<table>
<thead>
<tr>
<th>Obsah</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Úvod</td>
</tr>
<tr>
<td>1.1 What is new in QGIS 3.10</td>
</tr>
<tr>
<td>2 Předmluva</td>
</tr>
<tr>
<td>3 Konvence</td>
</tr>
<tr>
<td>3.1 GUI konvence</td>
</tr>
<tr>
<td>3.2 Textové nebo klávesové konvence</td>
</tr>
<tr>
<td>3.3 Platformy – pokyny</td>
</tr>
<tr>
<td>4 Funkce</td>
</tr>
<tr>
<td>4.1 Prohlížení dat</td>
</tr>
<tr>
<td>4.2 Prozkoumávání dat a vytváření map</td>
</tr>
<tr>
<td>4.3 Tvorba, editace, správa a export dat</td>
</tr>
<tr>
<td>4.4 Analýza dat</td>
</tr>
<tr>
<td>4.5 Publikování map na internetu</td>
</tr>
<tr>
<td>4.6 Rozšíření QGIS funkcí prostřednictvím pluginů (zásuvných modulů)</td>
</tr>
<tr>
<td>4.6.1 Základní Pluginy</td>
</tr>
<tr>
<td>4.6.2 Externí Python zásuvné moduly</td>
</tr>
<tr>
<td>4.7 Konzole Python</td>
</tr>
<tr>
<td>4.8 Známé problémy</td>
</tr>
<tr>
<td>4.8.1 Omezení počtu otevřených souborů</td>
</tr>
<tr>
<td>5 Začínáme</td>
</tr>
<tr>
<td>5.1 Installing QGIS</td>
</tr>
<tr>
<td>5.1.1 Installing from binaries</td>
</tr>
<tr>
<td>5.1.2 Installing from source</td>
</tr>
<tr>
<td>5.1.3 Installing on external media</td>
</tr>
<tr>
<td>5.1.4 Downloading sample data</td>
</tr>
<tr>
<td>5.2 Starting and stopping QGIS</td>
</tr>
<tr>
<td>5.3 Sample Session: Loading raster and vector layers</td>
</tr>
<tr>
<td>6 Working with Project Files</td>
</tr>
<tr>
<td>6.1 Introducing QGIS projects</td>
</tr>
<tr>
<td>6.2 Generating output</td>
</tr>
<tr>
<td>7 QGIS GUI</td>
</tr>
<tr>
<td>7.1 Menu lišta</td>
</tr>
<tr>
<td>7.1.1 Projekt</td>
</tr>
<tr>
<td>7.1.2 Editovat</td>
</tr>
<tr>
<td>7.1.3 Zobrazit</td>
</tr>
</tbody>
</table>
8 The Browser panel

8.1 Resources that can be opened / run from the Browser ........................................ 56
8.2 Browser panel top-level entries .................................................................................. 56
  8.2.1 Favorites .............................................................................................................. 56
  8.2.2 Prostorové záložky ............................................................................................. 56
  8.2.3 Home .................................................................................................................. 56
  8.2.4 / ............................................................................................................................ 57
  8.2.5 Geopackage ........................................................................................................ 57
  8.2.6 Spatialite ............................................................................................................. 57
  8.2.7 PostGIS .............................................................................................................. 57
  8.2.8 MSSQL ............................................................................................................... 58
  8.2.9 DB2 ...................................................................................................................... 58
  8.2.10 WMS/WMTS .................................................................................................... 58
  8.2.11 Vector Tiles ....................................................................................................... 59
  8.2.12 XYZ Tiles ......................................................................................................... 59
  8.2.13 WCS ................................................................................................................ 59
  8.2.14 WFS / OGC API - Features ............................................................................. 59
  8.2.15 OWS ................................................................................................................ 60
  8.2.16 ArcGIS Map Service ........................................................................................ 60
  8.2.17 ArcGIS Features Service ................................................................................ 60
  8.2.18 GeoNode .......................................................................................................... 60
8.3 Resources .................................................................................................................... 60

9 QGIS Konfigurace ........................................................................................................... 63
  9.1 Možnosti ................................................................................................................. 63
    9.1.1 General Settings ............................................................................................... 64
    9.1.2 System Settings ............................................................................................... 65
    9.1.3 CRS Settings .................................................................................................... 67
    9.1.4 Data Sources Settings ...................................................................................... 68
    9.1.5 Rendering Settings .......................................................................................... 70
    9.1.6 Canvas and Legend Settings .......................................................................... 73
    9.1.7 Map tools Settings .......................................................................................... 73
    9.1.8 Colors Settings ............................................................................................... 75
10 Práce s projekcemi 103
10.1 Přehled projekční podpory 103
10.2 Layer Coordinate Reference Systems 103
10.3 Project Coordinate Reference Systems 105
10.4 Coordinate Reference System Selector 107
10.5 Vlastní souřadnicový referenční systém 108
10.5.1 Integrate an NTv2-transformation in QGIS 108
10.6 Datum Transformations 108

11 Obecné nástroje 111
11.1 Kontextová nápověda 111
11.2 Panely 111
11.2.1 Layers Panel 111
11.2.2 Layer Styling Panel 116
11.2.3 Layer Order Panel 116
11.2.4 Overview Panel 118
11.2.5 Log Messages Panel 118
11.2.6 Undo/Redo Panel 118
11.2.7 Statistical Summary Panel 119
11.3 Vnořování projektů 121
11.4 Working with the map canvas 121
11.4.1 Vykreslování 121
11.4.2 Zooming and Panning 123
11.4.3 Prostorové založky 124
11.4.4 Dekorace 125
11.4.5 Anotační nástroje 132
11.4.6 Měření 134
11.5 Interacting with features 136
11.5.1 Selecting features 136
11.5.2 Identifying Features 139
11.6 Save and Share Layer Properties 142
11.6.1 Managing Custom Styles ........................................ 142
11.6.2 Storing Styles in a File or a Database ............................ 143
11.6.3 Layer definition file .............................................. 144
11.7 Storing values in Variables ........................................ 145
11.8 Authentication ..................................................... 145
11.9 Common widgets .................................................. 145
11.9.1 Výběr barvy ..................................................... 147
11.9.2 Symbol Widget .................................................. 150
11.9.3 Font Selector ................................................... 150
11.9.4 Unit Selector .................................................... 151
11.9.5 Režim michání .................................................. 152
11.9.6 Data defined override setup .................................... 152

12 The Style Library ..................................................... 155
12.1 The Style Manager .................................................. 155
12.1.1 The Style Manager dialog ....................................... 155
12.1.2 Setting a Color Ramp ........................................... 160
12.2 The Symbol Selector ............................................... 161
12.2.1 The symbol layer tree ......................................... 163
12.2.2 Configuring a symbol .......................................... 163
12.3 Setting a label ...................................................... 171
12.3.1 Formatting the label text ....................................... 171
12.3.2 Callouts tab ..................................................... 178
12.3.3 Placement tab ................................................... 179
12.3.4 Rendering tab ................................................... 181

13 Managing Data Source ................................................. 183
13.1 Opening Data ...................................................... 183
13.1.1 The Browser Panel ............................................. 185
13.1.2 The DB Manager ............................................... 188
13.1.3 Provider-based loading tools .................................. 188
13.1.4 QGIS Custom formats ......................................... 204
13.1.5 QLR - QGIS Layer Definition File .............................. 204
13.1.6 Connecting to web services .................................... 204
13.1.7 Handling broken file paths ..................................... 206
13.2 Creating Layers ..................................................... 206
13.2.1 Creating new vector layers ..................................... 206
13.2.2 Creating new layers from an existing layer .................... 212
13.2.3 Creating new DXf files ......................................... 216
13.2.4 Creating new layers from the clipboard ......................... 217
13.2.5 Creating virtual layers ......................................... 217
13.3 Exploring Data Formats and Fields ................................. 220
13.3.1 Raster data ..................................................... 220
13.3.2 Vektorová data ................................................ 220

14 Working with Vector Data ............................................. 229
14.1 The Vector Properties Dialog .................................... 229
14.1.1 Information Properties .......................................... 230
14.1.2 Source Properties .............................................. 230
14.1.3 Symbology Properties .......................................... 233
14.1.4 Labels Properties ............................................... 253
14.1.5 Diagrams Properties ............................................ 262
14.1.6 3D View Properties ............................................ 268
14.1.7 Fields Properties ............................................... 269
14.1.8 Attributes Form Properties ..................................... 270
14.1.9 Joins Properties ................................................ 276
14.1.10 Auxiliary Storage Properties .................................. 278
14.1.11 Actions Properties ............................................. 285
14.1.12 Display Properties ............................................. 290
14.1.13 Rendering Properties .................................................. 292
14.1.14 Variables Properties .................................................... 293
14.1.15 Metadata Properties ................................................... 293
14.1.16 Dependencies Properties ............................................. 293
14.1.17 Legend Properties ....................................................... 294
14.1.18 QGIS Server Properties ............................................... 294
14.1.19 Digitizing Properties ................................................. 295

14.2 Výrazy ................................................................. 299
14.2.1 The Expression string builder ........................................ 299
14.2.2 List of functions ........................................................ 300
14.2.3 Function Editor ......................................................... 319

14.3 Working with the Attribute Table ................................. 321
14.3.1 Foreword: Spatial and non-spatial tables ....................... 321
14.3.2 Introducing the attribute table interface .......................... 321
14.3.3 Interacting with features in an attribute table ................. 326
14.3.4 Using action on features .............................................. 328
14.3.5 Editing attribute values .............................................. 329
14.3.6 Creating one or many to many relations ......................... 333

14.4 Editování .............................................................. 341
14.4.1 Nastavení příchyťavého tolerance a vyhledání poloměru .... 341
14.4.2 Topologická editace ................................................... 343
14.4.3 Digitizing an existing layer ......................................... 345
14.4.4 Advanced digitizing .................................................. 352
14.4.5 Shape digitizing ....................................................... 359
14.4.6 The Advanced Digitizing panel ..................................... 362
14.4.7 The Processing in-place layer modifier ......................... 368

15 Working with Raster Data .................................................. 371
15.1 Raster Properties Dialog ................................................ 371
15.1.1 Information Properties ............................................... 372
15.1.2 Source Properties ..................................................... 372
15.1.3 Symbology Properties ............................................... 373
15.1.4 Transparency Properties ............................................. 380
15.1.5 Histogram Properties ............................................... 381
15.1.6 Rendering Properties ............................................... 382
15.1.7 Pyramids Properties .................................................. 382
15.1.8 Metadata Properties ................................................ 383
15.1.9 Legend Properties .................................................... 384
15.1.10 QGIS Server Properties ............................................. 384

15.2 Raster Analysis ........................................................ 384
15.2.1 Raster Calculator ..................................................... 384
15.2.2 Raster Alignment ..................................................... 388

16 Working with Mesh Data .................................................. 391
16.1 What’s a mesh? .......................................................... 391
16.2 Supported formats ...................................................... 393
16.3 Mesh Dataset Properties .............................................. 393
16.3.1 Information Properties ............................................... 393
16.3.2 Source Properties .................................................... 394
16.3.3 Symbology Properties ............................................... 394

17 Laying out the maps ....................................................... 399
17.1 Overview of the Print Layout ......................................... 399
17.1.1 Sample Session for beginners ..................................... 399
17.1.2 The Layout Manager .................................................. 400
17.1.3 Menus, tools and panels of the print layout .................. 401
17.2 Layout Items ............................................................ 414
17.2.1 Layout Items Common Options .................................... 414
17.2.2 The Map Item ......................................................... 419
20 Ověřovací systém
20.1 Authentication System Overview
20.1.1 Authentication System Overview
20.1.2 Master password
20.1.3 Authentication database
20.1.4 Authentication Configurations
20.1.5 Master Password and Auth Config Utilities
20.1.6 Using authentication configurations
21 GRASS GIS Integration

21.1 Demo dataset .............................................. 585
21.2 Loading GRASS raster and vector layers .................. 585
21.3 Importing data into a GRASS LOCATION via drag and drop .. 586
21.4 Managing GRASS data in QGIS Browser .................. 586
21.5 GRASS Options ............................................. 586
21.6 Starting the GRASS plugin ................................ 587
21.7 Opening GRASS mapset .................................... 587
21.8 GRASS LOCATION and MAPSET ............................ 587
21.9 Importing data into a GRASS LOCATION ................... 587
   21.9.1 Creating a new GRASS LOCATION ..................... 588
   21.9.2 Adding a new MAPSET ............................... 589
21.10 The GRASS vector data model ............................. 590
21.11 Creating a new GRASS vector layer ...................... 590
21.12 Digitizing and editing a GRASS vector layer .............. 591
21.13 The GRASS region tool .................................. 593
21.14 The GRASS Toolbox ...................................... 593
   21.14.1 Working with GRASS modules ....................... 594
   21.14.2 GRASS module examples .............................. 596
   21.14.3 Customizing the GRASS Toolbox ..................... 600

22 QGIS processing framework

22.1 Úvod ......................................................... 601
22.2 Configuring the Processing Framework ..................... 604
22.3 The Toolbox ................................................ 605
   22.3.1 The algorithm dialog ................................ 606
   22.3.2 Data objects generated by algorithms ............... 612
22.4 The history manager ...................................... 613
   22.4.1 The processing history ............................... 613
   22.4.2 The processing log .................................. 614
22.5 The graphical modeler ..................................... 614
   22.5.1 Definition of inputs .................................. 616
   22.5.2 Definition of the workflow ......................... 618
   22.5.3 Saving and loading models ......................... 621
   22.5.4 Editing a model ..................................... 621
   22.5.5 Editing model help files and meta-information ....... 622
   22.5.6 Exporting a model as a Python script .............. 624
   22.5.7 About available algorithms ......................... 624
22.6 The batch processing interface .......................... 624
   22.6.1 Úvod .................................................. 624
   22.6.2 The parameters table ................................ 624
   22.6.3 Filling the parameters table ......................... 626
   22.6.4 Executing the batch process ......................... 627
22.7 Using processing algorithms from the console .......... 627
   22.7.1 Calling algorithms from the Python console ......... 627
   22.7.2 Creating scripts and running them from the toolbox .. 632
## 22.8 Writing new Processing algorithms as Python scripts

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.8.1</td>
<td>Extending QgsProcessingAlgorithm</td>
<td>635</td>
</tr>
<tr>
<td>22.8.2</td>
<td>The @alg decorator</td>
<td>639</td>
</tr>
<tr>
<td>22.8.3</td>
<td>Input and output types for Processing Algorithms</td>
<td>641</td>
</tr>
<tr>
<td>22.8.4</td>
<td>Handling algorithm output</td>
<td>642</td>
</tr>
<tr>
<td>22.8.5</td>
<td>Communicating with the user</td>
<td>643</td>
</tr>
<tr>
<td>22.8.6</td>
<td>Documenting your scripts</td>
<td>643</td>
</tr>
<tr>
<td>22.8.7</td>
<td>Flags</td>
<td>643</td>
</tr>
<tr>
<td>22.8.8</td>
<td>Best practices for writing script algorithms</td>
<td>643</td>
</tr>
</tbody>
</table>

## 22.9 Configuring external applications

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.9.1</td>
<td>A note for Windows users</td>
<td>644</td>
</tr>
<tr>
<td>22.9.2</td>
<td>A note on file formats</td>
<td>644</td>
</tr>
<tr>
<td>22.9.3</td>
<td>A note on vector layer selections</td>
<td>644</td>
</tr>
<tr>
<td>22.9.4</td>
<td>SAGA</td>
<td>645</td>
</tr>
<tr>
<td>22.9.5</td>
<td>R scripts</td>
<td>646</td>
</tr>
<tr>
<td>22.9.6</td>
<td>R libraries</td>
<td>652</td>
</tr>
<tr>
<td>22.9.7</td>
<td>GRASS</td>
<td>652</td>
</tr>
<tr>
<td>22.9.8</td>
<td>LAStools</td>
<td>653</td>
</tr>
<tr>
<td>22.9.9</td>
<td>OTB Applications</td>
<td>653</td>
</tr>
</tbody>
</table>

## 23 Processing providers and algorithms

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.1</td>
<td>QGIS algorithm provider</td>
<td>655</td>
</tr>
<tr>
<td>23.1.1</td>
<td>Cartography</td>
<td>655</td>
</tr>
<tr>
<td>23.1.2</td>
<td>Databáze</td>
<td>662</td>
</tr>
<tr>
<td>23.1.3</td>
<td>File tools</td>
<td>668</td>
</tr>
<tr>
<td>23.1.4</td>
<td>Graphics</td>
<td>668</td>
</tr>
<tr>
<td>23.1.5</td>
<td>Interpolation</td>
<td>674</td>
</tr>
<tr>
<td>23.1.6</td>
<td>Layer tools</td>
<td>683</td>
</tr>
<tr>
<td>23.1.7</td>
<td>Modeler tools</td>
<td>685</td>
</tr>
<tr>
<td>23.1.8</td>
<td>Network analysis</td>
<td>687</td>
</tr>
<tr>
<td>23.1.9</td>
<td>Rastrová analýza</td>
<td>697</td>
</tr>
<tr>
<td>23.1.10</td>
<td>Raster terrain analysis</td>
<td>714</td>
</tr>
<tr>
<td>23.1.11</td>
<td>Raster tools</td>
<td>725</td>
</tr>
<tr>
<td>23.1.12</td>
<td>Vector analysis</td>
<td>731</td>
</tr>
<tr>
<td>23.1.13</td>
<td>Vector creation</td>
<td>749</td>
</tr>
<tr>
<td>23.1.14</td>
<td>Vector general</td>
<td>767</td>
</tr>
<tr>
<td>23.1.15</td>
<td>Vector geometry</td>
<td>792</td>
</tr>
<tr>
<td>23.1.16</td>
<td>Vector overlay</td>
<td>900</td>
</tr>
<tr>
<td>23.1.17</td>
<td>Vector selection</td>
<td>912</td>
</tr>
<tr>
<td>23.1.18</td>
<td>Vector table</td>
<td>924</td>
</tr>
<tr>
<td>23.2</td>
<td>GDAL algorithm provider</td>
<td>938</td>
</tr>
<tr>
<td>23.2.1</td>
<td>Rastrová analýza</td>
<td>938</td>
</tr>
<tr>
<td>23.2.2</td>
<td>Raster conversion</td>
<td>961</td>
</tr>
<tr>
<td>23.2.3</td>
<td>Raster extraction</td>
<td>967</td>
</tr>
<tr>
<td>23.2.4</td>
<td>Raster miscellaneous</td>
<td>972</td>
</tr>
<tr>
<td>23.2.5</td>
<td>Raster projections</td>
<td>986</td>
</tr>
<tr>
<td>23.2.6</td>
<td>Vector conversion</td>
<td>989</td>
</tr>
<tr>
<td>23.2.7</td>
<td>Vector geoprocessing</td>
<td>994</td>
</tr>
<tr>
<td>23.2.8</td>
<td>Vector miscellaneous</td>
<td>1000</td>
</tr>
<tr>
<td>23.3</td>
<td>LAStools algorithm provider</td>
<td>1008</td>
</tr>
<tr>
<td>23.3.1</td>
<td>blast2dem</td>
<td>1008</td>
</tr>
<tr>
<td>23.3.2</td>
<td>blast2iso</td>
<td>1010</td>
</tr>
<tr>
<td>23.3.3</td>
<td>las2dem</td>
<td>1011</td>
</tr>
<tr>
<td>23.3.4</td>
<td>las2iso</td>
<td>1013</td>
</tr>
<tr>
<td>23.3.5</td>
<td>las2las_filter</td>
<td>1015</td>
</tr>
<tr>
<td>23.3.6</td>
<td>las2las_project</td>
<td>1019</td>
</tr>
<tr>
<td>23.3.7</td>
<td>las2las_transform</td>
<td>1025</td>
</tr>
</tbody>
</table>
24 Zásuvné moduly
24.1 QGIS Plugins .................................................. 1091
  24.1.1 Core and External plugins .................................. 1091
  24.1.2 The Plugins Dialog ........................................ 1092
24.2 Using QGIS Core Plugins ...................................... 1097
  24.2.1 Coordinate Capture Plugin ................................ 1097
  24.2.2 DB Manager Plugin ........................................ 1097
  24.2.3 eVis Plugin .............................................. 1100
  24.2.4 Geometry Checker Plugin ................................ 1110
  24.2.5 Georeferencer Plugin .................................... 1113
  24.2.6 MetaSearch Catalog Client ................................ 1118
  24.2.7 Offline Editing Plugin ................................... 1124
  24.2.8 Topology Checker Plugin ................................ 1124
24.3 QGIS Python console .......................................... 1127
  24.3.1 The Interactive Console .................................. 1127
  24.3.2 The Code Editor .......................................... 1129
  24.3.3 Možnosti ................................................. 1129
25 Pomoc a podpora .................................................. 1131
  25.1 Mailing list .................................................. 1131
  25.1.1 Uživatelské QGIS ......................................... 1131
  25.1.2 Vývojáři QGIS ............................................ 1131
  25.1.3 Komunitní tým QGIS .................................... 1131
  25.1.4 Překlady QGIS ............................................ 1131
  25.1.5 Řídící výbor projektu QGIS (PSC) .......................... 1132
  25.1.6 Uživatelské skupiny QGIS ................................. 1132
  25.2 IRC .......................................................... 1132
  25.3 Commercial support .......................................... 1132
  25.4 BugTracker ................................................... 1132
  25.5 Blog ........................................................ 1133
  25.6 Zásuvné moduly .............................................. 1133
  25.7 Wiki ........................................................ 1133
26 Contributors ....................................................... 1135
  26.1 Authors ....................................................... 1135
  26.2 Translators .................................................... 1136
27 Dodatky ................................................................ 1137
  27.1 Appendix A: GNU General Public License .................... 1137
  27.2 Appendix B: GNU Free Documentation License ................ 1140
  27.3 Appendix C: QGIS File Formats ................................ 1145
    27.3.1 QGS/QGZ - The QGIS Project File Format ................. 1145
    27.3.2 QLR - The QGIS Layer Definition file ................... 1146
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>27.3.3</td>
<td>QML - The QGIS Style File Format</td>
<td>1149</td>
</tr>
<tr>
<td>27.4</td>
<td>Appendix D: QGIS R script syntax</td>
<td>1150</td>
</tr>
<tr>
<td>27.4.1</td>
<td>Inputs</td>
<td>1150</td>
</tr>
<tr>
<td>27.4.2</td>
<td>Outputs</td>
<td>1150</td>
</tr>
<tr>
<td>27.4.3</td>
<td>Syntax Summary for QGIS R scripts</td>
<td>1150</td>
</tr>
<tr>
<td>27.4.4</td>
<td>Examples</td>
<td>1152</td>
</tr>
</tbody>
</table>

28 Literature and Web References 1155
This is the user guide for the geographical information system (GIS) software QGIS. QGIS is subject to the GNU General Public License. More information is available on the QGIS homepage, https://www.qgis.org.

The contents of this document have been written and verified to the best of the knowledge of the authors and editors. Nevertheless, mistakes are possible.

Therefore, the authors, editors and publishers do not take any responsibility or liability for errors in this document and their possible consequences. We encourage you to report possible mistakes.

This document has been typeset with reStructuredText. It is available as reST source code on github, and online as HTML and PDF via https://www.qgis.org/en/docs/. Translated versions of this document can be browsed and downloaded via the documentation area of the QGIS project as well.

For more information about contributing to this document and about translation, please visit https://qgis.org/en/site/getinvolved/index.html.

Odkazy v tomto dokumentu

This document contains internal and external links. Clicking on an internal link moves within the document, while clicking on an external link opens an internet address.

Documentation Authors and Editors

The list of the persons who have contributed with writing, reviewing and translating the following documentation is available at Contributors.

Copyright (c) 2004 - 2020 QGIS Development Team

Internet: https://www.qgis.org

Licence tohoto dokumentu

Je povoleno kopírovat, šířit a/nebo upravovat tento dokument za podmínek GNU Free Documentation License, verze 1.3 nebo jakýchkoliv pozdějších verzí publikovaných nadací Free Software Foundation; bez Invariant Sections, bez úvodních a závěrečných textů. Kopie této licence je zahrnuta v dodatku Appendix B: GNU Free Documentation License.
1.1 What is new in QGIS 3.10

This release of QGIS includes hundreds of bug fixes and many new features and enhancements. We recommend that you use this version over previous releases. For a list of new features, visit the visual changelogs at https://qgis.org/en/site/forusers/visualchangelogs.html.
Vítejte v báječném světě geografických informačních systémů (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).

QGIS aims to be a user-friendly GIS, providing common functions and features. The initial goal of the project was to provide a GIS data viewer. QGIS has reached the point in its evolution where it is being used for daily GIS data-viewing needs, for data capture, for advanced GIS analysis, and for presentations in the form of sophisticated maps, atlases and reports. QGIS supports a wealth of raster and vector data formats, with new format support easily added using the plugin architecture.

QGIS is released under the GNU General Public License (GPL). Developing QGIS under this license means that you can inspect and modify the source code, and guarantees that you, our happy user, will always have access to a GIS program that is free of cost and can be freely modified. You should have received a full copy of the license with your copy of QGIS, and you can also find it in Appendix A: GNU General Public License.

Tip: Aktuální dokumentace

The latest version of this document can always be found in the documentation area of the QGIS website at https://www.qgis.org/en/docs/.
Tato část popisuje jednotné styly, které budou použity v této příručce.

### 3.1 GUI konvence

GUI konvenční styly jsou určeny k napodobení vzhledu grafického uživatelského rozhraní. Obecně platí, že styl bude odrážet non-hover vzhled, takže uživateli stačí zběžně prohlédnout uživatelské rozhraní, aby našel něco, co vypadá jako instruktáž v návodu.

- **Menu Možnosti**: Vrstva 🎨 Přidat vektorovou vrstvu or Nastavení 🎨 Nástrojové lišty 🎨 Digitalizace
- **Nástroj**: Přidat zastrojovou vrstvu
- **Button**: Save as Default
- **Dialogové okno Název**: Vlastnosti vrstvy
- **Záložka**: Obecné
- **Zaškrťvávací pole**: Render
- **Přepinač**: Postgis SRID 🟢EPSG ID
- **Zvolit číslo**: 1.00
- **Zvolit řetězec**: ...
- **Browse for a file**: ...
- **Zvolit barvu**: ...
- **Posuvník**: ...
- **Vložit text**: Display name 🟢 lakes.shp

Stín označuje klikací GUI komponenty.
3.2 Textové nebo klávesové konvence

Příručka též obsahuje styly vztahující se k textu, klávesovým zkratkám a kódování k označení různých subjektů, jako jsou třídy nebo metody. Tyto styly neodpovídají skutečnému vzhledu jakéhokoli textu či kódování v QGIS.

- Hyperlinks: https://qgis.org
- Kombinace kláves: Stisknout Ctrl+B, myšleno stisknout a držet klávesu Ctrl a poté stisknout klávesu B
- Název souboru: jezera.shp
- Název třídy: Nová Vrstva
- Metoda: classFactory
- Server: myhost.de
- Návod: qgis --Nápověda

Řádky kódu jsou označeny pevnou šírkou písma:

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

3.3 Platformy – pokyny

GUI sekvence a malá množství textu mohou být formátovány v řadě za sebou: klikněte na File QGIS Quit to close QGIS. To znamená, že v operačních systémech Linux, Unix a Windows, biste měli nejprve kliknout na nabídku Soubor a poté Zavřít, zatímco u macOS systému, biste měli nejprve kliknout na nabídku QGIS, pak Zavřít.

Větší množství textu může být formátováno jako seznam:

- ![Udělej tohle](image)
- ![Udělej tamto](image)
- ![Iosxl Nebo udělej tamto](image)

nebo jako odstavce:

- ![Udělej tohle a tohle a tohle. Pak udělej tohle a tohle a tohle, a tohle a tohle a tohle, a tohle a tohle a tohle.](image)
- ![Udělej tamto. Pak udělej tamto a tamto a tamto, a tamto a tamto a tamto, tamto a tamto a tamto, a tamto a tamto.](image)

Screenshots that appear throughout the user guide have been created on different platforms.
Funkce

QGIS offers a wealth of GIS functions, provided by core features and plugins. The locator bar makes it easy to search for functions, datasets and more.

A short summary of six general categories of features and plugins is presented below, followed by first insights into the integrated Python console.

4.1 Prohlížení dat

You can view combinations of vector and raster data (in 2D or 3D) in different formats and projections without conversion to an internal or common format. Supported formats include:

- Spatially-enabled tables and views using PostGIS, SpatiaLite and MS SQL Spatial, Oracle Spatial, vector formats supported by the installed OGR library, including GeoPackage, ESRI Shapefile, MapInfo, SDTS, GML and many more. See section Working with Vector Data.
- Rastrové a obrazové formáty podporované nainstalovanou knihovnou GDAL (Geospatial Data Abstraction Library), jako je GeoTIFF, Êrdas IMG, ArcInfo ASCII GRID, JPEG, PNG a mnoho dalších. Viz část Working with Raster Data.
- Mesh data (TINs and regular grids are supported). See Working with Mesh Data.
- GRASS rastrová a vektorová data z GRASS databází (location/mapset). Viz část GRASS GIS Integration.
- Online prostorová data ve formě OGC Web Services, což zahrnuje WMS, WMTS, WCS, WFS a WFS-T. Viz část Working with OGC Data.

The QGIS authentication infrastructure helps you manage user/password, certificates and keys for web services and other resources.

- Spreadsheets (ODS / XLSX)
4.2 Prozkoumávání dat a vytváření map

Můžete vytvářet mapy a interaktivně prozkoumávat prostorová data s přátelským grafickým uživatelským rozhraním. Mezi mnohé užitečné nástroje, které jsou k dispozici v GUI, patří:

- QGIS prohlížeč
- Reprojekce
- Správce databází
- Print layout
- Report
- Přehled panelů
- Prostorové záložky
- Vysvětlky k nástrojům
- Identifikace/výběr prvků
- Editace/zobrazení/vyhledávání vlastností
- Data-defined feature labeling
- Symbolika vektorových a rastrových nástrojů
- Mapové kompozice s vrstvami souřadnicových sítí
- North arrow, scale bar and copyright label for maps
- Podpora pro ukládání a obnovování projektů

4.3 Tvorba, editace, správa a export dat

Můžete vytvářet, upravovat, spravovat a exportovat vektorové a rastrové vrstvy v několika formátech. QGIS nabízí následující:

- Vector digitizing tools
- Ability to create and edit multiple file formats and GRASS vector layers
- Georeferenční zásuvný modul pro geokódování obrázků
- GPS tools to import and export GPX format, and convert other GPS formats to GPX or down/upload directly to a GPS unit (on Linux, usb: has been added to list of GPS devices)
- Podpora pro vizualizaci a editaci dat OpenStreetMap
- Ability to create spatial database tables from files with the DB Manager plugin
- Vylepšená manipulace s tabulkami geodatabáze
- Nástroje pro správu vektorových atributových tabulek
- Možnost uložení screenshotů jako georeferencované obrázky
- Nástroj DXF-Export s rozšířenými možnostmi pro export stylů a zásuvných modulů pro provádění CADovských funkcí
4.4 Analýza dat

You can perform spatial data analysis on spatial databases and other OGR-supported formats. QGIS currently offers vector analysis, raster analysis, sampling, geoprocessing, geometry and database management tools. You can also use the integrated GRASS tools, which include the complete GRASS functionality of more than 400 modules (see section **GRASS GIS Integration**). Or, you can work with the Processing plugin, which provides a powerful geospatial analysis framework to call native and third-party algorithms from QGIS, such as GDAL, SAGA, GRASS, R, and more (see section **Úvod**). All analysis functions are run in the background, allowing you to continue your work before the processing has finished.

The graphical modeller allows you to combine / chain functions into a complete workflow in an intuitive graphical environment.

4.5 Publikování map na internetu

QGIS can be used as a WMS, WMTS, WMS-C or WFS and WFS-T client, and as a WMS, WCS or WFS server (see section **Working with OGC Data**). Additionally, you can publish your data on the Internet using a webserver with QGIS Server, UMN MapServer or GeoServer installed.

4.6 Rozšíření QGIS funkcí prostřednictvím pluginů (zásuvných modulů)

QGIS lze přizpůsobit vaším speciálním potřebám díky rozšiřitelné plugin struktuře a knihovnám, které mohou být použity k vytvoření pluginů. Můžete dokonce vytvářet nové aplikace pomocí C++ nebo Python!

4.6.1 Základní Pluginy

Základní pluginy zahrnují:

1. DB Manager (exchange, edit and view layers and tables from/to databases; execute SQL queries)
2. eVIS (visualize events)
3. Geometry Checker (check geometries for errors)
4. Georeferencer GDAL (add projection information to rasters using GDAL)
5. GPS Tools (load and import GPS data)
6. GRASS 7 (integrate GRASS GIS)
7. MetaSearch Catalogue Client (interacting with metadata catalog services supporting the OGC Catalog Service for the Web (CSW) standard)
8. Offline Editing (allow offline editing and synchronizing with databases)
9. Processing (the spatial data processing framework for QGIS)
10. Topology Checker (find topological errors in vector layers)
4.6.2 Externí Python zásuvné moduly

QGIS nabízí rostoucí počet externích Pythonu pluginů, které jsou poskytovány komunitou. Tyto zásuvné moduly jsou umístěny v oficiálním archivu zásuvných modulů a mohou být snadno nainstalovány pomocí Python Plugin Installeru. Viz oddíl The Plugins Dialog.

4.7 Konzole Python

For scripting, it is possible to take advantage of an integrated Python console, which can be opened with: Plugins Python Console. The console opens as a non-modal utility window. For interaction with the QGIS environment, there is the qgis.utils.iface variable, which is an instance of QgisInterface. This interface provides access to the map canvas, menus, toolbars and other parts of the QGIS application. You can create a script, then drag and drop it into the QGIS window and it will be executed automatically.

For further information about working with the Python console and programming QGIS plugins and applications, please refer to QGIS Python console and PyQGIS-Developer-Cookbook.

4.8 Známé problémy

4.8.1 Omezení počtu otevřených souborů

Pokud otevíráte rozsáhlý QGIS projekt, a jste si jisti, že všechny vrstvy jsou validní, ale některé vrstvy jsou označeny jako špatné, pravděpodobně se potýkáte s tímto problémem. Linux (a jiné OS rovněž) má limit otevřených souborů během procesu. Zdrojové limity jsou od výrobce a dědičné. Příkaz ulimit, který je vestavěný, mění limity pouze pro aktuální proces. Nový limit bude zděděn každým odvozeným procesem.

Můžete vidět všechny aktuální informace o ulimit zadáním:

```
$ ulimit -aS
```

Můžete vidět aktuální povolený počet otevřených souborů od výrobce pomocí následujícího příkazu v konzoli:

```
$ ulimit -Sn
```

Chcete-li změnit limity pro existující relaci, budete moci použít něco jako:

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

Chcete-li je zafixovat navždy

Ve většině linuxových systémů, jsou zdrojové limity nastaveny na přihlášení modulem pam_limits podle nastavení obsažených v /etc/security/limits.conf nebo /etc/security/limits.d/*conf. Pokud máte kořenové oprávnění (root privilege, těž via sudo), měli byste být schopni upravit tyto soubory, ale, aby změny vstoupily v platnost, budete se muset znovu přihlásit.

Více informací:

This chapter provides a quick overview of installing QGIS, downloading QGIS sample data, and running a first simple session visualizing raster and vector data.

## 5.1 Installing QGIS

QGIS project provides different ways to install QGIS depending on your platform.

### 5.1.1 Installing from binaries

Standard installers are available for MS Windows and macOS. Binary packages (rpm and deb) or software repositories are provided for many flavors of GNU/Linux.

For more information and instructions for your operating system check https://download.qgis.org.

### 5.1.2 Installing from source

If you need to build QGIS from source, please refer to the installation instructions. They are distributed with the QGIS source code in a file called INSTALL. You can also find them online at https://github.com/qgis/QGIS/blob/master/INSTALL.md.

If you want to build a particular release and not the version in development, you should replace master with the release branch (commonly in the release-X_Y form) in the above-mentioned link (installation instructions may differ).
5.1.3 Installing on external media

It is possible to install QGIS (with all plugins and settings) on a flash drive. This is achieved by defining a `profiles-path` option that overrides the default `user profile` path and forces QSettings to use this directory, too. See section System Settings for additional information.

5.1.4 Downloading sample data

This user guide contains examples based on the QGIS sample dataset (also called the Alaska dataset).

The Windows installer has an option to download the QGIS sample dataset. If checked, the data will be downloaded to your Documents folder and placed in a folder called GIS Database. You may use Windows Explorer to move this folder to any convenient location. If you did not select the checkbox to install the sample dataset during the initial QGIS installation, you may do one of the following:

- Použijte GIS dat, které již máte
- Download sample data from https://github.com/qgis/QGIS-Sample-Data/archive/master.zip and unzip the archive on any convenient location on your system.
- Odinstalujte službu QGIS a znovu nainstalujte s volbou stažování dat (doporučeno pouze v případě, že výše uvedená řešení nejsou úspěšná)

For GNU/Linux and macOS, there are no dataset installation packages available as rpm, deb or dmg. To use the sample dataset, download it from https://github.com/qgis/QGIS-Sample-Data/archive/master.zip and unzip the archive on any convenient location on your system.

The Alaska dataset includes all GIS data that are used for the examples and screenshots in this user guide; it also includes a small GRASS database. The projection for the QGIS sample datasets is Alaska Albers Equal Area with units feet. The EPSG code is 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]]
```

If you intend to use QGIS as a graphical front end for GRASS, you can find a selection of sample locations (e.g., Spearfish or South Dakota) at the official GRASS GIS website, https://grass.osgeo.org/download/sample-data/.
5.2 Starting and stopping QGIS

QGIS can be started like any other application on your computer. This means that you can launch QGIS by:

- using the Applications menu, the Start menu, or the Dock
- Dvakrát klikněte na ikonu ve složce „Aplikace“ nebo plocha zástupce.
- double clicking an existing QGIS project file (with .qgz or .qgs extension). Note that this will also open the project.
- typing qgis in a command prompt (assuming that QGIS is added to your PATH or you are in its installation folder)

To stop QGIS, use:

- volbamenu Project[ ] Exit QGIS nebo použijte klávesovou zkratku Ctrl+Q
- :menuselection: QGIS –> Quit QGIS nebo použijte klávesovou zkratku Ctrl+Q
- or use the red cross at the top-right corner of the main interface of the application.

5.3 Sample Session: Loading raster and vector layers

Now that you have QGIS installed and a sample dataset available, we will demonstrate a first sample session. In this example, we will visualize a raster and a vector layer. We will use:

- the landcover raster layer (qgis_sample_data/raster/landcover.img)
- and the lakes vector layer (qgis_sample_data/gml/lakes.gml)

Where qgis_sample_data represents the path to the unzipped dataset.

1. Start QGIS as seen in Starting and stopping QGIS.

2. To load the files in QGIS:

   1. Click on the Open Data Source Manager icon. The Data Source Manager should open in Browser mode.
   2. Browse to the folder qgis_sample_data/raster/
   3. Select the ERDAS IMG file landcover.img and double-click it. The landcover layer is added in the background while the Data Source Manager window remains open.
   4. To load the lakes data, browse to the folder qgis_sample_data/gml/., and double-click the lakes.gml file to open it.
   6. Select the NAD27 / Alaska Alberts entry
   7. Click OK
   8. Close the Data Source Manager window

You now have the two layers available in your project in some random colours. Let’s do some customization on the lakes layer.

1. Select the Zoom In tool on the Navigation toolbar
2. Zoom to an area with some lakes
3. Double-click the lakes layer in the map legend to open the Properties dialog
4. To change the lakes color:
Obr. 5.1: Přidávání dat do nového projektu v QGIS
Obr. 5.2: Select the Coordinate Reference System of data
1. Click on the Symbology tab
2. Select blue as fill color.

![Layer Properties - lakes | Symbology](image)

Obr. 5.3: Selecting Lakes color

3. Press OK. Lakes are now displayed in blue in the map canvas.

5. To display the name of the lakes:
   1. Reopen the lakes layer Properties dialog
   2. Click on the Labels tab
   3. Select Single labels in the drop-down menu to enable labeling.
   4. From the Label with list, choose the NAMES field.
   5. Press Apply. Names will now load over the boundaries.

6. You can improve readability of the labels by adding a white buffer around them:
   1. Click the Buffer tab in the list on the left
   2. Check Draw text buffer
   3. Choose 3 as buffer size
   4. Click Apply
   5. Check if the result looks good, and update the value if needed.
   6. Finally click OK to close the Layer Properties dialog and apply the changes.

Let’s now add some decorations in order to shape the map and export it out of QGIS:
1. Select View \(\square\) Decorations \(\square\) Scale Bar menu
2. In the dialog that opens, check \(\square\) Enable Scale Bar option
3. Customize the options of the dialog as you want
4. Press Apply
5. Likewise, from the decorations menu, add more items (north arrow, copyright…) to the map canvas with custom properties.

6. Click Project \(\square\) Import/Export \(\square\) Export Map to Image…
7. Press Save in the opened dialog
8. Select a file location, a format and confirm by pressing Save again.
9. Press Project \(\square\) Save… to store your changes as a .qgz project file.

That’s it! You can see how easy it is to visualize raster and vector layers in QGIS, configure them and generate your map in an image format you can use in other softwares. Let’s move on to learn more about the available functionality, features and settings, and how to use them.

Poznámka: To continue learning QGIS through step-by-step exercises, follow the Training manual.

5.3. Sample Session: Loading raster and vector layers
6.1 Introducing QGIS projects

The state of your QGIS session is called a project. QGIS works on one project at a time. A settings can be project-specific or an application-wide default for new projects (see section Možnosti). QGIS can save the state of your workspace into a QGIS project file using the menu options Project Save or Project Save As.…

Poznámka: If the project has been modified the * symbol will appear in the title bar and QGIS will, by default, ask you if you would like to save the changes. This behavior is controlled by the Prompt to save project and data source changes when required setting under Settings Options General.

You can load existing projects into QGIS from the Browser panel or by through Project Open…. Project New from template or Project Open Recent.

At startup, a list of Project Templates and Recent Projects are displayed, including screenshots, names and file paths (for up to ten projects). The Recent Projects list is handy to access recently used projects. Double-click an entry to open the project or project template. You can also add a layer to create a new project automatically. The lists will then disappear, giving way to the map canvas.

If you want to clear your session and start fresh, go to Project New. This will prompt you to save the existing project if changes have been made since it was opened or last saved.

When you open a fresh project, the title bar will show Untitled Project until you save it.

The information saved in a project file includes:

• Layers added
• Which layers can be queried
• Layer properties, including symbolization and styles
• Projection for the map view
• Last viewed extent
• Print layouts
• Print layout elements with settings
Obr. 6.1: Starting a new project in QGIS

- Print layout atlas settings
- Digitizing settings
- Table Relations
- Project Macros
- Project default styles
- Plugins settings
- QGIS Server settings from the OWS settings tab in the Project properties
- Queries stored in the DB Manager

The project file is saved in XML format (see QGS/QGZ - The QGIS Project File Format). This means that it is possible to edit the file outside of QGIS if you know what you are doing. The project file format has been updated several times. Project files from older QGIS versions may not work properly any more.

**Poznámka:** By default, QGIS will warn you of version differences. This behavior is controlled in the General tab of Settings \(\equiv\) Options (Warn when opening a project file saved with an older version of QGIS).

Whenever you save a .qgs project file in QGIS, a backup of the file is created in the same directory as the project file, with the extension .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.

**Poznámka:** The Auxiliary Storage Properties mechanism makes a zipped project particularly useful, since it embeds auxiliary data.
Projects can also be saved/loaded to/from a PostgreSQL database using the following Project menu items:

- **Project Open from**
- **Project Save to**

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

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

## 6.2 Generating output

There are several ways to generate output from your QGIS session. We have already discussed saving as a project file in *Introducing QGIS projects*. Other ways to produce output files are:

- **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, size, … Georeferencing the image is possible. See *Exporting the map view* for more details.

- **Exporting to PDF files:** **Project Import/Export Export Map to PDF**… outputs the map canvas rendering to PDF at custom scale, resolution, and with some advanced settings (simplification, georeferencing,…). See *Exporting the map view* for more details.

- **Exporting to DXF files:** **Project Import/Export Export Project to DXF**… opens a dialog where you can define the ‘Symbology mode’, the ‘Symbology scale’ and vector layers you want to export to DXF. Through the ‘Symbology mode’, symbols from the original QGIS Symbology can be exported with high fidelity (see section *Creating new DXF files*).

- **Designing maps:** **Project New Print Layout**… opens a dialog where you can layout and print the current map canvas (see section *Laying out the maps*).
The QGIS graphical user interface (GUI) is shown in the figure below (the numbers 1 through 5 in yellow circles indicate important elements of the QGIS GUI, and are discussed below).

Obr. 7.1: QGIS uživatelské rozhraní se vzorovými daty Alaska

**Poznámka:** Vzhled Vašeho okna (záhlaví atd.) se může lišit v závislosti na Vašem operačním systému a správci okna.

The main QGIS GUI (Obr. 7.1) consists of five components / component types:

1. **Menu Bar**
2. **Toolbars**
3. **Panels**
4. **Map View**
5. **Status Bar**
7.1 Menu lišta

The Menu bar provides access to QGIS functions using standard hierarchical menus. The Menus, their options, associated icons and keyboard shortcuts are described below. The keyboard shortcuts can be reconfigured (Settings → Keyboard Shortcuts).

Most menu options have a corresponding tool and vice-versa. However, the Menus are not organized exactly like the toolbars. The locations of menu options in the toolbars are indicated below in the table. Plugins may add new options to Menus. For more information about tools and toolbars, see Nástrojové lišty.

Poznámka: QGIS is a cross-platform application. Tools are generally available on all platforms, but they may be placed in different menus, depending on the operating systems. The lists below show the most common locations, including known variations.

7.1.1 Projekt

The Project menu provides access and exit points for project files. It provides tools to:

- Create a New project file from scratch or use another project file as a template (see Project files options for template configuration)
- Open… a project from a file, a GeoPackage or a PostgreSQL database
- Close a project or revert it to its last saved state
- Save a project in .qgs or .qgz file format, either as a file or within a GeoPackage or PostgreSQL database
- Export the map canvas to different formats or use a print layout for more complex output
- Set project properties and snapping options for geometry editing.
## 7.1.2 Editovat

The *Edit* menu provides most of the native tools needed to edit layer attributes or geometry (see [*Editování*](#) for details).

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zpět</td>
<td>Ctrl+Z</td>
<td>Digitalizace</td>
<td>Zpět a znovu</td>
</tr>
<tr>
<td>Znovu</td>
<td>Ctrl+Shift+Z</td>
<td>Digitalizace</td>
<td>Zpět a znovu</td>
</tr>
<tr>
<td>Vyjmout prvky</td>
<td>Ctrl+X</td>
<td>Digitalizace</td>
<td>Cutting, Copying and Pasting Features</td>
</tr>
<tr>
<td>Kopírovat prvky</td>
<td>Ctrl+C</td>
<td>Digitalizace</td>
<td>Cutting, Copying and Pasting Features</td>
</tr>
</tbody>
</table>

**Pod Mac OS, **Ukončit QGIS příkaz odpovídá QGIS  **Ukončit QGIS** (Cmd+Q).**
### Menu Možnosti

<table>
<thead>
<tr>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vložit prvky</strong></td>
<td>Ctrl+V</td>
<td>Digitalizace</td>
</tr>
<tr>
<td>Paste Features as</td>
<td></td>
<td>Working with the Attribute Table</td>
</tr>
<tr>
<td><strong>New Vector Layer...</strong></td>
<td></td>
<td>Working with the Attribute Table</td>
</tr>
<tr>
<td><strong>Temporary Scratch Layer...</strong></td>
<td>Ctrl+Alt+V</td>
<td>Working with the Attribute Table</td>
</tr>
<tr>
<td><strong>Vybrat</strong></td>
<td>Atributy</td>
<td>Selecting features</td>
</tr>
<tr>
<td><strong>Select Feature(s)</strong></td>
<td>Atributy</td>
<td>Selecting features</td>
</tr>
<tr>
<td><strong>Select Features by Polygon</strong></td>
<td>Atributy</td>
<td>Selecting features</td>
</tr>
<tr>
<td><strong>Select Features by Freehand</strong></td>
<td>Atributy</td>
<td>Selecting features</td>
</tr>
<tr>
<td><strong>Select Features by Radius</strong></td>
<td>Atributy</td>
<td>Selecting features</td>
</tr>
<tr>
<td><strong>Select Features by Value...</strong></td>
<td>F3</td>
<td>Atributy</td>
</tr>
<tr>
<td><strong>Select Features by Expression...</strong></td>
<td>Ctrl+F3</td>
<td>Atributy</td>
</tr>
<tr>
<td><strong>Deselect Features from All Layers</strong></td>
<td>Ctrl+Shift+A</td>
<td>Atributy</td>
</tr>
<tr>
<td><strong>Reselect Features</strong></td>
<td>Atributy</td>
<td>Selecting features</td>
</tr>
<tr>
<td><strong>Select All Features</strong></td>
<td>Ctrl+A</td>
<td>Atributy</td>
</tr>
<tr>
<td><strong>Invert Feature Selection</strong></td>
<td>Atributy</td>
<td>Selecting features</td>
</tr>
<tr>
<td><strong>Add Record</strong></td>
<td>Ctrl+.</td>
<td>Digitalizace</td>
</tr>
<tr>
<td><strong>Add Point Feature</strong></td>
<td>Ctrl+.</td>
<td>Digitalizace</td>
</tr>
<tr>
<td><strong>Add Line Feature</strong></td>
<td>Ctrl+.</td>
<td>Digitalizace</td>
</tr>
<tr>
<td><strong>Add Polygon Feature</strong></td>
<td>Ctrl+.</td>
<td>Digitalizace</td>
</tr>
<tr>
<td><strong>Přidat kruhový řetězec</strong></td>
<td>Shape Digitizing</td>
<td>Add Circular string</td>
</tr>
<tr>
<td><strong>Přidat kruhový řetězec podle poloměru</strong></td>
<td>Shape Digitizing</td>
<td>Add Circular string</td>
</tr>
<tr>
<td><strong>Add Circle</strong></td>
<td>Shape Digitizing</td>
<td>Draw Circles</td>
</tr>
<tr>
<td><strong>Add Circle from 2 Points</strong></td>
<td>Shape Digitizing</td>
<td>Draw Circles</td>
</tr>
<tr>
<td><strong>Add Circle from 3 Points</strong></td>
<td>Shape Digitizing</td>
<td>Draw Circles</td>
</tr>
<tr>
<td><strong>Add Circle from 3 Tangents</strong></td>
<td>Shape Digitizing</td>
<td>Draw Circles</td>
</tr>
<tr>
<td><strong>Add Circle from 2 Tangents and a Point</strong></td>
<td>Shape Digitizing</td>
<td>Draw Circles</td>
</tr>
<tr>
<td><strong>Add Circle by a Center Point and Another Point</strong></td>
<td>Shape Digitizing</td>
<td>Draw Circles</td>
</tr>
<tr>
<td><strong>Add Rectangle</strong></td>
<td>Shape Digitizing</td>
<td>Draw Rectangles</td>
</tr>
</tbody>
</table>

**Pokračujte na další stránce**
<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Rectangle from Extent</td>
<td>![Add Rectangle from Extent]</td>
<td>Shape Digitizing</td>
<td>Draw Rectangles</td>
</tr>
<tr>
<td>Add Rectangle from Center and a Point</td>
<td>![Add Rectangle from Center and a Point]</td>
<td>Shape Digitizing</td>
<td>Draw Rectangles</td>
</tr>
<tr>
<td>Add Rectangle from 3 Points (Distance from 2nd and 3rd point)</td>
<td>![Add Rectangle from 3 Points (Distance from 2nd and 3rd point)]</td>
<td>Shape Digitizing</td>
<td>Draw Rectangles</td>
</tr>
<tr>
<td>Add Rectangle from 3 Points (Distance from projected point on segment p1 and p2)</td>
<td>![Add Rectangle from 3 Points (Distance from projected point on segment p1 and p2)]</td>
<td>Shape Digitizing</td>
<td>Draw Rectangles</td>
</tr>
<tr>
<td>Add Regular Polygon</td>
<td>![Add Regular Polygon]</td>
<td>Shape Digitizing</td>
<td>Draw Regular Polygons</td>
</tr>
<tr>
<td>Add Regular Polygon from Center and a Point</td>
<td>![Add Regular Polygon from Center and a Point]</td>
<td>Shape Digitizing</td>
<td>Draw Regular Polygons</td>
</tr>
<tr>
<td>Add Regular Polygon from Center and a Corner</td>
<td>![Add Regular Polygon from Center and a Corner]</td>
<td>Shape Digitizing</td>
<td>Draw Regular Polygons</td>
</tr>
<tr>
<td>Add Regular Polygon from 2 Points</td>
<td>![Add Regular Polygon from 2 Points]</td>
<td>Shape Digitizing</td>
<td>Draw Regular Polygons</td>
</tr>
<tr>
<td>Add Ellipse</td>
<td>![Add Ellipse]</td>
<td>Shape Digitizing</td>
<td>Draw Ellipses</td>
</tr>
<tr>
<td>Add Ellipse from Center and 2 Points</td>
<td>![Add Ellipse from Center and 2 Points]</td>
<td>Shape Digitizing</td>
<td>Draw Ellipses</td>
</tr>
<tr>
<td>Add Ellipse from Center and a Point</td>
<td>![Add Ellipse from Center and a Point]</td>
<td>Shape Digitizing</td>
<td>Draw Ellipses</td>
</tr>
<tr>
<td>Add Ellipse from Extent</td>
<td>![Add Ellipse from Extent]</td>
<td>Shape Digitizing</td>
<td>Draw Ellipses</td>
</tr>
<tr>
<td>Add Ellipse from Foci</td>
<td>![Add Ellipse from Foci]</td>
<td>Shape Digitizing</td>
<td>Draw Ellipses</td>
</tr>
<tr>
<td>Přesunout prvek/(prvky)</td>
<td>![Přesunout prvek/(prvky)]</td>
<td>Advanced Digitizing</td>
<td>Move Feature(s)</td>
</tr>
<tr>
<td>Copy and Move Feature(s)</td>
<td>![Copy and Move Feature(s)]</td>
<td>Advanced Digitizing</td>
<td>Move Feature(s)</td>
</tr>
<tr>
<td>Delete Selected</td>
<td>![Delete Selected]</td>
<td>Digitalizace</td>
<td>Deleting Selected Features</td>
</tr>
<tr>
<td>Upravit atributy vybraných prvků</td>
<td>![Upravit atributy vybraných prvků]</td>
<td>Digitalizace</td>
<td>Editing attribute values</td>
</tr>
<tr>
<td>Rotovat prvek/(prvky)</td>
<td>![Rotovat prvek/(prvky)]</td>
<td>Advanced Digitizing</td>
<td>Rotovat prvek/(prvky)</td>
</tr>
<tr>
<td>Zjednodušit prvek</td>
<td>![Zjednodušit prvek]</td>
<td>Advanced Digitizing</td>
<td>Zjednodušit prvek</td>
</tr>
<tr>
<td>Přidat prstenec</td>
<td>![Přidat prstenec]</td>
<td>Advanced Digitizing</td>
<td>Přidat prstenec</td>
</tr>
<tr>
<td>Přidat část</td>
<td>![Přidat část]</td>
<td>Advanced Digitizing</td>
<td>Přidat část</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 7.1 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vyplnit prstenec</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Fill Ring</td>
</tr>
<tr>
<td>Smazat prstenec</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Smazat prstenec</td>
</tr>
<tr>
<td>Smazat část</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Smazat část</td>
</tr>
<tr>
<td>Změnit tvar prvků</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Změnit tvar prvků</td>
</tr>
<tr>
<td>Odsazení křivky</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Odsazení křivek</td>
</tr>
<tr>
<td>Rozdělit objekt</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Rozdělit objekt</td>
</tr>
<tr>
<td>Rozdělit část</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Split parts</td>
</tr>
<tr>
<td>Sloučit vybrané prvky</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Sloučit vybrané prvky</td>
</tr>
<tr>
<td>Merge Attributes of Selected Features</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Sloučit atributy vybraných prvků</td>
</tr>
<tr>
<td>Vertex Tool (All Layers)</td>
<td></td>
<td>Digitalizace</td>
<td>Vertex tool</td>
</tr>
<tr>
<td>Vertex Tool (Current Layer)</td>
<td></td>
<td>Digitalizace</td>
<td>Vertex tool</td>
</tr>
<tr>
<td>Rotovat symboly bodů</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Rotovat symboly bodů</td>
</tr>
<tr>
<td>Odsazení bodových symbolů</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Offset Point Symbols</td>
</tr>
<tr>
<td>Reverse Line</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Reverse Line</td>
</tr>
<tr>
<td>Trim/extend Feature</td>
<td></td>
<td>Advanced Digitizing</td>
<td>Trim/Extend Feature</td>
</tr>
</tbody>
</table>

Tools that depend on the selected layer geometry type i.e. point, polyline or polygon, are activated accordingly:

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Bod</th>
<th>Polylíne</th>
<th>Polygon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move Feature(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy and Move Feature(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.1.3 Zobrazit

The map is rendered in map views. You can interact with these views using the View tools (see Working with the map canvas for more information). For example, you can:

- Create new 2D or 3D map views next to the main map canvas
- Zoom or pan to any place
- Query displayed features' attributes or geometry
- Enhance the map view with preview modes, annotations or decorations
- Access any panel or toolbar

The menu also allows you to reorganize the QGIS interface itself using actions like:

- **Toggle Full Screen Mode**: covers the whole screen while hiding the title bar
- **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
- **Toggle Map Only**: hides panels, toolbars, menus and status bar and only shows the map canvas. Combined with the full screen option, it makes your screen display only the map

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Map View</td>
<td>Ctrl+M</td>
<td></td>
<td>Zobrazení mapy</td>
</tr>
<tr>
<td>New 3D Map View</td>
<td>Ctrl+Alt+M</td>
<td></td>
<td>3D Map View</td>
</tr>
<tr>
<td>Posunout mapu</td>
<td></td>
<td>Map Navigation</td>
<td></td>
</tr>
<tr>
<td>Posunout mapu na výběr</td>
<td>Ctrl+Alt++</td>
<td></td>
<td>Zooming and Panning</td>
</tr>
<tr>
<td>Přiblížit</td>
<td>Ctrl+Alt+-</td>
<td></td>
<td>Zooming and Panning</td>
</tr>
<tr>
<td>Oddálit</td>
<td>Ctrl+Shift+I</td>
<td>Atributy</td>
<td>Identifying Features</td>
</tr>
<tr>
<td>Měřit</td>
<td>Ctrl+Shift+M</td>
<td>Atributy</td>
<td>Měření</td>
</tr>
<tr>
<td>Measure Line</td>
<td>Ctrl+Shift+M</td>
<td>Atributy</td>
<td>Měření</td>
</tr>
<tr>
<td>Measure Area</td>
<td>Ctrl+Shift+J</td>
<td>Atributy</td>
<td>Měření</td>
</tr>
<tr>
<td>Measure Angle</td>
<td>Ctrl+Shift+J</td>
<td>Atributy</td>
<td>Měření</td>
</tr>
<tr>
<td>Statistický souhrn</td>
<td>Ctrl+Shift+J</td>
<td>Atributy</td>
<td>Statistical Summary Panel</td>
</tr>
<tr>
<td>Přiblížit na rozměry okna</td>
<td>Ctrl+Shift+F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Přiblížit na výběr</td>
<td>Ctrl+J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Přiblížit na vrstvu</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zoom To Native Resolution (100%)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Tabulka 7.2 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zvětšit podle posledního výřezu</td>
<td></td>
<td>Map Navigation</td>
<td></td>
</tr>
<tr>
<td>Přiblížit na další</td>
<td></td>
<td>Map Navigation</td>
<td></td>
</tr>
<tr>
<td>Dekorace</td>
<td>Alt+V + D</td>
<td></td>
<td>Dekorace</td>
</tr>
<tr>
<td>Grid…</td>
<td></td>
<td></td>
<td>Mřížka</td>
</tr>
<tr>
<td>Scale Bar…</td>
<td></td>
<td></td>
<td>Měřítko</td>
</tr>
<tr>
<td>Image…</td>
<td></td>
<td></td>
<td>Image Decoration</td>
</tr>
<tr>
<td>North Arrow…</td>
<td></td>
<td></td>
<td>Směrová růžice</td>
</tr>
<tr>
<td>Title Label…</td>
<td></td>
<td></td>
<td>Title Label</td>
</tr>
<tr>
<td>Copyright Label…</td>
<td></td>
<td></td>
<td>Copyright Label</td>
</tr>
<tr>
<td>Layout Extents…</td>
<td></td>
<td></td>
<td>Layout Extents</td>
</tr>
<tr>
<td>Režim náhledu</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulate Photocopy (Grayscale)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulate Fax (Mono)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulate Color Blindness (Protanope)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulate Color Blindness (Deuteranope)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show Map Tips</td>
<td></td>
<td></td>
<td>Atributy</td>
</tr>
<tr>
<td>New Spatial Bookmark…</td>
<td>Ctrl+B</td>
<td></td>
<td>Map Navigation</td>
</tr>
<tr>
<td>Show Spatial Bookmarks</td>
<td>Ctrl+Shift+B</td>
<td></td>
<td>Map Navigation</td>
</tr>
<tr>
<td>Show Spatial Bookmark Manager</td>
<td></td>
<td></td>
<td>Prostorové záložky</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td></td>
<td>Map Navigation</td>
</tr>
<tr>
<td>Zobrazit všechny vrstvy</td>
<td>Ctrl+Shift+U</td>
<td></td>
<td>Layers Panel</td>
</tr>
<tr>
<td>Skrýt všechny vrstvy</td>
<td>Ctrl+Shift+H</td>
<td></td>
<td>Layers Panel</td>
</tr>
<tr>
<td>Show Selected Layers</td>
<td></td>
<td></td>
<td>Layers Panel</td>
</tr>
<tr>
<td>Hide Selected Layers</td>
<td></td>
<td></td>
<td>Layers Panel</td>
</tr>
<tr>
<td>Hide Deselected Layers</td>
<td></td>
<td></td>
<td>Layers Panel</td>
</tr>
<tr>
<td>Panely</td>
<td></td>
<td></td>
<td>Panely a nástrojové lišty</td>
</tr>
<tr>
<td>Advanced Digitizing</td>
<td></td>
<td></td>
<td>The Advanced Digitizing panel</td>
</tr>
<tr>
<td>Browser</td>
<td></td>
<td></td>
<td>The Browser Panel</td>
</tr>
<tr>
<td>Browser (2)</td>
<td></td>
<td></td>
<td>The Browser Panel</td>
</tr>
<tr>
<td>GPS Information</td>
<td></td>
<td></td>
<td>Live GPS tracking</td>
</tr>
<tr>
<td>GRASS Tools</td>
<td></td>
<td></td>
<td>GRASS GIS Integration</td>
</tr>
<tr>
<td>Layer Order</td>
<td></td>
<td></td>
<td>Layer Order Panel</td>
</tr>
<tr>
<td>Layer Styling</td>
<td></td>
<td></td>
<td>Layer Styling Panel</td>
</tr>
<tr>
<td>Layers</td>
<td></td>
<td></td>
<td>Layers Panel</td>
</tr>
<tr>
<td>Log Messages</td>
<td></td>
<td></td>
<td>Log Messages Panel</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 7.2 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td></td>
<td>Overview Panel</td>
<td></td>
</tr>
<tr>
<td>Processing Toolbox</td>
<td>The Toolbox</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results Viewer</td>
<td>The Toolbox</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Snapping and Digitizing Options</td>
<td>Nastavení přichytávací tolerance a vyhledání poloměru</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spatial Bookmark Manager</td>
<td>Prostorové záložky</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics</td>
<td></td>
<td>Statistical Summary Panel</td>
<td></td>
</tr>
<tr>
<td>Tile Scale</td>
<td></td>
<td>Sady dlaždic</td>
<td></td>
</tr>
<tr>
<td>Undo/Redo</td>
<td></td>
<td>Undo/Redo Panel</td>
<td></td>
</tr>
<tr>
<td>Nástrojové líšty</td>
<td>Panely a nástrojové líšty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Digitizing Toolbar</td>
<td>Advanced digitizing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attributes Toolbar</td>
<td>Managing Data Source</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Source Manager Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digitizing Toolbar</td>
<td>Digitizing an existing layer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Help Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label Toolbar</td>
<td>The Label Toolbar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Layers Toolbar</td>
<td>Managing Data Source</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Map Navigation Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plugins Toolbar</td>
<td>Zásuvné moduly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Raster Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shape Digitizing Toolbar</td>
<td>Shape digitizing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Snapping Toolbar</td>
<td>Nastavení přichytávací tolerance a vyhledání poloměru</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vector Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Toolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRASS</td>
<td>GRASS GIS Integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Přepnout režim celé obrazovky</td>
<td>F11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toggle Panel Visibility</td>
<td>Ctrl+Tab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toggle Map Only</td>
<td>Ctrl+Shift+Tab</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Under Linux KDE, Panels, Toolbars and Toggle Full Screen Mode are in the Settings menu.

### 7.1.4 Vrstva

The Layer menu provides a large set of tools to create new data sources, add them to a project or save modifications to them. Using the same data sources, you can also:

- **Duplicate** a layer to generate a copy where you can modify the name, style (symbology, labels, …), joins, … The copy uses the same data source as the original.
- **Copy and Paste** layers or groups from one project to another as a new instance whose properties can be modified independently. As for Duplicate, the layers are still based on the same data source.
- **or Embed Layers and Groups** from another project, as read-only copies which you cannot modify (see Vnořování projektů)

The Layer menu also contains tools to configure, copy or paste layer properties (style, scale, CRS…).
<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source Manager</td>
<td>Ctrl+L</td>
<td>Data Source Manager</td>
<td>Opening Data</td>
</tr>
<tr>
<td>Vytvořit vrstvu</td>
<td></td>
<td></td>
<td>Creating new vector layers</td>
</tr>
<tr>
<td>New GeoPackage Layer…</td>
<td>Ctrl+Shift+N</td>
<td>Data Source Manager</td>
<td>Creating a new GeoPackage layer</td>
</tr>
<tr>
<td>New Shapefile Layer…</td>
<td></td>
<td>Data Source Manager</td>
<td>Creating a new Shapefile layer</td>
</tr>
<tr>
<td>New SpatiaLite Layer…</td>
<td></td>
<td>Data Source Manager</td>
<td>Creating a new SpatiaLite layer</td>
</tr>
<tr>
<td>New Temporary Scratch Layer…</td>
<td></td>
<td>Data Source Manager</td>
<td>Creating a new Temporary Scratch Layer</td>
</tr>
<tr>
<td>New Virtual Layer…</td>
<td></td>
<td>Data Source Manager</td>
<td>Creating virtual layers</td>
</tr>
</tbody>
</table>

| Přidat vrstvu | | | Opening Data |
| Add Vector Layer…… | Ctrl+Shift+V | Manage Layers | Loading a layer from a file |
| Add Raster Layer… | Ctrl+Shift+R | Manage Layers | Loading a layer from a file |
| Add Mesh Layer… | | Manage Layers | Loading a mesh layer |
| Add Delimited Text Layer… | Ctrl+Shift+T | Manage Layers | Importing a delimited text file |
| Add PostGIS Layer… | Ctrl+Shift+D | Manage Layers | Database related tools |
| Add SpatiaLite Layer… | Ctrl+Shift+L | Manage Layers | SpatiaLite Layers |
| Add MSSQL Spatial Layer… | | Manage Layers | Database related tools |
| Add Oracle Spatial Layer… | | Manage Layers | Database related tools |
| Add DB2 Spatial Layer… | Ctrl+Shift+2 | Manage Layers | Database related tools |
| Add/Edit Virtual Layer… | | Manage Layers | Creating virtual layers |
| Add WMS/WMTS Layer… | Ctrl+Shift+W | Manage Layers | Loading WMS/WMTS Layers |
| Add ArcGIS MapServer Layer… | | Manage Layers | |
| Add WCS Layer… | | Manage Layers | WCS Client |
| Add WFS Layer… | | Manage Layers | WFS and WFS-T Client |
| Add ArcGIS FeatureServer Layer… | | Manage Layers | |

Přiložit vrstvy a skupiny…
Přidat ze souboru definice vrstvy…
Přidat ze souboru definice vrstvy…

Pokračujte na další stránce
### Menu Možnosti

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Otevřít atributovou tabulku</td>
<td>F6</td>
<td>Atributy</td>
<td>Working with the Attribute Table</td>
</tr>
<tr>
<td>Přepnut editaci</td>
<td></td>
<td>Digitalizace</td>
<td>Digitizing an existing layer</td>
</tr>
<tr>
<td>Uložit změny vrstvy</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Aktuální změny</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Save for Selected Layer(s)</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Rollback for Selected Layer(s)</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Cancel for Selected Layer(s)</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Save for all Layers</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Rollback for all Layers</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Cancel for all Layers</td>
<td></td>
<td>Digitalizace</td>
<td>Saving Edited Layers</td>
</tr>
<tr>
<td>Save As…</td>
<td></td>
<td></td>
<td>Creating new layers from an existing layer</td>
</tr>
<tr>
<td>Odstranit vrstvu/skupinu</td>
<td>Ctrl+D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplikovat vrstvu(vrstvy)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nastavit měřítko viditelnosti vrstvy(vrstev)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nastavit SRS vrstvy(vrstev)</td>
<td>Ctrl+Shift+C</td>
<td></td>
<td>Layer Coordinate Reference Systems</td>
</tr>
<tr>
<td>Nastaví projekt CRS z vrstvy</td>
<td></td>
<td></td>
<td>Project Coordinate Reference Systems</td>
</tr>
<tr>
<td>Layer Properties…</td>
<td></td>
<td></td>
<td>The Vector Properties Dialog, Raster Properties Dialog, Mesh Dataset Properties</td>
</tr>
<tr>
<td>Filtrovat…</td>
<td>Ctrl+F</td>
<td></td>
<td>Query Builder</td>
</tr>
<tr>
<td>Tvorba popisků</td>
<td></td>
<td></td>
<td>Labels Properties</td>
</tr>
<tr>
<td>Show in Overview</td>
<td></td>
<td></td>
<td>Overview Panel</td>
</tr>
<tr>
<td>Show All in Overview</td>
<td></td>
<td></td>
<td>Overview Panel</td>
</tr>
<tr>
<td>Hide All from Overview</td>
<td></td>
<td></td>
<td>Overview Panel</td>
</tr>
</tbody>
</table>

### 7.1.5 Nastavení

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Profiles</td>
<td>Working with User Profiles</td>
</tr>
<tr>
<td>default</td>
<td>Working with User Profiles</td>
</tr>
<tr>
<td>Open Active Profile Folder</td>
<td>Working with User Profiles</td>
</tr>
<tr>
<td>New Profile…</td>
<td>Working with User Profiles</td>
</tr>
<tr>
<td>Style Manager…</td>
<td>The Style Manager</td>
</tr>
<tr>
<td>Custom Projections…</td>
<td>Vlastní souřadnický referenční systém</td>
</tr>
<tr>
<td>Keyboard Shortcuts…</td>
<td>Keyboard shortcuts</td>
</tr>
<tr>
<td>Interface Customization…</td>
<td>Přizpůsobení</td>
</tr>
<tr>
<td>Možnosti…</td>
<td>Možnosti</td>
</tr>
</tbody>
</table>

Under Linux KDE, you’ll find more tools in the Settings menu such as Panels, Toolbars and Toggle Full Screen Mode.
7.1.6 Zásuvné moduly

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spravovat a instalovat zásuvné moduly...</td>
<td></td>
<td></td>
<td>The Plugins Dialog</td>
</tr>
<tr>
<td>Python Console</td>
<td>Ctrl+Alt+P</td>
<td>Plugins</td>
<td>QGIS Python console</td>
</tr>
</tbody>
</table>

Při prvním spuštění QGIS nejsou načteny všechny základní zásuvný moduly.

7.1.7 Vektor

This is what the Vector menu looks like if all core plugins are enabled.

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate Capture</td>
<td></td>
<td>Vector</td>
<td>Coordinate Capture Plugin</td>
</tr>
<tr>
<td>Check Geometries...</td>
<td></td>
<td>Vector</td>
<td>Geometry Checker Plugin</td>
</tr>
<tr>
<td>GPS Tools</td>
<td>Alt+O+G</td>
<td>Vector</td>
<td>GPS Plugin</td>
</tr>
<tr>
<td>Topology Checker</td>
<td>Alt+O+G</td>
<td>Vector</td>
<td>Topology Checker Plugin</td>
</tr>
<tr>
<td>Geoprocessing Tools</td>
<td>Alt+O+G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Buffer...</td>
<td></td>
<td>Buffer</td>
<td></td>
</tr>
<tr>
<td>Clip...</td>
<td></td>
<td>Clip</td>
<td></td>
</tr>
<tr>
<td>Convex Hull...</td>
<td></td>
<td>Convex hull</td>
<td></td>
</tr>
<tr>
<td>Difference...</td>
<td></td>
<td>Difference</td>
<td></td>
</tr>
<tr>
<td>Dissolve...</td>
<td></td>
<td>Dissolve</td>
<td></td>
</tr>
<tr>
<td>Intersection...</td>
<td></td>
<td>Intersection</td>
<td></td>
</tr>
<tr>
<td>Symmetrical Difference...</td>
<td></td>
<td>Symmetrical difference</td>
<td></td>
</tr>
<tr>
<td>Union...</td>
<td></td>
<td>Union</td>
<td></td>
</tr>
<tr>
<td>Eliminate Selected Polygons...</td>
<td></td>
<td>Eliminate selected polygons</td>
<td></td>
</tr>
<tr>
<td>Geometry Tools</td>
<td>Alt+O+E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centroids...</td>
<td></td>
<td>Centroids</td>
<td></td>
</tr>
<tr>
<td>Collect Geometries...</td>
<td></td>
<td>Collect geometries</td>
<td></td>
</tr>
<tr>
<td>Extract Vertices...</td>
<td></td>
<td>Extract vertices</td>
<td></td>
</tr>
<tr>
<td>Multipart to Singleparts...</td>
<td></td>
<td>Multipart to singleparts</td>
<td></td>
</tr>
<tr>
<td>Polygons to Lines...</td>
<td></td>
<td>Polygons to lines</td>
<td></td>
</tr>
<tr>
<td>Simplify...</td>
<td></td>
<td>Simplify</td>
<td></td>
</tr>
<tr>
<td>Check Validity...</td>
<td></td>
<td>Check validity</td>
<td></td>
</tr>
<tr>
<td>Delaunay Triangulation...</td>
<td></td>
<td>Delaunay triangulation</td>
<td></td>
</tr>
<tr>
<td>Densify by Count...</td>
<td></td>
<td>Densify by count</td>
<td></td>
</tr>
<tr>
<td>Add Geometry Attributes...</td>
<td></td>
<td>Add geometry attributes</td>
<td></td>
</tr>
<tr>
<td>Lines to Polygons...</td>
<td></td>
<td>Lines to polygons</td>
<td></td>
</tr>
<tr>
<td>Voronoi Polygons...</td>
<td></td>
<td>Voronoi polygons</td>
<td></td>
</tr>
<tr>
<td>Analysis Tools</td>
<td>Alt+O+A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line Intersection...</td>
<td></td>
<td>Line intersections</td>
<td></td>
</tr>
<tr>
<td>Mean Coordinate(s)...</td>
<td></td>
<td>Mean coordinate(s)</td>
<td></td>
</tr>
<tr>
<td>Basic Statistics for Fields...</td>
<td></td>
<td>Basic statistics for fields</td>
<td></td>
</tr>
<tr>
<td>Count Points in Polygon...</td>
<td></td>
<td>Count points in polygon</td>
<td></td>
</tr>
<tr>
<td>Distance Matrix...</td>
<td></td>
<td>Distance matrix</td>
<td></td>
</tr>
<tr>
<td>List Unique Values...</td>
<td></td>
<td>List unique values</td>
<td></td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
By default, QGIS adds Processing algorithms to the Vector menu, grouped by sub-menus. This provides shortcuts for many common vector-based GIS tasks from different providers. If not all these sub-menus are available, enable the Processing plugin in Plugins ▶️ Manage and Install Plugins….

Note that the list of the Vector menu tools can be extended with any Processing algorithms or some external plugins.

### 7.1.8 Rastr

This is what the Raster menu looks like if all core plugins are enabled.
By default, QGIS adds Processing algorithms to the Raster menu, grouped by sub-menus. This provides a shortcut for many common raster-based GIS tasks from different providers. If not all these sub-menus are available, enable the Processing plugin in Plugins Manage and Install Plugins.

Note that the list of the Raster menu tools can be extended with any Processing algorithms or some external plugins.

### 7.1.9 Databáze

This is what the Database menu looks like if all the core plugins are enabled. If no database plugins are enabled, there will be no Database menu.

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>eVis</td>
<td>Alt + D + E</td>
<td>eVis Plugin</td>
<td></td>
</tr>
<tr>
<td>eVis Database Connection</td>
<td></td>
<td>Databáze</td>
<td>eVis Plugin</td>
</tr>
<tr>
<td>eVis Event Id Tool</td>
<td></td>
<td>Databáze</td>
<td>eVis Plugin</td>
</tr>
<tr>
<td>eVis Event Browser</td>
<td></td>
<td>Databáze</td>
<td>eVis Plugin</td>
</tr>
<tr>
<td>Offline editing…</td>
<td>Alt + D + 0</td>
<td>Offline Editing Plugin</td>
<td></td>
</tr>
<tr>
<td>Convert to Offline Project…</td>
<td></td>
<td>Databáze</td>
<td>Offline Editing Plugin</td>
</tr>
<tr>
<td>Synchronize</td>
<td></td>
<td>Databáze</td>
<td>Offline Editing Plugin</td>
</tr>
<tr>
<td>DB Manager…</td>
<td></td>
<td>Databáze</td>
<td>DB Manager Plugin</td>
</tr>
</tbody>
</table>

Při prvním spuštění QGIS nejsou načteny všechny základní zásuvný moduly.
7.1.10 Web

This is what the Web menu looks like if all the core plugins are enabled. If no web plugins are enabled, there will be no Web menu.

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaSearch</td>
<td>Alt+W + M</td>
<td>MetaSearch Catalog Client</td>
<td></td>
</tr>
<tr>
<td>Help</td>
<td>Web</td>
<td>MetaSearch Catalog Client</td>
<td></td>
</tr>
</tbody>
</table>

Při prvním spuštění QGIS nejsou načteny všechny základní zásuvný moduly.

7.1.11 Mesh

The Mesh menu provides tools needed to manipulate mesh layers.

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh Calculator…</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.12 Zpracování

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolbox</td>
<td>Ctrl+Alt+T</td>
<td>The Toolbox</td>
<td></td>
</tr>
<tr>
<td>Grafický modelář…</td>
<td>Ctrl+Alt+G</td>
<td>The graphical modeler</td>
<td></td>
</tr>
<tr>
<td>History…</td>
<td>Ctrl+Alt+H</td>
<td>The history manager</td>
<td></td>
</tr>
<tr>
<td>Results Viewer</td>
<td>Ctrl+Alt+R</td>
<td>Configuring external applications</td>
<td></td>
</tr>
<tr>
<td>Edit Features In-Place</td>
<td>Ctrl+Alt+R</td>
<td>The Processing in-place layer modifier</td>
<td></td>
</tr>
</tbody>
</table>

Při prvním spuštění QGIS nejsou načteny všechny základní zásuvný moduly.
7.1.13 Nápověda

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obsah nápovědy</td>
<td>F1</td>
<td>Nápověda</td>
<td></td>
</tr>
<tr>
<td>API dokumentace</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plugins</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nahlásit problém</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Potřebujete komerční podporu?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QGIS domovská stránka</td>
<td>Ctrl+H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check QGIS Version</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>About</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QGIS Sustaining Members</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.14 QGIS

Toto menu je přístupné pod macOS a obsahuje některé OS spojené příkazy.

<table>
<thead>
<tr>
<th>Menu Možnosti</th>
<th>Zkratka</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preference</td>
<td></td>
</tr>
<tr>
<td>O QGIS</td>
<td></td>
</tr>
<tr>
<td>Skrýt QGIS</td>
<td></td>
</tr>
<tr>
<td>Ukázat vše</td>
<td></td>
</tr>
<tr>
<td>Skrýt ostatní</td>
<td></td>
</tr>
<tr>
<td>Ukončit QGIS</td>
<td>Cmd+Q</td>
</tr>
</tbody>
</table>

Preferences correspond to Settings, About QGIS corresponds to Help, About and Quit QGIS corresponds to Project Exit QGIS for other platforms.

7.2 Panely a nástrojové lišty

From the View menu (or Settings), you can switch QGIS widgets (Panels) and toolbars (Toolbars) on and off. To (de)activate any of them, right-click the menu bar or toolbar and choose the item you want. Panels and toolbars can be moved and placed wherever you like within the QGIS interface. The list can also be extended with the activation of Core or external plugins.

7.2.1 Nástrojové lišty

The toolbars provide access to most of the functions in the menus, plus additional tools for interacting with the map. Each toolbar item has pop-up help available. Hover your mouse over the item and a short description of the tool’s purpose will be displayed.

Každou nástrojovou lišťu lze pohybovat podle vašich potřeb. Navíc každou lišťu lze vypnout pomocí pravého tlačítka myši kontextového menu nebo podržením myši nad nástrojovou lišťou.

Tip: Obnovení panelu nástrojů
Obr. 7.2: Nástrojové lišty menu
If you have accidentally hidden a toolbar, you can get it back using View Toolbars (or Settings Toolbars). If, for some reason, a toolbar (or any other widget) totally disappears from the interface, you’ll find tips to get it back at restoring initial GUI.

### 7.2.2 Panely

QGIS provides many panels. Panels are special widgets that you can interact with (selecting options, checking boxes, filling values…) to perform more complex tasks.

Below is a list of the default panels provided by QGIS:

- **Výhodná digitalizace**
- the **Browser Panel**
- the **GPS informační panel**
- the **Identifikační panel**
- **Panel pořadí vrstev**
- the **Layer Styling Panel**
- the **Layers Panel**
- the **Log Messages Panel**
- the **Overview Panel**
- the **Toolbox zpracování**
- the **Result Viewer Panel**
- the **Spatial Bookmark Manager Panel**
- the **Statistics Panel**
- the **Měřítkový panel**
- the **Undo/Redo Panel**

### 7.3 Zobrazení mapy

#### 7.3.1 Exploring the map view

The map view (also called **Map canvas**) is the „business end“ of QGIS — maps are displayed in this area. The map displayed in this window will reflect the rendering (symbology, labeling, visibilities…) you applied to the layers you have loaded. It also depends on the layers and the project’s Coordinate Reference System (CRS).

When you add a layer (see e.g. Opening Data), 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.

While the focus is over the map view:

- it can be panned, shifting the display to another region of the map: this is performed using the Pan tool, the arrow keys, moving the mouse while any of the Space key, the middle mouse button or the mouse wheel is held down.
Obr. 7.3: Panelové menu

7.3. Zobrazení mapy
• it can be zoomed in and out, with the dedicated Zoom In and Zoom Out tools. This is also performed by rolling the wheel forward to zoom in and backwards to zoom out. The zoom is centered on the mouse cursor position.

You can customize the Zoom factor under the Settings ▾ Options ▾ Map tools menu.

• it can be zoomed to the full extent of loaded layers (Zoom Full), to a layer extent (Zoom to Layer) or to the extent of selected features (Zoom to Selection)

• you can navigate back/forward through canvas view history with the Zoom Last and Zoom Next buttons or using the back/forward mouse buttons.

By default, QGIS opens a single map view (so called „main map”), which is tightly bound to the Layers panel; the main map automatically reflects the changes you make in the Layers panel area. But it’s possible to have additional map views to preview different renderings of your dataset, side by side; while still relying on the layers properties as set in the Layers panel, each map view can display a different set of layers at different scale and extent.

7.3.2 Setting additional map views

To add a new map view, go to View ▾ New Map View. A new floating widget displaying the layers 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.

Obr. 7.4: Multiple map views with different settings

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

• Zoom Full, Zoom to Selection and Zoom to Layer 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…

### 7.3.3 Exporting the map view

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:

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:
Obr. 7.5: The Save Map as Image dialog

- **Export RDF metadata** of the document such as the title, author, date, description…

- **Create Geospatial PDF (GeoPDF):** Generate a georeferenced PDF file (requires GDAL version 3 or later). 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.

| Poznámka: | Since QGIS 3.10, with GDAL 3 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/](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 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).

- 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…
Obr. 7.6: The Save Map as PDF dialog
7.4 3D Map View

3D visualization support is offered through the 3D map view. You create and open a 3D map view via View \(\text{New 3D Map View}\). A floating QGIS panel will appear. The panel can be docked.

To begin with, the 3D map view has the same extent and view as the 2D main map canvas. A set of navigation tools are available to turn the view into 3D.

![3D Map View](image)

Obr. 7.7: 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
- **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
- **Configure**: the map view settings
7.4.1 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.

7.4.2 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 Repeat button pressed will repeatedly run the animation while clicking ⏯ stops a running animation.

It is possible to browse the different views of the camera, using the Keyframe list. Whenever a time is active, changing the map view will automatically update the associated position. You can also Edit keyframe (time only) or Remove keyframe.

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.

### 7.4.3 Scene Configuration

The 3D map view opens with some default settings you can customize. To do so, click the Configure… button at the top of the 3D canvas panel to open the 3D configuration window.

![3D Configuration](image)

Obr. 7.8: The 3D Map Configuration dialog
In the 3D Configuration window there are various options to fine-tune the 3D scene:

- **Camera's Field of view**: allowing to create panoramic scenes. Default value is 45°.
- **Terrain**: Before diving into the details, it is worth noting that 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 Flat terrain
    - a loaded DEM (Raster Layer)
    - an Online service, loading elevation tiles produced by Mapzen tools – more details at https://registry.opendata.aws/terrain-tiles/
  - **Elevation**: Raster layer to be used for generation of the terrain. This layer must contain a band that represents elevation.
  - **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.
  - **Map theme**: Allows you to select the set of layers to display in the map view from predefined map themes.

- **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**: You can add up to eight point lights, each with a particular position (in X, Y and Z), Color, Intensity and Attenuation.

![Obr. 7.9: The 3D Map Lights Configuration dialog](image-url)
• **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 labels:** 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**

### 7.4.4 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.

### 7.5 Stavová lišta

The status bar provides you with general information about the map view and processed or available actions, and offers you tools to manage the map view.

On the left side of the status bar, the locator bar, a quick search widget, helps you find and run any feature or options in QGIS. Simply type text associated with the item you are looking for (name, tag, keyword…) and you get a list that updates as you write. You can also limit the search scope using **locator filters**. Click the **button to select any of them and press the Configure entry for global settings.

In the area next to the locator bar, a summary of actions you’ve carried out will be shown when needed (such as selecting features in a layer, removing layer) or a long description of the tool you are hovering over (not available for all tools).

In case of lengthy operations, such as gathering of statistics in raster layers, executing Processing algorithms or rendering several layers in the map view, a progress bar is displayed in the status bar.

The **Coordinate** option shows the current position of the mouse, following it while moving across the map view. You can set the units (and precision) in the Project Properties… General tab. Click on the small button at the left of the textbox to toggle between the Coordinate option and the **Extents** option that displays the coordinates of the current bottom-left and top-right corners of the map view in map units.

Next to the coordinate display you will find the **Scale** display. It shows the scale of the map view. There is a scale selector, which allows you to choose between predefined and custom scales.

On the right side of the scale display, press the **button to lock the scale to use the magnifier to zoom in or out. The magnifier allows you to zoom in to a map without altering the map scale, making it easier to tweak the positions of labels and symbols accurately. The magnification level is expressed as a percentage. If the Magnifier has a level of 100%, then the current map is not magnified. Additionally, a default magnification value can be defined within
**Settings** \(\rightarrow\) **Options** \(\rightarrow\) **Rendering** \(\rightarrow\) **Rendering behavior**, which is very useful for high-resolution screens to enlarge small symbols.

To the right of the magnifier tool you can define a current clockwise rotation for your map view in degrees.

On the right side of the status bar, there is a small checkbox which can be used temporarily to prevent layers being rendered to the map view (see section *Vykreslování*).

To the right of the render functions, you find the \(\rightarrow\) **EPSG:code** button showing the current project CRS. Clicking on this opens the *Project Properties* dialog and lets you apply another CRS to the map view.

The \(\rightarrow\) **Messages** button next to it opens the *Log Messages Panel* which has information on underlying processes (QGIS startup, plugins loading, processing tools…)

Depending on the *Plugin Manager settings*, the status bar can sometimes show icons to the right to inform you about availability of \(\rightarrow\) new or \(\rightarrow\) upgradeable plugins. Click the icon to open the Plugin Manager dialog.

---

**Tip: Výpočet správného měřítka vašeho mapového plátna**

When you start QGIS, the default CRS is WGS 84 (EPSG 4326) and units are degrees. This means that QGIS will interpret any coordinate in your layer as specified in degrees. To get correct scale values, you can either manually change this setting in the *General* tab under *Project Properties*... (e.g. to meters), or you can use the \(\rightarrow\) **EPSG:code** icon seen above. In the latter case, the units are set to what the project projection specifies (e.g., \(+\text{units}=\text{us-ft}\)).

Všimněte si, že volba CRS při spuštění může být nastavena v *menuselection: „Nastavení -> Volba -> CRS“.*
The Browser panel

The QGIS Browser panel is a great tool for browsing, searching, inspecting, copying and loading QGIS resources. Only resources that QGIS knows how to handle are shown in the browser.

Using the Browser panel you can locate, inspect and add data, as described in *The Browser Panel*. In addition, the Browser panel supports drag and drop of many QGIS resources, such as project files, Python scripts, Processing scripts and Processing models.

Python scripts, Processing scripts and Processing models can also be opened for editing in an external editor and the graphical modeller.

You can drag and drop layers from the *Layers* panel to the *Browser* panel, for instance into a GeoPackage or a PostGIS database.

The browser panel (Obr. 8.1) is organised as an expandable hierarchy with some fixed top-level entries that organise the resources handled by the browser. Node entries are expanded by clicking on ▼ to the left of the entry name. A branch is collapsed by clicking on ▲. The Collapse All button collapses all top-level entries.

A filter (Filter Browser) can be used for searching based on entry names (both leaf entries and node entries in the hierarchy). Using the Options pull-down menu next to the filter text field, you can

- toggle *Case Sensitive* search
- set the *Filter pattern syntax* to one of
  - *Normal*
  - *Wildcard(s)*
  - *Regular Expressions*

The *Properties widget*, showing useful information about some entries / resources, can be enabled / disabled using the Enable/disable properties widget button. When enabled, it opens at the bottom of the browser panel, as shown in Obr. 8.2.

A second browser panels 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 located deep down in different branches of the browser hierarchy.
Obr. 8.1: The Browser panel
Obr. 8.2: The properties widget
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
- Copy entries

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 Favourite.

8.2.2 Prostorové záložky

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

8.2.3 Home

Your file system home directory / folder. By right-clicking on an entry, and choosing Add as a Favorite, the location will be added to Favorites. From the context menu, you can also

- add a directory, Geopackage or ESRI Shapefile format dataset (Add)
- hide the directory (Hide from Browser)
- toggle Fast Scan this Directory
- open the directory in your file manager (Open Directory)
- open the directory in a terminal window (Open in Terminal)
- inspect properties (Properties…, Directory Properties…)
8.2.4 / 

Your file system root directory / folder.

8.2.5 Geopackage

Geopackage files / databases. From the top level context menu, you can create a Geopackage file / database (Create Database…) or add an existing Geopackage file / database (New Connection…).

The context menu of each Geopackage lets you remove it from the list (Remove connection…), add a new layer or table to the Geopackage (Create new Layer or Table…), delete the Geopackage (Delete <name of geopackage>) and Compact Database (VACUUM).

For layer/table entries you can

- rename it (Rename Layer <layer name>…)
- export it (Export Layer To file)
- add it to the project Add Layer to Project
- delete it (Delete Layer)
- inspect properties (Layer Properties…, File Properties…)

8.2.6 SpatiaLite

SpatiaLite database connections.

From the top level context menu, you can create a SpatiaLite file / database (Create Database…) or add an existing SpatiaLite file / database (New Connection…).

The context menu of each SpatiaLite file lets you delete it (Delete).

For layer/table entries you can

- export it (Export Layer To file)
- add it to the project Add Layer to Project
- delete it (Delete Layer)
- inspect properties (Layer Properties…)

8.2.7 PostGIS

PostGIS database connections.

From the top level context menu, you can add a new connection (New Connection…).

The context menu of each connection lets you Refresh it, edit it Edit connection…., delete it (Delete connection) or Create Schema….

The context menu of each schema lets you Refresh, Rename Schema… or Delete Schema.

For layers/tables you can

- rename it (Rename Table…)
- remove its contents (Truncate Table…)
- export it (Export Layer To file)
- add it to the project (Add Layer to Project)
- delete it (Delete Layer)
• inspect its properties (Layer Properties…)

8.2.8 MSSQL

Microsoft SQL Server connections.

From the top level context menu, you can add a new connection (New Connection…).

The context menu of each connection lets you Refresh it, edit it Edit connection…, delete it (Delete connection) or Create Schema….

The context menu of each schema lets you Refresh, Rename Schema… or Delete Schema.

For layers/tables you can

• rename it (Rename Table…)
• remove its contents (Truncate Table…)
• export it (Export Layer To file)
• add it to the project (Add Layer to Project)
• delete it (Delete Layer)
• inspect its properties (Layer Properties…)

8.2.9 DB2

IBM DB2 database connections.

From the top level context menu, you can add a new connection (New Connection…).

The context menu of each connection lets you Refresh it, edit it Edit connection…, delete it (Delete connection) or Create Schema….

The context menu of each schema lets you Refresh, Rename Schema… or Delete Schema.

For layers/tables you can

• rename it (Rename Table…)
• remove its contents (Truncate Table…)
• export it (Export Layer To file)
• add it to the project (Add Layer to Project)
• delete it (Delete Layer)
• inspect its properties (Layer Properties…)

8.2.10 WMS/WMTS

Web Map Services (WMS) and Web Map Tile Services (WMTS)

From the top level context menu, you can add a new connection (New Connection…).

The context menu of each WSM/WMTS service lets you Refresh it, Edit… it and delete it (Delete).

Group layers can be added by dragging them onto the map canvas.

For WMS/WMTS layer entries you can

• export it (Export Layer To file)
• add it to the project (Add Layer to Project)
• inspect properties (Layer Properties…)

Kapitola 8. The Browser panel
8.2.11 Vector Tiles

Vector tile services

From the top level context menu, you add an existing service (*New Connection*...), and you can *Save Connections*... or *Load Connections*... to / from XML files.

8.2.12 XYZ Tiles

XYZ tile services

From the top level context menu, you add an existing service (*New Connection*...), and you can *Save Connections*... or *Load Connections*... to / from XML files.

For the XYZ tile service entries you can

- edit it (*Edit*...)
- delete it (*Delete*)
- export it (*Export Layer* [!] To file)
- add it to the project *Add Layer to Project*
- inspect properties (*Layer Properties*...)

8.2.13 WCS

Web Coverage Services

From the top level context menu, you can add a new connection (*New Connection*...).

The context menu of each WCS lets you *Refresh* it, *Edit*... it and delete it (*Delete*).

For WCS layer entries you can

- export it (*Export Layer* [!] To file)
- add it to the project (*Add Layer to Project*)
- inspect properties (*Layer Properties*...)

8.2.14 WFS / OGC API - Features

*Web Feature Services* (WFS) and *OGC API - Features services* (aka WFS3)

From the top level context menu, you can add a new connection (*New Connection*...).

The context menu of each WFS lets you *Refresh* it, *Edit*... it and delete it (*Delete*).

For WFS layer entries you can

- export it (*Export Layer* [!] To file)
- add it to the project (*Add Layer to Project*)
- inspect properties (*Layer Properties*...)

8.2. Browser panel top-level entries
8.2.15 OWS

Here you will find a read-only list of all your Open Web Services (OWS) - WMS / WCS / WFS / …

8.2.16 ArcGIS Map Service

8.2.17 ArcGIS Features Service

8.2.18 GeoNode

From the top level context menu, you can add a new connection (New Connection…).
The context menu of each service lets you Refresh it, Edit… it and delete it (Delete).

For the service layer entries you can

- export it (Export Layer To file)
- add it to the project (Add Layer to Project)
- inspect properties (Layer Properties…)

8.3 Resources

- Project files. The context menu for QGIS project files allows you to:
  - open it (Open Project)
  - extract symbols (Extract Symbols…) - open 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 layers 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...)

- 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...)
QGIS is highly configurable. Through the Settings menu, it provides different tools to:

- **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 Možnosti

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.

**Poznámka:** 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 General Settings

**Override System Locale**

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.

**Aplikace**

- 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 *Výběr barvy*)
- **Modeless data source manager dialog** to keep the data source manager dialog opened and allow interaction with QGIS interface while adding layers to project

**Soubory projektu**

- **Open project on launch**
  - '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.
- **Vytvořit nový projekt z výchozího projektu. Máte možnost vybrat Nástavbu aktuální projekt jako výchozí nebo Obnovit výchozí. Můžete procházet soubory a definovat adresář, kde najdete své uživatelsky definované šablony projektů. Ty budou přidávány z Projektů Nový ze šablony. V případě, že nejprve aktivujete Vytvořit nový projekt z výchozího projektu a poté uložíte projekt do složky šablony projektů.**
- **Prompt to save project and data source changes when required** to avoid losing changes you made.
- **Žádat o upozornění při odstraňování vrstvy**
• 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)'.

### 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.

When using an SVG file in a symbol or a label, QGIS allows you 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
- load the file from a remote URL: as above, the image will only be loaded on successful retrieval of the remote resource
- embed the SVG file into the item: the file is embedded inside the current project, style database, or print layout template. The SVG file is then always rendered as part of the item. This is a convenient way to create self-contained projects with custom SVG symbols which can be easily shared amongst different users and installations of QGIS.

It is also possible to extract the embedded SVG file from a symbol or label and save it on disk.

**Poznámka:** The above mentioned options for loading and storing an SVG file in a project are also applicable to raster images you may want to use for customizing symbols, labels or decorations.

#### Cesty k zásuvným modulům

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.

**Poznámka:** Documentation is versioned and translated only for QGIS Long Term Releases (LTR), meaning that if you are running a regular release (e.g., QGIS 3.0), the help button will by default open the next LTR manual page (i.e. 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.

#### QSettings

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

#### Prostředí

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.
Obr. 9.1: System environment variables in QGIS
Use custom variables (restart required - include separators). You can Add and Remove variables. 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 CRS Settings

Poznámka: For more information on how QGIS handles layer projection, please read the dedicated section at Práce s projekcemi.

Obr. 9.2: CRS Settings in QGIS

**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**
- **Použít CRS projektu**
- **Use a default CRS**

**Výchozí transformace souřadnic**

In this group, 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.
- **Planimetric measurements:** sets the default for the „planimetric measurements” property for newly created projects.

### 9.1.4 Data Sources Settings

**Atributy prvku a tabulka**

- **Open new attribute tables as docked windows**
- **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**. 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.

- **Reprezentace pro NULL hodnoty.** Zde můžete definovat hodnotu datových polí, které obsahují NULL hodnotu.

**Tip: 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.
Obr. 9.3: Data Sources Settings in QGIS
Zacházení s datovými zdroji

• **Scan for valid items in the browser dock**. You can choose between 'Check extension' and 'Check file contents'.

• **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.

• **Výzva pro rastrové subvrstvy při otevírání**. Některé podpůrné rastrové sady — jsou v GDAL nazývány jako podřízené datové sady. Například v netCDF souborech — pokud obsahují mnho netCDF proměnných, GDAL vidí každou proměnnou jako podřízenou datovou sadu. 'Možnosti' umožňují kontrolovat, jak se vypořádat s podvrstvami při otevření souboru se subvrstvami. Míste následující možnosti:
  – 'Vždy': Vždy se dotázat (pokud existují nějaké subvrstvy)
  – 'Pokud je potřeba': Dotázat se, jestli nemá vrstva žádná pásma, ale má podvrstvy
  – 'Nikdy': Nikdy nevyzve a nenačte nic
  – 'Načíst vše': Nikdy nevyzve, ale načte všechny subvrstvy

• **Ignore shapefile encoding declaration**. If a shapefile has encoding information, this will be ignored by QGIS.

• **Execute expressions on server-side if possible**. When requesting features from a datasource, QGIS will try to optimize requests by sending filter criteria directly to the server and only download the features which match the criteria. For example, if for a list on the user interface only the farmers which live in Bern should be listed, QGIS will send a WHERE "hometown" = 'Bern' to the database. In some cases, filter criteria are too complex to be translated from QGIS Expressions to database compatible SQL. In those cases, QGIS will download the whole data and filter locally to be on the safe side, which is much less performant.

By disabling this option, QGIS can be forced to always download the whole data and filter locally, at the expense of a performance penalty. This option is meant as a safety break and should only be deactivated if you identify a misbehavior of the QGIS expression translation engine.

• **Evaluate default values** defines whether default values from database provider should be calculated when digitizing the new feature (checked status) or when saving the changes.

### Hidden Browser Path

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**.

#### 9.1.5 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.

• **Pokud to jde, používat vyrovnávací paměť pro zrychlení překreslování**

• **Vykreslovat vrstvy paralelně pomocí více jader CPU**

• **Maximální počet jader k dispozici**

• **Interval aktualizace mapy (výchozí hodnota 250 ms)**

• **Enable feature simplification by default for newly added layers**

• **Simplification threshold**

• **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
Obr. 9.4: Rendering tab of Project Properties dialog
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
- Magnification level (see the magnifier)

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

Kvalita vykreslování
- Linie bude vykreslena jako méně rozřešená, nicméně na úkor rychlosti vykreslování

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.

Rastry
- S Výběr RGB pásma můžete definovat hodnotu pro červené, zelené a modré pásmo.
- 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' and 'Cubic'. 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).

Vylepšení kontrastu
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'.

For rasters rendering, you can also define the following options:
- Limity ořezu dle kumulativního počtu pixelů
- Multiplikátor směrodatné odchylky

Ladění
- Map canvas refresh to debug rendering duration in the Log Messages panel.
9.1.6 Canvas and 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. You can either 'Open layer properties', 'Open attribute table' or 'Open layer styling dock' with the double click.
  - 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
- **the Delay** in milliseconds of layers map tips display

9.1.7 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.

Nástroj pro měření

- Definujte Barva pravítka pro měřicí nástroje
- Definujte Desetinná místa
- 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', '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 degrees' or 'Map Units'
Obr. 9.6: Map tools Settings in QGIS
• Preferred angle units: options are 'Degrees', 'Radians', 'Gon/gradians', 'Minutes of arc', 'Seconds of arc', 'Degrees', 'Radians', 'Gon/gradians', 'Minutes of arc', 'Seconds of arc', 'Turns/revolutions', milliradians (SI definition) or mil (NATO/military definition)

Posouvat a zvětšovat

• Define a Zoom factor for zoom tools or wheel mouse

Předdefinovaná měřítka

Here, you find a list of predefined scales. 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.

9.1.8 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 Default 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.
9.1.9 Digitizing Settings

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

Vytváření prvku

- **Suppress attribute form pop-up after feature creation**: this choice can be overridden in each layer properties dialog.

- **Reuse last entered attribute values**.

- **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.

Gumička

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

Uchycení

- Enable snapping by default activates snapping when a project is opened
- Define Default snap mode (Vertex’, ‘Vertex and segment’, ‘Segment’)
- Define Východní přichytávací tolerance v mapových jednotkách nebo píxelech
- Define Vyhledávací poloměr pro editaci lomových bodů v mapových jednotkách nebo píxelech
- 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)

Symboly lomových bodů

- Zobrazit symboly pouze u vybraných prvků
- Define vertex Marker style (‘Cross’ (default), ‘Semi transparent circle’ or ‘None’)
- Define vertex Marker size (in millimeter)

Nástroj odsazení křivky

The next 3 options refer to the Offset Curve tool in Advanced digitizing. 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’
- Segmenty kvadrantu
- Miter limit

9.1.10 Layouts Settings

Výchozí nastavení kompozice

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

Vzhled mřížky

- Define the Grid style (‘Solid’, ‘Dots’, ‘Crosses’)
- Definujte Barva mřížky

Výchozí mřížka a vodítka

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

Layout Paths

9.1. Možnosti
**9.1.11 GDAL Settings**

GDAL is a data exchange library for vector and raster files. It provides drivers to read and or write data in different formats. The GDAL tab currently exposes the drivers for raster formats with their capabilities.

**GDAL driver options**

This frame provides ways to customize the behavior of 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.
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 (e.g., 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**

![Image of Pyramids profile settings](image)

Obr. 9.11: Sample of Pyramids profile

**GDAL drivers**

In this frame, you can define which GDAL driver is to use to read and/or write files, as in some cases more than one GDAL driver is available.

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

### 9.1.12 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.
Obr. 9.12: GDAL Settings in QGIS

Obr. 9.13: Variables Settings in QGIS
9.1.13 Authentication Settings

In the Authentication tab you can set authentication configurations and manage PKI certificates. See Ověřovací systém for more details.

![Authentication Settings in QGIS](image)

9.1.14 Network Settings

**Obecné**
- Define Vyhledávací adresy WMS, výchozí je http://geopole.org/wms/search?search=%1&type=rss
- Define Časový limit pro síťové požadavky (ms) - výchozí je 60000
- Define Default expiration period for WMS Capabilities (hours) - default is 24
- Define Výchozí expirační perioda pro dlaždice WMS-C/WMTS (hodin) - výchozí je 24
- Define Max retry in case of tile or feature request errors
- Define User-Agent

**Nastavení vyrovnávací paměti**
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:
Obr. 9.15: Proxy-settings in QGIS
– **Default Proxy**: Proxy is determined based on system’s proxy.

– **Socks5Proxy**: Druhový proxy server pro jakýkoliv druh připojení. Podporuje TCP, UDP, vazbu na port (příchozí spojení) a ověřování.

– **HttpProxy**: Implementace pomocí příkazu „CONNECT“, podporuje pouze odchozí TCP spojení; podporuje ověřování.

– **HttpCachingProxy**: Implementace pomocí běžných příkazů HTTP, je užitečná pouze v kontextu HTTP požadavků.

– **FtpCachingProxy**: Implementace pomocí FTP proxy, je užitečná pouze v souvislosti s požadavky FTP.

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

Excluding some URLs can be added to the text box below the proxy settings (see Figure_Network_Tab). 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/qt-5.9/qnetworkproxy.html#ProxyType-enum](https://doc.qt.io/qt-5.9/qnetworkproxy.html#ProxyType-enum)

**Tip: Použití proxy**

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.

### 9.1.15 Locator Settings

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

- **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.
- **Features in all layers (af)**: searches for matching attributes in the **display name** of each **searchable layers** and zooms to the selected feature.
- **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.
- **Processing (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.

For each filter, you can customize the filter, set whether it is enabled by default or not. The set of default locator filters can be extended by plugins, eg for OSM nominatim searches, direct database searching, layer catalog searches.

The locator search bar can be activated pressing Ctrl+K. Type your text to perform a search. By default, results are returned for all enabled locator filters but you can limit the search to a certain filter by prefixing your text with the locator filter prefix, ie. typing l cad will return only the layers whose name contains cad. Click on the result to execute the corresponding action, depending on the type of item.

Searching is handled using threads, so that results always become available as quickly as possible, regardless of whether any slow search filters may be installed. They also appear as soon as each result is encountered by each filter, which
means that e.g. a file search filter will show results one by one as the file tree is scanned. This ensures that the UI is always responsive even if a very slow search filter is present (e.g. one which uses an online service).

**Tip: Quick access to the locator’s configurations**

Click on the icon inside the locator widget on the status bar to display the list of filters you can use and a *Configure* entry that opens the *Locator* tab of the *Settings → Options…* menu.

![Locator Settings in QGIS](image)

**Obr. 9.16: Locator Settings in QGIS**

### 9.1.16 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.

**Varování: 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.
9.1.17 Acceleration Settings

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

9.1.18 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 QGIS processing framework.

9.2 Working with User Profiles

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
- 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:
Obr. 9.18: Acceleration tab

Obr. 9.19: Processing Settings tab in QGIS
• ~ represents the HOME directory, which on Windows is usually something like C: \Users\(user). 

• 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.

Unless changed, the profile of the last closed QGIS session will be used in the following QGIS sessions.

Tip: 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.

9.3 Vlastnosti projektu

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 in the browser). 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.

• Uveďte název projektu spolu s cestou k souboru projektu

• Zvolte barvu, kterou chcete použít pro vrstvy, když jsou vybrány.

• Zvolte barvu pozadí: barvu, kterou chcete použít pro mapový podklad.

• Nastavte, zda má být cesta k vrstvám v projektu uložena jako absolutní (úplná) nebo jako relativní k umístění souboru projektu. Můžete upřednostnit relativní cestu, pokud lze přesunout nebo sdílet vrstvy i soubory projektu, nebo pokud je projekt přístupný z počítačů na různých platformách.

• Můžete zvolit vyhýbání se artefaktům, je-li projekt vykreslen jako mapová dlaždice. Všimněte si, že změna této možnosti může vést k degradaci provedení.

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.

- 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 the units set in QGIS options but then overrides it for the current project, are used in:
  - Panel pro aktualizaci pole atributové tabulky
  - Kalkulace kalkulátoru pole
  - Určení nástroje odvození délky, obvodu a plochy
  - Výchozí jednotky zobrazované v dialogovém okně měření

The Coordinate display allows you to choose and customize the format of units to use to display the mouse coordinate in the status bar and the derived coordinates shown via the identify tool.

Finally, you can set a **Project predefined scales** list, which overrides the global predefined scales.

---

Obr. 9.20: General tab of the Project Properties dialog
9.3.2 Metadata Properties

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 CRS Properties

**Poznámka:** For more information on how QGIS handles project projection, please read the dedicated section at Práce s projekcemi.

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

- **No projection (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.

The CRS tab also 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.

9.3.4 Default Styles Properties

The Default Styles tab 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 default symbols (Marker, Line, Fill) to apply depending on the layer geometry type as well as a default Color Ramp
- Apply a default Opacity to new layers
- Assign random colors to symbols, modifying the symbols fill colors, hence avoiding same rendering for all layers.

Using the Style Manager button, you can also quickly access the Style Manager dialog and configure symbols and color ramps.

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.

**Tip:** Use project colors to quickly assign and update color widgets

Project colors can be referred 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.

### 9.3.5 Data Sources Properties

In the *Data Sources* tab, you can:

- **Automatically create transaction groups where possible**: When this mode is turned on, all (postgres) layers from the same database are synchronised in their edit state, i.e. when one layer is put into edit state, all are, when one layer is committed or one layer is rolled back, so are the others. Also, instead of buffering edit changes locally, they are directly sent to a transaction in the database which gets committed when the user clicks save layer. Note that you can (de)activate 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).

- **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.

- 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.

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.

![Data Sources tab](image)

Obr. 9.22: Data Sources tab

### 9.3.6 Relations Properties

The Relations tab is used to define 1:n relations. The relations 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 1:n relations support in Section Creating one or many to many relations.
9.3.7 Variables Properties

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.

9.3.8 Macros Properties

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

9.3.9 QGIS Server Properties

The tab QGIS Server 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 Configure your project and subsequent.

9.4 Přizpůsobení

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.

Poznámka: Before your changes are applied, you need to restart QGIS.

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. The configurable item can be:

- a Menu or some of its sub-menus from the Menu lišta
- a whole Panel (see Panely a nástrojové lišty)
- the Status bar described in Stavová lišta or some of its items
- a Toolbar: the whole bar or some of its icons
Obr. 9.24: Makro nastavení v QGIS

```python
1 - def openProject():
2     msgbar = iface.messageBar()
3 -     msgbar.pushMessage('WARNING', 'project contains sensitive data',
4             QgisMessageBar.WARNING).
5
6 - def saveProject():
7     pass
8
9 - def closeProject():
10    pass
11```

9.4. Přizpůsobení
Obr. 9.25: QGIS Server settings tab

- or any widget from any dialog in QGIS: label, button, combobox…

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 the Save To File button. This is a handy way to share a common QGIS interface among multiple users. Just click on the 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.

**Tip:** Easily restore predefined QGIS

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 QSettings 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.
Obr. 9.26: The Customization dialog
9.5 Keyboard shortcuts

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

![Keyboard Shortcuts dialog](image)

![Obr. 9.27: Define shortcut options](image)

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
- *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 as an *.XML* file 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:
- [--version]  display version information and exit
- [--snapshot filename]  emit snapshot of loaded datasets to given file
- [--width width]  width of snapshot to emit
- [--height height]  height of snapshot to emit
- [--lang language]  use language for interface text (changes existing...)
- override

- [--project projectfile]  load the given QGIS project
- [--extent xmin,ymin,xmax,ymax]  set initial map extent
- [--nologo]  hide splash screen
- [--noplugins]  don’t restore plugins on startup
- [--nocustomization]  don’t apply GUI customization
- [--customizationfile path]  use the given ini file as GUI customization
- [--globalseettingsfile path]  use the given ini file as Global Settings...
- (defaults)

- [--authdbdirectory path]  use the given directory for authentication...
- database

- [-code path]  run the given python file on load
- [-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.

- [-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.

- [-]  this text

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

(continues on next page)
Tip: 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.

--noverisoncheck

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.

--customizationfile

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

--globalsettingsfile

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. The default global settings is located in `your_QGIS_PKG_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.

The equivalent environment variable is `QGIS_GLOBAL_SETTINGS_FILE`.

9.6. Running QGIS with advanced settings
--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:

```python
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:

```bash
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 QgsAppScreen Shots::Categories).

--profile

Loads QGIS using a specific profile from the user’s profile folder. Unless changed, the selected profile will be used in the following QGIS sessions.

