QGIS Training Manual

QGIS Project

03 abr., 2025

Conteúdo

1 Introdução do curso **3**
- 1.1 Prefácio . . . 3
  - 1.1.1 Why QGIS? . . . 3
  - 1.1.2 Enquadramento . . . 4
  - 1.1.3 Licença . . . 4
  - 1.1.4 Capítulos Patrocinados . . . 5
  - 1.1.5 Autores . . . 5
  - 1.1.6 Contribuidores Individuais . . . 5
  - 1.1.7 Patrocinadores . . . 5
  - 1.1.8 Source files and Issue reports . . . 5
  - 1.1.9 Última Versão . . . 5
- 1.2 About the exercises . . . 6
  - 1.2.1 How to use this tutorial . . . 6
  - 1.2.2 Tiered course objectives . . . 6
  - 1.2.3 Data . . . 7

2 Module: Creating and Exploring a Basic Map **9**
- 2.1 Lesson: An Overview of the Interface . . . 9
  - 2.1.1 ★☆☆ Try Yourself: The Basics . . . 10
  - 2.1.2 ★☆☆ Try Yourself: 1 . . . 12
  - 2.1.3 ★☆☆ Try Yourself: 2 . . . 12
  - 2.1.4 What's Next? . . . 12
- 2.2 Lesson: Adding your first layers . . . 13
  - 2.2.1 ★☆☆ Follow Along: Prepare a map . . . 13
  - 2.2.2 ★☆☆ Try Yourself: . . . 16
  - 2.2.3 ★☆☆ Follow Along: Loading vector data from a GeoPackage Database . . . 16
  - 2.2.4 ★☆☆ Follow Along: Loading vector data from a SpatiaLite Database with the Browser . . . 17
  - 2.2.5 ★★☆ Try Yourself: Load More Vector Data . . . 19
  - 2.2.6 Follow Along: Reordering the Layers . . . 19
  - 2.2.7 In Conclusion . . . 20
  - 2.2.8 What's Next? . . . 20
- 2.3 Lesson: Navigating the Map Canvas . . . 20
  - 2.3.1 ★☆☆ Follow Along: Basic Navigation Tools . . . 20
  - 2.3.2 In Conclusion . . . 24
- 2.4 Lesson: Symbology . . . 24
  - 2.4.1 ★☆☆ Follow Along: Changing Colors . . . 25
  - 2.4.2 ★☆☆ Try Yourself: . . . 26
  - 2.4.3 ★☆☆ Follow Along: Changing Symbol Structure . . . 26
  - 2.4.4 ★☆☆ Try Yourself: . . . 27
  - 2.4.5 ★★☆ Follow Along: Scale-Based Visibility . . . 29
  - 2.4.6 ★★☆ Follow Along: Adding Symbol Layers . . . 30

2.4.7 ★★☆ Try Yourself: . . . 34
2.4.8 ★★☆ Follow Along: Ordering Symbol Levels . . . 34
2.4.9 ★★☆ Try Yourself: . . . 37
2.4.10 ★★★ Try Yourself: . . . 39
2.4.11 ★★☆ Follow Along: Symbol layer types . . . 41
2.4.12 ★★☆ Try Yourself: . . . 48
2.4.13 ★★★ Follow Along: Geometry generator symbology . . . 50
2.4.14 ★★★ Follow Along: Creating a Custom SVG Fill . . . 54
2.4.15 ★★☆ Follow Along: Masking . . . 60
2.4.16 ★★☆ Follow Along: Masking using points symbols . . . 62
2.4.17 In Conclusion . . . 64
2.4.18 Further Reading . . . 64
2.4.19 What's Next? . . . 65

3 Module: Classifying Vector Data 67
3.1 Lesson: Vector Attribute Data . . . 67
3.1.1 ★☆☆ Follow Along: Viewing Layer Attributes . . . 67
3.1.2 ★☆☆ Try Yourself: Exploring Vector Data Attributes . . . 71
3.1.3 In Conclusion . . . 72
3.1.4 What's Next? . . . 72
3.2 Lesson: Labels . . . 72
3.2.1 ★☆☆ Follow Along: Using Labels . . . 72
3.2.2 ★☆☆ Follow Along: Changing Label Options . . . 74
3.2.3 ★★☆ Follow Along: Using Labels Instead of Layer Symbology . . . 78
3.2.4 ★★☆ Try Yourself: Customize the Labels . . . 81
3.2.5 ★★☆ Follow Along: Labeling Lines . . . 85
3.2.6 ★★★ Follow Along: Data Defined Settings . . . 89
3.2.7 ★★★ Try Yourself: Using Data Defined Settings . . . 91
3.2.8 ★★★ Further Possibilities With Labeling . . . 93
3.2.9 In Conclusion . . . 94
3.2.10 What's Next? . . . 94
3.3 Lesson: Classification . . . 94
3.3.1 ★☆☆ Follow Along: Classifying Nominal Data . . . 94
3.3.2 ★☆☆ Try Yourself: More Classification . . . 99
3.3.3 ★★☆ Follow Along: Ratio Classification . . . 99
3.3.4 ★★☆ Try Yourself: Refine the Classification . . . 107
3.3.5 ★★★ Follow Along: Rule-based Classification . . . 107
3.3.6 In Conclusion . . . 111
3.3.7 What's Next? . . . 111

4 Module: Laying out the Maps 113
4.1 Lesson: Using Print Layout . . . 113
4.1.1 ★☆☆ Follow Along: The Layout Manager . . . 114
4.1.2 ★☆☆ Follow Along: Basic Map Composition . . . 115
4.1.3 ★☆☆ Follow Along: Adding a Title . . . 117
4.1.4 ★☆☆ Follow Along: Adding a Legend . . . 120
4.1.5 ★★☆ Follow Along: Customizing Legend Items . . . 121
4.1.6 ★☆☆ Follow Along: Exporting Your Map . . . 123
4.1.7 In Conclusion . . . 125
4.2 Lesson: Creating a Dynamic Print Layout . . . 125
4.2.1 ★★☆ Follow Along: Creating the dynamic map canvas . . . 125
4.2.2 ★★☆ Follow Along: Creating the dynamic header . . . 126
4.2.3 ★★☆ Follow Along: Creating labels for the dynamic header . . . 127
4.2.4 ★★☆ Follow Along: Adding pictures to the dynamic header . . . 129
4.2.5 ★★☆ Follow Along: Creating the scalebar of the dynamic header . . . 130
4.2.6 What's Next? . . . 131
4.3 Tarefa 1 . . . 131
4.3.1 In Conclusion . . . 131

5 Module: Creating Vector Data **133**
5.1 Lesson: Creating a New Vector Dataset . . . 133
5.1.1 ★☆☆ Follow Along: The Layer Creation Dialog . . . 133
5.1.2 ★☆☆ Follow Along: Data Sources . . . 136
5.1.3 ★☆☆ Try Yourself: Digitizing Polygons . . . 149
5.1.4 ★★☆ Follow Along: Using Vertex Editor Table . . . 150
5.1.5 ★☆☆ Try Yourself: Digitizing Lines . . . 153
5.1.6 In Conclusion . . . 156
5.1.7 What's Next? . . . 157
5.2 Lesson: Feature Topology . . . 157
5.2.1 ★★☆ Follow Along: Snapping . . . 157
5.2.2 ★★☆ Follow Along: Correct Topological Features . . . 159
5.2.3 ★★☆ Follow Along: Tool: Simplify Feature . . . 161
5.2.4 ★★☆ Try Yourself: Tool: Add Ring . . . 163
5.2.5 ★★☆ Try Yourself: Tool: Add Part . . . 164
5.2.6 ★★☆ Follow Along: Tool: Reshape Features . . . 165
5.2.7 ★★☆ Try Yourself: Tool: Split Features . . . 168
5.2.8 ★★★ Try Yourself: Tool: Merge Features . . . 169
5.2.9 In Conclusion . . . 170
5.2.10 What's Next? . . . 170
5.3 Lesson: Forms . . . 170
5.3.1 ★☆☆ Follow Along: Using QGIS" Form Design Functionality . . . 171
5.3.2 ★☆☆ Try Yourself: Using the Form to Edit Values . . . 172
5.3.3 ★★☆ Follow Along: Setting Form Field Types . . . 172
5.3.4 ★★★ Try Yourself: . . . 174
5.3.5 ★★★ Try Yourself: Creating Test Data . . . 176
5.3.6 ★★★ Follow Along: Creating a New Form . . . 177
5.3.7 ★★★ Follow Along: Associating the Form with Your Layer . . . 179
5.3.8 In Conclusion . . . 180
5.3.9 Further Reading . . . 180
5.3.10 What's Next? . . . 180
5.4 Lesson: Actions . . . 180
5.4.1 ★☆☆ Follow Along: Add a Field for Images . . . 180
5.4.2 ★☆☆ Follow Along: Creating an Action . . . 183
5.4.3 ★★☆ Follow Along: Searching the Internet . . . 187
5.4.4 ★★★ Follow Along: Open a Webpage Directly in QGIS . . . 190
5.4.5 In Conclusion . . . 192
5.4.6 What's Next? . . . 192

6 Module: Vector Analysis **193**
6.1 Lesson: Reprojecting and Transforming Data . . . 193
6.1.1 ★☆☆ Follow Along: Projections . . . 194
6.1.2 ★☆☆ Follow Along: «On the Fly» Reprojection . . . 194
6.1.3 ★★☆ Follow Along: Saving a Dataset to Another CRS . . . 195
6.1.4 ★★★ Follow Along: Creating Your Own Projection . . . 198
6.1.5 In Conclusion . . . 200
6.1.6 Further Reading . . . 200
6.1.7 What's Next? . . . 200
6.2 Lesson: Vector Analysis . . . 200
6.2.1 ★☆☆ The GIS Process . . . 201
6.2.2 ★☆☆ The Problem . . . 201
6.2.3 ★☆☆ The Data . . . 201
6.2.4 ★☆☆ Follow Along: Start a Project and get the Data . . . 201
6.2.5 ★☆☆ Try Yourself: Convert Layers" CRS . . . 203
6.2.6 ★☆☆ Follow Along: Analyzing the Problem: Distances From Schools and Roads . . . 205
6.2.7 ★☆☆ Try Yourself: Distance from schools . . . 212
6.2.8 ★☆☆ Follow Along: Overlapping Areas . . . 215
6.2.9 ★☆☆ Follow Along: Extract the Buildings . . . 218

6.2.10 ★★☆ Try Yourself: Further Filter our Buildings . . . 219
6.2.11 ★☆☆ Follow Along: Select Buildings of the Right Size . . . 222
6.2.12 ★☆☆ Try Yourself: . . . 223
6.2.13 In Conclusion . . . 224
6.2.14 What's Next? . . . 224
6.3 Lesson: Network Analysis . . . 224
6.3.1 ★☆☆ Follow Along: The Tools and the Data . . . 224
6.3.2 ★☆☆ Calculate the shortest path (point to point) . . . 225
6.3.3 ★★☆ Try Yourself: Fastest path . . . 227
6.3.4 ★★☆ Follow Along: Advanced options . . . 228
6.3.5 ★★★ Shortest path with speed limit . . . 231
6.3.6 ★★☆ Service area (from layer) . . . 234
6.3.7 In Conclusion . . . 236
6.3.8 What's Next? . . . 236
6.4 Lesson: Spatial Statistics . . . 236
6.4.1 ★☆☆ Follow Along: Create a Test Dataset . . . 237
6.4.2 ★☆☆ Follow Along: Basic Statistics . . . 240
6.4.3 ★☆☆ Follow Along: Compute statistics on distances between points . . . 242
6.4.4 ★☆☆ Follow Along: Nearest Neighbor Analysis (within layer) . . . 244
6.4.5 ★☆☆ Follow Along: Mean Coordinates . . . 245
6.4.6 ★☆☆ Follow Along: Image Histograms . . . 246
6.4.7 ★☆☆ Follow Along: Spatial Interpolation . . . 249
6.4.8 ★★☆ Try Yourself: Different interpolation methods . . . 249
6.4.9 In Conclusion . . . 250
6.4.10 What's Next? . . . 250

7 Module: Rasters **251**
7.1 Lesson: Working with Raster Data . . . 251
7.1.1 ★☆☆ Follow Along: Loading Raster Data . . . 251
7.1.2 ★☆☆ Follow Along: Create a Virtual Raster . . . 252
7.1.3 ★★★ Transforming Raster Data . . . 254
7.1.4 In Conclusion . . . 256
7.1.5 What's Next? . . . 256
7.2 Lesson: Changing Raster Symbology . . . 257
7.2.1 ★☆☆ Try Yourself: . . . 257
7.2.2 ★☆☆ Follow Along: Changing Raster Layer Symbology . . . 258
7.2.3 ★☆☆ Follow Along: Singleband gray . . . 258
7.2.4 ★☆☆ Follow Along: Singleband pseudocolor . . . 260
7.2.5 Follow Along: Changing the transparency . . . 262
7.2.6 In Conclusion . . . 265
7.2.7 Referência . . . 265
7.2.8 What's Next? . . . 265
7.3 Lesson: Terrain Analysis . . . 265
7.3.1 ★☆☆ Follow Along: Calculating a Hillshade . . . 265
7.3.2 ★☆☆ Follow Along: Using a Hillshade as an Overlay . . . 267
7.3.3 Follow Along: Finding the best areas . . . 268
7.3.4 ★★☆ Follow Along: Calculating the Slope . . . 268
7.3.5 ★★☆ Try Yourself: Calculating the aspect . . . 269
7.3.6 ★★☆ Follow Along: Finding the north-facing aspect . . . 271
7.3.7 ★★☆ Try Yourself: More criteria . . . 273
7.3.8 ★★☆ Follow Along: Combining Raster Analysis Results . . . 276
7.3.9 ★★☆ Follow Along: Simplifying the Raster . . . 278
7.3.10 ★★☆ Follow Along: Reclassifying the Raster . . . 280
7.3.11 ★☆☆ Follow Along: Querying the raster . . . 284
7.3.12 In Conclusion . . . 287
7.3.13 What's Next? . . . 287

8 Module: Completing the Analysis **289**

8.1 Lesson: Raster to Vector Conversion . . . 289
8.1.1 ★★☆ Follow Along: The *Raster to Vector* Tool . . . 289
8.1.2 ★★☆ Try Yourself: . . . 290
8.1.3 ★★☆ Follow Along: The *Vector to Raster* Tool . . . 291
8.1.4 In Conclusion . . . 293
8.1.5 What's Next? . . . 293
8.2 Lesson: Combining the Analyses . . . 293
8.2.1 ★★☆ Try Yourself: . . . 293
8.2.2 ★★☆ Try Yourself: Inspecting the Results . . . 294
8.2.3 ★★☆ Try Yourself: Refining the Analysis . . . 294
8.2.4 In Conclusion . . . 299
8.2.5 What's Next? . . . 299
8.3 Tarefa . . . 299
8.4 Lesson: Supplementary Exercise . . . 300
8.4.1 Problem Statement . . . 300
8.4.2 Solution Outline . . . 300
8.4.3 Follow Along: Setting up the Map . . . 300
8.4.4 Loading Data into the Map . . . 301
8.4.5 Changing the layer order . . . 302
8.4.6 Find the Correct Districts . . . 302
8.4.7 Clip the Rasters . . . 303
8.4.8 Changing the symbology of vector layers . . . 305
8.4.9 Changing the symbology of raster layers . . . 305
8.4.10 Clean up the map . . . 306
8.4.11 Create the hillshade . . . 306
8.4.12 Slope . . . 306
8.4.13 Try Yourself: Aspect . . . 307
8.4.14 Reclassifying rasters . . . 307
8.4.15 Combining rasters . . . 308
8.4.16 Finding rural areas . . . 308
8.4.17 Creating a negative buffer . . . 309
8.4.18 Vectorizing the raster . . . 310
8.4.19 Fixing geometry . . . 312
8.4.20 Determining the Intersection of vectors . . . 312
8.4.21 Calculating the area for each polygon . . . 312
8.4.22 Selecting areas of a given size . . . 314
8.4.23 Digitize the University of Cape Town . . . 314
8.4.24 Find the locations that are closest to the University of Cape Town . . . 315

9 Module: Plugins **317**
9.1 Lesson: Installing and Managing Plugins . . . 317
9.1.1 ★☆☆ Follow Along: Managing Plugins . . . 317
9.1.2 ★☆☆ Follow Along: Installing New Plugins . . . 318
9.1.3 ★☆☆ Follow Along: Configuring Additional Plugin Repositories . . . 320
9.1.4 In Conclusion . . . 322
9.1.5 What's Next? . . . 322
9.2 Lesson: Useful QGIS Plugins . . . 322
9.2.1 ★☆☆ Follow Along: The QuickMapServices Plugin . . . 322
9.2.2 ★☆☆ Follow Along: The QuickOSM Plugin . . . 324
9.2.3 ★☆☆ Follow Along: The DataPlotly Plugin . . . 327
9.2.4 In Conclusion . . . 331
9.2.5 What's Next? . . . 331

10 Module: Online Resources **333**
10.1 Lesson: Web Mapping Services . . . 333
10.1.1 ★☆☆ Follow Along: Loading a WMS Layer . . . 333
10.1.2 ★☆☆ Try Yourself: . . . 343
10.1.3 ★★★ Try Yourself: . . . 344

10.1.4 ★★★ Try Yourself: . . . 345
10.1.5 In Conclusion . . . 346
10.1.6 Further Reading . . . 346
10.1.7 What's Next? . . . 346
10.2 Lesson: Web Feature Services . . . 347
10.2.1 ★☆☆ Follow Along: Loading a WFS Layer . . . 347
10.2.2 ★★☆ Follow Along: Querying a WFS Layer . . . 351
10.2.3 In Conclusion . . . 354
10.2.4 What's Next? . . . 355

11 Module: QGIS Server **357**
11.1 Lesson: Install QGIS Server . . . 357
11.1.1 ★★☆ Follow Along: Install from packages . . . 357
11.1.2 ★★☆ Follow Along: QGIS Server Executable . . . 358
11.1.3 ★★★ HTTP Server Configuration . . . 358
11.1.4 ★★☆ Follow Along: Create another virtual host . . . 358
11.1.5 In Conclusion . . . 359
11.1.6 What's Next? . . . 359
11.2 Lesson: Serving WMS . . . 359
11.2.1 Registando . . . 361
11.2.2 GetMap requests . . . 363
11.2.3 ★★☆ Try Yourself: Change the Image and Layers parameters . . . 364
11.2.4 ★★☆ Follow Along: Use Filter, Opacities and Styles parameters . . . 364
11.2.5 ★★☆ Follow Along: Use Redlining . . . 365
11.2.6 GetPrint requests . . . 367
11.2.7 In Conclusion . . . 368
11.2.8 What's Next? . . . 368

12 Module: GRASS **369**
12.1 Lesson: GRASS Setup . . . 369
12.1.1 ★☆☆ Follow Along: Start a New GRASS Session . . . 369
12.1.2 ★☆☆ Follow Along: Start a New GRASS Project . . . 371
12.1.3 ★☆☆ Follow Along: Loading Vector Data into GRASS . . . 375
12.1.4 ★☆☆ Follow Along: Loading Raster Data into GRASS . . . 379
12.1.5 ★☆☆ Try Yourself: Add Layers to Mapset . . . 381
12.1.6 ★☆☆ Open an existing GRASS Mapset . . . 381
12.1.7 In Conclusion . . . 383
12.1.8 What's Next? . . . 383
12.2 Lesson: GRASS Tools . . . 383
12.2.1 ★☆☆ Follow Along: Create an aspect map . . . 383
12.2.2 ★☆☆ Follow Along: Get basic statistic of raster layer . . . 385
12.2.3 ★★☆ Follow Along: The Reclass Tool . . . 387
12.2.4 ★★☆ Try Yourself: Reclassify with your rules . . . 389
12.2.5 ★★☆ Follow Along: The Mapcalc Tool . . . 390
12.2.6 In Conclusion . . . 392

13 Module: Assessment **393**
13.1 Create a base map . . . 393
13.1.1 Add the point layer . . . 393
13.1.2 Add the line layer . . . 394
13.1.3 Add the polygon layer . . . 395
13.1.4 Create the raster backdrop . . . 395
13.1.5 Finalize the base map . . . 395
13.2 Analyze the data . . . 395
13.2.1 ★★☆ / ★★★ . . . 395
13.3 Final Map . . . 396

14 Module: Forestry Application **397**
14.1 Lesson: Forestry Module Presentation . . . 397

14.1.1 Forestry Sample Data . . . 398
14.2 Lesson: Georeferencing a Map . . . 398
14.2.1 ★☆☆ Scan the map . . . 398
14.2.2 ★☆☆ Follow Along: Georeferencing the scanned map . . . 399
14.2.3 In Conclusion . . . 405
14.2.4 What's Next? . . . 405
14.3 Lesson: Digitizing Forest Stands . . . 405
14.3.1 ★☆☆ Follow Along: Extracting the Forest Stands Borders . . . 405
14.3.2 ★☆☆ Try Yourself: Georeference the Green Pixels Image . . . 408
14.3.3 ★☆☆ Follow Along: Creating Supporting Points for Digitizing . . . 408
14.3.4 ★☆☆ Follow Along: Digitize the Forest Stands . . . 411
14.3.5 ★☆☆ Try Yourself: Finish Digitizing the Forest Stands . . . 415
14.3.6 ★☆☆ Follow Along: Joining the Forest Stand Data . . . 416
14.3.7 ★☆☆ Try Yourself: Adding Area and Perimeter . . . 419
14.3.8 In Conclusion . . . 420
14.3.9 What's Next? . . . 420
14.4 Lesson: Updating Forest Stands . . . 420
14.4.1 ★☆☆ Comparing the Old Forest Stands to Current Aerial Photographs . . . 420
14.4.2 ★☆☆ Interpreting the CIR Image . . . 421
14.4.3 ★☆☆ Try Yourself: Digitizing Forest Stands from CIR Imagery . . . 423
14.4.4 ★☆☆ Follow Along: Updating Forest Stands with Conservation Information . . . 425
14.4.5 ★☆☆ Try Yourself: Updating Forest Stands with Distance to the Stream . . . 429
14.4.6 In Conclusion . . . 429
14.4.7 What's Next? . . . 430
14.5 Lesson: Systematic Sampling Design . . . 430
14.5.1 Inventariar a Floresta . . . 430
14.5.2 ★☆☆ Follow Along: Implementing a Systematic Sampling Plot Design . . . 430
14.5.3 ★☆☆ Follow Along: Exporting Sample Plots as GPX format . . . 434
14.5.4 In Conclusion . . . 435
14.5.5 What's Next? . . . 435
14.6 Lesson: Creating Detailed Maps with the Atlas Tool . . . 435
14.6.1 ★☆☆ Follow Along: Preparing the Print Layout . . . 435
14.6.2 ★☆☆ Follow Along: Adding Background Map . . . 437
14.6.3 ★☆☆ Try Yourself: Changing the Symbology of the Layers . . . 437
14.6.4 ★☆☆ Try Yourself: Create a Basic Map Template . . . 439
14.6.5 ★☆☆ Follow Along: Adding More Elements to the Print Layout . . . 439
14.6.6 ★☆☆ Follow Along: Creating an Atlas Coverage . . . 442
14.6.7 ★☆☆ Follow Along: Setting Up the Atlas Tool . . . 444
14.6.8 ★☆☆ Follow Along: Editing the Coverage Layer . . . 446
14.6.9 ★☆☆ Follow Along: Printing the Maps . . . 449
14.6.10 In Conclusion . . . 449
14.6.11 What's Next? . . . 450
14.7 Lesson: Calculating the Forest Parameters . . . 450
14.7.1 ★☆☆ Follow Along: Adding the Inventory Results . . . 450
14.7.2 ★☆☆ Follow Along: Whole Forest Parameters Estimation . . . 451
14.7.3 ★☆☆ Follow Along: Estimating Stand Parameters . . . 451
14.7.4 In Conclusion . . . 455
14.7.5 What's Next? . . . 456
14.8 Lesson: DEM from LiDAR Data . . . 456
14.8.1 ★☆☆ Follow Along: Installing Lastools . . . 456
14.8.2 ★☆☆ Follow Along: Calculating a DEM with LAStools . . . 458
14.8.3 ★☆☆ Follow Along: Creating a Terrain Hillshade . . . 463
14.8.4 In Conclusion . . . 465
14.8.5 What's Next? . . . 466
14.9 Lesson: Map Presentation . . . 466
14.9.1 ★☆☆ Follow Along: Preparing the Map Data . . . 466
14.9.2 ★☆☆ Try Yourself: Try Different Blending Modes . . . 468
14.9.3 ★☆☆ Try Yourself: Using a Layout Template to Create the Map result . . . 469

14.9.4 In Conclusion . . . 472

15 Module: Database Concepts with PostgreSQL 473
15.1 Lesson: Introduction to Databases . . . 473
15.1.1 What is a Database? . . . 473
15.1.2 Tabelas . . . 473
15.1.3 Columns / Fields . . . 474
15.1.4 Records . . . 474
15.1.5 Datatypes . . . 474
15.1.6 Modelling an Address Database . . . 475
15.1.7 Database Theory . . . 475
15.1.8 Normalisation . . . 475
15.1.9 ★☆☆ Try Yourself: . . . 476
15.1.10 Indexes . . . 476
15.1.11 Sequences . . . 477
15.1.12 Entity Relationship Diagramming . . . 477
15.1.13 Constraints, Primary Keys and Foreign Keys . . . 479
15.1.14 Transactions . . . 480
15.1.15 In Conclusion . . . 480
15.1.16 What's Next? . . . 480
15.2 Lesson: Implementing the Data Model . . . 480
15.2.1 Install PostgreSQL . . . 480
15.2.2 Ajuda . . . 481
15.2.3 Create a database user . . . 481
15.2.4 Verify the new account . . . 481
15.2.5 Create a database . . . 482
15.2.6 Starting a database shell session . . . 482
15.2.7 Make Tables in SQL . . . 482
15.2.8 Create Keys in SQL . . . 484
15.2.9 Create Indexes in SQL . . . 485
15.2.10 Dropping Tables in SQL . . . 485
15.2.11 A word on pgAdmin III . . . 486
15.2.12 In Conclusion . . . 486
15.2.13 What's Next? . . . 486
15.3 Lesson: Adding Data to the Model . . . 486
15.3.1 Insert statement . . . 486
15.3.2 Sequencing Data Addition According to Constraints . . . 487
15.3.3 Try Yourself: ★★☆ . . . 487
15.3.4 Select data . . . 488
15.3.5 Update data . . . 488
15.3.6 Delete Data . . . 488
15.3.7 Try Yourself: ★★★ . . . 489
15.3.8 In Conclusion . . . 489
15.3.9 What's Next? . . . 489
15.4 Lesson: Queries . . . 489
15.4.1 Ordering Results . . . 489
15.4.2 Filtering . . . 490
15.4.3 Joins . . . 491
15.4.4 Sub-Select . . . 491
15.4.5 Aggregate Queries . . . 492
15.4.6 In Conclusion . . . 493
15.4.7 What's Next? . . . 493
15.5 Lesson: Views . . . 493
15.5.1 Creating a View . . . 493
15.5.2 Modifying a View . . . 494
15.5.3 Dropping a View . . . 494
15.5.4 In Conclusion . . . 494
15.5.5 What's Next? . . . 494

15.6 Lesson: Rules . . . 495
15.6.1 Creating a logging rule . . . 495
15.6.2 In Conclusion . . . 495
15.6.3 What's Next? . . . 496

16 Module: Spatial Database Concepts with PostGIS **497**
16.1 Lesson: PostGIS Setup . . . 497
16.1.1 Installing under Ubuntu . . . 498
16.1.2 Installing under Windows . . . 498
16.1.3 Installing on Other Platforms . . . 498
16.1.4 Configuring Databases to use PostGIS . . . 498
16.1.5 Looking at the installed PostGIS functions . . . 499
16.1.6 Spatial Reference Systems . . . 500
16.1.7 In Conclusion . . . 500
16.1.8 What's Next? . . . 501
16.2 Lesson: Simple Feature Model . . . 501
16.2.1 What is OGC . . . 501
16.2.2 What is the SFS Model . . . 501
16.2.3 Add a geometry field to table . . . 502
16.2.4 Add a constraint based on geometry type . . . 502
16.2.5 ★★★ Try Yourself: . . . 502
16.2.6 Populate geometry_columns table . . . 502
16.2.7 Add geometry record to table using SQL . . . 503
16.2.8 In Conclusion . . . 506
16.2.9 What's Next? . . . 506
16.3 Lesson: Import and Export . . . 506
16.3.1 shp2pgsql . . . 506
16.3.2 pgsql2shp . . . 507
16.3.3 ogr2ogr . . . 507
16.3.4 Gestor BD . . . 507
16.3.5 In Conclusion . . . 507
16.3.6 What's Next? . . . 507
16.4 Lesson: Spatial Queries . . . 508
16.4.1 Spatial Operators . . . 508
16.4.2 Spatial Indexes . . . 508
16.4.3 Try Yourself: ★★☆ . . . 509
16.4.4 PostGIS Spatial Functions Demo . . . 509
16.4.5 In Conclusion . . . 515
16.4.6 What's Next? . . . 515
16.5 Lesson: Geometry Construction . . . 515
16.5.1 Creating Linestrings . . . 515
16.5.2 Try Yourself: ★★☆ . . . 515
16.5.3 Creating Polygons . . . 516
16.5.4 Exercise: Linking Cities to People . . . 516
16.5.5 Looking at Our Schema . . . 518
16.5.6 Try Yourself: ★★★ . . . 518
16.5.7 Access Sub-Objects . . . 519
16.5.8 Data Processing . . . 519
16.5.9 Clipping . . . 519
16.5.10 Building Geometries from Other Geometries . . . 521
16.5.11 Geometry Cleaning . . . 522
16.5.12 Differences between tables . . . 522
16.5.13 Tablespaces . . . 523
16.5.14 In Conclusion . . . 523

17 Guia de Processamento QGIS **525**
17.1 Introdução . . . 525
17.2 Aviso importante antes de começar . . . 526

17.3 Setting-up the processing framework . . . 527
17.4 Running our first algorithm. The toolbox . . . 529
17.5 Mais algoritmos e tipo de dados . . . 533
17.6 SRC. Reprojetar . . . 540
17.7 Selecção . . . 543
17.8 A correr algoritmo externo . . . 545
17.9 Registo de processamento . . . 551
17.9.1 ★☆☆ Follow Along . . . 552
17.9.2 ★★☆ Follow Along . . . 552
17.10 The raster calculator. No-data values . . . 552
17.11 Calculadora vetorial . . . 557
17.12 Defining extents . . . 561
17.13 HTML outputs . . . 565
17.14 First analysis example . . . 568
17.15 Clipping and merging raster layers . . . 576
17.16 Análise hidrológica . . . 585
17.17 Starting with the model designer . . . 597
17.18 Modelos mais complexos . . . 608
17.19 Numeric calculations in the modeler . . . 613
17.20 A model within a model . . . 618
17.21 Using modeler-only tools for creating a model . . . 619
17.22 Interpolação . . . 625
17.23 More interpolation . . . 634
17.24 Iterative execution of algorithms . . . 641
17.25 More iterative execution of algorithms . . . 645
17.26 O interface do processamento em lote . . . 647
17.27 Models in the batch processing interface . . . 650
17.28 Pré- e pós-execução de encaixe da script . . . 651
17.29 Outros programas . . . 652
17.29.1 GRASS . . . 652
17.29.2 R . . . 652
17.29.3 Others . . . 653
17.29.4 Comparison among backends . . . 653
17.30 Interpolation and contouring . . . 654
17.30.1 Interpolação . . . 654
17.30.2 Contour . . . 654
17.31 Vector simplification and smoothing . . . 655
17.32 Planning a solar farm . . . 655
17.33 Utilize as scripts R no Processing . . . 656
17.33.1 Adicionar scripts . . . 656
17.33.2 Creating plots . . . 657
17.33.3 Create a vector . . . 661
17.33.4 Text and graph output from R - syntax . . . 664
17.34 Predicting landslides . . . 664

18 Module: Using Spatial Databases in QGIS **667**
18.1 Lesson: Working with Databases in the QGIS Browser . . . 667
18.1.1 ★☆☆ Follow Along: Adding Database Tables to QGIS using the Browser . . . 667
18.1.2 ★☆☆ Follow Along: Adding a filtered set of records as a Layer . . . 669
18.1.3 In Conclusion . . . 670
18.1.4 What's Next? . . . 670
18.2 Lesson: Using DB Manager to work with Spatial Databases in QGIS . . . 670
18.2.1 ★☆☆ Follow Along: Managing PostGIS Databases with DB Manager . . . 670
18.2.2 ★☆☆ Follow Along: Creating a New Table . . . 676
18.2.3 ★☆☆ Follow Along: Basic Database Administration . . . 677
18.2.4 ★☆☆ Follow Along: Executing SQL Queries with DB Manager . . . 678
18.2.5 Importing Data into a Database with DB Manager . . . 679
18.2.6 Exporting Data from a Database with DB Manager . . . 681

18.2.7 In Conclusion . . . 683
18.2.8 What's Next? . . . 683
18.3 Lesson: Working with SpatiaLite databases in QGIS . . . 683
18.3.1 ★☆☆ Follow Along: Creating a SpatiaLite database with the Browser . . . 683
18.3.2 In Conclusion . . . 685

19 Appendix: Contributing To This Manual **687**
19.1 Downloading Resources . . . 687
19.2 Manual Format . . . 687
19.3 Adding a Module . . . 687
19.4 Adding a Lesson . . . 688
19.5 Adding a Section . . . 689
19.5.1 Adding a «follow along» section . . . 689
19.5.2 Adding a «try yourself» section . . . 690
19.6 Add a Conclusion . . . 690
19.7 Add a Further Reading Section . . . 690
19.8 Add a What's Next Section . . . 690
19.9 Using Markup . . . 691
19.9.1 New concepts . . . 691
19.9.2 Emphasis . . . 691
19.9.3 Images . . . 691
19.9.4 Internal links . . . 691
19.9.5 External links . . . 692
19.9.6 Using monospaced text . . . 692
19.9.7 Labeling GUI items . . . 692
19.9.8 Menu selections . . . 692
19.9.9 Adding notes . . . 692
19.9.10 Adding a sponsorship/authorship note . . . 692
19.10 Thank You! . . . 693

20 A preparar os dados do exercício **695**
20.1 Create OSM based vector files . . . 695
20.2 Create SRTM DEM tiff files . . . 702
20.3 Create imagery tiff files . . . 702
20.4 Replace tokens . . . 703

CAPÍTULO 1

Introdução do curso

1.1 Prefácio

Welcome to our course! We will be showing you how to use QGIS easily and efficiently. If you are new to GIS, we will tell you what you need to get started. If you are an experienced user, you will see how QGIS fulfills all the functions you expect from a GIS program, and more!

1.1.1 Why QGIS?

As information becomes increasingly spatially aware, there is no shortage of tools able to fulfill some or all commonly used GIS functions. Why should anyone be using QGIS over some other GIS software package?

Here are only some of the reasons:

- *It's free, as in lunch.* Installing and using the QGIS program costs you a grand total of zero money. No initial fee, no recurring fee, nothing.
- *It's free, as in liberty.* If you need extra functionality in QGIS, you can do more than just hope it will be included in the next release. You can sponsor the development of a feature, or add it yourself if you are familiar with programming.
- *It's constantly developing.* Because anyone can add new features and improve on existing ones, QGIS never stagnates. The development of a new tool can happen as quickly as you need it to.
- *Extensive help and documentation is available.* If you're stuck with anything, you can turn to the extensive documentation, your fellow QGIS users, or even the developers.
- *Cross-platform.* QGIS can be installed on macOS, Windows and Linux.

Now that you know why you want to use QGIS, these exercises will make you know how.

1.1.2 Enquadramento

In 2008 we launched the Gentle Introduction to GIS, a completely free, open content resource for people who want to learn about GIS without being overloaded with jargon and new terminology. It was sponsored by the South African government and has been a phenomenal success, with people all over the world writing to us to tell us how they are using the materials to run University Training Courses, teach themselves GIS and so on. The Gentle Introduction is not a software tutorial, but rather aims to be a generic text (although we used QGIS in all examples) for someone learning about GIS. There is also the QGIS manual which provides a detailed functional overview of the QGIS application. However, it is not structured as a tutorial, but rather as a reference guide. At Linfiniti Consulting CC. we frequently run training courses and have realised that a third resource is needed - one that leads the reader sequentially through learning the key aspects of QGIS in a trainer-trainee format - which prompted us to produce this work.

This training manual is intended to provide all the materials needed to run a 5 day course on QGIS, PostgreSQL and PostGIS. The course is structured with content to suit novice, intermediate and advanced users alike and has many exercises complete with annotated answers throughout the text.

1.1.3 Licença

The Free Quantum GIS Training Manual by Linfiniti Consulting CC. is based on an earlier version from Linfiniti and is licensed under a Creative Commons Attribution 4.0 International. Permissions beyond the scope of this license may be available at below.

We have published this QGIS training manual under a liberal license that allows you to freely copy, modify and redistribute this work. A complete copy of the license is available at the end of this document. In simple terms, the usage guidelines are as follows:

- You may not represent this work as your own work, or remove any authorship text or credits from this work.
- You may not redistribute this work under more restrictive permissions than those under which it was provided to you.
- If you add a substantive portion to the work and contribute it back to the project (at least one complete module) you may add your name to the end of the authors list for this document (which will appear on the front page)
- If you contribute minor changes and corrections you may add yourself to the contributors list below.
- If you translate this document in its entirety, you may add your name to the authors list in the form «Translated by Joe Bloggs».
- If you sponsor a module or lesson, you may request the author to include an acknowledgement in the beginning of each lesson contributed, e.g.:

Nota: Esta lição é patrocinada pela MegaCorp.

- Se não tem certeza sobre o que está abrangido por esta licença, por favor contacte-nos através de office@linfiniti.com e daremos apoio sobre o que quer fazer é aceitável.
- If you publish this work under a self publishing site such as https://www.lulu.com we request that you donate the profits to the QGIS project.
- You may not commercialise this work, except with the expressed permission of the authors. To be clear, by commercialisation we mean that you may not sell for profit, create commercial derivative works (e.g. selling content for use as articles in a magazine). The exception to this is if all the profits are given to the QGIS project. You may (and we encourage you to do so) use this work as a text book when conducting training courses, even if the course itself is commercial in nature. In other words, you are welcome to make money by running a

training course that uses this work as a text book, but you may not profit off the sales of the book itself - all such profits should be contributed back to QGIS.

1.1.4 Capítulos Patrocinados

This work is by no means a complete treatise on all the things you can do with QGIS and we encourage others to add new materials to fill any gaps. Linfiniti Consulting CC. can also create additional materials for you as a commercial service, with the understanding that all such works produced should become part of the core content and be published under the same license.

1.1.5 Autores

- Rüdiger Thiede - Rudi has written the QGIS instructional materials and parts of the PostGIS materials.
- Tim Sutton (tim@kartoza.com) - Tim has overseen and guided the project and co-authored the PostgreSQL and PostGIS parts.
- Horst Düster (horst.duester@kappasys.ch) - Horst co-autor de partes do PostgreSQL e PostGIS
- Marcelle Sutton - Marcelle provided proof-reading and editorial advice during the creation of this work.

1.1.6 Contribuidores Individuais

O seu nome aqui!

1.1.7 Patrocinadores

- Universidade de Tecnologia da Península do Cabo

1.1.8 Source files and Issue reports

The source of this document is available at GitHub QGIS Documentation repository. Consult GitHub.com for instructions on how to use the git version control system.

Despite our efforts, you could find some errors or miss some information while following this training. Please report them at https://github.com/qgis/QGIS-Documentation/issues.

1.1.9 Última Versão

You can always obtain the latest version of this document by visiting the online version which is part of the QGIS documentation website (https://www.qgis.org/resources/hub/#documentation).

Nota: The documentation website contains links to both online and PDF versions of the Training manual and other parts of the QGIS documentation.

1.2 About the exercises

Now that you know why you want to use QGIS, we can show you how.

Aviso: This course includes instructions on adding, deleting and altering GIS datasets. We have provided training datasets for this purpose. Before using the techniques described here on your own data, always ensure you have proper backups!

1.2.1 How to use this tutorial

Any text that *looks like this* refers to something that you can see in the QGIS user interface.

Text that *looks* ► *like* ► *this* directs you through menus.

`This kind of text` refers to something you can type, such as a command.

`This/kind/of.text` refers to a path or filename.

`This+That` refers to a keyboard shortcut comprised of two buttons.

1.2.2 Tiered course objectives

This course caters to different user experience levels. Depending on which category you consider yourself to be in, you can expect a different set of course outcomes. Each category contains information that is essential for the next one, so it's important to do all exercises that are at or below your level of experience.

??? (Basic level) Basic

In this category, the course assumes that you have little or no prior experience with theoretical GIS knowledge or the operation of GIS software.

Limited theoretical background will be provided to explain the purpose of an action you will be performing in the program, but the emphasis is on learning by doing.

When you complete the course, you will have a better concept of the possibilities of GIS, and how to harness their power via QGIS.

??? (Moderate level) Intermediate

In this category, it is assumed that you have working knowledge and experience of the everyday uses of GIS software.

Building on the instructions for the beginner level will provide you with familiar ground, as well as to make you aware of the cases where QGIS does things slightly differently from other software you may be used to. You will also learn how to use analysis functions in QGIS.

When you complete the course, you should be comfortable with using QGIS for all of the functions you usually need for everyday use.

??? (Advanced level) Advanced

In this category, the assumption is that you are experienced with GIS software, have knowledge of and experience with spatial databases, using data on a remote server, perhaps writing scripts for analysis purposes, etc.

Building on the instructions for the other two levels will familiarize you with the approach that the QGIS interface follows, and will ensure that you know how to access the basic functions that you need. You will also be shown how to make use of the QGIS plugin system, database access, and so on.

When you complete the course, you should be well-acquainted with the everyday operation of QGIS, as well as its more advanced functions.

1.2.3 Data

The sample data that accompanies this resource is freely available and comes from the following sources:

- Streets and Places datasets from OpenStreetMap
- Property boundaries (urban and rural), water bodies from NGI
- SRTM DEM from the CGIAR CSI

Download the prepared dataset from the Training data repository and unzip the file. All the necessary data are provided in the `exercise_data` folder.

If you are an instructor, and would like to use more relevant data, you will find instructions for creating local data in the *A preparar os dados do exercício* appendix.

CAPÍTULO 2

Module: Creating and Exploring a Basic Map

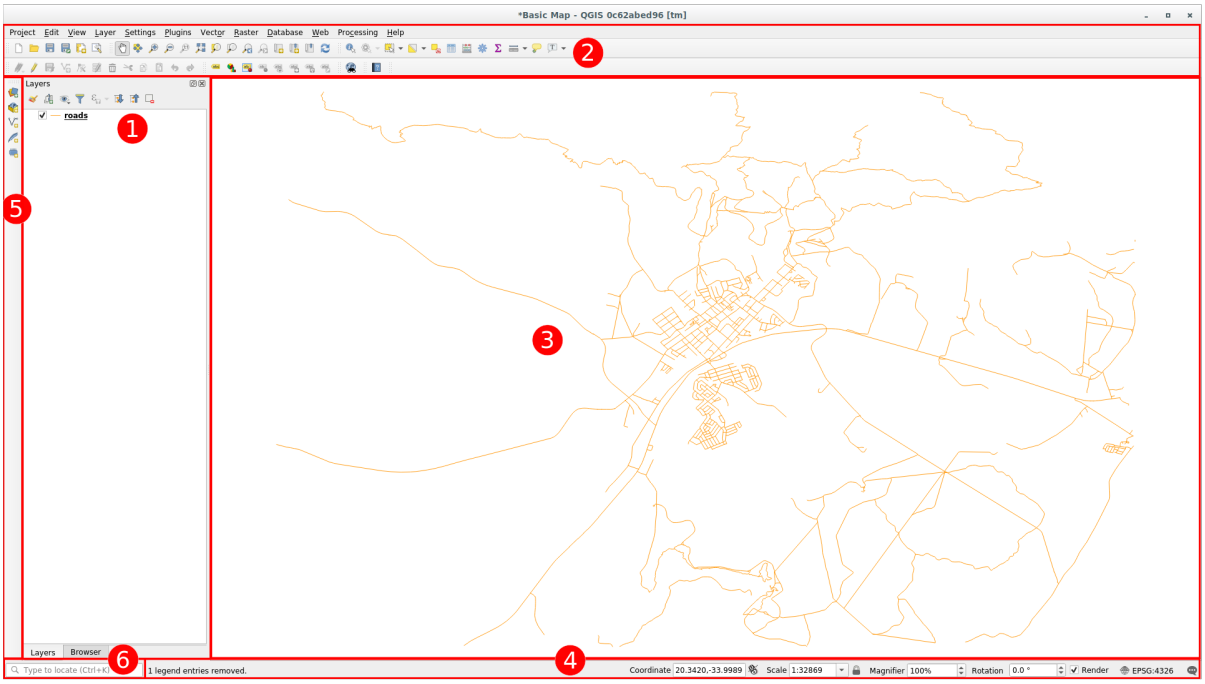
Neste módulo, irá criar um mapa simples que será usado mais tarde como base para demonstrar outras funcionalidades do QGIS.

2.1 Lesson: An Overview of the Interface

We will explore the QGIS user interface so that you are familiar with the menus, toolbars, map canvas and layers list that form the basic structure of the interface.

The goal for this lesson: To understand the basics of the QGIS user interface.

2.1.1 ??? Try Yourself: The Basics



The elements identified in the figure above are:

1. Layers List / Browser Panel
2. Barras de Ferramentas
3. Map canvas
4. Status bar
5. Side Toolbar
6. Locator bar

??? The Layers List

In the Layers list, you can see a list, at any time, of all the layers available to you.

Expanding collapsed items (by clicking the arrow or plus symbol beside them) will provide you with more information on the layer's current appearance.

Hovering over the layer will give you some basic information: layer name, type of geometry, coordinate reference system and the complete path of the location on your device.

Right-clicking on a layer will give you a menu with lots of extra options. You will be using some of them before long, so take a look around!

Nota: A vector layer is a dataset, usually of a specific kind of object, such as roads, trees, etc. A vector layer can consist of either points, lines or polygons.

The Browser Panel

The QGIS Browser is a panel in QGIS that lets you easily navigate in your database. You can have access to common vector files (e.g. ESRI Shapefile or MapInfo files), databases (e.g. PostGIS, Oracle, SpatiaLite, GeoPackage or MS SQL Server) and WMS/WFS connections. You can also view your GRASS data.

If you have saved a project, the Browser Panel will also give you quick access to all the layers stored in the same path of the project file under in the *Project Home* item.

Moreover, you can set one or more folder as **Favorites**: search under your path and once you have found the folder, right click on it and click on `Add as a Favorite`. You should then be able to see your folder in the *Favorites* item.

Dica: It can happen that the folders added to Favorite item have a really long name: don't worry right-click on the path and choose `Rename Favorite...` to set another name.

Toolbars

Your most often used sets of tools can be turned into toolbars for basic access. For example, the *Project* toolbar allows you to save, load, print, and start a new project. You can easily customize the interface to see only the tools you use most often, adding or removing toolbars as necessary via the *View* ► *Toolbars* menu.

Even if they are not visible in a toolbar, all of your tools will remain accessible via the menus. For example, if you remove the *Project* toolbar (which contains the *Save* button), you can still save your map by clicking on the *Project* menu and then clicking on *Save*.

The Map Canvas

This is where the map itself is displayed and where layers are loaded. In the map canvas you can interact with the visible layers: zoom in/out, move the map, select features and many other operations that we will deeply see in the next sections.

The Status Bar

Shows you information about the current map. Also allows you to adjust the map scale, the map rotation and see the mouse cursor's coordinates on the map.

The Side Toolbar

By default the Side toolbar contains the buttons to load the layer and all the buttons to create a new layer. But remember that you can move all the toolbars wherever it is more comfortable for you.

The Locator Bar

Within this bar you can access to almost all the objects of QGIS: layers, layer features, algorithms, spatial bookmarks, etc. Check all the different options in the locator_options section of the QGIS User Manual.

Dica: With the shortcut `Ctrl+K` you can easily access the bar.

2.1.2 ??? Try Yourself: 1

Try to identify the four elements listed above on your own screen, without referring to the diagram above. See if you can identify their names and functions. You will become more familiar with these elements as you use them in the coming days.

Answer

Refer back to the image showing the interface layout and check that you remember the names and functions of the screen elements.

2.1.3 ??? Try Yourself: 2

Try to find each of these tools on your screen. What is their purpose?

1.
2.
3.
4. *Render*
5.

Nota: If any of these tools is not visible on the screen, try enabling some toolbars that are currently hidden. Also keep in mind that if there isn't enough space on the screen, a toolbar may be shortened by hiding some of its tools. You can see the hidden tools by clicking on the double right arrow button in any such collapsed toolbar. You can see a tooltip with the name of any tool by holding your mouse over the tool for a while.

Answer

1. *Save as*
2. *Zoom to layer(s)*
3. *Invert selection*
4. *Rendering on/off*
5. *Measure line*

2.1.4 What's Next?

Now that you are familiar with the basics of the QGIS interface, in the next lesson we will see how to load some common data types.

2.2 Lesson: Adding your first layers

We will start the application, and create a basic map to use for examples and exercises.

The goal for this lesson: To get started with an example map.

Nota: Before starting this exercise, QGIS must be installed on your computer. Also, you should have downloaded the *sample data* to use.

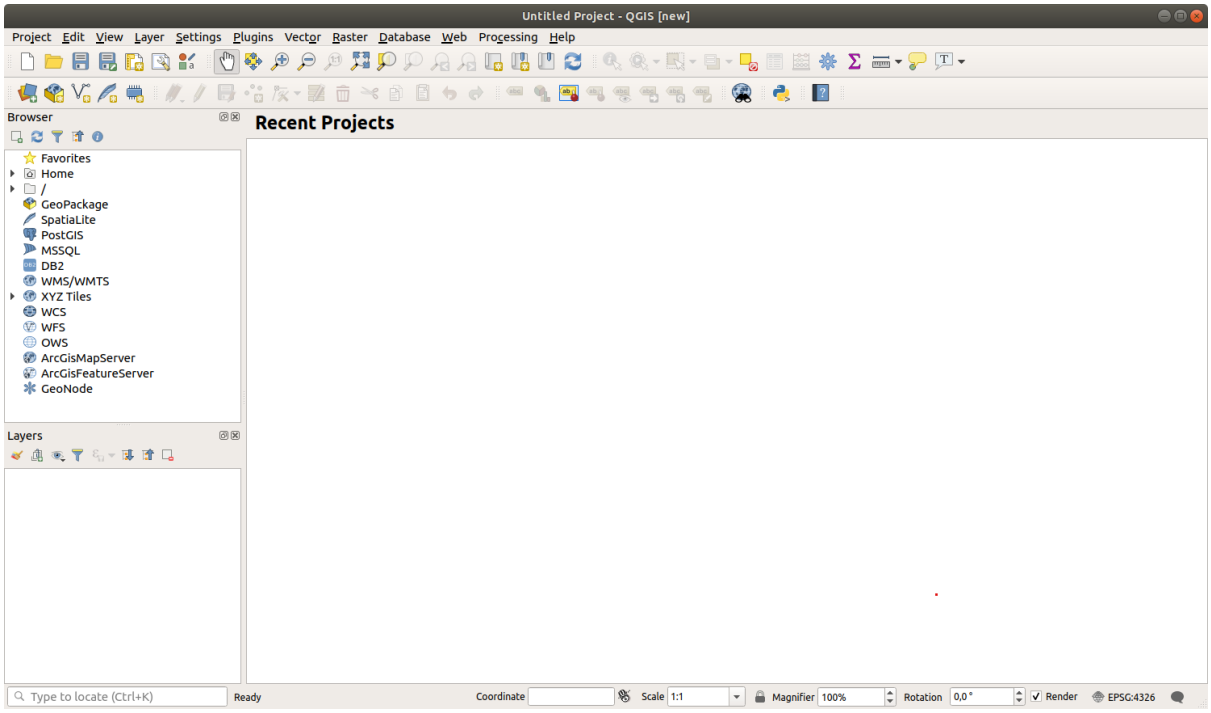
Launch QGIS from its desktop shortcut, menu item, etc., depending on how you configured its installation.

Nota: The screenshots for this course were taken in QGIS 3.4 running on Linux. Depending on your setup, the screens you encounter may well appear somewhat different. However, all the same buttons will still be available, and the instructions will work on any OS. You will need QGIS 3.4 (the latest version at time of writing) to use this course.

Let's get started right away!

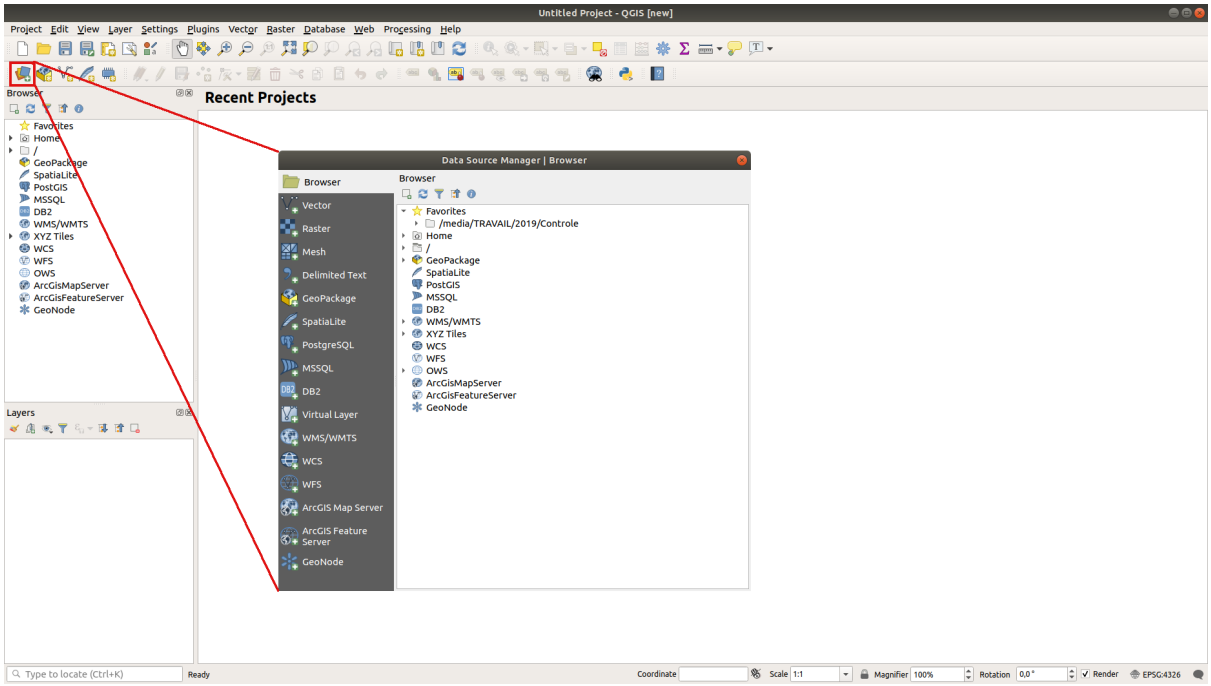
2.2.1 ??? Follow Along: Prepare a map

1. Open QGIS. You will have a new, blank map.

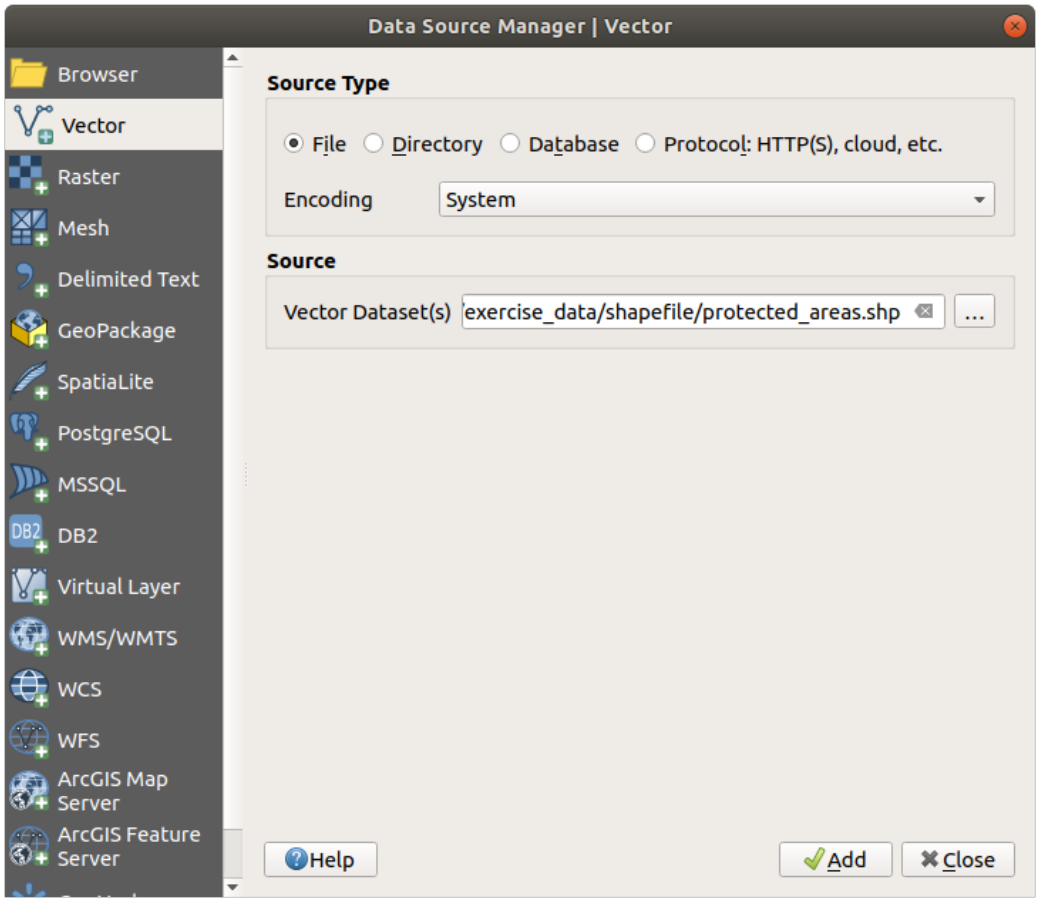


2. The *Data Source Manager* dialog allows you to choose the data to load depending on the data type. We'll use it to load our dataset: click the Open Data Source Manager button.

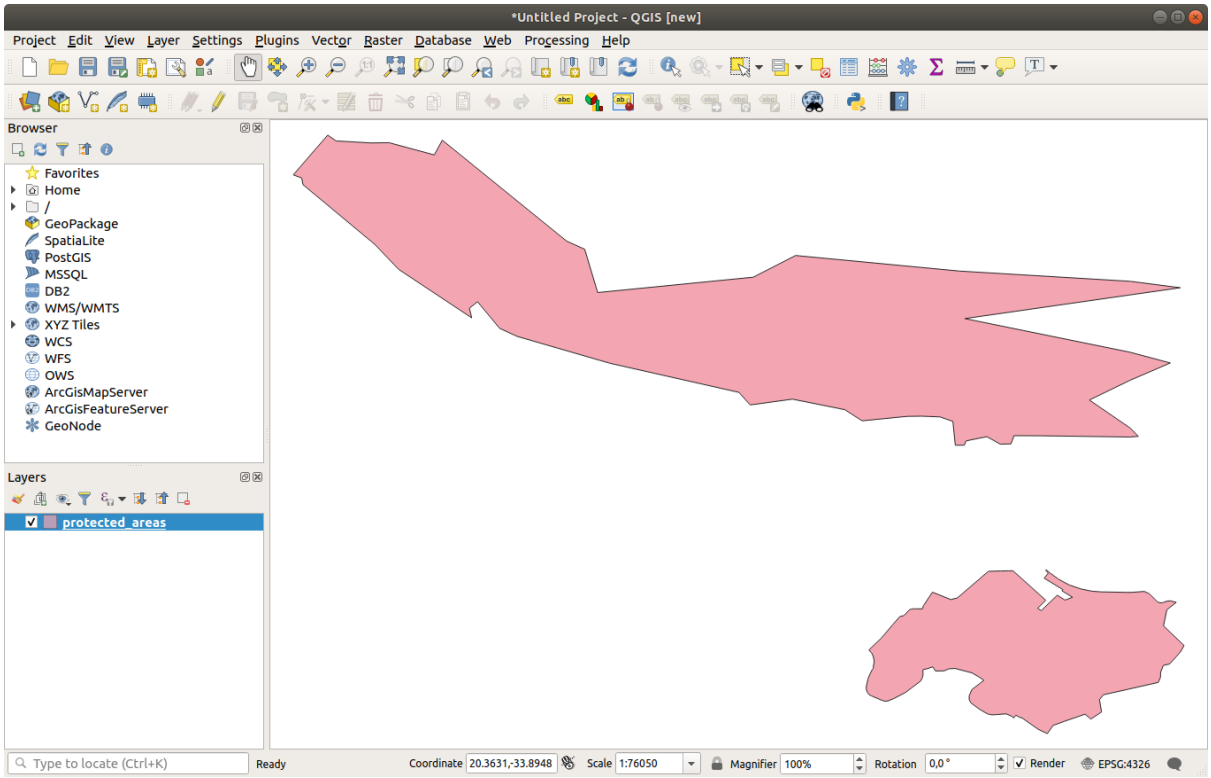
   If you can't find the icon, check that the *Data Source Manager* toolbar is enabled in the *View* ► *Toolbars* menu.



3. Load the `protected_areas.shp` vector dataset:
    1. Click on the *Vector* tab.
    2. Enable the *File* source type.
    3. Press the … button next to *Vector Dataset(s)*.
    4. Select the `exercise_data/shapefile/protected_areas.shp` file in your training directory.
    5. Click *Open*. You will see the original dialog, with the file path filled in.



6. Click *Add* here as well. The data you specified will now load: you can see a `protected_areas` item in the *Layers* panel (bottom left) with its features shown in the main map canvas.



Congratulations! You now have a basic map. Now would be a good time to save your work.

1. Click on the *Save As* button:
2. Save the map under a `solution` folder next to `exercise_data` and call it `basic_map.qgz`.

2.2.2 ??? Try Yourself:

Repeat the steps above to add the `places.shp` and `rivers.shp` layers from the same folder (`exercise_data/shapefile`) to the map.

Answer

In the main area of the dialog you should see many shapes with different colors. Each shape belongs to a layer you can identify by its color in the left panel (your colors may be different from the ones in Fig.2.1 below):

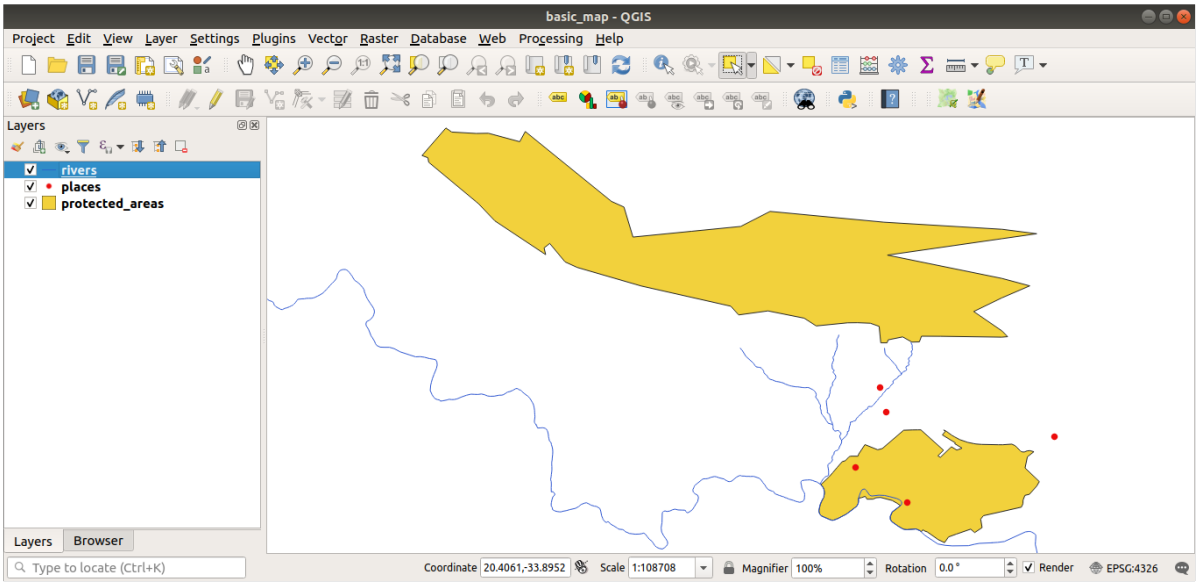


Figura2.1: A basic map

2.2.3 ??? Follow Along: Loading vector data from a GeoPackage Database

Databases allow you to store a large volume of associated data in one file. You may already be familiar with a database management system (DBMS) such as Libreoffice Base or MS Access. GIS applications can also make use of databases. GIS-specific DBMSes (such as PostGIS) have extra functions, because they need to handle spatial data.

The GeoPackage open format is a container that allows you to store GIS data (layers) in a single file. Unlike the ESRI Shapefile format (e.g. the `protected_areas.shp` dataset you loaded earlier), a single GeoPackage file can contain various data (both vector and raster data) in different coordinate reference systems, as well as tables without spatial information; all these features allow you to share data easily and avoid file duplication.

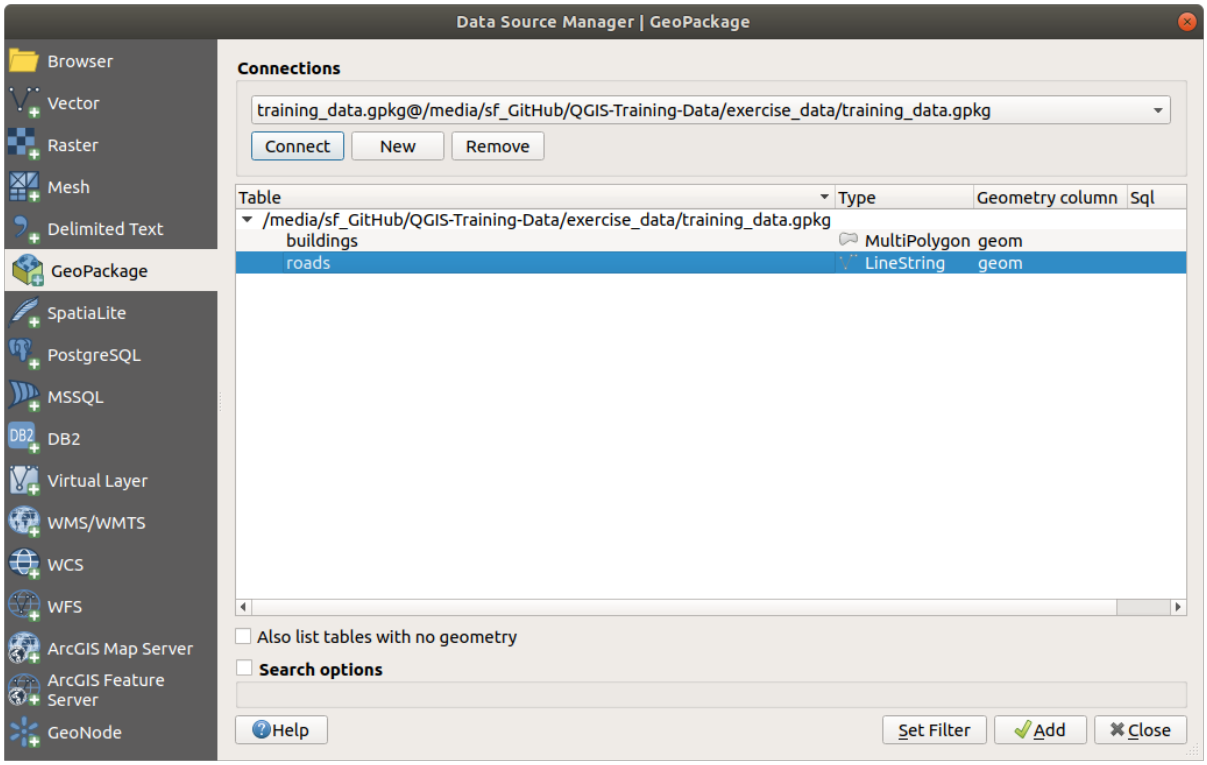
In order to load a layer from a GeoPackage, you will first need to create the connection to it:

1. Click on the Open Data Source Manager button.
2. On the left click on the *GeoPackage* tab.
3. Click on the *New* button and browse to the `training_data.gpkg` file in the `exercise_data` folder you downloaded before.

4. Select the file and press *Open*. The file path is now added to the Geopackage connections list, and appears in the drop-down menu.

You are now ready to add any layer from this GeoPackage to QGIS.

1. Click on the *Connect* button. In the central part of the window you should now see the list of all the layers contained in the GeoPackage file.
2. Select the *roads* layer and click on the *Add* button.



   A *roads* layer is added to the *Layers* panel with features displayed on the map canvas.

3. Click on *Close*.

Congratulations! You have loaded the first layer from a GeoPackage.

2.2.4 ??? Follow Along: Loading vector data from a SpatiaLite Database with the Browser

QGIS provides access to many other database formats. Like GeoPackage, the SpatiaLite database format is an extension of the SQLite library. And adding a layer from a SpatiaLite provider follows the same rules as described above: Create the connection –> Enable it –> Add the layer(s).

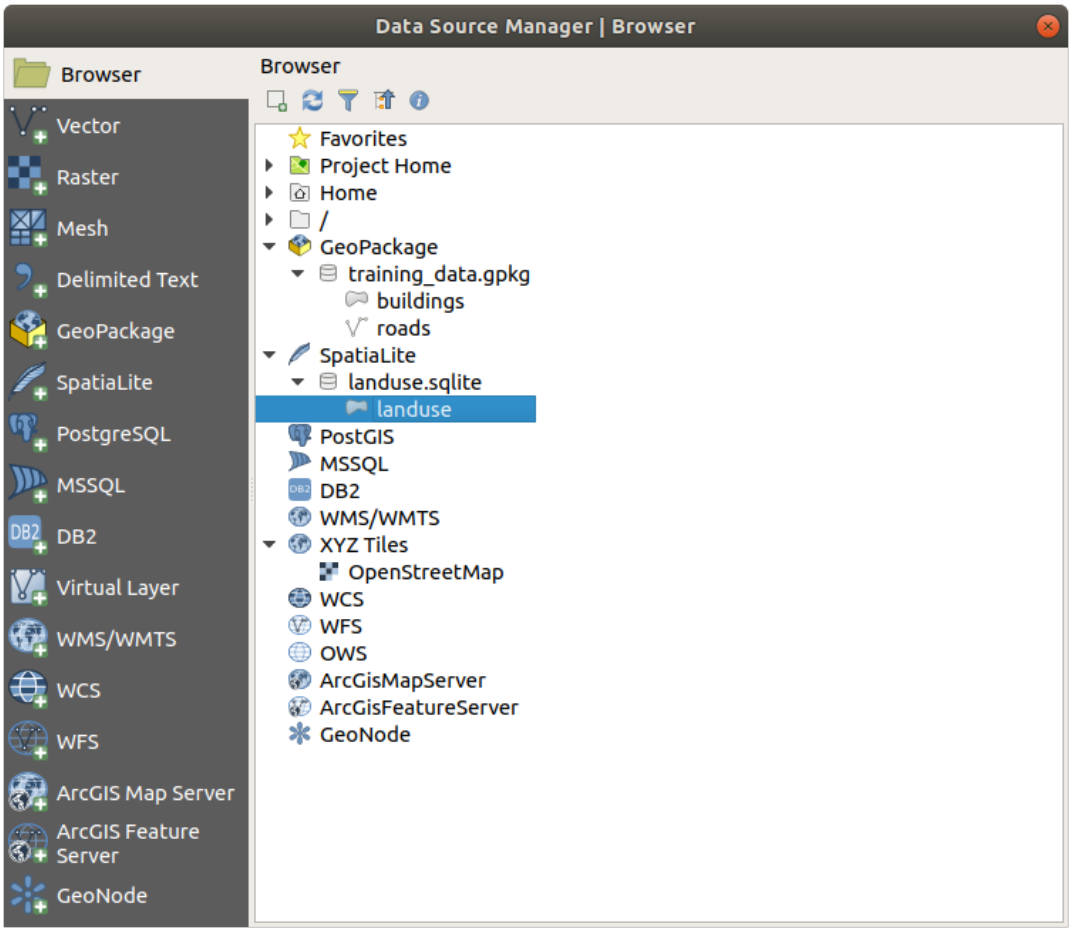
While this is one way to add SpatiaLite data to your map, let's explore another powerful way to add data: the *Browser*.

1. Click the icon to open the *Data Source Manager* window.
2. Click on the *Browser* tab.
3. In this tab you can see all the storage disks connected to your computer as well as entries for most of the tabs in the left. These allow quick access to connected databases or folders.

   For example, click on the drop-down icon next to the *GeoPackage* entry. You'll see the `training-data.gpkg` file we previously connected to (and its layers, if expanded).

4. Right-click the *SpatiaLite* entry and select *New Connection…*.
5. Navigate to the `exercise_data` folder, select the `landuse.sqlite` file and click *Open*.

   Notice that a *landuse.sqlite* entry has been added under the *SpatiaLite* one.
6. Expand the *landuse.sqlite* entry.
7. Double-click the *landuse* layer or select and drag-and-drop it onto the map canvas. A new layer is added to the *Layers* panel and its features are displayed on the map canvas.



Dica: Enable the *Browser* panel in *View* ► *Panels* ► and use it to add your data. It's a handy shortcut for the *Data Source Manager* ► *Browser* tab, with the same functionality.

Nota: Remember to save your project frequently! The project file doesn't contain any of the data itself, but it remembers which layers you loaded into your map.

2.2.5 ??? Try Yourself: Load More Vector Data

Load the following datasets from the `exercise_data` folder into your map using any of the methods explained above:

- *buildings*
- *water*

Answer

Your map should have seven layers:

- *protected_areas*
- *places*
- *rivers*
- *roads*
- *landuse*
- *buildings* (taken from `training_data.gpkg`) and
- *water* (taken from `exercise_data/shapefile`).

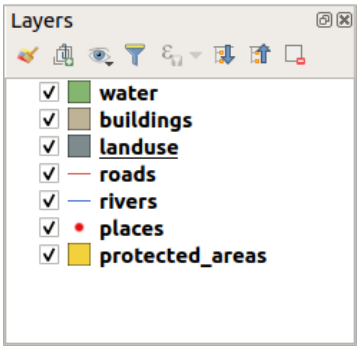
2.2.6 Follow Along: Reordering the Layers

The layers in your Layers list are drawn on the map in a certain order. The layer at the bottom of the list is drawn first, and the layer at the top is drawn last. By changing the order that they are shown on the list, you can change the order they are drawn in.

Nota: You can alter this behavior using the *Control rendering order* checkbox beneath the *Layer Order* panel. We will however not discuss this feature yet.

The order in which the layers have been loaded into the map is probably not logical at this stage. It's possible that the road layer is completely hidden because other layers are on top of it.

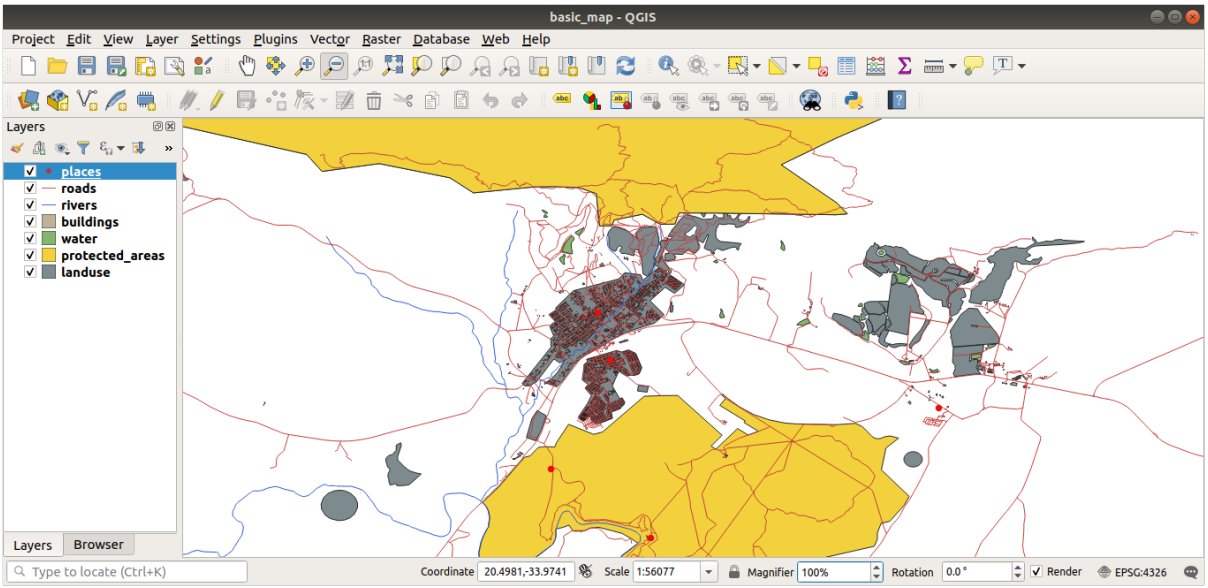
For example, this layer order…



… would result in roads and places being hidden as they run *underneath* the polygons of the landuse layer.

To resolve this problem:

1. Click and drag on a layer in the Layers list.
2. Reorder them to look like this:



You'll see that the map now makes more sense visually, with roads and buildings appearing above the land use regions.

2.2.7 In Conclusion

Now you've added all the layers you need from several different sources and created a basic map!

2.2.8 What's Next?

Now you're familiar with the basic function of the *Open Data Source Manager* button, but what about all the others? How does this interface work? Before we go on, let's take a look at some basic interaction with the QGIS interface. This is the topic of the next lesson.

2.3 Lesson: Navigating the Map Canvas

Esta secção irá focar-se nas ferramentas de navegação básicas do QGIS utilizadas para navegar na «Tela do Mapa». Estas ferramentas irão permitir-lhe explorar visualmente as camadas em escalas diferentes.

O objetivo desta lição: saiba como utilizar as ferramentas «Mover e Ampliar» no QGIS e saiba como escalar o mapa.

2.3.1 ??? Follow Along: Basic Navigation Tools

Antes de aprender a navegar na «Tela do Mapa», vamos adicionar algumas camadas que nós podemos explorar durante este tutorial.

1. Open a new blank project and using the steps learnt in *Create a Map*, load the previously seen `protected_areas`, `roads` and `buildings` layers to the project. The result view should look similar to the snippet in Fig.2.2 below (colors do not matter):

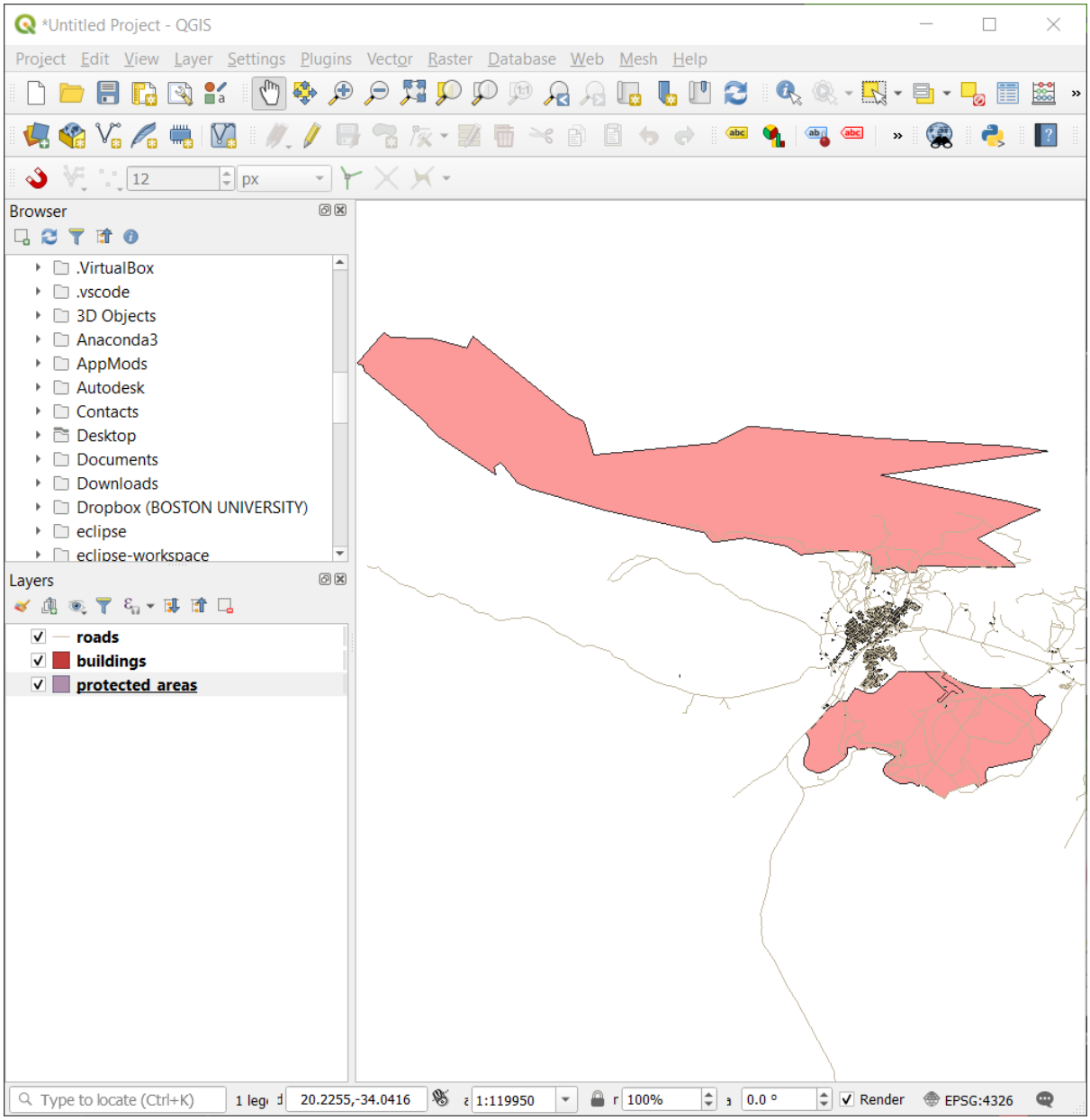


Figura2.2: Áreas protegidas, estradas e edifícios adicionados

Let's first learn how to use the Pan Tool.

1. In the *Map Navigation Toolbar*, make sure the Pan button is activated.
2. Move the mouse to the center of the Map Canvas area.
3. Left-click and hold, and drag the mouse in any direction to pan the map.

Next, let's zoom in and take a closer look at the layers we imported.

1. In the *Map Navigation Toolbar*, click on the Zoom In button.
2. Move your mouse to approximately the top left area of where there is the highest density of buildings and roads.
3. Clique esquerdo e mantenha.
4. Then drag the mouse, which will create a rectangle, and cover the dense area of buildings and roads (Fig.2.3).

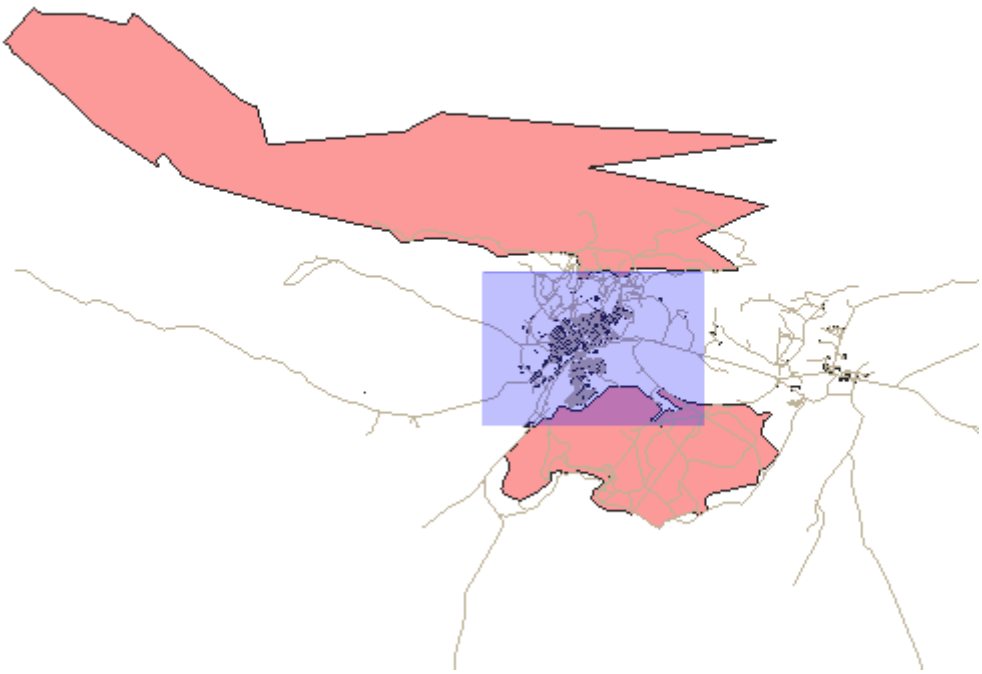
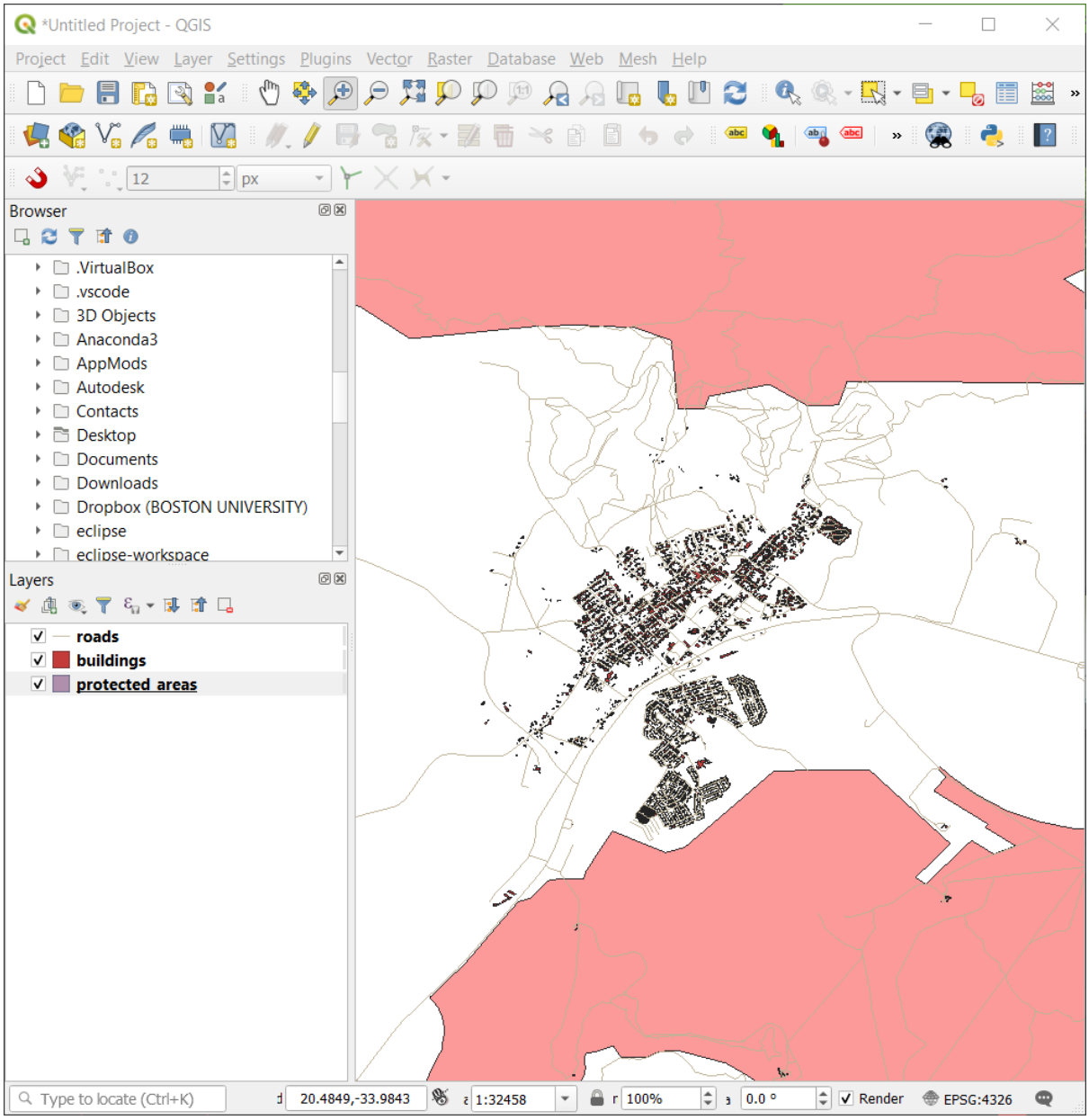


Figura2.3: Zoom +

5. Release the left click. This will zoom in to include the area that you selected with your rectangle.



6. To zoom out, select the Zoom Out button and perform the same action as you did for zooming in.

As you pan, zoom in, or zoom out, QGIS saves these views in a history. This allows you to backtrack to a previous view.

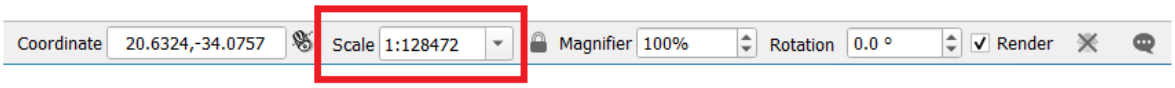
1. In the *Map Navigation Toolbar*, click on Zoom Last button to go to your previous view.

2. Click on Zoom Next button to proceed to move forward in your history.

Sometimes after exploring the data, we need to reset our view to the extent of all the layers. Instead of trying to use the Zoom Out tool multiple times, QGIS provides us with a button to do that action for us.

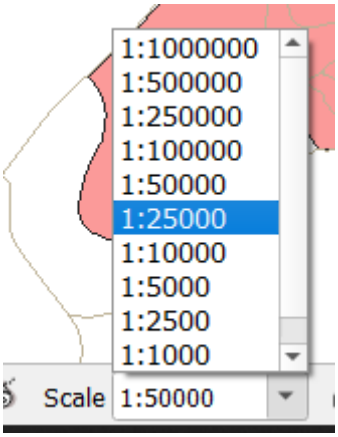
1. Click on the Zoom Full Extent button.

As you zoomed in and out, notice that the *Scale* value in the Status Bar changes. The *Scale* value represents the Map Scale. In general, the number to the right of *:* represents how many times smaller the object you are seeing in the Map Canvas is to the actual object in the real world.



You can also use this field to set the Map Scale manually.

1. In the Status Bar, click on the *Scale* text field.
2. Type in `50000` and press `Enter`. This will redraw the features in the Map Canvas to reflect the scale you typed in.
3. Alternatively, click on the options arrow of the *Scale* field to see the preset map scales.



4. Select *1:5000*. This will also update the map scale in the Map Canvas.

Now you know the basics of navigating the Map Canvas. Check out the User Manual on Zooming and Panning to learn about alternative ways of navigating the Map Canvas.

2.3.2 In Conclusion

Knowing how to navigate the Map Canvas is important, as it allows one to explore and visually inspect the layers. This could be done for initial data exploration, or to validate output of a spatial analysis.

2.4 Lesson: Symbology

The symbology of a layer is its visual appearance on the map. The basic strength of GIS over other ways of representing data with spatial aspects is that with GIS, you have a dynamic visual representation of the data you're working with.

Therefore, the visual appearance of the map (which depends on the symbology of the individual layers) is very important. The end user of the maps you produce will need to be able to easily see what the map represents. Equally as important, you need to be able to explore the data as you're working with it, and good symbology helps a lot.

In other words, having proper symbology is not a luxury or just nice to have. In fact, it's essential for you to use a GIS properly and produce maps and information that people will be able to use.

The goal for this lesson: To be able to create any symbology you want for any vector layer.

2.4.1 ??? Follow Along: Changing Colors

To change a layer's symbology, open its *Layer Properties*. Let's begin by changing the color of the `landuse` layer.

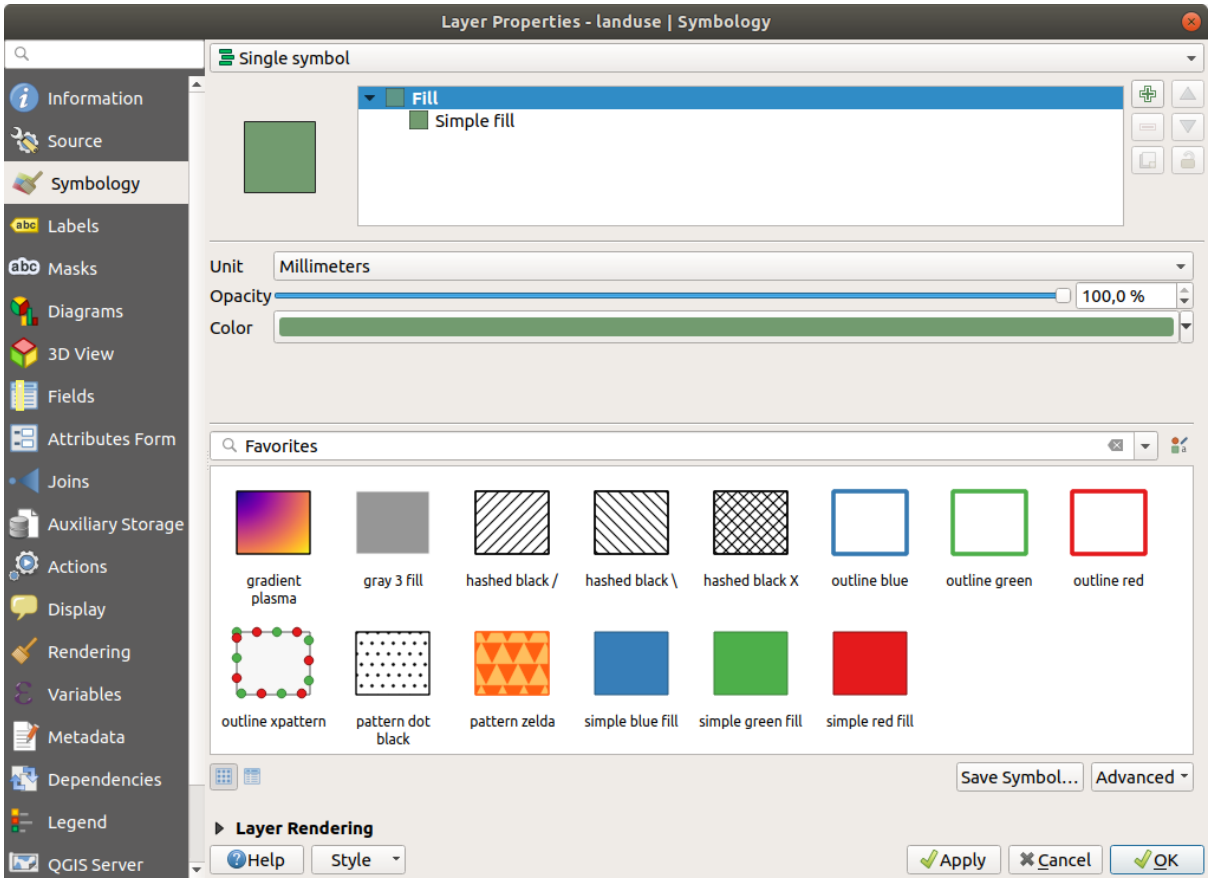
1. Right-click on the `landuse` layer in the layers list.
2. Select the menu item *Properties...* in the menu that appears.

Nota: By default, you can also access a layer's properties by double-clicking on the layer in the Layers list.

Dica: The button at the top of the *Layers* panel will open the *Layer Styling* panel. You can use this panel to change some properties of the layer: by default, changes will be applied immediately!

---

3. In the *Layer Properties* window, select the *Symbology* tab:



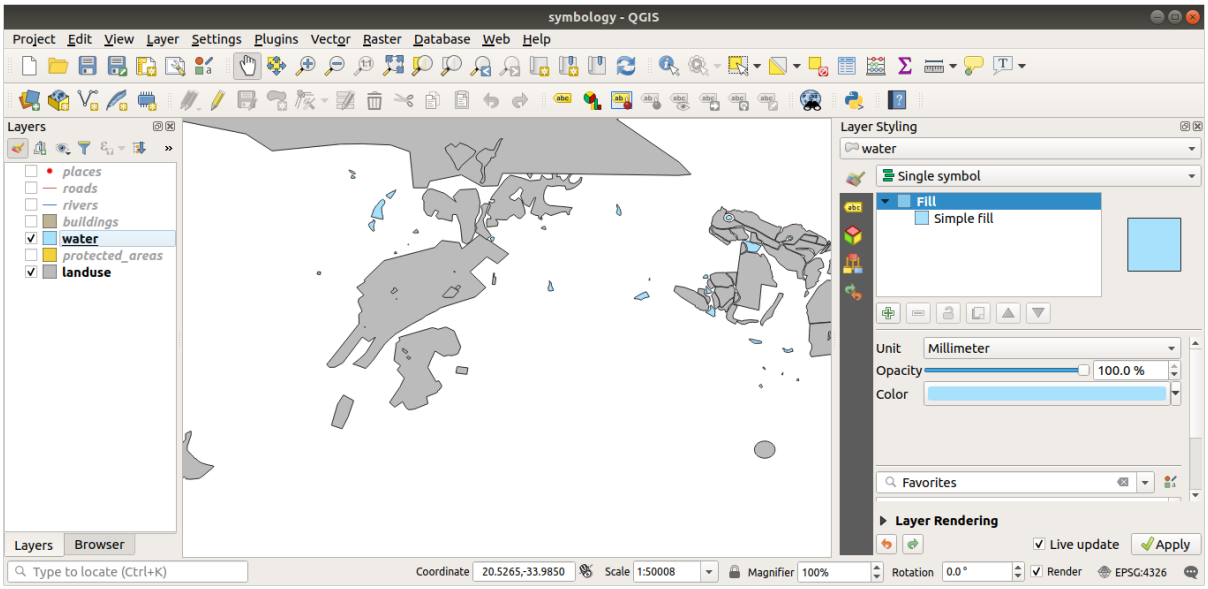
4. Click the color select button next to the *Color* label. A standard color dialog will appear.
5. Choose a gray color and click *OK*.
6. Click *OK* again in the *Layer Properties* window, and you will see the color change being applied to the layer.

2.4.2 ??? Try Yourself:

Change the color of the `water` layer to light blue. Try to use the *Layer Styling* panel instead of the *Layer Properties* menu.

Solution

- Verify that the colors are changing as you expect them to change.
- It is enough to select the `water` layer in the legend and then click on the Open the Layer Styling panel button. Change the color to one that fits the water layer.



If you want to work on only one layer at a time and don't want the other layers to distract you, you can hide a layer by clicking in the checkbox next to its name in the layers list. If the box is blank, then the layer is hidden.

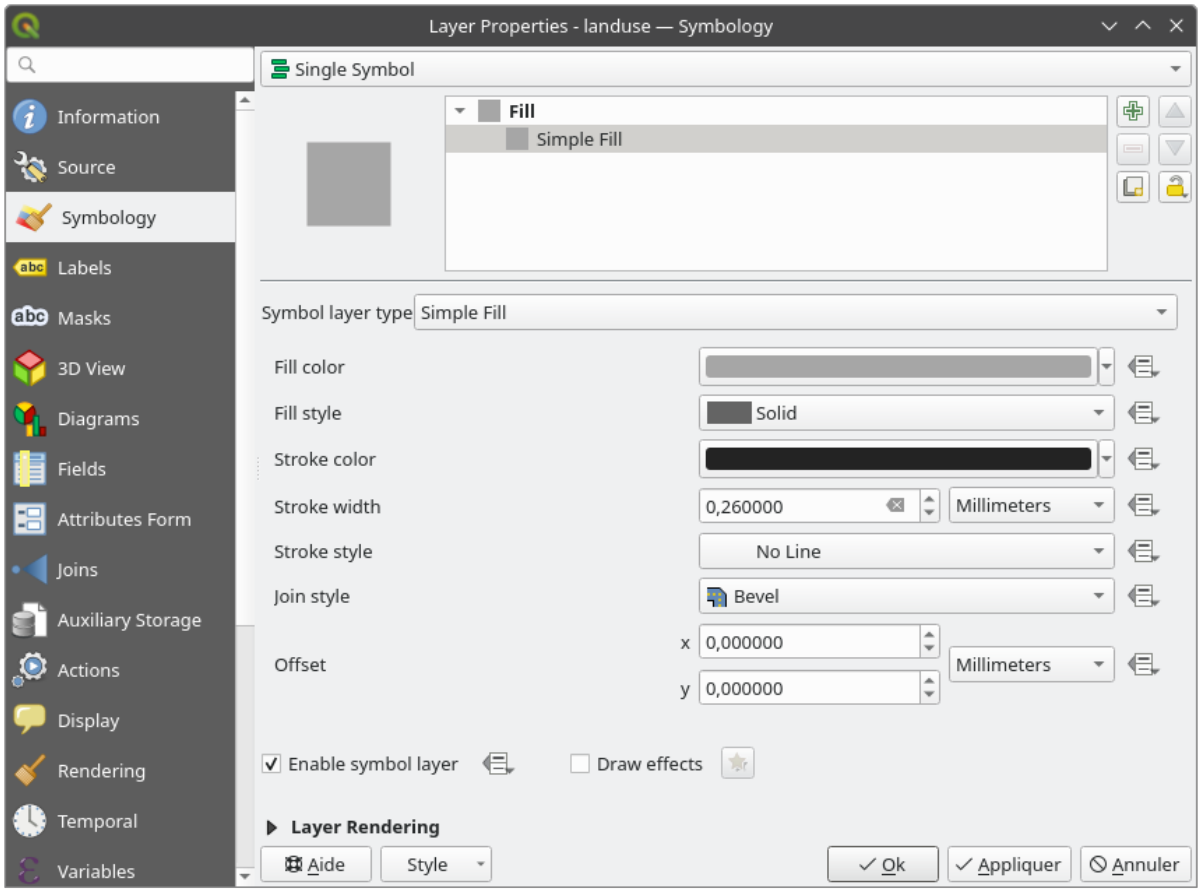
2.4.3 ??? Follow Along: Changing Symbol Structure

This is good stuff so far, but there's more to a layer's symbology than just its color. Next we want to eliminate the lines between the different land use areas so as to make the map less visually cluttered.

1. Open the *Layer Properties* window for the `landuse` layer.

   Under the *Symbology* tab, you will see the same kind of dialog as before. This time, however, you're doing more than just quickly changing the color.

2. In the symbol layers tree, expand the *Fill* dropdown and select the *Simple fill* option.
3. Click on the *Stroke style* dropdown. At the moment, it should be showing a short line and the words *Solid Line*.
4. Change this to *No Line*.



5. Click *OK*.

Now the `landuse` layer won't have any lines between areas.

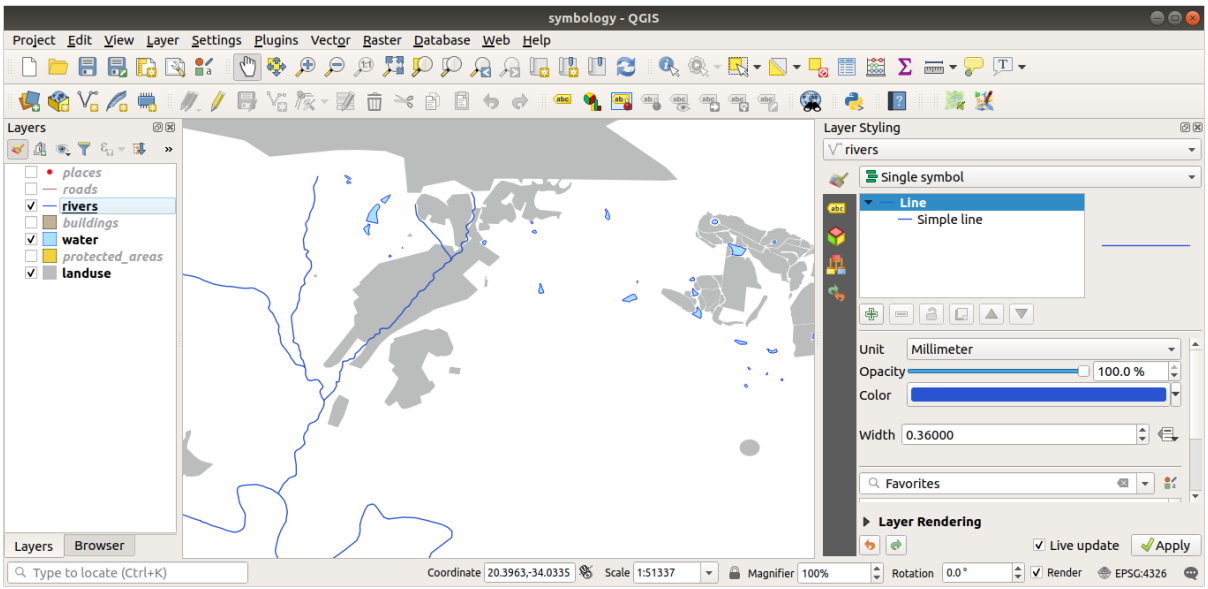
2.4.4 ??? Try Yourself:

- Change the `water` layer's symbology again so that it has a darker blue outline.
- Change the `rivers` layer's symbology to a sensible representation of waterways.

Remember: you can use the Open the Layer Styling panel button and see all the changes instantly. That panel also allows you to undo individual changes while symbolizing a layer.

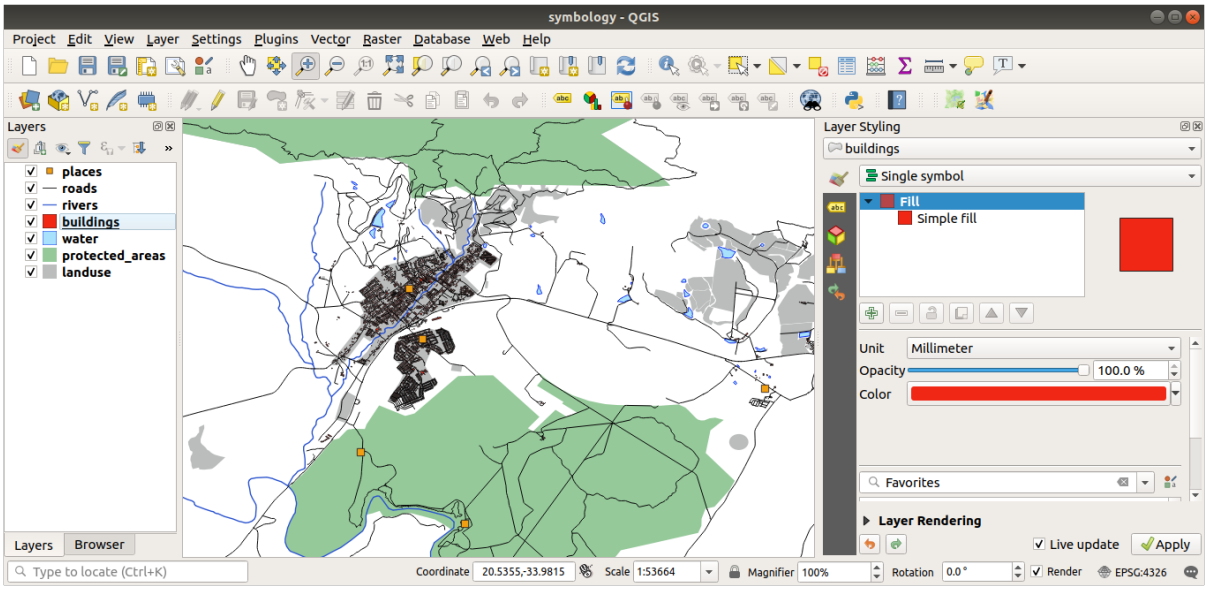
Answer

Your map should now look like this:



If you are a Beginner-level user, you may stop here.

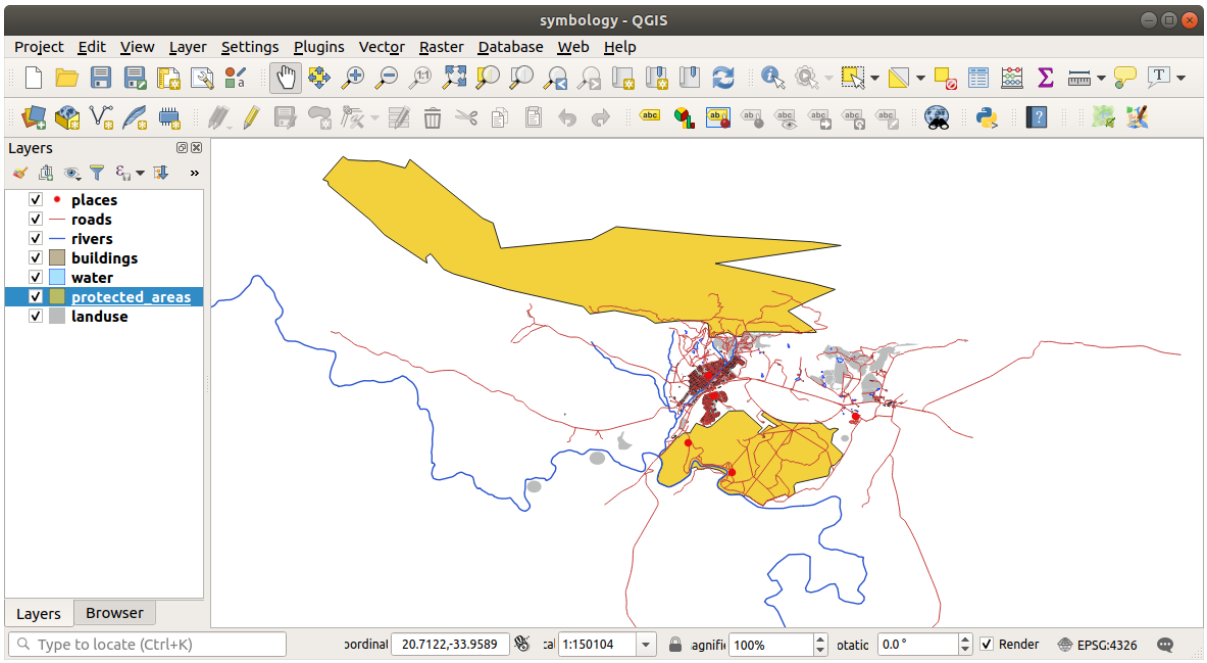
- Use the method above to change the colors and styles for all the remaining layers.
- Try using natural colors for the objects. For example, a road should not be red or blue, but can be gray or black.
- Also feel free to experiment with different *Fill style* and *Stroke style* settings for the polygons.



2.4.5 ??? Follow Along: Scale-Based Visibility

Sometimes you will find that a layer is not suitable for a given scale. For example, a dataset of all the continents may have low detail, and not be very accurate at street level. When that happens, you want to be able to hide the dataset at inappropriate scales.

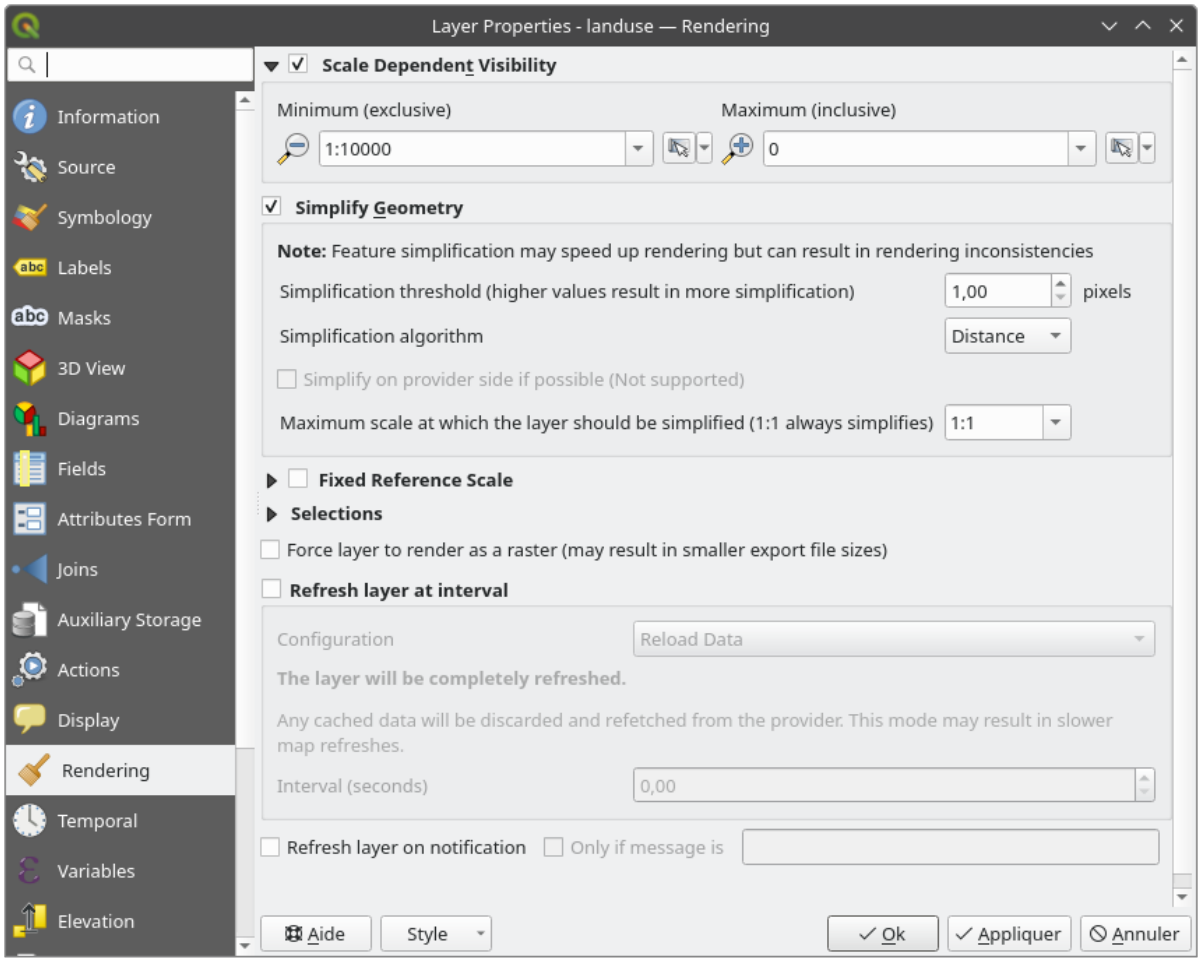
In our case, we may decide to hide the buildings from view at small scales. This map, for example…



… is not very useful. The buildings are hard to distinguish at that scale.

To enable scale-based rendering:

1. Open the *Layer Properties* dialog for the `buildings` layer.
2. Activate the *Rendering* tab.
3. Enable scale-based rendering by clicking on the checkbox labeled *Scale dependent visibility*:
4. Change the *Minimum* value to `1:10000`.



5. Click *OK*.

Test the effects of this by zooming in and out in your map, noting when the `buildings` layer disappears and reappears.

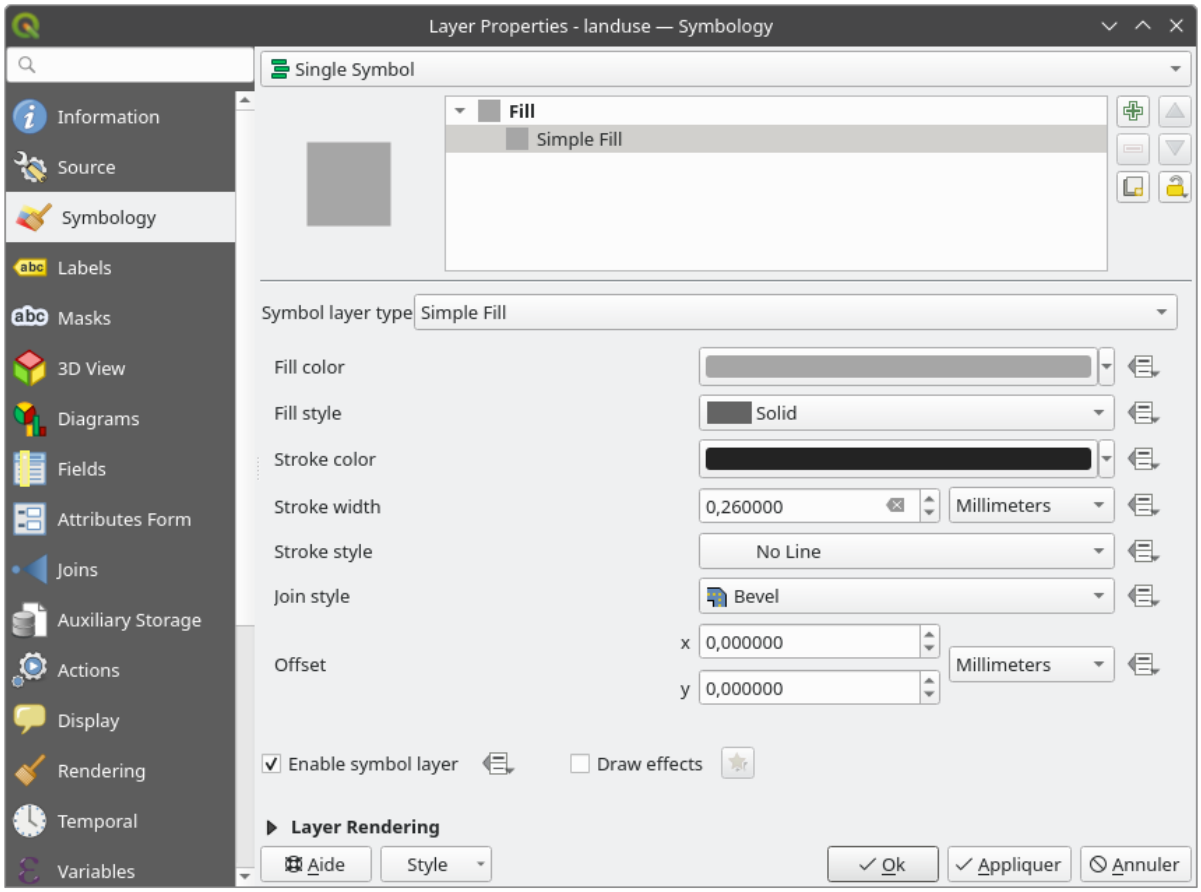
Nota: You can use your mouse wheel to zoom in increments. Alternatively, use the zoom tools to zoom to a window:

2.4.6 ??? Follow Along: Adding Symbol Layers

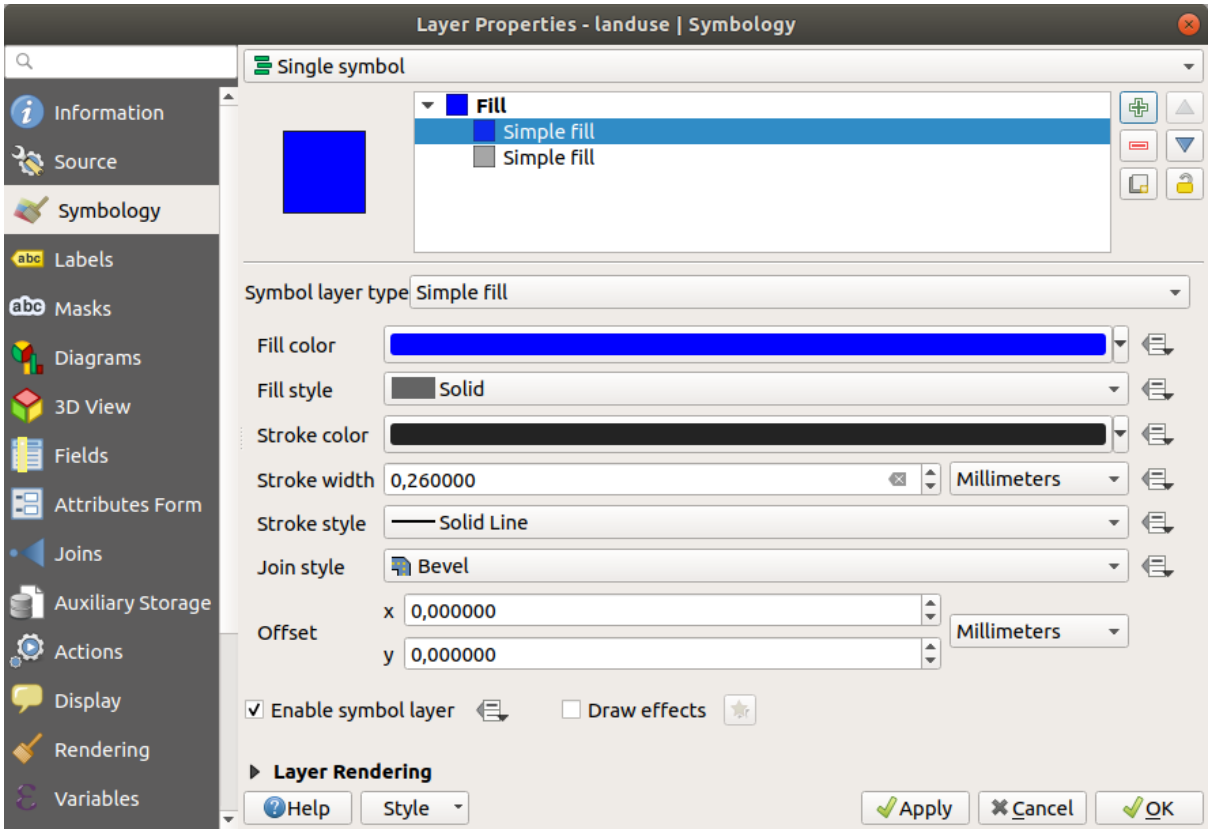
Now that you know how to change simple symbology for layers, the next step is to create more complex symbology. QGIS allows you to do this using symbol layers.

1. Go back to the `landuse` layer's symbol properties panel (by clicking *Simple fill* in the symbol layers tree).

   In this example, the current symbol has no outline (i.e., it uses the *No Line* border style).



2. Select the *Fill* level in the tree and click the Add symbol layer button. The dialog will change to look something like this, with a new symbol layer added:



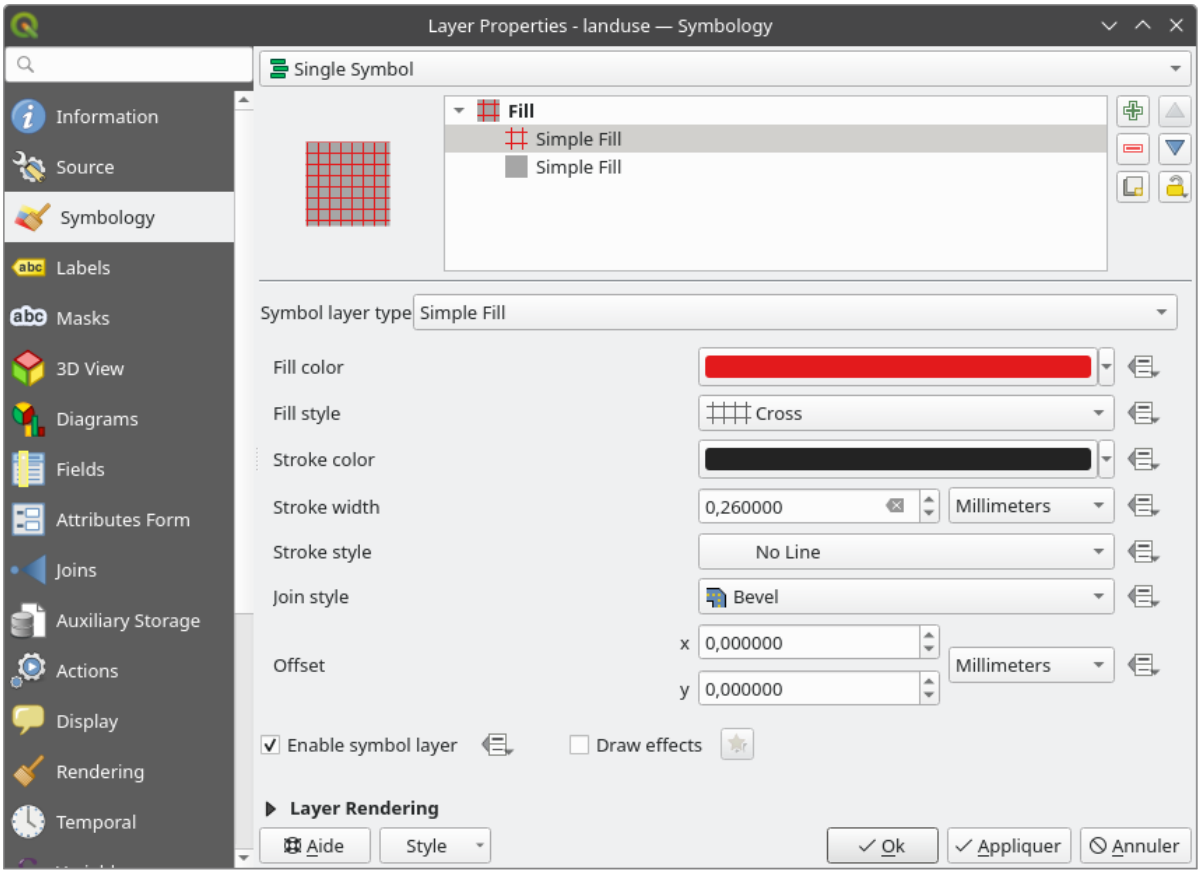
It may appear somewhat different in color, for example, but you're going to change that anyway.

Now there's a second symbol layer. Being a solid color, it will of course completely hide the previous kind of symbol. Plus, it has a *Solid Line* border style, which we don't want. Clearly this symbol has to be changed.

Nota: It's important not to get confused between a map layer and a symbol layer. A map layer is a vector (or raster) that has been loaded into the map. A symbol layer is part of the symbol used to represent a map layer. This course will usually refer to a map layer as just a layer, but a symbol layer will always be called a symbol layer, to prevent confusion.

With the new *Simple Fill* symbol layer selected:

1. Set the border style to *No Line*, as before.
2. Change the fill style to something other than *Solid* or *No brush*. For example:



3. Click *OK*.

Now you can see your results and tweak them as needed. You can even add multiple extra symbol layers and create a kind of texture for your layer that way.

It's fun! But it probably has too many colors to use in a real map…

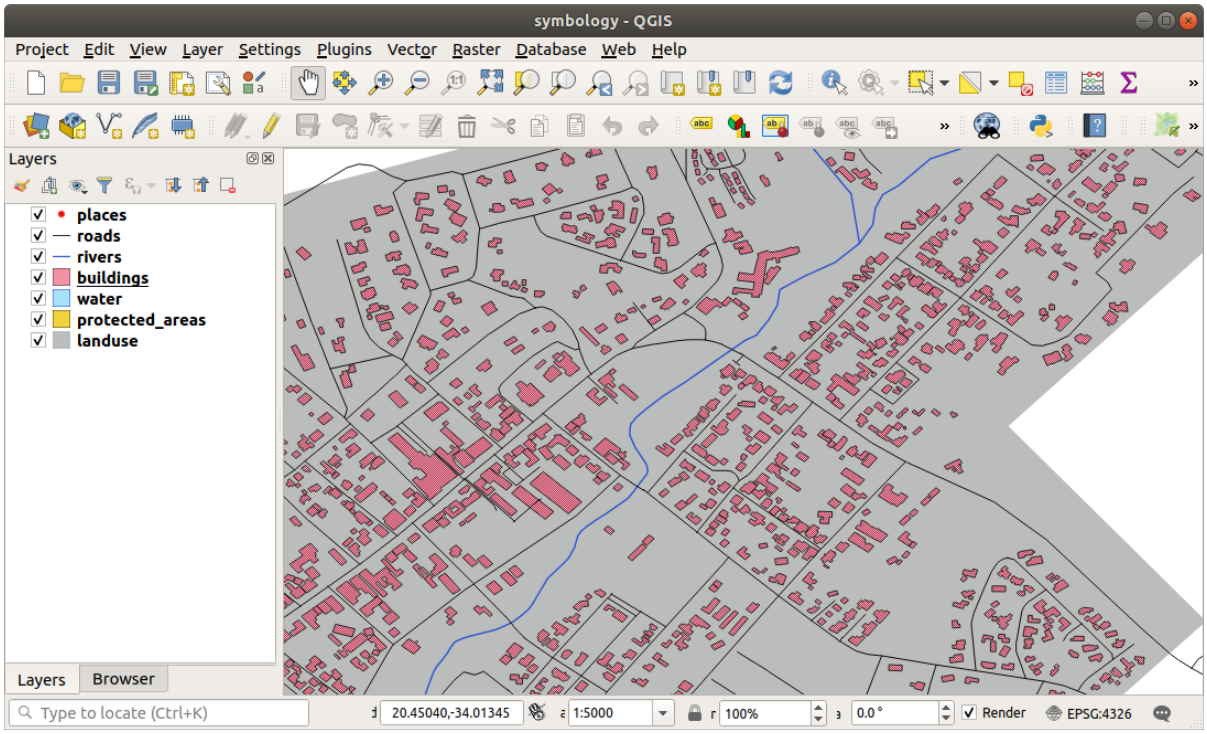
2.4.7 ??? Try Yourself:

Remembering to zoom in if necessary, create a simple, but not distracting texture for the `buildings` layer using the methods above.

Answer

Customize your `buildings` layer as you like, but remember that it has to be easy to tell different layers apart on the map.

Aqui tem um exemplo:



2.4.8 ??? Follow Along: Ordering Symbol Levels

When symbol layers are rendered, they are also rendered in a sequence, similar to the way the different map layers are rendered. This means that in some cases, having many symbol layers in one symbol can cause unexpected results.

1. Give the `roads` layer an extra symbol layer (using the method for adding symbol layers demonstrated above).
2. Give the base line a *Stroke width* of `1.5` and a black color.
3. Give the new, uppermost layer a thickness of `0.8` and a white color.

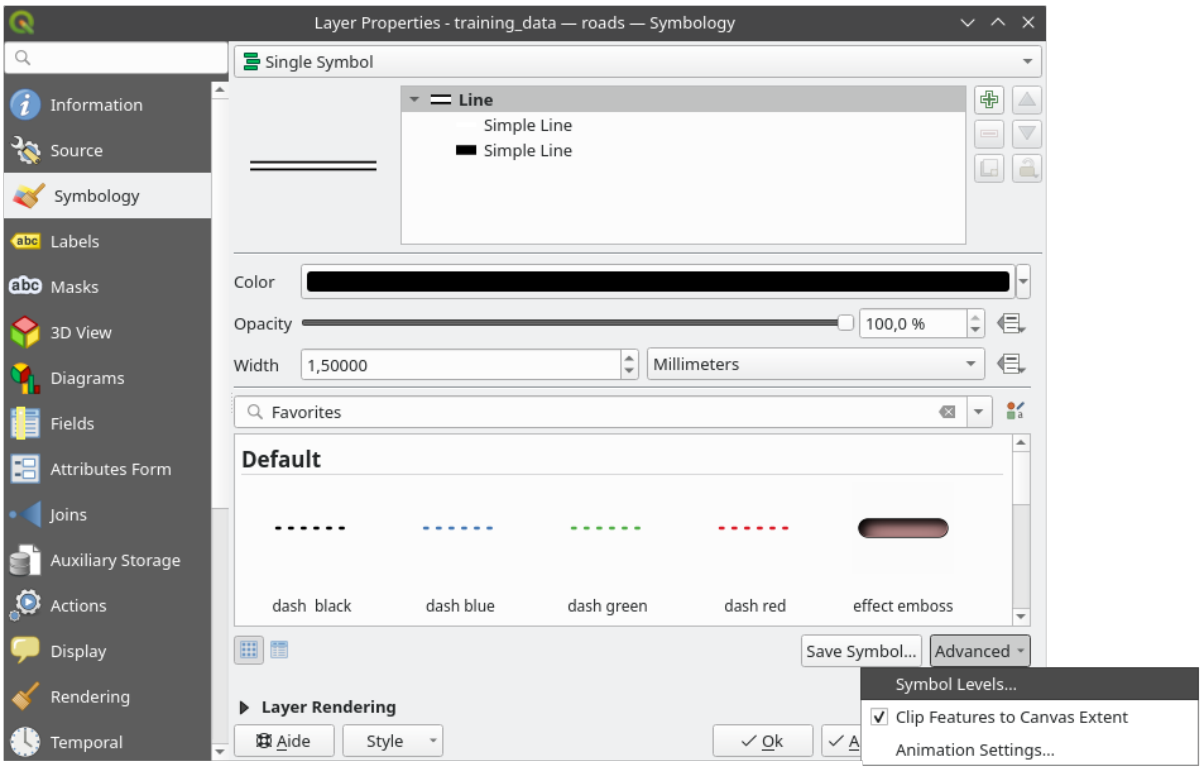
You'll notice that this happens:

Well, roads have now a *street* like symbology, but you see that lines are overlapping each others at each cross. That's not what we want at all!

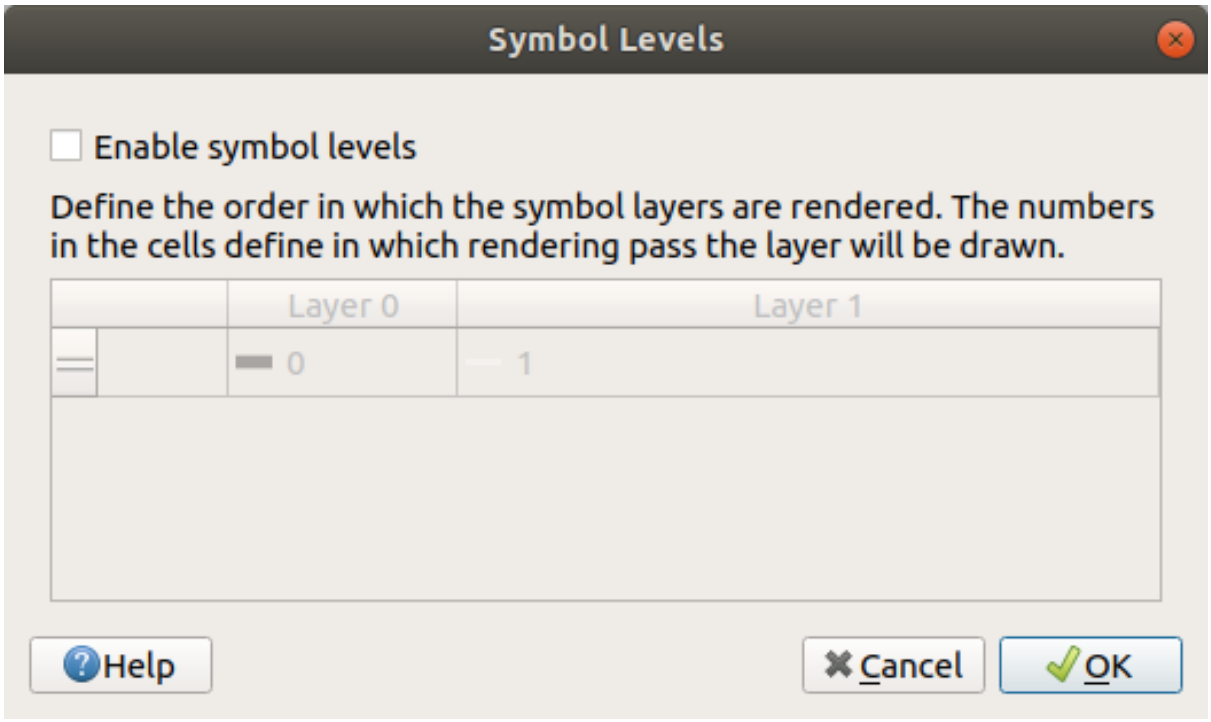
To prevent this from happening, you can sort the symbol levels and thereby control the order in which the different symbol layers are rendered.

To change the order of the symbol layers:

1. Select the topmost *Line* layer in the symbol layers tree.
2. Click *Advanced* ► *Symbol levels...* in the bottom right-hand corner of the window.

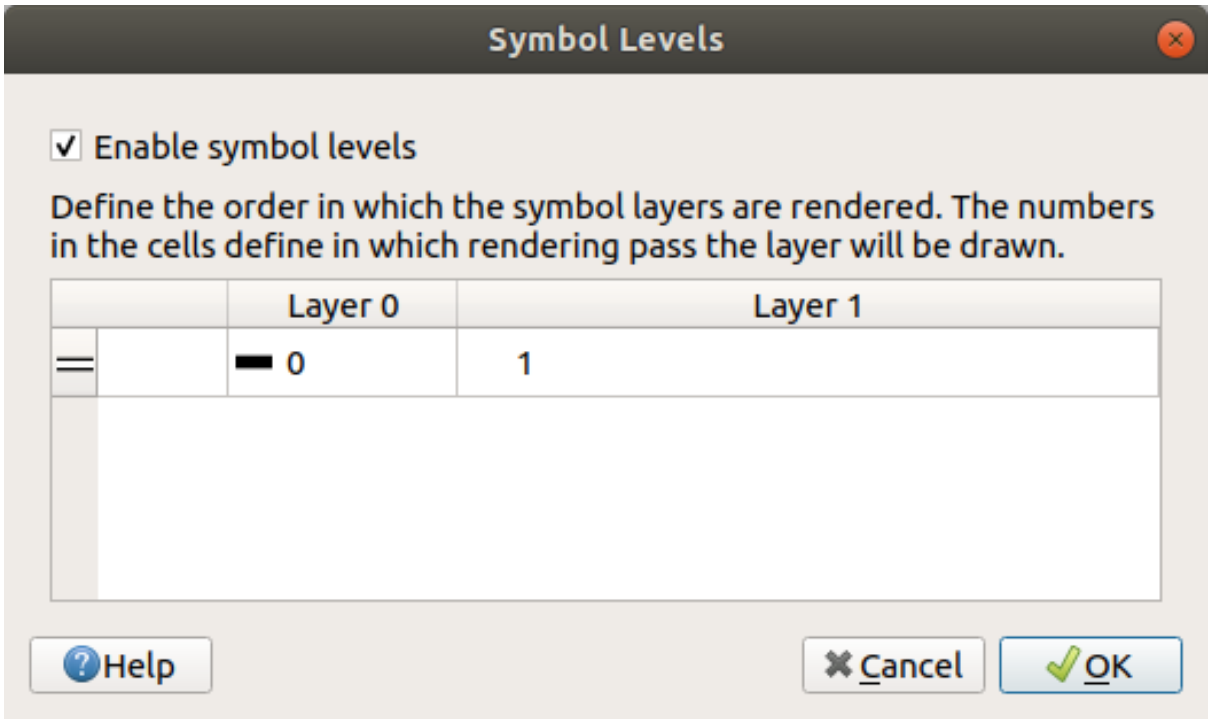


This will open a dialog like this:



3. Check *Enable symbol levels*. You can then set the layer order of each symbol by entering the corresponding level number. 0 is the bottom layer.

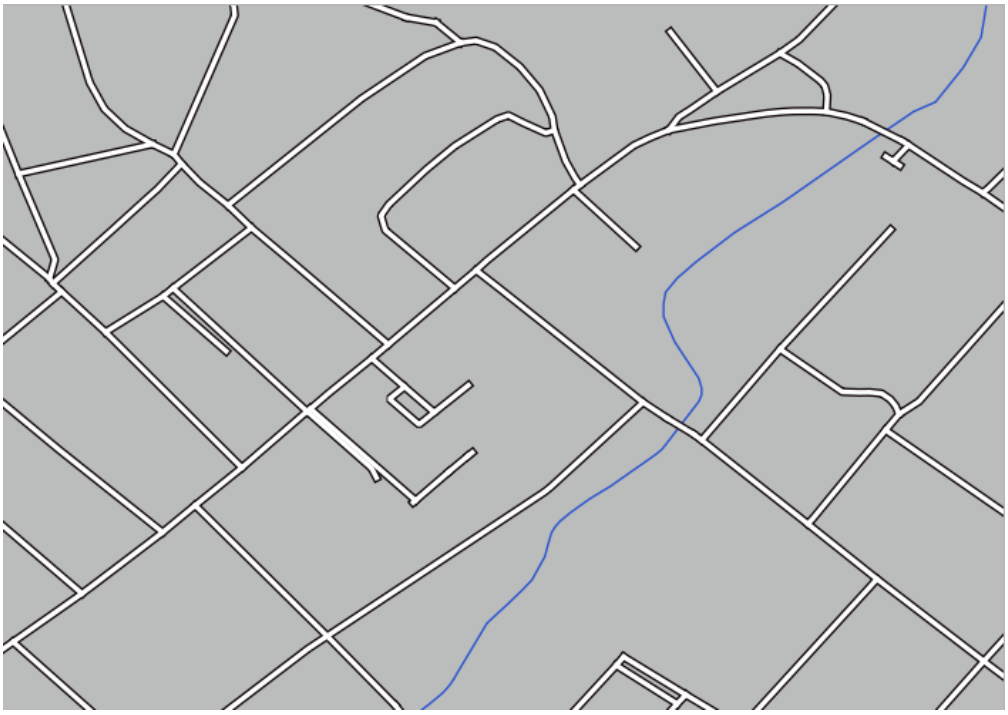
   In our case, we just want to activate the option, like this:



   This will render the white line above the thick black line borders:

4. Click *OK* twice to return to the map.

   The map will now look like this:



When you're done, remember to save the symbol itself so as not to lose your work if you change the symbol again in the future. You can save your current symbol style by clicking the *Save Style…* button at the bottom of the *Layer Properties* dialog. We will be using the *QGIS QML Style File* format.

Save your style in the `solution/styles/better_roads.qml` folder. You can load a previously saved style at any time by clicking the *Load Style…* button. Before you change a style, keep in mind that any unsaved style you are replacing will be lost.

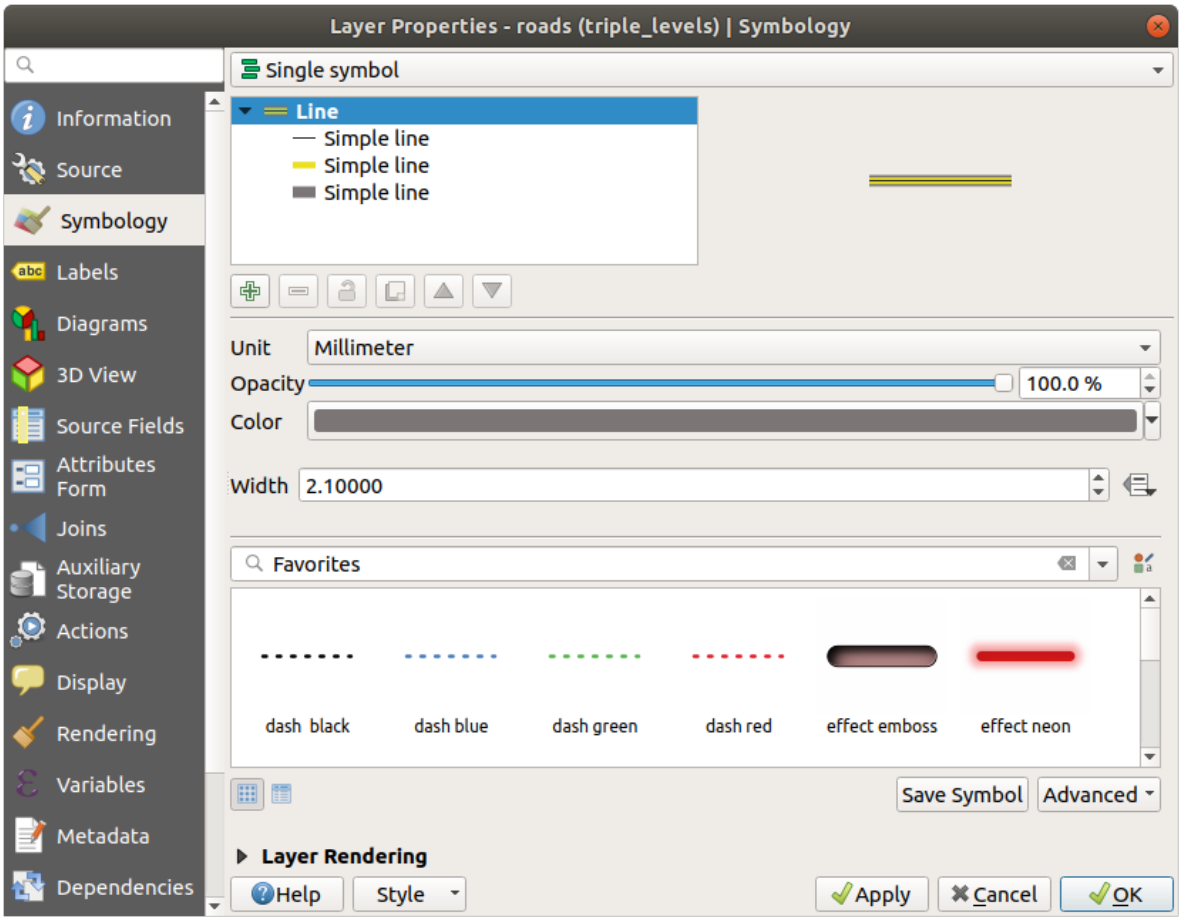
2.4.9 ??? Try Yourself:

Change the appearance of the `roads` layer again.

Make the roads narrow and yellow, with a thin, pale gray outline and a thin black line in the middle. Remember that you may need to change the layer rendering order via the *Advanced* ► *Symbol levels…* dialog.

Answer

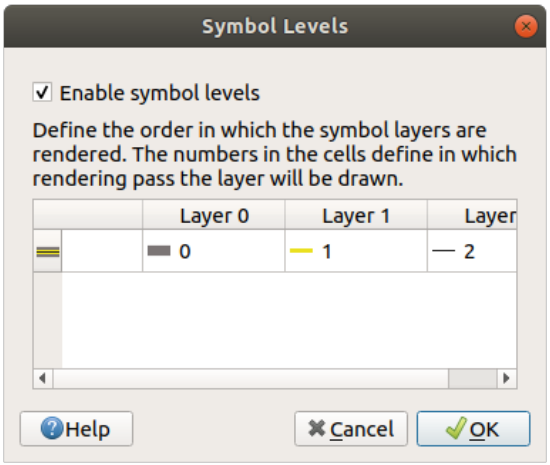
To make the required symbol, you need three symbol layers:



The lowest symbol layer is a broad, solid gray line. On top of it there is a slightly thinner solid yellow line and finally another thinner solid black line.

If your symbol layers resemble the above but you're not getting the result you want:

1. Check that your symbol levels look something like this:

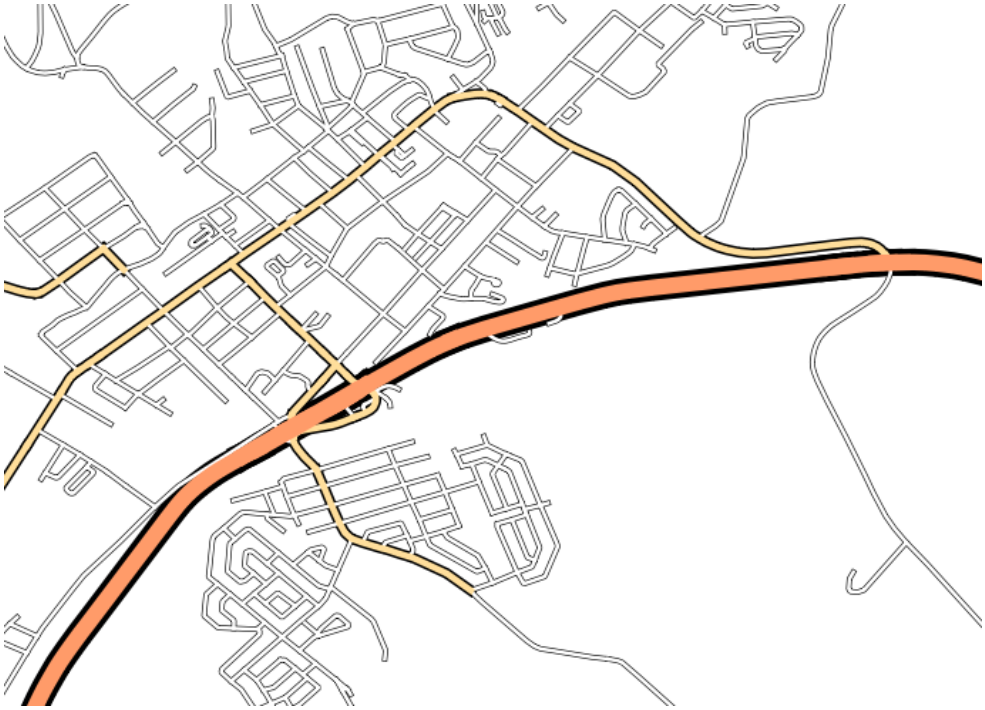


2. Now your map should look like this:

2.4.10 ??? Try Yourself:

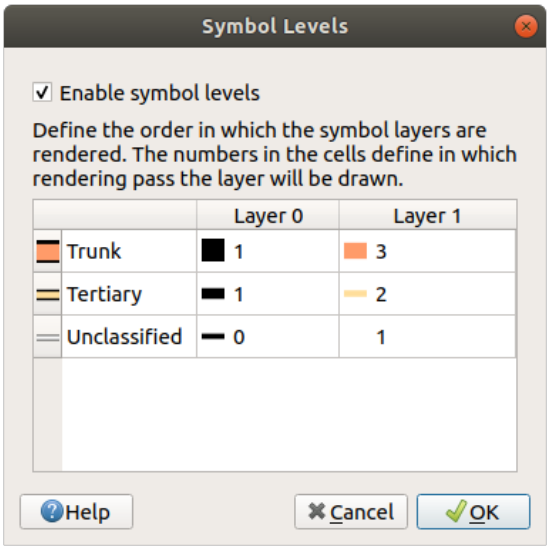
Symbol levels also work for classified layers (i.e., layers having multiple symbols). Since we haven't covered classification yet, you will work with some rudimentary pre-classified data.

1. Create a new project and add only the `roads` dataset.
2. Apply the style file `advanced_levels_demo.qml` provided in `exercise_data/styles` to the layer. This can be done through the *Style* ► *Load Style...* combobox at the bottom of the *Layer Properties* dialog.
3. Zoom in to the Swellendam area.
4. Using symbol layers, ensure that the outlines of layers flow into one another as per the image below:



Answer

1. Adjust your symbol levels to these values:



1. Experiment with different values to get different results.
2. Open your original map again before continuing with the next exercise.

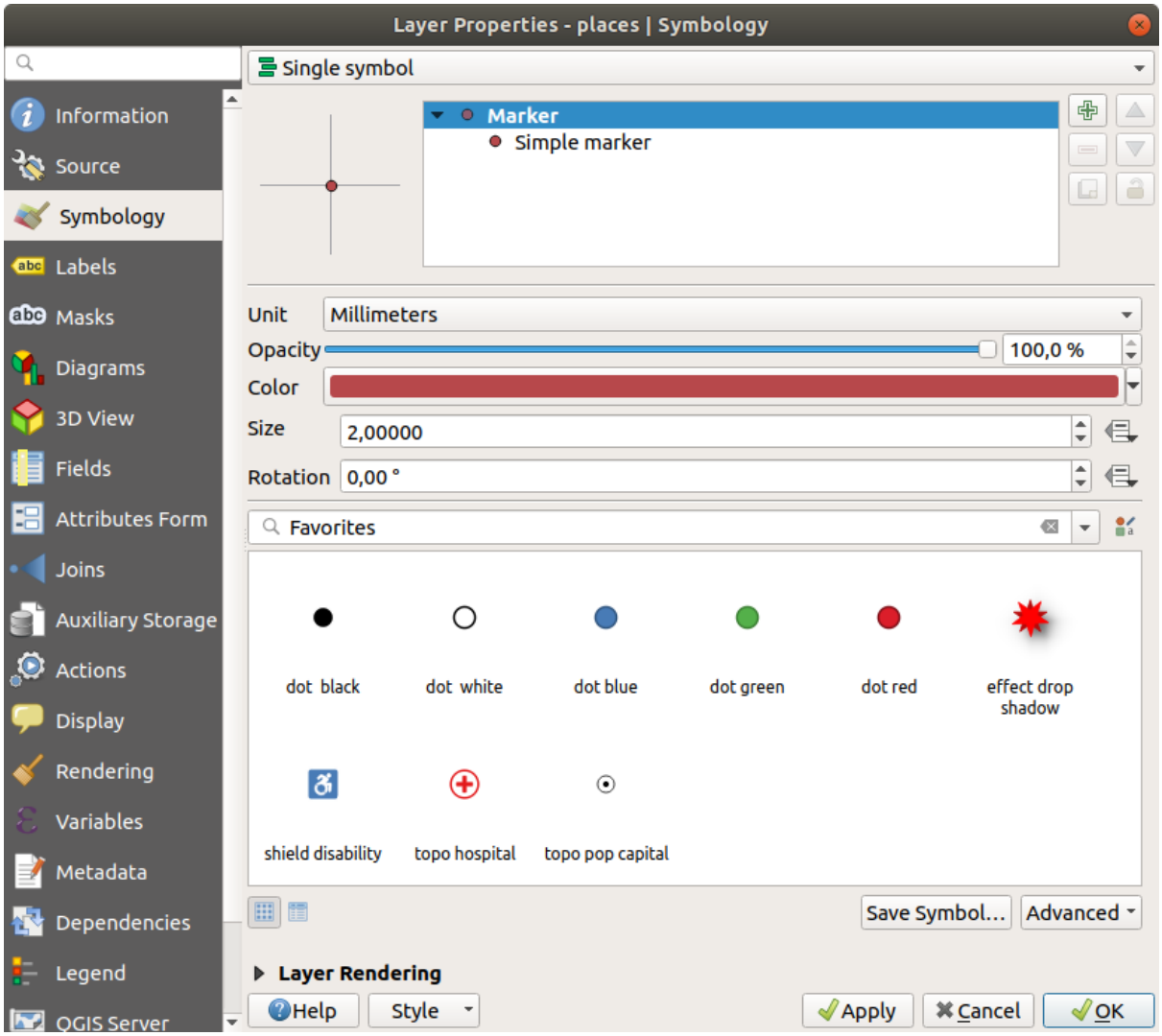
2.4.11 ??? Follow Along: Symbol layer types

In addition to setting fill colors and using predefined patterns, you can use different symbol layer types entirely. The only type we've been using up to now was the *Simple Fill* type. The more advanced symbol layer types allow you to customize your symbols even further.

