---------> Buffer vectors
gdal:buildvirtualraster ----> Build Virtual Raster
gdal:cliprasterbyextent ----> Clip raster by extent
gdal:cliprasterbymasklayer -> Clip raster by mask layer
gdal:clipvectorbyextent ----> Clip vector by extent
gdal:clipvectorbypolygon ---> Clip vector by mask layer
gdal:colorrelief -----------> Color relief
gdal:contour ---------------> Contour
gdal:convertformat ---------> Convert format
gdal:dissolve --------------> Dissolve
...
```

That's a list of all the available algorithm IDs, sorted by provider name and algorithm name, along with their corresponding names.

Once you know the command-line name of the algorithm, the next thing to do is to determine the right syntax to execute it. That means knowing which parameters are needed when calling the `run()` method.

There is a method to describe an algorithm in detail, which can be used to get a list of the parameters that an algorithm requires and the outputs that it will generate. To get this information, you can use the `algorithmHelp(id_of_the_algorithm)` method. Use the ID of the algorithm, not the full descriptive name.

Calling the method with `nativ:buffer` as parameter (`qgis:buffer` is an alias for `nativ:buffer` and will also work), you get the following description:

```
>>> processing.algorithmHelp("native:buffer")
Buffer (native:buffer)

This algorithm computes a buffer area for all the features in an
input layer, using a fixed or dynamic distance.

The segments parameter controls the number of line segments to
use to approximate a quarter circle when creating rounded
offsets.

The end cap style parameter controls how line endings are handled
in the buffer.

The join style parameter specifies whether round, miter or
beveled joins should be used when offsetting corners in a line.
```

(continues on next page)

(continuare din pagina precedentă)

```
The miter limit parameter is only applicable for miter join
styles, and controls the maximum distance from the offset curve
to use when creating a mitered join.


----------------
Input parameters
----------------

INPUT: Input layer

   Parameter type: QgsProcessingParameterFeatureSource

   Accepted data types:
           - str: layer ID
           - str: layer name
           - str: layer source
           - QgsProcessingFeatureSourceDefinition
           - QgsProperty
           - QgsVectorLayer

DISTANCE: Distance

   Parameter type: QgsProcessingParameterDistance

   Accepted data types:
           - int
           - float
           - QgsProperty

SEGMENTS: Segments

   Parameter type: QgsProcessingParameterNumber

   Accepted data types:
           - int
           - float
           - QgsProperty

END_CAP_STYLE: End cap style

   Parameter type: QgsProcessingParameterEnum

   Available values:
           - 0: Round
           - 1: Flat
           - 2: Square

   Accepted data types:
           - int
           - str: as string representation of int, e.g. '1'
           - QgsProperty

JOIN_STYLE: Join style

   Parameter type: QgsProcessingParameterEnum

   Available values:
           - 0: Round
           - 1: Miter
```

(continues on next page)

(continuare din pagina precedentă)

```
            - 2: Bevel

   Accepted data types:
            - int
            - str: as string representation of int, e.g. '1'
            - QgsProperty

MITER_LIMIT: Miter limit

   Parameter type: QgsProcessingParameterNumber

   Accepted data types:
            - int
            - float
            - QgsProperty

DISSOLVE: Dissolve result

   Parameter type: QgsProcessingParameterBoolean

   Accepted data types:
            - bool
            - int
            - str
            - QgsProperty

OUTPUT: Buffered

   Parameter type: QgsProcessingParameterFeatureSink

   Accepted data types:
            - str: destination vector file, e.g. 'd:/test.shp'
            - str: 'memory:' to store result in temporary memory layer
            - str: using vector provider ID prefix and destination URI,
                   e.g. 'postgres:...' to store result in PostGIS table
            - QgsProcessingOutputLayerDefinition
            - QgsProperty

----------------
Outputs
----------------

OUTPUT:  <QgsProcessingOutputVectorLayer>
   Buffered
```

Now you have everything you need to run any algorithm. As we have already mentioned, algorithms can be run using: `run()`. Its syntax is as follows:

```
>>> processing.run(name_of_the_algorithm, parameters)
```

Where parameters is a dictionary of parameters that depend on the algorithm you want to run, and is exactly the list that the `algorithmHelp()` method gives you.

```
>>> processing.run("native:buffer", {'INPUT': '/data/lines.shp',
               'DISTANCE': 100.0,
               'SEGMENTS': 10,
               'DISSOLVE': True,
               'END_CAP_STYLE': 0,
               'JOIN_STYLE': 0,
               'MITER_LIMIT': 10,
               'OUTPUT': '/data/buffers.shp'})
```

If a parameter is optional and you do not want to use it, then don't include it in the dictionary.

If a parameter is not specified, the default value will be used.

Depending on the type of parameter, values are introduced differently. The next list gives a quick review of how to introduce values for each type of input parameter:

- Raster Layer, Vector Layer or Table. Simply use a string with the name that identifies the data object to use (the name it has in the QGIS Table of Contents) or a filename (if the corresponding layer is not opened, it will be opened but not added to the map canvas). If you have an instance of a QGIS object representing the layer, you can also pass it as parameter.
- Enumeration. If an algorithm has an enumeration parameter, the value of that parameter should be entered using an integer value. To know the available options, you can use the `algorithmHelp()` command, as above. For instance, the `nativ:buffer` algorithm has an enumeration called JOIN_STYLE:

```
JOIN_STYLE: Join style

   Parameter type: QgsProcessingParameterEnum

   Available values:
           - 0: Round
           - 1: Miter
           - 2: Bevel

   Accepted data types:
           - int
           - str: as string representation of int, e.g. '1'
           - QgsProperty
```

  In this case, the parameter has three options. Notice that ordering is zero-based.

- Boolean. Use `True` or `False`.
- Multiple input. The value is a string with input descriptors separated by semicolons (`;`). As in the case of single layers or tables, each input descriptor can be the data object name, or its file path.
- Table Field from XXX. Use a string with the name of the field to use. This parameter is case-sensitive.
- Tabel fix. Tastați lista tuturor valorilor din tabel, separate prin virgulă (`,`) și incluse între ghilimele (`"`). Introducerea valorilor începe cu rândul de sus și se desfășoară de la stânga la dreapta. De asemenea, puteți utiliza o matrice 2-D cu valori care reprezintă tabelul.
- CRS. Introduceți numărul de cod EPSG pentr CRS-ul dorit.
- Extindere. Trebuie să utilizați un șir cu valorile `xmin`, `xmax`, `ymin` și `ymax` separate prin virgule (`,`).

Parametrii boolean, fișier, șir și numeric nu au nevoie de explicații suplimentare.

Input parameters such as strings, booleans, or numerical values have default values. The default value is used if the corresponding parameter entry is missing.

For output data objects, type the file path to be used to save it, just as it is done from the toolbox. If the output object is not specified, the result is saved to a temporary file (or skipped if it is an optional output). The extension of the file determines the file format. If you enter a file extension not supported by the algorithm, the default file format for that output type will be used, and its corresponding extension appended to the given file path.

Unlike when an algorithm is executed from the toolbox, outputs are not added to the map canvas if you execute that same algorithm from the Python console using `run()`, but `runAndLoadResults()` will do that.

The `run()` method returns a dictionary with one or more output names (the ones shown in the algorithm description) as keys and the file paths of those outputs as values:

```
>>> myresult = processing.run("native:buffer", {'INPUT': '/data/lines.shp',
                'DISTANCE': 100.0,
                'SEGMENTS': 10,
                'DISSOLVE': True,
```

(continues on next page)

(continuare din pagina precedentă)

```
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10,
                'OUTPUT': '/data/buffers.shp'})
>>> myresult['OUTPUT']
/data/buffers.shp
```

You can load feature output by passing the corresponding file paths to the `load()` method. Or you could use `runAndLoadResults()` instead of `run()` to load them immediately.

If you want to open an algorithm dialog from the console you can use the `createAlgorithmDialog` method. The only mandatory parameter is the algorithm name, but you can also define the dictionary of parameters so that the dialog will be filled automatically:

```
>>> my_dialog = processing.createAlgorithmDialog("native:buffer", {
                'INPUT': '/data/lines.shp',
                'DISTANCE': 100.0,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10,
                'OUTPUT': '/data/buffers.shp'})
>>> my_dialog.show()
```

The `execAlgorithmDialog` method opens the dialog immediately:

```
>>> processing.execAlgorithmDialog("native:buffer", {
                'INPUT': '/data/lines.shp',
                'DISTANCE': 100.0,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10,
                'OUTPUT': '/data/buffers.shp'})
```

27.7.2 Crearea scripturilor și execuția lor din caseta de instrumente

You can create your own algorithms by writing Python code. Processing scripts extend `QgsProcessingAlgorithm`, so you need to add some extra lines of code to implement mandatory functions. You can find *Create new script* (clean sheet) and *Create New Script from Template* (template that includes code for mandatory functions of `QgsProcessingAlgorithm`) under the *Scripts* dropdown menu on the top of the Processing toolbox. The Processing Script Editor will open, and that's where you should type your code. Saving the script from there in the `scripts` folder (the default folder when you open the save file dialog) with a `.py` extension should create the corresponding algorithm.

The name of the algorithm (the one you will see in the toolbox) is defined within the code.

Let's have a look at the following code, which defines a Processing algorithm that performs a buffer operation with a user defined buffer distance on a vector layer that is specified by the user, after first smoothing the layer.

```
from qgis.core import (QgsProcessingAlgorithm,
       QgsProcessingParameterNumber,
       QgsProcessingParameterFeatureSource,
       QgsProcessingParameterFeatureSink)

from qgis import processing

```

(continues on next page)

(continuare din pagina precedentă)

```
class algTest(QgsProcessingAlgorithm):
    INPUT_BUFFERDIST = 'BUFFERDIST'
    OUTPUT_BUFFER = 'OUTPUT_BUFFER'
    INPUT_VECTOR = 'INPUT_VECTOR'

    def __init__(self):
        super().__init__()

    def name(self):
        return "algTest"

    def displayName(self):
        return "algTest script"

    def createInstance(self):
        return type(self)()

    def initAlgorithm(self, config=None):
        self.addParameter(QgsProcessingParameterFeatureSource(
            self.INPUT_VECTOR, "Input vector"))
        self.addParameter(QgsProcessingParameterNumber(
            self.INPUT_BUFFERDIST, "Buffer distance",
            QgsProcessingParameterNumber.Double,
            100.0))
        self.addParameter(QgsProcessingParameterFeatureSink(
            self.OUTPUT_BUFFER, "Output buffer"))

    def processAlgorithm(self, parameters, context, feedback):
        #DO SOMETHING
        algresult = processing.run("native:smoothgeometry",
            {'INPUT': parameters[self.INPUT_VECTOR],
             'ITERATIONS':2,
             'OFFSET':0.25,
             'MAX_ANGLE':180,
             'OUTPUT': 'memory:'},
            context=context, feedback=feedback, is_child_algorithm=True)
        smoothed = algresult['OUTPUT']
        algresult = processing.run('native:buffer',
            {'INPUT': smoothed,
            'DISTANCE': parameters[self.INPUT_BUFFERDIST],
            'SEGMENTS': 5,
            'END_CAP_STYLE': 0,
            'JOIN_STYLE': 0,
            'MITER_LIMIT': 10,
            'DISSOLVE': True,
            'OUTPUT': parameters[self.OUTPUT_BUFFER]},
            context=context, feedback=feedback, is_child_algorithm=True)
        buffered = algresult['OUTPUT']
        return {self.OUTPUT_BUFFER: buffered}
```

After doing the necessary imports, the following `QgsProcessingAlgorithm` functions are specified:

- `name()`: The id of the algorithm (lowercase).
- `displayName()`: A human readable name for the algorithm.
- `createInstance()`: Create a new instance of the algorithm class.
- `initAlgorithm()`: Configure the parameterDefinitions and outputDefinitions.

  Here you describe the parameters and output of the algorithm. In this case, a feature source for the input, a feature sink for the result and a number for the buffer distance.

- `processAlgorithm()`: Do the work.

Here we first run the `smoothgeometry` algorithm to smooth the geometry, and then we run the `buffer` algorithm on the smoothed output. To be able to run algorithms from within another algorithm we have to set the `is_child_algorithm` argument to `True`. You can see how input and output parameters are used as parameters to the `smoothgeometry` and `buffer` algorithms.

There are a number of different parameter types available for input and output. Below is an alphabetically sorted list:

Tabelul 27.2: List of input and o

QgsProcessingParameterAggregate	QgsProcessingParameterAnnotationLayer	Qgs
QgsProcessingParameterBoolean	QgsProcessingParameterColor	Qgs
QgsProcessingParameterDatabaseSchema	QgsProcessingParameterDatabaseTable	Qgs
QgsProcessingParameterEnum	QgsProcessingParameterExpression	Qgs
QgsProcessingParameterFeatureSource	QgsProcessingParameterField	Qgs
QgsProcessingParameterFileDestination	QgsProcessingParameterFolderDestination	Qgs
QgsProcessingParameterLayoutItem	QgsProcessingParameterMapLayer	Qgs
QgsProcessingParameterMeshLayer	QgsProcessingParameterMultipleLayers	Qgs
QgsProcessingParameterPointCloudAttribute	QgsProcessingParameterPointCloudLayer	Qgs
QgsProcessingParameterRasterDestination	QgsProcessingParameterRasterLayer	Qgs
QgsProcessingParameterVectorDestination	QgsProcessingParameterVectorLayer	Qgs

The first parameter to the constructors is the name of the parameter, and the second is the description of the parameter (for the user interface). The rest of the constructor parameters are parameter type specific.

The input can be turned into QGIS classes using the `parameterAs` functions of `QgsProcessingAlgorithm`. For instance to get the number provided for the buffer distance as a double:

```
self.parameterAsDouble(parameters, self.INPUT_BUFFERDIST, context)).
```

The `processAlgorithm` function should return a dictionary containing values for every output defined by the algorithm. This allows access to these outputs from other algorithms, including other algorithms contained within the same model.

Well behaved algorithms should define and return as many outputs as makes sense. Non-feature outputs, such as numbers and strings, are very useful when running your algorithm as part of a larger model, as these values can be used as input parameters for subsequent algorithms within the model. Consider adding numeric outputs for things like the number of features processed, the number of invalid features encountered, the number of features output, etc. The more outputs you return, the more useful your algorithm becomes!

Feedback

The `feedback` object passed to `processAlgorithm()` should be used for user feedback / interaction. You can use the `setProgress()` function of the `feedback` object to update the progress bar (0 to 100) to inform the user about the progress of the algorithm. This is very useful if your algorithm takes a long time to complete.

The `feedback` object provides an `isCanceled()` method that should be monitored to enable cancelation of the algorithm by the user. The `pushInfo()` method of `feedback` can be used to send information to the user, and `reportError()` is handy for pushing non-fatal errors to users.

Algorithms should avoid using other forms of providing feedback to users, such as print statements or logging to `QgsMessageLog`, and should always use the feedback object instead. This allows verbose logging for the algorithm, and is also thread-safe (which is important, given that algorithms are typically run in a background thread).

Handling errors

If your algorithm encounters an error which prevents it from executing, such as invalid input values or some other condition from which it cannot or should not recover, then you should raise a `QgsProcessingException`. E.g.:

```
if feature['value'] < 20:
  raise QgsProcessingException('Invalid input value {}, must be >= 20'.
↪format(feature['value']))
```

Try to avoid raising `QgsProcessingException` for non-fatal errors (e.g. when a feature has a null geometry), and instead just report these errors via `feedback.reportError()` and skip the feature. This helps make your algorithm „model-friendly", as it avoids halting the execution of an entire algorithm when a non-fatal error is encountered.

Documentarea scripturilor

As in the case of models, you can create additional documentation for your scripts, to explain what they do and how to use them.

`QgsProcessingAlgorithm` provides the `helpString()`, `shortHelpString()` and `helpUrl()` functions for that purpose. Specify / override these to provide more help to the user.

`shortDescription()` is used in the tooltip when hovering over the algorithm in the toolbox.

27.7.3 Script de interceptare a pre- și post-execuției

Scripts can also be used as pre- and post-execution hooks that are run before and after an algorithm is run, respectively. This can be used to automate tasks that should be performed whenever an algorithm is executed.

The syntax is identical to the syntax explained above, but an additional global variable named `alg` is available, representing the algorithm that has just been (or is about to be) executed.

In the *General* group of the processing options dialog, you will find two entries named *Pre-execution script* and *Post-execution script* where the filenames of the scripts to be run in each case can be entered.

27.8 Using processing from the command line

QGIS comes with a tool called `QGIS Processing Executor` which allows you to run Processing algorithms and models (built-in or provided by plugins) directly from the command line without starting QGIS Desktop itself.

From a command line tool, run `qgis_process` and you should get:

```
QGIS Processing Executor - 3.27.0-Master 'Master' (3.27.0-Master)
Usage: C:\OSGeo4W\apps\qgis-dev\bin\qgis_process.exe [--help] [--version] [--json]␣
↪[--verbose] [--no-python] [command] [algorithm id, path to model file, or path␣
↪to Python script] [parameters]

Options:

  --help or -h      Output the help
  --version or -v   Output all versions related to QGIS Process
  --json            Output results as JSON objects
  --verbose         Output verbose logs
  --no-python       Disable Python support (results in faster startup)

Available commands:
```

(continues on next page)

(continuare din pagina precedentă)

```
  plugins          list available and active plugins
  plugins enable   enables an installed plugin. The plugin name must be specified,␣
↪e.g. "plugins enable cartography_tools"
  plugins disable  disables an installed plugin. The plugin name must be specified,
↪ e.g. "plugins disable cartography_tools"
  list             list all available processing algorithms
  help             show help for an algorithm. The algorithm id or a path to a␣
↪model file must be specified.
  run              runs an algorithm. The algorithm id or a path to a model file␣
↪and parameter values must be specified.
                   Parameter values are specified after -- with PARAMETER=VALUE␣
↪syntax.
                   Ordered list values for a parameter can be created by␣
↪specifying the parameter multiple times,
                   e.g. --LAYERS=layer1.shp --LAYERS=layer2.shp
                   Alternatively, a '-' character in place of the parameters␣
↪argument indicates that the parameters should be read from STDIN as a JSON␣
↪object.
                   The JSON should be structured as a map containing at least the
↪"inputs" key specifying a map of input parameter values.
                   This implies the --json option for output as a JSON object.
                   If required, the ellipsoid to use for distance and area␣
↪calculations can be specified via the "--ELLIPSOID=name" argument.
                   If required, an existing QGIS project to use during the␣
↪algorithm execution can be specified via the "--PROJECT_PATH=path" argument.
```

Notă: Only installed plugins that advertise `hasProcessingProvider=yes` in their `metadata.txt` file are recognized and can be activated or loaded by qgis_process tool.

Sugestie: Before calling qgis_process on a system without window manager (e.g. a headless server), you should set:

```
export QT_QPA_PLATFORM=offscreen
```

The command `list` can be used to get a list of all available providers and algorithms.

```
qgis_process list
```

The command `help` can be used to get further information about commands or algorithms.

```
qgis_process help qgis:regularpoints
```

The command `run` can be used to run an algorithm or model. Specify the name of the algorithm or a path to a model as first parameter.

```
qgis_process run native:buffer -- INPUT=source.shp DISTANCE=2 OUTPUT=buffered.shp
```

Acolo unde un parametru acceptă o listă de valori, setați aceeași variabilă de mai multe ori.

```
qgis_process run native:mergevectorlayers -- LAYERS=input1.shp LAYERS=input2.shp␣
↪OUTPUT=merged.shp
```

While running an algorithm a text-based feedback bar is shown, and the operation can be cancelled via `CTRL+C`.

The `run` command also supports further parameters.

- `--json` va formata rezultatul stdout într-un mod structurat JSON.
- `--ellipsoid` va seta elipsoidul la cel specificat.

- `--distance_units` va folosi unitățile de distanță specificate.
- `--area_units` va folosi unitățile de suptafață specificate.
- `--project_path` va încărca proiectul specificat pentru rularea algoritmului.

Parametrii de intrare complecși, adică tipurile de parametri care sunt ele însele specificate ca obiecte de tip dicționar pentru algoritmi, sunt acceptați de qgis_process. Pentru a indica faptul că parametrii vor fi specificați prin stdin, comanda qgis_process trebuie să urmeze formatul (cu un − la final, în locul listei obișnuite de argumente).

```
qgis_process run algorithmId -
```

The JSON object must contain an „inputs" key, which is a map of the input parameter values. E.g.

```
echo "{'inputs': {'INPUT': 'my_shape.shp', 'DISTANCE': 5}}" | qgis_process run␣
↪native:buffer -
```

Additionally, extra settings like the distance units, area units, ellipsoid and project path can be included in this JSON object:

```
{
 'ellipsoid': 'EPSG:7019',
 'distance_units': 'feet',
 'area_units': 'ha',
 'project_path': 'C:/temp/my_project.qgs'
 'inputs': {'DISTANCE': 5, 'SEGMENTS': 8 ... }
}
```

Specifying input parameters via stdin implies automatically the `JSON` output format for results.

27.9 Writing new Processing algorithms as Python scripts

There are two options for writing Processing algorithms using Python.

- *Extindere* `QgsProcessingAlgorithm`
- *Using the @alg decorator*

Within QGIS, you can use *Create new script* in the *Scripts* menu at the top of the *Processing Toolbox* to open the *Processing Script Editor* where you can write your code. To simplify the task, you can start with a script template by using *Create new script from template* from the same menu. This opens a template that extends `QgsProcessingAlgorithm`.

If you save the script in the `scripts` folder (the default location) with a `.py` extension, the algorithm will become available in the *Processing Toolbox*.

27.9.1 Extending QgsProcessingAlgorithm

The following code

1. takes a vector layer as input
2. counts the number of features
3. does a buffer operation
4. creates a raster layer from the result of the buffer operation
5. returns the buffer layer, raster layer and number of features

```
from qgis.PyQt.QtCore import QCoreApplication
from qgis.core import (QgsProcessing,
                       QgsProcessingAlgorithm,
                       QgsProcessingException,
                       QgsProcessingOutputNumber,
                       QgsProcessingParameterDistance,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterVectorDestination,
                       QgsProcessingParameterRasterDestination)
from qgis import processing


class ExampleProcessingAlgorithm(QgsProcessingAlgorithm):
    """
    This is an example algorithm that takes a vector layer,
    creates some new layers and returns some results.
    """

    def tr(self, string):
        """
        Returns a translatable string with the self.tr() function.
        """
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        # Must return a new copy of your algorithm.
        return ExampleProcessingAlgorithm()

    def name(self):
        """
        Returns the unique algorithm name.
        """
        return 'bufferrasterextend'

    def displayName(self):
        """
        Returns the translated algorithm name.
        """
        return self.tr('Buffer and export to raster (extend)')

    def group(self):
        """
        Returns the name of the group this algorithm belongs to.
        """
        return self.tr('Example scripts')

    def groupId(self):
        """
        Returns the unique ID of the group this algorithm belongs
        to.
        """
        return 'examplescripts'

    def shortHelpString(self):
        """
        Returns a localised short help string for the algorithm.
        """
        return self.tr('Example algorithm short description')

    def initAlgorithm(self, config=None):
        """
        Here we define the inputs and outputs of the algorithm.
```

(continues on next page)

(continuare din pagina precedentă)

```
        """
        # 'INPUT' is the recommended name for the main input
        # parameter.
        self.addParameter(
            QgsProcessingParameterFeatureSource(
                'INPUT',
                self.tr('Input vector layer'),
                types=[QgsProcessing.TypeVectorAnyGeometry]
            )
        )
        self.addParameter(
            QgsProcessingParameterVectorDestination(
                'BUFFER_OUTPUT',
                self.tr('Buffer output'),
            )
        )
        # 'OUTPUT' is the recommended name for the main output
        # parameter.
        self.addParameter(
            QgsProcessingParameterRasterDestination(
                'OUTPUT',
                self.tr('Raster output')
            )
        )
        self.addParameter(
            QgsProcessingParameterDistance(
                'BUFFERDIST',
                self.tr('BUFFERDIST'),
                defaultValue = 1.0,
                # Make distance units match the INPUT layer units:
                parentParameterName='INPUT'
            )
        )
        self.addParameter(
            QgsProcessingParameterDistance(
                'CELLSIZE',
                self.tr('CELLSIZE'),
                defaultValue = 10.0,
                parentParameterName='INPUT'
            )
        )
        self.addOutput(
            QgsProcessingOutputNumber(
                'NUMBEROFFEATURES',
                self.tr('Number of features processed')
            )
        )

    def processAlgorithm(self, parameters, context, feedback):
        """
        Here is where the processing itself takes place.
        """
        # First, we get the count of features from the INPUT layer.
        # This layer is defined as a QgsProcessingParameterFeatureSource
        # parameter, so it is retrieved by calling
        # self.parameterAsSource.
        input_featuresource = self.parameterAsSource(parameters,
                                                     'INPUT',
                                                     context)
        numfeatures = input_featuresource.featureCount()

```

(continues on next page)

(continuare din pagina precedentă)

```
        # Retrieve the buffer distance and raster cell size numeric
        # values. Since these are numeric values, they are retrieved
        # using self.parameterAsDouble.
        bufferdist = self.parameterAsDouble(parameters, 'BUFFERDIST',
                                            context)
        rastercellsize = self.parameterAsDouble(parameters, 'CELLSIZE',
                                                context)
        if feedback.isCanceled():
            return {}
        buffer_result = processing.run(
            'native:buffer',
            {
                # Here we pass on the original parameter values of INPUT
                # and BUFFER_OUTPUT to the buffer algorithm.
                'INPUT': parameters['INPUT'],
                'OUTPUT': parameters['BUFFER_OUTPUT'],
                'DISTANCE': bufferdist,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10
            },
            # Because the buffer algorithm is being run as a step in
            # another larger algorithm, the is_child_algorithm option
            # should be set to True
            is_child_algorithm=True,
            #
            # It's important to pass on the context and feedback objects to
            # child algorithms, so that they can properly give feedback to
            # users and handle cancelation requests.
            context=context,
            feedback=feedback)

        # Check for cancelation
        if feedback.isCanceled():
            return {}

        # Run the separate rasterization algorithm using the buffer result
        # as an input.
        rasterized_result = processing.run(
            'qgis:rasterize',
            {
                # Here we pass the 'OUTPUT' value from the buffer's result
                # dictionary off to the rasterize child algorithm.
                'LAYER': buffer_result['OUTPUT'],
                'EXTENT': buffer_result['OUTPUT'],
                'MAP_UNITS_PER_PIXEL': rastercellsize,
                # Use the original parameter value.
                'OUTPUT': parameters['OUTPUT']
            },
            is_child_algorithm=True,
            context=context,
            feedback=feedback)

        if feedback.isCanceled():
            return {}

        # Return the results
        return {'OUTPUT': rasterized_result['OUTPUT'],
                'BUFFER_OUTPUT': buffer_result['OUTPUT'],
```

(continues on next page)

(continuare din pagina precedentă)

```
185                 'NUMBEROFFEATURES': numfeatures}
```

Processing algorithm standard functions:

- **createInstance (mandatory)**
  Must return a new copy of your algorithm. If you change the name of the class, make sure you also update the value returned here to match!
- **name (mandatory)**
  Returns the unique algorithm name, used for identifying the algorithm.
- **displayName (mandatory)**
  Returns the translated algorithm name.
- **group**
  Returns the name of the group this algorithm belongs to.
- **groupId**
  Returns the unique ID of the group this algorithm belongs to.
- **shortHelpString**
  Returns a localised short help string for the algorithm.
- **initAlgorithm (mandatory)**
  Here we define the inputs and outputs of the algorithm.

  `INPUT` and `OUTPUT` are recommended names for the main input and main output parameters, respectively.

  If a parameter depends on another parameter, `parentParameterName` is used to specify this relationship (could be the field / band of a layer or the distance units of a layer).
- **processAlgorithm (mandatory)**
  This is where the processing takes place.

  Parameters are retrieved using special purpose functions, for instance `parameterAsSource` and `parameterAsDouble`.

  `processing.run` can be used to run other processing algorithms from a processing algorithm. The first parameter is the name of the algorithm, the second is a dictionary of the parameters to the algorithm. `is_child_algorithm` is normally set to `True` when running an algorithm from within another algorithm. `context` and `feedback` inform the algorithm about the environment to run in and the channel for communicating with the user (catching cancel request, reporting progress, providing textual feedback). When using the (parent) algorithm's parameters as parameters to „child" algorithms, the original parameter values should be used (e.g. `parameters['OUTPUT']`).

  It is good practice to check the feedback object for cancelation as much as is sensibly possible! Doing so allows for responsive cancelation, instead of forcing users to wait for unwanted processing to occur.

  The algorithm should return values for all the output parameters it has defined as a dictionary. In this case, that's the buffer and rasterized output layers, and the count of features processed. The dictionary keys must match the original parameter/output names.

27.9.2 The @alg decorator

Using the @alg decorator, you can create your own algorithms by writing the Python code and adding a few extra lines to supply additional information needed to make it a proper Processing algorithm. This simplifies the creation of algorithms and the specification of inputs and outputs.

One important limitation with the decorator approach is that algorithms created in this way will always be added to a user's Processing Scripts provider – it is not possible to add these algorithms to a custom provider, e.g. for use in plugins.

The following code uses the @alg decorator to

1. use a vector layer as input
2. count the number of features
3. do a buffer operation
4. create a raster layer from the result of the buffer operation
5. returns the buffer layer, raster layer and number of features

```
from qgis import processing
from qgis.processing import alg
from qgis.core import QgsProject

@alg(name='bufferrasteralg', label='Buffer and export to raster (alg)',
     group='examplescripts', group_label='Example scripts')
# 'INPUT' is the recommended name for the main input parameter
@alg.input(type=alg.SOURCE, name='INPUT', label='Input vector layer')
# 'OUTPUT' is the recommended name for the main output parameter
@alg.input(type=alg.RASTER_LAYER_DEST, name='OUTPUT',
           label='Raster output')
@alg.input(type=alg.VECTOR_LAYER_DEST, name='BUFFER_OUTPUT',
           label='Buffer output')
@alg.input(type=alg.DISTANCE, name='BUFFERDIST', label='BUFFER DISTANCE',
           default=1.0)
@alg.input(type=alg.DISTANCE, name='CELLSIZE', label='RASTER CELL SIZE',
           default=10.0)
@alg.output(type=alg.NUMBER, name='NUMBEROFFEATURES',
            label='Number of features processed')

def bufferrasteralg(instance, parameters, context, feedback, inputs):
    """
    Description of the algorithm.
    (If there is no comment here, you will get an error)
    """
    input_featuresource = instance.parameterAsSource(parameters,
                                                     'INPUT', context)
    numfeatures = input_featuresource.featureCount()
    bufferdist = instance.parameterAsDouble(parameters, 'BUFFERDIST',
                                            context)
    rastercellsize = instance.parameterAsDouble(parameters, 'CELLSIZE',
                                                context)
    if feedback.isCanceled():
        return {}
    buffer_result = processing.run('native:buffer',
                                   {'INPUT': parameters['INPUT'],
                                    'OUTPUT': parameters['BUFFER_OUTPUT'],
                                    'DISTANCE': bufferdist,
                                    'SEGMENTS': 10,
                                    'DISSOLVE': True,
                                    'END_CAP_STYLE': 0,
                                    'JOIN_STYLE': 0,
```

(continues on next page)

(continuare din pagina precedentă)

```
                                'MITER_LIMIT': 10
                                },
                               is_child_algorithm=True,
                               context=context,
                               feedback=feedback)
    if feedback.isCanceled():
        return {}
    rasterized_result = processing.run('qgis:rasterize',
                               {'LAYER': buffer_result['OUTPUT'],
                                'EXTENT': buffer_result['OUTPUT'],
                                'MAP_UNITS_PER_PIXEL': rastercellsize,
                                'OUTPUT': parameters['OUTPUT']
                               },
                               is_child_algorithm=True, context=context,
                               feedback=feedback)
    if feedback.isCanceled():
        return {}
    return {'OUTPUT': rasterized_result['OUTPUT'],
            'BUFFER_OUTPUT': buffer_result['OUTPUT'],
            'NUMBEROFFEATURES': numfeatures}
```

As you can see, it involves two algorithms («native:buffer» and «qgis:rasterize»). The last one («qgis:rasterize») creates a raster layer from the buffer layer that was generated by the first one («native:buffer»).

The part of the code where this processing takes place is not difficult to understand if you have read the previous chapter. The first lines, however, need some additional explanation. They provide the information that is needed to turn your code into an algorithm that can be run from any of the GUI components, like the toolbox or the model designer.

These lines are all calls to the `@alg` decorator functions that help simplify the coding of the algorithm.

- The @alg decorator is used to define the name and location of the algorithm in the Toolbox.
- The @alg.input decorator is used to define the inputs of the algorithm.
- The @alg.output decorator is used to define the outputs of the algorithm.

27.9.3 Input and output types for Processing Algorithms

Here is the list of input and output types that are supported in Processing with their corresponding alg decorator constants (the algfactory.py file contains the complete list of alg constants). Sorted on class name.

Input types

Clasa	Alg constant	Descriere
QgsProcessingParameterAnnotationL	alg.ANNOTATION_LAYER	An annotation layer
QgsProcessingParameterAuthConfig	alg.AUTH_CFG	Allows users to select from available authentication configurations or create new authentication configurations
QgsProcessingParameterBand	alg.BAND	A band of a raster layer
QgsProcessingParameterBoolean	alg.BOOL	A boolean value
QgsProcessingParameterColor	alg.COLOR	A color

continues on next page

Tabelul 27.3 – continuare din pagina precedentă

Clasa	Alg constant	Descriere
QgsProcessingParameterCoordinateO	alg. COORDINATE_OPERATION	A coordinate operation (for CRS transformations)
QgsProcessingParameterCrs	alg.CRS	A Coordinate Reference System
QgsProcessingParameterDatabaseSch	alg.DATABASE_SCHEMA	A database schema
QgsProcessingParameterDatabaseTab	alg.DATABASE_TABLE	A database table
QgsProcessingParameterDateTime	alg.DATETIME	A datetime (or a pure date or time)
QgsProcessingParameterDistance	alg.DISTANCE	A double numeric parameter for distance values
QgsProcessingParameterEnum	alg.ENUM	An enumeration, allowing for selection from a set of predefined values
QgsProcessingParameterExpression	alg.EXPRESSION	An expression
QgsProcessingParameterExtent	alg.EXTENT	A spatial extent defined by xmin, xmax, ymin, ymax
QgsProcessingParameterField	alg.FIELD	A field in the attribute table of a vector layer
QgsProcessingParameterFile	alg.FILE	A filename of an existing file
QgsProcessingParameterFileDestina	alg.FILE_DEST	A filename for a newly created output file
QgsProcessingParameterFolderDesti	alg.FOLDER_DEST	A folder (destination folder)
QgsProcessingParameterGeometry	alg.GEOMETRY	A geometry
QgsProcessingParameterNumber	alg.INT	An integer
QgsProcessingParameterLayout	alg.LAYOUT	A layout
QgsProcessingParameterLayoutItem	alg.LAYOUT_ITEM	A layout item
QgsProcessingParameterMapLayer	alg.MAPLAYER	A map layer
QgsProcessingParameterMapTheme	alg.MAP_THEME	A project map theme
QgsProcessingParameterMatrix	alg.MATRIX	A matrix
QgsProcessingParameterMeshLayer	alg.MESH_LAYER	A mesh layer
QgsProcessingParameterMultipleLay	alg.MULTILAYER	A set of layers
QgsProcessingParameterNumber	alg.NUMBER	A numerical value
QgsProcessingParameterPoint	alg.POINT	A point
QgsProcessingParameterPointCloudD	alg. POINTCLOUD_LAYER_DEST	A point cloud layer destination parameter, for specifying the destination path for a point cloud layer created by the algorithm
QgsProcessingParameterPointCloudL	alg.POINTCLOUD_LAYER	A point cloud layer
QgsProcessingParameterProviderCon	alg. PROVIDER_CONNECTION	An available connection for a database provider
QgsProcessingParameterRange	alg.RANGE	A number range
QgsProcessingParameterRasterLayer	alg.RASTER_LAYER	Un strat raster

continues on next page

Tabelul 27.3 – continuare din pagina precedentă

Clasa	Alg constant	Descriere
QgsProcessingParameterRasterDesti	alg.RASTER_LAYER_DEST	A raster layer destination parameter, for specifying the destination path for a raster layer created by the algorithm
QgsProcessingParameterScale	alg.SCALE	A map scale
QgsProcessingParameterFeatureSink	alg.SINK	A feature sink
QgsProcessingParameterFeatureSour	alg.SOURCE	A feature source
QgsProcessingParameterString	alg.STRING	Un șir de tip text
QgsProcessingParameterVectorLayer	alg.VECTOR_LAYER	Un strat vectorial
QgsProcessingParameterVectorDesti	alg.VECTOR_LAYER_DEST	A vector layer destination parameter, for specifying the destination path for a vector layer created by the algorithm
QgsProcessingParameterVectorTileD		A vector tile layer destination parameter, for specifying the destination path for a vector tile layer created by the algorithm

Output types

Clasa	Alg constant	Descriere
QgsProcessingOutputBoolean	alg.BOOL	A boolean value
QgsProcessingOutputNumber	alg.DISTANCE	A double numeric parameter for distance values
QgsProcessingOutputFile	alg.FILE	A filename of an existing file
QgsProcessingOutputFolder	alg.FOLDER	A folder
QgsProcessingOutputHtml	alg.HTML	HTML
QgsProcessingOutputNumber	alg.INT	A integer
QgsProcessingOutputLayerDefinition	alg.LAYERDEF	A layer definition
QgsProcessingOutputMapLayer	alg.MAPLAYER	A map layer
QgsProcessingOutputMultipleLayers	alg.MULTILAYER	A set of layers
QgsProcessingOutputNumber	alg.NUMBER	A numerical value
QgsProcessingOutputPointCloudLayer	alg.POINTCLOUD_LAYER	A point cloud layer
QgsProcessingOutputRasterLayer	alg.RASTER_LAYER	Un strat raster
QgsProcessingOutputString	alg.STRING	Un șir de tip text
QgsProcessingOutputVectorLayer	alg.VECTOR_LAYER	Un strat vectorial
QgsProcessingOutputVectorTileLayer		A vector tile layer

27.9.4 Handing algorithm output

When you declare an output representing a layer (raster or vector), the algorithm will try to add it to QGIS once it is finished.

- Raster layer output: QgsProcessingParameterRasterDestination / alg.RASTER_LAYER_DEST.
- Vector layer output: QgsProcessingParameterVectorDestination / alg.VECTOR_LAYER_DEST.

So even if the `processing.run()` method does not add the layers it creates to the user's current project, the two output layers (buffer and raster buffer) will be loaded, since they are saved to the destinations entered by the user (or to temporary destinations if the user does not specify destinations).

If a layer is created as output of an algorithm, it should be declared as such. Otherwise, you will not be able to properly use the algorithm in the modeler, since what is declared will not match what the algorithm really creates.

You can return strings, numbers and more by specifying them in the result dictionary (as demonstrated for „NUMBEROFFEATURES"), but they should always be explicitly defined as outputs from your algorithm. We encourage algorithms to output as many useful values as possible, since these can be valuable for use in later algorithms when your algorithm is used as part of a model.

27.9.5 Comunicarea cu utilizatorul

If your algorithm takes a long time to process, it is a good idea to inform the user about the progress. You can use `feedback` (`QgsProcessingFeedback`) for this.

The progress text and progressbar can be updated using two methods: `setProgressText(text)` and `setProgress(percent)`.

You can provide more information by using `pushCommandInfo(text)`, `pushDebugInfo(text)`, `pushInfo(text)` and `reportError(text)`.

If your script has a problem, the correct way of handling it is to raise a `QgsProcessingException`. You can pass a message as an argument to the constructor of the exception. Processing will take care of handling it and communicating with the user, depending on where the algorithm is being executed from (toolbox, modeler, Python console, …)

27.9.6 Documentarea scripturilor

You can document your scripts by overloading the `helpString()` and `helpUrl()` methods of `QgsProcessingAlgorithm`.

27.9.7 Flags

You can override the `flags()` method of `QgsProcessingAlgorithm` to tell QGIS more about your algorithm. You can for instance tell QGIS that the script shall be hidden from the modeler, that it can be canceled, that it is not thread safe, and more.

Sfat: By default, Processing runs algorithms in a separate thread in order to keep QGIS responsive while the processing task runs. If your algorithm is regularly crashing, you are probably using API calls which are not safe to do in a background thread. Try returning the QgsProcessingAlgorithm.FlagNoThreading flag from your algorithm's flags() method to force Processing to run your algorithm in the main thread instead.

27.9.8 Cele mai bune practici pentru scrierea algoritmilor pentru script

Here's a quick summary of ideas to consider when creating your script algorithms and, especially, if you want to share them with other QGIS users. Following these simple rules will ensure consistency across the different Processing elements such as the toolbox, the modeler or the batch processing interface.

- Do not load resulting layers. Let Processing handle your results and load your layers if needed.
- Always declare the outputs your algorithm creates.
- Do not show message boxes or use any GUI element from the script. If you want to communicate with the user, use the methods of the feedback object (`QgsProcessingFeedback`) or throw a `QgsProcessingException`.

There are already many processing algorithms available in QGIS. You can find code on the QGIS repo.

27.10 Configurarea Aplicațiilor Externe

The processing framework can be extended using additional applications. Algorithms that rely on external applications are managed by their own algorithm providers. Additional providers can be found as separate plugins, and installed using the QGIS Plugin Manager.

This section will show you how to configure the Processing framework to include these additional applications, and it will explain some particular features of the algorithms based on them. Once you have correctly configured the system, you will be able to execute external algorithms from any component like the toolbox or the model designer, just like you do with any other algorithm.

By default, algorithms that rely on an external application not shipped with QGIS are not enabled. You can enable them in the Processing settings dialog if they are installed on your system.

27.10.1 O notă pentru utilizatorii de Windows

If you are not an advanced user and you are running QGIS on Windows, you might not be interested in reading the rest of this chapter. Make sure you install QGIS in your system using the standalone installer. That will automatically install SAGA and GRASS in your system and configure them so they can be run from QGIS. All the algorithms from these providers will be ready to be run without needing any further configuration. If installing with the OSGeo4W application, make sure that you also select SAGA and GRASS for installation.

27.10.2 O notă privind formatele de fișiere

When using external software, opening a file in QGIS does not mean that it can be opened and processed in that other software. In most cases, other software can read what you have opened in QGIS, but in some cases, that might not be true. When using databases or uncommon file formats, whether for raster or vector layers, problems might arise. If that happens, try to use well-known file formats that you are sure are understood by both programs, and check the console output (in the log panel) to find out what is going wrong.

You might for instance get trouble and not be able to complete your work if you call an external algorithm with a GRASS raster layers as input. For this reason, such layers will not appear as available to algorithms.

You should, however, not have problems with vector layers, since QGIS automatically converts from the original file format to one accepted by the external application before passing the layer to it. This adds extra processing time, which might be significant for large layers, so do not be surprised if it takes more time to process a layer from a DB connection than a layer from a Shapefile format dataset of similar size.

Providers not using external applications can process any layer that you can open in QGIS, since they open it for analysis through QGIS.

All raster and vector output formats produced by QGIS can be used as input layers. Some providers do not support certain formats, but all can export to common formats that can later be transformed by QGIS automatically. As for input layers, if a conversion is needed, that might increase the processing time.

27.10.3 O notă privind selecțiile stratului vectorial

External applications may also be made aware of the selections that exist in vector layers within QGIS. However, that requires rewriting all input vector layers, just as if they were originally in a format not supported by the external application. Only when no selection exists, or the *Use only selected features* option is not enabled in the processing general configuration, can a layer be directly passed to an external application.

In other cases, exporting only selected features is needed, which causes longer execution times.

27.10.4 Using third-party Providers

SAGA

SAGA algorithms can be run from QGIS if SAGA is included with the QGIS installation.

If you are running Windows, both the stand-alone installer and the OSGeo4W installer include SAGA.

Despre limitările sistemului grilă SAGA

Most SAGA algorithms that require several input raster layers require them to have the same grid system. That is, they must cover the same geographic area and have the same cell size, so their corresponding grids match. When calling SAGA algorithms from QGIS, you can use any layer, regardless of its cell size and extent. When multiple raster layers are used as input for a SAGA algorithm, QGIS resamples them to a common grid system and then passes them to SAGA (unless the SAGA algorithm can operate with layers from different grid systems).

Definirea acestui sistem grilă comun este controlată de către utilizator, acest lucru putând făcut din zona grupului SAGA al ferestrei de setare. Există două modalități de stabilire a sistemului grilă țintă:

- Setare manuală. Puteți defini extinderea, prin stabilirea valorilor următorilor parametri:
  - *Reeșantionare min X*
  - *Reeșantionare max X*
  - *Reeșantionare min Y*
  - *Reeșantionare max Y*
  - *Reeșantionare dimensiune celulă*

  Rețineți că aplicația QGIS va reeșantiona straturile de intrare la această extindere, chiar dacă acestea nu se suprapun cu ea.
- Setare automată din straturile de intrare. Pentru a selecta această opțiune, doar să verificați opțiunea *Use min covering grid system for resampling*. Toate celelalte setări vor fi ignorate, iar extinderea minimă care acoperă toate straturile de intrare va fi utilizată. Dimensiunea celulei din stratul țintă reprezintă maximul dimensiunilor tuturor celulelor din straturile de intrare.

For algorithms that do not use multiple raster layers, or for those that do not need a unique input grid system, no resampling is performed before calling SAGA, and those parameters are not used.

Limitări pentru straturile multi-bandă

Unlike QGIS, SAGA has no support for multi-band layers. If you want to use a multiband layer (such as an RGB or multispectral image), you first have to split it into single-banded images. To do so, you can use the «SAGA/Grid - Tools/Split RGB image» algorithm (which creates three images from an RGB image) or the «SAGA/Grid - Tools/Extract band» algorithm (to extract a single band).

Limitări de mărime a celulelor

SAGA assumes that raster layers have the same cell size in the X and Y axis. If you are working with a layer with different values for horizontal and vertical cell size, you might get unexpected results. In this case, a warning will be added to the processing log, indicating that an input layer might not be suitable to be processed by SAGA.

Jurnalizare

When QGIS calls SAGA, it does so using its command-line interface, thus passing a set of commands to perform all the required operations. SAGA shows its progress by writing information to the console, which includes the percentage of processing already done, along with additional content. This output is filtered and used to update the progress bar while the algorithm is running.

Both the commands sent by QGIS and the additional information printed by SAGA can be logged along with other processing log messages, and you might find them useful to track what is going on when QGIS runs a SAGA algorithm. You will find two settings, namely *Log console output* and *Log execution commands*, to activate that logging mechanism.

Most other providers that use external applications and call them through the command-line have similar options, so you will find them as well in other places in the processing settings list.

R scripts and libraries

To enable R in Processing you need to install the **Processing R Provider** plugin and configure R for QGIS.

Configuration is done in *Provider* ► *R* in the *Processing* tab of *Settings* ► *Options*.

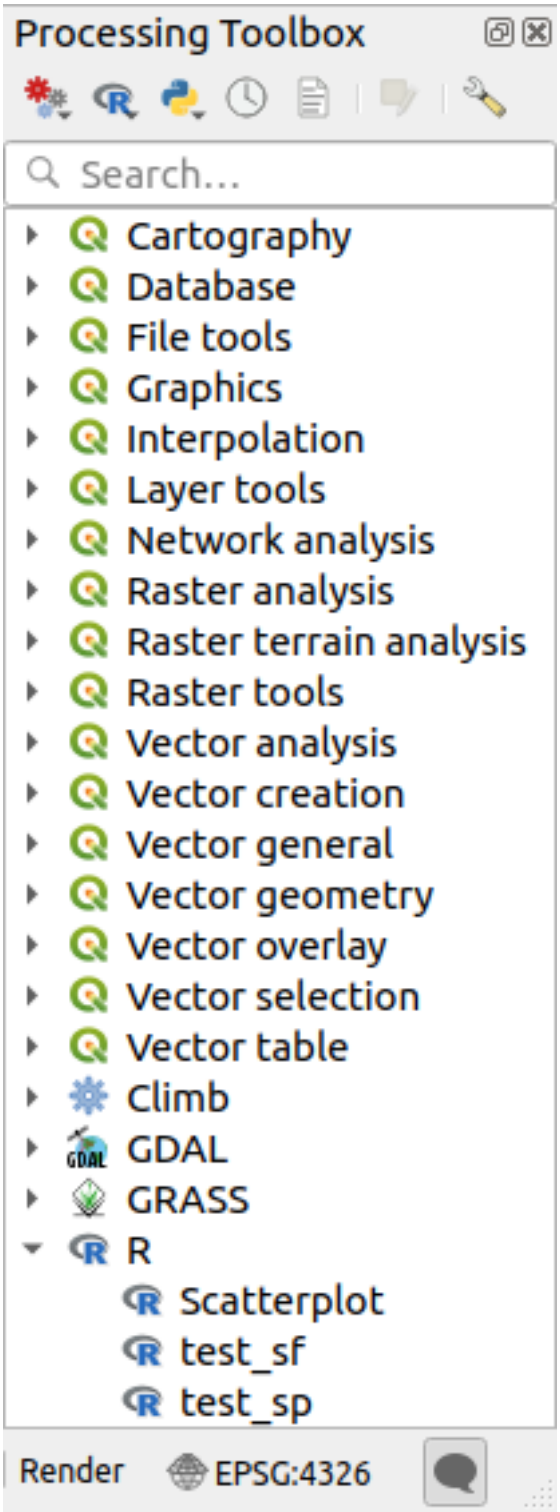
Depending on your operating system, you may have to use *R folder* to specify where your R binaries are located.

Notă: On **Windows** the R executable file is normally in a folder (`R-<version>`) under `C:\Program Files\R\`. Specify the folder and **NOT** the binary!

On **Linux** you just have to make sure that the R folder is in the PATH environment variable. If `R` in a terminal window starts R, then you are ready to go.

After installing the **Processing R Provider** plugin, you will find some example scripts in the *Processing Toolbox*:

- *Scatterplot* runs an R function that produces a scatter plot from two numerical fields of the provided vector layer.
- *test_sf* does some operations that depend on the `sf` package and can be used to check if the R package `sf` is installed. If the package is not installed, R will try to install it (and all the packages it depends on) for you, using the *Package repository* specified in *Provider* ► *R* in the Processing options. The default is https://cran.r-project.org/. Installing may take some time…
- *test_sp* can be used to check if the R package `sp` is installed. If the package is not installed, R will try to install it for you.



If you have R configured correctly for QGIS, you should be able to run these scripts.

Adding R scripts from the QGIS collection

R integration in QGIS is different from that of SAGA in that there is not a predefined set of algorithms you can run (except for some example script that come with the *Processing R Provider* plugin).

A set of example R scripts is available in the QGIS Repository. Perform the following steps to load and enable them using the *QGIS Resource Sharing* plugin.

1. Add the *QGIS Resource Sharing* plugin (you may have to enable *Show also experimental plugins* in the Plugin Manager *Settings*)
2. Open it (*Plugins* ► *Resource Sharing* ► *Resource Sharing*)
3. Choose the *Settings* tab
4. Click *Reload repositories*
5. Choose the *All* tab
6. Select *QGIS R script collection* in the list and click on the *Install* button
7. The collection should now be listed in the *Installed* tab
8. Close the plugin
9. Open the *Processing Toolbox*, and if everything is OK, the example scripts will be present under R, in various groups (only some of the groups are expanded in the screenshot below).

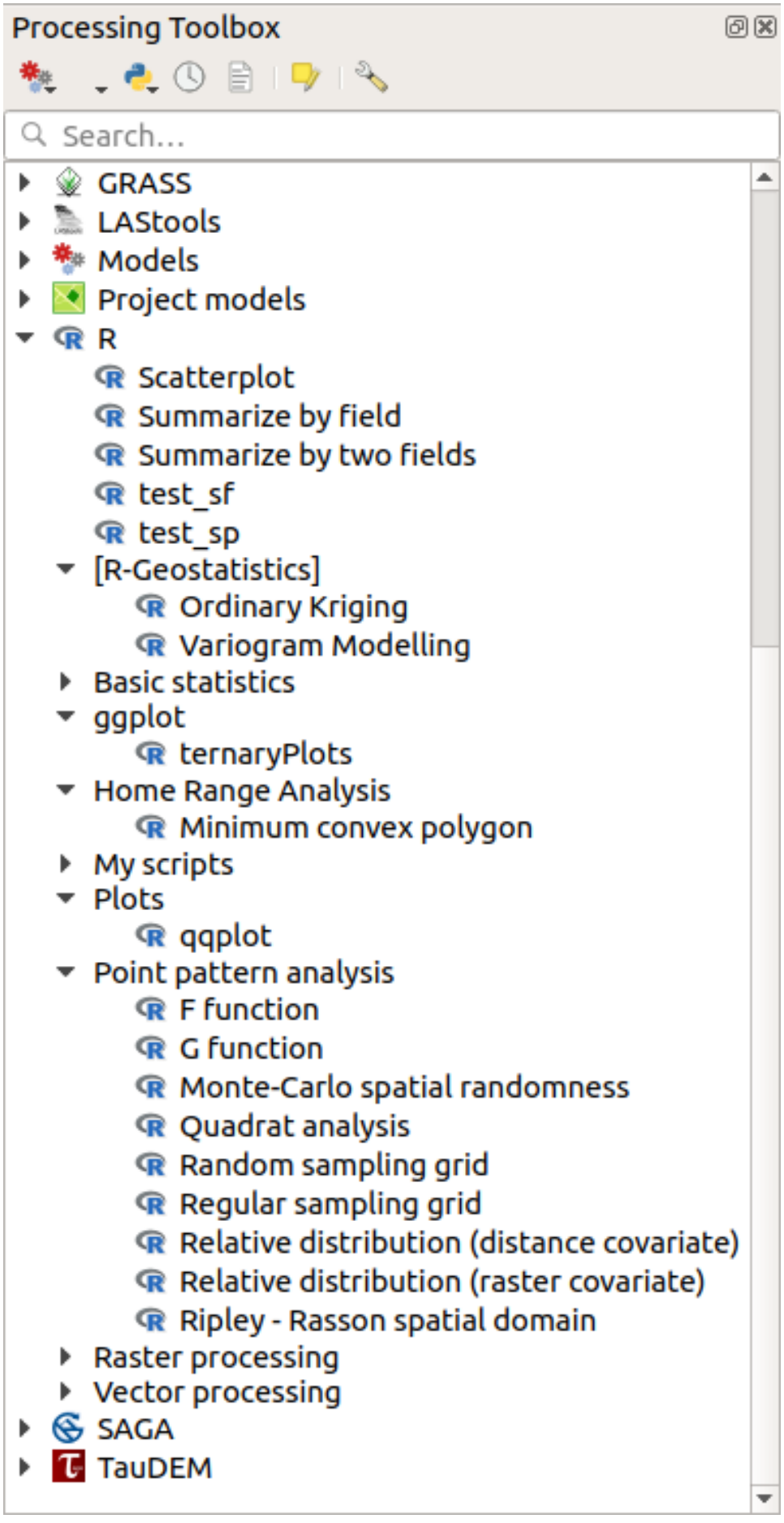
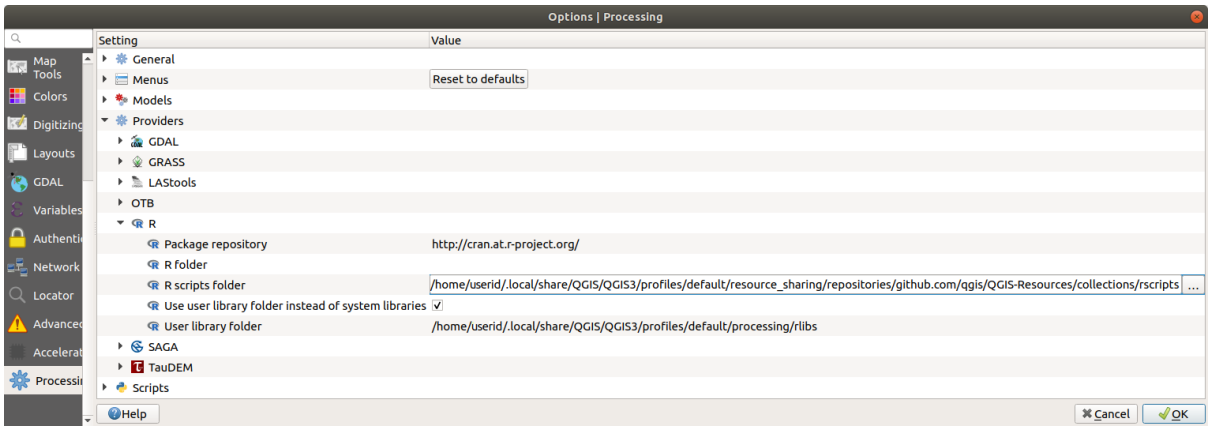


Fig. 27.36: The *Processing Toolbox* with some R scripts shown

The scripts at the top are the example scripts from the *Processing R Provider* plugin.

10. If, for some reason, the scripts are not available in the *Processing Toolbox*, you can try to:
    1. Open the Processing settings (*Settings* ► *Options* ► *Processing* tab)
    2. Go to *Providers* ► *R* ► *R scripts folder*
       - On Ubuntu, set the path to (or, better, include in the path):

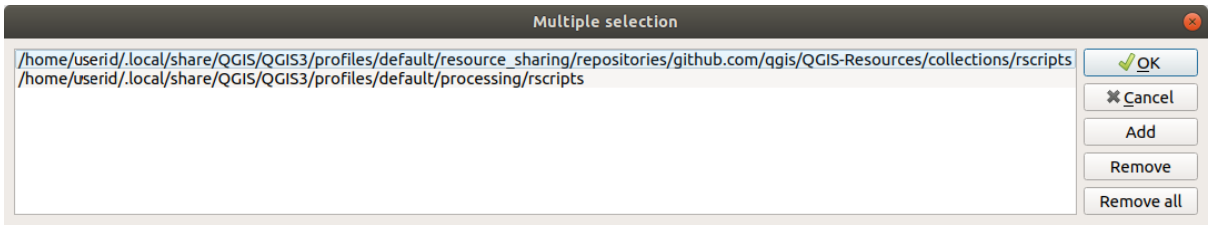
         /home/<user>/.local/share/QGIS/QGIS3/profiles/default/resource_sharing/repositories/github.com/qgis/QGIS-Resources/collections/rscripts



       - On Windows, set the path to (or, better, include in the path):

         C:\Users\<user>\AppData\Roaming\QGIS\QGIS3\profiles\default\resource_sharing\repositories\github.com\qgis\QGIS-Resources\collections\rscripts

    To edit, double-click. You can then choose to just paste / type the path, or you can navigate to the directory by using the … button and press the *Add* button in the dialog that opens. It is possible to provide several directories here. They will be separated by a semicolon („;”).



If you would like to get all the R scrips from the QGIS 2 on-line collection, you can select *QGIS R script collection (from QGIS 2)* instead of *QGIS R script collection*. You will probably find that scripts that depend on vector data input or output will not work.

Creating R scripts

You can write scripts and call R commands, as you would do from R. This section shows you the syntax for using R commands in QGIS, and how to use QGIS objects (layers, tables) in them.

To add an algorithm that calls an R function (or a more complex R script that you have developed and you would like to have available from QGIS), you have to create a script file that performs the R commands.

R script files have the extension `.rsx`, and creating them is pretty easy if you just have a basic knowledge of R syntax and R scripting. They should be stored in the R scripts folder. You can specify the folder (*R scripts folder*) in the *R* settings group in Processing settings dialog).

Let's have a look at a very simple script file, which calls the R method `spsample` to create a random grid within the boundary of the polygons in a given polygon layer. This method belongs to the `maptools` package. Since almost all the algorithms that you might like to incorporate into QGIS will use or generate spatial data, knowledge of spatial packages like `maptools` and `sp`/`sf`, is very useful.

```
##Random points within layer extent=name
##Point pattern analysis=group
##Vector_layer=vector
##Number_of_points=number 10
##Output=output vector
library(sp)
spatpoly = as(Vector_layer, "Spatial")
pts=spsample(spatpoly,Number_of_points,type="random")
spdf=SpatialPointsDataFrame(pts, as.data.frame(pts))
Output=st_as_sf(spdf)
```

The first lines, which start with a double Python comment sign (`##`), define the display name and group of the script, and tell QGIS about its inputs and outputs.

Notă: To find out more about how to write your own R scripts, have a look at the R Intro section in the training manual and consult the *QGIS R Syntax* section.

When you declare an input parameter, QGIS uses that information for two things: creating the user interface to ask the user for the value of that parameter, and creating a corresponding R variable that can be used as R function input.

In the above example, we have declared an input of type `vector`, named `Vector_layer`. When executing the algorithm, QGIS will open the layer selected by the user and store it in a variable named `Vector_layer`. So, the name of a parameter is the name of the variable that you use in R for accessing the value of that parameter (you should therefore avoid using reserved R words as parameter names).

Spatial parameters such as vector and raster layers are read using the `st_read()` (or `readOGR`) and `brick()` (or `readGDAL`) commands (you do not have to worry about adding those commands to your description file – QGIS will do it), and they are stored as `sf` (or `Spatial*DataFrame`) objects.

Table fields are stored as strings containing the name of the selected field.

Vector files can be read using the `readOGR()` command instead of `st_read()` by specifying `##load_vector_using_rgdal`. This will produce a `Spatial*DataFrame` object instead of an `sf` object.

Raster files can be read using the `readGDAL()` command instead of `brick()` by specifying `##load_raster_using_rgdal`.

If you are an advanced user and do not want QGIS to create the object for the layer, you can use `##pass_filenames` to indicate that you prefer a string with the filename. In this case, it is up to you to open the file before performing any operation on the data it contains.

With the above information, it is possible to understand the first lines of the R script (the first line not starting with a Python comment character).

```
library(sp)
spatpoly = as(Vector_layer, "Spatial")
pts=spsample(polyg,numpoints,type="random")
```

The `spsample` function is provided by the *sp* library, so the first thing we do is to load that library. The variable `Vector_layer` contains an `sf` object. Since we are going to use a function (`spsample`) from the *sp* library, we must convert the `sf` object to a `SpatialPolygonsDataFrame` object using the `as` function.

Then we call the `spsample` function with this object and the `numpoints` input parameter (which specifies the number of points to generate).

Since we have declared a vector output named `Output`, we have to create a variable named `Output` containing an `sf` object.

We do this in two steps. First we create a `SpatialPolygonsDataFrame` object from the result of the function, using the *SpatialPointsDataFrame* function, and then we convert that object to an `sf` object using the `st_as_sf` function (of the *sf* library).

You can use whatever names you like for your intermediate variables. Just make sure that the variable storing your final result has the defined name (in this case `Output`), and that it contains a suitable value (an `sf` object for vector layer output).

In this case, the result obtained from the `spsample` method had to be converted explicitly into an `sf` object via a `SpatialPointsDataFrame` object, since it is itself an object of class `ppp`, which can not be returned to QGIS.

If your algorithm generates raster layers, the way they are saved will depend on whether or not you have used the `##dontuserasterpackage` option. If you have used it, layers are saved using the `writeGDAL()` method. If not, the `writeRaster()` method from the `raster` package will be used.

If you have used the `##pass_filenames` option, outputs are generated using the `raster` package (with `writeRaster()`).

If your algorithm does not generate a layer, but a text result in the console instead, you have to indicate that you want the console to be shown once the execution is finished. To do so, just start the command lines that produce the results you want to print with the > («greater than») sign. Only output from lines prefixed with > are shown. For instance, here is the description file of an algorithm that performs a normality test on a given field (column) of the attributes of a vector layer:

```
##layer=vector
##field=field layer
##nortest=group
library(nortest)
>lillie.test(layer[[field]])
```

The output of the last line is printed, but the output of the first is not (and neither are the outputs from other command lines added automatically by QGIS).

If your algorithm creates any kind of graphics (using the `plot()` method), add the following line (`output_plots_to_html` used to be `showplots`):

```
##output_plots_to_html
```

This will cause QGIS to redirect all R graphical outputs to a temporary file, which will be opened once R execution has finished.

Both graphics and console results will be available through the processing results manager.

For more information, please check the R scripts in the official QGIS collection (you download and install them using the *QGIS Resource Sharing* plugin, as explained in *Adding R scripts from the QGIS collection*). Most of them are rather simple and will greatly help you understand how to create your own scripts.

Notă: The `sf`, `rgdal` and `raster` libraries are loaded by default, so you do not have to add the corresponding `library()` commands. However, other libraries that you might need have to be explicitly loaded by typing: `library(ggplot2)` (to load the `ggplot2` library). If the package is not already installed on your machine, Processing will try to download and install it. In this way the package will also become available in R Standalone. **Be aware** that if the package has to be downloaded, the script may take a long time to run the first time.

R libraries installed when running sf_test

The R script `sp_test` tries to load the R packages `sp` and `raster`.

The R script *sf_test* tries to load `sf` and `raster`. If these two packages are not installed, R may try to load and install them (and all the libraries that they depend on).

The following R libraries end up in `~/.local/share/QGIS/QGIS3/profiles/default/processing/rscripts` after `sf_test` has been run from the Processing Toolbox on Ubuntu with version 2.0 of the *Processing R Provider* plugin and a fresh install of *R* 3.4.4 (*apt* package `r-base-core` only):