--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_PKG_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"
```

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. A lot of settings can be set using the INI file such as WMS/WMTS, PostGIS connections, proxy settings, maptips…
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.
10 Práce s projekcemi

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 Přehled projekční podpory

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 Vlastní souřadnicový referenční systém 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 OGR or 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
Obr. 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, deferring 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.

- **Použít CRS projektu**

- **Use default layer CRS**, as set in the **Default CRS for layers** combobox above.

**Tip:** 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 figure_projection_options). 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.

Available options are:
Obr. 10.2: Dialog vlastností projektu
• No projection (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 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.

**Tip: 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

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.

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 Vlastní souřadnicový referenční systém

Pokud QGIS neposkytuje souřadnicový referenční systém, který potřebujete, můžete definovat vlastní CRS. Chcete-li definovat CRS, zvolte "vlastní CRS..." z menu Nastavení. Vlastní uživatelské systémy jsou uloženy ve vaší vlastní uživatelské QGIS databázi. Kromě vašich CRS obsahuje tato databáze vaše prostorové záložky a další vlastní data.


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.

Dialog Vlastní definice souřadnicového referenčního systému vyžaduje pouze dva parametry pro definování uživatelského SRS:

1. Popisný název
2. The cartographic parameters in PROJ or WKT format

To create a new CRS, click the Add new CRS button, enter a descriptive name, select the format, and the CRS parameters.

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

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=namenofthefile.gsb) to the Proj definition in the Parameters field of the Custom Coordinate Reference System Definition (Settings Custom Projections...).

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 CRS tab menu under the Default datum transformations group.
Obr. 10.3: Dialog vlastní CRS

Obr. 10.4: Setting an NTv2 transformation

10.6. Datum Transformations
• using **Ask for datum transformation if several are available:** when more than one appropriate datum transformation exists 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 of the transformation.

   In some cases a transformation may not be available for use on your system. In this case, the transformation will still be shown in this list but will not be selectable.
4. Find your preferred transformation, select it and click **OK**.

A row is added to the table under **Default Datum Transformations** with information about **Source CRS** and **Destination CRS** as well as the datum transformations (**Source Datum Transformation** and **Destination Datum Transformation**) to be used to transform between the CRSs.

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.

![Select Datum Transformations dialog](image)

**Obr. 10.5: Selecting a preferred default datum transformation**

Datum transformations set in the **Settings > Options > CRS** 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.
11.1 Kontextová nápověda

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.

11.2 Panely

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 Panely.

11.2.1 Layers Panel

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

Poznámka: The Z-ordering behavior can be overridden by the Layer Order panel.

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
- Manage Map Themes: control visibility of layers and arrange them in different map themes.
- 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.
• **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.

Obr. 11.1: Layer Toolbar in Layers Panel

---

**Poznámka:** 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:

- **Zobrazit všechny vrstvy**
- **Skrýt všechny vrstvy**
- **Show Selected Layers**
- **Hide Selected Layers**
- **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, i.e., the layer checked node items in the **Layers panel**. This applies to **symboologies** 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**

**Poznámka:** 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 (e.g., 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. Use the Remove Current Theme button to delete the active theme.

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 vector or raster layers added to the project, optionally organized in groups. Depending on the item selected in the panel, a right-click shows a dedicated set of options presented below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Vektorová vrstva</th>
<th>Rastrová vrstva</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom to Layer/Group</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Zoom to Selection</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Show in Overview</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Show Feature Count</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Copy Layer/Group</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Rename Layer/Group</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Zoom to Native Resolution (100%)</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Stretch Using Current Extent</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Update SQL Layer…</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Add Group</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Duplicate Layer</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Remove Layer/Group…</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Move Out of Group</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Move to Top</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Check and all its Parents</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Group Selected</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Otevřit atributovou tabulku</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Přepnut editaci</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Aktuální změny</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Filtrovat…</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Set Layer Scale Visibility…</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>
### Tabulka 11.1 — pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Option</th>
<th>Vektorová vrstva</th>
<th>Rastrová vrstva</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom to Visible Scale</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Set CRS 🔍</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Set Layer/Group CRS…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Set Project CRS from Layer</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Set Group WMS Data…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Mutually Exclusive Group</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Check and all its children (Ctrl-click)</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Uncheck and all its children (Ctrl-click)</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Make Permanent</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Export 🔍</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Save As…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Save Features As…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Save Selected Features As…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Save As Layer Definition File…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Save As QGIS Layer Style File…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Styles 🔍</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Copy Style</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Paste Style</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Add…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Rename Current…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Edit symbol…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Copy Symbol</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Paste Symbol</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Vlastnosti…</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
</tbody>
</table>

Table: Context menu from Layers Panel items

For GRASS vector layers, Toggle editing is not available. See section Digitizing and editing a GRASS vector layer for information on editing GRASS vector layers.

### Interact with groups and layers

Layers in the legend window can be organized into groups. There are two 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 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 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, icons appear next to the layer or group in the Layers panel to give more information about the layer/group. These symbols 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 setting
- 🔍 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. Click the icon to update the source path.
- ☑️ 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 OGR vector formats supported by QGIS.
- 🚨 to identify a layer that has no/unknown CRS

Editing vector layer style

From the Layers panel, you have shortcuts to change the layer rendering quickly and easily. Right-click on a vector layer and select Styles in the list in order to:

- see the styles currently applied to 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

Tip: 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) as the original layer and, for vector layers, have the same geometry type (point, line or polygon).

- rename the current style, add a new style (which is actually a copy of the current one) or delete the current style (when multiple styles are available).

Poznámka: The previous options are also available for raster or mesh layers.
• 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…: open the Symbol Selector dialog and change feature symbol (symbol, size, color…).

When using a classification symbology type (based on categorized, graduated or rule-based), the aforementioned symbol-level options are available from the class entry context menu. Also provided are the Toggle Items, Show All Items and Hide All Items entries to switch the visibility of all the classes of features. These avoid (un)checking items one by one.

Tip: Double-clicking a class leaf entry also opens the Symbol Selector dialog.

11.2.2 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:

• Set raster layer Symbology, Transparency, and Histogram properties. These options are the same as those in the Raster Properties Dialog.

• Set vector layer Symbology, 3D View and Labels properties. These options are the same as those in the The Vector Properties Dialog.

• 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.

Tip: Add custom tabs to the Layer Styling panel

Using PyQGIS, you can set new tabs to manage layer properties in the Layer Styling Panel. See https://nathanw.net/2016/06/29/qgis-style-dock-part-2-plugin-panels/ for an example.

11.2.3 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 figure_layer_order, you can see that the airports features are displayed over the alaska polygon despite those layers’ respective placement in the Layers panel.
Obr. 11.2: Defining a layer's symbology from the layer styling panel
Unchecking Control rendering order will revert to default behavior.

Obr. 11.3: Define a layer order independent of the legend

11.2.4 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.

11.2.5 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.

11.2.6 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.
11.2.7 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
- the column to use, or an expression
- 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:

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Řetězec</th>
<th>Integer</th>
<th>Float</th>
<th>Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Count Distinct Value</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Count Missing value</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Sum</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Mean</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Standard Deviation on Sample</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Minimal value</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Maximal value</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Rozsah</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Minority</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Majority</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Variety</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>First Quartile</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Third Quartile</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Inter Quartile Range</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Minimum Length</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Maximum Length</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Table: Statistics available for each field type

The statistical summary can be:

- returned for the whole layer or selected features only
- recalculated using the button when the underlying data source changes (eg. new or removed features/fields, attribute modification)
- copied to the clipboard and pasted as a table in another application
Obr. 11.4: Show statistics on a field
11.3 Vnořování projektů

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 figure_embed_dialog)
2. Hold down Ctrl (or Cmd) and click on the layers and groups you wish to retrieve
3. Click 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.

Like any other layer, an embedded layer can be removed from the project by right-clicking on the layer and clicking Remove.

Tip: 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.

11.4 Working with the map canvas

11.4.1 Vykreslování

By default, QGIS renders all visible layers whenever the map canvas is refreshed. The events that trigger a refresh of the map canvas include:

- adding a layer
- panning or zooming
QGIS 3.10 User Guide

• resizing the QGIS window
• changing the visibility of a layer or layers

QGIS vám umožňuje kontrolovat proces vykreslování mnohými cestami.

Vykreslování v závislosti na měřítku

Scale-dependent rendering allows you to specify the minimum and maximum scales at which a layer (raster or vector) will be visible. To set scale-dependent rendering, open the Properties dialog by double-clicking on the layer in the legend. On the Rendering tab, tick Scale dependent visibility and enter the Minimum (exclusive) and Maximum (inclusive) scale values.

You can also activate scale dependent visibility on a layer from the Layers panel. Right-click on the layer and in the context menu, select Set Layer Scale Visibility.

The button allow you to use the current map canvas scale as boundary of the range visibility.

Poznámka: When a layer is not rendered in the map canvas because the map scale is out of its visibility scale range, the layer is greyed in the Layers panel and a new option Zoom to Visible Scale appears in the layer context menu. Select it and the map is zoomed to the layer’s nearest visibility scale.

Kontrolování vykreslení mapy

Vykreslování mapy může být ovládáno různými způsoby, které jsou popsány níže.

Pozastavení vykreslování

To suspend rendering, click the Render checkbox in the bottom-right corner of the status bar. When Render is not checked, QGIS does not redraw the canvas in response to any of the events described in the section Vykreslování. 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

Zaškrtnutí Render checkboxu umožní vykreslení a způsobí okamžité obnovování mapového okna.

Nastavení možnosti přidání vrstvy

You can set an option to always load new layers without drawing them. This means the layer will be added to the map, but its visibility checkbox in the legend will be unchecked by default. To set this option, choose menu option Settings Options and click on the Rendering tab. Uncheck By default new layers added to the map should be displayed. Any layer subsequently added to the map will be off (invisible) by default.
Zastavení vykreslování

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 take a bit of time between pressing Esc for the map drawing to halt.

Ovlivnění kvality vykreslení

QGIS has an option to influence the rendering quality of the map. Choose menu option Settings Options, click on the Rendering tab and select or deselect Make lines appear less jagged at the expense of some drawing performance.

Zrychlení vykreslování

There are some settings that allow you to improve rendering speed. Open the QGIS options dialog using Settings Options, go to the Rendering tab and select or deselect the following checkboxes:

- Use render caching where possible to speed up redraws.
- Render layers in parallel using many CPU cores and then set the Max cores to use.
- The map renders in the background onto a separate image and each Map Update interval, 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.
- With 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.

11.4.2 Zooming and Panning

QGIS provides tools to zoom and pan to your area of interest.

Apart from using the pan and zoom-in/zoom-out icons on the toolbar with the mouse, you can also navigate with the mouse wheel, spacebar and arrow keys. A Zoom factor can be set under the Settings Options Map tools menu to define the scale behavior while zooming.

With the mouse wheel

You can press the mouse wheel to pan inside of the main window (on macOS, you may need to hold down the cmd key). You can roll the mouse wheel to zoom in and out on the map; the mouse cursor position will be the center of the zoomed area of interest. Holding down Ctrl while rolling the mouse wheel results in a finer zoom.

With the arrow keys

Panning the map is possible with the arrow keys. Place the mouse cursor inside the map area, and click on the arrow keys to pan up, down, left and right.

You can also use the space bar to cause mouse movements temporarily to pan the map. The PgUp and PgDown keys on your keyboard will cause the map display to zoom in or out following the zoom factor set. Pressing Ctrl++ or Ctrl+- also performs an immediate zoom in/out on the map canvas.

When certain map tools are active (Identify, Measure...), you can perform a zoom by holding down Shift and dragging a rectangle on the map to zoom to that area. This is not enabled for selection tools (since they use Shift for adding to selection) or edit tools.
11.4.3 Prostorové záložky

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.

Tvůření záložek

Abyste vytvořili záložku:

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.

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; the extent can be calculated from a loaded layer extent, the current map canvas or drawn over the current map canvas.

6. Indicate the CRS to use for the extent

7. Select whether the bookmark will be Saved in User Bookmarks or Project Bookmarks

8. Press Save to add the bookmark to the list

Můžete mít více záložek se stejným názvem.
Práce se záložkami

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:

<table>
<thead>
<tr>
<th>Task</th>
<th>Spatial Bookmark Manager</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Zoom to a Bookmark</strong></td>
<td>Double-click on it, or select the bookmark and press the Zoom to bookmark button.</td>
<td>Double-click on it, drag and drop it to the map canvas, or right-click the bookmark and select Zoom to Bookmark.</td>
</tr>
<tr>
<td><strong>Delete a bookmark</strong></td>
<td>Select the bookmark and click the Delete bookmark button. Confirm your choice.</td>
<td>Right-click the bookmark and select Delete Spatial Bookmark. Confirm your choice.</td>
</tr>
<tr>
<td><strong>Export bookmarks to XML</strong></td>
<td>Click the Import/Export Bookmarks button and select Export. All the bookmarks (user or project) are saved in an xml file.</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Import bookmarks from XML</strong></td>
<td>Click the Import/Export Bookmarks button and select Import. All bookmarks in the XML file are imported as user bookmarks.</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Edit bookmark</strong></td>
<td>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.</td>
<td>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).</td>
</tr>
</tbody>
</table>

You can also zoom to bookmarks by typing the bookmark name in the locator.

11.4.4 Dekorace

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.
**Mřížka**

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.

![Grid Properties Dialog](image)

Obr. 11.7: Dialog mřížky

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 approximately 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 using the OS 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.

   ![Obr. 11.8: The Title Decoration Dialog](image)

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 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.

![Copyright Label Decoration]

Obr. 11.9: 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 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.

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.
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.

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.
Měřítko

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.

QGIS only supports displaying the scale in the same units as your map frame. So, if the units of your project’s CRS are meters, you can’t create a scale bar in feet. Likewise, if you are using decimal degrees, you can’t create a scale bar to display distance in meters.

Přidání měřítka

1. Select menu option View ➤ Decorations ➤ Scale Bar… to open the dialog

![Scale Bar Decoration dialog](image)

Obr. 11.12: Dialog měřítka

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

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.

Obr. 11.13: 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 \Decoration Layout Extents to open the 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.

Tip: 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.
11.4.5 Anotační nástroje

Annotations are information added to the map canvas and shown within a balloon. This information can be of different types and annotations are added using the corresponding tools in the Attributes 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 figure_custom_annotation). 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.
To add an 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.

Annotations can be selected when an annotation tool is enabled. They can then be moved by map position (by dragging the map marker) or by moving only the balloon. The Move Annotation tool also allows you to move the balloon on the map canvas.
To 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.

**Poznámka:** If you press Ctrl+T while an Annotation tool (move annotation, text annotation, form annotation) is active, the visibility states of the items are inverted.

---

**Tip: 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

---

**11.4.6 Měření**

**General information**

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 and angles interactively**

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

**Poznámka:** **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 *Nastavení přichytávací tolerance a vyhledání poloměru*). 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.

By default, **Measure Line** measures real distances between given points according to a defined ellipsoid. 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. 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 (‘Meters’, ‘Kilometers’, ‘Feet’, ‘Yards’, ‘Miles’, ‘Nautical miles’, ‘Centimeters’, ‘Millimeters’, ‘Degrees’, ‘Map units’). 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.

---

**Obr. 11.17: Measure Distance**

**Obr. 11.18: Measure Area**

**Obr. 11.19: 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.

---

11.4. Working with the map canvas
11.5 Interacting with features

11.5.1 Selecting features

QGIS provides several tools to select features on the map canvas. Selection tools are available in the Edit ➤ Select menu or in the Attributes toolbar.

**Poznámka:** 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
- Select Features by Polygon
- Select Features by Freehand
- Select Features by Radius

**Poznámka:** Other than Select Features by Polygon, these manual selection tools allow you to select feature(s) on the map canvas with a single click.

**Poznámka:** Use the Select Features by Polygon tool to use an existing polygon to select overlapping features. 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.

**Tip:** 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.

While using the Select Feature(s) tool, holding Shift or Ctrl toggles whether a feature is selected (ie either adds to the current selection or remove from it).

For the other tools, different behaviors can be performed by holding down:

- Shift: add features to the current selection
- Ctrl: subtract 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 subtract 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...](image1) select features using expression dialog
- ![Select Features By Value...](image2) or press **F3**
- ![Deselect Features from All Layers](image3) or press **Ctrl+Shift+A** to deselect all selected features in all layers
- ![Select All Features](image4) or press **Ctrl+A** to select all features in the current layer
- ![Invert Feature Selection](image5) to invert the selection in the current layer

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](image6) 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 *Výrazy* for more information and examples.

**Tip: 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.

Alongside each field, there is a drop-down list with options to control the search behaviour:
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.

### 11.5.2 Identifying Features

The Identify tool allows you to interact with the map canvas and get information on features in a pop-up window. To identify features, use:

- **View Identify Features**
- **Ctrl+Shift+I** (or **Cmd+Shift+I**),
- **Identify Features** icon on the Attributes toolbar

#### Using the Identify Features tool

QGIS offers several ways to identify features with the **Identify Features** tool:

- **left click** identifies features according to the **selection mode** and the **selection mask** set in the **Identify Results** panel
- **right click** with **Identify Feature(s)** as **selection mode** set in the **Identify Results** panel fetches all snapped features from all visible layers. This opens a context menu, allowing the user to choose more precisely the features to identify or the action to execute on them.
- **right click** with **Identify Features by Polygon** as **selection mode** in the **Identify Results** panel identifies the features that overlap with the chosen existing polygon, according to the **selection mask** set in the **Identify Results** panel

**Tip:** Filter the layers to query with the **Identify Features** tool

Under **Layer Capabilities** in Project Properties… Data Sources, uncheck the **Identifiable** column next to a layer to avoid it being queried when using the **Identify Features** tool in a mode other than **Current Layer**. This is a handy way to return features from only layers that are of interest for you.

If you click on feature(s), the **Identify Results** dialog will list information about the feature(s) clicked. 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.

#### Feature information

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 Actions Properties).
- **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

• 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.

• **Data attributes**: This is the list of attribute fields and values for the feature that has been clicked.

**Poznámka**: Links in the feature’s attributes are clickable from the Identify Results panel and will open in your default web browser.

Obr. 11.21: Identify Results dialog
The Identify Results dialog

At the top of the window, you have a handful of tools:

- **Open Form** of the current feature
- **Expand tree**
- **Collapse tree**
- **Expand New Results by Default** to define whether the next identified feature's information should be collapsed or expanded
- **Clear Results**
- **Copy selected feature to clipboard**
- **Print selected HTML response**

- **selection mode to use to fetch features to identify:**
  - **Identify Features by area or single click**
  - **Identify Features by Polygon**
  - **Identify Features by Freehand**
  - **Identify Features by Radius**

**Poznámka:** When using **Identify Features by Polygon**, you can right-click any existing polygon and use it to identify overlapping features in another layer.

At the bottom of the window are the **Mode** and **View** combo boxes. **Mode** defines from which layers features should be identified:

- **Current layer**: only features from the selected layer are identified. The layer need not be visible in the canvas.
- **Top down, stop at first**: only features from the upper visible layer.
- **Top down**: all features from the visible layers. The results are shown in the panel.
- **Layer selection**: opens a context menu where the user selects the layer to identify features from, similar to a right-click. Only the chosen features will be shown in the result panel.

The **View** can be set as **Tree**, **Table** or **Graph**. 'Table' and 'Graph' views can only be set for raster layers.

The identify tool allows you to **Auto open form for single feature results**, found under **Identify Settings**. 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.

Další funkce lze nalézt v kontextovém menu identifikovaného prvku. Například, z kontextového menu můžete:

- Zobrazit tvar prvku
- Přiblížit na prvek
- Kopírovat prvek: zkopirovat všechnu geometrii a atributy prvku
- Toggle feature selection: Add identified feature to selection
- Kopírovat hodnotu atributu: zkopirovat pouze hodnotu atributu, na který kliknete
- Copy feature attributes: Copy the attributes of the feature
- Vymazat výsledek: odstranit výsledky v okně

11.5. Interacting with features 141
• Vymazat zvýrazněné: odstranit z mapy zvýraznění prvků
• Zvýraznit všechny
• Zvýraznit vrstvu
• Aktivovat vrstvu: zvolit vrstvu, aby byla aktivní
• Vlastnosti vrstvy: otevřít okno s vlastnostmi vrstvy
• Rozbalit vše
• Sbalit vše

11.6 Save and Share Layer Properties

11.6.1 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.

Obr. 11.22: 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.

**Poznámka**: 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.

**Tip**: Manage styles from layer context menu

Right-click on the layer in the Layers panel to copy, paste, add or rename layer styles.

### 11.6.2 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, MSSQL and Oracle. The layer style is saved inside a table (named `layer_styles`) in the database. Click on Style ➤ 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 the QGIS local database, a SQLite database in the `/~/.qgis2/` directory (where QGIS stores its local settings).

**Tip**: 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 MSSQL for instance). Use instead a plain text file if you want the style to be shared among databases.

**Poznámka:** 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, the `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.

**Tip:** 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).

### 11.6.3 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.
11.7 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, 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 ¥]
```

Will render the label like this:

```
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 Vlastnosti projektu)
- **vector layer variables** from the Layer Properties dialog (see The Vector Properties Dialog);
- **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.

**Poznámka:** 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.

11.8 Authentication

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 Ověřovací systém.

A master password needs to be set up when initializing the authentication system and its portable database.

11.9 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.
Obr. 11.24: Variables editor at the project level
11.9.1 Výběr barvy

The color dialog

The Select Color dialog will appear whenever you click the \icon{select-color} 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 \icon{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 \icon{color-wheel} or \icon{color-wheel}. With the first two tabs, you can browse to all possible color combinations and apply your choice to the item.

![Select Color Dialog](qgis3.10-user-guide/images/select-color.png)

Obr. 11.25: Tabulka rozsahu výběru barvy

In the \icon{color-swatches} tab, you can choose from a list of color palettes (see \icon{colors-settings} for details). All but the Recent colors palette can be modified with the \icon{add-current-color} and \icon{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 \icon{figure-color-selector})

Another option is to use the \icon{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.

**Tip: 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
**Poznámka:** 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.

![Quick color selector menu](image)

Obr. 11.27: Quick color selector menu

**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.

![Colorbrewer ramp](image)

Obr. 11.28: 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*
- 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

![Image of Color Ramps](image_url)

Obr. 11.29: Quick color ramp selection widget

### 11.9.2 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

### 11.9.3 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 the following options:

![Image of Font Selector](image_url)

Obr. 11.30: Font selector drop-down menu

• **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 a dialog to set text format parameters. Depending on the context, it can be the OS default **Text format** dialog or the QGIS custom dialog with advanced formatting options (opacity, orientation, buffer, background, shadow, …) as described in section **Formatting the label text**.

• **Copy Format** of the text

• and **Paste Format** to the text, speeding configuration.

### 11.9.4 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**

- **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.

- 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.

![Obr. 11.31: Adjust scaling range dialog](image-url)
11.9.5 Režimíchání

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.

11.9.6 Data defined override setup

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 \( \text{Data defined override} \) 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)

**Tip:** Use right-click to (de)activate the data override

When the data-defined override option is set up correctly the icon is yellow \( \text{data defined override} \) or \( \text{data defined override} \). If it is broken, the icon is red \( \text{data defined override} \) or \( \text{data defined override} \).

You can enable or disable a configured \( \text{data defined override} \) button by simply clicking the widget with the right mouse button.

Using the data-defined assistant interface

When the \( \text{Data defined override} \) button is associated with a numeric or color parameter, it has an **Assistant...** option that allows you to change how the data is applied to the parameter for each feature. The assistant allows you to:

- Define the **Input** data, ie:
  - the attribute to represent, using the Field listbox or the \( \text{Set column expression} \) function (see **Výrazy**).
  - the range of values to represent: you can manually enter the values or use the \( \text{Fetch value range from layer} \) button to fill these fields automatically with the minimum and maximum values returned by the chosen attribute or the 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 breakpoint(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:
  - the minimum and maximum values to apply to the selected property (in case of a color setting, you’ll need to provide a **color ramp**)
  - the **Scale method** of representation which can be **Flannery**, **Exponential**, **Surface** or **Radius**
  - the **Exponent** to use for data scaling
  - the output value or **color** to represent features with NULL values
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.

![Image of the varying size assistant dialog]

Obr. 11.32: The data-defined size assistant

The values presented in the varying size assistant above will set the size 'Data-defined override' with:

```
coalesce(scale_exp(Importance, 1, 20, 2, 10, 0.57), 1)
```
12.1 The Style Manager

12.1.1 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).

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);

• abc Label settings to manage label settings, which include the text formats and some layer-type specific settings such as label placement, priority, callouts, rendering…

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. figure_smart_group). 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.

Obr. 12.2: 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.

![Export Item(s) dialog](image)

Obr. 12.3: 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.

![Image of Import Item(s) dialog]

Obr. 12.4: 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.

![Image of Style Manager dialog]

Obr. 12.5: Opening a style items file

The dialog also allows to export single symbols as .PNG or .SVG files.
12.1.2 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, generate 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.

  ![Gradient color ramp](image)

  **Obr. 12.6: Example of custom gradient color ramp with multiple stops**

- **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’.

Obr. 12.7: cpt-city dialog with hundreds of color ramps

Tip: Easily adjust the color stops of the gradient color ramp

Double-clicking the ramp preview or drag-and-drop a color from the color spot onto the ramp preview adds a new color stop. Each color stop can be tweaked using the Výběr barvy widgets or by plotting each of its parameters. You can also reposition it using the mouse, the arrow keys (combine with Shift key for a larger move) or the Relative position spinbox. Pressing Delete stop as well as DEL key removes the selected color stop.

12.2 The Symbol Selector

The Symbol selector is the main dialog to design a symbol. You can create or edit Marker, Line or Fill Symbols. 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.
Obr. 12.8: Designing a Line symbol
12.2.1 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

12.2.2 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

Tip: 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.

Poznámka: 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

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.

Common parameters

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 Data defined override setup for more information)

• the Enable 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.

Poznámka: 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)

  ![Image of Simple Marker Symbol](image)

  Obr. 12.9: Designing a Simple Marker Symbol

- **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**: use installed fonts as marker symbols

- **Geometry generator** (see *The Geometry Generator*)

- **Raster image marker**: use an image (PNG, JPG, BMP ...) as marker symbol. The image can be a file on the disk, a remote URL or embedded in the style database (*more details*). Width and height of the image can be set independently or using the *lock aspect ratio*.

- **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 or embedded in the style database (more details).

**Poznámka:** SVG version requirements

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.

**Tip:** Enable SVG marker symbol customization

To have the possibility to change the colors of a SVG marker, you have to add the placeholders `param(fill)` for fill color, `param(outline)` for stroke color and `param(outline-width)` for stroke width. These placeholders can optionally be followed by a default value, e.g.:

```xml
<svg width="100%" height="100%">
<rect fill="param(fill) #ff0000" stroke="param(outline) #00ff00" stroke-width="param(outline-width) 10" width="100" height="100">
</svg>
```

**Line Symbols**

Appropriate for line geometry features, line symbols have following symbol layer types:

• **Simple line** (default): available settings are:

  The simple line symbol layer type has many of the same properties as the simple marker symbol, and in addition:

  – **Cap style**

  – **Use custom dash pattern**: overrides the Stroke style setting with a custom dash.

• **Arrow**: draws lines as curved (or not) arrows with a single or a double head with configurable width, length and thickness. To create a curved arrow the line feature must have at least three vertices. 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 The Geometry Generator)

• **Marker line**: repeats a marker symbol over the length of a line.

  – The markers placement can be at a regular distance or based on the line geometry: first, last or each vertex, on the central point of the line or of each segment, or on every curve point.

  – The markers placement can also be given an offset along the line

  – The **Rotate marker** option allows you to set 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.
Obr. 12.10: Designing a Simple Line Symbol
The marker line can also be offset from the line itself.

- **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**

![Obr. 12.11: Examples of hashed lines](image)

**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
- **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.

  The marker(s) can be placed on every part of a multi-part feature or only on its biggest part, and forced to be inside the polygon.

- **Geometry generator** (see *The Geometry Generator*).

- **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);

- **Line pattern fill**: fills the polygon with a hatching pattern of line symbol layer. You can set a rotation, the spacing between lines and an offset from the feature boundary;

- **Point pattern fill**: fills the polygon with a hatching pattern of marker symbol layer. You can set the distance and a displacement between rows of markers, and an offset from the feature boundary;

- **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.

- **SVG fill**: fills the polygon using SVG markers;
Obr. 12.12: Designing a Simple Fill Symbol
• **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;

• **Outline: Hashed line**: uses a *hash line symbol* layer to represent the polygon boundary (the interior rings, the exterior ring or all the rings).

• **Outline: Marker line**: uses a marker line symbol layer to represent the polygon boundary (the interior rings, the exterior ring or all the rings).

• **Outline: simple line**: uses a simple line symbol layer to represent the polygon boundary (the interior rings, the exterior ring 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.

---

**Poznámka:** When geometry type is polygon, you can choose to disable the automatic clipping of lines/polys 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).

---

### The Geometry Generator

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.

Some examples:

```sql
-- render 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 )

-- Given polygon layer1( id1, layer2_id, ...) and layer2( id2, fieldn...)
-- render layer1 with a line joining centroids of both where layer2_id = id2
make_line( centroid( $geometry ),
    centroid( geometry( get_feature( 'layer2', 'id2', attribute($currentfeature, 'layer2_id') ) ) )
)

-- Create a nice radial effect of points surrounding the central feature
-- point when used as a MultiPoint geometry generator
collect_geometries(
    array_foreach(
        generate_series( 0, 330, 30 ),
        project( $geometry, .2, radians( @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 \((x\text{ and } y\text{ components of the field})\)
- **or polar** coordinates: in this case, attributes define \(\text{Length}\) and \(\text{Angle}\). 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.

12.3 Setting a label

Labels are textual information you can display on vector features. They add details you could not necessarily represent using symbols.

The *Style Manager* dialog allows you to create a set of labels or text formats (ie the appearance of the text, including font, size, colors, shadow, background…). Each of these items could later be applied to layers in the *Labels* tab of the vector *Layer Properties* dialog or *Layer Styling* panel or using the *Layer Labeling Options* button of the *Labels toolbar*. You can also directly configure them in the abovementioned dialogs.

The *Label Settings* dialog allows you to configure smart labeling for vector layers. Setting a label includes configuring the *text format*, and how the label relates with the features or other labels (through *placement*, *rendering* and *callout)*.

12.3.1 Formatting the label text

To create text formats, you can:

1. Open the *Style Manager* dialog
2. Activate the *Text format* tab

![Obr. 12.13: Text formats in Style Manager dialog](image-url)
3. Press the **Add item** button. The *Text Settings* dialog opens with the following properties. As usual, these properties are *data-definable.*

**Text tab**

![Text Settings dialog](image)

Obr. 12.14: Text settings - Text tab

In the *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 struck through
- the *Size* in any supported unit
- the *Color*
- and the *Opacity*.

At the bottom, a text formats list widget shows a filterable list of text formats stored within your *style manager database*. This allows you to easily set text formats to match styles saved in the local style database, and also to add a new text format to the style database based on the current settings. Press the *Save format…* button to store the current text format in the *Style Manager*, providing a name and tag(s).

Likewise, a label settings list widget is shown when configuring labels, allowing you to pick from the widget or to add new styles to it.
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 All uppercase, All lowercase or Capitalize first letter. Note that the last option modifies only the first letter of each word and leaves the other letters in the text untouched.

- Under Spacing, change the space between words and between individual letters.

- 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.

- Use the Blend mode option to determine how your labels will mix with the map features below them (more details at Režim míchání).

- 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
Format the Alignment: typical values available are Left, Right 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.

- 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**

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 Režim míchání).

• 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
• 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.

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 Režim míchání).

• The Fill color, Stroke color and Stroke width for shape types other than the marker symbol. Use the Load symbol parameters to revert changes on an SVG symbol to its default settings.

• A Pen join style: it can be Round, Miter or Bevel (applies to rectangle and square shapes only)

• Draw effects to add advanced paint effects for improving text readability, eg through outer glows and blurs.

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 shadowed, ie:
  – The angle: clockwise, it depends on the underlying item orientation
  – The distance of offset from the item being shadowed
  – 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.
Obr. 12.17: Text settings - Background tab
Obr. 12.18: Text settings - Shadow tab
• 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 *Režim míchání*).

### 12.3.2 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.

![Image of a map with callouts](image)

Obr. 12.19: 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
2. Select the **Line style** with full capabilities of a *line symbol* including layer effects, and data-defined settings
3. Set the **Minimum length** of callout lines
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 **Offset from label area** option: controls the distance from the label closest corner where callout lines end. This avoids drawing lines right up against the text.
6. ![Draw lines to all features parts](image) from the feature’s label
7. Set an **Anchor point** on the (polygon) feature to use as end point of the connector line: available options are:
12.3.3 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

With the Cartographic placement mode, 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 either from the point feature itself or from the bounds of the symbol used to represent the feature. 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.

By default, placements are prioritised in the following order:

1. top right
2. top left
3. bottom right
4. bottom left
5. middle right
6. middle left
7. top, slightly right
8. bottom, slightly left.

Placement priority can, however, 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.

The Around point setting places the label in an equal radius (set in Distance) circle around the feature. The placement of the label can even be constrained using the Quadrant option.

With the Offset from point, labels are placed at a fixed offset from the point feature. You can select the Quadrant in which to place your label. You are also able to set the Offset X,Y distances between the points and their labels and can alter the angle of the label placement with the Rotation setting. Thus, placement in a selected quadrant with a defined rotation is possible.
Placement for line layers

Label options for line layers include □ Parallel, □ Curved or □ Horizontal. For the □ Parallel and □ Curved options, you can set the position to □ Above line, □ On line and □ Below line. It’s possible to select several options at once. In that case, QGIS will look for the optimal label position. For Parallel and curved placement options, you can also use the line orientation for the position of the label. Additionally, you can define a Maximum angle between curved characters when selecting the □ Curved option (see Figure_labels_placement_line).

Obr. 12.20: Label placement examples in lines

For all three placement options, in Repeat, you can set up a minimum distance for repeating labels. The distance can be in mm or in map units.

Placement for polygon layers

You can choose one of the following options for placing labels in polygons (see figure_labels_placement_polygon):

- □ Offset from centroid,
- □ Horizontal (slow),
- □ Around centroid,
- □ Free (slow),
- □ Using perimeter,
- and □ Using perimeter (curved).

In the Offset from centroid settings you can specify if the centroid is of the □ visible polygon or □ whole polygon. That means that either the centroid is used for the polygon you can see on the map or the centroid is determined for the whole polygon, no matter if you can see the whole feature on the map. You can place your label within a specific quadrant, and define offset and rotation.

The Around centroid setting places the label at a specified distance around the centroid. Again, you can define □ visible polygon or □ whole polygon for the centroid.

With the Horizontal (slow) or Free (slow) options, QGIS places at the best position either a horizontal or a rotated label inside the polygon.

With the Using perimeter option, the label will be drawn next to the polygon boundary. The label will behave like the parallel option for lines. You can define a position and a distance for the label. For the position, □ Above line, □
On line, Below line and Line orientation dependent position are possible. You can specify the distance between the label and the polygon outline, as well as the repeat interval for the label.

The Using perimeter (curved) option helps you draw the label along the polygon boundary, using a curved labeling. In addition to the parameters available with Using perimeter setting, you can set the Maximum angle between curved characters polygon, either inside or outside.

In the priority section you can define the priority with which labels are rendered for all three vector layer types (point, line, polygon). This placement option interacts with the labels from other vector layers in the map canvas. If there are labels from different layers in the same location, the label with the higher priority will be displayed and the others will be left out.

12.3.4 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).

Poznámka: 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.

• While rendering labels and in order to display readable labels, QGIS automatically evaluates the position of the labels and can hide some of them in case of collision. You can however choose to Show all labels for this layer (including colliding labels) in order to manually fix their placement (see The Label Toolbar).
• 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.

Feature options

Under Feature options:

• You can choose to label every part of a multi-part feature and limit the number of features to be labeled.
• 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.

Obstacles

An obstacle is a feature QGIS tries as far as possible to not place labels over. From the Obstacles frame, you can manage the covering relation between labels and features:

• Activate the Discourage labels from covering features option to decide whether features of the layer should act as obstacles for any label (including labels from other features in the same layer).

Instead of the whole layer, you can define a subset of features to use as obstacles, using the data-defined override control next to the option.

• The priority control slider for obstacles allows you to make labels prefer to overlap features from certain layers rather than others. A Low weight obstacle priority means that features of the layer are less considered as obstacles and thus more likely to be covered by labels. This priority 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 type of obstacle the features could be, by minimising the labels placement:
  – 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 boundary of the polygon (prefers placing labels outside or completely inside the polygon). E.g., it can be useful for regional boundary layers, where the features cover an entire area. In this case, it’s impossible to avoid placing labels within these features, and it looks much better to avoid placing them over the boundaries between features.
Managing Data Source

13.1 Opening Data

As part of an Open Source Software ecosystem, QGIS is built upon different libraries that, combined with its own providers, offer capabilities to read and often write a lot of formats:

- 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, DB2 or MSSQL Spatial, MySQL…

- Web map and data services (WM(T)S, WFS, WCS, CSW, XYZ tiles, ArcGIS services, …) are also handled by QGIS providers. See QGIS as OGC Data Client 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 - The QGIS Style File Format) and virtual and memory layers.

More than 80 vector and 140 raster formats are supported by GDAL and QGIS native providers.

Poznámka: 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 Settings → Options → GDAL menu (for raster) 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 button, available on the Data Source Manager Toolbar, or by pressing Ctrl+L. The Data Source Manager dialog offers a unified interface to open vector or raster file-based data as well as databases or web services supported by QGIS. It can be set modal or not with the Modeless data source manager dialog in the Settings → Options → General menu.
Obr. 13.1: 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.

### 13.1.1 The Browser Panel

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 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 figure_browser_panels. 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 Prostorové záložky)
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:
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** →
  - Directory…
  - GeoPackage…
  - ShapeFile…
- **Add as a Favorite**
- **Hide from Browser**
- **Fast Scan this Directory**
- **Open Directory**
- **Open in Terminal**
- **Vlastnosti…**
- **Directory Properties…**

Favourites, can also be removed and renamed:
• Rename favourite…
• Remove favourite

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:

• Delete File „<name of file>“…
• Export Layer → To File…
• Add Layer to Project
• Layer Properties
• 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.

Tip: 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.

Tip: 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.

13.1.2 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, MSSQL, DB2, 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.

13.1.3 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.

Poznámka: Some external plugins also provide tools to open specific format files in QGIS.
Obr. 13.3: DB Manager dialog

13.1. Opening Data
Loading a layer from a file

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.

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

6. You can specify the encoding for vector file if desired

7. Press Add to load the file in QGIS and display them in the map view. figure_vector_loaded shows QGIS after loading the alaska.shp file.

Obr. 13.6: QGIS with Shapefile of Alaska loaded

Poznámka: 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 ...

- 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, MSSQL 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.
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 layer and press Add. The layer will be added using the native mesh rendering.

Importing a delimited text file

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.
Obr. 13.8: 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**.

As you set the parser properties, a sample data preview updates at the bottom of the dialog.

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 appropriate 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.

Importing a DXF or DWG file

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.

**Poznámka:** 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 Target package, i.e. the new GeoPackage file that will store the data. If an existing file is provided, then it will be overwritten.
2. Specify the coordinate reference system of the data in the drawing file.
3. Check Expand block references to import the blocks in the drawing file as normal elements.
4. Check Use curves to promote the imported layers to a curved geometry type.
5. Use the Import button to select the DWG/DXF file to use (one per geopackage). 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.
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.
Obr. 13.9: Import dialog for DWG/DXF files
Importing OpenStreetMap Vectors

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.

SpatiaLite Layers

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

Loading GPS data in QGIS can be done using the core plugin GPS Tools. Instructions are found in section GPS Plugin.

GRASS

Working with GRASS vector data is described in section GRASS GIS Integration.

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 recommended 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 MSSQL Spatial Layer
- Add Oracle Spatial Layer… or by typing Ctrl+Shift+O
- Add DB2 Spatial Layer… or by typing Ctrl+Shift+2

These tools are accessible either from the Manage Layers Toolbar and the Layer ➤ Add Layer ➤ menu. Connecting to SpatiaLite database is described at SpatiaLite Layers.

Tip: 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 MSSQL Spatial;
- Connecting to Oracle Spatial;
- Connecting to DB2 Spatial.

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 figure_add_postgis_tables). To access the connection manager, click on the New button to display the 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.
  - **Username**: 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.

**Varování: 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.
Obr. 13.10: Create a New PostGIS Connection Dialog
Therefore, it is advisable to save your credentials in an Authentication configuration instead (Configurations tab - See Ověřovací systém 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](#)

**Tip:** 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 figure_add_postgis_tables).

**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

**Poznámka:** 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 need to do something like `export PGSERVICE=water_service` before doing your `psql` commands.

You can find all the PostgreSQL parameters [here](#).

**Poznámka:** If you don’t want to save the passwords in the service file you can use the `.pg_pass` option.

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.

**Varování:** 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 \textit{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;
- **Workspace**: Workspace to switch to.

Optionally, you can activate the following checkboxes:

- **Only look in metadata table**: restricts the displayed tables to those that are in the \texttt{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 \texttt{all_tables.num_rows}. Table extents are always determined with the \texttt{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**.

\textbf{Tip: Oracle Spatial Layers}

Normally, an Oracle Spatial layer is defined by an entry in the \texttt{USER_SDO_METADATA} table. To ensure that selection tools work correctly, it is recommended that your tables have a \texttt{primary key}.

Connecting to DB2 Spatial

In addition to some of the options described in \textit{Creating a stored Connection}, the connection to a DB2 database (see \textit{DB2 Spatial Layers} for more information) can be specified using either a Service/DSN name defined to ODBC or Driver, Host and Port.

An ODBC Service/DSN connection requires the service name defined to ODBC.

A driver/host/port connection requires:

- **Driver**: Name of the DB2 driver. Typically this would be IBM DB2 ODBC DRIVER.
- **DB2 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 \texttt{localhost} here.
- **DB2 Port**: Port number the DB2 database server listens on. The default DB2 LUW port is 50000. The default DB2 z/OS port is 446.

\textbf{Tip: DB2 Spatial Layers}

A DB2 Spatial layer is defined by a row in the \texttt{DB2GSE.ST_GEOMETRY_COLUMNS} view.
**Poznámka:** In order to work effectively with DB2 spatial tables in QGIS, it is important that tables have an INTEGER or BIGINT column defined as PRIMARY KEY and if new features are going to be added, this column should also have the GENERATED characteristic.

It is also helpful for the spatial column to be registered with a specific spatial reference identifier (most often 4326 for WGS84 coordinates). A spatial column can be registered by calling the `ST_Register_Spatial_Column` stored procedure.

### Connecting to MSQL Spatial

In addition to some of the options in *Creating a stored Connection*, creating a new MSQL connection dialog proposes you to fill a Provider/DSN name. You can also display available databases.

### Loading a Database Layer

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 *Query Builder*) 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.

**Tip:** 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.
13.1.4 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)

13.1.5 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.

13.1.6 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. Chapter Working with OGC Data contains descriptions of these capabilities.
Using XYZ Tile services

XYZ Tile services can be found in the *XYZ Tiles* top level entry in the *Browser*. By default, the OpenStreetMap XYZ Tile service is configured. You can add other services that use the XYZ Tile protocol by choosing *New Connection* in the XYZ Tiles context menu (right-click to open). *figure_xyz_tiles_openstreetmap* shows the dialog with the OpenStreetMap XYZ Tile service configuration.

![XYZ Connection](xyz.png)

Obr. 13.12: XYZ Tiles - OpenStreetMap configuration

Configurations can be saved (*Save Connections*) to XML and loaded (*Load Connections*) through the context menu. Authentication configuration is supported. The XML file for OpenStreetMap looks like this:

```xml
<!DOCTYPE connections>
<qgsXYZTilesConnections version="1.0">
  <xyztiles url="https://tile.openstreetmap.org/{z}/{x}/{y}.png" zmin="0" zmax="19" password="" name="OpenStreetMap" username="" authcfg="" referer="" />
</qgsXYZTilesConnections>
```

Once a connection to a XYZ tile service is set, right-click over the entry to:
• Edit… the XYZ connection settings
• Delete the connection
• 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.

Examples of XYZ Tile services:
• 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.

13.1.7 Handling broken file paths

When the path to a data source is wrong, QGIS opens the Handle Unavailable Layers dialog. You can double-click in the Datasource field or click Browse to fix the path. It is possible to continue working with your project with the broken path by clicking Keep Unavailable Layers. Your layer is then displayed in the Layers panel, but without any data until you fix the path using the Unavailable layer icon next to it in the Layers panel, or Change Data Source… in the layer contextual menu. Another possibility is to Remove Unavailable Layers. As the last step, click Apply changes.

When a layer path has been fixed, QGIS scans through all other broken paths and tries to auto-fix those that have the same broken file path.

13.2 Creating Layers

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.

13.2.1 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 figure_create_geopackage.

Obr. 13.13: 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 Digitizing an existing layer.

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 available, 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.

Creating a new Shapefile layer

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 figure_create_shapefile.

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 (point, multipoint, line or polygon)

4. Specify whether the geometry should have Z (+ M values) or M values

5. 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. 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 Digitizing an existing layer.

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 Figure_create_spatialite.

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* and *Decimal number*.
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 *Digitizing an existing layer*.

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*. 

Obr. 13.14: Creating a new Shapefile layer dialog
Obr. 13.15: Creating a New SpatiaLite layer dialog
Creating a new GPX layer

To create a new GPX file, you first need to load the GPS plugin. Plugins ➔ Plugin Manager... opens the Plugin Manager Dialog. Activate the GPS Tools checkbox.

When this plugin is loaded, choose Create Layer ➔ Create new GPX Layer... from the Layer menu. In the dialog, choose where to save the new file and press Save. Three new layers are added to the Layers Panel: waypoints, routes and tracks.

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 figure_create_temporary. 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. Specify the coordinate reference system using the button.

By default, a new temporary scratch layer is created without any attributes. You can later add them using the New Field button in the layer's attribute table dialog or the Fields tab of its properties 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.

Tip: 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 section and the saved file replaces the temporary one in the Layers panel.

13.2.2 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.

Common parameters

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 (possible values are layer, Map view or user-defined extent)
- 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…
- Resolution
- 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
Obr. 13.17: 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**. In case you set your fields behavior with some *Edit widgets*, e.g. value map, you can keep the displayed values in the layer by checking **Replace all selected raw fields values by displayed values**.

- **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

**Poznámka**: 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.

**Tip**: 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 Exploring Data Formats and Fields 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.

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.
Obr. 13.18: Saving as a new vector layer
13.2.3 Creating new DXF files

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** (e.g. to support polyline width);
- **Export label as MTEXT elements** or TEXT elements.

![Obr. 13.19: Exporting a project to DXF dialog](image-url)
13.2.4 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).

Poznámka: 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).

13.2.5 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.

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
```

Poznámka: 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.
Performance issues

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:

```sql
SELECT id 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:

```sql
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.
13.3 Exploring Data Formats and Fields

13.3.1 Raster data

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.

13.3.2 Vektorová data

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, DB2, Oracle Spatial, MSSQL Spatial databases, and many more), QGIS may handle some of their properties differently. Support is provided by the OGR Simple Feature Library. This section describes how to work with these specificities.

Poznámka: 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. .shp file containing the feature geometries
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 at https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

GDAL 3.1 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 The Browser Panel)
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.

Delimited Text Files

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 The Browser Panel or Importing a delimited text file). 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 must 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 *Downloading sample data*) 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 (e.g. 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 OGR driver assumes all fields are strings (i.e. text) unless it is told otherwise. You can create a CSVT file to tell OGR (and QGIS) the data type of the different columns:

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Název</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole number</td>
<td>Integer</td>
<td>4</td>
</tr>
<tr>
<td>Decimal</td>
<td>Real</td>
<td>3.456</td>
</tr>
<tr>
<td>Date</td>
<td>Date (YYYY-MM-DD)</td>
<td>2016-07-28</td>
</tr>
<tr>
<td>Time</td>
<td>Time (HH:MM:SS+nn)</td>
<td>18:33:12+00</td>
</tr>
<tr>
<td>Date &amp; Time</td>
<td>DateTime (YYYY-MM-DD HH:MM:SS+nn)</td>
<td>2016-07-28 18:33:12+00</td>
</tr>
</tbody>
</table>
```

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.

PostGIS Layers

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 OGR layers in QGIS.

Tip: PostGIS Layers

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.

Primary key

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.

Zobrazit

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.
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:

```sql
SET XML OPTION DOCUMENT;
```

Filter database side

QGIS allows to filter features already on server side. Check `Settings Options Data Sources` 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`.

Správce databází

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 `Práce s projekcemi` for more information about spatial reference systems and projections.

Tip: Exporting datasets from PostGIS

There is also a tool for exporting PostGIS datasets to Shapefile format: `pgsql2shp`. It is shipped within your PostGIS distribution.
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 OGR 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 X):

```
export PG_USE_COPY=YES
```

ogr2ogr does not create spatial indexes like shp2pgsql 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 Improving Performance).

### Improving Performance

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](https://postgis.net)).

**Tip:** 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 *Literature and Web References*) 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);
```

(continues on next page)
Vector layers crossing 180° longitude

Many GIS packages don’t wrap vector maps with a geographic reference system (lat/lon) crossing the 180 degrees longitude line (http://postgis.refractions.net/documentation/manual-2.0/ST_Shift_Longitude.html). As result, if we open such a map in QGIS, we could see two widely separated locations, that should appear near each other. In Figure_vector_crossing, 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.

A work-around is to transform the longitude values using PostGIS and the \texttt{ST\_Shift\_Longitude} function. This function reads every point/vertex in every component of every feature in a geometry, and if the longitude coordinate is < 0°, it adds 360° to it. The result is a 0° - 360° version of the data to be plotted in a 180°-centric map.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{vector_crossing.png}
\caption{Map in lat/lon crossing the 180° longitude line}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{vector_crossing_map.png}
\caption{Crossing 180° longitude applying the \texttt{ST\_Shift\_Longitude} function}
\end{figure}

Použití

- Import data into PostGIS (Importing Data into PostgreSQL) using, for example, the DB Manager plugin.
- Use the PostGIS command line interface to issue the following command (in this example, „TABLE“ is the actual name of your PostGIS table): \texttt{gis\_data=\# update TABLE set the\_geom=ST\_Shift\_Longitude(the\_geom)};
- 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).
**SpatiaLite Layers**

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](https://qgis.org/en-US/site/for-users/best-practices/layers.html). 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](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](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/drv_geojsonseq.html](https://gdal.org/drv_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.

**DB2 Spatial Layers**

IBM DB2 for Linux, Unix and Windows (DB2 LUW), IBM DB2 for z/OS (mainframe) and IBM DashDB products allow users to store and analyse spatial data in relational table columns. The DB2 provider for QGIS supports the full range of visualization, analysis and manipulation of spatial data in these databases.

User documentation on these capabilities can be found at the [DB2 z/OS KnowledgeCenter](https://www.ibm.com/support/knowledgecenter/SS5215/DB2 z/OS KnowledgeCenter), [DB2 LUW KnowledgeCenter](https://www.ibm.com/support/knowledgecenter/SS5215/DB2 LUW KnowledgeCenter) and [DB2 DashDB KnowledgeCenter](https://www.ibm.com/support/knowledgecenter/SS5215/DB2 DashDB KnowledgeCenter).

For more information about working with the DB2 spatial capabilities, check out the [DB2 Spatial Tutorial](https://www.ibm.com/support/knowledgecenter/SS5215/DB2 Spatial Tutorial) on IBM DeveloperWorks.

The DB2 provider currently only supports the Windows environment through the Windows ODBC driver.

The client running QGIS needs to have one of the following installed:

- **DB2 LUW**

- **IBM Data Server Driver Package**
IBM Data Server Client

To open a DB2 data in QGIS, see the *The Browser Panel* or *Loading a Database Layer* section.

If you are accessing a DB2 LUW database on the same machine or using DB2 LUW as a client, the DB2 executables and supporting files need to be included in the Windows path. This can be done by creating a batch file like the following with the name `db2.bat` and including it in the directory `%OSGEO4W_ROOT%/etc/ini`:

```batch
@echo off
REM Point the following to where DB2 is installed
SET db2path=C:\Program Files (x86)\sql1ib
REM This should usually be ok - modify if necessary
SET gskpath=C:\Program Files (x86)\ibm\gsk8
SET Path=%db2path%\BIN;%db2path%\FUNCTION;%gskpath%\lib64;%gskpath%\lib;%path%
```
14.1 The Vector Properties Dialog

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]
- **Diagrams**
- **3D View**[1]
- **Fields**
- **Attributes Form**
- **Joins**
- **Auxiliary Storage**
- **Actions**
- **Display**
- **Dependencies**
- **Variables**
- **Metadata**
- **Legend**
- **External plugins**[2]
- **Digitizing**
- **External plugins**[2]
- **tabs**
- **QGIS Server**

[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.

**Tip:** 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.
Poznámka: Because properties (symbology, label, actions, default values, forms…) of embedded layers (see Vnořování projektů) 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.

14.1.1 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:

- based on the provider of the layer (format of storage, path, geometry type, data source encoding, extent…);
- picked from the filled metadata (access, links, contacts, history…);
- or related to its geometry (spatial extent, CRS…) or its attributes (number of fields, characteristics of each…).

14.1.2 Source Properties

Use this tab to define general settings for the vector layer.

Obr. 14.1: Source tab in vector Layer Properties dialog

Other than setting the Layer name to display in the Layers Panel, available options include:
Coordinate Reference System

- Displays the layer’s Coordinate Reference System (CRS). You can change the layer’s CRS, 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 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 reprojecion algorithms from Processing or Save it into another layer.

- Create spatial index (only for OGR-supported formats).

- Update extents information for a layer.

Query Builder

The Query Builder dialog is accessible through the eponym 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 Figure_vector_querybuilder. In the example, the filter combines the attributes

- \( \text{toa} \) (DateTime field: \( \text{cast("toa" as character)} > '2017-05-17' \) and \( \text{cast("toa" as character)} < '2019-12-24T18:00:00' \)),
- \( \text{name} \) (String field: \( \text{name} > 'S' \)) and
- \( \text{FID} \) (Integer field: \( \text{FID > 10} \))

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, MSSQL….) 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.

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 (\(=, >, \ldots\)), string comparison operator (LIKE), and logical operators (AND, OR, \ldots) 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).

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.
Obr. 14.2: Query Builder
Tip: 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.

14.1.3 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 Layer rendering widget.

Tip: Switch quickly between different layer representations

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.

Tip: Export vector symbology

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.

Features rendering

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 a point displacement and a heatmap renderers available while polygon layers can also be rendered with the 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.

Poznámka: 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.
Single Symbol Renderer

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.

![Single Symbol Renderer](image)

Obr. 14.3: Single symbol line properties

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.

Categorized Renderer

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.

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 button. Using expressions for categorizing avoids the need to create an ad hoc field for symbology purposes (e.g., if your classification criteria are derived from one or more attributes).

   The expression used to classify features can be of any type; e.g., it can:
be a comparison. In this case, QGIS returns values 1 (True) and 0 (False). Some examples:

```plaintext
myfield >= 100
$id = @atlas_featureid
myfield % 2 = 0
within( $geometry, @atlas_geometry )
```

• combine different fields:

```plaintext
concat( field_1, ' ', field_2 )
```

• be a calculation on fields:

```plaintext
myfield % 2
year( myfield )
field_1 + field_2
substr( field_1, -3 )
```

• be used to transform linear values to discrete classes, e.g.:

```plaintext
CASE WHEN x > 1000 THEN 'Big' ELSE 'Small' END
```

• combine several discrete values into a single category, e.g.:

```plaintext
CASE
  WHEN building IN ('residence', 'mobile home') THEN 'residential'
  WHEN building IN ('commercial', 'industrial') THEN 'Commercial and...
  Industrial'
END
```

Tip: 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, i.e. 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 (e.g., 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.

Tip: 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 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.

Tip: Edit categories directly from the Layers panel
When a layer symbology is based on a categorized, graduated or rule-based symbology mode, you can edit each of the categories from the Layers Panel. Right-click on a sub-item of the layer and you will:

- **Toggle items visibility**
- **Show all items**
- **Hide all items**
- Modify the symbol color thanks to the *color selector* wheel
- *Edit symbol*... from the *symbol selector* dialog
- *Copy symbol*
- *Paste symbol*

---

**Graduated Renderer**

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 (using the fields listbox or the *Set value expression* function)
- The symbol (using the Symbol selector dialog)
- The legend format and the precision
- The method to use to change the symbol: color or size
- 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.

**Poznámka:** 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 Interval: each class has the same size (e.g. values from 0 to 16 and 4 classes, each class has a size of 4).
- Quantile: each class will have the same number of element inside (the idea of a boxplot).
- Natural Breaks (Jenks): the variance within each class is minimal while the variance between classes is maximal.
- Standard Deviation: classes are built depending on the standard deviation of the values.
- 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://astrostatistics.psu.edu/datasets/R/html/base/html/pretty.html)

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 *figure_graduated_symbology* shows the graduated rendering dialog for the major_rivers layer of the QGIS sample dataset.

Tip: Thematic maps using an expression

Categorized and graduated thematic maps can be created using the result of an expression. In the properties dialog for vector layers, the attribute chooser is extended with a ° Set column expression ° function. So you don’t need to write the
classification attribute to a new column in your attribute table if you want the classification attribute to be a composite of multiple fields, or a formula of some sort.

**Proportional Symbol and Multivariate Analysis**

Proportional Symbol and Multivariate Analysis are not rendering types available from the Symbology rendering dropdown 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).

![Symbol size settings](image)

Obr. 14.6: Scaling airports size based on elevation of the airport

**Creating multivariate analysis**

14.1. The Vector Properties Dialog 239
A multivariate analysis rendering helps you evaluate the relationship between two or more variables e.g., one can be represented by a color ramp while the other is represented by a size.

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.

Obr. 14.7: Multivariate example with scaled size legend

**Rule-based Renderer**

The Rule-based renderer is used to render all the features from a layer, using rule-based symbols whose aspect reflects the assignment of a selected feature’s attribute to a class. The rules are based on SQL statements and can be nested. The dialog allows rule grouping by filter or scale, and you can decide if you want to enable symbol levels or use only the first-matched rule.

To create a rule:

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. And press 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.

Selecting a rule, you can also organize its features in subclasses using the Refine selected rules drop-down menu. Automated rule refinement can be based on:

- scales;
- categories: applying a categorized renderer;
- or ranges: applying a graduated renderer.

Refined classes appear like sub-items of the rule, in a tree hierarchy and like above, you can set symbology of each class.

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 (categorized renderer) or Ranges (graduated renderer).

The created rules also appear in a tree hierarchy in the map legend. Double-click the rules in the map legend and the Symbology tab of the layer properties appears showing the rule that is the background for the symbol in the tree.

The example in figure_rule_based_symbology shows the rule-based rendering dialog for the rivers layer of the QGIS sample dataset.

### Point displacement Renderer

The Point Displacement renderer works to visualize all features of a point layer, even if they have the same location. To do this, the renderer takes the points falling in a given Distance tolerance from each other and places them around their barycenter following different 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.
The Center symbol widget helps you customize the symbol and color of the middle point. For the distributed points symbols, you can apply any of the No symbols, Single symbol, Categorized, Graduated or Rule-based renderer using the Renderer drop-down list and customize them using the Renderer Settings… button.

While the minimal spacing of the Displacement lines depends on the point symbol renderer’s, you can still customize some of its settings such as the Stroke width, Stroke color and Size adjustment (eg, to add more spacing between the rendered points).

Use the Labels group options to perform points labeling: the labels are placed near the displaced position of the symbol, and not at the feature real position. Other than the Label attribute, Label font and Label color, you can set the Minimum map scale to display the labels.

Poznámka: 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. Based on a specified Distance, points that fall within 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.

From the main dialog, you can:

- 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.
- use the Renderer drop-down list to apply any of the other feature rendering types to the layer (single, categorized, rule-based…). Then, push the Renderer Settings… button to configure features' symbology as usual. Note that this renderer is only visible on features that are not clustered. Also, when the symbol color is the same for all the point features inside a cluster, that color sets the @cluster_color variable of the cluster.
Poznámka: 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.
Inverted Polygon Renderer

The \textit{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.

![Image of Inverted Polygon dialog]

Obr. 14.11: Inverted Polygon dialog

Heatmap Renderer

With the \textit{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.

2.5D Renderer

Using the \textit{2.5D} renderer it’s possible to create a 2.5D effect on your layer’s features. You start by choosing a \textit{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 \textit{Angle} (in degrees) to recreate the viewer position (0° means west, growing in counter clockwise). Use advanced configuration options to set the \textit{Roof Color} and \textit{Wall Color}. If you would like to simulate solar radiation on the features walls, make sure to check the \textit{Shade walls based on aspect} option. You can also simulate a shadow by setting a \textit{Color} and \textit{Size} (in map units).

Tip: 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
Obr. 14.12: Heatmap dialog

Obr. 14.13: 2.5D dialog
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.

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 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 Režim míchání.

- **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…

Obr. 14.14: Layer rendering options

Other Settings

Symbol levels

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.
Poznámka: 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.

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**:

- set 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.

A preview of the legend is displayed in the right panel of the dialog and updated as you set the parameters. For collapsed legend, a leader line from the horizontal center of the symbol to the corresponding legend text is drawn.

![Obr. 14.17: Setting size scaled legend](image)

**Poznámka:** Currently, data-defined size legend for layer symbology can only be applied to point layer using single, categorized or graduated symbology.

### Draw effects

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 [figure_effects_source](image)). 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.
• **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*.

• **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.

• **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
The Colorize dialog allows you to change the brightness, contrast, saturation, and color of the effect. Drop Shadow also has the option to change the blur radius and the color of the effect.

**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.

**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.

**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.

**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.
- **Translate X,Y**: Changes the position of the item based on a distance given on the X and/or Y axis.

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 the 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 Layer rendering 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


14.1. The Vector Properties Dialog

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).

### 14.1.4 Labels Properties

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.

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 if you want to define labels based on expressions - See *Define labels based on expressions*.

Below are displayed options to customize the labels, under various tabs:
Setting the automated placement engine

You can use the automated placement settings to configure a global and automated behavior of the labels. In the top right corner of the Labels tab, click the button, opening a dialog with the following options:

- The **Number of candidates** controls how many label placement candidates should be generated for each feature type. The more candidates generated, the better the labeling will be - but at a cost of rendering speed. Smaller number of candidates results in less labels placed but faster redraws.

- **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.

### Rule-based labeling

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*.

To create a rule, select the **Rule-based labeling** option in the main drop-down list from the Labels tab and click the **+** button at the bottom of the dialog. Then fill the new dialog with a description and an expression to filter features. You can also set a scale range in which the label rule should be applied. The other options available in this dialog are the common settings seen beforehand.

A summary of existing rules is shown in the main dialog (see *figure_labels_rule_based*). 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 the **edit** button or a double-click.
Obr. 14.28: Rules settings

Kapitola 14. Working with Vector Data
Define labels based on expressions

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 figure_labels_expression, 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.

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.

Let’s have a look at some examples:

1. Label based on two fields 'name' and 'place' with a comma as separator:
2. Label based on two fields 'name' and 'place' with other texts:

   ```sql
   '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")
   ``

Returns:

| 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:

   ```sql
   concat('My name is ', name, ' and I live in ' || place)
   ``

Returns:

| 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:

   ```sql
   concat('My name is ', "name", '\n', 'I live in ', "place")
   ``

Returns:

| 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:

   ```sql
   'The area of ' || "place" || ' has a size of ' || round($area/10000) || ' ha'
   ``

Returns:

| 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:

   ```sql
   concat('This place is a ',
   CASE WHEN "population" <= 50000 THEN 'town' ELSE 'city' END)
   ``

Returns:

| 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):

   ```sql
   CASE WHEN "population" > 50000 THEN "NAME" END
   ``

Returns:
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 Výrazy chapter for more information and examples on expressions.

### Using data-defined override for labeling

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 a 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 defined override widget’s description and manipulation in Data defined override setup section.

**Tip:** 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 Data defined override setup.
The Label Toolbar

The Label Toolbar provides some tools to manipulate label or diagram properties.

While for readability, label has been used below to describe the Label toolbar, note that when mentioned in their name, the tools work almost the same way with diagrams:

- **Highlight Pinned Labels and Diagrams**. If the vector layer of the label is editable, then the highlighting is green, otherwise it's blue.

- **Toggles 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 dragging an area, you pin label(s). If you click or drag an area holding Shift, label(s) are unpinned. Finally, you can also click or drag an area holding Ctrl to toggle the pin status of label(s).

- **Show/Hide Labels and Diagrams**. If you click on the labels, or click and drag an area holding Shift, they are hidden. When a label is hidden, you just have to click on the feature to restore its visibility. If you drag an area, all the labels in the area will be restored.

- **Moves a Label or Diagram**. You just have to drag the label to the desired place.
- Rotate a Label. Click the label and move around and you get the text rotated.

- 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**.

### Varování: 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!

### Poznámka: 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** function (see **figure_labels_coordinate_data_defined**).

1. Import `lakes.shp` from the QGIS sample dataset.

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.

   ![Data defined](image)

   Obr. 14.33: Labeling of vector polygon layers with data-defined override

4. Zoom into a lake.

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 **figure_labels_move**). The new position of the label is saved in the `xlabel` and `ylabel` columns of the attribute table.

7. Using **The Geometry Generator** with the expression below, you can also add a linestring symbol layer to connect each lake to its moved label:

   ```python
   make_line( centroid( $geometry ), make_point( "xlabel", "ylabel" ) )
   ```

### Poznámka: The Auxiliary Storage Properties mechanism may be used with data-defined properties without having an editable data source.
The Diagrams tab allows you to add a graphic overlay to a vector layer (see `figure_diagrams_attributes`). The current core implementation of diagrams provides support for:

- No diagrams: the default value with no diagram displayed over the features;
- Pie charts, a circular statistical graphic divided into slices to illustrate numerical proportion. The arc length of each slice is proportional to the quantity it represents;
- Text diagrams, a horizontally divided circle showing statistics values inside;
- and Histograms.

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.

**Tip:** 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:

- Attributes
- Rendering
- Size
Attributes defines which variables to display in the diagram. Use the `add item` button to select the desired fields into the 'Assigned Attributes' panel. Generated attributes with `Výrazy` 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.

**Obr. 14.35: Diagram properties - Attributes tab**

**Vykreslování**

*Rendering* defines how the diagram looks like. It provides general settings that do not interfere with the statistic values such as:

- the graphic’s opacity, its outline width and color;
- and, depending on the type of diagram:
  - the width of the bar in case of histogram;
  - the circle background color in case of text diagram, and the font used for texts;
  - the orientation of the left line of the first slice represented in pie chart. Note that slices are displayed clockwise.

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;
• 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.

Obr. 14.36: Diagram properties - Rendering tab

Size

Size is the main tab to set how the selected statistics are represented. The diagram size units can be 'Millimeter', 'Points', Pixels, 'Map Units' or 'Inches'. You can use:

• Fixed size, an unique size to represent the graphic of all the features, except when displaying histogram
• or Scaled size, based on an expression using layer attributes.
**Placement**

*Placement* helps to define diagram position. According to the layer geometry type, it offers different options for the placement:

- 'Over the point' or 'Around the point' for point geometry. The latter variable requires a radius to follow.
- 'Over the line' or 'Around the line' for line geometry. Like point feature, the last variable requires a distance to respect and user 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 here you can also use the line orientation for the position of the diagram.
- 'Over the centroid', 'Around the centroid' (with a distance set), 'Perimeter' and anywhere 'Inside polygon' are the options for polygon features.

The diagram can also be placed using feature data to fill the coordinates $X$ and $Y$ fields.

The placement of the diagrams can interact with the labeling, so you can detect and solve position conflicts between diagrams and labels by setting the **Priority** slider value.
The **Options** tab has settings only in case of histogram. You can choose whether the bar orientation should be 'Up', 'Down', 'Right' and 'Left'.

**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.

**Case Study**

We will demonstrate an example and overlay on the Alaska boundary layer a text diagram showing temperature data from a climate vector layer. Both vector layers are part of the QGIS sample dataset (see section *Downloading sample data*).

1. First, click on the **Load Vector** icon, browse to the QGIS sample dataset folder, and load the two vector shape layers *alaska.shp* and *climate.shp*.
2. Double click the *climate* layer in the map legend to open the *Layer Properties* dialog.
3. Click on the **Diagrams** tab and from the **Diagram type** combo box, select 'Text diagram'.

---

**Obr. 14.38: Vector properties dialog with diagram properties, Placement tab**

Možnosti

The **Options** tab has settings only in case of histogram. You can choose whether the bar orientation should be 'Up', 'Down', 'Right' and 'Left'.

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.

Case Study

We will demonstrate an example and overlay on the Alaska boundary layer a text diagram showing temperature data from a climate vector layer. Both vector layers are part of the QGIS sample dataset (see section *Downloading sample data*).

1. First, click on the **Load Vector** icon, browse to the QGIS sample dataset folder, and load the two vector shape layers *alaska.shp* and *climate.shp*.
2. Double click the *climate* layer in the map legend to open the *Layer Properties* dialog.
3. Click on the **Diagrams** tab and from the **Diagram type** combo box, select 'Text diagram'.
4. In the Appearance tab, we choose a light blue as background color, and in the Size tab, we set a fixed size to 18 mm.

5. In the Position tab, placement could be set to 'Around Point'.

6. In the diagram, we want to display the values of the three columns T_F_JAN, T_F_JUL and T_F_MEAN.
   So, in the Attributes tab first select T_F_JAN and click the button, then repeat with T_F_JUL and finally T_F_MEAN.

7. Now click Apply to display the diagram in the QGIS main window.

8. You can adapt the chart size in the Size tab. Activate the Scaled size and set the size of the diagrams on the basis of the maximum value of an attribute and the Size option. If the diagrams appear too small on the screen, you can activate the Increase size of small diagrams checkbox and define the minimum size of the diagrams.

9. Change the attribute colors by double clicking on the color values in the Assigned attributes field. Figure_ diagrams_mapped gives an idea of the result.

10. Finally, click OK.

   ![Diagram from temperature data overlayed on a map](image)

   **Obr. 14.39: Diagram from temperature data overlayed on a map**

Remember that in the Position tab, a Data defined position of the diagrams is possible. Here, you can use attributes to define the position of the diagram. You can also set a scale-dependent visibility in the Appearance tab.

The size and the attributes can also be an expression. Use the button to add an expression. See Výrazy chapter for more information and example.

**Using data-defined override**

As mentioned above, you can use some custom data-defined to tune the diagrams rendering:

- position in Placement tab by filling X and Y fields
- visibility in Appearance tab by filling the Visibility field

See Using data-defined override for labeling for more information.
14.1.6 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 symbol whose properties can be data-defined or not.
- **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.

Depending on the layer geometry type, various properties are available for 3D rendering.

![Layer Properties - buildings | 3D View](image)

Obr. 14.40: 3D properties of a polygon layer

**Point Layers**

- You can define different simple 3D shapes like Sphere, Cylinder, Cube, Cone, Plane and Torus defined by their Radius, Size or Length. The unit of size of the 3D shapes refers to the CRS of the project.
- The shading of the 3D shapes can be defined by the menus Diffuse, Ambient, Specular and Shininess (see [https://en.wikipedia.org/wiki/Phong_reflection_model#Description](https://en.wikipedia.org/wiki/Phong_reflection_model#Description))
- If you choose 3D Model, the location will be determined by a simple point coordinate.
- For visualizing 3D point clouds you can use Billboard Shapes defined by the Billboard Height, Billboard symbol and Altitude clamping. The symbol will have a stable size.
• **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.

• **Translation** can be used to move objects in x, y and z axis.

• You can define a *Scale factor* for the 3D shape as well as a *Rotation* around the x-, y- and z-axis.

**Line layers**

• 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 can be defined in the menus *Diffuse*, *Ambient*, *Specular* and *Shininess*.

**Polygon Layers**

• As for the other ones, *Height* can be defined in CRS units.

• Again, *Extrusion* is possible for missing z-values.

• The *Altitude clamping*, *Altitude binding* can be defined as explained above.

• There is an additional option to **Add back faces** and **Invert normals**.

• You can define **Edges** by *Width* and *Color*.

**Application example**

To go through the settings explained above you can have a look at [https://public.cloudmergin.com/projects/saber/luxembourg/tree](https://public.cloudmergin.com/projects/saber/luxembourg/tree).

**14.1.7 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.

You can also rename fields by double-clicking its name. This is only supported for data providers like PostgreSQL, Oracle, Memory layer and some OGR layer depending on the OGR data format and 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.

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.

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. When serving the layer as WMS or WFS, you can also check here which fields could be retrieved.

![Image of Fields properties tab](image)

**Obr. 14.41: Fields properties tab**

### 14.1.8 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 e.g., 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.

The autogenerated form

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, as shown for example in figure_fields_form.

Obr. 14.42: 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 (that you can customize) 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. Drag and drop fields from the other panel to re-add them. The same field can be added multiple times.

3. Drag and drop fields within the Form Layout panel to reorder their position.

4. Add containers (tab or group frames) to associate fields that belong to the same category and better structure the form.
1. The first step is to use the icon to create a tab in which fields and groups will be displayed.

2. Then set the properties of the container, ie:
   - the name
   - the type, i.e., a tab or a group in container (a group inside a tab or another group)
   - and the number of columns the embedded fields should be distributed over.

3. You can create as many containers as you want; press the icon again to create another tab or a group frame under an existing tab.

5. The next step is to assign the relevant fields to each container, by simple drag and drop. Groups and tabs can also be moved in the same way.

6. 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

7. Apply the layer’s properties dialog

8. Open a feature attribute form (eg, using the Identify features tool) and it should display the new form.

Using custom ui-file

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).

An example is (in module MyForms.py):

```python
def open(dialog, layer, feature):
    geom = feature.geometry()
    control = dialog.findChild(QWidged, "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.
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

*Show label*: indicates whether the field name should be displayed in the form.

General options

- *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.

- *Label on top*: places the field name above or beside the widget in the feature form.
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.

**Poznámka:** 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…

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. `regexp_match(col0,'A-Za-z')` to ensure that the value of the field `col0` has only alphabetical letter. 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 a yellow or red cross is displayed near the field. 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 and it does not prevent you to save the changes with the „wrong” values;
  - A red cross can not be ignored and does not allow you to save your modifications until they meet the constraints. It appears when the Enforcement option is checked.

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:

- **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 also be used to display a hyperlink (to document path), a picture or a web page.

• **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.

• **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 widget lets you embed the feature form of the referenced layer on the feature form of the actual layer. See *Creating one or many to many relations*.

• **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 completion checkbox is enabled.

**Tip:** 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.

### 14.1.9 Joins Properties

The **Joins** tab allows you to associate features of the current layer (called **Target layer**) to features from another loaded vector layer (or table). The join is based on an attribute that is shared by the layers. The layers can be geometryless (tables) or not but their join attribute should be of the same type.

To create a join:

1. Click the **Add new join** button. The **Add vector join** dialog appears.
2. Select the **Join layer** you want to connect with the target vector layer.
3. Specify the **Join field** and the **Target field** that are common to both the join layer and the target layer.
4. Press **OK** and a summary of selected parameters is added to the **Join** panel.

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. QGIS provides more options to tweak the join:
Obr. 14.45: Join an attribute table to an existing vector layer
• 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

• 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

QGIS currently has support for joining non-spatial table formats supported by OGR (e.g., CSV, DBF and Excel), delimited text and the PostgreSQL provider.

14.1.10 Auxiliary Storage Properties

The regular way to customize styling and labeling is to use data-defined properties as described in Data defined override setup. 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:

Labeling

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):

As soon as an auxiliary layer is configured for the current data source, you can retrieve its information in the tab:

The auxiliary layer now has these characteristics:

• the primary key is ID,
• there are 0 features using an auxiliary field,
14.1. The Vector Properties Dialog

Obr. 14.46: Auxiliary Storage tab

Obr. 14.47: Auxiliary Layer creation dialog
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:

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:

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 (Obr. 14.47) is first displayed to select the primary key to use for joining.

Symbology

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:

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:

Once created, the auxiliary field can be retrieved in the auxiliary storage tab:
Auxiliary storage tables can contain additional data that should only belong to the project file. For instance, specific location or rotation for labels. Auxiliary data are saved in qgd files. New fields can be added from any data-defined widget when needed. Be aware that this information will NOT be saved in the data source but only in the project file.

Obr. 14.49: Auxiliary Fields
Obr. 14.50: Data-defined properties automatically created

Obr. 14.51: Data-defined property menu for symbol
14.1. The Vector Properties Dialog

New auxiliary field parameters

- **Type**: String
- **Name**: symbol0

Obr. 14.52: Name of the auxiliary field for a symbol

Obr. 14.53: Auxiliary field symbol
Attribute table and widgets

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:

![Obr. 14.54: Form with auxiliary fields](image)

Management

The Auxiliary Layer menu allows you to manage the auxiliary fields:

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 Labeling, 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.

14.1.11 Actions Properties

QGIS 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.

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 divided into six types and can be used like this:

- Generic, Mac, Windows and Unix actions start an external process.
- Python actions execute a Python expression.
- Generic and Python actions are visible everywhere.
- Mac, 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).

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).

**Defining Actions**

To define an attribute action, open the vector Layer Properties dialog and click on the Actions tab. In the Actions tab, click the Add an 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 using_actions below). 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 4 different choices:

1. **Feature Scope**: action is available when right click in the cell within the attribute table.
2. **Field Scope**: 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.
3. **Layer Scope**: 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.
4. **Canvas**: action is available in the main action button in the toolbar.

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
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 the 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.

If you are invoking an action that uses the %% notation, right-click on the field value in the Identify Results dialog or the Attribute Table dialog that you wish to pass to the application or script.

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
bash -c "echo "\"%taxon_name \%lat \%long\"" >> /tmp/species_localities.txt"
```

After selecting a few localities and running the action on each one, opening the output file will show something like this:

<table>
<thead>
<tr>
<th>Species</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acacia mearnsii -34.0800000000 150.0800000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acacia mearnsii -34.9000000000 150.1200000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acacia mearnsii -35.2200000000 149.9300000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acacia mearnsii -32.2700000000 150.4100000000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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. Make sure the lakes layer is loaded.
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 action type,

6. Enter a name for the action, for example Google Search.

7. Additionally you can add a Short Name or even an Icon.

8. Choose the action Scope. See Defining Actions for further information. Leave the default settings for this example.

9. For the action, we need to provide the name of the external program to run. In this case, we can use Firefox. If the program is not in your path, you need to provide the full path.

10. Following the name of the external application, add the URL used for doing a Google search, up to but not including the search term: https://www.google.com/search?q=

11. The text in the Action field should now look like this: https://www.google.com/search?q=

12. 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.

13. From the drop-down box, select ‘NAMES’ and click Insert.

14. Your action text now looks like this:

https://www.google.com/search?q=[%NAMES%]

15. To finalize and add the action, click the OK button.

Obr. 14.57: Edit action dialog configured with the example

This completes the action, and it is ready to use. The final text of the action should look like this:

https://www.google.com/search?q=[%NAMES%]
We can now use the action. 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:

![Image of湖泊和湖泊属性](image)

Obr. 14.58: Select feature and choose action

When we click on the action, it brings up Firefox 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:

```python
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:

```python
command = "firefox"
imagerelpath = "images/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):

```python
qgis.utils.iface.addVectorLayer('/yourpath/[filename].shp', '[layername]', 'ogr')

To add a raster (a TIF image in this example), it becomes:

```python
qgis.utils.iface.addRasterLayer('/yourpath/[filename].tif', '[layername]')
```

### 14.1.12 Display Properties

The Display tab helps you configure fields to use for feature identification:

- **The Display name**: based on a field or an expression. This is:
  - 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 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 no HTML Map Tip is set.

- The HTML Map Tip is specifically created for the map tips: it’s a more complex and full HTML text mixing fields, expressions and html tags (multiline, fonts, images, hyperlink …).

To activate map tips, select the menu option View → Show Map Tips or click on the Show Map Tips icon of the Attributes Toolbar. Map tip is a cross-session feature meaning that once activated, it stays on and apply to any layer in any project, even in future QGIS sessions until it’s toggled off.
Obr. 14.59: HTML code for map tip

Obr. 14.60: Map tip made with HTML code
14.1.13 Rendering Properties

Scale dependent visibility

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 Vykresování v závislosti na měřítku for more information.

Simplify geometry

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).

Obr. 14.61: Layer Geometry Simplification dialog

Poznámka: 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.

While rendering extremely detailed layers (e.g. polygon layers with a huge number of nodes), this 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.

Refresh layer at interval (seconds): set a timer to automatically refresh individual layers at a matching interval. Canvas updates are deferred in order to avoid refreshing multiple times if more than one layer has an auto update interval set.
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.

### 14.1.14 Variables Properties

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.

### 14.1.15 Metadata Properties

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 currently saved in the project file. They can also be saved in a .qmd file alongside file based layers or in a local .sqlite database for remote layers (e.g. PostGIS).

### 14.1.16 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.
14.1.17 Legend Properties

The Legend properties tab provides you with advanced settings for the Layers panel and/or the print layout legend. These options include:

- 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.

![Obr. 14.62: Setting text on symbols (left) and its rendering in the Layers panel (right)](image)

- 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.

14.1.18 QGIS Server Properties

The QGIS Server tab consists of Description, Attribution, MetadataURL, and LegendUrl sections.

You can add or edit a title and abstract for the layer in the Description section. It’s also possible to 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 DataUrl field.

Use Attribution to get attribute data from an XML metadata catalog.

In MetadataUrl, you can define the general path 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 LegendUrl 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.
14.1.19 Digitizing Properties

The Digitizing tab gives access to options that help to ensure the quality of digitized geometries.

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.
Obr. 14.64: The QGIS Digitizing tab in the vector layers properties dialog
Geometry Checks

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.
<table>
<thead>
<tr>
<th>Topology check option</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gap</strong> check will check for gaps between neighbouring polygons.</td>
<td>![Illustration of a gap check]</td>
</tr>
<tr>
<td><strong>Overlap</strong> check will check for overlaps between neighbouring polygons.</td>
<td>![Illustration of an overlap check]</td>
</tr>
<tr>
<td><strong>Missing vertex</strong> check will check for shared boundaries of neighbouring polygons where one border misses a vertex which is present on the other one.</td>
<td>![Illustration of a missing vertex check]</td>
</tr>
</tbody>
</table>

### 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.
14.2 Výrazy

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 …

14.2.1 The Expression string builder

Main dialog to build expressions, the Expression string builder is available from many parts in QGIS and, can particularly be accessed when:

- clicking the button;
- 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 Data defined override setup);
- building a geometry generator symbol layer;
- doing some geoprocessing.

The Expression builder dialog offers access to the:

- 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.

Some use cases of expressions:

- From Field Calculator, calculate a „pop_density” field using existing „total_pop” and „area_km2” fields:

  ```
  "total_pop" / "area_km2"
  ```

- Update the field „density_level” with categories according to the „pop_density” values:

  ```
  CASE WHEN "pop_density" < 50 THEN 'Low population density'
  WHEN "pop_density" >= 50 and "pop_density" < 150 THEN 'Medium population density'
  WHEN "pop_density" >= 150 THEN 'High population density'
END
  ```

- Update a region layer field with the names (comma separated) of contained airports:

  ```
  aggregate('airport_layer', 'concatenate', "name", within($geometry, geometry(@parent)), ',' )
  ```

- Apply a categorized style to all the features according to whether their average house price is smaller or higher than 10000€ per square metre:

  ```
  "price_m2" > 10000
  ```

- Using the „Select By Expression…“ tool, select all the features representing areas of “High population density” and whose average house price is higher than 10000€ per square metre:

  ```
  "density_level" = 'High population density' and "price_m2" > 10000
  ```

Likewise, the previous expression could also be used to define which features should be labeled or shown in the map.
Using expressions offers you a lot of possibilities.

**Tip: Use named parameters to improve the 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 at a later date!

### 14.2.2 List of functions

The *Expression* tab provides the main interface to write expressions using functions, layer’s fields and values. It contains following widgets:

- An expression editor area to type or paste 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.

**Tip: 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.

```qquery
/*
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 ) )
  ),
  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


```

- Under the expression editor, an *Output preview* displays the result of the expression evaluated on the first feature of the layer. In case of error, it indicates it and you can access 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.

**Tip:** 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. Double-clicking a value adds it to the expression editor.

**Tip:** 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.

![Expression Dialog](image)

Obr. 14.65: The Expression tab

### Aggregates Functions

This group contains functions which aggregate values over layers and fields.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregate</td>
<td>Returns an aggregate value calculated using features from another layer</td>
</tr>
<tr>
<td>array_agg</td>
<td>Returns an array of aggregated values from a field or expression</td>
</tr>
<tr>
<td>collect</td>
<td>Returns the multipart geometry of aggregated geometries from an expression</td>
</tr>
<tr>
<td>concatenate</td>
<td>Returns all aggregated strings from a field or expression joined by a delimiter</td>
</tr>
<tr>
<td>concatenate_unique</td>
<td>Returns all unique aggregated strings from a field or expression joined by a delimiter</td>
</tr>
<tr>
<td>count</td>
<td>Returns the count of matching features</td>
</tr>
<tr>
<td>count_distinct</td>
<td>Returns the count of distinct values</td>
</tr>
<tr>
<td>count_missing</td>
<td>Returns the count of missing (null) values</td>
</tr>
<tr>
<td>iqr</td>
<td>Returns the calculated inter quartile range from a field or expression</td>
</tr>
<tr>
<td>majority</td>
<td>Returns the aggregate majority of values (most commonly occurring value) from a field or expression</td>
</tr>
<tr>
<td>max_length</td>
<td>Returns the maximum length of strings from a field or expression</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Function Popis

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximum</td>
<td>Returns the aggregate maximum value from a field or expression</td>
</tr>
<tr>
<td>mean</td>
<td>Returns the aggregate mean value from a field or expression</td>
</tr>
<tr>
<td>median</td>
<td>Returns the aggregate median value from a field or expression</td>
</tr>
<tr>
<td>min_length</td>
<td>Returns the minimum length of strings from a field or expression</td>
</tr>
<tr>
<td>minimum</td>
<td>Returns the aggregate minimum value from a field or expression</td>
</tr>
<tr>
<td>minority</td>
<td>Returns the aggregate minority of values (least commonly occurring value) from a field or expression</td>
</tr>
<tr>
<td>q1</td>
<td>Returns the calculated first quartile from a field or expression</td>
</tr>
<tr>
<td>q3</td>
<td>Returns the calculated third quartile from a field or expression</td>
</tr>
<tr>
<td>range</td>
<td>Returns the aggregate range of values (maximum - minimum) from a field or expression</td>
</tr>
<tr>
<td>relation_aggregate</td>
<td>Returns an aggregate value calculated using all matching child features from a layer relation</td>
</tr>
<tr>
<td>stdev</td>
<td>Returns the aggregate standard deviation value from a field or expression</td>
</tr>
<tr>
<td>sum</td>
<td>Returns the aggregate summed value from a field or expression</td>
</tr>
</tbody>
</table>

### Examples:

- Return the maximum of the „passengers“ field from features in the layer grouped by „station_class“ field:

  ```
  maximum("passengers", group_by:="station_class")
  ```

- Calculate the total number of passengers for the stations inside the current atlas feature:

  ```
  aggregate('rail_stations','sum','passengers',
            intersects(@atlas_geometry, $geometry))
  ```

- Return the mean of the „field_from_related_table“ field for all matching child features using the „my_relation“ relation from the layer:

  ```
  relation_aggregate('my_relation', 'mean', "field_from_related_table")
  ```

  or:

  ```
  relation_aggregate(relation="my_relation", aggregate := 'mean',
                    expression := "field_from_related_table")
  ```

### 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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Returns an array containing all the values passed as parameter</td>
</tr>
<tr>
<td>array_all</td>
<td>Returns true if an array contains all the values of a given array</td>
</tr>
<tr>
<td>array_append</td>
<td>Returns an array with the given value added at the end</td>
</tr>
<tr>
<td>array_cat</td>
<td>Returns an array containing all the given arrays concatenated</td>
</tr>
<tr>
<td>array_contains</td>
<td>Returns true if an array contains the given value</td>
</tr>
<tr>
<td>array_distinct</td>
<td>Returns an array containing distinct values of the given array</td>
</tr>
<tr>
<td>array_filter</td>
<td>Returns an array with only the items for which an expression evaluates to true</td>
</tr>
<tr>
<td>array_find</td>
<td>Returns the index (0 for the first one) of a value within an array. Returns -1 if the value is not found.</td>
</tr>
<tr>
<td>array_first</td>
<td>Returns the first value of an array</td>
</tr>
<tr>
<td>array_foreach</td>
<td>Returns an array with the given expression evaluated on each item</td>
</tr>
</tbody>
</table>
Tabulka 14.2 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>array_get</td>
<td>Returns the Nth value (0 for the first one) of an array</td>
</tr>
<tr>
<td>array_insert</td>
<td>Returns an array with the given value added at the given position</td>
</tr>
<tr>
<td>array_intersect</td>
<td>Returns true if any element of array_1 exists in array_2</td>
</tr>
<tr>
<td>array_last</td>
<td>Returns the last element of an array</td>
</tr>
<tr>
<td>array_length</td>
<td>Returns the number of elements of an array</td>
</tr>
<tr>
<td>array_prepend</td>
<td>Returns an array with the given value added at the beginning</td>
</tr>
<tr>
<td>array_remove_all</td>
<td>Returns an array with all the entries of the given value removed</td>
</tr>
<tr>
<td>array_remove_at</td>
<td>Returns an array with the given index removed</td>
</tr>
<tr>
<td>array_reverse</td>
<td>Returns the given array with array values in reversed order</td>
</tr>
<tr>
<td>array_slice</td>
<td>Returns the values of the array from the start_pos argument up to and including the end_pos argument</td>
</tr>
<tr>
<td>array_sort</td>
<td>Returns the provided array with its elements sorted</td>
</tr>
<tr>
<td>array_to_string</td>
<td>Concatenates array elements into a string separated by a delimiter and using optional string for empty values</td>
</tr>
<tr>
<td>generate_series</td>
<td>Creates an array containing a sequence of numbers</td>
</tr>
<tr>
<td>regexp_matches</td>
<td>Returns an array of all strings captured by capturing groups, in the order the groups themselves appear in the supplied regular expression against a string</td>
</tr>
<tr>
<td>string_to_array</td>
<td>Splits string into an array using supplied delimiter and optional string for empty values</td>
</tr>
</tbody>
</table>

**Funkce barev**

Tato skupina obsahuje funkce pro manipulaci s barvami.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>color_cmyk</td>
<td>Returns a string representation of a color based on its cyan, magenta, yellow and black components</td>
</tr>
<tr>
<td>color_cmyka</td>
<td>Returns a string representation of a color based on its cyan, magenta, yellow, black and alpha (transparency) components</td>
</tr>
<tr>
<td>color_grayscale_average</td>
<td>Applies a grayscale filter and returns a string representation from a provided color</td>
</tr>
<tr>
<td>color_hsl</td>
<td>Returns a string representation of a color based on its hue, saturation, and lightness attributes</td>
</tr>
<tr>
<td>color_hsla</td>
<td>Returns a string representation of a color based on its hue, saturation, lightness and alpha (transparency) attributes</td>
</tr>
<tr>
<td>color_hsv</td>
<td>Returns a string representation of a color based on its hue, saturation, and value attributes</td>
</tr>
<tr>
<td>color_hsva</td>
<td>Returns a string representation of a color based on its hue, saturation, value and alpha (transparency) attributes</td>
</tr>
<tr>
<td>color_mix_rgb</td>
<td>Returns a string representing a color mixing the red, green, blue, and alpha values of two provided colors based on a given ratio</td>
</tr>
<tr>
<td>color_part</td>
<td>Returns a specific component from a color string, eg the red component or alpha component</td>
</tr>
<tr>
<td>color_rgb</td>
<td>Returns a string representation of a color based on its red, green, and blue components</td>
</tr>
<tr>
<td>color_rgba</td>
<td>Returns a string representation of a color based on its red, green, blue, and alpha (transparency) components</td>
</tr>
<tr>
<td>create_ramp</td>
<td>Returns a gradient ramp from a map of color strings and steps</td>
</tr>
<tr>
<td>darker</td>
<td>Returns a darker (or lighter) color string</td>
</tr>
<tr>
<td>lighter</td>
<td>Returns a lighter (or darker) color string</td>
</tr>
<tr>
<td>project_color</td>
<td>Returns a color from the project’s color scheme</td>
</tr>
<tr>
<td>ramp_color</td>
<td>Returns a string representing a color from a color ramp</td>
</tr>
<tr>
<td>set_color_part</td>
<td>Sets a specific color component for a color string, eg the red component or alpha component</td>
</tr>
</tbody>
</table>
Conditional Functions

Tato skupina obsahuje funkce pro zpracování podmíněné kontroly ve výrazech.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE WHEN … THEN … END</td>
<td>Evaluates an expression and returns a result if true. You can test multiple conditions.</td>
</tr>
<tr>
<td>CASE WHEN … THEN … ELSE … END</td>
<td>Evaluates an expression and returns a different result whether it’s true or false. You can test multiple conditions.</td>
</tr>
<tr>
<td>coalesce</td>
<td>Returns the first non-NULL value from the expression list.</td>
</tr>
<tr>
<td>if</td>
<td>Tests a condition and returns a different result depending on the conditional check.</td>
</tr>
<tr>
<td>nullif(value1, value2)</td>
<td>Returns a null value if value1 equals value2 otherwise it returns value1. This can be used to conditionally substitute values with NULL.</td>
</tr>
<tr>
<td>try</td>
<td>Tries an expression and returns its value if error-free, an alternative value (if provided) or Null if an error occurs.</td>
</tr>
</tbody>
</table>

Nějaký příklad:

- Vrátí hodnotu v případě, že první podmínka je pravda, jinak jiná hodnota:

  ```sql
  CASE WHEN "software" LIKE '%QGIS%' THEN 'QGIS' ELSE 'Other' END
  ```

Conversions Functions

Tato skupina obsahuje funkce pro převod jednoho datového typu na jiný (např. řetězec na celé číslo, celé číslo na řetězec).

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_date</td>
<td>Converts a string into a date object.</td>
</tr>
<tr>
<td>to_datetime</td>
<td>Converts a string into a datetime object.</td>
</tr>
<tr>
<td>to_dm</td>
<td>Converts a coordinate to degree, minute.</td>
</tr>
<tr>
<td>to_dms</td>
<td>Converts coordinate to degree, minute, second.</td>
</tr>
<tr>
<td>to_int</td>
<td>Converts a string to integer number.</td>
</tr>
<tr>
<td>to_interval</td>
<td>Converts a string to an interval type (can be used to take days, hours, months, etc. of a date).</td>
</tr>
<tr>
<td>to_real</td>
<td>Converts a string to a real number.</td>
</tr>
<tr>
<td>to_string</td>
<td>Converts number to string.</td>
</tr>
<tr>
<td>to_time</td>
<td>Converts a string into a time object.</td>
</tr>
</tbody>
</table>

Custom Functions

This group contains functions created by the user. See Function Editor for more details.

Funkce dat a časů

Tato skupina obsahuje funkce pro manipulaci s datumovými a časovými daty.
### Function Popis

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>Returns as an interval the difference between two dates or datetimes</td>
</tr>
<tr>
<td>day</td>
<td>Extracts the day from a date or datetime, or the number of days from an interval</td>
</tr>
<tr>
<td>day_of_week</td>
<td>Returns a number corresponding to the day of the week for a specified date or datetime</td>
</tr>
<tr>
<td>epoch</td>
<td>Returns the interval in milliseconds between the unix epoch and a given date value</td>
</tr>
<tr>
<td>hour</td>
<td>Extracts the hour from a date time or time, or the number of hours from an interval</td>
</tr>
<tr>
<td>minute</td>
<td>Extracts the minute from a date time or time, or the number of minutes from an interval</td>
</tr>
<tr>
<td>month</td>
<td>Extracts the month part from a date or datetime, or the number of months from an interval</td>
</tr>
<tr>
<td>now</td>
<td>Returns current date and time</td>
</tr>
<tr>
<td>second</td>
<td>Extracts the second from a date time or time, or the number of seconds from an interval</td>
</tr>
<tr>
<td>week</td>
<td>Extracts the week number from a date or datetime, or the number of weeks from an interval</td>
</tr>
<tr>
<td>year</td>
<td>Extracts the year part from a date or datetime, or the number of years from an interval</td>
</tr>
</tbody>
</table>

This group also shares several functions with the *Conversions Functions* (to_date, to_time, to_datetime, to_interval) and *Funkce řetězců* (format_date) groups.

### České příklady:

- Get today’s month and year in the “month_number/year” format:
  ```sql
  format_date(now(),'MM/yyyy')
  -- Returns '03/2017'
  ```

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:
  ```sql
  to_date('2017-09-29') - to_date(now())
  -- Returns <interval: 203 days>
  ```

- The same with time:
  ```sql
  to_datetime('2017-09-29 12:00:00') - to_datetime(now())
  -- Returns <interval: 202.49 days>
  ```

- Get the datetime of 100 days from now:
  ```sql
  now() + to_interval('100 days')
  -- Returns <datetime: 2017-06-18 01:00:00>
  ```

### Poznámka: Storing date and datetime and intervals on fields

The ability to store `date`, `time` and `datetime` values directly on fields may depend 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 stored in text type fields after using the `to_format()` 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)
Pole a hodnoty

Contains a list of fields from the layer.

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.

Files and Paths Functions

This group contains functions which manipulate file and path names.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>base_file_name</td>
<td>Returns the base name of the file without the directory or file suffix.</td>
</tr>
<tr>
<td>file_exists</td>
<td>Returns true if a file path exists.</td>
</tr>
<tr>
<td>file_name</td>
<td>Returns the name of a file (including the file extension), excluding the directory.</td>
</tr>
<tr>
<td>file_path</td>
<td>Returns the directory component of a file path, without the file name</td>
</tr>
<tr>
<td>file_size</td>
<td>Returns the size (in bytes) of a file.</td>
</tr>
<tr>
<td>file_suffix</td>
<td>Returns the file extension from a file path.</td>
</tr>
<tr>
<td>is_directory</td>
<td>Returns true if a path corresponds to a directory.</td>
</tr>
<tr>
<td>is_file</td>
<td>Returns true if a path corresponds to a file.</td>
</tr>
</tbody>
</table>

Fuzzy Matching Functions

This group contains functions for fuzzy comparisons between values.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>hamming_distance</td>
<td>Returns the number of characters at corresponding positions within the input strings where the characters are different</td>
</tr>
<tr>
<td>levenshein</td>
<td>Returns the minimum number of character edits (insertions, deletions or substitutions) required to change one string to another. Measure the similarity between two strings</td>
</tr>
<tr>
<td>longest_common_substring</td>
<td>Returns the longest common substring between two strings</td>
</tr>
<tr>
<td>soundex</td>
<td>Returns the Soundex representation of a string</td>
</tr>
</tbody>
</table>

General Functions

This group contains general assorted functions.
Function | Popis
---|---
env | Gets an environment variable and returns its content as a string. If the variable is not found, NULL will be returned.
eval | Evaluates an expression which is passed in a string. Useful to expand dynamic parameters passed as context variables or fields.
is_layer_visible | Returns true if a specified layer is visible
layer_property | Returns a property of a layer or a value of its metadata. It can be layer name, crs, geometry type, feature count…
var | Returns the value stored within a specified variable. See Variables below
with_variable | Creates and sets a variable for any expression code that will be provided as a third argument. Useful to avoid repetition in expressions where the same value needs to be used more than once.

### Funkce geometrie

This group contains functions that operate on geometry objects (e.g. buffer, transform, $\text{area}$).

Function | Popis
---|---
$\text{area}$ | Returns the area size of the current feature
Sgeometry | Returns the geometry of the current feature (can be used for processing with other functions)
Slength | Returns the length of the current line feature
Spolygon | Returns the perimeter of the current polygon feature
Sx | Returns the X coordinate of the current feature
Sx\_at(n) | Returns the X coordinate of the nth node of the current feature’s geometry
Sy | Returns the Y coordinate of the current feature
Sy\_at(n) | Returns the Y coordinate of the nth node of the current feature’s geometry
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
area | Returns the area of a geometry polygon feature. Calculations are in the Spatial Reference System of this geometry
azimuth | Returns the north-based azimuth as the angle in radians measured clockwise from the vertical on point\_a to point\_b
boundary | Returns the closure of the combinatorial boundary of the geometry (ie the topological boundary of the geometry - see also Boundary).
bounds | Returns a geometry which represents the bounding box of an input geometry. Calculations are in the Spatial Reference System of this geometry (see also Bounding boxes)
bounds\_height | Returns the height of the bounding box of a geometry. Calculations are in the Spatial Reference System of this geometry
bounds\_width | Returns the width of the bounding box of a geometry. Calculations are in the Spatial Reference System of this geometry
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 (see also Buffer)
buffer\_by\_m | Creates a buffer along a line geometry where the buffer diameter varies according to the M values at the line vertices (see also Variable width buffer (by M value))
centroid | Returns the geometric center of a geometry (see also Centroids)
closest\_point | Returns the point on a geometry that is closest to a second geometry
collect\_geometries | Collects a set of geometries into a multi-part geometry object (see also Collect geometries)
combine | Returns the combination of two geometries
contains(a,b) | Returns 1 (true) if and only if no points of b lie in the exterior of a, and at least one point of the interior of b lies in the interior of a
<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>convex_hull</td>
<td>Returns the convex hull of a geometry (this represents the minimum convex geometry that encloses all geometries within the set) (see also Convex hull)</td>
</tr>
<tr>
<td>crosses</td>
<td>Returns 1 (true) if the supplied geometries have some, but not all, interior points in common</td>
</tr>
<tr>
<td>difference(a,b)</td>
<td>Returns a geometry that represents that part of geometry a that does not intersect with geometry b (see also Difference)</td>
</tr>
<tr>
<td>disjoint</td>
<td>Returns 1 (true) if the geometries do not share any space together</td>
</tr>
<tr>
<td>distance</td>
<td>Returns the minimum distance (based on Spatial Reference System) between two geometries in projected units</td>
</tr>
<tr>
<td>distance_to_vertex</td>
<td>Returns the distance along the geometry to a specified vertex</td>
</tr>
<tr>
<td>end_point</td>
<td>Returns the last node from a geometry (see also Extract specific vertices)</td>
</tr>
<tr>
<td>extend</td>
<td>Extends the start and end of a linestring geometry by a specified amount (see also Extend lines)</td>
</tr>
<tr>
<td>exterior_ring</td>
<td>Returns a line string representing the exterior ring of a polygon geometry, or null if the geometry is not a polygon</td>
</tr>
<tr>
<td>extrude(geom,x,y)</td>
<td>Returns an extruded version of the input (Multi-) Curve or (Multi-) Linestring geometry with an extension specified by X and Y</td>
</tr>
<tr>
<td>flip_coordinates</td>
<td>Returns a copy of the geometry with the X and Y coordinates swapped (see also Swap X and Y coordinates)</td>
</tr>
<tr>
<td>force_rhr</td>
<td>Forcing a geometry to respect the Right-Hand-Rule (see also Force right-hand-rule)</td>
</tr>
<tr>
<td>geom_from_gml</td>
<td>Returns a geometry created from a GML representation of geometry</td>
</tr>
<tr>
<td>geom_from_wkt</td>
<td>Returns a geometry created from a well-known text (WKT) representation</td>
</tr>
<tr>
<td>geom_to_wkt</td>
<td>Returns the well-known text (WKT) representation of the geometry without SRID metadata</td>
</tr>
<tr>
<td>geometry</td>
<td>Returns a feature’s geometry</td>
</tr>
<tr>
<td>geometry_n</td>
<td>Returns the nth geometry from a geometry collection, or null if the input geometry is not a collection</td>
</tr>
<tr>
<td>haussdorff_distance</td>
<td>Returns basically a measure of how similar or dissimilar two geometries are, with a lower distance indicating more similar geometries</td>
</tr>
<tr>
<td>inclination</td>
<td>Returns the inclination measured from the zenith (0) to the nadir (180) on point_a to point_b</td>
</tr>
<tr>
<td>interior_ring_n</td>
<td>Returns the geometry of the nth interior ring from a polygon geometry, or null if the geometry is not a polygon</td>
</tr>
<tr>
<td>intersection</td>
<td>Returns a geometry that represents the shared portion of two geometries (see also Intersection)</td>
</tr>
<tr>
<td>intersects</td>
<td>Tests whether a geometry intersects another. Returns 1 (true) if the geometries spatially intersect (share any portion of space) and 0 if they don't</td>
</tr>
<tr>
<td>intersects_bbox</td>
<td>Tests whether a geometry’s bounding box overlaps another geometry’s bounding box. Returns 1 (true) if the geometries spatially intersect (share any portion of space) their bounding box, or 0 if they don't</td>
</tr>
<tr>
<td>is_closed</td>
<td>Returns true if a line string is closed (start and end points are coincident), false if a line string is not closed, or null if the geometry is not a line string</td>
</tr>
<tr>
<td>length</td>
<td>Returns length of a line geometry feature (or length of a string)</td>
</tr>
<tr>
<td>line_interpolate_angle</td>
<td>Returns the angle parallel to the geometry at a specified distance along a linestring geometry. Angles are in degrees clockwise from north.</td>
</tr>
<tr>
<td>line_interpolate_point</td>
<td>Returns the point interpolated by a specified distance along a linestring geometry. (see also Interpolate point on line)</td>
</tr>
<tr>
<td>line_locate_point</td>
<td>Returns the distance along a linestring corresponding to the closest position the linestring comes to a specified point geometry.</td>
</tr>
<tr>
<td>line_merge</td>
<td>Returns a (Multi-) LineString geometry, where any connected LineStrings from the input geometry have been merged into a single linestring.</td>
</tr>
<tr>
<td>line_substring</td>
<td>Returns the portion of a line or curve geometry falling between specified start and end distances (measured from the beginning of the line) (see also Line substring)</td>
</tr>
<tr>
<td>m</td>
<td>Returns the M value of a point geometry</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>make_circle</td>
<td>Vytvoří kružnou geometrii z centrálního bodu a poloměru</td>
</tr>
<tr>
<td>make_ellipse</td>
<td>Vytvoří eliptickou geometrii z centrálního bodu, osi a azimuth</td>
</tr>
<tr>
<td>make_line</td>
<td>Vytvoří liniovou geometrii z řetězku nebo polem bodů geometrie</td>
</tr>
<tr>
<td>make_point(x,y,z,m)</td>
<td>Vratí polohovou geometrii z X a Y (a možného Z a M) hodnot</td>
</tr>
<tr>
<td>make_point_m(x,y,m)</td>
<td>Vratí polohovou geometrii z X a Y a M hodnoty</td>
</tr>
<tr>
<td>make_polygon</td>
<td>Vytvoří polygonovou geometrii z vnějšího pružkování a polem vnějších</td>
</tr>
<tr>
<td>make_rectangle_3points</td>
<td>Vytvoří trojúhelník z 3 bodů</td>
</tr>
<tr>
<td>make_regular_polygon</td>
<td>Vytvoří regulární polygon</td>
</tr>
<tr>
<td>make_square</td>
<td>Vytvoří čtverec z diagonálu</td>
</tr>
<tr>
<td>make_triangle</td>
<td>Vytvoří trojúhelník polygon</td>
</tr>
<tr>
<td>minimal_circle</td>
<td>Vratí minimální obdélník vstupní geometrie (viz také Minimum enclosing circles)</td>
</tr>
<tr>
<td>nodes_to_points</td>
<td>Vratí multiplozní geometrii z každého výběru v vstupní geometrii</td>
</tr>
<tr>
<td>num_geometries</td>
<td>Vratí počet geometrií v geometrické kolekci, nebo null, pokud vstup</td>
</tr>
<tr>
<td>num_interior_rings</td>
<td>Vratí počet vnějších pružkování v polygonu a geometrické kolekci, nebo</td>
</tr>
<tr>
<td>num_points</td>
<td>Vratí počet vertexů v geometrii</td>
</tr>
<tr>
<td>num_rings</td>
<td>Vratí počet pružkování v polygonu a geometrické kolekci, nebo null,</td>
</tr>
<tr>
<td>offset_curve</td>
<td>Vratí geometrii vytvořené zaříděním liniové geometrie k řetězku. Důležité jsou v určitého prostoru v této geometrii (viz také Offset lines)</td>
</tr>
<tr>
<td>order_parts</td>
<td>Položí z aža Geometry a v kritérií deaktivuji GeoMetry</td>
</tr>
<tr>
<td>oriented_bbox</td>
<td>Vratí geometrii značenou za obdélníky vstupní geometrie (viz také Oriented minimum bounding box)</td>
</tr>
<tr>
<td>overlaps</td>
<td>Vratí obdélníku obdélníku a geometrické kolekci, pokud geometrie jsou</td>
</tr>
<tr>
<td>perimeter</td>
<td>Vratí perimetr geometrií polygonu kompozicí. Výpočet je v určitého</td>
</tr>
<tr>
<td>point_n</td>
<td>Vratí konkrétní bod v geometrii (viz také Extract specific vertices)</td>
</tr>
<tr>
<td>point_on_surface</td>
<td>Vratí konkrétní bod založený na povrchu geometrii (viz také Point on Surface)</td>
</tr>
<tr>
<td>pole_of_inaccessibility</td>
<td>Vratí přibližně pole inaccesibilnosti na povrchu, je nejdominantní</td>
</tr>
<tr>
<td>project</td>
<td>Vratí bod projekovaný z startového bodu za vzdálenost, azimuth a</td>
</tr>
<tr>
<td>relate</td>
<td>Vratí vztah nebo vzniknuv Dimensional Extended 9 Intersection Model (DE-9IM) representation of the relationship between two geometries</td>
</tr>
<tr>
<td>reverse</td>
<td>Vratí vzhled liniové geometrie po vzhledu jejích vertexů (viz také</td>
</tr>
<tr>
<td>segments_to_lines</td>
<td>Vratí multi liniovou geometrii z řetězku vstupní geometrii (viz také</td>
</tr>
<tr>
<td>shortest_line</td>
<td>Vratí nejkratší liniovou geometrii dvou geometrik a geometrii. Řetězec binarizuje se startem geometrie 1 a koncem geometrie 2</td>
</tr>
<tr>
<td>simplify</td>
<td>Vratí geometrii po eliminaci bodů s určitou vzdáleností (viz také</td>
</tr>
<tr>
<td>simplify_vw</td>
<td>Vratí geometrii po eliminaci bodů s určitém povrchem (viz také</td>
</tr>
<tr>
<td>single_sided_buffer</td>
<td>Vratí geometrii vytvořené zaříděním jednoho liniového geometrii.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>smooth</td>
<td>Smoothes a geometry by adding extra nodes which round off corners in the geometry (see also Smooth)</td>
</tr>
<tr>
<td>start_point</td>
<td>Returns the first node from a geometry (see also Extract specific vertices)</td>
</tr>
<tr>
<td>sym_difference</td>
<td>Returns a geometry that represents the portions of two geometries that do not intersect (see also Symmetrical difference)</td>
</tr>
<tr>
<td>tapered_buffer</td>
<td>Creates a buffer along a line geometry where the buffer diameter varies evenly over the length of the line (see also Tapered buffers)</td>
</tr>
<tr>
<td>touches</td>
<td>Tests whether a geometry touches another. Returns 1 (true) if the geometries have at least one point in common, but their interiors do not intersect</td>
</tr>
<tr>
<td>transform</td>
<td>Returns the geometry transformed from the source CRS to the destination CRS (see also Reproject layer)</td>
</tr>
<tr>
<td>translate</td>
<td>Returns a translated version of a geometry. Calculations are in the Spatial Reference System of the geometry (see also Translate)</td>
</tr>
<tr>
<td>union</td>
<td>Returns a geometry that represents the point set union of the geometries</td>
</tr>
<tr>
<td>wedge_buffer</td>
<td>Returns a wedge shaped buffer originating from a point geometry given an angle and radii (see also Create wedge buffers)</td>
</tr>
<tr>
<td>within (a,b)</td>
<td>Tests whether a geometry is within another. Returns 1 (true) if geometry a is completely inside geometry b</td>
</tr>
<tr>
<td>x</td>
<td>Returns the X coordinate of a point geometry, or the X coordinate of the centroid for a non-point geometry</td>
</tr>
<tr>
<td>x_max</td>
<td>Returns the maximum X coordinate of a geometry. Calculations are in the Spatial Reference System of this geometry</td>
</tr>
<tr>
<td>x_min</td>
<td>Returns the minimum X coordinate of a geometry. Calculations are in the Spatial Reference System of this geometry</td>
</tr>
<tr>
<td>y</td>
<td>Returns the Y coordinate of a point geometry, or the Y coordinate of the centroid for a non-point geometry</td>
</tr>
<tr>
<td>y_max</td>
<td>Returns the maximum Y coordinate of a geometry. Calculations are in the Spatial Reference System of this geometry</td>
</tr>
<tr>
<td>y_min</td>
<td>Returns the minimum Y coordinate of a geometry. Calculations are in the Spatial Reference System of this geometry</td>
</tr>
<tr>
<td>z</td>
<td>Returns the Z coordinate of a point geometry</td>
</tr>
</tbody>
</table>

Některé příklady:

- Return the X coordinate of the current feature’s centroid:
  
  \[
  \text{x( $geometry )}
  \]

- Send back a value according to feature’s area:
  
  \[
  \text{CASE WHEN $area > 10 000 THEN 'Larger' ELSE 'Smaller' END}
  \]

- You can manipulate the current geometry using the variable $geometry to create a buffer or get a point on the geometry’s surface:
  
  \[
  \text{buffer($geometry, 10 )}
  \]
  \[
  \text{point_on_surface($geometry )}
  \]

- Given a point feature, generate a closed line (using make_line) around the point’s geometry:
  
  \[
  \text{make_line(}
  \text{    -- using an array of points placed around the original}
  \text{    array_foreach(}
  \text{        -- list of angles for placing the projected points (every 90°)}
  \text{        array:=generate_series( 0, 360, 90 ),}
  \text{        -- translate the point 20 units in the given direction (angle)}
  \text{        expression:=project( $geometry, distance:=20, azimuth:=radians( @element ) },
  \text{    )}
  \]

(continues on next page)
Layout Functions

This group contains functions to manipulate print layout items properties.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_variables</td>
<td>Returns a map of variables from a layout item inside this print layout</td>
</tr>
</tbody>
</table>

An example:

- Get the scale of the 'Map 0' in the current print layout:

```sql
map_get( item_variables('Map 0'), 'map_scale' )
```

Map Layers

This group contains a list of the available layers in the current project. This offers a convenient way to write expressions referring to multiple layers, such as when performing aggregates, attribute or spatial queries.

It also provides some convenient functions to manipulate layers.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>deco-de_uri</td>
<td>Takes a layer and decodes the uri of the underlying data provider. Available information depends on the data provider type.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_json</td>
<td>Loads a json-formatted string</td>
</tr>
<tr>
<td>json_to_map</td>
<td>Creates a map from a hstore-formatted string</td>
</tr>
<tr>
<td>map</td>
<td>Returns a map containing all the keys and values passed as pair of parameters</td>
</tr>
<tr>
<td>map_akeys</td>
<td>Returns all the keys of a map as an array</td>
</tr>
<tr>
<td>map_avals</td>
<td>Returns all the values of a map as an array</td>
</tr>
<tr>
<td>map_concat</td>
<td>Returns a map containing all the entries of the given maps. If two maps contain the same key, the value of the second map is taken.</td>
</tr>
<tr>
<td>map_delete</td>
<td>Returns a map with the given key and its corresponding value deleted</td>
</tr>
<tr>
<td>map_exist</td>
<td>Returns true if the given key exists in the map</td>
</tr>
<tr>
<td>map_get</td>
<td>Returns the value of a map, given it's key</td>
</tr>
<tr>
<td>map_insert</td>
<td>Returns a map with an added key/value</td>
</tr>
<tr>
<td>map_to_hstore</td>
<td>Merges map elements into a hstore-formatted string</td>
</tr>
<tr>
<td>map_to_json</td>
<td>Merges map elements into a json-formatted string</td>
</tr>
<tr>
<td>to_json</td>
<td>Creates a json-formatted string from a map, an array or other value</td>
</tr>
</tbody>
</table>
Matematické funkce

Tato skupina obsahuje matematické funkce (např. odmocnina, sin a cos).

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>Returns the absolute value of a number</td>
</tr>
<tr>
<td>acos</td>
<td>Returns the inverse cosine of a value in radians</td>
</tr>
<tr>
<td>asin</td>
<td>Returns the inverse sine of a value in radians</td>
</tr>
<tr>
<td>atan</td>
<td>Returns the inverse tangent of a value in radians</td>
</tr>
<tr>
<td>atan2(y,x)</td>
<td>Returns the inverse tangent of Y/X by using the signs of the two arguments to determine the quadrant of the result</td>
</tr>
<tr>
<td>azimuth(a,b)</td>
<td>Returns the north-based azimuth as the angle in radians measured clockwise from the vertical on point a to point b</td>
</tr>
<tr>
<td>ceil</td>
<td>Rounds a number upwards</td>
</tr>
<tr>
<td>clamp</td>
<td>Restricts an input value to a specified range</td>
</tr>
<tr>
<td>cos</td>
<td>Returns the cosine of an angle in radians</td>
</tr>
<tr>
<td>degrees</td>
<td>Converts from radians to degrees</td>
</tr>
<tr>
<td>exp</td>
<td>Returns exponential of a value</td>
</tr>
<tr>
<td>floor</td>
<td>Rounds a number downwards</td>
</tr>
<tr>
<td>inclination</td>
<td>Returns the inclination measured from the zenith (0) to the nadir (180) on point_a to point_b.</td>
</tr>
<tr>
<td>ln</td>
<td>Returns the natural logarithm of the passed expression</td>
</tr>
<tr>
<td>log</td>
<td>Returns the value of the logarithm of the passed value and base</td>
</tr>
<tr>
<td>log10</td>
<td>Returns the value of the base 10 logarithm of the passed expression</td>
</tr>
<tr>
<td>max</td>
<td>Returns the largest not null value in a set of values</td>
</tr>
<tr>
<td>min</td>
<td>Returns the smallest not null value in a set of values</td>
</tr>
<tr>
<td>pi</td>
<td>Returns the value of pi for calculations</td>
</tr>
<tr>
<td>radians</td>
<td>Converts from degrees to radians</td>
</tr>
<tr>
<td>rand</td>
<td>Returns the random integer within the range specified by the minimum and maximum argument (inclusive)</td>
</tr>
<tr>
<td>randf</td>
<td>Returns the random float within the range specified by the minimum and maximum argument (inclusive)</td>
</tr>
<tr>
<td>round</td>
<td>Rounds to number of decimal places</td>
</tr>
<tr>
<td>scale_exp</td>
<td>Transforms a given value from an input domain to an output range using an exponential curve</td>
</tr>
<tr>
<td>scale_linear</td>
<td>Transforms a given value from an input domain to an output range using linear interpolation</td>
</tr>
<tr>
<td>sin</td>
<td>Returns the sine of an angle in radians</td>
</tr>
<tr>
<td>sqrt</td>
<td>Returns the square root of a value</td>
</tr>
<tr>
<td>tan</td>
<td>Returns the tangent of an angle in radians</td>
</tr>
</tbody>
</table>

Operátory

This group contains operators (e.g., +, -, *). Note that for most of the mathematical functions below, if one of the inputs is NULL then the result is NULL.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>a + b</td>
<td>Addition of two values (a plus b)</td>
</tr>
<tr>
<td>a - b</td>
<td>Subtraction of two values (a minus b)</td>
</tr>
<tr>
<td>a * b</td>
<td>Multiplication of two values (a multiplied by b)</td>
</tr>
<tr>
<td>a / b</td>
<td>Division of two values (a divided by b)</td>
</tr>
<tr>
<td>a % b</td>
<td>Remainder of division of a by b (e.g., 7 % 2 = 1, or 2 fits into 7 three times with remainder 1)</td>
</tr>
<tr>
<td>a ^ b</td>
<td>Power of two values (for example, 2^2=4 or 2^3=8)</td>
</tr>
<tr>
<td>a &lt; b</td>
<td>Compares two values and evaluates to 1 if the left value is less than the right value (a is smaller than b)</td>
</tr>
</tbody>
</table>
### Funkce a Popis

<table>
<thead>
<tr>
<th>Funkce</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a &lt;= b</code></td>
<td>Porovná dva hodnoty a vrací 1, pokud je levá hodnota menší nebo rovna pravé hodnotě.</td>
</tr>
<tr>
<td><code>a &lt;&gt; b</code></td>
<td>Porovná dva hodnoty a vrací 1, pokud jsou nerozlišené.</td>
</tr>
<tr>
<td><code>a = b</code></td>
<td>Porovná dva hodnoty a vrací 1, pokud jsou rovny.</td>
</tr>
<tr>
<td><code>a != b</code></td>
<td>Hodnoty <code>a</code> a <code>b</code> jsou nerozlišené.</td>
</tr>
<tr>
<td><code>a &gt; b</code></td>
<td>Porovná dva hodnoty a vrací 1, pokud je levá hodnota větší než pravá hodnota (a je větší než b).</td>
</tr>
<tr>
<td><code>a &gt;= b</code></td>
<td>Porovná dva hodnoty a vrací 1, pokud je levá hodnota větší nebo rovna pravé hodnotě.</td>
</tr>
<tr>
<td><code>a ~ b</code></td>
<td>Hodnota <code>a</code> se zhodnuje s regulární expresí <code>b</code>.</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td><code>'\n'</code></td>
<td>Vloží novou řádku do řetězce.</td>
</tr>
<tr>
<td><code>LIKE</code></td>
<td>Vrací 1, pokud první parametr se shoduje s poskytnutým vzorcem.</td>
</tr>
<tr>
<td><code>ILIKE</code></td>
<td>Vrací 1, pokud první parametr se shoduje s poskytnutým vzorcem bez ohledu na velikost písmen (ILIKE se může použít místo LIKE, abys shodoval vzorek bez ohledu na velikost písmen).</td>
</tr>
<tr>
<td><code>a IS b</code></td>
<td>Skontrolovává, zda dva hodnoty jsou shodné. Vrací 1, pokud <code>a</code> je shodné se <code>b</code>.</td>
</tr>
<tr>
<td><code>a OR b</code></td>
<td>Vrací 1, pokud některý z podmínek <code>a</code> nebo <code>b</code> je pravda.</td>
</tr>
<tr>
<td><code>a AND b</code></td>
<td>Vrací 1, pokud obě podmínky <code>a</code> a <code>b</code> jsou pravda.</td>
</tr>
<tr>
<td><code>NOT</code></td>
<td>Negativní hodnota.</td>
</tr>
<tr>
<td>„Column_name“</td>
<td>hodnota pole <code>Column_name</code>, pozornost se neztrácí na štědré značení, viděte níže.</td>
</tr>
<tr>
<td>„string“</td>
<td>řetězec, pozornost se neztrácí na štědré značení, viděte níže.</td>
</tr>
<tr>
<td><code>NULL</code></td>
<td>hodnota NULL.</td>
</tr>
<tr>
<td><code>a IS NULL</code></td>
<td>hodnota <code>a</code> je NULL.</td>
</tr>
<tr>
<td><code>a IS NOT NULL</code></td>
<td>hodnota <code>a</code> není NULL.</td>
</tr>
<tr>
<td><code>a IN (value[,value])</code></td>
<td>hodnota <code>a</code> je zvednuta v zadaných hodnotách.</td>
</tr>
<tr>
<td><code>a NOT IN (value[,value])</code></td>
<td>hodnota <code>a</code> není zvednuta v zadaných hodnotách.</td>
</tr>
</tbody>
</table>

### Některé příklady:

- Připojí řetězec a hodnotu z názvu sloupce:

  ```sql
  'My feature''s id is: ' || | "gid"
  ```

- Test, zda „description“ atribut sloupce začíná na „Hello“ řetězci v hodnotě (pozornost se neztrácí na polohu v charakteru `%`):

  ```sql
  "description" LIKE 'Hello\%
  ```
Processing Functions

This group contains functions that operate on processing algorithms.