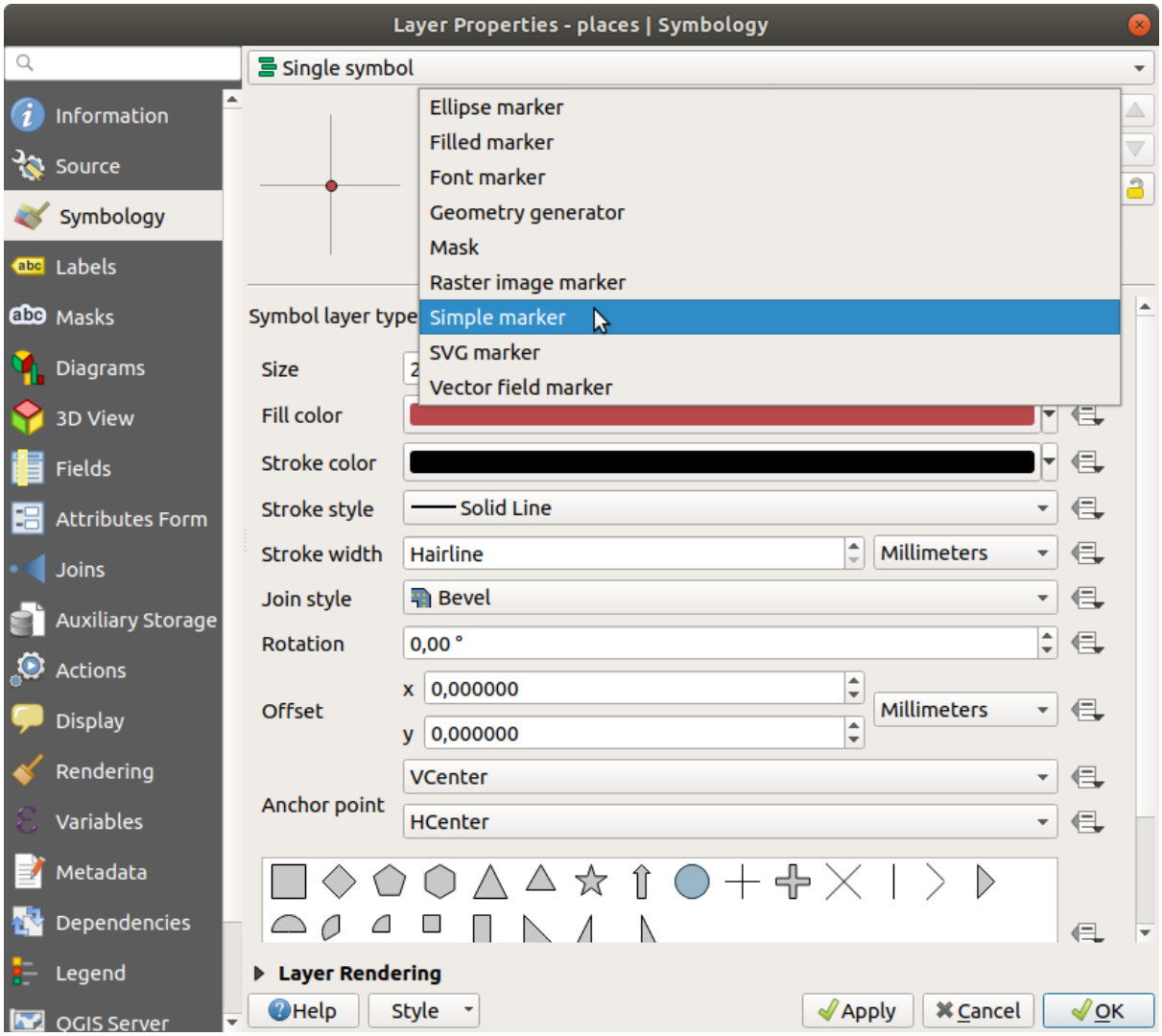
Each type of vector (point, line and polygon) has its own set of symbol layer types. First we will look at the types available for points.

Point Symbol Layer Types

1. Uncheck all the layers except for `places`.
2. Change the symbol properties for the `places` layer:



3. You can access the various symbol layer types by selecting the *Simple marker* layer in the symbol layers tree, then click the *Symbol layer type* dropdown:

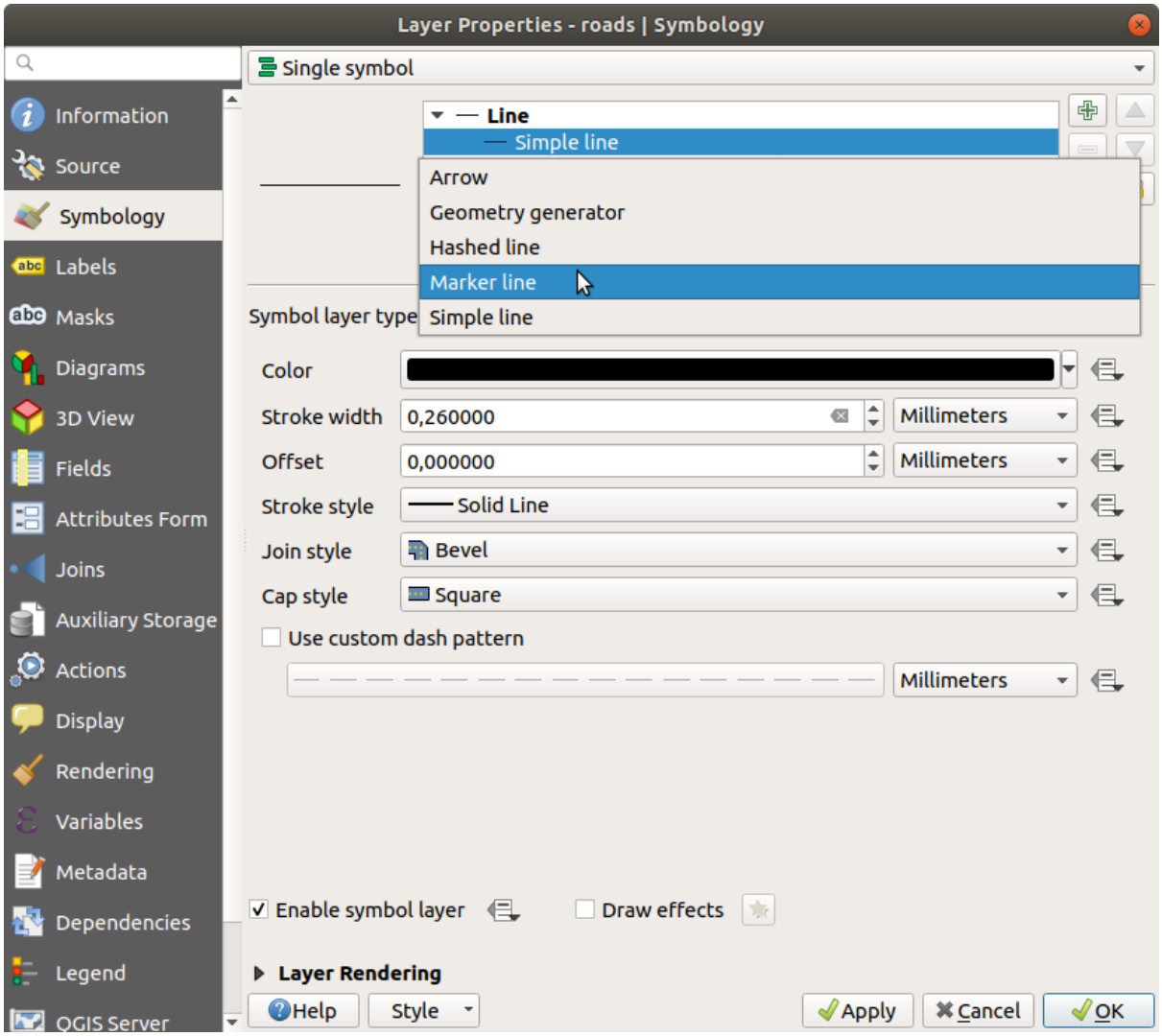


4. Investigate the various options available to you, and choose a symbol with styling you think is appropriate.
5. If in doubt, use a round *Simple marker* with a white border and pale green fill, with a *Size* of `3.00` and a *Stroke width* of `0.5`.

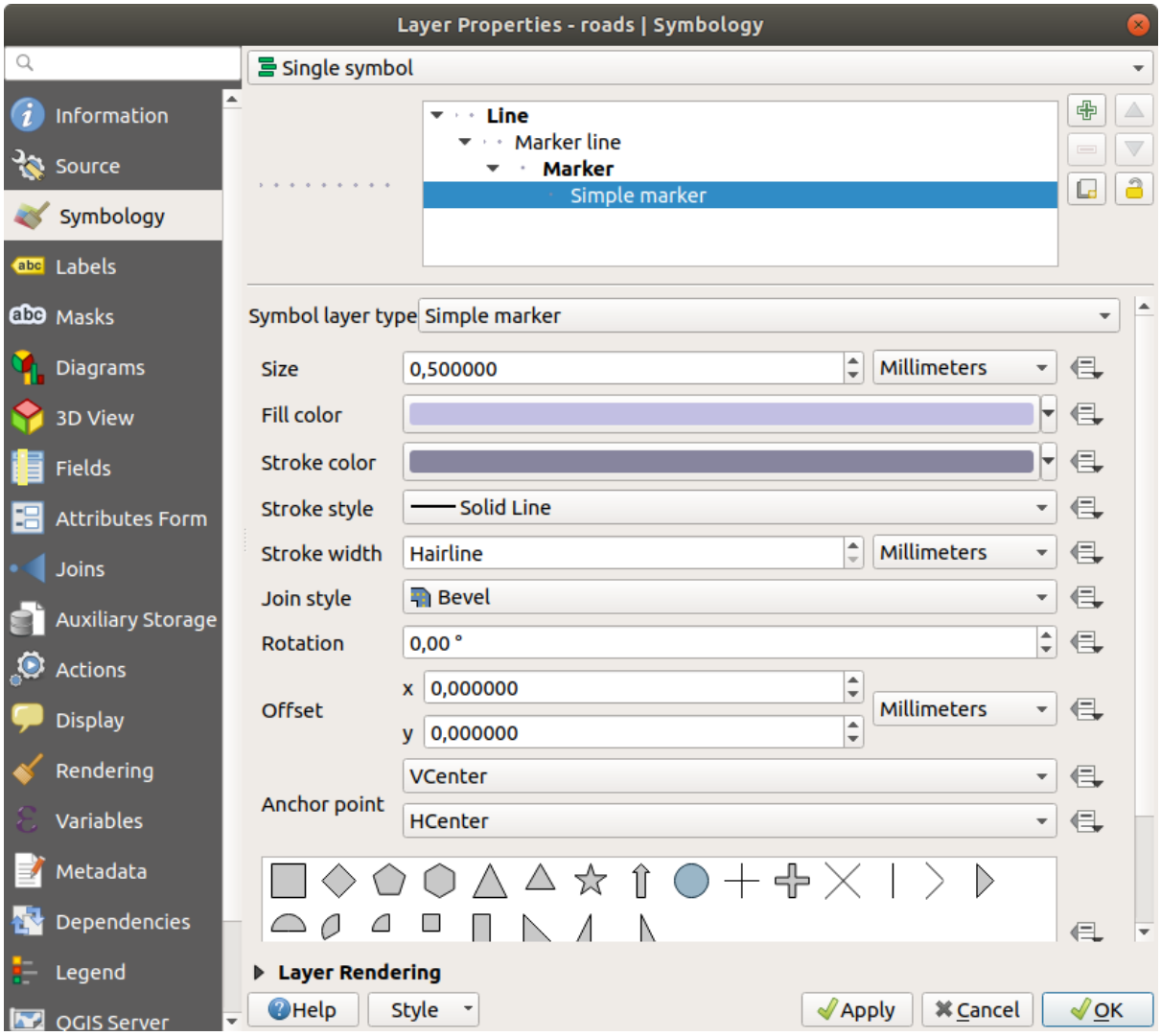
Line Symbol Layer Types

To see the various options available for line data:

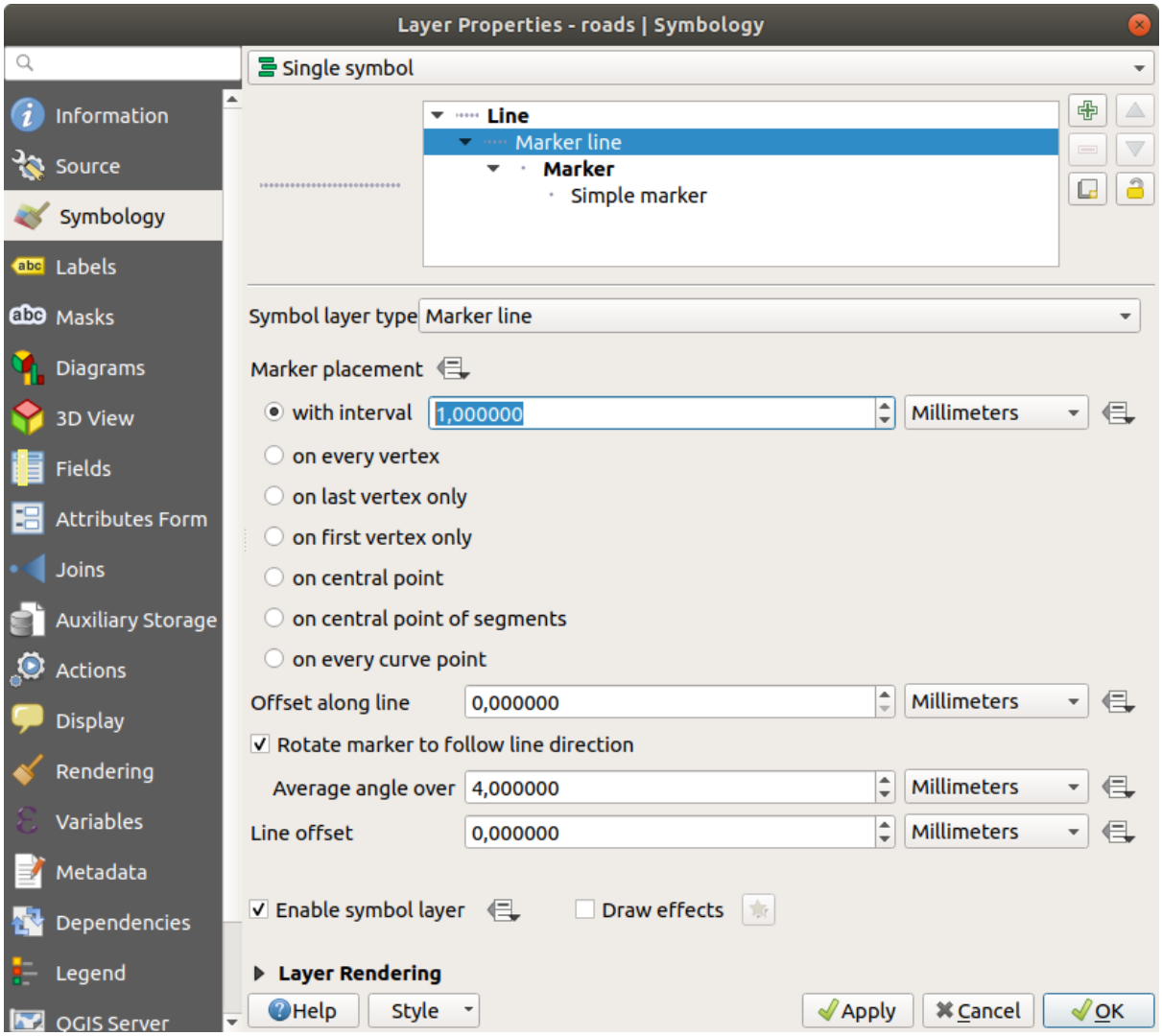
1. Change the *Symbol layer type* for the `roads` layer's topmost symbol layer to *Marker line*:



2. Select the *Simple marker* layer in the symbol layers tree. Change the symbol properties to match this dialog:



3. Select the *Marker line* layer and change the interval to `1.00`:



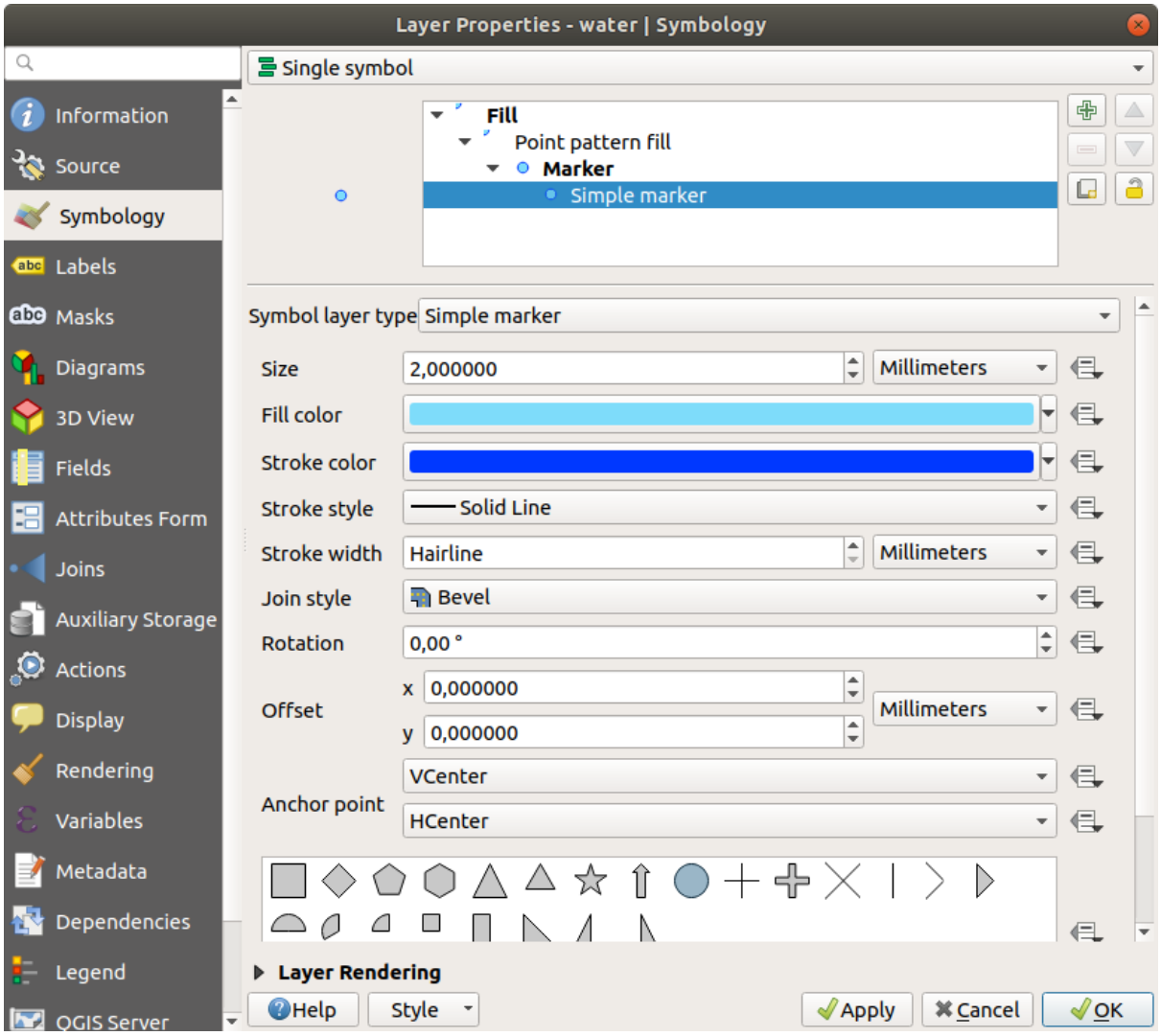
4. Ensure that the symbol levels are correct (via the *Advanced* ► *Symbol levels* dialog we used earlier) before applying the style.

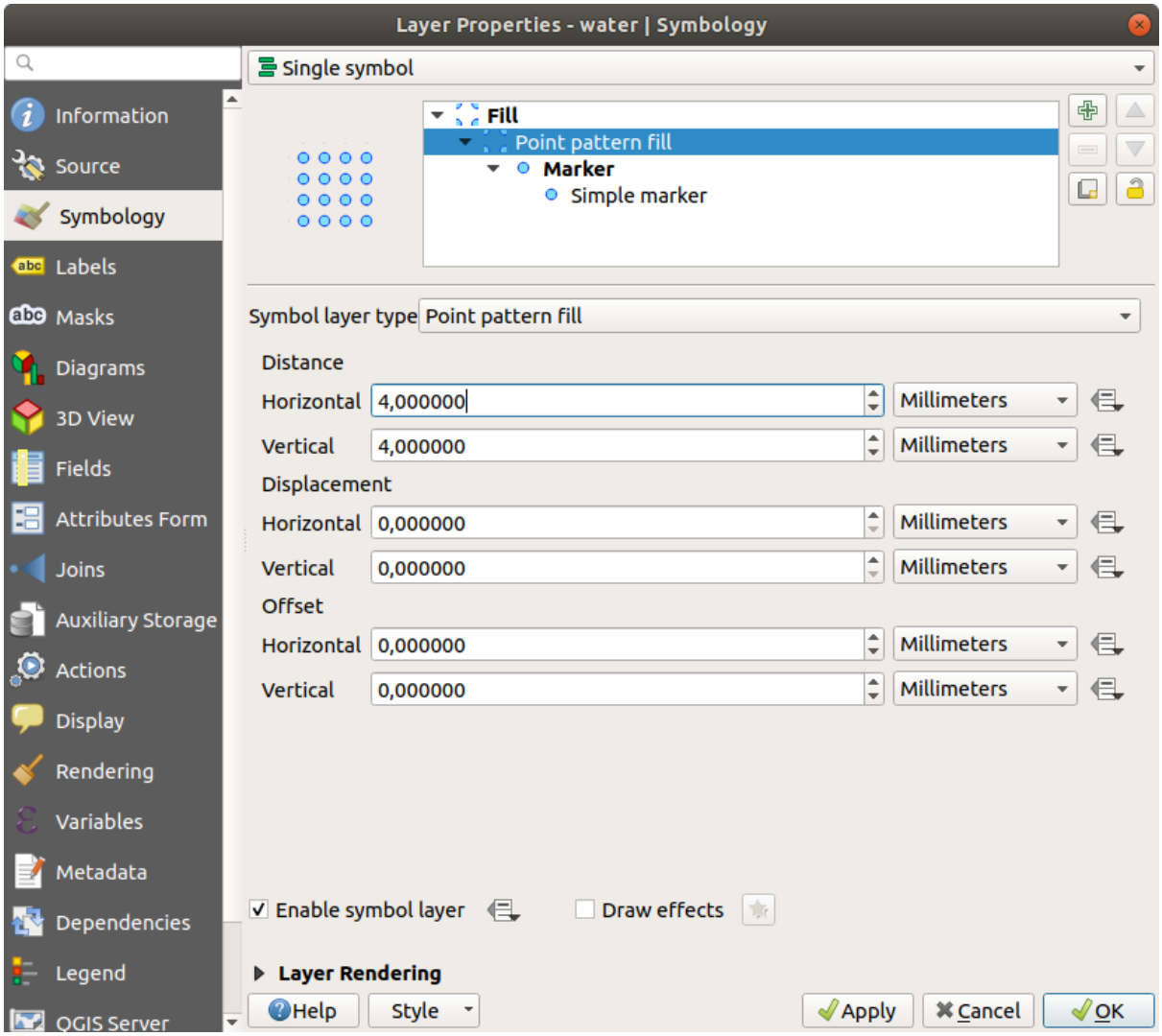
Once you have applied the style, take a look at its results on the map. As you can see, these symbols change direction along with the road but don't always bend along with it. This is useful for some purposes, but not for others. If you prefer, you can change the symbol layer in question back to the way it was before.

Polygon Symbol Layer Types

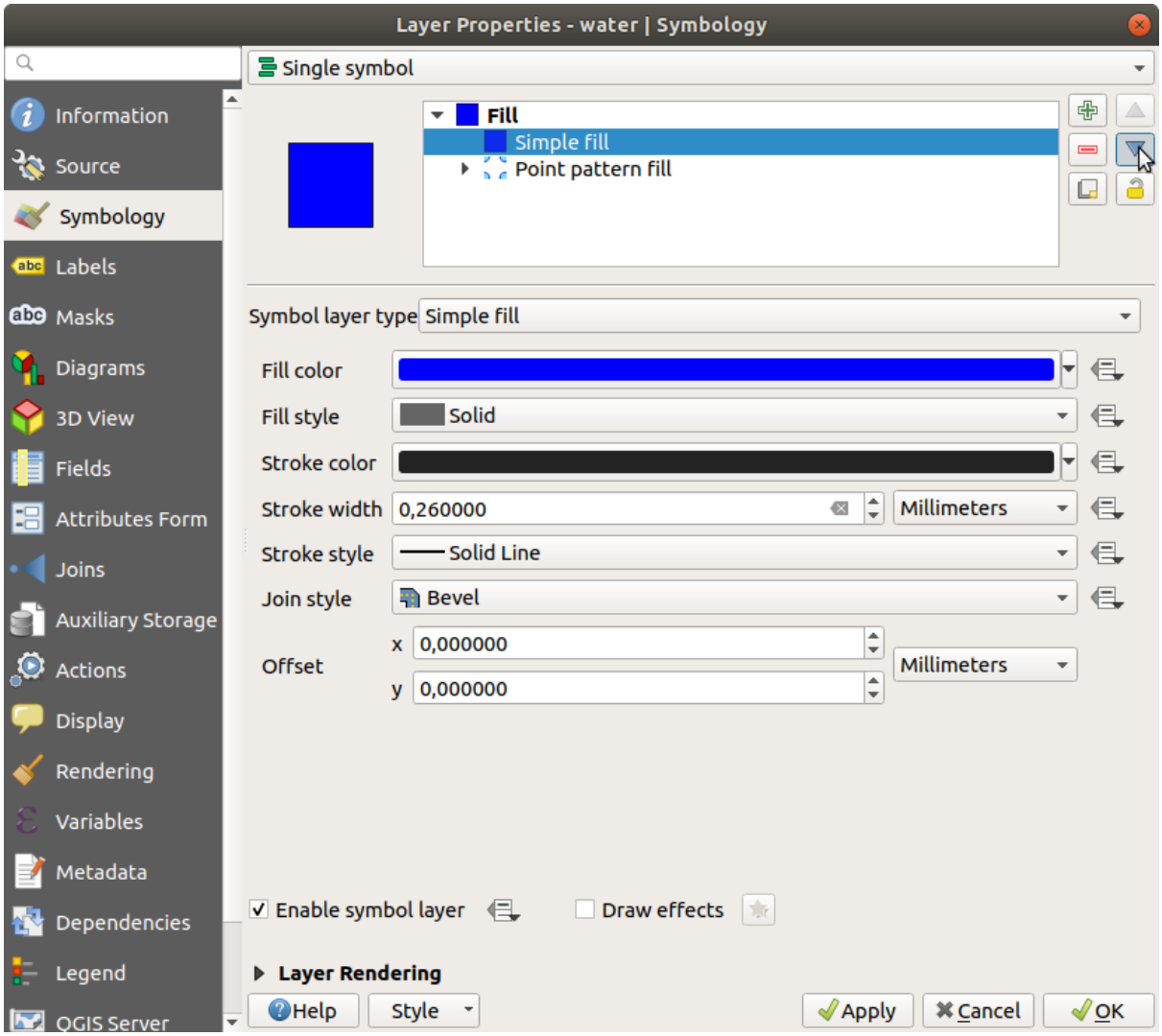
To see the various options available for polygon data:

1. Change the *Symbol layer type* for the `water` layer, as before for the other layers.
2. Investigate what the different options on the list can do.
3. Choose one of them that you find suitable.
4. If in doubt, use the *Point pattern fill* with the following options:


Layer Properties - water | Symbology
Information
Source
Symbology
Labels
Masks
Diagrams
3D View
Fields
Attributes Form
Joins
Auxiliary Storage
Actions
Display
Rendering
Variables
Metadata
Dependencies
Legend
QGIS Server
Single symbol
Fill
Point pattern fill
Marker
Simple marker
Symbol layer type
Simple marker
Size
2,000000
Millimeters
Fill color
Stroke color
Stroke style
Solid Line
Stroke width
Hairline
Millimeters
Join style
Bevel
Rotation
0,00 °
Offset
x
0,000000
y
0,000000
Millimeters
Anchor point
VCenter
HCenter
Layer Rendering
Help
Style
Apply
Cancel
OK



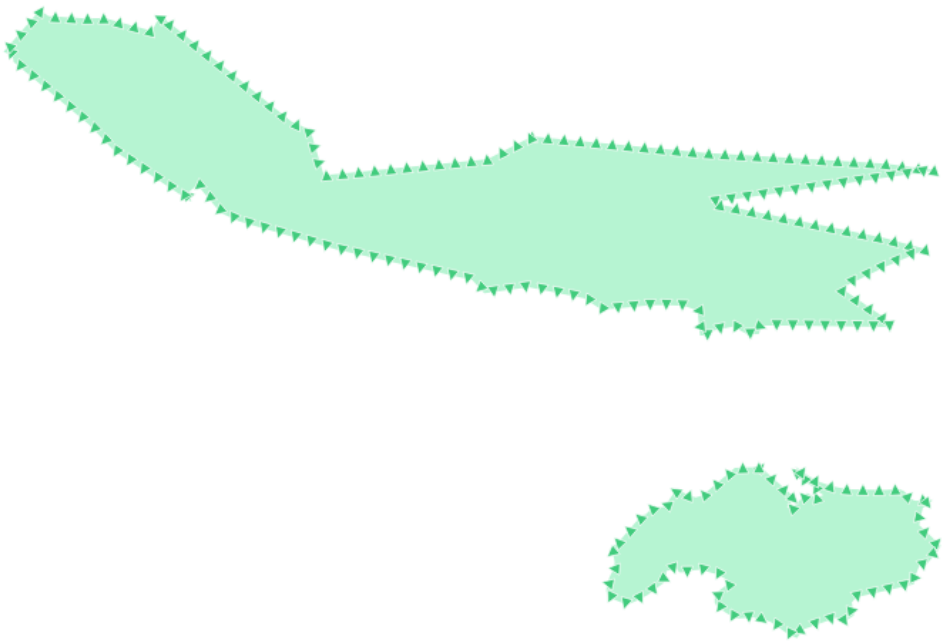
5. Add a new symbol layer with a normal *Simple fill*.
6. Make it the same light blue with a darker blue border.
7. Move it underneath the point pattern symbol layer with the *Move down* button:



As a result, you have a textured symbol for the water layer, with the added benefit that you can change the size, shape and distance of the individual dots that make up the texture.

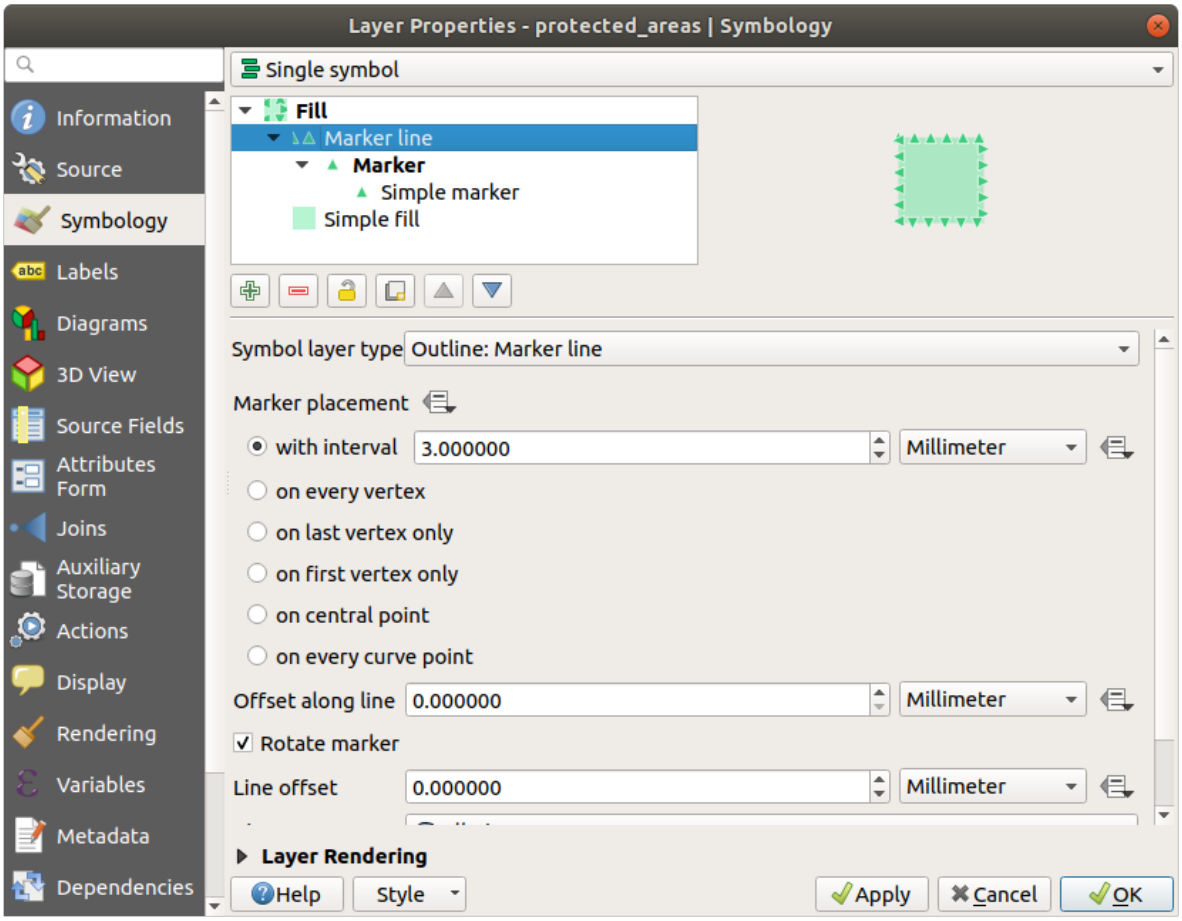
2.4.12 ??? Try Yourself:

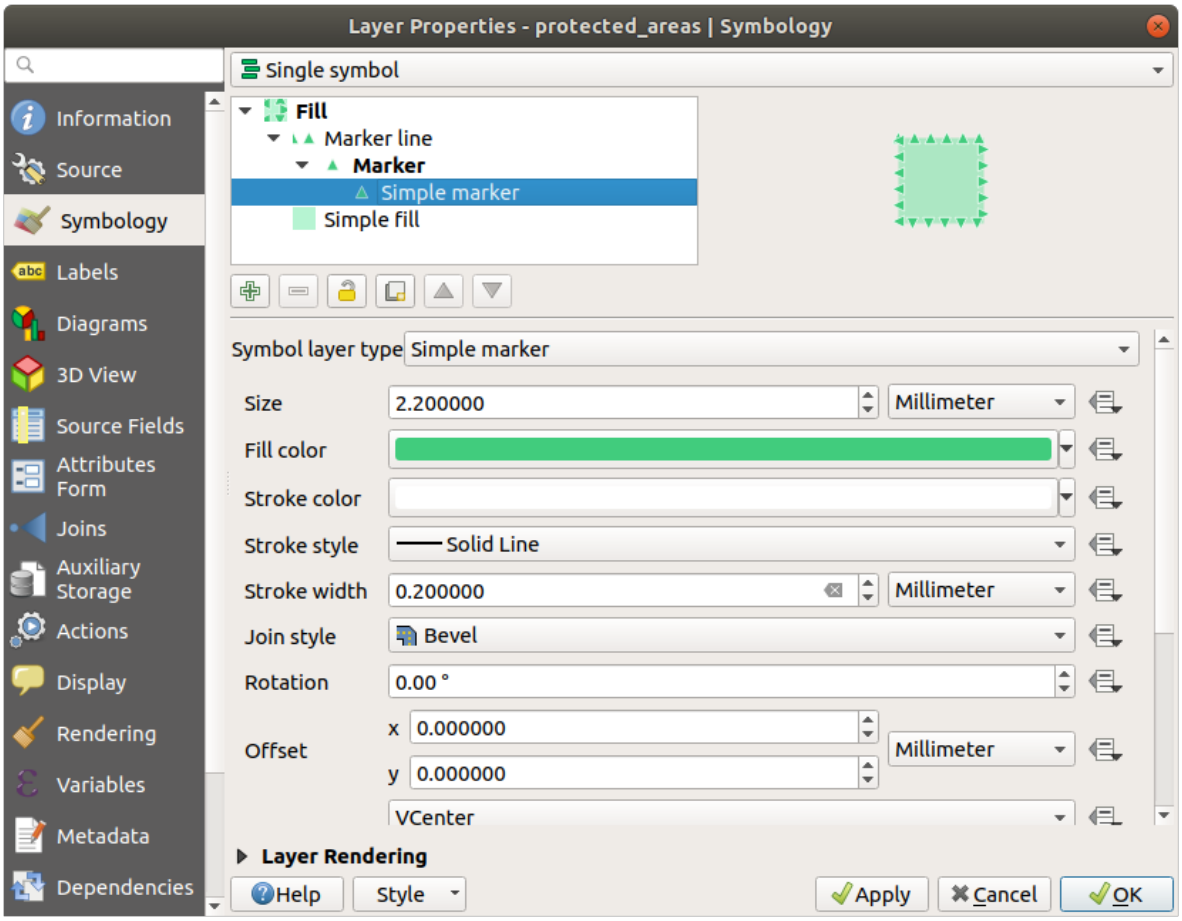
Apply a green transparent fill color to the `protected_areas` layer, and change the outline to look like this:



Answer

Here are examples of the symbol structure:





2.4.13 ??? Follow Along: Geometry generator symbology

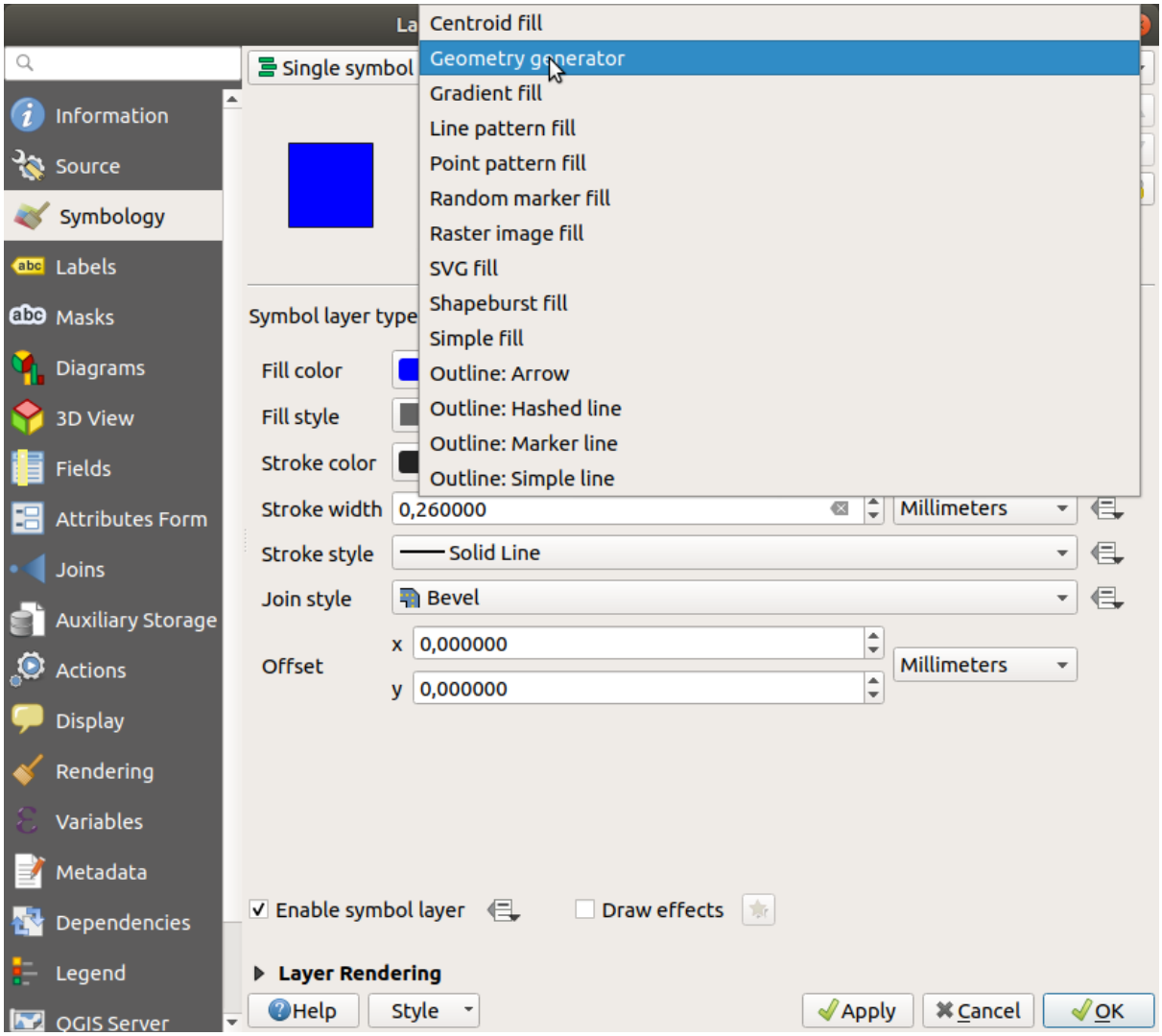
You can use the Geometry generator symbology with all layer types (points, lines and polygons). The resulting symbol depends directly on the layer type.

Very briefly, the Geometry generator symbology allows you to run some spatial operations within the symbology itself. For example you can run a real centroid spatial operation on a polygon layer without creating a point layer.

Moreover, you have all the styling options to change the appearance of the resulting symbol.

Let's give it a try!

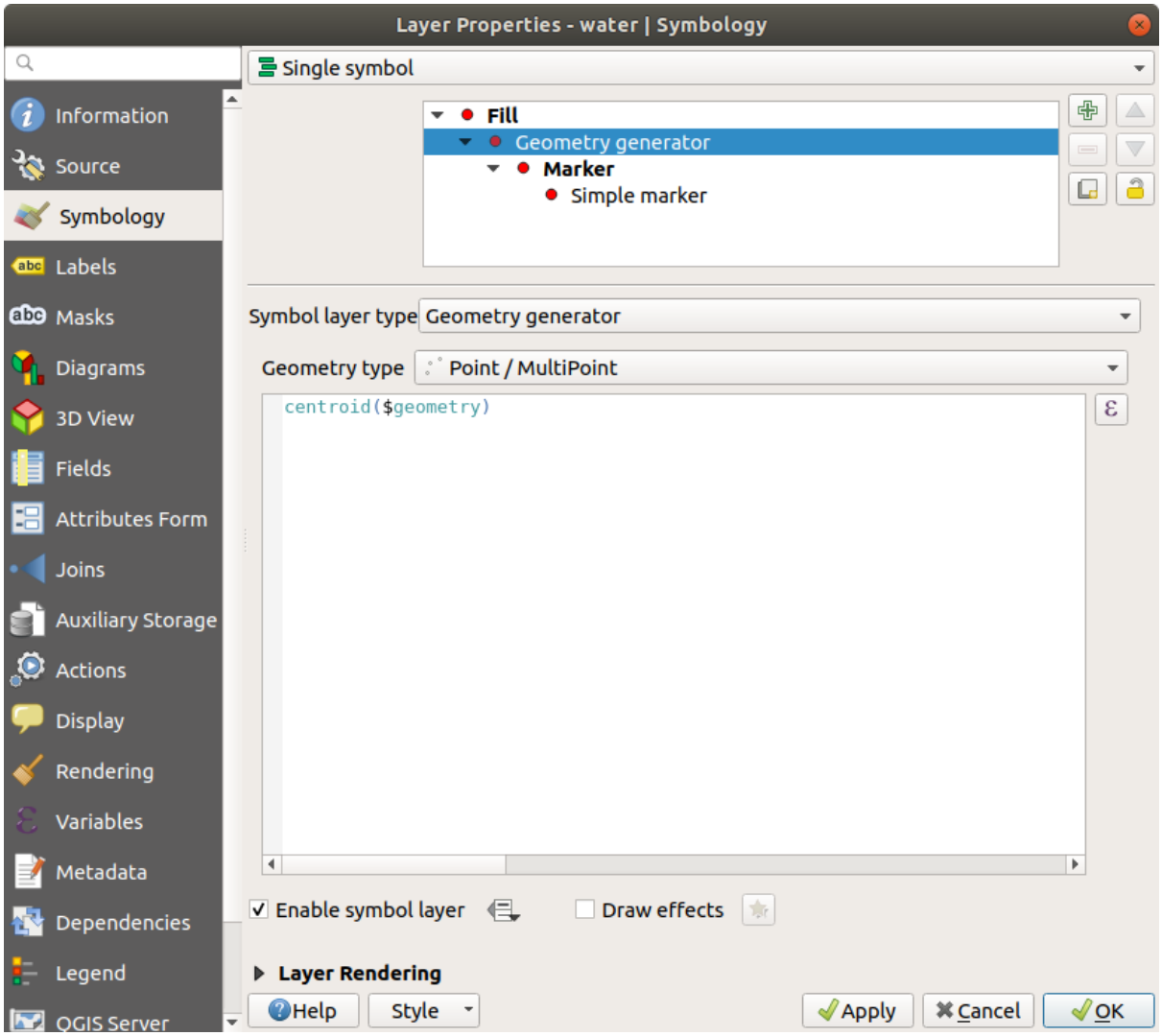
1. Select the `water` layer.
2. Click on *Simple fill* and change the *Symbol layer type* to *Geometry generator*.



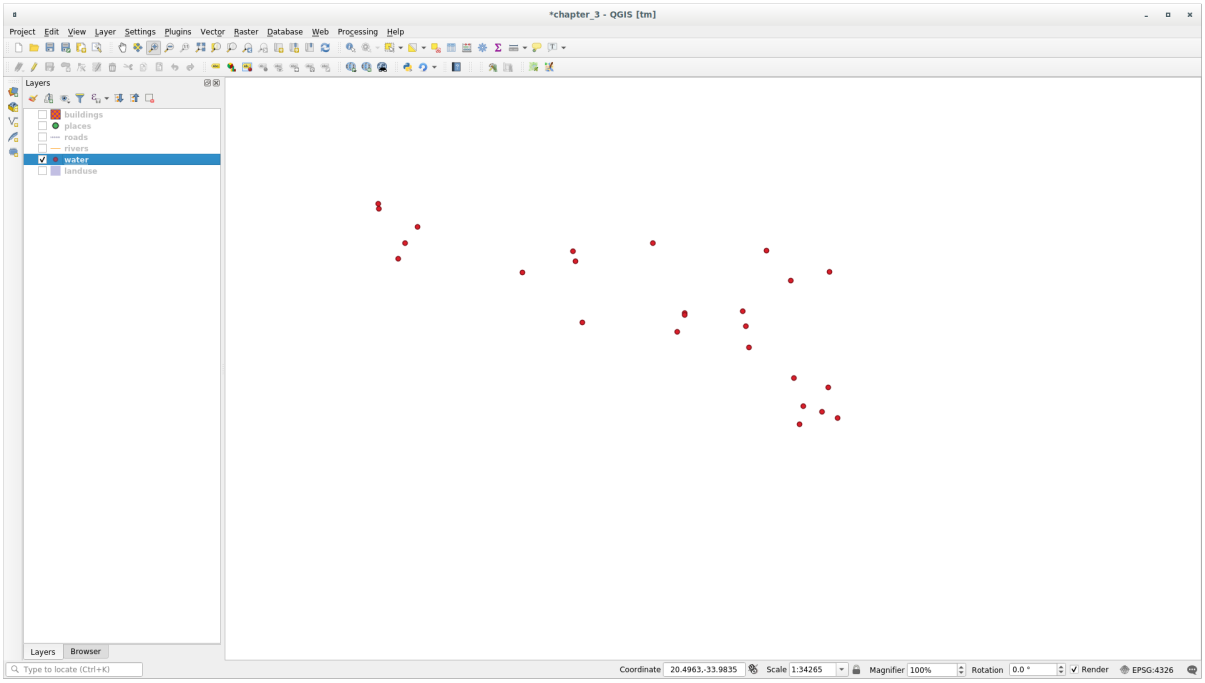
3. Before to start writing the spatial query we have to choose the Geometry Type in output. In this example we are going to create centroids for each feature, so change the Geometry Type to *Point / Multipoint*.

4. Now let's write the query in the query panel:

```
centroid($geometry)
```



5. When you click on *OK* you will see that the `water` layer is rendered as a point layer! We have just run a spatial operation within the layer symbology itself, isn't that amazing?



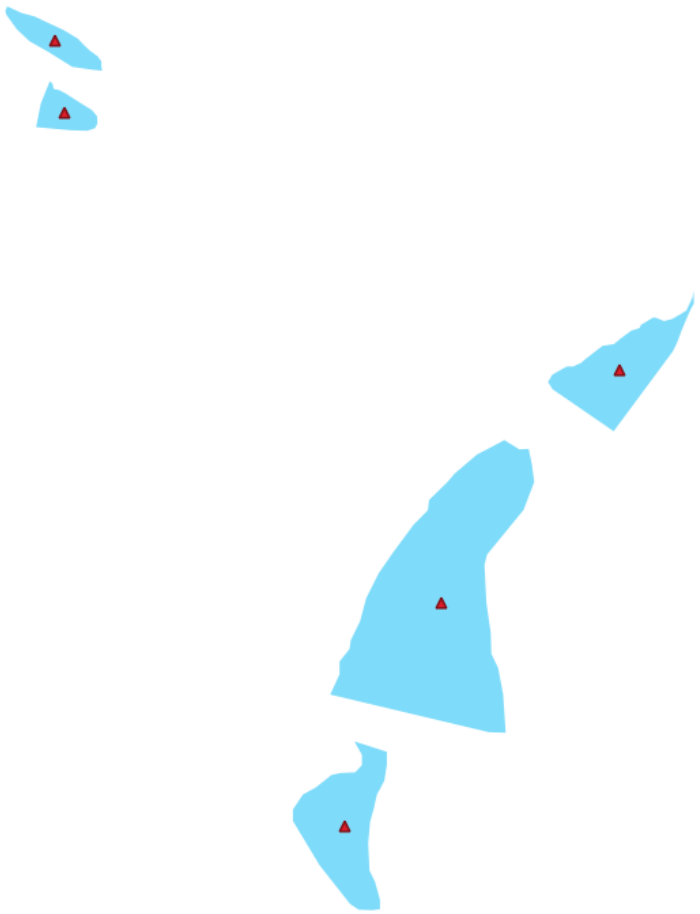
With the Geometry generator symbology you can really go over the edge of *normal* symbology.

??? Try Yourself:

Geometry generator are just another symbol level. Try to add another *Simple fill* underneath the *Geometry generator* one.

Change also the appearance of the Simple marker of the Geometry generator symbology.

The final result should look like this:



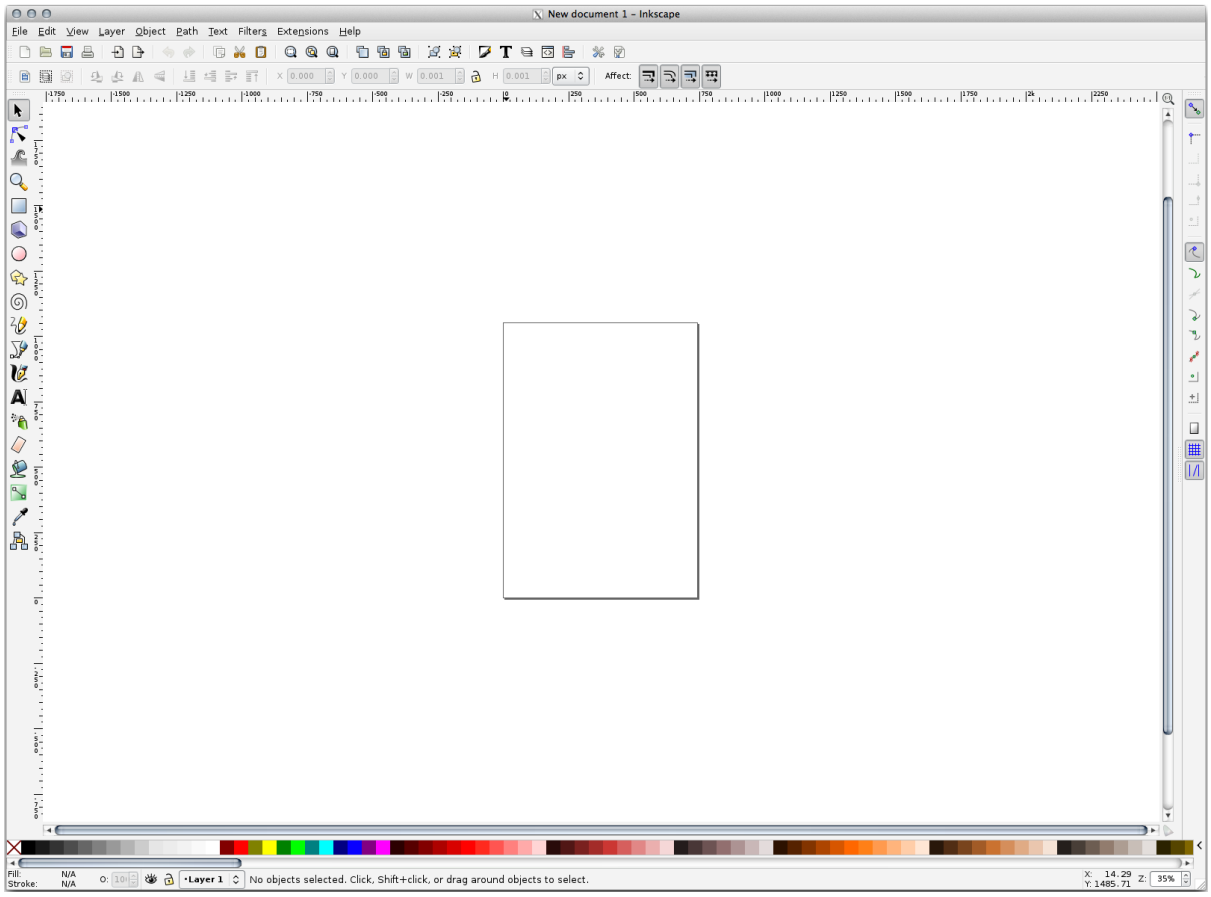
Answer

1. Click on the button to add another Symbol level.
2. Move the new symbol at the bottom of the list clicking the button.
3. Choose a good color to fill the water polygons.
4. Click on *Marker* of the Geometry generator symbology and change the circle with another shape as your wish.
5. Try experimenting other options to get more useful results.

2.4.14 ??? Follow Along: Creating a Custom SVG Fill

Nota: To do this exercise, you will need to have the free vector editing software Inkscape installed.

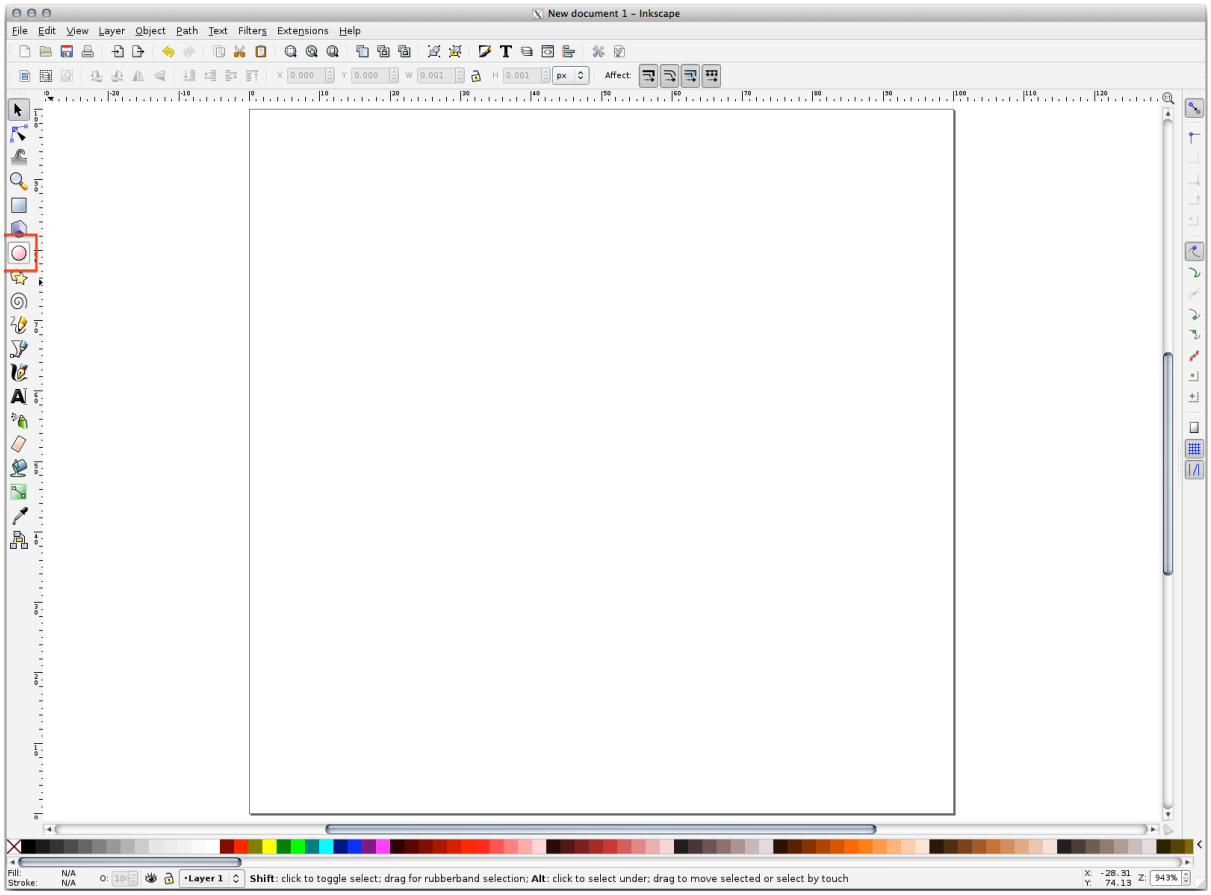
1. Start the Inkscape program. You will see the following interface:



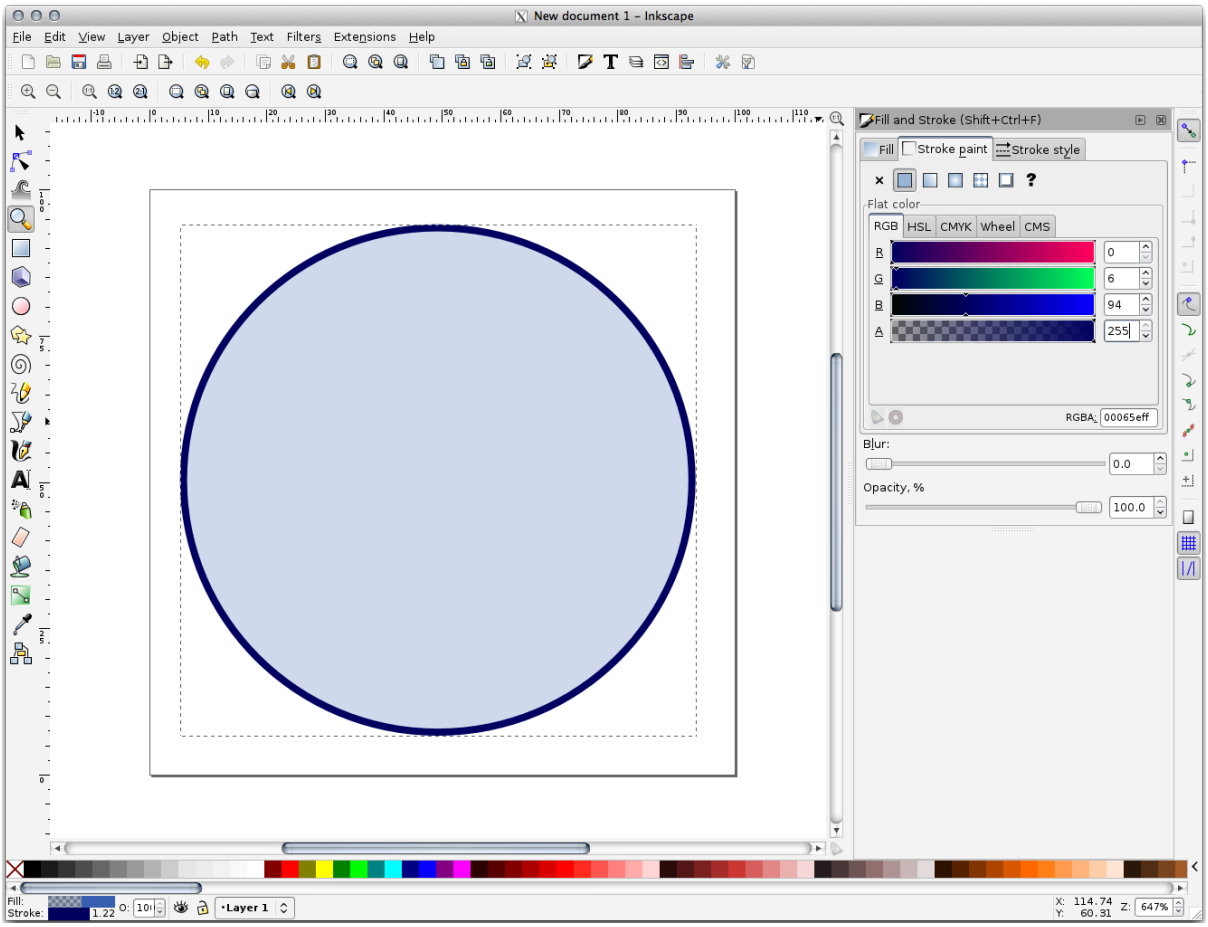
You should find this familiar if you have used other vector image editing programs, like Corel.

First, we'll change the canvas to a size appropriate for a small texture.

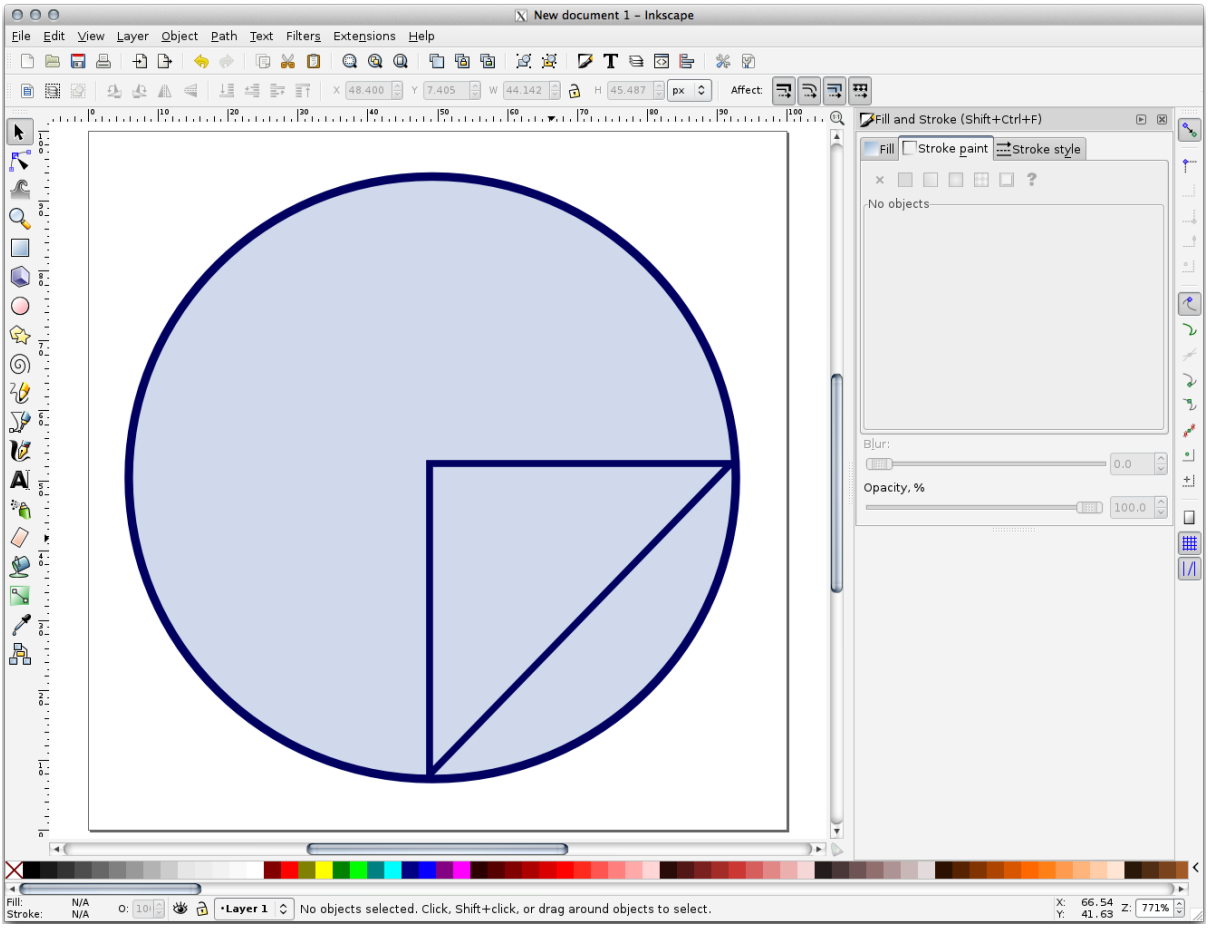
2. Click on the menu item *File* ► *Document Properties*. This will give you the *Document Properties* dialog.
3. Change the *Units* to *px*.
4. Change the *Width* and *Height* to `100`.
5. Close the dialog when you are done.
6. Click on the menu item *View* ► *Zoom* ► *Page* to see the page you are working with.
7. Select the *Circle* tool:



8. Click and drag on the page to draw an ellipse. To make the ellipse turn into a circle, hold the `Ctrl` button while you're drawing it.
9. Right-click on the circle you just created and open its *Fill and Stroke* options. You can modify its rendering, such as:
    1. Change the *Fill* color to a somehow pale grey-blue,
    2. Assign to the border a darker color in *Stroke paint* tab,
    3. And reduce the border thickness under *Stroke style* tab.



10. Draw a line using the *Pencil* tool:
    1. Click once to start the line. Hold `Ctrl` to make it snap to increments of 15 degrees.
    2. Move the pointer horizontally and place a point with a simple click.
    3. Click and snap to the vertex of the line and trace a vertical line, ended by a simple click.
    4. Now join the two end vertices.
    5. Change the color and width of the triangle symbol to match the circle's stroke and move it around as necessary, so that you end up with a symbol like this one:

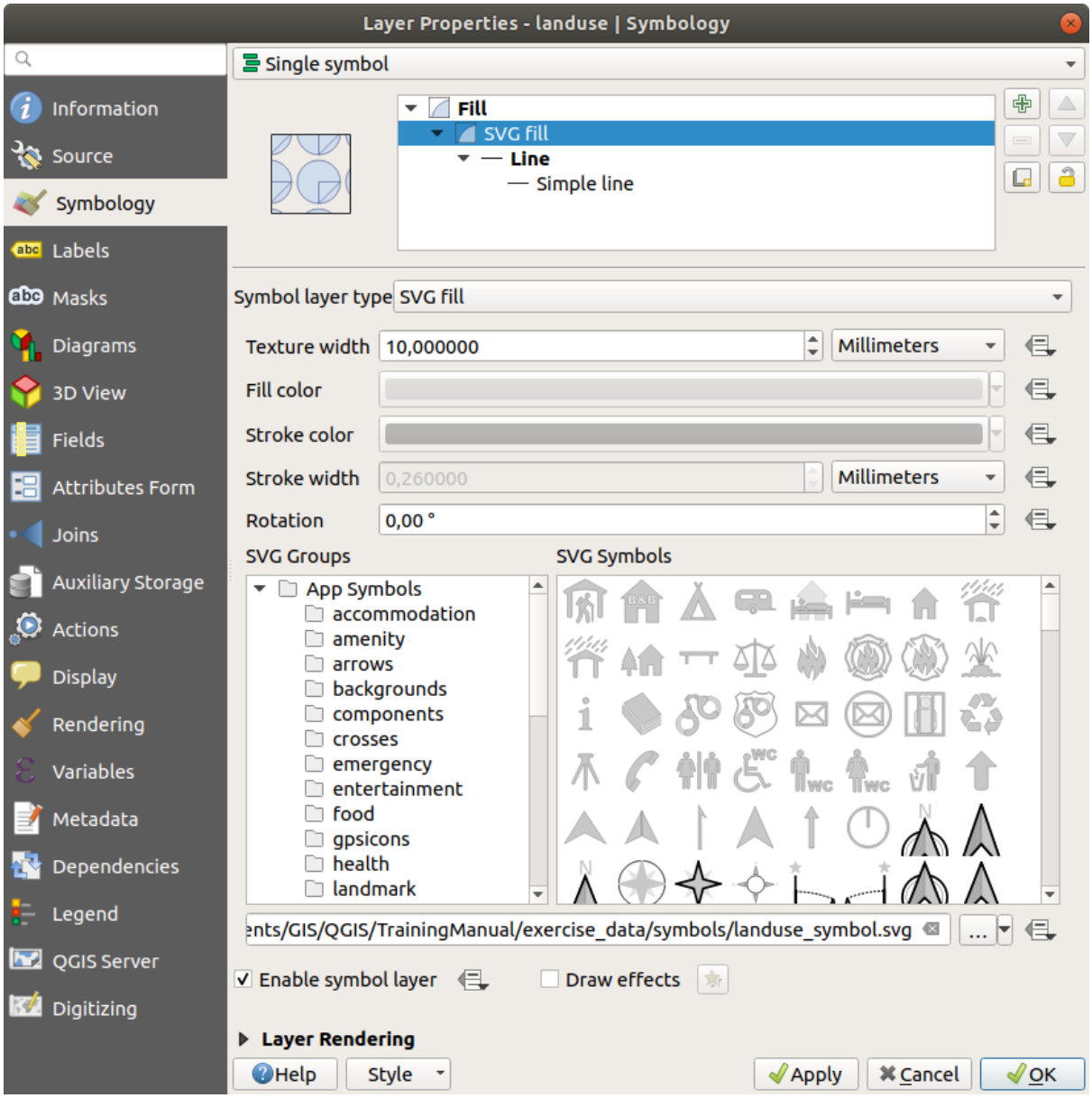


11. If the symbol you get satisfies you, then save it as `landuse_symbol` under the directory that the course is in, under `exercise_data/symbols`, as SVG file.

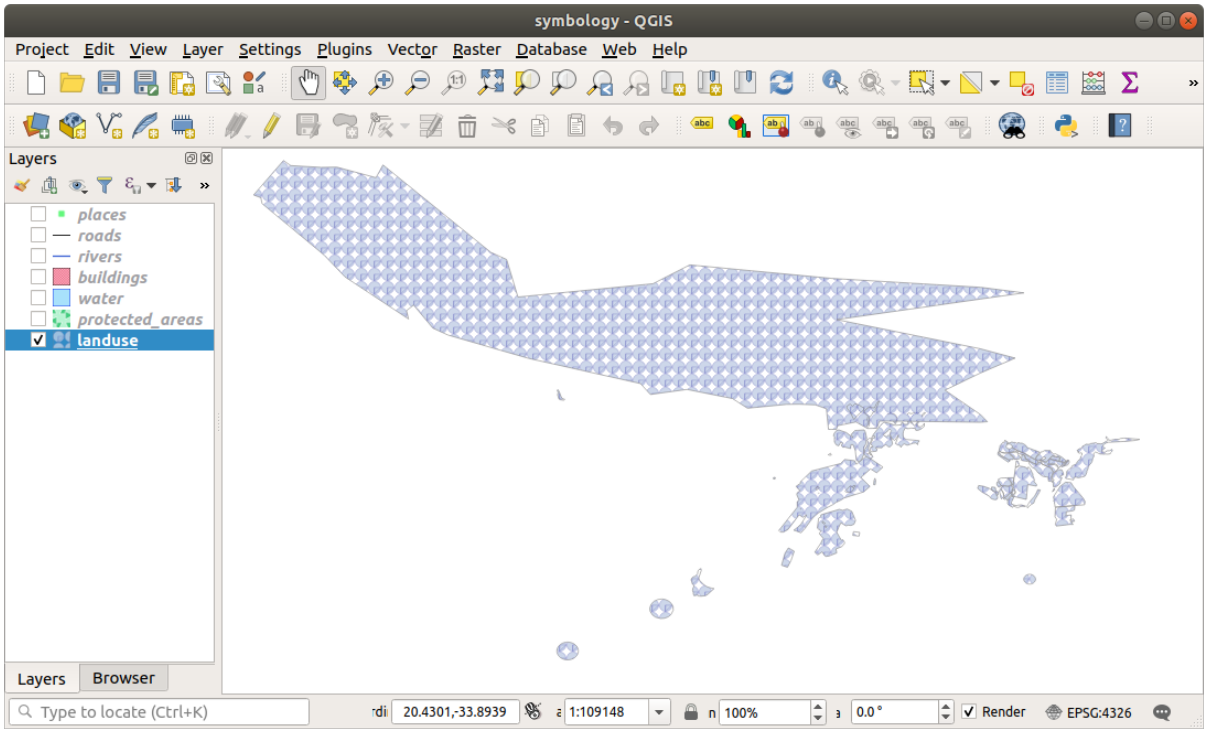
In QGIS:

1. Open the *Layer Properties* for the `landuse` layer.
2. In the *Symbology* tab, change the symbol structure by changing the *Symbol Layer Type* to *SVG Fill* shown below.
3. Click the … button and then *Select File…* to select your SVG image.

   It's added to the symbol tree and you can now customize its different characteristics (colors, angle, effects, units…).



Once you validate the dialog, features in `landuse` layer should now be covered by a set of symbols, showing a texture like the one on the following map. If textures are not visible, you may need to zoom in the map canvas or set in the layer properties a bigger *Texture width*.



2.4.15 ??? Follow Along: Masking

Masking feature allows you to mask some symbol layer using another symbol layer or label.

In the following exercise, we want to configure rendering so that polygon labels will mask the black part of lines layer.

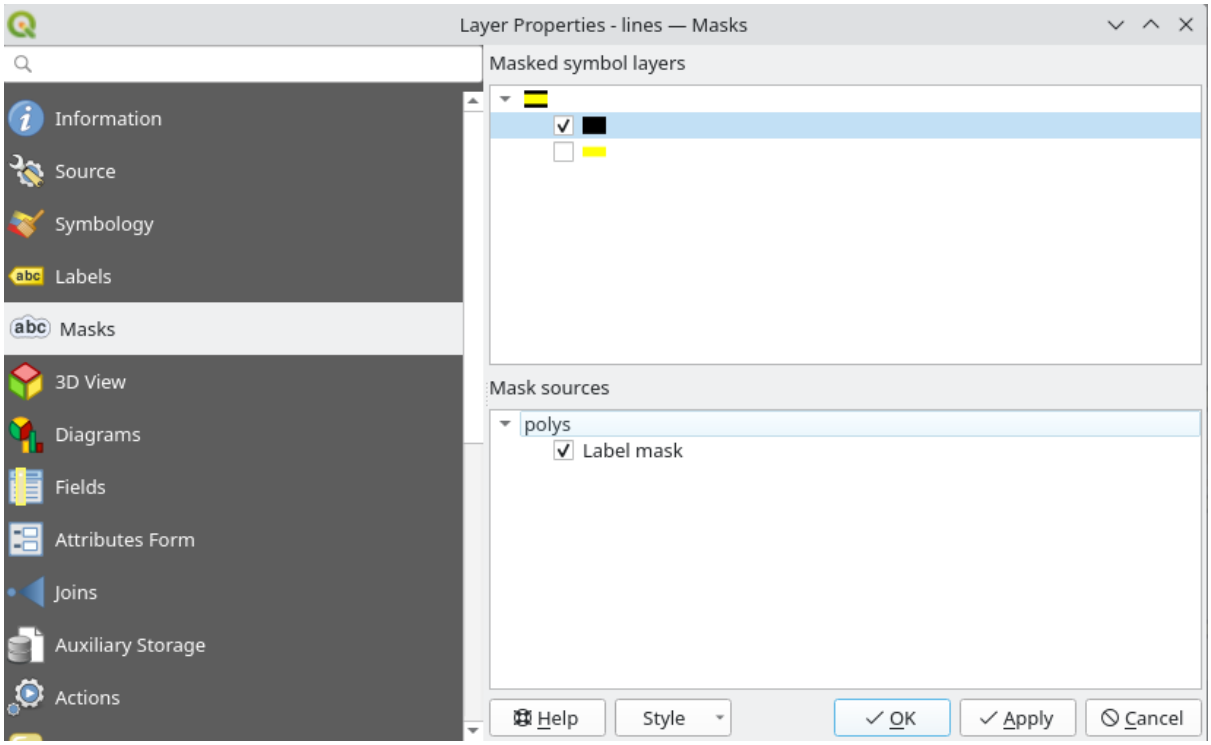
1. Load the `selective_masking.qgs` from dataset in directory `exercise_data/masking`.
2. Open the *Layer Properties* window for the `polys` layer.

   Under the *Labels* tab, you will see that *Single Labels* have already been defined.
3. From the *Labels* tab, select the *Mask* (and not *Masks* in general category list)
4. Enable label mask by checking the *Enable mask* option

   A message tells you that the now defined label mask shape can be later selected as a mask source to mask other symbol layers. Let's do that.
5. Click *OK*
6. Open the *Layer Properties* window for the `lines` layer.
7. Go to the *Masks* tab
8. Click on the black line part of lines checkbox from masked symbol layers

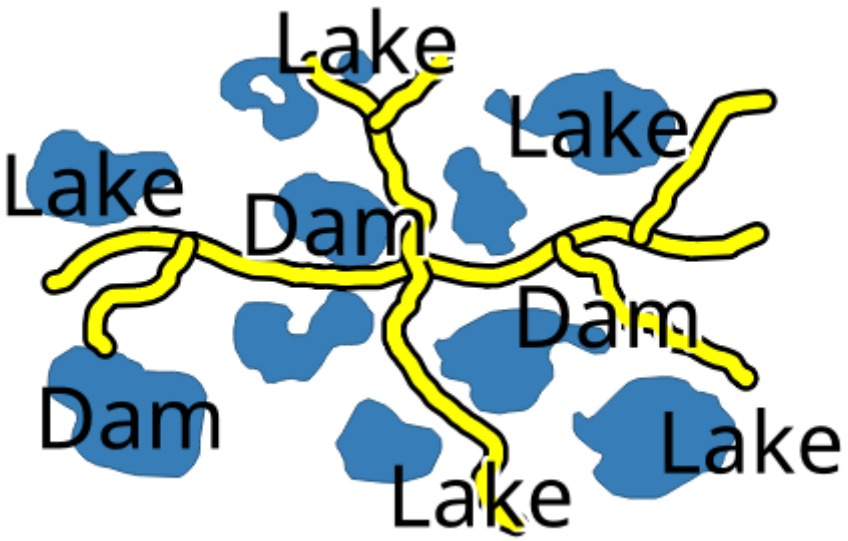
   A message warns you that you have to select both masking and masked symbol layers to be able to save masking configuration. Again, let's do that.
9. Click on `polys` *Label mask* checkbox

   The message disappears, you can now safely save your configuration.



10. Click *OK*

You can see now that labels are masking the black part of lines, leaving only the yellow part.



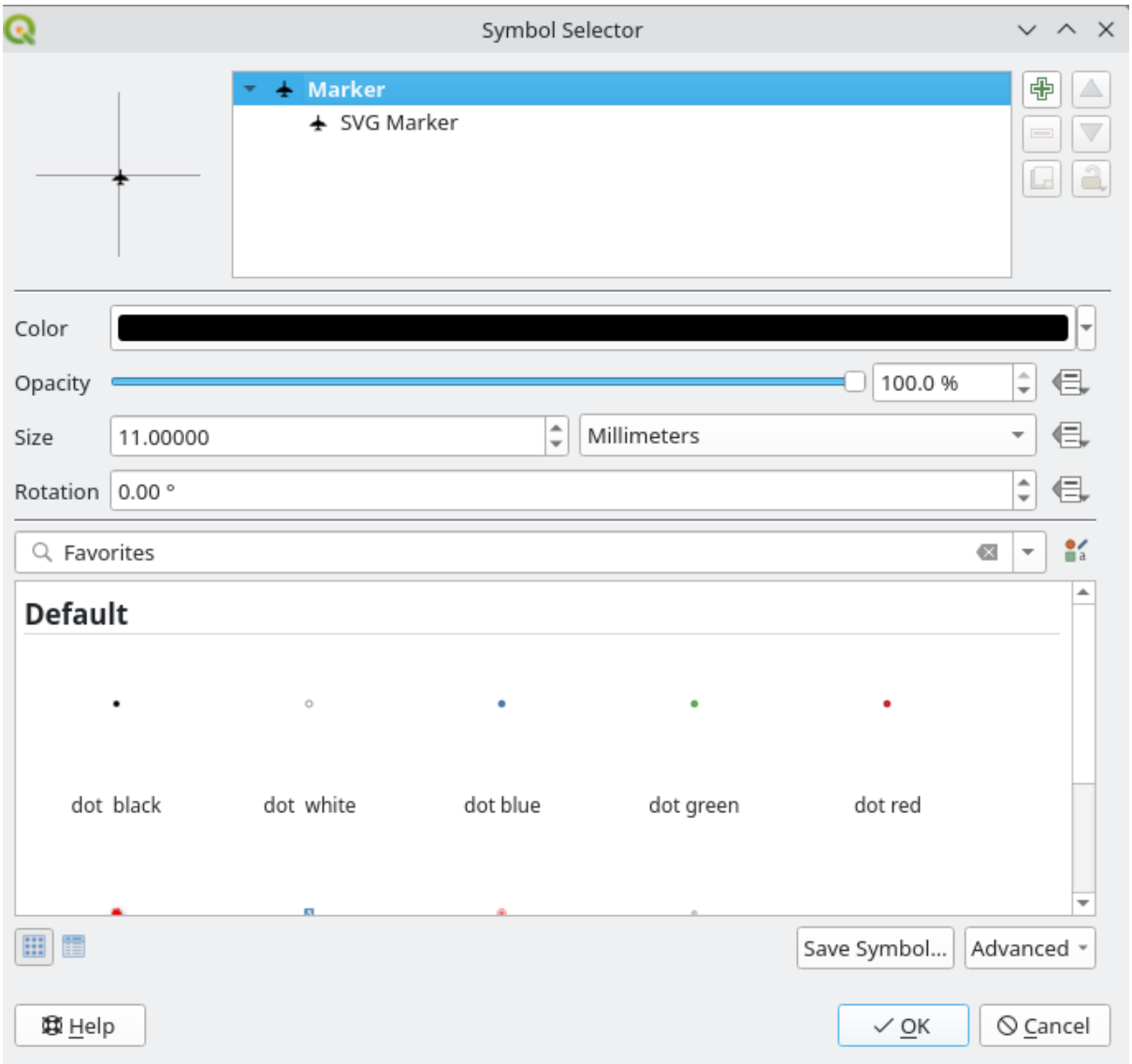
2.4.16 ??? Follow Along: Masking using points symbols

We want now to mask the black part of lines using only black planes that represent the `points` layer (i.e. `B52` and `Jet` points categories). For the sake of visibility, you can switch off `polys` layer visibility.

1. Open the *Layer Properties* window for the `points` layer.

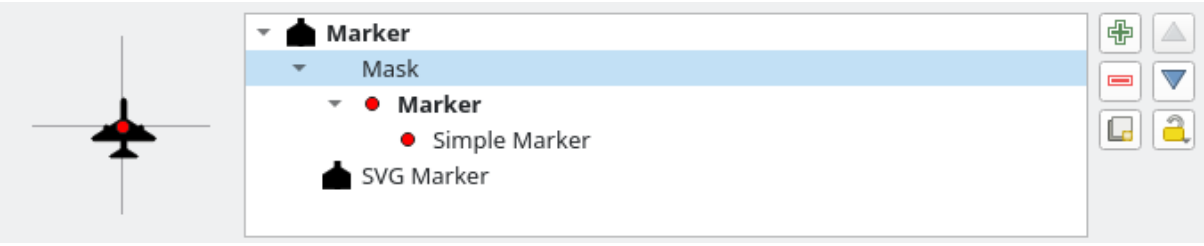
   Under the *Symbology* tab, you will see that a *Categorized* renderer has been defined with 3 different symbols depending on the `Class` field value.

2. Double click on `B52` plane icon, within the *Symbol* column



3. Click the Add symbol layer button and select the `Mask` *Symbol layer type*

   A new hierarchy `Mask` (*symbol layer*) > `Marker` (*symbol*) > `Simple Marker` (*symbol layer*) appear below the root symbol `Marker`.

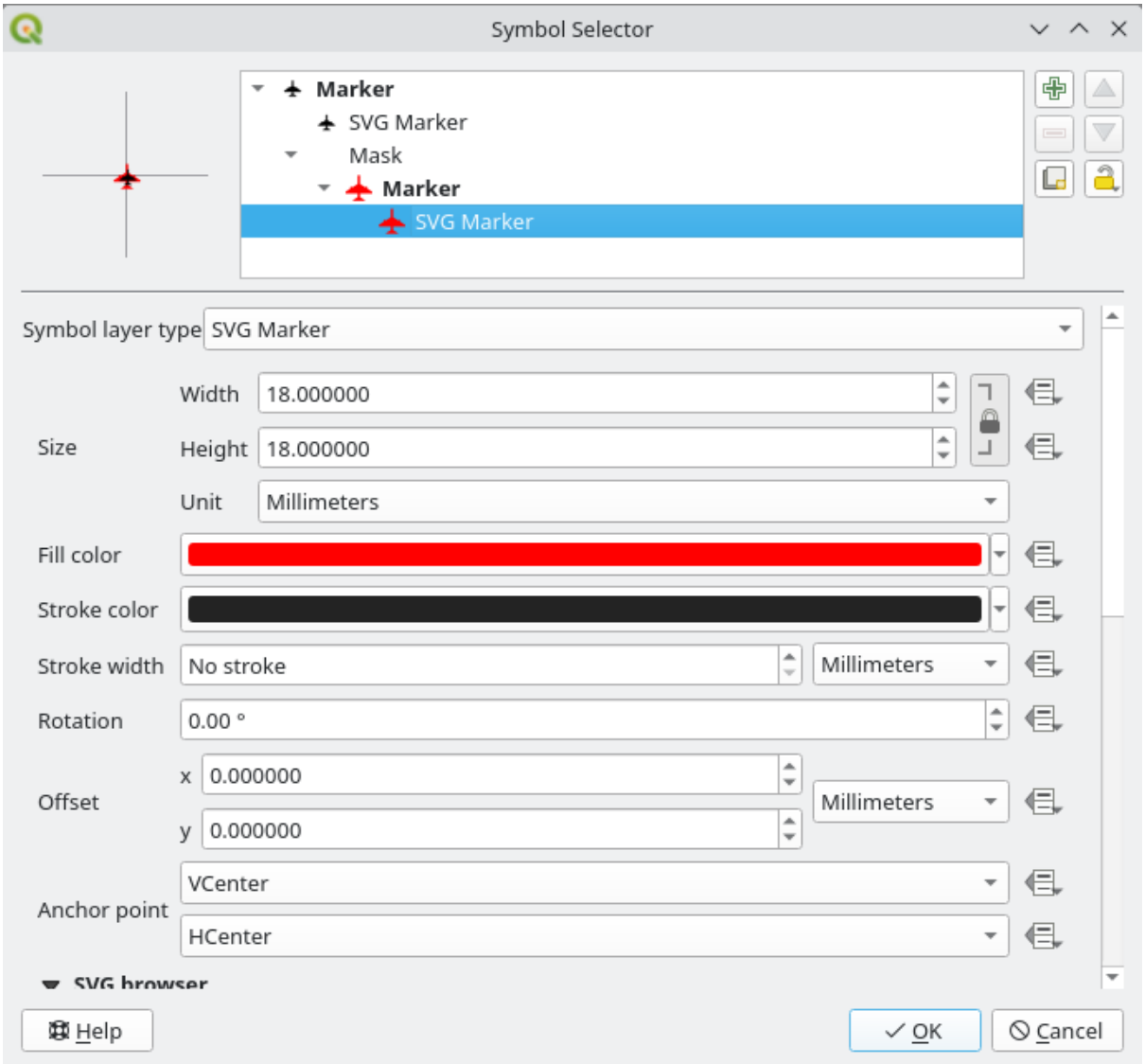


Now, let's define a mask that is identical to our plane marker but a little bit bigger.

4. Select the newly created `Simple Marker` symbol layer and change its *Symbol layer type* to `SVG marker`
5. Set the SVG file path to be the same as the already existing SVG marker, and the size to be bigger (24 millimeters is a pretty good choice for this use case).

   Mask color doesn't matter here, only the shape is used in masking. Symbol layer order is also irrelevant in that situation.

   You should get something like that:

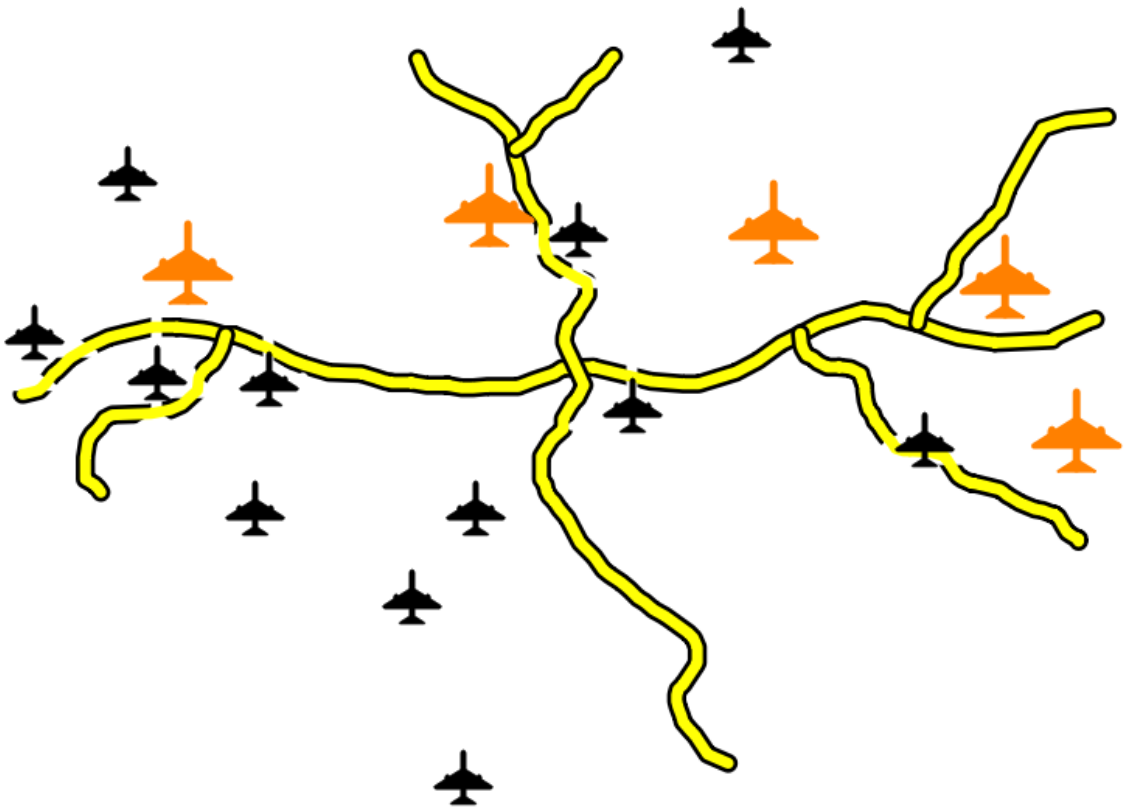


6. Repeat the same operations on `Jet` symbol.
7. Click *OK*

8. Open the *Layer Properties* window for the `lines` layer.
9. Go to the *Masks* tab.

   The black line part of lines checkbox from masked symbol layers is already checked from our previous configuration.
10. Uncheck the *Label mask* `polys` checkbox and check both `B52` and `Jet` *Mask symbol layer*
11. Click *OK*

You now see that black planes are masking the black part of lines, with only the yellow part visible.



2.4.17 In Conclusion

Changing the symbology for the different layers has transformed a collection of vector files into a legible map. Not only can you see what's happening, it's even nice to look at!

2.4.18 Further Reading

Examples of Beautiful Maps

2.4.19 What's Next?

Changing symbols for whole layers is useful, but the information contained within each layer is not yet available to someone reading these maps. What are the streets called? Which administrative regions do certain areas belong to? What are the relative surface areas of the farms? All of this information is still hidden. The next lesson will explain how to represent this data on your map.

Nota: Did you remember to save your map recently?

CAPÍTULO 3

Module: Classifying Vector Data

A classificação de dados vetoriais permite atribuir símbolos diferentes aos elementos (objetos diferentes na mesma camada), de acordo com os seus atributos. Isso permite a alguém que use o mapa veja facilmente os atributos de vários elementos.

3.1 Lesson: Vector Attribute Data

Vector data is arguably the most common kind of data in the daily use of GIS. The vector model represents the location and shape of geographic features using points, lines and polygons (and for 3D data also surfaces and volumes), while their other properties are included as attributes (often presented as a table in QGIS).

Up to now, none of the changes we have made to the map have been influenced by the objects that are being shown. In other words, all the land use areas look alike, and all the roads look alike. When looking at the map, the viewers don't know anything about the roads they are seeing; only that there is a road of a certain shape in a certain area.

But the whole strength of GIS is that all the objects that are visible on the map also have attributes. Maps in a GIS aren't just pictures. They represent not only objects in locations, but also information about those objects.

The goal for this lesson: To learn about the structure of vector data and explore the attribute data of an object

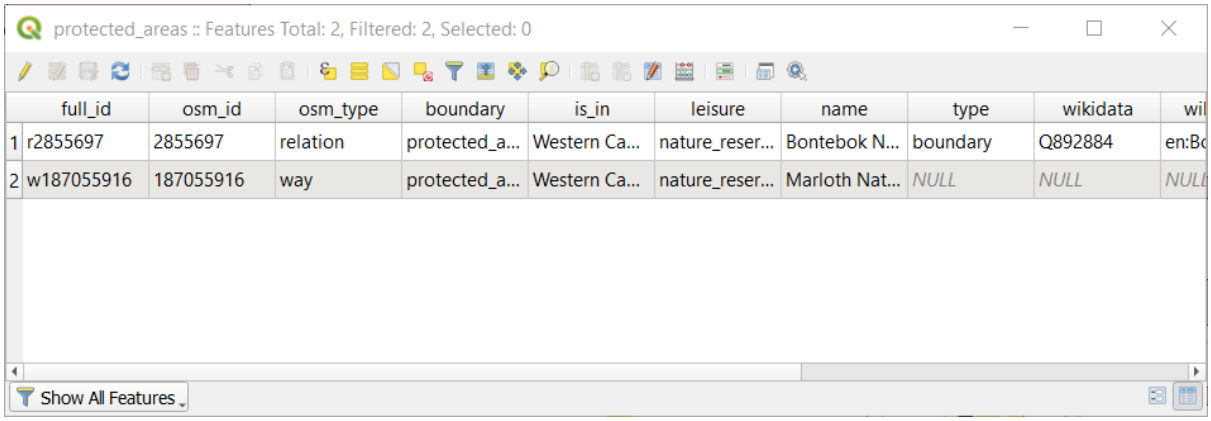
3.1.1 ??? Follow Along: Viewing Layer Attributes

It's important to know that the data you will be working with does not only represent **where** objects are in space, but also tells you **what** those objects are.

From the previous exercise, you should have the `protected_areas` layer loaded in your map. If it is not loaded, then you can find the `protected_areas.shp` *ESRI Shapefile* format dataset in directory `exercise_data/shapefile`.

The polygons representing the protected areas constitute the **spatial data**, but we can learn more about the protected areas by exploring the **attribute table**.

1. In the *Layers* panel, click on the `protected_areas` layer to select it.
2. In the *Layer* menu, click the Open Attribute Table button (also accessible from top toolbars buttons). This will open a new window showing the attribute table of the `protected_areas` layer.



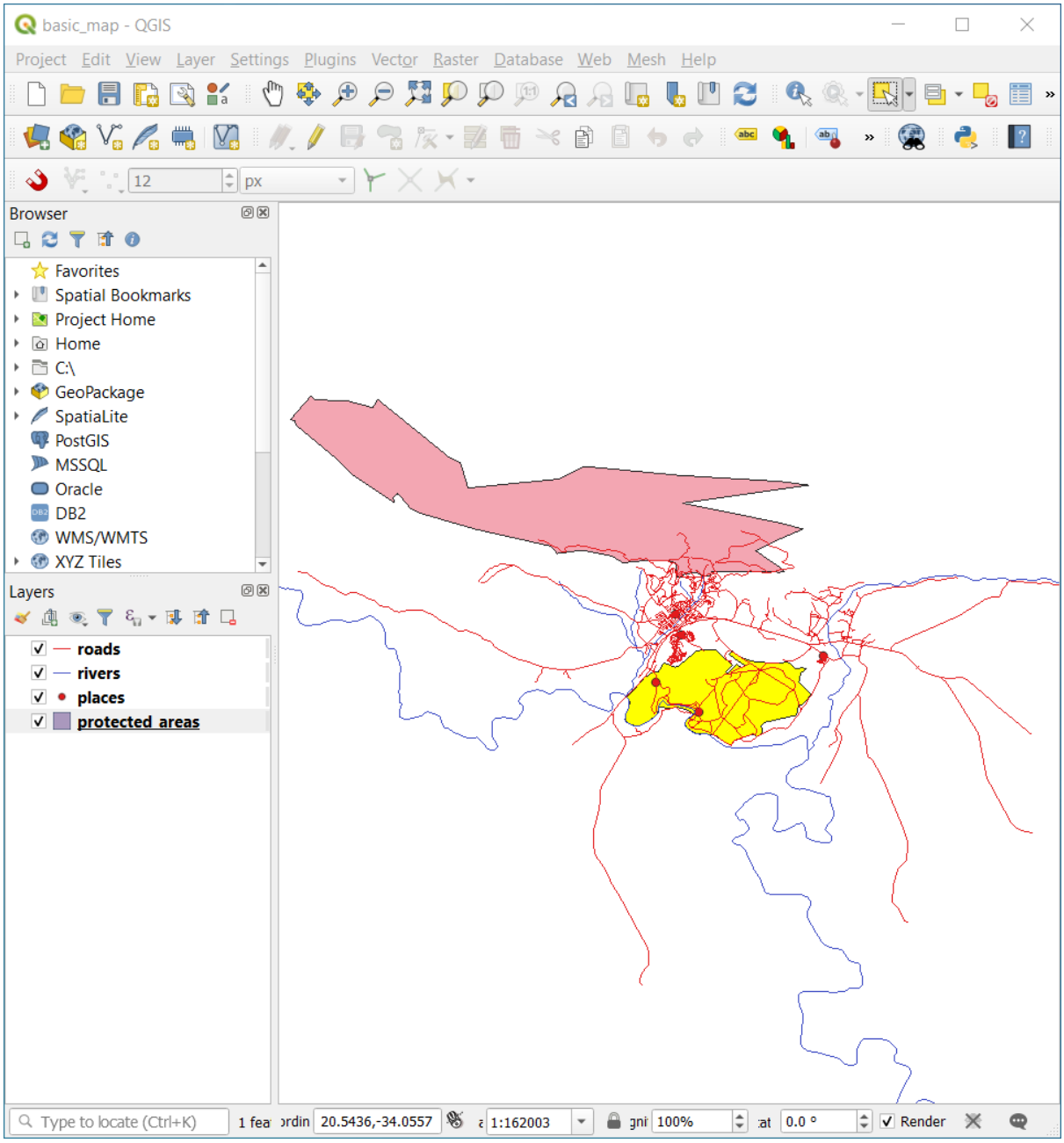
A row is called a **record** and is associated with a **feature** in the Canvas Map, such as a polygon. A column is called a **field** (or an **attribute**), and has a name that helps describe it, such as `name` or `id`. Values in the cells are known as **attribute values**. These definitions are commonly used in GIS, so it is good to become familiar with them.

In the `protected_areas` layer, there are two **features**, which are represented by the two polygons we see on the Map Canvas.

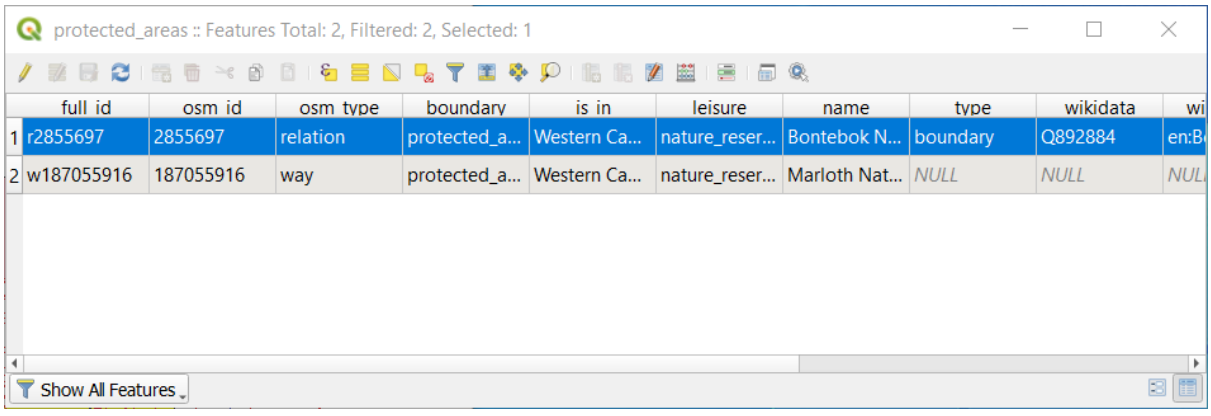
Nota: In order to understand what the **fields** and **attribute values** represent, one may need to find documentation (or metadata) describing the meaning of the attribute values. This is usually available from the creator of the data set.

Next, let's see how a record in the attribute table is linked to a polygon feature that we see on the Map Canvas.

1. Go back to the main QGIS window.
2. In the *Edit* ► *Select* menu, click on the *Select Feature(s)* button.
3. Make sure the `protected_areas` layer is still selected in the *Layers* panel.
4. Move your mouse to the Map Canvas and left click on the smaller of the two polygons. The polygon will turn yellow indicating it is selected.

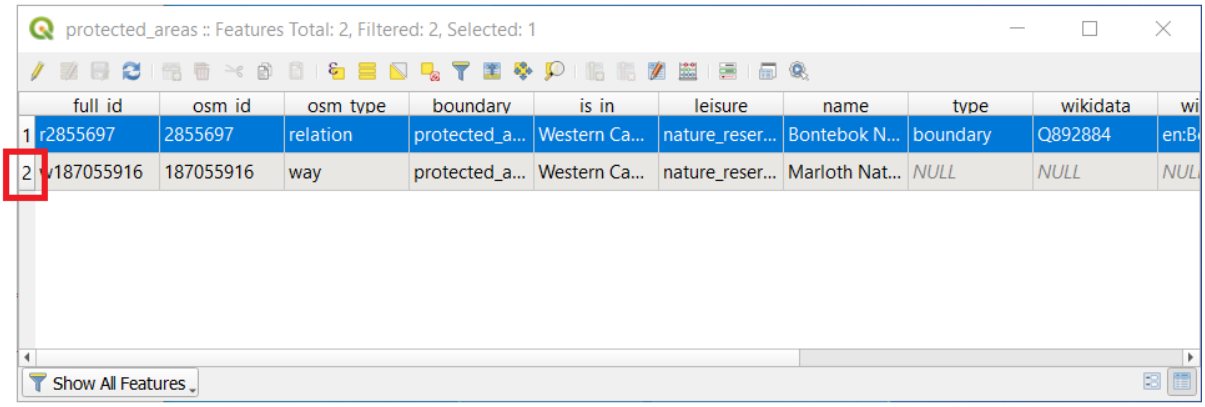


5. Go back to the *Attribute Table* window, and you should see a record (row) highlighted. These are the attribute values of the selected polygon.



You can also select a feature using the Attribute Table.

1. In the *Attribute Table* window, on the far left, click on the row number of the record that is currently not selected.



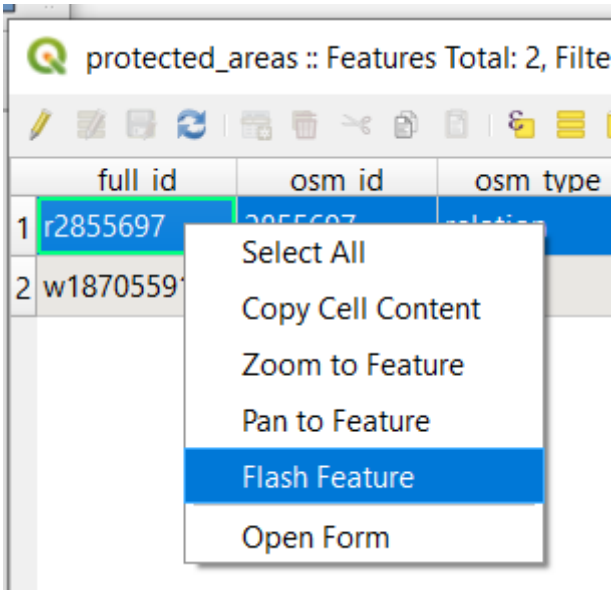
2. Go back to the main QGIS window and look at the Map Canvas. You should see the larger of the two polygons colored yellow.

3. To deselect the feature, go to the *Attribute Table* window and click on Deselect all features from the layer button.

Sometimes there are many features shown on the Map Canvas and it might be difficult to see which feature is selected from the Attribute Table. Another way to identify the location of a feature is to use the *Flash Feature* tool.

1. In the *Attribute Table*, right-click on any cell in the row that has the attribute value `r2855697` for the field `full_id`.

2. In the context menu, click on *Flash Feature* and watch the Map Canvas.

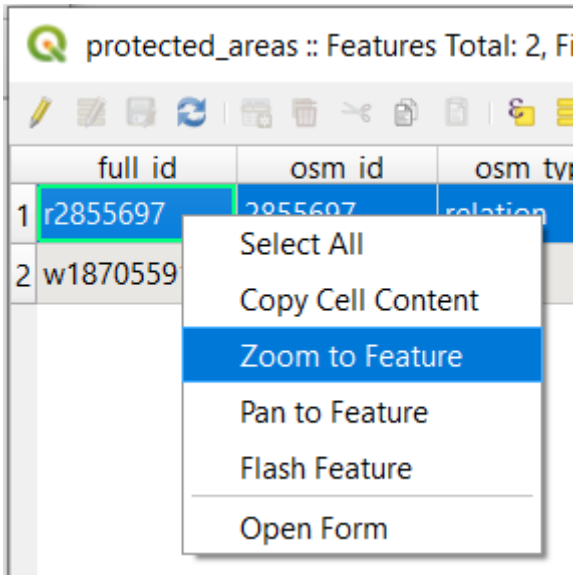


You should see the polygon flash red a few times. If you missed it, try it again.

Another useful tool is the *Zoom to Feature* tool, that tells QGIS to zoom to the feature of interest.

1. In the *Attribute Table*, right-click on any cell in the row that has the attribute value `r2855697` for the field `full_id`.

2. In the context menu, click on *Zoom to Feature*



Look at the Map Canvas. The polygon should now occupy the extent of the Map Canvas area.

You may now close the attribute table.

3.1.2 ??? Try Yourself: Exploring Vector Data Attributes

1. How many fields are available in the *rivers* layer?
2. Tell us a bit about the `town` places in your dataset.
3. Open the attribute table for the *places* layer. Which field would be the most useful to represent in label form, and why?

Answer

- There should be 9 fields in the *rivers* layer:
  1. Select the layer in the *Layers* panel.
  2. Right-click and choose *Open Attribute Table*, or press the button on the *Attributes Toolbar* (it can be enabled from *View* ► *Toolbars* menu).
  3. Count the number of columns.

  A quicker approach could be to double-click the *rivers* layer, open the *Layer properties* ► *Fields* tab, where you will find a numbered list of the table's fields.
- Information about towns is available in the *places* layer. Open its attribute table as you did with the *rivers* layer: there are two features whose *place* attribute is set to `town`: *Swellendam* and *Buffeljagsrivier*. You can add comment on other fields from these two records, if you like.
- The `name` field is the most useful to show as labels. This is because all its values are unique for every object and are very unlikely to contain *NULL* values. If your data contains some *NULL* values, do not worry as long as most of your places have names.

3.1.3 In Conclusion

You now know how to use the attribute table to see what is actually in the data you're using. Any dataset will only be useful to you if it has the attributes that you care about. If you know which attributes you need, you can quickly decide if you're able to use a given dataset, or if you need to look for another one that has the required attribute data.

3.1.4 What's Next?

Different attributes are useful for different purposes. Some of them can be represented directly as text for the map user to see. You'll learn how to do this in the next lesson.

3.2 Lesson: Labels

Labels can be added to a map to show any information about an object. Any vector layer can have labels associated with it. These labels rely on the attribute data of a layer for their content.

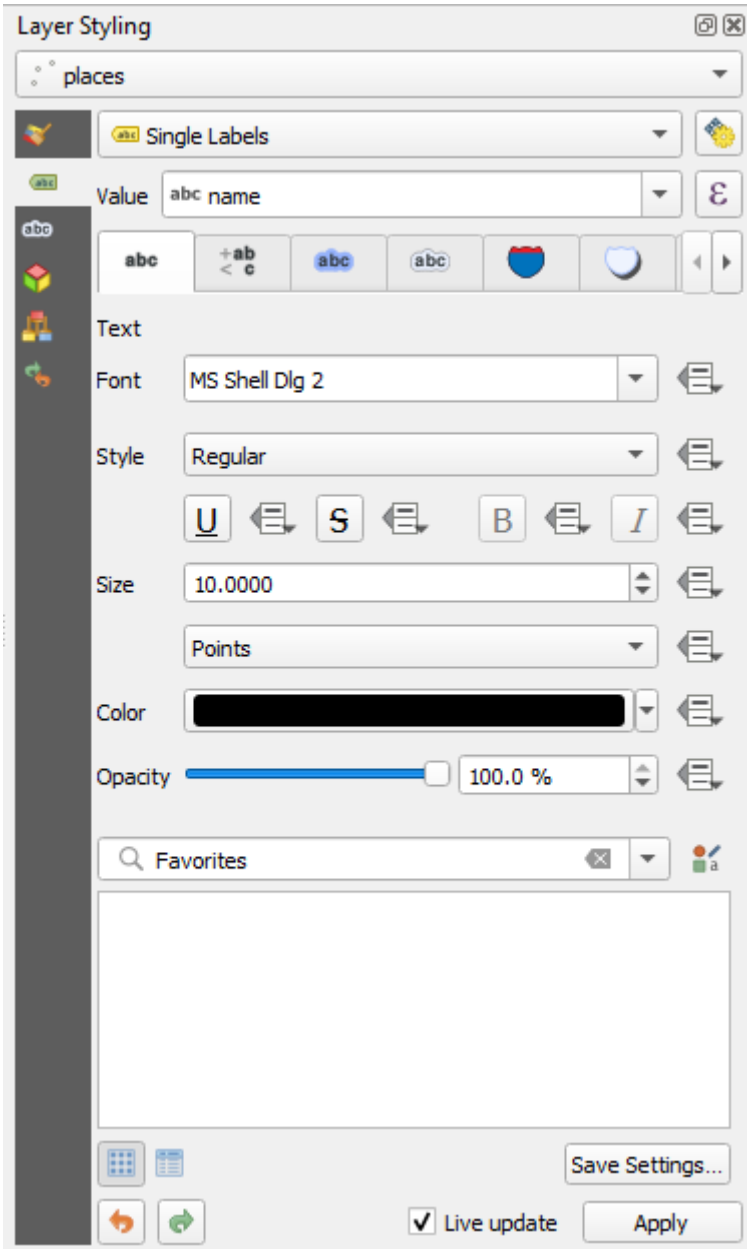
The goal for this lesson: To apply useful and good-looking labels to a layer.

3.2.1 ??? Follow Along: Using Labels

First, ensure that the abc button is visible in the GUI:

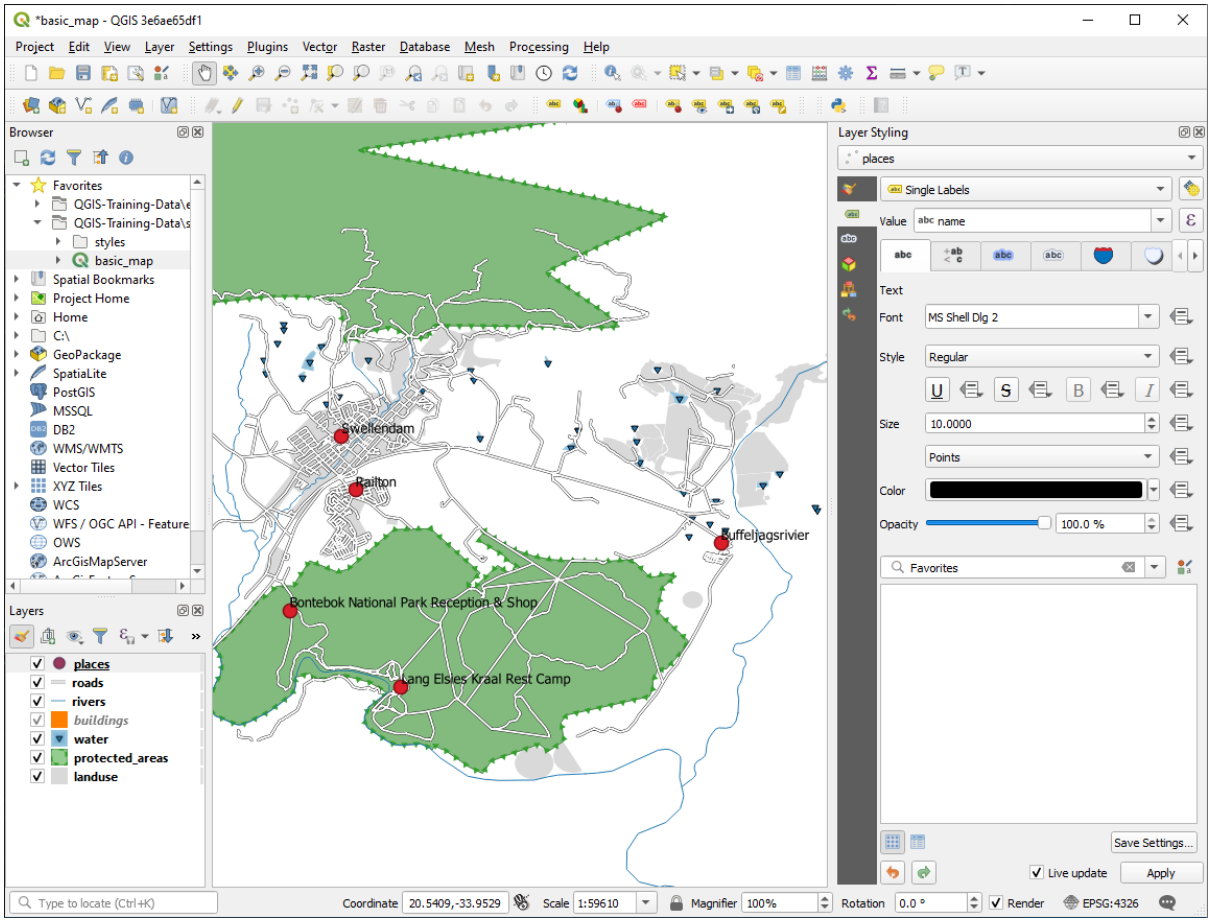
1. Go to the menu item *View* ► *Toolbars*
2. Ensure that the *Label Toolbar* item has a check mark next to it. If it doesn't, click on the *Label Toolbar* item to activate it.
3. Click on the `places` layer in the *Layers* panel so that it is highlighted
4. Click on the abc toolbar button to open the *Labels* tab of the *Layer Styling* panel
5. Switch from *No Labels* to abc *Single Labels*

   You'll need to choose which field in the attributes will be used for the labels. In the previous lesson, you decided that the `name` field was the most suitable one for this purpose.
6. Select `name` from the Value list:



7. Click *Apply*

The map should now have labels like this:



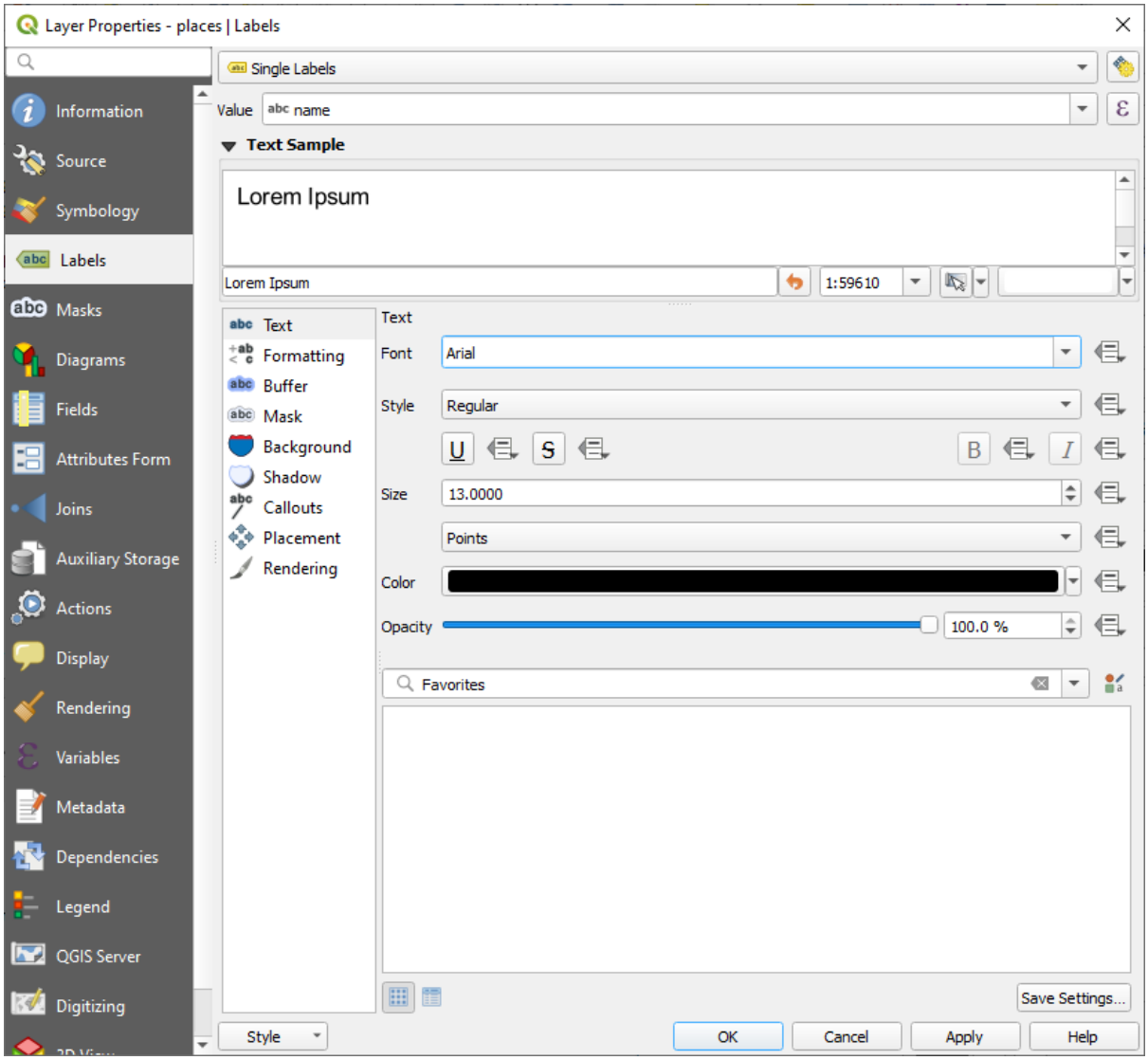
3.2.2 Follow Along: Changing Label Options

Depending on the styles you chose for your map in earlier lessons, you might find that the labels are not appropriately formatted and either overlap or are too far away from their point markers.

Nota: Above, you used the button in the *Label Toolbar* to open the *Layer Styling* panel. As with *Symbology*, the same label options are available via both the *Layer Styling* panel and the *Layer Properties* dialog. Here, you'll use the *Layer Properties* dialog.

---

1. Open the *Layer Properties* dialog by double-clicking on the `places` layer
2. Select the *Labels* tab
3. Make sure *Text* is selected in the left-hand options list, then update the text formatting options to match those shown here:

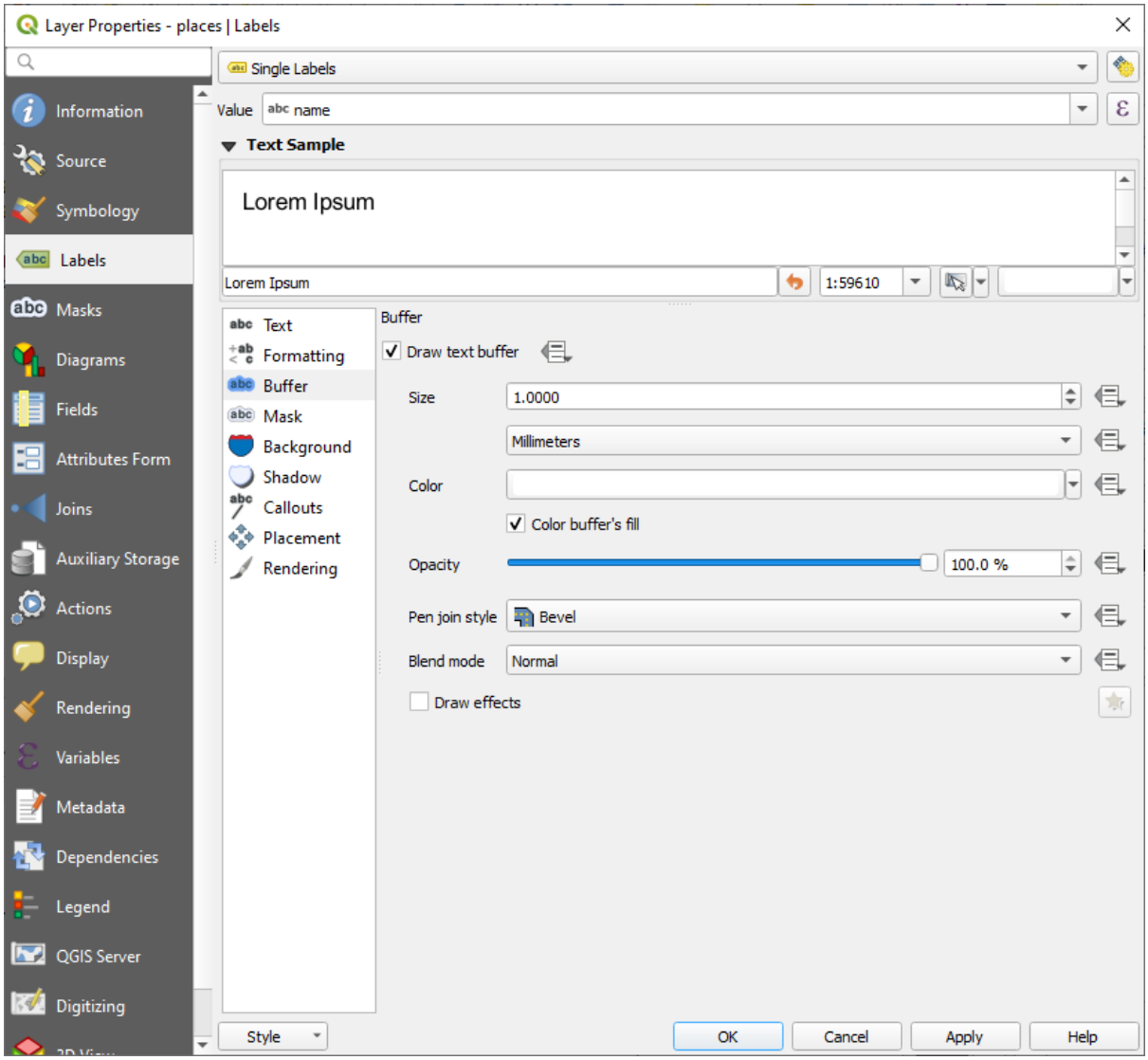


4. Click *Apply*

   That font may be larger and more familiar to users, but its readability is still dependent on what layers are rendered beneath it. To solve this, let's take a look at the *Buffer* option.

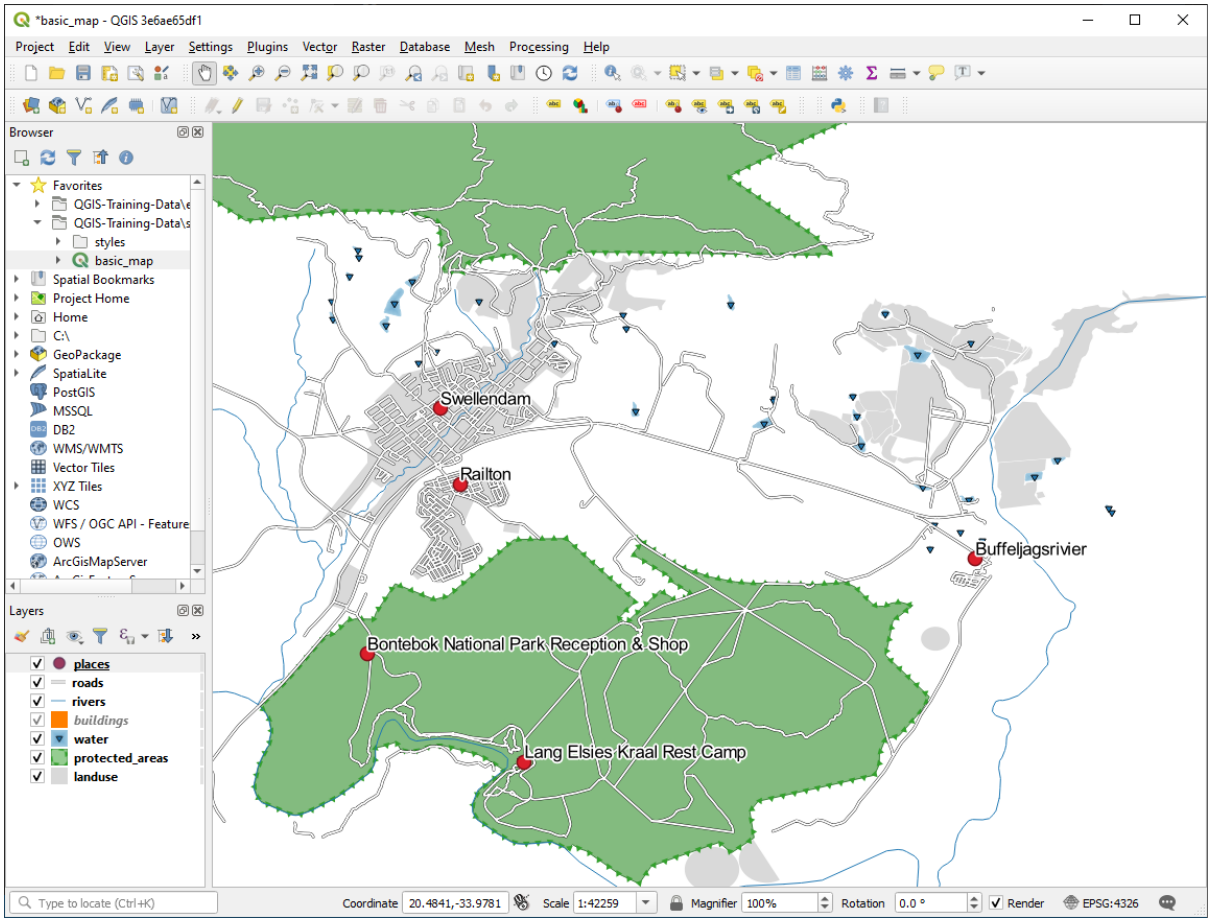
5. Select *Buffer* from the left-hand options list

6. Select the checkbox next to *Draw text buffer*, then choose options to match those shown here:



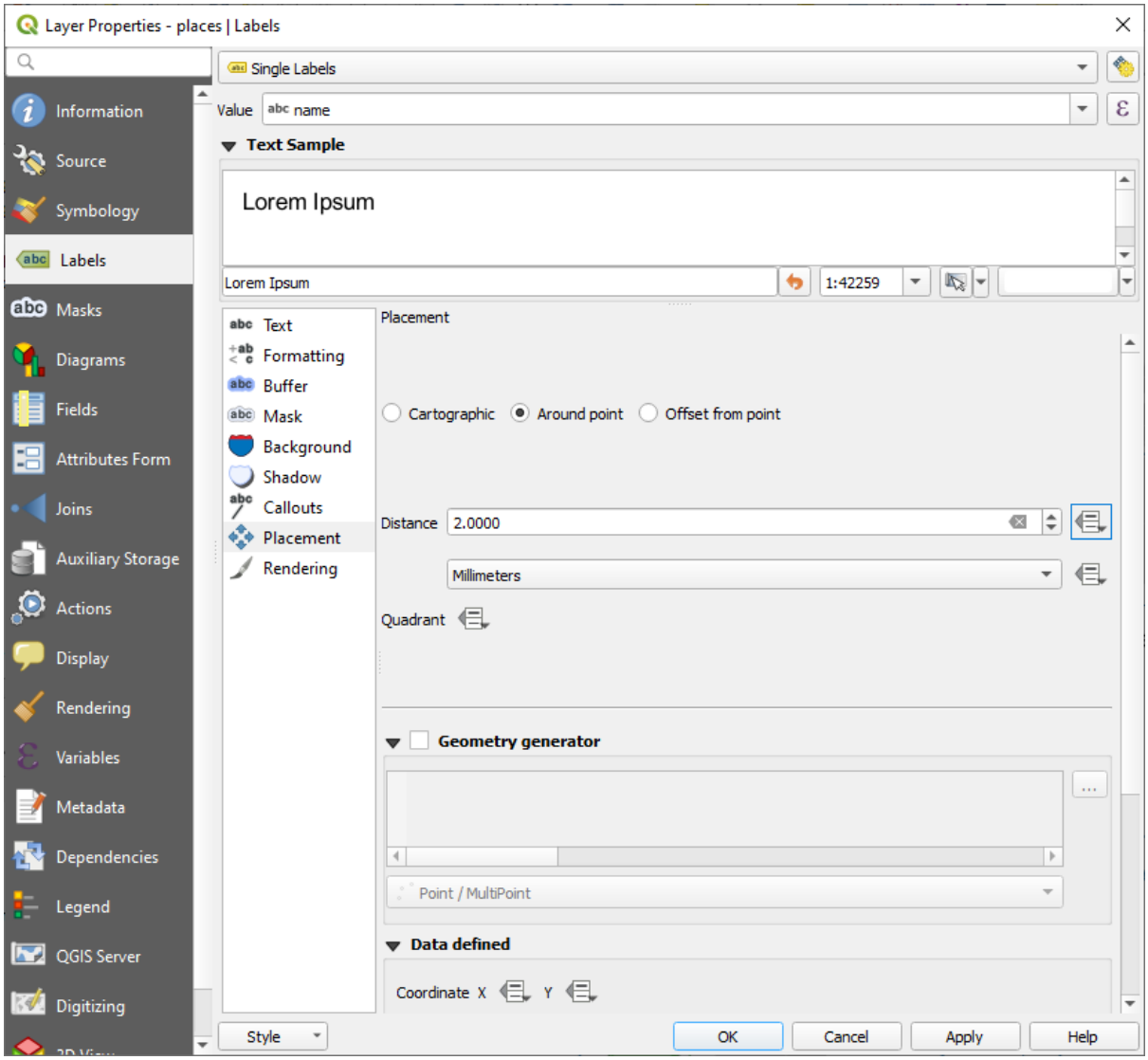
7. Click *Apply*

   You'll see that this adds a colored buffer or border to the place labels, making them easier to pick out on the map:



Now we can address the positioning of the labels in relation to their point markers.

8. Select *Placement* from the left-hand options list
9. Select *Around point* and change the value of *Distance* to `2.0` Millimeters:



10. Click *Apply*

    You'll see that the labels are no longer overlapping their point markers.

3.2.3 ??? Follow Along: Using Labels Instead of Layer Symbology

In many cases, the location of a point doesn't need to be very specific. For example, most of the points in the `places` layer refer to entire towns or suburbs, and the specific point associated with such features is not that specific on a large scale. In fact, giving a point that is too specific is often confusing for someone reading a map.

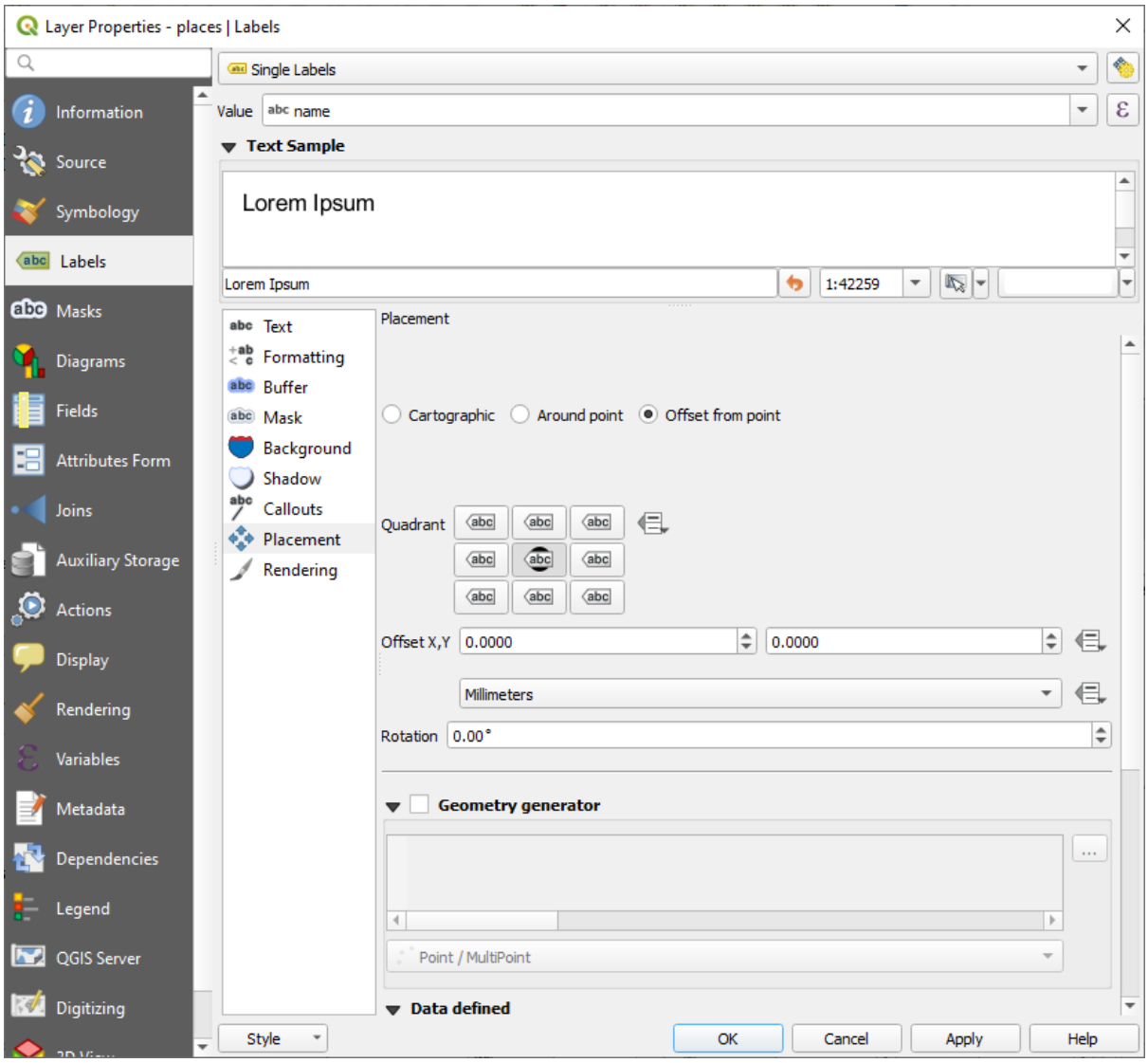
To name an example: on a map of the world, the point given for the European Union may be somewhere in Poland, for instance. To someone reading the map, seeing a point labeled *European Union* in Poland, it may seem that the capital of the European Union is therefore in Poland.

So, to prevent this kind of misunderstanding, it's often useful to deactivate the point symbols and replace them completely with labels.

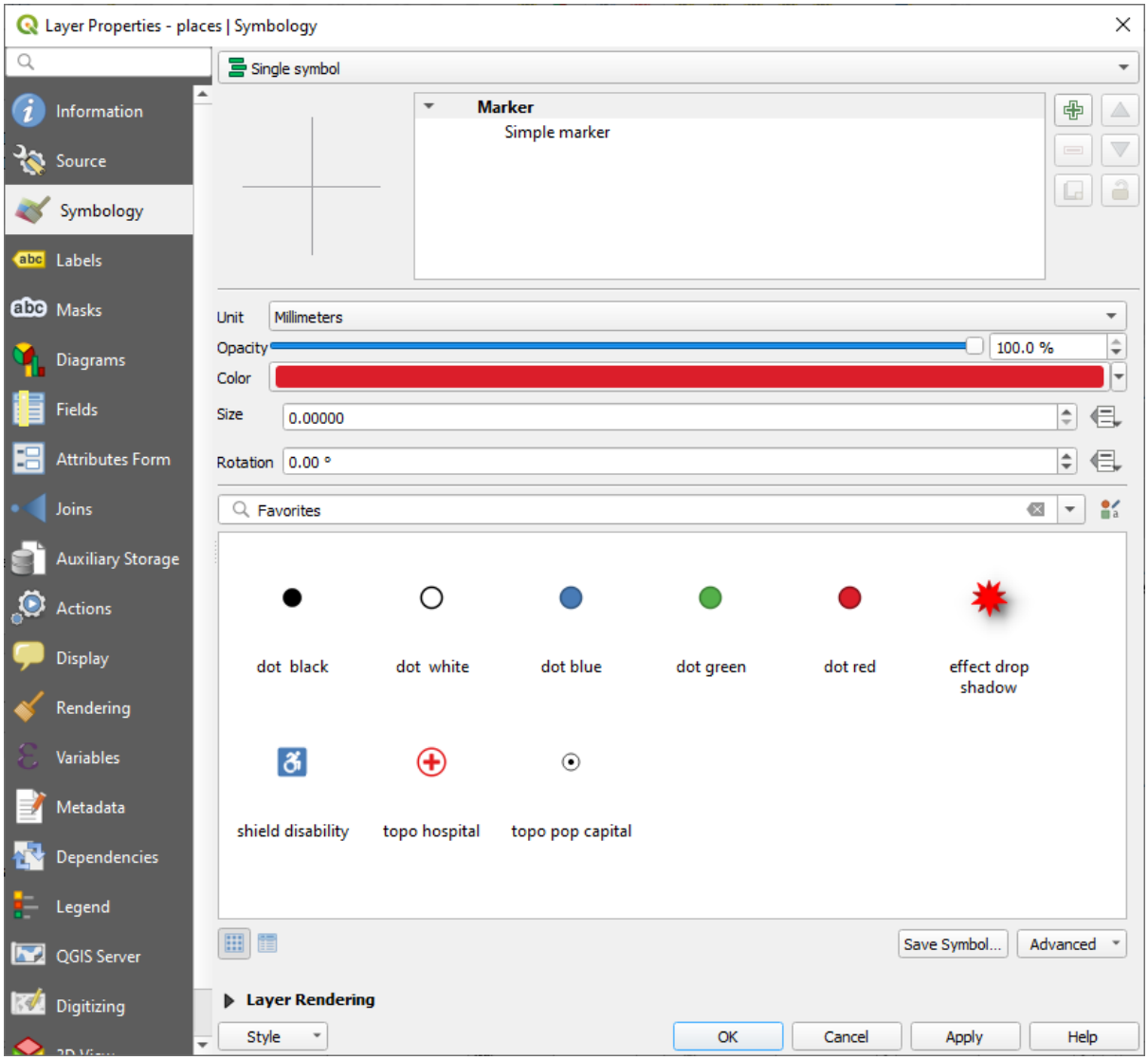
In QGIS, you can do this by changing the position of the labels to be rendered directly over the points they refer to.

1. Open the *Labels* tab of the *Layer Properties* dialog for the `places` layer
2. Select the *Placement* option from the options list
3. Click on the *Offset from point* button

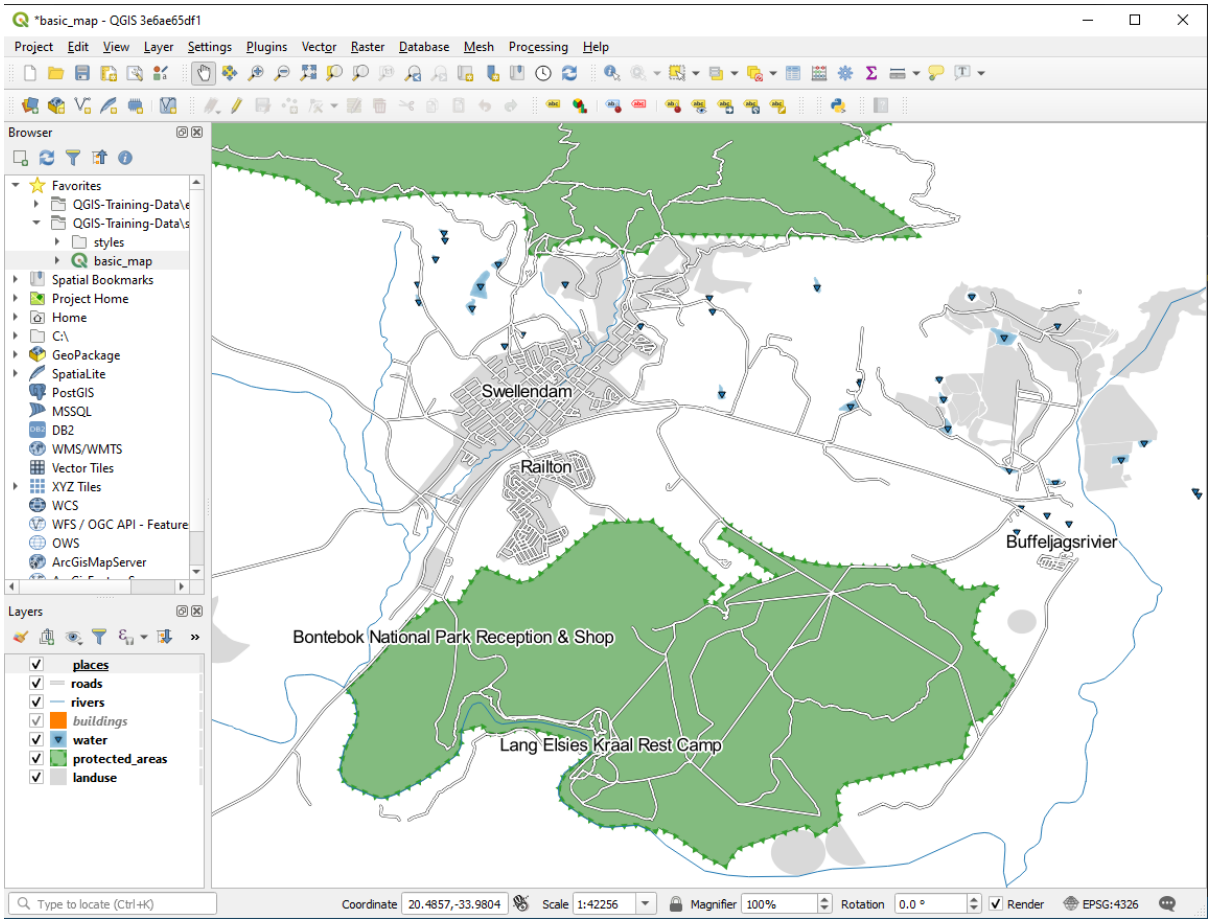
This will reveal the *Quadrant* options which you can use to set the position of the label in relation to the point marker. In this case, we want the label to be centered on the point, so choose the center quadrant:



4. Hide the point symbols by editing the layer *Symbology* as usual, and setting the size of the *Marker* size to `0.0`:



5. Click *Apply* and you'll see this result:



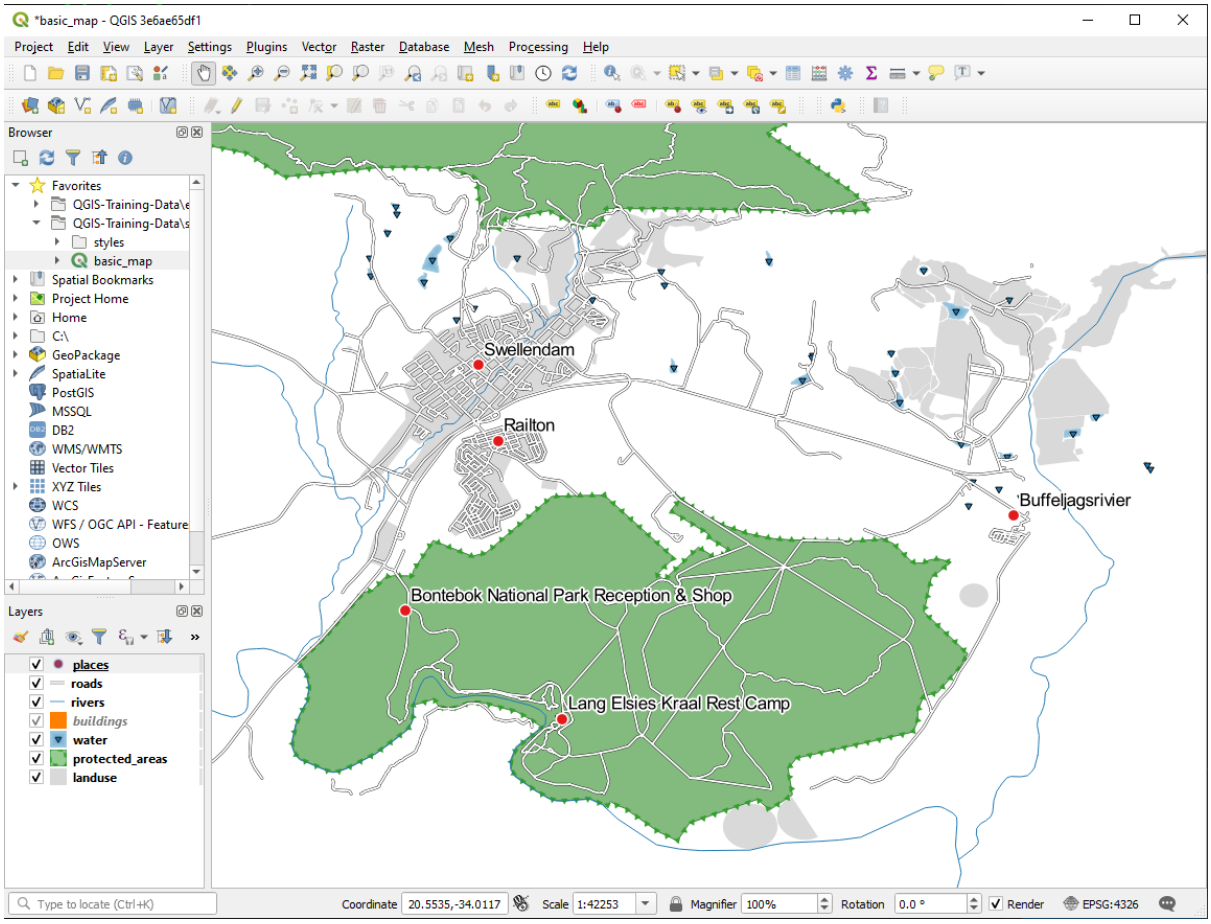
If you were to zoom out on the map, you would see that some of the labels disappear at larger scales to avoid overlapping. Sometimes this is what you want when dealing with datasets that have many points, but at other times you will lose useful information this way. There is another possibility for handling cases like this, which we'll cover in a later exercise in this lesson. For now, zoom out and click on the abc button in the toolbar and see what happens.

3.2.4 ??? Try Yourself: Customize the Labels

- Reset the label and symbol settings to have a point marker and a label offset of `2.0` Millimeters.

Answer

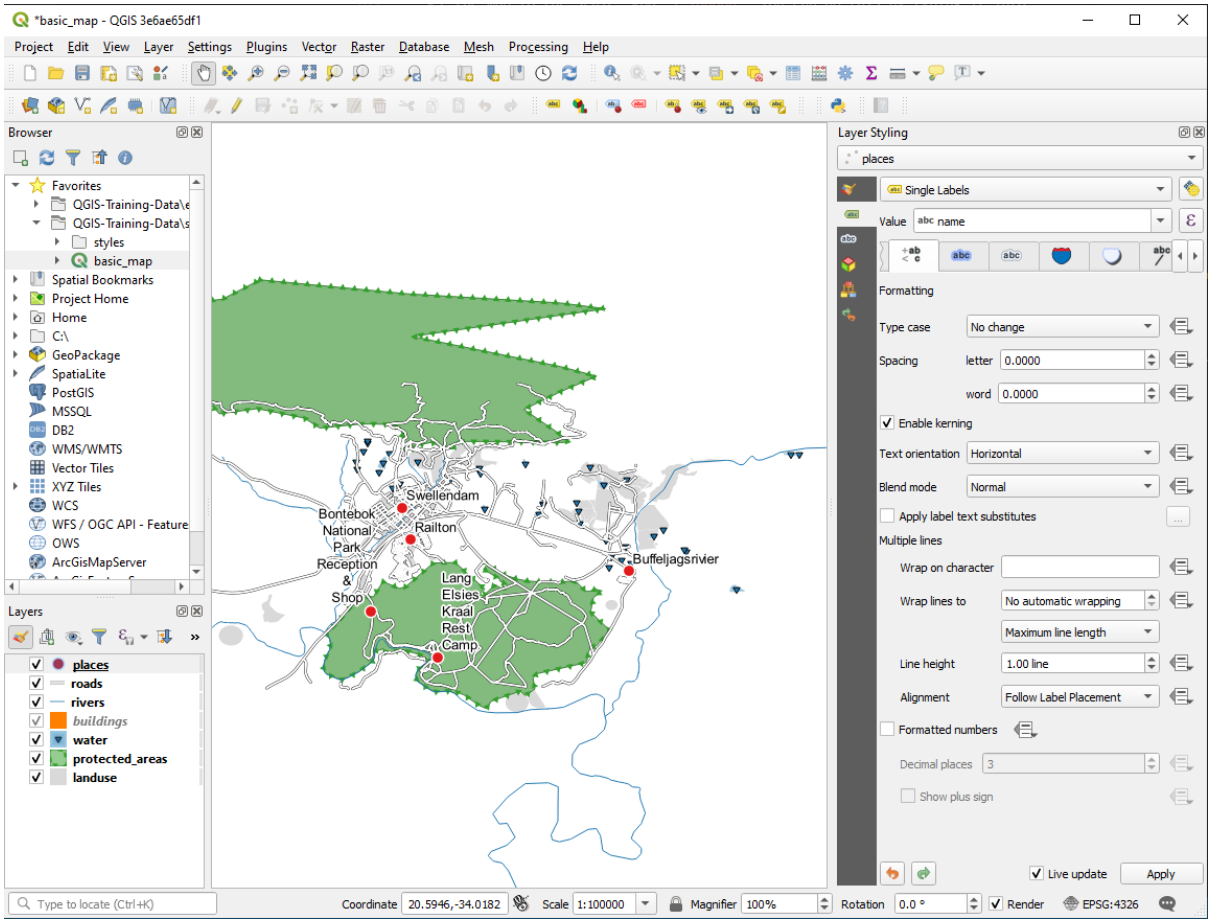
Your map should now show the marker points and the labels should be offset by 2mm. The style of the markers and labels should allow both to be clearly visible on the map:



- Set the map to the scale `1:100000`. You can do this by typing it into the *Scale* box in the *Status Bar*. Modify your labels to be suitable for viewing at this scale.

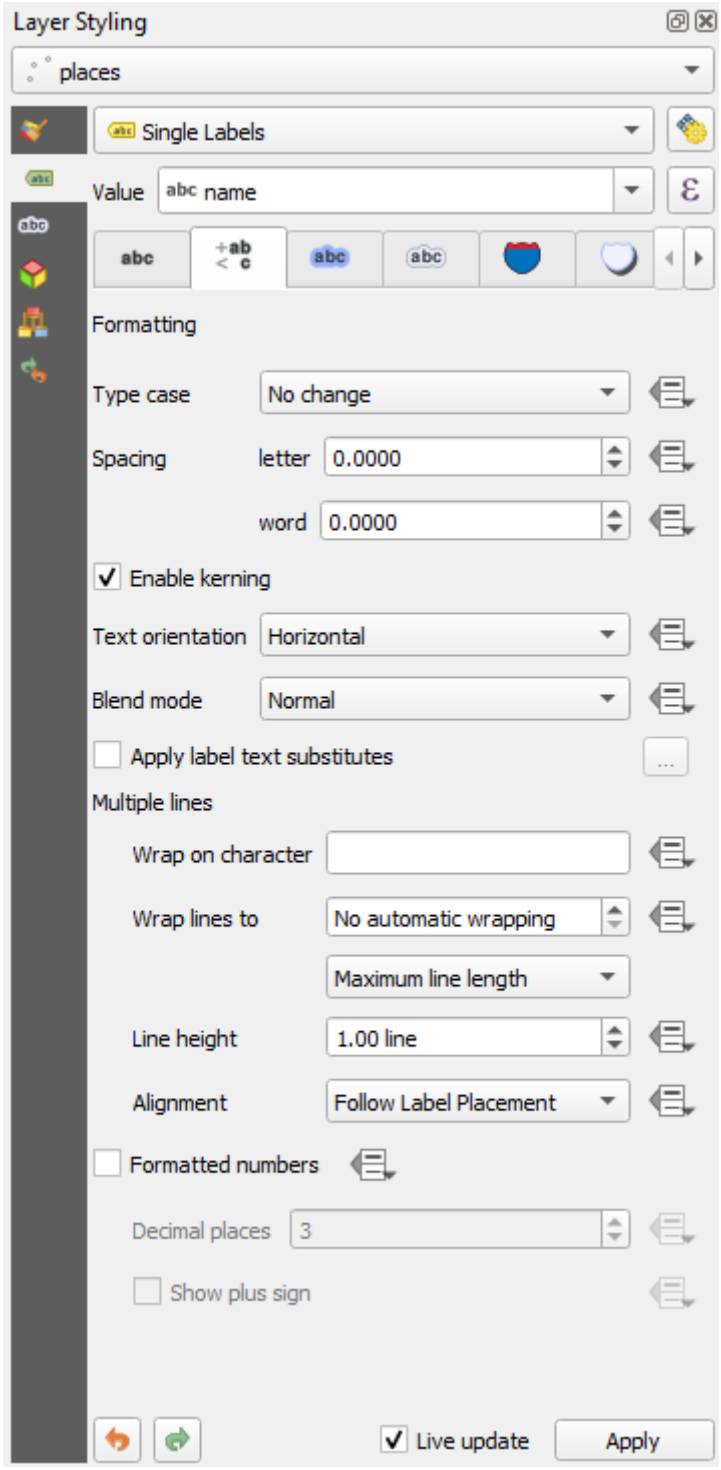
Answer

One possible solution has this final product:



To arrive at this result:

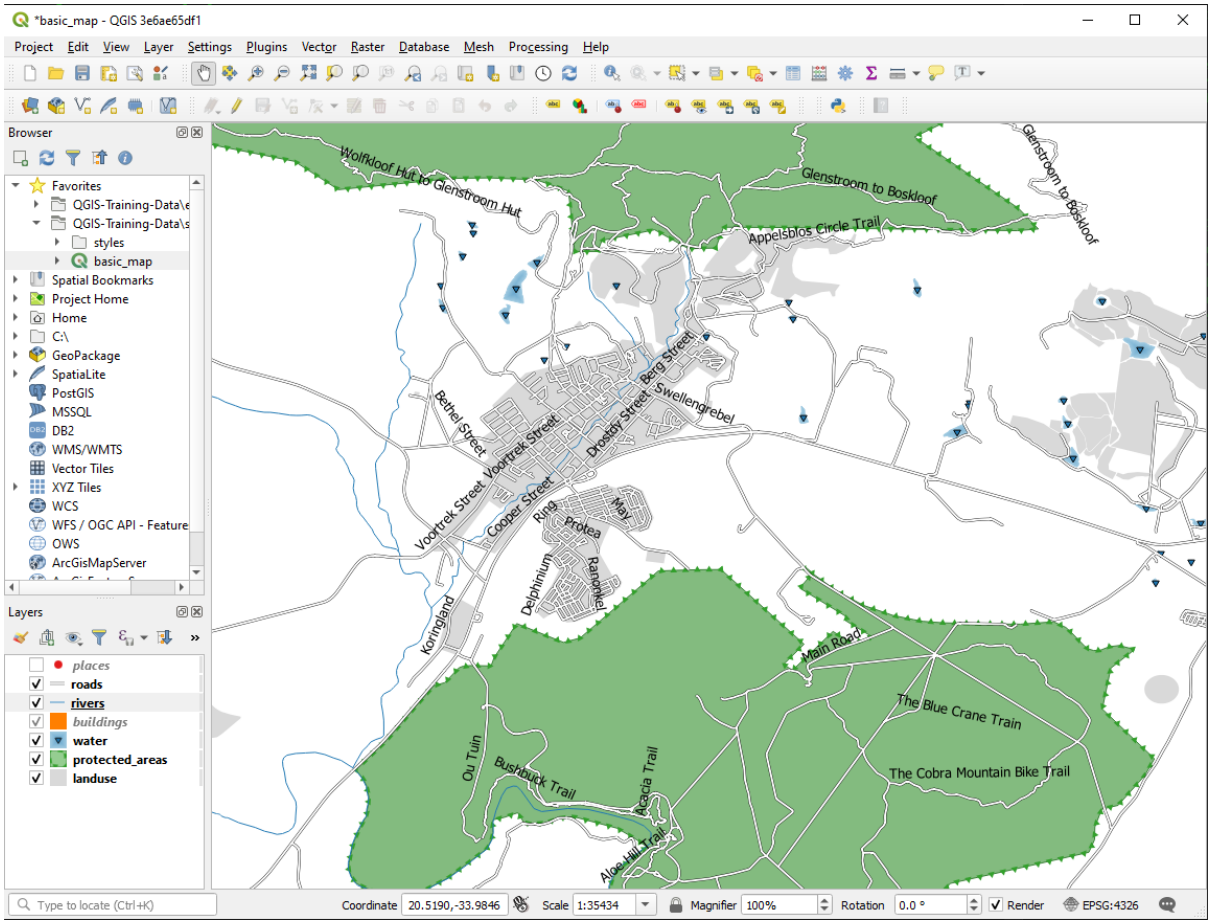
- Use a font size of `10`
- Use an around point placement distance of `1.5 mm`
- Use a marker size of `3.0 mm`
- In addition, this example uses the *Wrap on character* option:



- Enter a `space` in this field and click *Apply* to achieve the same effect. In our case, some of the place names are very long, resulting in names with multiple lines which is not very user friendly. You might find this setting to be more appropriate for your map.