```
abind, askpass, assertthat, backports, base64enc, BH, bit, bit64, blob, brew,␣
↪callr, classInt, cli, colorspace, covr, crayon, crosstalk, curl, DBI, deldir,
desc, dichromat, digest, dplyr, e1071, ellipsis, evaluate, fansi, farver, fastmap,␣
↪gdtools, ggplot2, glue, goftest, gridExtra, gtable, highr, hms,
htmltools, htmlwidgets, httpuv, httr, jsonlite, knitr, labeling, later, lazyeval,␣
↪leafem, leaflet, leaflet.providers, leafpop, leafsync, lifecycle, lwgeom,
magrittr, maps, mapview, markdown, memoise, microbenchmark, mime, munsell, odbc,␣
↪openssl, pillar, pkgbuild, pkgconfig, pkgload, plogr, plyr, png, polyclip,
praise, prettyunits, processx, promises, ps, purrr, R6, raster, RColorBrewer, Rcpp,
↪ reshape2, rex, rgeos, rlang, rmarkdown, RPostgres, RPostgreSQL,
rprojroot, RSQLite, rstudioapi, satellite, scales, sf, shiny, sourcetools, sp,␣
↪spatstat, spatstat.data, spatstat.utils, stars, stringi, stringr, svglite,
sys, systemfonts, tensor, testthat, tibble, tidyselect, tinytex, units, utf8, uuid,
↪ vctrs, viridis, viridisLite, webshot, withr, xfun, XML, xtable
```

GRASS

Configuring GRASS is not much different from configuring SAGA. First, the path to the GRASS folder has to be defined, but only if you are running Windows.

By default, the Processing framework tries to configure its GRASS connector to use the GRASS distribution that ships along with QGIS. This should work without problems for most systems, but if you experience problems, you might have to configure the GRASS connector manually. Also, if you want to use a different GRASS installation, you can change the setting to point to the folder where the other version is installed. GRASS 7 is needed for algorithms to work correctly.

If you are running Linux, you just have to make sure that GRASS is correctly installed, and that it can be run without problem from a terminal window.

GRASS algorithms use a region for calculations. This region can be defined manually using values similar to the ones found in the SAGA configuration, or automatically, taking the minimum extent that covers all the input layers used to execute the algorithm each time. If the latter approach is the behavior you prefer, just check the *Use min covering region* option in the GRASS configuration parameters.

LAStools

To use LAStools in QGIS, you need to download and install LAStools on your computer and install the LAStools plugin (available from the official repository) in QGIS.

On Linux platforms, you will need Wine to be able to run some of the tools.

LAStools is activated and configured in the Processing options (*Settings* ► *Options*, *Processing* tab, *Providers* ► *LAStools*), where you can specify the location of LAStools (*LAStools folder*) and Wine (*Wine folder*). On Ubuntu, the default Wine folder is `/usr/bin`.

OTB Applications

OTB applications are fully supported within the QGIS Processing framework.

OTB (Orfeo ToolBox) is an image processing library for remote sensing data. It also provides applications that provide image processing functionalities. The list of applications and their documentation are available in OTB CookBook

---

Notă: Note that OTB is not distributed with QGIS and needs to be installed separately. Binary packages for OTB can be found on the download page.

To configure QGIS processing to find the OTB library:

1. Open the processing settings: *Settings* ► *Options* ► *Processing*
2. You can see `OTB` under *Providers* menu:
    1. Expand the *OTB* entry
    2. Set the *OTB folder*. This is the location of your OTB installation.
    3. Set the *OTB application folder*. This is the location of your OTB applications (`<PATH_TO_OTB_INSTALLATION>/lib/otb/applications`)
    4. Click *OK* to save the settings and close the dialog.

If settings are correct, OTB algorithms will be available in the *Processing Toolbox*.

Documentation of OTB settings available in QGIS Processing

- **OTB folder**: This is the directory where OTB is available.
- **OTB application folder**: This is the location(s) of OTB applications.

  Multiple paths are allowed.
- **Logger level** (optional): Level of logger to use by OTB applications.

  The level of logging controls the amount of detail printed during algorithm execution. Possible values for logger level are `INFO`, `WARNING`, `CRITICAL`, `DEBUG`. This value is `INFO` by default. This is an advanced user configuration.
- **Maximum RAM to use** (optional): by default, OTB applications use all available system RAM.

  You can, however, instruct OTB to use a specific amount of RAM (in MB) using this option. A value of 256 is ignored by the OTB processing provider. This is an advanced user configuration.
- **Geoid file** (optional): Path to the geoid file.

  This option sets the value of the elev.dem.geoid and elev.geoid parameters in OTB applications. Setting this value globally enables users to share it across multiple processing algorithms. Empty by default.
- **SRTM tiles folder** (optional): Directory where SRTM tiles are available.

  SRTM data can be stored locally to avoid downloading of files during processing. This option sets the value of elev.dem.path and elev.dem parameters in OTB applications. Setting this value globally enables users to share it across multiple processing algorithms. Empty by default.

Compatibility and Troubleshoot

Starting from OTB 6.6.1, new releases of OTB are made compatible with at least the latest QGIS version available at that time.

If you have issues with OTB applications in QGIS Processing, please open an issue on the OTB bug tracker, using the `qgis` label.

Additional information about OTB and QGIS can be found in OTB Cookbook.

CAPITOLUL 28

Furnizorii de prelucrare și algoritmi

Algoritmii de procesare și parametrii acestora (așa cum sunt prezentați în interfața cu utilizatorul) sunt documentați aici.

28.1 Furnizorul algoritmului QGIS

Furnizorul algortimului QGIS implementează diverse operații de analiză și de geoprocesare utilizând, în general, doar QGIS API. Deci, aproape toți algoritmii acestui furnizor vor lucra „direct din cutie”, fără nici o configurare suplimentară.

Acest furnizor include câțiva algoritmi ai pluginurilor, la care se adaugă, de asemenea, și proprii algoritmi.

28.1.1 3D Tiles

`NEW in 3.34`

Convert B3DM to GLTF

Converts files from the legacy `.B3DM` format to `.GLTF` or `.GLB`.

Parameters

Label	Name	Type	Description
Input B3DM	`INPUT`	[3D Tile]	Input file to convert.
Output file	`OUTPUT`	[3D Tile] Default: `[Save to temporary file]`	Specify the output 3D tile file. One of: • Save to a Temporary File • Save to File…

Outputs

Label	Name	Type	Description
Output file	`OUTPUT`	[3D Tile]	Output layer in `.GLTF` format.

Python code

Algorithm ID: `native:b3dmtogltf`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Convert GLTF to vector features

Converts `.GLTF` or `.GLB` file contents to standard vector layer formats.

Parameters

Label	Name	Type	Description
Input GLTF	`INPUT`	[3D Tile]	Input file to convert.
Output polygons Optional	`OUTPUT_POLYGON`	[vector: polygon] Default: `[Create temporary layer]`	Specify the output vector layer to store polygon features. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Save to File… • Save to Geopackage… • Save to Database Table… The file encoding can also be changed here.
Output lines Optional	`OUTPUT_LINES`	[vector: line] Default: `[Create temporary layer]`	Specify the output vector layer to store line features. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Save to File… • Save to Geopackage… • Save to Database Table… The file encoding can also be changed here.

Outputs

Label	Name	Type	Description
Output polygons	`OUTPUT_POLYGON`	[vector: polygon]	A vector layer storing the polygon features in input layer.
Output lines	`OUTPUT_LINES`	[vector: line]	A vector layer storing the line features in input layer.

Python code

Algorithm ID: `native:gltftovector`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

28.1.2 Cartography

Align points to features

Calculates the rotation required to align point features with their nearest feature from another reference layer. A new field is added to the output layer which is filled with the angle (in degrees, clockwise) to the nearest reference feature.

Optionally, the output layer's symbology can be set to automatically use the calculated rotation field to rotate marker symbols. If desired, a maximum distance to use when aligning points can be set, to avoid aligning isolated points to distant features.

Sugestie: This algorithm is designed for use cases like aligning building point symbols to follow the nearest road direction.

Allows *features in-place modification* of point features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Point features to calculate the rotation for
Reference layer	`REFERENCE_LAYE`	[vector: oricare]	Layer to find the closest feature from for rotation calculation
Maximum distance to consider Opţional	`MAX_DISTANCE`	[număr] Implicit: Nesetat	If no reference feature is found within this distance, no rotation is assigned to the point feature.
Angle field name	`FIELD_NAME`	[string] Implicit: «rotation»	Field in which to store the rotation value.
Automatically apply symbology	`APPLY_SYMBOLOG`	[boolean] Implicit: True	Rotates the symbol marker of the features using the angle field value
Aligned layer	`OUTPUT`	[vector: punct] Implicit: `[Salvare într-un fișier temporar]`	Specify the rotated output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Aligned layer	`OUTPUT`	[vector: punct]	The point layer appended with a rotation field. If loaded to QGIS, it is applied by default the input layer symbology, with a data-defined rotation of its marker symbol.

Cod Python

ID Algorithm: `nativ:angletonearest`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Combine style databases

Combines multiple QGIS style databases into a single style database. If items of the same type with the same name exist in different source databases these will be renamed to have unique names in the output combined database.

Vezi și:

Create style database from project

Parametri

Etichetă	Nume	Tipul	Descriere
Input databases	`INPUT`	[file] [list]	Files containing QGIS style items
Objects to combine	`OBJECTS`	[enumeration] [list]	Types of style items in the input databases you would like to put in the new database. These can be: • 0 — *Simboluri* • 1 — *Rampe de culoare* • 2 — *Formate de text* • 3 — *Setări pentru etichete*
Output style database	`OUTPUT`	[file] Implicit: `[Salvare într-un fișier temporar]`	Output `.XML` file combining the selected style items. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Color ramp count	`COLORRAMPS`	[număr]	
Label settings count	`LABELSETTINGS`	[număr]	
Output style database	`OUTPUT`	[file]	Output `.XML` file combining the selected style items
Symbol count	`SYMBOLS`	[număr]	
Text format count	`TEXTFORMATS`	[număr]	

Cod Python

ID Algorithm: `nativ:combinestyles`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create categorized renderer from styles

Sets a vector layer's renderer to a categorized renderer using matching symbols from a style database. If no style file is specified, symbols from the user's current *symbol library* are used instead.

A specified expression or field is used to create categories for the renderer. Each category is individually matched to the symbols which exist within the specified QGIS XML style database. Whenever a matching symbol name is found, the category's symbol will be set to this matched symbol.

If desired, outputs can also be tables containing lists of the categories which could not be matched to symbols, and symbols which were not matched to categories.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer to apply a categorized style to
Categorize using expression	`FIELD`	[expresie]	Field or expression to categorize the features
Style database (leave blank to use saved symbols)	`STYLE`	[file]	File (`.XML`) containing the symbols to apply to the input layer categories. The file can be obtained from the Style Manager *Share symbols* tool. If no file is specified, QGIS local symbols library is used.
Use case-sensitive match to symbol names	`CASE_SENSITIVE`	[boolean] Implicit: *False*	If True (checked), applies a case sensitive comparison between the categories and symbols names
Ignore non-alphanumeric characters while matching	`TOLERANT`	[boolean] Implicit: *False*	If True (checked), non-alphanumeric characters in the categories and symbols names will be ignored, allowing greater tolerance during the match.
Non-matching categories Opţional	`NON_MATCHING_C`	[tabelă] Implicit: `[Omitere rezultat]`	Output table for categories which do not match any symbol in the database. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Non-matching symbol names Opţional	`NON_MATCHING_S`	[tabelă] Implicit: `[Omitere rezultat]`	Output table for symbols from the provided style database which do not match any category. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Non-matching categories	NON_MATCHING_C	[tabelă]	Lists categories which could not be matched to any symbol in the provided style database
Non-matching symbol names	NON_MATCHING_S	[tabelă]	Lists symbols from the provided style database which could not match any category
Categorized layer	OUTPUT	[la fel ca intrarea]	The input vector layer with the categorized style applied. No new layer is output.

Cod Python

ID Algorithm: `nativ:categorizeusingstyle`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create style database from project

Extracts all style objects (symbols, color ramps, text formats and label settings) from a QGIS project.

The extracted symbols are saved to a QGIS style database (XML format), which can be managed and imported via the *Style Manager* dialog.

Vezi și:

Combine style databases

Parametri

Etichetă	Nume	Tipul	Descriere
Input project (leave blank to use current) Opţional	`INPUT`	[file]	A QGIS project file to extract the style items from
Objects to extract	`OBJECTS`	[enumeration] [list]	Types of style items in the input project you would like to put in the new database. These can be: • 0 — *Simboluri* • 1 — *Rampe de culoare* • 2 — *Formate de text* • 3 — *Setări pentru etichete*
Output style database	`OUTPUT`	[file] Implicit: `[Salvare într-un fişier temporar]`	Specify the output `.XML` file for the selected style items. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Color ramp count	`COLORRAMPS`	[număr]	Number of color ramps
Label settings count	`LABELSETTINGS`	[număr]	Number of label settings
Output style database	`OUTPUT`	[file]	Output `.XML` file for the selected style items
Symbol count	`SYMBOLS`	[număr]	Number of symbols
Text format count	`TEXTFORMATS`	[număr]	Number of text formats

Cod Python

ID Algorithm: `nativ:stylefromproject`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export atlas layout as image

Exports the atlas of a print layout as image files (e.g. PNG or JPEG images).

If a coverage layer is set, the selected layout's atlas settings exposed in this algorithm will be overwritten. In this case, an empty filter or sort by expression will turn those settings off.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Atlas layout	`LAYOUT`	[layout]	Layout to export
Coverage layer Opţional	`COVERAGE_LAYER`	[vector: oricare]	Layer to use to generate the atlas
Filter expression	`FILTER_EXPRESS`	[expresie]	Expression to use to filter out atlas features
Sort expression Opţional	`SORTBY_EXPRESS`	[expresie]	Expression to use to sort the atlas features
Reverse sort order Opţional	`SORTBY_REVERSE`	[boolean]	Determines if sorting should be inverted. Used when a sort expression is provided.
Output filename expression	`FILENAME_EXPRE`	[expresie] Implicit: «output_»||@atlas_fe	Expression for use to generate filenames
Output folder	`FOLDER`	[folder]	Destination folder where the images will be generated

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Map layers to assign to unlocked map item(s) Opţional	`LAYERS`	[enumeration] [layer]	Layers to display in the map item(s) whose contents are not locked
Image format	`EXTENSION`	[enumerare] Implicit: png	File format of the generated output(s). The list of available formats varies depending on OS and installed drivers.
DPI Opţional	`DPI` Implicit: Nesetat	[număr]	DPI of the output file(s). If not set, the value in the print layout settings will be used.
Generate world file	`GEOREFERENCE`	[boolean] Implicit: True	Determines if a world file should be generated
Export RDF metadata	`INCLUDE_METADA`	[boolean] Implicit: True	Determines if RDF metadata (title, author, …) should be generated
Enable antialiasing	`ANTIALIAS`	[boolean] Implicit: True	Determines if antialiasing should be enabled

Rezultat

Etichetă	Nume	Tipul	Descriere
Image file	`OUTPUT`	[file]	Image files generated by the atlas layout

Cod Python

ID Algorithm: `nativ:atlaslayouttoimage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export atlas layout as PDF (multiple files)

Exports the atlas of a print layout to multiple PDF files.

If a coverage layer is set, the selected layout's atlas settings exposed in this algorithm will be overwritten. In this case, an empty filter or sort by expression will turn those settings off.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Atlas layout	`LAYOUT`	[layout]	Layout to export
Coverage layer Opţional	`COVERAGE_LAYER`	[vector: oricare]	Layer to use to generate the atlas
Filter expression	`FILTER_EXPRESS`	[expresie]	Expression to use to filter out atlas features
Sort expression Opţional	`SORTBY_EXPRESS`	[expresie]	Expression to use to sort the atlas features
Reverse sort order Opţional	`SORTBY_REVERSE`	[boolean]	Determines if sorting should be inverted. Used when a sort expression is provided.
Output filename Opţional	`OUTPUT_FILENAM`	[expresie]	Name pattern of the PDF output files.
Output folder	`OUTPUT_FOLDER`	[folder]	Destination folder for the output PDF files.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Map layers to assign to unlocked map item(s) Opţional	`LAYERS`	[enumeration] [layer]	Layers to display in the map item(s) whose contents are not locked
DPI Opţional	`DPI` Implicit: Nesetat	[număr]	DPI of the output file(s). If not set, the value in the print layout settings will be used.
Always export as vectors	`FORCE_VECTOR`	[boolean] Implicit: *False*	Determines if vectorial data should be left as vectors
Always export as raster	`FORCE_RASTER`	[boolean] Implicit: *False*	Forces all the items in the map to be rasterized. This parameter takes precedence over the `FORCE_VECTOR` parameter.
Append georeference information	`GEOREFERENCE`	[boolean] Implicit: True	Determines if a world file should be generated
Export RDF metadata	`INCLUDE_METADA`	[boolean] Implicit: True	Determines if RDF metadata (title, author, …) should be generated
Disable tiled raster layer exports	`DISABLE_TILED`	[boolean] Implicit: *False*	Determines if raster should be tiled
Simplify geometries to reduce output file size	`SIMPLIFY`	[boolean] Implicit: True	Determines if geometries should be simplified to reduce output file size
Text export	`TEXT_FORMAT`	[enumerare] Implicit: 0	Determines if text should be exported as path or text objects. Possible options are: • 0 - Always export text as paths (recommended) • 1 - Always export texts as text objects
Image compression	`IMAGE_COMPRESS`	[enumerare] Implicit: 0	Determines compression level of the image and how suitable the file could be for printing outputs or post-production in external applications. Possible options are: • 0 - Lossy (JPEG) • 1 - Lossless

Rezultat

Etichetă	Nume	Tipul	Descriere
PDF file	`OUTPUT`	[file]	PDF file corresponding to the exported atlas layout

Cod Python

Algorithm ID: `native:atlaslayouttomultiplepdf`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export atlas layout as PDF (single file)

Exports the atlas of a print layout as a single PDF file.

If a coverage layer is set, the selected layout's atlas settings exposed in this algorithm will be overwritten. In this case, an empty filter or sort by expression will turn those settings off.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Atlas layout	`LAYOUT`	[layout]	Layout to export
Coverage layer Opţional	`COVERAGE_LAYER`	[vector: oricare]	Layer to use to generate the atlas
Filter expression	`FILTER_EXPRESS`	[expresie]	Expression to use to filter out atlas features
Sort expression Opţional	`SORTBY_EXPRESS`	[expresie]	Expression to use to sort the atlas features
Reverse sort order Opţional	`SORTBY_REVERSE`	[boolean]	Determines if sorting should be inverted. Used when a sort expression is provided.
PDF file	`OUTPUT`	[file] Implicit: [Salvare într-un fișier temporar]	Name (including path) of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Map layers to assign to unlocked map item(s) Opţional	`LAYERS`	[enumeration] [layer]	Layers to display in the map item(s) whose contents are not locked
DPI Opţional	`DPI` Implicit: Nesetat	[număr]	DPI of the output file(s). If not set, the value in the print layout settings will be used.
Always export as vectors	`FORCE_VECTOR`	[boolean] Implicit: *False*	Determines if vectorial data should be left as vectors
Always export as raster	`FORCE_RASTER`	[boolean] Implicit: *False*	Forces all the items in the map to be rasterized. This parameter takes precedence over the `FORCE_VECTOR` parameter.
Append georeference information	`GEOREFERENCE`	[boolean] Implicit: True	Determines if a world file should be generated
Export RDF metadata	`INCLUDE_METADA`	[boolean] Implicit: True	Determines if RDF metadata (title, author, …) should be generated
Disable tiled raster layer exports	`DISABLE_TILED`	[boolean] Implicit: *False*	Determines if raster should be tiled
Simplify geometries to reduce output file size	`SIMPLIFY`	[boolean] Implicit: True	Determines if geometries should be simplified to reduce output file size
Text export	`TEXT_FORMAT`	[enumerare] Implicit: 0	Determines if text should be exported as path or text objects. Possible options are: • 0 - Always export text as paths (recommended) • 1 - Always export texts as text objects
Image compression	`IMAGE_COMPRESS`	[enumerare] Implicit: 0	Determines compression level of the image and how suitable the file could be for printing outputs or post-production in external applications. Possible options are: • 0 - Lossy (JPEG) • 1 - Lossless

Rezultat

Etichetă	Nume	Tipul	Descriere
PDF file	`OUTPUT`	[file]	PDF file corresponding to the exported atlas layout

Cod Python

ID Algorithm: `nativ:atlaslayouttopdf`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export print layout as image

Exports a print layout as an image file (e.g. PNG or JPEG images)

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Print layout	`LAYOUT`	[layout]	Layout to export
Image file	`OUTPUT`	[file] Implicit: [Salvare într-un fișier temporar]	Name (including path) of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Map layers to assign to unlocked map item(s) Opţional	`LAYERS`	[enumeration] [layer]	Layers to display in the map item(s) whose contents are not locked
DPI Opţional	`DPI` Implicit: Nesetat	[număr]	DPI of the output file(s). If not set, the value in the print layout settings will be used.
Generate world file	`GEOREFERENCE`	[boolean] Implicit: True	Determines if a world file should be generated
Export RDF metadata	`INCLUDE_METADA`	[boolean] Implicit: True	Determines if RDF metadata (title, author, …) should be generated
Enable antialiasing	`ANTIALIAS`	[boolean] Implicit: True	Determines if antialiasing should be enabled

Rezultat

Etichetă	Nume	Tipul	Descriere
Image file	`OUTPUT`	[file]	Image file corresponding to the exported print layout

Cod Python

ID Algorithm: `nativ:printlayouttoimage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export print layout as PDF

Exports a print layout as a PDF file.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Print Layout	`LAYOUT`	[layout]	Layout to export
PDF file	`OUTPUT`	[file] Implicit: [Salvare într-un fișier temporar]	Name (including path) of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Map layers to assign to unlocked map item(s) Opţional	LAYERS	[enumeration] [layer]	Layers to display in the map item(s) whose contents are not locked
DPI Opţional	DPI Implicit: Nesetat	[număr]	DPI of the output file(s). If not set, the value in the print layout settings will be used.
Always export as vectors	FORCE_VECTOR	[boolean] Implicit: *False*	Determines if vectorial data should be left as vectors
Always export as raster	FORCE_RASTER	[boolean] Implicit: *False*	Forces all the items in the map to be rasterized. This parameter takes precedence over the FORCE_VECTOR parameter.
Append georeference information	GEOREFERENCE	[boolean] Implicit: True	Determines if a world file should be generated
Export RDF metadata	INCLUDE_METADA	[boolean] Implicit: True	Determines if RDF metadata (title, author, …) should be generated
Disable tiled raster layer exports	DISABLE_TILED	[boolean] Implicit: *False*	Determines if raster should be tiled
Simplify geometries to reduce output file size	SIMPLIFY	[boolean] Implicit: True	Determines if geometries should be simplified to reduce output file size
Text export	TEXT_FORMAT	[enumerare] Implicit: 0	Determines if text should be exported as path or text objects. Possible options are: • 0 - Always export text as paths (recommended) • 1 - Always export texts as text objects
Image compression	IMAGE_COMPRESS	[enumerare] Implicit: 0	Determines compression level of the image and how suitable the file could be for printing outputs or post-production in external applications. Possible options are: • 0 - Lossy (JPEG) • 1 - Lossless
Export layers as separate PDF files	SEPARATE_LAYER	[boolean] Implicit: *False*	If True, then a separate PDF file will be created per layer per map item in the layout. Additionally, separate PDF files may be created for other complex layout items, resulting in a set of PDF files which contain logical atomic components of the layout.

Rezultat

Etichetă	Nume	Tipul	Descriere
PDF file	`OUTPUT`	[file]	PDF file(s) corresponding to the exported print layout

Cod Python

ID Algorithm: `nativ:printlayouttopdf`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract labels

Extracts label information from a rendered map at a given extent and scale.

If a map theme is provided, the rendered map will match the visibility and symbology of that theme. If left blank, all visible layers from the project will be used. Extracted label information include: position (served as point geometries), the associated layer name and feature ID, label text, rotation (in degree, clockwise), multiline alignment, and font details.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Map extent	`EXTENT`	[extent]	Extent of the map to extract the labels from Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Map scale	`SCALE`	[scale]	Extracted labels will be rendered using their properties set at this scale.
Map theme Opţional	`MAP_THEME`	[maptheme]	A map theme displaying the layers to extract the labels from. If unset, labels of the currently visible layers are extracted.
Include unplaced labels Opţional	`INCLUDE_UNPLAC`	[boolean] Implicit: True	Specify whether all overlapping labels should be extracted, including the conflicting (thus unplaced) ones.
Extracted labels	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the extent(s). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Map resolution (in DPI) Opţional	`DPI` Implicit: 96.0	[număr]	

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted labels	`OUTPUT`	[vector: punct]	Point vector layer representing the fetched labels. Each feature has attributes identifying its source (layer, feature ID) and the assigned labeling properties (text, font, size, rotation, …). A default style with labeling and null symbol is also applied to the layer. **Atenționare:** Because some of the generated fields have name with more than 10 characters, using the ESRI shapefile format (`.SHP`) to store the output may lead to unexpected rendering while loading the layer in QGIS.

Cod Python

Algorithm ID: `native:extractlabels`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Print layout map extent to layer

Creates a polygon layer containing the extent of a print layout map item (or items), with attributes specifying the map size (in layout units, i.e. the *reference map* units), scale and rotation.

If the map item parameter is specified, then only the matching map extent will be exported. If it is not specified, all map extents from the layout will be exported.

Optionally, a specific output CRS can be specified. If it is not specified, the original map item CRS will be used.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Print layout	`LAYOUT`	[enumerare]	A print layout in the current project
Map item Opţional	`MAP`	[enumerare] Implicit: *Toate elementele hărții*	The map item(s) whose information you want to extract. If none is provided then all the map items are processed.
Extindere	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the extent(s). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Overrride CRS Opţional	`CRS`	[crs] Implicit: *CRS-ul Stratului*	Select the CRS for the layer in which the information will be reported.

Rezultat

Etichetă	Nume	Tipul	Descriere
Map height	HEIGHT	[număr]	
Extindere	OUTPUT	[vector: poligon]	Output polygon vector layer containing extents of all the input layout map item(s)
Map rotation	ROTATION	[număr]	
Map scale	SCALE	[număr]	
Map width	WIDTH	[număr]	

Cod Python

ID Algorithm: `nativ:printlayoutmapextenttolayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Set layer style

Applies a provided style to a layer. The style must be defined in a QML file.

No new output are created: the style is immediately assigned to the layer.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	INPUT	[strat]	Input layer you want to apply the style to
Style file	STYLE	[file]	Path to the `.qml` file of the style

Rezultat

Etichetă	Nume	Tipul	Descriere
	OUTPUT	[la fel ca intrarea]	The input layer with the new style assigned. No new layer is created.

Cod Python

ID Algorithm: `nativ:setlayerstyle`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Topological coloring

Assigns a color index to polygon features in such a way that no adjacent polygons share the same color index, whilst minimizing the number of colors required.

The algorithm allows choice of method to use when assigning colors.

A minimum number of colors can be specified if desired. The color index is saved to a new attribute named **color_id**.

The following example shows the algorithm with four different colors chosen; as you can see each color class has the same amount of features.

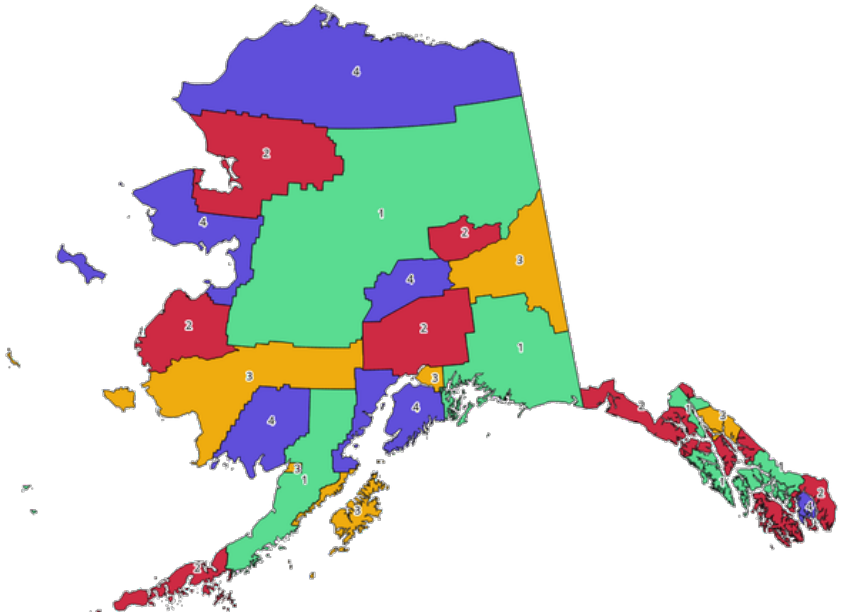


Fig. 28.1: Topological colors example

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	The input polygon layer
Minimum number of colors	`MIN_COLORS`	[număr] Implicit: 4	The minimum number of colors to assign. Minimum 1, maximum 1000.
Minimum distance between features	`MIN_DISTANCE`	[număr] Implicit: 0.0	Prevent nearby (but non-touching) features from being assigned equal colors. Minimum 0.0.

continues on next page

Tabelul 28.4 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Balance color assignment	`BALANCE`	[enumerare] Implicit: 0	Options are: • 0 — După numărul de entități Attempts to assign colors so that the count of features assigned to each individual color index is balanced. • 1 — După zona alocată Assigns colors so that the total area of features assigned to each color is balanced. This mode can be useful to help avoid large features resulting in one of the colors appearing more dominant on a colored map. • 2 — După distanța dintre culori Assigns colors in order to maximize the distance between features of the same color. This mode helps to create a more uniform distribution of colors across a map.
Colored	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Colored	`OUTPUT`	[vector: poligon]	Polygon vector layer with an added `color_id` column

Cod Python

Algorithm ID: `qgis:topologicalcoloring`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Transfer annotations from main layer

Transfers all *annotations* from the main annotation layer in a project to a new annotation layer. Items placement can then be adjusted within the layer stack.

Parametri

Etichetă	Nume	Tipul	Descriere
New layer name	`LAYER_NAME`	[string] Implicit: «Adnotări»	Name of the annotations layer to create

Rezultat

Etichetă	Nume	Tipul	Descriere
New layer name	`OUTPUT`	[strat]	A layer with items from the main annotation layer

Cod Python

Algorithm ID: `native:transferannotationsfrommain`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.3 Database

Export to PostgreSQL

Exports a vector layer to a PostgreSQL database, creating a new relation. If a relation with the same name exists, it can be removed before the new relation is created. Prior to this a connection between QGIS and the PostgreSQL database has to be created (see eg *Creating a stored Connection*).

Parameters

Label	Name	Type	Description
Layer to import	`INPUT`	[vector: any]	Vector layer to add to the database
Database (connection name)	`DATABASE`	[string]	Name of the database connection (not the database name). Existing connections will be shown in the combobox.

continues on next page

Tabelul 28.6 – continuare din pagina precedentă

Label	Name	Type	Description
Schema (schema name) Optional	SCHEMA	[string] Default: «public»	Name of the schema to store the data. It can be a new one or already exist.
Table to import to (leave blank to use layer name) Optional	TABLENAME	[string] Default: «»	Defines a table name for the imported vector file. If nothing is added, the layer name will be used.
Primary key field Optional	PRIMARY_KEY	[tablefield: any]	Sets the primary key field from an existing field in the vector layer. A column with **unique** values can be used as Primary key for the database.
Geometry column	GEOMETRY_COLUM	[string] Default: «geom»	Defines the name of the geometry column in the new PostGIS table. Geometry information for the features is stored in this column.
Encoding Optional	ENCODING	[string] Default: «UTF-8»	Defines the encoding of the output layer
Overwrite	OVERWRITE	[boolean] Default: True	If the specified table exists, setting this option to `True` will make sure that it is deleted and a new table will be created before the features are added. If this option is `False` and the table exists, the algorithm will throw an exception („relation already exists").
Create spatial index	CREATEINDEX	[boolean] Default: True	Specifies whether to create a spatial index or not
Convert field names to lowercase	LOWERCASE_NAME	[boolean] Default: True	Converts the field names of the input vector layer to lowercase
Drop length constraint on character fields	DROP_STRING_LE	[boolean] Default: False	Should length constraints on character fields be dropped or not
Create single-part geometries instead of multi-part	FORCE_SINGLEPA	[boolean] Default: False	Should the features of the output layer be single-part instead of multi-part. By default the existing geometries information are preserved.

Outputs

The algorithm has no output.

Python code

Algorithm ID: `qgis:importintopostgis`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Export to SpatiaLite

Exports a vector layer to a SpatiaLite database. Prior to this a connection between QGIS and the SpatiaLite database has to be created (see eg *Straturile SpatiaLite*).

Parameters

Label	Name	Type	Description
Layer to import	`INPUT`	[vector: any]	Vector layer to add to the database
File database	`DATABASE`	[vector: any]	The SQLite/SpatiaLite database file to connect to
Table to import to (leave blank to use layer name) Optional	`TABLENAME`	[string] Default: «»	Defines the table name for the imported vector file. If nothing is specified, the layer name will be used.
Primary key field Optional	`PRIMARY_KEY`	[tablefield: any]	Use a field in the input vector layer as the primary key
Geometry column	`GEOMETRY_COLUM`	[string] Default: «geom»	Defines the name of the geometry column in the new SpatiaLite table. Geometry information for the features is stored in this column.
Encoding Optional	`ENCODING`	[string] Default: «UTF-8»	Defines the encoding of the output layer
Overwrite	`OVERWRITE`	[boolean] Default: True	If the specified table exists, setting this option to `True` will make sure that it is deleted and a new table will be created before the features of the layer is added. If this option is `False` and the table exists, the algorithm will throw an exception („table already exists”).
Create spatial index	`CREATEINDEX`	[boolean] Default: True	Specifies whether to create a spatial index or not
Convert field names to lowercase	`LOWERCASE_NAME`	[boolean] Default: True	Convert the field names of the input vector layer to lowercase
Drop length constraint on character fields	`DROP_STRING_LE`	[boolean] Default: False	Should length constraints on character fields be dropped or not
Create single-part geometries instead of multi-part	`FORCE_SINGLEPA`	[boolean] Default: False	Should the features of the output layer be single-part instead of multi-part. By default the existing geometries information are preserved.

Outputs

The algorithm has no output.

Python code

Algorithm ID: `qgis:importintospatialite`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Package layers

Adds layers to a GeoPackage.

If the GeoPackage exists and `Overwrite existing GeoPackage` is checked, it will be overwritten (removed and recreated). If the GeoPackage exists and `Overwrite existing GeoPackage` is not checked, the layer will be appended.

Parameters

Label	Name	Type	Description
Input layers	`LAYERS`	[vector: any] [list]	The (vector) layers to import into the GeoPackage. Raster layers are not supported. If a raster layer is added, a `QgsProcessingException` will be thrown.
Overwrite existing GeoPackage	`OVERWRITE`	[boolean] Default: False	If the specified GeoPackage exists, setting this option to `True` will make sure that it is deleted and a new one will be created before the layers are added. If set to `False`, layers will be appended.
Save layer styles into GeoPackage	`SAVE_STYLES`	[boolean] Default: True	Save the layer styles
Save only selected features	`SELECTED_FEATU`	[boolean] Default: False	If a layer has a selection, setting this option to `True` will result in only selected features being saved. For layers without a selection all features will be saved.
Export related layers following relations defined in the project	`EXPORT_RELATED`	[boolean] Default: False	If an input layer has *relations* set in the project, setting this option to `True` will result in exporting also its related layer(s). If the layer has features selected, then only their related features will be exported unless the related layer was also an input layer.
Destination GeoPackage	`OUTPUT`	[file] Default: `[Save to temporary file]`	Specify where to store the GeoPackage file. One of • Save to a Temporary File • Save to File…

Outputs

Label	Name	Type	Description
Layers within new package	`OUTPUT_LAYERS`	[string] [list]	The list of layers added to the GeoPackage.

Python code

Algorithm ID: `native:package`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

PostgreSQL execute and load SQL

Allows a SQL database query to be performed on a PostgreSQL database connected to QGIS and loads the result. The algorithm **won't** create a new layer: it is designed to run queries on the layer itself.

Example

1. Set all the values of an existing field to a fixed value. The SQL query string will be:

   ```
   UPDATE your_table SET field_to_update=20;
   ```

   In the example above, the values of the field `field_to_update` of the table `your_table` will be all set to `20`.

2. Create a new `area` column and calculate the area of each feature with the `ST_AREA` PostGIS function.

   ```
   -- Create the new column "area" on the table your_table"
   ALTER TABLE your_table ADD COLUMN area double precision;
   -- Update the "area" column and calculate the area of each feature:
   UPDATE your_table SET area=ST_AREA(geom);
   ```

Vezi și:

PostgreSQL execute SQL, *Executare SQL*, *SpatiaLite execute SQL*

Parameters

Label	Name	Type	Description
Database (connection name)	`DATABASE`	[string]	The database connection (not the database name). Existing connections will be shown in the combobox.
SQL query	`SQL`	[string]	Defines the SQL query, for example `'UPDATE my_table SET field=10'`.
Unique ID field name	`ID_FIELD`	[string] Default: id	Sets the primary key field (a column in the result table)
Geometry field name Optional	`GEOMETRY_FIELD`	[string] Default: «geom»	Name of the geometry column (a column in the result table)

Outputs

Label	Name	Type	Description
SQL layer	`OUTPUT`	[vector: any]	The resulting vector layer to be loaded into QGIS.

Python code

Algorithm ID: `qgis:postgisexecuteandloadsql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

PostgreSQL execute SQL

Allows a SQL database query to be performed on a PostgreSQL database connected to QGIS. The algorithm **won't** create a new layer: it is designed to run queries on the layer itself.

Example

1. Set all the values of an existing field to a fixed value. The SQL query string will be:

   ```
   UPDATE your_table SET field_to_update=20;
   ```

   In the example above, the values of the field `field_to_update` of the table `your_table` will be all set to `20`.

2. Create a new `area` column and calculate the area of each feature with the `ST_AREA` PostGIS function.

   ```
   -- Create the new column "area" on the table your_table"
   ALTER TABLE your_table ADD COLUMN area double precision;
   -- Update the "area" column and calculate the area of each feature:
   UPDATE your_table SET area=ST_AREA(geom);
   ```

Vezi și:

PostgreSQL execute and load SQL, *Executare SQL*, *SpatiaLite execute SQL*

Parameters

Label	Name	Type	Description
Database (connection name)	`DATABASE`	[string]	The database connection (not the database name). Existing connections will be shown in the combobox.
SQL query	`SQL`	[string]	Defines the SQL query, for example `'UPDATE my_table SET field=10'`.

Outputs

No output is created. The SQL query is executed in place.

Python code

Algorithm ID: `native:postgisexecutesql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

SpatiaLite execute SQL

Allows a SQL database query to be performed on a SpatiaLite database. The algorithm **won't** create a new layer: it is designed to run queries on the layer itself.

Vezi și:

PostgreSQL execute SQL, *Executare SQL*

For some SQL query examples see *PostGIS SQL Query Examples*.

Parameters

Label	Name	Type	Description
File Database	`DATABASE`	[vector]	The SQLite/SpatiaLite database file to connect to
SQL query	`SQL`	[string] Default: «»	Defines the SQL query, for example `'UPDATE my_table SET field=10'`.

Outputs

No output is created. The SQL query is executed in place.

Python code

Algorithm ID: `native:spatialiteexecutesql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

SpatiaLite execute SQL (registered DB)

Allows a SQL database query to be performed on a SpatiaLite database connected to QGIS. The algorithm **won't** create a new layer: it is designed to run queries on the layer itself.

Vezi și:

PostgreSQL execute SQL, *Executare SQL*

For some SQL query examples see *PostGIS SQL Query Examples*.

Parameters

Label	Name	Type	Description
Database	`DATABASE`	[enumeration] Default: not set	Select a SQLite/SpatiaLite database connected to the current session
SQL query	`SQL`	[string] Default: «»	Defines the SQL query, for example `'UPDATE my_table SET field=10'`.

Outputs

No output is created. The SQL query is executed in place.

Python code

Algorithm ID: `native:spatialiteexecutesqlregistered`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

28.1.4 File tools

Download file

Downloads a file specified using a URL (using for instance `http:` or `file:`). In other words you can copy/paste a URL and download the file.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
URL	`URL`	[string]	The URL of the file to download.
File destination Opţional	`OUTPUT`	[string] Implicit: `[Salvare într-un fişier temporar]`	Specification of the file destination. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Method	`METHOD`	[enumerare] Implicit: 0	The HTTP method to use for the request. Options are: • 0 — GET • 1 — POST
Data Opţional	`DATA`	[string]	The data to add in the body if the request is a POST.

Rezultat

Etichetă	Nume	Tipul	Descriere
File destination	`OUTPUT`	[string]	The location of the downloaded file

Cod Python

Algorithm ID: `qgis:filedownloader`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.5 GPS

Convert GPS data

Uses the GPSBabel tool to convert a GPS data file from a range of formats to the GPX standard format.

Parametri

Etichetă	Nume	Tipul	Descriere
Fişier de intrare	`INPUT`	[file]	File containing the data to convert
Format	`FORMAT`	[enumerare]	Format of the file to convert, from this list.
Feature type	`FEATURE_TYPE`	[enumerare] Implicit: 0	The type of data to convert • 0 — Puncte de interes • 1 — Rute • 2 — Piste
Rezultat	`OUTPUT`	[vector: oricare] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output GPX file. One of: • Salvare într-un Fişier Temporar • Salvare în Fişier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT_LAYER`	[vector: oricare]	Output layer with data in GPX standard format

Cod Python

Algorithm ID: `native:convertgpsdata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afişează atunci când treceţi peste algoritmul din caseta instrumentelor de procesare. *Dicţionarul parametrilor* furnizează valorile şi NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeţi *Utilizarea algoritmilor de procesare din consolă*.

Convert GPX feature type

Uses the GPSBabel tool to convert GPX features from one type to another (e.g. converting all waypoint features to a route).

Parametri

Etichetă	Nume	Tipul	Descriere
Fișier de intrare	`INPUT`	[file]	File containing the data to convert
Conversion	`CONVERSION`	[enumerare] Implicit: 0	The type of conversion to apply • 0 — Punctele de interes dintr-o rută • 1 — Punctele de interes de pe un traseu • 2 — Rutele, pornind de la punctele de interes • 3 — Traseele, pornind de la punctele de interes
Rezultat	`OUTPUT`	[vector: point or line] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Rezultat	`OUTPUT`	[vector: oricare]	Output layer with converted GPX features

Cod Python

Algorithm ID: `native:convertgpxfeaturetype`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Download GPS data from device

Uses the GPSBabel tool to download data from a GPS device into the GPX standard format.

Parametri

Etichetă	Nume	Tipul	Descriere
Device	`DEVICE`	[enumerare] Implicit:*Garmin serial*	The GPS device used to create the data. Must be declared in *GPS Settings* dialog.
Port	`PORT`	[enumerare]	The port the device is connected to. Available ports depend on the OS.
Feature type	`FEATURE_TYPE`	[enumerare] Implicit: 0	The type of data to convert • 0 — Puncte de interes • 1 — Rute • 2 — Piste
Rezultat	`OUTPUT`	[vector: oricare] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT_LAYER`	[vector: oricare]	Output layer with data in GPX standard format

Cod Python

Algorithm ID: `native:downloadgpsdata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Upload GPS data to device

Uses the GPSBabel tool to upload data to a GPS device from the GPX standard format.

Parametri

Etichetă	Nume	Tipul	Descriere
Fişier de intrare	`INPUT`	[file]	`.GPX` file containing the data to upload
Device	`DEVICE`	[enumerare] Implicit:*Garmin serial*	The GPS device you would like to upload the data to. Must be declared in *GPS Settings* dialog.
Port	`PORT`	[enumerare]	The port the device is connected to. Available ports depend on the OS.
Feature type	`FEATURE_TYPE`	[enumerare] Implicit: 0	The type of data to upload • 0 — Puncte de interes • 1 — Rute • 2 — Piste

Rezultat

No output is provided. If successful, data are loaded to the device.

Cod Python

Algorithm ID: `native:uploadgpsdata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.6 Interpolarea

Heatmap (kernel density estimation)

Creates a density (heatmap) raster of an input point vector layer using kernel density estimation.

The density is calculated based on the number of points in a location, with larger numbers of clustered points resulting in larger values. Heatmaps allow easy identification of *hotspots* and clustering of points.

Parametri

Etichetă	Nume	Tipul	Descriere
Point layer	`INPUT`	[vector: punct]	Point vector layer to use for the heatmap
Radius	`RADIUS`	[număr] Implicit: 100.0	Heatmap search radius (or kernel bandwidth) in map units. The radius specifies the distance around a point at which the influence of the point will be felt. Larger values result in greater smoothing, but smaller values may show finer details and variation in point density.
Output raster size	`PIXEL_SIZE`	[număr] Implicit: 0.1	Pixel size of the output raster layer in layer units. In the GUI, the size can be specified by the number of rows (`Number of rows`) / columns (`Number of columns`) **or** the pixel size(`Pixel Size X` / `Pixel Size Y`). Increasing the number of rows or columns will decrease the cell size and increase the file size of the output raster. The values in `Rows`, `Columns`, `Pixel Size X` and `Pixel Size Y` will be updated simultaneously - doubling the number of rows will double the number of columns, and the cell size will be halved. The extent of the output raster will remain the same (approximately).
Radius from field Opţional	`RADIUS_FIELD`	[tablefield: numeric]	Sets the search radius for each feature from an attribute field in the input layer.
Weight from field Opţional	`WEIGHT_FIELD`	[tablefield: numeric]	Allows input features to be weighted by an attribute field. This can be used to increase the influence certain features have on the resultant heatmap.
Kernel shape	`KERNEL`	[enumerare] Implicit: *0*	Controls the rate at which the influence of a point decreases as the distance from the point increases. Different kernels decay at different rates, so a triweight kernel gives features greater weight for distances closer to the point then the Epanechnikov kernel does. Consequently, triweight results in "sharper" hotspots and Epanechnikov results in "smoother" hotspots. There are many shapes available (please see the Wikipedia page for further information): • 0 — Quartic • 1 — Triunghiular • 2 — Uniform • 3 — Triweight • 4 — Epanechnikov

continues on next page

Tabelul 28.15 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Decay ratio (Triangular kernels only) Opţional	`DECAY`	[număr] Implicit: *0.0*	Can be used with Triangular kernels to further control how heat from a feature decreases with distance from the feature. • A value of 0 (=minimum) indicates that the heat will be concentrated in the center of the given radius and completely extinguished at the edge. • A value of 0.5 indicates that pixels at the edge of the radius will be given half the heat as pixels at the center of the search radius. • A value of 1 means the heat is spread evenly over the whole search radius circle. (This is equivalent to the 'Uniform' kernel.) • A value greater than 1 indicates that the heat is higher towards the edge of the search radius than at the center.
Output value scaling	`OUTPUT_VALUE`	[enumerare] Implicit: *Raw*	Allow to change the values of the output heatmap raster. One of: • 0 — Brut • 1 — Scalat
Heatmap	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer with kernel density values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Heatmap	`OUTPUT`	[raster]	Raster layer with kernel density values

Example: Creating a Heatmap

For the following example, we will use the `airports` vector point layer from the QGIS sample dataset (see *Descărcarea datelor demonstrative*). Another excellent QGIS tutorial on making heatmaps can be found at http://qgistutorials.com.

In Fig. 28.2, the airports of Alaska are shown.

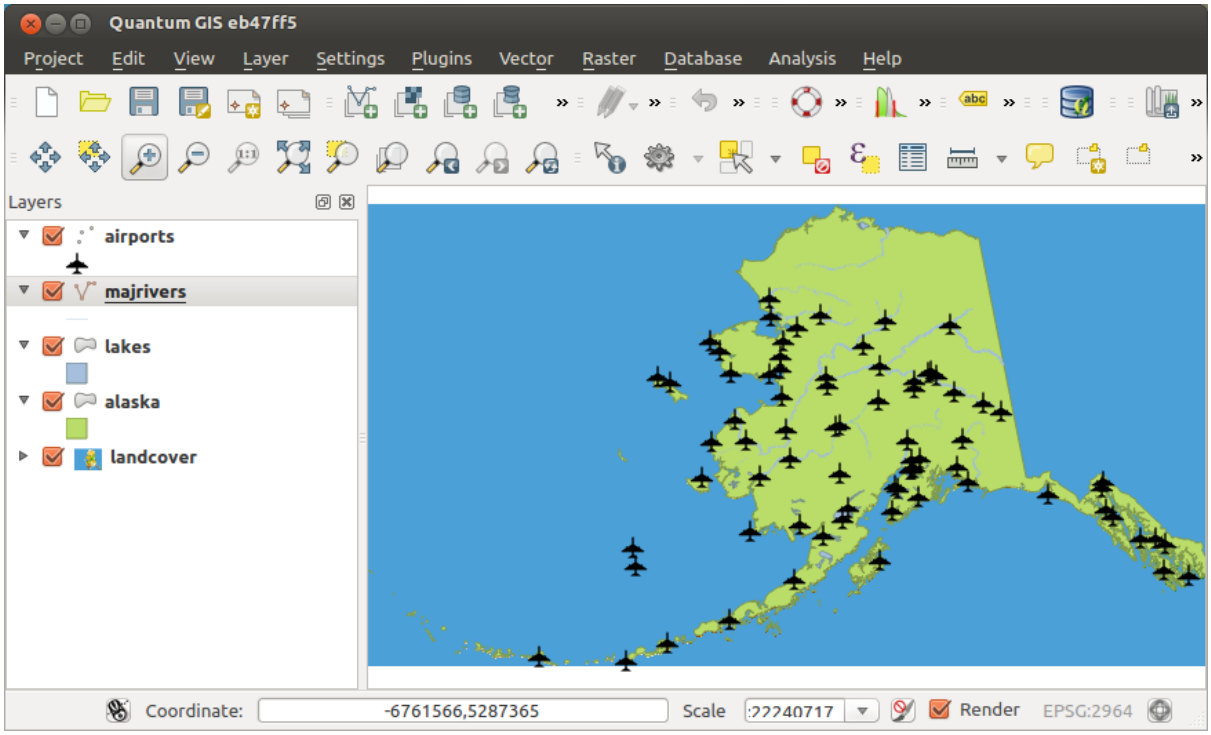


Fig. 28.2: Airports of Alaska

1. Open the *Heatmap (Kernel Density Estimation)* algorithm from the QGIS *Interpolation* group
2. In the *Point layer* field, select `airports` from the list of point layers loaded in the current project.
3. Change the *Radius* to `1000000` meters.
4. Change the *Pixel size X* to `1000`. The *Pixel size Y*, *Rows* and *Columns* will be automatically updated.
5. Click on *Run* to create and load the airports heatmap (see Fig. 28.4).

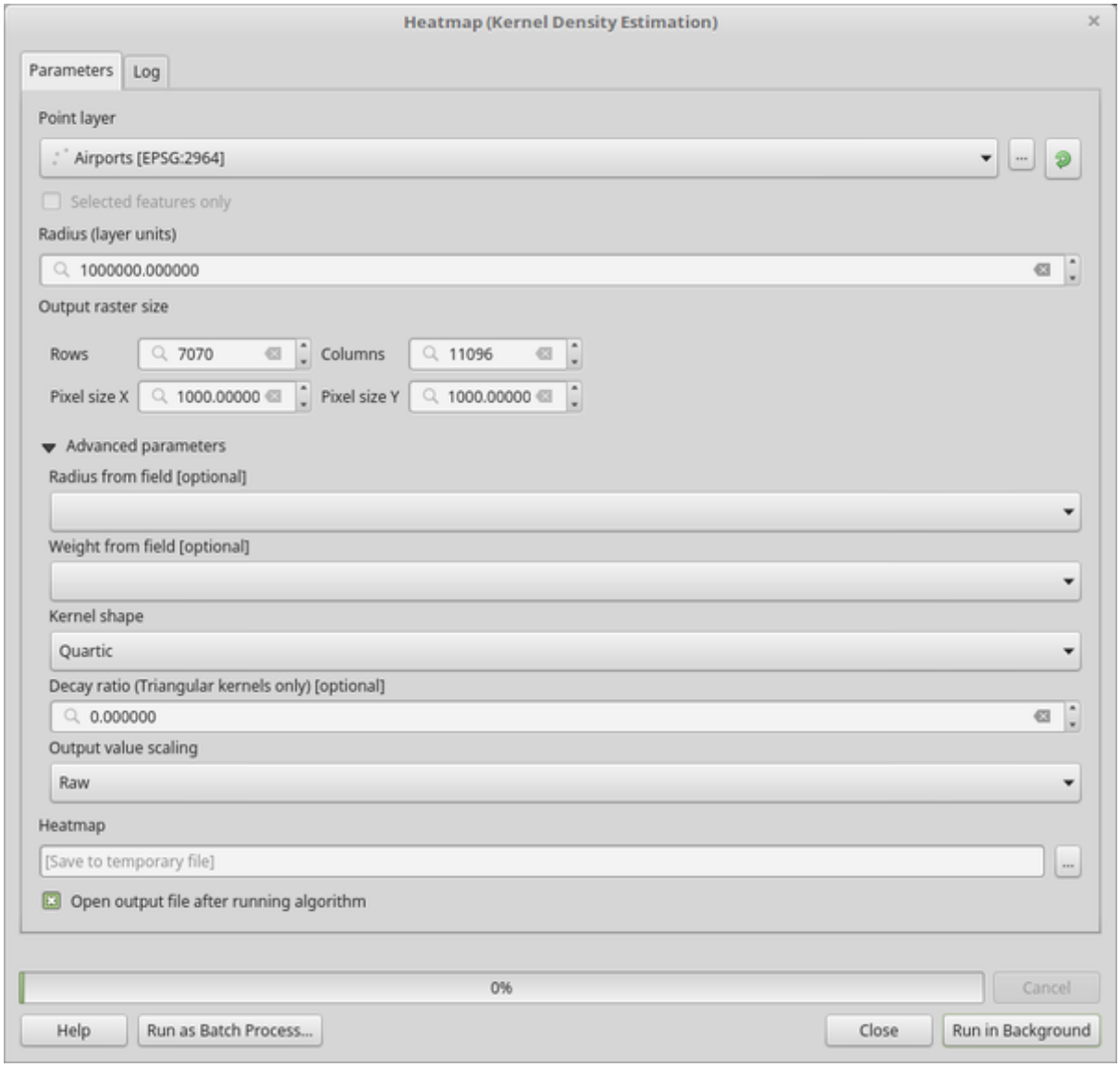


Fig. 28.3: The Heatmap Dialog

QGIS will generate the heatmap and add it to your map window. By default, the heatmap is shaded in greyscale, with lighter areas showing higher concentrations of airports. The heatmap can now be styled in QGIS to improve its appearance.

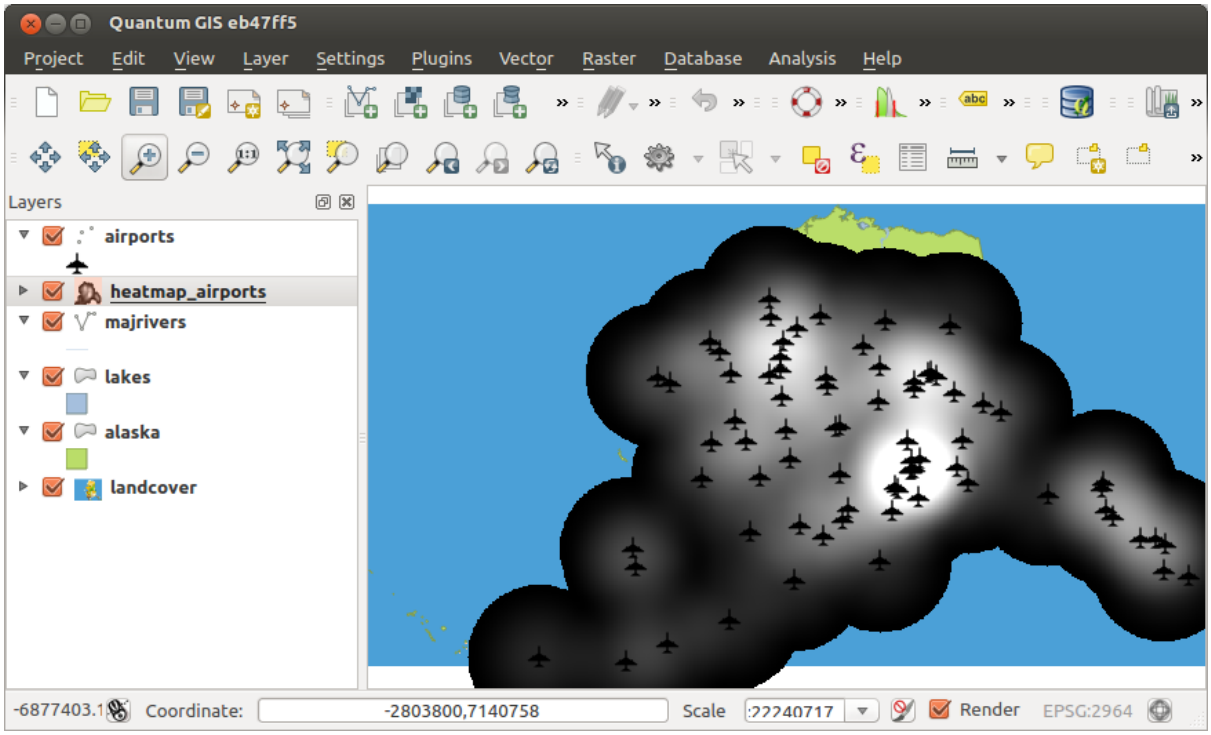


Fig. 28.4: The heatmap after loading looks like a grey surface

1. Open the properties dialog of the `heatmap_airports` layer (select the layer `heatmap_airports`, open the context menu with the right mouse button and select *Properties*).
2. Select the *Symbology* tab.
3. Change the *Render type* to «Singleband pseudocolor».
4. Select a suitable *Color ramp* , for instance `YlOrRd`.
5. Click the *Classify* button.
6. Press *OK* to update the layer.

The final result is shown in Fig. 28.5.

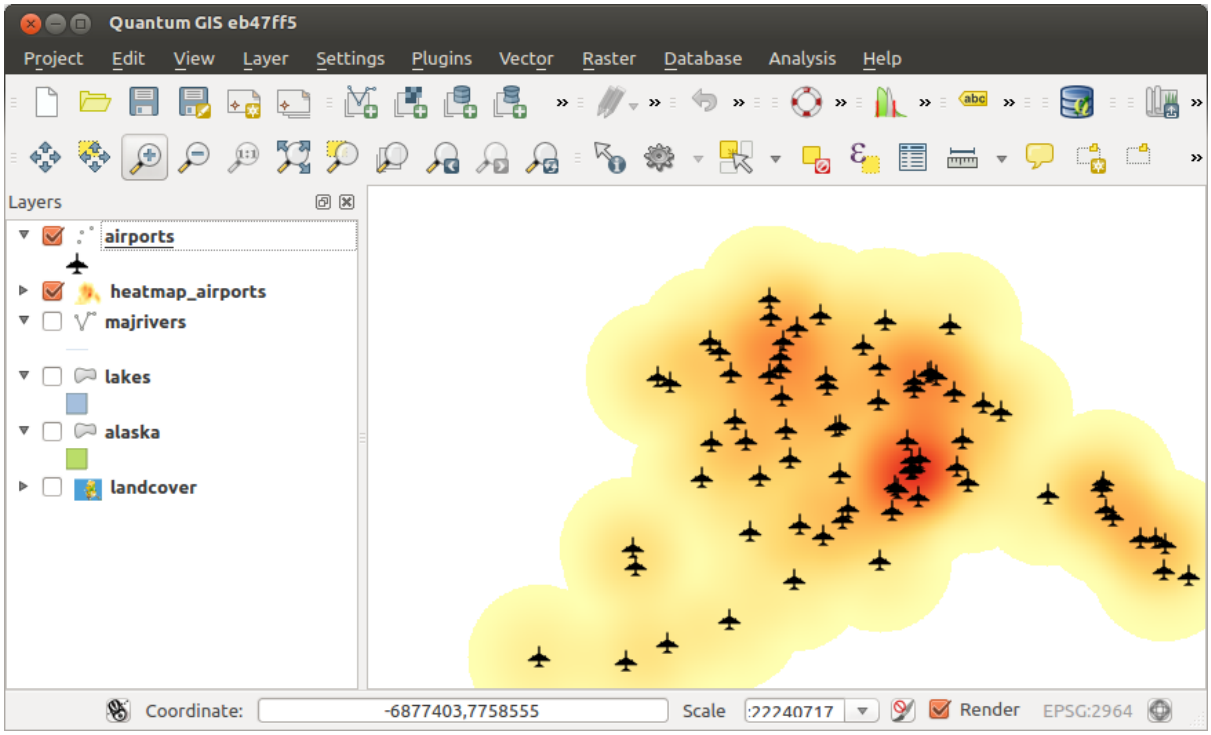


Fig. 28.5: Styled heatmap of airports of Alaska

Cod Python

Algorithm ID: `qgis:heatmapkerneldensityestimation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Interpolare IDW

Generates an Inverse Distance Weighted (IDW) interpolation of a point vector layer.

Sample points are weighted during interpolation such that the influence of one point relative to another declines with distance from the unknown point you want to create.

The IDW interpolation method also has some disadvantages: the quality of the interpolation result can decrease, if the distribution of sample data points is uneven.

Furthermore, maximum and minimum values in the interpolated surface can only occur at sample data points.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat(uri) de intrare	`INTERPOLATION_`	[string]	Vector layer(s) and field(s) to use for the interpolation, coded in a string (see the `ParameterInterpolationData` class in InterpolationWidgets for more details). The following GUI elements are provided to compose the interpolation data string: • **Strat vectorial** [vector: oricare] • **Interpolation attribute** [tablefield: numeric]: Attribute to use in the interpolation • **Use Z-coordinate for interpolation** [boolean]: Uses the layer's stored Z values (Default: False) For each of the added layer-field combinations, a type can be chosen: • *Points* • *Structured lines* • *Break lines* In the string, the layer-field elements are separated by `'::|::'`. The sub-elements of the layer-field elements are separated by `'::~::'`.
Distance coefficient P	`DISTANCE_COEFF`	[număr] Implicit: 2.0	Sets the distance coefficient for the interpolation. Minimum: 0.0, maximum: 100.0.
Extent (xmin, xmax, ymin, ymax)	`EXTENT`	[extent]	Extent of the output raster layer. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

continues on next page

Tabelul 28.17 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output raster size	`PIXEL_SIZE`	[număr] Implicit: 0.1	Pixel size of the output raster layer in layer units. In the GUI, the size can be specified by the number of rows (`Number of rows`) / columns (`Number of columns`) **or** the pixel size(`Pixel Size X` / `Pixel Size Y`). Increasing the number of rows or columns will decrease the cell size and increase the file size of the output raster. The values in `Rows`, `Columns`, `Pixel Size X` and `Pixel Size Y` will be updated simultaneously - doubling the number of rows will double the number of columns, and the cell size will be halved. The extent of the output raster will remain the same (approximately).
Interpolated	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Raster layer of interpolated values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated	`OUTPUT`	[raster]	Raster layer of interpolated values

Cod Python

Algorithm ID: `qgis:idwinterpolation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Line Density

Calculates for each raster cell, the density measure of linear features within a circular neighbourhood. This measure is obtained by summing all the line segments intersecting the circular neighbourhood and dividing this sum by the area of such neighbourhood. A weighting factor can be applied to the line segments.

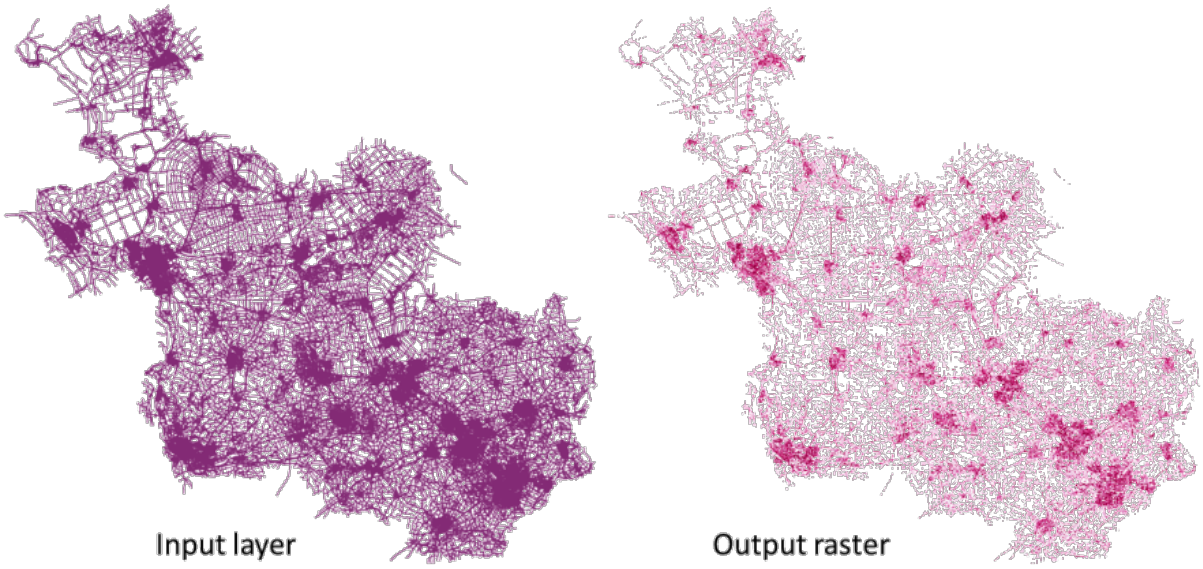


Fig. 28.6: Line density example. Input layer source: Roads Overijssel - The Netherlands (OSM).

Parametri

Etichetă	Nume	Tipul	Descriere
Stratul liniar de intrare	`INPUT`	[vector: oricare]	Input vector layer containing line features
Weight field	`WEIGHT`	[număr]	Field of the layer containing the weight factor to use during the calculation
Search Radius	`RADIUS`	[număr] Implicit: 10	Radius of the circular neighbourhood. Units can be specified here.
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 10	Pixel size of the output raster layer in layer units. The raster has square pixels.
Line density raster	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	The output as a raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Line density raster	`OUTPUT`	[raster]	The output line density raster layer.

Cod Python

ID Algorithm: `nativ:linedensity`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

TIN Interpolation

Generează o interpolare de tip Rețea Neregulată Triangulată (TIN) pentru un strat vectorial de tip punct.

With the TIN method you can create a surface formed by triangles of nearest neighbor points. To do this, circumcircles around selected sample points are created and their intersections are connected to a network of non overlapping and as compact as possible triangles. The resulting surfaces are not smooth.

The algorithm creates both the raster layer of the interpolated values and the vector line layer with the triangulation boundaries.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat(uri) de intrare	`INTERPOLATION_`	[string]	Vector layer(s) and field(s) to use for the interpolation, coded in a string (see the `ParameterInterpolationData` class in InterpolationWidgets for more details). The following GUI elements are provided to compose the interpolation data string: • **Strat vectorial** [vector: oricare] • **Interpolation attribute** [tablefield: numeric]: Attribute to use in the interpolation • **Use Z-coordinate for interpolation** [boolean]: Uses the layer's stored Z values (Default: False) For each of the added layer-field combinations, a type can be chosen: • *Points* • *Structured lines* • *Break lines* In the string, the layer-field elements are separated by `'::|::'`. The sub-elements of the layer-field elements are separated by `'::~::'`.
Interpolation method	`METHOD`	[enumerare] Implicit: 0	Set the interpolation method to be used. One of: • *Linear* • *Clough-Toucher (cubic)*

continues on next page

Tabelul 28.21 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Extent (xmin, xmax, ymin, ymax)	`EXTENT`	[extent]	Extent of the output raster layer. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Output raster size	`PIXEL_SIZE`	[număr] Implicit: 0.1	Pixel size of the output raster layer in layer units. In the GUI, the size can be specified by the number of rows (`Number of rows`) / columns (`Number of columns`) **or** the pixel size(`Pixel Size X` / `Pixel Size Y`). Increasing the number of rows or columns will decrease the cell size and increase the file size of the output raster. The values in `Rows`, `Columns`, `Pixel Size X` and `Pixel Size Y` will be updated simultaneously - doubling the number of rows will double the number of columns, and the cell size will be halved. The extent of the output raster will remain the same (approximately).
Interpolated	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	The output TIN interpolation as a raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…
Triangulation	`TRIANGULATION`	[vector: linie] Implicit: `[Omitere rezultat]`	The output TIN as a vector layer. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated	`OUTPUT`	[raster]	The output TIN interpolation as a raster layer
Triangulation	`TRIANGULATION`	[vector: linie]	The output TIN as a vector layer.

Cod Python

Algorithm ID: `qgis:tininterpolation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.7 Layer tools

Export layer(s) information

Creates a polygon layer with features corresponding to the extent of selected layer(s).

Additional layer details (CRS, provider name, file path, layer name, subset filter, abstract and attribution) are attached as attributes to each feature.

Parametri

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`LAYERS`	[vector: oricare] [list]	Input vector layers to get information on.
Rezultat	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specification of the output layer with information. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Rezultat	`OUTPUT`	[vector: poligon]	Polygon vector layer showing extent of input layers and associated information in attributes.

Cod Python

ID Algorithm: `nativ:exportlayersinformation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export to spreadsheet

Exports the attributes of a selection of vector layers into a spreadsheet document or optionally appends them to an existing spreadsheet as additional sheets.

Parametri

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`LAYERS`	[vector: oricare] [list]	Input vector layers. The output spreadsheet will consist of a sheet, for each layer, that contains the attributes of this layer.
Use field aliases as column headings	`USE_ALIAS`	[boolean] Implicit: False	Use the field aliases from the attribute table for the spreadsheet.
Export formatted values instead of raw values	`FORMATTED_VALU`	[boolean] Implicit: False	If `True`, exports the formatted, human readable values (e.g., from a *value map or value relation*) to the spreadsheet.
Overwrite existing spreadsheet	`OVERWRITE`	[boolean] Implicit: True	If the specified spreadsheet exists, setting this option to `True` will overwrite the existing spreadsheet. If this option is `False` and the spreadsheet exists, the layers will be appended as additional sheets.
Destination spreadsheet	`OUTPUT`	[file] Implicit: `[Salvare într-un fișier temporar]`	Output spreadsheet with a sheet for every layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Destination spreadsheet	`OUTPUT`	[file]	Spreadsheet with a sheet for every layer.
Layers within spreadsheet	`OUTPUT_LAYERS`	[list]	The list of sheets added to the spreadsheet.

Cod Python

ID Algorithm: `nativ:exporttospreadsheet`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extragere extindere din strat

Generates a vector layer with the minimum bounding box (rectangle with N-S orientation) that covers all the input features.

The output layer contains a single bounding box for the whole input layer.

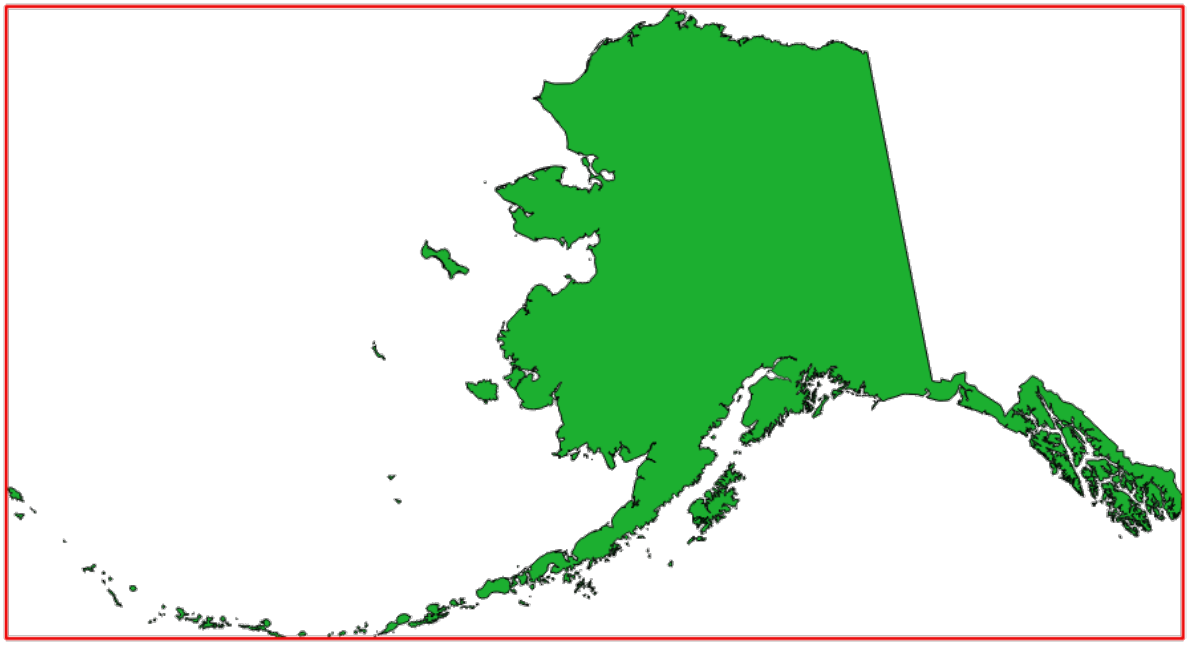


Fig. 28.7: In red the bounding box of the source layer

Default menu: *Vector* ► *Research Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat	`INPUT`	[strat]	Strat de intrare
Extindere	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the polygon vector layer for the output extent. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extindere	`OUTPUT`	[vector: poligon]	Output (polygon) vector layer with the extent (minimum bounding box)

Cod Python

Algorithm ID: `qgis:polygonfromlayerextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.8 Suprafață poliedrică

Export contours

Creates contours as a vector layer from a mesh scalar dataset.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	`INPUT`	[mesh]	The mesh layer to export data from
Dataset groups	`DATASET_GROUPS`	[layer] [list]	The dataset groups
Dataset time	`DATASET_TIME`	[datetime]	The time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date

continues on next page

Tabelul 28.23 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Increment between contour levels Opţional	`INCREMENT`	[număr] Implicit: *Nesetat*	Interval between generated levels.
Minimum contour level Opţional	`MINIMUM`	[număr] Implicit: *Nesetat*	Starting level values of contours.
Maximum contour level Opţional	`MAXIMUM`	[număr] Implicit: *Nesetat*	Maximum values of contours, i.e. no generated levels will be greater than this value.
List of contours level Opţional	`CONTOUR_LEVEL_`	[număr] Implicit: *Nesetat*	List of wanted levels of contours (separated by commas). If filled, the increment, minimum, and maximum fields will not be considered.
Output coordinate system Opţional	`CRS_OUTPUT`	[crs]	Coordinate Reference System to assign to the output
Exported contour lines	`OUTPUT_LINES`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line layer representing the contours of the mesh layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Exported contour polygons	`OUTPUT_POLYGON`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon layer representing the contours of the mesh layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Exported contour lines	`OUTPUT_LINES`	[vector: linie]	Line layer representing the contours of the mesh layer.
Exported contour polygons	`OUTPUT_POLYGON`	[vector: poligon]	Polygon layer representing the contours of the mesh layer.

Cod Python

ID Algorithm: `nativ:meshcontours`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export cross section dataset values on lines from mesh

Extracts a mesh dataset's values from lines contained in a vector layer.

Each line is discretized with a resolution distance parameter for extraction of values on its vertices.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	`INPUT`	[mesh]	The mesh layer to export data from
Dataset groups	`DATASET_GROUPS`	[layer] [list]	The dataset groups
Dataset time	`DATASET_TIME`	[datetime]	The time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date
Lines for data export	`INPUT_LINES`	[vector: linie]	Lines where the data will be extracted from the dataset mesh
Line segmentation resolution	`RESOLUTION`	[număr] Implicit: 10.0	The distance between points on the lines where the data will be extracted from the dataset mesh.
Digits count for dataset value	`DATASET_DIGITS`	[număr] Implicit: 2	Number of digits to round dataset values
Exported data CSV file	`OUTPUT`	[file] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Exported data CSV file	`OUTPUT`	[file]	

Cod Python

ID Algorithm: `nativ:meshexportcrosssection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export mesh edges

Exports a mesh layer's edges to a line vector layer, with the dataset values on edges as attribute values.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	`INPUT`	[mesh]	The mesh layer to export data from
Dataset groups	`DATASET_GROUPS`	[layer] [list]	The dataset groups
Dataset time	`DATASET_TIME`	[datetime]	The time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date
Output coordinate system Opţional	`CRS_OUTPUT`	[crs]	Coordinate Reference System to assign to the output
Export vector option	`VECTOR_OPTION`	[enumerare]	Coordinate type of vector value exportation. • 0 — Cartezian (x,y) • 1 — Polar (magnitudine, grad) • 2 — Cartezian și polar
Output vector layer	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specification of the output file. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Output vector layer	OUTPUT	[vector: linie]	Output vector line layer containing the edges of the input mesh layer with associated dataset values

Cod Python

ID Algorithm: `nativ:exportmeshedges`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export mesh faces

Exports a mesh layer's faces to a polygon vector layer, with the dataset values on faces as attribute values.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	INPUT	[mesh]	The mesh layer to export data from
Dataset groups	DATASET_GROUPS	[layer] [list]	The dataset groups
Dataset time	DATASET_TIME	[datetime]	The time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date
Output coordinate system Opţional	CRS_OUTPUT	[crs]	Coordinate Reference System to assign to the output
Export vector option	VECTOR_OPTION	[enumerare]	Coordinate type of vector value exportation. • 0 — Cartezian (x,y) • 1 — Polar (magnitudine, grad) • 2 — Cartezian și polar
Output vector layer	OUTPUT	[vector: poligon] Implicit: [Creare Strat temporar]	Specification of the output file. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Output vector layer	OUTPUT	[vector: poligon]	Output vector polygon layer containing the faces of the input mesh layer with associated dataset values

Cod Python

ID Algorithm: `nativ:exportmeshfaces`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export mesh on grid

Exports a mesh layer's dataset values to a gridded point vector layer, with the dataset values on this point as attribute values.

For data on volume (3D stacked dataset values), the exported dataset values are averaged on faces using the method defined in *the mesh layer properties* (default is Multi level averaging method). 1D meshes are not supported.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	INPUT	[mesh]	The mesh layer to export data from
Dataset groups	DATASET_GROUPS	[layer] [list]	The dataset groups
Dataset time	DATASET_TIME	[datetime]	The time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date

continues on next page

Tabelul 28.27 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Extindere Opţional	`EXTENT`	[extent]	Specify the spatial extent on which to process the data. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Grid spacing Opţional	`GRID_SPACING`	[număr] Implicit: 10.0	Spacing between the sample points to use
Output coordinate system Opţional	`CRS_OUTPUT`	[crs]	Coordinate Reference System to assign to the output
Export vector option	`VECTOR_OPTION`	[enumerare]	Coordinate type of vector value exportation. • 0 — Cartezian (x,y) • 1 — Polar (magnitudine, grad) • 2 — Cartezian și polar
Output vector layer	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specification of the output file. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Output vector layer	`OUTPUT`	[vector: punct]	Output vector point layer with dataset values computed from the overlaid face.

Cod Python

ID Algorithm: `nativ:exportmeshongrid`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export mesh vertices

Exports a mesh layer's vertices to a point vector layer, with the dataset values on vertices as attribute values.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	`INPUT`	[mesh]	The mesh layer to export data from
Dataset groups	`DATASET_GROUPS`	[layer] [list]	The dataset groups
Dataset time	`DATASET_TIME`	[datetime]	The time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date
Output coordinate system Opţional	`CRS_OUTPUT`	[crs]	Coordinate Reference System to assign to the output
Export vector option	`VECTOR_OPTION`	[enumerare]	Coordinate type of vector value exportation. • 0 — Cartezian (x,y) • 1 — Polar (magnitudine, grad) • 2 — Cartezian și polar
Output vector layer	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specification of the output file. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Output vector layer	`OUTPUT`	[vector: punct]	Output vector point layer containing the vertices of the input mesh layer with associated dataset values

Cod Python

ID Algorithm: `nativ:exportmeshvertices`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export time series values from points of a mesh dataset

Extracts a mesh dataset's time series values from points contained in a vector layer.

If the time step is kept to its default value (0 hours), the time step used is the one of the two first datasets of the first selected dataset group.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	`INPUT`	[mesh]	The mesh layer to extract data from
Dataset groups	`DATASET_GROUPS`	[layer] [list]	The dataset groups
Starting time	`STARTING_TIME`	[datetime]	The start of the time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date
Finishing time	`FINISHING_TIME`	[datetime]	The end of the time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date
Time step (hours) Opţional	`TIME_STEP`	[număr] Implicit: 0	Time between two consecutive steps to extract. Keep `0` to use time step of the first selected dataset group.
Points for data export	`INPUT_POINTS`	[vector: punct]	Vector layer containing points where the data will be extracted from the dataset mesh
Digits count for coordinates	`COORDINATES_DI`	[număr]	Number of digits to round coordinate values Implicit: 2
Digits count for dataset value	`DATASET_DIGITS`	[număr] Implicit: 2	Number of digits to round dataset values

continues on next page

Tabelul 28.29 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Exported data CSV file	`OUTPUT`	[file] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Exported data CSV file	`OUTPUT`	[file]	`.CSV` file containing the mesh dataset time series values at the overlaying point features

Cod Python

ID Algorithm: `nativ:meshexporttimeseries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rasterize mesh dataset

Creates a raster layer from a mesh dataset.

For data on volume (3D stacked dataset values), the exported dataset values are averaged on faces using the method defined in *the mesh layer properties* (default is Multi level averaging method). 1D meshes are not supported.

Parametri

Etichetă	Nume	Tipul	Descriere
Input mesh layer	`INPUT`	[mesh]	The mesh layer to export data from
Dataset groups	`DATASET_GROUPS`	[layer] [list]	The dataset groups
Dataset time	`DATASET_TIME`	[datetime]	The time range to take into account • 0 — Ora canevasului curent • 1 — Date/Time definite • 2 — Pasul de timp al setului de date

continues on next page

Tabelul 28.30 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Extindere Opţional	`EXTENT`	[extent]	Specify the spatial extent on which to process the data. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size of the output raster layer.
Output coordinate system Opţional	`CRS_OUTPUT`	[crs]	Coordinate Reference System to assign to the output
Output raster layer	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Output raster layer	`OUTPUT`	[raster]	Output raster layer with dataset values computed from the mesh layer.

Cod Python

ID Algorithm: `nativ:meshrasterize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

TIN mesh creation

Creates a TIN mesh layer from vector layers. The TIN mesh is created using a Delaunay triangulation.

Parametri

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	SOURCE_DATA	[vector: oricare] [list]	Vector layers to combine to generate the mesh layer
Vector layer	GUI ONLY	[vector: oricare] [list]	A selector for the vector layers to combine to generate the mesh layer
Value on vertex	GUI ONLY	[tablefield: any]	A selector of the field to use from the selected layer. Each vertex is assigned the corresponding value of its original feature.
Use Z-coordinate for value on vertex	GUI ONLY	[boolean] Implicit: *False*	If checked, the Z value of vector layer points or polygons/lines vertices will be used to assign the Z value of the vertex mesh layer. Only available if the input layers are 3D.
Output format	MESH_FORMAT	[enumerare] Implicit: 2DM	Output format of the generated layer • 0 — 2DM • 1 — SELAFIN • 2 — PLY • 3 — Ugrid
Output coordinate system Opţional	CRS_OUTPUT	[crs]	Coordinate Reference System to assign to the output
Output file	OUTPUT_MESH	[mesh] Implicit: [Salvare într-un fişier temporar]	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Output file	OUTPUT_MESH	[mesh]	Output mesh layer with dataset values computed from the vector layers.

Cod Python

ID Algorithm: `nativ:tinmeshcreation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.9 Instrumentele modelatorului

Atenționare: These tools are only available in the model designer. They are not available in the Processing Toolbox.

Calculate expression

`NEW in 3.34`

It calculates the result of a QGIS expression and eliminates the need to use the same expression multiple times throughout a model when the same result is required more than once. Additionally, it enables use cases that would otherwise not be possible. For instance, you can generate a timestamp value once and use it multiple times within the model, if the timestamp were recalculated every time, the values would vary during the model's runtime.

Parametri

Etichetă	Nume	Tipul	Descriere
Input	`INPUT`	[expresie]	Expression to calculate

Rezultate

Etichetă	Nume	Tipul	Descriere
Value	`OUTPUT`	[Result Value]	Calculated result value, the data type of the output will vary based on the specific expression used in the algorithm.

Cod Python

Algorithm ID: `native:calculateexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Ramura Condițională

Adaugă o ramură condiționată într-un model, permițând executarea unor părți ale modelului pe baza rezultatului evaluării unei expresii. Mai ales prin utilizarea dependențelor instrumentelor, pentru a controla fluxul unui model.

Parametri

Etichetă	Nume	Tipul	Descriere
Câmp	`BRANCH`	[șir]	Numele condiției
Câmp	`CONDITION`	[expresie]	Expresia de evaluat

Rezultate

Fără.

Cod Python

ID Algoritm: `nativ:condition`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Creează director

Creează un director nou pe un sistem de fișiere. Directoarele vor fi create recursiv, creând toate directoarele părinte necesare pentru a construi calea completă a directorului specificat. Nu vor fi generate erori dacă directorul există deja.

Parametri

Etichetă	Nume	Tipul	Descriere
Calea directorului	`PATH`	[șir]	Calea dosarului de creat

Rezultate

Etichetă	Nume	Tipul	Descriere
Rezultat	`OUTPUT`	[folder]	Folder creat

Cod Python

ID Algorithm: `nativ:createdirectory`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Filtrare entitate

Filtrează entitățile din stratul de intrare și le redirecționează către una sau mai multe ieșiri. Dacă nu cunoașteți numele atributelor care sunt comune tuturor straturilor de intrare posibile, filtrarea este posibilă numai pentru geometria entităților și mecanismele generale de înregistrare, cum ar fi `$id` și `uuid`.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul de intrare.
Rezultate și filtre (una sau mai multe)	`OUTPUT_<name of the filter>`	[la fel ca intrarea]	Straturile de ieșire cu filtre (atât timp cât există filtre).

Rezultate

Etichetă	Nume	Tipul	Descriere
Rezultat (una sau mai multe)	`nativ:filter_1 of filter>`	[la fel ca intrarea]	Straturile de ieșire cu entitățile filtrate (atât timp cât există filtre).

Cod Python

ID Algorithm: `nativ:filter`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Filtrează straturile după tip

Filtrează straturile după tipul lor. Straturile de intrare vor fi direcționate către diferite ieșiri, după genul straturilor: vectorial sau raster.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[strat]	Strat de Hartă Generic

Rezultate

Etichetă	Nume	Tipul	Descriere
Entități vectoriale Opţional	`VECTOR`	[vector]	Un Strat Vectorial al intrării, dacă este compatibil
Strat raster Opţional	`RASTER`	[raster]	Un Strat Raster al intrării, dacă este compatibil

Cod Python

ID Algorithm: `nativ:filterlayersbytype`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Încarcă stratul în proiect

Încarcă un strat în proiectul curent.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat	`INPUT`	[strat]	Stratul de încărcat în legendă
Numele stratului încărcat	`NAME`	[șir]	Numele stratului încărcat

Rezultate

Etichetă	Nume	Tipul	Descriere
Strat	`OUTPUT`	[la fel ca intrarea]	Stratul încărcat (redenumit)

Cod Python

ID Algorithm: `nativ:loadlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Generare excepție

Raises an exception and cancels a model's execution. The exception message can be customized, and optionally an expression based condition can be specified. If an expression condition is used, then the exception will only be raised if the expression result is true. A false result indicates that no exception will be raised, and the model execution can continue uninterrupted.

Parametri

Etichetă	Nume	Tipul	Descriere
Mesaj	`MESSAGE`	[șir]	Mesajul de afișat
Condiție Opţional	`CONDITION`	[expresie]	Expresia de evaluat în cazul true

Rezultate

A message in the log panel.

Cod Python

ID Algorithm: `nativ:raiseexception`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raise message

Raises an information message in the log. The message can be customized, and optionally an expression based condition can be specified. If an expression condition is used, then the message will only be logged if the expression result is true. A false result indicates that no message will be logged.

Parametri

Etichetă	Nume	Tipul	Descriere
Information message	`MESSAGE`	[șir]	Mesajul de afișat
Condiție Opţional	`CONDITION`	[expresie]	Expresia de evaluat în cazul true

Rezultate

A message in the log panel.

Cod Python

Algorithm ID: `native:raisemessage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Generare excepție

Raises a warning message in the log. The warning message can be customized, and optionally an expression based condition can be specified. If an expression condition is used, then the warning will only be logged if the expression result is true. A false result indicates that no warning will be logged.

Parametri

Etichetă	Nume	Tipul	Descriere
Mesaj	`MESSAGE`	[șir]	Mesajul de afișat
Condiție Opţional	`CONDITION`	[expresie]	Expresia de evaluat în cazul true

Rezultate

A message in the log panel.

Cod Python

ID Algorithm: `nativ:raisewarning`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Redenumire strat

Redenumește un strat.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat	`INPUT`	[strat]	Stratul de redenumit
New name	`NAME`	[șir]	The new name of the layer

Rezultate

Etichetă	Nume	Tipul	Descriere
Strat	`OUTPUT`	[la fel ca intrarea]	The (renamed) output layer

Cod Python

ID Algorithm: `nativ:renamelayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Salvare jurnal în fișier

Saves the model's execution log to a file. Optionally, the log can be saved in a HTML formatted version.

Parametri

Etichetă	Nume	Tipul	Descriere
Utilizează HTML	`USE_HTML`	[Boolean] Implicit: False	Use HTML formatting

Rezultate

Etichetă	Nume	Tipul	Descriere
File	`OUTPUT`	[șir]	Destination of the log

Cod Python

ID Algorithm: `nativ:savelog`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Set project variable

Sets an expression variable for the current project.

Parametri

Etichetă	Nume	Tipul	Descriere
Numele variabilei	`NAME`	[șir]	Numele variabilei
Valoarea variabilei	`VALUE`	[șir]	Valoarea de înmagazinat

Rezultate

Fără.

Cod Python

ID Algorithm: `nativ:setprojectvariable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

String concatenation

Concatenates two strings into a single one in the Processing Modeler.

Parametri

Etichetă	Nume	Tipul	Descriere
Intrare 1	`INPUT_1`	[șir]	First string
Intrare 2	`INPUT_2`	[șir]	Second string

Rezultate

Etichetă	Nume	Tipul	Descriere
Concatenation	`CONCATENATION`	[șir]	The concatenated string

Cod Python

ID Algorithm: `nativ:stringconcatenation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Variable distance buffer

Atenționare: This algorithm is deprecated and can be removed anytime. Prefer using *Buffer* algorithm instead.

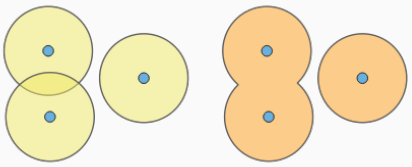
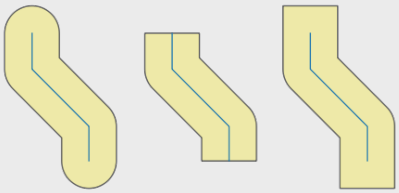
Computes a buffer area for all the features in an input layer.

The size of the buffer for a given feature is defined by an attribute, so it allows different features to have different buffer sizes.

Vezi și:

Buffer

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Distance field	`DISTANCE`	[tablefield: numeric]	Attribute for the distance radius of the buffer
Segments	`SEGMENTS`	[număr] Implicit: *5*	Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.
Dissolve result	`DISSOLVE`	[boolean] Implicit: *False*	Choose to dissolve the final buffer, resulting in a single feature covering all input features.  Fig. 28.8: Normal and dissolved buffer
End cap style	`END_CAP_STYLE`	[enumerare] Implicit: Round	Controls how line endings are handled in the buffer.  Fig. 28.9: Round, flat and square cap styles
Join style	`JOIN_STYLE`	[enumerare] Implicit: Round	Specifies whether round, miter or beveled joins should be used when offsetting corners in a line.
Miter limit	`MITER_LIMIT`	[număr] Implicit: 2.0	Only applicable for mitered join styles, and controls the maximum distance from the offset curve to use when creating a mitered join.

Rezultate

Etichetă	Nume	Tipul	Descriere
Buffer	`OUTPUT`	[vector: poligon]	Buffer polygon vector layer.

Cod Python

Algorithm ID: `qgis:variabledistancebuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.10 Network analysis

Service area (from layer)

Returns all the edges or parts of edges of a network that can be reached within a distance or a time, starting from a point layer. This allows evaluation of accessibility within a network, e.g. what are the places I can navigate to on a road network without spending cost greater than a given value (the cost can be distance or time).

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Vector layer representing network	`INPUT`	[vector: linie]	Line vector layer representing the network to be covered
Vector layer with start points	`START_POINTS`	[vector: punct]	Point vector layer whose features are used as start points to generate the service areas
Path type to calculate	`STRATEGY`	[enumerare] Implicit: 0	The type of path to calculate. One of: • 0 — Cel mai scurt • 1 — Cel mai rapid
Travel cost (distance for „Shortest”, time for „Fastest”)	`TRAVEL_COST`	[număr] Implicit: 0	The value is estimated as a distance (in the network layer units) when looking for the *Shortest* path and as time (in hours) for the *Fastest* path.

continues on next page

Tabelul 28.33 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Service area (lines)	`OUTPUT_LINES`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line layer for the service area. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Service area (boundary nodes)	`OUTPUT`	[vector: punct] Implicit: `[Omitere rezultat]`	Specify the output point layer for the service area boundary nodes. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Direction field Opţional	`DIRECTION_FIEL`	[tablefield: string] Implicit: 0.0	The field used to specify directions for the network edges. The values used in this field are specified with the three parameters `Value for forward direction`, `Value for backward direction` and `Value for both directions`. Forward and reverse directions correspond to a one-way edge, „both directions” indicates a two-way edge. If a feature does not have a value in this field, or no field is set then the default direction setting (provided with the `Default direction` parameter) is used.
Value for forward direction Opţional	`VALUE_FORWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a forward direction
Value for backward direction Opţional	`VALUE_BACKWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a backward direction
Value for both directions Opţional	`VALUE_BOTH`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify bidirectional edges

continues on next page

Tabelul 28.34 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Default direction	`DEFAULT_DIRECT`	[enumerare] Implicit: 2	If a feature has no value set in the direction field or if no direction field is set, then this direction value is used. One of: • 0 — Direcția înainte • 1 — Direcția înapoi • 2 — Ambele direcții
Speed field Opţional	`SPEED_FIELD`	[tablefield: string]	Field providing the speed value (in `km/h`) for the edges of the network when looking for the fastest path. If a feature does not have a value in this field, or no field is set then the default speed value (provided with the `Default speed` parameter) is used.
Default speed (km/h)	`DEFAULT_SPEED`	[număr] Implicit: 50.0	Value to use to calculate the travel time if no speed field is provided for an edge
Topology tolerance	`TOLERANCE`	[număr] Implicit: 0.0	Two lines with nodes closer than the specified tolerance are considered connected
Include upper/lower bound points	`INCLUDE_BOUNDS`	[boolean] Implicit: *False*	Creates a point layer output with two points for each edge at the boundaries of the service area. One point is the start of that edge, the other is the end.

Rezultat

Etichetă	Nume	Tipul	Descriere
Service area (boundary nodes)	`OUTPUT`	[vector: punct]	The output point layer with the service area boundary nodes.
Service area (lines)	`OUTPUT_LINES`	[vector: linie]	Line layer representing the parts of the network that can be serviced by the start points, for the given cost.

Cod Python

Algorithm ID: `qgis:serviceareafromlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Service area (from point)

Returns all the edges or parts of edges of a network that can be reached within a given distance or time, starting from a point feature. This allows the evaluation of accessibility within a network, e.g. what are the places I can navigate to on a road network without spending a cost greater than a given value (the cost can be distance or time).

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Vector layer representing the network	`INPUT`	[vector: linie]	Line vector layer representing the network to be covered
Start point (x, y)	`START_POINT`	[coordinates]	Coordinate of the point to calculate the service area around.
Path type to calculate	`STRATEGY`	[enumerare] Implicit: 0	The type of path to calculate. One of: • 0 — Cel mai scurt • 1 — Cel mai rapid
Travel cost (distance for „Shortest”, time for „Fastest”)	`TRAVEL_COST`	[număr] Implicit: 0	The value is estimated as a distance (in the network layer units) when looking for the *Shortest* path and as time (in hours) for the *Fastest* path.
Service area (lines)	`OUTPUT_LINES`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line layer for the service area. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Service area (boundary nodes)	`OUTPUT`	[vector: punct] Implicit: `[Omitere rezultat]`	Specify the output point layer for the service area boundary nodes. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Direction field Opţional	`DIRECTION_FIEL`	[tablefield: string] Implicit: 0.0	The field used to specify directions for the network edges. The values used in this field are specified with the three parameters `Value for forward direction`, `Value for backward direction` and `Value for both directions`. Forward and reverse directions correspond to a one-way edge, „both directions" indicates a two-way edge. If a feature does not have a value in this field, or no field is set then the default direction setting (provided with the `Default direction` parameter) is used.
Value for forward direction Opţional	`VALUE_FORWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a forward direction
Value for backward direction Opţional	`VALUE_BACKWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a backward direction
Value for both directions Opţional	`VALUE_BOTH`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify bidirectional edges
Default direction	`DEFAULT_DIRECT`	[enumerare] Implicit: 2	If a feature has no value set in the direction field or if no direction field is set, then this direction value is used. One of: • 0 — Direcția înainte • 1 — Direcția înapoi • 2 — Ambele direcții
Speed field Opţional	`SPEED_FIELD`	[tablefield: string]	Field providing the speed value (in `km/h`) for the edges of the network when looking for the fastest path. If a feature does not have a value in this field, or no field is set then the default speed value (provided with the `Default speed` parameter) is used.
Default speed (km/h)	`DEFAULT_SPEED`	[număr] Implicit: 50.0	Value to use to calculate the travel time if no speed field is provided for an edge
Topology tolerance	`TOLERANCE`	[număr] Implicit: 0.0	Two lines with nodes closer than the specified tolerance are considered connected

Include upper/lower bound points	`INCLUDE_BOUNDS`	[boolean] Implicit: *False*	Creates a point layer output with two points for each edge at the boundaries of the service area. One point is the start of that edge, the other is the end.

Rezultat

Etichetă	Nume	Tipul	Descriere
Service area (boundary nodes)	`OUTPUT`	[vector: punct]	The output point layer with the service area boundary nodes.
Service area (lines)	`OUTPUT_LINES`	[vector: linie]	Line layer representing the parts of the network that can be serviced by the start point, for the given cost.

Cod Python

ID Algorithm: `nativ:serviceareafrompoint`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Shortest path (layer to point)

Computes the optimal (shortest or fastest) routes from multiple start points defined by a vector layer and a given end point.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Vector layer representing network	`INPUT`	[vector: linie]	Line vector layer representing the network to be covered
Path type to calculate	`STRATEGY`	[enumerare] Implicit: 0	The type of path to calculate. One of: • 0 — Cel mai scurt • 1 — Cel mai rapid
Vector layer with start points	`START_POINTS`	[vector: punct]	Point vector layer whose features are used as start points of the routes
End point (x, y)	`END_POINT`	[coordinates]	Point feature representing the end point of the routes
Shortest path	`OUTPUT`	[vector: linie]	Specify the output line layer for the shortest paths. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Direction field Opţional	`DIRECTION_FIEL`	[tablefield: string] Implicit: 0.0	The field used to specify directions for the network edges. The values used in this field are specified with the three parameters `Value for forward direction`, `Value for backward direction` and `Value for both directions`. Forward and reverse directions correspond to a one-way edge, „both directions" indicates a two-way edge. If a feature does not have a value in this field, or no field is set then the default direction setting (provided with the `Default direction` parameter) is used.
Value for forward direction Opţional	`VALUE_FORWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a forward direction
Value for backward direction Opţional	`VALUE_BACKWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a backward direction
Value for both directions Opţional	`VALUE_BOTH`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify bidirectional edges
Default direction	`DEFAULT_DIRECT`	[enumerare] Implicit: 2	If a feature has no value set in the direction field or if no direction field is set, then this direction value is used. One of: • 0 — Direcția înainte • 1 — Direcția înapoi • 2 — Ambele direcții
Speed field Opţional	`SPEED_FIELD`	[tablefield: string]	Field providing the speed value (in `km/h`) for the edges of the network when looking for the fastest path. If a feature does not have a value in this field, or no field is set then the default speed value (provided with the `Default speed` parameter) is used.
Default speed (km/h)	`DEFAULT_SPEED`	[număr] Implicit: 50.0	Value to use to calculate the travel time if no speed field is provided for an edge
Topology tolerance	`TOLERANCE`	[număr] Implicit: 0.0	Two lines with nodes closer than the specified tolerance are considered connected

Rezultat

Etichetă	Nume	Tipul	Descriere
Shortest path	`OUTPUT`	[vector: linie]	Line layer of the shortest or fastest path from each of the start points to the end point

Cod Python

ID Algorithm: `nativ:shortestpathlayertopoint`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Shortest path (point to layer)

Computes the optimal (shortest or fastest) routes between a given start point and multiple end points defined by a point vector layer.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Vector layer representing network	`INPUT`	[vector: linie]	Line vector layer representing the network to be covered
Path type to calculate	`STRATEGY`	[enumerare] Implicit: 0	The type of path to calculate. One of: • 0 — Cel mai scurt • 1 — Cel mai rapid
Start point (x, y)	`START_POINT`	[coordinates]	Point feature representing the start point of the routes
Vector layer with end points	`END_POINTS`	[vector: punct]	Point vector layer whose features are used as end points of the routes
Shortest path	`OUTPUT`	[vector: linie]	Specify the output line layer for the shortest paths. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Direction field Opţional	`DIRECTION_FIEL`	[tablefield: string] Implicit: 0.0	The field used to specify directions for the network edges. The values used in this field are specified with the three parameters `Value for forward direction`, `Value for backward direction` and `Value for both directions`. Forward and reverse directions correspond to a one-way edge, „both directions" indicates a two-way edge. If a feature does not have a value in this field, or no field is set then the default direction setting (provided with the `Default direction` parameter) is used.
Value for forward direction Opţional	`VALUE_FORWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a forward direction
Value for backward direction Opţional	`VALUE_BACKWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a backward direction
Value for both directions Opţional	`VALUE_BOTH`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify bidirectional edges
Default direction	`DEFAULT_DIRECT`	[enumerare] Implicit: 2	If a feature has no value set in the direction field or if no direction field is set, then this direction value is used. One of: • 0 — Direcția înainte • 1 — Direcția înapoi • 2 — Ambele direcții
Speed field Opţional	`SPEED_FIELD`	[tablefield: string]	Field providing the speed value (in `km/h`) for the edges of the network when looking for the fastest path. If a feature does not have a value in this field, or no field is set then the default speed value (provided with the `Default speed` parameter) is used.
Default speed (km/h)	`DEFAULT_SPEED`	[număr] Implicit: 50.0	Value to use to calculate the travel time if no speed field is provided for an edge
Topology tolerance	`TOLERANCE`	[număr] Implicit: 0.0	Two lines with nodes closer than the specified tolerance are considered connected

Rezultat

Etichetă	Nume	Tipul	Descriere
Shortest path	`OUTPUT`	[vector: linie]	Line layer of the shortest or fastest path from each of the start points to the end point

Cod Python

ID Algorithm: `nativ:shortestpathpointtolayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Shortest path (point to point)

Computes the optimal (shortest or fastest) route between a given start point and a given end point.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Vector layer representing network	`INPUT`	[vector: linie]	Line vector layer representing the network to be covered
Path type to calculate	`STRATEGY`	[enumerare] Implicit: 0	The type of path to calculate. One of: • 0 — Cel mai scurt • 1 — Cel mai rapid
Start point (x, y)	`START_POINT`	[coordinates]	Point feature representing the start point of the routes
End point (x, y)	`END_POINT`	[coordinates]	Point feature representing the end point of the routes
Shortest path	`OUTPUT`	[vector: linie]	Specify the output line layer for the shortest paths. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Direction field Opţional	`DIRECTION_FIEL`	[tablefield: string] Implicit: 0.0	The field used to specify directions for the network edges. The values used in this field are specified with the three parameters `Value for forward direction`, `Value for backward direction` and `Value for both directions`. Forward and reverse directions correspond to a one-way edge, „both directions" indicates a two-way edge. If a feature does not have a value in this field, or no field is set then the default direction setting (provided with the `Default direction` parameter) is used.
Value for forward direction Opţional	`VALUE_FORWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a forward direction
Value for backward direction Opţional	`VALUE_BACKWARD`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify edges with a backward direction
Value for both directions Opţional	`VALUE_BOTH`	[string] Implicit: «» (șirul vid)	Value set in the direction field to identify bidirectional edges
Default direction	`DEFAULT_DIRECT`	[enumerare] Implicit: 2	If a feature has no value set in the direction field or if no direction field is set, then this direction value is used. One of: • 0 — Direcția înainte • 1 — Direcția înapoi • 2 — Ambele direcții
Speed field Opţional	`SPEED_FIELD`	[tablefield: string]	Field providing the speed value (in `km/h`) for the edges of the network when looking for the fastest path. If a feature does not have a value in this field, or no field is set then the default speed value (provided with the `Default speed` parameter) is used.
Default speed (km/h)	`DEFAULT_SPEED`	[număr] Implicit: 50.0	Value to use to calculate the travel time if no speed field is provided for an edge
Topology tolerance	`TOLERANCE`	[număr] Implicit: 0.0	Two lines with nodes closer than the specified tolerance are considered connected

Rezultat

Etichetă	Nume	Tipul	Descriere
Shortest path	`OUTPUT`	[vector: linie]	Line layer of the shortest or fastest path from each of the start point to the end point

Cod Python

ID Algorithm: `nativ:shortestpathpointtopoint`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.11 Plots

Bar plot

Creates a bar plot from a category and a layer field.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Category field name	`NAME_FIELD`	[tablefield: any]	Categorical field to use for grouping the bars (X axis)
Value field	`VALUE_FIELD`	[tablefield: any]	Value to use for the plot (Y axis).
Bar plot	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Bar plot	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:barplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Box plot

Creates a box plot from a category field and a numerical layer field.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Category name field	`NAME_FIELD`	[tablefield: any]	Categorical field to use for grouping the boxes (X axis)
Value field	`VALUE_FIELD`	[tablefield: any]	Value to use for the plot (Y axis).
Additional statistic lines	`MSD`	[enumerare] Implicit: 0	Additional statistics information to add to the plot. One of: • 0 — Arată Media • 1 — Arată Abaterea Standard • 2 — Nu se afișează abaterea medie și standard
Box plot	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Box plot	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:boxplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Mean and standard deviation plot

Creates a box plot with mean and standard deviation values.

Parametri

Etichetă	Nume	Tipul	Descriere
Input table	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Category name field	`NAME_FIELD`	[tablefield: any]	Categorical field to use for grouping the boxes (X axis)
Value field	`VALUE_FIELD`	[tablefield: any]	Value to use for the plot (Y axis).
Plot	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Plot	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:meanandstandarddeviationplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Polar plot

Generates a polar plot based on the value of an input vector layer.

Two fields must be entered as parameters: one that defines the category each feature (to group features) and another one with the variable to plot (this has to be a numeric one).

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Category name field	`NAME_FIELD`	[tablefield: any]	Categorical field to use for grouping the features (X axis)
Value field	`VALUE_FIELD`	[tablefield: any]	Value to use for the plot (Y axis).
Polar plot	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Polar plot	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:polarplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster layer histogram

Generates a histogram with the values of a raster layer.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Numărul benzii	`BAND`	[raster band]	Raster band to use for the histogram
number of bins	`BINS`	[număr] Implicit: 10	The number of bins to use in the histogram (X axis). Minimum 2.
Histogram	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Histogram	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:rasterlayerhistogram`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Vector layer histogram

Generates a histogram with the values of the attribute of a vector layer.

The attribute to use for computing the histogram must be numeric.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Attribute	`FIELD`	[tablefield: any]	Value to use for the plot (Y axis).
number of bins	`BINS`	[număr] Implicit: 10	The number of bins to use in the histogram (X axis). Minimum 2.
Histogram	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Histogram	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:vectorlayerhistogram`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Vector layer scatterplot

Creates a simple `X` - `Y` scatter plot for a vector layer.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
X attribute	`XFIELD`	[tablefield: any]	Field to use for the X axis
Y attribute	`YFIELD`	[tablefield: any]	Field to use for the Y axis
Scatterplot	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Scatterplot	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:vectorlayerscatterplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Vector layer scatterplot 3D

Creates a 3D scatter plot for a vector layer.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
X attribute	`XFIELD`	[tablefield: any]	Field to use for the X axis
Y attribute	`YFIELD`	[tablefield: any]	Field to use for the Y axis
Z attribute	`ZFIELD`	[tablefield: any]	Field to use for the Z axis
Histogram	`OUTPUT`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specify the HTML file for the plot. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Histogram	`OUTPUT`	[html]	HTML file with the plot. Available in the *Processing* ► *Result Viewer*.

Cod Python

Algorithm ID: `qgis:scatter3dplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.12 Point Cloud Conversion

`NEW in 3.32`

Convert format

Converts a point cloud to a different file format, e.g. creates a compressed `.LAZ`.

Parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to convert
Converted	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the point cloud file to use as output. One of: • Save to a Temporary File • Save to File…

Outputs

Label	Name	Type	Description
Converted	`OUTPUT`	[point cloud]	Output point cloud layer in a modified file format. Currently supported formats are `.LAS`, `.LAZ`, `.COPC.LAZ` and `.VPC`.

Python code

Algorithm ID: `pdal:convertformat`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Export to raster

Exports point cloud data to a 2D raster grid having cell size of given resolution, writing values from the specified attribute.

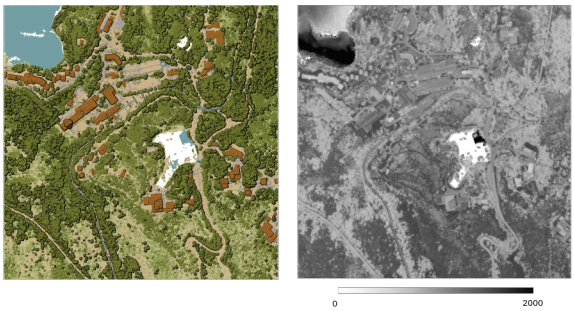


Fig. 28.10: Raster output using Intensity attribute of points

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to export
Attribute	`ATTRIBUTE`	[field] [enumeration]	A Field of the point cloud layer to extract the values from
Resolution of the density raster	`RESOLUTION`	[number] Default: 1.0	Cell size of the output raster
Tile size for parallel runs	`TILE_SIZE`	[number] Default: 1000	
Exported	`OUTPUT`	[raster] Default: `[Save to temporary file]`	Specify the raster file to export the data to. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
X origin of a tile for parallel runs Optional	`ORIGIN_X`	[number]	
Y origin of a tile for parallel runs Optional	`ORIGIN_Y`	[number]	

Outputs

Label	Name	Type	Description
Exported	`OUTPUT`	[raster]	Output raster layer features of the point cloud layer are exported to. Currently supported format is `.TIF`.

Python code

Algorithm ID: `pdal:exportraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Export raster (using triangulation)

Exports point cloud data to a 2D raster grid using a triangulation of points and then interpolating cell values from triangles.

Notă: Using this algorithm can be slower if you are dealing with a large dataset. If your point cloud is dense, you can export your ground points as a raster using the *Export to raster* algorithm.

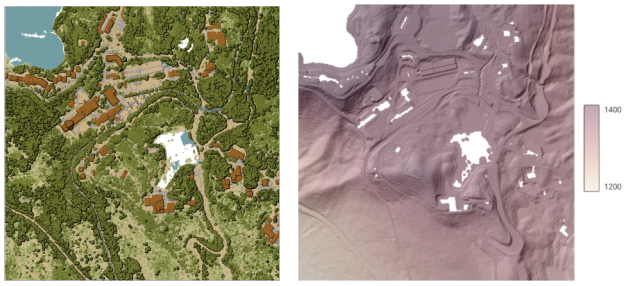


Fig. 28.11: Terrain raster output generated by point cloud triangulation

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to export
Resolution of the density raster	`RESOLUTION`	[number] Default: 1.0	Cell size of the output raster
Tile size for parallel runs	`TILE_SIZE`	[number] Default: 1000	
Exported	`OUTPUT`	[raster] Default: `[Save to temporary file]`	Specify the raster file to export the data to. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
X origin of a tile for parallel runs Optional	`ORIGIN_X`	[number]	
Y origin of a tile for parallel runs Optional	`ORIGIN_Y`	[number]	

Outputs

Label	Name	Type	Description
Exported (using triangulation)	`OUTPUT`	[raster]	Output raster layer features of the point cloud layer are exported to. Currently supported format is `.TIF`.

Python code

Algorithm ID: `pdal:exportrastertin`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Export to vector

Exports point cloud data to a vector layer with 3D points (a GeoPackage), optionally with extra attributes.

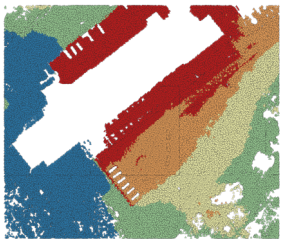


Fig. 28.12: Exporting point cloud (ground points) to a vector layer styled based on the elevation

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to export
Attribute Optional	`ATTRIBUTE`	[field] [list]	One or more fields of the point cloud layer to export with the points.
Exported	`OUTPUT`	[vector] Default: `[Save to temporary file]`	Specify the vector file to export the data to. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

Outputs

Label	Name	Type	Description
Exported	`OUTPUT`	[vector]	Output vector layer features of the point cloud layer are exported to. Currently supported format is `.GPKG`.

Python code

Algorithm ID: `pdal:exportvector`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

28.1.13 Point Cloud Data Management

`NEW in 3.32`

Assign projection

Assigns a Coordinate Reference System to a point cloud layer, if it is missing or wrong. A new layer is created.

Vezi şi:

Reproject

Parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to assign a CRS to
Desired CRS	`CRS`	[crs]	The CRS to apply to the layer
Output layer	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the point cloud file to use as output. One of: • Save to a Temporary File • Save to File…

Outputs

Label	Name	Type	Description
Converted	`OUTPUT`	[point cloud]	Output point cloud layer with a new CRS. Currently supported formats are `.LAS`, `.LAZ`, `.COPC.LAZ` and `.VPC`.

Python code

Algorithm ID: `pdal:assignprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Build virtual point cloud (VPC)

Creates a *virtual point cloud (VPC)* from input point cloud data.

If you leave the optional parameters unchecked, the VPC file will be built very quickly as the algorithm will only read metadata of input files. With any of the optional parameters set, the algorithm will read all points which can take some time.

Fig. 28.13: Generating Virtual Point Cloud with overview from a set of point cloud tiles

Vezi și:

Merge

Parameters

Label	Name	Type	Description
Input layers	`LAYERS`	[point cloud] [list]	Input point cloud layers to combine inside a virtual point cloud layer
Calculate boundary polygon	`BOUNDARY`	[boolean] Default: False	Set to True to show the exact boundaries of data (rather than just rectangular extent)
Calculate statistics	`STATISTICS`	[boolean] Default: False	Set to True to understand ranges of values of the various attributes
Build overview point cloud	`OVERVIEW`	[boolean] Default: False	Generates a single "thinned" point cloud of all the input data (using only every 1000th point from original data). The overview point cloud will be created next to the VPC file - for example, for `mydata.vpc`, the overview point cloud would be named `mydata-overview.copc.laz`.
Virtual point cloud	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the point cloud file to build the data into. One of: • Save to a Temporary File • Save to File…

Outputs

Label	Name	Type	Description
Virtual point cloud	`OUTPUT`	[raster]	Output point cloud layer combining all the input data, as a virtual file.

Python code

Algorithm ID: `pdal:virtualpointcloud`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Clip

Clips a point cloud layer by a polygon layer so that the resulting point cloud contains only points within the polygons.

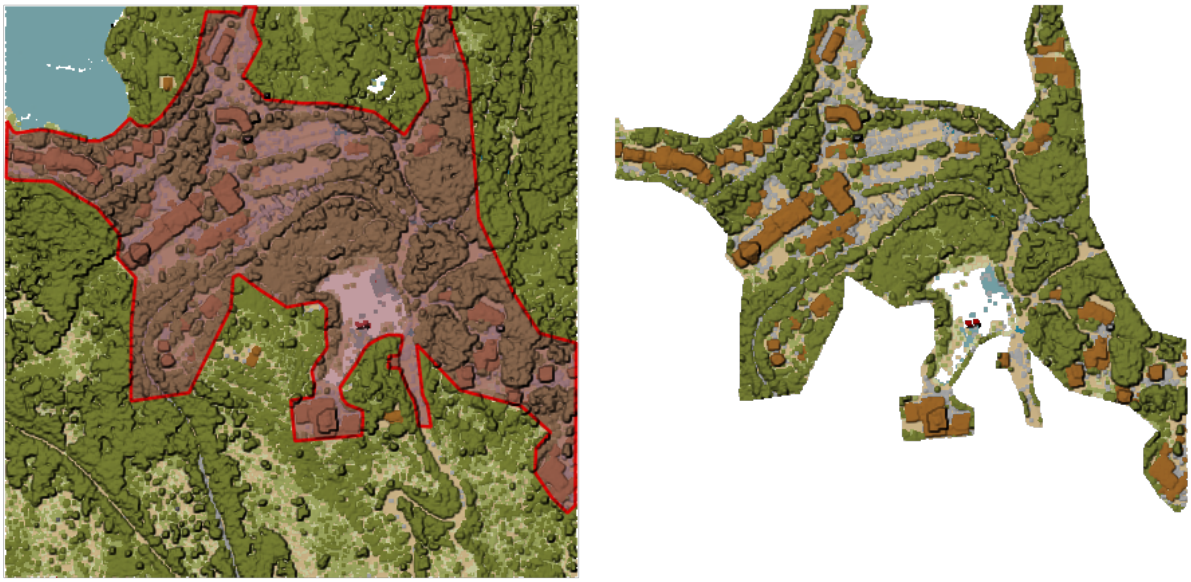


Fig. 28.14: Clipping an input point cloud layer with a polygon coverage

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to clip
Clipping polygons	`OVERLAY`	[vector: polygon]	Polygon vector layer to use as coverage for clipping the points
Clipped	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the point cloud file to export the clipped points to. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

Outputs

Label	Name	Type	Description
Clipped	`OUTPUT`	[raster]	Output point cloud whose features are the points within the coverage polygon layer.

Python code

Algorithm ID: `pdal:clip`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Create COPC

Creates the index for all the input point cloud files in a batch mode.

Parameters

Label	Name	Type	Description
Input layers	`LAYERS`	[point cloud] [list]	Input point cloud layers to create an index for
Output directory Optional	`OUTPUT`	[folder] Default: `[Skip output]`	Specify the folder to create the new files in. One of: • Skip Output • Save to a Temporary Directory • Save to Directory

Outputs

Label	Name	Type	Description
Output directory	`OUTPUT`	[folder]	Output folder containing point cloud layers with accompanying COPC index files.

Python code

Algorithm ID: `pdal:createcopc`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Information

Outputs basic metadata from an input point cloud file.

Example of output information:

```
LAS           1.4
point format  6
count         56736130
scale         0.001 0.001 0.001
offset        431749.999 5440919.999 968.898
extent        431250 5440420 424.266
              432249.999 5441419.999 1513.531
crs           ETRS89 / UTM zone 34N (N-E) (EPSG:3046)  (vertical CRS missing!)
units         horizontal=metre  vertical=unknown

Attributes:
- X floating 8
- Y floating 8
```

(continues on next page)

(continuare din pagina precedentă)

```
- Z floating 8
- Intensity unsigned 2
- ReturnNumber unsigned 1
- NumberOfReturns unsigned 1
- ScanDirectionFlag unsigned 1
- EdgeOfFlightLine unsigned 1
- Classification unsigned 1
- ScanAngleRank floating 4
- UserData unsigned 1
- PointSourceId unsigned 2
- GpsTime floating 8
- ScanChannel unsigned 1
- ClassFlags unsigned 1
```

Parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to extract metadata information from
Layer information	`OUTPUT`	[file] Default: `[Save to temporary file]`	Specify the file to store the metadata information. One of: • Save to a Temporary File • Save to File…

Outputs

Label	Name	Type	Description
Layer information	`OUTPUT`	[vector]	`HTML` file to store the metadata information.

Python code

Algorithm ID: `pdal:info`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Merge

Merges multiple point cloud files into a single one.

Vezi și:

Build virtual point cloud (VPC)

Parameters

Basic parameters

Label	Name	Type	Description
Input layers	`LAYERS`	[point cloud] [list]	Input point cloud layers to merge into a single one
Merged	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the output point cloud merging input files. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

Outputs

Label	Name	Type	Description
Merged	`OUTPUT`	[point cloud]	Output point cloud layer merging all the input files.

Python code

Algorithm ID: `pdal:merge`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Reproject

Reprojects a point cloud to a different Coordinate Reference System (CRS).

Vezi și:

Asignare proiecție

Parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to reproject to a different CRS
Target CRS	`CRS`	[crs]	The CRS to apply to the layer
Reprojected	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the reprojected point cloud file. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Coordinate operation Optional	`OPERATION`	[datum]	The *datum transformation* to use to reproject the data between the origin and target systems.

Outputs

Label	Name	Type	Description
REPROJECTED	`OUTPUT`	[point cloud]	Output point cloud layer in the target CRS.

Python code

Algorithm ID: `pdal:reproject`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Thin (by sampling radius)

Creates a thinned version of the point cloud by performing sampling by distance point (reduces the number of points within a certain radius).

Vezi și:

Thin (by skipping points)

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to create a thinned version from
Sampling radius (in map units)	`SAMPLING_RADIU`	[number] Default: 1.0	Distance within which points are sampled to a unique point
Thinned (by radius)	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the output point cloud with reduced points. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

Outputs

Label	Name	Type	Description
Thinned (by radius)	`OUTPUT`	[point cloud]	Output point cloud layer with reduced points.

Python code

Algorithm ID: `pdal:thinbyradius`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Thin (by skipping points)

Creates a thinned version of the point cloud by keeping only every N-th point (reduces the number of points by skipping nearby points).

Vezi și:

Thin (by sampling radius)

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to create a thinned version from
Number of points to skip	`POINTS_NUMBER`	[number] Default: 1	Keep only every N-th point in the input layer
Thinned (by decimation)	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the output point cloud with reduced points. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

Outputs

Label	Name	Type	Description
Thinned (by decimation)	`OUTPUT`	[point cloud]	Output point cloud layer with reduced points.

Python code

Algorithm ID: `pdal:thinbydecimate`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Tile

Creates tiles from input point cloud files, recommended for best performance (in display or analysis) with such datasets in QGIS.

Vezi și:

Build virtual point cloud (VPC), *Create COPC*

Parameters

Basic parameters

Label	Name	Type	Description
Input layers	`LAYERS`	[point cloud] [list]	Input point cloud layers to create tiles from
Tile length	`LENGTH`	[number] Default: 1000.0	Size of the edge of each generated tile
Output directory	`OUTPUT`	[folder] Default: `[Save to temporary folder]`	Specify the folder to store the generated tiles. One of: • Save to a Temporary Directory • Save to Directory

Advanced parameters

Label	Name	Type	Description
Assign CRS Optional	`CRS`	[crs]	The CRS to apply to the layer

Outputs

Label	Name	Type	Description
Output directory	`OUTPUT`	[folder]	Output folder containing the tiles generated from input files.

Python code

Algorithm ID: `pdal:tile`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

28.1.14 Point Cloud Extraction

`NEW in 3.32`

Boundary

Exports a polygon file containing point cloud layer boundary. It may contain holes and it may be a multi-part polygon.

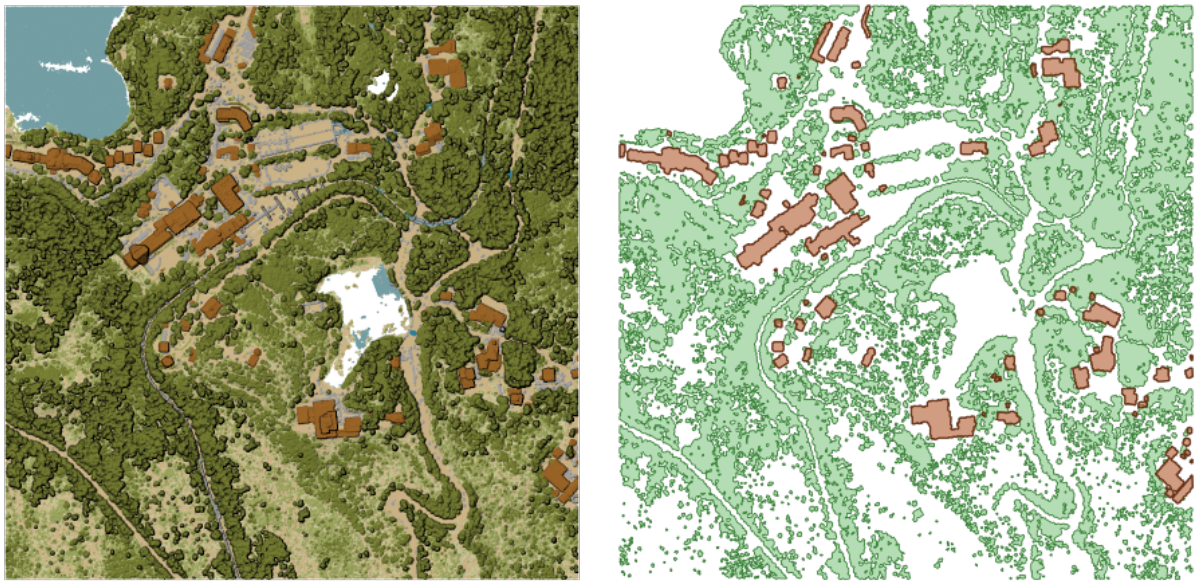


Fig. 28.15: Extracting high vegetation and building polygons from an input point cloud layer

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to calculate boundary for
Resolution of cells used to calculate boundary Optional	`RESOLUTION`	[number]	Resolution of cells used to calculate boundary
Minimal number of points in a cell to consider cell occupied Optional	`THRESHOLD`	[number]	Minimal number of points in a cell to consider cell occupied
Boundary	`OUTPUT`	[vector: polygon] Default: `[Save to temporary file]`	Specify the point cloud file to use as output. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

Outputs

Label	Name	Type	Description
Boundary	`OUTPUT`	[vector: polygon]	Output polygon vector layer with the point cloud boundary. Currently supported format is `.GPKG`.

Python code

Algorithm ID: `pdal:boundary`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Density

Exports a raster file based on the number of points within each raster cell - useful for quality checking of point cloud datasets.

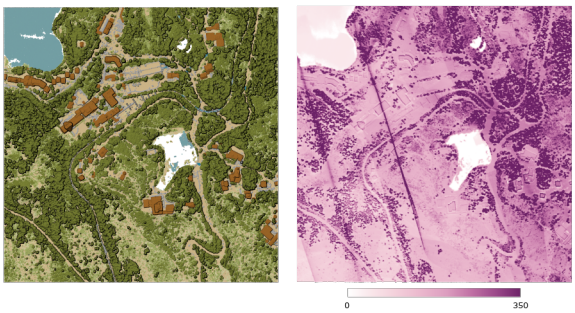


Fig. 28.16: Point density (number of points per 2x2 m) as a raster

Parameters

Basic parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to export
Resolution of the density raster	`RESOLUTION`	[number] Default: 1.0	Cell size of the output raster
Tile size for parallel runs	`TILE_SIZE`	[number] Default: 1000	Size of the tiles to split the data into for parallel runs
Density	`OUTPUT`	[raster] Default: `[Save to temporary file]`	Specify the raster file to export the data to. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Label	Name	Type	Description
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
X origin of a tile for parallel runs Optional	`ORIGIN_X`	[number]	
Y origin of a tile for parallel runs Optional	`ORIGIN_Y`	[number]	

Outputs

Label	Name	Type	Description
Density	`OUTPUT`	[raster]	Output raster layer with number of points within each cell. Currently supported format is `.TIF`.

Python code

Algorithm ID: `pdal:density`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

filter

Extracts point from the input point cloud which match PDAL expression and/or are inside of a cropping rectangle.

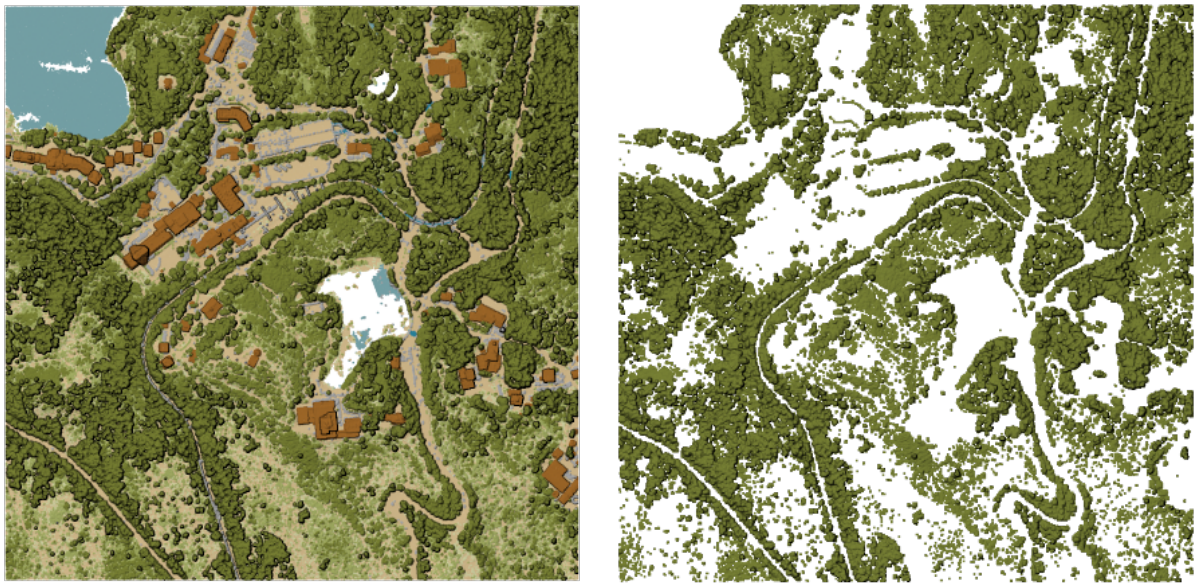


Fig. 28.17: Filtering of high vegetation class from an input point cloud layer

Parameters

Label	Name	Type	Description
Input layer	`INPUT`	[point cloud]	Input point cloud layer to export
Filter expression Optional	`FILTER_EXPRESS`	[expression]	A *PDAL expression* for selecting a subset of features in the point cloud data
Cropping extent Optional	`FILTER_EXTENT`	[extent]	A map extent for selecting a subset of features in the point cloud data Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Filtered	`OUTPUT`	[point cloud] Default: `[Save to temporary file]`	Specify the point cloud file to export the data to. One of: • Save to a Temporary File • Save to File…

Outputs

Label	Name	Type	Description
Filtered	`OUTPUT`	[point cloud]	Output point cloud layer with the filtered features of the input point cloud layer. Currently supported formats are `.LAS`, `.LAZ`, `.COPC.LAZ` and `.VPC`.

Python code

Algorithm ID: `pdal:filter`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

28.1.15 Analizarea rasterelor

Cell stack percent rank from value

Calculates the cell-wise percentrank value of a stack of rasters based on a single input value and writes them to an output raster.

At each cell location, the specified value is ranked among the respective values in the stack of all overlaid and sorted cell values from the input rasters. For values outside of the stack value distribution, the algorithm returns NoData because the value cannot be ranked among the cell values.

There are two methods for percentile calculation:

- Inclusive linear interpolation (PERCENTRANK.INC)
- Exclusive linear interpolation (PERCENTRANK.EXC)

The linear interpolation methods return the unique percent rank for different values. Both interpolation methods follow their counterpart methods implemented by LibreOffice or Microsoft Excel.

The output raster's extent and resolution is defined by a reference raster. Input raster layers that do not match the cell size of the reference raster layer will be resampled using nearest neighbor resampling. NoData values in any of the input layers will result in a NoData cell output if the „Ignore NoData values" parameter is not set. The output raster data type will always be `Float32`.

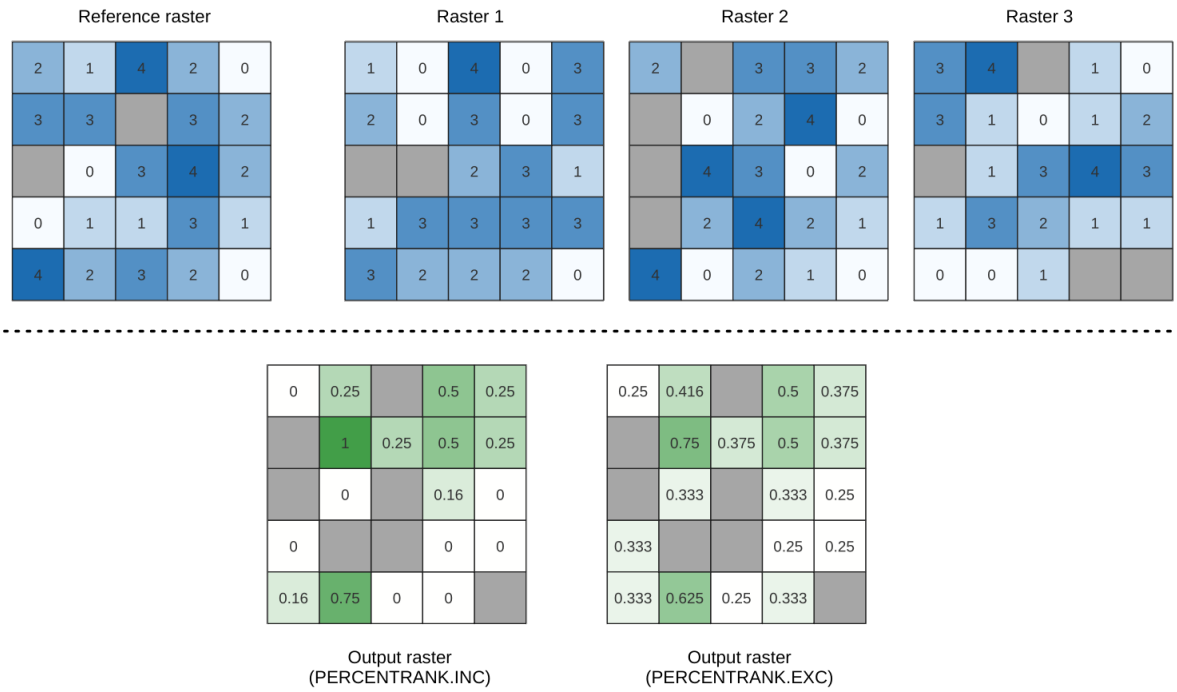


Fig. 28.18: Percent ranking Value = 1. `NoData` cells (grey) are ignored.

Vezi și:

Cell stack percentile, *Cell stack percentrank from raster layer*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	Raster layers to evaluate. If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters
Method	`METHOD`	[enumerare] Implicit: 0	Method for percentile calculation: • 0 — Interpolare liniară inclusivă (PERCENTRANK.INC) • 1 — Interpolare liniară exclusivă (PERCENTRANK.EXC)
Value	`VALUE`	[număr] Implicit: 10.0	Value to rank among the respective values in the stack of all overlaid and sorted cell values from the input rasters
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: True	If unchecked, any NoData cells in the input layers will result in a NoData cell in the output raster
Reference layer	`REFERENCE_LAYE`	[raster]	The reference layer for the output layer creation (extent, CRS, pixel dimensions)

continues on next page

Tabelul 28.44 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`OUTPUT_NODATA_`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:cellstackpercentrankfromvalue`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Cell stack percentile

Calculates the cell-wise percentile value of a stack of rasters and writes the results to an output raster. The percentile to return is determined by the percentile input value (ranges between 0 and 1). At each cell location, the specified percentile is obtained using the respective value from the stack of all overlaid and sorted cell values of the input rasters.

There are three methods for percentile calculation:

- Nearest rank: returns the value that is nearest to the specified percentile
- Inclusive linear interpolation (PERCENTRANK.INC)
- Exclusive linear interpolation (PERCENTRANK.EXC)

The linear interpolation methods return the unique values for different percentiles. Both interpolation methods follow their counterpart methods implemented by LibreOffice or Microsoft Excel.

The output raster's extent and resolution is defined by a reference raster. Input raster layers that do not match the cell size of the reference raster layer will be resampled using nearest neighbor resampling. NoData values in any of the input layers will result in a NoData cell output if the „Ignore NoData values" parameter is not set. The output raster data type will always be `Float32`.

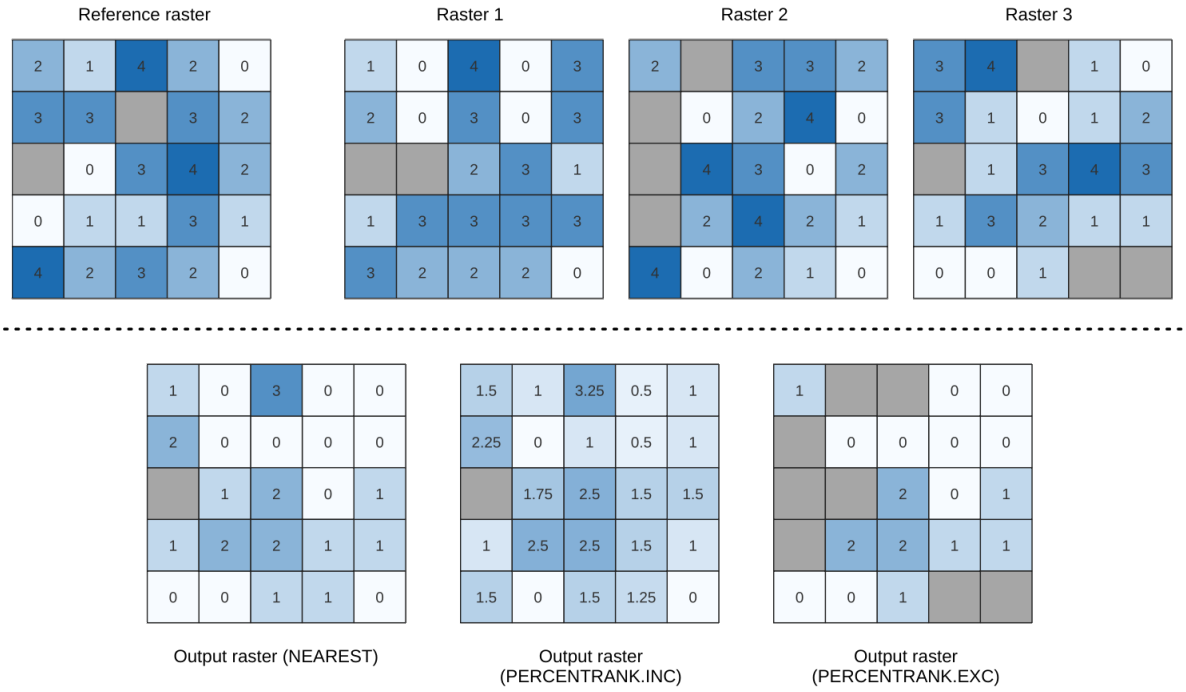


Fig. 28.19: Percentile = 0.25. `NoData` cells (grey) are ignored.

Vezi și:

Cell stack percentile, *Cell stack percentrank from raster layer*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	Raster layers to evaluate. If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters
Method	`METHOD`	[enumerare] Implicit: 0	Method for percentile calculation: • 0 — Cel mai apropiat rang: returnează valoarea care este cea mai apropiată de percentila specificată • 1 — Interpolare liniară inclusivă (PERCENTILE.INC) • 2 — Interpolare liniară exclusivă (PERCENTILE.EXC)
Percentile	`VALUE`	[număr] Implicit: 0.25	Value to rank among the respective values in the stack of all overlaid and sorted cell values from the input rasters. Between 0 and 1.
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: True	If unchecked, any NoData cells in the input layers will result in a NoData cell in the output raster
Reference layer	`REFERENCE_LAYE`	[raster]	The reference layer for the output layer creation (extent, CRS, pixel dimensions)
Output layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`OUTPUT_NODATA_`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:cellstackpercentile`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Cell stack percentrank from raster layer

Calculates the cell-wise percentrank value of a stack of rasters based on an input value raster and writes them to an output raster.

At each cell location, the current value of the value raster is ranked among the respective values in the stack of all overlaid and sorted cell values of the input rasters. For values outside of the the stack value distribution, the algorithm returns NoData because the value cannot be ranked among the cell values.

There are two methods for percentile calculation:

- Inclusive linear interpolation (PERCENTRANK.INC)
- Exclusive linear interpolation (PERCENTRANK.EXC)

The linear interpolation methods return the unique values for different percentiles. Both interpolation methods follow their counterpart methods implemented by LibreOffice or Microsoft Excel.

The output raster's extent and resolution is defined by a reference raster. Input raster layers that do not match the cell size of the reference raster layer will be resampled using nearest neighbor resampling. NoData values in any of the input layers will result in a NoData cell output if the „Ignore NoData values" parameter is not set. The output raster data type will always be `Float32`.

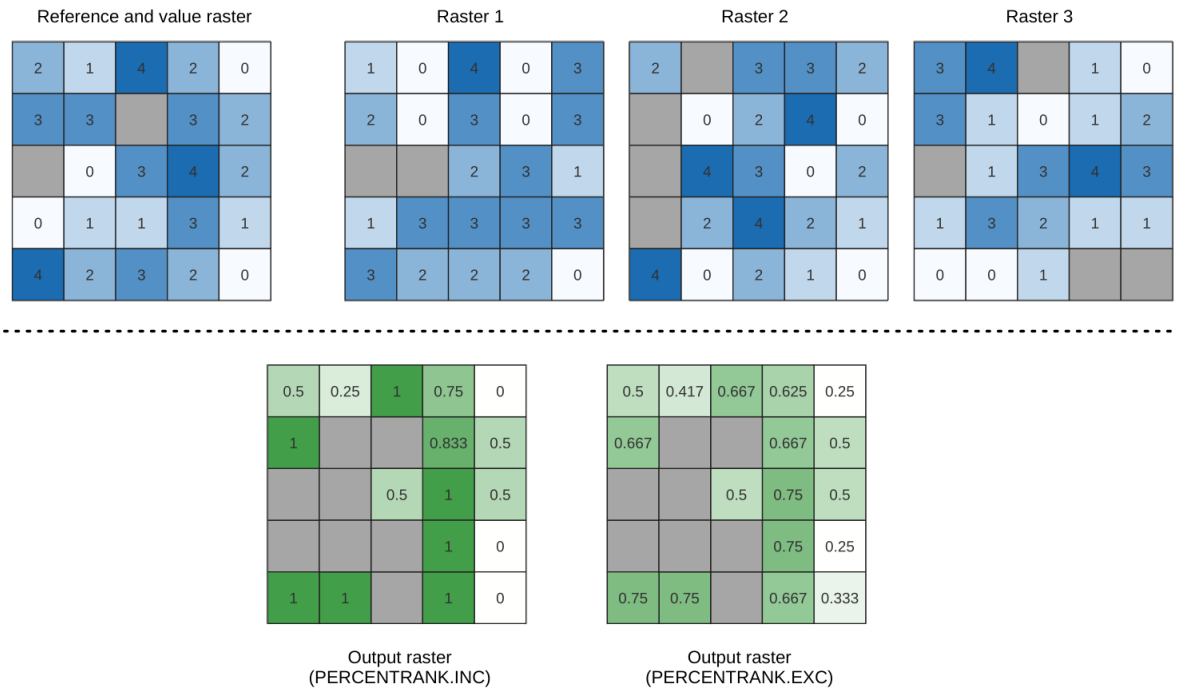


Fig. 28.20: Ranking the value raster layer cells. `NoData` cells (grey) are ignored.

Vezi și:

Cell stack percentile, *Cell stack percent rank from value*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	Raster layers to evaluate. If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters
Value raster layer	`INPUT_VALUE_RA`	[raster]	The layer to rank the values among the stack of all overlaid layers
Value raster band	`VALUE_RASTER_B`	[integer] Implicit: 1	Band of the „value raster layer” to compare to
Method	`METHOD`	[enumerare] Implicit: 0	Method for percentile calculation: • 0 — Interpolare liniară inclusivă (PERCENTRANK.INC) • 1 — Interpolare liniară exclusivă (PERCENTRANK.EXC)
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: True	If unchecked, any NoData cells in the input layers will result in a NoData cell in the output raster
Reference layer	`REFERENCE_LAYE`	[raster]	The reference layer for the output layer creation (extent, CRS, pixel dimensions)

continues on next page

Tabelul 28.48 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`OUTPUT_NODATA_`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:cellstackpercentrankfromrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Cell statistics

Computes per-cell statistics based on input raster layers and for each cell writes the resulting statistics to an output raster. At each cell location, the output value is defined as a function of all overlaid cell values of the input rasters.

By default, a NoData cell in ANY of the input layers will result in a NoData cell in the output raster. If the *Ignore NoData values* option is checked, then NoData inputs will be ignored in the statistic calculation. This may result in NoData output for locations where all cells are NoData.

The *Reference layer* parameter specifies an existing raster layer to use as a reference when creating the output raster. The output raster will have the same extent, CRS, and pixel dimensions as this layer.

Calculation details: Input raster layers that do not match the cell size of the reference raster layer will be resampled using `nearest neighbor resampling`. The output raster data type will be set to the most complex data type present in the input datasets except when using the functions `Mean`, `Standard deviation` and `Variance` (data type is always `Float32` or `Float64` depending on input float type) or `Count` and `Variety` (data type is always `Int32`).

- `Count`: The count statistic will always result in the number of cells without NoData values at the current cell location.
- `Median`: If the number of input layers is even, the median will be calculated as the arithmetic mean of the two middle values of the ordered cell input values.
- `Minority/Majority`: If no unique minority or majority could be found, the result is NoData, except all input cell values are equal.

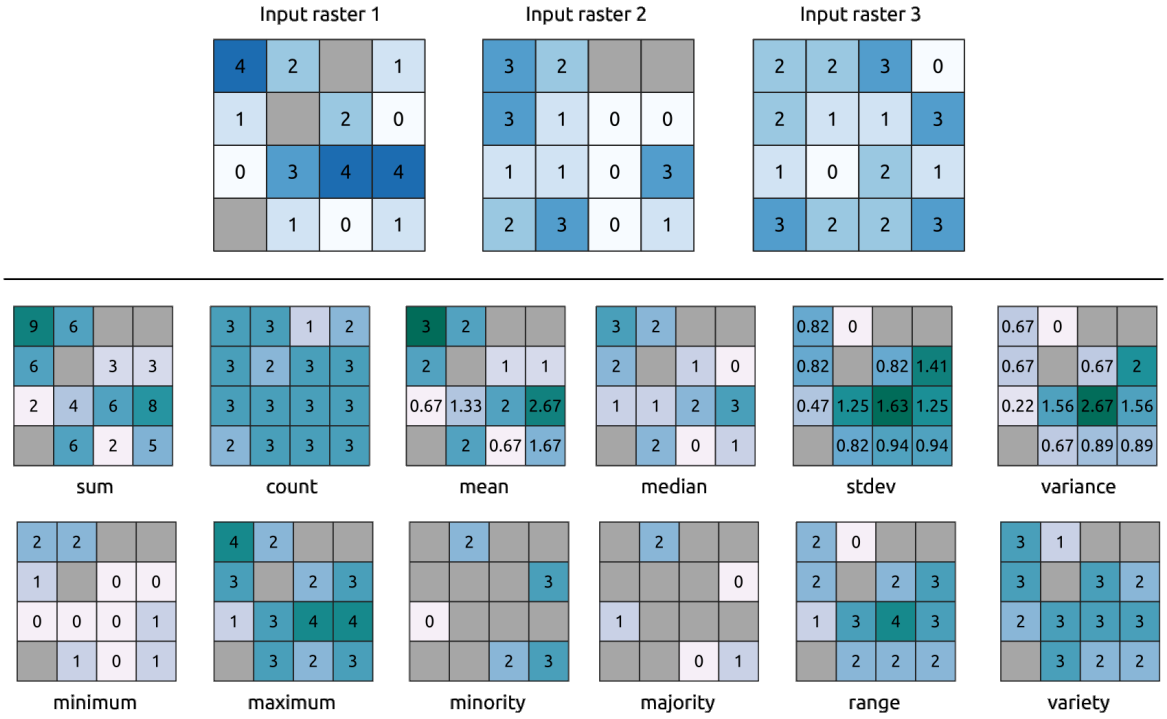


Fig. 28.21: Example with all the statistic functions. `NoData` cells (grey) are taken into account.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	Input raster layers
Statistic	`STATISTIC`	[enumerare] Implicit: 0	Available statistics. Options: • 0 — Sumă • 1 — Total • 2 — Medie • 3 — Median • 4 — Abatere standard • 5 — Varianță • 6 — Minimum • 7 — Maximum • 8 — Minoritar (cea mai mică valoare comună) • 9 — Majoritate (valoarea cea mai comună) • 10 — Interval (max - min) • 11 — Varietate (număr de valori unice)
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: True	Calculate statistics also for all cells stacks, ignoring NoData occurrence.
Reference layer	`REF_LAYER`	[raster]	The reference layer to create the output layer from (extent, CRS, pixel dimensions)
Output layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value Opţional	`OUTPUT_NO_DATA`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
CRS authority identifier	`CRS_AUTHID`	[crs]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Raster de ieșire	`OUTPUT`	[raster]	Output raster layer containing the result
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer

Cod Python

ID Algorithm: `nativ:cellstatistics`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Equal to frequency

Evaluates on a cell-by-cell basis the frequency (number of times) the values of an input stack of rasters are equal to the value of a value layer. The output raster extent and resolution are defined by the input raster layer and is always of `Int32` type.

If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters - use GDAL to use other bands in the analysis. The output NoData value can be set manually.

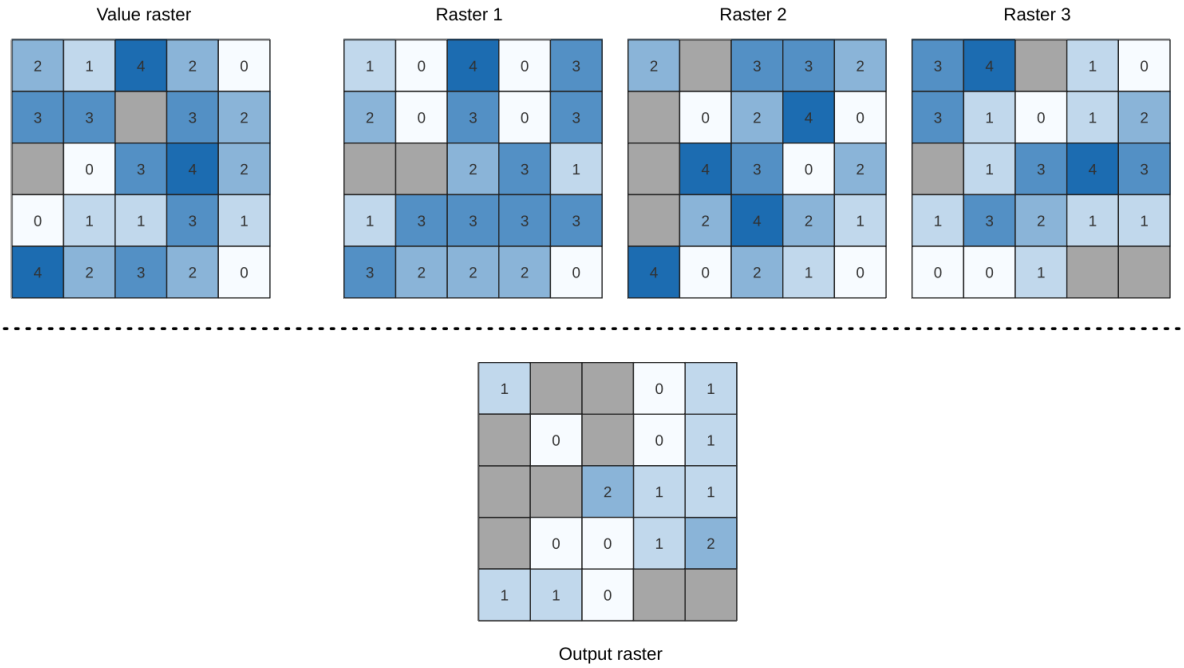


Fig. 28.22: For each cell in the output raster, the value represents the number of times that the corresponding cells in the list of rasters are the same as the value raster. `NoData` cells (grey) are taken into account.

Vezi și:

Greater than frequency, *Less than frequency*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input value raster	`INPUT_VALUE_RA`	[raster]	The input value layer serves as reference layer for the sample layers
Value raster band	`INPUT_VALUE_RA`	[raster band] Implicit: Prima bandă a stratului raster	Select the band you want to use as sample
Input raster layers	`INPUT_RASTERS`	[raster] [list]	Raster layers to evaluate. If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: False	If unchecked, any NoData cells in the value raster or the data layer stack will result in a NoData cell in the output raster
Output layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value Opţional	OUTPUT_NO_DATA	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	OUTPUT	[raster]	Output raster layer containing the result
CRS authority identifier	CRS_AUTHID	[string]	The coordinate reference system of the output raster layer
Extindere	EXTENT	[string]	The spatial extent of the output raster layer
Count of cells with equal value occurrences	FOUND_LOCATION	[număr]	
Height in pixels	HEIGHT_IN_PIXE	[număr]	The number of rows in the output raster layer
Total pixel count	TOTAL_PIXEL_CO	[integer]	The count of pixels in the output raster layer
Mean frequency at valid cell locations	MEAN_FREQUENCY	[număr]	
Count of value occurrences	OCCURRENCE_COU	[număr]	
Width in pixels	WIDTH_IN_PIXEL	[integer]	The number of columns in the output raster layer

Cod Python

ID Algorithm: `nativ:equaltofrequency`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fuzzify raster (gaussian membership)

Transforms an input raster to a fuzzified raster by assigning a membership value to each pixel, using a Gaussian membership function. Membership values range from 0 to 1. In the fuzzified raster, a value of 0 implies no membership of the defined fuzzy set, whereas a value of 1 means full membership. The gaussian membership function is defined as $\mu(x) = e^{-f1*(x-f2)^2}$, where *f1* is the spread and *f2* the midpoint.

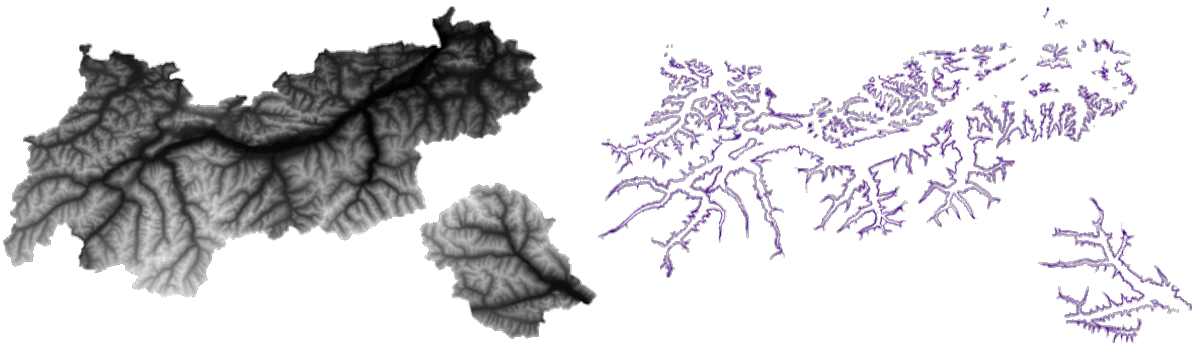


Fig. 28.23: Fuzzify raster example. Input raster source: Land Tirol - data.tirol.gv.at.

Vezi și:

Fuzzify raster (large membership) Fuzzify raster (linear membership), *Fuzzify raster (near membership)*, *Fuzzify raster (power membership)*, *Fuzzify raster (small membership)*

Parametri

Etichetă	Nume	Tipul	Descriere
Input Raster	`INPUT`	[raster]	Stratul raster de intrare
Band Number	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that you want to fuzzify.
Function midpoint	`FUZZYMIDPOINT`	[număr] Implicit: 10	Midpoint of the gaussian function
Function spread	`FUZZYSPREAD`	[număr] Implicit: 0.01	Spread of the gaussian function
Fuzzified raster	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Fuzzified raster	`OUTPUT`	[la fel ca intrarea]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[crs]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:fuzzifyrastergaussianmembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fuzzify raster (large membership)

Transforms an input raster to a fuzzified raster by assigning a membership value to each pixel, using a Large membership function. Membership values range from 0 to 1. In the fuzzified raster, a value of 0 implies no membership of the defined fuzzy set, whereas a value of 1 means full membership. The large membership function is defined as $\mu(x) = \frac{1}{1 + \left(\frac{x}{f2}\right)^{-f1}}$, where *f1* is the spread and *f2* the midpoint.

Vezi și:

Fuzzify raster (gaussian membership), *Fuzzify raster (linear membership)*, *Fuzzify raster (near membership)*, *Fuzzify raster (power membership)*, *Fuzzify raster (small membership)*

Parametri

Etichetă	Nume	Tipul	Descriere
Input Raster	`INPUT`	[raster]	Stratul raster de intrare
Band Number	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that you want to fuzzify.
Function midpoint	`FUZZYMIDPOINT`	[număr] Implicit: 50	Midpoint of the large function
Function spread	`FUZZYSPREAD`	[număr] Implicit: 5	Spread of the large function
Fuzzified raster	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Fuzzified raster	`OUTPUT`	[la fel ca intrarea]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[crs]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:fuzzifyrasterlargemembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fuzzify raster (linear membership)

Transforms an input raster to a fuzzified raster by assigning a membership value to each pixel, using a Linear membership function. Membership values range from 0 to 1. In the fuzzified raster, a value of 0 implies no membership of the defined fuzzy set, whereas a value of 1 means full membership. The linear function is defined as $\mu(X)\begin{cases}0 & x \leq a \\ \frac{x-a}{b-a} & a < x < b \\ 1 & x \geq b\end{cases}$, where a is the low bound and b the high bound. This equation assigns membership values using a linear transformation for pixel values between the low and high bounds. Pixels values smaller than the low bound are given 0 membership whereas pixel values greater than the high bound are given 1 membership.

Vezi și:

Fuzzify raster (gaussian membership), *Fuzzify raster (large membership)*, *Fuzzify raster (near membership)*, *Fuzzify raster (power membership)*, *Fuzzify raster (small membership)*

Parametri

Etichetă	Nume	Tipul	Descriere
Input Raster	`INPUT`	[raster]	Stratul raster de intrare
Band Number	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that you want to fuzzify.
Low fuzzy membership bound	`FUZZYLOWBOUND`	[număr] Implicit: 0	Low bound of the linear function
High fuzzy membership bound	`FUZZYHIGHBOUND`	[număr] Implicit: 1	High bound of the linear function
Fuzzified raster	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Fuzzified raster	`OUTPUT`	[la fel ca intrarea]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[crs]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:fuzzifyrasterlinearmembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fuzzify raster (near membership)

Transforms an input raster to a fuzzified raster by assigning a membership value to each pixel, using a Near membership function. Membership values range from 0 to 1. In the fuzzified raster, a value of 0 implies no membership of the defined fuzzy set, whereas a value of 1 means full membership. The near membership function is defined as $\mu(x) = \frac{1}{1 + f1 * (x - f2)^2}$, where *f1* is the spread and *f2* the midpoint.

Vezi și:

Fuzzify raster (gaussian membership), *Fuzzify raster (large membership)*, *Fuzzify raster (linear membership)*, *Fuzzify raster (power membership)*, *Fuzzify raster (small membership)*

Parametri

Etichetă	Nume	Tipul	Descriere
Input Raster	`INPUT`	[raster]	Stratul raster de intrare
Band Number	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that you want to fuzzify.
Function midpoint	`FUZZYMIDPOINT`	[număr] Implicit: 50	Midpoint of the near function
Function spread	`FUZZYSPREAD`	[număr] Implicit: 0.01	Spread of the near function
Fuzzified raster	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Fuzzified raster	`OUTPUT`	[la fel ca intrarea]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[crs]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:fuzzifyrasternearmembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fuzzify raster (power membership)

Transforms an input raster to a fuzzified raster by assigning a membership value to each pixel, using a Power membership function. Membership values range from 0 to 1. In the fuzzified raster, a value of 0 implies no membership of the defined fuzzy set, whereas a value of 1 means full membership. The power function is defined as $\mu(x)\begin{cases}0 & x \le a\\ \left(\frac{x-a}{b-a}\right)^{f1} & a < x < b\\ 1 & x \ge b\end{cases}$, where a is the low bound, b is the high bound, and $f1$ the exponent. This equation assigns membership values using the power transformation for pixel values between the low and high bounds. Pixels values smaller than the low bound are given 0 membership whereas pixel values greater than the high bound are given 1 membership.

Vezi și:

Fuzzify raster (gaussian membership), *Fuzzify raster (large membership)*, *Fuzzify raster (linear membership)*, *Fuzzify raster (near membership)*, *Fuzzify raster (small membership)*

Parametri

Etichetă	Nume	Tipul	Descriere
Input Raster	`INPUT`	[raster]	Stratul raster de intrare
Band Number	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that you want to fuzzify.
Low fuzzy membership bound	`FUZZYLOWBOUND`	[număr] Implicit: 0	Low bound of the power function
High fuzzy membership bound	`FUZZYHIGHBOUND`	[număr] Implicit: 1	High bound of the power function
High fuzzy membership bound	`FUZZYEXPONENT`	[număr] Implicit: 2	Exponent of the power function
Fuzzified raster	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Fuzzified raster	`OUTPUT`	[la fel ca intrarea]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[crs]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:fuzzifyrasterpowermembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fuzzify raster (small membership)

Transforms an input raster to a fuzzified raster by assigning a membership value to each pixel, using a Small membership function. Membership values range from 0 to 1. In the fuzzified raster, a value of 0 implies no membership of the defined fuzzy set, whereas a value of 1 means full membership. The small membership function is defined as $\mu(x) = \frac{1}{1 + \left(\frac{x}{f2}\right)^{f1}}$, where *f1* is the spread and *f2* the midpoint.

Vezi și:

Fuzzify raster (gaussian membership), *Fuzzify raster (large membership)* *Fuzzify raster (linear membership)*, *Fuzzify raster (near membership)*, *Fuzzify raster (power membership)*

Parametri

Etichetă	Nume	Tipul	Descriere
Input Raster	`INPUT`	[raster]	Stratul raster de intrare
Band Number	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that you want to fuzzify.
Function midpoint	`FUZZYMIDPOINT`	[număr] Implicit: 50	Midpoint of the small function
Function spread	`FUZZYSPREAD`	[număr] Implicit: 5	Spread of the small function
Fuzzified raster	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Fuzzified raster	`OUTPUT`	[la fel ca intrarea]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[crs]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:fuzzifyrastersmallmembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Greater than frequency

Evaluates on a cell-by-cell basis the frequency (number of times) the values of an input stack of rasters are equal to the value of a value raster. The output raster extent and resolution is defined by the input raster layer and is always of `Int32` type.

If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters - use GDAL to use other bands in the analysis. The output NoData value can be set manually.

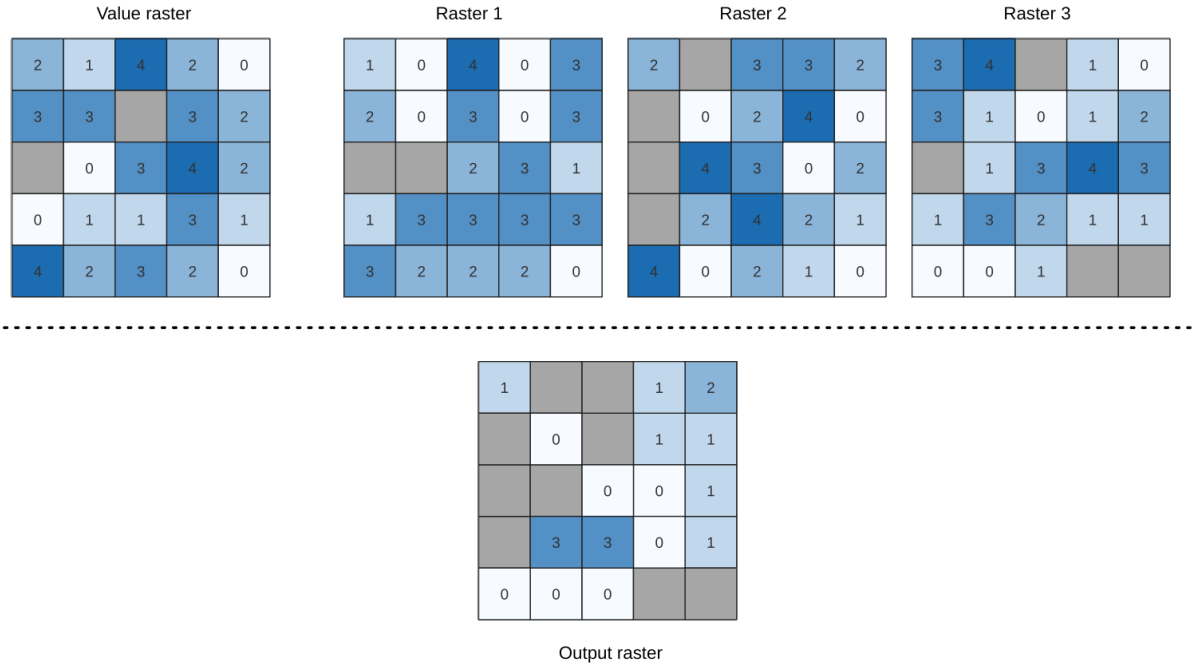


Fig. 28.24: For each cell in the output raster, the value represents the number of times that the corresponding cells in the list of rasters are greater than the value raster. `NoData` cells (grey) are taken into account.

Vezi şi:

Equal to frequency, *Less than frequency*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input value raster	`INPUT_VALUE_RA`	[raster]	The input value layer serves as reference layer for the sample layers
Value raster band	`INPUT_VALUE_RA`	[raster band] Implicit: Prima bandă a stratului raster	Select the band you want to use as sample
Input raster layers	`INPUT_RASTERS`	[raster] [list]	Raster layers to evaluate. If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters

continues on next page

Tabelul 28.54 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: False	If unchecked, any NoData cells in the value raster or the data layer stack will result in a NoData cell in the output raster
Output layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value Opţional	`OUTPUT_NO_DATA`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Count of cells with equal value occurrences	`FOUND_LOCATION`	[număr]	
Height in pixels	`HEIGHT_IN_PIXE`	[număr]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer
Mean frequency at valid cell locations	`MEAN_FREQUENCY`	[număr]	
Count of value occurrences	`OCCURRENCE_COU`	[număr]	
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer

Cod Python

ID Algorithm: `nativ:greaterthanfrequency`

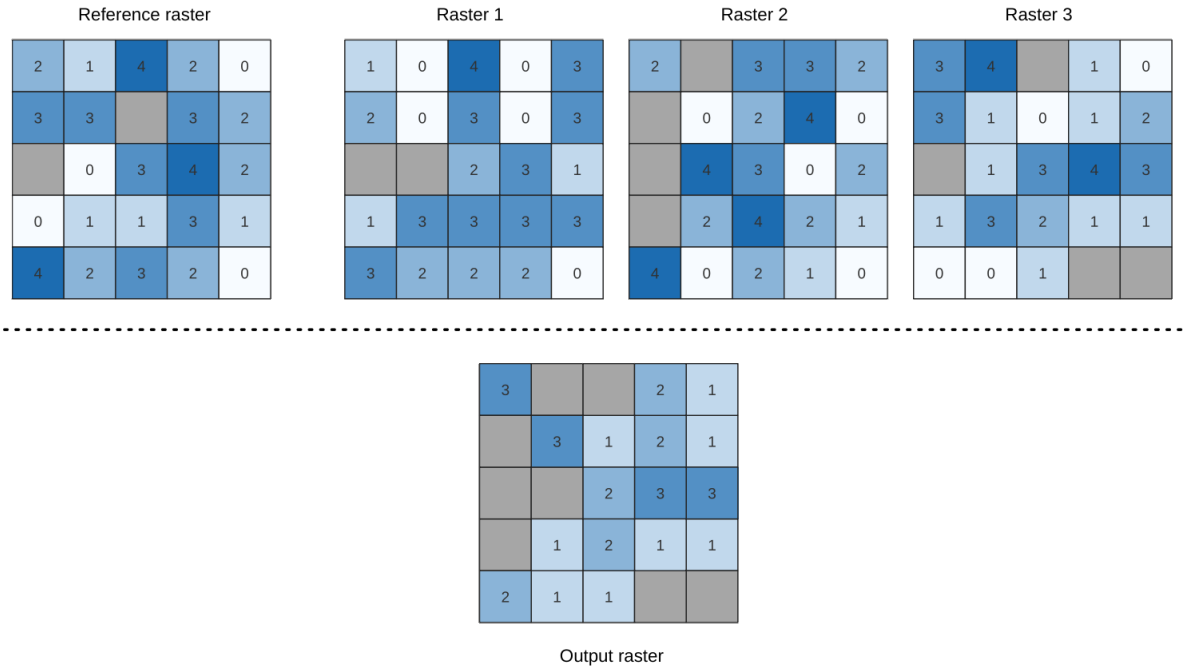
```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Highest position in raster stack

Evaluates on a cell-by-cell basis the position of the raster with the highest value in a stack of rasters. Position counts start with 1 and range to the total number of input rasters. The order of the input rasters is relevant for the algorithm. If multiple rasters feature the highest value, the first raster will be used for the position value.