<table>
<thead>
<tr>
<th>Function</th>
<th>Pops</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>Returns the value of a processing algorithm input parameter</td>
</tr>
</tbody>
</table>

Rasters Functions

This group contains functions to operate on raster layer.

<table>
<thead>
<tr>
<th>Function</th>
<th>Pops</th>
</tr>
</thead>
<tbody>
<tr>
<td>raster_statistic</td>
<td>Returns statistics from a raster layer</td>
</tr>
<tr>
<td>raster_value</td>
<td>Returns the raster band value at the provided point</td>
</tr>
</tbody>
</table>

Record and Attributes Functions

Tato skupina obsahuje funkce, které operují se záznamy identifikátorů.

<table>
<thead>
<tr>
<th>Function</th>
<th>Pops</th>
</tr>
</thead>
<tbody>
<tr>
<td>$currentfeature</td>
<td>Returns the current feature being evaluated. This can be used with the 'attribute' function to evaluate attribute values from the current feature.</td>
</tr>
<tr>
<td>$id</td>
<td>Returns the feature id of the current row</td>
</tr>
<tr>
<td>attribute</td>
<td>Returns the value of a specified attribute from a feature</td>
</tr>
<tr>
<td>attributes</td>
<td>Returns a map of all attributes from a feature, with field names as map keys</td>
</tr>
<tr>
<td>get_feature</td>
<td>Returns the first feature of a layer matching a given attribute value</td>
</tr>
<tr>
<td>get_feature_by_id</td>
<td>Returns the feature of a layer matching the given feature ID</td>
</tr>
<tr>
<td>is_selected</td>
<td>Returns if a feature is selected</td>
</tr>
<tr>
<td>num_selected</td>
<td>Returns the number of selected features on a given layer</td>
</tr>
<tr>
<td>represent_value</td>
<td>Returns the configured representation value for a field value (convenient with some widget types)</td>
</tr>
<tr>
<td>sql_fetch_and_increment</td>
<td>Manage autoincrementing values in SQLite databases</td>
</tr>
<tr>
<td>uuid</td>
<td>Generates a Universally Unique Identifier (UUID) for each row. Each UUID is 38 characters long.</td>
</tr>
</tbody>
</table>

Některé příklady:

- Return the first feature in layer „LayerA“ whose field „id“ has the same value as the field „name“ of the current feature (a kind of jointure):
  ```
  get_feature( 'layerA', 'id', attribute( $currentfeature, 'name') )
  ```

- Calculate the area of the joined feature from the previous example:
  ```
  area( geometry( get_feature( 'layerA', 'id', attribute( $currentfeature, 'name') ) ) )
  ```
Funkce řetězců

Tato skupina obsahuje funkce, které pracují s řetězci (např. které nahrazují, převádí na velká písmena).

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>Vrátí znak spojený s řetězcem v Unicode kódu</td>
</tr>
<tr>
<td>concat</td>
<td>Spojí řetězce do jednoho řetězce</td>
</tr>
<tr>
<td>format</td>
<td>Formátuje řetězec pomocí předaných argumentů</td>
</tr>
<tr>
<td>format_date</td>
<td>Formátuje řetězec jako datum podle určitého formátu</td>
</tr>
<tr>
<td>format_number</td>
<td>Vrátí číslo formátované s desetinnou čárkou a tisíčky</td>
</tr>
<tr>
<td>left</td>
<td>Vrátí n posledních charakterů řetězce</td>
</tr>
<tr>
<td>length</td>
<td>Vrátí délku řetězce nebo délku linie</td>
</tr>
<tr>
<td>lower</td>
<td>Vrátí řetězec ve velkých písmenách</td>
</tr>
<tr>
<td>lpad</td>
<td>Vrátí řetězec doplněný do vyznačené délky nule</td>
</tr>
<tr>
<td>regexp_match</td>
<td>Vrátí první pozici řetězce odpovídající regulární expresii</td>
</tr>
<tr>
<td>regexp_replace</td>
<td>Vrátí řetězec s nahránými regulární expresí</td>
</tr>
<tr>
<td>regexp_substr</td>
<td>Vrátí podřetězek odpovídající regulární expresii</td>
</tr>
<tr>
<td>replace</td>
<td>Vrátí řetězec s převedenými argumenty</td>
</tr>
<tr>
<td>right</td>
<td>Vrátí n posledních charakterů řetězce</td>
</tr>
<tr>
<td>rpad</td>
<td>Vrátí řetězec doplněný do vyznačené délky nule</td>
</tr>
<tr>
<td>strpos</td>
<td>Vrátí první pozici řetězce odpovídající regulární expresii</td>
</tr>
<tr>
<td>substr</td>
<td>Vrátí podřetězek odpovídající regulární expresii</td>
</tr>
<tr>
<td>title</td>
<td>Vrátí řetězec v nápisu s prvním velkým písmenem</td>
</tr>
<tr>
<td>trim</td>
<td>Odstraní prázdné znaky řetězec</td>
</tr>
<tr>
<td>upper</td>
<td>Vrátí řetězec ve velkých písmenách</td>
</tr>
<tr>
<td>wordwrap</td>
<td>Vrátí řetězec podle vyznačeného počtu znaků</td>
</tr>
</tbody>
</table>

About fields concatenation

Můžete spojit řetězce nebo hodnoty pomocí operátorů || nebo + nebo funkce concat, s speciálními charakteristikami:

- Operátor + může být chybový, pokud máte číselný operand, nicmto v rozsahu tisíce.

```
"My feature"'s id is: ' + "gid" -> triggers an error as gid is an integer
```

- Chybový je i týmo, kdy je jedna hodnota NULL, a to || nebo + vynuluje výstup: NULL.

```
"country_name" || NULL -> NULL
concat("My feature"'s id is: ", NULL) -> My feature's id is
concat("firstname", "nickname", "lastname") -> Chuck Norris (if empty nickname)
"firstname" + "nickname" + "lastname" -> NULL (if one field is empty)
```

- Pro ostatní případy, je to případ toho, co se ti vyhodí:

```
"My country is ' + "country_name" + ' (if "country_code" is ')
"My country is ' || "country_name" || ' (if "country_code" || ')
concat("My country is ", "country_name", ' if "country_code", ')
# All the above return: My country is France (FR)
```
Variables

This group contains dynamic variables related to the application, the project file and other settings. The availability of variables depends on the context:

- from the Select by expression dialog
- from the Field calculator dialog
- from the layer properties dialog
- from the print layout

To use these variables in an expression, they should be preceded by the @ character (e.g., @row_number).

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm_id</td>
<td>The unique ID of an algorithm</td>
</tr>
<tr>
<td>atlas_feature</td>
<td>The current atlas feature (as feature object)</td>
</tr>
<tr>
<td>atlas_featureid</td>
<td>The current atlas feature ID</td>
</tr>
<tr>
<td>atlas_featurenumber</td>
<td>The current atlas feature number in the layout</td>
</tr>
<tr>
<td>atlas_filename</td>
<td>The current atlas file name</td>
</tr>
<tr>
<td>atlas_geometry</td>
<td>The current atlas feature geometry</td>
</tr>
<tr>
<td>atlas_layerid</td>
<td>The current atlas coverage layer ID</td>
</tr>
<tr>
<td>atlas_layername</td>
<td>The current atlas coverage layer name</td>
</tr>
<tr>
<td>atlas_pagename</td>
<td>The current atlas page name</td>
</tr>
<tr>
<td>atlas_totalfeatures</td>
<td>The total number of features in atlas</td>
</tr>
<tr>
<td>canvas_cursor_point</td>
<td>The last cursor position on the canvas in the project’s geographical coordinates</td>
</tr>
<tr>
<td>cluster_color</td>
<td>The color of symbols within a cluster, or NULL if symbols have mixed colors</td>
</tr>
<tr>
<td>cluster_size</td>
<td>The number of symbols contained within a cluster</td>
</tr>
<tr>
<td>current_feature</td>
<td>The feature currently being edited in the attribute form or table row</td>
</tr>
<tr>
<td>current_geometry</td>
<td>The geometry of the feature currently being edited in the form or the table row</td>
</tr>
<tr>
<td>fullextent_maxx</td>
<td>Maximum x value from full canvas extent (including all layers)</td>
</tr>
<tr>
<td>fullextent_maxy</td>
<td>Maximum y value from full canvas extent (including all layers)</td>
</tr>
<tr>
<td>fullextent_minx</td>
<td>Minimum x value from full canvas extent (including all layers)</td>
</tr>
<tr>
<td>fullextent_miny</td>
<td>Minimum y value from full canvas extent (including all layers)</td>
</tr>
<tr>
<td>geometry_part_count</td>
<td>The number of parts in rendered feature’s geometry</td>
</tr>
<tr>
<td>geometry_part_num</td>
<td>The current geometry part number for feature being rendered</td>
</tr>
<tr>
<td>geometry_point_count</td>
<td>The number of points in the rendered geometry’s part</td>
</tr>
<tr>
<td>geometry_point_num</td>
<td>The current point number in the rendered geometry’s part</td>
</tr>
<tr>
<td>grid_axis</td>
<td>The current grid annotation axis (e.g., x’ for longitude, y’ for latitude)</td>
</tr>
<tr>
<td>grid_number</td>
<td>The current grid annotation value</td>
</tr>
<tr>
<td>item_id</td>
<td>The layout item user ID (not necessarily unique)</td>
</tr>
<tr>
<td>item_uuid</td>
<td>The layout item unique ID</td>
</tr>
<tr>
<td>layer</td>
<td>The current layer</td>
</tr>
<tr>
<td>layer_id</td>
<td>The ID of current layer</td>
</tr>
<tr>
<td>layer_name</td>
<td>The name of current layer</td>
</tr>
<tr>
<td>layout_dpi</td>
<td>The composition resolution (DPI)</td>
</tr>
<tr>
<td>layout_name</td>
<td>The layout name</td>
</tr>
<tr>
<td>layout_numpages</td>
<td>The number of pages in the layout</td>
</tr>
<tr>
<td>layout_page</td>
<td>The page number of the current item in the layout</td>
</tr>
<tr>
<td>layout_pageheight</td>
<td>The active page height in the layout (in mm)</td>
</tr>
<tr>
<td>layout_pagewidth</td>
<td>The active page width in the layout (in mm)</td>
</tr>
<tr>
<td>legend_column_count</td>
<td>The number of columns in the legend</td>
</tr>
<tr>
<td>legend_filter_by_map</td>
<td>Indicates if the content of the legend is filtered by the map</td>
</tr>
<tr>
<td>legend_filter_out_atlas</td>
<td>Indicates if the atlas is filtered out of the legend</td>
</tr>
<tr>
<td>legend_split_layers</td>
<td>Indicates if layers can be split in the legend</td>
</tr>
<tr>
<td>legend_title</td>
<td>The title of the legend</td>
</tr>
</tbody>
</table>
Tabulka 14.8 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Function</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>legend_wrap_string</td>
<td>The character(s) used to wrap the legend text</td>
</tr>
<tr>
<td>map_crs</td>
<td>The Coordinate reference system of the current map</td>
</tr>
<tr>
<td>map_crs_acronym</td>
<td>The acronym of the Coordinate reference system of the current map</td>
</tr>
<tr>
<td>map_crs_definition</td>
<td>The full definition of the Coordinate reference system of the current map</td>
</tr>
<tr>
<td>map_crs_description</td>
<td>The name of the Coordinate reference system of the current map</td>
</tr>
<tr>
<td>map_crs_ellipsoid</td>
<td>The acronym of the ellipsoid of the Coordinate reference system of the current map</td>
</tr>
<tr>
<td>map_crs_proj4</td>
<td>The Proj4 definition of the Coordinate reference system of the current map</td>
</tr>
<tr>
<td>map_crs_wkt</td>
<td>The WKT definition of the Coordinate reference system of the current map</td>
</tr>
<tr>
<td>map_extent</td>
<td>The geometry representing the current extent of the map</td>
</tr>
<tr>
<td>map_extent_center</td>
<td>The point feature at the center of the map</td>
</tr>
<tr>
<td>map_extent_height</td>
<td>The current height of the map</td>
</tr>
<tr>
<td>map_extent_width</td>
<td>The current width of the map</td>
</tr>
<tr>
<td>map_id</td>
<td>The ID of current map destination. This will be 'canvas' for canvas renders, and the item ID for layout map renders</td>
</tr>
<tr>
<td>map_layer_ids</td>
<td>The list of map layer IDs visible in the map</td>
</tr>
<tr>
<td>map_layers</td>
<td>The list of map layers visible in the map</td>
</tr>
<tr>
<td>map_rotation</td>
<td>The current rotation of the map</td>
</tr>
<tr>
<td>map_scale</td>
<td>The current scale of the map</td>
</tr>
<tr>
<td>map_units</td>
<td>The units of map measurements</td>
</tr>
<tr>
<td>notification_message</td>
<td>Content of the notification message sent by the provider (available only for actions triggered by provider notifications).</td>
</tr>
<tr>
<td>parent</td>
<td>Refers to the current feature in the parent layer, providing access to its attributes and geometry when filtering an aggregate function</td>
</tr>
<tr>
<td>project_abstract</td>
<td>The project abstract, taken from project metadata</td>
</tr>
<tr>
<td>project_area_units</td>
<td>The area unit for the current project, used when calculating areas of geometries</td>
</tr>
<tr>
<td>project_author</td>
<td>The project author, taken from project metadata</td>
</tr>
<tr>
<td>project_basename</td>
<td>The basename of current project’s filename (without path and extension)</td>
</tr>
<tr>
<td>project_creation_date</td>
<td>The project creation date, taken from project metadata</td>
</tr>
<tr>
<td>project_crs</td>
<td>The Coordinate reference system of the project</td>
</tr>
<tr>
<td>project_crs_acronym</td>
<td>The acronym of the Coordinate reference system of the project</td>
</tr>
<tr>
<td>project_crs_definition</td>
<td>The full definition of the Coordinate reference system of the project</td>
</tr>
<tr>
<td>project_crs_description</td>
<td>The description of the Coordinate reference system of the project</td>
</tr>
<tr>
<td>project_crs_ellipsoid</td>
<td>The ellipsoid of the Coordinate reference system of the project</td>
</tr>
<tr>
<td>project_crs_proj4</td>
<td>The Proj4 representation of the Coordinate reference system of the project</td>
</tr>
<tr>
<td>project_crs_wkt</td>
<td>The WKT (well known text) representation of the coordinate reference system of the project</td>
</tr>
<tr>
<td>project_distance_units</td>
<td>The distance unit for the current project, used when calculating lengths of geometries and distances</td>
</tr>
<tr>
<td>project_ellipsoid</td>
<td>The name of the ellipsoid of the current project, used when calculating geodetic areas or lengths of geometries</td>
</tr>
<tr>
<td>project_filename</td>
<td>The filename of the current project</td>
</tr>
<tr>
<td>project_folder</td>
<td>The folder of the current project</td>
</tr>
<tr>
<td>project_home</td>
<td>The home path of the current project</td>
</tr>
<tr>
<td>project_identifier</td>
<td>The project identifier, taken from the project’s metadata</td>
</tr>
<tr>
<td>project_keywords</td>
<td>The project keywords, taken from the project’s metadata</td>
</tr>
<tr>
<td>project_last_saved</td>
<td>Date/time when project was last saved.</td>
</tr>
<tr>
<td>project_path</td>
<td>The full path (including file name) of the current project</td>
</tr>
<tr>
<td>project_title</td>
<td>The title of current project</td>
</tr>
<tr>
<td>project_units</td>
<td>The units of the project’s CRS</td>
</tr>
<tr>
<td>qgis_locale</td>
<td>The current language of QGIS</td>
</tr>
<tr>
<td>qgis_os_name</td>
<td>The current Operating system name, eg ’windows’, ’linux’ or ’osx’</td>
</tr>
<tr>
<td>qgis_platform</td>
<td>The QGIS platform, eg ’desktop’ or ’server’</td>
</tr>
<tr>
<td>qgis_release_name</td>
<td>The current QGIS release name</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Některé příklady:

- Return the X coordinate of a map item center in layout:

  \[
  \text{x( map_get( item_variables( 'map1' ), 'map_extent_center' ) )}
  \]

- Return, for each feature in the current layer, the number of overlapping airport features:

  \[
  \text{aggregate( layer:'airport', aggregate:'count', expression:'code', filter:intersects( $geometry, geometry( @parent ) ) )}
  \]

- 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'
    )
  )
  ```

Recent Functions

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.
14.2.3 Function Editor

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.

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.

Obr. 14.66: The Function Editor tab
Here's a short example on how to create your own functions:

```python
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 short example creates a function `my_sum` that will give you a function with two values. When using the `args='auto'` function argument the number of function arguments required will be calculated by the number of arguments the function has been defined with in Python (minus `feature` and `parent`).

This function can then be used in expressions:

![Custom Function added to the Expression tab](obr.14.67.png)

Obr. 14.67: Custom Function added to the Expression tab

Further information about creating Python code can be found in the PyQGIS-Developer-Cookbook.
14.3 Working with the Attribute Table

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.

14.3.1 Foreword: Spatial and non-spatial tables

QGIS allows you to load spatial and non-spatial layers. This currently includes tables supported by OGR and delimited text, as well as the PostgreSQL, MSSQL, SpatiaLite, DB2 and Oracle provider. 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.

14.3.2 Introducing the attribute table interface

To open the attribute table for a vector layer, activate the layer by clicking on it in the Layers Panel. 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.

![Attribute Table for regions layer](Obr. 14.68: Attribute Table for regions layer)
The buttons at the top of the attribute table window provide the following functionality:

**Tabulka 14.9: Available Tools**

<table>
<thead>
<tr>
<th>Ikona</th>
<th>Label</th>
<th>Účel</th>
<th>Default Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>![toggle editing mode]</td>
<td>Toggle editing mode</td>
<td>Enable editing functionalities</td>
<td>Ctrl+E</td>
</tr>
<tr>
<td>![toggle multi edit mode]</td>
<td>Toggle multi edit mode</td>
<td>Update multiple fields of many features</td>
<td></td>
</tr>
<tr>
<td>![reload]</td>
<td>Reload the table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>![add feature]</td>
<td>Add feature</td>
<td>Add new geometryless feature</td>
<td></td>
</tr>
<tr>
<td>![delete selected features]</td>
<td>Delete selected features</td>
<td>Remove selected features from the layer</td>
<td></td>
</tr>
<tr>
<td>![cut selected features to clipboard]</td>
<td>Cut selected features to clipboard</td>
<td></td>
<td>Ctrl+X</td>
</tr>
<tr>
<td>![copy selected features to clipboard]</td>
<td>Copy selected features to clipboard</td>
<td></td>
<td>Ctrl+C</td>
</tr>
<tr>
<td>![paste features from clipboard]</td>
<td>Paste features from clipboard</td>
<td>Insert new features from copied ones</td>
<td>Ctrl+V</td>
</tr>
<tr>
<td>![select features using an expression]</td>
<td>Select features using an expression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>![select all]</td>
<td>Select All</td>
<td>Select all features in the layer</td>
<td>Ctrl+A</td>
</tr>
<tr>
<td>![invert selection]</td>
<td>Invert selection</td>
<td>Invert the current selection in the layer</td>
<td>Ctrl+R</td>
</tr>
<tr>
<td>![deselect all]</td>
<td>Deselect all</td>
<td>Deselect all features in the current layer</td>
<td>Ctrl+Shift+A</td>
</tr>
<tr>
<td>![filter/select features using form]</td>
<td>Filter/Select features using form</td>
<td></td>
<td>Ctrl+F</td>
</tr>
<tr>
<td>![move selected to top]</td>
<td>Move selected to top</td>
<td>Move selected rows to the top of the table</td>
<td>Ctrl+P</td>
</tr>
<tr>
<td>![zoom map to the selected rows]</td>
<td>Zoom map to the selected rows</td>
<td></td>
<td>Ctrl+J</td>
</tr>
<tr>
<td>![new field]</td>
<td>New field</td>
<td>Add a new field to the data source</td>
<td>Ctrl+W</td>
</tr>
<tr>
<td>![delete field]</td>
<td>Delete field</td>
<td>Remove a field from the data source</td>
<td></td>
</tr>
<tr>
<td>![open field calculator]</td>
<td>Open field calculator</td>
<td>Update field for many features in a row</td>
<td>Ctrl+I</td>
</tr>
<tr>
<td>![conditional formatting]</td>
<td>Conditional formatting</td>
<td>Enable table formatting</td>
<td></td>
</tr>
<tr>
<td>![dock attribute table]</td>
<td>Dock attribute table</td>
<td>Allows to dock/undock the attribute table</td>
<td></td>
</tr>
<tr>
<td>![actions]</td>
<td>Actions</td>
<td>Lists the actions related to the layer</td>
<td></td>
</tr>
</tbody>
</table>

**Poznámka:** Depending on the format of the data and the OGR 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 *Editing attribute values*).
**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.

- 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. Once you markered the feature in yellow in the list it is selected in yellow on the canvas. Use the magnifying glass on top of the attribute table to zoom to the feature. Clicking on an entry in the list (without using the rectangles) makes a feature flash in red color once so you can see where it is situated.

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“.

---

Obr. 14.69: 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 configure what can be displayed in the attribute table and how.

Hiding and organizing columns and enabling actions

By right-clicking in a column header, you can choose to hide it from the attribute table. To change several columns behavior at once, unhide a column or change the order of the columns, choose Organize columns …. In the new dialog, you can:

- check/uncheck columns you want to show or hide
- 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
- enable a new virtual Actions column that displays in each row a drop-down box or button list of actions for each row, see Actions Properties for more information about actions.

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
- Autosize to resize at the best fit the column.

It can also be changed by dragging the boundary on the right of the column heading. The new size of the column is maintained for the layer, and restored at the next opening of the attribute table.

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 row using multiple columns you can write concat(col0, col1).

In form view, features identifier can be sorted using the Sort by preview expression option.

Tip: 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…).

You can enable the conditional formatting panel clicking on the top right of the attributes window in table view (not available 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 chosen from a list of predefined formats or created based on properties like:
  - background and text colors;
  - use of icon;
  - bold, italic, underline, or strikeout;
  - font.

Obr. 14.70: Conditional Formatting of an attribute table

14.3. Working with the Attribute Table 325
14.3.3 Interacting with features in an attribute table

Selecting features

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):

- Select By Expression…
- Select Features By Value…
- Deselect Features from All Layers
- Select All Features
- Invert Feature Selection.

It is also possible to select features using the Filtering and selecting features using forms.

Filtering features

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
- Show Features visible on map
- Show Edited and New Features
- Field Filter - allows the user to filter based on value of a field: choose a column from a list, type a value and press Enter to filter. Then, only the matching features are shown in the attribute table.
- 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.
It is also possible to **filter features using forms**.

**Poznámka:** 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**.

---

**Tip:** 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.

---

### 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.

Obr. 14.71: 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**.

---

14.3. **Working with the Attribute Table**
14.3.4 Using action on features

Users have several possibilities to manipulate feature with the contextual menu like:

- **Select all** (`Ctrl+A`) the features;
- 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.

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 **Actions Properties** 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 OGR creation options within the dialog.

### 14.3.5 Editing attribute values

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 figure_field_calculator). 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.

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. Load the shapefile railroads.shp in QGIS and press Open Attribute Table.
2. Click on Toggle editing mode and open the Field Calculator dialog.
3. Select the Create a new field checkbox to save the calculations into a new field.
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.
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 an area to be evaluated as you digitize features or to automatically calculate a duration between dates that may change (e.g., using `now()` function).

Poznámka: Use of Virtual Fields

- 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.

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.

**Poznámka:** 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.

Obr. 14.75: 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.

**Poznámka:** 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.
14.3.6 Creating one or many to many relations

Relations are a technique often used in databases. The concept is that features (rows) of different layers (tables) can belong to each other.

Introducing 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.

Obr. 14.76: Alaska region with airports

Layers in 1-N relations

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 shapefile and the airports shapefile which has a foreign key field (fk_region) to the layer regions. This means, that each airport belongs to exactly one region while each region can have any number of airports (a typical one to many relation).

Foreign keys in 1-N relations

In addition to the already existing attributes in the airports attribute table, you’ll need 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 field fk_region will always contain an id of a region. It can be seen like a pointer to the region it belongs to. And 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) and all you have to do is to tell QGIS the relations between your tables.
Defining 1-N relations (Relation Manager)

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 can define the primary key of the referenced layer, so it is ID. For this layer you can define multiple referenced fields by using the + button.

- **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. When using multiple field relations you can add another referencing field by using the + button.

- **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.

Obr. 14.77: Relation Manager
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.

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.

- The button will add a new record to the airport layer attribute table. And it will assign the new airport to the current region by default.

- The button allows you to copy one or more child features.

- The button will delete the selected airport permanently.

- The symbol will open 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 will unlink the selected airport 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 table view and form view where the later lets you view all the airports in their respective form.

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.

Obr. 14.79: 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 Obr. 14.80, 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.
Obr. 14.80: Select parent features with child values
### Introducing 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 airlines 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.

```sql
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);
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:
### 14.3. Working with the Attribute Table

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.

![Relations and autodiscover](image)

Obr. 14.81: 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.

**Poznámka: 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.

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.

```
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;
```
Obr. 14.82: Set relationship cardinality

Obr. 14.83: N-M relationship between airports and airlines
Poznámka: 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 are layers (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.

Poznámka: 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.

14.4 Editování

QGIS has various capabilities for editing OGR, SpatiaLite, PostGIS, MSSQL Spatial and Oracle Spatial vector layers and tables.

Poznámka: The procedure for editing GRASS layers is different - see section Digitizing and editing a GRASS vector layer for details.

Tip: Souběžné editace

This version of 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.

14.4.1 Nastavení přichytávací tolerance a vyhledání poloměru

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.

Přichytávací 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 snapping tolerance setting affects all tools that work with tolerance.

You can enable / disable snapping by using the Enable snapping button on the Snapping Toolbar or pressing s. The snapping mode, tolerance value, and units can also be configured in this toolbar.

The snapping configuration can also be set in Project Snapping Options.…

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 and tolerance on a layer basis (see figure_edit_snapping). 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.

As for snapping mode, you can choose between **To vertex**, **To segment**, and **To vertex and segment**.

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.

![Snapping options](image)

Obr. 14.84: Snapping options (Advanced Configuration mode)

**Poznámka:** 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.

---

**Tip:** 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.
Enable snapping on intersections

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.

Vyhledání poloměru

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. The search radius for vertex edits can be defined under the Settings Options Digitizing tab (this is where you define the snapping default values).

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.

14.4.2 Topologická editace

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.

Povolení topologické editace

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.

Avoid overlap of new polygons

When the snapping mode is set to Advanced configuration, for polygon layers, there’s an option called Avoid overlap. This option prevents you from drawing new features that overlap existing ones in the selected layer, speeding up digitizing of adjacent polygons.

With avoid overlap enabled, if you already have one polygon, you can digitize a second one such that they intersect. QGIS will then cut the second polygon to the boundary of the existing one. The advantage is that you don’t have to digitize all vertices of the common boundary.

Poznámka: If the new geometry is totally covered by existing ones, it gets cleared, and QGIS will show an error message.

Varování: 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.
Geometry Checker

A core plugin can help the user to find the geometry invalidity. You can find more information on this plugin at Geometry Checker Plugin.

Automatic Tracing

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.
   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.

Poznámka: 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.

Poznámka: 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.

Tip: 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.
14.4.3 Digitizing an existing layer

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 Exploring Data Formats and Fields), and the underlying data source is writable (i.e., its files are not read-only).

Tip: 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 Advanced digitizing. You can select and unselect both under View Toolbars.

Using the basic digitizing tools, you can perform the following functions:

<table>
<thead>
<tr>
<th>Ikona</th>
<th>Účel</th>
<th>Ikona</th>
<th>Účel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Current edits</td>
<td></td>
<td>Přepnout editaci</td>
</tr>
<tr>
<td></td>
<td>Save layer edits</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add new record</td>
<td>Add Feature: Capture Point</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add Feature: Capture Line</td>
<td>Add Feature: Capture Polygon</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vertex Tool (All Layers)</td>
<td>Vertex Tool (Current Layer)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify the attributes of all selected features simultaneously</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vymazat vybrané</td>
<td></td>
<td>Vyjmout prvky</td>
</tr>
<tr>
<td></td>
<td>Kopírovat prvky</td>
<td></td>
<td>Vložit prvky</td>
</tr>
<tr>
<td></td>
<td>Zpět</td>
<td></td>
<td>Znovu</td>
</tr>
</tbody>
</table>

Table Editing: Vector layer basic editing toolbar

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 Edit 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.

Tip: Save Regularly

Remember to Save Layer Edits regularly. This will also check that your data source can accept all the changes.
Adding Features

Depending on the layer type, you can use the 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. 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. Having set the geometry precision in the layer properties you can use snap to grid here to create features based on a regular distance.

2. For line or polygon geometries, keep on left-clicking for each additional point you wish to capture or use automatic tracing capability to accelerate the digitization. This will create consecutive straight lines between the vertices you place.

**Poznámka:** Pressing Delete or Backspace key reverts the last node you add.

3. When you have finished adding points, right-click anywhere on the map area to confirm you have finished entering the geometry of that feature.

**Poznámka:** While digitizing line or polygon geometries, you can switch back and forth between the linear Add feature tools and circular string tools to create compound curved geometries.

**Tip:** 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.

4. The attribute window will appear, allowing you to enter the information for the new feature. Figure edit values shows setting attributes for a fictitious new river in Alaska. However, in the Digitizing menu under the Settings Options menu, you can also activate:

   - 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.

![Obr. 14.85: Enter Attribute Values Dialog after digitizing a new vector feature](image-url)
**Vertex tool**

**Poznámka: QGIS 3 major changes**

In QGIS 3, the node tool has been fully redesigned and renamed to *vertex tool*. It was previously working with „click and drag“ ergonomy, and now uses a „click - click“ workflow. This allows major improvements like taking profit of the advanced digitizing panel with the vertex tool while digitizing or editing objects of multiple layers at the same time.

For any editable vector layer, the **Vertex tool** provides manipulation capabilities of feature vertices similar to CAD programs. It is possible to simply select multiple vertices at once and to move, add or delete them altogether. The vertex tool also supports the topological editing feature. This tool is selection persistent, so when some operation is done, selection stays active for this feature and tool.

It is important to set the property **Search Radius** to a number greater than zero. Otherwise, QGIS will not be able to tell which vertex is being edited and will display a warning.

**Tip: Vertex Markers**

The current version of QGIS supports three 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.

**Basic operations**

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, or by clicking and dragging a rectangle around some vertices. When a vertex is selected, its color changes to blue. To add more vertices to the current selection, hold down the Shift key while clicking. To remove vertices from the selection, hold down Ctrl.

- **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).

  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, a virtual new node appears on the segment center. Simply grab it to add a new vertex. A double-click on any location of the boundary also creates a new node. For lines, a virtual node is also proposed at both extremities of a line to extend it.

- **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 again on the desired new location. All the selected vertices will move together. If snapping is enabled, the whole selection can jump to the nearest vertex or line. You can use Advanced Digitizing Panel constraints for distance, angles, exact X Y location before the second click.

  Here you can use the snap-to-grid feature. Having set a value for the **geometry precision** in the layer properties, a grid appears on a zoom level according to the Geometry precision.

Each change made with the vertex 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, and the vertex tool provides tooltips to identify a vertex by hovering the pointer over it.
Obr. 14.86: Batch vertex selection using \texttt{Shift+R}

Obr. 14.87: Virtual nodes for adding vertices
The Vertex Editor Panel

When using the Vertex tool on a feature, it is possible to right click to open the Vertex Editor panel listing 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). Simply select a row in the table does select the corresponding vertex in the map canvas, and vice versa. Simply change a coordinate in the table and your vertex position is updated. You can also select multiple rows and delete them altogether.

Poznámka: Changed behavior in QGIS 3.4

Right click on a feature will immediately show the vertex editor and lock this feature, thus disabling the editing of any other features. While being locked, a feature is exclusive for editing: Selecting and moving of vertices and segments by clicking or dragging is only possible for this feature. New vertices can only be added to the locked feature. Also, the vertex editor panel now opens itself automatically upon activating the vertex tool, and its position/docked state remembered across uses.

Cutting, Copying and Pasting Features

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.

Tip: 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.

As an example, we will copy some lakes to a new layer:

1. Load the layer you want to copy from (source layer)
2. Load or create the layer you want to copy to (target layer)
3. Start editing for target layer
4. Make the source layer active by clicking on it in the legend
5. Use the Select Features by area or single click tool to select the feature(s) on the source layer
6. Click on the Copy Features tool
7. Make the destination layer active by clicking on it in the legend
8. Click on the Paste Features tool
9. Stop editing and save the changes

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.

Poznámka: Congruency of Pasted Features

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.

Tip: Copy string attribute into another

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 starting with GDAL/OGR 1.10 knows to auto-extend string and integer fields to dynamically accommodate for the length of the data to be inserted.

Deleting Selected Features

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.

Zpět a znovu

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 Figure_edit_undo). 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.

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.

Saving Edited Layers

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 (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.

Tip: Data Integrity
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 \(\text{Save for Selected Layers}\) to save all changes you made in multiple layers. You also have the opportunity to \(\text{Rollback for Selected Layers}\), so that the digitization may be withdrawn for all selected layers. If you want to stop editing the selected layers, \(\text{Cancel for Selected Layer(s)}\) is an easy way.

The same functions are available for editing all layers of the project.

---

**Tip: 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).

---

### 14.4.4 Advanced digitizing

<table>
<thead>
<tr>
<th>Ikona</th>
<th>Učel</th>
<th>Ikona</th>
<th>Učel</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Enable Advanced Digitizing Tools" /></td>
<td>Enable Advanced Digitizing Tools</td>
<td><img src="#" alt="Enable Tracing" /></td>
<td>Enable Tracing</td>
</tr>
<tr>
<td><img src="#" alt="Move Feature(s)" /></td>
<td>Move Feature(s)</td>
<td><img src="#" alt="Copy and Move Feature(s)" /></td>
<td>Copy and Move Feature(s)</td>
</tr>
<tr>
<td><img src="#" alt="Rotovat prvek(prvky)" /></td>
<td>Rotovat prvek(prvky)</td>
<td><img src="#" alt="Zjednodušit prvek" /></td>
<td>Zjednodušit prvek</td>
</tr>
<tr>
<td><img src="#" alt="Přidat prstenec" /></td>
<td>Přidat prstenec</td>
<td><img src="#" alt="Přidat část" /></td>
<td>Přidat část</td>
</tr>
<tr>
<td><img src="#" alt="Fill Ring" /></td>
<td>Fill Ring</td>
<td><img src="#" alt="Swap direction" /></td>
<td>Swap direction</td>
</tr>
<tr>
<td><img src="#" alt="Smazat prstenec" /></td>
<td>Smazat prstenec</td>
<td><img src="#" alt="Smazat část" /></td>
<td>Smazat část</td>
</tr>
<tr>
<td><img src="#" alt="Odsazení křivky" /></td>
<td>Odsazení křivky</td>
<td><img src="#" alt="Změnit tvar prvků" /></td>
<td>Změnit tvar prvků</td>
</tr>
<tr>
<td><img src="#" alt="Split Parts" /></td>
<td>Split Parts</td>
<td><img src="#" alt="Rozdělit objekt" /></td>
<td>Rozdělit objekt</td>
</tr>
<tr>
<td><img src="#" alt="Sloučit atributy vybraných prvků" /></td>
<td>Sloučit atributy vybraných prvků</td>
<td><img src="#" alt="Sloučit vybrané prvky" /></td>
<td>Sloučit vybrané prvky</td>
</tr>
<tr>
<td><img src="#" alt="Rotovat symboly bodů" /></td>
<td>Rotovat symboly bodů</td>
<td><img src="#" alt="Offset Point Symbols" /></td>
<td>Offset Point Symbols</td>
</tr>
</tbody>
</table>

Table Advanced Editing: Vector layer advanced editing toolbar
Move Feature(s)

The tool allows you to move existing features:

1. Select the feature(s) to move.
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 endpoint of the translation.
4. Click on the map canvas: the whole features are moved to new location.

Likewise, you can create a translated copy of the feature(s) using the tool.

Poznámka: If no feature is selected when you first click on the map canvas with any of the Move Feature(s) or Copy and Move Feature(s) tools, then only the feature under the mouse is affected by the action. So, if you want to move several features, they should be selected first.

Rotovat prvky

Use the tool to rotate one or multiple features in the map canvas:

1. Press the 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 icon.
Zjednodušit prvek

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, 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.

Poznámka: 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.

Přidat část

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.

Smazat část

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.
**Přidat prstenec**

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.

**Fill Ring**

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.

**Smazat prstenec**

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.

**Změnit tvar prvků**

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.

![Obr. 14.91: Reshape line](image-url)
Tip: 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.

Obr. 14.92: 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.

Poznámka: 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.

Odsazení křivek

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. 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.
**Reverse Line**

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 
2. Click on the line. The direction of the line is reversed.

**Rozdělit objekt**

Use the tool to split a feature into two or more new and independent features, i.e. each geometry corresponding to a new row in the attribute table.

To split line or polygon features:

1. Select the tool.
2. Draw a line across the feature(s) you want to split. If a selection is active, only selected features are split. When set, *default values or clauses* are applied to corresponding fields and other attributes of the parent feature are by default copied to the new features.
3. You can then as usually modify any of the attributes of any resulting feature.

**Tip: Split a polyline into new features in one-click**

Using the tool, snap and click on an existing vertex of a polyline feature to split that feature into two new features.

**Split parts**

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 icon.

**Tip: Split a polyline into new parts in one-click**

Using the 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.

**Sloučit vybrané prvky**

The 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 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 Skip all fields to use empty attributes;
• or, 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).

**Poznámka:** 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.

**Sloučit atributy vybraných prvků**

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.

**Rotovat symboly bodů**

The **Rotate Point Symbols** tool allows you to change the rotation of point symbols in the map canvas.

1. First of all, apply to the symbol a data-defined rotation:
   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.

**Poznámka:** 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.
2. Then click on a point feature in the map canvas with the [Rotate Point Symbols] and move the mouse around, holding the left button pressed. A red arrow with the rotation value will be visualized (see [Figure_rotate_point]).

3. Release the left mouse button again, the symbol is defined with this new rotation and the rotation field is updated in the layer’s attribute table.

**Tip:** Pokud podržíte klávesu Ctrl, provede se rotace o 15 stupňů.

### Offset Point Symbols

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 the symbol, field which will then be populated with the offset coordinates while moving the symbol in the map canvas.

**Poznámka:** 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.

### Trim/Extend Feature

When a digitized line is too short or too long to snap to another line (missing or crossing the line), it is necessary to be able to extend or shorten the segment.

The [Trim/Extend] tool allows you to also modify (multi)lines AND (multi)polygons. Moreover, it is not necessarily the end of the lines that is concerned; any segment of a geometry can be modified.

**Poznámka:** This can lead to invalid geometries.

**Poznámka:** You must activate segment snapping for this tool to work.

The tool asks you to select a limit (a segment) with respect to which another segment will be extended or trimmed. Unlike the vertex tool, a check is performed to modify only the layer being edited.

When both segments are in 3D, the tool performs an interpolation on the limit segment to get the Z value.

In the case of a trim, you must select the part that will be shortened by clicking on it.

### 14.4.5 Shape digitizing

The Shape Digitizing toolbar offers a set of tools to draw regular shapes and curved geometries.
Add Circular string

The Add circular string or Add circular string by radius buttons allow users to add line or polygon features with a circular geometry.

Creating features with these tools follow the same rule as of other digitizing tools: left-click to place vertices and right-click to finish the geometry. While drawing the geometry, you can switch from one tool to the other as well as to the linear geometry tools, creating some compound geometries.

Poznámka: 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.

Draw Circles

There is a set of tools for drawing circles. The tools are described below.

Circles are converted into circular strings. Therefore, as explained in Add Circular string, if allowed by the data provider, it will be saved as a curved geometry, if not, QGIS will segmentize the circular arcs.

- Add circle from 2 points: The two points define the diameter and the orientation of the circle. (Left-click, right-click)
- Add circle from 3 points: Draws a circle from three known points on the circle. (Left-click, left-click, right-click)
- Add circle from center and a point: Draws a circle with a given center and a point on the circle (Left-click, right-click). When used with the The Advanced Digitizing panel this tool can become a „Add circle from center and radius“ tool by setting and locking the distance value after first click.
- Add circle from 3 tangents: Draws a circle that is tangential to three segments. Note that you must activate snapping to segments (See Nastavení přichytávací tolerance a vyhledání poloměru). Click on a segment to add a tangent. If two tangents are parallel, an error message appears and the input is cleared. (Left-click, left-click, right-click)
- Add 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.

Draw Ellipses

There is a set of tools for drawing ellipses. The tools are described below.

Ellipses cannot be converted as circular strings, so they will always be segmented.

- Add Ellipse from center and two points: Draws an ellipse with a given center, major axis and minor axis. (Left-click, left-click, right-click)
- Add Ellipse from center and a point: Draws an ellipse into a bounding box with the center and a corner. (Left-click, right-click)
- Add Ellipse from extent: Draws an ellipse into a bounding box with two opposite corners. (Left-click, right-click)
- Add Ellipse from foci: Draws an ellipse by 2 points for foci and a point on the ellipse. (Left-click, left-click, right-click)
Draw Rectangles

There is a set of tools for drawing rectangles. The tools are described below.

- **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. (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)

Obr. 14.94: Draw rectangle from 3 points using distance (right) and projected (left)

Draw Regular Polygons

There is a set of tools for drawing regular polygons. The tools are described below. Left-click to place the first point. A dialog appears, where you can set the number of polygon edges. Right-click to finish the regular polygon.

- **Regular polygon from two points**: Draws a regular polygon where the two points determine the length and angle of the first edge.
- **Regular polygon from center and a point**: Draws a regular polygon from the provided center point. The second point determines the angle and distance to the middle of an edge.
- **Regular polygon from center and a corner**: Same as the preceding tool, but the second point determines the angle and distance to a vertex.
14.4.6 The Advanced Digitizing panel

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 enter coordinates directly so that you can make a precise definition of your new geometry.

Obr. 14.95: The Advanced Digitizing panel

The Advanced Digitizing panel can be open 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.

Poznámka: The tools are not enabled if the map view is in geographic coordinates.

Concepts

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.

Snapping Settings

Click the button to set the Advanced Digitizing Tool snapping settings. You can make the tool snap to common angles. The options are:

- Do not snap to common angles
- Snap to 30º angles
- Snap to 45º angles
- Snap to 90º angles

You can also control the snapping to features. The options are:

- Do not snap to vertices or segments
- Snap according to project configuration
- Snap to all layers
Keyboard shortcuts

To speed up the use of Advanced Digitizing Panel, there are a couple of keyboard shortcuts available:

<table>
<thead>
<tr>
<th>Key</th>
<th>Simple</th>
<th>Ctrl+ or Alt+</th>
<th>Shift+</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Set distance</td>
<td>Lock distance</td>
<td>Toggle relative angle to last segment</td>
</tr>
<tr>
<td>A</td>
<td>Set angle</td>
<td>Lock angle</td>
<td>Toggle relative X to last vertex</td>
</tr>
<tr>
<td>X</td>
<td>Set X coordinate</td>
<td>Lock X coordinate</td>
<td>Toggle relative X to last vertex</td>
</tr>
<tr>
<td>Y</td>
<td>Set Y coordinate</td>
<td>Lock Y coordinate</td>
<td>Toggle relative Y to last vertex</td>
</tr>
<tr>
<td>C</td>
<td>Toggle construction mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>Toggle perpendicular and parallel modes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 choose the X and Y coordinates where you want to start editing the feature, then:

- Click the x text box (or use the X keyboard shortcut).
- 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.
- Click the y text box (or use the Y keyboard shortcut).
- 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.

Two blue dotted lines and a green cross identify the exact coordinates you entered. Start digitizing by clicking on the map canvas; the mouse position is locked at the green cross.

You can continue digitizing by free hand, adding a new pair of coordinates, or you can type the segment’s length (distance) and angle.

If you want to draw a segment of a given length, click the d (distance) text box (keyboard shortcut D), type the distance value (in map units) and 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 clicked point is surrounded by a circle whose radius is the value entered in the distance text box.
Finally, you can also choose the angle of the segment. As described before, click the $a$ (angle) text box (keyboard shortcut A), type the angle value (in degrees), and press Enter or click the buttons on the right to lock it. In this way the segment will follow the desired angle:

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 $a$ 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 $x$ or $y$ buttons to the left of the $x$ or $y$ text boxes (or press Shift+X or Shift+Y) to toggle relative coordinates to the previous vertex. With these options on, coordinates measurement will consider the last vertex to be the X and Y axes origin.
Continuous lock

Both in absolute or relative reference digitizing, angle, distance, X and Y 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.

To draw a perpendicular segment, during the editing click the Perpendicular icon (keyboard shortcut P) to activate it. Before drawing the perpendicular line, click on the segment of an existing feature that you want to be perpendicular to (the line of the existing feature will be colored in light orange); you should see a blue dotted line where your feature will be snapped:

Obr. 14.99: Perpendicular digitizing

To draw a parallel feature, the steps are the same: click on the Parallel icon (keyboard shortcut P twice), click on the segment you want to use as reference and start drawing your feature:

These two tools just find the right angle of the perpendicular and parallel angle and lock this parameter during your editing.

Construction mode

You can enable and disable construction mode by clicking on the Construction 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 and Y 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 Construction button, you have to activate also the construction mode by clicking the Construction 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:
Kapitola 14. Working with Vector Data

Obr. 14.100: Parallel digitizing

Obr. 14.101: 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 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.

Obr. 14.102: 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.

Obr. 14.103: Points at given distance and angle

14.4. Editování
14.4.7 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.
4. Find the algorithm you’d like to run and double-click it.

Obr. 14.104: Processing algorithms: all (left) vs polygon in-place editors (right)
Poznámka: 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.
15.1 Raster Properties Dialog

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
- Transparency
- Histogram
- Rendering
- Pyramids
- Metadata
- Legend
- QGIS Server

Tip: 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.

Poznámka: Because properties (symbology, label, actions, default values, forms…) of embedded layers (see Vnořování projektů) 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.

15.1.1 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:

- based on the provider of the layer (format of storage, path, 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 provided metadata: access, links, contacts, history… as well as dataset information (CRS, Extent, bands…).

15.1.2 Source Properties

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.

Obr. 15.1: Raster Layer Properties - Source Dialog
15.1.3 Symbology Properties

Band rendering

QGIS offers four different *Render types*. The choice of renderer depends on the data type.

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.

**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'.

**Poznámka: Contrast enhancement**

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.

**Tip: Viewing a Single Band of a Multiband Raster**

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

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*).
Obr. 15.2: Raster Symbology - Multiband color rendering
Obr. 15.3: Raster Symbology - Paletted unique value rendering
Singleband gray

This renderer allows you to render a single band layer 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'.

![Singleband gray renderer settings](image)

Obr. 15.4: Raster Symbology - Singleband gray rendering

Singleband pseudocolor

This is a render option for single-band files that include a continuous palette. You can also create color maps for a bands of a multiband raster.

Using a *Band* of the layer and a *values range*, three types of color *Interpolation* are available:

- Discrete (a <= symbol appears in the header of the *Value* column)
- Linear
- Exact (an = symbol appears in the header of the *Value* column)

The *Color ramp* drop down lists the available color ramps. 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 and in the QML file.

The *Label unit suffix* is a label added after the value in the legend.

For classification *Mode* 'Equal interval', you only need to select the *number of classes* and press the button *Classify*. For *Mode* 'Continuous', QGIS creates classes automatically depending on *Min* and *Max*.

The button *Add values manually* adds a value to the table. The button *Remove selected row* deletes a value from the table. Double clicking in the *Value* column lets you insert a specific value. Double clicking in the *Color* column opens the dialog *Change color*, where you can select a color to apply for that value. Further, you can also add labels for each color, 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*… for the selection
- *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.

The *Clip out of range values* allows QGIS to not render pixel greater than the *Max* value.
Obr. 15.5: Raster Symbology - Singleband pseudocolor rendering
Hillshade

Render a band of the raster layer using hillshading.

Options:
- **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).

**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.
- **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**: 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).

**Poznámka**: 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 rendering

For all kinds of Band rendering, the Color rendering set.

You can achieve special rendering effects for your raster file(s) by using one of the blending modes (see Režim míchání).

Further settings can be made by modifying the Brightness, Saturation and Contrast. You can also use a Grayscale option, where you can choose between 'Off', 'By lightness', 'By luminosity' and 'By average'. For one Hue in the color table, you can modify the 'Strength'.

### Resampling

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' or 'Cubic' 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.

At the bottom of the Symbology tab, you can see a thumbnail of the layer, its legend symbol, and the palette.
15.1.4 Transparency Properties

QGIS has the ability to set the transparency level of a raster layer. Use the transparency 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.

Additionally, you can enter a raster value that should be treated as an Additional no data value.

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. Click the Add values manually button. A new row will appear in the pixel list.
  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.

As you can see, it is quite easy to set custom transparency, but it can be quite a lot of work. Therefore, you can use the button Export to file to save your transparency list to a file. The button Import from file loads your transparency settings and applies them to the current raster layer.
15.1.5 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).

Obr. 15.10: Raster Histogram
15.1.6 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) scale, 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 Vykreslování v závislosti na měřítku for more information.

- **Refresh layer at interval (seconds)**: set a timer to automatically refresh individual layers. Canvas updates are deferred in order to avoid refreshing multiple times if more than one layer has an auto update interval set.

15.1.7 Pyramids Properties

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.

You must have write access in the directory where the original data is stored to build pyramids.

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.

**Poznámka:** 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:

- Nearest Neighbour
- Average
- Gauss
- Cubic
- Cubic Spline
- Laczos
- Mode
- Žádná

Finally, click Build Pyramids to start the process.

15.1.8 Metadata Properties

The Metadata tab provides you with options to create and edit a metadata report on your layer. See vector layer metadata properties for more information.
15.1.9 Legend Properties

The Legend tab 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.

15.1.10 QGIS Server Properties

From the QGIS Server tab, information can be provided for Description, Attribution, MetadataUrl and LegendUrl.

15.2 Raster Analysis

15.2.1 Raster Calculator

The Raster Calculator in the Raster menu allows you to perform calculations on the basis of existing raster pixel values (see figure_raster_calculator). The results are written to a new raster layer with a GDAL-supported format.

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 then define the extent of the calculation area based on an input raster layer, or based on X,Y coordinates and on columns and rows, to set the resolution of
the output layer. If the input layer has a different resolution, the values will be resampled with the nearest neighbor algorithm.

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. Stay tuned for more operators to come!

With the **Add result to project** checkbox, the result layer will automatically be added to the legend area and can be visualized.

**Examples**

**Convert elevation values from meters to feet**

Creating an elevation raster in feet from a raster in meters, you need to use the conversion factor for meters to feet: 3.28. The expression is:

```
"elevation@1" * 3.28
```

**Using a mask**

If you want to mask out parts of a raster – say, for instance, because you are only interested in elevations above 0 meters – you can use the following expression to create a mask and apply the result to a raster in one step.

```
("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
Obr. 15.15: QGIS Server in Raster Properties
Obr. 15.16: Raster Calculator (abs, min and max added in 3.10)
to 0. This creates the mask on the fly.

If you want to classify a raster – say, for instance into two elevation classes, you can use the following expression to create a raster with two values 1 and 2 in one step.

\[
(*elevation@1* < 50) * 1 + (*elevation@1* >= 50) * 2
\]

In other words, for every cell less than 50 set its value to 1. For every cell greater than or equal 50 set its value to 2.

### 15.2.2 Raster Alignment

This tool is able to take several rasters as input and to align them perfectly, that means:

- reproject to the same CRS,
- resample to the same cell size and offset in the grid,
- clip to a region of interest,
- rescale values when required.

All rasters will be saved in another files.

First, open the tools from Raster... Align Raster... and click on the Add new raster button to choose one existing raster in QGIS. Select an output file to save the raster after the alignment, the resampling method and if the tools need to Rescale values according to the cell size. The resampling method can be (see figure_raster_align_edit):

- Nearest Neighbor
- Bilinear (2x2 kernel)
- Cubic (4x4 kernel): Cubic Convolution Approximation
- Cubic B-Spline (4x4 kernel): Cubic B-Spline Approximation
- Lanczos (6x6 kernel): Lanczos windowed sinc interpolation
- Average: computes the average of all non-NODATA contributing pixels
- Mode: selects the value which appears most often of all the sampled points
- Maximum, Minimum, Mediane, First Quartile (Q1) or Third Quartile (Q3) of all non-NODATA contributing pixels

**Poznámka:** Methods like maximum, minimum, mediane, first and third quartiles are available only if QGIS is built with GDAL >= 2.0.

In the main Align raster dialog, you can still Edit file settings or Remove an existing file from the list of raster layers. You can also choose one or more other options (see figure_raster_align):

- Select the Reference Layer,
- Transform into a new CRS,
- Setup a different Cell size,
- Setup a different Grid Offset,
- Clip to Extent: it can be user-defined or based on a layer or the map view
- Output Size,
- Add aligned raster to the map canvas.
15.2. Raster Analysis

Obr. 15.17: Select Raster Resampling Options

Obr. 15.18: Raster Alignment
16.1 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 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

QGIS can currently render mesh data using triangles or regular quads.

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:

Obr. 16.1: Different mesh types
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:

<table>
<thead>
<tr>
<th>10 metre wind</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 metre speed at time=t1</td>
<td>17251</td>
<td>24918</td>
<td>32858</td>
<td>...</td>
</tr>
<tr>
<td>10 metre speed at time=t2</td>
<td>19168</td>
<td>23001</td>
<td>36418</td>
<td>...</td>
</tr>
<tr>
<td>10 metre speed at time=t3</td>
<td>21085</td>
<td>30668</td>
<td>17251</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10m wind direction time=t1</td>
<td>[20,2]</td>
<td>[20,3]</td>
<td>[20,4.5]</td>
<td>...</td>
</tr>
<tr>
<td>10m wind direction time=t2</td>
<td>[21,3]</td>
<td>[21,4]</td>
<td>[21,5.5]</td>
<td>...</td>
</tr>
<tr>
<td>10m wind direction time=t3</td>
<td>[22,4]</td>
<td>[22,5]</td>
<td>[22,6.5]</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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.
16.2 Supported formats

QGIS accesses mesh data using the MDAL drivers. Hence, the natively supported formats are:

- **NetCDF**: Generic format for scientific data
- **GRIB**: Format commonly used in meteorology
- **XMDF**: As an example, hydraulic outputs from TUFLOW modelling package
- **DAT**: Outputs of various hydrodynamic modelling packages (e.g. BASEMENT, HYDRO_AS-2D, TUFLOW)
- **3Di**: 3Di modelling package format based on Climate and Forecast Conventions (http://cfconventions.org/)
- **Some examples of mesh datasets can be found at** https://apps.ecmwf.int/datasets/data/interim-full-daily/levtype=sfc/

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.

16.3 Mesh Dataset Properties

16.3.1 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) uri, vertex count, face count and dataset groups count.
The **Source** tab displays basic information about the selected mesh, including:

- the Layer name to display in the **Layers** panel
- setting 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 CRS applied to the layer is wrong or if none was applied.

Use the **Assign Extra Dataset to Mesh** button to add more groups to the current mesh layer.

### 16.3.3 Symbology Properties

Click the ![Symbology](image) button to activate the dialog as shown in the following image:

Symbology properties are divided in several tabs:

- **General**
- **Contours Symbology**
- **Vectors Symbology**
- **Rendering**
General

The tab presents the following items:

- groups available in the mesh dataset
- dataset in the selected group(s), for example, if the layer has a temporal dimension
- metadata if available
- blending mode available for the selected dataset.

The slider, the combo box, and the \(<, >, >|\) buttons allow to explore another dimension of the data, if available. As the slider moves, the metadata is presented accordingly. See the figure Mesh groups below as an example. The map canvas will display the selected dataset group as well.

You can apply symbology to each group using the tabs.
**Contours Symbology**

Under *Groups*, click on to show contours with default visualization parameters.

In the tab, you can see and change the current visualization options of contours for the selected group, as shown in **Obr. 16.7** below:

![Contour Symbology](image.png)

**Obr. 16.7: Styling Contours in a Mesh Layer**

- Use the slide bar or combo box to set the opacity of the current group.
- Use *Load* to adjust the min and max values of the current group.
- The *Interpolation* list contains three options to render contours: *Linear*, *Discrete* and *Exact*.
- The *Color ramp* widget opens the *color ramp drop-down shortcut*.
- The *Label unit suffix* is a label added after the value in the legend.
- By selecting *Continuous* in the classification *Mode*, QGIS creates classes automatically considering the *Min* and *Max* values. With ‘Equal interval’, you only need to select the number of classes using the combo box *Classes* and press the button *Classify*.

The button adds a value to the individual color table. The button deletes a
value from the individual color table. Double clicking on the value column lets you insert a specific value. Double clicking on the color column opens the dialog *Change color*, where you can select a color to apply on that value.

**Vectors Symbology**

In the tab ![image](image-url) click on ![image](image-url) to display vectors if available. The map canvas will display the vectors in the selected group with default parameters. Click on the tab ![image](image-url) to change the visualization parameters for vectors as shown in the image below:

![Layer Styling](image-url)

**Obr. 16.8: Styling Vectors in a Mesh Layer**

The line width can be set using the combo box or typing the value. The color widget opens the dialog *Change color*, where you can select a color to apply to vectors.

Enter values for *Min* and *Max* to filter vectors according to their magnitude.

Check on the box ![image](image-url) *Display Vectors on User Grid* and specify the *X spacing* and the *Y spacing*, QGIS will render the vector considering the given spacing.

With the Head Options *Head Options*, QGIS allows the shape of the arrow head to be set by specifying width and length (in percentage).
Vector’s Arrow length can be rendered in QGIS in three different ways:

- Defined by Min and Max: You specify the minimum and maximum length for the vectors, QGIS will adjust their visualization accordingly
- Scale to magnitude: You specify the (multiplying) factor to use
- Fixed: all the vectors are shown with the same length

**Vykreslování**

In the tab, QGIS offers two possibilities to display the grid, as shown in Obr. 16.9:

- Native Mesh Rendering that shows quadrants
- Triangular Mesh Rendering that display triangles

The line width and color can be changed in this dialog, and both the grid renderings can be turned off.
Laying out the maps

With Print Layouts and Reports you can create maps and atlases, and print them or save them as image, PDF or SVG files.

17.1 Overview of the Print Layout

The print layout provides growing layout and printing capabilities. It allows you to add elements such as the QGIS map canvas, text labels, images, legends, scale bars, basic shapes, arrows, attribute tables 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.

17.1.1 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 toolbar or choose File New Print Layout. You will be prompted to choose a title for the new layout.

To demonstrate how to create a map please follow the next instructions.

1. On the left side, select the Add new 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 new scale bar 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 new 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. Select the Select/Move item icon to select the map on the canvas and move it a bit.
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.

17.1.2 The Layout Manager

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:

- 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 button in the Layout Toolbar.

![Obr. 17.1: The Print Layout Manager](image)

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 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.

**Tip:** 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.

### 17.1.3 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: the current QGIS map canvas, text labels, images, legends, scale bars, basic shapes, arrows, attribute tables and HTML frames. 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 contents of the map item.

*figure_layout_overview* shows the initial view of the print layout before any elements are added.

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 The Items Panel for more information).

- The Undo History panel displays a 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* 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 **Layout Items** 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 **Generate an 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.
<table>
<thead>
<tr>
<th>Tool</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>📛 Save Project</td>
<td>Ctrl+S</td>
<td>Layout</td>
<td>Introducing QGIS projects</td>
</tr>
<tr>
<td>🍎 New Layout</td>
<td>Ctrl+N</td>
<td>Layout</td>
<td>The Layout Manager</td>
</tr>
<tr>
<td>📌 Duplicate Layout</td>
<td></td>
<td>Layout</td>
<td>The Layout Manager</td>
</tr>
<tr>
<td>🗑️ Delete Layout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>🗑️ Layout Manager…</td>
<td></td>
<td></td>
<td>The Layout Manager</td>
</tr>
<tr>
<td>🌈 Layout Properties…</td>
<td></td>
<td></td>
<td>The Layout Panel</td>
</tr>
<tr>
<td>Rename Layout…</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>📑 Add Pages…</td>
<td></td>
<td></td>
<td>Working with the page properties</td>
</tr>
<tr>
<td>🌀 Add Items from Template</td>
<td></td>
<td></td>
<td>Creating a layout item</td>
</tr>
<tr>
<td>🏡 Save as Template…</td>
<td></td>
<td></td>
<td>The Layout Manager</td>
</tr>
<tr>
<td>📌 Export as Image…</td>
<td></td>
<td></td>
<td>Export as Image</td>
</tr>
<tr>
<td>🌈 Export as SVG…</td>
<td></td>
<td></td>
<td>Export as SVG</td>
</tr>
<tr>
<td>🌞 Export as PDF…</td>
<td></td>
<td></td>
<td>Export as PDF</td>
</tr>
<tr>
<td>Page Setup…</td>
<td>Ctrl+Shift+P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>🌞 Print…</td>
<td>Ctrl+P</td>
<td></td>
<td>Creating an Output</td>
</tr>
<tr>
<td>Close</td>
<td>Ctrl+Q</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.
### Tabulka 17.1: Available Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo (last change)</td>
<td>Ctrl+Z</td>
<td>Layout</td>
<td>The Undo History Panel: Revert and Restore actions</td>
</tr>
<tr>
<td>Redo (last reverted change)</td>
<td>Ctrl+Y</td>
<td>Layout</td>
<td>The Undo History Panel: Revert and Restore actions</td>
</tr>
<tr>
<td>Delete</td>
<td>Del</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cut</td>
<td>Ctrl+X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td>Ctrl+C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paste</td>
<td>Ctrl+V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paste in place</td>
<td>Ctrl+Shift+V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select All</td>
<td>Ctrl+A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deselect all</td>
<td>Ctrl+Shift+A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invert Selection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Next Item Below</td>
<td>Ctrl+Alt+[</td>
<td></td>
<td>Interacting with layout items</td>
</tr>
<tr>
<td>Select Next Item above</td>
<td>Ctrl+Alt+]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pan Layout</td>
<td>P</td>
<td>Toolbox</td>
<td></td>
</tr>
<tr>
<td>Zoom</td>
<td>Z</td>
<td>Toolbox</td>
<td></td>
</tr>
<tr>
<td>Select/Move Item</td>
<td>V</td>
<td>Toolbox</td>
<td>The Map Item</td>
</tr>
<tr>
<td>Move Content</td>
<td>C</td>
<td>Toolbox</td>
<td>The Node-Based Shape Items</td>
</tr>
<tr>
<td>Edit Nodes Item</td>
<td></td>
<td>Toolbox</td>
<td></td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Navigation</td>
<td></td>
</tr>
<tr>
<td>Preview</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Přiblížit</td>
<td>Ctrl++</td>
<td>Navigation</td>
<td></td>
</tr>
<tr>
<td>Oddálit</td>
<td>Ctrl+-</td>
<td>Navigation</td>
<td></td>
</tr>
<tr>
<td>Zoom to 100%</td>
<td>Ctrl+1</td>
<td>Navigation</td>
<td></td>
</tr>
<tr>
<td>Přiblížit na rozměry okna</td>
<td>Ctrl+0</td>
<td>Navigation</td>
<td></td>
</tr>
<tr>
<td>Zoom to Width</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show Grid</td>
<td>Ctrl+'</td>
<td>Guides and Grid</td>
<td></td>
</tr>
<tr>
<td>Snap to Grid</td>
<td>Ctrl+Shift+'</td>
<td>Guides and Grid</td>
<td></td>
</tr>
<tr>
<td>Show Guides</td>
<td>Ctrl+;</td>
<td>Guides and Grid</td>
<td></td>
</tr>
<tr>
<td>Snap to Guides</td>
<td>Ctrl+Shift+;</td>
<td>Guides and Grid</td>
<td></td>
</tr>
<tr>
<td>Smart Guides</td>
<td>Ctrl+Alt+;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Guides…</td>
<td></td>
<td>The Guides Panel</td>
<td></td>
</tr>
<tr>
<td>Clear Guides</td>
<td></td>
<td>The Guides Panel</td>
<td></td>
</tr>
<tr>
<td>Show Rulers</td>
<td>Ctrl+R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show Bounding Boxes</td>
<td>Ctrl+Shift+B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show Pages</td>
<td></td>
<td>Panely a nástrojové lišty</td>
<td></td>
</tr>
<tr>
<td>Nástrojové lišty</td>
<td></td>
<td>Panely a nástrojové lišty</td>
<td></td>
</tr>
<tr>
<td>Panely</td>
<td></td>
<td>Panely a nástrojové lišty</td>
<td></td>
</tr>
<tr>
<td>Toggle Full Screen</td>
<td>F11</td>
<td>Zobrazit</td>
<td></td>
</tr>
<tr>
<td>Toggle Panel Visibility</td>
<td>Ctrl+Tab</td>
<td>Zobrazit</td>
<td></td>
</tr>
</tbody>
</table>

**Items menu**

The Items helps you configure items’ position in the layout and the relations between them (see Interacting with layout items).
Add Item menu

These are tools to create layout items. Each of them is deeply described in *Layout Items* chapter.
## Atlas menu

<table>
<thead>
<tr>
<th>Tool</th>
<th>Zkratka</th>
<th>Panel nástrojů</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview Atlas</td>
<td>Ctrl+ALT+/</td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>First Feature</td>
<td>Ctrl+&lt;</td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Previous Feature</td>
<td>Ctrl+&lt;,</td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Next Feature</td>
<td>Ctrl+.</td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Last feature</td>
<td>Ctrl+&gt;</td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Print Atlas…</td>
<td></td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Export Atlas as Images</td>
<td></td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Export Atlas as SVG…</td>
<td></td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Export Atlas as PDF…</td>
<td></td>
<td>Atlas</td>
<td>Preview and generate an atlas</td>
</tr>
<tr>
<td>Atlas Settings</td>
<td></td>
<td>Atlas</td>
<td>Generate an Atlas</td>
</tr>
</tbody>
</table>

### Settings Menu

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 Grid for more details);
- **Layout Paths**: to manage list of custom paths to search print templates.

### 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.

![Layout Panel Screenshot](image.png)

Obr. 17.3: 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 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 The Guides Panel 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).

Poznámka: 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.

Export settings

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.
Variables

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.

![Variables Editor in the Print Layout]

Obr. 17.4: 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.

Adding a new page

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:

- the number of pages to insert;
- 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.
Updating page properties

Any page can be later customized through the Page Item Properties panel. Right-click on a page and select Page Properties…. The Item Properties panel opens with settings such as:

- 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.

The Guides Panel

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.

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.

**Tip: 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.
The Items Panel

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.

For any selected 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 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 *locked* 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:

- Revert last change
- Restore last change

This can also be done by mouse click within the *Undo history* panel (see *figure_layout*). The History panel lists the last actions done within the print layout. Just select the point you want to revert to and once you do new action all the actions done after the selected one will be removed.

Obr. 17.8: Undo History in the Print Layout
17.2 Layout Items

17.2.1 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);

![New Item Properties dialog](image)

Obr. 17.9: 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.

**Poznámka:** 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 The Node-Based Shape Items 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 The Layout Manager) through the Layout Add Items from Template… command.

**Tip:** Add layout items using the file browser

---

QGIS 3.10 User Guide

Kapitola 17. Laying out the maps
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, just 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 form a square.

Likewise, automated tools are available to organize multiple items position by distributing equidistantly:

- edges (left, right, top or bottom) of items;
- centers of items either 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 View 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

Raising or lowering the visual hierarchy for elements 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 compared to 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.

There are several alignment options available within the Align selected items pull-down menu (see figure_layout_common_align). 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.
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 figure_layout_common).

Obr. 17.11: Common Item Properties groups

Poznámka: 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 Data defined override setup.

- 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).
- The Rotation sets the rotation of the element (in degrees).
- The Frame shows or hides the frame around the item. Use the Color, Thickness and Join style widgets to adjust those properties.
- Use the Background color menu for setting a background color. Click on the [Color...] button to display a dialog where you can pick a color or choose from a custom setting. Transparency can be adjusted through altering the alpha field settings.

17.2. Layout Items
• 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.

Obr. 17.12: 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 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.

Rendering mode

QGIS allows advanced rendering for layout items just like vector and raster layers.

Obr. 17.13: Rendering mode

• 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 Režim míchání for description of each effect).
• **Transparency**: 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**: You can decide to make an item invisible in all exports. After activating this checkbox, the item will not be included in export to PDF, print etc..

**Variables**

The Variables lists all the variables available at the layout item's level (which includes all global, project and composition’s variables). Map items also include Map settings variables that provide easy access to values like the map’s scale, extent, and so on.

In Variables, it’s also possible to manage 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.

**17.2.2 The Map Item**

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:

**The Toolbar**

The Map Item Properties panel embeds a toolbar with the following functionalities:

• ![Update map preview](image)
• ![Set map canvas to match main canvas extent](image)
• ![View current map extent in main canvas](image)
• ![Set map scale to match main canvas scale](image)
• ![Set main canvas to match current map scale](image)
• ![Bookmarks](image): set the map item extent to match an existing spatial bookmark
• ![Interactively edit map extent](image): pan and zoom interactively within the map item
• ![Labeling settings](image): control feature label behaviour (placement, visibility…) in the layout map item extent:
  - ![Set a Margin from map edges](image), a data definable distance from the map item’s limits inside which no label should be displayed
  - ![Allow truncated labels on edges of map](image): 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.
Obr. 17.14: Map Item Properties Panel
– **Label blocking items**: allows other layout items (such as scalebars, north arrows, inset maps, etc) to be marked as 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.

### Main properties

In the **Main properties** group (see figure_layout_map) 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;
- The **Draw map canvas items** lets you show in the print layout annotations that are placed on the main map canvas.

### Layers

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 figure_layout_map_layers) helps you do that.

![Obr. 17.15: Map Layers group](image)

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:

```plaintext
concat ('layer 1', '|', 'layer 2')
```

### Extents

The Extents group of the map item properties panel provides the following functionalities (see figure_layout_map_extents):

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. Clicking the Set to Map Canvas Extent button sets the extent of the layout map item to the extent of the main map canvas. The button View Extent in Map Canvas does exactly the opposite; it updates the extent of the main map canvas to the extent of the layout map item.

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.

### 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.
Grids

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.

Obr. 17.17: Map Grids Dialog

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. If not changed, it will follow the Map CRS. The Change button lets you set it to a different CRS. Once set, it can be changed back to default by selecting any group heading (e.g Geographic Coordinate System) under Predefined Coordinate Reference Systems in the CRS selection dialog.
- the Interval type to use for the grid references. Available options are Map Unit, Fit Segment Width, Millimeter or Centimeter:
  - choosing Fit Segment Width will dynamically select the grid interval based on the map extent to a „pretty” interval. When selected, the Minimum and Maximum intervals can be set.
  - the other options allow you to set the distance between two consecutive grid references in the X and Y directions.
- the Offset from the map item edges, in the X and/or the Y direction
- and the Blend mode of the grid (see Režim míchání) when compatible.
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.
Coordinates

The ✔️ Draw coordinates checkbox allows you to add coordinates to the map frame. 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.

You can choose which annotation to show. The options are: show all, latitude only, longitude only, or disable (none). This is useful when the map is rotated. The annotation can be drawn inside or outside the map frame. The annotation direction can be defined as horizontal, vertical ascending or vertical descending.

Finally, you can define the annotation font, font color, distance from the map frame and the precision of the drawn coordinates.

![Grid Draw Coordinates dialog](image)

Obr. 17.20: Grid Draw Coordinates dialog

Overviews

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:

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 Figure_layout_map_overview). 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.

Obr. 17.21: Map Overviews group
17.2.3 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 (Obr. 17.22):

- **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.

### 17.2.4 The Label Item

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 *figure_layout_label*):

![Obr. 17.23: Label Item Properties Panel](image)

**Main properties**

The *Main properties* group is the place to provide the text (it can be in HTML) or the expression to build the label. Expressions need to be surrounded by [% and %] in order to be interpreted as such.

- Labels can be interpreted as HTML code: check [Render as HTML](#). You can now insert a URL, a clickable image that links to a web page or something more complex.
- You can also use *expressions*: click on [Insert an expression](#) button, write your formula as usual and when the dialog is applied, QGIS automatically adds the surrounding characters.

**Poznámka:** Clicking the *Insert an Expression* button when no selection is made in the textbox will append the new expression to the existing text. If you want to update an existing text, you need to select it the part of interest beforehand.

You can combine HTML rendering and expressions, leading to e.g. a text like:

```html
[%("<b>Check out the new logo for "| | '<a href="https://www.qgis.org" title="Nice...-logo" target="blank">QGIS '</a>' || @qgis_short_version || '</a>') || %]
```

(continues on next page)
which will render: Check out the new logo for QGIS 3.0: 📚

**Appearance**

- Define *Font* by clicking on the *Font…* button or a *Font color* by pushing the *color widget*.
- 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 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...  
        layer_property( @element, 'name' ) -- retrieve each layer name  
      ),  
        "\n" -- converts the list to string separated by breaklines  
    )
  ```
17.2.5 The Legend Item

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 figure_layout_legend):

![Legend Item Properties Panel](Obr.17.24: Legend Item Properties Panel)

### Main properties

The Main properties group of the legend Item Properties panel provides the following functionalities (see figure_layout_legend_ppt):

![Legend Main properties group](Obr.17.25: Legend Main properties group)

In Main properties you can:

- 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.
Poznámka: *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.

**Legend items**

The *Legend items* group of the legend *Item Properties* panel provides the following functionalities (see *figure_layout_legend_items*):

![Legend Items](image)

**Obr. 17.26: 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 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.
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 categorized layer, you can append to each class in the legend their number of features, for class (number):

1. Select the layer entry in the legend tree
2. Press the \( \Sigma \) button, opening the Expression String Builder dialog
3. Enter the following expression:
   \[
   \text{concat}( \text{symbol}_\text{label}, \text{', ', symbol}_\text{count})
   \]
4. Press OK

- The \( \Sigma \) button adds a feature count for each class of vector layer.

- The \( \Sigma \) 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 \( \Sigma \) Only show items inside linked map, only the legend items visible in the linked map will be listed in the legend. This tool remains available when \( \Sigma \) 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 \( \Sigma \) Only show items inside current atlas feature option.

### Fonts

The Fonts group of the legend Item Properties panel provides the following functionalities:

- You can change the font of the legend title, group, subgroup and item (feature) in the legend item using the font selector widget.
- 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).
- You set the Color of the labels using the color selector widget. The selected color will apply to all the font items in the legend.
Obr. 17.27: Legend Fonts properties
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 field. This value can be made dynamic e.g., following atlas features, legend contents, the frame size…

- Equal column widths sets how legend columns should be adjusted.

- The Split layers option allows a categorized or a graduated layer legend to be divided between columns.

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

- 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 Thickness.
WMS LegendGraphic and Spacing

The WMS LegendGraphic and Spacing groups of the legend Item Properties panel provide the following functionalities (see figure_layout_legend_wms):

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 around title, groups, subgroups, symbols, labels, boxes, columns and lines can be customized through this dialog.

17.2.6 The Scale Bar Item

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 \textit{Add Scale Bar} tool following \textit{items creation instructions} to add a new scale bar item that you can later manipulate the same way as exposed in \textit{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 figure_layout_scalebar):

Main properties

The Main properties group of the scale bar Item Properties panel provides the following functionalities (see figure_layout_scalebar_ppt):

1. First, choose the map the scale bar will be attached to
2. Then, choose the style of the scale bar. Six styles are available:
   - Single box and Double box styles, which contain one or two lines of boxes alternating colors;
   - Middle, Up or Down line ticks;
   - Numeric, where the scale ratio is printed (e.g., 1:50000).
3. Set properties as appropriate

Units and Segments

The Units and Segments groups of the scale bar Item Properties panel (not available for the Numeric style) provide the following functionalities (see figure_layout_scalebar_units):

In these two groups, you can set how the scale bar will be represented.

- 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… which may force unit conversions.
- 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.
- You can define how many Segments will be drawn on the left and on the right side of the scale bar.
- You can set how long each segment will be (Fixed width), or limit the 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 the range set.
Obr. 17.30: WMS LegendGraphic and Spacing groups
Obr. 17.31: Scale Bar Item Properties Panel

Obr. 17.32: Scale Bar Main properties group

Obr. 17.33: Scale Bar Units and Segments groups
• *Height* is used to define the height of the bar.

**Display**

The *Display* group of the scale bar *Item Properties* panel provides the following functionalities (see *figure_layout_scalebar_display*):

You can define how the scale bar will be displayed in its frame.

- *Box margin*: space between text and frame borders
- *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
- *Line width*: line width of the scale bar drawing
- *Join style*: Corners at the end of scale bar in Bevel, Miter or Round style (only available for Scale bar style Single Box & Double Box)
- *Cap style*: End of all lines in style Square, Round or Flat (only available for Scale bar style Line Ticks Up, Down and Middle)
- *Alignment*: Puts text on the left, center or right side of the frame (works only for Scale bar style Numeric)

**Fonts and colors**

The *Fonts and colors* group of the scale bar *Item Properties* panel provides the following functionalities (see *figure_layout_scalebar_fonts*):

You can define the *fonts* and *colors* used for the scale bar. These properties are data-definable.

- Use the *Font* button to set the *properties* (size, font, color, letter spacing, shadow, background…) of the scale bar label.

Example: The following code applied to the bold property of the scale labels will display texts in bold when they are a multiple of 500:

```python
returns True (or 1) if the value displayed on the bar
is a multiple of 500
@scale_value % 500 == 0
```
17.2.7 The Attribute Table Item

Any layer in the project can have its attributes shown in the print layout. You can use this to decorate and explain your map with information about underlying data. Use the Add Attribute Table 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 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 figure_layout_table):

- **Fill color**: set the first fill color
- **Secondary fill color**: set the second fill color
- **Line color**: set the color of the lines of the Scale Bar

Fill colors are only used for *Single Box* and *Double Box* styles.

**Obr. 17.35: Scale Bar Fonts and colors groups**

**Obr. 17.36: Attribute table Item Properties Panel**
Main properties

The Main properties group of the attribute table provides the following functionalities (see figure_layout_table_ppt):

Obr. 17.37: 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 (Obr. 17.40) 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 Generate an Atlas), there are two additional Source possible:

- Current atlas feature (see figure_layout_table_atlas): 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 figure_layout_table_relation): 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.

Obr. 17.38: Attribute table Main properties for 'Current atlas feature'

Obr. 17.39: Attribute table Main properties for 'Relation children'

- The button Attributes… starts the Select Attributes dialog, (see figure_layout_table_select) 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.
In the **Columns** section you can:

- 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.

In the **Sorting** section you can:

- Add an attribute to sort the table with. Select an attribute and set the sorting order to **Ascending** or **Descending** and press the + button. A new line is added to the sort order list.

- 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 sort order list.
Feature filtering

The Feature filtering group of the attribute table provides the following functionalities (see figure_layout_table_filter):

![Feature Filtering](image)

Obr. 17.41: Attribute table Feature filtering Group

You can:

- Define the **Maximum rows** to be displayed.
- 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'.

Appearance

The Appearance group of the attribute table provides the following functionalities (see figure_layout_table_appearance):

- 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 figure *layout_table_background*).

Obr. 17.43: Attribute table Advanced Background Dialog

  • With the *Wrap text on* option, you can define a character on which the cell content will be wrapped 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*.  

17.2. Layout Items
Show grid

The Show grid group of the attribute table provides the following functionalities (see figure_layout_table_grid):

- 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.

Fonts and text styling

The Fonts and text styling group of the attribute table provides the following functionalities (see figure_layout_table_fonts):

- You can define Font and Color for Table heading and Table contents, using font and color selector widgets.
- 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 figure_layout_table_select).
Frames

The Frames group of the attribute table properties provides the following functionalities (see figure_layout_table_frames):

- 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.

17.2.8 The Picture and the North Arrow Items

The Picture item is a tool that helps decorate your map with pictures, logos… It can also be used to add north arrows, despite the dedicated North arrow tool.

The Picture Item

You can add a picture with the Add Picture following items creation instructions and manipulate it the same way as exposed in Interacting with layout items.

By default, the picture item is a blank frame that you can customize using its Item Properties panel. Other than the items common properties, this feature has the following functionalities (see figure_layout_image):

You first have to select the image you want to display. There are several ways to set the Image source:

1. In the Main properties group, use the … Browse button of image source to select a file on your computer. The browser will start in the SVG-libraries provided with QGIS. Besides SVG, you can also select other image formats like .png or .jpg.

2. You can enter the source directly in the Image source text field. You can even provide a remote URL-address to a picture.

3. From the Search directories area you can also select an image from the loaded previews to set the image source. These images are by default provided by folders set in Settings Options System SVG Paths.
4. Use the **data defined override** button to set the image source from a feature attribute or using a regular expression.

**Poznámka:** In the *Search directories* group, you can use the *Add* and *Remove* buttons in the group to customize the list of folders to fetch and preview images from.

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 aspect ratio of picture;
- **Stretch**: stretches image to fit inside the frame, ignores aspect ratio;
- **Clip**: use this mode for raster images only, it sets the size of the image to original image size without scaling and the frame is used to clip the image, so only the part of the image inside the frame is visible;
- **Zoom and resize frame**: enlarges image to fit frame, then resizes frame to fit resultant image;
- **Resize frame to image size**: sets size of frame to match original size of image without scaling.

Depending on the selected *Resize mode*, the *Placement* and *Image rotation* options are disabled or not. With *Placement* you can select the position of the image inside its frame.

QGIS default provided `.SVG` files are customizable, meaning that you can easily apply different *Fill color*, *Stroke color* (including opacity) and *Stroke width* than the original, using their corresponding feature in the *SVG Parameters* group. These properties can as well be **data-defined**.

If you add a `.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)"`

You can read this blog post to see an example.
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 a selected map item; this is a convenient feature for north arrows that you can align with either:

- the **Grid north**: direction of a grid line which is parallel to the central meridian on the national/local grid;
- or the **True north**: direction of a meridian of longitude which converges on the North Pole.

You can also apply a declination *Offset* to the picture rotation.

### The North Arrow Item

You can add a north arrow with the *Add North Arrow* 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 filled with the map over which the north arrow item is drawn. If none, then it falls back to the *reference map*.

**Poznámka:** Many of the north arrows do not have an ‘N’ added in the north arrow, this is done on purpose for languages that do not use an ‘N’ for North, so they can use another letter.

![North Arrow Selection](Obr. 17.48: North arrows available for selection in provided SVG library)
17.2.9 The HTML Frame Item

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.

The HTML item can be customized using its Item Properties panel. Other than the items common properties, this feature has the following functionalities (see figure_layout_html):

![HTML frame properties](image)

Obr. 17.49: HTML Frame, the Item Properties Panel

**HTML Source**

The HTML Source group of the HTML frame Item Properties panel provides the following functionalities (see figure_layout_html_ppt):

![HTML Source properties](image)

Obr. 17.50: HTML frame, the HTML Source properties

- 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.

- In Source you can enter text in the textbox with some HTML tags or provide a full HTML page.

- The Insert an Expression button can be used to insert 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.
• Activate Evaluate QGIS expressions in HTML code to see the result of the expression you have included, otherwise you will see the expression instead.

• Use the Refresh HTML button to refresh the HTML frame(s) and see the result of changes.

Frames

The Frames group of the HTML frame Item Properties panel provides the following functionalities (see figure_layout_html_frames):

![Frames panel](image)

Obr. 17.51: HTML frame, the Frames properties

• With Resize mode you can select how to render the HTML 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 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.

• Activate Don’t draw background if frame is empty prevents the HTML frame being drawn if the frame is empty.

Use smart page breaks and User style sheet

The Use smart page breaks dialog and User style sheet dialog of the HTML frame Item Properties panel provides the following functionalities (see figure_layout_html_breaks):

• Activate Use smart page breaks to prevent the html frame contents from breaking mid-way a line of text so it continues nice and smooth in the next frame.

• 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>.
17.2.10 The Shape Items

QGIS provides a couple of tools to draw regular or more complex shapes over the print layout.

**Poznámka:** 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* and *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*.

**Poznámka:** 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 figure *layout_label*):

1. The *Main properties* group shows and allows you to switch the type of the shape item (*Ellipse*, *Rectangle* or *Triangle*) inside the given frame.

2. You can set the style of the shape using the advanced *symbol* and *color* selector widget…

3. For the rectangle shape, you can set in different units the value of the *Corner radius* to round of the corners.

---

Obr. 17.52: HTML frame, Use smart page breaks and User style sheet properties

```html
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.
```
The Node-Based Shape Items

While the Add Shape tool provides a way to create simple and predefined geometric items, 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.

To add a node-based shape:

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.

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.

![Arrow Properties Panel](image)

Obr. 17.55: Polyline Node Shape Item Properties Panel

**The Arrow Item**

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

A specific tool is provided to edit node-based shapes through Edit Nodes Item. Within this mode, you can select a node by clicking on it (a marker is displayed on the selected node). A selected node can be moved either by dragging it or by using the arrow keys. Moreover, in this mode, you are able to add nodes to an existing shape: double-click on a segment and a node is added at the place you click. Finally, you can remove the currently selected node by hitting the Del key.

17.3 Creating an Output

Obr. 17.56 shows an example print layout including all the types of layout items described in the previous section.

Obr. 17.56: 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.
17.3.1 Export settings

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

17.3.2 Export as Image

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.

**Poznámka:** 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.

**Tip:** 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.
17.3.3 Export as 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 Export settings).

   • 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.

![SVG Export Options](image)

Obr. 17.58: SVG Export Options

**Poznámka**: Currently, the SVG output is very basic. This is not a QGIS problem, but a problem with the underlying Qt library. This will hopefully be sorted out in future versions.

### 17.3.4 Export as PDF

To export a layout as PDF:

1. Click the ![Export as PDF](image) 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 **Export settings**).
   • **Append georeference information**
   • **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.
   • **Create Geospatial PDF (GeoPDF)**: Generate a georeferenced PDF file (requires GDAL version 3 or later).
   • **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.

456 Kapitola 17. Laying out the maps
• **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).

![PDF Export Options]

Obr. 17.59: PDF Export Options

**Poznámka**: Since QGIS 3.10, with GDAL 3, 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).

**Poznámka**: Exporting a print layout to formats that supports georeferencing (e.g. PDF and TIFF) creates a georeferenced output by default.

### 17.3.5 Generate an 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.

**Tip**: 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 figure_layout_atlas):

- **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 The Label Item 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
```

or, another combination:

```
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 Data defined override setup 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:

```plaintext
@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.

This is just one example of how you can use some advanced settings with atlas.

**Preview and generate an atlas**

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:

- First feature
- Previous feature
- Next feature
- Last feature

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 Creating an Output 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.

**Poznámka:** 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).

**Tip:** Print a specific atlas feature

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.

```javascript
// 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 = '';
    // For each related child feature, create a list element
    feature.properties.my_relation.forEach(function(child_feature) {
      // 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.

### 17.4 Creating a Report

This section will help you set up a report in QGIS.

#### 17.4.1 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.

**Poznámka:** 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.
17.4.2 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:

17.4.3 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
17.4. Creating a Report

![Image of QGIS Report Organizer and layout settings](image-url)
Kapitola 17. Laying out the maps
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 and data defined the text under Main Properties with the help of Insert 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:

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
17.4. Creating a Report
Kapitola 17. Laying out the maps
17.4. Creating a Report

- Main Properties
- Layers
- Extents
- Controlled by Report
  - Margin around feature
  - Predefined scale (best fit)
  - Fixed scale
- Grids
- Overviews
- Position and Size
- Rotation
- Frame
- Background
- Item ID
- Rendering
- Variables
Obr. 17.62: The report header, a page for each state, and the report footer.
Creating a Report

The Report Organizer allows you to create reports for your QGIS project. You can include headers, layers, fields, and sorting options. The example shown includes a group for "Admin Level 1 - adm1name" and another for "Populated Places - NAME". The layer selected is "Populated Places" and the field used is "NAME". The report includes the option to show the header when no matching features are found and to sort the data in ascending order. The body of the report is also included.
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):

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 (adminname) from which the file name is generated (in this case by transforming the string in the adminname attribute to uppercase, and appending _flag.png):

| concat(@project_folder, '/naturalearth/pictures/', upper("adminname"), '_flag.png') |

This means that the pictures are located in the naturalearth/pictures subdirectory of the project file directory.
17.4. Creating a Report
Kapitola 17. Laying out the maps
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:

17.4.4 Export settings

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!

**Poznámka:** The current information was adapted from a North Road blog, Exploring Reports in QGIS 3.0 - the Ultimate Guide!
17.4. Creating a Report

Mexican ports

Port: Campeche

Port: Coatzacoalcos

Port: Cozumel

Port: Frontera

Port: La Paz
18.1 QGIS as OGC Data Client

The Open Geospatial Consortium (OGC) is an international organization with membership of more than 300 commercial, governmental, nonprofit and research organizations worldwide. Its members develop and implement standards for geospatial content and services, GIS data processing and exchange.

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.opengeospatial.org/.

Important OGC specifications supported by QGIS are:

- **WMS** — Web Map Service (*WMS/WMTS Client*)
- **WMTS** — Web Map Tile Service (*WMS/WMTS Client*)
- **WFS** — Web Feature Service (*WFS and WFS-T Client*)
- **WFS-T** — Web Feature Service - Transactional (*WFS and WFS-T Client*)
- **WCS** — Web Coverage Service (*WCS Client*)
- **WPS** — Web Processing Service
- **CSW** — Catalog Service for the Web
- **SFS** — Simple Features for SQL (*PostGIS Layers*)
- **GML** — Geography Markup Language

OGC services are increasingly being used to exchange geospatial data between different GIS implementations and data stores. QGIS can deal with the above specifications as a client, being **SFS** (through support of the PostgreSQL / PostGIS data provider, see section *PostGIS Layers*).
18.1.1 WMS/WMTS Client

Overview of WMS Support

QGIS currently can act as a WMS client that understands WMS 1.1, 1.1.1 and 1.3 servers. In particular, it has been tested against publicly accessible servers such as DEMIS.

A WMS server acts upon requests by the client (e.g., QGIS) for a raster map with a given extent, set of layers, symbolization style, and transparency. The WMS server then consults its local data sources, rasterizes the map, and sends it back to the client in a raster format. For QGIS, this format would typically be JPEG or PNG.

WMS is generically a REST (Representational State Transfer) service rather than a full-blown Web service. As such, you can actually take the URLs generated by QGIS and use them in a web browser to retrieve the same images that QGIS uses internally. This can be useful for troubleshooting, as there are several brands of WMS server on the market and they all have their own interpretation of the WMS standard.

WMS layers can be added quite simply, as long as you know the URL to access the WMS server, you have a serviceable connection to that server, and the server understands HTTP as the data transport mechanism.

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 everytime the Connect button in the Add Layer(s) from WMS(T) Server 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 no older than 24H.

Overview of WMTS Support

QGIS can also act as a WMTS client. WMTS is an OGC standard for distributing tile sets of geospatial data. This is a faster and more efficient way of distributing data than WMS because with WMTS, the tile sets are pre-generated, and the client only requests the transmission of the tiles, not their production. A WMS request typically involves both the generation and transmission of the data. A well-known example of a non-OGC standard for viewing tiled geospatial data is Google Maps.

In order to display the data at a variety of scales close to what the user might want, the WMTS tile sets are produced at several different scale levels and are made available for the GIS client to request them.

This diagram illustrates the concept of tile sets:

![WMTS Concept Diagram]

Obr. 18.1: Concept of WMTS tile sets

The two types of WMTS interfaces that QGIS supports are via Key-Value-Pairs (KVP) and RESTful. These two interfaces are different, and you need to specify them to QGIS differently.

1. In order to access a WMTS KVP service, a QGIS user must open the WMS/WMTS interface and add the following string to the URL of the WMTS tile service:
An example of this type of address is

```plaintext
https://opencache.statkart.no/gatekeeper/gk/gk.open_wmts?service=WMTS&request=GetCapabilities
```

For testing the topo2 layer in this WMTS works nicely. Adding this string indicates that a WMTS web service is to be used instead of a WMS service.

2. The **RESTful WMTS** service takes a different form, a straightforward URL. The format recommended by the OGC is:

```plaintext
{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`.

**Poznámka:** 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](https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification) for more information about this specification.

When you read WMTS, you can often think WMS-C also.

### Selecting WMS/WMTS Servers

The first time you use the WMS feature in QGIS, there are no servers defined.

Begin by clicking the `Add WMS layer` button on the toolbar, or selecting `Layer  Add WMS Layer…`

The dialog `Add Layer(s) from a Server` for adding layers from the WMS server appears. You can add some servers to play with by clicking the `Add Default Servers` button. This will add two WMS demo servers for you to use: the WMS servers of the DM Solutions Group and Lizardtech. To define a new WMS server in the `Layers` tab, select the `New` button. Then enter the parameters to connect to your desired WMS server, as listed in `table_OGC_wms`:

<table>
<thead>
<tr>
<th>Název</th>
<th>A name for this connection. This name will be used in the Server Connections drop-down box so that you can distinguish it from other WMS servers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>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.</td>
</tr>
<tr>
<td>Jméno uživatele</td>
<td>Username to access a secured WMS server. This parameter is optional.</td>
</tr>
<tr>
<td>Heslo</td>
<td>Password for a basic authenticated WMS server. This parameter is optional.</td>
</tr>
<tr>
<td>Ignore GetMap URI</td>
<td><a href="#">Ignore GetMap URI reported in capabilities. Use given URI from URL field above.</a></td>
</tr>
<tr>
<td>Ignore GetFeatureInfo URI</td>
<td><a href="#">Ignore GetFeatureInfo URI reported in capabilities. Use given URI from URL field above.</a></td>
</tr>
</tbody>
</table>

Table OGC 1: WMS Connection Parameters

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 & Proxy` 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.

Once the new WMS server connection has been created, it will be preserved for future QGIS sessions.
Tip: On WMS Server URLs

Be sure, when entering the WMS server URL, that you have the base URL only. For example, you shouldn’t have fragments such as request=GetCapabilities or version=1.0.0 in your URL.

Varování: 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 Ověřovací systém for more details.

Loading WMS/WMTS Layers

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 of the WMS dialog.

Your screen should now look a bit like figure_OGC_add_wms, which shows the response provided by the European Soil Portal WMS server.

Obr. 18.2: Dialog for adding a WMS server, showing its available layers

Image Encoding
The *Image encoding* section lists the formats that are supported by both the client and server. Choose one depending on your image accuracy requirements.

**Tip: Image Encoding**

You will typically find that a WMS server offers you the choice of JPEG or PNG image encoding. JPEG is a lossy compression format, whereas PNG faithfully reproduces the raw raster data.

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.

Use PNG if you want precise representations of the original data and you don't mind the increased data transfer requirements.

**Options**

The Options area of the dialog provides a text field where you can add a *Layer name* for the WMS layer. This name will appear in the legend after loading the layer.

Below the layer name, you can define *Tile size* if you want to set tile sizes (e.g., 256x256) to split up the WMS request into multiple requests.

The *Feature limit for GetFeatureInfo* defines what features from the server to query.

If you select a WMS from the list, a field with the default projection provided by the mapserver appears. If the *Change…* button is active, you can click on it and change the default projection of the WMS to another CRS provided by the WMS server.

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 things you can't see in the current map.

**Layer Order**

The *Layer Order* tab lists the selected layers available from the current connected WMS server. You may notice that some layers are expandable; this means that the layer can be displayed in a choice of image styles.

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.

**Tip: WMS Layer Ordering**

WMS layers rendered by a server are overlaid in the order listed in the Layers section, from top to bottom of the list. If you want to change the overlay order, you can use the *Layer Order* tab.

**Transparency**

In this version of QGIS, the *Global transparency* setting from the Layer Properties is hard coded to be always on, where available.

**Tip: WMS Layer Transparency**

The availability of WMS image transparency depends on the image encoding used: PNG and GIF support transparency, whilst JPEG leaves it unsupported.

**Coordinate Reference System**

A coordinate reference system (CRS) is the OGC terminology for a QGIS projection.

Each WMS layer can be presented in multiple CRSs, depending on the capability of the WMS server.

To choose a CRS, select *Change…* and a dialog similar to the one shown in Obr. 10.3 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.
Server search

Within QGIS, you can search for WMS servers. *Figure_OGC_search* shows the *Server Search* tab with the *Add Layer(s) from a Server* dialog.

![Figure_OGC_search](image)

Obr. 18.3: Dialog for searching WMS servers after some keywords

As you can see, it is possible to enter a search string in the text field and hit the *Search* button. After a short while, the search result will be populated into the list below the text field. Browse the result list and inspect your search results within the table. To visualize the results, select a table entry, press the *Add Selected Row to WMS List* button and change back to the *Layers* tab. QGIS has automatically updated your server list, and the selected search result is already enabled in the list of saved WMS servers in the *Layers* tab. You only need to request the list of layers by clicking the *Connect* button. This option is quite handy when you want to search maps by specific keywords.

Basically, this option is a front end to the API of [http://geopole.org](http://geopole.org).

Sady dlaždic

When using WMTS (Cached WMS) services like

```
https://opencache.statkart.no/gatekeeper/gk/open_wmts?\n    service=WMTS&request=GetCapabilities
```

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. 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.
Using the Identify Tool

Once you have added a WMS server, and if any layer from a WMS server is queryable, you can then use the Identify tool to select a pixel on the map canvas. A query is made to the WMS server for each selection made. The results of the query are returned in plain text. The formatting of this text is dependent on the particular WMS server used.

Format selection

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.

GML format support

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.

```plaintext
# 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 send as multipart and contains-
→ XSD)
# in MAP define OUTPUTFORMAT (example):
OUTPUTFORMAT
  NAME "OGRGML"
  MIME_TYPE "ogr/gml"
  DRIVER "OGR/GML"
  FORMAT_OPTION "FORM=multipart"
END

# in WEB METADATA define formats (example):
"wms_getfeatureinfo_formatlist" "OGRGML,text/html"

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.

Metadata Tab

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 Literature and Web References), but here are a few handy definitions:

- **Server Properties**
  - **WMS Version** — The WMS version supported by the server.
  - **Image Formats** — The list of MIME-types the server can respond with when drawing the map. QGIS supports whatever formats the underlying Qt libraries were built with, which is typically at least `image/png` and `image/jpeg`.

18.1. QGIS as OGC Data Client 485
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.

Layer Properties

- **Selected** — Whether or not this layer was selected when its server was added to this project.
- **Visible** — Whether or not this layer is selected as visible in the legend (not yet used in this version of QGIS).
- **Can Identify** — Whether or not this layer will return any results when the Identify tool is used on it.
- **Can be Transparent** — Whether or not this layer can be rendered with transparency. This version of QGIS will always use transparency if this is `Yes` and the image encoding supports transparency.
- **Can Zoom In** — Whether or not this layer can be zoomed in by the server. This version of QGIS assumes all WMS layers have this set to `Yes`. Deficient layers may be rendered strangely.
- **Cascade Count** — WMS servers can act as a proxy to other WMS servers to get the raster data for a layer. This entry shows how many times the request for this layer is forwarded to peer WMS servers for a result.
- **Fixed Width, Fixed Height** — Whether or not this layer has fixed source pixel dimensions. This version of QGIS assumes all WMS layers have this set to nothing. Deficient layers may be rendered strangely.
- **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`.
- **Available in CRS** — The projections that this layer can be rendered in by the WMS server. These are listed in the WMS-native format.
- **Available in style** — The image styles that this layer can be rendered in by the WMS server.

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.
WMS Client Limitations

Not all possible WMS client functionality had been included in this version of QGIS. Some of the more noteworthy exceptions follow.

Editing WMS Layer Settings

Once you’ve completed the Add WMS layer procedure, there is no way to change the settings. A work-around is to delete the layer completely and start again.

WMS Servers Requiring Authentication

Currently, publicly accessible and secured WMS services are supported. The secured WMS servers can be accessed by public authentication. You can add the (optional) credentials when you add a WMS server. See section Selecting WMS/WMTS Servers for details.

Tip: Accessing secured OGC-layers

If you need to access secured layers with secured methods other than basic authentication, you can use InteProxy as a transparent proxy, which does support several authentication methods. More information can be found in the InteProxy manual at https://inteproxy.wald.intevation.org.

Tip: QGIS WMS Mapserver

Since Version 1.7.0, QGIS has its own implementation of a WMS 1.3.0 Mapserver. Read more about this in chapter QGIS as OGC Data Server.

18.1.2 WCS Client

A Web Coverage Service (WCS) provides access to raster data in forms that are useful for client-side rendering, as input into scientific models, and for other clients. The WCS may be compared to the WFS and the WMS. As WMS and WFS service instances, a WCS allows clients to choose portions of a server’s information holdings based on spatial constraints and other query criteria.

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).

The native WCS provider handles all network requests and uses all standard QGIS network settings (especially proxy). It is also possible to select cache mode (‘always cache’, ‘prefer cache’, ‘prefer network’, ‘always network’), and the provider also supports selection of time position, if temporal domain is offered by the server.

Varování: 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 Ověřovací systém for more details.
18.1.3 WFS and WFS-T Client

In QGIS, a WFS layer behaves pretty much like any other vector layer. You can identify and select features, and view the attribute table. Since QGIS 1.6, editing WFS-T is also supported.

In general, adding a WFS layer is very similar to the procedure used with WMS. The difference is that there are no default servers defined, so we have to add our own.

**Loading a WFS Layer**

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 we create a connection to the WFS server first.

1. Open the Data Source Manager dialog by pressing the Open Data Source Manager button
2. Enable the WFS 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)

![Create a New WFS Connection](image)

Obr. 18.4: Creating a connection to a WFS server

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.
• Invert axis orientation.

• And depending on the WFS version:
  – Force to Ignore axis orientation (WFS 1.1/WFS 2.0)
  – 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.

**Varování:** 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 *Ověřovací systém* 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.

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**

   • or **Build query** to specify particular features to retrieve, by either using the corresponding button or double-clicking the target layer.

5. Click **Add** to add the layer to the map.

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.

**Poznámka:** QGIS supports different versions of the WFS protocol, with background download and progressive rendering, on-disk caching of downloaded features and version autodetection.

**Tip:** Finding WFS Servers

You can find additional WFS servers by using Google or your favourite search engine. There are a number of lists with public URLs, some of them maintained and some not.
Obr. 18.5: Adding a WFS layer
18.2 QGIS as OGC Data Server

QGIS Server is an open source WMS 1.3, WFS 1.0.0, WFS 1.1.0 and WCS 1.1.1 implementation that, in addition, implements advanced cartographic features for thematic mapping. QGIS Server is a FastCGI/CGI (Common Gateway Interface) application written in C++ that works together with a web server (e.g., Apache, Nginx). It has Python plugin support allowing for fast and efficient development and deployment of new features.

QGIS Server uses QGIS as back end for the GIS logic and for map rendering. Furthermore, the Qt library is used for graphics and for platform-independent C++ programming. In contrast to other WMS software, the QGIS Server uses cartographic rules as a configuration language, both for the server configuration and for the user-defined cartographic rules.

As QGIS desktop and QGIS Server use the same visualization libraries, the maps that are published on the web look the same as in desktop GIS.

In the following sections, we will provide a sample configuration to set up a QGIS Server on Linux (Debian, Ubuntu and derivatives) and on Windows. For more information about server plugin development, please read server_plugins.

18.2.1 Začínáme

Installation on Debian-based systems

We will give a short and simple installation how-to for a minimal working configuration on Debian based systems (including Ubuntu and derivatives). However, many other distributions and OSs provide packages for QGIS Server.

Requirements and steps to add official QGIS repositories to install current QGIS Server on a Debian based system are provided in QGIS installers page.

Poznámka: In Ubuntu you can use your regular user, prepending `sudo` to commands requiring admin permissions. In Debian you can work as admin (`root`), without using `sudo`.

We strongly suggest installing the LTR version.

Once the chosen repository is configured, installation is simply done with:

```
apt install qgis-server
# if you want to install server plugins, also:
apt install python-qgis
```

You can test the installation by running:

```
/usr/lib/cgi-bin/qgis_mapserv.fcgi
```

If you get the following output, the server is correctly installed:

```
QFSFileEngine::open: No file name specified
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment variable.
Content-Length: 206
Content-Type: text/xml; charset=utf-8
<ServiceExceptionReport version="1.3.0" xmlns="https://www.opengis.net/ogc">
  <ServiceException code="Service configuration error">Service unknown or unsupported</ServiceException>
</ServiceExceptionReport>
```
Let’s add a sample project. You can use your own, or one from Training demo data:

```bash
mkdir /home/qgis/projects/
cd /home/qgis/projects/
wget https://github.com/qgis/QGIS-Training-Data/archive/v2.0.zip
unzip v2.0.zip
mv QGIS-Training-Data-2.0/exercise_data/qgis-server-tutorial-data/world.qgs .
mv QGIS-Training-Data-2.0/exercise_data/qgis-server-tutorial-data/naturalearth.sqlite .
```

Of course, you can use your favorite GIS software to open this file and take a look on the configuration and available layers.

**HTTP Server configuration**

To run QGIS server you need a web server. Recommended choices are **Apache** or **Nginx**.

**Poznámka:** In the following, please replace localhost with the name or IP address of your server.

### Apache

Install Apache and **mod_fcgid**:

```bash
apt install apache2 libapache2-mod-fcgid a2enmod cgi
```

QGIS Server is now available at [http://localhost/](http://localhost/). To check, type in a browser:

```
http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
```

If you get something like:

```
<WMS_Capabilities version="1.3.0" xsi:schemaLocation="http://www.opengis.net/wms...
```

the server is correctly installed and responds through Apache.

Let’s now add **mod_fcgid** configuration directives for QGIS Server:

```bash
# Tell QGIS Server instances to use a specific display number for xvfb
# necessary for printing, see below
FcgidInitialEnv DISPLAY ":99"
# Activate QGIS log (different from apache logs)
FcgidInitialEnv QGIS_SERVER_LOG_FILE /var/log/qgis/qgisserver.log
FcgidInitialEnv QGIS_SERVER_LOG_LEVEL "0"
FcgidInitialEnv QGIS_SERVER_LOG_STDERR "1"
FcgidInitialEnv QGIS_DEBUG 1
# Add a default QGIS project
SetEnv QGIS_PROJECT_FILE /home/qgis/projects/world.qgs
# QGIS_AUTH_DB_DIR_PATH must lead to a directory writeable by www-data
FcgidInitialEnv QGIS_AUTH_DB_DIR_PATH "/var/www/qgis-server/qgisserverdb/
FcgidInitialEnv QGIS_AUTH_PASSWORD_FILE "/var/www/qgis-server/qgisserverdb/qgis-auth.db"
```

(continues on next page)
<IfModule mod_fcgid.c>
# Longer timeout for WPS... default = 40
FcgidIOTimeout 120
FcgidMaxRequestLen 26214400
FcgidConnectTimeout 60
</IfModule>

Poznámka: See the mod_fcgid documentation for more information on the Fcgid parameters used. And see below (xvfb) to understand when and why the DISPLAY environment variable needs to be set.

These directives can be added either to /etc/apache2/mods-enabled/fcgid.conf for a system-wide configuration, or to a specific Apache VirtualHost, if you want QGIS server to be available only for that address; the default one is available at /etc/apache2/sites-available/000-default.conf.

<Location /qgisserver>
SetHandler fcgid-script
FcgidWrapper /usr/lib/cgi-bin/qgis_mapserv.fcgi virtual
Options +ExecCGI -MultiViews +FollowSymLinks
Require all granted
</Location>

Then create all the needed directories with appropriate permissions:

```bash
mkdir -p /var/log/qgis/
chown www-data:www-data /var/log/qgis
mkdir -p /var/www/qgis-server/qgisserverdb/
chown www-data:www-data /var/www/qgis-server/qgisserverdb/
```

Now restart Apache for the new configuration to be taken into account:

```bash
systemctl restart apache2
```

QGIS Server is now available at http://localhost/qgisserver. To check, type in a browser, as in the simple case:

http://localhost/qgisserver/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities

**NGINX**

You can also use QGIS Server with NGINX. Unlike Apache, NGINX does not automatically spawn FastCGI processes. The FastCGI processes are to be started by something else.

On Debian-based systems, you can use `spawn-fcgi` or `fcgiwrap` to start and manage the QGIS Server processes. Official Debian packages exist for both.

Varování: fcgiwrap is easier to set up than spawn-fcgi, because it’s already wrapped in a Systemd service. But it also leads to a solution that is much slower than using spawn-fcgi. With fcgiwrap a new QGIS Server process is created on each request, meaning that the QGIS Server initialization process, which includes reading and parsing the QGIS project file, is done on each request. With spawn-fcgi, the QGIS Server process remains alive between requests, resulting in much better performance. For that reason, spawn-fcgi is recommended for production use.

Another option is to rely on Systemd, the init system for GNU/Linux that most Linux distributions use today. One of the advantages of this method is that it requires no other components or processes. It’s meant to be simple, yet robust and efficient for production deployments.
Install NGINX:

```
apt install nginx
```

### spawn-fcgi

If you want to use spawn-fcgi, the first step is to install the package:

```
apt install spawn-fcgi
```

Then, introduce the following block in your NGINX server configuration:

```
location /qgisserver {
  gzip off;
  include fastcgi_params;
  fastcgi_pass unix:/var/run/qgisserver.socket;
}
```

And restart NGINX to take into account the new configuration:

```
service nginx restart
```

Finally, considering that there is no default service file for spawn-fcgi, you have to manually start QGIS Server in your terminal:

```
spawn-fcgi -s /var/run/qgisserver.socket \
  -U www-data -G www-data -n \
  /usr/lib/cgi-bin/qgis_mapserv.fcgi
```

todo: Add instructions to add a spawn-fcgi.service

QGIS Server is now available at http://localhost/qgisserver.

---

**Poznámka:** With the above command spawn-fcgi spawns only one QGIS Server process. To use more than one QGIS Server process you can combine spawn-fcgi with the multiwatch tool, which is also packaged in Debian.

Of course, you can add an init script (like a qgis-server.service file with systemd) to start QGIS Server at boot time or whenever you want.

todo: Add instructions to add a qgis-server.service

### fcgiwrap

Using fcgiwrap is much easier to setup than spawn-fcgi but it's much slower. You first have to install the corresponding package:

```
apt install fcgiwrap
```

Then, introduce the following block in your NGINX server configuration:

```
location /qgisserver {
  gzip off;
  include fastcgi_params;
  fastcgi_pass unix:/var/run/fcgiwrap.socket;
  fastcgi_param SCRIPT_FILENAME /usr/lib/cgi-bin/qgis_mapserv.fcgi;
}
```

Finally, restart NGINX and fcgiwrap to take into account the new configuration:
service nginx restart
service fcgiwrap restart

QGIS Server is now available at http://localhost/qgisserver.

**Configuration**

The **include fastcgi_params** used in previous configuration is important as it adds the parameters from `/etc/nginx/fastcgi_params`:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY_STRING</td>
<td>$query_string</td>
</tr>
<tr>
<td>REQUEST_METHOD</td>
<td>$request_method</td>
</tr>
<tr>
<td>CONTENT_TYPE</td>
<td>$content_type</td>
</tr>
<tr>
<td>CONTENT_LENGTH</td>
<td>$content_length</td>
</tr>
<tr>
<td>SCRIPT_NAME</td>
<td>$fastcgi_script_name</td>
</tr>
<tr>
<td>REQUEST_URI</td>
<td>$request_uri</td>
</tr>
<tr>
<td>DOCUMENT_URI</td>
<td>$document_uri</td>
</tr>
<tr>
<td>DOCUMENT_ROOT</td>
<td>$document_root</td>
</tr>
<tr>
<td>SERVER_PROTOCOL</td>
<td>$server_protocol</td>
</tr>
<tr>
<td>REQUEST_SCHEME</td>
<td>$scheme</td>
</tr>
<tr>
<td>DOCUMENT_URI</td>
<td>$document_uri</td>
</tr>
<tr>
<td>DOCUMENT_ROOT</td>
<td>$document_root</td>
</tr>
<tr>
<td>SERVER_PROTOCOL</td>
<td>$server_protocol</td>
</tr>
<tr>
<td>REQUEST_SCHEME</td>
<td>$scheme</td>
</tr>
<tr>
<td>SERVER_ADDR</td>
<td>$server_addr</td>
</tr>
<tr>
<td>SERVER_PORT</td>
<td>$server_port</td>
</tr>
<tr>
<td>SERVER_NAME</td>
<td>$server_name</td>
</tr>
<tr>
<td>GATEWAY_INTERFACE</td>
<td>CGI/1.1</td>
</tr>
<tr>
<td>SERVER_SOFTWARE</td>
<td>nginx/$nginx_version</td>
</tr>
<tr>
<td>REMOTE_ADDR</td>
<td>$remote_addr</td>
</tr>
<tr>
<td>REMOTE_PORT</td>
<td>$remote_port</td>
</tr>
<tr>
<td>SERVER_ADDR</td>
<td>$server_addr</td>
</tr>
<tr>
<td>SERVER_PORT</td>
<td>$server_port</td>
</tr>
<tr>
<td>SERVER_NAME</td>
<td>$server_name</td>
</tr>
<tr>
<td>REDIRECT_STATUS</td>
<td>200</td>
</tr>
</tbody>
</table>

Of course, you may override these variables in your own configuration. For example:

```bash
include fastcgi_params;
fastcgi_param SERVER_NAME domain.name.eu;
```

Moreover, you can use some **Environment variables** to configure QGIS Server. With NGINX as HTTP Server, you have to use `fastcgi_param` to define these variables as shown below:

```bash
fastcgi_param QGIS_DEBUG 1;
fastcgi_param QGIS_SERVER_LOG_FILE /var/log/qgis/qgisserver.log;
fastcgi_param QGIS_SERVER_LOG_LEVEL 0;
```

**Poznámka:** When using spawn-fcgi, you may directly define environment variables before running the server. For example: `export QGIS_SERVER_LOG_FILE=/var/log/qgis/qgisserver.log`
This method to deploy QGIS Server relies on two Systemd units:

- a **Socket unit**
- and a **Service unit**.

The **QGIS Server Socket unit** defines and creates a file system socket, used by NGINX to start and communicate with QGIS Server. The Socket unit has to be configured with `Accept=false`, meaning that the calls to the `accept()` system call are delegated to the process created by the Service unit. It is located in `/etc/systemd/system/qgis-server@.socket`, which is actually a template:

```ini
[Unit]
Description=QGIS Server Listen Socket (instance %i)

[Socket]
Accept=false
ListenStream=/var/run/qgis-server-%i.sock
SocketUser=www-data
SocketGroup=www-data
SocketMode=0600

[Install]
WantedBy=sockets.target
```

Now enable and start sockets:

```
sudo systemctl enable qgis-server@1.socket
sudo systemctl start qgis-server@1.socket
sudo systemctl enable qgis-server@2.socket
sudo systemctl start qgis-server@2.socket
sudo systemctl enable qgis-server@3.socket
sudo systemctl start qgis-server@3.socket
sudo systemctl enable qgis-server@4.socket
sudo systemctl start qgis-server@4.socket
```

The **QGIS Server Service unit** defines and starts the QGIS Server process. The important part is that the Service process’ standard input is connected to the socket defined by the Socket unit. This has to be configured using `StandardInput=socket` in the Service unit configuration located in `/etc/systemd/system/qgis-server@.service`:

```ini
[Unit]
Description=QGIS Server Service (instance %i)

[Service]
User=www-data
Group=www-data
StandardOutput=null
StandardError=journal
StandardInput=socket
ExecStart=/usr/lib/cgi-bin/qgis_mapserv.fcgi
EnvironmentFile=/etc/qgis-server/env

[Install]
WantedBy=multi-user.target
```

Now start socket service:

```
sudo systemctl start qgis-server@sockets.service
```

Note that the QGIS Server **environment variables** are defined in a separate file, `/etc/qgis-server/env`. It could look like this:
Finally, introduce the NGINX configuration for this setup:

```bash
upstream qgis-server_backend {
  server unix:/var/run/qgis-server-1.sock;
  server unix:/var/run/qgis-server-2.sock;
  server unix:/var/run/qgis-server-3.sock;
  server unix:/var/run/qgis-server-4.sock;
}

server {
  location /qgis {
    gzip off;
    include fastcgi_params;
    fastcgi_pass qgis-server_backend;
  }
}
```

Now restart NGINX for the new configuration to be taken into account:

```bash
service nginx restart
```

Thanks to Oslandia for sharing their tutorial.

### Xvfb

QGIS Server needs a running X Server to be fully usable, in particular for printing. On servers it is usually recommended not to install it, so you may use `xvfb` to have a virtual X environment.

To install the package:

```bash
apt install xvfb
```

Then, according to your HTTP server, you should configure the `DISPLAY` parameter or directly use `xvfb-run`.

With Apache you just add to your Fcgi configuration (see above):

```bash
FcgidInitialEnv DISPLAY ":99"
```

Create the service file:

```bash
sh -c \
"echo \n"[Unit]
Description=X Virtual Frame Buffer Service
After=network.target

[Service]
ExecStart=/usr/bin/Xvfb :99 -screen 0 1024x768x24 -ac +extension GLX +render -noremote

[Install]
WantedBy=multi-user.target" \
> /etc/systemd/system/xvfb.service"
```

Enable, start and check the status of the `xvfb.service`:
systemctl enable xvfb.service
systemctl start xvfb.service
systemctl status xvfb.service

Now restart Apache for the new configuration to be taken into account:

systemctl restart apache2

With NGINX and spawn-fcgi using xvfb-run:

xvfb-run /usr/bin/spawn-fcgi -f /usr/lib/cgi-bin/qgis_marserv.fcgi \
    -s /tmp/qgisserver.socket \
    -G www-data -U www-data -n

The other option is to start a virtual X server environment with a specific display number thanks to Xvfb:

/usr/bin/Xvfb :99 -screen 0 1024x768x24 -ac +extension GLX +render -noreset

Then we just have to set the DISPLAY environment variable in the HTTP server configuration. For example with NGINX:

fastcgi_param DISPLAY ":99";

Installation on Windows

QGIS Server can also be installed on Windows systems. While the QGIS Server package is available in the 64 bit version of the OSGeo4W network installer (https://qgis.org/en/site/forusers/download.html) there is no Apache (or other web server) package available, so this must be installed by other means.

A simple procedure is the following:

- Download the XAMPP installer (https://www.apachefriends.org/download.html) for Windows and install Apache
- Download the OSGeo4W installer, follow the „Advanced Install“ and install both the QGIS Desktop and QGIS Server packages
- Edit the httpd.conf file (C:\xampp\apache\httpd.conf if the default installation paths have been used) and make the following changes:

From:

ScriptAlias /cgi-bin/ "C:/xampp/cgi-bin/"

To:

ScriptAlias /cgi-bin/ "c:/OSGeo4W64/apps/qgis/bin/"

From:

<Directory "C:/xampp/cgi-bin">
AllowOverride None
Options None
Require all granted
</Directory>

To:

<Directory "c:/OSGeo4W64/apps/qgis/bin">
SetHandler cgi-script
AllowOverride None
</Directory>

(continues on next page)
18.2. QGIS as OGC Data Server
Options ExecCGI
Order allow,deny
Allow from all
Require all granted
</Directory>

From:

AddHandler cgi-script .cgi .pl .asp

To:

AddHandler cgi-script .cgi .pl .asp .exe

Then at the bottom of httpd.conf add:

```
SetEnv GDAL_DATA "C:\OSGeo4W64\share\gdal"
SetEnv QGIS_AUTH_DB_DIR_PATH "C:\OSGeo4W64\apps\qgis\resources"
SetEnv PYTHONHOME "C:\OSGeo4W64\apps\Python37"
SetEnv PATH "C:\OSGeo4W64\bin;C:\OSGeo4W64\apps\qgis\bin;C:\OSGeo4W64\apps\Qt5\bin;...
SetEnv QGIS_PREFIX_PATH "C:\OSGeo4W64\apps\qgis"
SetEnv QT_PLUGIN_PATH "C:\OSGeo4W64\apps\qgis\qtplugins;C:\OSGeo4W64\apps\Qt5\...
```

Restart the Apache web server from the XAMPP Control Panel and open browser window to testing a GetCapabilities request to QGIS Server

```
http://localhost/cgi-bin/qgis_mapserv.fcgi.exe?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
```

Serve a project

Now that QGIS Server is installed and running, we just have to use it.

Obviously, we need a QGIS project to work on. Of course, you can fully customize your project by defining contact information, precise some restrictions on CRS or even exclude some layers. Everything you need to know about that is described later in Configure your project.

But for now, we are going to use a simple project already configured and previously downloaded in /home/qgis/projects/world.qgs, as described above.

By opening the project and taking a quick look on layers, we know that 4 layers are currently available:

- airports
- places
- countries
- countries_shapeburst

You don’t have to understand the full request for now but you may retrieve a map with some of the previous layers thanks to QGIS Server by doing something like this in your web browser to retrieve the countries layer:

```
http://localhost/qgisserver?
MAP=/home/qgis/projects/world.qgs
LAYERS=countries
SERVICE=WMS
REQUEST=GetMap
CRS=EPSG:4326
```

(continues on next page)
If you obtain the next image, then QGIS Server is running correctly:

![Server response to a basic GetMap request](image)

Obr. 18.6: Server response to a basic GetMap request

Note that you may define `PROJECT_FILE` environment variable to use a project by default instead of giving a `MAP` parameter (see Environment variables).

For example with `spawn-fcgi`:

```bash
export PROJECT_FILE=/home/qgis/projects/world.qgs
spawn-fcgi -f /usr/lib/bin/cgi-bin/qgis_mapserv.fcgi \
-s /var/run/qgisserver.socket \n-U www-data -G www-data -n
```

**Configure your project**

To provide a new QGIS Server WMS, WFS or WCS, you have to create a QGIS project file with some data or use one of your current project. Define the colors and styles of the layers in QGIS and the project CRS, if not already defined.

Then, go to the QGIS Server menu of the Project Properties… dialog and provide some information about the OWS in the fields under Service Capabilities. This will appear in the GetCapabilities response of the WMS, WFS or WCS.

If you don’t check Service capabilities, QGIS Server will use the information given in the `wms_metadata.xml` file located in the `cgi-bin` folder.

**WMS capabilities**

In the WMS capabilities section, you can define the extent advertised in the WMS GetCapabilities response by entering the minimum and maximum X and Y values in the fields under Advertised extent. Clicking Use Current Canvas Extent sets these values to the extent currently displayed in the QGIS map canvas. By checking CRS restrictions, you can restrict in which coordinate reference systems (CRS) QGIS Server will offer to render maps. It is recommended that you restrict the offered CRS as this reduces the size of the WMS GetCapabilities response. Use the button below to select those CRSs from the Coordinate Reference System Selector, or click Used to add the CRSs used in the QGIS project to the list.

If you have print layouts defined in your project, they will be listed in the GetProjectSettings response, and they can be used by the GetPrint request to create prints, using one of the print layout layouts as a template. This is a QGIS-specific extension to the WMS 1.3.0 specification. If you want to exclude any print layout from being published by
Obr. 18.7: Definitions for a QGIS Server WMS/WFS/WCS project
the WMS, check Exclude layouts and click the button below. Then, select a print layout from the Select print layout dialog in order to add it to the excluded layouts list.

If you want to exclude any layer or layer group from being published by the WMS, check Exclude Layers and click the button below. This opens the Select restricted layers and groups dialog, which allows you to choose the layers and groups that you don’t want to be published. Use the Shift or Ctrl key if you want to select multiple entries. It is recommended that you exclude from publishing the layers that you don’t need as this reduces the size of the WMS GetCapabilities response which leads to faster loading times on the client side.

You can receive requested GetFeatureInfo as plain text, XML and GML. Default is XML, text or GML format depends the output format chosen for the GetFeatureInfo request.

If you wish, you can check Add geometry to feature response. This will include the bounding box for each feature in the GetFeatureInfo response. See also the WITH_GEOMETRY parameter.

As many web clients can’t display circular arcs in geometries you have the option to segmentize the geometry before sending it to the client in a GetFeatureInfo response. This allows such clients to still display a feature’s geometry (e.g. for highlighting the feature). You need to check the Segmentize feature info geometry to activate the option.

You can also use the GetFeatureInfo geometry precision option to set the precision of the GetFeatureInfo geometry. This enables you to save bandwidth when you don’t need the full precision.

If you want QGIS Server to advertise specific request URLs in the WMS GetCapabilities response, enter the corresponding URL in the Advertised URL field.

Furthermore, you can restrict the maximum size of the maps returned by the GetMap request by entering the maximum width and height into the respective fields under Maximums for GetMap request.

If one of your layers uses the Map Tip display (i.e. to show text using expressions) this will be listed inside the GetFeatureInfo output. If the layer uses a Value Map for one of its attributes, this information will also be shown in the GetFeatureInfo output.

WFS capabilities

In the WFS capabilities area you can select the layers you want to publish as WFS, and specify if they will allow update, insert and delete operations. If you enter a URL in the Advertised URL field of the WFS capabilities section, QGIS Server will advertise this specific URL in the WFS GetCapabilities response.

WCS capabilities

In the WCS capabilities area, you can select the layers that you want to publish as WCS. If you enter a URL in the Advertised URL field of the WCS capabilities section, QGIS Server will advertise this specific URL in the WCS GetCapabilities response.

Fine tuning your OWS

For vector layers, the Fields menu of the Layer Layer Properties dialog allows you to define for each attribute if it will be published or not. By default, all the attributes are published by your WMS and WFS. If you don’t want a specific attribute to be published, uncheck the corresponding checkbox in the WMS or WFS column.

You can overlay watermarks over the maps produced by your WMS by adding text annotations or SVG annotations to the project file. See the Anotační nástroje section for instructions on creating annotations. For annotations to be displayed as watermarks on the WMS output, the Fixed map position checkbox in the Annotation text dialog must be unchecked. This can be accessed by double clicking the annotation while one of the annotation tools is active. For SVG annotations, you will need either to set the project to save absolute paths (in the General menu of the Project Properties dialog) or to manually modify the path to the SVG image so that it represents a valid relative path.
18.2.2 Services

QGIS Server is able to serve data according to standard protocols as described by the Open Geospatial Consortium (OGC):

- WMS 1.1.0 and 1.3.0
- WFS 1.0.0 and 1.1.0
- WFS3 (OGC API - Features)
- WCS 1.1.1
- WMTS 1.0.0

Extra vendor parameters and requests are supported in addition to the original standard that greatly enhance the possibilities of customizing its behavior thanks to the QGIS rendering engine.

Web Map Service (WMS)

The 1.1.0 and 1.3.0 WMS standards implemented in QGIS Server provide a HTTP interface to request map or legend images generated from a QGIS project. A typical WMS request defines the QGIS project to use, the layers to render as well as the image format to generate. Basic support is also available for Styled Layer Descriptor (SLD).

Specifications:

- WMS 1.1.0
- WMS 1.3.0
- SLD 1.1.0 WMS profile

Standard requests provided by QGIS Server:

<table>
<thead>
<tr>
<th>Request</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCapabilities</td>
<td>Returns XML metadata with information about the server</td>
</tr>
<tr>
<td>GetMap</td>
<td>Returns a map</td>
</tr>
<tr>
<td>GetFeatureInfo</td>
<td>Retrieves data (geometry and values) for a pixel location</td>
</tr>
<tr>
<td>GetLegendGraphics</td>
<td>Returns legend symbols</td>
</tr>
</tbody>
</table>

Vendor requests provided by QGIS Server:

<table>
<thead>
<tr>
<th>Request</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPrint</td>
<td>Returns a QGIS composition</td>
</tr>
<tr>
<td>GetProjectSettings</td>
<td>Returns specific information about QGIS Server</td>
</tr>
</tbody>
</table>

GetMap

Standard parameters for the GetMap request according to the OGC WMS 1.1.0 and 1.3.0 specifications:
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMS)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetMap)</td>
</tr>
<tr>
<td>LAYERS</td>
<td>No</td>
<td>Layers to display</td>
</tr>
<tr>
<td>STYLES</td>
<td>No</td>
<td>Layers' style</td>
</tr>
<tr>
<td>SRS / CRS</td>
<td>Yes</td>
<td>Coordinate reference system</td>
</tr>
<tr>
<td>BBOX</td>
<td>No</td>
<td>Map extent</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Yes</td>
<td>Width of the image in pixels</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Yes</td>
<td>Height of the image in pixels</td>
</tr>
<tr>
<td>FORMAT</td>
<td>No</td>
<td>Image format</td>
</tr>
<tr>
<td>TRANSPARENT</td>
<td>No</td>
<td>Transparent background</td>
</tr>
<tr>
<td>SLD</td>
<td>No</td>
<td>URL of an SLD to be used for styling</td>
</tr>
<tr>
<td>SLD_BODY</td>
<td>No</td>
<td>In-line SLD (XML) to be used for styling</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>Specify the QGIS project file</td>
</tr>
<tr>
<td>BGCOLOR</td>
<td>No</td>
<td>Specify the background color</td>
</tr>
<tr>
<td>DPI</td>
<td>No</td>
<td>Specify the output resolution</td>
</tr>
<tr>
<td>IMAGE_QUALITY</td>
<td>No</td>
<td>JPEG compression</td>
</tr>
<tr>
<td>OPACITIES</td>
<td>No</td>
<td>Opacity for layer or group</td>
</tr>
<tr>
<td>FILTER</td>
<td>No</td>
<td>Subset of features</td>
</tr>
<tr>
<td>SELECTION</td>
<td>No</td>
<td>Highlight features</td>
</tr>
</tbody>
</table>
| FILE_NAME              | No       | Only for FORMAT=application/dxf
File name of the downloaded file |
| FORMAT_OPTIONS         | No       | Only for FORMAT=application/dxf key:value pairs separated by semicolon.
  • SCALE: to be used for symbology rules, filters and styles (not actual scaling of the data - data remains in the original scale).
  • MODE: corresponds to the export options offered in the QGIS Desktop DXF export dialog. Possible values are NOSYMBOLOLOGY, FEATURESYMBOLOLOGY and SYMBOLLAYERSYMBOLOLOGY.
  • LAYERSATTRIBUTES: specify a field that contains values for DXF layer names - if not specified, the original QGIS layer names are used.
  • USE_TITLE_AS_LAYERNAME: if enabled, the title of the layer will be used as layer name.
  • CODEC: specify a codec to be used for encoding. Default is ISO-8859-1 check the QGIS desktop DXF export dialog for valid values.
| TILED                 | No       | Working in tiled mode                      |

**URL example:**

```plaintext
http://localhost/qgis_server?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&MAP=/home/qgis/projects/world.qgs
&LAYERS=mylayer1,mylayer2,mylayer3
&STYLES=style1,default,style3
&OPACITIES=125,200,125
&CRS=EPSG:4326
&WIDTH=400
&HEIGHT=400
```

(continues on next page)
SERVICE

This parameter has to be WMS in case of the GetMap request.

VERSION

This parameter allows to specify the version of the service to use. Available values for the VERSION parameter are:

- 1.1.0
- 1.3.0

If no version is indicated in the request, then 1.3.0 is used by default.

According to the version number, slight differences have to be expected as explained later for the next parameters:

- CRS/SRS
- BBOX

REQUEST

This parameter is GetMap in case of the GetMap request.

LAYERS

This parameter allows to specify the layers to display on the map. Names have to be separated by a comma.

In addition, QGIS Server introduced some options to select layers by:

- a short name
- the layer id

The short name of a layer may be configured through Properties Metadata in layer menu. If the short name is defined, then it’s used by default instead of the layer’s name:

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mynickname1,mynickname2
&...

Moreover, there’s a project option allowing to select layers by their id in OWS Server WMS capabilities menu of the Project Properties… dialog. To activate this option, the checkbox Use layer ids as names has to be selected.

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayerid1,mylayerid2
&...
**STYLES**

This parameter can be used to specify a layer’s style for the rendering step. Styles have to be separated by a comma. The name of the default style is `default`.

**SRS / CRS**

This parameter allows to indicate the map output Spatial Reference System in WMS 1.1.0 and has to be formed like `EPSG:XXXX`. Note that CRS is also supported if current version is 1.1.0.

For WMS 1.3.0, CRS parameter is preferable but SRS is also supported.

Note that if both CRS and SRS parameters are indicated in the request, then it’s the current version indicated in VERSION parameter which is decisive.

In the next case, the SRS parameter is kept whatever the VERSION parameter because CRS is not indicated:

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&SRS=EPSG:2854
&...
```

In the next case, the SRS parameter is kept instead of CRS because of the VERSION parameter:

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.1.0
&CRS=EPSG:4326
&SRS=EPSG:2854
&...
```

In the next case, the CRS parameter is kept instead of SRS because of the VERSION parameter:

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&CRS=EPSG:4326
&SRS=EPSG:2854
&...
```

**BBOX**

This parameter allows to specify the map extent with units according to the current CRS. Coordinates have to be separated by a comma.