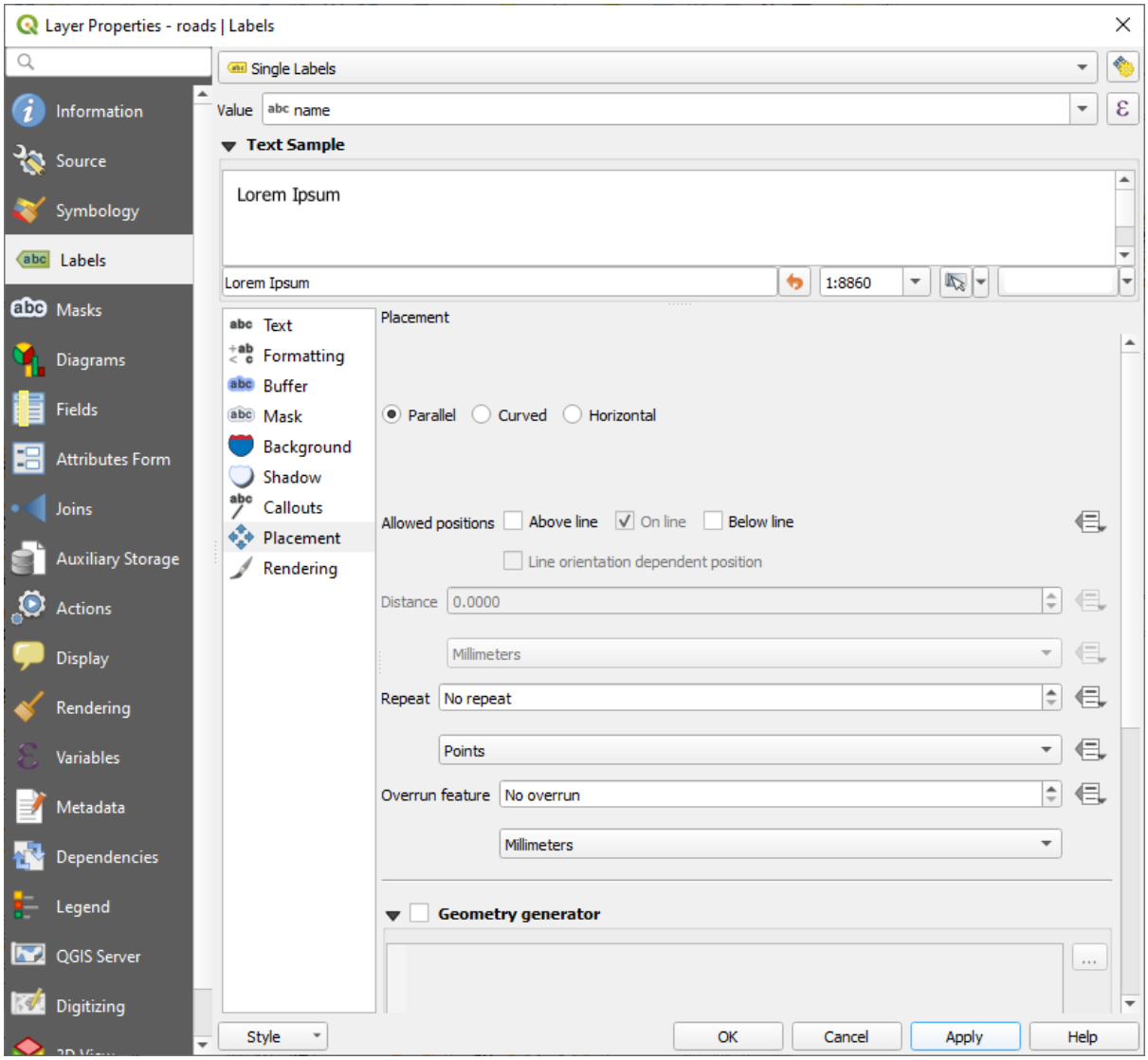
3.2.5 ??? Follow Along: Labeling Lines

Now that you know how labeling works, there's an additional problem. Points and polygons are easy to label, but what about lines? If you label them the same way as the points, your results would look like this:



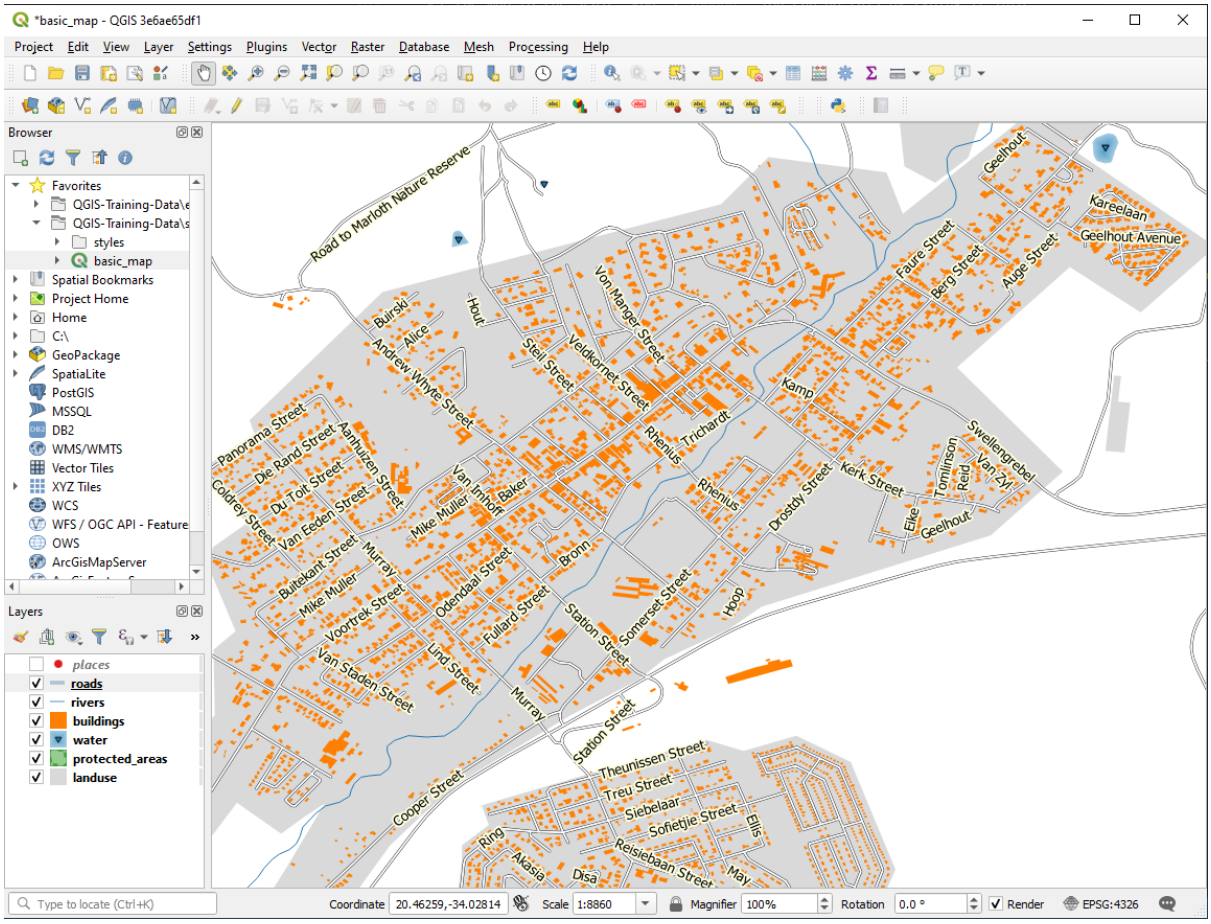
We will now reformat the `roads` layer labels so that they are easy to understand.

1. Hide the `places` layer so that it doesn't distract you
2. Activate *Single Labels* for the `roads` layer as you did above for `places`
3. Set the font *Size* to `10` so that you can see more labels
4. Zoom in on the Swellendam town area
5. In the *Labels* tab's *Placement* tab, choose the following settings:



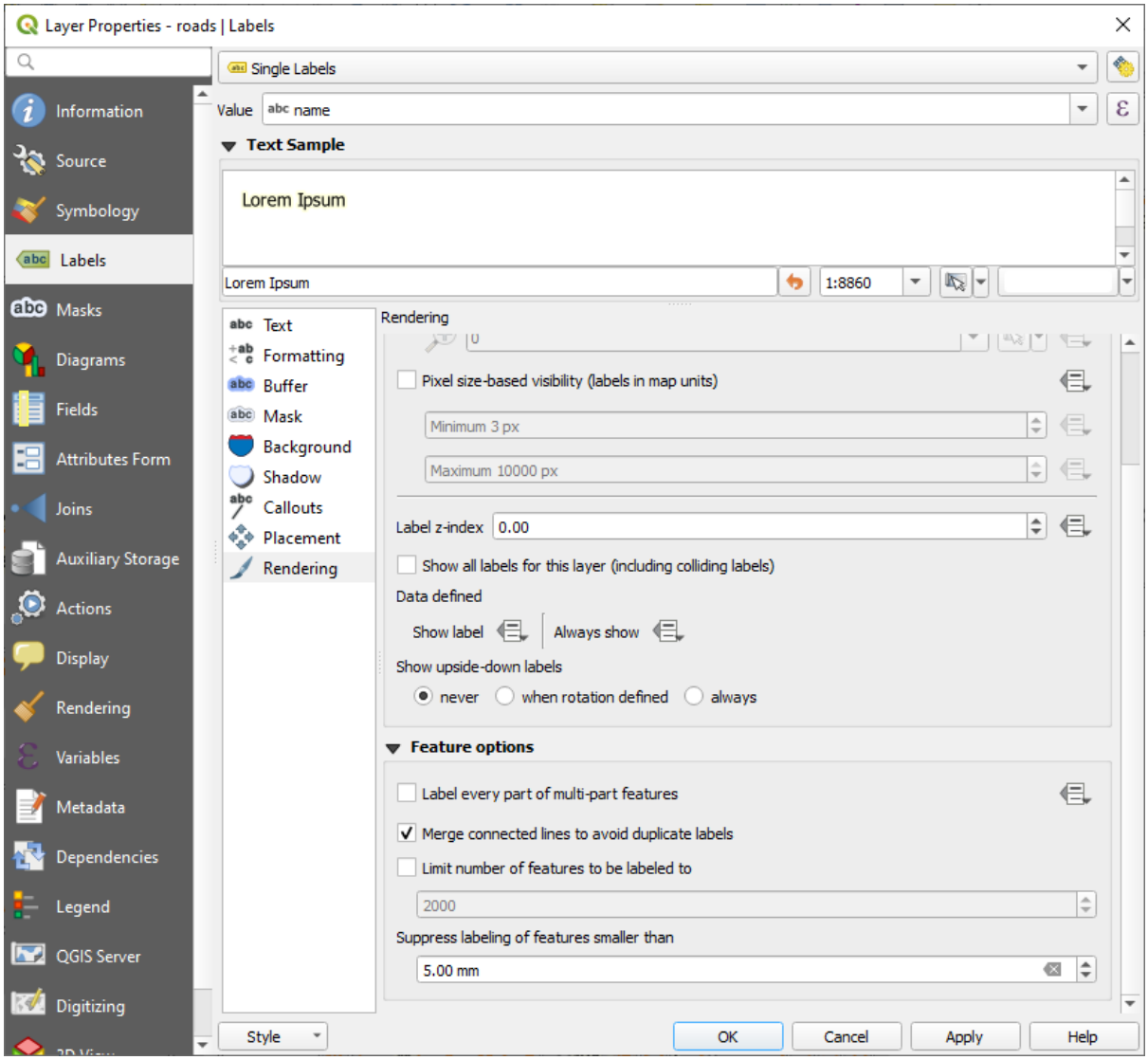
You'll probably find that the text styling has used default values and the labels are consequently very hard to read. Update the *Text* to use a dark-grey or black *Color* and the *Buffer* to use a light-yellow *Color*.

The map will look somewhat like this, depending on scale:



You'll see that some of the road names appear more than once and that's not always necessary. To prevent this from happening:

6. In the *Labels* tab of the *Layer Properties* dialog, choose the *Rendering* option and select *Merge connected lines to avoid duplicate labels* as shown:



7. Click *OK*

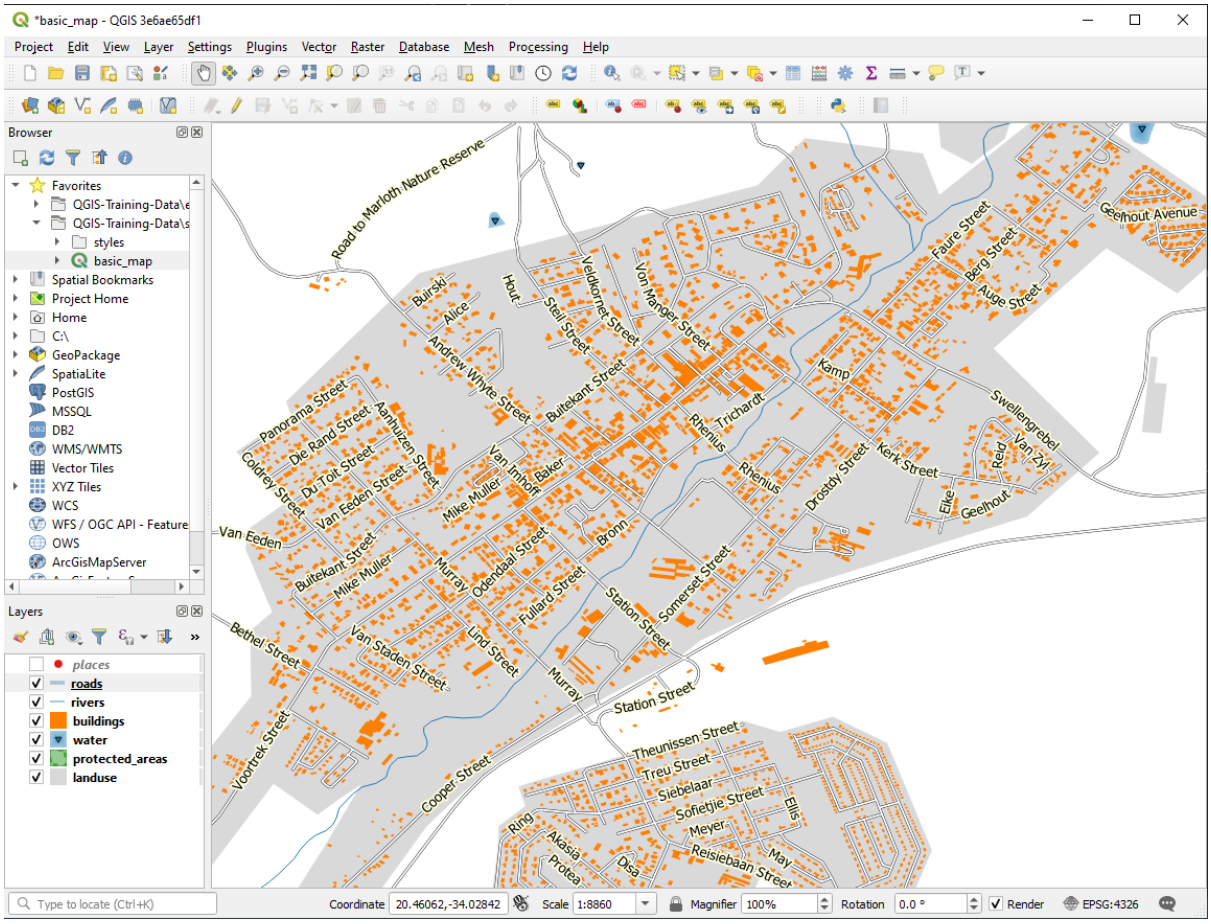
   Another useful function is to prevent labels being drawn for features too short to be of notice.

8. In the same *Rendering* panel, set the value of *Suppress labeling of features smaller than …* to `5.00 mm` and note the results when you click *Apply*

   Try out different *Placement* settings as well. As we've seen before, the *Horizontal* option is not a good idea in this case, so let's try the *Curved* option instead.

9. Select the *Curved* option in the *Placement* panel of the *Labels* tab

Here's the result:

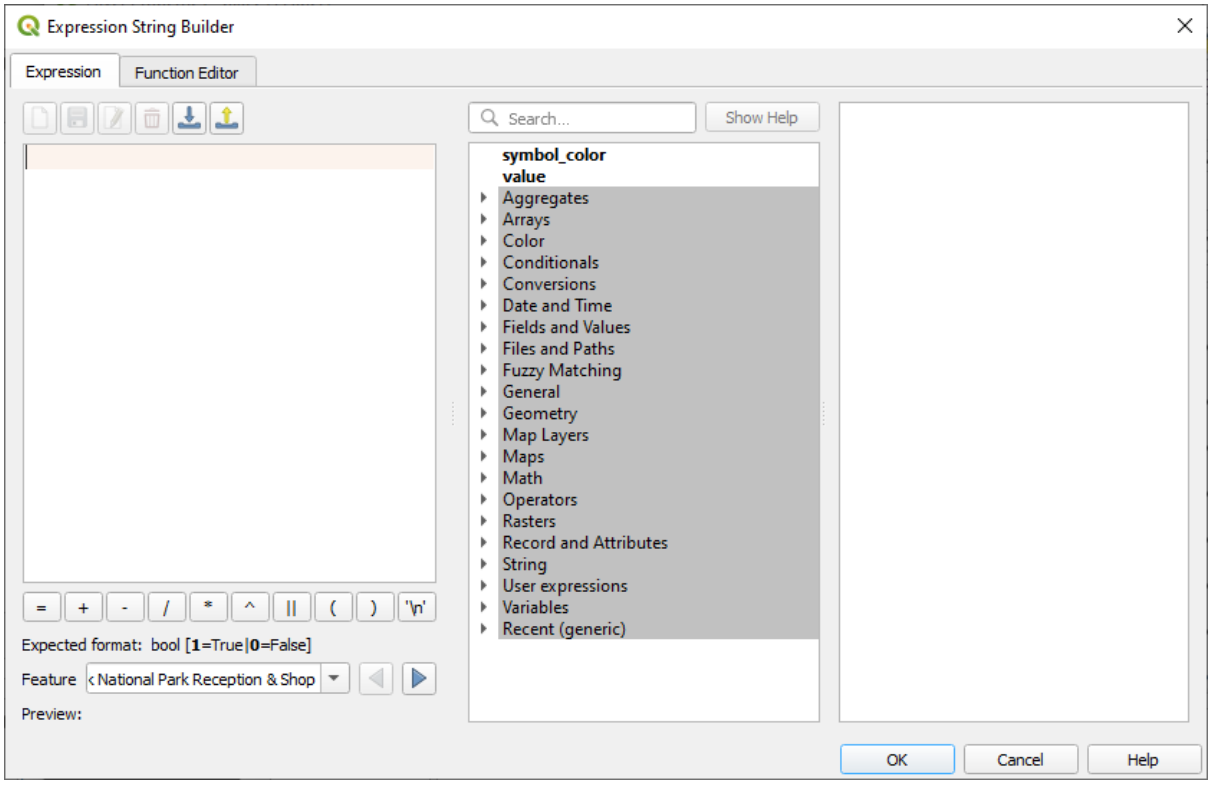


As you can see, this hides some labels that were previously visible, because of the difficulty of making some of them follow twisting street lines while still being legible. It makes other labels much more useful since they track the roads rather than float in space between them. You can decide which of these options to use, depending on what you think seems more useful or what looks better.

3.2.6 ??? Follow Along: Data Defined Settings

1. Deactivate labeling for the `roads` layer
2. Reactivate labeling for the `places` layer
3. Open the attribute table for `places` via the button

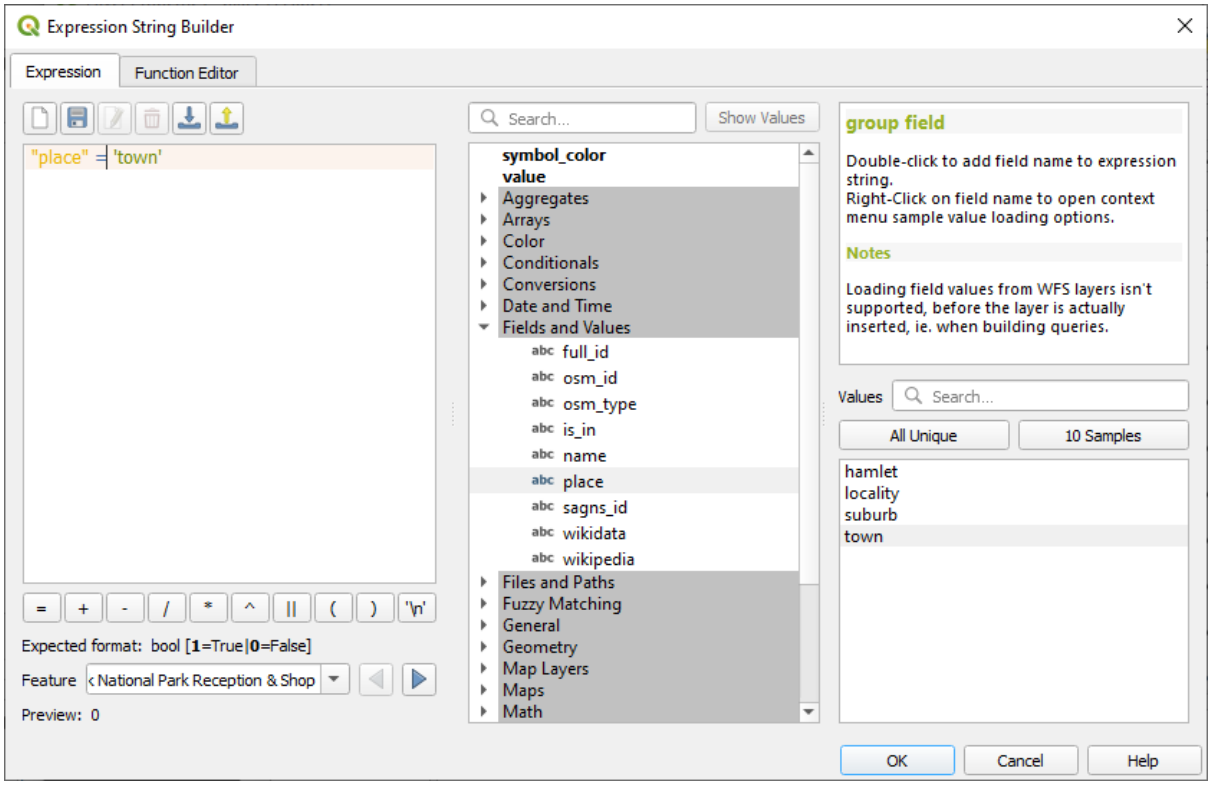
   It has one *field* which is of interest to us now: `place` which defines the type of urban area for each *record*. We can use this data to influence the label styles.
4. Navigate to the *Text* panel in the `places` *Labels* panel
5. Click the button next to the Italic text button beneath *Style* and select *Edit…* to open the *Expression String Builder*:



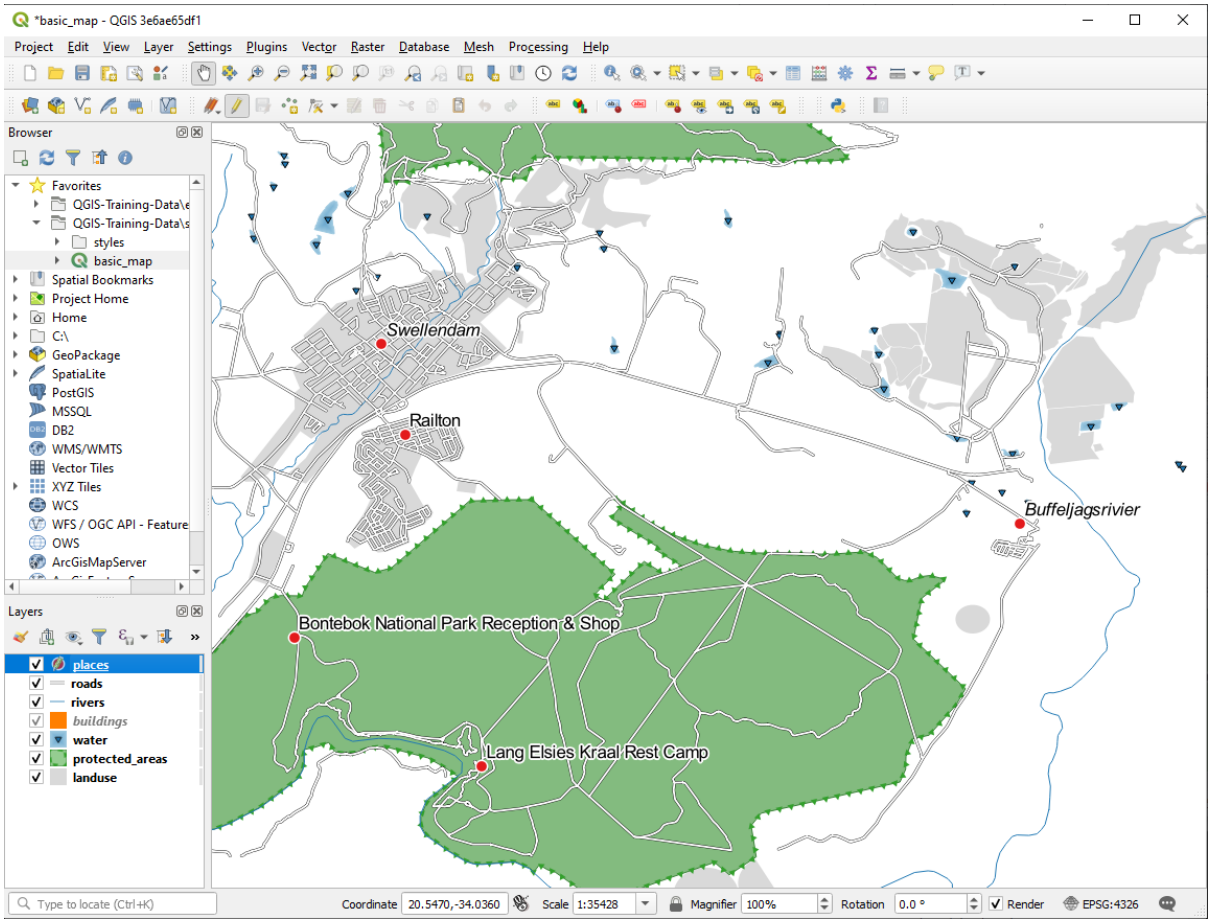
6. Under *Fields and Values*, double click on `place` and then click *All Unique*. This will list all unique values of the `place` field of this layer. Add a = in the text editor and then double click on `town`.

   Alternatively, you can type: `"place" = 'town'` directly in the text editor.

7. Click *OK* twice:



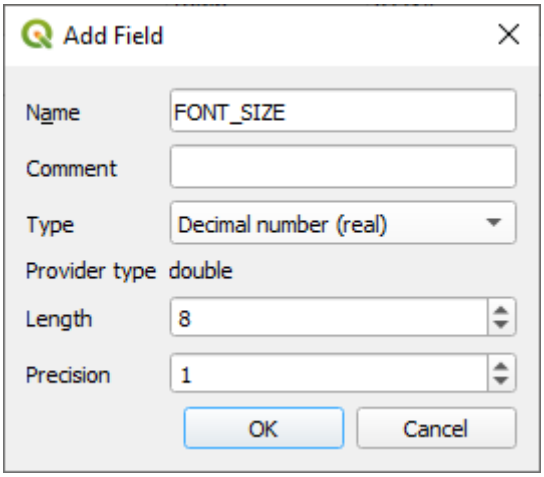
Notice that the labels for all places whose `place` field matches `town` are displayed in italics.



3.2.7 ??? Try Yourself: Using Data Defined Settings

Nota: We're jumping ahead a bit here to demonstrate some advanced labeling settings. At the advanced level, it's assumed that you'll know what the following means. If you don't, feel free to leave out this section and come back later when you've covered the requisite materials.

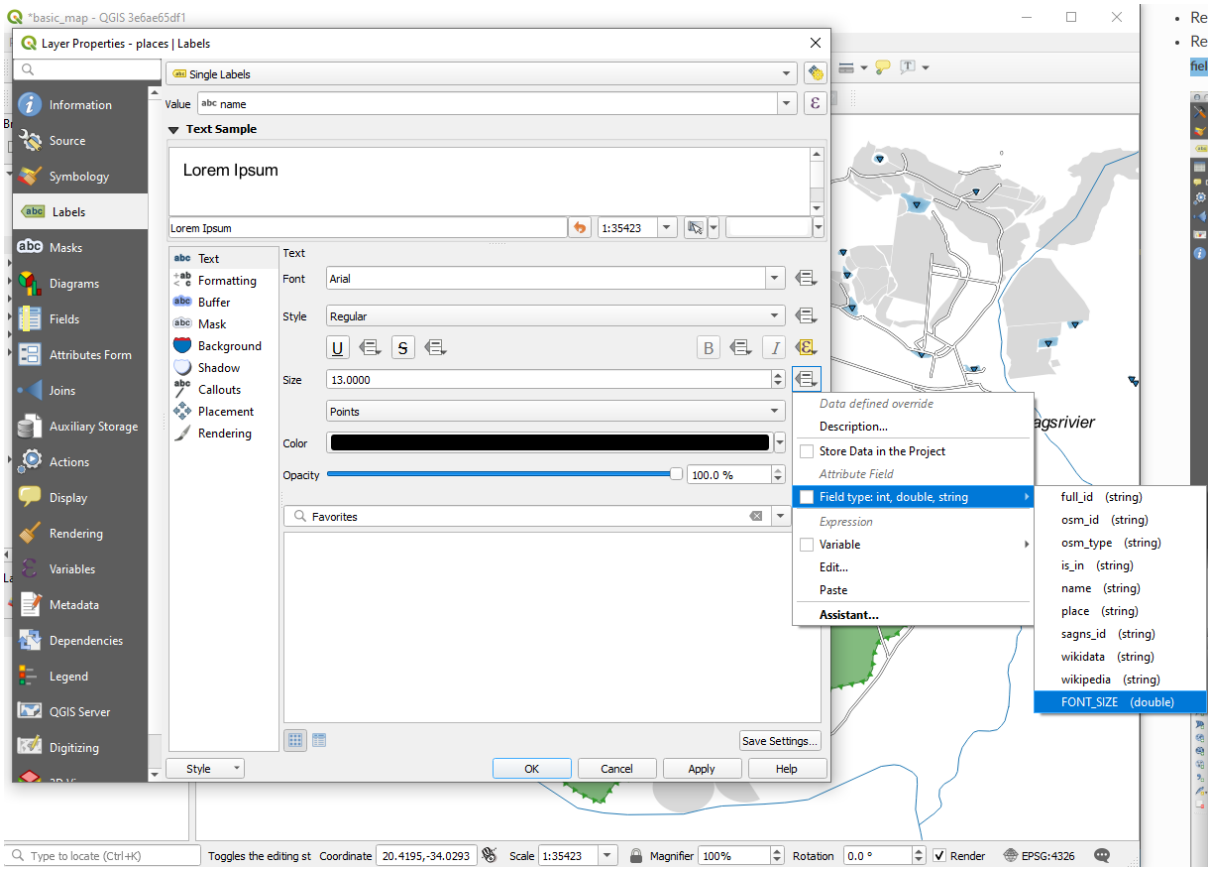
1. Open the Attribute Table for `places`
2. Enter edit mode by clicking the button
3. Add a new column with the button
4. Configure it like this:



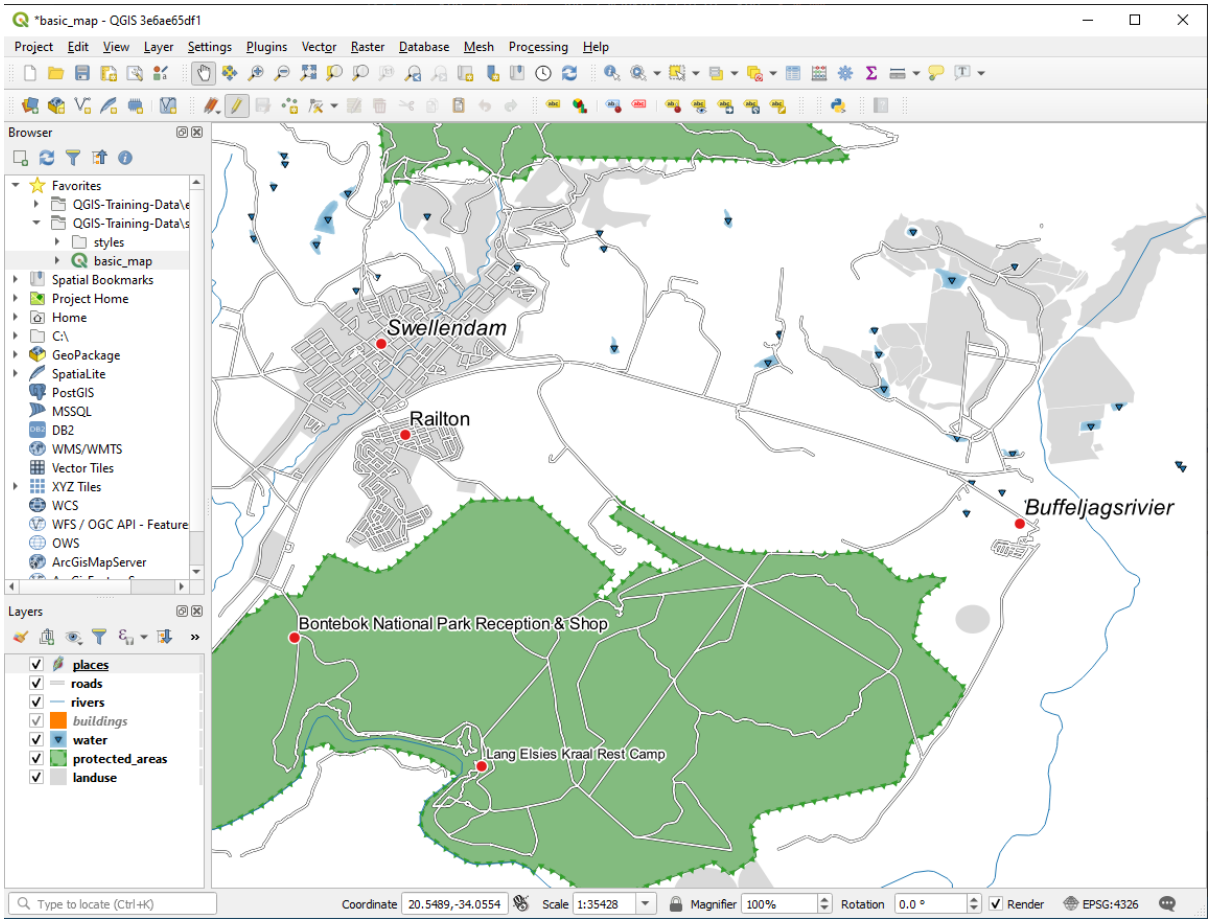
5. Use this to set custom font sizes for each different type of place (each key in the `place` field)

Answer

1. Still in edit mode, set the `FONT_SIZE` values to whatever you prefer. The example uses `16` for towns, `14` for suburbs, `12` for localities, and `10` for hamlets.
2. Remember to save changes and exit edit mode
3. Return to the *Text* formatting options for the `places` layer and select `FONT_SIZE` in the *Attribute field* of the font size data defined override dropdown:



Your results, if using the above values, should be this:



3.2.8 ??? Further Possibilities With Labeling

We can't cover every option in this course, but be aware that the *Label* tab has many other useful functions. You can set scale-based rendering, alter the rendering priority for labels in a layer, and set every label option using layer attributes. You can even set the rotation, XY position, and other properties of a label (if you have attribute fields allocated for the purpose), then edit these properties using the tools adjacent to the main *Layer Labeling Options* button:

(These tools will be active if the required attribute fields exist and you are in edit mode.)

Feel free to explore more possibilities of the labeling system.

3.2.9 In Conclusion

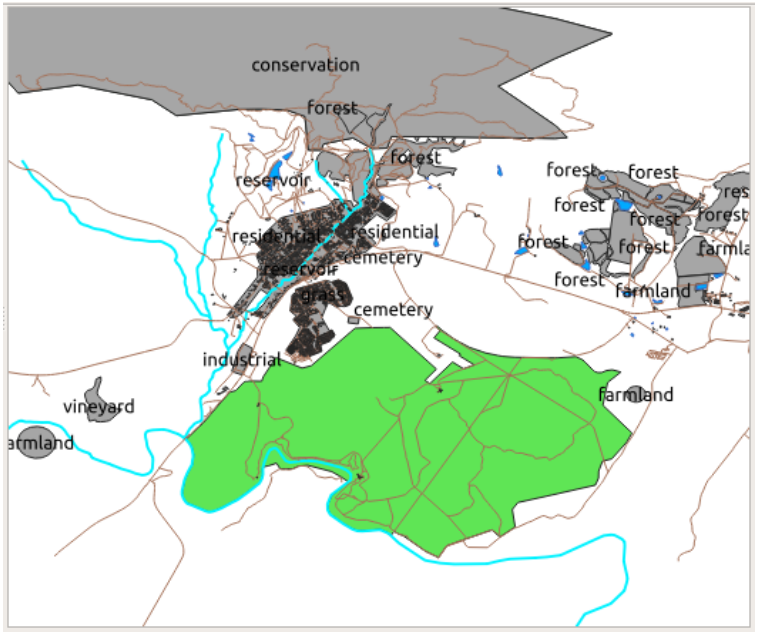
You've learned how to use layer attributes to create dynamic labels. This can make your map a lot more informative and stylish!

3.2.10 What's Next?

Now that you know how attributes can make a visual difference for your map, how about using them to change the symbology of objects themselves? That's the topic for the next lesson!

3.3 Lesson: Classification

Labels are a good way to communicate information such as the names of individual places, but they can't be used for everything. For example, let us say that someone wants to know what each `landuse` area is used for. Using labels, you would get this:

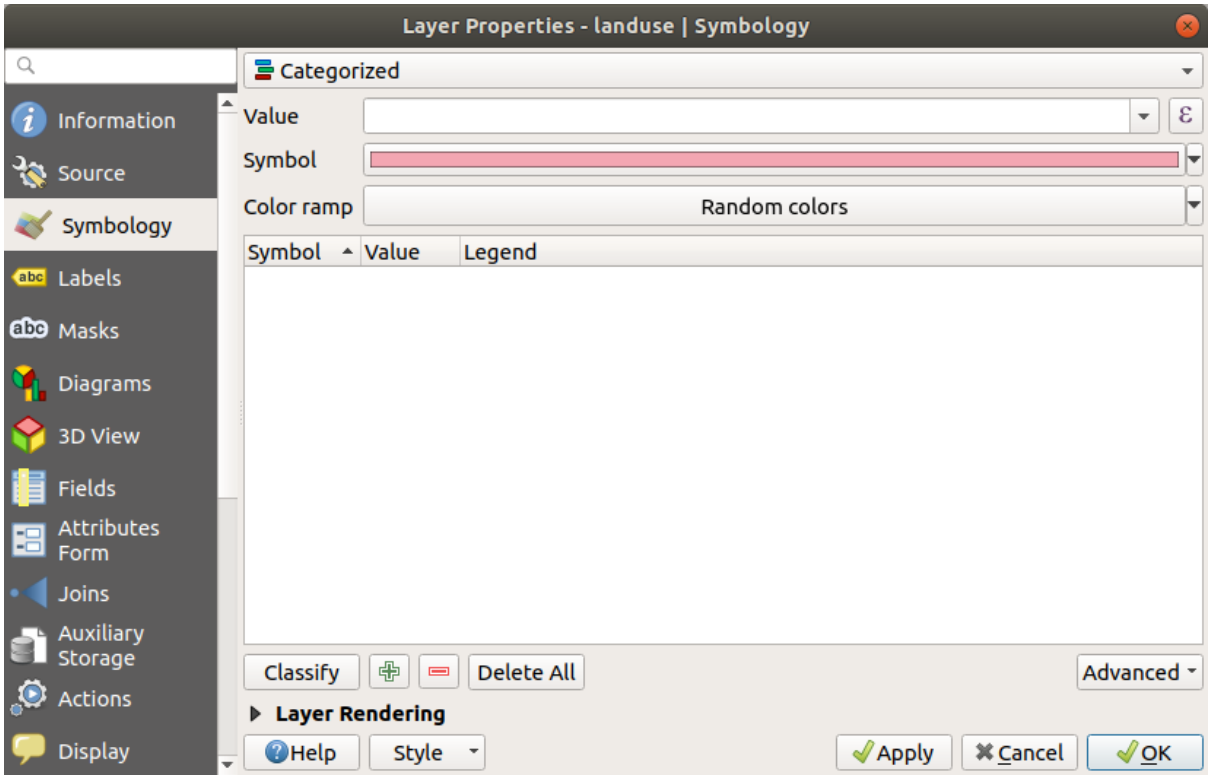


This makes the map's labeling difficult to read and even overwhelming if there are numerous different landuse areas on the map.

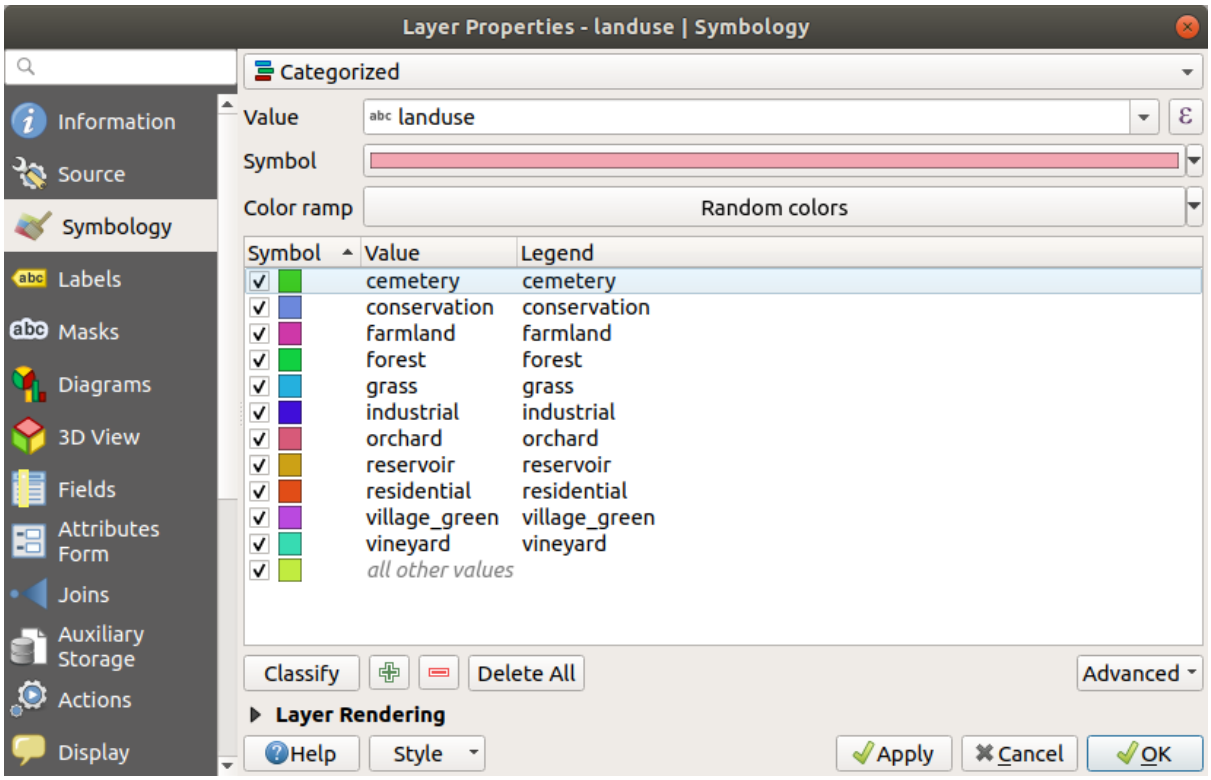
The goal for this lesson: To learn how to classify vector data effectively.

3.3.1 ??? Follow Along: Classifying Nominal Data

1. Open the *Layer Properties* dialog for the `landuse` layer
2. Go to the *Symbology* tab
3. Click on the dropdown that says *Single Symbol* and change it to *Categorized*:

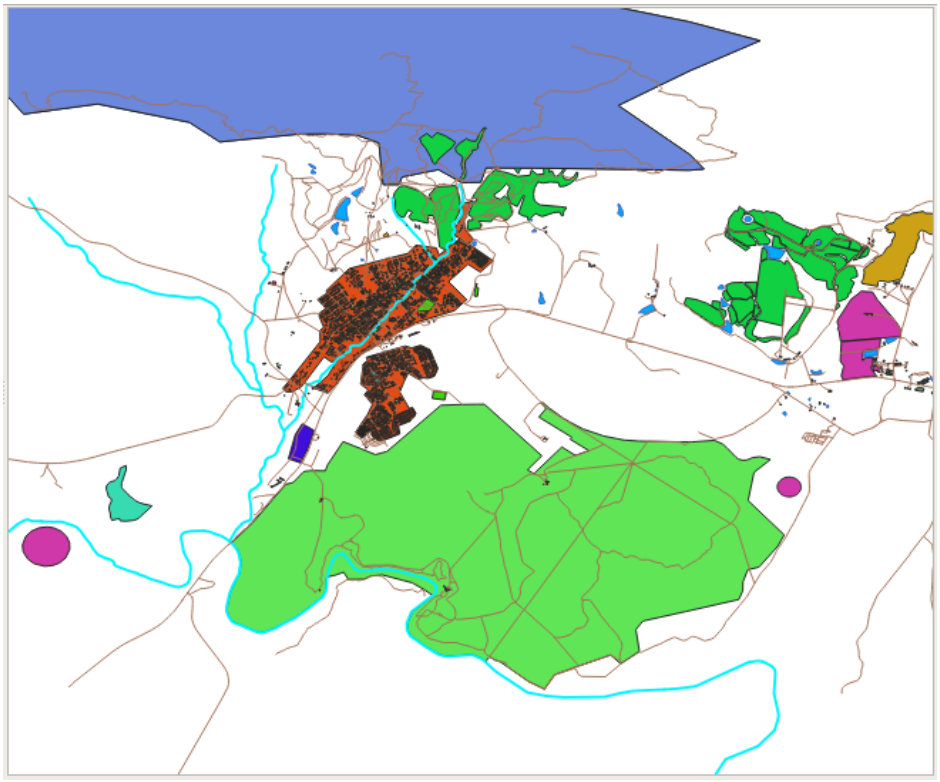


4. In the new panel, change the *Value* to `landuse` and the *Color ramp* to *Random colors*
5. Click the button labeled *Classify*

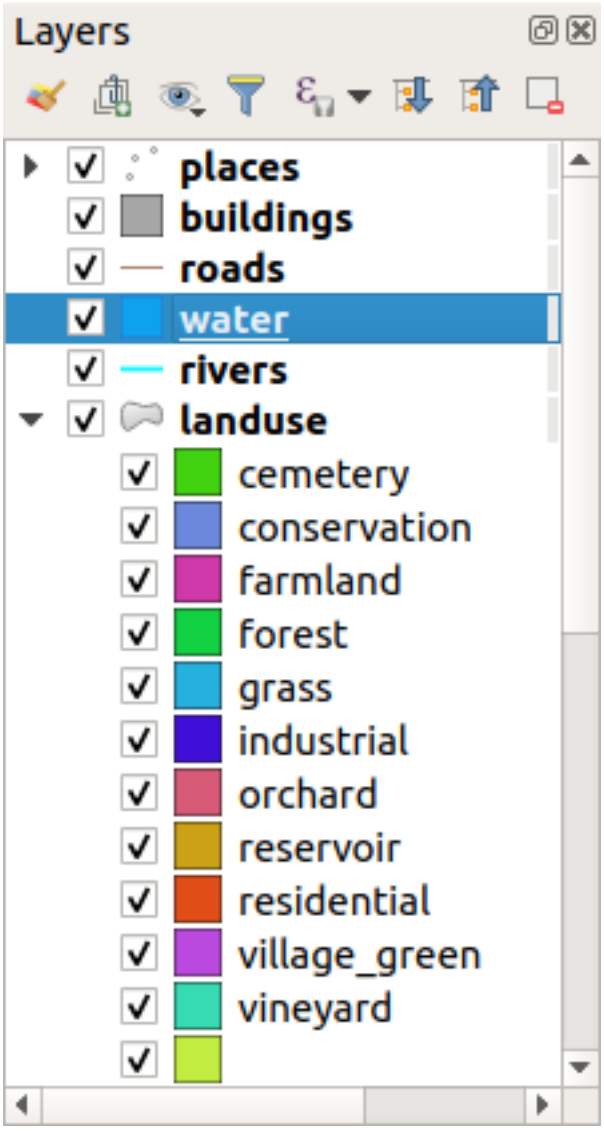


6. Click *OK*

   You'll see something like this:

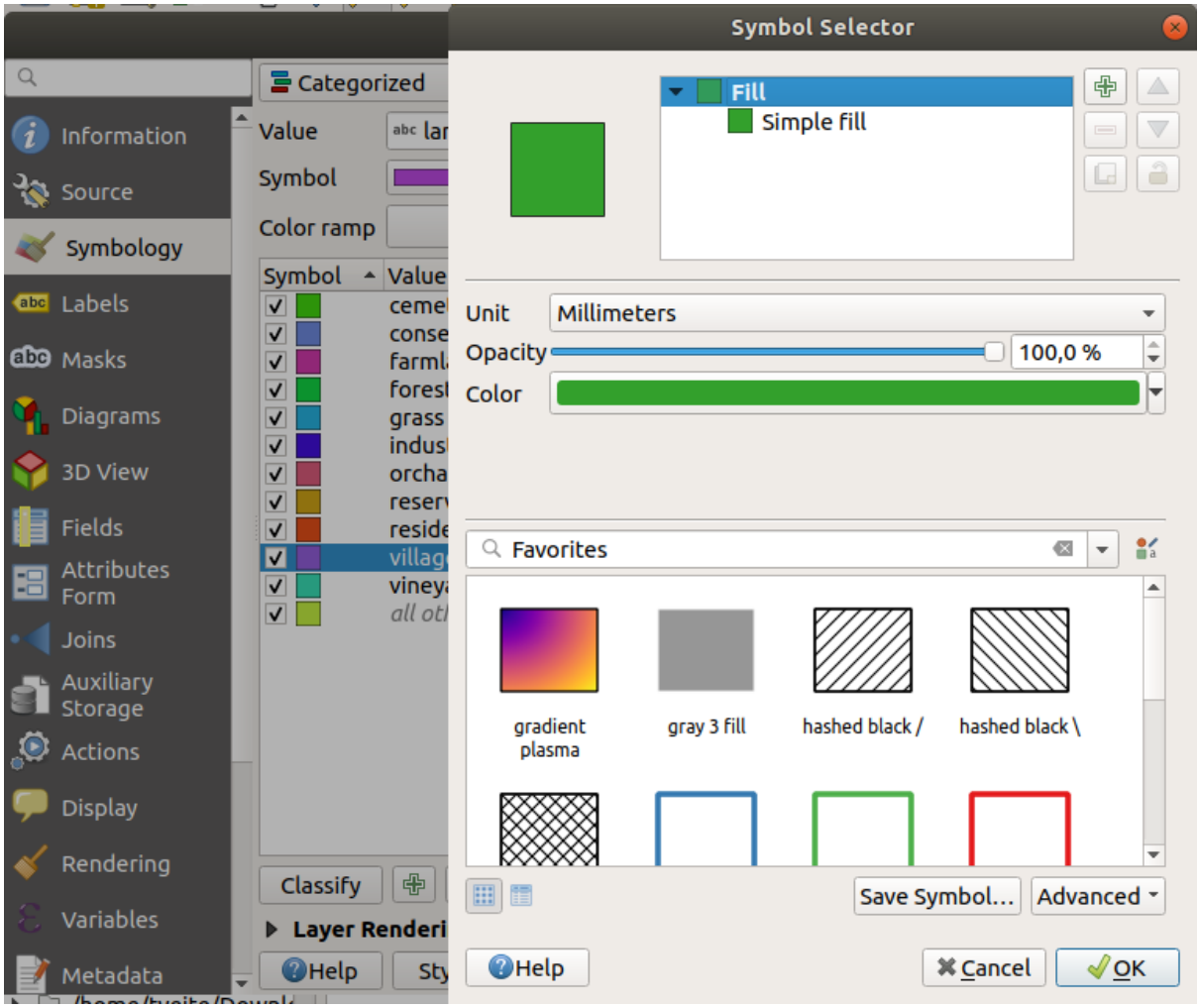


7. Click the arrow (or plus sign) next to `landuse` in the *Layers* panel, you'll see the categories explained:

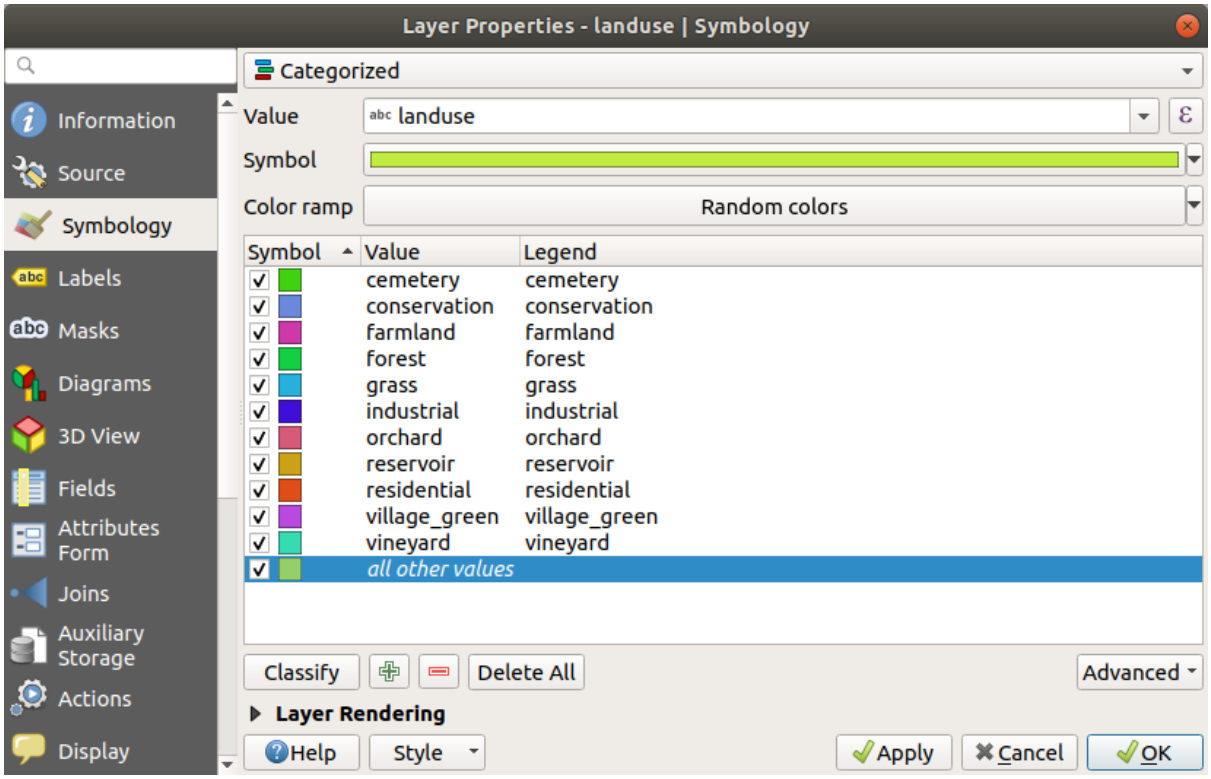


Now our landuse polygons are colored and are classified so that areas with the same land use are the same color.

8. If you wish to, you can change the symbol of each landuse area by double-clicking the relevant color block in the *Layers* panel or in the *Layer Properties* dialog:



Notice that there is one category that's empty:



This empty category is used to color any objects which do not have a landuse value defined or which have a *NULL* value. It can be useful to keep this empty category so that areas with a *NULL* value are still represented on the map. You may like to change the color to more obviously represent a blank or *NULL* value.

Remember to save your map now so that you don't lose all your hard-earned changes!

3.3.2 ??? Try Yourself: More Classification

Use the knowledge you gained above to classify the `buildings` layer. Set the categorisation against the `building` field and use the *Spectral* color ramp.

Nota: Remember to zoom into an urban area to see the results.

3.3.3 ??? Follow Along: Ratio Classification

There are four types of classification: *nominal*, *ordinal*, *interval* and *ratio*.

In **nominal** classification, the categories that objects are classified into are name-based; they have no order. For example: town names, district codes, etc. Symbols that are used for nominal data should not imply any order or magnitude.

- For points, we can use symbols of different shape.
- For polygons, we can use different types of hatching or different colours (avoid mixing light and dark colours).
- For lines, we can use different dash patterns, different colours (avoid mixing light and dark colours) and different symbols along the lines.

In **ordinal** classification, the categories are arranged in a certain order. For example, world cities are given a rank depending on their importance for world trade, travel, culture, etc. Symbols that are used for ordinal data should imply order, but not magnitude.

- For points, we can use symbols with light to dark colours.
- For polygons, we can use graduated colours (light to dark).
- For lines, we can use graduated colours (light to dark).

In **interval** classification, the numbers are on a scale with positive, negative and zero values. For example: height above/below sea level, temperature in degrees Celsius. Symbols that are used for interval data should imply order and magnitude.

- For points, we can use symbols with varying size (small to big).
- For polygons, we can use graduated colours (light to dark) or add diagrams of varying size.
- For lines, we can use thickness (thin to thick).

In **ratio** classification, the numbers are on a scale with only positive and zero values. For example: temperature above absolute zero (0 degrees Kelvin), distance from a point, the average amount of traffic on a given street per month, etc. Symbols that are used for ratio data should imply order and magnitude.

- For points, we can use symbols with varying size (small to big).
- For polygons, we can use graduated colours (light to dark) or add diagrams of varying size.
- For lines, we can use thickness (thin to thick).

In the example above, we used nominal classification to color each record in the `landuse` layer based on its `landuse` attribute. Now we will use ratio classification to classify the records by area.

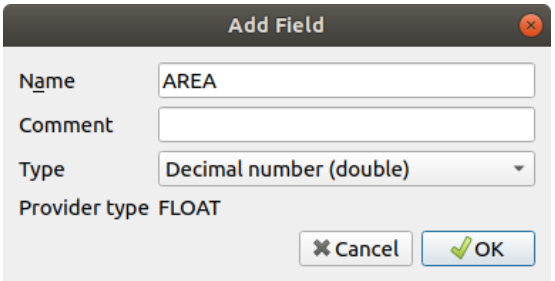
We are going to reclassify the layer, so existing classes will be lost if not saved. To store the current classification:

1. Open the layer's properties dialog
2. Click the *Save Style* ... button in the *Style* drop-down menu.
3. Select *Rename Current...*, enter `land usage` and press *OK*.

   The categories and their symbols are now saved in the layer's properties.
4. Click now on the *Add...* entry of the *Style* drop-down menu and create a new style named `ratio`. This will store the new classification.
5. Close the *Layer Properties* dialog

We want to classify the landuse areas by size, but there is a problem: they don't have a size field, so we'll have to make one.

1. Open the Attributes Table for the `landuse` layer.
2. Enter edit mode by clicking the Toggle editing button
3. Add a new column of decimal type, called `AREA`, using the New field button:



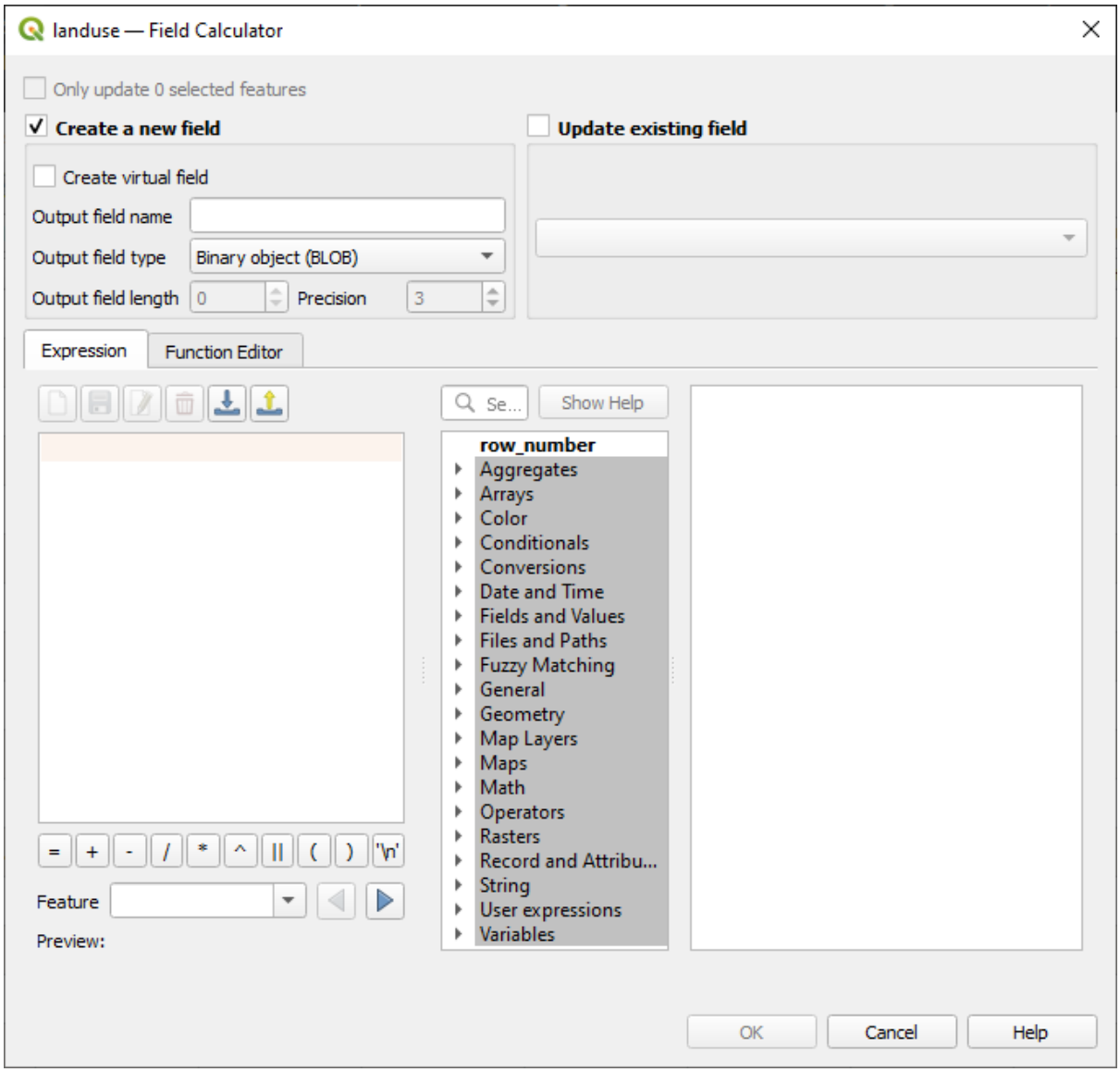
4. Click *OK*

   The new field will be added (at the far right of the table; you may need to scroll horizontally to see it). However, at the moment it is not populated, it just has a lot of *NULL* values.

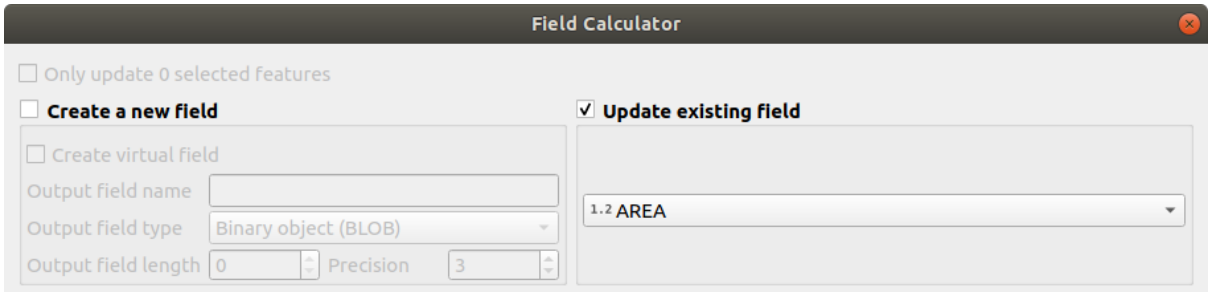
   To solve this problem, we will need to calculate the areas.

1. Open the field calculator with the button.

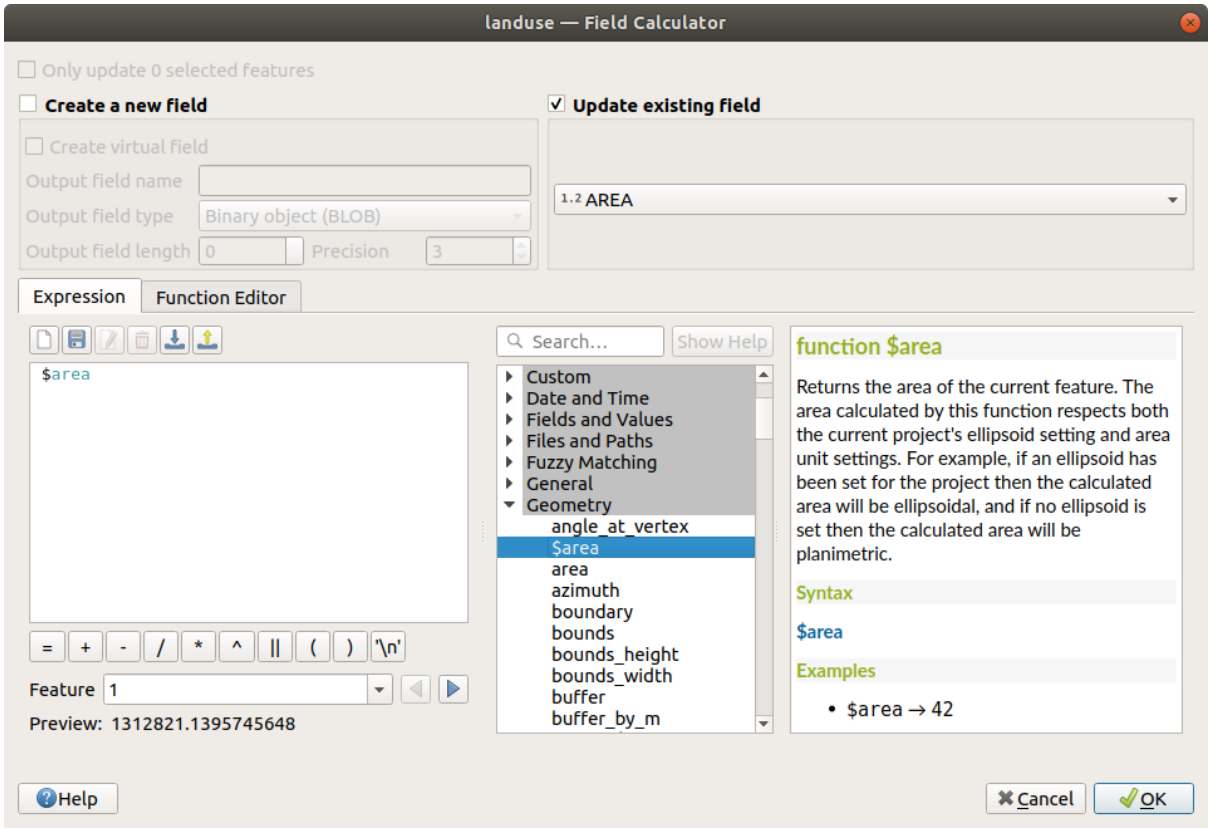
   You will get this dialog:



2. Check the *Update existing fields*
3. Select *AREA* in the fields drop-down menu



4. Under the *Expression* tab, expand the *Geometry* functions group in the list and find *$area*
5. Double-click on it so that it appears in the *Expression* field



6. Click *OK*
7. Scroll to the `AREA` field in the attribute table and you will notice that it is populated with values (you may need to click the column header to refresh the data).

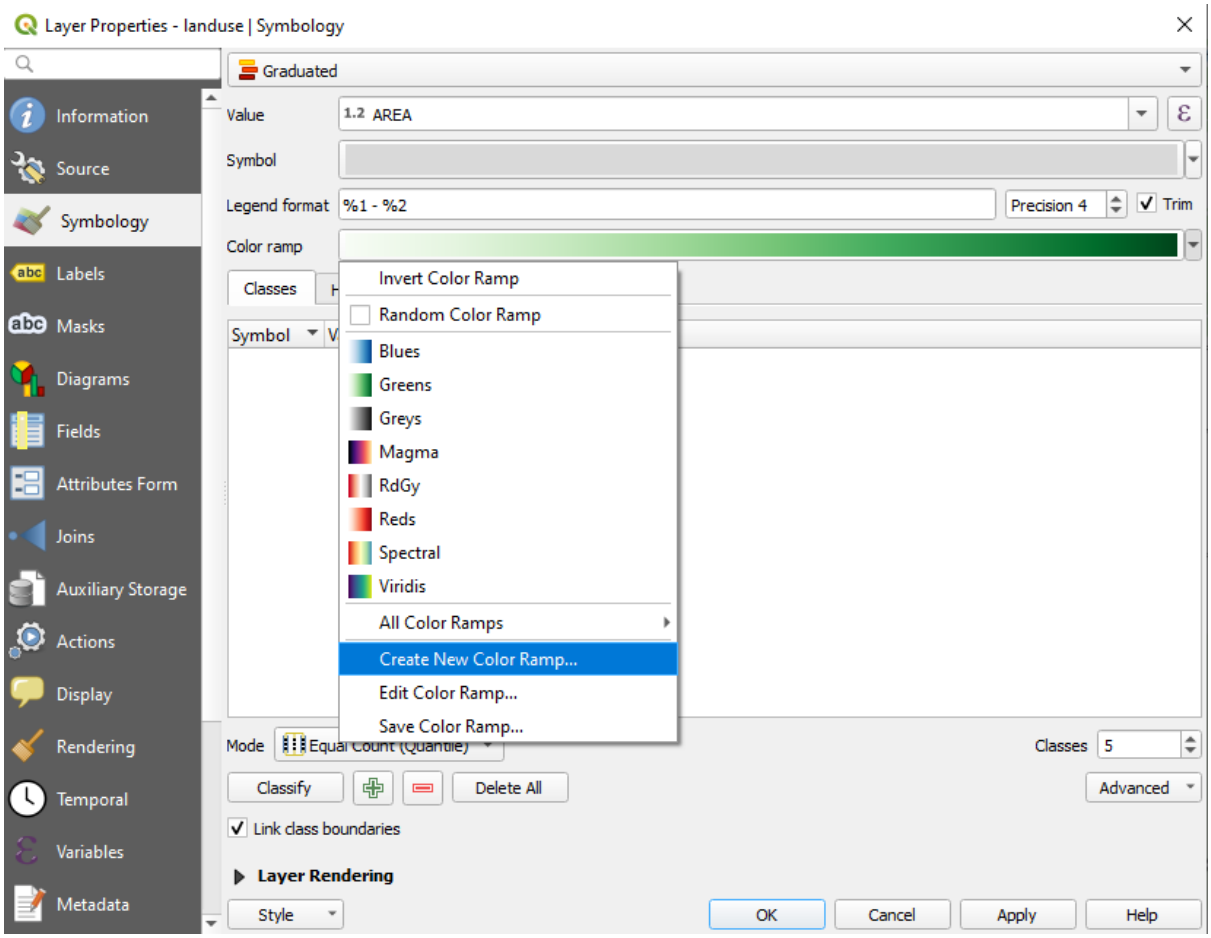
Nota: These areas respect the project's area unit settings, so they may be in square meters or square degrees.

---

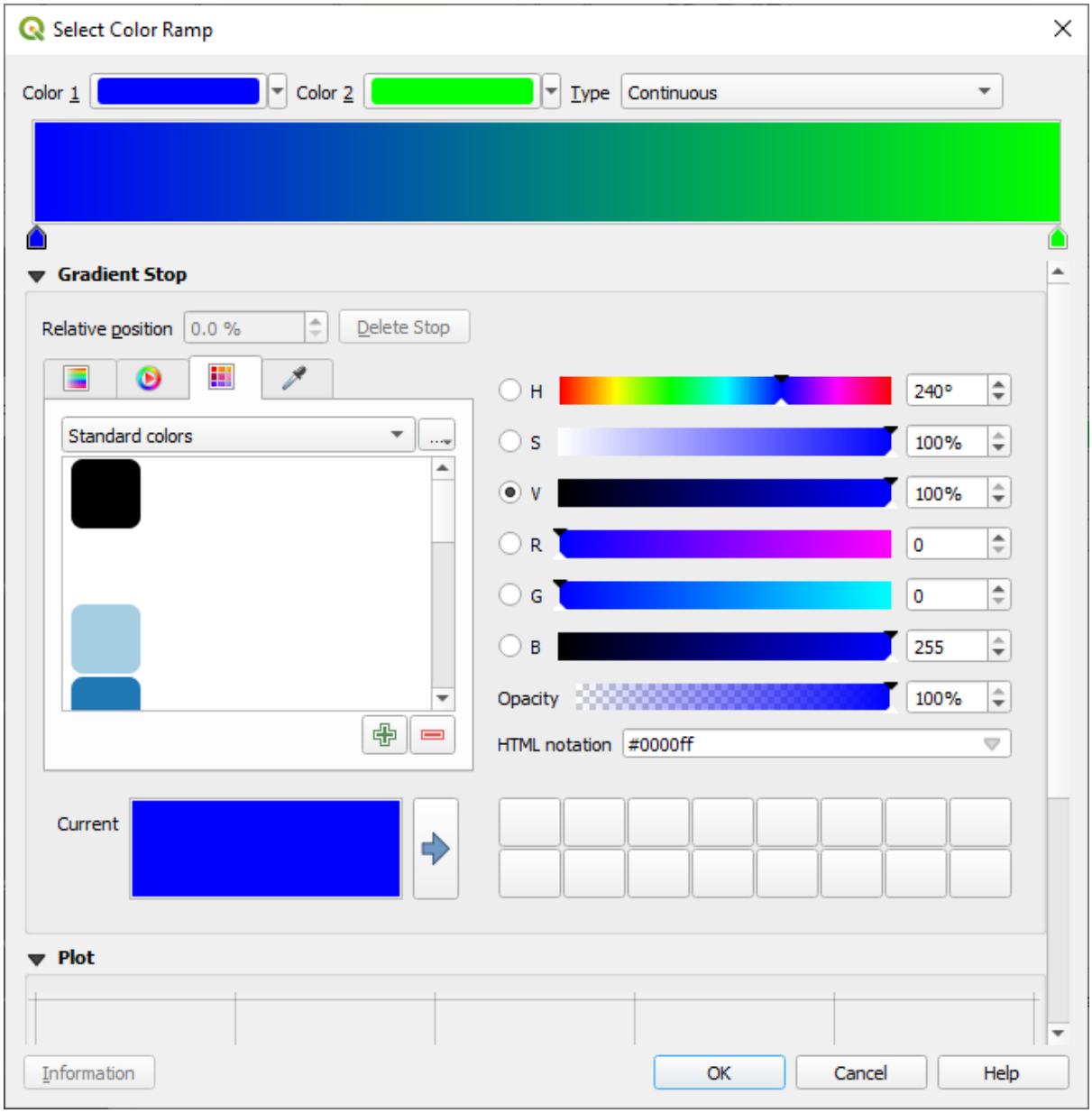
5. Press to save the edits and exit the edit mode with Toggle editing
6. Close the attribute table

Now that we have the data, let's use them to render the `landuse` layer.

1. Open the *Layer properties* dialog's *Symbology* tab for the `landuse` layer
2. Change the classification style from *Categorized* to *Graduated*
3. Change the *Value* to `AREA`
4. Under *Color ramp*, choose the option *Create New Color Ramp...*:



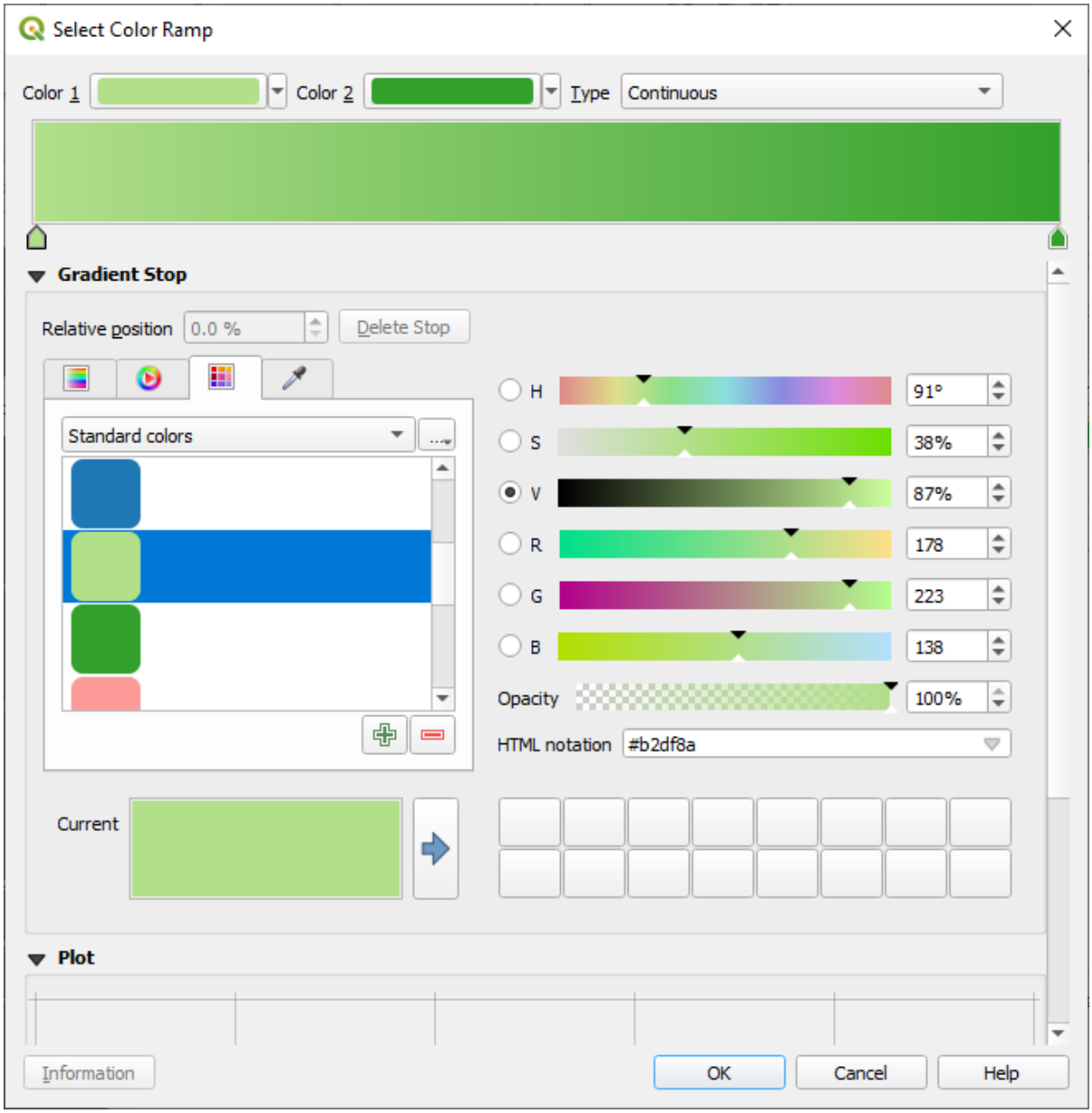
5. Choose *Gradient* (if it's not selected already) and click *OK*. You will see this:



You'll be using this to denote area, with small areas as *Color 1* and large areas as *Color 2*.

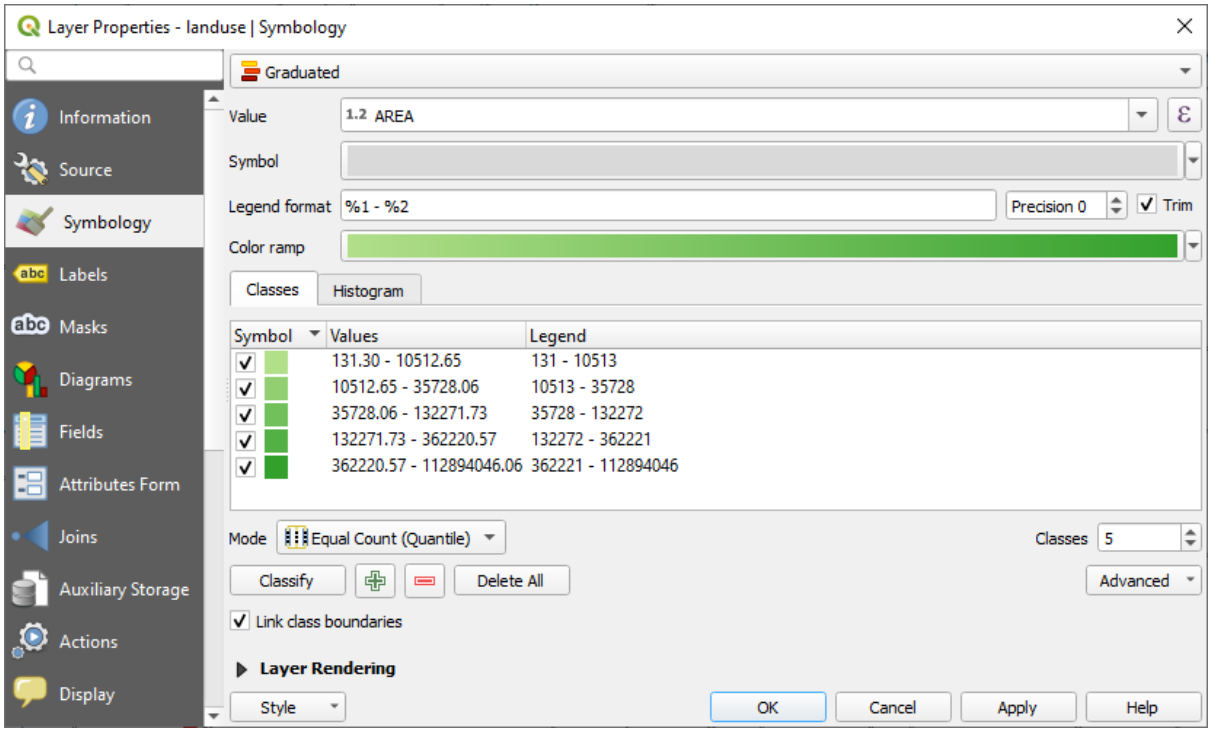
6. Choose appropriate colors

   In the example, the result looks like this:



7. Click *OK*
8. You can save the colour ramp by selecting *Save Color Ramp...* under the *Color ramp* tab. Choose an appropriate name for the colour ramp and click *Save*. You will now be able to select the same colour ramp easily under *All Color Ramps*.
9. Under *Mode* choose Equal Count (Quantile) .
10. Click *Classify*

    Now you will have something like this:



Leave everything else as-is.

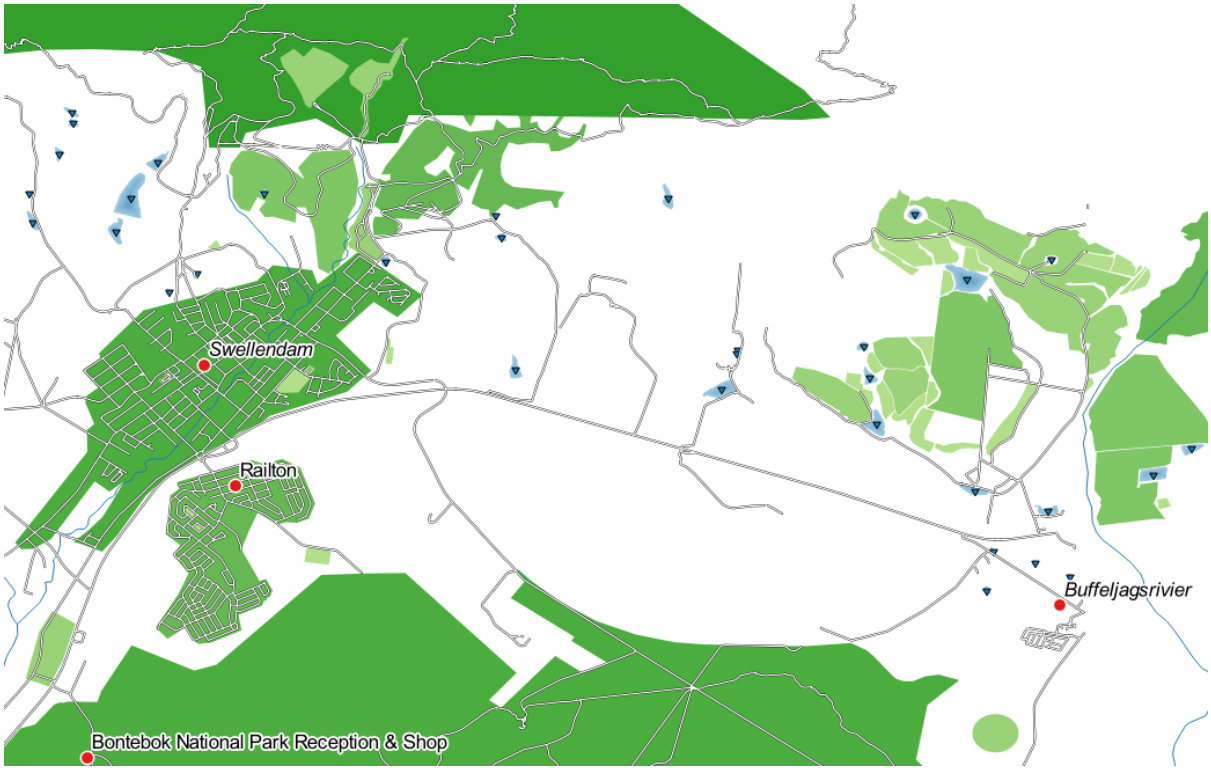
11. Click *OK*:

3.3.4 ??? Try Yourself: Refine the Classification

- Change the values of *Mode* and *Classes* until you get a classification that makes sense.

Answer

The settings you used might not be the same, but with the values *Classes* = `6` and *Mode* = *Natural Breaks (Jenks)* (and using the same colors, of course), the map will look like this:



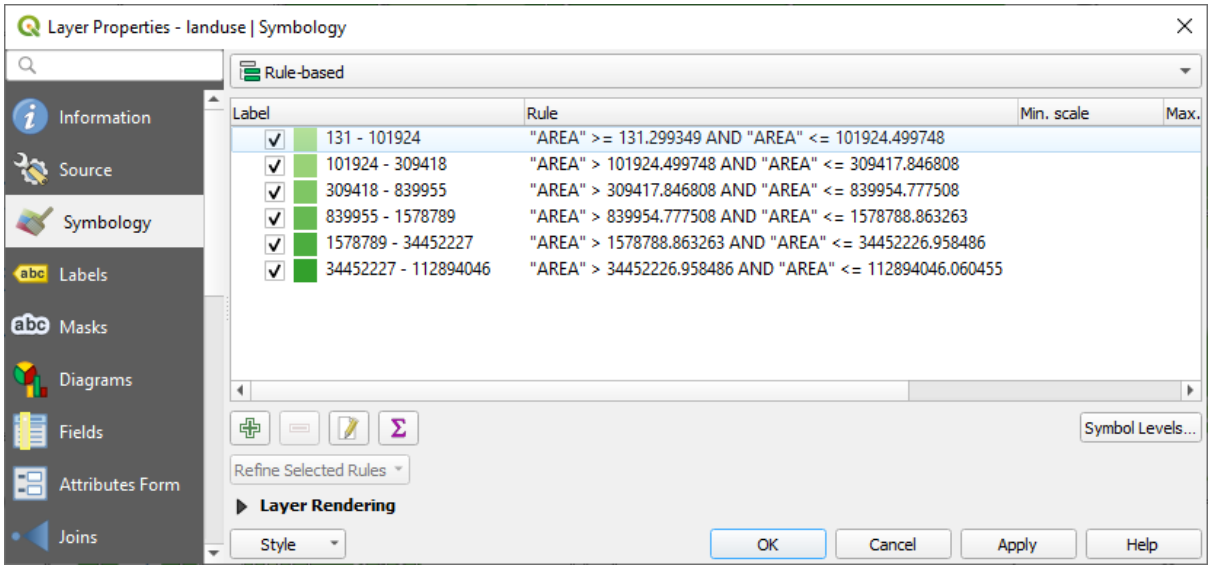
3.3.5 ??? Follow Along: Rule-based Classification

It's often useful to combine multiple criteria for a classification, but unfortunately normal classification only takes one attribute into account. That's where rule-based classification comes in handy.

In this lesson, we will represent the `landuse` layer in a way to easily identify Swellendam city from the other residential area, and from the other types of landuse (based on their area).

1. Open the *Layer Properties* dialog for the `landuse` layer
2. Switch to the *Symbology* tab
3. Switch the classification style to *Rule-based*

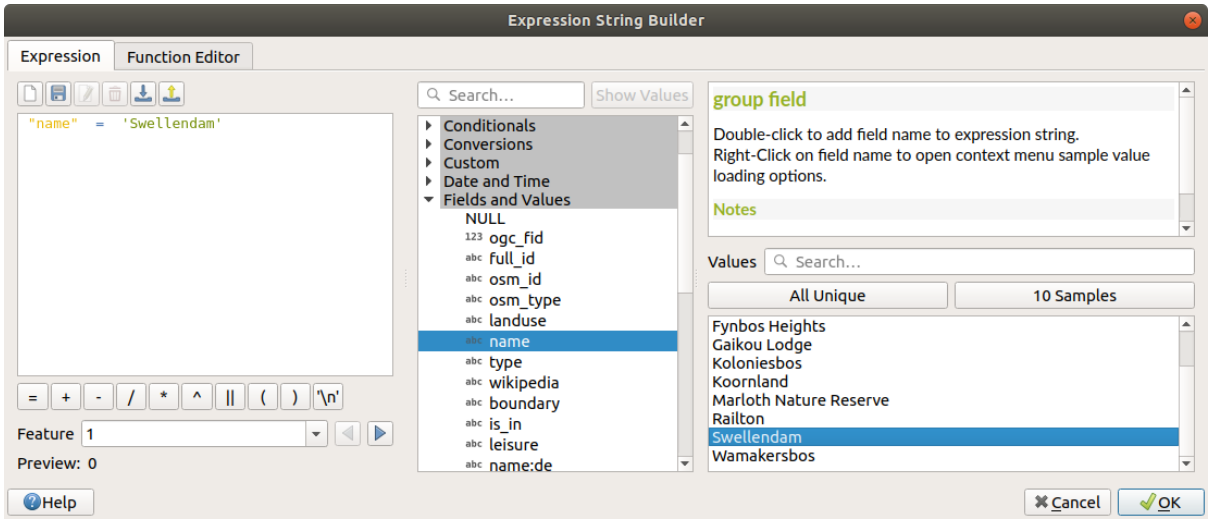
   QGIS will automatically show the rules that represent the current classification implemented for this layer. For example, after completing the exercise above, you may see something like this:



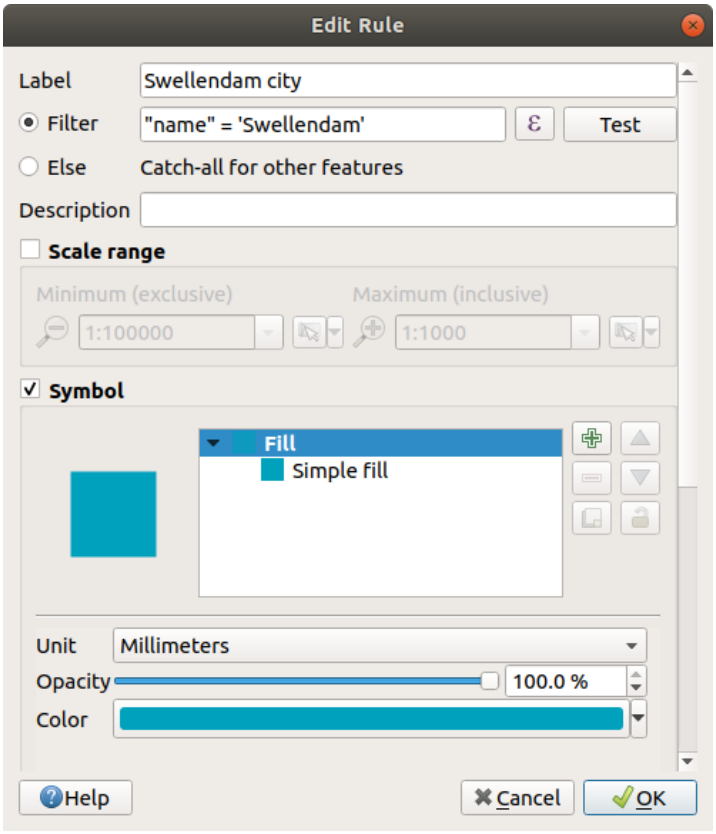
4. Click and drag to select all the rules
5. Use the Remove selected rules button to remove all of the existing rules

Let's now add our custom rules.

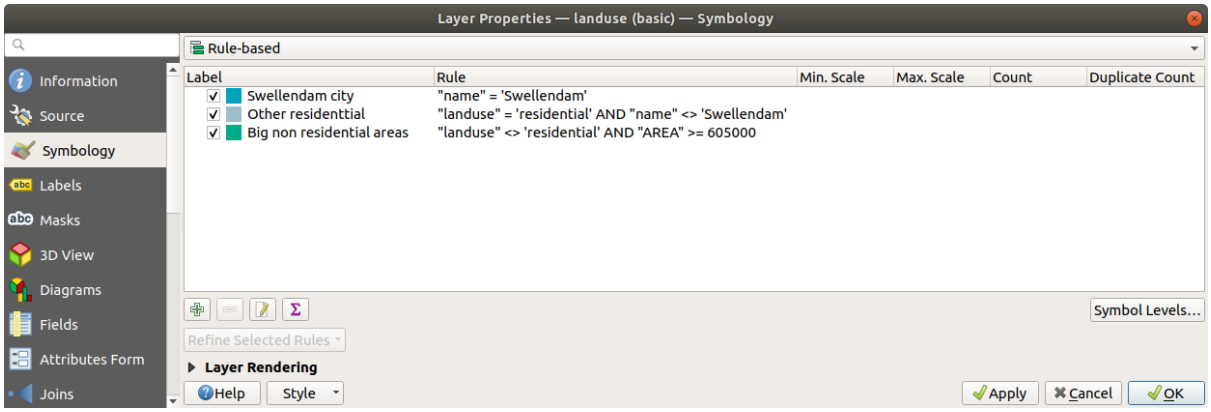
1. Click the Add rule button
2. The *Edit rule* dialog then appears
3. Enter `Swellendam city` as *Label*
4. Click the ε button next to the *Filter* text area to open the *Expression String Builder*
5. Enter the criterion `"name" = 'Swellendam'` and validate



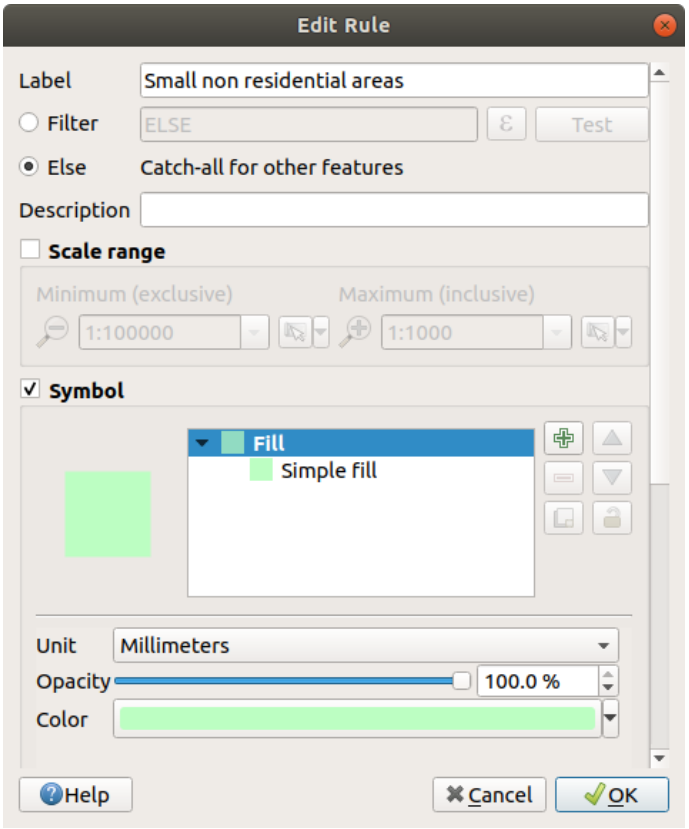
6. Back to the *Edit rule* dialog, assign it a darker grey-blue color in order to indicate the town's importance in the region and remove the border



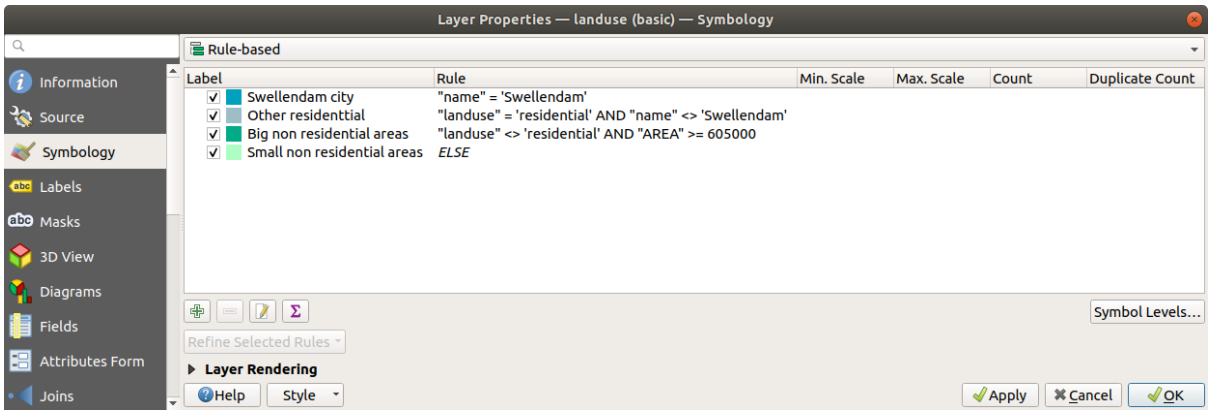
7. Prima *OK*
8. Repeat the steps above to add the following rules:
    1. **Other residential** label with the criterion `"landuse" = 'residential' AND "name" <> 'Swellendam'`. Choose a pale blue-grey *Fill color*
    2. **Big non residential areas** label with the criterion `"landuse" <> 'residential' AND "AREA" >= 605000`. Choose a mid-green color.



    These filters are exclusive, in that they exclude areas on the map (non-residential areas which are smaller than 605000 (square meters) are not included in any of the rules).
    3. We will catch the remaining features using a new rule labeled **Small non residential areas**. Instead of a filter expression, Check the ◉ *Else*. Give this category a suitable pale green color.



Your rules should now look like this:



9. Apply this symbology

Your map will look something like this:

Now you have a map with Swellendam the most prominent residential area and other non-residential areas colored according to their size.

3.3.6 In Conclusion

Symbology allows us to represent the attributes of a layer in an easy-to-read way. It allows us as well as the map reader to understand the significance of features, using any relevant attributes that we choose. Depending on the problems you face, you'll apply different classification techniques to solve them.

3.3.7 What's Next?

Now we have a nice-looking map, but how are we going to get it out of QGIS and into a format we can print out, or make into an image or PDF? That's the topic of the next lesson!

CAPÍTULO 4

Module: Laying out the Maps

In this module, you'll learn how to use the QGIS print layout to produce quality maps with all the requisite map components.

4.1 Lesson: Using Print Layout

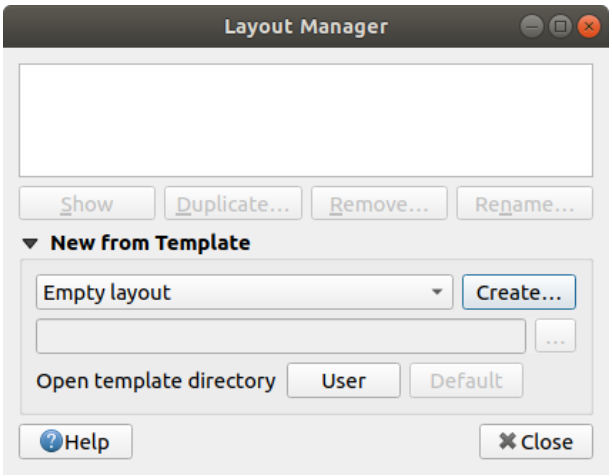
Now that you've got a map, you need to be able to print it or to export it to a document. The reason is, a GIS map file is not an image. Rather, it saves the state of the GIS program, with references to all the layers, their labels,colors, etc. So for someone who doesn't have the data or the same GIS program (such as QGIS), the map file will be useless. Luckily, QGIS can export its map file to a format that anyone's computer can read, as well as printing out the map if you have a printer connected. Both exporting and printing is handled via the *Print Layout*.

The goal for this lesson: To use the QGIS *Print Layout* to create a basic map with all the required settings.

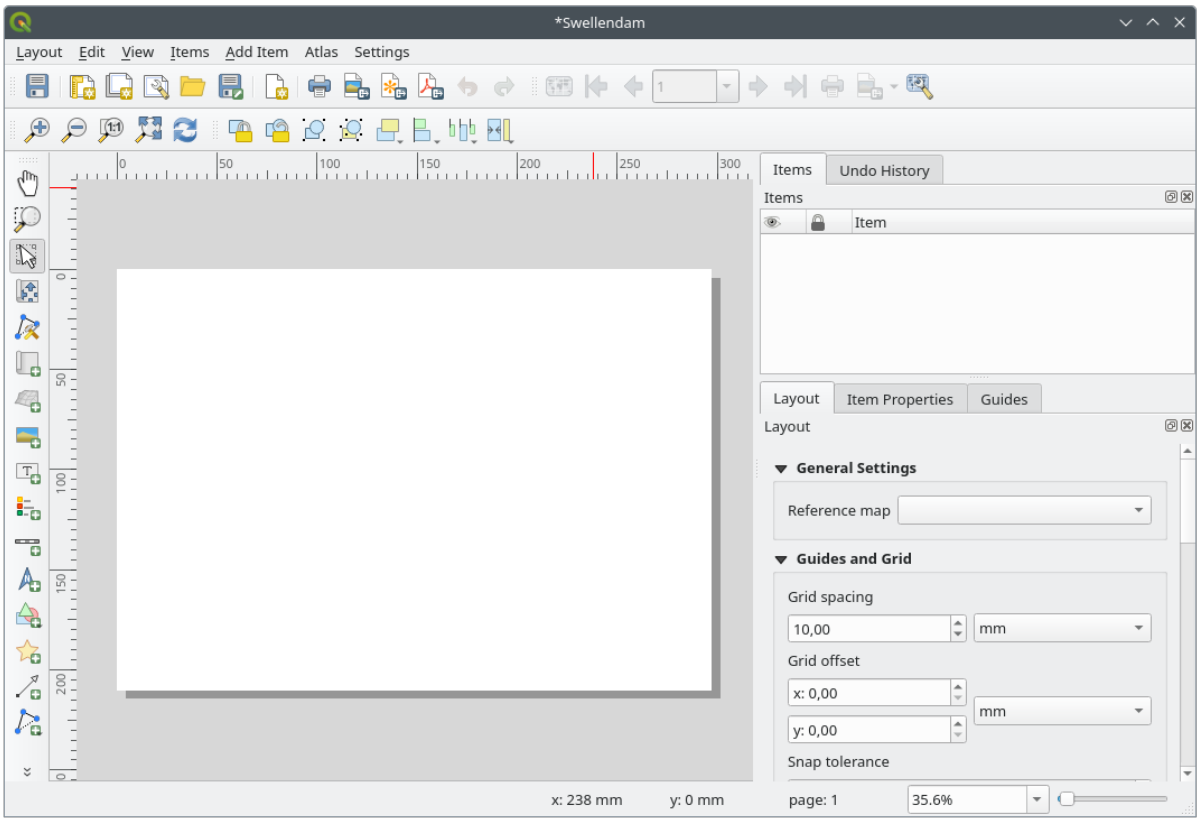
4.1.1 ??? Follow Along: The Layout Manager

QGIS allows you to create multiple maps using the same map file. For this reason, it has a tool called the *Layout Manager*.

1. Click on the *Project* ► *Layout Manager…* menu entry to open this tool. You'll see a blank *Layout manager* dialog appear.

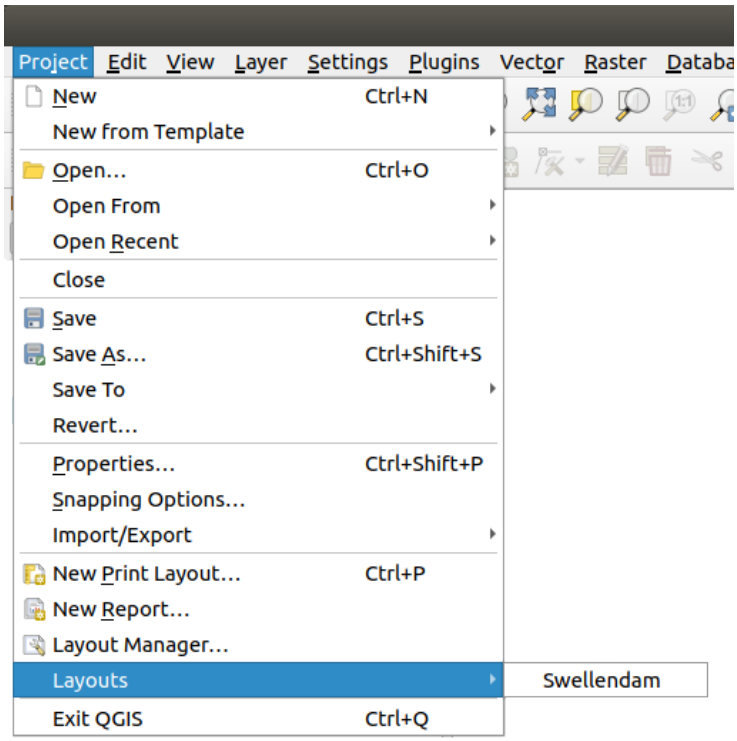


2. Under *New from Template*, select *Empty layout* and press the *Create…* button.
3. Give the new layout the name of Swellendam and click *OK*.
4. You will now see the *Print Layout* window:



You could also create this new layout via the *Project* ► *New Print Layout…* menu.

Whichever route you take, the new print layout is now accessible from the *Project* ► *Layouts* ► menu, as in the image below.



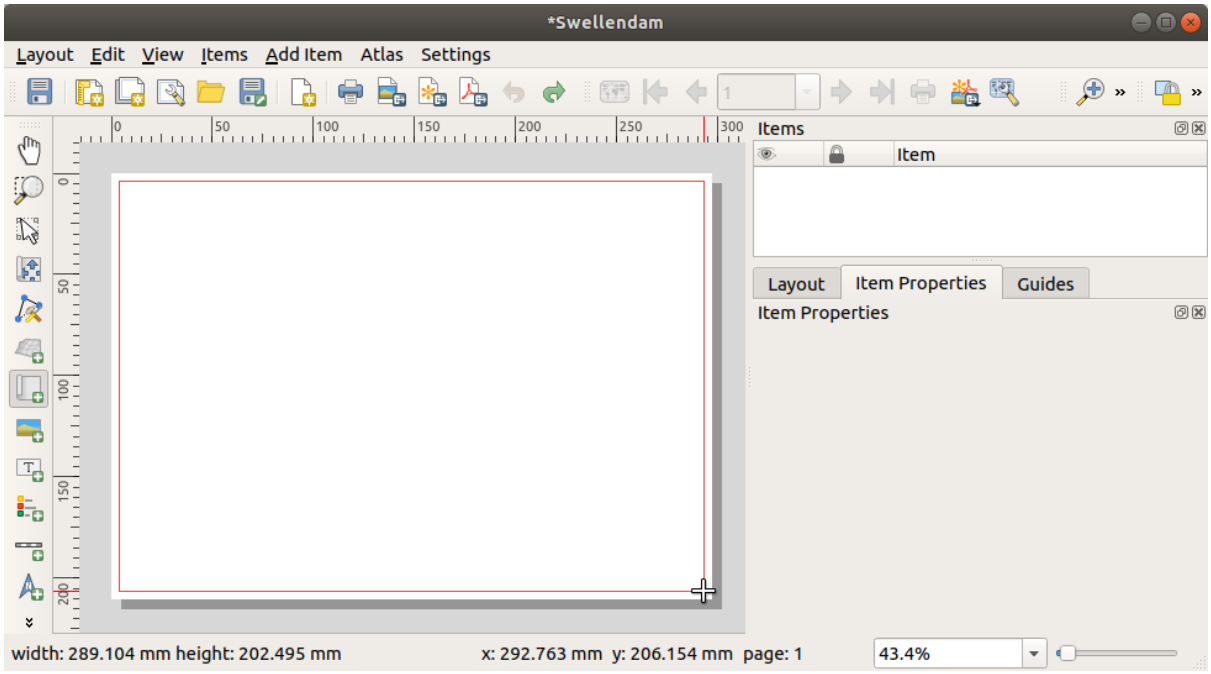
4.1.2 ??? Follow Along: Basic Map Composition

In this example, the composition was already the way we wanted it. Ensure that yours is as well.

1. Right-click on the sheet in the central part of the layout window and choose *Page properties...* in the context menu.
2. Check that the values in the *Item Properties* tab are set to the following:
   - *Size*: `A4`
   - *Orientation*: `Landscape`

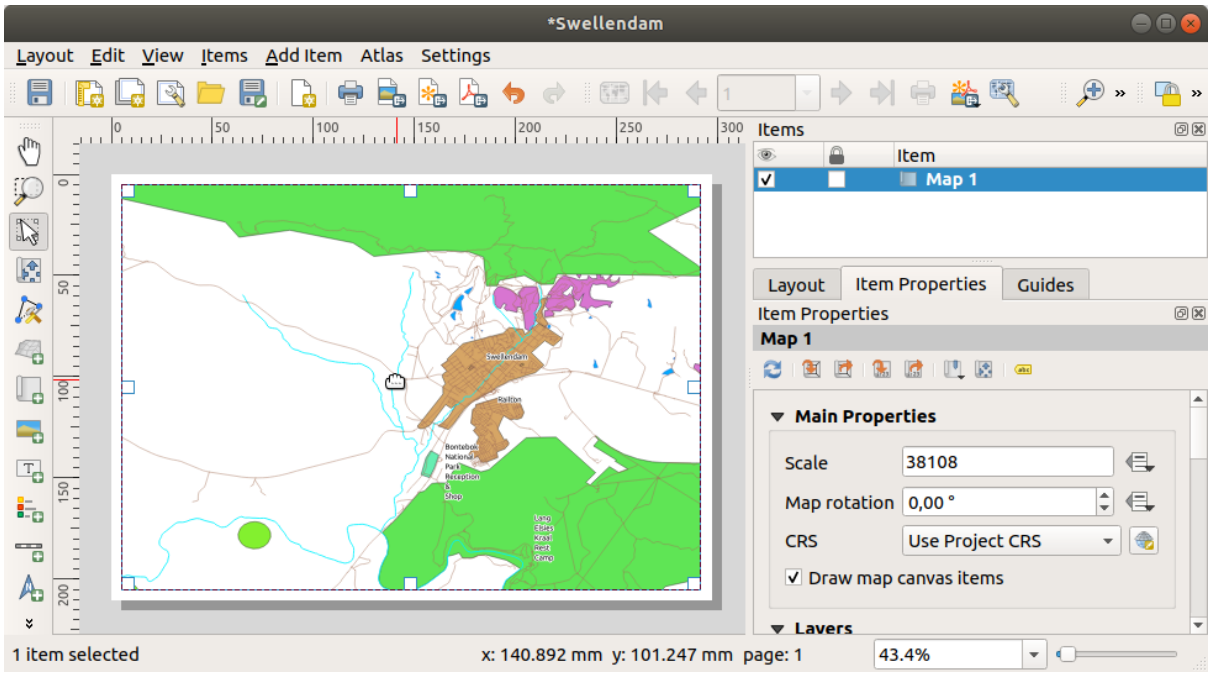
   Now you've got the page layout the way you wanted it, but this page is still blank. It clearly lacks a map. Let's fix that!
3. Click on the Add Map button.

   With this tool activated, you will be able to place a map on the page.
4. Click and drag a box on the blank page:

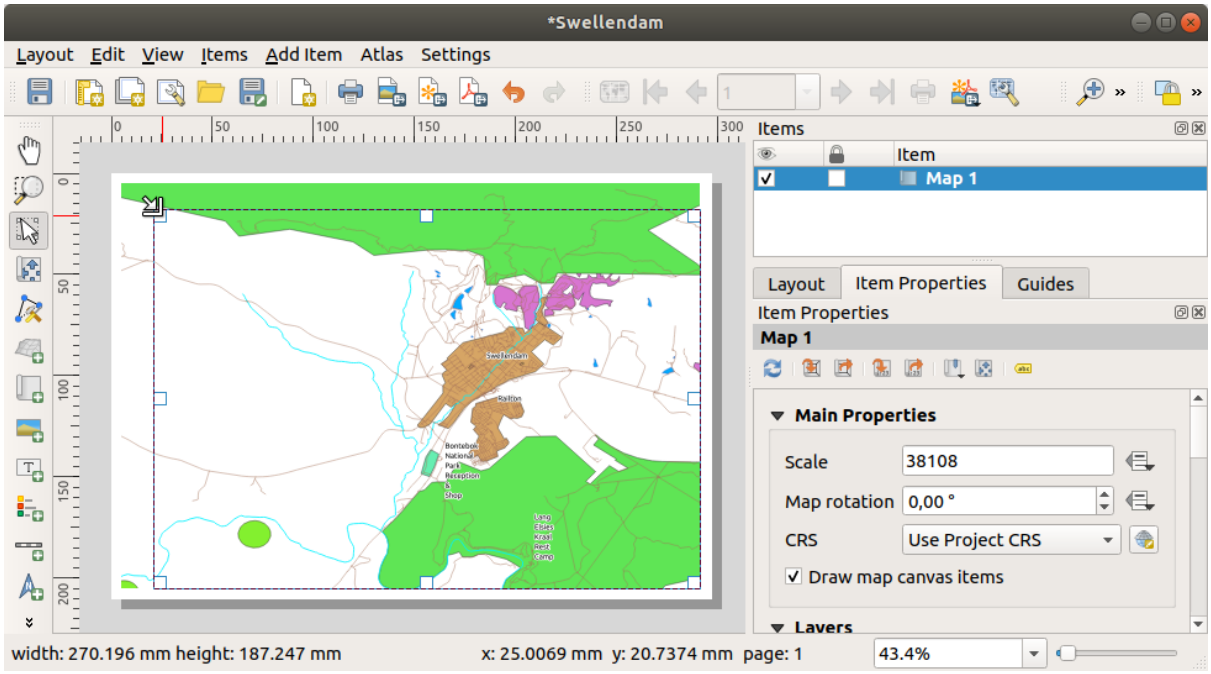


The map will appear on the page.

5. Move the map by clicking and dragging it around:



6. Resize it by clicking and dragging the boxes on the edges:



Nota: Your map may look a lot different, of course! This depends on how your own project is set up. But not to worry! These instructions are general, so they will work the same regardless of what the map itself looks like.

7. Be sure to leave margins along the edges, and a space along the top for the title.
8. Zoom in and out on the page (but not the map!) by using these buttons:

9. Zoom and pan the map in the main QGIS window. You can also pan the map using the Move item content tool.

   The map view updates as you zoom in or zoom out.

10. If, for any reason, the map view does not refresh correctly, you can force the map to refresh by clicking the Refresh view button.

    Remember that the size and position you've given the map doesn't need to be final. You can always come back and change it later if you're not satisfied. For now, you need to ensure that you've saved your work on this map. Because a *Print Layout* in QGIS is part of the main map file, you must save your project.

11. Go to the *Layout* ► *Save Project*. This is a convenient shortcut to the one in the main dialog.

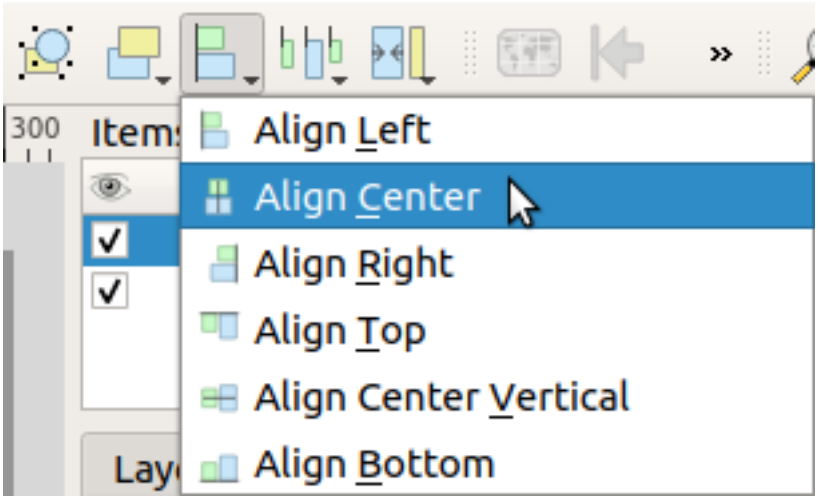
4.1.3 ??? Follow Along: Adding a Title

Now your map is looking good on the page, but your readers/users are not being told what's going on yet. They need some context, which is what you'll provide for them by adding map elements. First, let us add a title.

1. Click on the Add Label button
2. Click on the page, above the map, accept the suggested values in the *New Item Properties* dialog, and a label will appear at the top of the map.
3. Resize it and place it in the top center of the page. It can be resized and moved in the same way that you resized and moved the map.

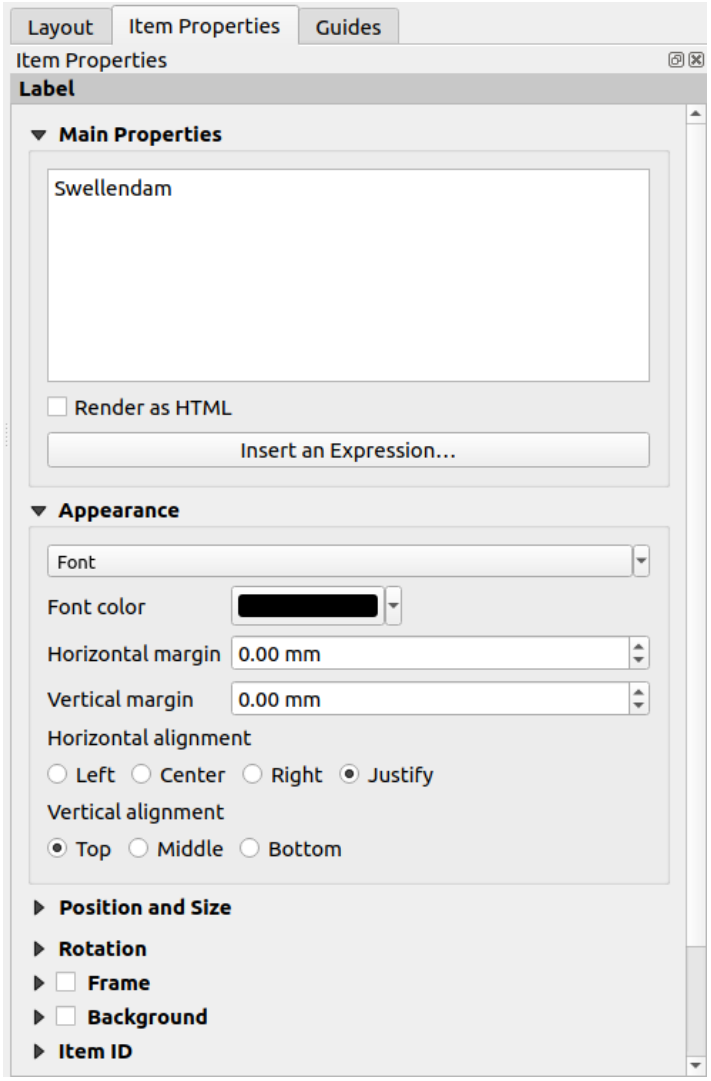
As you move the title, you'll notice that guidelines appear to help you position the title in the center of the page. However, there is also a tool in the Actions Toolbar to help position the title relative to the map (not the page):

4. Click the map to select it
5. Hold in `Shift` on your keyboard and click on the label so that both the map and the label are selected.
6. Look for the Align selected items left button and click on the dropdown arrow next to it to reveal the positioning options and click *Align center*:



Now the label frame is centered on the map, but not the contents. To center the contents of the label:

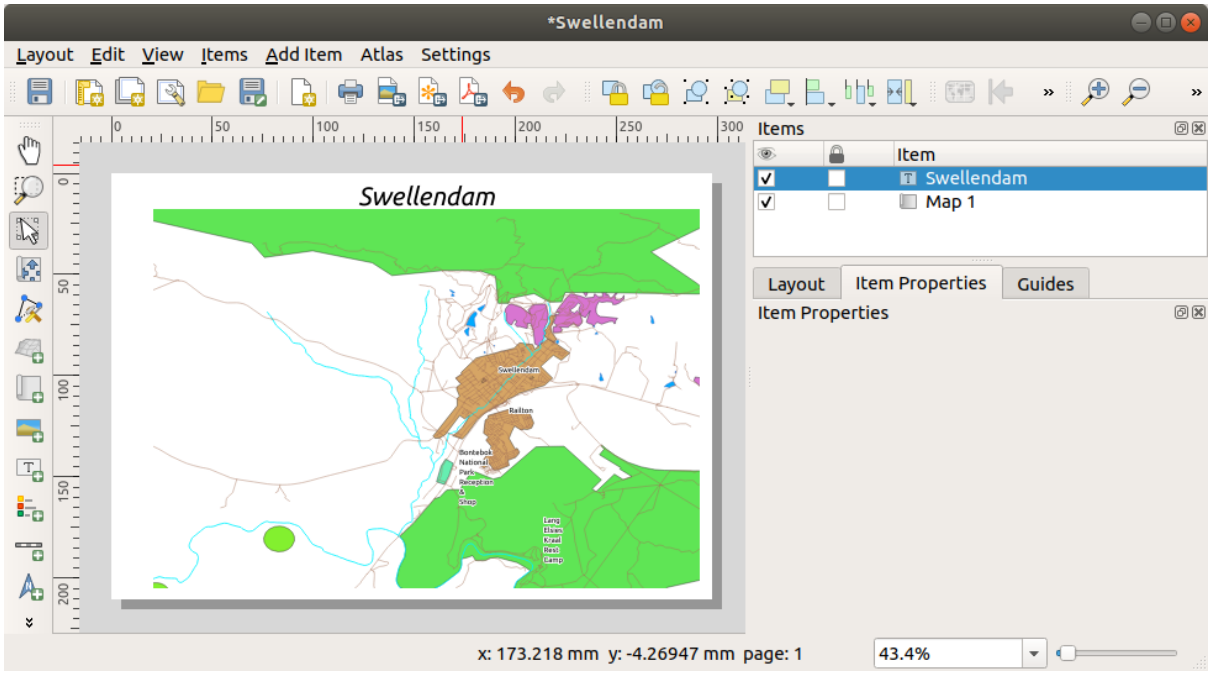
1. Select the label by clicking on it.
2. Click on the *Item Properties* tab in the side panel of the layout window.
3. Change the text of the label to «Swellendam»:



4. Use this interface to set the font and alignment options under the *Appearance* section:
    1. Choose a large but sensible font (the example will use the default font with a size of `36`)
    2. Set the *Horizontal Alignment* to *Center*.

    You can also change the font color, but it's probably best to keep it black as per the default.
5. The default setting is not to add a frame to the title's text box. However, if you wish to add a frame, you can do so:
    1. In the *Item Properties* tab, scroll down until you see the *Frame* option.
    2. Click the *Frame* checkbox to enable the frame. You can also change the frame's color and width.

In this example, we won't enable the frame, so here is our page so far:



To make sure that you don't accidentally move these elements around now that you've aligned them, you can lock items into place:

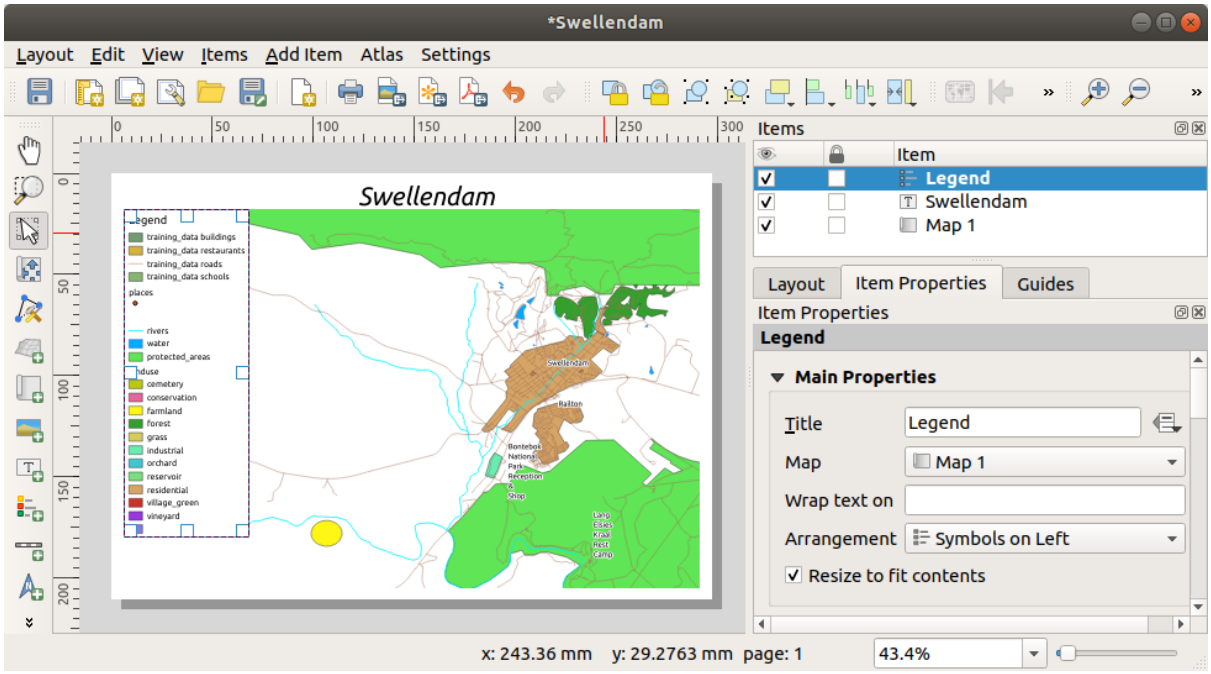
1. Select both the label and the map items
2. Click the Lock Selected Items button in the *Actions* Toolbar.

Nota: Click the Unlock All Items button in the *Actions* Toolbar to be able to edit the items again.

4.1.4 ??? Follow Along: Adding a Legend

The map reader also needs to be able to see what various things on the map actually mean. In some cases, like the place names, this is quite obvious. In other cases, it's more difficult to guess, like the colors of the forests. Let's add a new legend.

1. Click on the Add Legend button
2. Click on the page to place the legend, accept the suggested values in the *New Item Properties* dialog,
3. A legend is added to the layout page, showing layers symbology as set in the main dialog.
4. As usual, you can click and move the item to where you want it:



4.1.5 ??? Follow Along: Customizing Legend Items

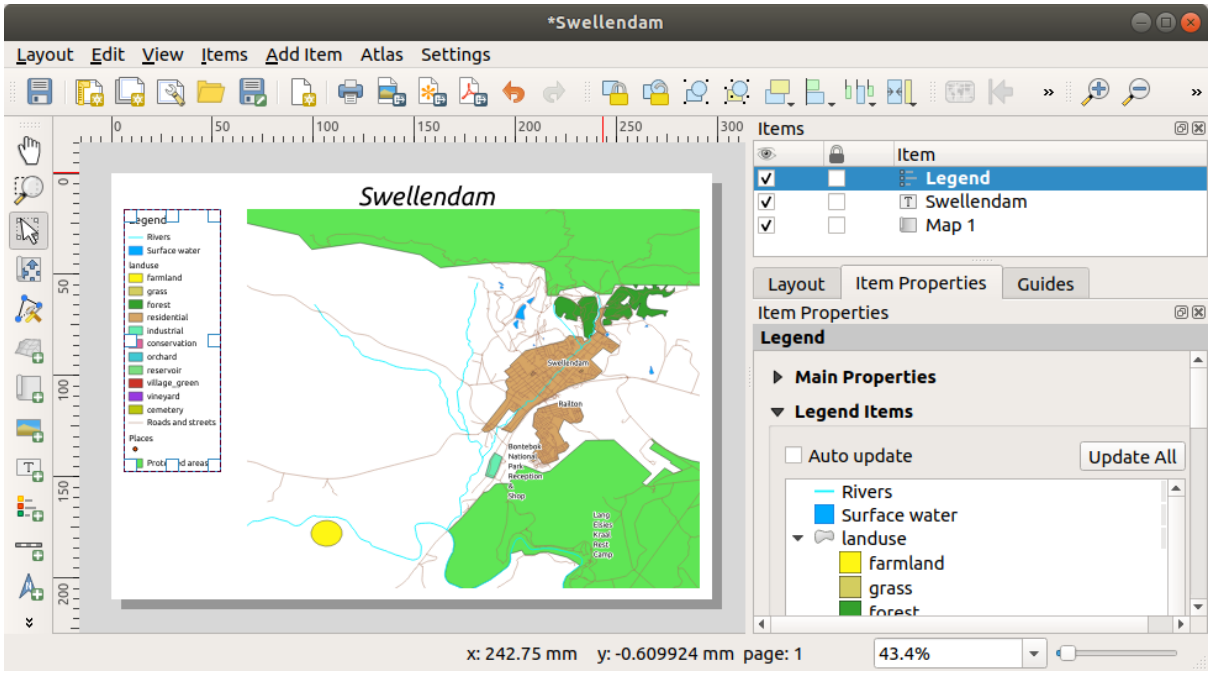
Not everything on the legend is necessary, so let's remove some unwanted items.

1. In the *Item Properties* tab, you'll find the *Legend items* group.
2. Uncheck the *Auto update* box, allowing you to directly modify the legend items
3. Select the entry with *buildings*
4. Delete it from the legend by clicking the button

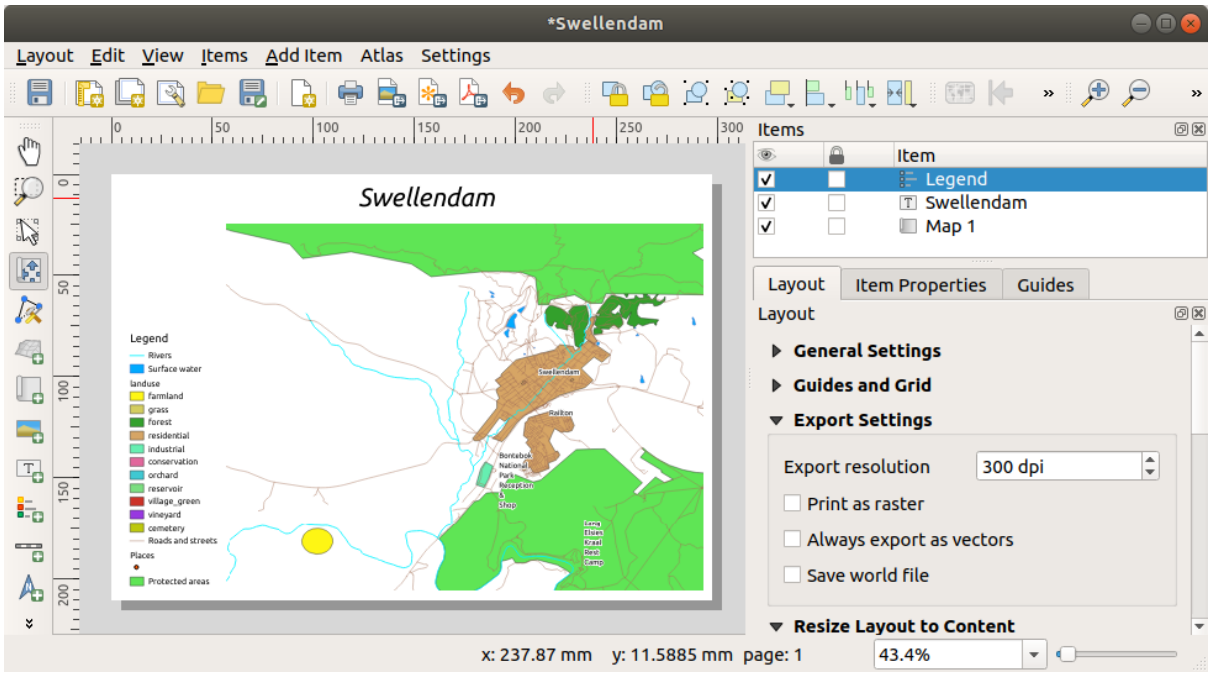
You can also rename items.

1. Select a layer from the same list.
2. Click the Edit selected item properties button.
3. Rename the layers to `Places`, `Roads and Streets`, `Surface Water`, and `Rivers`.

You can also reorder the items:



As the legend will likely be widened by the new layer names, you may wish to move and resize the legend and or map. This is the result:



4.1.6 ??? Follow Along: Exporting Your Map

Nota: Did you remember to save your work often?

Finally the map is ready for export! You'll see the export buttons near the top left corner of the layout window:

- Print Layout: interfaces with a printer. Since the printer options will differ depending on the model of printer that you're working with, it's probably better to consult the printer manual or a general guide to printing for more information on this topic.

  The other buttons allow you to export the map page to a file.

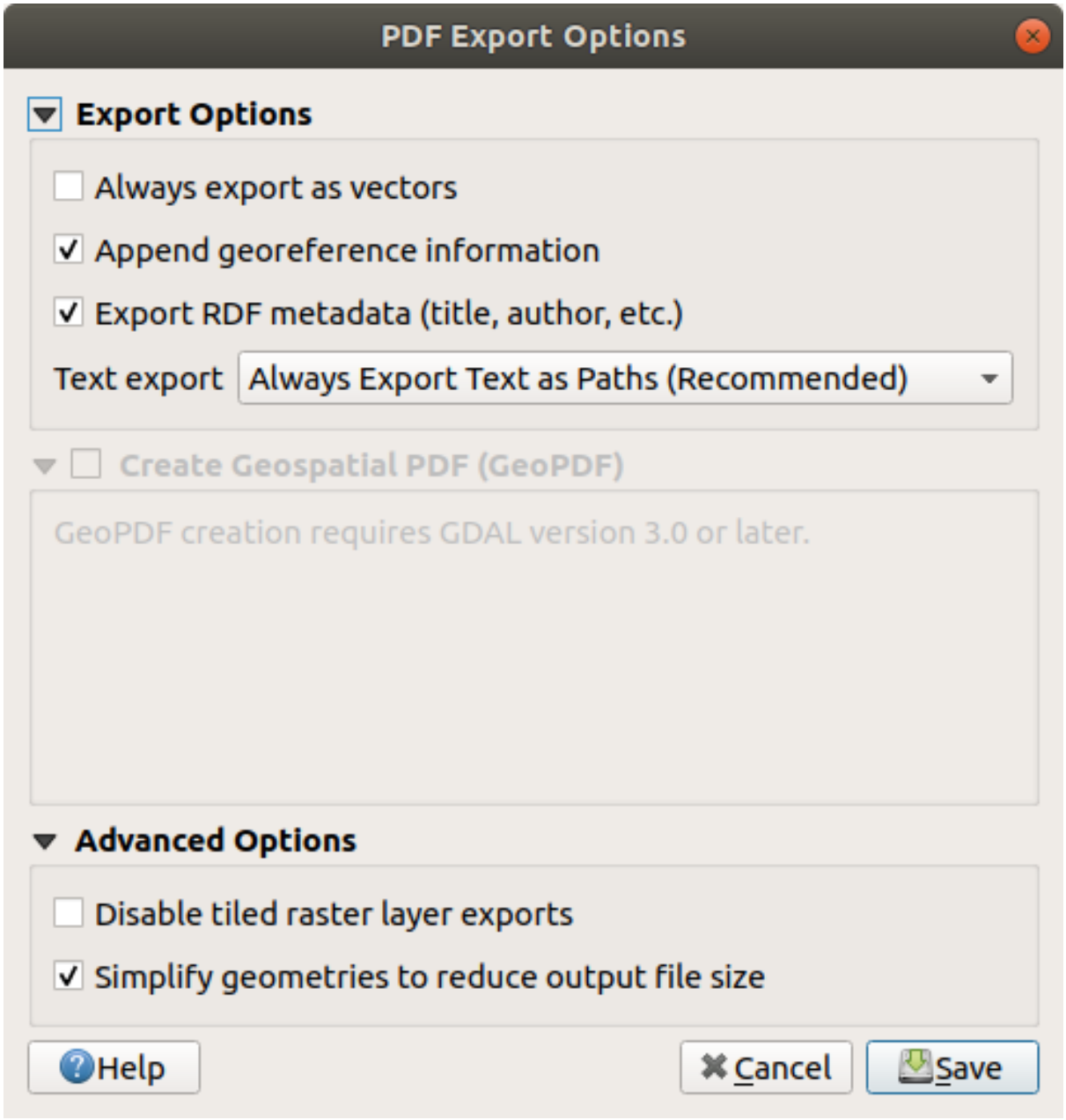
- Export as Image: gives you a selection of various common image formats to choose from. This is probably the simplest option, but the image it creates is «dead» and difficult to edit.

- Export as SVG: If you're sending the map to a cartographer (who may want to edit the map for publication), it's best to export as an SVG. SVG stands for «Scalable Vector Graphic», and can be imported to programs like Inkscape or other vector image editing software.

- Export as PDF: If you need to send the map to a client, it's most common to use a PDF, because it's easier to set up printing options for a PDF. Some cartographers may prefer PDF as well, if they have a program that allows them to import and edit this format.

For our purposes, we're going to use PDF.

1. Click the Export as PDF button
2. Choose a save location and a file name as usual. The following dialog will show up.



3. You can safely use the default values now and click *Save*.

   QGIS will proceed to the map export and push a message on top of the print layout dialog as soon as it finishes.

4. Click the hyperlink in the message to open the folder in which the PDF has been saved in your system's file manager

5. Open it and see how your layout looks.

   Everything is OK? Congratulations on your first completed QGIS map project!

6. Anything unsatisfying? Go back to the QGIS window, do the appropriate modifications and export again.

7. Remember to save your project file.

4.1.7 In Conclusion

Now you know how to create a basic static map layout. We can go a step further and create a map layout that adapts dynamically, with more layout items.

4.2 Lesson: Creating a Dynamic Print Layout

Now that you have learned to create a basic map layout we go a step further and create a map layout that adapts dynamically to our map extent and to the page properties, e.g. when you change the size of the page. Also, the date of creation will adapt dynamically.

4.2.1 ??? Follow Along: Creating the dynamic map canvas

1. Load the ESRI Shapefile format datasets `protected_areas.shp`, `places.shp`, `rivers.shp` and `water.shp` into the map canvas and adapt its properties to suit your own convenience.
2. After everything is rendered and symbolized to your liking, click the New Print Layout icon in the *Project* toolbar or choose *Project* ► *New Print Layout*. You will be prompted to choose a title for the new print layout.
3. We want to create a map layout consisting of a header and a map of the region near Swellendam, South Africa. The layout should have a margin of 7.5 mm and the header should be 36mm high.
4. Create a map item called `main map` on the canvas and go to the *Layout* panel. Scroll down to the *Variables* section and find the *Layout* part. Here we set some variables you can use all over the dynamic print layout. Go to the *Layout* panel and scroll down to the *Variables* section. The first variable will define the margin. Press the button and type in the name `sw_layout_margin`. Set the value to `7.5`. Press the button again and type in the name `sw_layout_height_header`. Set the value to `36`.
5. Now you are ready to create the position and the size of the map canvas automatically by means of the variables. Make sure that your map item is selected, go to the *Item Properties* panel, scroll down to and open the *Position and Size* section. Click the Data defined override for *X* and from the *Variables* entry, choose `@sw_layout_margin`.
6. Click the Data defined override for *Y*, choose *Edit…* and type in the formula:

```
to_real(@sw_layout_margin) + to_real(@sw_layout_height_header)
```

7. You can create the size of the map item by using the variables for *Width* and *Height*. Click the Data defined override for *Width* and choose *Edit …* again. Fill in the formula:

```
@layout_pagewidth - @sw_layout_margin * 2
```

Click the Data defined override for *Height* and choose *Edit ….* Here fill in the formula:

```
@layout_pageheight - @sw_layout_height_header - @sw_layout_margin * 2
```

8. We will also create a grid containing the coordinates of the main canvas map extent. Go to *Item Properties* again and choose the *Grids* section. Insert a grid by clicking the button. Click on *Modify grid …* and set the *Interval* for *X*, *Y* and *Offset* according to the map scale you chose in the QGIS main canvas. The *Grid type Cross* is very well suited for our purposes.

4.2.2 ??? Follow Along: Creating the dynamic header

1. Insert a rectangle which will contain the header with the Add Shape button. In the *Items* panel enter the name `header`.
2. Again, go to the *Item Properties* and open the *Position and Size* section. Using Data defined override, choose the `sw_layout_margin` variable for *X* as well as for *Y*. *Width* shall be defined by the expression:

```
@layout_pagewidth - @sw_layout_margin * 2
```

   and *Height* by the `sw_layout_height_header` variable.

3. We will insert a horizontal line and two vertical lines to divide the header into different sections using the Add Node Item. Create a horizontal line and two vertical lines and name them `Horizontal line`, `Vertical line 1` `Vertical line 2`.
   1. For the horizontal line:
      1. Set *X* to the variable `sw_layout_margin`
      2. Set the expression for *Y* to:

```
@sw_layout_margin + 8
```

      3. Set the expression for *Width* to:

```
@layout_pagewidth -  @sw_layout_margin * 3 - 53.5
```

   2. For the first vertical line:
      1. Set the expression for *X* to:

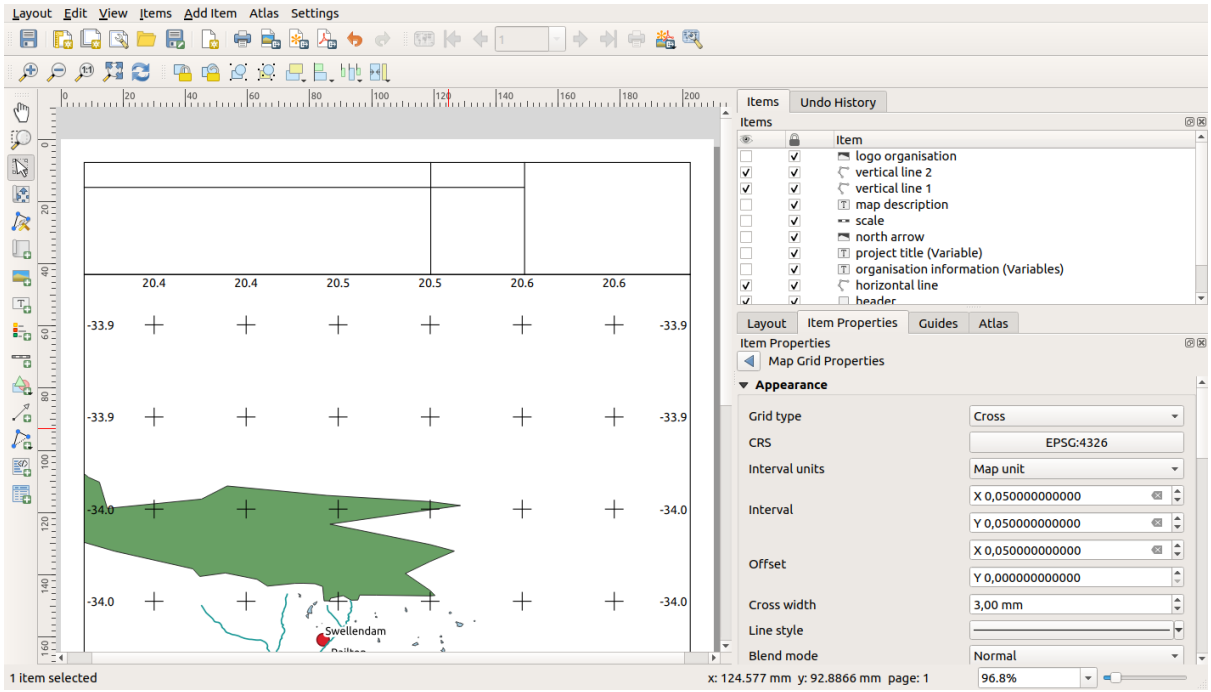
```
@layout_pagewidth - @sw_layout_margin * 2 - 53.5
```

      2. Set *Y* to the variable `sw_layout_margin`
      3. The height must be the same as the header we created, so set *Height* to the variable `sw_layout_height_header`.
   3. The second vertical line is placed to the left of the first one.
      1. Set the expression for *X* to:

```
@layout_pagewidth - @sw_layout_margin * 2 - 83.5
```

      2. Set *Y* to the variable `sw_layout_margin`
      3. The height shall be the same as the other vertical line, so set *Height* to the variable `sw_layout_height_header`.

   The figure below shows the structure of our dynamic layout. We will fill the areas created by the lines with some elements.



4.2.3 ??? Follow Along: Creating labels for the dynamic header

1. The title of your QGIS project can be included automatically. The title is set in the *Project Properties*. Insert a label with the Add Label button and enter the name `project title (variable)`. In the *Main Properties* of the *Items Properties* Panel enter the expression:

```
[%@project_title%]
```

   Set the position of the label.

   1. For *X*, use the expression:

```
@sw_layout_margin + 3
```

   2. For *Y*, use the expression:

```
@sw_layout_margin + 0.25
```

   3. For *Width*, use the expression:

```
@layout_pagewidth - @sw_layout_margin *2 - 90
```

   4. Enter `11.25` for *Height*

   Under *Appearance* set the Font size to 16 pt.

2. The second label will include a description of the map you created. Again, insert a label and name it `map description`. In the *Main Properties* enter the text `map description`. In the *Main Properties* we will also include:

```
printed on: [%format_date(now(),'dd.MM.yyyy')%]
```

   Here we used two `Date and Time` functions (`now` and `format_date`).

   Set the position of the label.

   1. For *X*, use the expression:

```
@sw_layout_margin + 3
```

2. For *Y*, use the expression:

```
@sw_layout_margin + 11.5
```

3. The third label will include information about your organisation. First we will create some variables in the *Variables* menu of the *Item Properties*. Go to the *Layout* menu, click the button each time and enter the names `o_department`, `o_name` , `o_adress` and `o_postcode`. In the second row enter the information about your organisation. We will use these variables in the *Main Properties* section.

   In *Main Properties* enter:

```
[% @o_name %]
[% @o_department %]
[% @o_adress %]
[% @o_postcode %]
```

   Set the position of the label.

   1. For *X*, use the expression:

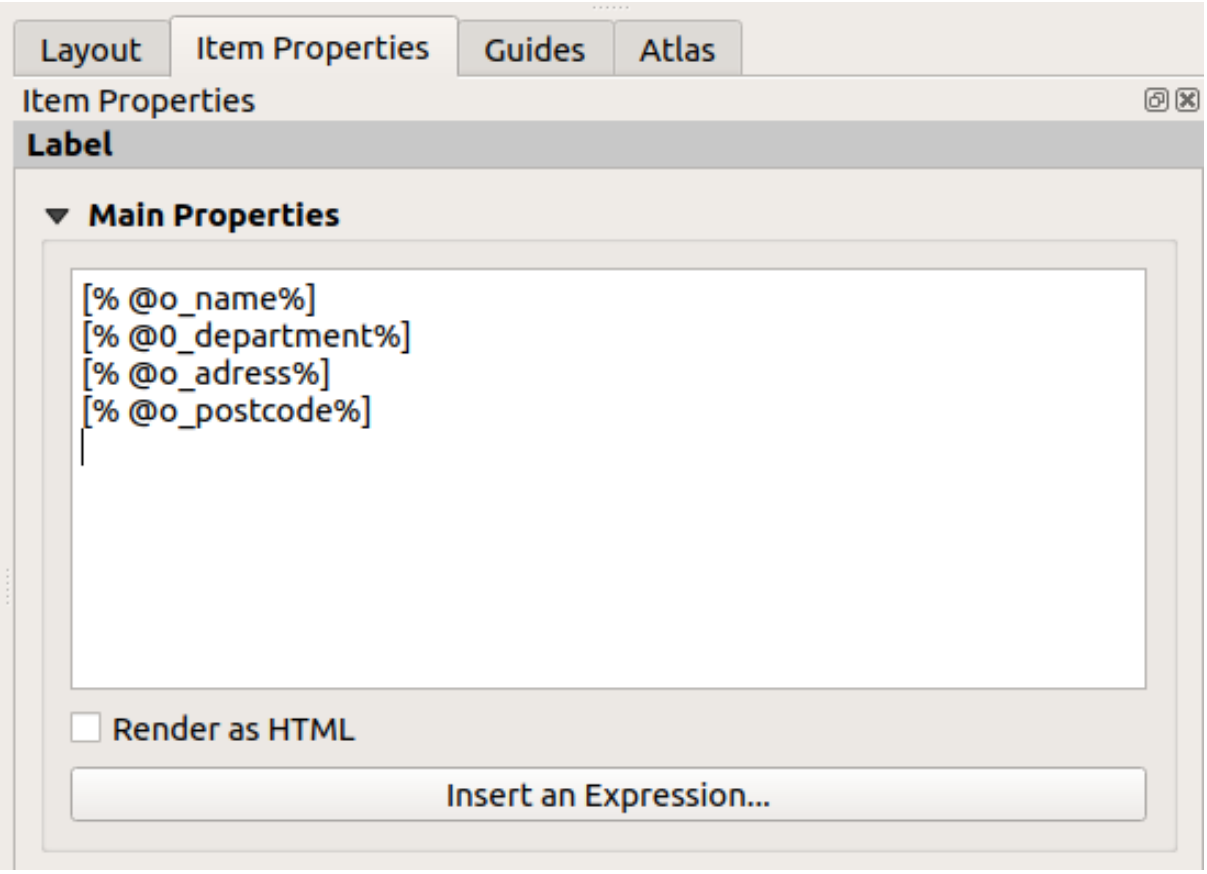
```
@layout_pagewidth - @sw_layout_margin - 49.5
```

   2. For *Y*, use the expression:

```
@sw_layout_margin + 15.5
```

   3. For *Width*, use `49.00`
   4. For *Height*, use the expression:

```
@sw_layout_height_header - 15.5
```



4.2.4 ??? Follow Along: Adding pictures to the dynamic header

1. Use the Add Picture button to place a picture above your label `organisation information`. After entering the name `organisation logo` define the position and size of the logo:
   1. For *X*, use the expression:

      ```
      @layout_pagewidth - @sw_layout_margin - 49.5
      ```

   2. For *Y*, use the expression:

      ```
      @sw_layout_margin + 3.5
      ```

   3. For *Width*, use `39.292`
   4. For *Height*, use `9.583`

   To include a logo of your organisation you have to save your logo under your home directory and enter the path under *Main Properties* ► *Image Source*.

2. Our layout still needs a north arrow. This will also be inserted by using Add North Arrow. We will use the default north arrow. Define the position:
   1. For *X*, use the expression:

      ```
      @layout_pagewidth - @sw_layout_margin * 2 - 78
      ```

   2. For *Y*, use the expression:

      ```
      @sw_layout_margin + 9
      ```

3. For *Width*, use `21.027`
4. For *Height*, use `21.157`

4.2.5 ??? Follow Along: Creating the scalebar of the dynamic header

1. To insert a scalebar in the header click on Add Scale Bar and place it in the rectangle above the north arrow. In *Map* under the *Main Properties* choose your `main map(Map 1)`. This means that the scale changes automatically according to the extent you choose in the QGIS main canvas. Choose the *Style Numeric*. This means that we insert a simple scale without a scalebar. The scale still needs a position and size.
   1. For *X*, use the expression:

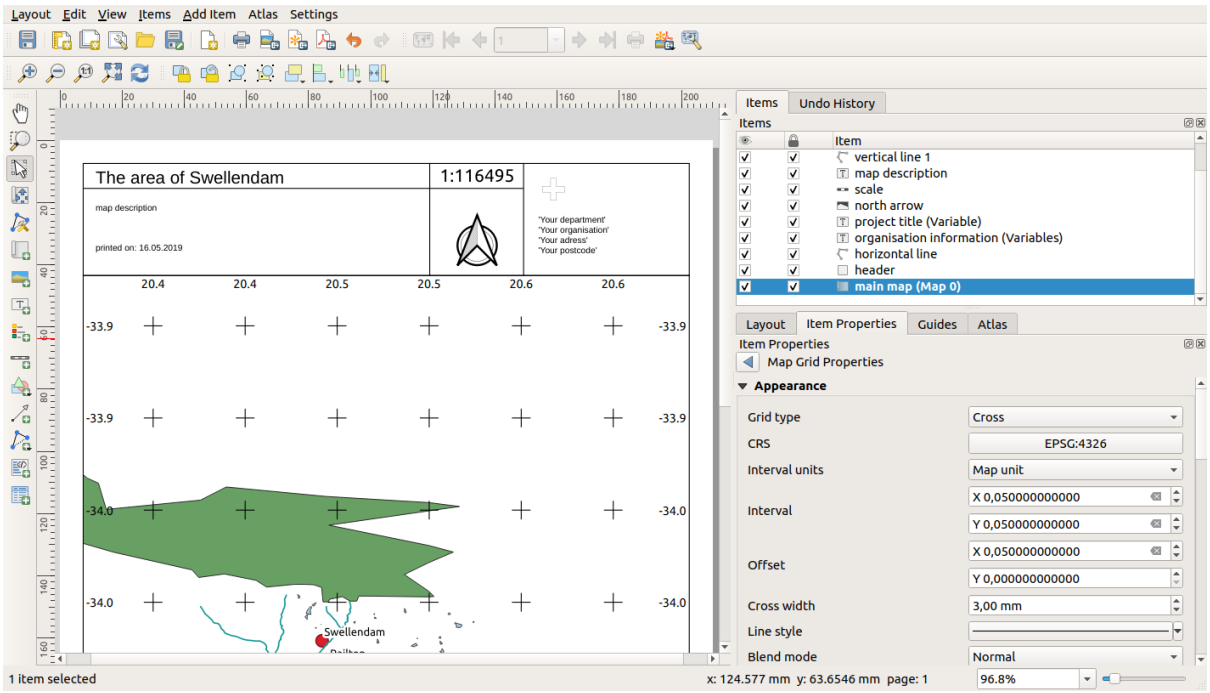
      ```
      @layout_pagewidth - @sw_layout_margin * 2 - 78
      ```

   2. For *Y*, use the expression:

      ```
      @sw_layout_margin + 1
      ```

   3. For *Width*, use `25`
   4. For *Height*, use `8`
   5. Place the `Reference point` in the center.

Congratulations! You have created your first dynamic map layout. Take a look at the layout and check if everything looks the way you want it! The dynamic map layout reacts automatically when you change the *page properties*. For example, if you change the page size from DIN A4 to DIN A3, click the Refresh view button and the page design is adapted.