If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters - use GDAL to use other bands in the analysis. Any NoData cells in the raster layer stack will result in a NoData cell in the output raster unless the „ignore NoData" parameter is checked. The output NoData value can be set manually. The output rasters extent and resolution is defined by a reference raster layer and is always of `Int32` type.



Vezi și:

Lowest position in raster stack

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input raster layers	`INPUT_RASTERS`	[raster] [list]	List of raster layers to compare with
Reference layer	`REFERENCE_LAYE`	[raster]	The reference layer for the output layer creation (extent, CRS, pixel dimensions)
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: False	If unchecked, any NoData cells in the data layer stack will result in a NoData cell in the output raster

continues on next page

Tabelul 28.56 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster containing the result. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`OUTPUT_NODATA_`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:highestpositioninrasterstack`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Less than frequency

Evaluates on a cell-by-cell basis the frequency (number of times) the values of an input stack of rasters are less than the value of a value raster. The output raster extent and resolution is defined by the input raster layer and is always of `Int32` type.

If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters - use GDAL to use other bands in the analysis. The output NoData value can be set manually.

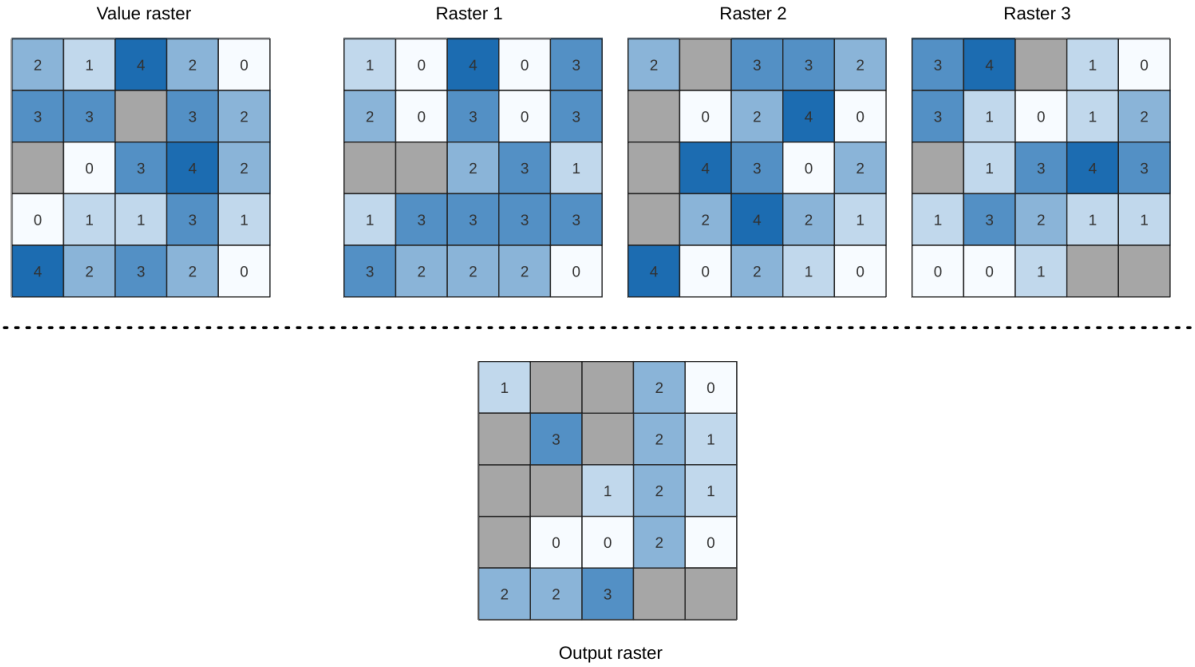


Fig. 28.25: For each cell in the output raster, the value represents the number of times that the corresponding cells in the list of rasters are less than the value raster. `NoData` cells (grey) are taken into account.

Vezi şi:

Equal to frequency, *Greater than frequency*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input value raster	`INPUT_VALUE_RA`	[raster]	The input value layer serves as reference layer for the sample layers
Value raster band	`INPUT_VALUE_RA`	[raster band] Implicit: Prima bandă a stratului raster	Select the band you want to use as sample
Input raster layers	`INPUT_RASTERS`	[raster] [list]	Raster layers to evaluate. If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters

continues on next page

Tabelul 28.58 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: False	If unchecked, any NoData cells in the value raster or the data layer stack will result in a NoData cell in the output raster
Output layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value Opţional	`OUTPUT_NO_DATA`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Count of cells with equal value occurrences	`FOUND_LOCATION`	[număr]	
Height in pixels	`HEIGHT_IN_PIXE`	[număr]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer
Mean frequency at valid cell locations	`MEAN_FREQUENCY`	[număr]	
Count of value occurrences	`OCCURRENCE_COU`	[număr]	
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer

Cod Python

ID Algorithm: `nativ:lessthanfrequency`

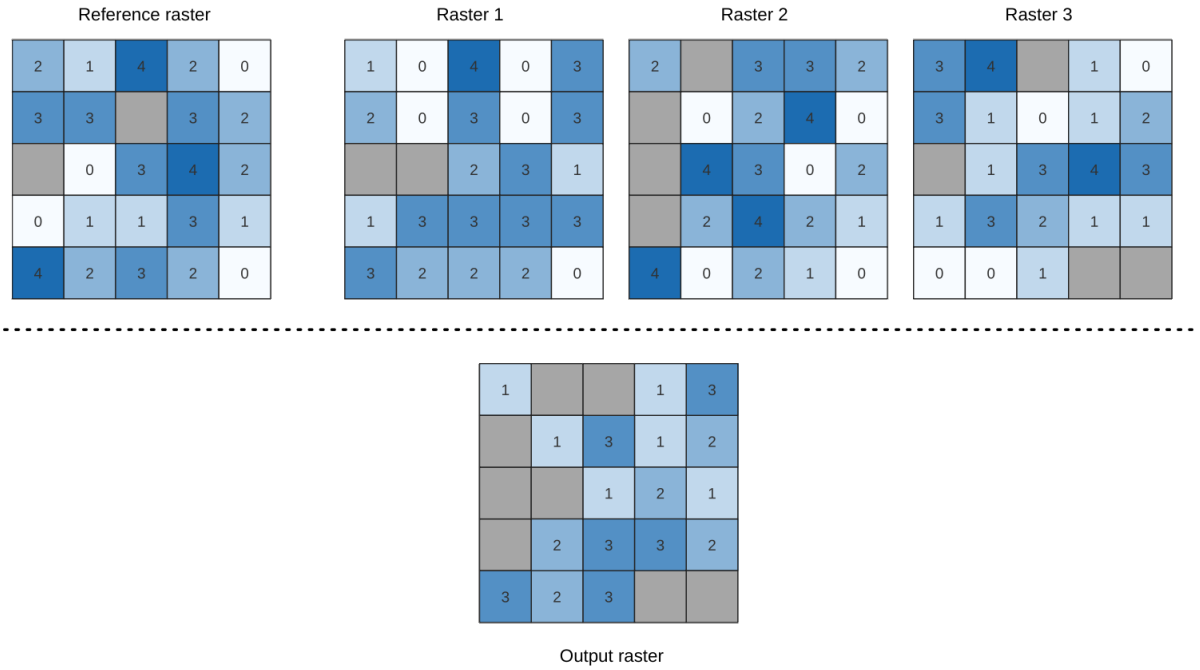
```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Lowest position in raster stack

Evaluates on a cell-by-cell basis the position of the raster with the lowest value in a stack of rasters. Position counts start with 1 and range to the total number of input rasters. The order of the input rasters is relevant for the algorithm. If multiple rasters feature the lowest value, the first raster will be used for the position value.

If multiband rasters are used in the data raster stack, the algorithm will always perform the analysis on the first band of the rasters - use GDAL to use other bands in the analysis. Any NoData cells in the raster layer stack will result in a NoData cell in the output raster unless the „ignore NoData" parameter is checked. The output NoData value can be set manually. The output rasters extent and resolution is defined by a reference raster layer and is always of `Int32` type.



Vezi și:

Highest position in raster stack

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input raster layers	`INPUT_RASTERS`	[raster] [list]	List of raster layers to compare with
Reference layer	`REFERENCE_LAYE`	[raster]	The reference layer for the output layer creation (extent, CRS, pixel dimensions)
Ignore NoData values	`IGNORE_NODATA`	[boolean] Implicit: False	If unchecked, any NoData cells in the data layer stack will result in a NoData cell in the output raster

continues on next page

Tabelul 28.60 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster containing the result. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`OUTPUT_NODATA_`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Output raster layer containing the result
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer

Cod Python

ID Algorithm: `nativ:lowestpositioninrasterstack`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster boolean AND

Calculates the boolean `AND` for a set of input rasters. If all of the input rasters have a non-zero value for a pixel, that pixel will be set to `1` in the output raster. If any of the input rasters have `0` values for the pixel it will be set to `0` in the output raster.

The reference layer parameter specifies an existing raster layer to use as a reference when creating the output raster. The output raster will have the same extent, CRS, and pixel dimensions as this layer.

By default, a NoData pixel in ANY of the input layers will result in a NoData pixel in the output raster. If the *Treat NoData values as false* option is checked, then NoData inputs will be treated the same as a `0` input value.

Vezi și:

Raster boolean OR

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	List of input raster layers
Reference layer	`REF_LAYER`	[raster]	The reference layer to create the output layer from (extent, CRS, pixel dimensions)
Treat NoData values as false	`NODATA_AS_FALS`	[boolean] Implicit: False	Treat NoData values in the input files as 0 when performing the operation
Output layer	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster containing the result. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`NO_DATA`	[număr] Implicit: -9999.0	Value to use for NoData in the output layer

continues on next page

Tabelul 28.63 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tipul datelor de ieșire	DATA_TYPE	[enumerare] Implicit: 5	Output raster data type. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Extindere	EXTENT	[string]	The spatial extent of the output raster layer
CRS authority identifier	CRS_AUTHID	[crs]	The coordinate reference system of the output raster layer
Width in pixels	WIDTH_IN_PIXEL	[integer]	The number of columns in the output raster layer
Height in pixels	HEIGHT_IN_PIXE	[integer]	The number of rows in the output raster layer
Total pixel count	TOTAL_PIXEL_CO	[integer]	The count of pixels in the output raster layer
NoData pixel count	NODATA_PIXEL_C	[integer]	The count of NoData pixels in the output raster layer
True pixel count	TRUE_PIXEL_COU	[integer]	The count of True pixels (value = 1) in the output raster layer
False pixel count	FALSE_PIXEL_CO	[integer]	The count of False pixels (value = 0) in the output raster layer
Output layer	OUTPUT	[raster]	Output raster layer containing the result

Cod Python

ID Algorithm: `nativ:rasterbooleanand`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster boolean OR

Calculates the boolean `OR` for a set of input rasters. If all of the input rasters have a zero value for a pixel, that pixel will be set to `0` in the output raster. If any of the input rasters have `1` values for the pixel it will be set to `1` in the output raster.

The reference layer parameter specifies an existing raster layer to use as a reference when creating the output raster. The output raster will have the same extent, CRS, and pixel dimensions as this layer.

By default, a NoData pixel in ANY of the input layers will result in a NoData pixel in the output raster. If the *Treat NoData values as false* option is checked, then NoData inputs will be treated the same as a `0` input value.

Vezi și:

Raster boolean AND

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	List of input raster layers
Reference layer	`REF_LAYER`	[raster]	The reference layer to create the output layer from (extent, CRS, pixel dimensions)
Treat NoData values as false	`NODATA_AS_FALS`	[boolean] Implicit: False	Treat NoData values in the input files as 0 when performing the operation
Output layer	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster containing the result. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

continues on next page

Tabelul 28.66 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	NO_DATA	[număr] Implicit: -9999.0	Value to use for NoData in the output layer
Tipul datelor de ieşire	DATA_TYPE	[enumerare] Implicit: 5	Output raster data type. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Extindere	EXTENT	[string]	The spatial extent of the output raster layer
CRS authority identifier	CRS_AUTHID	[crs]	The coordinate reference system of the output raster layer
Width in pixels	WIDTH_IN_PIXEL	[integer]	The number of columns in the output raster layer
Height in pixels	HEIGHT_IN_PIXE	[integer]	The number of rows in the output raster layer
Total pixel count	TOTAL_PIXEL_CO	[integer]	The count of pixels in the output raster layer
NoData pixel count	NODATA_PIXEL_C	[integer]	The count of NoData pixels in the output raster layer
True pixel count	TRUE_PIXEL_COU	[integer]	The count of True pixels (value = 1) in the output raster layer
False pixel count	FALSE_PIXEL_CO	[integer]	The count of False pixels (value = 0) in the output raster layer
Output layer	OUTPUT	[raster]	Output raster layer containing the result

Cod Python

ID Algorithm: `nativ:rasterbooleanor`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Calculator raster

Performs algebraic operations using raster layers.

The resulting layer will have its values computed according to an expression. The expression can contain numerical values, operators and references to any of the layers in the current project.

Vezi și:

Raster calculator (virtual), *Calculator raster*, *Calculatorul Raster*

Parametri

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	List of input raster layers
Expression	`EXPRESSION`	[expression]	*Raster-based expression* that will be used to calculate the output raster layer.
Output extent Opţional	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. If the extent is not specified, the minimum extent that covers all the selected reference layers will be used. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Output cell size (leave empty to set automatically) Opţional	`CELLSIZE`	[număr]	Cell size of the output raster layer. If the cell size is not specified, the minimum cell size of the selected reference layer(s) will be used. The cell size will be the same for the X and Y axes.
Output CRS Opţional	`CRS`	[crs]	CRS of the output raster layer. If the output CRS is not specified, the CRS of the first reference layer will be used.

continues on next page

Tabelul 28.68 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Calculated	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Calculated	`OUTPUT`	[raster]	Output raster file with the calculated values.

Cod Python

Algorithm ID: `native:rastercalc`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster calculator (virtual)

Performs algebraic operations using raster layers and generates in-memory result.

The resulting layer will have its values computed according to an expression. The expression can contain numerical values, operators and references to any of the layers in the current project.

A virtual raster layer is a raster layer defined by its URI and whose pixels are calculated on-the-fly. It's not a new file on disk; the virtual layer is still connected to the rasters used in the calculation meaning that deleting or moving these rasters would break it. A *Layer name* can be provided, otherwise the calculation expression is used as such. Removing the virtual layer from the project deletes it, and it can be made persistent in file using the layer *Export* ► *Save as…* contextual menu.

Vezi și:

Calculator raster, *Calculator raster*, *Calculatorul Raster*

Parametri

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	List of input raster layers
Expression	`EXPRESSION`	[expression]	*Raster-based expression* that will be used to calculate the output raster layer.

continues on next page

Tabelul 28.69 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output extent Opţional	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. If the extent is not specified, the minimum extent that covers all the selected reference layers will be used. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Output cell size (leave empty to set automatically) Opţional	`CELLSIZE`	[număr]	Cell size of the output raster layer. If the cell size is not specified, the minimum cell size of the selected reference layer(s) will be used. The cell size will be the same for the X and Y axes.
Output CRS Opţional	`CRS`	[crs]	CRS of the output raster layer. If the output CRS is not specified, the CRS of the first reference layer will be used.
Output layer name Opţional	`LAYER_NAME`	[string]	The name to assign to the generated layer. If not set, the text of the calculation expression is used.

Rezultat

Etichetă	Nume	Tipul	Descriere
Calculated	`OUTPUT`	[raster]	Output virtual raster layer with the calculated values.

Cod Python

Algorithm ID: `native:virtualrastercalc`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster layer properties

Returns basic properties of the given raster layer, including the extent, size in pixels and dimensions of pixels (in map units), number of bands, and NoData value.

This algorithm is intended for use as a means of extracting these useful properties to use as the input values to other algorithms in a model - e.g. to allow to pass an existing raster's pixel sizes over to a GDAL raster algorithm.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Numărul benzii Opţional	`BAND`	[raster band] Implicit: Nesetat	Whether to also return properties of a specific band. If a band is specified, the noData value for the selected band is also returned.

Rezultat

Etichetă	Nume	Tipul	Descriere
Number of bands in raster	`BAND_COUNT`	[număr]	The number of bands in the raster
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The raster layer extent in the CRS
Band has a NoData value set	`HAS_NODATA_VAL`	[Boolean]	Indicates whether the raster layer has a value set for NoData pixels in the selected band
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of columns in the raster layer
Band NoData value	`NODATA_VALUE`	[număr]	The value (if set) of the NoData pixels in the selected band
Pixel size (height) in map units	`PIXEL_HEIGHT`	[integer]	Vertical size in map units of the pixel
Pixel size (width) in map units	`PIXEL_WIDTH`	[integer]	Horizontal size in map units of the pixel
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of rows in the raster layer
Maximum x-coordinate	`X_MAX`	[număr]	
Minimum x-coordinate	`X_MIN`	[număr]	
Maximum y-coordinate	`Y_MAX`	[număr]	
Minimum y-coordinate	`Y_MIN`	[număr]	

Cod Python

ID Algorithm: `nativ:rasterlayerproperties`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster layer statistics

Calculates basic statistics from the values in a given band of the raster layer. The output is loaded in the *Processing* ► *Results viewer* menu.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Numărul benzii	`BAND`	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose the band you want to get statistics for.
Statistics	`OUTPUT_HTML_FI`	[html] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Maximum value	`MAX`	[număr]	
Mean value	`MEAN`	[număr]	
Minimum value	`MIN`	[număr]	

continues on next page

Tabelul 28.70 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Statistics	OUTPUT_HTML_FI	[html]	The output file contains the following information: • Analyzed file: path of the raster layer • Minimum value: minimum value of the raster • Maximum value: maximum value of the raster • Range: difference between the maximum and minimum values • Sum: total sum of the values • Mean value: mean of the values • Standard deviation: standard deviation of the values • Sum of the squares: sum of the squared differences of each observation from the overall mean
Range	RANGE	[număr]	
Standard deviation	STD_DEV	[număr]	
Sum	SUM	[număr]	
Sum of the squares	SUM_OF_SQUARES	[număr]	

Cod Python

Algorithm ID: `native:rasterlayerstatistics`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster layer unique values report

Returns the count and area of each unique value in a given raster layer.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	INPUT	[raster]	Stratul raster de intrare
Numărul benzii	BAND	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose the band you want to get statistics for.

continues on next page

Tabelul 28.71 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Unique values report	`OUTPUT_HTML_FI`	[file] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…
Unique values table	`OUTPUT_TABLE`	[tabelă] Implicit: `[Omitere rezultat]`	Specification of the table for unique values: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
CRS authority identifier	`CRS_AUTHID`	[string]	The coordinate reference system of the output raster layer
Extindere	`EXTENT`	[string]	The spatial extent of the output raster layer
Height in pixels	`HEIGHT_IN_PIXE`	[integer]	The number of rows in the output raster layer
NoData pixel count	`NODATA_PIXEL_C`	[număr]	The number of NoData pixels in the output raster layer
Total pixel count	`TOTAL_PIXEL_CO`	[integer]	The count of pixels in the output raster layer
Unique values report	`OUTPUT_HTML_FI`	[html]	The output HTML file contains the following information: • Analyzed file: the path of the raster layer • Extent: xmin, ymin, xmax, ymax coordinates of the extent • Projection: projection of the layer • Width in pixels: number of columns and pixel width size • Height in pixels: number of rows and pixel width size • Total pixel count: count of all the pixels • NoData pixel count: count of pixels with NoData value
Unique values table	`OUTPUT_TABLE`	[tabelă]	A table with three columns: • *value*: pixel value • *count*: count of pixels with this value • m^2: total area in square meters of pixels with this value.
Width in pixels	`WIDTH_IN_PIXEL`	[integer]	The number of columns in the output raster layer

Cod Python

ID Algorithm: `nativ:rasterlayeruniquevaluesreport`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster layer zonal statistics

Calculates statistics for a raster layer's values, categorized by zones defined in another raster layer.

Vezi și:

Zonal statistics

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[raster]	Stratul raster de intrare
Numărul benzii	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband choose the band for which you want to calculate the statistics.
Zones layer	`ZONES`	[raster]	Raster layer defining zones. Zones are given by contiguous pixels having the same pixel value.
Zones band number	`ZONES_BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that defines the zones
Statistics	`OUTPUT_TABLE`	[tabelă] Implicit: `[Creare Strat temporar]`	Specification of the output report. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Reference layer Opţional	REF_LAYER	[enumerare] Implicit: 0	Raster layer used to calculate the centroids that will be used as reference when determining the zones in the output layer. One of: • 0 — Strat de intrare: zonele sunt determinate prin eșantionarea valorii stratului raster al zonelor la centroidul fiecărui pixel din stratul raster sursă • 1 — Stratul zonelor: stratul raster de intrare va fi eșantionat la centroidul fiecărui pixel din stratul raster al zonelor

Rezultat

Etichetă	Nume	Tipul	Descriere
CRS authority identifier	CRS_AUTHID	[string]	The coordinate reference system of the output raster layer
Extindere	EXTENT	[string]	The spatial extent of the output raster layer
Height in pixels	HEIGHT_IN_PIXE	[integer]	The number of rows in the output raster layer
NoData pixel count	NODATA_PIXEL_C	[număr]	The number of NoData pixels in the output raster layer
Statistics	OUTPUT_TABLE	[tabelă]	The output layer contains the following information **for each zone**: • Area: the area in square raster units in the zone; • Sum: the total sum of the pixel values in the zone; • Count: the number of pixels in the zone; • Min: the minimum pixel value in the zone; • Max: the maximum pixel value in the zone; • Mean: the mean of the pixel values in the zone;
Total pixel count	TOTAL_PIXEL_CO	[număr]	The count of pixels in the output raster layer
Width in pixels	WIDTH_IN_PIXEL	[număr]	The number of columns in the output raster layer

Cod Python

ID Algorithm: `nativ:rasterlayerzonalstats`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster surface volume

Calculates the volume under a raster surface relative to a given base level. This is mainly useful for Digital Elevation Models (DEM).

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de INTRARE	`INPUT`	[raster]	Input raster, representing a surface
Numărul benzii	`BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band that shall define the surface.
Base level	`LEVEL`	[număr] Implicit: 0.0	Define a base or reference value. This base is used in the volume calculation according to the `Method` parameter (see below).
Method	`METHOD`	[enumerare] Implicit: 0	Define the method for the volume calculation given by the difference between the raster pixel value and the `Base level`. Options: • 0 — Numără Doar Deasupra Nivelului de Bază: doar pixelii de deasupra nivelului de bază vor fi adăugați volumului. • 1 — Numără Doar Dedesubtul Nivelului de Bază: doar pixelii de sub nivelul de bază vor fi adăugați volumului. • 2 — Subtract Volumes Below Base level: pixels above the base level will add to the volume, pixels below the base level will subtract from the volume. • 3 — Add Volumes Below Base level: Add the volume regardless whether the pixel is above or below the base level. This is equivalent to sum the absolute values of the difference between the pixel value and the base level.

continues on next page

Tabelul 28.76 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Surface volume report	OUTPUT_HTML_FI	[html] Implicit: [Salvare într-un fișier temporar]	Specification of the output HTML report. One of: • Skip Output • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Surface volume table	OUTPUT_TABLE	[tabelă] Implicit: [Omitere rezultat]	Specification of the output table. One of: • Skip Output • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Volume	VOLUME	[număr]	The calculated volume
Area	AREA	[număr]	The area in square map units
Pixel_count	PIXEL_COUNT	[număr]	The total number of pixels that have been analyzed
Surface volume report	OUTPUT_HTML_FI	[html]	The output report (containing volume, area and pixel count) in HTML format
Surface volume table	OUTPUT_TABLE	[tabelă]	The output table (containing volume, area and pixel count)

Cod Python

ID Algorithm: `nativ:rastersurfacevolume`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Reclassify by layer

Reclassifies a raster band by assigning new class values based on the ranges specified in a vector table.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat raster	`INPUT_RASTER`	[raster]	Raster layer to reclassify
Numărul benzii	`RASTER_BAND`	[raster band] Implicit: Prima bandă a stratului raster	If the raster is multiband, choose the band you want to reclassify.
Layer containing class breaks	`INPUT_TABLE`	[vector: oricare]	Vector layer containing the values to use for classification.
Minimum class value field	`MIN_FIELD`	[tablefield: numeric]	Field with the minimum value of the range for the class. Use `-inf` to capture the lowest possible value.
Maximum class value field	`MAX_FIELD`	[tablefield: numeric]	Field with the maximum value of the range for the class. Use `inf` to capture the highest possible value.
Output value field	`VALUE_FIELD`	[tablefield: numeric]	Field with the value that will be assigned to the pixels that fall in the class (between the corresponding min and max values). Use `nan` to set the value of the range to NoData.
Reclassified raster	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`NO_DATA`	[număr] Implicit: -9999.0	Value to apply to NoData values.
Range boundaries	`RANGE_BOUNDARI`	[enumerare] Implicit: 0	Defines comparison rules for the classification. Options: • 0 — min < value <= max • 1 — min <= value < max • 2 — min <= value <= max • 3 — min < value < max
Use NoData when no range matches value	`NODATA_FOR_MIS`	[boolean] Implicit: False	Applies the NoData value to band values that do not fall in any class. If False, the original value is kept.

continues on next page

Tabelul 28.78 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the format of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Reclassified raster	`OUTPUT`	[raster]	Output raster layer with reclassified band values

Cod Python

ID Algorithm: `nativ:reclassifybylayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Reclassify by table

Reclassifies a raster band by assigning new class values based on the ranges specified in a fixed table.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat raster	`INPUT_RASTER`	[raster]	Raster layer to reclassify
Numărul benzii	`RASTER_BAND`	[raster band] Implicit: 1	Raster band for which you want to recalculate values.
Reclassification table	`TABLE`	[tabelă]	A 3-columns table to fill with the values to set the boundaries of each class (`Minimum` and `Maximum`) and the new `Value` to assign to the band values that fall in the class. The value `-inf` can be used as minimum, `inf` as maximum, and `nan` can be used to set the output value to NoData.
Reclassified raster	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output NoData value	`NO_DATA`	[număr] Implicit: -9999.0	Value to apply to NoData values.
Range boundaries	`RANGE_BOUNDARI`	[enumerare] Implicit: 0	Defines comparison rules for the classification. Options: • 0 — min < value <= max • 1 — min <= value < max • 2 — min <= value <= max • 3 — min < value < max
Use NoData when no range matches value	`NODATA_FOR_MIS`	[boolean] Implicit: False	Applies the NoData value to band values that do not fall in any class. If False, the original value is kept.

continues on next page

Tabelul 28.80 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the format of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Reclassified raster	`OUTPUT`	[raster]	Output raster layer with reclassified band values

Cod Python

ID Algorithm: `nativ:reclassifybytable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rescale raster

Rescales raster layer to a new value range, while preserving the shape (distribution) of the raster's histogram (pixel values). Input values are mapped using a linear interpolation from the source raster's minimum and maximum pixel values to the destination minimum and miximum pixel range.

By default the algorithm preserves the original NoData value, but there is an option to override it.

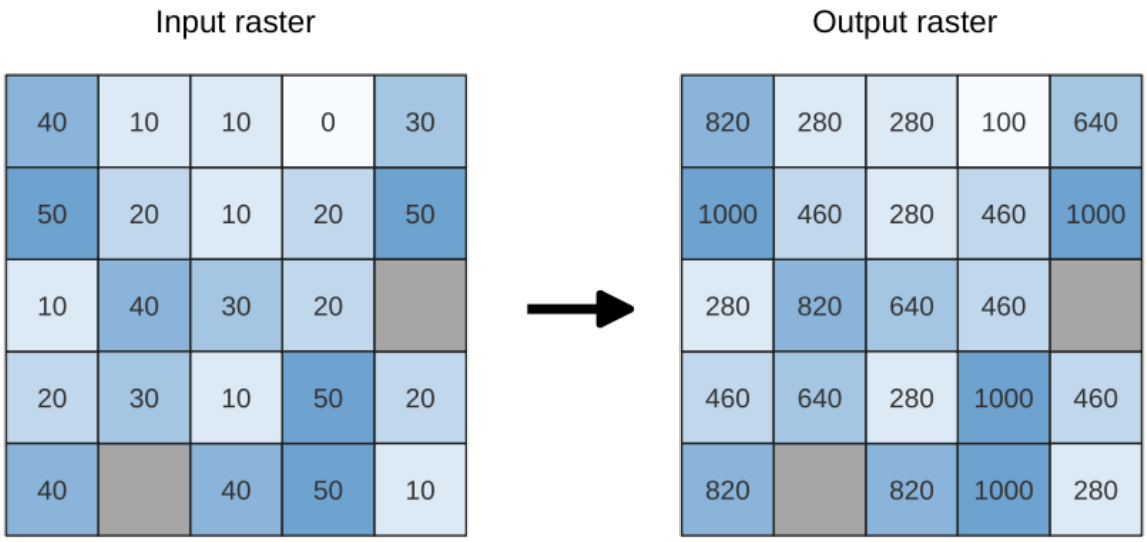


Fig. 28.26: Rescaling values of a raster layer from [0 - 50] to [100 - 1000]

Parametri

Etichetă	Nume	Tipul	Descriere
Input Raster	`INPUT`	[raster]	Raster layer to use for rescaling
Numărul benzii	`BAND`	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose a band.
New minimum value	`MINIMUM`	[număr] Default value: 0.0	Minimum pixel value to use in the rescaled layer
New maximum value	`MAXIMUM`	[număr] Default value: 255.0	Maximum pixel value to use in the rescaled layer
New NoData value Opţional	`NODATA`	[număr] Default value: Not set	Value to assign to the NoData pixels. If unset, original NoData values are preserved.
Rescaled	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Rescaled	`OUTPUT`	[raster]	Output raster layer with rescaled band values

Cod Python

ID Algorithm: `nativ:rescaleraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Round raster

Rounds the cell values of a raster dataset according to the specified number of decimals.

Alternatively, a negative number of decimal places may be used to round values to powers of a base n. For example, with a Base value n of 10 and Decimal places of -1, the algorithm rounds cell values to multiples of 10, -2 rounds to multiples of 100, and so on. Arbitrary base values may be chosen, the algorithm applies the same multiplicative principle. Rounding cell values to multiples of a base n may be used to generalize raster layers.

The algorithm preserves the data type of the input raster. Therefore byte/integer rasters can only be rounded to multiples of a base n, otherwise a warning is raised and the raster gets copied as byte/integer raster.

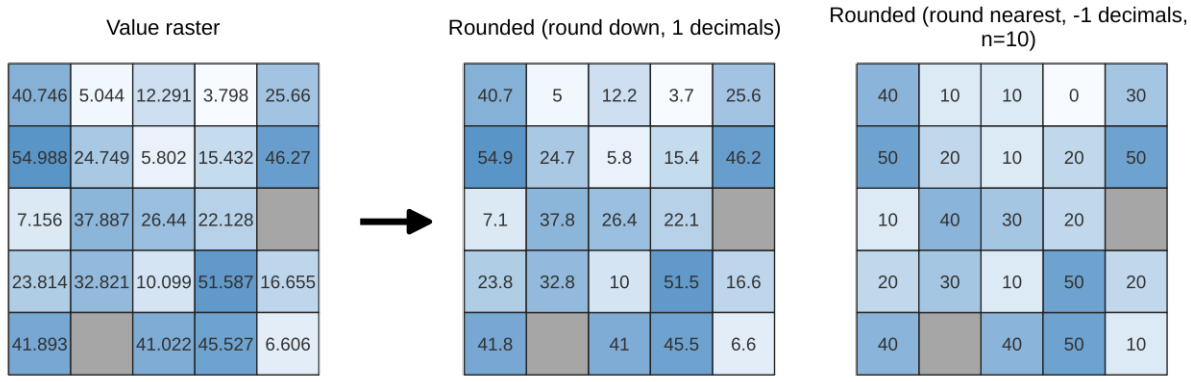


Fig. 28.27: Rounding values of a raster

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input raster	`INPUT`	[raster]	The raster to process.
Numărul benzii	`BAND`	[număr] Implicit: 1	The band of the raster
Rounding direction	`ROUNDING_DIREC`	[list] Implicit: 1	How to choose the target rounded value. Options are: • 0 — Rotunjire în sus • 1 — Rotunjire la valoarea cea mai apropiată • 2 — Rotunjire în jos
Number of decimals places	`DECIMAL_PLACES`	[număr] Implicit: 2	Number of decimals places to round to. Use negative values to round cell values to a multiple of a base n
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Base n for rounding to multiples of n	`BASE_N`	[număr] Implicit: 10	When the `DECIMAL_PLACES` parameter is negative, raster values are rounded to multiples of the base n value

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	The output raster layer with values rounded for the selected band.

Cod Python

ID Algorithm: `nativ:roundrastervalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Sample raster values

Extracts raster values at the point locations. If the raster layer is multiband, each band is sampled.

The attribute table of the resulting layer will have as many new columns as the raster layer band count.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: punct]	Point vector layer to use for sampling
Raster Layer	`RASTERCOPY`	[raster]	Raster layer to sample at the given point locations.
Output column prefix	`COLUMN_PREFIX`	[string] Implicit: «SAMPLE_»	Prefix for the names of the added columns.
Sampled Opţional	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output layer containing the sampled values. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Sampled	`OUTPUT`	[vector: punct]	The output layer containing the sampled values.

Cod Python

ID Algorithm: `nativ:rastersampling`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Zonal histogram

Appends fields representing counts of each unique value from a raster layer contained within polygon features.

The output layer attribute table will have as many fields as the unique values of the raster layer that intersects the polygon(s).

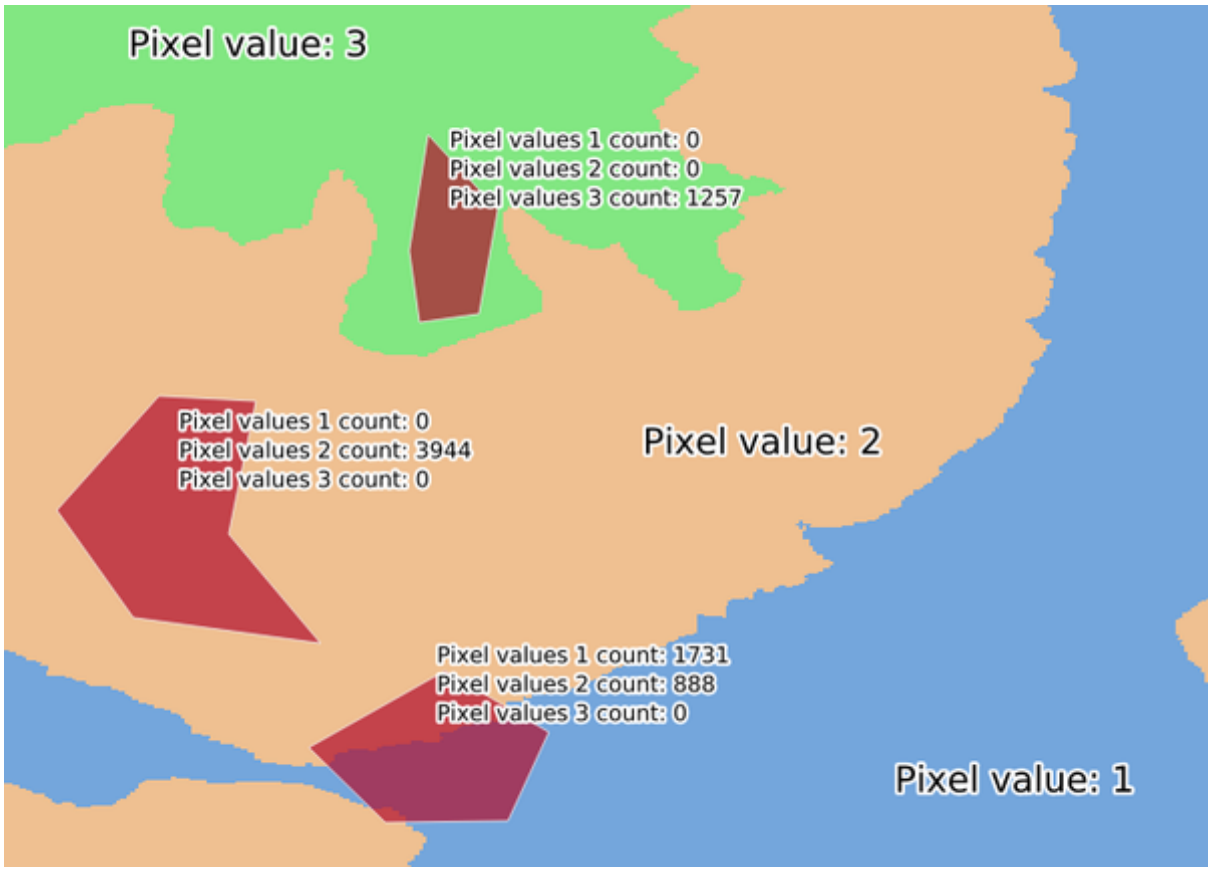


Fig. 28.28: Raster layer histogram example

Parametri

Etichetă	Nume	Tipul	Descriere
Strat raster	`INPUT_RASTER`	[raster]	Input raster layer.
Numărul benzii	`RASTER_BAND`	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose a band.
Vector layer containing zones	`INPUT_VECTOR`	[vector: poligon]	Vector polygon layer that defines the zones.
Output column prefix	`COLUMN_PREFIX` Opţional	[string] Implicit: «HISTO_»	Prefix for the output columns names.
Output zones	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector polygon layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Output zones	`OUTPUT`	[vector: poligon]	The output vector polygon layer.

Cod Python

ID Algorithm: `nativ:zonalhistogram`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Zonal statistics

Calculates statistics of a raster layer for each feature of an overlapping polygon vector layer.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Vector polygon layer that contains the zones.
Strat raster	`INPUT_RASTER`	[raster]	Input raster layer.
Raster band	`RASTER_BAND`	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose a band for the statistics.
Output column prefix	`COLUMN_PREFIX`	[string] Implicit: «_»	Prefix for the output columns names.
Statistics to calculate	`STATISTICS`	[enumeration] [list] Implicit: [0,1,2]	List of statistical operator for the output. Options: • 0 — Total • 1 — Sumă • 2 — Medie • 3 — Median • 4 — Dev. st. • 5 — Minimum • 6 — Maximum • 7 — Interval • 8 — Minoritate • 9 — Majoritate • 10 — Varietate • 11 — Variantă
Zonal Statistics	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector polygon layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Zonal Statistics	`OUTPUT`	[vector: poligon]	The zone vector layer with added statistics.

Cod Python

ID Algorithm: `nativ:zonalstatisticsfb`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.16 Crearea Rasterelor

Create constant raster layer

Generates raster layer for given extent and cell size filled with the specified value.

Additionally an output data type can be specified. The algorithm will abort if a value has been entered that cannot be represented by the selected output raster data type.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Constant value	`NUMBER`	[număr] Implicit: 1	Constant pixel value for the output raster layer.
Constant	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 5	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Byte • 1 — Integer16 • 2 — Unsigned Integer16 • 3 — Integer32 • 4 — Unsigned Integer32 • 5 — Float32 • 6 — Float64

Rezultat

Etichetă	Nume	Tipul	Descriere
Constant	`OUTPUT`	[raster]	Raster covering the desired extent with the specified pixel size and value.

Cod Python

ID Algorithm: `nativ:createconstantrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (binomial distribution)

Generates a raster layer for given extent and cell size filled with binomially distributed random values.

By default, the values will be chosen given an N of 10 and a probability of 0.5. This can be overridden by using the advanced parameter for N and probability. The raster data type is set to Integer types (Integer16 by default). The binomial distribution random values are defined as positive integer numbers. A floating point raster will represent a cast of integer values to floating point.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 0	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Integer16 • 1 — Unsigned Integer16 • 2 — Integer32 • 3 — Unsigned Integer32 • 4 — Float32 • 5 — Float64
N	`N`	[număr] Implicit: 10	
Probabilitate	`PROBABILITY`	[număr] Implicit: 0.5	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with random values

Cod Python

ID Algorithm: `nativ:createrandombinomialrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (exponential distribution)

Generates a raster layer for given extent and cell size filled with exponentially distributed random values.

By default, the values will be chosen given a lambda of 1.0. This can be overridden by using the advanced parameter for lambda. The raster data type is set to Float32 by default as the exponential distribution random values are floating point numbers.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 0	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Float32 • 1 — Float64
Lambda	`LAMBDA`	[număr] Implicit: 1.0	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with random values

Cod Python

ID Algorithm: `nativ:createrandomexponentialrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (gamma distribution)

Generates a raster layer for given extent and cell size filled with gamma distributed random values.

By default, the values will be chosen given an alpha and beta value of 1.0. This can be overridden by using the advanced parameter for alpha and beta. The raster data type is set to Float32 by default as the gamma distribution random values are floating point numbers.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 0	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Float32 • 1 — Float64
Alpha	`ALPHA`	[număr] Implicit: 1.0	
Beta	`BETA`	[număr] Implicit: 1.0	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with randomly distributed values

Cod Python

ID Algorithm: `nativ:createrandomgammarasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (geometric distribution)

Generates a raster layer for given extent and cell size filled with geometrically distributed random values.

By default, the values will be chosen given a probability of 0.5. This can be overridden by using the advanced parameter for mean value. The raster data type is set to Integer types (Integer16 by default). The geometric distribution random values are defined as positive integer numbers. A floating point raster will represent a cast of integer values to floating point.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 0	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Integer16 • 1 — Unsigned Integer16 • 2 — Integer32 • 3 — Unsigned Integer32 • 4 — Float32 • 5 — Float64
Probabilitate	`PROBABILITY`	[număr] Implicit: 0.5	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with randomly distributed values

Cod Python

ID Algorithm: `nativ:createrandomgeometricrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (negative binomial distribution)

Generates a raster layer for given extent and cell size filled with negative binomially distributed random values.

By default, the values will be chosen given a distribution parameter k of 10.0 and a probability of 0.5. This can be overridden by using the advanced parameters for k and probability. The raster data type is set to Integer types (Integer16 by default). The negative binomial distribution random values are defined as positive integer numbers. A floating point raster will represent a cast of integer values to floating point.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 0	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Integer16 • 1 — Unsigned Integer16 • 2 — Integer32 • 3 — Unsigned Integer32 • 4 — Float32 • 5 — Float64
Distribution parameter k	`K_PARAMETER`	[număr] Implicit: 10	
Probabilitate	`PROBABILITY`	[număr] Implicit: 0.5	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with randomly distributed values

Cod Python

ID Algorithm: `nativ:createrandomnegativebinomialrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (normal distribution)

Generates a raster layer for given extent and cell size filled with normally distributed random values.

By default, the values will be chosen given a mean of 0.0 and a standard deviation of 1.0. This can be overridden by using the advanced parameters for mean and standard deviation value. The raster data type is set to Float32 by default as the normal distribution random values are floating point numbers.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 0	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Float32 • 1 — Float64
Media distribuției normale	`MEAN`	[număr] Implicit: 0.0	
Abaterea standard a distribuției normale	`STDDEV`	[număr] Implicit: 1.0	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with randomly distributed values

Cod Python

ID Algorithm: `nativ:createrandomnormalrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (poisson distribution)

Generates a raster layer for given extent and cell size filled with poisson distributed random values.

By default, the values will be chosen given a mean of 1.0. This can be overridden by using the advanced parameter for mean value. The raster data type is set to Integer types (Integer16 by default). The poisson distribution random values are positive integer numbers. A floating point raster will represent a cast of integer values to floating point.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 0	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Integer16 • 1 — Unsigned Integer16 • 2 — Integer32 • 3 — Unsigned Integer32 • 4 — Float32 • 5 — Float64
Mean	`MEAN`	[număr] Implicit: 1.0	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with randomly distributed values

Cod Python

ID Algorithm: `nativ:createrandompoissonrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create random raster layer (uniform distribution)

Generates a raster layer for given extent and cell size filled with random values.

By default, the values will range between the minimum and maximum value of the specified output raster type. This can be overridden by using the advanced parameters for lower and upper bound value. If the bounds have the same value or both are zero (default) the algorithm will create random values in the full value range of the chosen raster data type. Choosing bounds outside the acceptable range of the output raster type will abort the algorithm.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Desired extent	`EXTENT`	[extent]	Specify the spatial extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: CRS-ul proiectului	CRS for the output raster layer
Dimensiune pixel	`PIXEL_SIZE`	[număr] Implicit: 1.0	Pixel size (X=Y) in map units.
Raster de ieșire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Output raster data type	`OUTPUT_TYPE` Implicit: 5	[enumerare]	Defines the data type of the output raster file. Options: • 0 — Byte • 1 — Integer16 • 2 — Unsigned Integer16 • 3 — Integer32 • 4 — Unsigned Integer32 • 5 — Float32 • 6 — Float64
Lower bound for random number range	`LOWER_BOUND`	[număr] Implicit: 0.0	
Upper bound for random number range	`UPPER_BOUND`	[număr] Implicit: 0.0	

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	Raster covering the desired extent with the cell size filled with randomly distributed values

Cod Python

ID Algorithm: `nativ:createrandomuniformrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.17 Raster terrain analysis

Aspectul

Calculates the aspect of the Digital Terrain Model in input. The final aspect raster layer contains values from 0 to 360 that express the slope direction, starting from north (0°) and continuing clockwise.

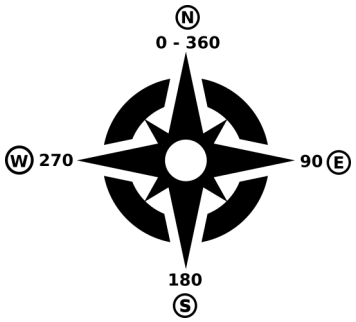


Fig. 28.29: Aspect values

The following picture shows the aspect layer reclassified with a color ramp:

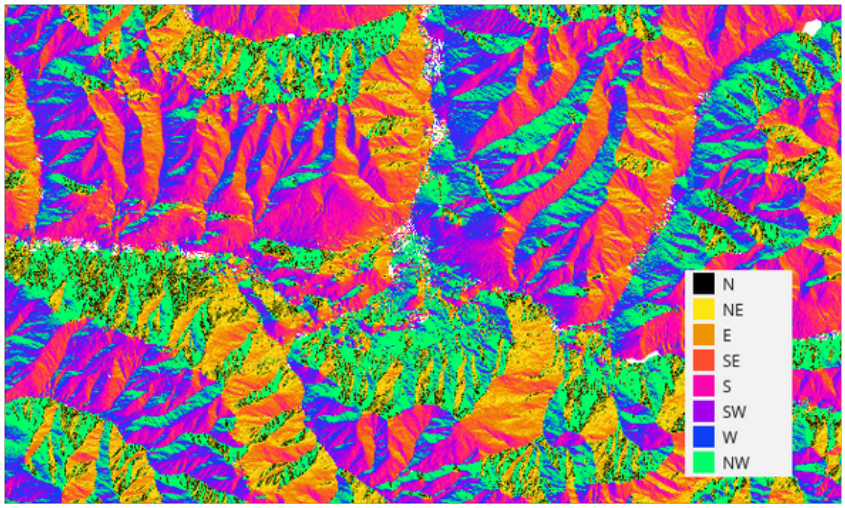


Fig. 28.30: Aspect layer reclassified

Parametri

Etichetă	Nume	Tipul	Descriere
Elevation layer	`INPUT`	[raster]	Digital Terrain Model raster layer
Z factor	`Z_FACTOR`	[număr] Implicit: 1.0	Vertical exaggeration. This parameter is useful when the Z units differ from the X and Y units, for example feet and meters. You can use this parameter to adjust for this. The default is 1 (no exaggeration).
Aspect	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output aspect raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Aspect	`OUTPUT`	[raster]	The output aspect raster layer

Cod Python

Algorithm ID: `qgis:aspect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

DTM filter (slope-based)

`NEW in 3.34`

Can be used to filter a digital elevation model in order to classify its cells into ground and object (non-ground) cells.

The tool uses concepts as described by Vosselman (2000) and is based on the assumption that a large height difference between two nearby cells is unlikely to be caused by a steep slope in the terrain. The probability that the higher cell might be non-ground increases when the distance between the two cells decreases. Therefore the filter defines a maximum height difference (`dz_max`) between two cells as a function of the distance (`d`) between the cells (`dz_max( d ) = d`). A cell is classified as terrain if there is no cell within the kernel radius to which the height difference is larger than the allowed maximum height difference at the distance between these two cells.

The approximate terrain slope (`s`) parameter is used to modify the filter function to match the overall slope in the study area (`dz_max( d ) = d * s`). A 5 % confidence interval (`ci = 1.65 * sqrt( 2 * stddev )`) may be used to modify the filter function even further by either relaxing (`dz_max( d ) = d * s + ci`) or amplifying (`dz_max( d ) = d * s - ci`) the filter criterium.

References: Vosselman, G. (2000): Slope based filtering of laser altimetry data. IAPRS, Vol. XXXIII, Part B3, Amsterdam, The Netherlands, 935-942

Vezi şi:

This tool is a port of the SAGA DTM Filter (slope-based)

Parametri

Etichetă	Nume	Tipul	Descriere
Input layer	`INPUT`	[raster]	Digital Terrain Model raster layer
Band number	`BAND`	[number] [list]	The band of the DEM to consider
Kernel radius (pixels)	`RADIUS`	[număr] Default: 5	The radius of the filter kernel (in pixels). Must be large enough to reach ground cells next to non-ground objects.
Terrain slope (%, pixel size/vertical units)	`TERRAIN_SLOPE`	[număr] Default: 30	The approximate terrain slope in `%`. The terrain slope must be adjusted to account for the ratio of height units vs raster pixel dimensions. Used to relax the filter criterium in steeper terrain.
Filter modification	`FILTER_MODIFIC`	[list] Default: 0	Choose whether to apply the filter kernel without modification or to use a confidence interval to relax or amplify the height criterium. • 0 - None • 1 - Relax filter • 2 - Amplify
Standard deviation	`STANDARD_DEVIA`	[număr] Default: 0.1	The standard deviation used to calculate a 5% confidence interval applied to the height threshold.
Output layer (ground) Opţional	`OUTPUT_GROUND`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the filtered DEM containing only cells classified as ground. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

continues on next page

Tabelul 28.84 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output layer (non-ground objects) Opţional	`OUTPUT_NONGROU`	[raster] Implicit: `[Omitere rezultat]`	Specify the non-ground objects removed by the filter. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer (ground)	`OUTPUT_GROUND`	[raster]	The filtered DEM containing only cells classified as ground.
Output layer (non-ground objects)	`OUTPUT_NONGROU`	[raster]	The non-ground objects removed by the filter.

Cod Python

Algorithm ID: `native:dtmslopebasedfilter`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Hillshade

Calculates the hillshade raster layer given an input Digital Terrain Model.

The shading of the layer is calculated according to the sun position: you have the options to change both the horizontal angle (azimuth) and the vertical angle (sun elevation) of the sun.

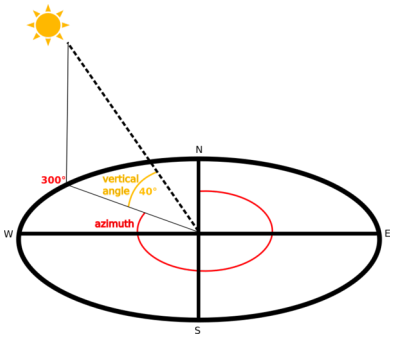


Fig. 28.31: Azimuth and vertical angle

The hillshade layer contains values from 0 (complete shadow) to 255 (complete sun). Hillshade is used usually to better understand the relief of the area.

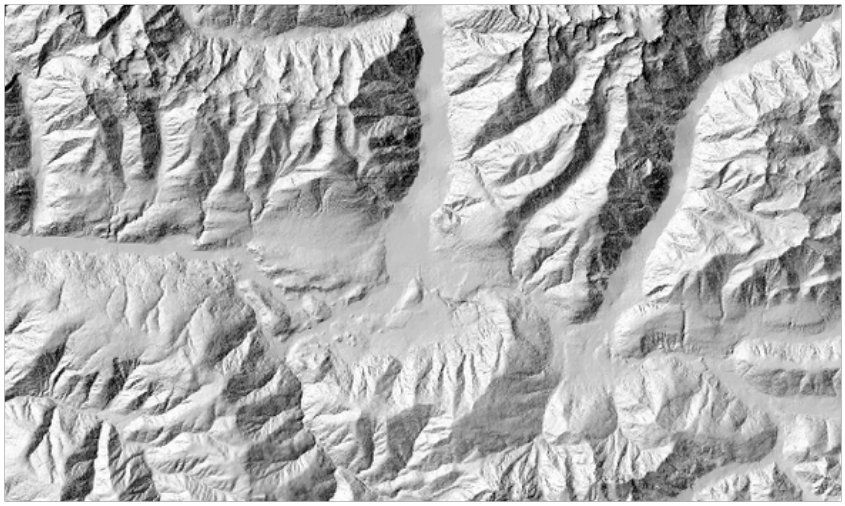


Fig. 28.32: Hillshade layer with azimuth 300 and vertical angle 45

Particularly interesting is to give the hillshade layer a transparency value and overlap it with the elevation raster:

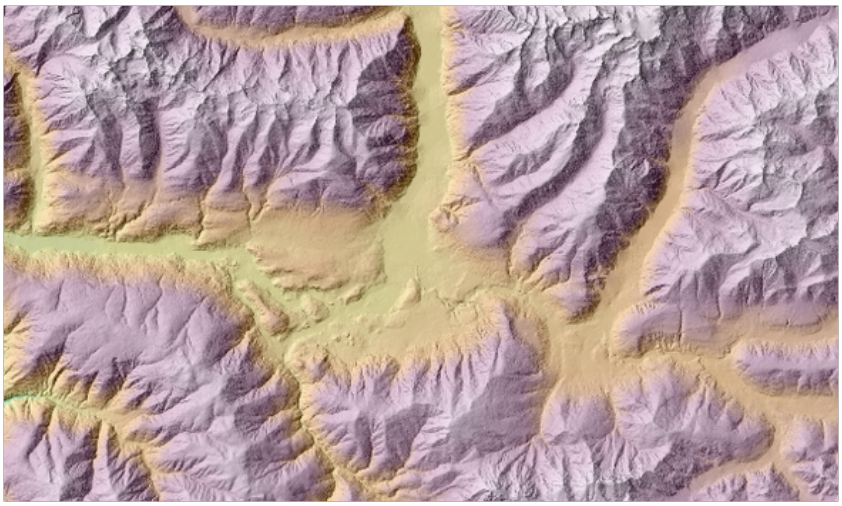


Fig. 28.33: Overlapping the hillshade with the elevation layer

Parametri

Etichetă	Nume	Tipul	Descriere
Elevation layer	`INPUT`	[raster]	Digital Terrain Model raster layer
Z factor	`Z_FACTOR`	[număr] Implicit: 1.0	Vertical exaggeration. This parameter is useful when the Z units differ from the X and Y units, for example feet and meters. You can use this parameter to adjust for this. Increasing the value of this parameter will exaggerate the final result (making it look more „hilly”). The default is 1 (no exaggeration).

continues on next page

Tabelul 28.85 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Azimuth (horizontal angle)	`AZIMUTH`	[număr] Implicit: 300.0	Set the horizontal angle (in degrees) of the sun (clockwise direction). Range: 0 to 360. 0 is north.
Vertical angle	`V_ANGLE`	[număr] Implicit: 40.0	Set the vertical angle (in degrees) of the sun, that is the height of the sun. Values can go from 0 (minimum elevation) to 90 (maximum elevation).
Hillshade	`OUTPUT`	[raster] Default: `Save to temporary file`	Specify the output hillshade raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Hillshade	`OUTPUT`	[raster]	The output hillshade raster layer

Cod Python

Algorithm ID: `qgis:hillshade`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Hypsometric curves

Calculates hypsometric curves for an input Digital Elevation Model. Curves are produced as CSV files in an output folder specified by the user.

A hypsometric curve is a cumulative histogram of elevation values in a geographical area.

You can use hypsometric curves to detect differences in the landscape due to the geomorphology of the territory.

Parametri

Etichetă	Nume	Tipul	Descriere
DEM to analyze	`INPUT_DEM`	[raster]	Digital Terrain Model raster layer to use for calculating altitudes
Boundary layer	`BOUNDARY_LAYER`	[vector: poligon]	Polygon vector layer with boundaries of areas used to calculate hypsometric curves
Step	`STEP`	[număr] Implicit: 100.0	Vertical distance between curves

continues on next page

Tabelul 28.86 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Use % of area instead of absolute value	`USE_PERCENTAGE`	[boolean] Implicit: *False*	Write area percentage to "Area" field of the CSV file instead of the absolute area
Hypsometric curves	`OUTPUT_DIRECTO`	[folder]	Specify the output folder for the hypsometric curves. One of: • Salvare Intr-un Director Temporar • Salvare în Director

Rezultat

Etichetă	Nume	Tipul	Descriere
Hypsometric curves	`OUTPUT_DIRECTO`	[folder]	Directory containing the files with the hypsometric curves. For each feature from the input vector layer, a CSV file with area and altitude values will be created. The file names start with `histogram_`, followed by layer name and feature ID.

	A	B
1	Area	Elevation
2	177475194.383	307
3	233206029.24	407
4	295553735.793	507
5	394718815.615	607
6	501801102.615	707
7	624399019.792	807
8	828877274.39	907
9	1042693465.68	1007
10	1277373021.81	1107
11	1556443975.41	1207
12	1888617494.27	1307
13	2248520437.31	1407
14	2627916813.17	1507
15	3010880212.04	1607
16	3411087555.34	1707

Cod Python

Algorithm ID: `qgis:hypsometriccurves`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Relief

Creates a shaded relief layer from digital elevation data. You can specify the relief color manually, or you can let the algorithm choose automatically all the relief classes.

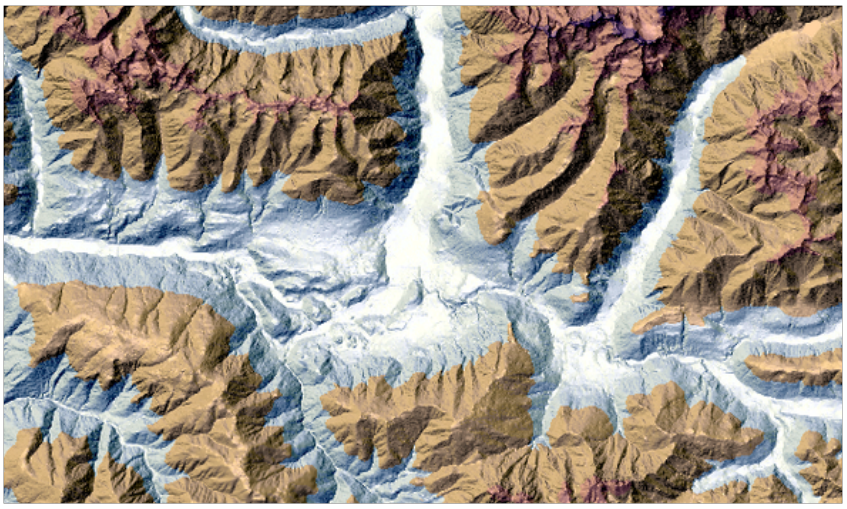


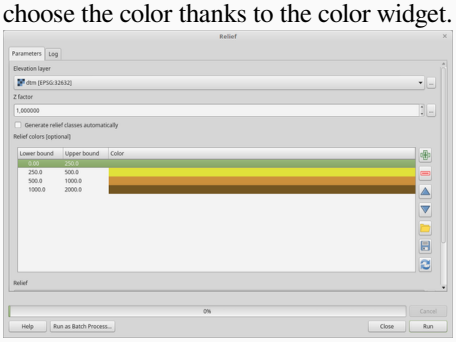
Fig. 28.34: Relief layer

Parametri

Etichetă	Nume	Tipul	Descriere
Elevation layer	`INPUT`	[raster]	Digital Terrain Model raster layer
Z factor	`Z_FACTOR`	[număr] Implicit: 1.0	Vertical exaggeration. This parameter is useful when the Z units differ from the X and Y units, for example feet and meters. You can use this parameter to adjust for this. Increasing the value of this parameter will exaggerate the final result (making it look more „hilly”). The default is 1 (no exaggeration).
Generate relief classes automatically	`AUTO_COLORS`	[boolean] Implicit: *False*	If you check this option the algorithm will create all the relief color classes automatically

continues on next page

Tabelul 28.87 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Relief colors Opţional	`COLORS`	[table widget]	Use the table widget if you want to choose the relief colors manually. You can add as many color classes as you want: for each class you can choose the lower and upper bound and finally by clicking on the color row you can choose the color thanks to the color widget.  Fig. 28.35: Manually setting of relief color classes The buttons in the right side panel give you the chance to: add or remove color classes, change the order of the color classes already defined, open an existing file with color classes and save the current classes as file.
Relief	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output relief raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…
Frequency distribution Opţional	`FREQUENCY_DIST`	[tabelă] Implicit: `[Omitere rezultat]`	Specify the CSV table for the output frequency distribution. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Relief	`OUTPUT`	[raster]	The output relief raster layer
Frequency distribution	`OUTPUT`	[tabelă]	The output frequency distribution

Cod Python

Algorithm ID: `qgis:relief`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Ruggedness index

Calculates the quantitative measurement of terrain heterogeneity described by Riley et al. (1999). It is calculated for every location, by summarizing the change in elevation within the 3x3 pixel grid.

Each pixel contains the difference in elevation from a center cell and the 8 cells surrounding it.

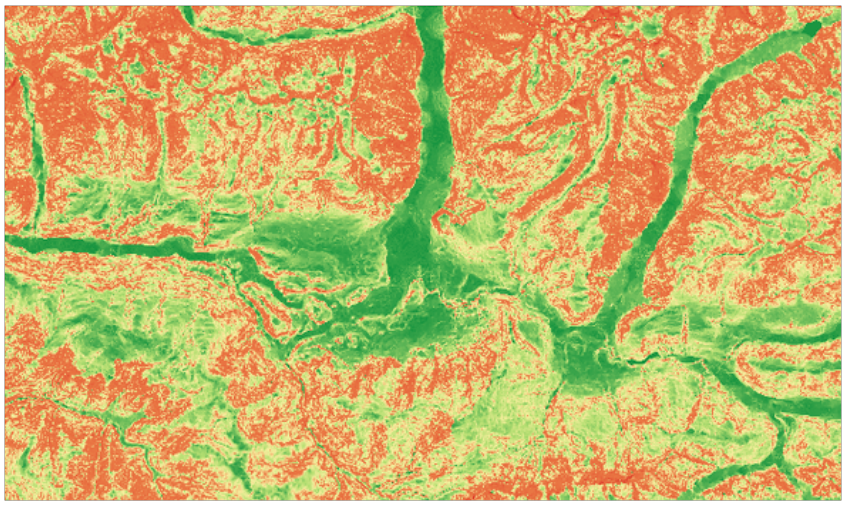


Fig. 28.36: Ruggedness layer from low (red) to high values (green)

Parametri

Etichetă	Nume	Tipul	Descriere
Elevation layer	`INPUT`	[raster]	Digital Terrain Model raster layer
Z factor	`Z_FACTOR`	[număr] Implicit: 1.0	Vertical exaggeration. This parameter is useful when the Z units differ from the X and Y units, for example feet and meters. You can use this parameter to adjust for this. Increasing the value of this parameter will exaggerate the final result (making it look more rugged). The default is 1 (no exaggeration).
Ruggedness	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output ruggedness raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Ruggedness	`OUTPUT`	[raster]	The output ruggedness raster layer

Cod Python

Algorithm ID: `qgis:ruggednessindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Panta

Calculates the slope from an input raster layer. The slope is the angle of inclination of the terrain and is expressed in **degrees**.

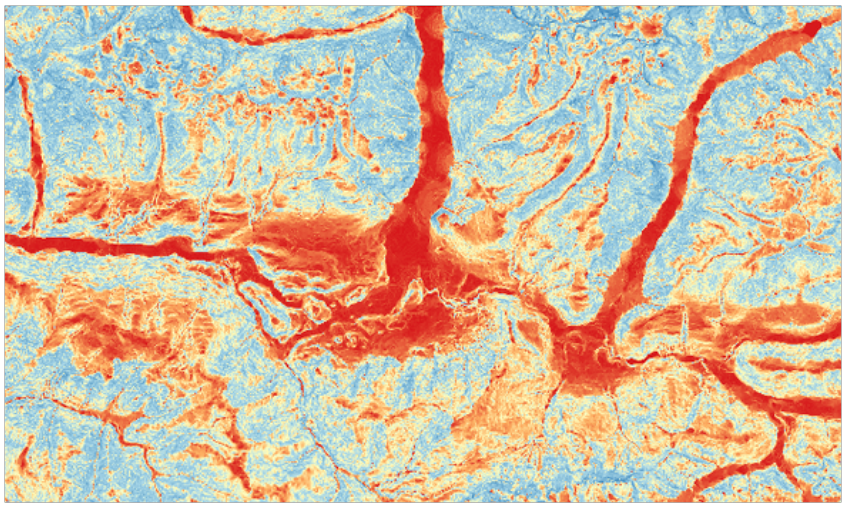


Fig. 28.37: Flat areas in red, steep areas in blue

Parametri

Etichetă	Nume	Tipul	Descriere
Elevation layer	`INPUT`	[raster]	Digital Terrain Model raster layer
Z factor	`Z_FACTOR`	[număr] Implicit: 1.0	Vertical exaggeration. This parameter is useful when the Z units differ from the X and Y units, for example feet and meters. You can use this parameter to adjust for this. Increasing the value of this parameter will exaggerate the final result (making it steeper). The default is 1 (no exaggeration).
Slope	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output slope raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Slope	`OUTPUT`	[raster]	The output slope raster layer

Cod Python

Algorithm ID: `qgis:slope`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.18 Raster tools

Align raster

`NEW in 3.34`

Aligns raster by resampling it to the same cell size and reprojecting to the same CRS as a reference raster.

Atenționare: This algorithm is ONLY available in the *Model Designer* context.

Parametri

Etichetă	Nume	Tipul	Descriere
Input layer	`INPUT`	[raster]	Input raster layer to align
Resampling method	`RESAMPLING_MET`	[enumerare] Implicit: 0	Method to use for input layer resampling. `Available options` are: • 0 — Nearest Neighbour • 1 — Bilinear (2x2 kernel) • 2 — Cubic (4x4 kernel) • 3 — Cubic B-Spline (4x4 kernel) • 4 — Lanczos (6x6 kernel) • 5 — Average • 6 — Mode • 7 — Maximum • 8 — Minimum • 9 — Median • 10 — First quartile (Q1) • 11 — Third quartile (Q3)
Rescale values according to the cell size	`RESCALE`	[boolean] Default: No	
Reference layer	`REFERENCE_LAYE`	[raster]	A raster layer that will be used to fetch extent, cell size and CRS that will be applied to input layers.
Override reference CRS Opţional	`CRS`	[crs]	CRS to be used instead of the reference layer's
Override reference cell size X Opţional	`CELL_SIZE_X`	[număr]	Cell size in X direction to be used instead of the reference layer's
Override reference cell size Y Opţional	`CELL_SIZE_Y`	[număr]	Cell size in Y direction to be used instead of the reference layer's
Override reference grid offset X Opţional	`GRID_OFFSET_X`	[număr]	Offset in X direction to apply to cells grid
Override reference grid offset Y Opţional	`GRID_OFFSET_Y`	[număr]	Offset in Y direction to apply to cells grid

continues on next page

Tabelul 28.88 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Clip to extent Opţional	`EXTENT`	[extent]	Specify the extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Aligned raster	`OUTPUT`	[raster]	output raster with pixels resampled

Rezultat

Etichetă	Nume	Tipul	Descriere
Aligned raster	`OUTPUT`	[raster]	output raster with pixels resampled

Cod Python

Algorithm ID: `native:alignraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Align rasters

`NEW in 3.34`

Aligns rasters by resampling them to the same cell size and reprojecting to the same CRS as a reference raster.

Atenționare: This algorithm is NOT available in the *Model Designer* context.

Parametri

Etichetă	Nume	Tipul	Descriere
Input layers	`INPUT`	[raster] [list]	List of input raster layers with resampling options associated (filled as a `QgsProcessingParameterAlignRasterLayers` item): **Input layer [string] (`inputFile`)** Full path of the input layer to align **Output file [string] (`outputFile`)** Full path of the corresponding aligned output layer **Resampling method [enumeration] (`resampleMethod`)** Method to use for input layer resampling. `Available options` are: • 0 — Nearest Neighbour • 1 — Bilinear (2x2 kernel) • 2 — Cubic (4x4 kernel) • 3 — Cubic B-Spline (4x4 kernel) • 4 — Lanczos (6x6 kernel) • 5 — Average • 6 — Mode • 7 — Maximum • 8 — Minimum • 9 — Median • 10 — First quartile (Q1) • 11 — Third quartile (Q3) **Rescale values according to the cell size** [boolean] (`rescale`)
Reference layer	`REFERENCE_LAYE`	[raster]	A raster layer that will be used to fetch extent, cell size and CRS that will be applied to input layers.
Override reference CRS Opţional	`CRS`	[crs]	CRS to be used instead of the reference layer's
Override reference cell size X Opţional	`CELL_SIZE_X`	[număr]	Cell size in X direction to be used instead of the reference layer's
Override reference cell size Y Opţional	`CELL_SIZE_Y`	[număr]	Cell size in Y direction to be used instead of the reference layer's
Override reference grid offset X Opţional	`GRID_OFFSET_X`	[număr]	Offset in X direction to apply to cells grid
Override reference grid offset Y Opţional	`GRID_OFFSET_Y`	[număr]	Offset in Y direction to apply to cells grid

continues on next page

Tabelul 28.89 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Clip to extent Opţional	`EXTENT`	[extent]	Specify the extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

Rezultat

Etichetă	Nume	Tipul	Descriere
Aligned rasters	`OUTPUT_LAYERS`	[raster] [list]	output rasters with pixels resampled

Cod Python

Algorithm ID: `native:alignrasters`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Convert map to raster

Creates a raster image of map canvas content.

A *map theme* can be selected to render a predetermined set of layers with a defined style for each layer.

Alternatively, a single layer can be selected if no map theme is set.

If neither map theme nor layer is set, the current map content will be rendered. The minimum extent entered will internally be extended to be a multiple of the tile size.

Parametri

Etichetă	Nume	Tipul	Descriere
Minimum extent to render (xmin, xmax, ymin, ymax)	`EXTENT`	[extent]	Specify the extent of the output raster layer. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Tile size	`TILE_SIZE`	[număr] Implicit: 1024	Size of the tile of the output raster layer. Minimum value: 64.
Map units per pixel	`MAP_UNITS_PER_`	[număr] Implicit: 100.0	Pixel size (in map units). Minimum value: 0.0
Make background transparent	`MAKE_BACKGROUN`	[boolean] Implicit: False	Allows exporting the map with a transparent background. Outputs an RGBA (instead of RGB) image if set to `True`.
Map theme to render Opţional	`MAP_THEME`	[enumerare]	Use an existing *map theme* for the rendering.
Single layer to render Opţional	`LAYER`	[enumerare]	Choose a single layer for the rendering
Output layer	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[raster]	Stratul raster de ieșire

Cod Python

Algorithm ID: `native:rasterize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Umplere celule NoData

Resets the NoData values in the input raster to a chosen value, resulting in raster dataset with no NoData pixels.

The algorithm respects the input raster data type, e.g. a floating point fill value will be truncated when applied to an integer raster.

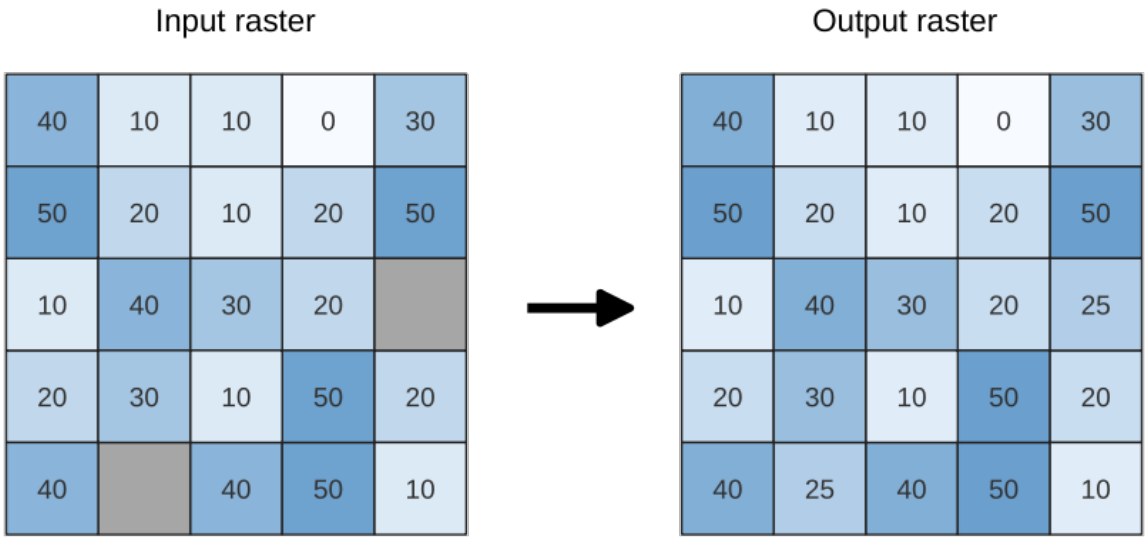


Fig. 28.38: Filling NoData values (in grey) of a raster

Parametri

Etichetă	Nume	Tipul	Descriere
Input raster	`INPUT`	[raster]	The raster to process.
Numărul benzii	`BAND`	[număr] Implicit: 1	The band of the raster
Fill value	`FILL_VALUE`	[număr] Implicit: 1.0	Set the value to use for the NoData pixels
Raster de ieşire	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Raster de ieșire	`OUTPUT`	[raster]	The output raster layer with filled data cells.

Cod Python

ID Algorithm: `nativ:fillnodata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Generate XYZ tiles (Directory)

Generates raster "XYZ" tiles using the current QGIS project as individual images to a directory structure.

Optionally, a Leaflet HTML output file using the generated tiles as a map layer could be created.

Parametri

Basic parameters

Etichetă	Nume	Tipul	Descriere
Extent (xmin, xmax, ymin, ymax)	`EXTENT`	[extent]	Specify the extent of the tiles. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Minimum zoom	`ZOOM_MIN`	[număr] Implicit: 12	Minimum 0, maximum 25.
Maximum zoom	`ZOOM_MAX`	[număr] Implicit: 12	Minimum 0, maximum 25.

continues on next page

Tabelul 28.91 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
DPI	`DPI`	[număr] Implicit: 96	Minimum 48, maximum 600.
Background color Opţional	`BACKGROUND_COL`	[color] Implicit: QColor(0, 0, 0, 0)	Choose the background color for the tiles
Enable antialiasing	`ANTIALIAS`	[boolean] Default: True	Determines if antialiasing should be enabled
Tile format	`TILE_FORMAT`	[enumerare] Implicit: 0	One of: • 0 — PNG • 1 — JPG
Quality (JPG only) Opţional	`QUALITY`	[număr] Implicit: 75	Minimum 1, maximum 100.
Metatile size Opţional	`METATILESIZE`	[număr] Implicit: 4	Specify a custom metatile size when generating XYZ tiles. Larger values may speed up the rendering of tiles and provide better labelling (fewer gaps without labels) at the expense of using more memory. Minimum 1, maximum 20.
Tile width Opţional	`TILE_WIDTH`	[număr] Implicit: 256	Minimum 1, maximum 4096.
Tile height Opţional	`TILE_HEIGHT`	[număr] Implicit: 256	Minimum 1, maximum 4096.
Use inverted tile Y axis (TMS conventions) Opţional	`TMS_CONVENTION`	[boolean] Implicit: False	
Director de ieşire Opţional	`OUTPUT_DIRECTO`	[folder] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output directory (for the tiles). One of: • Skip Output • Salvare Intr-un Director Temporar • Salvare în Director
Output html (Leaflet) Opţional	`OUTPUT_HTML`	[html] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output HTML file. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Advanced parameters

`NEW in 3.30`

Etichetă	Nume	Tipul	Descriere
Leaflet HTML output title Opţional	`HTML_TITLE`	[string] Default: Not set	HTML <title>-tag used for the Leaflet HTML output file.
Leaflet HTML output attribution Opţional	`HTML_ATTRIBUTI`	[string] Default: Not set	Custom map attribution used for the Leaflet HTML output file. HTML links are possible.

continues on next page

Tabelul 28.92 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Include OpenStreetMap basemap in Leaflet HTML output Opţional	`HTML_OSM`	[boolean] Implicit: False	An OpenStreetMap basemap layer (source: https://tile.openstreetmap.org) is included in the Leaflet HTML output file. Proper map attribution is added automatically.

Rezultat

Etichetă	Nume	Tipul	Descriere
Director de ieşire	`OUTPUT_DIRECTO`	[folder]	Output directory (for the tiles)
Output html (Leaflet)	`OUTPUT_HTML`	[html]	The output HTML (Leaflet) file

Cod Python

Algorithm ID: `native:tilesxyzdirectory`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Generate XYZ tiles (MBTiles)

Generates raster “XYZ” tiles using the current QGIS project as a single file in the “MBTiles” format.

Parametri

Etichetă	Nume	Tipul	Descriere
Extent (xmin, xmax, ymin, ymax)	`EXTENT`	[extent]	Specify the extent of the tiles. It will internally be extended to a multiple of the tile size. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Minimum zoom	`ZOOM_MIN`	[număr] Implicit: 12	Minimum 0, maximum 25.
Maximum zoom	`ZOOM_MAX`	[număr] Implicit: 12	Minimum 0, maximum 25.
DPI	`DPI`	[număr] Implicit: 96	Minimum 48, maximum 600.
Background color Opţional	`BACKGROUND_COL`	[color] Implicit: QColor(0, 0, 0, 0)	Choose the background color for the tiles
Enable antialiasing	`ANTIALIAS`	[boolean] Default: True	Determines if antialiasing should be enabled
Tile format	`TILE_FORMAT`	[enumerare] Implicit: 0	One of: • 0 — PNG • 1 — JPG
Quality (JPG only) Opţional	`QUALITY`	[număr] Implicit: 75	Minimum 1, maximum 100.
Metatile size Opţional	`METATILESIZE`	[număr] Implicit: 4	Specify a custom metatile size when generating XYZ tiles. Larger values may speed up the rendering of tiles and provide better labelling (fewer gaps without labels) at the expense of using more memory. Minimum 1, maximum 20.
Output file (for MBTiles)	`OUTPUT_FILE`	[file] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output file. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Output file (for MBTiles)	`OUTPUT_FILE`	[file]	The output file.

Cod Python

Algorithm ID: `native:tilesxyzmbtiles`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.19 Vector analysis

Basic statistics for fields

Generates basic statistics for a field of the attribute table of a vector layer.

Numeric, date, time and string fields are supported.

The statistics returned will depend on the field type.

Statistics are generated as an HTML file and are available in the *Processing* ► *Results viewer*.

Default menu: *Vector* ► *Analysis Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Input vector	`INPUT_LAYER`	[vector: oricare]	Vector layer to calculate the statistics on
Field to calculate statistics on	`FIELD_NAME`	[tablefield: any]	Any supported table field to calculate the statistics
Statistics Opţional	`OUTPUT_HTML_FI`	[html] Implicit: `[Salvare într-un fişier temporar]`	Specification of the file for the calculated statistics. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Statistics	`OUTPUT_HTML_FI`	[html]	HTML file with the calculated statistics
Count	`COUNT`	[număr]	
Number of unique values	`UNIQUE`	[număr]	
Number of empty (null) values	`EMPTY`	[număr]	
Number of non-empty values	`FILLED`	[număr]	
Minimum value	`MIN`	[la fel ca intrarea]	
Maximum value	`MAX`	[la fel ca intrarea]	
Minimum length	`MIN_LENGTH`	[număr]	
Maximum length	`MAX_LENGTH`	[număr]	
Mean length	`MEAN_LENGTH`	[număr]	
Coefficient of Variation	`CV`	[număr]	
Sum	`SUM`	[număr]	
Mean value	`MEAN`	[număr]	
Standard deviation	`STD_DEV`	[număr]	
Range	`RANGE`	[număr]	
Median	`MEDIAN`	[număr]	
Minority (rarest occurring value)	`MINORITY`	[la fel ca intrarea]	
Majority (most frequently occurring value)	`MAJORITY`	[la fel ca intrarea]	
First quartile	`FIRSTQUARTILE`	[număr]	
Third quartile	`THIRDQUARTILE`	[număr]	
Interquartile Range (IQR)	`IQR`	[număr]	

Cod Python

Algorithm ID: `qgis:basicstatisticsforfields`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Climb along line

Calculates the total climb and descent along line geometries. The input layer must have Z values present. If Z values are not available, the *Drape (set Z value from raster)* algorithm may be used to add Z values from a DEM layer.

The output layer is a copy of the input layer with additional fields that contain the total climb (`climb`), total descent (`descent`), the minimum elevation (`minelev`) and the maximum elevation (`maxelev`) for each line geometry. If the input layer contains fields with the same names as these added fields, they will be renamed (field names will be altered to „name_2”, „name_3”, etc, finding the first non-duplicate name).

Parametri

Etichetă	Nume	Tipul	Descriere
Line layer	`INPUT`	[vector: linie]	Line layer to calculate the climb for. Must have Z values
Climb layer	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specification of the output (line) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Climb layer	`OUTPUT`	[vector: linie]	Line layer containing new attributes with the results from climb calculations.
Total climb	`TOTALCLIMB`	[număr]	The sum of the climb for all the line geometries in the input layer
Total descent	`TOTALDESCENT`	[număr]	The sum of the descent for all the line geometries in the input layer
Minimum elevation	`MINELEVATION`	[număr]	The minimum elevation for the geometries in the layer
Maximum elevation	`MAXELEVATION`	[număr]	The maximum elevation for the geometries in the layer

Cod Python

Algorithm ID: `qgis:climbalongline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Count points in polygon

Takes a point and a polygon layer and counts the number of points from the point layer in each of the polygons of the polygon layer.

A new polygon layer is generated, with the exact same content as the input polygon layer, but containing an additional field with the points count corresponding to each polygon.

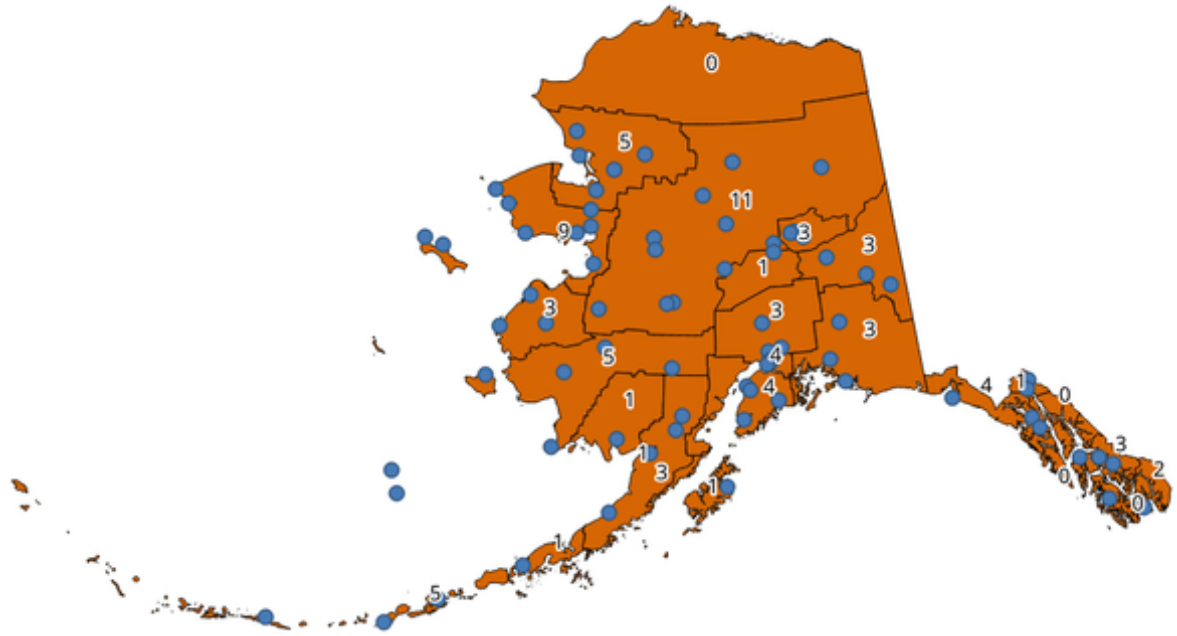


Fig. 28.39: The labels in the polygons show the point count

An optional weight field can be used to assign weights to each point. Alternatively, a unique class field can be specified. If both options are used, the weight field will take precedence and the unique class field will be ignored.

Allows *features in-place modification* of polygon features

`Default menu`: *Vector* ► *Analysis Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Polygons	`POLYGONS`	[vector: poligon]	Polygon layer whose features are associated with the count of points they contain
Points	`POINTS`	[vector: punct]	Point layer with features to count
Weight field Opţional	`WEIGHT`	[tablefield: any]	A field from the point layer. The count generated will be the sum of the weight field of the points contained by the polygon. If the weight field is not numeric, the count will be `0`.
Class field Opţional	`CLASSFIELD`	[tablefield: any]	Points are classified based on the selected attribute and if several points with the same attribute value are within the polygon, only one of them is counted. The final count of the points in a polygon is, therefore, the count of different classes that are found in it.
Count field name	`FIELD`	[string] Implicit: «NUMPOINTS»	The name of the field to store the count of points

continues on next page

Tabelul 28.95 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Count	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specification of the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Count	`OUTPUT`	[vector: poligon]	Resulting layer with the attribute table containing the new column with the points count

Cod Python

ID Algorithm: `nativ:countpointsinpolygon`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

DBSCAN clustering

Clusters point features based on a 2D implementation of Density-based spatial clustering of applications with noise (DBSCAN) algorithm.

The algorithm requires two parameters, a minimum cluster size, and the maximum distance allowed between clustered points.

Vezi și:

ST-DBSCAN clustering, *K-means clustering*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Layer to analyze
Minimum cluster size	`MIN_SIZE`	[număr] Implicit: 5	Minimum number of features to generate a cluster

continues on next page

Tabelul 28.96 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Maximum distance between clustered points	`EPS`	[număr] Implicit: 1.0	Distance beyond which two features can not belong to the same cluster (eps)
Clusters	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the vector layer for the result of the clustering. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Treat border points as noise (DBSCAN*) Opţional	`DBSCAN*`	[boolean] Implicit: *False*	If checked, points on the border of a cluster are themselves treated as unclustered points, and only points in the interior of a cluster are tagged as clustered.
Cluster field name	`FIELD_NAME`	[string] Implicit: «CLUSTER_ID»	Name of the field where the associated cluster number shall be stored
Cluster size field name	`SIZE_FIELD_NAM`	[string] Implicit: «CLUSTER_SIZE»	Name of the field with the count of features in the same cluster

Rezultat

Etichetă	Nume	Tipul	Descriere
Clusters	`OUTPUT`	[vector: punct]	Vector layer containing the original features with a field setting the cluster they belong to
Number of clusters	`NUM_CLUSTERS`	[număr]	The number of clusters discovered

Cod Python

ID Algorithm: `nativ:dbscanclustering`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Distance matrix

Calculates for point features distances to their nearest features in the same layer or in another layer.

Default menu: *Vector* ► *Analysis Tools*

Vezi și:

Join attributes by nearest

Parametri

Etichetă	Nume	Tipul	Descriere
Input point layer	`INPUT`	[vector: punct]	Point layer for which the distance matrix is calculated (**from** points)
Input unique ID field	`INPUT_FIELD`	[tablefield: any]	Field to use to uniquely identify features of the input layer. Used in the output attribute table.
Target point layer	`TARGET`	[vector: punct]	Point layer containing the nearest point(s) to search (**to** points)
Target unique ID field	`TARGET_FIELD`	[tablefield: any]	Field to use to uniquely identify features of the target layer. Used in the output attribute table.
Output matrix type	`MATRIX_TYPE`	[enumerare] Implicit: 0	Different types of calculation are available: • 0 — Matrice distanțelor liniare (N * *k* x 3): pentru fiecare punct de intrare, raportează distanța până la fiecare dintre cele mai apropiate *k* puncte țintă. Matricea rezultat constă din până la *k* rânduri per punct de intrare, iar fiecare rând are trei coloane: *InputID*, *TargetID* și *Distance*. • 1 — Matricea distanțelor standard (N x T). • 2 — Sumarul matricii distanțelor (medie, dev. std., min, max): pentru fiecare punct de intrare, raportează statisticile privind distanțele până la punctele țintă.
Use only the nearest (k) target points	`NEAREST_POINTS`	[număr] Implicit: 0	You can choose to calculate the distance to all the points in the target layer (*0*) or limit to a number (*k*) of closest features.
Distance matrix	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specification of the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Distance matrix	`OUTPUT`	[vector: punct]	Point (or MultiPoint for the „Linear (N * *k* x 3)" case) vector layer containing the distance calculation for each input feature. Its features and attribute table depend on the selected output matrix type.

Cod Python

Algorithm ID: `qgis:distancematrix`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Distance to nearest hub (line to hub)

Creates lines that join each feature of an input vector to the nearest feature in a destination layer. Distances are calculated based on the *center* of each feature.

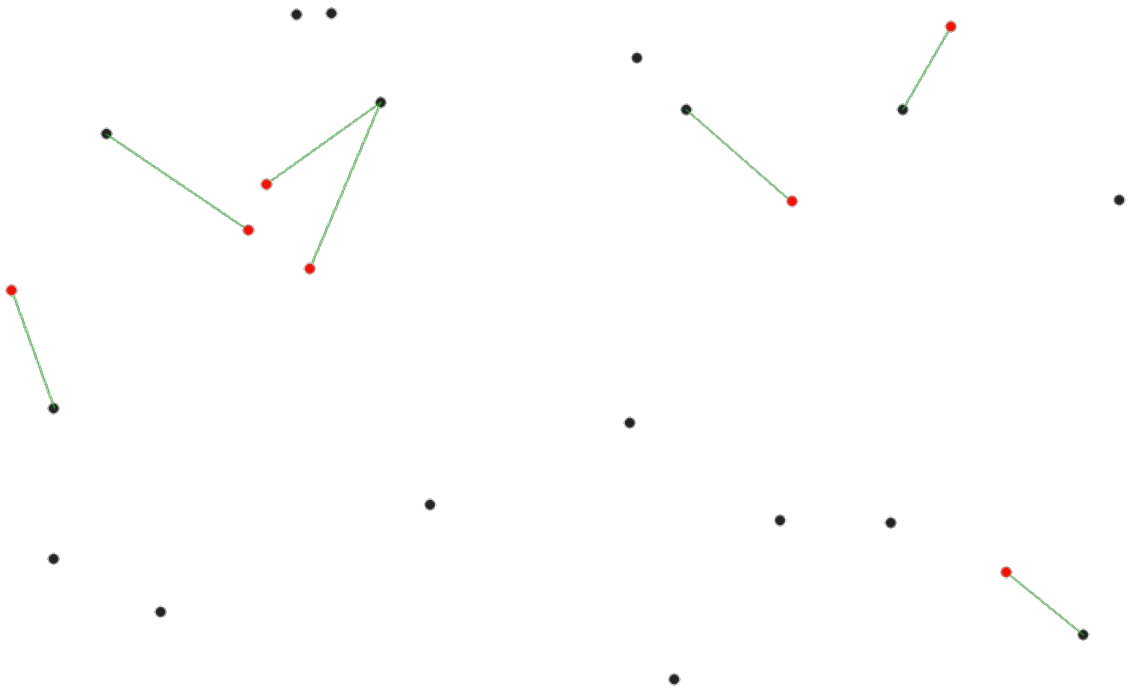


Fig. 28.40: Display the nearest hub for the red input features

Vezi și:

Distance to nearest hub (points), *Join attributes by nearest*

Parametri

Etichetă	Nume	Tipul	Descriere
Source points layer	`INPUT`	[vector: oricare]	Vector layer for which the nearest feature is searched
Destination hubs layer	`HUBS`	[vector: oricare]	Vector layer containing the features to search for
Hub layer name attribute	`FIELD`	[tablefield: any]	Field to use to uniquely identify features of the destination layer. Used in the output attribute table
Measurement unit	`UNIT`	[enumerare] Implicit: 0	Units in which to report the distance to the closest feature: • 0 — Metri • 1 — Picioare • 2 — Mile • 3 — Kilometri • 4 — Unitățile stratului
Hub distance	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line vector layer connecting the matching points. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Hub distance	`OUTPUT`	[vector: linie]	Line vector layer with the attributes of the input features, the identifier of their closest feature and the calculated distance.

Cod Python

Algorithm ID: `qgis:distancetonearesthublinetohub`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Distance to nearest hub (points)

Creates a point layer representing the *center* of the input features with the addition of two fields containing the identifier of the nearest feature (based on its center point) and the distance between the points.

Vezi și:

Distance to nearest hub (line to hub), *Join attributes by nearest*

Parametri

Etichetă	Nume	Tipul	Descriere
Source points layer	`INPUT`	[vector: oricare]	Vector layer for which the nearest feature is searched
Destination hubs layer	`HUBS`	[vector: oricare]	Vector layer containing the features to search for
Hub layer name attribute	`FIELD`	[tablefield: any]	Field to use to uniquely identify features of the destination layer. Used in the output attribute table
Measurement unit	`UNIT`	[enumerare] Implicit: 0	Units in which to report the distance to the closest feature: • 0 — Metri • 1 — Picioare • 2 — Mile • 3 — Kilometri • 4 — Unitățile stratului
Hub distance	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output point vector layer with the nearest hub. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Hub distance	`OUTPUT`	[vector: punct]	Point vector layer representing the center of the source features with their attributes, the identifier of their closest feature and the calculated distance.

Cod Python

Algorithm ID: `qgis:distancetonearesthubpoints`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Join by lines (hub lines)

Creates hub and spoke diagrams by connecting lines from points on the Spoke layer to matching points in the Hub layer.

Determination of which hub goes with each point is based on a match between the Hub ID field on the hub points and the Spoke ID field on the spoke points.

If input layers are not point layers, a point on the surface of the geometries will be taken as the connecting location.

Optionally, geodesic lines can be created, which represent the shortest path on the surface of an ellipsoid. When geodesic mode is used, it is possible to split the created lines at the antimeridian (±180 degrees longitude), which can improve rendering of the lines. Additionally, the distance between vertices can be specified. A smaller distance results in a denser, more accurate line.

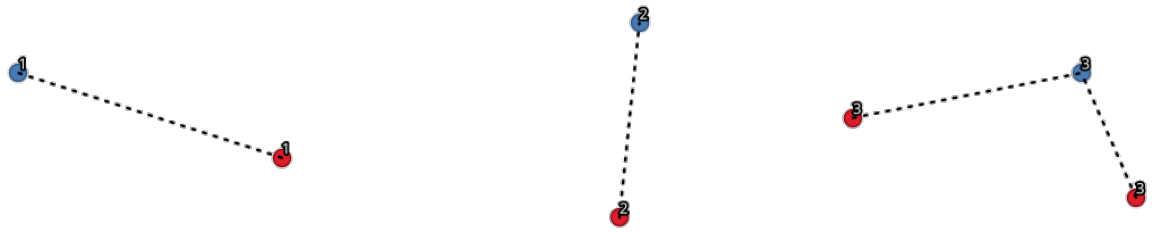


Fig. 28.41: Join points based on a common field / attribute

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Hub layer	`HUBS`	[vector: oricare]	Strat de intrare
Hub ID field	`HUB_FIELD`	[tablefield: any]	Field of the hub layer with ID to join
Hub layer fields to copy (leave empty to copy all fields) Opţional	`HUB_FIELDS`	[tablefield: any] [list]	The field(s) of the hub layer to be copied. If no field(s) are chosen all fields are taken.
Spoke layer	`SPOKES`	[vector: oricare]	Additional spoke point layer
Spoke ID field	`SPOKE_FIELD`	[tablefield: any]	Field of the spoke layer with ID to join

continues on next page

Tabelul 28.100 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Spoke layer fields to copy (leave empty to copy all fields) Opţional	SPOKE_FIELDS	[tablefield: any] [list]	Field(s) of the spoke layer to be copied. If no fields are chosen all fields are taken.
Create geodesic lines	GEODESIC	[boolean] Implicit: *False*	Create geodesic lines (the shortest path on the surface of an ellipsoid)
Hub lines	OUTPUT	[vector: linie] Implicit: [Creare Strat temporar]	Specify the output hub line vector layer. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Distance between vertices (geodesic lines only)	GEODESIC_DISTA	[număr] Implicit: 1000.0 (kilometri)	Distance between consecutive vertices (in kilometers). A smaller distance results in a denser, more accurate line
Split lines at antimeridian (±180 degrees longitude)	ANTIMERIDIAN_S	[boolean] Implicit: *False*	Split lines at ±180 degrees longitude (to improve rendering of the lines)

Rezultat

Etichetă	Nume	Tipul	Descriere
Hub lines	OUTPUT	[vector: linie]	The resulting line layer connecting matching points in input layers

Cod Python

ID Algorithm: `nativ:hublines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

K-means clustering

Calculates the 2D distance based k-means cluster number for each input feature.

K-means clustering aims to partition the features into k clusters in which each feature belongs to the cluster with the nearest mean. The mean point is represented by the barycenter of the clustered features.

If input geometries are lines or polygons, the clustering is based on the centroid of the feature.

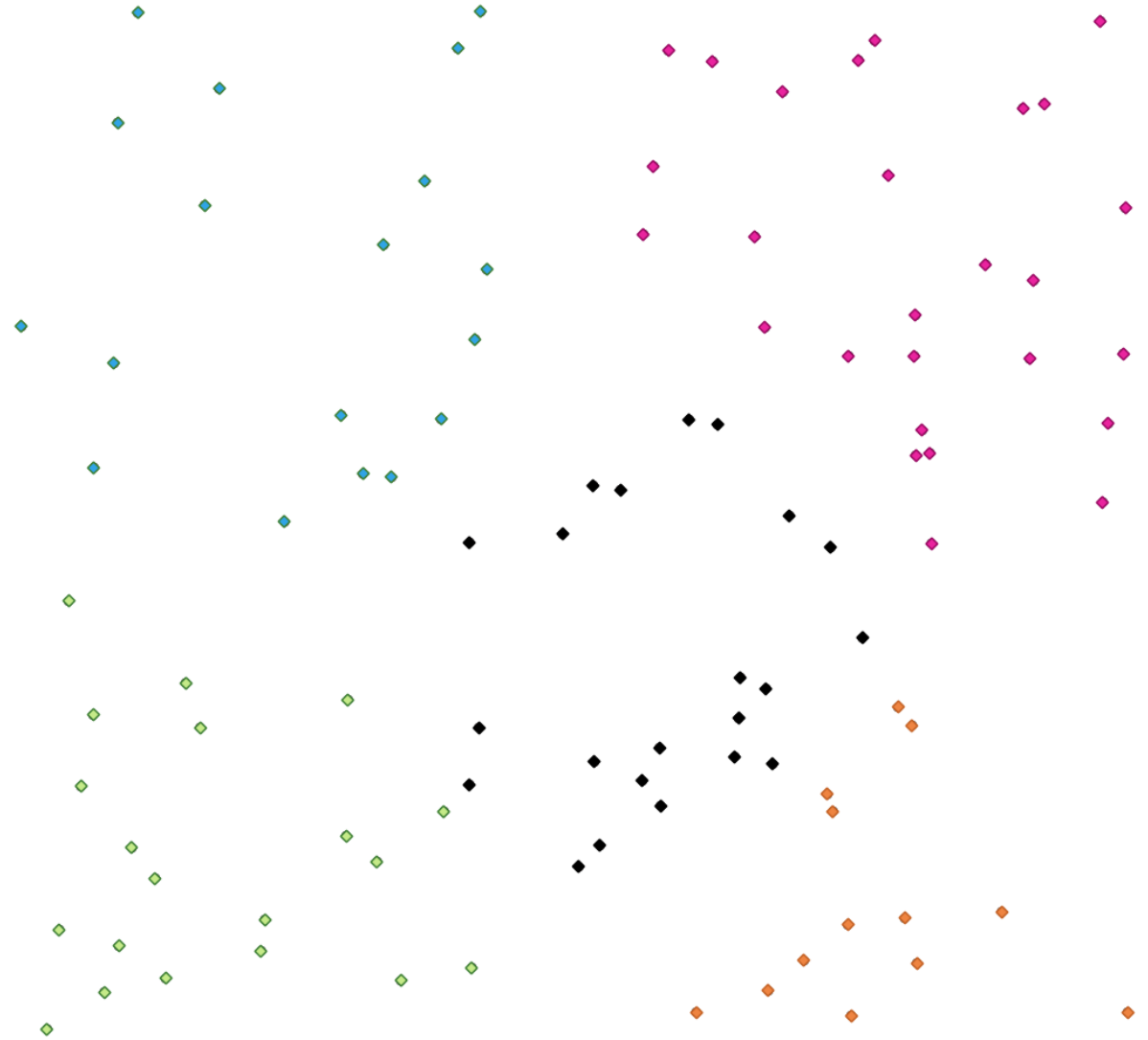


Fig. 28.42: A five class point clusters

Vezi și:

DBSCAN clustering, *ST-DBSCAN clustering*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to analyze
Number of clusters	`CLUSTERS`	[număr] Implicit: 5	Number of clusters to create with the features
Clusters	`OUTPUT`	[vector: oricare] Implicit: `[Creare strat temporar]`	Specify the output vector layer for generated the clusters. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Cluster field name	`FIELD_NAME`	[string] Implicit: «CLUSTER_ID»	Name of the field where the associated cluster number shall be stored
Cluster size field name	`SIZE_FIELD_NAM`	[string] Implicit: «CLUSTER_SIZE»	Name of the field with the count of features in the same cluster

Rezultat

Etichetă	Nume	Tipul	Descriere
Clusters	`OUTPUT`	[vector: oricare]	Vector layer containing the original features with fields specifying the cluster they belong to and their number in it

Cod Python

ID Algorithm: `nativ:kmeansclustering`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

List unique values

Lists unique values of an attribute table field and counts their number.

Default menu: *Vector* ► *Analysis Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to analyze
Target field(s)	`FIELDS`	[tablefield: any]	Field to analyze
Unique values Opţional	`OUTPUT`	[tabelă] Implicit: `[Creare strat temporar]`	Specify the summary table layer with unique values. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
HTML report Opţional	`OUTPUT_HTML_FI`	[html] Implicit: `[Salvare într-un fișier temporar]`	HTML report of unique values in the *Processing* ► *Results viewer*. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Unique values	`OUTPUT`	[tabelă]	Summary table layer with unique values
HTML report	`OUTPUT_HTML_FI`	[html]	HTML report of unique values. Can be opened from the *Processing* ► *Results viewer*
Total unique values	`TOTAL_VALUES`	[număr]	The number of unique values in the input field
Unique values concatenated	`UNIQUE_VALUES`	[string]	A string with the comma separated list of unique values found in the input field

Cod Python

Algorithm ID: `qgis:listuniquevalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Mean coordinate(s)

Computes a point layer with the center of mass of geometries in an input layer.

An attribute can be specified as containing weights to be applied to each feature when computing the center of mass.

If an attribute is selected in the parameter, features will be grouped according to values in this field. Instead of a single point with the center of mass of the whole layer, the output layer will contain a center of mass for the features in each category.

Default menu: *Vector* ► *Analysis Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Weight field Opţional	`WEIGHT`	[tablefield: numeric]	Field to use if you want to perform a weighted mean
Unique ID field	`UID`	[tablefield: numeric]	Unique field on which the calculation of the mean will be made
Mean coordinates	`OUTPUT`	[vector: punct] Implicit: `[Creare strat temporar]`	Specify the (point vector) layer for the result. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Mean coordinates	`OUTPUT`	[vector: punct]	Resulting point(s) layer

Cod Python

ID Algorithm: `nativ:meancoordinates`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

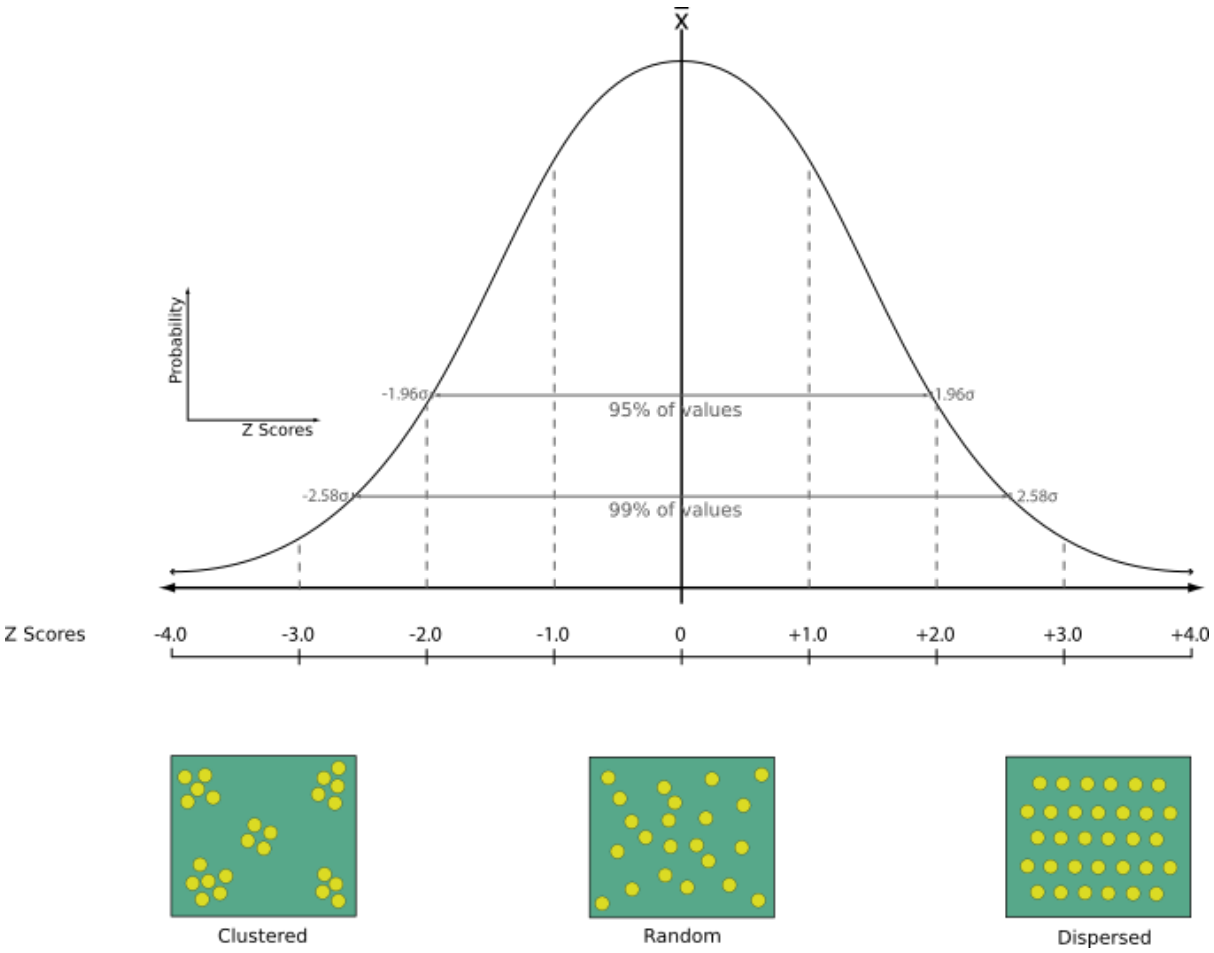
ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Nearest neighbour analysis

Performs nearest neighbor analysis for a point layer. The output tells you how your data are distributed (clustered, randomly or distributed).

Output is generated as an HTML file with the computed statistical values:

- Observed mean distance
- Expected mean distance
- Nearest neighbour index
- Number of points
- Z-Score: Comparing the Z-Score with the normal distribution tells you how your data are distributed. A low Z-Score means that the data are unlikely to be the result of a spatially random process, while a high Z-Score means that your data are likely to be a result of a spatially random process.



Default menu: *Vector* ► *Analysis Tools*

Vezi și:

Join attributes by nearest

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Point vector layer to calculate the statistics on
Nearest neighbour Opţional	`OUTPUT_HTML_FI`	[html] Implicit: `[Salvare într-un fişier temporar]`	Specification of the HTML file for the computed statistics. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Nearest neighbour	`OUTPUT_HTML_FI`	[html]	HTML file with the computed statistics
Observed mean distance	`OBSERVED_MD`	[număr]	Observed mean distance
Expected mean distance	`EXPECTED_MD`	[număr]	Expected mean distance
Nearest neighbour index	`NN_INDEX`	[număr]	Nearest neighbour index
Number of points	`POINT_COUNT`	[număr]	Number of points
Z-Score	`Z_SCORE`	[număr]	Z-Score

Cod Python

ID Algorithm: `nativ:nearestneighbouranalysis`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Overlap analysis

Calculates the area and percentage cover by which features from an input layer are overlapped by features from a selection of overlay layers.

New attributes are added to the output layer reporting the total area of overlap and percentage of the input feature overlapped by each of the selected overlay layers.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul de intrare.
Overlap layers	`LAYERS`	[vector: oricare] [list]	The overlay layers.
Overlap	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Grid size Opţional	`GRID_SIZE`	[număr] Implicit: Nesetat	If provided, the input geometries are snapped to a grid of the given size, and the result vertices are computed on that same grid. Requires GEOS 3.9.0 or higher.

Rezultat

Etichetă	Nume	Tipul	Descriere
Overlap	`OUTPUT`	[la fel ca intrarea]	The output layer with additional fields reporting the overlap (in map units and percentage) of the input feature overlapped by each of the selected layers.

Cod Python

ID Algorithm: `nativ:calculatevectoroverlaps`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Shortest line between features

Creates a line layer as the shortest line between the source and the destination layer. By default only the first nearest feature of the destination layer is taken into account. The n-nearest neighboring features number can be specified. If a maximum distance is specified, then only features which are closer than this distance will be considered.

The output features will contain all the source layer attributes, all the attributes from the n-nearest feature and the additional field of the distance.

Important: This algorithm uses purely Cartesian calculations for distance, and does not consider geodetic or ellipsoid properties when determining feature proximity. The measurement and output coordinate system is based on the coordinate system of the source layer.

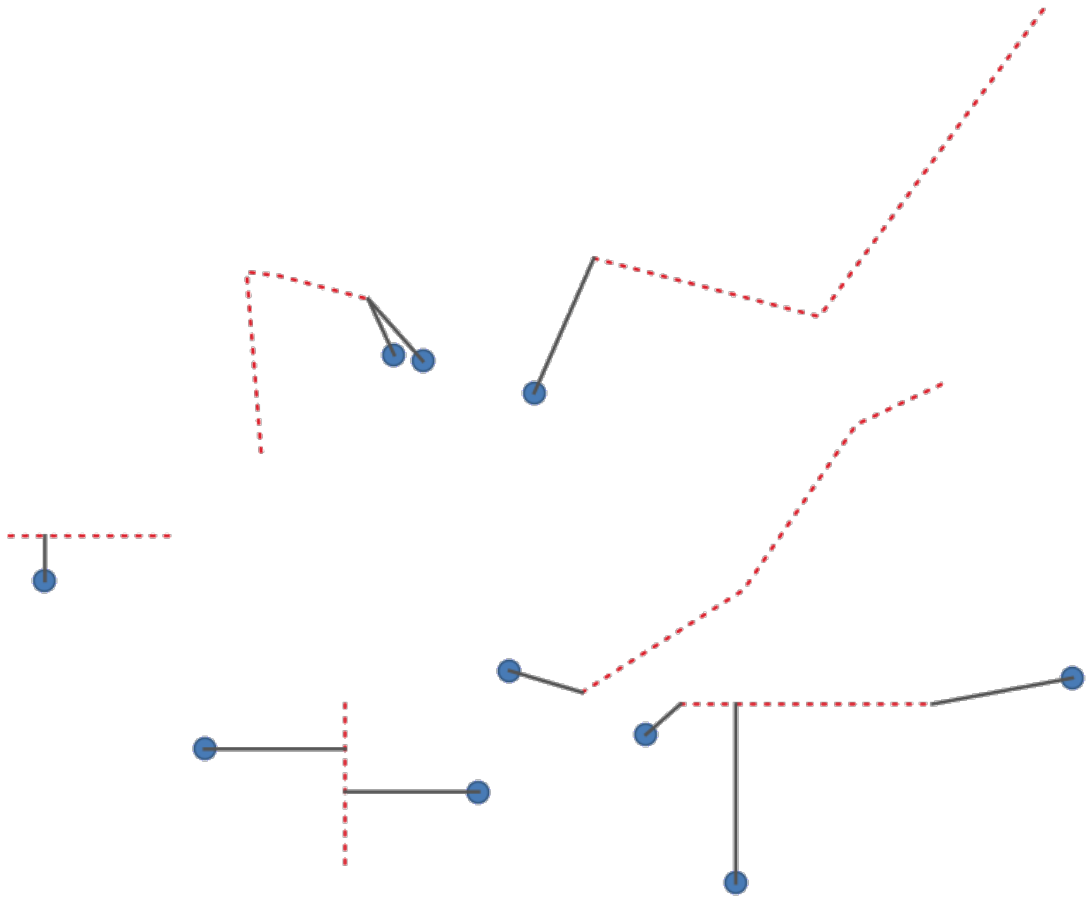


Fig. 28.43: Shortest line from point features to lines

Parametri

Etichetă	Nume	Tipul	Descriere
Source layer	`SOURCE`	[vector: oricare]	Origin layer for which to search for nearest neighbors
Destination layer	`DESTINATION`	[vector: oricare]	Target Layer in which to search for nearest neighbors
Method	`METHOD`	[enumerare] Implicit: 0	Shortest distance calculation method Possible values are: • 0 — Distanța până la cel mai apropiat punct al entității • 1 — Distanța până la centroidul entității
Maximum number of neighbors	`NEIGHBORS`	[număr] Implicit: 1	Maximum number of neighbors to look for
Maximum distance Opţional	`DISTANCE`	[număr]	Only destination features which are closer than this distance will be considered.
Shortest lines	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[vector: linie]	Line vector layer joining source features to their nearest neighbor(s) in the destination layer. Contains all attributes for both source and destination features, and the computed distance.

Cod Python

Algorithm ID: `native:shortestline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

ST-DBSCAN clustering

Clusters point features based on a 2D implementation of spatiotemporal Density-based clustering of applications with noise (ST-DBSCAN) algorithm.

Vezi și:

DBSCAN clustering, *K-means clustering*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Layer to analyze
Date/time field	`DATETIME_FIELD`	[tablefield: date]	Field containing the temporal information
Minimum cluster size	`MIN_SIZE`	[număr] Implicit: 5	Minimum number of features to generate a cluster
Maximum distance between clustered points	`EPS`	[număr] Implicit: 1.0	Distance beyond which two features can not belong to the same cluster (eps)
Maximum time duration between clustered points	`EPS2`	[număr] Implicit: 0.0 (zile)	Time duration beyond which two features can not belong to the same cluster (eps2). Available time units are milliseconds, seconds, minutes, hours, days and weeks.
Clusters	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the vector layer for the result of the clustering. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Treat border points as noise (DBSCAN*) Opţional	`DBSCAN*`	[boolean] Implicit: *False*	If checked, points on the border of a cluster are themselves treated as unclustered points, and only points in the interior of a cluster are tagged as clustered.
Cluster field name	`FIELD_NAME`	[string] Implicit: «CLUSTER_ID»	Name of the field where the associated cluster number shall be stored
Cluster size field name	`SIZE_FIELD_NAM`	[string] Implicit: «CLUSTER_SIZE»	Name of the field with the count of features in the same cluster

Rezultat

Etichetă	Nume	Tipul	Descriere
Clusters	`OUTPUT`	[vector: punct]	Vector layer containing the original features with a field setting the cluster they belong to
Number of clusters	`NUM_CLUSTERS`	[număr]	The number of clusters discovered

Cod Python

ID Algorithm: `nativ:stdbscanclustering`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Statistics by categories

Calculates statistics of a field depending on a parent class. The parent class is a combination of values from other fields.

Parametri

Etichetă	Nume	Tipul	Descriere
Input vector layer	`INPUT`	[vector: oricare]	Input vector layer with unique classes and values
Field to calculate statistics on (if empty, only count is calculated) Opţional	`VALUES_FIELD_N`	[tablefield: any]	If empty only the count will be calculated
Field(s) with categories	`CATEGORIES_FIE`	[vector: oricare] [list]	The fields that (combined) define the categories
Statistics by category	`OUTPUT`	[tabelă] Implicit: `[Creare Strat temporar]`	Specify the output table for the generated statistics. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Statistics by category	`OUTPUT`	[tabelă]	Table containing the statistics

Depending on the type of the field being analyzed, the following statistics are returned for each grouped value:

Statistici	Șirul	Numeric	Data
Count (`COUNT`)	☑	☑	☑
Unique values (`UNIQUE`)	☑		☑
Empty (null) values (`EMPTY`)	☑		☑
Non-empty values (`FILLED`)	☑		☑
Minimal value (`MIN`)	☑	☑	☑
Maximal value (`MAX`)	☑	☑	☑
Range (`RANGE`)		☑	
Sum (`SUM`)		☑	
Mean value (`MEAN`)		☑	
Median value (`MEDIAN`)		☑	
Standard Deviation (`STD_DEV`)		☑	
Coefficient of variation (`CV`)		☑	
Minority (rarest occurring value - `MINORITY`)		☑	
Majority (most frequently occurring value - `MAJORITY`)		☑	
First Quartile (`FIRSTQUARTILE`)		☑	
Third Quartile (`THIRDQUARTILE`)		☑	
Inter Quartile Range (`IQR`)		☑	
Minimum Length (`MIN_LENGTH`)	☑		
Mean Length (`MEAN_LENGTH`)	☑		
Maximum Length (`MAX_LENGTH`)	☑		

Cod Python

Algorithm ID: `qgis:statisticsbycategories`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Sum line lengths

Takes a polygon layer and a line layer and measures the total length of lines and the total number of them that cross each polygon.

The resulting layer has the same features as the input polygon layer, but with two additional attributes containing the length and count of the lines across each polygon.

Allows *features in-place modification* of polygon features

Default menu: *Vector ► Analysis Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Lines	`LINES`	[vector: linie]	Input vector line layer
Polygons	`POLYGONS`	[vector: poligon]	Polygon vector layer
Lines length field name	`LEN_FIELD`	[string] Implicit: «LENGTH»	Name of the field for the lines length
Lines count field name	`COUNT_FIELD`	[string] Implicit: «COUNT»	Name of the field for the lines count
Line length	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon layer with generated statistics. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Line length	`OUTPUT`	[vector: poligon]	Polygon output layer with fields of lines length and line count

Cod Python

ID Algorithm: `nativ:sumlinelengths`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.20 Vector creation

Array of offset (parallel) lines

Creates copies of line features in a layer, by creating multiple offset versions of each feature. Each new version is incrementally offset by a specified distance.

Positive distance will offset lines to the left, and negative distances will offset them to the right.

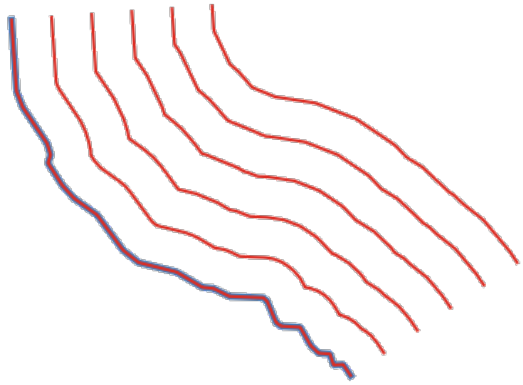


Fig. 28.44: In blue the source layer, in red the offset one

Allows *features in-place modification* of line features

Vezi și:

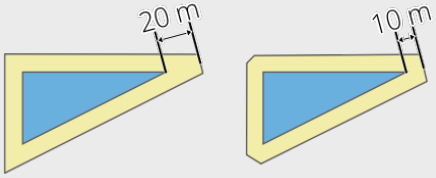
Offset lines, *Array of translated features*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer to use for the offsets.
Number of features to create	`COUNT`	[number] Implicit: 10	Number of offset copies to generate for each feature
Offset step distance	`OFFSET`	[number] Implicit: 1.0	Distance between two consecutive offset copies
Offset lines	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line layer with offset features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Segments	`SEGMENTS`	[număr] Implicit: 8	Number of line segments to use to approximate a quarter circle when creating rounded offsets
Join style	`JOIN_STYLE`	[enumerare] Implicit: 0	Specify whether round, miter or beveled joins should be used when offsetting corners in a line. One of: • 0 — Round • 1 — Miter • 2 — Bevel Fig. 28.45: Round, miter, and bevel join styles
Miter limit	`MITER_LIMIT`	[număr] Implicit: 2.0	Sets the maximum distance from the offset geometry to use when creating a mitered join as a factor of the offset distance (only applicable for miter join styles). Minimum: 1.0  Fig. 28.46: A 10m buffer with a limit of 2 and a 10m buffer with a limit of 1

Rezultat

Etichetă	Nume	Tipul	Descriere
Offset lines	`OUTPUT`	[vector: linie]	Output line layer with offset features. The original features are also copied.

Cod Python

ID Algorithm: `nativ:arrayoffsetlines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Array of translated features

Creates copies of features in a layer by creating multiple translated versions of each. Each copy is incrementally displaced by a preset amount in the X, Y and/or Z axis.

M values present in the geometry can also be translated.

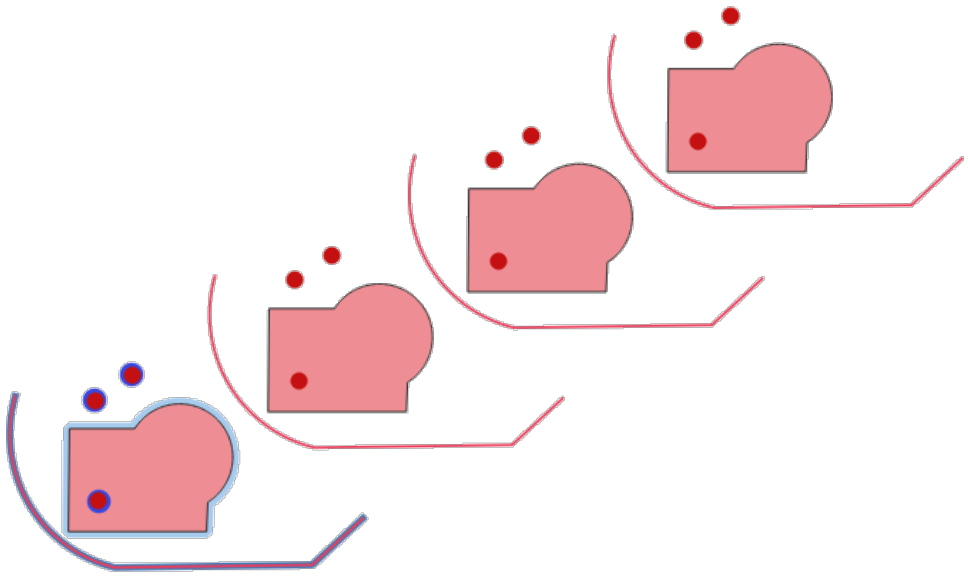


Fig. 28.47: Input layers in blue tones, output layers with translated features in red tones

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Translate, *Array of offset (parallel) lines*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer to translate
Number of features to create	`COUNT`	[number] Implicit: 10	Number of copies to generate for each feature
Step distance (x-axis)	`DELTA_X`	[number] Implicit: 0.0	Displacement to apply on the X axis
Step distance (y-axis)	`DELTA_Y`	[number] Implicit: 0.0	Displacement to apply on the Y axis
Step distance (z-axis)	`DELTA_Z`	[number] Implicit: 0.0	Displacement to apply on the Z axis
Step distance (m values)	`DELTA_M`	[number] Implicit: 0.0	Displacement to apply on M

continues on next page

Tabelul 28.110 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Translated	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Output vector layer with translated (moved) copies of the features. The original features are also copied. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Translated	`OUTPUT`	[la fel ca intrarea]	Output vector layer with translated (moved) copies of the features. The original features are also copied.

Cod Python

ID Algorithm: `nativ:arraytranslatedfeatures`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create grid

Creates a vector layer with a grid covering a given extent. Grid cells can have different shapes:

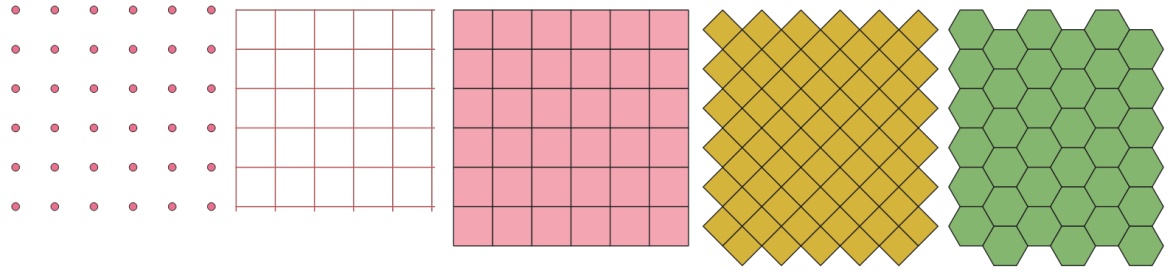


Fig. 28.48: Different grid cell shapes applied to the same extent, without overlaps

The size and/or placement of each element in the grid is defined using a horizontal and vertical spacing. The CRS of the output layer must be defined. The grid extent and the spacing values must be expressed in the coordinates and units of this CRS.

Default menu: *Vector ► Research Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Grid type	`TYPE`	[enumerare] Implicit: 0	Shape of the grid. One of: • 0 — Point • 1 — Line • 2 — Dreptunghi (poligon) • 3 — Diamant (poligon) • 4 — Hexagon (poligon)
Grid extent	`EXTENT`	[extent]	Extent of the grid Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Horizontal spacing	`HSPACING`	[număr] Implicit: 1.0	Size of a grid cell on the X-axis
Vertical spacing	`VSPACING`	[număr] Implicit: 1.0	Size of a grid cell on the Y-axis
Horizontal overlay	`HOVERLAY`	[număr] Implicit: 0.0	Overlay distance between two consecutive grid cells on the X-axis
Vertical overlay	`VOVERLAY`	[număr] Implicit: 0.0	Overlay distance between two consecutive grid cells on the Y-axis
Grid CRS	`CRS`	[crs] Implicit: *CRS-ul proiectului*	Coordinate reference system to apply to the grid
Grid	`OUTPUT`	[vector: oricare] Implicit: `[Creare Strat temporar]`	Resulting vector grid layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Grid	`OUTPUT`	[vector: oricare]	Resulting vector grid layer. The output geometry type (point, line or polygon) depends on the *Grid type*. Features are created from top to bottom, left to right. The attribute table is filled with: • an `id` • coordinates on the `left`, `right`, `top` and `bottom` sides • and their placement in the grid: `row_index` and `column_index` (available for point, rectangle and hexagon grid types)

Cod Python

ID Algorithm: `nativ:creategrid`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create points layer from table

Creates points layer from a table with columns that contain coordinates fields.

Besides X and Y coordinates you can also specify Z and M fields.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer or a table.
X field	`XFIELD`	[tablefield: any]	Field containing the X coordinate
Y field	`YFIELD`	[tablefield: any]	Field containing the Y coordinate
Z field Opţional	`ZFIELD`	[tablefield: any]	Field containing the Z coordinate
M field Opţional	`MFIELD`	[tablefield: any]	Field containing the M value
CRS-ul destinației	`TARGET_CRS`	[crs] Default: `EPSG:4326`	Coordinate reference system to use for layer. The provided coordinates are assumed to be compliant.

continues on next page

Tabelul 28.112 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Points from table	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the resulting point layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Points from table	`OUTPUT`	[vector: punct]	The resulting point layer

Cod Python

ID Algorithm: `nativ:createpointslayerfromtable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Generate points (pixel centroids) along line

Generates a point vector layer from an input raster and line layer.

The points correspond to the pixel centroids that intersect the line layer.

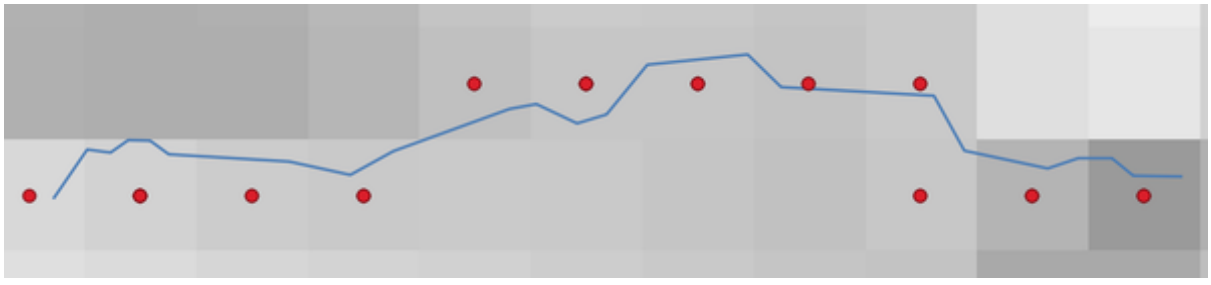


Fig. 28.49: Points of the pixel centroids

Parametri

Etichetă	Nume	Tipul	Descriere
Strat raster	`INPUT_RASTER`	[raster]	Stratul raster de intrare
Vector layer	`INPUT_VECTOR`	[vector: linie]	Input line vector layer
Points along line	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Resulting point layer with pixel centroids. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Points along line	`OUTPUT`	[vector: punct]	Resulting point layer with pixel centroids

Cod Python

Algorithm ID: `qgis:generatepointspixelcentroidsalongline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Generate points (pixel centroids) inside polygon

Generates a point vector layer from an input raster and polygon layer.

The points correspond to the pixel centroids that intersect the polygon layer.

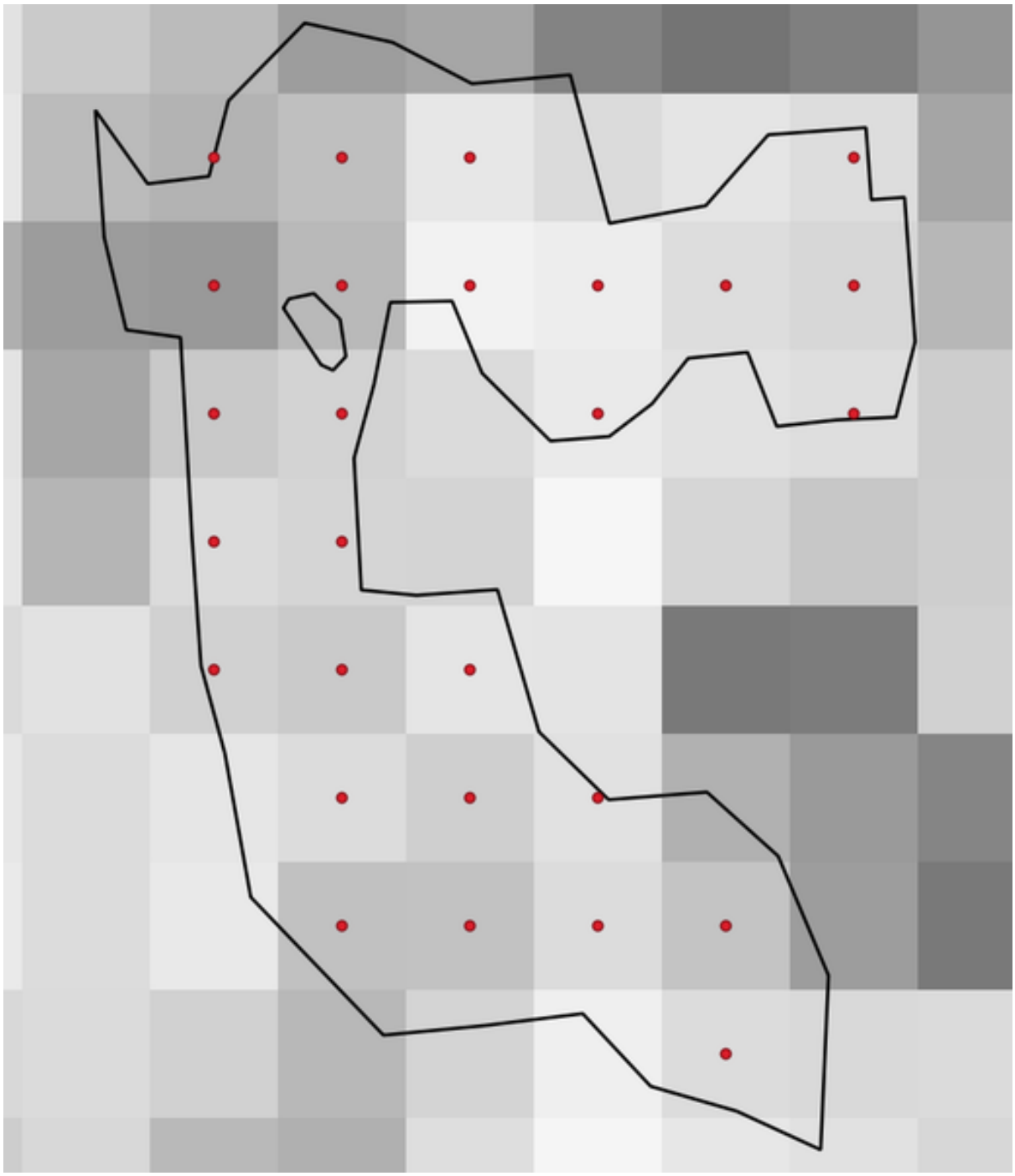


Fig. 28.50: Points of the pixel centroids

Parametri

Etichetă	Nume	Tipul	Descriere
Strat raster	`INPUT_RASTER`	[raster]	Stratul raster de intrare
Vector layer	`INPUT_VECTOR`	[vector: poligon]	Input polygon vector layer
Points inside polygons	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Resulting point layer of pixel centroids. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Points inside polygons	`OUTPUT`	[vector: punct]	Resulting point layer of pixel centroids

Cod Python

ID Algorithm: `nativ:generatepointspixelcentroidsinsidepolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Import geotagged photos

Creates a point layer corresponding to the geotagged locations from JPEG or HEIC/HEIF images from a source folder.

The point layer will contain a single PointZ feature per input file from which the geotags could be read. Any altitude information from the geotags will be used to set the point's Z value.

Besides longitude and latitude also altitude, direction and timestamp information, if present in the photo, will be added to the point as attributes.

Parametri

Etichetă	Nume	Tipul	Descriere
Input folder	`FOLDER`	[folder]	Path to the source folder containing the geotagged photos
Scan recursively	`RECURSIVE`	[boolean] Implicit: *False*	If checked, the folder and its subfolders will be scanned
Photos Opţional	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the point vector layer for the geotagged photos. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Invalid photos table Opţional	`INVALID`	[tabelă] Implicit: `[Omitere rezultat]`	Specify the table of unreadable or non-geotagged photos. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Photos	`OUTPUT`	[vector: punct]	Point vector layer with geotagged photos. The form of the layer is automatically filled with paths and photo previews settings.
Invalid photos table Opţional	`INVALID`	[tabelă]	Table of unreadable or non-geotagged photos can also be created.

Cod Python

ID Algorithm: `nativ:importphotos`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Points to path

Converts a point layer to a line layer, by joining points in an order defined by an expression or a field in the input point layer.

Points can be grouped by a field or an expression to distinguish line features.

In addition to the line vector layer, a text file is output that describes the resulting line as a start point and a sequence of bearings / directions (relative to azimuth) and distances.

Parametri

Etichetă	Nume	Tipul	Descriere
Input point layer	`INPUT`	[vector: punct]	Input point vector layer
Create closed paths	`CLOSE_PATH`	[boolean] Implicit: *False*	If checked, the first and last points of the line will be connected and close the generated path
Order expression Opţional	`ORDER_EXPRESSI`	[expresie]	Field or expression providing the order to connect the points in the path. If not set, the feature ID (`$id`) is used.
Sort text containing numbers naturally Opţional	`NATURAL_SORT`	[boolean] Implicit: *False*	If checked, naturally sorts the features based on the provided expression (i.e., «a9» < «a10»).
Path group expression Opţional	`GROUP_EXPRESSI`	[expresie]	Point features of the same value in the field or expression will be grouped in the same line. If not set, a single path is drawn with all the input points.
Paths	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the line vector layer of the path. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Directory for text output Opţional	`OUTPUT_TEXT_DI`	[folder] Implicit: `[Omitere rezultat]`	Specify the directory that will contain the description files of points and paths. One of: • Skip Output • Salvare Intr-un Director Temporar • Salvare în Director

Rezultat

Etichetă	Nume	Tipul	Descriere
Paths	`OUTPUT`	[vector: linie]	Line vector layer of the path
Directory for text output	`OUTPUT_TEXT_DI`	[folder]	Directory containing description files of points and paths

Cod Python

ID Algorithm: `nativ:pointstopath`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Puncte aleatorii de-a lungul liniei

Creates a new point layer, with points placed on the lines of another layer.

For each line in the input layer, a given number of points is added to the resulting layer. The procedure for adding a point is to:

1. randomly select a line feature from the input layer
2. if the feature is multi-part, randomly select a part of it
3. randomly select a segment of that line
4. randomly select a position on that segment.

The procedure means that curved parts of the lines (with relatively short segments) will get more points than straight parts (with relatively long segments), as demonstrated in the illustration below, where the output of the *Random points along lines* algorithm can be compared with the output of the *Random points on lines* algorithm (that produces points with an, on average, even distribution along the lines).

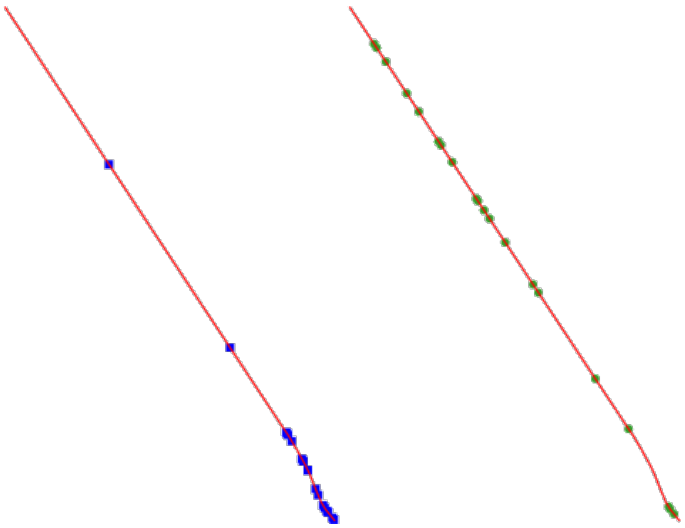


Fig. 28.51: Example algorithm output. Left: *Random points along line*, right: *Random points on lines*

A minimum distance can be specified, to avoid points being too close to each other.

Vezi și:

Random points on lines

Parametri

Etichetă	Nume	Tipul	Descriere
Input point layer	`INPUT`	[vector: linie]	Input line vector layer
Number of points	`POINTS_NUMBER`	[număr] Implicit: 1	Number of points to create
Minimum distance between points	`MIN_DISTANCE`	[număr] Implicit: 0.0	The minimum distance between points
Random points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	The output random points. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Random points	`OUTPUT`	[vector: punct]	The output random points layer.

Cod Python

Algorithm ID: `qgis:qgisrandompointsalongline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random points in extent

Creates a new point layer with a given number of random points, all of them within a given extent.

A distance factor can be specified, to avoid points being too close to each other. If the minimum distance between points makes it impossible to create new points, either distance can be decreased or the maximum number of attempts may be increased.

Default menu: *Vector ► Research Tools*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input extent	`EXTENT`	[extent]	Map extent for the random points Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Number of points	`POINTS_NUMBER`	[număr] Implicit: 1	Number of point to create
Minimum distance between points	`MIN_DISTANCE`	[număr] Implicit: 0.0	The minimum distance between points
CRS-ul destinației	`TARGET_CRS`	[crs] Implicit: *CRS-ul proiectului*	CRS of the random points layer
Random points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	The output random points. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Maximum number of search attempts given the minimum distance	`MAX_ATTEMPTS`	[număr] Implicit: 200	Maximum number of attempts to place the points

Rezultat

Etichetă	Nume	Tipul	Descriere
Random points	`OUTPUT`	[vector: punct]	The output random points layer.

Cod Python

ID Algorithm: `nativ:randompointsinextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random points in layer bounds

Creates a new point layer with a given number of random points, all of them within the extent of a given layer.

A minimum distance can be specified, to avoid points being too close to each other.

Default menu: *Vector ► Research Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Input polygon layer defining the area
Number of points	`POINTS_NUMBER`	[număr] Implicit: 1	Number of points to create
Minimum distance between points	`MIN_DISTANCE`	[număr] Implicit: 0.0	The minimum distance between points
Random points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	The output random points. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Random points	`OUTPUT`	[vector: punct]	The output random points layer.

Cod Python

Algorithm ID: `qgis:randompointsinlayerbounds`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random points in polygons

Creates a point layer with points placed inside the polygons of another layer.

For each feature (polygon / multi-polygon) geometry in the input layer, the given number of points is added to the result layer.

Per feature and global minimum distances can be specified in order to avoid points being too close in the output point layer. If a minimum distance is specified, it may not be possible to generate the specified number of points for each feature. The total number of generated points and missed points are available as output from the algorithm.

The illustration below shows the effect of per feature and global minimum distances and zero/non-zero minimum distances (generated with the same seed, so at least the first point generated will be the same).

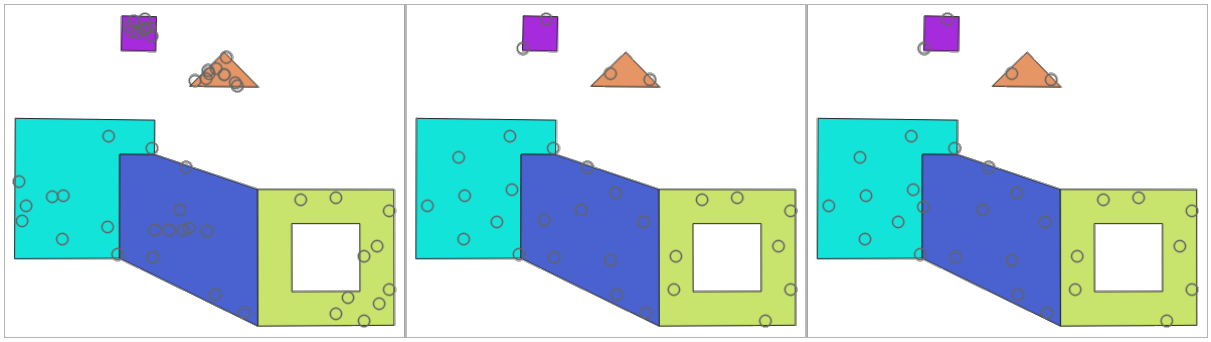


Fig. 28.52: Ten points per polygon feature, *left*: min. distances = 0, *middle*: min.distances = 1, *right*: min. distance = 1, global min. distance = 0

The maximum number of tries per point can be specified. This is only relevant for non-zero minimum distance.

A seed for the random number generator can be provided, making it possible to get identical random number sequences for different runs of the algorithm.

The attributes of the polygon feature on which a point was generated can be included (*Include polygon attributes*).

If you want approximately the same point density for all the features, you can data-define the number of points using the area of the polygon feature geometry.

Vezi și:

Random points inside polygons

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input polygon layer	`INPUT`	[vector: linie]	Input polygon vector layer
Number of points for each feature	`POINTS_NUMBER`	[number] Implicit: 1	Number of points to create
Minimum distance between points Opţional	`MIN_DISTANCE`	[number] Implicit: 0.0	The minimum distance between points within one polygon feature
Random points in polygons	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	The output random points. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Global minimum distance between points Opţional	`MIN_DISTANCE_G`	[number] Implicit: 0.0	The global minimum distance between points. Should be smaller than the *Minimum distance between points (per feature)* for that parameter to have an effect.
Maximum number of search attempts (for Min. dist. > 0) Opţional	`MAX_TRIES_PER_`	[number] Implicit: 10	The maximum number of tries per point. Only relevant if the minimum distance between points is set (and greater than 0).
Random seed Opţional	`SEED`	[număr] Implicit: Nesetat	The seed to use for the random number generator.
Include polygon attributes	`INCLUDE_POLYGO`	[boolean] Implicit: True	If set, a point will get the attributes from the line on which it is placed.

Rezultat

Etichetă	Nume	Tipul	Descriere
Random points in polygons	`OUTPUT`	[vector: punct]	The output random points layer.
Number of features with empty or no geometry	`FEATURES_WITH_`	[număr]	

continues on next page

Tabelul 28.118 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Total number of points generated	`OUTPUT_POINTS`	[număr]	
Number of missed points	`POINTS_MISSED`	[număr]	The number of points that could not be generated due to the minimum distance constraint.
Number of features with missed points	`POLYGONS_WITH_`	[număr]	Not including features with empty or no geometry

Cod Python

ID Algorithm: `nativ:randompointsinpolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random points inside polygons

Creates a new point layer with a given number of random points inside each polygon of the input polygon layer.

Two sampling strategies are available:

- Points count: number of points for each feature
- Points density: density of points for each feature

A minimum distance can be specified, to avoid points being too close to each other.

Default menu: *Vector ► Research Tools*

Vezi și:

Random points in polygons

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Input polygon vector layer
Sampling strategy	`STRATEGY`	[enumerare] Implicit: 0	Sampling strategy to use. One of: • 0 — Total puncte: numărul de puncte pentru fiecare entitate • 1 — Densitatea punctelor: densitatea punctelor pentru fiecare entitate
Point count or density	`VALUE`	[number] Implicit: 1.0	The number or density of points, depending on the chosen *Sampling strategy*.
Minimum distance between points	`MIN_DISTANCE`	[număr] Implicit: 0.0	The minimum distance between points

continues on next page

Tabelul 28.119 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Random points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	The output random points. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Random points	`OUTPUT`	[vector: punct]	The output random points layer.

Cod Python

Algorithm ID: `qgis:randompointsinsidepolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random points on lines

Creates a point layer with points placed on the lines of another layer.

For each feature (line / multi-line) geometry in the input layer, the given number of points is added to the result layer.

Per feature and global minimum distances can be specified in order to avoid points being too close in the output point layer. If a minimum distance is specified, it may not be possible to generate the specified number of points for each feature. The total number of generated points and missed points are available as output from the algorithm.

The illustration below shows the effect of per feature and global minimum distances and zero/non-zero minimum distances (generated with the same seed, so at least the first point generated will be the same).

Fig. 28.53: Five points per line feature, *left*: min. distances = 0, *middle*: min.distances != 0, *right*: min. distance != 0, global min. distance = 0

The maximum number of tries per point can be specified. This is only relevant for non-zero minimum distance.

A seed for the random number generator can be provided, making it possible to get identical random number sequences for different runs of the algorithm.

The attributes of the line feature on which a point was generated can be included (*Include line attributes*).

If you want approximately the same point density for all the line features, you can data-define the number of points using the length of the line feature geometry.

Vezi şi:

Puncte aleatorii de-a lungul liniei

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Stratul liniar de intrare	`INPUT`	[vector: linie]	Input line vector layer
Number of points for each feature	`POINTS_NUMBER`	[number] Implicit: 1	Number of points to create

continues on next page

Tabelul 28.120 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Minimum distance between points (per feature) Opţional	MIN_DISTANCE	[number] Implicit: 0.0	The minimum distance between points within one line feature
Random points on lines	OUTPUT	[vector: punct] Implicit: [Creare Strat temporar]	The output random points. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansaţi

Etichetă	Nume	Tipul	Descriere
Global minimum distance between points Opţional	MIN_DISTANCE_G	[number] Implicit: 0.0	The global minimum distance between points. Should be smaller than the *Minimum distance between points (per feature)* for that parameter to have an effect.
Maximum number of search attempts (for Min. dist. > 0) Opţional	MAX_TRIES_PER_	[number] Implicit: 10	The maximum number of tries per point. Only relevant if the minimum distance between points is set (and greater than 0).
Random seed Opţional	SEED	[număr] Implicit: Nesetat	The seed to use for the random number generator.
Include line attributes	INCLUDE_LINE_A	[boolean] Implicit: True	If set, a point will get the attributes from the line on which it is placed.

Rezultat

Etichetă	Nume	Tipul	Descriere
Random points on lines	OUTPUT	[vector: punct]	The output random points layer.
Number of features with empty or no geometry	FEATURES_WITH_	[număr]	
Number of features with missed points	LINES_WITH_MIS	[număr]	Not including features with empty or no geometry
Total number of points generated	POINTS_GENERAT	[număr]	

continues on next page

Tabelul 28.122 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Number of missed points	`POINTS_MISSED`	[număr]	The number of points that could not be generated due to the minimum distance constraint.

Cod Python

ID Algorithm: `nativ:randompointsonlines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster pixels to points

Creates a vector layer of points corresponding to each pixel in a raster layer.

Converts a raster layer to a vector layer, by creating point features for each individual pixel's center in the raster layer. Any NoData pixels are skipped in the output.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat raster	`INPUT_RASTER`	[raster]	Stratul raster de intrare
Numărul benzii	`RASTER_BAND`	[raster band]	Raster band to extract data from
Field name	`FIELD_NAME`	[string] Implicit: «VALUE»	Name of the field to store the raster band value
Vector points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the resulting point layer of pixels centroids. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Vector points	`OUTPUT`	[vector: punct]	Resulting point layer with pixels centroids

Cod Python

ID Algorithm: `nativ:pixelstopoints`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster pixels to polygons

Creates a vector layer of polygons corresponding to each pixel in a raster layer.

Converts a raster layer to a vector layer, by creating polygon features for each individual pixel's extent in the raster layer. Any NoData pixels are skipped in the output.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat raster	`INPUT_RASTER`	[raster]	Stratul raster de intrare
Numărul benzii	`RASTER_BAND`	[raster band]	Raster band to extract data from
Field name	`FIELD_NAME`	[string] Implicit: «VALUE»	Name of the field to store the raster band value
Vector polygons	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the resulting polygon layer of pixel extents. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Vector polygons	`OUTPUT`	[vector: poligon]	Resulting polygon layer of pixel extents

Cod Python

ID Algorithm: `nativ:pixelstopolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Regular points

Creates a new point layer with its points placed in a regular grid within a given extent.

The grid is specified either by the spacing between the points (same spacing for all dimensions) or by the number of points to generate. In the latter case, the spacing will be determined from the extent. In order to generate a full rectangular grid, at least the number of points specified by the user is generated for the latter case.

Random offsets to the point spacing can be applied, resulting in a non-regular point pattern.

Default menu: *Vector ► Research Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Input extent (xmin, xmax, ymin, ymax)	`EXTENT`	[extent]	Map extent for the random points Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Point spacing/count	`SPACING`	[număr] Implicit: 100	Spacing between the points, or the number of points, depending on whether `Use point spacing` is checked or not.
Initial inset from corner (LH side)	`INSET`	[număr] Implicit: 0.0	Offsets the points relative to the upper left corner. The value is used for both the X and Y axis.
Apply random offset to point spacing	`RANDOMIZE`	[boolean] Implicit: *False*	If checked the points will have a random spacing
Use point spacing	`IS_SPACING`	[boolean] Implicit: True	If unchecked the point spacing is not taken into account
Output layer CRS	`CRS`	[crs] Implicit: *CRS-ul proiectului*	CRS of the random points layer
Regular points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output regular point layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Regular points	`OUTPUT`	[vector: punct]	The output regular point layer.

Cod Python

Algorithm ID: `qgis:regularpoints`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.21 Vectori, în general

Asignare proiecție

Asignează o nouă proiecție unui strat vectorial.

It creates a new layer with the exact same features and geometries as the input one, but assigned to a new CRS. The geometries are **not** reprojected, they are just assigned to a different CRS.

This algorithm can be used to repair layers which have been assigned an incorrect projection.

Attributes are not modified by this algorithm.

Vezi și:

Define Shapefile projection, *Find projection*, *Reproject layer*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer with wrong or missing CRS
Assigned CRS	`CRS`	[crs] Default: `EPSG:4326 - WGS84`	Select the new CRS to assign to the vector layer
Assigned CRS Opţional	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Assigned CRS	`OUTPUT`	[la fel ca intrarea]	Vector layer with assigned projection

Cod Python

ID Algorithm: `nativ:assignprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Batch Nominatim geocoder

Performs batch geocoding using the Nominatim service against an input layer string field. The output layer will have a point geometry reflecting the geocoded location as well as a number of attributes associated to the geocoded location.

Allows *features in-place modification* of point features

Notă: This algorithm is compliant with the usage policy of the Nominatim geocoding service provided by the OpenStreetMap Foundation.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer to geocode the features
Address field	`FIELD`	[tablefield: string]	Field containing the addresses to geocode
Geocoded	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output layer containing only the geocoded addresses. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Geocoded	`OUTPUT`	[vector: punct]	Vector layer with point features corresponding to the geocoded addresses

Cod Python

ID Algorithm: `nativ:batchnominatimgeocoder`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Convert layer to spatial bookmarks

Creates spatial bookmarks corresponding to the extent of features contained in a layer.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: linie, poligon]	The input vector layer
Bookmark destination	`DESTINATION`	[enumerare] Implicit: 0	Select the destination for the bookmarks. One of: • 0 — Marcajele proiectului • 1 — Marcajele utilizatorului
Name field	`NAME_EXPRESSIO`	[expresie]	Field or expression that will give names to the generated bookmarks
Group field	`GROUP_EXPRESSI`	[expresie]	Field or expression that will provide groups for the generated bookmarks

Rezultat

Etichetă	Nume	Tipul	Descriere
Count of bookmarks added	`COUNT`	[număr]	

Cod Python

ID Algorithm: `nativ:layertobookmarks`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Convert spatial bookmarks to layer

Creates a new layer containing polygon features for stored spatial bookmarks. The export can be filtered to only bookmarks belonging to the current project, to all user bookmarks, or a combination of both.

Parametri

Etichetă	Nume	Tipul	Descriere
Bookmark source	`SOURCE`	[enumeration] [list] Implicit: [0,1]	Select the source(s) of the bookmarks. One or more of: • 0 — Marcajele proiectului • 1 — Marcajele utilizatorului
Output CRS	`CRS`	[crs] Default: `EPSG:4326 - WGS 84`	The CRS of the output layer
Rezultat	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Rezultat	`OUTPUT`	[vector: poligon]	The output (bookmarks) vector layer

Cod Python

ID Algorithm: `nativ:bookmarkstolayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create attribute index

Creates an index against a field of the attribute table to speed up queries. The support for index creation depends on both the layer's data provider and the field type.

No outputs are created: the index is stored on the layer itself.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Select the vector layer you want to create an attribute index for
Attribute to index	`FIELD`	[tablefield: any]	Field of the vector layer

Rezultat

Etichetă	Nume	Tipul	Descriere
Indexed layer	`OUTPUT`	[la fel ca intrarea]	A copy of the input vector layer with an index for the specified field

Cod Python

ID Algorithm: `nativ:createattributeindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create spatial index

Creates an index to speed up access to the features in a layer based on their spatial location. Support for spatial index creation is dependent on the layer's data provider.

No new output layers are created.

Default menu: *Vector ► Data Management Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare

Rezultat

Etichetă	Nume	Tipul	Descriere
Indexed layer	`OUTPUT`	[la fel ca intrarea]	A copy of the input vector layer with a spatial index

Cod Python

ID Algorithm: `nativ:createspatialindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Define Shapefile projection

Sets the CRS (projection) of an existing Shapefile format dataset to the provided CRS. It is very useful when a Shapefile format dataset is missing the `prj` file and you know the correct projection.

Contrary to the *Asignare proiecție* algorithm, it modifies the current layer and will not output a new layer.

Notă: For Shapefile datasets, the `.prj` and `.qpj` files will be overwritten - or created if missing - to match the provided CRS.

Default menu: *Vector ► Data Management Tools*

Vezi și:

Asignare proiecție, *Find projection*, *Reproject layer*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer with missing projection information
CRS	`CRS`	[crs]	Select the CRS to assign to the vector layer

Rezultat

Etichetă	Nume	Tipul	Descriere
	`INPUT`	[la fel ca intrarea]	The input vector layer with the defined projection

Cod Python

Algorithm ID: `qgis:definecurrentprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Delete duplicate geometries

Finds and removes duplicated geometries.

Attributes are not checked, so in case two features have identical geometries but different attributes, only one of them will be added to the result layer.

Vezi și:

Drop geometries, *Remove null geometries*, *Delete duplicates by attribute*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The layer with duplicate geometries you want to clean
Cleaned	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Count of discarded duplicate records	`DUPLICATE_COUN`	[număr]	Count of discarded duplicate records
Cleaned	`OUTPUT`	[la fel ca intrarea]	The output layer without any duplicated geometries
Count of retained records	`RETAINED_COUNT`	[număr]	Count of unique records

Cod Python

ID Algorithm: `nativ:deleteduplicategeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Delete duplicates by attribute

Deletes duplicate rows by only considering the specified field / fields. The first matching row will be retained, and duplicates will be discarded.

Optionally, these duplicate records can be saved to a separate output for analysis.

Vezi și:

Delete duplicate geometries

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul de intrare
Fields to match duplicates by	`FIELDS`	[tablefield: any] [list]	Fields defining duplicates. Features with identical values for all these fields are considered duplicates.
Filtered (no duplicates)	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output layer containing the unique features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

continues on next page

Tabelul 28.124 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Filtered (duplicates) Opţional	`DUPLICATES`	[la fel ca intrarea] Implicit: `[Omitere rezultat]`	Specify the output layer containing only the duplicates. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Filtered (duplicates) Opţional	`DUPLICATES`	[la fel ca intrarea] Implicit: `[Omitere rezultat]`	Vector layer containing the removed features. Will not be produced if not specified (left as `[Skip output]`).
Count of discarded duplicate records	`DUPLICATE_COUN`	[număr]	Count of discarded duplicate records
Filtered (no duplicates)	`OUTPUT`	[la fel ca intrarea]	Vector layer containing the unique features.
Count of retained records	`RETAINED_COUNT`	[număr]	Count of unique records

Cod Python

ID Algorithm: `nativ:removeduplicatesbyattribute`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Detect dataset changes

Compares two vector layers, and determines which features are unchanged, added or deleted between the two. It is designed for comparing two different versions of the same dataset.

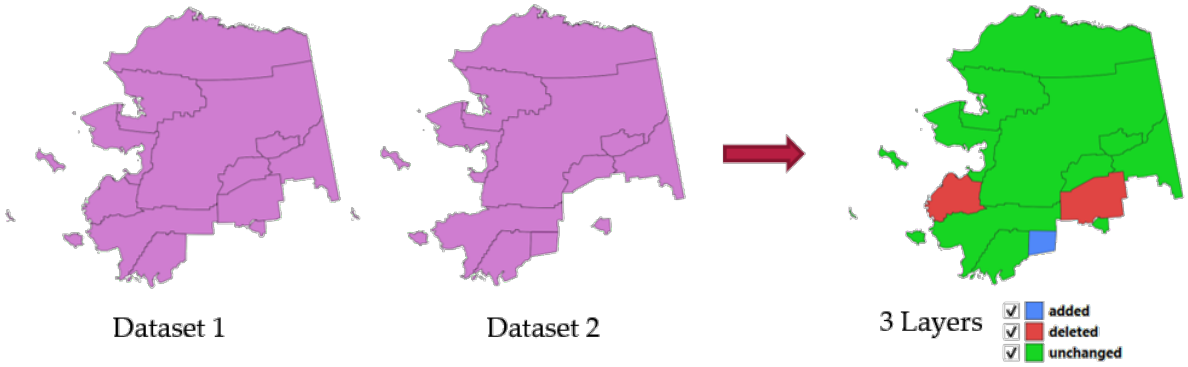


Fig. 28.54: Detect dataset change example

Parametri

Etichetă	Nume	Tipul	Descriere
Original layer	`ORIGINAL`	[vector: oricare]	The vector layer considered as the original version
Revised layer	`REVISED`	[vector: oricare]	The revised or modified vector layer
Attributes to consider for match Opţional	`COMPARE_ATTRIB`	[tablefield: any] [list]	Attributes to consider for match. By default, all attributes are compared.
Geometry comparison behavior Opţional	`MATCH_TYPE`	[enumerare] Implicit: 1	Defines the criteria for comparison. Options: • 0 — Exact Match: includes the order and vertices count of geometries • 1 — Tolerant Match (Topological Equality): geometries are considered equal
Unchanged features Opţional	`UNCHANGED`	[vector: same as Original layer]	Specify the output vector layer containing the unchanged features. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

continues on next page

Tabelul 28.126 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Added features Opţional	ADDED	[vector: same as Original layer]	Specify the output vector layer containing the added features. One of: • Skip Output • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Deleted features Opţional	DELETED	[vector: same as Original layer]	Specify the output vector layer containing the deleted features. One of: • Skip Output • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Unchanged features	UNCHANGED	[vector: same as Original layer]	Vector layer containing the unchanged features.
Added features	ADDED	[vector: same as Original layer]	Vector layer containing the added features.
Deleted features	DELETED	[vector: same as Original layer]	Vector layer containing the deleted features.
Count of unchanged features	UNCHANGED_COUN	[număr]	Count of unchanged features.
Count of features added in revised layer	ADDED_COUNT	[număr]	Count of features added in revised layer.
Count of features deleted from original layer	DELETED_COUNT	[număr]	Count of features deleted from original layer.

Cod Python

ID Algorithm: `nativ:detectvectorchanges`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Drop geometries

Creates a simple *geometryless* copy of the input layer attribute table. It keeps the attribute table of the source layer.

If the file is saved in a local folder, you can choose between many file formats.

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Delete duplicate geometries, *Remove null geometries*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer
Dropped geometries	`OUTPUT`	[tabelă]	Specify the output geometryless layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Dropped geometries	`OUTPUT`	[tabelă]	The output geometryless layer. A copy of the original attribute table.

Cod Python

ID Algorithm: `nativ:dropgeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Executare SQL

Runs a simple or complex query based only on SELECT with SQL syntax on the source layer.

Input datasources are identified with input1, input2... inputN and a simple query will look like SELECT * FROM input1.

Beside a simple query, you can add expressions or variables within the SQL query parameter itself. This is particulary useful if this algorithm is executed within a Processing model and you want to use a model input as a parameter of the query. An example of a query will then be SELECT * FROM [% @table %] where @table is the variable that identifies the model input.

The result of the query will be added as a new layer.

Vezi și:

SpatiaLite execute SQL, *PostgreSQL execute SQL*

Parametri

Etichetă	Nume	Tipul	Descriere
Additional input datasources (called input1, .., inputN in the query)	INPUT_DATASOUR	[vector: oricare] [list]	List of layers to query. In the SQL editor you can refer these layers with their **real** name or also with **input1**, **input2**, **inputN** depending on how many layers have been chosen.
Interogare SQL	INPUT_QUERY	[string]	Type the string of your SQL query, e.g. SELECT * FROM input1.
Unique identifier field Opţional	INPUT_UID_FIEL	[string]	Specify the column with unique ID
Geometry field Opţional	INPUT_GEOMETRY	[string]	Specify the geometry field
Tipul geometriei Opţional	INPUT_GEOMETRY	[enumerare] Implicit: 0	Choose the geometry of the result. By default the algorithm will autodetect it. One of: • 0 — Autodetectare • 1 — Fără geometrie • 2 — Punct • 3 — Șir de Linii • 4 — Poligon • 5 — MultiPunct • 6 — Multiple Șiruri de Linii • 7 — MultiPoligoane
CRS Opţional	INPUT_GEOMETRY	[crs]	The CRS to assign to the output layer
SQL Output	OUTPUT	[vector: oricare] Implicit: [Creare Strat temporar]	Specify the output layer created by the query. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
SQL Output	`OUTPUT`	[vector: oricare]	Stratul vectorial creat de interogare

Cod Python

Algorithm ID: `qgis:executesql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export layers to DXF

Exports layers to DXF file. For each layer, you can choose a field whose values are used to split features in generated destination layers in DXF output.

Vezi și:

Crearea fișierelor DXF noi

Parametri

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`LAYERS`	[vector: oricare] [list]	Input vector layers to export
Symbology mode	`SYMBOLOGY_MODE`	[enumerare] Implicit: 0	Type of symbology to apply to output layers. You can choose between: • 0 — Fără Simbologie • 1 — Simbologia Entității • 2 — Symbol Layer Symbology
Symbology scale	`SYMBOLOGY_SCAL`	[scale] Implicit: 1:1 000 000	Default scale of data export.
Codificare	`ENCODING`	[enumerare]	Encoding to apply to layers.
CRS	`CRS`	[crs]	Choose the CRS for the output layer.
Use layer title as name	`USE_LAYER_TITL`	[boolean] Implicit: *False*	Name the output layer with the layer title (as set in QGIS) instead of the layer name.
Force 2D	`FORCE_2D`	[boolean] Implicit: *False*	
Export labels as MTEXT elements	`MTEXT`	[boolean] Implicit: *False*	Exports labels as MTEXT or TEXT elements
DXF	`OUTPUT`	[file] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output DXF file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
DXF	`OUTPUT`	[file]	`.DXF` file containing the input layers

Cod Python

ID Algorithm: `nativ:dxfexport`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract selected features

Saves the selected features as a new layer.

Notă: If the selected layer has no selected features, the newly created layer will be empty.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Layer to save the selection from
Selected features	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the vector layer for the selected features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Selected features	`OUTPUT`	[la fel ca intrarea]	Vector layer with only the selected features, or no feature if none was selected.

Cod Python

ID Algorithm: `nativ:saveselectedfeatures`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract Shapefile encoding

Extracts the attribute encoding information embedded in a Shapefile. Both the encoding specified by an optional `.cpg` file and any encoding details present in the `.dbf` LDID header block are considered.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	ESRI Shapefile (`.SHP`) Layer to extract the encoding information.

Rezultat

Etichetă	Nume	Tipul	Descriere
Shapefile encoding	`ENCODING`	[string]	Encoding information specified in the input file
CPG encoding	`CPG_ENCODING`	[string]	Encoding information specified in any optional `.CPG` file
LDID encoding	`LDID_ENCODING`	[string]	Encoding information specified in `.dbf` LDID header block

Cod Python

ID Algorithm: `nativ:shpencodinginfo`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Find projection

Creates a shortlist of candidate coordinate reference systems, for instance for a layer with an unknown projection.

The area that the layer is expected to cover must be specified via the target area parameter. The coordinate reference system for this target area must be known to QGIS.

The algorithm operates by testing the layer's extent in every known reference system and then listing any for which the bounds would be near the target area if the layer was in this projection.

Vezi și:

Asignare proiecție, *Define Shapefile projection*, *Reproject layer*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Layer with unknown projection
Target area for layer (xmin, xmax, ymin, ymax)	`TARGET_AREA`	[extent]	The area that the layer covers. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS candidates	`OUTPUT`	[tabelă] Implicit: `[Creare Strat temporar]`	Specify the table (geometryless layer) for the CRS suggestions (EPSG codes). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
CRS candidates	`OUTPUT`	[tabelă]	A table with all the CRS (EPSG codes) of the matching criteria.

Cod Python

Algorithm ID: `qgis:findprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Flatten relationship

Flattens a *relationship* for a vector layer, exporting a single layer containing one parent feature per related child feature. This master feature contains all the attributes for the related features. This allows to have the relation as a plain table that can be e.g. exported to CSV.

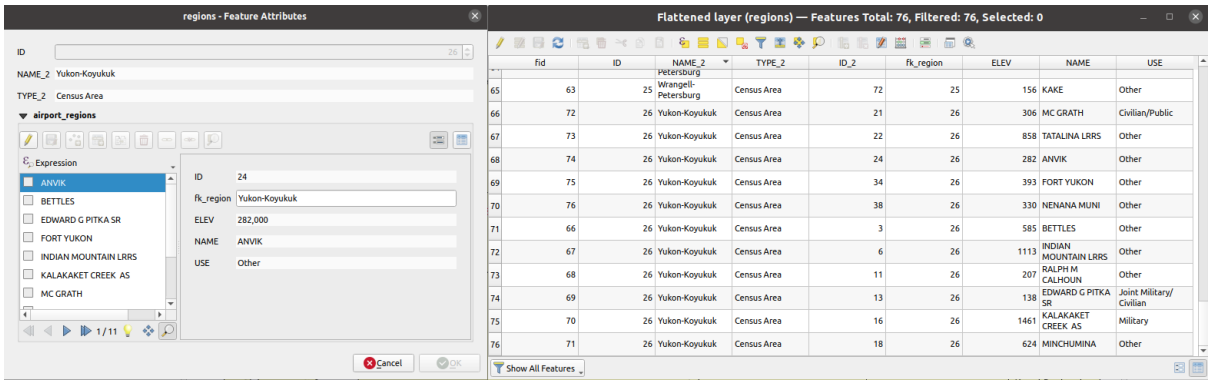


Fig. 28.55: Form of a region with related children (left) - A duplicate region feature for each related child, with joined attributes (right)

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Layer with the relationship that should be de-normalized
Flattened Layer Opţional	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (flattened) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Flattened layer	`OUTPUT`	[la fel ca intrarea]	A layer containing master features with all the attributes for the related features

Cod Python

ID Algorithm: `nativ:flattenrelationships`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Join attributes by field value

Takes an input vector layer and creates a new vector layer that is an extended version of the input one, with additional attributes in its attribute table.

The additional attributes and their values are taken from a second vector layer. An attribute is selected in each of them to define the join criteria.

Vezi şi:

Join attributes by nearest, *Join attributes by location*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Input vector layer. The output layer will consist of the features of this layer with attributes from matching features in the second layer.
Table field	`FIELD`	[tablefield: any]	Field of the source layer to use for the join
Stratul de intrare 2	`INPUT_2`	[vector: oricare]	Layer with the attribute table to join
Table field 2	`FIELD_2`	[tablefield: any]	Field of the second (join) layer to use for the join The type of the field must be equal to (or compatible with) the input table field type.
Layer 2 fields to copy Opţional	`FIELDS_TO_COPY`	[tablefield: any] [list]	Select the specific fields you want to add. By default all the fields are added.
Join type	`METHOD`	[enumerare] Implicit: 1	The type of the final joined layer. One of: • 0 — Create separate feature for each matching feature (one-to-many) • 1 — Take attributes of the first matching feature only (one-to-one)

continues on next page

Tabelul 28.132 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Discard records which could not be joined	DISCARD_NONMAT	[boolean] Implicit: True	Check if you don't want to keep the features that could not be joined
Joined field prefix Opţional	PREFIX	[string]	Add a prefix to joined fields in order to easily identify them and avoid field name collision
Joined layer Opţional	OUTPUT	[la fel ca intrarea] Implicit: [Creare Strat temporar]	Specify the output vector layer for the join. One of: • Skip Output • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fişier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Unjoinable features from first layer Opţional	NON_MATCHING	[la fel ca intrarea] Implicit: [Omitere rezultat]	Specify the output vector layer for unjoinable features from first layer. One of: • Skip Output • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fişier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Number of joined features from input table	JOINED_COUNT	[număr]	
Unjoinable features from first layer Opţional	NON_MATCHING	[la fel ca intrarea]	Vector layer with the non-matched features
Joined layer Opţional	OUTPUT	[la fel ca intrarea]	Output vector layer with added attributes from the join
Number of unjoinable features from input table Opţional	UNJOINABLE_COU	[număr]	

Cod Python

ID Algorithm: `nativ:joinattributestable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Join attributes by location

Takes an input vector layer and creates a new vector layer that is an extended version of the input one, with additional attributes in its attribute table.

The additional attributes and their values are taken from a second vector layer. A spatial criteria is applied to select the values from the second layer that are added to each feature from the first layer.

Default menu: *Vector ► Data Management Tools*

Vezi și:

Join attributes by nearest, *Join attributes by field value*, *Join attributes by location (summary)*

Exploring spatial relations

Geometric predicates are boolean functions used to determine the spatial relation a feature has with another by comparing whether and how their geometries share a portion of space.

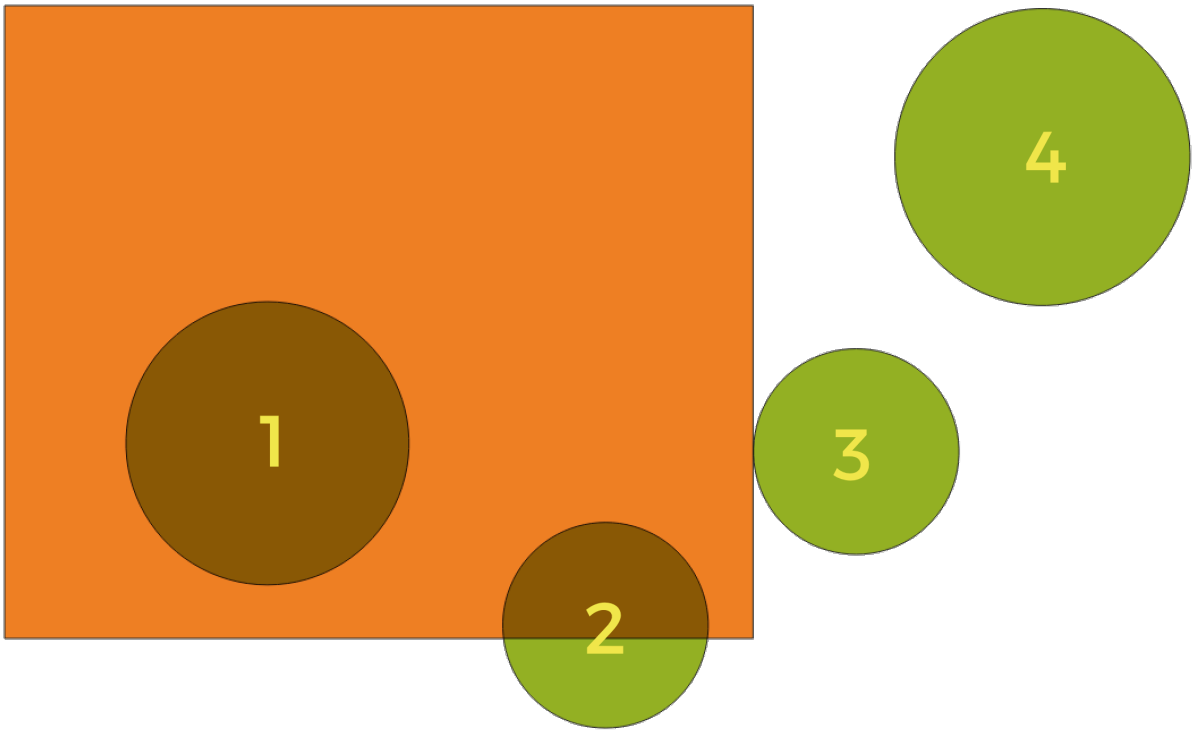


Fig. 28.56: Looking for spatial relations between layers

Folosind figura de mai sus, căutăm cercurile verzi comparându-le spațial cu entitatea dreptunghiulară portocalie. Predicatele geometrice disponibile sunt:

Intersect
Testează dacă o geometrie o intersectează pe alta. Returnează 1 (adevărat) dacă geometriile se intersectează spațial (împart orice porțiune de spațiu - se suprapun sau se ating) și 0 dacă nu se intersectează. În imaginea de mai sus, vor fi returnate cercurile 1, 2 și 3.

Contain
Returnează 1 (adevărat) dacă și numai dacă niciun punct al lui b nu se află în exteriorul lui a, iar cel puțin un punct din interiorul lui b se află în interiorul lui a. În imagine, nu este returnat niciun cerc, ar fi însă cazul dreptunghiului dacă l-ați căuta invers, întrucât conține cercul 1 complet. Acest predicat geometric este opusul lui *se află în interior*.

Disjoint
Returnează 1 (adevărat) dacă geometriile nu împart nicio porțiune de spațiu (fără suprapunere, fără atingere). Doar cercul 4 este returnat.

Equal
Returnează 1 (adevărat) dacă și numai dacă geometriile sunt exact aceleași. Niciun cerc nu va fi returnat.

Touch
Testează dacă o geometrie o atinge pe alta. Returnează 1 (adevărat) dacă geometriile au cel puțin un punct în comun, iar interiorul lor nu se intersectează. Doar cercul 3 este returnat.

Overlap
Testează dacă o geometrie se suprapune peste alta. Returnează 1 (adevărat) dacă geometriile împart spațiul, sunt de aceeași dimensiune, dar nu sunt complet conținute una în cealaltă. Doar cercul 2 este returnat.

Are within
Testează dacă o geometrie este conținută în alta. Returnează 1 (adevărat) dacă geometria a se află complet în interiorul geometriei b. Doar cercul 1 este returnat.

Cross
Returnează 1 (adevărat) dacă geometriile furnizate au unele, dar nu toate, punctele interioare, în comun, iar traversarea reală este de o dimensiune mai mică decât cea mai mare geometrie furnizată. De exemplu, o linie care traversează un poligon se va încrucișa ca o linie (adevărat). Două linii care se încrucișează se vor încrucișa ca punct (adevărat). Două poligoane se încrucișează ca poligon (fals). În imagine, niciun cerc nu va fi returnat.

Parametri

Etichetă	Nume	Tipul	Descriere
Join to features in	`INPUT`	[vector: oricare]	Input vector layer. The output layer will consist of the features of this layer with attributes from matching features in the second layer.
Where the features	`PREDICATE`	[enumeration] [list] Implicit: [0]	Type of spatial relation the source feature should have with the target feature so that they could be joined. One or more of: • 0 — intersectează • 1 — conţin • 2 — egal • 3 — atinge • 4 — se suprapune • 5 — se află în interior • 6 — traversează If more than one condition is chosen, at least one of them (OR operation) has to be met for a feature to be extracted.

continues on next page

Tabelul 28.133 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
By comparing to	`JOIN`	[vector: oricare]	The join layer. Features of this vector layer will **add** their attributes to the source layer attribute table if they satisfy the spatial relationship.
Fields to add (leave empty to use all fields) Opţional	`JOIN_FIELDS`	[tablefield: any] [list]	Select the specific fields you want to add from the join layer. By default all the fields are added.
Join type	`METHOD`	[enumerare]	The type of the final joined layer. One of: • 0 — Create separate feature for each matching feature (one-to-many) • 1 — Take attributes of the first matching feature only (one-to-one) • 2 — Take attributes of the feature with largest overlap only (one-to-one)
Discard records which could not be joined	`DISCARD_NONMAT`	[boolean] Implicit: *False*	Remove from the output the input layer's features which could not be joined
Joined field prefix Opţional	`PREFIX`	[string]	Add a prefix to joined fields in order to easily identify them and avoid field name collision
Joined layer Opţional	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the join. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Unjoinable features from first layer Opţional	`NON_MATCHING`	[la fel ca intrarea] Implicit: `[Omitere rezultat]`	Specify the output vector layer for unjoinable features from first layer. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Number of joined features from input table	`JOINED_COUNT`	[număr]	
Unjoinable features from first layer Opţional	`NON_MATCHING`	[la fel ca intrarea]	Vector layer of the non-matched features
Joined layer	`OUTPUT`	[la fel ca intrarea]	Output vector layer with added attributes from the join

Cod Python

ID Algorithm: `nativ:joinattributesbylocation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Join attributes by location (summary)

Takes an input vector layer and creates a new vector layer that is an extended version of the input one, with additional attributes in its attribute table.

The additional attributes and their values are taken from a second vector layer. A spatial criteria is applied to select the values from the second layer that are added to each feature from the first layer.

The algorithm calculates a statistical summary for the values from matching features in the second layer (e.g. maximum value, mean value, etc).

Vezi şi:

Join attributes by location

Exploring spatial relations

Geometric predicates are boolean functions used to determine the spatial relation a feature has with another by comparing whether and how their geometries share a portion of space.

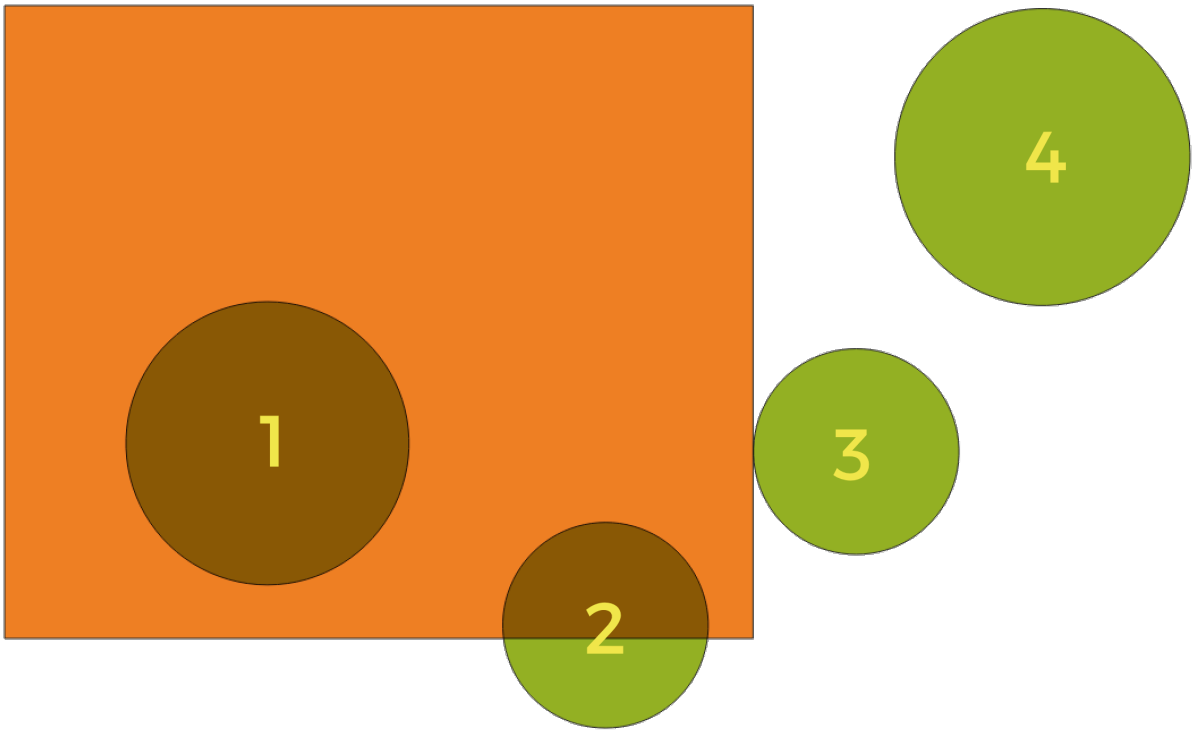


Fig. 28.57: Looking for spatial relations between layers

Folosind figura de mai sus, căutăm cercurile verzi comparându-le spațial cu entitatea dreptunghiulară portocalie. Predicatele geometrice disponibile sunt:

Intersect
Testează dacă o geometrie o intersectează pe alta. Returnează 1 (adevărat) dacă geometriile se intersectează spațial (împart orice porțiune de spațiu - se suprapun sau se ating) și 0 dacă nu se intersectează. În imaginea de mai sus, vor fi returnate cercurile 1, 2 și 3.

Contain
Returnează 1 (adevărat) dacă și numai dacă niciun punct al lui b nu se află în exteriorul lui a, iar cel puțin un punct din interiorul lui b se află în interiorul lui a. În imagine, nu este returnat niciun cerc, ar fi însă cazul dreptunghiului dacă l-ați căuta invers, întrucât conține cercul 1 complet. Acest predicat geometric este opusul lui *se află în interior*.

Disjoint
Returnează 1 (adevărat) dacă geometriile nu împart nicio porțiune de spațiu (fără suprapunere, fără atingere). Doar cercul 4 este returnat.

Equal
Returnează 1 (adevărat) dacă și numai dacă geometriile sunt exact aceleași. Niciun cerc nu va fi returnat.

Touch
Testează dacă o geometrie o atinge pe alta. Returnează 1 (adevărat) dacă geometriile au cel puțin un punct în comun, iar interiorul lor nu se intersectează. Doar cercul 3 este returnat.

Overlap
Testează dacă o geometrie se suprapune peste alta. Returnează 1 (adevărat) dacă geometriile împart spațiul, sunt de aceeași dimensiune, dar nu sunt complet conținute una în cealaltă. Doar cercul 2 este returnat.

Are within
Testează dacă o geometrie este conținută în alta. Returnează 1 (adevărat) dacă geometria a se află complet în interiorul geometriei b. Doar cercul 1 este returnat.

Cross
Returnează 1 (adevărat) dacă geometriile furnizate au unele, dar nu toate, punctele interioare, în comun, iar

traversarea reală este de o dimensiune mai mică decât cea mai mare geometrie furnizată. De exemplu, o linie care traversează un poligon se va încrucișa ca o linie (adevărat). Două linii care se încrucișează se vor încrucișa ca punct (adevărat). Două poligoane se încrucișează ca poligon (fals). În imagine, niciun cerc nu va fi returnat.