However, a slight difference has to be noticed according to the current VERSION parameter. In WMS 1.1.0, coordinates are formed like `minx, miny, maxx, maxy` or `minlong, minlat, maxlong, maxlat`. For example:

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.1.0
&SRS=epsg:4326
&BBOX=-180,-90,180,90
&...
```
But the axis is reversed in WMS 1.3.0, so coordinates are formed like: miny, minx, maxy, maxx or minlat, minlong, maxlat, maxlong. For example:

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&CRS=epsg:4326
&BBOX=-90,-180,90,180
&...
```

**WIDTH**

This parameter allows to specify the width in pixels of the output image.

**HEIGHT**

This parameter allows to specify the height in pixels of the output image.

**FORMAT**

This parameter may be used to specify the format of map image. Available values are:

- `jpg`
- `jpeg`
- `image/jpeg`
- `image/png`
- `image/png; mode=1bit`
- `image/png; mode=8bit`
- `image/png; mode=16bit`
- `application/dxf` Only layers that have read access in the WFS service are exported in the DXF format.

**URL example:**

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&FORMAT=application/dxf
&LAYERS=Haltungen,Normschacht,Spezialbauwerke
&STYLES= 
&CRS=EPSG%3A21781&BBOX=696136.28844801,245797.12108743,696318.91114315,245939.25832905
&WIDTH=1042
&HEIGHT=811
&FORMAT_OPTIONS=MODE:SYMBOL LAYERSYMBOL;SCALE:250&FILE_NAME=plan.dxf
```
TRANSPARENT

This boolean parameter can be used to specify the background transparency. Available values are (not case sensitive):

- TRUE
- FALSE

However, this parameter is ignored if the format of the map image indicated with FORMAT is different from PNG.

MAP

This parameter allows to define the QGIS project file to use.

As mentioned in GetMap parameters table, MAP is mandatory because a request needs a QGIS project to actually work. However, the QGIS_PROJECT_FILE environment variable may be used to define a default QGIS project. In this specific case, MAP is not longer a required parameter. For further information you may refer to Advanced configuration.

BGCOLOR

This parameter allows to indicate a background color for the map image. However it cannot be combined with TRANSPARENT parameter in case of PNG images (transparency takes priority). The colour may be literal or in hexadecimal notation.

URL example with the literal notation:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&BGCOLOR=green
&...
```

URL example with the hexadecimal notation:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&BGCOLOR=0x00FF00
&...
```

DPI

This parameter can be used to specify the requested output resolution.
**IMAGE_QUALITY**

This parameter is only used for JPEG images. By default, the JPEG compression is ~1.

You can change the default per QGIS project in the **OWS Server 🗺 WMS capabilities** menu of the **Project 🕒 Properties…** dialog. If you want to override it in a GetMap request you can do it using the **IMAGE_QUALITY** parameter.

**OPACITIES**

Opacity can be set on layer or group level. Allowed values range from 0 (fully transparent) to 255 (fully opaque).

**FILTER**

A subset of layers can be selected with the **FILTER** parameter. The syntax is basically the same as for the QGIS subset string. However, there are some restrictions to avoid SQL injections into databases via QGIS Server. If a dangerous string is found in the parameter, QGIS Server will return the next error:

```
Indeed, text strings need to be enclosed with quotes (single quotes for strings, double quotes for attributes). A space between each word / special character is mandatory. Allowed Keywords and special characters are 'AND','OR','IN','='','<','>','=','!','(',')'.
Semicolons in string expressions are not allowed.
```

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayer1,mylayer2,mylayer3
&FILTER=mylayer1:"col1";mylayer1,mylayer2:"col2" = 'blabla'
&...
```

In this example, the same filter (field `col2` equals the string `blabla`) is applied to layers `mylayer1` and `mylayer2`, while the filter on `col1` is only applied to `mylayer1`.

**Poznámka:** It is possible to make attribute searches via GetFeatureInfo and omit the X/Y parameter if a FILTER is there. QGIS Server then returns info about the matching features and generates a combined bounding box in the XML output.

**SELECTION**

The **SELECTION** parameter can highlight features from one or more layers. Vector features can be selected by passing comma separated lists with feature ids.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayer1,mylayer2
&SELECTION=mylayer1:3,6,9;mylayer2:1,5,6
&...
```

The following image presents the response from a GetMap request using the **SELECTION** option e.g. `http://myserver.com/...&SELECTION=countries:171,65`.

As those features id’s correspond in the source dataset to **France** and **Romania** they’re highlighted in yellow.
Obr. 18.8: Server response to a GetMap request with SELECTION parameter
TILED

Set the TILED parameter to TRUE to tell QGIS Server to work in tiled mode, and to apply the Tile buffer configured in the QGIS project.

When TILED is TRUE and when a non-zero Tile buffer is configured in the QGIS project, features outside the tile extent are drawn to avoid cut symbols at tile boundaries.

TILED defaults to FALSE.

GetFeatureInfo

Standard parameters for the GetFeatureInfo request according to the OGC WMS 1.1.0 and 1.3.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMS)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>See GetMap</td>
</tr>
<tr>
<td>LAYERS</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>STYLES</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>SRS / CRS</td>
<td>Yes</td>
<td>See GetMap</td>
</tr>
<tr>
<td>BBOX</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Yes</td>
<td>See GetMap</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Yes</td>
<td>See GetMap</td>
</tr>
<tr>
<td>TRANSPARENT</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>INFO_FORMAT</td>
<td>No</td>
<td>Output format</td>
</tr>
<tr>
<td>QUERY_LAYERS</td>
<td>Yes</td>
<td>Layers to query</td>
</tr>
<tr>
<td>FEATURE_COUNT</td>
<td>No</td>
<td>Maximum number of features to return</td>
</tr>
<tr>
<td>I</td>
<td>No</td>
<td>Pixel column of the point to query</td>
</tr>
<tr>
<td>X</td>
<td>No</td>
<td>Same as I parameter, but in WMS 1.1.0</td>
</tr>
<tr>
<td>J</td>
<td>No</td>
<td>Pixel row of the point to query</td>
</tr>
<tr>
<td>A</td>
<td>No</td>
<td>Same as J parameter, but in WMS 1.1.0</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>See GetMap</td>
</tr>
<tr>
<td>FILTER</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>FI_POINT_TOLERANCE</td>
<td>No</td>
<td>Tolerance in pixels for point layers</td>
</tr>
<tr>
<td>FI_LINE_TOLERANCE</td>
<td>No</td>
<td>Tolerance in pixels for line layers</td>
</tr>
<tr>
<td>FI_POLYGON_TOLERANCE</td>
<td>No</td>
<td>Tolerance in pixels for polygon layers</td>
</tr>
<tr>
<td>FILTER_GEOM</td>
<td>No</td>
<td>Geometry filtering</td>
</tr>
<tr>
<td>WITH_MAPTIP</td>
<td>No</td>
<td>Add map tips to the output</td>
</tr>
<tr>
<td>WITH_GEOMETRY</td>
<td>No</td>
<td>Add geometry to the output</td>
</tr>
</tbody>
</table>

URL example:

http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&MAP=/home/qgis/projects/world.qgs
&LAYERS=mylayer1,mylayer2,mylayer3
&CRS=EPSG:4326
&WIDTH=400
&HEIGHT=400

(continues on next page)
INFO_FORMAT

This parameter may be used to specify the format of the result. Available values are:

- text/xml
- text/html
- text/plain
- application/vnd.ogc.gml
- application/json

QUERY_LAYERS

This parameter specifies the layers to display on the map. Names are separated by a comma.

In addition, QGIS Server introduces options to select layers by:

- short name
- layer id

See the LAYERS parameter defined in See GetMap for more information.

FEATURE_COUNT

This parameter specifies the maximum number of features per layer to return. For example if QUERY_LAYERS is set to layer1, layer2 and FEATURE_COUNT is set to 3 then a maximum of 3 features from layer1 will be returned. Likewise a maximum of 3 features from layer2 will be returned.

By default, only 1 feature per layer is returned.

I

This parameter, defined in WMS 1.3.0, allows you to specify the pixel column of the query point.

X

Same parameter as I, but defined in WMS 1.1.0.
J

This parameter, defined in WMS 1.3.0, allows you to specify the pixel row of the query point.

A

Same parameter as J, but defined in WMS 1.1.0.

FI_POINT_TOLERANCE

This parameter specifies the tolerance in pixels for point layers.

FI_LINE_TOLERANCE

This parameter specifies the tolerance in pixels for line layers.

FI_POLYGON_TOLERANCE

This parameter specifies the tolerance in pixels for polygon layers.

FILTER_GEOM

This parameter specifies a WKT geometry with which features have to intersect.

WITH_MAPTIP

This parameter specifies whether to add map tips to the output.
Available values are (not case sensitive):

- TRUE
- FALSE

WITH_GEOMETRY

This parameter specifies whether to add geometries to the output. To use this feature you must first enable the Add geometry to feature response option in the QGIS project. See Configure your project.
Available values are (not case sensitive):

- TRUE
- FALSE
GetPrint

QGIS Server has the capability to create print layout output in pdf or pixel format. Print layout windows in the published project are used as templates. In the GetPrint request, the client has the possibility to specify parameters of the contained layout maps and labels.

Parameters for the GetPrint request:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>Specify the QGIS project file</td>
</tr>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMS)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetPrint)</td>
</tr>
<tr>
<td>LAYERS</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Yes</td>
<td>Layout template to use</td>
</tr>
<tr>
<td>SRS / CRS</td>
<td>Yes</td>
<td>See GetMap</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Yes</td>
<td>Output format</td>
</tr>
<tr>
<td>ATLAS_PK</td>
<td>No</td>
<td>Atlas features</td>
</tr>
<tr>
<td>STYLES</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>TRANSPARENT</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>OPACITIES</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>SELECTION</td>
<td>No</td>
<td>See GetMap</td>
</tr>
<tr>
<td>mapX:EXTENT</td>
<td>No</td>
<td>Extent of the map 'X'</td>
</tr>
<tr>
<td>mapX:LAYERS</td>
<td>No</td>
<td>Layers of the map 'X'</td>
</tr>
<tr>
<td>mapX:STYLES</td>
<td>No</td>
<td>Layers' style of the map 'X'</td>
</tr>
<tr>
<td>mapX:SCALE</td>
<td>No</td>
<td>Layers' scale of the map 'X'</td>
</tr>
<tr>
<td>mapX:ROTATION</td>
<td>No</td>
<td>Rotation of the map 'X'</td>
</tr>
<tr>
<td>mapX:GRID_INTERVAL_X</td>
<td>No</td>
<td>Grid interval on x axis of the map 'X'</td>
</tr>
<tr>
<td>mapX:GRID_INTERVAL_Y</td>
<td>No</td>
<td>Grid interval on y axis of the map 'X'</td>
</tr>
</tbody>
</table>

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetPrint
&MAP=/home/qgis/projects/world.qgs
&CRS=EPSG:4326
&FORMAT=png
&map0:EXTENT=-180,-90,180,90
&map0:LAYERS=mylayer1,mylayer2,mylayer3
&map0:OPACITIES=125,200,125
&map0:ROTATION=45
```

Note that the layout template may contain more than one map. In this way, if you want to configure a specific map, you have to use mapX: parameters where X is a positive number that you can retrieve thanks to the GetProjectSettings request.

For example:

```
<WMS_Capabilities>
...
<ComposerTemplates xsi:type="wms:_ExtendedCapabilities">
<ComposerTemplate width="297" height="210" name="Druckzusammenstellung 1"/>
<ComposerMap width="171" height="133" name="map0"/>
<ComposerMap width="49" height="46" name="map1"/>
</ComposerTemplate>
</ComposerTemplates>
...
</WMS_Capabilities>
```
SERVICE

This parameter has to be WMS.

REQUEST

This parameter has to be GetPrint for the GetPrint request.

TEMPLATE

This parameter can be used to specify the name of a layout template to use for printing.

FORMAT

This parameter specifies the format of map image. Available values are:

- jpg
- jpeg
- image/jpeg
- png
- image/png
- svg
- image/svg
- image/svg+xml
- pdf
- application/pdf

If the FORMAT parameter is different from one of these values, then an exception is returned.

ATLAS_PK

This parameter allows activation of Atlas rendering by indicating which features we want to print. In order to retrieve an atlas with all features, the * symbol may be used (according to the maximum number of features allowed in the project configuration).

When FORMAT is pdf, a single PDF document combining the feature pages is returned. For all other formats, a single page is returned.

mapX:EXTENT

This parameter specifies the extent for a layout map item as xmin,ymin,xmax,ymax.
mapX:ROTATION

This parameter specifies the map rotation in degrees.

mapX:GRID INTERVAL X

This parameter specifies the grid line density in the X direction.

mapX:GRID INTERVAL Y

This parameter specifies the grid line density in the Y direction.

mapX:SCALE

This parameter specifies the map scale for a layout map item. This is useful to ensure scale based visibility of layers and labels even if client and server may have different algorithms to calculate the scale denominator.

mapX:LAYERS

This parameter specifies the layers for a layout map item. See See GetMap for more information on this parameter.

mapX:STYLES

This parameter specifies the layers’ styles defined in a specific layout map item. See See GetMap for more information on this parameter.

GetLegendGraphics

Several additional parameters are available to change the size of the legend elements:

- **BOXSPACE** space between legend frame and content (mm)
- **LAYERSPACE** vertical space between layers (mm)
- **LAYERTITLESPEC** vertical space between layer title and items following (mm)
- **SYMBOLSPACE** vertical space between symbol and item following (mm)
- **ICONLABELSPACE** horizontal space between symbol and label text (mm)
- **SYMBOLWIDTH** width of the symbol preview (mm)
- **SYMBOLHEIGHT** height of the symbol preview (mm)

These parameters change the font properties for layer titles and item labels:

- **LAYERFONTFAMILY / ITEMFONTFAMILY** font family for layer title / item text
- **LAYERFONTBOLD / ITEMFONTBOLD** TRUE to use a bold font
- **LAYERFONTSIZE / ITEMFONTSIZE** Font size in point
- **LAYERFONTITALIC / ITEMFONTITALIC** TRUE to use italic font
- **LAYERFONTCOLOR / ITEMFONTCOLOR** Hex color code (e.g. #FF0000 for red)
- **LAYERTITLE** FALSE to get only the legend graphics without the layer title
- **RULELABEL**
Content based legend. These parameters let the client request a legend showing only the symbols for the features falling into the requested area:

- **BBOX** the geographical area for which the legend should be built
- **CRS / SRS** the coordinate reference system adopted to define the BBOX coordinates
- **WIDTH / HEIGHT** if set these should match those defined for the GetMap request, to let QGIS Server scale symbols according to the map view image size.

Content based legend features are based on the UMN MapServer implementation:

- **SHOWFEATURECOUNT** if set to `TRUE` adds in the legend the feature count of the features like in the following image:

```
Tram lines [10]

1 [2]
2 [2]
3 [4]
4 [2]
```

### GetProjectSettings

This request type works similar to **GetCapabilities**, but it is more specific to QGIS Server and allows a client to read additional information which is not available in the GetCapabilities output:

- initial visibility of layers
- information about vector attributes and their edit types
- information about layer order and drawing order
- list of layers published in WFS

### Web Feature Service (WFS)

The 1.0.0 and 1.1.0 WFS standards implemented in QGIS Server provide a HTTP interface to query geographic features from a QGIS project. A typical WFS request defines the QGIS project to use and the layer to query.

Specifications document according to the version number of the service:

- WFS 1.0.0
- WFS 1.1.0

Standard requests provided by QGIS Server:

<table>
<thead>
<tr>
<th>Request</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCapabilities</td>
<td>Returns XML metadata with information about the server</td>
</tr>
<tr>
<td>GetFeature</td>
<td>Returns a selection of features</td>
</tr>
<tr>
<td>DescribeFeatureType</td>
<td>Returns a description of feature types and properties</td>
</tr>
<tr>
<td>Transaction</td>
<td>Allows features to be inserted, updated or deleted</td>
</tr>
</tbody>
</table>
GetFeature

Standard parameters for the GetFeature request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request</td>
</tr>
<tr>
<td>TYPENAME</td>
<td>No</td>
<td>Name of layers</td>
</tr>
<tr>
<td>OUTPUTFORMAT</td>
<td>No</td>
<td>Output Format</td>
</tr>
<tr>
<td>RESULTTYPE</td>
<td>No</td>
<td>Type of the result</td>
</tr>
<tr>
<td>PROPERTYNAME</td>
<td>No</td>
<td>Name of properties to return</td>
</tr>
<tr>
<td>MAXFEATURES</td>
<td>No</td>
<td>Maximum number of features to return</td>
</tr>
<tr>
<td>SRSNAME</td>
<td>No</td>
<td>Coordinate reference system</td>
</tr>
<tr>
<td>FEATUREID</td>
<td>No</td>
<td>Filter the features by ids</td>
</tr>
<tr>
<td>FILTER</td>
<td>No</td>
<td>OGC Filter Encoding</td>
</tr>
<tr>
<td>BBOX</td>
<td>No</td>
<td>Map Extent</td>
</tr>
<tr>
<td>SORTBY</td>
<td>No</td>
<td>Sort the results</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>Specify the QGIS project file</td>
</tr>
<tr>
<td>STARTINDEX</td>
<td>No</td>
<td>Paging</td>
</tr>
<tr>
<td>GEOMETRYNAME</td>
<td>No</td>
<td>Type of geometry to return</td>
</tr>
<tr>
<td>EXP_FILTER</td>
<td>No</td>
<td>Expression filtering</td>
</tr>
</tbody>
</table>

SERVICE

This parameter has to be WFS in case of the GetFeature request.

For example:

```
http://localhost/qgissserver?
SERVICE=WFS
&...
```

VERSION

This parameter allows to specify the version of the service to use. Available values for the VERSION parameter are:

- 1.0.0
- 1.1.0

If no version is indicated in the request, then 1.1.0 is used by default.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&...
```
REQUEST

This parameter is `GetFeature` in case of the `GetFeature` request.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&...
```

RESULTTYPE

This parameter may be used to specify the kind of result to return. Available values are:

- results: the default behavior
- hits: returns only a feature count

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&RESULTTYPE=hits
&...
```

GEOMETRYNAME

This parameter can be used to specify the kind of geometry to return for features. Available values are:

- extent
- centroid
- none

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&GEOMETRYNAME=centroid
&...
```

STARTINDEX

This parameter is standard in WFS 2.0, but it’s an extension for WFS 1.0.0. Actually, it can be used to skip some features in the result set and in combination with `MAXFEATURES`, it provides the ability to page through results.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&...
```
Web Map Tile Service (WMTS)

The **1.0.0** WMTS standard implemented in QGIS Server provides a HTTP interface to request tiled map images generated from a QGIS project. A typical WMTS request defined the QGIS project to use, some WMS parameters like layers to render, as well as tile parameters.

Specifications document of the service:

- **WMTS 1.0.0**

Standard requests provided by QGIS Server:

<table>
<thead>
<tr>
<th>Request</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCapabilities</td>
<td>Returns XML metadata with information about the server</td>
</tr>
<tr>
<td>GetTile</td>
<td>Returns a tile</td>
</tr>
<tr>
<td>GetFeatureInfo</td>
<td>Retrieves data (geometry and values) for a pixel location</td>
</tr>
</tbody>
</table>

**GetCapabilities**

Standard parameters for the **GetCapabilities** request according to the OGC WMTS 1.0.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMTS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetCapabilities)</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>Specify the QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

```
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetCapabilities
&MAP=/home/qgis/projects/world.qgs
```

**SERVICE**

This parameter has to be **WMTS** in case of the **GetCapabilities** request.
REQUEST

This parameter is *GetCapabilities* in case of the *GetCapabilities* request.

MAP

This parameter allows to define the QGIS project file to use.

GetTile

Standard parameters for the *GetTile* request according to the OGC WMTS 1.0.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMTS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetTile)</td>
</tr>
<tr>
<td>LAYER</td>
<td>Yes</td>
<td>Layer identifier</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Yes</td>
<td>Output format of the tile</td>
</tr>
<tr>
<td>TILEMATRIXSET</td>
<td>Yes</td>
<td>Name of the pyramid</td>
</tr>
<tr>
<td>TILEMATRIX</td>
<td>Yes</td>
<td>Meshing</td>
</tr>
<tr>
<td>TILEROW</td>
<td>Yes</td>
<td>Row coordinate in the mesh</td>
</tr>
<tr>
<td>TILECOL</td>
<td>Yes</td>
<td>Column coordinate in the mesh</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>Specify the QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

```url
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetTile
&MAP=/home/qgis/projects/world.qgs
&LAYER=mylayer
&FORMAT=image/png
&TILEMATRIXSET=EPSG:4326
&TILEROW=0
&TILECOL=0
```

SERVICE

This parameter has to be WMTS in case of the *GetTile* request.
REQUEST

This parameter is GetTile in case of the GetTile request.

LAYER

This parameter allows to specify the layer to display on the tile.

In addition, QGIS Server introduced some options to select a layer by:

- a short name
- the layer id

The short name of a layer may be configured through Properties Metadata in layer menu. If the short name is defined, then it’s used by default instead of the layer’s name:

```
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetTile
&LAYER=mynickname
&...
```

Moreover, there’s a project option allowing to select layers by their id in OWS Server WMS capabilities menu of the Project Project Properties dialog. To activate this option, the checkbox Use layer ids as names has to be selected.

```
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetTile
&LAYER=mylayerid1
&...
```

FORMAT

This parameter may be used to specify the format of tile image. Available values are:

- jpg
- jpeg
- image/jpeg
- image/png

If the FORMAT parameter is different from one of these values, then the default format PNG is used instead.

TILEMATRIXSET

This parameter defines the CRS to use when computing the underlying pyramid. Format: EPSG:XXXX.
TILEMATRIX

This parameter allows to define the matrix to use for the output tile.

TILEROW

This parameter allows to select the row of the tile to get within the matrix.

TILECOL

This parameter allows to select the column of the tile to get within the matrix.

MAP

This parameter allows to define the QGIS project file to use.

As mentioned in *GetMap parameters table*, MAP is mandatory because a request needs a QGIS project to actually work. However, the `QGIS_PROJECT_FILE` environment variable may be used to define a default QGIS project. In this specific case, MAP is not longer a required parameter. For further information you may refer to *Advanced configuration*.

GetFeatureInfo

Standard parameters for the *GetFeatureInfo* request according to the OGC WMTS 1.0.0 specifications:

- **WMS 1.1.0**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMTS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetFeatureInfo)</td>
</tr>
<tr>
<td>LAYER</td>
<td>Yes</td>
<td>Layer identifier</td>
</tr>
<tr>
<td>INFOFORMAT</td>
<td>No</td>
<td>Output format</td>
</tr>
<tr>
<td>I</td>
<td>No</td>
<td>X coordinate of a pixel</td>
</tr>
<tr>
<td>J</td>
<td>No</td>
<td>Y coordinate of a pixel</td>
</tr>
<tr>
<td>TILEMATRIXSET</td>
<td>Yes</td>
<td>See GetTile</td>
</tr>
<tr>
<td>TILEMATRIX</td>
<td>Yes</td>
<td>See GetTile</td>
</tr>
<tr>
<td>TILEROW</td>
<td>Yes</td>
<td>See GetTile</td>
</tr>
<tr>
<td>TILECOL</td>
<td>Yes</td>
<td>See GetTile</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>Specify the QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

```
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetFeatureInfo
&MAP=/home/qgis/projects/world.qgs
&LAYER=mylayer
&INFOFORMAT=image/html
&I=10
&J=5
```
SERVICE

This parameter has to be WMTS in case of the GetFeatureInfo request.

REQUEST

This parameter is GetFeatureInfo in case of the GetFeatureInfo request.

MAP

This parameter allows to define the QGIS project file to use.

As mentioned in GetMap parameters table, MAP is mandatory because a request needs a QGIS project to actually work. However, the QGIS_PROJECT_FILE environment variable may be used to define a default QGIS project. In this specific case, MAP is not longer a required parameter. For further information you may refer to Advanced configuration.

LAYER

This parameter allows to specify the layer to display on the tile.

In addition, QGIS Server introduced some options to select a layer by:

- a short name
- the layer id

The short name of a layer may be configured through Properties Metadata in layer menu. If the short name is defined, then it’s used by default instead of the layer’s name:

```html
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetFeatureInfo
&LAYER=mynickname
&...
```

Moreover, there’s a project option allowing to select layers by their id in OWS Server WMS capabilities menu of the Project Project Properties dialog. To activate this option, the checkbox Use layer ids as names has to be selected.

```html
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetFeatureInfo
&LAYER=mylayerid1
&...
```

INFOFORMAT

This parameter allows to define the output format of the result. Available values are:

- text/xml
- text/html
- text/plain
- application/vnd.ogc.gml

The default value is text/plain.
This parameter allows to define the X coordinate of the pixel for which we want to retrieve underlying information.

This parameter allows to define the Y coordinate of the pixel for which we want to retrieve underlying information.

**WFS3 (OGC API Features)**

WFS3 is the first implementation of the new generation of OGC protocols. It is described by the OGC API - Features - Part 1: Core document.

Here is a quick informal summary of the most important differences between the well known WFS protocol and WFS3:

- WFS3 is based on a REST API
- WFS3 API must follow the OPENAPI specifications
- WFS3 supports multiple output formats but it does not dictate any (only GeoJSON and HTML are currently available in QGIS WFS3) and it uses content negotiation to determine which format is to be served to the client
- JSON and HTML are first class citizens in WFS3
- WFS3 is self-documenting (through the /api endpoint)
- WFS3 is fully navigable (through links) and browsable

**Důležité:** While the WFS3 implementation in QGIS can make use of the MAP parameter to specify the project file, no extra query parameters are allowed by the OPENAPI specification. For this reason it is strongly recommended that MAP is not exposed in the URL and the project file is specified in the environment by other means (i.e. setting QGIS_PROJECT_FILE in the environment through a web server rewrite rule).

**Poznámka:** The API endpoint provides comprehensive documentation of all supported parameters and output formats of your service. The following paragraphs will only describe the most important ones.

### Resource representation

The QGIS Server WFS3 implementation currently supports the following resource representation (output) formats:

- HTML
- JSON

The format that is actually served will depend on content negotiation, but a specific format can be explicitly requested by appending a format specifier to the endpoints.

Supported format specifier extensions are:

- .json
- .html

Additional format specifier aliases may be defined by specific endpoints:

- .openapi: alias for .json supported by the API endpoint
- .geojson: alias for .json supported by the Features and Feature endpoints
Endpoints

The API provides a list of endpoints that the clients can retrieve. The system is designed in such a way that every response provides a set of links to navigate through all the provided resources.

Endpoints points provided by the QGIS implementation are:

<table>
<thead>
<tr>
<th>Název</th>
<th>Path</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Landing Page</td>
<td>/</td>
<td>General information about the service and provides links to all available endpoints</td>
</tr>
<tr>
<td>Conformance</td>
<td>/conformance</td>
<td>Information about the conformance of the service to the standards</td>
</tr>
<tr>
<td>API</td>
<td>/api</td>
<td>Full description of the endpoints provided by the service and the returned documents structure</td>
</tr>
<tr>
<td>Collections</td>
<td>/collections</td>
<td>List of all collections (i.e., vector layers) provided by the service</td>
</tr>
<tr>
<td>Collection</td>
<td>/collections/{collectionId}</td>
<td>Information about a collection (name, metadata, extent etc.)</td>
</tr>
<tr>
<td>Funkce</td>
<td>/collections/{collectionId}/items</td>
<td>List of the features provided by the collection</td>
</tr>
<tr>
<td>Feature</td>
<td>/collections/{collectionId}/items/{featureId}</td>
<td>Information about a single feature</td>
</tr>
</tbody>
</table>

Landing Page

The main endpoint is the Landing Page. From that page it is possible to navigate to all the available service endpoints. The Landing Page must provide links to

- the API definition (path /api link relations service-desc and service-doc),
- the Conformance declaration (path /conformance, link relation conformance), and
- the Collections (path /collections, link relation data).

Obr. 18.9: Server WFS3 landing page
API Definition

The API Definition is an OPENAPI-compliant description of the API provided by the service. In its HTML representation it is a browsable page where all the endpoints and their response formats are accurately listed and documented. The path of this endpoint is /api

The API definition provides a comprehensive and authoritative documentation of the service, including all supported parameters and returned formats.

**Poznámka:** This endpoint is analogue to WFS's GetCapabilities

Collections list

The collections endpoint provides a list of all the collections available in the service. Since the service „serves“ a single QGIS project the collections are the vector layers from the current project (if they were published as WFS in the project properties). The path of this endpoint is /collections/.

![Obr. 18.10: Server WFS3 collections list page](image)

Collection detail

While the collections endpoint does not provide detailed information about each available collection, that information is available in the /collections/{collectionId} endpoints. Typical information includes the extent, a description, CRSs and other metadata.

The HTML representation also provides a browsable map with the available features.

Features list

This endpoint provides a list of all features in a collection knowing the collection ID. The path of this endpoint is /collections/{collectionId}/items.

The HTML representation also provides a browsable map with the available features.

**Poznámka:** This endpoint is analogue to GetFeature in WFS 1 and WFS 2.
Obr. 18.11: Server WFS3 collection detail page

Obr. 18.12: Server WFS3 features list page

18.2. QGIS as OGC Data Server
Feature detail

This endpoint provides all the available information about a single feature, including the feature attributes and its geometry. The path of this endpoint is /collections/{collectionId}/items/{itemId}.

The HTML representation also provides a browsable map with the feature geometry.

Obr. 18.13: Server WFS3 feature detail page

Pagination

Pagination of a long list of features is implemented in the OGC API through next and prev links. QGIS server constructs these links by appending limit and offset as query string parameters.

URL example:

```
```

Poznámka: The maximum acceptable value for limit can be configured with the QGIS_SERVER_API_WFS3_MAX_LIMIT server configuration setting (see: Environment variables).

Feature filtering

The features available in a collection can be filtered/searched by specifying one or more filters.
Date and time filter

Collections with date and/or datetime attributes can be filtered by specifying a datetime argument in the query string. By default the first date/datetime field is used for filtering. This behavior can be configured by setting a „Date“ or „Time“ dimension in the QGIS Server Dimension section of the layer properties dialog.

The date and time filtering syntax is fully described in the API Definition and also supports ranges (begin and end values are included) in addition to single values.

URL examples:

Returns only the features with date dimension matching 2019-01-01


Returns only the features with datetime dimension matching 2019-01-01T01:01:01


Returns only the features with datetime dimension in the range 2019-01-01T01:01:01 - 2019-01-01T12:00:00

http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-01T01:01:01/2019-01-01T12:00:00

Bounding box filter

A bounding box spatial filter can be specified with the bbox parameter:

The order of the comma separated elements is:

- Lower left corner, WGS 84 longitude
- Lower left corner, WGS 84 latitude
- Upper right corner, WGS 84 longitude
- Upper right corner, WGS 84 latitude

Poznámka: The OGC specifications also allow a 6 item bbox specifier where the third and sixth items are the Z components, this is not yet supported by QGIS server.

URL example:

http://localhost/qgisserver/wfs3/collection_one/items.json?bbox=-180,-90,180,90

If the CRS of the bounding box is not WGS84 (http://www.opengis.net/def/crs/OGC/1.3/CRS84), a different CRS can be specified by using the optional parameter bbox-crs. The CRS format identifier must be in the OGC URI format:

URL example:

**Attribute filters**

Attribute filters can be combined with the bounding box filter and they are in the general form: `attribute name=attribute value`. Multiple filters can be combined using the AND operator.

**URL example:**
filters all features where attribute name equals „my value“

http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=my%20value

Partial matches are also supported by using a * („star”) operator:

**URL example:**
filters all features where attribute name ends with „value“

http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=*value

**Attribute selection**

The feature attributes returned by a Features list call can be limited by adding a comma separated list of attribute names in the optional properties query string argument.

**URL example:**
returns only the name attribute

http://localhost/qgisserver/wfs3/collection_one/items.json?properties=name

**The HTML template language**

The HTML representation uses a set of HTML templates to generate the response. The template is parsed by a template engine called inja. The templates can be customized by overriding them (see: Template overrides). The template has access to the same data that are available to the JSON representation and a few additional functions are available to the template:

**Custom template functions**

- `path_append(path)`: appends a directory path to the current url
- `path_chomp(n)`: removes the specified number „n“ of directory components from the current url path
- `json_dump()`: prints the JSON data passed to the template
- `static(path)`: returns the full URL to the specified static path. For example: „static(„/style/black.css“)” with a root path „http://localhost/qgisserver/wfs3“ will return „http://localhost/qgisserver/wfs3/static/style/black.css“.
- `links_filter(links, key, value)`: Returns filtered links from a link list
- `content_type_name(content_type)`: Returns a short name from a content type, for example „text/html“ will return „HTML“.
Template overrides

Templates and static assets are stored in subdirectories of the QGIS server default API resource directory (/usr/share/qgis/resources/server/api/ on a Linux system), the base directory can be customized by changing the environment variable QGIS_SERVER_API_RESOURCES_DIRECTORY.

A typical Linux installation will have the following directory tree:

```
/usr/share/qgis/resources/server/api/
  └── ogc
      ├── schema.json
      └── static
          ├── jsonFormatter.min.css
          ├── jsonFormatter.min.js
          └── style.css
          └── templates
              └── wfs3
                  ├── describeCollection.html
                  ├── describeCollections.html
                  ├── footer.html
                  ├── getApiDescription.html
                  ├── getFeature.html
                  ├── getFeatures.html
                  ├── getLandingPage.html
                  └── getRequirementClasses.html
                      └── header.html
                      └── leaflet_map.html
                      └── links.html
```

To override the templates you can copy the whole tree to another location and point QGIS_SERVER_API_RESOURCES_DIRECTORY to the new location.

Extra parameters supported by all request types

The following extra parameters are supported by all protocols.

- **FILE_NAME**: if set, the server response will be sent to the client as a file attachment with the specified file name.

  **Poznámka**: Not available for WFS3.

- **MAP**: Similar to MapServer, the MAP parameter can be used to specify the path to the QGIS project file. You can specify an absolute path or a path relative to the location of the server executable (qgis_mapserv.fcgi). If not specified, QGIS Server searches for .qgs files in the directory where the server executable is located.

  **Example**:  
  ```
  http://localhost/cgi-bin/qgis_mapserv.fcgi?
  REQUEST=GetMap&MAP=/home/qgis/projects/world.qgs&...
  ```

  **Poznámka**: You can define a QGIS_PROJECT_FILE as an environment variable to tell the server executable where to find the QGIS project file. This variable will be the location where QGIS will look for the project file. If not defined it will use the MAP parameter in the request and finally look at the server executable directory.
REDLINING

This feature is available and can be used with GetMap and GetPrint requests.

The redlining feature can be used to pass geometries and labels in the request which are overlapped by the server over the standard returned image (map). This permits the user to put emphasis or maybe add some comments (labels) to some areas, locations etc. that are not in the standard map.

The request is in the format:

http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi?map=/world.qgs&SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap...

&HIGHLIGHT_GEOM=POLYGON((590000 5647000, 590000 6110620, 2500000 6110620, 2500000...

&HIGHLIGHT_SYMBOL=<StyledLayerDescriptor><UserStyle><Name>Highlight</Name>

&HIGHLIGHT_LABELSTRING=Write label here

&HIGHLIGHT_LABELBUFFERSIZE=1.5

Here is the image outputed by the above request in which a polygon and a label are drawn on top of the normal map:

You can see there are several parameters in this request:

- **HIGHLIGHT_GEOM**: You can add POINT, MULTILINESTRING, POLYGON etc. It supports multipart geometries. Here is an example: HIGHLIGHT_GEOM=MULTILINESTRING((0 0, 0 1, 1 1)). The coordinates should be in the CRS of the GetMap/GetPrint request.

- **HIGHLIGHT_SYMBOL**: This controls how the geometry is outlined and you can change the stroke width, color and opacity.

- **HIGHLIGHT_LABELSTRING**: You can pass your labeling text to this parameter.

- **HIGHLIGHT_LABELSIZE**: This parameter controls the size of the label.

- **HIGHLIGHT_LABELCOLOR**: This parameter controls the label color.

- **HIGHLIGHT_LABELBUFFERCOLOR**: This parameter controls the label buffer color.

- **HIGHLIGHT_LABELBUFFERSIZE**: This parameter controls the label buffer size.

External WMS layers

QGIS Server allows including layers from external WMS servers in WMS GetMap and WMS GetPrint requests. This is especially useful if a web client uses an external background layer in the web map. For performance reasons, such layers should be directly requested by the web client (not cascaded via QGIS server). For printing however, these layers should be cascaded via QGIS server in order to appear in the printed map.

External layers can be added to the LAYERS parameter as EXTERNAL_WMS:<layername>. The parameters for the external WMS layers (e.g. url, format, dpiMode, crs, layers, styles) can later be given as service parameters <layername>:<parameter>. In a GetMap request, this might look like this:

http://localhost/qgisserver?

SERVICE=WMS&REQUEST=GetMap...

&LAYERS=EXTERNAL_WMS:basemap,layer1,layer2

(continues on next page)
Obr. 18.14: Server response to a GetMap request with redlining parameters
Similarly, external layers can be used in GetPrint requests:

```http
http://localhost/qgisserver?
SERVICE=WMS
...
&amp;REQUEST=GetPrint&amp;TEMPLATE=A4
&amp;map0:layers=EXTERNAL_WMS:basemap,layer1,layer2
&amp;map0:EXTENT=<minx,miny,maxx,maxy>
&amp;basemap:url=http://externalserver.com/wms.fcgi
&amp;basemap:format=image/jpeg
&amp;basemap:dpiMode=7
&amp;basemap:crs=EPSG:2056
&amp;basemap:layers=orthofoto
&amp;basemap:styles=default
```

18.2.3 Zásuvné moduly

Instalace

To install the HelloWorld example plugin for testing the servers, you firstly have to create a directory to hold server plugins. This will be specified in the virtual host configuration and passed on to the server through an environment variable:

```bash
mkdir -p /var/www/qgis-server/plugins
cd /var/www/qgis-server/plugins
wget https://github.com/elpaso/qgis-helloserver/archive/master.zip
unzip master.zip
mv qgis-helloserver-master HelloServer
```

HTTP Server configuration

Apache

To be able to use server plugins, FastCGI needs to know where to look. So, we have to modify the Apache configuration file to indicate the QGIS_PLUGINPATH environment variable to FastCGI:

```conf
FcgidInitialEnv QGIS_PLUGINPATH "/var/www/qgis-server/plugins"
```

Moreover, a basic HTTP authorization is necessary to play with the HelloWorld plugin previously introduced. So we have to update the Apache configuration file a last time:

```conf
# Needed for QGIS HelloServer plugin HTTP BASIC auth
<IfModule mod_fcgid.c>
  RewriteEngine on
  RewriteCond %{HTTP:Authorization} .
  RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

Then, restart Apache:
How to use a plugin

Test the server with the HelloWorld plugin:

```bash
wget -q -O - "http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=HELLO"
HelloServer!
```

You can have a look at the default GetCapabilities of the QGIS server at:

```plaintext
http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
```

### 18.2.4 Advanced configuration

**Logging**

To log requests sent to the server, you have to set the following environment variables:

- `QGIS_SERVER_LOG_LEVEL`
- `QGIS_SERVER_LOG_FILE`
- `QGIS_SERVER_LOG_STDERR`

Take a look on Environment variables to understand their meanings.

**Environment variables**

You can configure some aspects of QGIS Server by setting environment variables. According to the HTTP server and how you run QGIS Server, there are several ways to define these variables. This is fully described in HTTP Server configuration.
<table>
<thead>
<tr>
<th>Název</th>
<th>Popis</th>
<th>Default</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>QGIS_OPTIONS_PATH</td>
<td>Specifies the path to the directory with settings. It works the same way as QGIS application --optionspath option. It is looking for settings file in &lt;QGIS_OPTIONS_PATH&gt;/QGIS/QGIS3.ini.</td>
<td>;</td>
<td>All</td>
</tr>
</tbody>
</table>
| QUERY_STRING          | The query string, normally passed by the web server. This variable can be useful while testing QGIS server binary from the command line. For example for testing a GetCapabilities request on the command line to a project that also requires a PostgreSQL connection defined in a pg_service.conf file: 

```
PGSERVICEFILE=/etc/pg_service.conf
QUERY_STRING="MAP=/home/qgis/
projects/world.qgs&
SERVICE=WMS&
REQUEST=GetCapabilities" \\
/usr/lib/cgi-bin/qgis_mapserv.fcgi
```

The result should be either the content of the GetCapabilities response or, if something is wrong, an error message. | ;       | All      |
| QGIS_PROJECT_FILE     | The .qgs or .qgz project file, normally passed as a parameter in the query string (with MAP), you can also set it as an environment variable (for example by using mod_rewrite Apache module). Note that you may also indicate a project stored in PostgreSQL, e.g. `postgresql://localhost:5432?sslmode=disable&dbname=mydb&schema=myschema&project=myproject`. | ;       | All      |
| QGIS_SERVER_LOG_FILE  | Specify path and filename. Make sure that server has proper permissions for writing to file. File should be created automatically, just send some requests to server. If it's not there, check permissions. QGIS_SERVER_LOG_FILE is deprecated since QGIS 3.4. File logging support will be removed in QGIS 4.0. | ;       | All      |
| QGIS_SERVER_LOG_STDERR| Activate logging to stderr. This variable has no effect when QGIS_SERVER_LOG_FILE is set.  

- 0 or false (case insensitive) 
- 1 or true (case insensitive)  

false  | All      |
| MAX_CACHE_LAYERS      | Specify the maximum number of cached layers (default: 100).                                                                                                                                          | 100     | All      |
| QGIS_PLUGINPATH       | Useful if you are using Python plugins for the server, this sets the folder that is searched for Python plugins.                                                                                      | ;       | All      |
| QGIS_SERVER_LOG_LEVEL | Specify desired log level. Available values are:                                                                                                                                                    | 0       | All      |

- 0 or INFO (log all requests)  
- 1 or WARNING  
- 2 or CRITICAL (log just critical errors, suitable for production sessions)  

false  | WMS      |
| QGIS_SERVER_PARALLEL_RENDERING | Parallel rendering for WMS operations.                                                                                                                        | false  | WMS      |
Settings summary

When QGIS Server is starting, you have a summary of all configurable parameters thanks to environment variables. Moreover, the value currently used and the origin is also displayed.

For example with spawn-fcgi:

```
export QGIS_OPTIONS_PATH=/home/user/.local/share/QGIS/QGIS3/profiles/default/
export QGIS_SERVER_LOG_FILE=/home/user/qserv.log
export QGIS_SERVER_LOG_LEVEL=2
```

QGIS Server Settings:

- `QGIS_OPTIONS_PATH` / '' (Override the default path for user configuration): '/home/user/.local/share/QGIS/QGIS3/profiles/default/' (read from `ENVIRONMENT_VARIABLE`)
- `QGIS_SERVER_PARALLEL_RENDERING` / '/qgis/parallelRendering' (Activate/Deactivate parallel rendering for WMS getMap request): 'true' (read from `INI_FILE`)
- `QGIS_SERVER_MAX_THREADS` / '/qgis/max_threads' (Number of threads to use when parallel rendering is activated): '4' (read from `INI_FILE`)
- `QGIS_SERVER_LOG_LEVEL` / '' (Log level): '2' (read from `ENVIRONMENT_VARIABLE`)
- `QGIS_SERVER_LOG_FILE` / '' (Log file): '/tmp/qserv.log' (read from `ENVIRONMENT_VARIABLE`)
- `QGIS_PROJECT_FILE` / '' (QGIS project file): '' (read from `DEFAULT_VALUE`)
- `MAX_CACHE_LAYERS` / '' (Specify the maximum number of cached layers): '100' (read from `DEFAULT_VALUE`)
- `QGIS_SERVER_CACHE_DIRECTORY` / '/cache/directory' (Specify the cache directory): '/root/.local/share/QGIS/QGIS3/profiles/default/cache' (read from `DEFAULT_VALUE`)
- `QGIS_SERVER_CACHE_SIZE` / '/cache/size' (Specify the cache size): '52428800' (read from `INI_FILE`)

Ini file used to initialize settings: /home/user/.local/share/QGIS/QGIS3/profiles/default/QGIS/QGIS3.ini

In this particular case, we know that `QGIS_SERVER_MAX_THREADS` and `QGIS_SERVER_PARALLEL_RENDERING` values are read from the ini file found in `QGIS_OPTIONS_PATH` directory (which is defined through an environment variable). The corresponding entries in the ini file are `/qgis/max_threads` and `/qgis/parallel_rendering` and their values are `true` and `4` threads.
**Short name for layers, groups and project**

A number of elements have both a `<Name>` and a `<Title>`. The **Name** is a text string used for machine-to-machine communication while the **Title** is for the benefit of humans.

For example, a dataset might have the descriptive Title “Maximum Atmospheric Temperature” and be requested using the abbreviated **Name** “ATMAX”. User can already set title for layers, groups and project.

OWS name is based on the name used in layer tree. This name is more a label for humans than a name for machine-to-machine communication.

QGIS Server supports:

- short name line edits to layers properties You can change this by right clicking on a layer, choose Properties Metadata tab Description Short name.

- WMS data dialog to layer tree group (short name, title, abstract)

  By right clicking on a layer group and selecting the Set Group WMS data option you will get:

  ![Set group WMS data](image)

  **Obr. 18.15: Set group WMS data dialog**

  - short name line edits to project properties - add a regexp validator "^[A-Za-z][A-Za-z0-9\-_]*" to short name line edit accessible through a static method

  - add a regexp validator "^[A-Za-z][A-Za-z0-9\-_]*" to short name line edit accessible through a static method

  You can choose a short name for the project root by going to Project properties OWS Server Service capabilities Short name.

- add a TreeName element in the fullProjectSettings

If a short name has been set for layers, groups or project it is used by QGIS Server as the layer name.

**Connection to service file**

In order to make apache aware of the PostgreSQL service file (see the PostgreSQL Service connection file section) you need to make your `*.conf` file look like:

```
SetEnv PGSERVICEFILE /home/web/.pg_service.conf

<Directory "/home/web/apps2/bin/">
  AllowOverride None
  ....
```

540 Kapitola 18. Working with OGC Data
Add fonts to your linux server

Keep in mind that you may use QGIS projects that point to fonts that may not exist by default on other machines. This means that if you share the project, it may look different on other machines (if the fonts don’t exist on the target machine).

In order to ensure this does not happen you just need to install the missing fonts on the target machine. Doing this on desktop systems is usually trivial (double clicking the fonts).

For linux, if you don’t have a desktop environment installed (or you prefer the command line) you need to:

- On Debian based systems:

```
sudo su
mkdir -p /usr/local/share/fonts/truetype/myfonts && cd /usr/local/share/fonts/
-truetype/myfonts

# copy the fonts from their location
cp /fonts_location/* .
chown root *

chown root *
chown root *
```

- On Fedora based systems:

```
sudo su
mkdir /usr/share/fonts/myfonts && cd /usr/share/fonts/myfonts

# copy the fonts from their location
cp /fonts_location/* .
chown root *

chown root *
```

18.2.5 Kontejnerové nasazení

Existuje mnoho způsobů, jak používat kontejnerovou aplikaci, od nejjednodušších (jednoduché sestavení image Dockeru) po sofistikované (Kubernetes a tak dále).


Rada: Docker spouští předbalenou aplikaci (tzv. images), kterou lze načíst jako zdroje (Dockerfile a prostředky) k vytvoření nebo již vytvořené z registrů (soukromých nebo veřejných).

Jednoduché docker images

Protože docker image ve veřejném registru neexistuje, budete to muset postavit. Chcete-li tak učinit, vytvořte adresář: file: qgis-server a v jeho adresáři:

- vytvořte soubor :file: Dockerfile s tímto obsahem:

```
FROM debian:buster-slim
ENV LANG=en_EN.UTF-8
```

(continues on next page)
RUN apt-get update \
  && apt-get install --no-install-recommends --no-install-suggests --allow-unauthenticated -y \
  gnupg \n  ca-certificates \n  wget \n  locales \n  localedef -i en_US -f UTF-8 en_US.UTF-8 \n  && wget -O - https://qgis.org/downloads/qgis-2019.gpg.key | gpg --import \n  && gpg --export --armor 8D5A5B203548E5004487DD1951F523511C7028C3 | apt-key add .. \n  && echo "deb http://qgis.org/debian buster main" >> /etc/apt/sources.list.d/\n  && apt-get update \n  && apt-get install --no-install-recommends --no-install-suggests --allow-unauthenticated -y \
  qgis-server \n  spawn-fcgi \n  xauth \n  xvfb \n  && apt-get remove --purge -y \
  gnupg \n  wget \n  && rm -rf /var/lib/apt/lists/*

RUN useradd -m qgis

ENV TINI_VERSION v0.17.0
ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/tini /tini
RUN chmod +x /tini

ENV QGIS_PREFI_PATH /usr
ENV QGIS_SERVER_LOG_STDERR 1
ENV QGIS_SERVER_LOG_LEVEL 2

COPY cmd.sh /home/qgis/cmd.sh
RUN chown qgis:qgis /home/qgis/cmd.sh

USER qgis

WORKDIR /home/qgis

ENTRYPOINT ["/tini", "--"]

CMD ["/home/qgis/cmd.sh"]

• vytvořte soubor: file: cmd.sh s tímto obsahem:

```bash
#!/bin/bash

[[ $DEBUG == "1" ]] && env

exec /usr/bin/xvfb-run --auto-servernum --server-num=1 /usr/bin/spawn-fcgi -p 5555 -n -d /home/qgis -- /usr/lib/cgi-bin/qgis_mapserv.fcgi
```

• vytvořit image pomocí:

docker build -f Dockerfile -t qgis-server ./
První pokus

Ke spuštění serveru budete potřebovat soubor projektu QGis. Můžete použít jeden z vašich nebo vyberte [tuto ukázku](#).

Nyní můžete server spustit pomocí:

```
docker network create qgis
docker run --network qgis --rm --name qgis-server --hostname=qgis-server 
-v $(pwd)/data:/data:ro -p 5555:5555 
-e "QGIS_PROJECT_FILE=/data/osm.qgs" 
gis-server
```

Použitelné možnosti:
- `-d`: běží na pozadí
- `-rm`: remove the container when it is stopped
- `-name`: name of the container to be created
- `-net`: (previously created) sub network
- `-hostname`: container hostname, for later referencing
- `-v`: local data directory to be mounted in the container
- `-p`: host/container port mapping
- `-e`: environment variable to be used in the container

To check, type `docker ps | grep qgis-server` and you should see a line with `qgis-server`:

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
<th>CREATED</th>
<th>STATUS</th>
<th>NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>4de8192da76e</td>
<td>qgis-server</td>
<td>&quot;/tini -- /home/qgis...&quot;</td>
<td>3 seconds ago</td>
<td>Up 2 seconds</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0.0.0:5555-&gt;5555/tcp</td>
<td>qgis-server</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Použitelný vzorek

Protože server přijímá pouze připojení fastcgi, musíte mít server HTTP, který tento protokol zpracovává. K tomu musíme vytvořit jednoduchý konfigurační soubor Nginx a spustit image Nginx.

Create a file nginx.conf in the current directory with this content:

```
server {
  listen 80;
  server_name _;
  location / {
    root /usr/share/nginx/html;
    index index.html index.htm;
  }
  location /qgis-server {
    proxy_buffers 16 16k;
    proxy_buffer_size 16k;
    gzip off;
    include fastcgi_params;
    fastcgi_pass qgis-server:5555;
  }
}
```

A zadejte tento příkaz:
Chcete-li zkontrolovat dostupnost funkcí, zadejte do prohlížeče http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities

**Cleanup**

To cleanup the running images, type:

docker stop qgis-server nginx

**Docker stacks**

Předchozí metoda je skriptovatelná, ale není snadno zabalitelná, standardizovaná nebo snadno spravovatelná.

Chcete-li pracovat se sadou image dockeru, můžete použít docker stack spravovaný orchestrátorem. Ve stacku image fungují ve stejné privátní sítí a můžete spustit/zastavit celý stack nebo nasadit stack na jiné pracovníky. Existuje mnoho orchestrátorů, například Swarm, Kubernetes a Mesos.

V následujícím textu představíme jednoduché konfigurace pro účely testování. Nejsou vhodné pro výrobu.

**Swarm/docker-compose**

Docker now has its own orchestrator: Swarm (compatible with docker-compose files). You have to enable it (the Mac version will also work with Linux).

**Popis Stacku**

Now that you have Swarm working, create the service file (see deploy swarm) `qgis-stack.yaml`:

```
version: '3.7'

services:
  qgis-server:
    # Should use version with utf-8 locale support:
    image: qgis-server:latest
    volumes:
    - REPLACE_WITH_FULL_PATH/data:/data:ro
    environment:
    - LANG=en_EN.UTF-8
    - QGIS_PROJECT_FILE=/data/osm.qgs
    - QGIS_SERVER_LOG_LEVEL=0 # INFO (log all requests)
    - DEBUG=1 # display env before spawning QGIS Server

  nginx:
    image: nginx:1.13
    ports:
    - 8080:80
    volumes:
    - REPLACE_WITH_FULL_PATH/nginx.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
    - qgis-server
```

Chcete-li nasadit (nebo aktualizovat) Stack, zadejte:
docker stack deploy -c qgis-stack.yaml qgis-stack

Check the stack deployment status until you obtain 1/1 in the **replicas** column:

docker stack services qgis-stack

Something like:

```bash
<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>MODE</th>
<th>REPLICA</th>
<th>IMAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>gmx7ewivwsgt</td>
<td>qgis_nginx</td>
<td>replicated</td>
<td>1/1</td>
<td>nginx:1.13</td>
</tr>
<tr>
<td>10v2e7c143u3</td>
<td>qgis_qgis-server</td>
<td>replicated</td>
<td>1/1</td>
<td>qgis-server:latest</td>
</tr>
</tbody>
</table>
```

To check WMS capabilities, type in a web browser http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities

**Cleanup**

Chcete-li vyčistit, zadejte:

docker stack rm qgis-stack

**Kubernetes**

**Installation**

If you have a **Docker Desktop** installation, using Kubernetes (aka k8s) is pretty straightforward: *enable k8s.*

If not, follow the **minikube tutorial** or **microk8s** for Ubuntu.

As Kubernetes installation can be really complex, we will only focus on aspects used by this demo. For further / deeper information, check the **official documentation.**

**microk8s**

microk8s needs extra steps: you have to enable the registry and tag the qgis-server image in order to have Kubernetes find the created images.

Nejprve povolte registr:

```
microk8s enable dashboard dns registry
```

Poté označte a nahraje image do nově vytvořeného registru:

```
docker tag qgis-server 127.0.0.1:32000/qgis-server && docker push 127.0.0.1:32000/qgis-server
```

Finally, add or complete the `/etc/docker/daemon.json` to have your registry `127.0.0.1:32000` listed in the `insecure-registries` field:

```json
{
    "insecure-registries": ["127.0.0.1:32000"]
}
```
Creating manifests

Kubernetes popisuje objekty, které se mají nasadit do manifestů yaml. Existuje mnoho různých druhů, ale k vystavení nasazení interním nebo externím účelům použijeme pouze nasazení (manipulační pody, tj. Image dockeru) a služby.

Deployment manifests

Create a file `deployments.yaml` with this content:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: qgis-server
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      myLabel: qgis-server
  template:
    metadata:
      labels:
        myLabel: qgis-server
    spec:
      containers:
      - name: qgis-server
        image: localhost:32000/qgis-server:latest
        imagePullPolicy: IfNotPresent
        env:
          - name: LANG
            value: en_EN.UTF-8
          - name: QGIS_PROJECT_FILE
            value: /data/osm.qgs
          - name: QGIS_SERVER_LOG_LEVEL
            value: "0"
          - name: DEBUG
            value: "1"
        ports:
        - containerPort: 5555
      volumes:
      - name: qgis-data
        hostPath:
          path: REPLACE_WITH_FULL_PATH/data
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: qgis-nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      myLabel: qgis-nginx
  template:
    metadata:
```

(continues on next page)
labels:
  myLabel: qgis-nginx
spec:
  containers:
  - name: qgis-nginx
    image: nginx:1.13
    ports:
      - containerPort: 80
    volumeMounts:
      - name: nginx-conf
        mountPath: /etc/nginx/conf.d/default.conf
  volumes:
  - name: nginx-conf
    hostPath:
      path: REPLACE_WITH_FULL_PATH/nginx.conf

Service manifests

Create a file services.yaml with this content:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: qgis-server
  namespace: default
spec:
  type: ClusterIP
  selector:
    myLabel: qgis-server
  ports:
  - port: 5555
    targetPort: 5555
---
apiVersion: v1
kind: Service
metadata:
  name: qgis-nginx
  namespace: default
spec:
  type: NodePort
  selector:
    myLabel: qgis-nginx
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30080
```

Deploying manifests

To deploy the images and services in Kubernetes, one can use the dashboard (click on the + on the upper right) or the command line.

**Poznámka:** When using the command line with microk8s you will have to prefix each command with `microk8s`.

To deploy or update your manifests:
Chcete-li zkontrolovat, co je aktuálně nasazeno:

```bash
kubectl apply -k ./
```

Měli byste získat něco jako:

```plaintext
NAME                READY STATUS   RESTARTS AGE
pod/qgis-nginx-54845ff6f6-8skp9 1/1  Running  0 27m
pod/qgis-server-75df8ddd89-c7t7s 1/1  Running  0 27m
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>service/Kubernetes</td>
<td>~5h51m</td>
<td>ClusterIP</td>
<td>10.152.183.1</td>
<td>&lt;none&gt;</td>
<td>443/TCP</td>
</tr>
<tr>
<td>service/qgis-exec-server</td>
<td>~35m</td>
<td>ClusterIP</td>
<td>10.152.183.218</td>
<td>&lt;none&gt;</td>
<td>5555/TCP</td>
</tr>
<tr>
<td>service/qgis-nginx</td>
<td>~27m</td>
<td>NodePort</td>
<td>10.152.183.234</td>
<td>&lt;none&gt;</td>
<td>80:30080/TCP</td>
</tr>
<tr>
<td>service/qgis-server</td>
<td>~27m</td>
<td>ClusterIP</td>
<td>10.152.183.132</td>
<td>&lt;none&gt;</td>
<td>5555/TCP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>UP-TO-DATE</th>
<th>AVAILABLE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment.apps/qgis-nginx</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
<td>27m</td>
</tr>
<tr>
<td>deployment.apps/qgis-server</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
<td>27m</td>
</tr>
</tbody>
</table>

To read nginx/qgis logs, type:

```bash
kubectl logs -f POD_NAME
```

To check WMS capabilities, type in a web browser `http://localhost:30080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities`

**Cleanup**

Chcete-li vyčistit, zadejte:

```bash
kubectl delete -n default service/qgis-server service/qgis-nginx deployment/qgis-nginx deployment/qgis-server
```

**Cloud deployment**

Správa vlastního clusteru serverů za účelem nasazení kontejnerových aplikací je složitá práce. Musíte zvládnout více problémů, jako je hardware, šířka pásma a zabezpečení na různých úrovních.

Cloudová řešení pro nasazení mohou být dobrou alternativou, když se nechcete soustředit na správu infrastruktury.

A cloud deployment may use proprietary mechanisms, but they are also compatible with the stages explained previously (`docker images` and `stack management`).
AWS usecase

With Amazon AWS, through ECS (Elastic Container Service) functionalities, you can use docker-compose or Kubernetes compatible wrappers to manage your stack. You will have to create an image registry for your custom images to be accessible.

To use docker-compose alike functionalities, you need to install the `ecs-cli` client and have proper permissions / roles. Then, with the help of the `ecs-cli compose` commands (see the `ecs-cli compose manual` and `ecs-cli tutorial`), you can reuse the stack description.

To use Kubernetes, you can use the AWS web console or the command line tool `eksctl` and have the proper permissions / roles. Then with a well configured kubectl environment, you can reuse the Kubernetes manifests.
19.1 GPS Plugin

19.1.1 What is 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 tracklog or track 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.

Poznámka: QGIS supports also GNSS receivers. But we keep using the term GPS in this documentation.

19.1.2 Loading GPS data from a 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, you first need to load the plugin. Plugins Plugin Manager… opens the Plugin Manager Dialog. Activate the GPS Tools checkbox. When this plugin is loaded, a button with a small handheld GPS device will show up in the toolbar and in Layer Create Layer:

- GPS Tools
- Create new GPX Layer

For working with GPS data, we provide an example GPX file available in the QGIS sample dataset: qgis_sample_data/gps/national_monuments.gpx. See section Downloading sample data for more information about the sample data.
1. Select Vector GPS Tools or click the GPS Tools icon in the toolbar and open the Load GPX file tab (see figure_GPS).

2. Browse to the folder qgis_sample_data/gps/, select the GPX file national_monuments.gpx and click Open.

Obr. 19.1: The GPS Tools dialog window

Use the Browse... button to select the GPX file, then use the checkboxes to select the feature types you want to load from that GPX file. Each feature type will be loaded in a separate layer when you click OK. The file national_monuments.gpx only includes waypoints.

**Poznámka:** 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 [https://www.topografix.com/GPX/1/1/](https://www.topografix.com/GPX/1/1/).

### 19.1.3 GPSBabel

Since QGIS uses GPX files, you need a way to convert other GPS file formats to GPX. This can be done for many formats using the free program GPSBabel, which is available at [https://www.gpsbabel.org](https://www.gpsbabel.org). This program can also transfer GPS data between your computer and a GPS device. QGIS uses GPSBabel to do these things, so it is recommended that you install it. However, if you just want to load GPS data from GPX files you will not need it. Version 1.2.3 of GPSBabel is known to work with QGIS, but you should be able to use later versions without any problems.

### 19.1.4 Importing GPS data

To import GPS data from a file that is not a GPX file, you use the tool Import other file in the GPS Tools dialog. Here, you select the file that you want to import (and the file type), which feature type you want to import from it, where you want to store the converted GPX file and what the name of the new layer should be. Note that not all GPS data formats will support all three feature types, so for many formats you will only be able to choose between one or two types.
19.1.5 Downloading GPS data from a device

QGIS can use GPSBabel to download data from a GPS device directly as new vector layers. For this we use the Download from GPS tab of the GPS Tools dialog (see Figure_GPS_download). Here, we select the type of GPS device, the port that it is connected to (or USB if your GPS supports this), the feature type that you want to download, the GPX file where the data should be stored, and the name of the new layer.

![Figure_GPS_download](Image)

Obr. 19.2: The download tool

The device type you select in the GPS device menu determines how GPSBabel tries to communicate with your GPS device. If none of the available types work with your GPS device, you can create a new type (see section Defining new device types).

The port may be a file name or some other name that your operating system uses as a reference to the physical port in your computer that the GPS device is connected to. It may also be simply USB, for USB-enabled GPS units.

- On Linux, this is something like /dev/ttyS0 or /dev/ttyS1.
- On Windows, it is COM1 or COM2.

When you click OK, the data will be downloaded from the device and appear as a layer in QGIS.

19.1.6 Uploading GPS data to a device

You can also upload data directly from a vector layer in QGIS to a GPS device using the Upload to GPS tab of the GPS Tools dialog. To do this, you simply select the layer that you want to upload (which must be a GPX layer), your GPS device type, and the port (or USB) that it is connected to. Just as with the download tool, you can specify new device types if your device isn’t in the list.

This tool is very useful in combination with the vector-editing capabilities of QGIS. It allows you to load a map, create waypoints and routes, and then upload them and use them on your GPS device.

19.1.7 Defining new device types

There are lots of different types of GPS devices. The QGIS developers can’t test all of them, so if you have one that does not work with any of the device types listed in the Download from GPS and Upload to GPS tools, you can define your own device type for it. You do this by using the GPS device editor, which you start by clicking the Edit Devices button in the download or the upload tab.

To define a new device, you simply click the New Device button, enter a name, enter download and upload commands for your device, and click the Update Device button. The name will be listed in the device menus in the upload and download windows – it can be any string. The download command is the command that is used to download data from the device to a GPX file. This will probably be a GPSBabel command, but you can 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.
%type will be replaced by -w if you are downloading waypoints, -r if you are downloading routes and -t if you are downloading tracks. These are command-line options that tell GPSBabel which feature type to download.

%in will be replaced by the port name that you choose in the download window and %out will be replaced by the name you choose for the GPX file that the downloaded data should be stored in. So, if you create a device type with the download command gpsbabel %type -i garmin -o gpx %in %out (this is actually the download command for the predefined device type `Garmin serial') 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.

The upload command is the command that is used to upload data to the device. The same keywords are used, but %in is now replaced by the name of the GPX file for the layer that is being uploaded, and %out is replaced by the port name.

You can learn more about GPSBabel and its available command line options at https://www.gpsbabel.org.

Once you have created a new device type, it will appear in the device lists for the download and upload tools.

### 19.1.8 Download of points/tracks from GPS units

As described in previous sections QGIS uses GPSBabel to download points/tracks directly in the project. QGIS comes out of the box with a pre-defined profile to download from Garmin devices. Unfortunately there is a bug #6318 that does not allow create other profiles, so downloading directly in QGIS using the GPS Tools is at the moment limited to Garmin USB units.

**Garmin GPSMAP 60cs**

**MS Windows**

Install the Garmin USB drivers from https://www8.garmin.com/support/download_details.jsp?id=591

Connect the unit. Open GPS Tools and use type=garmin serial and port=usb: Fill the fields *Layer name* and *Output file*. Sometimes it seems to have problems saving in a certain folder, using something like c:\temp usually works.

**Ubuntu/Mint GNU/Linux**

It is first needed an issue about the permissions of the device, as described at https://wiki.openstreetmap.org/wiki/USB_Garmin_on_GNU/Linux. You can try to create a file /etc/udev/rules.d/51-garmin.rules containing this rule

```
ATTRS{idVendor}="091e", ATTRS{idProduct}="0003", MODE="666"
```

After that is necessary to be sure that the *garmin_gps* kernel module is not loaded

```
rmmod garmin_gps
```

and then you can use the GPS Tools. Unfortunately there seems to be a bug #7182 and usually QGIS freezes several times before the operation work fine.
BTGP-38KM datalogger (only Bluetooth)

MS Windows

The already referred bug does not allow to download the data from within QGIS, so it is needed to use GPSBabel from the command line or using its interface. The working command is

```
gpsbabel -t -i skytraq,baud=9600,initbaud=9600 -f COM9 -o gpx -F C:/GPX/aaa.gpx
```

Ubuntu/Mint GNU/Linux

Use same command (or settings if you use GPSBabel GUI) as in Windows. On Linux it maybe somehow common to get a message like

```
skytraq: Too many read errors on serial port
```

it is just a matter to turn off and on the datalogger and try again.

BlueMax GPS-4044 datalogger (both BT and USB)

MS Windows

Poznámka: It needs to install its drivers before using it on Windows 7. See in the manufacturer site for the proper download.

Downloading with GPSBabel, both with USB and BT returns always an error like

```
gpsbabel -t -i mtk -f COM12 -o gpx -F C:/temp/test.gpx
mtk_logger: Can't create temporary file data.bin
Error running gpsbabel: Process exited unsuccessfully with code 1
```

Ubuntu/Mint GNU/Linux

With USB

After having connected the cable use the dmesg command to understand what port is being used, for example /dev/ttyACM3. Then as usual use GPSBabel from the CLI or GUI

```
gpsbabel -t -i mtk -f /dev/ttyACM3 -o gpx -F /home/user/bluemax.gpx
```

With Bluetooth

Use Blueman Device Manager to pair the device and make it available through a system port, then run GPSBabel

```
gpsbabel -t -i mtk -f /dev/rfcomm0 -o gpx -F /home/user/bluemax_bt.gpx
```

19.2 Live GPS tracking

To activate live GPS tracking in QGIS, you need to select View Panels GPS Information Panel or press Ctrl+0. You will get a new docked window on the left side of the canvas.

There are four possible screens in this GPS tracking window:

- GPS position coordinates and an interface for manually entering vertices and features
- GPS signal strength of satellite connections
- GPS options screen (see figure_gps_options)
With a plugged-in GPS receiver (has to be supported by your operating system), a simple click on Connect connects the GPS to QGIS. A second click (now on Disconnect) disconnects the GPS receiver from your computer. For GNU/Linux, gpd support is integrated to support connection to most GPS receivers. Therefore, you first have to configure gpd properly to connect QGIS to it.

**Varování:** If you want to record your position to the canvas, you have to create a new vector layer first and switch it to editable status to be able to record your track.

### 19.2.1 Position and additional attributes

If the GPS is receiving signals from satellites, you will see your position in latitude, longitude and altitude together with additional attributes.

![GPS Information Panel](image)

Obr. 19.3: GPS tracking position and additional attributes
19.2.2 GPS signal strength

Here, you can see the signal strength of the satellites you are receiving signals from.

![GPS signal strength](image)

Obr. 19.4: GPS tracking signal strength

19.2.3 GPS options

In case of connection problems, you can switch between:

- **Autodetect**
- **Internal**
- **Serial device**
- **gpsd** (selecting the Host, Port and Device your GPS is connected to)

A click on *Connect* again initiates the connection to the GPS receiver.

You can activate **Automatically save added features** when you are in editing mode. Or you can activate **Automatically add points** to the map canvas with a certain width and color.

Activating **Cursor**, you can use a slider to shrink and grow the position cursor on the canvas.

You can also set an **Acquisition interval (seconds)** and a **Distance threshold (meters)** parameters to keep the cursor still active when the receiver is in static conditions.

Activating **Map centering** allows you to decide in which way the canvas will be updated. This includes 'always', 'when leaving', if your recorded coordinates start to move out of the canvas, or 'never', to keep map extent.

Finally, you can activate **Log file** and define a path and a file where log messages about the GPS tracking are logged.

If you want to set a feature manually, you have to go back to **Position** and click on *Add Point* or *Add Track Point*. 
Obr. 19.5: GPS tracking options window
19.2.4 Connect to a Bluetooth GPS for live tracking

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.

Remember the number of the COM port assigned to the GPS connection as resulting by the Bluetooth properties.

After the GPS has been recognized, make the pairing for the connection. Usually the authorization code is 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.

19.2.5 Using 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 just click Connect in the Live tracking panel using the Autodetect mode.

**Ubuntu/Mint GNU/Linux**

As for Windows the easiest way is to use a server in the middle, in this case GPSD, so

```
sudo apt install gpsd
```

Then load the garmin_gps kernel module

```
sudo modprobe garmin_gps
```

And then connect the unit. Then check with dmesg the actual device being used bu the unit, for example /dev/ttyUSB0. Now you can launch gpsd

```
gpsd /dev/ttyUSB0
```

And finally connect with the QGIS live tracking tool.
19.2.6 Using BTGP-38KM datalogger (only Bluetooth)

Using GPSSD (under Linux) or GPSGate (under Windows) is effortless.

19.2.7 Using BlueMax GPS-4044 datalogger (both BT and 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 the right port.

**Ubuntu/Mint GNU/Linux**

**For USB**

The live tracking works both with GPSSD

```
gpsd /dev/ttyACM3
```

or without it, by connecting the QGIS live tracking tool directly to the device (for example `/dev/ttyACM3`).

**For Bluetooth**

The live tracking works both with GPSSD

```
gpsd /dev/rfcomm0
```

or without it, by connecting the QGIS live tracking tool directly to the device (for example `/dev/rfcomm0`).
20.1 Authentication System Overview

Obr. 20.1: Anatomy of authentication system
20.1.1 Authentication database

The new authentication system stores authentication configurations in an SQLite database file located, by default, at `<profile directory>/qgis-auth.db`.

This authentication database can be moved between QGIS installations without affecting other current QGIS user preferences, as it is completely separate from normal QGIS settings. A configuration ID (a random 7-character alphanumeric string) is generated when initially storing a configuration to the database. This represents the configuration, thereby allowing the ID to be stored in plain text application components, (such as project, plugin, or settings files) without disclosure of its associated credentials.

**Poznámka:** The parent directory of the `qgis-auth.db` can be set using the following environment variable, `QGIS_AUTH_DB_DIR_PATH`, or set on the command line during launch with the `--authdbdirectory` option.

20.1.2 Master password

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.

![Image of Enter Credentials dialog](Obr. 20.2: Input new master password)

**Poznámka:** A path to a file containing the master password can be set using the following environment variable, `QGIS_AUTH_PASSWORD_FILE`.

**Managing the master password**

Once set, the master password can be reset; the current master password will be needed prior to resetting. During this process, there is an option to generate a complete backup of the current database.

If the user forgets the master password, there is no way to retrieve or override it. There is also no means of retrieving encrypted information without knowing the master password.

If a user inputs their existing password incorrectly three times, the dialog will offer to erase the database.
20.1.3 Authentication Configurations

You can manage authentication configurations from Configurations in the Authentication tab of the QGIS Options dialog (Settings → Options).

Use the button to add a new configuration, the button to remove configurations, and the button to modify existing ones.

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.

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 Authentication Methods).

20.1.4 Authentication Methods

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.

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.
Obr. 20.5: Configurations editor

Obr. 20.6: Adding config from within Configuration editor
Obr. 20.7: WMS connection dialog showing Add, Edit, and Remove authentication configuration buttons

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Works with</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>Basic authentication</td>
<td>postgres, db2, ows, wfs, wcs, wms, ogr, gdal, proxy</td>
</tr>
<tr>
<td>EsriToken</td>
<td>ESRI token based authentication</td>
<td>arcgismapserver, arcgisfeatureserver</td>
</tr>
<tr>
<td>Identity-Cert</td>
<td>PKI stored identity certificate</td>
<td>ows, wfs, wcs, wms</td>
</tr>
<tr>
<td>OAuth2</td>
<td>OAuth2 authentication</td>
<td>ows, wfs, wcs, wms</td>
</tr>
<tr>
<td>PKI-Paths</td>
<td>PKI paths authentication</td>
<td>ows, wfs, wcs, wms, postgres</td>
</tr>
<tr>
<td>PKI-PKCS#12</td>
<td>PKI PKCS#12 authentication</td>
<td>ows, wfs, wcs, wms, postgres</td>
</tr>
</tbody>
</table>

Obr. 20.8: Available method plugins list

Obr. 20.9: Basic HTTP authentication configs
Obr. 20.10: ESRI Token authentication configs
OAuth2 authentication configs

**Name**: Required

**Resource**: Optional URL resource

**OAuth2 authentication**

- **Configure**
  - Defined
  - Software

- **Grant Flow**: Authorization Code

- **Description**: 

- **Request URL**: Required

- **Token URL**: Required

- **Refresh Token URL**: Optional

- **Redirect URL**: Optional

- **Client ID**: Required

- **Client Secret**: Required

- **Scope**: Optional (space delimiter)

- **API Key**: Optional

**Advanced**

- **Token Session**: Persist between launches

- **Access Method**: Header

- **Request Timeout**: 30 seconds

**Extra initial request parameters**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value (unencoded)</th>
</tr>
</thead>
</table>

*Note: Saving writes directly to authentication database*

[Image of a form with fields for OAuth2 authentication configurations]
Obr. 20.12: PKI paths authentication configs

Obr. 20.13: PKI PKCS#12 file paths authentication configs

Obr. 20.14: Stored Identity authentication configs
Poznámka: 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.

20.1.5 Master Password and Auth Config Utilities

Under the Options menu (Settings > Options) in the Authentication tab, there are several utility actions to manage the authentication database and configurations:

- **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.
20.1.6 Using authentication configurations

Typically, an authentication configuration is selected in a configuration dialog for a network services (such as WMS). However, the selector widget can be embedded anywhere authentication is needed or in non-core functionality, like in third-party PyQGIS or C++ plugins.

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.

20.1.7 Python bindings

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 Security Considerations concerning Python access.
20.2 User Authentication Workflows

20.2.1 HTTP(S) authentication

One of the most common resource connections is via HTTP(S), e.g. web mapping servers, and authentication method plugins often work for these types of connections. Method plugins have access to the HTTP request object and can manipulate both the request as well as its headers. This allows for many forms of internet-based authentication. When connecting via HTTP(S) using the standard username/password authentication method will attempt HTTP BASIC authentication upon connection.

20.2.2 Database authentication

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 new auth system, the key=value will be an abstracted representation of the credentials, e.g. authfg=81t21b9.

20.2.3 PKI authentication

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.

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.

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.
Obr. 20.20: Konfigurace WMS připojení pro HTTP BASIC

Obr. 20.21: Konfigurace Postgres SSL s klíčem PKI
Obr. 20.22: PKI configuration workflow
Poznámka: 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.

Authorities

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.)

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.

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.

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.

Varování: Changing the default trust policy may result in problems with secure connections.
Obr. 20.24: Authorities editor

Obr. 20.25: Certificate info dialog
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.

**Poznámka:** If you are pasting certificate information into the PEM text field, note that encrypted certificates are not supported.
Identities

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.

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.

20.2.4 Handling bad layers

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.

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.
Obr. 20.30: PEM/DER identity import

Obr. 20.31: PKCS#12 identity import

Obr. 20.32: Handle bad layers with authentication
20.2.5 Changing authentication config ID

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 and 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.

---

**Varování**: 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.
20.2.6 QGIS Server support

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:

- Authentication database needs to be available
- Authentication database’s master password needs to be available

When instantiating the authentication system, Server will create or use `qgis-auth.db` in `~/.qgis2/` 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.

**Poznámka:** `QGIS_AUTH_PASSWORD_FILE` variable will be removed from the Server environment immediately after accessing.

20.2.7 SSL server exceptions

![Diagram of SSL server exceptions](image)

Obr. 20.35: SSL server exception

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.

**Varování:** 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.

**Poznámka:** 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):

Once an SSL configuration is saved to the database, it can be edited or deleted.
Obr. 20.36: Manually adding configuration
Obr. 20.37: Přidávání konfigurace při SSL chybě

Obr. 20.38: Existující SSL konfigurace
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):

```python
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.

### 20.3 Security Considerations

Once the master password is entered, the API is open to access authentication configs in the authentication database, similar to how Firefox works. However, in the initial implementation, no wall against PyQGIS access has been defined. This may lead to issues where a user downloads/installs a malicious PyQGIS plugin or standalone app that gains access to authentication credentials.

The quick solution for initial release of feature is to just not include most PyQGIS bindings for the authentication system.

Another simple, though not robust, fix is to add a combobox in Settings » Options » Authentication (defaults to „never“):

```
"Allow Python access to authentication system"
Choices: [ confirm once per session | always confirm | always allow | never]
```

Such an option’s setting would need to be saved in a location non-accessible to Python, e.g. the authentication database, and encrypted with the master password.

- Another option may be to track which plugins the user has specifically
- 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.

• Alternatively, access to sensitive authentication system data from Python 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.

20.3.1 Restrictions

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-oss1 (OpenSSL) plugin is loaded at run-time. The qca-oss1 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-oss1 is not found at run-time.
GRASS GIS Integration

GRASS integration provides access to GRASS GIS databases and functionalities (see GRASS-PROJECT in Literature and Web References). 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.

21.1 Demo dataset

As an example, we will use the QGIS Alaska dataset (see section Downloading sample data). 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/sample-data/.

21.2 Loading GRASS raster and vector layers

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.

Tip: GRASS Data Loading

If you don’t see GRASS location item, verify in Help About Providers if GRASS vector provider is loaded.
21.3 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 GRASS Options.

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.

21.4 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.

21.5 GRASS Options

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.

21.6 Starting the GRASS plugin

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
- Close Mapset
- Open GRASS Tools
- Display Current GRASS Region
- GRASS Options
21.7 Opening GRASS mapset

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.

21.8 GRASS LOCATION and 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 subregions, or as workspaces for individual team members (see Neteler & Mitasova 2008 in Literature and Web References). In order to analyse 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/OGR-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.)

21.9 Importing data into a GRASS LOCATION

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 Downloading sample data).

1. Start QGIS and make sure the GRASS plugin is loaded.
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. Now click the Open GRASS tools icon. The GRASS Toolbox (see section The GRASS Toolbox) dialog appears.
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. Browse to the folder `raster` in the QGIS 'Alaska' dataset and select the file `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. Browse to the folder `gml` in the QGIS 'Alaska' dataset and select the file `lakes.gml` as OGR file.

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 *Successfully finished*, click *View Output*. The `lakes_grass` vector layer is now imported into GRASS and will be visualized in the QGIS canvas.

### 21.9.1 Creating a new GRASS LOCATION

As an example, here is the sample GRASS LOCATION `alaska`, which is projected in the Albers Equal Area projection using feet as units. This sample GRASS LOCATION `alaska` will be used for all examples and exercises in the following GRASS-related sections. It is useful to download and install the dataset on your computer (see *Downloading sample data*).

1. Start QGIS and make sure the GRASS plugin is loaded.

2. Visualize the `alaska.shp` shapefile (see section *Loading a layer from a file*) from the QGIS Alaska dataset (see *Downloading sample data*).

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 *Adding a new MAPSET*) or to create a new LOCATION altogether. Select `Create new location` (see figure `grass_new_location`).

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 *Práce s projekcemi*)).


10. Click *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. Click *Next*. 

Kapitola 21. GRASS GIS Integration
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 Literature and Web References).

14. Check out the summary to make sure it's correct and click Finish.

15. The new LOCATION, 'alaska', and two MAPSETS, 'demo' and 'PERMANENT', are created. The currently opened working set is 'demo', as you defined.

16. Notice that some of the tools in the GRASS toolbar that were disabled are now enabled.

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 Importing data into a GRASS LOCATION). You can also use the already-existing vector and raster data in the sample GRASS LOCATION 'alaska', included in the QGIS 'Alaska' dataset Downloading sample data, and move on to section The GRASS vector data model.

21.9.2 Adding a new 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 Literature and Web References, and section The GRASS region tool).

1. Start QGIS and make sure the GRASS plugin is loaded.

2. In the GRASS toolbar, click on the icon to bring up the MAPSET wizard.

3. Select the GRASS database (GISDBASE) folder grassdata with the LOCATION 'alaska', where we want to add a further MAPSET called 'test'.

4. Click 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 figure_grass_new_location) 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.
The GRASS vector data model

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.

Attributes can be stored inside the GRASS LOCATION as dBase, SQLite3 or in external database tables, for example, PostgreSQL, MySQL, Oracle, etc.

Attributes in database tables are linked to geometry elements using a 'category' value. 'Category' (key, ID) is an integer attached to geometry primitives, and it is used as the link to one key column in the database table.

Tip: Learning the GRASS Vector Model

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/documentation/manuals/ for more information, books and tutorials in several languages.

Creating a new GRASS vector layer

To create a new GRASS vector layer, select one of following items from mapset context menu in the browser:

- New Point Layer
- New Line Layer
- New Polygon Layer

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).
21.12 Digitizing and editing a GRASS vector layer

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

VAROVÁNÍ: 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 vectormap if changes are written to the map.

Toolbar

The ‘Digitizing Toolbar’ has some specific tools when a GRASS layer is edited:

<table>
<thead>
<tr>
<th>Íkonė</th>
<th>Tool</th>
<th>Účel</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌝</td>
<td>New Point</td>
<td>Digitize new point</td>
</tr>
<tr>
<td>🌅</td>
<td>New Line</td>
<td>Digitize new line</td>
</tr>
<tr>
<td>🌞</td>
<td>New Boundary</td>
<td>Digitize new boundary</td>
</tr>
<tr>
<td>🌞</td>
<td>New Centroid</td>
<td>Digitize new centroid (label existing area)</td>
</tr>
<tr>
<td>🌞</td>
<td>New Closed Boundary</td>
<td>Digitize new closed boundary</td>
</tr>
</tbody>
</table>

Table GRASS Digitizing: GRASS Digitizing Tools

Tip: Digitizing polygons in 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.

**Attributes**

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.

**Tip:** 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.

**Tip:** 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.

**Tip:** Do not remove 'topo_symbol' field from attribute table, that would make features invisible because the renderer is based on that column.

**Uchycení**

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.

**Tip:** Use layer's CRS also for canvas when editing.

**Limitations**

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.

**Tip:** GRASS Edit Permissions

You must be the owner of the GRASS MAPSET you want to edit. It is impossible to edit data layers in a MAPSET that is not yours, even if you have write permission.
21.13 The GRASS region tool

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 The GRASS Toolbox.

21.14 The GRASS Toolbox

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.
21.14.1 Working with GRASS modules

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 3.10 is available in the GRASS wiki at https://grasswiki.osgeo.org/wiki/GRASS-QGIS_relevant_module_list.

It is also possible to customize the GRASS Toolbox content. This procedure is described in section Customizing the GRASS Toolbox.

As shown in figure_grass_toolbox, 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.

Options

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.

![Obr. 21.4: GRASS Toolbox Module Options](image)

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.

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
21.14. The GRASS Toolbox

Obr. 21.5: GRASS Toolbox Module Output

Obr. 21.6: 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.

Tip: Display results immediately
If you want to display your calculation results immediately in your map canvas, you can use the 'View Output' button at the bottom of the module tab.

21.14.2 GRASS module examples
The following examples will demonstrate the power of some of the GRASS modules.

Creating contour lines
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 Importing data into a GRASS LOCATION.

- 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.
- In the list of tool categories, double-click 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 Working with GRASS modules).
- 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.)
- Type into the Name for output vector map the name 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.

Since this is a large region, it will take a while to display. After it finishes rendering, you can open the layer properties window to change the line color so that the contours appear clearly over the elevation raster, as in The Vector Properties Dialog.

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.

Tip: The simplify tool
Note that QGIS has a Vector Geometry Tools Simplify geometries tool that works just like the GRASS v.generalize Douglas-Peucker 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.

• Check that the .ctour_100' vector appears as the Name of input vector.

• 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.

• You may change the color of the vector to display it clearly on the raster background and to contrast with the original contour lines. You will notice that the new contour lines have smoother corners than the original while staying faithful to the original overall shape.

Tip: Other uses for 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.

Creating a Hillshade 3-D effect

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.

• Then click r.shaded.relief to open the module.

• Change the azimuth angle 270 to 315.

• Enter gtopo30_shade for the new hillshade raster, and click Run.

• When the process completes, add the hillshade raster to the map. You should see it displayed in grayscale.

• 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%.
You should now have the `gtopo30` elevation with its colormap and transparency setting displayed **above** the grayscale hillshade map. In order to see the visual effects of the hillshading, turn off the `gtopo30_shade` map, then turn it back on.

**Using the GRASS shell**

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.

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, switch 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.

![Image](image_url)

Obr. 21.9: Displaying shaded relief created with the GRASS module r.shaded.relief

**Raster statistics in a vector map**

The next example shows how a GRASS module can aggregate raster data and add columns of statistics for each polygon in a vector map.

- Again using the Alaska data, refer to *Importing data into a GRASS LOCATION* 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).
- From the Toolbox, choose *Vector Manage features*, and open the module **v.centroids**.
- Enter as the output vector map 'forest_areas' and run the module.
- 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 \(\text{Unique value}\) and set the Classification field to 'VEGDESC'. (Refer to the explanation of the symbology tab in Symbology Properties of the vector section.)
- Next, reopen the GRASS Toolbox and open *Vector Vector update by other maps*.
- Click on the **v.rast.stats** module. Enter gtopo30 and 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.
21.14.3 Customizing the GRASS Toolbox

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.

A sample XML file for generating the module `v.buffer` looks like this:

```xml
<?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" />
  <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.
This chapter introduces the QGIS processing framework, a geoprocessing environment that can be used to call native and third-party algorithms from QGIS, making your spatial analysis tasks more productive and easy to accomplish.

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. Close the dialog.

A Processing menu is now available in the top menu bar. From there you can reach the main components of this framework.

In the following sections, we will review how to use the graphical elements of this framework and make the most out of each one of them.

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.
- The Graphical Modeler: Several algorithms can be combined graphically using the modeler to define a workflow, creating a single process that involves several subprocesses.
- 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.
- The Batch Processing interface: This interface allows you to execute batch processes and automate the execution of a single algorithm on multiple datasets.

In the following sections, we will review each one of these elements in detail.
Obr. 22.1: Nástroje zpracování

Obr. 22.2: Modelář zpracování
Obr. 22.3: Processing History

Obr. 22.4: Batch Processing interface
22.2 Configuring the Processing Framework

As has been mentioned, the configuration menu gives access to a new dialog where you can configure how algorithms work. Configuration parameters are structured in separate blocks that you can select on the left-hand side of the dialog.

Along with the aforementioned *Output folder* entry, the *General* block contains parameters for setting the default rendering style for output layers (that is, layers generated by using algorithms from any of the framework GUI components). 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.

![Obr. 22.5: Rendering Styles](image)

Select the style file (.qml) that you want for each output and press *OK*.

Other configuration parameters in the *General* group are listed below:

- *Use filename as layer name*. 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, then 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.

- *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.

- *Use only selected features*. If this option is selected, whenever a vector layer is used as input for an algorithm, only its selected features will be used. If the layer has no selected features, all features will be used.

- *Pre-execution script file* and *Post-execution script file*. These parameters refer to scripts written using the processing scripting functionality, and are explained in the section covering scripting and the console.

Apart from the *General* block in the settings dialog, you will also find a block for algorithm providers. Each entry in this block contains an *Activate* item that you can use to make algorithms appear or not in the toolbox. Also, some algorithm providers have their own configuration items, which we will explain later when covering particular algorithm providers.
22.3 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.

![Processing Toolbox](image)

Obr. 22.6: Nástroje zpracování

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.

In the upper part of the toolbox dialog, you will find a set of tools to:

- 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.

**Poznámka:** At the top of the list of algorithms are displayed the most recent used tools; handy if you want to reexecute any.

![Processing Toolbox showing search results](image)

Obr. 22.7: Processing Toolbox showing search results

To execute a tool, just double-click on its name in the toolbox.

### 22.3.1 The algorithm dialog

Once you double-click on the name of the algorithm that you want to execute, a dialog similar to that in the figure below is shown (in this case, the dialog corresponds to the **Centroids** algorithm).

This dialog 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.

Although the number and type of parameters depend on the characteristics of the algorithm, the structure is similar for all of them. The parameters found in the table can be of one of the following types.

- A **raster layer**, to select from a list of all such layers available (currently opened) 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.

- A **vector layer**, to select from a list of all vector layers available in QGIS. Layers not loaded in QGIS can be selected as well, as in the case of raster layers, but only if the algorithm does not require a table field selected from the attributes table of the layer. In that case, only opened layers can be selected, since they need to be open so as to retrieve the list of field names available.

  You will see an iterator button by each vector layer selector, as shown in the figure below.

  If the algorithm contains several of them, you will be able to toggle just one of them. If the button corresponding to a vector input is toggled, the algorithm will be executed iteratively on each one of its features, instead of
Obr. 22.8: Algorithm Dialog - Parameters

Obr. 22.9: Vector iterator button
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.

**Poznámka:** By default, the parameters dialog will show a description of the CRS of each 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.

- A **table**, to select from a list of all available in QGIS. Non-spatial tables are loaded into QGIS like vector layers, and in fact they are treated as such by the program. Currently, the list of available tables that you will see when executing an algorithm that needs one of them is restricted to tables coming from files in dBase (.dbf) or Comma-Separated Values (.csv) formats.
- 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 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 northermost coordinate of another one.

![Expression based Input](image)

**Obr. 22.10:** Vstup založený na výrazu

- 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**, to be entered by four numbers representing its xmin, xmax, ymin, ymax limits. Clicking on the button on the right-hand side of the value selector, a pop-up menu will appear, giving you options:
  - to select the value from a layer or the current canvas extent;
  - or to define it by dragging directly onto the map canvas.

If you select the first option, you will see a window like the next one.
If you select the second one, the parameters window will hide itself, so you can click and drag onto the canvas. Once you have defined the selected rectangle, the dialog will reappear, containing the values in the extent text box.

- 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.

- A small table to be edited by the user. These are used to define parameters like lookup tables or convolution kernels, among others.

  Click on the button on the right side to see the table and edit its values.

  Depending on the algorithm, the number of rows can be modified or not by using the buttons on the right side of the window.

**Poznámka:** Some algorithms require many parameter to run, e.g. in the *Raster calculator* 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.

Along with the *Parameters* tab, there is another tab named *Log* (see figure below). Information provided by the algorithm during its execution is written in this tab, and allow you to track the execution and be aware and have more details about the algorithm as it runs. Notice that not all algorithms write information to this tab, and many of them might run silently without producing any output other than the final files.
Obr. 22.14: Multiple Selection

Obr. 22.15: Pevná tabulka
Obr. 22.16: 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.

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).

**A note on projections**

Processing algorithm execution are always performed in the input layer coordinate reference system (CRS). Due to QGIS's on-the-fly reprojection 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.

You still can execute the algorithm, but be aware that in most cases that will produce wrong results, such as empty layers due to input layers not overlapping.

**Tip:** 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.

### 22.3.2 Data objects generated by algorithms

Data objects generated by an algorithm can be of any of the following types:

- A raster layer
- A vector layer
- A table
- An HTML file (used for text and graphical outputs)

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 textbox (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.

For all the other types of output, you will find a checkbox that you can use to tell the algorithm whether to load the file once it is generated by the algorithm or not. By default, all files are opened.

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).

### 22.4 The history manager

#### 22.4.1 The processing history

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 easy to track and control all the work that has been developed using the Processing framework, and to reproduce it.

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.

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. When you right-click on an entry, you can Create Test... using the con-
cerned algorithm and parameters, following instructions at https://github.com/qgis/QGIS/blob/release-3_10/python/plugins/processing/tests/README.md.

22.4.2 The processing log

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.

22.5 The graphical modeler

The graphical modeler 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 graphical modeler, 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 graphical modeler can be opened from the Processing menu (Processing → Graphical Modeler).

The modeler has a working canvas where the structure of the model and the workflow it represents are shown. The left part of the window is a panel with two tabs that can be used to add new elements to the model.

Creating a model involves two 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 it happens with all the 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.
Obr. 22.18: Modelář
22.5.1 Definition of inputs

The first step is to define the inputs for the model. The following elements are found in the Inputs tab on the left side of the modeler window:

- Authentication Configuration
- Boolean
- SRS
- Color
- Vzdálenost
- Enum
- Výraz
- Rozsah
- Mapovač polí
- Soubor/Složka
- Mapová vrstva
- Matice
- Vícenásobný vstup
- Číslo
- Bod
- Print Layout
- Print Layout Item
- Rozsah
- Rastrové pásmo
- Rastrová vrstva
- Scale
- Řetězec
- Vektorové prvky
- Vektorové pole
- Vektorová vrstva

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 basic element (the description, which is what the user will see when executing the model). 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.

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.

You can also add inputs by dragging the input type from the list and dropping it in the position where you want it in the modeler canvas.
Obr. 22.19: Model Parameters Definition

Obr. 22.20: Model Parameters
22.5.2 Definition of the workflow

Once the inputs have been defined, it is time to define the algorithms of the model. Algorithms can be found in the Algorithms tab, grouped much in the same way as they are in the Processing toolbox.

To add an algorithm to a model, double-click on its name or drag and drop it, just like for inputs. 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 ones shown next correspond to the QGIS 'Drape (set Z value from raster)' algorithm and the QGIS 'Climb along line' algorithm.

As you can see, some differences exist. Instead of the file output box that was used to set the file path for output layers and tables, a simple text box is used here. If the layer generated by the algorithm is just a temporary result that will be used as the input of another algorithm and should not be kept as a final result, just do not edit that text box. Typing anything in it means that the result is final and the text that you supply will be the description for the output, which will be the output the user will see when executing the model.

Selecting the value of each parameter is also a bit different, since there are important differences between the context of the modeler and that of the toolbox. Let’s see how to introduce the values for each type of parameter.

- Layers (raster and vector) and tables. These are selected from a list, but in this case, the possible values are not the layers or tables currently loaded in QGIS, but the list of model inputs of the corresponding type, or other layers or tables generated by algorithms already added to the model.
- Numerical values. Literal values can be introduced directly in the text box. Clicking on the button beside the text box, expressions can be entered. Available variables for expressions include numerical inputs of the model, outputs from model algorithms and also statistical values from available layers within the model.
" String. Literal strings can be typed in the corresponding text box. Clicking on the button beside the text box, expressions can be entered, as for numerical values.

- Vector Field. The fields of a vector layer cannot be known at design time, since they depend on the selection of the user each time the model is executed. To set the value for this parameter, type the name of a field directly in the text box, or use the list to select a table field. The validity of the selected field will be checked at run time.

In all cases, you will find an additional parameter named Parent algorithms 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, which 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 Parent algorithms 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. This is useful to make the structure of the model more clear and intuitive. Links between elements are updated automatically. You can zoom in and out by using the mouse wheel.

You can run your algorithm any time by clicking on the Run button. 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.
Obr. 22.23: A complete model
22.5.3 Saving and loading models

Use the Save button to save the current model and the Open button to open any previously saved model. Models are saved with the .model3 extension. If the model has been 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.

The models folder can be set from the Processing configuration dialog, under the Modeler group.

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 an image, PDF or SVG

A model can also be exported as an image, SVG or PDF (for illustration purposes).

22.5.4 Editing a 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:

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:

![Could not remove element](image)

Obr. 22.25: Cannot Delete Algorithm

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.

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.

![Model With Deactivated Algorithms](image)

Obr. 22.26: Model With Deactivated Algorithms

When right-clicking on an algorithm that is not active, you will see a *Activate* menu option that you can use to reactivate it.

### 22.5.5 Editing model help files and meta-information

You can document your models from the modeler itself. Just click on the *Edit Model Help* button, and a dialog like the one shown next will appear.

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. 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.

Model help is saved as part of the model itself.
Obr. 22.27: Editing Help
22.5.6 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 of creating such a Python script is to create a model and then to export it as a Python file.

To do so, right click on the name of the model in the Processing Toolbox and choose Export Model as Python Algorithm….

22.5.7 About available algorithms

You might notice that 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.

22.6 The batch processing interface

22.6.1 Úvod

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.

To execute an algorithm as a batch process, right-click on its name in the toolbox and select the Execute as batch process option in the pop-up menu that will appear.

![Obr. 22.28: Batch Processing from right-click](image)

If you have the execution dialog of the algorithm open, you can also start the batch processing interface from there, clicking on the Run as batch process… button.

22.6.2 The parameters table

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.

Each line of this table 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.

By default, the table contains just two rows. You can add or remove rows using the buttons on the lower part of the window.

Once the size of the table has been set, it has to be filled with the desired values.
22.6. The batch processing interface

Obr. 22.29: Batch Processing From Algorithm Dialog

Obr. 22.30: Batch Processing
22.6.3 Filling the parameters table

For most parameters, setting the value is trivial. Just type the value or select it from the list of available options, depending on the parameter type.

Filenames for input data objects are introduced directly typing or, more conveniently, clicking on the … button on the right hand of the cell, which will show a context menu with two option: one for selecting from the layers currently opened and another to select from the filesystem. This second option, when selected, shows a typical file chooser dialog. Multiple files can be selected at once. If the input parameter represents a single data object and several files are selected, each one of them will be put in a separate row, adding new ones if needed. If the parameter represents a multiple input, all the selected files will be added to a single cell, separated by semicolons (,).

Layer identifiers can be directly typed in the parameter text box. You can enter the full path to a file or the name of a layer that is currently loaded in the current QGIS project. The name of the layer will be automatically resolved to its source path. Notice that, if several layers have the same name, this might cause unexpected results due to ambiguity.

Output data objects are always saved to a file and, unlike when executing an algorithm from the toolbox, saving to a temporary file or database is not permitted. You can type the name 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 autocompletion of other cells in the same column (same parameter).

If the default value (‘Do not autocomplete’) 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. This way, it is much easier to fill the table, and the batch process can be defined with less effort.

Automatic filling can be done by simply adding correlative numbers to the selected file path, or by appending the value of another field at the same row. This is particularly useful for naming output data objects according to input ones.
22.6.4 Executing the batch process

To execute the batch process once you have introduced all the necessary values, just click on OK. Progress of the global batch task will be shown in the progress bar in the lower part of the dialog.

22.7 Using processing algorithms from the console

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.

The code that you can execute from the Python console, even if it does not call any specific processing method, can be converted into a new algorithm that you can later call from the toolbox, the graphical modeler or any other component, just like you do with any other algorithm. In fact, some algorithms that you can find in the toolbox are simple scripts.

In this section, we will see how to use processing algorithms from the QGIS Python console, and also how to write algorithms using Python.

22.7.1 Calling algorithms from the Python console

The first thing you have to do is to import the processing functions with the following line:

```python
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:

```python
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 ----------- Tesselate
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 `native:buffer` as parameter (`qgis:buffer` is an alias for `native:buffer` and will also work), you get the following description:

```python
>>> 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.

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

(continues on next page)
- 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
- 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
Now you have everything you need to run any algorithm. As we have already mentioned, algorithms can be run using:

```python
run()```

Its syntax is as follows:

```python
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.

```python
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 `native: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.
• Fixed Table. Type the list of all table values separated by commas (,) and enclosed between quotes ("). Values start on the upper row and go from left to right. You can also use a 2-D array of values representing the table.

• CRS. Enter the EPSG code number of the desired CRS.

• Extent. You must use a string with xmin, xmax, ymin and ymax values separated by commas (,).

Boolean, file, string and numerical parameters do not need any additional explanations.

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:

```python
>>> myresult = 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'})
>>> 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:

```python
>>> 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:

```python
>>> 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'})
```
22.7.2 Creating scripts and running them from the toolbox

You can create your own algorithms by writing Python code. Processing scripts extend \texttt{QgsProcessingAlgorithm}, so you need to add some extra lines of code to implement mandatory functions. You can find \texttt{Create new script} (clean sheet) and \texttt{Create New Script from Template} (template that includes code for mandatory functions of \texttt{QgsProcessingAlgorithm}) under the \texttt{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 \texttt{scripts} folder (the default folder when you open the save file dialog) with a \texttt{.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.