4.2.6 What's Next?

On the next page, you will be given an assignment to complete. This will allow you to practice the techniques you have learned so far.

4.3 Tarefa 1

Abra o seu projecto de mapa e reveja-o na integra. Se encontrar pequenos erros ou detalhes que gostaria de ter melhorado, faça-o agora.

Enquanto estiver a personalizar o seu mapa, questione-se a si mesmo. O mapa é fácil de ler e de interpretar por alguém sem conhecimento prévio dos dados? Se visse este mapa na internet, num poster ou numa revista, iria chamar a minha atenção? Iria querer interpretar este mapa se ele não fosse de minha autoria?

If you're doing this course at a ★☆☆ Basic or ★★☆ Intermediate level, read up on techniques from the more advanced sections. If you see something you'd like to do in your map, why not try to implement it?

If this course is being presented to you, the course presenter may require you to submit a final version of your map, exported to PDF, for evaluation. If you're doing this course by yourself, it's recommended that you evaluate your own map using the same criteria. Your map will be evaluated on the overall appearance and symbology of the map itself, as well as the appearance and layout of the map page and elements. Remember that the emphasis for evaluating the appearance of maps will always be *ease of use*. The nicer the map is to look at and the easier it is to understand at a glance, the better.

Happy customizing!

4.3.1 In Conclusion

The first four modules have taught you how to create and style a vector map. In the next four modules, you'll learn how to use QGIS for a complete GIS analysis. This will include creating and editing vector data; analyzing vector data; using and analyzing raster data; and using GIS to solve a problem from start to finish, using both raster and vector data sources.

CAPÍTULO 5

Module: Creating Vector Data

Criar mapas utilizando dados existentes é apenas o começo. Neste módulo, irá aprender a alterar dados vetoriais existentes e criar novos conjuntos de dados.

5.1 Lesson: Creating a New Vector Dataset

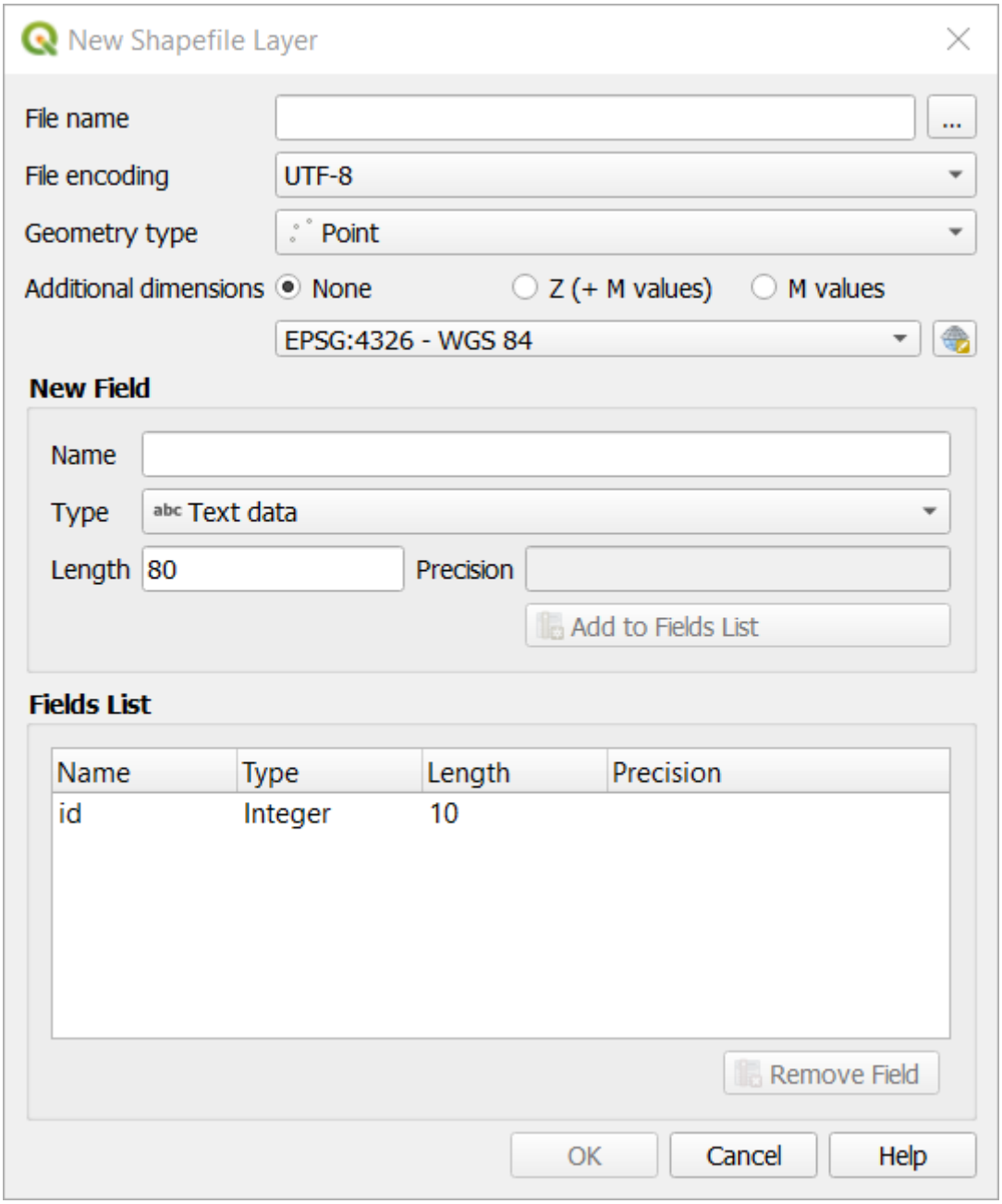
The data that you use has to come from somewhere. For most common applications, the data exists already; but the more particular and specialized the project, the less likely it is that the data will already be available. In such cases, you'll need to create your own new data.

The goal for this lesson: To create a new vector dataset.

5.1.1 ??? Follow Along: The Layer Creation Dialog

Before you can add new vector data, you need a vector dataset to add it to. In our case, you'll begin by creating new data entirely, rather than editing an existing dataset. Therefore, you'll need to define your own new dataset first.

1. Open QGIS and create a new blank project.
2. Navigate to and click on the menu entry *Layer* ► *Create Layer* ► *New Shapefile Layer*. You'll be presented with the *New Shapefile Layer* dialog, which will allow you to define a new layer.

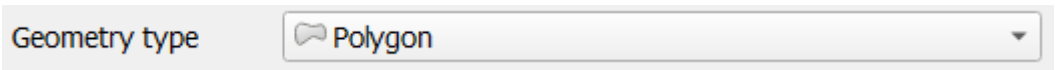


3. Click … for the *File name* field. A save dialog will appear.
4. Navigate to the `exercise_data` directory.
5. Save your new layer as `school_property.shp`.

   It's important to decide which kind of dataset you want at this stage. Each different vector layer type is «built differently» in the background, so once you've created the layer, you can't change its type.

   For the next exercise, we're going to create new features which describe areas. For such features, you'll need to create a polygon dataset.

6. For *Geometry Type*, select *Polygon* from the drop down menu:

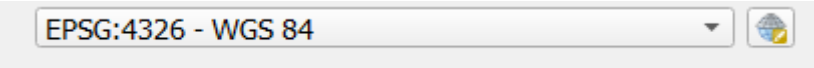


   This has no impact on the rest of the dialog, but it will cause the correct type of geometry to be used when the

vector dataset is created.

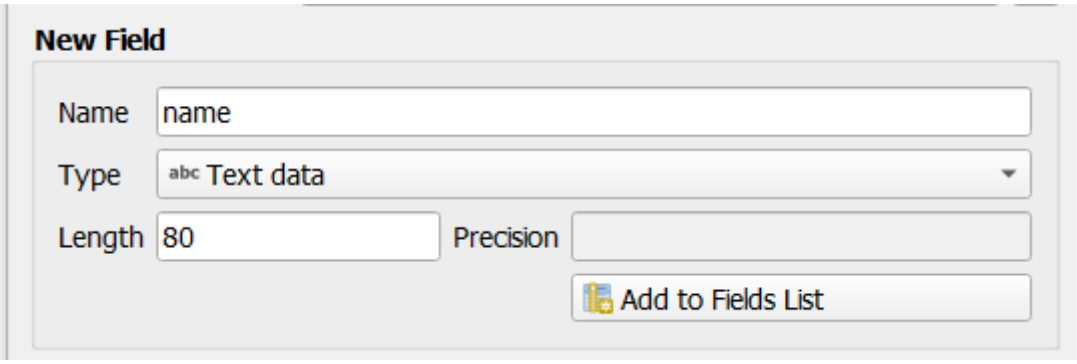
The next field allows you to specify the Coordinate Reference System, or CRS. CRS is a method of associating numerical coordinates with a position on the surface of the Earth. See the User Manual on Working with Projections to learn more.

For this example we will use the default CRS associated with this project, which is WGS84.

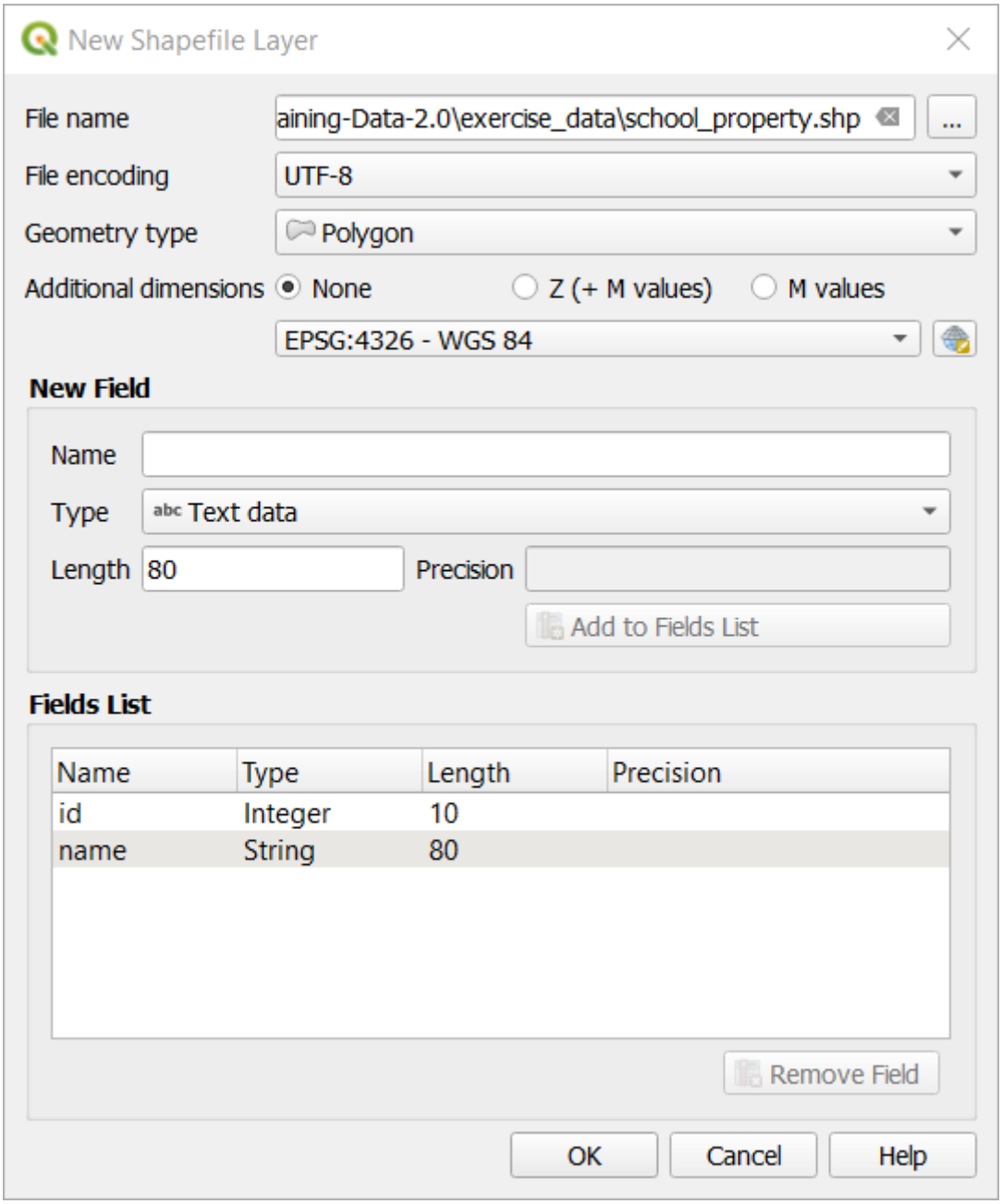


Next there is a collection of fields grouped under *New Field*. By default, a new layer has only one attribute, the `id` field (which you should see in the *Fields list*) below. However, in order for the data you create to be useful, you actually need to say something about the features you'll be creating in this new layer. For our current purposes, it will be enough to add one field called `name` that will hold `Text data` and will be limited to text length of `80` characters.

7. Replicate the setup below, then click the *Add to Fields List* button:



8. Check that your dialog now looks like this:



9. Click *OK*

The new layer should appear in your *Layers* panel.

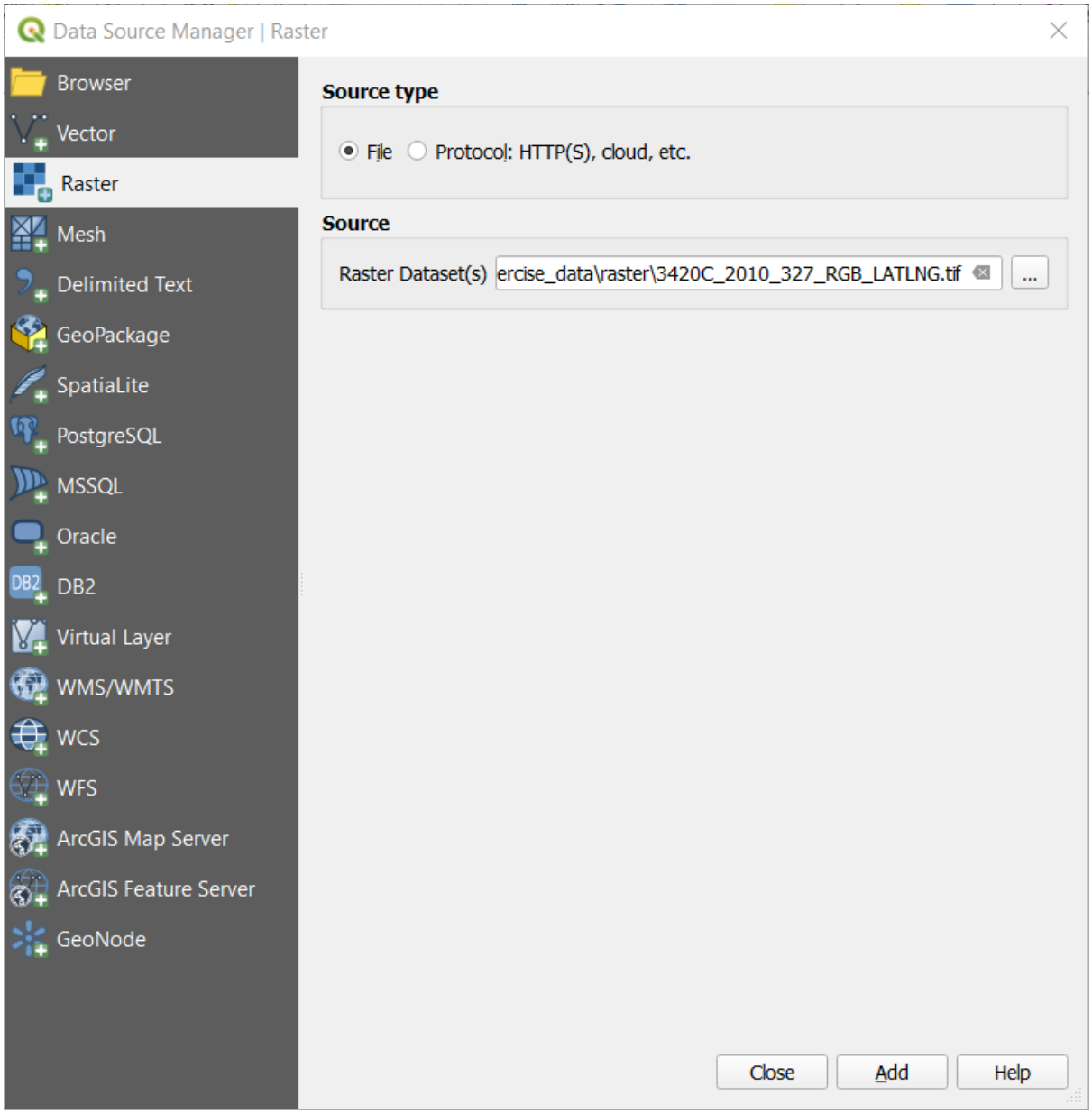
5.1.2 ??? Follow Along: Data Sources

When you create new data, it obviously has to be about objects that really exist on the ground. Therefore, you'll need to get your information from somewhere.

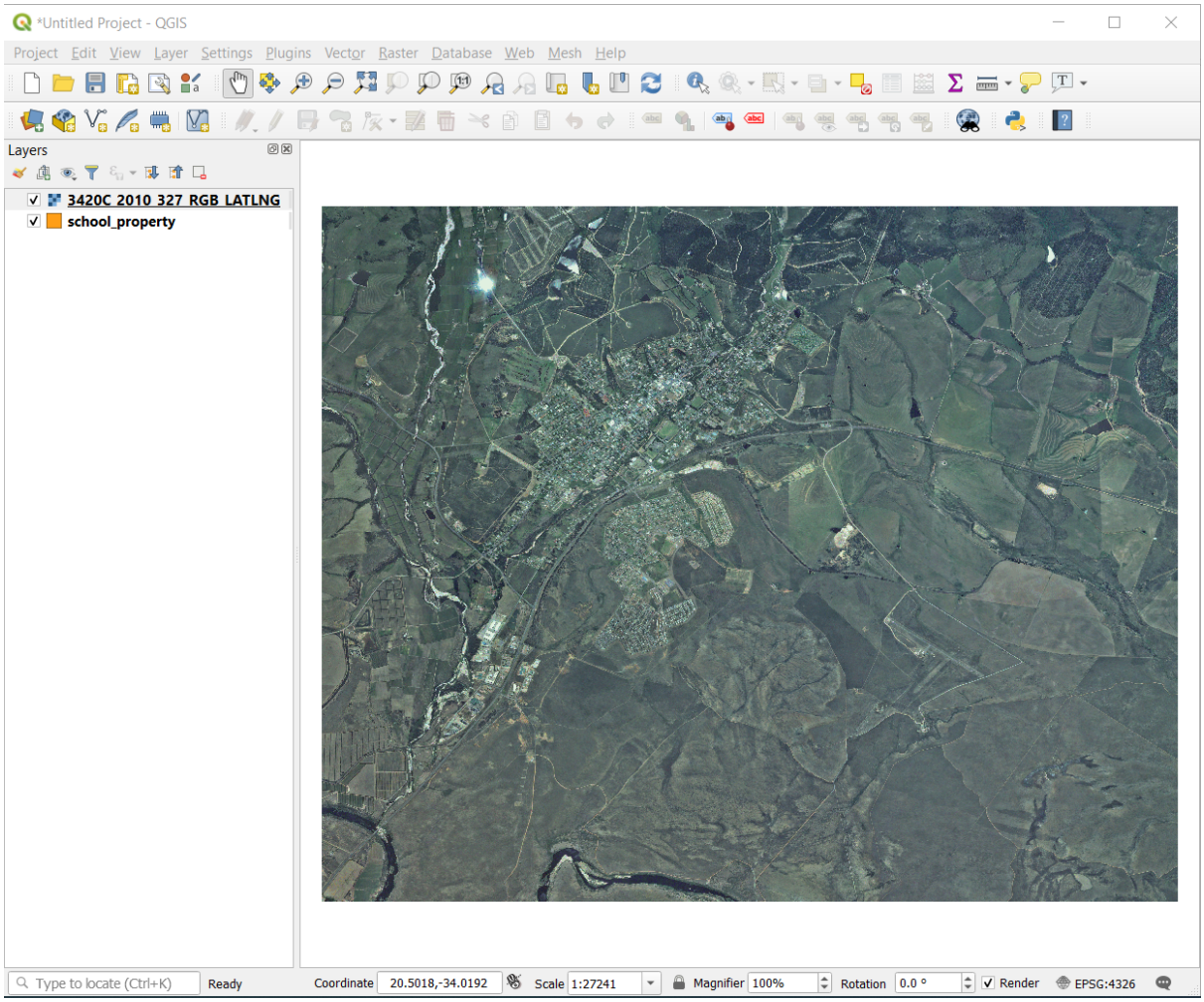
There are many different ways to obtain data about objects. For example, you could use a GPS to capture points in the real world, then import the data into QGIS afterwards. Or you could survey points using a theodolite, and enter the coordinates manually to create new features. Or you could use the digitizing process to trace objects from remote sensing data, such as satellite imagery or aerial photography.

For our example, you'll be using the digitizing approach. Sample raster datasets are provided, so you'll need to import them as necessary.

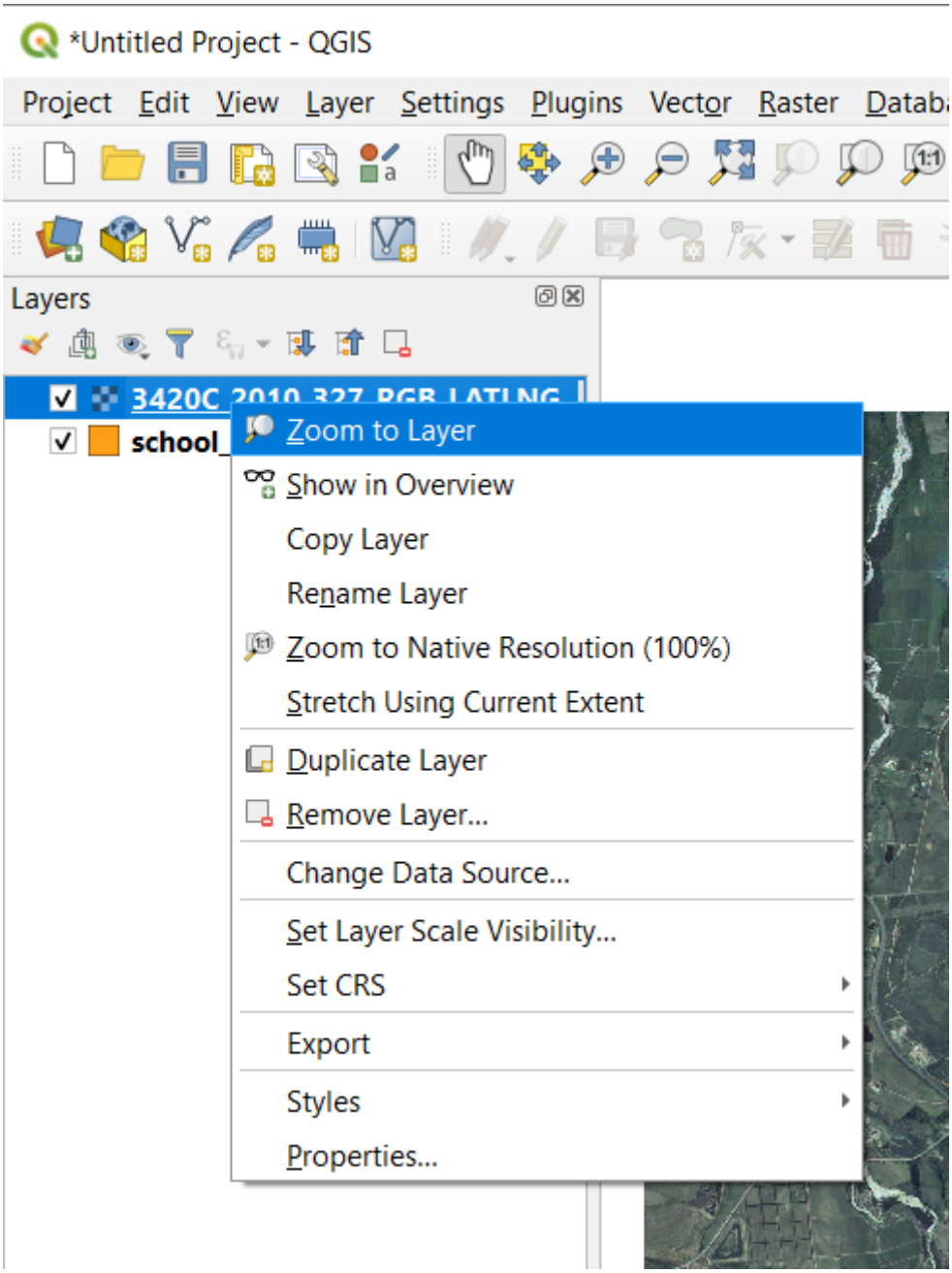
1. Click on Data Source Manager button.
2. Select *Raster* on the left side.
3. In the *Source* panel, click on the … button:
4. Navigate to `exercise_data/raster/`.
5. Select the file `3420C_2010_327_RGB_LATLNG.tif`.
6. Click *Open* to close the dialogue window.



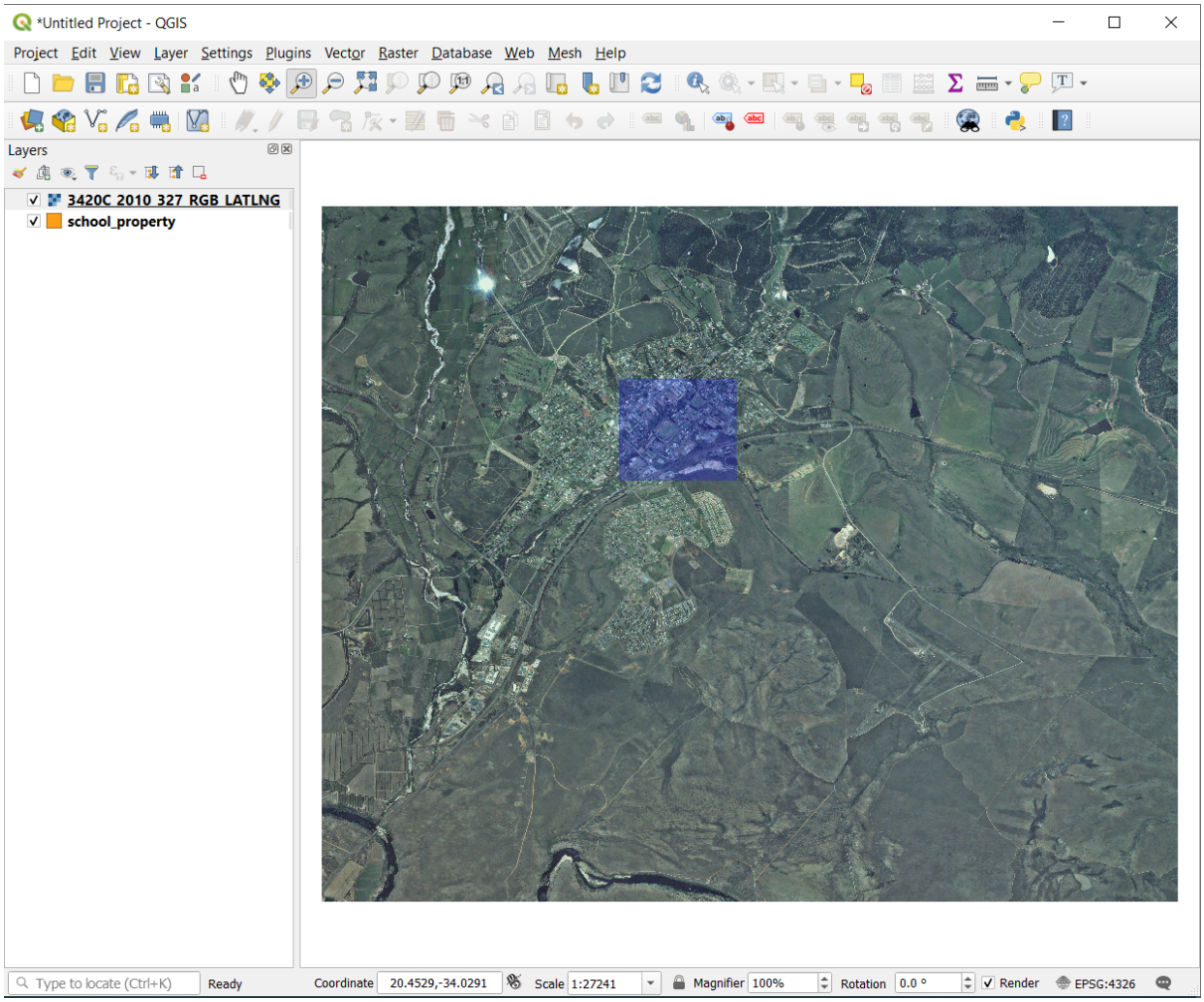
7. Click *Add* and *Close*. An image will load into your map.



8. If you don't see an aerial image appear, select the new layer, right click, and choose *Zoom to Layer* in the context menu.



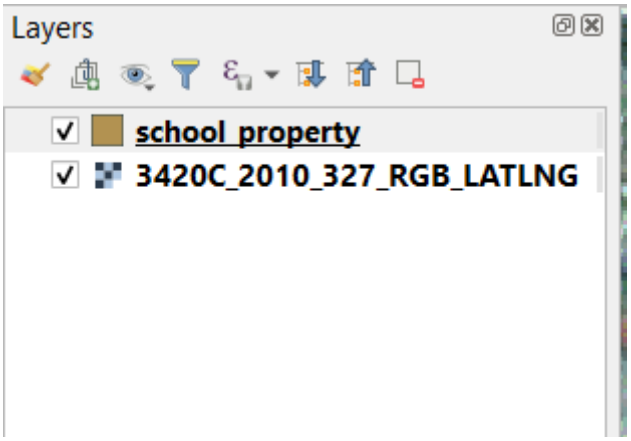
9. Click on the Zoom In button, and zoom to the area highlighted in blue below:



Now you are ready to digitize these three fields:

Before starting to digitize, let's move the `school_property` layer above the aerial image.

1. Select `school_property` layer in the *Layers* pane and drag it to the top.



In order to begin digitizing, you'll need to enter **edit mode**. GIS software commonly requires this to prevent you

from accidentally editing or deleting important data. Edit mode is switched on or off individually for each layer.

To enter edit mode for the `school_property` layer:

1. Click on the `school_property` layer in the *Layers* panel to select it.
2. Click on the Toggle Editing button.

   If you can't find this button, check that the *Digitizing* toolbar is enabled. There should be a check mark next to the *View* ► *Toolbars* ► *Digitizing* menu entry.

   As soon as you are in edit mode, you'll see that some digitizing tools have become active:

   - Capture Polygon
   - Vertex Tool

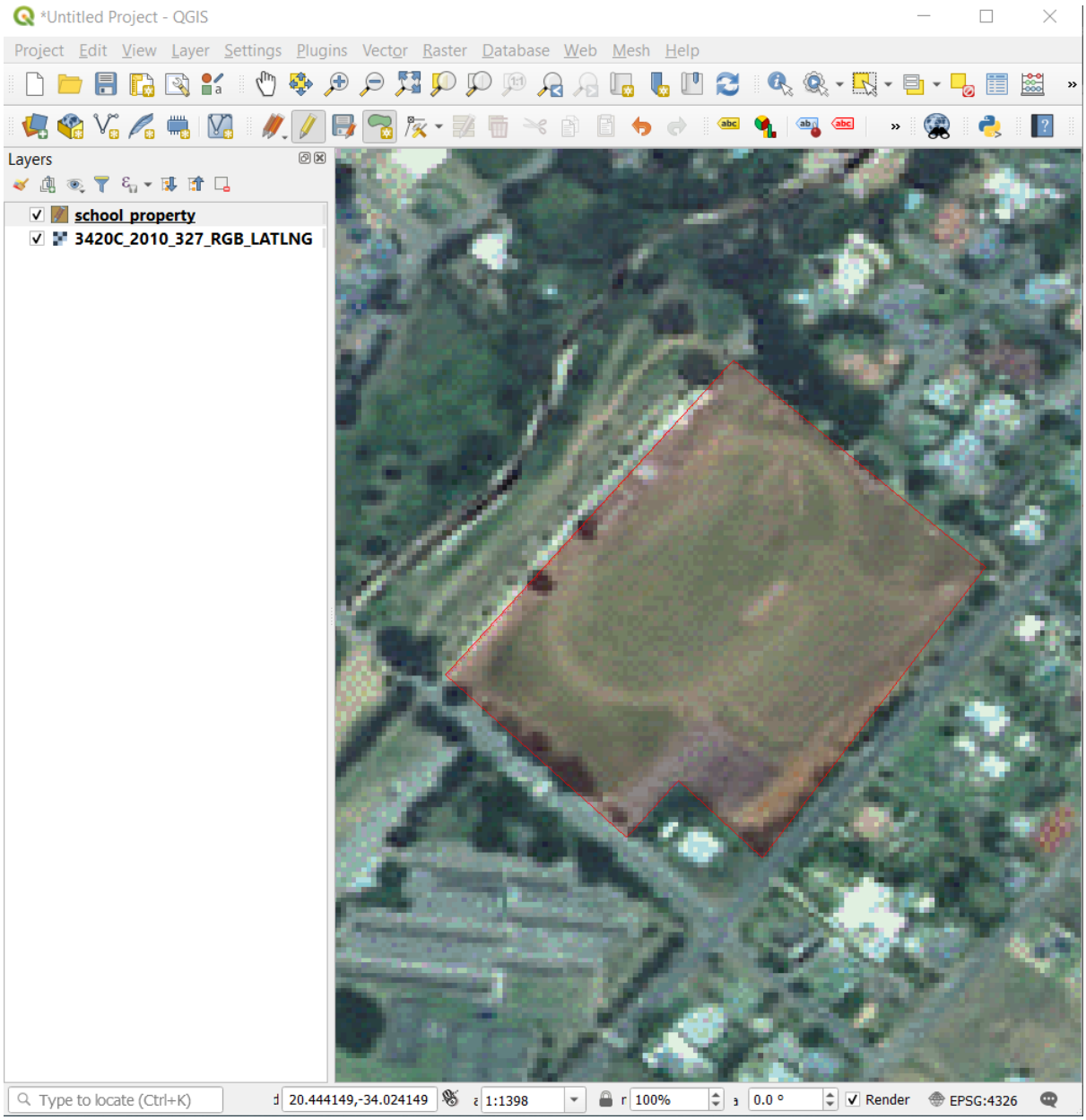
   Other relevant buttons are still inactive, but will become active when we start interacting with our new data.

   Notice that the layer `school_property` in the *Layers* panel now has the pencil icon, indicating that it is in edit mode.
3. Click on the Capture Polygon button to begin digitizing our school fields.

   You'll notice that your mouse cursor has become a crosshair. This allows you to more accurately place the points you'll be digitizing. Remember that even when you're using the digitizing tool, you can zoom in and out on your map by rolling the mouse wheel, and you can pan around by holding down the mouse wheel and dragging around in the map.

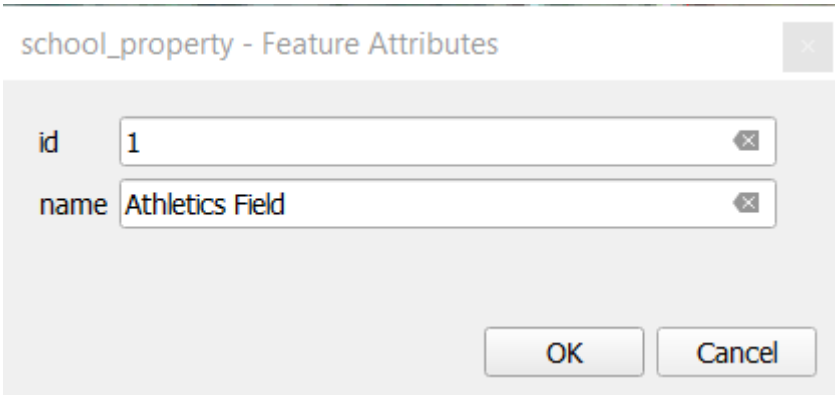
   The first feature you'll be digitizing is the athletics field:

4. Start digitizing by clicking on a point somewhere along the edge of the field.
5. Place more points by clicking further along the edge, until the shape you're drawing completely covers the field.

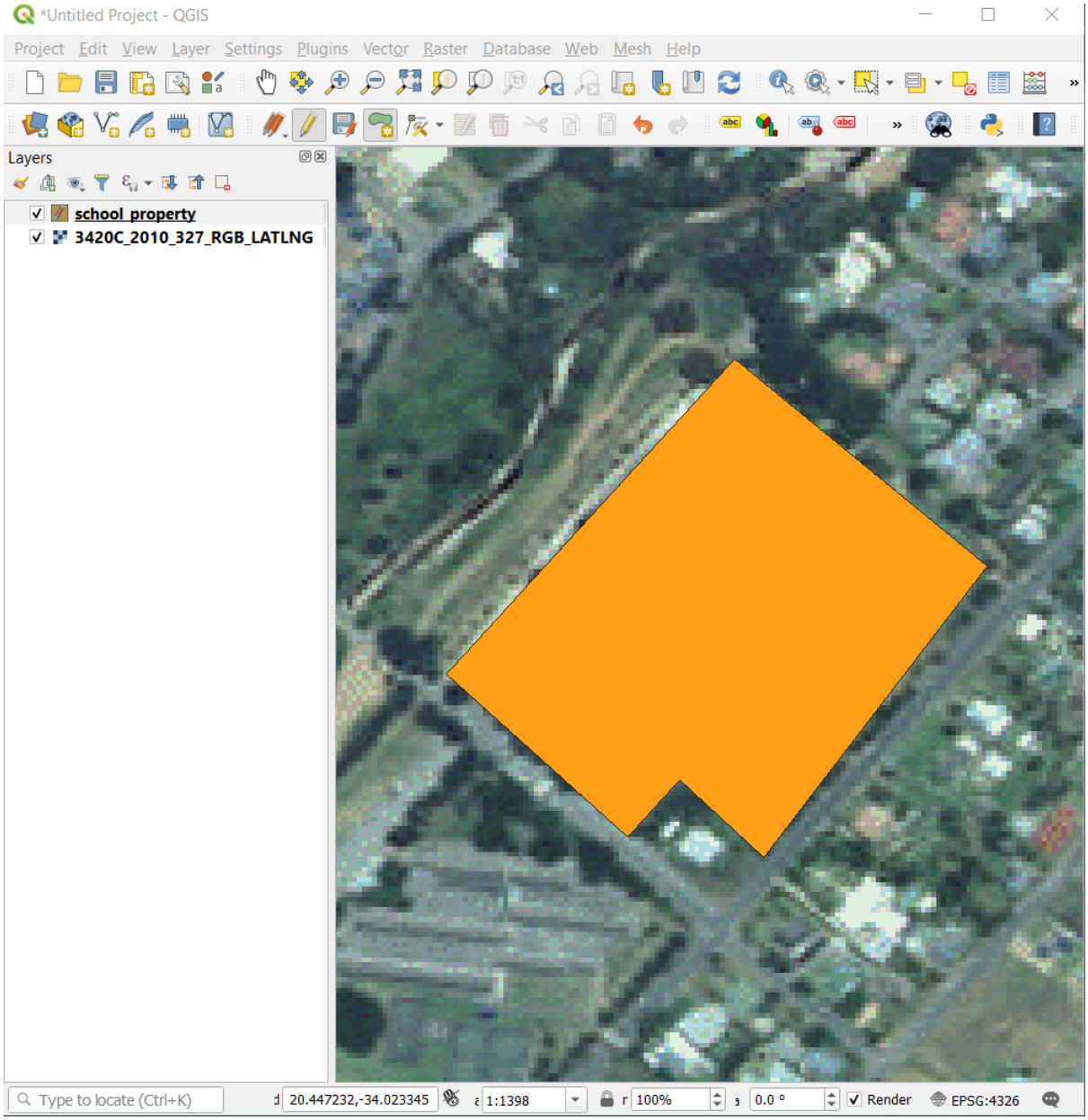


6. After placing your last point, right click to finish drawing the polygon. This will finalize the feature and show you the *Attributes* dialog.

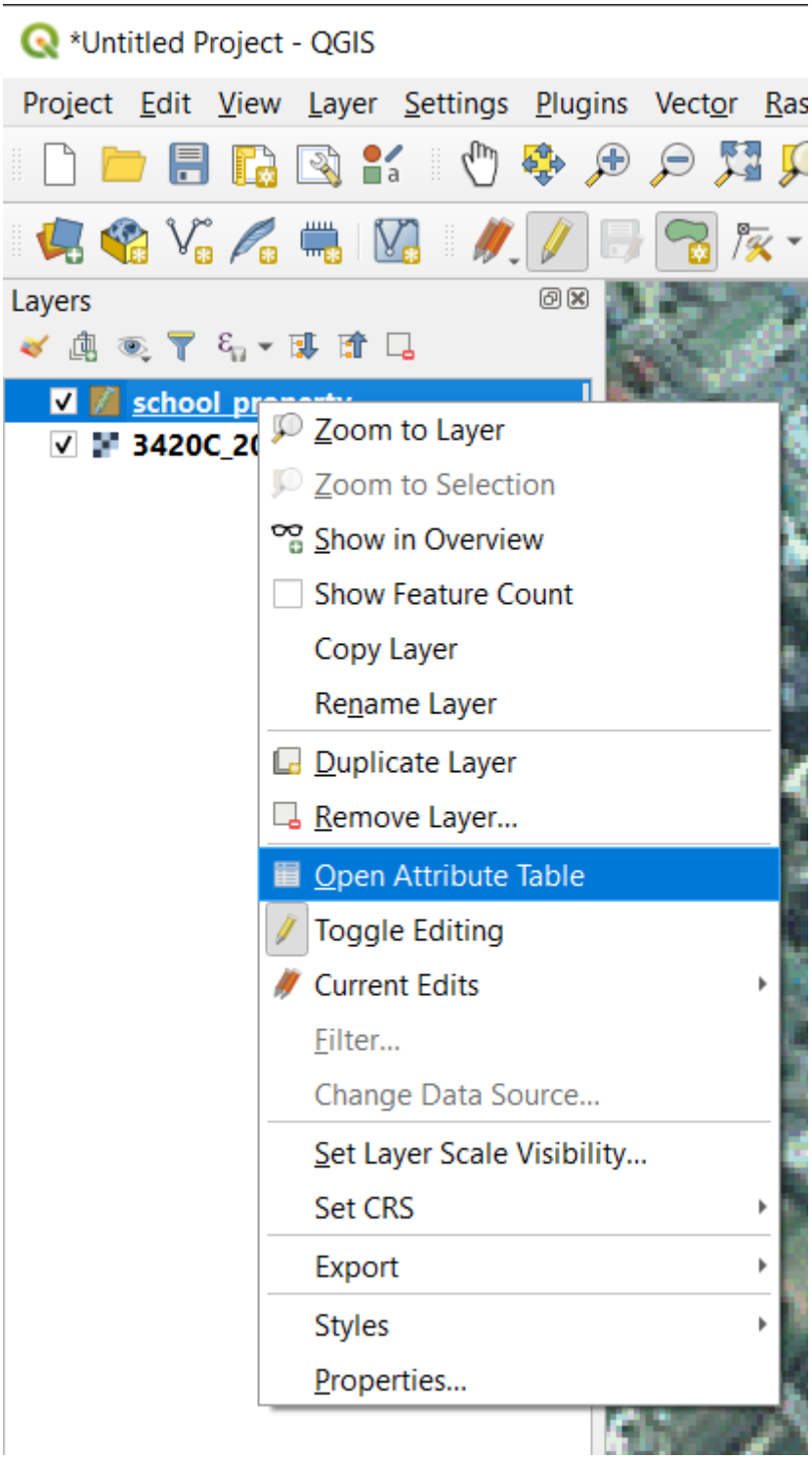
7. Fill in the values as below:



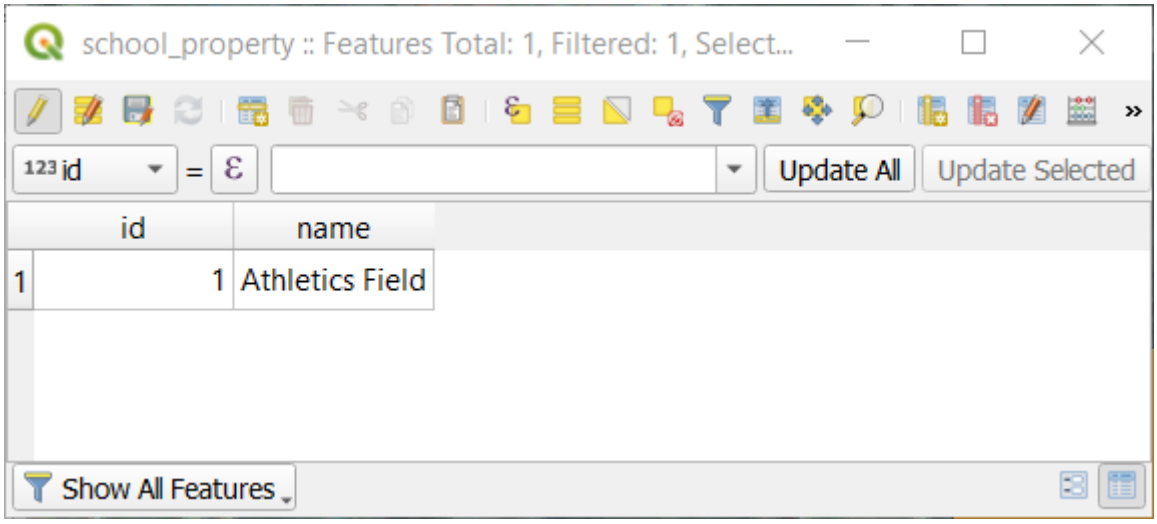
8. Click *OK*, and you have created a new feature!



9. In the *Layers* panel select the `school_property` layer.
10. Right click and choose *Open Attribute Table* in the context menu.



In the table you will see the feature you just added. While in edit mode you can update the attributes data by double click on the cell you want to update.



11. Close the attribute table.
12. To save the new feature we just created, click on Save Edits button.

Remember, if you've made a mistake while digitizing a feature, you can always edit it after you're done creating it. If you've made a mistake, continue digitizing until you're done creating the feature as above. Then:

1. Click on Vertex Tool button.
2. Hover the mouse over a vertex you want to move and left click on the vertex.
3. Move the mouse to the correct location of the vertex, and left click. This will move the vertex to the new location.

The same procedure can be used to move a line segment, but you will need to hover over the midpoint of the line segment.

If you want to undo a change, you can press the Undo button or `Ctrl+Z`.

4. Remember to save your changes by clicking the Save Edits button.
5. When done editing, click the Toggle Editing button to get out of edit mode.

5.1.3 ??? Try Yourself: Digitizing Polygons

Digitize the school itself and the upper field. Use this image to assist you:

Remember that each new feature needs to have a unique `id` value!

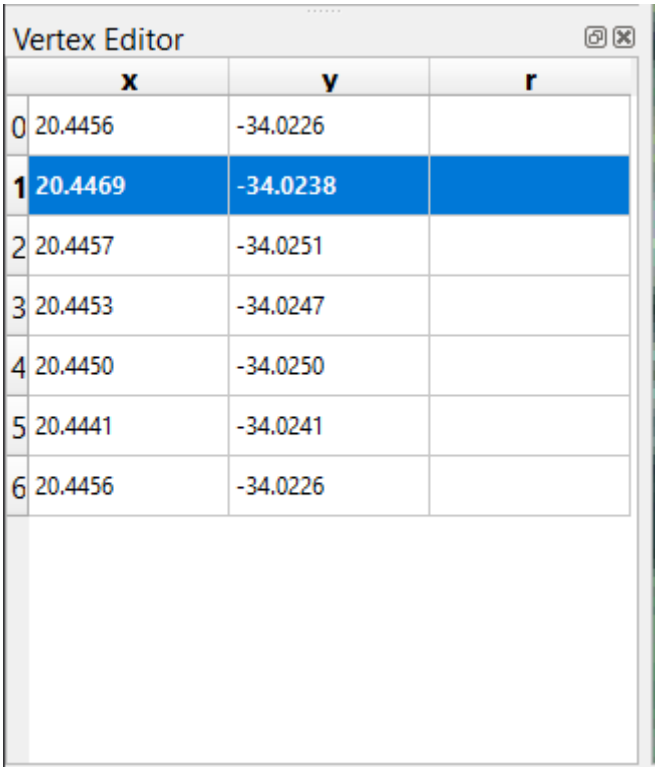
Nota: When you're done adding features to a layer, remember to save your edits and then exit edit mode.

Nota: You can style the fill, outline and label placement and formatting of the `school_property` using techniques learnt in earlier lessons.

5.1.4 ??? Follow Along: Using Vertex Editor Table

Another way to edit a feature is to manually enter the actual coordinate values for each vertex using the *Vertex Editor* table.

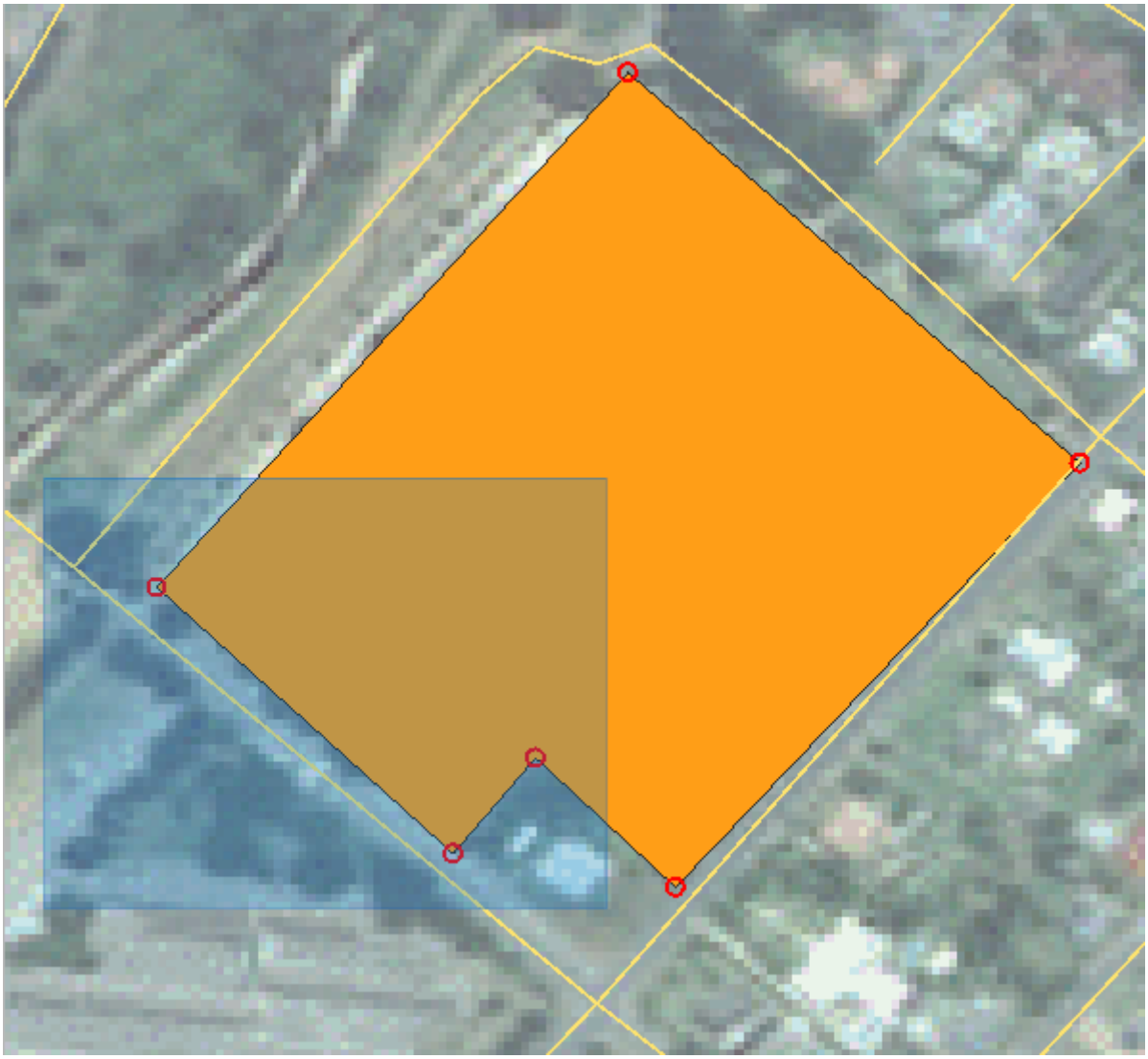
1. Make sure you are in edit mode on layer `school_property`.
2. If not already activated, click on Vertex Tool button.
3. Move the mouse over one of the polygon features you created in the `school_property` layer and right click on it. This will select the feature and a *Vertex Editor* pane will appear.



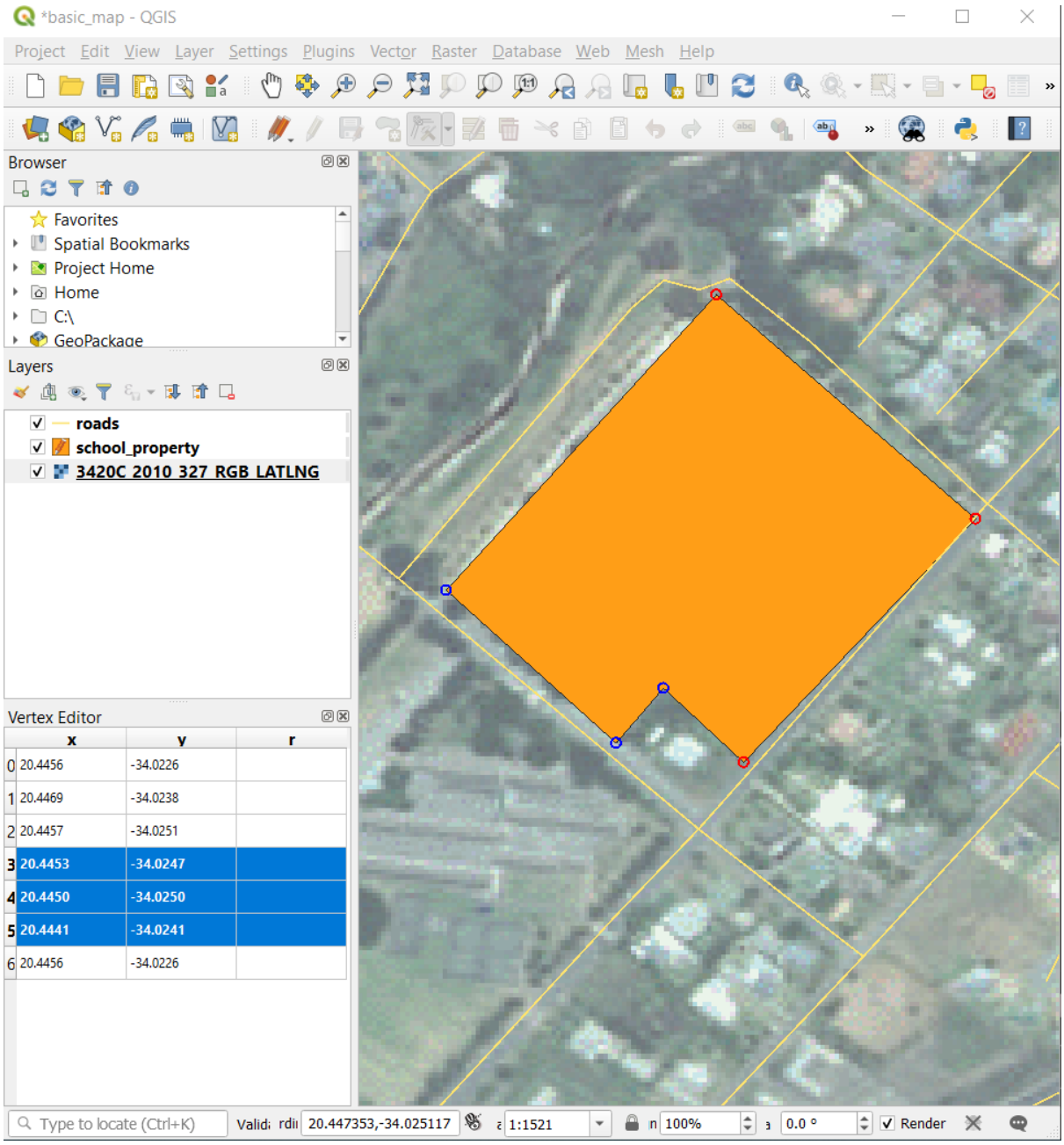
Nota: This table contains the coordinates for the vertices of the feature. Notice there are seven vertices for this feature, but only six are visually identified in the map area. Upon closer inspection, one will notice that row 0 and 6 have identical coordinates. These are the start and end vertices of the feature geometry, and are required in order to create a closed polygon feature.

---

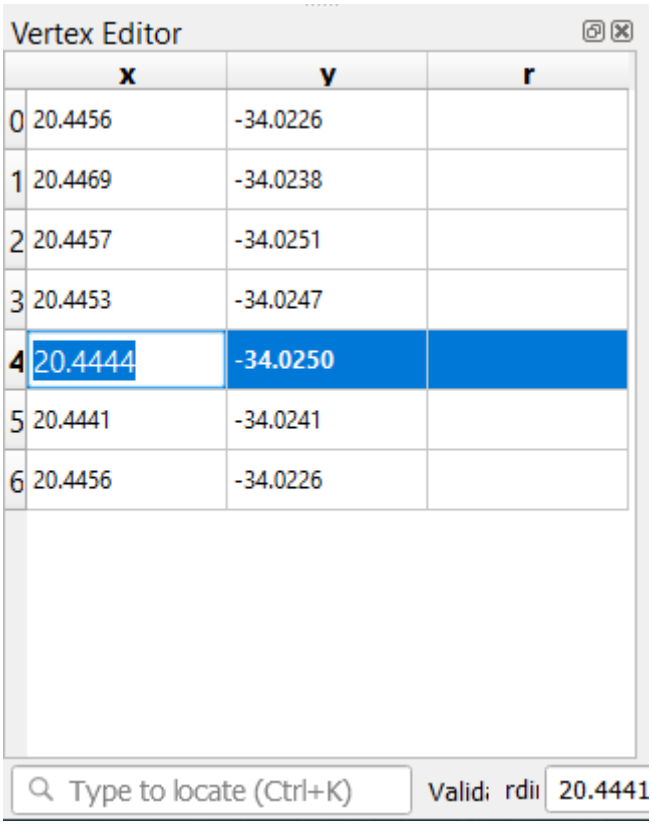
4. Click and drag a box over a vertex, or multiple vertices, of the selected feature.



The selected vertices will change to a color blue and the *Vertex Editor* table will have the corresponding rows highlighted, which contain the coordinates of the vertices.



5. To update a coordinate, double left click on the cell in the table that you want to edit and enter the updated value. In this example, the x coordinate of row `4` is updated from `20.4450` to `20.4444`.



6. After entering the updated value, hit the enter key to apply the change. You will see the vertex move to the new location in the map window.

7. When done editing, click the Toggle Editing button to get out of edit mode, and save your edits.

5.1.5 ??? Try Yourself: Digitizing Lines

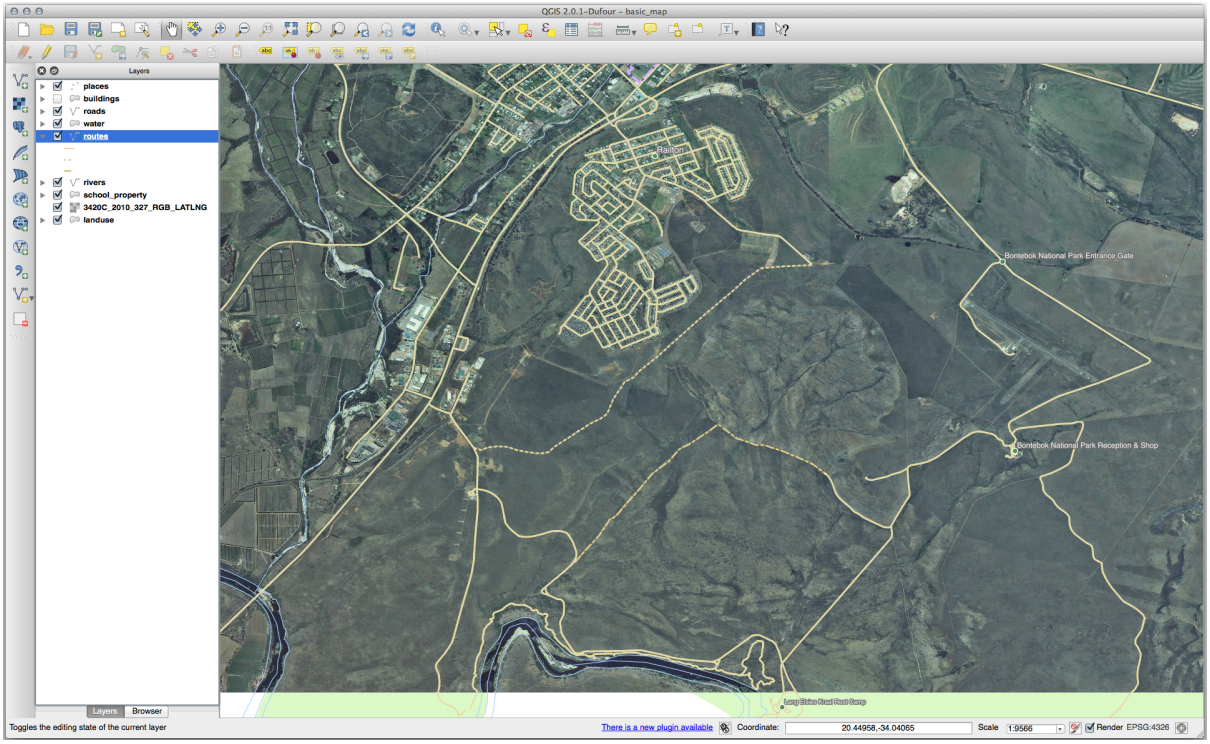
We are going to digitize two routes which are not already marked on the roads layer; one is a path, the other is a track. Our path runs along the southern edge of the suburb of Railton, starting and ending at marked roads:

Our track is a little further to the south:

1. If the *roads* layer is not yet in your map, then add the `roads` layer from the GeoPackage file `training-data.gpkg` included in the `exercise_data` folder of the training data you downloaded. You can read *Follow Along: Loading vector data from a GeoPackage Database* for a how-to.
2. Create a new ESRI Shapefile line dataset called `routes.shp` in the `exercise_data` directory, with attributes `id` and `type` (use the approach above to guide you.)
3. Activate edit mode on the *routes* layer.
4. Since you are working with a line feature, click on the Add Line button to initiate line digitizing mode.
5. One at a time, digitize the path and the track on the `routes` layer. Try to follow the routes as accurately as possible, adding additional points along corners or turns.
6. Set the `type` attribute value to `path` or `track`.
7. Use the *Layer Properties* dialog to add styling to your routes. Feel free to use different styles for paths and tracks.
8. Save your edits and toggle off editing mode by pressing the Toggle Editing button.

Answer

The symbology doesn't matter, but the results should look more or less like this:



5.1.6 In Conclusion

Now you know how to create features! This course doesn't cover adding point features, because that's not really necessary once you've worked with more complicated features (lines and polygons). It works exactly the same, except that you only click once where you want the point to be, give it attributes as usual, and then the feature is created.

Knowing how to digitize is important because it's a very common activity in GIS programs.

5.1.7 What's Next?

Features in a GIS layer aren't just pictures, but objects in space. For example, adjacent polygons know where they are in relation to one another. This is called **topology**. In the next lesson you'll see an example of why this can be useful.

5.2 Lesson: Feature Topology

Topology is a useful aspect of vector data layers, because it minimizes errors such as overlap or gaps.

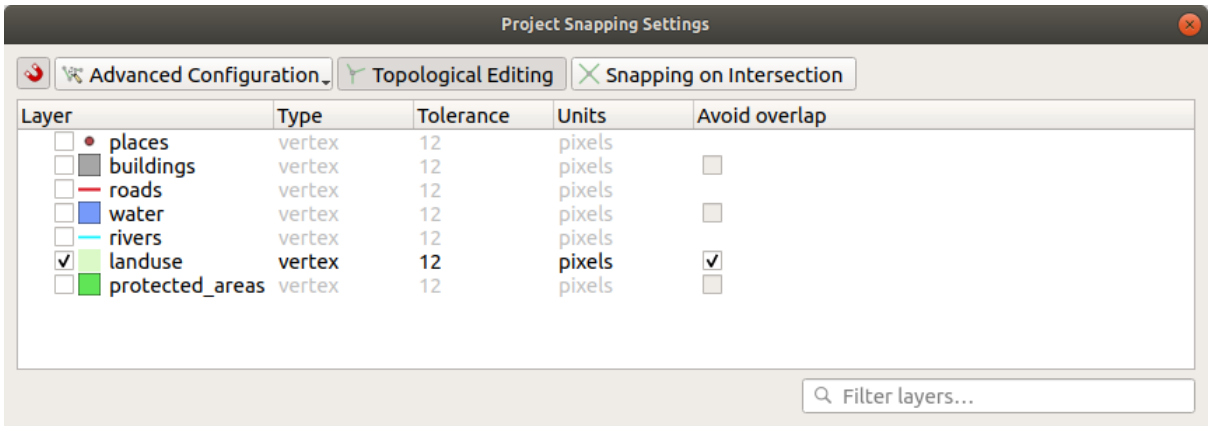
For example: if two features share a border, and you edit the border using topology, then you won't need to edit first one feature, then another, and carefully line up the borders so that they match. Instead, you can edit their shared border and both features will change at the same time.

The goal for this lesson: To understand topology using examples.

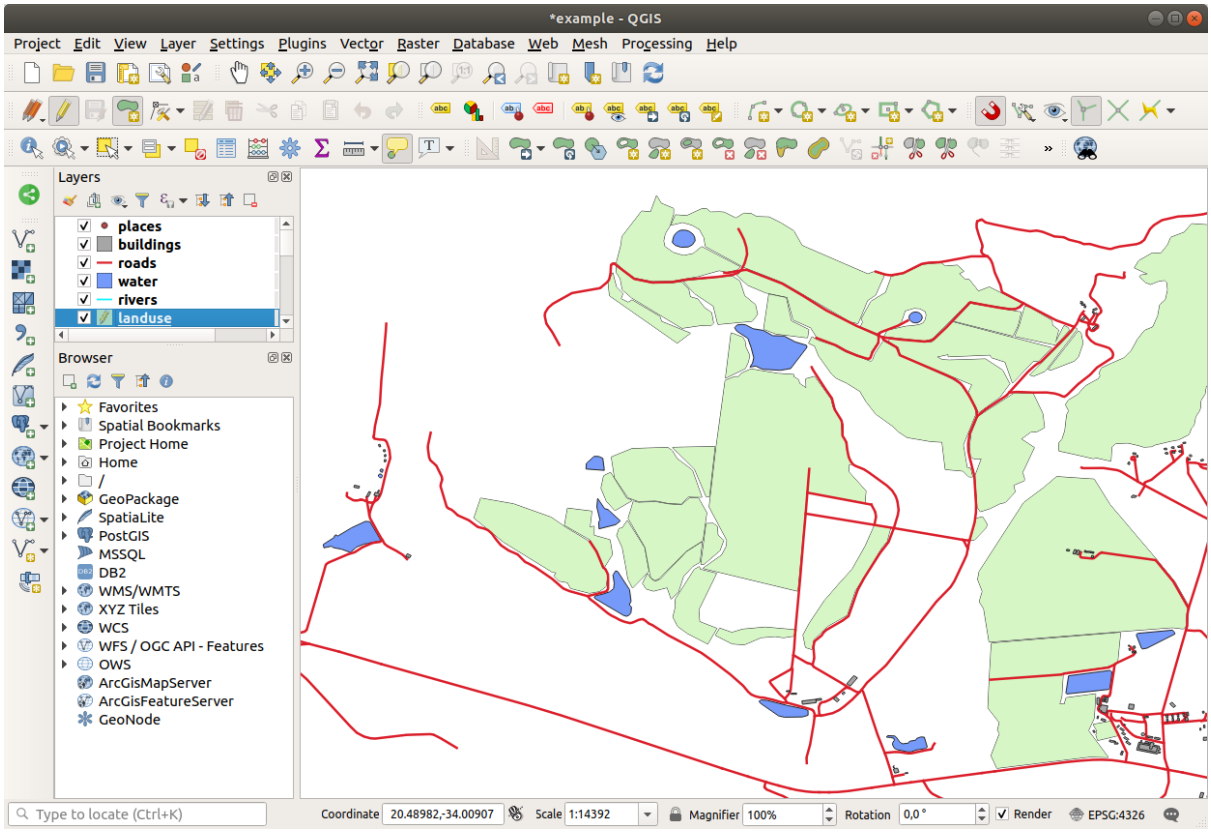
5.2.1 ??? Follow Along: Snapping

Snapping makes topological editing easier. This will allow your mouse cursor to snap to other objects while you digitize. To set snapping options:

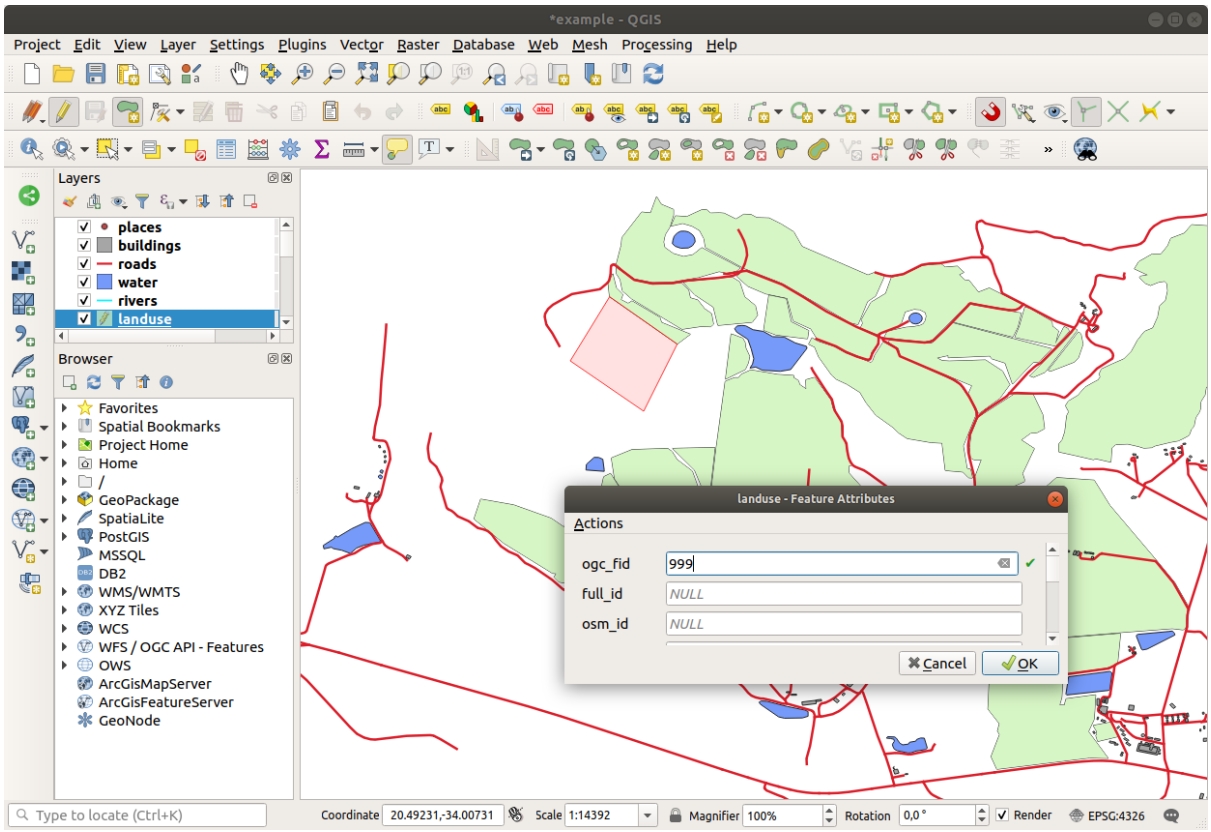
1. Navigate to the menu entry *Project* ► *Snapping Options…*.
2. Set up your *Snapping options* dialog to activate the `landuse` layer with *Type vertex* and tolerance `12` pixels:



3. Make sure that the box in the *Avoid overlap* column is checked.
4. Leave the dialog.
5. Select the *landuse* layer and enter edit mode ( )
6. Check (under *View* ► *Toolbars*) that the *Advanced Digitizing* toolbar is enabled.
7. Zoom to this area (enable layers and labels if necessary):



8. Digitize this new (fictional) area shown in red:



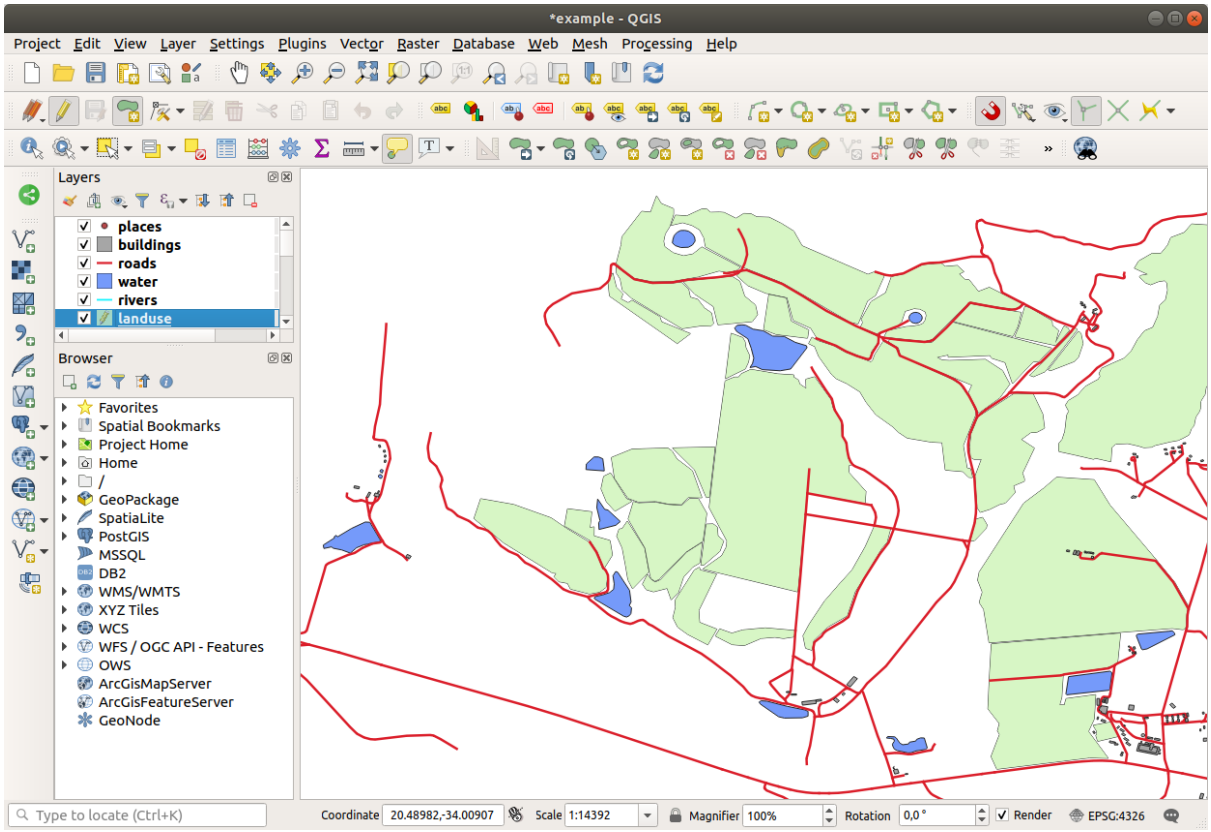
9. When prompted, give it an *OGC_FID* of `999`, but feel free to leave the other values unchanged.

If you are careful while digitizing, and allow the cursor to snap to the vertices of adjoining areas, you'll notice that there won't be any gaps between your new area and the existing adjacent areas.

10. Note the undo and redo tools in the *Advanced Digitizing* toolbar.

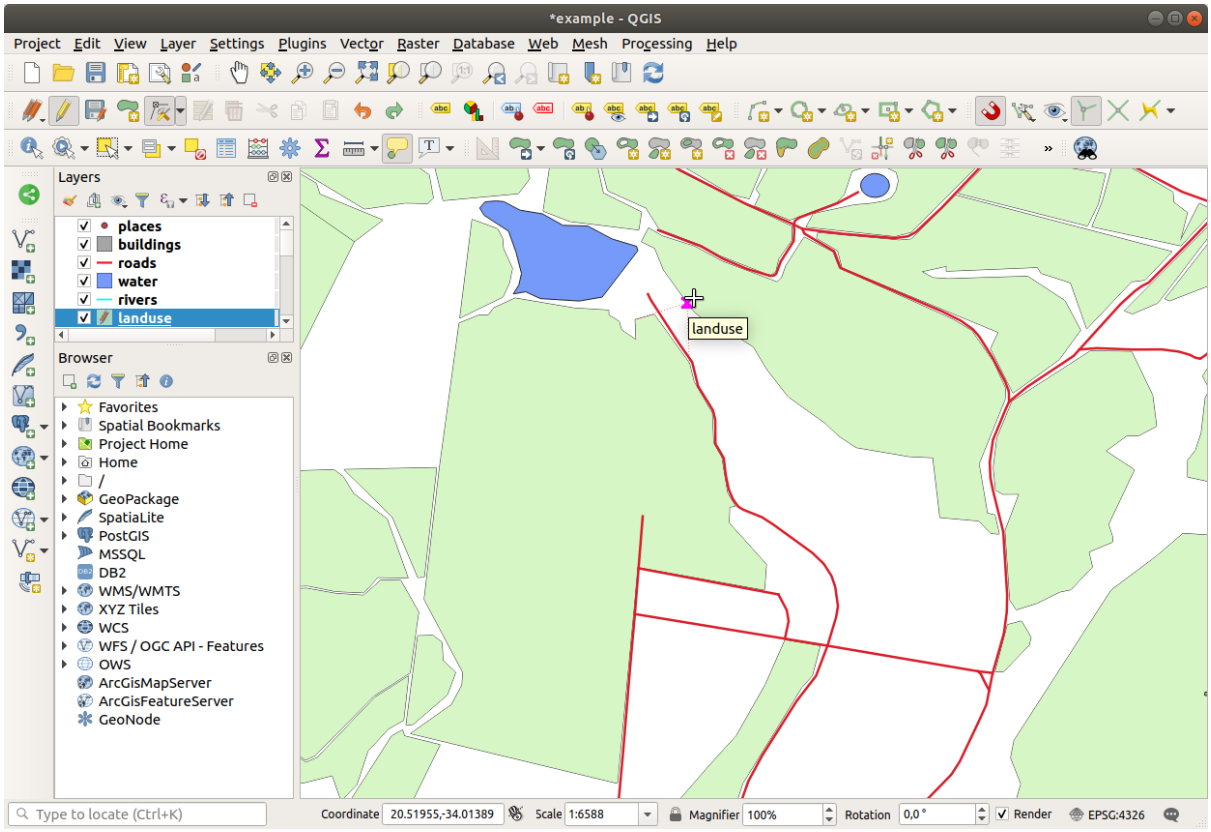
5.2.2 ??? Follow Along: Correct Topological Features

Topology features can sometimes need to be updated. In our study area, an area has been turned into forest, so the `landuse` layer need an update. We will therefore expand and join some forest features in this area:



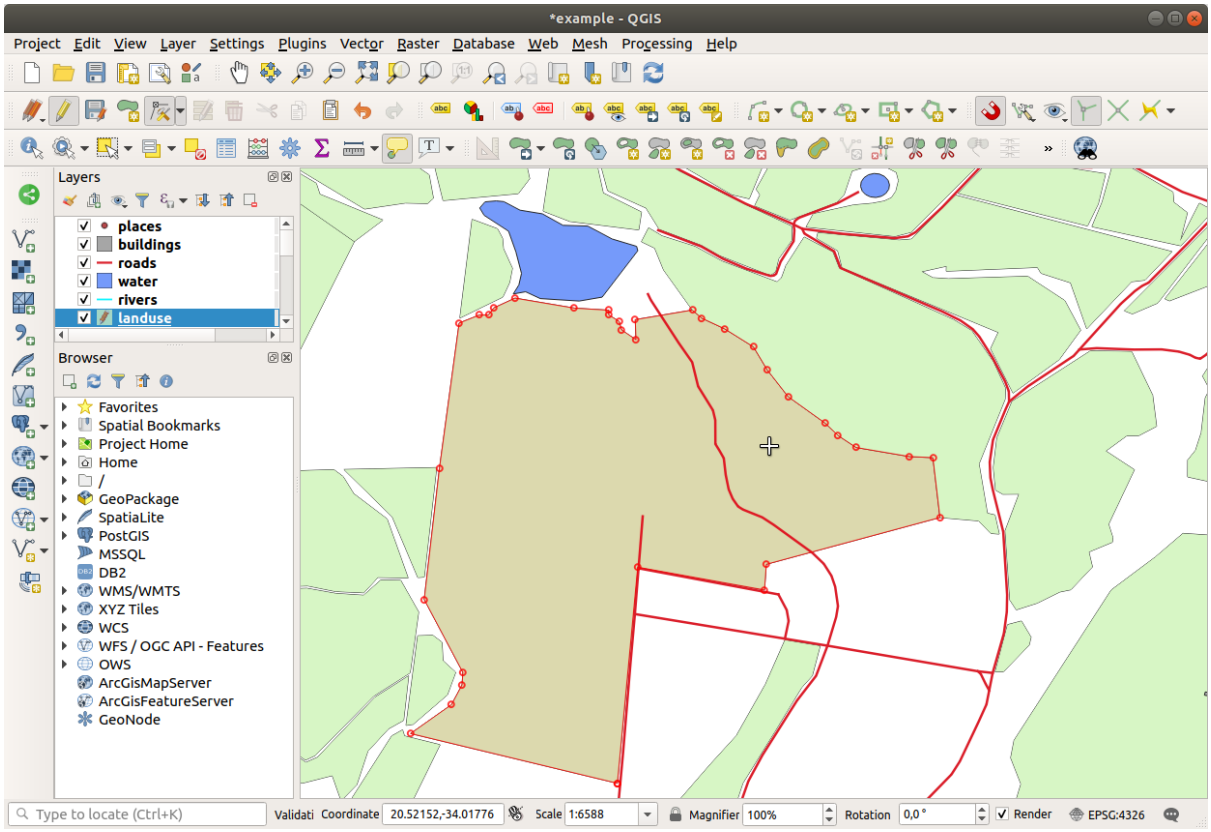
Instead of creating new polygons to join the forest areas, we are going to use the *Vertex Tool* to edit and join existing polygons.

1. Enter edit mode (if it is not active already)
2. Select the Vertex Tool tool.
3. Choose an area of forest, select a vertex, and move it to an adjoining vertex so that the two forest features meet:



4. Click on the other vertices and snap them into place.

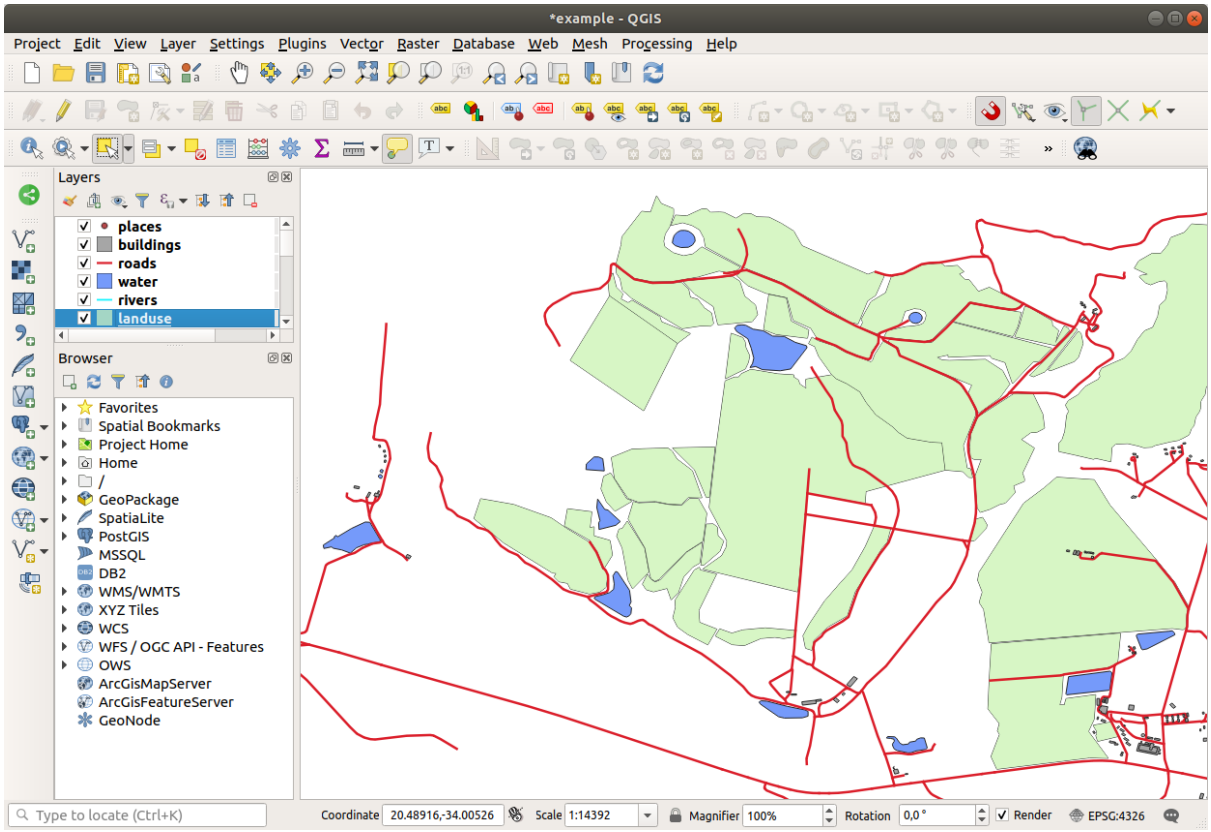
   The topologically correct border looks like this:



Go ahead and join a few more areas using the *Vertex Tool*.

You can also use the Add Polygon Feature tool to fill the gap between the two forest polygons. If you have enabled *Avoid overlap*, you don't have to add every single vertex - they will be added automatically if your new polygon overlaps the existing ones.

If you are using our example data, you should have a forest area looking something like this:

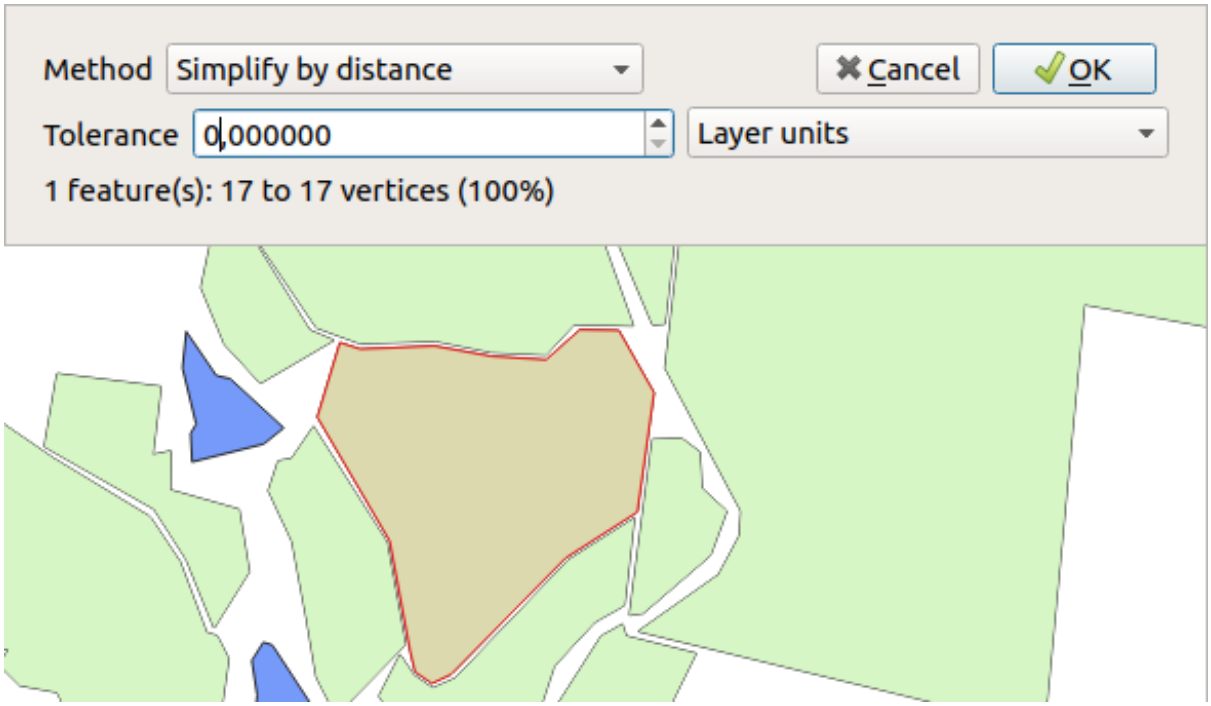


Don't worry if you have joined more, less or different areas of forest.

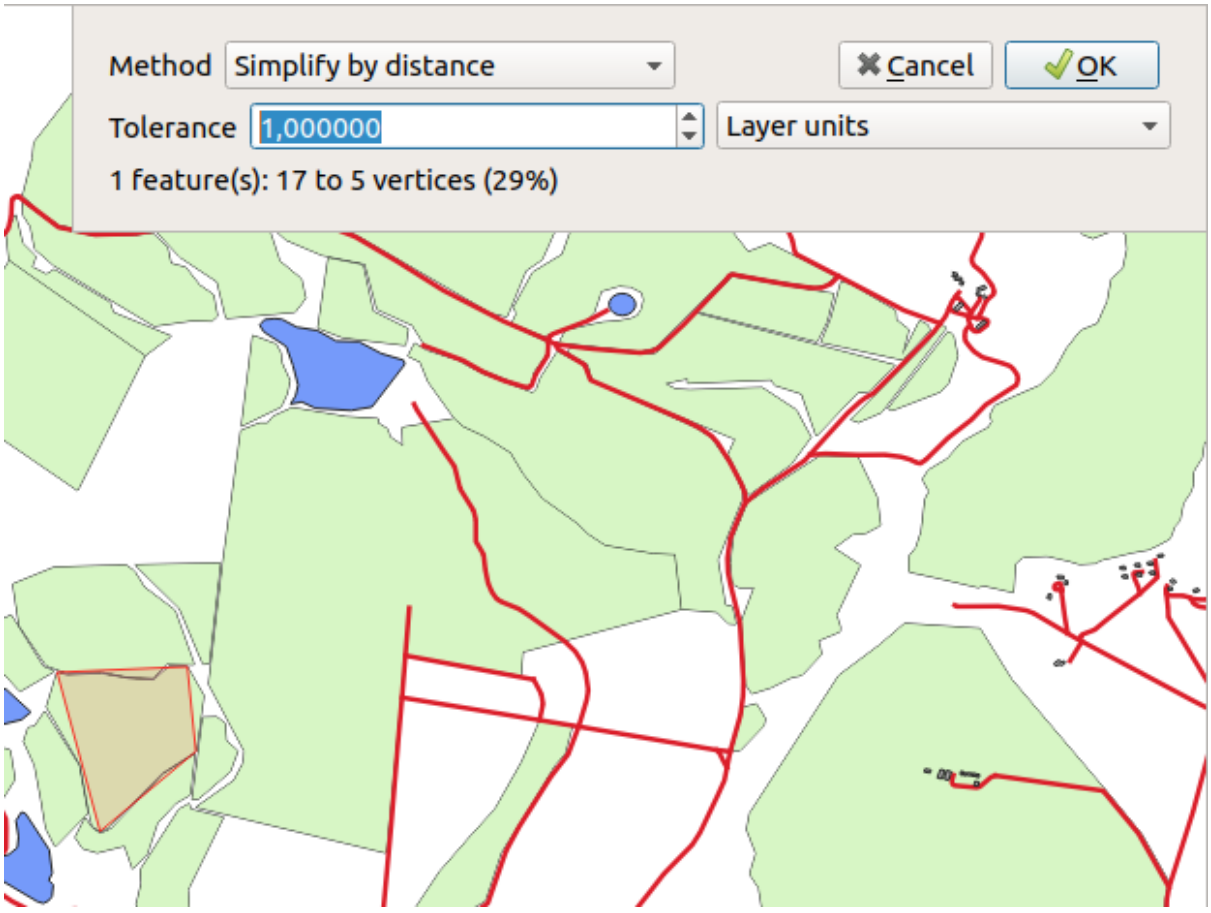
5.2.3 ??? Follow Along: Tool: Simplify Feature

Continuing on the same layer, we will test the Simplify Feature tool:

1. Click on it to activate it.
2. Click on one of the areas which you joined using either the *Vertex Tool* or *Add Feature* tool. You will see this dialog:



3. Modify the *Tolerance* and watch what happens:



This allows you to reduce the number of vertices.

4. Click *OK*

The advantage of this tool is that it provides you with a simple and intuitive interface for generalization. But notice that the tool ruins topology. The simplified polygon no longer shares boundaries with its adjacent polygons, as it should. So this tool is better suited for stand-alone features.

Before you go on, set the polygon back to its original state by undoing the last change.

5.2.4 ??? Try Yourself: Tool: Add Ring

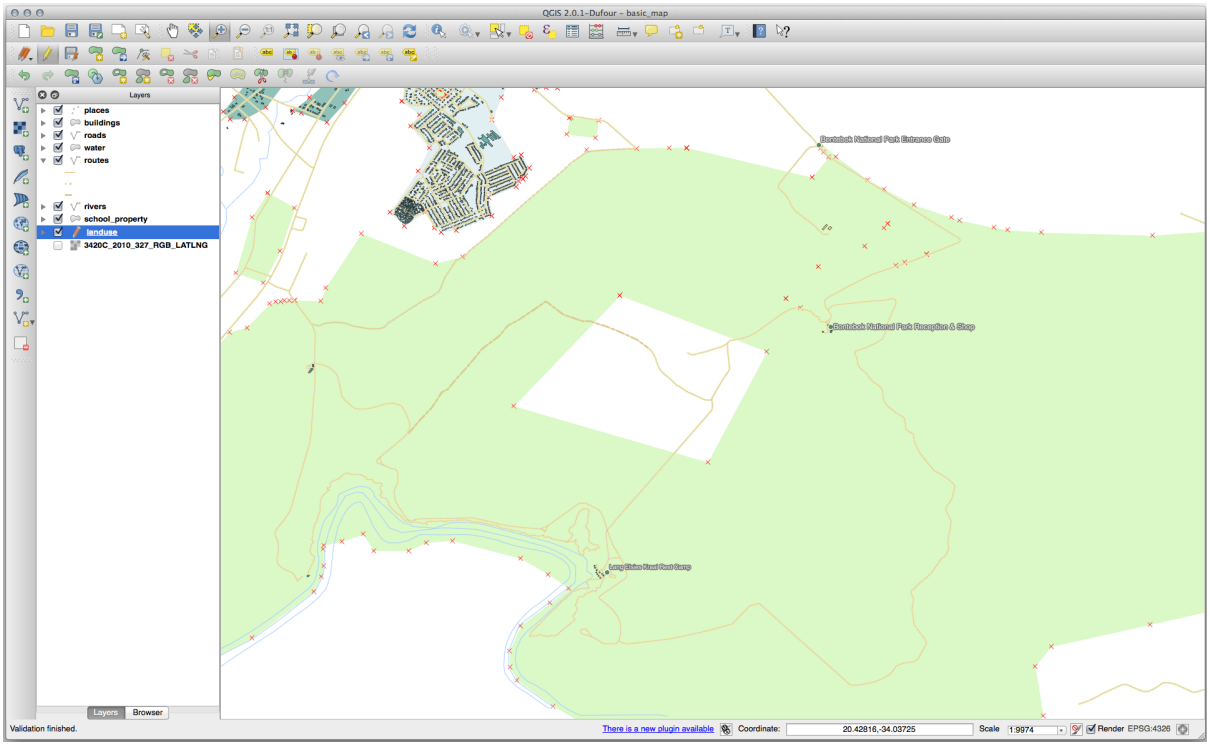
The Add Ring tool allows you to add an interior ring to a polygon feature (cut a hole in the polygon), as long as the hole is completely contained within the polygon (touching the boundary is OK). For example, if you have digitized the outer boundaries of South Africa and you need to add a hole for Lesotho, you would use this tool.

If you experiment with the tool, you may notice that the snapping options can prevent you from creating a ring inside a polygon. So you are advised to turn off snapping before cutting a hole.

1. Disable snapping for the `landuse` layer using the Enable Snapping button (or use the shortcut `s`).
2. Use the Add Ring tool to create a hole in the middle of a polygon geometry.
3. Draw a polygon over the target feature, as if you were using the Add polygon tool.
4. When you right-click, the hole will be visible.
5. Remove the hole you just created using the Delete Ring tool. Click inside the hole to delete it.

Answer

The exact shape doesn't matter, but you should be getting a hole in the middle of your feature, like this one:



- Undo your edit before continuing with the exercise for the next tool.

5.2.5 ??? Try Yourself: Tool: Add Part

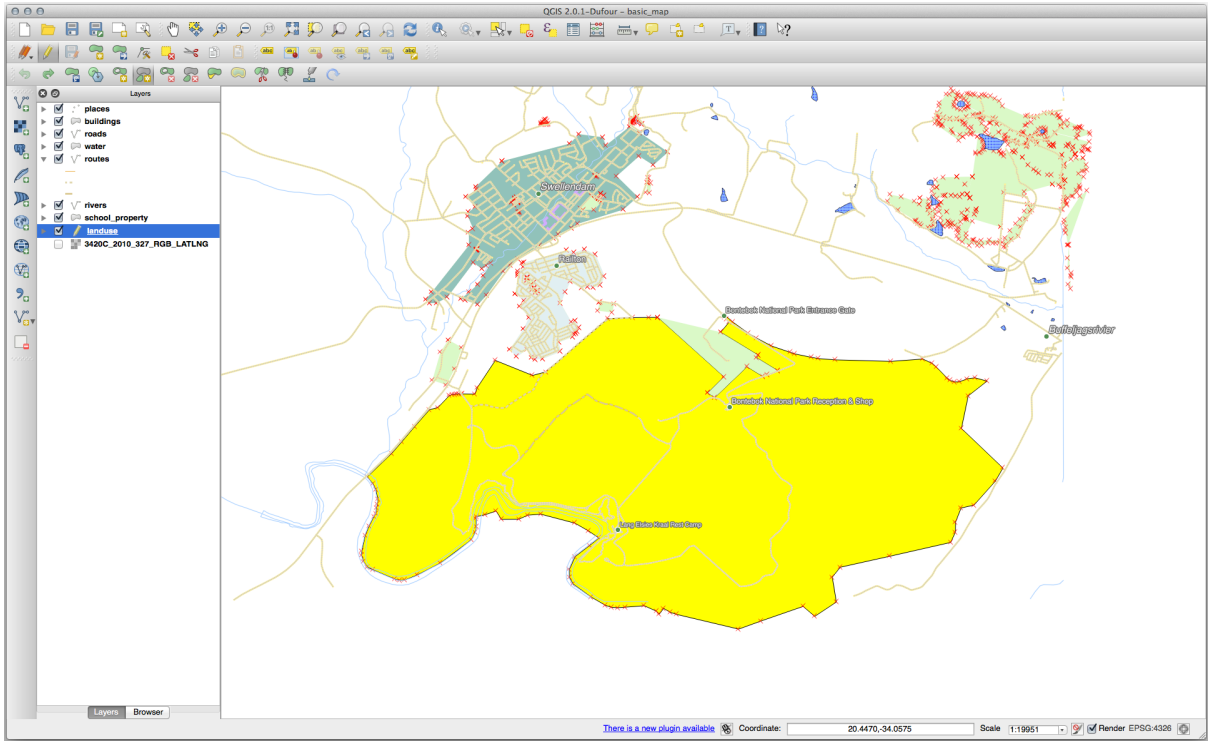
The Add Part tool allows you to add a new part to a feature, that is not directly connected to the main feature. For example, if you have digitized the boundaries of mainland South Africa, but you haven't yet added the Prince Edward Islands, you would use this tool to create them.

1. Select the polygon to which you wish to add the part by using the Select Features by area or single click tool.
2. Use the *Add Part* tool to add an outlying area.
3. Delete the part you just created using the Delete Part tool.

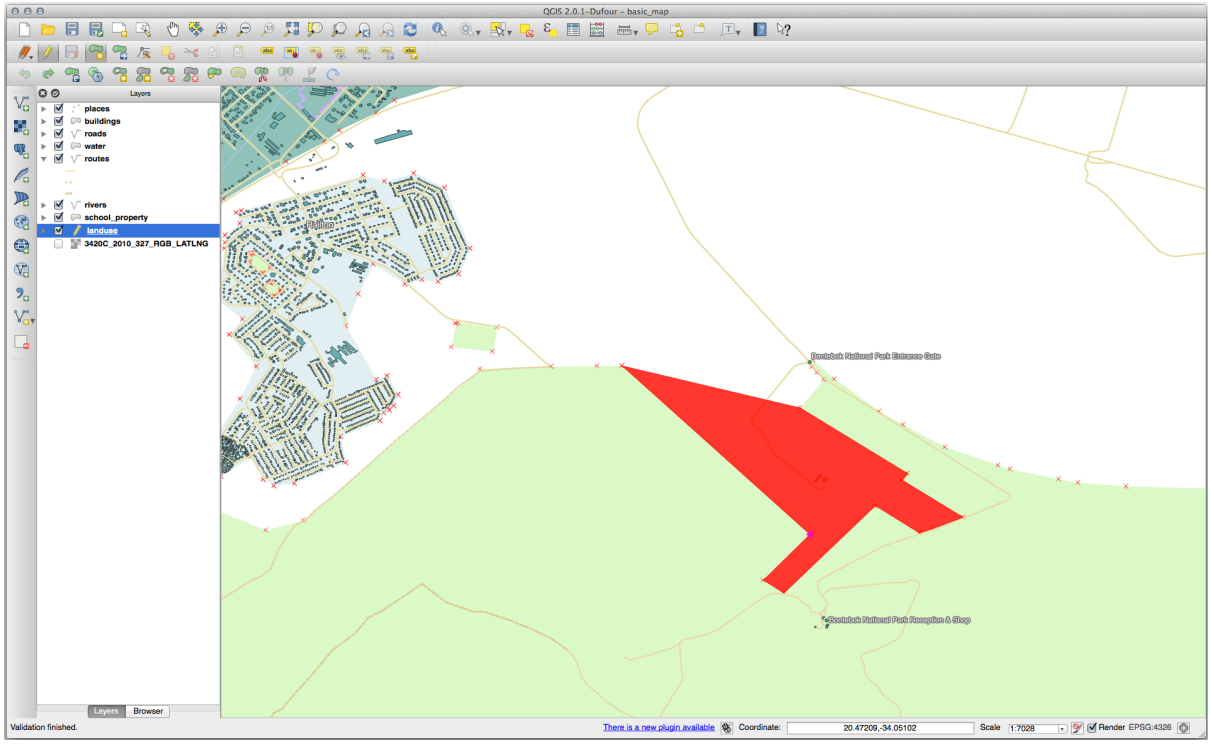
> **Nota:** Click inside the part to delete it.

Answer

1. First select the Bontebok National Park:



2. Now add your new part:



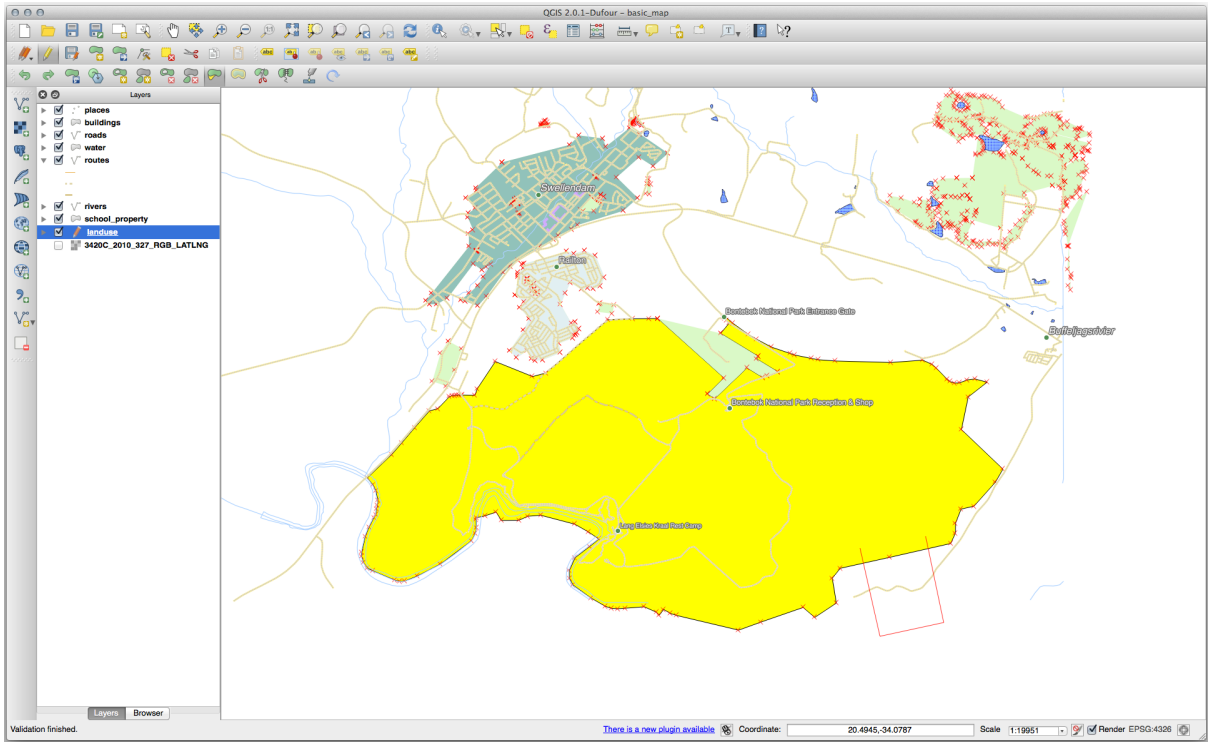
3. Undo your edit before continuing with the exercise for the next tool.

5.2.6 ??? Follow Along: Tool: Reshape Features

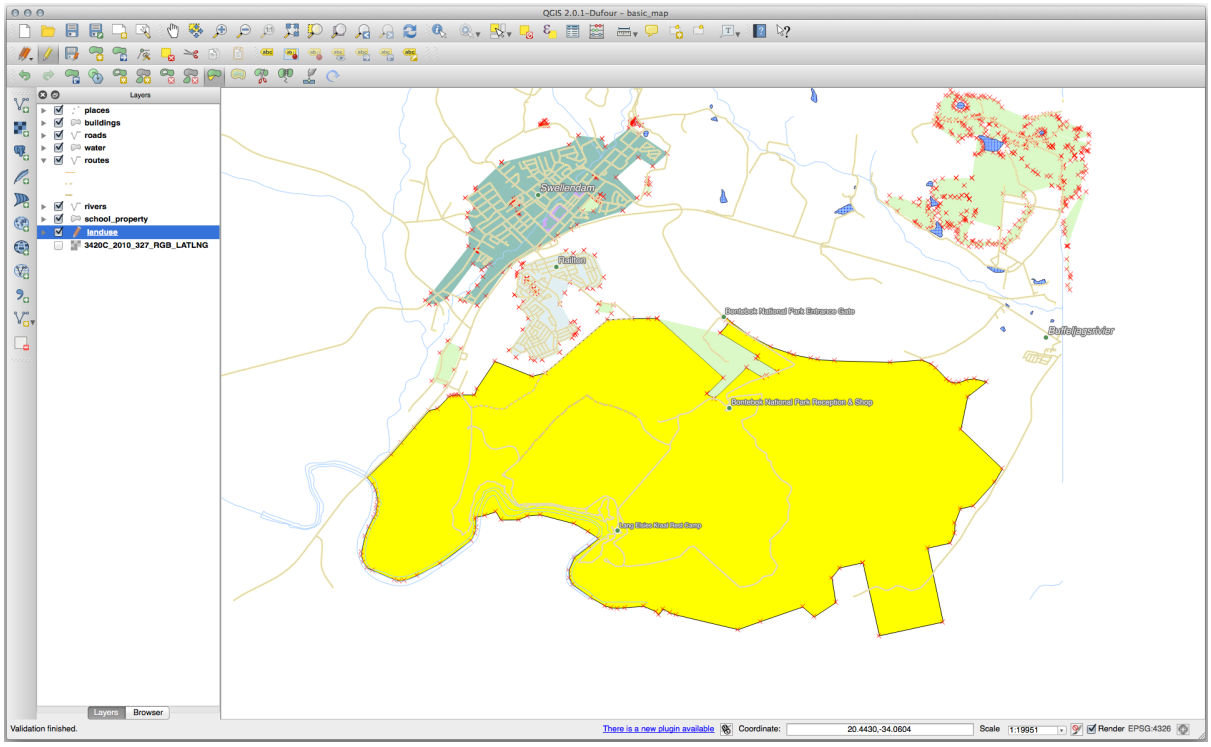
The Reshape Features tool is used to extend a polygon feature or cut away a part of it (along the boundary).

Extending:

1. Select the polygon using the Select Features by area or single click tool.
2. Left-click inside the polygon to start drawing.
3. Draw a shape outside the polygon. The last vertex should be back inside the polygon.
4. Right-click to finish the shape:



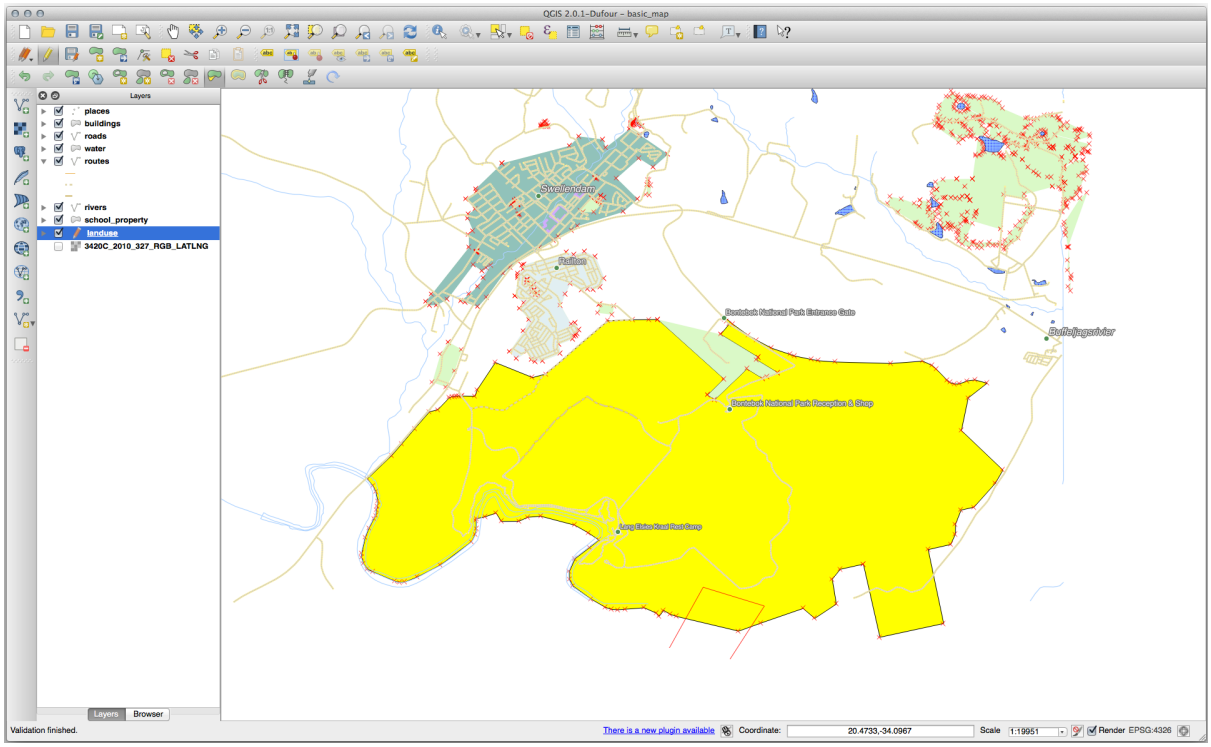
This will give a result similar to:



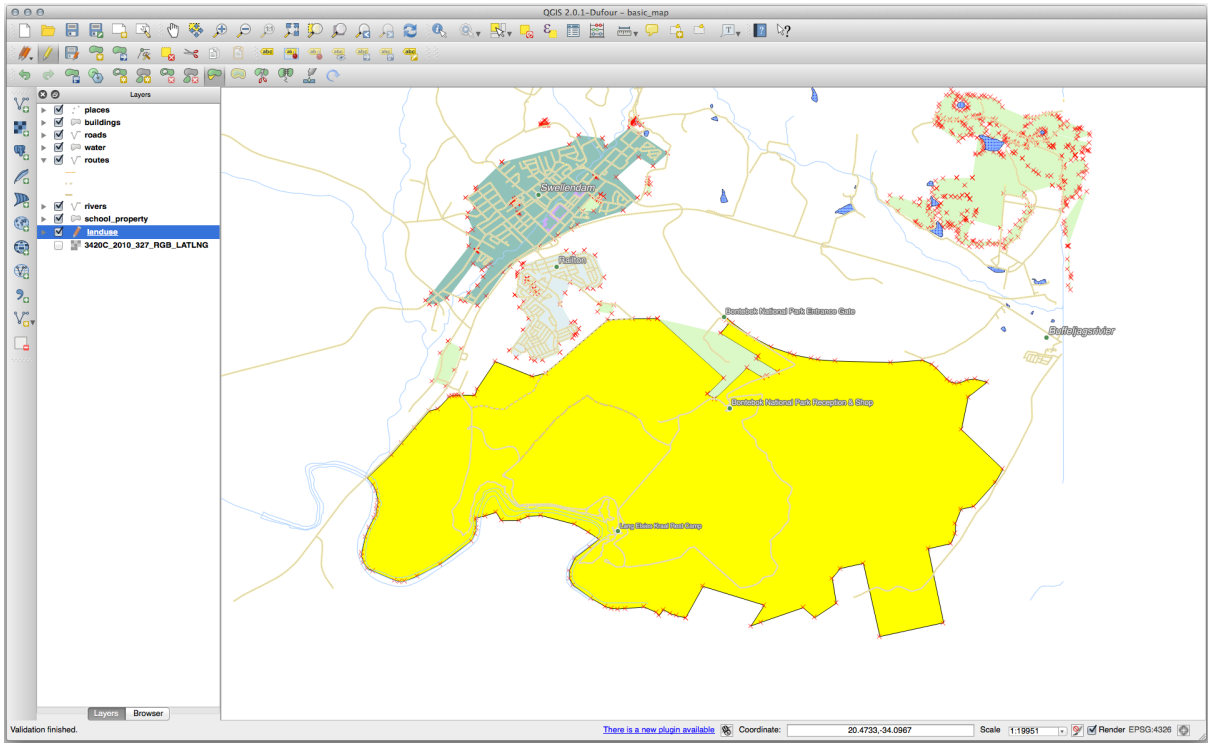
Cut away a part:

1. Select the polygon using the Select Features by area or single click tool.
2. Click outside the polygon.
3. Draw a shape inside the polygon. The last vertex must be back outside the polygon.

4. Right-click outside the polygon:



The result of the above:

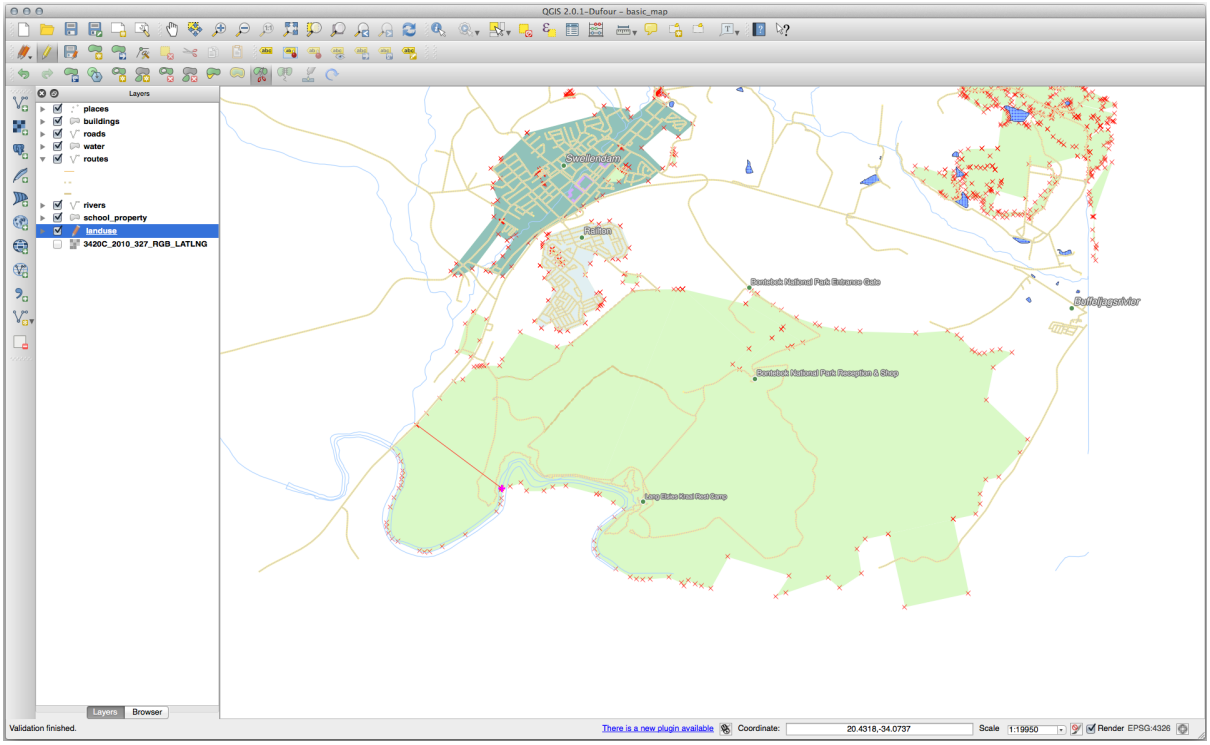


5.2.7 ??? Try Yourself: Tool: Split Features

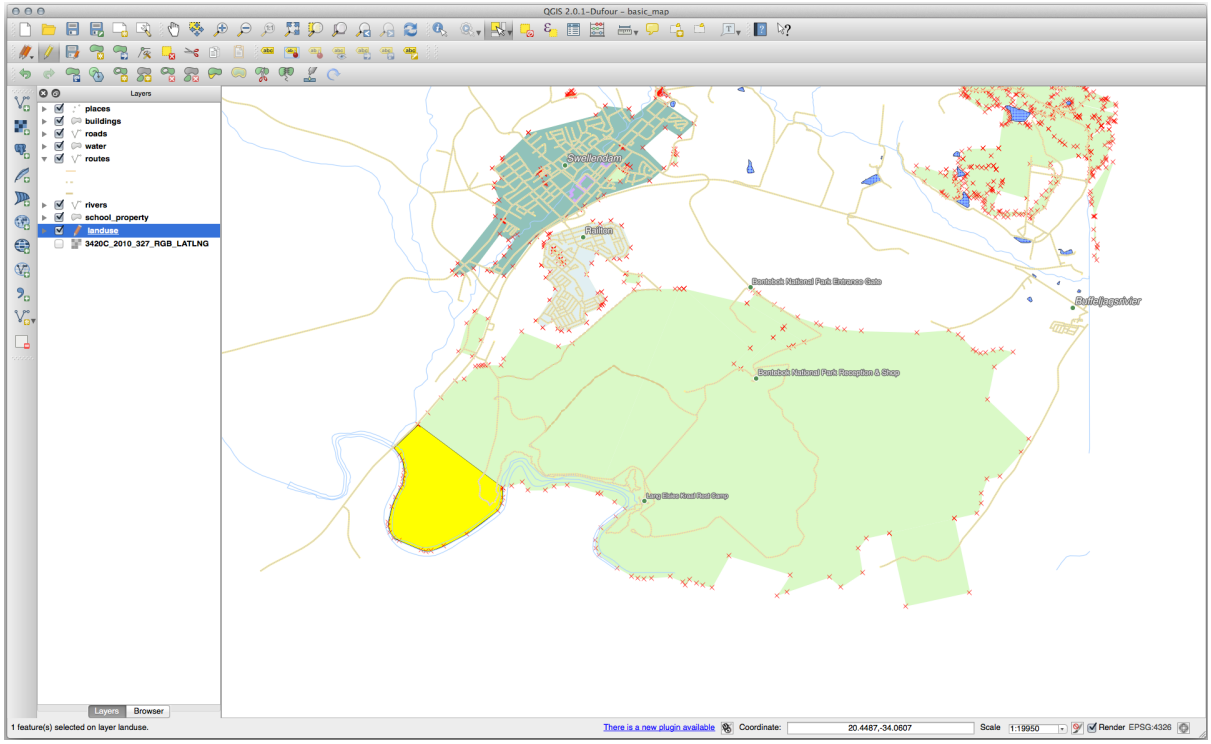
The Split Features tool is similar to the Reshape Features tool, except that it does not delete either of the two parts. Instead, it keeps them both.

We will use the tool to split a corner from a polygon.

1. First, select the `landuse` layer and re-enable snapping for it.
2. Select the Split Features tool and click on a vertex to begin drawing a line.
3. Draw the bounding line.
4. Click a vertex on the «opposite» side of the polygon you wish to split and right-click to complete the line:



5. At this point, it may seem as if nothing has happened. But remember that the `landuse` layer is rendered without border lines, so the new division line will not be shown.
6. Use the Select Features by area or single click tool to select the part you just split out; the new feature will now be highlighted:



5.2.8 ??? Try Yourself: Tool: Merge Features

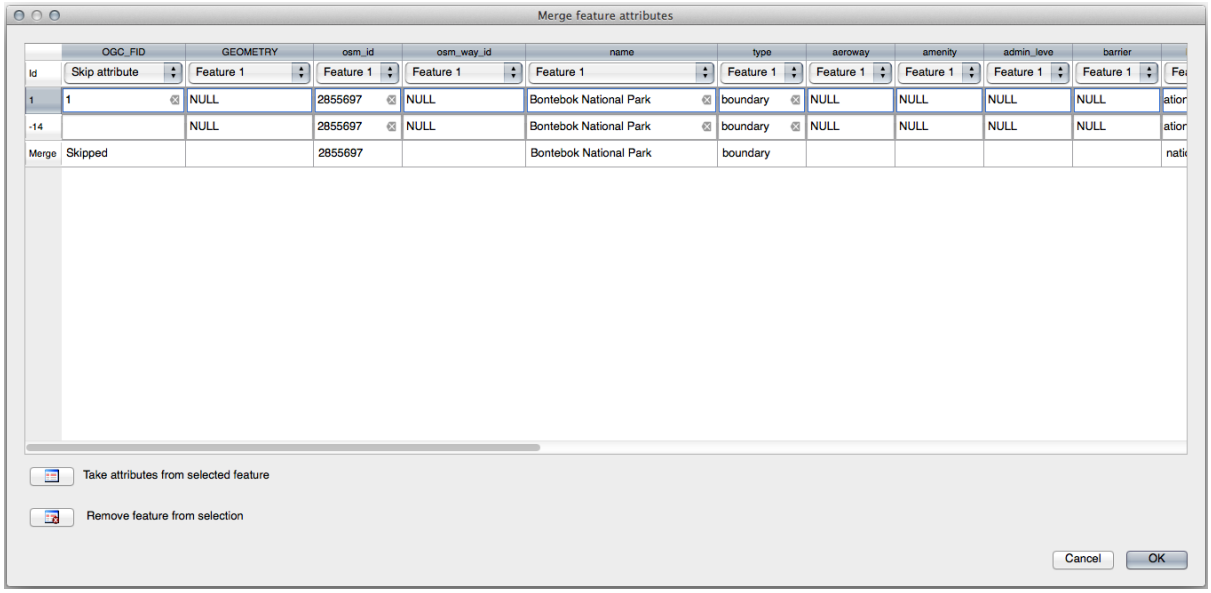
Now we will re-join the feature you just split out to the remaining part of the polygon:

1. Experiment with the Merge Selected Features and Merge Attributes of Selected Features tools.
2. Note the differences.

Answer

- Use the *Merge Selected Features* tool, making sure to first select both of the polygons you wish to merge.
- Use the feature with the *OGC_FID* of `1` as the source of your attributes (click on its entry in the dialog, then click the *Take attributes from selected feature* button):

If you're using a different dataset, it is highly likely that your original polygon's *OGC_FID* will not be `1`. Just choose the feature which has an *OGC_FID*.



Using the *Merge Attributes of Selected Features* tool will keep the geometries distinct, but give them the same attributes.

5.2.9 In Conclusion

Topology editing is a powerful tool that allows you to create and modify objects quickly and easily, while ensuring that they remain topologically correct.

5.2.10 What's Next?

Now you know how to digitize the shape of the objects easily, but adding attributes is still a bit of a headache! Next we will show you how to use forms, making attribute editing simpler and more effective.

5.3 Lesson: Forms

When you add new data via digitizing, you're presented with a dialog that lets you fill in the attributes for that feature. However, this dialog is not, by default, very nice to look at. This can cause a usability problem, especially if you have large datasets to create, or if you want other people to help you digitize and they find the default forms to be confusing.

Fortunately, QGIS lets you create your own custom dialogs for a layer. This lesson shows you how.

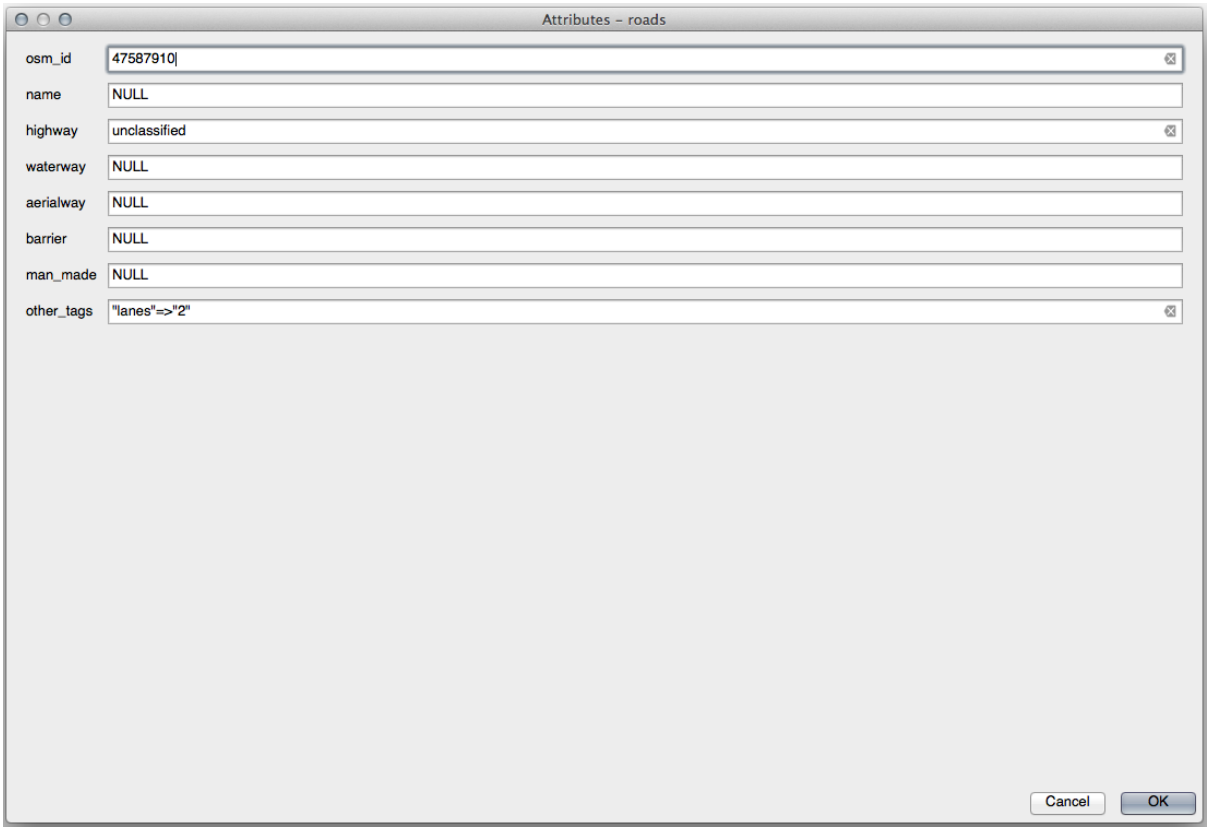
The goal for this lesson: To create a form for a layer.

5.3.1 ??? Follow Along: Using QGIS" Form Design Functionality

1. Select the `roads` layer in the *Layers* panel
2. Enter *Edit Mode* as before
3. Open the `roads` layer's attribute table
4. Right-click on any cell in the table. A short menu will appear, that includes the *Open form* entry.
5. Click on it to see the form that QGIS generates for this layer

Obviously it would be nice to be able to do this while looking at the map, rather than needing to search for a specific street in the *Attribute Table* all the time.

1. Select the *roads* layer in the *Layers* panel
2. Using the Identify Features tool, click on any street in the map.
3. The *Identify Results* panel opens and shows a tree view of the fields values and other general information about the clicked feature.
4. At the top of the panel, check the *Auto open form for single feature results* checkbox in the Identify Settings menu.
5. Now, click again on any street in the map. Along the previous *Identify Results* dialog, you'll see the now-familiar form:

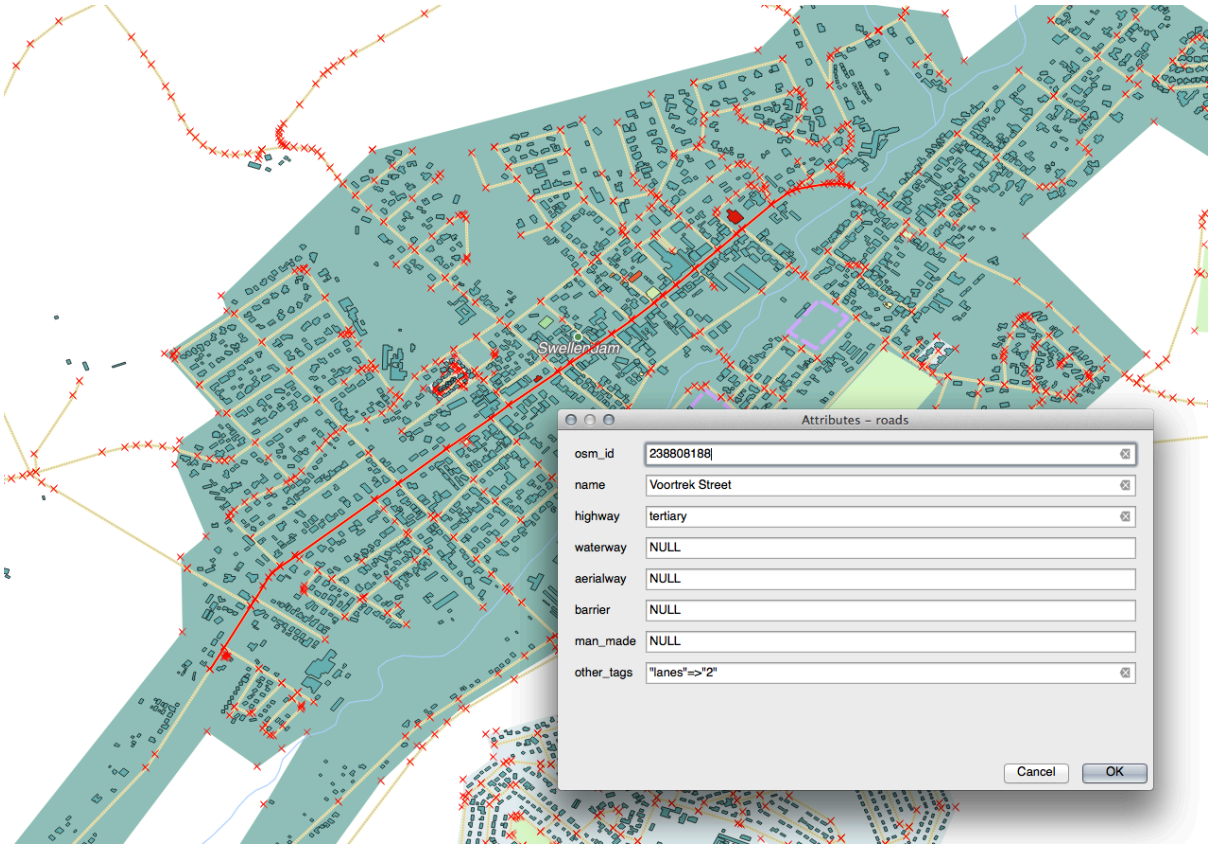


6. Each time you click on a single feature with the *Identify* tool, its form pops up as long as the *Auto open form* is checked.

5.3.2 ??? Try Yourself: Using the Form to Edit Values

If you are in edit mode, you can use this form to edit a feature's attributes.

1. Activate edit mode (if it isn't already activated).
2. Using the Identify Features tool, click on the main street running through Swellendam:

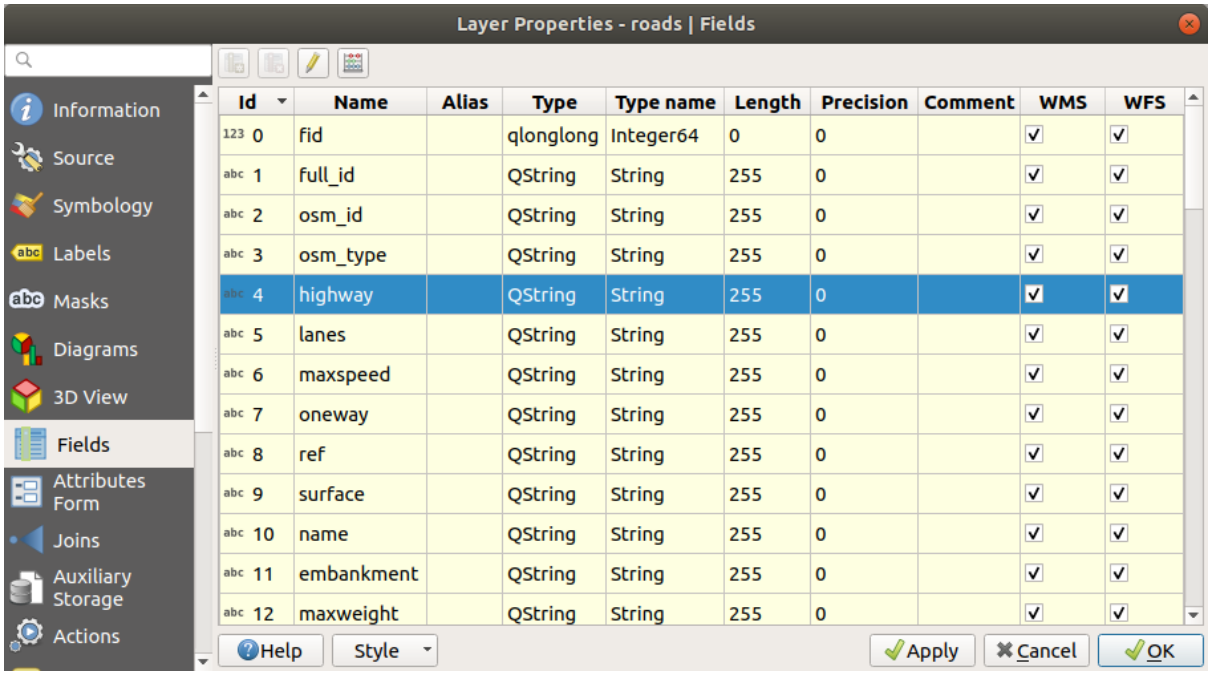


3. Edit its *highway* value to be `secondary`
4. Exit edit mode and save your edits
5. Open the *Attribute Table* and note that the value has been updated in the attributes table and therefore in the source data

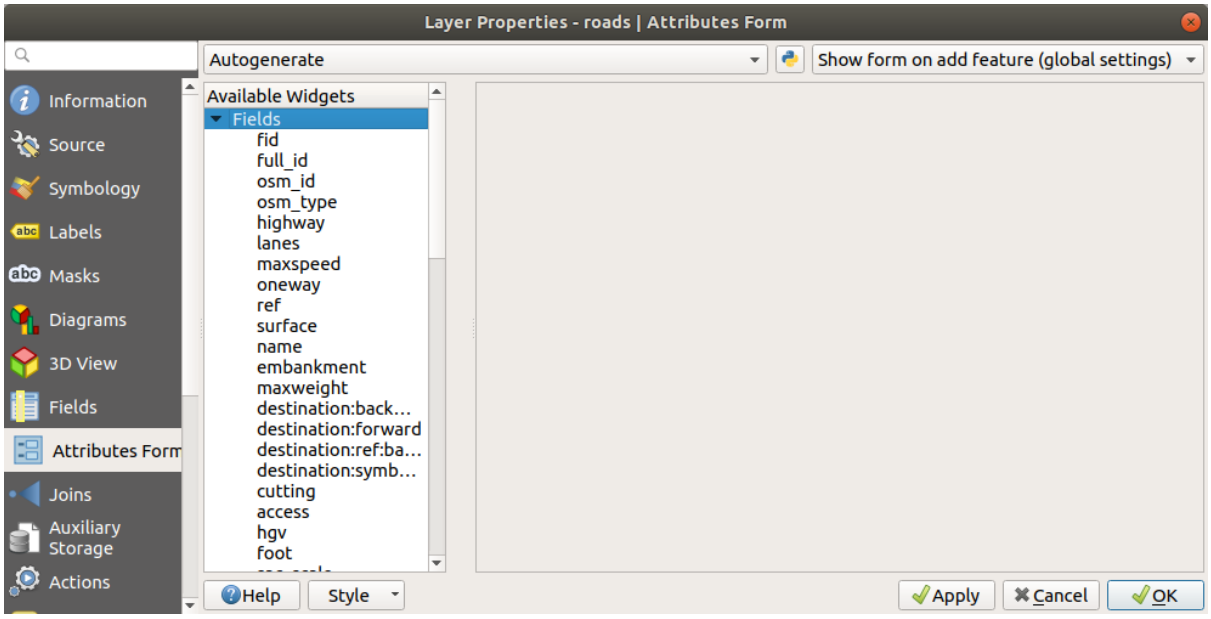
5.3.3 ??? Follow Along: Setting Form Field Types

It's nice to edit things using a form, but you still have to enter everything by hand. Fortunately, forms have different kinds of so-called *widgets* that allow you to edit data in various different ways.

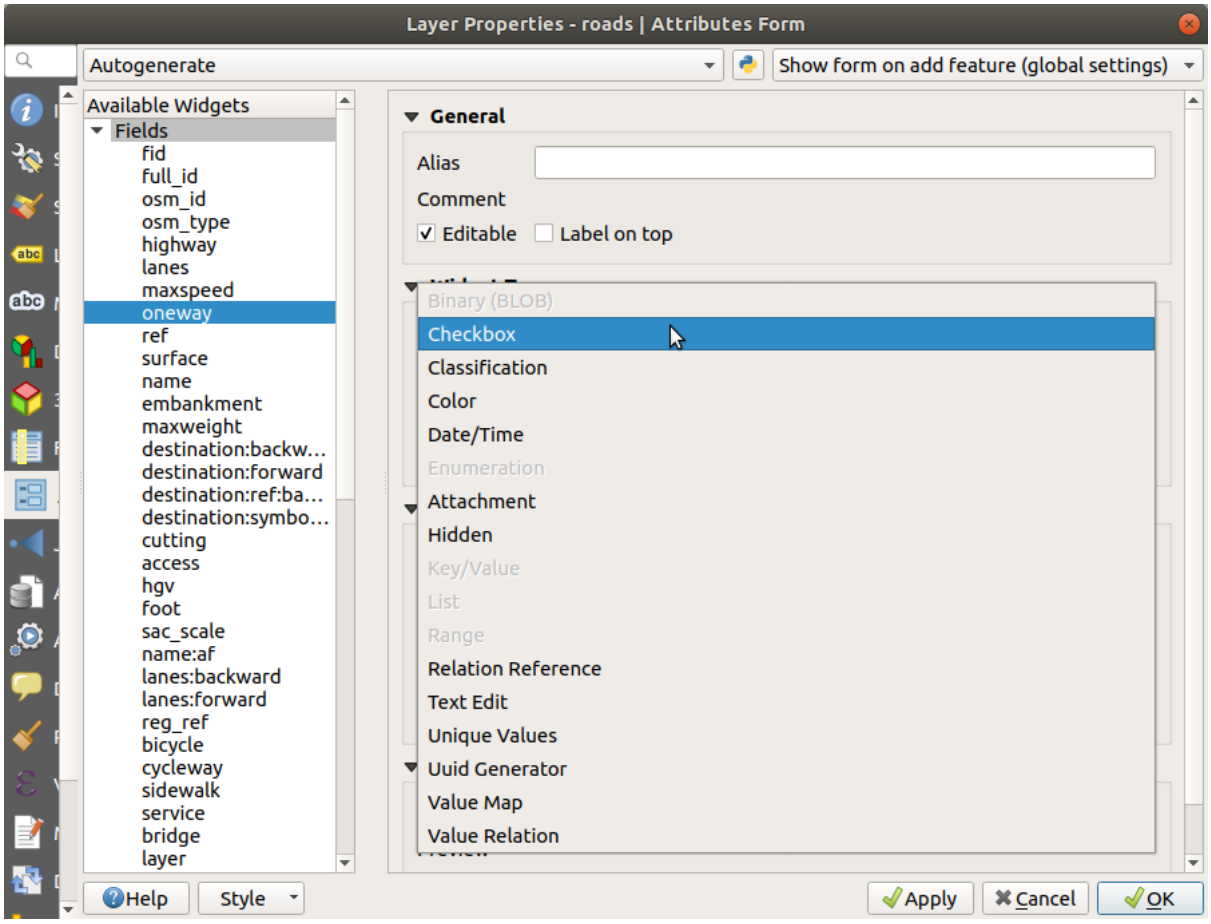
1. Open the *roads* layer's *Properties…*
2. Switch to the *Fields* tab. You'll see this:



3. Switch to the *Attributes Form* tab. You'll see this:



4. Click on the `oneway` row and choose *Checkbox* as *Widget Type* in the list of options:



5. Click *OK*
6. Enter edit mode (if the `roads` layer is not already in edit mode)
7. Click on the Identify Features tool
8. Click on the same main road you chose earlier

You will now see that the *oneway* attribute has a checkbox next to it denoting `True` (checked) or `False` (unchecked).

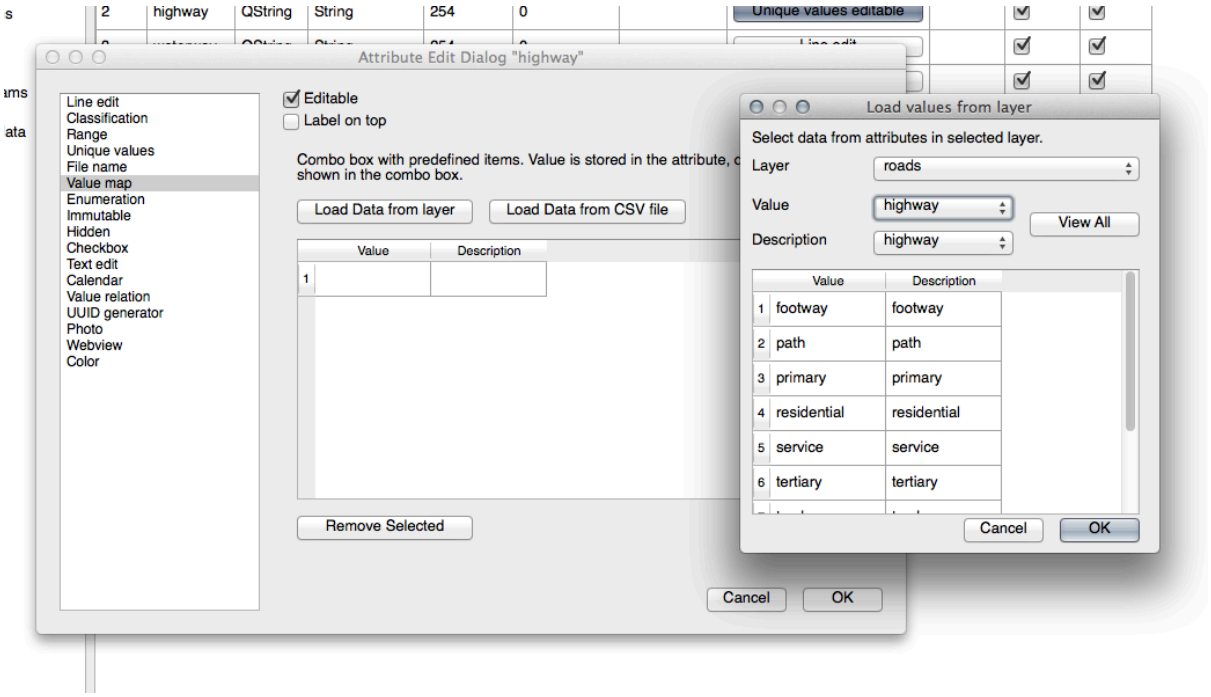
5.3.4 ??? Try Yourself:

Set a more appropriate form widget for the *highway* field.

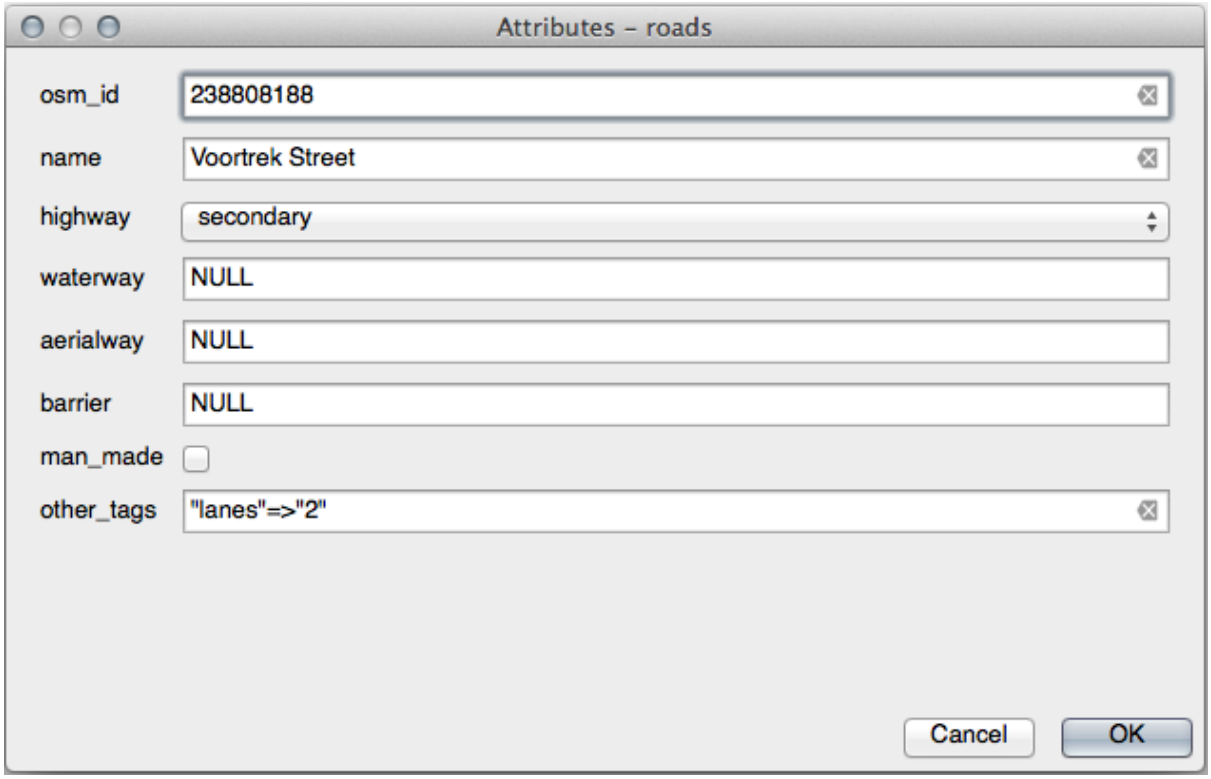
Answer

For the *TYPE*, there is obviously a limited amount of types that a road can be, and if you check the attribute table for this layer, you'll see that they are predefined.

1. Set the widget to *Value Map* and click *Load Data from Layer*.
2. Select *roads* in the *Label* dropdown and *highway* for both the *Value* and *Description* options:



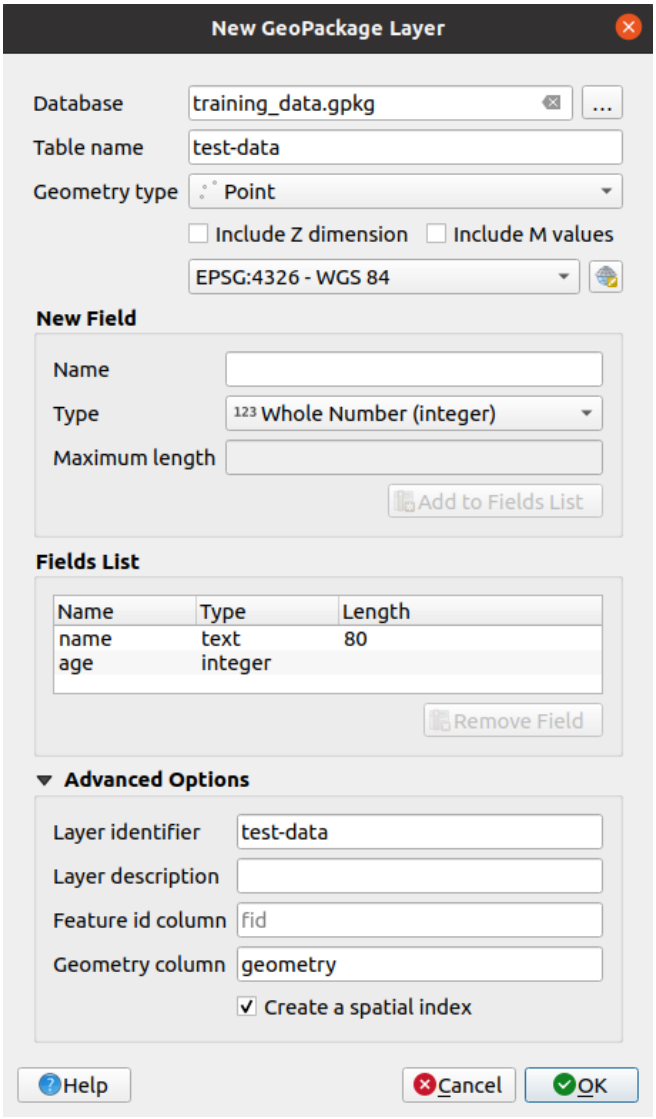
3. Click *OK* three times.
4. If you use the *Identify* tool on a street now while edit mode is active, the dialog you get should look like this:



5.3.5 ??? Try Yourself: Creating Test Data

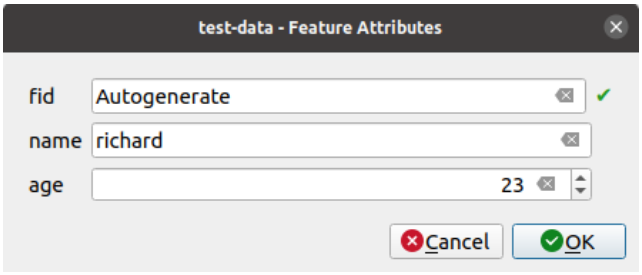
You can also design your own custom form completely from scratch.

1. Create a simple point layer named `test-data` with two attributes:
   - name (text)
   - age (integer)



2. Capture a few points on your new layer using the digitizing tools so that you have a little data to play with. You should be presented with the default QGIS generated attribute capture form each time you capture a new point.

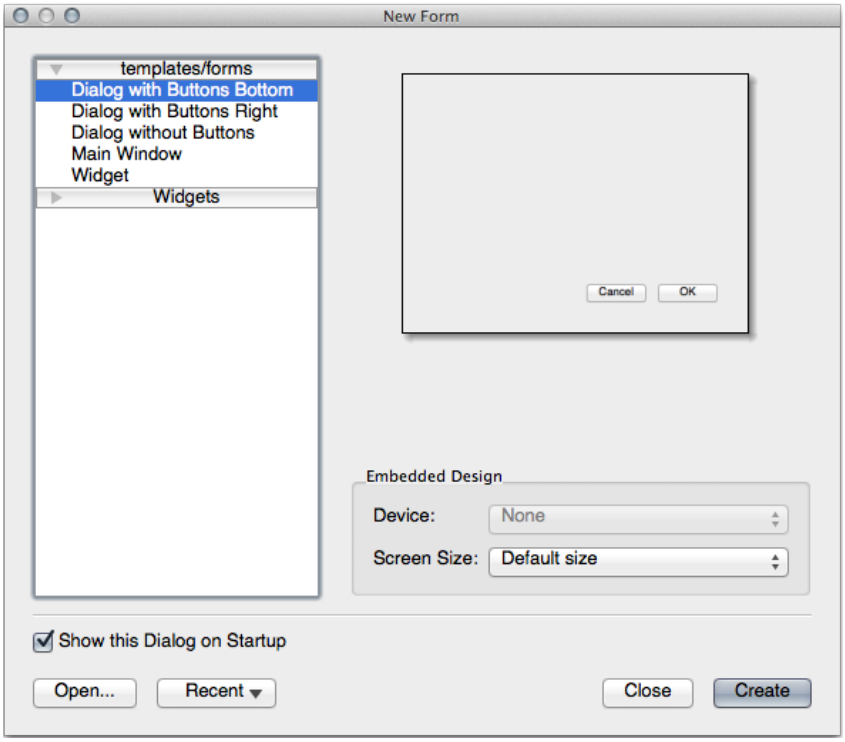
Nota: You may need to disable Snapping if still enabled from earlier tasks.