Parametri

Etichetă	Nume	Tipul	Descriere
Join to features in	`INPUT`	[vector: oricare]	Input vector layer. The output layer will consist of the features of this layer with attributes from matching features in the second layer.
Where the features	`PREDICATE`	[enumeration] [list] Implicit: [0]	Type of spatial relation the source feature should have with the target feature so that they could be joined. One or more of: • 0 — intersectează • 1 — conţin • 2 — egal • 3 — atinge • 4 — se suprapune • 5 — se află în interior • 6 — traversează If more than one condition is chosen, at least one of them (OR operation) has to be met for a feature to be extracted.
By comparing to	`JOIN`	[vector: oricare]	The join layer. Features of this vector layer will **add** summaries of their attributes to the source layer attribute table if they satisfy the spatial relationship.
Fields to summarize (leave empty to use all fields) Opţional	`JOIN_FIELDS`	[tablefield: any] [list]	Select the specific fields you want to add from the join layer. By default all the fields are added.

continues on next page

Tabelul 28.134 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Summaries to calculate (leave empty to use all fields) Opţional	`SUMMARIES`	[enumeration] [list] Implicit: []	For each input feature, statistics are calculated on joined fields of their matching features. One or more of: • 0 — count • 1 — unique • 2 — min • 3 — max • 4 — range • 5 — sum • 6 — mean • 7 — median • 8 — stddev • 9 — minority • 10 — majority • 11 — q1 • 12 — q3 • 13 — iqr • 14 — empty • 15 — filled • 16 — min_length • 17 — max_length • 18 — mean_length
Discard records which could not be joined	`DISCARD_NONMAT`	[boolean] Implicit: *False*	Remove from the output the input layer's features which could not be joined
Joined layer	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the join. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Joined layer	`OUTPUT`	[la fel ca intrarea]	Output vector layer with summarized attributes from the join

Cod Python

Algorithm ID: `qgis:joinbylocationsummary`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Join attributes by nearest

Takes an input vector layer and creates a new vector layer with additional fields in its attribute table. The additional attributes and their values are taken from a second vector layer. Features are joined by finding the closest features from each layer.

By default only the nearest feature is joined, but the join can also join to the k-nearest neighboring features.

If a maximum distance is specified, only features which are closer than this distance will be matched.

Vezi și:

Nearest neighbour analysis, *Join attributes by field value*, *Join attributes by location*, *Distance matrix*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul de intrare.
Stratul de intrare 2	`INPUT_2`	[vector: oricare]	The join layer.
Layer 2 fields to copy (leave empty to copy all fields)	`FIELDS_TO_COPY`	[fields]	Join layer fields to copy (if empty, all fields will be copied).
Discard records which could not be joined	`DISCARD_NONMAT`	[boolean] Implicit: *False*	Remove from the output the input layer records which could not be joined
Joined field prefix	`PREFIX`	[string]	Joined field prefix
Maximum nearest neighbors	`NEIGHBORS`	[număr] Implicit: 1	Maximum number of nearest neighbors
Maximum distance	`MAX_DISTANCE`	[număr]	Maximum search distance
Joined layer Opţional	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the vector layer containing the joined features. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

continues on next page

Tabelul 28.135 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Unjoinable features from first layer	NON_MATCHING	[la fel ca intrarea] Implicit: [Omitere rezultat]	Specify the vector layer containing the features that could not be joined. One of: • Skip Output • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Joined layer	OUTPUT	[la fel ca intrarea]	The output joined layer.
Unjoinable features from first layer	NON_MATCHING	[la fel ca intrarea]	Layer containing the features from first layer that could not be joined to any features in the join layer.
Number of joined features from input table	JOINED_COUNT	[număr]	Number of features from the input table that have been joined.
Number of unjoinable features from input table	UNJOINABLE_COU	[număr]	Number of features from the input table that could not be joined.

Cod Python

ID Algorithm: `nativ:joinbynearest`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

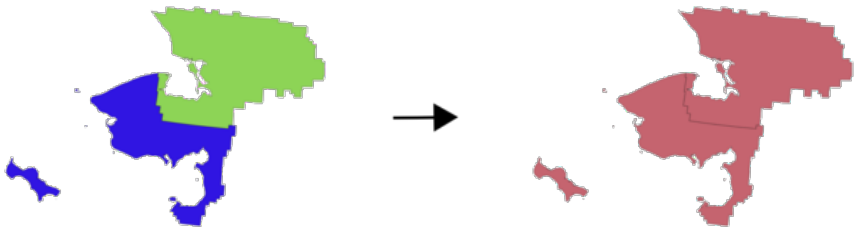
Merge vector layers

Combines multiple vector layers of the **same geometry** type into a single one.

The attribute table of the resulting layer will contain the fields from all input layers. If fields with the same name but different types are found then the exported field will be automatically converted into a string type field. New fields storing the original layer name and source are also added.

If any input layers contain Z or M values, then the output layer will also contain these values. Similarly, if any of the input layers are multi-part, the output layer will also be a multi-part layer.

Optionally, the destination coordinate reference system (CRS) for the merged layer can be set. If it is not set, the CRS will be taken from the first input layer. All layers will be reprojected to match this CRS.



Default menu: *Vector ► Data Management Tools*

Vezi și:

Split vector layer

Parametri

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`LAYERS`	[vector: oricare] [list]	The layers that are to be merged into a single layer. Layers should be of the same geometry type.
Destination CRS Opţional	`CRS`	[crs]	Choose the CRS for the output layer. If not specified, the CRS of the first input layer is used.
Merged	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Merged	`OUTPUT`	[la fel ca intrarea]	Output vector layer containing all the features and attributes from the input layers.

Cod Python

ID Algorithm: `nativ:mergevectorlayers`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Order by expression

Sorts a vector layer according to an expression: changes the feature index according to an expression.

Be careful, it might not work as expected with some providers, the order might not be kept every time.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Input vector layer to sort
Expression	`EXPRESSION`	[expresie]	Expression to use for the sorting
Sort ascending	`ASCENDING`	[boolean] Implicit: True	If checked the vector layer will be sorted from small to large values.
Sort nulls first	`NULLS_FIRST`	[boolean] Implicit: *False*	If checked, Null values are placed first
Ordered	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Ordered	`OUTPUT`	[la fel ca intrarea]	Output (sorted) vector layer

Cod Python

ID Algorithm: `nativ:orderbyexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Repair Shapefile

Repairs a broken ESRI Shapefile dataset by (re)creating the SHX file.

Parametri

Etichetă	Nume	Tipul	Descriere
Input Shapefile	`INPUT`	[file]	Full path to the ESRI Shapefile dataset with a missing or broken SHX file

Rezultat

Etichetă	Nume	Tipul	Descriere
Repaired layer	`OUTPUT`	[vector: oricare]	The input vector layer with the SHX file repaired

Cod Python

ID Algorithm: `nativ:repairshapefile`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Reproject layer

Reprojects a vector layer in a different CRS. The reprojected layer will have the same features and attributes of the input layer.

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Asignare proiecție, *Define Shapefile projection*, *Find projection*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Input vector layer to reproject
CRS-ul destinației	`TARGET_CRS`	[crs] Default: `EPSG:4326 - WGS 84`	Destination coordinate reference system
Convert curved geometries to straight segments `NEW in 3.32` Opţional	`CONVERT_CURVED`	[boolean] Implicit: *False*	If checked, curved geometries will be converted to straight segments in the process, avoiding potential distortion issues.
Reproiectat	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Coordinate Operation Opţional	`OPERATION`	[string]	Specific operation to use for a particular reprojection task, instead of always forcing use of the current project's transformation settings. Useful when reprojecting a particular layer and control over the exact transformation pipeline is required. Requires proj version >= 6. Read more at *Datum Transformations*.

Rezultat

Etichetă	Nume	Tipul	Descriere
Reproiectat	`OUTPUT`	[la fel ca intrarea]	Output (reprojected) vector layer

Cod Python

ID Algorithm: `nativ:reprojectlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Save vector features to file

Saves vector features to a specified file dataset.

For dataset formats supporting layers, an optional layer name parameter can be used to specify a custom string. Optional GDAL-defined dataset and layer options can be specified. For more information on this, read the online GDAL documentation on the format.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Entități vectoriale	`INPUT`	[vector: oricare]	Input vector layer.
Saved features	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specify the file to save the features to. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Layer name Opţional	LAYER_NAME	[string]	Name to use for the output layer
GDAL dataset options Opţional	DATASOURCE_OPT	[string]	GDAL dataset creation options of the output format. Separate individual options with semicolons.
GDAL layer options Opţional	LAYER_OPTIONS	[string]	GDAL layer creation options of the output format. Separate individual options with semicolons.
Action to take on pre-existing file	ACTION_ON_EXIS	[enumerare] Implicit: 0	How to manage existing features. Valid methods are: 0 — Create or overwrite file 1 — Create or overwrite layer 2 — Append features to existing layer, but do not create new fields 3 — Append features to existing layer, and create new fields if needed

Rezultat

Etichetă	Nume	Tipul	Descriere
Saved features	OUTPUT	[la fel ca intrarea]	Vector layer with the saved features.
File name and path	FILE_PATH	[string]	Output file name and path.
Layer name	LAYER_NAME	[string]	Name of the layer, if any.

Cod Python

ID Algorithm: `nativ:savefeatures`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Set layer encoding

Sets the encoding used for reading a layer's attributes. No permanent changes are made to the layer, rather it affects only how the layer is read during the current session.

Notă: Changing the encoding is only supported for some vector layer data sources.

Parametri

Etichetă	Nume	Tipul	Descriere
Saved features	`INPUT`	[vector: oricare]	Vector layer to set the encoding.
Codificare	`ENCODING`	[string]	Text encoding to assign to the layer in the current QGIS session.

Rezultat

Etichetă	Nume	Tipul	Descriere
Output layer	`OUTPUT`	[la fel ca intrarea]	Input vector layer with the set encoding.

Cod Python

ID Algorithm: `nativ:setlayerencoding`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Split features by character

Features are split into multiple output features by splitting a field's value at a specified character. For instance, if a layer contains features with multiple comma separated values contained in a single field, this algorithm can be used to split these values up across multiple output features. Geometries and other attributes remain unchanged in the output. Optionally, the separator string can be a regular expression for added flexibility.

Allows *features in-place modification* of point, line, and polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Split using values in the field	`FIELD`	[tablefield: any]	Field to use for splitting
Split value using character	`CHAR`	[string]	Character to use for splitting
Use regular expression separator	`REGEX`	[boolean] Implicit: *False*	

continues on next page

Tabelul 28.137 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Split	`OUTPUT`	[la fel ca intrarea] Implicit: `Creare Strat temporar`	Specify output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Split	`OUTPUT`	[la fel ca intrarea]	The output vector layer.

Cod Python

ID Algorithm: `nativ:splitfeaturesbycharacter`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Split vector layer

Creates a set of vectors in an output folder based on an input layer and an attribute. The output folder will contain as many layers as the unique values found in the desired field.

The number of files generated is equal to the number of different values found for the specified attribute.

It is the opposite operation of *merging*.

Default menu: *Vector ► Data Management Tools*

Vezi și:

Merge vector layers

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Unique ID field	`FIELD`	[tablefield: any]	Field to use for splitting
Director de ieşire	`OUTPUT`	[folder] Implicit: `[Salvare într-un fişier temporar]`	Specify the directory for the output layers. One of: • Salvare Intr-un Director Temporar • Salvare în Director

Parametri avansaţi

Etichetă	Nume	Tipul	Descriere
Output file type Opţional	`FILE_TYPE`	[enumerare] Implicit: `gpkg` în fereastra de dialog	Select the extension of the output files. If not specified or invalid, the output files format will be the one set in the „Default output vector layer extension" Processing setting.

Rezultat

Etichetă	Nume	Tipul	Descriere
Director de ieşire	`OUTPUT`	[folder]	The directory for the output layers
Output layers	`OUTPUT_LAYERS`	[same as input] [list]	The output vector layers resulting from the split.

Cod Python

ID Algorithm: `nativ:splitvectorlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Truncate table

Truncates a layer, by deleting all features from within the layer.

Atenționare: This algorithm modifies the layer in place, and deleted features cannot be restored!

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de Intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare

Rezultat

Etichetă	Nume	Tipul	Descriere
Truncated layer	`OUTPUT`	[folder]	The truncated (empty) layer

Cod Python

ID Algorithm: `nativ:truncatetable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.22 Vector geometry

Add geometry attributes

Computes geometric properties of the features in a vector layer and includes them in the output layer.

It generates a new vector layer with the same content as the input one, but with additional attributes, containing geometric measurements based on a selected CRS.

The attributes added to the table depend on the geometry type and dimension of the input layer:

- for **point** layers: X (`xcoord`), Y (`ycoord`), Z (`zcoord`) coordinates and/or M value (`mvalue`)
- for **line** layers: `length` and, for the LineString and CompoundCurve geometry types, the feature `sinuosity` and straight distance (`straightdis`)
- for **polygon** layers: `perimeter` and `area`

Default menu: *Vector* ► *Geometry Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Calculate using	`CALC_METHOD`	[enumerare] Implicit: 0	Calculation parameters to use for the geometric properties. One of: • 0 — CRS-ul stratului • 1 — CRS-ul proiectului • 2 — Elipsoidal
Added geom info	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (input copy with geometry) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Added geom info	`OUTPUT`	[la fel ca intrarea]	Copy of the input vector layer with the addition of the geometry fields

Cod Python

Algorithm ID: `qgis:exportaddgeometrycolumns`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Affine transform

Applies an affine transformation to the layer geometries. Affine transformations can include translation, scaling and rotation. The operations are performed in the following order: scale, rotation, and translation.

Z and M values (if present) can be translated and scaled.

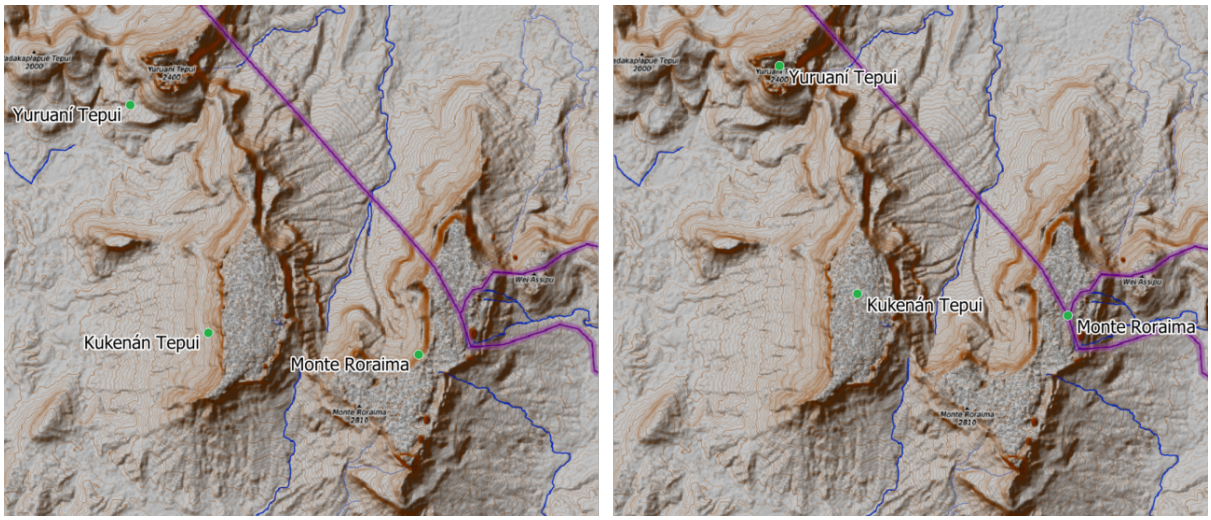


Fig. 28.58: Vector point layer (green dots) before (left), and after (right) an affine transformation (translation).

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Translate

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Translation (x-axis)	`DELTA_X`	[number] Implicit: 0	Displacement to apply on the X axis.
Translation (y-axis)	`DELTA_Y`	[number] Implicit: 0	Displacement to apply on the Y axis.
Translation (z-axis)	`DELTA_Z`	[number] Implicit: 0	Displacement to apply on the Z axis.
Translation (m-values)	`DELTA_M`	[number] Implicit: 0	Offset to apply on m values.
Scale factor (x-axis)	`SCALE_X`	[number] Implicit: 1	Scaling value (expansion or contraction) to apply on the X axis.
Scale factor (y-axis)	`SCALE_Y`	[number] Implicit: 1	Scaling value (expansion or contraction) to apply on the Y axis.
Scale factor (z-axis)	`SCALE_Z`	[number] Implicit: 1	Scaling value (expansion or contraction) to apply on the Z axis.

continues on next page

Tabelul 28.140 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Scale factor (m-values)	`SCALE_M`	[number] Implicit: 1	Scaling value (expansion or contraction) to apply on m values.
Rotation around z-axis (degrees counter-clockwise)	`ROTATION_Z`	[number] Implicit: 0	Angle of the rotation in degrees.
Transformed	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Transformed	`OUTPUT`	[la fel ca intrarea]	Output (transformed) vector layer.

Cod Python

ID Algorithm: `nativ:affinetransform`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Aggregate

Takes a vector or table layer and creates a new layer by aggregating features based on a `group by` expression.

Features for which `group by` expression returns the same value are grouped together.

It is possible to group all source features together using constant value in `group by` parameter, example: NULL.

It is also possible to group features by multiple fields using Array function, example: Array(„Field1”, „Field2”).

Geometries (if present) are combined into one multipart geometry for each group. Output attributes are computed depending on each given aggregate definition.

This algorithm allows to use the default *aggregates functions* of the QGIS Expression engine.

Vezi și:

Collect geometries, *Dizolvare*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Group by expression	`GROUP_BY`	[tablefield: any] Implicit: «NULL»	Choose the grouping field. If *NULL* all features will be grouped.

continues on next page

Tabelul 28.142 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Aggregates	AGGREGATES	[list]	List of output layer field definitions. Example of a field definition: *{«aggregate»: «sum», «delimiter»: «,», «input»: « $area», «length»: 10, «name»: «totarea», «precision»: 0, «type»: 6}* By default, the list contains all the fields of the input layer. In the GUI, you can edit these fields and their definitions, and you can also: • Click the button to add a new field. • Click to delete the selected field. • Use and to change order of the fields. • Click to reset to the default (the fields of the input layer). For each of the fields you'd like to retrieve information from, you need to define the following: **Input expression [expression] (input)** Field or expression from the input layer. **Aggregate function [enumeration] (aggregate)** *Function* to use on the input expression to return the aggregated value. Default: *concatenate* (for string data type), *sum* (for numeric data type) **Delimiter [string] (delimiter)** Text string to separate aggregated values, for example in case of concatenation. Implicit: , **Output field name [string] (name)** Name of the aggregated field in the output layer. By default input field name is kept. **Type [enumeration] (type)** Data type of the output field. One of: • 1 — Boolean • 2 — Întreg • 4 — Integer64 • 6 — Double • 10 — Șir • 14 — Dată • 16 — DateTime **Length [number] (length)** Length of the output field. **Precision [number] (precision)** Precision of the output field.

continues on next page

Tabelul 28.142 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Load fields from layer	GUI only	[vector: oricare]	You can load fields from another layer and use them for the aggregation
Aggregated	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (aggregate) layer One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Aggregated	`OUTPUT`	[la fel ca intrarea]	Multigeometry vector layer with the aggregated values

Cod Python

ID Algorithm: `nativ:aggregate`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Boundary

Returns the closure of the combinatorial boundary of the input geometries (i.e. the topological boundary of the geometry).

Only for polygon and line layers.

For **polygon geometries** , the boundary consists of all the lines making up the rings of the polygon.

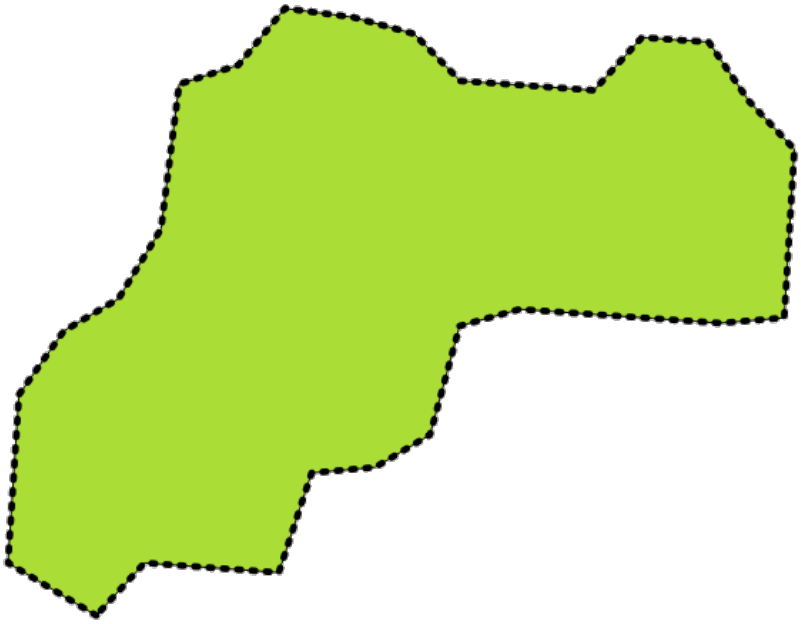


Fig. 28.59: Boundaries (black dashed line) of the source polygon layer

For **lines geometries**, the boundaries are their end points.

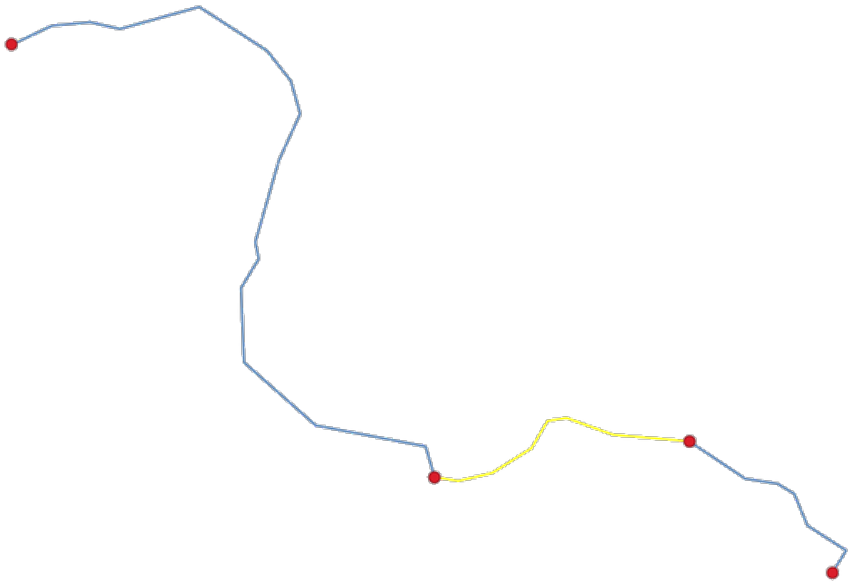


Fig. 28.60: Boundary layer (red points) for lines. In yellow a selected feature.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Boundary	`OUTPUT`	[vector: point, line] Implicit: `[Creare Strat temporar]`	Specify the output (boundary) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Boundary	`OUTPUT`	[vector: point, line]	Boundaries from the input layer (point for line, and line for polygon)

Cod Python

ID Algorithm: `nativ:boundary`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Bounding boxes

Calculates the bounding box (envelope) of each feature in an input layer. Polygon and line geometries are supported.

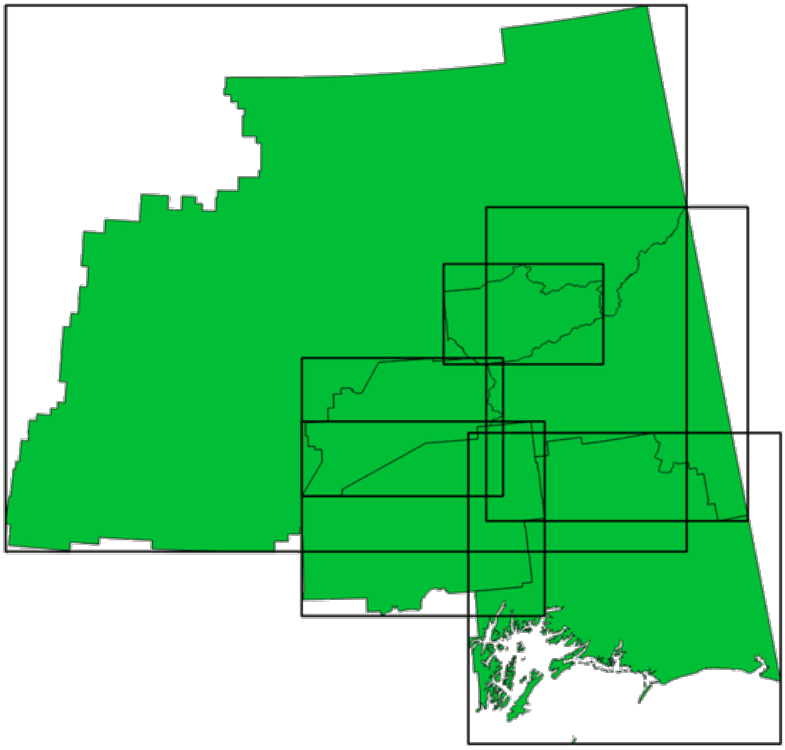


Fig. 28.61: Black lines represent the bounding boxes of each polygon feature

Allows *features in-place modification* of polygon features

Vezi şi:

Minimum bounding geometry

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Bounds	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output (bounding box) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Bounds	`OUTPUT`	[vector: poligon]	Bounding boxes of input layer. Other than the input attributes, the output layer also contains following fields: `width`, `height`, `area` and `perimeter` of the generated polygon.

Cod Python

ID Algorithm: `nativ:boundingboxes`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Buffer

Computes a buffer area for all the features in an input layer, using a fixed or data defined distance.

It is possible to use a negative distance for polygon input layers. In this case the buffer will result in a smaller polygon (setback).

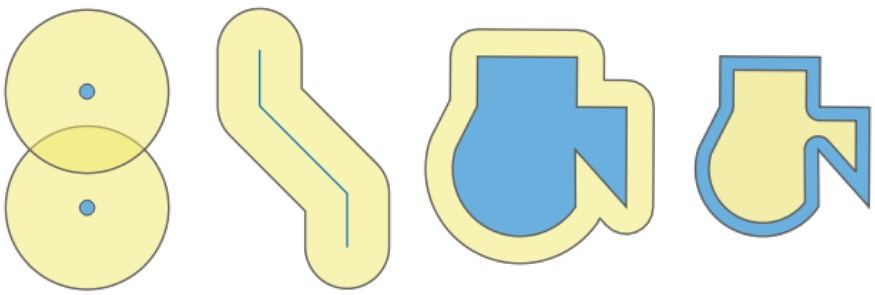


Fig. 28.62: Buffer (in yellow) of points, line, polygon with positive buffer, and polygon with negative buffer

Allows *features in-place modification* of polygon features

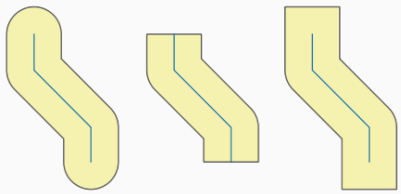
Default menu: *Vector ► Geoprocessing Tools*

Vezi și:

Variable distance buffer, *Multi-ring buffer (constant distance)*, *Variable width buffer (by M value)*

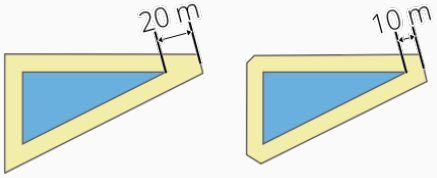
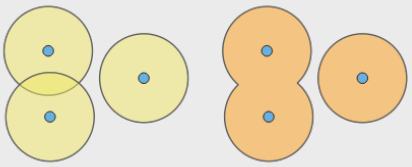
Parametri

Parametri de bază

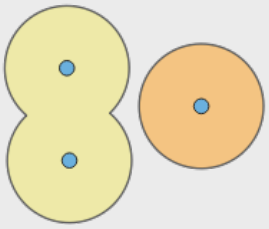
Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Distance	`DISTANCE`	[number] Implicit: 10.0	Buffer distance (from the boundary of each feature). You can use the Data Defined button on the right to choose a field from which the radius will be calculated. This way you can have different radius for each feature (see *Variable distance buffer*).
Segments	`SEGMENTS`	[număr] Implicit: 5	Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.
End cap style	`END_CAP_STYLE`	[enumerare] Implicit: 0	Controls how line endings are handled in the buffer. One of: • 0 — Round • 1 — Plat • 2 — Pătrat  Fig. 28.63: Round, flat and square cap styles
Join style	`JOIN_STYLE`	[enumerare] Implicit: 0	Specifies whether round, miter or beveled joins should be used when offsetting corners in a line. Options are: • 0 — Round • 1 — Miter • 2 — Bevel Fig. 28.64: Round, miter, and bevel join styles

continues on next page

Tabelul 28.143 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Miter limit	`MITER_LIMIT`	[număr] Implicit: 2.0	Sets the maximum distance from the offset geometry to use when creating a mitered join as a factor of the offset distance (only applicable for miter join styles). Minimum: 1.0  Fig. 28.65: A 10m buffer with a limit of 2 and a 10m buffer with a limit of 1
Dissolve result	`DISSOLVE`	[boolean] Implicit: *False*	Dissolve the final buffer. If `True` (checked), overlapping buffers will be dissolved (combined) into a single multipart feature.  Fig. 28.66: Standard (three single part features - left), dissolved (1 multipart feature with 2 parts - right)
Buffered	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output (buffer) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Keep disjoint features separate	`SEPARATE_DISJO`	[boolean] Implicit: *False*	If `True` (checked) and dissolved is checked, features that do not overlap or touch will be exported as separate features (instead of parts of a multipart feature).  Fig. 28.67: Results in 2 single part features

Rezultat

Etichetă	Nume	Tipul	Descriere
Buffered	`OUTPUT`	[vector: poligon]	Output (buffer) polygon layer

Cod Python

ID Algorithm: `nativ:buffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Centroids

Creates a new point layer, with points representing the centroids of the geometries of the input layer.

The centroid is a single point representing the barycenter (of all parts) of the feature, so it can be outside the feature borders. But can also be a point on each part of the feature.

The attributes of the points in the output layer are the same as for the original features.

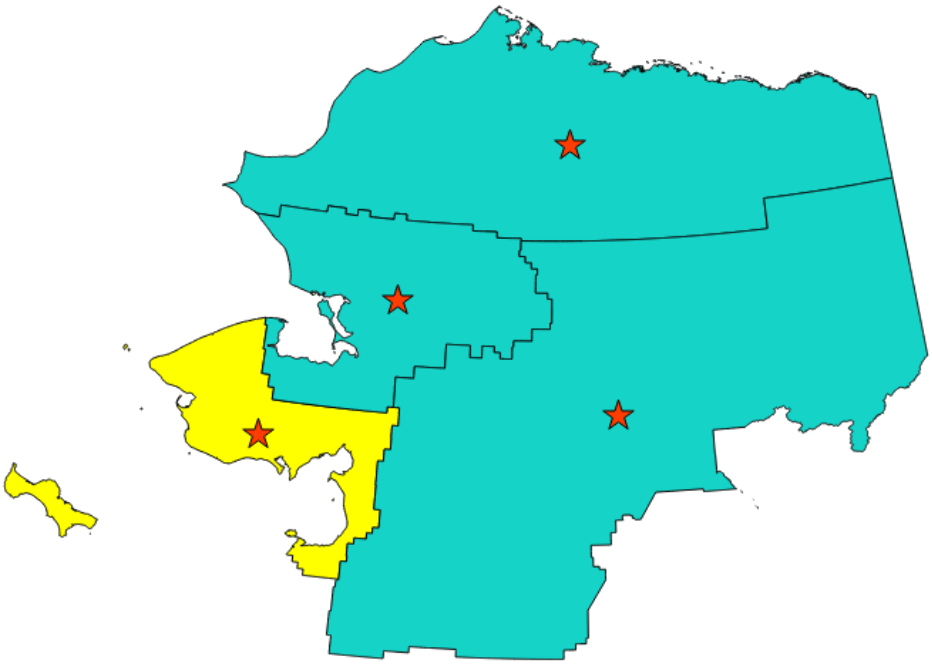


Fig. 28.68: The red stars represent the centroids of the features of the input layer.

Allows *features in-place modification* of point features

Default menu: *Vector ► Geometry Tools*

Vezi și:

Point on Surface

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Create centroid for each part	`ALL_PARTS`	[boolean] Implicit: *False*	If True (checked), a centroid will be created for each part of the geometry
Centroids	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output (centroid) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Centroids	`OUTPUT`	[vector: punct]	Output point vector layer (centroids)

Cod Python

ID Algorithm: `nativ:centroids`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Check validity

Performs a validity check on the geometries of a vector layer.

The geometries are classified in three groups (valid, invalid and error) and for each group, a vector layer with its features is generated:

- The **Valid output** layer contains only the valid features (without topological errors).
- The **Invalid output** layer contains all the invalid features found by the algorithm.
- The **Error output** layer is a point layer that points to where the invalid features were found.

The attribute tables of the generated layers will contain some additional information („message” for the **error** layer, „FID” and „_errors” for the **invalid** layer and only „FID” for the **valid** layer):

The attribute table of each generated vector layer will contain some additional information (number of errors found and types of error):

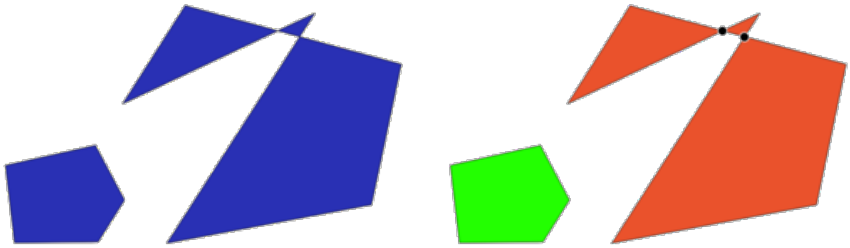


Fig. 28.69: Left: the input layer. Right: the valid layer (green), the invalid layer (orange)

Default menu: *Vector ► Geometry Tools*

Vezi și:

Fix geometries and the core plugin *Geometry Checker Plugin*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT_LAYER`	[vector: oricare]	Stratul vectorial de intrare
Method	`METHOD`	[enumerare] Implicit: 2	Method to use to check validity. Options: • 0: The one selected in digitizing settings • 1: QGIS • 2: GEOS
Ignore ring self intersection	`IGNORE_RING_SE`	[boolean] Implicit: *False*	Ignore self intersecting rings when checking for validity.
Valid output	`VALID_OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the vector layer to contain a copy of the valid features of the source layer. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Invalid output	`INVALID_OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Vector layer containing copy of the invalid features of the source layer with the field `_errors` listing the summary of the error(s) found. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

continues on next page

Tabelul 28.144 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Error output	`ERROR_OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Point layer of the exact position of the validity problems detected with the `message` field describing the error(s) found. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Count of errors	`ERROR_COUNT`	[număr]	The number of geometries that caused errors.
Error output	`ERROR_OUTPUT`	[vector: punct]	Point layer of the exact position of the validity problems detected with the `message` field describing the error(s) found.
Count of invalid features	`INVALID_COUNT`	[număr]	The number of invalid geometries.
Invalid output	`INVALID_OUTPUT`	[la fel ca intrarea]	Vector layer containing copy of the invalid features of the source layer with the field `_errors` listing the summary of the error(s) found.
Count of valid features	`VALID_COUNT`	[număr]	The number of valid geometries.
Valid output	`VALID_OUTPUT`	[la fel ca intrarea]	Vector layer containing a copy of the valid features of the source layer.

Cod Python

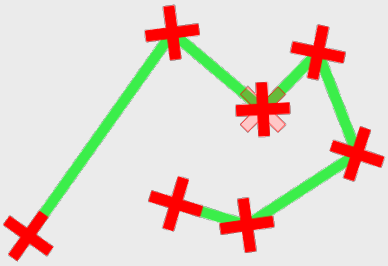
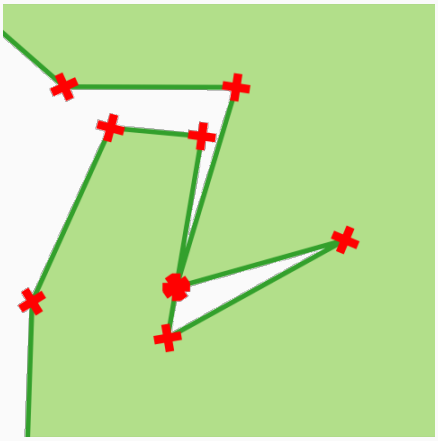
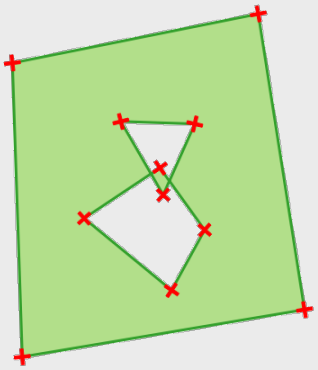
Algorithm ID: `qgis:checkvalidity`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

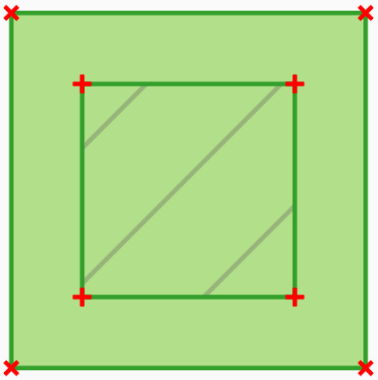
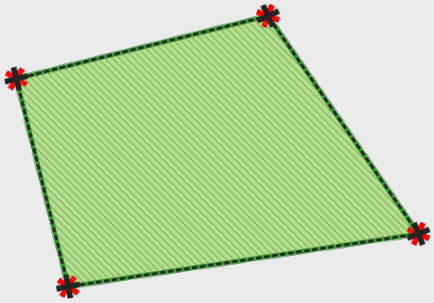
Types of error messages and their meanings

Tabelul 28.146: If the GEOS method is used the following error messages can occur:

Error message	Explanation	Exemplu
Repeated point	This error happens when a given vertex is repeated.	
Ring self-intersection	This error happens when a geometry touches itself and generates a ring.	
Self-intersection	This error happens when a geometry touches itself.	
Topology validation error		
Hole lies outside shell		
Holes are nested		
Interior is disconnected		

continues on next page

Tabelul 28.146 – continuare din pagina precedentă

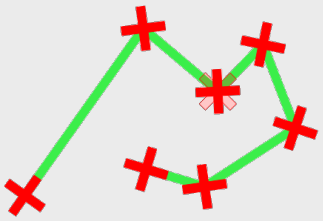
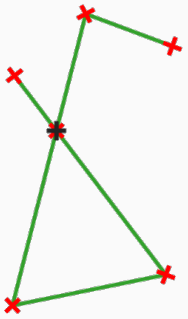
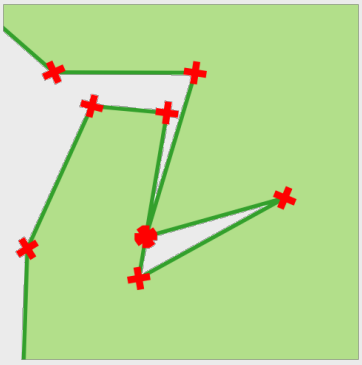
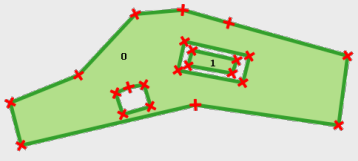
Error message	Explanation	Exemplu
Nested shells	This error happens when a polygon geometry is on top of another polygon geometry.	
Duplicate rings	This error happens when two rings (exterior or interior) of a polygon geometry are identical	
Too few points in geometry component		
Invalid coordinate	For a point geometry, this error happens when the geometry does not have a proper coordinate pair. The coordinate pair does not contain a latitude value and a longitude value in that order.	
Ring is not closed		

Tabelul 28.147: If the QGIS method is used the following error messages can occur:

Error message	Explanation	Exemplu
Segment %1 of ring %2 of polygon %3 intersects segment %4 of ring %5 of polygon %6 at %7		
Ring %1 with less than four points		
Ring %1 not closed		
Line %1 with less than two points		

continues on next page

Tabelul 28.147 – continuare din pagina precedentă

Error message	Explanation	Exemplu
Line %1 contains %n duplicate node(s) at %2	This error happens when consecutive points on a line have the same coordinates.	
Segments %1 and %2 of line %3 intersect at %4	This error happens when a line self intersects (two segments of the line intersect each other).	
Ring self-intersection	This error happens when an outer or inner (island) ring / boundary of a polygon geometry intersects itself.	
Ring %1 of polygon %2 not in exterior ring		
Polygon %1 lies inside polygon %2	This error happens when a part of a MultiPolygon geometry is inside a hole of a MultiPolygon geometry.	

Collect geometries

Takes a vector layer and collects its geometries into new multipart geometries.

One or more attributes can be specified to collect only geometries belonging to the same class (having the same value for the specified attributes), alternatively all geometries can be collected.

All output geometries will be converted to multi geometries, even those with just a single part. This algorithm does not dissolve overlapping geometries - they will be collected together without modifying the shape of each geometry part.

See the «Promote to multipart» or «Aggregate» algorithms for alternative options.

Default menu: *Vector ► Geometry Tools*

Vezi și:

Aggregate, *Promote to multipart*, *Dizolvare*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Unique ID fields	`FIELD`	[tablefield: any] [list]	Choose one or more attributes to collect the geometries
Collected	`OUTPUT`	[la fel ca intrarea]	Vector layer with collected geometries

Rezultat

Etichetă	Nume	Tipul	Descriere
Collected	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the collected geometries. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Cod Python

ID Algorithm: `nativ:collect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Concave hull

Computes the concave hull of the features from an input point layer.

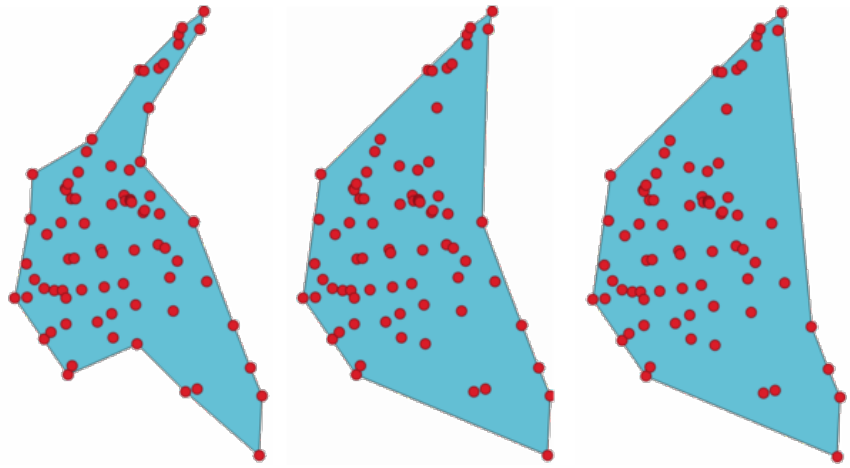


Fig. 28.70: Concave hulls with different thresholds (0.3, 0.6, 0.9)

Vezi și:

Convex hull

Parametri

Etichetă	Nume	Tipul	Descriere
Input point layer	`INPUT`	[vector: punct]	Input point vector layer
Threshold	`ALPHA`	[număr] Implicit: 0.3	Number from 0 (maximum concave hull) to 1 (convex hull).
Allow holes	`HOLES`	[boolean] Implicit: True	Choose whether to allow holes in the final concave hull
Split multipart geometry into singlepart geometries	`NO_MULTIGEOMET`	[boolean] Implicit: *False*	Check if you want to have singlepart geometries instead of multipart ones.
Concave hull	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Concave hull	`OUTPUT`	[vector: poligon]	The output vector layer

Cod Python

Algorithm ID: `native:concavehull`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Convert geometry type

Generates a new layer based on an existing one, with a different type of geometry.

The attribute table of the output layer is the same as the one of the input layer.

Not all conversions are possible. For instance, a line layer can be converted to a point layer, but a point layer cannot be converted to a line layer.

Vezi și:

Polygonize, *Lines to polygons*, *Polygons to lines*, *Points to path*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
New geometry type	`TYPE`	[enumerare] Implicit: 0	Geometry type to apply to the output features. One of: • 0 — Centroizi • 1 — Noduri • 2 — Șiruri de Linii • 3 — Multiple Șiruri de Linii • 4 — Poligoane
Converted	`OUTPUT`	[vector: oricare] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Converted	`OUTPUT`	[vector: oricare]	Output vector layer - the type depends on the parameters

Cod Python

Algorithm ID: `qgis:convertgeometrytype`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Convert to curved geometries

Converts a geometry into its curved geometry equivalent.

Already curved geometries will be retained without change.

Allows *features in-place modification* of line and polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Stratul vectorial de intrare
Maximum distance tolerance	`DISTANCE`	[număr] Implicit: 0.000001	The maximum distance allowed between the original location of vertices and where they would fall on the converted curved geometries
Maximum angle tolerance	`ANGLE`	[număr] Implicit: 0.000001	Segments are considered as suitable for replacing with an arc if the points are all regularly spaced on the candidate arc. This parameter specifies the maximum angular deviation (in degrees) allowed when testing for regular point spacing. Between 0 and 45°.
Curves	`OUTPUT`	[vector: curbă compusă sau poligon curb] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Curves	`OUTPUT`	[vector: curbă compusă sau poligon curb]	Output vector layer with curved geometries

Cod Python

ID Algorithm: `nativ:converttocurves`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Convex hull

Calculates the convex hull for each feature in an input layer.

See the «Minimum bounding geometry» algorithm for a convex hull calculation which covers the whole layer or grouped subsets of features.

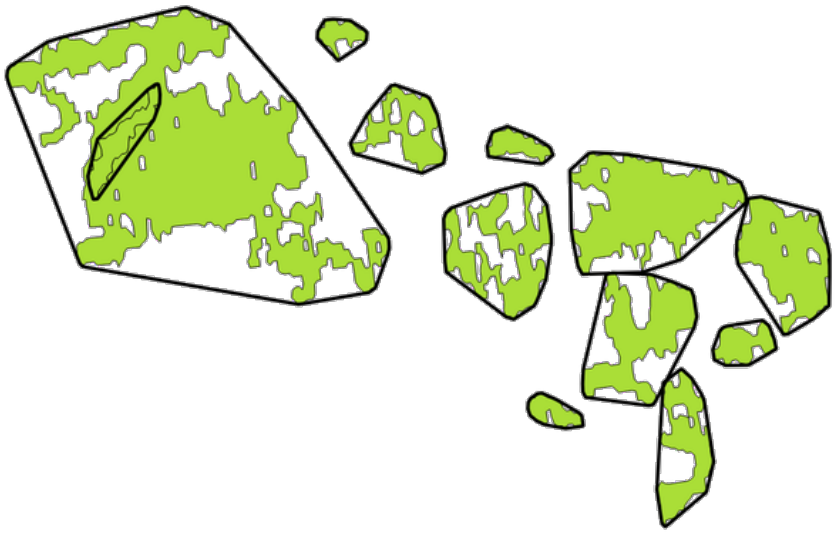


Fig. 28.71: Black lines identify the convex hull for each layer feature

Allows *features in-place modification* of polygon features

Default menu: *Vector* ► *Geoprocessing Tools*

Vezi și:

Minimum bounding geometry, *Concave hull*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Convex hull	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Convex hull	`OUTPUT`	[vector: poligon]	The output (convex hull) vector layer

Cod Python

ID Algorithm: `nativ:convexhull`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create layer from extent

Creates a new vector layer that contains a single feature with geometry matching the extent of the input layer.

It can be used in models to convert a literal extent (`xmin`, `xmax`, `ymin`, `ymax` format) into a layer which can be used for other algorithms which require a layer based input.

Vezi și:

Create layer from point

Parametri

Etichetă	Nume	Tipul	Descriere
Extent (xmin, xmax, ymin, ymax)	`INPUT`	[extent]	Input extent Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Extindere	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extindere	`OUTPUT`	[vector: poligon]	The output (extent) vector layer

Cod Python

ID Algorithm: `nativ:extenttolayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create layer from point

Creates a new vector layer that contains a single feature with geometry matching a point parameter. It can be used in models to convert a point into a point layer for algorithms which require a layer based input.

Vezi și:

Create layer from extent

Parametri

Etichetă	Nume	Tipul	Descriere
Point	`INPUT`	[coordinates]	Input point, including CRS info (example: `397254,6214446 [EPSG:32632]`). If the CRS is not provided, the Project CRS will be used. The point can be specified by clicking on the map canvas.
Point	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Point	`OUTPUT`	[vector: punct]	The output point vector layer containing the input point.

Cod Python

ID Algorithm: `nativ:pointtolayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Create wedge buffers

Creates wedge shaped buffers from input points.

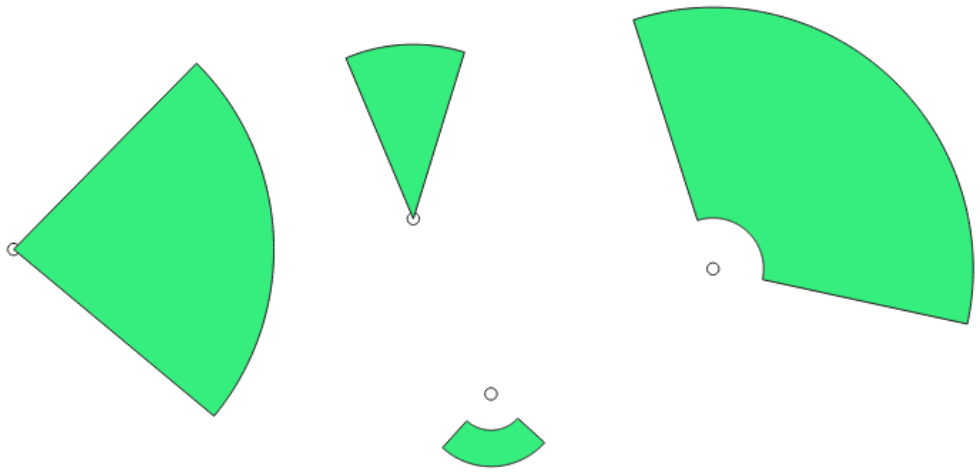


Fig. 28.72: Wedge buffers

The native output from this algorithm are CurvePolygon geometries, but these may be automatically segmentized to Polygons depending on the output format.

Vezi şi:

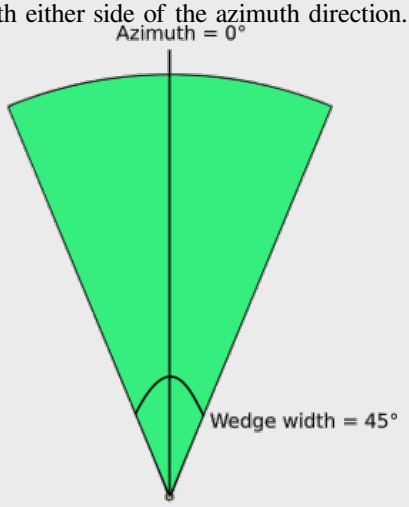
Buffer, *Variable width buffer (by M value)*, *Tapered buffers*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Input point vector layer
Azimuth (degrees from North)	`AZIMUTH`	[number] Implicit: 0.0	Angle (in degrees) as the middle value of the wedge

continues on next page

Tabelul 28.150 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Wedge width (in degrees)	`WIDTH`	[number] Implicit: 45.0	Width (in degrees) of the buffer. The wedge will extend to half of the angular width either side of the azimuth direction.  Fig. 28.73: Azimuth and width values of the wedge buffer
Outer radius	`OUTER_RADIUS`	[number] Implicit: 1.0	The outer *size* (length) of the wedge: the size is meant from the source point to the edge of the wedge shape.
Inner radius Opţional	`INNER_RADIUS`	[number] Implicit: 0.0	Inner radius value. If 0 the wedge will begin from the source point.
Buffers	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Buffers	`OUTPUT`	[vector: poligon]	The output (wedge buffer) vector layer

Cod Python

ID Algorithm: `nativ:wedgebuffers`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Delaunay triangulation

Creates a polygon layer with the Delaunay triangulation corresponding to the input point layer.

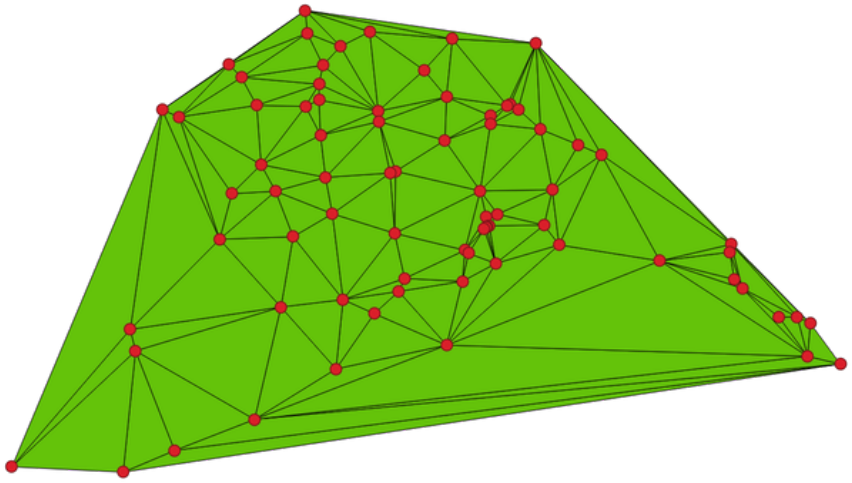


Fig. 28.74: Delaunay triangulation on points

Default menu: *Vector* ► *Geometry Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Input point vector layer
Tolerance Opţional	`TOLERANCE`	[număr] Implicit: 0.0	Specifies an optional snapping tolerance which can be used to improve the robustness of the triangulation.
Add point IDs to output	`ADD_ATTRIBUTES`	[boolean] Implicit: True	Specifies whether fields storing involved point features ID should be added to the output. If False, an `id` field is used to identify the polygons.
Delaunay triangulation	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Delaunay triangulation	`OUTPUT`	[vector: poligon]	The output (Delaunay triangulation) vector layer

Cod Python

Algorithm ID: `native:delaunaytriangulation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Delete holes

Takes a polygon layer and removes holes in polygons. It creates a new vector layer in which polygons with holes have been replaced by polygons with only their external ring. Attributes are not modified.

An optional minimum area parameter allows removing only holes which are smaller than a specified area threshold. Leaving this parameter at `0.0` results in all holes being removed.

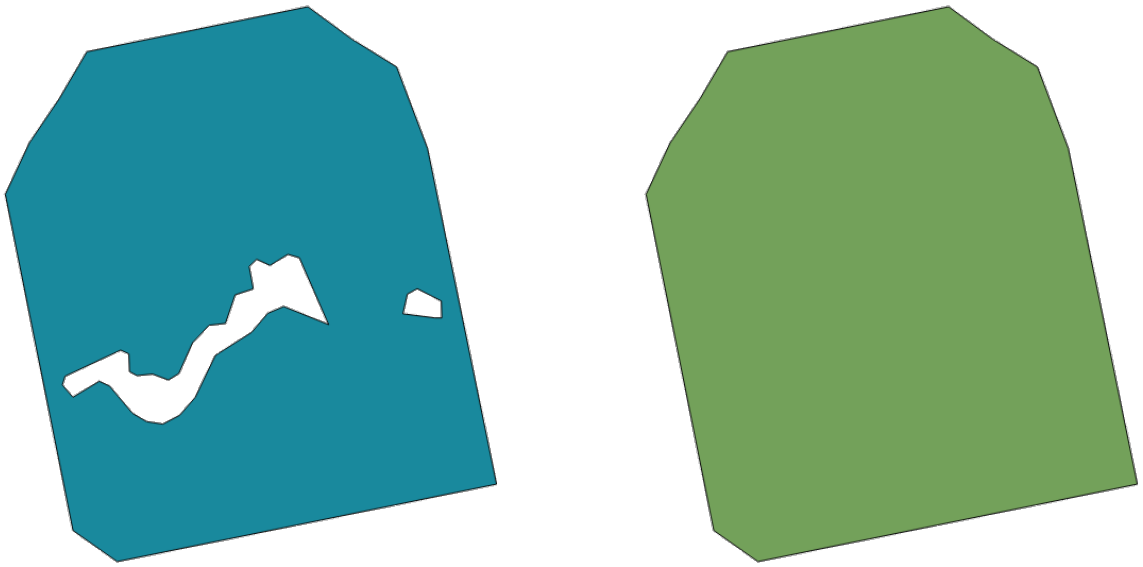


Fig. 28.75: Before and after the cleaning

Allows *features in-place modification* of polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Input polygon vector layer
Remove holes with area less than Opţional	`MIN_AREA`	[number] Implicit: 0.0	Only holes with an area less than this threshold will be deleted. With a value of `0.0`, **all** the holes will be deleted.
Cleaned	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Cleaned	`OUTPUT`	[la fel ca intrarea]	The output (cleaned) vector layer

Cod Python

ID Algorithm: `nativ:deleteholes`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Densify by count

Takes a polygon or line layer and generates a new one in which the geometries have a larger number of vertices than the original one.

If the geometries have Z or M values present then these will be linearly interpolated at the added vertices.

The number of new vertices to add to each segment is specified as an input parameter.

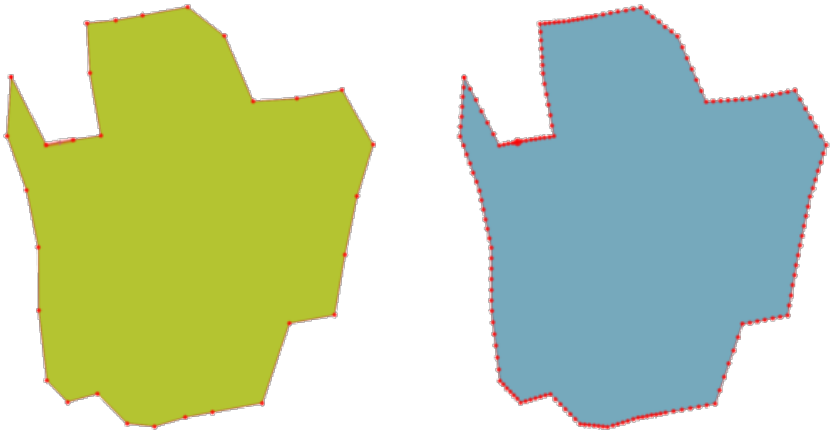


Fig. 28.76: Red points show the vertices before and after the densify

Allows *features in-place modification* of line and polygon features

Default menu: *Vector ► Geometry Tools*

Vezi și:

Densify by interval

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Vertices to add	`VERTICES`	[număr] Implicit: 1	Number of vertices to add to each segment
Densified	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Densified	`OUTPUT`	[la fel ca intrarea]	The output (densified) vector layer

Cod Python

ID Algorithm: `nativ:densifygeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Densify by interval

Takes a polygon or line layer and generates a new one in which the geometries have a larger number of vertices than the original one.

The geometries are densified by adding regularly placed extra vertices inside each segment so that the maximum distance between any two vertices does not exceed the specified distance.

If the geometries have Z or M values present then these will be linearly interpolated at the added vertices.

Example

Specifying a distance of 3 would cause the segment `[0 0] -> [10 0]` to be converted to `[0 0] -> [2.5 0] -> [5 0] -> [7.5 0] -> [10 0]`, since 3 extra vertices are required on the segment and spacing these at 2.5 increments allows them to be evenly spaced over the segment.

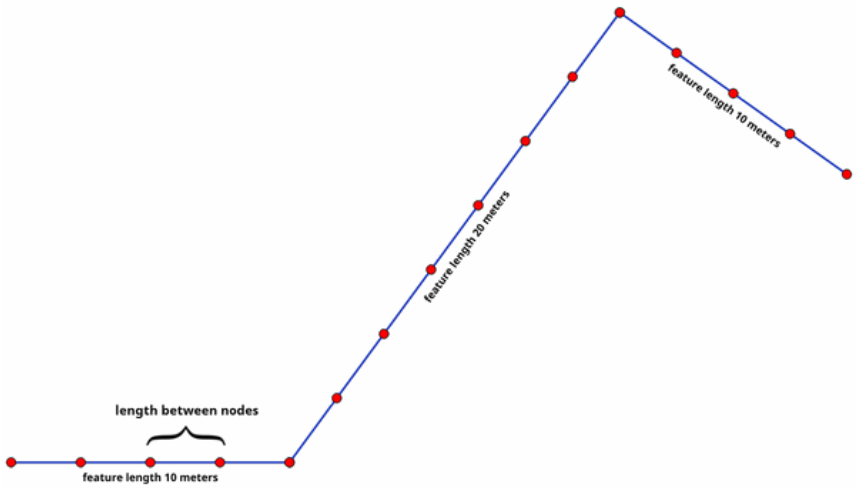


Fig. 28.77: Densify geometry at a given interval

Allows *features in-place modification* of line and polygon features

Vezi și:

Densify by count

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Interval between vertices to add	`INTERVAL`	[number] Implicit: 1.0	Maximum distance between two consecutive vertices
Densified	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Densified	`OUTPUT`	[la fel ca intrarea]	The output (densified) vector layer

Cod Python

ID Algorithm: `nativ:densifygeometriesgivenaninterval`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Dizolvare

Takes a vector layer and combines its features into new features. One or more attributes can be specified to dissolve features belonging to the same class (having the same value for the specified attributes), alternatively all features can be dissolved to a single feature.

All output geometries will be converted to multi geometries. In case the input is a polygon layer, common boundaries of adjacent polygons being dissolved will get erased. If enabled, the optional „Keep disjoint features separate” setting will cause features and parts that do not overlap or touch to be exported as separate features (instead of parts of a single multipart feature).

The resulting attribute table will have the same fields as the input layer. The values in the output layer's fields are the ones of the first input feature that happens to be processed.

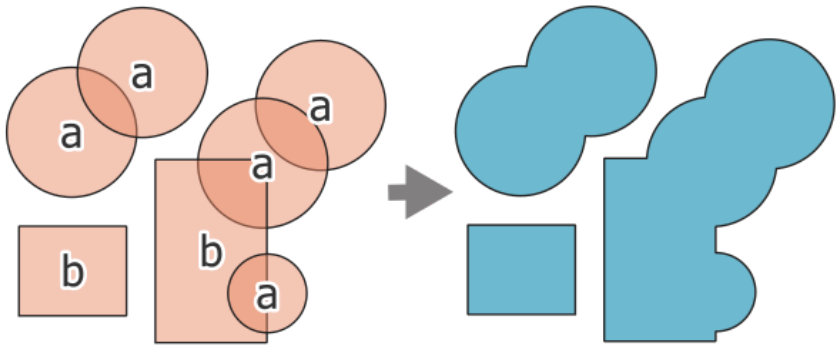


Fig. 28.78: Dissolving a layer into a single multipart feature

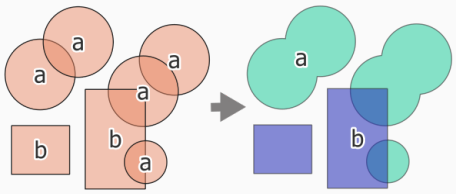
Default menu: *Vector ► Geoprocessing Tools*

Vezi şi:

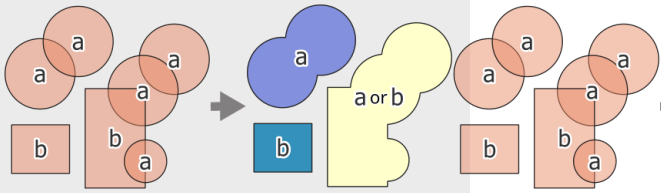
Aggregate, *Collect geometries*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Dissolve field(s) Opţional	`FIELD`	[tablefield: any] [list] Implicit: []	Features having the same value for the selected field(s) will be replaced with a single one and their geometries are merged. If no field is provided then all the features are dissolved, resulting in a single (multipart) feature.  Fig. 28.79: Dissolve the polygon layer on a common attribute (2 multipart features)
Dissolved	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fişier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Keep disjoint features separate	`SEPARATE_DISJO`	[boolean] Implicit: *False*	Parts of dissolved features are exported as separate features (instead of parts of a multipart feature).  Fig. 28.80: source (left), dissolve all (3 distinct features - right) Fig. 28.81: source (l distinct features - rig

Rezultat

Etichetă	Nume	Tipul	Descriere
Dissolved	`OUTPUT`	[la fel ca intrarea]	The output vector layer with dissolved geometries

Cod Python

ID Algorithm: `nativ:dissolve`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Drape (set Z value from raster)

Uses values sampled from a band within a raster layer to set the Z value for every overlapping vertex in the feature geometry. The raster values can optionally be scaled by a preset amount.

If Z values already exist in the layer, they will be overwritten with the new value. If no Z values exist, the geometry will be upgraded to include the Z dimension.

Allows *features in-place modification* of point, line, and polygon features with Z enabled

Vezi și:

Set M value from raster, *Set Z value*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Strat raster	`RASTER`	[raster]	Raster layer with Z values
Numărul benzii	`BAND`	[raster band] Implicit: 1	The raster band to take the Z values from
Value for NoData or non-intersecting vertices	`NODATA`	[number] Implicit: 0	Value to use in case the vertex does not intersect (a valid pixel of) the raster
Scale factor	`SCALE`	[number] Implicit: 1.0	Scaling value: the band values are multiplied by this value.
Offset	`OFFSET`	[number] Implicit: 0.0	Offset value: it is algebraically added to the band values after applying the „Scale factor”.
Updated	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer (with Z values from the raster layer). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Updated	`OUTPUT`	[la fel ca intrarea]	The output vector layer with Z values from the raster layer

Cod Python

ID Algorithm: `nativ:setzfromraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Drop M/Z values

Removes M (measure) or Z (altitude) values from input geometries.

Vezi și:

Set M value, *Set Z value*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer with M or Z values
Drop M Values	`DROP_M_VALUES`	[boolean] Implicit: *False*	Removes the M values from the geometries
Drop Z Values	`DROP_Z_VALUES`	[boolean] Implicit: *False*	Removes the Z values from the geometries
Z/M Dropped	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Z/M Dropped	`OUTPUT`	[la fel ca intrarea]	The output vector layer (identical to the input layer, except that the M and/or Z dimensions have been removed from the geometries).

Cod Python

ID Algorithm: `nativ:dropmzvalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Eliminate selected polygons

Combines selected polygons of the input layer with certain adjacent polygons by erasing their common boundary. The adjacent polygon can be either the one with the largest or smallest area or the one sharing the largest common boundary with the polygon to be eliminated.

Eliminate is normally used to get rid of sliver polygons, i.e. tiny polygons that are a result of polygon intersection processes where boundaries of the inputs are similar but not identical.

Default menu: *Vector ► Geoprocessing Tools*

Vezi și:

Fix geometries

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Input polygon vector layer
Merge selection with the neighboring polygon with the	`MODE`	[enumerare] Default: Nici o valoare	Choose the parameter to use in order to get rid of the selected polygons: • 0 — Cea Mai Mare Suprafață • 1 — Cea Mai Mică Suprafață • 2 — Cea Mai Mare Graniță Comună
Eliminated	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Eliminated	`OUTPUT`	[vector: poligon]	The output polygon vector layer.

Cod Python

Algorithm ID: `qgis:eliminateselectedpolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Explode lines

Takes a lines layer and creates a new one in which each line layer is replaced by a set of lines representing the segments in the original line.

Each line in the resulting layer contains only a start and an end point, with no intermediate vertices between them.

Fig. 28.82: The original line layer and the exploded one

Allows *features in-place modification* of line features

Vezi și:

Subdivide, *Line substring*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Exploded	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Exploded	`OUTPUT`	[vector: linie]	The output line vector layer with features representing each segment of the input layer.

Cod Python

ID Algorithm: `nativ:explodelines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extend lines

Extends line geometry by a specified amount at the start and end of the line.

Lines are extended using the bearing of the first and last segment in the line.

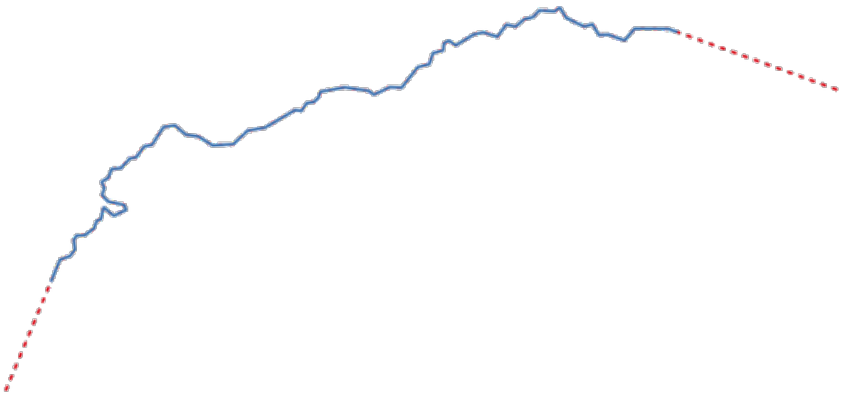


Fig. 28.83: The red dashes represent the initial and final extension of the original layer

Allows *features in-place modification* of line features

Vezi și:

Line substring

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Start distance	`START_DISTANCE`	[number]	Distance by which to extend the first segment of the line (starting point)
End distance	`END_DISTANCE`	[number]	Distance by which to extend the last segment of the line (ending point)
Extended	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extended	`OUTPUT`	[vector: linie]	The output (extended) line vector layer.

Cod Python

ID Algorithm: `nativ:extendlines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract M values

Extracts M values from geometries into feature attributes.

By default only the M value from the first vertex of each feature is extracted, however the algorithm can optionally calculate statistics on all of the geometry's M values, including sum, mean, minimum and maximum.

Vezi și:

Extract Z values, *Set M value*, *Drop M/Z values*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Summaries to calculate	`SUMMARIES`	[enumerare] Implicit: [0]	Statistics on the M values of a geometry. One or more of: • 0 — Primul • 1 — Ultimul • 2 — Total • 3 — Sumă • 4 — Medie • 5 — Median • 6 — Dev. st. (pop) • 7 — Minimum • 8 — Maximum • 9 — Interval • 10 — Minoritate • 11 — Majoritate • 12 — Varietate • 13 — Q1 • 14 — Q3 • 15 — IQR
Output column prefix	`COLUMN_PREFIX`	[string] Implicit: «m_»	The prefix for the output (M) column
Extracted	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted	`OUTPUT`	[la fel ca intrarea]	The output vector layer (with M values)

Cod Python

ID Algorithm: `nativ:extractmvalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract specific vertices

Takes a vector layer and generates a point layer with points representing specific vertices in the input geometries.

For instance, this algorithm can be used to extract the first or last vertices in the geometry. The attributes associated to each point are the same ones associated to the feature that the vertex belongs to.

The vertex indices parameter accepts a comma separated string specifying the indices of the vertices to extract. The first vertex corresponds to an index of 0, the second vertex has an index of 1, etc. Negative indices can be used to find vertices at the end of the geometry, e.g., an index of -1 corresponds to the last vertex, -2 corresponds to the second last vertex, etc.

Additional fields are added to the vertices indicating the specific vertex position (e.g., 0, -1, etc), the original vertex index, the vertex's part and its index within the part (as well as its ring for polygons), distance along the original geometry and bisector angle of vertex for the original geometry.

Allows *features in-place modification* of point features

Vezi și:

Extract vertices, *Filter vertices by M value*, *Filter vertices by Z value*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Vertex indices	`VERTICES`	[string] Implicit: «0»	Comma-separated string of the indices of the vertices to extract.
Vertices	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Vertices	`OUTPUT`	[vector: punct]	The output (point) vector layer containing the specified vertices from the input layer geometries.

Cod Python

ID Algorithm: `nativ:extractspecificvertices`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract vertices

Takes a vector layer and generates a point layer with points representing the vertices in the input geometries.

The attributes associated to each point are the same ones associated to the feature that the vertex belongs to.

Additional fields are added to the vertices indicating the vertex index (beginning at 0), the feature's part and its index within the part (as well as its ring for polygons), distance along original geometry and bisector angle of vertex for original geometry.

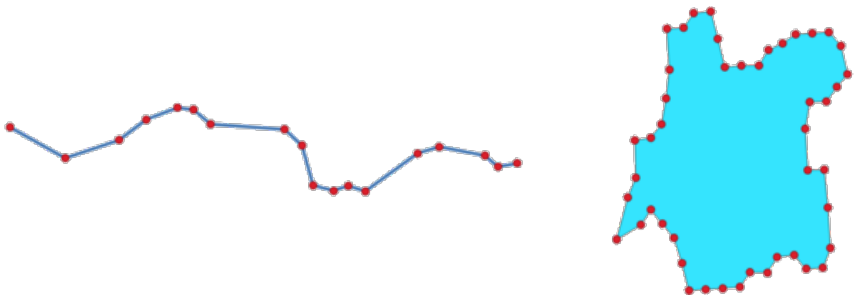


Fig. 28.84: Vertices extracted for line and polygon layer

Allows *features in-place modification* of point features

Default menu: *Vector ► Geometry Tools*

Vezi și:

Extract specific vertices, *Filter vertices by M value*, *Filter vertices by Z value*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Vertices	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Vertices	`OUTPUT`	[vector: punct]	The output (point) vector layer containing the vertices from the input layer geometries.

Cod Python

ID Algorithm: `nativ:extractvertices`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract Z values

Extracts Z values from geometries into feature attributes.

By default only the Z value from the first vertex of each feature is extracted, however the algorithm can optionally calculate statistics on all of the geometry's Z values, including sum, mean, minimum and maximum.

Vezi și:

Extract M values, *Set Z value*, *Drop M/Z values*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare

continues on next page

Tabelul 28.153 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Summaries to calculate	SUMMARIES	[enumerare] Implicit: [0]	Statistics on the Z values of a geometry. One or more of: • 0 — Primul • 1 — Ultimul • 2 — Total • 3 — Sumă • 4 — Medie • 5 — Median • 6 — Dev. st. (pop) • 7 — Minimum • 8 — Maximum • 9 — Interval • 10 — Minoritate • 11 — Majoritate • 12 — Varietate • 13 — Q1 • 14 — Q3 • 15 — IQR
Output column prefix	COLUMN_PREFIX	[string] Implicit: «z_»	The prefix for the output (Z) column
Extracted	OUTPUT	[la fel ca intrarea] Implicit: [Creare Strat temporar]	Specify the output layer. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted	OUTPUT	[la fel ca intrarea]	The output vector layer (with Z values)

Cod Python

ID Algorithm: `nativ:extractzvalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Filter vertices by M value

Filters away vertices based on their M value, returning geometries with only vertex points that have a M value greater than or equal to the specified minimum value and/or less than or equal to the maximum value.

If the minimum value is not specified then only the maximum value is tested, and similarly if the maximum value is not specified then only the minimum value is tested.

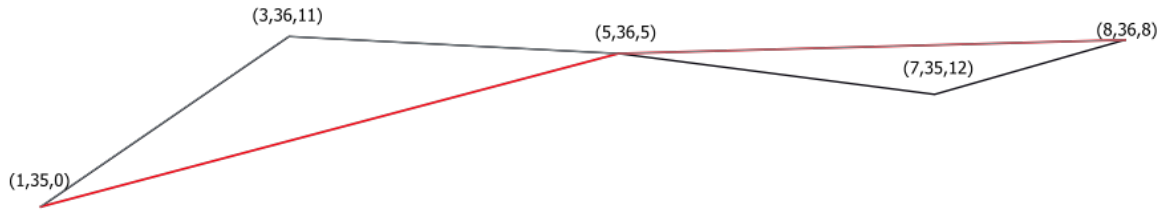


Fig. 28.85: The red line represents the black line with only vertices whose M value is <=10.

Allows *features in-place modification* of line and polygon features with M enabled

Notă: Depending on the input geometry attributes and the filters used, the resultant geometries created by this algorithm may no longer be valid.

Vezi şi:

Filter vertices by Z value, *Extract vertices*, *Extract specific vertices*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer to remove vertices from
Minimum Opţional	`MIN`	[number] Implicit: *Nesetat*	Minimum of M values allowed
Maximum Opţional	`MAX`	[number] Implicit: *Nesetat*	Maximum of M values allowed
Filtered	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fişier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Filtered	`OUTPUT`	[la fel ca intrarea]	The output vector layer of features with only the filtered vertices.

Cod Python

ID Algorithm: `nativ:filterverticesbym`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Filter vertices by Z value

Filters away vertices based on their Z value, returning geometries with only vertex points that have a Z value greater than or equal to the specified minimum value and/or less than or equal to the maximum value.

If the minimum value is not specified then only the maximum value is tested, and similarly if the maximum value is not specified then only the minimum value is tested.

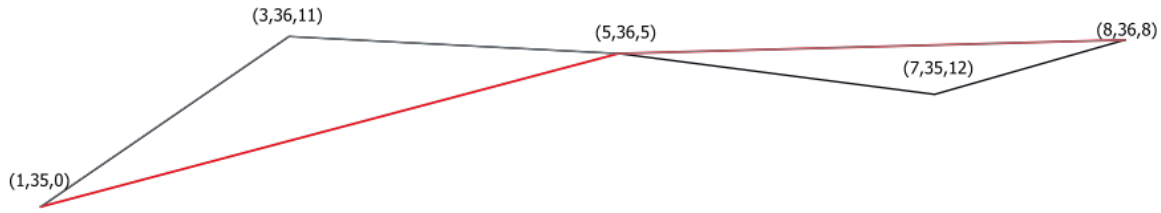


Fig. 28.86: The red line represents the black line with only vertices whose Z value is <=10.

Allows *features in-place modification* of line and polygon features with Z enabled

Notă: Depending on the input geometry attributes and the filters used, the resultant geometries created by this algorithm may no longer be valid. You may need to run the *Fix geometries* algorithm to ensure their validity.

Vezi și:

Filter vertices by M value, *Extract vertices*, *Extract specific vertices*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer to remove vertices from
Minimum Opţional	`MIN`	[number] Implicit: *Nesetat*	Minimum of Z values allowed
Maximum Opţional	`MAX`	[number] Implicit: *Nesetat*	Maximum of Z values allowed
Filtered	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Filtered	`OUTPUT`	[la fel ca intrarea]	The output vector layer of features with only the filtered vertices.

Cod Python

ID Algorithm: `nativ:filterverticesbyz`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fix geometries

Attempts to create a valid representation of a given invalid geometry without losing any of the input vertices. Already valid geometries are returned without further intervention. Always outputs multi-geometry layer.

Allows *features in-place modification* of point, line, and polygon features without M enabled

Notă: M values will be dropped from the output.

Vezi şi:

Check validity

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Repair method	`METHOD`	[enumerare] Implicit: 1	Method used to repair the geometries. One of: • 0 — `Linework`: combines all rings into a set of noded lines and then extracts valid polygons from that linework • 1 — `Structure`: first makes all rings valid and then merges shells and subtracts holes from shells to generate valid result. Assumes that holes and shells are correctly categorized. Requires QGIS version built with GEOS 3.10 or later (check *Help* ► *About* menu).
Fixed geometries	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Fixed geometries	`OUTPUT`	[la fel ca intrarea]	The output vector layer with fixed geometries.

Cod Python

ID Algorithm: `nativ:fixgeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Force right-hand-rule

Forces polygon geometries to respect the Right-Hand-Rule, in which the area that is bounded by a polygon is to the right of the boundary. In particular, the exterior ring is oriented in a clockwise direction and any interior rings in a counter-clockwise direction.

Allows *features in-place modification* of polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Stratul vectorial de intrare
Reoriented	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Reoriented	`OUTPUT`	[vector: poligon]	The output vector layer with reoriented geometries.

Cod Python

ID Algorithm: `nativ:forcerhr`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Geodesic line split at antimeridian

Splits a line into multiple geodesic segments, whenever the line crosses the antimeridian (±180 degrees longitude).

Splitting at the antimeridian helps the visual display of the lines in some projections. The returned geometry will always be a multi-part geometry.

Whenever line segments in the input geometry cross the antimeridian, they will be split into two segments, with the latitude of the breakpoint being determined using a geodesic line connecting the points either side of this segment. The current project ellipsoid setting will be used when calculating this breakpoint.

If the input geometry contains M or Z values, these will be linearly interpolated for the new vertices created at the antimeridian.

Allows *features in-place modification* of line features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Split	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Split	`OUTPUT`	[vector: linie]	The output line vector layer split at the antimeridian.

Cod Python

ID Algorithm: `nativ:antimeridiansplit`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Geometry by expression

Updates existing geometries (or creates new geometries) for input features by use of a QGIS expression.

This allows complex geometry modifications which can utilize all the flexibility of the QGIS expression engine to manipulate and create geometries for output features.

For help with QGIS expression functions, see the inbuilt help available in the *expression builder*.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare

continues on next page

Tabelul 28.154 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output geometry type	`OUTPUT_GEOMETR`	[enumerare] Implicit: 0	The output geometry strongly depends on the expression: for instance, if you create a buffer the geometry type has to be polygon. One of: • 0 — Poligon • 1 — Line • 2 — Punct
Output geometry has z values	`WITH_Z`	[boolean] Implicit: *False*	Choose if the output geometry should include the Z dimension
Output geometry has m values	`WITH_M`	[boolean] Implicit: *False*	Choose if the output geometry should include the M dimension
Geometry expression	`EXPRESSION`	[expresie] Implicit: «$geometry»	Add the geometry expression you want to use. You can use the button to open the Expression Dialog. The dialog lists all the relevant expressions, together with their help and guide.
Modified geometry	`OUTPUT`	[vector: oricare] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Modified geometry	`OUTPUT`	[vector: oricare]	The output vector layer

Cod Python

ID Algorithm: `nativ:geometrybyexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Interpolate point on line

Creates a point geometry interpolated at a set distance along line or curve geometries.

Z and M values are linearly interpolated from existing values.

If a multipart geometry is encountered, only the first part is considered when calculating the substring.

If the specified distance is greater than the input feature's length, the resultant feature will have a null geometry.

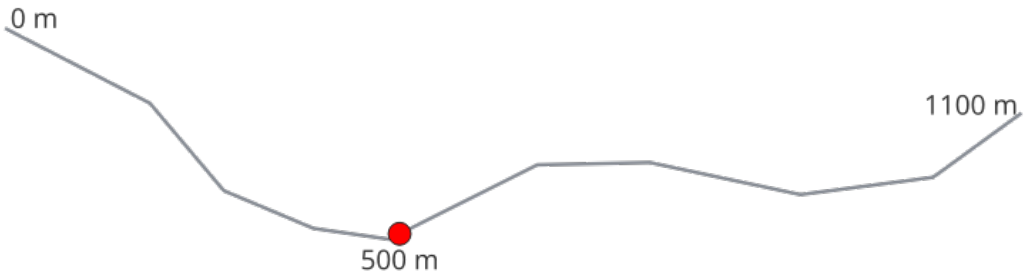


Fig. 28.87: Interpolated point at 500m of the beginning of the line

Vezi și:

Points along geometry

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Distance	`DISTANCE`	[number] Implicit: 0.0	Distance from the beginning of the line
Interpolated points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated points	`OUTPUT`	[vector: punct]	The output point vector layer with features at a set distance along the line or polygon boundary

Cod Python

ID Algorithm: `nativ:interpolatepoint`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Keep N biggest parts

Takes a layer with polygons or multipolygons and returns a new layer in which only the *n* largest polygons of each multipolygon feature are kept. If a feature has *n* or fewer parts, the feature will just be copied.

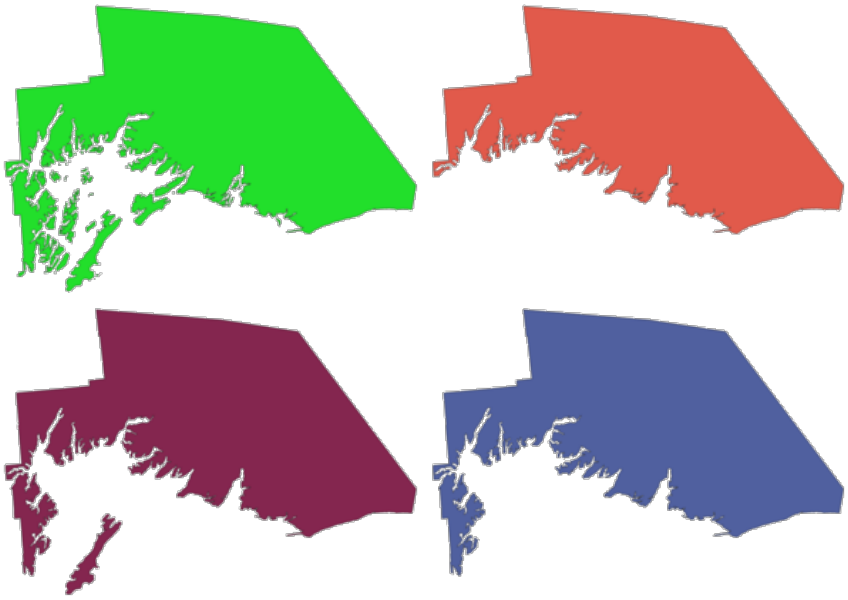


Fig. 28.88: Clockwise from top left: original multipart feature, one, two and three biggest parts kept

Parametri

Etichetă	Nume	Tipul	Descriere
Polygons	`INPUT`	[vector: poligon]	Input polygon vector layer
Parts to keep	`PARTS`	[număr] Implicit: 1	Number of parts to keep. If 1, only the biggest part of the feature will be kept.
Parts	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Parts	`OUTPUT`	[vector: poligon]	The output polygon vector layer with the N biggest parts of each feature

Cod Python

Algorithm ID: `qgis:keepnbiggestparts`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Line substring

Returns the portion of a line (or curve) which falls between the specified start and end distances (measured from the beginning of the line).

Z and M values are linearly interpolated from existing values.

If a multipart geometry is encountered, only the first part is considered when calculating the substring.

Fig. 28.89: Substring line with starting distance set at 0 meters and the ending distance at 250 meters.

Allows *features in-place modification* of line features

Vezi și:

Extend lines

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Start distance	`START_DISTANCE`	[number]	Distance along the input line to the start point of the output feature
End distance	`END_DISTANCE`	[number]	Distance along the input line to the end point of the output feature
Substring	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Substring	`OUTPUT`	[vector: linie]	The output line vector layer.

Cod Python

ID Algorithm: `nativ:linesubstring`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Lines to polygons

Generates a polygon layer using as polygon rings the lines from an input line layer.

The attribute table of the output layer is the same as the one of the input layer.

Default menu: *Vector ► Geometry Tools*

Vezi și:

Polygons to lines, *Polygonize*, *Convert geometry type*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Polygons	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Polygons	`OUTPUT`	[vector: poligon]	The output polygon vector layer.

Cod Python

Algorithm ID: `qgis:linestopolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Merge lines

Joins all connected parts of MultiLineString geometries into single LineString geometries.

If any parts of the input MultiLineString geometries are not connected, the resultant geometry will be a MultiLineString containing any lines which could be merged and any non-connected line parts.

Allows *features in-place modification* of line features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Merged	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Merged	`OUTPUT`	[vector: linie]	The output (merged) line vector layer.

Cod Python

ID Algorithm: `nativ:mergelines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Minimum bounding geometry

Creates geometries which enclose the features from an input layer. The features can be grouped by a field. The output layer will then contain one feature per group value with a geometry (MBB) that covers the geometries of the features with matching value.

The following enclosing geometry types are supported:

- bounding box (envelope)
- oriented rectangle
- circle
- convex hull

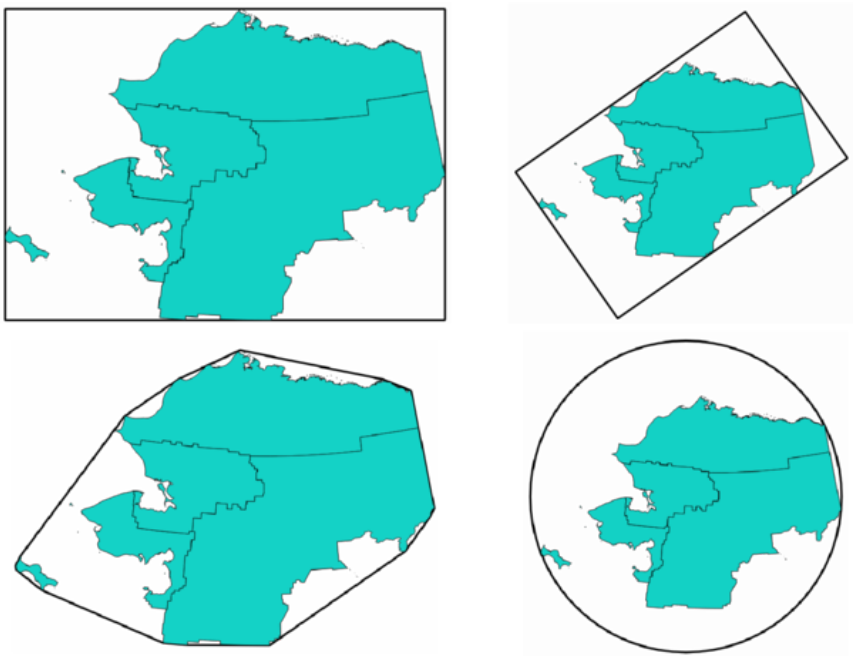


Fig. 28.90: Clockwise from top left: envelope, oriented rectangle, circle, convex hull

Vezi și:

Minimum enclosing circles

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Câmp Opţional	`FIELD`	[tablefield: any]	Features can be grouped by a field. If set, this causes the output layer to contain one feature per grouped value with a minimal geometry covering only the features with matching values.
Tipul geometriei	`TYPE`	[enumerare] Implicit: 0	Enclosing geometry types. One of: • 0 — Anvelopă (Casetă de încadrare) • 1 — Dreptunghi Orientat Minim • 2 — Cerc de Încadrare Minim • 3 — Carcasă Convexă
Bounding geometry	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Bounding geometry	`OUTPUT`	[vector: poligon]	The output (bounding) polygon vector layer.

Cod Python

Algorithm ID: `qgis:minimumboundinggeometry`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Minimum enclosing circles

Calculates the minimum enclosing circles of the features in the input layer.

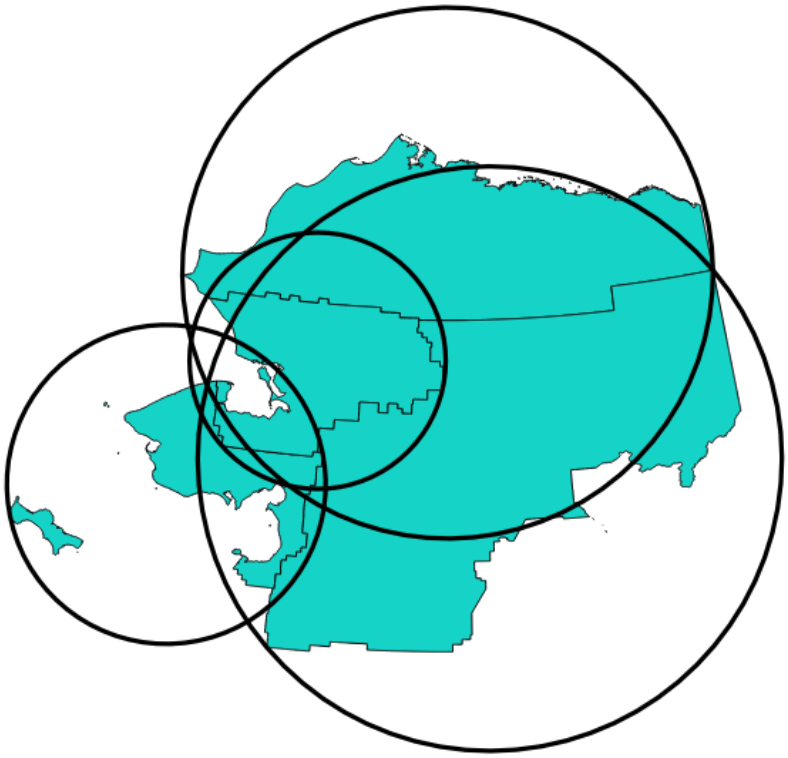


Fig. 28.91: Enclosing circles for each feature

Allows *features in-place modification* of polygon features

Vezi și:

Minimum bounding geometry

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Number of segments in circles	`SEGMENTS`	[număr] Implicit: 72	The number of segments used to approximate a circle. Minimum 8, maximum 100000.
Minimum enclosing circles	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Minimum enclosing circles	`OUTPUT`	[vector: poligon]	The output polygon vector layer.

Cod Python

ID Algorithm: `nativ:minimumenclosingcircle`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Multi-ring buffer (constant distance)

Computes multi-ring (*donut*) buffer for the features of the input layer, using a fixed or dynamic distance and number of rings.

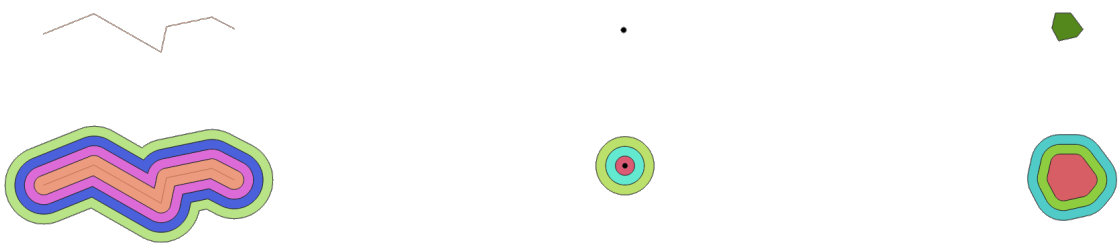


Fig. 28.92: Multi-ring buffer for a line, point and polygon layer

Allows *features in-place modification* of polygon features

Vezi și:

Buffer, *Variable distance buffer*, *Rectangles, ovals, diamonds*, *Single sided buffer*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Number of rings	`RINGS`	[number] Implicit: 1	The number of rings. It can be a unique value (same number of rings for all the features) or it can be taken from features data (the number of rings depends on feature values).
Distance between rings	`DISTANCE`	[number] Implicit: 1.0	Distance between the rings. It can be a unique value (same distance for all the features) or it can be taken from features data (the distance depends on feature values).
Multi-ring buffer (constant distance)	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Multi-ring buffer (constant distance)	`OUTPUT`	[vector: poligon]	The output polygon vector layer.

Cod Python

ID Algorithm: `nativ:multiringconstantbuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Multipart to singleparts

Splits multipart features in the input layer into singlepart features.

The attributes of the output layer are the same as the original ones but divided into single features.

Fig. 28.93: Left the multipart source layer and right the single part output result

Allows *features in-place modification* of point, line, and polygon features

Default menu: *Vector* ► *Geometry Tools*

Vezi și:

Collect geometries, *Promote to multipart*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Single parts	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Single parts	`OUTPUT`	[la fel ca intrarea]	The output vector layer.

Cod Python

ID Algorithm: `nativ:multiparttosingleparts`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Offset lines

Offsets lines by a specified distance. Positive distances will offset lines to the left, and negative distances will offset them to the right.

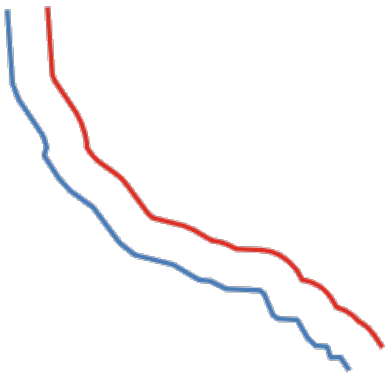


Fig. 28.94: In blue the source layer, in red the offset one

Allows *features in-place modification* of line features

Vezi și:

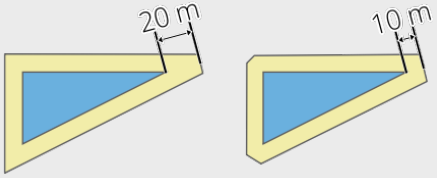
Array of offset (parallel) lines, *Translate*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Distance	`DISTANCE`	[number] Implicit: 10.0	Offset distance. You can use the Data Defined button on the right to choose a field from which the radius will be calculated. This way you can have different radius for each feature (see *Variable distance buffer*).
Segments	`SEGMENTS`	[număr] Implicit: 8	Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.
Join style	`JOIN_STYLE`	[enumerare] Implicit: 0	Specifies whether round, miter or beveled joins should be used when offsetting corners in a line. Options are: • 0 — Round • 1 — Miter • 2 — Bevel Fig. 28.95: Round, miter, and bevel join styles

continues on next page

Tabelul 28.158 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Miter limit	`MITER_LIMIT`	[număr] Implicit: 2.0	Sets the maximum distance from the offset geometry to use when creating a mitered join as a factor of the offset distance (only applicable for miter join styles). Minimum: 1.0  Fig. 28.96: A 10m buffer with a limit of 2 and a 10m buffer with a limit of 1
Offset	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output (offset) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Offset	`OUTPUT`	[vector: linie]	Output (offset) line layer

Cod Python

ID Algorithm: `nativ:offsetline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Oriented minimum bounding box

Calculates the minimum area rotated rectangle for each feature in the input layer.

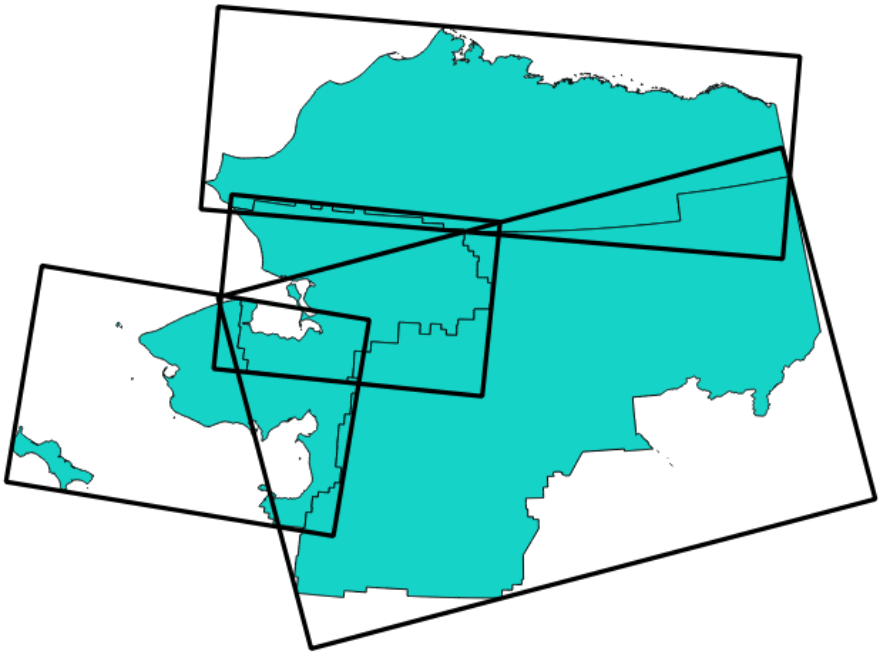


Fig. 28.97: Oriented minimum bounding box

Allows *features in-place modification* of polygon features

Vezi şi:

Minimum bounding geometry

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Bounding boxes	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Bounding boxes	`OUTPUT`	[vector: poligon]	The output polygon vector layer.

Cod Python

ID Algorithm: `nativ:orientedminimumboundingbox`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Orthogonalize

Attempts to orthogonalize the geometries of the input line or polygon layer. This process shifts the vertices in the geometries to try to make every angle in the geometry either a right angle or a straight line.

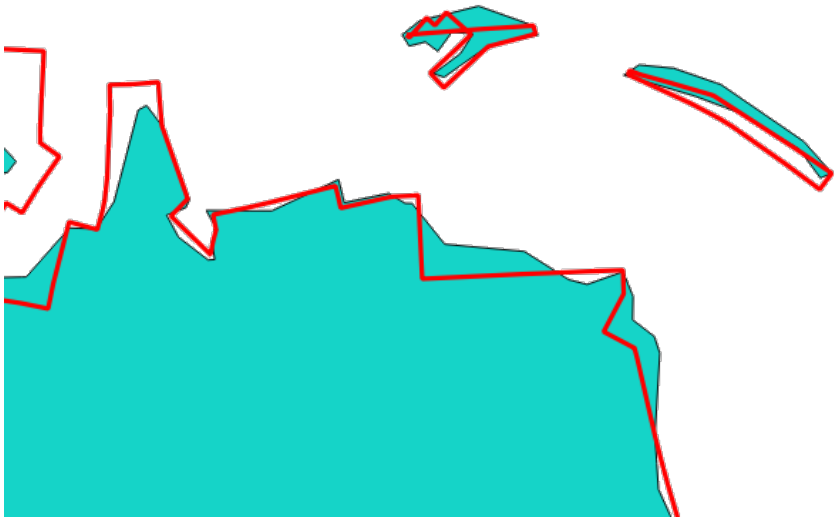


Fig. 28.98: In blue the source layer and in the red orthogonalized result

Allows *features in-place modification* of line and polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Maximum angle tolerance (degrees)	`ANGLE_TOLERANC`	[număr] Implicit: 15	Specify the maximum deviation from a right angle or straight line a vertex can have for it to be adjusted. Smaller tolerances mean that only vertices which are already closer to right angles will be adjusted, and larger tolerances mean that vertices which deviate further from right angles will also be adjusted.
Maximum algorithm iterations	`MAX_ITERATIONS`	[număr] Implicit: 1000	Setting a larger number for the maximum number of iterations will result in a more orthogonal geometry at the cost of extra processing time.
Orthogonalized	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Orthogonalized	`OUTPUT`	[la fel ca intrarea]	The output polygon vector layer with adjusted angles.

Cod Python

ID Algorithm: `nativ:orthogonalize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Point on Surface

For each feature of the input layer, returns a point that is guaranteed to lie on the surface of the feature geometry.

Allows *features in-place modification* of point features

Vezi și:

Centroids

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Create point on surface for each part	`ANGLE_TOLERANC`	[boolean]	If checked, a point will be created for each part of the geometry.
Point	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output point vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Point	`OUTPUT`	[vector: punct]	The output point vector layer.

Cod Python

ID Algorithm: `nativ:pointonsurface`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Points along geometry

Creates points at regular intervals along line or polygon geometries. Created points will have new attributes added for the distance along the geometry and the angle of the line at the point.

An optional start and end offset can be specified, which controls how far from the start and end of the geometry the points should be created.

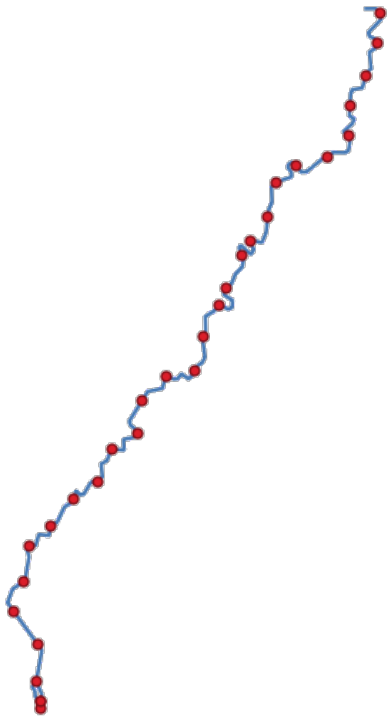


Fig. 28.99: Points created along the source line layer

Vezi și:

Interpolate point on line

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Distance	`DISTANCE`	[number] Implicit: 1.0	Distance between two consecutive points along the line
Start offset	`START_OFFSET`	[number] Implicit: 0.0	Distance from the beginning of the input line, representing the position of the first point.
End offset	`END_OFFSET`	[number] Implicit: 0.0	Distance from the end of the input line, representing the position beyond which no point feature shoud be created.

continues on next page

Tabelul 28.160 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Interpolated points	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated points	`OUTPUT`	[vector: punct]	Point vector layer with features placed along lines or polygon boundaries of the input layer.

Cod Python

ID Algorithm: `nativ:pointsalonglines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Points displacement

Given a distance of proximity, identifies nearby point features and radially distributes them over a circle whose center represents their barycenter. A convenient tool to scatter overlaid features.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Input point vector layer
Minimum distance to other points	`PROXIMITY`	[număr] Implicit: 1.0	Distance below which point features are considered close. Close features are distributed altogether.
Displacement distance	`DISTANCE`	[număr] Implicit: 1.0	Radius of the circle on which close features are placed
Horizontal distribution for two point case	`HORIZONTAL`	[boolean] Implicit: *False*	When only two points are identified as close, aligns them horizontally on the circle instead of vertically.

continues on next page

Tabelul 28.161 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Displaced	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Displaced	`OUTPUT`	[vector: punct]	Output point vector layer

Cod Python

Algorithm ID: `qgis:pointsdisplacement`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Pole of inaccessibility

Calculates the pole of inaccessibility for a polygon layer, which is the most distant internal point from the boundary of the surface.

This algorithm uses the «polylabel» algorithm (Vladimir Agafonkin, 2016), which is an iterative approach guaranteed to find the true pole of inaccessibility within a specified tolerance. A more precise tolerance (lower value) requires more iterations and will take longer to calculate.

The distance from the calculated pole to the polygon boundary will be stored as a new attribute in the output layer.

Fig. 28.100: Pole of inaccessibility

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Stratul vectorial de intrare
Tolerance	`TOLERANCE`	[număr] Implicit: 1.0	Set the tolerance for the calculation
Point	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Point	`OUTPUT`	[vector: punct]	The output point vector layer

Cod Python

ID Algorithm: `nativ:poleofinaccessibility`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Polygonize

Creates a polygon layer whose features boundaries are generated from a line layer of **closed** features.

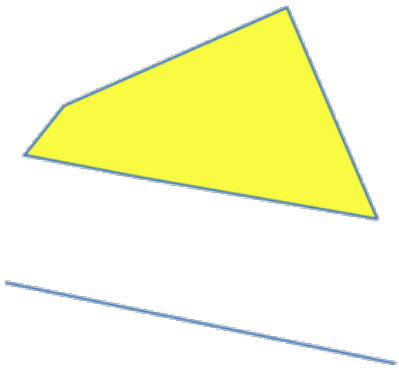


Fig. 28.101: The yellow polygons generated from the closed lines

Notă: The line layer must have closed shapes in order to be transformed into a polygon.

Vezi și:

Polygons to lines, *Lines to polygons*, *Convert geometry type*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Keep fields from the input layer Opţional	`KEEP_FIELDS`	[boolean] Implicit: *False*	Check to keep the fields (only the table structure, not the values) of the input layer
Polygons from lines	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output polygon vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Polygons from lines	`OUTPUT`	[vector: poligon]	The output polygon vector layer from lines

Cod Python

ID Algorithm: `nativ:polygonize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Polygons to lines

Takes a polygon layer and creates a line layer, with lines representing the boundaries of the polygons in the input layer.

The attribute table of the output layer is the same as the one of the input layer.

Fig. 28.102: Black lines as the result of the algorithm

Default menu: *Vector ► Geometry Tools*

Vezi și:

Lines to polygons, *Polygonize*, *Convert geometry type*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Input polygon vector layer
Lines	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Lines	`OUTPUT`	[vector: linie]	The output line vector layer from polygons

Cod Python

ID Algorithm: `nativ:polygonstolines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Project points (Cartesian)

Projects point geometries by a specified distance and bearing (azimuth).

Allows *features in-place modification* of point features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Input point vector layer
Bearing (degrees from North)	`BEARING`	[number] Implicit: 0.0	Clockwise angle starting from North, in degree (°) unit
Distance	`DISTANCE`	[number] Implicit: 1.0	Distance to offset geometries, in layer units
Projected	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output point vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Projected	`OUTPUT`	[vector: punct]	The output (projected) point vector layer

Cod Python

ID Algorithm: `nativ:projectpointcartesian`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Promote to multipart

Takes a vector layer with singlepart geometries and generates a new one in which all geometries are multipart.

Input features which are already multipart features will remain unchanged.

This algorithm can be used to force geometries to multipart types in order to be compatible with data providers that require multipart features.

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Aggregate, *Collect geometries*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Multiparts	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output multipart vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Multiparts	`OUTPUT`	[la fel ca intrarea]	The output multipart vector layer

Cod Python

ID Algorithm: `nativ:promotetomulti`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rectangles, ovals, diamonds

Creates a buffer area with a rectangle, oval or diamond shape for each feature of the input point layer.

The shape parameters can be fixed for all features or dynamic using a field or an expression.

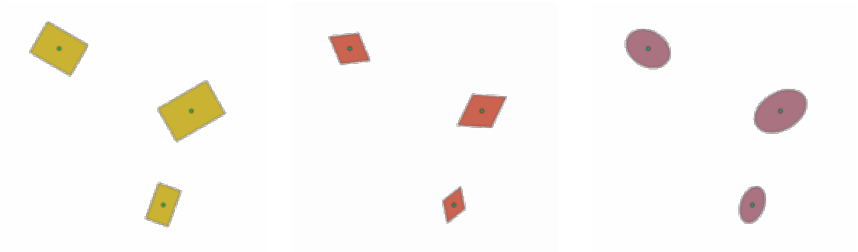


Fig. 28.103: Different buffer shapes with dynamic parameters

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Input point vector layer
Buffer shape	`SHAPE`	[enumerare]	The shape to use. One of: • 0 — Dreptunghiuri • 1 — Ovaluri • 2 — Diamante
Lățime	`WIDTH`	[number] Implicit: 1.0	Width of the buffer shape
Înălțime	`HEIGHT`	[number] Implicit: 1.0	Height of the buffer shape
Rotation Opţional	`ROTATION`	[number] Default: Nici o valoare	Rotation of the buffer shape
Number of segments	`SEGMENTS`	[număr] Implicit: 36	Number of segments for a full circle (*Ovals* shape)
Rezultat	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Rezultat	`OUTPUT`	[vector: poligon]	The output vector layer (with the buffer shapes)

Cod Python

ID Algorithm: `nativ:rectanglesovalsdiamonds`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Remove duplicate vertices

Removes duplicate vertices from features, wherever removing the vertices does not result in a degenerate geometry.

The tolerance parameter specifies the tolerance for coordinates when determining whether vertices are identical.

By default, Z values are not considered when detecting duplicate vertices. E.g. two vertices with the same X and Y coordinate but different Z values will still be considered duplicate and one will be removed. If the *Use Z Value* parameter is true, then the Z values are also tested and vertices with the same X and Y but different Z will be maintained.

Allows *features in-place modification* of point, line, and polygon features

Notă: Duplicate vertices are not tested between different parts of a multipart geometry, e.g. a multipoint geometry with overlapping points will not be changed by this method.

Vezi și:

Extract vertices, *Extract specific vertices*, *Delete duplicate geometries*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Tolerance	`TOLERANCE`	[number] Implicit: 0.000001	Vertices closer than the specified distance are considered duplicates
Use Z value	`USE_Z_VALUE`	[boolean] Implicit: *False*	If the *Use Z Value* parameter is true, then the Z values are also tested and vertices with the same X and Y but different Z will be maintained.

continues on next page

Tabelul 28.164 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Cleaned	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Cleaned	`OUTPUT`	[la fel ca intrarea]	The output vector layer (without duplicate vertices)

Cod Python

ID Algorithm: `nativ:removeduplicatevertices`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Remove null geometries

Removes any features which do not have a geometry from a vector layer. All other features will be copied unchanged.

The features with null geometries can be saved to a separate layer.

If *Also remove empty geometries* is checked, the algorithm removes features whose geometries have no coordinates, i.e., geometries that are empty. In that case, also the null output will reflect this option, containing both null and empty geometries.

Vezi și:

Delete duplicate geometries

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer (with non-NULL geometries)
Also remove empty geometries	`REMOVE_EMPTY`	[boolean]	

continues on next page

Tabelul 28.165 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Non null geometries	`OUTPUT` Opţional	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the non-NULL (and non-empty) geometries. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Null geometries Opţional	`NULL_OUTPUT`	[la fel ca intrarea] Implicit: `[Omitere rezultat]`	Specify the output vector layer for the NULL (and empty) geometries. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Null geometries	`NULL_OUTPUT`	[la fel ca intrarea]	Output vector layer (for NULL and, if chosen, empty geometries)
Non null geometries	`OUTPUT`	[la fel ca intrarea]	The output vector layer (without NULL and, if chosen, empty geometries)

Cod Python

ID Algorithm: `nativ:removenullgeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Inversează sensul liniei

Inverts the direction of a line layer.

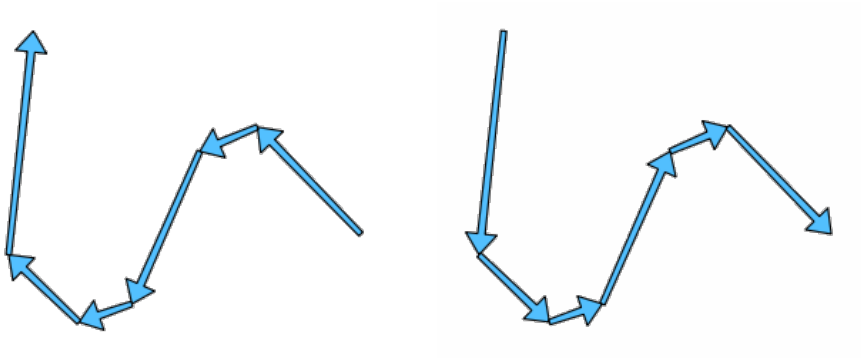


Fig. 28.104: Before and after the direction inversion

Allows *features in-place modification* of line features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Reversed	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Reversed	`OUTPUT`	[vector: linie]	The output line vector layer (with reversed lines)

Cod Python

ID Algorithm: `nativ:reverselinedirection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rotate

Rotates feature geometries by the specified angle clockwise. The rotation occurs around each feature's centroid, or optionally around a unique preset point.

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Translate, *Swap X and Y coordinates*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Rotation (degrees clockwise)	`ANGLE`	[number] Implicit: 0.0	Angle of the rotation in degrees
Rotation anchor point (x, y) Opţional	`ANCHOR`	[point] Default: Nici o valoare	X,Y coordinates of the point to rotate the features around. If not set the rotation occurs around each feature's centroid.
Rotated	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer (with rotated geometries). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Rotated	`OUTPUT`	[la fel ca intrarea]	The output vector layer with rotated geometries

Cod Python

ID Algorithm: `nativ:rotatefeatures`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Roundness

Calculates the roundness of each feature and stores it as a new field. The input vector layer must contain polygons.

The roundness of a polygon is defined as $4\pi \times$ polygon area / perimeter^2. The roundness value varies between 0 and 1. A perfect circle has a roundness of 1, while a completely flat polygon has a roundness of 0.

Notă: The algorithm returns NULL for multipart polygon features.

Allows *features in-place modification* of polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Stratul vectorial de intrare
Roundness	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output vector layer (with roundness field). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Rotated	`OUTPUT`	[la fel ca intrarea]	The output vector layer with roundness value in a field

Cod Python

Algorithm ID: `native:roundness`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Segmentize by maximum angle

Segmentizes a geometry by converting curved sections to linear sections.

The segmentization is performed by specifying the maximum allowed radius angle between vertices on the straightened geometry (e.g the angle of the arc created from the original arc center to consecutive output vertices on the linearized geometry). Non-curved geometries will be retained without change.

Vezi și:

Segmentize by maximum distance, *Simplify*, *Smooth*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Maximum angle between vertices (degrees)	`ANGLE`	[number] Implicit: 5.0	Maximum allowed radius angle between vertices on the straightened geometry
Segmentized	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer (with segmentized geometries). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Segmentized	`OUTPUT`	[la fel ca intrarea]	The output vector layer with segmentized geometries

Cod Python

ID Algorithm: `nativ:segmentizebymaxangle`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Segmentize by maximum distance

Segmentizes a geometry by converting curved sections to linear sections.

The segmentization is performed by specifying the maximum allowed offset distance between the original curve and the segmentized representation. Non-curved geometries will be retained without change.

Vezi și:

Segmentize by maximum angle, *Simplify*, *Smooth*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Maximum offset distance	`DISTANCE`	[number] Implicit: 1.0	Maximum allowed offset distance between the original curve and the segmentized representation, in the layer units.
Segmentized	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer (with segmentized geometries). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Segmentized	`OUTPUT`	[la fel ca intrarea]	The output vector layer with segmentized geometries

Cod Python

ID Algorithm: `nativ:segmentizebymaxdistance`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

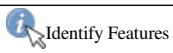
Set M value

Sets the M value for geometries in a layer.

If M values already exist in the layer, they will be overwritten with the new value. If no M values exist, the geometry will be upgraded to include M values and the specified value used as the initial M value for all geometries.

Allows *features in-place modification* of point, line, and polygon features with M enabled

---

Sfat: Use the  button to check the added M value: the results are available in the *Identify Results* dialog.

Vezi și:

Set M value from raster, *Set Z value*, *Drop M/Z values*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
M Value	`M_VALUE`	[number] Implicit: 0.0	M value to assign to the feature geometries
M Added	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
M Added	`OUTPUT`	[la fel ca intrarea]	The output vector layer (with M values assigned to the geometries)

Cod Python

ID Algorithm: `nativ:setmvalue`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Set M value from raster

Uses values sampled from a band within a raster layer to set the M value for every overlapping vertex in the feature geometry. The raster values can optionally be scaled by a preset amount.

If M values already exist in the layer, they will be overwritten with the new value. If no M values exist, the geometry will be upgraded to include M values.

Allows *features in-place modification* of point, line, and polygon features with M enabled

Vezi și:

Drape (set Z value from raster), *Set M value*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Strat raster	`RASTER`	[raster]	Raster layer with M values
Numărul benzii	`BAND`	[raster band] Implicit: 1	The raster band from which the M values are taken
Value for NoData or non-intersecting vertices	`NODATA`	[number] Implicit: 0.0	Value to use in case the vertex does not intersect (a valid pixel of) the raster
Scale factor	`SCALE`	[number] Implicit: 1.0	Scaling value: the band values are multiplied by this value.
Offset	`OFFSET`	[number] Implicit: 0.0	Offset value: it is algebraically added to the band values after applying the „Scale factor”.
Updated	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer (with updated M values). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Updated	`OUTPUT`	[la fel ca intrarea]	The output vector layer (with updated M values)

Cod Python

ID Algorithm: `nativ:setmfromraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Set Z value

Sets the Z value for geometries in a layer.

If Z values already exist in the layer, they will be overwritten with the new value. If no Z values exist, the geometry will be upgraded to include Z values and the specified value used as the initial Z value for all geometries.

Allows *features in-place modification* of point, line, and polygon features with Z enabled

---

Sfat: Use the Identify Features button to check the added Z value: the results are available in the *Identify Results* dialog.

Vezi și:

Drape (set Z value from raster), *Set M value*, *Drop M/Z values*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Z Value	`Z_VALUE`	[number] Implicit: 0.0	Z value to assign to the feature geometries
Z Added	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Z Added	`OUTPUT`	[la fel ca intrarea]	The output vector layer (with Z values assigned)

Cod Python

ID Algorithm: `nativ:setzvalue`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Simplify

Simplifies the geometries in a line or polygon layer. It creates a new layer with the same features as the ones in the input layer, but with geometries containing a lower number of vertices.

The algorithm gives a choice of simplification methods, including distance based (the „Douglas-Peucker" algorithm), area based („Visvalingam" algorithm) and snapping geometries to grid.

Fig. 28.105: Clockwise from top left: source layer and increasing simplification tolerances

Allows *features in-place modification* of line and polygon features

Default menu: *Vector* ► *Geometry Tools*

Vezi și:

Smooth, *Densify by count*, *Densify by interval*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Simplification method	`METHOD`	[enumerare] Implicit: 0	Simplification method. One of: • 0 — Distanță (Douglas-Peucker) • 1 — Acroșare la Grilă • 2 — Arie (Visvalingam)

continues on next page

Tabelul 28.167 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tolerance	`TOLERANCE`	[number] Implicit: 1.0	Threshold tolerance (in units of the layer): if the distance between two nodes is smaller than the tolerance value, the segment will be simplified and vertices will be removed.
Simplified	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (simplified) vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Simplified	`OUTPUT`	[la fel ca intrarea]	The output (simplified) vector layer

Cod Python

ID Algorithm: `nativ:simplifygeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Single sided buffer

Computes a buffer on lines by a specified distance on one side of the line only.

Buffer always results in a polygon layer.

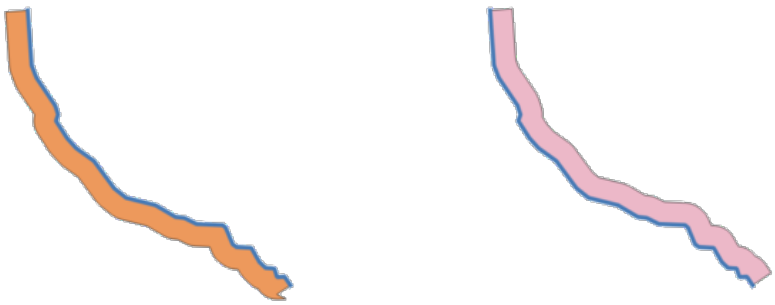
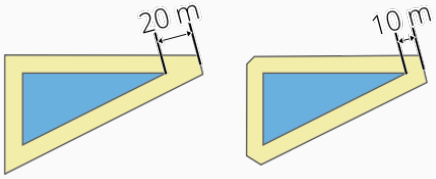


Fig. 28.106: Left versus right side buffer on the same vector line layer

Vezi şi:

Buffer

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Distance	`DISTANCE`	[număr] Implicit: 10.0	Buffer distance.
Side	`SIDE`	[enumerare] Implicit: 0	Which side to create the buffer on. One of: • 0 – Left • 1 – Right
Segments	`SEGMENTS`	[număr] Implicit: 8	Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.
Join style	`JOIN_STYLE`	[enumerare] Implicit: 0	Specifies whether round, miter or beveled joins should be used when offsetting corners in a line. Options are: • 0 — Round • 1 — Miter • 2 — Bevel Fig. 28.107: Round, miter, and bevel join styles
Miter limit	`MITER_LIMIT`	[număr] Implicit: 2.0	Sets the maximum distance from the offset geometry to use when creating a mitered join as a factor of the offset distance (only applicable for miter join styles). Minimum: 1.0  Fig. 28.108: A 10m buffer with a limit of 2 and a 10m buffer with a limit of 1
Buffer	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output (buffer) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Buffer	`OUTPUT`	[vector: poligon]	Output (buffer) polygon layer

Cod Python

ID Algorithm: `nativ:singlesidedbuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Smooth

Smooths the geometries in a line or polygon layer by adding more **vertices and corners** to the feature geometries.

The iterations parameter dictates how many smoothing iterations will be applied to each geometry. A higher number of iterations results in smoother geometries with the cost of greater number of nodes in the geometries.

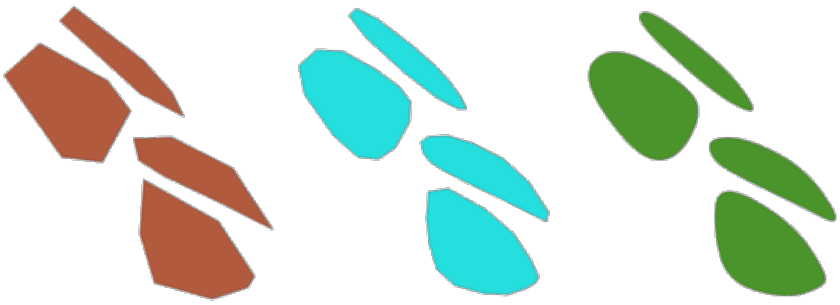


Fig. 28.109: Increasing number of iterations causes smoother geometries

The offset parameter controls how „tightly” the smoothed geometries follow the original geometries. Smaller values results in a tighter fit, and larger values will create a looser fit.

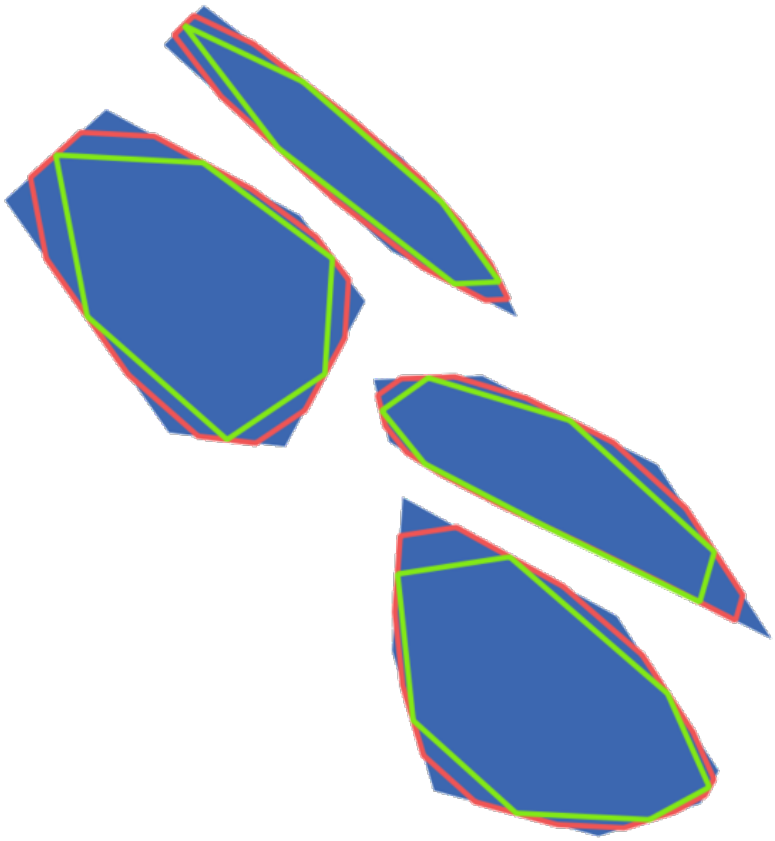


Fig. 28.110: Blue: the input layer. Offset 0.25 gives the red line, while offset 0.50 gives the green line.

The maximum angle parameter can be used to prevent smoothing of nodes with large angles. Any node where the angle of the segments to either side is larger than this will not be smoothed. For example, setting the maximum angle to 90 degrees or lower would preserve right angles in the geometry.

Allows *features in-place modification* of line and polygon features

Vezi și:

Simplify, *Densify by count*, *Densify by interval*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Input line or polygon vector layer
Iterations	`ITERATIONS`	[number] Implicit: 1	Increasing the number of iterations will give smoother geometries (and more vertices).
Offset	`OFFSET`	[number] Implicit: 0.25	Increasing values will *move* the smoothed lines / boundaries further away from the input lines / boundaries.
Maximum node angle to smooth	`MAX_ANGLE`	[number] Implicit: 180.0	Every node below this value will be smoothed

continues on next page

Tabelul 28.169 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Smoothed	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (smoothed) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Smoothed	`OUTPUT`	[la fel ca intrarea]	Output (smoothed) vector layer

Cod Python

ID Algorithm: `nativ:smoothgeometry`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Snap geometries to layer

Snaps the geometries in a layer either to the geometries from another layer, or to geometries within the same layer.

Matching is done based on a tolerance distance, and vertices will be inserted or removed as required to make the geometries match the reference geometries.

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Snap points to grid

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Reference layer	`REFERENCE_LAYE`	[vector: oricare]	Vector layer to snap to
Tolerance	`TOLERANCE`	[număr] Implicit: 10.0	Control how close input vertices need to be to the reference layer geometries before they are snapped.

continues on next page

Tabelul 28.170 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Behavior	`BEHAVIOR`	[enumerare] Implicit: 0	Snapping can be done to an existing node or a segment (its closest point to the vertex to move). Available snapping options: • 0 — Se preferă nodurile de aliniere și se introduc vârfuri suplimentare acolo unde este cazul Prefer to snap to nodes, even when a segment may be closer than a node. New nodes will be inserted to make geometries follow each other exactly when inside allowable tolerance. • 1 — Se preferă cel mai apropiat punct și se introduc vârfuri suplimentare acolo unde este cazul Snap to closest point, regardless of it is a node or a segment. New nodes will be inserted to make geometries follow each other exactly when inside allowable tolerance. • 2 — Se preferă nodurile de aliniere, fără introducerea vârfurilor suplimentare Prefer to snap to nodes, even when a segment may be closer than a node. No new nodes will be inserted. • 3 — Se preferă cel mai apropiat punct, fără introducerea vârfurilor suplimentare Snap to closest point, regardless of it is a node or a segment. No new nodes will be inserted. • 4 — Se deplasează doar punctele finale, preferându-se alinierea nodurilor Only snap start/end points of lines (point features will also be snapped, polygon features will not be modified), prefer to snap to nodes. • 5 — Se deplasează doar punctele finale, preferându-se cel mai apropiat punct Only snap start/end points of lines (point features will also be snapped, polygon features will not be modified), snap to closest point • 6 — Acroșează punctele finale doar la puncte finale Only snap the start/end points of lines to other start/end points of lines • 7 — Acroșare la nodurile de ancorare (doar un singur strat)

continues on next page

Tabelul 28.170 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Snapped geometry	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (snapped) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Snapped geometry	`OUTPUT`	[la fel ca intrarea]	Output (snapped) vector layer

Cod Python

ID Algorithm: `nativ:snapgeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Snap points to grid

Modifies the coordinates of geometries in a vector layer, so that all points or vertices are snapped to the closest point of a grid.

If the snapped geometry cannot be calculated (or is totally collapsed) the feature's geometry will be cleared.

Snapping can be performed on the X, Y, Z or M axis. A grid spacing of 0 for any axis will disable snapping for that axis.

Allows *features in-place modification* of point, line, and polygon features

Notă: Snapping to grid may generate an invalid geometry in some corner cases.

Vezi și:

Snap geometries to layer

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
X Grid Spacing	`HSPACING`	[number] Implicit: 1.0	Grid spacing on the X axis
Y Grid Spacing	`VSPACING`	[number] Implicit: 1.0	Grid spacing on the Y axis
Z Grid Spacing	`ZSPACING`	[number] Implicit: 0.0	Grid spacing on the Z axis
M Grid Spacing	`MSPACING`	[number] Implicit: 0.0	Grid spacing on the M axis
Snapped	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (snapped) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Snapped	`OUTPUT`	[la fel ca intrarea]	Output (snapped) vector layer

Cod Python

ID Algorithm: `nativ:snappointstogrid`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Split lines by maximum length

Takes a line (or curve) layer and splits each feature into multiple parts, where each part is of a specified maximum length. Z and M values at the start and end of the new line substrings are linearly interpolated from existing values.

Allows *features in-place modification* of line features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	The input line vector layer
Maximum line length	`LENGTH`	[number] Implicit: 10.0	The maximum length of a line in the output.
Split	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Split	`OUTPUT`	[vector: linie]	The new line vector layer - the length of the feature geometries is less than or equal to the length specified in the LENGTH parameter.

Cod Python

ID Algorithm: `nativ:splitlinesbylength`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Subdivide

Subdivides the geometry. The returned geometry will be a collection containing subdivided parts from the original geometry, where no part has more than the specified maximum number of nodes.

This is useful for dividing a complex geometry into less complex parts, easier to spatially index and faster to perform spatial operations. Curved geometries will be segmentized before subdivision.

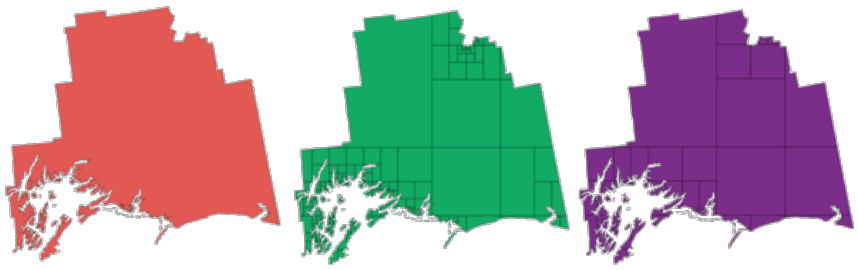


Fig. 28.111: Left the input layer, middle maximum nodes value is 100 and right maximum value is 200

Allows *features in-place modification* of point, line, and polygon features

Notă: Subdividing a geometry can generate geometry parts that may not be valid and may contain self-intersections.

Vezi și:

Explode lines, *Line substring*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer
Maximum nodes in parts	`MAX_NODES`	[number] Implicit: 256	Maximum number of vertices each new geometry part is allowed to have. Fewer *sub-parts* for higher values.
Subdivided	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output (subdivided) vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Subdivided	`OUTPUT`	[la fel ca intrarea]	Output vector layer

Cod Python

ID Algorithm: `nativ:subdivide`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Swap X and Y coordinates

Switches the X and Y coordinate values in input geometries.

It can be used to repair geometries which have accidentally had their latitude and longitude values reversed.

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Translate, *Rotate*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer
Swapped	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Swapped	`OUTPUT`	[la fel ca intrarea]	Output (swapped) vector layer

Cod Python

ID Algorithm: `nativ:swapxy`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Tapered buffers

Creates tapered buffer along line geometries, using a specified start and end buffer diameter.

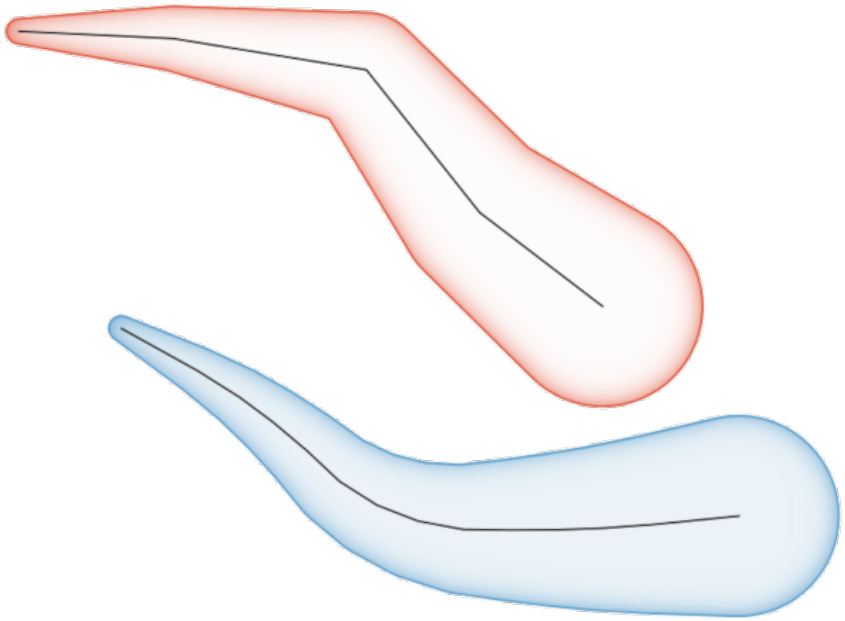


Fig. 28.112: Tapered buffer example

Vezi și:

Variable width buffer (by M value), *Buffer*, *Create wedge buffers*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Start width	`START_WIDTH`	[number] Implicit: 0.0	Represents the radius of the buffer applied at the start point of the line feature
End width	`END_WIDTH`	[number] Implicit: 0.0	Represents the radius of the buffer applied at the end point of the line feature.

continues on next page

Tabelul 28.172 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Segments	`SEGMENTS`	[number] Implicit: 16	Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.
Buffered	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output (buffer) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Buffered	`OUTPUT`	[vector: poligon]	Output (buffer) polygon layer

Cod Python

ID Algorithm: `nativ:taperedbuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Tessellate

Tessellates a polygon geometry layer, dividing the geometries into triangular components.

The output layer consists of multipolygon geometries for each input feature, with each multipolygon consisting of multiple triangle component polygons.

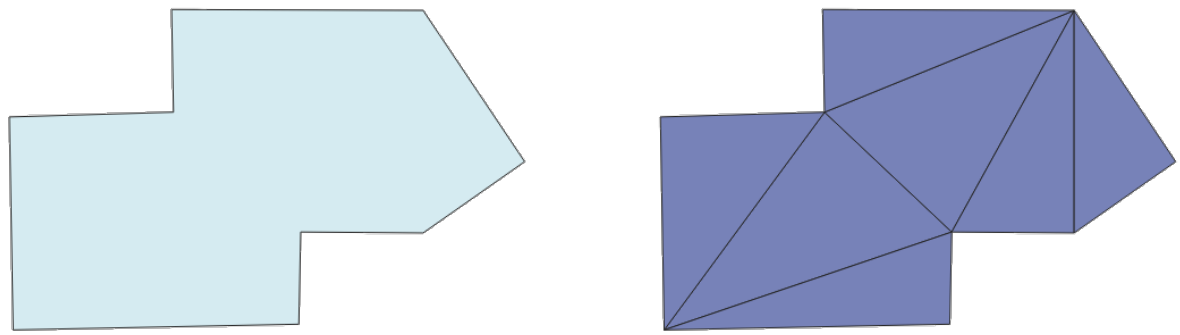


Fig. 28.113: Tessellated polygon (right)

Allows *features in-place modification* of polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: poligon]	Input polygon vector layer
Tesselated	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Tesselated	`OUTPUT`	[vector: poligon]	Output multipolygonZ layer

Cod Python

Algorithm ID: `3d:tessellate`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Transect

Creates transects on vertices for (multi)linestring.

A transect is a line oriented from an angle (by default perpendicular) to the input polylines (at vertices).

Field(s) from feature(s) are returned in the transect with these new fields:

- TR_FID: ID of the original feature
- TR_ID: ID of the transect. Each transect have an unique ID
- TR_SEGMENT: ID of the segment of the linestring
- TR_ANGLE: Angle in degrees from the original line at the vertex
- TR_LENGTH: Total length of the transect returned
- TR_ORIENT: Side of the transect (only on the left or right of the line, or both side)

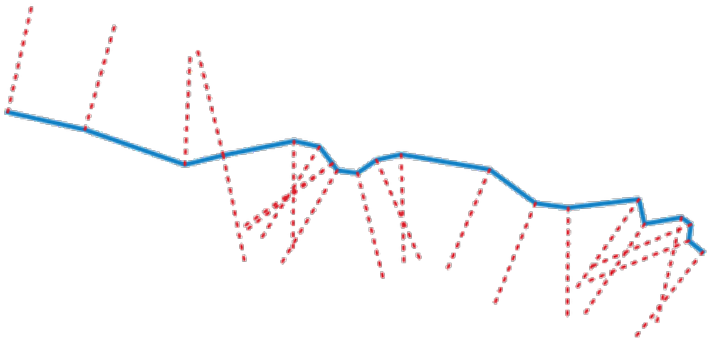


Fig. 28.114: Dashed red lines represent the transect of the input line layer

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Length of the transect	`LENGTH`	[number] Implicit: 5.0	Length in map unit of the transect
Angle in degrees from the original line at the vertices	`ANGLE`	[number] Implicit: 90.0	Change the angle of the transect
Side to create the transect	`SIDE`	[enumerare]	Choose the side of the transect. Available options are: • 0 — Stânga • 1 — Dreapta • 2 — Ambele
Transect	`OUTPUT`	[vector: linie] Implicit: `[Creare Strat temporar]`	Specify the output line layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Transect	`OUTPUT`	[vector: linie]	Output line layer

Cod Python

ID Algorithm: `nativ:transect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Translate

Moves the geometries within a layer, by offsetting with a predefined X and Y displacement.

Z and M values present in the geometry can also be translated.

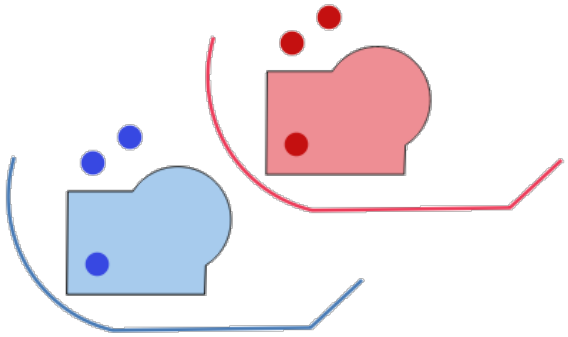


Fig. 28.115: Dashed lines represent the translated geometry of the input layer

Allows *features in-place modification* of point, line, and polygon features

Vezi și:

Array of translated features, *Offset lines*, *Rotate*, *Swap X and Y coordinates*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Offset distance (x-axis)	`DELTA_X`	[number] Implicit: 0.0	Displacement to apply on the X axis
Offset distance (y-axis)	`DELTA_Y`	[number] Implicit: 0.0	Displacement to apply on the Y axis
Offset distance (z-axis)	`DELTA_Z`	[number] Implicit: 0.0	Displacement to apply on the Z axis
Offset distance (m values)	`DELTA_M`	[number] Implicit: 0.0	Displacement to apply on the M axis

continues on next page

Tabelul 28.174 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Translated	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Translated	`OUTPUT`	[la fel ca intrarea]	Output vector layer

Cod Python

ID Algorithm: `nativ:translategeometry`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Variable width buffer (by M value)

Creates variable width buffers along lines, using the M value of the line geometries as the diameter of the buffer at each vertex.

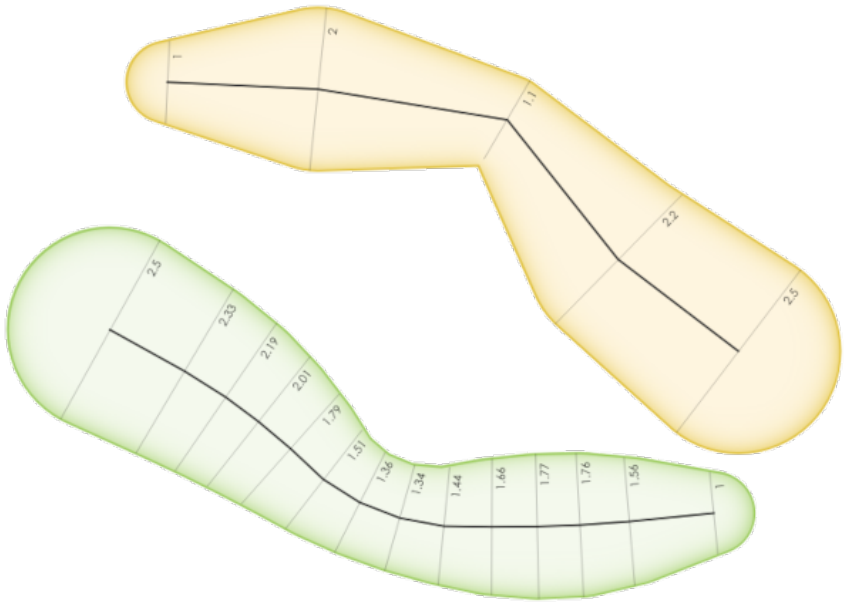


Fig. 28.116: Variable buffer example

Vezi și:

Tapered buffers, *Buffer*, *Set M value*, *Variable distance buffer*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line vector layer
Segments	`SEGMENTS`	[number] Implicit: 16	Number of the buffer segments per quarter circle. It can be a unique value (same value for all the features), or it can be taken from features data (the value can depend on feature attributes).
Buffered	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output (buffer) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Buffered	`OUTPUT`	[vector: poligon]	Variable buffer polygon layer

Cod Python

ID Algorithm: `nativ:bufferbym`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Voronoi polygons

Takes a point layer and generates a polygon layer containing the Voronoi polygons (known also as Thiessen polygons) corresponding to those input points.

Any location within a Voronoi polygon is closer to the associated point than to any other point.

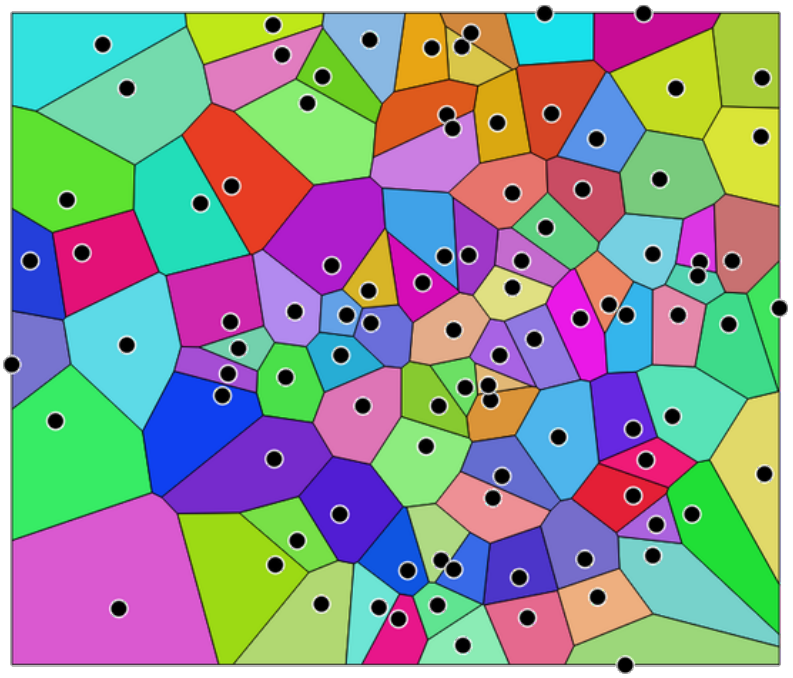


Fig. 28.117: Voronoi polygons

Default menu: *Vector ► Geometry Tools*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Input point vector layer
Buffer region (% of extent)	`BUFFER`	[număr] Implicit: 0.0	The extent of the output layer will be this much bigger than the extent of the input layer
Tolerance Opţional	`TOLERANCE`	[număr] Implicit: 0.0	Specifies an optional snapping tolerance which can be used to improve the robustness of the voronoi.
Copy attributes from input features	`COPY_ATTRIBUTE`	[boolean] Implicit: True	Specifies whether fields storing involved point features ID should be added to the output. If False, an `id` field is created to identify the polygons.
Voronoi polygons	`OUTPUT`	[vector: poligon] Implicit: `[Creare Strat temporar]`	Specify the output layer (with the Voronoi polygons). One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Voronoi polygons	`OUTPUT`	[vector: poligon]	Voronoi polygons of the input point vector layer

Cod Python

Algorithm ID: `native:voronoipolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.23 Vector overlay

Clip

Clips a vector layer using the features of an additional polygon layer.

Only the parts of the features in the input layer that fall within the polygons of the overlay layer will be added to the resulting layer.

Atenționare: Geometry modification only

This operation modifies only the features geometry. The attribute values of the features are **not modified**, although properties such as area or length of the features will be modified by the overlay operation. If such properties are stored as attributes, those attributes will have to be manually updated.

This algorithm uses spatial indexes on the providers, prepared geometries and apply a clipping operation if the geometry isn't wholly contained by the mask geometry.

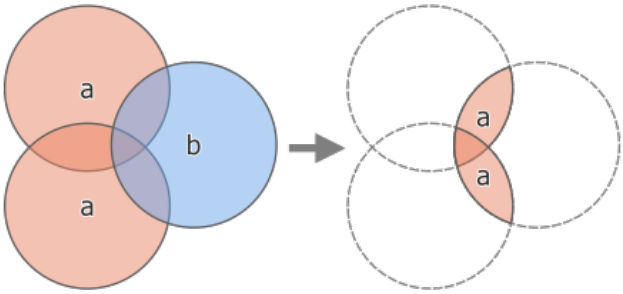


Fig. 28.118: Clipping operation between a two-feature input layer «a» and a single feature overlay layer «b» (left) - resulting in a new layer with the modified «a» features (right)

Allows *features in-place modification* of point, line, and polygon features

Default menu: *Vector* ► *Geoprocessing Tools*

Vezi și:

Intersection, *Difference*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer containing the features to be clipped
Overlay layer	`OVERLAY`	[vector: poligon]	Layer containing the clipping features
Clipped	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the features from the input layer that are inside the overlay (clipping) layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Clipped	`OUTPUT`	[la fel ca intrarea]	Layer containing features from the input layer split by the overlay layer.

Cod Python

Algorithm ID: `qgis:clip`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Difference

Extracts features from the input layer that don't fall within the boundaries of the overlay layer.

Input layer features that partially overlap the overlay layer feature(s) are split along the boundary of those feature(s) and only the portions outside the overlay layer features are retained.

Atenționare: Geometry modification only

This operation modifies only the features geometry. The attribute values of the features are **not modified**, although properties such as area or length of the features will be modified by the overlay operation. If such properties are stored as attributes, those attributes will have to be manually updated.

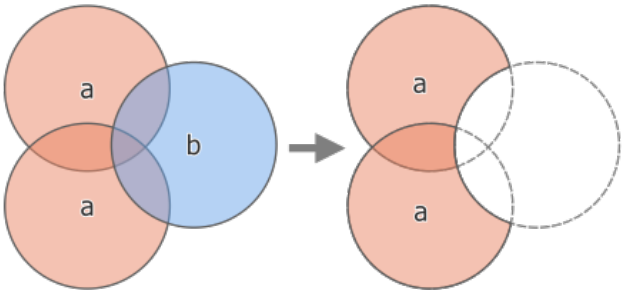


Fig. 28.119: Difference operation between a two-features input layer «a» and a single feature overlay layer «b» (left) - resulting in a new layer with the modified «a» features (right)

Allows *features in-place modification* of point, line, and polygon features

Default menu: *Vector ► Geoprocessing Tools*

Vezi și:

Difference (multiple), *Symmetrical difference*, *Clip*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to extract (parts of) features from.
Overlay layer	`OVERLAY`	[vector: oricare]	Layer containing the geometries that will be subtracted from the input layer geometries. It is expected to have at least as many dimensions (point: 0D, line: 1D, polygon: 2D, volume: 3D) as the input layer geometries.
Difference	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the (parts of) features from the input layer that are not inside the overlay layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Grid size Opţional	`GRID_SIZE`	[număr] Implicit: Nesetat	If provided, the input geometries are snapped to a grid of the given size, and the result vertices are computed on that same grid. Requires GEOS 3.9.0 or higher.

Rezultat

Etichetă	Nume	Tipul	Descriere
Difference	`OUTPUT`	[la fel ca intrarea]	Layer containing (parts of) features from the input layer not overlapping the overlay layer.

Cod Python

Algorithm ID: `qgis:difference`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Difference (multiple)

Extracts features from the input layer that fall completely outside or only partially overlap the features from any of the overlay layer(s).

For each overlay layer the difference is calculated between the result of all previous difference operations and this overlay layer. Input layer features that partially overlap feature(s) in the overlay layer are split along those features» boundary and only the portions outside the overlay layer features are retained.

Atenționare: Geometry modification only

This operation modifies only the features geometry. The attribute values of the features are **not modified**, although properties such as area or length of the features will be modified by the overlay operation. If such properties are stored as attributes, those attributes will have to be manually updated.

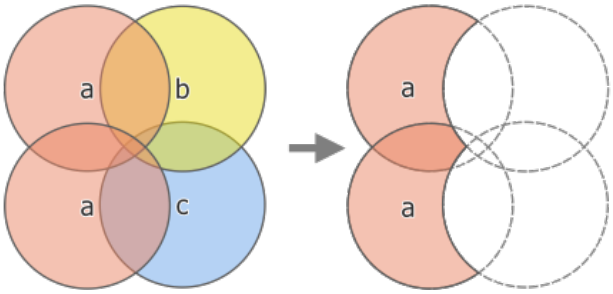


Fig. 28.120: Difference operation between a two-feature input layer «a» and single feature overlay layers «b» and «c» (left) - resulting in a new layer with the modified «a» features (right)

Vezi și:

Difference, *Symmetrical difference*, *Clip*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to extract (parts of) features from.
Overlay layers	`OVERLAYS`	[vector: oricare] [list]	List of layers containing the geometries that will be subtracted from the input layer geometries. They are expected to have at least as many dimensions (point: 0D, line: 1D, polygon: 2D, volume: 3D) as the input layer geometries.
Difference	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the (parts of) features from the input layer that do not overlap features of the overlay layers. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Difference	`OUTPUT`	[la fel ca intrarea]	Layer containing (parts of) features from the input layer not overlapping features from the overlay layers.

Cod Python

Algorithm ID: `qgis:multidifference`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract/clip by extent

Creates a new vector layer that only contains features which fall within a specified extent.

Any features which intersect the extent will be included.

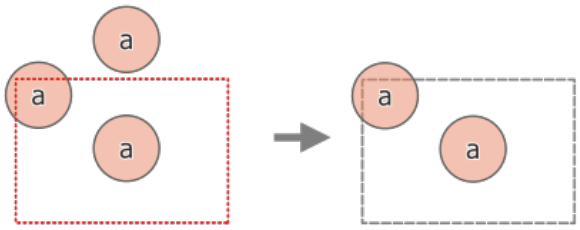
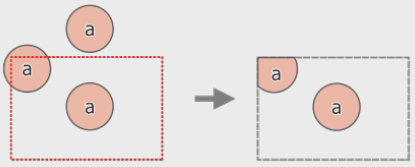


Fig. 28.121: Extract operation between a three-feature input layer «a» and a dashed extent (left) - resulting features with dashed extent for reference (right)

Vezi și:

Clip

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to extract (parts of) features from.
Extent (xmin, xmax, ymin, ymax)	`EXTENT`	[extent]	Extent for clipping. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Clip features to extent	`CLIP`	[boolean] Implicit: *False*	If checked, output geometries will be automatically converted to multi geometries to ensure uniform output types. Moreover the geometries will be clipped to the extent chosen instead of taking the whole geometry as output.  Fig. 28.122: Extract operation between a three-feature input layer «a» and a dashed extent (left) - resulting features with dashed extent for reference (right)
Extracted	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the features from the input layer that are inside the clip extent. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted	`OUTPUT`	[la fel ca intrarea]	Layer containing the clipped features.

Cod Python

Algorithm ID: `qgis:extractbyextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Intersection

Extracts the portions of features from the input layer that overlap features in the overlay layer.

Features in the intersection layer are assigned the attributes of the overlapping features from both the input and overlay layers.

Atenționare: Geometry modification only

This operation modifies only the features geometry. The attribute values of the features are **not modified**, although properties such as area or length of the features will be modified by the overlay operation. If such properties are stored as attributes, those attributes will have to be manually updated.

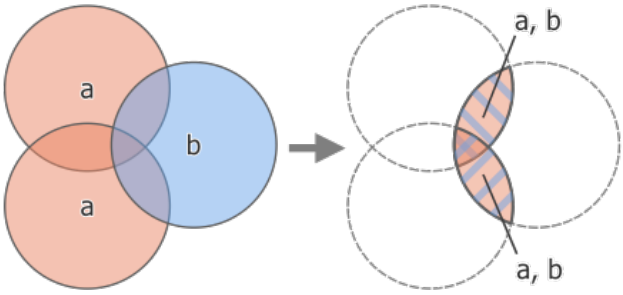


Fig. 28.123: Intersection operation between a two-feature input layer «a» and a single feature overlay layer «b» (left) - overlapping areas become a new two-feature layer with both layers» attributes (right)

Default menu: *Vector* ► *Geoprocessing Tools*

Vezi și:

Intersection (multiple), *Clip*, *Difference*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to extract (parts of) features from.
Overlay layer	`OVERLAY`	[vector: oricare]	Layer containing the features to check for overlap. Its features» geometry is expected to have at least as many dimensions (point: 0D, line: 1D, polygon: 2D, volume: 3D) as the input layer's.
Input fields to keep (leave empty to keep all fields) Opţional	`INPUT_FIELDS`	[tablefield: any] [list] Default: Nici o valoare	Field(s) of the input layer to keep in the output. If no fields are chosen all fields are taken.
Overlay fields to keep (leave empty to keep all fields) Opţional	`OVERLAY_FIELDS`	[tablefield: any] [list] Default: Nici o valoare	Field(s) of the overlay layer to keep in the output. If no fields are chosen all fields are taken. Duplicate field names will be appended a count suffix to avoid collision.
Intersection	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain (the parts of) the features from the input layer that overlap one or more features from the overlay layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Overlay fields prefix Opţional	`OVERLAY_FIELDS_`	[string]	Add a prefix to identify fields of the overlay layer. Duplicate field names will be appended a count suffix to avoid collision.
Grid size Opţional	`GRID_SIZE`	[număr] Implicit: Nesetat	If provided, the input geometries are snapped to a grid of the given size, and the result vertices are computed on that same grid. Requires GEOS 3.9.0 or higher.

Rezultat

Etichetă	Nume	Tipul	Descriere
Intersection	`OUTPUT`	[la fel ca intrarea]	Layer containing (parts of) features from the input layer that overlap the overlay layer.

Cod Python

Algorithm ID: `qgis:intersection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Intersection (multiple)

Extracts the overlapping portions of features in the input and all overlay layers.

Features in the output layer are assigned the attributes of the overlapping features from both the input and overlay layers.

Atenționare: Geometry modification only

This operation modifies only the features geometry. The attribute values of the features are **not modified**, although properties such as area or length of the features will be modified by the overlay operation. If such properties are stored as attributes, those attributes will have to be manually updated.

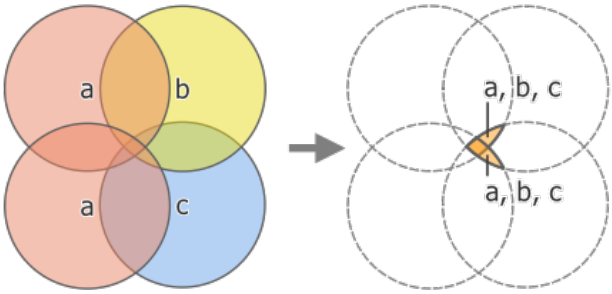


Fig. 28.124: Intersection operation between a two-feature input layer «a» and single feature overlay layers «b» and «c» (left) - overlapping areas become a new two-feature layer with all layers» attributes (right)

Vezi și:

Intersection, *Clip*, *Difference*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to extract (parts of) features from.
Overlay layers	`OVERLAYS`	[vector: oricare] [list]	Layers containing the features to check for overlap. The features» geometry is expected to have at least as many dimensions (point: 0D, line: 1D, polygon: 2D, volume: 3D) as the input layer's.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Overlay fields prefix Opţional	`OVERLAY_FIELDS_`	[string]	Add a prefix to identify fields of the overlay layers. Duplicate field names will be appended a count suffix to avoid collision.

Rezultat

Etichetă	Nume	Tipul	Descriere
Intersection	`OUTPUT`	[la fel ca intrarea]	Layer containing (parts of) features from the input layer that overlap all the overlay layers.

Cod Python

Algorithm ID: `qgis:multiintersection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Line intersections

Creates point features where the lines from the two layers intersect.

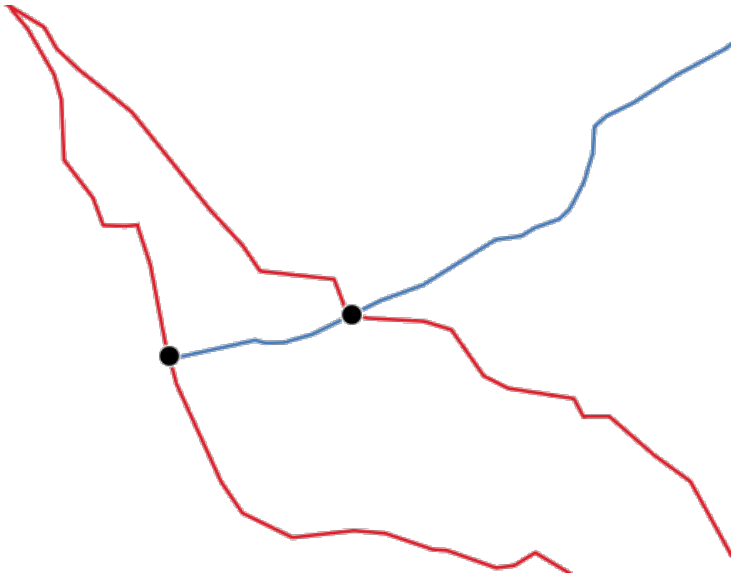


Fig. 28.125: Points of intersection

Default menu: *Vector* ► *Analysis Tools*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie]	Input line layer.
Intersect layer	`INTERSECT`	[vector: linie]	Layer to use to find line intersections.
Input fields to keep (leave empty to keep all fields) Opţional	`INPUT_FIELDS`	[tablefield: any] [list] Default: Nici o valoare	Field(s) of the input layer to keep in the output. If no fields are chosen all fields are taken.
Intersect fields to keep (leave empty to keep all fields) Opţional	`INTERSECT_FIEL`	[tablefield: any] [list] Default: Nici o valoare	Field(s) of the intersect layer to keep in the output. If no fields are chosen all fields are taken. Duplicate field names will be appended a count suffix to avoid collision.
Intersection	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the intersection points of the lines from the input and overlay layers. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansaţi

Etichetă	Nume	Tipul	Descriere
Intersect fields prefix Opţional	`INTERSECT_FIEL`	[string]	Add a prefix to identify fields of the intersect layer.

Rezultat

Etichetă	Nume	Tipul	Descriere
Intersections	`OUTPUT`	[vector: punct]	Point vector layer of the lines intersections, with both layers» attributes.

Cod Python

Algorithm ID: `qgis:lineintersections`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Split with lines

Splits the lines or polygons in one layer using the lines or polygon rings in another layer to define the breaking points. Intersection between geometries in both layers are considered as split points.

Output will contain multi geometries for split features.

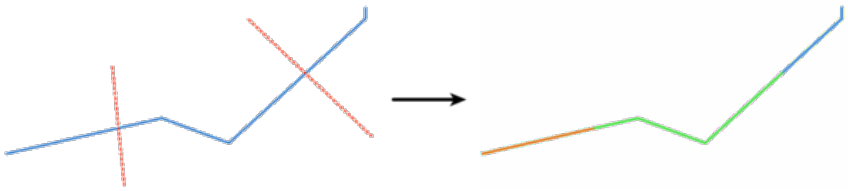


Fig. 28.126: Split lines

Allows *features in-place modification* of line and polygon features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: linie, poligon]	Layer containing the lines or polygons to split.
Split layer	`LINES`	[vector: linie, poligon]	Layer whose lines or rings are used to define the breaking points.
Split	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the splitted (in case they are intersected by a line in the split layer) line/polygon features from the input layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Split	`OUTPUT`	[la fel ca intrarea]	Output vector layer with split lines or polygons from input layer.

Cod Python

Algorithm ID: `qgis:splitwithlines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Symmetrical difference

Creates a layer containing features from both the input and overlay layers but with the overlapping areas between the two layers removed.

The attribute table of the symmetrical difference layer contains attributes and fields from both the input and overlay layers.

Atenționare: Geometry modification only

This operation modifies only the features geometry. The attribute values of the features are **not modified**, although properties such as area or length of the features will be modified by the overlay operation. If such properties are stored as attributes, those attributes will have to be manually updated.

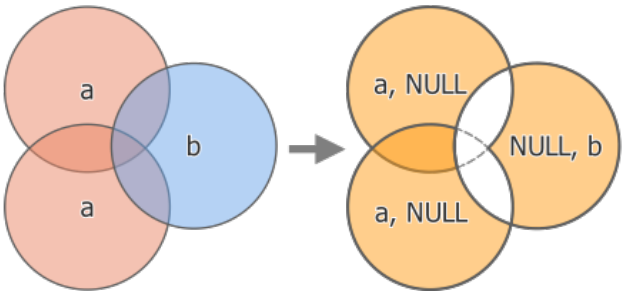


Fig. 28.127: Symmetrical difference operation between a two-features input layer «a» and a single feature overlay layer «b» (left) - resulting three-feature layer with both layers» attributes (right)

Default menu: *Vector ► Geoprocessing Tools*

Vezi și:

Difference, *Clip*, *Intersection*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	First layer to extract (parts of) features from.
Overlay layer	`OVERLAY`	[vector: oricare]	Second layer to extract (parts of) features from. Ideally the geometry type should be the same as input layer.
Symmetrical difference	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain (the parts of) the features from the input and overlay layers that do not overlap features from the other layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Overlay fields prefix Opţional	`OVERLAY_FIELDS_`	[string]	Add a prefix to identify fields of the overlay layer. Duplicate field names will be appended a count suffix to avoid collision.
Grid size Opţional	`GRID_SIZE`	[număr] Implicit: Nesetat	If provided, the input geometries are snapped to a grid of the given size, and the result vertices are computed on that same grid. Requires GEOS 3.9.0 or higher.

Rezultat

Etichetă	Nume	Tipul	Descriere
Symmetrical difference	`OUTPUT`	[la fel ca intrarea]	Layer containing (parts of) features from each layer not overlapping the other layer, with both layers» attributes.

Cod Python

Algorithm ID: `qgis:symmetricaldifference`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Union

Checks overlaps between features within the input layer and creates separate features for overlapping and non-overlapping parts. The area of overlap will create as many identical overlapping features as there are features that participate in that overlap.

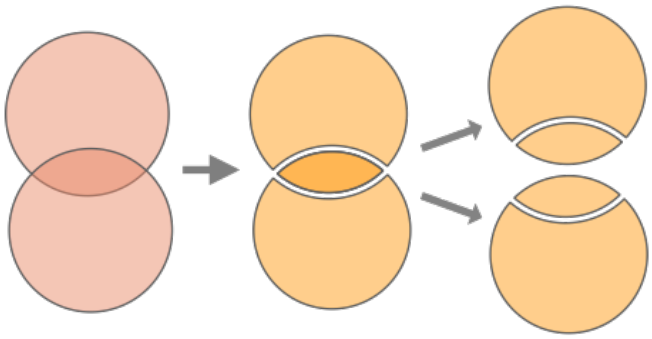


Fig. 28.128: Union operation with a single input layer with two overlapping features (left) - resulting in four features (middle) - features moved for clarity (right)

An overlay layer can also be used, in which case features from each layer are split at their overlap with features from the other one, creating a layer containing all the portions from both input and overlay layers. Features on the same layer will not split each other. The attribute table of the union layer is filled with attribute values from the respective original layer for non-overlapping features, and attribute values from both layers for overlapping features.

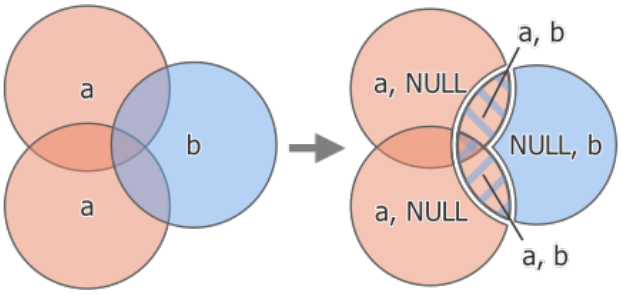


Fig. 28.129: Union operation between a two-feature input layer «a» and a single feature overlay layer «b» (left) - resulting five-feature layer with attributes from both layers (right)

Notă: With an overlay layer, features on the same layer will not split each other. If you want to split overlaps on the same layer as well as other layers, first run the algorithm with multiple layers then run the algorithm again with only the previous output.

Default menu: *Vector* ► *Geoprocessing Tools*

Vezi şi:

Union (multiple), *Clip*, *Difference*, *Intersection*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer to split at any intersections.
Overlay layer Opţional	`OVERLAY`	[vector: oricare]	Layer that will be combined to the first one. Ideally the geometry type should be the same as input layer.
Union	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the (split and duplicated) features from the input layer and the overlay layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansaţi

Etichetă	Nume	Tipul	Descriere
Overlay fields prefix Opţional	`OVERLAY_FIELDS_`	[string]	Add a prefix to identify fields of the overlay layer. Duplicate field names will be appended a count suffix to avoid collision.
Grid size Opţional	`GRID_SIZE`	[număr] Implicit: Nesetat	If provided, the input geometries are snapped to a grid of the given size, and the result vertices are computed on that same grid. Requires GEOS 3.9.0 or higher.

Rezultat

Etichetă	Nume	Tipul	Descriere
Union	`OUTPUT`	[la fel ca intrarea]	Layer containing all the overlapping and non-overlapping parts from the processed layer(s).

Cod Python

Algorithm ID: `qgis:union`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Union (multiple)

Checks overlaps between features within the input layer and creates separate features for overlapping and non-overlapping parts. The area of overlap will create as many identical overlapping features as there are features that participate in that overlap.

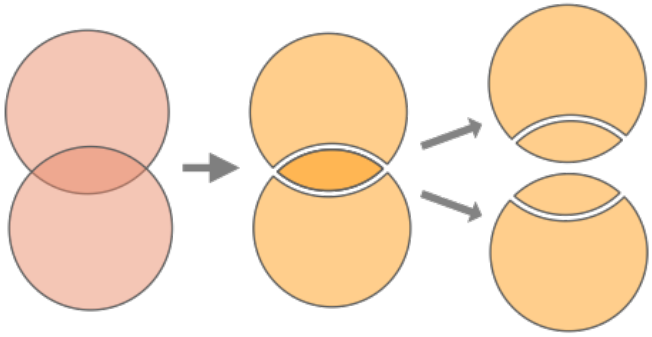


Fig. 28.130: Union operation with a single input layer with two overlapping features (left) - resulting in four features (middle) - features moved for clarity (right)

Multiple overlay layers can also be used, in which case features from each layer are split at their overlap with features from all other layers, creating a layer containing all the portions from both input and overlay layers. Features on the same layer will not split each other. The attribute table of the Union layer is filled with attribute values from the respective original layer for non-overlapping features, and attribute values from overlay layers for overlapping features.

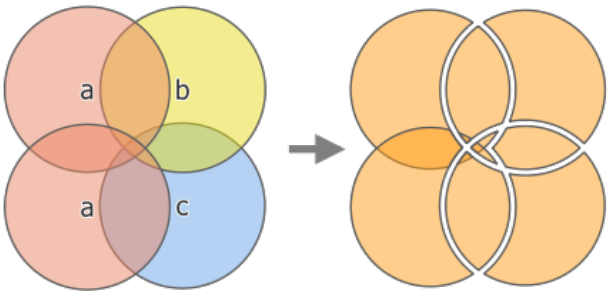


Fig. 28.131: Union operation between a two-feature input layer «a» and single feature overlay layers «b» and «c» (left) - resulting eleven-feature layer with attributes from all layers (right)

Notă: With an overlay layer, features on the same layer will not split each other. If you want to split overlaps on the same layer as well as other layers, first run the algorithm with multiple layers then run the algorithm again with only the previous output.

Vezi şi:

Union, *Clip*, *Difference*, *Intersection*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer to split at any intersections.
Overlay layers Opţional	`OVERLAYS`	[vector: oricare] [list]	Layers that will be combined to the first one. Ideally the geometry type should be the same as input layer.
Union	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the layer to contain the (split and duplicated) features from the input layer and the overlay layers. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Overlay fields prefix Opţional	`OVERLAY_FIELDS_`	[string]	Add a prefix to identify fields of the overlay layers. Duplicate field names will be appended a count suffix to avoid collision.

Rezultat

Etichetă	Nume	Tipul	Descriere
Union	`OUTPUT`	[la fel ca intrarea]	Layer containing all the overlapping and non-overlapping parts from the processed layer(s), with all layers» attributes.

Cod Python

Algorithm ID: `qgis:multiunion`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.24 Vector selection

Extract by attribute

Creates two vector layers from an input layer: one will contain only matching features while the second will contain all the non-matching features.

The criteria for adding features to the resulting layer is based on the values of an attribute from the input layer.

Vezi și:

Select by attribute

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Layer to extract features from.
Selection attribute	`FIELD`	[tablefield: any]	Filtering field of the layer
Operator	`OPERATOR`	[enumerare] Implicit: 0	Many different operators are available: • 0 — = • 1 — ≠ • 2 — >>• 3 — >= • 4 — < • 5 — <= • 6 — începe cu • 7 — conţine • 8 — is null • 9 — is not null • 10 — nu conține
Value Opţional	`VALUE`	[string]	Value to be evaluated
Extracted (attribute)	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for matching features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

continues on next page

Tabelul 28.188 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Extracted (non-matching)	`FAIL_OUTPUT`	[la fel ca intrarea] Implicit: `[Omitere rezultat]`	Specify the output vector layer for non-matching features. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted (attribute)	`OUTPUT`	[la fel ca intrarea]	Vector layer with matching features from the input layer
Extracted (non-matching)	`FAIL_OUTPUT`	[la fel ca intrarea]	Vector layer with non-matching features from the input layer

Cod Python

Algorithm ID: `qgis:extractbyattribute`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract by expression

Creates two vector layers from an input layer: one will contain only matching features while the second will contain all the non-matching features.

The criteria for adding features to the resulting layer is based on a QGIS expression. For more information about expressions see the *Expresii*.

Vezi și:

Select by expression

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Expression	`EXPRESSION`	[expresie]	Expression to filter the vector layer
Matching features	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for matching features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Non-matching	`FAIL_OUTPUT`	[la fel ca intrarea] Implicit: `[Omitere rezultat]`	Specify the output vector layer for non-matching features. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Matching features	`OUTPUT`	[la fel ca intrarea]	Vector layer with matching features from the input layer
Non-matching	`FAIL_OUTPUT`	[la fel ca intrarea]	Vector layer with non-matching features from the input layer

Cod Python

Algorithm ID: `qgis:extractbyexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract by location

Creates a new vector layer that only contains matching features from an input layer.

The criteria for adding features to the resulting layer is based on the spatial relationship between each feature and the features in an additional layer.

Vezi și:

Selectare după locație, *Extract within distance*

Exploring spatial relations

Geometric predicates are boolean functions used to determine the spatial relation a feature has with another by comparing whether and how their geometries share a portion of space.

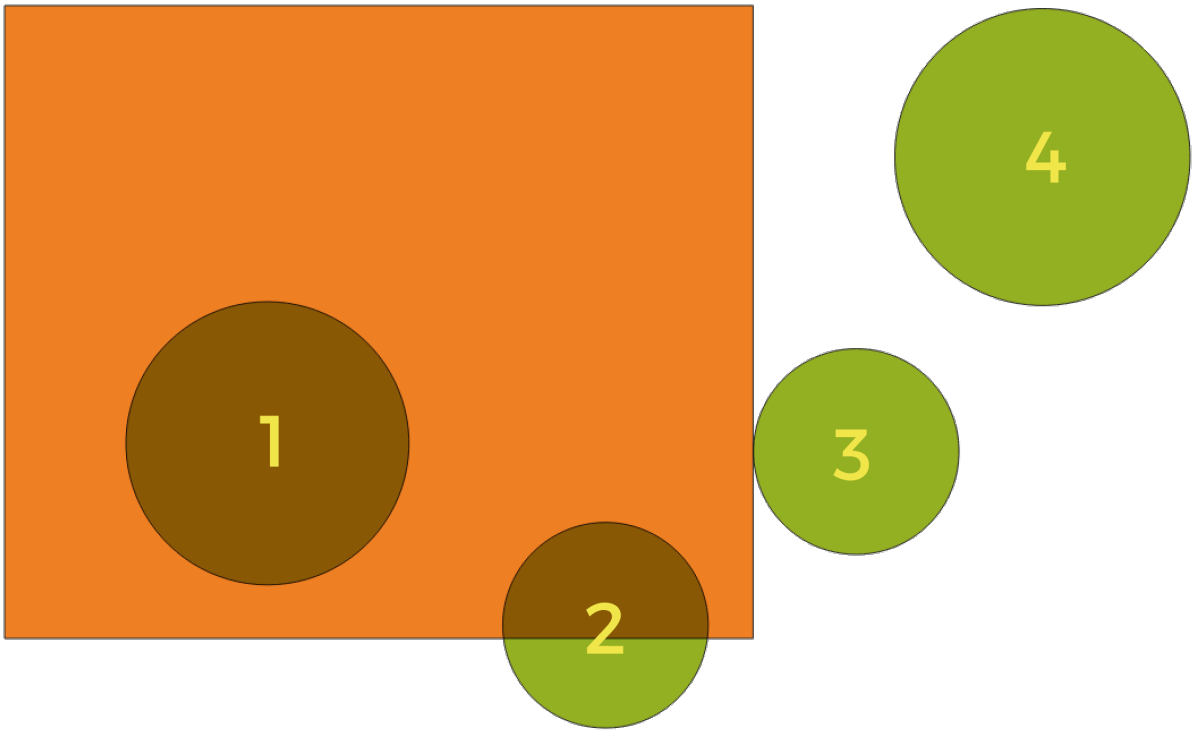


Fig. 28.132: Looking for spatial relations between layers

Folosind figura de mai sus, căutăm cercurile verzi comparându-le spațial cu entitatea dreptunghiulară portocalie. Predicatele geometrice disponibile sunt:

Intersect
: Testează dacă o geometrie o intersectează pe alta. Returnează 1 (adevărat) dacă geometriile se intersectează spațial (împart orice porțiune de spațiu - se suprapun sau se ating) și 0 dacă nu se intersectează. În imaginea de mai sus, vor fi returnate cercurile 1, 2 și 3.

Contain
: Returnează 1 (adevărat) dacă și numai dacă niciun punct al lui b nu se află în exteriorul lui a, iar cel puțin un punct din interiorul lui b se află în interiorul lui a. În imagine, nu este returnat niciun cerc, ar fi însă cazul dreptunghiului dacă l-ați căuta invers, întrucât conține cercul 1 complet. Acest predicat geometric este opusul lui *se află în interior*.

Disjoint
: Returnează 1 (adevărat) dacă geometriile nu împart nicio porțiune de spațiu (fără suprapunere, fără atingere). Doar cercul 4 este returnat.

Equal
Returnează 1 (adevărat) dacă și numai dacă geometriile sunt exact aceleași. Niciun cerc nu va fi returnat.

Touch
Testează dacă o geometrie o atinge pe alta. Returnează 1 (adevărat) dacă geometriile au cel puțin un punct în comun, iar interiorul lor nu se intersectează. Doar cercul 3 este returnat.

Overlap
Testează dacă o geometrie se suprapune peste alta. Returnează 1 (adevărat) dacă geometriile împart spațiul, sunt de aceeași dimensiune, dar nu sunt complet conținute una în cealaltă. Doar cercul 2 este returnat.

Are within
Testează dacă o geometrie este conținută în alta. Returnează 1 (adevărat) dacă geometria a se află complet în interiorul geometriei b. Doar cercul 1 este returnat.

Cross
Returnează 1 (adevărat) dacă geometriile furnizate au unele, dar nu toate, punctele interioare, în comun, iar traversarea reală este de o dimensiune mai mică decât cea mai mare geometrie furnizată. De exemplu, o linie care traversează un poligon se va încrucișa ca o linie (adevărat). Două linii care se încrucișează se vor încrucișa ca punct (adevărat). Două poligoane se încrucișează ca poligon (fals). În imagine, niciun cerc nu va fi returnat.

Parametri

Etichetă	Nume	Tipul	Descriere
Extract features from	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Where the features (geometric predicate)	`PREDICATE`	[enumeration] [list] Implicit: [0]	Type of spatial relation the input feature should have with an intersect feature so that it could be selected. One or more of: • 0 — intersectează • 1 — conţin • 2 — disjunge • 3 — egal • 4 — atinge • 5 — se suprapun • 6 — se află în interior • 7 — traversează If more than one condition is chosen, at least one of them (OR operation) has to be met for a feature to be extracted.
By comparing to the features from	`INTERSECT`	[vector: oricare]	Intersection vector layer
Extracted (location)	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the features that have the chosen spatial relationship(s) with one or more features in the comparison layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted (location)	`OUTPUT`	[la fel ca intrarea]	Vector layer with features from the input layer that have the chosen spatial relationship(s) with one or more features in the comparison layer.

Cod Python

Algorithm ID: `qgis:extractbylocation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract within distance

Creates a new vector layer that only contains matching features from an input layer. Features are copied wherever they are within the specified maximum distance from the features in an additional reference layer.

Vezi și:

Select within distance, *Extract by location*

Parametri

Etichetă	Nume	Tipul	Descriere
Extract features from	`INPUT`	[vector: oricare]	Input vector layer to copy features from
By comparing to the features from	`REFERENCE`	[vector: oricare]	Vector layer whose features closeness is used
Where the features are within	`DISTANCE`	[număr] Implicit: 100	The maximum distance around reference features to select input features within
Modify current selection by	`METHOD`	[enumerare] Implicit: 0	How the selection of the algorithm should be managed. One of: • 0 — creează o nouă selecție • 1 — adăugare la selecția curentă • 2 — selectare în cadrul selecției curente • 3 — eliminare din selecția curentă

continues on next page

Tabelul 28.190 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Extracted (location)	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the features that are within the set distance from reference features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted (location)	`OUTPUT`	[la fel ca intrarea]	Vector layer with features from the input layer matching the condition of distance from reference features

Cod Python

ID Algorithm: `nativ:extractwithindistance`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Filtrează după tipul geometriei

Filtrează entitățile după tipul geometriei lor. Entitățile primite vor fi direcționate către diferite ieșiri, în funcție de tipul geometriei: punct, linie sau poligon.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul de evaluat

Rezultat

Etichetă	Nume	Tipul	Descriere
Entitățile punctuale Opţional	`POINTS`	[vector: punct]	Strat cu puncte
Entitățile liniare Opţional	`LINES`	[vector: linie]	Strat cu linii
Entitățile poligonale Opţional	`POLYGONS`	[vector: poligon]	Strat cu poligoane
Funcții fără geometrie Opţional	`NO_GEOMETRY`	[tabelă]	Stratul vectorial fără geometrie

Cod Python

ID Algorithm: `nativ:filterbygeometry`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random extract

Takes a vector layer and generates a new one that contains only a subset of the features in the input layer.

The subset is defined randomly, based on feature IDs, using a percentage or count value to define the total number of features in the subset.

Vezi și:

Random selection

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Source vector layer to select the features from
Method	`METHOD`	[enumerare] Implicit: 0	Random selection methods. One of: • 0 — Numărul de entități selectate • 1 — Procentajul entităților selectate
Number/percentage of selected features	`NUMBER`	[număr] Implicit: 10	Number or percentage of features to select

continues on next page

Tabelul 28.191 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Extracted (random)	`OUTPUT`	[vector: oricare] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the randomly selected features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted (random)	`OUTPUT`	[la fel ca intrarea]	Vector layer containing randomly selected features from the input layer

Cod Python

Algorithm ID: `qgis:randomextract`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random extract within subsets

Takes a vector layer and generates a new one that contains only a subset of the features in the input layer.

The subset is defined randomly, based on feature IDs, using a percentage or count value to define the total number of features in the subset. The percentage/count value is not applied to the whole layer, but instead to each category. Categories are defined according to a given attribute.

Vezi și:

Random selection within subsets

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer to select the features from
ID field	`FIELD`	[tablefield: any]	Category of the source vector layer to select the features from
Method	`METHOD`	[enumerare] Implicit: 0	Random selection method. One of: • 0 — Numărul de entități selectate • 1 — Procentajul entităților selectate

continues on next page

Tabelul 28.192 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Number/percentage of selected features	`NUMBER`	[număr] Implicit: 10	Number or percentage of features to select
Extracted (random stratified)	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer for the randomly selected features. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Extracted (random stratified)	`OUTPUT`	[la fel ca intrarea]	Vector layer containing randomly selected features from the input layer

Cod Python

Algorithm ID: `qgis:randomextractwithinsubsets`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random selection

Takes a vector layer and selects a subset of its features. No new layer is generated by this algorithm.

The subset is defined randomly, based on feature IDs, using a percentage or count value to define the total number of features in the subset.

Default menu: *Vector ► Research Tools*

Vezi și:

Random extract

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer for the selection
Method	`METHOD`	[enumerare] Implicit: 0	Random selection method. One of: • 0 — Numărul de entități selectate • 1 — Procentajul entităților selectate
Number/percentage of selected features	`NUMBER`	[număr] Implicit: 10	Number or percentage of features to select

Rezultat

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[la fel ca intrarea]	The input layer with features selected

Cod Python

Algorithm ID: `qgis:randomselection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Random selection within subsets

Takes a vector layer and selects a subset of its features. No new layer is generated by this algorithm.

The subset is defined randomly, based on feature IDs, using a percentage or count value to define the total number of features in the subset.

The percentage/count value is not applied to the whole layer, but instead to each category.