```python
from qgis.core import (QgsProcessingAlgorithm,
                       QgsProcessingParameterNumber,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFeatureSink)

from qgis import processing

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]},
                                   (continues on next page)
```

(continues on next page)
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 define a dummy function for the `onFinish` parameter for `run`. This is the `no_post_process` function. 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:

- `QgsProcessingParameterAuthConfig`
- `QgsProcessingParameterBand`
- `QgsProcessingParameterBoolean`
- `QgsProcessingParameterColor`
- `QgsProcessingParameterCrs`
- `QgsProcessingParameterDistance`
- `QgsProcessingParameterEnum`
- `QgsProcessingParameterExpression`
- `QgsProcessingParameterExtent`
- `QgsProcessingParameterFeatureSink`
- `QgsProcessingParameterFeatureSource`
- `QgsProcessingParameterField`
- `QgsProcessingParameterFile`
- `QgsProcessingParameterFileDestination`
- `QgsProcessingParameterFolderDestination`
- `QgsProcessingParameterLayout`
- `QgsProcessingParameterLayoutItem`
- `QgsProcessingParameterMapLayer`
- `QgsProcessingParameterMatrix`
- `QgsProcessingParameterMeshLayer`
QgsProcessingParameterMultipleLayers
QgsProcessingParameterNumber
QgsProcessingParameterPoint
QgsProcessingParameterRange
QgsProcessingParameterRasterDestination
QgsProcessingParameterRasterLayer
QgsProcessingParameterScale
QgsProcessingParameterString
QgsProcessingParameterVectorDestination
QgsProcessingParameterVectorLayer

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:

```python
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.:

```python
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.

**Documenting your scripts**

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.

### 22.7.3 Pre- and post-execution script hooks

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.

### 22.8 Writing new Processing algorithms as Python scripts

There are two options for writing Processing algorithms using Python.

- *Extending `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.

### 22.8.1 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

```python
from qgis.PyQt.QtCore import QCoreApplication
from qgis.core import (QgsProcessing, 
QgsProcessingAlgorithm, 
QgsProcessingException, 
QgsProcessingOutputNumber, 
QgsProcessingParameterDistance, 
```
(continues on next page)
from qgis import processing

class ExampleProcessingAlgorithm(QgsProcessingAlgorithm):
    
    def tr(self, string):
        
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        
        return ExampleProcessingAlgorithm()

    def name(self):
        
        return 'bufferrasterextend'

    def displayName(self):
        
        return self.tr('Buffer and export to raster (extend)')

    def group(self):
        
        return self.tr('Example scripts')

    def groupId(self):
        
        return 'examplescripts'

    def shortHelpString(self):
        
        return self.tr('Example algorithm short description')

    def initAlgorithm(self, config=None):
        
        return

        self.addParameter(QgsProcessingParameterFeatureSource(continues on next page)
'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):
    
    # 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()

    # 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,
    'SEGSMENTS': 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'],
    '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.

  \[
  \text{INPUT and OUTPUT are recommended names for the main input and main output parameters, respectively.}
  \]

  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.

### 22.8.2 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

```python
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
(continues on next page)

22.8. Writing new Processing algorithms as Python scripts 639
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 graphical
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.

### 22.8.3 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 (algfactory.py contains the complete list of alg constants). Sorted on class name.

#### Input types

<table>
<thead>
<tr>
<th>Class</th>
<th>Alg constant</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>QgsProcessingParameterAuthConfig</td>
<td>alg.AUTH_CFG</td>
<td>Allows users to select from available authentication configurations or create new authentication configurations</td>
</tr>
<tr>
<td>QgsProcessingParameterBand</td>
<td>alg.BAND</td>
<td>A band of a raster layer</td>
</tr>
<tr>
<td>QgsProcessingParameterBoolean</td>
<td>alg.BOOL</td>
<td>A boolean value</td>
</tr>
<tr>
<td>QgsProcessingParameterColor</td>
<td>alg.COLOR</td>
<td>A color</td>
</tr>
<tr>
<td>QgsProcessingParameterCrs</td>
<td>alg.CRS</td>
<td>A Coordinate Reference System</td>
</tr>
<tr>
<td>QgsProcessingParameterDistance</td>
<td>alg.DISTANCE</td>
<td>A double numeric parameter for distance values</td>
</tr>
<tr>
<td>QgsProcessingParameterEnum</td>
<td>alg.ENUM</td>
<td>An enumeration, allowing for selection from a set of predefined values</td>
</tr>
<tr>
<td>QgsProcessingParameterExpression</td>
<td>alg.EXPRESSION</td>
<td>An expression</td>
</tr>
<tr>
<td>QgsProcessingParameterExtent</td>
<td>alg.EXTENT</td>
<td>A spatial extent defined by xmin, xmax, ymin, ymax</td>
</tr>
<tr>
<td>QgsProcessingParameterField</td>
<td>alg.FIELD</td>
<td>A field in the attribute table of a vector layer</td>
</tr>
<tr>
<td>QgsProcessingParameterFile</td>
<td>alg.FILE</td>
<td>A filename of an existing file</td>
</tr>
<tr>
<td>QgsProcessingParameterFileDestination</td>
<td>alg.FILE_DEST</td>
<td>A filename for a newly created output file</td>
</tr>
<tr>
<td>QgsProcessingParameterFolderDestination</td>
<td>alg.FOLDER_DEST</td>
<td>A folder</td>
</tr>
<tr>
<td>QgsProcessingParameterNumber</td>
<td>alg.INT</td>
<td>An integer</td>
</tr>
<tr>
<td>QgsProcessingParameterLayout</td>
<td>alg.LAYOUT</td>
<td>A layout</td>
</tr>
<tr>
<td>QgsProcessingParameterLayoutItem</td>
<td>alg.LAYOUT_ITEM</td>
<td>A layout item</td>
</tr>
<tr>
<td>QgsProcessingParameterMapLayer</td>
<td>alg.MAPLAYER</td>
<td>A map layer</td>
</tr>
<tr>
<td>QgsProcessingParameterMatrix</td>
<td>alg.MATRIX</td>
<td>A matrix</td>
</tr>
<tr>
<td>QgsProcessingParameterMeshLayer</td>
<td>alg.MESH_LAYER</td>
<td>A mesh layer</td>
</tr>
<tr>
<td>QgsProcessingParameterMultipleLayer</td>
<td>alg.MULTILAYER</td>
<td>A set of layers</td>
</tr>
<tr>
<td>QgsProcessingParameterNumber</td>
<td>alg.NUMBER</td>
<td>A numerical value</td>
</tr>
<tr>
<td>QgsProcessingParameterPoint</td>
<td>alg.POINT</td>
<td>A point</td>
</tr>
<tr>
<td>QgsProcessingParameterRange</td>
<td>alg.RANGE</td>
<td>A number range</td>
</tr>
<tr>
<td>QgsProcessingParameterRasterLayer</td>
<td>alg.RASTER_LAYER</td>
<td>A raster layer</td>
</tr>
</tbody>
</table>
22.8.4 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.
22.8.5 Communicating with the user

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, …)

22.8.6 Documenting your scripts

You can document your scripts by overloading the `helpString()` and `helpUrl()` methods of `QgsProcessingAlgorithm`.

22.8.7 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.

**Tip:** 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.

22.8.8 Best practices for writing script algorithms

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 [https://github.com/qgis/QGIS/blob/release-3_10/python/plugins/processing/algos/qgis](https://github.com/qgis/QGIS/blob/release-3_10/python/plugins/processing/algos/qgis).
22.9 Configuring external applications

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 graphical modeler, 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.

22.9.1 A note for Windows users

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.

22.9.2 A note on file formats

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.

22.9.3 A note on vector layer selections

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.
22.9.4 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.

About SAGA grid system limitations

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).

The definition of that common grid system is controlled by the user, and you will find several parameters in the SAGA group of the settings window to do so. There are two ways of setting the target grid system:

- Setting it manually. You define the extent by setting the values of the following parameters:
  - Resampling min X
  - Resampling max X
  - Resampling min Y
  - Resampling max Y
  - Resampling cellsize

  Notice that QGIS will resample input layers to that extent, even if they do not overlap with it.

- Setting it automatically from input layers. To select this option, just check the Use min covering grid system for resampling option. All the other settings will be ignored and the minimum extent that covers all the input layers will be used. The cell size of the target layer is the maximum of all cell sizes of the input layers.

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.

Limitations for multi-band layers

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).

Limitations in cell size

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.
Logging

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.

22.9.5 R scripts

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.

**Poznámka:** 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 http://cran.at.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)
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).

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 scripts 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.

```r
##Random points within layer extent=layer
##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.
22.9. Configuring external applications
Poznámka: 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).

```r
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:

```r
##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`):

```r
##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 elsewhere). Most of them are rather simple and will greatly help you understand how to create your own scripts.

**Poznámka:** 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.
22.9.6 R libraries

The R script `sp_test` tries to load the R packages `sp` and `raster`.

R libraries installed when running `sf_test`

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
- deltor
- dplyr
- dtraj
- digest
- dplyr
- e1071
- ellipsis
- evaluate
- fansi
- farver
- fastmap
- gdtools
- ggplot2
- glue
- goftest
- gridExtra
- gtable
- highr
- hms
- htmltools
- htmlwidgets
- httpuv
- htr
- 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
- processx
- promises
- ps
- purrr
- R6
- raster
- RColorBrewer
- Rcpp
- reshape2
- rex
- rgeos
- rlang
- rmrgrad
- RPostgres
- RPostgreSQL
- rprojroot
- RSQLite
- rstudioapi
- satellite
- scales
- sf
- shiny
- sourcescan
- sp
- spatstat
- spatstat.data
- spatstat.utils
- stars
- stringi
- stringr
- svglite
- sys
- systemfonts
- tensor
- testthat
- tidby
- tidyselect
- tinytex
- units
- utf8
- uuid
- vctrs
- viridis
- viridisLite
- webshot
- withr
- xfun
- XML
- xtable

22.9.7 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.
22.9.8 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.

22.9.9 OTB Applications

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

Poznámka: 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 (left panel)*
2. You can see OTB under „Providers“:
   1. Expand the OTB tab
   2. Tick the Activate option
   3. Set the OTB folder. This is the location of your OTB installation.
   4. Set the OTB application folder. This is the location of your OTB applications (PATH_TO_OTB_INSTALLATION>/lib/otb/applications)
   5. 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

- **Activate**: This is a checkbox to activate or deactivate the OTB provider. An invalid OTB setting will uncheck this when saved.
- **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 between QGIS and OTB versions**

OTB compiled with GDAL 3.X is not compatible with QGIS 3.10. This is the case for the binary packages of OTB 7.1 and above. Therefore QGIS 3.10 is only compatible with OTB official binary packages 6.6.1 and 7.0.0.

**Troubleshoot**

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 here.
Processing algorithms and their parameters (as presented in the user interface) are documented here.

23.1 QGIS algorithm provider

QGIS algorithm provider implements various analysis and geoprocessing operations using mostly only QGIS API. So almost all algorithms from this provider will work „out of the box“ without any additional configuration.

This provider incorporates some algorithms from plugins and also adds its own algorithms.

23.1.1 Cartography

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.

Viz také:

Create style database from project
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input databases</td>
<td>INPUT</td>
<td>[file] [list]</td>
<td>Files containing QGIS style items</td>
</tr>
</tbody>
</table>
| 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 — Symbols  
  • 1 — Color ramps  
  • 2 — Text formats  
  • 3 — Label settings |
| Output style database  | OUTPUT      | [file]              | Default: [Save to temporary file]  
Output .XML file combining the selected style items. One of:  
  • Save to a Temporary Layer (TEMPORARY_OUTPUT)  
  • Save to File...  
The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color ramp count</td>
<td>COLORRAMPS</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Label settings count</td>
<td>LABELSETTINGS</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Output style database</td>
<td>OUTPUT</td>
<td>[file]</td>
<td>Output .XML file combining the selected style items</td>
</tr>
<tr>
<td>Symbol count</td>
<td>SYMBOLS</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Text format count</td>
<td>TEXTFORMATS</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis:combinestyles`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer to apply a categorized style to</td>
</tr>
<tr>
<td>Categorize using expression</td>
<td>FIELD</td>
<td>[expression]</td>
<td>Field or expression to categorize the features</td>
</tr>
<tr>
<td>Style database (leave blank to use saved symbols)</td>
<td>STYLE</td>
<td>[file]</td>
<td>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.</td>
</tr>
<tr>
<td>Use case-sensitive match to symbol names</td>
<td>CASE_SENSITIVE</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Default: False</td>
<td>If True (checked), applies a case sensitive comparison between the categories and symbols names</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore non-alphanumeric characters while matching</td>
<td>TOLERANT</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Default: False</td>
<td>If True (checked), non-alphanumeric characters in the categories and symbols names will be ignored, allowing greater tolerance during the match.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-matching categories</td>
<td>NON_MATCHING_CATEGORIES</td>
<td>[table]</td>
<td>Output table for categories which do not match any symbol in the database. One of:</td>
</tr>
<tr>
<td>Optional</td>
<td>[Skip output]</td>
<td>• Skip output</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to File…</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
<td></td>
</tr>
<tr>
<td>Non-matching symbol names</td>
<td>NON_MATCHING_SYMBOLS</td>
<td>[table]</td>
<td>Output table for symbols from the provided style database which do not match any category. One of:</td>
</tr>
<tr>
<td>Optional</td>
<td>[Skip output]</td>
<td>• Skip output</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to File…</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
<td></td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-matching categories</td>
<td>NON_MATCHING_CATEGORIES</td>
<td>[table]</td>
<td>Lists categories which could not be matched to any symbol in the provided style database</td>
</tr>
<tr>
<td>Non-matching symbol names</td>
<td>NON_MATCHING_SYMBOLS</td>
<td>[table]</td>
<td>Lists symbols from the provided style database which could not match any category</td>
</tr>
<tr>
<td>Categorized layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The input vector layer with the categorized style applied. No new layer is output.</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis: categorizeusingstyle

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

Combine style databases

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input project (leave blank to use current)</td>
<td>INPUT</td>
<td>[file]</td>
<td>A QGIS project file to extract the style items from</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objects to extract</td>
<td>OBJECTS</td>
<td>[enumeration] [list]</td>
<td>Types of style items in the input project you would like to put in the new database. These can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Symbols</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Color ramps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Text formats</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Label settings</td>
</tr>
<tr>
<td>Output style database</td>
<td>OUTPUT</td>
<td>[file]</td>
<td>Specify the output .XML file for the selected style items. One of:</td>
</tr>
</tbody>
</table>
| | | | • Save to a Temporary Layer (TEMPORARY.Output)
| | | | • Save to File… |
| | | | The file encoding can also be changed here. |

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color ramp count</td>
<td>COLORRAMP</td>
<td>[number]</td>
<td>Number of color ramps</td>
</tr>
<tr>
<td>Label settings count</td>
<td>LABELSETTINGS</td>
<td>[number]</td>
<td>Number of label settings</td>
</tr>
<tr>
<td>Output style database</td>
<td>OUTPUT</td>
<td>[file]</td>
<td>Output .XML file for the selected style items</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol count</td>
<td>SYMBOLS</td>
<td>[number]</td>
<td>Number of symbols</td>
</tr>
<tr>
<td>Text format count</td>
<td>TEXTFORMATS</td>
<td>[number]</td>
<td>Number of text formats</td>
</tr>
</tbody>
</table>
**Python code**

**Algorithm ID:** qgis:stylefromproject

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print layout</td>
<td>LAYOUT</td>
<td>[enumeration]</td>
<td>A print layout in the current project</td>
</tr>
<tr>
<td>Map item</td>
<td>MAP</td>
<td>[enumeration]</td>
<td>The map item(s) whose information you want to extract. If none is provided then all the map items are processed.</td>
</tr>
<tr>
<td>Override CRS</td>
<td>CRS</td>
<td>[crs]</td>
<td>Select the CRS for the layer in which the information will be reported.</td>
</tr>
<tr>
<td>Extent</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer for the extent(s). One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map height</td>
<td>HEIGHT</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Extent</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Output polygon vector layer containing extents of all the input layout map item(s)</td>
</tr>
<tr>
<td>Map rotation</td>
<td>ROTATION</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Map scale</td>
<td>SCALE</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Map width</td>
<td>WIDTH</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:printlayoutmapextenttolayer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

![Topological colors example](image)

Obr. 23.1: Topological colors example
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>The input polygon layer</td>
</tr>
<tr>
<td>Minimum number of colors</td>
<td>MIN_COLORS</td>
<td>[number]</td>
<td>The minimum number of colors to assign. Minimum 1, maximum 1000.</td>
</tr>
<tr>
<td>Minimum distance between features</td>
<td>MIN_DISTANCE</td>
<td>[number]</td>
<td>Prevent nearby (but non-touching) features from being assigned equal colors. Minimum 0.0.</td>
</tr>
<tr>
<td>Balance color assignment</td>
<td>BALANCE</td>
<td>[enumeration]</td>
<td>Options are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — By feature count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Attempts to assign colors so that the count of features assigned to each individual color index is balanced.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — By assigned area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — By distance between colors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Colored</th>
<th>OUTPUT</th>
<th>[vector: polygon]</th>
<th>Specify the output layer. One of:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colored</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Polygon vector layer with an added color_id column</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:topologicalcoloring

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
23.1.2 Databáze

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

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer to import</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer to add to the database</td>
</tr>
<tr>
<td>Database (connection name)</td>
<td>DATABASE</td>
<td>[string]</td>
<td>Name of the database connection (not the database name). Existing connections will be shown in the combobox.</td>
</tr>
<tr>
<td>Schema (schema name) Optional</td>
<td>SCHEMA</td>
<td>[string] Default: .public'</td>
<td>Name of the schema to store the data. It can be a new one or already exist.</td>
</tr>
<tr>
<td>Table to import to (leave blank to use layer name) Optional</td>
<td>TABLENAME</td>
<td>[string] Default: '</td>
<td>Defines a table name for the imported vector file. If nothing is added, the layer name will be used.</td>
</tr>
<tr>
<td>Primary key field Optional</td>
<td>PRIMARY_KEY</td>
<td>[tablefield: any]</td>
<td>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.</td>
</tr>
<tr>
<td>Geometry column</td>
<td>GEOMETRY_COLUMN</td>
<td>[string] Default: .geom'</td>
<td>Defines the name of the geometry column in the new PostGIS table. Geometry information for the features is stored in this column.</td>
</tr>
<tr>
<td>Encoding Optional</td>
<td>ENCODING</td>
<td>[string] UTF-8'</td>
<td>Defines the encoding of the output layer</td>
</tr>
<tr>
<td>Overwrite</td>
<td>OVERWRITE</td>
<td>[boolean] Default: True</td>
<td>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 (&quot;relation already exists&quot;).</td>
</tr>
<tr>
<td>Create spatial index</td>
<td>CREATEINDEX</td>
<td>[boolean] Default: True</td>
<td>Specifies whether to create a spatial index or not</td>
</tr>
<tr>
<td>Convert field names to lowercase</td>
<td>LOWERCASE_NAMES</td>
<td>[boolean] Default: True</td>
<td>Converts the field names of the input vector layer to lowercase</td>
</tr>
<tr>
<td>Drop length constraint on character fields</td>
<td>DROP_STRING_LENGTH</td>
<td>[boolean] Default: False</td>
<td>Should length constraints on character fields be dropped or not</td>
</tr>
<tr>
<td>Create single-part geometries instead of multi-part</td>
<td>FORCE_SINGLEPART</td>
<td>[boolean] Default: False</td>
<td>Should the features of the output layer be single-part instead of multi-part. By default the existing geometries information are preserved.</td>
</tr>
</tbody>
</table>
Outputs

The algorithm has no output.

Python code

**Algorithm ID**: `qgis:importintopostgis`

```python
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 *Using processing algorithms from the console* 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 *SpatiaLite Layers*).

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer to import</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer to add to the database</td>
</tr>
<tr>
<td>File database</td>
<td>DATABASE</td>
<td>[vector: any]</td>
<td>The SQLite/SpatiaLite database file to connect to</td>
</tr>
<tr>
<td>Table to import to (leave blank to use layer name)</td>
<td>TABLENAME</td>
<td>[string]</td>
<td>Defines the table name for the imported vector file. If nothing is specified, the layer name will be used.</td>
</tr>
<tr>
<td>Optional</td>
<td>PRIMARY_KEY</td>
<td>[tablefield: any]</td>
<td>Use a field in the input vector layer as the primary key</td>
</tr>
<tr>
<td>Geom. column</td>
<td>GEOMETRY_COLUMN</td>
<td>[string]</td>
<td>Default: `geom' Defines the name of the geometry column in the new SpatiaLite table. Geometry information for the features is stored in this column.</td>
</tr>
<tr>
<td>Optional</td>
<td>ENCODING</td>
<td>[string]</td>
<td>Default: `UTF-8' Defines the encoding of the output layer</td>
</tr>
<tr>
<td>Overwrite</td>
<td>OVERWRITE</td>
<td>[boolean]</td>
<td>If the specified table exists, setting this option to <em>True</em> 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 <em>False</em> and the table exists, the algorithm will throw an exception (&quot;table already exists&quot;).</td>
</tr>
<tr>
<td>Create spatial index</td>
<td>CREATEINDEX</td>
<td>[boolean]</td>
<td>Specifies whether to create a spatial index or not</td>
</tr>
<tr>
<td>Convert field names to lowercase</td>
<td>LOWERCASE_NAMES</td>
<td>[boolean]</td>
<td>Convert the field names of the input vector layer to lowercase</td>
</tr>
<tr>
<td>Drop length constraint on character fields</td>
<td>DROP_STRING_LENGTH</td>
<td>[boolean]</td>
<td>Should length constraints on character fields be dropped or not</td>
</tr>
</tbody>
</table>
### Outputs

The algorithm has no output.

### Python code

**Algorithm ID:** qgis:importintospatialite

```python
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 *Using processing algorithms from the console* 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

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layers</strong></td>
<td>LAYERS</td>
<td>[vector: any] [list]</td>
<td>The (vector) layers to import into the GeoPackage. Raster layers are not supported. If a raster layer is added, a QgsProcessingException will be thrown.</td>
</tr>
<tr>
<td><strong>Overwrite existing GeoPackage</strong></td>
<td>OVERWRITE</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Save layer styles into GeoPackage</strong></td>
<td>SAVE_STYLES</td>
<td>[boolean]</td>
<td>Save the layer styles</td>
</tr>
<tr>
<td><strong>Destination GeoPackage</strong></td>
<td>OUTPUT</td>
<td>[file]</td>
<td>If not specified the GeoPackage will be saved in the temporary folder.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layers within new package</td>
<td>OUTPUT_LAYERS</td>
<td>[string] [list]</td>
<td>The list of layers added to the GeoPackage.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:package

```python
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 [Using processing algorithms from the console](#) 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:

   ```sql
   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.

   ```sql
   -- 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);
   ```

Viz také:

*PostgreSQL execute SQL, Execute SQL, SpatiaLite execute SQL*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database (connection name)</td>
<td>DATABASE</td>
<td>[string]</td>
<td>The database connection (not the database name). Existing connections will be shown in the combobox.</td>
</tr>
<tr>
<td>SQL query</td>
<td>SQL</td>
<td>[string]</td>
<td>Defines the SQL query, for example 'UPDATE my_table SET field=10'.</td>
</tr>
<tr>
<td>Unique ID field name</td>
<td>ID_FIELD</td>
<td>[string]</td>
<td>Sets the primary key field (a column in the result table)</td>
</tr>
<tr>
<td>Geometry field name</td>
<td>GEOMETRY_FIELD</td>
<td>[string]</td>
<td>Name of the geometry column (a column in the result table)</td>
</tr>
</tbody>
</table>

23.1. QGIS algorithm provider
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL layer</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>The resulting vector layer to be loaded into QGIS.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:postgisexecuteandloadsql`

```python
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 *Using processing algorithms from the console* 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:

   ```sql
   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.

   ```sql
   -- 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);
   ```

**Viz také:**

*PostgreSQL execute and load SQL*, *Execute SQL*, *SpatiaLite execute SQL*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database (connection name)</td>
<td>DATABASE</td>
<td>[string]</td>
<td>The database connection (not the database name). Existing connections will be shown in the combobox.</td>
</tr>
<tr>
<td>SQL query</td>
<td>SQL</td>
<td>[string]</td>
<td>Defines the SQL query, for example 'UPDATE my_table SET field=10'.</td>
</tr>
</tbody>
</table>
### Outputs

No output is created. The SQL query is executed in place.

### Python code

**Algorithm ID:** qgis:postgisexecutesql

```python
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 *Using processing algorithms from the console* 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 connected to QGIS. The algorithm won’t create a new layer: it is designed to run queries on the layer itself.

**Viz také:**

*PostgreSQL execute SQL, Execute SQL*

For some SQL query examples see *PostGIS SQL Query Examples.*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Database</td>
<td>DATABASE</td>
<td>[vector]</td>
<td>The SQLite/SpatiaLite database file to connect to</td>
</tr>
<tr>
<td>SQL query</td>
<td>SQL</td>
<td>[string]</td>
<td>Defines the SQL query, for example 'UPDATE my_table SET field=10'.</td>
</tr>
</tbody>
</table>

### Outputs

No output is created. The SQL query is executed in place.

### Python code

**Algorithm ID:** qgis:spatialiteexecutesql

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
23.1.3 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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>URL</td>
<td>[string]</td>
<td>The URL of the file to download.</td>
</tr>
<tr>
<td>File destination</td>
<td>OUTPUT</td>
<td>[string]</td>
<td>Specification of the file destination. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>File destination</td>
<td>OUTPUT</td>
<td>[string]</td>
<td>The location of the downloaded file</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:filedownloader

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

23.1.4 Graphics

Bar plot

Creates a bar plot from a category and a layer field.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Category field name</td>
<td>NAME_FIELD</td>
<td>[tablefield: any]</td>
<td>Categorical field to use for grouping the bars (X axis)</td>
</tr>
<tr>
<td>Value field</td>
<td>VALUE_FIELD</td>
<td>[tablefield: any]</td>
<td>Value to use for the plot (Y axis).</td>
</tr>
<tr>
<td>Bar plot</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bar plot</strong></td>
<td>OUTPUT</td>
<td>[html]</td>
<td>HTML file with the plot. Available in the Processing Result Viewer.</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:barplot

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Box plot**

Creates a box plot from a category field and a numerical layer field.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td><strong>Category name field</strong></td>
<td>NAME_FIELD</td>
<td>[tablefield: any]</td>
<td>Categorical field to use for grouping the boxes (X axis)</td>
</tr>
<tr>
<td><strong>Value field</strong></td>
<td>VALUE_FIELD</td>
<td>[tablefield: any]</td>
<td>Value to use for the plot (Y axis).</td>
</tr>
<tr>
<td><strong>Additional statistic lines</strong></td>
<td>MSD</td>
<td>[enumeration]</td>
<td>Additional statistics information to add to the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Show Mean</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Show Standard Deviation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Don’t show mean and standard deviation</td>
</tr>
<tr>
<td><strong>Box plot</strong></td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default:  [Save to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box plot</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>HTML file with the plot. Available in the Processing Result Viewer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:boxplot

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Mean and standard deviation plot**

Creates a box plot with mean and standard deviation values.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input table</td>
<td>INPUT</td>
<td>[vector:any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Category name field</td>
<td>NAME_FIELD</td>
<td>[tablefield:any]</td>
<td>Categorical field to use for grouping the boxes (X axis)</td>
</tr>
<tr>
<td>Value field</td>
<td>VALUE_FIELD</td>
<td>[tablefield:any]</td>
<td>Value to use for the plot (Y axis).</td>
</tr>
<tr>
<td>Plot</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>HTML file with the plot. Available in the Processing Result Viewer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:meanandstandarddeviationplot

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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).

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Category name field</td>
<td>NAME_FIELD</td>
<td>[tablefield: any]</td>
<td>Categorical field to use for grouping the features (X axis)</td>
</tr>
<tr>
<td>Value field</td>
<td>VALUE_FIELD</td>
<td>[tablefield: any]</td>
<td>Value to use for the plot (Y axis).</td>
</tr>
<tr>
<td>Polar plot</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

**Python code**

**Algorithm ID:** qgis:polarplot

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### Raster layer histogram

Generates a histogram with the values of a raster layer.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>Raster band to use for the histogram</td>
</tr>
<tr>
<td>number of bins</td>
<td>BINS</td>
<td>[number]</td>
<td>The number of bins to use in the histogram (X axis). Minimum 2.</td>
</tr>
<tr>
<td>Histogram</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

**Python code**

**Algorithm ID:** qgis:rasterlayerhistogram

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Attribute</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Value to use for the plot (Y axis).</td>
</tr>
<tr>
<td>number of bins</td>
<td>BINS</td>
<td>[number]</td>
<td>The number of bins to use in the histogram (X axis). Minimum 2.</td>
</tr>
<tr>
<td>Histogram</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
Python code

Algorithm ID: qgis:vectorlayerhistogram

```python
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The \textit{algorithm id} is displayed when you hover over the algorithm in the Processing Toolbox. The \textit{parameter dictionary} provides the parameter \textit{NAMEs} and values. See \textit{Using processing algorithms from the console} for details on how to run processing algorithms from the Python console.

Vector layer scatterplot

Creates a simple $X$ - $Y$ scatter plot for a vector layer.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>X attribute</td>
<td>XFIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for the X axis</td>
</tr>
<tr>
<td>Y attribute</td>
<td>YFIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for the Y axis</td>
</tr>
<tr>
<td>Scatterplot</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

Python code

Algorithm ID: qgis:vectorlayerscatterplot

```python
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

The \textit{algorithm id} is displayed when you hover over the algorithm in the Processing Toolbox. The \textit{parameter dictionary} provides the parameter \textit{NAMEs} and values. See \textit{Using processing algorithms from the console} for details on how to run processing algorithms from the Python console.
Vector layer scatterplot 3D

Creates a 3D scatter plot for a vector layer.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>X attribute</td>
<td>XFIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for the X axis</td>
</tr>
<tr>
<td>Y attribute</td>
<td>YFIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for the Y axis</td>
</tr>
<tr>
<td>Z attribute</td>
<td>ZFIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for the Z axis</td>
</tr>
<tr>
<td>Histogram</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for the plot. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

Python code

**Algorithm ID:** qgis:scatter3dplot

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.1.5 Interpolation

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.
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Point vector layer to use for the heatmap</td>
</tr>
<tr>
<td>Radius</td>
<td>RADIUS</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Output raster size</td>
<td>PIXEL_SIZE</td>
<td>[number]</td>
<td>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).</td>
</tr>
<tr>
<td>Radius from field</td>
<td>RADIUS_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Sets the search radius for each feature from an attribute field in the input layer.</td>
</tr>
<tr>
<td>Weight from field</td>
<td>WEIGHT_FIELD</td>
<td>[tablefield: numeric]</td>
<td>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.</td>
</tr>
<tr>
<td>Kernel shape</td>
<td>KERNEL</td>
<td>[enumeration]</td>
<td>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):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Quartic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Triangular</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Uniform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Triweight</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Epanechnikov</td>
</tr>
</tbody>
</table>
Decay ratio (Triangular kernels only)
Optional

<table>
<thead>
<tr>
<th>Decay ratio (Triangular kernels only)</th>
<th>DECAY</th>
<th>[number]</th>
<th>Default: 0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Can be used with Triangular kernels</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to further control how heat from a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature decreases with distance from</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the feature.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• A value of 0 (=minimum) indicates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>that the heat will be concentrated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>in the center of the given radius</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and completely extinguished at the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>edge.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• A value of 0.5 indicates that</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pixels at the edge of the radius</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>will be given half the heat as</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pixels at the center of the search</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>radius.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• A value of 1 means the heat is</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spread evenly over the whole search</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>radius circle. (This is equivalent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to the ‘Uniform’ kernel.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• A value greater than 1 indicates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>that the heat is higher towards the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>edge of the search radius than at</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the center.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output value scaling

<table>
<thead>
<tr>
<th>Output value scaling</th>
<th>OUTPUT_VALUE</th>
<th>[enumeration]</th>
<th>Default: Raw</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allow to change the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>values of the output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>heatmap raster. One</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 0 — Raw</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 1 — Scaled</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Heatmap

<table>
<thead>
<tr>
<th>Heatmap</th>
<th>OUTPUT</th>
<th>[raster]</th>
<th>Default: [Save to temporary file]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the output raster layer with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kernel density values. One of:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Save to a Temporary File</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Save to File…</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The file encoding can also be changed here.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heatmap</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Raster layer with kernel density values</td>
</tr>
</tbody>
</table>

Example: Creating a Heatmap

For the following example, we will use the airports vector point layer from the QGIS sample dataset (see Downloading sample data). Another excellent QGIS tutorial on making heatmaps can be found at http://qgistutorials.com.

In Figure Heatmap data processing, the airports of Alaska are shown.

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 Figure Heatmap created processing).

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.
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 Figure_HeatmapStyled_processing.

**Python code**

Algorithm ID: `qgis:heatmapkerneldensityestimation`

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Obr. 23.3: The Heatmap Dialog
Obr. 23.4: The heatmap after loading looks like a grey surface

Obr. 23.5: Styled heatmap of airports of Alaska
IDW Interpolation

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer(s)</td>
<td>INTERPOLATION_DATA</td>
<td>[string]</td>
<td>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: • Vector layer [vector: any] • 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 '::</td>
</tr>
<tr>
<td>Distance coefficient P</td>
<td>DISTANCE_COEFFICIENT</td>
<td>[number]</td>
<td>Sets the distance coefficient for the interpolation. Minimum: 0.0, maximum: 100.0.</td>
</tr>
<tr>
<td>Extent (xmin, xmax, ymin, ymax)</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Extent of the output raster layer. You have to declare the output extent by either choosing it from the map canvas, selecting it from another layer or type it manually.</td>
</tr>
</tbody>
</table>
### Output raster size

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output raster size</td>
<td>PIXEL_SIZE</td>
<td>[number]</td>
<td>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).</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Interpolated | OUTPUT       | [raster]  | Raster layer of interpolated values. One of:  
  - Save to a Temporary File  
  - Save to File...  
  The file encoding can also be changed here. |

### Python code

**Algorithm ID:** qgis:idwinterpolation

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### TIN Interpolation

Generates a Triangulated Irregular Network (TIN) interpolation of a point vector layer.

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.
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Input layer(s)   | INTERPOLATION_DATA            | [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:  
  • Vector layer [vector: any]  
  • 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                      | [enumeration] | Set the interpolation method to be used. One of:  
  • Linear  
  • Clough-Toucher (cubic) |
| Extent (xmin, xmax, ymin, ymax) | EXTENT                      | [extent] | Extent of the output raster layer. You have to declare the output extent by either choosing it from the map canvas, selecting it from another layer or type it manually. |
| Output raster size | PIXEL_SIZE                   | [number] | 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] | The output TIN interpolation as a raster layer. One of:  
  • Save to a Temporary File  
  • Save to File...  
  The file encoding can also be changed here. |

Pokračujte na další stránce
### Tabulka 23.10 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Triangulation | TRIANGULATION | [vector: line] | The output TIN as a vector layer. One of:  
|             |           | Default: [Skip output] | • Skip Output  
|             |           |                       | • Create Temporary Layer (TEMPORARY_OUTPUT)  
|             |           |                       | • Save to File…  
|             |           |                       | • Save to Geopackage…  
|             |           |                       | • Save to PostGIS Table… |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The output TIN interpolation as a raster layer</td>
</tr>
<tr>
<td>Triangulation</td>
<td>TRIANGULATION</td>
<td>[vector: line]</td>
<td>The output TIN as a vector layer.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:tininterpolation

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.1.6 Layer tools

#### Extract layer extent

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.

**Default menu:** Vector ➤ Research Tools

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>INPUT</td>
<td>[layer]</td>
<td>Input layer</td>
</tr>
</tbody>
</table>
| Extent      | OUTPUT    | [vector: polygon]     | Specify the polygon vector layer for the output extent. One of:  
|             |           | Default: [Create temporary layer] | • Create Temporary Layer (TEMPORARY_OUTPUT)  
|             |           |                       | • Save to File…  
|             |           |                       | • Save to Geopackage…  
|             |           |                       | • Save to PostGIS Table…  
|             |           |                       | The file encoding can also be changed here. |
Obr. 23.6: In red the bounding box of the source layer

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Output (polygon) vector layer with the extent (minimum bounding box)</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis: polygonfromlayerextent`

```python
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 [*Using processing algorithms from the console*](https://qgis.org/en/latest/manuals/3.10/qgis_manual/using_processing.html) for details on how to run processing algorithms from the Python console.
23.1.7 Modeler tools

These tools are only available in the Graphical Modeler. They are not available in the Processing Toolbox.

Load layer into project

Loads a layer to the current project.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>INPUT</td>
<td>[layer]</td>
<td>Layer to load in the legend</td>
</tr>
<tr>
<td>Loaded layer name</td>
<td>NAME</td>
<td>[string]</td>
<td>Name of the loaded layer</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The (renamed) loaded layer</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** qgis:loadlayer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Rename layer**

Renames a layer.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>INPUT</td>
<td>[layer]</td>
<td>Layer to rename</td>
</tr>
<tr>
<td>New name</td>
<td>NAME</td>
<td>[string]</td>
<td>The new name of the layer</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The (renamed) output layer</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:renamelayer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**String concatenation**

Concatenates two strings into a single one in the Processing Modeler.
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 1</td>
<td>INPUT_1</td>
<td>[string]</td>
<td>First string</td>
</tr>
<tr>
<td>Input 2</td>
<td>INPUT_2</td>
<td>[string]</td>
<td>Second string</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation</td>
<td>CONCATENATION</td>
<td>[string]</td>
<td>The concatenated string</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:stringconcatenation

```python
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 [Using processing algorithms from the console](#) for details on how to run processing algorithms from the Python console.

#### 23.1.8 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).

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector layer representing network</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Line vector layer representing the network to be covered</td>
</tr>
<tr>
<td>Vector layer with start points</td>
<td>START_POINTS</td>
<td>[vector: point]</td>
<td>Point vector layer whose features are used as start points to generate the service areas</td>
</tr>
<tr>
<td>Path type to calculate</td>
<td>STRATEGY</td>
<td>[enumeration]</td>
<td>The type of path to calculate. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Shortest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Fastest</td>
</tr>
<tr>
<td>Travel cost (distance for „Shortest“; time for „Fastest“)</td>
<td>TRAVEL_COST</td>
<td>[number]</td>
<td>The value is estimated as a distance (in the network layer units) when looking for the Shortest path and as time (in seconds) for the Fastest path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Default: 0</td>
</tr>
</tbody>
</table>

23.1. QGIS algorithm provider 687
### Direction field
Optional

**DIRECTION_FIELD**

**[tablefield: string]**

Default: 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
Optional

**VALUE_FORWARD**

**[string]**

Default: ‘’ (empty string)

Value set in the direction field to identify edges with a forward direction.

### Value for backward direction
Optional

**VALUE_BACKWARD**

**[string]**

Default: ‘’ (empty string)

Value set in the direction field to identify edges with a backward direction.

### Value for both directions
Optional

**VALUE_BOTH**

**[string]**

Default: ‘’ (empty string)

Value set in the direction field to identify bidirectional edges.

### Default direction
Optional

**DEFAULT_DIRECTION**

**[enumeration]**

Default: 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 — Forward direction
- 1 — Backward direction
- 2 — Both directions

### Speed field
Optional

**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)
Optional

**DEFAULT_SPEED**

**[number]**

Default: 50.0

Value to use to calculate the travel time if no speed field is provided for an edge.

### Topology tolerance
Optional

**TOLERANCE**

**[number]**

Default: 0.0

Two lines with nodes closer than the specified tolerance are considered connected.

### Include upper/lower bound points
Optional

**INCLUDE_BOUNDS**

**[boolean]**

Default: 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.

Pokračujte na ďalší stránce
Tabulka 23.14 – pokračujte na předchozí stránce

| Service area (lines) | OUTPUT_LINES | [vector: line] | Specify the output line layer for the service area. One of:
| | | Default: [Create temporary layer] |  • Skip output
| | | |  • Create Temporary Layer (TEMPORARY_OUTPUT)
| | | |  • Save to File…
| | | |  • Save to Geopackage…
| | | |  • Save to PostGIS Table…
| | | |  The file encoding can also be changed here.
| Service area (boundary nodes) | OUTPUT | [vector: point] | Specify the output point layer for the service area boundary nodes. One of:
| | | Default: [Skip output] |  • Skip output
| | | |  • Create Temporary Layer (TEMPORARY_OUTPUT)
| | | |  • Save to File…
| | | |  • Save to Geopackage…
| | | |  • Save to PostGIS Table…
| | | |  The file encoding can also be changed here.

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service area (boundary nodes)</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output point layer with the service area boundary nodes.</td>
</tr>
<tr>
<td>Service area (lines)</td>
<td>OUTPUT_LINES</td>
<td>[vector: line]</td>
<td>Line layer representing the parts of the network that can be serviced by the start points, for the given cost.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:serviceareafromlayer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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).
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector layer representing the network</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Line vector layer representing the network to be covered</td>
</tr>
<tr>
<td>Start point (x, y)</td>
<td>START_POINT</td>
<td>[coordinates]</td>
<td>Coordinate of the point to calculate the service area around.</td>
</tr>
</tbody>
</table>
| Path type to calculate | STRATEGY | [enumeration] | The type of path to calculate. One of:
- 0 — Shortest
- 1 — Fastest |
| Travel cost | TRAVEL_COST | [number] | The value is estimated as a distance (in the network layer units) when looking for the Shortest path and as time (in seconds) for the Fastest path. |
| Advanced parameters | GUI only | | Group of advanced network analysis parameters - see below. |
| Service area (lines) | OUTPUT_LINES | [vector: line] | Specify the output line layer for the service area. One of:
- Skip output
- Create Temporary Layer (TEMPORARY_OUTPUT)
- Save to File…
- Save to Geopackage…
- Save to PostGIS Table…
| Service area (boundary nodes) | OUTPUT | [vector: point] | Specify the output point layer for the service area boundary nodes. One of:
- Skip output
- Create Temporary Layer (TEMPORARY_OUTPUT)
- Save to File…
- Save to Geopackage…
- Save to PostGIS Table…

## Advanced parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction field</td>
<td>DIRECTION_FIELD</td>
<td>[tablefield: string]</td>
<td>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.</td>
</tr>
</tbody>
</table>
Tabulka 23.16 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value for forward direction</td>
<td>VALUE_FORWARD</td>
<td>[string]</td>
<td>Value set in the direction field to identify edges with a forward direction</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ; (empty string)</td>
<td></td>
</tr>
<tr>
<td>Value for backward direction</td>
<td>VALUE_BACKWARD</td>
<td>[string]</td>
<td>Value set in the direction field to identify edges with a backward direction</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ; (empty string)</td>
<td></td>
</tr>
<tr>
<td>Value for both directions</td>
<td>VALUE_BOTH</td>
<td>[string]</td>
<td>Value set in the direction field to identify bidirectional edges</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ; (empty string)</td>
<td></td>
</tr>
<tr>
<td>Default direction</td>
<td>DEFAULT_DIRECT</td>
<td>[enumeration]</td>
<td>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:</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: 2</td>
<td>• 0 — Forward direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Backward direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Both directions</td>
</tr>
<tr>
<td>Speed field</td>
<td>SPEED_FIELD</td>
<td>[tablefield: string]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default speed (km/h)</td>
<td>DEFAULT_SPEED</td>
<td>[number]</td>
<td>Value to use to calculate the travel time if no speed field is provided for an edge</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: 50.0</td>
<td></td>
</tr>
<tr>
<td>Topology tolerance</td>
<td>TOLERANCE</td>
<td>[number]</td>
<td>Two lines with nodes closer than the specified tolerance are considered connected</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Include upper/lower bound points</td>
<td>INCLUDE_BOUNDS</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service area (boundary nodes)</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output point layer with the service area boundary nodes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service area (lines)</td>
<td>OUTPUT_LINES</td>
<td>[vector: line]</td>
<td>Line layer representing the parts of the network that can be serviced by the start point, for the given cost.</td>
</tr>
</tbody>
</table>

23.1. QGIS algorithm provider
Python code

**Algorithm ID:** `qgis:serviceareafrompoint`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector layer representing network</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Line vector layer representing the network to be covered</td>
</tr>
</tbody>
</table>
| Path type to calculate         | STRATEGY                  | [enumeration]      | The type of path to calculate. One of:  
                                  |                           | • 0 — Shortest  
                                  |                           | • 1 — Fastest  
| Vector layer with start points | START_POINTS              | [vector: point]    | 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               |
| Advanced parameters            | GUI only                  |                    | The *Advanced parameters* group:                                      |
| Direction field                | DIRECTION_FIELD           | [tablefield: string] | The field used to specify directions for the network edges.  
                                  |                           | Default: 0.0              | 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    | VALUE_FORWARD             | [string]           | Value set in the direction field to identify edges with a forward direction |
| Value for backward direction   | VALUE_BACKWARD            | [string]           | Value set in the direction field to identify edges with a backward direction |
| Value for both directions      | VALUE_BOTH                | [string]           | Value set in the direction field to identify bidirectional edges      |

Pokračujte na další stránce
Tabulka 23.17 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default direction</td>
<td>DEFAULT_DIRECTION</td>
<td>[enumeration]</td>
<td>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:</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>• 0 — Forward direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Backward direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Both directions</td>
</tr>
<tr>
<td>Speed field</td>
<td>SPEED_FIELD</td>
<td>[tablefield: string]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Two lines with nodes closer than the specified tolerance are considered connected</td>
</tr>
<tr>
<td>Default speed (km/h)</td>
<td>DEFAULT_SPEED</td>
<td>[number]</td>
<td>Value to use to calculate the travel time if no speed field is provided for an edge</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topology tolerance</td>
<td>TOLERANCE</td>
<td>[number]</td>
<td>Two lines with nodes closer than the specified tolerance are considered connected</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>End of the Advanced parameters group</td>
</tr>
<tr>
<td>Shortest path</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line layer for the shortest paths. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest path</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Line layer of the shortest or fastest path from each of the start points to the end point</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:shortestpathlayertopoint

```python
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 [Using processing algorithms from the console](#) for details on how to run processing algorithms from the Python console.
**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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector layer representing network</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Line vector layer representing the network to be covered</td>
</tr>
<tr>
<td>Path type to calculate</td>
<td>STRATEGY</td>
<td>[enumeration]</td>
<td>The type of path to calculate. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>- 0 — Shortest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1 — Fastest</td>
</tr>
<tr>
<td>Start point (x, y)</td>
<td>START_POINT</td>
<td>[coordinates]</td>
<td>Point feature representing the start point of the routes</td>
</tr>
<tr>
<td>Vector layer with end points</td>
<td>END_POINTS</td>
<td>[vector: point]</td>
<td>Point vector layer whose features are used as end points of the routes</td>
</tr>
<tr>
<td>Direction field</td>
<td>DIRECTION_FIELD</td>
<td>[tablefield: string]</td>
<td>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.</td>
</tr>
<tr>
<td>Value for forward direction</td>
<td>VALUE_FORWARD</td>
<td>[string]</td>
<td>Value set in the direction field to identify edges with a forward direction</td>
</tr>
<tr>
<td>Optional Advanced</td>
<td></td>
<td>Default: ‘‘ (empty string)</td>
<td></td>
</tr>
<tr>
<td>Value for backward direction</td>
<td>VALUE_BACKWARD</td>
<td>[string]</td>
<td>Value set in the direction field to identify edges with a backward direction</td>
</tr>
<tr>
<td>Optional Advanced</td>
<td></td>
<td>Default: ‘‘ (empty string)</td>
<td></td>
</tr>
<tr>
<td>Value for both directions</td>
<td>VALUE_BOTH</td>
<td>[string]</td>
<td>Value set in the direction field to identify bidirectional edges</td>
</tr>
<tr>
<td>Optional Advanced</td>
<td></td>
<td>Default: ‘‘ (empty string)</td>
<td></td>
</tr>
<tr>
<td>Default direction</td>
<td>DEFAULT_DIRECTION</td>
<td>[enumeration]</td>
<td>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:</td>
</tr>
<tr>
<td>Optional Advanced</td>
<td></td>
<td>Default: 2</td>
<td>- 0 — Forward direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1 — Backward direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 2 — Both directions</td>
</tr>
<tr>
<td>Speed field</td>
<td>SPEED_FIELD</td>
<td>[tablefield: string]</td>
<td>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.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Tabulka 23.18 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default speed (km/h)</strong></td>
<td>DEFAULT_SPEED</td>
<td>[number]</td>
<td>Value to use to calculate the travel time if no speed field is provided for an edge</td>
</tr>
<tr>
<td>Optional Advanced</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Topology tolerance</strong></td>
<td>TOLERANCE</td>
<td>[number]</td>
<td>Two lines with nodes closer than the specified tolerance are considered connected</td>
</tr>
<tr>
<td>Optional Advanced</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Shortest path</strong></td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line layer for the shortest paths. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shortest path</strong></td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Line layer of the shortest or fastest path from each of the start points to the end point</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:shortestpathpointtolayer`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### Shortest path (point to point)

Computes the optimal (shortest or fastest) route between a given start point and a given end point.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Advanced</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vector layer representing network</strong></td>
<td>INPUT</td>
<td></td>
<td>[vector: line]</td>
<td>Line vector layer representing the network to be covered</td>
</tr>
<tr>
<td><strong>Path type to calculate</strong></td>
<td>STRATEGY</td>
<td></td>
<td>[enumeration]</td>
<td>The type of path to calculate. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0 — Shortest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1 — Fastest</td>
</tr>
<tr>
<td><strong>Start point (x, y)</strong></td>
<td>START_POINT</td>
<td></td>
<td>[coordinates]</td>
<td>Point feature representing the start point of the routes</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Advanced</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------</td>
<td>----------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>End point (x, y)</td>
<td>END_POINT</td>
<td>Advanced</td>
<td>[coordinates]</td>
<td>Point feature representing the end point of the routes</td>
</tr>
<tr>
<td><strong>Direction field</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td>DIRECTION_FIELD</td>
<td>[tablefield: string]</td>
<td>Default: 0.0</td>
<td>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.</td>
</tr>
<tr>
<td>Value for forward direction</td>
<td>VALUE_FORWARD</td>
<td>Optional</td>
<td>[string]</td>
<td>Value set in the direction field to identify edges with a forward direction</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: ‘’ (empty string)</td>
<td></td>
</tr>
<tr>
<td>Value for backward direction</td>
<td>VALUE_BACKWARD</td>
<td>Optional</td>
<td>[string]</td>
<td>Value set in the direction field to identify edges with a backward direction</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: ‘’ (empty string)</td>
<td></td>
</tr>
<tr>
<td>Value for both directions</td>
<td>VALUE_BOTH</td>
<td>Optional</td>
<td>[string]</td>
<td>Value set in the direction field to identify bidirectional edges</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: ‘’ (empty string)</td>
<td></td>
</tr>
<tr>
<td>Default direction</td>
<td>DEFAULT_DIRECTION</td>
<td>Optional</td>
<td>[enumeration]</td>
<td>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 — Forward direction • 1 — Backward direction • 2 — Both directions</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: 2</td>
<td></td>
</tr>
<tr>
<td>Speed field</td>
<td>SPEED_FIELD</td>
<td>Optional</td>
<td>[tablefield: string]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default speed (km/h)</td>
<td>DEFAULT_SPEED</td>
<td>Optional</td>
<td>[number]</td>
<td>Value to use to calculate the travel time if no speed field is provided for an edge</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: 50.0</td>
<td></td>
</tr>
<tr>
<td>Topology tolerance</td>
<td>TOLERANCE</td>
<td>Optional</td>
<td>[number]</td>
<td>Two lines with nodes closer than the specified tolerance are considered connected</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
</tbody>
</table>
Tabulka 23.19 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Advanced</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest path</td>
<td>OUTPUT</td>
<td></td>
<td>[vector: line]</td>
<td>Specify the output line layer for the shortest paths. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest path</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Line layer of the shortest or fastest path from each of the start point to the end point</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:shortestpathpointtopoint

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.1.9 Rastrová analýza

**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.

Viz také:

**Raster boolean OR**
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layers</td>
<td>INPUT</td>
<td>[raster] [list]</td>
<td>List of input raster layers</td>
</tr>
<tr>
<td>Reference layer</td>
<td>REF_LAYER</td>
<td>[raster]</td>
<td>The reference layer to create the output layer from (extent, CRS, pixel dimensions)</td>
</tr>
<tr>
<td>Treat nodata values as false</td>
<td>NODATA_AS_FALSE</td>
<td>[boolean]</td>
<td>Default: False. Treat nodata values in the input files as 0 when performing the operation</td>
</tr>
<tr>
<td>Output no data value</td>
<td>NO_DATA</td>
<td>[number]</td>
<td>Default: -9999.0. Value to use for nodata in the output layer</td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Default: 5. Output raster data type. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>The extent of the output raster layer</td>
</tr>
<tr>
<td>CRS authority identifier</td>
<td>CRS_AUTHID</td>
<td>[crs]</td>
<td>The coordinate reference system of the output raster layer</td>
</tr>
<tr>
<td>Width in pixels</td>
<td>WIDTH_IN_PIXELS</td>
<td>[integer]</td>
<td>The width in pixels of the output raster layer</td>
</tr>
<tr>
<td>Height in pixels</td>
<td>HEIGHT_IN_PIXELS</td>
<td>[integer]</td>
<td>The height in pixels of the output raster layer</td>
</tr>
<tr>
<td>Total pixel count</td>
<td>TOTAL_PIXEL_COUNT</td>
<td>[integer]</td>
<td>The count of pixels in the output raster layer</td>
</tr>
<tr>
<td>NODATA pixel count</td>
<td>NODATA_PIXEL_COUNT</td>
<td>[integer]</td>
<td>The count of nodata pixels in the output raster layer</td>
</tr>
<tr>
<td>True pixel count</td>
<td>TRUE_PIXEL_COUNT</td>
<td>[integer]</td>
<td>The count of True pixels (value = 1) in the output raster layer</td>
</tr>
<tr>
<td>False pixel count</td>
<td>FALSE_PIXEL_COUNT</td>
<td>[integer]</td>
<td>The count of False pixels (value = 0) in the output raster layer</td>
</tr>
<tr>
<td>Output layer</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer containing the result</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:rasterbooleanand

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

**Viz také:**

*Raster boolean AND*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layers</td>
<td>INPUT</td>
<td>[raster] [list]</td>
<td>List of input raster layers</td>
</tr>
<tr>
<td>Reference layer</td>
<td>REF_LAYER</td>
<td>[raster]</td>
<td>The reference layer to create the output layer from (extent, CRS, pixel dimensions)</td>
</tr>
<tr>
<td>Treat nodata values as false</td>
<td>NODATA_AS_FALSE</td>
<td>[boolean] Default: False</td>
<td>Treat nodata values in the input files as 0 when performing the operation</td>
</tr>
<tr>
<td>Output no data value</td>
<td>NO_DATA</td>
<td>[number] Default: -9999.0</td>
<td>Value to use for nodata in the output layer</td>
</tr>
</tbody>
</table>
| Output data type       | DATA_TYPE           | [enumeration] Default: 5 | Output raster data type. Options:  
  • 0 — Byte  
  • 1 — Int16  
  • 2 — UInt16  
  • 3 — UInt32  
  • 4 — Int32  
  • 5 — Float32  
  • 6 — Float64  
  • 7 — CInt16  
  • 8 — CInt32  
  • 9 — CFloat32  
  • 10 — CFloat64 |
| Output layer           | OUTPUT              | [raster]      | Output raster layer                                                 |
## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>The extent of the output raster layer</td>
</tr>
<tr>
<td>CRS authority identifier</td>
<td>CRS_AUTHID</td>
<td>[crs]</td>
<td>The coordinate reference system of the output raster layer</td>
</tr>
<tr>
<td>Width in pixels</td>
<td>WIDTH_IN_PIXELS</td>
<td>[integer]</td>
<td>The width in pixels of the output raster layer</td>
</tr>
<tr>
<td>Height in pixels</td>
<td>HEIGHT_IN_PIXELS</td>
<td>[Integer]</td>
<td>The height in pixels of the output raster layer</td>
</tr>
<tr>
<td>Total pixel count</td>
<td>TOTAL_PIXEL_COUNT</td>
<td>[integer]</td>
<td>The count of pixels in the output raster layer</td>
</tr>
<tr>
<td>NODATA pixel count</td>
<td>NODATA_PIXEL_COUNT</td>
<td>[integer]</td>
<td>The count of nodata pixels in the output raster layer</td>
</tr>
<tr>
<td>True pixel count</td>
<td>TRUE_PIXEL_COUNT</td>
<td>[Integer]</td>
<td>The count of True pixels (value = 1) in the output raster layer</td>
</tr>
<tr>
<td>False pixel count</td>
<td>FALSE_PIXEL_COUNT</td>
<td>[Integer]</td>
<td>The count of False pixels (value = 0) in the output raster layer</td>
</tr>
<tr>
<td>Output layer</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer containing the result</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:rasterbooleanor

```python
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 [Using processing algorithms from the console](#) for details on how to run processing algorithms from the Python console.

### Raster calculator

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.

**Poznámka:** When using the calculator in *The batch processing interface* or from the *QGIS Python console* the files to use have to be specified. The corresponding layers are referred using the base name of the file (without the full path). For instance, if using a layer at `path/to/my/rasterfile.tif`, the first band of that layer will be referred as `rasterfile.tif@1`.

### Parameters
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layers</td>
<td>GUI only</td>
<td></td>
<td>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: <code>layer_name@band_number</code>. For instance, the first band from a layer named DEM will be referred as DEM@1.</td>
</tr>
<tr>
<td>Operators</td>
<td>GUI only</td>
<td></td>
<td>Contains some calculator like buttons that can be used to fill the expression box.</td>
</tr>
<tr>
<td>Expression</td>
<td>EXPRESSION</td>
<td>[string]</td>
<td>Expression that will be used to calculate the output raster layer. You can use the operator buttons provided to type directly the expression in this box.</td>
</tr>
<tr>
<td>Predefined expressions</td>
<td>GUI only</td>
<td></td>
<td>You can use the predefined NDVI expression or you can define new expressions for calculations. The Add… button loads a defined expression (and lets you set the parameters). The Save… button lets you define a new expression.</td>
</tr>
<tr>
<td>Reference layer(s) (used for automated extent, cell size, and CRS)</td>
<td>LAYERS</td>
<td>[raster] [list]</td>
<td>Layer(s) that will be used to fetch extent, cell size and CRS. By choosing the layer in this box you avoid filling in all the other parameters by hand. Raster layers are referred by their name and the number of the band: <code>layer_name@band_number</code>. For instance, the first band from a layer named DEM will be referred as DEM@1.</td>
</tr>
<tr>
<td>Cell size (use 0 or empty to set it automatically)</td>
<td>CELLSIZE</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Output extent (xmin, xmax, ymin, ymax)</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>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.</td>
</tr>
<tr>
<td>Output CRS</td>
<td>CRS</td>
<td>[crs]</td>
<td>CRS of the output raster layer. If the output CRS is not specified, the CRS of the first reference layer will be used.</td>
</tr>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specification of the output raster. One of: • Save to a Temporary File • Save to File… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster file with the calculated values.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis:rastercalculator`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose the band you want to get statistics for.</td>
</tr>
</tbody>
</table>
| Output      | OUTPUT_HTML_FILE | [html] | Specification of the output file:  
  • Skip Output  
  • Save to a Temporary File  
  • Save to File…  
  The file encoding can also be changed here. |

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum value</td>
<td>MAX</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Mean value</td>
<td>MEAN</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td>MIN</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>
### Tabulka 23.25 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>OUTPUT_HTML_FILE</td>
<td>[html]</td>
<td>The output file contains the following information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Analyzed file: path of the raster layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Minimum value: minimum value of the raster</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Maximum value: maximum value of the raster</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Range: difference between the maximum and minimum values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Sum: total sum of the values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mean value: mean of the values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Standard deviation: standard deviation of the values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Sum of the squares: sum of the squared differences of each observation from the overall mean</td>
</tr>
<tr>
<td>Range</td>
<td>RANGE</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Standard deviation</td>
<td>STD_DEV</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>SUM</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Sum of the squares</td>
<td>SUM_OF_SQUARES</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:rasterlayerstatistics

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

### Raster layer unique values report

Returns the count and area of each unique value in a given raster layer.

#### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose the band you want to get statistics for.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Default: The first band of the input layer</td>
</tr>
<tr>
<td>Unique values report</td>
<td>OUTPUT_HTML_FILE</td>
<td>[file]</td>
<td>Specification of the output file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

---

23.1. QGIS algorithm provider 703
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique values table</td>
<td>OUTPUT_TABLE</td>
<td>table</td>
<td>Specification of the table for unique values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to GeoPackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRS authority identifier</td>
<td>CRS_AUTHID</td>
<td>crs</td>
<td></td>
</tr>
<tr>
<td>Extent</td>
<td>EXTENT</td>
<td>extent</td>
<td></td>
</tr>
<tr>
<td>Height in pixels</td>
<td>HEIGHT_IN_PIXELS</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>NODATA pixel count</td>
<td>NODATA_PIXEL_COUNT</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>Total pixel count</td>
<td>TOTAL_PIXEL_COUNT</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>Unique values report</td>
<td>OUTPUT_HTML_FILE</td>
<td>html</td>
<td>The output HTML file contains the following information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Analyzed file: the path of the raster layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Extent: xmin, ymin, xmax, ymax coordinates of the extent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Projection: projection of the layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Width in pixels: number of columns and pixel width size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Height in pixels: number of rows and pixel width size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Total pixel count: count of all the pixels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• NODATA pixel count: count of pixels with NODATA value</td>
</tr>
<tr>
<td>Unique values table</td>
<td>OUTPUT_TABLE</td>
<td>table</td>
<td>A table with three columns:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• value: pixel value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• count: count of pixels with this value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• (m^2): total area in square meters of pixels with this value</td>
</tr>
<tr>
<td>Width in pixels</td>
<td>WIDTH_IN_PIXELS</td>
<td>number</td>
<td></td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:rasterlayeruniquevaluesreport`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
**Raster layer zonal statistics**

Calculates statistics for a raster layer’s values, categorized by zones defined in another raster layer.

**Viz také:**

*Zonal statistics*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband choose the band for which you want to calculate the statistics.</td>
</tr>
<tr>
<td>Zones layer</td>
<td>ZONES</td>
<td>[raster]</td>
<td>Raster layer defining zones. Zones are given by contiguous pixels having the same pixel value.</td>
</tr>
<tr>
<td>Zones band number</td>
<td>ZONES_BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose the band that defines the zones</td>
</tr>
</tbody>
</table>
| Reference layer | REF_LAYER   | [enumeration] | Raster layer used to calculate the centroids that will be used as reference when determining the zones in the output layer. One of:
  • 0 — Input layer
  • 1 — Zones layer |
| Statistics     | OUTPUT_TABLE| [table]       | Table with the calculated statistics                                 |

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRS authority identifier</td>
<td>CRS_AUTHID</td>
<td>[crs]</td>
<td></td>
</tr>
<tr>
<td>Extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td></td>
</tr>
<tr>
<td>Height in pixels</td>
<td>HEIGHT_IN_PIXELS</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>NODATA pixel count</td>
<td>NODATA_PIXEL_COUNT</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>

*Pokračujte na další stránce*
### Statistics

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics</td>
<td>OUTPUT_TABLE</td>
<td>The output layer contains the following information for each zone:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Area: the area in square raster units in the zone;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sum: the total sum of the pixel values in the zone;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Count: the number of pixels in the zone;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Min: the minimum pixel value in the zone;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Max: the maximum pixel value in the zone;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mean: the mean of the pixel values in the zone;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total pixel count</th>
<th>TOTAL_PIXEL_COUNT</th>
<th>[number]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width in pixels</td>
<td>WIDTH_IN_PIXELS</td>
<td>[number]</td>
</tr>
</tbody>
</table>

#### Python code

**Algorithm ID:** qgis:rasterlayerzonalstats

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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).

#### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT layer</td>
<td>INPUT</td>
<td>Input raster, representing a surface</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>If the raster is multiband, choose the band that shall define the surface.</td>
</tr>
<tr>
<td>Base level</td>
<td>LEVEL</td>
<td>Define a base or reference value. This base is used in the volume calculation according to the <code>Method</code> parameter (see below).</td>
</tr>
</tbody>
</table>
Tabulka 23.30 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Method | METHOD | [enumeration] | Define the method for the volume calculation given by the difference between the raster pixel value and the Base level. Options:  

- 0 — Count Only Above Base Level: only pixels above the base level will add to the volume.  
- 1 — Count Only Below Base Level: only pixels below the base level will add to the volume.  
- 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. |

| Surface volume report | OUTPUT_HTML_FILE | [html] | Specification of the output HTML report. One of:  

- Skip output  
- Save to Temporary File  
- Save to File…  

The file encoding can also be changed here. |

| Surface volume table | OUTPUT_TABLE | [table] | Specification of the output table. One of:  

- Skip output  
- Create Temporary Layer (TEMPORARY_OUTPUT)  
- Save to File…  
- Save to Geopackage…  
- Save to PostGIS Table…  

The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>VOLUME</td>
<td>[number]</td>
<td>The calculated volume</td>
</tr>
<tr>
<td>Area</td>
<td>AREA</td>
<td>[number]</td>
<td>The area in square map units</td>
</tr>
<tr>
<td>Pixel_count</td>
<td>PIXEL_COUNT</td>
<td>[number]</td>
<td>The total number of pixels that have been analyzed</td>
</tr>
<tr>
<td>Surface volume report</td>
<td>OUTPUT_HTML_FILE</td>
<td>[html]</td>
<td>The output report (containing volume, area and pixel count) in HTML format</td>
</tr>
<tr>
<td>Surface volume table</td>
<td>OUTPUT_TABLE</td>
<td>[table]</td>
<td>The output table (containing volume, area and pixel count)</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** qgis:rastersurfacevolume

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Reclassify by layer

Reclassifies a raster band by assigning new class values based on the ranges specified in a vector table.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Raster layer to reclassify</td>
</tr>
<tr>
<td>Band number</td>
<td>RASTER_BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose the band you want to reclassify.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: The first band of the raster layer</td>
<td></td>
</tr>
<tr>
<td>Layer containing class breaks</td>
<td>INPUT_TABLE</td>
<td>[vector: any]</td>
<td>Vector layer containing the values to use for classification.</td>
</tr>
<tr>
<td>Minimum class value field</td>
<td>MIN_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field with the minimum value of the range for the class.</td>
</tr>
<tr>
<td>Maximum class value field</td>
<td>MAX_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field with the maximum value of the range for the class.</td>
</tr>
<tr>
<td>Output value field</td>
<td>VALUE_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field with the value that will be assigned to the pixels that fall in the class (between the corresponding min and max values).</td>
</tr>
<tr>
<td>Output no data value</td>
<td>NO_DATA</td>
<td>[number]</td>
<td>Value to apply to no data values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: -9999.0</td>
<td></td>
</tr>
<tr>
<td>Range boundaries</td>
<td>RANGE_BOUNDARIES</td>
<td>[enumeration]</td>
<td>Defines comparison rules for the classification. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — min &lt; value &lt;= max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — min &lt;= value &lt; max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — min &lt;= value &lt;= max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — min &lt; value &lt; max</td>
</tr>
<tr>
<td>Use no data when no range matches value</td>
<td>NODATA_FOR_MISS</td>
<td>[boolean]</td>
<td>Values that do not belong to a class will result in the no data value. If False, the original value is kept.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Tabulka 23.31 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output data type</strong></td>
<td></td>
<td><strong>DATA_TYPE</strong></td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[enumeration]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
<tr>
<td><strong>Reclassified raster</strong></td>
<td></td>
<td><strong>OUTPUT</strong></td>
<td>Specification of the output raster. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[raster]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reclassified raster</strong></td>
<td></td>
<td><strong>OUTPUT</strong></td>
<td>Output raster layer with reclassified band values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[raster]</td>
<td></td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** **qgis:reclassifybylayer**

```python
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 _Using processing algorithms from the console_ for details on how to run processing algorithms from the Python console.

### Reclassify by table

Reclassifies a raster band by assigning new class values based on the ranges specified in a fixed table.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raster layer</strong></td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Raster layer to reclassify</td>
</tr>
<tr>
<td><strong>Band number</strong></td>
<td>RASTER_BAND</td>
<td>[raster band]</td>
<td>Raster band for which you want to recompute values.</td>
</tr>
<tr>
<td></td>
<td>Default: 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reclassification table</strong></td>
<td>TABLE</td>
<td>[table]</td>
<td>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.</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>------------------------</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Output no data value</strong></td>
<td>NO_DATA</td>
<td>[number]</td>
<td>Value to apply to no data values.</td>
</tr>
<tr>
<td><strong>Range boundaries</strong></td>
<td>RANGE_BOUNDARIES</td>
<td>[enumeration]</td>
<td>Defines comparison rules for the classification. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — min &lt; value &lt;= max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — min &lt;= value &lt; max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — min &lt;= value &lt;= max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — min &lt; value &lt; max</td>
</tr>
<tr>
<td><strong>Use no data when no range matches value</strong></td>
<td>NODATA_FOR_MISSING</td>
<td>[boolean]</td>
<td>Applies the no data value to band values that do not fall in any class. If False, the original value is kept.</td>
</tr>
<tr>
<td><strong>Output data type</strong></td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the format of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 5</td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
<tr>
<td><strong>Reclassified raster</strong></td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specification of the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: {Save to temporary file}</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reclassified raster</strong></td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The output raster layer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: {Save to temporary file}</td>
<td></td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:reclassifybytable

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Point Layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Point vector layer to use for sampling</td>
</tr>
<tr>
<td>Raster Layer to sample</td>
<td>RASTERCOPY</td>
<td>[raster]</td>
<td>Raster layer to sample at the given point locations.</td>
</tr>
<tr>
<td>Output column prefix</td>
<td>COLUMN_PREFIX</td>
<td>[string]</td>
<td>Prefix for the names of the added columns.</td>
</tr>
<tr>
<td>Sampled Points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output layer containing the sampled values. One of:</td>
</tr>
<tr>
<td>(Optional)</td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to GeoPackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampled Points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output layer containing the sampled values.</td>
</tr>
<tr>
<td>(Optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:rastersampling

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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).

![Raster layer histogram example](image)

Obr. 23.7: Raster layer histogram example

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer.</td>
</tr>
<tr>
<td>Band number</td>
<td>RASTER_BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose a band.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: The first band of the input layer</td>
<td></td>
</tr>
<tr>
<td>Vector layer containing zones</td>
<td>INPUTVECTOR</td>
<td>[vector: polygon]</td>
<td>Vector polygon layer that defines the zones.</td>
</tr>
<tr>
<td>Output column prefix</td>
<td>COLUMN_PREFIX</td>
<td>[string]</td>
<td>Prefix for the output columns names.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: <code>HISTO_</code></td>
<td></td>
</tr>
<tr>
<td>Output zones</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector polygon layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to GeoPackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Optional)</td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td></td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:zonalhistogram

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Zonal statistics

Calculates statistics of a raster layer for each feature of an overlapping polygon vector layer.

**Varování:** No new output file will be created. The algorithm adds new columns to the source vector layer.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer.</td>
</tr>
<tr>
<td>Raster band</td>
<td>RASTER_BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose a band for the statistics.</td>
</tr>
<tr>
<td></td>
<td>Default: The first band of the input layer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vector layer contain-</td>
<td>INPUT_VECTOR</td>
<td>[vector: polygon]</td>
<td>Vector polygon layer that defines the zones.</td>
</tr>
<tr>
<td>ing zones</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output column prefix</td>
<td>COLUMN_PREFIX</td>
<td>[string]</td>
<td>Prefix for the output columns names.</td>
</tr>
<tr>
<td></td>
<td>Default: '_'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Tabulka 23.33** – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statistics to calculate</strong></td>
<td><strong>STATISTICS</strong></td>
<td>[enumeration] [list]</td>
<td>List of statistical operator for the output. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [0,1,2]</td>
<td>• 0 — Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Sum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Mean</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Median</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — St. dev.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Minimum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Maximum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — Minority</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — Majority</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — Variety</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — Variance</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vector layer containing zones</strong></td>
<td><strong>INPUT VECTOR</strong></td>
<td>[vector: polygon]</td>
<td>The input zone vector layer with added statistics.</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:zonalstatistics

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**23.1.10 Raster terrain analysis**

**Aspect**

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.

![Aspect values](image)

Obr. 23.8: Aspect values
The following picture shows the aspect layer reclassified with a color ramp:

Obr. 23.9: Aspect layer reclassified

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Digital Terrain Model raster layer</td>
</tr>
<tr>
<td>Z factor</td>
<td>Z_FACTOR</td>
<td>[number]</td>
<td>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).</td>
</tr>
<tr>
<td>Aspect</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output aspect raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspect</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The output aspect raster layer</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:aspect

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

![Obr. 23.10: Azimuth and vertical angle](image)

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.

Particularly interesting is to give the hillshade layer a transparency value and overlap it with the elevation raster:

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation layer</td>
<td>INPUT</td>
<td>raster</td>
<td>Digital Terrain Model raster layer</td>
</tr>
<tr>
<td>Z factor</td>
<td>Z_FACTOR</td>
<td>[number] Default: 1.0</td>
<td>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).</td>
</tr>
<tr>
<td>Azimuth (horizontal angle)</td>
<td>AZIMUTH</td>
<td>[number] Default: 300.0</td>
<td>Set the horizontal angle (in degrees) of the sun (clockwise direction). Range: 0 to 360. 0 is north.</td>
</tr>
<tr>
<td>Vertical angle</td>
<td>V_ANGLE</td>
<td>[number] Default: 40.0</td>
<td>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).</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hillshade</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output hillshade raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Python code

Algorithm ID: qgis:hillshade

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

## 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.

## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEM to analyze</td>
<td>INPUT_DEM</td>
<td>[raster]</td>
<td>Digital Terrain Model raster layer to use for calculating altitudes</td>
</tr>
<tr>
<td>Boundary layer</td>
<td>BOUNDARY_LAYER</td>
<td>[vector: polygon]</td>
<td>Polygon vector layer with boundaries of areas used to calculate hypsometric curves</td>
</tr>
<tr>
<td>Step</td>
<td>STEP</td>
<td>[number]</td>
<td>Default: 100.0</td>
</tr>
<tr>
<td>Use % of area instead of absolute value</td>
<td>USE_PERCENTAGE</td>
<td>[boolean]</td>
<td>Write area percentage to “Area” field of the CSV file instead of the absolute area</td>
</tr>
<tr>
<td>Hypsometric curves</td>
<td>OUTPUT_DIRECTORY</td>
<td>[folder]</td>
<td>Specify the output folder for the hypsometric curves. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Obr. 23.11: Hillshade layer with azimuth 300 and vertical angle 45

Obr. 23.12: Overlapping the hillshade with the elevation layer
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypsometric curves</td>
<td>OUTPUT_DIRECTORY</td>
<td>[folder]</td>
<td>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.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:hypsometriccurves

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

![Relief Layer Image](image)

Obr. 23.13: Relief layer

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Digital Terrain Model raster layer</td>
</tr>
<tr>
<td>Z factor</td>
<td>Z_FACTOR</td>
<td>[number]</td>
<td>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).</td>
</tr>
<tr>
<td>Generate relief classes</td>
<td>AUTO_COLORS</td>
<td>[boolean]</td>
<td>If you check this option the algorithm will create all the relief color classes automatically</td>
</tr>
<tr>
<td>automatically</td>
<td>Default: False</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
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.

Obr. 23.14: 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.

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relief colors</td>
<td>COLORS</td>
<td>[table widget]</td>
<td>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.</td>
</tr>
<tr>
<td>Relief</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output relief raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Frequency distribution</td>
<td>FREQUENCY_DISTRIBUTION</td>
<td>[table]</td>
<td>Specify the CSV table for the output frequency distribution. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relief</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The output relief raster layer</td>
</tr>
<tr>
<td>Frequency distribution</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>The output frequency distribution</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** qgis:relief

```python
code
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

![Ruggedness index example](image)

**Obr. 23.15:** Ruggedness layer from low (red) to high values (green)
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Digital Terrain Model raster layer</td>
</tr>
<tr>
<td>Z factor</td>
<td>Z_FACTOR</td>
<td>[number]</td>
<td>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). Default: 1.0</td>
</tr>
</tbody>
</table>
| Ruggedness    | OUTPUT | [raster] | Specify the output ruggedness raster layer. One of:  
- Save to a Temporary Layer (TEMPORARY_OUTPUT)  
- Save to File...  
The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruggedness</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The output ruggedness raster layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:ruggednessindex

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Slope

Calculates the slope from an input raster layer. The slope is the angle of inclination of the terrain and is expressed in degrees.

In the following picture you can see to the left the DTM layer with the elevation of the terrain while to the right the calculated slope:
Obr. 23.16: Flat areas in red, steep areas in blue

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Digital Terrain Model raster layer</td>
</tr>
<tr>
<td>Z factor</td>
<td>Z_FACTOR</td>
<td>[number]</td>
<td>Default: 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).</td>
</tr>
</tbody>
</table>
| Slope        | OUTPUT | [raster] | Specify the output slope raster layer. One of:  
  - Save to a Temporary Layer (TEMPORARY_OUTPUT)  
  - Save to File…  
  The file encoding can also be changed here.     |
## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slope</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The output slope raster layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID**: qgis:slope

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.1.11 Raster tools

#### 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.
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum extent to render (xmin, xmax, ymin, ymax)</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Specify the extent of the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Use CanvasExtent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Select Extent on Canvas</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Use LayerExtent… It will internally be extended to a multiple of the tile size.</td>
</tr>
<tr>
<td>Tile size</td>
<td>TILE_SIZE</td>
<td>[number]</td>
<td>Size of the tile of the output raster layer. Minimum value: 64.</td>
</tr>
<tr>
<td>Map units per pixel</td>
<td>MAP_UNITS_PER_PIXEL</td>
<td>[number]</td>
<td>Pixel size (in map units). Minimum value: 0.0</td>
</tr>
<tr>
<td>Make background transparent</td>
<td>MAKE_BACKGROUND</td>
<td>[boolean]</td>
<td>Allows exporting the map with a transparent background. Outputs an RGBA (instead of RGB) image if set to True.</td>
</tr>
<tr>
<td>Map theme to render</td>
<td>MAP_THEME</td>
<td>[enumeration]</td>
<td>Use an existing map theme for the rendering.</td>
</tr>
<tr>
<td>Single layer to render</td>
<td>LAYER</td>
<td>[enumeration]</td>
<td>Choose a single layer for the rendering</td>
</tr>
<tr>
<td>Output layer</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specification of the output raster. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer</td>
</tr>
</tbody>
</table>

## Python code

Algorithm ID: `qgis:rasterize`

```python
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 "Using processing algorithms from the console" for details on how to run processing algorithms from the Python console.
Create constant raster layer

Generates a raster layer where all pixels have the same value.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desired extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Specify the extent of the output raster layer. One of:</td>
</tr>
<tr>
<td>(xmin, xmax, ymin, ymax)</td>
<td></td>
<td></td>
<td>• Use Canvas Extent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Select Extent on Canvas</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Use Layer Extent...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It will internally be extended to a multiple of the tile size.</td>
</tr>
<tr>
<td>Target CRS</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td>Project CRS for the output raster layer</td>
</tr>
<tr>
<td>Pixel size</td>
<td>PIXEL_SIZE</td>
<td>[number]</td>
<td>Pixel size (X=Y) in map units. Minimum value: 0.01</td>
</tr>
<tr>
<td>Constant value</td>
<td>NUMBER</td>
<td>[number]</td>
<td>Constant pixel value for the output raster layer.</td>
</tr>
<tr>
<td>Constant</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specification of the output raster. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Raster covering the desired extent with the specified pixel size and value.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:createconstantrasterlayer

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Generate XYZ tiles (Directory)

Generates raster “XYZ” tiles using the current QGIS project as individual images to a directory structure.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent (xmin, xmax, ymin, ymax)</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Specify the extent of the tiles. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Use Canvas Extent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Select Extent on Canvas</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Use Layer Extent...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It will internally be extended to a multiple of the tile size.</td>
</tr>
<tr>
<td>Minimum zoom</td>
<td>ZOOM_MIN</td>
<td>[number]</td>
<td>Default: 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum 0, maximum 25.</td>
</tr>
<tr>
<td>Maximum zoom</td>
<td>ZOOM_MAX</td>
<td>[number]</td>
<td>Default: 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum 0, maximum 25.</td>
</tr>
<tr>
<td>DPI</td>
<td>DPI</td>
<td>[number]</td>
<td>Default: 96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum 48, maximum 600.</td>
</tr>
<tr>
<td>Background color Optional</td>
<td>BACKGROUND_COLOR</td>
<td>[color]</td>
<td>Default: QColor(0, 0, 0, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Choose the background color for the tiles</td>
</tr>
<tr>
<td>Tile format</td>
<td>TILE_FORMAT</td>
<td>[enumeration]</td>
<td>Default: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 0 — PNG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1 — JPG</td>
</tr>
<tr>
<td>Quality (JPG only) Optional</td>
<td>QUALITY</td>
<td>[number]</td>
<td>Default: 75</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum 1, maximum 100.</td>
</tr>
<tr>
<td>Metatile size Optional</td>
<td>METATILESIZE</td>
<td>[number]</td>
<td>Default: 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Tile width Optional</td>
<td>TILE_WIDTH</td>
<td>[number]</td>
<td>Default: 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum 1, maximum 4096.</td>
</tr>
<tr>
<td>Tile height Optional</td>
<td>TILE_HEIGHT</td>
<td>[number]</td>
<td>Default: 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum 1, maximum 4096.</td>
</tr>
<tr>
<td>Use inverted tile Y axis (TMS conventions) Optional</td>
<td>TMS_CONVENTION</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specification of the output raster. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary Directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Directory...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Output directory</td>
<td>OUTPUT_DIRECTORY</td>
<td>[folder]</td>
<td>Default: [Save to temporary folder]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specification of the output HTML file. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td>Output html (Leaflet)</td>
<td>OUTPUT_HTML</td>
<td>[html]</td>
<td>Default: [Save to temporary file]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specification of the output HTML file. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output directory</td>
<td>OUTPUT_DIRECTORY</td>
<td>[folder]</td>
<td>Output directory (for the tiles)</td>
</tr>
<tr>
<td>Output html (Leaf-let)</td>
<td>OUTPUT_HTML</td>
<td>[html]</td>
<td>The output HTML (Leaflet) file</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:tilesxyzdirectory

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Generate XYZ tiles (MBTiles)

Generates raster “XYZ” tiles using the current QGIS project as a single file in the “MBTiles” format.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent (xmin, xmax, ymin, ymax)</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Specify the extent of the tiles. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Use Canvas Extent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Select Extent on Canvas</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Use Layer Extent…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It will internally be extended to a multiple</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of the tile size.</td>
</tr>
<tr>
<td>Minimum zoom</td>
<td>ZOOM_MIN</td>
<td>[number]</td>
<td>Minimum 0, maximum 25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Default: 12</td>
</tr>
<tr>
<td>Maximum zoom</td>
<td>ZOOM_MAX</td>
<td>[number]</td>
<td>Minimum 0, maximum 25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Default: 12</td>
</tr>
<tr>
<td>DPI</td>
<td>DPI</td>
<td>[number]</td>
<td>Minimum 48, maximum 600.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Default: 96</td>
</tr>
<tr>
<td>Background color</td>
<td>BACKGROUND_COLOR</td>
<td>[color]</td>
<td>Choose the background color for the tiles</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: QColor(0, 0, 0, 0)</td>
</tr>
<tr>
<td>Tile format</td>
<td>TILE_FORMAT</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — PNG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — JPG</td>
</tr>
<tr>
<td>Quality (JPG only)</td>
<td>QUALITY</td>
<td>[number]</td>
<td>Minimum 1, maximum 100.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Default: 75</td>
</tr>
</tbody>
</table>
**Tabulka 23.39 – pokračujte na předchozí stránce**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metatile size</strong></td>
<td>METATILESIZE</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Output file (for MBTiles)</strong></td>
<td>OUTPUT_FILE</td>
<td>[file]</td>
<td>Specification of the output file. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

**Python code**

**Algorithm ID:** qgis:tilesxyzmbtiles

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Set style for raster layer**

Sets the style of a raster layer. The style must be defined as a QML file.

No new output are created: the QML style is assigned to the raster layer chosen.

**Viz také:**

*Set style for vector layer*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raster layer</strong></td>
<td>INPUT</td>
<td>[raster]</td>
<td>The raster layer</td>
</tr>
<tr>
<td><strong>Style file</strong></td>
<td>STYLE</td>
<td>[file]</td>
<td>Path to the QML style file.</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>The raster layer with the chosen style</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis: setstylefor raster layer`

```python
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 NAMESPACEs and values. See [Using processing algorithms from the console](#) for details on how to run processing algorithms from the Python console.

#### 23.1.12 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

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input vector</td>
<td>INPUT_LAYER</td>
<td>[vector: any]</td>
<td>Vector layer to calculate the statistics on</td>
</tr>
<tr>
<td>Field to calculate statistics on</td>
<td>FIELD_NAME</td>
<td>[tablefield: any]</td>
<td>Any supported table field to calculate the statistics</td>
</tr>
<tr>
<td>Statistics</td>
<td>OUTPUT_HTML_FILE</td>
<td>[html]</td>
<td>HTML file for the calculated statistics</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics</td>
<td>OUTPUT_HTML_FILE</td>
<td>[html]</td>
<td>HTML file with the calculated statistics</td>
</tr>
<tr>
<td>Count</td>
<td>COUNT</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Number of unique values</td>
<td>UNIQUE</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Number of empty (null) values</td>
<td>EMPTY</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Number of non-empty values</td>
<td>FILLED</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td>MIN</td>
<td>[same as input]</td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td>MAX</td>
<td>[same as input]</td>
<td></td>
</tr>
<tr>
<td>Minimum length</td>
<td>MIN_LENGTH</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Maximum length</td>
<td>MAX_LENGTH</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Mean length</td>
<td>MEAN_LENGTH</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: `qgis:basicstatisticsforfields`

```python
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 `Using processing algorithms from the console` for details on how to run processing algorithms from the Python console.

### 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).

#### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Line layer to calculate the climb for. Must have Z values</td>
</tr>
<tr>
<td>Climb layer</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output (line) layer</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Climb layer</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Line layer containing new attributes with the results from climb calculations.</td>
</tr>
<tr>
<td>Total climb</td>
<td>TOTALCLIMB</td>
<td>[number]</td>
<td>The sum of the climb for all the line geometries in the input layer</td>
</tr>
<tr>
<td>Total descent</td>
<td>TOTALDESCENT</td>
<td>[number]</td>
<td>The sum of the descent for all the line geometries in the input layer</td>
</tr>
<tr>
<td>Minimum elevation</td>
<td>MINELEVATION</td>
<td>[number]</td>
<td>The minimum elevation for the geometries in the layer</td>
</tr>
<tr>
<td>Maximum elevation</td>
<td>MAXELEVATION</td>
<td>[number]</td>
<td>The maximum elevation for the geometries in the layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:climbalongline

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Obr. 23.17: 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.

Default menu: Vector ➤ Analysis Tools

## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polygons</td>
<td>POLYGONS</td>
<td>[vector: polygon]</td>
<td>Polygon layer whose features are associated with the count of points they contain</td>
</tr>
<tr>
<td>Points</td>
<td>POINTS</td>
<td>[vector: point]</td>
<td>Point layer with features to count</td>
</tr>
<tr>
<td>Weight field</td>
<td>WEIGHT</td>
<td>[tablefield: any]</td>
<td>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.</td>
</tr>
<tr>
<td>Class field</td>
<td>CLASSFIELD</td>
<td>[tablefield: any]</td>
<td>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.</td>
</tr>
<tr>
<td>Count field name</td>
<td>FIELD</td>
<td>[string]</td>
<td>The name of the field to store the count of points</td>
</tr>
<tr>
<td>Count</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specification of the output layer</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Resulting layer with the attribute table containing the new column with the points count</td>
</tr>
</tbody>
</table>

### 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.

Viz také:

K-means clustering
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Layer to analyze</td>
</tr>
<tr>
<td>Minimum cluster size</td>
<td>MIN_SIZE</td>
<td>[number]</td>
<td>Minimum number of features to generate a cluster</td>
</tr>
<tr>
<td>Maximum distance between clustered points</td>
<td>EPS</td>
<td>[number]</td>
<td>Distance beyond which two features cannot belong to the same cluster (eps)</td>
</tr>
<tr>
<td>Cluster field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Name of the field where the associated cluster number shall be stored</td>
</tr>
<tr>
<td>Treat border points as noise (DBSCAN*) Optional</td>
<td>DBSCAN*</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Clusters</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Vector layer for the result of the clustering</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Vector layer containing the original features with a field setting the cluster they belong to</td>
</tr>
<tr>
<td>Number of clusters</td>
<td>NUM_CLUSTERS</td>
<td>[number]</td>
<td>The number of clusters discovered</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** qgis:dbscanclustering

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

## Distance matrix

Calculates for point features distances to their nearest features in the same layer or in another layer.

**Default menu:** Vector Analysis Tools

**Viz také:**

Join attributes by nearest
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input point layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Point layer for which the distance matrix is calculated (from points)</td>
</tr>
<tr>
<td>Input unique ID field</td>
<td>INPUT_FIELD</td>
<td>[tablefield: any]</td>
<td>Field to use to uniquely identify features of the input layer. Used in the output attribute table.</td>
</tr>
<tr>
<td>Target point layer</td>
<td>TARGET</td>
<td>[vector: point]</td>
<td>Point layer containing the nearest point(s) to search (to points)</td>
</tr>
<tr>
<td>Target unique ID field</td>
<td>TARGET_FIELD</td>
<td>[tablefield: any]</td>
<td>Field to use to uniquely identify features of the target layer. Used in the output attribute table.</td>
</tr>
<tr>
<td>Output matrix type</td>
<td>MATRIX_TYPE</td>
<td>[enumeration]</td>
<td>Different types of calculation are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Linear (N * k x 3) distance matrix: for each input point, reports the distance to each of the k nearest target points. The output matrix consists of up to k rows per input point, and each row has three columns: InputID, TargetID and Distance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Standard (N x T) distance matrix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Summary distance matrix (mean, std. dev., min, max): for each input point, reports statistics on the distances to its target points.</td>
</tr>
<tr>
<td>Use only the nearest (k) target points</td>
<td>NEAREST_POINTS</td>
<td>[number] Default: 0</td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance matrix</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>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.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance matrix</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>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.</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** qgis:distancematrix

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

![Diagram of line to hub](image)

Obr. 23.18: Display the nearest hub for the red input features

Viz také:

*Distance to nearest hub (points), Join attributes by nearest*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source points layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer for which the nearest feature is searched</td>
</tr>
<tr>
<td>Destination hubs layer</td>
<td>HUBS</td>
<td>[vector: any]</td>
<td>Vector layer containing the features to search for</td>
</tr>
<tr>
<td>Hub layer name attribute</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field to use to uniquely identify features of the destination layer. Used in the output attribute table</td>
</tr>
<tr>
<td>Measurement unit</td>
<td>UNIT</td>
<td>[enumeration]</td>
<td>Units in which to report the distance to the closest feature:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Meters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Feet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Miles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Kilometers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Layer units</td>
</tr>
<tr>
<td>Hub distance</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Line vector layer for the distance matrix output</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hub distance</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Line vector layer with the attributes of the input features, the identifier of their closest feature and the calculated distance.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:distancetonearesthublinetohub

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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.

Viz také:

*Distance to nearest hub (line to hub), Join attributes by nearest*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source points layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer for which the nearest feature is searched</td>
</tr>
<tr>
<td>Destination hubs layer</td>
<td>HUBS</td>
<td>[vector: any]</td>
<td>Vector layer containing the features to search for</td>
</tr>
<tr>
<td>Hub layer name attribute</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field to use to uniquely identify features of the destination layer. Used in the output attribute table</td>
</tr>
<tr>
<td>Measurement unit</td>
<td>UNIT</td>
<td>[enumeration]</td>
<td>Units in which to report the distance to the closest feature:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Meters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Feet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Miles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Kilometers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Layer units</td>
</tr>
<tr>
<td>Hub distance</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Point vector layer for the distance matrix output.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hub distance</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Point vector layer with the attributes of the input features, the identifier of their closest feature and the calculated distance.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:distancetonearesthubpoints

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

Obr. 23.19: Join points based on a common field / attribute
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hub layer</td>
<td>HUBS</td>
<td>[vector: any]</td>
<td>Input layer</td>
</tr>
<tr>
<td>Hub ID field</td>
<td>HUB_FIELD</td>
<td>[tablefield: any]</td>
<td>Field of the hub layer with ID to join</td>
</tr>
<tr>
<td>Hub layer fields to copy (leave empty to copy all fields)</td>
<td>HUB_FIELDS</td>
<td>[tablefield: any]</td>
<td>The field(s) of the hub layer to be copied. If no field(s) are chosen all fields are taken.</td>
</tr>
<tr>
<td>Spoke layer</td>
<td>SPOKES</td>
<td>[vector: any]</td>
<td>Additional spoke point layer</td>
</tr>
<tr>
<td>Spoke ID field</td>
<td>SPOKE_FIELD</td>
<td>[tablefield: any]</td>
<td>Field of the spoke layer with ID to join</td>
</tr>
<tr>
<td>Spoke layer fields to copy (leave empty to copy all fields)</td>
<td>SPOKE_FIELDS</td>
<td>[tablefield: any]</td>
<td>Field(s) of the spoke layer to be copied. If no fields are chosen all fields are taken.</td>
</tr>
<tr>
<td>Create geodesic lines</td>
<td>GEODESIC</td>
<td>[boolean]</td>
<td>Create geodesic lines (the shortest path on the surface of an ellipsoid)</td>
</tr>
<tr>
<td>Distance between vertices (geodesic lines only)</td>
<td>GEODESIC_DISTANCE</td>
<td>[number]</td>
<td>Distance between consecutive vertices (in kilometers). A smaller distance results in a denser, more accurate line</td>
</tr>
<tr>
<td>Split lines at antimeridian (±180 degrees longitude)</td>
<td>ANTIMERIDIAN_SPLIT</td>
<td>[boolean]</td>
<td>Split lines at ±180 degrees longitude (to improve rendering of the lines)</td>
</tr>
<tr>
<td>Hub lines</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The resulting line layer</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hub lines</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The resulting line layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:hublines`

```python
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 **Using processing algorithms from the console** for details on how to run processing algorithms from the Python console.

### 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.

*Viz také:*  
`DBSCAN clustering`
Obr. 23.20: A five class point clusters
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer to analyze</td>
</tr>
<tr>
<td>Number of clusters</td>
<td>CLUSTERS</td>
<td>[number]</td>
<td>Number of clusters to create with the features</td>
</tr>
<tr>
<td>Cluster field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Name of the cluster number field</td>
</tr>
<tr>
<td>Clusters</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Vector layer for generated the clusters</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Vector layer containing the original features with a field specifying the cluster they belong to</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:kmeansclustering

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

List unique values

Lists unique values of an attribute table field and counts their number.

Default menu: Vector ☰ Analysis Tools

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer to analyze</td>
</tr>
<tr>
<td>Target field(s)</td>
<td>FIELDS</td>
<td>[tablefield: any]</td>
<td>Field to analyze</td>
</tr>
<tr>
<td>Unique values</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>Summary table layer with unique values</td>
</tr>
<tr>
<td>HTML report</td>
<td>OUTPUT_HTML_FILE</td>
<td>[html]</td>
<td>HTML report of unique values in the Processing ☰ Results viewer</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique values</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>Summary table layer with unique values</td>
</tr>
<tr>
<td>HTML report</td>
<td>OUTPUT_HTML_FILE</td>
<td>[html]</td>
<td>HTML report of unique values. Can be opened from the Processing Results viewer</td>
</tr>
<tr>
<td>Total unique values</td>
<td>TOTAL_VALUES</td>
<td>[number]</td>
<td>The number of unique values in the input field</td>
</tr>
<tr>
<td>UNIQUE_VALUES</td>
<td>Unique values</td>
<td>[string]</td>
<td>A string with the comma separated list of unique values found in the input field</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:listuniquevalues

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Weight field</td>
<td>WEIGHT</td>
<td>[tablefield: numeric]</td>
<td>Field to use if you want to perform a weighted mean</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unique ID field</td>
<td>UID</td>
<td>[tablefield: numeric]</td>
<td>Unique field on which the calculation of the mean will be made</td>
</tr>
<tr>
<td>Mean coordinates</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The (point vector) layer for the result</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean coordinates</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Resulting point(s) layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:meancoordinates

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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

**Viz také:**

Join attributes by nearest

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Point vector layer to calculate the statistics on</td>
</tr>
<tr>
<td>Nearest neighbour</td>
<td>OUTPUT_HTML_FILE</td>
<td>[html]</td>
<td>HTML file for the computed statistics</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest neighbour</td>
<td>OUTPUT_HTML_FILE [html]</td>
<td>[html]</td>
<td>HTML file with the computed statistics</td>
</tr>
<tr>
<td>Observed mean distance</td>
<td>OBSERVED_MD [number]</td>
<td>[number]</td>
<td>Observed mean distance</td>
</tr>
<tr>
<td>Expected mean distance</td>
<td>EXPECTED_MD [number]</td>
<td>[number]</td>
<td>Expected mean distance</td>
</tr>
<tr>
<td>Nearest neighbour index</td>
<td>NN_INDEX [number]</td>
<td>[number]</td>
<td>Nearest neighbour index</td>
</tr>
<tr>
<td>Number of points</td>
<td>POINT_COUNT [number]</td>
<td>[number]</td>
<td>Number of points</td>
</tr>
<tr>
<td>Z-Score</td>
<td>Z_SCORE [number]</td>
<td>[number]</td>
<td>Z-Score</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: `qgis:nearestneighbouranalysis`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input layer.</td>
</tr>
<tr>
<td>Overlap layers</td>
<td>LAYERS</td>
<td>[vector: any][list]</td>
<td>The overlay layers.</td>
</tr>
<tr>
<td>Output layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Create temporary layer]</td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>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.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:overlapanalysis

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Statistics by categories

Calculates statistics of a field depending on a parent class. The parent class is a combination of values from other fields.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input vector layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer with unique classes and values</td>
</tr>
<tr>
<td>Field to calculate</td>
<td>VALUES_FIELD_NAME</td>
<td>[tablefield: any]</td>
<td>If empty only the count will be calculated</td>
</tr>
<tr>
<td>statistics on (if empty, only count is calculated) Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field(s) with categories</td>
<td>CATEGORIES_FIELD_NAME</td>
<td>[vector: any]</td>
<td>The fields that (combined) define the categories</td>
</tr>
<tr>
<td>Statistics by category</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>Table for the generated statistics</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics by category</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>Table containing the statistics</td>
</tr>
</tbody>
</table>

Depending on the type of the field being analyzed, the following statistics are returned for each grouped value:

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Retěžec</th>
<th>Numeric</th>
<th>Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count (COUNT)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.46 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Retézec</th>
<th>Numeric</th>
<th>Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique values (UNIQUE)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Empty (null) values (EMPTY)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Non-empty values (FILLED)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Minimal value (MIN)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Maximal value (MAX)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Range (RANGE)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Sum (SUM)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Mean value (MEAN)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Median value (MEDIAN)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Standard Deviation (STD_DEV)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Coefficient of variation (CV)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Minority (rarest occurring value - MINORITY)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Majority (most frequently occurring value - MAJORITY)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>First Quartile (FIRSTQUARTILE)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Third Quartile (THIRDQUARTILE)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Inter Quartile Range (IQR)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Minimum Length (MIN_LENGTH)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Mean Length (MEAN_LENGTH)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Maximum Length (MAX_LENGTH)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:statisticsbycategories

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Default menu: Vector \[\] Analysis Tools
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines</td>
<td>LINES</td>
<td>[vector: line]</td>
<td>Input vector line layer</td>
</tr>
<tr>
<td>Polygons</td>
<td>POLYGONS</td>
<td>[vector: polygon]</td>
<td>Polygon vector layer</td>
</tr>
<tr>
<td>Lines length field name</td>
<td>LEN_FIELD</td>
<td>[string]</td>
<td>Name of the field for the lines length</td>
</tr>
<tr>
<td>Lines count field name</td>
<td>COUNT_FIELD</td>
<td>[string]</td>
<td>Name of the field for the lines count</td>
</tr>
<tr>
<td>Line length</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output polygon vector layer</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line length</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Polygon output layer with fields of lines length and line count</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:sumlinelengths

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.1.13 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.

Obr. 23.21: In blue the source layer, in red the offset one

Allows features in-place modification
**Viz také:**

*Offset lines, Array of translated features*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer to use for the offsets.</td>
</tr>
<tr>
<td>Number of features to create</td>
<td>COUNT</td>
<td>[number ≥ 1] Default: 10</td>
<td>Number of offset copies to generate for each feature</td>
</tr>
<tr>
<td>Offset step distance</td>
<td>OFFSET</td>
<td>[number ≥ 1] Default: 1.0</td>
<td>Distance between two consecutive offset copies</td>
</tr>
<tr>
<td>Segments</td>
<td>SEGMENTS</td>
<td>[number] Default: 8</td>
<td>Number of line segments to use to approximate a quarter circle when creating rounded offsets</td>
</tr>
<tr>
<td>Join style</td>
<td>JOIN_STYLE</td>
<td>[enumeration] Default: 0</td>
<td>Specify whether round, miter or beveled joins should be used when offsetting corners in a line. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Round</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Miter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Bevel</td>
</tr>
<tr>
<td>Miter limit</td>
<td>MITER_LIMIT</td>
<td>[number] Default: 2.0</td>
<td>Only applicable for mitered join styles, and controls the maximum distance from the offset curve to use when creating a mitered join.</td>
</tr>
<tr>
<td>Offset lines</td>
<td>OUTPUT</td>
<td>[vector: line] Default: [Create temporary layer]</td>
<td>Specify the output line layer with offset features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset lines</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Output line layer with offset features. The original features are also copied.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:arrayoffsetlines

```python
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 [Using processing algorithms from the console](#) for details on how to run processing algorithms from the Python console.
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.

![Image of array of translated features]

Obr. 23.22: Input layers in blue tones, output layers with translated features in red tones

- Allows *features in-place modification*
- Viz také:
  - *Translate, Array of offset (parallel) lines*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer to translate</td>
</tr>
<tr>
<td>Number of features to create</td>
<td>COUNT</td>
<td>[number ]</td>
<td>Number of copies to generate for each feature</td>
</tr>
<tr>
<td>Step distance (x-axis)</td>
<td>DELTA_X</td>
<td>[number ]</td>
<td>Displacement to apply on the X axis</td>
</tr>
<tr>
<td>Step distance (y-axis)</td>
<td>DELTA_Y</td>
<td>[number ]</td>
<td>Displacement to apply on the Y axis</td>
</tr>
<tr>
<td>Step distance (z-axis)</td>
<td>DELTA_Z</td>
<td>[number ]</td>
<td>Displacement to apply on the Z axis</td>
</tr>
<tr>
<td>Step distance (m values)</td>
<td>DELTA_M</td>
<td>[number ]</td>
<td>Displacement to apply on M</td>
</tr>
</tbody>
</table>

Pokračujte na dali stránce
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Translated  | OUTPUT        | [same as input]         | Output vector layer with translated (moved) copies of the features. The original features are also copied. One of:  
• Create Temporary Layer (TEMPORARY_OUTPUT)  
• Save to File...  
• Save to Geopackage...  
• Save to PostGIS Table...  
The file encoding can also be changed here. |

### Python code

**Algorithm ID:** `qgis:arraytranslatedfeatures`  

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### Create grid

Creates a vector layer with a grid covering a given extent. Grid cells can have different shapes:

![Different grid cell shapes](image)

Obr. 23.23: Different grid cell shapes

The size 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*
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grid type</strong></td>
<td>TYPE</td>
<td>[enumeration]</td>
<td>Shape of the grid. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Point</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Line</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Rectangle (polygon)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Diamond (polygon)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Hexagon (polygon)</td>
</tr>
<tr>
<td><strong>Grid extent</strong></td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Extent of the grid</td>
</tr>
<tr>
<td><strong>Horizontal spacing</strong></td>
<td>HSPACING</td>
<td>[number]</td>
<td>Size of the grid cell on the X-axis</td>
</tr>
<tr>
<td><strong>Vertical spacing</strong></td>
<td>VSPACING</td>
<td>[number]</td>
<td>Size of the grid cell on the Y-axis</td>
</tr>
<tr>
<td><strong>Horizontal overlay</strong></td>
<td>HOVERLAY</td>
<td>[number]</td>
<td>Overlay distance between two consecutive grid cells on the X-axis</td>
</tr>
<tr>
<td><strong>Vertical overlay</strong></td>
<td>VOVERLAY</td>
<td>[number]</td>
<td>Overlay distance between two consecutive grid cells on the Y-axis</td>
</tr>
<tr>
<td><strong>Grid CRS</strong></td>
<td>CRS</td>
<td>[crs]</td>
<td>Coordinate reference system to apply to the grid</td>
</tr>
<tr>
<td><strong>Grid</strong></td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Resulting vector grid layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer] (TEMPORARY_OUTPUT)</td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grid</strong></td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Resulting vector grid layer. The output geometry type (point, line or polygon) depends on the Grid type.</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** qgis:creategrid

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer or a table.</td>
</tr>
<tr>
<td>X field</td>
<td>XFIELD</td>
<td>[tablefield: any]</td>
<td>Field containing the X coordinate</td>
</tr>
<tr>
<td>Y field</td>
<td>YFIELD</td>
<td>[tablefield: any]</td>
<td>Field containing the Y coordinate</td>
</tr>
<tr>
<td>Z field</td>
<td>ZFIELD</td>
<td>[tablefield: any]</td>
<td>Field containing the Z coordinate</td>
</tr>
<tr>
<td>M field</td>
<td>MFIELD</td>
<td>[tablefield: any]</td>
<td>Field containing the M value</td>
</tr>
<tr>
<td>Target CRS</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td>Coordinate reference system to use for layer. The provided coordinates are assumed to be compliant.</td>
</tr>
<tr>
<td>Points from table</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the resulting point layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points from table</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The resulting point layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:createpointslayerfromtable

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

![Image of points along line](image)

Obr. 23.24: Points of the pixel centroids

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Vector layer</td>
<td>INPUTVECTOR</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Points along line</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Resulting point layer with pixel centroids. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points along line</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Resulting point layer with pixel centroids</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:generatepointspixelcentroidsalongline

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

Obr. 23.25: Points of the pixel centroids
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Vector layer</td>
<td>INPUT_VECTOR</td>
<td>[vector: polygon]</td>
<td>Input polygon vector layer</td>
</tr>
<tr>
<td>Points inside polygons</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Resulting point layer of pixel centroids. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points inside polygons</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Resulting point layer of pixel centroids</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:generatepointspixelcentroidsinsidepolygons

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Import geotagged photos

Creates a point layer corresponding to the geotagged locations from JPEG 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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input folder</td>
<td>FOLDER</td>
<td>[folder]</td>
<td>Path to the source folder containing the geotagged photos</td>
</tr>
<tr>
<td>Scan recursively</td>
<td>RECURSIVE</td>
<td>[boolean]</td>
<td>If checked, the folder and its subfolders will be scanned</td>
</tr>
</tbody>
</table>
Tabulka 23.52 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Photos</strong></td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the point vector layer for the geotagged photos. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td><strong>Invalid photos table</strong></td>
<td>INVALID</td>
<td>[table]</td>
<td>Specify the table of unreadable or non-geotagged photos. One of:</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Photos</strong></td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Point vector layer with geotagged photos. The form of the layer is automatically filled with paths and photo previews settings.</td>
</tr>
<tr>
<td><strong>Invalid photos table</strong></td>
<td>INVALID</td>
<td>[table]</td>
<td>Table of unreadable or non-geotagged photos can also be created.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID**: qgis:importphotos

```python
import processing
def run(algorithm_id, parameter_dictionary):
    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 [Using processing algorithms from the console](Using processing algorithms from the console) for details on how to run processing algorithms from the Python console.

**Points to path**

Converts a point layer to a line layer, by joining points in an order defined by a field in the input point layer (if the order field is a date/time field, the format must be specified).

Points can be grouped by a field 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.
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input point layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Order field</td>
<td>ORDER_FIELD</td>
<td>[tablefield: any]</td>
<td>Field containing the order to connect the points in the path</td>
</tr>
<tr>
<td>Group field Optional</td>
<td>GROUP_FIELD</td>
<td>[tablefield: any]</td>
<td>Point features of the same value in the field will be grouped in the same line. If not set, a single path is drawn with all the input points.</td>
</tr>
<tr>
<td>Date format (if order field is Date/Time) Optional</td>
<td>DATE_FORMAT</td>
<td>[string]</td>
<td>The format to use for the Order field parameter. Specify this only if the Order field is of type Date/Time.</td>
</tr>
<tr>
<td>Paths</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the line vector layer of the path. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Directory for text output</td>
<td>OUTPUT_TEXT_DIR</td>
<td>[folder]</td>
<td>Specify the directory that will contain the description files of points and paths. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to a Temporary Directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Directory...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paths</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Line vector layer of the path</td>
</tr>
<tr>
<td>Directory for text output</td>
<td>OUTPUT</td>
<td>[folder]</td>
<td>Directory containing description files of points and paths</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:pointstopath

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
Random points along line

Creates a new point layer, with points placed in the lines of another layer.
For each line in the input layer, a given number of points is added to the resulting layer. A minimum distance can be specified, to avoid points being too close to each other.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input point layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Number of points</td>
<td>POINTS_NUMBER</td>
<td>[number]</td>
<td>Number of points to create</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1</td>
<td></td>
</tr>
<tr>
<td>Minimum distance between points</td>
<td>MIN_DISTANCE</td>
<td>[number]</td>
<td>The minimum distance between points</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:qgisrandompointsalongline

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Random points in extent

Creates a new point layer with a given number of random points, all of them within a given extent.
A minimum distance can be specified, to avoid points being too close to each other.

**Default menu:** Vector Research Tools
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Map extent for the random points</td>
</tr>
<tr>
<td>Number of points</td>
<td>POINTS_NUMBER</td>
<td>[number]</td>
<td>Number of point to create</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1</td>
<td></td>
</tr>
<tr>
<td>Minimum distance between points</td>
<td>MIN_DISTANCE</td>
<td>[number]</td>
<td>The minimum distance between points</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Target CRS</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td>CRS of the random points layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: Project CRS</td>
<td></td>
</tr>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points layer.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:randompointsinextent

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input polygon layer defining the area</td>
</tr>
<tr>
<td>Number of points</td>
<td>POINTS_NUMBER</td>
<td>[number]</td>
<td>Number of points to create</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1</td>
<td></td>
</tr>
<tr>
<td>Minimum distance between points</td>
<td>MIN_DISTANCE</td>
<td>[number]</td>
<td>The minimum distance between points</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:randompointsinlayerbounds

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input polygon vector layer</td>
</tr>
<tr>
<td>Sampling strategy</td>
<td>STRATEGY</td>
<td>[enumeration]</td>
<td>Sampling strategy to use. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Points count: number of points for each feature</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Points density: density of points for each feature</td>
</tr>
<tr>
<td>Point count or density</td>
<td>VALUE</td>
<td>[number]</td>
<td>The number or density of points, depending on the chosen Sampling strategy.</td>
</tr>
<tr>
<td>Minimum distance between points</td>
<td>MIN_DISTANCE</td>
<td>[number]</td>
<td>The minimum distance between points</td>
</tr>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output random points layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:randompointsinsidepolygons

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>RASTER_BAND</td>
<td>[raster band]</td>
<td>Raster band to extract data from</td>
</tr>
<tr>
<td>Field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Name of the field to store the raster band value</td>
</tr>
<tr>
<td>Vector points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the resulting point layer of pixels centroids. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Resulting point layer with pixels centroids</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:pixelstopoints

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>RASTER_BAND</td>
<td>[raster band]</td>
<td>Raster band to extract data from</td>
</tr>
<tr>
<td>Field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Name of the field to store the raster band value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vector polygons</th>
<th>OUTPUT</th>
<th>[vector: polygon]</th>
<th>Specify the resulting polygon layer of pixel extents. One of:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector polygons</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Resulting polygon layer of pixel extents</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:pixelstopolygons

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input extent (xmin, xmax, ymin, ymax)</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Map extent for the random points</td>
</tr>
<tr>
<td>Point spacing/count</td>
<td>SPACING</td>
<td>[number] Default: 100</td>
<td>Spacing between the points, or the number of points, depending on whether Use point spacing is checked or not.</td>
</tr>
<tr>
<td>Initial inset from corner (LH side)</td>
<td>INSET</td>
<td>[number] Default: 0.0</td>
<td>Offsets the points relative to the upper left corner. The value is used for both the X and Y axis.</td>
</tr>
<tr>
<td>Apply random offset to point spacing</td>
<td>RANDOMIZE</td>
<td>[boolean] Default: False</td>
<td>If checked the points will have a random spacing</td>
</tr>
<tr>
<td>Use point spacing</td>
<td>IS_SPACING</td>
<td>[boolean] Default: True</td>
<td>If unchecked the point spacing is not taken into account</td>
</tr>
<tr>
<td>Output layer CRS</td>
<td>CRS</td>
<td>[crs] Default: Project CRS</td>
<td>CRS of the random points layer</td>
</tr>
<tr>
<td>Regular points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output regular point layer. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Save to File… • Save to Geopackage… • Save to PostGIS Table… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output regular point layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:regularpoints

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
23.1.14 Vector general

Assign projection

Assigns a new projection to a vector layer.

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.

Viz také:

Define Shapefile projection, Find projection, Reproject layer

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer with wrong or missing CRS</td>
</tr>
<tr>
<td>Assigned CRS</td>
<td>CRS</td>
<td>[crs]</td>
<td>Select the new CRS to assign to the vector layer</td>
</tr>
<tr>
<td>Assigned CRS (Optional)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output layer containing only the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>duplicates. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned CRS</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with assigned projection</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:assignprojection

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Build virtual vector

Creates a virtual vector layer that contains a set of vector layers. The output virtual vector layer will not be open 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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input datasources</td>
<td>INPUT</td>
<td>[vector: any] [list]</td>
<td>Select the vector layers you want to use to build the virtual vector</td>
</tr>
<tr>
<td>Create „unioned“ VRT</td>
<td>UNIONED</td>
<td>[boolean]</td>
<td>Check if you want to unite all the vectors in a single vrt file</td>
</tr>
<tr>
<td>Virtual vector</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output layer containing only the duplicates. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual vector</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>The output virtual vector made from the chosen sources</td>
</tr>
</tbody>
</table>

Convert layer to spatial bookmarks

Creates spatial bookmarks corresponding to the extent of features contained in a layer.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>The input vector layer</td>
</tr>
<tr>
<td>Bookmark destination</td>
<td>DESTINATION</td>
<td>[enumeration]</td>
<td>Select the destination for the bookmarks. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Project bookmarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — User bookmarks</td>
</tr>
<tr>
<td>Name field</td>
<td>NAME_EXPRESSION</td>
<td>[expression]</td>
<td>Field or expression that will give names to the generated bookmarks</td>
</tr>
<tr>
<td>Group field</td>
<td>GROUP_EXPRESSION</td>
<td>[expression]</td>
<td>Field or expression that will provide groups for the generated bookmarks</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count of bookmarks added</td>
<td>COUNT</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: `qgis:layertobookmarks`

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookmark source</td>
<td>SOURCE</td>
<td>[enumeration] [list]</td>
<td>Select the source(s) of the bookmarks. One or more of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [0,1]</td>
<td>• 0 — Project bookmarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — User bookmarks</td>
</tr>
<tr>
<td>Output CRS</td>
<td>CRS</td>
<td>[crs]</td>
<td>The CRS of the output layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: EPSG:4326</td>
<td>- WGS 84</td>
</tr>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output (bookmarks) vector layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:bookmarkstolayer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Select the vector layer you want to create an attribute index for</td>
</tr>
<tr>
<td>Attribute to index</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field of the vector layer</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexed layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>A copy of the input vector layer with an index for the specified field</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:createattributeindex

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexed layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>A copy of the input vector layer with a spatial index</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:createspatialindex

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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 *Assign projection* algorithm, it modifies the current layer and will not output a new layer.

**Poznámka:** 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

**Viz také:**

*Assign projection, Find projection, Reproject layer*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer with missing projection information</td>
</tr>
<tr>
<td>CRS</td>
<td>CRS</td>
<td>[crs]</td>
<td>Select the CRS to assign to the vector layer</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT</td>
<td>INPUT</td>
<td>[same as input]</td>
<td>The input vector layer with the defined projection</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:definecurrentprojection

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

Remove null geometries, Delete duplicates by attribute

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The layer with duplicate geometries you want to clean</td>
</tr>
<tr>
<td>Cleaned</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count of discarded duplicate records</td>
<td>DUPLICATE_COUNT</td>
<td>[number]</td>
<td>Count of discarded duplicate records</td>
</tr>
<tr>
<td>Cleaned</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output layer without any duplicated geometries</td>
</tr>
<tr>
<td>Count of retained records</td>
<td>RETAINED_COUNT</td>
<td>[number]</td>
<td>Count of unique records</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis:deleteduplicategeometries`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

**Viz také:**

*Delete duplicate geometries*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input layer</td>
</tr>
<tr>
<td>Fields to match duplicates by</td>
<td>FIELDS</td>
<td>[tablefield: any]</td>
<td>Fields defining duplicates. Features with identical values for all these fields are considered duplicates.</td>
</tr>
</tbody>
</table>
| Filtered (no duplicates) | OUTPUT  | [same as input] | Specify the output layer containing the unique features. One of:
|                           |         |                 | • Create Temporary Layer
|                           |         |                 | (TEMPORARY_OUTPUT)
|                           |         |                 | • Save to File...
|                           |         |                 | • Save to Geopackage...
|                           |         |                 | • Save to PostGIS Table...
|                           |         |                 | The file encoding can also be changed here. |
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtered (duplicates) (Optional)</td>
<td>DUPLICATES</td>
<td>[same as input]</td>
<td>Vector layer containing the removed features. Will not be produced if not specified (left as [Skip output]).</td>
</tr>
<tr>
<td>Count of discarded duplicate records</td>
<td>DUPLICATE_COUNT</td>
<td>[number]</td>
<td>Count of discarded duplicate records</td>
</tr>
<tr>
<td>Filtered (no duplicates)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer containing the unique features.</td>
</tr>
<tr>
<td>Count of retained records</td>
<td>RETAINED_COUNT</td>
<td>[number]</td>
<td>Count of unique records</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID**: `qgis:deleteduplicatesbyattribute`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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

**Viz také**:

*Delete duplicate geometries, Remove null geometries*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer</td>
</tr>
<tr>
<td>Dropped geometries</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>Specify the output geometryless layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropped geometries</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>The output geometryless layer. A copy of the original attribute table.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:dropgeometries

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Execute SQL

Runs a simple or complex query 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 particularly 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.

**Viz také:**

*Spatialite execute SQL, PostgreSQL execute SQL*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional input datasources (called input1, .., inputN in the query)</td>
<td>INPUT_DATASOURCES</td>
<td>[vector: any] [list]</td>
<td>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.</td>
</tr>
<tr>
<td>SQL query</td>
<td>INPUT_QUERY</td>
<td>[string]</td>
<td>Type the string of your SQL query, e.g. SELECT * FROM input1.</td>
</tr>
<tr>
<td>Unique identifier field Optional</td>
<td>INPUT_UID_FIELD</td>
<td>[string]</td>
<td>Specify the column with unique ID</td>
</tr>
<tr>
<td>Geometry field Optional</td>
<td>INPUT_GEOMETRY_FIELD</td>
<td>[string]</td>
<td>Specify the geometry field</td>
</tr>
<tr>
<td>Geometry type Optional</td>
<td>INPUT_GEOMETRY_TYPE</td>
<td>[enumeration]</td>
<td>Choose the geometry of the result. By default the algorithm will autodetect it. One of: 0 — Autodetect 1 — No geometry 2 — Point 3 — LineString 4 — Polygon 5 — MultiPoint 6 — MultiLineString 7 — MultiPolygon</td>
</tr>
<tr>
<td>CRS Optional</td>
<td>INPUT_GEOMETRY_CRS</td>
<td>[crs]</td>
<td>The CRS to assign to the output layer</td>
</tr>
<tr>
<td>SQL Output</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Specify the output layer created by the query. One of:  • Create Temporary Layer (TEMPORARY_OUTPUT)  • Save to File…  • Save to Geopackage…  • Save to PostGIS Table…  The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Output</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Vector layer created by the query</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:executesql

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Extract selected features

Saves the selected features as a new layer.

**Poznámka**: If the selected layer has no selected features, the newly created layer will be empty.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer to save the selection from</td>
</tr>
<tr>
<td>Selected features</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the vector layer for the selected features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected features</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with only the selected features, or no feature if none was selected.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID**: qgis:saveselectedfeatures

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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.

**Viz také:**

- Assign projection, Define Shapefile projection, Reproject layer
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer with unknown projection</td>
</tr>
<tr>
<td>Target area for layer</td>
<td>TARGET_AREA</td>
<td>[extent]</td>
<td>The area that the layer covers. The options for specifying the extent are:</td>
</tr>
<tr>
<td>(xmin, xmax, ymin, ymax)</td>
<td></td>
<td></td>
<td>• Use Canvas Extent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Select Extent on Canvas</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Use Layer Extent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It is also possible to provide the extent coordinates directly (xmin, xmax, ymin, ymax).</td>
</tr>
<tr>
<td>CRS candidates</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>Specify the table (geometryless layer) for the CRS suggestions (EPSG codes). One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRS candidates</td>
<td>OUTPUT</td>
<td>[table]</td>
<td>A table with all the CRS (EPSG codes) of the matching criteria.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:findprojection

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

**Viz také:**

*Join attributes by nearest, Join attributes by location*
 Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Layer</strong></td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer. The output layer will consist of the features of this layer with attributes from matching features in the second layer.</td>
</tr>
<tr>
<td><strong>Table field</strong></td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field of the source layer to use for the join</td>
</tr>
<tr>
<td><strong>Input layer 2</strong></td>
<td>INPUT_2</td>
<td>[vector: any]</td>
<td>Layer with the attribute table to join</td>
</tr>
<tr>
<td><strong>Table field 2</strong></td>
<td>FIELD_2</td>
<td>[tablefield: any]</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Layer 2 fields to copy</strong></td>
<td>FIELDS_TO_COPY</td>
<td>[tablefield: any]</td>
<td>Select the specific fields you want to add. By default all the fields are added.</td>
</tr>
</tbody>
</table>
| **Join type**              | METHOD      | [enumeration]         | 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) |
| **Discard records which could not be joined** | DISCARD_NONMATCHING | [boolean]  | Check if you don’t want to keep the features that could not be joined |
| **Joined field prefix**    | PREFIX      | [string]              | Add a prefix to joined fields in order to easily identify them and avoid field name collision |
| **Joined layer**           | OUTPUT      | [same as input]       | Specify the output vector layer for the join. One of:  
  • Create Temporary Layer (TEMPORARY_OUTPUT)  
  • Save to File…  
  • Save to Geopackage…  
  • Save to PostGIS Table…  
  The file encoding can also be changed here. |
| **Unjoinable features from first layer** | NON_MATCHING | [same as input]       | Specify the output vector layer for unjoinable features from first layer. One of:  
  • Skip output  
  • Create Temporary Layer (TEMPORARY_OUTPUT)  
  • Save to File…  
  • Save to Geopackage…  
  • Save to PostGIS Table…  
  The file encoding can also be changed here. |
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of joined features from input table</td>
<td>JOINED_COUNT</td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>Unjoinable features from first layer (Optional)</td>
<td>NON_MATCHING</td>
<td>[same as input]</td>
<td>Vector layer with non-matched features</td>
</tr>
<tr>
<td>Joined layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer with added attributes from the join</td>
</tr>
<tr>
<td>Number of unjoinable features from input table (Optional)</td>
<td>UNJOINABLE_COUNT</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:joinattributestable

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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

**Viz také:**

Join attributes by nearest, Join attributes by field value, Join attributes by location (summary)

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer. The output layer will consist of the features of this layer with attributes from matching features in the second layer.</td>
</tr>
<tr>
<td>Join layer</td>
<td>JOIN</td>
<td>[vector: any]</td>
<td>The attributes of this vector layer will be added to the source layer attribute table.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric predicate</td>
<td>PREDICATE</td>
<td>[enumeration] [list]</td>
<td>[enumeration] [list]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [0]</td>
<td>Select the geometric criteria. One or more of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 0 — intersects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1 — contains</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 2 — equals</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 3 — touches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 4 — overlaps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 5 — within</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 6 — crosses</td>
</tr>
<tr>
<td>Fields to add (leave empty to use all fields)</td>
<td>JOIN_FIELDS</td>
<td>[tablefield: any] [list]</td>
<td>Select the specific fields you want to add. By default all the fields are added.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Join type</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>The type of the final joined layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 0 — Create separate feature for each matching feature (one-to-many)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1 — Take attributes of the first matching feature only (one-to-one)</td>
</tr>
<tr>
<td>Discard records which could not be joined</td>
<td>DISCARD_NONMATCHING</td>
<td>[boolean]</td>
<td>Remove from the output the input layer records which could not be joined</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joined field prefix</td>
<td>PREFIX</td>
<td>[string]</td>
<td>Add a prefix to joined fields in order to easily identify them and avoid field name collision</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joined layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer for the join. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Unjoinable features from first layer</td>
<td>NON_MATCHING</td>
<td>[same as input]</td>
<td>Specify the output vector layer for unjoinable features from first layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Skip output]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Skip output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of joined features from input table</td>
<td>JOINED_COUNT</td>
<td>[number]</td>
<td>Vector layer of the non-matched features</td>
</tr>
<tr>
<td>Unjoinable features from first layer Optional</td>
<td>NON_MATCHING</td>
<td>[same as input]</td>
<td>Vector layer of the non-matched features</td>
</tr>
<tr>
<td>Joined layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer with added attributes from the join</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:joinattributesbylocation

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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).

**Viz také:**

*Join attributes by location*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer. The output layer will consist of the features of this layer with attributes from matching features in the second layer.</td>
</tr>
<tr>
<td>Join layer</td>
<td>JOIN</td>
<td>[vector: any]</td>
<td>The attributes of this vector layer will be added to the source layer attribute table.</td>
</tr>
</tbody>
</table>

Podrobné informace: [Pokračujte na další stránce](#)
<table>
<thead>
<tr>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Geometric predicate</strong></td>
<td>PREDICATE</td>
<td>Select the geometric criteria. One or more of: 0 — intersects 1 — contains 2 — equals 3 — touches 4 — overlaps 5 — within 6 — crosses</td>
</tr>
<tr>
<td><strong>Fields to summarize (leave empty to use all fields)</strong></td>
<td>JOIN_FIELDS</td>
<td>Select the specific fields you want to add and summarize. By default all the fields are added.</td>
</tr>
<tr>
<td><strong>Summaries to calculate (leave empty to use all fields)</strong></td>
<td>SUMMARIES</td>
<td>Choose which type of summary you want to add to each field and for each feature. 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</td>
</tr>
<tr>
<td><strong>Discard records which could not be joined</strong></td>
<td>DISCARD_NONMATCHING</td>
<td>Remove from the output the input layer records which could not be joined</td>
</tr>
<tr>
<td><strong>Joined layer</strong></td>
<td>OUTPUT</td>
<td>Specify the output vector layer for the join. One of: Create Temporary Layer (TEMPORARY_OUTPUT) Save to File... Save to Geopackage... Save to PostGIS Table... The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joined layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer with summarized attributes from the join</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:joinbylocationsummary

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

**Viz také:**

- Near neighbour analysis
- Join attributes by field value
- Join attributes by location
- Distance matrix

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input layer.</td>
</tr>
<tr>
<td>Input layer 2</td>
<td>INPUT_2</td>
<td>[vector: any]</td>
<td>The join layer.</td>
</tr>
<tr>
<td>Layer 2 fields to copy (leave empty to copy all fields)</td>
<td>FIELDS_TO_COPY</td>
<td>[fields]</td>
<td>Join layer fields to copy (if empty, all fields will be copied).</td>
</tr>
<tr>
<td>Discard records which could not be joined</td>
<td>DISCARD_NONMATCH</td>
<td>[boolean]</td>
<td>Remove from the output the input layer records which could not be joined</td>
</tr>
<tr>
<td>Joined field prefix</td>
<td>PREFIX</td>
<td>[string]</td>
<td>Joined field prefix</td>
</tr>
<tr>
<td>Maximum nearest neighbors</td>
<td>NEIGHBORS</td>
<td>[number]</td>
<td>Maximum number of nearest neighbors</td>
</tr>
<tr>
<td>Maximum distance</td>
<td>MAX_DISTANCE</td>
<td>[number]</td>
<td>Maximum search distance</td>
</tr>
</tbody>
</table>
Tabulka 23.63 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joined layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the vector layer containing the joined features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Unjoinable features</td>
<td>NON_MATCHING</td>
<td>[same as input]</td>
<td>Specify the vector layer containing the features that could not be joined. One of:</td>
</tr>
<tr>
<td>from first layer</td>
<td></td>
<td></td>
<td>• Skip output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joined layer</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output joined layer.</td>
</tr>
<tr>
<td>Unjoinable features</td>
<td>NON_MATCHING</td>
<td>[same as input]</td>
<td>Layer containing the features from first layer that could not be joined to any features in the join layer.</td>
</tr>
<tr>
<td>from first layer</td>
<td></td>
<td>[number]</td>
<td>Number of features from the input table that have been joined.</td>
</tr>
<tr>
<td>Number of joined</td>
<td>JOINED_COUNT</td>
<td>[number]</td>
<td>Number of features from the input table that could not be joined.</td>
</tr>
<tr>
<td>features from input</td>
<td></td>
<td>[number]</td>
<td></td>
</tr>
<tr>
<td>table</td>
<td>UNJOINABLE_COUNT</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:joinbynearest

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Merge vector layers

Combines multiple vector layers of the same geometry type into a single one.

If attributes tables are different, the attribute table of the resulting layer will contain the attributes from all input layers. Non-matching fields will be appended at the end of the attribute table.

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

**Viz také:**

*Split vector layer*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Layers</strong></td>
<td>LAYERS</td>
<td>[vector: any] [list]</td>
<td>The layers that are to be merged into a single layer. Layers should be of the same geometry type.</td>
</tr>
<tr>
<td><strong>Destination CRS</strong> Optional</td>
<td>CRS</td>
<td>[crs]</td>
<td>Choose the CRS for the output layer. If not specified, the CRS of the first input layer is used.</td>
</tr>
<tr>
<td><strong>Merged</strong></td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Merged</strong></td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer containing all the features and attributes from the input layers.</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:mergevectorlayers

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer to sort</td>
</tr>
<tr>
<td>Expression</td>
<td>EXPRESSION</td>
<td>[expression]</td>
<td>Expression to use for the sorting</td>
</tr>
<tr>
<td>Sort ascending</td>
<td>ASCENDING</td>
<td>[boolean]</td>
<td>If checked the vector layer will be sorted from small to large values.</td>
</tr>
<tr>
<td>Sort nulls first</td>
<td>NULLS_FIRST</td>
<td>[boolean]</td>
<td>If checked, Null values are placed first</td>
</tr>
<tr>
<td>Ordered</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output (sorted) vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:orderbyexpression

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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**

**Viz také:**

- Assign projection, Define Shapefile projection, Find projection

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer to reproject</td>
</tr>
<tr>
<td>Target CRS</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td>Destination coordinate reference system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EPSG:4326</td>
<td>WGS 84</td>
</tr>
<tr>
<td>Reprojected</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reprojected</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output (reprojected) vector layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:reprojectlayer

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Set style for vector layer

Sets the style of a vector layer. The style must be defined in a QML file.

No new output are created: the style is immediately assigned to the vector layer.

Viz také:

Set style for raster layer

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer you want to set the style for</td>
</tr>
<tr>
<td>Style file</td>
<td>STYLE</td>
<td>[file]</td>
<td>qml file of the style</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[same as input]</td>
<td>The input vector layer with the new style</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:setstyleforvectorlayer

```
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Split using values in the field</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for splitting</td>
</tr>
<tr>
<td>Split value using character</td>
<td>CHAR</td>
<td>[string]</td>
<td>Character to use for splitting</td>
</tr>
<tr>
<td>Use regular expression separator</td>
<td>REGEX</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:splitfeaturesbycharacter

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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

Viz také:

Merge vector layers

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Unique ID field</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for splitting</td>
</tr>
<tr>
<td>Output directory</td>
<td>OUTPUT</td>
<td>[folder]</td>
<td>Specify the directory for the output layers. One of: • Save to a Temporary Directory • Save to Directory... The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output directory</td>
<td>OUTPUT</td>
<td>[folder]</td>
<td>The directory for the output layers</td>
</tr>
<tr>
<td>Output layers</td>
<td>OUTPUT_LAYERS</td>
<td>[same as input]</td>
<td>The output vector layers resulting from the split.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:splitvectorlayer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Truncate table

Truncates a layer, by deleting all features from within the layer.