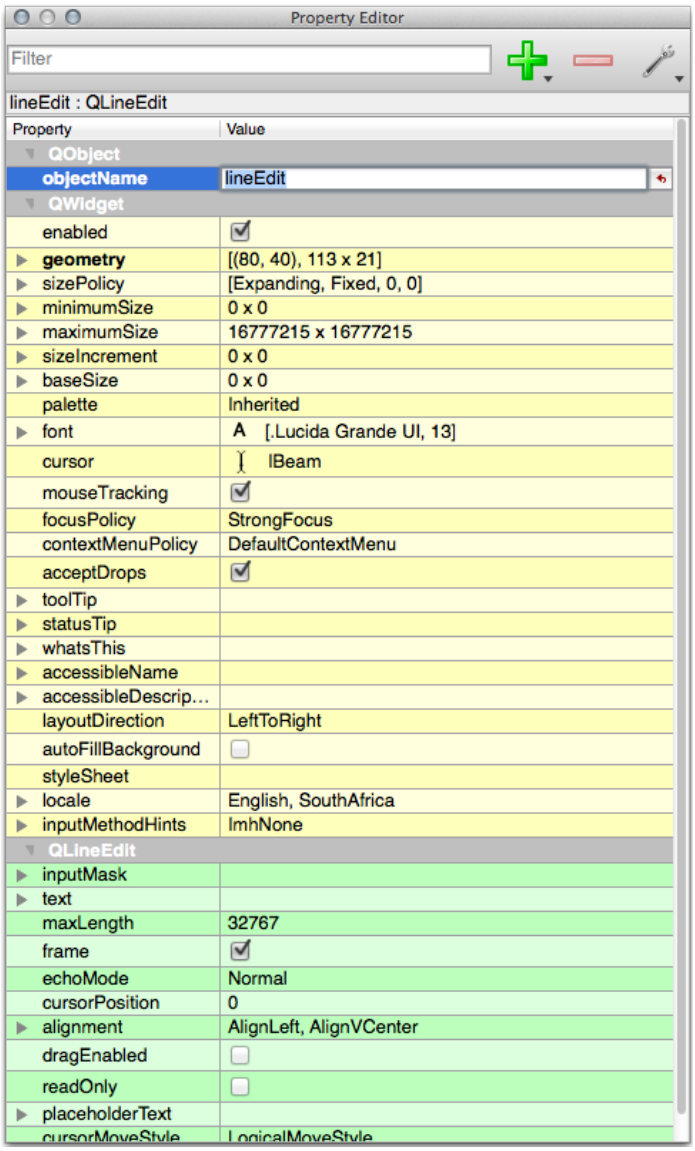
5.3.6 ??? Follow Along: Creating a New Form

Now we want to create our own custom form for the attribute data capture phase. To do this, you need to have *QT Designer* installed (only needed for the person who creates the forms).

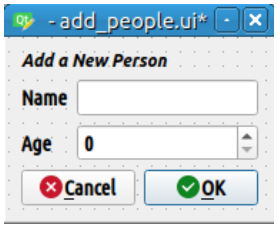
1. Start *QT Designer*.
2. In the dialog that appears, create a new dialog:



3. Look for the *Widget Box* along the left of your screen (default). It contains an item called *Line Edit*.
4. Click and drag this item into your form. This creates a new *Line Edit* in the form.
5. With the new line edit element selected, you'll see its *properties* along the side of your screen (on the right by default):



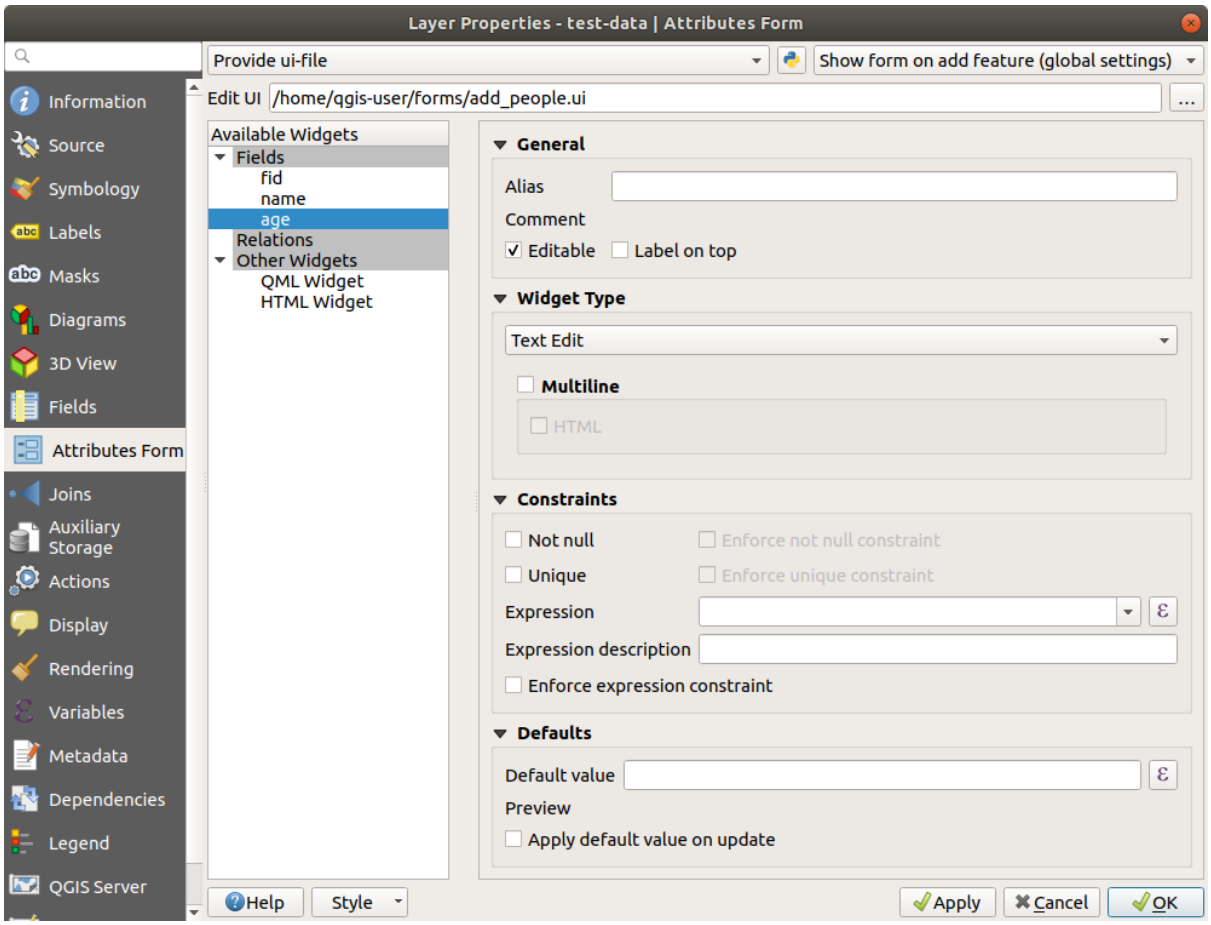
6. Set its name to `name`.
7. Using the same approach, create a new *Spin Box* and set its name to `age`.
8. Add a *Label* with the text `Add a New Person` in a bold font (look in the object *properties* to find out how to set this). Alternatively, you may want to set the title of the dialog itself (rather than adding a label).
9. Add a *Label* for your *Line Edit* and your *Spin Box*.
10. Arrange the elements to your own desire.
11. Click anywhere in your dialog.
12. Find the *Lay Out in a Form Layout* button (in a toolbar along the top edge of the screen, by default). This lays out your dialog automatically.
13. Set the dialog's maximum size (in its properties) to `200` (width) by `150` (height).
14. Your form should now look similar to this:



15. Save your new form as `exercise_data/forms/add_people.ui`
16. When it's done saving, you can close *Qt Designer*

5.3.7 ??? Follow Along: Associating the Form with Your Layer

1. Go back to QGIS
2. Double click the *test-data* layer in the legend to access its properties.
3. Click on the *Attributes Form* tab in the *Layer Properties* dialog
4. In the *Attribute editor layout* dropdown, select *Provide ui-file*.
5. Click the ellipsis button and choose the `add_people.ui` file you just created:



6. Click *OK* on the *Layer Properties* dialog
7. Enter edit mode and capture a new point
8. When you do so, you will be presented with your custom dialog (instead of the generic one that QGIS usually creates).

9. If you click on one of your points using the Identify Features tool, you can now bring up the form by right clicking in the identify results window and choosing *View Feature Form* from the context menu.
10. If you are in edit mode for this layer, that context menu will show *Edit Feature Form* instead, and you can then adjust the attributes in the new form even after initial capture.

5.3.8 In Conclusion

Using forms, you can make life easier for yourself when editing or creating data. By editing widget types or creating an entirely new form from scratch, you can control the experience of someone who digitizes new data for that layer, thereby minimizing misunderstandings and unnecessary errors.

5.3.9 Further Reading

If you completed the advanced section above and have knowledge of Python, you may want to check out this blog entry about creating custom feature forms with Python logic, which allows advanced functions including data validation, autocompletion, etc.

5.3.10 What's Next?

Opening a form on identifying a feature is one of the standard actions that QGIS can perform. However, you can also direct it to perform custom actions that you define. This is the subject of the next lesson.

5.4 Lesson: Actions

Now that you have seen a default action in the previous lesson, it is time to define your own actions.

An action is something that happens when you click on a feature. It can add a lot of extra functionality to your map, allowing you to retrieve additional information about an object, for example. Assigning actions can add a whole new dimension to your map!

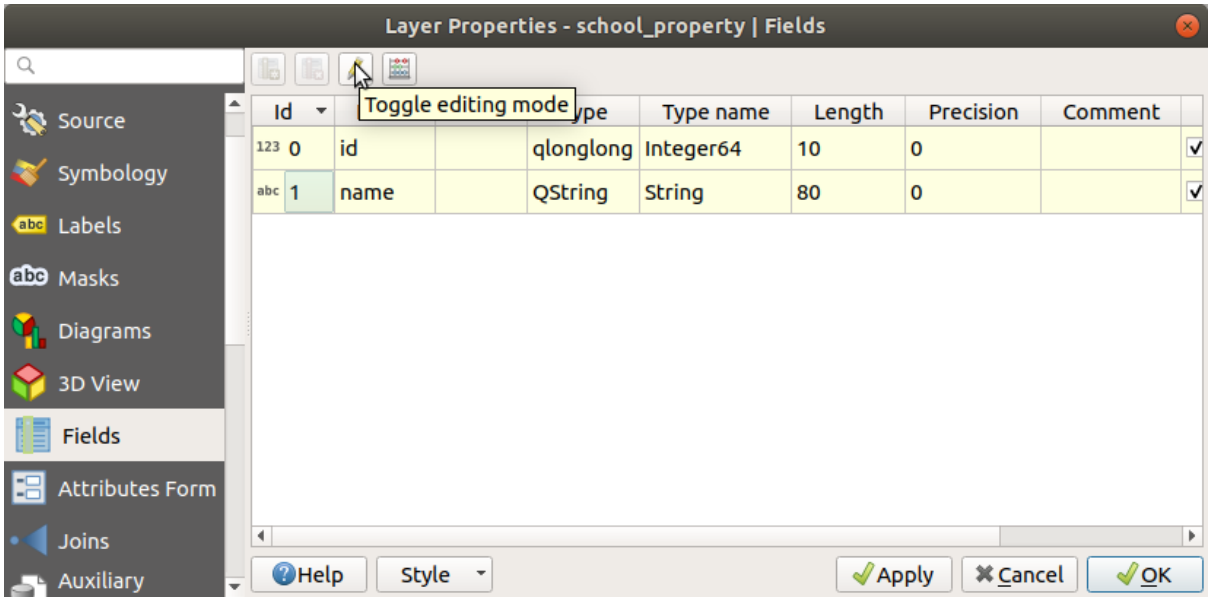
The goal for this lesson: To learn how to add custom actions.

In this lesson you will use the *school_property* layer you created previously. The sample data include photos of each of the three properties you digitized. What we are going to do is to associate each property with its image. Then we will create an action that will open the image for a property when clicking on the property.

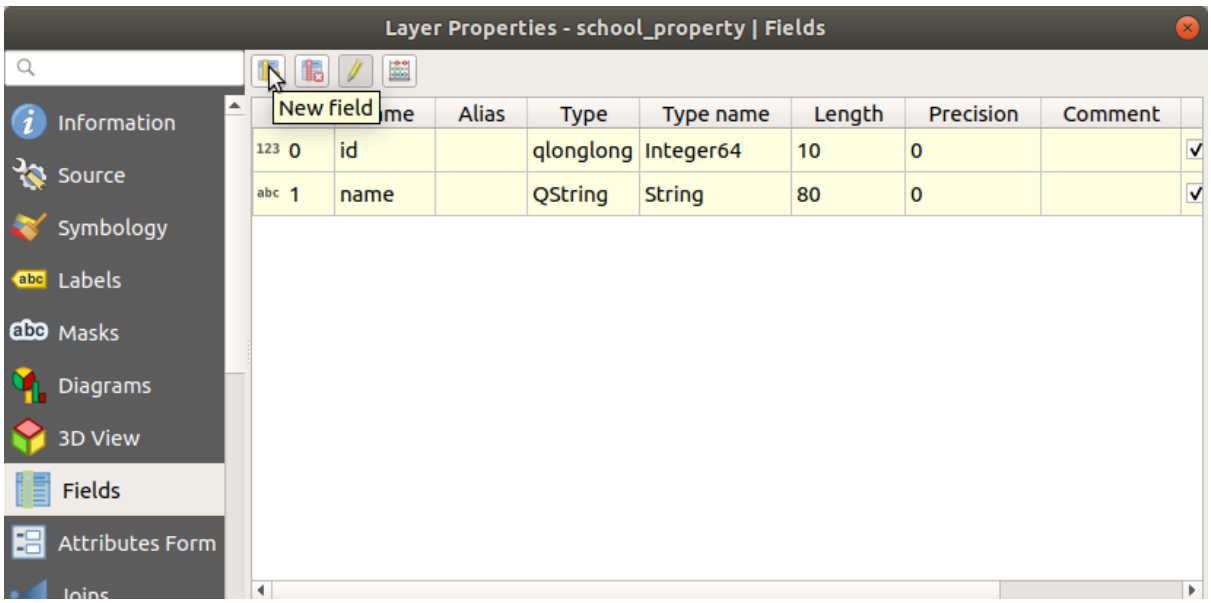
5.4.1 ??? Follow Along: Add a Field for Images

The `school_property` layer has no way to associate an image with a property yet. First we will create a field for this purpose.

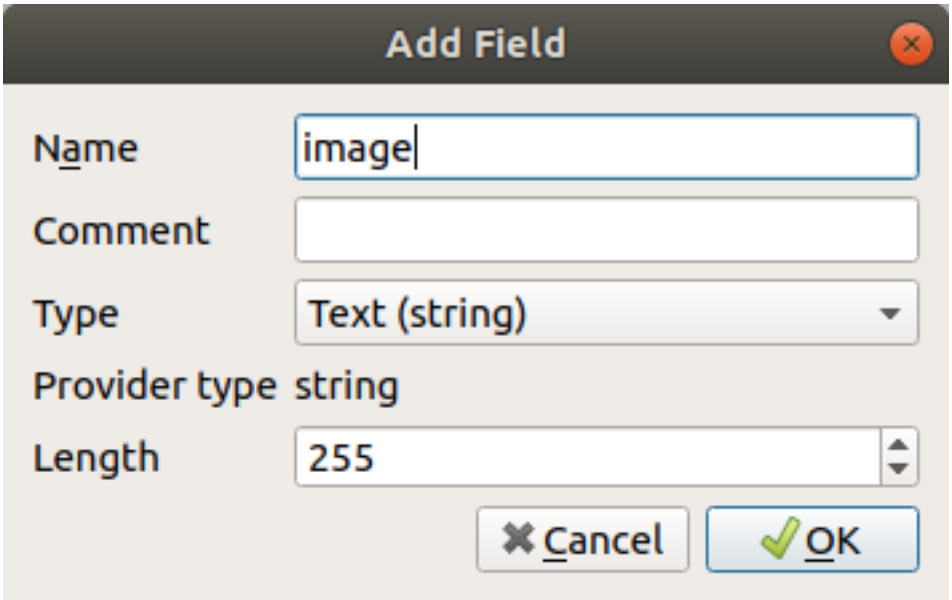
1. Open the *Layer Properties* dialog.
2. Click on the *Fields* tab.
3. Toggle editing mode:



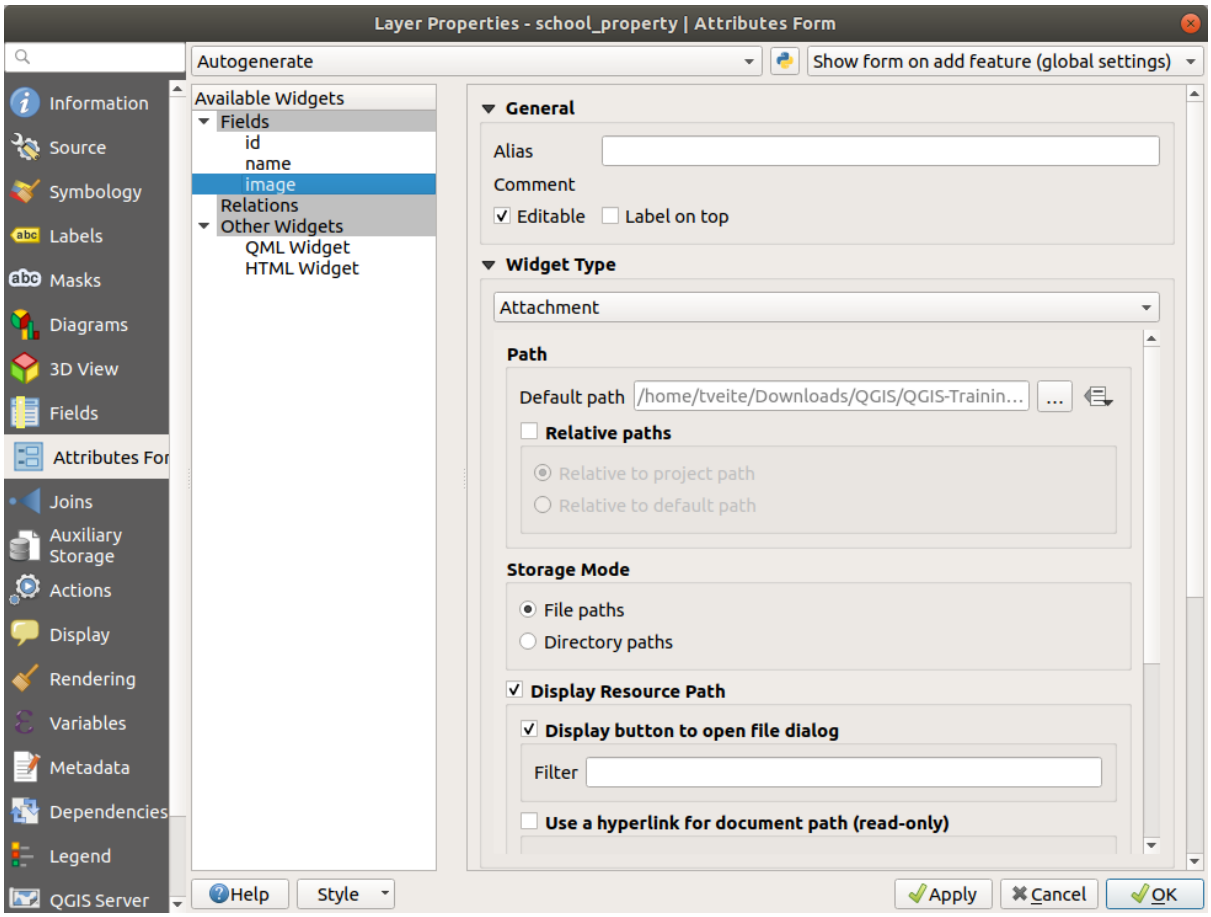
4. Add a new column:



5. Enter the values below:

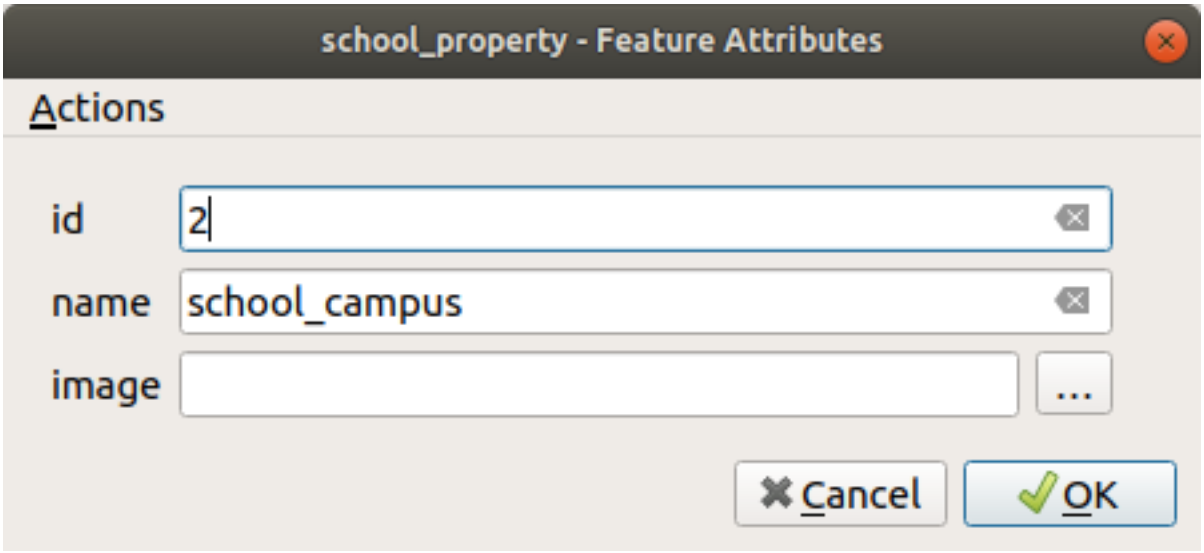


6. After the field has been created, move to the *Attributes Form* tab and select the `image` field.
7. Set *Widget Type* to *Attachment*:



8. Click *OK* in the *Layer Properties* dialog.
9. Use the *Identify* tool to click on one of the three features in the *school_property* layer.

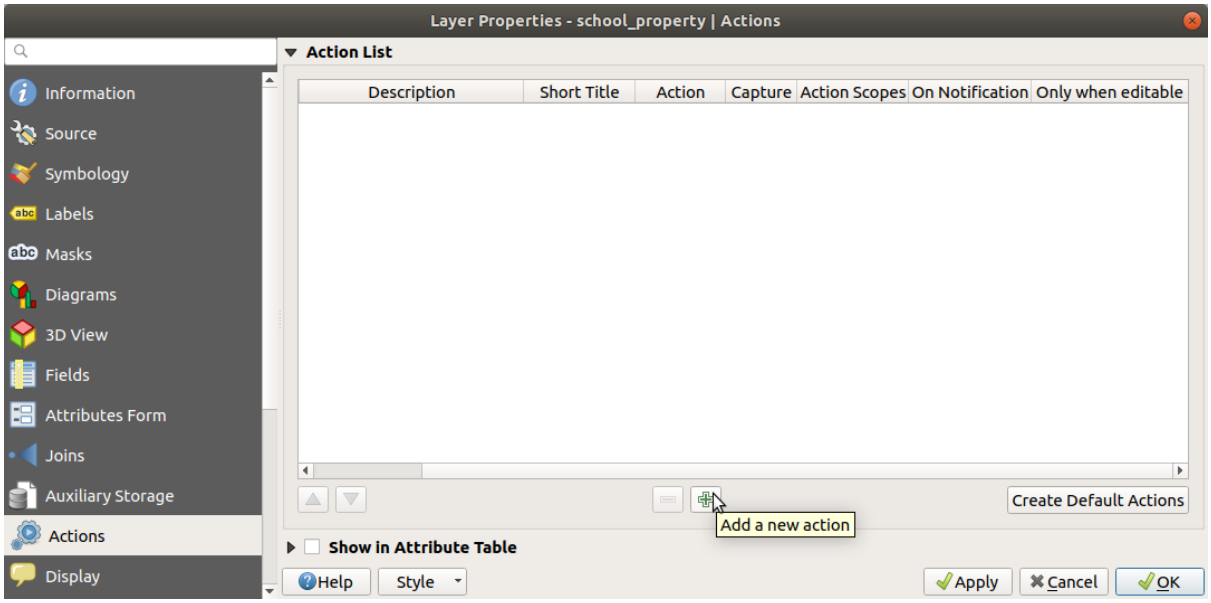
   Since you are still in edit mode, the dialog should be active and look like this:



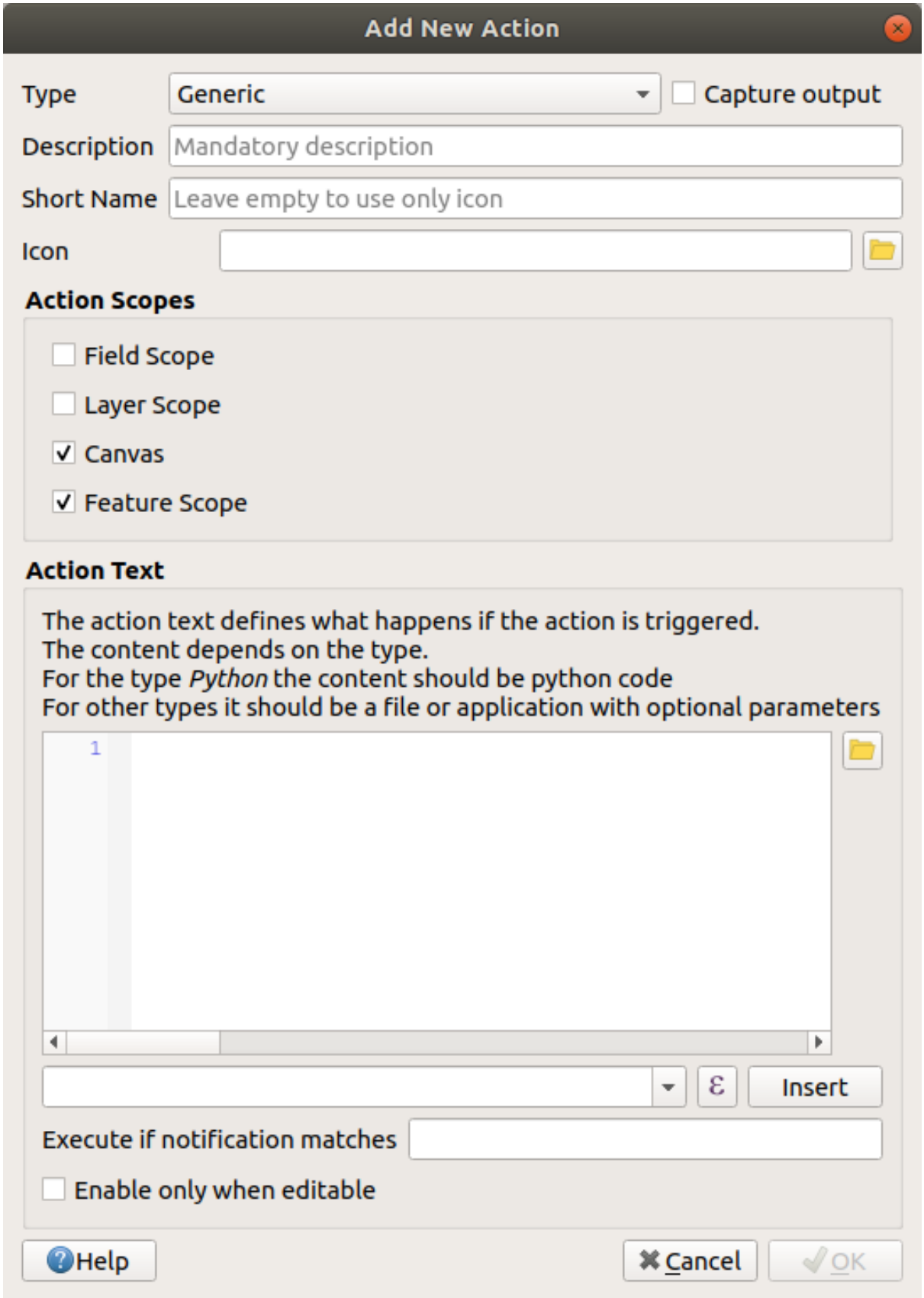
10. Click on the browse button (the … next to the *image* field).
11. Select the path for your image. The images are in `exercise_data/school_property_photos/` and are named the same as the features they should be associated with.
12. Click *OK*.
13. Associate all of the images with the correct features using this method.
14. Save your edits and exit edit mode.

5.4.2 ??? Follow Along: Creating an Action

1. Open the *Actions* tab for the *school_property* layer, and click on the Add a new action button.



2. In the *Add New Action* dialog, enter the words `Show Image` into the *Description* field:



What to do next varies according to your operating system, so choose the appropriate course to follow:

- Windows

Click on the *Type* dropdown and choose *Open*.

- Ubuntu Linux

  Under *Action*, write `eog` for the *Gnome Image Viewer*, or write `display` to use *ImageMagick*. Remember to put a space after the command!

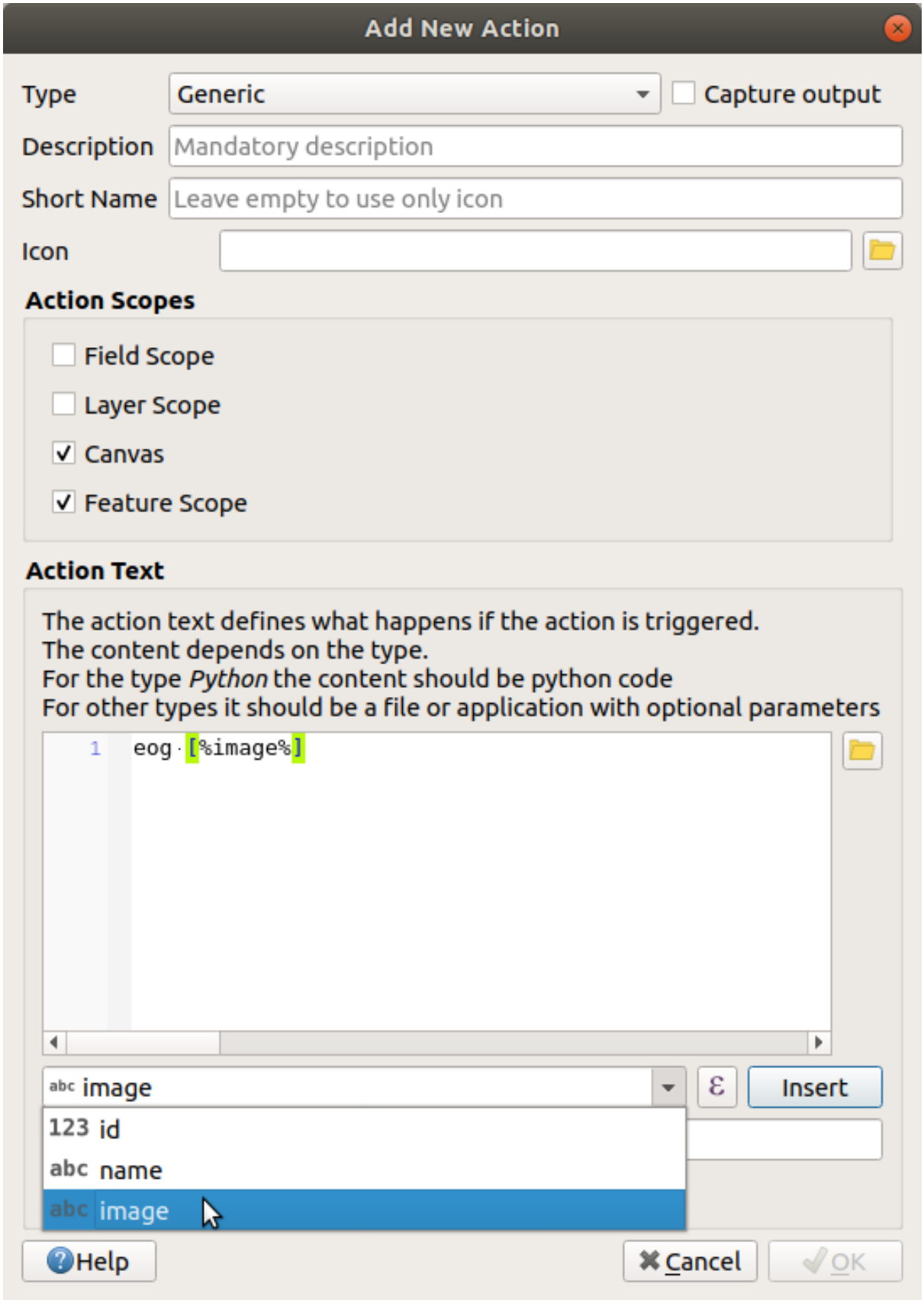
- macOS

  1. Click on the *Type* dropdown and choose *Mac*.
  2. Under *Action*, write `open`. Remember to put a space after the command!

Now you can continue writing the command.

You want to open the image, and QGIS knows where the image is. All it needs to do is to tell the *Action* where the image is.

3. Select *image* from the list:

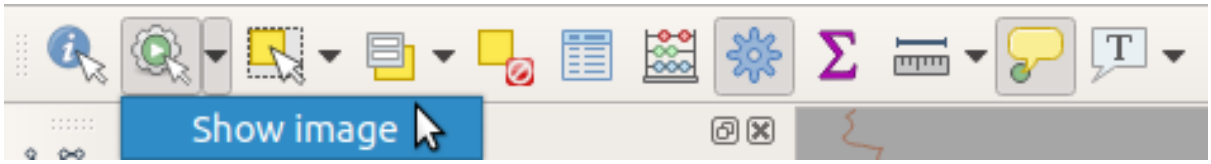


4. Click the *Insert field* button. QGIS will add the phrase `[% "image" %]` in the *Action Text* field.
5. Click the *OK* button to close the *Add New Action* dialog

6. Click *OK* to close the *Layer Properties* dialog

Now it is time to test the new action:

1. Click on the *school_property* layer in the *Layers* panel so that it is highlighted.
2. Find the Run feature action button (in the *Attributes Toolbar*).
3. Click on the down arrow to the right of this button. There is only one action defined for this layer so far, which is the one you just created.



4. Click the button itself to activate the tool.
5. Using this tool, click on any of the three school properties.

   The image for that property should open.

5.4.3 ??? Follow Along: Searching the Internet

Let's say we are looking at the map and want to know more about the area that a farm is in. Suppose you know nothing of the area in question and want to find general information about it. Your first impulse, considering that you're using a computer right now, would probably be to Google the name of the area. So let's tell QGIS to do that automatically for us!

1. Open the attribute table for the *landuse* layer.

   We will be using the `name` field for each of our landuse areas to search Google.
2. Close the attribute table.
3. Go back to *Actions* in *Layer Properties*.
4. Click on the *Create Default Actions* button to add a number of pre-defined actions.
5. Remove all the actions but the *Open URL* action with the short name *Search Web* using the Remove the selected action button below.
6. Double-click on the remaining action to edit it
7. Change the *Description* to `Google Search`, and remove the content of the *Short Name* field.
8. Make sure that *Canvas* is among the checked *Action scopes*.

   What to do next varies according to your operating system, so choose the appropriate course to follow:

   - Windows

     Under *Type*, choose *Open*. This will tell Windows to open an Internet address in your default browser, such as Internet Explorer.
   - Ubuntu Linux

     Under *Action*, write `xdg-open`. This will tell Ubuntu to open an Internet address in your default browser, such as Chrome or Firefox.
   - macOS

     Under *Action*, write `open`. This will tell macOS to open an Internet address in your default browser, such as Safari.

Now you can continue writing the command

Whichever command you used above, you need to tell it which Internet address to open next. You want it to visit Google, and to search for a phrase automatically.

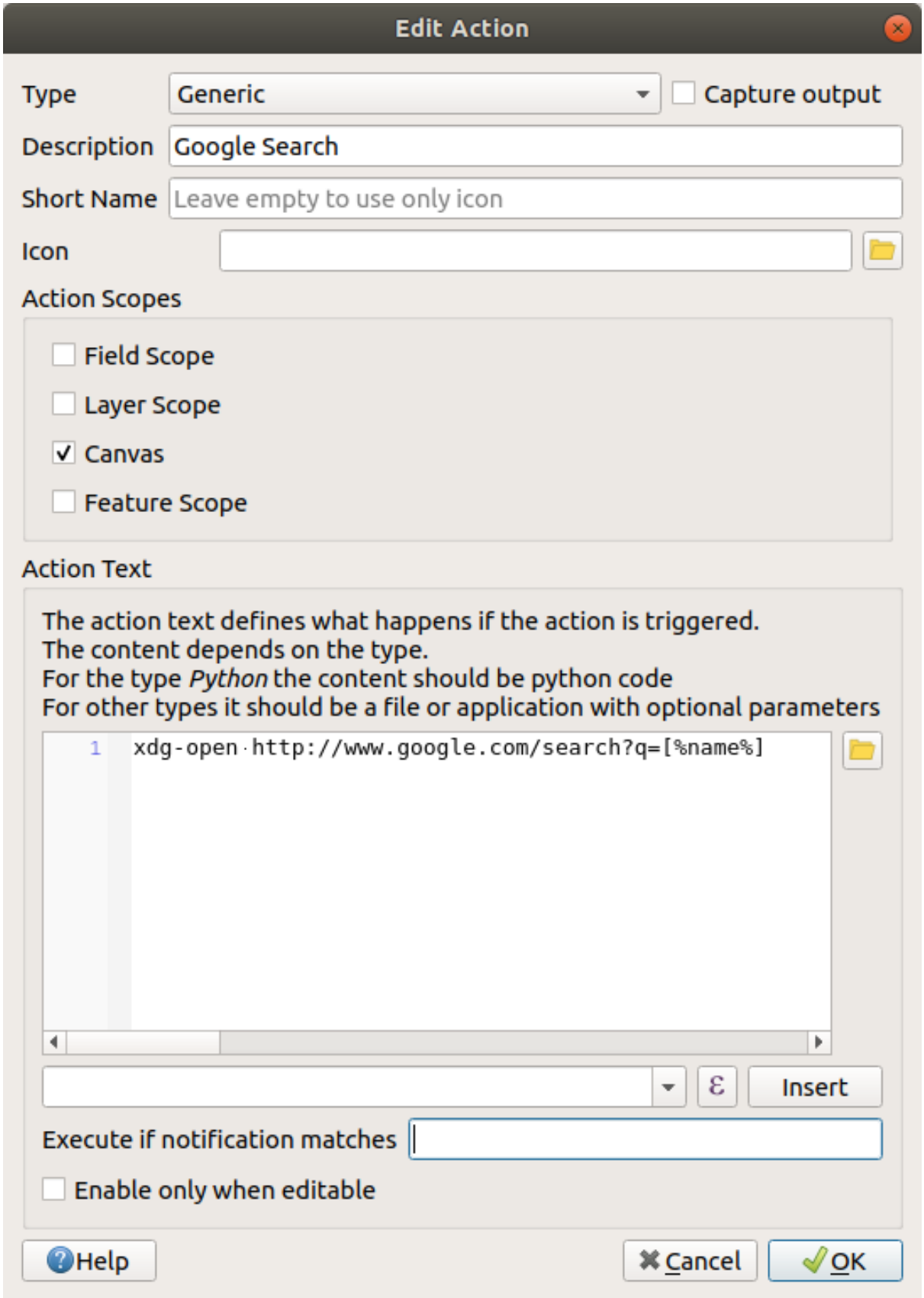
Usually when you use Google, you enter your search phrase into the Google Search bar. But in this case, you want your computer to do this for you. The way you tell Google to search for something (if you don't want to use its search bar directly) is by giving your Internet browser the address `https://www.google.com/search?q=SEARCH_PHRASE`, where `SEARCH_PHRASE`` is what you want to search for. Since we don't know what phrase to search for yet, we will just enter the first part (without the search phrase).

9. In the *Action* field, write `https://www.google.com/search?q=`. Remember to add a space after your initial command before writing this in!

   Now you want QGIS to tell the browser to tell Google to search for the value of `name` for any feature that you could click on.

10. Select the *name* field.

11. Click *Insert* button:



What this means is that QGIS is going to open the browser and send it to the address `https://www.google.com/search?q=[% "name" %]`. `[% "name" %]` tells QGIS to use the contents of the `name` field as the phrase to search for.

So if, for example, the landuse area you click on is named `Marloth Nature Reserve`, QGIS is going to send the browser to `https://www.google.com/search?q=Marloth%20Nature%20Reserve`, which will cause your browser to visit Google, which will in turn search for «Marloth Nature Reserve».

12. If you have not done so already, set everything up as explained above.
13. Click the *OK* button to close the *Add New Action* dialog
14. Click *OK* to close the *Layer Properties* dialog

Now to test the new action.

1. With the *landuse* layer active in the *Layers* panel, click on the down arrow to the right of the Run feature action button, and select the only action (`Google Search`) defined for this layer.
2. Click on any landuse area you can see on the map. Your browser will now open, and will start a Google search for the place that is recorded as that area's `name` value.

Nota: If your action doesn't work, check that everything was entered correctly; typos are common with this kind of work!

5.4.4 ??? Follow Along: Open a Webpage Directly in QGIS

Above, you've seen how to open a webpage in an external browser. There are some shortcomings with this approach in that it adds an unknowable dependency – will the end-user have the software required to execute the action on their system? As you've seen, they don't necessarily even have the same kind of base command for the same kind of action, if you don't know which OS they will be using. With some OS versions, the above commands to open the browser might not work at all. This could be an insurmountable problem.

However, QGIS sits on top of the incredibly powerful and versatile Qt library. Also, QGIS actions can be arbitrary, tokenized (i.e. using variable information based on the contents of a field attribute) Python commands!

Now you will see how to use a python action to show a web page. It is the same general idea as opening a site in an external browser, but it requires no browser on the user's system since it uses the Qt QWebView class (which is a webkit based html widget) to display the content in a pop-up window.

Let us use Wikipedia this time. So the URL you request will look like this:

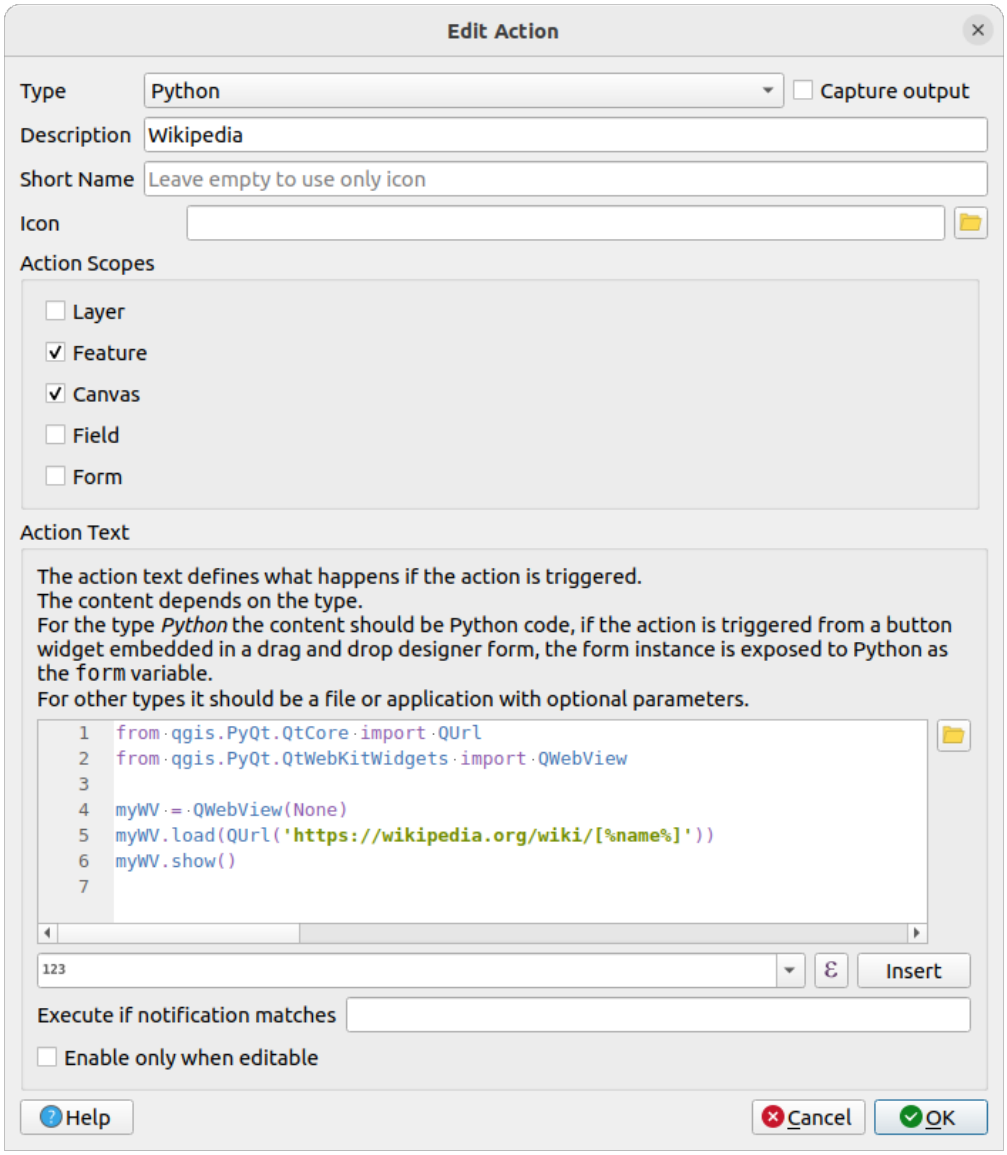
`https://wikipedia.org/wiki/SEARCH_PHRASE`

To create the layer action:

1. Open the *Layer Properties* dialog and head over to the *Actions* tab.
2. Set up a new action using the following properties for the action:
   - *Type*: `Python`
   - *Description*: `Wikipedia`
   - *Scope*: `Feature`, `Canvas`
   - *Action Text*:

```
from qgis.PyQt.QtCore import QUrl
from qgis.PyQt.QtWebKitWidgets import QWebView

myWV = QWebView(None)
myWV.load(QUrl('https://wikipedia.org/wiki/[%name%]'))
myWV.show()
```



There are a couple of things going on here:

- `[%name%]` will be replaced by the actual attribute value when the action is invoked (as before).
- The code simply creates a new `QWebView` instance, sets its URL, and then calls `show()` on it to make it visible as a window on the user's desktop.

You could also use this approach to display an image without requiring that the users have a particular image viewer on their system.

3. Try to use the methods described above to load a Wikipedia page using the Wikipedia action you just created.

5.4.5 In Conclusion

Actions allow you to give your map extra functionality, useful to the end-user who views the same map in QGIS. Due to the fact that processes called can be shell commands for any operating system, as well as Python, the sky is the limit in terms of the functions you could incorporate!

5.4.6 What's Next?

Now that you've done all kinds of vector data creation, you will learn how to analyze the data to solve problems. That is the topic of the next module.

CAPÍTULO 6

Module: Vector Analysis

Now that you have edited a few features, you must want to know what else one can do with them. Having features with attributes is nice, but when all is said and done, this doesn't really tell you anything that a normal, non-GIS map can't.

A principal vantagem de um SIG é esta: *um SIG pode responder a perguntas.*

For the next three modules, we'll endeavor to answer a *research question* using GIS functions. For example, you are an estate agent and you are looking for a residential property in Swellendam for clients who have the following criteria:

1. It needs to be in Swellendam.
2. It must be within reasonable driving distance of a school (say 1km).
3. It must be more than 100m squared in size.
4. Closer than 50m to a main road.
5. Closer than 500m to a restaurant.

Within the next few modules, we'll harness the power of GIS analysis tools to locate suitable farm properties for this new residential development.

6.1 Lesson: Reprojecting and Transforming Data

Let us talk about Coordinate Reference Systems (CRSs) again. We have touched on this briefly before, but haven't discussed what it means practically.

The goal for this lesson: To reproject and transform vector datasets.

6.1.1 ??? Follow Along: Projections

The CRS that all the data, as well as the map itself are in right now is called *WGS84*. This is a very common Geographic Coordinate System (GCS) for representing data. But there's a problem, as we will see.

1. Save your current map
2. Then open the map of the world which you will find under `exercise_data/world/world.qgs`
3. Zoom in to South Africa by using the *Zoom In* tool
4. Try setting a scale in the *Scale* field, which is in the *Statusbar* along the bottom of the screen. While over South Africa, set this value to `1:5 000 000` (one to five million).
5. Pan around the map while keeping an eye on the *Scale* field

Notice the scale changing? That's because you are moving away from the one point that you zoomed into at `1:5 000 000`, which was at the center of your screen. All around that point, the scale is different.

To understand why, think about a globe of the Earth. It has lines running along it from North to South. These longitude lines are far apart at the equator, but they meet at the poles.

In a GCS, you are working on this sphere, but your screen is flat. When you try to represent the sphere on a flat surface, distortion occurs, similar to what would happen if you cut open a tennis ball and tried to flatten it out. What this means on a map is that the longitude lines stay equally far apart from each other, even at the poles (where they are supposed to meet). This means that, as you travel away from the equator on your map, the scale of the objects that you see gets larger and larger. What this means for us, practically, is that there is no constant scale on our map!

To solve this, let's use a Projected Coordinate System (PCS) instead. A PCS «projects» or converts the data in a way that makes allowance for the scale change and corrects it. Therefore, to keep the scale constant, we should reproject our data to use a PCS.

6.1.2 ??? Follow Along: «On the Fly» Reprojection

By default, QGIS reprojects data «on the fly». What this means is that even if the data itself is in another CRS, QGIS can project it as if it were in a CRS of your choice.

You can change the CRS of the project by clicking on the Current projection button in the bottom right corner of QGIS.

1. In the dialog that appears, type the word `global` into the *Filter* field. A few CRSs should appear in the *Predefined Reference Systems* field below.
2. Select *WGS 84 / NSIDC EASE-Grid 2.0 Global | EPSG:6933* entry by clicking on it, and then click *OK*.

   Notice how the shape of South Africa changes. All projections work by changing the apparent shapes of objects on Earth.
3. Zoom to a scale of `1:5 000 000` again, as before.
4. Pan around the map.

   Notice how the scale stays the same!

«On the fly» reprojection is also used for combining datasets that are in different CRSs.

1. Add another vector layer to your map which has the data for South Africa only. You will find it as `exercise_data/world/RSA.shp`.
2. Load it. A quick way to see its CRS is by hovering the mouse over the layer in the legend. It is `EPSG:3410`.

   What do you notice?

   The layer is visible even if it has a different CRS from the *continents* one.

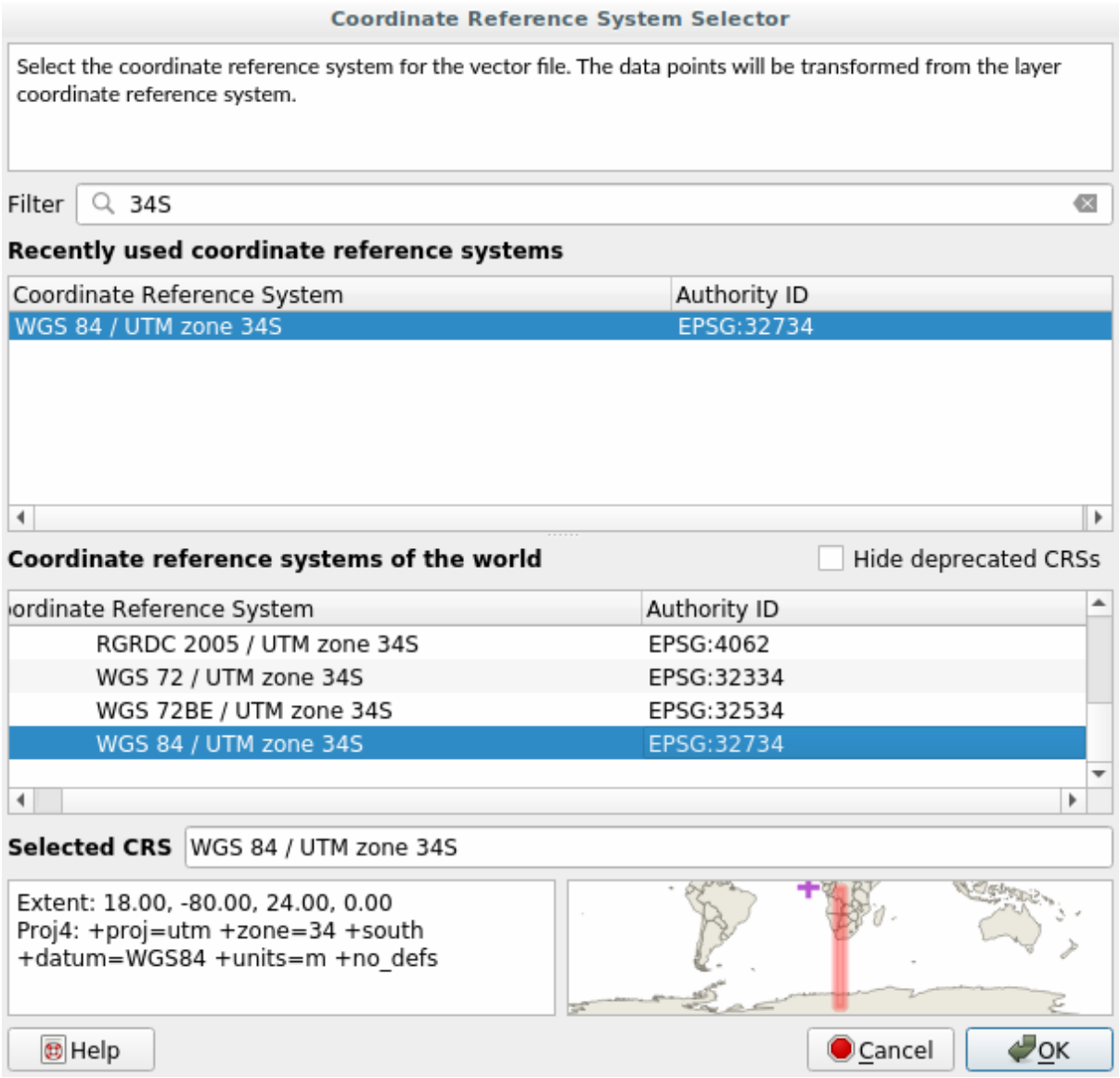
6.1.3 ??? Follow Along: Saving a Dataset to Another CRS

Sometimes you need to export an existing dataset with another CRS. As we will see in the next lesson, if you need to make distance calculations on layer, it is always better to have the layer in a projected coordinate system.

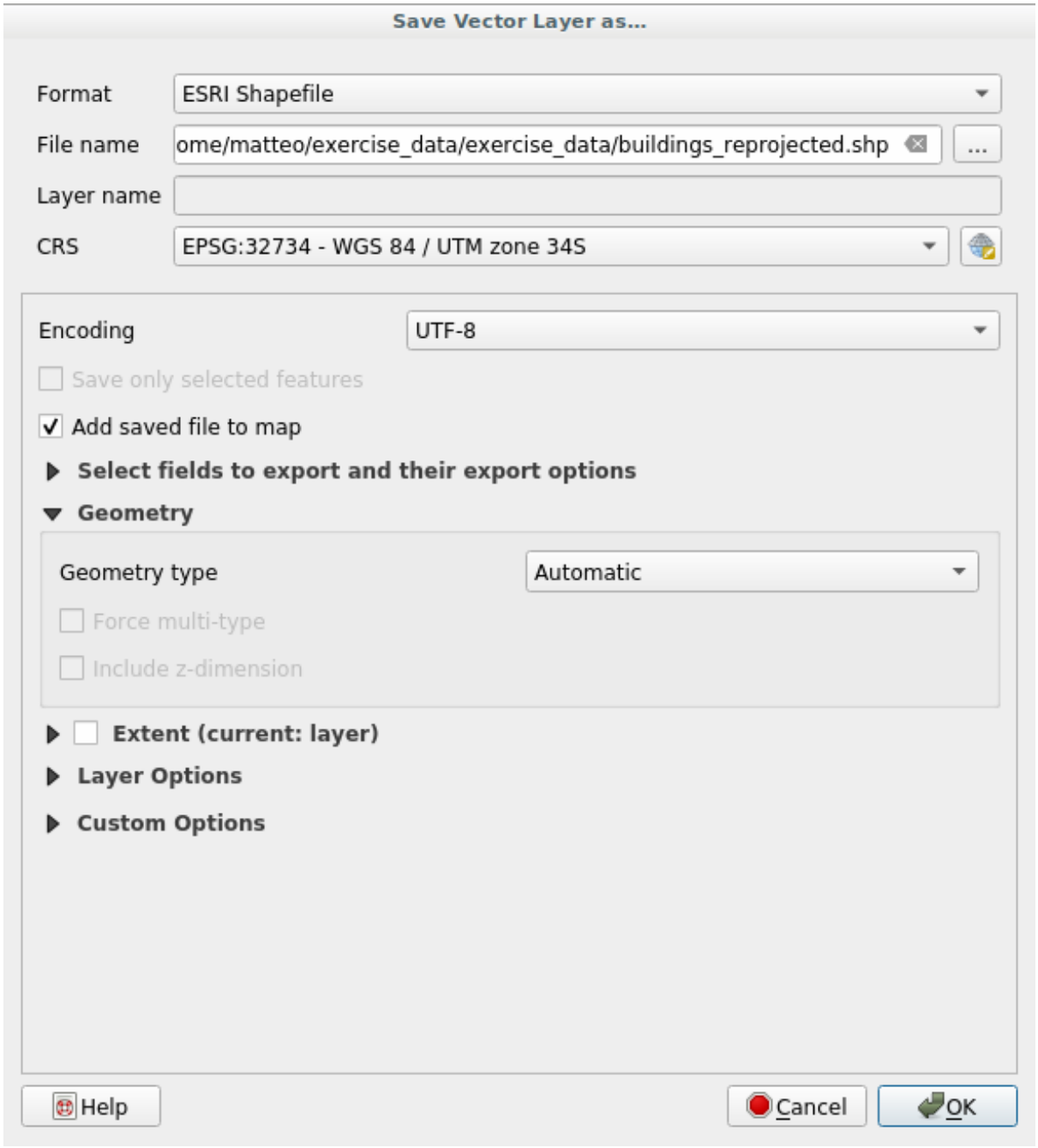
Be aware that the "on the fly" reprojection is related to the **project** and not to single layers. This means that a layer can have a different CRS from the project even if you see it in the *correct* position.

You can easily export the layer with another CRS.

1. Add the `buildings` dataset from `training_data.gpkg`
2. Right-click on the `buildings` layer in the *Layers* panel
3. Select *Export* ► *Save Features As...* in the menu that appears. You will be shown the *Save Vector Layer as...* dialog.
4. Click on the *Browse* button next to the *File name* field
5. Navigate to `exercise_data/` and specify the name of the new layer as `buildings_reprojected.shp`.
6. Change the value of the *CRS*. Only the recent CRSs used will be shown in the drop-down menu. Click on the Select projection button next to the drop-down menu.
7. The *Coordinate Reference System Selector* dialog will appear. In its *Filter* field, search for `34S`.
8. Select *WGS 84 / UTM zone 34S | EPSG:32734* from the list



9. Leave the other options unchanged. The *Save Vector Layer as...* dialog now looks like this:



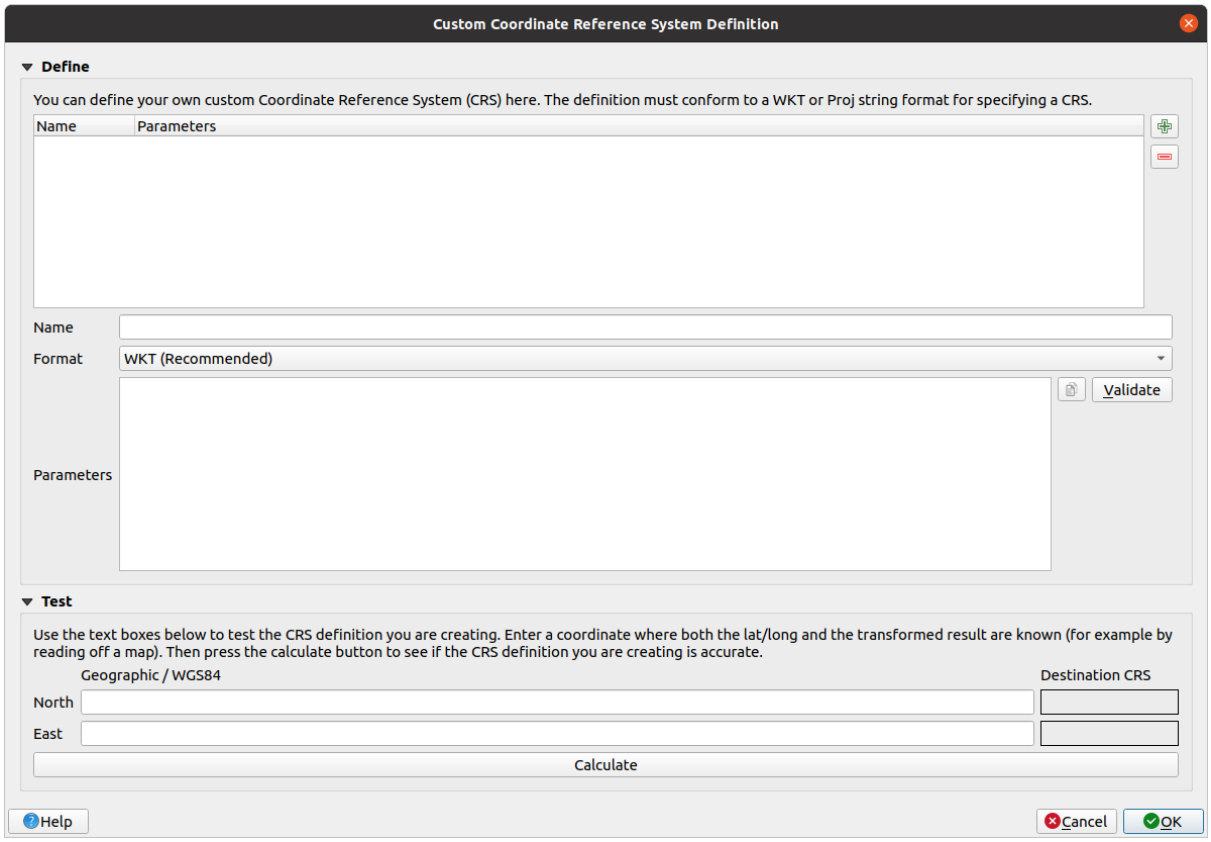
10. Click *OK*

You can now compare the old and new projections of the layer and see that they are in two different CRS but they are still overlapping.

6.1.4 ??? Follow Along: Creating Your Own Projection

There are many more projections than just those included in QGIS by default. You can also create your own projections.

1. Start a new map
2. Load the `world/oceans.shp` dataset
3. Go to *Settings* ► *Custom Projections...* and you will see this dialog.



4. Click on the Add new CRS button to create a new projection
5. An interesting projection to use is called `Van der Grinten I`. Enter its name in the *Name* field.

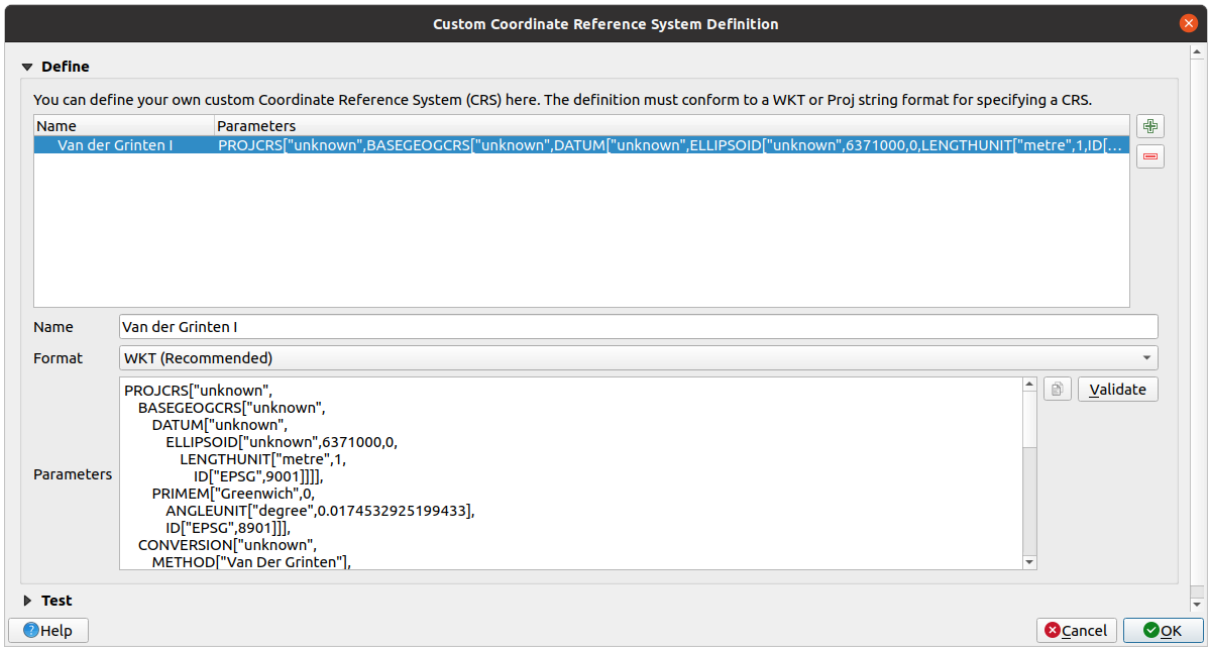
   This projection represents the Earth on a circular field instead of a rectangular one, as most other projections do.
6. In *Format*, select *WKT (Recommended)*
7. Add the following string in the *Parameters* field:

```
PROJCRS["unknown",
    BASEGEOGCRS["unknown",
        DATUM["unknown",
            ELLIPSOID["unknown",6371000,0,
                LENGTHUNIT["metre",1,
                    ID["EPSG",9001]]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8901]]],
    CONVERSION["unknown",
        METHOD["Van Der Grinten"],
        PARAMETER["Longitude of natural origin",0,
```

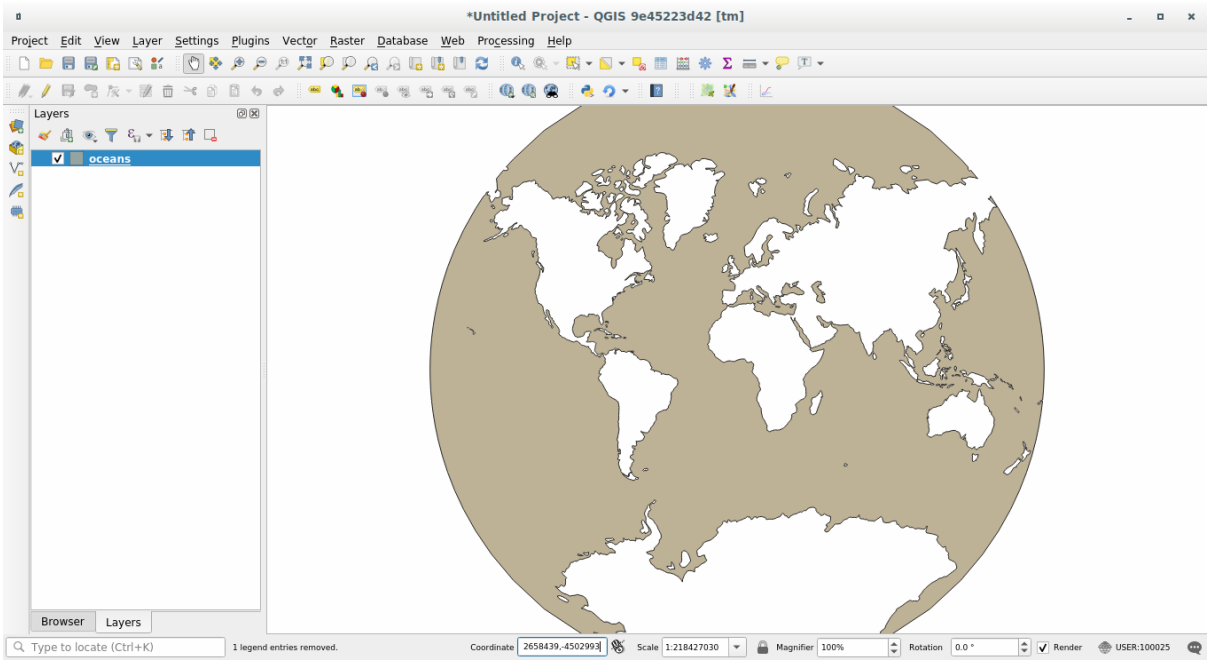
(continues on next page)

(continuação da página anterior)

```
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["False easting",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1,
                ID["EPSG",9001]]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1,
                ID["EPSG",9001]]]]
```



8. Click *OK*
9. Click on the Current CRS button at the right of the status bar to change the project CRS
10. Choose your newly defined projection (search for its name in the *Filter* field)
11. On applying this projection, the map will be reprojected thus:



6.1.5 In Conclusion

Different projections are useful for different purposes. By choosing the correct projection, you can ensure that the features on your map are being represented accurately.

6.1.6 Further Reading

Materials for the *Advanced* section of this lesson were taken from this article.

Read further information on Coordinate Reference Systems.

6.1.7 What's Next?

In the next lesson you will learn how to analyze vector data using QGIS" various vector analysis tools.

6.2 Lesson: Vector Analysis

Vector data can also be analyzed to reveal how different features interact with each other in space. There are many different analysis-related functions, so we won't go through them all. Rather, we will pose a question and try to solve it using the tools that QGIS provides.

The goal for this lesson: To ask a question and solve it using analysis tools.

6.2.1 ??? The GIS Process

Before we start, it would be useful to give a brief overview of a process that can be used to solve a problem. The way to go about it is:

1. State the Problem
2. Get the Data
3. Analyze the Problem
4. Present the Results

6.2.2 ??? The Problem

Let's start off the process by deciding on a problem to solve. For example, you are an estate agent and you are looking for a residential property in Swellendam for clients who have the following criteria:

1. It needs to be in Swellendam
2. It must be within reasonable driving distance of a school (say 1km)
3. It must be more than 100m squared in size
4. Closer than 50m to a main road
5. Closer than 500m to a restaurant

6.2.3 ??? The Data

To answer these questions, we are going to need the following data:

1. The residential properties (buildings) in the area
2. The roads in and around the town
3. The location of schools and restaurants
4. The size of buildings

These data are available through OSM, and you should find that the dataset you have been using throughout this manual also can be used for this lesson.

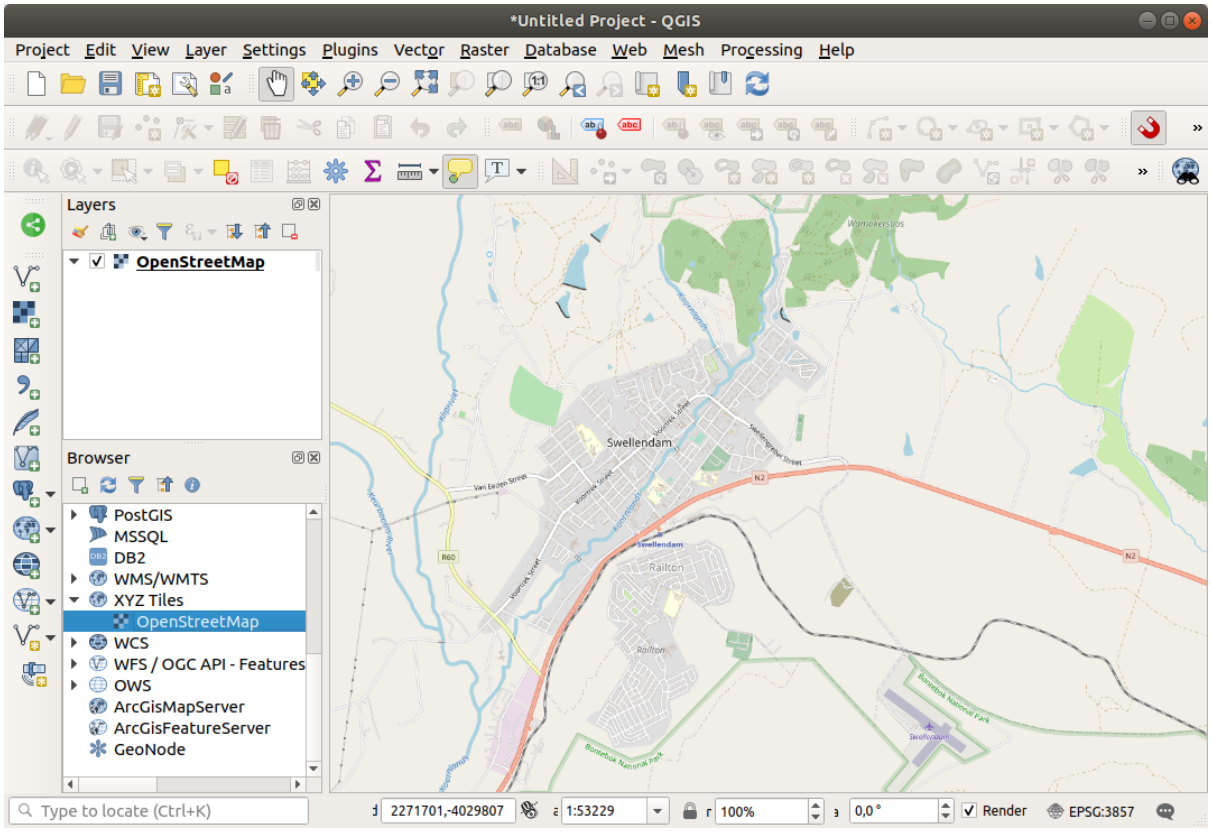
If you want to download data from another area, jump to the *Introduction Chapter* to read how to do it.

Nota: Although OSM downloads have consistent data fields, the coverage and detail does vary. If you find that your chosen region does not contain information on restaurants, for example, you may need to chose a different region.

6.2.4 ??? Follow Along: Start a Project and get the Data

We first need to load the data to work with.

1. Start a new QGIS project
2. If you want, you can add a background map. Open the *Browser* and load the *OSM* background map from the *XYZ Tiles* menu.



3. In the `training_data.gpkg` Geopackage database, you will find most the datasets we will use in this chapter:

    1. `buildings`
    2. `roads`
    3. `restaurants`
    4. `schools`

    Load them, and also `landuse.sqlite`.

4. Zoom to the layer extent to see Swellendam, South Africa

    Before proceeding we will filter the *roads* layer, in order to have only some specific road types to work with.

    Some roads in OSM datasets are listed as `unclassified`, `tracks`, `path` and `footway`. We want to exclude these from our dataset and focus on the other road types, more suitable for this exercise.

    Moreover, OSM data might not be updated everywhere, and we will also exclude `NULL` values.

5. Right click on the `roads` layer and choose *Filter…*.

6. In the dialog that pops up we filter these features with the following expression:

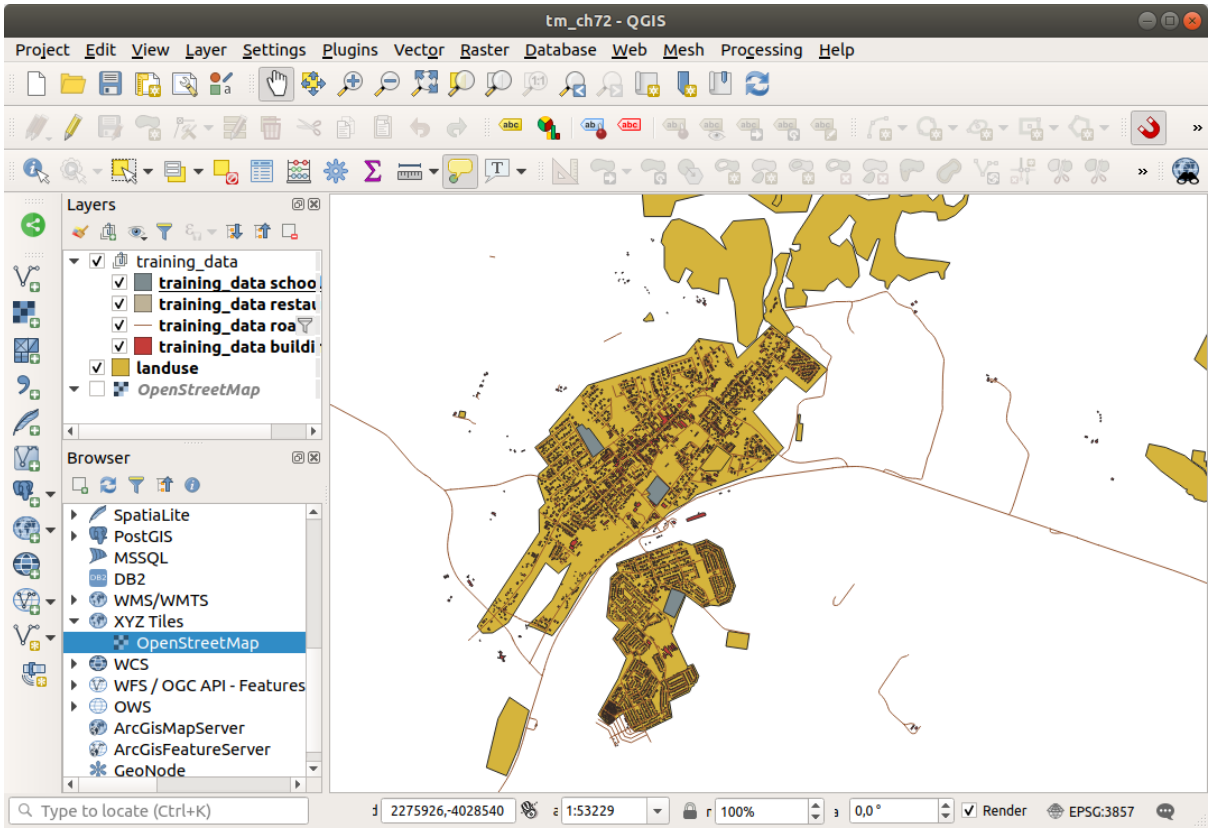
```
"highway" NOT IN ('footway', 'path', 'unclassified', 'track') AND "highway" IS␣
↪NOT NULL
```

    The concatenation of the two operators `NOT` and `IN` excludes all the features that have these attribute values in the `highway` field.

    `IS NOT NULL` combined with the `AND` operator excludes roads with no value in the `highway` field.

    Note the icon next to the *roads* layer. It helps you remember that this layer has a filter activated, so some features may not be available in the project.

The map with all the data should look like the following one:



6.2.5 ??? Try Yourself: Convert Layers" CRS

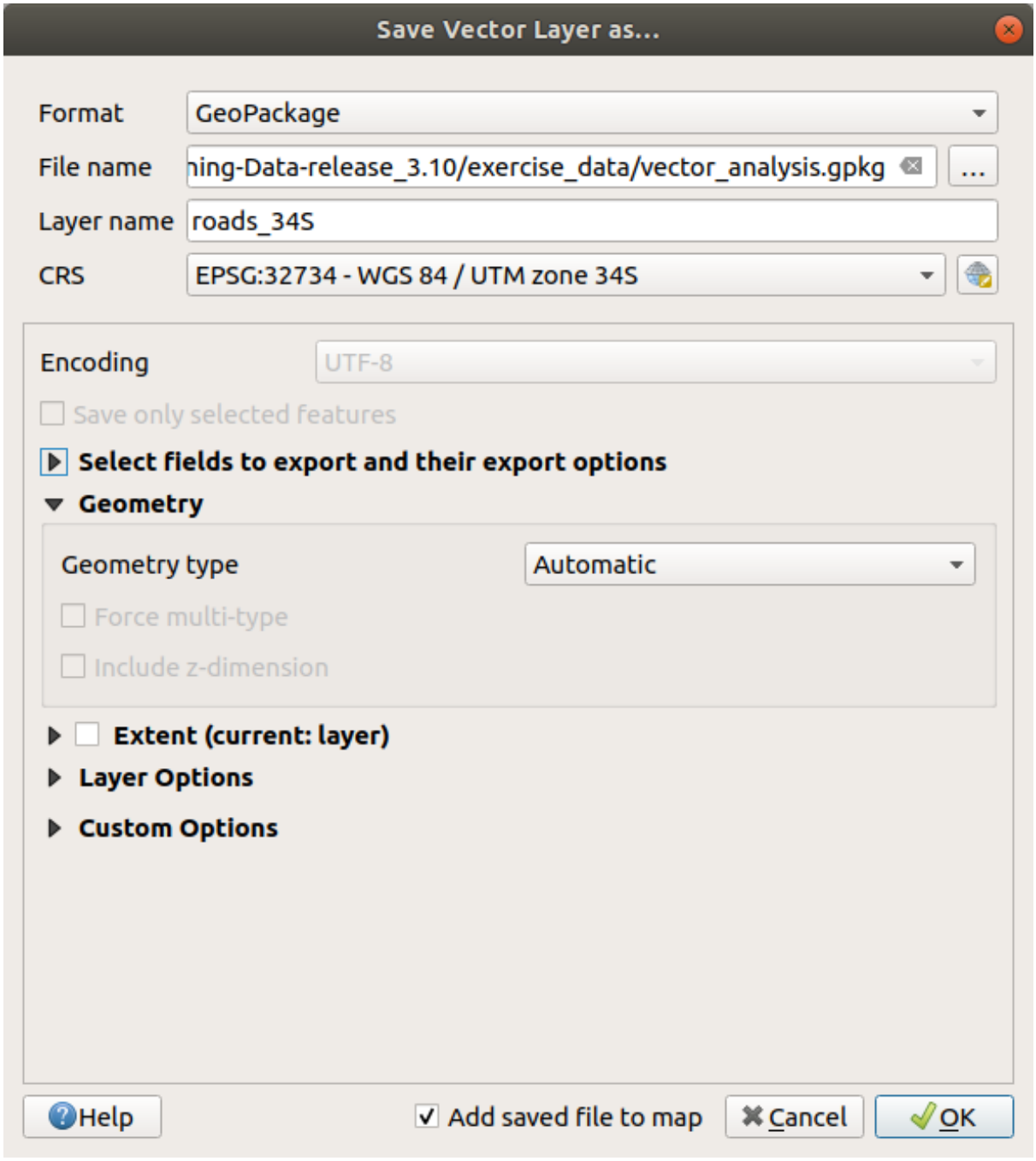
Because we are going to be measuring distances within our layers, we need to change the layers" CRS. To do this, we need to select each layer in turn, save the layer to a new one with our new projection, then import that new layer into our map.

You have many different options, e.g. you can export each layer as an ESRI Shapefile format dataset, you can append the layers to an existing GeoPackage file, or you can create another GeoPackage file and fill it with the new reprojected layers. We will show the last option, so the `training_data.gpkg` will remain clean. Feel free to choose the best workflow for yourself.

Nota: In this example, we are using the *WGS 84 / UTM zone 34S* CRS, but you should use a UTM CRS which is more appropriate for your region.

---

1. Right click the *roads* layer in the *Layers* panel
2. Click *Export –> Save Features As…*
3. In the *Save Vector Layer As* dialog choose *GeoPackage* as *Format*
4. Click on … for the *File name*, and name the new GeoPackage `vector_analysis`
5. Change the *Layer name* to `roads_34S`
6. Change the *CRS* to *WGS 84 / UTM zone 34S*
7. Click on *OK*:



This will create the new GeoPackage database and add the `roads_34S` layer.

8. Repeat this process for each layer, creating a new layer in the `vector_analysis.gpkg` GeoPackage file with `_34S` appended to the original name.

   On macOS, press the *Replace* button in the dialog that pops up to allow QGIS to overwrite the existing GeoPackage.

   Nota: When you choose to save a layer to an existing GeoPackage, QGIS will **add** that layer next to the existing layers in the GeoPackage, if no layer of the same name already exists.

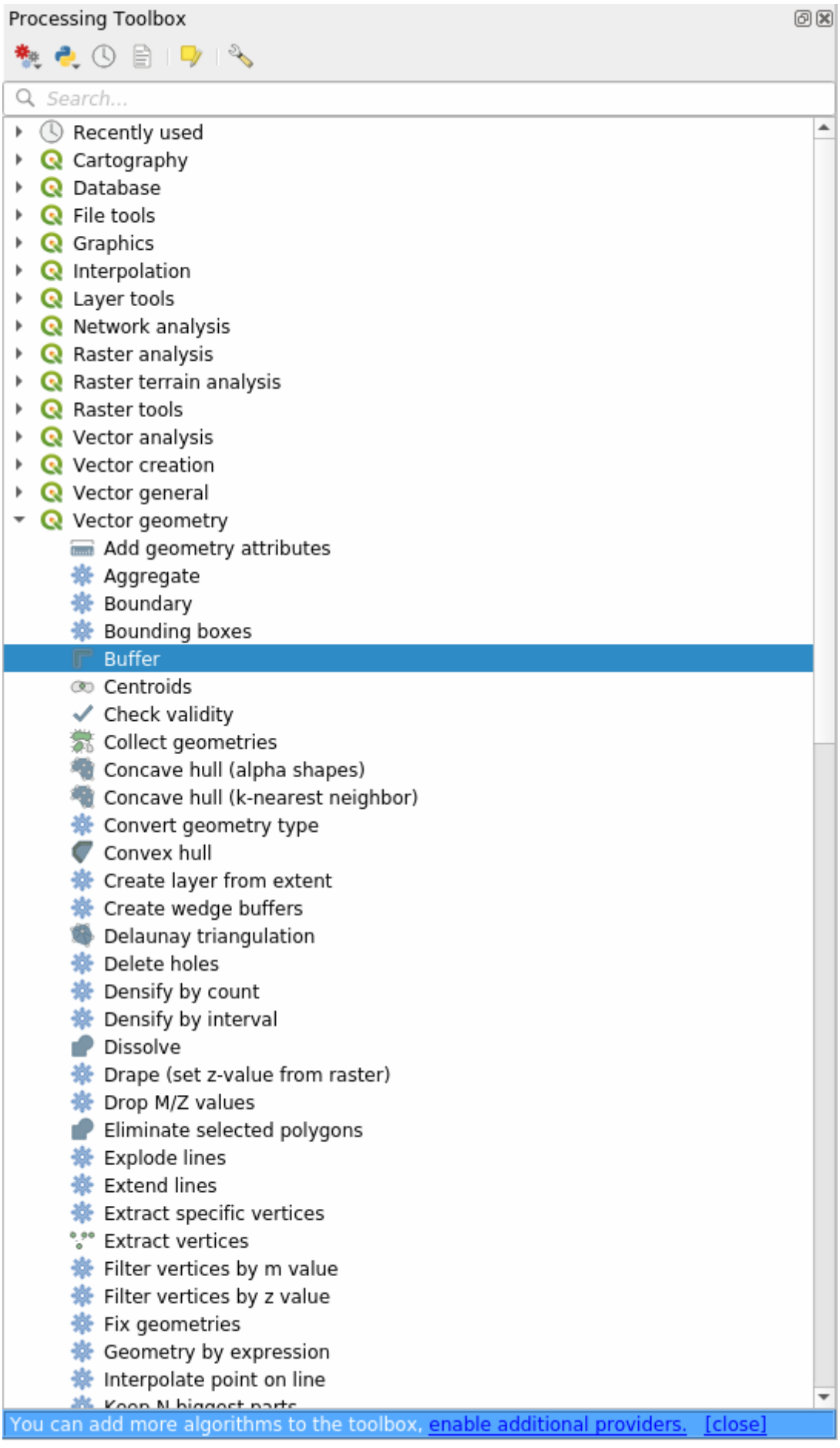
9. Remove each of the old layers from the project
10. Once you have completed the process for all the layers, right click on any layer and click *Zoom to layer extent* to focus the map to the area of interest.

Now that we have converted OSM data to a UTM projection, we can begin our calculations.

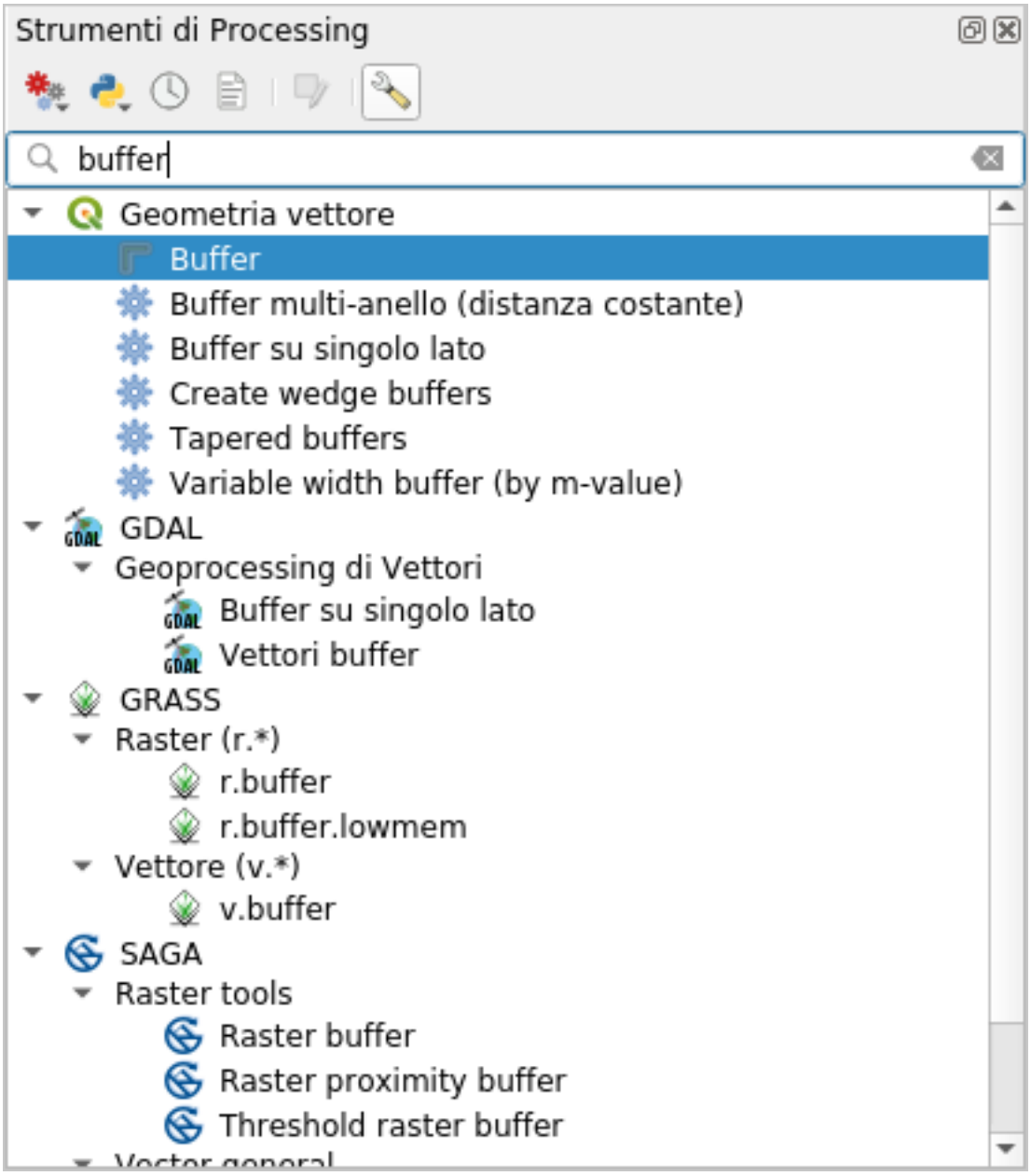
6.2.6 ??? Follow Along: Analyzing the Problem: Distances From Schools and Roads

QGIS allows you to calculate distances between any vector object.

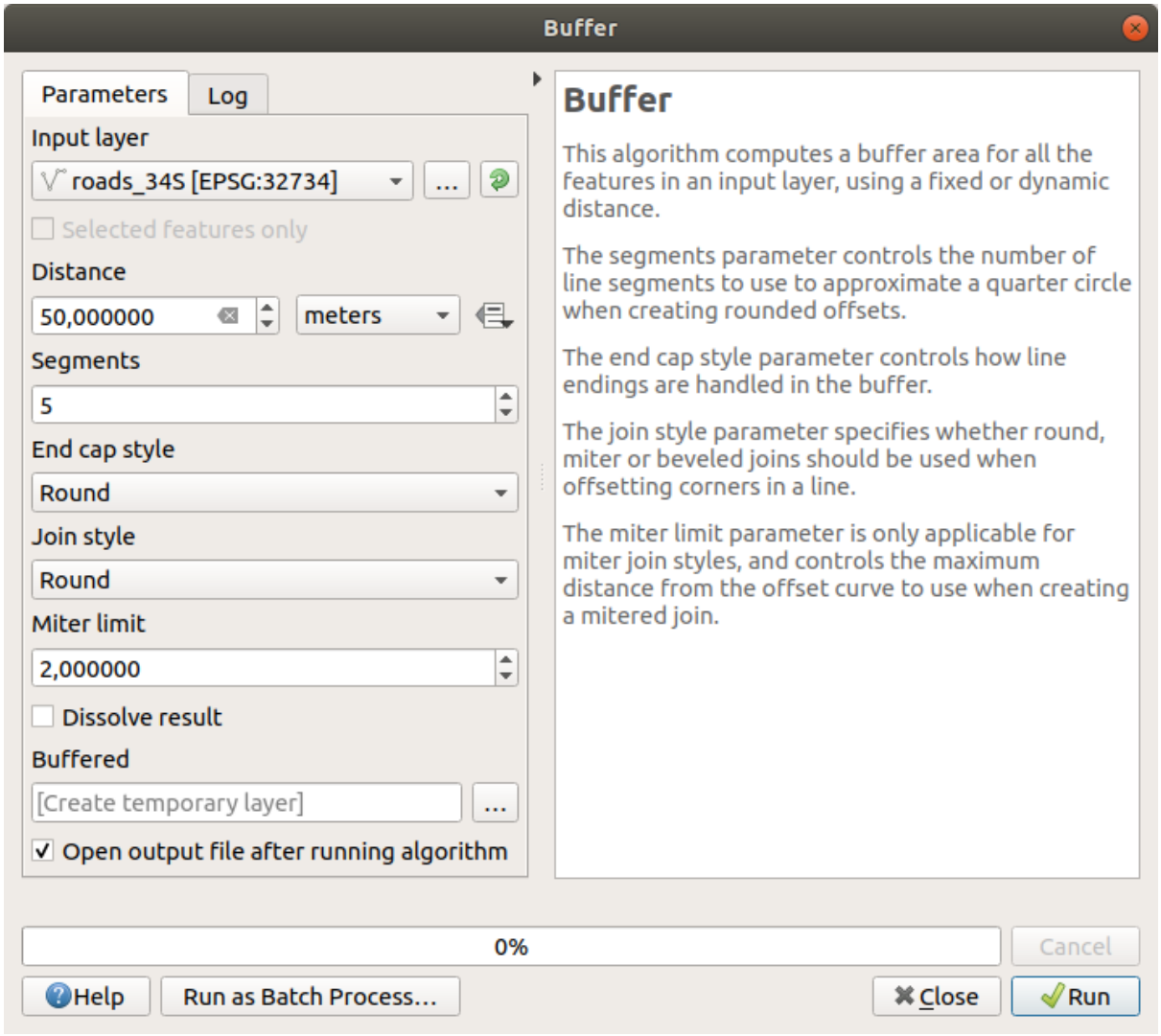
1. Make sure that only the `roads_34S` and `buildings_34S` layers are visible (to simplify the map while you're working)
2. Click on the *Processing* ► *Toolbox* to open the analytical *core* of QGIS. Basically, **all** algorithms (for vector **and** raster analysis) are available in this toolbox.
3. We start by calculating the area around the `roads_34S` by using the *Buffer* algorithm. You can find it in the *Vector Geometry* group.


Processing Toolbox
Search...
Recently used
Cartography
Database
File tools
Graphics
Interpolation
Layer tools
Network analysis
Raster analysis
Raster terrain analysis
Raster tools
Vector analysis
Vector creation
Vector general
Vector geometry
Add geometry attributes
Aggregate
Boundary
Bounding boxes
Buffer
Centroids
Check validity
Collect geometries
Concave hull (alpha shapes)
Concave hull (k-nearest neighbor)
Convert geometry type
Convex hull
Create layer from extent
Create wedge buffers
Delaunay triangulation
Delete holes
Densify by count
Densify by interval
Dissolve
Drape (set z-value from raster)
Drop M/Z values
Eliminate selected polygons
Explode lines
Extend lines
Extract specific vertices
Extract vertices
Filter vertices by m value
Filter vertices by z value
Fix geometries
Geometry by expression
Interpolate point on line
You can add more algorithms to the toolbox, enable additional providers. [close]

Or you can type `buffer` in the search menu in the upper part of the toolbox:



4. Double click on it to open the algorithm dialog
5. Select `roads_34S` as *Input layer*, set *Distance* to 50 and use the default values for the rest of the parameters.



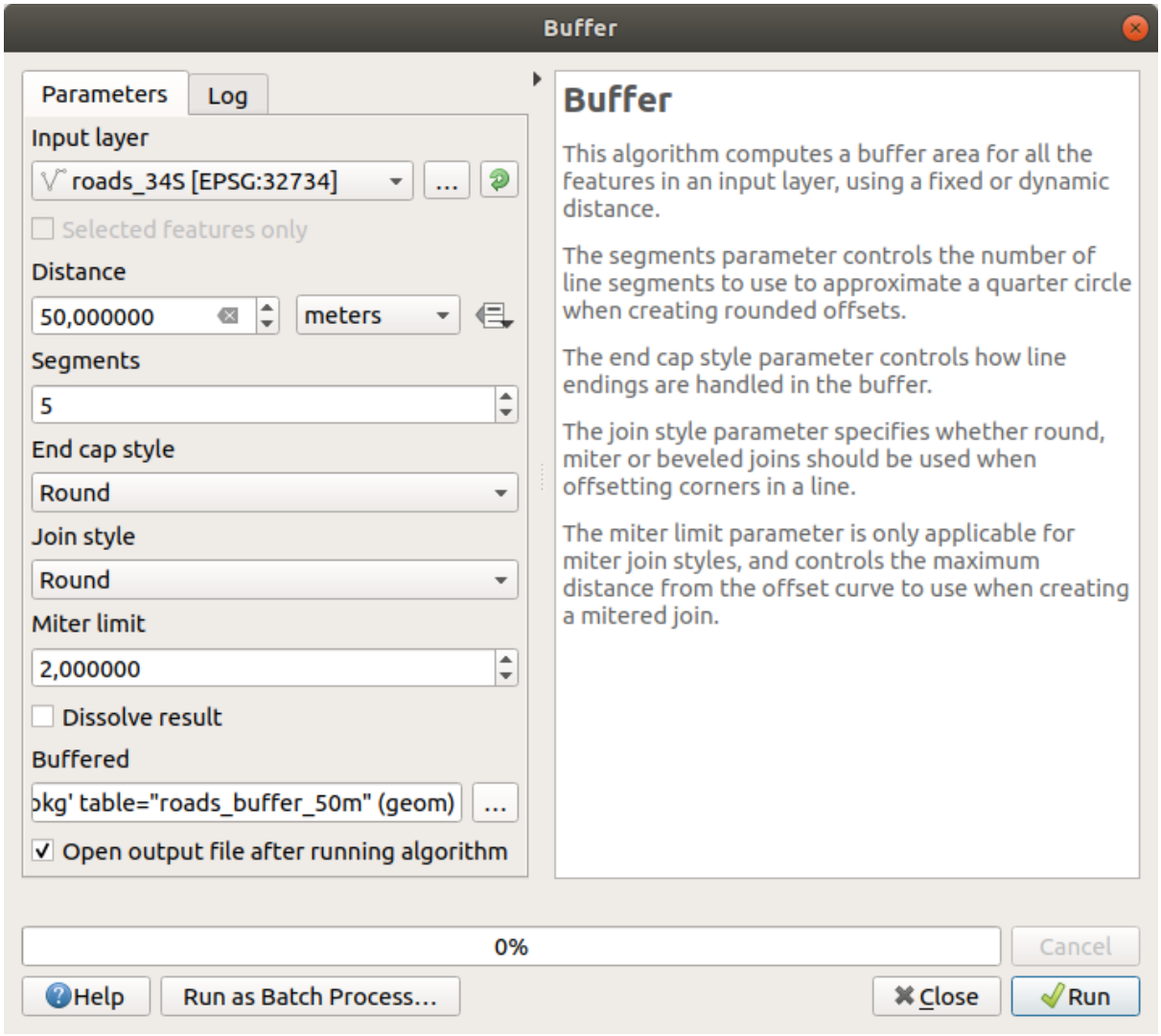
6. The default *Distance* is in meters because our input dataset is in a Projected Coordinate System that uses meter as its basic measurement unit. You can use the combo box to choose other projected units like kilometers, yards, etc.

   Nota: If you are trying to make a buffer on a layer with a Geographical Coordinate System, Processing will warn you and suggest to reproject the layer to a metric Coordinate System.

   ---

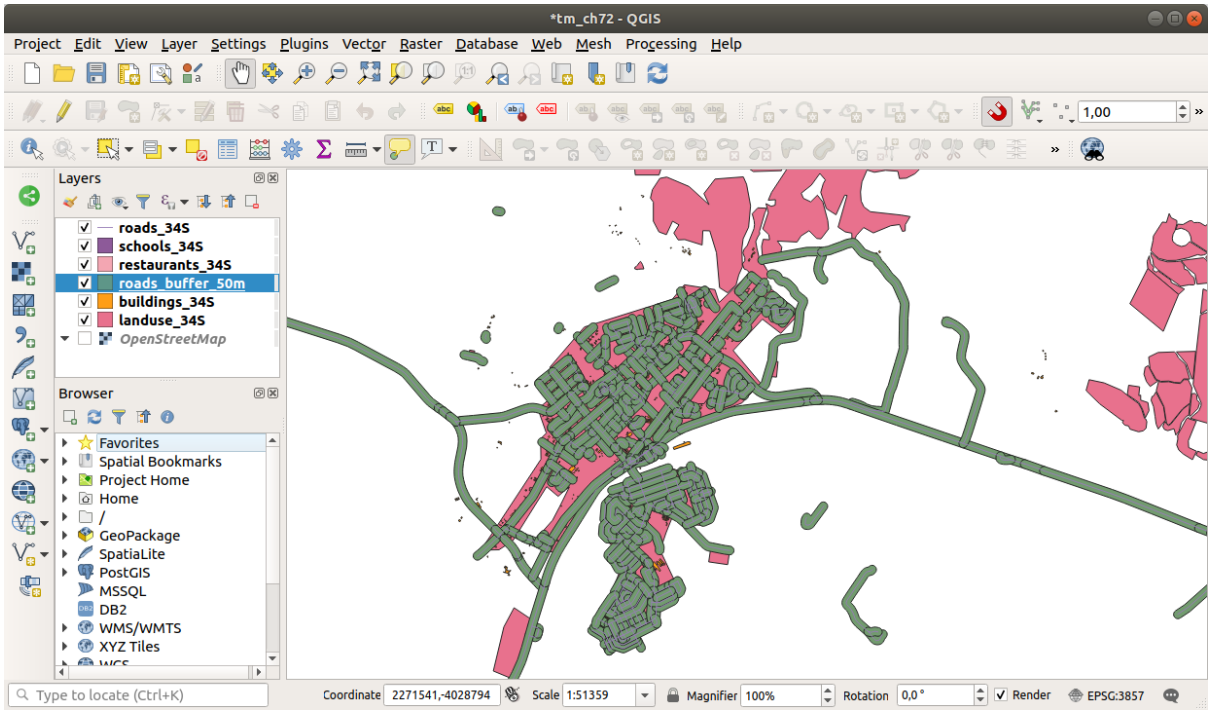
7. By default, *Processing* creates temporary layers and adds them to the *Layers* panel. You can also append the result to the GeoPackage database by:

   1. Clicking on the … button and choose *Save to GeoPackage…*
   2. Naming the new layer `roads_buffer_50m`
   3. Saving it in the `vector_analysis.gpkg` file



8. Click on *Run*, and then close the *Buffer* dialog

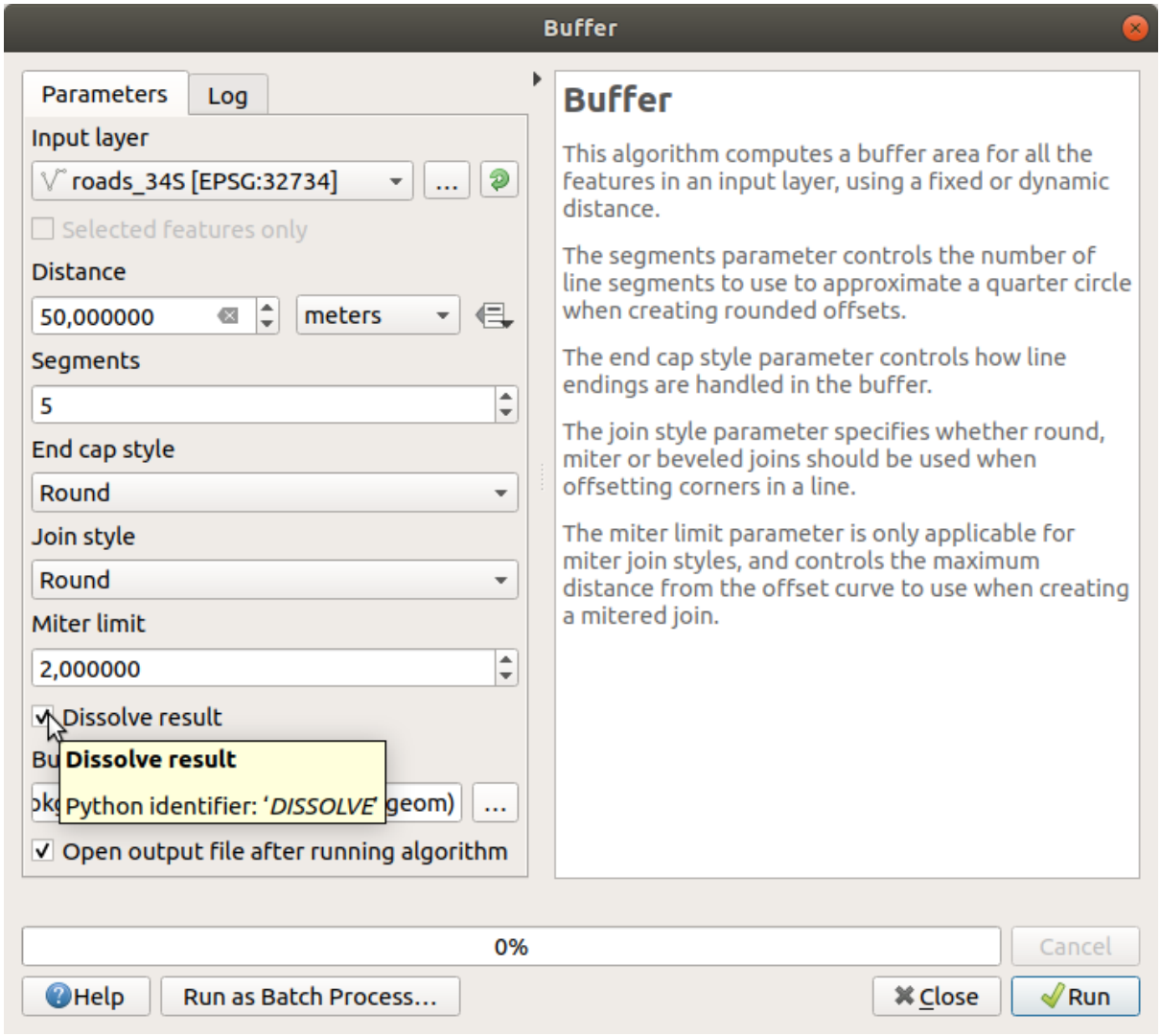
   Now your map will look something like this:



If your new layer is at the top of the *Layers* list, it will probably obscure much of your map, but this gives you all the areas in your region which are within 50m of a road.

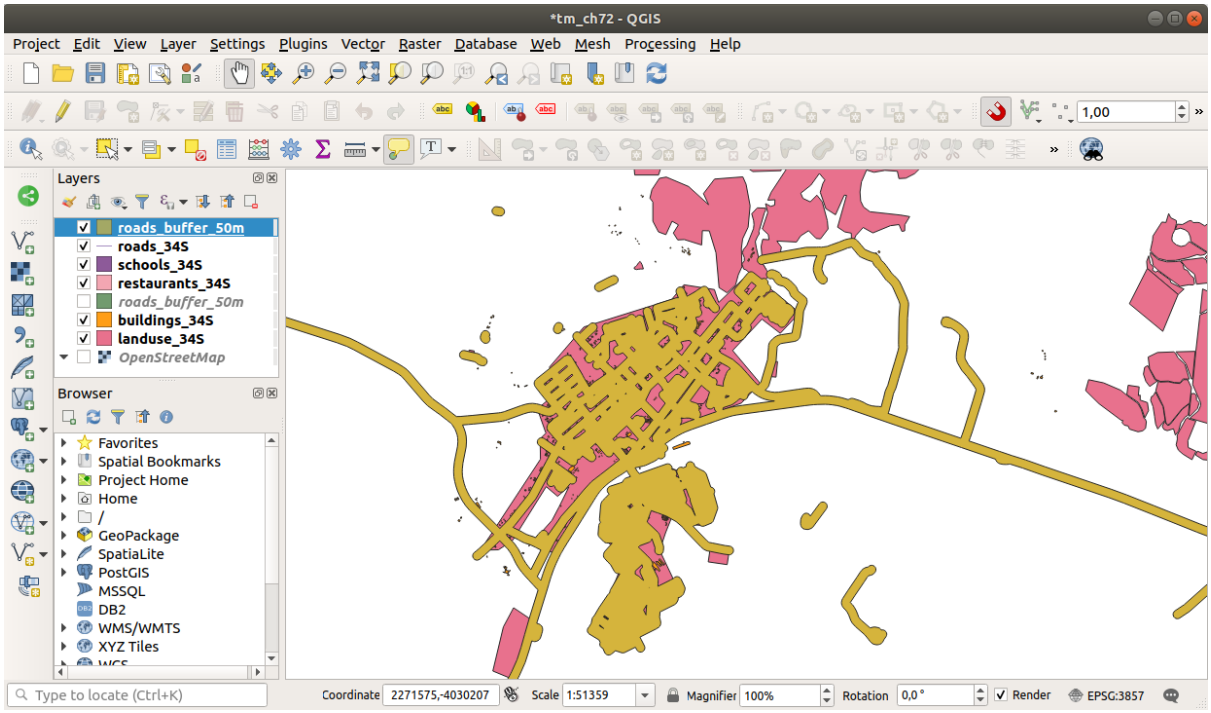
Notice that there are distinct areas within your buffer, which correspond to each individual road. To get rid of this problem:

1. Uncheck the *roads_buffer_50m* layer and re-create the buffer with *Dissolve results* enabled.



2. Save the output as *roads_buffer_50m_dissolved*
3. Click *Run* and close the *Buffer* dialog

Once you have added the layer to the *Layers* panel, it will look like this:



Now there are no unnecessary subdivisions.

Nota: The *Short Help* on the right side of the dialog explains how the algorithm works. If you need more information, just click on the *Help* button in the bottom part to open a more detailed guide of the algorithm.

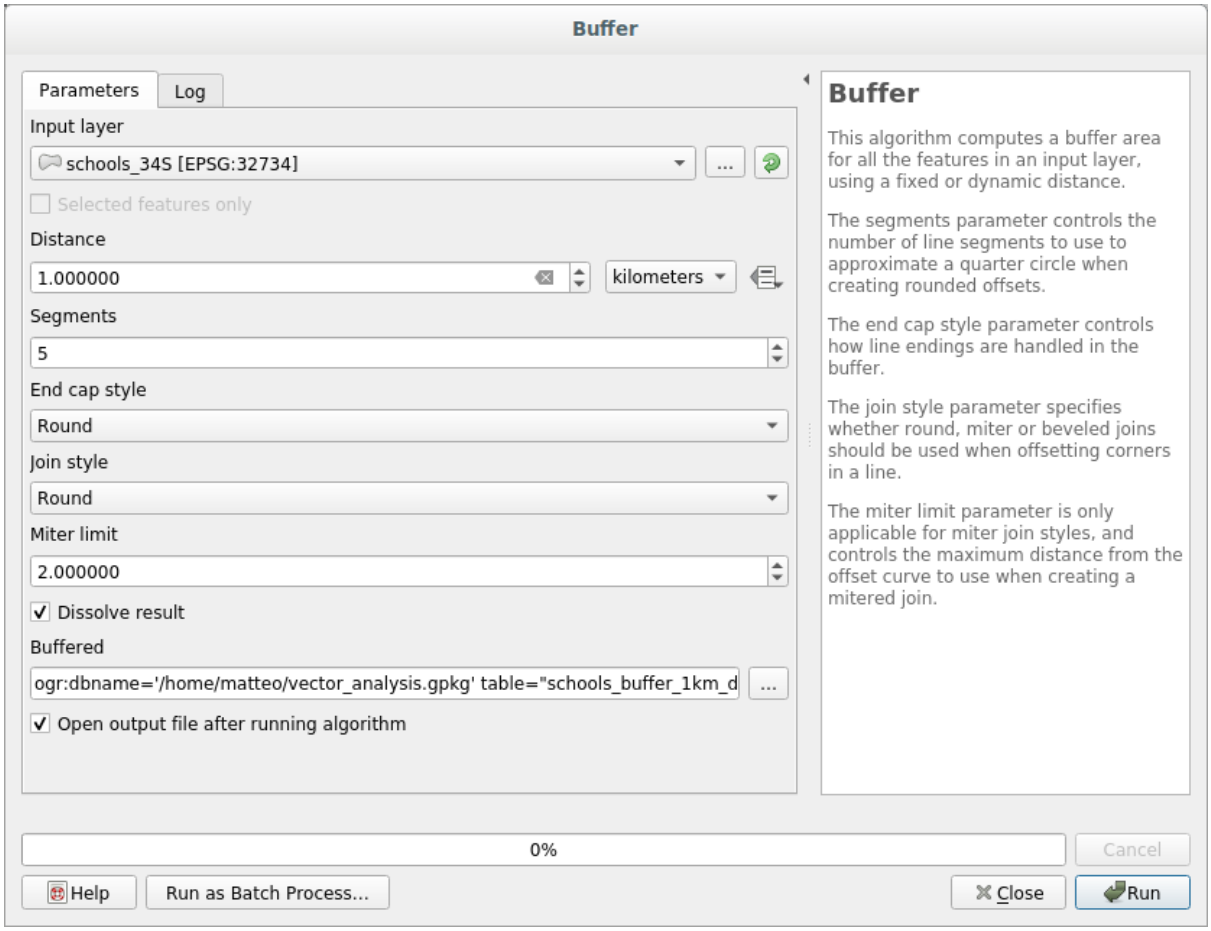
6.2.7 ??? Try Yourself: Distance from schools

Use the same approach as above and create a buffer for your schools.

It shall be `1 km` in radius. Save the new layer in the `vector_analysis.gpkg` file as `schools_buffer_1km_dissolved`.

Answer

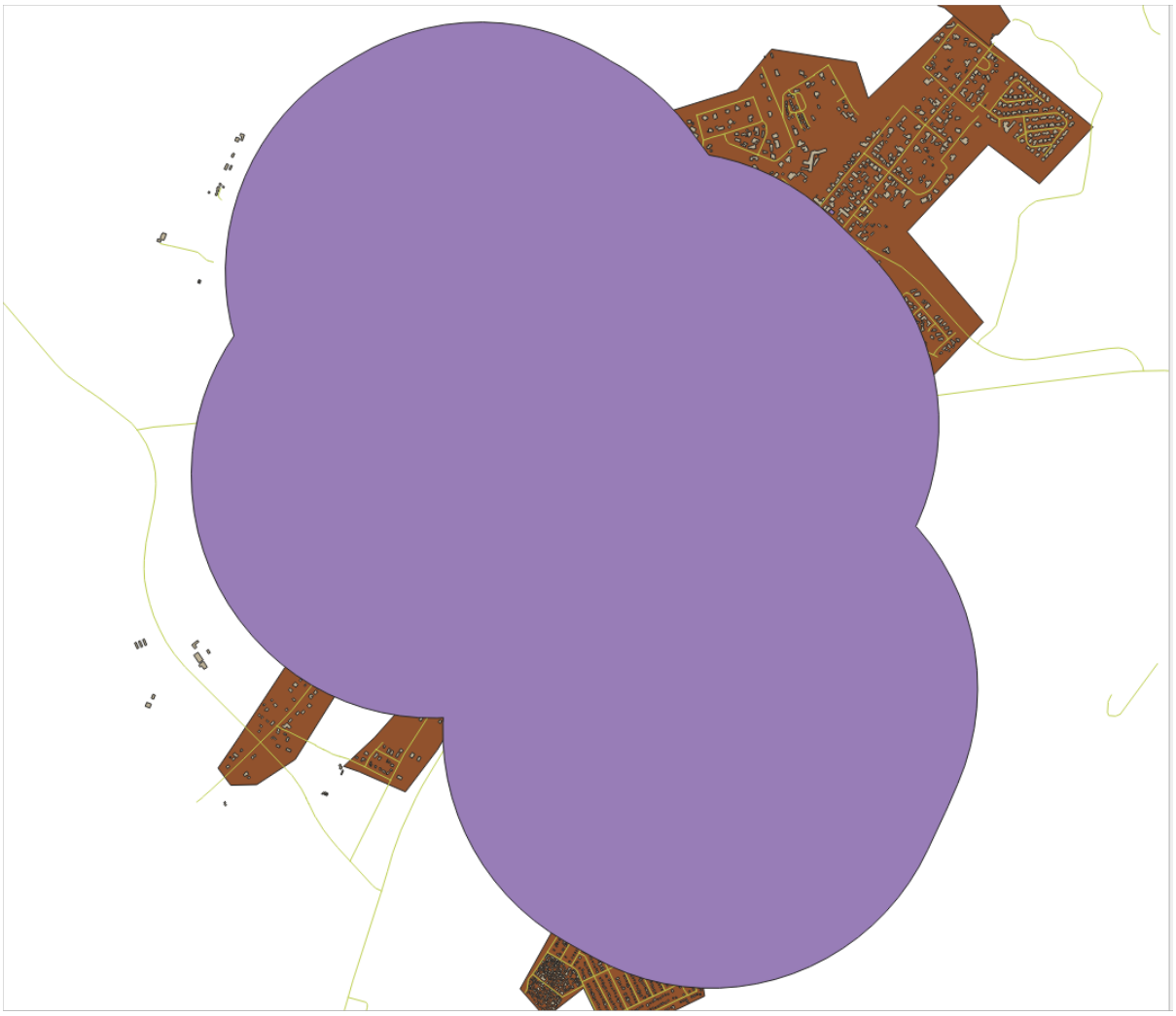
- Your buffer dialog should look like this:



The *Buffer distance* is *1* kilometer.

- The *Segments to approximate* value is set to *20*. This is optional, but it's recommended, because it makes the output buffers look smoother. Compare this:

To this:

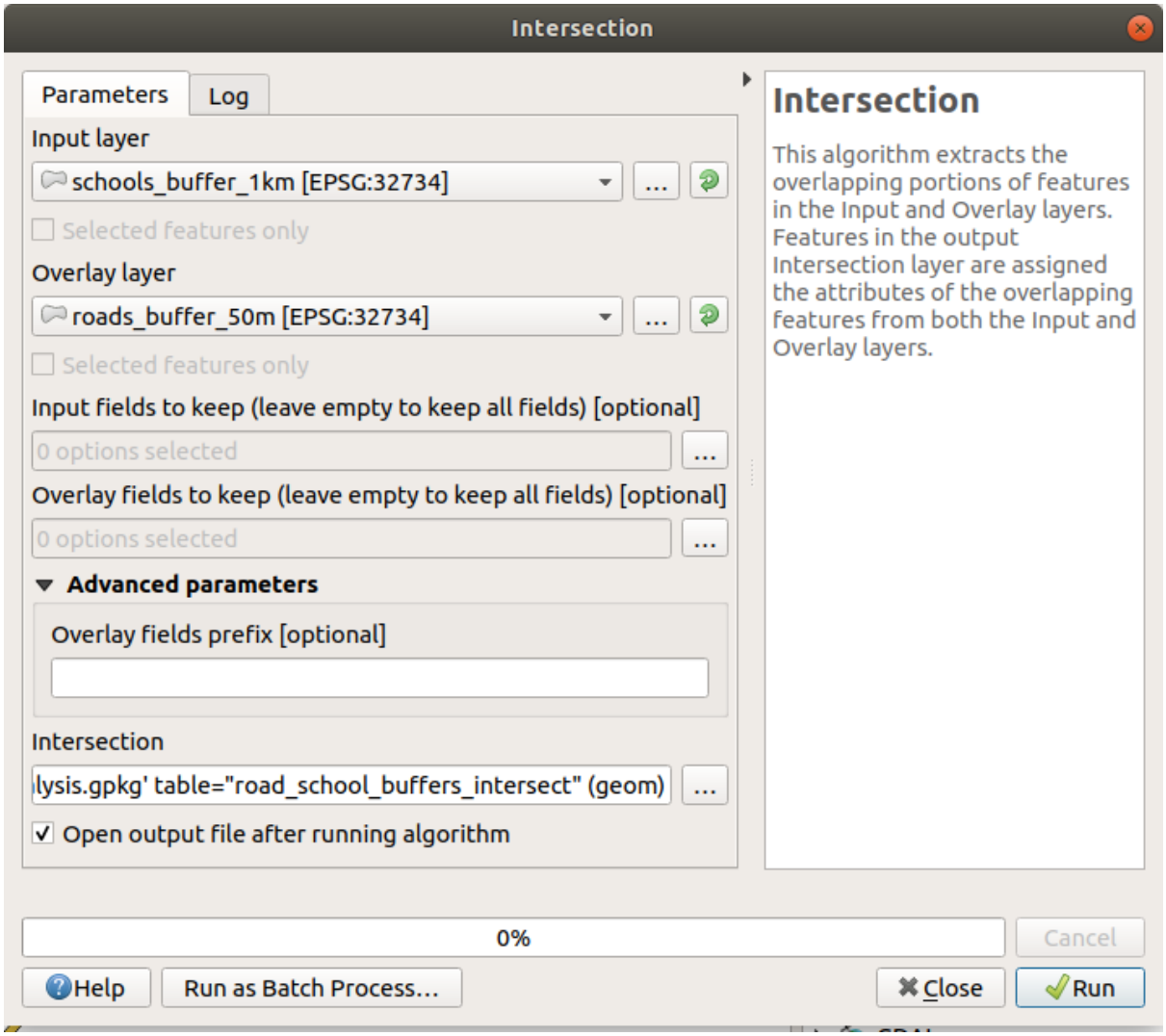


The first image shows the buffer with the *Segments to approximate* value set to *5* and the second shows the value set to *20*. In our example, the difference is subtle, but you can see that the buffer's edges are smoother with the higher value.

6.2.8 ??? Follow Along: Overlapping Areas

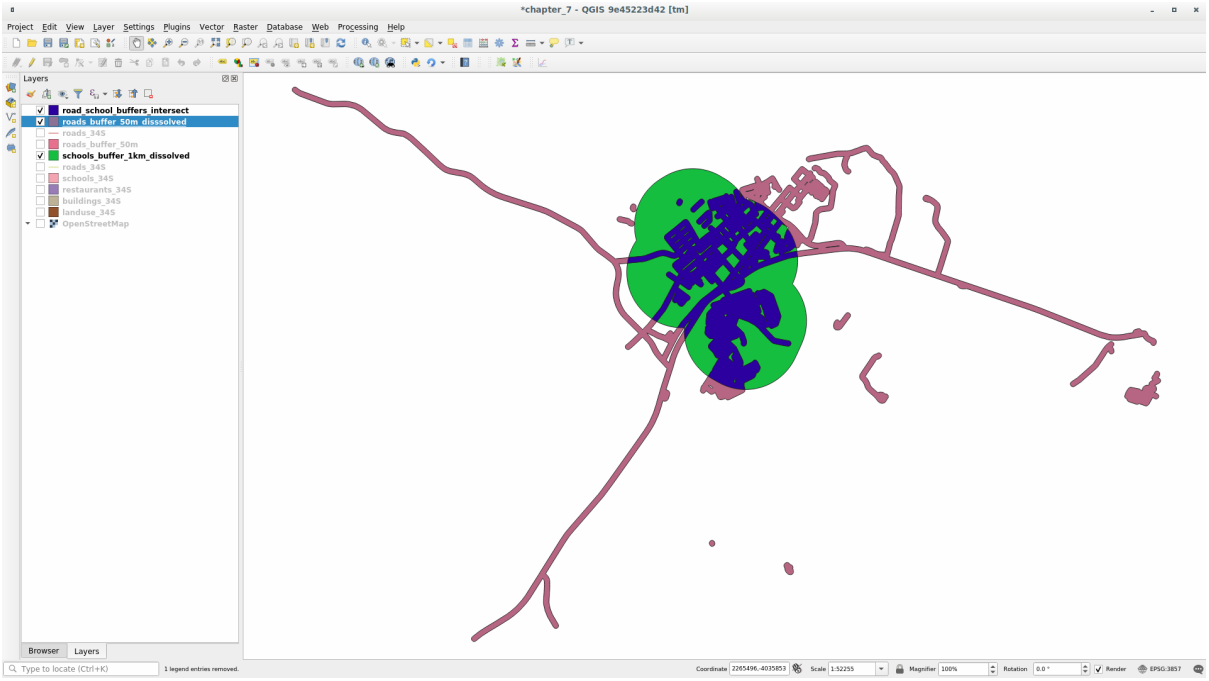
Now we have identified areas where the road is less than 50 meters away and areas where there is a school within 1 km (direct line, not by road). But obviously, we only want the areas where both of these criteria are satisfied. To do that, we will need to use the *Intersect* tool. You can find it in *Vector Overlay* group in the *Processing Toolbox*.

1. Use the two buffer layers as *Input layer* and *Overlay layer*, choose `vector_analysis.gpkg` GeoPackage in *Intersection* with *Layer name* `road_school_buffers_intersect`. Leave the rest as suggested (default).

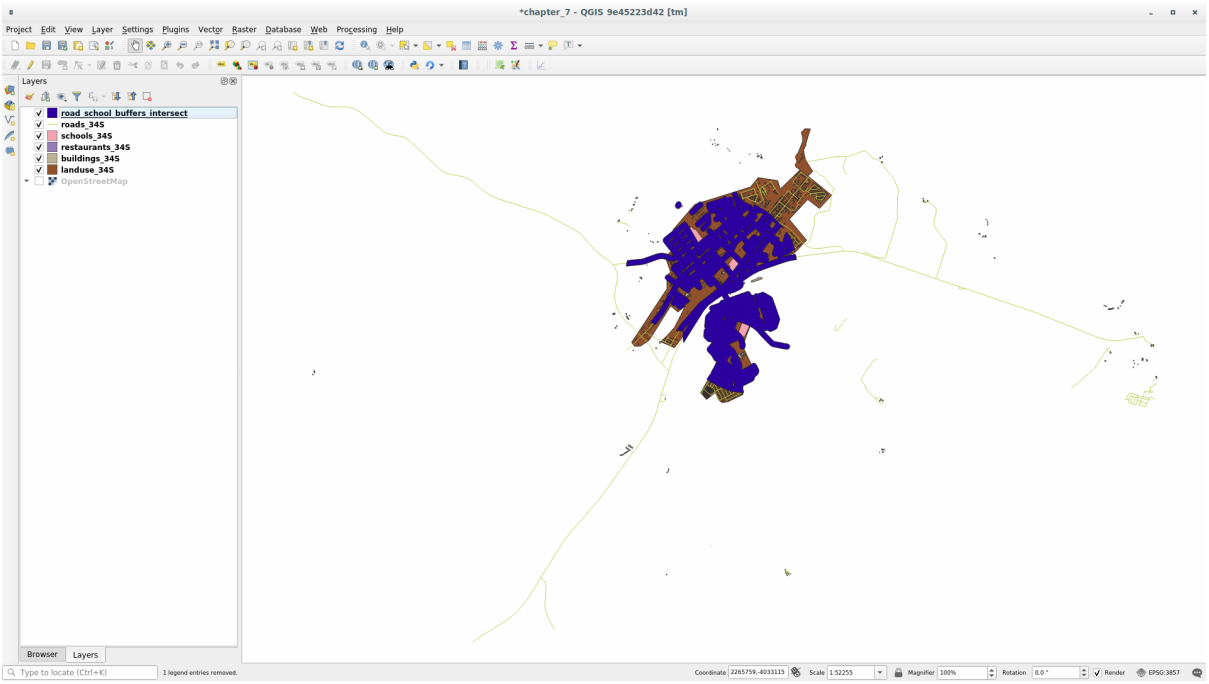


2. Click *Run*.

   In the image below, the blue areas are where both of the distance criteria are satisfied.



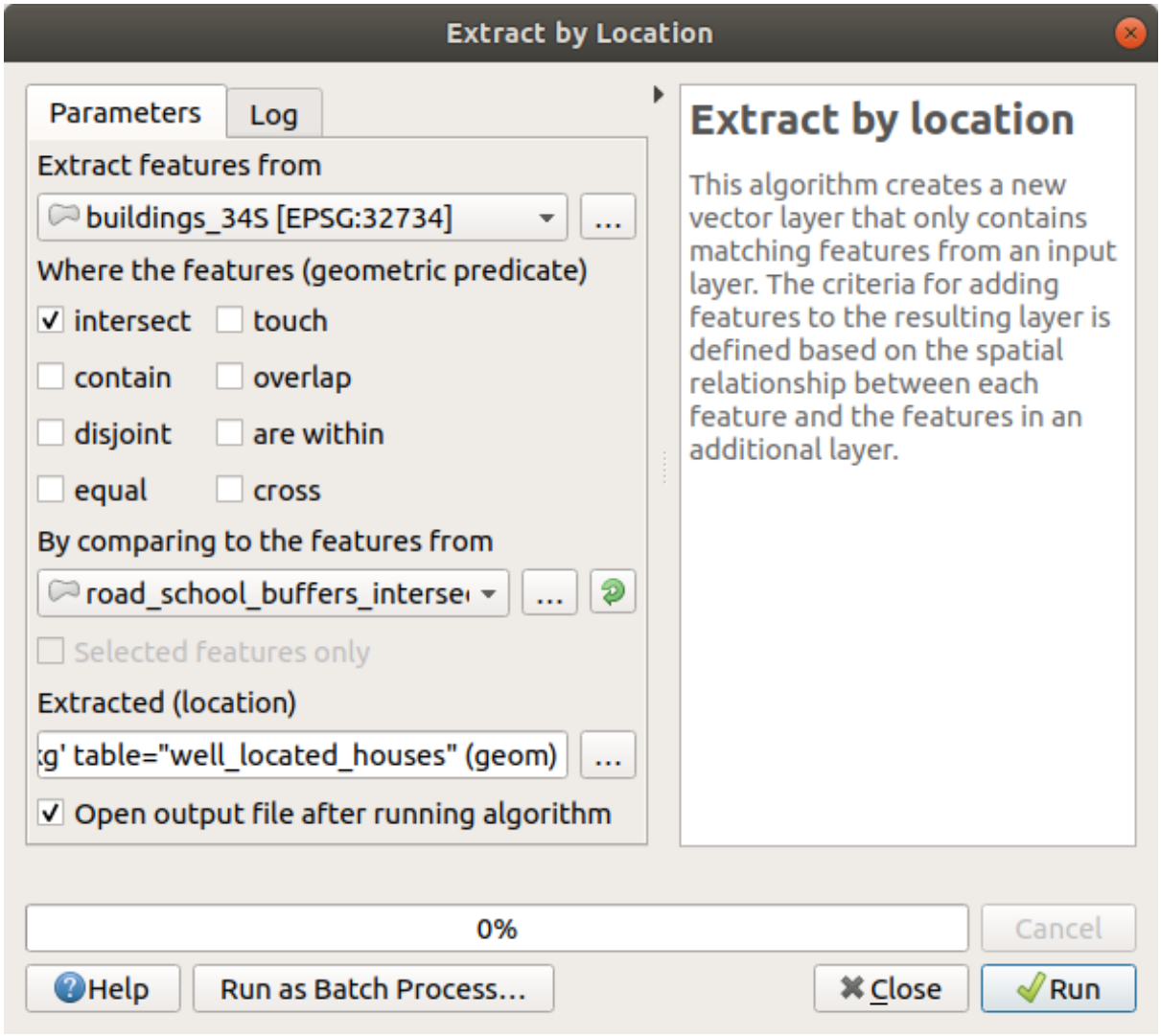
3. You may remove the two buffer layers and only keep the one that shows where they overlap, since that's what we really wanted to know in the first place:



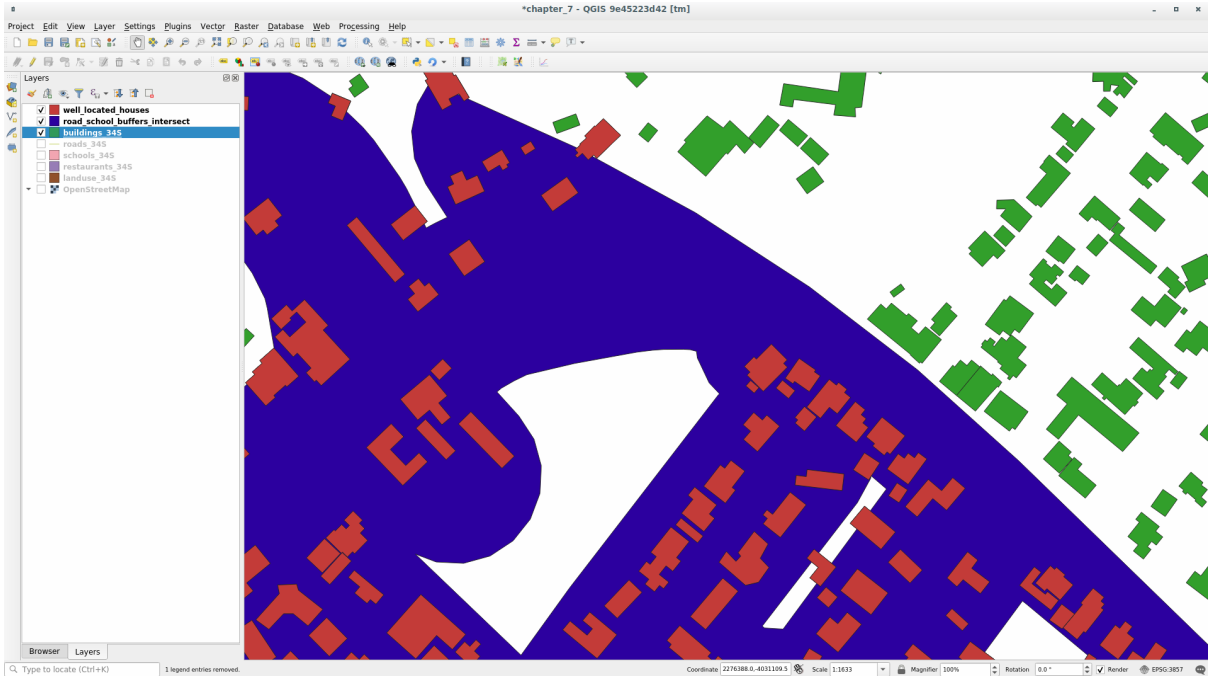
6.2.9 ??? Follow Along: Extract the Buildings

Now you've got the area that the buildings must overlap. Next, you want to extract the buildings in that area.

1. Look for the menu entry *Vector Selection* ► *Extract by location* within the *Processing Toolbox*
2. Select `buildings_34S` in *Extract features from*. Check *intersect* in *Where the features (geometric predicate)*, select the buffer intersection layer in *By comparing to the features from*. Save to the `vector_analysis.gpkg`, and name the layer `well_located_houses`.



3. Click *Run* and close the dialog
4. You will probably find that not much seems to have changed. If so, move the *well_located_houses* layer to the top of the layers list, then zoom in.



The red buildings are those which match our criteria, while the buildings in green are those which do not.

5. Now you have two separated layers and can remove `buildings_34S` from the layer list.

6.2.10 ??? Try Yourself: Further Filter our Buildings

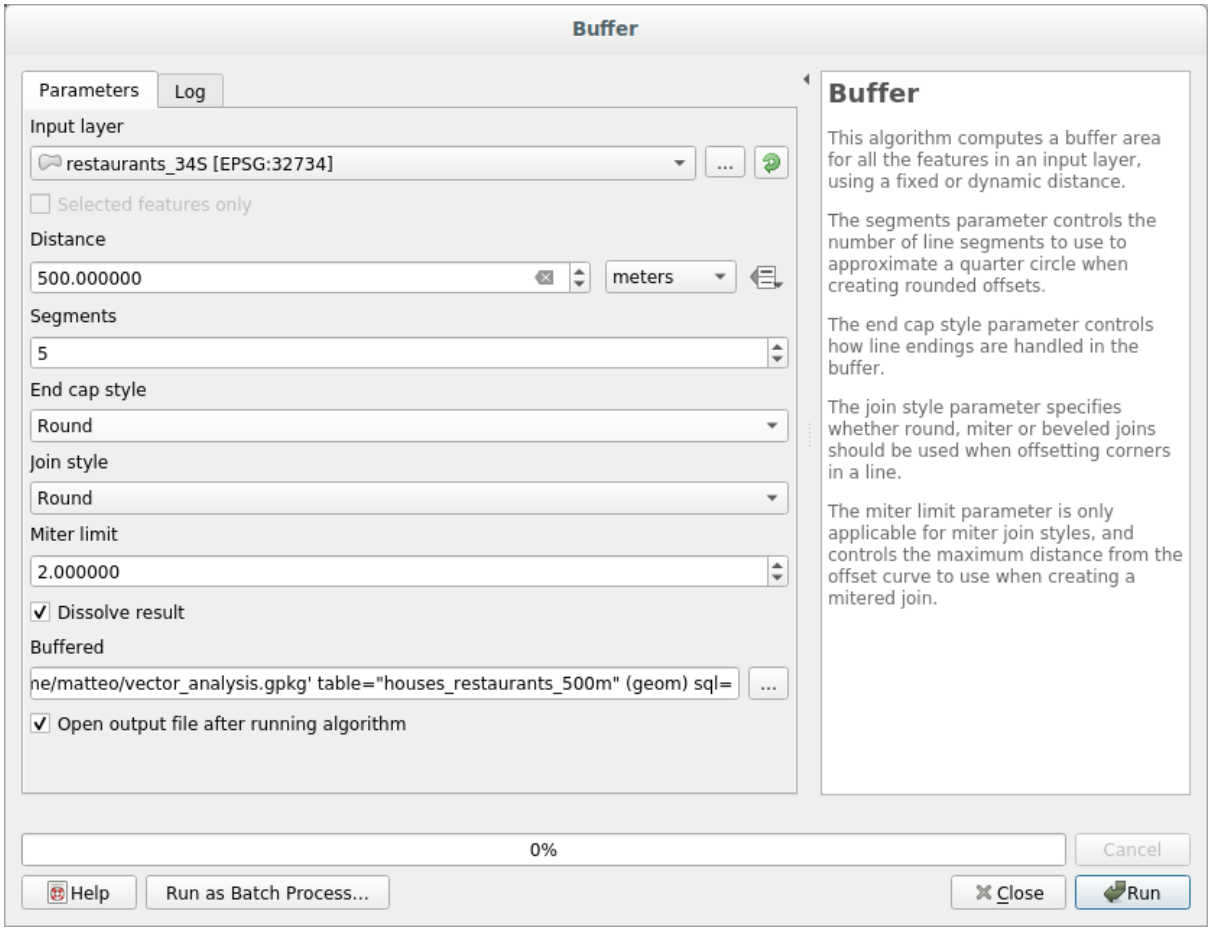
We now have a layer which shows us all the buildings within 1km of a school and within 50m of a road. We now need to reduce that selection to only show buildings which are within 500m of a restaurant.

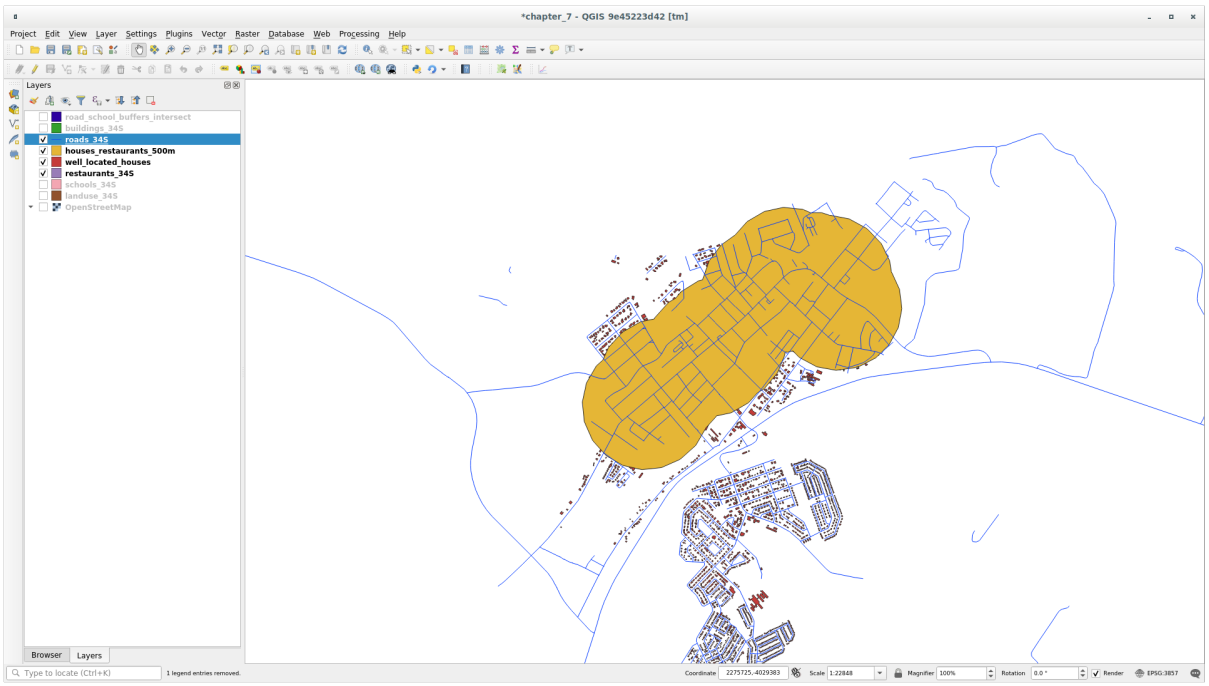
Using the processes described above, create a new layer called *houses_restaurants_500m* which further filters your *well_located_houses* layer to show only those which are within 500m of a restaurant.

Answer

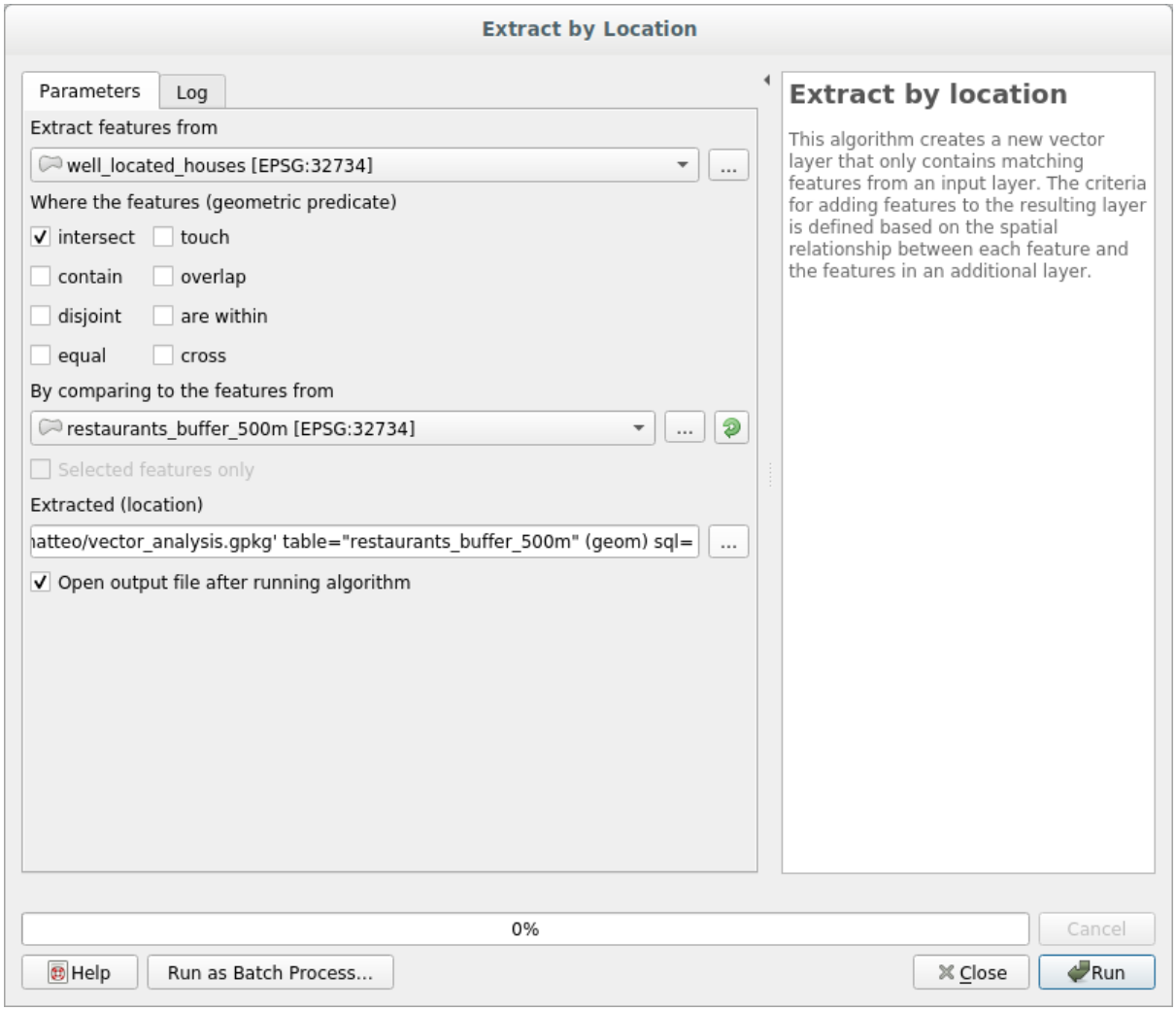
To create the new *houses_restaurants_500m* layer, we go through a two step process:

1. First, create a buffer of 500m around the restaurants and add the layer to the map:

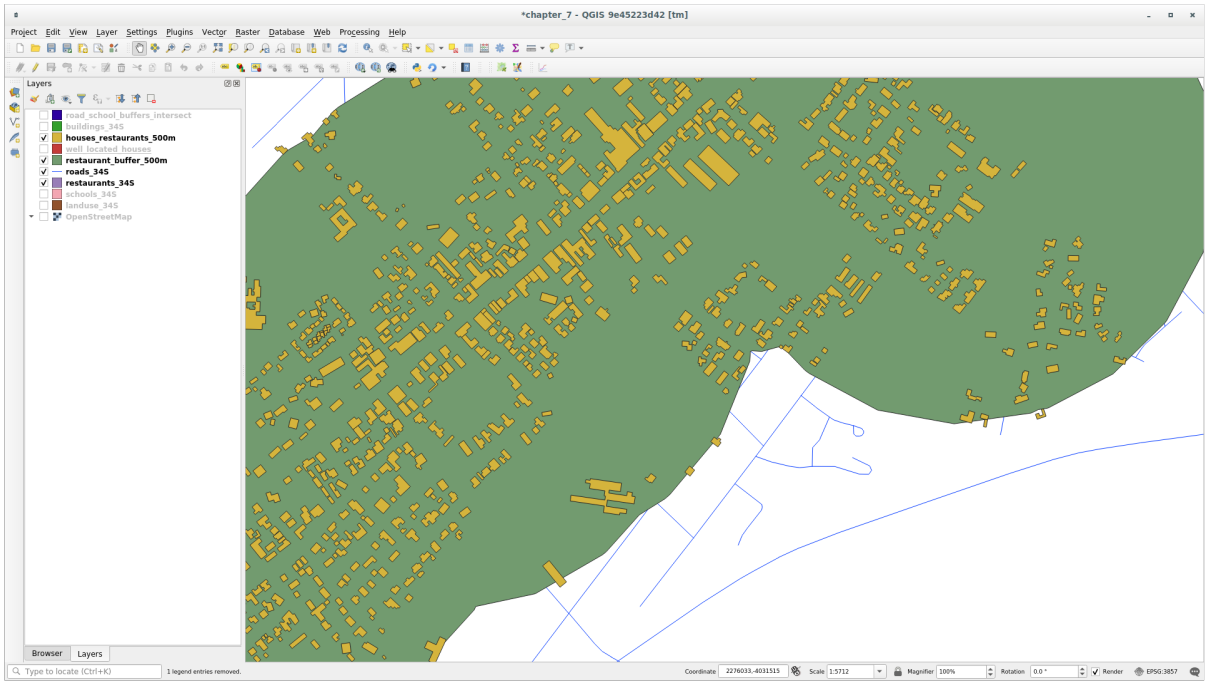




2. Next, extract buildings within that buffer area:



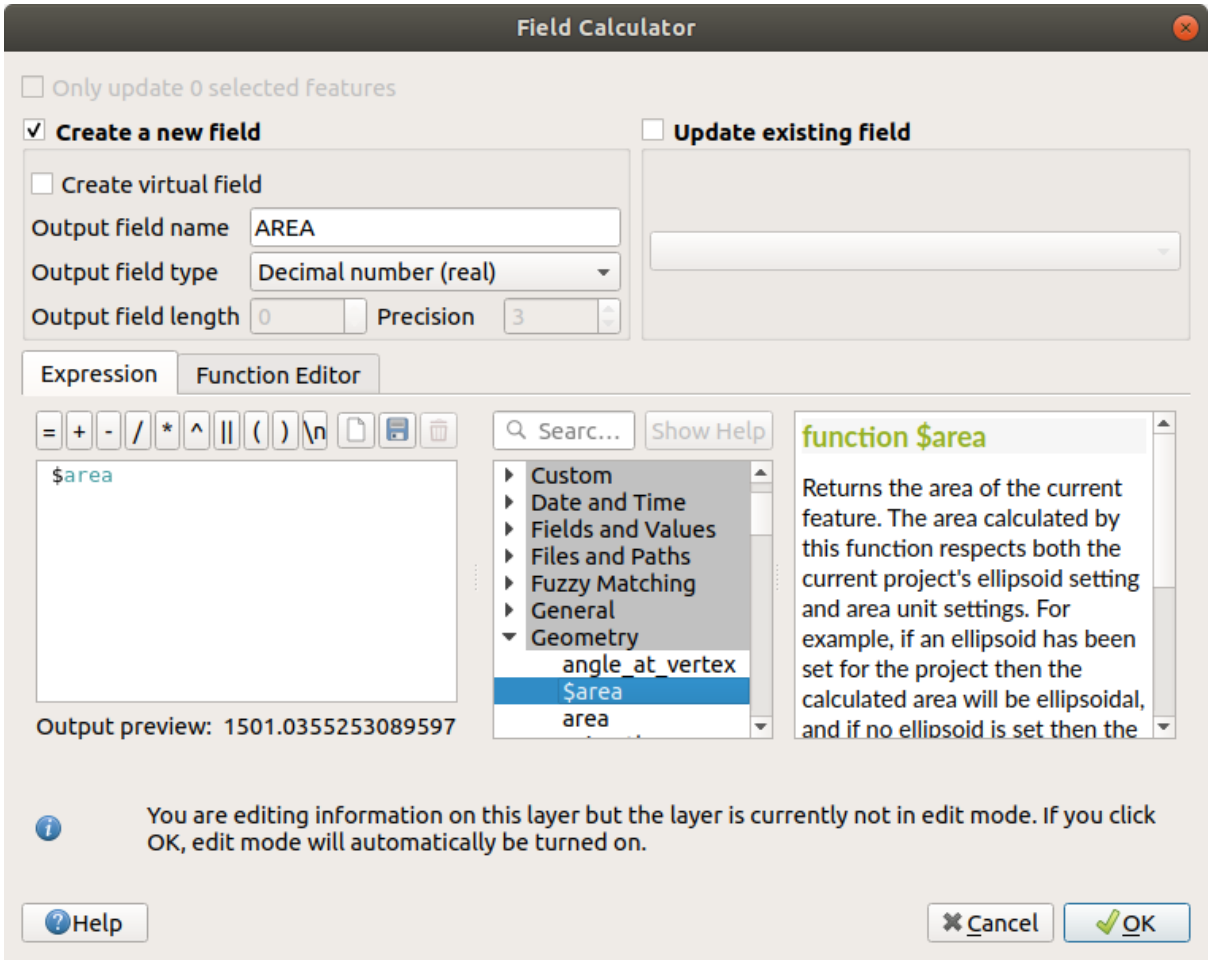
Your map should now show only those buildings which are within 50m of a road, 1km of a school and 500m of a restaurant:



6.2.11 Follow Along: Select Buildings of the Right Size

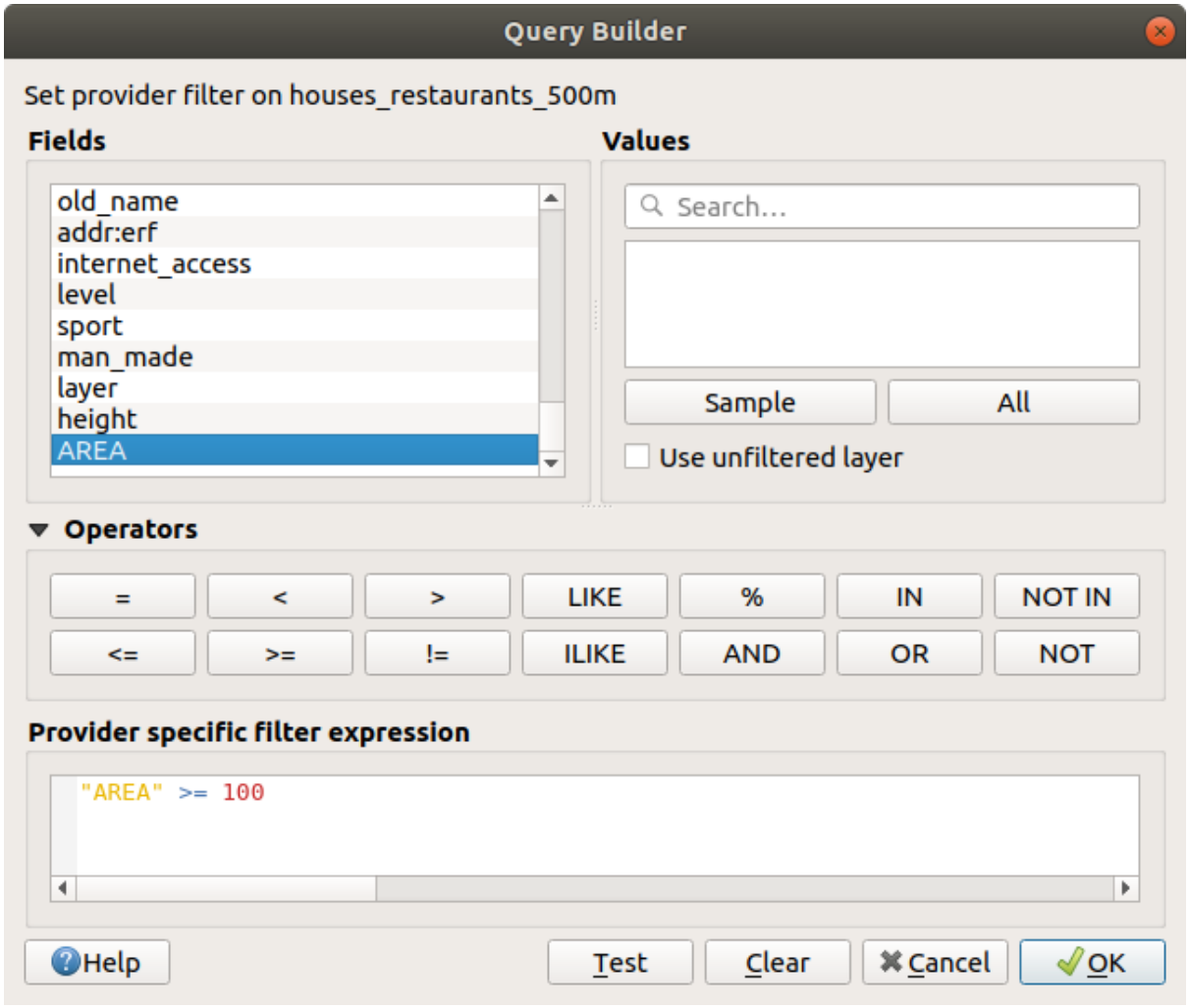
To see which buildings are of the correct size (more than 100 square meters), we need to calculate their size.