Categories are defined according to a given attribute, which is also specified as an input parameter for the algorithm.

No new outputs are created.

Default menu: *Vector ► Research Tools*

Vezi și:

Random extract within subsets

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer to select features in
ID field	`FIELD`	[tablefield: any]	Category of the input layer to select the features from
Method	`METHOD`	[enumerare] Implicit: 0	Random selection method. One of: • 0 — Numărul de entități selectate • 1 — Procentajul entităților selectate
Number/percentage of selected features	`NUMBER`	[număr] Implicit: 10	Number or percentage of features to select

Rezultat

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[la fel ca intrarea]	The input layer with features selected

Cod Python

Algorithm ID: `qgis:randomselectionwithinsubsets`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Select by attribute

Creates a selection in a vector layer.

The criteria for selecting features is based on the values of an attribute from the input layer.

Vezi și:

Extract by attribute

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Vector layer to select features in
Selection attribute	`FIELD`	[tablefield: any]	Filtering field of the layer

continues on next page

Tabelul 28.193 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Operator	`OPERATOR`	[enumerare] Implicit: 0	Many different operators are available: • 0 — = • 1 — ≠ • 2 — >= • 3 — >= • 4 — < • 5 — <= • 6 — începe cu • 7 — conţine • 8 — is null • 9 — is not null • 10 — nu conține
Value Opţional	`VALUE`	[string]	Value to be evaluated
Modify current selection by	`METHOD`	[enumerare] Implicit: 0	How the selection of the algorithm should be managed. One of: • 0 — creează o nouă selecție • 1 — adăugare la selecția curentă • 2 — eliminare din selecția curentă • 3 — selectare în cadrul selecției curente

Rezultat

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[la fel ca intrarea]	The input layer with features selected

Cod Python

Algorithm ID: `qgis:selectbyattribute`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Select by expression

Creates a selection in a vector layer.

The criteria for selecting features is based on a QGIS expression. For more information about expressions see the *Expresii*.

Vezi și:

Extract by expression

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Expression	`EXPRESSION`	[expresie]	Expression to filter the input layer
Modify current selection by	`METHOD`	[enumerare] Implicit: 0	How the selection of the algorithm should be managed. One of: • 0 — creează o nouă selecție • 1 — adăugare la selecția curentă • 2 — eliminare din selecția curentă • 3 — selectare în cadrul selecției curente

Rezultat

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[la fel ca intrarea]	The input layer with features selected

Cod Python

Algorithm ID: `qgis:selectbyexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Selectare după locație

Creates a selection in a vector layer.

The criteria for selecting features is based on the spatial relationship between each feature and the features in an additional layer.

Default menu: *Vector ► Research Tools*

Vezi și:

Extract by location, *Select within distance*

Exploring spatial relations

Geometric predicates are boolean functions used to determine the spatial relation a feature has with another by comparing whether and how their geometries share a portion of space.

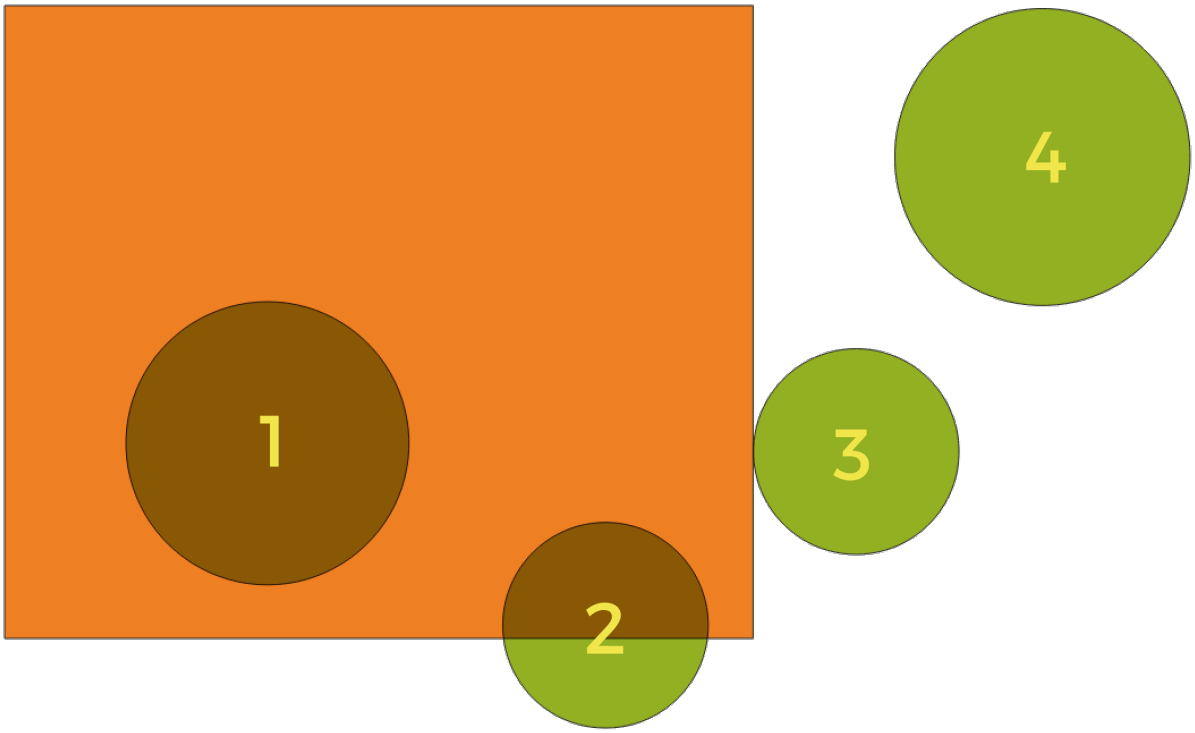


Fig. 28.133: Looking for spatial relations between layers

Folosind figura de mai sus, căutăm cercurile verzi comparându-le spațial cu entitatea dreptunghiulară portocalie. Predicatele geometrice disponibile sunt:

Intersect
: Testează dacă o geometrie o intersectează pe alta. Returnează 1 (adevărat) dacă geometriile se intersectează spațial (împart orice porțiune de spațiu - se suprapun sau se ating) și 0 dacă nu se intersectează. În imaginea de mai sus, vor fi returnate cercurile 1, 2 și 3.

Contain
: Returnează 1 (adevărat) dacă și numai dacă niciun punct al lui b nu se află în exteriorul lui a, iar cel puțin un punct din interiorul lui b se află în interiorul lui a. În imagine, nu este returnat niciun cerc, ar fi însă cazul dreptunghiului dacă l-ați căuta invers, întrucât conține cercul 1 complet. Acest predicat geometric este opusul lui *se află în interior*.

Disjoint
: Returnează 1 (adevărat) dacă geometriile nu împart nicio porțiune de spațiu (fără suprapunere, fără atingere). Doar cercul 4 este returnat.

Equal
: Returnează 1 (adevărat) dacă și numai dacă geometriile sunt exact aceleași. Niciun cerc nu va fi returnat.

Touch
: Testează dacă o geometrie o atinge pe alta. Returnează 1 (adevărat) dacă geometriile au cel puțin un punct în comun, iar interiorul lor nu se intersectează. Doar cercul 3 este returnat.

Overlap
: Testează dacă o geometrie se suprapune peste alta. Returnează 1 (adevărat) dacă geometriile împart spațiul, sunt de aceeași dimensiune, dar nu sunt complet conținute una în cealaltă. Doar cercul 2 este returnat.

Are within
Testează dacă o geometrie este conținută în alta. Returnează 1 (adevărat) dacă geometria a se află complet în interiorul geometriei b. Doar cercul 1 este returnat.

Cross
Returnează 1 (adevărat) dacă geometriile furnizate au unele, dar nu toate, punctele interioare, în comun, iar traversarea reală este de o dimensiune mai mică decât cea mai mare geometrie furnizată. De exemplu, o linie care traversează un poligon se va încrucișa ca o linie (adevărat). Două linii care se încrucișează se vor încrucișa ca punct (adevărat). Două poligoane se încrucișează ca poligon (fals). În imagine, niciun cerc nu va fi returnat.

Parametri

Etichetă	Nume	Tipul	Descriere
Select features from	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Where the features (geometric predicate)	`PREDICATE`	[enumeration] [list] Implicit: [0]	Type of spatial relation the input feature should have with an intersect feature so that it could be selected. One or more of: • 0 — intersectează • 1 — conţin • 2 — disjunge • 3 — egal • 4 — atinge • 5 — se suprapun • 6 — se află în interior • 7 — traversează If more than one condition is chosen, at least one of them (OR operation) has to be met for a feature to be extracted.
By comparing to the features from	`INTERSECT`	[vector: oricare]	Intersection vector layer
Modify current selection by	`METHOD`	[enumerare] Implicit: 0	How the selection of the algorithm should be managed. One of: • 0 — creează o nouă selecție • 1 — adăugare la selecția curentă • 2 — selectare în cadrul selecției curente • 3 — eliminare din selecția curentă

Rezultat

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[la fel ca intrarea]	The input layer with features selected

Cod Python

Algorithm ID: `qgis:selectbylocation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Select within distance

creates a selection in a vector layer. Features are selected wherever they are within the specified maximum distance from the features in an additional reference layer.

Vezi și:

Extract within distance, *Selectare după locație*

Parametri

Etichetă	Nume	Tipul	Descriere
Select features from	`INPUT`	[vector: oricare]	Input vector layer to select features from
By comparing to the features from	`REFERENCE`	[vector: oricare]	Vector layer whose features closeness is used
Where the features are within	`DISTANCE`	[număr] Implicit: 100	The maximum distance around reference features to select input features
Modify current selection by	`METHOD`	[enumerare] Implicit: 0	How the selection of the algorithm should be managed. One of: • 0 — creează o nouă selecție • 1 — adăugare la selecția curentă • 2 — selectare în cadrul selecției curente • 3 — eliminare din selecția curentă

Rezultat

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[la fel ca intrarea]	The input layer with features selected

Cod Python

ID Algorithm: `nativ:selectwithindistance`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.25 Tabela Vectorială

Add autoincremental field

Adds a new integer field to a vector layer, with a sequential value for each feature.

This field can be used as a unique ID for features in the layer. The new attribute is not added to the input layer but a new layer is generated instead.

The initial starting value for the incremental series can be specified. Optionally, the incremental series can be based on grouping fields and a sort order for features can also be specified.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer.
Field name	`FIELD_NAME`	[string] Implicit: «AUTO»	Name of the field with autoincremental values
Start values at Opţional	`START`	[număr] Implicit: 0	Choose the initial number of the incremental count
Modulus value Opţional	`MODULUS`	[număr] Implicit: 0	Specifying an optional modulus value will restart the count to START whenever the field value reaches the modulus value. `0` means no restart.
Group values by Opţional	`GROUP_FIELDS`	[tablefield: any] [list]	Select grouping field(s): instead of a single count run for the whole layer, a separate count is processed for each value returned by the combination of these fields.
Sort expression Opţional	`SORT_EXPRESSIO`	[expresie]	Use an expression to sort the features in the layer either globally or if set, based on group fields.
Sort ascending	`SORT_ASCENDING`	[boolean] Implicit: True	When a `sort expression` is set, use this option to control the order in which features are assigned values.
Sort nulls first	`SORT_NULLS_FIR`	[boolean] Implicit: *False*	When a `sort expression` is set, use this option to set whether *Null* values are counted first or last.

continues on next page

Tabelul 28.196 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Incremented	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer with the auto increment field. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Incremented	`OUTPUT`	[la fel ca intrarea]	Vector layer with auto incremental field

Cod Python

ID Algorithm: `nativ:addautoincrementalfield`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Add field to attributes table

Adds a new field to a vector layer.

The name and characteristics of the attribute are defined as parameters.

The new attribute is not added to the input layer but a new layer is generated instead.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul de intrare
Field name	`FIELD_NAME`	[string]	Name of the new field
Field type	`FIELD_TYPE`	[enumerare] Implicit: 0	Type of the new field. You can choose between: • 0 — Integer • 1 — Zecimal • 2 — Șir
Field length	`FIELD_LENGTH`	[număr] Implicit: 10	Length of the field

continues on next page

Tabelul 28.197 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Field precision	`FIELD_PRECISIO`	[număr] Implicit: 0	Precision of the field. Useful with Float field type.
Field alias `NEW in 3.32` Opţional	`FIELD_ALIAS`	[string]	Set a name to use as alias for the field. Not supported by all format types.
Field comment `NEW in 3.32` Opţional	`FIELD_COMMENT`	[string]	Store a comment describing the field. Not supported by all format types.
Added	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Added	`OUTPUT`	[la fel ca intrarea]	Vector layer with new field added

Cod Python

ID Algorithm: `nativ:addfieldtoattributestable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Add unique value index field

Takes a vector layer and an attribute and adds a new numeric field.

Values in this field correspond to values in the specified attribute, so features with the same value for the attribute will have the same value in the new numeric field.

This creates a numeric equivalent of the specified attribute, which defines the same classes.

The new attribute is not added to the input layer but a new layer is generated instead.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul de intrare.
Class field	`FIELD`	[tablefield: any]	Features that have the same value for this field will get the same index.
Output field name	`FIELD_NAME`	[string] Implicit: «NUM_FIELD»	Name of the new field containing the indexes.
Layer with index field	`OUTPUT`	[vector: oricare] Implicit: `[Creare Strat temporar]`	Vector layer with the numeric field containing indexes. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.
Class summary	`SUMMARY_OUTPUT`	[tabelă] Implicit: `[Omitere rezultat]`	Specify the table to contain the summary of the class field mapped to the corresponding unique value. One of: • Skip Output • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Layer with index field	`OUTPUT`	[la fel ca intrarea]	Vector layer with the numeric field containing indexes.
Class summary	`SUMMARY_OUTPUT`	[tabelă]	Table with summary of the class field mapped to the corresponding unique value.

Cod Python

ID Algorithm: `nativ:adduniquevalueindexfield`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Add X/Y fields to layer

Adds X and Y (or latitude/longitude) fields to a point layer. The X/Y fields can be calculated in a different CRS to the layer (e.g. creating latitude/longitude fields for a layer in a projected CRS).

Allows *features in-place modification* of point features

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: punct]	Stratul de intrare.
Coordinate system	`CRS`	[crs] Implicit: „EPSG:4326”	Coordinate reference system to use for the generated x and y fields.
Field prefix Opţional	`PREFIX`	[string]	Prefix to add to the new field names to avoid name collisions with fields in the input layer.
Added fields	`OUTPUT`	[vector: punct] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Added fields	`OUTPUT`	[vector: punct]	The output layer - identical to the input layer but with two new double fields, `x` and `y`.

Cod Python

ID Algorithm: `nativ:addxyfieldstolayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Advanced Python field calculator

Adds a new attribute to a vector layer, with values resulting from applying an expression to each feature.

The expression is defined as a Python function.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Result field name	`FIELD_NAME`	[string] Implicit: «NewField»	Name of the new field
Field type	`FIELD_TYPE`	[enumerare] Implicit: 0	Type of the new field. One of: • 0 — Integer • 1 — Zecimal • 2 — Șir
Field length	`FIELD_LENGTH`	[număr] Implicit: 10	Length of the field
Field precision	`FIELD_PRECISIO`	[număr] Implicit: 3	Precision of the field. Useful with Float field type.
Global expression Opţional	`GLOBAL`	[string]	The code in the global expression section will be executed only once before the calculator starts iterating through all the features of the input layer. Therefore, this is the correct place to import necessary modules or to calculate variables that will be used in subsequent calculations.
Formula	`FORMULA`	[string]	The Python formula to evaluate. Example: To calculate the area of an input polygon layer you can add: `value = $geom.area()`
Calculated	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the vector layer with the new calculated field. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Calculated	`OUTPUT`	[la fel ca intrarea]	Vector layer with the new calculated field

Cod Python

Algorithm ID: `qgis:advancedpythonfieldcalculator`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Drop field(s)

Takes a vector layer and generates a new one that has the same features but without the selected columns.

Vezi și:

Retain fields

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer to drop field(s) from
Fields to drop	`COLUMN`	[tablefield: any] [list]	The field(s) to drop
Remaining fields	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer with the remaining fields. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Remaining fields	`OUTPUT`	[la fel ca intrarea]	Vector layer with the remaining fields

Cod Python

ID Algorithm: `nativ:deletecolumn`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Explode HStore Field

Creates a copy of the input layer and adds a new field for every unique key in the HStore field.

The expected field list is an optional comma separated list. If this list is specified, only these fields are added and the HStore field is updated. By default, all unique keys are added.

The PostgreSQL HStore is a simple key-value store used in PostgreSQL and GDAL (when reading an OSM file with the `other_tags` field.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
HStore field	`FIELD`	[tablefield: any]	The field(s) to drop
Expected list of fields separated by a comma Opţional	`EXPECTED_FIELD`	[string] Implicit: „	Comma-separated list of fields to extract. The HStore field will be updated by removing these keys.
Exploded	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output vector layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Exploded	`OUTPUT`	[la fel ca intrarea]	Output vector layer

Cod Python

ID Algorithm: `nativ:explodehstorefield`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extract binary field

Extracts contents from a binary field, saving them to individual files. Filenames can be generated using values taken from an attribute in the source table or based on a more complex expression.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Input vector layer containing the binary data
Binary field	`FIELD`	[tablefield: any]	Field containing the binary data
File name	`FILENAME`	[expresie]	Field or expression-based text to name each output file
Destination folder	`FOLDER`	[folder] Implicit: `[Salvare într-un fișier temporar]`	Folder in which to store the output files. One of: • Salvare Intr-un Director Temporar • Salvare în Director

Rezultat

Etichetă	Nume	Tipul	Descriere
Folder	`FOLDER`	[folder]	The folder that contains the output files.

Cod Python

ID Algorithm: `nativ:extractbinary`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Calculatorul de câmpuri

Opens the field calculator (see *Expresii*). You can use all the supported expressions and functions.

A new layer is created with the result of the expression.

The field calculator is very useful when used in *The model designer*.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The layer to calculate on
Output field name	`FIELD_NAME`	[string]	The name of the field for the results
Output field type	`FIELD_TYPE`	[enumerare] Implicit: 0	The type of the field. One of: • 0 — Float • 1 — Integer • 2 — Șir • 3 — Dată
Output field width	`FIELD_LENGTH`	[număr] Implicit: 10	The length of the result field (minimum 0)
Field precision	`FIELD_PRECISIO`	[număr] Implicit: 3	The precision of the result field (minimum 0, maximum 15)
Create new field	`NEW_FIELD`	[boolean] Implicit: True	Should the result field be a new field
Formula	`FORMULA`	[expresie]	The formula to use to calculate the result
Output file	`OUTPUT`	[vector: oricare] Implicit: `[Creare Strat temporar]`	Specification of the output layer. • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Calculated	`OUTPUT`	[vector: oricare]	Output layer with the calculated field values

Cod Python

ID Algorithm: `nativ:fieldcalculator`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Refactor fields

Allows editing the structure of the attribute table of a vector layer.

Fields can be modified in their type and name, using a fields mapping.

The original layer is not modified. A new layer is generated, which contains a modified attribute table, according to the provided fields mapping.

Notă: When using a template layer with *constraints* on fields, the information is displayed in the widget with a coloured background and tooltip. Treat this information as a hint during configuration. No constraints will be added on an output layer nor will they be checked or enforced by the algorithm.

The Refactor fields algorithm allows to:

- Change field names and types
- Add and remove fields
- Reorder fields
- Calculate new fields based on expressions
- Load field list from another layer

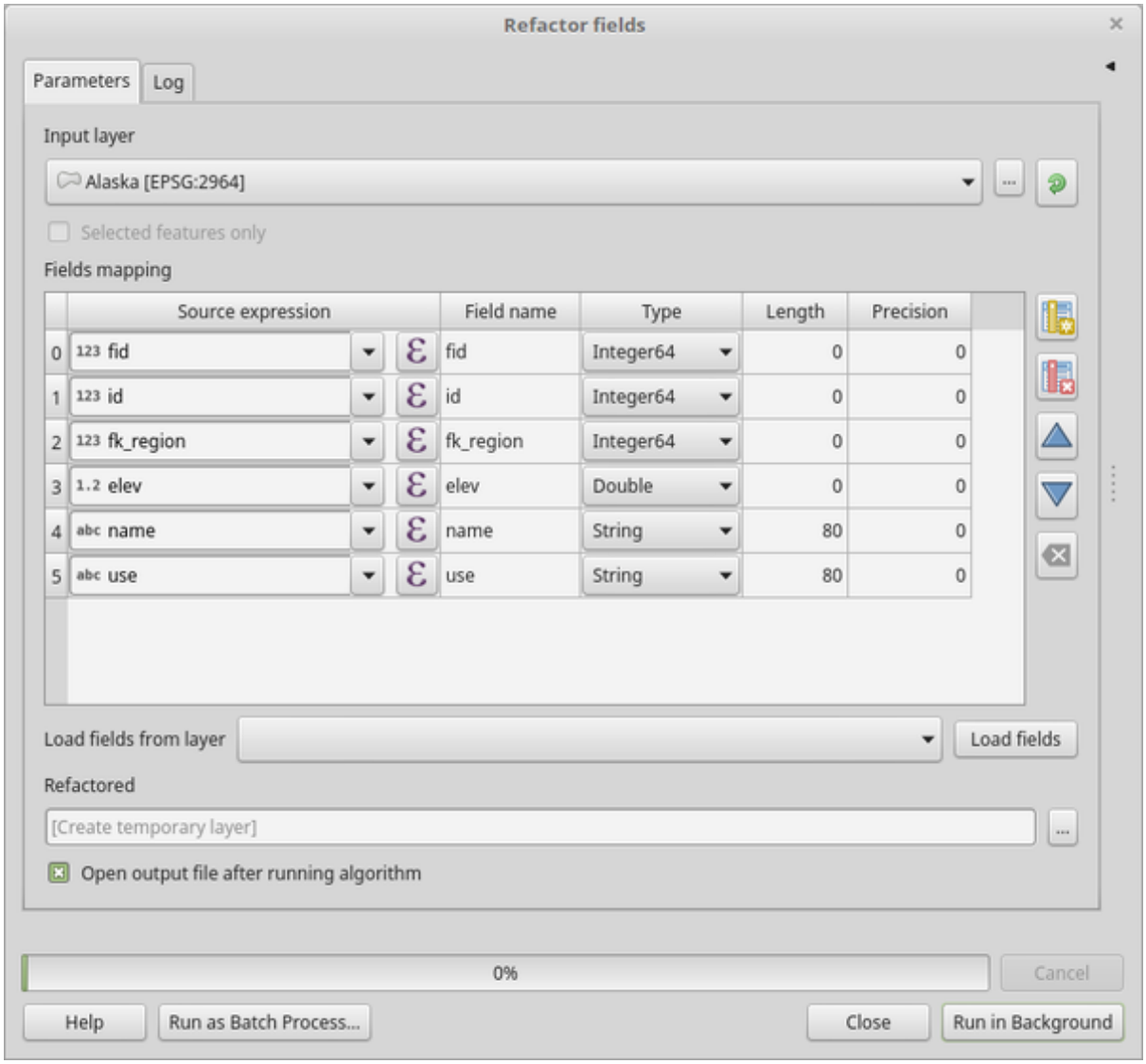


Fig. 28.134: Refactor fields dialog

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The layer to modify

continues on next page

Tabelul 28.203 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Fields mapping	`FIELDS_MAPPING`	[list]	List of output fields with their definitions. The embedded table lists all the fields of the source layer and allows you to edit them: • Click to create a new field. • Click to remove a field. • Use and to change the selected field order. • Click to reset to the default view. • Click to invert the selection in the fields list. For each of the fields you'd like to reuse, you need to fill the following options: ***Source expression*** **(`expression`) [expression]** Field or expression from the input layer. ***Field name*** **(`name`) [string]** Name of the field in the output layer. By default input field name is kept. ***Type*** **(`type`) [enumeration]** Data type of the output field. Available types depend on the output layer provider. ***Length*** **(`length`) [number]** Length of the output field. ***Precision*** **(`precision`) [number]** Precision of the output field. ***Constraints*** **(`constraints`) [string]** When using a template layer, indicates whether there are constraints applied to the template field. Hover over the cell to display the constraints. ***Field alias*** **(`field_alias`) [string]** **`NEW in 3.32`** Set a name to use as alias for the field. Not supported by all format types. Existing aliases are displayed and will be copied to the destination layer if supported. ***Field comment*** **(`field_comment`) [string]** **`NEW in 3.32`** Store a comment describing the field. Not supported by all format types. Existing comments are displayed and will be copied to the destination layer if supported. ***Load fields from template layer*** Allows to select a layer from the current project as a template and (with *Load fields*) fill the above „Fields mapping" options with its fields and their definitions.

continues on next page

Tabelul 28.203 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Refactored	`OUTPUT`	[vector: oricare] Implicit: `[Creare Strat temporar]`	Specification of the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Refactored	`OUTPUT`	[vector: oricare]	Output layer with refactored fields

Cod Python

ID Algorithm: `nativ:refactorfields`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rename field

Renames an existing field from a vector layer.

The original layer is not modified. A new layer is generated where the attribute table contains the renamed field.

Vezi și:

Refactor fields

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer
Field to rename	`FIELD`	[tablefield: any]	The field to be altered
New field name	`NEW_NAME`	[string]	The new field name

continues on next page

Tabelul 28.204 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Renamed	`OUTPUT`	[vector: same as input] Implicit: `[Creare Strat temporar]`	Specification of the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Renamed	`OUTPUT`	[vector: same as input]	Output layer with the renamed field

Cod Python

Algorithm ID: `qgis:renametablefield`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Retain fields

Takes a vector layer and generates a new one that retains only the selected fields. All other fields will be dropped.

Vezi și:

Drop field(s)

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer
Fields to retain	`FIELDS`	[tablefield: any] [list]	List of fields to keep in the layer
Retained fields	`OUTPUT`	[vector: same as input] Implicit: `[Creare Strat temporar]`	Specification of the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Retained fields	`OUTPUT`	[vector: same as input]	Output layer with the retained fields

Cod Python

ID Algorithm: `nativ:retainfields`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Text to float

Modifies the type of a given attribute in a vector layer, converting a text attribute containing numeric strings into a numeric attribute (e.g. «1» to `1.0`).

The algorithm creates a new vector layer so the source one is not modified.

If the conversion is not possible the selected column will have `NULL` values.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer.
Text attribute to convert to float	`FIELD`	[tablefield: string]	The string field for the input layer that is to be converted to a float field.
Float from text	`OUTPUT`	[la fel ca intrarea] Implicit: `[Creare Strat temporar]`	Specify the output layer. One of: • Create Temporary Layer (`TEMPORARY_OUTPUT`) • Salvare în Fișier… • Salvare în Geopackage… • Salvare în Tabela Bazei de Date… • Adăugare în Stratul… The file encoding can also be changed here.

Rezultat

Etichetă	Nume	Tipul	Descriere
Float from text	`OUTPUT`	[la fel ca intrarea]	Output vector layer with the string field converted into a float field

Cod Python

Algorithm ID: `qgis:texttofloat`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.1.26 Plăcuțe Vectoriale

Download vector tiles

`NEW in 3.32`

Downloads vector tiles of an input vector tile layer and saves them in a local vector tile file.

Parametri

Etichetă	Denumire	Tip	Descriere
Input layer	`INPUT`	[plăcuțe vectoriale]	A vector tile layer to extract some tiles from
Extindere	`EXTENT`	[extent]	Specify the spatial extent of the downloaded area. It will internally be extended to a multiple of the tile size. Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Maximum zoom level to download	`MAX_ZOOM`	[număr] Default: 10	Defines how far to zoom in and fetch data from the tiles
Tile limit	`TILE_LIMIT`	[număr] Default: 100	Maximum number of tiles to download, considering the zoom levels and the extent.
Output	`OUTPUT`	[plăcuțe vectoriale] Implicit: [Salvare într-un fișier temporar]	Specification of the output vector tile file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultate

Etichetă	Denumire	Tip	Descriere
Output	`OUTPUT`	[plăcuțe vectoriale]	A local vector tile file storing the downloaded tiles.

Cod Python

Algorithm ID: `native:downloadvectortiles`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când cursorul mouse-ului se deplasează deasupra denumirii unui algoritm din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și numele parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Scriere plăcuțe vectoriale (MBTiles)

Exportă unul sau mai multe straturi vectoriale sub formă de plăcuțe vectoriale, acestea având o dimensiune redusă și fiind optimizate pentru generarea rapidă a hărții.

MBTiles este o specificație pentru stocarea și transferul imediat al plăcuțelor de hartă în bazele de date SQLite. Fișierele MBTiles sunt cunoscute sub numele de seturi de plăcuțe.

Parametri

Etichetă	Denumire	Tip	Descriere
Straturi de intrare	`INPUT`	[vector: oricare] [list]	O listă a straturilor care vor fi combinate pentru a genera plăcuțe vectoriale
Nivelul minim de zoom	`MIN_ZOOM`	[număr] Implicit: 0	Cel mai scăzut nivel de zoom pentru care setul de plăcuțe oferă date. Setați între 0 și 24.
Nivelul maxim de zoom	`MAX_ZOOM`	[număr] Implicit: 3	Cel mai mare nivel de zoom pentru care setul de plăcuțe oferă date. Setați între 0 și 24.
Extindere Opţional	`EXTENT`	[extent] Implicit: Nesetat	Întinderea maximă a hărții generate. Limitele trebuie să definească o zonă acoperită de toate nivelurile de zoom.
Metadată: Denumire Opţional	`META_NAME`	[șir]	Numele setului de plăcuțe
Metadată: Descriere Opţional	`META_DESCRIPTI`	[șir]	Descrierea conținutului setului de plăcuțe
Metadată: Atribuire Opţional	`META_ATTRIBUTI`	[șir]	Un șir de atribuire, care oferă informații despre sursa datelor și/sau stilul hărții.
Metadată: Versiune Opţional	`META_VERSION`	[șir]	Versiunea setului de plăcuțe. Aceasta se referă la o revizuire a setului în sine, nu a specificației MBTiles.
Metadată: Tip Opţional	`META_TYPE`	[șir]	Tipul setului de plăcuțe. Valorile posibile sunt `suprapunere` sau `strat de bază`.
Metadată: Centru Opţional	`META_CENTER`	[șir]	Centrul implicit de vizualizare a hărții (șir de numere reprezentând longitudinea, latitudinea și nivelul de zoom, separate prin virgulă). Exemplu: `-122.1906,37.7599,11`
Destinație MBTiles	`OUTPUT`	[plăcuțe vectoriale] Implicit: [Salvare într-un fișier temporar]	Fișierul de ieșire al plăcuțelor vectoriale. Se alege dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultate

Etichetă	Denumire	Tip	Descriere
Destinație MBTiles	`OUTPUT`	[fișier]	Fișierul *.mbtiles* al plăcuțelor vectoriale de ieșire.

Cod Python

ID Algorithm: `nativ:writevectortiles_mbtiles`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când cursorul mouse-ului se deplasează deasupra denumirii unui algoritm din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și numele parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Scriere plăcuțe vectoriale (XYZ)

Exportă unul sau mai multe straturi vectoriale sub formă de plăcuțe vectoriale, acestea având o dimensiune redusă și fiind optimizate pentru generarea rapidă a hărții.

Parametri

Etichetă	Denumire	Tip	Descriere
Fișier șablon	`XYZ_TEMPLATE`	[șir] Implicit: «{z}/{x}/{y}.pbf»	Șablon de generare a adresei plăcuțelor vectoriale
Straturi de intrare	`INPUT`	[vector: oricare] [list]	O listă a straturilor care vor fi combinate pentru a genera plăcuțe vectoriale
Nivelul minim de zoom	`MIN_ZOOM`	[număr] Implicit: 0	Cel mai scăzut nivel de zoom pentru care setul de plăcuțe oferă date. Setați între 0 și 24.
Nivelul maxim de zoom	`MAX_ZOOM`	[număr] Implicit: 3	Cel mai mare nivel de zoom pentru care setul de plăcuțe oferă date. Setați între 0 și 24.
Extindere Opţional	`EXTENT`	[extent] Implicit: Nesetat	Întinderea maximă a hărții generate. Limitele trebuie să definească o zonă acoperită de toate nivelurile de zoom.
Director de ieșire	`OUTPUT_DIRECTO`	[folder] Implicit: [Salvare într-un dosar temporar]	Folderul plăcuțelor vectoriale de ieșire. Se alege dintre: • Salvare Intr-un Director Temporar • Salvare în Director

Rezultate

Etichetă	Denumire	Tip	Descriere
Director de ieșire	`OUTPUT_DIRECTO`	[folder]	Un folder care conține diverse subseturi de fișiere cu plăcuțe vectoriale (`.pbf`) stocate în subfolderele corespunzătoare nivelurilor de zoom.

Cod Python

ID Algorithm: `nativ:writevectortiles_xyz`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când cursorul mouse-ului se deplasează deasupra denumirii unui algoritm din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și numele parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.2 Furnizorul algoritmului GDAL

GDAL (Geospatial Data Abstraction Library) este o bibliotecă de conversie între diverse formate de date geospațiale raster și vectoriale. Algoritmii din Cadrul de Procesare sunt derivați din Programele raster GDAL și Programele vectoriale GDAL.

28.2.1 Analizarea rasterelor

Aspectul

Generates an aspect map from any GDAL-supported elevation raster. Aspect is the compass direction that a slope faces. The pixels will have a value from 0-360° measured in degrees from north indicating the azimuth. On the northern hemisphere, the north side of slopes is often shaded (small azimuth from 0°-90°), while the southern side receives more solar radiation (higher azimuth from 180°-270°).

This algorithm is derived from the GDAL DEM utility.

Default menu: *Raster* ► *Analysis*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Introduceți stratul raster al elevației
Numărul benzii	`BAND`	[raster band] Implicit: 1	Numărul benzii de utilizat ca elevație
Return trigonometric angle instead of azimuth	`TRIG_ANGLE`	[boolean] Implicit: *False*	Activating the trigonometric angle results in different categories: 0° (East), 90° (North), 180° (West), 270° (South).
Return 0 for flat instead of -9999	`ZERO_FLAT`	[boolean] Implicit: *False*	Activating this option will insert a 0-value for the value -9999 on flat areas.
Compute edges	`COMPUTE_EDGES`	[boolean] Implicit: *False*	Generates edges from the elevation raster
Use Zevenbergen&Thorı formula instead of the Horn's one	`ZEVENBERGEN`	[boolean] Implicit: *False*	Activates Zevenbergen&Thorne formula for smooth landscapes

continues on next page

Tabelul 28.206 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Aspect	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Aspect	`OUTPUT`	[raster]	Output raster with angle values in degrees

Cod Python

Algorithm ID: `gdal:aspect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Color relief

Generates a color relief map from any GDAL-supported elevation raster. Color reliefs can particularly be used to depict elevations. The Algorithm outputs a 4-band raster with values computed from the elevation and a text-based color configuration file. By default, the colors between the given elevation values are blended smoothly and the result is a nice colorized elevation raster.

This algorithm is derived from the GDAL DEM utility.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Introduceți stratul raster al elevației
Numărul benzii	`BAND`	[raster band] Implicit: 1	Numărul benzii de utilizat ca elevație
Compute edges	`COMPUTE_EDGES`	[boolean] Implicit: *False*	Generates edges from the elevation raster
Color configuration file	`COLOR_TABLE`	[file]	A text-based color configuration file
Matching mode	`MATCH_MODE`	[enumerare] Implicit: 2	One of: • 0 — Utilizează potrivirea strictă a culorilor • 1 — Utilizează cele mai apropiate cvadruple RGBA • 2 — Utilizează culori bine amestecate
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options
Color relief	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Color relief	`OUTPUT`	[raster]	A 4-band output raster

Cod Python

Algorithm ID: `gdal:colorrelief`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Fill NoData

Fill raster regions with NoData values by interpolation from edges. The values for the NoData regions are calculated by the surrounding pixel values using inverse distance weighting. After the interpolation a smoothing of the results takes place. Input can be any GDAL-supported raster layer. This algorithm is generally suitable for interpolating missing regions of fairly continuously varying rasters (such as elevation models for instance). It is also suitable for filling small holes and cracks in more irregularly varying images (like airphotos). It is generally not so great for interpolating a raster from sparse point data.

This algorithm is derived from the GDAL fillnodata utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Numărul benzii	`BAND`	[raster band] Implicit: 1	The band to operate on. NoData values must be represented by the value 0.
Maximum distance (in pixels) to search out for values to interpolate	`DISTANCE`	[număr] Implicit: 10	The number of pixels to search in all directions to find values to interpolate from
Number of smoothing iterations to run after the interpolation	`ITERATIONS`	[număr] Implicit: 0	The number of 3x3 filter passes to run (0 or more) to smoothen the results of the interpolation.
Do not use default validity mask for the input band	`NO_MASK`	[boolean] Implicit: *False*	Activates the user-defined validity mask
Validity mask	`MASK_LAYER`	[raster]	A raster layer that defines the areas to fill.

continues on next page

Tabelul 28.209 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Filled	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (`|`).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Filled	`OUTPUT`	[raster]	Output raster

Cod Python

Algorithm ID: `gdal:fillnodata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Grid (Data metrics)

Computes some data metrics using the specified window and output grid geometry.

This algorithm is derived from the GDAL grid utility.

Default menu: *Raster* ► *Analysis*

Vezi și:

GDAL grid tutorial

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Point layer	`INPUT`	[vector: punct]	Input point vector layer
Data metric to use	`METRIC`	[enumerare] Implicit: 0	One of: • 0 — Minimum, valoarea minimă, găsită în elipsa de căutare a nodului grilei • 1 — Maximum, valoarea maximă, găsită în elipsa de căutare a nodului grilei • 2 — Interval, diferența dintre valorile minime și maxime găsite în elipsa de căutare a nodului grilei • 3 — Total, numărul total al punctelor datelor găsite în elipsa de căutare a nodului grilei • 4 — Distanța medie, distanța medie dintre nodul grilei (centrul elipsei de căutare) și toate punctele datelor găsite în elipsa de căutare a nodului grilei • 5 — Average distance between points, an average distance between the data points found in grid node search ellipse. The distance between each pair of points within ellipse is calculated and average of all distances is set as a grid node value
The first radius of search ellipse	`RADIUS_1`	[număr] Implicit: 0.0	The first radius (X axis if rotation angle is 0) of the search ellipse
The second radius of search ellipse	`RADIUS_2`	[număr] Implicit: 0.0	The second radius (Y axis if rotation angle is 0) of the search ellipse
Angle of search ellipse rotation in degrees (counter clockwise)	`ANGLE`	[număr] Implicit: 0.0	Angle of ellipse rotation in degrees. Ellipse rotated counter clockwise.
Minimum number of data points to use	`MIN_POINTS`	[număr] Implicit: 0.0	Minimum number of data points to average. If less amount of points found the grid node considered empty and will be filled with NoData marker.

continues on next page

Tabelul 28.211 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
NoData	`NODATA`	[număr] Implicit: 0.0	NoData marker to fill empty points
Interpolated (data metrics)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer with interpolated values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Z value from field Opţional	`Z_FIELD`	[tablefield: numeric]	Field for the interpolation
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

continues on next page

Tabelul 28.212 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated (data metrics)	`OUTPUT`	[raster]	Output raster with interpolated values

Cod Python

Algorithm ID: `gdal:griddatametrics`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Grid (IDW with nearest neighbor searching)

Folosește metoda de îngrilare Distanța Inversă ridicată la o Putere în combinație cu metoda celui mai apropiat vecin. Este ideal atunci când se utilizează un număr maxim de puncte cu date.

This algorithm is derived from the GDAL grid utility.

Vezi și:

GDAL grid tutorial

Parametri

Etichetă	Nume	Tipul	Descriere
Point layer	`INPUT`	[vector: punct]	Input point vector layer
Weighting power	`POWER`	[număr] Implicit: 2.0	Weighting power
Smoothing	`SMOOTHING`	[număr] Implicit: 0.0	Smoothing parameter
The radius of the search circle	`RADIUS`	[număr] Implicit: 1.0	The radius of the search circle
Maximum number of data points to use	`MAX_POINTS`	[număr] Implicit: 12	Do not search for more points than this number.
Minimum number of data points to use	`MIN_POINTS`	[număr] Implicit: 0	Minimum number of data points to average. If less amount of points found the grid node considered empty and will be filled with NoData marker.
NoData	`NODATA`	[număr] Implicit: 0.0	NoData marker to fill empty points
Interpolated (IDW with NN search)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer with interpolated values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Z value from field Opţional	`Z_FIELD`	[tablefield: numeric]	Field for the interpolation
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).

continues on next page

Tabelul 28.214 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated (IDW with NN search)	`OUTPUT`	[raster]	Output raster with interpolated values

Cod Python

Algorithm ID: `gdal:gridinversedistancenearestneighbor`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Grilă (Distanța inversă ridicată la o putere)

Metoda de îngrilare Distanța Inversă ridicată la o Putere reprezintă un interpolator mediu ponderat.

You should supply the input arrays with the scattered data values including coordinates of every data point and output grid geometry. The function will compute interpolated value for the given position in output grid.

This algorithm is derived from the GDAL grid utility.

Default menu: *Raster* ► *Analysis*

Vezi și:

GDAL grid tutorial

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Point layer	`INPUT`	[vector: punct]	Input point vector layer
Weighting power	`POWER`	[număr] Implicit: 2.0	Weighting power
Smothing	`SMOOTHING`	[număr] Implicit: 0.0	Smoothing parameter
The first radius of search ellipse	`RADIUS_1`	[număr] Implicit: 0.0	The first radius (X axis if rotation angle is 0) of the search ellipse
The second radius of search ellipse	`RADIUS_2`	[număr] Implicit: 0.0	The second radius (Y axis if rotation angle is 0) of the search ellipse
Angle of search ellipse rotation in degrees (counter clockwise)	`ANGLE`	[număr] Implicit: 0.0	Angle of ellipse rotation in degrees. Ellipse rotated counter clockwise.
Maximum number of data points to use	`MAX_POINTS`	[număr] Implicit: 0	Do not search for more points than this number.
Minimum number of data points to use	`MIN_POINTS`	[număr] Implicit: 0	Minimum number of data points to average. If less amount of points found the grid node considered empty and will be filled with NoData marker.
NoData	`NODATA`	[număr] Implicit: 0.0	NoData marker to fill empty points
Interpolat (IDW)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer with interpolated values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Z value from field Opţional	`Z_FIELD`	[tablefield: numeric]	Field for the interpolation
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolat (IDW)	`OUTPUT`	[raster]	Output raster with interpolated values

Cod Python

Algorithm ID: `gdal:gridinversedistance`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Grid (Linear)

The Linear method perform linear interpolation by computing a Delaunay triangulation of the point cloud, finding in which triangle of the triangulation the point is, and by doing linear interpolation from its barycentric coordinates within the triangle. If the point is not in any triangle, depending on the radius, the algorithm will use the value of the nearest point or the NoData value.

This algorithm is derived from the GDAL grid utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Point layer	`INPUT`	[vector: punct]	Input point vector layer
Search distance	`RADIUS`	[număr] Implicit: -1.0	In case the point to be interpolated does not fit into a triangle of the Delaunay triangulation, use that maximum distance to search a nearest neighbour, or use NoData otherwise. If set to `-1`, the search distance is infinite. If set to `0`, NoData value will be used.
NoData	`NODATA`	[număr] Implicit: 0.0	NoData marker to fill empty points
Interpolated (Linear)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer with interpolated values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Z value from field Opţional	`Z_FIELD`	[tablefield: numeric]	Field for the interpolation
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated (Linear)	`OUTPUT`	[raster]	Output raster with interpolated values

Cod Python

Algorithm ID: `gdal:gridlinear`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Grid (Moving average)

The Moving Average is a simple data averaging algorithm. It uses a moving window of elliptic form to search values and averages all data points within the window. Search ellipse can be rotated by specified angle, the center of ellipse located at the grid node. Also the minimum number of data points to average can be set, if there are not enough points in window, the grid node considered empty and will be filled with specified NoData value.

This algorithm is derived from the GDAL grid utility.

Default menu: *Raster* ► *Analysis*

Vezi și:

GDAL grid tutorial

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Point layer	`INPUT`	[vector: punct]	Input point vector layer
The first radius of search ellipse	`RADIUS_1`	[număr] Implicit: 0.0	The first radius (X axis if rotation angle is 0) of the search ellipse
The second radius of search ellipse	`RADIUS_2`	[număr] Implicit: 0.0	The second radius (Y axis if rotation angle is 0) of the search ellipse
Angle of search ellipse rotation in degrees (counter clockwise)	`ANGLE`	[număr] Implicit: 0.0	Angle of ellipse rotation in degrees. Ellipse rotated counter clockwise.
Minimum number of data points to use	`MIN_POINTS`	[număr] Implicit: 0.0	Minimum number of data points to average. If less amount of points found the grid node considered empty and will be filled with NoData marker.
NoData	`NODATA`	[număr] Implicit: 0.0	NoData marker to fill empty points

continues on next page

Tabelul 28.219 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Interpolated (moving average)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Z value from field Opţional	`Z_FIELD`	[tablefield: numeric]	Field for the interpolation
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated (moving average)	`OUTPUT`	[raster]	Output raster with interpolated values

Cod Python

Algorithm ID: `gdal:gridaverage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Grid (Nearest neighbor)

The Nearest Neighbor method doesn't perform any interpolation or smoothing, it just takes the value of nearest point found in grid node search ellipse and returns it as a result. If there are no points found, the specified NoData value will be returned.

This algorithm is derived from the GDAL grid utility.

Default menu: *Raster ► Analysis*

Vezi și:

GDAL grid tutorial

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Point layer	`INPUT`	[vector: punct]	Input point vector layer
The first radius of search ellipse	`RADIUS_1`	[număr] Implicit: 0.0	The first radius (X axis if rotation angle is 0) of the search ellipse
The second radius of search ellipse	`RADIUS_2`	[număr] Implicit: 0.0	The second radius (Y axis if rotation angle is 0) of the search ellipse
Angle of search ellipse rotation in degrees (counter clockwise)	`ANGLE`	[număr] Implicit: 0.0	Angle of ellipse rotation in degrees. Ellipse rotated counter clockwise.
NoData	`NODATA`	[număr] Implicit: 0.0	NoData marker to fill empty points

continues on next page

Tabelul 28.221 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Interpolated (Nearest neighbour)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output raster layer with interpolated values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Z value from field Opţional	`Z_FIELD`	[tablefield: numeric]	Field for the interpolation
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Interpolated (Nearest neighbour)	`OUTPUT`	[raster]	Output raster with interpolated values

Cod Python

Algorithm ID: `gdal:gridnearestneighbor`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Hillshade

Outputs a raster with a nice shaded relief effect. It's very useful for visualizing the terrain. You can optionally specify the azimuth and altitude of the light source, a vertical exaggeration factor and a scaling factor to account for differences between vertical and horizontal units.

This algorithm is derived from the GDAL DEM utility .

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input Elevation raster layer
Numărul benzii	`BAND`	[raster band] Implicit: 1	Band containing the elevation information
Z factor (vertical exaggeration)	`Z_FACTOR`	[număr] Implicit: 1.0	The factor exaggerates the height of the output elevation raster
Scale (ratio of vert. units to horiz.)	`SCALE`	[număr] Implicit: 1.0	The ratio of vertical units to horizontal units
Azimuth of the light	`AZIMUTH`	[număr] Implicit: 315.0	Defines the azimuth of the light shining on the elevation raster in degrees. If it comes from the top of the raster the value is 0, if it comes from the east it is 90 a.s.o.
Altitude of the light	`ALTITUDE`	[număr] Implicit: 45.0	Defines the altitude of the light, in degrees. 90 if the light comes from above the elevation raster, 0 if it is raking light.
Compute edges	`COMPUTE_EDGES`	[boolean] Implicit: *False*	Generates edges from the elevation raster

continues on next page

Tabelul 28.223 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Use Zevenbergen&Thorı formula (instead of the Horn's one)	`ZEVENBERGEN`	[boolean] Implicit: *False*	Activates Zevenbergen&Thorne formula for smooth landscapes
Combined shading	`COMBINED`	[boolean] Implicit: *False*	
Multidirectional shading	`MULTIDIRECTION`	[boolean] Implicit: *False*	
Hillshade	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output raster layer with interpolated values. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Hillshade	`OUTPUT`	[raster]	Output raster with interpolated values

Cod Python

Algorithm ID: `gdal:hillshade`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Near black

Converts nearly black/white borders to black.

This algorithm will scan an image and try to set all pixels that are nearly or exactly black, white or one or more custom colors around the collar to black or white. This is often used to „fix up" lossy compressed airphotos so that color pixels can be treated as transparent when mosaicking.

This algorithm is derived from the GDAL nearblack utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input Elevation raster layer
How far from black (white)	`NEAR`	[număr] Implicit: 15	Select how far from black, white or custom colors the pixel values can be and still considered near black, white or custom color.
Search for nearly white pixels instead of nearly black	`WHITE`	[boolean] Implicit: *False*	Search for nearly white (255) pixels instead of nearly black pixels
Nearblack	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Nearblack	`OUTPUT`	[raster]	Output raster

Cod Python

Algorithm ID: `gdal:nearblack`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Proximity (raster distance)

Generates a raster proximity map indicating the distance from the center of each pixel to the center of the nearest pixel identified as a target pixel. Target pixels are those in the source raster for which the raster pixel value is in the set of target pixel values.

This algorithm is derived from the GDAL proximity utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input Elevation raster layer
Numărul benzii	`BAND`	[raster band] Implicit: 1	Band containing the elevation information
A list of pixel values in the source image to be considered target pixels Opţional	`VALUES`	[string] Implicit: „	A list of target pixel values in the source image to be considered target pixels. If not specified, all non-zero pixels will be considered target pixels.
Distance units	`UNITS`	[enumerare] Implicit: 1	Indicate whether distances generated should be in pixel or georeferenced coordinates. One of: • 0 — Coordonate georeferențiate • 1 — Coordonatele pixelului

continues on next page

Tabelul 28.227 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
The maximum distance to be generated Opţional	MAX_DISTANCE	[număr] Implicit: 0.0	The maximum distance to be generated. The NoData value will be used for pixels beyond this distance. If a NoData value is not provided, the output band will be queried for its NoData value. If the output band does not have a NoData value, then the value 65535 will be used. Distance is interpreted according to the value of *Distance units*.
Value to be applied to all pixels that are within the maxdist of target pixels Opţional	REPLACE	[număr] Implicit: 0.0	Specify a value to be applied to all pixels that are closer than the maximum distance from target pixels (including the target pixels) instead of a distance value.
NoData value to use for the destination proximity raster Opţional	NODATA	[număr] Implicit: 0.0	Specify the NoData value to use for the output raster
Proximity map	OUTPUT	[raster] Implicit: [Salvare într-un fişier temporar]	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	EXTRA	[string] Default: Nici o valoare	Add extra GDAL command line options

continues on next page

Tabelul 28.228 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Proximity map	`OUTPUT`	[raster]	Output raster

Cod Python

Algorithm ID: `gdal:proximity`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Roughness

Outputs a single-band raster with values computed from the elevation. Roughness is the degree of irregularity of the surface. It's calculated by the largest inter-cell difference of a central pixel and its surrounding cell. The determination of the roughness plays a role in the analysis of terrain elevation data, it's useful for calculations of the river morphology, in climatology and physical geography in general.

This algorithm is derived from the GDAL DEM utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Introduceți stratul raster al elevației
Numărul benzii	`BAND`	[raster band] Implicit: 1	Numărul benzii de utilizat ca elevație
Compute edges	`COMPUTE_EDGES`	[boolean] Implicit: *False*	Generates edges from the elevation raster
Roughness	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).

Rezultat

Etichetă	Nume	Tipul	Descriere
Roughness	`OUTPUT`	[raster]	Single-band output roughness raster. The value -9999 is used as NoData value.

Cod Python

Algorithm ID: `gdal:roughness`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Sieve

Removes raster polygons smaller than a provided threshold size (in pixels) and replaces them with the pixel value of the largest neighbour polygon. It is useful if you have a large amount of small areas on your raster map.

This algorithm is derived from the GDAL sieve utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Introduceți stratul raster al elevației
Threshold	`THRESHOLD`	[număr] Implicit: 10	Only raster polygons smaller than this size will be removed
Use 8-connectedness	`EIGHT_CONNECTE`	[boolean] Implicit: *False*	Use eight connectedness instead of four connectedness
Do not use the default validity mask for the input band	`NO_MASK`	[boolean] Implicit: *False*	
Validity mask Opţional	`MASK_LAYER`	[raster]	Validity mask to use instead of the default
Sieved	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Additional command-line parameters Opţional	EXTRA	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Sieved	OUTPUT	[raster]	Output raster layer.

Cod Python

Algorithm ID: `gdal:sieve`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Panta

Generates a slope map from any GDAL-supported elevation raster. Slope is the angle of inclination to the horizontal. You have the option of specifying the type of slope value you want: degrees or percent slope.

This algorithm is derived from the GDAL DEM utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	INPUT	[raster]	Input Elevation raster layer
Numărul benzii	BAND	[raster band] Implicit: 1	Band containing the elevation information
Ratio of vertical units to horizontal	SCALE	[număr] Implicit: 1.0	The ratio of vertical units to horizontal units
Slope expressed as percent (instead of degrees)	AS_PERCENT	[boolean] Implicit: *False*	Express slope as percent instead of degrees
Compute edges	COMPUTE_EDGES	[boolean] Implicit: *False*	Generates edges from the elevation raster

continues on next page

Tabelul 28.233 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Use Zevenbergen&Thorn formula (instead of the Horn's one)	`ZEVENBERGEN`	[boolean] Implicit: *False*	Activates Zevenbergen&Thorne formula for smooth landscapes
Slope	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opţiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Slope	`OUTPUT`	[raster]	Output raster

Cod Python

Algorithm ID: `gdal:slope`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Terrain Ruggedness Index (TRI)

Outputs a single-band raster with values computed from the elevation. TRI stands for Terrain Ruggedness Index, which is defined as the mean difference between a central pixel and its surrounding cells.

This algorithm is derived from the GDAL DEM utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Introduceți stratul raster al elevației
Numărul benzii	`BAND`	[raster band] Implicit: 1	Numărul benzii de utilizat ca elevație
Compute edges	`COMPUTE_EDGES`	[boolean] Implicit: *False*	Generates edges from the elevation raster
Terrain Ruggedness Index	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).

Rezultat

Etichetă	Nume	Tipul	Descriere
Terrain Ruggedness Index	`OUTPUT`	[raster]	Output ruggedness raster. The value -9999 is used as NoData value.

Cod Python

Algorithm ID: `gdal:triterrainruggednessindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Topographic Position Index (TPI)

Outputs a single-band raster with values computed from the elevation. TPI stands for Topographic Position Index, which is defined as the difference between a central pixel and the mean of its surrounding cells.

This algorithm is derived from the GDAL DEM utility.

Default menu: *Raster* ► *Analysis*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Introduceți stratul raster al elevației
Numărul benzii	`BAND`	[raster band] Implicit: 1	The number of the band to use for elevation values
Compute edges	`COMPUTE_EDGES`	[boolean] Implicit: *False*	Generates edges from the elevation raster
Terrain Ruggedness Index	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).

Rezultat

Etichetă	Nume	Tipul	Descriere
Terrain Ruggedness Index	`OUTPUT`	[raster]	Output raster.

Cod Python

Algorithm ID: `gdal:tpitopographicpositionindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.2.2 Raster conversion

gdal2xyz

Converts raster data to XYZ ASCII file format.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Raster layer to convert
Numărul benzii	`BAND`	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose the band you want to convert
Source NoData Opţional	`NODATA_INPUT`	[număr] Default: Nici o valoare	Input pixel value to treat as „NoData” (GDAL >= 3.7).
Destination NoData Opţional	`NODATA_OUTPUT`	[număr] Default: Nici o valoare	Assign specified „NoData” value to output (GDAL >= 3.7).
Do not output NoData values	`SKIP_NODATA`	[boolean] Implicit: False	Do not output „NoData” values (GDAL >= 3.3).
Output comma-separated values	`CSV`	[boolean] Implicit: False	Sets whether the output file should be of type comma-separated values (csv).
XYZ ASCII file	`OUTPUT`	[file] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
XYZ ASCII file	`INPUT`	[tabelă]	Table file containing the values exported from the raster band.

Cod Python

Algorithm ID: `gdal:gdal2xyz`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

PCT to RGB

Converts an 8 bit paletted image to a 24 bit RGB. It will convert a pseudocolor band from the input file to an RGB file of the desired format.

This algorithm is derived from the GDAL pct2rgb utility.

Default menu: *Raster* ► *Conversion*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input 8 bit raster image
Numărul benzii	`BAND`	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose the band you want to convert
Generate a RGBA file	`RGBA`	[boolean] Implicit: False	Sets whether the output file should be of type RGBA.
PCT to RGB	`OUTPUT`	[file] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output file. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
PCT to RGB	`OUTPUT`	[raster]	24 bit RGB raster image

Cod Python

Algorithm ID: `gdal:pcttorgb`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Polygonize (raster to vector)

Creates vector polygons for all connected regions of pixels in the raster sharing a common pixel value. Each polygon is created with an attribute indicating the pixel value of that polygon.

This algorithm is derived from the GDAL polygonize utility.

Default menu: *Raster* ► *Conversion*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Numărul benzii	`BAND`	[raster band] Implicit: Prima bandă a stratului de intrare	If the raster is multiband, choose the band you want to use
Name of the field to create	`FIELD`	[string] Implicit: «DN»	Specify the field name for the attributes of the connected regions.
Use 8-connectedness	`EIGHT_CONNECTE`	[boolean] Implicit: False	If not set, raster cells must have a common border to be considered connected (*4-connected*). If set, touching raster cells are also considered connected (*8-connected*).
Vectorized	`OUTPUT`	[vector: poligon] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output (polygon) vector layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Vectorized	`OUTPUT`	[vector: poligon]	Output vector layer

Cod Python

Algorithm ID: `gdal:polygonize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rearrange bands

Creates a new raster using selected band(s) from a given raster layer. The algorithm also makes it possible to reorder the bands for the newly-created raster.

This algorithm is derived from the GDAL translate utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Selected band(s)	`BANDS`	[raster band] [list] Default: Nici o valoare	Ordered list of the bands to use to create the new raster
Converted	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 0	Defines the data type of the output raster file. Options: • 0 — Se Utilizează Tipul Datelor din Stratul de Intrare • 1 — Byte (Întreg, fără semn, opt biți (quint8)) • 2 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 3 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 6 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 7 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 8 — CInt16 (Complex Int16) • 9 — CInt32 (Complex Int32) • 10 — CFloat32 (Complex Float32) • 11 — CFloat64 (Complex Float64) • 12 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Converted	`OUTPUT`	[raster]	Output raster layer with rearranged bands.

Cod Python

Algorithm ID: `gdal:rearrange_bands`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

RGB to PCT

Converts a 24 bit RGB image into a 8 bit paletted. Computes an optimal pseudo-color table for the given RGB-image using a median cut algorithm on a downsampled RGB histogram. Then it converts the image into a pseudo-colored image using the color table. This conversion utilizes Floyd-Steinberg dithering (error diffusion) to maximize output image visual quality.

If you want to classify a raster map and want to reduce the number of classes it can be helpful to downsample your image with this algorithm before.

This algorithm is derived from the GDAL rgb2pct utility.

Default menu: *Raster* ► *Conversion*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input (RGB) raster layer
Number of colors	`NCOLORS`	[număr] Implicit: 2	The number of colors the resulting image will contain. A value from 2-256 is possible.
RGB to PCT	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specification of the output raster. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
RGB to PCT	`OUTPUT`	[raster]	Output raster layer.

Cod Python

Algorithm ID: `gdal:rgbtopct`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Translate (convert format)

Converts raster data between different formats.

This algorithm is derived from the GDAL translate utility.

Default menu: *Raster* ► *Conversion*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Override the projection of the output file Opţional	`TARGET_CRS`	[crs]	Specify a projection for the output file
Assign a specified NoData value to output bands Opţional	`NODATA`	[număr] Implicit: Nesetat	Defines the value to use for NoData in the output raster
Copy all subdatasets of this file to individual output files	`COPY_SUBDATASE`	[boolean] Implicit: False	Create individual files for subdatasets
Converted	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output (translated) raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options
Tipul datelor de ieşire	`DATA_TYPE`	[enumerare] Implicit: 0	Defines the data type of the output raster file. Options: • 0 — Se Utilizează Tipul Datelor din Stratul de Intrare • 1 — Byte (Întreg, fără semn, opt biți (quint8)) • 2 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 3 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 6 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 7 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 8 — CInt16 (Complex Int16) • 9 — CInt32 (Complex Int32) • 10 — CFloat32 (Complex Float32) • 11 — CFloat64 (Complex Float64) • 12 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

Rezultat

Etichetă	Nume	Tipul	Descriere
Converted	`OUTPUT`	[raster]	Output (translated) raster layer.

Cod Python

Algorithm ID: `gdal:translate`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.2.3 Raster extraction

Clip raster by extent

Clips any GDAL-supported raster file to a given extent.

This algorithm is derived from the GDAL translate utility.

Default menu: *Raster* ► *Extraction*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	The input raster

continues on next page

Tabelul 28.245 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Clipping extent	`EXTENT`	[extent]	Extent that should be used for the output raster. Only pixels within the specified bounding box will be included in the output. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Override the projection for the output file	`OVERCRS`	[boolean] Implicit: *False*	If checked, the output file is assigned the input layer CRS.
Assign a specified NoData value to output bands Opţional	`NODATA`	[număr] Default: Nici o valoare	Defines a value that should be inserted for the NoData values in the output raster
Clipped (extent)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Tipul datelor de ieșire	DATA_TYPE	[enumerare] Implicit: 0	Defines the format of the output raster file. Opţiuni: • 0 — Se Utilizează Tipul Datelor din Stratul de Intrare • 1 — Byte (Întreg, fără semn, opt biți (quint8)) • 2 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 3 — UInt16 (Număr întreg, fără semn, șaisprezece biți (quint16)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 6 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 7 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 8 — CInt16 (Complex Int16) • 9 — CInt32 (Complex Int32) • 10 — CFloat32 (Complex Float32) • 11 — CFloat64 (Complex Float64) • 12 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)
Additional command-line parameters Opţional	EXTRA	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Clipped (extent)	`OUTPUT`	[raster]	Output raster layer clipped by the given extent

Cod Python

Algorithm ID: `gdal:cliprasterbyextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Clip raster by mask layer

Clips any GDAL-supported raster by a vector mask layer.

This algorithm is derived from the GDAL warp utility.

Default menu: *Raster* ► *Extraction*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	The input raster
Mask layer	`MASK`	[vector: poligon]	Vector mask for clipping the raster
CRS-ul sursei	`SOURCE_CRS`	[crs]	Set the coordinate reference to use for the input raster
CRS-ul destinației	`TARGET_CRS`	[crs]	Set the coordinate reference to use for the mask layer

continues on next page

Tabelul 28.246 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Target extent Opţional	TARGET_EXTENT	[extent]	Extent of the output file to be created Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Assign a specified NoData value to output bands Opţional	NODATA	[număr] Default: Nici o valoare	Defines a value that should be inserted for the NoData values in the output raster
Create an output alpha band	ALPHA_BAND	[boolean] Implicit: *False*	Creates an alpha band for the result. The alpha band then includes the transparency values of the pixels.
Match the extent of the clipped raster to the extent of the mask layer	CROP_TO_CUTLIN	[boolean] Implicit: True	Applies the vector layer extent to the output raster if checked.
Keep resolution of input raster	KEEP_RESOLUTIO	[boolean] Implicit: *False*	The resolution of the output raster will not be changed
Set output file resolution	SET_RESOLUTION	[boolean] Implicit: *False*	Shall the output resolution (cell size) be specified
X Resolution to output bands Opţional	X_RESOLUTION	[număr] Default: Nici o valoare	The width of the cells in the output raster
Y Resolution to output band Opţional	Y_RESOLUTION	[număr] Default: Nici o valoare	The height of the cells in the output raster
Use multithreaded warping implementation	MULTITHREADING	[boolean] Implicit: *False*	Two threads will be used to process chunks of image and perform input/output operation simultaneously. Note that computation is not multithreaded itself.

continues on next page

Tabelul 28.246 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Clipped (mask)	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 0	Defines the format of the output raster file. Opţiuni: • 0 — Se Utilizează Tipul Datelor din Stratul de Intrare • 1 — Byte (Întreg, fără semn, opt biți (quint8)) • 2 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 3 — UInt16 (Număr întreg, fără semn, șaisprezece biți (quint16)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 6 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 7 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 8 — CInt16 (Complex Int16) • 9 — CInt32 (Complex Int32) • 10 — CFloat32 (Complex Float32) • 11 — CFloat64 (Complex Float64) • 12 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Clipped (mask)	`OUTPUT`	[raster]	Output raster layer clipped by the vector layer

Cod Python

Algorithm ID: `gdal:cliprasterbymasklayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Curbe de nivel

Extracts contour lines from any GDAL-supported elevation raster.

This algorithm is derived from the GDAL contour utility.

Default menu: *Raster* ► *Extraction*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input raster
Numărul benzii	`BAND`	[raster band] Implicit: 1	Raster band to create the contours from
Interval between contour lines	`INTERVAL`	[număr] Implicit: 10.0	Defines the interval between the contour lines in the given units of the elevation raster (minimum value 0)
Attribute name (if not set, no elevation attribute is attached) Opţional	`FIELD_NAME`	[string] Implicit: «ELEV»	Provides a name for the attribute in which to put the elevation.
Offset from zero relative to which to interpret intervals Opţional	`OFFSET`	[număr] Implicit: 0.0	
Contours	`OUTPUT`	[vector: linie] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output vector layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Produce 3D vector	`CREATE_3D`	[boolean] Implicit: *False*	Forces production of 3D vectors instead of 2D. Includes elevation at every vertex.
Treat all raster values as valid	`IGNORE_NODATA`	[boolean] Implicit: *False*	Ignores any NoData values in the dataset.
Input pixel value to treat as „NoData” Opţional	`NODATA`	[număr] Default: Nici o valoare	Defines a value that should be inserted for the NoData values in the output raster
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options. Refer to the corresponding GDAL utility documentation.

Rezultat

Etichetă	Nume	Tipul	Descriere
Contours	`OUTPUT`	[vector: linie]	Output vector layer with contour lines

Cod Python

Algorithm ID: `gdal:contour`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Contour Polygons

Extracts contour polygons from any GDAL-supported elevation raster.

This algorithm is derived from the GDAL contour utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input raster
Numărul benzii	`BAND`	[raster band] Implicit: 1	Raster band to create the contours from

continues on next page

Tabelul 28.248 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Interval between contour lines	`INTERVAL`	[număr] Implicit: 10.0	Defines the interval between the contour lines in the given units of the elevation raster (minimum value 0)
Offset from zero relative to which to interpret intervals Opţional	`OFFSET`	[număr] Implicit: 0.0	
Attribute name for minimum elevation of contour polygon Opţional	`FIELD_NAME_MIN`	[string] Implicit: «ELEV_MIN»	Provides a name for the attribute in which to put the minimum elevation of contour polygon. If not provided no minimum elevation attribute is attached.
Attribute name for maximum elevation of contour polygon Opţional	`FIELD_NAME_MAX`	[string] Implicit: «ELEV_MAX»	Provides a name for the attribute in which to put the maximum elevation of contour polygon. If not provided no maximum elevation attribute is attached.
Contours	`OUTPUT`	[vector: poligon] Implicit: `[Salvare într-un fişier temporar]`	Specification of the output vector layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Produce 3D vector	`CREATE_3D`	[boolean] Implicit: *False*	Forces production of 3D vectors instead of 2D. Includes elevation at every vertex.
Treat all raster values as valid	`IGNORE_NODATA`	[boolean] Implicit: *False*	Ignores any NoData values in the dataset.
Input pixel value to treat as „NoData” Opţional	`NODATA`	[număr] Default: Nici o valoare	Defines a value that should be inserted for the NoData values in the output raster
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options. Refer to the corresponding GDAL utility documentation.

Rezultat

Etichetă	Nume	Tipul	Descriere
Contours	`OUTPUT`	[vector: poligon]	Output vector layer with contour polygons

Cod Python

Algorithm ID: `gdal:contour_polygon`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.2.4 Raster miscellaneous

Build overviews (pyramids)

To speed up rendering time of raster layers overviews (pyramids) can be created. Overviews are lower resolution copies of the data which QGIS uses depending of the level of zoom.

This algorithm is derived from the GDAL addo utility.

Default menu: *Raster* ► *Miscellaneous*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Stratul raster de intrare
Remove all existing overviews	`CLEAN`	[boolean] Implicit: *False*	Removes existing overviews from the raster. By default these are not removed.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Overview levels	`LEVELS`	[string] Implicit: «2 4 8 16»	Defines the number of overview levels calculated by the original resolution of the input raster layer. By default 4 levels will be taken into consideration.
Resampling method Opţional	`RESAMPLING`	[enumerare] Implicit: 0	Calculates the overviews with a defined resampling method. Possible resampling methods are: • 0 – Nearest Neighbour (`nearest`) • 1 – Average (`average`) • 2 – Gaussian (`gauss`) • 3 – Cubic (4x4 kernel) (`cubic`) • 4 – Cubic B-Spline (4x4 kernel) (`cubicspline`) • 5 – Lanczos (6x6 kernel) (`lanczos`) • 6 – Average MP (`average_mp`) • 7 – Average in Mag/Phase Space (`average_magphase`) • 8 – Mode (`mode`)
Overviews format Opţional	`FORMAT`	[enumerare] Implicit: 0	The overviews can be stored internally, or externally as GTiff or ERDAS Imagine file. By default the overviews are stored in the output raster. Possible formats methods are: • 0 – Internal (if possible) • 1 – External (GTiff .ovr) • 2 – External (ERDAS Imagine .aux)
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Pyramidized	`OUTPUT`	[raster]	Output raster layer with overviews

Cod Python

Algorithm ID: `gdal:overviews`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Build virtual raster

Builds a VRT (Virtual Dataset) that is a mosaic of the list of input GDAL-supported rasters. With a mosaic you can merge several raster files.

This algorithm is derived from the GDAL buildvrt utility.

Default menu: *Raster* ► *Miscellaneous*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	GDAL-supported raster layers.
Resolution	`RESOLUTION`	[enumerare] Implicit: 0	The output resolution of the mosaic. By default the average resolution of the raster files will be chosen. Opţiuni: • 0 — Medie (`average`) • 1 — Cel mai mare (`highest`) • 2 — Cel mai mic (`lowest`)
Place each input file into a separate band	`SEPARATE`	[boolean] Implicit: *False*	With «True» you can define that each raster file goes into a separated stacked band in the VRT band.
Allow projection difference	`PROJ_DIFFERENC`	[boolean] Implicit: *False*	Allows that the output bands have different projections derived from the projection of the input raster layers.
Virtual	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Add alpha mask band to VRT when source raster has none	`ADD_ALPHA`	[boolean] Implicit: *False*	Adds an alpha mask band to the VRT when the source raster has none.
Override projection for the output file Opţional	`ASSIGN_CRS`	[crs] Default: Nici o valoare	Overrides the projection for the output file. No reprojection is done.
Resampling algorithm	`RESAMPLING`	[enumerare] Implicit: 0	The resampling algorithm to use. Options: • 0 — Cel Mai Apropiat Vecin (`nearest`) • 1 — Bilinear (2x2 kernel) (`bilinear`) • 2 — Cubic (4x4 kernel) (`cubic`) • 3 — Cubic B-Spline (4x4 kernel) (`cubicspline`) • 4 — Lanczos (6x6 kernel) (`lanczos`) • 5 — Medie (`average`) • 6 — Mode (`mode`)
NoData value(s) for input bands (space separated) Opţional	`SRC_NODATA`	[string] Default: Nici o valoare	Space separated NoData value(s) for input band(s)
Additional command-line parameters	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Virtual	`OUTPUT`	[raster]	Stratul raster de ieșire

Cod Python

Algorithm ID: `gdal:buildvirtualraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

gdal2tiles

Generates a directory with small tiles and metadata, following the OSGeo Tile Map Service Specification. See also the OpenGIS Web Map Tile Service Implementation Standard. Simple web pages with viewers based on Google Maps, OpenLayers and Leaflet are generated as well. To explore your maps on-line in the web browser, you only need to upload the generated directory onto a web server.

This algorithm also creates the necessary metadata for Google Earth (KML SuperOverlay), in case the supplied map uses `EPSG:4326` projection.

ESRI world files and embedded georeferencing is used during tile generation, but you can publish a picture without proper georeferencing too.

This algorithm is derived from the GDAL gdal2tiles utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	GDAL-supported raster layer.
Tile cutting profile	`PROFILE`	[enumerare] Implicit: 0	One of: • 0 — Mercator (`mercator`) • 1 — Geodetic (`geodetic`) • 2 — Raster (`raster`)
Zoom levels to render Opţional	`ZOOM`	[string] Implicit: „	
Web viewer to generate	`VIEWER`	[enumerate] Implicit: 0	One of: • 0 — Toate (`all`) • 1 — GoogleMaps (`google`) • 2 — OpenLayers (`openlayers`) • 3 — Leaflet (`leaflet`) • 4 — Niciunul (`none`)
Title of the map Opţional	`TITLE`	[string] Implicit: „	
Copyright of the map	`COPYRIGHT`	[string] Implicit: „	
Director de ieșire	`OUTPUT`	[folder] Implicit: `[Salvare într-un fișier temporar]`	Specify the output folder for the tiles. One of: • Salvare Intr-un Director Temporar • Salvare în Director

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Resampling method	`RESAMPLING`	[enumerare] Implicit: 0	The resampling algorithm to use. Options: • 0 — Medie (`average`) • 1 — Nearest Neighbour (`near`) • 2 — Bilinear (2x2 kernel) (`bilinear`) • 3 — Cubic (4x4 kernel) (`cubic`) • 4 — Cubic B-Spline (4x4 kernel) (`cubicspline`) • 5 — Lanczos (6x6 kernel) (`lanczos`) • 6 — Antialias (`antialias`)
The spatial reference system used for the source input data Opţional	`SOURCE_CRS`	[crs] Default: Nici o valoare	
Transparency value to assign to the input data Opţional	`NODATA`	[număr] Implicit: 0.0	
URL address where the generated tiles are going to be published Opţional	`URL`	[string] Implicit: „	
Google Maps API key (http://code.google.c Opţional	`GOOGLE_KEY`	[string] Implicit: „	Your Google maps API key.
Bing Maps API key (https://www.bingm Opţional	`BING_KEY`	[string] Implicit: „	Your Bing maps API key.
Generate only missing files	`RESUME`	[boolean] Implicit: *False*	
Generate KML for Google Earth	`KML`	[boolean] Implicit: *False*	
Avoid automatic generation of KML files for EPSG:4326	`NO_KML`	[boolean] Implicit: *False*	

Rezultat

Etichetă	Nume	Tipul	Descriere
Director de ieșire	`OUTPUT`	[folder]	The output folder (for the tiles)

Cod Python

Algorithm ID: `gdal:gdal2tiles`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Merge

Merges raster files in a simple way. Here you can use a pseudocolor table from an input raster and define the output raster type. All the images must be in the same coordinate system.

This algorithm is derived from the GDAL merge utility.

Default menu: *Raster* ► *Miscellaneous*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Straturi de intrare	`INPUT`	[raster] [list]	Input raster layers
Grab pseudocolor table from first layer	`PCT`	[boolean] Implicit: *False*	The pseudocolor table from the first layer will be used for the coloring
Place each input file into a separate band	`SEPARATE`	[boolean] Implicit: *False*	Place each input file into a separate band

continues on next page

Tabelul 28.252 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tipul datelor de ieșire	`DATA_TYPE`	[enumerare] Implicit: 5	Defines the format of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)
Merged	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Input pixel value to treat as „NoData” Opţional	NODATA_INPUT	[număr] Default: Nici o valoare	Ignores pixels from files being merged in with this pixel value
Assign specified „NoData” value to output Opţional	NODATA_OUTPUT	[număr] Default: Nici o valoare	Assigns the specified NoData value to output bands.
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters	EXTRA	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Merged	OUTPUT	[raster]	Stratul raster de ieșire

Cod Python

Algorithm ID: `gdal:merge`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Pansharpening

Performs a pan-sharpening operation. It can create a „classic” output dataset (such as GeoTIFF), or a VRT dataset describing the pan-sharpening operation.

See GDAL Pansharpen.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Spectral dataset	`SPECTRAL`	[raster]	Input (spectral) raster layer
Panchromatic dataset	`PANCHROMATIC`	[raster]	Input (panchromatic) raster layer
Rezultat	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specify the output (sharpened) raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Resampling algorithm	`RESAMPLING`	[enumerare] Implicit: 2	The resampling algorithm to use. Options: • 0 — Cel Mai Apropiat Vecin (`nearest`) • 1 — Bilinear (2x2 kernel) (`bilinear`) • 2 — Cubic (4x4 kernel) (`cubic`) • 3 — Cubic B-Spline (4x4 kernel) (`cubicspline`) • 4 — Lanczos (6x6 kernel) (`lanczos`) • 5 — Medie (`average`)
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Rezultat	`OUTPUT`	[raster]	Output (sharpened) raster layer

Cod Python

Algorithm ID: `gdal:pansharp`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Calculator raster

Command line raster calculator with numpy syntax. Use any basic arithmetic supported by numpy arrays, such as +, -, *, and / along with logical operators, such as >. Note that all input rasters must have the same dimensions, but no projection checking is performed.

See the GDAL Raster Calculator utility docs.

Vezi și:

Calculator raster, *Raster calculator (virtual)*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Stratul de intrare A	`INPUT_A`	[raster]	First input raster layer (mandatory)
Number of raster band for A	`BAND_A`	[raster band]	Band for input layer A (mandatory)
Stratul de intrare B Opţional	`INPUT_B`	[raster] Default: Nici o valoare	Second input raster layer
Number of raster band for B Opţional	`BAND_B`	[raster band]	Band for input layer B
Stratul de intrare C Opţional	`INPUT_C`	[raster] Default: Nici o valoare	Third input raster layer
Number of raster band for C Opţional	`BAND_C`	[raster band]	Band for input layer C
Stratul de intrare D Opţional	`INPUT_D`	[raster] Default: Nici o valoare	Fourth input raster layer

continues on next page

Tabelul 28.254 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Number of raster band for D Opţional	BAND_D	[raster band]	Band for input layer D
Stratul de intrare E Opţional	INPUT_E	[raster] Default: Nici o valoare	Fifth input raster layer
Number of raster band for E Opţional	BAND_E	[raster band]	Band for input layer E
Stratul de intrare F Opţional	INPUT_F	[raster]	Sixth input raster layer
Number of raster band for F Opţional	BAND_F	[raster band] Default: Nici o valoare	Band for input layer F
Calculation in gdalnumeric syntax using +-/* or any numpy array functions (i.e. logical_and())	FORMULA	[string] Implicit: „	Formula de calcul. Exemple: • A*(A>0) — afișează valoarea rasterului A dacă valoarea lui A este mai mare decât 0. În caz contrar, rezultatul este 0. • A*(A>0 și A>B) — afișează valoarea lui A dacă acea valoare este mai mare decât 0 și mai mare decât valoarea lui B. În caz contrar, rezultatul este 0. • A*logical_or(A<=177, A>=185) — afișează valoarea lui A dacă A <= 177 sau A >= 185. În caz contrar, rezultatul este 0. • sqrt(A*A+B*B) — Afișează rădăcina pătrată a sumei dintre valorile lui A pătrat și a lui B pătrat.
Set output NoData value Opţional	NO_DATA	[număr] Default: Nici o valoare	Value to use for NoData
Handling of extent differences	EXTENT_OPT	[enumerare] Implicit: 0	Determines how to handle rasters with different extents. Only available with GDAL 3.3+. Supported options are: • 0 — Ignore • 1 — Fail • 2 — Union • 3 — Intersect

continues on next page

Tabelul 28.254 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Output extent Opţional	`INPUT`	[extent]	Custom extent of the output raster. Only available with GDAL 3.3+. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Output raster type	`RTYPE`	[enumerare] Implicit: 5	Defines the data type of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)
Calculated	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specify the output (calculated) raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	EXTRA	[string] Implicit: „	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Calculated	OUTPUT	[raster]	Output (calculated) raster layer

Cod Python

Algorithm ID: `gdal:rastercalculator`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Raster information

The gdalinfo program lists various information about a GDAL supported raster dataset.

This algorithm is derived from the GDAL info utility.

Default menu: *Raster ► Miscellaneous*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	INPUT	[raster]	Stratul raster de intrare

continues on next page

Tabelul 28.256 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Force computation of the actual min/max values for each band	`MIN_MAX`	[boolean] Implicit: *False*	Forces computation of the actual min/max values for each band in the dataset
Read and display image statistics (force computation if necessary)	`STATS`	[boolean] Implicit: *False*	Reads and displays image statistics. Forces computation if no statistics are stored in an image.
Suppress GCP info	`NO_GCP`	[boolean] Implicit: *False*	Suppresses ground control points list printing. It may be useful for datasets with huge amount of GCPs, such as L1B AVHRR or HDF4 MODIS which contain thousands of them.
Suppress metadata info	`NO_METADATA`	[boolean] Implicit: *False*	Suppresses metadata printing. Some datasets may contain a lot of metadata strings.
Layer information	`OUTPUT`	[html] Implicit: `[Salvare într-un fişier temporar]`	Specify the HTML file for output. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Layer information	`OUTPUT`	[html]	The HTML file containing information about the input raster layer

Cod Python

Algorithm ID: `gdal:gdalinfo`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Retile

Retiles a set of input tiles. All the input tiles must be georeferenced in the same coordinate system and have a matching number of bands. Optionally pyramid levels are generated.

This algorithm is derived from the GDAL Retile utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input files	`INPUT`	[raster] [list]	The input raster files
Tile width	`TILE_SIZE_X`	[număr] Implicit: 256	Width of the tiles in pixels (minimum 0)
Tile height	`TILE_SIZE_Y`	[număr] Implicit: 256	Height of the tiles in pixels (minimum 0)
Overlap in pixels between consecutive tiles	`OVERLAP`	[număr] Implicit: 0	
Number of pyramid levels to build	`LEVELS`	[număr] Implicit: 1	Minimum: 0
Director de ieşire	`OUTPUT`	[folder] Implicit: `[Salvare într-un fişier temporar]`	Specify the output folder for the tiles. One of: • Salvare Intr-un Director Temporar • Salvare în Director
CSV file containing the tile(s) georeferencing information	`OUTPUT_CSV`	[file] Implicit: `[Omitere rezultat]`	Specify the output file for the tiles. One of: • Skip Output • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Source coordinate reference system Opţional	SOURCE_CRS	[crs] Default: Nici o valoare	
Resampling method	RESAMPLING	[enumerare] Implicit: 0	The resampling algorithm to use. Options: • 0 — Cel Mai Apropiat Vecin (nearest) • 1 — Bilinear (2x2 kernel) (bilinear) • 2 — Cubic (4x4 kernel) (cubic) • 3 — Cubic B-Spline (4x4 kernel) (cubicspline) • 4 — Lanczos (6x6 kernel) (lanczos)
Column delimiter used in the CSV file Opţional	DELIMITER	[string] Implicit: «;»	Delimiter to use in the CSV file containing the tile(s) georeferencing information
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters Opţional	EXTRA	[string] Implicit: „	Add extra GDAL command line options

continues on next page

Tabelul 28.259 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Tipul datelor de ieșire	DATA_TYPE	[enumerare] Implicit: 5	Defines the format of the output raster file. Options: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)
Build only the pyramids	ONLY_PYRAMIDS	[boolean] Implicit: *False*	
Use separate directory for each tile row	DIR_FOR_ROW	[boolean] Implicit: *False*	

Rezultat

Etichetă	Nume	Tipul	Descriere
Director de ieșire	OUTPUT	[folder]	The output folder for the tiles.
CSV file containing the tile(s) georeferencing information	OUTPUT_CSV	[file]	The CSV file with georeferencing information for the tiles.

Cod Python

Algorithm ID: `gdal:retile`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Tile index

Builds a vector layer with a record for each input raster file, an attribute containing the filename, and a polygon geometry outlining the raster. This output is suitable for use with MapServer as a raster tileindex.

This algorithm is derived from the GDAL Tile Index utility.

Default menu: *Raster* ► *Miscellaneous*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Input files	`LAYERS`	[raster] [list]	The input raster files. Can be multiple files.
Field name to hold the file path to the indexed rasters	`PATH_FIELD_NAM` Optional	[string] Implicit: «location»	The output field name to hold the file path/location to the indexed rasters.
Store absolute path to the indexed rasters	`ABSOLUTE_PATH`	[boolean] Implicit: *False*	Set whether the absolute path to the raster files is stored in the tile index file. By default the raster filenames will be put in the file exactly as they are specified in the command.
Skip files with different projection reference	`PROJ_DIFFERENC`	[boolean] Implicit: *False*	Only files with same projection as files already inserted in the tile index will be inserted. Default does not check projection and accepts all inputs.
Tile index	`OUTPUT`	[vector: poligon] Implicit: `[Salvare într-un fișier temporar]`	Specify the polygon vector layer to write the index to. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Transform geometries to the given CRS Opţional	`TARGET_CRS`	[crs]	Geometries of input files will be transformed to the specified target coordinate reference system. Default creates simple rectangular polygons in the same coordinate reference system as the input rasters.
The name of the field to store the SRS of each tile Opţional	`CRS_FIELD_NAME`	[string]	The name of the field to store the SRS of each tile
The format in which the CRS of each tile must be written	`CRS_FORMAT`	[enumerare] Implicit: 0	Format for the CRS. One of: • 0 – Auto (`AUTO`) • 1 – Well-known text (`WKT`) • 2 – EPSG (`EPSG`) • 3 – Proj.4 (`PROJ`)

Rezultat

Etichetă	Nume	Tipul	Descriere
Tile index	`OUTPUT`	[vector: poligon]	The polygon vector layer with the tile index.

Cod Python

Algorithm ID: `gdal:tileindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Viewshed

Calculates a viewshed raster from an input raster DEM using method defined in Wang2000 for a user defined point.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Introduceți stratul raster al elevației

continues on next page

Tabelul 28.262 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Numărul benzii	`BAND`	[raster band] Implicit: 1	Numărul benzii de utilizat ca elevație
Observer location	`OBSERVER`	[point]	The location of the observer
Observer height	`OBSERVER_HEIGH`	[număr] Implicit: 1.0	The altitude of the observer, in the DEM units
Target height	`TARGET_HEIGHT`	[număr] Implicit: 1.0	The altitude of the target element, in the DEM units
Maximum distance from observer to compute visibility	`MAX_DISTANCE`	[număr] Implicit: 100.0	Maximum distance from observer to compute visibility, in the DEM units
Rezultat	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Output raster layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Additional command-line parameters	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Rezultat	`OUTPUT`	[raster]	The raster layer displaying the viewshed.

Cod Python

Algorithm ID: `gdal:viewshed`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.2.5 Proiecții raster

Asignare proiecție

Aplică un sistem de coordonate unui set de date raster.

Acest algoritm este derivat din utilitatrul de editare GDAL.

Default menu: *Raster* ► *Projections*

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT_LAYER`	[raster]	Stratul raster de intrare
Desired CRS	`CRS`	[crs]	Proiecția (CRS-ul) stratului de ieșire

Rezultat

Etichetă	Nume	Tipul	Descriere
Layer with projection	`OUTPUT`	[raster]	The output raster layer (with the new projection information)

Cod Python

Algorithm ID: `gdal:assignprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Extragere proiecție

Extracts the projection of a raster file and writes it into a *world* file with extension `.wld`.

This algorithm is derived from the GDAL srsinfo utility.

Default menu: *Raster* ► *Projections*

Parametri

Etichetă	Nume	Tipul	Descriere
Fișier de intrare	`INPUT_LAYER`	[raster]	Input raster The raster layer has to be file based, as the algorithm uses the path to the raster file as the location of the generated `.wld` file. Using a non-file raster layer will lead to an error.
Create also .prj file	`PRJ_FILE_CREAT`	[boolean] Implicit: False	If this is activated a `.prj` file containing the projection information is also created.

Rezultat

Etichetă	Nume	Tipul	Descriere
Fișier world	`WORLD_FILE`	[file]	Text file with extension `.wld` containing transformation parameters for the raster file.
ESRI Shapefile prj file	`PRJ_FILE`	[file]	Text file with `.prj` extension that describes the CRS. Will be `None` if *Create also .prj file* is False.

Cod Python

Algorithm ID: `gdal:extractprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Warp (reproiectare)

Reprojects a raster layer into another Coordinate Reference System (CRS). The output file resolution and the resampling method can be chosen.

This algorithm is derived from the GDAL warp utility.

Default menu: *Raster* ► *Projections*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[raster]	Input raster layer to reproject
CRS-ul sursei Opţional	`SOURCE_CRS`	[crs]	Defines the CRS of the input raster layer
CRS-ul destinației Opţional	`TARGET_CRS`	[crs] Default: `EPSG:4326`	The CRS of the output layer
Resampling method to use	`RESAMPLING`	[enumerare] Implicit: 0	Pixel value resampling method to use. Options: • 0 — Nearest Neighbour • 1 — Bilinear (2x2 kernel) • 2 — Cubic (4x4 kernel) • 3 — Cubic B-Spline (4x4 kernel) • 4 — Lanczos (6x6 kernel) • 5 — Medie • 6 — Mod • 7 — Maximum • 8 — Minimum • 9 — Median • 10 — First quartile (Q1) • 11 — Third quartile (Q3)
NoData value for output bands Opţional	`NODATA`	[număr] Default: Nici o valoare	Sets NoData value for output bands. If not provided, then NoData values will be copied from the source dataset.
Output file resolution in target georeferenced units Opţional	`TARGET_RESOLUT`	[număr] Default: Nici o valoare	Defines the output file resolution of reprojection result
Reproiectat	`OUTPUT`	[raster] Implicit: `[Salvare într-un fișier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Tipul datelor de ieșire	DATA_TYPE	[enumerare] Implicit: 0	Defines the format of the output raster file. Options: • 0 — Se Utilizează Tipul Datelor din Stratul de Intrare • 1 — Byte (Întreg, fără semn, opt biți (quint8)) • 2 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 3 — UInt16 (Număr întreg, fără semn, șaisprezece biți (quint16)) • 4 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 5 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 6 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 7 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 8 — CInt16 (Complex Int16) • 9 — CInt32 (Complex Int32) • 10 — CFloat32 (Complex Float32) • 11 — CFloat64 (Complex Float64) • 12 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)

continues on next page

Tabelul 28.265 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Georeferenced extents of output file to be created Opţional	`TARGET_EXTENT`	[extent]	Sets the georeferenced extent of the output file to be created (in the *Target CRS* by default. In the *CRS of the target raster extent*, if specified). Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
CRS of the target raster extent Opţional	`TARGET_EXTENT_`	[crs]	Specifies the CRS in which to interpret the coordinates given for the extent of the output file. This must not be confused with the target CRS of the output dataset. It is instead a convenience e.g. when knowing the output coordinates in a geodetic long/lat CRS, but wanting a result in a projected coordinate system.
Use multithreaded warping implementation	`MULTITHREADING`	[boolean] Implicit: False	Two threads will be used to process chunks of the image and perform input/output operations simultaneously. Note that the computation itself is not multithreaded.
Additional command-line parameters Opţional	`EXTRA`	[string] Default: Nici o valoare	Add extra GDAL command line options.

Rezultat

Etichetă	Nume	Tipul	Descriere
Reproiectat	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Reprojected output raster layer

Cod Python

Algorithm ID: `gdal:warpreproject`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.2.6 Vector conversion

Convert format

Converts any OGR-supported vector layer into another OGR-supported format.

This algorithm is derived from the ogr2ogr utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Convert all layers from dataset	`CONVERT_ALL_LA`	[boolean] Implicit: False	Converts the whole dataset. Supported output formats for this option are `GPKG` and `GML`.
Converted	`OUTPUT`	[la fel ca intrarea]	Specification of the output vector layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier… For `Save to File`, the output format has to be specified. All GDAL vector formats are supported. For `Save to a Temporary File` the QGIS default vector format will be used.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: «» (fără opțiuni adiționale)	Additional GDAL creation options.

Rezultat

Etichetă	Nume	Tipul	Descriere
Converted	`OUTPUT`	[la fel ca intrarea]	The output vector layer

Cod Python

Algorithm ID: `gdal:convertformat`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rasterize (overwrite with attribute)

Overwrites a raster layer with values from a vector layer. New values are assigned based on the attribute value of the overlapping vector feature.

This algorithm is derived from the GDAL rasterize utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Input raster layer	`INPUT_RASTER`	[raster]	Stratul raster de intrare
Field to use for a burn-in value Opţional	`FIELD`	[tablefield: numeric]	Defines the attribute field to use to set the pixels values

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Add burn in values to existing raster values	`ADD`	[boolean] Implicit: False	If False, pixels are assigned the selected field's value. If True, the selected field's value is added to the value of the input raster layer.
Additional command-line parameters Opţional	`EXTRA`	[string] Implicit: „	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Rasterized	`OUTPUT`	[raster]	The overwritten input raster layer

Cod Python

Algorithm ID: `gdal:rasterize_over`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rasterize (overwrite with fixed value)

Overwrites parts of a raster layer with a fixed value. The pixels to overwrite are chosen based on the supplied (overlapping) vector layer.

This algorithm is derived from the GDAL rasterize utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Input raster layer	`INPUT_RASTER`	[raster]	Stratul raster de intrare
A fixed value to burn	`BURN`	[număr] Implicit: 0.0	The value to burn

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Add burn in values to existing raster values	`ADD`	[boolean] Implicit: False	If False, pixels are assigned the fixed value. If True, the fixed value is added to the value of the input raster layer.
Additional command-line parameters Opţional	`EXTRA`	[string] Implicit: „	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Rasterized	`OUTPUT`	[raster]	The overwritten input raster layer

Cod Python

Algorithm ID: `gdal:rasterize_over_fixed_value`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Rasterize (vector to raster)

Converts vector geometries (points, lines and polygons) into a raster image.

This algorithm is derived from the GDAL rasterize utility.

Default menu: *Raster* ► *Conversion*

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Field to use for a burn-in value Opţional	`FIELD`	[tablefield: numeric]	Defines the attribute field from which the attributes for the pixels should be chosen
A fixed value to burn Opţional	`BURN`	[număr] Implicit: 0.0	A fixed value to burn into a band for all features.
Burn value extracted from the „Z” values of the feature Opţional	`USE_Z`	[boolean] Implicit: False	Indicates that a burn value should be extracted from the „Z” values of the feature. Works with points and lines (linear interpolation along each segment). For polygons, works properly only if they are flat (same Z value for all vertices)
Output raster size units	`UNITS`	[enumerare] Implicit: 0	Units to use when defining the output raster size/resolution. One of: • 0 — Pixeli • 1 — Unități georeferențiate
Width/Horizontal resolution	`WIDTH`	[număr] Implicit: 0.0	Sets the width (if size units is „Pixels”) or horizontal resolution (if size units is „Georeferenced units”) of the output raster. Minimum value: 0.0.

continues on next page

Tabelul 28.268 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Height/Vertical resolution	`HEIGHT`	[număr] Implicit: 0.0	Sets the height (if size units is „Pixels”) or vertical resolution (if size units is „Georeferenced units”) of the output raster.
Output extent Opţional	`EXTENT`	[extent]	Extent of the output raster layer. If the extent is not specified, the minimum extent that covers the selected reference layer(s) will be used. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Assign a specified NoData value to output bands Opţional	`NODATA`	[număr] Implicit: 0.0	Assigns a specified NoData value to output bands
Rasterized	`OUTPUT`	[raster] Implicit: `[Salvare într-un fişier temporar]`	Specificare strat raster de ieșire. Unul dintre: • Salvare într-un Fișier Temporar • Salvare în Fișier… For `Save to File`, the output format has to be specified. All GDAL raster formats are supported. For `Save to a Temporary File` the QGIS default raster format will be used.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: „	Pentru adăugarea uneia sau mai multor opțiuni de control la crearea rasterului (ex.: culori, dimensiunea blocului, compresia fișierului…). Pentru ușurință, vă puteți baza pe profiluri predefinite (consultați *Secțiunea opțiunilor driver-ului GDAL*). Procesarea în Serie și Designerul de Modele: separați opțiunile multiple cu un caracter pipe (|).
Tipul datelor de ieșire	DATA_TYPE	[enumerare] Implicit: 5	Defines the format of the output raster file. Opţiuni: • 0 — Byte (Întreg, fără semn, opt biți (quint8)) • 1 — Int16 (Întreg, cu semn, șaisprezece biți (qint16)) • 2 — UInt16 (Întreg, fără semn, șaisprezece biți (quint16)) • 3 — UInt32 (Treizeci și doi de biți în virgulă mobilă (quint32)) • 4 — Int32 (Treizeci și doi de biți în virgulă mobilă (qint32)) • 5 — Float32 (Treizeci și doi de biți în virgulă mobilă (float)) • 6 — Float64 (Șaizeci și patru de biți în virgulă mobilă (double)) • 7 — CInt16 (Complex Int16) • 8 — CInt32 (Complex Int32) • 9 — CFloat32 (Complex Float32) • 10 — CFloat64 (Complex Float64) • 11 — Int8 (Eight bit signed integer (qint8)) Opțiunile disponibile depind de versiunea GDAL construită cu ajutorul QGIS (consultați meniul *Help* ► *About*)
Pre-initialize the output image with value Opţional	INIT	[număr]	Pre-initializes the output image bands with this value. Not marked as the NoData value in the output file. The same value is used in all the bands.
Invert rasterization	INVERT	[boolean] Implicit: False	Burns the fixed burn value, or the burn value associated with the first feature into all parts of the image not inside the provided polygon.
Additional command-line parameters Opţional	EXTRA	[string] Implicit: „	Add extra GDAL command line options

Rezultat

Etichetă	Nume	Tipul	Descriere
Rasterized	`OUTPUT`	[raster]	Stratul raster de ieșire

Cod Python

Algorithm ID: `gdal:rasterize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.2.7 Vector geoprocessing

Buffer vectors

Create buffers around the features of a vector layer.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer
Geometry column name	`GEOMETRY`	[string] Implicit: «geometry»	The name of the input layer geometry column to use
Buffer distance	`DISTANCE`	[number] Default: 10.0	Minimum: 0.0
Dissolve by attribute Optional	`FIELD`	[tablefield: any] Default: None	Field to use for dissolving
Dissolve results	`DISSOLVE`	[boolean] Default: False	**If set, the result is dissolved.** If no field is set for dissolving, all the buffers are dissolved into one feature.
Produce one feature for each geometry in any kind of geometry collection in the source file	`EXPLODE_COLLEC`	[boolean] Default: False	

continues on next page

Tabelul 28.269 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Buffer	`OUTPUT`	[vector: polygon] Default: `[Save to temporary file]`	Specify the output buffer layer. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Etichetă	Nume	Tipul	Descriere
Additional creation options Optional	`OPTIONS`	[string] Default: «» (no additional options)	Additional GDAL creation options.

Outputs

Etichetă	Nume	Tipul	Descriere
Buffer	`OUTPUT`	[vector: polygon]	The output buffer layer

Python code

Algorithm ID: `gdal:buffervectors`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Clip vector by extent

Clips any OGR-supported vector file to a given extent.

This algorithm is derived from the GDAL ogr2ogr utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer

continues on next page

Tabelul 28.271 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Clip extent	`EXTENT`	[extent]	Defines the bounding box that should be used for the output vector file. It has to be defined in target CRS coordinates. Available methods are: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`
Clipped (extent)	`OUTPUT`	[same as input] Default: `[Save to temporary file]`	Specify the output (clipped) layer. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Etichetă	Nume	Tipul	Descriere
Additional creation options Optional	`OPTIONS`	[string] Default: «» (no additional options)	Additional GDAL creation options.

Outputs

Etichetă	Nume	Tipul	Descriere
Clipped (extent)	`OUTPUT`	[same as input]	The output (clipped) layer. The default format is „ESRI Shapefile”.

Python code

Algorithm ID: `gdal:clipvectorbyextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Clip vector by mask layer

Clips any OGR-supported vector layer by a mask polygon layer.

This algorithm is derived from the GDAL ogr2ogr utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input vector layer
Mask layer	`MASK`	[vector: polygon]	Layer to be used as clipping extent for the input vector layer.
Clipped (mask)	`OUTPUT`	[same as input] Default: `[Save to temporary file]`	The output (masked) layer. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Etichetă	Nume	Tipul	Descriere
Additional creation options Optional	`OPTIONS`	[string] Default: «» (no additional options)	Additional GDAL creation options.

Outputs

Etichetă	Nume	Tipul	Descriere
Clipped (mask)	`OUTPUT`	[same as input]	The output (masked) layer. The default format is „ESRI Shapefile”.

Python code

Algorithm ID: `gdal:clipvectorbypolygon`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Dissolve

Dissolve (combine) geometries that have the same value for a given attribute / field. The output geometries are multipart.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	The input layer to dissolve
Dissolve field Optional	`FIELD`	[tablefield: any]	The field of the input layer to use for dissolving
Geometry column name	`GEOMETRY`	[string] Implicit: «geometry»	The name of the input layer geometry column to use for dissolving.
Dissolved	`OUTPUT`	[same as input] Default: `[Save to temporary file]`	Specify the output layer. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Etichetă	Nume	Tipul	Descriere
Produce one feature for each geometry in any kind of geometry collection in the source file	`EXPLODE_COLLEC`	[boolean] Default: False	Produce one feature for each geometry in any kind of geometry collection in the source file
Keep input attributes	`KEEP_ATTRIBUTE`	[boolean] Default: False	Keep all attributes from the input layer
Count dissolved features	`COUNT_FEATURES`	[boolean] Default: False	Count the dissolved features and include it in the output layer.
Compute area and perimeter of dissolved features	`COMPUTE_AREA`	[boolean] Default: False	**Compute the area and perimeter of dissolved features and** include them in the output layer
Compute min/max/sum/mean for attribute	`COMPUTE_STATIS`	[boolean] Default: False	Calculate statistics (min, max, sum and mean) for the numeric attribute specified and include them in the output layer
Numeric attribute to calculate statistics on Optional	`STATISTICS_ATT`	[tablefield: numeric]	The numeric attribute to calculate statistics on
Additional creation options Optional	`OPTIONS`	[string] Default: «» (no additional options)	Additional GDAL creation options.

Outputs

Etichetă	Nume	Tipul	Descriere
Dissolved	`OUTPUT`	[same as input]	**The output multipart geometry layer (with dissolved** geometries)

Python code

Algorithm ID: `gdal:dissolve`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Offset curve

Offsets lines by a specified distance. Positive distances will offset lines to the left, and negative distances will offset them to the right.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: line]	Stratul liniar de intrare
Geometry column name	`GEOMETRY`	[string] Implicit: «geometry»	The name of the input layer geometry column to use
Offset distance (left-sided: positive, right-sided: negative)	`DISTANCE`	[number] Default: 10.0	
Offset curve	`OUTPUT`	[vector: line] Default: `[Save to temporary file]`	Specify the output line layer. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Etichetă	Nume	Tipul	Descriere
Additional creation options Optional	`OPTIONS`	[string] Default: «» (no additional options)	Additional GDAL creation options.

Outputs

Etichetă	Nume	Tipul	Descriere
Offset curve	`OUTPUT`	[vector: line]	The output offset curve layer

Python code

Algorithm ID: `gdal:offsetcurve`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

One side buffer

Creates a buffer on one side (right or left) of the lines in a line vector layer.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: line]	Stratul liniar de intrare
Geometry column name	`GEOMETRY`	[string] Implicit: «geometry»	The name of the input layer geometry column to use
Buffer distance	`DISTANCE`	[number] Default: 10.0	
Buffer side	`BUFFER_SIDE`	[enumeration] Default: 0	One of: • 0 — Dreapta • 1 — Stânga
Dissolve by attribute Optional	`FIELD`	[tablefield: any] Default: None	Field to use for dissolving

continues on next page

Tabelul 28.275 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Dissolve all results	`DISSOLVE`	[boolean] Default: False	**If set, the result is dissolved.** If no field is set for dissolving, all the buffers are dissolved into one feature.
Produce one feature for each geometry in any kind of geometry collection in the source file	`EXPLODE_COLLEC`	[boolean] Default: False	
One-sided buffer	`OUTPUT`	[vector: polygon] Default: `[Save to temporary file]`	Specify the output buffer layer. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Etichetă	Nume	Tipul	Descriere
Additional creation options Optional	`OPTIONS`	[string] Default: «» (no additional options)	Additional GDAL creation options.

Outputs

Etichetă	Nume	Tipul	Descriere
One-sided buffer	`OUTPUT`	[vector: polygon]	The output buffer layer

Python code

Algorithm ID: `gdal:onesidebuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

Points along lines

Generates a point on each line of a line vector layer at a distance from start. The distance is provided as a fraction of the line length.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: line]	Stratul liniar de intrare
Geometry column name	`GEOMETRY`	[string] Implicit: «geometry»	The name of the input layer geometry column to use
Distance from line start represented as a fraction of line length	`DISTANCE`	[number] Implicit: 0.5 (mijlocul liniei)	
Points along lines	`OUTPUT`	[vector: point] Default: `[Save to temporary file]`	Specify the output point layer. One of: • Save to a Temporary File • Save to File…

Advanced parameters

Etichetă	Nume	Tipul	Descriere
Additional creation options Optional	`OPTIONS`	[string] Default: «» (no additional options)	Additional GDAL creation options.

Outputs

Etichetă	Nume	Tipul	Descriere
Points along line	`OUTPUT`	[vector: point]	The output point layer

Python code

Algorithm ID: `gdal:pointsalonglines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The *algorithm id* is displayed when you hover over the algorithm in the Processing Toolbox. The *parameter dictionary* provides the parameter NAMEs and values. See *Utilizarea algoritmilor de procesare din consolă* for details on how to run processing algorithms from the Python console.

28.2.8 Vector miscellaneous

Build virtual vector

Creates a virtual vector layer that contains a set of vector layers. The output virtual vector layer will not be opened in the current project.

This algorithm is especially useful in case another algorithm needs multiple layers but accept only one `vrt` in which the layers are specified.

Parametri

Etichetă	Nume	Tipul	Descriere
Input datasources	`INPUT`	[vector: oricare] [list]	Select the vector layers you want to use to build the virtual vector
Create „unioned" VRT	`UNIONED`	[boolean] Implicit: False	Check if you want to unite all the vectors in a single `vrt` file
Virtual vector	`OUTPUT`	[la fel ca intrarea] Implicit: `[Salvare într-un fișier temporar]`	Specify the output layer containing only the duplicates. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier…

Rezultat

Etichetă	Nume	Tipul	Descriere
Virtual vector	`OUTPUT`	[vector: oricare]	The output virtual vector made from the chosen sources

Cod Python

Algorithm ID: `gdal:buildvirtualvector`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Executare SQL

Runs a simple or complex query with SQL syntax on the source layer. The result of the query will be added as a new layer.

This algorithm is derived from the GDAL ogr2ogr utility.

Parametri

Parametri de bază

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	OGR-supported input vector layer
SQL expression	`SQL`	[string]	Defines the SQL query, for example `SELECT * FROM my_table WHERE name is not null`.
SQL dialect	`DIALECT`	[enumerare] Implicit: 0	SQL dialect to use. One of: • 0 — Niciunul • 1 — OGR SQL • 2 — SQLite
SQL result	`OUTPUT`	[vector: oricare]	Specification of the output layer. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier… For `Save to File`, the output format has to be specified. All GDAL vector formats are supported. For `Save to a Temporary File` the default output vector layer format will be used.

Parametri avansați

Etichetă	Nume	Tipul	Descriere
Opțiuni suplimentare de creare Opţional	`OPTIONS`	[string] Implicit: «» (fără opțiuni adiționale)	Additional GDAL creation options.

Rezultat

Etichetă	Nume	Tipul	Descriere
SQL result	`OUTPUT`	[vector: oricare]	Stratul vectorial creat de interogare

Cod Python

Algorithm ID: `gdal:executesql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export to PostgreSQL (available connections)

Imports vector layers inside a PostgreSQL database on the basis of an available connection. The connection has to *be defined properly* beforehand. Be aware that the checkboxes «Save Username» and «Save Password» are activated. Then you can use the algorithm.

This algorithm is derived from the GDAL ogr2ogr utility.

Parametri

Etichetă	Nume	Tipul	Descriere
Database (connection name)	`DATABASE`	[string]	The PostgreSQL database to connect to
Strat de intrare	`INPUT`	[vector: oricare]	OGR-supported vector layer to export to the database
Shape encoding Opţional	`SHAPE_ENCODING`	[string] Implicit: „	Sets the encoding to apply to the data
Output geometry type	`GTYPE`	[enumerare] Implicit: 0	Defines the output geometry type. One of: • 0 — • 1 — NONE • 2 — GEOMETRY • 3 — POINT • 4 — LINESTRING • 5 — POLYGON • 6 — GEOMETRYCOLLECTION • 7 — MULTIPOINT • 8 — MULTIPOLYGON • 9 — MULTILINESTRING
Assign an output CRS Opţional	`A_SRS`	[crs] Default: Nici o valoare	Defines the output CRS of the database table
Reproject to this CRS on output Opţional	`T_SRS`	[crs] Default: Nici o valoare	Reprojects/transforms to this CRS on output

continues on next page

Tabelul 28.278 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Override source CRS Opţional	S_SRS	[crs] Default: Nici o valoare	Overrides the input layer CRS
Schema (schema name) Opţional	SCHEMA	[string] Implicit: «public»	Defines the schema for the database table
Table to export to (leave blank to use layer name) Opţional	TABLE	[string] Implicit: „	Defines a name for the table that will be imported into the database. By default the table name is the name of the input vector file.
Primary Key (new field) Opţional	PK	[string] Implicit: «id»	Defines which attribute field will be the primary key of the database table
Primary Key (existing field, used if the above option is left empty) Opţional	PRIMARY_KEY	[tablefield: any] Default: Nici o valoare	Defines which attribute field in the exported layer will be the primary key of the database table
Geometry column name Opţional	GEOCOLUMN	[string] Implicit: «geom»	Defines in which attribute field of the database there will be the geometry information
Vector dimensions Opţional	DIM	[enumerare] Implicit: 0 (2D)	Defines if the vector file to be imported has 2D or 3D data. One of: • 0 — 2 • 1 — 3
Distance tolerance for simplification Opţional	SIMPLIFY	[string] Implicit: „	Defines a distance tolerance for the simplification of the vector geometries to be imported. By default there is no simplification.
Maximum distance between 2 nodes (densification) Opţional	SEGMENTIZE	[string] Implicit: „	The maximum distance between two nodes. Used to create intermediate points. By default there is no densification.
Select features by extent (defined in input layer CRS) Opţional	SPAT	[extent] Default: Nici o valoare	You can select features from a given extent that will be in the output table. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as `xmin, xmax, ymin, ymax`

continues on next page

Tabelul 28.278 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Clip the input layer using the above (rectangle) extent Opţional	CLIP	[boolean] Implicit: False	The input layer will be clipped by the extent you defined before
Select features using a SQL „WHERE" statement (Ex: column="value") Opţional	WHERE	[string] Implicit: „	Defines with a SQL „WHERE" statement which features should be selected from the input layer
Group N features per transaction (Default: 2000) Opţional	GT	[string] Implicit: „	You can group the input features in transactions where N defines the size. By default N limits the transaction size to 20000 features.
Overwrite existing table Opţional	OVERWRITE	[boolean] Implicit: True	If there is a table with the same name in the database, and if this option is set to True, the table will be overwritten.
Append to existing table Opţional	APPEND	[boolean] Implicit: False	If checked / True the vector data will be appended to an existing table. New fields found in the input layer are ignored. By default a new table will be created.
Append and add new fields to existing table Opţional	ADDFIELDS	[boolean] Implicit: False	If activated the vector data will be appended to an existing table, there won't be a new table created. New fields found in input layer are added to the table. By default a new table will be created.
Do not launder columns/table names Opţional	LAUNDER	[boolean] Implicit: False	With this option checked you can prevent the default behaviour (converting column names to lowercase, removing spaces and other invalid characters).
Do not create Spatial Index Opţional	INDEX	[boolean] Implicit: False	Prevents a spatial index for the output table from being created. By default, a spatial index is added.
Continue after a failure, skipping the failed feature Opţional	SKIPFAILURES	[boolean] Implicit: False	
Promote to Multipart Opţional	PROMOTETOMULTI	[boolean] Implicit: True	Casts features geometry type to multipart in the output table
Keep width and precision of input attributes Opţional	PRECISION	[boolean] Implicit: True	Avoids modifying column attributes to comply with input data
Opțiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: «» (fără opțiuni adiționale)	Additional GDAL creation options.

Rezultat

This algorithm has no output.

Cod Python

Algorithm ID: `gdal:importvectorintopostgisdatabaseavailableconnections`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Export to PostgreSQL (new connection)

Imports vector layers inside a PostgreSQL database. A new connection to the PostGIS database must be created.

This algorithm is derived from the GDAL ogr2ogr utility.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	OGR-supported vector layer to export to the database
Shape encoding Opţional	`SHAPE_ENCODING`	[string] Implicit: „	Sets the encoding to apply to the data
Output geometry type	`GTYPE`	[enumerare] Implicit: 0	Defines the output geometry type. One of: • 0 — • 1 — NONE • 2 — GEOMETRY • 3 — POINT • 4 — LINESTRING • 5 — POLYGON • 6 — GEOMETRYCOLLECTION • 7 — MULTIPOINT • 8 — MULTIPOLYGON • 9 — MULTILINESTRING
Assign an output CRS Opţional	`A_SRS`	[crs] Default: Nici o valoare	Defines the output CRS of the database table
Reproject to this CRS on output Opţional	`T_SRS`	[crs] Default: Nici o valoare	Reprojects/transforms to this CRS on output
Override source CRS Opţional	`S_SRS`	[crs] Default: Nici o valoare	Overrides the input layer CRS
Host Opţional	`HOST`	[string] Implicit: «localhost»	Name of the database host
Port Opţional	`PORT`	[string] Implicit: «5432»	Port number the PostgreSQL database server listens on

continues on next page

Tabelul 28.279 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Username Opţional	USER	[string] Implicit: „	User name used to log in to the database
Database name Opţional	DBNAME	[string] Implicit: „	Name of the database
Password Opţional	PASSWORD	[string] Implicit: „	Password used with Username to connect to the database
Schema (schema name) Opţional	SCHEMA	[string] Implicit: «public»	Defines the schema for the database table
Table name, leave blank to use input name Opţional	TABLE	[string] Implicit: „	Defines a name for the table that will be imported into the database. By default the table name is the name of the input vector file.
Primary Key (new field) Opţional	PK	[string] Implicit: «id»	Defines which attribute field will be the primary key of the database table
Primary Key (existing field, used if the above option is left empty) Opţional	PRIMARY_KEY	[tablefield: any] Default: Nici o valoare	Defines which attribute field in the exported layer will be the primary key of the database table
Geometry column name Opţional	GEOCOLUMN	[string] Implicit: «geom»	Defines in which attribute field to store the geometry information
Vector dimensions Opţional	DIM	[enumerare] Implicit: 0 (2D)	Defines if the vector file to be imported has 2D or 3D data. One of: • 0 — 2D • 1 — 3D
Distance tolerance for simplification Opţional	SIMPLIFY	[string] Implicit: „	Defines a distance tolerance for the simplification of the vector geometries to be imported. By default there is no simplification.
Maximum distance between 2 nodes (densification) Opţional	SEGMENTIZE	[string] Implicit: „	The maximum distance between two nodes. Used to create intermediate points. By default there is no densification.

continues on next page

Tabelul 28.279 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Select features by extent (defined in input layer CRS) Opţional	SPAT	[extent] Default: Nici o valoare	You can select features from a given extent that will be in the output table. Metodele disponibile sunt: • Calculate from layer…: uses extent of a layer loaded in the current project • Calculate from layout map…: uses extent of a *layout map item* in the active project • Calculate from bookmark…: uses extent of a saved *bookmark* • Use map canvas extent • Draw on canvas: click and drag a rectangle delimiting the area to take into account • Enter the coordinates as xmin, xmax, ymin, ymax
Clip the input layer using the above (rectangle) extent Opţional	CLIP	[boolean] Implicit: False	The input layer will be clipped by the extent you defined before
Fields to include (leave empty to use all fields) Opţional	FIELDS	[string] [list] Implicit: []	Defines fields to keep from the imported vector file. If none is selected, all the fields are imported.
Select features using a SQL „WHERE" statement (Ex: column="value") Opţional	WHERE	[string] Implicit: „	Defines with a SQL „WHERE" statement which features should be selected for the output table
Group N features per transaction (Default: 2000) Opţional	GT	[string] Implicit: „	You can group the input features in transactions where N defines the size. By default N limits the transaction size to 20000 features.
Overwrite existing table Opţional	OVERWRITE	[boolean] Implicit: True	If there is a table with the same name in the database, and if this option is set to True, the table will be overwritten.
Append to existing table Opţional	APPEND	[boolean] Implicit: False	If checked / True the vector data will be appended to an existing table. New fields found in the input layer are ignored. By default a new table will be created.
Append and add new fields to existing table Opţional	ADDFIELDS	[boolean] Implicit: False	If activated the vector data will be appended to an existing table, there won't be created a new table. New fields found in input layer are added to the table. By default a new table will be created.
Do not launder columns/table names Opţional	LAUNDER	[boolean] Implicit: False	With this option checked you can prevent the default behaviour (converting column names to lowercase, removing spaces and other invalid characters).

continues on next page

Tabelul 28.279 – continuare din pagina precedentă

Etichetă	Nume	Tipul	Descriere
Do not create Spatial Index Opţional	INDEX	[boolean] Implicit: False	Prevents a spatial index for the output table from being created. By default, a spatial index is added.
Continue after a failure, skipping the failed feature Opţional	SKIPFAILURES	[boolean] Implicit: False	
Promote to Multipart Opţional	PROMOTETOMULTI	[boolean] Implicit: True	Casts features geometry type to multipart in the output table
Keep width and precision of input attributes Opţional	PRECISION	[boolean] Implicit: True	Avoids modifying column attributes to comply with input data
Opţiuni suplimentare de creare Opţional	OPTIONS	[string] Implicit: «» (fără opțiuni adiționale)	Additional GDAL creation options.

Rezultat

This algorithm has no output.

Cod Python

Algorithm ID: `gdal:importvectorintopostgisdatabasenewconnection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

Vector Information

Creates an information file that lists information about an OGR-supported data source. The output will be shown in a «Result» window and can be written into a HTML-file. The information includes the geometry type, feature count, the spatial extent, the projection information and many more.

This algorithm is derived from the GDAL ogrinfo utility.

Parametri

Etichetă	Nume	Tipul	Descriere
Strat de intrare	`INPUT`	[vector: oricare]	Stratul vectorial de intrare
Summary output only Opţional	`SUMMARY_ONLY`	[boolean] Implicit: True	
Suppress metadata info Opţional	`NO_METADATA`	[boolean] Implicit: False	
Layer information	`OUTPUT`	[html] Implicit: `[Salvare într-un fişier temporar]`	Specify the output HTML file that includes the file information. One of: • Salvare într-un Fișier Temporar • Salvare în Fișier… If no HTML-file is defined the output will be written to a temporary file

Rezultat

Etichetă	Nume	Tipul	Descriere
Layer information	`OUTPUT`	[html]	The output HTML-file that includes the file information.

Cod Python

Algorithm ID: `gdal:ogrinfo`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

ID-ul algoritmului se afișează atunci când treceți peste algoritmul din caseta instrumentelor de procesare. *Dicționarul parametrilor* furnizează valorile și NUMELE parametrilor. Pentru detalii despre rularea algoritmilor de procesare din consola Python, parcurgeți *Utilizarea algoritmilor de procesare din consolă*.

28.3 Furnizorul aplicațiilor OTB

OTB (Orfeo ToolBox) este o bibliotecă de procesare a imaginilor obținute prin teledetecție. De asemenea, dispune de aplicații care oferă funcționalități de procesare a imaginilor. Lista aplicațiilor și documentația acestora sunt indicate în Cartea de rețete OTB

CAPITOLUL 29

Plugin-uri

29.1 Plugin-uri QGIS

QGIS a fost conceput cu o arhitectură de plugin. Acest lucru permite adăugarea ușoară a multor caracteristici și funcții noi. Multe dintre caracteristicile din QGIS sunt de fapt implementate ca plugin-uri.

29.1.1 Plugin-uri de bază și externe

Plugin-urile QGIS sunt implementate fie ca **Plugin-uri de Bază**, fie ca **Plugin-uri Externe**.

Core Plugins are maintained by the QGIS Development Team and are automatically part of every QGIS distribution. They are written in one of two languages: **C++** or **Python**.

Most of External Plugins are currently written in Python. They are stored either in the «Official» QGIS Repository at https://plugins.qgis.org/plugins/ or in external repositories and are maintained by the individual authors. Detailed documentation about the usage, minimum QGIS version, home page, authors, and other important information are provided for the plugins in the Official repository. For other external repositories, documentation might be available with the external plugins themselves. External plugins documentation is not included in this manual.

To install or activate a plugin, go to *Plugins* menu and select *Manage and install plugins...*. Installed external python plugins are placed under the `python/plugins` folder of the active *user profile* path.

Paths to Custom C++ plugins libraries can also be added under *Settings* ► *Options* ► *System*.

29.1.2 Dialogul Plugin-urilor

Fila setărilor

At the bottom of the left panel, the *Settings* tab is the main place you can configure which plugins can be displayed in your application. You can use the following options:

- *Check for Updates on Startup*. Whenever an installed plugin has update available, QGIS will inform you *Every Time QGIS starts*, *Once a Day*, *Every 3 Days*, *Every Week*, *Every 2 Weeks* or *Every month*.
- *Show also Experimental Plugins*. QGIS will show you plugins in early stages of development, which are generally unsuitable for production use. For these plugins, you can install either the stable or the experimental version, and at any moment switch from one to the other.
- *Show also Deprecated Plugins*. These plugins are usually unmaintained because they have replacement functions in QGIS, a lack of maintainers, they rely on functions that are no longer available in QGIS… They are generally unsuitable for production use and appear grayed in the plugins list.

By default, in the *Plugin Repositories* section, QGIS provides you with its official plugin repository with the URL `https://plugins.qgis.org/plugins/plugins.xml?qgis=version` (where `<version>` represents the exact QGIS version you are running). To add external author repositories, click *Add…* and fill in the *Repository Details* form with a name and the URL. The URL can be of `http://` or `file://` protocol type.

The default QGIS repository is an open repository and you don't need any authentication to access it. You can however deploy your own plugin repository and require an authentication (basic authentication, PKI). You can get more information on QGIS authentication support in *Autentificarea* chapter.

If you do not want one or more of the added repositories, they can be disabled from the Settings tab via the *Edit…* button, or completely removed with the *Delete* button.

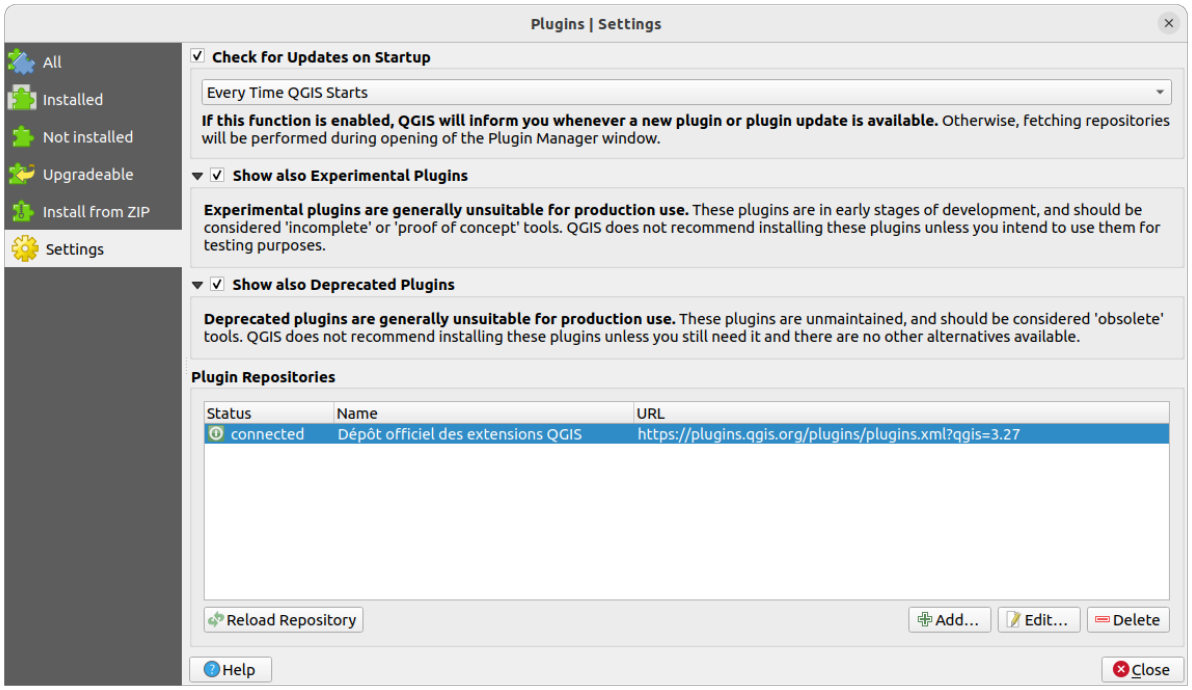


Fig. 29.1: The *Settings* tab

Browsing the plugins

The tabs

The upper tabs in the *Plugins* dialog provide you with lists of plugins based on their install, creation or update status. Depending on the plugins settings, available tabs can be:

- *All*: shows all the available plugins in the enabled repositories
- *Installed*: shows both the plugins you installed and the core plugins that are installed by default and you can not uninstall
- *Not installed*: shows uninstalled or not yet installed plugins in the enabled repositories
- *New*: shows plugins released since the last *Check for Updates on Startup*
- *Upgradeable*: shows installed plugins that have published a more recent version in the repository
- *Invalid*: shows all installed plugins that are currently broken for any reason (missing dependency, errors while loading, incompatible functions with QGIS version…)

At the top of the tabs, a *Search* function helps you find any plugin using metadata information (author, name, description, tag,…).

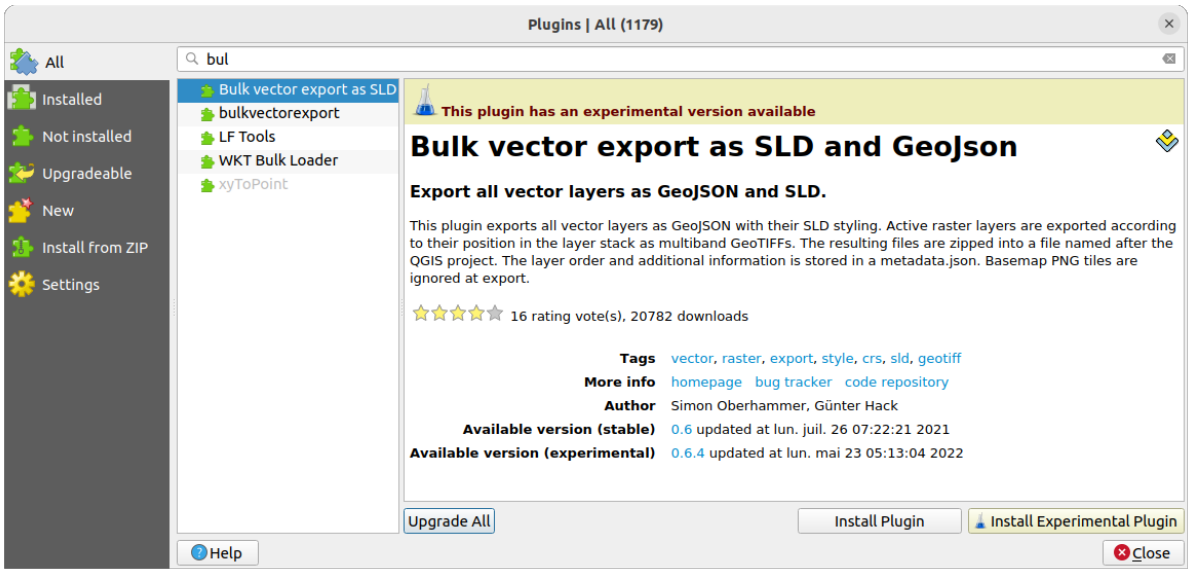


Fig. 29.2: Searching a plugin from the *All* tab

The Plugins

Select a plugin and you will have some metadata displayed in the right panel:

- information on whether the plugin is experimental or has an experimental version available (if *Show also Experimental Plugins* is checked)
- summary and description
- vot(urile) de evaluare (se poate vota pentru plugin-ul preferat!)
- etichete
- unele link-uri utile, ca pagina de start, cea a actualizărilor și a depozitului de cod

- autor(i)
- version(s) available with link to download page in the repository, or path to local folder for installed plugins

The *Plugin Manager* dialog allows you to interact with the latest version of the plugins. When enabled, the experimental version can be shown only if it is more recent than the latest stable version. Depending on the active tab, whether the selected plugin is installed, you will be given some of the following options:

- *Install*: installs the latest stable version of the selected plugin
- *Install Experimental Plugin*: installs the experimental version of the selected plugin
- *Reinstall Plugin*: installs the same stable version of the plugin e.g. after it has failed to load
- *Reinstall Experimental Plugin*: installs the same stable version of the plugin e.g. after it has failed to load
- *Upgrade Plugin*: upgrades selected plugin to its latest stable version
- *Upgrade Experimental Plugin*: upgrades selected plugin to its experimental version
- *Upgrade All*: upgrades all installed plugins to their more recent stable or experimental version (depending on whether their previously installed version was stable or experimental).
- *Downgrade Plugin*: moves from the experimental version of the plugin to its previous stable version
- *Downgrade Experimental Plugin*: moves from an experimental version of the plugin to its latest published experimental version. This may occur when playing with a not yet published version.
- *Uninstall Plugin*: removes the installed plugin from the user profile

An installed plugin displays a checkbox on its left. Uncheck it to temporarily deactivate the plugin.

Right-click on a plugin in the list and you will be able to sort the plugins list by various metadata. The new order applies to all the tabs. Sort options are:

- *Sort by Name*
- *Sort by Downloads*
- *Sort by Vote*
- *Sort by Status*
- *Sort by Date Created*
- *Sort by Date Updated*

The Install from ZIP tab

The *Install from ZIP* tab provides a file selector widget to import plugins in a zipped format, e.g. plugins downloaded directly from their repository. Encrypted files are supported.

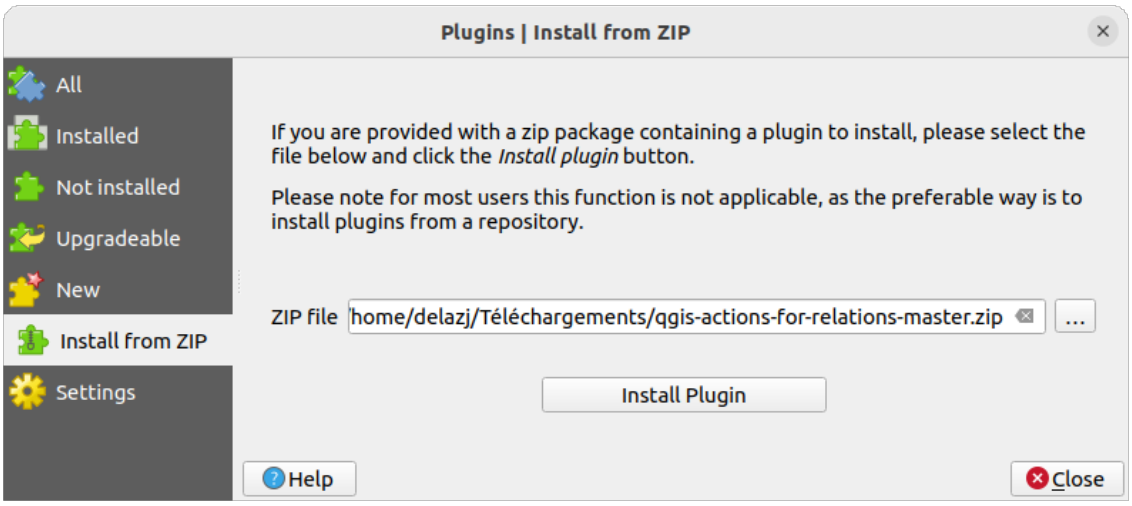


Fig. 29.3: The *Install from zip* tab

29.2 Using QGIS Core Plugins

29.2.1 DB Manager Plugin

The DB Manager Plugin is intended to be the main tool to integrate and manage spatial database formats supported by QGIS (PostGIS, SpatiaLite, GeoPackage, Oracle Spatial, Virtual layers) in one user interface. The DB Manager Plugin provides several features. You can drag layers from the QGIS Browser into the DB Manager, and it will import your layer into your spatial database. You can drag and drop tables between spatial databases and they will get imported.

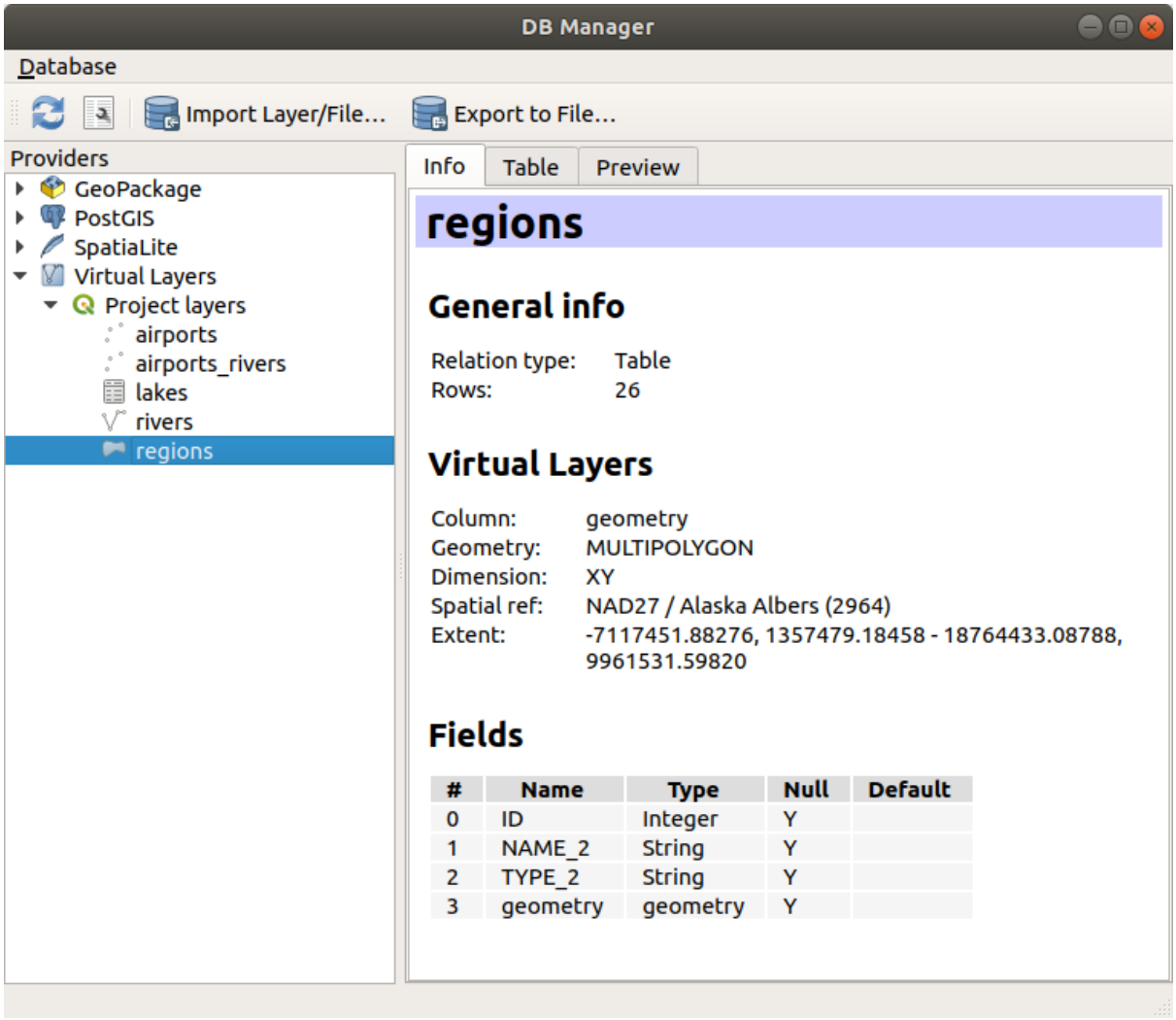


Fig. 29.4: DB Manager dialog

The *Database* menu allows you to connect to an existing database, to start the SQL window and to exit the DB Manager Plugin. Once you are connected to an existing database, the menus *Schema* (relevant for DBMSs, such as PostGIS / PostgreSQL) and *Table* will appear.

The *Schema* menu includes tools to create and delete (only if empty) schemas and, if topology is available (e.g. with PostGIS topology), to start a *TopoViewer*.

The *Table* menu allows you to create and edit tables and to delete tables and views. It is also possible to empty tables and to move tables between schemas. You can *Run Vacuum Analyze* for the selected table. *Vacuum* reclaims space and makes it available for reuse, and *analyze* updates statistics that is used to determine the most efficient way to execute a query. *Change Logging...* allows you to add change logging support to a table. Finally, you can *Import Layer/File...* and *Export to File...*.

Notă: Using the DB Manager it is possible to add comments for tables and columns of a PostgreSQL Database.

The *Providers* window lists all existing databases supported by QGIS. With a double-click, you can connect to the database. With the right mouse button, you can rename and delete existing schemas and tables. Tables can also be added to the QGIS canvas with the context menu.

If connected to a database, the **main** window of the DB Manager offers four tabs. The *Info* tab provides information about the table and its geometry, as well as about existing fields, constraints and indexes. It allows you to create a spatial index on a the selected table. The *Table* tab shows the table, and the *Preview* tab renders the geometries as

preview. When you open an *SQL Window*, it will be placed in a new tab.

Working with the SQL Window

You can use the DB Manager to execute SQL queries against your spatial database. Queries can be saved and loaded, and there the *SQL Query Builder* will help you formulate your queries. You can even view spatial output by checking *Load as new layer* and specifying *Column(s) with unique values* (IDs), *Geometry column* and *Layer name (prefix)*. It is possible to highlight a portion of the SQL to only execute that portion when pressing `Ctrl+R` or clicking the *Execute* button.

The *Query History* button stores the last 20 queries of each database and provider.

Double clicking on an entry will add the string to the SQL window.

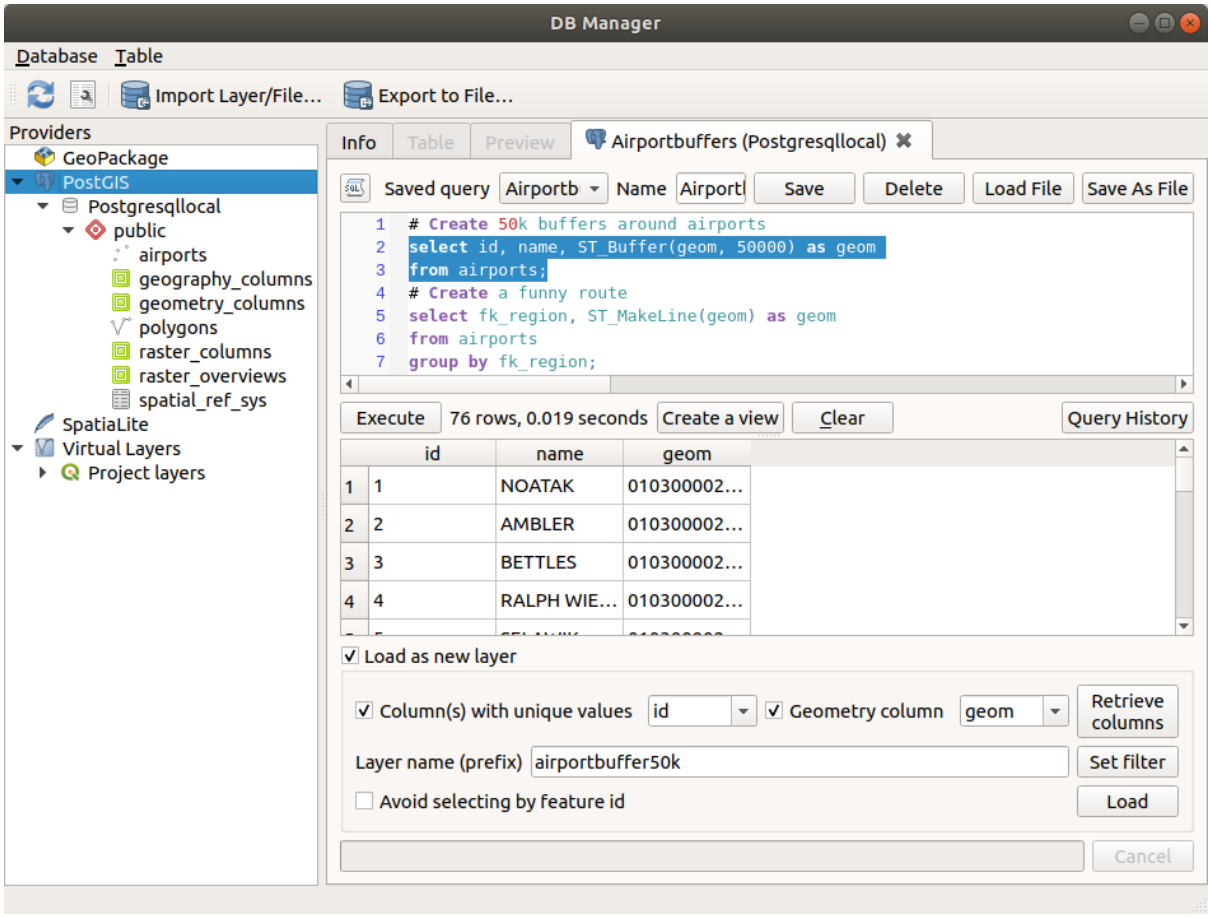


Fig. 29.5: Executing SQL queries in the DB Manager SQL window

Notă: The SQL Window can also be used to create Virtual Layers. In that case, instead of selecting a database, select **QGIS Layers** under **Virtual Layers** before opening the SQL Window. See *Creating virtual layers* for instructions on the SQL syntax to use.

29.2.2 Geometry Checker Plugin

Geometry Checker is a powerful core plugin to check and fix the geometry validity of a layer. It is available from the *Vector* menu (*Check Geometries…*).

Configuring the checks

The *Check Geometries* dialog shows different grouped settings in the first tab (*Setup*):

- *Input vector layers*: to select the layers to check. A *Only selected features* checkbox can be used to restrict the checking to the geometries of the selected features.
- *Allowed geometry types* gives the chance to restrict the geometry type of the input layer(s) to:
  - Point
  - Multipoint
  - Line
  - Multiline
  - Polygon
  - Multipolygon
- *Geometry validity*. Depending on geometry types you can choose between:
  - *Self intersections*
  - *Duplicate nodes*
  - *Self contacts*
  - *Polygon with less than 3 nodes*.
- *Geometry properties*. Depending on geometry types, different options are available:
  - *Polygons and multipolygons may not contain any holes*
  - *Multipart objects must consist of more than one part*
  - *Lines must not have dangles*
- *Geometry conditions*. Allows you to add some condition to validate the geometries with:
  - *Minimal segment length (map units)* 1,00
  - *Minimum angle between segment (deg)* 1,00
  - *Minimal polygon area (map units sqr.)* 1,00
  - *No sliver polygons* with a *Maximum thinness* 1,00 and a *Max. area (map units sqr.)* 1,00
- *Topology checks*. Depending on geometry types, many different options are available:
  - *Checks for duplicates*
  - *Checks for features within other features*
  - *Checks for overlaps smaller than* 1,00
  - *Checks for gaps smaller than* 1,00

  - *Points must be covered by lines*
  - *Points must properly lie inside a polygon*
  - *Lines must not intersect any other lines*
  - *Lines must not intersect with features of layer*
  - *Polygons must follow boundaries of layer*
- *Tolerance*. You can define the tolerance of the check in map layer units.
- *Output vector layer* gives the choice to:
  - *Modify input layer*
  - *Create new layers*

When you are happy with the configuration, you can click on the *Run* button.

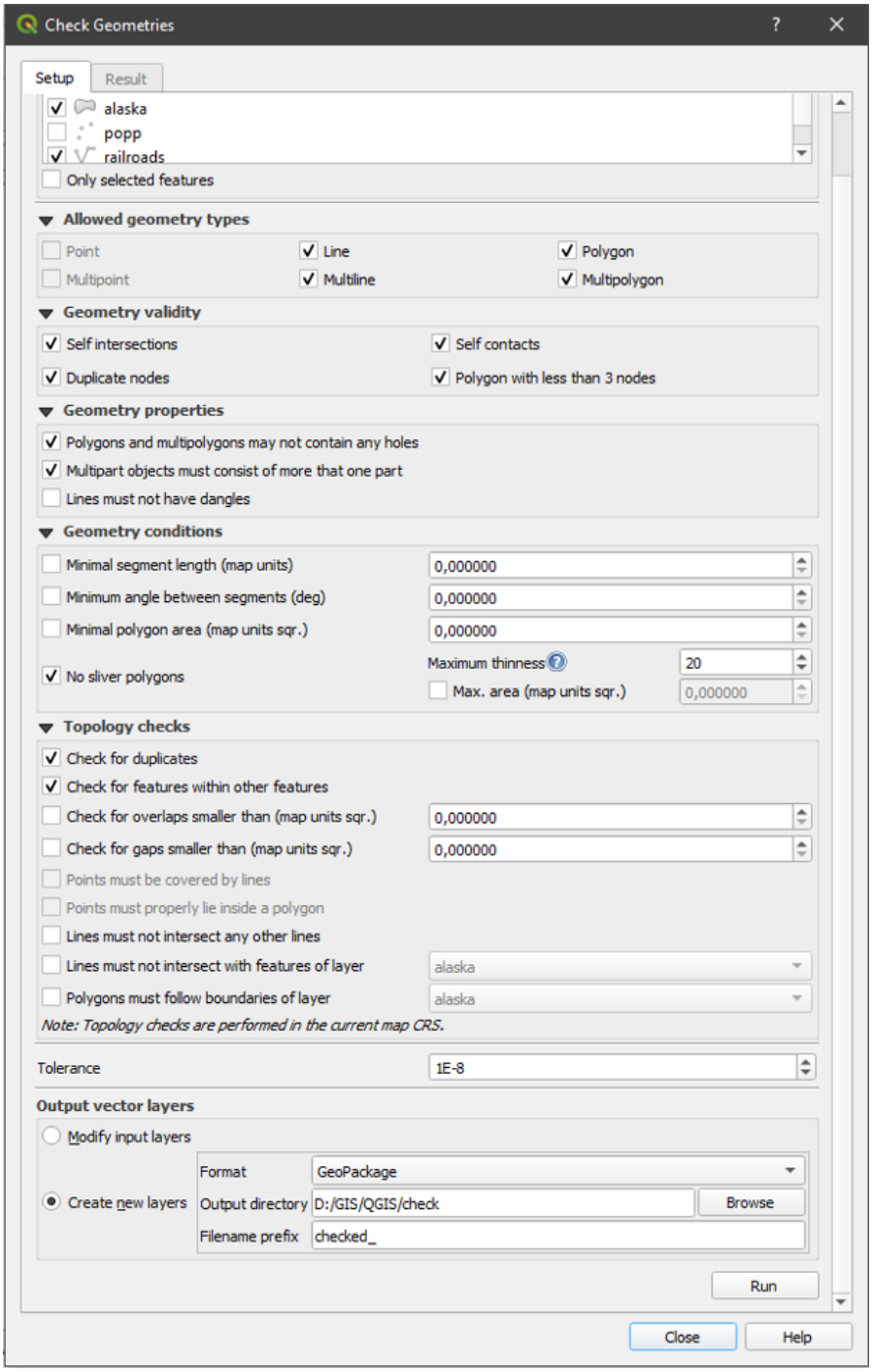


Fig. 29.6: The Geometry Checker Plugin

The *Geometry Checker Plugin* can find the following errors:

- Self intersections: a polygon with a self intersection
- Duplicate nodes: two duplicates nodes in a segment
- Holes: hole in a polygon
- Segment length: a segment length lower than a threshold
- Minimum angle: two segments with an angle lower than a threshold
- Minimum area: polygon area lower than a threshold

- Silver polygon: this error come from very small polygon (with small area) with a large perimeter
- Duplicates features
- Feature within feature
- Overlaps: polygon overlapping
- Gaps: gaps between polygons

The following figure shows the different checks made by the plugin.

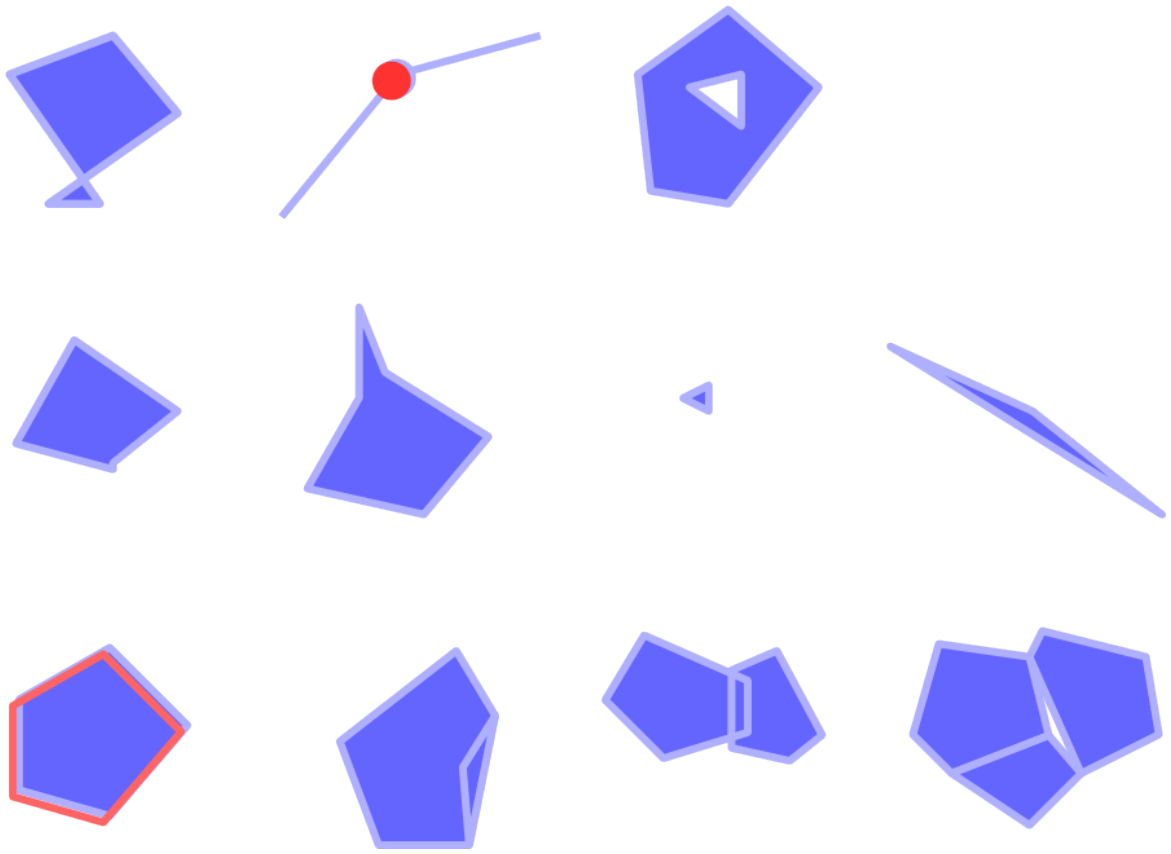


Fig. 29.7: Some checks supported by the plugin

Analysing the results

The results appear in the second tab (*Result*) and as an overview layer of the errors in the canvas (its name has the default prefix `checked_`). A table lists the *Geometry check result* with one error per row and columns containing: the layer name, an ID, the error type, then the coordinates of the error, a value (depending on the type of the error) and finally the resolution column which indicates the resolution of the error. At the bottom of this table, you can *Export* the error into different file formats. You also have a counter with the number of total errors and fixed ones.

You can select a row to see the location of the error. You can change this behavior by selecting another action between *Error* (default), *Feature*, *Don't move*, and *Highlight selected features*.

Below the zoom action when clicking on the table row, you can:

- *Show selected features in attribute table*
- *Fix selected errors using default resolution*
- *Fix selected errors, prompt for resolution method* You will see a window to choose the resolution's method among which:

  - Merge with neighboring polygon with longest shared edge
  - Merge with neighboring polygon with largest area
  - Merge with neighboring polygon with identical attribute value, if any, or leave as is
  - Delete feature
  - No action
- *Error resolution settings* allows you to change the default resolution method depending on the error type

Sfat: Fix multiple errors

You can fix multiple errors by selecting more than one row in the table with the *CTRL + click* action.

Finally, you can choose which *Attribute to use when merging features by attribute value*.

29.2.3 MetaSearch Catalog Client

Introducere

MetaSearch is a QGIS plugin to interact with metadata catalog services, supporting both the OGC API - Records and OGC Catalog Service for the Web (CSW) standards.

MetaSearch provides an easy and intuitive approach and user-friendly interface to searching metadata catalogs within QGIS.

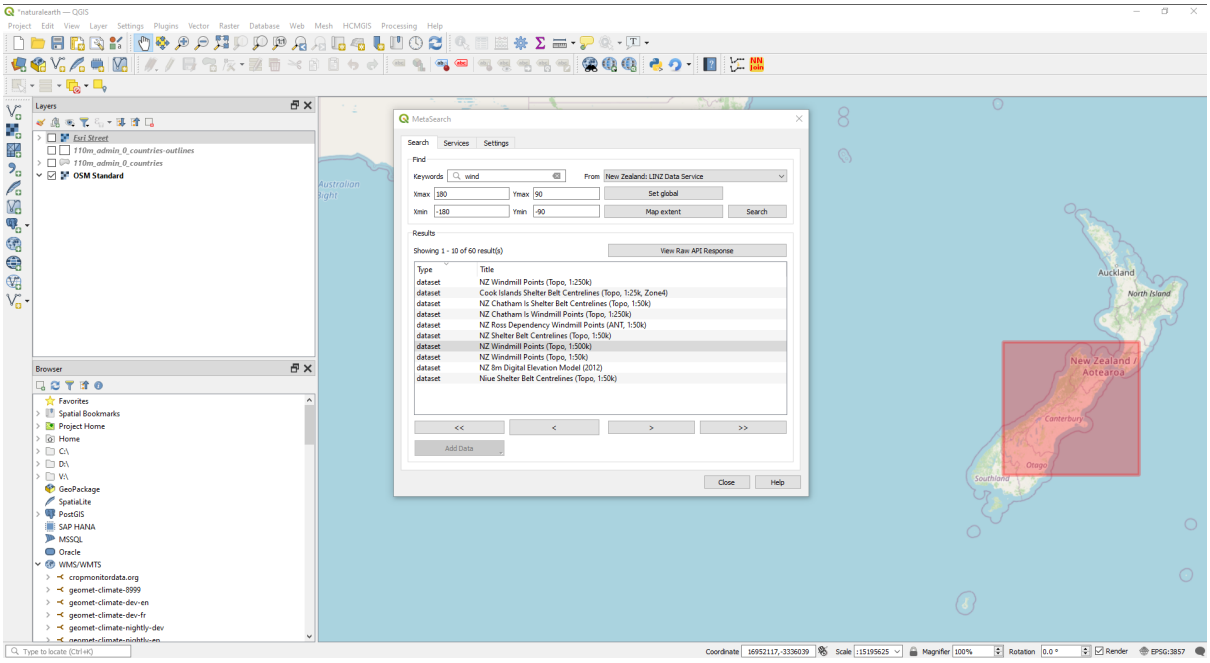


Fig. 29.8: Search and results of Services in MetaSearch

Working with Metadata Catalogs in QGIS

MetaSearch is included by default in QGIS, with all of its dependencies, and can be enabled from the QGIS Plugin Manager.

OGC API - Records

OGC API - Records is an OGC (Open Geospatial Consortium) standard for the discovery of geospatial resources on the Web.

CSW (Catalog Service for the Web)

CSW (Catalog Service for the Web) is an OGC (Open Geospatial Consortium) specification that defines common interfaces to discover, browse and query metadata about data, services, and other potential resources.

Startup

To start MetaSearch, click the icon or select *Web* ► *MetaSearch* ► *MetaSearch* via the QGIS main menu. The MetaSearch dialog will appear. The main GUI consists of three tabs: *Services*, *Search* and *Settings*.

Managing Catalog Services

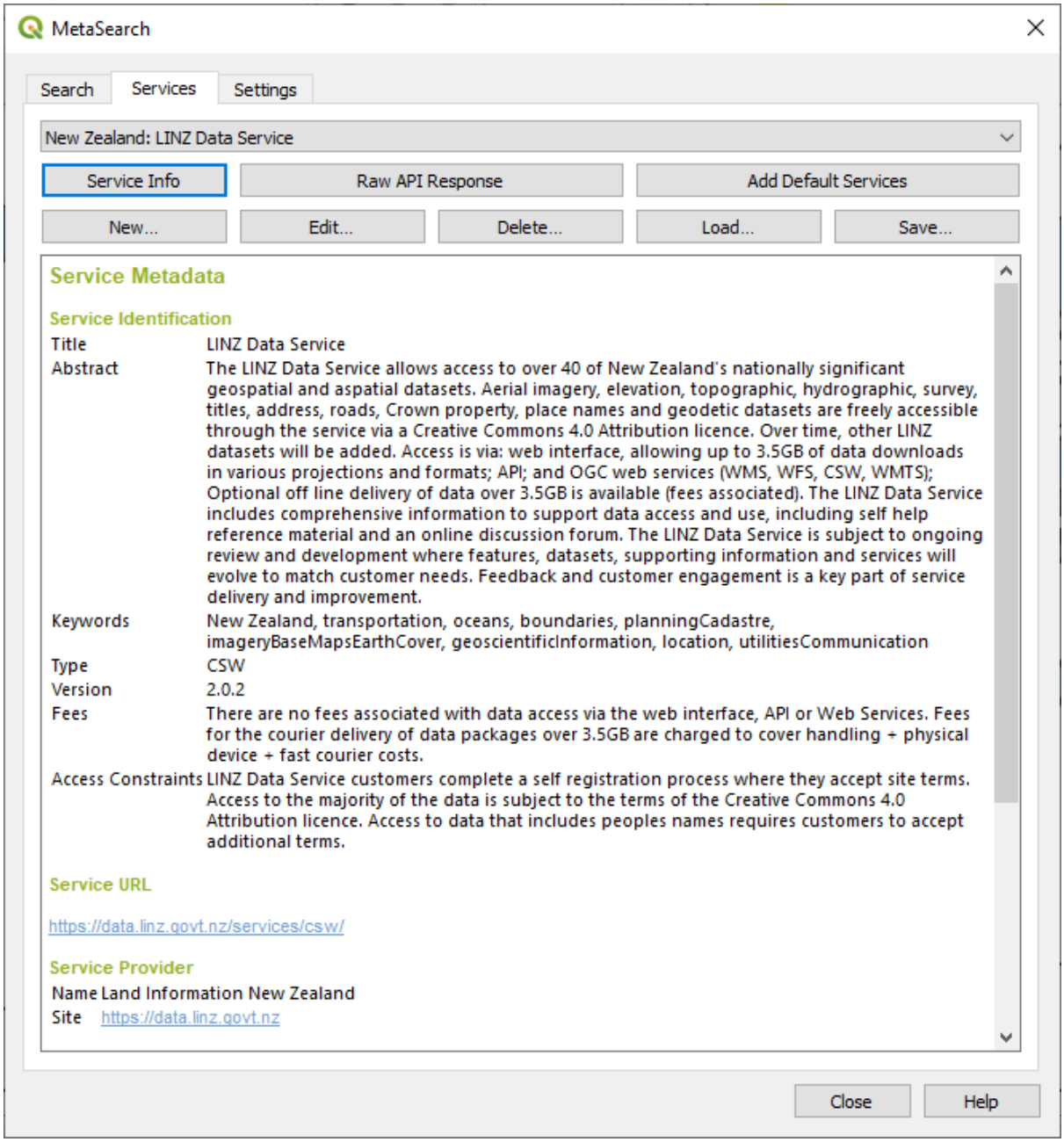


Fig. 29.9: Managing Catalog Services

The *Services* tab allows the user to manage all available catalog services. MetaSearch provides a default list of Catalog Services, which can be added by pressing the *Add Default Services* button.

To find all listed Catalog Service entries, click the dropdown select box.

To add a Catalog Service entry:

1. Click the *New* button
2. Enter a *Name* for the service, as well as the *URL* (endpoint). Note that for OGC CSW 2.0.2 Catalogs, only the base URL is required (not a full GetCapabilities URL). For OGC API - Records Catalogs, the URL should be the path to the collection endpoint
3. If the Catalog requires authentication, enter the appropriate *User name* and *Password* credentials.

4. Click *OK* to add the service to the list of entries.

To edit an existing Catalog Service entry:

1. Select the entry you would like to edit
2. Click the *Edit* button
3. And modify the *Name* or *URL* values
4. Clic pe *OK*

To delete a Catalog Service entry, select the entry you would like to delete and click the *Delete* button. You will be asked to confirm deleting the entry.

MetaSearch allows loading and saving connections to an XML file. This is useful when you need to share settings between applications. Below is an example of the XML file format.