Varování: This algorithm modifies the layer in place, and deleted features cannot be restored!

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truncated layer</td>
<td>OUTPUT</td>
<td>[folder]</td>
<td>The truncated (empty) layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:truncatetable

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
23.1.15 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 \[Geometry Tools\]

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Calculate using</td>
<td>CALC_METHOD</td>
<td>[enumeration]</td>
<td>Calculation parameters to use for the geometric properties. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Layer CRS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Project CRS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Ellipsoidal</td>
</tr>
<tr>
<td>Added geom info</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output (input copy with geometry) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added geom info</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Copy of the input vector layer with the addition of the geometry fields</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:exportaddgeometrycolumns

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

**Viz také:**

*Collect geometries, Dissolve*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Group by expression</td>
<td>GROUP_BY</td>
<td>[tablefield: any]</td>
<td>Choose the grouping field. If NULL all features will be grouped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: <code>NULL</code></td>
<td></td>
</tr>
</tbody>
</table>
### Aggregates

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| List of output layer field definitions. Example of a field definition: `{aggregate: 'sum', delimiter: ';', input: 'Sarea', length: 10, name: 'totarea', precision: 0, type: 6}` | AGGREGATES | [list] | 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. Default: , **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 — Integer • 4 — Integer64 • 6 — Double • 10 — String • 14 — Date • 16 — DateTime **Length** `[number](length)` Length of the output field. **Precision** `[number](precision)` Precision of the output field. |}
## Aggregated

**Label:** Aggregated  
**Type:** OUTPUT  
**Popis:** Specify the output (aggregate) layer. One of:
- Create Temporary Layer (TEMPORARY_OUTPUT)
- Save to File...
- Save to Geopackage...
- Save to PostGIS Table...

The file encoding can also be changed here.

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Multigeometry vector layer with the aggregated values</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:aggregate`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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.

For **lines geometries**, the boundaries are their end points.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
</tbody>
</table>
| **Boundary** | OUTPUT | [vector: point, line] | Specify the output (boundary) layer. One of:
- Create Temporary Layer (TEMPORARY_OUTPUT)
- Save to File...
- Save to Geopackage...
- Save to PostGIS Table...

The file encoding can also be changed here. |
Obr. 23.26: Boundaries (black dashed line) of the source polygon layer
Obr. 23.27: Boundary layer (red points) for lines. In yellow a selected feature.

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary</td>
<td>OUTPUT</td>
<td>[vector: point, line]</td>
<td>Boundaries from the input layer (point for line, and line for polygon)</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** Qgs:boundary

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
**Bounding boxes**

Calculates the bounding box (envelope) of each feature in an input layer. Polygon and line geometries are supported.

Obr. 23.28: Black lines represent the bounding boxes of each polygon feature

*Allows* features *in-place modification*

**Viz také:**

*Minimum bounding geometry*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>INPUT [vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Bounds</td>
<td>OUTPUT</td>
<td>OUTPUT [vector: polygon]</td>
<td>Specify the output (bounding box) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounds</td>
<td>OUTPUT</td>
<td>OUTPUT [vector: polygon]</td>
<td>Bounding boxes of input layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:boundingboxes

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Buffer

Computes a buffer area for all the features in an input layer, using a fixed 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).

![Obr. 23.29: Buffer (in yellow) of points, line and polygon](image)

allows features in-place modification

**Default menu:** Vector ➔ Geoprocessing Tools

**Viz také:**

Variable distance buffer, Multi-ring buffer (constant distance), Variable width buffer (by M value)
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td><strong>Distance</strong></td>
<td>DISTANCE</td>
<td>[number]</td>
<td>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).</td>
</tr>
<tr>
<td><strong>Segments</strong></td>
<td>SEGMENTS</td>
<td>[number]</td>
<td>Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.</td>
</tr>
<tr>
<td><strong>End cap style</strong></td>
<td>END_CAP_STYLE</td>
<td>[enumeration]</td>
<td>Controls how line endings are handled in the buffer. One of: • 0 — Round • 1 — Flat • 2 — Square</td>
</tr>
<tr>
<td><strong>Join style</strong></td>
<td>JOIN_STYLE</td>
<td>[enumeration]</td>
<td>Specifies whether round, miter or beveled joins should be used when offsetting corners in a line. Options are: • 0 — Round • 1 — Miter • 2 — Bevel</td>
</tr>
<tr>
<td><strong>Miter limit</strong></td>
<td>MITER_LIMIT</td>
<td>[number]</td>
<td>Controls the maximum distance from the offset curve to use when creating a mitered join (only applicable for miter join styles). Minimum: 1.</td>
</tr>
<tr>
<td><strong>Dissolve result</strong></td>
<td>DISSOLVE</td>
<td>[boolean]</td>
<td>Dissolve the final buffer. If True (checked), overlapping buffers will be dissolved (combined) into a new feature.</td>
</tr>
<tr>
<td><strong>Buffered</strong></td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output (buffer) layer. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Save to File... • Save to Geopackage... • Save to PostGIS Table... The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffered</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Output (buffer) polygon layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:buffer

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Obr. 23.32: The red stars represent the centroids of the features of the input layer.
Default menu: Vector Geometry Tools

Viz také:

Point on Surface

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Create centroid for each part</td>
<td>ALL_PARTS</td>
<td>[boolean ]</td>
<td>If True (checked), a centroid will be created for each part of the geometry</td>
</tr>
<tr>
<td>Centroids</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output (centroid) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centroids</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Output point vector layer (centroids)</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:centroids

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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):

Default menu: Vector Geometry Tools
Obr. 23.33: Left: the input layer. Right: the valid layer (green), the invalid layer (orange)

Viz také:
*Fix geometries* and the core plugin *Geometry Checker Plugin*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT_LAYER</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Method</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>Method to use to check validity. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 2</td>
<td>• 0: The one selected in digitizing settings</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: QGIS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2: GEOS</td>
</tr>
<tr>
<td>Ignore ring self intersection</td>
<td>IGNORE_RING_SELF_INTERSECTION</td>
<td>[boolean]</td>
<td>Ignore self intersecting rings when checking for validity.</td>
</tr>
<tr>
<td>Valid output</td>
<td>VALID_OUTPUT</td>
<td>[same as input]</td>
<td>Specify the vector layer to contain a copy of the valid features of the source layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Skip output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Invalid output</td>
<td>INVALID_OUTPUT</td>
<td>[same as input]</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Skip output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Tabulka 23.70 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error output</strong></td>
<td>ERROR_OUTPUT</td>
<td>[vector: point]</td>
<td>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) • Save to File… • Save to Geopackage… • Save to PostGIS Table… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count of errors</strong></td>
<td>ERROR_COUNT</td>
<td>[number]</td>
<td>The number of geometries that caused errors.</td>
</tr>
<tr>
<td><strong>Error output</strong></td>
<td>ERROR_OUTPUT</td>
<td>[vector: point]</td>
<td>Point layer of the exact position of the validity problems detected with the message field describing the error(s) found.</td>
</tr>
<tr>
<td><strong>Count of invalid features</strong></td>
<td>INVALID_COUNT</td>
<td>[number]</td>
<td>The number of invalid geometries.</td>
</tr>
<tr>
<td><strong>Invalid output</strong></td>
<td>INVALID_OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer containing copy of the invalid features of the source layer with the field _errors listing the summary of the error(s) found.</td>
</tr>
<tr>
<td><strong>Count of valid features</strong></td>
<td>VALID_COUNT</td>
<td>[number]</td>
<td>The number of valid geometries.</td>
</tr>
<tr>
<td><strong>Valid output</strong></td>
<td>VALID_OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer containing a copy of the valid features of the source layer.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:checkvalidity`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### Types of error messages and their meanings

---

804 Kapitola23. Processing providers and algorithms
Tabulka 23.72: If the GEOS method is used the following error messages can occur:

<table>
<thead>
<tr>
<th>Error message</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeated point</td>
<td>This error happens when a given vertex is repeated.</td>
<td><img src="image1.png" alt="Example" /></td>
</tr>
<tr>
<td>Ring self-intersection</td>
<td>This error happens when a geometry touches itself and generates a ring.</td>
<td><img src="image2.png" alt="Example" /></td>
</tr>
<tr>
<td>Self-intersection</td>
<td>This error happens when a geometry touches itself.</td>
<td><img src="image3.png" alt="Example" /></td>
</tr>
<tr>
<td>Topology validation error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole lies outside shell</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Holes are nested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interior is disconnected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nested shells</td>
<td>This error happens when a polygon geometry is on top of another polygon geometry.</td>
<td><img src="image4.png" alt="Example" /></td>
</tr>
</tbody>
</table>
### Error message

<table>
<thead>
<tr>
<th>Error message</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate rings</td>
<td>This error happens when two rings (exterior or interior) of a polygon geometry are identical</td>
<td></td>
</tr>
<tr>
<td>Too few points in geometry component</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invalid coordinate</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Ring is not closed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabulka 23.73: If the QGIS method is used the following error messages can occur:

<table>
<thead>
<tr>
<th>Error message</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment <code>%1</code> of ring <code>%2</code> of polygon <code>%3</code> intersects segment <code>%4</code> of ring <code>%5</code> of polygon <code>%6</code> at <code>%7</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ring <code>%1</code> with less than four points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ring <code>%1</code> not closed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line <code>%1</code> with less than two points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line <code>%1</code> contains <code>%n</code> duplicate node(s) at <code>%2</code></td>
<td>This error happens when consecutive points on a line have the same coordinates.</td>
<td></td>
</tr>
<tr>
<td>Segments <code>%1</code> and <code>%2</code> of line <code>%3</code> intersect at <code>%4</code></td>
<td>This error happens when a line self intersects (two segments of the line intersect each other).</td>
<td></td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
**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`

**Viz také:**

`Aggregate, Promote to multipart, Dissolve`

---

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Unique ID fields</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Choose one or more attributes to collect the geometries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[list]</td>
<td></td>
</tr>
<tr>
<td>Collected</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with collected geometries</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collected</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer for the collected geometries. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:collect

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### Concave hull (alpha shapes)

Computes the concave hull of the features in an input point layer.

![Obr. 23.34: Concave hulls with different thresholds (0.3, 0.6, 0.9)](image)

**Viz také:**

*Convex hull, Concave hull (k-nearest neighbor)*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input point layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Threshold</td>
<td>ALPHA</td>
<td>[number]</td>
<td>Number from 0 (maximum concave hull) to 1 (convex hull).</td>
</tr>
<tr>
<td></td>
<td>Default: 0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allow holes</td>
<td>HOLES</td>
<td>[boolean]</td>
<td>Choose whether to allow holes in the final concave hull</td>
</tr>
<tr>
<td></td>
<td>Default: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Split multipart geometry into single-part geometries</td>
<td>NO_MULTIGEOMETRY</td>
<td>[boolean]</td>
<td>Check if you want to have singlepart geometries instead of multipart ones.</td>
</tr>
<tr>
<td></td>
<td>Default: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concave hull</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concave hull</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output vector layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:concavehull

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**Concave hull (k-nearest neighbor)**

Generates a concave hull polygon from a set of points. If the input layer is a line or polygon layer, it will use the vertices.

The number of neighbors to consider determines the concaveness of the output polygon. A lower number will result in a concave hull that follows the points very closely, while a higher number will have a smoother shape. The minimum number of neighbor points to consider is 3. A value equal to or greater than the number of points will result in a convex hull.

If a field is selected, the algorithm will group the features in the input layer using unique values in that field and generate individual polygons in the output layer for each group.

**Viz také:**

Concave hull (alpha shapes)
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Number of neighboring points to consider (a lower number is more concave, a higher number is smoother)</td>
<td>RNEIGHBORS</td>
<td>[number]</td>
<td>Default: 3. Determines the concaveness of the output polygon. A small number will result in a concave hull that follows the points very closely, while a high number will make the polygon look more like the convex hull (if the number is equal to or larger than the number of features, the result will be the convex hull). Minimum value: 3.</td>
</tr>
<tr>
<td>Field</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Default: None. If specified, one concave hull polygon is generated for each unique value of the field (by selecting features using this value).</td>
</tr>
<tr>
<td>Concave hull</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Default: [Create temporary layer (TEMPORARY_OUTPUT)] Specify the output vector layer. One of: • Create Temporary Layer • Save to File… • Save to Geopackage… • Save to PostGIS Table… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concave hull</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:knearestconcavehull

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Convert geometry type

Generates a new layer based on an existing one, with a different type of geometry.

Not all conversions are possible. For instance, a line can be converted to a point, but a point cannot be converted to a line. A line can also be converted to a polygon.

Viz také:

Polygonize, Lines to polygons
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>New geometry type</td>
<td>TYPE</td>
<td>[enumeration]</td>
<td>Geometry type to apply to the output features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Centroids</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Linestrings</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Multilinestrings</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Polygons</td>
</tr>
<tr>
<td>Converted</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converted</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Output vector layer - the type depends on the parameters</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** qgis:convertgeometrytype

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

## 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.

- Allows features in-place modification

**Default menu:** Vector ➤ Geoprocessing Tools

**Viz také:**

- Minimum bounding geometry, Concave hull (alpha shapes)
Obr. 23.35: Black lines identify the convex hull for each layer feature

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Convex hull</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex hull</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output (convex hull) vector layer</td>
</tr>
</tbody>
</table>
**Python code**

**Algorithm ID:** qgis:convexhull

```python
import processing
type.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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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 (\(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\) format) into a layer which can be used for other algorithms which require a layer based input.

**Viz také:**

*Create layer from point*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent ((x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}))</td>
<td>INPUT</td>
<td>[extent]</td>
<td>Input extent</td>
</tr>
<tr>
<td>Extent</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output (extent) vector layer</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:extenttolist

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Viz také:

Create layer from extent

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>INPUT</td>
<td>[coordinates]</td>
<td>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.</td>
</tr>
</tbody>
</table>
| Point   | OUTPUT  | [vector: point] | Specify the output layer. One of:  
  • Create Temporary Layer (TEMPORARY_OUTPUT)  
  • Save to File...  
  • Save to Geopackage...  
  • Save to PostGIS Table...  
  The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output point vector layer containing the input point.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:pointtolayer

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Create wedge buffers

Creates wedge shaped buffers from input points.

Obr. 23.36: Wedge buffers

The native output from this algorithm are CurvePolygon geometries, but these may be automatically segmentized to Polygons depending on the output format.

Viz také:
Buffer, Variable width buffer (by M value), Tapered buffers

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Azimuth (degrees from North)</td>
<td>AZIMUTH</td>
<td>[number ≥ 0]</td>
<td>Angle (in degrees) as the middle value of the wedge</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
</tbody>
</table>
### Tabulka 23.75 – pokračování na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Wedge width (in degrees)</strong></td>
<td><strong>WIDTH</strong></td>
<td>[number] Default: 45.0</td>
<td>Width (in degrees) of the buffer. The wedge will extend to half of the angular width either side of the azimuth direction. Azimuth = 0°</td>
</tr>
<tr>
<td><strong>Outer radius</strong></td>
<td><strong>OUTER_RADIUS</strong></td>
<td>[number] Default: 1.0</td>
<td>The outer size (length) of the wedge: the size is meant from the source point to the edge of the wedge shape.</td>
</tr>
<tr>
<td><strong>Inner radius</strong></td>
<td><strong>INNER_RADIUS</strong></td>
<td>[number] Default: 0.0</td>
<td>Inner radius value. If 0 the wedge will begin from the source point.</td>
</tr>
<tr>
<td><strong>Buffers</strong></td>
<td><strong>OUTPUT</strong></td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffers</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output (wedge buffer) vector layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:wedgebuffers`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
Delaunay triangulation

Creates a polygon layer with the Delaunay triangulation corresponding to the input point layer.

![Delaunay triangulation on points](image)

Obr. 23.38: Delaunay triangulation on points

**Default menu:** Vector → Geometry Tools

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Delaunay triangulation</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delaunay triangulation</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output (Delaunay triangulation) vector layer</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** qgis:delaunaytriangulation

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

![Obr. 23.39: Before and after the cleaning](image)

Allows *features in-place modification*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input polygon vector layer</td>
</tr>
<tr>
<td><strong>Remove holes with area less than</strong></td>
<td>MIN_AREA</td>
<td>[number ]</td>
<td>Only holes with an area less than this threshold will be deleted. If 0.0 is added, all the holes will be deleted.</td>
</tr>
<tr>
<td><strong>Optional</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cleaned</strong></td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td>****</td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td>****</td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td>****</td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td>****</td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleaned</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output (cleaned) vector layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:deleteholes

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.

![Obr. 23.40: Red points show the vertices before and after the densify](image)

- Allows features in-place modification
- Default menu: Vector ➔ Geometry Tools
- Viz také:
  - Densify by interval
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Vertices to add</td>
<td>VERTICES</td>
<td>[number]</td>
<td>Number of vertices to add to each segment</td>
</tr>
<tr>
<td>Densified</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Densified</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output (densified) vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:densifygeometries

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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 \([0 \ 0] \rightarrow [10 \ 0]\) to be converted to \([0 \ 0] \rightarrow [2.5 \ 0] \rightarrow [5 \ 0] \rightarrow [7.5 \ 0] \rightarrow [10 \ 0]\), 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.

- Allows features in-place modification

Viz také:

Densify by count
Obr. 23.41: Densify geometry at a given interval

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Interval between vertices to add</td>
<td>INTERVAL</td>
<td>[number ]</td>
<td>Maximum distance between two consecutive vertices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1.0</td>
<td></td>
</tr>
<tr>
<td>Densified</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Densified</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output (densified) vector layer</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:densifygeometriesgivenaninterval

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**Dissolve**

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.

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.

Obr. 23.42: Dissolve the polygon layer on a common attribute

**Default menu:** Vector ➤ Geoprocessing Tools

**Viz také:**

Aggregate, Collect geometries
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Dissolve field(s)</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>[list]</td>
<td>Default: []</td>
</tr>
<tr>
<td>Dissolved</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissolved</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer with dissolved geometries</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:dissolve

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

**Viz také:**

*Set M value from raster, Set Z value*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Raster layer</td>
<td>RASTER</td>
<td>[raster]</td>
<td>Raster layer with Z values</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The raster band to take the Z values from</td>
</tr>
<tr>
<td>Value for no-data or non-intersecting vertices</td>
<td>NODATA</td>
<td>[number ≤ ]</td>
<td>Value to use in case the vertex does not intersect (a valid pixel of) the raster</td>
</tr>
<tr>
<td>Scale factor</td>
<td>SCALE</td>
<td>[number ≤ ]</td>
<td>Scaling value: the band values are multiplied by this value.</td>
</tr>
<tr>
<td>Updated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer (with Z values from the raster layer). One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer with Z values from the raster layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis: setzfromraster

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Drop M/Z values

Removes M (measure) or Z (altitude) values from input geometries.

**Viz také:**

*Set M value, Set Z value*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer with M or Z values</td>
</tr>
<tr>
<td>Drop M Values</td>
<td>DROP_M_VALUES</td>
<td>[boolean]</td>
<td>Removes the M values from the geometries</td>
</tr>
<tr>
<td>Drop Z Values</td>
<td>DROP_Z_VALUES</td>
<td>[boolean]</td>
<td>Removes the Z values from the geometries</td>
</tr>
<tr>
<td>Z/M Dropped</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Outputs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z/M Dropped</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (identical to the input layer, except that the M and/or Z dimensions have been removed from the geometries).</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:dropmzvalues

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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

**Viz také:**

Fix geometries
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input polygon vector layer</td>
</tr>
</tbody>
</table>
| Merge selection with the neighboring polygon with the | MODE           | [enumeration]          | Choose the parameter to use in order to get rid of the selected polygons:  
|                     |                | Default: None         |  
|                     |                |                       | • 0 — Largest Area                |
|                     |                |                       | • 1 — Smallest Area              |
|                     |                |                       | • 2 — Largest Common Boundary    |
| Eliminated          | OUTPUT         | [vector: polygon]     | Specify the output vector layer. One of:  
|                     |                | Default: [Create temporary layer] |  
|                     |                |                       | • Create Temporary Layer          |
|                     |                |                       | (TEMPORARY_OUTPUT)                |
|                     |                |                       | • Save to File…                  |
|                     |                |                       | • Save to Geopackage…             |
|                     |                |                       | • Save to PostGIS Table…          |
|                     |                |                       | The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

Python code

**Algorithm ID:** qgis:eliminateselectedpolygons

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

- **Allows** *features in-place modification*
- **Viz také:**
  - *Subdivide*, *Line substring*
Obr. 23.43: The original line layer and the exploded one
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Exploded</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploded</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output line vector layer with features</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>representing each segment of the input layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:explodelines

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

- Allows features in-place modification

Viz také:

*Line substring*
Obr. 23.44: The red dashes represent the initial and final extension of the original layer

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Start distance</td>
<td>START_DISTANCE</td>
<td>[number]</td>
<td>Distance by which to extend the first segment of the line (starting point)</td>
</tr>
<tr>
<td>End distance</td>
<td>END_DISTANCE</td>
<td>[number]</td>
<td>Distance by which to extend the last segment of the line (ending point)</td>
</tr>
<tr>
<td>Extended</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output (extended) line vector layer.</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:extendlines

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

*Extract Z values*, *Set M value*, *Drop M/Z values*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Summaries to calculate</td>
<td>SUMMARIES</td>
<td>[enumeration]</td>
<td>Statistics on the M values of a geometry. One or more of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [0]</td>
<td>• 0 — First</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Last</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Sum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Mean</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Median</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — St.dev (pop)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Minimum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — Maximum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — Minority</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — Majority</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — Variety</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — Q1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — Q3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — IQR</td>
</tr>
<tr>
<td>Output column prefix</td>
<td>COLUMN_PREFIX</td>
<td>[string]</td>
<td>The prefix for the output (M) column</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: .m_</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td>Extracted</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td></td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (with M values)</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:extractmvalues

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

Extract vertices, Filter vertices by M value, Filter vertices by Z value

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Vertex indices</td>
<td>VERTICES</td>
<td>[string]</td>
<td>Comma-separated string of the indices of the vertices to extract.</td>
</tr>
<tr>
<td>Vertices</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertices</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output (point) vector layer containing the specified vertices from the input layer geometries.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:extractspecificvertices

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Obr. 23.45: Vertices extracted for line and polygon layer

Default menu: Vector Geometry Tools

Viz také:

Extract specific vertices, Filter vertices by M value, Filter vertices by Z value
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Vertices</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertices</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output (point) vector layer containing the vertices from the input layer geometries.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:extractvertices

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

Extract M values, Set Z value, Drop M/Z values

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
</tbody>
</table>
Tabulka 23.78 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summaries to calculate</td>
<td>SUMMARIES</td>
<td>[enumeration]</td>
<td>Statistics on the Z values of a geometry. One or more of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [0]</td>
<td>• 0 — First</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Last</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Sum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Mean</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Median</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — St.dev (pop)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Minimum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — Maximum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — Minority</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — Majority</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — Variety</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — Q1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — Q3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — IQR</td>
</tr>
<tr>
<td>Output column prefix</td>
<td>COLUMN_PREFIX</td>
<td>[string]</td>
<td>The prefix for the output (Z) column</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: <em>z</em></td>
<td></td>
</tr>
<tr>
<td>Extracted</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (with Z values)</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:extractzvalues

```python
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 NANEs and values. See *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

Obr. 23.46: The red line represents the black line with only vertices whose M value is <=10.

Poznámka: Depending on the input geometry attributes and the filters used, the resultant geometries created by this algorithm may no longer be valid.

Viz také:
Filter vertices by Z value, Extract vertices, Extract specific vertices

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer to remove vertices from</td>
</tr>
<tr>
<td>Minimum</td>
<td>MIN</td>
<td>[number]</td>
<td>Minimum of M values allowed</td>
</tr>
<tr>
<td>Maximum</td>
<td>MAX</td>
<td>[number]</td>
<td>Maximum of M values allowed</td>
</tr>
<tr>
<td>Filtered</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtered</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer of features with only the filtered vertices.</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:filterverticesbym

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.

Obr. 23.47: The red line represents the black line with only vertices whose Z value is <=10.

---

**Poznámka:** 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.

---

**Viz také:**

Filter vertices by M value, Extract vertices, Extract specific vertices

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer to remove vertices from</td>
</tr>
<tr>
<td>Minimum</td>
<td>MIN</td>
<td>[number ]</td>
<td>Minimum of Z values allowed</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: Not set</td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td>MAX</td>
<td>[number ]</td>
<td>Maximum of Z values allowed</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: Not set</td>
<td></td>
</tr>
<tr>
<td>Filtered</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtered</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer of features with only the filtered vertices.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:filterverticesbyz

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

**Poznámka:** M values will be dropped from the output.

- **Allowed features in-place modification**
- **Viz také:**
  - *Check validity*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Fixed geometries</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed geometries</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer with fixed geometries.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis:fixgeometries`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Reoriented</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
Python code

Algorithm ID: qgis:forcerhr

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Split</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output line vector layer split at the antimeridian.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:antimeridiansplit

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Output geometry type</td>
<td>OUTPUT_GEOMETRY [enumeration]</td>
<td>Default: 0</td>
<td>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 — Polygon • 1 — Line • 2 — Point</td>
</tr>
<tr>
<td>Output geometry has z values</td>
<td>WITH_Z [boolean]</td>
<td>Default: False</td>
<td>Choose if the output geometry should include the Z dimension</td>
</tr>
<tr>
<td>Output geometry has m values</td>
<td>WITH_M [boolean]</td>
<td>Default: False</td>
<td>Choose if the output geometry should include the M dimension</td>
</tr>
<tr>
<td>Geometry expression</td>
<td>EXPRESSION [expression]</td>
<td>Default: '$geometry'</td>
<td>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.</td>
</tr>
<tr>
<td>Modified geometry</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Specify the output vector layer. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Save to File… • Save to Geopackage… • Save to PostGIS Table… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified geometry</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>The output vector layer</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:geometrybyexpression

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

![Obr. 23.48: Interpolated point at 500m of the beginning of the line](image)

Viz také:

**Points along geometry**

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Distance from the beginning of the line</td>
</tr>
<tr>
<td>Interpolated points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: <a href="TEMPORARY_OUTPUT">Create temporary layer</a></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Viz také:

**Points along geometry**
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output point vector layer with features at a set distance along the line or polygon boundary</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:interpolatepoint

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

Obr. 23.49: Clockwise from top left: original multipart feature, one, two and three biggest parts kept
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polygons</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input polygon vector layer</td>
</tr>
<tr>
<td>Parts to keep</td>
<td>PARTS</td>
<td>[number]</td>
<td>Number of parts to keep. If 1, only the biggest part of the feature will be kept.</td>
</tr>
<tr>
<td>Parts</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output polygon vector layer with the N biggest parts of each feature</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:keepnbiggestparts

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Allow features in-place modification

Viz také:

Extend lines
Obr. 23.50: Substring line with starting distance set at 0 meters and the ending distance at 250 meters.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Start distance</td>
<td>START_DISTANCE</td>
<td>[number ]</td>
<td>Distance along the input line to the start point of the output feature</td>
</tr>
<tr>
<td>End distance</td>
<td>END_DISTANCE</td>
<td>[number ]</td>
<td>Distance along the input line to the end point of the output feature</td>
</tr>
<tr>
<td>Substring</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substring</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output line vector layer.</td>
</tr>
</tbody>
</table>

### Python code

Algorithm ID: qgis:linesubstring

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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 from of the input line layer.

Default menu: Vector Geometry Tools

Viz také:
Polygons to lines, Polygonize

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Polygons</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

Python code

Algorithm ID: qgis:linestopolygons

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Merged</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merged</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output (merged) line vector layer.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:mergelines

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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

Viz také:

Minimum enclosing circles

846 Kapitola 23. Processing providers and algorithms
Obr. 23.51: Clockwise from top left: envelope, oriented rectangle, circle, convex hull
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td><strong>Field</strong></td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Geometry type</strong></td>
<td>TYPE</td>
<td>[enumeration]</td>
<td>Enclosing geometry types. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Envelope (Bounding Box)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Minimum Oriented Rectangle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Minimum Enclosing Circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Convex Hull</td>
</tr>
<tr>
<td><strong>Bounding geometry</strong></td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bounding geometry</strong></td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output (bounding) polygon vector layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis:minimumboundinggeometry`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Minimum enclosing circles

Calculates the minimum enclosing circles of the features in the input layer.

- Allows features in-place modification

Viz také:  

*Minimum bounding geometry*
Obr. 23.52: Enclosing circles for each feature
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Number of segment in circles</td>
<td>SEGMENTS</td>
<td>[number]</td>
<td>The number of segment used to approximate a circle. Minimum 8, maximum 100000.</td>
</tr>
<tr>
<td>Minimum enclosing circles</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

Python code

Algorithm ID: qgis:minimumenclosingcircle

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

![Image of multi-ring buffer]

Obr. 23.53: Multi-ring buffer for a line, point and polygon layer

 Allows features in-place modification

Viz také:
Buffer, Variable distance buffer, Rectangles, ovals, diamonds (fixed), Rectangles, ovals, diamonds (variable), Single sided buffer

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Number of rings</td>
<td>RINGS</td>
<td>[number ]</td>
<td>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).</td>
</tr>
<tr>
<td>Distance between rings</td>
<td>DISTANCE</td>
<td>[number ]</td>
<td>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).</td>
</tr>
<tr>
<td>Multi-ring buffer (constant distance)</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output polygon vector layer. One of: • Create Temporary Layer (TEMPORARY_OUTPUT) • Save to File… • Save to Geopackage… • Save to PostGIS Table… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

Python code

Algorithm ID: qgis:multiringconstantbuffer

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
**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.

![Example](image)

Obr. 23.54: Left the multipart source layer and right the single part output result

*Allows features in-place modification*

**Default menu:** Vector ➤ Geometry Tools

Viz také:

*Collect geometries, Promote to multipart*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Single parts</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single parts</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:multiparttosingleparts

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

![Image](23.55: In blue the source layer, in red the offset one)

Allows features in-place modification

Viz také: Array of offset (parallel) lines, Translate

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>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).</td>
</tr>
<tr>
<td>Segments</td>
<td>SEGMENTS</td>
<td>[number]</td>
<td>Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.</td>
</tr>
<tr>
<td>Join style</td>
<td>JOIN_STYLE</td>
<td>[enumeration]</td>
<td>Specifies whether round, miter or beveled joins should be used when offsetting beveled lines in a line. Options are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Round</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Miter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Bevel</td>
</tr>
<tr>
<td>Miter limit</td>
<td>MITER_LIMIT</td>
<td>[number]</td>
<td>Controls the maximum distance from the offset curve to use when creating a mitered join (only applicable for miter join styles). Minimum: 1.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.83 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output (offset) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
<td></td>
</tr>
</tbody>
</table>

Outsput

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Output (offset) line layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:offsetline

```python
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 `Using processing algorithms from the console` for details on how to run processing algorithms from the Python console.

Oriented minimum bounding box

Calculates the minimum area rotated rectangle for each feature in the input layer.

![ Allows features in-place modification ]

Viz také:

Minimum bounding geometry

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td>Bounding boxes</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
<td></td>
</tr>
</tbody>
</table>
Obr. 23.56: Oriented minimum bounding box
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

Python code

**Algorithm ID:** qgis:orientedminimumboundingbox

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Obr. 23.57: In blue the source layer and in the red orthogonalized result

Allows features in-place modification
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td><strong>Maximum angle tolerance</strong></td>
<td>ANGLE_TOLERANCE</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Maximum algorithm iterations</strong></td>
<td>MAX_ITERATIONS</td>
<td>[number]</td>
<td>Setting a larger number for the maximum number of iterations will result in a more orthogonal geometry at the cost of extra processing time.</td>
</tr>
<tr>
<td>Orthogonalized</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orthogonalized</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output polygon vector layer with adjusted angles.</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** qgis:orthogonalize

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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

Viz také:

Centroids

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Create point on surface</td>
<td>ANGLE_TOLERANCE</td>
<td>[boolean]</td>
<td>If checked, a point will be created for each part of the geometry.</td>
</tr>
<tr>
<td>for each part</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output point vector layer. One</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output point vector layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:pointonsurface

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

Interpolate point on line
Obr. 23.58: Points created along the source line layer
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Distance between two consecutive points along the line</td>
</tr>
<tr>
<td></td>
<td>START_OFFSET</td>
<td>[number]</td>
<td>Distance from the beginning of the input line, representing the position of the first point.</td>
</tr>
<tr>
<td></td>
<td>END_OFFSET</td>
<td>[number]</td>
<td>Distance from the end of the input line, representing the position beyond which no point feature should be created.</td>
</tr>
<tr>
<td>Interpolated points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated points</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Point vector layer with features placed along lines or polygon boundaries of the input layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:pointsalonglines

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Minimum distance to other points</td>
<td>PROXIMITY</td>
<td>[number]</td>
<td>Default: 1.0</td>
</tr>
<tr>
<td>Displacement distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Default: 1.0</td>
</tr>
<tr>
<td>Horizontal distribution for two point case</td>
<td>HORIZONTAL</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Displaced</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Default: [Create temporary layer]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displaced</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Output point vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:pointsdisplacement

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.
Obr. 23.59: Pole of inaccessibility

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Tolerance</td>
<td>TOLERANCE</td>
<td>[number]</td>
<td>Set the tolerance for the calculation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1.0</td>
<td></td>
</tr>
<tr>
<td>Point</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output polygon vector layer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output point vector layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:poleofinaccessibility

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
Polygonize

Creates a polygon layer whose features boundaries are generated from a line layer of closed features.

Obr. 23.60: The yellow polygons generated from the closed lines

**Poznámka:** The line layer must have closed shapes in order to be transformed into a polygon.

**Viz také:**

*Polygons to lines*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Keep table structure of line layer</td>
<td>KEEP_FIELDS</td>
<td>[boolean]</td>
<td>Check to copy the original attributes of the input layer</td>
</tr>
<tr>
<td>Default: False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polygons from lines</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output polygon vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Create temporary layer]</td>
<td>Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polygons from lines</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output polygon vector layer from lines</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:polygonsize

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Polygons to lines

Takes a polygon layer and creates a line layer, with lines representing the boundaries of the polygons in the input layer.

Obr. 23.61: Black lines as the result of the algorithm

Default menu: Vector Geometry Tools

Viz také:

Polygonize

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input polygon vector layer</td>
</tr>
<tr>
<td>Lines</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGSI Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output line vector layer from polygons</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:polygontolines

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Project points (Cartesian)

Projects point geometries by a specified distance and bearing (azimuth).

☑️ Allows *features in-place modification*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Bearing (degrees from North)</td>
<td>BEARING</td>
<td>[number] Default: 0.0</td>
<td>Clockwise angle starting from North, in degree (°) unit</td>
</tr>
<tr>
<td>Distance</td>
<td>DISTANCE</td>
<td>[number] Default: 1.0</td>
<td>Distance to offset geometries, in layer units</td>
</tr>
</tbody>
</table>
| Projected              | OUTPUT       | [vector: point] Default: [Create temporary layer] | Specify the output point vector layer. One of:
  • Create Temporary Layer (TEMPORARY_OUTPUT)
  • Save to File...
  • Save to Geopackage...
  • Save to PostGIS Table...
  The file encoding can also be changed here. |
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projected</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output (projected) point vector layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis:projectpointcartesian`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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*

Viz také:

*Aggregate, Collect geometries*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
</tbody>
</table>
| Multiparts | OUTPUT      | [same as input]          | Specify the output multipart vector layer. One of:
|            |             |                          | - Create Temporary Layer (TEMPORARY_OUTPUT) |
|            |             |                          | - Save to File…                           |
|            |             |                          | - Save to Geopackage…                      |
|            |             |                          | - Save to PostGIS Table…                   |
|            |             |                          | The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiparts</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output multipart vector layer</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis: promotetomulti

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Rectangles, ovals, diamonds (fixed)

Creates a buffer area for all the features in an input layer with different shape choice.

Parameters can vary depending on the shape chosen.

![Different buffer shapes](image)

Obr. 23.62: Different buffer shapes

Viz také:

Rectangles, ovals, diamonds (variable)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
</tbody>
</table>
| Buffer shape | SHAPE | [enumeration] | The shape to use. One of:  
• 0 — Rectangles  
• 1 — Ovals  
• 2 — Diamonds |
| Width | WIDTH | [number] | Width of the buffer shape |
| Default: 1.0 |
| Height | HEIGHT | [number] | Height of the buffer shape |
| Default: 1.0 |
| Rotation | ROTATION | [number] | Rotation of the buffer shape |
| Default: None |
| Number of segments | SEGMENTS | [number] | Number of segments for a full circle (Ovals shape) |
| Default: 36 |
Tabulka 23.88 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Output | OUTPUT | [vector: polygon] | Specify the output vector layer. One of:  
| Default: | Create Temporary Layer (TEMPORARY_OUTPUT)  
| | Save to File…  
| | Save to Geopackage…  
| | Save to PostGIS Table…  
| | The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output vector layer (with the buffer shapes)</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:rectanglesovalsdiamonds

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Rectangles, ovals, diamonds (variable)**

Creates a buffer area for all the features in an input layer with different shape choice. Buffer shape parameters are specified through attribute of the input layer.

![Different buffer shapes with different parameters](image)

Obr. 23.63: Different buffer shapes with different parameters

**Viz také:**

*Rectangles, ovals, diamonds (fixed)*
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Buffer shape</td>
<td>SHAPE</td>
<td>[enumeration]</td>
<td>The shape to use. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Rectangles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Ovals</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Diamonds</td>
</tr>
<tr>
<td>Width field</td>
<td>WIDTH</td>
<td>[tablefield: numeric]</td>
<td>Width of the buffer shape</td>
</tr>
<tr>
<td>Height field</td>
<td>HEIGHT</td>
<td>[tablefield: numeric]</td>
<td>Height of the buffer shape</td>
</tr>
<tr>
<td>Rotation field</td>
<td>ROTATION</td>
<td>[tablefield: numeric]</td>
<td>Rotation of the buffer shape</td>
</tr>
<tr>
<td>Number of segment</td>
<td>SEGMENTS</td>
<td>[number]</td>
<td>Number of segments for a full circle (Ovals shape)</td>
</tr>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output vector layer (with the buffer shapes)</td>
</tr>
</tbody>
</table>

### 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.

_Poznámka:_ 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.

_Access feature in-place modification_

_Viz také:_

Extract vertices, Extract specific vertices, Delete duplicate geometries
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Tolerance</td>
<td>TOLERANCE</td>
<td>[number]</td>
<td>Vertices closer than the specified distance are considered duplicates</td>
</tr>
<tr>
<td>Use Z value</td>
<td>USE_Z_VALUE</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Cleaned</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleaned</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (without duplicate vertices)</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:removeduplicatevertices

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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.

**Viz také:**

*Delete duplicate geometries*
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer (with non-NUL geometries)</td>
</tr>
<tr>
<td>Non null geometries</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer for the non-NUL geometries. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Null geometries</td>
<td>NULL_OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer for the NULL geometries. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null geometries</td>
<td>NULL_OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (only NULL geometries)</td>
</tr>
<tr>
<td>Non null geometries</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (without NULL geometries)</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** qgis:removenullgeometries

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

## Reverse line direction

Inverts the direction of a line layer.

- Allows *features in-place modification*
Obr. 23.64: Before and after the direction inversion

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Reversed</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reversed</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output line vector layer (with reversed lines)</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:reverselinedirection

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
**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*

**Viz také:**

*Translate, Swap X and Y coordinates*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Rotation (degrees clockwise)</td>
<td>ANGLE</td>
<td>[number]</td>
<td>Angle of the rotation in degrees</td>
</tr>
<tr>
<td>Default: 0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation anchor point (x, y)</td>
<td>ANCHOR</td>
<td>[point]</td>
<td>X,Y coordinates of the point to rotate the features around. If not set the rotation occurs around each feature’s centroid.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer (with rotated geometries). One of:</td>
</tr>
<tr>
<td>Default: [Create temporary layer]</td>
<td></td>
<td></td>
<td>* Create Temporary Layer (TEMPORARY_OUTPUT)*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer with rotated geometries</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:rotatefeatures

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
Segmentize by maximum angle

Segmentizes a geometry by converting curved sections to linear sections.

The segmentation 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.

Viz také:

Segmentize by maximum distance, Simplify, Smooth

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Maximum angle between vertices (degrees)</td>
<td>ANGLE</td>
<td>[number ] Default: 5.0</td>
<td>Maximum allowed radius angle between vertices on the straightened geometry</td>
</tr>
<tr>
<td>Segmentized</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer (with segmentized geometries). One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmentized</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer with segmentized geometries</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:segmentizebymaxangle

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Viz také:

*Segmentize by maximum angle, Simplify, Smooth*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Maximum offset distance</td>
<td>DISTANCE</td>
<td>[number ]</td>
<td>Default: 1.0</td>
</tr>
<tr>
<td>Segmentized</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer (with segmentized geometries). One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td>Outputs</td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmentized</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer with segmentized geometries</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:segmentizebymaxdistance

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Tip: Use the Identify Features button to check the added M value: the results are available in the Identify Results dialog.

Viz také:

Set M value from raster, Set Z value, Drop M/Z values

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>M Value</td>
<td>M_VALUE</td>
<td>[number ]</td>
<td>M value to assign to the feature geometries</td>
</tr>
<tr>
<td>M Added</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>M Added</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (with M values as-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>signed to the geometries)</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:setmvalue

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Viz také:

*Drape (set Z value from raster), Set M value*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Raster layer</td>
<td>RASTER</td>
<td>[raster]</td>
<td>Raster layer with M values</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The raster band from which the M values are taken</td>
</tr>
<tr>
<td>Value for no-data or non-intersecting vertices</td>
<td>NODATA</td>
<td>[number]</td>
<td>Value to use in case the vertex does not intersect (a valid pixel of) the raster</td>
</tr>
<tr>
<td>Scale factor</td>
<td>SCALE</td>
<td>[number]</td>
<td>Scaling value: the band values are multiplied by this value.</td>
</tr>
<tr>
<td>Updated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer (with updated M values). One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (with updated M values)</td>
</tr>
</tbody>
</table>

**Python code**

_Algorithm ID:_ qgis:setmfromraster

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

**Tip:** Use the `Identify Features` button to check the added Z value: the results are available in the `Identify Results` dialog.

Viz také:

*Drape (set Z value from raster), Set M value, Drop M/Z values*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Z Value</td>
<td>Z_VALUE</td>
<td>[number ]</td>
<td>Z value to assign to the feature geometries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Z Added</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z Added</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer (with Z values as-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>assigned)</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:setzvalue

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

☑ Allows *features in-place modification*

Default menu: Vector ☑️ *Geometry Tools*

**Viz také:**

*Smooth, Densify by count, Densify by interval*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Simplification method</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>Simplification method. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Distance (Douglas-Peucker)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Snap to grid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Area (Visvalingam)</td>
</tr>
<tr>
<td>Tolerance</td>
<td>TOLERANCE</td>
<td>[number ☐️]</td>
<td>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.</td>
</tr>
<tr>
<td>Simplified</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output (simplified) vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output (simplified) vector layer</td>
</tr>
</tbody>
</table>
Obr. 23.65: Clockwise from top left: source layer and increasing simplification tolerances.
Python code

Algorithm ID: qgis:simplifygeometries

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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.

Obr. 23.66: Left versus right side buffer on the same vector line layer

Viz také:

*Buffer*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Buffer distance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 10.0</td>
<td></td>
</tr>
<tr>
<td>Side</td>
<td>SIDE</td>
<td>[enumeration]</td>
<td>Which side to create the buffer on. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 – Left</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 – Right</td>
</tr>
<tr>
<td>Segments</td>
<td>SEGMENTS</td>
<td>[number]</td>
<td>Controls the number of line segments to use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 8</td>
<td>to approximate a quarter circle when crea-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ting rounded offsets.</td>
</tr>
</tbody>
</table>
Join style

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| JOIN_STYLE | JOIN_STYLE  | [enumeration]    | Specifies whether round, miter or beveled joins should be used when offsetting corners in a line. Options are:  
|            |             |                 | • 0 — Round  
|            |             |                 | • 1 — Miter  
|            |             |                 | • 2 — Bevel |

Miter limit

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MITER_LIMIT</td>
<td>MITER_LIMIT</td>
<td>[number]</td>
<td>Controls the maximum distance from the offset curve to use when creating a mitered join (only applicable for miter join styles). Minimum: 1.0</td>
</tr>
</tbody>
</table>

Buffer

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| OUTPUT  | OUTPUT| [vector: polygon]| Specify the output (buffer) layer. One of:  
|         |       |                 | • Create Temporary Layer (TEMPORARY_OUTPUT)  
|         |       |                 | • Save to File...  
|         |       |                 | • Save to Geopackage...  
|         |       |                 | • Save to PostGIS Table...  
|         |       |                 | The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Output (buffer) polygon layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:singlesidedbuffer

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Smooth

Smoothes 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.

The offset parameter controls how „tightly“ the smoothed geometries follow the original geometries. Smaller values result in a tighter fit, and larger values will create a looser fit.

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.

允許 features in-place modification

Viz také:

Simplify, Densify by count, Densify by interval
Obr. 23.67: Increasing number of iterations causes smoother geometries

Obr. 23.68: Blue: the input layer. Offset 0.25 gives the red line, while offset 0.50 gives the green line.
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Input line or polygon vector layer</td>
</tr>
<tr>
<td>Iterations</td>
<td>ITERATIONS</td>
<td>[number] Default: 1</td>
<td>Increasing the number of iterations will give smoother geometries (and more vertices).</td>
</tr>
<tr>
<td>Offset</td>
<td>OFFSET</td>
<td>[number] Default: 0.25</td>
<td>Increasing values will move the smoothed lines / boundaries further away from the input lines / boundaries.</td>
</tr>
<tr>
<td>Maximum node angle to smooth</td>
<td>MAX_ANGLE</td>
<td>[number] Default: 180.0</td>
<td>Every node below this value will be smoothed</td>
</tr>
<tr>
<td>Smoothed</td>
<td>OUTPUT</td>
<td>[same as input] Default: Create temporary layer</td>
<td>Specify the output (smoothed) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smoothed</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output (smoothed) vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:smoothgeometry

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

Snap points to grid
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Reference layer</td>
<td>REFERENCE_LAYER</td>
<td>[vector: any]</td>
<td>Vector layer to snap to</td>
</tr>
<tr>
<td>Tolerance</td>
<td>TOLERANCE</td>
<td>[number]</td>
<td>Default: 10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control how close input vertices need to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>be to the reference layer geometries befo-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>re they are snapped.</td>
</tr>
<tr>
<td>Behavior*</td>
<td>BEHAVIOR</td>
<td>[enumeration]</td>
<td>Default: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Snapping can be done to an existing node</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>or a segment (its closest point to the vert-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ex to move). Available snapping options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Prefer aligning nodes, insert extra</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vertices where required</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Prefer closest point, insert extra</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vertices where required</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Prefer aligning nodes, don’t insert</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>new vertices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Prefer closest point, don’t insert</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>new vertices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Move end points only, prefer aligning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Move end points only, prefer closest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>point</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Snap end points to end points only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Snap to anchor nodes (single layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>only)</td>
</tr>
<tr>
<td>Snapped geometry</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output (snapped) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snapped geometry</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output (snapped) vector layer</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID**: `qgis:snapgeometries`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

**Poznámka:** Snapping to grid may generate an invalid geometry in some corner cases.

✅ Allows features in-place modification

Viz také:

*Snap geometries to layer*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>X Grid Spacing</td>
<td>HSPACING</td>
<td>[number ≤ ]</td>
<td>Grid spacing on the X axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1.0</td>
<td></td>
</tr>
<tr>
<td>Y Grid Spacing</td>
<td>VSPACING</td>
<td>[number ≤ ]</td>
<td>Grid spacing on the Y axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1.0</td>
<td></td>
</tr>
<tr>
<td>Z Grid Spacing</td>
<td>ZSPACING</td>
<td>[number ≤ ]</td>
<td>Grid spacing on the Z axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>M Grid Spacing</td>
<td>MSPACING</td>
<td>[number ≤ ]</td>
<td>Grid spacing on the M axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Snapped</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output (snapped) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>temporary layer]</td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snapped</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output (snapped) vector layer</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: qgis:snappointstogrid

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>The input line vector layer</td>
</tr>
<tr>
<td>Maximum line length</td>
<td>LENGTH</td>
<td>[number]</td>
<td>The maximum length of a line in the output.</td>
</tr>
<tr>
<td>Split</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The new line vector layer - the length of the feature geometries is less than or equal to the length specified in the LENGTH parameter.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:splitlinesbylength

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
**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.

![Subdivision Example](image)

Obr. 23.69: Left the input layer, middle maximum nodes value is 100 and right maximum value is 200

---

**Poznámka:** Subdividing a geometry can generate geometry parts that may not be valid and may contain self-intersections.

- Allows *features in-place modification*

  **Viz také:**

  *Explode lines, Line substring*

---

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer</td>
</tr>
<tr>
<td>Maximum nodes in parts</td>
<td>MAX_NODES</td>
<td>[number [≥ ]]</td>
<td>Maximum number of vertices each new geometry part is allowed to have. Fewer sub-parts for higher values.</td>
</tr>
<tr>
<td>Subdivided</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output (subdivided) vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subdivided</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: `qgis:subdivide`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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](Images/allow_in_place_modification.png)

Viz také:

*Translate, Rotate*

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer</td>
</tr>
<tr>
<td>Swapped</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
</tbody>
</table>

Specify the output vector layer. One of:

- Create Temporary Layer
  (TEMPORARY_OUTPUT)
- Save to File...
- Save to Geopackage...
- Save to PostGIS Table...

The file encoding can also be changed here.

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swapped</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output (swapped) vector layer</td>
</tr>
</tbody>
</table>

23.1. QGIS algorithm provider
Python code

Algorithm ID: qgis:swapxy

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**Tapered buffers**

Creates tapered buffer along line geometries, using a specified start and end buffer diameter.

![Tapered buffer example](image)

Obr. 23.70: Tapered buffer example

**Viz také:**

Variable width buffer (by M value), Buffer, Create wedge buffers
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Start width</td>
<td>START_WIDTH</td>
<td>[number ]</td>
<td>Represents the radius of the buffer applied at the start point of the line feature</td>
</tr>
<tr>
<td>End width</td>
<td>END_WIDTH</td>
<td>[number ]</td>
<td>Represents the radius of the buffer applied at the end point of the line feature</td>
</tr>
<tr>
<td>Segments</td>
<td>SEGMENTS</td>
<td>[number ]</td>
<td>Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.</td>
</tr>
<tr>
<td>Buffered</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output (buffer) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffered</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Output (buffer) polygon layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:taperedbuffer

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Allows features in-place modification
Obr. 23.71: Tessellated polygon (right)

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: polygon]</td>
<td>Input polygon vector layer</td>
</tr>
<tr>
<td>Tesselated</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tesselated</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Output multipolygonZ layer</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID**: qgis:tessellate

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
**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)

![Dashed red lines represent the transect of the input line layer](image)

Obr. 23.72: Dashed red lines represent the transect of the input line layer

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Length of the transect</td>
<td>LENGTH</td>
<td>[number ]</td>
<td>Length in map unit of the transect</td>
</tr>
<tr>
<td>Angle in degrees from the original line at the vertices</td>
<td>ANGLE</td>
<td>[number ]</td>
<td>Change the angle of the transect</td>
</tr>
<tr>
<td>Side to create the transect</td>
<td>SIDE</td>
<td>[enumeration]</td>
<td>Choose the side of the transect. Available options are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Left</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Both</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.98 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transect</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transect</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Output line layer</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:transect

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

![Obr. 23.73: Dashed lines represent the translated geometry of the input layer](image)

- Allows *features in-place modification*
- **Viz také:**
  - *Array of translated features, Offset lines, Rotate, Swap X and Y coordinates*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Offset distance (x-axis)</td>
<td>DELTA_X</td>
<td>[number ]</td>
<td>Displacement to apply on the X axis</td>
</tr>
<tr>
<td>Offset distance (y-axis)</td>
<td>DELTA_Y</td>
<td>[number ]</td>
<td>Displacement to apply on the Y axis</td>
</tr>
<tr>
<td>Offset distance (z-axis)</td>
<td>DELTA_Z</td>
<td>[number ]</td>
<td>Displacement to apply on the Z axis</td>
</tr>
<tr>
<td>Offset distance (m values)</td>
<td>DELTA_M</td>
<td>[number ]</td>
<td>Displacement to apply on the M axis</td>
</tr>
<tr>
<td>Translated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:translategeometry

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Variable distance buffer

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.

Poznámka: This algorithm is only available from the Graphical modeler.

Viz také:

Buffer
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Distance field</td>
<td>DISTANCE</td>
<td>[tablefield: numeric]</td>
<td>Attribute for the distance radius of the buffer</td>
</tr>
<tr>
<td>Segments</td>
<td>SEGMENTS</td>
<td>[number]</td>
<td>Controls the number of line segments to use to approximate a quarter circle when creating rounded offsets.</td>
</tr>
<tr>
<td>Dissolve result</td>
<td>DISSOLVE</td>
<td>[boolean]</td>
<td>Choose to dissolve the final buffer, resulting in a single feature covering all input features.</td>
</tr>
<tr>
<td>End cap style</td>
<td>END_CAP_STYLE</td>
<td>[enumeration]</td>
<td>Controls how line endings are handled in the buffer.</td>
</tr>
<tr>
<td>Join style</td>
<td>JOIN_STYLE</td>
<td>[enumeration]</td>
<td>Specifies whether round, miter or beveled joins should be used when offsetting corners in a line.</td>
</tr>
<tr>
<td>Miter limit</td>
<td>MITER_LIMIT</td>
<td>[number]</td>
<td>Only applicable for mitered join styles, and controls the maximum distance from the offset curve to use when creating a mitered join.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>

### Python code

**Algorithm ID:** qgis:variablendistancebuffer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

Obr. 23.76: Variable buffer example

Viz také:

Tapered buffers, Buffer, Set M value

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line vector layer</td>
</tr>
<tr>
<td>Segments</td>
<td>SEGMENTS</td>
<td>[number ]</td>
<td>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).</td>
</tr>
<tr>
<td>Buffered</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output (buffer) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGis Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffered</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Variable buffer polygon layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:bufferbym

```
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

**Default menu:** Vector ➤ Geometry Tools

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Buffer region (% of extent)</td>
<td>BUFFER</td>
<td>[number]</td>
<td>The extent of the output layer will be this much bigger than the extent of the input layer</td>
</tr>
</tbody>
</table>
| Voronoi polygons       | OUTPUT    | [vector: polygon] | Specify the output layer (with the Voronoi polygons). One of:
|                        |           |                 | • Create Temporary Layer (TEMPORARY_OUTPUT)                 |
|                        |           |                 | • Save to File…                                             |
|                        |           |                 | • Save to Geopackage…                                       |
|                        |           |                 | • Save to PostGIS Table…                                    |
|                        |           |                 | The file encoding can also be changed here.                 |

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voronoi polygons</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Voronoi polygons of the input point vector layer</td>
</tr>
</tbody>
</table>
Obr. 23.77: Voronoi polygons
Python code

```
Algorithm ID: qgis:voronoipolygons

```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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.1.16 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.

**Varování: Feature modification**

The attributes of the features are not modified, although properties such as area or length of the features will be modified by the clipping 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.

Obr. 23.78: Clipping operation between a two-features input layer and a single feature overlay layer (left) - resulting features are moved for clarity (right)

- Allows features in-place modification
- Default menu: Vector ® Geoprocessing Tools
- Viz také:
### Intersection, Difference

#### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer containing the features to be clipped</td>
</tr>
<tr>
<td>Overlay layer</td>
<td>OVERLAY</td>
<td>[vector: polygon]</td>
<td>Layer containing the clipping features</td>
</tr>
</tbody>
</table>
| Clipped     | OUTPUT      | [same as input]     | Specify the layer to contain the features from the input layer that are inside the overlay (clipping) layer. One of:  
• Create Temporary Layer  
• Save to File…  
• Save to Geopackage…  
• Save to PostGIS Table……  
The file encoding can also be changed here. |

#### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clipped</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Layer containing features from the input layer split by the overlay layer.</td>
</tr>
</tbody>
</table>

#### Python code

**Algorithm ID:** qgis:clip

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

#### 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.

Attributes are not modified (see warning).

- Allows features in-place modification
- **Default menu:** Vector ➤ Geoprocessing Tools
- **Viz také:**
  - Symmetrical difference, Clip
Obr. 23.79: Difference operation between a two-features input layer and a single feature overlay layer (left) - resulting features are moved for clarity (right)

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer to extract (parts of) features from.</td>
</tr>
<tr>
<td>Overlay layer</td>
<td>OVERLAY</td>
<td>[vector: any]</td>
<td>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.</td>
</tr>
</tbody>
</table>
| Difference     | OUTPUT    | [same as input]  | 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  
  • Save to File…  
  • Save to Geopackage…  
  • Save to PostGIS Table…  
  The file encoding can also be changed here. |

The file encoding can also be changed here.
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Layer containing (parts of) features from the input layer not overlapping the overlay layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:difference

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

**Viz také:**

*Clip*
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer to extract (parts of) features from.</td>
</tr>
<tr>
<td>Clip features to extent</td>
<td>CLIP</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
</tbody>
</table>
| Extracted             | OUTPUT      | [same as input] | Specify the layer to contain the features from the input layer that are inside the clip extent. One of:  
- Create Temporary Layer  
- Save to File…  
- Save to Geopackage…  
- Save to PostGIS Table……  
The file encoding can also be changed here. |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Layer containing the clipped features.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `qgis:extractbyextent`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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.

Attributes are not modified (see *warning*).

**Default menu:** *Vector [Vector Geoprocessing Tools]*

**Viz také:**

*Clip, Difference*
Obr. 23.80: The intersection operation: A two-features input layer and a single feature overlay layer (left) - resulting features are moved for clarity (right)

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector:any]</td>
<td>Layer to extract (parts of) features from.</td>
</tr>
<tr>
<td>Overlay layer</td>
<td>OVERLAY</td>
<td>[vector:any]</td>
<td>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.</td>
</tr>
<tr>
<td>Input fields to keep (leave empty to keep all fields)</td>
<td>INPUT_FIELDS</td>
<td>[tablefield: any] [list]</td>
<td>Field(s) of the input layer to keep in the output. If no fields are chosen all fields are taken.</td>
</tr>
<tr>
<td>Overlay fields to keep (leave empty to keep all fields)</td>
<td>OVERLAY_FIELDS</td>
<td>[tablefield: any] [list]</td>
<td>Field(s) of the overlay layer to keep in the output. If no fields are chosen all fields are taken.</td>
</tr>
<tr>
<td>Overlay fields prefix</td>
<td>OVERLAY_FIELDS_PREFIX</td>
<td>[string]</td>
<td>Prefix to add to the field names of the intersect layer's fields to avoid name collisions with fields in the input layer.</td>
</tr>
</tbody>
</table>
| Intersection                                | OUTPUT           | [same as input]  | 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  
  - Save to File…  
  - Save to Geopackage…  
  - Save to PostGIS Table…..  
  The file encoding can also be changed here. |

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intersect</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Layer containing (parts of) features from the input layer that overlap the overlay layer.</td>
</tr>
</tbody>
</table>
Line intersections

Creates point features where the lines from the two layers intersect.

![Points of intersection](image)

**Default menu:** Vector Analysis Tools

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>Input line layer.</td>
</tr>
<tr>
<td>Intersect layer</td>
<td>INTERSECT</td>
<td>[vector: line]</td>
<td>Layer to use to find line intersections.</td>
</tr>
<tr>
<td><strong>Input fields to keep (leave empty to keep all fields)</strong> Optional</td>
<td>INPUT_FIELDS</td>
<td>[tablefield: any] [list]</td>
<td>Field(s) of the input layer to keep in the output. If no fields are chosen all fields are taken.</td>
</tr>
<tr>
<td><strong>Intersect fields to keep (leave empty to keep all fields)</strong> Optional</td>
<td>INTERSECT_FIELDS</td>
<td>[tablefield: any] [list]</td>
<td>Field(s) of the intersect layer to keep in the output. If no fields are chosen all fields are taken.</td>
</tr>
<tr>
<td>Intersect fields prefix</td>
<td>OVERLAY_FIELDS_PREFIX</td>
<td>prefix</td>
<td>Prefix to add to the field names of the intersect layer's fields to avoid name collisions with fields in the input layer.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.103 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Intersection | OUTPUT       | [vector: point] | Specify the layer to contain the intersection points of the lines from the input and overlay layers. One of:  
|           |               |                 | • Create Temporary Layer                                            |
|           |               |                 | • Save to File…                                                     |
|           |               |                 | • Save to Geopackage…                                               |
|           |               |                 | • Save to PostGIS Table…                                            |
|           |               |                 | The file encoding can also be changed here.                         |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intersections</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Point vector layer with the intersections.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:lineintersections

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Split with lines**

Splits the lines or polygons in one layer using the lines 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.

Obr. 23.82: Split lines

![Split lines diagram](image)

- Allows *features in-place modification*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line, polygon]</td>
<td>Layer containing the lines or polygons to split.</td>
</tr>
<tr>
<td>Split layer</td>
<td>LINES</td>
<td>[vector: line]</td>
<td>Line layer whose lines are used to define the breaking points.</td>
</tr>
</tbody>
</table>
| Split       | OUTPUT | [same as input] | 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
  - Save to File…
  - Save to Geopackage…
  - Save to PostGIS Table……
  The file encoding can also be changed here. |

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer with split lines or polygons from input layer.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:splitwithlines

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Attributes are not modified (see warning).

Default menu: Vector ‣ Geoprocessing Tools

Viz také:

Difference, Clip, Intersection
Obr. 23.83: Symmetrical difference operation between a two-features input layer and a single feature overlay layer (left) - resulting features are moved for clarity (right)

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>First layer to extract (parts of) features from.</td>
</tr>
<tr>
<td>Overlay layer</td>
<td>OVERLAY</td>
<td>[vector: any]</td>
<td>Second layer to extract (parts of) features from. Ideally the geometry type should be the same as input layer.</td>
</tr>
<tr>
<td>Overlay fields prefix</td>
<td>OVERLAY_FIELDS_PREFIX</td>
<td>[string]</td>
<td>Prefix to add to the field names of the overlay layer's fields to avoid name collisions with fields in the input layer.</td>
</tr>
</tbody>
</table>
| Symmetrical difference | OUTPUT           | [same as input]| 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  
  * Save to File...  
  * Save to Geopackage...  
  * Save to PostGIS Table...  
  The file encoding can also be changed here. |
### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetrical difference</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Layer containing (parts of) features from each layer not overlapping the other layer.</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:symmetricaldifference

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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.

![Union operation](image)

**Obr. 23.84:** Union operation with a single input layer of three overlapping features (left) - resulting features are 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. 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.

**Poznámka:** For **union(A,B)** algorithm, if there are overlaps among geometries of layer A or among geometries
Obr. 23.85: Union operation between a two-features input layer and a single feature overlay layer (left) - resulting features are moved for clarity (right)

of layer B, these are not resolved: you need to do \texttt{union(union(A,B))} to resolve all overlaps, i.e. run single layer \texttt{union(X)} on the produced result \(X=union(A,B)\).

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|}
\hline
\textbf{Label} & \textbf{Název} & \textbf{Type} & \textbf{Popis} \\
\hline
\textbf{Input layer} & \texttt{INPUT} & \texttt{vector:any} & Input vector layer to split at any intersections. \\
\hline
\textbf{Overlay layer} & \texttt{OVERLAY} & \texttt{vector:any} & Layer that will be combined to the first one. Ideally the geometry type should be the same as input layer. \\
\textbf{Optional} & & & \\
\hline
\textbf{Overlay fields prefix} & \texttt{OVERLAY\_FIELDS} & \texttt{[same as input]} & Prefix to add to the field names of the overlay layer’s fields to avoid name collisions with fields in the input layer. \\
\textbf{Optional} & & & \\
\hline
\textbf{Union} & \texttt{OUTPUT} & \texttt{[same as input]} & Specify the layer to contain the (split and duplicated) features from the input layer and the overlay layer. One of:
- Create Temporary Layer
- Save to File…
- Save to Geopackage…
- Save to PostGIS Table……
  The file encoding can also be changed here. \\
\hline
\end{tabular}
\end{table}
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Layer containing all the overlapping and non-overlapping parts from the processed layer(s).</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:union

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.1.17 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.

Viz také:

Select by attribute

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Layer to extract features from.</td>
</tr>
<tr>
<td>Selection attribute</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Filtering field of the layer</td>
</tr>
<tr>
<td>Operator</td>
<td>OPERATOR</td>
<td>[enumeration]</td>
<td>Many different operators are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — =</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — ≠</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — &gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — &gt;=</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — &lt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — &lt;=</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — begins with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — contains</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — is null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — is not null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — does not contain</td>
</tr>
<tr>
<td>Value Optional</td>
<td>VALUE</td>
<td>[string]</td>
<td>Value to be evaluated</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Extracted (attribute) | OUTPUT | [same as input] | Specify the output vector layer for matching features. One of:
Default: [Create Temporary Layer (TEMPORARY_OUTPUT)]
- Create Temporary Layer
- Save to File...
- Save to Geopackage...
- Save to PostGIS Table...
The file encoding can also be changed here.

Extracted (non-matching) | FAIL_OUTPUT | [same as input] | Specify the output vector layer for non-matching features. One of:
Default: [Skip output]
- Skip Output
- Create Temporary Layer (TEMPORARY_OUTPUT)
- Save to File...
- Save to Geopackage...
- Save to PostGIS Table...

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted (attribute)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with matching features from the input layer</td>
</tr>
<tr>
<td>Extracted (non-matching)</td>
<td>FAIL_OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with non-matching features from the input layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** qgis:extractbyattribute

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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 *Výrazy*.

**Viz také:**

*Select by expression*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Expression</td>
<td>EXPRESSION</td>
<td>[expression]</td>
<td>Expression to filter the vector layer</td>
</tr>
<tr>
<td>Matching features</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer for matching features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Non-matching</td>
<td>FAIL_OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer for non-matching features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching features</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with matching features from the input layer</td>
</tr>
<tr>
<td>Non-matching</td>
<td>FAIL_OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with non-matching features from the input layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:extractbyexpression

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Available geometric predicates are:

**Intersect** Tests whether a geometry intersects another. Returns 1 (true) if the geometries spatially intersect (share any portion of space - overlap or touch) and 0 if they don’t. In the picture above, this will select circles 1, 2 and 3.
Obr. 23.86: In this example, the dataset from which we want to select (the source vector layer) consists of the green circles, the orange rectangle is the dataset that it is being compared to (the intersection vector layer).

**Contain** Returns 1 (true) if and only if no points of b lie in the exterior of a, and at least one point of the interior of b lies in the interior of a. In the picture, no circle is selected, but the rectangle would be if you would select it the other way around, as it contains a circle completely. This is the opposite of are within.

**Disjoint** Returns 1 (true) if the geometries do not share any portion of space (no overlap, not touching). Only circle 4 is selected.

**Equal** Returns 1 (true) if and only if geometries are exactly the same. No circles will be selected.

**Touch** Tests whether a geometry touches another. Returns 1 (true) if the geometries have at least one point in common, but their interiors do not intersect. Only circle 3 is selected.

**Overlap** Tests whether a geometry overlaps another. Returns 1 (true) if the geometries share space, are of the same dimension, but are not completely contained by each other. Only circle 2 is selected.

**Are within** Tests whether a geometry is within another. Returns 1 (true) if geometry a is completely inside geometry b. Only circle 1 is selected.

**Cross** Returns 1 (true) if the supplied geometries have some, but not all, interior points in common and the actual crossing is of a lower dimension than the highest supplied geometry. For example, a line crossing a polygon will cross as a line (selected). Two lines crossing will cross as a point (selected). Two polygons cross as a polygon (not selected).

Viz také:

*Select by location*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract features from</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Where the features (geometric predicate)</td>
<td>PREDICATE</td>
<td>[enumeration] [list]</td>
<td>Spatial condition for the selection. One or more of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — intersect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — contain</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — disjoint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — equal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — touch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — overlap</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — are within</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — cross</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If more than one condition is chosen, at least one of them (OR operation) has to be met for a feature to be extracted.</td>
</tr>
<tr>
<td>By comparing to the features from</td>
<td>INTERSECT</td>
<td>[vector: any]</td>
<td>Intersection vector layer</td>
</tr>
<tr>
<td>Extracted (location)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted (location)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with features from the input layer that have the chosen spatial relationship(s) with one or more features in the comparison layer.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:extractbylocation

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Viz také:

Random selection

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Source vector layer to select the features from</td>
</tr>
<tr>
<td>Method</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>Random selection methods. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Number of selected features</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Percentage of selected features</td>
</tr>
<tr>
<td>Number/percentage of</td>
<td>NUMBER</td>
<td>[number]</td>
<td>Number or percentage of features to select</td>
</tr>
<tr>
<td>selected features</td>
<td></td>
<td></td>
<td>Default: 10</td>
</tr>
<tr>
<td>Extracted (random)</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Specify the output vector layer for the randomly selected features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Vector layer containing randomly selected features</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted (random)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer containing randomly selected features from the input layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:randomextract

```python
import processing
tmp_output = 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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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.

Viz také:

Random selection within subsets

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector:any]</td>
<td>Vector layer to select the features from</td>
</tr>
<tr>
<td>ID field</td>
<td>FIELD</td>
<td>[tablefield:any]</td>
<td>Category of the source vector layer to select the features from</td>
</tr>
<tr>
<td>Method</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>Random selection method. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Number of selected features</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Percentage of selected features</td>
</tr>
<tr>
<td>Number/percentage of selected features</td>
<td>NUMBER</td>
<td>[number]</td>
<td>Number or percentage of features to select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 10</td>
<td></td>
</tr>
<tr>
<td>Extracted (random stratified)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer for the randomly selected features. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer] (TEMPORARY_OUTPUT)</td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted (random stratified)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer containing randomly selected features from the input layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:randomextractwithinsubsets

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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

Viz také:

Random extract

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer for the selection</td>
</tr>
<tr>
<td>Method</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>Random selection method. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Number of selected features</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Percentage of selected features</td>
</tr>
<tr>
<td>Number/percentage of selected features</td>
<td>NUMBER</td>
<td>[number]</td>
<td>Number or percentage of features to select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 10</td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[same as input]</td>
<td>The input layer with features selected</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:randomselection

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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

Viz také:
Random extract within subsets

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer to select features in</td>
</tr>
<tr>
<td>ID field</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Category of the input layer to select the fea-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tures from</td>
</tr>
<tr>
<td>Method</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>Random selection method. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Number of selected features</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Percentage of selected features</td>
</tr>
<tr>
<td>Number/percentage of</td>
<td>NUMBER</td>
<td>[number]</td>
<td>Number or percentage of features to select</td>
</tr>
<tr>
<td>selected features</td>
<td></td>
<td>Default: 10</td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[same as input]</td>
<td>The input layer with features selected</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:randomselectionwithinsubsets

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Viz také:

Extract by attribute

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Vector layer to select features in</td>
</tr>
<tr>
<td>Selection attribute</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Filtering field of the layer</td>
</tr>
</tbody>
</table>
Tabulka 23.108 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>OPERATOR</td>
<td>[enumeration]</td>
<td>Many different operators are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — =</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — ≠</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — &gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — &gt;=</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — &lt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — &lt;=</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — begins with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — contains</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — is null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — is not null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — does not contain</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Optional</th>
<th>VALUE</th>
<th>[string]</th>
<th>Value to be evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify current selection by</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>Default: 0</td>
<td>How the selection of the algorithm should be managed. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0 — creating new selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1 — adding to current selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 2 — selecting within current selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 3 — removing from current selection</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[same as input]</td>
<td>The input layer with features selected</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `qgis:selectbyattribute`

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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 Výrazy.

**Viz také:**

*Extract by expression*
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Expression</td>
<td>EXPRESSION</td>
<td>[expression]</td>
<td>Expression to filter the input layer</td>
</tr>
<tr>
<td>Modify current selection by</td>
<td>METHOD</td>
<td>[enumeration]</td>
<td>How the selection of the algorithm should be managed. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — creating new selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — adding to current selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — selecting within current selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — removing from current selection</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[same as input]</td>
<td>The input layer with features selected</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:selectbyexpression

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Select by location

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.

Available geometric predicates are:

**Intersect** Tests whether a geometry intersects another. Returns 1 (true) if the geometries spatially intersect (share any portion of space - overlap or touch) and 0 if they don’t. In the picture above, this will select circles 1, 2 and 3.

**Contain** Returns 1 (true) if and only if no points of b lie in the exterior of a, and at least one point of the interior of b lies in the interior of a. In the picture, no circle is selected, but the rectangle would be if you would select it the other way around, as it contains a circle completely. This is the opposite of *are within*.

**Disjoint** Returns 1 (true) if the geometries do not share any portion of space (no overlap, not touching). Only circle 4 is selected.

**Equal** Returns 1 (true) if and only if geometries are exactly the same. No circles will be selected.

**Touch** Tests whether a geometry touches another. Returns 1 (true) if the geometries have at least one point in common, but their interiors do not intersect. Only circle 3 is selected.

**Overlap** Tests whether a geometry overlaps another. Returns 1 (true) if the geometries share space, are of the same dimension, but are not completely contained by each other. Only circle 2 is selected.
Obr. 23.87: In this example, the dataset from which we want to select (the source vector layer) consists of the green circles, the orange rectangle is the dataset that it is being compared to (the intersection vector layer).

**Are within** Tests whether a geometry is within another. Returns 1 (true) if geometry a is completely inside geometry b. Only circle 1 is selected.

**Cross** Returns 1 (true) if the supplied geometries have some, but not all, interior points in common and the actual crossing is of a lower dimension than the highest supplied geometry. For example, a line crossing a polygon will cross as a line (selected). Two lines crossing will cross as a point (selected). Two polygons cross as a polygon (not selected).

**Default menu**: Vector → Research Tools

**Viz také**: Extract by location

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>features</td>
<td>INPUT [vector: any]</td>
<td>Input vector layer</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.109 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| Where the features (geometric predicate) | PREDICATE | [enumeration] [list] | Spatial condition for the selection. One or more of:  
• 0 — intersect  
• 1 — contain  
• 2 — disjoint  
• 3 — equal  
• 4 — touch  
• 5 — overlap  
• 6 — are within  
• 7 — cross  
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: any] | Intersection vector layer |
| Modify current selection by | METHOD | [enumeration] | How the selection of the algorithm should be managed. One of:  
• 0 — creating new selection  
• 1 — adding to current selection  
• 2 — selecting within current selection  
• 3 — removing from current selection |

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[same as input]</td>
<td>The input layer with features selected</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:selectbylocation

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

### 23.1.18 Vector table

**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.
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer.</td>
</tr>
<tr>
<td>Field name</td>
<td>FIELD_NAME</td>
<td>[string] Default: AUTO</td>
<td>Name of the field with autoincremental values</td>
</tr>
<tr>
<td>Start values at</td>
<td>START</td>
<td>[number] Default: 0</td>
<td>Choose the initial number of the incremental count</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group values by</td>
<td>GROUP_FIELDS</td>
<td>[tablefield: any] [list]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sort expression</td>
<td>SORT_EXPRESSION</td>
<td>[expression]</td>
<td>Use an expression to sort the features in the layer either globally or if set, based on group fields.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sort ascending</td>
<td>SORT_ASCENDING</td>
<td>[boolean] Default: True</td>
<td>When a sort expression is set, use this option to control the order in which features are assigned values.</td>
</tr>
<tr>
<td>Sort nulls first</td>
<td>SORT_NULLS_FIRST</td>
<td>[boolean] Default: False</td>
<td>When a sort expression is set, use this option to set whether Null values are counted first or last.</td>
</tr>
<tr>
<td>Incremented</td>
<td>OUTPUT</td>
<td>[same as input] Default: [Create temporary layer]</td>
<td>Specify the output vector layer with the auto increment field. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremented</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with auto incremental field</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** `qgis:addautoincrementalfield`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
Add field to attributes table

Add 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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input layer</td>
</tr>
<tr>
<td>Field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Name of the new field</td>
</tr>
<tr>
<td>Field type</td>
<td>FIELD_TYPE</td>
<td>[enumeration]</td>
<td>Type of the new field. You can choose between:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Float</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — String</td>
</tr>
<tr>
<td>Field length</td>
<td>FIELD_LENGTH</td>
<td>[number]</td>
<td>Length of the field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 10</td>
<td></td>
</tr>
<tr>
<td>Field precision</td>
<td>FIELD_PRECISION</td>
<td>[number]</td>
<td>Precision of the field. Useful with Float field type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td></td>
</tr>
<tr>
<td>Added</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table……</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with new field added</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:addfieldtoattributestable

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input layer.</td>
</tr>
<tr>
<td>Class field</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Features that have the same value for this field will get the same index.</td>
</tr>
<tr>
<td>Output field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Name of the new field containing the indexes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 'NUM_FIELD'</td>
<td></td>
</tr>
<tr>
<td>Layer with index field</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Vector layer with the numeric field containing indexes. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td>Class summary</td>
<td>SUMMARY_OUTPUT</td>
<td>[table]</td>
<td>Specify the table to contain the summary of the class field mapped to the corresponding unique value. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Skip output]</td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer with index field</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with the numeric field containing indexes.</td>
</tr>
<tr>
<td>Class summary</td>
<td>SUMMARY_OUTPUT</td>
<td>[table]</td>
<td>Table with summary of the class field mapped to the corresponding unique value.</td>
</tr>
</tbody>
</table>
**Python code**

**Algorithm ID:** qgis:adduniquevalueindexfield

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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).

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>The input layer.</td>
</tr>
<tr>
<td>Coordinate system</td>
<td>CRS</td>
<td>[crs]</td>
<td>Coordinate reference system to use for the generated x and y fields.</td>
</tr>
<tr>
<td>Field prefix</td>
<td>PREFIX</td>
<td>[string]</td>
<td>Prefix to add to the new field names to avoid name collisions with fields in the input layer.</td>
</tr>
<tr>
<td>Added fields</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added fields</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output layer - identical to the input layer but with two new double fields, x and y.</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:addxyfieldstolayer

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
## 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.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Result field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Name of the new field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 'NewField'</td>
<td></td>
</tr>
<tr>
<td>Field type</td>
<td>FIELD_TYPE</td>
<td>[enumeration]</td>
<td>Type of the new field. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Float</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — String</td>
</tr>
<tr>
<td>Field length</td>
<td>FIELD_LENGTH</td>
<td>[number]</td>
<td>Length of the field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 10</td>
<td></td>
</tr>
<tr>
<td>Field precision</td>
<td>FIELD_PRECISION</td>
<td>[number]</td>
<td>Precision of the field. Useful with Float field type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 3</td>
<td></td>
</tr>
<tr>
<td>Global expression</td>
<td>GLOBAL</td>
<td>[string]</td>
<td>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.</td>
</tr>
<tr>
<td>Formula</td>
<td>FORMULA</td>
<td>[string]</td>
<td>The Python formula to evaluate. Example: To calculate the area of an input polygon layer you can add: value = $geom.area()</td>
</tr>
<tr>
<td>Calculated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the vector layer with the new calculated field. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create temporary layer]</td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculated</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with the new calculated field</td>
</tr>
</tbody>
</table>
**Python code**

**Algorithm ID:** qgis:advancedpythonfieldcalculator

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Drop field(s)**

Takes a vector layer and generates a new one that has the same features but without the selected columns.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer to drop field(s) from</td>
</tr>
<tr>
<td>Fields to drop</td>
<td>COLUMN</td>
<td>[tablefield: any] [list]</td>
<td>The field(s) to drop</td>
</tr>
<tr>
<td>Remaining fields</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output vector layer with the remaining fields. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table…</td>
</tr>
</tbody>
</table>

The file encoding can also be changed here.

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remaining fields</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Vector layer with the remaining fields</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:deletecolumn

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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 OGR (when reading an OSM file with the other_tags field).

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>HStore field</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>The field(s) to drop</td>
</tr>
<tr>
<td>Expected list of fields</td>
<td>EXPECTED_FIELDS</td>
<td>[string]</td>
<td>Comma-separated list of fields to extract. The HStore field will be updated by removing these keys.</td>
</tr>
<tr>
<td>separated by a comma</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expected list of fields</td>
<td>EXP</td>
<td>[string]</td>
<td>Specify the output vector layer. One of:</td>
</tr>
<tr>
<td>separated by a comma</td>
<td>E</td>
<td>Default: ';'</td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td>EXP_OUTPUT</td>
<td>[same as input]</td>
<td>• Save to Geopackage…</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Create</td>
<td>• Save to PostGIS Table…</td>
</tr>
<tr>
<td></td>
<td></td>
<td>temporary layer]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploded</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:explodehstorefield

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.
**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer containing the binary data</td>
</tr>
<tr>
<td>Binary field</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field containing the binary data</td>
</tr>
<tr>
<td>File name</td>
<td>FILENAME</td>
<td>[expression]</td>
<td>Field or expression-based text to name each output file</td>
</tr>
<tr>
<td>Destination folder</td>
<td>FOLDER</td>
<td>[folder]</td>
<td>Folder in which to store the output files. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary Directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Directory...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folder</td>
<td>FOLDER</td>
<td>[folder]</td>
<td>The folder that contains the output files.</td>
</tr>
</tbody>
</table>

**Python code**

Algorithm ID: `qgis:extractbinary`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Feature filter**

Filters features from the input layer and redirects them to one or several outputs. If you do not know about any attribute names that are common to all possible input layers, filtering is only possible on the feature geometry and general record mechanisms, such as $id and uuid.

**Poznámka:** This algorithm is only available from the Graphical modeler.
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output (one or more)</td>
<td>native:filter_&lt;name of filter&gt;</td>
<td>same as input</td>
<td>The output layers with filtered features (as many as there are filters).</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** qgis:featurefilter

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Field calculator

Opens the field calculator (see *Výrazy*). 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 graphical modeler*.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The layer to calculate on</td>
</tr>
<tr>
<td>Output field name</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>The name of the field for the results</td>
</tr>
<tr>
<td>Output field type</td>
<td>FIELD_TYPE</td>
<td>[enumeration]</td>
<td>The type of the field. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Float</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — String</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Date</td>
</tr>
<tr>
<td>Output field width</td>
<td>FIELD_LENGTH</td>
<td>[number]</td>
<td>The length of the result field (minimum 0)</td>
</tr>
<tr>
<td>Field precision</td>
<td>FIELD_PRECISION</td>
<td>[number]</td>
<td>The precision of the result field (minimum 0, maximum 15)</td>
</tr>
<tr>
<td>Create new field</td>
<td>NEW_FIELD</td>
<td>[boolean]</td>
<td>Should the result field be a new field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: True</td>
<td></td>
</tr>
<tr>
<td>Formula</td>
<td>FORMULA</td>
<td>[expression]</td>
<td>The formula to use to calculate the result</td>
</tr>
<tr>
<td>Output file</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Specification of the output layer</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculated</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Output layer with the calculated field values</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: qgis:fieldcalculator

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Refactor layer fields 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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The layer to modify</td>
</tr>
</tbody>
</table>
## Tabulka 23.117 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields mapping</td>
<td>FIELDS_MAPPING</td>
<td>[list]</td>
<td>List of output fields with their definitions. The embedded table lists all the fields of the source layer and allows you to edit them:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Click to create a new field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Click to remove a field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Use and to change the selected field order.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Click to reset to the default view.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For each of the fields you'd like to reuse, you need to fill the following options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Source expression</strong> <em>(expression)</em> <em>(expression)</em> Field or expression from the input layer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Field name</strong> <em>(name)</em> <em>(string)</em> Name of the field in the output layer. By default input field name is kept.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Type</strong> <em>(type)</em> <em>(enumeration)</em> Data type of the output field. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Date (14)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• DateTime (16)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Double (6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Integer (2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Integer64 (4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• String (10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Boolean (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Length</strong> <em>(length)</em> <em>(number)</em> Length of the output field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Precision</strong> <em>(precision)</em> <em>(number)</em> Precision of the output field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fields from another layer can be loaded into the field list in <em>Load fields from layer</em>.</td>
</tr>
<tr>
<td>Refactored</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Specification of the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table……</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refactored</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Output layer with refactored fields</td>
</tr>
</tbody>
</table>
Obr. 23.88: Refactor fields dialog
**Python code**

**Algorithm ID:** qgis:refactorfields

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer.</td>
</tr>
<tr>
<td>Text attribute to</td>
<td>FIELD</td>
<td>[tablefield: string]</td>
<td>The string field for the input layer that is to be converted to a float field.</td>
</tr>
<tr>
<td>convert to float</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Float from text</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Geopackage...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to PostGIS Table......</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float from text</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Output vector layer with the string field converted into a float field</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** qgis:texttofloat

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

23.1. QGIS algorithm provider
23.2 GDAL algorithm provider

GDAL (Geospatial Data Abstraction Library) is a translator library for raster and vector geospatial data formats. Algorithms in the Processing Framework are derived from the GDAL raster programs and GDAL vector programs.

23.2.1 Rastrová analýza

Aspect

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 [9] Analysis

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input elevation raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The number of the band to use as elevation</td>
</tr>
<tr>
<td>Default: 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Return trigonometric angle</td>
<td>TRIG_ANGLE</td>
<td>[boolean]</td>
<td>Activating the trigonometric angle results in different categories: 0° (East), 90° (North), 180° (West), 270° (South).</td>
</tr>
<tr>
<td>instead of azimuth</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Return 0 for flat instead of -9999</td>
<td>ZERO_FLAT</td>
<td>[boolean]</td>
<td>Activating this option will insert a 0-value for the value -9999 on flat areas.</td>
</tr>
<tr>
<td>Compute edges</td>
<td>COMPUTE_EDGES</td>
<td>[boolean]</td>
<td>Generates edges from the elevation raster</td>
</tr>
<tr>
<td>Use Zevenbergen&amp;Thorne formula instead of the Horn's one</td>
<td>ZEVENBERGEN</td>
<td>[boolean]</td>
<td>Activates Zevenbergen&amp;Thorne formula for smooth landscapes</td>
</tr>
<tr>
<td>Default: False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td></td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Aspect</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspect</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster with angle values in degrees</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:aspect

```python
import processing
processing.run("gdal:aspect", {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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input elevation raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The number of the band to use as elevation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1</td>
<td></td>
</tr>
<tr>
<td>Compute edges</td>
<td>COMPUTE_EDGES</td>
<td>[boolean]</td>
<td>Generates edges from the elevation raster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Color configuration file</td>
<td>COLOR_TABLE</td>
<td>[file]</td>
<td>A text-based color configuration file</td>
</tr>
<tr>
<td>Matching mode</td>
<td>MATCH_MODE</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 2</td>
<td>• 0 — Use strict color matching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Use closest RGBA quadruples</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Use smoothly blended colours</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: '.'</td>
<td>raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Color relief</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color relief</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>A 4-band output raster</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: `gdal:colorrelief`

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Fill nodata

Fill raster regions with no data values by interpolation from edges. The values for the no-data 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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The band to operate on. No-data values must be represented by the value 0.</td>
</tr>
<tr>
<td>Maximum distance (in pixels) to search out for values to interpolate</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>The number of pixels to search in all directions to find values to interpolate from</td>
</tr>
<tr>
<td>Number of smoothing iterations to run after the interpolation</td>
<td>ITERATIONS</td>
<td>[number]</td>
<td>The number of 3x3 filter passes to run (0 or more) to smoothen the results of the interpolation.</td>
</tr>
<tr>
<td>Do not use default validity mask for the input band</td>
<td>NO_MASK</td>
<td>[boolean]</td>
<td>Activates the user-defined validity mask</td>
</tr>
<tr>
<td>Validity mask</td>
<td>MASK_LAYER</td>
<td>[raster]</td>
<td>A raster layer that defines the areas to fill.</td>
</tr>
<tr>
<td>Additional creation options Optional</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Tabulka 23.120 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional command-line parameters</strong> Optional</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Default: None Add extra GDAL command line options</td>
</tr>
<tr>
<td><strong>Filled</strong></td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Default: [Save to temporary file] Specification of the output raster layer. One of:  &lt;br&gt;• Save to a Temporary File  &lt;br&gt;• Save to File…  &lt;br&gt;The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Filled</strong></td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `gdal:fillnodata`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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*

**Viz také:**

GDAL grid tutorial

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Point layer</strong></td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
</tbody>
</table>
### Tabulka 23.121 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data metric to use</td>
<td>METRIC</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — Minimum, minimum value found in grid node search ellipse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Maximum, maximum value found in grid node search ellipse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Range, a difference between the minimum and maximum values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>found in grid node search ellipse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Count, a number of data points found in grid node search ellipse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Average distance, an average distance between the grid node</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>center of the search ellipse and all of the data points found in grid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>node search ellipse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Average distance between points, an average distance between</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the data points found in grid node search ellipse. The distance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>between each pair of points within ellipse is calculated and average</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of all distances is set as a grid node value</td>
</tr>
<tr>
<td>The first radius of search ellipse</td>
<td>RADIUS_1</td>
<td>[number]</td>
<td>The first radius (X axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>The second radius of search ellipse</td>
<td>RADIUS_2</td>
<td>[number]</td>
<td>The second radius (Y axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>Angle of search ellipse rotation in</td>
<td>ANGLE</td>
<td>[number]</td>
<td>Angle of ellipse rotation in degrees. Ellipse rotated counter</td>
</tr>
<tr>
<td>degrees (counter clockwise)</td>
<td></td>
<td>Default: 0</td>
<td>clockwise.</td>
</tr>
<tr>
<td>Minimum number of data points to</td>
<td>MIN_POINTS</td>
<td>[number]</td>
<td>Minimum number of data points to average. If less amount of points</td>
</tr>
<tr>
<td>use</td>
<td></td>
<td>Default: 0</td>
<td>found the grid node considered empty and will be filled with NODATA</td>
</tr>
<tr>
<td>Nodata</td>
<td>NODATA</td>
<td>[number]</td>
<td>No data marker to fill empty points</td>
</tr>
<tr>
<td>Z value from field</td>
<td>Z_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field for the interpolation</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td>created (colors, block size, file compression...). For convenience,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
</tbody>
</table>
### Grid (IDW with nearest neighbor searching)

Computes the Inverse Distance to a Power gridding combined to the nearest neighbor method. Ideal when a maximum number of data points to use is required.

This algorithm is derived from the GDAL grid utility.

Viz také:

GDAL grid tutorial

---

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 5</td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
</tbody>
</table>

| Interpolated (data metrics) | OUTPUT | [raster] | Specify the output raster layer with interpolated values. One of: |
|                            |        |          |   • Save to a Temporary File  |
|                            |        |          |   • Save to File...  |
|                            |        |          | The file encoding can also be changed here. |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated (data metrics)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster with interpolated values</td>
</tr>
</tbody>
</table>

### Python code

Algorithm ID: `gdal:griddatametrics`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

---

23.2. GDAL algorithm provider
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Point layer</strong></td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td><strong>Weighting power</strong></td>
<td>POWER</td>
<td>[number] Default: 2.0</td>
<td>Weighting power</td>
</tr>
<tr>
<td><strong>Smoothing</strong></td>
<td>SMOOTHING</td>
<td>[number] Default: 0.0</td>
<td>Smoothing parameter</td>
</tr>
<tr>
<td><strong>The radius of the search circle</strong></td>
<td>RADIUS</td>
<td>[number] Default: 1.0</td>
<td>The radius of the search circle</td>
</tr>
<tr>
<td><strong>Maximum number of data points to use</strong></td>
<td>MAX_POINTS</td>
<td>[number] Default: 12</td>
<td>Do not search for more points than this number.</td>
</tr>
<tr>
<td><strong>Minimum number of data points to use</strong></td>
<td>MIN_POINTS</td>
<td>[number] Default: 0</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Nodata</strong></td>
<td>NODATA</td>
<td>[number] Default: 0.0</td>
<td>No data marker to fill empty points</td>
</tr>
<tr>
<td><strong>Z value from field</strong></td>
<td>Z_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field for the interpolation</td>
</tr>
<tr>
<td><strong>Additional creation options</strong></td>
<td>OPTIONS</td>
<td>[string] Default: ;</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td><strong>Additional command-line parameters</strong></td>
<td>EXTRA</td>
<td>[string] Default: None</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td><strong>Output data type</strong></td>
<td>DATA_TYPE</td>
<td>[enumeration] Default: 5</td>
<td>Defines the data type of the output raster file. Options: 0 - Byte, 1 - Int16, 2 - UInt16, 3 - UInt32, 4 - Int32, 5 - Float32, 6 - Float64, 7 - CInt16, 8 - CInt32, 9 - CFloat32, 10 - CFloat64</td>
</tr>
<tr>
<td><strong>Interpolated (IDW with NN search)</strong></td>
<td>OUTPUT</td>
<td>[raster] Default: [Save to temporary file]</td>
<td>Specify the output raster layer with interpolated values. One of: Save to a Temporary File, Save to File... The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated (IDW with NN search)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster with interpolated values</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:gridinversedistancenearestneighbor

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Grid (Inverse distance to a power)

The Inverse Distance to a Power gridding method is a weighted average interpolator.

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

Viz také:

GDAL grid tutorial

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>Weighting power</td>
<td>POWER</td>
<td>[number] Default: 2.0</td>
<td>Weighting power</td>
</tr>
<tr>
<td>Smoothing</td>
<td>SMOOTHING</td>
<td>[number] Default: 0.0</td>
<td>Smoothing parameter</td>
</tr>
<tr>
<td>The first radius of search ellipse</td>
<td>RADIUS_1</td>
<td>[number] Default: 0.0</td>
<td>The first radius (X axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>The second radius of search ellipse</td>
<td>RADIUS_2</td>
<td>[number] Default: 0.0</td>
<td>The second radius (Y axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>Angle of search ellipse rotation in degrees (counter clockwise)</td>
<td>ANGLE</td>
<td>[number] Default: 0.0</td>
<td>Angle of ellipse rotation in degrees. Ellipse rotated counter clockwise.</td>
</tr>
<tr>
<td>Maximum number of data points to use</td>
<td>MAX_POINTS</td>
<td>[number] Default: 0</td>
<td>Do not search for more points than this number.</td>
</tr>
</tbody>
</table>
Tabulka 23.123 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum number of data points to use</td>
<td>MIN_POINTS</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Nodata</td>
<td>NODATA</td>
<td>[number]</td>
<td>No data marker to fill empty points</td>
</tr>
<tr>
<td>Z value from field</td>
<td>Z_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field for the interpolation</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
<tr>
<td>Interpolated (IDW)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer with interpolated values. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated (IDW)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster with interpolated values</td>
</tr>
</tbody>
</table>
### Python code

**Algorithm ID:** `gdal:gridinversedistance`

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

### 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.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Point layer</strong></td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td><strong>Search distance</strong></td>
<td>RADIUS</td>
<td>[number]</td>
<td>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, no data value will be used.</td>
</tr>
<tr>
<td><strong>Nodata</strong></td>
<td>NODATA</td>
<td>[number]</td>
<td>No data marker to fill empty points</td>
</tr>
<tr>
<td><strong>Z value from field</strong></td>
<td>Z_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field for the interpolation</td>
</tr>
<tr>
<td><strong>Additional creation options</strong></td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td><strong>Additional command-line parameters</strong></td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
</tbody>
</table>

Pokračujte na daši stránce
<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output data type</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td>Default: 5</td>
<td>- 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 10 — CFloat64</td>
</tr>
</tbody>
</table>

| Interpolated (Linear) | OUTPUT       | Specify the output raster layer with interpolated values. One of:   |
|                       | [raster]     | - Save to a Temporary File                                          |
|                       | Default:     | - Save to File...                                                   |
|                       | [Save to temporary file] | The file encoding can also be changed here.                        |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated (Linear)</td>
<td>OUTPUT</td>
<td>Output raster with interpolated values</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `gdal:gridlinear`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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*

**Viz také:**

GDAL grid tutorial
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>The first radius of search ellipse</td>
<td>RADIUS_1</td>
<td>[number]</td>
<td>The first radius (X axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>The second radius of search ellipse</td>
<td>RADIUS_2</td>
<td>[number]</td>
<td>The second radius (Y axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>Minimum number of data points to use</td>
<td>MIN_POINTS</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Nodata</td>
<td>NODATA</td>
<td>[number]</td>
<td>No data marker to fill empty points</td>
</tr>
<tr>
<td>Z value from field</td>
<td>Z_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field for the interpolation</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td>Interpolated (moving average)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer. One of:</td>
</tr>
</tbody>
</table>

**Output data type**

- **DATA_TYPE**
  - Default: 5
  - Defines the data type of the output raster file. Options:
    - 0 — Byte
    - 1 — Int16
    - 2 — UInt16
    - 3 — UInt32
    - 4 — Int32
    - 5 — Float32
    - 6 — Float64
    - 7 — CInt16
    - 8 — CInt32
    - 9 — CFloat32
    - 10 — CFloat64

**Interpolated (moving average)**

- **OUTPUT**
  - Default: [Save to temporary file]
  - Specify the output raster layer. One of:
    - Save to a Temporary File
    - Save to File...
    - The file encoding can also be changed here.
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated (moving average)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster with interpolated values</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:gridaverage

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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

Viz také:

GDAL grid tutorial

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point layer</td>
<td>INPUT</td>
<td>[vector: point]</td>
<td>Input point vector layer</td>
</tr>
<tr>
<td>The first radius of search ellipse</td>
<td>RADIUS_1</td>
<td>[number]</td>
<td>The first radius (X axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>The second radius of search ellipse</td>
<td>RADIUS_2</td>
<td>[number]</td>
<td>The second radius (Y axis if rotation angle is 0) of the search ellipse</td>
</tr>
<tr>
<td>Nodata</td>
<td>NODATA</td>
<td>[number]</td>
<td>No data marker to fill empty points</td>
</tr>
<tr>
<td>Z value from field</td>
<td>Z_FIELD</td>
<td>[tablefield: numeric]</td>
<td>Field for the interpolation</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Table 23.126 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 5</td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
<tr>
<td>Interpolated (Nearest neighbour)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer with interpolated values. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolated (Nearest neighbour)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster with interpolated values</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:gridnearestneighbor

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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 [Raster] Analysis

23.2. GDAL algorithm provider
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input Elevation raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>Band containing the elevation information</td>
</tr>
<tr>
<td>Z factor (vertical exaggeration)</td>
<td>Z_FACTOR</td>
<td>[number]</td>
<td>The factor exaggerates the height of the output elevation raster</td>
</tr>
<tr>
<td>Scale (ratio of vert. units to horiz.)</td>
<td>SCALE</td>
<td>[number]</td>
<td>The ratio of vertical units to horizontal units</td>
</tr>
<tr>
<td>Azimuth of the light</td>
<td>AZIMUTH</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Altitude of the light</td>
<td>ALTITUDE</td>
<td>[number]</td>
<td>Defines the altitude of the light, in degrees. 90 if the light comes from above the elevation raster, 0 if it is raking light.</td>
</tr>
<tr>
<td>Compute edges</td>
<td>COMPUTE_EDGES</td>
<td>[boolean]</td>
<td>Generates edges from the elevation raster</td>
</tr>
<tr>
<td>Use Zevenbergen&amp;Thorne formula (instead of the Horn’s one)</td>
<td>ZEVENBERGEN</td>
<td>[boolean]</td>
<td>Activates Zevenbergen&amp;Thorne formula for smooth landscapes</td>
</tr>
<tr>
<td>Combined shading</td>
<td>COMBINED</td>
<td>[boolean]</td>
<td></td>
</tr>
<tr>
<td>Multidirectional shading</td>
<td>MULTIDIRECTION</td>
<td>[boolean]</td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hillshade</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer with interpolated values. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hillshade</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster with interpolated values</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: gdal:hillshade

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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 air photos so that color pixels can be treated as transparent when mosaicking.

This algorithm is derived from the GDAL nearblack utility.