1. Select the *houses_restaurants_500m* layer and open the *Field Calculator* by clicking on the Open Field Calculator button in the main toolbar or in the attribute table window
2. Select *Create a new field*, set the *Output field name* to `AREA`, choose *Decimal number (real)* as *Output field type*, and choose `$area` from the *Geometry* group.



   The new field `AREA` will contain the area of each building in square meters.

3. Click *OK*. The `AREA` field has been added at the end of the attribute table.
4. Click the Toggle Editing button to finish editing, and save your edits when prompted.
5. In the *Source* tab of the layer properties, set the *Provider Feature Filter* to `"AREA >= 100`.



6. Click *OK*.

Your map should now only show you those buildings which match our starting criteria and which are more than 100 square meters in size.

6.2.12 ??? Try Yourself:

Save your solution as a new layer, using the approach you learned above for doing so. The file should be saved within the same GeoPackage database, with the name `solution`.

6.2.13 In Conclusion

Using the GIS problem solving approach together with QGIS vector analysis tools, you were able to solve a problem with multiple criteria quickly and easily.

6.2.14 What's Next?

In the next lesson, we will look at how to calculate the shortest distance along roads from one point to another.

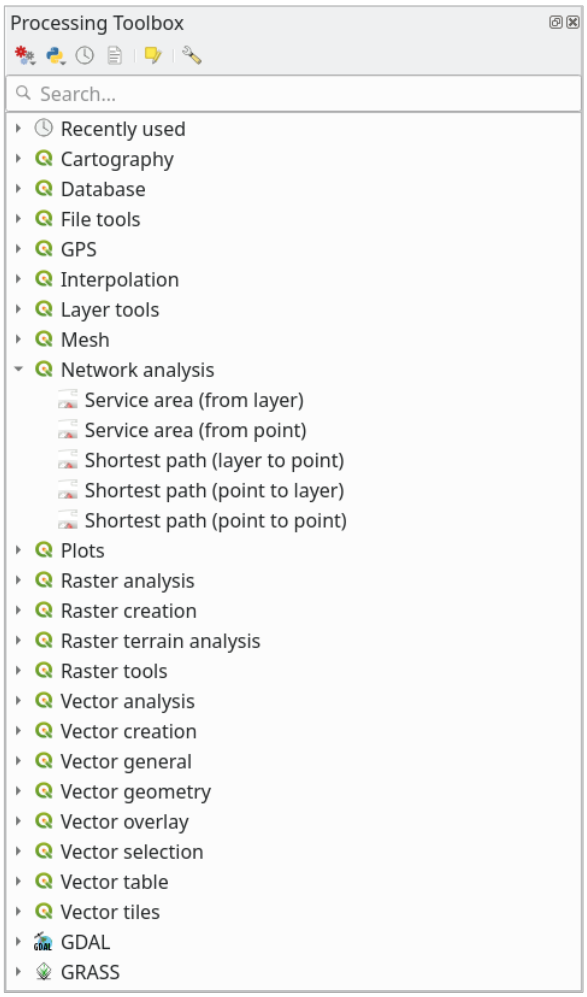
6.3 Lesson: Network Analysis

Calculating the shortest distance between two points is a common GIS task. Tools for this can be found in the *Processing Toolbox*.

Objectivo deste módulo: aprender a utilizar os algoritmos de *Análise de Rede*.

6.3.1 ??? Follow Along: The Tools and the Data

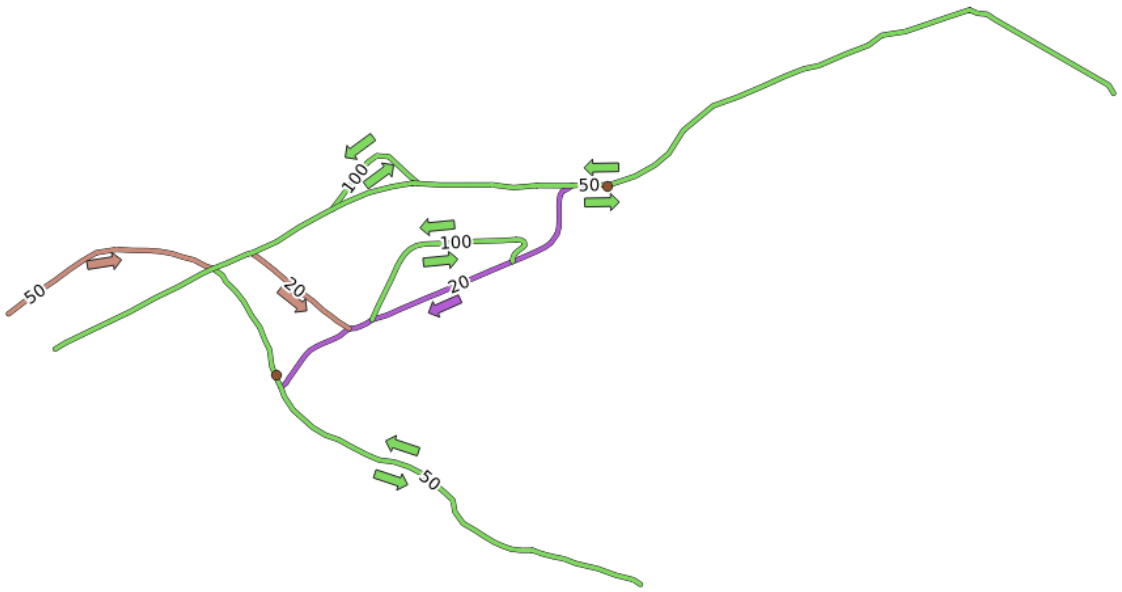
Pode encontrar todos os algoritmos de análise de rede no menu *Processamento* ► *Análise de rede*. Existem várias ferramentas disponíveis:



Open the project `exercise_data/network_analysis/network.qgz`. It contains two layers:

- `network_points`
- `network_lines`

The *network_lines* layer has already a style that helps to understand the road network.



Os algoritmos de caminho mais curto disponibilizam formas de calcular tanto o caminho mais curto, como o caminho mais rápido, entre dois pontos da rede, dados:

- start and end points selected on the map
- um ponto de partida seleccionado no mapa e vários pontos de chegada seleccionados numa camada de pontos
- vários pontos de partida seleccionados numa camada de pontos e um ponto de chegada seleccionado no mapa

Mãos à obra.

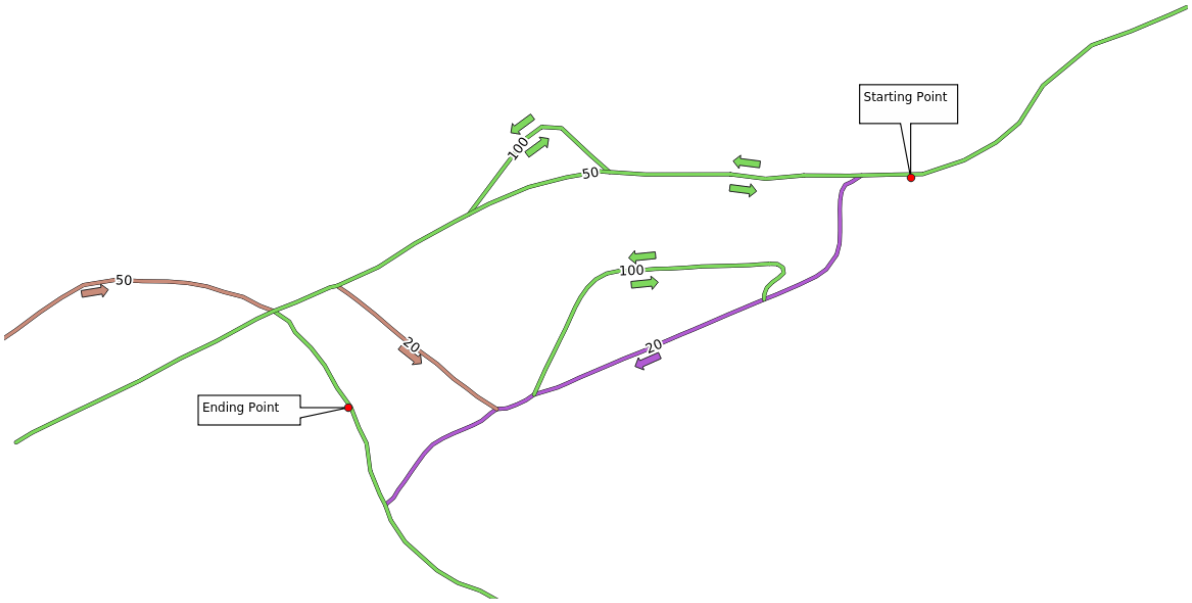
6.3.2 ??? Calculate the shortest path (point to point)

Análise da rede ► *Caminho mais curto (ponto para ponto)* permite-lhe calcular o percurso mais curto entre dois pontos seleccionados manualmente no mapa.

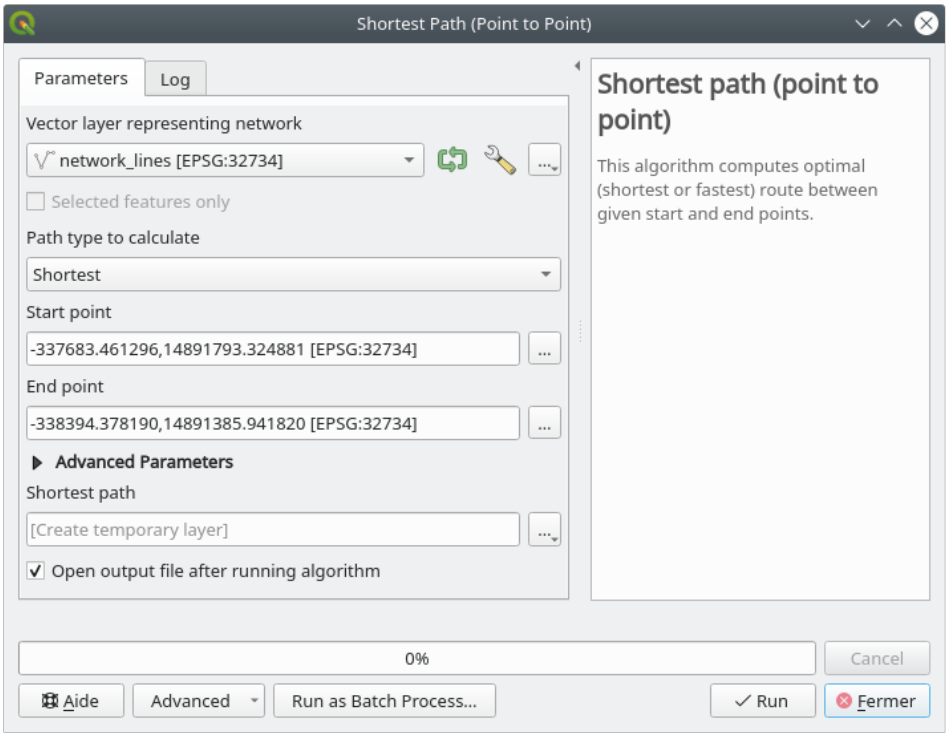
Neste exemplo vamos calcular o caminho mais **curto** (e não mais rápido) entre dois pontos.

1. Escolha o algoritmo *Caminho mais curto (ponto para ponto)*
2. Em *Camada vectorial representando a rede* seleccione *network_lines*
3. Use `Shortest` for *Path type to calculate*

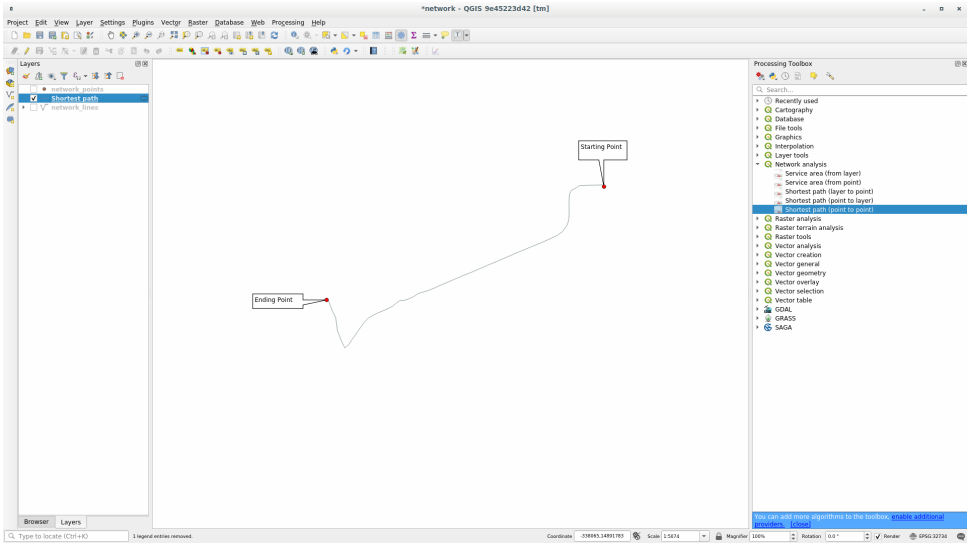
   Use these two points as starting and ending points for the analysis:



4. Click on the … button next to *Start point (x, y)* and choose the location tagged with `Starting Point` in the picture. Enable the snapping options for an accurate selection. The coordinates of the clicked point are added.
5. Do the same thing, but choosing the location tagged with `Ending point` for *End point (x, y)*
6. Clique no botão *Executar*:



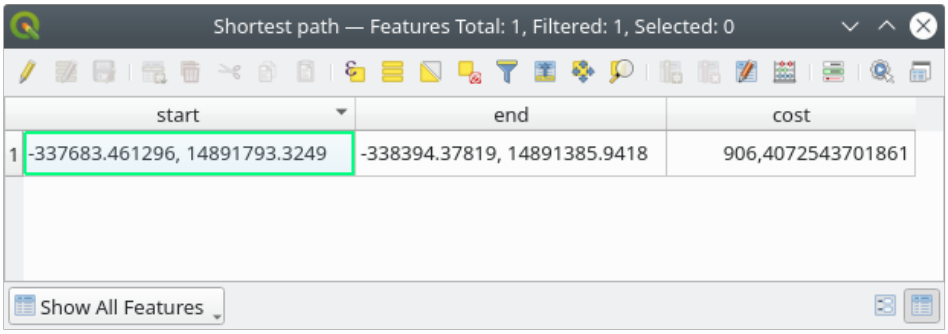
7. A new line layer is created representing the shortest path between the chosen points. Uncheck the `network_lines` layer to see the result better:



8. Open the attribute table of the output layer. It contains three fields, representing the coordinates of the start and end points and the **cost**.

   Ao escolher para *Tipo de caminho a calcular*, o parâmetro `Mais curto`, o campo **cost** irá conter a **distância** entre os dois pontos, nas unidades definidas para a camada.

   Neste caso, a distância *mais curta* entre os pontos escolhidos é aproximadamente `1000` metros:



Now that you know how to use the tool, feel free to test other locations.

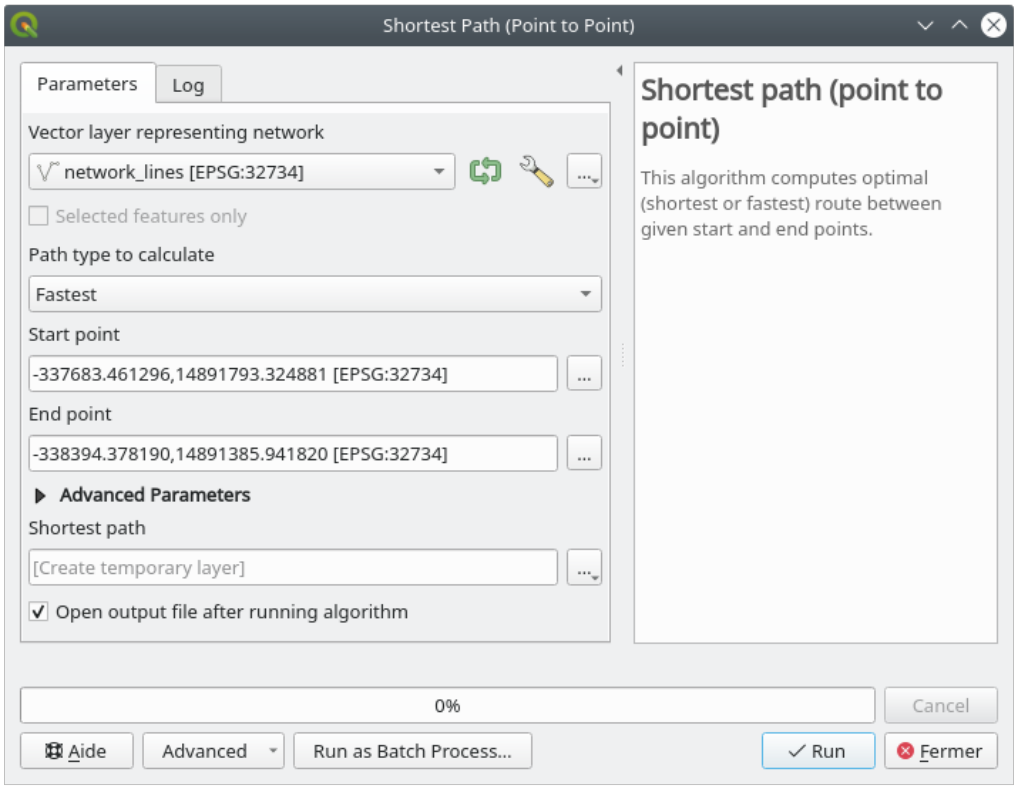
6.3.3 ??? Try Yourself: Fastest path

Com os mesmos dados do exercício anterior, tente calcular o caminho mais rápido entre os dois pontos.

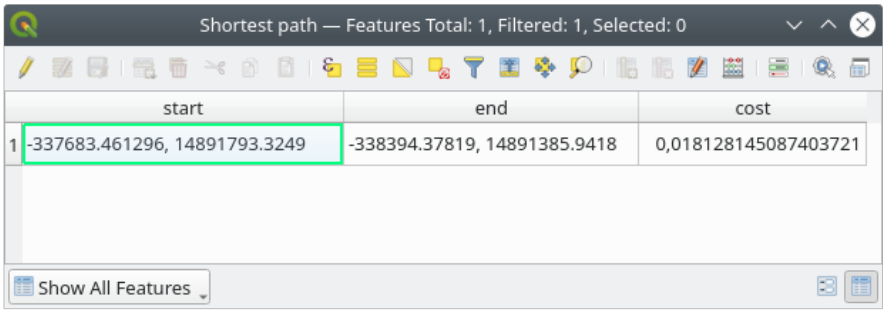
De quanto tempo necessita para ir do ponto de partida até ao ponto de chegada?

Answer

1. Open *Network Analysis* ► *Shortest Path (Point to Point)* and fill the dialog as:



2. Make sure that the *Path type to calculate* is `Fastest`.
3. Click on *Run* and close the dialog.
4. Open now the attribute table of the output layer. The *cost* field contains the travel time between the two points (as fraction of hours):



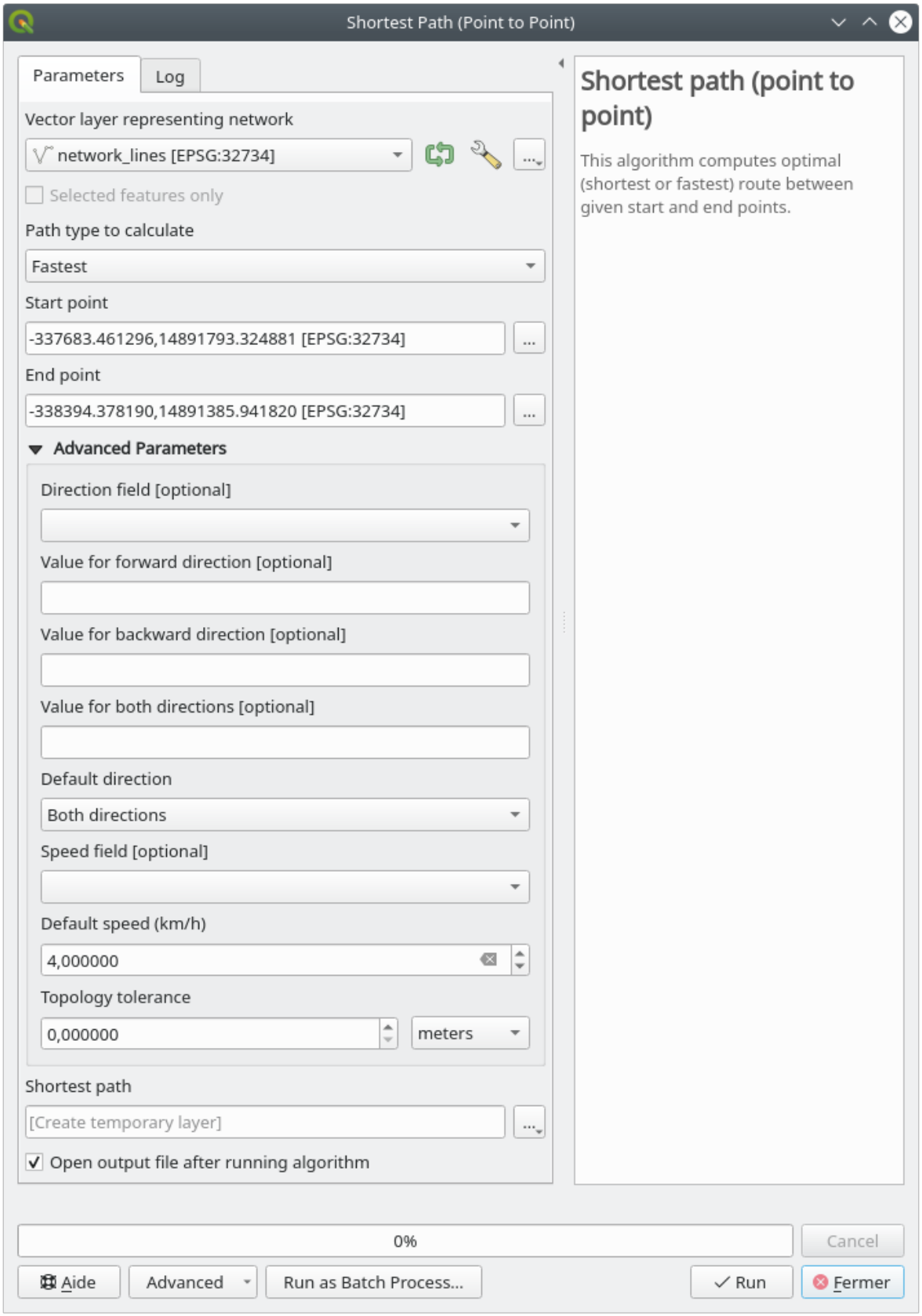
6.3.4 ??? Follow Along: Advanced options

Let us explore some more options of the Network Analysis tools. In the *previous exercise* we calculated the **fastest** route between two points. As you can imagine, the time depends on the travel **speed**.

We will use the same layers and starting and ending points of the previous exercises.

1. Escolha o algoritmo *Caminho mais curto (ponto para ponto)*
2. Preencha *Camada vectorial representando a rede*, *Ponto de partida (x, y)* e *Ponto de chegada (x, y)* conforme fez anteriormente.

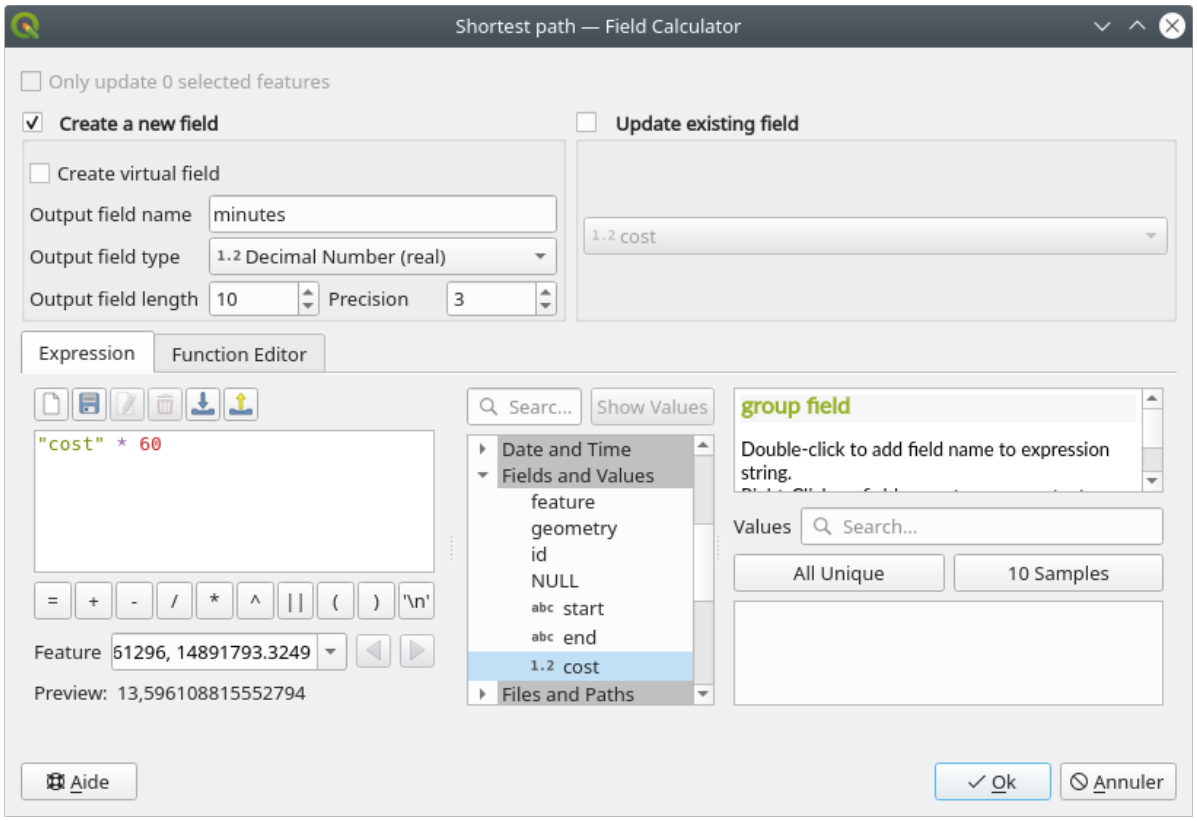
3. Em *Tipo de caminho a calcular* escolha `Mais rápido`
4. Abra o menu *Parâmetros avançados*
5. Em *Velocidade padrão (km/h)*, altere o valor predefinido (`50`) para `4`



6. Clique em *Executar*
7. Uma vez executado o algoritmo, feche a caixa de diálogo e abra a tabela de atributos da camada resultante.

The *cost* field contains the value according to the speed parameter you have chosen. We can convert the *cost* field from hours with fractions to the more readable *minutes* values.

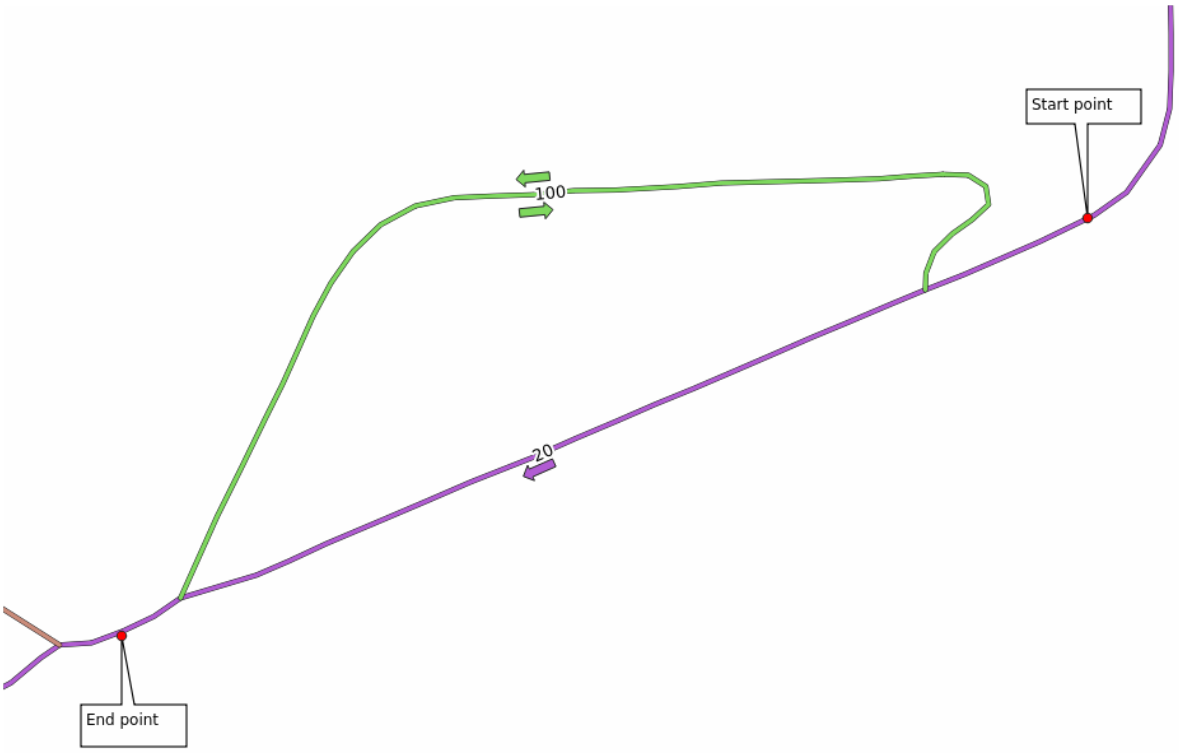
8. Open the field calculator by clicking on the icon
9. Add a new field to store the path cost in *minutes*.



That's it! Now you know how many minutes it will take to get from one point to the other one if the whole network speed is at `4 km/h`.

6.3.5 ??? Shortest path with speed limit

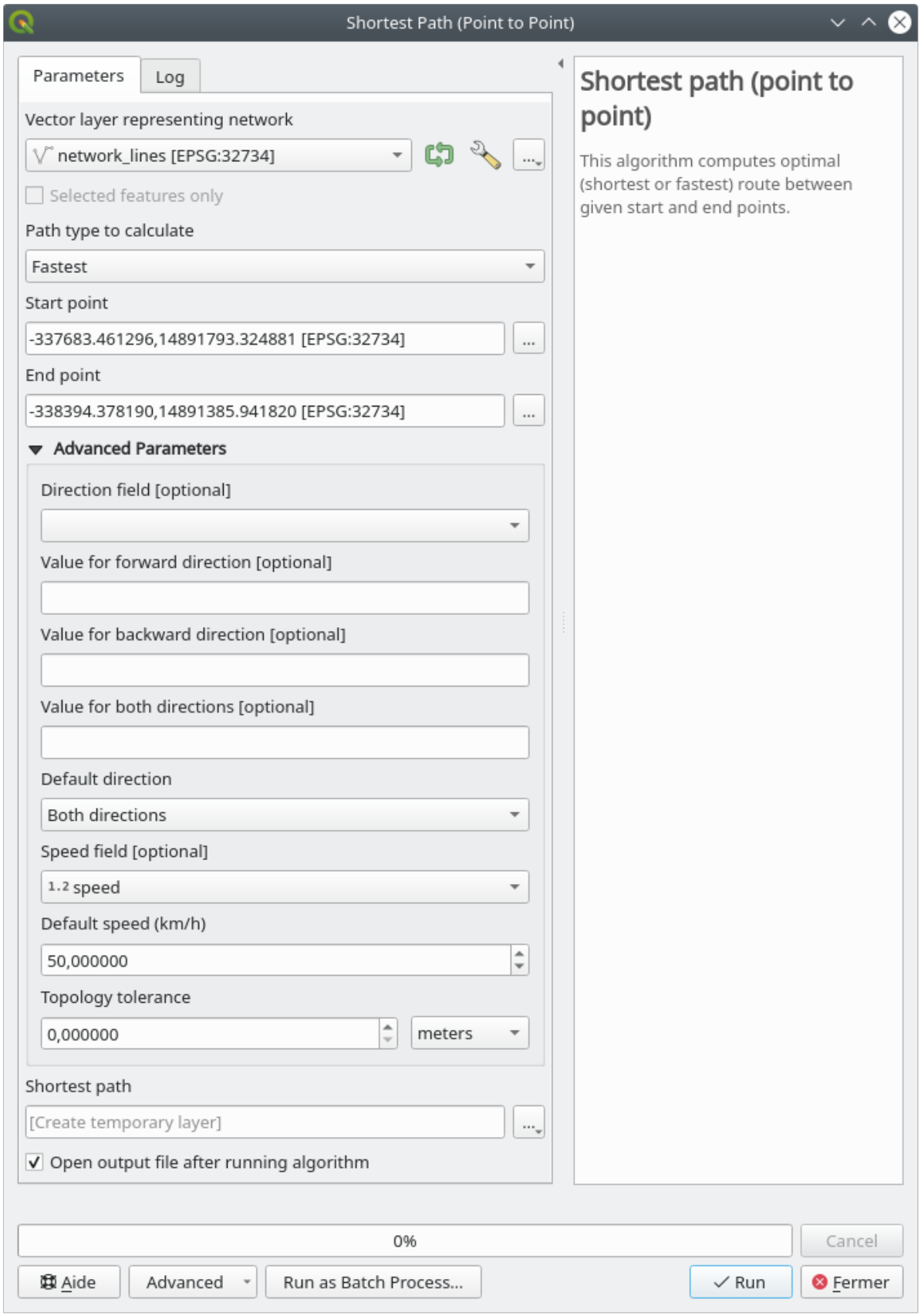
A ferramenta de processamento Análise de rede, possui outras opções interessantes. Olhando para o seguinte mapa:



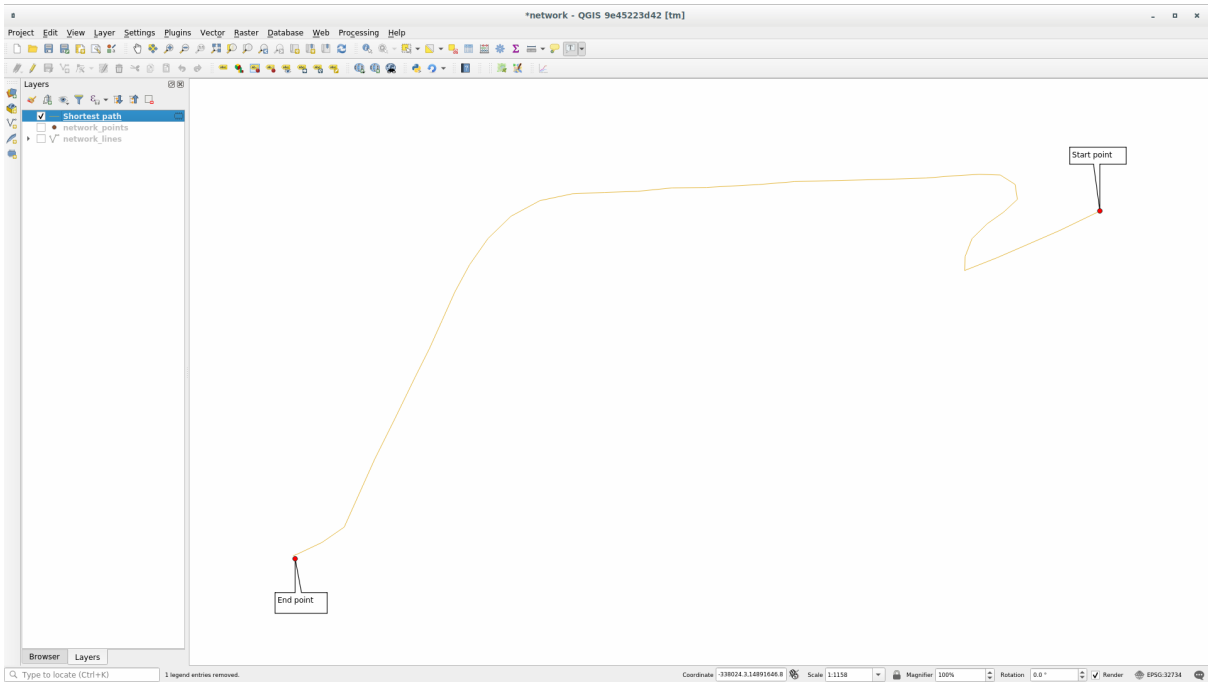
gostaríamos de determinar o percurso **mais rápido**, tendo em consideração os **limites de velocidade** de cada estrada (as etiquetas indicam os limites, em km/h). Sem considerar esses limites, o caminho **mais curto** seria, como é óbvio, o de cor púrpura. Mas nessa estrada o limite de velocidade é de 20 km/h, enquanto que na estrada verde se pode circular a 100 km/h!

Assim como fizemos no primeiro exercício, vamos seleccionar *Análise de rede* ► *Caminho mais curto (ponto a ponto)* e capturar os pontos de partida e chegada.

1. Abra o algoritmo *Análise de rede* ► *Caminho mais curto (ponto a ponto)*
2. Seleccione *network_lines* como parâmetro em *Camada vectorial representando a rede*
3. Em *Tipo de caminho a calcular* escolha `Mais rápido`
4. Select the *Start point (x, y)* and *End point (x, y)* as we did before
5. Abra o menu *Parâmetros avançados*
6. Choose the *speed* field as the *Speed Field* parameter. With this option the algorithm will take into account the speed limits for each road.



7. Clique no botão *Executar*
8. Turn off the `network_lines` layer to better see the result



Como pode verificar, o caminho mais rápido não corresponde ao caminho mais curto.

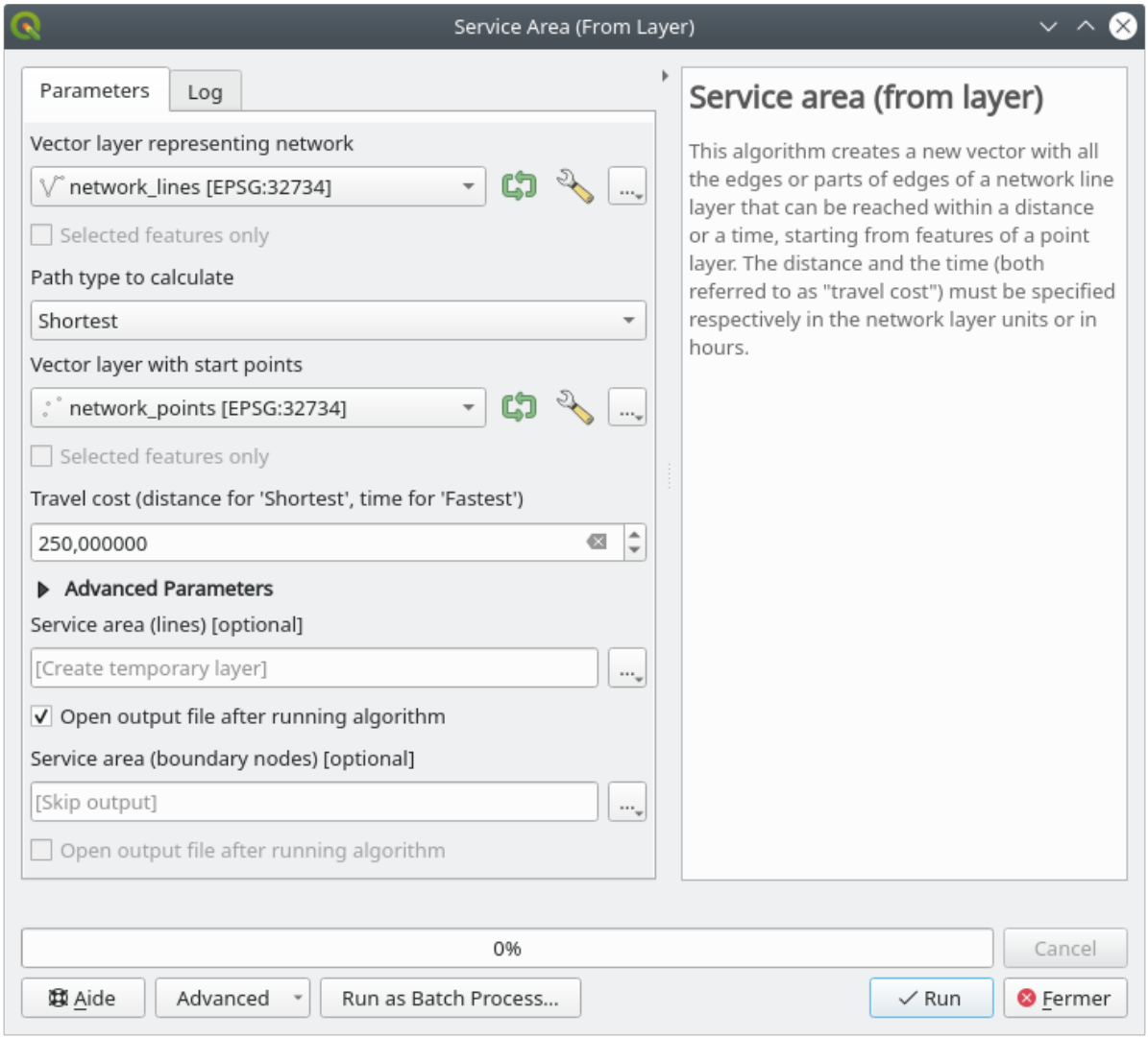
6.3.6 ??? Service area (from layer)

O algoritmo *Análise de rede* ► *Área de serviço (da camada)* oferece resposta à questão: dada uma camada de pontos, quais são todas as áreas que se podem alcançar, dentro de uma determinada distância ou de um determinado tempo de percurso?

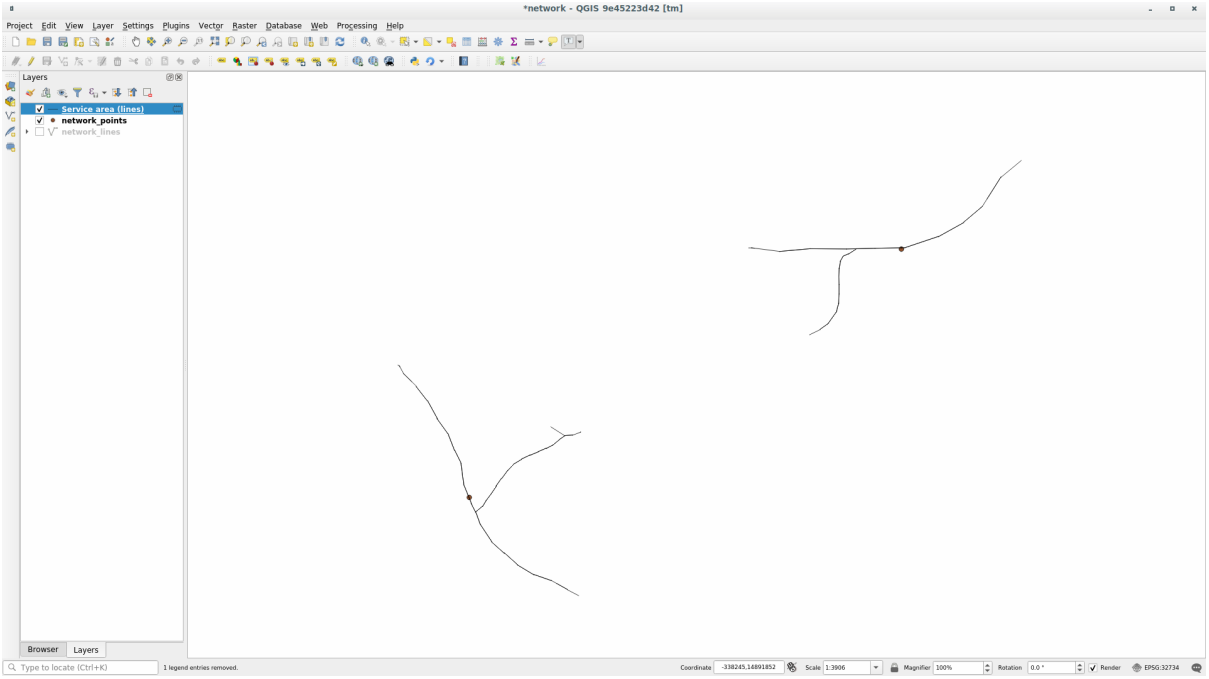
Nota: The *Network Analysis* ► *Service area (from point)* is the same algorithm, but it allows you to manually choose the point on the map.

Dada uma distância de `250` metros, queremos saber, para cada ponto da camada *network_points*, quão longe nos é possível deslocar-nos através da rede.

1. Uncheck all the layers except `network_points`
2. Abra o algoritmo *Análise de rede* ► *Área de serviço (da camada)*
3. Choose `network_lines` for *Vector layer representing network*
4. Choose `network_points` for *Vector layer with start points*
5. Em *Tipo de caminho a calcular* escolha `Mais curto`
6. Enter `250` for the *Travel cost* parameter
7. Click on *Run* and close the dialog



A camada resultante apresenta, para cada ponto, os percursos que se podem efectuar, sem exceder a distância máxima de 250 metros:



Porreiro, hã?

6.3.7 In Conclusion

Now you know how to use *Network analysis* algorithm to solve shortest and fastest path problems.

E está pronto para fazer um pouco de análise e estatística espacial em camadas de dados vectoriais. Vamos lá!

6.3.8 What's Next?

A seguir verá como executar algoritmos de estatística espacial sobre dados vectoriais.

6.4 Lesson: Spatial Statistics

Nota: Lesson developed by Linfiniti and S Motala (Cape Peninsula University of Technology)

Spatial statistics allows you to analyze and understand what is going on in a given vector dataset. QGIS includes many useful tools for statistical analysis.

The goal for this lesson: To know how to use QGIS" spatial statistics tools within the *Processing Toolbox*.

6.4.1 ??? Follow Along: Create a Test Dataset

We will create a random set of points, to get a dataset to work with.

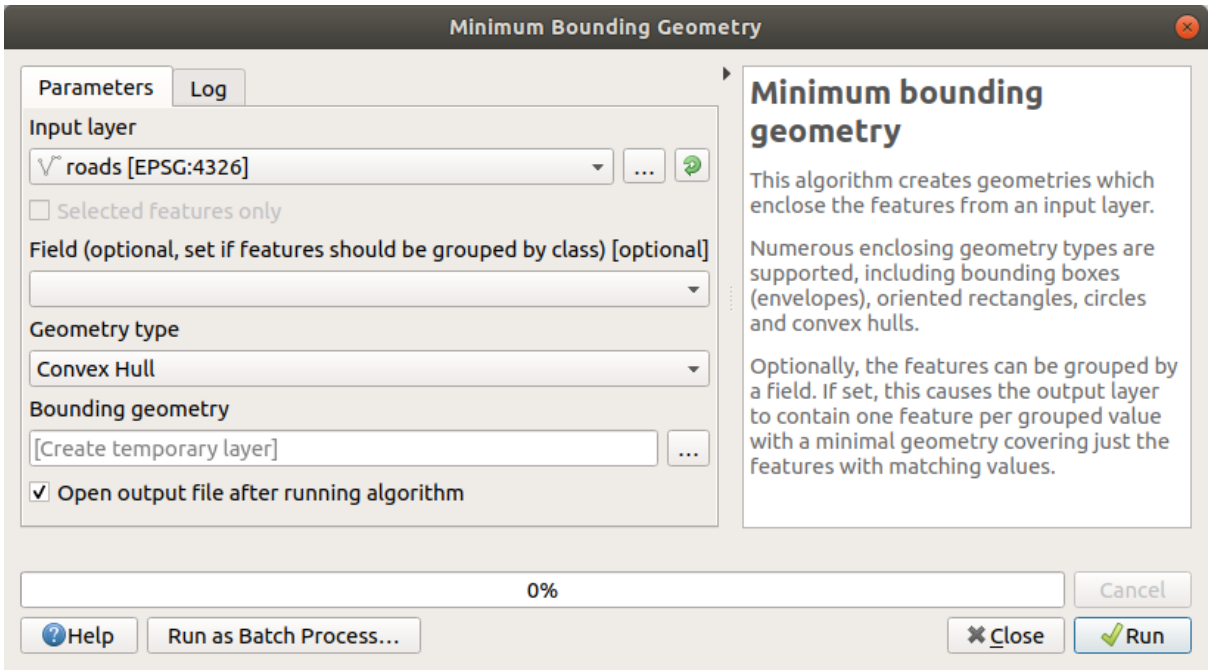
To do so, you will need a polygon dataset to define the area you want to create the points in.

We will use the area covered by streets.

1. Start a new project
2. Add your `roads` dataset, as well as `srtm_41_19` (elevation data) found in `exercise_data/raster/SRTM/`.

   Nota: You might find that the SRTM DEM layer has a different CRS to that of the roads layer. QGIS is reprojecting both layers in a single CRS. For the following exercises this difference does not matter, but feel free to reproject (as shown earlier in this module).

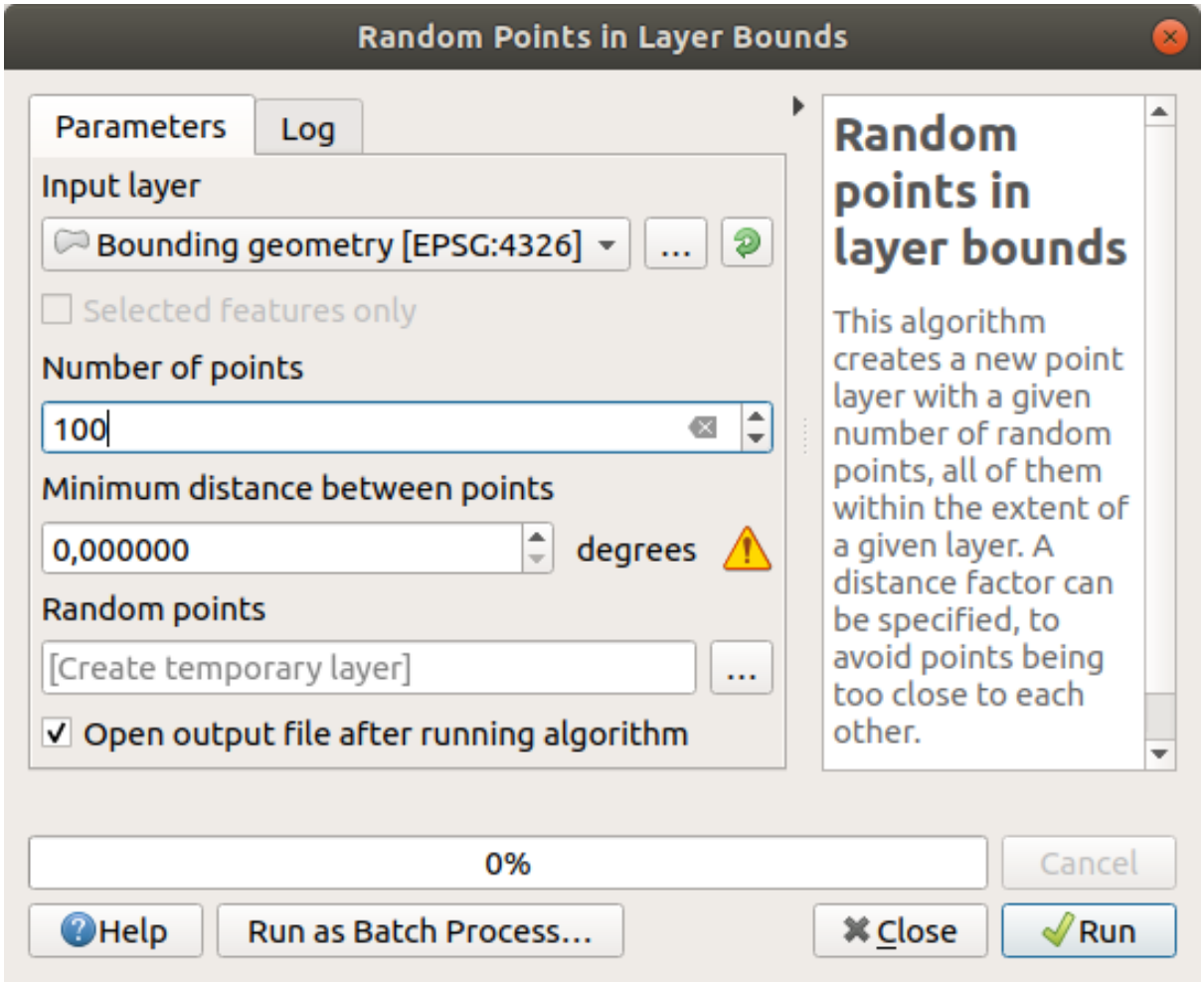
3. Open *Processing* toolbox
4. Use the *Vector Geometry* ► *Minimum bounding geometry* tool to generate an area enclosing all the roads by selecting `Convex Hull` as the *Geometry Type*:



   As you know, if you don't specify the output, *Processing* creates temporary layers. It is up to you to save the layers immediately or at a later stage.

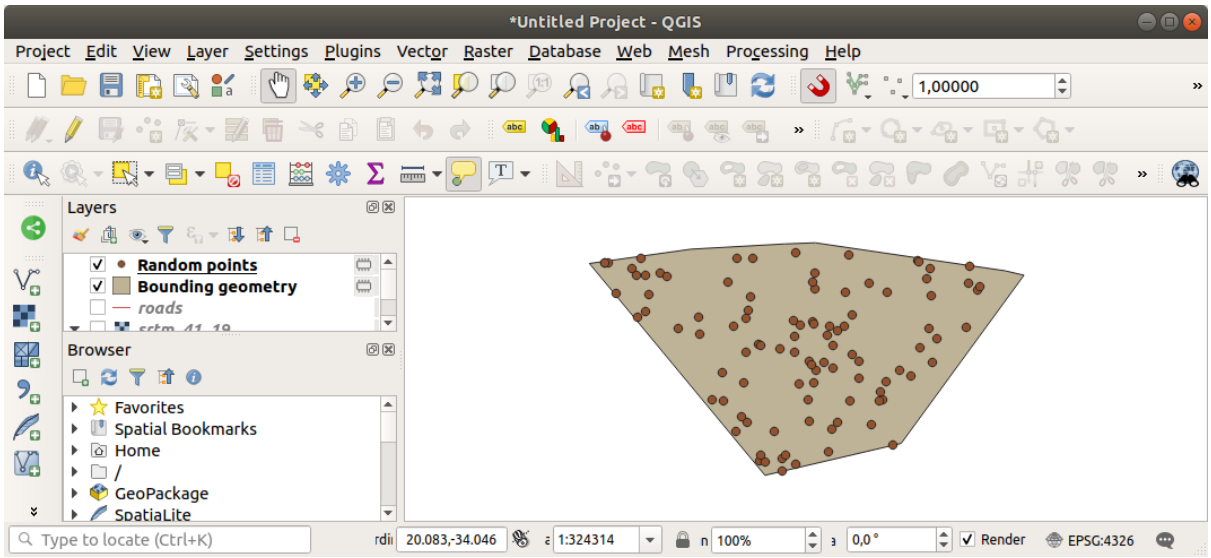
Creating random points

- Create 100 random points in this area using the tool at *Vector Creation* ► *Random points in layer bounds*, with a minimum distance of `0.0`:



Nota: The yellow warning sign tells you that that parameter concerns distances. The *Bounding geometry* layer is in a Geographical Coordinate System and the algorithm is just reminding you this. For this example we won't use this parameter so you can ignore it.

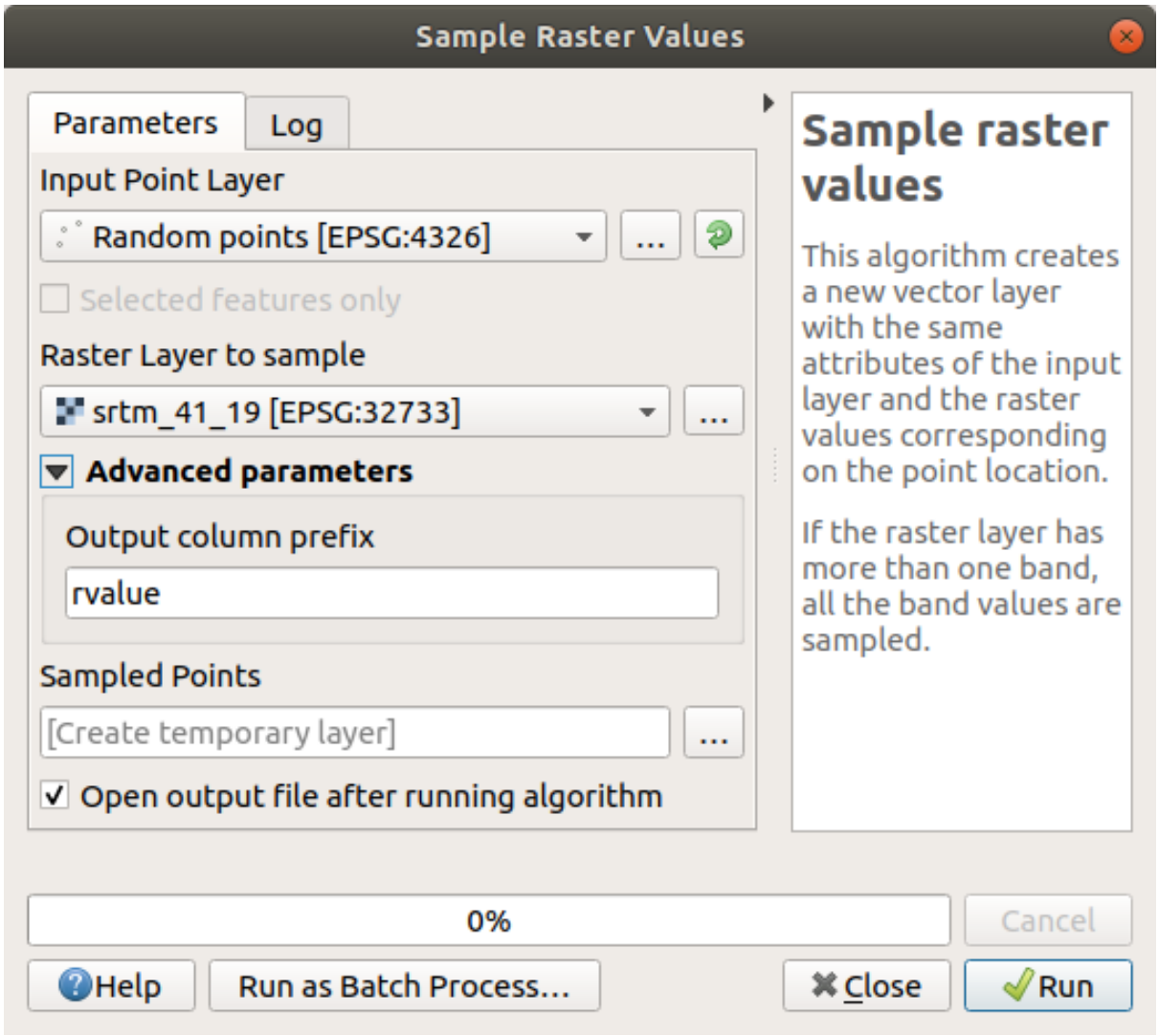
If needed, move the generated random point to the top of the legend to see them better:



Sampling the data

To create a sample dataset from the raster, you'll need to use the *Raster Analysis* ► *Sample raster values* algorithm. This tool samples the raster at the locations of the points and adds the raster values in new field(s) depending on the number of bands in the raster.

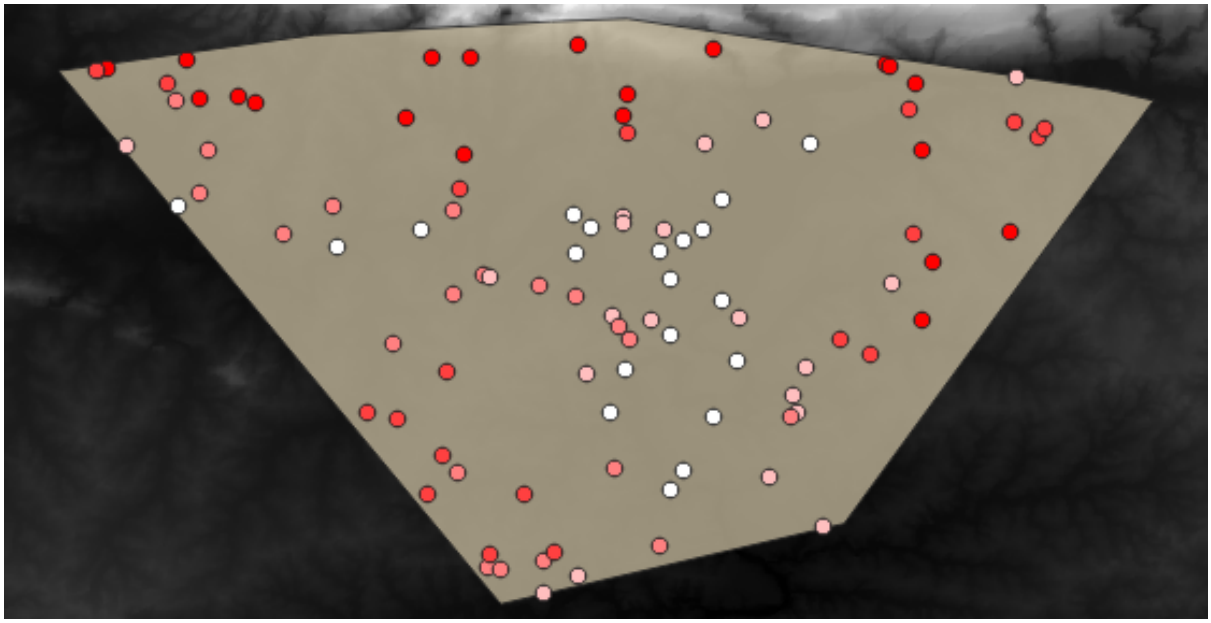
1. Open the *Sample raster values* algorithm dialog
2. Select `Random_points` as the layer containing sampling points, and the SRTM raster as the band to get values from. The default name of the new field is `rvalue_N`, where `N` is the number of the raster band. You can change the name of the prefix if you want.



3. Press *Run*

Now you can check the sampled data from the raster file in the attribute table of the `Sampled Points` layer. They will be in a new field with the name you have chosen.

A possible sample layer is shown here:



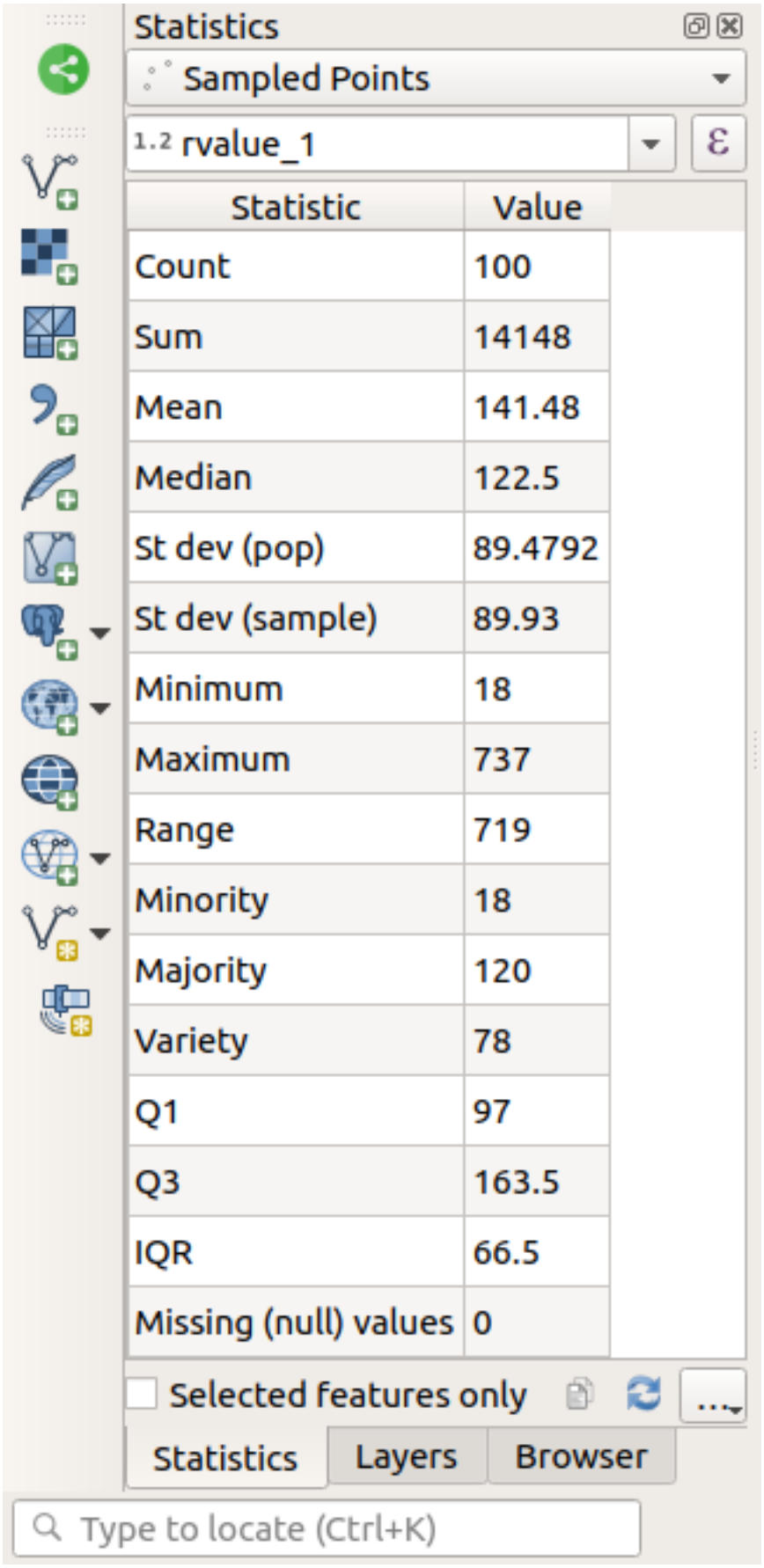
The sample points are classified using the `rvalue_1` field such that red points are at a higher altitude.

You will be using this sample layer for the rest of the statistical exercises.

6.4.2 ??? Follow Along: Basic Statistics

Now get the basic statistics for this layer.

1. Click on the Σ Show statistical summary icon in the *Attributes Toolbar*. A new panel will pop up.
2. In the dialog that appears, specify the `Sampled Points` layer as the source.
3. Select the *rvalue_1* field in the field combo box. This is the field you will calculate statistics for.
4. The *Statistics* Panel will be automatically updated with the calculated statistics:


Statistics
Sampled Points
1.2 rvalue_1
Statistic
Value
Count
100
Sum
14148
Mean
141.48
Median
122.5
St dev (pop)
89.4792
St dev (sample)
89.93
Minimum
18
Maximum
737
Range
719
Minority
18
Majority
120
Variety
78
Q1
97
Q3
163.5
IQR
66.5
Missing (null) values
0
Selected features only
Statistics
Layers
Browser
Type to locate (Ctrl+K)

Nota: You can copy the values by clicking on the Copy Statistics To Clipboard button and paste the results into a spreadsheet.

5. Close the *Statistics* Panel when done

Many different statistics are available:

Count
The number of samples/values.

Sum
The values added together.

Mean
The mean (average) value is simply the sum of the values divided by the number of values.

Median
If you arrange all the values from smallest to greatest, the middle value (or the average of the two middle values, if N is an even number) is the median of the values.

St Dev (pop)
The standard deviation. Gives an indication of how closely the values are clustered around the mean. The smaller the standard deviation, the closer values tend to be to the mean.

Minimum
The minimum value.

Maximum
The maximum value.

Range
The difference between the minimum and maximum values.

Q1
First quartile of the data.

Q3
Third quartile of the data.

Missing (null) values
The number of missing values.

6.4.3 ??? Follow Along: Compute statistics on distances between points

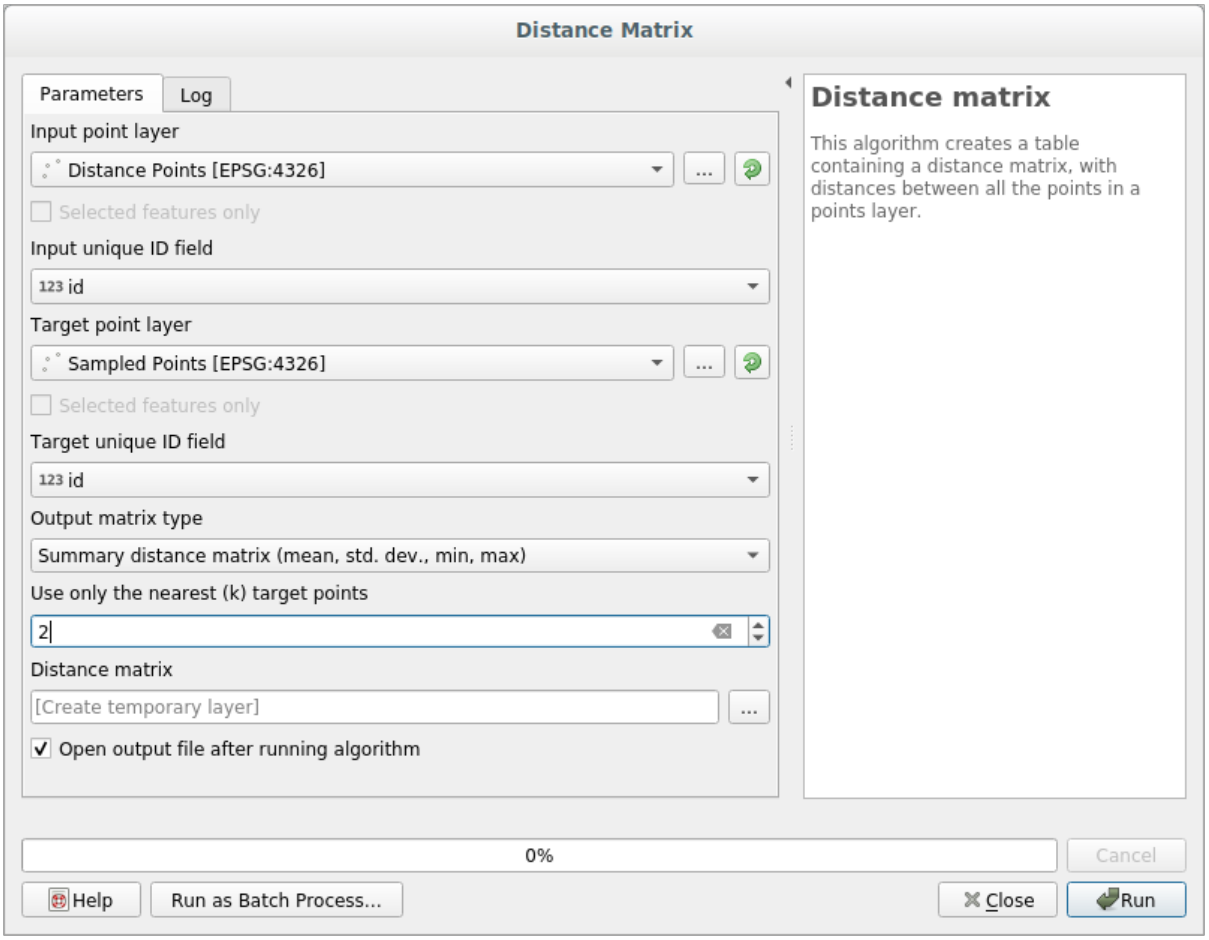
1. Create a new temporary point layer.
2. Enter edit mode, and digitize three points somewhere among the other points.

   Alternatively, use the same random point generation method as before, but specify only **three** points.
3. Save your new layer as *distance_points* in the format you prefer.

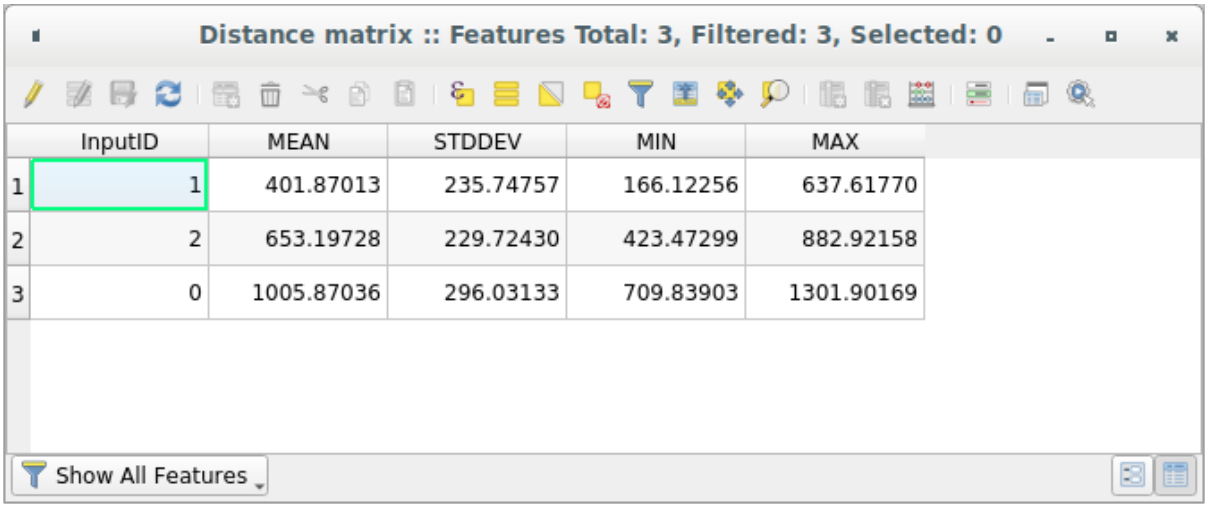
To generate statistics on the distances between points in the two layers:

1. Open the *Vector Analysis* ► *Distance matrix* tool.
2. Select the `distance_points` layer as the input layer, and the `Sampled Points` layer as the target layer.
3. Set their `id` field as unique field references
4. Change the *Output matrix type* option into *Summary distance matrix*.
5. set value of *Use only the nearest (k) target points* to `2`.

6. If you want you can save the output layer as a file or just run the algorithm and save the temporary output layer later.



7. Click *Run* to generate the distance matrix layer.
8. Open the attribute table of the generated layer: values refer to the distances between the *distance_points* features and their two nearest points in the *Sampled Points* layer:



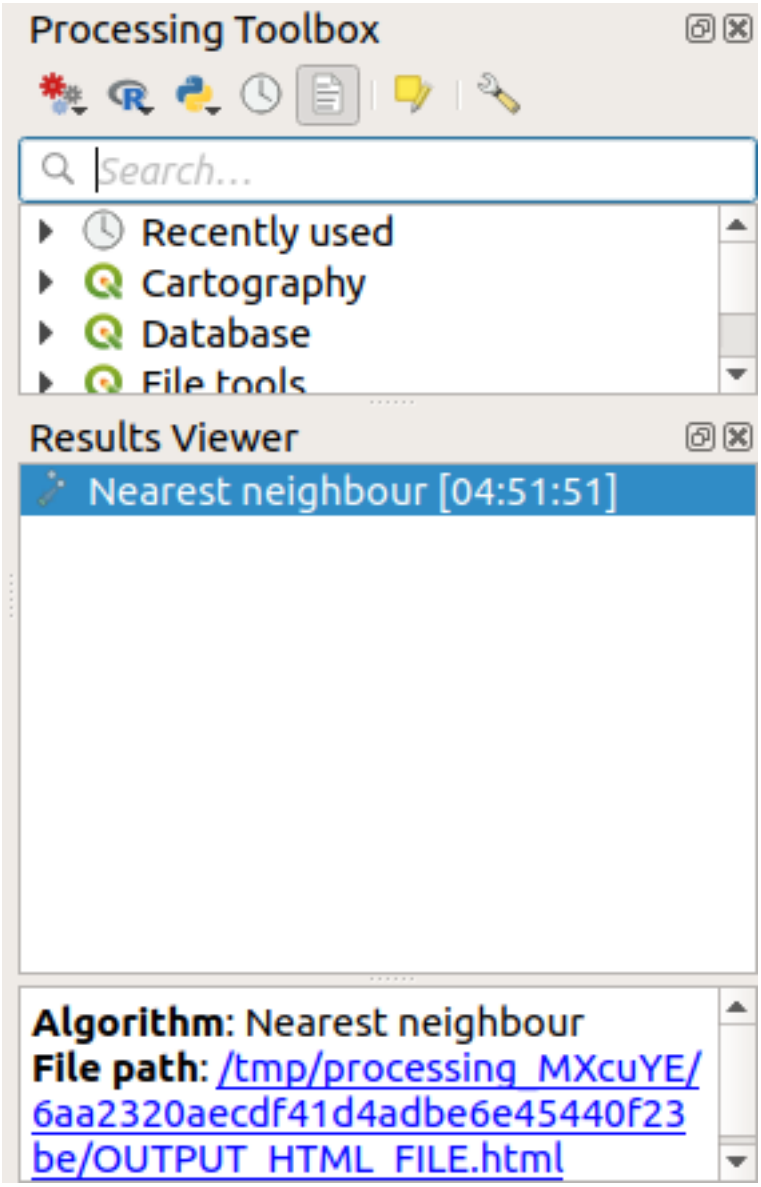
With these parameters, the *Distance Matrix* tool calculates distance statistics for each point of the input layer with respect to their two nearest points in the target layer. The fields of the output layer contain the mean, standard deviation, minimum and maximum for the calculated distances.

For further testing, you may want to modify the *Output matrix type* option or the number of target points.

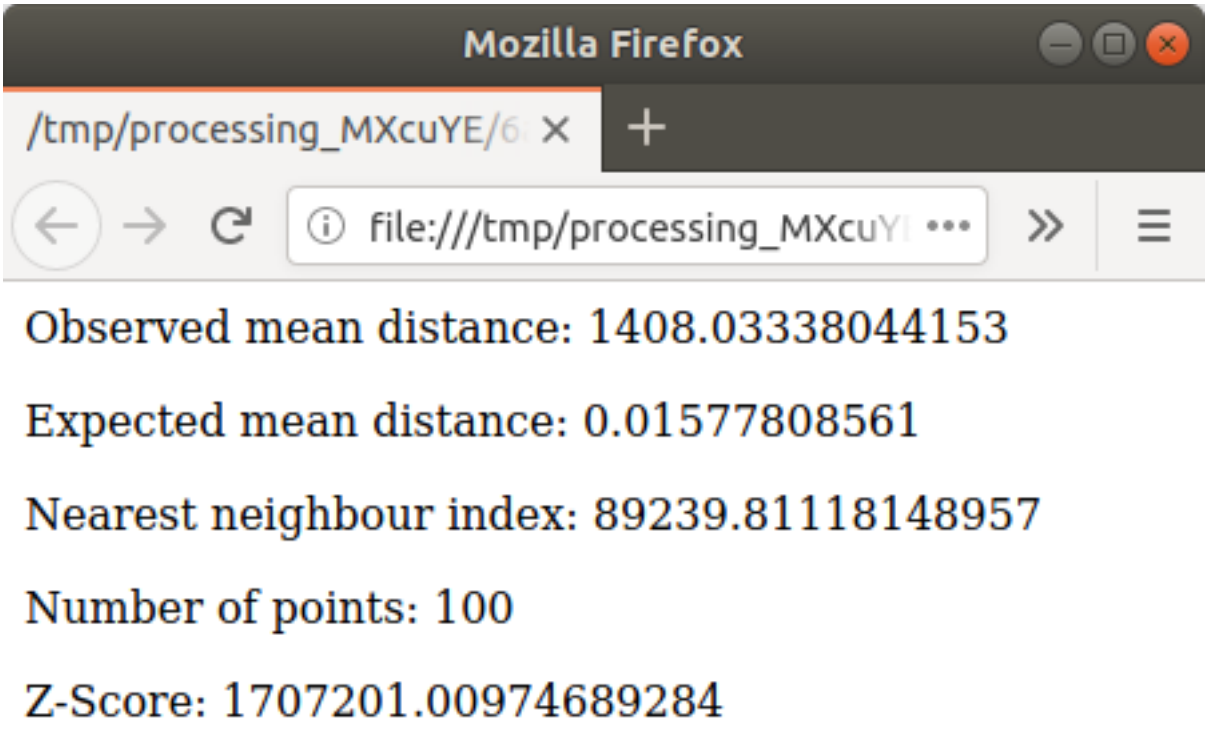
6.4.4 ??? Follow Along: Nearest Neighbor Analysis (within layer)

To do a nearest neighbor analysis of a point layer:

1. Choose *Vector analysis* ► *Nearest neighbor analysis*.
2. In the dialog that appears, select the *Random points* layer and click *Run*.
3. The results will appear in the Processing *Result Viewer* Panel.



4. Click on the blue link to open the `html` page with the results:



6.4.5 ??? Follow Along: Mean Coordinates

To get the mean coordinates of a dataset:

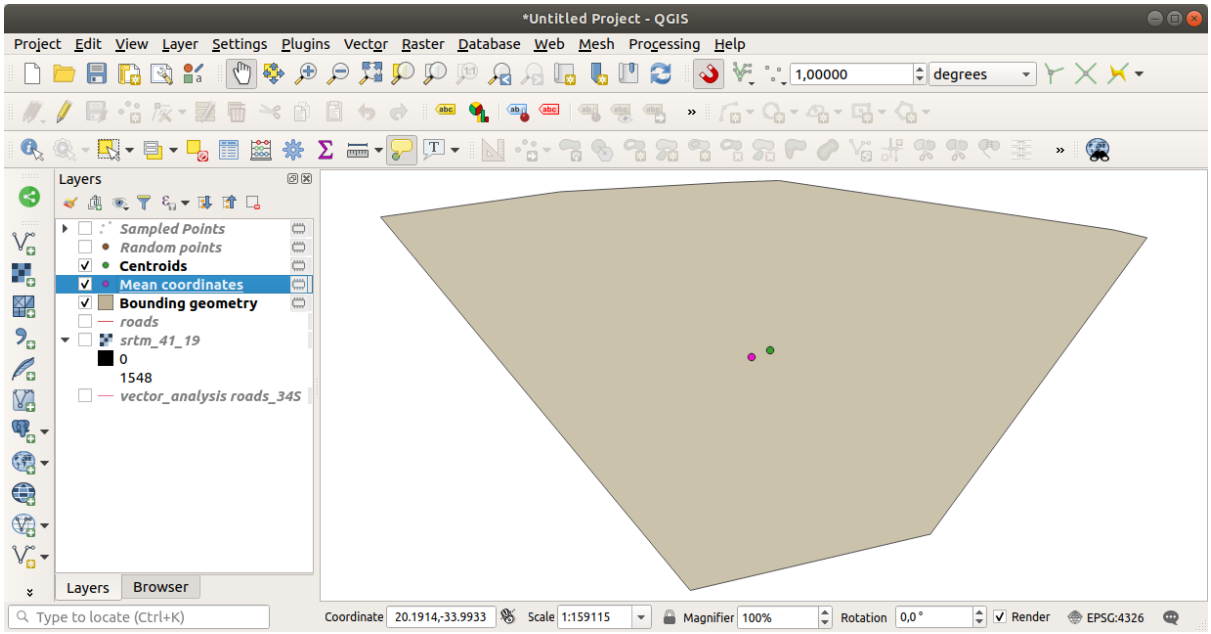
1. Start *Vector analysis* ► *Mean coordinate(s)*
2. In the dialog that appears, specify *Random points* as *Input layer*, and leave the optional choices unchanged.
3. Click *Run*.

Let us compare this to the central coordinate of the polygon that was used to create the random sample.

1. Start *Vector geometry* ► *Centroids*
2. In the dialog that appears, select `Bounding geometry` as the input layer.

As you can see, the mean coordinates (pink point) and the center of the study area (in green) don't necessarily coincide.

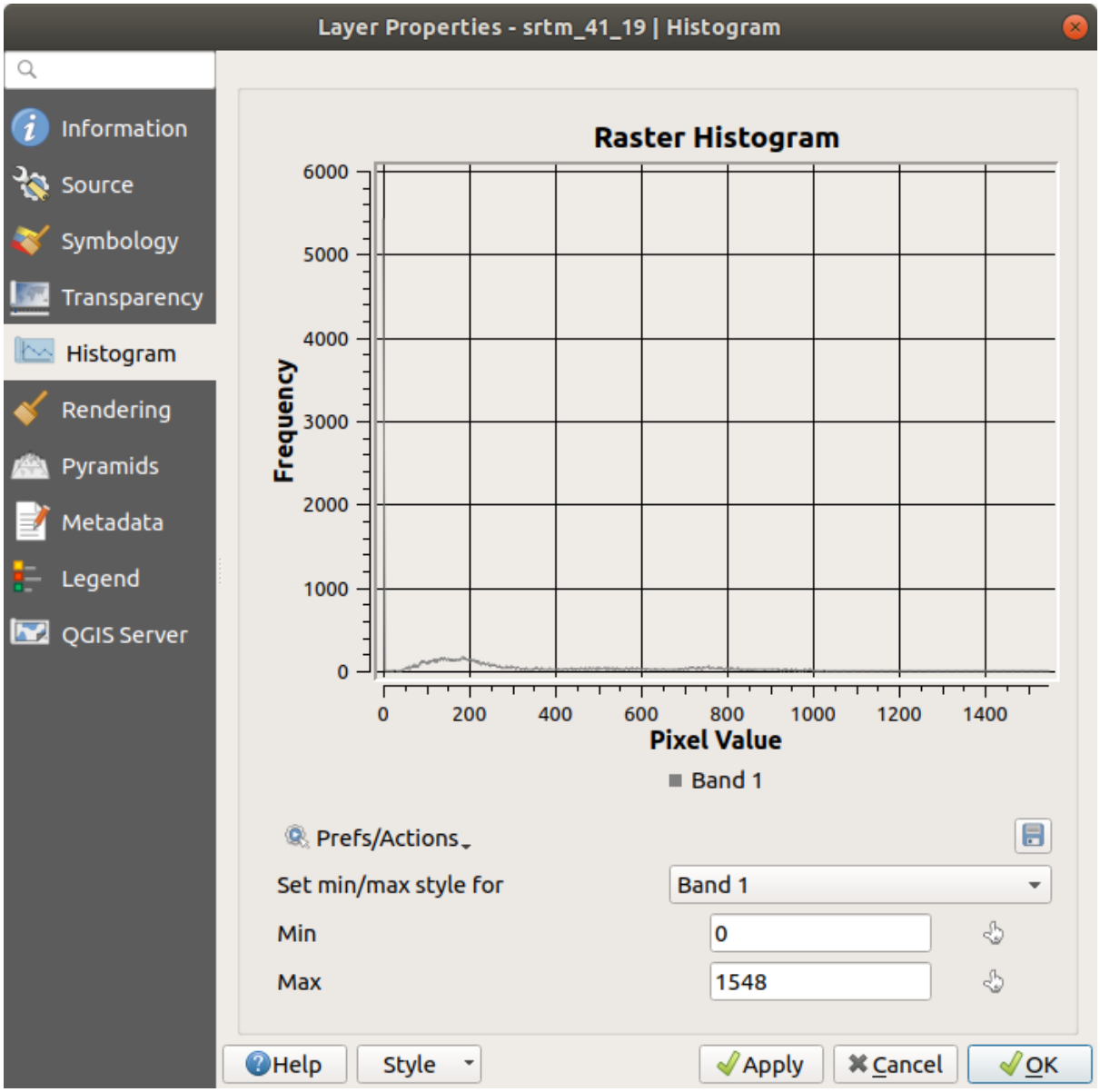
The centroid is the barycenter of the layer (the barycenter of a square is the center of the square) while the mean coordinates represent the average of all node coordinates.



6.4.6 ??? Follow Along: Image Histograms

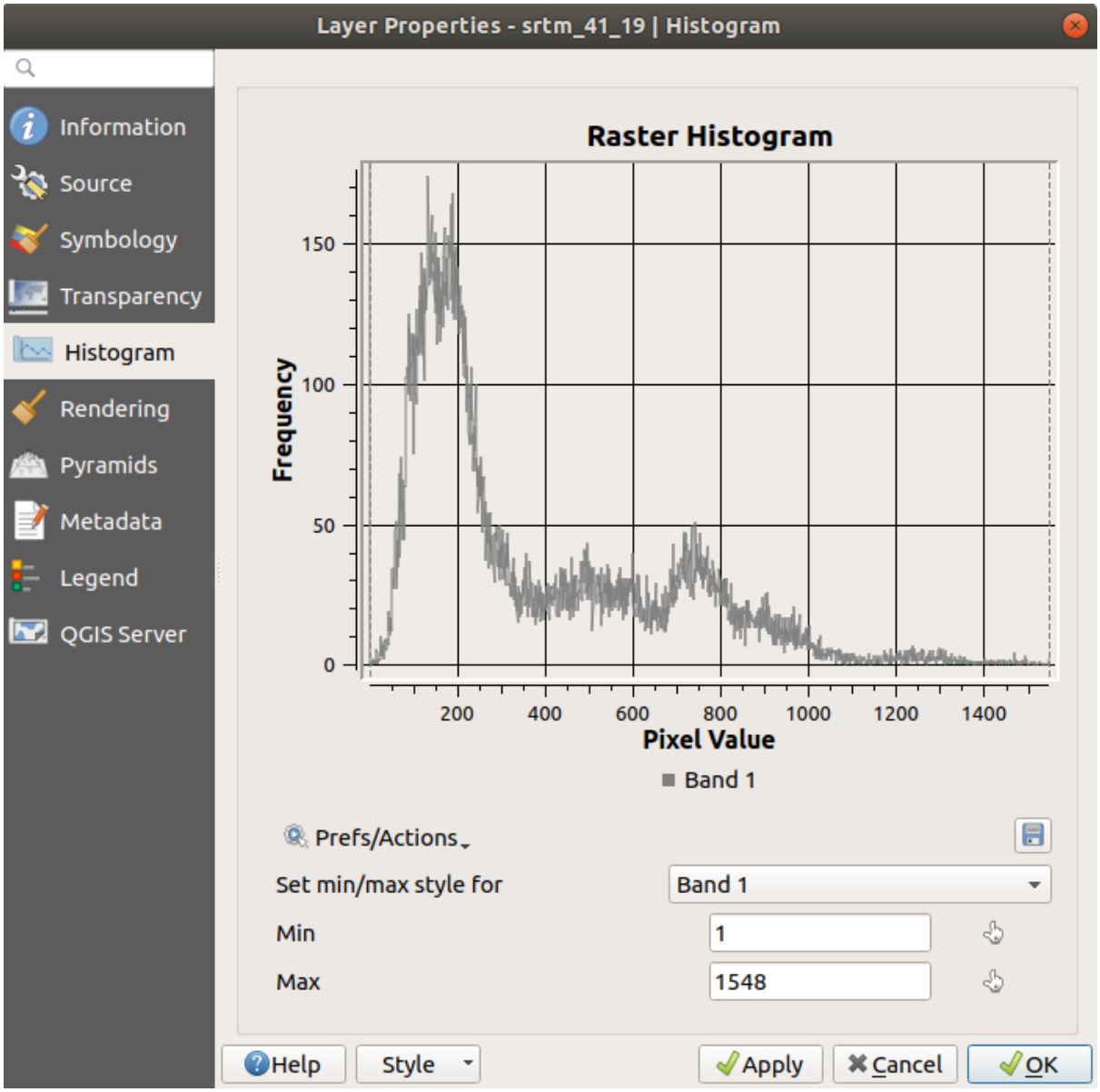
The histogram of a dataset shows the distribution of its values. The simplest way to demonstrate this in QGIS is via the image histogram, available in the *Layer Properties* dialog of any image layer (raster dataset).

1. In your *Layers* panel, right-click on the `srtm_41_19` layer
2. Select *Properties*
3. Choose the *Histogram* tab. You may need to click on the *Compute Histogram* button to generate the graphic. You will see a graph that shows the frequency distribution for the raster values.



4. The graph can be exported as an image with the Save plot button
5. You can see more detailed information about the layer in the *Information* tab (the mean and max values are estimated, and may not be exact).

The mean value is `332.8` (estimated to 324.3), and the maximum value is `1699` (estimated to 1548)! You can zoom in the histogram. Since there are a lot of pixels with value `0`, the histogram looks compressed vertically. By zooming in to cover everything but the peak at `0`, you will see more details:



Nota: If the mean and maximum values are not the same as above, it can be due to the min/max value calculation. Open the *Symbology* tab and expand the *Min / Max Value Settings* menu. Choose *Min / max* and click on *Apply*.

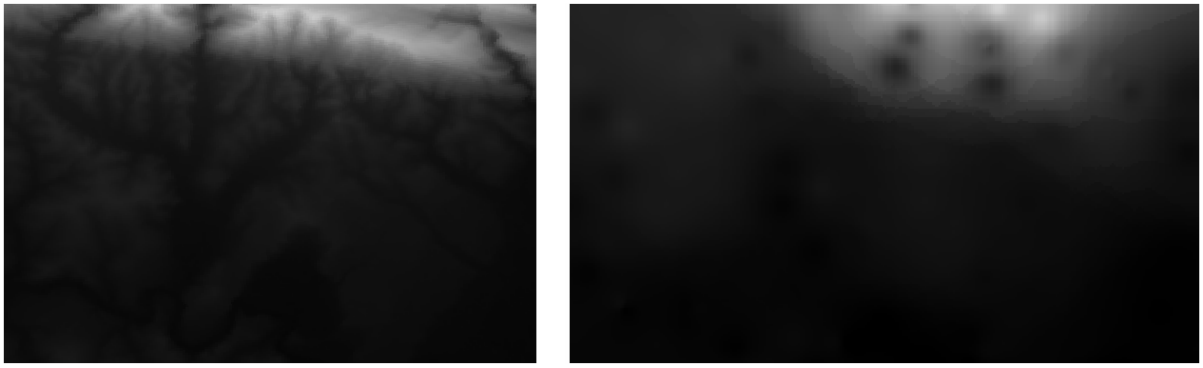
Keep in mind that a histogram shows you the distribution of values, and not all values are necessarily visible on the graph.

6.4.7 ??? Follow Along: Spatial Interpolation

Let's say you have a collection of sample points from which you would like to extrapolate data. For example, you might have access to the *Sampled points* dataset we created earlier, and would like to have some idea of what the terrain looks like.

1. To start, launch the *GDAL* ► *Raster analysis* ► *Grid (IDW with nearest neighbor searching)* tool in the *Processing Toolbox*.
2. For *Point layer* select `Sampled points`
3. Set *Weighting power* to `5.0`
4. In *Advanced parameters*, set *Z value from field* to `rvalue_1`
5. Finally click on *Run* and wait until the processing ends
6. Close the dialog

Here is a comparison of the original dataset (left) to the one constructed from our sample points (right). Yours may look different due to the random nature of the location of the sample points.



As you can see, 100 sample points aren't really enough to get a detailed impression of the terrain. It gives a very general idea, but it can be misleading as well.

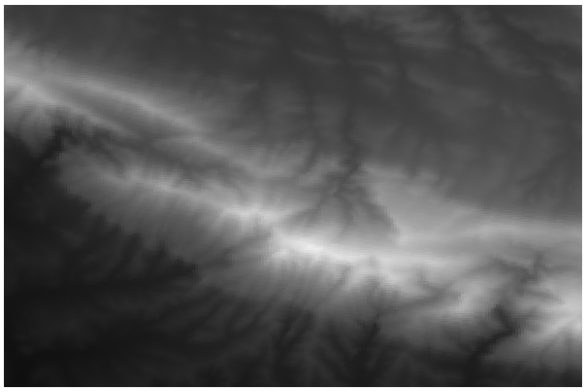
6.4.8 ??? Try Yourself: Different interpolation methods

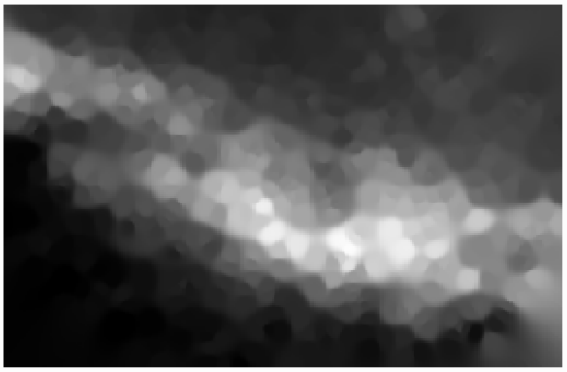
1. Use the processes shown above to create a set of 10 000 random points

   Nota: If the number of points is really big, the processing time can take a long time.

2. Use these points to sample the original DEM
3. Use the *Grid (IDW with nearest neighbor searching)* tool on this dataset.
4. Set *Power* and *Smoothing* to `5.0` and `2.0`, respectively.

The results (depending on the positioning of your random points) will look more or less like this:





This is a better representation of the terrain, due to the greater density of sample points. Remember, larger samples give better results.

6.4.9 In Conclusion

QGIS has a number of tools for analyzing the spatial statistical properties of datasets.

6.4.10 What's Next?

Now that we have covered vector analysis, why not see what can be done with rasters? That is what we will do in the next module!

CAPÍTULO 7

Module: Rasters

Anteriormente já usamos rasters para a digitalização, mas os dados raster também podem ser utilizados diretamente. Neste capítulo, vamos explicar como isto pode ser feito no QGIS.

7.1 Lesson: Working with Raster Data

Raster data is quite different from vector data. Vector data has discrete features with geometries constructed out of vertices, and perhaps connected with lines and/or areas. Raster data, however, is like any image. Although it may portray various properties of objects in the real world, these objects don't exist as separate objects. Rather, they are represented using pixels with different values.

During this module you are going to use raster data to supplement your existing GIS analysis.

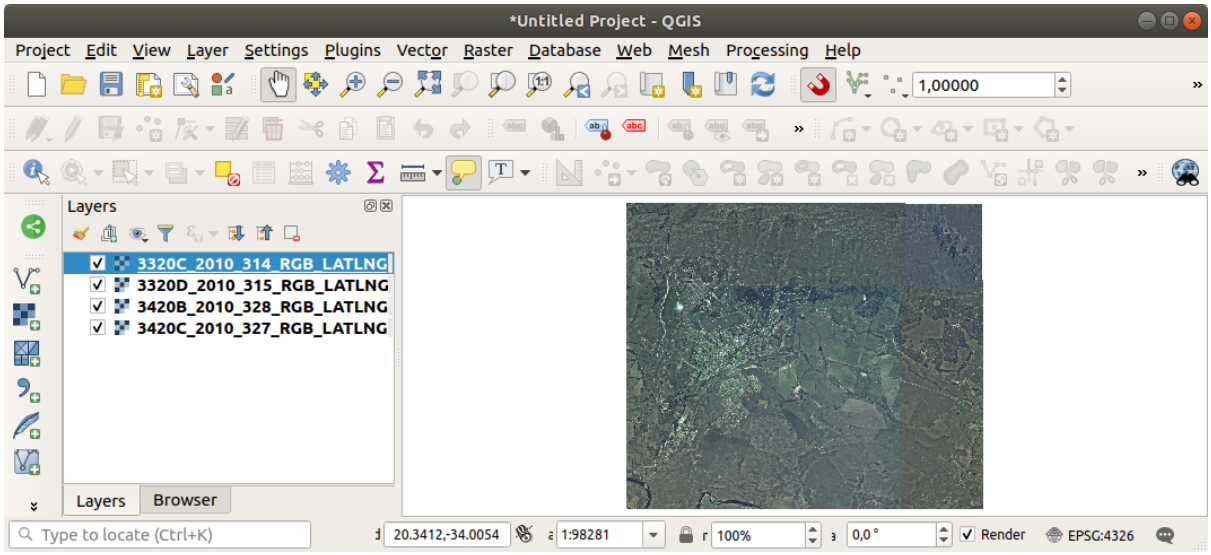
The goal for this lesson: To learn how to work with raster data in QGIS.

7.1.1 ??? Follow Along: Loading Raster Data

Raster data can be loaded with the same methods we used for vector data. However we suggest to use the *Browser* Panel.

1. Open the *Browser* Panel and expand the `exercise_data/raster` folder.
2. Load all the data in this folder:
    - `3320C_2010_314_RGB_LATLNG.tif`
    - `3320D_2010_315_RGB_LATLNG.tif`
    - `3420B_2010_328_RGB_LATLNG.tif`
    - `3420C_2010_327_RGB_LATLNG.tif`

You should see the following map:



There we have it - four aerial images covering our study area.

7.1.2 ??? Follow Along: Create a Virtual Raster

Now as you can see from this, your solution layer lies across all four images. What this means is that you are going to have to work with four rasters all the time. That's not ideal. It would be better to have one file to work with.

Luckily, QGIS allows you to do exactly this, and without needing to actually create a new raster file. You can create a **Virtual Raster**. This is also often called a *Catalog*, which explains its function. It's not really a new raster. Rather, it is a way to organize your existing rasters into one catalog: one file for easy access.

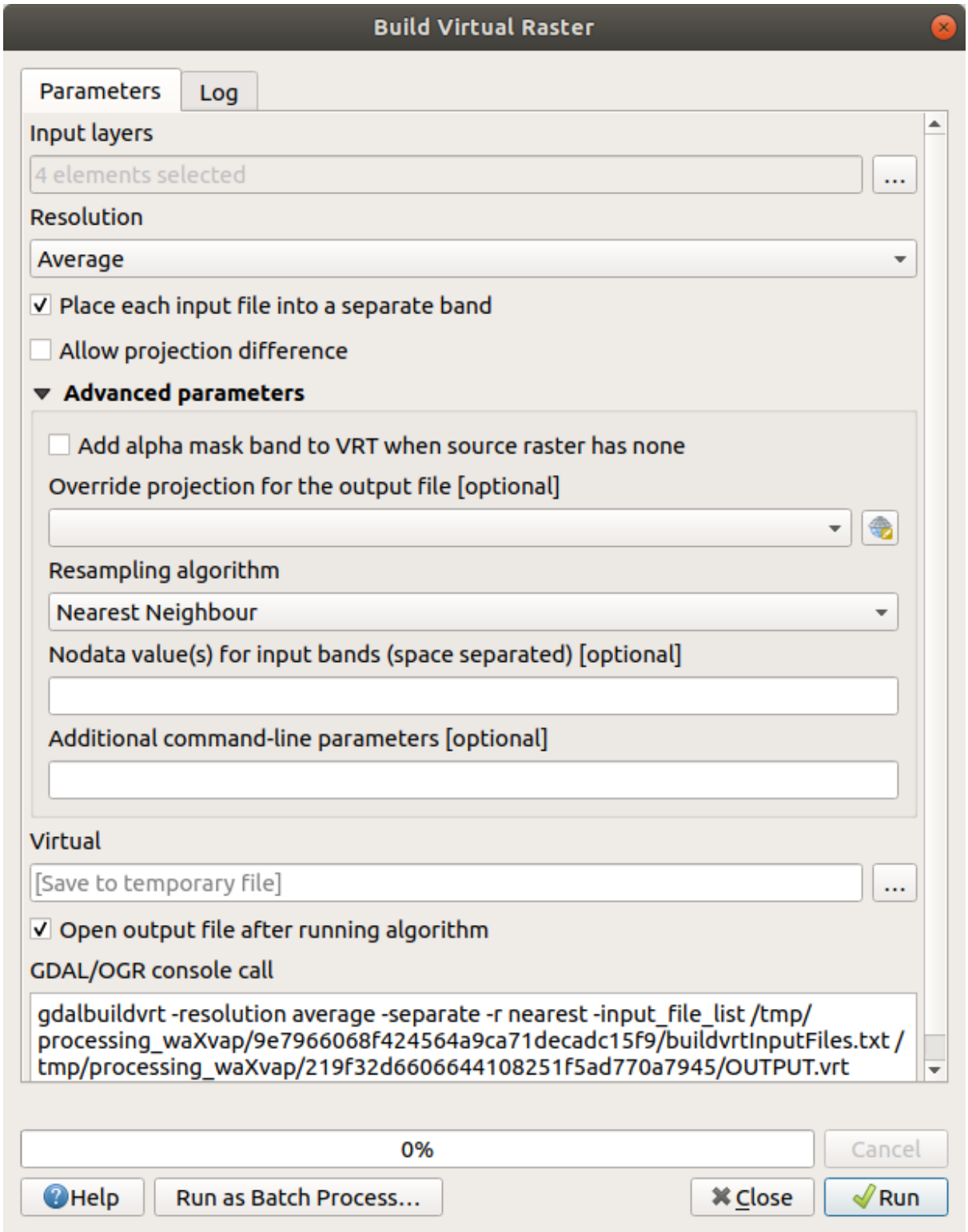
To make a catalog we will use the *Processing* ► *Toolbox*.

1. Open the *Build virtual raster* algorithm from the *GDAL* ► *Raster miscellaneous*;
2. In the dialog that appears, click on the … button next to the *Input layers* parameter and check all the layers or use the *Select All* button;
3. Uncheck *Place each input file into a separate band*. Notice the code that is generated at the bottom of the dialog under *GDAL/OGR console call*. This is the command that will be executed when you click *Run*. The text updates as you change different options in the dialog.

   Nota: Keep in mind that you can copy and paste the text in the `OSGeo Shell` (Windows user) or `Terminal` (Linux and OSX users) to run the command. You can also create a script for each GDAL command. This is very handy when the procedure is taking a long time or when you want to schedule specific tasks. Use the *Help* button to get more help on the syntax of the command.

4. Finally click on *Run*.

Nota: As you know from the previous modules, *Processing* creates temporary layers by default. To save the file, click on the … button under *Virtual*.



You can now remove the original four rasters from the *Layers* Panel and leave only the output virtual catalog raster.

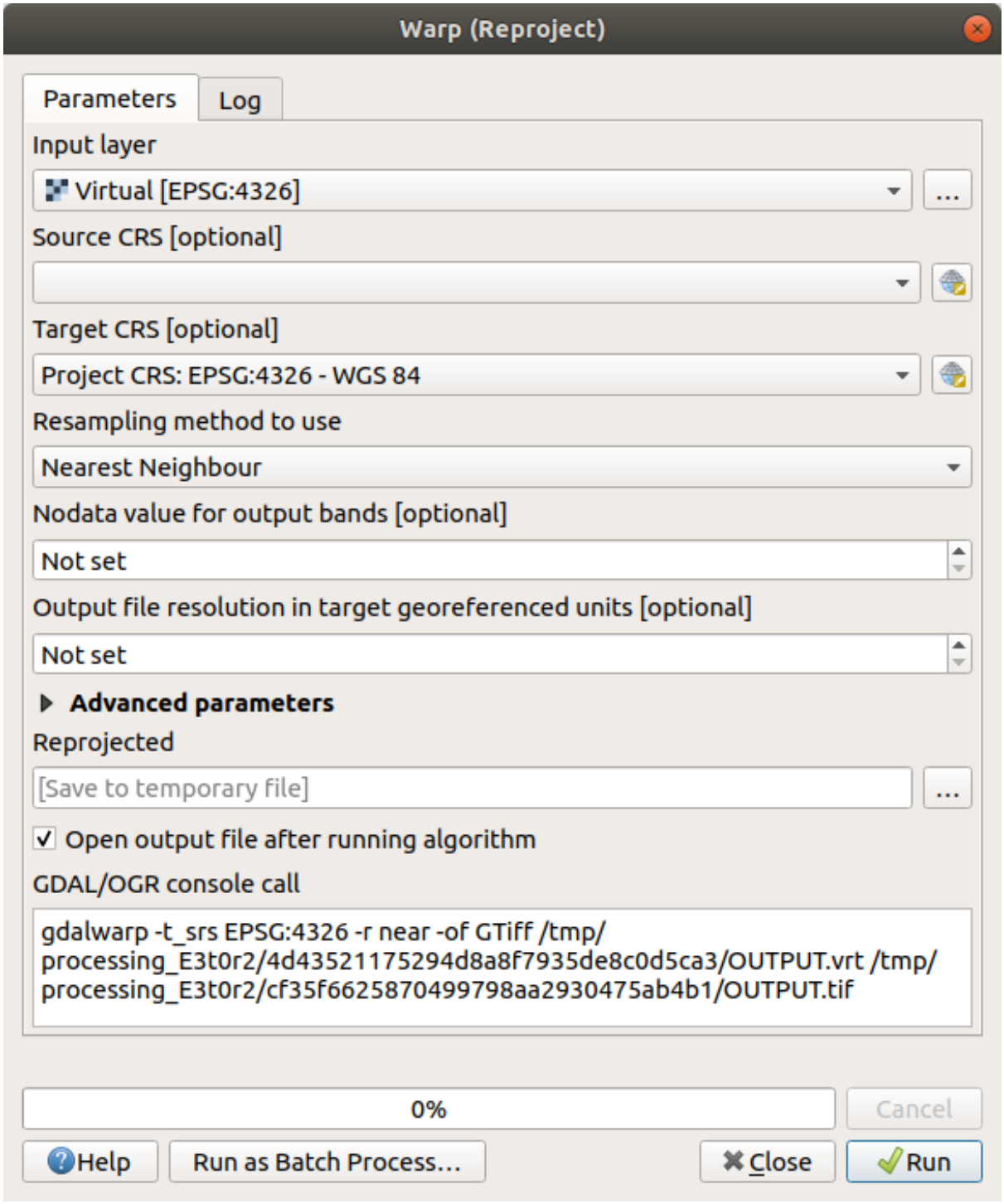
7.1.3 ??? Transforming Raster Data

The above methods allow you to virtually merge datasets using a catalog, and to reproject them «on the fly». However, if you are setting up data that you'll be using for quite a while, it may be more efficient to create new rasters that are already merged and reprojected. This improves performance while using the rasters in a map, but it may take some time to set up initially.

Reprojecting rasters

Open *Warp (reproject)* from *GDAL* ► *Raster projections*.

You can also reproject virtual rasters (catalogs), enable multithreaded processing, and more.



Merging rasters

If you need to create a new raster layer and save it to disk you can use the merge algorithm.

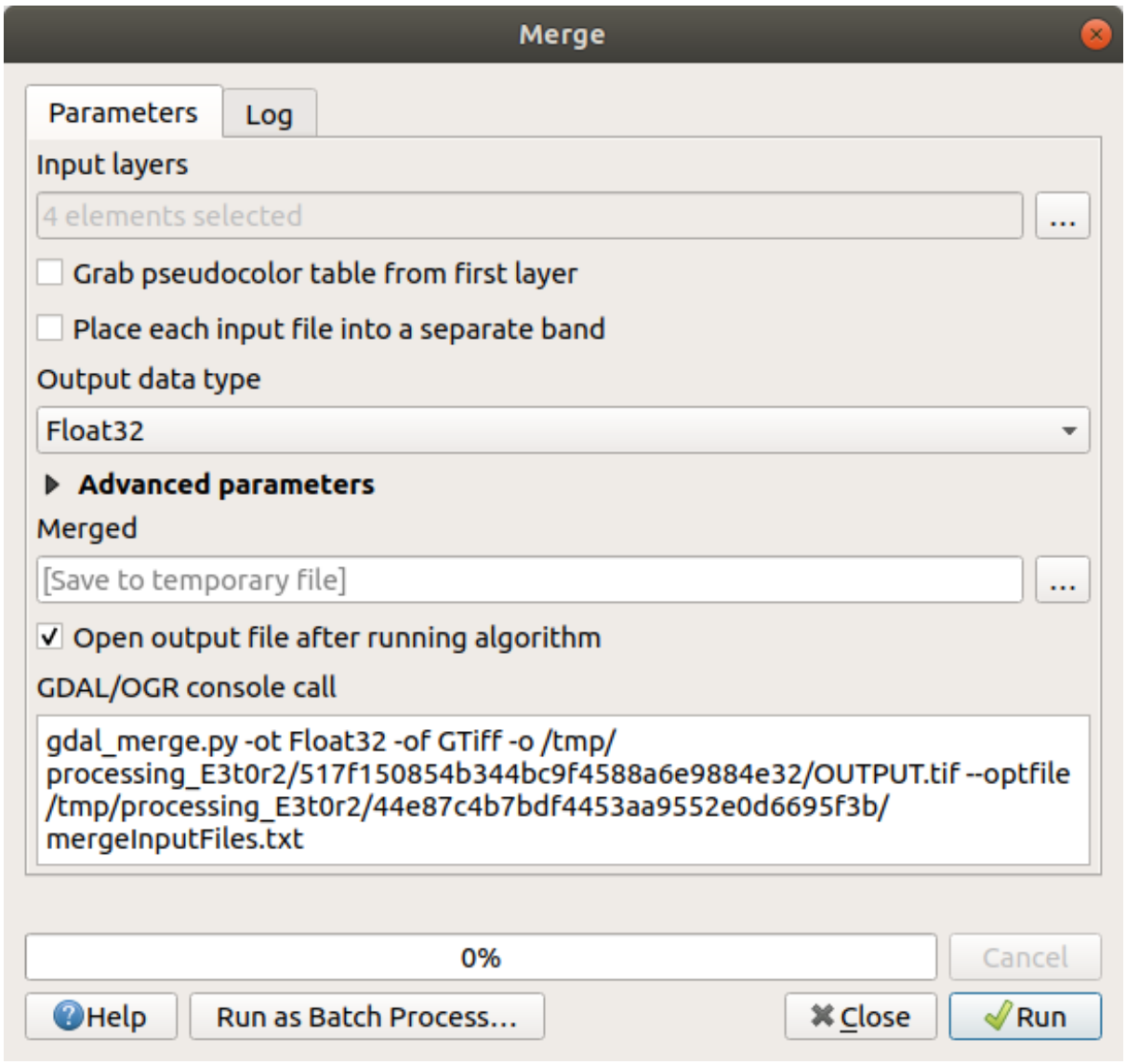
Nota: Depending on how many raster files you are merging and their resolution, the new raster file created can be really big. Consider instead to create a raster catalog as described in the *Create a Virtual Raster* section.

---

1. Click on the *Merge* algorithm from the *GDAL* ► *Raster miscellaneous* menu.
2. As we did for the *Create a Virtual raster*, use the … button to choose which layers you want to merge.

You can also specify a Virtual raster as input, and then all of the rasters that it consists of will be processed.

3. If you know the GDAL library, you can also add your own options by opening the *Advanced parameters* menu.



7.1.4 In Conclusion

QGIS makes it easy to include raster data into your existing projects.

7.1.5 What's Next?

Next, we'll use raster data that isn't aerial imagery, and see how symbolization is useful in the case of rasters as well.

7.2 Lesson: Changing Raster Symbology

Not all raster data are aerial photos. There are many other forms of raster data, and in many of those cases, it is essential to symbolize the them so that they becomes properly visible and useful.

The goal for this lesson: To change the symbology for a raster layer.

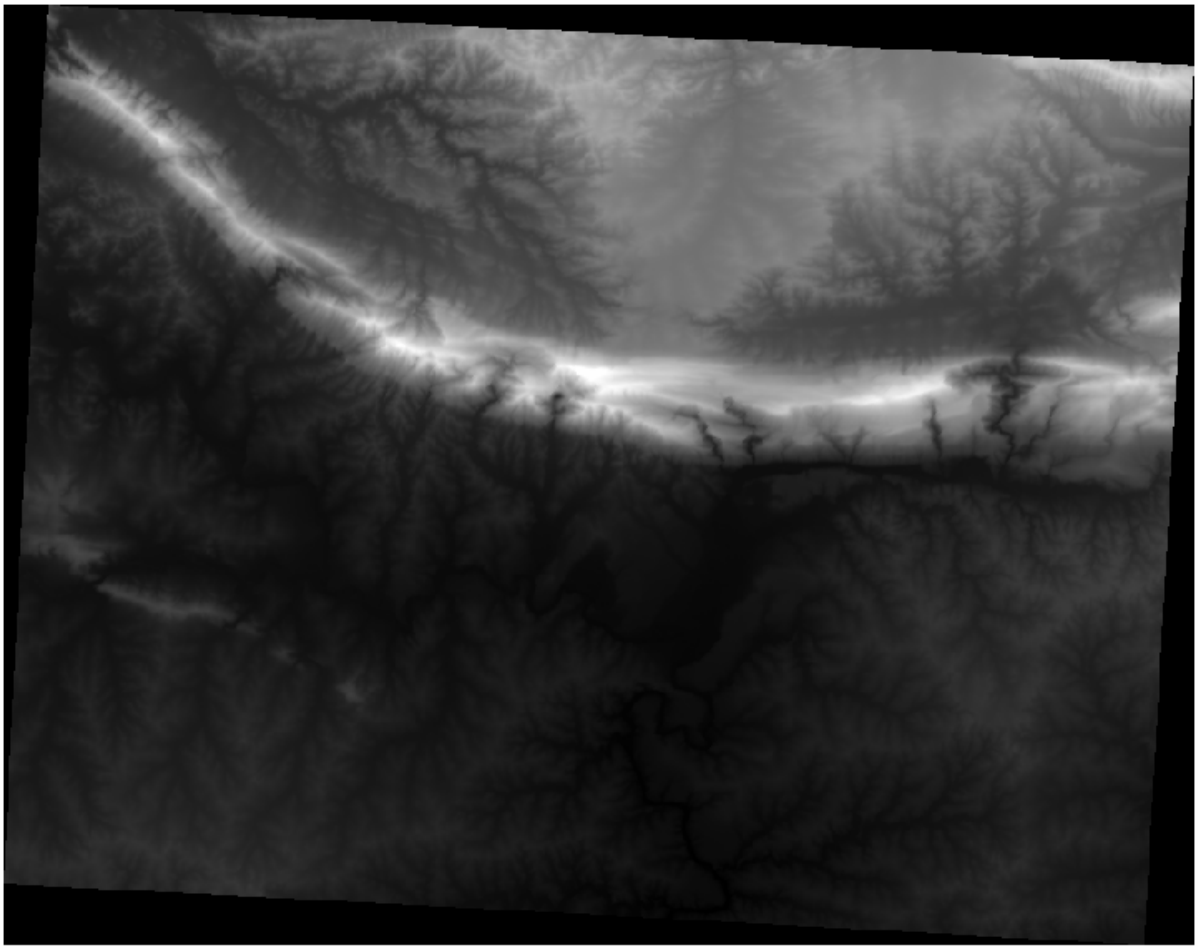
7.2.1 ??? Try Yourself:

1. Use the *Browser* Panel to load `srtm_41_19.tif`, found under `exercise_data/raster/SRTM/`
2. Zoom to the extent of this layer by right-clicking on it in the *Layers panel* and selecting *Zoom to Layer*.

This dataset is a *Digital Elevation Model (DEM)*. It is a map of the elevation (altitude) of the terrain, allowing us to see where the mountains and valleys are, for example.

While each pixel of the dataset of the previous section contained color information, in a *DEM*, each pixel contains elevation values.

Once the DEM is loaded, you will notice that it is a grayscale representation:



QGIS has automatically applied a stretch to the pixel values of the image for visualization purposes, and we will learn more about how this works as we continue.

7.2.2 ??? Follow Along: Changing Raster Layer Symbology

You have two different options to change the raster symbology:

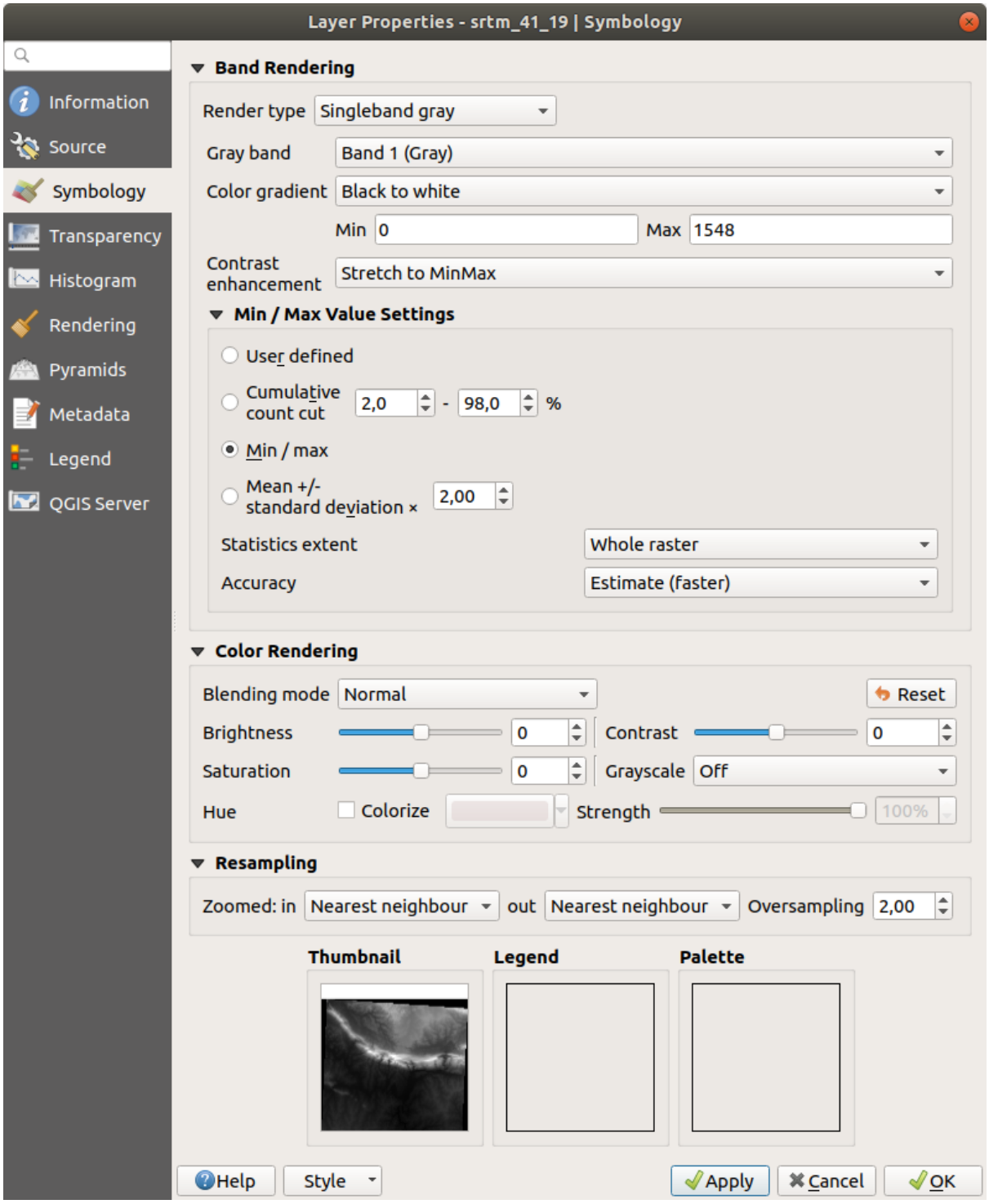
1. Within the *Layer Properties* dialog, by right-clicking on the layer in the Layer tree and selecting the *Properties* option. Then switch to the *Symbology* tab
2. By clicking on the Open the Layer Styling panel button right above the *Layers* panel (shortcut `F7`). This will open the *Layer Styling* panel, where you can switch to the Symbology tab.

Choose the method you prefer to work with.

7.2.3 ??? Follow Along: Singleband gray

When you load a raster file, if it is not a photo image like the ones of the previous section, the default style is set to a grayscale gradient.

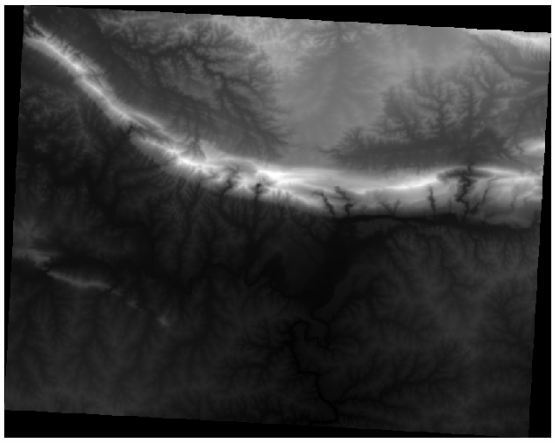
Let's explore some of the features of this renderer.



The default *Color gradient* is set to `Black to white`, meaning that low pixel values are black and while high values are white. Try to invert this setting to `White to black` and see the results.

Very important is the *Contrast enhancement* parameter: by default it is set to `Stretch to MinMax` meaning that the pixel values are stretched to the minimum and maximum values.

Look at the difference with the enhancement (left) and without (right):



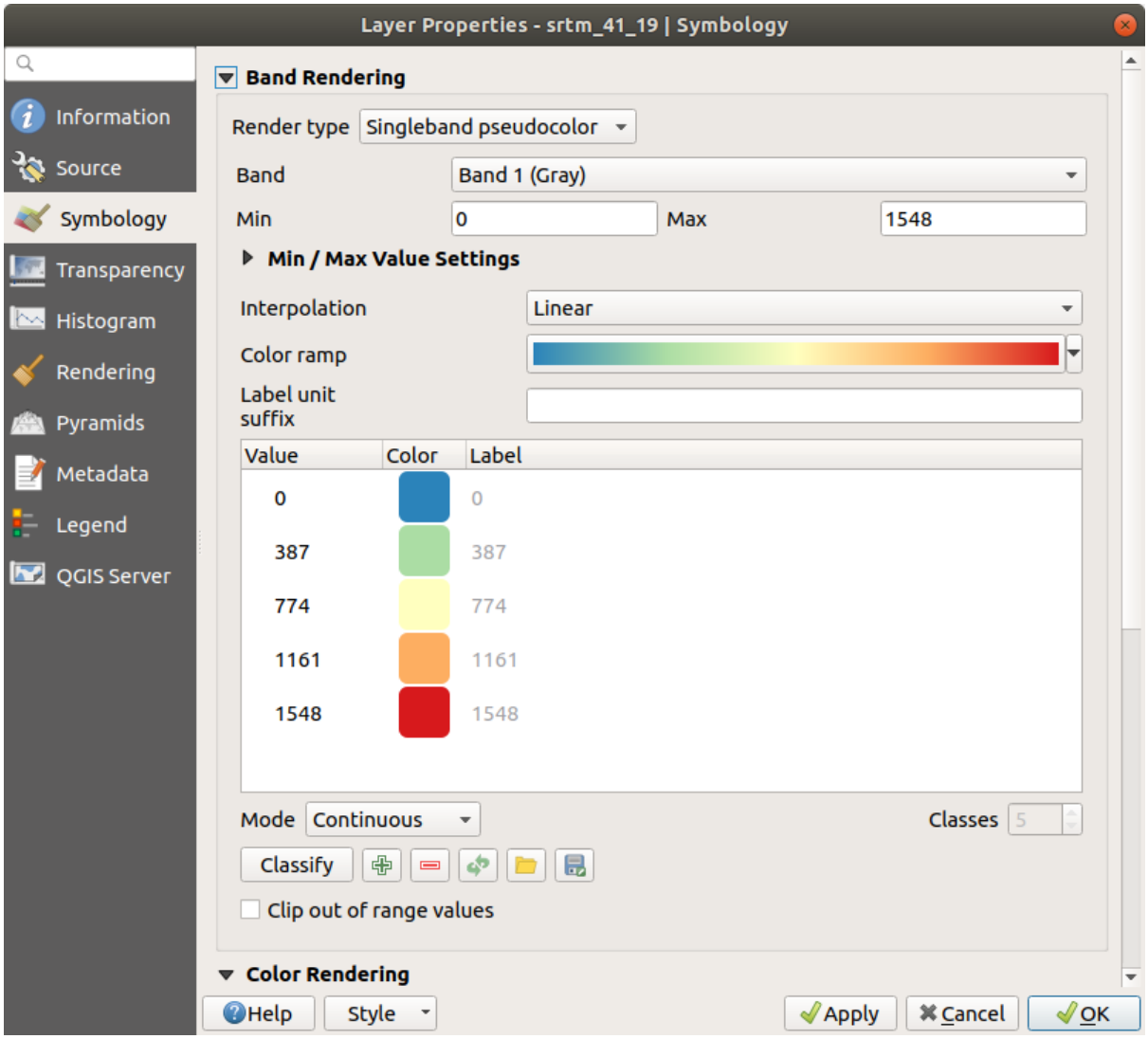
But what are the minimum and maximum values that should be used for the stretch? The ones that are currently under *Min / Max Value Settings*. There are many ways to calculate the minimum and maximum values and use them for the stretch:

1. **User Defined**: you enter the *Min* and *Max* values manually
2. **Cumulative count cut**: this is useful when you have some extreme low or high values. It *cuts* the `2%` (or the value you choose) of these values
3. **Min / max**: the *Real* or *Estimated* minimum and maximum values of the raster
4. **Mean +/- standard deviation**: the values will be calculated according to the mean value and the standard deviation

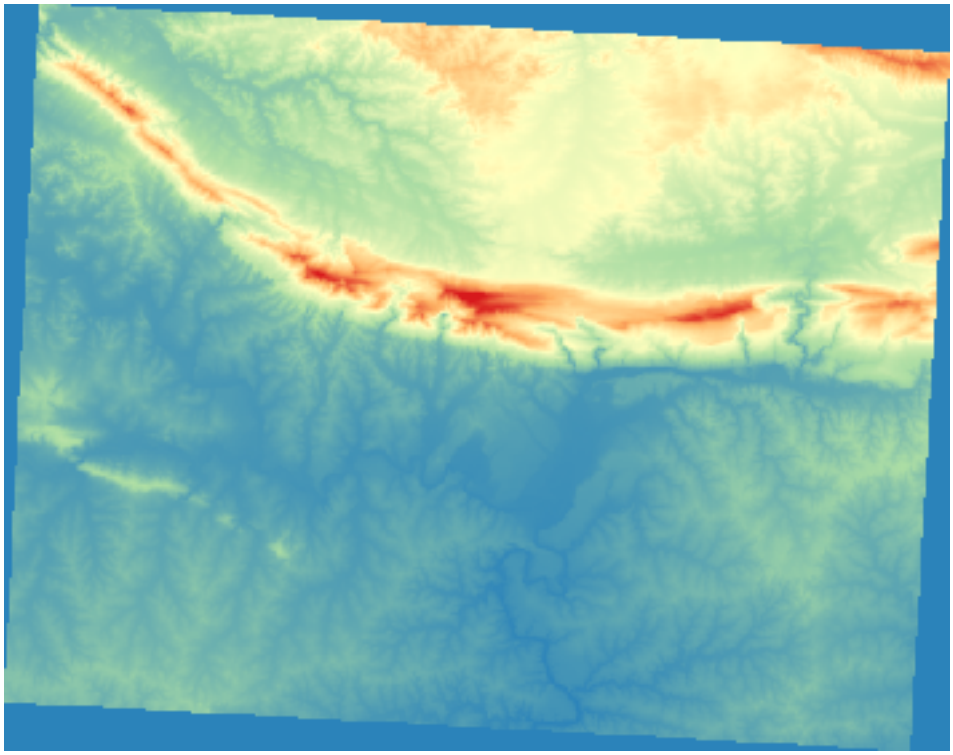
7.2.4 ??? Follow Along: Singleband pseudocolor

Grayscales are not always great styles for raster layers. Let's try to make the DEM more colorful.

- Change the *Render type* to *Singleband pseudocolor*. If you don't like the default colors loaded, select another *Color ramp*
- Click the *Classify* button to generate a new color classification
- If it is not generated automatically click on the *OK* button to apply this classification to the DEM



You'll see the raster looking like this:

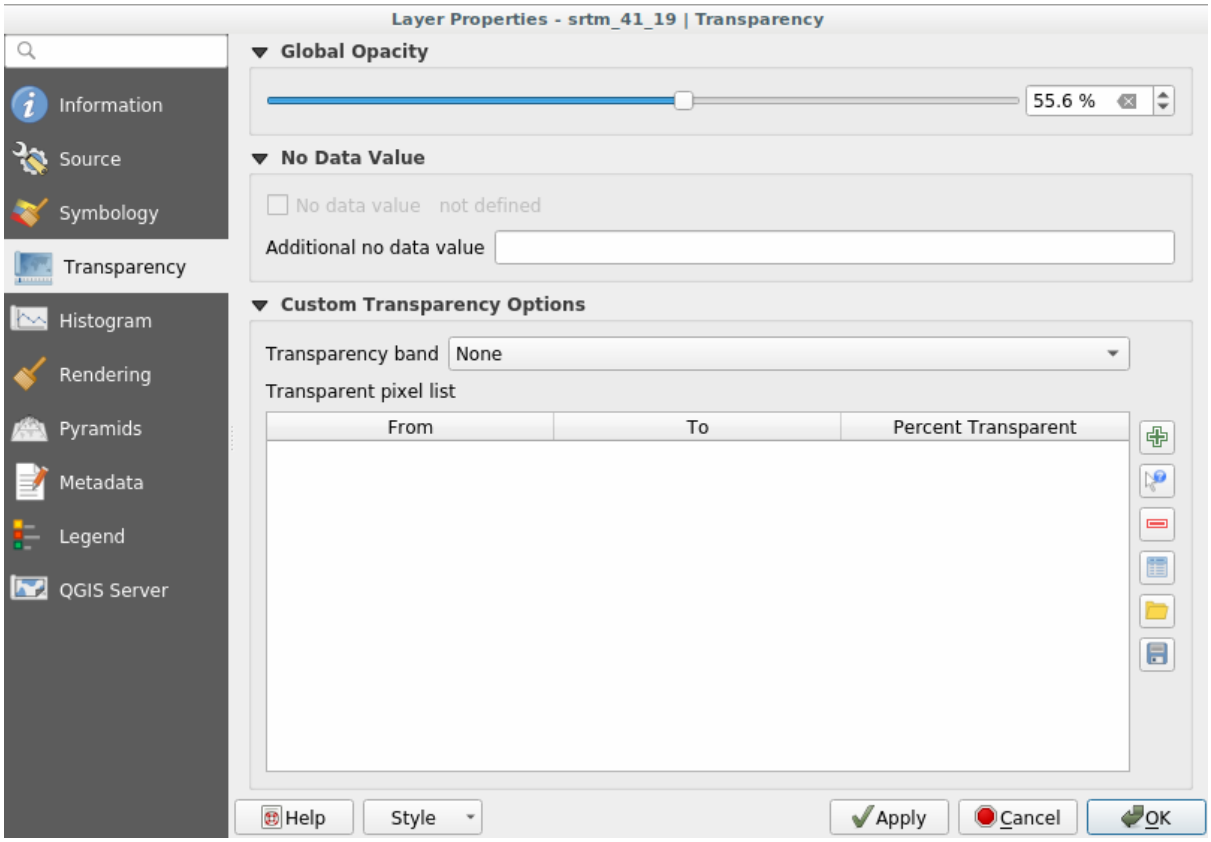


This is an interesting way of looking at the DEM. You will now see that the values of the raster are again properly displayed, going from blue for the lower areas to red for the higher ones.

7.2.5 Follow Along: Changing the transparency

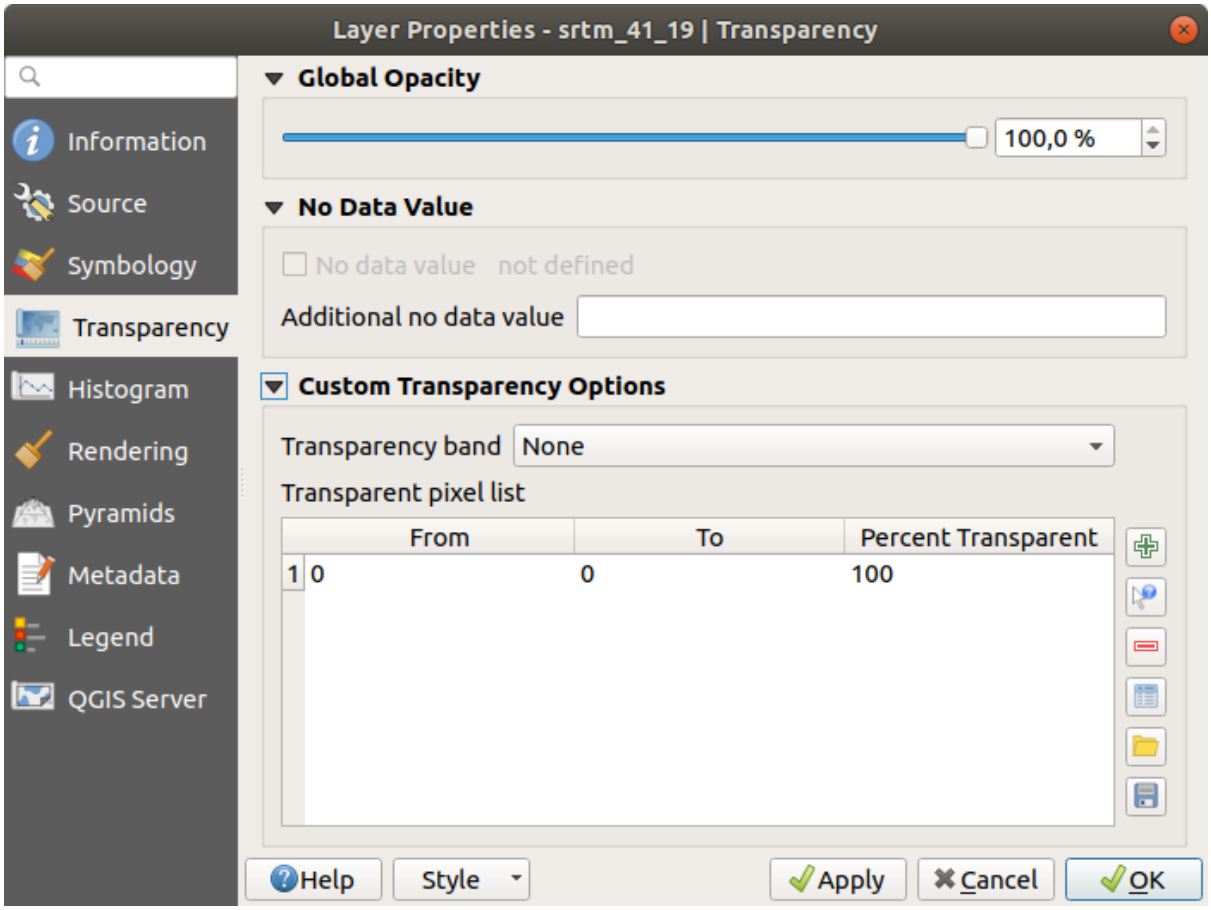
Sometimes changing the transparency of the whole raster layer can help you to see other layers covered by the raster itself and better understand the study area.

To change the transparency of the whole raster switch to the *Transparency* tab and use the slider of the *Global Opacity* to lower the opacity:

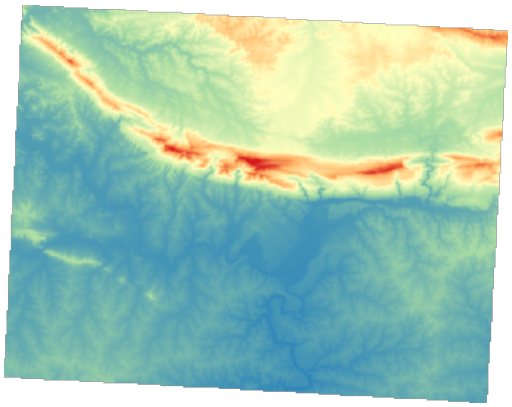


More interesting is changing the transparency for some pixel values. For example in the raster we used you can see a homogeneous color at the corners. To set these pixels as transparent, go to *Custom Transparency Options* in the *Transparency* tab.

- By clicking on the Add values manually button, you can add a range of values and set their transparency percentage
- For single values the Add values from display button is more useful
- Click on the Add values from display button. The dialog disappears, and you can interact with the map.
- Click on the homogeneous color in a corner of the DEM
- You will see that the transparency table will be filled with the clicked values:



- Click on *OK* to close the dialog and see the changes.



See? The corners are now 100% transparent.

7.2.6 In Conclusion

These are some the basic functions to get you started with raster symbology. QGIS also gives you many other options, such as symbolizing a layer using paletted/unique values, representing different bands with different colors in a multispectral image, or making an automatic hillshade effect (useful only with DEM raster files).

7.2.7 Referência

The SRTM dataset was obtained from CGIAR CSI website.

7.2.8 What's Next?

Now that we can see our data displayed properly, let's investigate how we can analyze it further.

7.3 Lesson: Terrain Analysis

Certain types of rasters allow you to gain more insight into the terrain that they represent. Digital Elevation Models (DEMs) are particularly useful in this regard. In this lesson you will use terrain analysis tools to find out more about the study area for the proposed residential development from earlier.

The goal for this lesson: To use terrain analysis tools to derive more information about the terrain.

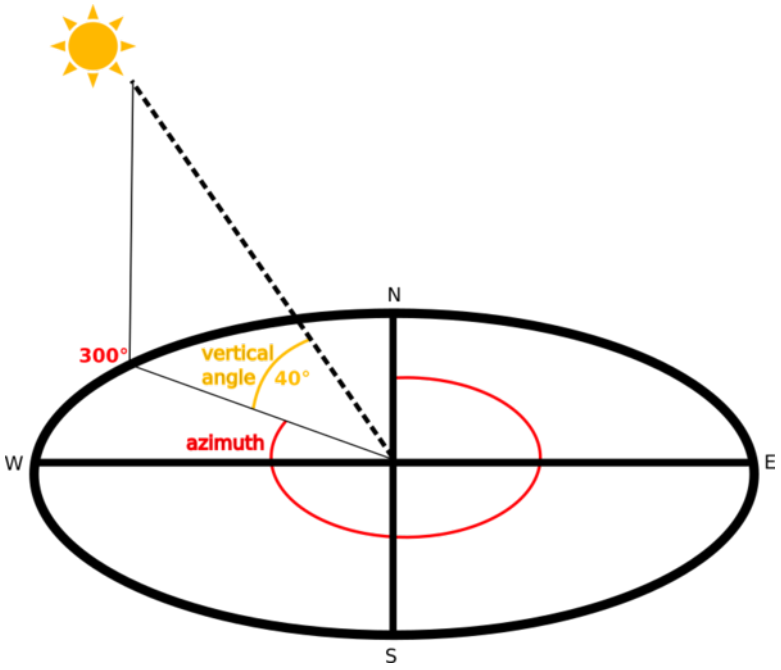
7.3.1 ??? Follow Along: Calculating a Hillshade

We are going to use the same DEM layer as in the previous lesson. If you are starting this chapter from scratch, use the *Browser* panel and load the `raster/SRTM/srtm_41_19.tif`.

The DEM layer shows you the elevation of the terrain, but it can sometimes seem a little abstract. It contains all the 3D information about the terrain that you need, but it doesn't look like a 3D object. To get a better impression of the terrain, it is possible to calculate a *hillshade*, which is a raster that maps the terrain using light and shadow to create a 3D-looking image.

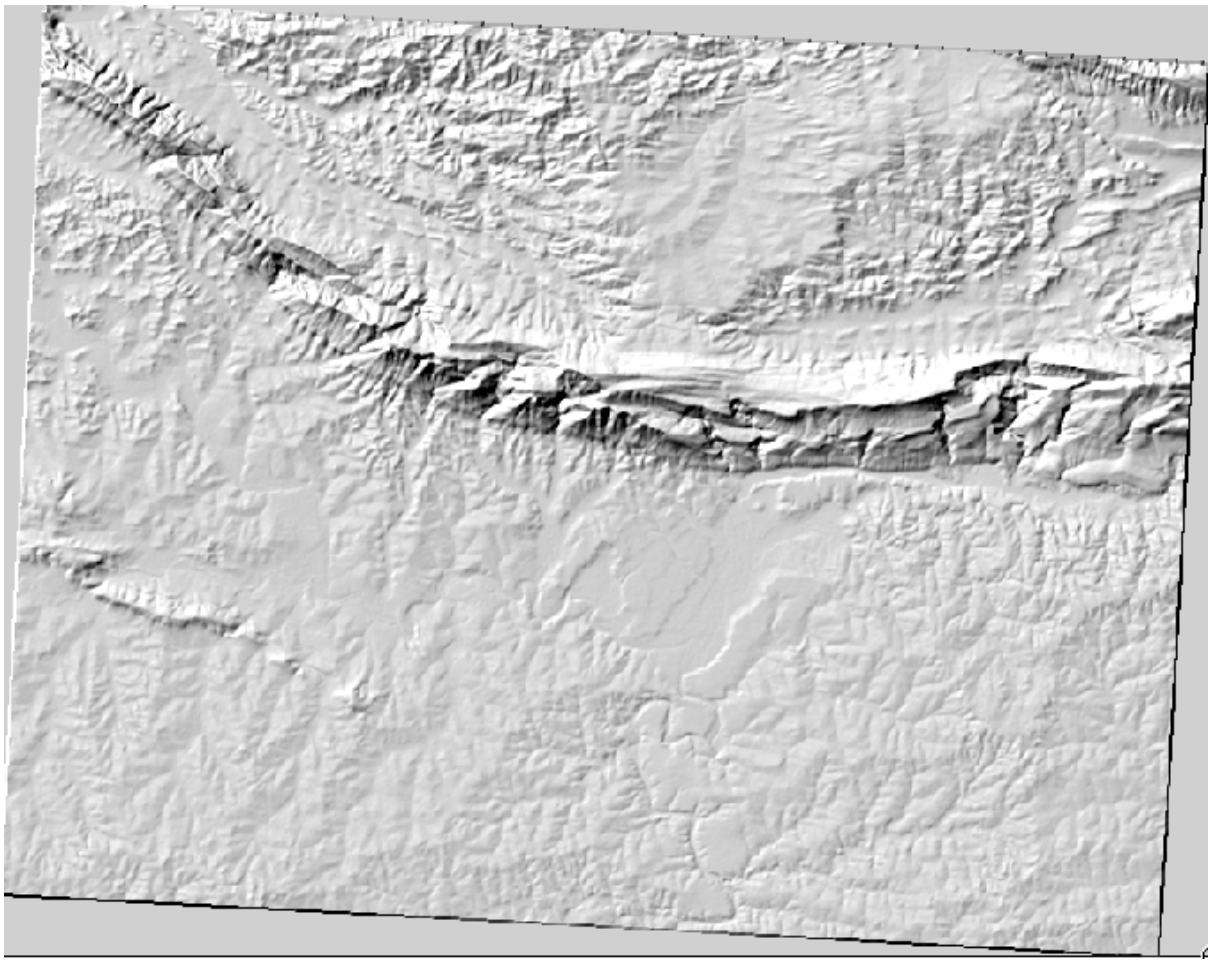
We are going to use algorithms in the *Raster* ► *Raster terrain analysis* menu.

1. Click on the *Hillshade* menu
2. The algorithm allows you to specify the position of the light source: *Azimuth* has values from 0 (North) through 90 (East), 180 (South) and 270 (West), while the *Vertical angle* sets how high the light source is (0 to 90 degrees).
3. We will use the following values:
   - *Z factor* at `1.0`
   - *Azimuth (horizontal angle)* at `300.0°`
   - *Vertical angle* at `40.0°`



4. Save the file in a new folder `exercise_data/raster_analysis/` with the name `hillshade.tif`
5. Finally click on *Run*

You will now have a new layer called *hillshade* that looks like this:



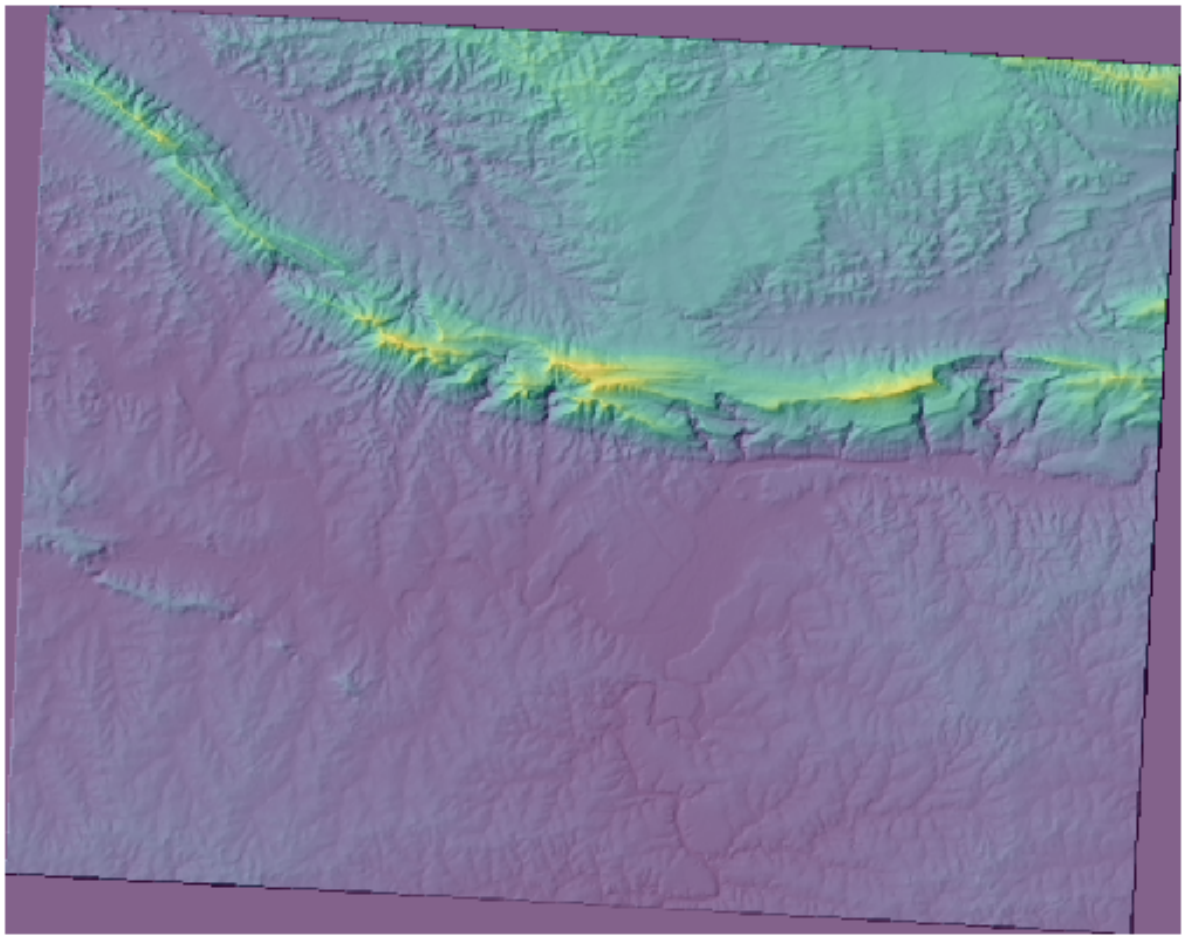
That looks nice and 3D, but can we improve on this? On its own, the hillshade looks like a plaster cast. Can't we use it together with our other, more colorful rasters somehow? Of course we can, by using the hillshade as an overlay.

7.3.2 ??? Follow Along: Using a Hillshade as an Overlay

A hillshade can provide very useful information about the sunlight at a given time of day. But it can also be used for aesthetic purposes, to make the map look better. The key to this is setting the hillshade to being mostly transparent.

1. Change the symbology of the original *srtm_41_19* layer to use the *Pseudocolor* scheme as in the previous exercise
2. Hide all the layers except the *srtm_41_19* and *hillshade* layers
3. Click and drag the *srtm_41_19* to be beneath the *hillshade* layer in the *Layers* panel
4. Set the *hillshade* layer to be transparent by clicking on the *Transparency* tab in the layer properties
5. Set the *Global opacity* to `50%`.

   You'll get a result like this:



6. Switch the *hillshade* layer off and back on in the *Layers* panel to see the difference it makes.

Using a hillshade in this way, it's possible to enhance the topography of the landscape. If the effect doesn't seem strong enough to you, you can change the transparency of the *hillshade* layer; but of course, the brighter the hillshade becomes, the dimmer the colors behind it will be. You will need to find a balance that works for you.

Remember to save the project when you are done.

7.3.3 Follow Along: Finding the best areas

Think back to the estate agent problem, which we last addressed in the *Vector Analysis* lesson. Let us imagine that the buyers now wish to purchase a building and build a smaller cottage on the property. In the Southern Hemisphere, we know that an ideal plot for development needs to have areas on it that:

- are north-facing
- with a slope of less than 5 degrees
- But if the slope is less than 2 degrees, then the aspect doesn't matter.

Let's find the best areas for them.

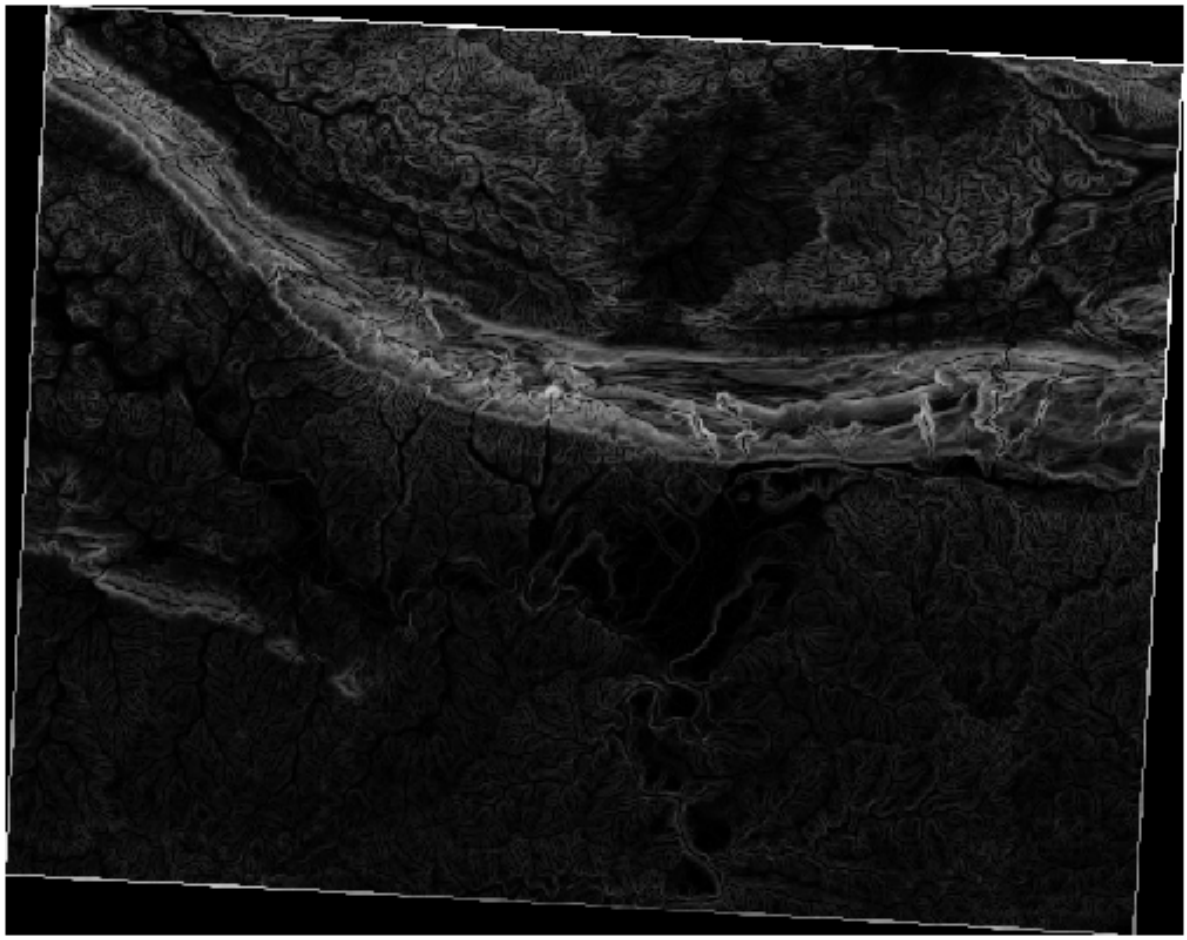
7.3.4 ??? Follow Along: Calculating the Slope

Slope informs about how steep the terrain is. If, for example, you want to build houses on the land there, then you need land that is relatively flat.