```
<?xml version="1.0" encoding="UTF-8"?>
<qgsCSWConnections version="1.0">
    <csw type="OGC CSW 2.0.2" name="Data.gov CSW" url="https://catalog.data.gov/
↪csw-all"/>
    <csw type="OGC CSW 2.0.2" name="Geonorge - National CSW service for Norway"␣
↪url="https://www.geonorge.no/geonetwork/srv/eng/csw"/>
    <csw type="OGC CSW 2.0.2" name="Geoportale Nazionale - Servizio di ricerca␣
↪Italiano" url="http://www.pcn.minambiente.it/geoportal/csw"/>
    <csw type="OGC CSW 2.0.2" name="LINZ Data Service" url="http://data.linz.govt.
↪nz/feeds/csw"/>
    <csw type="OGC CSW 2.0.2" name="Nationaal Georegister (Nederland)" url="http://
↪www.nationaalgeoregister.nl/geonetwork/srv/eng/csw"/>
    <csw type="OGC CSW 2.0.2" name="RNDT - Repertorio Nazionale dei Dati␣
↪Territoriali - Servizio di ricerca" url="http://www.rndt.gov.it/RNDT/CSW"/>
    <csw type="OGC CSW 2.0.2" name="UK Location Catalogue Publishing Service" url=
↪"http://csw.data.gov.uk/geonetwork/srv/en/csw"/>
    <csw type="OGC CSW 2.0.2" name="UNEP/GRID-Geneva Metadata Catalog" url="http://
↪metadata.grid.unep.ch:8080/geonetwork/srv/eng/csw"/>
</qgsCSWConnections>
```

To load a list of entries:

1. Click the *Load* button. A new window will appear.
2. Click the *Browse* button and navigate to the XML file of entries you wish to load.
3. Click *Open*. The list of entries will be displayed.
4. Select the entries you wish to add from the list and click *Load*.

Click the *Service Info* button to display information about the selected Catalog Service such as service identification, service provider and contact information. If you would like to view the raw API response, click the *Raw API Response* button. A separate window will open displaying server information in raw JSON or XML format.

Searching Catalog Services

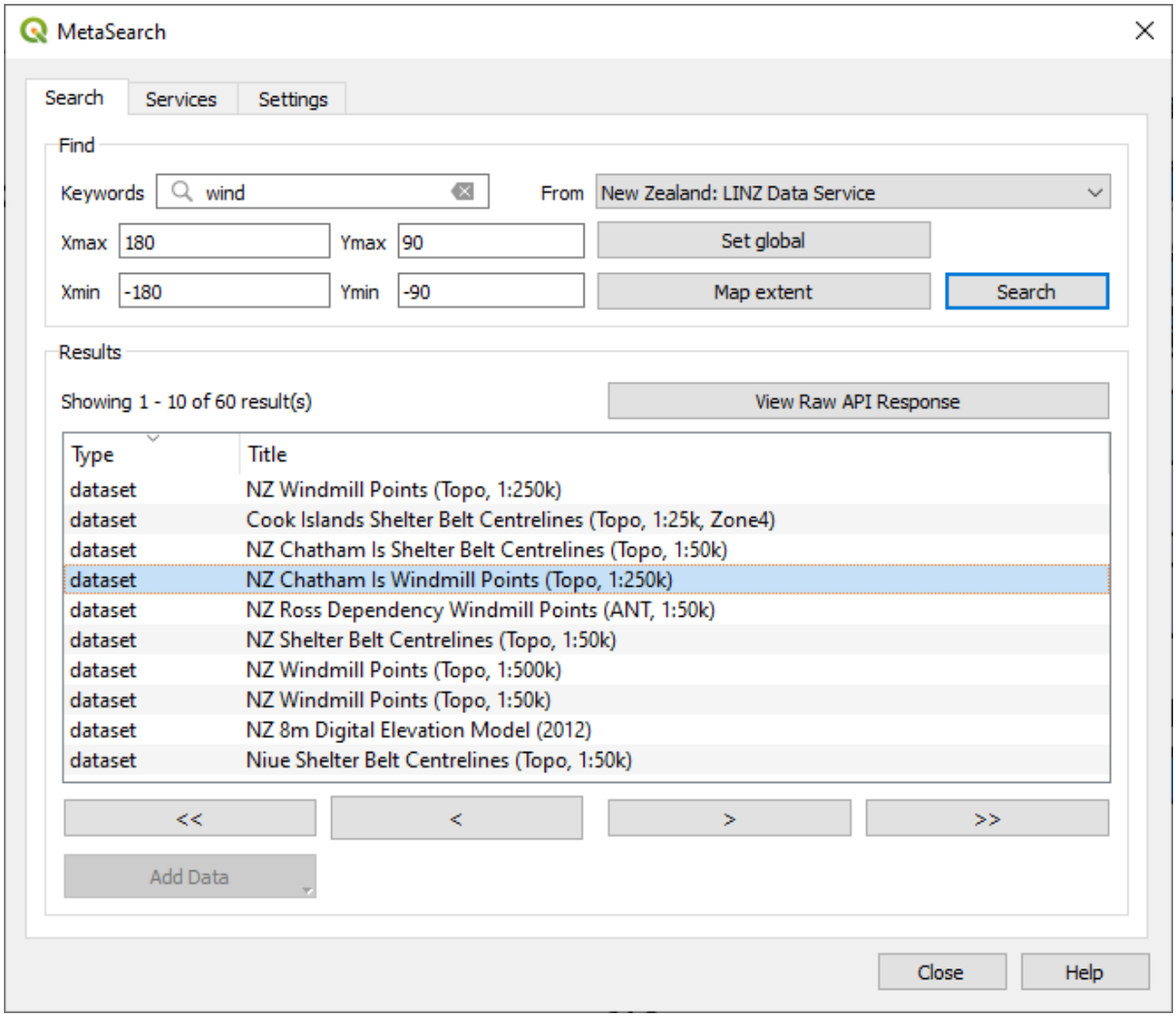


Fig. 29.10: Searching catalog services

The *Search* tab allows the user to query Catalog Services for data and services, set various search parameters and view results.

The following search parameters are available:

- *Keywords*: free text search keywords;
- *From*: the Catalog Service to perform the query against;
- **Bounding box**: the spatial area of interest to filter, defined by *Xmax*, *Xmin*, *Ymax*, and *Ymin*. Click *Set Global* to do a global search, click *Map Extent* to do a search in the visible area, or enter values manually.

Clicking the *Search* button will search the selected Metadata Catalog. Search results are displayed in a list, and can be sorted by clicking on the column header. You can navigate through search results with the directional buttons below the search results.

Select a result and:

- Click the *View Raw API Response* button to open a window with the service response in raw JSON or XML format.
- If the metadata record has an associated bounding box, a footprint of the bounding box will be displayed on the map.

- Double-click the record to display the record metadata with any associated access links. Clicking a link opens the link in the user's web browser.
- If the record is a supported web service (WMS/WMTS, WFS, WCS, ArcGIS REST Service, etc.) or a GIS File, the *Add Data* button will be enabled. When clicking this button, MetaSearch will verify if this is a valid OWS. The service will then be added to the appropriate QGIS connection list, and the appropriate connection dialog will appear. If you choose *Add GIS File* the linked GIS file will be added to the current project.

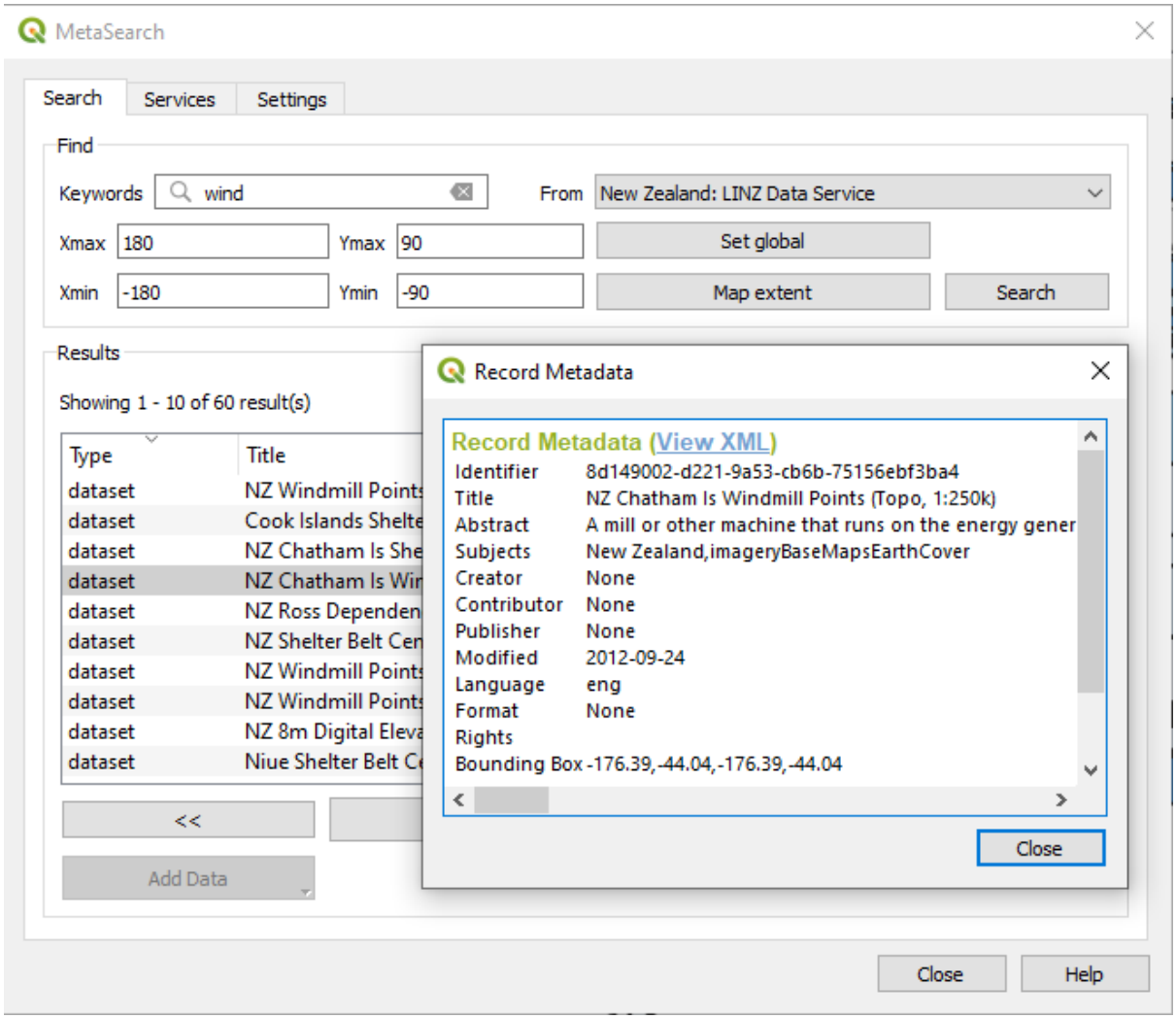


Fig. 29.11: Metadata record display

Setări

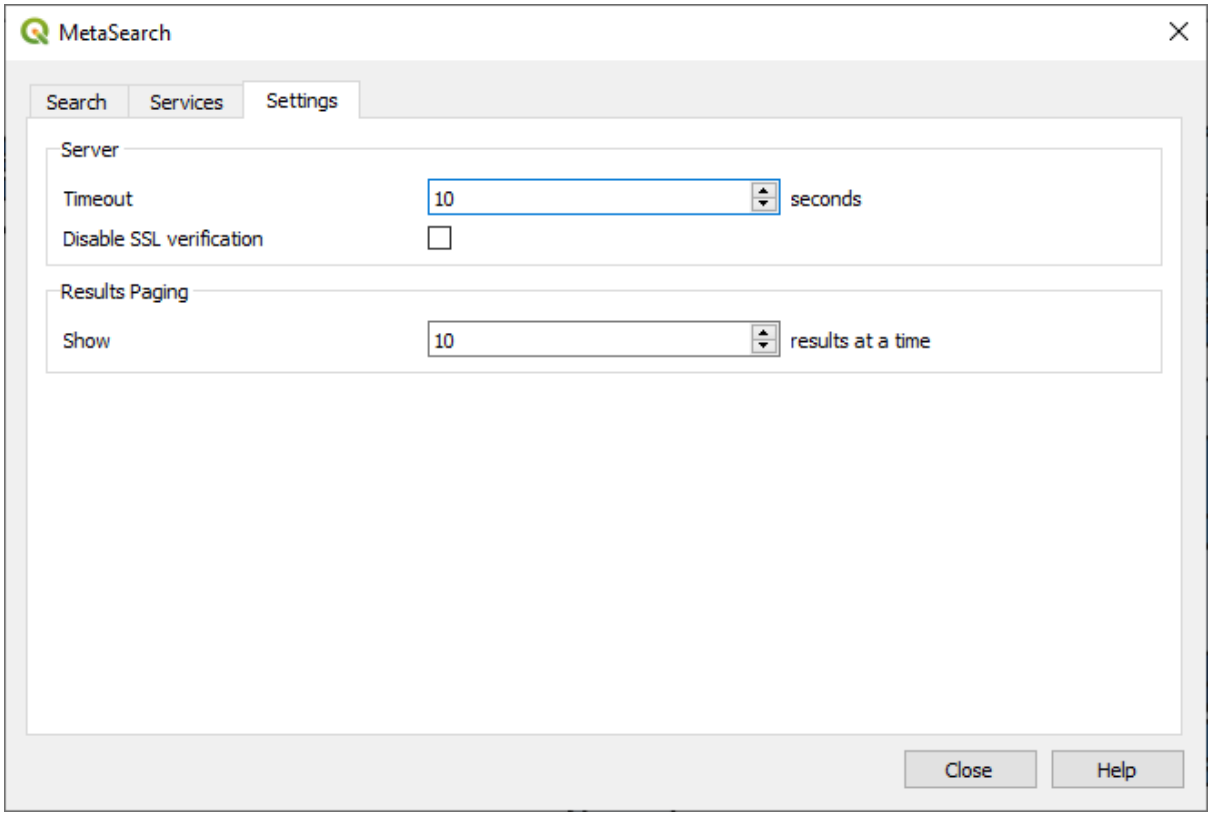


Fig. 29.12: MetaSearch settings

You can fine tune MetaSearch with the following *Settings*:

- *Server Timeout*: when searching metadata catalogs, the number of seconds for blocking connection attempt. Default value is 10.
- *Disable SSL verification*: option to switch off ssl verification.
- *Results paging*: when searching metadata catalogs, the number of results to show per page. Default value is 10.

Catalog Server Errors

In some cases, the Catalog will work in a web browser, but not in MetaSearch. This may be due to the Catalog server's configuration/setup. Catalog server providers should ensure URLs are consistent and up to date in their configuration (this is common in HTTP -> HTTPS redirection scenarios). Please see the pycsw FAQ item for a deeper explanation of the issue and fix. Although the FAQ item is pycsw specific it can also apply in general to other Catalog servers.

29.2.4 Offline Editing Plugin

For data collection, it is a common situation to work with a laptop or a cell phone offline in the field. Upon returning to the network, the changes need to be synchronized with the master datasource (e.g., a PostGIS database). If several persons are working simultaneously on the same datasets, it is difficult to merge the edits by hand, even if people don't change the same features.

The Offline Editing Plugin automates the synchronisation by copying the content of the datasource to a SpatiaLite or GeoPackage database and storing the offline edits to dedicated tables. After being connected to the network again, it is possible to apply the offline edits to the master dataset.

To use the plugin:

1. Open a project with some vector layers (e.g., from an Esri Shapefile, PostGIS or WFS-T datasource).
2. Assuming you have already enabled the plugin (see *Plugin-uri de bază și externe*) go to *Database* ► *Offline Editing* ► *Convert to offline project*. The eponym dialog opens.
3. Select the *Storage type*. It can be of *GeoPackage* or *SpatiaLite* database type.
4. Use the *Browse* button to indicate the location of the database in which to store the *Offline data*. It can be an existing file or one to create.
5. In the *Select remote layers* section, check the layers you'd like to save. The content of the layers is saved to database tables.

   Notă: Since target database formats do not have native list support, the offline editing plugin transforms {string, number} list fields into string fields where values are separated by commas. This allows reading and edit of the contents of those fields when offline.

   If you would like to handle both the field from the original layer and the offline layer, you can rely on the *try()* and *array* expression functions, e.g.:

   ```
   try(array_contains("field",1),array_contains(string_to_array("field"),1))
   ```

   ---

6. You can check *Only synchronize selected features if a selection is present* allowing to only save and work on a subset. It can be invaluable in case of large layers.

   This is all!
7. Save your project and bring it on the field.
8. Edit the layers offline.
9. After being connected again, upload the changes using *Database* ► *Offline Editing* ► *Synchronize*.

Notă: Layers that are used offline are marked with the icon in the *Layers* panel.

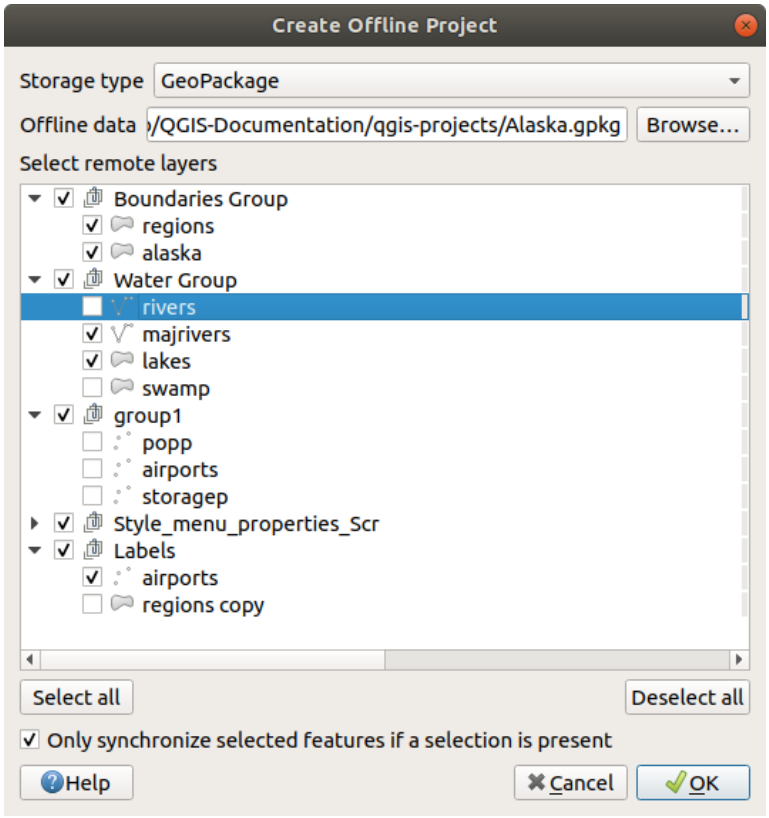


Fig. 29.13: Create an offline project

29.2.5 Topology Checker Plugin

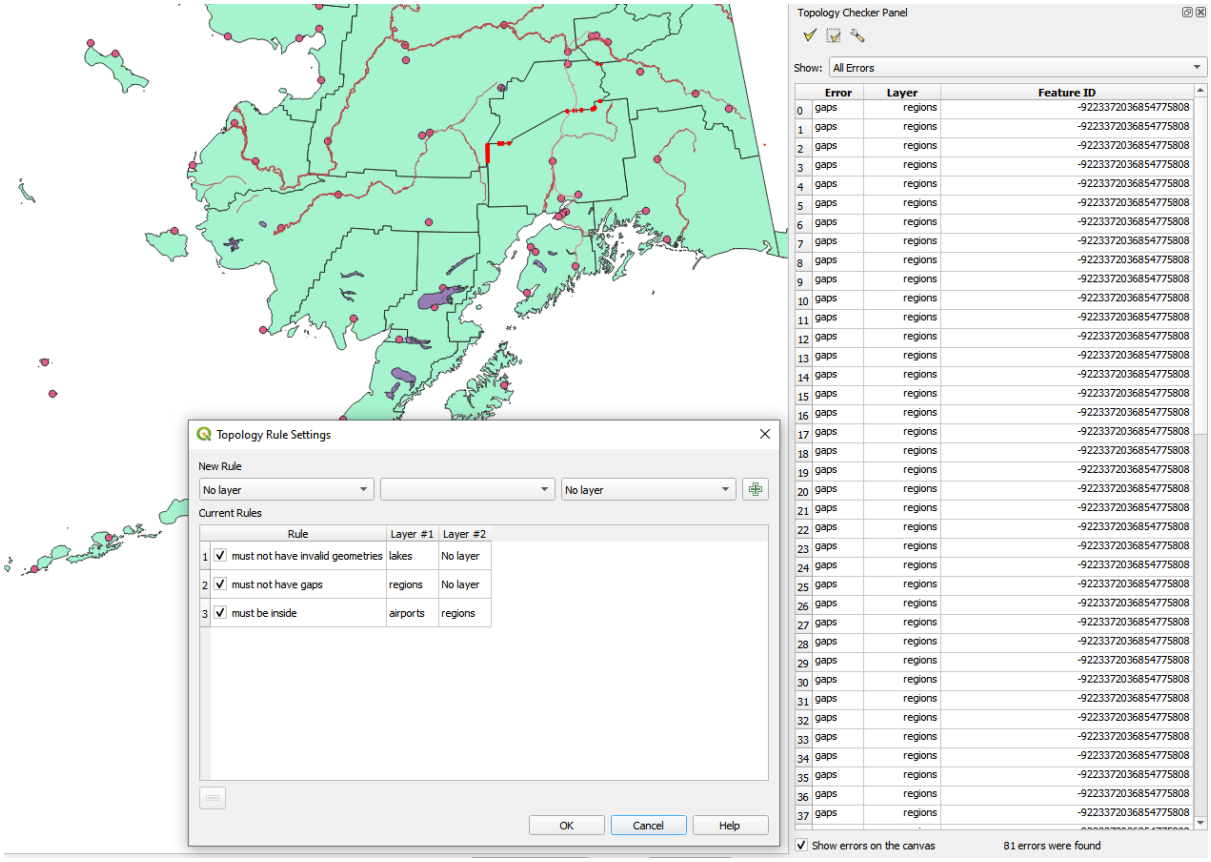


Fig. 29.14: The Topology Checker Plugin

Topology describes the relationships between points, lines and polygons that represent the features of a geographic region. With the Topology Checker plugin, you can look over your vector files and check the topology with several topology rules. These rules check with spatial relations whether your features «Equal», «Contain», «Cover», are «CoveredBy», «Cross», are «Disjoint», «Intersect», «Overlap», «Touch» or are «Within» each other. It depends on your individual questions which topology rules you apply to your vector data (e.g., normally you won't accept overshoots in line layers, but if they depict dead-end streets you won't remove them from your vector layer).

QGIS has a built-in topological editing feature, which is great for creating new features without errors. But existing data errors and user-induced errors are hard to find. This plugin helps you find such errors through a list of rules.

To enable the *Topology checker* plugin:

1. Go to *Plugins* menu
2. Open *Manage and Install plugins* and choose *Installed*
3. Enable *Topology checker*
4. Close the *Plugin manager* dialog. A *Topology checker* entry is added to the *Vector* menu.

After you enabled *Topology checker* open it and choose *Configure* to create your topology rules.

On **point layers** the following rules are available:

- *Must be covered by*: Here you can choose a vector layer from your project. Points that aren't covered by the given vector layer occur in the «Error» field.
- *Must be covered by endpoints of*: Here you can choose a line layer from your project.

- *Must be inside*: Here you can choose a polygon layer from your project. The points must be inside a polygon. Otherwise, QGIS writes an «Error» for the point.
- *Must not have duplicates*: Whenever a point is represented twice or more, it will occur in the «Error» field.
- *Must not have invalid geometries*: Checks whether the geometries are valid.
- *Must not have multi-part-geometries*: All multi-part points are written into the «Error» field.

On **line layers**, the following rules are available:

- *End points must be covered by*: Here you can select a point layer from your project.
- *Must not have dangles*: This will show the overshoots in the line layer.
- *Must not have duplicates*: Whenever a line feature is represented twice or more, it will occur in the «Error» field.
- *Must not have invalid geometries*: Checks whether the geometries are valid.
- *Must not have multi-part geometries*: Sometimes, a geometry is actually a collection of simple (single-part) geometries. Such a geometry is called multi-part geometry. If it contains just one type of simple geometry, we call it multi-point, multi-linestring or multi-polygon. All multi-part lines are written into the «Error» field.
- *Must not have pseudos*: A line geometry's endpoint should be connected to the endpoints of two other geometries. If the endpoint is connected to only one other geometry's endpoint, the endpoint is called a pseudo node.

On **polygon layers**, the following rules are available:

- *Must contain*: Polygon layer must contain at least one point geometry from the second layer.
- *Must not have duplicates*: Polygons from the same layer must not have identical geometries. Whenever a polygon feature is represented twice or more it will occur in the «Error» field.
- *Must not have gaps*: Adjacent polygons should not form gaps between them. Administrative boundaries could be mentioned as an example (US state polygons do not have any gaps between them…).
- *Must not have invalid geometries*: Checks whether the geometries are valid. Some of the rules that define a valid geometry are:
  - Polygon rings must close.
  - Rings that define holes should be inside rings that define exterior boundaries.
  - Rings may not self-intersect (they may neither touch nor cross one another).
  - Rings may not touch other rings, except at a point.
- *Must not have multi-part geometries*: Sometimes, a geometry is actually a collection of simple (single-part) geometries. Such a geometry is called multi-part geometry. If it contains just one type of simple geometry, we call it multi-point, multi-linestring or multi-polygon. For example, a country consisting of multiple islands can be represented as a multi-polygon.
- *Must not overlap*: Adjacent polygons should not share common area.
- *Must not overlap with*: Adjacent polygons from one layer should not share common area with polygons from another layer.

When you create a *New rule* click on the Add rule to include it to the *Current rules*. You can enable or disable individual rules by clicking on the checkbox. Right-clicking over a rule provides the following options:

- *Select All* the rules
- *Activate* or *Deactivate* the selected rules
- *Toggle activation* of selected rules
- *Delete* selected rules. This can also be achieved with the Delete selected rules button.

Press *OK* and then choose from the *Topology checker* panel:

- Validate All: applies the active rules to all the features of the involved layer(s)
- or Validate Extent: applies the active rules to the features of the involved layer(s), within the current map canvas. The button is kept pushed and the results will update as the map canvas extent changes.

Errors will show up in the table of results containing type of error, layer and feature ID. Use *Filter errors by rule* menu to filter the errors to a specific error type.

Check *Show errors on the canvas* to show error location on the canvas. Clicking a row in the table will zoom the map canvas to the concerned feature, where you can use *QGIS digitizing tools* to fix the error.

Below is the list of Core plugins provided with QGIS. They are not necessarily enabled by default.

Pictogramă	Plugin	Descriere	Manual Reference
	DB Manager	Manage your databases within QGIS	*DB Manager Plugin*
	Verificatorul de Geometrii	Check and repair errors in vector geometries	*Geometry Checker Plugin*
	GRASS 7	GRASS functionality	*Integrarea GRASS GIS*
	GRASS GIS provider	GRASS GIS Processing functionality	*Integrarea GRASS GIS*
	MetaSearch Catalog Client	Interact with metadata catalog services (CSW)	*MetaSearch Catalog Client*
	Offline Editing	Offline editing and synchronizing with database	*Offline Editing Plugin*
	OrfeoToolbox provider	OrfeoToolbox Processing provider	*Furnizorul aplicațiilor OTB*
	Procesare	Spatial data processing framework	*Cadrul de procesare al QGIS*
	Topology Checker	Find topological errors in vector layers	*Topology Checker Plugin*

Notă: To use the Core Plugins GRASS 7, GRASS GIS provider, or OrfeoToolbox provider, they have to be configured. Informations can be found *here*.

29.3 Consola Python din QGIS

As you will see later in this chapter, QGIS has been designed with a plugin architecture. Plugins can be written in Python, a very famous language in the geospatial world.

QGIS brings a Python API (see PyQGIS Developer Cookbook for some code sample) to let the user interact with its objects (layers, feature or interface). QGIS also has a Python console.

The QGIS Python Console is an interactive shell for Python command executions. It also has a Python file editor that allows you to edit and save your Python scripts. Both console and editor are based on PyQScintilla2 package. To open the console go to *Plugins* ► *Python Console* (`Ctrl+Alt+P`).

29.3.1 Consola interactivă

The console is a Python interpreter that allows you to execute Python commands. Modules from QGIS (analysis, core, gui, server, processing, 3d) and Qt (QtCore, QtGui, QtNetwork, QtWidgets, QtXml) as well as Python's math, os, re and sys modules are already imported and can be used directly.

Consola interactivă este alcătuită dintr-o bară de instrumente, o zonă de intrare și una de ieșire.

Fig. 29.15: The Python Console

Bara de instrumente

Bara instrumentelor propune următoarele instrumente:

- Clear Console to wipe the output area;
- Run Command available in the input area: same as pressing `Enter`;
- Show Editor: toggles *Editorul de Cod* visibility;
- Options...: opens a dialog to configure *console properties*;
- Help... provides a menu to access various documentation:
  - *Python Console Help* (the current page)
  - PyQGIS API documentation
  - PyQGIS Cookbook
- Dock Code Editor to dock or undock the panel in QGIS interface

Input area

The Console input area main features are:

- Code completion, highlighting syntax and calltips for the following APIs:
  - Python
  - PyQGIS
  - PyQt5
  - QScintilla2
  - osgeo-gdal-ogr

- `Ctrl+Alt+Space` to view the auto-completion list if enabled in the *Python Console settings*;
- Execute code snippets from the input area by typing and pressing `Enter` or *Run Command*;
- Execute code snippets from the output area using the *Enter Selected* from the contextual menu or pressing `Ctrl+E`;
- Browse the command history from the input area using the `Up` and `Down` arrow keys and execute the command you want;
- `Ctrl+Shift+Space` to view the command history: double-clicking a row will execute the command. The *Command History* dialog can also be accessed from context menu of input area;
- Save and clear the command history. The history will be saved into the `console_history.txt` file under the active *user profile* folder;
- Type the following special commands:
  - `?` to show a help of the Python Console
  - `_api` to open QGIS C++ API documentation or `_api(object)` for a specific object documentation (in QGIS C++ API or Qt API documentation)
  - `_pyqgis` to open QGIS Python API documentation or `_pyqgis(object)` for a specific object documentation (in QGIS Python API or Qt API documentation)
  - `_cookbook` to open PyQGIS Cookbook.
  - `!` followed by a command to execute Shell commands from the Python Console. The console will start a subprocess, and forward its output to the Python Console Output. While the subprocess is running, the Python Console Input switches to STDIN mode and forwards entered character to the child process. This makes it possible to send confirmation when the child program asks for it. When the Console is in STDIN mode, pressing `Ctrl+C` kills the subprocess. It is also possible to affect the result of a command to a variable with the syntax `var = !cmd`