Default menu: Raster Analysis

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input Elevation raster layer</td>
</tr>
<tr>
<td>How far from black (white)</td>
<td>NEAR</td>
<td>[number]</td>
<td>Default: 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Select how far from black, white or custom colors the pixel values can be and still considered near black, white or custom color.</td>
</tr>
<tr>
<td>Search for nearly white pixels instead of nearly black</td>
<td>WHITE</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Search for nearly white (255) pixels instead of nearly black pixels</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Default: ';'</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Default: None</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Nearblack</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Default: [Save to temporary file]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specify the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearblack</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:nearblack

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input Elevation raster layer</td>
</tr>
<tr>
<td><strong>Band number</strong></td>
<td>BAND</td>
<td>[raster band]</td>
<td>Band containing the elevation information</td>
</tr>
<tr>
<td><strong>A list of pixel values in the source image to be considered target pixels</strong></td>
<td>VALUES</td>
<td>[string]</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Distance units</strong></td>
<td>UNITS</td>
<td>[enumeration]</td>
<td>Indicate whether distances generated should be in pixel or georeferenced coordinates. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Georeferenced coordinates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Pixel coordinates</td>
</tr>
<tr>
<td><strong>The maximum distance to be generated</strong></td>
<td>MAX_DISTANCE</td>
<td>[number]</td>
<td>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.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Tabulka 23.129 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value to be applied to all pixels that are within the maxdist of target pixels</td>
<td>REPLACE</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td>NODATA</td>
<td>[number]</td>
<td>Specify the nodata value to use for the output raster</td>
</tr>
<tr>
<td>Optional</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Optional</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
<tr>
<td>Proximity map</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proximity map</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** gdal:proximity

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

### 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

#### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input elevation raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The number of the band to use as elevation</td>
</tr>
<tr>
<td>Compute edges</td>
<td>COMPUTE_EDGES</td>
<td>[boolean]</td>
<td>Generates edges from the elevation raster</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Optional</td>
<td>OPTIONS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roughness</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

#### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roughness</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Single-band output roughness raster. The value -9999 is used as nodata value.</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: gdal:roughness

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input elevation raster layer</td>
</tr>
<tr>
<td>Threshold</td>
<td>THRESHOLD</td>
<td>[number]</td>
<td>Only raster polygons smaller than this size will be removed</td>
</tr>
<tr>
<td>Use 8-connectedness</td>
<td>EIGHT_CONNECTED</td>
<td>[boolean]</td>
<td>Use eight connectedness instead of four connectedness</td>
</tr>
<tr>
<td>Do not use the default validity mask for the input band</td>
<td>NO_MASK</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Validity mask</td>
<td>MASK_LAYER</td>
<td>[raster]</td>
<td>Validity mask to use instead of the default</td>
</tr>
<tr>
<td>Optional Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Sieved</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sieved</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer.</td>
</tr>
</tbody>
</table>
**Python code**

**Algorithm ID:** `gdal:sieve`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Slope**

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*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input Elevation raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>Band containing the elevation information</td>
</tr>
<tr>
<td>Ratio of vertical units to horizontal</td>
<td>SCALE</td>
<td>[number]</td>
<td>The ratio of vertical units to horizontal units</td>
</tr>
<tr>
<td>Default: 1</td>
<td>Default: 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slope expressed as percent (instead of degrees)</td>
<td>AS_PERCENT</td>
<td>[boolean]</td>
<td>Express slope as percent instead of degrees</td>
</tr>
<tr>
<td>Default: False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compute edges</td>
<td>COMPUTE_EDGES</td>
<td>[boolean]</td>
<td>Generates edges from the elevation raster</td>
</tr>
<tr>
<td>Default: False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use Zevenbergen&amp;Thorne formula (instead of the Horn's one)</td>
<td>ZEVENBERGEN</td>
<td>[boolean]</td>
<td>Activates Zevenbergen&amp;Thorne formula for smooth landscapes</td>
</tr>
<tr>
<td>Default: False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see <em>GDAL driver options section</em>).</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td></td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Slope</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer. One of:</td>
</tr>
<tr>
<td>Default: [Save to temporary file]</td>
<td></td>
<td></td>
<td>- Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slope</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:slope

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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 [Raster Analysis]

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input elevation raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The number of the band to use as elevation</td>
</tr>
<tr>
<td>Compute edges</td>
<td>COMPUTE_EDGES</td>
<td>[boolean]</td>
<td>Generates edges from the elevation raster</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
</tbody>
</table>
| Terrain Ruggedness Index     | OUTPUT | [raster] | Specify the output raster layer. One of:  
|                              |       |                  | • Save to a Temporary File  
|                              |       |                  | • Save to File...  
|                              |       |                  | The file encoding can also be changed here.        |

23.2. GDAL algorithm provider 959
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terrain Ruggedness Index</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output ruggedness raster. The value -9999 is used as nodata value.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: `gdal:triterrainruggednessindex`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input elevation raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>The number of the band to use for elevation values</td>
</tr>
<tr>
<td>Compute edges</td>
<td>COMPUTE_EDGES</td>
<td>[boolean]</td>
<td>Generates edges from the elevation raster</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see <em>GDAL driver options section</em>).</td>
</tr>
<tr>
<td>Terrain Ruggedness Index</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terrain Ruggedness Index</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:tpitopographicpositionindex

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.2.2 Raster conversion

**gdal2xyz**

Converts raster data to XYZ ASCII file format.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Raster layer to convert</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose the band you want to convert</td>
</tr>
<tr>
<td>Output comma-separated values</td>
<td>CSV</td>
<td>[boolean]</td>
<td>Sets whether the output file should be of type comma-separated values (csv).</td>
</tr>
<tr>
<td>XYZ ASCII file</td>
<td>OUTPUT</td>
<td>[file]</td>
<td>Specification of the output file. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ ASCII file</td>
<td>INPUT</td>
<td>[table]</td>
<td>Table file containing the values exported from the raster band.</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** `gdal:gdal2xyz`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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*

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input 8 bit raster image</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose the band you want to convert</td>
</tr>
<tr>
<td>Generate a RGBA file</td>
<td>RGBA</td>
<td>[boolean]</td>
<td>Sets whether the output file should be of type RGBA.</td>
</tr>
<tr>
<td>PCT to RGB</td>
<td>OUTPUT</td>
<td>[file]</td>
<td>Specification of the output file. One of: • Save to a Temporary File • Save to File… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCT to RGB</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>24 bit RGB raster image</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** `gdal:pcttorgb`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>If the raster is multiband, choose the band you want to use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: The first band of the input layer</td>
<td></td>
</tr>
<tr>
<td>Name of the field to create</td>
<td>FIELD</td>
<td>[string]</td>
<td>Specify the field name for the attributes of the connected regions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: ‘DN’</td>
<td></td>
</tr>
<tr>
<td>Use 8-connectedness</td>
<td>EIGHT_CONNECTED</td>
<td>[boolean]</td>
<td>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).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Vectorized</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specification of the output (polygon) vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vectorized</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Output vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:polygonize

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Selected band(s)</td>
<td>[raster band]</td>
<td>Ordered list of the bands to use to create the new raster</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Output data type</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0 — Use Input Layer Data Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 4 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 5 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 6 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 7 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 8 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 9 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 10 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 11 — CFloat64</td>
</tr>
<tr>
<td>Converted</td>
<td>[raster]</td>
<td>Specification of the output raster. One of:</td>
</tr>
<tr>
<td></td>
<td>Default: Save to temporary file</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converted</td>
<td>[raster]</td>
<td>Output raster layer with rearranged bands.</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: gdal:rearrange_bands

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input (RGB) raster layer</td>
</tr>
<tr>
<td>Number of colors</td>
<td>NCOLORS</td>
<td>[number]</td>
<td>The number of colors the resulting image will contain. A value from 2-256 is possible.</td>
</tr>
<tr>
<td>RGB to PCT</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specification of the output raster. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB to PCT</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:rgbtopct

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
**Translate (convert format)**

Converts raster data between different formats.

This algorithm is derived from the GDAL `translate` utility.

**Default menu:** Raster Conversion

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Override the projection of the output file</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td>Specify a projection for the output file</td>
</tr>
<tr>
<td>Assign a specified nodata value to output bands</td>
<td>NODATA</td>
<td>[number]</td>
<td>Defines the value to use for nodata in the output raster</td>
</tr>
<tr>
<td>Copy all subdatasets of this file to individual output files</td>
<td>COPY_SUBDATASET</td>
<td>[boolean]</td>
<td>Create individual files for subdatasets</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the data type of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 — Use Input Layer Data Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11 — CFloat64</td>
</tr>
<tr>
<td>Converted</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specification of the output (translated) raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converted</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output (translated) raster layer.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:translate

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.2.3 Raster extraction

**Clip raster by extent**

Clips any GDAL-supported raster file to a given extent.

This algorithm is derived from the GDAL Grid utility.

**Default menu:** Raster Extraction

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>The input raster</td>
</tr>
<tr>
<td>Clipping extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Extent that should be used for the output raster. Only pixels within the specified bounding box will be included in the output.</td>
</tr>
<tr>
<td>Assign a specified nodata value to output bands</td>
<td>NODATA</td>
<td>[number]</td>
<td>Default: None Defines a value that should be inserted for the nodata values in the output raster</td>
</tr>
<tr>
<td>Optional</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Output data type

<table>
<thead>
<tr>
<th>Label</th>
<th>Data Type</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the format of the output raster file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Use Input Layer Data Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — CFloat64</td>
</tr>
</tbody>
</table>

### Additional command-line parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Clipped (extent)

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clipped (extent)</td>
<td>OUTPUT</td>
<td>Specification of the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The file encoding can also be changed here</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clipped (extent)</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer clipped by the given extent</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `gdal:cliprasterbyextent`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
Clip raster by mask layer

Clips any GDAL-supported raster by a vector mask layer. This algorithm is derived from the GDAL grid utility.

Default menu: Raster \(\square\) Extraction

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>The input raster</td>
</tr>
<tr>
<td>Mask layer</td>
<td>EXTENT</td>
<td>[vector: polygon]</td>
<td>Vector mask for clipping the raster</td>
</tr>
<tr>
<td>Source CRS</td>
<td>SOURCE_CRS</td>
<td>[crs]</td>
<td></td>
</tr>
<tr>
<td>Target CRS</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td></td>
</tr>
<tr>
<td>Assign a specified nodata va-</td>
<td>NODATA</td>
<td>[number]</td>
<td>Defines a value that should be inserted for the nodata values in</td>
</tr>
<tr>
<td>lue to output bands</td>
<td></td>
<td>Default: None</td>
<td>the output raster</td>
</tr>
<tr>
<td>Create an output alpha band</td>
<td>ALPHA_BAND</td>
<td>[boolean]</td>
<td>Creates an alpha band for the result. The alpha band then includes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td>the transparency values of the pixels.</td>
</tr>
<tr>
<td>Match the extent of the clip-</td>
<td>CROP_TO_CUTLINE</td>
<td>[boolean]</td>
<td>Applies the vector layer extent to the output raster if checked.</td>
</tr>
<tr>
<td>ped raster to the extent of</td>
<td></td>
<td>Default: True</td>
<td></td>
</tr>
<tr>
<td>the mask layer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keep resolution of input ra-</td>
<td>KEEP_RESOLUTION</td>
<td>[boolean]</td>
<td>The resolution of the output raster will not be changed</td>
</tr>
<tr>
<td>ster</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Set output file resolution</td>
<td>SET_RESOLUTION</td>
<td>[boolean]</td>
<td>Shall the output resolution (cell size) be specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>X Resolution to output ban-</td>
<td>X_RESOLUTION</td>
<td>[number]</td>
<td>The width of the cells in the output raster</td>
</tr>
<tr>
<td>ds</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Y Resolution to output ban-</td>
<td>Y_RESOLUTION</td>
<td>[number]</td>
<td>The height of the cells in the output raster</td>
</tr>
<tr>
<td>ds</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Use multithreaded warping</td>
<td>MULTITHREADING</td>
<td>[boolean]</td>
<td></td>
</tr>
<tr>
<td>implementation</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ' '</td>
<td>be created (colors, block size, file compression…). For conveni-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ence, you can rely on prede-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fined profiles (see GDAL driver options section).</td>
</tr>
</tbody>
</table>

Pokračujte na daší stránce
### Output data type

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DATA_TYPE</strong></td>
<td><strong>[enumeration]</strong></td>
<td><strong>Default: 0</strong></td>
<td>Defines the format of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Use Input Layer Data Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — CFloat64</td>
</tr>
</tbody>
</table>

### Additional command-line parameters

| Optional          | **EXTRA** | [string] | Default: None | Add extra GDAL command line options |

### Clipped (mask)

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th>[raster]</th>
<th>Default: [Save to temporary file]</th>
<th>Specification of the output raster layer. One of:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OUTPUT</strong></td>
<td><strong>[raster]</strong></td>
<td></td>
<td>Output raster layer clipped by the vector layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID**: `gdal:cliprasterbymasklayer`

```python
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 [Using processing algorithms from the console](#) for details on how to run processing algorithms from the Python console.
## Contour

Extracts contour lines from any GDAL-supported elevation raster.

This algorithm is derived from the GDAL contour utility.

**Default menu:** *Raster ➤ Extraction*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster</td>
</tr>
<tr>
<td>Band number</td>
<td>BAND</td>
<td>[raster band]</td>
<td>Raster band to create the contours from</td>
</tr>
<tr>
<td>Interval between contour lines</td>
<td>INTERVAL</td>
<td>[number]</td>
<td>Defines the interval between the contour lines in the given units of the elevation raster (minimum value 0)</td>
</tr>
<tr>
<td>Attribute name (if not set, no elevation attribute is attached) Optional</td>
<td>FIELD_NAME</td>
<td>[string]</td>
<td>Defines the attribute name for the field containing the values of the contour lines.</td>
</tr>
<tr>
<td>Produce 3D vector</td>
<td>CREATE_3D</td>
<td>[boolean]</td>
<td>Forces production of 3D vectors instead of 2D. Includes elevation at every vertex.</td>
</tr>
<tr>
<td>Treat all raster values as valid</td>
<td>IGNORE_NODATA</td>
<td>[boolean]</td>
<td>Ignores any nodata values in the dataset.</td>
</tr>
<tr>
<td>Input pixel value to treat as „nodata“ Optional</td>
<td>NODATA</td>
<td>[number]</td>
<td>Defines a value that should be inserted for the nodata values in the output raster</td>
</tr>
<tr>
<td>Offset from zero relative to which to interpret intervals Optional</td>
<td>OFFSET</td>
<td>[number]</td>
<td>Default: 0.0</td>
</tr>
<tr>
<td>Additional command-line parameters Optional</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Additional creation options Optional</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Contours</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specification of the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: <code>{Save to temporary file}</code></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contours</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Output vector layer with contour lines</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:contour

```python
import processing
goingal.contour
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

### 23.2.4 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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Overview levels</td>
<td>LEVELS</td>
<td>[string]</td>
<td>Defines the number of overview levels cal-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>culated by the original resolution of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>input raster layer. By default 4 levels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>will be taken into consideration.</td>
</tr>
<tr>
<td>Remove all existing</td>
<td>CLEAN</td>
<td>[boolean]</td>
<td>Removes existing overviews from the ras-</td>
</tr>
<tr>
<td>overviews</td>
<td></td>
<td></td>
<td>ter. By default these are not removed.</td>
</tr>
<tr>
<td>Resampling method</td>
<td>RESAMPLING</td>
<td>[enumeration]</td>
<td>Calculates the overviews with a defined re-</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>sampling method. Possible resampling me-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>thods are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 – Nearest Neighbour (nearest)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 – Average (average)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 – Gaussian (gauss)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 – Cubic Convolution (cubic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 – B-Spline Convolution (cubicspline)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 – Lanczos Windowed Sinc (lanczos)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 – Average MP (average_mp)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 – Average in Mag/Phase Space (average_magphase)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 – Mode (mode)</td>
</tr>
</tbody>
</table>
Tabulka 23.144 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overviews format</td>
<td>FORMAT</td>
<td>[enumeration]</td>
<td>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)</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pyramidized</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pyramidized</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer with overviews</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:overviews

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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 \[ Miscellaneous \]

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layers</td>
<td>INPUT</td>
<td>[raster]</td>
<td>GDAL-supported raster layers.</td>
</tr>
<tr>
<td>Resolution</td>
<td>RESOLUTION</td>
<td>[enumeration]</td>
<td>The output resolution of the mosaic. By default the average resolution of the raster files will be chosen. Options: • 0 — Average (average) • 1 — Highest (highest) • 2 — Lowest (lowest)</td>
</tr>
</tbody>
</table>
### Place each input file into a separate band

**Label:** SEPARATE  
**Type:** [boolean]  
**Default:** True  
**Popis:** With 'True' you can define that each raster file goes into a separated stacked band in the VRT band.

### Allow projection difference

**Label:** PROJ_DIFFERENCE  
**Type:** [boolean]  
**Default:** False  
**Popis:** Allows that the output bands have different projections derived from the projection of the input raster layers.

### Add alpha mask band to VRT when source raster has none

**Label:** ADD_ALPHA  
**Type:** [boolean]  
**Default:** False  
**Popis:** Adds an alpha mask band to the VRT when the source raster has none.

### Override projection for the output file (optional)

**Label:** ASSIGN_CRS  
**Type:** [crs]  
**Default:** None  
**Popis:** Overrides the projection for the output file. No reprojection is done.

### Resampling algorithm

**Label:** RESAMPLING  
**Type:** [enumeration]  
**Default:** 0  
**Popis:** The resampling algorithm to be used. Options:  
- 0 — Nearest Neighbour (nearest)  
- 1 — Bilinear (bilinear)  
- 2 — Cubic Convolution (cubic)  
- 3 — B-Spline Convolution (cubicspline)  
- 4 — Lanczos Windowed Sinc (lanczos)  
- 5 — Average (average)  
- 6 — Mode (mode)

### Nodata value(s) for input bands (space separated) Optional

**Label:** SRC_NODATA  
**Type:** [string]  
**Default:** None  
**Popis:** Space separated Nodata value(s) for input band(s)

### Additional command-line parameters

**Label:** EXTRA  
**Type:** [string]  
**Default:** None  
**Popis:** Add extra GDAL command line options

### Virtual

**Label:** OUTPUT  
**Type:** [raster]  
**Default:** [Save to temporary file]  
**Popis:** Specification of the output raster layer. One of:  
- Save to a Temporary File  
- Save to File...  
The file encoding can also be changed here.

## Outputs

### Virtual

**Label:** OUTPUT  
**Type:** [raster]  
**Popis:** Output raster layer
Python code

Algorithm ID: gdal:buildvirtualraster

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>GDAL-supported raster layer.</td>
</tr>
<tr>
<td>Tile cutting profile</td>
<td>PROFILE</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Mercator (mercator)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Geodetic (geodetic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Raster (raster)</td>
</tr>
<tr>
<td>Zoom levels to render</td>
<td>ZOOM</td>
<td>[string]</td>
<td>Default: ‘’</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web viewer to generate</td>
<td>VIEWER</td>
<td>[enumerate]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — All (all)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — GoogleMaps (google)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — OpenLayers (openlayers)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Leaflet (leaflet)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — None (none)</td>
</tr>
<tr>
<td>Title of the map</td>
<td>TITLE</td>
<td>[string]</td>
<td>Default: ‘’</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copyright of the map</td>
<td>COPYRIGHT</td>
<td>[string]</td>
<td>Default: ‘’</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Resampling method

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resampling method</strong></td>
<td>RESAMPLING</td>
<td>[enumeration] 0</td>
<td>The resampling algorithm to be used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Average (average)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Nearest neighbour (near)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Bilinear (bilinear)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Cubic (cubic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Cubic spline (cubicspline)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Lanczos Windowed sinc (lanczos)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Antialias (antialias)</td>
</tr>
<tr>
<td>The spatial reference system used for the source input data</td>
<td>SOURCE_CRS</td>
<td>[crs] Default: None</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transparency value to assign to the input data</td>
<td>NODATA</td>
<td>[number] Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL address where the generated tiles are going to be published</td>
<td>URL</td>
<td>[string] Default:</td>
<td>Your Google maps API key.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>Google Maps API key</td>
<td>GOOGLE_KEY</td>
<td>[string] Default:</td>
<td>Your Bing maps API key.</td>
</tr>
<tr>
<td>(<a href="http://code.google.com/apis/maps/signup.html">http://code.google.com/apis/maps/signup.html</a>)</td>
<td></td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bing Maps API key</td>
<td>BING_KEY</td>
<td>[string] Default:</td>
<td>Your Bing maps API key.</td>
</tr>
<tr>
<td>(<a href="https://www.bingmapsportal.com/">https://www.bingmapsportal.com/</a>)</td>
<td></td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate only missing files</td>
<td>RESUME</td>
<td>[boolean] Default: False</td>
<td></td>
</tr>
<tr>
<td>Generate KML for Google Earth</td>
<td>KML</td>
<td>[boolean] Default: False</td>
<td></td>
</tr>
<tr>
<td>Avoid automatic generation of KML files for EPSG:4326</td>
<td>NO_KML</td>
<td>[boolean] Default: False</td>
<td></td>
</tr>
<tr>
<td>Output directory</td>
<td>OUTPUT</td>
<td>[folder] Default:</td>
<td>Specify the output folder for the tiles.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Save to temporary file]</td>
<td></td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output directory</td>
<td>OUTPUT</td>
<td>[folder]</td>
<td>The output folder (for the tiles)</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:gdal2tiles

```
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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]

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layers</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layers</td>
</tr>
<tr>
<td>Grab pseudocolor table from first layer</td>
<td>PCT</td>
<td>[boolean]</td>
<td>The pseudocolor table from the first layer will be used for the coloring</td>
</tr>
<tr>
<td></td>
<td>SEPARATE</td>
<td>[boolean]</td>
<td>Place each input file into a separate band</td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the format of the output raster file. Options:</td>
</tr>
<tr>
<td>Input pixel value to treat as „noda-</td>
<td>NODATA_</td>
<td>[number]</td>
<td>Ignores pixels from files being merged in</td>
</tr>
<tr>
<td>ta“ Optional</td>
<td>_INPUT</td>
<td></td>
<td>with this pixel value</td>
</tr>
</tbody>
</table>
Tabulka 23.147 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign specified „nodata“ value to output</td>
</tr>
<tr>
<td>Optional</td>
</tr>
<tr>
<td>NODATA_OUTPUT</td>
</tr>
<tr>
<td>[number]</td>
</tr>
<tr>
<td>Default: None</td>
</tr>
<tr>
<td>Assigns the specified nodata value to output bands.</td>
</tr>
<tr>
<td>Additional creation options</td>
</tr>
<tr>
<td>Optional</td>
</tr>
<tr>
<td>OPTIONS</td>
</tr>
<tr>
<td>[string]</td>
</tr>
<tr>
<td>Default: ;</td>
</tr>
<tr>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
</tr>
<tr>
<td>EXTRA</td>
</tr>
<tr>
<td>[string]</td>
</tr>
<tr>
<td>Default: None</td>
</tr>
<tr>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Merged</td>
</tr>
<tr>
<td>OUTPUT</td>
</tr>
<tr>
<td>[raster]</td>
</tr>
<tr>
<td>Default: [Save to temporary file]</td>
</tr>
<tr>
<td>Specification of the output raster layer. One of:</td>
</tr>
<tr>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td>• Save to File...</td>
</tr>
<tr>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merged</td>
</tr>
<tr>
<td>OUTPUT</td>
</tr>
<tr>
<td>[raster]</td>
</tr>
<tr>
<td>Output raster layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:merge

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectral dataset</td>
</tr>
<tr>
<td>SPECTRAL</td>
</tr>
<tr>
<td>[raster]</td>
</tr>
<tr>
<td>Input (spectral) raster layer</td>
</tr>
<tr>
<td>Panchromatic dataset</td>
</tr>
<tr>
<td>PANCHROMATIC</td>
</tr>
<tr>
<td>[raster]</td>
</tr>
<tr>
<td>Input (panchromatic) raster layer</td>
</tr>
</tbody>
</table>
### Resampling algorithm

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resampling algorithm</strong></td>
<td>RESAMPLING</td>
<td>[enumeration]</td>
<td>The resampling algorithm to be used Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 — Nearest Neighbour (nearest)</td>
<td>For adding one or more creation options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 — Bilinear (bilinear)</td>
<td>that control the raster to be created (colors,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 — Cubic (cubic)</td>
<td>block size, file compression...). For convenience, you can rely</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 — Cubic Spline (cubicspline)</td>
<td>on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 — Lanczos Windowed Sinc (lanczos)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 — Average (average)</td>
<td></td>
</tr>
<tr>
<td><strong>Additional creation options</strong></td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>that control the raster to be created (colors, block size, file</td>
</tr>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>compression...). For convenience, you can rely on predefined profiles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td>(see GDAL driver options section).</td>
</tr>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specify the output (sharpened) raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 — Save to Temporary File</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 — Save to File...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
<td></td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output (sharpened) raster layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** gdal:pansharp

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### Raster calculator

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.

**Viz také:**

*Raster calculator*
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer A</td>
<td>INPUT_A</td>
<td>[raster]</td>
<td>First input raster layer (mandatory)</td>
</tr>
<tr>
<td>Number of raster band for A</td>
<td>BAND_A</td>
<td>[raster band]</td>
<td>Band for input layer A (mandatory)</td>
</tr>
<tr>
<td>Input layer B</td>
<td>INPUT_B</td>
<td>[raster]</td>
<td>Second input raster layer</td>
</tr>
<tr>
<td>Number of raster band for B</td>
<td>BAND_B</td>
<td>[raster band]</td>
<td>Band for input layer B</td>
</tr>
<tr>
<td>Input layer C</td>
<td>INPUT_C</td>
<td>[raster]</td>
<td>Third input raster layer</td>
</tr>
<tr>
<td>Number of raster band for C</td>
<td>BAND_C</td>
<td>[raster band]</td>
<td>Band for input layer C</td>
</tr>
<tr>
<td>Input layer D</td>
<td>INPUT_D</td>
<td>[raster]</td>
<td>Fourth input raster layer</td>
</tr>
<tr>
<td>Number of raster band for D</td>
<td>BAND_D</td>
<td>[raster band]</td>
<td>Band for input layer D</td>
</tr>
<tr>
<td>Input layer E</td>
<td>INPUT_E</td>
<td>[raster]</td>
<td>Fifth input raster layer</td>
</tr>
<tr>
<td>Number of raster band for E</td>
<td>BAND_E</td>
<td>[raster band]</td>
<td>Band for input layer E</td>
</tr>
<tr>
<td>Input layer F</td>
<td>INPUT_F</td>
<td>[raster]</td>
<td>Sixth input raster layer</td>
</tr>
<tr>
<td>Number of raster band for F</td>
<td>BAND_F</td>
<td>[raster band]</td>
<td>Band for input layer F</td>
</tr>
</tbody>
</table>
| Calculation in gdalnumeric syntax using +/-/* or any numpy array functions (i.e. logical_and()) | FORMULA     | [string]      | The calculation formula. Examples:  
  • A*(A>0) — outputs the value of the raster A if the value of A is greater than 0. If not, outputs 0.  
  • A*(A>0 and A>B) — outputs the value of A if that value is bigger than 0 and bigger than the value of B. If not, outputs 0.  
  • A*logical_or(A<=177, A>=185) — outputs the value of A if A <= 177 or A >= 185. If not, outputs 0.  
  • sqrt(A*A+B*B) — Outputs the square root of the sum of the value of A squared and the value of B squared. |
| Set output nodata value      | NO_DATA     | [number]      | Value to use for nodata                                 |

Pokračujte na další stránce
### Tabulka 23.149 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output raster type</strong></td>
<td><strong>RTYPE</strong></td>
<td>[enumeration]</td>
<td>Defines the format of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td><strong>Additional creation options</strong></td>
<td><strong>OPTIONS</strong></td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td>(colors, block size, file compression...). For convenience, you can rely on predefined profiles</td>
</tr>
<tr>
<td><strong>Additional command-line parameters</strong></td>
<td><strong>EXTRA</strong></td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td></td>
</tr>
<tr>
<td><strong>Calculated</strong></td>
<td><strong>OUTPUT</strong></td>
<td>[raster]</td>
<td>Specify the output (calculated) raster layer. One of:</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: [Save</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to temporary file]</td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Calculated</strong></td>
<td><strong>OUTPUT</strong></td>
<td>[raster]</td>
<td>Output (calculated) raster layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `gdal:rastercalculator`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Force computation of the actual min/max values for each band</td>
<td>MIN_MAX</td>
<td>[boolean]</td>
<td>Forces computation of the actual min/max values for each band in the dataset</td>
</tr>
<tr>
<td>Read and display image statistics (force computation if necessary)</td>
<td>STATS</td>
<td>[boolean]</td>
<td>Reads and displays image statistics. Forces computation if no statistics are stored in an image.</td>
</tr>
<tr>
<td>Suppress GCP info</td>
<td>NO_GCP</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Suppress metadata info</td>
<td>NO_METADATA</td>
<td>[boolean]</td>
<td>Suppresses metadata printing. Some datasets may contain a lot of metadata strings.</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Layer information</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the HTML file for output. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer information</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>The HTML file containing information about the input raster layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `gdal:gdalinfo`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input files</td>
<td>INPUT</td>
<td>[raster] [list]</td>
<td>The input raster files</td>
</tr>
<tr>
<td>Tile width</td>
<td>TILE_SIZE_X</td>
<td>[number]</td>
<td>Default: 256 Width of the tiles in pixels (minimum 0)</td>
</tr>
<tr>
<td>Tile height</td>
<td>TILE_SIZE_Y</td>
<td>[number]</td>
<td>Default: 256 Height of the tiles in pixels (minimum 0)</td>
</tr>
<tr>
<td>Overlap in pixels between consecutive tiles</td>
<td>OVERLAP</td>
<td>[number]</td>
<td>Default: 0</td>
</tr>
<tr>
<td>Number of pyramid levels to build</td>
<td>LEVELS</td>
<td>[number]</td>
<td>Default: 1 Minimum: 0</td>
</tr>
<tr>
<td>Source coordinate reference system</td>
<td>SOURCE_CRS</td>
<td>[crs]</td>
<td>Default: None</td>
</tr>
<tr>
<td>Resampling method</td>
<td>RESAMPLING</td>
<td>[enumeration]</td>
<td>Default: 0 The resampling algorithm to be used Options: 0 — Nearest Neighbour (nearest) 1 — Bilinear (bilinear) 2 — Cubic (cubic) 3 — Cubic Spline (cubicspline) 4 — Lanczos Windowed Sinc (lanczos)</td>
</tr>
<tr>
<td>Column delimiter used in the CSV file</td>
<td>DELIMITER</td>
<td>[string]</td>
<td>Default: ; Delimiter to use in the CSV file containing the tile(s) georeferencing information</td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Default: ‘\’ For adding one or more creation options that control the raster to be created (colors, block size, file compression…). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Additional command-line parameters</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Default: ‘\’ Add extra GDAL command line options</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.151 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the format of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 5</td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
<tr>
<td>Build only the pyramids</td>
<td>ONLY_PYRAMIDS</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Use separate directory for each tile row</td>
<td>DIR_FOR_ROW</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Output directory</td>
<td>OUTPUT</td>
<td>[folder]</td>
<td>Specify the output folder for the tiles. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary folder]</td>
<td>• Save to Temporary Directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Directory...</td>
</tr>
<tr>
<td>CSV file containing the tile(s) georeferencing information</td>
<td>OUTPUT_CSV</td>
<td>[file]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Skip output]</td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output directory</td>
<td>OUTPUT</td>
<td>[folder]</td>
<td>The output folder for the tiles.</td>
</tr>
<tr>
<td>CSV file containing the tile(s) georeferencing information</td>
<td>OUTPUT_CSV</td>
<td>[file]</td>
<td>The CSV file with georeferencing information for the tiles.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:retile

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
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 tile index.

This algorithm is derived from the GDAL Tile Index utility.

**Default menu:** *Raster ➤ Miscellaneous*

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input files</td>
<td>LAYERS</td>
<td>[raster] [list]</td>
<td>The input raster files. Can be multiple files.</td>
</tr>
<tr>
<td>Field name to hold the file path to the indexed rasters</td>
<td>PATH_FIELD_NAME</td>
<td>[string]</td>
<td>The output field name to hold the file path/location to the indexed rasters.</td>
</tr>
<tr>
<td>Optional</td>
<td>Default: 'location'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store absolute path to the indexed rasters</td>
<td>ABSOLUTE_PATH</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td>Default: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skip files with different projection reference</td>
<td>PROJ_DIFFERENCE</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td>Default: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transform geometries to the given CRS</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The name of the field to store the SRS of each tile</td>
<td>CRS_FIELD_NAME</td>
<td>[string]</td>
<td>The name of the field to store the SRS of each tile</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The format in which the CRS of each tile must be written</td>
<td>CRS_FORMAT</td>
<td>[enumeration]</td>
<td>Format for the CRS. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 – Auto (AUTO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 – Well-known text (WKT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 – EPSG (EPSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 – Proj.4 (PROJ)</td>
</tr>
<tr>
<td>Tile index</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the polygon vector layer to write the index to. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile index</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The polygon vector layer with the tile index.</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:tileindex

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.2.5 Raster projections

Assign projection

Applies a coordinate system to a raster dataset.

This algorithm is derived from the GDAL edit utility.

**Default menu:** Raster → Projections

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT_LAYER</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Desired CRS</td>
<td>CRS</td>
<td>[crs]</td>
<td>The projection (CRS) of the output layer</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer with projection</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The output raster layer (with the new projection information)</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:assignprojection

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Extract projection

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

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input file</td>
<td>INPUT_LAYER</td>
<td>[raster]</td>
<td>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.</td>
</tr>
<tr>
<td>Create also .prj file</td>
<td>PRJ_FILE_CREATE</td>
<td>[boolean]</td>
<td>Default: False. If this is activated a .prj file containing the projection information is also created.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>World file</td>
<td>WORLD_FILE</td>
<td>[file]</td>
<td>Text file with extension .wld containing transformation parameters for the raster file.</td>
</tr>
<tr>
<td>ESRI Shapefile prj file</td>
<td>PRJ_FILE</td>
<td>[file]</td>
<td>Text file with .prj extension that describes the CRS. Will be None if Create also .prj file is False.</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:extractprojection

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Warp (reproject)

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
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input layer</strong></td>
<td>INPUT</td>
<td>[raster]</td>
<td>Input raster layer to reproject</td>
</tr>
<tr>
<td><strong>Source CRS</strong> Optional</td>
<td>SOURCE_CRS</td>
<td>[crs]</td>
<td>Defines the CRS of the input raster layer</td>
</tr>
<tr>
<td><strong>Target CRS</strong> Optional</td>
<td>TARGET_CRS</td>
<td>[crs]</td>
<td>The CRS of the output layer</td>
</tr>
<tr>
<td><strong>Resampling method to use</strong></td>
<td>RESAMPLING</td>
<td>[enumeration]</td>
<td>Pixel value resampling method to use. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Nearest neighbour</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Bilinear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Cubic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — Cubic spline</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Lanczos windowed sinc</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Average</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Maximum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — Minimum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — Median</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — First quartile</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — Third quartile</td>
</tr>
<tr>
<td><strong>Nodata value for output bands</strong></td>
<td>NODATA</td>
<td>[number]</td>
<td>Sets nodata value for output bands. If not provided, then nodata values will be copied from the source dataset.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Output file resolution in target georeferenced units</strong> Optional</td>
<td>TARGET_RESOLUTION</td>
<td>[number]</td>
<td>Defines the output file resolution of reprojec</td>
</tr>
<tr>
<td><strong>Additional creation options</strong></td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Output data type</strong></td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the format of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Use input layer data type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — CFloat64</td>
</tr>
<tr>
<td><strong>Georeferenced extents of output file to be created</strong> Optional</td>
<td>TARGET_EXTENT</td>
<td>[extent]</td>
<td>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).</td>
</tr>
</tbody>
</table>
### CRS of the target raster extent

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRS of the target raster extent</td>
<td>TARGET_EXTENT_CRS</td>
<td>[crs]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Use multithreaded warping implementation | MULTITHREADING | [boolean] | Default: 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 | EXTRA | [string] | Default: None | Add extra GDAL command line options. |

| Reprojected | OUTPUT | [raster] | Default: [Save to temporary file] | Specification of the output raster layer. One of:  
  - Save to a Temporary File  
  - Save to File…  
  The file encoding can also be changed here. |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reprojected</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Reprojected output raster layer</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** gdal:warpreproject

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.2.6 Vector conversion

**Convert format**

Converts any OGR-supported vector layer into another OGR-supported format.

This algorithm is derived from the ogr2ogr utility.
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Additional GDAL creation options.</td>
</tr>
<tr>
<td>Converted</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specification of the output vector layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here. 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.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converted</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output vector layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:convertformat

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Input raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>Field to use for a burn-in value</td>
<td>FIELD</td>
<td>[tablefield: numeric]</td>
<td>Defines the attribute field to use to set the pixels values</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.154 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add burn in values to existing raster values</td>
<td>ADD</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Additional command-line parameters Optional</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rasterized</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The overwritten input raster layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:rasterize_over

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Input raster layer</td>
<td>INPUT_RASTER</td>
<td>[raster]</td>
<td>Input raster layer</td>
</tr>
<tr>
<td>A fixed value to burn</td>
<td>BURN</td>
<td>[number]</td>
<td>The value to burn</td>
</tr>
<tr>
<td>Default: 0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add burn in values to existing raster values</td>
<td>ADD</td>
<td>[boolean]</td>
<td>If False, pixels are assigned the fixed value. If True, the fixed value is added to the value of the input raster layer.</td>
</tr>
<tr>
<td>Default: False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional command-line parameters Optional</td>
<td>EXTRA</td>
<td>[string]</td>
<td>Add extra GDAL command line options</td>
</tr>
<tr>
<td>Default: ;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rasterized</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>The overwritten input raster layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: `gdal:rasterize_over_fixed_value`

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**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**

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Field to use for a burn-in value</td>
<td>FIELD</td>
<td>[tablefield: numeric]</td>
<td>Defines the attribute field from which the attributes for the pixels should be chosen</td>
</tr>
<tr>
<td>A fixed value to burn</td>
<td>BURN</td>
<td>[number] Default: 0.0</td>
<td>A fixed value to burn into a band for all features.</td>
</tr>
<tr>
<td>Output raster size units</td>
<td>UNITS</td>
<td>[enumeration] Default: 0</td>
<td>Units to use when defining the output raster size/resolution. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Pixels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Georeferenced units</td>
</tr>
<tr>
<td>Width/Horizontal resolution</td>
<td>WIDTH</td>
<td>[number] Default: 0.0</td>
<td>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.</td>
</tr>
<tr>
<td>Height/Vertical resolution</td>
<td>HEIGHT</td>
<td>[number] Default: 0.0</td>
<td>Sets the height (if size units is „Pixels“) or vertical resolution (if size units is „Georeferenced units“) of the output raster.</td>
</tr>
<tr>
<td>Output extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>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.</td>
</tr>
</tbody>
</table>
## Tabulka 23.156 - pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign a specified nodata value to output bands</td>
<td>NODATA</td>
<td>[number]</td>
<td>Assigns a specified nodata value to output bands</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: 0.0</td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>For adding one or more creation options that control the raster to be created (colors, block size, file compression...). For convenience, you can rely on predefined profiles (see GDAL driver options section).</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td></td>
</tr>
<tr>
<td>Output data type</td>
<td>DATA_TYPE</td>
<td>[enumeration]</td>
<td>Defines the format of the output raster file. Options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — Int16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — UInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — UInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — Int32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — Float32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — Float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — CInt16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — CInt32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — CFloat32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — CFloat64</td>
</tr>
<tr>
<td>Pre-initialize the output image with value</td>
<td>INIT</td>
<td>[number]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invert rasterization</td>
<td>INVERT</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Rasterized</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Specification of the output raster layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here. 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.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rasterized</td>
<td>OUTPUT</td>
<td>[raster]</td>
<td>Output raster layer</td>
</tr>
</tbody>
</table>

23.2. GDAL algorithm provider
Python code

Algorithm ID: gdal:rasterize

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.2.7 Vector geoprocessing

Buffer vectors

Create buffers around the features of a vector layer.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer</td>
</tr>
<tr>
<td>Geometry column name</td>
<td>GEOMETRY</td>
<td>[string]</td>
<td>The name of the input layer geometry column to use</td>
</tr>
<tr>
<td>Buffer distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Minimum: 0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 10.0</td>
<td></td>
</tr>
<tr>
<td>Dissolve by attribute</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Field to use for dissolving</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>Dissolve results</td>
<td>DISSOLVE</td>
<td>[boolean]</td>
<td>If set, the result is dissolved. If no field is set for dissolving, all the buffers are dissolved into one feature.</td>
</tr>
<tr>
<td>Produce one feature for each geometry in any kind of geometry collection in the source file</td>
<td>EXPLODE_COLLECTION</td>
<td>[boolean]</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Additional GDAL creation options.</td>
</tr>
<tr>
<td>(optional)</td>
<td></td>
<td>Default: ‘’ (no additional options)</td>
<td></td>
</tr>
<tr>
<td>Buffer</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output buffer layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>
**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output buffer layer</td>
</tr>
</tbody>
</table>

**Python code**

Algorithm ID: gdal:buffervectors

```python
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 Using processing algorithms from the console 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 ogr2ogr utility.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer</td>
</tr>
<tr>
<td>Clip extent</td>
<td>EXTENT</td>
<td>[extent]</td>
<td>Defines the bounding box that should be used for the output vector file. It has to be defined in target CRS coordinates.</td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Default: '.' (no additional options)</td>
</tr>
<tr>
<td>Clipped (extent)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output (clipped) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: ['Save to temporary file']</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clipped (extent)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output (clipped) layer. The default format is „ESRI Shapefile“.</td>
</tr>
</tbody>
</table>
Python code

Algorithm ID: gdal:clipvectorbyextent

```python
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 Using processing algorithms from the console 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 ogr2ogr utility.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input vector layer</td>
</tr>
<tr>
<td>Mask layer</td>
<td>MASK</td>
<td>[vector: polygon]</td>
<td>Layer to be used as clipping extent for the input vector layer.</td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Default: '.' (no additional options) Additional GDAL creation options.</td>
</tr>
<tr>
<td>Clipped (mask)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output (masked) layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clipped (mask)</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output (masked) layer. The default format is „ESRI Shapefile“.</td>
</tr>
</tbody>
</table>
Dissolve

Dissolve (combine) geometries that have the same value for a given attribute / field. The output geometries are multipart.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>The input layer to dissolve</td>
</tr>
<tr>
<td>Dissolve field (optional)</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>The field of the input layer to use for dissolving</td>
</tr>
<tr>
<td>Geometry column name</td>
<td>GEOMETRY</td>
<td>[string]</td>
<td>The name of the input layer geometry column to use for dissolving.</td>
</tr>
<tr>
<td>Produce one feature for each geometry in any kind of geometry collection in the source file</td>
<td>EXPLODE_COLLECTION</td>
<td>[boolean]</td>
<td>Produce one feature for each geometry in any kind of geometry collection in the source file</td>
</tr>
<tr>
<td>Keep input attributes</td>
<td>KEEP_ATTRIBUTES</td>
<td>[boolean]</td>
<td>Keep all attributes from the input layer</td>
</tr>
<tr>
<td>Count dissolved features</td>
<td>COUNT_FEATURES</td>
<td>[boolean]</td>
<td>Count the dissolved features and include it in the output layer.</td>
</tr>
<tr>
<td>Compute area and perimeter of dissolved features</td>
<td>COMPUTE_AREA</td>
<td>[boolean]</td>
<td>Compute the area and perimeter of dissolved features and include them in the output layer.</td>
</tr>
<tr>
<td>Compute min/max/sum/mean for attribute</td>
<td>COMPUTE_STATISTICS</td>
<td>[boolean]</td>
<td>Calculate statistics (min, max, sum and mean) for the numeric attribute specified and include them in the output layer.</td>
</tr>
<tr>
<td>Numeric attribute to calculate statistics on (optional)</td>
<td>STATISTICS_ATTRIBUTE</td>
<td>[tablefield: numeric]</td>
<td>The numeric attribute to calculate statistics on.</td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Additional GDAL creation options.</td>
</tr>
<tr>
<td>Dissolved</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>Specify the output layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissolved</td>
<td>OUTPUT</td>
<td>[same as input]</td>
<td>The output multipart geometry layer (with dissolved geometries)</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** gdal:dissolve

```python
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 *Using processing algorithms from the console* 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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>The input line layer</td>
</tr>
<tr>
<td>Geometry column name</td>
<td>GEOMETRY</td>
<td>[string]</td>
<td>The name of the input layer geometry column to use</td>
</tr>
<tr>
<td>Offset distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Default: 10.0</td>
</tr>
<tr>
<td>(left-sided: positive, right-sided: negative)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional creation options</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Default: ' ' (no additional options)</td>
</tr>
<tr>
<td>(optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset curve</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>Specify the output line layer. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset curve</td>
<td>OUTPUT</td>
<td>[vector: line]</td>
<td>The output offset curve layer</td>
</tr>
</tbody>
</table>

Python code

**Algorithm ID:** gdal:offsetcurve

```python
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 *Using processing algorithms from the console* 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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>The input line layer</td>
</tr>
<tr>
<td>Geometry column name</td>
<td>GEOMETRY</td>
<td>[string]</td>
<td>Default: 'geometry' The name of the input layer geometry column to use</td>
</tr>
<tr>
<td>Buffer distance</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Default: 10.0</td>
</tr>
<tr>
<td>Buffer side</td>
<td>BUFFER_SIDE</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td>Dissolve by attribute</td>
<td>FIELD</td>
<td>[tablefield: any]</td>
<td>Default: None</td>
</tr>
<tr>
<td>Dissolve all results</td>
<td>DISSOLVE</td>
<td>[boolean]</td>
<td>If set, the result is dissolved. If no field is set for dissolving, all the buffers are dissolved into one feature.</td>
</tr>
<tr>
<td>Produce one feature for each geometry in any kind of geometry collection in the source file</td>
<td>EXPLODE_COLLECTION</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Default: '*' (no additional options)</td>
</tr>
<tr>
<td>One-sided buffer</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>Specify the output buffer layer. One of:</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-sided buffer</td>
<td>OUTPUT</td>
<td>[vector: polygon]</td>
<td>The output buffer layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:onesidebuffer

```python
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 *Using processing algorithms from the console* 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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: line]</td>
<td>The input line layer</td>
</tr>
<tr>
<td>Geometry column name</td>
<td>GEOMETRY</td>
<td>[string]</td>
<td>The name of the input layer geometry column to use</td>
</tr>
<tr>
<td>Distance from line start represented as a fraction of line length</td>
<td>DISTANCE</td>
<td>[number]</td>
<td>Default: 0.5 (middle of the line)</td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Default: ' (no additional options)</td>
</tr>
<tr>
<td>Points along line</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>Specify the output point layer. One of: * Save to a Temporary File * Save to File… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points along line</td>
<td>OUTPUT</td>
<td>[vector: point]</td>
<td>The output point layer</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: gdal:pointsalonglines

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.2.8 Vector miscellaneous

Execute 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.
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>OGR-supported input vector layer</td>
</tr>
<tr>
<td>SQL expression</td>
<td>SQL</td>
<td>[string]</td>
<td>Defines the SQL query, for example <code>SELECT * FROM my_table WHERE name is not null</code>.</td>
</tr>
</tbody>
</table>
| SQL dialect     | DIALECT | [enumeration] | Default: 0  
|                 |        |             | SQL dialect to use. One of:  
|                 |        |             | • 0 — None  
|                 |        |             | • 1 — OGR SQL  
|                 |        |             | • 2 — SQLite                                                        |
| Additional creation options (optional) | OPTIONS | [string]    | Default: ‘’ (no additional options)  
|                 |        |             | Additional GDAL creation options.                                    |
| SQL result      | OUTPUT | [vector: any] | Specification of the output layer. One of:  
|                 |        |             | • Save to a Temporary File  
|                 |        |             | • Save to File...  
|                 |        |             | The file encoding can also be changed here.  
|                 |        |             | 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. |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL result</td>
<td>OUTPUT</td>
<td>[vector: any]</td>
<td>Vector layer created by the query</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** `gdal:executesql`  

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 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.
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database (connection name)</td>
<td>DATABASE</td>
<td>[string]</td>
<td>The PostgreSQL database to connect to</td>
</tr>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>OGR-supported vector layer to export to the database</td>
</tr>
<tr>
<td>Shape encoding Optional</td>
<td>SHAPE_ENCODING</td>
<td>[string]</td>
<td>Default: '</td>
</tr>
<tr>
<td>Output geometry type</td>
<td>GTYPE</td>
<td>[enumeration]</td>
<td>Default: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assign an output CRS Optional</td>
<td>A_SRS</td>
<td>[crs]</td>
<td>Default: None</td>
</tr>
<tr>
<td>Reproject to this CRS on output Optional</td>
<td>T_SRS</td>
<td>[crs]</td>
<td>Default: None</td>
</tr>
<tr>
<td>Override source CRS Optional</td>
<td>S_SRS</td>
<td>[crs]</td>
<td>Default: None</td>
</tr>
<tr>
<td>Schema (schema name) Optional</td>
<td>SCHEMA</td>
<td>[string]</td>
<td>Default: 'public'</td>
</tr>
<tr>
<td>Table to export to (leave blank to use layer name) Optional</td>
<td>TABLE</td>
<td>[string]</td>
<td>Default: '</td>
</tr>
<tr>
<td>Primary Key (new field) Optional</td>
<td>PK</td>
<td>[string]</td>
<td>Default: 'id'</td>
</tr>
<tr>
<td>Primary Key (existing field, used if the above option is left empty) Optional</td>
<td>PRIMARY_KEY</td>
<td>[tablefield: any]</td>
<td>Default: None</td>
</tr>
<tr>
<td>Geometry column name Optional</td>
<td>GEOCOLUMN</td>
<td>[string]</td>
<td>Default: 'geom'</td>
</tr>
<tr>
<td>Vector dimensions Optional</td>
<td>DIM</td>
<td>[enumeration]</td>
<td>Default: 0 (2D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tabulka 23.163 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distance tolerance for simplification</strong></td>
<td>SIMPLIFY</td>
<td>[string]</td>
<td>Defines a distance tolerance for the simplification of the vector geometries to be imported. By default there is no simplification.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td></td>
</tr>
<tr>
<td><strong>Maximum distance between 2 nodes (densification)</strong></td>
<td>SEGMENTIZE</td>
<td>[string]</td>
<td>The maximum distance between two nodes. Used to create intermediate points. By default there is no densification.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td></td>
</tr>
<tr>
<td><strong>Select features by extent (defined in input layer CRS)</strong></td>
<td>SPAT</td>
<td>[extent]</td>
<td>You can select features from a given extent that will be in the output table.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>None</td>
</tr>
<tr>
<td><strong>Clip the input layer using the above (rectangle) extent</strong></td>
<td>CLIP</td>
<td>[boolean]</td>
<td>The input layer will be clipped by the extent you defined before.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>False</td>
</tr>
<tr>
<td><strong>Select features using a SQL „WHERE“ statement (Ex: column=“value“)</strong></td>
<td>WHERE</td>
<td>[string]</td>
<td>Defines with a SQL „WHERE“ statement which features should be selected from the input layer</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td></td>
</tr>
<tr>
<td><strong>Group N features per transaction</strong> (Default: 2000)</td>
<td>GT</td>
<td>[string]</td>
<td>You can group the input features in transactions where N defines the size. By default N limits the transaction size to 20000 features.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td></td>
</tr>
<tr>
<td><strong>Overwrite existing table</strong></td>
<td>OVERWRITE</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>True</td>
</tr>
<tr>
<td><strong>Append to existing table</strong></td>
<td>APPEND</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>False</td>
</tr>
<tr>
<td><strong>Append and add new fields to existing table</strong></td>
<td>ADDFIELDS</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>False</td>
</tr>
<tr>
<td><strong>Do not launder columns/table names</strong></td>
<td>LAUNDER</td>
<td>[boolean]</td>
<td>With this option checked you can prevent the default behaviour (converting column names to lowercase, removing spaces and other invalid characters).</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>False</td>
</tr>
<tr>
<td><strong>Do not create Spatial Index</strong></td>
<td>INDEX</td>
<td>[boolean]</td>
<td>Prevents a spatial index for the output table from being created. By default, a spatial index is added.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>False</td>
</tr>
<tr>
<td><strong>Continue after a failure, skipping the failed feature</strong></td>
<td>SKIPFAILURES</td>
<td>[boolean]</td>
<td>If checked / True 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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>False</td>
</tr>
<tr>
<td><strong>Promote to Multi-part</strong></td>
<td>PROMOTETOMULTI</td>
<td>[boolean]</td>
<td>Casts features geometry type to multipart in the output table</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default:</td>
<td>True</td>
</tr>
</tbody>
</table>
Tabulka 23.163 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep width and precision of input attributes</td>
<td>PRECISION</td>
<td>[boolean]</td>
<td>Avoids modifying column attributes to comply with input data</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Additional GDAL creation options.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Outputs**

This algorithm has no output.

**Python code**

**Algorithm ID:** gdal:importvectorintopostgisdatabaseavailableconnections

```python
import processing
def 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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**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.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>OGR-supported vector layer to export to the database</td>
</tr>
<tr>
<td>Shape encoding</td>
<td>SHAPE_ENCODING</td>
<td>[string]</td>
<td>Sets the encoding to apply to the data</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output geometry type</td>
<td>GTYPE</td>
<td>[enumeration]</td>
<td>Defines the output geometry type. One of:</td>
</tr>
<tr>
<td>Default: 0</td>
<td></td>
<td></td>
<td>• 0 —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — GEOMETRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — POINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — LINestring</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — POLYGON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — GEOMETRYCOLLECTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — MULTIPoint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — MULTIPOLYGON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — MULTILINESTRING</td>
</tr>
<tr>
<td>Assign an output CRS</td>
<td>A_SRS</td>
<td>[crs]</td>
<td>Defines the output CRS of the database table</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>Reproject to this CRS on output</td>
<td>T_SRS</td>
<td>[crs]</td>
<td>Reprojects/transforms to this CRS on output</td>
</tr>
<tr>
<td>Override source CRS</td>
<td>S_SRS</td>
<td>[crs]</td>
<td>Overrides the input layer CRS</td>
</tr>
<tr>
<td>Optional</td>
<td>Host</td>
<td>HOST</td>
<td>Name of the database host</td>
</tr>
<tr>
<td>Portable</td>
<td>Port</td>
<td>PORT</td>
<td>Port number the PostgreSQL database server listens on</td>
</tr>
<tr>
<td>Optional</td>
<td>Username</td>
<td>USER</td>
<td>User name used to log in to the database</td>
</tr>
<tr>
<td>Optional</td>
<td>Database name</td>
<td>DBNAME</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Optional</td>
<td>Password</td>
<td>PASSWORD</td>
<td>Password used with Username to connect to the database</td>
</tr>
<tr>
<td>Optional</td>
<td>Schema (schema name)</td>
<td>SCHEMA</td>
<td>Defines the schema for the database table</td>
</tr>
<tr>
<td>Optional</td>
<td>Table name, leave blank to use input name</td>
<td>TABLE</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td>Primary Key (new field)</td>
<td>PK</td>
<td>Defines which attribute field will be the primary key of the database table</td>
</tr>
<tr>
<td>Optional</td>
<td>Primary Key (existing field, used if the above option is left empty)</td>
<td>PRIMARY_KEY</td>
<td>Defines which attribute field in the exported layer will be the primary key of the database table</td>
</tr>
<tr>
<td>Optional</td>
<td>Geometry column name</td>
<td>GEOCOLUMN</td>
<td>Defines in which attribute field to store the geometry information</td>
</tr>
<tr>
<td>Optional</td>
<td>Vector dimensions</td>
<td>DIM</td>
<td>Defines if the vector file to be imported has 2D or 3D data. One of: &lt;ul&gt;&lt;li&gt;0 — 2D&lt;/li&gt;&lt;li&gt;1 — 3D&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
<tr>
<td>Optional</td>
<td>Distance tolerance for simplification</td>
<td>SIMPLIFY</td>
<td>Defines a distance tolerance for the simplification of the vector geometries to be imported. By default no simplification there is no simplification.</td>
</tr>
<tr>
<td>Optional</td>
<td>Maximum distance between 2 nodes (densification)</td>
<td>SEGMENTIZE</td>
<td>The maximum distance between two nodes. Used to create intermediate points. By default there is no densification.</td>
</tr>
<tr>
<td>Optional</td>
<td>Select features by extent (defined in input layer CRS)</td>
<td>SPAT</td>
<td>You can select features from a given extent that will be in the output table.</td>
</tr>
</tbody>
</table>
Tabulka 23.164 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clip the input layer using the above (rectangle) extent</td>
<td>CLIP</td>
<td>[boolean]</td>
<td>The input layer will be clipped by the extent you defined before</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Fields to include (leave empty to use all fields)</td>
<td>FIELDS</td>
<td>[string] [list]</td>
<td>Defines fields to keep from the imported vector file. If none is selected, all the fields are imported.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: []</td>
<td></td>
</tr>
<tr>
<td>Select features using a SQL „WHERE“ statement (Ex: column=&quot;value&quot;)</td>
<td>WHERE</td>
<td>[string]</td>
<td>Defines with a SQL „WHERE“ statement which features should be selected for the output table</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td></td>
</tr>
<tr>
<td>Group N features per transaction (Default: 2000)</td>
<td>GT</td>
<td>[string]</td>
<td>You can group the input features in transactions where N defines the size. By default N limits the transaction size to 20000 features.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: ';'</td>
<td></td>
</tr>
<tr>
<td>Overwrite existing table</td>
<td>OVERWRITE</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: True</td>
<td></td>
</tr>
<tr>
<td>Append to existing table</td>
<td>APPEND</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Append and add new fields to existing table</td>
<td>ADDFIELDS</td>
<td>[boolean]</td>
<td>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.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Do not launder columns/table names</td>
<td>LAUNDER</td>
<td>[boolean]</td>
<td>With this option checked you can prevent the default behaviour (converting column names to lowercase, removing spaces and other invalid characters).</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Do not create Spatial Index</td>
<td>INDEX</td>
<td>[boolean]</td>
<td>Prevents a spatial index for the output table from being created. By default, a spatial index is added.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Continue after a failure, skipping the failed feature</td>
<td>SKIPFAILURES</td>
<td>[boolean]</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Promote to Multi-part</td>
<td>PROMOTETOMULTI</td>
<td>[boolean]</td>
<td>Casts features geometry type to multipart in the output table</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: True</td>
<td></td>
</tr>
<tr>
<td>Keep width and precision of input attributes</td>
<td>PRECISION</td>
<td>[boolean]</td>
<td>Avoids modifying column attributes to comply with input data</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: True</td>
<td></td>
</tr>
<tr>
<td>Additional creation options (optional)</td>
<td>OPTIONS</td>
<td>[string]</td>
<td>Additional GDAL creation options.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: ';' (no additional options)</td>
<td></td>
</tr>
</tbody>
</table>
Outputs

This algorithm has no output.

Python code

Algorithm ID: `gdal:importvectorintopostgisdatabase

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

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.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>INPUT</td>
<td>[vector: any]</td>
<td>Input vector layer</td>
</tr>
<tr>
<td>Summary output only</td>
<td>SUMMARY_ONLY</td>
<td>[boolean]</td>
<td>Optional</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: True</td>
<td></td>
</tr>
<tr>
<td>Suppress metadata info</td>
<td>NO_METADATA</td>
<td>[boolean]</td>
<td>Optional</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>Layer information</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>Specify the output HTML file that includes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: [Save</td>
<td>the file information. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to temporary file]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If no HTML-file is defined the output will</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>be written to a temporary file</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer information</td>
<td>OUTPUT</td>
<td>[html]</td>
<td>The output HTML-file that includes the file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>information.</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** gdal:ogrinfo

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

23.3 LAStools algorithm provider

LAStools is a collection of highly efficient, multicore command line tools for LiDAR data processing.

23.3.1 blast2dem

**Popis**

Turns points (up to billions) via seamless Delaunay triangulation implemented using streaming into large elevation, intensity, or RGB rasters.

For more info see the blast2dem page and its online README file.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>open</td>
<td>LAS tools GUI</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td>input</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file containing the points to be rastered in LAS/LAZ format.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
Tabulka 23.166 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| filter (by return, classification, flag) | FILTER_RETURN_CLASS_FLAGS1 | [enumeration] | Specifies which points to use to construct the temporary TIN that is then rasterized. One of:  
- 0 — —  
- 1 — keep_last  
- 2 — keep_first  
- 3 — keep_middle  
- 4 — keep_single  
- 5 — drop_single  
- 6 — keep_double  
- 7 — keep_class 2  
- 8 — keep_class 2 8  
- 9 — keep_class 8  
- 10 — keep_class 6  
- 11 — keep_class 9  
- 12 — keep_class 3 4 5  
- 13 — keep_class 2 6  
- 14 — drop_class 7  
- 15 — drop_withheld  
- 16 — drop_synthetic  
- 17 — drop_overlap  
- 18 — keep_withheld  
- 19 — keep_synthetic  
- 20 — keep_keypoint  
- 21 — keep_overlap |
| step size / pixel size | STEP | [number] | Default: 1.0  
| Attribute | ATTRIBUTE | [enumeration] | Specifies the attribute that is to be rastered. One of:  
- 0 — elevation  
- 1 — slope  
- 2 — intensity  
- 3 — rgb |
| Product | PRODUCT | [enumeration] | Default: 0  
| Use tile bounding box (after tiling with buffer) | USE_TILE_BB | [boolean] | Default: False  
| additional command line parameter(s) Optional | ADDITIONAL_OPTIONS | [string] | Default: '.'  
| | | | Specifies which points to use to construct the temporary TIN that is then rasterized. One of:  
- 0 — —  
- 1 — keep_last  
- 2 — keep_first  
- 3 — keep_middle  
- 4 — keep_single  
- 5 — drop_single  
- 6 — keep_double  
- 7 — keep_class 2  
- 8 — keep_class 2 8  
- 9 — keep_class 8  
- 10 — keep_class 6  
- 11 — keep_class 9  
- 12 — keep_class 3 4 5  
- 13 — keep_class 2 6  
- 14 — drop_class 7  
- 15 — drop_withheld  
- 16 — drop_synthetic  
- 17 — drop_overlap  
- 18 — keep_withheld  
- 19 — keep_synthetic  
- 20 — keep_keypoint  
- 21 — keep_overlap |

Pokračujte na další stránce
## Tabulka 23.166 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
</table>
| **Output raster file**    | OUTPUT_RASTER | [raster] | Specifies where the output raster is stored. Use image rasters like TIF, PNG, and JPG for false color, gray ramps, and hillshades. Use value rasters like TIF, BIL, IMG, ASC, DTM, FLT, XYZ, and CSV for actual values. One of:  
  • Skip Output  
  • Save to a Temporary File  
  • Save to File…  
  The file encoding can also be changed here. |
| **Outputs**               |               |        |       |
| **Output raster file**    | OUTPUT_RASTER | [raster] | The output raster |

### Python code

**Algorithm ID:** lastools:blast2dem

```
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

#### 23.3.2 blast2iso

### Popis

Turns points (up to billions) via seamless Delaunay triangulation implemented using streaming into iso-contour lines.

For more info see the blast2iso page and its online README file.

### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>open GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file containing the points to be used for creating iso-contour lines.</td>
</tr>
<tr>
<td>smooth underly-ing TIN</td>
<td>SMOOTH</td>
<td>[number]</td>
<td>Specifies if and with how many passes the temporary TIN should be smoothed</td>
</tr>
<tr>
<td>extract isoline with a spacing of</td>
<td>ISO_EVERY</td>
<td>[number]</td>
<td>Specifies spacing at which iso-contour lines are getting extracted (contour interval)</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>clean isolines shorter than (0 = do not clean)</td>
<td>CLEAN</td>
<td>[number]</td>
<td>Omits iso-contour lines that are shorter than the specified length</td>
</tr>
<tr>
<td>simplify segments shorter than (0 = do not simplify)</td>
<td>SIMPLIFY_LENGTH</td>
<td>[number]</td>
<td>Rudimentary simplification of iso-contour line segments that are shorter than the specified length.</td>
</tr>
<tr>
<td>simplify segment pairs with area less than (0 = do not simplify)</td>
<td>SIMPLIFY_AREA</td>
<td>[number]</td>
<td>Rudimentary simplification of bumps formed by consecutive line segments whose area is smaller than the specified size.</td>
</tr>
<tr>
<td>additional command line parameter(s)</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
<tr>
<td>Output vector file</td>
<td>OUTPUT_VECTOR</td>
<td>[vector: line]</td>
<td>The output line vector layer with contours</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output vector file</td>
<td>OUTPUT_VECTOR</td>
<td>[vector: line]</td>
<td>The output line vector layer with contours</td>
</tr>
</tbody>
</table>

## Python code

**Algorithm ID:** lastools:blast2iso

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.3.3 las2dem

#### Popis

Turns points (up to 20 million) via a temporary Delaunay triangulation that is rasterized with a user-defined step size into an elevation, intensity, or RGB raster.

For more info see the las2dem page and its online README file.
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>verbose</strong></td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td><strong>run new 64 bit executable</strong></td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td><strong>open LASTools GUI</strong></td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td><strong>input LAS/LAZ file</strong></td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file containing the points to be rastered in LAS/LAZ format.</td>
</tr>
</tbody>
</table>
| **filter (by return, classification, flags)** | FILTER_RETURN_CLASS_FLAGS1 | [enumeration] | Specifies which points to use to construct the temporary TIN that is then rasterized. One of: 
  • 0 — —
  • 1 — keep_last
  • 2 — keep_first
  • 3 — keep_middle
  • 4 — keep_single
  • 5 — drop_single
  • 6 — keep_double
  • 7 — keep_class 2
  • 8 — keep_class 2 8
  • 9 — keep_class 8
  • 10 — keep_class 6
  • 11 — keep_class 9
  • 12 — keep_class 3 4 5
  • 13 — keep_class 3
  • 14 — keep_class 4
  • 15 — keep_class 5
  • 16 — keep_class 2 6
  • 17 — drop_class 7
  • 18 — drop_withheld
  • 19 — drop_synthetic
  • 20 — drop_overlap
  • 21 — keep_withheld
  • 22 — keep_synthetic
  • 23 — keep_keypoint
  • 24 — keep_overlap
| **step size / pixel size**    | STEP           | [number]      | Default: 1.0                                                          |
| **Attribute**                 | ATTRIBUTE      | [enumeration] | Specifies the attribute to rasterise. One of: 
  • 0 — elevation
  • 1 — slope
  • 2 — intensity
  • 3 — rgb
  • 4 — edge_longest
  • 5 — edge_shortest
### 23.3. LaStools algorithm provider

#### 23.3.4 las2iso

**Popis**

Turns point clouds (up to 20 million per file) into iso-contour lines by creating a temporary Delaunay triangulation on which the contours are then traced.

For more info see the las2iso page and its online README file.

---

**Algorithm ID:** lastools:las2dem

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
## Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td></td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file containing the points to be used for creating iso-contour lines.</td>
</tr>
<tr>
<td>smooth underlying TIN</td>
<td>SMOOTH</td>
<td>[number]</td>
<td>Specifies if and with how many passes the temporary TIN should be smoothed</td>
</tr>
<tr>
<td>extract isoline with a spacing of</td>
<td>ISO_EVERY</td>
<td>[number]</td>
<td>Specifies spacing at which iso-contour lines are getting extracted (contour interval)</td>
</tr>
<tr>
<td>clean isolines shorter than (0 = do not clean)</td>
<td>CLEAN</td>
<td>[number]</td>
<td>Omits iso-contour lines that are shorter than the specified length</td>
</tr>
<tr>
<td>simplify segments shorter than (0 = do not simplify)</td>
<td>SIMPLIFY_LENGTH</td>
<td>[number]</td>
<td>Rudimentary simplification of iso-contour line segments that are shorter than the specified length.</td>
</tr>
<tr>
<td>simplify segment pairs with area less than (0 = do not simplify)</td>
<td>SIMPLIFY_AREA</td>
<td>[number]</td>
<td>Rudimentary simplification of bumps formed by consecutive line segments whose area is smaller than the specified size.</td>
</tr>
<tr>
<td>additional command line parameter(s) Optional</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
</tbody>
</table>

### Output vector file

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output vector file</td>
<td>OUTPUT_VECTOR</td>
<td>[vector: line]</td>
<td>The output line vector layer with contours</td>
</tr>
</tbody>
</table>

## Outputs
Python code

Algorithm ID: lastools:las2iso

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

23.3.5 las2las_filter

**Popis**

Uses las2las to filter LiDAR points based on different attributes and to write the surviving subset of points to a new LAZ or LAS file.

For more info see the las2las page and its online README file.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file containing the points to be used for creating iso-contour lines.</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>filter (by return, classification, flags)</td>
<td>FILTER_RETURN_CLASS_FLAGS</td>
<td>[enumeration]</td>
<td>Filters points based on various options such as return, classification, or flags. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — keep_last</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — keep_first</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — keep_middle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — keep_single</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — drop_single</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — keep_double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — keep_class 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — keep_class 28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — keep_class 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — keep_class 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — keep_class 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — keep_class 3 4 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — keep_class 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — keep_class 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — keep_class 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16 — keep_class 2 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 17 — drop_class 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 18 — drop_withheld</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 19 — drop_synthetic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 20 — drop_overlap</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 21 — keep_withheld</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 22 — keep_synthetic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 23 — keep_keypoint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 24 — keep_overlap</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| second filter (by return, classification, flags) | FILTER_RETURN_CLASS_FLAGS2 | [enumeration] Default: 0 | Filters points based on various options such as return, classification, or flags. One of:  
• 0 — —  
• 1 — keep_last  
• 2 — keep_first  
• 3 — keep_middle  
• 4 — keep_single  
• 5 — drop_single  
• 6 — keep_double  
• 7 — keep_class 2  
• 8 — keep_class 2 8  
• 9 — keep_class 8  
• 10 — keep_class 6  
• 11 — keep_class 9  
• 12 — keep_class 3 4 5  
• 13 — keep_class 3  
• 14 — keep_class 4  
• 15 — keep_class 5  
• 16 — keep_class 2 6  
• 17 — drop_class 7  
• 18 — drop_withheld  
• 19 — drop_synthetic  
• 20 — drop_overlap  
• 21 — keep_withheld  
• 22 — keep_synthetic  
• 23 — keep_keypoint  
• 24 — keep_overlap |
| filter (by coordinate, intensity, GPS time, …) | FILTER_COORDS_INTENSITY1 | [enumeration] Default: 0 | Filters points based on various other options (that require a value as argument). One of:  
• 0 — —  
• 1 — drop_x_above  
• 2 — drop_x_below  
• 3 — drop_y_above  
• 4 — drop_y_below  
• 5 — drop_z_above  
• 6 — drop_z_below  
• 7 — drop_intensity_above  
• 8 — drop_intensity_below  
• 9 — drop_gps_time_above  
• 10 — drop_gps_time_below  
• 11 — drop_scan_angle_above  
• 12 — drop_scan_angle_below  
• 13 — keep_point_source  
• 14 — drop_point_source  
• 15 — drop_point_source_above  
• 16 — drop_point_source_below  
• 17 — keep_user_data  
• 18 — drop_user_data  
• 19 — drop_user_data_above  
• 20 — drop_user_data_below  
• 21 — keep_every_nth  
• 22 — keep_random_fraction  
• 23 — thin_with_grid |
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>value for filter (by coordinate, intensity, GPS time, ...)</td>
<td>FILTER_COORDS_INTENSITY1</td>
<td>[number]</td>
<td>Default: None</td>
</tr>
</tbody>
</table>
| second filter (by coordinate, intensity, GPS time, ...) | FILTER_COORDS_INTENSITY2 | [enumeration] | Default: 0 | Filters points based on various other options (that require a value as argument). One of:  
• 0 —  
• 1 — drop_x_above  
• 2 — drop_x_below  
• 3 — drop_y_above  
• 4 — drop_y_below  
• 5 — drop_z_above  
• 6 — drop_z_below  
• 7 — drop_intensity_above  
• 8 — drop_intensity_below  
• 9 — drop_gps_time_above  
• 10 — drop_gps_time_below  
• 11 — drop_scan_angle_above  
• 12 — drop_scan_angle_below  
• 13 — keep_point_source  
• 14 — drop_point_source  
• 15 — drop_point_source_above  
• 16 — drop_point_source_below  
• 17 — keep_user_data  
• 18 — drop_user_data  
• 19 — drop_user_data_above  
• 20 — drop_user_data_below  
• 21 — keep_every_nth  
• 22 — keep_random_fraction  
• 23 — thin_with_grid |
| value for second filter (by coordinate, intensity, GPS time, ...) | FILTER_COORDS_INTENSITY2 | [number] | Default: None | The value to use as the argument for the filter selected above |
| additional command line parameter(s) | ADDITIONAL_OPTIONS | [string] | Default: ';' | Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user. |
| Output LAS/LAZ file | OUTPUT_LASLAZ | [file] | Default: [Skip output] | Specifies where the output point cloud is stored. Use LAZ for compressed output, LAS for uncompressed output, and TXT for ASCII. One of:  
• Skip Output  
• Save to a Temporary File  
• Save to File…  
The file encoding can also be changed here. |
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>The output LAS/LAZ format file</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: lastools:las2las_filter

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.3.6 las2las_project

Transform LAS/LAZ files in a folder to another CRS.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>Input LAS/LAZ file</td>
</tr>
<tr>
<td>source projection</td>
<td>SOURCE_PROJECTI</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — epsg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — utm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — sp83</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — sp27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — longlat</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — latlong</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — ecef</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>source utm zone</td>
<td>SOURCE_UTM</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — 1 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — 2 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — 3 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — 4 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — 5 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — 6 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — 7 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — 8 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — 9 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — 10 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — 11 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — 12 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — 13 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — 14 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — 15 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16 — 16 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 17 — 17 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 18 — 18 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 19 — 19 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 20 — 20 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 21 — 21 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 22 — 22 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 23 — 23 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 24 — 24 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 25 — 25 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 26 — 26 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 27 — 27 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 28 — 28 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 29 — 29 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 30 — 30 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 31 — 31 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 32 — 32 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 33 — 33 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 34 — 34 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 35 — 35 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 36 — 36 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 37 — 37 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 38 — 38 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 39 — 39 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 40 — 40 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 41 — 41 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 42 — 42 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 43 — 43 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 44 — 44 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 45 — 45 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 46 — 46 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 47 — 47 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 48 — 48 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 49 — 49 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 50 — 50 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 51 — 51 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 52 — 52 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 53 — 53 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 54 — 54 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 55 — 55 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 56 — 56 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 57 — 57 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 58 — 58 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 59 — 59 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 60 — 60 (north)</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>--------------------</td>
<td>---------</td>
</tr>
<tr>
<td>source state plane code</td>
<td>SOURCE_SP</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>0 ———</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 — AK_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 — AK_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 — AK_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 — AK_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 — AK_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 — AK_6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 — AK_7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8 — AK_8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9 — AK_9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 — AL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11 — AL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12 — AR_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13 — AR_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14 — AZ_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15 — AZ_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16 — AZ_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17 — CA_I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>18 — CA_II</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19 — CA_III</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20 — CA_IV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>21 — CA_V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>22 — CA_VI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>23 — CA_VII</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24 — CO_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25 — CO_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>26 — CO_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>27 — CT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>28 — DE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>29 — FL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>30 — FL_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31 — FL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32 — GA_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>33 — GA_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>34 — HI_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>35 — HI_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>36 — HI_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>37 — HI_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>38 — HI_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>39 — IA_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40 — IA_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>41 — ID_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>42 — ID_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>43 — ID_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>44 — IL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>45 — IL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>46 — IN_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>47 — IN_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>48 — KS_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>49 — KS_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>50 — KY_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>51 — KY_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>52 — LA_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>53 — LA_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>54 — MA_I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>55 — MA_M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>56 — MD</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>------------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| target projection | TARGET_PROJECT | enumeration | One of:
|               |            | Default: 0 | • 0 — —
|               |            |            | • 1 — epsg
|               |            |            | • 2 — utm
|               |            |            | • 3 — sp83
|               |            |            | • 4 — sp27
|               |            |            | • 5 — longlat
|               |            |            | • 6 — latlong
|               |            |            | • 7 — ecef

Pokračujte na další stránce
### Tabulka 23.171 - pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>target utm zone</td>
<td>TARGET_UTM</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — 1 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — 2 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — 3 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — 4 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — 5 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — 6 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — 7 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — 8 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — 9 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — 10 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — 11 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — 12 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — 13 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — 14 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — 15 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16 — 16 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 17 — 17 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 18 — 18 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 19 — 19 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 20 — 20 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 21 — 21 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 22 — 22 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 23 — 23 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 24 — 24 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 25 — 25 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 26 — 26 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 27 — 27 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 28 — 28 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 29 — 29 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 30 — 30 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 31 — 31 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 32 — 32 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 33 — 33 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 34 — 34 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 35 — 35 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 36 — 36 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 37 — 37 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 38 — 38 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 39 — 39 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 40 — 40 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 41 — 41 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 42 — 42 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 43 — 43 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 44 — 44 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 45 — 45 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 46 — 46 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 47 — 47 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 48 — 48 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 49 — 49 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 50 — 50 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 51 — 51 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 52 — 52 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 53 — 53 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 54 — 54 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 55 — 55 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 56 — 56 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 57 — 57 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 58 — 58 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 59 — 59 (north)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 60 — 60 (north)</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>target state plane code</td>
<td>TARGET_SP</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — AK_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — AK_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — AK_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — AK_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — AK_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — AK_6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — AK_7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — AK_8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — AK_9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — AL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — AL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — AR_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — AR_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — AZ_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — AZ_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16 — AZ_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 17 — CA_I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 18 — CA_H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 19 — CA_HH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 20 — CA_IV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 21 — CA_V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 22 — CA_VI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 23 — CA_VII</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 24 — CO_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 25 — CO_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 26 — CO_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 27 — CT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 28 — DE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 29 — FL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 30 — FL_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 31 — FL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 32 — GA_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 33 — GA_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 34 — HI_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 35 — HI_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 36 — HI_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 37 — HI_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 38 — HI_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 39 — IA_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 40 — IA_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 41 — ID_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 42 — ID_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 43 — ID_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 44 — IL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 45 — IL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 46 — IN_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 47 — IN_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 48 — KS_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 49 — KS_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 50 — KY_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 51 — KY_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 52 — LA_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 53 — LA_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 54 — MA_I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 55 — MA_M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 56 — MD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tabulka 23.171 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>additional command line parameter(s)</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[folder]</td>
<td>Specifies where the folder for the output point clouds. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary Directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to Directory...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>The output LAS/LAZ format file</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: lastools:las2las_project

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

23.3.7 las2las_transform

**Popis**

Uses las2las to filter LiDAR points based on different attributes and to write the surviving subset of points to a new LAZ or LAS file.

For more info see the las2las page and its online README file.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: False</td>
<td></td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The first file containing points to be merged</td>
</tr>
<tr>
<td>Label</td>
<td>Type</td>
<td>Popis</td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>transform (coordinates)</td>
<td>TRANSFORM_COORDINATE</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[enumeration]</td>
<td>Default: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td>value for transform (coordinates)</td>
<td>TRANSFORM_COORDINATE_ARG</td>
<td>The value that specifies the amount of translating, scaling, or clamping done by the transform selected above.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: ''</td>
<td>The value that specifies the amount of translating, scaling, or clamping done by the transform selected above.</td>
<td></td>
</tr>
<tr>
<td>second transform (coordinates)</td>
<td>TRANSFORM_COORDINATE</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[enumeration]</td>
<td>Default: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td>value for second transform (coordinates)</td>
<td>TRANSFORM_COORDINATE</td>
<td>The value that specifies the amount of translating, scaling, or clamping done by the transform selected above.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: ''</td>
<td>The value that specifies the amount of translating, scaling, or clamping done by the transform selected above.</td>
<td></td>
</tr>
<tr>
<td>transform (intensities, scan angles, GPS times, ...)</td>
<td>TRANSFORM_OTHER</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[enumeration]</td>
<td>Default: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
<td>Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:</td>
<td></td>
</tr>
<tr>
<td>value for transform (intensities, scan angles, GPS times, ...)</td>
<td>TRANSFORM_OTHER</td>
<td>The value that specifies the amount of scaling, translating, clamping or setting that is done by the transform selected above.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: ''</td>
<td>The value that specifies the amount of scaling, translating, clamping or setting that is done by the transform selected above.</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| second transform (intensities, scan angles, GPS times, ...) | TRANSFORM_OTHER2 | [enumeration] | Either translate, scale, or clamp the X, Y, or Z coordinate by the value specified below. One of:  
• 0 — —  
• 1 — scale_intensity  
• 2 — translate_intensity  
• 3 — clamp_intensity_above  
• 4 — clamp_intensity_below  
• 5 — scale_scan_angle  
• 6 — translate_scan_angle  
• 7 — translate_gps_time  
• 8 — set_classification  
• 9 — set_user_data  
• 10 — set_point_source  
• 11 — scale_rgb_up  
• 12 — scale_rgb_down  
• 13 — repair_zero_returns |
| value for second transform (intensities, scan angles, GPS times, ...) | TRANSFORM_OTHER2_ARG | [string] | The value that specifies the amount of scaling, translating, clamping or setting that is done by the transform selected above. |
| operations (first 7 need an argument) | OPERATION | [enumeration] | One of:  
• 0 — —  
• 1 — set_point_type  
• 2 — set_point_size  
• 3 — set_version_minor  
• 4 — set_version_major  
• 5 — start_at_point  
• 6 — stop_at_point  
• 7 — remove_vlr  
• 8 — auto_reoffset  
• 9 — week_to_adjusted  
• 10 — adjusted_to_week  
• 11 — auto_reoffset  
• 12 — scale_rgb_up  
• 13 — scale_rgb_down  
• 14 — remove_all_vlrs  
• 15 — remove_extra  
• 16 — clip_to_bounding_box |
| argument for operation | OPERATIONARG | [string] | The value to use as the argument for the operation selected above |
| additional command line parameter(s) Optional | ADDITIONAL_OPTIONS | [string] | Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user. |
Tabulka 23.172 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>Specifies where the output point cloud is stored. Use LAZ for compressed output, LAS for uncompressed output, and TXT for ASCII. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default:</td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Skip output]</td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File…</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>The output (merged) LAS/LAZ format file</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: lastools:las2las_transform

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.3.8 las2txt

Popis

Translates a LAS/LAZ file to a text file.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>Default: None</td>
</tr>
<tr>
<td>parse_string</td>
<td>PARSE</td>
<td>[string]</td>
<td>Default: 'xyz'</td>
</tr>
<tr>
<td>additional command line parameters</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Default: ';'</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
</tbody>
</table>
Tabulka 23.173 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output ASCII file</strong></td>
<td>OUTPUT GENERIC</td>
<td>[file]</td>
<td>Specify the output file. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Create Temporary Layer (TEMPORARY_OUTPUT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output ASCII file</strong></td>
<td>OUTPUT GENERIC</td>
<td>[file]</td>
<td>The output file</td>
</tr>
</tbody>
</table>

**Python code**

Algorithm ID: lastools:las2txt

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**23.3.9 lasindex**

**Popis**

<put algorithm description here>

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>verbose</strong></td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>Default: None</td>
</tr>
<tr>
<td>append *.lax file to *.laz file</td>
<td>APPEND_LAX</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>is mobile or terrestrial LiDAR (not airborne)</td>
<td>MOBILE_OR_TERRESTRIAL</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>additional command line parameters</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Default: ' '</td>
</tr>
</tbody>
</table>
| Optional                       |                  |                 | Specifies other command-line switches not available via this menu but known to the (advanced) LASTools user.
Outputs

The algorithm has no output.

Python code

Algorithm ID: lastools:lasindex

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.3.10 lasgrid

Grids a selected attribute (e.g. elevation, intensity, classification, scan angle, …) of a large point clouds with a user-defined step size onto raster using a particular method (e.g. min, max, average).

For more info see the lasgrid page and its online README file.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Default: False Generates more textual control output to the console</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False The file containing the points to be rastered in LAS/LAZ format.</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>filter (by return,</td>
<td>FILTER_RETURN_CLASS_FLAGS1</td>
<td>[enumeration]</td>
<td>Specifies the subset of points to use for the gridding. One of:</td>
</tr>
<tr>
<td>classification, flags)</td>
<td>Default: 0</td>
<td></td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — keep_last</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — keep_first</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — keep_middle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — keep_single</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — drop_single</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — keep_double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — keep_class 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — keep_class 2 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — keep_class 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — keep_class 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — keep_class 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — keep_class 3 4 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — keep_class 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — keep_class 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — keep_class 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16 — keep_class 2 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 17 — drop_class 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 18 — drop_withheld</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 19 — drop_synthetic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 20 — drop_overlap</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 21 — keep_withheld</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 22 — keep_synthetic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 23 — keep_keypoint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 24 — keep_overlap</td>
</tr>
</tbody>
</table>

| step size / pixel size       | STEP                              | [number]      | Specifies the size of the cells of the grid the TIN is rasterized onto |
| Attribute                    | ATTRIBUTE                          | [enumeration] | Specifies the attribute to rasterise. One of:                        |
|                              | Default: 0                         |               | • 0 — elevation                                                      |
|                              |                                   |               | • 1 — intensity                                                     |
|                              |                                   |               | • 2 — rgb                                                           |
|                              |                                   |               | • 3 — classification                                                |

| Method                       | METHOD                             | [enumeration] | Specifies how the attributes falling into one cell are turned into a raster value. One of: |
|                              | Default: 0                         |               | • 0 — lowest                                                        |
|                              |                                   |               | • 1 — heighest                                                      |
|                              |                                   |               | • 2 — average                                                      |
|                              |                                   |               | • 3 — stddev                                                       |

| use tile bounding box (after tiling with buffer) | USE_TILE_BB | [boolean] | Specifies to limit the rastered area to the tile bounding box (only meaningful for input LAS/LAZ tiles that were created with lastile). |
| additional command line parameter(s) Optional | ADDITIONAL_OPTIONS | [string] | Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user. |

Pokračujte na další stránce
Tabulka 23.175 - pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output raster file</td>
<td>OUTPUT_RASTER</td>
<td>[raster]</td>
<td>Specifies where the output raster is stored. Use image rasters like TIF, PNG, and JPG for false color, gray ramps, and hillshades. Use value rasters like TIF, BIL, IMG, ASC, DTM, FLT, XYZ, and CSV for actual values. One of:&lt;br&gt;• Skip Output&lt;br&gt;• Save to a Temporary File&lt;br&gt;• Save to File…&lt;br&gt;The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output raster file</td>
<td>OUTPUT_RASTER</td>
<td>[raster]</td>
<td>The output raster</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: lastools:lasgrid

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.3.11 lasinfo

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>open GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file to get information about.</td>
</tr>
<tr>
<td>compute density</td>
<td>COMPUTE_DENSITY</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>repair bounding box</td>
<td>REPAIR_BB</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>repair counters</td>
<td>REPAIR_COUNTERS</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Label</td>
<td>Název</td>
<td>Type</td>
<td>Popis</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-----------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>histogram</td>
<td>HISTO1</td>
<td>[enumeration]</td>
<td>First histogram. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — z</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — intensity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — classification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — scan_angle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — user_data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — point_source</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — gps_time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — Z</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — attribute0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — attribute1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — attribute2</td>
</tr>
<tr>
<td>bin size</td>
<td>HISTO1_BIN</td>
<td>[number]</td>
<td>Default: 1.0</td>
</tr>
<tr>
<td>histogram</td>
<td>HISTO2</td>
<td>[enumeration]</td>
<td>Second histogram. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — z</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — intensity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — classification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — scan_angle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — user_data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — point_source</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — gps_time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — Z</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — attribute0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — attribute1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — attribute2</td>
</tr>
<tr>
<td>bin size</td>
<td>HISTO2_BIN</td>
<td>[number]</td>
<td>Default: 1.0</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Tabulka 23.176 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
<td>HISTO3</td>
<td>[enumeration]</td>
<td>Third histogram. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td>• 0 — —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 — x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 — y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 — z</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 — intensity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 — classification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 — scan_angle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 — user_data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8 — point_source</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9 — gps_time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 — X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11 — Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 — Z</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 13 — attribute0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 14 — attribute1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 15 — attribute2</td>
</tr>
<tr>
<td>bin size</td>
<td>HISTO3_BIN</td>
<td>[number]</td>
<td>Default: 1.0</td>
</tr>
<tr>
<td>additional command line parameter(s)</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
<tr>
<td>Output ASCII file</td>
<td>OUTPUT_GENERIC</td>
<td>[file]</td>
<td>Default: [Skip output]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specifies where the output is stored. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output ASCII file</td>
<td>OUTPUT_GENERIC</td>
<td>[file]</td>
<td>The file with the output</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** lastools:lasinfo

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
23.3.12 lasmerge

Merge up to seven LAS/LAZ files into one.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>files are flightlines</td>
<td>FILES_ARE_FLIGHTLINES</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>apply file source ID</td>
<td>APPLY_FILE_SOURCE_ID</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The first file containing points to be merged</td>
</tr>
<tr>
<td>2nd file</td>
<td>FILE2</td>
<td>[file]</td>
<td>The second file to merge</td>
</tr>
<tr>
<td>3rd file</td>
<td>FILE3</td>
<td>[file]</td>
<td>The third file to merge</td>
</tr>
<tr>
<td>4th file</td>
<td>FILE4</td>
<td>[file]</td>
<td>The fourth file to merge</td>
</tr>
<tr>
<td>5th file</td>
<td>FILE5</td>
<td>[file]</td>
<td>The fifth file to merge</td>
</tr>
<tr>
<td>6th file</td>
<td>FILE6</td>
<td>[file]</td>
<td>The sixth file to merge</td>
</tr>
<tr>
<td>7th file</td>
<td>FILE7</td>
<td>[file]</td>
<td>The seventh file to merge</td>
</tr>
<tr>
<td>additional command line parameter(s)</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
</tbody>
</table>
| Output LAS/LAZ file | OUTPUT_LASLAZ | [file] | Specifies where the output point cloud is stored. Use LAZ for compressed output, LAS for uncompressed output, and TXT for ASCII. One of:  
  - Skip Output  
  - Save to a Temporary File  
  - Save to File…  
  The file encoding can also be changed here. |
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>The output (merged) LAS/LAZ format file</td>
</tr>
</tbody>
</table>

Python code

Algorithm ID: lastools:lasmerge

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.3.13 lasprecision

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>open GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Starts the GUI of LAStools with pre-populated input files</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file the input point cloud</td>
</tr>
<tr>
<td>additional command line parameter(s) Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDITIONAL_OPTS</td>
<td>ADDITIONAL_OPTS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
<tr>
<td>Output ASCII file</td>
<td>OUTPUT_GENERIC</td>
<td>[file]</td>
<td>Specifies where the output ASCII file is stored. One of:</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: [Skip output]</td>
<td>- Skip Output</td>
</tr>
<tr>
<td>- Save to a Temporary File</td>
<td></td>
<td>- Save to File…</td>
<td></td>
</tr>
<tr>
<td>The file encoding can also be changed here.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output ASCII file</td>
<td>OUTPUT_GENERIC</td>
<td>[file]</td>
<td>The output ASCII file</td>
</tr>
</tbody>
</table>
Python code

**Algorithm ID:** lastools:lasprecision

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### 23.3.14 lasquery

**Popis**

<put algorithm description here>

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>verbose</strong></td>
<td>VERBOSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td><strong>open LAStools GUI</strong></td>
<td>GUI</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td><strong>input LAS/LAZ file</strong></td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file the input point cloud</td>
</tr>
<tr>
<td><strong>area of interest</strong></td>
<td>AOI</td>
<td>[extent]</td>
<td>The extent</td>
</tr>
<tr>
<td><strong>additional command line parameter(s)</strong></td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
</tbody>
</table>

**Outputs**

**Python code**

**Algorithm ID:** lastools:lasquery

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
23.3.15 lasvalidate

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file the input point cloud</td>
</tr>
<tr>
<td>save report to '*_LVS.xml'</td>
<td>ONE_REPORT_PER</td>
<td>[boolean]</td>
<td></td>
</tr>
<tr>
<td>additional command line parameter(s)</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output XML file</td>
<td>OUTPUT_GENERIC</td>
<td>[file]</td>
<td>Specifies where the output XML file is stored. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Skip Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output XML file</td>
<td>OUTPUT_GENERIC</td>
<td>[file]</td>
<td>The output XML file</td>
</tr>
</tbody>
</table>

**Python code**

**Algorithm ID:** lastools:lasvalidate

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.3.16 laszip

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>VERBOUSE</td>
<td>[boolean]</td>
<td>Generates more textual control output to the console</td>
</tr>
<tr>
<td>run new 64 bit executable</td>
<td>CPU64</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>open LAStools GUI</td>
<td>GUI</td>
<td>[boolean]</td>
<td>Starts the GUI of LAStools with pre-populated input files</td>
</tr>
<tr>
<td>input LAS/LAZ file</td>
<td>INPUT_LASLAZ</td>
<td>[file]</td>
<td>The file to be zipped</td>
</tr>
<tr>
<td>only report size</td>
<td>REPORT_SIZE</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>create spatial indexing file (*.lax)</td>
<td>CREATE_LAX</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>append *.lax into *.laz file</td>
<td>APPEND_LAX</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>additional command line parameter(s)</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Default: ''</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>Default: [Skip output]</td>
</tr>
</tbody>
</table>
|                                 |                     |            | Specifies where the output point cloud is stored. Use LAZ for compressed output, LAS for uncompressed output, and TXT for ASCII. One of:  
|                                 |                     |            | • Skip Output                                                      |
|                                 |                     |            | • Save to a Temporary File                                          |
|                                 |                     |            | • Save to File...                                                  |
|                                 |                     |            | The file encoding can also be changed here.                          |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>The output file</td>
</tr>
</tbody>
</table>

### Python code

**Algorithm ID:** lastools:laszip

```python
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 [Using processing algorithms from the console](#) for details on how to run processing algorithms from the Python console.

#### 23.3.17 txt2las

**Parameters**
Tabulka 23.182 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip the first n lines</td>
<td>SKIP</td>
<td>[number]</td>
<td>Default: 0</td>
</tr>
<tr>
<td>resolution of x and y coordinate</td>
<td>SCALE_FACTOR_XY</td>
<td>[number]</td>
<td>Default: 0.01</td>
</tr>
<tr>
<td>resolution of z coordinate</td>
<td>SCALE_FACTOR_Z</td>
<td>[number]</td>
<td>Default: 0.01</td>
</tr>
<tr>
<td>resolution of z coordinate</td>
<td>SCALE_FACTOR_Z</td>
<td>[number]</td>
<td>Default: 0.01</td>
</tr>
<tr>
<td>source projection</td>
<td>PROJECTION</td>
<td>[enumeration]</td>
<td>Default: 0</td>
</tr>
<tr>
<td>source epsg code</td>
<td>EPSG_CODE</td>
<td>[number]</td>
<td></td>
</tr>
</tbody>
</table>

One of:
- 0 — —
- 1 — epsg
- 2 — utm
- 3 — sp83
- 4 — sp27
- 5 — longlat
- 6 — latlong
- 7 — ecef

Pokračujte na další stránce
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>utm zone</td>
<td>UTM</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
</tbody>
</table>
|            |       | Default: 0    | • 0 — —
|            |       |               | • 1 — 1 (north) |
|            |       |               | • 2 — 2 (north) |
|            |       |               | • 3 — 3 (north) |
|            |       |               | • 4 — 4 (north) |
|            |       |               | • 5 — 5 (north) |
|            |       |               | • 6 — 6 (north) |
|            |       |               | • 7 — 7 (north) |
|            |       |               | • 8 — 8 (north) |
|            |       |               | • 9 — 9 (north) |
|            |       |               | • 10 — 10 (north) |
|            |       |               | • 11 — 11 (north) |
|            |       |               | • 12 — 12 (north) |
|            |       |               | • 13 — 13 (north) |
|            |       |               | • 14 — 14 (north) |
|            |       |               | • 15 — 15 (north) |
|            |       |               | • 16 — 16 (north) |
|            |       |               | • 17 — 17 (north) |
|            |       |               | • 18 — 18 (north) |
|            |       |               | • 19 — 19 (north) |
|            |       |               | • 20 — 20 (north) |
|            |       |               | • 21 — 21 (north) |
|            |       |               | • 22 — 22 (north) |
|            |       |               | • 23 — 23 (north) |
|            |       |               | • 24 — 24 (north) |
|            |       |               | • 25 — 25 (north) |
|            |       |               | • 26 — 26 (north) |
|            |       |               | • 27 — 27 (north) |
|            |       |               | • 28 — 28 (north) |
|            |       |               | • 29 — 29 (north) |
|            |       |               | • 30 — 30 (north) |
|            |       |               | • 31 — 31 (north) |
|            |       |               | • 32 — 32 (north) |
|            |       |               | • 33 — 33 (north) |
|            |       |               | • 34 — 34 (north) |
|            |       |               | • 35 — 35 (north) |
|            |       |               | • 36 — 36 (north) |
|            |       |               | • 37 — 37 (north) |
|            |       |               | • 38 — 38 (north) |
|            |       |               | • 39 — 39 (north) |
|            |       |               | • 40 — 40 (north) |
|            |       |               | • 41 — 41 (north) |
|            |       |               | • 42 — 42 (north) |
|            |       |               | • 43 — 43 (north) |
|            |       |               | • 44 — 44 (north) |
|            |       |               | • 45 — 45 (north) |
|            |       |               | • 46 — 46 (north) |
|            |       |               | • 47 — 47 (north) |
|            |       |               | • 48 — 48 (north) |
|            |       |               | • 49 — 49 (north) |
|            |       |               | • 50 — 50 (north) |
|            |       |               | • 51 — 51 (north) |
|            |       |               | • 52 — 52 (north) |
|            |       |               | • 53 — 53 (north) |
|            |       |               | • 54 — 54 (north) |
|            |       |               | • 55 — 55 (north) |
|            |       |               | • 56 — 56 (north) |
|            |       |               | • 57 — 57 (north) |
|            |       |               | • 58 — 58 (north) |
|            |       |               | • 59 — 59 (north) |
|            |       |               | • 60 — 60 (north) |
|            |       |               | • 61 — 1 (south) |
|            |       |               | • 62 — 2 (south) |
|            |       |               | • 63 — 3 (south) |
|            |       |               | • 64 — 4 (south) |
|            |       |               | • 65 — 5 (south) |
|            |       |               | • 66 — 6 (south) |
|            |       |               | • 67 — 7 (south) |
|            |       |               | • 68 — 8 (south) |
|            |       |               | • 69 — 9 (south) |
|            |       |               | • 70 — 10 (south) |
|            |       |               | • 71 — 11 (south) |
|            |       |               | • 72 — 12 (south) |
|            |       |               | • 73 — 13 (south) |
|            |       |               | • 74 — 14 (south) |
|            |       |               | • 75 — 15 (south) |
|            |       |               | • 76 — 16 (south) |
|            |       |               | • 77 — 17 (south) |
|            |       |               | • 78 — 18 (south) |
|            |       |               | • 79 — 19 (south) |
|            |       |               | • 80 — 20 (south) |
|            |       |               | • 81 — 21 (south) |
|            |       |               | • 82 — 22 (south) |
|            |       |               | • 83 — 23 (south) |
|            |       |               | • 84 — 24 (south) |
|            |       |               | • 85 — 25 (south) |
|            |       |               | • 86 — 26 (south) |
|            |       |               | • 87 — 27 (south) |
|            |       |               | • 88 — 28 (south) |
|            |       |               | • 89 — 29 (south) |
|            |       |               | • 90 — 30 (south) |
|            |       |               | • 91 — 31 (south) |
|            |       |               | • 92 — 32 (south) |
|            |       |               | • 93 — 33 (south) |
|            |       |               | • 94 — 34 (south) |
|            |       |               | • 95 — 35 (south) |
|            |       |               | • 96 — 36 (south) |
|            |       |               | • 97 — 37 (south) |
|            |       |               | • 98 — 38 (south) |
|            |       |               | • 99 — 39 (south) |
|            |       |               | • 100 — 40 (south) |

23.3. LASTools algorithm provider

Pokračujte na předchozí stránce
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>state plane code</td>
<td>SP</td>
<td>[enumeration]</td>
<td>One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 0 —  —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1 — AK_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 2 — AK_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 3 — AK_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 4 — AK_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 5 — AK_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 6 — AK_6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 7 — AK_7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 8 — AK_8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 9 — AK_9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 10 — AL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 11 — AL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 12 — AR_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 13 — AR_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 14 — AZ_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 15 — AZ_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 16 — AZ_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 17 — CA_I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 18 — CA_II</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 19 — CA_III</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 20 — CA_IV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 21 — CA_V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 22 — CA_VI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 23 — CA_VII</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 24 — CO_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 25 — CO_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 26 — CO_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 27 — CT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 28 — DE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 29 — FL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 30 — FL_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 31 — FL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 32 — GA_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 33 — GA_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 34 — HI_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 35 — HI_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 36 — HI_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 37 — HI_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 38 — HI_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 39 — IA_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 40 — IA_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 41 — ID_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 42 — ID_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 43 — ID_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 44 — IL_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 45 — IL_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 46 — IN_E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 47 — IN_W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 48 — KS_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 49 — KS_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 50 — KY_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 51 — KY_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 52 — LA_N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 53 — LA_S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 54 — MA_I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 55 — MA_M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 56 — MD</td>
</tr>
</tbody>
</table>

1042 Kapitola 23. Processing providers and algorithms
Tabulka 23.182 – pokračujte na předchozí stránce

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>additional command line parameter(s)</td>
<td>ADDITIONAL_OPTIONS</td>
<td>[string]</td>
<td>Specifies other command-line switches not available via this menu but known to the (advanced) LAStools user.</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>Default: `</td>
<td></td>
</tr>
</tbody>
</table>
| Output LAS/LAZ file | OUTPUT_LASLAZ | [file] | Specifies where the output point cloud is stored. Use LAZ for compressed output, LAS for uncompressed output, and TXT for ASCII. One of:  
  • Skip Output  
  • Save to a Temporary File  
  • Save to File…  
  The file encoding can also be changed here. |

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>output LAS/LAZ file</td>
<td>OUTPUT_LASLAZ</td>
<td>[file]</td>
<td>The output file</td>
</tr>
</tbody>
</table>

**Python code**

*Algorithm ID:* `lastools:txt2las`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

## 23.4 TauDEM algorithm provider

**TauDEM** (Terrain Analysis Using Digital Elevation Models) is a set of Digital Elevation Model (DEM) tools for the extraction and analysis of hydrologic information from topography as represented by a DEM. This is software developed at Utah State University (USU) for hydrologic digital elevation model analysis and watershed delineation.

TauDEM is distributed as a set of standalone command line executable programs for a Windows and source code for compiling and use on other systems.

**Poznámka:** Please remember that Processing contains only the interface description, so you need to install TauDEM 5.0.6 by yourself and configure Processing properly.

Documentation for TauDEM algorithms derived from official TauDEM documentation
23.4.1 Basic Grid Analysis

D-Infinity Contributing Area

Popis

Calculates a grid of specific catchment area which is the contributing area per unit contour length using the multiple flow direction D-infinity approach. D-infinity flow direction is defined as steepest downward slope on planar triangular facets on a block centered grid. The contribution at each grid cell is taken as the grid cell length (or when the optional weight grid input is used, from the weight grid). The contributing area of each grid cell is then taken as its own contribution plus the contribution from upslope neighbors that have some fraction draining to it according to the D-infinity flow model. The flow from each cell either all drains to one neighbor, if the angle falls along a cardinal (0, π/2, π, 3π/2) or ordinal (π/4, 3π/4, 5π/4, 7π/4) direction, or is on an angle falling between the direct angle to two adjacent neighbors. In the latter case the flow is proportioned between these two neighbor cells according to how close the flow direction angle is to the direct angle to those cells. The contour length used here is the grid cell size. The resulting units of the specific catchment area are length units the same as those of the grid cell size.

When the optional weight grid is not used, the result is reported in terms of specific catchment area, the upslope area per unit contour length, taken here as the number of cells times grid cell length (cell area divided by cell length). This assumes that grid cell length is the effective contour length, in the definition of specific catchment area and does not distinguish any difference in contour length dependent upon the flow direction. When the optional weight grid is used, the result is reported directly as a summation of weights, without any scaling.

If the optional outlet point shapefile is used, only the outlet cells and the cells upslope (by the D-infinity flow model) of them are in the domain to be evaluated.

By default, the tool checks for edge contamination. This is defined as the possibility that a contributing area value may be underestimated due to grid cells outside of the domain not being counted. This occurs when drainage is inwards from the boundaries or areas with „no data“ values for elevation. The algorithm recognizes this and reports „no data“ for the contributing area. It is common to see streaks of „no data“ values extending inwards from boundaries along...
flow paths that enter the domain at a boundary. This is the desired effect and indicates that contributing area for these grid cells is unknown due to it being dependent on terrain outside of the domain of data available. Edge contamination checking may be turned off in cases where you know it is not an issue or want to ignore these problems, if for example, the DEM has been clipped along a watershed outline.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-infinity flow directions</td>
<td>DINF_FLOWDIR</td>
<td>[raster]</td>
<td>A grid of flow directions based on the D-infinity flow method using the steepest slope of a triangular facet. Flow direction is determined as the direction of the steepest downward slope on the 8 triangular facets of a 3x3 block centered grid. Flow direction is encoded as an angle in radians, counterclockwise from east as a continuous (floating point) quantity between 0 and 2\pi. The resulting flow in a grid is then usually interpreted as being proportioned between the two neighboring cells that define the triangular facet with the steepest downward slope.</td>
</tr>
<tr>
<td>Outlets</td>
<td>OUTLETS</td>
<td>[vector: point]</td>
<td>A point shapefile defining the outlets of interest. If this input file is used, only the cells upslope of these outlet cells are considered to be within the domain being evaluated.</td>
</tr>
<tr>
<td>Weight grid</td>
<td>WEIGHT_GRID</td>
<td>[raster]</td>
<td>A grid giving contribution to flow for each cell. These contributions (also sometimes referred to as weights or loadings) are used in the contributing area accumulation. If this input file is not used, the result is reported in terms of specific catchment area (the upslope area per unit contour length) taken as the number of cells times grid cell length (cell area divided by cell length).</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
A flag that indicates whether the tool should check for edge contamination. Edge contamination is defined as the possibility that a contributing area value may be underestimated due to the fact that grid cells outside of the domain have not been evaluated. This occurs when drainage is inwards from the boundaries or areas with NODATA values for elevation. The algorithm recognizes this and reports NODATA for the impacted cells. It is common to see streaks of NODATA values extending inwards from boundaries along flow paths that enter the domain at a boundary. This is the desired effect and indicates that contributing area for these grid cells is unknown due to it being dependent on terrain outside of the domain of available data. Edge contamination checking may be turned off in cases where you know this is not an issue, or want to ignore these problems, if for example, the DEM has been clipped along a watershed outline.

Specification of the output raster. One of:
- Save to a Temporary File
- Save to File…
The file encoding can also be changed here.

A grid of specific catchment area which is the contributing area per unit contour length using the multiple flow direction D-infinity approach. The contributing area of each grid cell is then taken as its own contribution plus the contribution from upslope neighbors that have some fraction draining to it according to the D-infinity flow model.

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
D-Infinity Flow Directions

Popis

Assigns a flow direction based on the D-infinity flow method using the steepest slope of a triangular facet (Tarboton, 1997, “A New Method for the Determination of Flow Directions and Contributing Areas in Grid Digital Elevation Models”, Water Resources Research, 33(2): 309-319). Flow direction is defined as steepest downward slope on planar triangular facets on a block centered grid. Flow direction is encoded as an angle in radians counter-clockwise from east as a continuous (floating point) quantity between 0 and 2π. The flow direction angle is determined as the direction of the steepest downward slope on the eight triangular facets formed in a 3 x 3 grid cell window centered on the grid cell of interest. The resulting flow in a grid is then usually interpreted as being proportioned between the two neighboring cells that define the triangular facet with the steepest downward slope.

A block-centered representation is used with each elevation value taken to represent the elevation of the center of the corresponding grid cell. Eight planar triangular facets are formed between each grid cell and its eight neighbors. Each of these has a downslope vector which when drawn outwards from the center may be at an angle that lies within or outside the 45 degree (π/4 radian) angle range of the facet at the center point. If the slope vector angle is within the facet angle, it represents the steepest flow direction on that facet. If the slope vector angle is outside a facet, the steepest flow direction associated with that facet is taken along the steepest edge. The slope and flow direction associated with the grid cell is taken as the magnitude and direction of the steepest downslope vector from all eight facets. Slope is measured as drop/distance, i.e. tan of the slope angle.

In the case where no slope vectors are positive (downslope), the flow direction is set using the method of Garbrecht and Martz (1997) for the determination of flow across flat areas. This makes flat areas drain away from high ground and towards low ground. The flow path grid to enforce drainage along existing streams is an optional input, and if used, takes precedence over elevations for the setting of flow directions.

The D-infinity flow direction algorithm may be applied to a DEM that has not had its pits filled, but it will then result in „no data“ values for the D-infinity flow direction and slope associated with the lowest point of the pit.

---

23.4. TauDEM algorithm provider
Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pit filled elevation</td>
<td>PIT_FILLED</td>
<td>[raster]</td>
<td>A grid of elevation values. This is usually the output of the „Pit Remove“ tool, in which case it is elevations with pits removed. Pits are low elevation areas in digital elevation models (DEMs) that are completely surrounded by higher terrain. They are generally taken to be artifacts of the digitization process that interfere with the processing of flow across DEMs. So they are removed by raising their elevation to the point where they just drain off the domain. This step is not essential if you have reason to believe that the pits in your DEM are real. If a few pits actually exist and so should not be removed, while at the same time others are believed to be artifacts that need to be removed, the actual pits should have NODATA elevation values inserted at their lowest point. NODATA values serve to define edges of the domain in the flow field, and elevations are only raised to where flow is off an edge, so an internal NODATA value will stop a pit from being removed, if necessary.</td>
</tr>
<tr>
<td>D-infinity flow directions</td>
<td>DINF_FLOWDIR</td>
<td>[raster]</td>
<td>Specification of the output flow direction raster. One of: • Save to a Temporary File • Save to File… The file encoding can also be changed here.</td>
</tr>
<tr>
<td>D-infinity slope</td>
<td>DINF_SLOPE</td>
<td>[raster]</td>
<td>Specification of the output slope raster. One of: • Save to a Temporary File • Save to File… The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

Outputs
<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-infinity flow directions</td>
<td>DINF_FLOWDIR</td>
<td>[raster]</td>
<td>A grid of flow directions based on the D-infinity flow method using the steepest slope of a triangular facet. Flow direction is determined as the direction of the steepest downward slope on the 8 triangular facets of a 3x3 block centered grid. Flow direction is encoded as an angle in radians, counterclockwise from east as a continuous (floating point) quantity between 0 and 2π. The resulting flow in a grid is then usually interpreted as being proportioned between the two neighboring cells that define the triangular facet with the steepest downward slope.</td>
</tr>
</tbody>
</table>

**Algorithm ID:** taudem:dinfflowdir

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**D8 Contributing Area**

**Popis**

Calculates a grid of contributing areas using the single direction D8 flow model. The contribution of each grid cell is taken as one (or when the optional weight grid is used, the value from the weight grid). The contributing area for each grid cell is taken as its own contribution plus the contribution from upslope neighbors that drain in to it according to the D8 flow model.

If the optional outlet point shapefile is used, only the outlet cells and the cells upslope (by the D8 flow model) of them are in the domain to be evaluated.

By default, the tool checks for edge contamination. This is defined as the possibility that a contributing area value may be underestimated due to grid cells outside of the domain not being counted. This occurs when drainage is inwards from the boundaries or areas with "no data" values for elevation. The algorithm recognizes this and reports "no data" for the contributing area. It is common to see streaks of "no data" values extending inwards from boundaries along flow paths that enter the domain at a boundary. This is the desired effect and indicates that contributing area for these grid cells is unknown due to it being dependent on terrain outside of the domain of data available. Edge contamination checking may be turned off in cases where you know this is not an issue or want to ignore these problems, if for example, the DEM has been clipped along a watershed outline.
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D8 flow directions</strong></td>
<td><strong>D8_FLOWDIR</strong></td>
<td>[raster]</td>
<td>A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can be obtained as the output of the „D8 Flow Directions“ tool.</td>
</tr>
<tr>
<td><strong>Outlets</strong></td>
<td><strong>OUTLETS</strong></td>
<td>[vector: point]</td>
<td>A point shapefile defining the outlets of interest. If this input file is used, only the cells upslope of these outlet cells are considered to be within the domain being evaluated.</td>
</tr>
<tr>
<td><strong>Weight grid</strong></td>
<td><strong>WEIGHT_GRID</strong></td>
<td>[raster]</td>
<td>A grid giving contribution to flow for each cell. These contributions (also sometimes referred to as weights or loadings) are used in the contributing area accumulation. If this input file is not used, the contribution to flow will assumed to be one for each grid cell.</td>
</tr>
<tr>
<td><strong>Check for edge contamination</strong></td>
<td><strong>EDGE_CONTAMINATION</strong></td>
<td>[boolean]</td>
<td>A flag that indicates whether the tool should check for edge contamination. Edge contamination is defined as the possibility that a contributing area value may be underestimated due to the fact that grid cells outside of the domain have not been evaluated. This occurs when drainage is inwards from the boundaries or areas with NODATA values for elevation. The algorithm recognizes this and reports NODATA for the impacted cells. It is common to see streaks of NODATA values extending inwards from boundaries along flow paths that enter the domain at a boundary. This is the desired effect and indicates that contributing area for these grid cells is unknown due to it being dependent on terrain outside of the domain of available data. Edge contamination checking may be turned off in cases where you know this is not an issue, or want to ignore these problems, if for example, the DEM has been clipped along a watershed outline.</td>
</tr>
</tbody>
</table>
| **D8 specific catchment area** | **D8_CONTRIB_AREA** | [raster] | Specification of the output raster. One of:  
  • Save to a Temporary File  
  • Save to File…  
  The file encoding can also be changed here. |
Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D8 specific catchment area</td>
<td>D8_CONTRIB_AREA</td>
<td>[raster]</td>
<td>A grid of contributing area values calculated as the cells own contribution plus the contribution from upslope neighbors that drain in to it according to the D8 flow model.</td>
</tr>
</tbody>
</table>

**Algorithm ID:** taudem:aread8

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**D8 Flow Directions**

**Popis**

Creates 2 grids. The first contains the flow direction from each grid cell to one of its adjacent or diagonal neighbors, calculated using the direction of steepest descent. The second contain the slope, as evaluated in the direction of steepest descent, and is reported as drop/distance, i.e. tan of the angle. Flow direction is reported as NODATA for any grid cell adjacent to the edge of the DEM domain, or adjacent to a NODATA value in the DEM. In flat areas, flow directions are assigned away from higher ground and towards lower ground using the method of Garbrecht and Martz (1997). The D8 flow direction algorithm may be applied to a DEM that has not had its pits filled, but it will then result in NODATA values for flow direction and slope at the lowest point of each pit.

D8 Flow Direction Coding:

- 1 — East
- 2 — Northeast
- 3 — North
- 4 — Northwest
- 5 — West
- 6 — Southwest
- 7 — South
- 8 — Southeast

The flow direction routing across flat areas is performed according to the method described by Garbrecht, J. and L. W. Martz, (1997), „The Assignment of Drainage Direction Over Flat Surfaces in Raster Digital Elevation Models“, Journal of Hydrology, 193: 204-213.
### Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pit filled elevation</td>
<td>PIT_FILLED</td>
<td>[raster]</td>
<td>A grid of elevation values. This is usually the output of the „Pit Remove“ tool, in which case it is elevations with pits removed. Pits are low elevation areas in digital elevation models (DEMs) that are completely surrounded by higher terrain. They are generally taken to be artifacts of the digitization process that interfere with the processing of flow across DEMs. So they are removed by raising their elevation to the point where they just drain off the domain. This step is not essential if you have reason to believe that the pits in your DEM are real. If a few pits actually exist and so should not be removed, while at the same time others are believed to be artifacts that need to be removed, the actual pits should have NODATA elevation values inserted at their lowest point. NODATA values serve to define edges of the domain in the flow field, and elevations are only raised to where flow is off an edge, so an internal NODATA value will stop a pit from being removed, if necessary.</td>
</tr>
</tbody>
</table>

#### D8 flow directions

| D8_FLOWDIR | [raster] | Specification of the output flow direction raster. One of:  
• Save to a Temporary File  
• Save to File…  
The file encoding can also be changed here. |

#### D8 slope

| D8_SLOPE | [raster] | Specification of the output slope raster. One of:  
• Save to a Temporary File  
• Save to File…  
The file encoding can also be changed here. |

### Outputs

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D8 flow directions</td>
<td>D8_FLOWDIR</td>
<td>[raster]</td>
<td>A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope.</td>
</tr>
<tr>
<td>D8 slope</td>
<td>D8_SLOPE</td>
<td>[raster]</td>
<td>A grid giving slope in the D8 flow direction. This is measured as drop/distance.</td>
</tr>
</tbody>
</table>

**Algorithm ID:** `taudem:d8flowdir`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
Grid Network

Popis

Creates 3 grids that contain for each grid cell: 1) the longest path, 2) the total path, and 3) the Strahler order number. These values are derived from the network defined by the D8 flow model.

The longest upslope length is the length of the flow path from the furthest cell that drains to each cell. The total upslope path length is the length of the entire grid network upslope of each grid cell. Lengths are measured between cell centers taking into account cell size and whether the direction is adjacent or diagonal.

Strahler order is defined as follows: A network of flow paths is defined by the D8 Flow Direction grid. Source flow paths have a Strahler order number of one. When two flow paths of different order join the order of the downstream flow path is the order of the highest incoming flow path. When two flow paths of equal order join the downstream flow path order is increased by 1. When more than two flow paths join the downstream flow path order is calculated as the maximum of the highest incoming flow path order or the second highest incoming flow path order + 1. This generalizes the common definition to cases where more than two flow paths join at a point.

Where the optional mask grid and threshold value are input, the function is evaluated only considering grid cells that lie in the domain with mask grid value greater than or equal to the threshold value. Source (first order) grid cells are taken as those that do not have any other grid cells from inside the domain draining in to them, and only when two of these flow paths join is order propagated according to the ordering rules. Lengths are also only evaluated counting paths within the domain greater than or equal to the threshold.

If the optional outlet point shapefile is used, only the outlet cells and the cells upslope (by the D8 flow model) of them are in the domain to be evaluated.

Parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D8 flow directions</td>
<td>D8_FLOWDIR</td>
<td>[raster]</td>
<td>A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can be obtained as the output of the „D8 Flow Directions“ tool.</td>
</tr>
<tr>
<td>Mask Grid</td>
<td>MASK_GRID</td>
<td>[raster]</td>
<td>A grid that is used to determine the domain do be analyzed. If the mask grid value &gt;= mask threshold (see below), then the cell will be included in the domain. While this tool does not have an edge contamination flag, if edge contamination analysis is needed, then a mask grid from a function like „D8 Contributing Area“ that does support edge contamination can be used to achieve the same result.</td>
</tr>
<tr>
<td>Mask threshold</td>
<td>THRESHOLD</td>
<td>[number]</td>
<td>Default: 100.0 This input parameter is used in the calculation mask grid value &gt;= mask threshold to determine if the grid cell is in the domain to be analyzed.</td>
</tr>
<tr>
<td>Outlets</td>
<td>OUTLETS</td>
<td>[vector: point]</td>
<td>A point shapefile defining the outlets of interest. If this input file is used, only the cells upslope of these outlet cells are considered to be within the domain being evaluated.</td>
</tr>
</tbody>
</table>

Pokračujte na další stránce
### Longest upslope length

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longest upslope</td>
<td>LONGEST_PATH</td>
<td>[raster]</td>
<td>Specification of the output raster with total upslope lengths. One of:</td>
</tr>
<tr>
<td>length</td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Default:** [Save to temporary file]

**Specification of the output raster with total upslope lengths. One of:**

- Save to a Temporary File
- Save to File...

The file encoding can also be changed here.

### Total upslope length

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total upslope</td>
<td>TOTAL_PATH</td>
<td>[raster]</td>
<td>Specification of the output raster with upslope lengths. One of:</td>
</tr>
<tr>
<td>length</td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Default:** [Save to temporary file]

**Specification of the output raster with upslope lengths. One of:**

- Save to a Temporary File
- Save to File...

The file encoding can also be changed here.

### Strahler network order

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strahler network</td>
<td>STRAHLER_ORDER[raster]</td>
<td>[raster]</td>
<td>Specification of the output raster with Strahler network order. One of:</td>
</tr>
<tr>
<td>order</td>
<td></td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Default:** [Save to temporary file]

**Specification of the output raster with Strahler network order. One of:**

- Save to a Temporary File
- Save to File...

The file encoding can also be changed here.

### Algorithm ID: taudem:gridnet

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Pit Remove**

**Popis**

Identifies all pits in the DEM and raises their elevation to the level of the lowest pour point around their edge. Pits are low elevation areas in digital elevation models (DEMs) that are completely surrounded by higher terrain. They are generally taken to be artifacts that interfere with the routing of flow across DEMs, so are removed by raising their elevation to the point where they drain off the edge of the domain. The pour point is the lowest point on the boundary of the “watershed” draining to the pit. This step is not essential if you have reason to believe that the pits in your DEM are real. If a few pits actually exist and so should not be removed, while at the same time others are believed to be artifacts that need to be removed, the actual pits should have NODATA elevation values inserted at their lowest point. NODATA values serve to define edges in the domain, and elevations are only raised to where flow is off an edge, so an internal NODATA value will stop a pit from being removed, if necessary.

**Parameters**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation</td>
<td>ELEVATION</td>
<td>[raster]</td>
<td>A digital elevation model (DEM) grid to serve as the base input for the terrain analysis and stream delineation.</td>
</tr>
<tr>
<td>Depression mask</td>
<td>DEPRESSION_MASK</td>
<td>[raster]</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consider only 4 way neighbors</td>
<td>FOUR_NEIGHBOURS</td>
<td>[boolean]</td>
<td>Default: False</td>
</tr>
<tr>
<td>Pit removed elevation</td>
<td>PIT_FILLED</td>
<td>[raster]</td>
<td>Specification of the (pit filled) output raster. One of:</td>
</tr>
<tr>
<td></td>
<td>Default:</td>
<td></td>
<td>• Save to a Temporary File</td>
</tr>
<tr>
<td></td>
<td>[Save to temporary file]</td>
<td></td>
<td>• Save to File...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The file encoding can also be changed here.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Label</th>
<th>Název</th>
<th>Type</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pit removed elevation</td>
<td>PIT_FILLED</td>
<td>[raster]</td>
<td>A grid of elevation values with pits removed so that flow is routed off of the domain.</td>
</tr>
</tbody>
</table>

**Algorithm ID:** `taudem:pitremove`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
23.4.2 Specialized Grid Analysis

D8 Distance To Streams

Popis

Computes the horizontal distance to stream for each grid cell, moving downslope according to the D8 flow model, until a stream grid cell is encountered.

Parameters

**D8 Flow Direction Grid [raster]** This input is a grid of flow directions that are encoded using the D8 method where all flow from a cell goes to a single neighboring cell in the direction of steepest descent. This grid can be obtained as the output of the "D8 Flow Directions" tool.

**Stream Raster Grid [raster]** A grid indicating streams. Such a grid can be created by several of the tools in the "Stream Network Analysis" toolset. However, the tools in the "Stream Network Analysis" toolset only create grids with a value of 0 for no stream, or 1 for stream cells. This tool can also accept grids with values greater than 1, which can be used in conjunction with the Threshold parameter to determine the location of streams. This allows Contributing Area grids to be used to define streams as well as the normal Stream Raster grids. This grid expects integer (long integer) values and any non-integer values will be truncated to an integer before being evaluated.

**Threshold [number]** This value acts as threshold on the Stream Raster Grid to determine the location of streams. Cells with a Stream Raster Grid value greater than or equal to the Threshold value are interpreted as streams.

Default: 50

Outputs

**Output Distance to Streams [raster]** A grid giving the horizontal distance along the flow path as defined by the D8 Flow Directions Grid to the streams in the Stream Raster Grid.

Algorithm ID: taudem:d8hdisttostrm

```python
import processing
taudem.run("taudem:d8hdisttostrm", {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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

D-Infinity Avalanche Runout

Popis

Identifies an avalanche’s affected area and the flow path length to each cell in that affected area. All cells downslope from each source area cell, up to the point where the slope from the source to the affected area is less than a threshold angle called the Alpha Angle can be in the affected area. This tool uses the D-infinity multiple flow direction method for determining flow direction. This will likely cause very small amounts of flow to be dispersed to some downslope cells that might overstate the affected area, so a threshold proportion can be set to avoid this excess dispersion. The flow path length is the distance from the cell in question to the source cell that has the highest angle.