To calculate the slope, you need to use the *Slope* algorithm of the *Processing* ► *Raster terrain analysis*.

1. Open the algorithm
2. Choose *srtm_41_19* as the *Elevation layer*
3. Keep the *Z factor* at `1.0`
4. Save the output as a file with the name `slope.tif` in the same folder as the `hillshade.tif`
5. Clique em *Executar*

Now you'll see the slope of the terrain, each pixel holding the corresponding slope value. Black pixels show flat terrain and white pixels, steep terrain:



7.3.5 ??? Try Yourself: Calculating the aspect

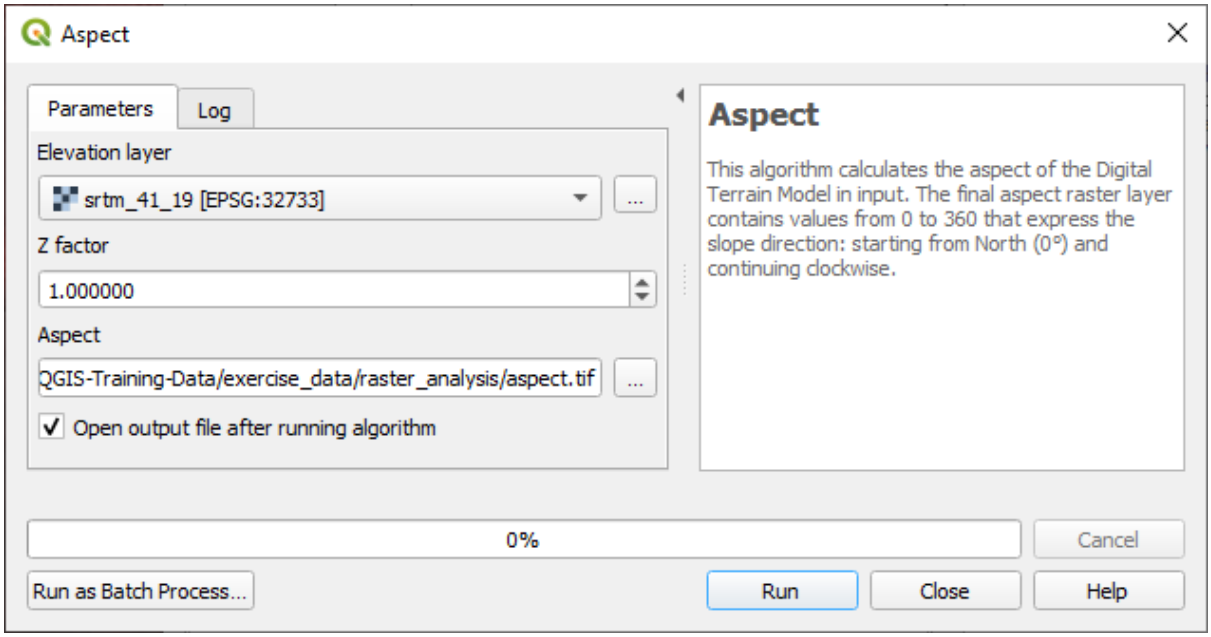
Aspect is the compass direction that the slope of the terrain faces. An aspect of 0 means that the slope is North-facing, 90 East-facing, 180 South-facing, and 270 West-facing.

Since this study is taking place in the Southern Hemisphere, properties should ideally be built on a north-facing slope so that they can remain in the sunlight.

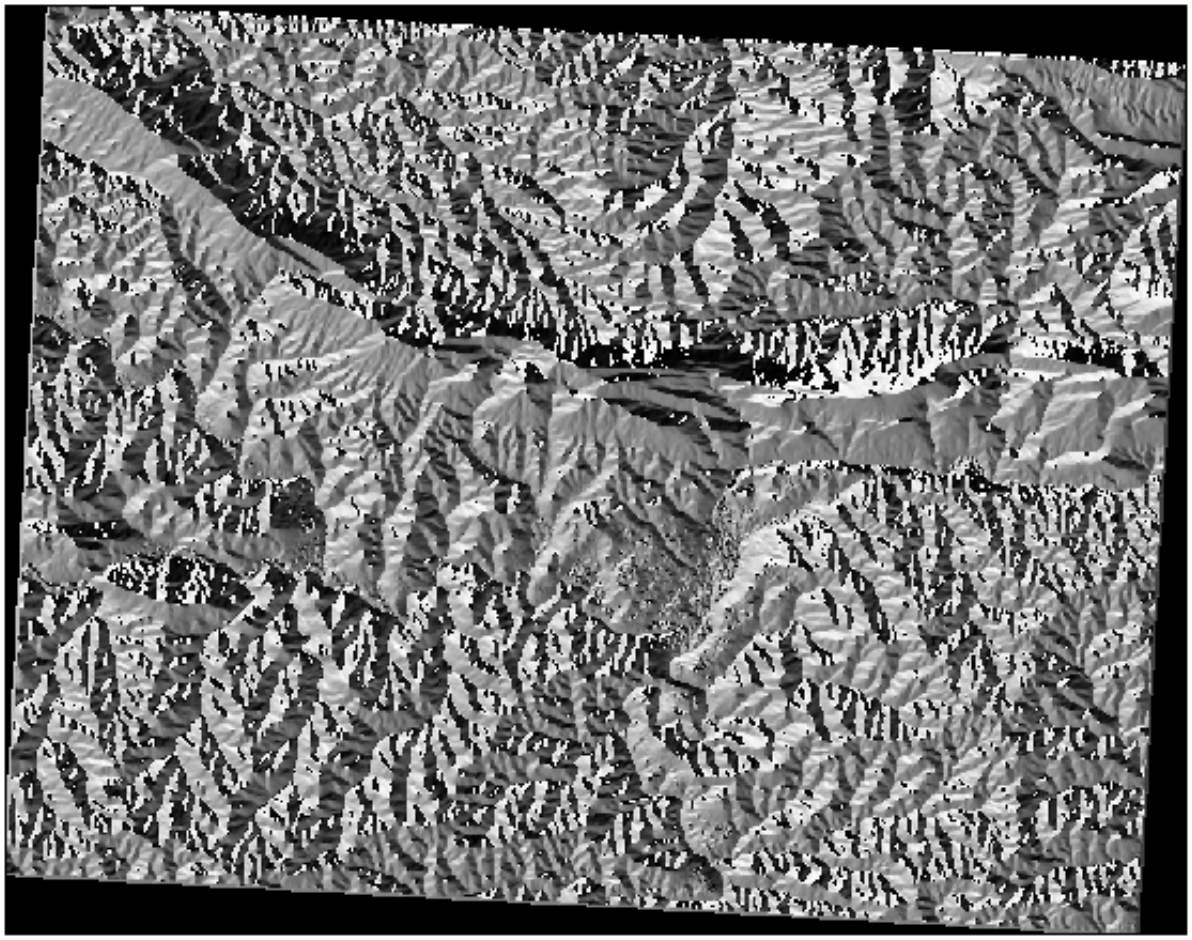
Use the *Aspect* algorithm of the *Processing* ► *Raster terrain analysis* to get the `aspect.tif` layer saved along with the `slope.tif`.

Answer

Set your *Aspect* dialog up like this:


Aspect
Parameters
Log
Elevation layer
srtm_41_19 [EPSG:32733]
Z factor
1.000000
Aspect
QGIS-Training-Data/exercise_data/raster_analysis/aspect.tif
Open output file after running algorithm
Aspect
This algorithm 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.
0%
Cancel
Run as Batch Process...
Run
Close
Help

Your result:



7.3.6 ??? Follow Along: Finding the north-facing aspect

Now, you have rasters showing you the slope as well as the aspect, but you have no way of knowing where ideal conditions are satisfied at once. How could this analysis be done?

The answer lies with the *Raster calculator*.

QGIS has different raster calculators available:

- *Raster* ► *Raster Calculator*
- In processing:
  - *Raster Analysis* ► *Raster calculator*
  - *GDAL* ► *Raster miscellaneous* ► *Raster calculator*
  - *SAGA* ► *Raster calculus* ► *Raster calculator*

Each tool is leading to the same results, but the syntax may be slightly different and the availability of operators may vary.

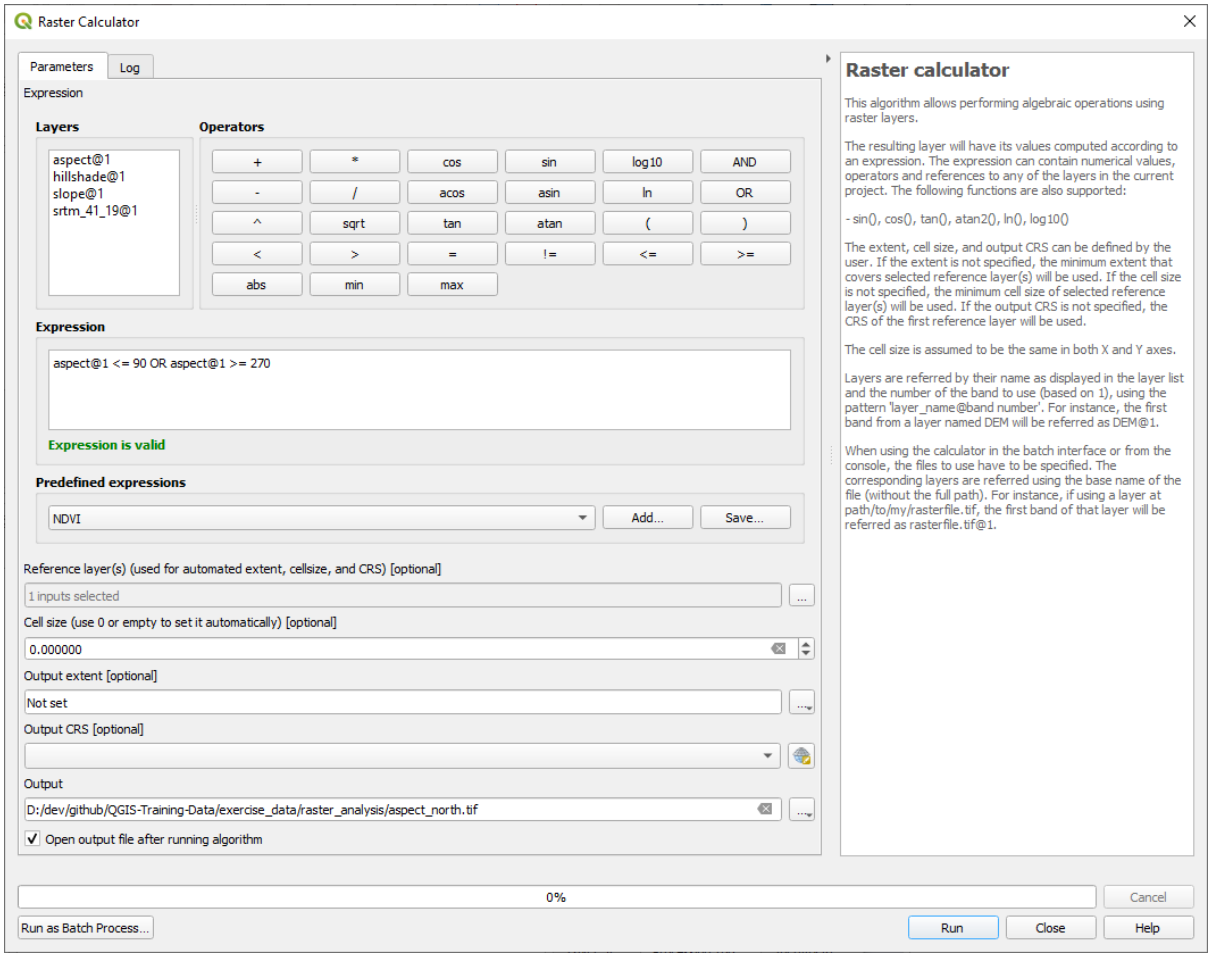
We will use *Raster Analysis* ► *Raster calculator* in the *Processing Toolbox*

1. Open the tool by double clicking on it.
   - The upper left part of the dialog lists all the loaded raster layers as `name@N`, where `name` is the name of the layer and `N` is the band.
   - In the upper right part you will see a lot of different operators. Stop for a moment to think that a raster is an image. You should see it as a 2D matrix filled with numbers.
2. North is at 0 (zero) degrees, so for the terrain to face north, its aspect needs to be greater than 270 degrees or less than 90 degrees. Therefore the formula is:

   ```
   aspect@1 <= 90 OR aspect@1 >= 270
   ```

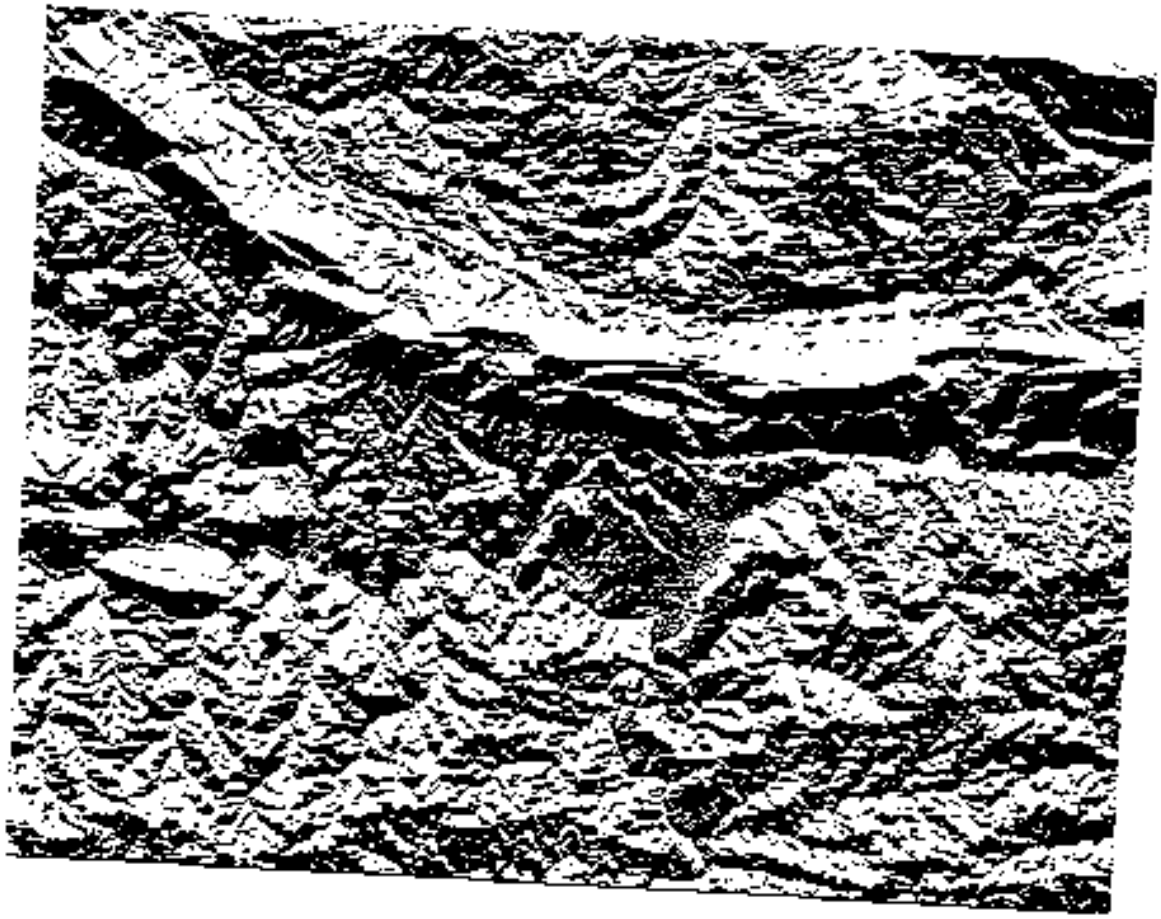
3. Now you have to set up the raster details, like the cell size, extent and CRS. This can be done manually or it can be automatically set by choosing a `Reference layer`. Choose this last option by clicking on the … button next to the *Reference layer(s)* parameter.
4. In the dialog, choose the *aspect* layer, because we want to obtain a layer with the same resolution.
5. Save the layer as `aspect_north.tif`.

   The dialog should look like:



6. Finally click on *Run*.

Your result will be this:



The output values are `0` or `1`. What does it mean? For each pixel in the raster, the formula we wrote returns whether it matches the conditions or not. Therefore the final result will be **False** (0) and **True** (1).

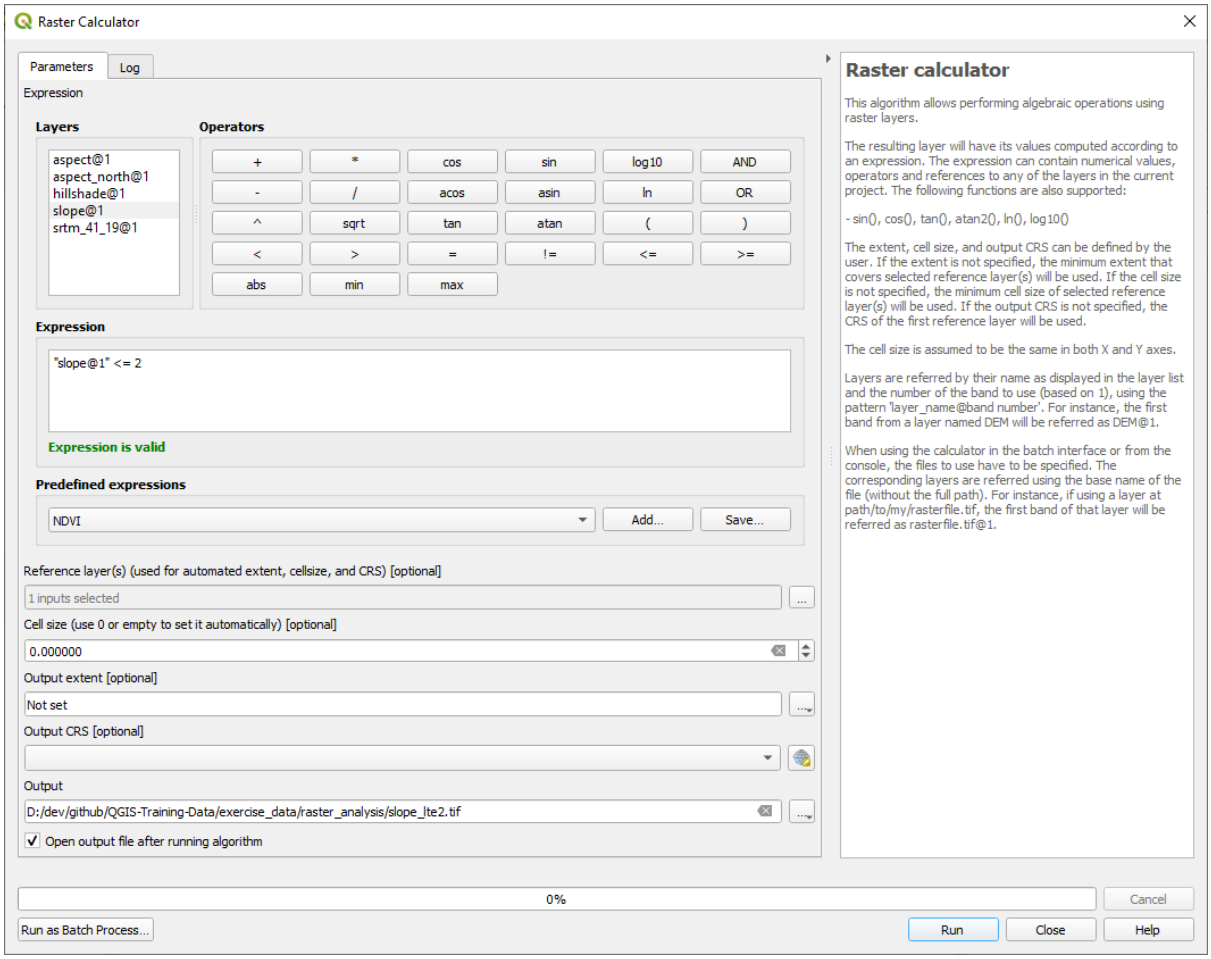
7.3.7 ??? Try Yourself: More criteria

Now that you have done the aspect, create two new layers from the DEM.

- The first shall identify areas where the slope is less than or equal to `2` degrees
- The second is similar, but the slope should be less than or equal to `5` degrees.
- Save them under `exercise_data/raster_analysis` as `slope_lte2.tif` and `slope_lte5.tif`.

Answer

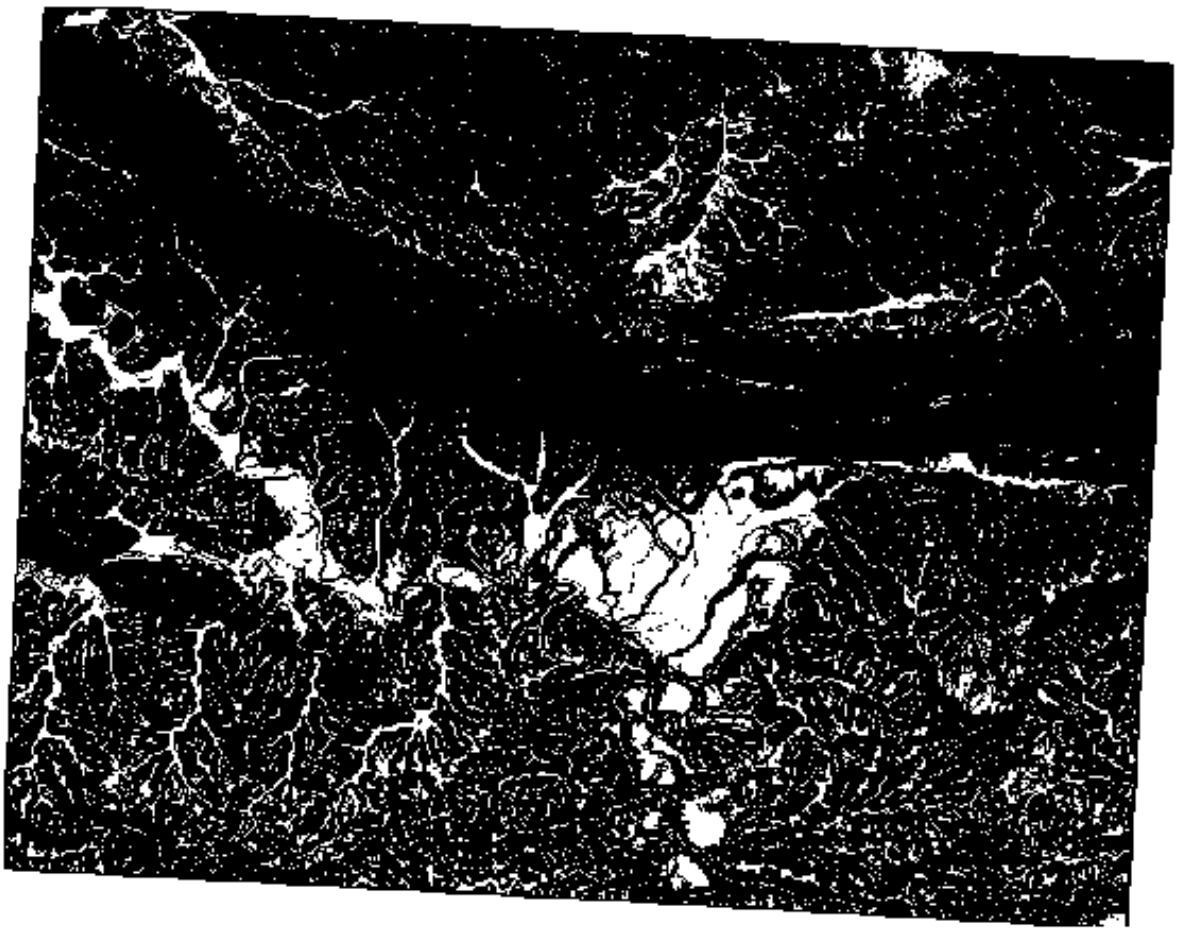
- Set your *Raster calculator* dialog up with:
- the following expression: `slope@1 <= 2`
- the `slope` layer as the *Reference layer(s)*



- For the 5 degree version, replace the `2` in the expression and file name with `5`.

Your results:

- 2 degrees:



- 5 degrees:

7.3.8 ??? Follow Along: Combining Raster Analysis Results

Now you have generated three raster layers from the DEM:

- *aspect_north*: terrain facing north
- *slope_lte2*: slope equal to or below 2 degrees
- *slope_lte5*: slope equal to or below 5 degrees

Where the condition is met, the pixel value is `1`. Elsewhere, it is `0`. Therefore, if you multiply these rasters, the pixels that have a value of `1` for all of them will get a value of `1` (the rest will get `0`).

The conditions to be met are:

- at or below 5 degrees of slope, the terrain must face north
- at or below 2 degrees of slope, the direction that the terrain faces does not matter.

Therefore, you need to find areas where the slope is at or below five degrees `AND` the terrain is facing north, `OR` the slope is at or below 2 degrees. Such terrain would be suitable for development.

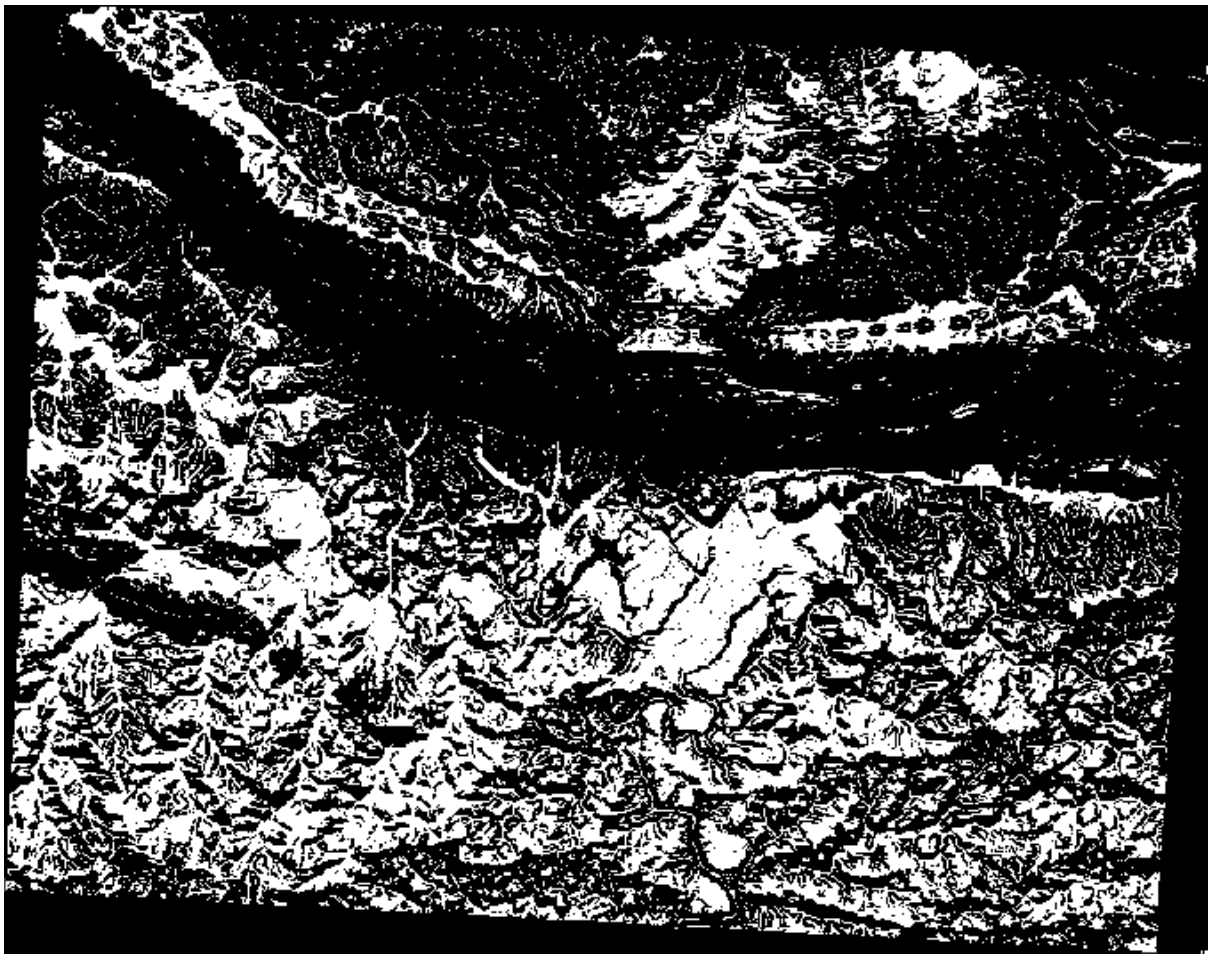
To calculate the areas that satisfy these criteria:

1. Open the *Raster calculator* again
2. Use this expression in *Expression*:

```
( aspect_north@1 = 1 AND slope_lte5@1 = 1 ) OR slope_lte2@1 = 1
```

3. Set the *Reference layer(s)* parameter to `aspect_north` (it does not matter if you choose another - they have all been calculated from `srtm_41_19`)
4. Save the output under `exercise_data/raster_analysis/` as `all_conditions.tif`
5. Click *Run*

O resultado:



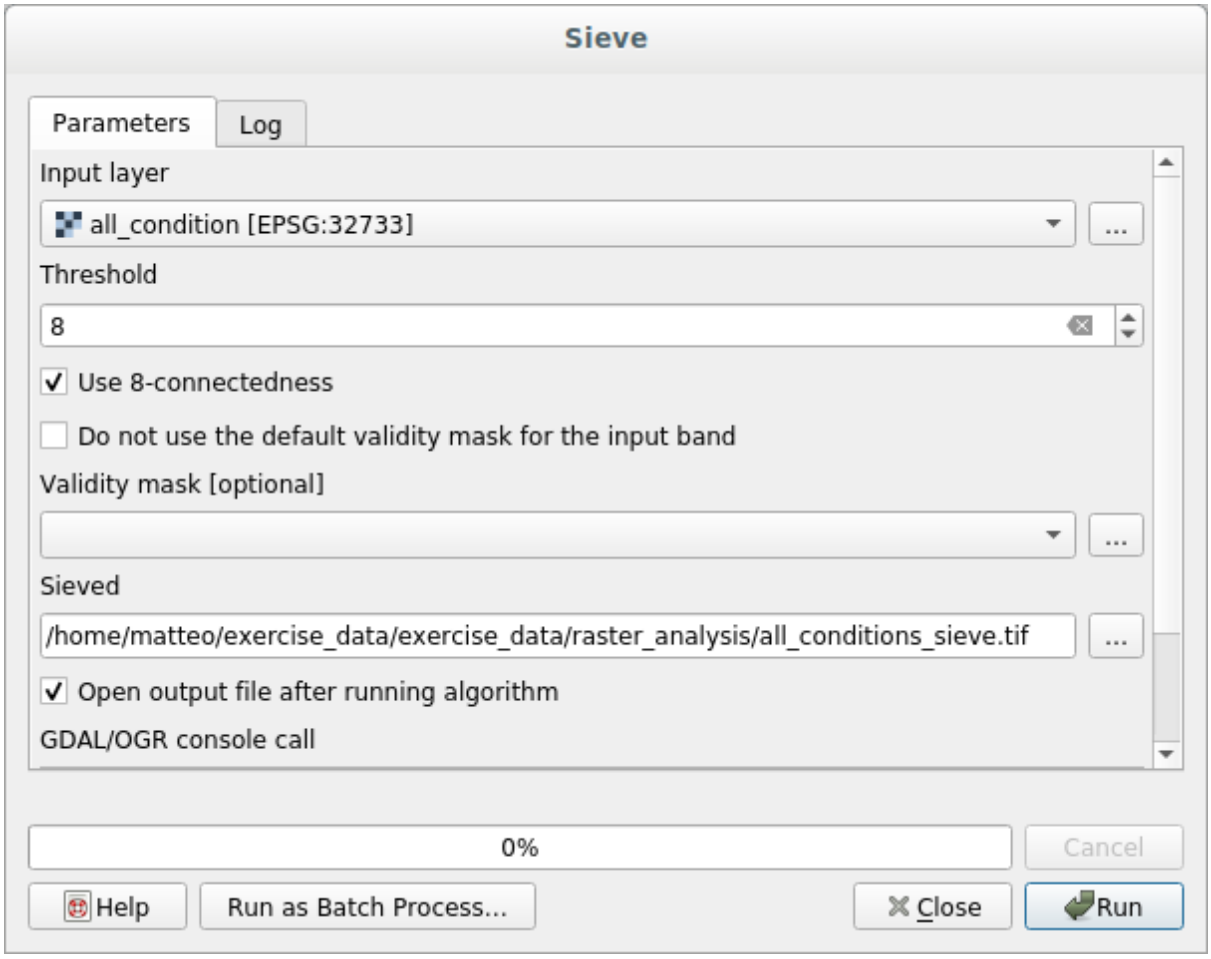
Dica: The previous steps could have been simplified using the following command:

```
((aspect@1 <= 90 OR  aspect@1 >= 270) AND slope@1 <= 5) OR slope@1 <= 2
```

7.3.9 ??? Follow Along: Simplifying the Raster

As you can see from the image above, the combined analysis has left us with many, very small areas where the conditions are met (in white). But these aren't really useful for our analysis, since they are too small to build anything on. Let us get rid of all these tiny unusable areas.

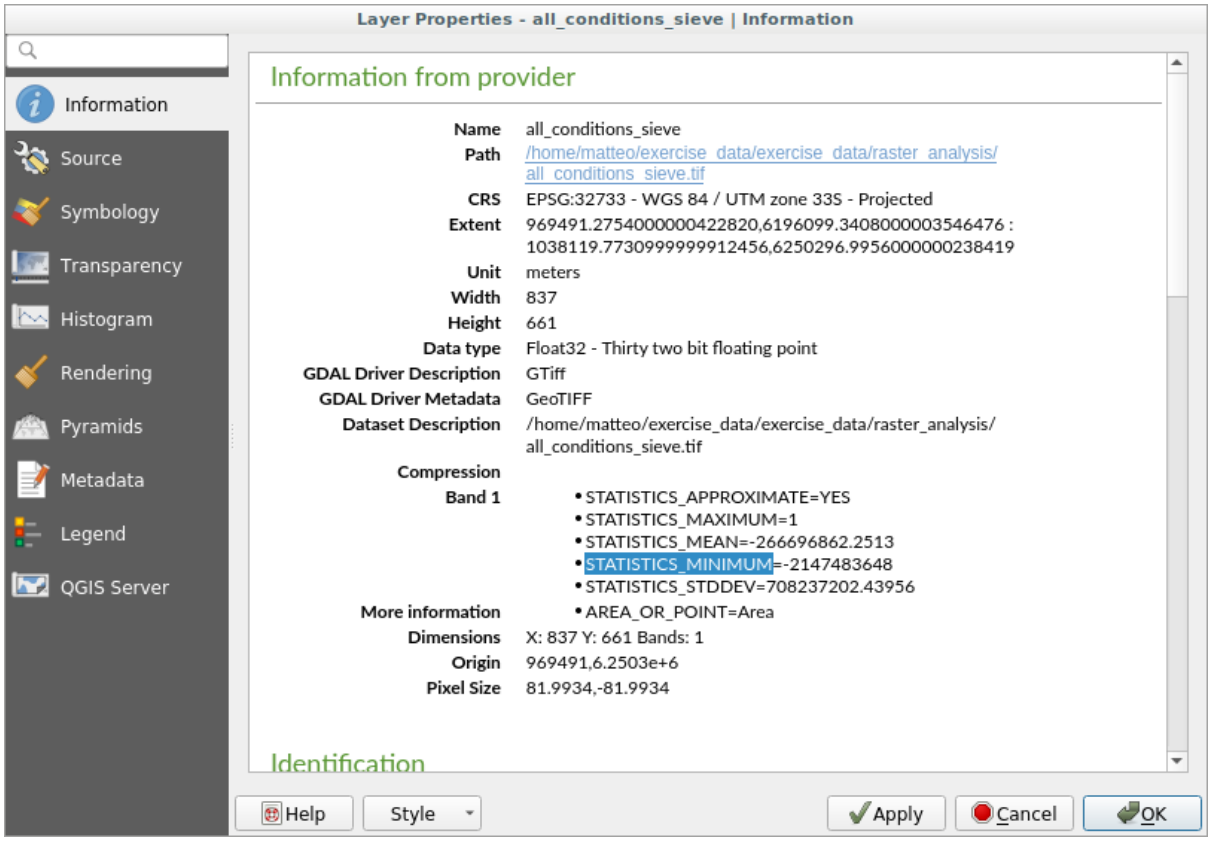
1. Open the *Sieve* tool (*GDAL* ► *Raster Analysis* in the *Processing Toolbox*)
2. Set the *Input file* to `all_conditions`, and the *Sieved* to `all_conditions_sieve.tif` (under `exercise_data/raster_analysis/`).
3. Set the *Threshold* to 8 (minimum eight contiguous pixels), and check *Use 8-connectedness*.



Once processing is done, the new layer will be loaded.

What is going on? The answer lies in the new raster file's metadata.

4. View the metadata under the *Information* tab of the *Layer Properties* dialog. Look the `STATISTICS_MINIMUM` value:



This raster, like the one it is derived from, should only feature the values `1` and `0`, but it has also a very large negative number. Investigation of the data shows that this number acts as a null value. Since we are only after areas that weren't filtered out, let us set these null values to zero.

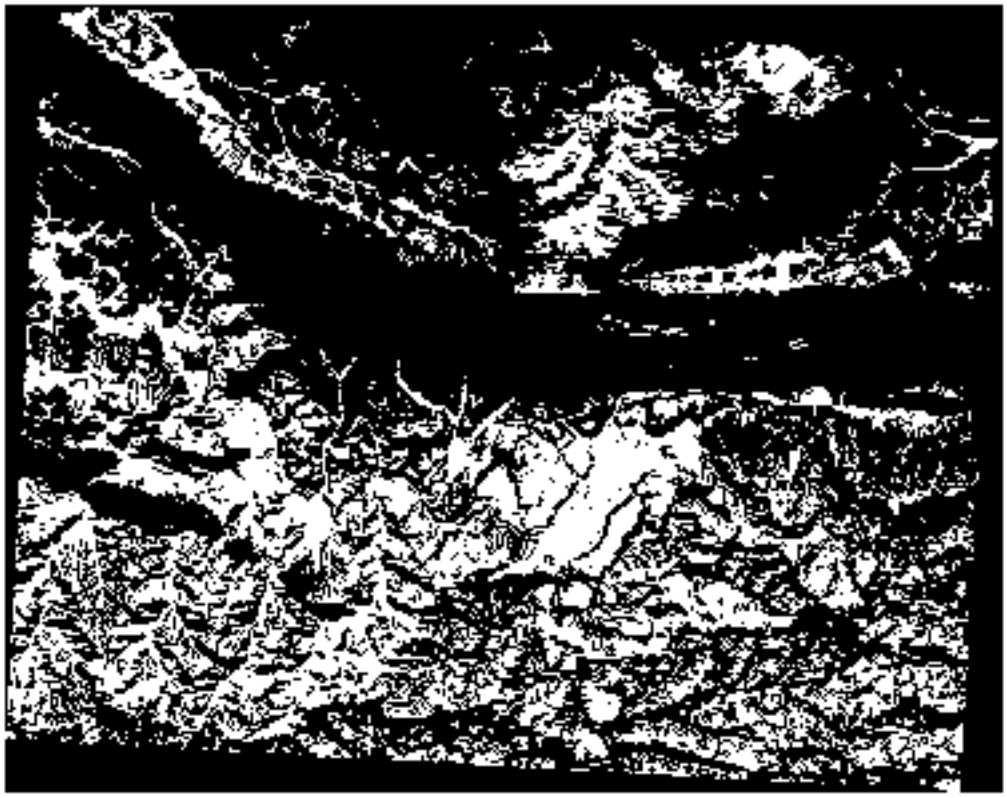
5. Open the *Raster Calculator*, and build this expression:

```
(all_conditions_sieve@1 <= 0) = 0
```

This will maintain all non-negative values, and set the negative numbers to zero, leaving all the areas with value `1` intact.

6. Save the output under `exercise_data/raster_analysis/` as `all_conditions_simple.tif`.

Your output looks like this:



This is what was expected: a simplified version of the earlier results. Remember that if the results you get from a tool aren't what you expected, viewing the metadata (and vector attributes, if applicable) can prove essential to solving the problem.

7.3.10 ??? Follow Along: Reclassifying the Raster

We have used the *Raster calculator* to do calculations on raster layers. There is another powerful tool that we can use to extract information from existing layers.

Back to the `aspect` layer. We know now that it has numerical values within a range from 0 through 360. What we want to do is to *reclassify* this layer to other discrete values (from 1 to 4), depending on the aspect:

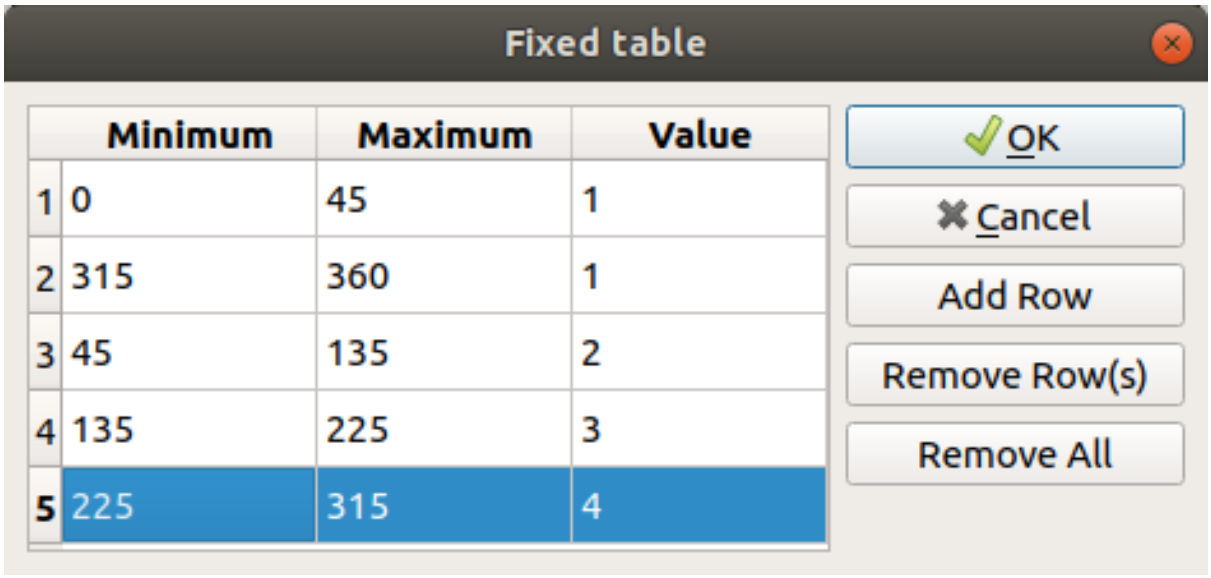
- 1 = North (from 0 to 45 and from 315 to 360);
- 2 = East (from 45 to 135)
- 3 = South (from 135 to 225)

- 4 = West (from 225 to 315)

This operation can be achieved with the raster calculator, but the formula would become very very large.

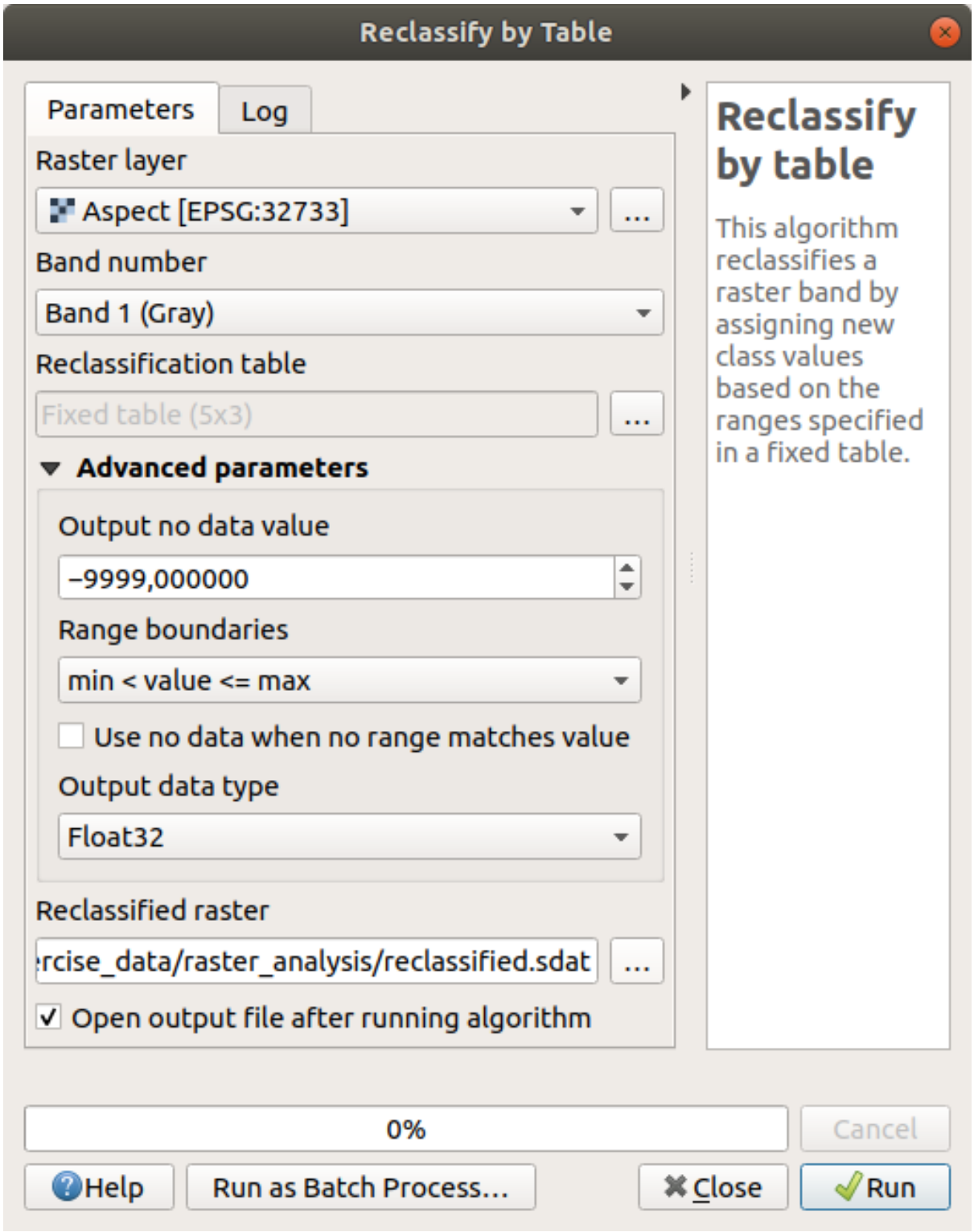
The alternative tool is the *Reclassify by table* tool in *Raster analysis* in the *Processing Toolbox*.

1. Open the tool
2. Choose *aspect* as the `Input raster layer`
3. Click on the … of *Reclassification table*. A table-like dialog will pop up, where you can choose the minimum, maximum and new values for each class.
4. Click on the *Add row* button and add 5 rows. Fill in each row as the following picture and click *OK*:



   The method used by the algorithm to treat the threshold values of each class is defined by the *Range boundaries*.

5. Save the layer as `reclassified.tif` in the `exercise_data/raster_analysis/` folder



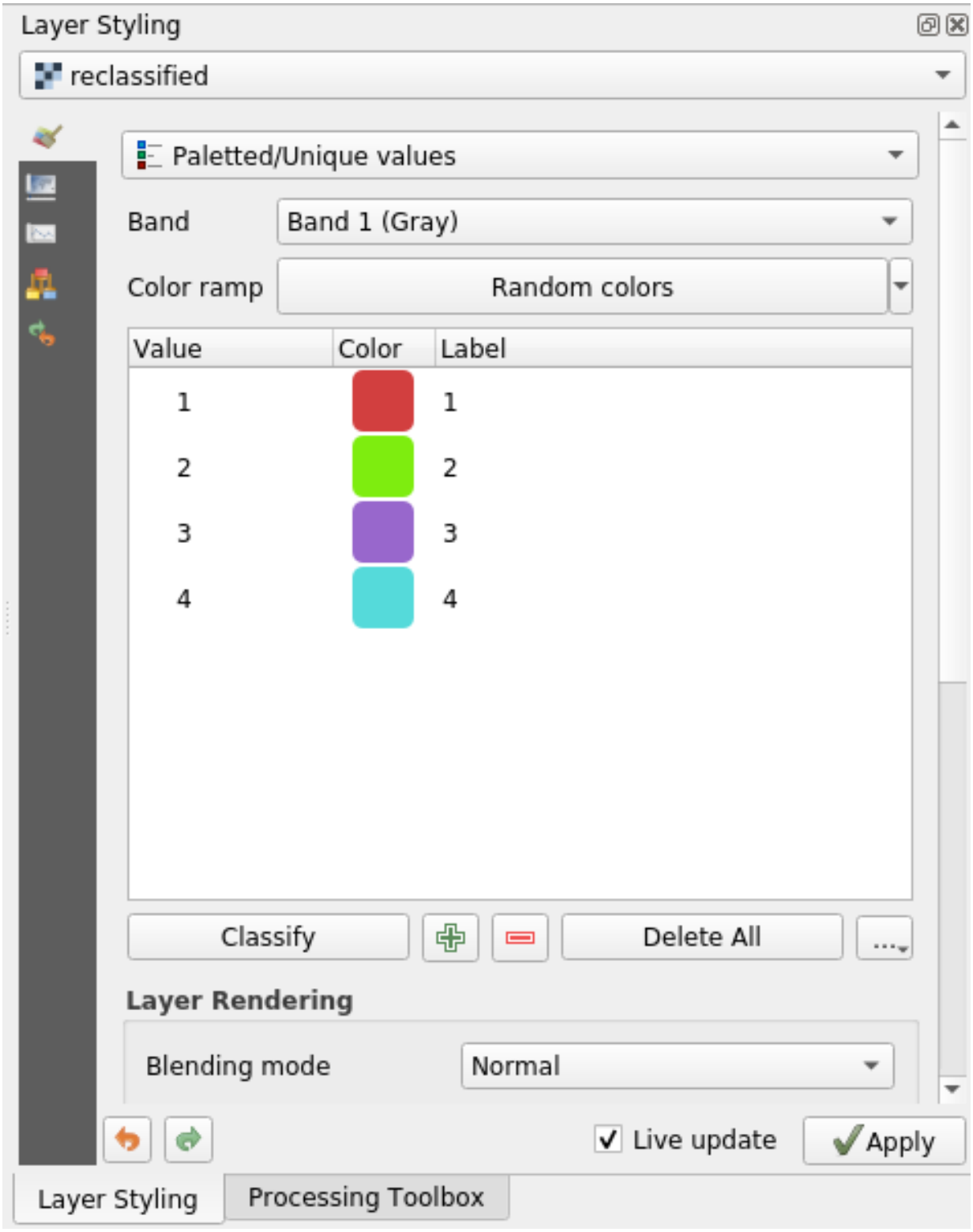
6. Clique em *Executar*

If you compare the native *aspect* layer with the *reclassified* one, there are not big differences. But by looking at the legend, you can see that the values go from `1` to `4`.

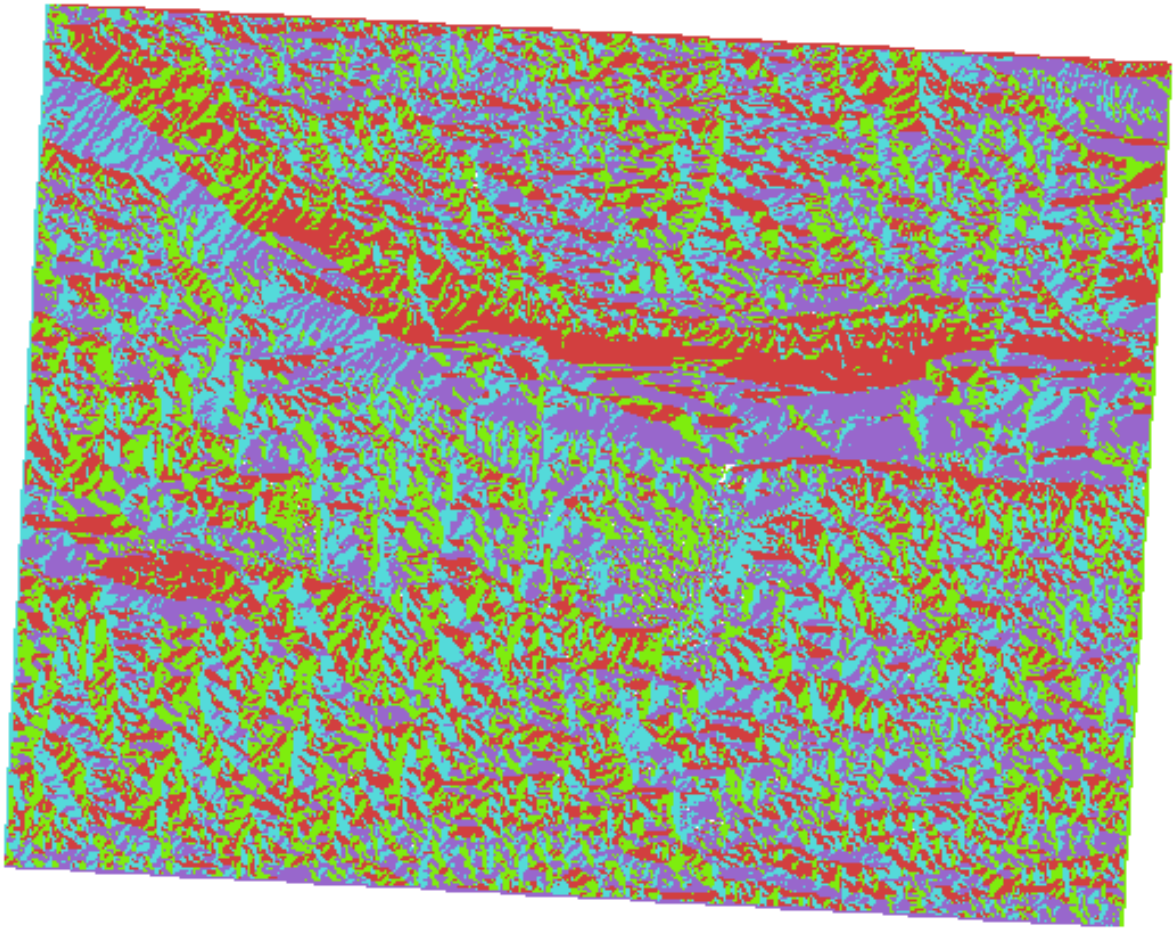
Let us give this layer a better style.

1. Open the *Layer Styling* panel
2. Choose *Paletted/Unique values*, instead of *Singleband gray*

3. Click on the *Classify* button to automatically fetch the values and assign them random colors:



The output should look like this (you can have different colors given that they have been randomly generated):



With this reclassification and the paletted style applied to the layer, you can immediately differentiate the aspect areas.

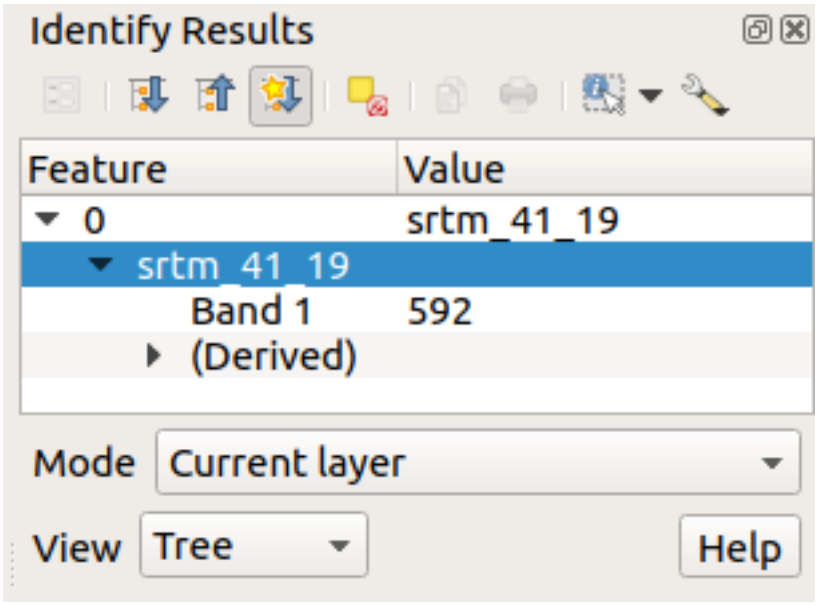
7.3.11 ??? Follow Along: Querying the raster

Unlike vector layers, raster layers don't have an attribute table. Each pixel contains one or more numerical values (singleband or multiband rasters).

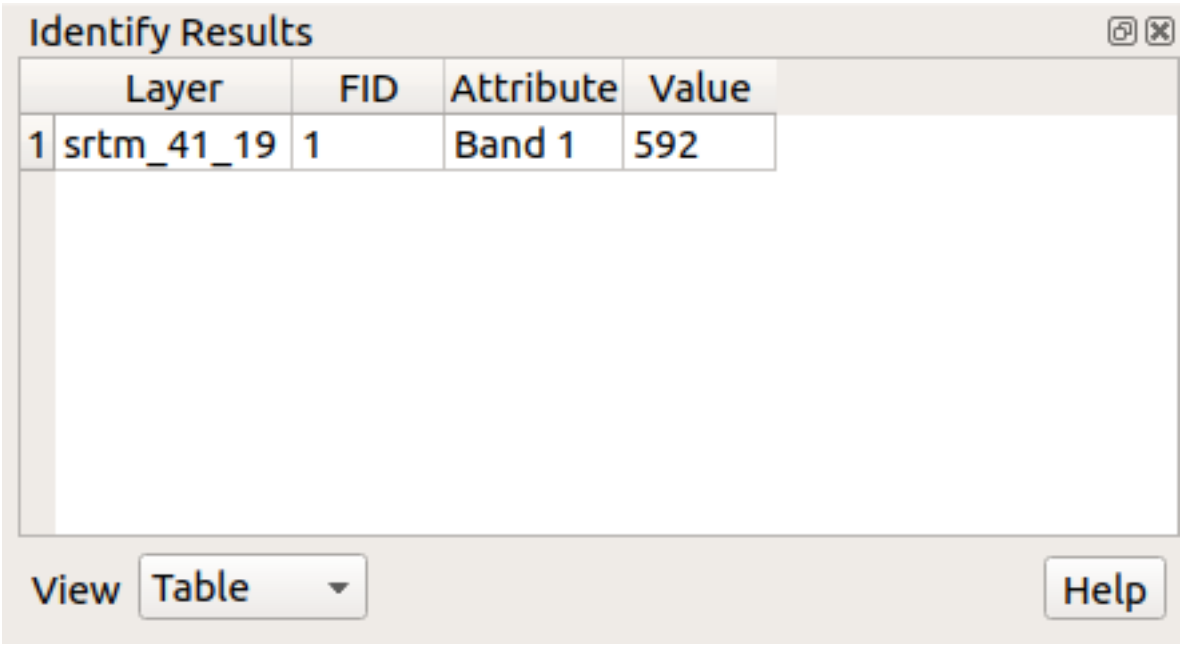
All the raster layers we used in this exercise consist of just one band. Depending on the layer, pixel values may represent elevation, aspect or slope values.

How can we query the raster layer to get the value of a pixel? We can use the Identify Features button!

1. Select the tool from the Attributes toolbar.
2. Click on a random location of the *srtm_41_19* layer. *Identify Results* will appear with the value of the band at the clicked location:

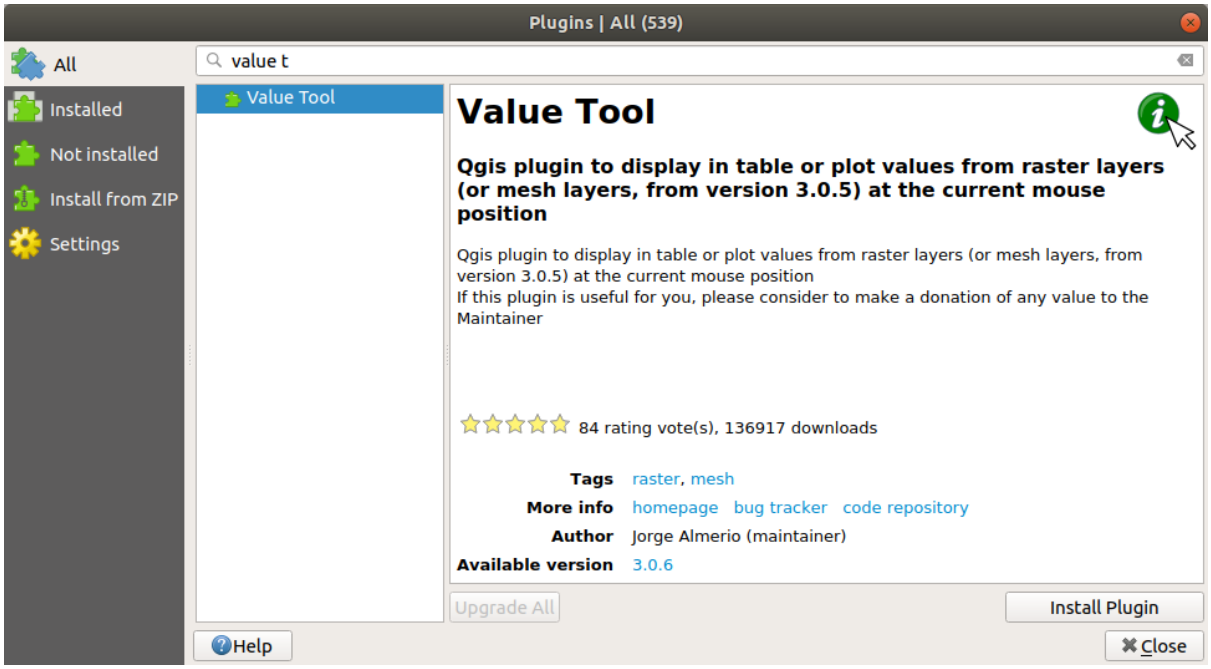


3. You can change the output of the *Identify Results* panel from the current `tree` mode to a `table` one by selecting *Table* in the *View* menu at the bottom of the panel:



Clicking each pixel to get the value of the raster could become annoying after a while. We can use the *Value Tool* plugin to solve this problem.

1. Vá a *Plugins* ► *Manage/Install Plugins...*
2. In the *All* tab, type `value t` in the search box
3. Select the *Value Tool* plugin, press *Install Plugin* and then *Close* the dialog.

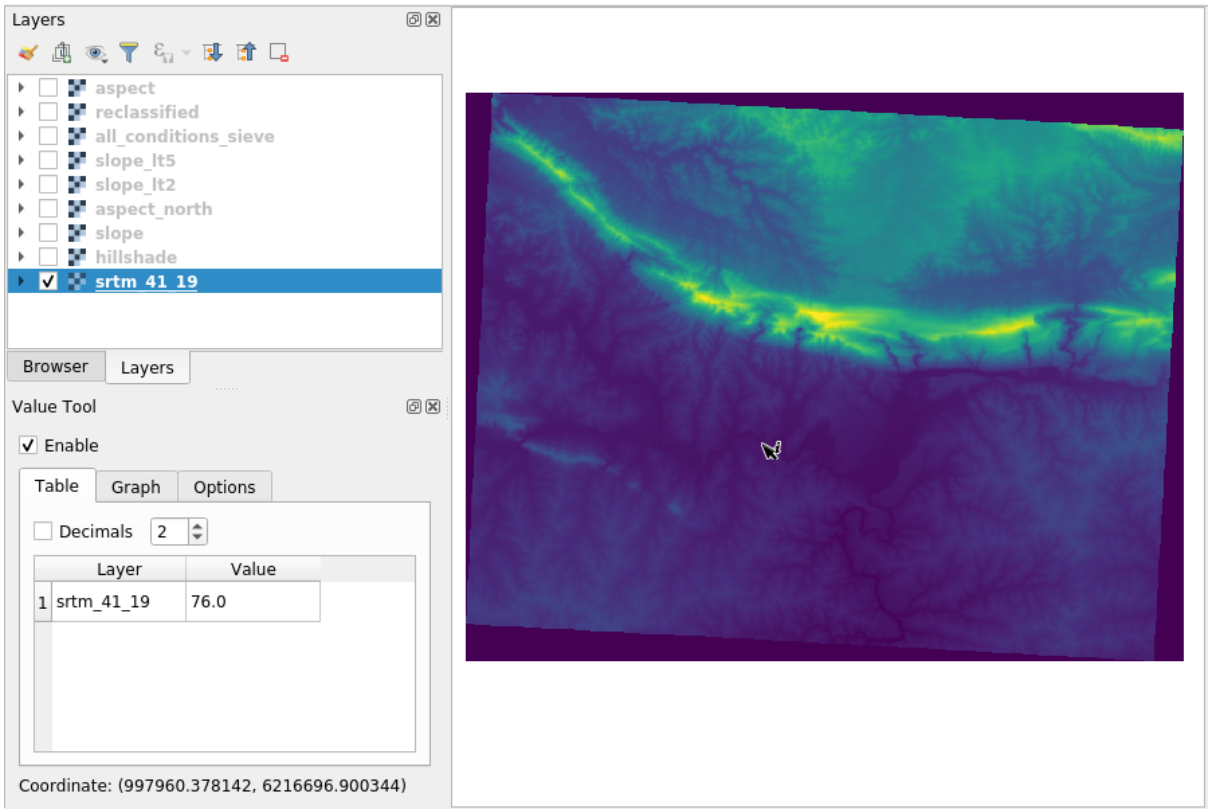


The new *Value Tool* panel will appear.

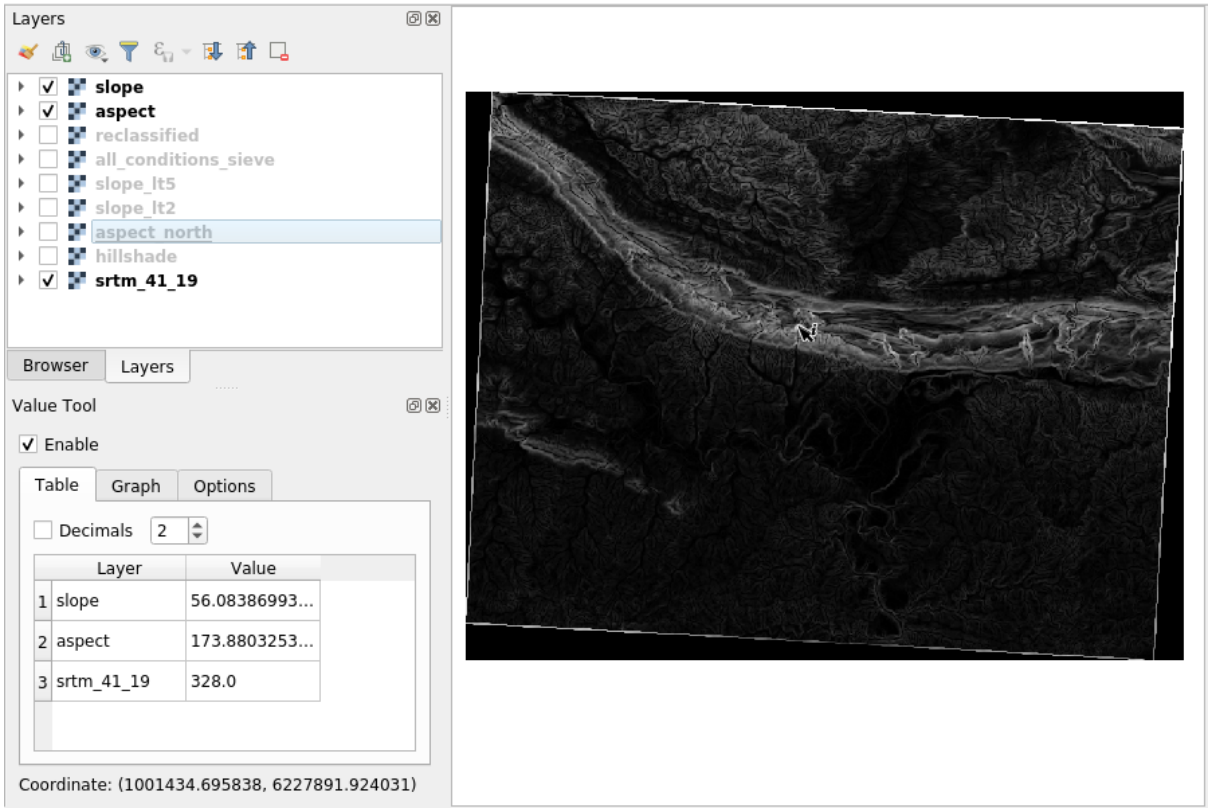
Dica: If you close the panel you can reopen it by enabling it in the *View* ► *Panels* ► *Value Tool* or by clicking on the icon in the toolbar.

---

4. To use the plugin just check the *Enable* checkbox and be sure that the `srtm_41_19` layer is active (checked) in the *Layers* panel.
5. Move the cursor over the map to see the value of the pixels.



6. But there is more. The Value Tool plugin allows you to query **all** the active raster layers in the *Layers* panel. Set the *aspect* and *slope* layers active again and hover the mouse on the map:



7.3.12 In Conclusion

You've seen how to derive all kinds of analysis products from a DEM. These include hillshade, slope and aspect calculations. You've also seen how to use the raster calculator to further analyze and combine these results. Finally you learned how to reclassify a layer and how to query the results.

7.3.13 What's Next?

Now you have two analyses: the vector analysis which shows you the potentially suitable plots, and the raster analysis that shows you the potentially suitable terrain. How can these be combined to arrive at a final result for this problem? That's the topic for the next lesson, starting in the next module.

CAPÍTULO 8

Module: Completing the Analysis

Tem agora duas partes de uma análise: uma parte vetorial e outra matricial. Neste módulo, vai ver como combiná-las. Vai concluir a análise e apresentar os resultados finais.

8.1 Lesson: Raster to Vector Conversion

A conversão entre formatos matriciais e vetoriais permite fazer uso dos dados de ambos, quando se resolve um problema SIG, bem como utilizar os vários métodos de análise, únicos a estes dois formatos de dados geográficos. Isto aumenta a flexibilidade que se tem considerando fontes de dados e métodos de processamento para resolver um problema SIG.

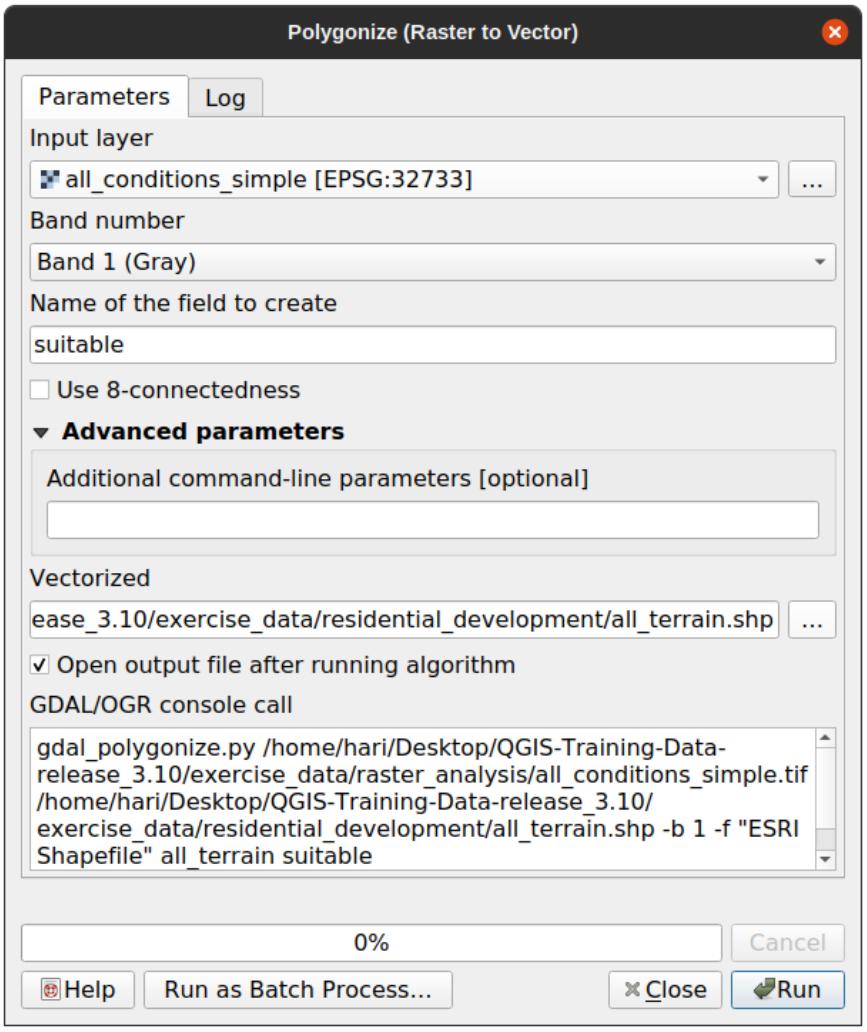
Para combinar uma análise matricial e vetorial, precisa de converter um tipo de dados para o outro. Vamos converter o resultado matricial, da lição anterior, para um vetor.

O objetivo para esta lição: Obter o resultado matricial num vetorial que pode ser utilizado para completar a análise.

8.1.1 ??? Follow Along: The *Raster to Vector* Tool

Abra o mapa do último módulo, `raster_analysis.qgs`. Aí deve estar o ficheiro `all_conditions_simple.tif` calculado no exercício anterior.

- Clique em *Raster* ► *Conversão* ► *Vetorizar (Raster to Vector)*. A caixa de diálogo aparecerá.
- Configure-o assim:



- Change the field name (describing the values of the raster) to *suitable*.
- Save the layer under `exercise_data/residential_development` as `all_terrain.shp`.

Now you have a vector file which contains all the values of the raster, but the only areas you're interested in are those that are suitable; i.e., those polygons where the value of *suitable* is `1`. You can change the style of this layer if you want to have a clearer visualization of it.

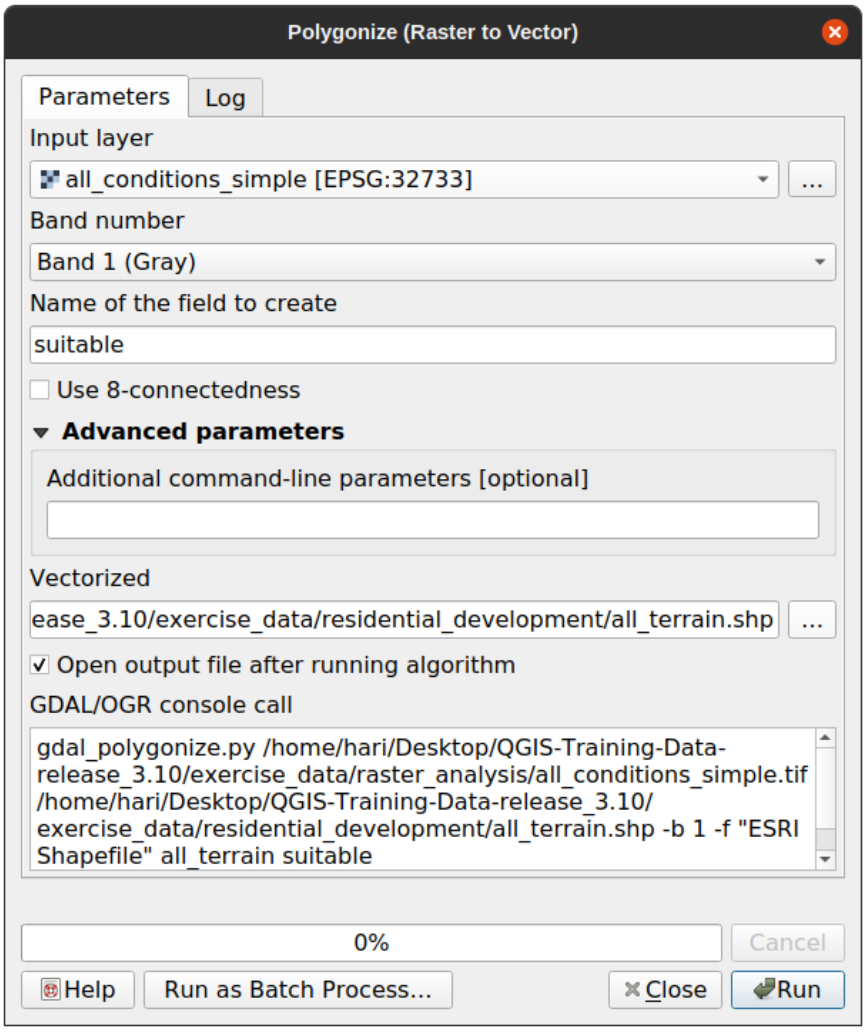
8.1.2 ??? Try Yourself:

Recorra ao módulo sobre análise vetorial.

- Create a new vector file that contains only the polygons where *suitable* has the value of `1`.
- Save the new file under `exercise_data/residential_development/` as `suitable_terrain.shp`.

Answer

1. Open the *Query Builder* by right-clicking on the *all_terrain* layer in the *Layers* panel, and selecting the *Properties* ► *Source* tab.
2. Then build the query `"suitable" = 1`.
3. Click *OK* to filter out all the polygons where this condition isn't met. When viewed over the original raster, the areas should overlap perfectly:

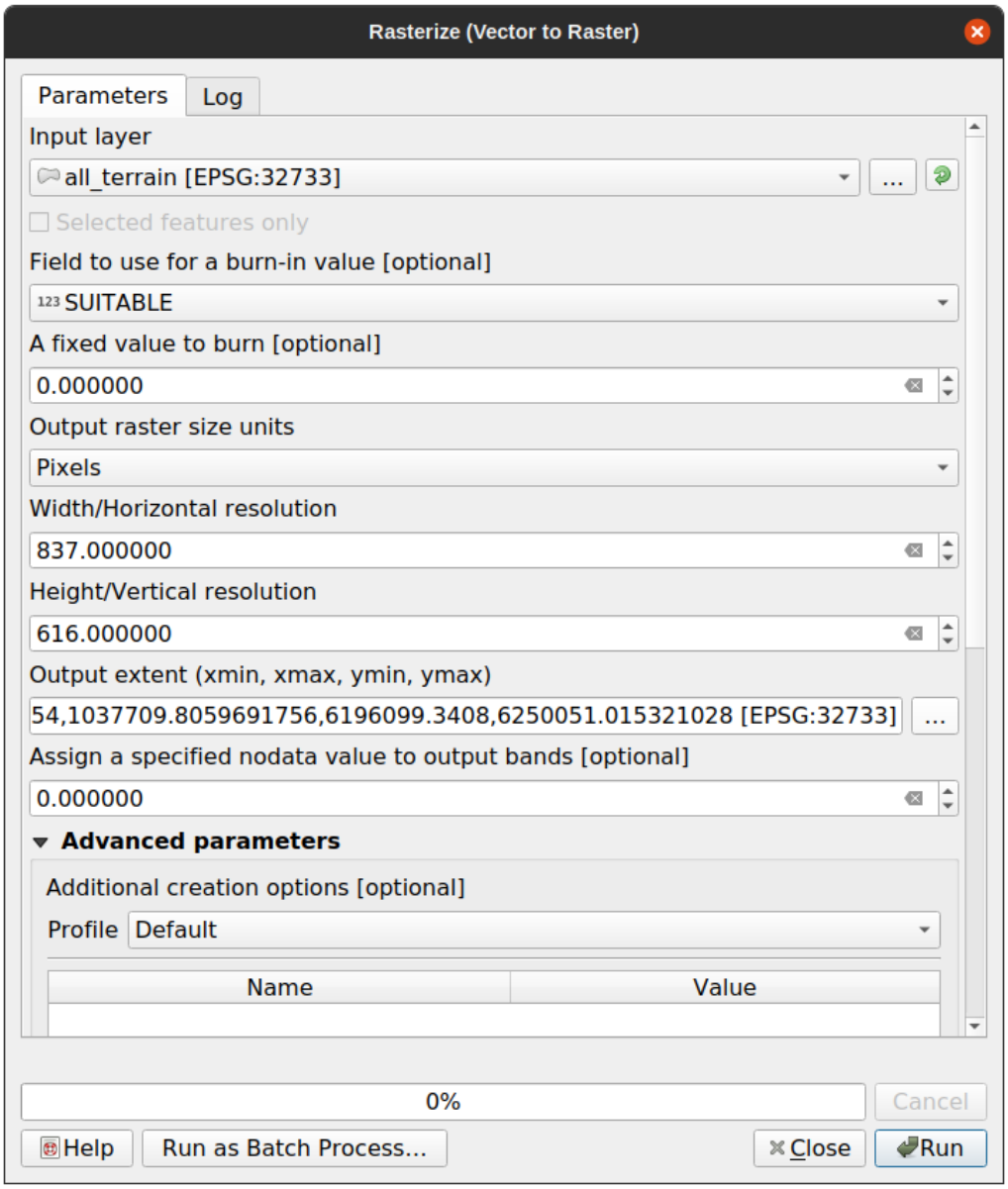


4. You can save this layer by right-clicking on the *all_terrain* layer in the *Layers* panel and choosing *Save As…*, then continue as per the instructions.

8.1.3 Follow Along: The *Vector to Raster* Tool

Although unnecessary for our current problem, it's useful to know about the opposite conversion from the one performed above. Convert to raster the `suitable_terrain.shp` vector file you just created in previous step.

- Clique em *Raster* ► *Conversão* ► *Rasterizar (Vector para Raster)* para iniciar esta ferramenta, depois configure-a como indicado na seguinte captura de ecrã:



- *Input layer* is *all_terrain*.
- Field name is *suitable*.
- *Output raster size units* is *Pixels*.
- *Width* and *Height* are `837` and `661`, respectively.
- Get the *Output extent* from the *all_terrain* layer.
- Set output file *Rasterized* to `exercise_data/residential_development/raster_conversion.tif`.

Nota: O tamanho da imagem de saída é definido aqui para ser igual ao da imagem original que foi vetorizada. Para ver as dimensões de uma imagem, abra os seus metadados (no separador *Metadados* em :guilabel:`Propriedades da camada).

- Clique em *OK* na caixa de diálogo para iniciar o processo de conversão.
- Quando estiver completo, avalie o seu sucesso comparando o novo dado matricial com o original. Eles devem corresponder, exatamente, pixel por pixel.

8.1.4 In Conclusion

Converter entre formatos matriciais e vetoriais permite ampliar a aplicabilidade dos dados e não precisa levar à degradação dos dados.

8.1.5 What's Next?

Agora que temos os resultados da análise do relevo disponíveis em formato vetorial, podem ser utilizados para resolver o problema de quais os edifícios que devem ser considerados para o desenvolvimento residencial.

8.2 Lesson: Combining the Analyses

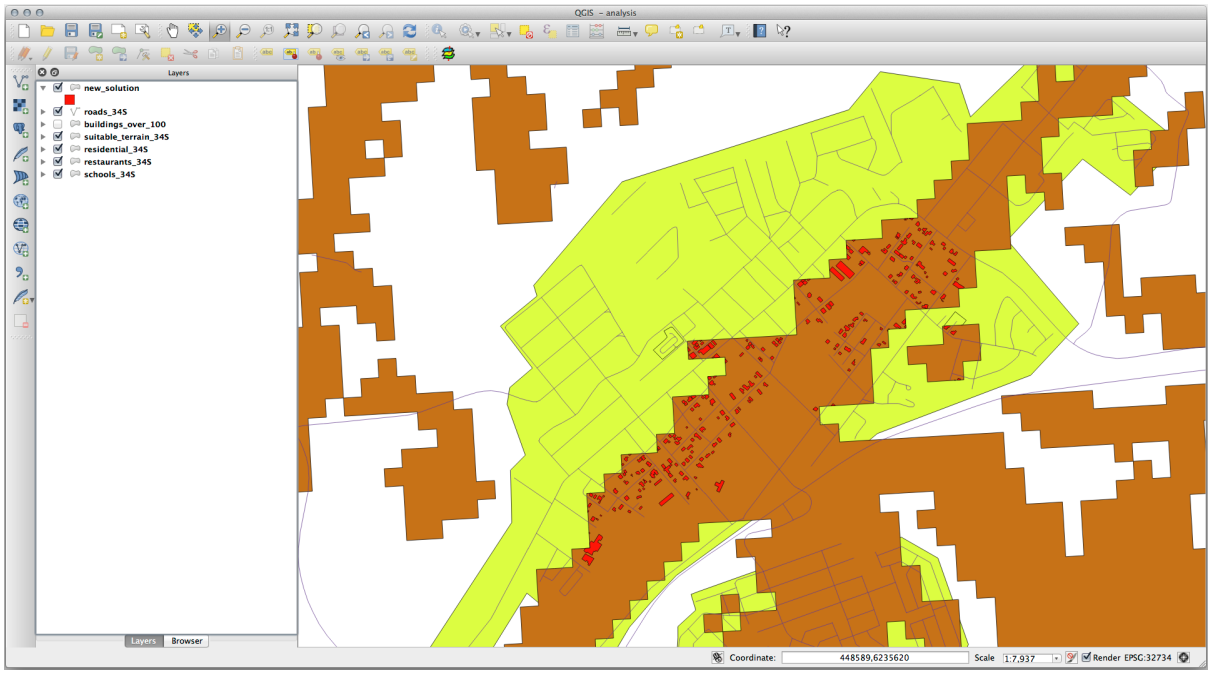
Using the vectorized results of the raster analysis will allow you to select only those buildings on suitable terrain.

The goal for this lesson: To use the vectorized terrain results to select suitable plots.

8.2.1 ??? Try Yourself:

1. Save your current map (`raster_analysis.qgs`).
2. Open the map which you created during the vector analysis earlier (you should have saved the file as `analysis.qgs`).
3. In the *Layers* panel, enable these layers:
   - *hillshade*,
   - *solution* (or *buildings_over_100*)
4. In addition to these layers, which should already be loaded in the map from when you worked on it before, also add the `suitable_terrain.shp` dataset.
5. If you are missing some layers, you should find them in `exercise_data/residential_development/`
6. Use the *Intersection* tool (*Vector* ► *Geoprocessing Tools*) to create a new vector layer called `new_solution.shp` which contains only those buildings which intersect the *suitable_terrain* layer.

You should now have a layer showing certain buildings as your solution, for example:



8.2.2 Try Yourself: Inspecting the Results

Look at each of the buildings in your *new_solution* layer. Compare them with the *suitable_terrain* layer by changing the symbology for the *new_solution* layer so that it has outlines only. What do you notice about some of the buildings? Are they all suitable just because they intersect with the *suitable_terrain* layer? Why or why not? Which ones would you deem to be unsuitable?

Answer

You may notice that some of the buildings in your `new_solution` layer have been «sliced» by the *Intersection* tool. This shows that only part of the building - and therefore only part of the property - lies on suitable terrain. We can therefore sensibly eliminate those buildings from our dataset.

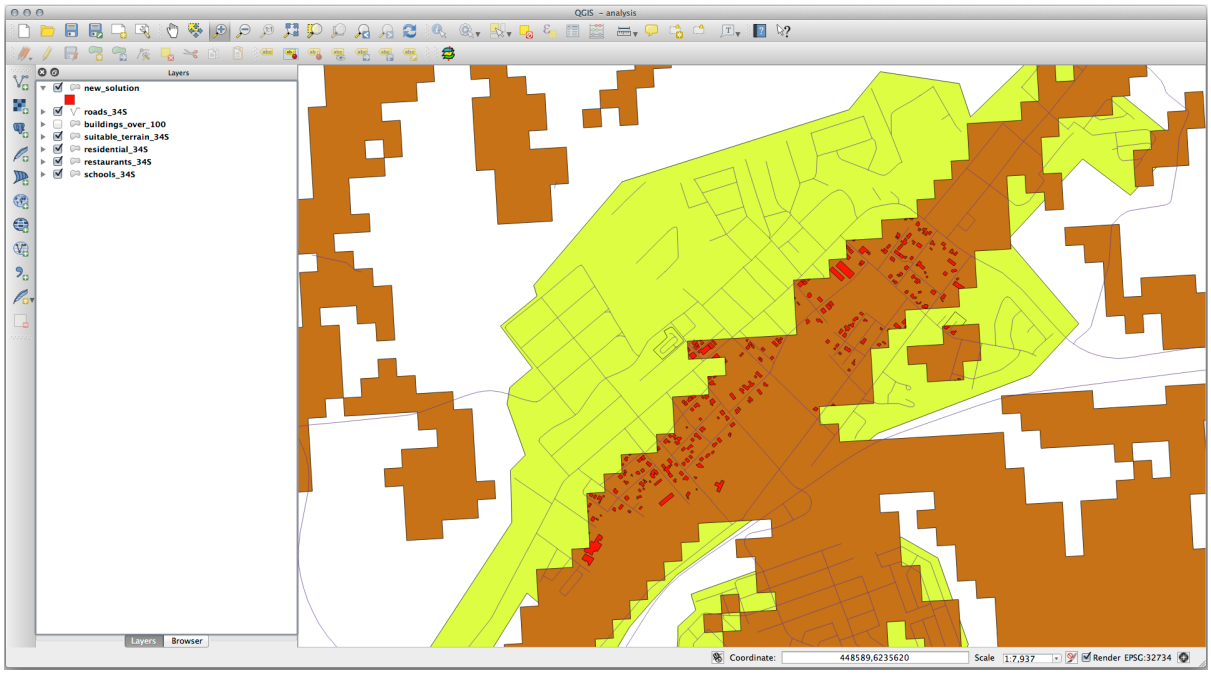
8.2.3 Try Yourself: Refining the Analysis

You can see from the results that some buildings which were included were not really suitable, so we can now refine the analysis.

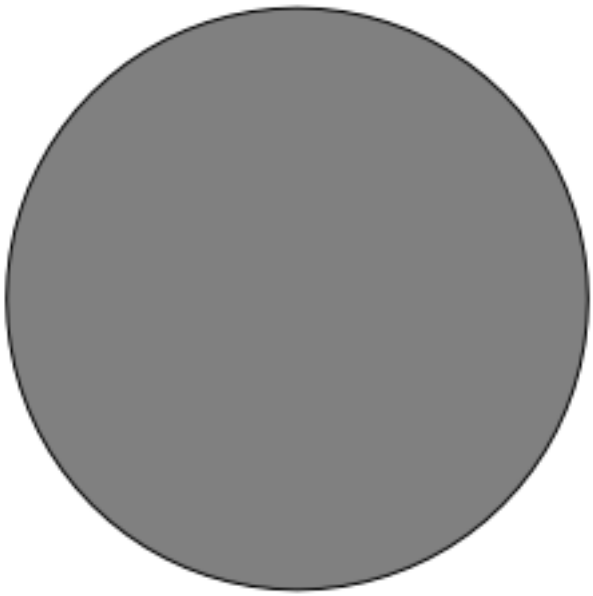
We want to ensure that our analysis returns only those buildings which fall entirely within the *suitable_terrain* layer. How would you achieve this? Use one or more Vector Analysis tools and remember that our buildings are all over 100m squared in size.

Answer

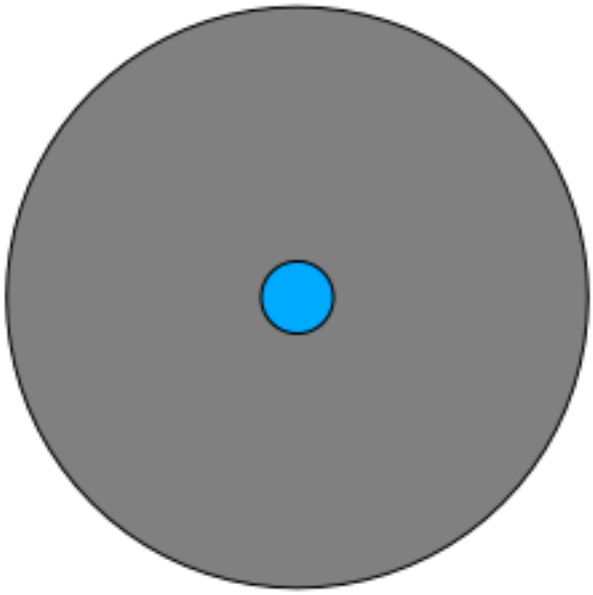
At the moment, your analysis should look something like this:



Consider a circular area, continuous for 100 meters in all directions.



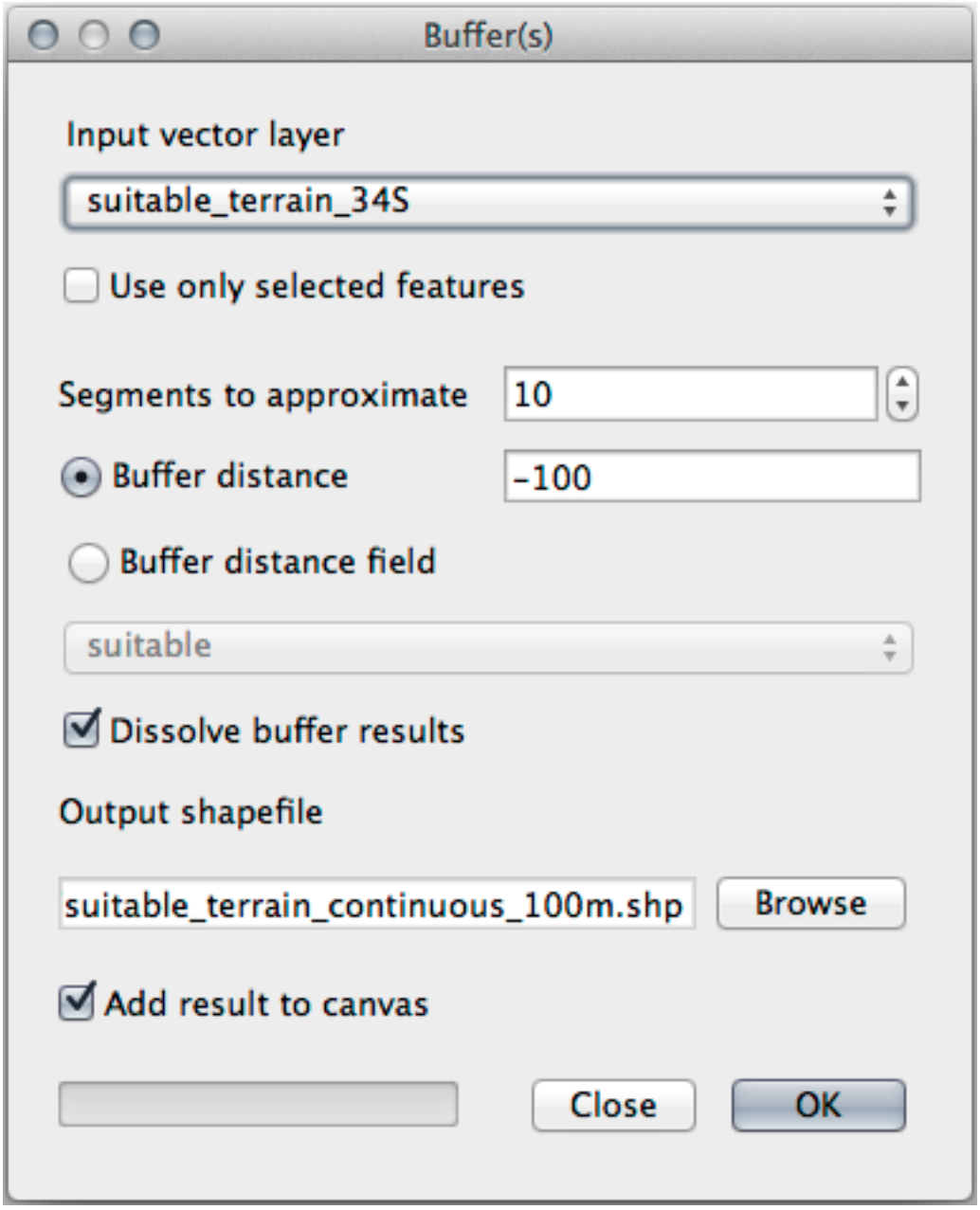
If it is greater than 100 meters in radius, then subtracting 100 meters from its size (from all directions) will result in a part of it being left in the middle.



Therefore, you can run an *interior buffer* of 100 meters on your existing *suitable_terrain* vector layer. In the output of the buffer function, whatever remains of the original layer will represent areas where there is suitable terrain for 100 meters beyond.

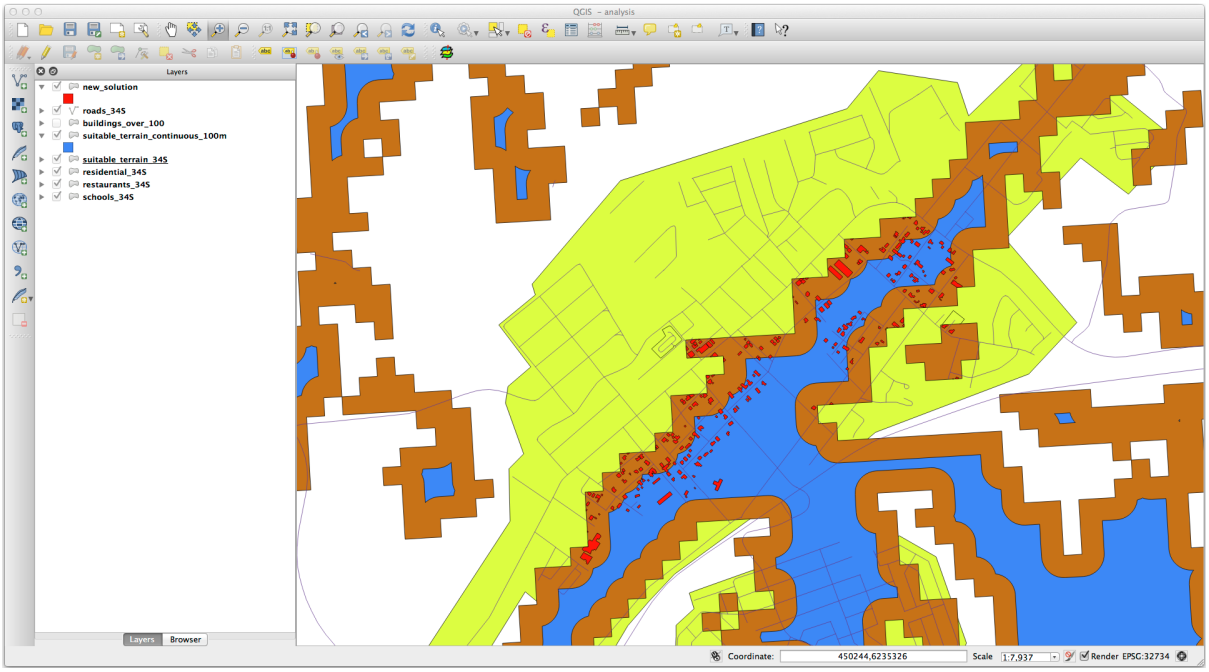
To demonstrate:

1. Go to *Vector* ► *Geoprocessing Tools* ► *Buffer(s)* to open the Buffer(s) dialog.
2. Configure-o assim:

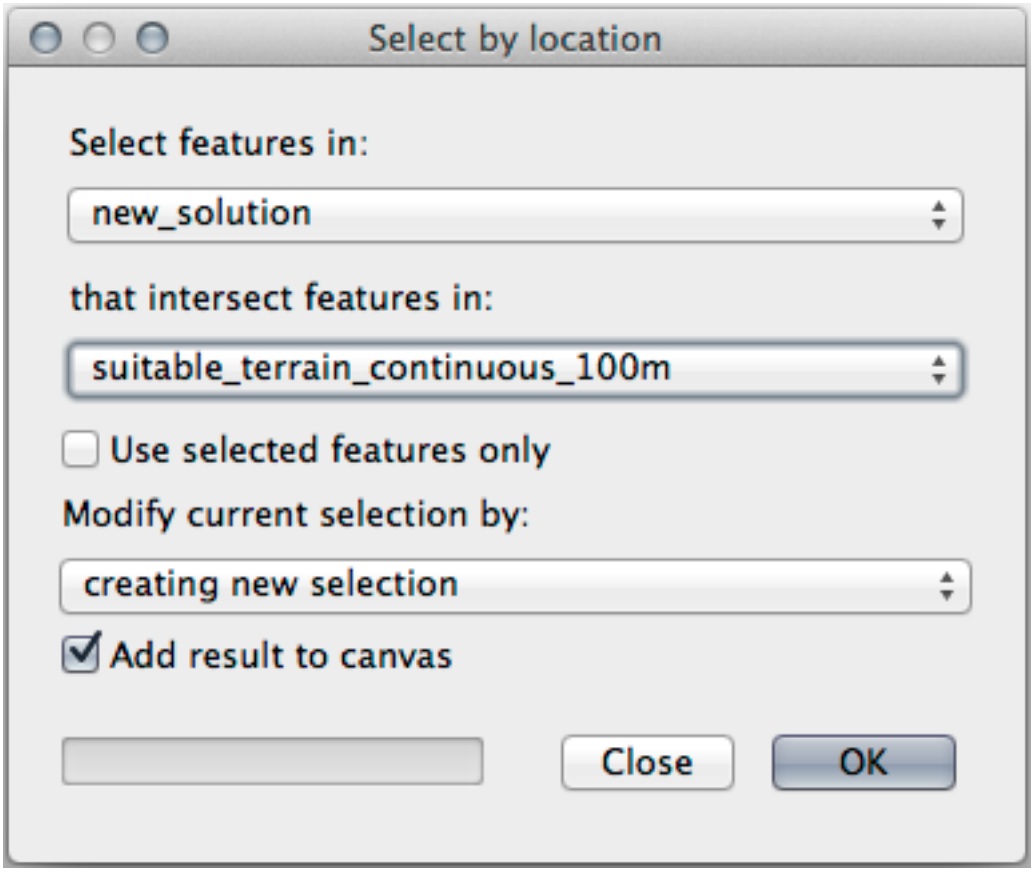


3. Use the *suitable_terrain* layer with `10` segments and a buffer distance of `-100`. (The distance is automatically in meters because your map is using a projected CRS.)
4. Save the output in `exercise_data/residential_development/` as `suitable_terrain_continuous100m.shp`.
5. If necessary, move the new layer above your original *suitable_terrain* layer.

   Your results will look like something like this:

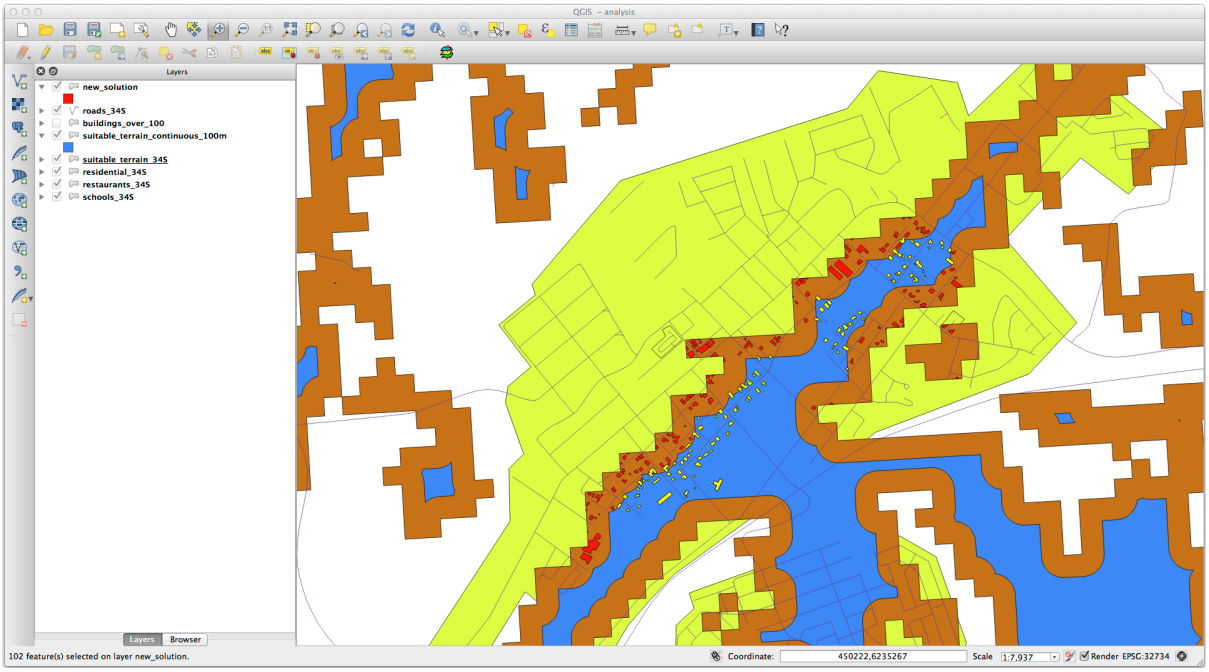


6. Now use the *Select by Location* tool (*Vector* ► *Research Tools* ► *Select by location*).
7. Set up like this:



8. Select features in *new_solution* that intersect features in *suitable_terrain_continuous100m.shp*.

   This is the result:



The yellow buildings are selected. Although some of the buildings fall partly outside the new *suitable_terrain_continuous100m* layer, they lie well within the original *suitable_terrain* layer and therefore meet all of our requirements.

9. Save the selection under `exercise_data/residential_development/` as `final_answer.shp`.

8.2.4 In Conclusion

You have now answered the original research question, and can offer an opinion (with reasons, backed by analysis) for a recommendation regarding which property to develop.

8.2.5 What's Next?

Next you will present these results as part of your second assignment.

8.3 Tarefa

Using the print layout, make a new map representing the results of your analysis. Include these layers:

- *places* (with labels),
- *hillshade*,
- *solution* (or *new_solution*),
- *roads* and
- either *aerial_photos* or *DEM*.

Write a short explanatory text to accompany it. Include in this text the criteria that were used in considering a house for purchase and subsequent development, as well as explaining your recommendations for which buildings are suitable.

8.4 Lesson: Supplementary Exercise

In this lesson, you will be guided through a complete GIS analysis in QGIS.

Nota: Lesson developed by Linfiniti Consulting (South Africa) and Siddique Motala (Cape Peninsula University of Technology)

8.4.1 Problem Statement

You are tasked with finding areas in and around the Cape Peninsula that are suitable habitats for a rare fynbos plant species. The extent of your area of investigation covers Cape Town and the Cape Peninsula between Melkbosstrand in the north and Strand in the south. Botanists have provided you with the following preferences exhibited by the species in question:

- It grows on east facing slopes
- It grows on slopes with a gradient between 15% and 60%
- It grows in areas that have a total annual rainfall of > 1000 mm
- It will only be found at least 250 m away from any human settlement
- The area of vegetation in which it occurs should be at least 6000 m^2 in area

As a student at the University, you have agreed to search for the plant in four different suitable areas of land. You want those four suitable areas to be the ones that are closest to the University of Cape Town where you live. Use your GIS skills to determine where you should go to look.

8.4.2 Solution Outline

The data for this exercise can be found in the `exercise_data/more_analysis` folder.

You are going to find the four suitable areas that are closest to the University of Cape Town.

The solution will involve:

1. Analyzing a DEM raster layer to find the east facing slopes and the slopes with the correct gradients
2. Analyzing a rainfall raster layer to find the areas with the correct amount of rainfall
3. Analyzing a zoning vector layer to find areas that are away from human settlement and are of the correct size

8.4.3 Follow Along: Setting up the Map

1. Click on the Current CRS button in the lower right corner of the screen. Under the *CRS* tab of the dialog that appears, use the «Filter» tool to search for «33S». Select the entry *WGS 84 / UTM zone 33S* (with EPSG code `32733`).
2. Click *OK*

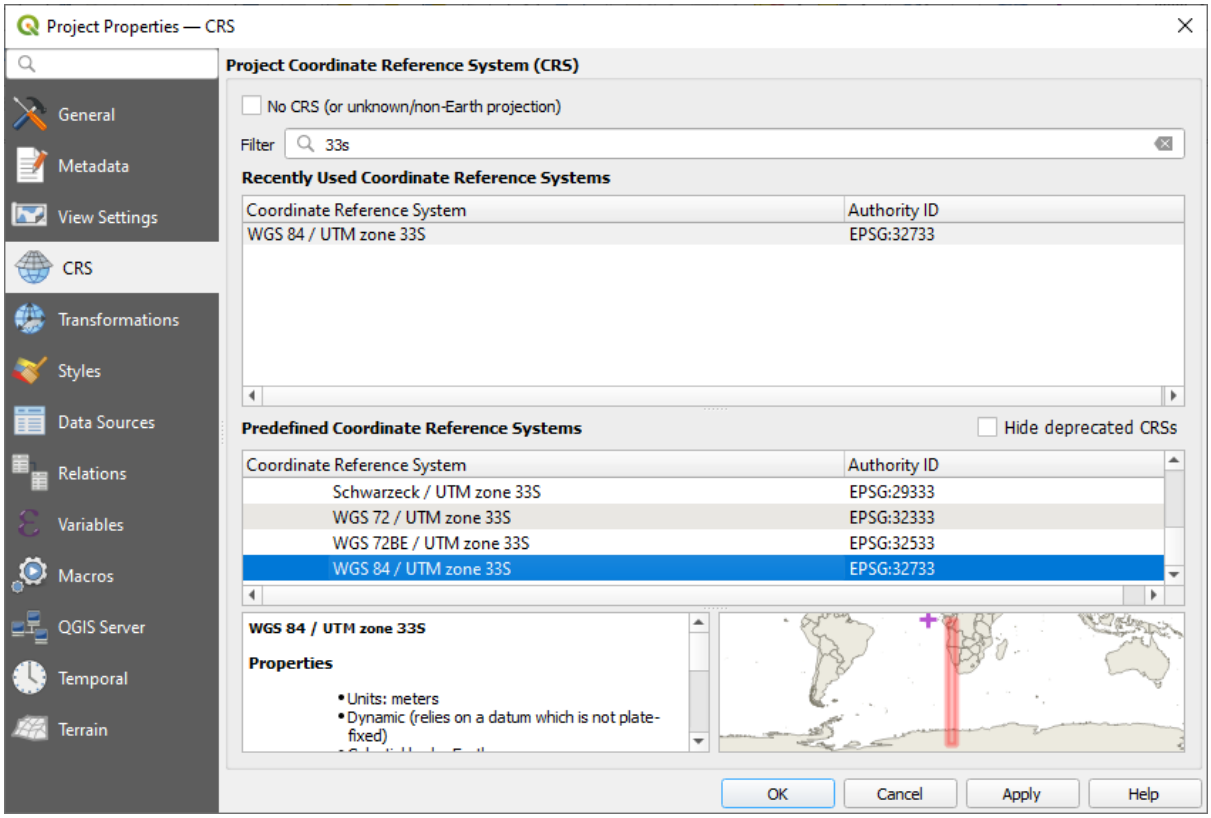


Figura8.1: Setting up the CRS

3. Save the project file by clicking on the Save Project toolbar button, or use the *Project* ► *Save As…* menu item.

   Save it in a new directory called `Rasterprac`, that you should create somewhere on your computer. You will save whatever layers you create in this directory as well. Save the project as `your_name_fynbos.qgs`.

8.4.4 Loading Data into the Map

In order to process the data, you will need to load the necessary layers (street names, zones, rainfall, DEM, districts) into the map canvas.

For vectors...

1. Click on the Open Data Source Manager button in the *Data Source Manager Toolbar*, and enable the *Vector* tab in the dialog that appears, or use the *Layer* ► *Add Layer* ► *Add Vector Layer…* menu item
2. Ensure that *File* is selected
3. Click on the … button to browse for vector dataset(s)
4. In the dialog that appears, open the `exercise_data/more_analysis/Streets` directory
5. Select the file `Street_Names_UTM33S.shp`
6. Click *Open*.

   The dialog closes and shows the original dialog, with the file path specified in the text field next to *Vector dataset(s)*. This allows you to ensure that the correct file is selected. It is also possible to enter the file path in this field manually, should you wish to do so.

7. Click *Add*. The vector layer will be loaded into your map. Its color is automatically assigned. You will change it later.
8. Rename the layer to `Streets`
    1. Right-click on it in the *Layers* panel (by default, the pane along the left-hand side of the screen)
    2. Click *Rename* in the dialog that appears and rename it, pressing the `Enter` key when done
9. Repeat the vector adding process, but this time select the `Generalised_Zoning_Dissolve_UTM33S.shp` file in the `Zoning` directory.
10. Rename it to `Zoning`.
11. Load also the vector layer `admin_boundaries/Western_Cape_UTM33S.shp` into your map.
12. Rename it to `Districts`.

For rasters...

1. Click on the Open Data Source Manager button and enable the *Raster* tab in the dialog that appears, or use the *Layer* ► *Add Layer* ► *Add Raster Layer…* menu item
2. Ensure that *File* is selected
3. Navigate to the appropriate file, select it, and click *Open*
4. Do this for each of the following two raster files, `DEM/SRTM.tif` and `rainfall/reprojected/rainfall.tif`
5. Rename the SRTM raster to `DEM` and the rainfall raster to `Rainfall` (with an initial capital)

8.4.5 Changing the layer order

Click and drag layers up and down in the *Layers* panel to change the order they appear in on the map so that you can see as many of the layers as possible.

Now that all the data is loaded and properly visible, the analysis can begin. It is best if the clipping operation is done first. This is so that no processing power is wasted on computing values in areas that are not going to be used anyway.

8.4.6 Find the Correct Districts

Due to the aforementioned area of investigation, we need to limit our districts to the following ones:

- `Bellville`
- `Cape`
- `Goodwood`
- `Kuils River`
- `Mitchells Plain`
- `Simon Town`
- `Wynberg`

1. Right-click on the `Districts` layer in the *Layers* panel.
2. In the menu that appears, select the *Filter…* menu item. The *Query Builder* dialog appears.
3. You will now build a query to select only the candidate districts:

1. In the *Fields* list, double-click on the `NAME_2` field to make it appear in the *SQL where clause* text field below
2. Click the *IN* button to append it to the SQL query
3. Open the brackets
4. Click the *All* button below the (currently empty) *Values* list.

   After a short delay, this will populate the *Values* list with the values of the selected field (`NAME_2`).
5. Double-click the value `Bellville` in the *Values* list to append it to the SQL query.
6. Add a comma and double-click to add `Cape` district
7. Repeat the previous step for the remaining districts
8. Close the brackets

Figura8.2: Query builder

The final query should be (the order of the districts in the brackets does not matter):

```
"NAME_2" in ('Bellville', 'Cape', 'Goodwood', 'Kuils River',
             'Mitchells Plain', 'Simon Town', 'Wynberg')
```

Nota: You can also use the `OR` operator; the query would look like this:

```
"NAME_2" = 'Bellville' OR "NAME_2" = 'Cape' OR
"NAME_2" = 'Goodwood' OR "NAME_2" = 'Kuils River' OR
"NAME_2" = 'Mitchells Plain' OR "NAME_2" = 'Simon Town' OR
"NAME_2" = 'Wynberg'
```

1. Click *OK* twice.

   The districts shown in your map are now limited to those in the list above.

8.4.7 Clip the Rasters

Now that you have an area of interest, you can clip the rasters to this area.

1. Open the clipping dialog by selecting the menu item *Raster* ► *Extraction* ► *Clip Raster by Mask Layer…*
2. In the *Input layer* dropdown list, select the `DEM` layer
3. In the *Mask layer* dropdown list, select the `Districts` layer
4. Scroll down and specify an output location in the *Clipped (mask)* text field by clicking the … button and choosing *Save to File…*
   1. Navigate to the `Rasterprac` directory
   2. Enter a file name - `DEM_clipped.tif`
   3. Save
5. Make sure that *Open output file after running algorithm* is checked
6. Click *Run*

   After the clipping operation has completed, leave the *Clip Raster by Mask Layer* dialog open, to be able to reuse the clipping area
7. Select the `Rainfall` raster layer in the *Input layer* dropdown list and save your output as `Rainfall_clipped.tif`

8. Do not change any other options. Leave everything the same and click *Run*.
9. After the second clipping operation has completed, you may close the *Clip Raster by Mask Layer* dialog
10. Save the map

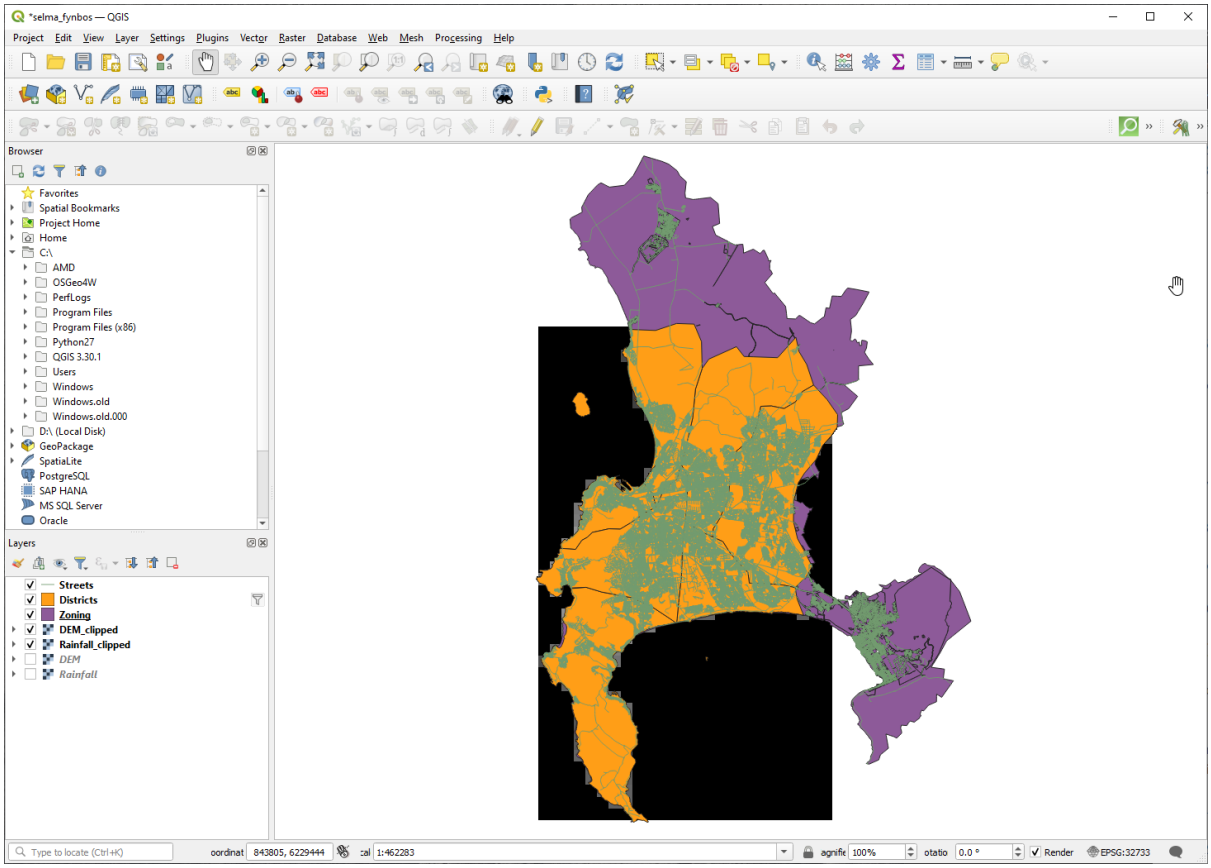


Figura8.3: Map view with filtered vector, clipped raster and reordered layers

Align the rasters

For our analysis we need the rasters to have the same CRS and they have to be aligned.

First we change the resolution of our rainfall data to 30 meters (pixel size):

1. In the *Layers* panel, ensure that `Rainfall_clipped` is the active layer (i.e., it is highlighted by having been clicked on)
2. Click on the *Raster* ► *Projections* ► *Warp (Reproject)…* menu item to open the *Warp (Reproject)* dialog
3. Under *Resampling method to use*, select *Bilinear (2x2 kernel)* from the drop down menu
4. Set *Output file resolution in target georeferenced units* to `30`
5. Scroll down to *Reprojected* and save the output in your `rainfall/reprojected` directory as `Rainfall30.tif`.
6. Make sure that *Open output file after running algorithm* is checked

Figura8.4: Warp (Reproject) Rainfall_clipped

Then we align the DEM:

1. In the *Layers* panel, ensure that `DEM_clipped` is the active layer (i.e., it is highlighted by having been clicked on)
2. Click on the *Raster* ► *Projections* ► *Warp (Reproject)…* menu item to open the *Warp (Reproject)* dialog
3. Under *Target CRS*, select *Project CRS: EPSG:32733 - WGS 84 / UTM zone 33S* from the drop down menu
4. Under *Resampling method to use*, select *Bilinear (2x2 kernel)* from the drop down menu
5. Set *Output file resolution in target georeferenced units* to `30`
6. Scroll down to *Georeferenced extents of output file to be created*. Use the button to the right of the text box to select *Calculate from Layer* ► *Rainfall30*.
7. Scroll down to *Reprojected* and save the output in your `DEM/reprojected` directory as `DEM30.tif`.
8. Make sure that *Open output file after running algorithm* is checked

In order to properly see what's going on, the symbology for the layers needs to be changed.

8.4.8 Changing the symbology of vector layers

1. In the *Layers* panel, right-click on the *Streets* layer
2. Select *Properties* from the menu that appears
3. Switch to the *Symbology* tab in the dialog that appears
4. Click on the *Line* entry in the top widget
5. Select a symbol in the list below or set a new one (color, transparency, …)
6. Click *OK* to close the *Layer Properties* dialog. This will change the rendering of the *Streets* layer.
7. Follow a similar process for the *Zoning* layer and choose an appropriate color for it

8.4.9 Changing the symbology of raster layers

Raster layer symbology is somewhat different.

1. Open the *Properties* dialog for the *Rainfall30* raster layer
2. Switch to the *Symbology* tab. You'll notice that this dialog is very different from the version used for vector layers.
3. Expand *Min/Max Value Settings*
4. Ensure that the button *Mean +/- standard deviation* is selected
5. Make sure that the value in the associated box is `2.00`
6. For *Contrast enhancement*, make sure it says *Stretch to MinMax*
7. For *Color gradient*, change it to *White to Black*
8. Click *OK*

Figura8.5: Raster symbology

The `Rainfall30` raster, if visible, should change colors, allowing you to see different brightness values for each pixel

9. Repeat this process for the `DEM30` layer, but set the standard deviations used for stretching to `4.00`

8.4.10 Clean up the map

1. Remove the original `Rainfall` and `DEM` layers, as well as `Rainfall_clipped` and `DEM_clipped` from the *Layers* panel:
   - Right-click on these layers and select *Remove*.

   Nota: This will not remove the data from your storage device, it will merely take it out of your map.

2. Save the map
3. You can now hide the vector layers by unchecking the box next to them in the *Layers* panel. This will make the map render faster and will save you some time.

8.4.11 Create the hillshade

In order to create the hillshade, you will need to use an algorithm that was written for this purpose.

1. In the *Layers* panel, ensure that `DEM30` is the active layer (i.e., it is highlighted by having been clicked on)
2. Click on the *Raster* ► *Analysis* ► *Hillshade...* menu item to open the *Hillshade* dialog
3. Scroll down to *Hillshade* and save the output in your `Rasterprac` directory as `hillshade.tif`
4. Make sure that *Open output file after running algorithm* is checked
5. Click *Run*
6. Wait for it to finish processing.

Figura8.6: Raster analysis Hillshade

The new `hillshade` layer has appeared in the *Layers* panel.

1. Right-click on the `hillshade` layer in the *Layers* panel and bring up the *Properties* dialog
2. Click on the *Transparency* tab and set the *Global Opacity* slider to `20%`
3. Click *OK*
4. Note the effect when the transparent hillshade is superimposed over the clipped DEM. You may have to change the order of your layers, or click off the `Rainfall30` layer in order to see the effect.

8.4.12 Slope

1. Click on the *Raster* ► *Analysis* ► *Slope...* menu item to open the *Slope* algorithm dialog
2. Select `DEM30` as *Input layer*
3. Check *Slope expressed as percent instead of degrees*. Slope can be expressed in different units (percent or degrees). Our criteria suggest that the plant of interest grows on slopes with a gradient between 15% and 60%. So we need to make sure our slope data is expressed as a percent.
4. Specify an appropriate file name and location for your output.
5. Make sure that *Open output file after running algorithm* is checked

6. Click *Run*

Figura8.7: Raster analysis Slope

The slope image has been calculated and added to the map. As usual, it is rendered in grayscale. Change the symbology to a more colorful one:

1. Open the layer *Properties* dialog (as usual, via the right-click menu of the layer)
2. Click on the *Symbology* tab
3. Where it says *Singleband gray* (in the *Render type* dropdown menu), change it to *Singleband pseudocolor*
4. Choose *Mean +/- standard deviation x* for *Min / Max Value Settings* with a value of `2.0`
5. Select a suitable *Color ramp*
6. Click *Run*

8.4.13 Try Yourself: Aspect

Use the same approach as for calculating the slope, choosing *Aspect…* in the *Raster* ► *Analysis* menu.

Remember to save the project periodically.

8.4.14 Reclassifying rasters

1. Choose *Raster* ► *Raster calculator…*
2. Specify your `Rasterprac` directory as the location for the *Output layer* (click on the … button), and save it as `slope15_60.tif`
3. Ensure that the *Open output file after running algorithm* box is selected.

   In the *Raster bands* list on the left, you will see all the raster layers in your *Layers* panel. If your Slope layer is called *slope*, it will be listed as `slope@1`. Indicating band 1 of the slope raster.
4. The slope needs to be between `15` and `60` degrees.

   Using the list items and buttons in the interface, build the following expression:

```
(slope@1 > 15) AND (slope@1 < 60)
```

5. Set the *Output layer* field to an appropriate location and file name.
6. Click *Run*.

Figura8.8: Raster calculator Slope

Now find the correct aspect (east-facing: between `45` and `135` degrees) using the same approach.

1. Build the following expression:

```
(aspect@1 > 45) AND (aspect@1 < 135)
```

You will know it worked when all of the east-facing slopes are white in the resulting raster (it's almost as if they are being lit by the morning sunlight).

Find the correct rainfall (greater than `1000` mm) the same way. Use the following expression:

```
Rainfall30@1 > 1000
```

Now that you have all three criteria each in separate rasters, you need to combine them to see which areas satisfy all the criteria. To do so, the rasters will be multiplied with each other. When this happens, all overlapping pixels with a value of `1` will retain the value of `1` (i.e. the location meets the criteria), but if a pixel in any of the three rasters has the value of `0` (i.e. the location does not meet the criteria), then it will be `0` in the result. In this way, the result will contain only the overlapping areas that meet all of the appropriate criteria.

8.4.15 Combining rasters

1. Open the *Raster Calculator* (*Raster* ► *Raster Calculator…*)
2. Build the following expression (with the appropriate names for your layers):

   ```
   [aspect45_135] * [slope15_60] * [rainfall_1000]
   ```

3. Set the output location to the `Rasterprac` directory
4. Name the output raster `aspect_slope_rainfall.tif`
5. Ensure that *Open output file after running algorithm* is checked
6. Click *Run*

The new raster now properly displays the areas where all three criteria are satisfied.

Save the project.

Figura8.9: Map view where all three criteria are satisfied

The next criterion that needs to be satisfied is that the area must be `250` m away from urban areas. We will satisfy this requirement by ensuring that the areas we compute are inside rural areas, and are `250` m or more from the edge of the area. Hence, we need to find all rural areas first.

8.4.16 Finding rural areas

1. Hide all layers in the *Layers* panel
2. Unhide the `Zoning` vector layer
3. Right-click on it and bring up the *Attribute Table* dialog. Note the many different ways that the land is zoned here. We want to isolate the rural areas. Close the Attribute table.
4. Right-click on the `Zoning` layer and select *Filter…* to bring up the *Query Builder* dialog
5. Build the following query:

   ```
   "Gen_Zoning" = 'Rural'
   ```

   See the earlier instructions if you get stuck.
6. Click *OK* to close the *Query Builder* dialog. The query should return one feature.

Query builder Zoning

You should see the rural polygons from the `Zoning` layer. You will need to save these.

1. In the right-click menu for `Zoning`, select *Export* ► *Save Features As…*.
2. Save your layer under the `Rasterprac` directory
3. Name the output file `rural.shp`
4. Click *OK*
5. Save the project

Now you need to exclude the areas that are within `250m` from the edge of the rural areas. Do this by creating a negative buffer, as explained below.

8.4.17 Creating a negative buffer

1. Click the menu item *Vector* ► *Geoprocessing Tools* ► *Buffer…*
2. In the dialog that appears, select the `rural` layer as your input vector layer (*Selected features only* should not be checked)
3. Set *Distance* to `-250`. The negative value means that the buffer will be an internal buffer. Make sure that the units are meters in the dropdown menu.
4. Check *Dissolve result*
5. In *Buffered*, place the output file in the `Rasterprac` directory, and name it `rural_buffer.shp`
6. Click *Save*
7. Click *Run* and wait for the processing to complete
8. Close the *Buffer* dialog.

   Make sure that your buffer worked correctly by noting how the `rural_buffer` layer is different from the `rural` layer. You may need to change the drawing order in order to observe the difference.
9. Remove the `rural` layer
10. Save the project

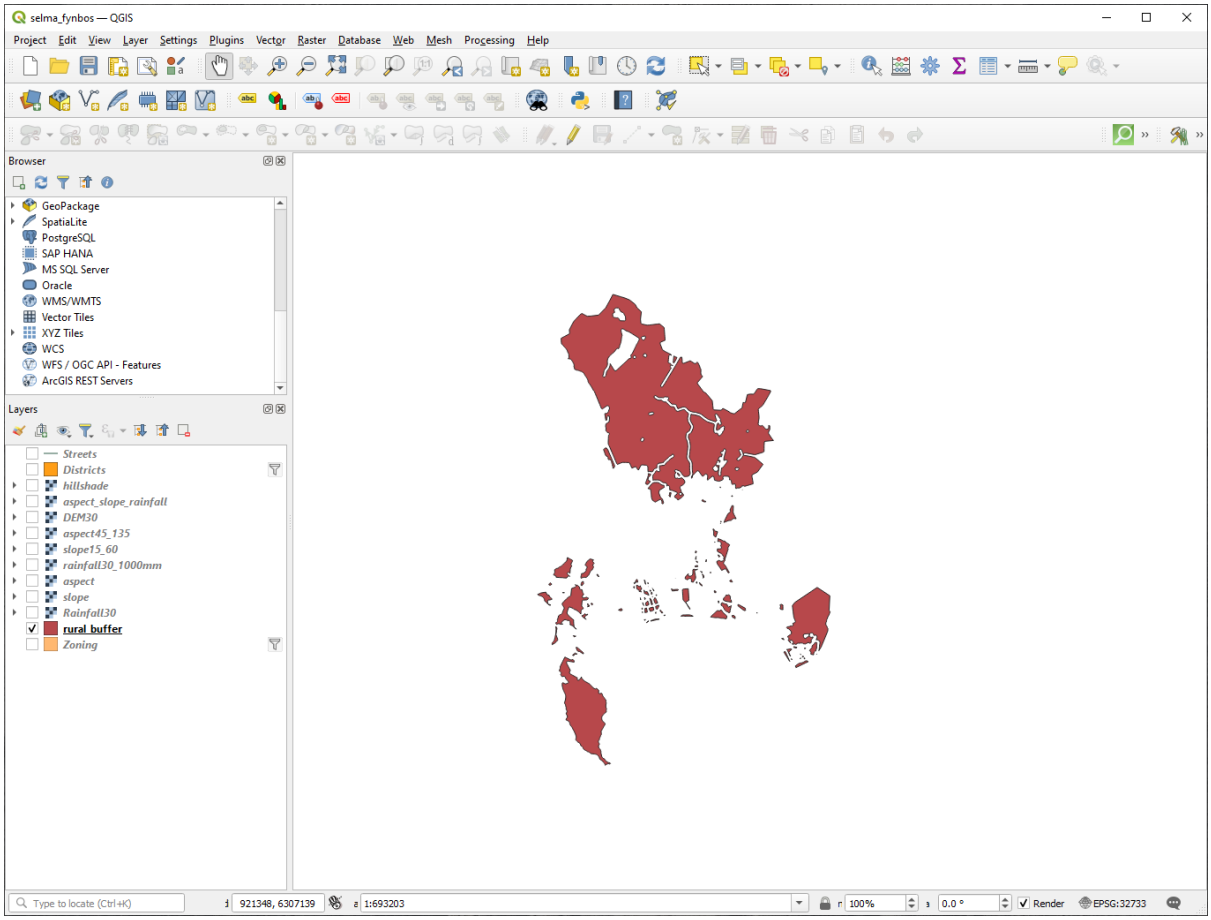


Figura8.10: Map view with rural buffer

Now you need to combine your `rural_buffer` vector layer with the `aspect_slope_rainfall` raster. To combine them, we will need to change the data format of one of the layers. In this case, you will vectorize the raster, since vector layers are more convenient when we want to calculate areas.

8.4.18 Vectorizing the raster

1. Click on the menu item *Raster* ► *Conversion* ► *Polygonize (Raster to Vector)…*
2. Select the `aspect_slope_rainfall` raster as *Input layer*
3. Set *Name of the field to create* to `suitable` (the default field name is `DN` - Digital number data)
4. Save the output. Under *Vectorized*, select *Save file as*. Set the location to `Rasterprac` and name the file `aspect_slope_rainfall_all.shp`.
5. Ensure that *Open output file after running algorithm* is checked
6. Click *Run*
7. Close the dialog when processing is complete

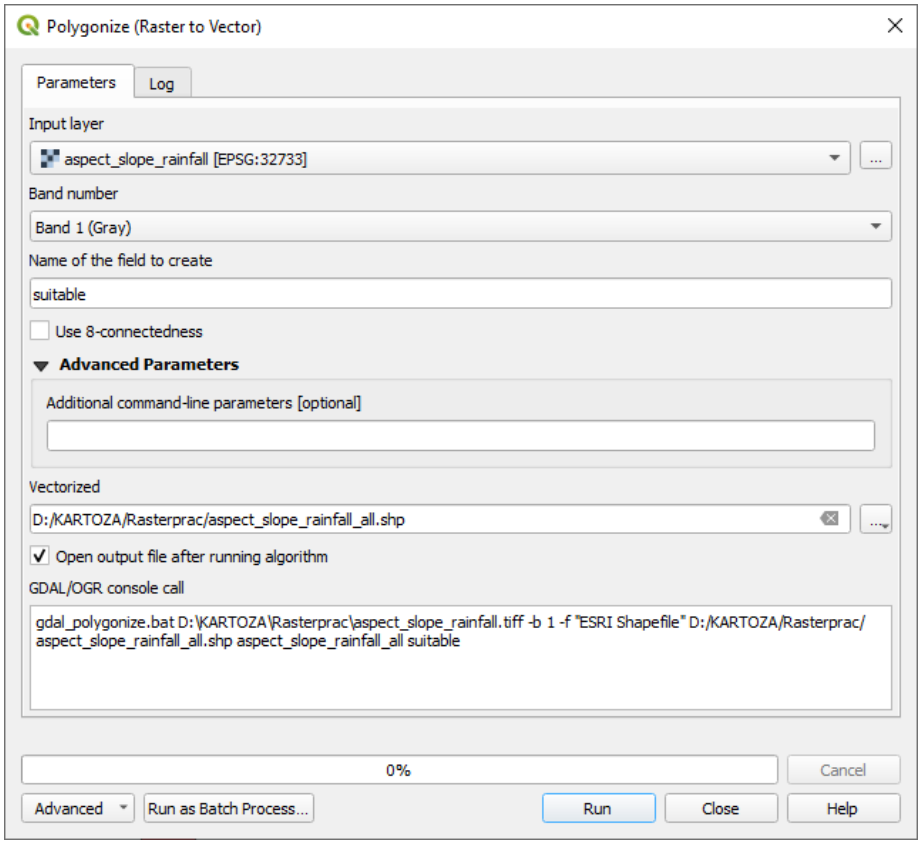


Figura8.11: Raster to Vector

All areas of the raster have been vectorized, so you need to select only the areas that have a value of `1` in the `suitable` field. (Digital Number.

1. Open the *Query Builder* dialog (right-click - *Filter…*) for the new vector layer
2. Build this query:

```
"suitable" = 1
```

3. Click *OK*
4. After you are sure the query is complete (and only the areas that meet all three criteria, i.e. with a value of `1` are visible), create a new vector file from the results, using the *Export* ► *Save Features As…* in the layer's right-click menu
5. Save the file in the `Rasterprac` directory
6. Name the file `aspect_slope_rainfall_1.shp`
7. Remove the `aspect_slope_rainfall_all` layer from your map
8. Save your project

When we use an algorithm to vectorize a raster, sometimes the algorithm yields what is called «Invalid geometries», i.e. there are empty polygons, or polygons with mistakes in them, that will be difficult to analyze in the future. So, we need to use the «Fix Geometry» tool.

8.4.19 Fixing geometry

1. In the *Processing Toolbox*, search for «Fix geometries», and *Execute…* it
2. For the *Input layer*, select `aspect_slope_rainfall_1`
3. Under *Fixed geometries*, select *Save file as*, and save the output to `Rasterprac` and name the file `fixed_aspect_slope_rainfall.shp`.
4. Ensure that *Open output file after running algorithm* is checked
5. Click *Run*
6. Close the dialog when processing is complete

Now that you have vectorized the raster, and fixed the resulting geometry, you can combine the aspect, slope, and rainfall criteria with the distance from human settlement criteria by finding the intersection of the `fixed_aspect_slope_rainfall` layer and the `rural_buffer` layer.

8.4.20 Determining the Intersection of vectors

1. Click the menu item *Vector* ► *Geoprocessing Tools* ► *Intersection…*
2. In the dialog that appears, select the `rural_buffer` layer as *Input layer*
3. For the *Overlay layer*, select the `fixed_aspect_slope_rainfall` layer
4. In *Intersection*, place the output file in the `Rasterprac` directory
5. Name the output file `rural_aspect_slope_rainfall.shp`
6. Click *Save*
7. Click *Run* and wait for the processing to complete
8. Close the *Intersection* dialog.

   Make sure that your intersection worked correctly by noting that only the overlapping areas remain.
9. Save the project

The next criteria on the list is that the area must be greater than `6000` m^2. You will now calculate the polygon areas in order to identify the areas that are the appropriate size for this project.

8.4.21 Calculating the area for each polygon

1. Open the new vector layer's right-click menu
2. Select *Open attribute table*
3. Click the Toggle editing button in the top left corner of the table, or press `Ctrl+e`
4. Click the Open field calculator button in the toolbar along the top of the table, or press `Ctrl+i`
5. In the dialog that appears, make sure that *Create new field* is checked, and set the *Output field name* to `area` The output field type should be a decimal number (real). Set *Precision* to `1` (one decimal).
6. In the *Expression* area, type:

   ```
   $area
   ```

   This means that the field calculator will calculate the area of each polygon in the vector layer and will then populate a new integer column (called `area`) with the computed value.

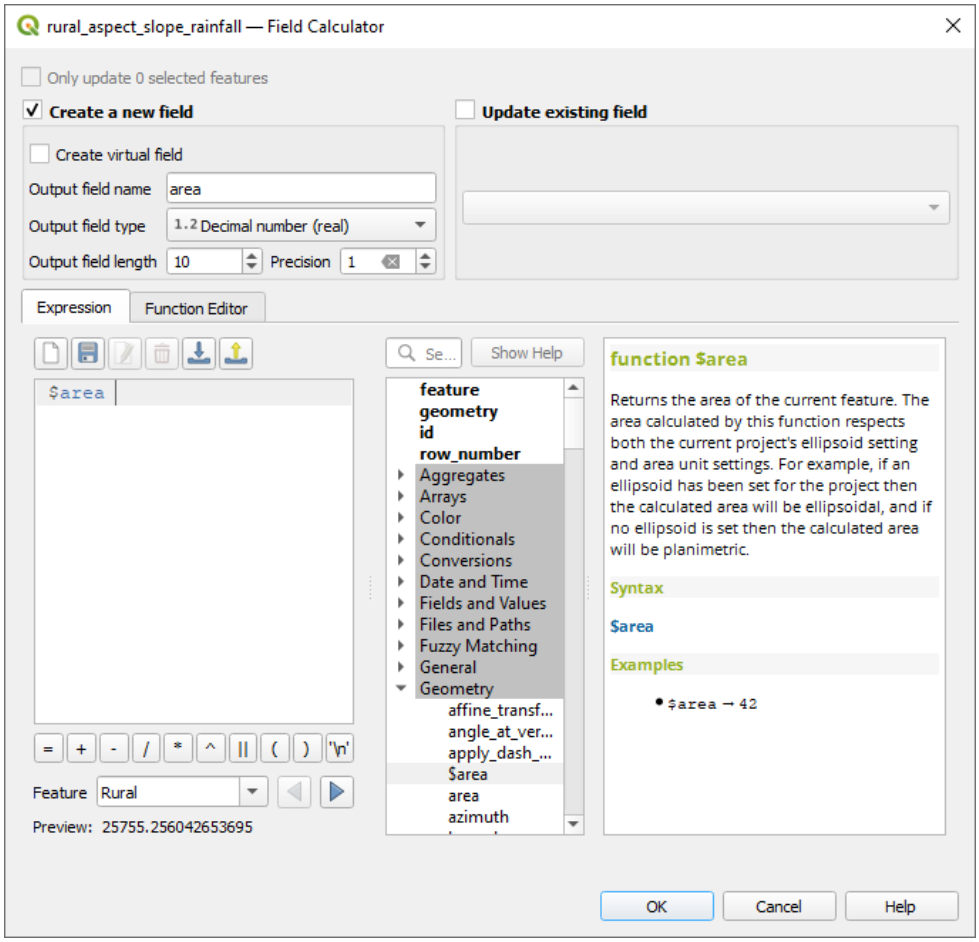


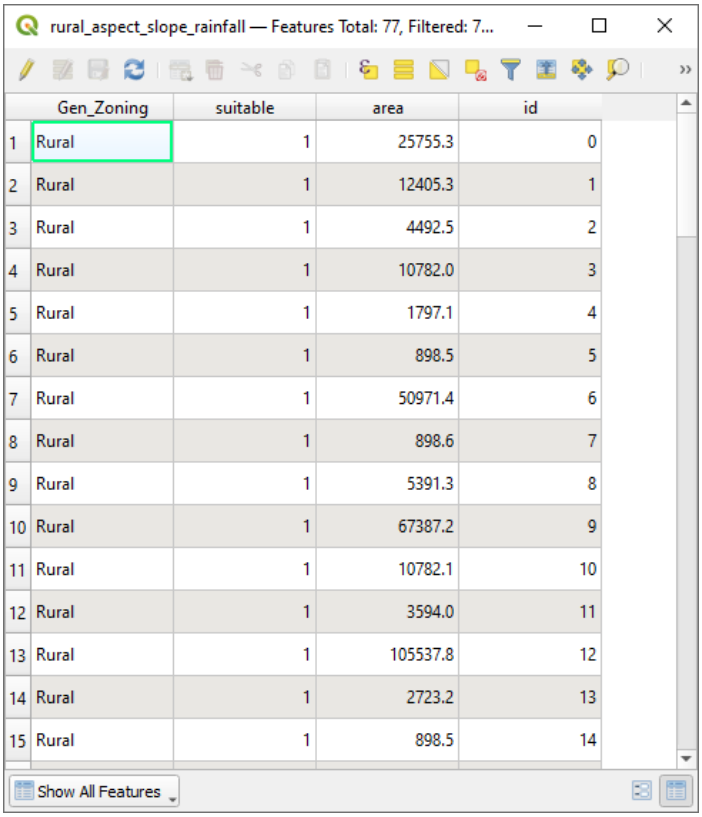
Figura8.12: Field Calculator

7. Click *OK*
8. Do the same thing for another new field called `id`. In *Field calculator expression*, type:

```
$id
```

   This ensures that each polygon has a unique ID for identification purposes.

9. Click Toggle editing again, and save your edits if prompted to do so



	Gen_Zoning	suitable	area	id
1	Rural	1	25755.3	0
2	Rural	1	12405.3	1
3	Rural	1	4492.5	2
4	Rural	1	10782.0	3
5	Rural	1	1797.1	4
6	Rural	1	898.5	5
7	Rural	1	50971.4	6
8	Rural	1	898.6	7
9	Rural	1	5391.3	8
10	Rural	1	67387.2	9
11	Rural	1	10782.1	10
12	Rural	1	3594.0	11
13	Rural	1	105537.8	12
14	Rural	1	2723.2	13
15	Rural	1	898.5	14

Figura8.13: Attribute table with area and id columns

8.4.22 Selecting areas of a given size

Now that the areas are known:

1. Build a query (as usual) to select only the polygons that are larger than `6000` m^2. The query is:

   ```
   "area" > 6000
   ```

2. Save the selection in the `Rasterprac` directory as a new vector layer called `suitable_areas.shp`.

You now have the suitable areas that meet all of the habitat criteria for the rare fynbos plant, from which you will pick the four areas that are nearest to the University of Cape Town.

8.4.23 Digitize the University of Cape Town

1. Create a new vector layer in the `Rasterprac` directory as before, but this time, use *Point* as *Geometry type* and name it `university.shp`
2. Ensure that it is in the correct CRS (`Project CRS:EPSG:32733 - WGS 84 / UTM zone 33S`)
3. Finish creating the new layer (click *OK*)
4. Hide all layers except the new `university` layer and the `Streets` layer.
5. Add a background map (OSM):
   1. Go to the *Browser* panel and navigate to *XYZ Tiles* ► *OpenStreetMap*
   2. Drag and drop the `OpenStreetMap` entry to the bottom of the *Layers* panel

   Using your internet browser, look up the location of the University of Cape Town. Given Cape Town's unique topography, the university is in a very recognizable location. Before you return to QGIS, take note of where the university is located, and what is nearby.

6. Ensure that the `Streets` layer clicked on, and that the `university` layer is highlighted in the *Layers* panel
7. Navigate to the *View* ► *Toolbars* menu item and ensure that *Digitizing* is selected. You should then see a toolbar icon with a pencil on it (Toggle editing). This is the *Toggle Editing* button.
8. Click the *Toggle editing* button to enter *edit mode*. This allows you to edit a vector layer
9. Click the Add Point Feature button, which should be nearby the Toggle editing button
10. With the *Add feature* tool activated, left-click on your best estimate of the location of the University of Cape Town
11. Supply an arbitrary integer when asked for the `id`
12. Click *OK*
13. Click the Save Layer Edits button
14. Click the *Toggle editing* button to stop your editing session
15. Save the project

8.4.24 Find the locations that are closest to the University of Cape Town

1. Go to the *Processing Toolbox*, locate the *Join Attributes by Nearest* algorithm (*Vector general* ► *Join Attributes by Nearest*) and execute it
2. *Input layer* should be `university`, and *Input layer 2* `suitable_areas`
3. Set an appropriate output location and name (*Joined layer*)
4. Set the *Maximum nearest neighbors* to `4`
5. Ensure that *Open output file after running algorithm* is checked
6. Leave the rest of the parameters with their default values
7. Click *Run*

The resulting point layer will contain four features - they will all have the location of the university and its attributes, and in addition, the attributes of the nearby suitable areas (including the `id`), and the distance to that location.

1. Open the attribute table of the result of the join
2. Note the `id` of the four nearest suitable areas, and then close the attribute table
3. Open the attribute table of the `suitable_areas` layer
4. Build a query to select the four suitable areas closest to the university (selecting them using the `id` field)

This is the final answer to the research question.

For your submission, create a fully labeled layout that includes the semi-transparent hillshade layer over an appealing raster of your choice (such as the DEM or the slope raster, for example). Also include the university and the `suitable_areas` layer, with the four suitable areas that are closest to the university highlighted. Follow all the best practices for cartography in creating your output map.

CAPÍTULO 9

Module: Plugins

Os módulos permitem aumentar as funcionalidade do QGIS. Neste capítulo, iremos ver como activar e usar os módulos.

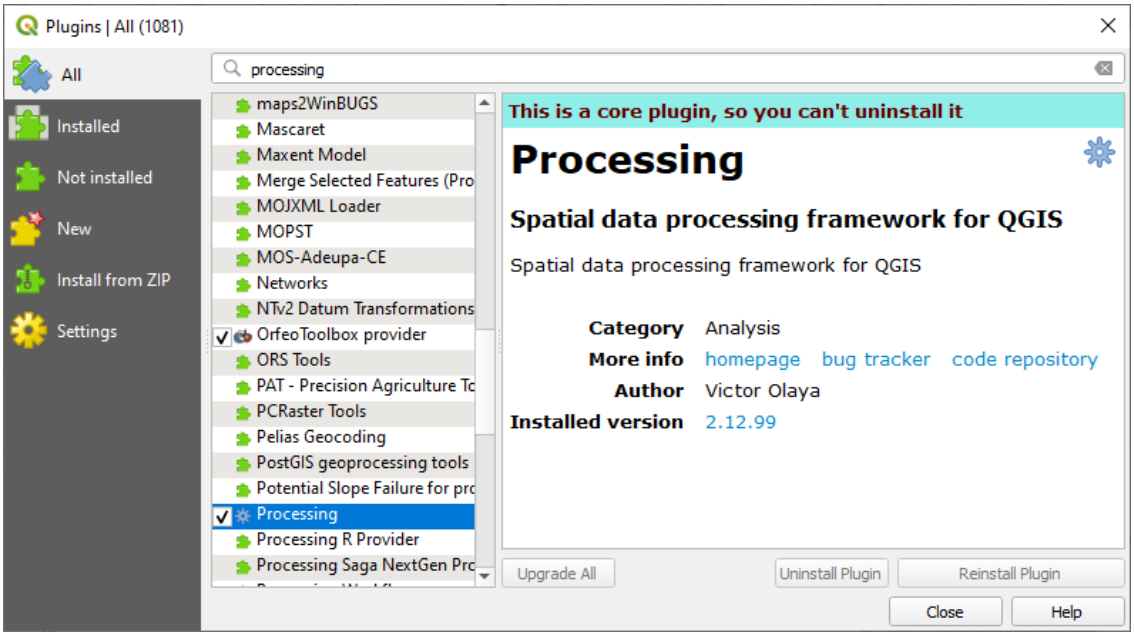
9.1 Lesson: Installing and Managing Plugins

To begin using plugins, you need to know how to download, install and activate them. To do this, you will learn how to use the *Plugin Installer* and *Plugin Manager*.

The goal for this lesson: To understand and use QGIS" plugin system.

9.1.1 ??? Follow Along: Managing Plugins

1. To open the *Plugin Manager*, click on the menu item *Plugins* ► *Manage and Install Plugins*.
2. In the dialog that opens, find the *Processing* plugin:

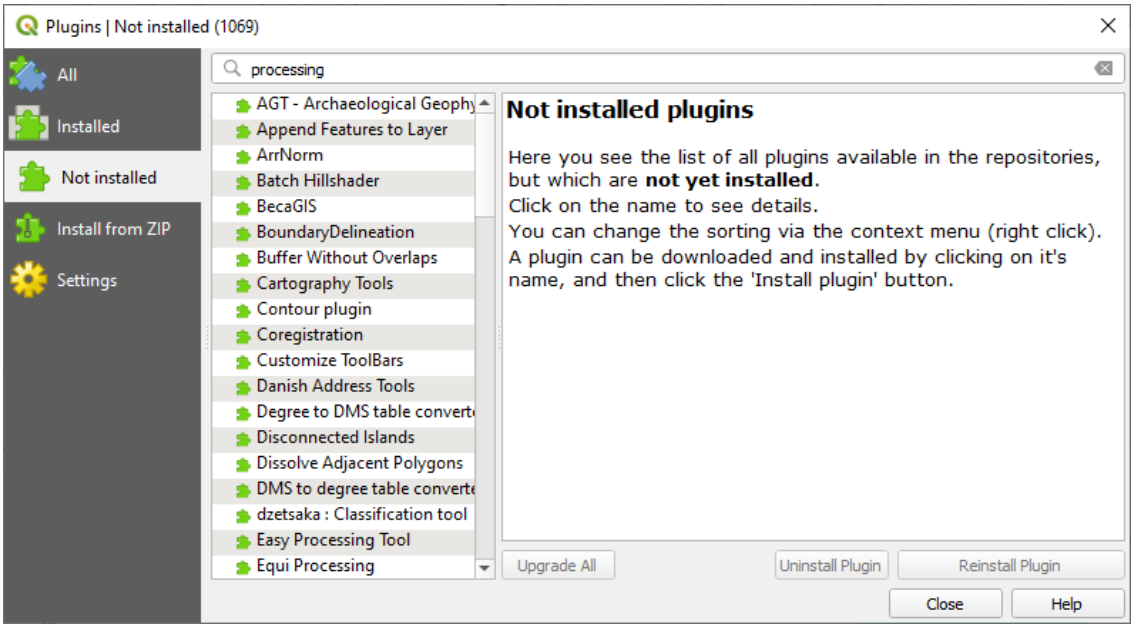


3. Click in the box next to this plugin and uncheck it to deactivate it.
4. Click *Close*.
5. Looking at the menu, you will notice that the *Processing* menu is is now gone. This means that many of the processing functions you have been using before have disappeared! For example look at the *Vector* ► and *Raster* ► menus. This is because they are part of the *Processing* plugin, which needs to be activated to use them.
6. Open the *Plugin Manager* again and reactivate the *Processing* plugin by clicking in the checkbox next to it.
7. *Close* the dialog. The *Processing* menu and functions should be available again.

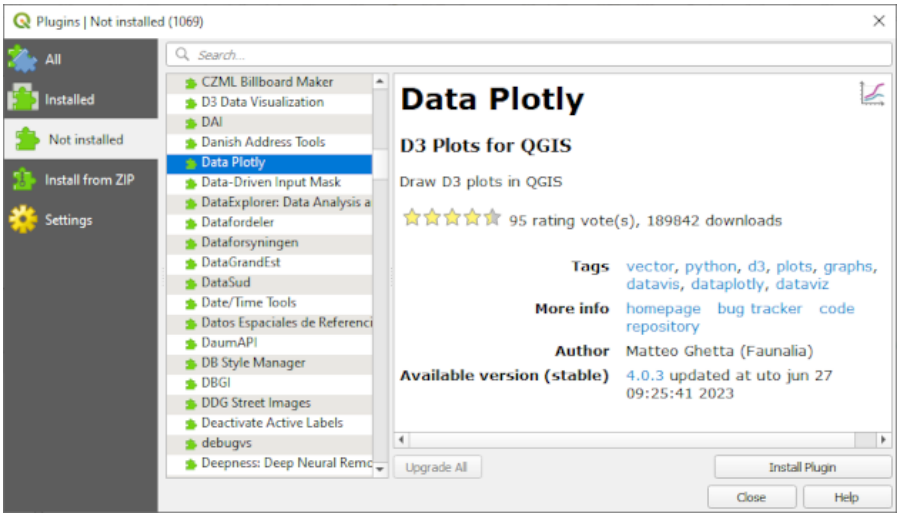
9.1.2 ??? Follow Along: Installing New Plugins

The list of plugins that you can activate and deactivate draws from the plugins that you currently have installed. To install new plugins:

1. Select the *Not Installed* option in the *Plugin Manager* dialog. The plugins available for you to install will be listed here. This list will vary depending on your existing system setup.



2. Find information about the plugin by selecting it in the list



3. Install the one(s) you are interested in by clicking the *Install Plugin* button below the plugin information panel.

Nota: if the plugin has some error it will be listed in the *Invalid* tab. You can then contact the plugin owner to fix the problem.