```
>>> !echo QGIS Rocks!
QGIS Rocks

>>> !gdalinfo --version
GDAL 3.6.2, released 2023/01/02

>>> !pip install black
# Install black python formatter using pip (if available)

>>> sql_formats = !ogrinfo --formats | grep SQL
>>> sql_formats
['SQLite -vector- (rw+v): SQLite / Spatialite', '  MSSQLSpatial -vector-␣
↪(rw+): Microsoft SQL Server Spatial Database', '  PostgreSQL -vector-␣
↪(rw+): PostgreSQL/PostGIS', '  MySQL -vector- (rw+): MySQL', '  PGDUMP -
↪vector- (w+v): PostgreSQL SQL dump']
```

Sfat: Reuse executed commands from the output panel

You can execute code snippets from the output panel by selecting some text and pressing `Ctrl+E`. No matter if selected text contains the interpreter prompt (`>>>`, `...`).

29.3.2 Editorul de Cod

Use the Show Editor button to enable the editor widget. It allows editing and saving Python files and offers advanced functionalities to manage your code (comment and uncomment code, check syntax, share the code via GitHub and much more). Main features are:

- Code completion, highlighting syntax and calltips for the following APIs:
  - Python
  - PyQGIS
  - PyQt5
  - QScintilla2
  - osgeo-gdal-ogr
- `Ctrl+Space` to view the auto-completion list.
- Sharing code snippets via *GitHub*.
- `Ctrl+4` Verificare sintaxă.
- Search bar (open it with the default Desktop Environment shortcut, usually `Ctrl+F`):
  - Use the default Desktop Environment shortcut to find next/previous (`Ctrl+G` and `Shift+Ctrl+G`);
  - Automatically find first match when typing in find box;
  - Set initial find string to selection when opening find;
  - Pressing `Esc` closes the find bar.
- Object inspector: a class and function browser;
- Go to an object definition with a mouse click (from Object inspector);
- Execute code snippets with the *Run Selected* command in contextual menu;
- Execute the whole script with the *Run Script* command (this creates a byte-compiled file with the extension `.pyc`).

Notă: Running partially or totally a script from the *Code Editor* outputs the result in the Console output area.

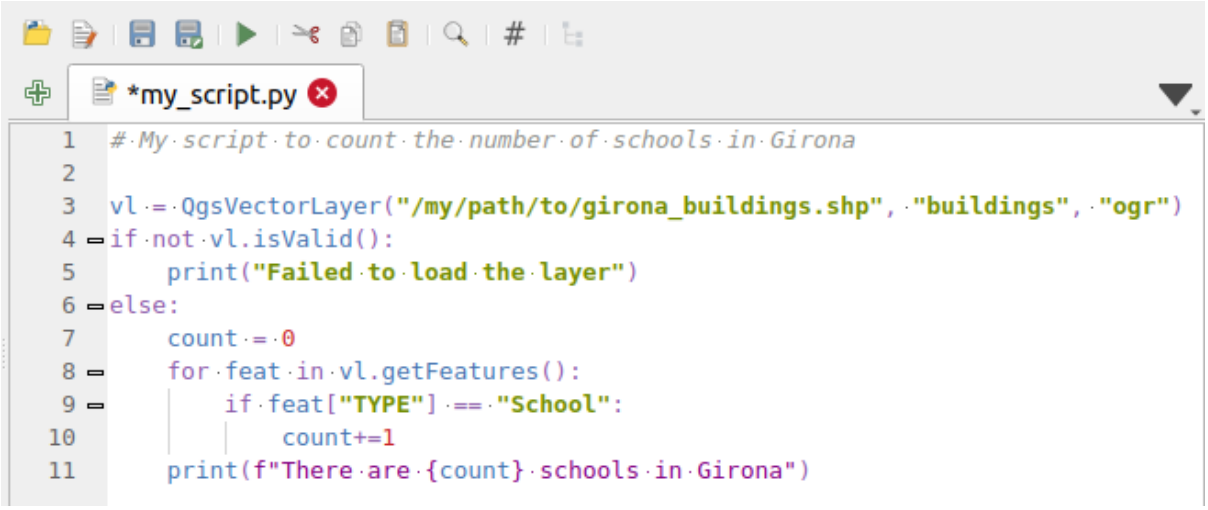


Fig. 29.16: The Python Console editor

Sfat: Save the options

To save the state of console's widgets you have to close the Python Console from the close button. This allows you to save the geometry to be restored to the next start.

CAPITOLUL 30

Asistență și Ajutor

30.1 Liste de discuții

QGIS este în curs de dezvoltare activă și, ca atare, nu va funcționa întotdeauna așa cum vă așteptați. Modul preferat de a obține ajutor este prin folosirea listei de discuții a utilizatorilor QGIS. Întrebările dvs. vor ajunge la un public mai larg, iar răspunsurile să fie în beneficiul altor persoane.

30.1.1 Utilizatori QGIS

Această listă de corespondență este utilizată pentru discuții despre QGIS în general, precum și pentru întrebări specifice privind instalarea și utilizarea sa. Vă puteți abona la lista de e-mail qgis-users accesând următorul link: https://lists.osgeo.org/mailman/listinfo/qgis-user

30.1.2 Dezvoltatori QGIS

If you are a developer facing problems of a more technical nature, you may want to join the qgis-developer mailing list. This list is also a place where people can chime in and collect and discuss QGIS related UX (User Experience) / usability issues. It's here: https://lists.osgeo.org/mailman/listinfo/qgis-developer

30.1.3 Echipa Comunității QGIS

This list deals with topics like documentation, context help, user guide, web sites, blog, mailing lists, forums, and translation efforts. If you would like to work on the user guide as well, this list is a good starting point to ask your questions. You can subscribe to this list at: https://lists.osgeo.org/mailman/listinfo/qgis-community-team

30.1.4 Traduceri QGIS

This list deals with the translation efforts. If you like to work on the translation of the website, manuals or the graphical user interface (GUI), this list is a good starting point to ask your questions. You can subscribe to this list at: https://lists.osgeo.org/mailman/listinfo/qgis-tr

30.1.5 Comitetul de Dirijare a Proiectului QGIS (PSC)

This list is used to discuss Steering Committee issues related to overall management and direction of QGIS. You can subscribe to this list at: https://lists.osgeo.org/mailman/listinfo/qgis-psc

30.1.6 Grupuri de utilizatori QGIS

In order to locally promote QGIS and contribute to its development, some QGIS communities are organized into QGIS User Groups. These groups are places to discuss local topics, organize regional or national user meetings, organize sponsoring of features… The list of current user groups is available at https://qgis.org/en/site/forusers/usergroups.html

Sunteți bineveniți să vă abonați la oricare dintre liste. Vă rugăm să vă aduceți și dumneavoastră aportul, răspunzând la întrebări și împărtășindu-vă experiențele.

30.2 Matrix / IRC

Matrix (https://matrix.org) is a project for decentralised chat. QGIS has a room accessible by the #qgis:osgeo.org alias (among others) which bridges to the libera.chat #qgis IRC channel.

To use Matrix:

1. Create an account (at matrix.org is easiest, but if you have an OSGeo account you can also use your OSGeo id as matrix id)
2. Install a client (Elements is easiest, but see Matrix clients for more), OR point your browser to https://matrix.to/#/#qgis:osgeo.org

To use IRC:

1. Install an IRC client
2. Connect to irc://irc.libera.chat/#qgis or point your browser to https://web.libera.chat/?channels=#qgis

30.3 Suport comercial

Commercial support for QGIS is also available. Check the website https://qgis.org/en/site/forusers/commercial_support.html for more information.

30.4 BugTracker

While the qgis-users mailing list is useful for general «How do I do XYZ in QGIS?»-type questions, you may wish to notify us about bugs in QGIS. You can submit bug reports using the QGIS bug tracker.

Please bear in mind that your bug may not always enjoy the priority you might hope for (depending on its severity). Some bugs may require significant developer effort to remedy, and the manpower is not always available for this.

Feature requests can be submitted as well using the same ticket system as for bugs. Please make sure to select the type `Feature request`.

If you have found a bug and fixed it yourself, you can submit a Pull Request on the Github QGIS Project.

Pentru mai multe detalii, vizitați Erori, funcționalități și probleme și submit_patch

30.5 Blog

The QGIS community also runs a weblog at https://plugins.qgis.org/planet/, which has some interesting articles for users and developers. Many other QGIS blogs exist, and you are invited to contribute with your own QGIS blog!

30.6 Plugin-uri

The website https://plugins.qgis.org is the official QGIS plugins web portal. Here, you find a list of all stable and experimental QGIS plugins available via the «Official QGIS Plugin Repository».

30.7 Wiki

Lastly, we maintain a WIKI web site at https://github.com/qgis/QGIS/wiki where you can find a variety of useful information relating to QGIS development, release plans, links to download sites, message-translation hints and more. Check it out, there are some goodies inside!

CAPITOLUL 31

Contribuții

QGIS este dezvoltat de o echipă dedicată de voluntari și organizații. Ne străduim să fim o comunitate primitoare pentru persoanele de orice rasă, credință, sex și statut social. În orice moment, vă puteți alătura nouă <https://qgis.org/en/site/getinvolved/index.html>`_.

31.1 Autori

Mai jos sunt enumerate persoanele care își dedică timpul și energia pentru a scrie, revizui și actualiza întreaga documentație QGIS.

Tim Sutton	Yves Jacolin	Jacob Lanstorp	Gary E. Sherman	Richard Duivenvoorde
Tara Athan	Anita Graser	Arnaud Morvan	Gavin Macaulay	Luca Casagrande
K. Koy	Hugo Mercier	Akbar Gumbira	Marie Silvestre	Jürgen E. Fischer
Fran Raga	Eric Goddard	Martin Dobias	Diethard Jansen	Saber Razmjooei
Ko Nagase	Nyall Dawson	Matthias Kuhn	Andreas Neumann	Harrissou Sant-anna
Manel Clos	David Willis	Larissa Junek	Paul Blottière	Sebastian Dietrich
Chris Mayo	Stephan Holl	Magnus Homann	Bernhard Ströbl	Alessandro Pasotti
N. Horning	Radim Blazek	Joshua Arnott	Luca Manganelli	Marco Hugentobler
Andre Mano	Mie Winstrup	Frank Sokolic	Vincent Picavet	Jean-Roc Morreale
Andy Allan	Victor Olaya	Tyler Mitchell	René-Luc D'Hont	Marco Bernasocchi
Ilkka Rinne	Werner Macho	Chris Berkhout	Nicholas Duggan	Jonathan Willitts
David Adler	Lars Luthman	Brendan Morely	Raymond Nijssen	Carson J.Q. Farmer
Jaka Kranjc	Mezene Worku	Patrick Sunter	Steven Cordwell	Stefan Blumentrath
Andy Schmid	Vincent Mora	Alexandre Neto	Hien Tran-Quang	Alexandre Busquets
João Gaspar	Tom Kralidis	Alexander Bruy	Paolo Cavallini	Milo Van der Linden
Peter Ersts	Ujaval Gandhi	Dominic Keller	Giovanni Manghi	Maximilian Krambach
Anne Ghisla	Dick Groskamp	Uros Preloznik	Stéphane Brunner	QGIS Korean Translator
Zoltan Siki	Håvard Tveite	Matteo Ghetta	Salvatore Larosa	Konstantinos Nikolaou
Tom Chadwin	Larry Shaffer	Nathan Woodrow	Martina Savarese	Godofredo Contreras
Astrid Emde	Luigi Pirelli	Thomas Gratier	Giovanni Allegri	GiordanoPezzola
Paolo Corti	Tudor Bărăscu	Maning Sambale	Claudia A. Engel	Yoichi Kayama
Otto Dassau	Denis Rouzaud	Nick Bearman	embelding	ajazepk
Ramon	Andrei	zstadler	icephale	Rosa Aguilar
Patrice Pineault	Jörn Gutzeit	Felix Feckler	Benoît de Mezzo	Étienne Trimaille

continues on next page

Tabelul 31.1 – continuare din pagina precedentă

Andrea Giudiceandrea	Julien Cabieces	roya0045	Sebastian Gutwein	Jessica Veenstra
Ryan Welfle	Martin Pergler	Ivan Ivanov	muranamihdk	Loïc Bartoletti
Tomasz Taraś	Ian Maddaus	Jürnjakob Dugge	Roman Bug	Damiano Lombardi
Marc Ducobu	Philip Albrecht	Dennis Milechin	Cody Martin	Savinaud Mickaël
Stefan Uhrig	Ariadni-Karolina Alexiou	Björn Hinkeldey	Benjamin Riley	MorriganR
Thayer Young	Shane Carey	Ian Turton	Emma Hain	Germán Carrillo
Jakob Miksch	Nicolas Boisteault	Bertrand Rix	Jorge Rosales	

31.2 Traducători

QGIS este o aplicație multilingvă care beneficiază totodată de o documentație tradusă în câteva limbi. Multe altele sunt disponibile și vor fi lansate imediat ce vor atinge un procent rezonabil de traducere. Dacă doriți să ajutați la îmbunătățirea unei limbi sau să solicitați una nouă, consultați https://qgis.org/en/site/getinvolved/index.html.

Traducerile actuale sunt posibile datorită următorilor:

Limba	Contribuții
Bahasa Indonesia	Emir Hartato, I Made Anombawa, Januar V. Simarmata, Muhammad Iqnaul Haq Siregar, Trias Aditya
Chineză (Tradițională)	Calvin Ngei, Zhang Jun, Richard Xie
Chineză (Simplificată)	Xu Baocai
Olandeză	Carlo van Rijswijk, Dick Groskamp, Diethard Jansen, Raymond Nijssen, Richard Duivenvoorde, Willem Hoffman
Finlandeză	Matti Mäntynen, Kari Mikkonen
Franceză	Arnaud Morvan, Augustin Roche, Didier Vanden Berghe, Dofabien, Etienne Trimaille, Francis Gasc, Harrissou Sant-anna, Jean-Roc Morreale, Jérémy Garniaux, Loïc Buscoz, Lsam, Marc-André Saia, Marie Silvestre, Mathieu Bossaert, Mathieu Lattes, Mayeul Kauffmann, Médéric Ribreux, Mehdi Semchaoui, Michael Douchin, Nicolas Boisteault, Nicolas Rochard, Pascal Obstetar, Robin Prest, Rod Bera, Stéphane Henriod, Stéphane Possamai, sylther, Sylvain Badey, Sylvain Maillard, Vincent Picavet, Xavier Tardieu, Yann Leveille-Menez, yoda89
Galiciană	Xan Vieiro
Germană	Jürgen E. Fischer, Otto Dassau, Stephan Holl, Werner Macho
Hindi	Harish Kumar Solanki
Italiană	Alessandro Fanna, Anne Ghisla, Flavio Rigolon, Giuliano Curti, Luca Casagrande, Luca Delucchi, Marco Braida, Matteo Ghetta, Maurizio Napolitano, Michele Beneventi, Michele Ferretti, Roberto Angeletti, Paolo Cavallini, Stefano Campus
Japoneză	Baba Yoshihiko, Minoru Akagi, Norihiro Yamate, Takayuki Mizutani, Takayuki Nuimura, Yoichi Kayama
Coreeană	Capitolul OSGeo Coreean
Poloneză	Andrzej Świąder, Borys Jurgiel, Ewelina Krawczak, Jakub Bobrowski, Mateusz Łoskot, Michał Kułach, Michał Smoczyk, Milena Nowotarska, Radosław Pasiok, Robert Szczepanek, Tomasz Paul
Portugheză	Alexandre Neto, Duarte Carreira, Giovanni Manghi, João Gaspar, Joana Simões, Leandro Infantini, Nelson Silva, Pedro Palheiro, Pedro Pereira, Ricardo Sena
Portugheză (Brazilia)	Arthur Nanni, Felipe Sodré Barros, Leônidas Descovi Filho, Marcelo Soares Souza, Narcélio de Sá Pereira Filho, Sidney Schaberle Goveia
Română	Alex Bădescu, Bogdan Păcurar, Georgiana Ioanovici, Ionuț Iosifescu-Enescu, Sorin Călinică, Tudor Bărăscu
Rusă	Alexander Bruy, Artem Popov
Spaniolă	Carlos Dávila, Diana Galindo, Edwin Amado, Gabriela Awad, Javier César Aldariz, Mayeul Kauffmann, Fran Raga
Ucraineană	Alexander Bruy

31.3 Statistica traducerilor

Mai jos este prezentată activitatea de traducere pentru QGIS 3.34 cu Suport pe Termen Lung. Doar limbile care au atins 5% până la momentul lansării versiunii sunt publicate aici.

Ultima actualizare: 2024-02-27

Numărul de expresii	Numărul limbilor	Procentajul global al traducerii
29146	**58**	**14.92%**

Limba	Procentajul traducerii (%)	Limba	Procentajul traducerii (%)	Limba	Procentajul traducerii (%)
Albaneză	1.12	Arabă	4.63	Azeră	0.87
Bască	1.89	Bengaleză	1.06	Bulgară	3.41
Birmaneză	0.97	Catalană	2.03	Chineză Simplificată	29.67
Chineză Tradițională	2.2	Croată	0.98	Cehă	6.74
Daneză	1.59	Olandeză	100.0	Estonă	2.25
Finlandeză	2.22	Franceză	85.75	Galiciană	1.36
Georgiană	0.98	Germană	22.93	Greacă	1.25
Ebraică	1.68	Hindi	1.19	Maghiară	19.22
Igbo	0.87	Indoneziană	3.64	Italiană	100.0
Japoneză	99.93	Kabyle	0.97	Coreeană	86.8
Lituaniană	11.42	Macedoneană	0.99	Malay	0.91
Malayalam	0.97	Marathi	1.01	Mongolă	0.98
N'ko	2.53	Norwegian Bokmål	3.73	Panjabi (Punjabi)	0.0
Persană	0.69	Poloneză	2.75	Portugheză (Brazilia)	70.65
Portugheză (Portugalia)	8.61	Română	34.92	Rusă	15.47
Sârbă	0.13	Slovacă	2.44	Slovenă	3.66
Spaniolă	99.96	Suedeză	1.95	Tagalog	0.97
Tamilă	1.41	Telugu	0.89	Thai	0.98
Turcă	3.37	Ucraineană	3.68	Urdu	0.86
Vietnameză	1.11				

CAPITOLUL 32

Documente adiționale

32.1 Anexa A: Licența Publică Generală GNU

Versiunea 2, Iunie 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Oricui îi este permisă copierea şi distribuirea de copii identice ale acestui document, dar fără modificarea lui.

Preambul

Licențele majorității programelor sunt concepute pentru a vă priva de libertatea de a le partaja și de a le modifica. Prin contrast, intenția Licenței Publice Generale GNU este de a vă garanta libertatea de a distribui și modifica programele gratuite - pentru a se asigura că programele sunt gratuite pentru toți utilizatorii. Această Licență Publică Generală se aplică majorității programelor aparținând Free Software Foundation precum și tuturor celorlalte programe ai căror autori decid să o folosească. (În schimb, alte programe aparținând Free Software Foundation se află sub Licența Publică Generală GNU pentru Biblioteci.) De asemenea, ea poate fi aplicată și programelor dumneavoastră.

Când vorbim de software gratuit, ne referim la libertate, și nu la preț. Licențele noastre sunt concepute să vă garanteze libertatea de a distribui copii ale programelor gratuite (și de a oferi acest serviciu contra cost, dacă doriți), de a primi sau de a obține codul sursă, dacă doriți, de a schimba programul sau a folosi porțiuni din el în noi programe libere, și de a ști că puteți face toate aceste lucruri.

Pentru a vă proteja drepturile, trebuie să impunem restricții împotriva oricui ar încerca să vă conteste aceste drepturi sau să vă ceară să renunțați la ele. Aceste restricții implică anumite responsabilități pentru dumneavoastră dacă distribuiți copii ale programelor, sau dacă le modificați.

De exemplu, dacă distribuiți copii ale unui astfel de program, indiferent dacă o faceți gratuit sau contra unei sume de bani, trebuie să cedați beneficiarilor toate drepturile pe care le aveți dumneavoastră. Trebuie să vă asigurați că ei primesc, sau pot primi, codul sursă. În plus, trebuie să le arătați care sunt termenii în care primesc programul, pentru a ști care le sunt drepturile.

Vă protejăm drepturile în două rânduri: (1) prin stabilirea drepturilor de autor pentru program, și (2) prin această Licență care vă conferă dreptul legal de a copia, distribui și/sau modifica programul.

De asemenea, pentru propria noastră protecție cât și pentru cea a autorilor, vrem să ne asigurăm că toată lumea înțelege că nu există niciun fel de garanție pentru acest program gratuit. Dacă programul este modificat de altcineva

și distribuit mai departe, vrem ca beneficiarii programului să știe că ceea ce au nu este originalul, în așa fel încât nicio problemă introdusă de altcineva nu va avea un efect negativ asupra reputației autorilor inițiali.

În final, orice program liber este în mod constant amenințat de patentele software. Vrem să evităm pericolul ca cei ce redistribuie programele libere să obțină patente, practic transformând programul într-unul proprietar. Pentru a preveni aceasta, facem clară poziția noastră conform căreia orice patent trebuie acordat fie în așa fel, încât să poată fi licențiat și fără restricții pentru uzul gratuit al oricui, fie să nu necesite licență.

Termenii și condițiile exacte de copiere, distribuire și modificare sunt specificate în următoarele paragrafe. TERMENI ȘI CONDIȚII PENTRU COPIERE, DISTRIBUIRE ȘI MODIFICARE

0. Această Licență se aplică oricărui program sau proiect ce conține o mențiune a deținătorului drepturilor de autor spunând că poate fi distribuit în termenii acestei Licențe Publice Generale. Prin „Program", în continuare, vom înțelege orice asemenea program sau proiect, iar prin „lucru bazat pe Program" vom înțelege fie programul fie orice alt proiect derivat din Program, conform cu legea drepturilor de autor: adică, un proiect ce conține Programul sau porțiuni din el, fie în forma originală fie modificată și/sau tradusă în altă limbă. (În restul acestui document traducerile vor fi incluse fără restricții în termenul „modificare".) Fiecare persoană autorizată de această Licență va fi desemnată prin termenul „dumneavoastră".

   Activitățile care nu sunt de copiere, distribuire și modificare nu sunt acoperite de această Licență; ele sunt în afara scopului ei. Activitatea de executare a Programului nu este restricționată, iar rezultatul Programului este acoperit de licență doar în cazul în care conținutul său constituie un proiect bazat pe Program (independent de faptul că a fost obținut prin rularea Programului). Măsura în care acest lucru este adevărat depinde de natura Programului.

1. Puteţi copia şi distribui copii nemodificate ale codului sursă al Programului în forma în care îl primiți, prin orice mediu, cu condiția să specificaţi vizibil pe fiecare copie autorul şi lipsa oricărei garanții, să păstrați intacte toate notele referitoare la această Licenţă şi la absenţa oricărei garanții şi să distribuiţi o copie a acestei Licenţe cu fiecare copie a Programului.

   Puteţi pretinde o retribuție financiară pentru actul fizic de transfer al unei copii, şi puteți oferi garanție contra cost.

2. Puteți efectua modificări asupra copiilor Programului, sau asupra oricăror porțiuni ale sale, creând astfel un proiect bazat pe Program, iar copierea și distribuirea unor asemenea modificări sau proiecte se pot face conform termenilor Secțiunii 1 de mai sus, doar dacă toate condițiile următoarele sunt îndeplinite:

   a) Trebuie ca fișierele modificate să conțină notițe proeminent vizibile, care să menționeze faptul că dumneavoastră le-ați modificat, precum și data fiecărei modificări.

   b) Trebuie ca orice proiect pe care îl distribuiți sau publicați, care în întregime sau în parte conține sau este derivat din Program, sau din oricare parte a acestuia, să poată fi licențiat gratuit și în întregime tuturor părților terțe, în termenii acestei Licențe.

   c) Dacă programul modificat citește comenzi în mod interactiv, trebuie să îl modificați în așa fel încât, atunci când este pornit în mod interactiv, să afișeze un mesaj referitor la drepturile de autor, precum și o notă în care se menționează lipsa oricărei garanții (sau faptul că dumneavoastră oferiți o garanție). De asemenea, trebuie specificat faptul că utilizatorii pot redistribui programul în aceste condiții, precum și o explicație a modalității în care poate fi obținut textul acestei Licențe. (Excepție: dacă Programul este interactiv, dar nu afișează în mod normal un asemenea mesaj, nu este necesar ca proiectul bazat pe Program să afișeze un mesaj.)

   Aceste cerințe se aplică Programului modificat în întregime. Dacă pot fi identificate secțiuni ale proiectului care nu sunt derivate din Program, și pot fi considerate de sine stătătoare, atunci această Licență și termenii săi nu se aplică acelor secțiuni când sunt distribuite ca proiecte separate. Când distribuiți aceleași secțiuni ca parte a unui întreg care este un proiect bazat pe Program, distribuirea întregului proiect trebuie să fie făcută în acord cu termenii acestei Licențe, ale cărei permisiuni pentru alte licențe se extind asupra întregului, și deci asupra fiecărei secțiuni în parte, indiferent de autor.

   Astfel, nu este în intenția acestei secțiuni să pretindă drepturi sau să conteste drepturile dumneavoastră asupra unui proiect efectuat în întregime de dumneavoastră. Intenția este de a exercita dreptul de a controla distribuția proiectelor derivate sau colective bazate pe Program.

   În plus, pura agregare pe un mediu de stocare sau distribuție cu Programul (sau cu un proiect bazat pe Program) a unui alt proiect, care nu este bazat pe Program, nu aduce acel proiect sub incidența acestei Licențe.

3. Puteți copia și distribui Programul (sau un proiect bazat pe el, conform Secțiunii 2) în format obiect sau executabil conform termenilor Secțiunilor 1 și 2 de mai sus, cu condiția să îndepliniți una dintre condițiile de mai jos:

   a) Să îl oferiți însoțit de codul sursă corespunzător, în format citibil de către mașină, care trebuie să fie distribuit în termenii Secțiunilor 1 și 2 de mai sus, pe un mediu de distribuție uzual transportului de software; sau

   b) Să îl oferiți însoțit de o ofertă scrisă, validă pentru cel puțin trei ani, pentru o taxă care să nu depășească costul fizic al efectuării distribuției sursei, de a oferi o copie completă, în format citibil de către mașină, a codului sursă, distribuit în termenii Secțiunilor 1 și 2 de mai sus, pe un mediu de distribuție uzual transportului de software; sau

   c) Să îl oferiți însoțit de informația pe care ați primit-o referitoare la oferta de a distribui codul sursa corespunzător. (Această alternativă este permisă numai pentru distribuiri necomerciale și doar dacă ați primit programul în format obiect sau executabil împreună cu această ofertă, în conformitate cu Subsecțiunea b de mai sus.)

   Codul sursă al unui proiect este forma preferată în care se fac modificări asupra proiectului. Pentru un proiect executabil, codul sursă complet înseamnă codul sursă al tuturor modulelor pe care le conține, împreună cu toate fișierele asociate conținând definiții ale interfețelor și scripturile folosite pentru a controla compilarea și instalarea executabilului. Cu toate acestea, ca o excepție, nu este obligatorie distribuirea împreună cu codul sursă a acelor componente care sunt în mod normal distribuite (în format sursă sau binar) cu componentele majore (compilator, nucleu, etc.) ale sistemului de operare sub care rulează executabilul, exceptând situația în care acea componentă acompaniază executabilul.

   Dacă distribuția executabilului sau codului obiect este făcută prin oferirea permisiunii de copiere dintr-un loc dedicat, atunci oferirea permisiunii de copiere a codului sursă din același loc este considerată distribuire a codului sursă, chiar dacă beneficiarul nu este obligat să copieze codul sursă împreuna cu codul obiect.

4. Nu puteți copia, modifica, sub-autoriza sau distribui Programul decât așa cum este prevăzut în această Licență. Orice încercare de a copia, modifica, sub-autoriza sau distribui Programul în alți termeni va duce la anularea drepturilor ce vă revin conform acestei Licențe. Cu toate acestea, nu vor fi anulate drepturile celor ce au primit copii sau drepturi de la dumneavoastră conform cu această Licență, atâta timp cât rămân în conformitate cu ea.

5. Nu sunteți obligați să acceptați această Licență, deoarece nu ați semnat-o. Cu toate acestea, numai această Licență vă permite să modificați Programul sau proiectele derivate din el. Aceste acțiuni sunt interzise prin lege dacă nu acceptați această Licență. În consecință, prin modificarea sau distribuirea Programului (sau a oricărui proiect bazat pe Program), indicați în mod implicit acceptarea acestei Licențe și a tuturor termenilor și condițiilor de copiere, distribuire sau modificare a Programului sau proiectelor bazate pe el.

6. De fiecare dată când redistribuiți Programul (sau orice proiect bazat pe Program), beneficiarul primește o licență de la licențiatorul original care îi permite să copieze, distribuie sau modifice Programul în aceiași termeni și condiții. Nu puteți impune nici o restricție adițională asupra exercitării drepturilor pe care destinatarul le primește prin această Licență. Nu sunteți responsabil cu impunerea respectării acestei Licențe de către o terță parte.

7. În cazul în care, ca o consecință a unei decizii judecătorești, sau pretinsă încălcare a unui patent, sau pentru orice altă cauză (nu neapărat limitată la chestiuni legate de patente), vi se impun condiții (prin hotărâre judecătorească, înțelegere sau alte mijloace) care contravin condițiilor acestei Licențe, acest lucru nu vă permite nerespectarea condițiilor acestei Licențe. Dacă nu puteți face în așa fel încât să satisfaceți simultan obligațiile din această Licență și alte obligații pertinente, atunci, ca o consecință, vă este interzisă distribuirea Programului. De exemplu, dacă o autorizație de folosire a unui patent nu vă permite redistribuirea gratuită a Programului de către oricine îl primește de la dumneavoastră, direct sau indirect, atunci singurul mod în care puteți satisface simultan aceste condiții și Licența de față este să nu distribuiți Programul în niciun fel.

   Dacă vreo porțiune a acestei secțiuni este invalidată sau de neaplicat în anumite circumstanțe, restul secțiunii continuă să se aplice, iar secțiunea în întregime se aplică în toate celelalte circumstanțe.

   Nu este în intenția acestei secțiuni să vă determine să încălcați vreun patent sau alte pretenții de drepturi de proprietate, sau să contestați valabilitatea oricăror asemenea pretenții; această secțiune are ca scop unic protejarea integrității sistemului de distribuire de programe libere, care este implementat prin licențe publice. Multe persoane au contribuit generos la spectrul larg de programe distribuite prin acest sistem, bazându-se

pe aplicarea sa consistentă; este la latitudinea autorului/donatorului să decidă dacă este dispus să distribuie programe prin orice alt sistem, și o persoană autorizată să folosească acele programe nu poate impune acea decizie.

Această secțiune este dedicată clarificării a ceea ce este considerat a fi o consecință a restului acestei Licențe.

8. Dacă distribuția și/sau folosirea Programului este restricționată în anumite țări, din cauza patentelor, sau din cauza unor interfețe aflate sub incidența unor drepturi de autor restrictive, deținătorul drepturilor de autor ce plasează Programul sub această Licență poate adăuga o limitare geografica a distribuirii ce exclude acele țări, în așa fel încât distribuirea să fie permisă doar în, sau între, țările care nu sunt excluse. Într-un asemenea caz, Licența încorporează această limitare ca și cum ar fi scrisă în corpul acestei Licențe.

9. Free Software Foundation poate publica din când în când noi versiuni și/sau versiuni revăzute, ale Licenței Publice Generale. Asemenea versiuni noi vor fi similare în spirit versiunii prezente, dar pot diferi în anumite detalii, pentru a adresa noi probleme sau situații.

   Fiecărei versiuni îi este asociat un număr unic. Dacă programul specifică faptul că i se aplică o versiune a acestei Licențe și „orice altă versiune ulterioară”, aveți opțiunea de a urma termenii și condițiile acelei versiuni, sau ai oricărei versiuni ulterioare publicate de Free Software Foundation. Dacă Programul nu specifică un număr de versiune, puteți alege orice versiune publicată vreodată de Free Software Foundation.

10. Dacă doriți să incorporați părți ale Programului în alte programe libere ale căror condiții de distribuție sunt diferite, cereți permisiunea autorului. Pentru programe ale căror drepturi de autor aparțin Free Software Foundation, cereți permisiunea de la Free Software Foundation; uneori facem excepții pentru aceasta. Decizia noastră va fi ghidată de cele două scopuri de a prezerva statutul liber al tuturor proiectelor derivate din programele noastre libere și de a promova distribuirea și refolosirea programelor în general.

NICIO GARANȚIE

11. DEOARECE PROGRAMUL ESTE OFERIT SUB O LICENȚĂ CE NU IMPLICĂ NICI UN COST, NU EXISTĂ NICIO GARANȚIE PENTRU PROGRAM, ÎN MĂSURA PERMISĂ DE LEGILE CE SE APLICĂ. EXCEPTÂND SITUAȚIILE UNDE ESTE SPECIFICAT ALTFEL ÎN SCRIS, DEȚINĂTORII DREPTURILOR DE AUTOR ȘI/SAU ALTE PĂRȚI IMPLICATE OFERĂ PROGRAMUL „ÎN FORMA EXISTENTĂ” FĂRĂ NICIO GARANȚIE DE NICIUN FEL, EXPLICITĂ SAU IMPLICITĂ, INCLUZÂND, DAR FĂRĂ A FI LIMITATĂ LA, GARANȚII IMPLICITE DE VANDABILITATE ȘI CONFORMITATE UNUI ANUMIT SCOP. VĂ ASUMAȚI ÎN ÎNTREGIME RISCUL ÎN CEEA CE PRIVEȘTE CALITATEA ȘI PERFORMANȚA ACESTUI PROGRAM. ÎN CAZUL ÎN CARE PROGRAMUL SE DOVEDEȘTE A FI DEFECT, VĂ ASUMAȚI ÎN ÎNTREGIME COSTUL TUTUROR SERVICIILOR, REPARAȚIILOR ȘI CORECȚIILOR NECESARE.

12. ÎN NICIO SITUAȚIE, EXCEPTÂND CAZURILE ÎN CARE ESTE CERUT DE LEGEA APLICABILĂ SAU CA REZULTAT AL UNEI ÎNȚELEGERI SCRISE, UN DEȚINĂTOR AL DREPTURILOR DE AUTOR, SAU ORICE ALTĂ PARTE CARE POATE MODIFICA ȘI/SAU REDISTRIBUI PROGRAMUL CONFORM PERMISIUNILOR DE MAI SUS NU VA FI FĂCUT RĂSPUNZĂTOR PENTRU PAGUBELE DUMNEAVOASTRĂ, INCLUSIV CELE GENERALE, SPECIALE, ÎNTÂMPLĂTOARE SAU REZULTANTE, APĂRUTE DIN FOLOSIREA SAU INABILITATEA DE A FOLOSI PROGRAMUL (INCLUZÂND, DAR FĂRĂ A FI LIMITAT LA PIERDEREA SAU DETERIORAREA DATELOR, SAU PIERDERILE SUFERITE DE DUMNEAVOASTRĂ SAU TERȚE PERSOANE, SAU O INCAPACITATE A PROGRAMULUI DE A INTEROPERA CU ALTE PROGRAME), CHIAR DACĂ DEȚINĂTORUL SAU TERȚA PARTE A FOST PREVENITĂ ASUPRA POSIBILITĂȚII UNOR ASEMENEA PAGUBE.

Excepția QGIS Qt de la GPL

În plus, ca o excepție specială, Echipa de Dezvoltare QGIS oferă permisiunea de a lega codul acestui program cu biblioteca Qt, inclusiv, dar fără a se limita la următoarele versiuni (atât gratuite și comerciale): Qt/Non-commerical Windows, Qt/Windows, Qt/X11, Qt/Mac, și Qt/Embedded (sau cu versiuni modificate ale Qt care folosesc aceeași licență ca și Qt) și de a distribui combinații ale celor două legate. Trebuie să vă supuneți Licenței Generale Publice GNU în toate privințele pentru toate codul utilizat, altul decât Qt. Dacă modificați acest fișier, puteți prelungi această excepție de la versiunea de fișier, dar nu aveți obligația să faceți acest lucru. Dacă nu doriți acest lucru, ștergeți această declarație de excepție din versiunea dumneavoastră.

32.2 Anexa B: Licența GNU pentru Documentația Liberă

Versiunea 1.3, 3 Noiembrie 2008

Copyright 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc

https://www.fsf.org/

Oricui îi este permisă copierea şi distribuirea de copii identice ale acestui document, dar fără modificarea lui.

Preambul

Scopul acestei Licenţe este de a conferi „gratuitate" unui manual, colecții de texte, sau altui document funcțional și folositor, în sensul libertății: de a asigura tuturor permisiunea de copiere şi redistribuire, cu sau fără modificări, în scopuri comerciale şi necomerciale. Ca scop secundar, această Licenţă rezervă autorului şi editorului dreptul de a fi creditaţi pentru munca lor, atât timp cât nu sunt responsabili pentru modificările efectuate de către alţii.

Acestă Licență conferă un fel de „obligații", ceea ce înseamnă că lucrările derivate dintr-un document trebuie să fie și ele libere, la rândul lor. Această Licență este inspirată de Licența Publică Generală GNU, care este o licență similară, concepută pentru a acoperi softul liber.

Am creat această Licenţă pentru a fi de folos manualelor pentru softul liber, deoarece un soft liber necesită o documentaţie liberă: un program trebuie însoţit de manuale care oferă aceeaşi libertate de folosire ca şi softul. Acestă Licenţă nu este limitată, însă, la manualele pentru soft; ea poate fi folosită pentru textul oricărei lucrări, indiferent de subiect sau de modul de publicare. Această Licenţă este recomandată în principal pentru lucrări care servesc drept referinţă sau au fost scrise în scop de instruire.

1. APLICABILITATE ŞI DEFINIŢII

Această Licenţă se aplică oricărei lucrări sau manual, în orice mediu, care conţine o notă, inclusă de către deţinătorul dreptului de autor, care permite distribuţia în termenii acestei Licenţe. Această notă conferă dreptul universal, fără indemnizaţie şi nelimitat ca durată de a folosi lucrarea în condiţiile de faţă. Termenul **Document**, de mai jos, se referă la un astfel de manual sau lucrare. Orice membru din public este un beneficiar al acestei Licenţe şi va fi desemnat prin termenul **Dvs.**. Se consideră, în mod automat, că aţi acceptat termenii acestei Licenţe, în urma copierii, modificării sau distribuirii unei lucrări într-un mod care necesită permisiunea autorului, în condițiile legii drepturilor de autor.

O „**Versiune Modificată**" a Documentului este orice lucrare conţinând Documentul sau o porţiune de-a lui, copiată identic sau cu modificări şi/sau tradusă într-o altă limbă.

O „**Secţiune Secundară**" este o anexă cu titlu, sau o secţiune menţionată în cuprins care are ca scop exclusiv descrierea relaţiei editorilor sau a autorilor Documentului cu subiectul Documentului (sau cu aspecte conexe) şi care nu conţine referiri directe la subiectul Documentului. (Astfel, dacă Documentul este în parte manual de matematică, o Secţiune Secundară nu poate conţine deloc explicaţii matematice.) Poate exista doar o conexiune istorică cu subiectul și cu problemele înrudite cu subiectul, ori pot fi prezentate puncte de vedere legale, comerciale, filozofice, etice sau politice legate de acesta.

„**Secţiunile Neschimbabile**" sunt anumite Secţiuni Secundare ale căror titluri sunt specificate ca fiind acele titluri de Secţiuni Neschimbabile din nota ce permite distribuţia Documentului sub acoperirea acestei Licenţe. Dacă o secţiune nu este conformă cu definiţia de mai sus a unei Secţiuni Secundare atunci ea nu poate fi desemnată ca fiind Neschimbabilă. Documentul poate să nu conţină Secţiuni Neschimbabile. Dacă Documentul nu specifică vreo Secţiune Neschimbabilă atunci se consideră că nu există nici una.

„**Textele De Copertă**" sunt anumite pasaje scurte de text care sunt listate ca Texte Pentru Coperta I sau ca Texte Pentru Coperta IV în nota care specifică distribuirea Documentului sub acoperirea acestei Licenţe. Un Text Pentru Coperta I poate avea cel mult 5 cuvinte, iar un Text Pentru Coperta IV poate avea cel mult 25 de cuvinte.

O copie „**Transparentă**" a Documentului este o copie în format electronic, reprezentată într-un format ale cărui specificaţii sunt disponibile publicului, fiind uşor de modificat cu ajutorul unui editor de text generic sau (pentru imagini compuse din pixeli) cu un editor grafic generic ori (pentru desene) cu un editor larg răspândit de grafică vectorială, şi care poate fi folosit ca intrare în procesoarele de text sau de transformare automată în diverse formate adecvate ca intrare pentru procesoarele de text. O copie făcută într-un format de fişier Transparent dar care, prin prezenţa sau absenţa anumitor elemente specifice formatului, descurajează sau împiedică modificările ulterioare, nu reprezintă o copie Transparentă. Un format de imagine nu este Transparent dacă este folosit pentru a reprezenta o cantitate substanţială de text. O copie care nu este „Transparentă" se numește „Opacă"

Exemple de formate compatibile cu copiile Transparente: textul ASCII fără marcaje, formatul de intrare Texinfo, formatele de intrare LaTeX, SGML şi XML folosind un DTD public, HTML simplu şi standard, fişierele PostScript şi PDF modificabile. Exemple de formate Transparente pentru imagine: PNG, XCF şi JPG. Formatele Opace includ formate de text ce pot fi citite şi editate doar de procesoare de text proprietare, SGML şi XML pentru care DTD-ul şi/sau uneltele de procesare nu sunt disponibile, HTML generat automat, documentele PostScript şi PDF produse de diverse procesoare de text doar în scopul printării/afişării.

„**Pagina de Titlu**" înseamnă, pentru o carte tipărită, pagina cu titlul şi paginile următoare, necesare pentru a prezenta, lizibil, materialul care trebuie tipărit, conform acestei Licenţe, pe Pagina de Titlu. Pentru lucrări care nu au o pagină cu titlu propriu-zisă, „Pagina de Titlu" este textul aflat lângă principala apariţie a titlului lucrării, precedând începutul corpului Documentului.

„**Editorul**" reprezintă orice persoană sau entitate care distribuie copii ale documentului pentru public.

O secţiune „Numită XYZ" este o subunitate a Documentului, al cărei titlu este, fie XYZ, fie conţine XYZ în paranteze, după textul care traduce XYZ în altă limbă. (Aici XYZ înlocuieşte nume specifice ce vor fi menţionate mai jos, ca de exemplu „Mulţumiri", „Dedicaţii", „Giruri" sau „Istorie".) Pentru a „Păstra Titlul" unei astfel de secţiuni atunci când modificaţi Documentul înseamnă că va rămânea o secțiune „Numită XYZ", conform acestei definiţii.

Documentul poate include Limitări de Responsabilitate ataşate notificării care afirmă că această Licenţă se aplică Documentului. Aceste se consideră a fi incluse prin referinţă în această Licenţă, dar numai cu privire la limitările de responsabilitate: orice alte implicaţii pe care aceste Limitări de Responsabilitate le-ar putea avea sunt nule şi nu au nici un efect asupra înţelesului acestei Licenţe.

2. COPII IDENTICE

Puteţi copia şi distribui Documentul pe orice mediu, fie comercial sau necomercial, atâta timp cât această Licenţă, notificările de drepturi de autor şi notificarea de licenţă care spune că această Licenţă se aplică acestui Document, sunt reproduse în toate copiile, şi atâta timp cât nu adăugaţi nici un fel de altă condiţie în afară de cele prezente în această Licenţă. Nu aveţi dreptul să luaţi măsuri tehnice de a obstrucţiona sau controla citirea sau recopierea copiilor pe care le faceţi sau le distribuiţi. Aveţi totuşi dreptul să acceptaţi compensaţii în schimbul copiilor. Dacă distribuiţi un număr suficient de mare de copii, atunci trebuie să respectaţi şi condiţiile din secţiunea 3.

Aveți, de asemenea, dreptul să împrumutați copii în aceleași condiții ca cele de mai sus, și aveți dreptul să afișați copii.

3. COPIEREA ÎN CANTITĂȚI MARI

Dacă publicați copii tipărite (sau copii în medii care folosesc de obicei coperți tipărite) ale Documentului, în număr mai mare de 100 și dacă notificarea de licență a Documentului cere Texte de Copertă, trebuie să includeți copiile pe coperți care să conțină, clar și lizibil, toate aceste Texte de Copertă: Textele Pentru Coperta I pe coperta I și Texte Pentru Coperta IV pe coperta IV. Ambele coperți trebuie de asemenea să vă identifice în mod clar și lizibil ca editor al respectivelor copii. Coperta I trebuie să prezinte titlul în întregime, cu toate cuvintele din titlu la fel de vizibile și proeminente. Puteți adăuga alte materiale pe copertă în plus. Copierea cu modificările limitate la coperți, atâta timp cât satisfac aceste condiții, pot fi tratate în toate celelalte aspecte ca și copii identice.

Dacă textele necesare pentru oricare dintre coperți sunt prea voluminoase pentru a încăpea în mod lizibil, trebuie să puneți primele rânduri (atâtea cât încap în mod rezonabil) pe coperta efectivă și să continuați cu restul pe pagini adiacente.

Dacă publicați sau distribuiți copii Opace ale Documentului în număr mai mare de 100, trebuie ori să includeți câte o copie Transparentă în format electronic împreună cu fiecare copie Opacă, ori să specificați în sau împreună cu fiecare copie Opacă o locație din rețeaua electronică la care publicul general care folosește rețeaua să aibă acces pentru a descărca, folosind un protocol standard public, copii complete, Transparente ale documentului, fără adăugarea oricărui material adițional. Dacă folosiți a doua opțiune trebuie să faceți demersuri rezonabil de prudente ca atunci când începeți distribuirea copiilor Opace să vă asigurați că această copie Transparentă va rămâne accesibilă, în acest fel, la locația respectivă timp de cel puțin un an după distribuția ultimei copii Opace (în mod direct sau prin agenți ori distribuitori) a respectivei ediții pentru public.

Se cere, dar nu în mod necesar, să contactați autorii Documentului cu o perioadă bună înainte de a distribui orice cantitate mare de copii, pentru a le da ocazia să vă pună la dispoziție o versiune actualizată a Documentului.

4. MODIFICĂRI

Puteți copia și distribui o Versiune Modificată a Documentului în condițiile secțiunilor 2 și 3 de mai sus, cu condiția de a acoperi Versiunea Modificată sub exact această Licență, cu Versiunea Modificată ținând locul Documentului, astfel licențiind distribuirea și modificările Versiunii Modificate oricui intră în posesia unei copii ale acesteia. În plus, trebuie să faceți următoarele lucruri în Versiunea Modificată:

A. Folosiți în Pagina de Titlu (și pe coperți, dacă există) un titlu diferit de cel al Documentului, și de versiunile sale anterioare (care trebuie, dacă există, să fie listate în secțiunea de Istorie a Documentului). Puteți folosi același titlu ca o versiune anterioară dacă editorul original al acelei copii vă dă permisiunea.

B. Listați pe Pagina de Titlu, ca autori, una sau mai multe dintre persoanele sau entitățile responsabile în calitate de autori pentru modificările Versiunii Modificate, împreună cu cel puțin cinci dintre autorii principali ai Documentului (toți autorii principali, dacă are mai puțin de cinci), în afară de cazul că aceștia vă eliberează de această obligație.

C. Includeți pe Pagina de Titlu numele editorului Versiunii Modificate în calitate de editor.

D. Păstrați toate notificările de drepturi de autor ale Documentului.

E. Adăugați o notificare de drepturi de autori relevantă pentru modificările Dvs. adiacent celorlalte notificări de drepturi de autor.

F. Includeți, imediat după notificările de drepturi de autor, o notificare de licență dând permisiune publică de a folosi Versiunea Modificată în condițiile acestei Licențe, sub forma prezentată în Apendicele de mai jos.

G. Păstrați în acea notificare de licență lista integrală a Secțiunilor Neschimbabile și Textele de Copertă necesare, date în notificarea de licență a Documentului.

H. Includeți o copie nealterată a acestei Licențe.

I. Păstrați secțiunea Numită „Istorie”, Păstrați-i Titlul și adăugați-i un element care să indice măcar titlul, anul, noii autori și editorul Versiunii Modificate așa cum este dat pe Pagina de Titlu. Dacă nu există o secțiune Numită „Istorie” în Document, creați una în care indicați titlul, anul, autorii și editorul Documentului așa cum este dat pe Pagina de Titlu al acestuia și apoi adăugați un element care să descrie Versiunea Modificată așa cum a fost cerut în fraza precedentă.

J. Păstrați locația de rețea, dacă există, dată în Document pentru acces public la o copie Transparentă a Documentului, cât și locațiile de rețea date în Document pentru versiunile mai vechi pe care s-a bazat acesta. Acestea pot fi incluse în secțiunea Numită „Istorie”. Puteți omite locația de rețea a unei lucrări care a fost publicată cu cel puțin patru ani înainte de Documentul în sine, sau dacă editorul original al versiunii la care se referă vă dă permisiunea.

K. Pentru orice secțiune Numită „Mulțumiri” sau „Dedicații” Păstrați Titlul secțiunii și păstrați în secțiunile respective toată substanța și tonul mulțumirilor și/sau dedicațiilor fiecărui contribuitor.

L. Păstrați toate Secțiunile Neschimbabile ale Documentului, nealterate ca text și ca titluri. Numerotarea secțiunilor sau echivalentul numerotării nu sunt considerate ca făcând parte din titlurile secțiunilor.

M. Ștergeți orice secțiune Numită „Giruri”. O astfel de secțiune nu poate fi inclusă în Versiunea Modificată.

N. Nu modificați titlul nici unei secțiuni existente pentru a fi Numită „Giruri” sau pentru a intra în conflict cu vreo Secțiune Neschimbabilă.

O. Păstrați toate Limitările de Responsabilitate.

Dacă Versiunea Modificată include secțiuni noi incluse în titlu sau anexe care se califică drept Secțiuni Secundare și nu conțin material copiat din Document, aveți dreptul la alegerea Dvs. să numiți unele sau toate acestea ca fiind Neschimbabile. Pentru a face aceasta, adăugați-le titlurile la lista de Secțiuni Neschimbabile în notificarea de licență a Versiunii Modificate. Aceste titluri trebuie să fie distincte față de toate celelalte titlurile de secțiune.

Puteți adăuga o secțiune Numită „Giruri” doar dacă aceasta conține numai girurile a diverse entități asupra Versiunii Modificate—de exemplu recenzii sau faptul că textul a fost aprobat de o organizație ca fiind o definiție autoritară a unui standard.

Puteți adăuga un pasaj de cel mult cinci cuvinte ca Text Pentru Coperta I și un pasaj de cel mult 25 de cuvinte ca Text Pentru Coperta IV la sfârșitul Textelor De Copertă în Versiunea Modificată. Numai un singur pasaj poate fi adăugat la Textul Pentru Coperta I și unul la Textul Pentru Coperta IV de către (sau prin aranjament cu) orice entitate. Dacă Documentul conține deja texte de copertă pentru coperta respectivă, adăugat în prealabil de Dvs. sau prin aranjament

cu aceeași entitate în numele căreia acționați, atunci nu puteți adăuga un altul, însă puteți să-l înlocuiți pe cel vechi numai cu permisiunea explicită a editorului anterior care l-a adăugat pe cel vechi.

Autorul (autorii) și editorul (editorii) Documentului nu vă dau prin această Licență permisiunea de a le folosi numele pentru publicitate sau pentru a pretinde sau implica vreo girare a oricărei Versiuni Modificate.

5. COMBINAREA DOCUMENTELOR

Puteți combina Documentul cu alte documente acoperite de această Licență sub termenii definiți în secțiunea 4 de mai sus pentru versiuni modificate, cu condiția să includeți în versiunea combinată toate Secțiunile Neschimbabile ale tuturor documentelor originale, nemodificate, și să le listați pe toate ca Secțiuni Neschimbabile ale versiunii combinate în notificarea de licență, cât și să păstrați toate Limitările de Responsabilitate.

Versiunea modificată nu trebuie să conțină decât o singură copie a acestei Licențe, iar duplicatele identice ale Secțiunilor Neschimbabile pot fi înlocuite cu o singură copie. Dacă există Secțiuni Neschimbabile cu nume identice și conținut diferit, schimbați-le numele adăugând la sfârșitul titlului, în paranteză, ori numele autorului sau al editorului original al acelei secțiuni dacă acesta este cunoscut, ori un număr unic. Faceți aceleași modificări respective titlurilor secțiunilor în lista de Secțiuni Neschimbabile din notificarea de licență a versiunii combinate.

În versiunea combinată trebuie să combinați și toate secțiunile Numite „Istorie" din diversele documente originale, creând o secțiune unică Numită „Istorie"; la fel trebuie să combinați și toate secțiunile Numite „Mulțumiri" cât și cele Numite „Dedicații". Trebuie să ștergeți toate secțiunile Numite „Giruri".

6. COLECȚII DE DOCUMENTE

Puteți crea o colecție formată din Document și alte documente acoperite de această Licență și să înlocuiți copiile individuale ale acestei Licențe din diversele documente cu o singură copie care să fie inclusă în colecție cu condiția să urmați regulile acestei Licențe pentru copii identice pentru fiecare document în toate celelalte privințe.

Puteți să extrageți un document dintr-o astfel de colecție și să-l distribuiți individual sub această Licență cu condiția de a include o copie a acestei Licențe în documentul extras și să urmați condițiile acestei Licențe în toate celelalte privințe în legătură cu copiile identice ale acelui document.

7. AGREGAREA CU LUCRĂRI INDEPENDENTE

O compilație a Documentului sau a unui derivat al său cu orice document sau lucrare separată independentă, în sau pe un volum de stocare sau distribuire se numește „agregat" dacă drepturile de autor rezultate în urma compilării nu sunt folosite pentru a limita drepturile legale ale utilizatorilor compilației mai mult decât permit lucrările individuale. Când Documentul este inclus într-un agregat, această Licență nu se aplică celorlalte lucrări din agregat care nu sunt ele însele rezultate derivate ale Documentului.

Dacă cerințele legate de Textele de Copertă din secțiunea 3 se aplică acestor copii ale Documentului, atunci dacă Documentul este mai puțin de jumătate din întregul agregat atunci Textele de Copertă ale Documentului pot fi puse pe coperți care să separe Documentul în cadrul agregatului, sau pe un echivalent electronic al acestora, dacă Documentul se prezintă în format electronic. Altfel ele trebuie să apară pe coperțile tipărite care îmbracă întreg agregatul.

8. TRADUCERE

Traducerea este considerată o formă de modificare, drept care puteți distribui traduceri ale Documentului sub cerințele secțiunii 4. Înlocuirea Secțiunilor Neschimbabile cu traduceri ale acestora necesită permisiune specială din partea celor care dețin drepturile de autor, însă puteți include traduceri ale unora dintre sau tuturor Secțiunilor Neschimbabile împreună cu variantele originale ale acestora. Puteți include o traducere a acestei Licențe cât și toate notificările de licență din Document, cât și Limitările de Responsabilitate atâta timp cât includeți și versiunea originală în engleză a acestei Licențe, plus versiunile originale ale respectivelor notificări de licență și limitări de responsabilitate. În cazul apariției oricăror discrepanțe între versiunea tradusă și versiunea originală a acestei Licențe, a vreunei notificări de licență sau a vreunei limitări de responsabilitate, versiunea originală are prioritate.

Dacă vreo secțiune din Document este Numită „Mulțumiri", „Dedicații" sau „Istorie" cerința (din secțiunea 4) de a-i Păstra Titlul (secțiunea 1) va necesita în mod normal schimbarea titlului în sine.

9. REZILIERE

Nu puteți copia, modifica, sublicenția sau distribui Documentul decât în condițiile specificate explicit în această Licență. Orice copiere, modificare sau redistribuire a Documentului în vreo altă condiție este nulă și vă va anula în mod automat drepturile conferite de această Licență.

Cu toate acestea, dacă încetați orice încălcare a acestei Licențe, licența din partea titularului dreptului de autor este reinstaurată (a) cu titlu provizoriu, cu excepția cazului când titularul dreptului de autor încetează în mod explicit și în cele din urmă licența, și (b) permanent, în cazul în care titularul dreptului de autor nu vă anunță încălcarea, prin mijloace reonabile, în termen de 60 de zile de la încetare.

În plus, licența de la titularul particular al dreptului de autor este repusă permanent în cazul în care titularul dreptului de autor vă anunță de încălcare prin mijloace rezonabile, și este prima dată când ați primit o notificare de încălcare a acestei Licențe (pentru orice lucrare), din partea titularului dreptului de autor, și ați încetat încălcarea cu 30 de zile înainte de primirea notificării.

Încetarea drepturilor dumneavoastră, în conformitate cu această secțiune, nu încetează licențele părților care au primit copii sau drepturi de la dumneavoastră sub această Licență. Dacă drepturile dumneavoastră au fost terminate și nu s-au repus permanent, primirea unei copii ale aceluiași material nu vă dă nici un drept să-l folosiți.

10. VERSIUNI VIITOARE ALE ACESTEI LICENȚE

Fundația Free Software Foundation poate publica, din când în când, versiuni noi, revizuite ale acestei Licențe GNU pentru Documentația liberă. Aceste noi versiuni vor păstra spiritul acestei versiuni dar pot diferi în privința detaliilor, cu scopul de a se adresa unor noi probleme reale sau potențiale. A se vedea http://www.gnu.org/copyleft/

Fiecărei versiuni ale acestei Licențe îi este asociat un număr de versiune distinct. Dacă Documentul specifică un anumit număr de versiune „sau orice versiune ulterioară" al acestei Licențe, aveți de ales între a vă conforma termenilor și condițiilor ori ale versiunii specificate explicit sau ale oricărei variante ulterioare publicate (nu ca variantă preliminară) de către Free Software Foundation. Dacă Documentul nu specifică un număr de versiune al acestei Licențe atunci puteți alege orice versiune publicată (nu ca variantă preliminară) de către Free Software Foundation.

11. RELICENŢIEREA

„Site-ul de Colaborare Masivă a Multiautorilor" (sau „MMC Site") înseamnă orice server www care publică lucrări posibil de a fi supuse drepturilor de autor şi, de asemenea, oferă facilităţi proeminente pentru oricine editează aceste lucrări. Un server pubic wiki este un exemplu în care oricine poate edita lucrări scrise. Un „Site de Colaborare Masivă a Multiautorilor" (sau „MMC Site") în conţinutul său înseamnă o mulţime de lucrări susceptibile de a fi supuse licenţelor supsue site-ului MMC.

„CC-BZ-SA" înseamnă licenţă a Creative Commons Attribution-Share Alike 3.0 publicată de Corporaţia Creative Commons, o corporaţie nonprofit cu sediul principal la San Francisco, California, şi deasemenea viitoarele variante de „obligații" a acestei licenţe publicată de aceeaşi organizaţie.

„Încorporarea" înseamnă publicarea ori republicarea unui Document, în întregime sau în parte, ca parte a unui alt document.

Un MMC este „eligibil pentru reautorizare" dacă este licențiat sub această Licență, și în cazul în care toate lucrările care au fost publicate mai întâi în această Licență în altă parte decât acest MMC, și, ulterior, au fost încorporate în totalitate sau în parte în MMC, (1) nu a avut texte de copertă sau secțiuni invariante, și (2) au fost astfel incluse până la 1 noiembrie 2008.

Operatorul unui site MMC poate republica un MMC conținut în site sub CC-BY-SA în același loc, în orice moment înainte de 1 august 2009, cu condiția ca MMC să fie eligibil pentru reautorizare.

ADDENDUM: Cum să utilizaţi această licenţă pentru documentele dvs.

Pentru a utiliza această licenţă într-un document pe care l-aţi scris, includeți o copie a Licenţei în document şi introduceți următoarele notificări referitoare la dreptul de autor și la licență, imediat după pagina de titlu:

```
Copyright © YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

Dacă aveți Secțiuni Neschimbabile, Texte pentru Coperta I și Texte pentru Coperta IV, înlocuiți linia „cu … Texte." cu:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

Dacă aveți Secțiuni Neschimbabile, fără Texte de Copertă, sau o altă combinație a celor trei, fuzionați ambele alternative pentru a se potrivi situației.

Dacă documentul conține exemple netriviale de cod de programare, vă recomandăm publicarea acestor exemple, în paralel, sub o licență pentru softul liber, cum ar fi Licența Publică Generală GNU, pentru a permite utilizarea lor în software-ul liber.

32.3 Anexa C: Formatele Fișierelor QGIS

32.3.1 QGS/QGZ - Formatul Fișierelor Proiect

The **QGS** format is an XML format for storing QGIS projects. The **QGZ** format is a compressed (zip) archive containing a QGS file and a QGD file. The **QGD** file is the associated sqlite database of the qgis project that contain auxiliary data for the project. If there are no auxiliary data, the QGD file will be empty.

A QGIS file contains everything that is needed for storing a QGIS project, including:

- project title
- project CRS
- the layer tree
- snapping settings
- relations
- the map canvas extent
- project models
- legenda
- mapview docks (2D and 3D)
- the layers with links to the underlying datasets (data sources) and other layer properties including extent, SRS, joins, styles, renderer, blend mode, opacity and more.
- project properties

The figures below show the top level tags in a QGS file and the expanded `ProjectLayers` tag.

Fig. 32.1: The top level tags in a QGS file

```
−<projectlayers>
 −<maplayer styleCategories="AllStyleCategories" readOnly="0" autoRefreshTime="0" autoRefreshEnabled="0" refreshOnNotifyEnabled="0" maxScale="0"
  geometry="Polygon" labelsEnabled="0" type="vector" simplifyDrawingHints="1" hasScaleBasedVisibilityFlag="0" simplifyDrawingTol="1"
  simplifyMaxScale="1" minScale="1e+8" simplifyAlgorithm="0" simplifyLocal="1" refreshOnNotifyMessage="">
  +<extent></extent>
   <id>watersheds_b62efa19_8809_4406_b6ec_2951ac4c94c5</id>
  −<datasource>
     ./QGIS-Training-Data-2.0/exercise_data/processing/generalize/watersheds.shp
   </datasource>
  +<keywordList></keywordList>
   <layername>watersheds</layername>
  +<srs></srs>
  +<resourceMetadata></resourceMetadata>
   <provider encoding="UTF-8">ogr</provider>
   <vectorjoins/>
   <layerDependencies/>
   <dataDependencies/>
   <legend type="default-vector"/>
   <expressionfields/>
  +<map-layer-style-manager current="default"></map-layer-style-manager>
   <auxiliaryLayer/>
  +<flags></flags>
  +<renderer-v2 symbollevels="0" enableorderby="0" type="singleSymbol" forceraster="0"></renderer-v2>
  +<customproperties></customproperties>
   <blendMode>0</blendMode>
   <featureBlendMode>0</featureBlendMode>
   <layerOpacity>1</layerOpacity>
  +<SingleCategoryDiagramRenderer diagramType="Histogram" attributeLegend="1"></SingleCategoryDiagramRenderer>
  +<DiagramLayerSettings priority="0" linePlacementFlags="18" dist="0" showAll="1" placement="1" obstacle="0" zIndex="0"></DiagramLayerSettings>
  +<geometryOptions removeDuplicateNodes="0" geometryPrecision="0"></geometryOptions>
  +<fieldConfiguration></fieldConfiguration>
  +<aliases></aliases>
   <excludeAttributesWMS/>
   <excludeAttributesWFS/>
  +<defaults></defaults>
  +<constraints></constraints>
  +<constraintExpressions></constraintExpressions>
   <expressionfields/>
  +<attributeactions></attributeactions>
  +<attributetableconfig actionWidgetStyle="dropDown" sortExpression="" sortOrder="0"></attributetableconfig>
  +<conditionalstyles></conditionalstyles>
   <editform tolerant="1"/>
   <editforminit/>
   <editforminitcodesource>0</editforminitcodesource>
   <editforminitfilepath/>
  +<editforminitcode></editforminitcode>
   <featformsuppress>0</featformsuppress>
   <editorlayout>generatedlayout</editorlayout>
  +<editable></editable>
  +<labelOnTop></labelOnTop>
   <widgets/>
   <previewExpression>ID</previewExpression>
   <mapTip/>
  </maplayer>
 </projectlayers>
```

Fig. 32.2: The expanded top level ProjectLayers tag of a QGS file

32.3.2 QLR - Fișier cu Definiții ale Straturilor

A Layer Definition file (QLR) is an XML file that contains a pointer to the layer data source in addition to QGIS style information for the layer.

The use case for this file is simple: To have a single file for opening a data source and bringing in all the related style information. QLR files also allow you to mask the underlying datasource in an easy to open file.

An example of QLR usage is for opening MS SQL layers. Rather than having to go to the MS SQL connection dialog, connect, select, load and finally style, you can simply add a .qlr file that points to the correct MS SQL layer with all the necessary style included.

In the future a .qlr file may hold a reference to more than one layer.

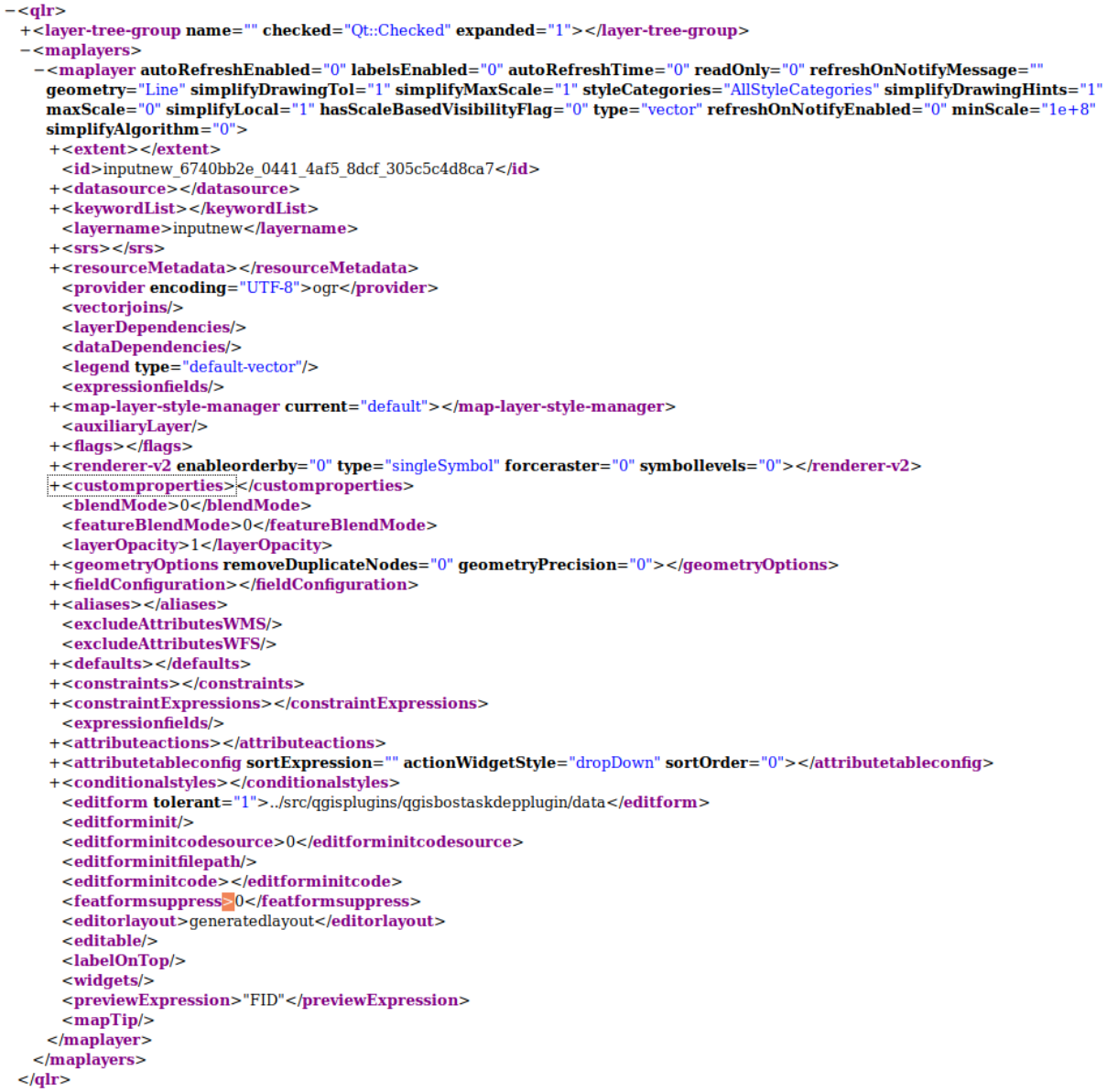


Fig. 32.3: The top level tags of a QLR file

32.3.3 QML - Formatul Fișierelor de Stilizare

QML is an XML format for storing layer styling.

A QML file contains all the information QGIS can handle for the rendering of feature geometries including symbol definitions, sizes and rotations, labelling, opacity and blend mode and more.

The figure below shows the top level tags of a QML file (with only `renderer_v2` and its `symbol` tag expanded).

```
−<qgis version="3.4.13-Madeira" styleCategories="AllStyleCategories" readOnly="0" maxScale="0"
 labelsEnabled="0" simplifyDrawingHints="1" hasScaleBasedVisibilityFlag="0" simplifyDrawingTol="1"
 simplifyMaxScale="1" minScale="1e+8" simplifyAlgorithm="0" simplifyLocal="1">
 +<flags></flags>
 −<renderer-v2 symbollevels="0" enableorderby="0" type="singleSymbol" forceraster="0">
   −<symbols>
     +<symbol clip_to_extent="1" name="0" alpha="1" type="fill" force_rhr="0"></symbol>
    </symbols>
    <rotation/>
    <sizescale/>
  </renderer-v2>
 +<customproperties></customproperties>
  <blendMode>0</blendMode>
  <featureBlendMode>0</featureBlendMode>
  <layerOpacity>1</layerOpacity>
 +<SingleCategoryDiagramRenderer diagramType="Histogram" attributeLegend="1">
  </SingleCategoryDiagramRenderer>
 +<DiagramLayerSettings priority="0" linePlacementFlags="18" dist="0" showAll="1" placement="1"
  obstacle="0" zIndex="0">
  </DiagramLayerSettings>
 +<geometryOptions removeDuplicateNodes="0" geometryPrecision="0"></geometryOptions>
 +<fieldConfiguration></fieldConfiguration>
 +<aliases></aliases>
  <excludeAttributesWMS/>
  <excludeAttributesWFS/>
 +<defaults></defaults>
 +<constraints></constraints>
 +<constraintExpressions></constraintExpressions>
  <expressionfields/>
 +<attributeactions></attributeactions>
 +<attributetableconfig actionWidgetStyle="dropDown" sortExpression="" sortOrder="0">
  </attributetableconfig>
 +<conditionalstyles></conditionalstyles>
  <editform tolerant="1"/>
  <editforminit/>
  <editforminitcodesource>0</editforminitcodesource>
  <editforminitfilepath/>
 +<editforminitcode></editforminitcode>
  <featformsuppress>0</featformsuppress>
  <editorlayout>generatedlayout</editorlayout>
 +<editable></editable>
 +<labelOnTop></labelOnTop>
  <widgets/>
  <previewExpression>ID</previewExpression>
  <mapTip/>
  <layerGeometryType>2</layerGeometryType>
</qgis>
```

Fig. 32.4: The top level tags of a QML file (only the renderer_v2 tag with its symbol tag is expanded)

32.4 Anexa D: Sintaxa scripturilor R din QGIS

Contributed by Matteo Ghetta - funded by Scuola Superiore Sant'Anna

Writing R scripts in Processing is a bit tricky because of the special syntax.

A Processing R script starts with defining its **Inputs** and **Outputs**, each preceded with double hash characters (##).

Before the inputs, the group to place the algoritm in can be specified. If the group already exists, the algorithm will be added to it, if not, the group will be created. In the example below, the name of the group is *My group*:

```
##My Group=group
```

32.4.1 Intrări

All input data and parameters have to be specified. There are several types of inputs:

- vector: `##Layer = vector`
- vector field: `##F = Field Layer` (where *Layer* is the name of an input vector layer the field belongs to)
- raster: `##r = raster`
- table: `##t = table`
- number: `##Num = number`
- string: `##Str = string`
- boolean: `##Bol = boolean`
- elements in a dropdown menu. The items must be separated with semicolons `;`: `##type=selection point;lines;point+lines`

32.4.2 Rezultat

Ca și intrările, fiecare ieșire trebuie să fie definită la începutul script-ului:

- vector: `##output= output vector`
- raster: `##output= output raster`
- table: `##output= output table`
- plots: `##output_plots_to_html` (##showplots in earlier versions)
- To show R output in the *Result Viewer*, put > in front of the command whose output you would like to show.

32.4.3 Syntax Summary for QGIS R scripts

A number of input and output parameter types are offered.

Input parameter types

Parametru	Exemplu de sintaxă	Obiecte returnate
vector	Layer = vector	sf object (or SpatialDataFrame object, if ##load_vector_using_rgdal is specified)
vector point	Layer = vector point	sf object (or SpatialDataFrame object, if ##load_vector_using_rgdal is specified)
vector line	Layer = vector line	sf object (or SpatialDataFrame object, if ##load_vector_using_rgdal is specified)
vector polygon	Layer = vector polygon	sf object (or SpatialPolygonsDataFrame object, if ##load_vector_using_rgdal is used)
multiple vector	Layer = multiple vector	sf object (or SpatialDataFrame objects if ##load_vector_using_rgdal is specified)
tabelă	Layer = table	dataframe convertit din csv, obiect implicit al funcției `read.csv`
field	Field = Field Layer	numele câmpului selectat, ex.: `"Area"`
raster	Layer = raster	Obiect RasterBrick, obiect implicit al pachetului `raster`
multiple raster	Layer = multiple raster	Obiect RasterBrick, obiect implicit al pachetului `raster`
number	N = number	numărul întreg sau zecimal ales
string	S = string	textul adăugat în casetă
longstring	LS = longstring	șirul de caractere adăugat în casetă, ar putea fi mai lung decât șirul normal
selection	S = selection first;second;third	șirul cu elementele care vor putea fi selectate din meniul derulant
crs	C = crs	string of the resulting CRS chosen, in the format: `"EPSG:4326"`
extent	E = extent	Obiectul extindere din pachetul `raster`, puteți extrage valorile astfel: `E@xmin`
point	P = point	când faceți clic pe hartă, puteți vedea coordonatele punctului
file	F = file	calea fișierului ales, ex.: „/home/matteo/file.txt”
folder	F = folder	calea directorului ales, ex.: „/home/matteo/Downloads”

A parameter can be **OPTIONAL**, meaning that it can be ignored.

In order to set an input as optional, you add the string `optional` **before** the input, e.g:

```
##Layer = vector
##Field1 = Field Layer
##Field2 = optional Field Layer
```

Output parameter types

Parametru	Exemplu de sintaxă
vector	Output = output vector
raster	Output = output raster
tabelă	Output = output table
file	Output = output file

Notă: You can save plots as `png` from the *Processing Result Viewer*, or you can choose to save the plot directly from the algorithm interface.

Corpul scriptului

The script body follows R syntax and the **Log** panel can help you if there is something wrong with your script.

Remember that you have to load all additional libraries in the script:

```
library(sp)
```

32.4.4 Exemple

Exemplu de ieșire vectorială

Let's take an algorithm from the online collection that creates random points from the extent of an input layer:

```
##Point pattern analysis=group
##Layer=vector polygon
##Size=number 10
##Output=output vector
library(sp)
spatpoly = as(Layer, "Spatial")
pts=spsample(spatpoly,Size,type="random")
spdf=SpatialPointsDataFrame(pts, as.data.frame(pts))
Output=st_as_sf(spdf)
```

Explanation (per line in the script):

1. `Analiza modelului de puncte` reprezintă grupul algoritmului
2. `Stratul` reprezintă stratul de intrare **vectorial**
3. `Size` is a **numerical** parameter with a default value of 10
4. `Ieșire` reprezintă stratul **vectorial** care va fi creat de către algoritm
5. `library(sp)` loads the **sp** library
6. `spatpoly = as(Layer, "Spatial")` translate to an sp object
7. Call the `spsample` function of the `sp` library and run it using the input defined above (`Layer` and `Size`)
8. Create a *SpatialPointsDataFrame* object using the `SpatialPointsDataFrame` function
9. Create the output vector layer using the `st_as_sf` function

That's it! Just run the algorithm with a vector layer you have in the QGIS Legend, choose the number of random point. The resulting layer will be added to your map.

Exemplu de ieșire raster

The following script will perform basic ordinary kriging to create a raster map of interpolated values from a specified field of the input point vector layer by using the `autoKrige` function of the `automap` R package. It will first calculate the kriging model and then create a raster. The raster is created with the `raster` function of the raster R package:

```
##Basic statistics=group
##Layer=vector point
##Field=Field Layer
##Output=output raster
##load_vector_using_rgdal
require("automap")
require("sp")
require("raster")
```

(continues on next page)

(continuare din pagina precedentă)

```
table=as.data.frame(Layer)
coordinates(table)= ~coords.x1+coords.x2
c = Layer[[Field]]
kriging_result = autoKrige(c~1, table)
prediction = raster(kriging_result$krige_output)
Output<-prediction
```

By using `##load_vector_using_rgdal`, the input vector layer will be made available as a `SpatialDataFrame` objects, so we avoid having to translate it from an `sf` object.

Exemplu de generare a unui tabel

Let's edit the `Summary Statistics` algorithm so that the output is a table file (csv).

The script body is the following:

```
##Basic statistics=group
##Layer=vector
##Field=Field Layer
##Stat=Output table
Summary_statistics<-data.frame(rbind(
    sum(Layer[[Field]]),
    length(Layer[[Field]]),
    length(unique(Layer[[Field]])),
    min(Layer[[Field]]),
    max(Layer[[Field]]),
    max(Layer[[Field]])-min(Layer[[Field]]),
    mean(Layer[[Field]]),
    median(Layer[[Field]]),
    sd(Layer[[Field]]))),
  row.names=c("Sum:","Count:","Unique values:","Minimum value:","Maximum value:",
↪"Range:","Mean value:","Median value:","Standard deviation:"))
colnames(Summary_statistics)<-c(Field)
Stat<-Summary_statistics
```

The third line specifies the **Vector Field** in input and the fourth line tells the algorithm that the output should be a table.

The last line will take the `Stat` object created in the script and convert it into a `csv` table.

Example with console output

We can use the previous example and instead of creating a table, print the result in the **Result Viewer**:

```
##Basic statistics=group
##Layer=vector
##Field=Field Layer
Summary_statistics<-data.frame(rbind(
sum(Layer[[Field]]),
length(Layer[[Field]]),
length(unique(Layer[[Field]])),
min(Layer[[Field]]),
max(Layer[[Field]]),
max(Layer[[Field]])-min(Layer[[Field]]),
mean(Layer[[Field]]),
median(Layer[[Field]]),
sd(Layer[[Field]])),row.names=c("Sum:","Count:","Unique values:","Minimum value:",
↪"Maximum value:","Range:","Mean value:","Median value:","Standard deviation:"))
colnames(Summary_statistics)<-c(Field)
>Summary_statistics
```

The script is exactly the same as the one above except for two edits:

1. no output specified (the fourth line has been removed)
2. the last line begins with >, telling Processing to make the object available through the result viewer

Exemplu de grafic

To create plots, you have to use the `##output_plots_to_html` parameter as in the following script:

```
##Basic statistics=group
##Layer=vector
##Field=Field Layer
##output_plots_to_html
####output_plots_to_html
qqnorm(Layer[[Field]])
qqline(Layer[[Field]])
```

The script uses a field (`Field`) of a vector layer (`Layer`) as input, and creates a *QQ Plot* (to test the normality of the distribution).

The plot is automatically added to the Processing *Result Viewer*.

32.5 Anexa E: Conexiuni de Rețea ale Aplicației QGIS

Aceasta este o listă a conexiunilor de rețea pre-configurate/automatizate pe care le realizează QGIS. Unele dintre conexiuni sunt inițiate de utilizator, întrucât necesită o intervenție anterioară pentru a se desfășura, altele au loc automat.

Denumire	Scop	Mod UX	Server	Informații trimise către server	Informații stocate pe server
qgis.org					
Document API Python	Documentația online PyQGIS	Inițiere de către utilizator	https://qgis.org/pyqgis/%1/search.html?q=%2	IP, versiune QGIS, OS	Jurnalul IP-urilor de la care s-a accesat serverul
version.qg					
Verificare acțiune nouă	Notifică utilizatorul despre disponibilitatea noilor versiuni QGIS	Automat	https://version.qgis.org	IP, versiune QGIS, OS	Jurnalul IP-urilor de la care s-a accesat serverul
feed.qgis.					
Flux QGIS	Preia din flux știrile despre QGIS	Automat	https://feed.qgis.org	IP, QGIS version, language code, last download timestamp, OS	IP in server log; QGIS version, OS and IP are aggregated and used to collect some statistics
plugins.qg					
Verifică actualizări pluginuril	Notifică utilizatorul despre actualizările pluginurilor	Inițiat de utilizator/aut (configurabil	https://plugins.qgis.org	IP, versiune QGIS, OS	Jurnalul IP-urilor de la care s-a accesat serverul
Lista Pluginuril	Preia lista pluginurilor	Inițiere de către utilizator	https://plugins.qgis.org	IP, versiune QGIS, OS	Jurnalul IP-urilor de la care s-a accesat serverul
Instalare plugin	Descarcă și instalează pachetul unui plugin	Inițiere de către utilizator	https://plugins.qgis.org	IP, versiune QGIS, OS	Incrementează cu o unitate contorul de descărcare a pluginurilor
Stiluri	Listează stilurile transmise de către utilizatori	Inițiere de către utilizator	https://plugins.qgis.org/styles	IP, versiune QGIS, OS	Incrementează cu o unitate contorul descărcărilor
Părți t3rțe					
Date despre teren	Generează un DEM pentru vizualizări 3D	Inițiere de către utilizator	https://s3.amazonaws.com/elevation-tiles-prod/terrarium/{z}/{x}/{y	IP, versiune QGIS, OS	consultați Amazon TOS
Google Map Geocoder	Servicii de geocodificare	Inițiere de către utilizator	https://maps.googleapis.com/maps/api/geocode/json	IP, versiune QGIS, OS	Consultați Google Maps API TOS
Geocoder Nominatir	Servicii de geocodificare	Inițiere de către utilizator	https://nominatim.qgis.org/ui/search.html	IP, versiune QGIS, OS	
Grilă geodezică	Adaugă o nouă grilă PROJ	Inițiere de către utilizator	https://cdn.proj.org	IP, versiune PROJ	Jurnalele de acces sunt șterse definitiv după o zi

CAPITOLUL 33

Literatură și Referințe Web

GDAL-SOFTWARE-SUITE. Biblioteca de extracție a datelor geospațiale. https://gdal.org, 2013.

GRASS-PROJECT. Geographic resource analysis support system. https://grass.osgeo.org, 2013.

NETELER, M., AND MITASOVA, H. GIS cu sursă deschisă: O abordare GRASS GIS, 2008.

OGR-SOFTWARE-SUITE. Geospatial data abstraction library. https://gdal.org, 2013.

OPEN-GEOSPATIAL-CONSORTIUM. Web map service (1.1.1) implementation specification. https://portal.ogc.org/files/?artifact_id=1081&version=1&format=pdf, 2002.

OPEN-GEOSPATIAL-CONSORTIUM. Web map service (1.3.0) implementation specification. https://portal.ogc.org/files/?artifact_id=14416&format=pdf, 2004.

POSTGIS-PROJECT. Spatial support for postgresql. http://www.refractions.net/products/postgis/, 2013.