All points downslope from the source area are potentially in the affected area, but not beyond a point where the slope from the source to the affected area is less than a threshold angle called the Alpha Angle.

Slope is to be measured using the straight line distance from source point to evaluation point.
23.4. TauDEM algorithm provider

Elevations

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6.99</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>6.98</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>6.97</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>6.96</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Yellow cell is the source
Green: downslope of source

Straight-line distance from highest point of source

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.414214</td>
<td>2.236068</td>
<td>3.162278</td>
<td>4.123106</td>
<td>5.09902</td>
</tr>
<tr>
<td>2</td>
<td>2.236068</td>
<td>2.828427</td>
<td>3.605551</td>
<td>4.472136</td>
<td>5.385185</td>
</tr>
<tr>
<td>3</td>
<td>3.162278</td>
<td>3.605551</td>
<td>4.242641</td>
<td>5</td>
<td>5.630952</td>
</tr>
<tr>
<td>4</td>
<td>4.123106</td>
<td>4.472136</td>
<td>5</td>
<td>5.656884</td>
<td>6.403124</td>
</tr>
<tr>
<td>5</td>
<td>5.09902</td>
<td>5.385185</td>
<td>5.830952</td>
<td>6.403124</td>
<td>7.071068</td>
</tr>
</tbody>
</table>

Yellow cell is the source
Green: downslope of source

Drop in elevation from highest point in source

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3.01</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3.02</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3.03</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.04</td>
<td>0</td>
</tr>
</tbody>
</table>

Yellow cell is the source
Green: downslope of source

2 The cell size (a fiddle factor for me to make sensible values)
18 The threshold angle for being in the runout zone

The slope angle from the highest point in the source to each cell

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>13</td>
<td>9</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>13</td>
<td>19</td>
<td>23</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>13</td>
<td>17</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>0</td>
</tr>
</tbody>
</table>

Yellow cell is the source
Green: downslope of source
Grey cells are BOTH downslope of the source AND have a sufficiently steep angle to be in the runout zone

1057
It makes more physical sense to me for the angle to be measured along the flow path. Nevertheless it is equally easy to code straight line angles as angles along the flow path, so an option that allows switching will be provided. The most practical way to evaluate avalanche runout is to keep track of the source point with the greatest angle to each point. Then the recursive upslope flow algebra approach will look at a grid cell and all its upslope neighbors that flow to it. Information from the upslope neighbors will be used to calculate the angle to the grid cell in question and retain it in the runout zone if the angle exceeds the alpha angle. This procedure makes the assumption that the maximum angle at a grid cell will be from the set of cells that have maximum angles to the inflowing neighbors. This will always be true of angle is calculated along a flow path, but I can conceive of cases where flow paths bend back on themselves where this would not be the case for straight line angles.

The D-infinity multiple flow direction field assigns flow from each grid cell to multiple downslope neighbors using proportions ($P_{ik}$) that vary between 0 and 1 and sum to 1 for all flows out of a grid cell. It may be desirable to specify a threshold $T$ that this proportion has to exceed before a grid cell is counted as flowing to a downslope grid cell, e.g. $P_{ik} > T$ ($=0.2$ say) to avoid dispersion to grid cells that get very little flow. $T$ will be specified as a user input. If all upslope grid cells are to be used $T$ may be input as 0.

Avalanche source sites are to be input as a short integer grid (name suffix *ass, e.g. demass) comprised of positive values where avalanches may be triggered and 0 values elsewhere.

The following grids are output:

- $rz$ — A runout zone indicator with value 0 to indicate that this grid cell is not in the runout zone and value > 0 to indicate that this grid cell is in the runout zone. Since there may be information in the angle to the associated source site, this variable will be assigned the angle to the source site (in degrees)

- $dm$ — Along flow distance from the source site that has the highest angle to the point in question

**Parameters**

**D-Infinity Flow Direction Grid [raster]** A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This can be created by the tool „D-Infinity Flow Directions“.

**Pit Filled Elevation Grid [raster]** This input is a grid of elevation values. As a general rule, it is recommended that you use a grid of elevation values that have had the pits removed for this input. Pits are generally taken to be artifacts that interfere with the analysis of flow across them. This grid can be obtained as the output of the „Pit Remove“ tool, in which case it contains elevation values where the pits have been filled to the point where they just drain.

**Avalanche Source Site Grid [raster]** This is a grid of source areas for snow avalanches that are commonly identified manually using a mix of experience and visual interpretation of maps. Avalanche source sites are to be input as a short integer grid (name suffix *ass, e.g. demass) comprised of positive values where avalanches may be triggered and 0 values elsewhere.

**Proportion Threshold [number]** This value is a threshold proportion that is used to limit the dispersion of flow caused by using the D-infinity multiple flow direction method for determining flow direction. The D-infinity multiple flow direction method often causes very small amounts of flow to be dispersed to some downslope cells that might overstate the affected area, so a threshold proportion can be set to avoid this excess dispersion.

Default: 0.2

**Alpha Angle Threshold [number]** This value is the threshold angle, called the Alpha Angle, that is used to determine which of the cells downslope from the source cells are in the affected area. Only the cells downslope from each source area cell, up to the point where the slope from the source to the affected area is less than a threshold angle are in the affected area.

Default: 18

**Measure distance along flow path [boolean]** This option selects the method used to measure the distance used to calculate the slope angle. If option is True then measure it along the flow path, where the False option causes the slope to be measure along the straight line distance from the source cell to the evaluation cell.

Default: True
Outputs

**Runout Zone Grid [raster]** This grid identifies the avalanche’s runout zone (affected area) using a runout zone indicator with value 0 to indicate that this grid cell is not in the runout zone and value > 0 to indicate that this grid cell is in the runout zone. Since there may be information in the angle to the associated source site, this variable will be assigned the angle to the source site (in degrees).

**Path Distance Grid [raster]** This is a grid of the flow distance from the source site that has the highest angle to each cell.

**Algorithm ID:** taudem:dinfavalanche

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**D-Infinity Concentration Limited Accumulation**

**Popis**

This function applies to the situation where an unlimited supply of a substance is loaded into flow at a concentration or solubility threshold \( C_{\text{sol}} \) over a region indicated by an indicator grid (dg). It is a grid of the concentration of a substance at each location in the domain, where the supply of substance from a supply area is loaded into the flow at a concentration or solubility threshold. The flow is first calculated as a D-infinity weighted contributing area of an input Effective Runoff Weight Grid (notionally excess precipitation). The concentration of substance over the supply area (indicator grid) is at the concentration threshold. As the substance moves downslope with the D-infinity flow field, it is subject to first order decay in moving from cell to cell as well as dilution due to changes in flow. The decay multiplier grid gives the fractional (first order) reduction in quantity in moving from grid cell \( x \) to the next downslope cell. If the outlets shapefile is used, the tool only evaluates the part of the domain that contributes flow to the locations given by the shapefile. This is useful for tracking a contaminant or compound from an area with unlimited supply of that compound that is loaded into a flow at a concentration or solubility threshold over a zone and flow from the zone may be subject to decay or attenuation.

The indicator grid (dg) is used to delineate the area of the substance supply using the \((0, 1)\) indicator function \( i(x) \). \( A[\cdot] \) denotes the weighted accumulation operator evaluated using the D-Infinity Contributing Area function. The Effective Runoff Weight Grid gives the supply to the flow (e.g. the excess rainfall if this is overland flow) denoted as \( w(x) \). The specific discharge is then given by:

\[
Q(x) = A[w(x)]
\]

This weighted accumulation \( Q(x) \) is output as the Overland Flow Specific Discharge Grid. Over the substance supply area concentration is at the threshold (the threshold is a saturation or solubility limit). If \( i(x) = 1 \), then

\[
C(x) = C_{\text{sol}}, \text{ and } L(x) = C_{\text{sol}} Q(x),
\]

where \( L(x) \) denotes the load being carried by the flow. At remaining locations, the load is determined by load accumulation and the concentration by dilution:

\[
L(x) = L(i, j) = \sum_{k \text{ contributing neighbors}} p_k d(i_k, j_k) L(i_k, j_k)
\]

\[
C(x) = L(x)/Q(x)
\]
Here \( d(x) = d(i, j) \) is a decay multiplier giving the fractional (first order) reduction in mass in moving from grid cell \( x \) to the next downslope cell. If travel (or residence) times \( t(x) \) associated with flow between cells are available \( d(x) \) may be evaluated as \( \exp(-k \cdot t(x)) \) where \( k \) is a first order decay parameter. The Concentration grid output is \( C(x) \). If the outlets shapefile is used, the tool only evaluates the part of the domain that contributes flow to the locations given by the shapefile.

Useful for tracking a contaminant released or partitioned to flow at a fixed threshold concentration.
Parameters

**D-Infinity Flow Direction Grid** [raster] A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This grid can be created by the function "D-Infinity Flow Directions".

**Disturbance Indicator Grid** [raster] A grid that indicates the source zone of the area of substance supply and must be 1 inside the zone and 0 or NODATA over the rest of the domain.

**Decay Multiplier Grid** [raster] A grid giving the factor by which flow leaving each grid cell is multiplied before accumulation on downslope grid cells. This may be used to simulate the movement of an attenuating or decaying substance. If travel (or residence) times \( t(x) \) associated with flow between cells are available \( c_l(x) \) may be evaluated as \( \exp(-k \cdot t(x)) \) where \( k \) is a first order decay parameter.

**Effective Runoff Weight Grid** [raster] A grid giving the input quantity (notionally effective runoff or excess precipitation) to be used in the D-infinity weighted contributing area evaluation of Overland Flow Specific Discharge.

**Outlets shapefile** [vector: point] Optional
This optional input is a point shapefile defining outlets of interest. If this file is used, the tool will only evaluate the area upslope of these outlets.

**Concentration Threshold** [number] The concentration or solubility threshold. Over the substance supply area, concentration is at this threshold.
Default: 1.0

**Check for edge contamination** [boolean] This option determines whether the tool should check for edge contamination. Edge contamination is defined as the possibility that a value may be underestimated due to grid cells outside of the domain not being considered when determining contributing area.
Default: True

Outputs

**Concentration Grid** [raster] A grid giving the resulting concentration of the compound of interest in the flow.

Algorithm ID: **taudem:dinfconclimaccum**

```python
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 **Using processing algorithms from the console** for details on how to run processing algorithms from the Python console.

D-Infinity Decaying Accumulation

**Popis**

The D-Infinity Decaying Accumulation tool creates a grid of the accumulated quantity at each location in the domain where the quantity accumulates with the D-infinity flow field, but is subject to first order decay in moving from cell to cell. By default, the quantity contribution of each grid cell is the cell length to give a per unit width accumulation, but can optionally be expressed with a weight grid. The decay multiplier grid gives the fractional (first order) reduction in quantity in accumulating from grid cell \( x \) to the next downslope cell.

A decayed accumulation operator \( D[A] \) takes as input a mass loading field \( m(x) \) expressed at each grid location as \( m(i, j) \) that is assumed to move with the flow field but is subject to first order decay in moving from cell to cell. The output is the accumulated mass at each location \( DA(x) \). The accumulation of \( m \) at each grid cell can be numerically evaluated.
DA[m(x)] = DA(i, j) = m(i, j) Δ^2 + \sum_{k \text{ contributing neighbors}} p_k d(i_k, j_k) DA(i_k, j_k)

Here \(d(x) = d(i, j)\) is a decay multiplier giving the fractional (first order) reduction in mass in moving from grid cell \(x\) to the next downslope cell. If travel (or residence) times \(t(x)\) associated with flow between cells are available \(d(x)\) may be evaluated as \(\exp(-k \cdot t(x))\) where \(k\) is a first order decay parameter. The weight grid is used to represent the mass loading \(m(x)\). If not specified this is taken as 1. If the outlets shapefile is used the function is only evaluated on that part of the domain that contributes flow to the locations given by the shapefile.

Useful for a tracking contaminant or compound subject to decay or attenuation.
Parameters

D-Infinity Flow Direction Grid [raster] A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This grid can be created by the function "D-Infinity Flow Directions".

Decay Multiplier Grid [raster] A grid giving the factor by which flow leaving each grid cell is multiplied before accumulation on downslope grid cells. This may be used to simulate the movement of an attenuating substance.

Weight Grid [raster] Optional
A grid giving weights (loadings) to be used in the accumulation. If this optional grid is not specified, weights are taken as the linear grid cell size to give a per unit width accumulation.

Outlets Shapefile [vector: point] Optional
This optional input is a point shapefile defining outlets of interest. If this file is used, the tool will only evaluate the area upslope of these outlets.

Check for edge contamination [boolean] This option determines whether the tool should check for edge contamination. Edge contamination is defined as the possibility that a value may be underestimated due to grid cells outside of the domain not being considered when determining contributing area.

Default: True

Outputs

Decayed Specific Catchment Area Grid [raster] The D-Infinity Decaying Accumulation tool creates a grid of the accumulated mass at each location in the domain where mass moves with the D-infinity flow field, but is subject to first order decay in moving from cell to cell.

Algorithm ID: taudem:dinfdecayaccum

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

D-Infinity Distance Down

Popis

Calculates the distance downslope to a stream using the D-infinity flow model. The D-infinity flow model is a multiple flow direction model, because the outflow from each grid cell is proportioned between up to 2 downslope grid cells. As such, the distance from any grid cell to a stream is not uniquely defined. Flow that originates at a particular grid cell may enter the stream at a number of different cells. The statistical method may be selected as the longest, shortest or weighted average of the flow path distance to the stream. Also one of several ways of measuring distance may be selected: the total straight line path (Pythagoras), the horizontal component of the straight line path, the vertical component of the straight line path, or the total surface flow path.
## Parameters

**D-Infinity Flow Direction Grid** [raster] A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This can be created by the tool „D-Infinity Flow Directions“.

**Pit Filled Elevation Grid** [raster] This input is a grid of elevation values. As a general rule, it is recommended that you use a grid of elevation values that have had the pits removed for this input. Pits are generally taken to be artifacts that interfere with the analysis of flow across them. This grid can be obtained as the output of the „Pit Remove“ tool, in which case it contains elevation values where the pits have been filled to the point where they just drain.

**Stream Raster Grid** [raster] A grid indicating streams, by using a grid cell value of 1 on streams and 0 off streams. This is usually the output of one of the tools in the „Stream Network Analysis“ toolset.

**Weight Path Grid** [raster] Optional

A grid giving weights (loadings) to be used in the distance calculation. This might be used for example where only flow distance through a buffer is to be calculated. The weight is then 1 in the buffer and 0 outside it. Alternatively the weight may reflect some sort of cost function for travel over the surface, perhaps representing travel time or attenuation of a process. If this input file is not used, the loadings will assumed to be one for each grid cell.

**Statistical Method** [enumeration] Statistical method used to calculate the distance down to the stream. In the D-Infinity flow model, the outflow from each grid cell is proportioned between two downslope grid cells. Therefore, the distance from any grid cell to a stream is not uniquely defined. Flow that originates at a particular grid cell may enter the stream at a number of cells. The distance to the stream may be defined as the longest (maximum), shortest (minimum) or weighted average of the distance down to the stream.

Options:
- 0 — Minimum
- 1 — Maximum
- 2 — Average

Default: 2

**Distance Method** [enumeration] Distance method used to calculate the distance down to the stream. One of several ways of measuring distance may be selected: the total straight line path (Pythagoras), the horizontal component of the straight line path (horizontal), the vertical component of the straight line path (vertical), or the total surface flow path (surface).

Options:
- 0 — Pythagoras
- 1 — Horizontal
- 2 — Vertical
- 3 — Surface

Default: 1

**Check for edge contamination** [boolean] A flag that determines whether the tool should check for edge contamination. This is defined as the possibility that a value may be underestimated due to grid cells outside of the domain not being counted. In the context of Distance Down this occurs when part of a flow path traced downslope from a grid cell leaves the domain without reaching a stream grid cell. With edge contamination checking selected, the algorithm recognizes this and reports no data for the result. This is the desired effect and indicates that values for these grid cells is unknown due to it being dependent on terrain outside of the domain of data available. Edge contamination checking may be overridden in cases where you know this is not an issue or want to evaluate the distance using only the fraction of flow paths that terminate at a stream.

Default: True
Outputs

D-Infinity Drop to Stream Grid [raster] Grid containing the distance to stream calculated using the D-infinity flow model and the statistical and path methods chosen.

Algorithm ID: taudem:dinfdistdown

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

D-Infinity Distance Up

Popis

This tool calculates the distance from each grid cell up to the ridge cells along the reverse D-infinity flow directions. Ridge cells are defined to be grid cells that have no contribution from grid cells further upslope. Given the convergence of multiple flow paths at any grid cell, any given grid cell can have multiple upslope ridge cells. There are three statistical methods that this tool can use: maximum distance, minimum distance and waited flow average over these flow paths. A variant on the above is to consider only grid cells that contribute flow with a proportion greater than a user specified threshold (t) to be considered as upslope of any given grid cell. Setting t=0.5 would result in only one flow path from any grid cell and would give the result equivalent to a D8 flow model, rather than D-infinity flow model, where flow is proportioned between two downslope grid cells. Finally there are several different optional paths that can be measured: the total straight line path (Pythagoras), the horizontal component of the straight line path, the vertical component of the straight line path, or the total surface flow path.

Parameters

D-Infinity Flow Direction Grid [raster] A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This can be created by the tool „D-Infinity Flow Directions“.

Pit Filled Elevation Grid [raster] This input is a grid of elevation values. As a general rule, it is recommended that you use a grid of elevation values that have had the pits removed for this input. Pits are generally taken to be artifacts that interfere with the analysis of flow across them. This grid can be obtained as the output of the „Pit Remove“ tool, in which case it contains elevation values where the pits have been filled to the point where they just drain.

Slope Grid [raster] This input is a grid of slope values. This is measured as drop/distance and it is most often obtained as the output of the „D-Infinity Flow Directions“ tool.

Statistical Method [enumeration] Statistical method used to calculate the distance down to the stream. In the D-Infinity flow model, the outflow from each grid cell is proportioned between two downslope grid cells. Therefore, the distance from any grid cell to a stream is not uniquely defined. Flow that originates at a particular grid cell may enter the stream at a number of cells. The distance to the stream may be defined as the longest (maximum), shortest (minimum) or weighted average of the distance down to the stream.

Options:

- 0 — Minimum
- 1 — Maximum
- 2 — Average

Default: 2
Distance Method [enumeration]  Distance method used to calculate the distance down to the stream. One of several ways of measuring distance may be selected: the total straight line path (Pythagoras), the horizontal component of the straight line path (horizontal), the vertical component of the straight line path (vertical), or the total surface flow path (surface).

Options:
- 0 — Pythagoras
- 1 — Horizontal
- 2 — Vertical
- 3 — Surface

Default: 1

Proportion Threshold [number]  The proportion threshold parameter where only grid cells that contribute flow with a proportion greater than this user specified threshold (t) is considered to be upslope of any given grid cell. Setting t=0.5 would result in only one flow path from any grid cell and would give the result equivalent to a D8 flow model, rather than D-Infinity flow model, where flow is proportioned between two downslope grid cells.

Default: 0.5

Check for edge contamination [boolean]  A flag that determines whether the tool should check for edge contamination. This is defined as the possibility that a value may be underestimated due to grid cells outside of the domain not being counted.

Default: True

Outputs

D-Infinity Distance Up [raster]  Grid containing the distances up to the ridge calculated using the D-Infinity flow model and the statistical and path methods chosen.

Algorithm ID: taudem:dinfdistup

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

D-Infinity Reverse Accumulation

Popis

This works in a similar way to evaluation of weighted Contributing area, except that the accumulation is by propagating the weight loadings upslope along the reverse of the flow directions to accumulate the quantity of weight loading downslope from each grid cell. The function also reports the maximum value of the weight loading downslope from each grid cell in the Maximum Downslope grid.

This function is designed to evaluate and map the hazard due to activities that may have an effect downslope. The example is land management activities that increase runoff. Runoff is sometimes a trigger for landslides or debris flows, so the weight grid here could be taken as a terrain stability map. Then the reverse accumulation provides a measure of the amount of unstable terrain downslope from each grid cell, as an indicator of the danger of activities that may increase runoff, even though there may be no potential for any local impact.
**Parameters**

**D-Infinity Flow Direction Grid [raster]** A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This can be created by the tool „D-Infinity Flow Directions“.

**Weight Grid [raster]** A grid giving weights (loadings) to be used in the accumulation.

**Outputs**

**Reverse Accumulation Grid [raster]** The grid giving the result of the „Reverse Accumulation“ function. This works in a similar way to evaluation of weighted Contributing area, except that the accumulation is by propagating the weight loadings upslope along the reverse of the flow directions to accumulate the quantity of loading downslope from each grid cell.

**Maximum Downslope Grid [raster]** The grid giving the maximum of the weight loading grid downslope from each grid cell.

**Algorithm ID:** `taudem:dinfrevaccum`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.
This function is designed to calculate the transport and deposition of a substance (e.g. sediment) that may be limited by both supply and the capacity of the flow field to transport it. This function accumulates substance flux (e.g. sediment transport) subject to the rule that transport out of any grid cell is the minimum between supply and transport capacity, $T_{cap}$. The total supply at a grid cell is calculated as the sum of the transport in from upslope grid cells, $T_{in}$, plus the local supply contribution, $E$ (e.g. erosion). This function also outputs deposition, $D$, calculated as total supply minus actual transport.

$$T_{out} = \min(E + \sum T_{in}, T_{cap})$$

$$D = E + \sum T_{in} - T_{out}$$

Here $E$ is the supply. $T_{out}$ at each grid cell becomes $T_{in}$ for downslope grid cells and is reported as Transport limited accumulation ($tla$). $D$ is deposition ($tdep$). The function provides the option to evaluate concentration of a compound (contaminant) adhered to the transported substance. This is evaluated as follows:

$$L_{in} = \sum T_{in} C_{in}$$

Where $L_{in}$ is the total incoming compound loading and $C_{in}$ and $T_{in}$ refer to the Concentration and Transport entering from each upslope grid cell.

$$T_{out} < \sum T_{in}$$

If

else

where $C_s$ is the concentration supplied locally and the difference in the second term on the right represents the additional supply from the local grid cell. Then, $C_{out}$ at each grid cell comprises is the concentration grid output from this function.

If the outlets shapefile is used the tool only evaluates that part of the domain that contributes flow to the locations given by the shapefile.

Transport limited accumulation is useful for modeling erosion and sediment delivery, including the spatial dependence of sediment delivery ratio and contaminant that adheres to sediment.
\[ L_{\text{out}} = L_{\text{in}} \left( T_{\text{out}} / \sum T_{\text{in}} \right) \]

\[ L_{\text{out}} = L_{\text{in}} + C_s \left( T_{\text{out}} - \sum T_{\text{in}} \right) \]

**Parameters**

**D-Infinity Flow Direction Grid** A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This can be created by the tool „D-Infinity Flow Directions“.

**Supply Grid** A grid giving the supply (loading) of material to a transport limited accumulation function. In the application to erosion, this grid would give the erosion detachment, or sediment supplied at each grid cell.

**Transport Capacity Grid** A grid giving the transport capacity at each grid cell for the transport limited accumulation function. In the application to erosion this grid would give the transport capacity of the carrying flow.

**Input Concentration Grid** A grid giving the concentration of a compound of interest in the supply to the transport limited accumulation function. In the application to erosion, this grid would give the concentration of say phosphorous adhered to the eroded sediment.

**Outlets Shapefile** Optional

This optional input is a point shapefile defining outlets of interest. If this file is used, the tool will only evaluate the area upslope of these outlets.

**Check for edge contamination** This option determines whether the tool should check for edge contamination. Edge contamination is defined as the possibility that a value may be underestimated due to grid cells outside of the domain not being considered when determining the result.

Default: True

**Outputs**

**Transport Limited Accumulation Grid** This grid is the weighted accumulation of supply accumulated respecting the limitations in transport capacity and reports the transport rate calculated by accumulating the substance flux subject to the rule that the transport out of any grid cell is the minimum of the total supply (local supply plus transport in) to that grid cell and the transport capacity.

**Deposition Grid** A grid giving the deposition resulting from the transport limited accumulation. This is the residual from the transport in to each grid cell minus the transport capacity out of the grid cell. The deposition grid is calculated as the transport in + the local supply - the transport out.

**Output Concentration Grid** If an input concentration in supply grid is given, then this grid is also output and gives the concentration of a compound (contaminant) adhered or bound to the transported substance (e.g. sediment) is calculated.

\[ C_{\text{out}} = L_{\text{out}} / T_{\text{out}} \]
Algorithm ID: unknown

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

D-Infinity Transport Limited Accumulation

**Popis**

This function is designed to calculate the transport and deposition of a substance (e.g. sediment) that may be limited by both supply and the capacity of the flow field to transport it. This function accumulates substance flux (e.g. sediment transport) subject to the rule that transport out of any grid cell is the minimum between supply and transport capacity, $T_{cap}$. The total supply at a grid cell is calculated as the sum of the transport in from upslope grid cells, $T_{in}$, plus the local supply contribution, $E$ (e.g. erosion). This function also outputs deposition, $D$, calculated as total supply minus actual transport.

\[
T_{out} = \min(E + \sum T_{in}, T_{cap})
\]

\[
D = E + \sum T_{in} - T_{out}
\]

Here $E$ is the supply. $T_{out}$ at each grid cell becomes $T_{in}$ for downslope grid cells and is reported as Transport limited accumulation ($tla$). $D$ is deposition ($tdep$). The function provides the option to evaluate concentration of a compound (contaminant) adhered to the transported substance. This is evaluated as follows:

\[
L_{in} = \sum T_{in} C_{in}
\]

Where $L_{in}$ is the total incoming compound loading and $C_{in}$ and $T_{in}$ refer to the Concentration and Transport entering from each upslope grid cell.

\[
T_{out} < \sum T_{in}
\]

If

else

where $C_{s}$ is the concentration supplied locally and the difference in the second term on the right represents the additional supply from the local grid cell. Then,

$C_{out}$ at each grid cell comprises is the concentration grid output from this function.
If the outlets shapefile is used the tool only evaluates that part of the domain that contributes flow to the locations given by the shapefile.

Transport limited accumulation is useful for modeling erosion and sediment delivery, including the spatial dependence of sediment delivery ratio and contaminant that adheres to sediment.

Parameters

D-Infinity Flow Direction Grid [raster] A grid giving flow direction by the D-infinity method. Flow direction is measured in radians, counter clockwise from east. This can be created by the tool "D-Infinity Flow Directions".

Supply Grid [raster] A grid giving the supply (loading) of material to a transport limited accumulation function. In the application to erosion, this grid would give the erosion detachment, or sediment supplied at each grid cell.

Transport Capacity Grid [raster] A grid giving the transport capacity at each grid cell for the transport limited accumulation function. In the application to erosion this grid would give the transport capacity of the carrying flow.

Outlets Shapefile [vector: point] Optional
This optional input is a point shapefile defining outlets of interest. If this file is used, the tool will only evaluate the area upslope of these outlets.

Check for edge contamination [boolean] This option determines whether the tool should check for edge contamination. Edge contamination is defined as the possibility that a value may be underestimated due to grid cells outside of the domain not being considered when determining the result.

Default: True

Outputs

Transport Limited Accumulation Grid [raster] This grid is the weighted accumulation of supply accumulated respecting the limitations in transport capacity and reports the transport rate calculated by accumulating the substance flux subject to the rule that the transport out of any grid cell is the minimum of the total supply (local supply plus transport in) to that grid cell and the transport capacity.

Deposition Grid [raster] A grid giving the deposition resulting from the transport limited accumulation. This is the residual from the transport in to each grid cell minus the transport capacity out of the grid cell. The deposition grid is calculated as the transport in + the local supply - the transport out.

Algorithm ID: tauDEM:dinftranslimaccum

\[
L_{out} = L_{in} \left( \frac{T_{out}}{\sum T_{in}} \right)
\]

\[
L_{out} = L_{in} + C_s \left( T_{out} - \sum T_{in} \right)
\]

\[
C_{out} = L_{out} / T_{out}
\]
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

### D-Infinity Upslope Dependence

#### Popis

The D-Infinity Upslope Dependence tool quantifies the amount each grid cell in the domain contributes to a destination set of grid cells. D-Infinity flow directions proportion flow from each grid cell between multiple downslope grid cells. Following this flow field downslope the amount of flow originating at each grid cell that reaches the destination zone is defined. Upslope influence is evaluated using a downslope recursion, examining grid cells downslope from each grid cell, so that the map produced identifies the area upslope where flow through the destination zone originates, or the area it depends on, for its flow.

The figures below illustrate the amount each source point in the domain $x$ (blue) contributes to the destination point or zone $y$ (red). If the indicator weighted contributing area function is denoted $I(y; x)$ giving the weighted contribution using a unit value (1) from specific grid cells $y$ to grid cells $x$, then the upslope dependence is: $D(x; y) = I(y; x)$.

This is useful for example to track where flow or a flow related substance or contaminant that enters a destination area may come from.
Parameters

**D-Infinity Flow Direction Grid [raster]** A grid giving flow direction by the D-Infinity method where the flow direction angle is determined as the direction of the steepest downward slope on the eight triangular facets formed in a 3x3 grid cell window centered on the grid cell of interest. This grid can be produced using the „D-Infinity Flow Direction“ tool.

**Destination Grid [raster]** A grid that encodes the destination zone that may receive flow from upslope. This grid must be 1 inside the zone y and 0 over the rest of the domain.

Outputs

**Output Upslope Dependence Grid [raster]** A grid quantifying the amount each source point in the domain contributes to the zone defined by the destination grid.

Algorithm ID: taudem:dinfupdependence

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Slope Average Down

**Popis**

This tool computes slope in a D8 downslope direction averaged over a user selected distance. Distance should be specified in horizontal map units.

Parameters

**D8 Flow Direction Grid [raster]** This input is a grid of flow directions that are encoded using the D8 method where all flow from a cell goes to a single neighboring cell in the direction of steepest descent. This grid can be obtained as the output of the „D8 Flow Directions“ tool.

**Pit Filled Elevation Grid [raster]** This input is a grid of elevation values. As a general rule, it is recommended that you use a grid of elevation values that have had the pits removed for this input. Pits are generally taken to be artifacts that interfere with the analysis of flow across them. This grid can be obtained as the output of the „Pit Remove“ tool, in which case it contains elevation values where the pits have been filled to the point where they just drain.

**Downslope Distance [number]** Input parameter of downslope distance over which to calculate the slope (in horizontal map units).

Default: 50
Outs

Slope Average Down Grid [raster] This output is a grid of slopes calculated in the D8 downslope direction, averaged over the selected distance.

Algorithm ID: taudem:slopeavedown

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Slope Over Area Ratio

Popis

Calculates the ratio of the slope to the specific catchment area (contributing area). This is algebraically related to the more common \(\ln(\frac{a}{\tan \beta})\) wetness index, but contributing area is in the denominator to avoid divide by 0 errors when slope is 0.

Parameters

Slope Grid [raster] A grid of slope. This grid can be generated using either the „D8 Flow Directions“ tool or the „D-Infinity Flow Directions“ tool.

Specific Catchment Area Grid [raster] A grid giving the contributing area value for each cell taken as its own contribution plus the contribution from upslope neighbors that drain into it. Contributing area is counted in terms of the number of grid cells (or summation of weights). This grid can be generated using either the „D8 Contributing Area“ tool or the „D-Infinity Contributing Area“ tool.

Outputs

Slope Divided By Area Ratio Grid [raster] A grid of the ratio of slope to specific catchment area (contributing area). This is algebraically related to the more common \(\ln(\frac{a}{\tan \beta})\) wetness index, but contributing area is in the denominator to avoid divide by 0 errors when slope is 0.

Algorithm ID: taudem:slopearearatio

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Topographic wetness index

Popis

Calculates the topographic wetness index (TWI).

Parameters

Slope [raster] A grid of slope. This grid can be generated using either the „D8 Flow Directions“ tool or the „D-Infinity Flow Directions“ tool.

Specific catchment area [raster] A grid giving the contributing area value for each cell taken as its own contribution plus the contribution from upslope neighbors that drain in to it. Contributing area is counted in terms of the number of grid cells (or summation of weights). This grid can be generated using either the „D8 Contributing Area“ tool or the „D-Infinity Contributing Area“ tool.

Outputs

Wetness index [raster] A grid of the wetness index (TWI).

Algorithm ID: taudem:twi

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

23.4.3 Stream Network Analysis

Connect down

Popis

Parameters

D8 flow directions [raster] A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can be obtained as the output of the „D8 Flow Directions“ tool.

D8 contribution area [raster]

Watershed [raster]

Grid cells move to downstream [number]

Outlets [vector: point] Optional

A point shape file defining outlets of interest. If this input file is used, only the area upslope of these outlets will be evaluated by the tool.
Outputs

Extreme Upslope Values Grid [raster] A grid of the maximum/minimum upslope values.

Algorithm ID: taudem:connectdown

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

D8 Extreme Upslope Value

Popis

Evaluates the extreme (either maximum or minimum) upslope value from an input grid based on the D8 flow model. This is intended initially for use in stream raster generation to identify a threshold of the slope times area product that results in an optimum (according to drop analysis) stream network.

If the optional outlet point shapefile is used, only the outlet cells and the cells upslope (by the D8 flow model) of them are in the domain to be evaluated.

By default, the tool checks for edge contamination. This is defined as the possibility that a result may be underestimated due to grid cells outside of the domain not being counted. This occurs when drainage is inwards from the boundaries or areas with „no data“ values for elevation. The algorithm recognizes this and reports „no data“ for the result for these grid cells. It is common to see streaks of „no data“ values extending inwards from boundaries along flow paths that enter the domain at a boundary. This is the desired effect and indicates that the result for these grid cells is unknown due to it being dependent on terrain outside of the domain of data available. Edge contamination checking may be turned off in cases where you know this is not an issue or want to ignore these problems, if for example, the DEM has been clipped along a watershed outline.

Parameters

D8 Flow Directions Grid [raster] A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can be obtained as the output of the „D8 Flow Directions“ tool.

Upslope Values Grid [raster] This is the grid of values of which the maximum or minimum upslope value is selected. The values most commonly used are the slope times area product needed when generating stream rasters according to drop analysis.

Outlets Shapefile [vector: point] Optional

A point shape file defining outlets of interest. If this input file is used, only the area upslope of these outlets will be evaluated by the tool.

Check for edge contamination [boolean] A flag that indicates whether the tool should check for edge contamination.

Default: True

Use max upslope value [boolean] A flag to indicate whether the maximum or minimum upslope value is to be calculated.

Default: True
Outputs

**Extreme Upslope Values Grid** [raster] A grid of the maximum/minimum upslope values.

**Algorithm ID:** taudem:d8flowpathextremeup

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**Gage Watershed**

**Popis**

Calculates Gage Watersheds Grid. Each grid cell is labeled with the identifier (from column `id`) of the gage to which it drains directly without passing through any other gages.

**Parameters**

**D8 Flow Directions Grid** [raster] A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can be obtained as the output of the **D8 Flow Directions** tool.

**Gages Shapefile** [vector: point] A point shapefile defining the gages to which watersheds will be delineated. This shapefile should have a column `id`. Grid cells draining directly to each point in this shapefile will be labeled with this id.

**Outputs**

**Gage Watershed Grid** [raster] A grid identifies each gage watershed. Each grid cell is labeled with the identifier (from column `id`) of the gage to which it drains directly without passing through any other gages.

**Downstream Identifiers File** [file] Text file giving watershed downslope connectivity

**Algorithm ID:** taudem:gagewatershed

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**Length Area Stream Source**

**Popis**

Creates an indicator grid (1, 0) that evaluates \( A \geq (M)(L_y) \) based on upslope path length, D8 contributing area grid inputs, and parameters \( M \) and \( y \). This grid indicates likely stream source grid cells. This is an experimental method with theoretical basis in Hack’s law which states that for streams \( L \sim A^{0.6} \). However for hillslopes with parallel flow \( L \sim A \). So a transition from hillslopes to streams may be represented by \( L \sim A^{0.8} \) suggesting identifying grid cells as stream cells if \( A > M \ (L^{(1/0.8)}) \).
Parameters

Length Grid [raster] A grid of the maximum upslope length for each cell. This is calculated as the length of the flow path from the furthest cell that drains to each cell. Length is measured between cell centers taking into account cell size and whether the direction is adjacent or diagonal. It is this length \( L \) that is used in the formula, \( A > (M) (L^y) \), to determine which cells are considered stream cells. This grid can be obtained as an output from the „Grid Network“ tool.

Contributing Area Grid [raster] A grid of contributing area values for each cell that were calculated using the D8 algorithm. The contributing area for a cell is the sum of its own contribution plus the contribution from all upslope neighbors that drain to it, measured as a number of cells. This grid is typically obtained as the output of the „D8 Contributing Area“ tool. In this tool, it is the contributing area \( A \) that is compared in the formula \( A > (M) (L^y) \) to determine the transition to a stream.

Threshold [number] The multiplier threshold \( M \) parameter which is used in the formula: \( A > (M) (L^y) \), to identify the beginning of streams.

Default: 0.03

Exponent [number] The exponent \( y \) parameter which is used in the formula: \( A > (M) (L^y) \), to identify the beginning of streams. In branching systems, Hack’s law suggests that \( L = 1/M A(1/y) \) with \( 1/y = 0.6 \) (or 0.56) \( y \) about 1.7). In parallel flow systems \( L \) is proportional to \( A \) \( y \) about 1). This method tries to identify the transition between these two paradigms by using an exponent \( y \) somewhere in between \( y \) about 1.3).

Default: 1.3

Outputs

Stream Source Grid [raster] An indicator grid (1,0) that evaluates \( A >= (M)(L^y) \), based on the maximum upslope path length, the D8 contributing area grid inputs, and parameters \( M \) and \( y \). This grid indicates likely stream source grid cells.

Algorithm ID: taudem:lengtharea

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

Move Outlets To Streams

Popis

Moves outlet points that are not aligned with a stream cell from a stream raster grid, downslope along the D8 flow direction until a stream raster cell is encountered, the „max_dist“ number of grid cells are examined, or the flow path exits the domain (i.e. a „no data“ value is encountered for the D8 flow direction). The output file is a new outlets shapefile where each point has been moved to coincide with the stream raster grid, if possible. A field „dist_moved“ is added to the new outlets shapefile to indicate the changes made to each point. Points that are already on a stream cell are not moved and their „dist_moved“ field is assigned a value 0. Points that are initially not on a stream cell are moved by sliding them downslope along the D8 flow direction until one of the following occurs: a) A stream raster grid cell is encountered before traversing the „max_dist“ number of grid cells. In which case, the point is moved and the „dist_moved“ field is assigned a value indicating how many grid cells the point was moved. b) More than the „max_number“ of grid cells are traversed, or c) the traversal ends up going out of the domain (i.e., a „no data“ D8 flow direction value is encountered). In which case, the point is not moved and the „dist_moved“ field is assigned a value of -1.
Parameters

**D8 Flow Direction Grid** [raster] A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can be obtained as the output of the „D8 Flow Directions“ tool.

**Stream Raster Grid** [raster] This output is an indicator grid (1, 0) that indicates the location of streams, with a value of 1 for each of the stream cells and 0 for the remainder of the cells. This file is produced by several different tools in the „Stream Network Analysis“ toolset.

**Outlets Shapefile** [vector: point] A point shape file defining points of interest or outlets that should ideally be located on a stream, but may not be exactly on the stream due to the fact that the shapefile point locations may not have been accurately registered with respect to the stream raster grid.

**Maximum Number of Grid Cells to traverse** [number] This input parameter is the maximum number of grid cells that the points in the input outlet shapefile will be moved before they are saved to the output outlet shapefile.

Default: 50

Outputs

**Output Outlet Shapefile** [vector: point] A point shape file defining points of interest or outlets. This file has one point in it for each point in the input outlet shapefile. If the original point was located on a stream, then the point was not moved. If the original point was not on a stream, the point was moved downslope according to the D8 flow direction until it reached a stream or the maximum distance had been reached. This file has an additional field „dist_moved“ added to it which is the number of cells that the point was moved. This field is 0 if the cell was originally on a stream, -1 if it was not moved because there was not a stream within the maximum distance, or some positive value if it was moved.

Algorithm ID: taudem:moveoutletstostreams

```
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

**Peuker Douglas**

**Popis**

Creates an indicator grid (1, 0) of upward curved grid cells according to the Peuker and Douglas algorithm.

With this tool, the DEM is first smoothed by a kernel with weights at the center, sides, and diagonals. The Peuker and Douglas (1975) method (also explained in Band, 1986), is then used to identify upwardly curving grid cells. This technique flags the entire grid, then examines in a single pass each quadrant of 4 grid cells, and unflags the highest. The remaining flagged cells are deemed „upwardly curved“, and when viewed, resemble a channel network. This proto-channel network generally lacks connectivity and requires thinning, issues that were discussed in detail by Band (1986).
Parameters

**Elevation Grid [raster]** A grid of elevation values. This is usually the output of the „Pit Remove“ tool, in which case it is elevations with pits removed.

**Center Smoothing Weight [number]** The center weight parameter used by a kernel to smooth the DEM before the tool identifies upwardly curved grid cells.
   Default: 0.4

**Side Smoothing Weight [number]** The side weight parameter used by a kernel to smooth the DEM before the tool identifies upwardly curved grid cells.
   Default: 0.1

**Diagonal Smoothing Weight [number]** The diagonal weight parameter used by a kernel to smooth the DEM before the tool identifies upwardly curved grid cells.
   Default: 0.05

Outputs

**Stream Source Grid [raster]** An indicator grid (1, 0) of upward curved grid cells according to the Peuker and Douglas algorithm, and if viewed, resembles a channel network. This proto-channel network generally lacks connectivity and requires thinning, issues that were discussed in detail by Band (1986).

See also


**Algorithm ID:** `taudem:peukerdouglas`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Peuker Douglas stream

Popis

Parameters

Outputs

**Stream source [raster]** An indicator grid (1, 0) of upward curved grid cells according to the Peuker and Douglas algorithm, and if viewed, resembles a channel network. This proto-channel network generally lacks connectivity and requires thinning, issues that were discussed in detail by Band (1986).

**Algorithm ID:** `taudem:peukerdouglasstream`
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**Slope Area Combination**

**Popis**

Creates a grid of slope-area values $= (S_m) (A_n)$ based on slope and specific catchment area grid inputs, and parameters $m$ and $n$. This tool is intended for use as part of the slope-area stream raster delineation method.

**Parameters**

**Slope Grid** [raster] This input is a grid of slope values. This grid can be obtained from the „D-Infinity Flow Directions“ tool.

**Contributing Area Grid** [raster] A grid giving the specific catchment area for each cell taken as its own contribution (grid cell length or summation of weights) plus the proportional contribution from upslope neighbors that drain in to it. This grid is typically obtained from the „D-Infinity Contributing Area“ tool.

**Slope Exponent** [number] The slope exponent ($m$) parameter which will be used in the formula: $(S_m) (A_n)$, that is used to create the slope-area grid.

Default: 2

**Area Exponent** [number] The area exponent ($n$) parameter which will be used in the formula: $(S_m) (A_n)$, that is used to create the slope-area grid.

Default: 1

**Outputs**

**Slope Area Grid** [raster] A grid of slope-area values $= (S_m) (A_n)$ calculated from the slope grid, specific catchment area grid, $m$ slope exponent parameter, and $n$ area exponent parameter.

Algorithm ID: taudem:slopearea

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

**Slope area stream definition**

**Popis**

Creates a grid of slope-area values $= (S_m) (A_n)$ based on slope and specific catchment area grid inputs, and parameters $m$ and $n$. This tool is intended for use as part of the slope-area stream raster delineation method.
Parameters

D8 flow directions [raster]

D-infinity Contributing Area [raster] A grid giving the specific catchment area for each cell taken as its own contribution (grid cell length or summation of weights) plus the proportional contribution from upslope neighbors that drain in to it. This grid is typically obtained from the „D-Infinity Contributing Area“ tool.

Slope [raster] This input is a grid of slope values. This grid can be obtained from the „D-Infinity Flow Directions“ tool.

Mask grid [raster]

Outlets [vector: point]

Pit-filled grid for drop analysis [raster]

D8 contributing area for drop analysis [raster]

Slope Exponent [number] The slope exponent (m) parameter which will be used in the formula: \((Sm)(An)\), that is used to create the slope-area grid.

Default: 2

Area Exponent [number] The area exponent (n) parameter which will be used in the formula: \((Sm)(An)\), that is used to create the slope-area grid.

Default: 1

Accumulation threshold [number]

Minimum threshold [number]

Maximum threshold [number]

Number of drop thresholds [number]

Type of threshold step [enumeration].

Options:
- 0 — Logarithmic
- 1 — Linear

Default: 0

Check for edge contamination [boolean]

Select threshold by drop analysis [boolean]

Outputs

Stream raster [raster]

Slope area [raster] A grid of slope-area values = \((Sm)(An)\) calculated from the slope grid, specific catchment area grid, m slope exponent parameter, and n area exponent parameter.

Maximum upslope [raster]

Drop analysis [file]

Algorithm ID: taudem:slopeareastreamdef

```
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Stream Definition By Threshold

**Popis**

Operates on any grid and outputs an indicator (1, 0) grid identifying cells with input values >= the threshold value. The standard use is to use an accumulated source area grid to as the input grid to generate a stream raster grid as the output. If you use the optional input mask grid, it limits the domain being evaluated to cells with mask values >= 0. When you use a D-infinity contributing area grid (*sca) as the mask grid, it functions as an edge contamination mask. The threshold logic is:

\[
src = ((ssa >\text{ thresh}) & (mask >\text{ s0})) \ ? \ 1:0
\]

**Parameters**

**Accumulated Stream Source Grid** [raster] This grid nominally accumulates some characteristic or combination of characteristics of the watershed. The exact characteristic(s) varies depending on the stream network raster algorithm being used. This grid needs to have the property that grid cell values are monotonically increasing downslope along D8 flow directions, so that the resulting stream network is continuous. While this grid is often from an accumulation, other sources such as a maximum upslope function will also produce a suitable grid.

**Threshold** [number] This parameter is compared to the value in the Accumulated Stream Source grid (*ssa) to determine if the cell should be considered a stream cell. Streams are identified as grid cells for which ssa value is >= this threshold.

Default: 100

**Mask Grid** [raster] Optional

This optional input is a grid that is used to mask the domain of interest and output is only provided where this grid is >= 0. A common use of this input is to use a D-Infinity contributing area grid as the mask so that the delineated stream network is constrained to areas where D-infinity contributing area is available, replicating the functionality of an edge contamination mask.

**Outputs**

**Stream Raster Grid** [raster] This is an indicator grid (1, 0) that indicates the location of streams, with a value of 1 for each of the stream cells and 0 for the remainder of the cells.

**Algorithm ID**: taudem:threshold

```python
import processing
processing.run("taudem:threshold", {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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.
Stream definition with drop analysis

Popis

Parameters

Outputs

Algorithm ID: `taudem:streamdefdropanalysis`

```python
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 *Using processing algorithms from the console* for details on how to run processing algorithms from the Python console.

Stream Drop Analysis

Popis

Applies a series of thresholds (determined from the input parameters) to the input accumulated stream source grid (*ssa*) grid and outputs the results in the *drp.txt* file the stream drop statistics table. This function is designed to aid in the determination of a geomorphologically objective threshold to be used to delineate streams. Drop Analysis attempts to select the right threshold automatically by evaluating a stream network for a range of thresholds and examining the constant drop property of the resulting Strahler streams. Basically it asks the question: Is the mean stream drop for first order streams statistically different from the mean stream drop for higher order streams, using a T-test. Stream drop is the difference in elevation from the beginning to the end of a stream defined as the sequence of links of the same stream order. If the T-test shows a significant difference then the stream network does not obey this „law“ so a larger threshold needs to be chosen. The smallest threshold for which the T-test does not show a significant difference gives the highest resolution stream network that obeys the constant stream drop „law“ from geomorphology, and is the threshold chosen for the „objective“ or automatic mapping of streams from the DEM. This function can be used in the development of stream network rasters, where the exact watershed characteristic(s) that were accumulated in the accumulated stream source grid vary based on the method being used to determine the stream network raster.

![Stream Drop Analysis Diagram](image)

The constant stream drop „law“ was identified by Broscoe (1959). For the science behind using this to determine a stream delineation threshold, see Tarboton et al. (1991, 1992), Tarboton and Ames (2001).
Parameters

D8 Contributing Area Grid [raster] A grid of contributing area values for each cell that were calculated using the D8 algorithm. The contributing area for a cell is the sum of its own contribution plus the contribution from all upslope neighbors that drain to it, measured as a number of cells or the sum of weight loadings. This grid can be obtained as the output of the „D8 Contributing Area“ tool. This grid is used in the evaluation of drainage density reported in the stream drop table.

D8 Flow Direction Grid [raster] A grid of D8 flow directions which are defined, for each cell, as the direction of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can be obtained as the output of the „D8 Flow Directions“ tool.

Pit Filled Elevation Grid [raster] A grid of elevation values. This is usually the output of the „Pit Remove“ tool, in which case it is elevations with pits removed.

Accumulated Stream Source Grid [raster] This grid must be monotonically increasing along the downslope D8 flow directions. It it compared to a series of thresholds to determine the beginning of the streams. It is often generated by accumulating some characteristic or combination of characteristics of the watershed with the „D8 Contributing Area“ tool, or using the maximum option of the „D8 Flow Path Extreme“ tool. The exact method varies depending on the algorithm being used.

Outlets Shapefile [vector: point] A point shapefile defining the outlets upstream of which drop analysis is performed.

Minimum Threshold [number] This parameter is the lowest end of the range searched for possible threshold values using drop analysis. This technique looks for the smallest threshold in the range where the absolute value of the t-statistic is less than 2. For the science behind the drop analysis see Tarboton et al. (1991, 1992), Tarboton and Ames (2001).

Default: 5

Maximum Threshold [number] This parameter is the highest end of the range searched for possible threshold values using drop analysis. This technique looks for the smallest threshold in the range where the absolute value of the t-statistic is less than 2. For the science behind the drop analysis see Tarboton et al. (1991, 1992), Tarboton and Ames (2001).

Default: 500

Number of Threshold Values [number] The parameter is the number of steps to divide the search range into when looking for possible threshold values using drop analysis. This technique looks for the smallest threshold in the range where the absolute value of the t-statistic is less than 2. For the science behind the drop analysis see Tarboton et al. (1991, 1992), Tarboton and Ames (2001).

Default: 10

Spacing for Threshold Values [enumeration] This parameter indicates whether logarithmic or linear spacing should be used when looking for possible threshold values using drop analysis.

Options:

• 0 — Logarithmic
• 1 — Linear

Default: 0
Outputs

D-Infinity Drop to Stream Grid [file] This is a comma delimited text file with the following header line:

```
:: Threshold, DrainDen, NoFirstOrd, NoHighOrd, MeanDFirstOrd, MeanDHighOrd, StdDevFirstOrd, StdDevHighOrd, T
```

The file then contains one line of data for each threshold value examined, and then a summary line that indicates the optimum threshold value. This technique looks for the smallest threshold in the range where the absolute value of the t-statistic is less than 2. For the science behind the drop analysis, see Tarboton et al. (1991, 1992), Tarboton and Ames (2001).

Algorithm ID: taudem:dropanalysis

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

See also


Stream Reach and Watershed

**Popis**

This tool produces a vector network and shapefile from the stream raster grid. The flow direction grid is used to connect flow paths along the stream raster. The Strahler order of each stream segment is computed. The subwatershed draining to each stream segment (reach) is also delineated and labeled with the value identifier that corresponds to the WSNO (watershed number) attribute in the Stream Reach Shapefile.

This tool orders the stream network according to the Strahler ordering system. Streams that don’t have any other streams draining in to them are order 1. When two stream reaches of different order join the order of the downstream reach is the order of the highest incoming reach. When two reaches of equal order join the downstream reach order is increased by 1. When more than two reaches join the downstream reach order is calculated as the maximum of the highest incoming reach order or the second highest incoming reach order + 1. This generalizes the common definition to cases where more than two reaches join at a point. The network topological connectivity is stored in the Stream Network Tree file, and coordinates and attributes from each grid cell along the network are stored in the Network Coordinates file.

The stream raster grid is used as the source for the stream network, and the flow direction grid is used to trace connections within the stream network. Elevations and contributing area are used to determine the elevation and contributing area attributes in the network coordinate file. Points in the outlets shapefile are used to logically split stream reaches to facilitate representing watersheds upstream and downstream of monitoring points. The program
uses the attribute field „id” in the outlets shapefile as identifiers in the Network Tree file. This tool then translates the
text file vector network representation in the Network Tree and Coordinates files into a shapefile. Further attributes
are also evaluated. The program has an option to delineate a single watershed by representing the entire area draining
to the Stream Network as a single value in the output watershed grid.

Parameters

**Pit Filled Elevation Grid**[raster] A grid of elevation values. This is usually the output of the „Pit
Remove“ tool, in which case it is elevations with pits removed.

**D8 Flow Direction Grid**[raster] A grid of D8 flow directions which are defined, for each cell, as the di-
rection of the one of its eight adjacent or diagonal neighbors with the steepest downward slope. This grid can
be obtained as the output of the „D8 Flow Directions“ tool.

**D8 Drainage Area**[raster] A grid giving the contributing area value in terms of the number of grid cells (or the
summation of weights) for each cell taken as its own contribution plus the contribution from upslope neighbors
that drain into it using the D8 algorithm. This is usually the output of the „D8 Contributing Area“ tool and
is used to determine the contributing area attribute in the Network Coordinate file.

**Stream Raster Grid**[raster] An indicator grid indicating streams, by using a grid cell value of 1 on streams
and 0 off streams. Several of the „Stream Network Analysis“ tools produce this type of grid. The Stream
Raster Grid is used as the source for the stream network.

**Outlets Shapefile as Network Nodes**[vector: point] Optional

A point shape file defining points of interest. If this file is used, the tool will only delineate the stream network
upstream of these outlets. Additionally, points in the Outlets Shapefile are used to logically split stream reaches
to facilitate representing watersheds upstream and downstream of monitoring points. This tool REQUIRES
THAT THERE BE an integer attribute field „id“ in the Outlets Shapefile, because the „id“ values are used as
identifiers in the Network Tree file.

**Delineate Single Watershed**[boolean] This option causes the tool to delineate a single watershed by
representing the entire area draining to the Stream Network as a single value in the output watershed grid.
Otherwise a separate watershed is delineated for each stream reach. Default is False (separate watershed).

Default: False

Outputs

**Stream Order Grid**[raster] The Stream Order Grid has cells values of streams ordered according to the
Strahler order system. The Strahler ordering system defines order 1 streams as stream reaches that don’t have
any other reaches draining in to them. When two stream reaches of different order join the order of the down-
stream reach is the order of the highest incoming reach. When two reaches of equal order join the downstream
reach order is increased by 1. When more than two reaches join the downstream reach order is calculated
as the maximum of the highest incoming reach order or the second highest incoming reach order + 1. This
generalizes the common definition to cases where more than two flow paths reaches join at a point.

**Watershed Grid**[raster] This output grid identified each reach watershed with a unique ID number, or in the
case where the delineate single watershed option was checked, the entire area draining to the stream network
is identified with a single ID.

**Stream Reach Shapefile**[vector: line] This output is a polyline shapefile giving the links in a stream ne-
twork. The columns in the attribute table are:

- **LINKNO** — Link Number. A unique number associated with each link (segment of channel between
  junctions). This is arbitrary and will vary depending on number of processes used

- **DSLINKNO** — Link Number of the downstream link. -1 indicates that this does not exist

- **USLINKNO1** — Link Number of first upstream link. (-1 indicates no link upstream, i.e. for a source
  link)
• **USLINKNO2** — Link Number of second upstream link. (-1 indicates no second link upstream, i.e. for a source link or an internal monitoring point where the reach is logically split but the network does not bifurcate)

• **DSNODEID** — Node identifier for node at downstream end of stream reach. This identifier corresponds to the „id“ attribute from the Outlets shapefile used to designate nodes

• **Order** — Strahler Stream Order

• **Length** — Length of the link. The units are the horizontal map units of the underlying DEM grid

• **Magnitude** — Shreve Magnitude of the link. This is the total number of sources upstream

• **DS_Cont_Ar** — Drainage area at the downstream end of the link. Generally this is one grid cell upstream of the downstream end because the drainage area at the downstream end grid cell includes the area of the stream being joined

• **Drop** — Drop in elevation from the start to the end of the link

• **Slope** — Average slope of the link (computed as drop/length)

• **Straight_L** — Straight line distance from the start to the end of the link

• **US_Cont_Ar** — Drainage area at the upstream end of the link

• **WSNO** — Watershed number. Cross reference to the *w.shp and *w grid files giving the identification number of the watershed draining directly to the link

• **DOUT_END** — Distance to the eventual outlet (i.e. the most downstream point in the stream network) from the downstream end of the link

• **DOUT_START** — Distance to the eventual outlet from the upstream end of the link

• **DOUT_MID** — Distance to the eventual outlet from the midpoint of the link

**Network Connectivity Tree [file]** This output is a text file that details the network topological connectivity is stored in the Stream Network Tree file. Columns are as follows:

• Link Number (Arbitrary — will vary depending on number of processes used)

• Start Point Number in Network coordinates (*coord.dat) file (Indexed from 0)

• End Point Number in Network coordinates (*coord.dat) file (Indexed from 0)

• Next (Downstream) Link Number. Points to Link Number. -1 indicates no links downstream, i.e. a terminal link

• First Previous (Upstream) Link Number. Points to Link Number. -1 indicates no upstream links

• Second Previous (Upstream) Link Numbers. Points to Link Number. -1 indicates no upstream links. Where only one previous link is -1, it indicates an internal monitoring point where the reach is logically split, but the network does not bifurcate

• Strahler Order of Link

• Monitoring point identifier at downstream end of link. -1 indicates downstream end is not a monitoring point

• Network magnitude of the link, calculated as the number of upstream sources (following Shreve)

**Network Coordinates [file]** This output is a text file that contains the coordinates and attributes of points along the stream network. Columns are as follows:

• X coordinate

• Y Coordinate

• Distance along channels to the downstream end of a terminal link

• Elevation

• Contributing area
Algorithm ID: taudem:streamnet

```python
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 Using processing algorithms from the console for details on how to run processing algorithms from the Python console.

## 23.5 OTB applications provider

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.
24.1 QGIS Plugins

QGIS has been designed with a plugin architecture. This allows many new features and functions to be easily added to the application. Some of the features in QGIS are actually implemented as plugins.

24.1.1 Core and External plugins

QGIS plugins are implemented either as Core Plugins or External Plugins. 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.

Poznámka: According to the plugin manager settings, QGIS main interface can display an icon on the right of the status bar to inform you that there are updates for your installed plugins or new plugins available.
24.1.2 The Plugins Dialog

The tabs in the Plugins dialog allow the user to install, uninstall and upgrade plugins in different ways. Each plugin has some metadata displayed in the right panel:

- information on whether the plugin is experimental
- description
- rating vote(s) (you can vote for your preferred plugin!)
- tags
- some useful links to the home page, tracker and code repository
- author(s)
- version available

At the top of the dialog, a Search function helps you find any plugin using metadata information (author, name, description…). It is available in nearly every tab (except Settings).

The Settings tab

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 a new plugin or a plugin update is 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.
- Show also deprecated plugins. Because they use functions that are no longer available in QGIS, these plugins are set deprecated and generally unsuitable for production use. They appear among invalid plugins list.

By default, QGIS provides you with its official plugin repository with the URL https://plugins.qgis.org/plugins/plugins.xml?qgis=3.0 (in case of QGIS 3.0) in the Plugin repositories section. 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 Authentication 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.

The All tab

In the All tab, all the available plugins are listed, including both core and external plugins. Use Upgrade All to look for new versions of the plugins. Furthermore, you can use Install Plugin if a plugin is listed but not installed, Uninstall Plugin as well as Reinstall Plugin if a plugin is installed. An installed plugin can be temporarily de/activated using the checkbox.
Obr. 24.1: The 🌞 Settings tab

Obr. 24.2: The 🌊 All tab
The Installed tab

In the **Installed** tab, you'll find listed the Core plugins, that you can not uninstall. You can extend this list with external plugins that can be uninstalled and reinstalled any time, using the **Uninstall Plugin** and **Reinstall Plugin** buttons. You can **Upgrade All** the plugins here as well.

![The Installed tab](image1)

---

Obr. 24.3: The **Installed** tab

The Not installed tab

The **Not installed** tab lists all plugins available that are not installed. You can use the **Install Plugin** button to implement a plugin into QGIS.

The Upgradeable and New tabs

The **Upgradeable** and **New** tabs are enabled when new plugins are added to the repository or a new version of an installed plugin is released. If you activated **Show also experimental plugins** in the **Settings** menu, those also appear in the list giving you opportunity to early test upcoming tools.

Installation can be done with the **Install Plugin**, **Upgrade Plugin** or **Upgrade All** buttons.

The Invalid tab

The **Invalid** tab lists all installed plugins that are currently broken for any reason (missing dependency, errors while loading, incompatible functions with QGIS version...). You can try the **Reinstall Plugin** button to fix an invalidated plugin but most of the times the fix will be elsewhere (install some libraries, look for another compatible plugin or help to upgrade the broken one).
Obr. 24.4: The Not installed tab

Obr. 24.5: The Upgradeable tab
Obr. 24.6: The 🔄 Invalid tab

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.

Obr. 24.7: The 🔄 Install from zip tab
24.2 Using QGIS Core Plugins

24.2.1 Coordinate Capture Plugin

The coordinate capture plugin is easy to use and provides the ability to display coordinates on the map canvas for two selected coordinate reference systems (CRS).

![Coordinate Capture Plugin](image)

Obr. 24.8: Coordinate Capture Plugin

1. Start QGIS, select Properties... from the Project menu and click on the CRS tab. As an alternative, you can also click on the **CRS** icon in the lower right-hand corner of the status bar.
2. Select a projected coordinate system of your choice (see also Práce s projekcemi).
3. Activate the coordinate capture plugin in the Plugin Manager (see The Plugins Dialog) and ensure that the dialog is visible by going to View Panels and ensuring that Coordinate Capture is enabled. The coordinate capture dialog appears as shown in Figure figure_coordinate_capture. Alternatively, you can also look for Vector Coordinate Capture.
4. Click on the **Click to use for coordinate display** icon and select a different CRS from the one you selected above.
5. To start capturing coordinates, click on Start Capture. You can now click anywhere on the map canvas and the plugin will show the coordinates for both of your selected CRS.
6. To enable mouse coordinate tracking, click the **mouse tracking** icon.
7. You can also copy selected coordinates to the clipboard.

24.2.2 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.

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....
Obr. 24.9: DB Manager dialog
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 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.

---

**Obr. 24.10: Executing SQL queries in the DB Manager SQL window**

---

**Poznámka:** 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.
24.2.3 eVis Plugin

(This section is derived from Horning, N., K. Koy, P. Ersts. 2009. eVis (v1.1.0) User’s Guide. American Museum of Natural History, Center for Biodiversity and Conservation. Available from https://www.amnh.org/research/center-for-biodiversity-conservation/capacity-development/biodiversity-informatics, and released under the GNU FDL.)

The Biodiversity Informatics Facility at the American Museum of Natural History’s (AMNH) Center for Biodiversity and Conservation (CBC) has developed the Event Visualization Tool (eVis), another software tool to add to the suite of conservation monitoring and decision support tools for guiding protected area and landscape planning. This plugin enables users to easily link geocoded (i.e., referenced with latitude and longitude or X and Y coordinates) photographs, and other supporting documents, to vector data in QGIS.

eVis is now automatically installed and enabled in new versions of QGIS, and as with all plugins, it can be disabled and enabled using the Plugin Manager (see The Plugins Dialog).

The eVis plugin is made up of three modules: the ‘Database Connection tool’, 'Event ID tool', and the 'Event Browser'. These work together to allow viewing of geocoded photographs and other documents that are linked to features stored in vector files, databases, or spreadsheets.

Event Browser

The Event Browser module provides the functionality to display geocoded photographs that are linked to vector features displayed in the QGIS map window. Point data, for example, can be from a vector file that can be input using QGIS or it can be from the result of a database query. The vector feature must have attribute information associated with it to describe the location and name of the file containing the photograph and, optionally, the compass direction the camera was pointed when the image was acquired. Your vector layer must be loaded into QGIS before running the Event Browser.

Launch the Event Browser module

To launch the Event Browser module, click on Database eVis eVis Event Browser. This will open the Generic Event Browser window.

The Event Browser window has three tabs displayed at the top of the window. The Display tab is used to view the photograph and its associated attribute data. The Options tab provides a number of settings that can be adjusted to control the behavior of the eVis plugin. Lastly, the Configure External Applications tab is used to maintain a table of file extensions and their associated application to allow eVis to display documents other than images.

Understanding the Display window

To see the Display window, click on the Display tab in the Event Browser window. The Display window is used to view geocoded photographs and their associated attribute data.

A. Display window: A window where the photograph will appear.

B. Zoom in button: Zoom in to see more detail. If the entire image cannot be displayed in the display window, scroll bars will appear on the left and bottom sides of the window to allow you to pan around the image.

C. Zoom out button: Zoom out to see more area.

D. Zoom to full extent button: Displays the full extent of the photograph.

E. Attribute information window: All of the attribute information for the point associated with the photograph being viewed is displayed here. If the file type being referenced in the displayed record is not an image but is of a file type defined in the Configure External Applications tab, then when you double-click on the value of the field containing the path to the file, the application to open the file will be launched to view or hear the contents of the file. If the file extension is recognized, the attribute data will be displayed in green.
Obr. 24.11: The eVis display window
F. **Navigation buttons**: Use the Previous and Next buttons to load the previous or next feature when more than one feature is selected.

**Understanding the Options window**

![eVis Options window](image)

**Obr. 24.12: The eVis Options window**

A. **File path**: A drop-down list to specify the attribute field that contains the directory path or URL for the photographs or other documents being displayed. If the location is a relative path, then the checkbox must be clicked. The base path for a relative path can be entered in the **Base Path** text box below. Information about the different options for specifying the file location are noted in the section **Specifying the location and name of a photograph** below.

B. **Compass bearing**: A drop-down list to specify the attribute field that contains the compass bearing associated with the photograph being displayed. If compass bearing information is available, it is necessary to click the checkbox below the drop-down menu title.

C. **Compass offset**: Compass offsets can be used to compensate for declination (to adjust bearings collected using magnetic bearings to true north bearings). Click the **Manual** radio button to enter the offset in the text box or click the **From Attribute** radio button to select the attribute field containing the offsets. For both of these options, east declinations should be entered using positive values, and west declinations should use negative values.

D. **Directory base path**: The base path onto which the relative path defined in **Figure_eVis_options** (A) will be appended.

E. **Replace path**: If this checkbox is checked, only the file name from A will be appended to the base path.
F. **Apply rule to all documents**: If checked, the same path rules that are defined for photographs will be used for non-image documents such as movies, text documents, and sound files. If not checked, the path rules will only apply to photographs, and other documents will ignore the base path parameter.

G. **Remember settings**: If the checkbox is checked, the values for the associated parameters will be saved for the next session when the window is closed or when the *Save* button below is pressed.

H. **Reset values**: Resets the values on this line to the default setting.

I. **Restore defaults**: This will reset all of the fields to their default settings. It has the same effect as clicking all of the *Reset* buttons.

J. **Save**: This will save the settings without closing the *Options* pane.

### Understanding the Configure External Applications window

Obr. 24.13: The *eVis* External Applications window

A. **File reference table**: A table containing file types that can be opened using *eVis*. Each file type needs a file extension and the path to an application that can open that type of file. This provides the capability of opening a broad range of files such as movies, sound recordings, and text documents instead of only images.

B. **Add new file type**: Add a new file type with a unique extension and the path for the application that can open the file.

C. **Delete current row**: Delete the file type highlighted in the table and defined by a file extension and a path to an associated application.

### Specifying the location and name of a photograph

The location and name of the photograph can be stored using an absolute or relative path, or a URL if the photograph is available on a web server. Examples of the different approaches are listed in Table *evis_examples*.

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>FILE</th>
<th>BEARING</th>
</tr>
</thead>
<tbody>
<tr>
<td>780596</td>
<td>1784017</td>
<td>C:\Workshop\eVis_Data\groundphotos\DSC_0168.JPG</td>
<td>275</td>
</tr>
<tr>
<td>780596</td>
<td>1784017</td>
<td>/groundphotos/DSC_0169.JPG</td>
<td>80</td>
</tr>
<tr>
<td>780819</td>
<td>1784015</td>
<td><a href="https://biodiversityinformatics.amnh.org/%5C">https://biodiversityinformatics.amnh.org/\</a></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>evis_testdata/DSC_0170.JPG</td>
<td></td>
</tr>
<tr>
<td>780596</td>
<td>1784017</td>
<td>pdf:<a href="https://www.testsite.com/attachments.php?%5C">https://www.testsite.com/attachments.php?\</a></td>
<td>76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>attachment_id-12</td>
<td></td>
</tr>
</tbody>
</table>
Specifying the location and name of other supporting documents

Supporting documents such as text documents, videos, and sound clips can also be displayed or played by eVis. To do this, it is necessary to add an entry in the file reference table that can be accessed from the Configure External Applications window in the Generic Event Browser that matches the file extension to an application that can be used to open the file. It is also necessary to have the path or URL to the file in the attribute table for the vector layer. One additional rule that can be used for URLs that don’t contain a file extension for the document you want to open is to specify the file extension before the URL. The format is — file extension:URL. The URL is preceded by the file extension and a colon; this is particularly useful for accessing documents from wikis and other web sites that use a database to manage the web pages (see Table evis_examples).

Using the Event Browser

When the Event Browser window opens, a photograph will appear in the display window if the document referenced in the vector file attribute table is an image and if the file location information in the Options window is properly set. If a photograph is expected and it does not appear, it will be necessary to adjust the parameters in the Options window.

If a supporting document (or an image that does not have a file extension recognized by eVis) is referenced in the attribute table, the field containing the file path will be highlighted in green in the attribute information window if that file extension is defined in the file reference table located in the Configure External Applications window. To open the document, double-click on the green-highlighted line in the attribute information window. If a supporting document is referenced in the attribute information window and the file path is not highlighted in green, then it will be necessary to add an entry for the file's filename extension in the Configure External Applications window. If the file path is highlighted in green but does not open when double-clicked, it will be necessary to adjust the parameters in the Options window so the file can be located by eVis.

If no compass bearing is provided in the Options window, a red asterisk will be displayed on top of the vector feature that is associated with the photograph being displayed. If a compass bearing is provided, then an arrow will appear pointing in the direction indicated by the value in the compass bearing display field in the Event Browser window. The arrow will be centered over the point that is associated with the photograph or other document.

To close the Event Browser window, click on the Close button from the Display window.

Event ID Tool

The „Event ID“ module allows you to display a photograph by clicking on a feature displayed in the QGIS map window. The vector feature must have attribute information associated with it to describe the location and name of the file containing the photograph and, optionally, the compass direction the camera was pointed when the image was acquired. This layer must be loaded into QGIS before running the „Event ID“ tool.

Launch the Event ID module

To launch the „Event ID“ module, either click on the Event ID icon or click on Database eVis Event ID Tool. This will cause the cursor to change to an arrow with an „i“ on top of it signifying that the ID tool is active.

To view the photographs linked to vector features in the active vector layer displayed in the QGIS map window, move the Event ID cursor over the feature and then click the mouse. After clicking on the feature, the Event Browser window is opened and the photographs on or near the clicked locality are available for display in the browser. If more than one photograph is available, you can cycle through the different features using the Previous and Next buttons. The other controls are described in the Event Browser section of this guide.
Database connection

The 'Database Connection' module provides tools to connect to and query a database or other ODBC resource, such as a spreadsheet.

eVis can directly connect to the following types of databases: PostgreSQL, MySQL, and SQLite; it can also read from ODBC connections (e.g., MS Access). When reading from an ODBC database (such as an Excel spreadsheet), it is necessary to configure your ODBC driver for the operating system you are using.

Launch the Database Connection module

To launch the 'Database Connection' module, either click on the appropriate icon or click on Database eVis Database Connection. This will launch the Database Connection window. The window has three tabs: Predefined Queries, Database Connection, and SQL Query. The Output Console window at the bottom of the window displays the status of actions initiated by the different sections of this module.

Connect to a database

Click on the Database Connection tab to open the database connection interface. Next, use the Database Type combo box to select the type of database that you want to connect to. If a password or username is required, that information can be entered in the Username and Password textboxes.

Enter the database host in the Database Host textbox. This option is not available if you selected 'MS Access' as the database type. If the database resides on your desktop, you should enter "localhost".

Enter the name of the database in the Database Name textbox. If you selected 'ODBC' as the database type, you need to enter the data source name.

When all of the parameters are filled in, click on the Connect button. If the connection is successful, a message will be written in the Output Console window stating that the connection was established. If a connection was not established, you will need to check that the correct parameters were entered above.

A. Database Type: A drop-down list to specify the type of database that will be used.
B. Database Host: The name of the database host.
C. Port: The port number if a MySQL or PostgreSQL database type is selected.
D. Database Name: The name of the database.
E. Connect: A button to connect to the database using the parameters defined above.
F. Output Console: The console window where messages related to processing are displayed.
G. Username: Username for use when a database is password protected.
H. Password: Password for use when a database is password protected.
I. Predefined Queries: Tab to open the „Predefined Queries“ window.
J. Database Connection: Tab to open the „Database Connection“ window.
K. SQL Query: Tab to open the „SQL Query“ window.
L. Help: Displays the online help.
M. OK: Closes the main „Database Connection“ window.
Running SQL queries

SQL queries are used to extract information from a database or ODBC resource. In eVis, the output from these queries is a vector layer added to the QGIS map window. Click on the SQL Query tab to display the SQL query interface. SQL commands can be entered in this text window. A helpful tutorial on SQL commands is available at https://www.w3schools.com/sql. For example, to extract all of the data from a worksheet in an Excel file, select * from [sheet1$] where sheet1 is the name of the worksheet.

Click on the Run Query button to execute the command. If the query is successful, a Database File Selection window will be displayed. If the query is not successful, an error message will appear in the Output Console window.

In the Database File Selection window, enter the name of the layer that will be created from the results of the query in the Name of New Layer textbox.

A. SQL Query Text Window: A screen to type SQL queries.
B. Run Query: Button to execute the query entered in the SQL Query Window.
C. Console Window: The console window where messages related to processing are displayed.
D. Help: Displays the online help.
E. OK: Closes the main Database Connection window.

Use the X Coordinate and Y Coordinate combo boxes to select the fields from the database that stores the X (or longitude) and Y (or latitude) coordinates. Clicking on the OK button causes the vector layer created from the SQL query to be displayed in the QGIS map window.

To save this vector file for future use, you can use the QGIS 'Save as...' command that is accessed by right-clicking on the layer name in the QGIS map legend and then selecting 'Save as...'

Tip: Creating a vector layer from a Microsoft Excel Worksheet
When creating a vector layer from a Microsoft Excel Worksheet, you might see that unwanted zeros ("0") have been inserted in the attribute table rows beneath valid data. This can be caused by deleting the values for these cells in Excel using the **Backspace** key. To correct this problem, you need to open the Excel file (you'll need to close QGIS if you are connected to the file, to allow you to edit the file) and then use **Edit Delete** to remove the blank rows from the file. To avoid this problem, you can simply delete several rows in the Excel Worksheet using **Edit Delete** before saving the file.

### Running predefined queries

With predefined queries, you can select previously written queries stored in XML format in a file. This is particularly helpful if you are not familiar with SQL commands. Click on the **Predefined Queries** tab to display the predefined query interface.

To load a set of predefined queries, click on the **Open File** icon. This opens the **Open File** window, which is used to locate the file containing the SQL queries. When the queries are loaded, their titles as defined in the XML file will appear in the drop-down menu located just below the **Open File** icon. The full description of the query is displayed in the text window under the drop-down menu.

Select the query you want to run from the drop-down menu and then click on the **SQL Query** tab to see that the query has been loaded into the query window. If it is the first time you are running a predefined query or are switching databases, you need to be sure to connect to the database.

Click on the **Run Query** button in the **SQL Query** tab to execute the command. If the query is successful, a **Database File Selection** window will be displayed. If the query is not successful, an error message will appear in the **Output Console** window.

- **Open File**: Launches the „Open File“ file browser to search for the XML file holding the predefined queries.
Obr. 24.16: The eVis Predefined Queries tab

B. **Predefined Queries**: A drop-down list with all of the queries defined by the predefined queries XML file.

C. **Query description**: A short description of the query. This description is from the predefined queries XML file.

D. **Console Window**: The console window where messages related to processing are displayed.

E. **Help**: Displays the online help.

F. **OK**: Closes the main „Database Connection“ window.

**XML format for eVis predefined queries**

The XML tags read by eVis
Tag | Popis
----|---------
**query** | Defines the beginning and end of a query statement.
**shortdescription** | A short description of the query that appears in the eVis drop-down menu.
**description** | A more detailed description of the query displayed in the Predefined Query text window.
**databasetype** | The database type, defined in the Database Type drop-down menu in the Database Connection tab.
**databaseport** | The port as defined in the Port text box in the Database Connection tab.
**databasename** | The database name as defined in the Database Name text box in the Database Connection tab.
**databaseusername** | The database username as defined in the Username text box in the Database Connection tab.
**databasepassword** | The database password as defined in the Password text box in the Database Connection tab.
**sqlstatement** | The SQL command.
**autoconnect** | A flag ("true" or "false") to specify if the above tags should be used to automatically connect to the database without running the database connection routine in the Database Connection tab.

A complete sample XML file with three queries is displayed below:

```xml
<?xml version="1.0"?>
<doc>
<query>
  <shortdescription>Import all photograph points</shortdescription>
  <description>This command will import all of the data in the SQLite database to QGIS</description>
  <databasetype>SQLITE</databasetype>
  <databaseport /></databaseport>
  <databasename>C:\textbackslash Workshop\textbackslash eVis\Data\textbackslash PhotoPoints.db</databasename>
  <databaseusername /></databaseusername>
  <databasepassword /></databasepassword>
  <sqlstatement>SELECT Attributes.*, Points.x, Points.y FROM Attributes LEFT JOIN Points ON Points.rec_id=Attributes.point_ID</sqlstatement>
  <autoconnect>false</autoconnect>
</query>
<query>
  <shortdescription>Import photograph points "looking across Valley"</shortdescription>
  <description>This command will import only points that have photographs "looking across a valley" to QGIS</description>
  <databasetype>SQLITE</databasetype>
  <databaseport /></databaseport>
  <databasename>C:\Workshop\eVis_Data\PhotoPoints.db</databasename>
  <databaseusername /></databaseusername>
  <databasepassword /></databasepassword>
  <sqlstatement>SELECT Attributes.*, Points.x, Points.y FROM Attributes LEFT JOIN Points ON Points.rec_id=Attributes.point_ID where COMMENTS='Looking across valley'</sqlstatement>
  <autoconnect>false</autoconnect>
</query>
<query>
  <shortdescription>Import photograph points that mention "limestone"</shortdescription>
  <description>This command will import only points that have photographs that mention "limestone" to QGIS</description>
  <databasetype>SQLITE</databasetype>
  <databaseport /></databaseport>
  <databasename>C:\Workshop\eVis_Data\PhotoPoints.db</databasename>
  <databaseusername /></databaseusername>
  <databasepassword /></databasepassword>
  <sqlstatement>SELECT Attributes.*, Points.x, Points.y FROM Attributes LEFT JOIN Points ON Points.rec_id=Attributes.point_ID where COMMENTS contains 'limestone'</sqlstatement>
  <autoconnect>false</autoconnect>
</query>
</doc>
```

(continues on next page)
24.2.4 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**: to allow only some geometry types like point, multipoint, line, multiline, polygon and multipolygon.

- **Geometry validity**: depending on geometry types, the user can choose Self intersections, Duplicate nodes, Self contacts and Polygon with less than 3 nodes.

- **Geometry properties**: depending on geometry types, the user can choose Polygons and multipolygons may not contain any holes, Multipart objects must consist of more than one part and Lines must not have dangles.

- **Geometry conditions**: user can add some condition to validate the geometries with a minimal segment length, a minimum angle between segment, a minimal polygon area and sliver polygons detection.

- **Topology checks**: depending on geometry types, the user can choose Checks for duplicates, Checks for features within other features, Checks for overlaps smaller than, Checks for gaps smaller than, 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 the user how get the result between modify the current layer and create a new layer.

When you are happy with the configuration, you can click on the Run button.

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;

```xml
<database type="SQLITE"/>
<databasehost/>
<databaseport/>
<dbname>C:\Workshop\eVis_Data\PhotoPoints.db</dbname>
<databaseusername/>
<databasepassword/>
<sqlstatement>SELECT Attributes.*, Points.x, Points.y FROM Attributes LEFT JOIN Points ON Points.rec_id=Attributes.point_ID where COMMENTS like '%limestone%'</sqlstatement>
<autoconnect>false</autoconnect>
</query>
</doc>
```
Obr. 24.17: The Geometry Checker Plugin
• 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.

Obr. 24.18: 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 contour of 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.

Tip: Fix multiple errors
You can fix multiple errors by selecting more than one row in the table with the CTRL + click action.

The default action can be changed with the last icon Error resolution settings. For some type of errors, you can change the default action between some specific action or No action.

Finally, you can choose which attribute to use when merging features by attribute value.

24.2.5 Georeferencer Plugin

The Georeferencer Plugin is a tool for generating world files for rasters. It allows you to reference rasters 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 raster is to locate points on the raster for which you can accurately determine coordinates.

Features

<table>
<thead>
<tr>
<th>Ikona</th>
<th>Účel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open raster</td>
<td>Start georeferencing</td>
</tr>
<tr>
<td>Generate GDAL Script</td>
<td>Load GCP Points</td>
</tr>
<tr>
<td>Save GCP Points As</td>
<td>Transformation settings</td>
</tr>
<tr>
<td>Add Point</td>
<td>Delete Point</td>
</tr>
<tr>
<td>Move GCP Point</td>
<td>Pan</td>
</tr>
<tr>
<td>Zoom In</td>
<td>Zoom Out</td>
</tr>
<tr>
<td>Zoom To Layer</td>
<td>Zoom Last</td>
</tr>
<tr>
<td>Zoom Next</td>
<td>Link Georeferencer to QGIS</td>
</tr>
<tr>
<td>Link QGIS to Georeferencer</td>
<td>Full histogram stretch</td>
</tr>
<tr>
<td>Local histogram stretch</td>
<td></td>
</tr>
</tbody>
</table>

Table Georeferencer: Georeferencer Tools
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 plugin will compute the world file parameters. The more coordinates you provide, the better the result will be.

The first step is to start QGIS, load the Georeferencer Plugin (see The Plugins Dialog) and click on Raster Georeferencer, which appears in the QGIS menu bar. The Georeferencer Plugin dialog appears as shown in figure_georeferencer_dialog.

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.

Obr. 24.19: Georeferencer Plugin 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 button, add points to the main working area and enter their coordinates (see Figure). For this procedure you have three options:
   - Click on a point in the raster image and enter the X and Y coordinates manually.
   - Click on a point in the raster image and choose the From canvas button to add the X and Y coordinates with the help of a georeferenced map already loaded in the QGIS map canvas.
   - With the button, you can move the GCPs in both windows, if they are at the wrong place.

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 on the plugin dialog to zoom and pan the working area in order to locate a relevant set of GCP points.

Obr. 24.20: 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 plugin 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.

Available Transformation algorithms

Depending on how many ground control points you have captured, you may want to use different transformation algorithms. Choice of transformation algorithm is also dependent on the type and quality of input data and the amount of geometric distortion that you are willing to introduce to the final result.

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. This algorithm likely won't be sufficient if you are dealing with scanned material.
- The Helmert transformation performs simple scaling and rotation transformations.
- The Polynomial algorithms 1-3 are among the most widely used algorithms introduced to match source and destination ground control points. The most widely used polynomial algorithm is the second-order polyno-
mial transformation, which allows some curvature. First-order polynomial transformation (affine) preserves collinearity and allows scaling, translation and rotation only.

- The Thin Plate Spline (TPS) algorithm is a more modern georeferencing method, which is able to introduce local deformations in the data. This algorithm is useful when very low quality originals are being georeferenced.
- The Projective transformation is a linear rotation and translation of coordinates.

Define the Resampling method

The type of resampling you choose will likely depending on your input data and the ultimate objective of the exercise. If you don’t want to change statistics of the image, you might want to choose ‘Nearest neighbour’, whereas a ‘Cubic resampling’ will likely provide a more smoothed result.

It is possible to choose between five different resampling methods:

1. Nearest neighbour
2. Linear
3. Cubic
4. Cubic Spline
5. Lanczos
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 SRS (Spatial Reference System) for the georeferenced raster (see Práce s projekcemi).

- 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.

- Finally, Load in QGIS 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 define whether you want to show GCP coordinates and/or IDs.

- As residual units, pixels and map units can be chosen.

- For the PDF report, a left and right margin can be defined and you can also set the paper size for the PDF map.

- Finally, you can activate to Show Georeferencer window docked.

Running the transformation

After all GCPs have been collected and all transformation settings are defined, just press the button to create the new georeferenced raster.
24.2.6 MetaSearch Catalog Client

Úvod

MetaSearch is a QGIS plugin to interact with metadata catalog services, supporting the OGC Catalog Service for the Web (CSW) standard.

MetaSearch provides an easy and intuitive approach and user-friendly interface to searching metadata catalogs within QGIS.

![Obr. 24.22: Search and results of Services in MetaSearch](image)

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.

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 \( \text{MetaSearch} \) 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

![Managing Catalog Services](image)

Obr. 24.23: 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 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 only the base URL is required (not a full GetCapabilities URL).
3. If the CSW 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. Click **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 for 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
<?xml version="1.0" encoding="UTF-8"?>
<qgsCSWConnections version="1.0">
  <csw name="Data.gov CSW" url="https://catalog.data.gov/csw-all"/>
  <csw name="Geonorge - National CSW service for Norway" url="https://www.geonorge.no/geonetwork/srv/eng/csw"/>
  <csw name="Geoportale Nazionale - Servizio di ricerca Italiano" url="http://www.pcn.minambiente.it/geoportal/csw"/>
  <csw name="LINZ Data Service" url="http://data.linz.govt.nz/feeds/csw"/>
  <csw name="Nationaal Georegister (Nederland)" url="http://www.nationaalgeoregister.nl/geonetwork/srv/eng/csw"/>
  <csw name="RNDT - Repertorio Nazionale dei Dati Territoriali - Servizio di ricerca" url="http://www.rndt.gov.it/RNDT/CSW"/>
  <csw name="UK Location Catalogue Publishing Service" url="http://csw.data.gov.uk/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 XML response, click the **GetCapabilities Response** button. A separate window will open displaying Capabilities XML.
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 on defined by $X_{max}$, $X_{min}$, $Y_{max}$, and $Y_{min}$. Click *Set Global* to do a global search, click *Map Extent* to do a search on the visible area only or manually enter custom values as desired.

Clicking the *Search* button will search the selected Metadata Catalog. Search results are displayed in a list and are sortable 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 Search Results as XML* button to open a window with the service response in raw XML format.
• if the metadata record has an associated bounding box, a footprint of the bounding box will be displayed on
the map;
• double-clicking the record displays the record metadata with any associated access links. Clicking the links
opens the link in the user’s web browser;
• if the record is a supported web service (WMS/WMTS, WFS, WCS, ArcGIS MapServer, ArcGIS Feature-
Server, etc.), the Add Data button will be enabled for the user to add to QGIS. When clicking this button,
MetaSearch will verify if this is a valid OWS. The service will then be added to the appropriate QGIS con-
nection list, and the appropriate connection dialog will then appear.

Obr. 24.25: Metadata record display

Nastavení

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.
• Results paging: when searching metadata catalogs, the number of results to show per page. Default value is 10.
Obr. 24.26: MetaSearch settings
24.2.7 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 a datasource (usually PostGIS or WFS-T) 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 a PostGIS or WFS-T datasource).
2. Assuming you have already enabled the plugin (see Core and External plugins) 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.
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.
7. This is all!
8. Save your project and bring it on the field.
9. Edit the layers offline.
10. After being connected again, upload the changes using Database Offline Editing Synchronize.

24.2.8 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 overshots 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.

It is very simple to create topology rules with the Topology Checker plugin.

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.
Obr. 24.27: Create an offline project

Obr. 24.28: The Topology Checker Plugin
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.

Below is the list of Core plugins provided with QGIS. They are not necessarily enabled by default.
The QGIS Python Console is an interactive shell for the 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).

### 24.3.1 The Interactive Console

The interactive console is composed of a toolbar, an input area and an output one.

**Panel nástrojů**

The toolbar proposes the following tools:

- Clear Console: to wipe the output area;
- Run Command: available in the input area: same as pressing `Enter`;
- Show Editor: toggles *The Code Editor* visibility;
- Options…: opens a dialog to configure console properties (see *Možnosti*);
- Help…: browses the current documentation.

---

### 24.3 QGIS Python console

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 the 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).
Console

The console 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 Možnosti;
- 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 file ~/.qgis2/console_history.txt;
- Open QGIS C++ API documentation by typing _api;
- Open QGIS Python API documentation by typing _pyqgis.
- Open PyQGIS Cookbook by typing _cookbook.

Tip: 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 (>>> ...).
24.3.2 The Code Editor

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 codepad.org 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 codepad.org.
- Ctrl+4 Syntax check.
- 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).

Poznámka: Running partially or totally a script from the Code Editor outputs the result in the Console output area.

24.3.3 Možnosti

Accessible from the Console toolbar and the contextual menus of the Console output panel and the Code Editor, the Python Console Settings help manage and control the Python console behavior.

For both Console and Editor you can specify:

- 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)
- Typing
  - Automatic parentheses insertion: Enables autoclosing for parentheses
  - Automatic insertion of the `import` string on `from xxx`: Enables insertion of `import` when specifying imports

For Editor you can also specify:

- Run and Debug
Obr. 24.30: The Python Console editor

- Enable Object Inspector (switching between tabs may be slow): Enable the object inspector.
- 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.

* Font and Colors: Here you can specify the font to use in the editor and the colors to use for highlighting

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.

### Tip: 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.
Pomoc a podpora

25.1 Mailing list

QGIS je v aktivním vývoji a jako takový nebude vždy fungovat tak, jak jste očekávali. Preferovaný způsob, jak získat pomoc, je připojení se k mailing listu uživatelů qgis. Vaše dotazy se dostanou k širšímu publiku a odpovědi budou prospěšné ostatním.

25.1.1 Uživatelé QGIS

This mailing list is used for discussion about QGIS in general, as well as specific questions regarding its installation and use. You can subscribe to the qgis-users mailing list by visiting the following URL: https://lists.osgeo.org/mailman/listinfo/qgis-user

25.1.2 Vývojáři 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

25.1.3 Komunitní tým 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
25.1.4 Překlady 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

25.1.5 Řídící výbor projektu 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

25.1.6 Uživatelské skupiny 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

Jste vítáni k zapsání se k jakémukoliv z těchto seznamů. Prosíme nezapomeňte přispět do seznamů zodpovězením otázek a sdílením Vašich zkušeností.

25.2 IRC

We also maintain a presence on IRC - visit us by joining the #qgis channel on irc.freenode.net. Please wait for a response to your question, as many folks on the channel are doing other things and it may take a while for them to notice your question. If you missed a discussion on IRC, not a problem! We log all discussion, so you can easily catch up. Just go to https://qgis.org/irclogs and read the IRC-logs.

25.3 Commercial support

Commercial support for QGIS is also available. Check the website https://qgis.org/en/site/forusers/commercial_support.html for more information.

25.4 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.

Mějte na paměti, že vaše chyba nemusí mít vždy přednost, ve kterou můžete doufáv (v závislosti na její závažnosti). Některé chyby mohou vyžadovat větší úsilí vývojářů k nápravě a pracovní síla nemusí být vždy k dispozici.

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.

Read Bugs, Features and Issues and submit_patch for more details.
25.5 Blog

The QGIS community also runs a weblog at https://planet.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!

25.6 Zásuvné moduly

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'.

25.7 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!
Contributors

QGIS is an open source project developed by a team of dedicated volunteers and organisations. We strive to be a welcoming community for people of all race, creed, gender and walks of life. At any moment, you can get involved.

### 26.1 Authors

Below are listed people who dedicate their time and energy to write, review, and update the whole QGIS documentation.

<table>
<thead>
<tr>
<th>Name</th>
<th>Name</th>
<th>Name</th>
<th>Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim Sutton</td>
<td>Yves Jacolin</td>
<td>Jacob Lanstorp</td>
<td>Gary E. Sherman</td>
<td>Richard Duivenvoorde</td>
</tr>
<tr>
<td>Tara Athan</td>
<td>Anita Graser</td>
<td>Arnaud Morvan</td>
<td>Gavin Macaulay</td>
<td>Luca Casagrande</td>
</tr>
<tr>
<td>K. Koy</td>
<td>Hugo Mercier</td>
<td>Akgar Gumbira</td>
<td>Marie Silvestre</td>
<td>Jürgen E. Fischer</td>
</tr>
<tr>
<td>Fran Raga</td>
<td>Eric Goddard</td>
<td>Martin Dobias</td>
<td>Diethard Jansen</td>
<td>Saber Razmjooei</td>
</tr>
<tr>
<td>Ko Nagase</td>
<td>Nyall Dawson</td>
<td>Matthias Kuhn</td>
<td>Andreas Neumann</td>
<td>Harrissou Sant-anna</td>
</tr>
<tr>
<td>Mel Clos</td>
<td>David Willis</td>
<td>Larissa Junek</td>
<td>Paul Blottière</td>
<td>Sebastian Dietrich</td>
</tr>
<tr>
<td>Chris Mayo</td>
<td>Stephan Holl</td>
<td>Magnus Homann</td>
<td>Bernhard Ströbl</td>
<td>Alessandro Pasotti</td>
</tr>
<tr>
<td>N. Horning</td>
<td>Radim Blazek</td>
<td>Joshua Arnott</td>
<td>Luca Manganelli</td>
<td>Marco Hugentobler</td>
</tr>
<tr>
<td>Andre Mano</td>
<td>Mie Winstrup</td>
<td>Frank Sokolic</td>
<td>Vincent Picavet</td>
<td>Jean-Roc Morreale</td>
</tr>
<tr>
<td>Andy Allan</td>
<td>Victor Olaya</td>
<td>Tyler Mitchell</td>
<td>René-Luc D'Hont</td>
<td>Marco Bernasocchi</td>
</tr>
<tr>
<td>Ilkka Rinne</td>
<td>Werner Macho</td>
<td>Chris Berkhout</td>
<td>Nicholas Duggan</td>
<td>Jonathan Willits</td>
</tr>
<tr>
<td>David Adler</td>
<td>Lars Luthman</td>
<td>Brendan Morely</td>
<td>Raymond Nijsen</td>
<td>Carson J.Q. Farmer</td>
</tr>
<tr>
<td>Jaka Kranjc</td>
<td>Mezene Worku</td>
<td>Patrick Sunter</td>
<td>Steven Cordwell</td>
<td>Stefan Blumentrath</td>
</tr>
<tr>
<td>Andy Schmid</td>
<td>Vincent Mora</td>
<td>Alexandre Neto</td>
<td>Hien Tran-Quang</td>
<td>Alexandre Busquets</td>
</tr>
<tr>
<td>João Gaspar</td>
<td>Tom Kralidis</td>
<td>Alexander Bruy</td>
<td>Paolo Cavallini</td>
<td>Milo Van der Linden</td>
</tr>
<tr>
<td>Peter Erstz</td>
<td>Ujaval Gandhi</td>
<td>Dominic Keller</td>
<td>Giovanni Manghi</td>
<td>Maximilian Krambach</td>
</tr>
<tr>
<td>Anne Ghisla</td>
<td>Dick Groskamp</td>
<td>Uros Prelozni k</td>
<td>Stéphane Brunner</td>
<td>QGIS Koran Translator</td>
</tr>
<tr>
<td>Zoltan Siki</td>
<td>Håvard Tveite</td>
<td>Matteo Ghetta</td>
<td>Salvatore Larosa</td>
<td>Konstantinos Nikolaou</td>
</tr>
<tr>
<td>Tom Chadwin</td>
<td>Larry Shaffer</td>
<td>Nathan Woodrow</td>
<td>Martina Savarese</td>
<td>Godofredo Contreras</td>
</tr>
<tr>
<td>Astrid Emde</td>
<td>Luigi Pirelli</td>
<td>Thomas Gratier</td>
<td>Giovanni Allegri</td>
<td>GiordanoPezzola</td>
</tr>
<tr>
<td>Paolo Corti</td>
<td>Tudor Bărăscu</td>
<td>Maning Sambale</td>
<td>Claudia A. Engel</td>
<td>Yoichi Kayama</td>
</tr>
<tr>
<td>Otto Dassau</td>
<td>Denis Kouzaud</td>
<td>Nick Bearman</td>
<td>embelding</td>
<td>ajazepk</td>
</tr>
<tr>
<td>Ramon</td>
<td>Andrei</td>
<td>zstadler</td>
<td>icephale</td>
<td></td>
</tr>
</tbody>
</table>
26.2 Translators

QGIS is a multi-language application and as is, also publishes a documentation translated into several languages. Many other languages are being translated and would be released as soon as they reach a reasonable percentage of translation. If you wish to help improving a language or request a new one, please see https://qgis.org/en/site/getinvolved/index.html.

The current translations are made possible thanks to:

<table>
<thead>
<tr>
<th>Language</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bahasa Indonesia</td>
<td>Emir Hartato, I Made Anombawa, Januar V. Simarmata, Muhammad Iqnaul Haq Siregar, Trias Aditya</td>
</tr>
<tr>
<td>Chinese (Traditional)</td>
<td>Calvin Ngei, Zhang Jun, Richard Xie</td>
</tr>
<tr>
<td>Dutch</td>
<td>Carlo van Rijswijk, Dick Groskamp, Diethard Jansen, Raymond Nijssen, Richard Duivenvoorde, Willem Hoffman</td>
</tr>
<tr>
<td>Finnish</td>
<td>Matti Mäntynen, Kari Mikkonen</td>
</tr>
<tr>
<td>French</td>
<td>Arnaud Morvan, Augustin Roche, Didier Vanden Berghe, Dofabien, Etienne Trimailhe, Harris-sou Sant-anna, Jean-Roc Morreale, Jérémy Garnaux, Loïc Buscoz, Lsam, Marc-André Saia, Marie Sivestre, 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</td>
</tr>
<tr>
<td>Galician</td>
<td>Xan Vieiro</td>
</tr>
<tr>
<td>German</td>
<td>Jürgen E. Fischer, Otto Dassau, Stephan Holl, Werner Macho</td>
</tr>
<tr>
<td>Italian</td>
<td>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</td>
</tr>
<tr>
<td>Japanese</td>
<td>Baba Yoshihiko, Minoru Akagi, Northiro Yamate, Takayuki Mizutani, Takayuki Numura, Yoichi Kayama</td>
</tr>
<tr>
<td>Korean</td>
<td>OSGeo Korean Chapter</td>
</tr>
<tr>
<td>Polish</td>
<td>Andrzej Świąder, Borys Jurgiel, Ewelina Krawczak, Jakub Bobrowski, Mateusz Łoskot, Michał Kulach, Michał Smoczyk, Milena Nowotarska, Radosław Pasiok, Robert Szczepanek, Tomasz Paul</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Alexandre Neto, Duarte Carreira, Giovanni Manghi, João Gaspar, Joana Simões, Leandro Infantini, Nelson Silva, Pedro Palheiro, Pedro Pereira, Ricardo Sera</td>
</tr>
<tr>
<td>Portuguese (Brasil)</td>
<td>Arthur Nanni, Felipe Sodré Barros, Leónidas Descovi Filho, Marcelo Soares Souza, Narcélio de Sá Pereira Filho, Sidney Schaberle Goveia</td>
</tr>
<tr>
<td>Romanian</td>
<td>Alex Bădescu, Bogdan Pacurar, Georgiana Ioanovici, Lonut Losifescu-Enescu, Sorin Călinică, Tudor Bărăscu</td>
</tr>
<tr>
<td>Russian</td>
<td>Alexander Bruy, Artem Popov</td>
</tr>
<tr>
<td>Spanish</td>
<td>Carlos Dávila, Diana Galindo, Edwin Amado, Gabriela Awad, Javier César Aldariz, Mayeul Kauffmann, Fran Raga</td>
</tr>
<tr>
<td>Ukrainian</td>
<td>Alexander Bruy</td>
</tr>
</tbody>
</table>
27.1 Appendix A: GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Každému je povoleno kopírovat a šířit doslovné kopie tohoto licenčního dokumentu, ale není povoleno je měnit.

Úvod

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this,
we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The „Program”, below, refers to any such program or work, and a „work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term „modification”.) Each licensee is addressed as „you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of
the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address
new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and „any later version“, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM „AS IS“ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THE PROGRAM OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

QGIS Qt exception for GPL

In addition, as a special exception, the QGIS Development Team gives permission to link the code of this program with the Qt library, including but not limited to the following versions (both free and commercial): Qt/Non-commercial Windows, Qt/Windows, Qt/X11, Qt/Mac, and Qt/Embedded (or with modified versions of Qt that use the same license as Qt), and distribute linked combinations including the two. You must obey the GNU General Public License in all respects for all of the code used other than Qt. If you modify this file, you may extend this exception to your version of the file, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

27.2 Appendix B: GNU Free Documentation License

Verze 1.3, 3. listopad 2008
http://fsf.org/
Každému je povoleno kopírovat a šířit doslovně kopie tohoto licenčního dokumentu, ale není povoleno je měnit.

Úvod

Smyslem této licence je učinit manuál, učebnici nebo jiný funkční a užitečný dokument „svobodným“ v tomto smyslu svobody: zaručit každému skutečnou svobodu pro kopírování a šíření, s úpravami či bez úprav, ať už komerčně nebo nekomerčně. Za druhé, tato licence zajišťuje pro autora i vydavatele možnost zásluh za jejich práci, aniž by byl odpovědným za úpravy, které udělali jiní.

Tato licence je druh „copyleftu“, což znamená, že odvozená díla od daného dokumentu musí být sama taktéž svobodná. Doplňuje GNU všeobecnou veřejnou licenci, což je copyleftová licence určená pro svobodný software.
Navrhlí jsme tuto licenci, k použití pro manuály ke svobodnému softwaru, protože svobodný software vyžaduje svobodnou dokumentaci: Svobodný program by měl přijít s manuály poskytujícími stejné svobody, jako software. Avšak tato licence není omezena softwarovými manuály; může být použita pro jakékolik textové dílo, nelehde na téma nebo zda je vydáno jako tištěná kniha. Doporučujeme tuto licenci zejména pro díla, jejichž účelem je výuka nebo poskytování informací.

1. POUŽITELNOST A DEFINICE

Tato licence platí pro jakýkoli manuál nebo jiné dílo, v jakémkoliv médiu, které obsahuje oznámení, umístěné držitelem autorských práv, které říká, že dílo může být šířeno za podmínek této licence. Toto oznámení uděluje celkově dostupnou, bezplatnou licenci, neomezenou v trvání, pro použití daného díla za podmínek stanovených zde. Tento dokument, níže, odkazuje na další takový manuál nebo dílo. Každý člen veřejnosti je držitelem licence, a je označován jako „vy“. Licencí přijímáte, pokud kopírujete, upravujete nebo distribuujete dílo takovým způsobem, který vyžaduje povolení podléhající autorstvímu zákonu.

„Upravená verze“ dokumentu znamená jakékoliv dílo obsahující dokument nebo jeho část, ať už zkopiírovaný dostověrně, nebo s úpravami a/nebo přeložený do jiného jazyka.

„Druhotný oddíl“ je pojmenovaný jako dodatek nebo vstupní díl dokumentační části, který se zabývá výhradně vztahem vydavatelů nebo autorů dokumentu k celkovému tématu dokumentu (nebo souvisejících záležitostí) a neobsahuje nic, co by mohlo přímo spadat do rámce tohoto celkového tému. (Je-li tedy dokument částí učebnice matematiky, druhotný oddíl nesmí vysvětlovat cokoli z matematiky.) Tento vztah vytváří absolutně omezující historického spojení s tématem či souvisejícími záležitostmi nebo s právním, obchodním, filozofickým, etickým či politickým postojem k nim.

„Neměnné oddíly“ jsou z jisté části druhotnými oddíly, jejichž názvy jsou označeny tak, jako názvy neměnných oddílů s poznámkou, kde stojí, že se dokument zveřejňuje pod touto licencí. Pokud se oddíl nepřizpůsobí výše uvedené definici druhotného oddílu, pak není povoleno, aby byl označen za neměnný. Dokument může obsahovat prázdné neměnné oddíly. V případě, že dokument neurčí žádné neměnné oddíly, pak v něm žádné nejsou.

The „Cover Texts“ are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A „Transparent“ copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not Transparent is called Opaque.

Examples of suitable formats for Transparent copies include plain ASCII without markup, TeXinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

**Titulní stránka** znamená utisknou knihou samotnou titulní stránku a také následující stránky, které jsou čitelně potřebné k uchovávání materiálu, který tato licence vyžaduje, aby se objevil na titulní stránce. U děl ve formátech, které nemají žádnou titulní stránku jako takovou, znamená „titulní stránka“ text v blízkosti nejvýznamnějšího vzhledu názvu díla, který předchází začátek textu.

The „publisher“ means any person or entity that distributes copies of the Document to the public.

A section „Entitled XYZ“ means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as „Acknowledgements“, „Dedications“, „Endorsements“, or „History“.) To „Preserve the Title“ of such a section when you modify the Document means that it remains a section „Entitled XYZ“ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards...
disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

Můžete kopírovat a distribuovat dokument na jakémkoliv médiu, ať už komerčně nebo nekomerčně, za předpokladu, že tato licence, upozornění na autorská práva a upozornění na licenci uvádějící, že tato licence se vztahuje na dokument, jsou reprodukovány ve všech kopích a že nepridáte žádné další podmínky k těm z této licence. Nesmíte používat technická opatření, která by bránila nebo kontrolovala čtení nebo další kopírování kopii, které vytváříte nebo distribuujete. Výměnou za kopie však můžete přijmout náhradu. Pokud distribuujete dostatečně velký počet kopii, musíte rovněž dodržet podmínky v části 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.
I. Preserve the section Entitled „History“, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled „History“ in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the „History“ section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. U jakékoli částí s názvem „Poděkování“ nebo „Věnování“ zachovejte název sekce a v této části uchovejte veškerou podstatu a tón každého z uznaní a / nebo věnování uvedených přispěvatełů.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled „Endorsements“. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled „Endorsements“ or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled „Endorsements“, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled „History“ in the various original documents, forming one section Entitled „History“; likewise combine any sections Entitled „Acknowledgements“, and any sections Entitled „Dedications“. You must delete all sections Entitled „Endorsements“.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an „aggregate“ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

If a section in the Document is Entitled „Acknowledgements“, „Dedications“, or „History“, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License „or any later version“ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING
“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A „Massive Multiauthor Collaboration“ (or „MMC“) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

**ADDENDUM: How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
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".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the „with … Texts.“ line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

Pokud váš dokument obsahuje netrivální příklady programového kódu, doporučujeme tyto příklady uvolnit paralelně pod vaší volnou licencí svobodného softwaru, jako je GNU General Public License, aby bylo povoleno jejich použití ve svobodném softwaru.

**27.3 Appendix C: QGIS File Formats**

**27.3.1 QGS/QGZ - The QGIS Project File Format**

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
• legend
• 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.

```xml
<qgis version="3.4.13-Madeira" projectname=""/>
<title/>
<homePath path=""/>
<evaluateDefaultValues active="0"/>
<trust active="0"/>
+xml:projectCrs><projectCrs>
+<layer-tree-group><layer-tree-group>
+<snapping-settings tolerance="12" unit="1" enabled="0" type="1" mode="2" intersection-snapping="0"/>
</snapping-settings>
<relations/>
+<mapcanvas name="theMapCanvas" annotationsVisible="1">
+<units>meters</units>
+<extent/>
+<rotation>0</rotation>
+<destinationrsr><destinationrsr>
+<rendermaptile>0</rendermaptile>
</mapcanvas>
+<projectModels/>
+<legend updateDrawingOrder="true"></legend>
+<mapViewDocks/>
+<mapViewDocks3D/>
+<projectlayers><projectlayers>
+<layerorder><layerorder>
+<properties><properties>
+<visibility-presets/>
+<transformContext/>
+<projectMetadata><projectMetadata>
+<Annotations/>
+<Layouts/>
+<qgis>
```

Obr. 27.1: The top level tags in a QGS file

### 27.3.2 QLR - The QGIS Layer Definition file

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.
Obr. 27.2: The expanded top level ProjectLayers tag of a QGIS file
Obr. 27.3: The top level tags of a QLR file
27.3.3 QML - The QGIS Style File Format

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).

```xml
<qgis version="3.14.13-Madeira" styleCategories="AllStyleCategories" readOnly="0" maxScale="0" labelsEnabled="0" simplifyDrawingHints="1" hasScaleBasedVisibilityFlag="0" simplifyDrawingTol="1" simplifyMaxScale="1" minScale="1e+0" simplifyAlgorithm="0" simplifyLocal="1">
  <flags>
    <flag/>
  </flags>
  <renderer_v2 symbolLevels="0" enableOrderBy="0" type="singleSymbol" forceRaster="0">
    <symbol clip_to_extent="1" name="0" alpha="1" type="fill" force_rhr="0"/>
    <symbol/>
    <rotation/>
    <sizescaler/>
  </renderer_v2>
  <customProperties>
    <blendMode>0</blendMode>
    <featureBlendMode>0</featureBlendMode>
    <layerOpacity>1</layerOpacity>
  </customProperties>
  <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"/>
  <childConfiguration/>
  <excludeAttributesWMS/>
  <excludeAttributesWFS/>
  <defaults/>
  <constraints>
    <constraintExpressions/>
    <expressionFields/>
    <attributeactions>
      <attributeableconfig actionWidgetStyle="dropDown" sortExpression="0" sortOrder="0"/>
    </attributeableconfig>
    <conditionalstyles/>
    <editform tolerant="1"/>
    <editforminitcode/>
    <editforminitcodesource/>
    <editforminitfilepath/>
    <editforminitcode/>
    <featform suppress="0" featformsuppress="0" editorlayout="generatedlayout"/>
    <editable/>
    <labelOnTop/>
    <widgets/>
    <previewExpression>ID</previewExpression>
    <mapTip/>
    <layerGeometryType>2</layerGeometryType>
  </qgis>
```

Obr. 27.4: The top level tags of a QML file (only the renderer_v2 tag with its symbol tag is expanded)
27.4 Appendix D: QGIS R script syntax

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 algorithm 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:

```r
##My Group=group
```

27.4.1 Inputs

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`

27.4.2 Outputs

As for the inputs, each output has to be defined at the beginning of the script:

- **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.**

27.4.3 Syntax Summary for QGIS R scripts

A number of input and output parameter types are offered.
### Input parameter types

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax example</th>
<th>Returning objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector</td>
<td>Layer = vector</td>
<td>sf object (or SpatialDataFrame object, if <code>##load_vector_using_rgdal</code> is specified)</td>
</tr>
<tr>
<td>vector point</td>
<td>Layer = vector point</td>
<td>sf object (or SpatialDataFrame object, if <code>##load_vector_using_rgdal</code> is specified)</td>
</tr>
<tr>
<td>vector line</td>
<td>Layer = vector line</td>
<td>sf object (or SpatialDataFrame object, if <code>##load_vector_using_rgdal</code> is specified)</td>
</tr>
<tr>
<td>vector polygon</td>
<td>Layer = vector polygon</td>
<td>sf object (or SpatialPolygonsDataFrame object, if <code>##load_vector_using_rgdal</code> is used)</td>
</tr>
<tr>
<td>multiple vector</td>
<td>Layer = multiple vector</td>
<td>sf object (or SpatialDataFrame objects if <code>##load_vector_using_rgdal</code> is specified)</td>
</tr>
<tr>
<td>table</td>
<td>Layer = table</td>
<td>dataframe conversion from csv, default object of <code>read.csv</code> function</td>
</tr>
<tr>
<td>field</td>
<td>Field = Field Layer</td>
<td>name of the Field selected, e.g. &quot;Area&quot;</td>
</tr>
<tr>
<td>raster</td>
<td>Layer = raster</td>
<td>RasterBrick object, default object of <code>raster</code> package</td>
</tr>
<tr>
<td>multiple raster</td>
<td>Layer = multiple raster</td>
<td>RasterBrick objects, default object of <code>raster</code> package</td>
</tr>
<tr>
<td>number</td>
<td>N = number</td>
<td>integer or floating number chosen</td>
</tr>
<tr>
<td>string</td>
<td>S = string</td>
<td>string added in the box</td>
</tr>
<tr>
<td>longstring</td>
<td>LS = longstring</td>
<td>string added in the box, could be longer than the normal string</td>
</tr>
<tr>
<td>selection</td>
<td>S = selection</td>
<td>string of the selected item chosen in the dropdown menu</td>
</tr>
<tr>
<td>crs</td>
<td>C = crs</td>
<td>string of the resulting CRS chosen, in the format: &quot;EPSG:4326&quot;</td>
</tr>
<tr>
<td>extent</td>
<td>E = extent</td>
<td>Extent object of the <code>raster</code> package, you can extract values as <code>E@xmin</code></td>
</tr>
<tr>
<td>point</td>
<td>P = point</td>
<td>when clicked on the map, you have the coordinates of the point</td>
</tr>
<tr>
<td>file</td>
<td>F = file</td>
<td>path of the file chosen, e.g. &quot;/home/matteo/file.txt&quot;</td>
</tr>
<tr>
<td>folder</td>
<td>F = folder</td>
<td>path of the folder chosen, e.g. &quot;/home/matteo/Downloads&quot;</td>
</tr>
</tbody>
</table>

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:

```r
##Layer = vector
##Field1 = Field Layer
##Field2 = optional Field Layer
```

### Output parameter types

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax example</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector</td>
<td>Output = output vector</td>
</tr>
<tr>
<td>raster</td>
<td>Output = output raster</td>
</tr>
<tr>
<td>table</td>
<td>Output = output table</td>
</tr>
<tr>
<td>file</td>
<td>Output = output file</td>
</tr>
</tbody>
</table>

**Poznámka:** You can save plots as `png` from the *Processing Result Viewer*, or you can choose to save the plot directly from the algorithm interface.
Script body

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)
```

### 27.4.4 Examples

#### Example with vector output

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. Point pattern analysis is the group of the algorithm
2. Layer is the input vector layer
3. Size is a numerical parameter with a default value of 10
4. Output is the vector layer that will be created by the algorithm
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.

#### Example with raster output

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)
table = as.data.frame(Layer)
coordinates(table) = ~coords.x1+coords.x2
c = Layer[[Field]]
kriaging_result = autoKrige(c~1, table)
prediction = raster(kriaging_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.

Example with table output

Let's edit the `Summary Statistics` algorithm so that the output is a table file (csv).

The script body is the following:

```r
##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`:

```r
##Basic statistics=group
##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

**Example with plot**

To create plots, you have to use the `##output_plots_to_html` parameter as in the following script:

```python
##Basic statistics=group
##Layer=vector
##Field=Field Layer
##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*.
Literature and Web References