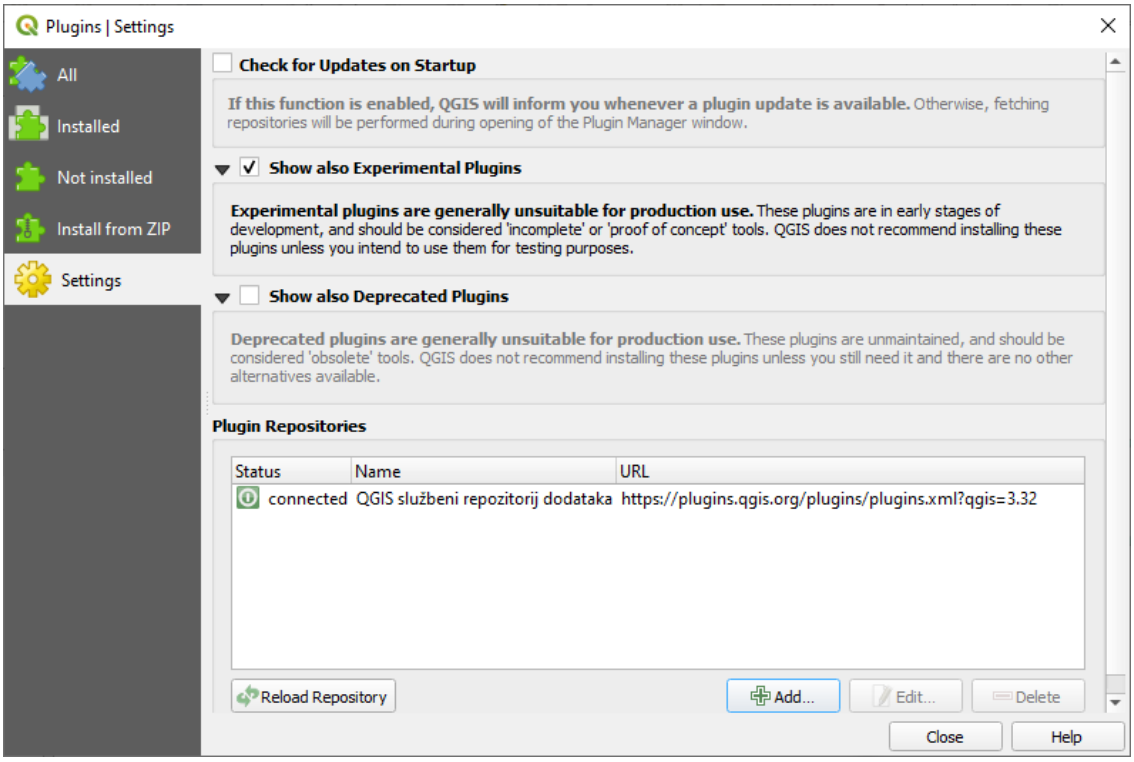
9.1.3 ??? Follow Along: Configuring Additional Plugin Repositories

The plugins that are available to you for installation depend on which plugin *repositories* you are configured to use.

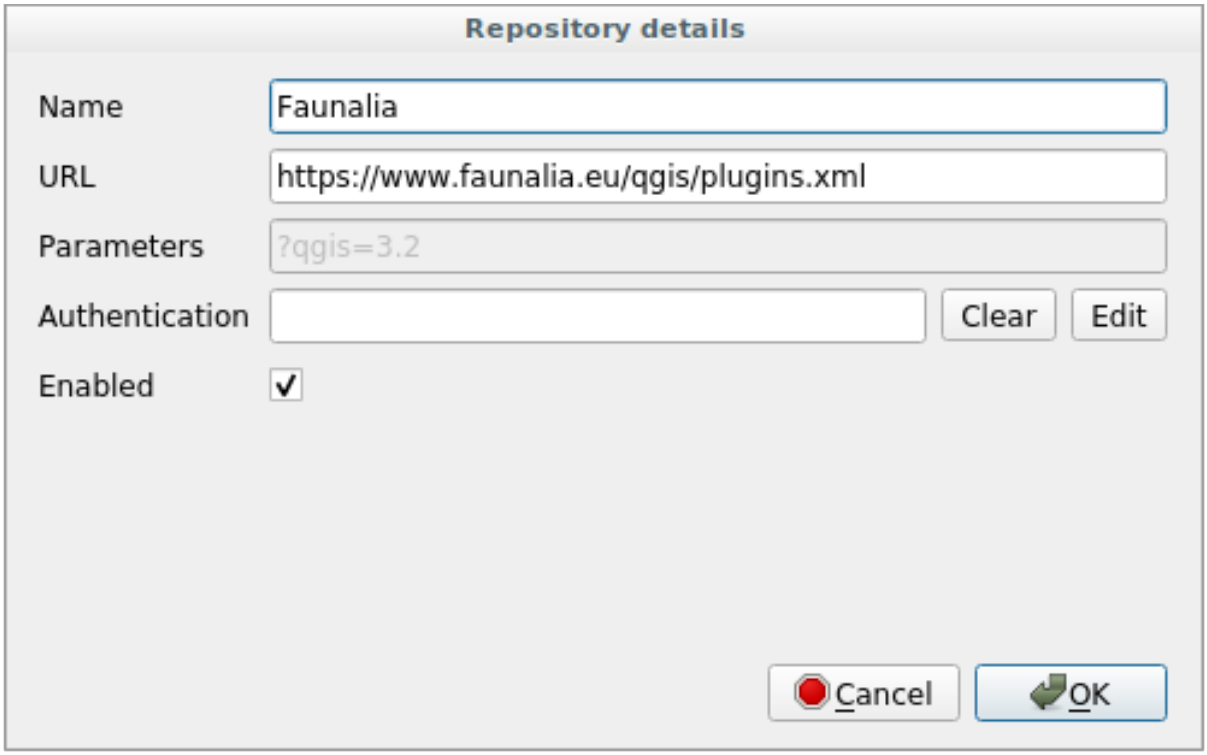
QGIS plugins are stored online in repositories. By default, only the official repository is active, meaning that you can only access plugins that are published there. Given the diversity of available tools, this repository should meet most of your needs.

It is possible, however, to try out more plugins than the default ones. First, you want to configure additional repositories. To do this:

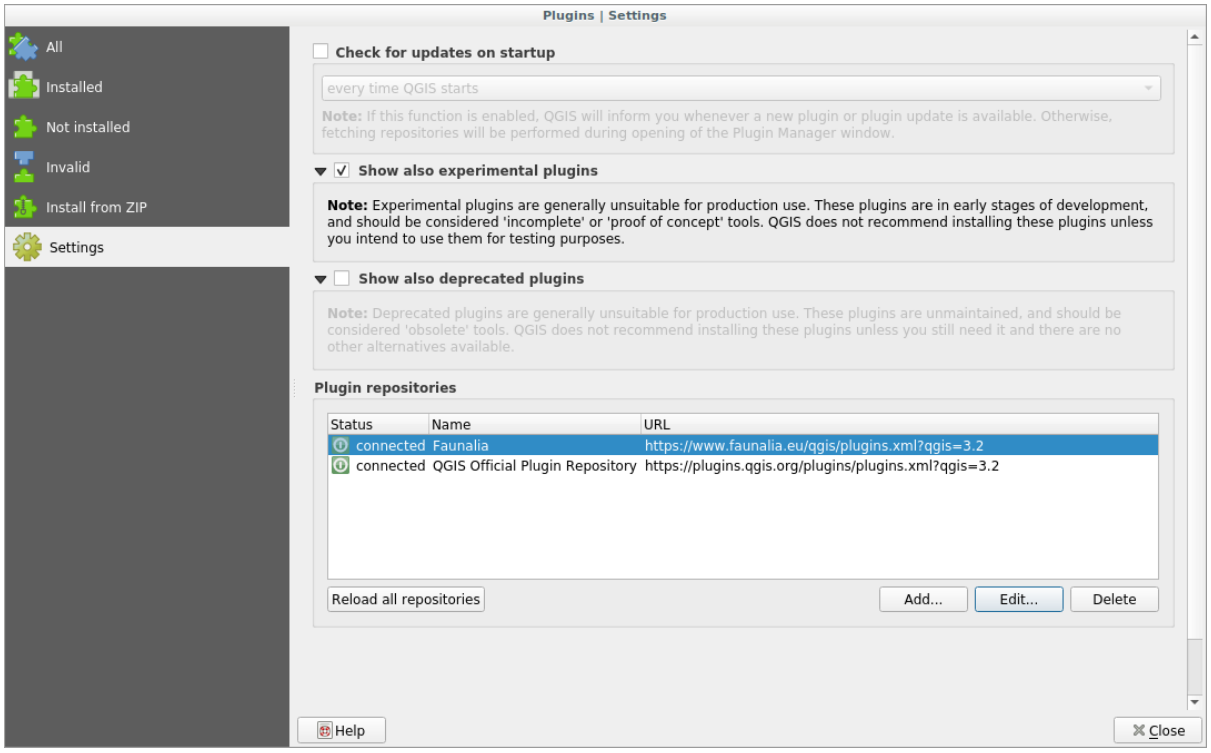
1. Open the *Settings* tab in the *Plugin Manager* dialog



2. Click *Add* to find and add a new repository.
3. Provide a Name and URL for the new repository you want to configure and make sure the *Enabled* checkbox is selected.



4. You will now see the new plugin repo listed in the list of configured Plugin Repositories



5. You can also select the option to display Experimental Plugins by selecting the *Show also experimental plugins* checkbox.
6. If you now switch back to the *Not Installed* tab, you will see that additional plugins are available for installation.
7. To install a plugin, click on it in the list and then on the *Install plugin* button.

9.1.4 In Conclusion

Installing plugins in QGIS should be straightforward and effective!

9.1.5 What's Next?

Next we'll introduce you to some useful plugins as examples.

9.2 Lesson: Useful QGIS Plugins

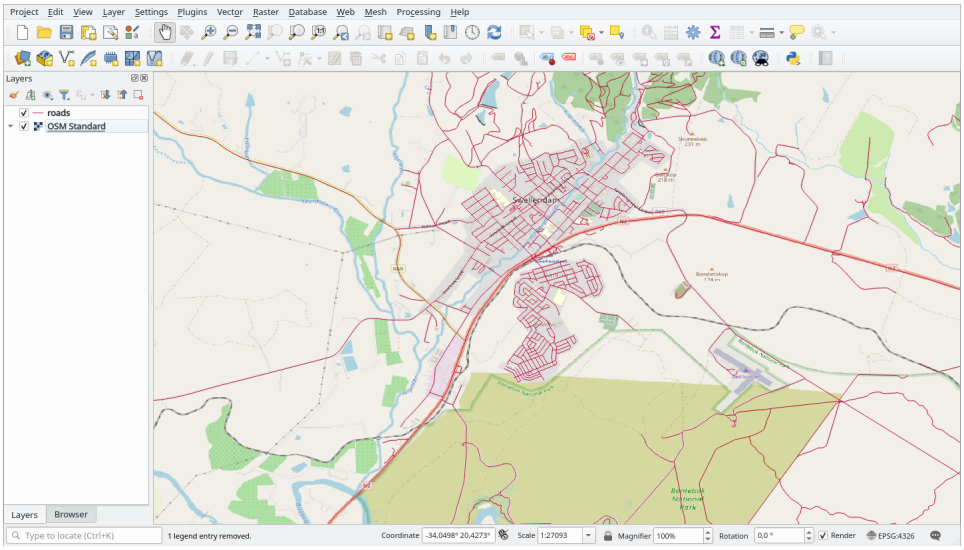
Now that you can install, enable and disable plugins, let's see how this can help you in practice by looking at some examples of useful plugins.

The goal for this lesson: To familiarize yourself with the plugin interface and get acquainted with some useful plugins.

9.2.1 ??? Follow Along: The QuickMapServices Plugin

The QuickMapServices plugin is a simple and easy to use plugin that adds base maps to your QGIS project. It has many different options and settings. Let's start to explore some of its features.

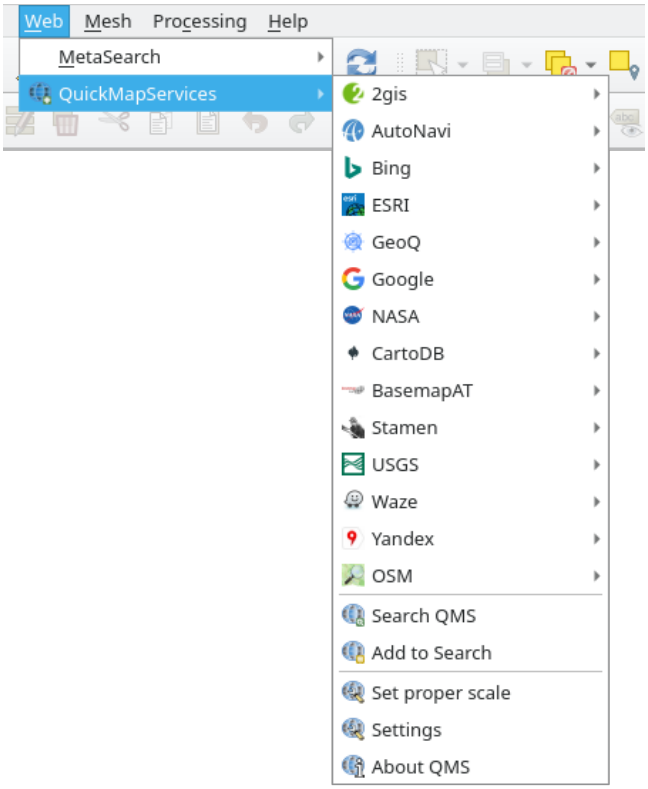
1. Start a new map and add the *roads* layer from the `training_data` Geopackage.
2. Install the **QuickMapServices** plugin.
3. Click on *Web* ► *QuickMapServices*. The first menu lists different map providers (`OSM`, `NASA`) with available maps.
4. Click on an entry and you would load the base map into your project.



Nice! But one of the main strengths of QMS is to provide access to many data providers. Let's add them.

1. Click on *Web* ► *QuickMapServices* ► *Settings*
2. Go to the *More services* tab.
3. Read carefully the message of this tab and if you agree click on the *Get Contributed pack* button.

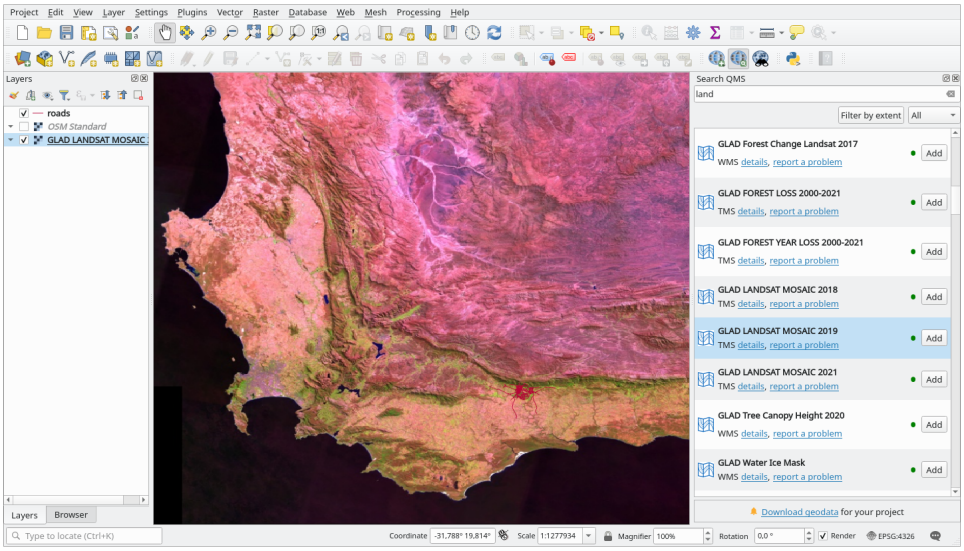
4. Click *Save*.
5. Reopen the *Web* ► *QuickMapServices* menu you will see that more providers are available.



6. Choose the one that best fits your needs, and load the data in the project!

It is also possible to search trough the now available data providers

1. Open the plugin's search tab by clicking on *Web* ► *QuickMapServices* ► *Search QMS*. This option of the plugin allows you to filter the available base maps by the current extent of the map canvas or using a search word.
2. Click on the *Filter by extent* and you should see one service available. If no service is found, zoom out and pan around the world (or your location) or search with a keyword.
3. Click on the *Add* button next to a returned dataset to load it.
4. The base map will be loaded and you will have a background for the map.



9.2.2 ??? Follow Along: The QuickOSM Plugin

With an incredible simple interface, the QuickOSM plugin allows you to download OpenStreetMap data.

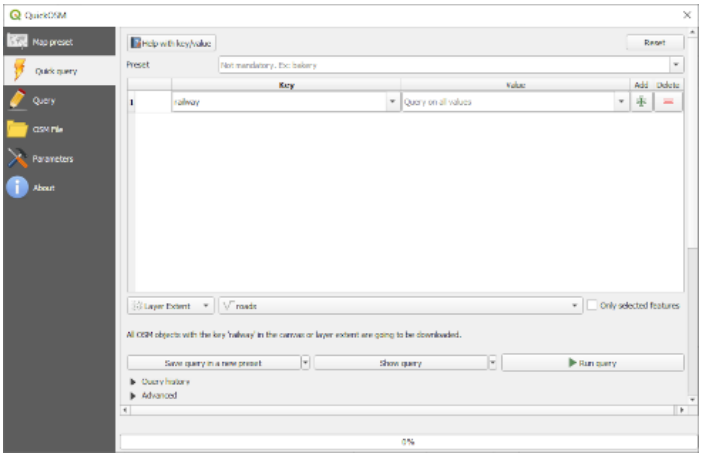
??? Follow Along: Using the Quick Query

1. Start a new empty project and add the *roads* layer from the `training_data` GeoPackage.
2. Install the **QuickOSM** plugin. The plugin adds two new buttons in the QGIS Toolbar and is accessible in the *Vector* ► *QuickOSM* menu.
3. Open the QuickOSM dialog. The plugin has many different tabs: we will use the *Quick Query* one.
4. You can download specific features by selecting a generic *Key* or be more specific and choose a specific *Key* and *Value* pair.

   Dica: if you are not familiar with the *Key* and *Value* system, click on the *Help with key/value* button. It will open a web page with a complete description of this concept of OpenStreetMap.

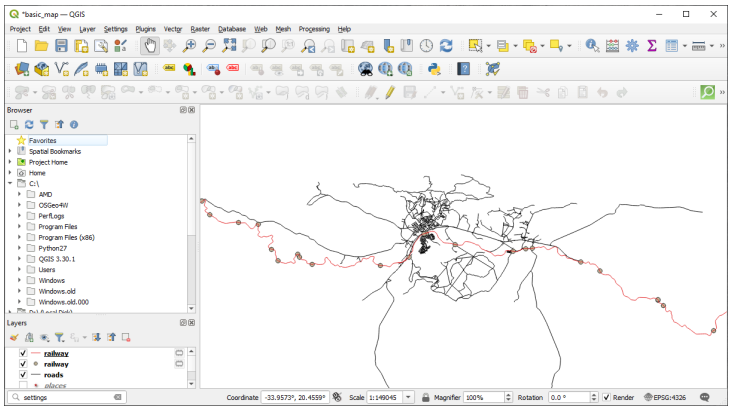
   ---

5. Look for *railway* in the *Key* menu and let the *Value* be empty: so we are downloading all the *railway* features without specifying any values.
6. Select *Layer Extent* in the next drop-down menu and choose *roads*.
7. Click on the *Run query* button.



After some seconds the plugin will download all the features tagged in OpenStreetMap as `railway` and load them directly into the map.

Nothing more! All the layers are loaded in the legend and are shown in the map canvas.



Aviso: QuickOSM creates temporary layer when downloading the data. If you want to save them permanently, click on the icon next to the layer and choose the options you prefer. Alternatively you can open the *Advanced* menu in QuickOSM and choose where to save the data in the *Directory* menu.

??? Follow Along: The QuickOSM Query engine

The quickest way to download data from QuickOSM plugin is using the *Quick query* tab and set some small parameters. But if you need some more specific data?

If you are an OpenStreetMap query master you can use QuickOSM plugin also with your personal queries.

QuickOSM has an incredible data parser that, together with the amazing query engine of Overpass, lets you download data with your specific needs.

For example: we want to download the mountain peaks that belongs into a specific mountain area known as Dolomites.

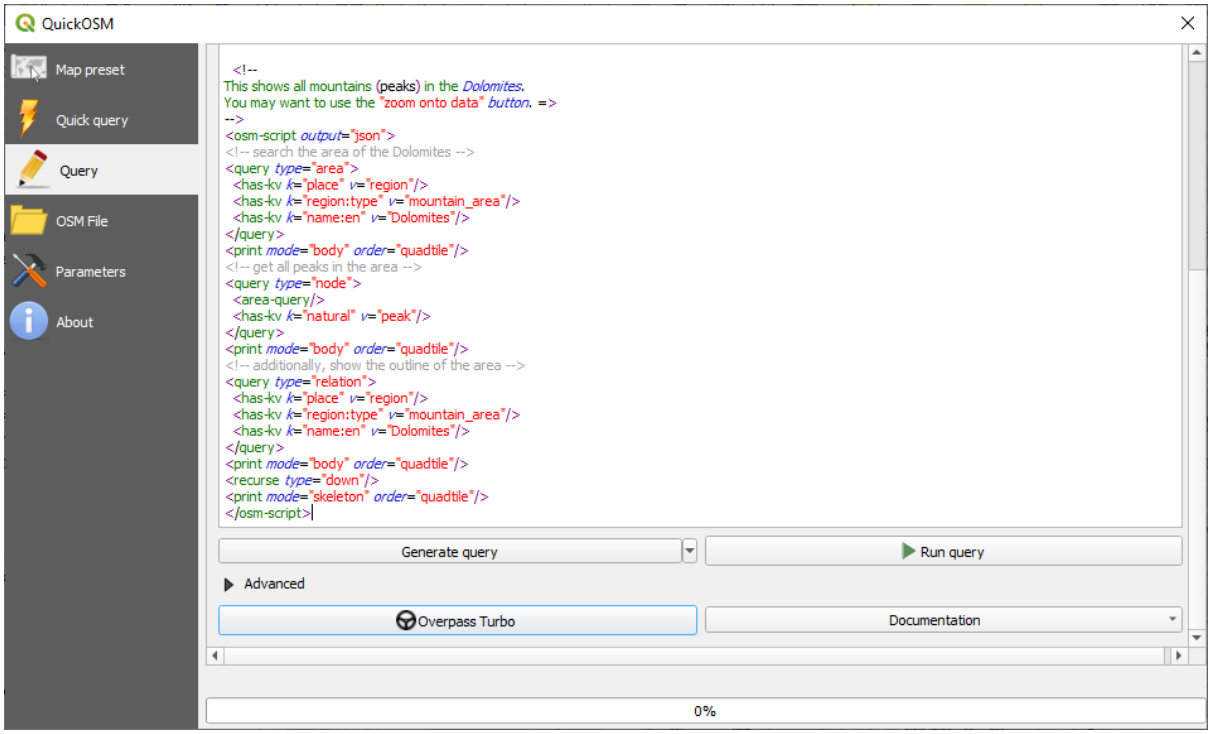
You cannot achieve this task with the *Quick query* tab, you have to be more specific and write your own query. Let's try to do this.

1. Start a new project.
2. Open the QuickOSM plugin and click on the *Query* tab.
3. Copy and paste the following code into the query canvas:

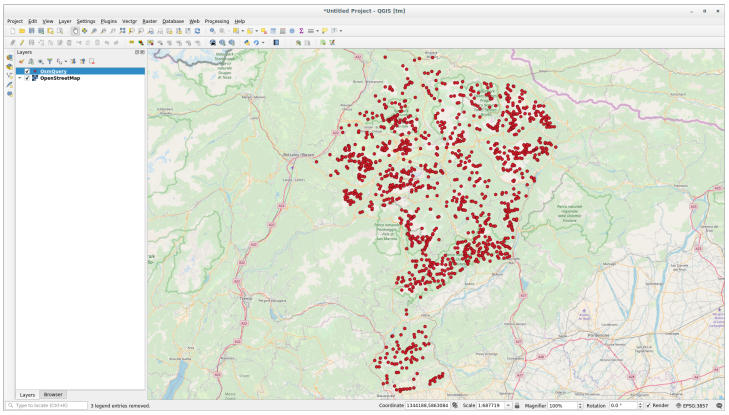
```
  <!--
This shows all mountains (peaks) in the Dolomites.
You may want to use the "zoom onto data" button. =>
-->
<osm-script output="json">
<!-- search the area of the Dolomites -->
<query type="area">
  <has-kv k="place" v="region"/>
  <has-kv k="region:type" v="mountain_area"/>
  <has-kv k="name:en" v="Dolomites"/>
</query>
<print mode="body" order="quadtile"/>
<!-- get all peaks in the area -->
<query type="node">
  <area-query/>
  <has-kv k="natural" v="peak"/>
</query>
<print mode="body" order="quadtile"/>
<!-- additionally, show the outline of the area -->
<query type="relation">
  <has-kv k="place" v="region"/>
  <has-kv k="region:type" v="mountain_area"/>
  <has-kv k="name:en" v="Dolomites"/>
</query>
<print mode="body" order="quadtile"/>
<recurse type="down"/>
<print mode="skeleton" order="quadtile"/>
</osm-script>
```

Nota: This query is written in a `xml` like language. If you are more used to the `Overpass QL` you can write the query in this language.

4. And click on *Run Query*:



The mountain peaks layer will be downloaded and shown in QGIS:



You can write complex queries using the Overpass Query language. Take a look at some example and try to explore the query language.

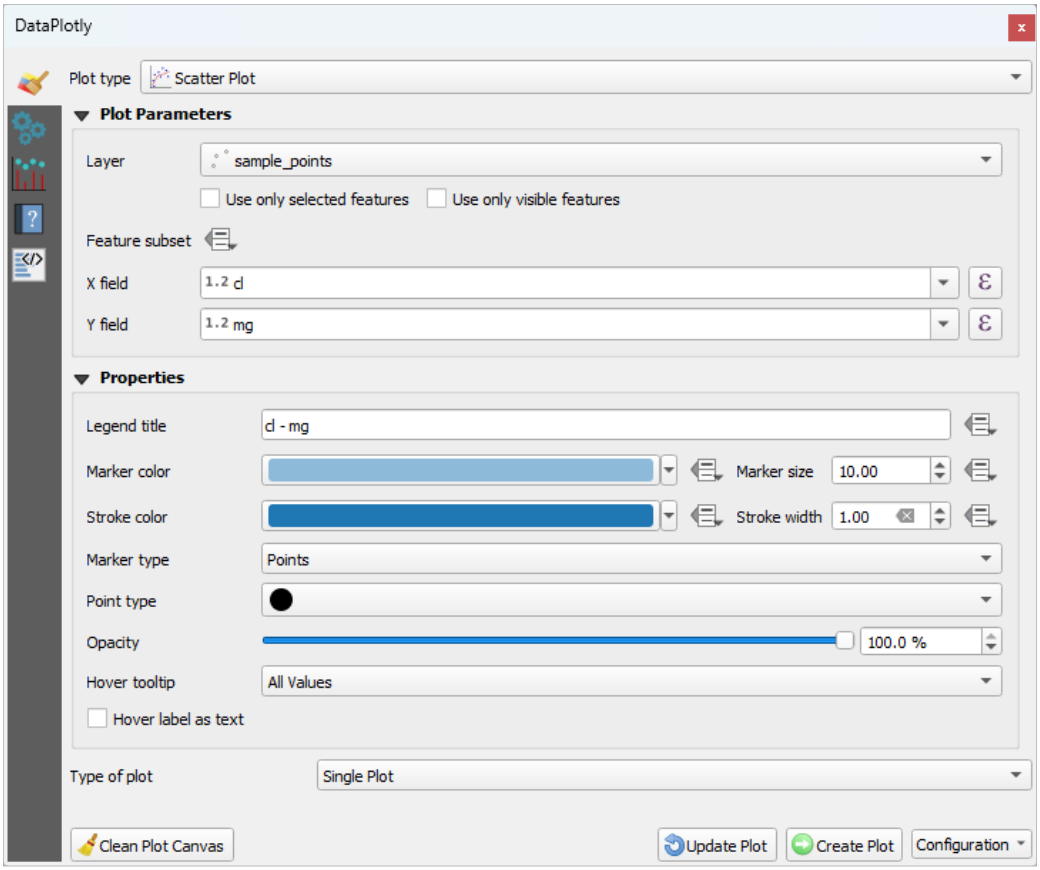
9.2.3 ??? Follow Along: The DataPlotly Plugin

The **DataPlotly** plugin allows you to create D3 plots of vector attributes data thanks to the plotly library.

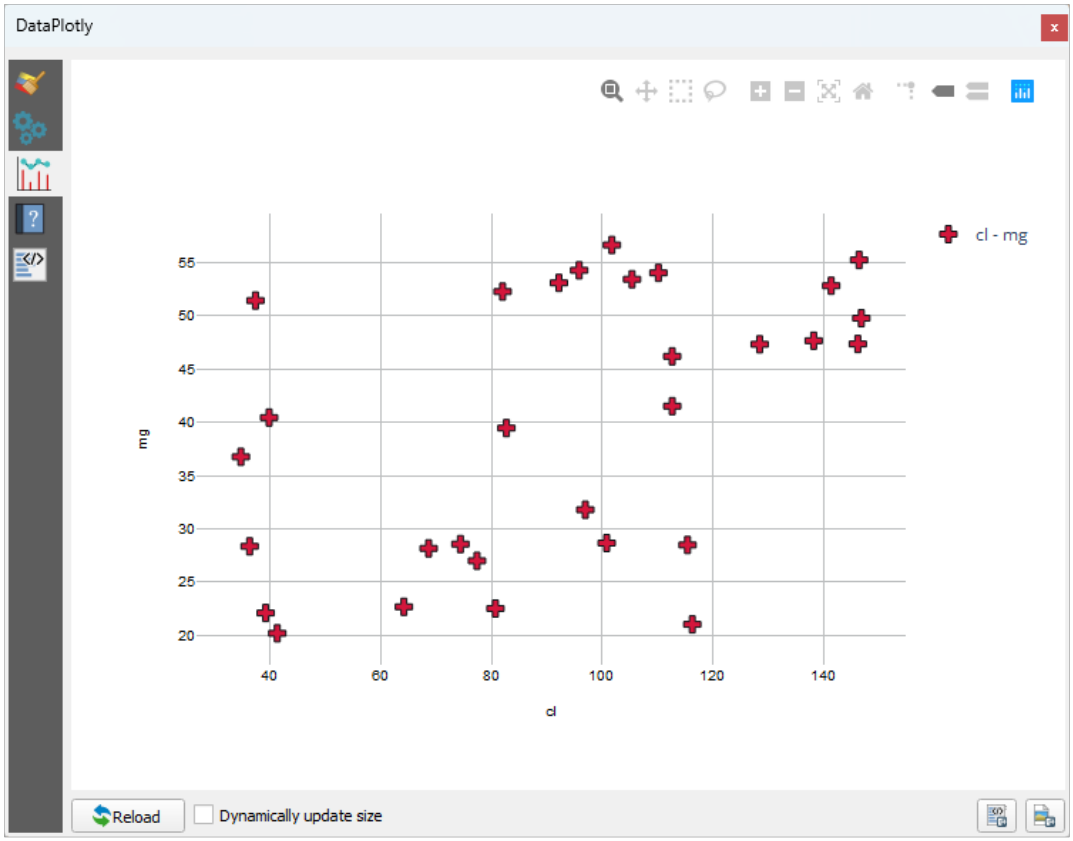
1. Start a new project
2. Load the *sample_points* layer from the `exercise_data/plugins` folder
3. Install the plugin following the guidelines described in *??? Follow Along: Installing New Plugins* searching *Data Plotly*
4. Open the plugin by clicking on the new icon in the toolbar or in the *Plugins* ► *Data Plotly* menu

In the following example we are creating a simple `Scatter Plot` of two fields of the *sample_points* layer. In the DataPlotly Panel:

1. Choose *sample_points* in the Layer filter, *cl* for the *X Field* and *mg* for the *Y Field*:



2. If you want you can change the colors, the marker type, the transparency and many other settings: try to change some parameters to create the plot below.



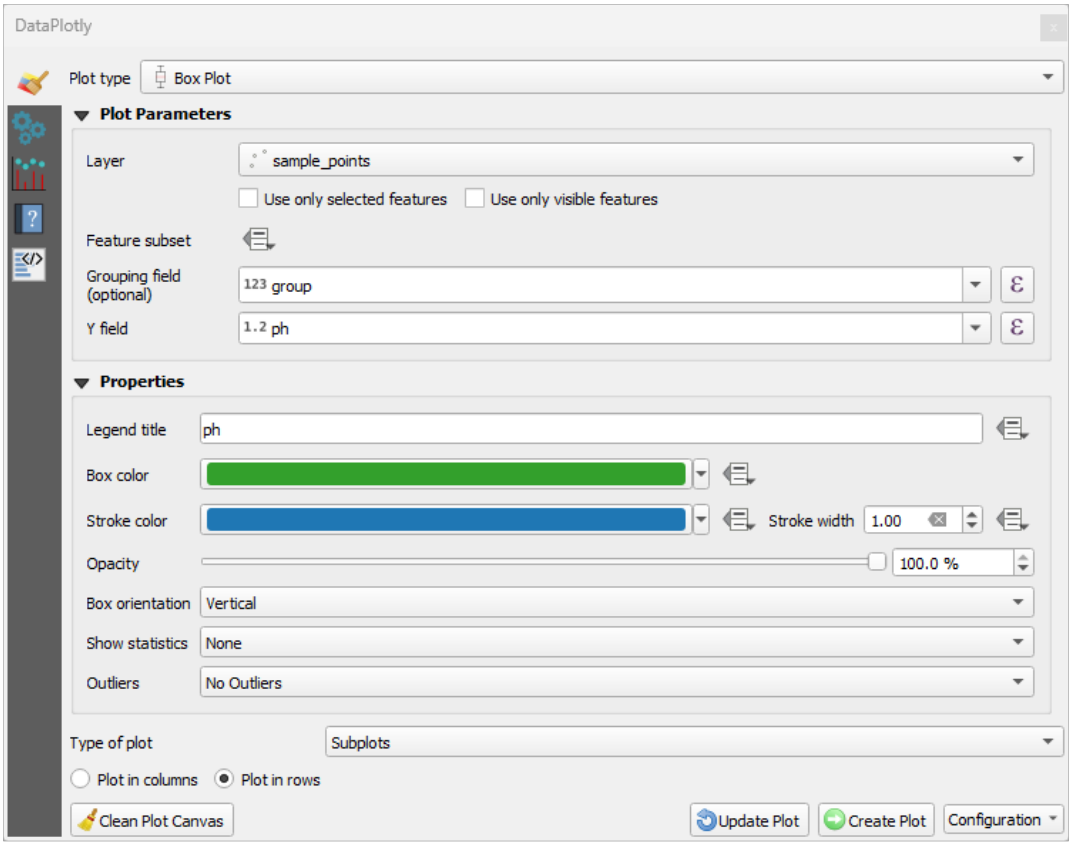
3. Once you have set all the parameters, click on the *Create Plot* button to create the plot.

The plot is interactive: this means you can use all the upper buttons to resize, move, or zoom in/out the plot canvas. Moreover, each element of the plot is interactive: by clicking or selecting one or more point on the plot, the corresponding point(s) will be selected in the plot canvas.

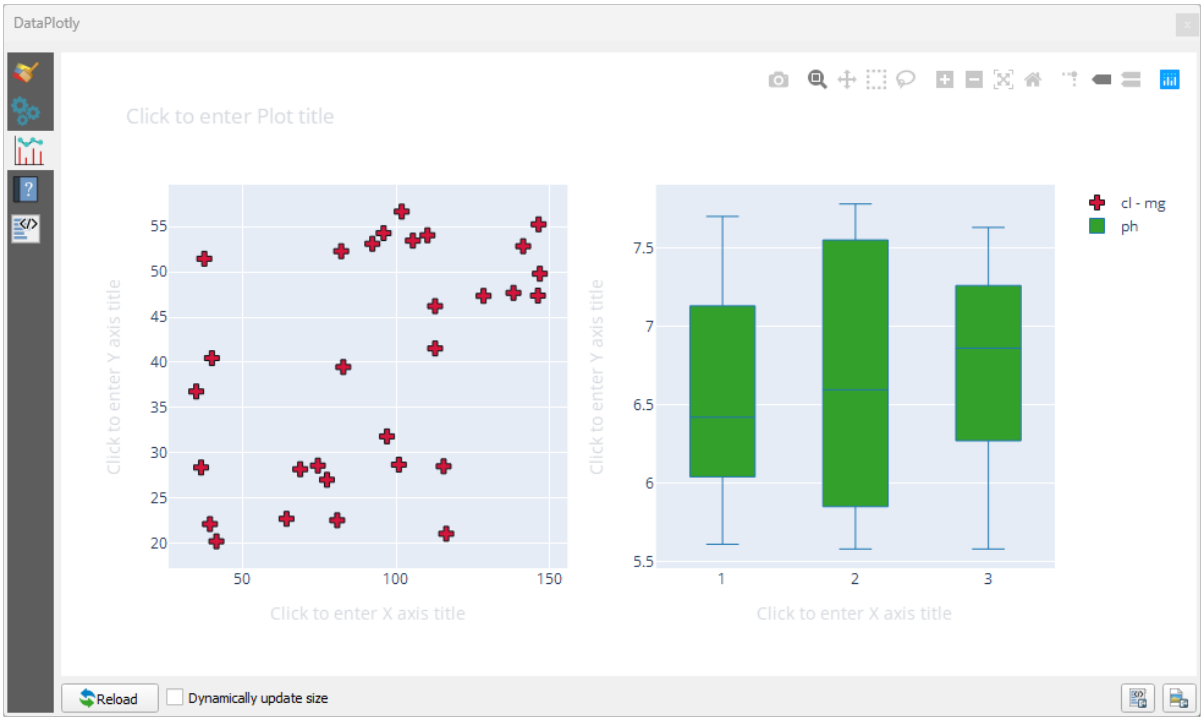
You can save the plot as a `png` static image or as an `html` file by clicking on the or on the button in the lower right corner of the plot.

There is more. Sometimes it can be useful to have two (or more) plots showing different plot types with different variables on the same page. Let's do this!

1. Go back to the main plot settings tab by clicking on the button in the upper left corner of the plugin panel
2. Change the *Plot Type* to *Box Plot*
3. Choose *group* as *Grouping Field* and *ph* as *Y Field*
4. In the lower part of the panel, change the *Type of Plot* from *SinglePlot* to *SubPlots* and let the default option *Plot in Rows* selected.



5. Once done click on the *Create Plot* button to draw the plot



Now both scatter plot and box plot are shown in the same plot page. You still have the chance to click on each plot item and select the corresponding features in the map canvas.

Dica: Each plot has its own manual page available in the tab. Try to explore all the plot types and see all the other settings available.

9.2.4 In Conclusion

There are many useful plugins available for QGIS. Using the built-in tools for installing and managing these plugins, you can find new plugins and make optimum use of them.

9.2.5 What's Next?

Next we'll look at how to use layers that are hosted on remote servers in real time.

CAPÍTULO 10

Module: Online Resources

When considering data sources for a map, there is no need to be restricted to data which you have saved on the computer you're working on. There are online data sources which you can load data from as long as you are connected to the Internet.

In this module, we'll cover two kinds of web-based GIS services: Web Mapping Services (WMS) and Web Feature Services (WFS).

10.1 Lesson: Web Mapping Services

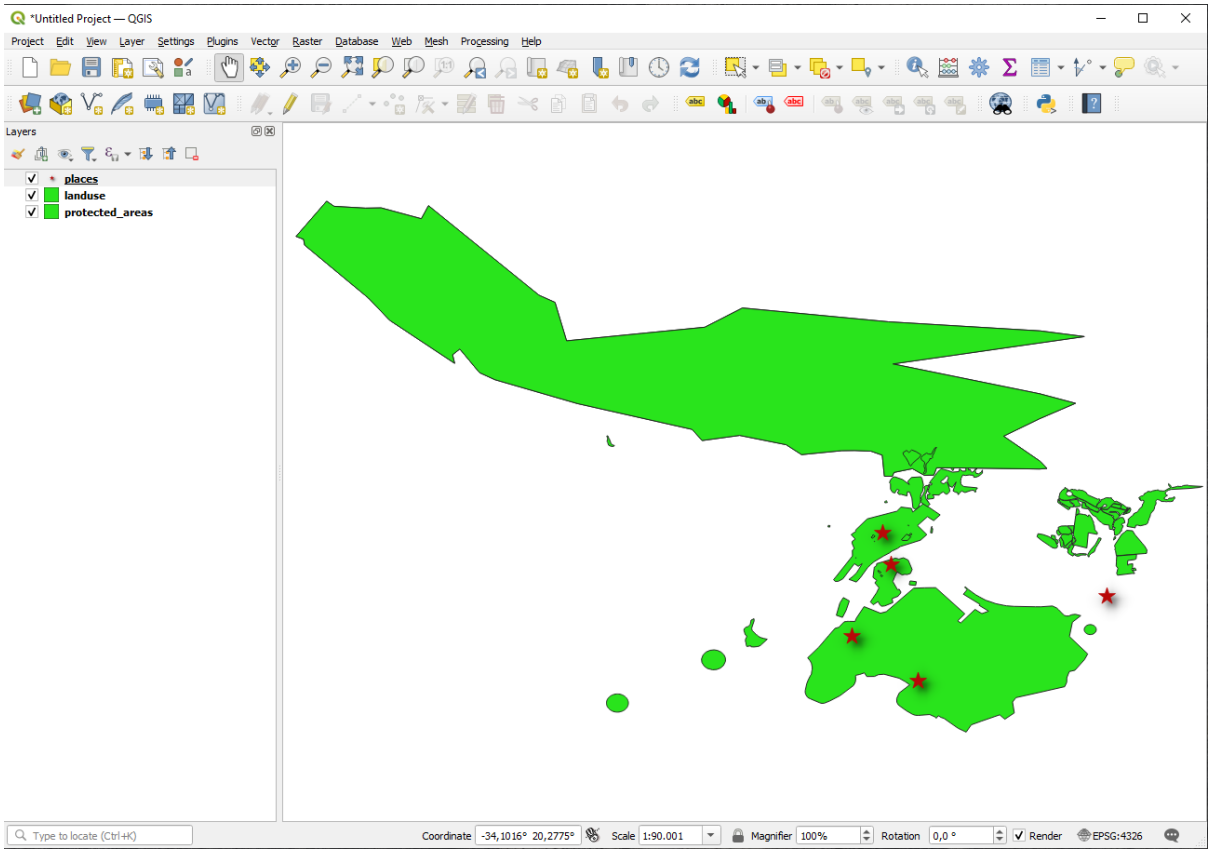
A Web Mapping Service (WMS) is a service hosted on a remote server. Similar to a website, you can access it as long as you have a connection to the server. Using QGIS, you can load a WMS directly into your existing map.

From the lesson on plugins, you will remember that it's possible to load a new raster image from Google, for example. However, this is a once-off transaction: once you have downloaded the image, it doesn't change. A WMS is different in that it's a live service that will automatically refresh its view if you pan or zoom on the map.

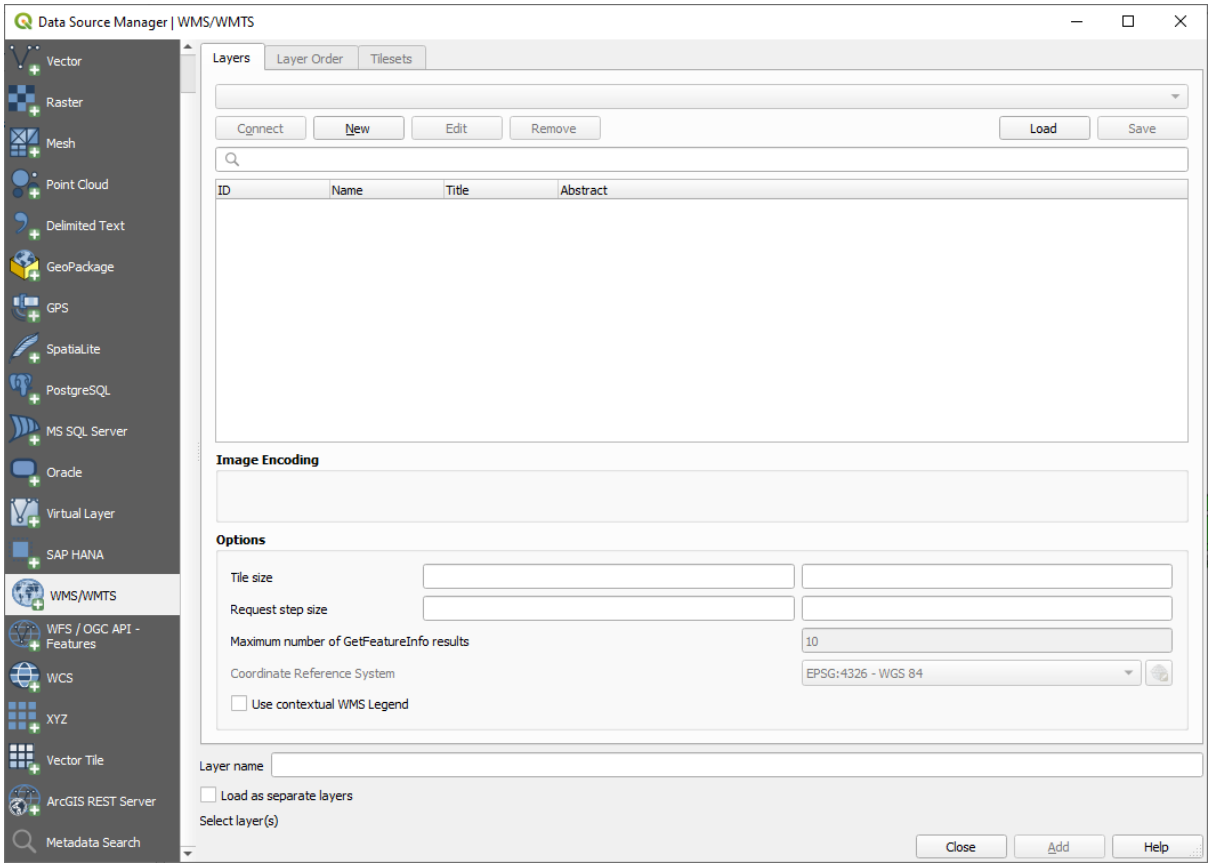
The goal for this lesson: To use a WMS and understand its limitations.

10.1.1 ??? Follow Along: Loading a WMS Layer

For this exercise, you can either use the basic map you made at the start of the course, or just start a new map and load some existing layers into it. For this example, we used a new map and loaded the original *places*, *landuse* and *protected_areas* layers and adjusted the symbology:



1. Load these layers into a new map, or use your original map with only these layers visible.
2. Before starting to add the WMS layer, deactivate «on the fly» projection (*Project* ► *Properties…* ► *CRS* tab, check *No CRS (or unknown/non-Earth projection)*. This may cause the layers to no longer overlap properly, but don't worry: we'll fix that later.
3. To add WMS layers, click on the button to open the *Data Source Manager* dialog and enable the *WMS/WMTS* tab.

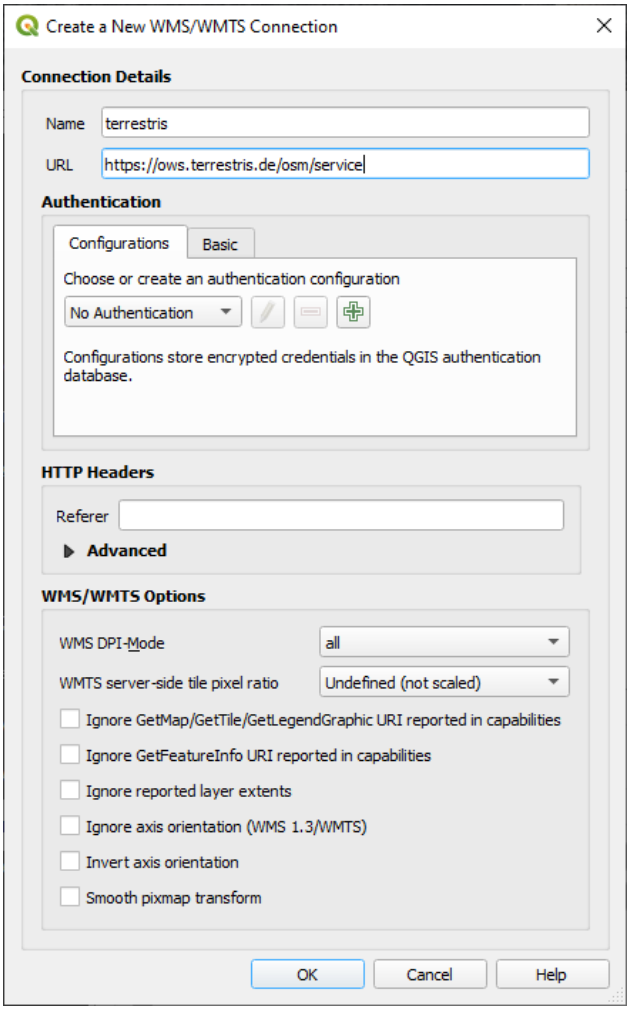


Remember how you connected to a SpatiaLite or GeoPackage database at the beginning of the course. The *landuse*, *buildings*, and *roads* layers are stored in a database. To use those layers, you first needed to connect to the database. Using a WMS is similar, with the exception that the layers are on a remote server.

4. To create a new connection to a WMS, click on the *New* button.

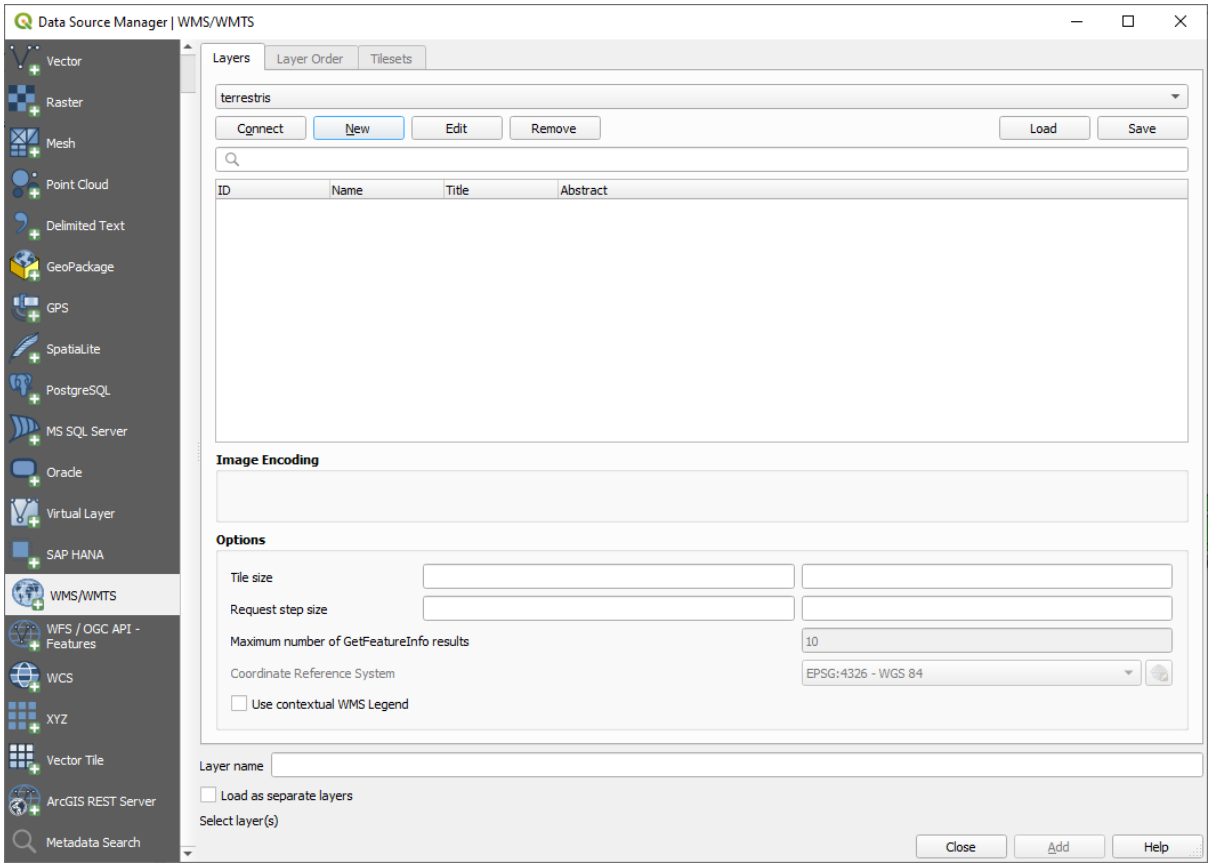
   You'll need a WMS address to continue. There are several free WMS servers available on the Internet. One of these is terrestris, which makes use of the OpenStreetMap dataset.

5. To make use of this WMS, set it up in your current dialog, like this:

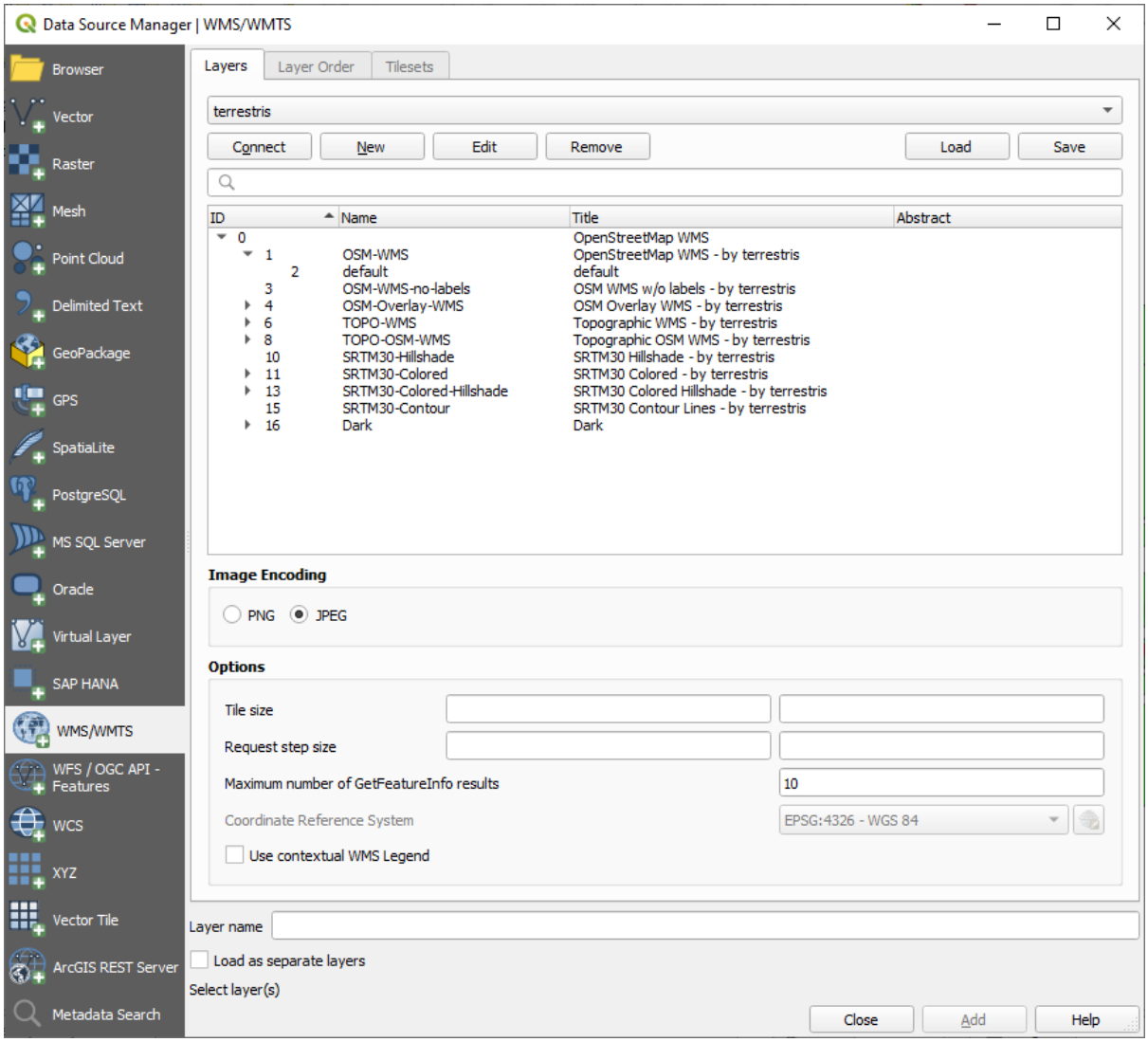


- The value of the *Name* field should be `terrestris`.
- The value of the *URL* field should be `https://ows.terrestris.de/osm/service`.

6. Click *OK*. You should see the new WMS server listed:

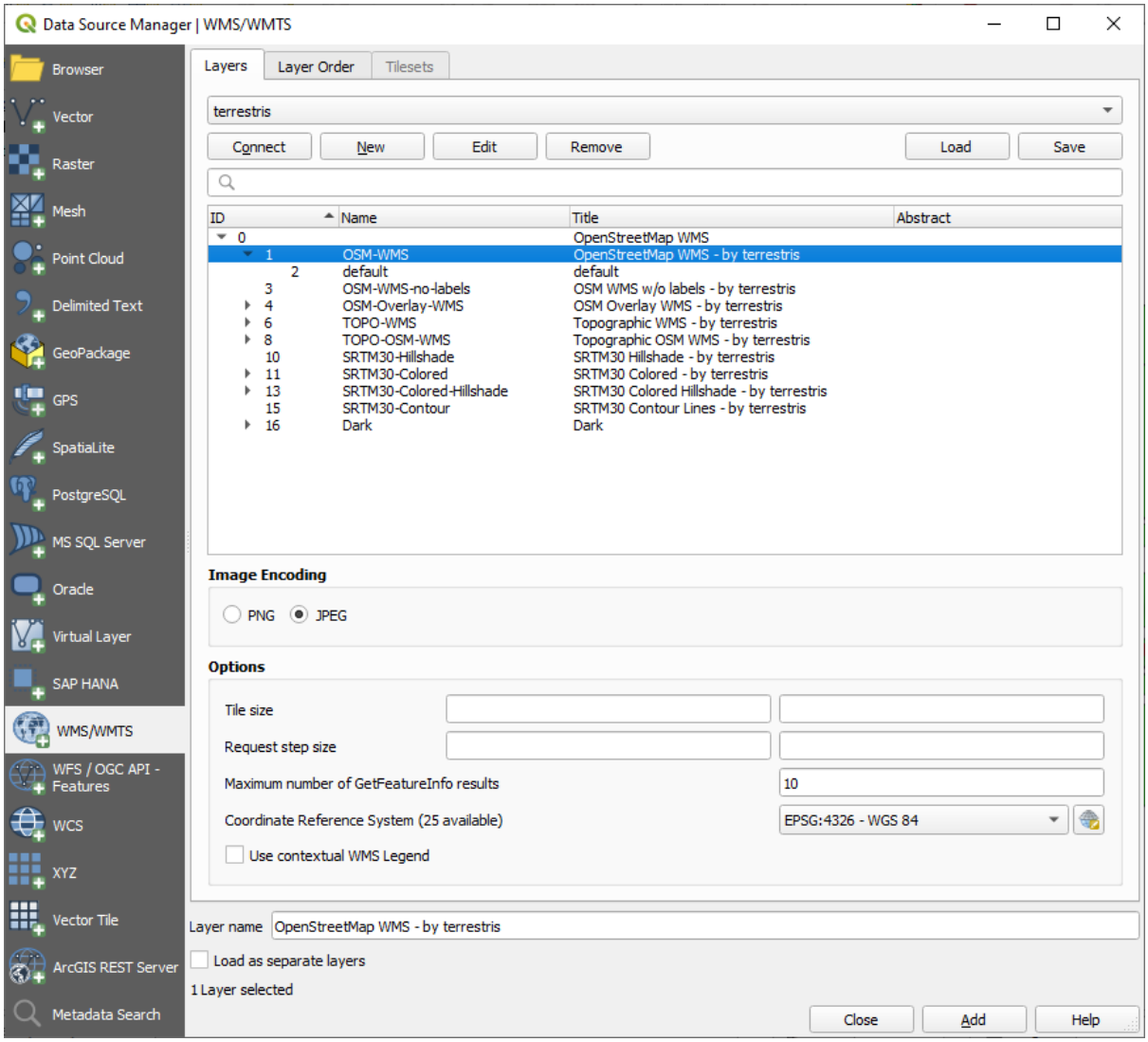


7. Click *Connect*. In the list below, you should now see these new entries loaded:



These are all the layers hosted by this WMS server.

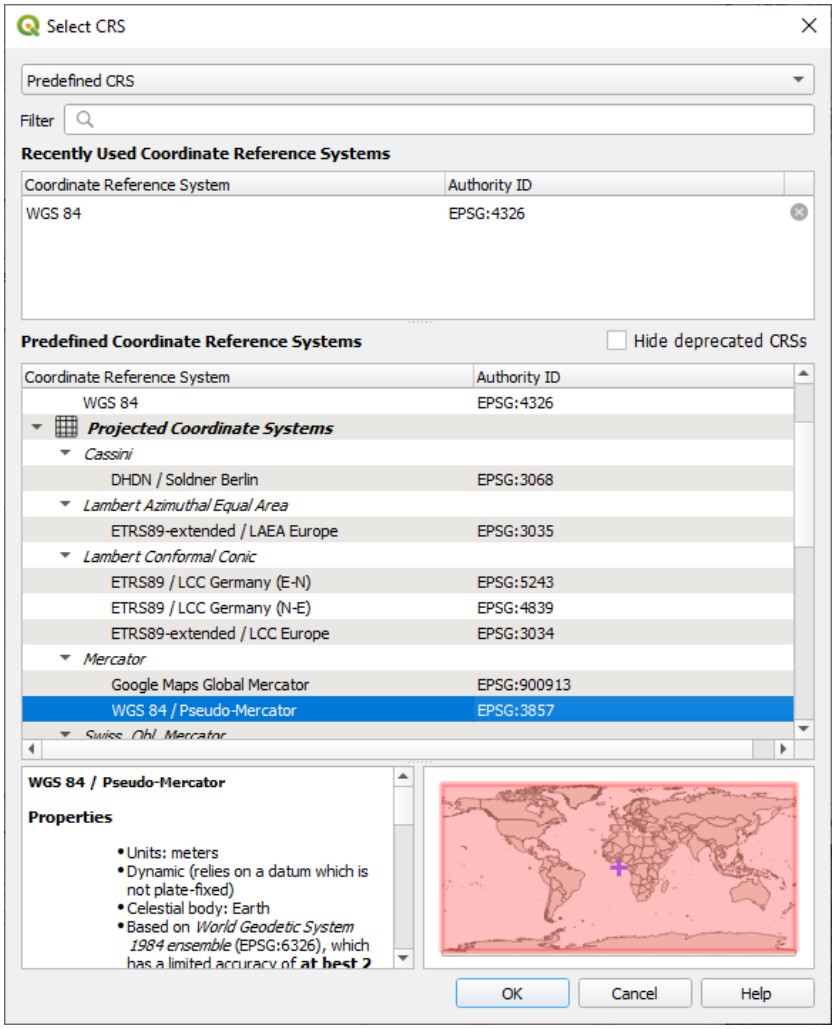
8. Click once on the *OSM-WMS* layer. This will display the default *Coordinate Reference System* in use and the number of CRSes supported by the dataset:



Since we're not using `EPSG:4326 - WGS 84` for our map, let's find one that meets our needs.

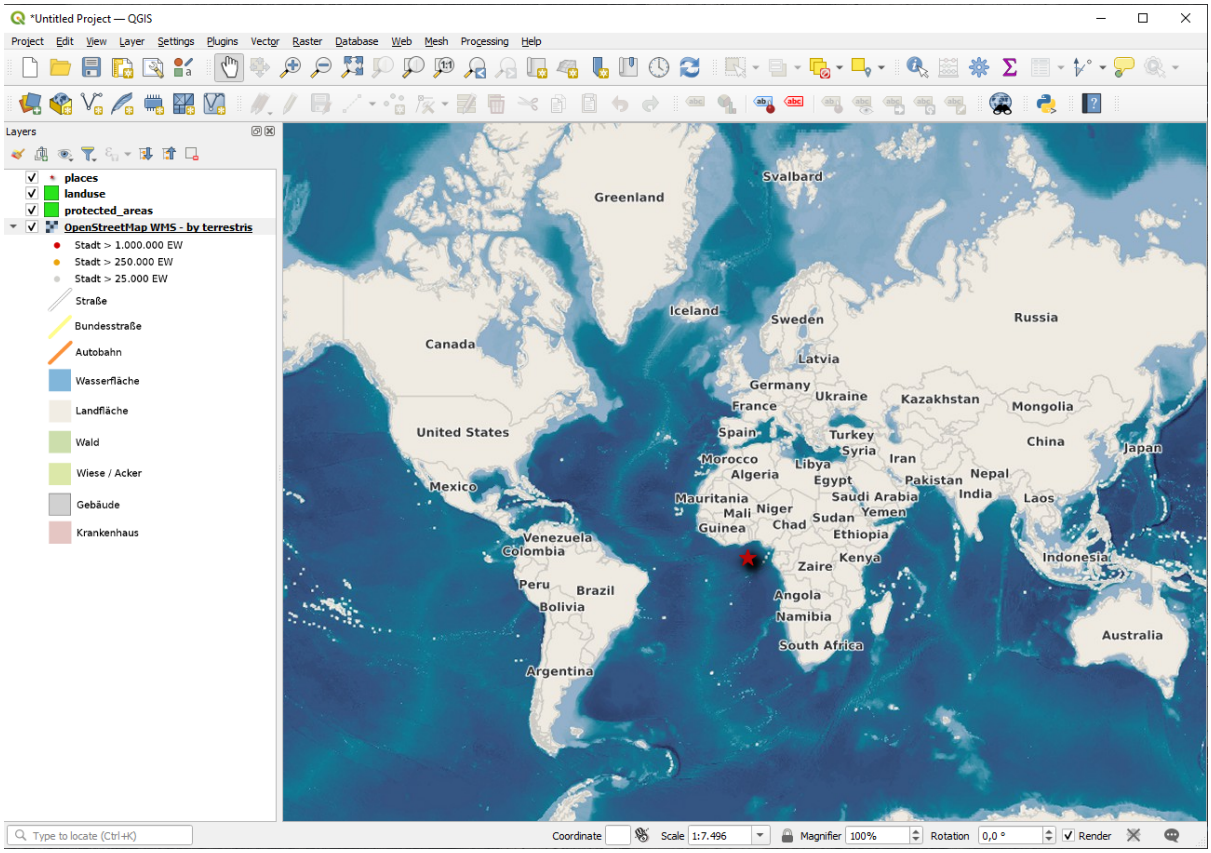
1. Click the

button. You will see a standard *Coordinate Reference System Selector* dialog showing all the CRS the data is published with.
2. We want a *projected* CRS, so let's choose *WGS 84 / Pseudo-Mercator* with `EPSG:3857`. You can use the top *Filter* widget.



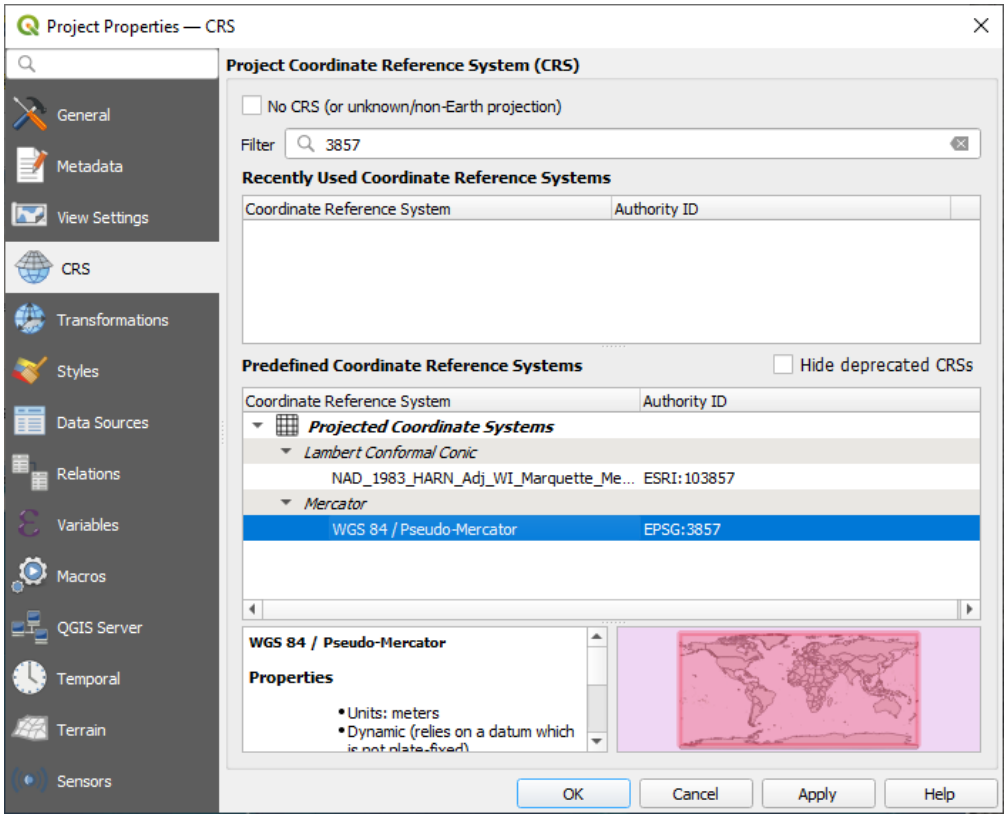
1. Click *OK*. The Coordinate Reference System associated with the entry has changed.

9. Click *Add* to load the layer in your project using *Layer name*` (default is *OpenStreetMap WMS - by terrestris*).
10. Close the *Data Source Manager* dialog if not done automatically
11. In the *Layers* panel, click and drag it to the bottom of the list.
12. Zoom out in order to get a global view of the layers. You will notice that your layers aren't located correctly (near west of Africa). This is because «on the fly» projection is disabled.



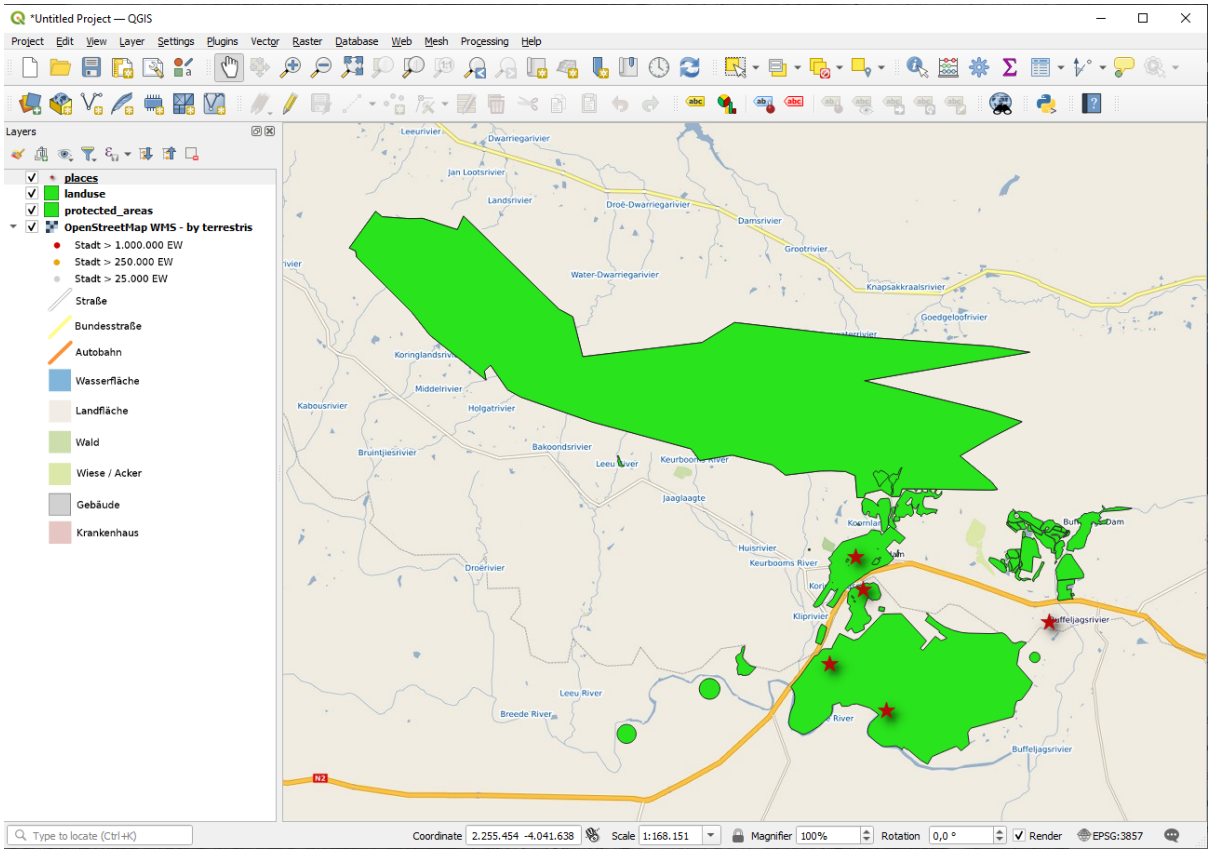
13. Let's enable the reprojection again, but using the same projection as the *OpenStreetMap WMS* layer, which is *WGS 84 / Pseudo Mercator*.

    1. Open the *Project* ► *Properties...* ► *CRS* tab
    2. Uncheck *No CRS (or unknown/non-Earth projection)*
    3. Choose *WGS 84 / Pseudo Mercator* from the list.



4. Click *OK*.

14. Now right-click on one of your own layers in the *Layers* panel and click *Zoom to layer(s)*. You should see the Swellendam area:



Note how the WMS layer's streets and our own streets overlap. That's a good sign!

The nature and limitations of WMS

By now you may have noticed that this WMS layer actually has many features in it. It has streets, rivers, nature reserves, and so on. What's more, even though it looks like it's made up of vectors, it seems to be a raster, but you can't change its symbology. Why is that?

This is how a WMS works: it's a map, similar to a normal map on paper, that you receive as an image. What usually happens is that you have vector layers, which QGIS renders as a map. But using a WMS, those vector layers are on the WMS server, which renders it as a map and sends that map to you as an image. QGIS can display this image, but can't change its symbology, because all that is handled on the server.

This has several advantages, because you don't need to worry about the symbology. It's already worked out, and should be nice to look at on any competently designed WMS.

On the other hand, you can't change the symbology if you don't like it, and if things change on the WMS server, then they'll change on your map as well. This is why you sometimes want to use a Web Feature Service (WFS) instead, which gives you vector layers separately, and not as part of a WMS-style map.

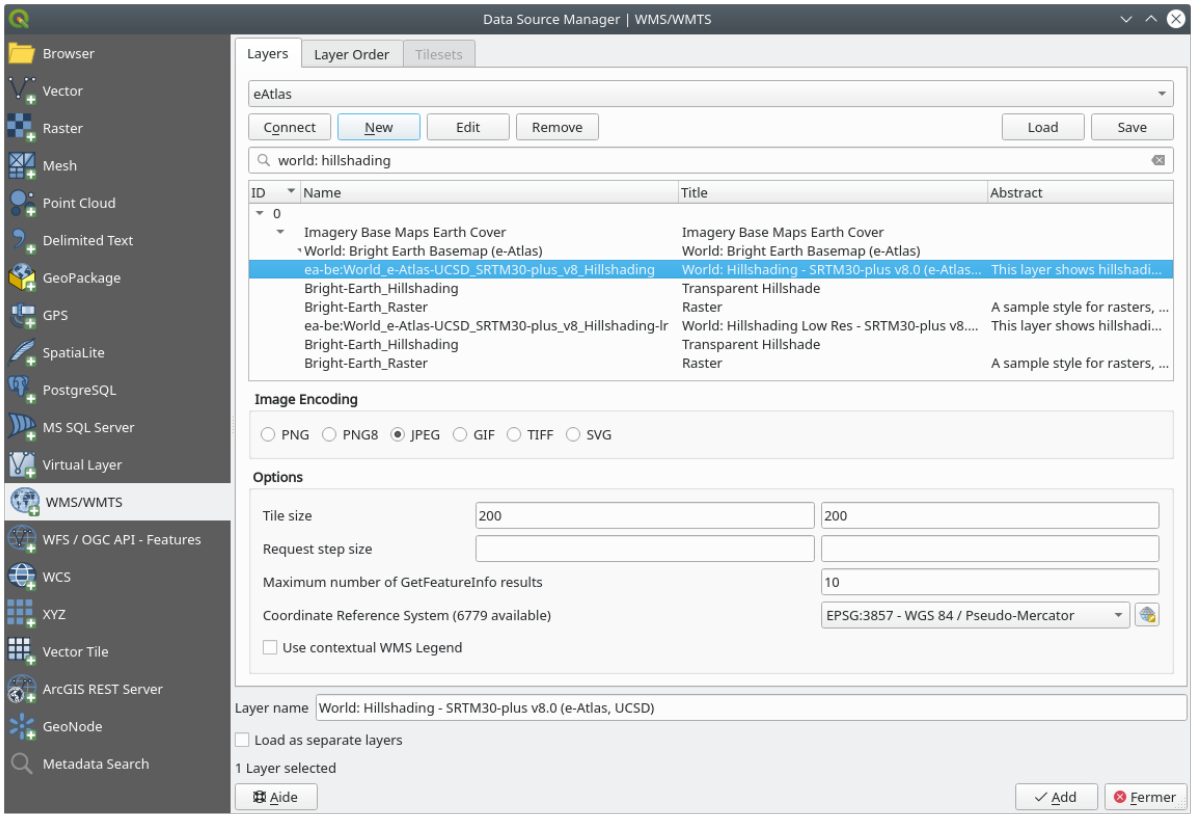
This will be covered in the next lesson, however. First, let's add another WMS layer.

10.1.2 ??? Try Yourself:

1. Add the `eAtlas` WMS server at this URL: `https://maps.eatlas.org.au/maps/wms`
2. Load a *World: Hillshading* layer into the map.
3. You might want to set its *Encoding* to *JPEG* and its *Tile size* option to `200` by `200`, so that it loads faster.
4. Your map should look like this (you may need to re-order the layers, and apply some transparency):

Answer

1. Go to the Data Source Manager, WMS / WMTS tab and create a new connection entry
2. Use the text box to filter the list of layers and select the corresponding layer
3. Remember to check/turn its CRS into *EPSG:3857 - WGS 84 / Pseudo Mercator* as the rest of the map.



4. After the layer is loaded, you can modify its *Opacity* value (under the *Transparency* properties tab)

10.1.3 ??? Try Yourself:

Part of the difficulty of using WMS is finding a good (free) server.

- Find a new WMS at directory.spatineo.com (or elsewhere online). It must not have associated fees or restrictions, and must have coverage over the Swellendam study area.

  Remember that what you need in order to use a WMS is only its URL (and preferably some sort of description).

Answer

Spatineo is one of many places you can search for a OGC data. Type in the text area a name, keyword, location of your interest to see if you get some working results. For this lesson, you might want to filter the results to include WMS only.

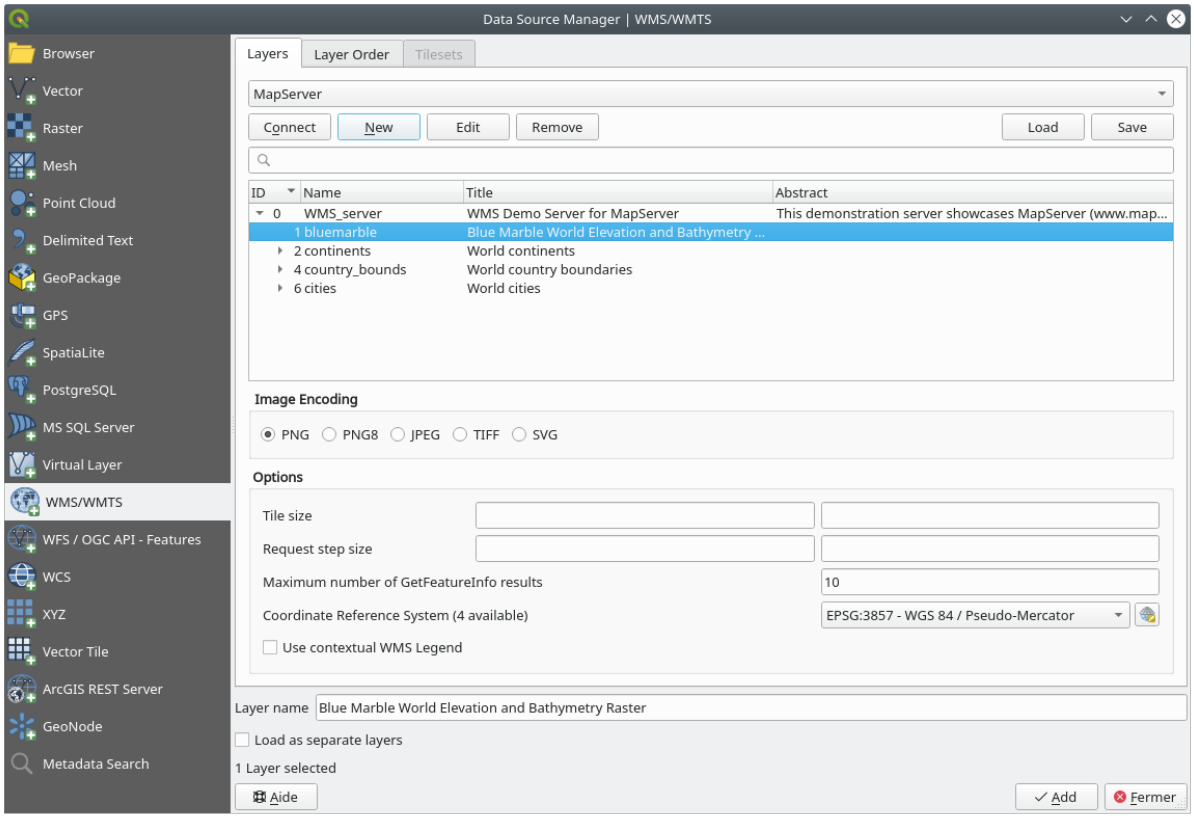
You may notice that many WMS servers are not always available. Sometimes this is temporary, sometimes it is permanent. An example of a WMS server that worked at the time of writing is the *MapServer Demonstration Server* WMS at https://demo.mapserver.org/cgi-bin/wms?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities. It does not require fees or have access constraints, and it is global. Therefore, it does satisfy the requirements. Keep in mind, however, that this is merely an example. There are many other WMS servers to choose from.

10.1.4 ??? Try Yourself:

- Add the *bluemarble* layer from the *MapServer Demonstration Server*. Is that a suitable dataset for our study area?

Answer

1. Hide all other WMS layers to prevent them from rendering unnecessarily in the background.
2. Use the same approach as before to add the new server and the appropriate layer as hosted on that server:



3. If you zoom into the Swellendam area, you'll notice that this dataset has a low resolution:

Therefore, it's better not to use this data for the current map. The Blue Marble data is more suitable at global or national scales

10.1.5 In Conclusion

Using a WMS, you can add inactive maps as backdrops for your existing map data.

10.1.6 Further Reading

- Spatineo Directory
- OpenStreetMap.org list of WMS servers

10.1.7 What's Next?

Now that you've added an inactive map as a backdrop, you'll be glad to know that it's also possible to add features (such as the other vector layers you added before). Adding features from remote servers is possible by using a Web Feature Service (WFS). That's the topic of the next lesson.

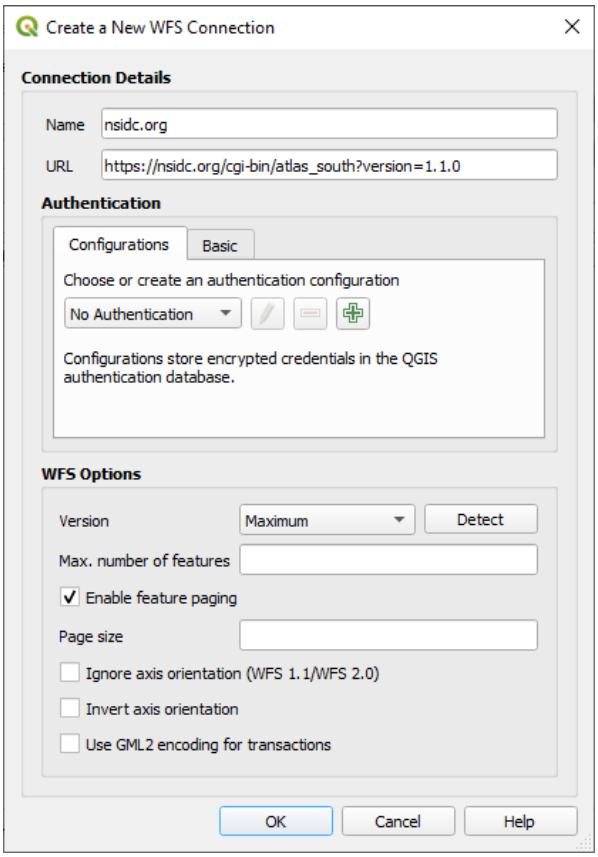
10.2 Lesson: Web Feature Services

A Web Feature Service (WFS) provides its users with GIS data in formats that can be loaded directly in QGIS. Unlike a WMS, which provides you only with a map which you can't edit, a WFS gives you access to the features themselves.

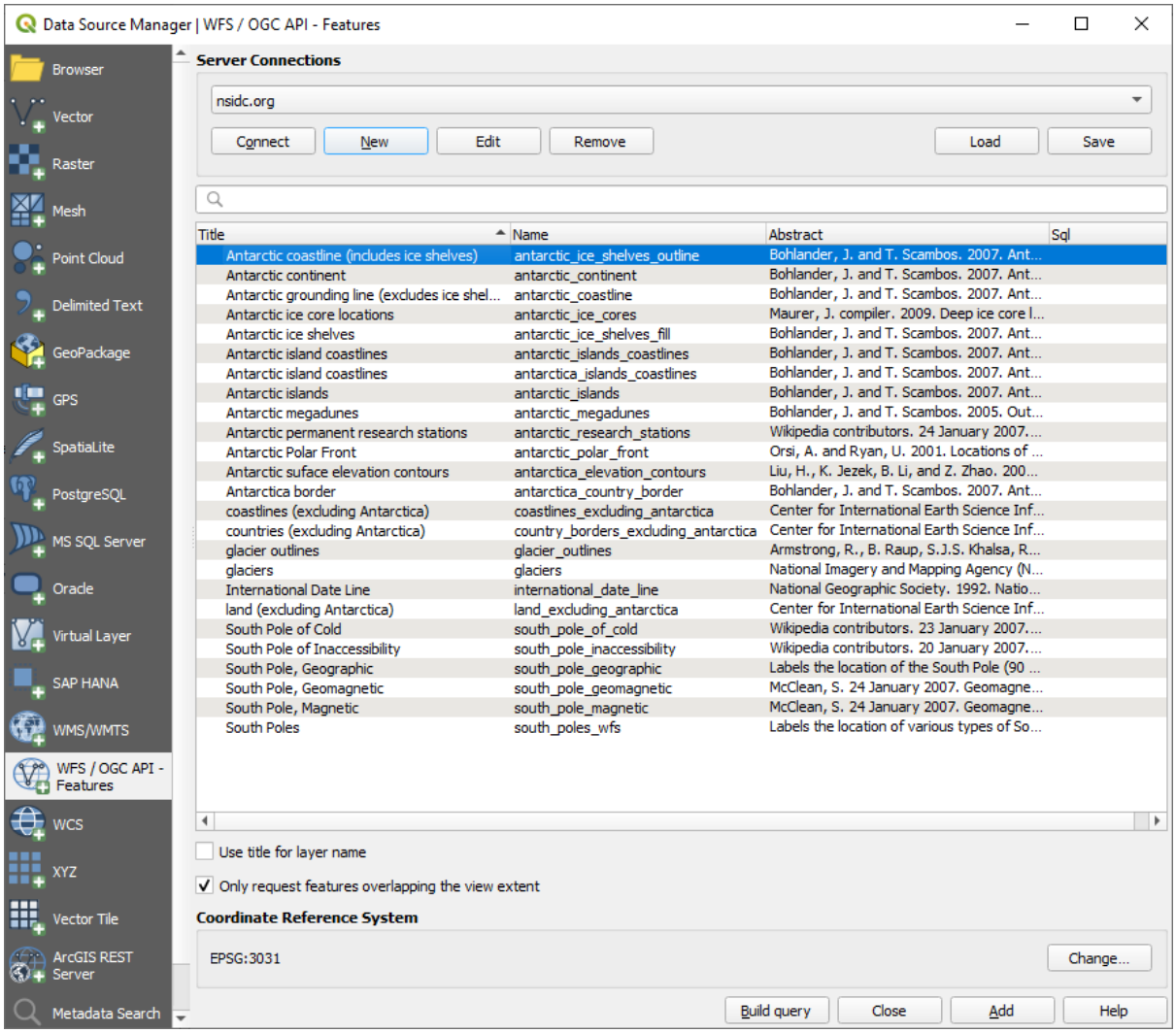
The goal for this lesson: To use a WFS and understand how it differs from a WMS.

10.2.1 ??? Follow Along: Loading a WFS Layer

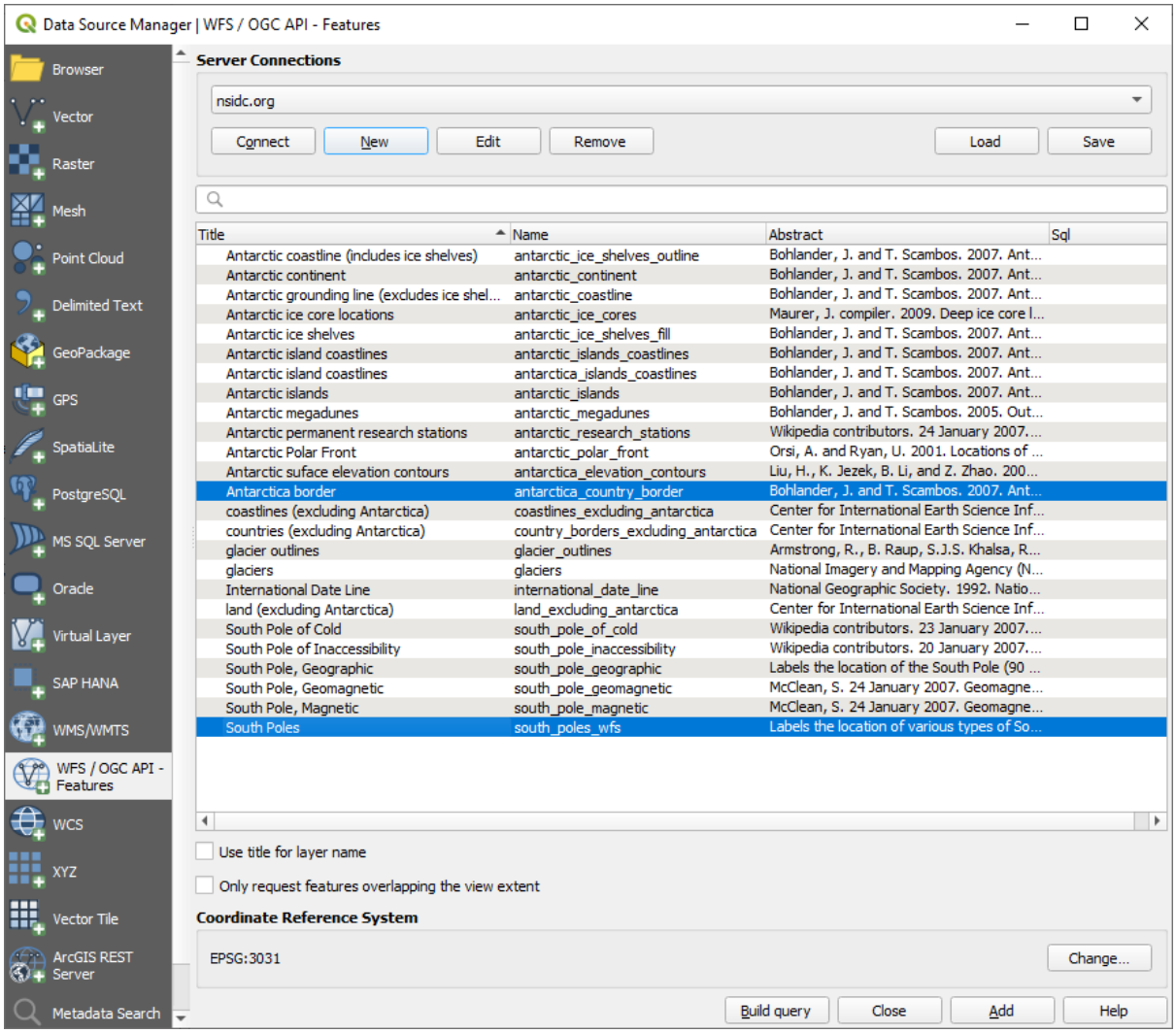
1. Start a new map. This is for demo purposes and won't be saved.
2. Click the Open Data Source Manager button.
3. Enable the *WFS / OGC API - Features* tab.
4. Click the *New* button.
5. In the dialog that appears, enter the *Name* as `nsidc.org` and the *URL* as `https://nsidc.org/cgi-bin/atlas_south?version=1.1.0`.



6. Click *OK*, and the new connection will appear in your *Server connections*.
7. Click the *Connect*. A list of the available layers will appear:

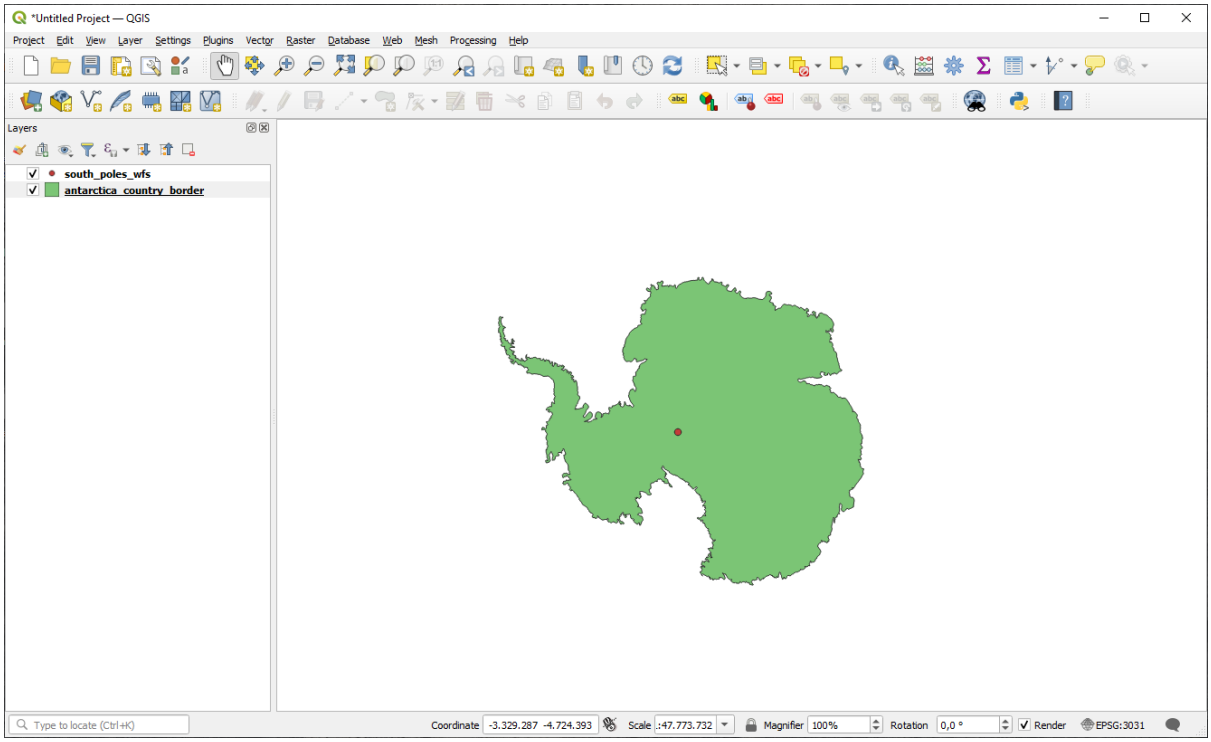


8. Uncheck the *Only request features overlapping the view extent* option below the layers list, since your current map canvas may not cover our area of interest: Antarctica.
9. Find the layer *antarctica_country_border*. You can use the *Filter* box at the top.
10. Click on the layer to select it:
11. Find and select also the layer *south_poles_wfs*. You might need to hold `Ctrl`.



12. Click *Add*.

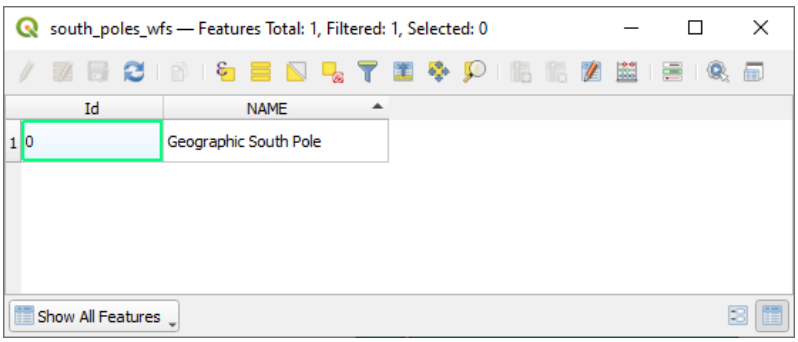
    It may take a while to load the layers. When they are loaded, they will appear in the map, showing the outlines of Antarctica and a few points over.



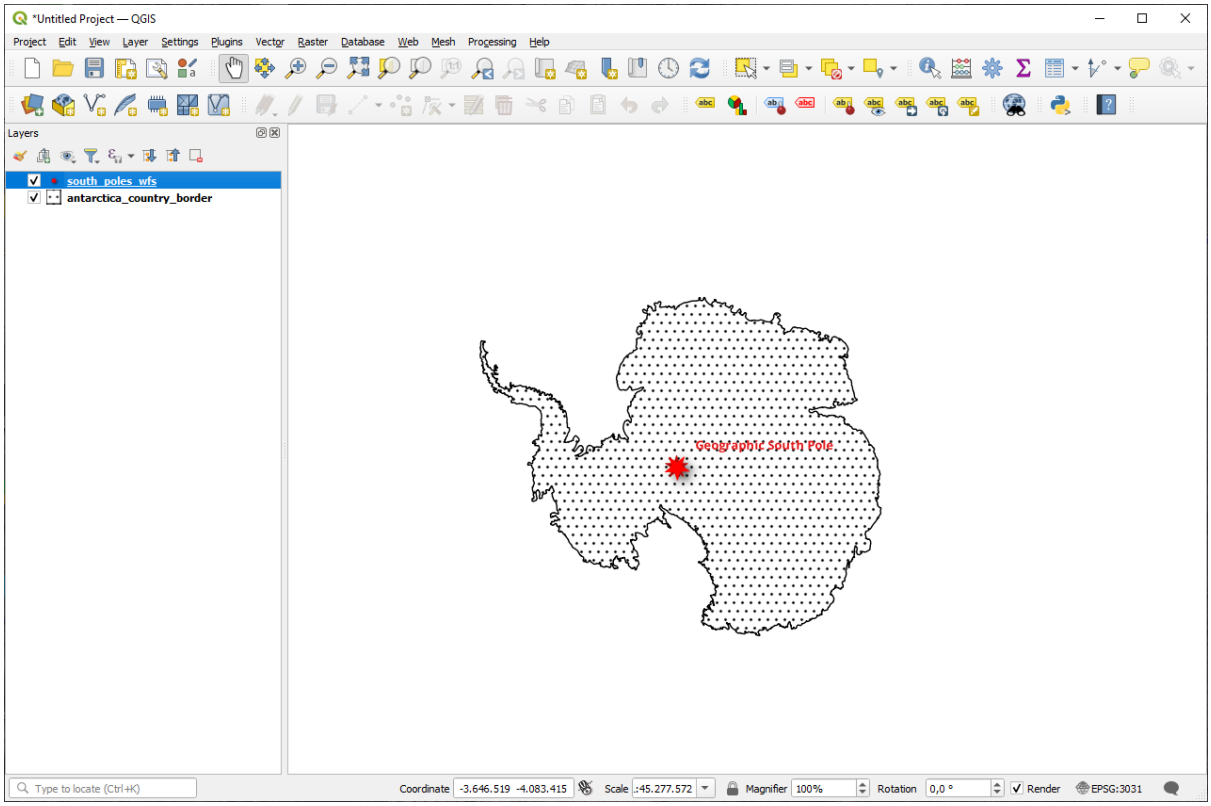
How is this different from having a WMS layer?

13. Select any of the layers and you'll notice that feature selection and attribute table tools are enabled. These are vector layers.

14. Select the *south_poles_wfs* layer and open its attribute table. You should see this:



Since the points have attributes, we are able to label them, as well as change their symbology. Here's an example:



Differences from WMS layers

A Web Feature Service returns the layer itself, not just a map rendered from it. This gives you direct access to the data, meaning that you can change its symbology and run analysis functions on it. However, this is at the cost of much more data being transmitted. This will be especially obvious if the layers you're loading have complicated shapes, a lot of attributes, or many features; or even if you're just loading a lot of layers. WFS layers typically take a very long time to load because of this.

10.2.2 ??? Follow Along: Querying a WFS Layer

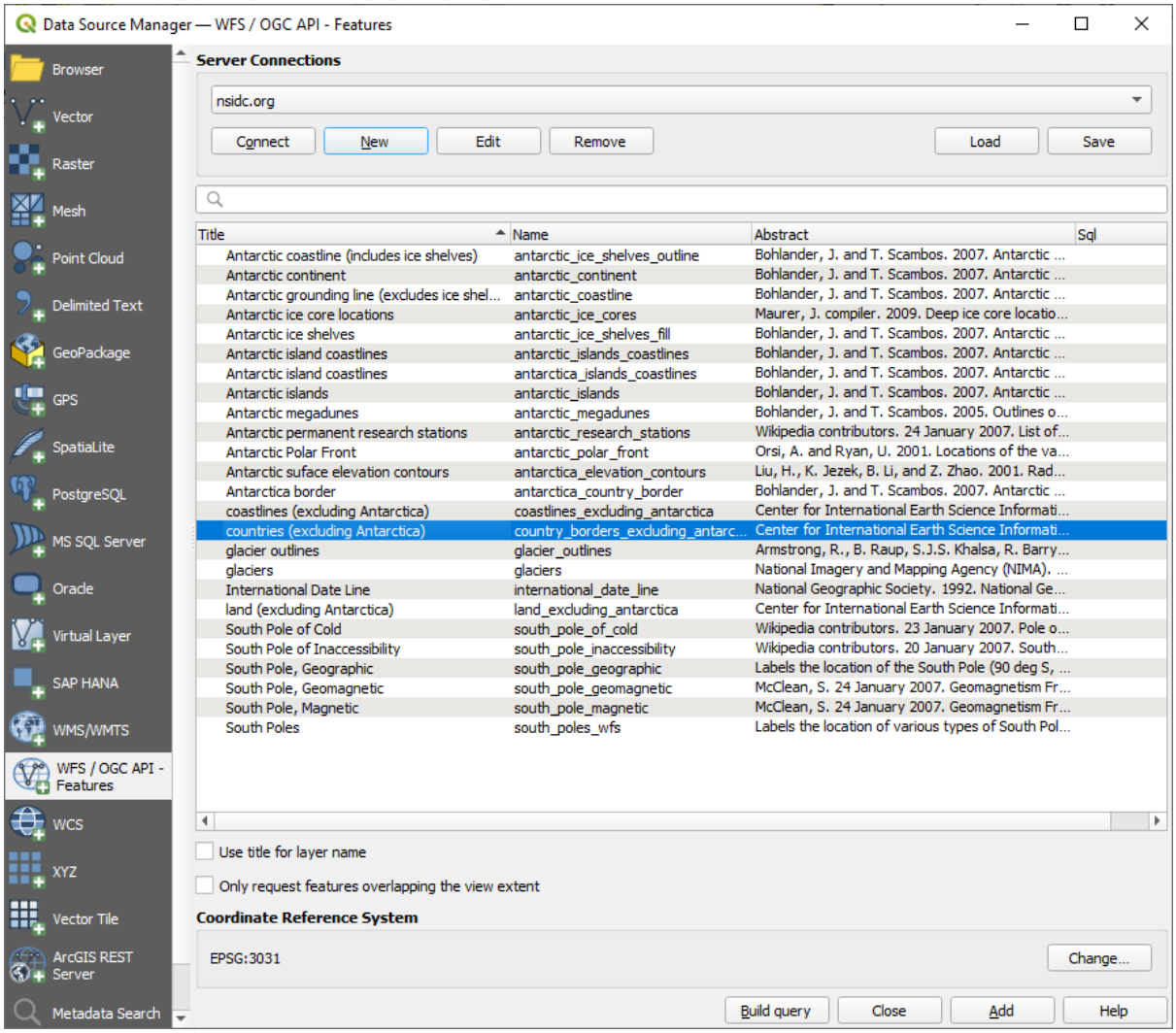
Although it is of possible to query a WFS layer after having loaded it, it's often more efficient to query it before you load it. That way, you're only requesting the features you want, meaning that you use far less bandwidth.

For example, on the WFS server we're currently using, there is a layer called *countries (excluding Antarctica)*. Let's say that we want to know where South Africa is relative to the *south_poles_wfs* layer (and perhaps also the *antarctica_country_border* layer) that's already been loaded.

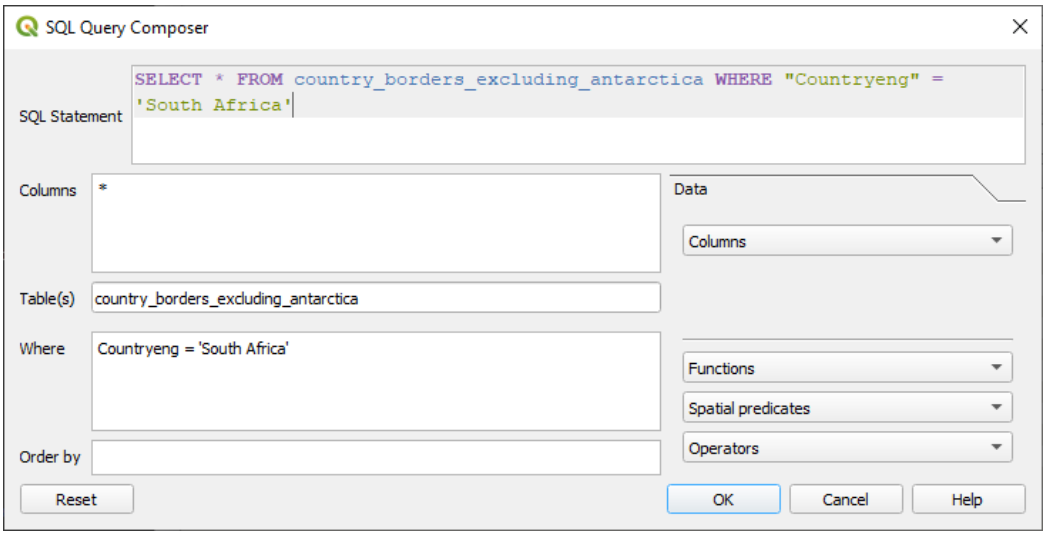
There are two ways to do this. You can load the whole *countries …* layer, and then build a query as usual once it's loaded. However, transmitting the data for all the countries in the world and then only using the data for South Africa seems a bit wasteful of bandwidth. Depending on your connection, this dataset can take several minutes to load.

The alternative is to build the query as a filter before even loading the layer from the server.

1. Enable the *WFS / OGC API Features* tab in the *Data Source Manager* dialog
2. Connect to the server we used before and you should see the list of available layers.
3. Find and double-click the *countries (excluding Antarctica)* layer. The layer name is `country_borders_excluding_antarctica`. You can also select the layer and press *Build query* button at the bottom of the dialog:

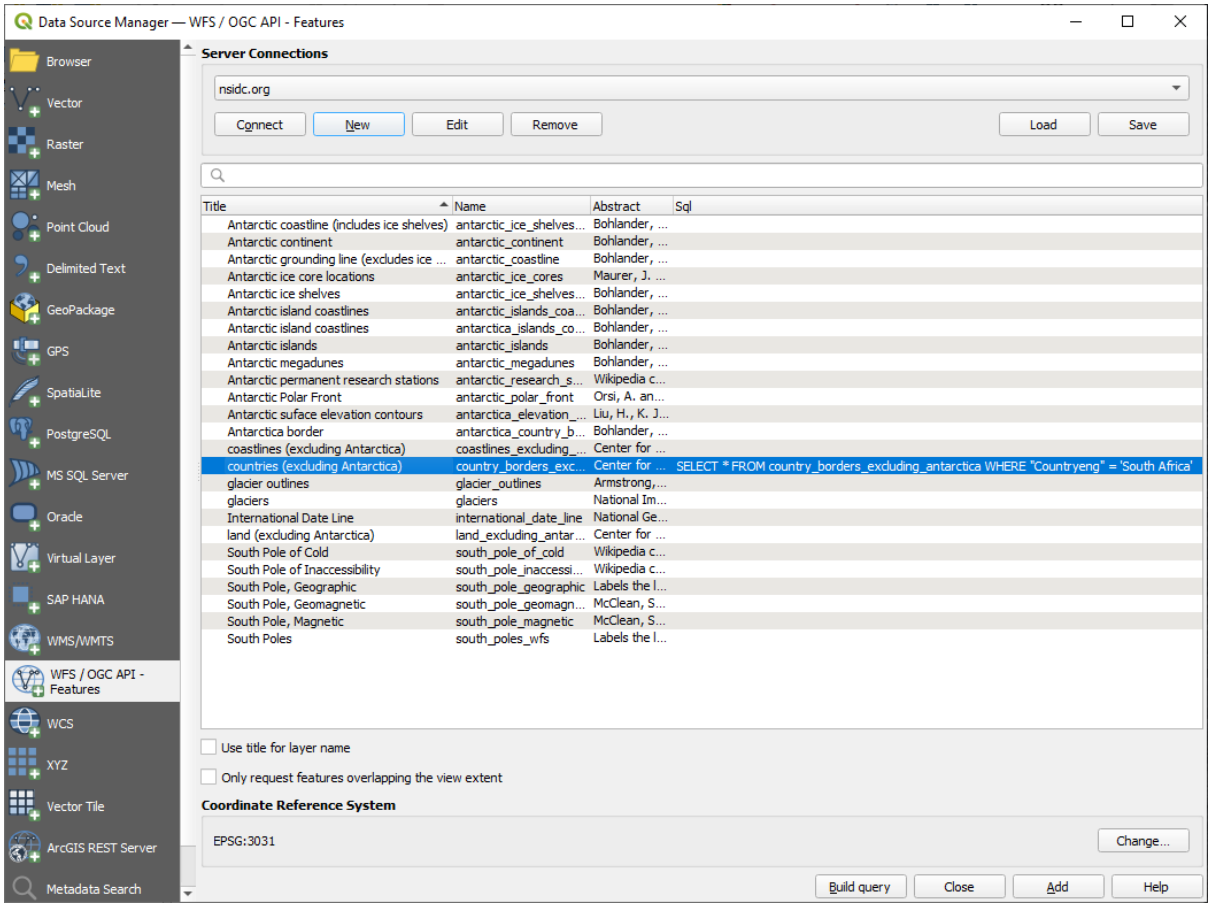


4. In the dialog that appears, type the following `SELECT * FROM country_borders_excluding_antarctica WHERE "Countryeng" = 'South Africa'` query in the *SQL Statement* box.

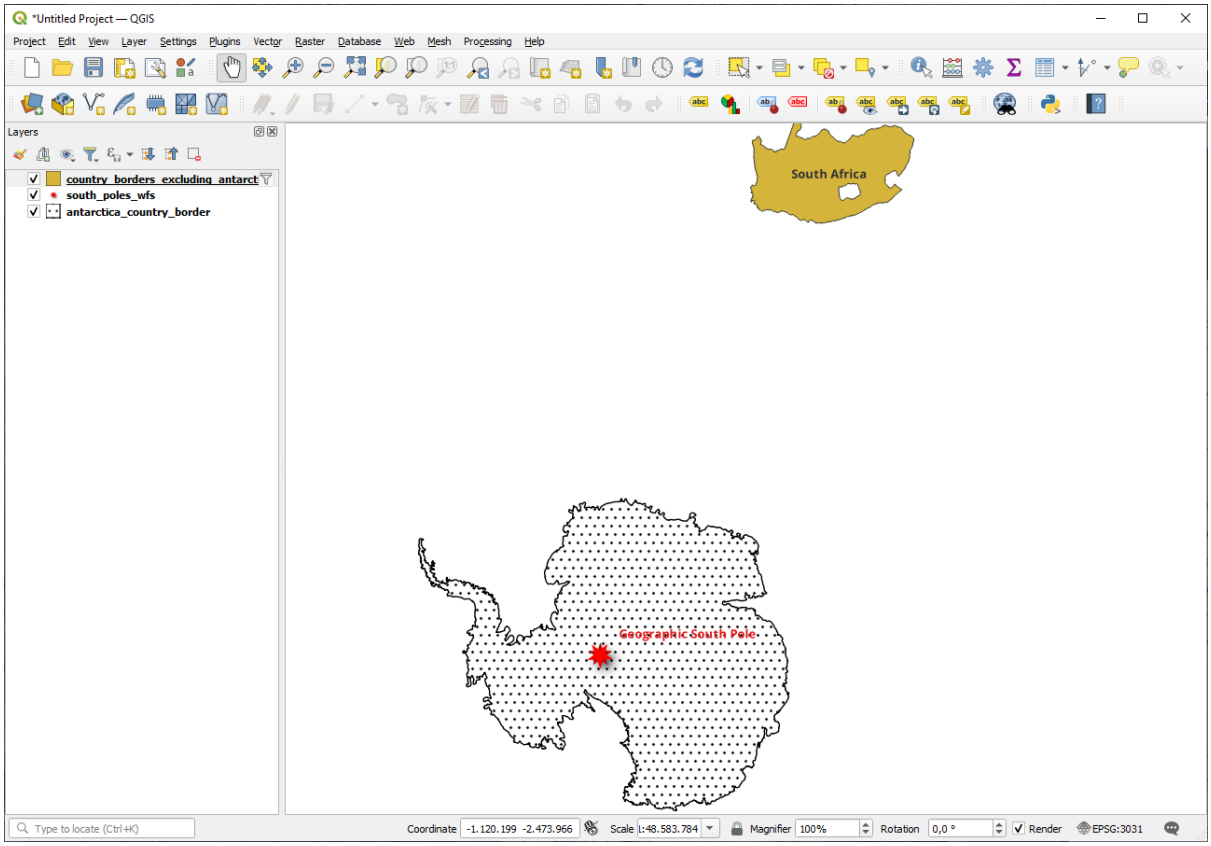


5. Press *OK*.

6. The expression used will appear as the *Sql* value of the target layer:



7. Click *Add* with the layer selected as above. Only the country with the `Countryeng` value of `South Africa` will load from that layer:



Did you notice the icon next to the `country_borders_excluding_antarctica` layer? It indicates that the loaded layer is filtered and does not display in the project all of its features.

8. You don't have to, but if you tried both methods, you'll notice that this is a lot faster than loading all the countries before filtering them!

Notes on WFS availability

It is rare to find a WFS hosting features you need, if your needs are very specific. The reason why Web Feature Services are relatively rare is because of the large amounts of data that must be transmitted to describe a whole feature. It is therefore not very cost-effective to host a WFS rather than a WMS, which sends only images.

The most common type of WFS you'll encounter will therefore probably be on a local network or even on your own computer, rather than on the Internet.

10.2.3 In Conclusion

WFS layers are preferable over WMS layers if you need direct access to the attributes and geometries of the layers. However, considering the amount of data that needs to be downloaded (which leads to speed problems and also a lack of easily available public WFS servers) it's not always possible to use a WFS instead of a WMS.

10.2.4 What's Next?

Next, you'll see how to use QGIS Server to provide OGC services.

CAPÍTULO 11

Module: QGIS Server

Este módulo é um contributo de Tudor Bărăscu.

Neste módulo é abordada a forma de instalar e usar o QGIS Servidor.

To learn more about QGIS Server, read the QGIS-Server-manual.

11.1 Lesson: Install QGIS Server

The goal for this lesson: To learn how to install **QGIS Server** on Debian Stretch. With negligible variations you can also follow it for any Debian based distribution like Ubuntu and its derivatives.

Nota: 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`.

11.1.1 ??? Follow Along: Install from packages

In this lesson we're going to do only the install from packages as shown here .

Install QGIS Server with:

```
apt install qgis-server --no-install-recommends --no-install-suggests

# if you want to install server plugins, also:
apt install python3-qgis
```

QGIS Server should be used in production without QGIS Desktop (with the accompanying X Server) installed on the same machine.

11.1.2 ??? Follow Along: QGIS Server Executable

The QGIS Server executable is `qgis_mapserv.fcgi`. You can check where it has been installed by running `find / -name 'qgis_mapserv.fcgi'` which should output something like `/usr/lib/cgi-bin/qgis_mapserv.fcgi`.

Optionally, if you want to do a command line test at this time you can run the `/usr/lib/cgi-bin/qgis_mapserv.fcgi --version` command which should output something like:

```
QGIS 3.21.0-Master 'Master' (1c70953f1e)
QGIS code revision 1c70953f1e
Qt version 5.15.2
Python version 3.9.5
GDAL/OGR version 3.2.2
PROJ version 7.2.1
EPSG Registry database version v10.008 (2020-12-16)
GEOS version 3.9.0-CAPI-1.16.2
SQLite version 3.34.1
OS Ubuntu 21.04
```

We'll see later on how to make WMS requests.

11.1.3 ??? HTTP Server Configuration

In order to access on the installed QGIS server from an Internet Browser we need to use an HTTP server. The Apache HTTP Server installation process is detailed in httpserver section.

Nota: If you installed QGIS Server without running an X Server (included in Linux Desktop) and if you also want to use the `GetPrint` command then you should install a fake X Server and tell QGIS Server to use it. You can do that by following the Xvfb installation process.

11.1.4 ??? Follow Along: Create another virtual host

Let's create another Apache virtual host pointing to QGIS Server. You can choose whatever name you like (`coco.bango`, `super.duper.training`, `example.com`, etc.) but for simplicity sake we're going to use `myhost`.

- Let's set up the `myhost` name to point to the localhost IP by adding `127.0.0.1 x` to the `/etc/hosts` with the following command: `sh -c "echo '127.0.0.1 myhost' >> /etc/hosts"` or by manually editing the file with `gedit /etc/hosts`.
- We can check that `myhost` points to the localhost by running in the terminal the `ping myhost` command which should output:

```
qgis@qgis:~$ ping myhost
PING myhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.029 ms
```

- Let's try if we can access QGIS Server from the `myhost` site by doing: `curl http://myhost/cgi-bin/qgis_mapserv.fcgi` or by accessing the url from your Debian box browser. You will probably get:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /cgi-bin/qgis_mapserv.fcgi was not found on this server.</p>
```

(continues on next page)

(continuação da página anterior)

```
<hr>
<address>Apache/2.4.25 (Debian) Server at myhost Port 80</address>
</body></html>
```

- Apache doesn't know that he's supposed to answer requests pointing to the server named `myhost`. In order to setup the virtual host the simplest way would be to make a `myhost.conf` file in the `/etc/apache2/sites-available` directory that has the same content as `qgis.demo.conf` except for the `ServerName` line that should be `ServerName myhost`. You could also change where the logs go as otherwise the logs for the two virtual hosts would be shared but this is optional.
- Let's now enable the virtual host with `a2ensite myhost.conf` and then reload the Apache service with `service apache2 reload`.
- If you try again to access the http://myhost/cgi-bin/qgis_mapserv.fcgi url you'll notice everything is working now!

11.1.5 In Conclusion

You learned how to install different QGIS Server versions from packages, how to configure Apache with QGIS Server, on Debian based Linux distros.

11.1.6 What's Next?

Now that you've installed QGIS Server and it's accessible through the HTTP protocol, we need to learn how to access some of the services it can offer. The topic of the next lesson is to learn how to access QGIS Server WMS services.

11.2 Lesson: Serving WMS

The data used for this exercise are available in the `qgis-server-tutorial-data` subdirectory of the *training data* you downloaded. For convenience and to avoid possible permissions problems, we will assume that these files are stored in `/home/qgis/projects` directory. Hence, adapt the following instructions to your path.

The demo data contains a QGIS project named `world.qgs` that is already prepared to be served with QGIS Server. If you want to use your own project or you want to learn how a project is prepared see the Creatingwmsfromproject section.

Nota: This module presents the URLs so that the audience can easily distinguish the parameters and parameters values. While the normal format is:

```
...&field1=value1&field2=value2&field3=value3
```

this tutorial uses:

```
&field1=value1
&field2=value2
&field3=value3
```

Pasting them into Mozilla Firefox works properly but other web browsers like Chrome may add unwanted spaces between the `field:parameter` pairs. So, if you encounter this issue you can either use Firefox or modify the URLs so that they're in one line format.

Let's make a WMS GetCapabilities request in the web browser or with curl:

```
http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi
?SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetCapabilities
&map=/home/qgis/projects/world.qgs
```

In the Apache config from the previous lesson the `QGIS_PROJECT_FILE` variable sets the default project to `/home/qgis/projects/world.qgs`. However, in the above request we made use of the **map** parameter to be explicit and to show it can be used to point at any project. If you delete the **map** parameter from the above request QGIS Server will output the same response.

By pointing any WMS client to the `GetCapabilities` URL, the client gets in response an XML document with metadata of the Web Map Server's information, e.g. what layers does it serve, the geographical coverage, in what format, what version of WMS etc.

As QGIS is also a ogc-wms you can create a new WMS server connection with the help of the above GetCapabilities url. See the *Lesson: Web Mapping Services* or the ogc-wms-servers section on how to do it.

By adding the `countries` WMS layer to your QGIS project you should get an image like the one below:

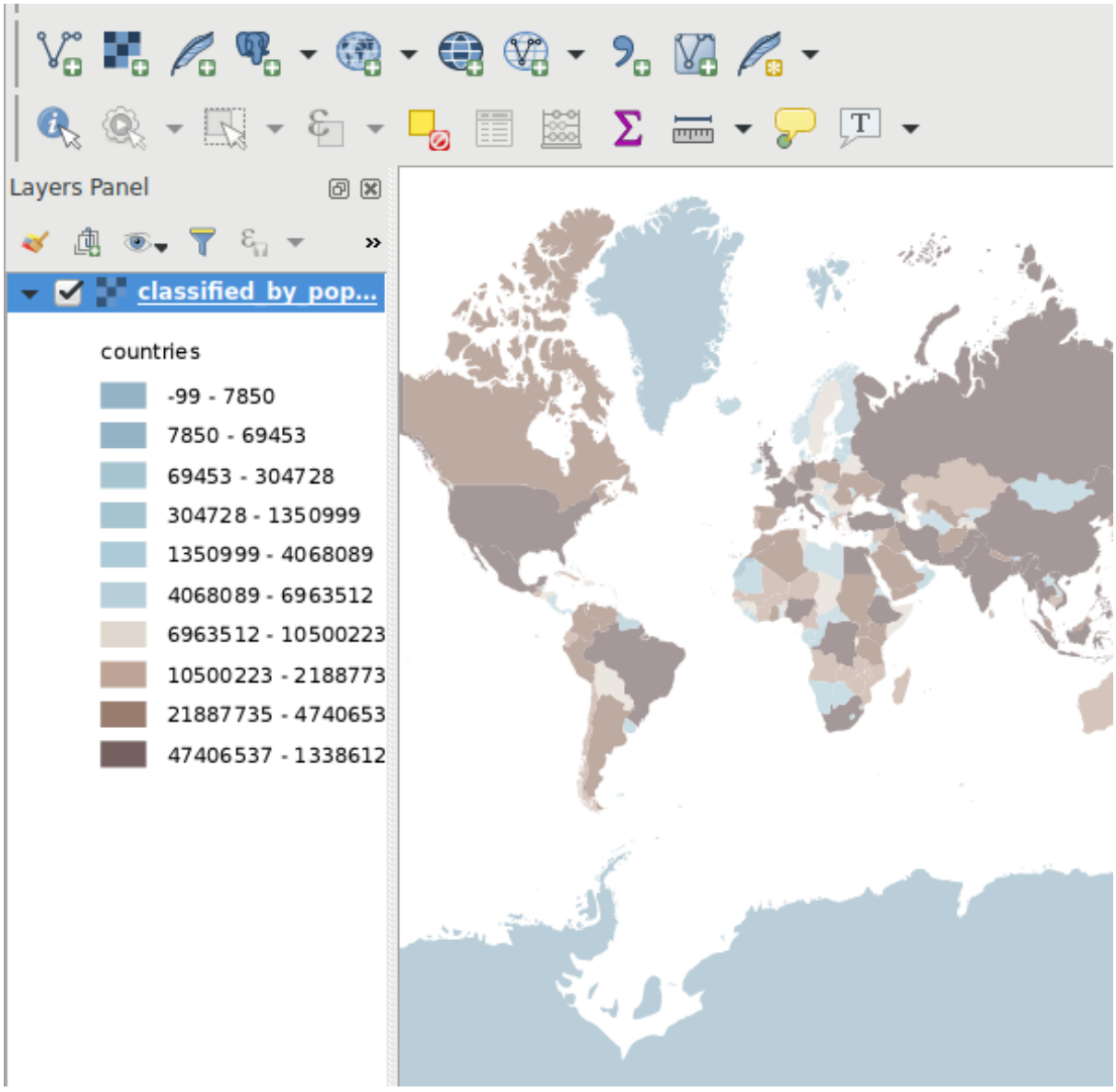


Figura11.1: QGIS Desktop consuming the QGIS Server countries layer WMS service

Nota: QGIS Server serves layers that are defined in the `world.qgs` project. By opening the project with QGIS you can see there are multiple styles for the countries layer. QGIS Server is also aware of this and you can choose the style you want in your request. The `classified_by_population` style was chosen in the above image.

11.2.1 Registando

When you're setting up a server, the logs are always important as they show you what's going on. We have setup in the `*.conf` file the following logs:

- QGIS Server log at `/logs/qgisserver.log`
- `qgisplatform.demo` Apache access log at `qgisplatform.demo.access.log`
- `qgisplatform.demo` Apache error log at `qgisplatform.demo.error.log`

The log files are simply text files so you can use a text editor to check them out. You can also use the `tail` command in a terminal: `sudo tail -f /logs/qgisserver.log`.

This will continuously output in the terminal what's written in that log file. You can also have three terminals opened for each of the log files like so:

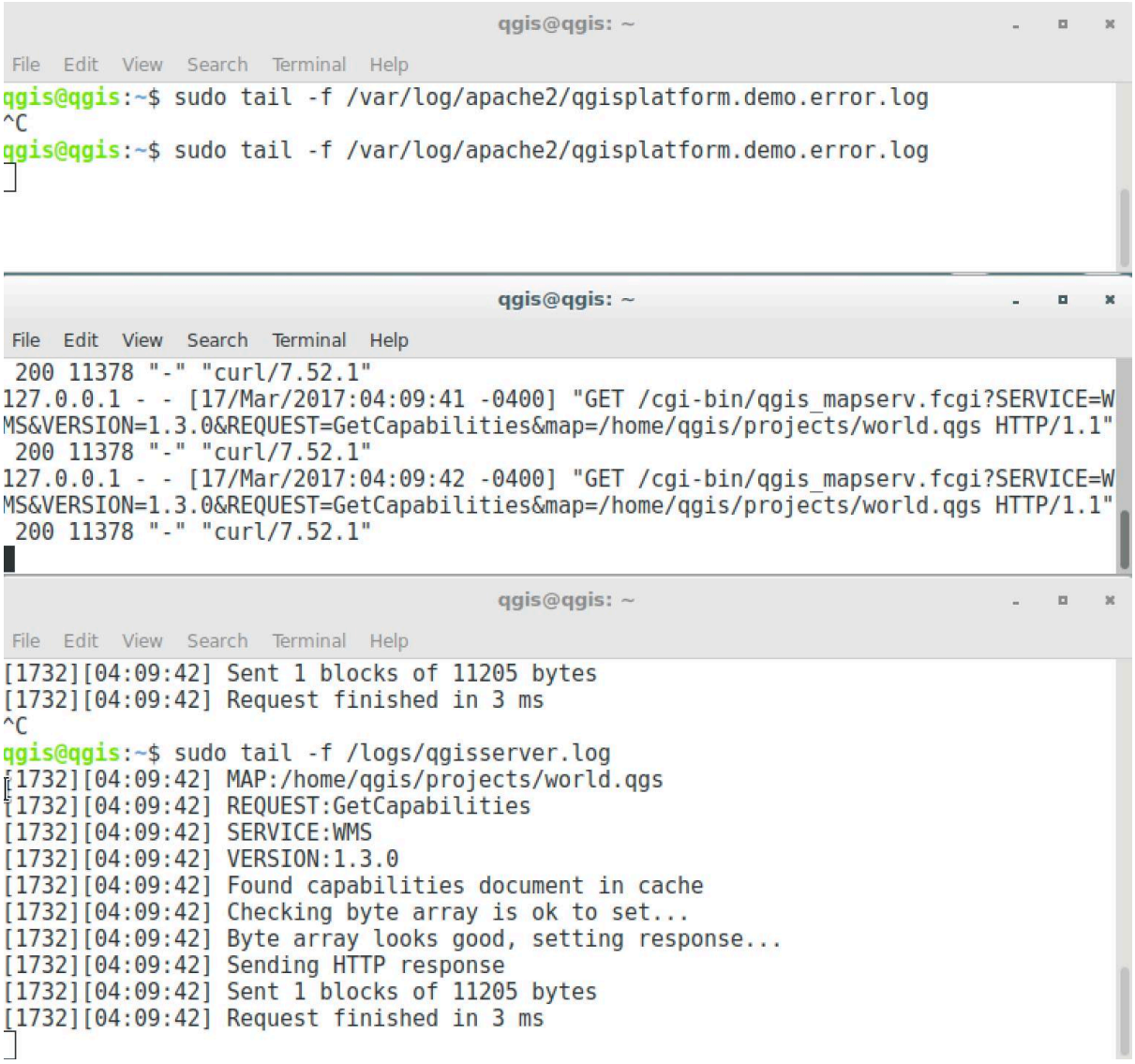


Figura11.2: Using the `tail` command to visualise QGIS Server logs output

When you use QGIS Desktop to consume the QGIS Server WMS services you will see all the requests QGIS sends to the Server in the access log, the errors of QGIS Server in the QGIS Server log etc.

Nota:

- If you look at the logs in the following sections you should get a better understanding on what's happening.
- By restarting Apache while looking in the QGIS Server log you can find some extra pointers on how things work.

11.2.2 GetMap requests

In order to display the `countries` layer, QGIS Desktop, like any other WMS client, is using `GetMap` requests.

A simple request looks like:

```
http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi
?MAP=/home/qgis/projects/world.qgs
&SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&BBOX=-432786,4372992,3358959,7513746
&SRS=EPSG:3857
&WIDTH=665
&HEIGHT=551
&LAYERS=countries
&FORMAT=image/jpeg
```

The above request should output the following image:

Figure: simple GetMap request to QGIS Server

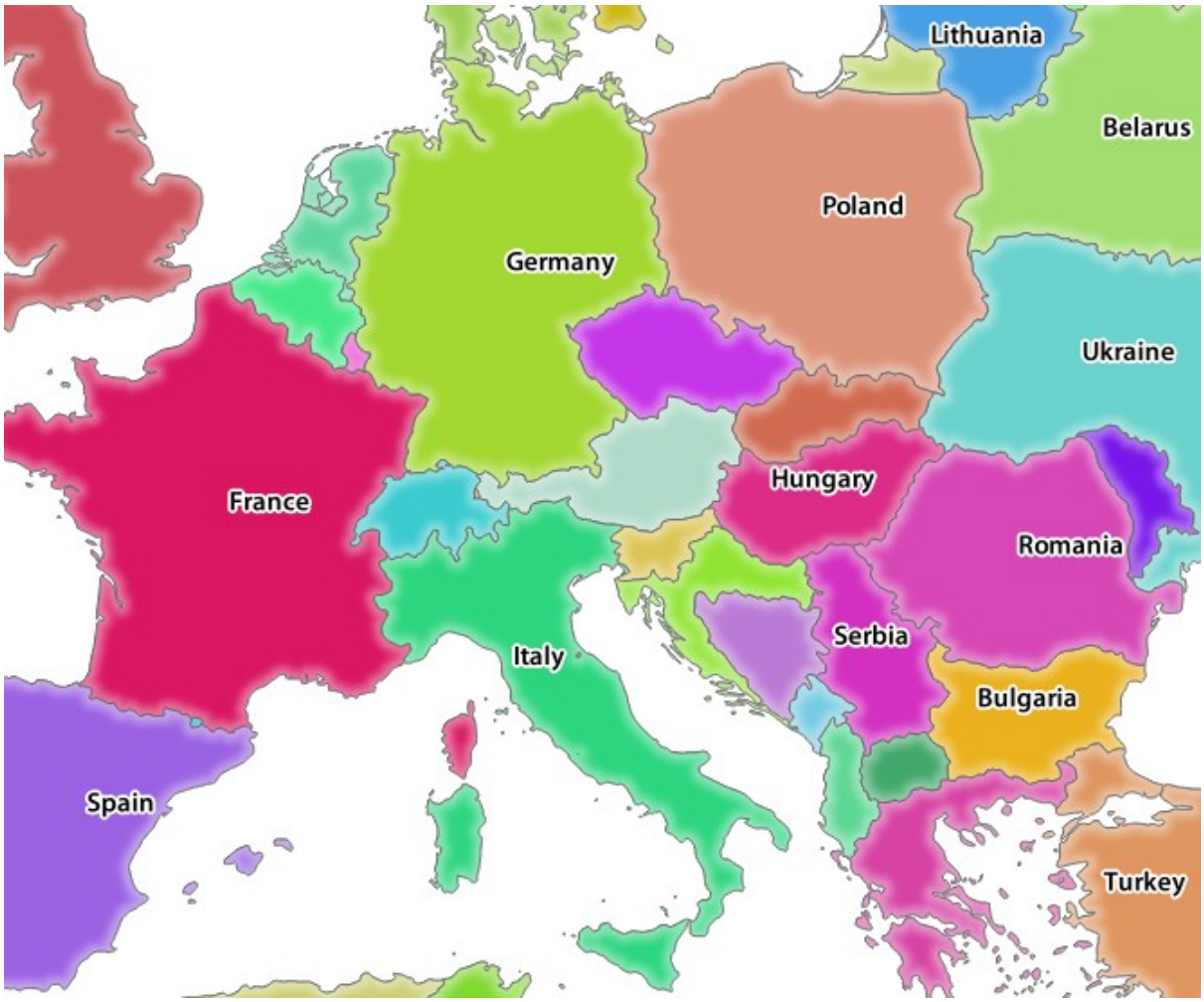


Figura11.3: Qgis Server response after a simple GetMap request

11.2.3 ??? Try Yourself: Change the Image and Layers parameters

Based on the request above, let's replace the `countries` layer with another.

In order to see what other layers are available you could open up the `world.qgs` project in QGIS and look at its contents. Keep in mind though that the WMS clients don't have access to the QGIS project, they just look at the capabilities document contents.

Also, there's a configuration option so that some of the layers existing in the QGIS project are ignored by QGIS when serving the WMS service.

So, you could look at the layer list when you point QGIS Desktop to the `GetCapabilities` URL or you could try yourself finding other layer names in the `GetCapabilities` XML response.

One of the layer names that you could find and works is `countries_shapeburst`. You may find others but keep in mind some may not be visible at such a small scale so you could get a blank image as response.

You can also play around with others parameters from above, like changing the returned image type to `image/png`.

11.2.4 ??? Follow Along: Use Filter, Opacities and Styles parameters

Let's do another request that adds another layer, some basic parameters, **FILTER** and **OPACITIES**, but also uses the standard STYLES parameter.

```
http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi
?MAP=/home/qgis/projects/world.qgs
&SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&BBOX=-432786,4372992,3358959,7513746
&SRS=EPSG:3857
&WIDTH=665
&HEIGHT=551
&FORMAT=image/jpeg
&LAYERS=countries,countries_shapeburst
&STYLES=classified_by_name,blue
&OPACITIES=255,30
&FILTER=countries:"name" IN ( 'Germany' , 'Italy' )
```

The above request should output the following image:

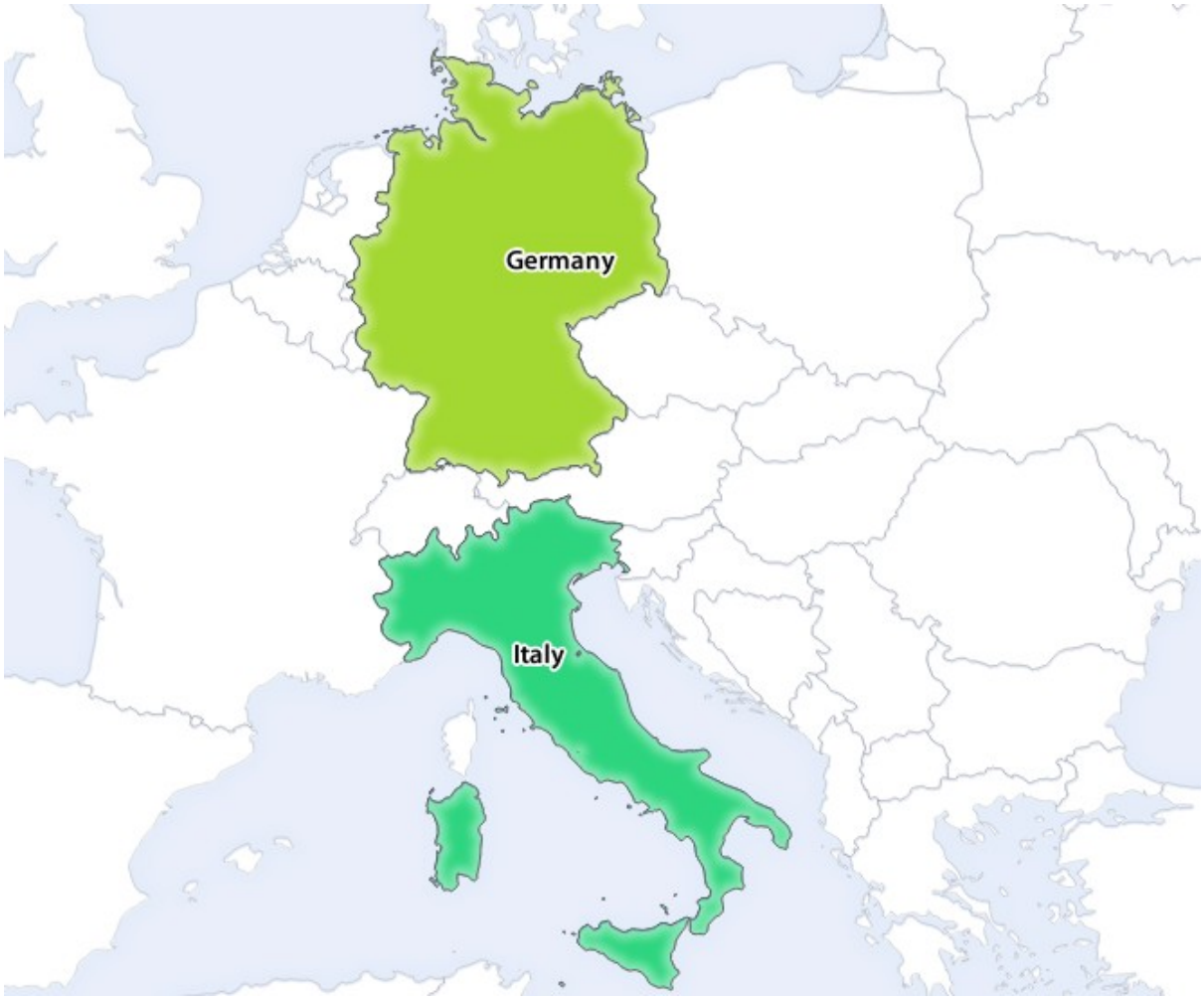


Figura11.4: Response to a GetMap request with FILTER and OPACITIES parameters

As you can see from the above image, among other things, we told QGIS Server to render only **Germany** and **Italy** from the countries layer.

11.2.5 ??? Follow Along: Use Redlining

Let's do another GetMap request that makes use of the redlining feature and of the **SELECTION** parameter detailed in the Basics section:

```
http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi
?MAP=/home/qgis/projects/world.qgs
&SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&BBOX=-432786,4372992,3358959,7513746
&SRS=EPSG:3857
&WIDTH=665
&HEIGHT=551
&LAYERS=countries,countries_shapeburst
&FORMAT=image/jpeg
&HIGHLIGHT_GEOM=POLYGON((590000 6900000, 590000 7363000, 2500000 7363000, 2500000␣
↪6900000, 590000 6900000))
&HIGHLIGHT_SYMBOL=<StyledLayerDescriptor><UserStyle><Name>Highlight</Name>
↪<FeatureTypeStyle><Rule><Name>Symbol</Name><LineSymbolizer><Stroke><SvgParameter␣
```

(continues on next page)

(continuação da página anterior)

```
↪name="stroke">%233a093a</SvgParameter><SvgParameter name="stroke-opacity">1</
↪SvgParameter><SvgParameter name="stroke-width">1.6</SvgParameter></Stroke></
↪LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></StyledLayerDescriptor>
&HIGHLIGHT_LABELSTRING=QGIS Tutorial
&HIGHLIGHT_LABELSIZE=30
&HIGHLIGHT_LABELCOLOR=%23000000
&HIGHLIGHT_LABELBUFFERCOLOR=%23FFFFFF
&HIGHLIGHT_LABELBUFFERSIZE=3
&SELECTION=countries:171,65
```

Pasting the above request in your web browser should output the following image:

Figura11.5: Response to a request with the REDLINING feature and SELECTION parameter

You can see from the above image that the countries with the 171 and 65 ids were highlighted in yellow (Romania and France) by using the **SELECTION** parameter and we used the **REDLINING** feature to overlay a rectangle with the **QGIS Tutorial** label.

11.2.6 GetPrint requests

One very nice feature of QGIS Server is that it makes use of the QGIS Desktop print layouts. You can learn about it in the wms_getprint section.

If you open the `world.qgs` project with QGIS Desktop you will find a print layout named `Population distribution`. A simplified `GetPrint` request that exemplifies this amazing feature is:

```
http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi
?map=/home/qgis/projects/world.qgs
&SERVICE=WMS
&VERSION=1.3.0&
REQUEST=GetPrint
&FORMAT=pdf
&TRANSPARENT=true
&SRS=EPSG:3857
&DPI=300
&TEMPLATE=Population distribution
&map0:extent=-432786,4372992,3358959,7513746
&LAYERS=countries
```

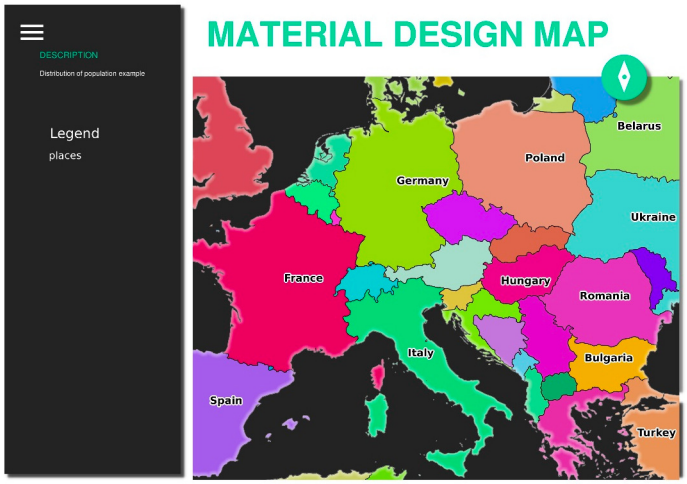


Figura11.6: Shows the pdf resulted from the above GetPrint request

Naturally, it's hard to write your `GetMap`, `GetPrint` etc. requests.

QGIS Web Client or QWC is a Web client project that can work alongside QGIS Server so that you can publish your projects on the Web or help you create QGIS Server requests for a better understanding about the possibilities.

You can install it like this:

- As user `qgis` go to the home directory with `cd /home/qgis`.
- Download the QWC project from here and unzip it.
- Make a symbolic link to the `/var/www/html` directory as it's the `DocumentRoot` that we've setup in the virtual host configuration. If you unzipped the archive under `/home/qgis/Downloads/QGIS-Web-Client-master` we can do that with `sudo ln -s /home/qgis/Downloads/QGIS-Web-Client-master /var/www/html/`.
- Access http://qgisplatform.demo/QGIS-Web-Client-master/site/qgiswebclient.html?map=/home/qgis/projects/world.qgs from your Web browser.

Now you should be able to see the Map as in the following figure:

Figura11.7: QGIS Web Client consuming the world.qgs project

If you click the Print button in QWC you can interactively create `GetPrint` requests. You can also click the `?` icon in the QWC to access the available help so that you can better discover the QWC possibilities.

11.2.7 In Conclusion

You learned how use QGIS Server to provide WMS Services.

11.2.8 What's Next?

Next, you'll see how to use QGIS as a frontend for the famous GRASS GIS.

CAPÍTULO 12

Module: GRASS

GRASS (Sistema de Suporte para Análise de Recursos Geográficos) é um SIG conhecido de código aberto com uma grande variedade de funções úteis para SIG. Foi lançado em 1984, e teve muitas melhorias e funcionalidades adicionais desde então. O QGIS permite que faça uso directamente das poderosas ferramentas do SIG GRASS.

12.1 Lesson: GRASS Setup

Using GRASS in QGIS requires you to think of the interface in a slightly different way. Remember that you're not working in QGIS directly, but working in GRASS *via* QGIS. Hence, make sure you have installed QGIS Desktop with Grass support.

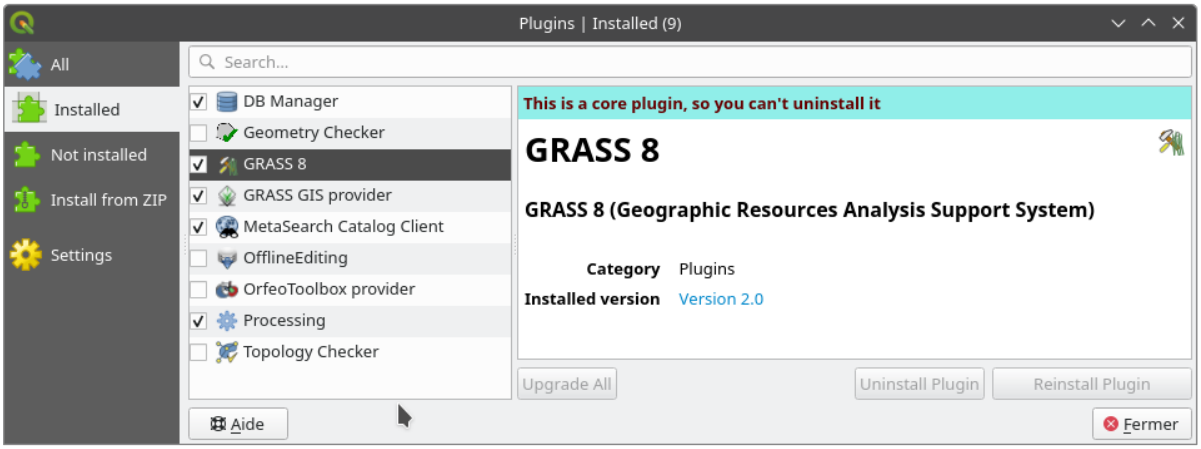
To open a QGIS session with GRASS available on Windows you have to click on the `QGIS Desktop with GRASS` icon.

The goal for this lesson: To begin a GRASS project in QGIS.

12.1.1 ??? Follow Along: Start a New GRASS Session

To launch GRASS from within QGIS, you need to activate it as with any other plugin:

1. First, open a new QGIS project.
2. In the *Plugin Manager*, enable *GRASS 8* in the list:



The GRASS toolbar and the GRASS panel will appear:

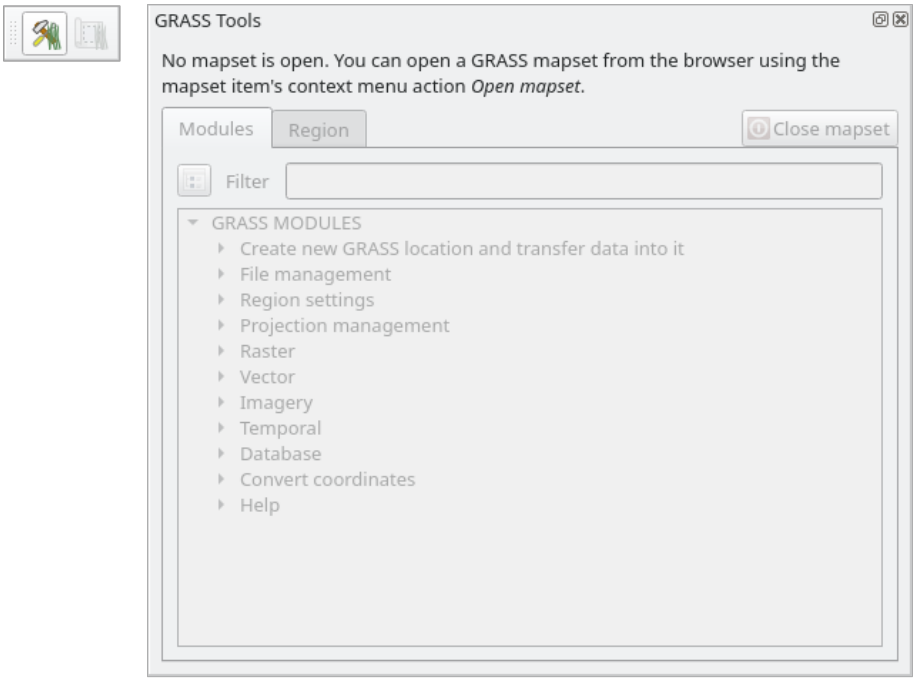


Figura12.1: GRASS Toolbar and Panel

The GRASS panel is not active because, before you can use GRASS, you need to create a `Mapset`. GRASS always works in a database environment, which means that you need to import all the data you want to use into a GRASS database.

The GRASS database has a straightforward structure, even if at a first look it seems very complicated. The most important thing you should know is that the upper level of the database is the `Location`. Each `Location` can contain different `Mapset`: in **every** `Mapset` you will find the `PERMANENT` Mapset because it is created by default by GRASS. Each `Mapset` contains the data (raster, vector, etc) in a particular structure, but don't worry, GRASS will take care of this for you.

Just remember: `Location` contains `Mapset` that contains the data. For more information visit the GRASS website.

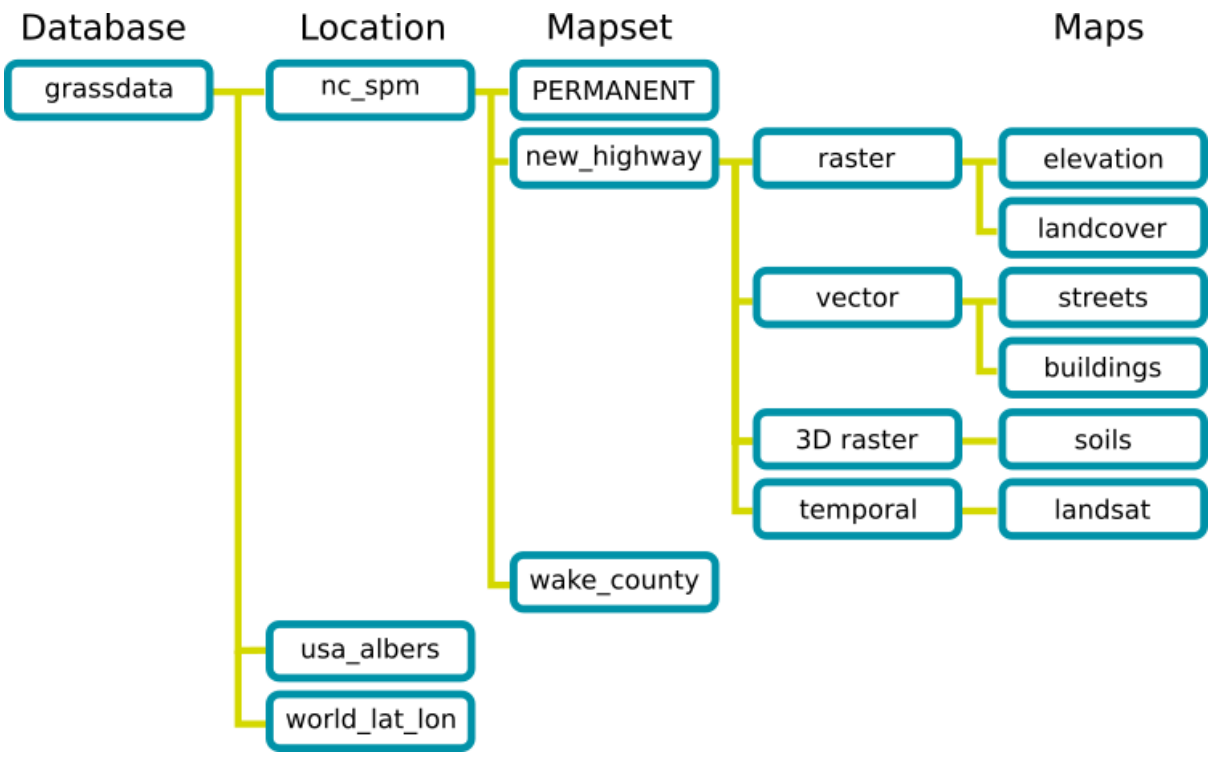
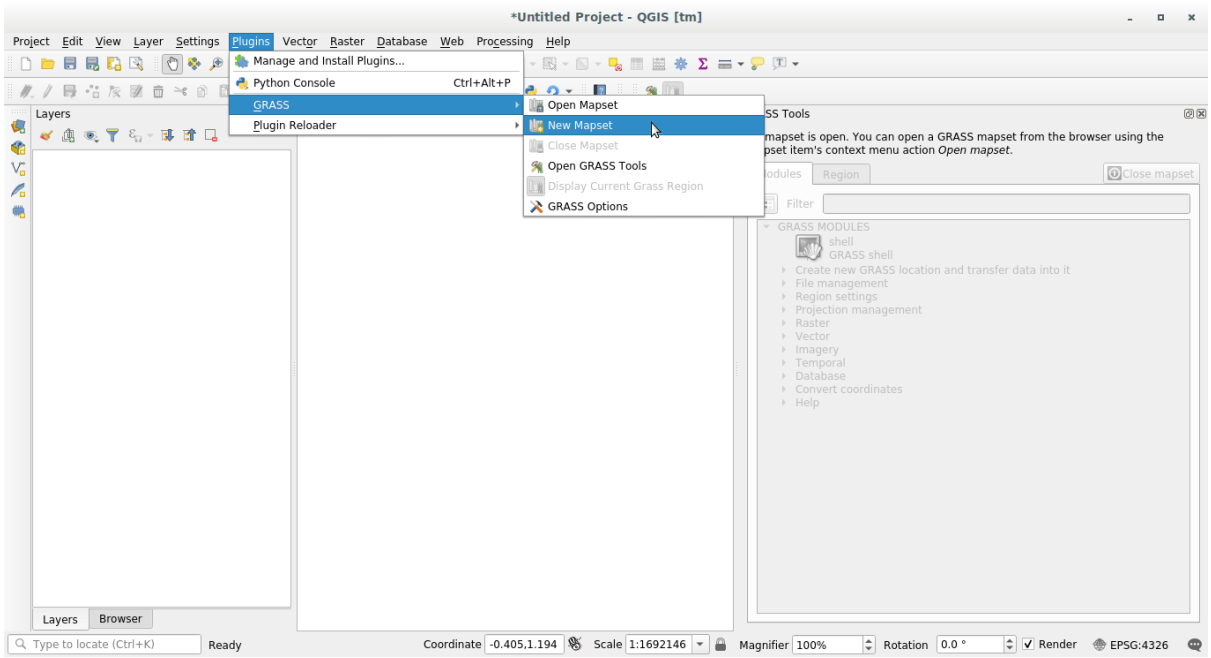


Figura12.2: GRASS database structure (from GRASS docs)

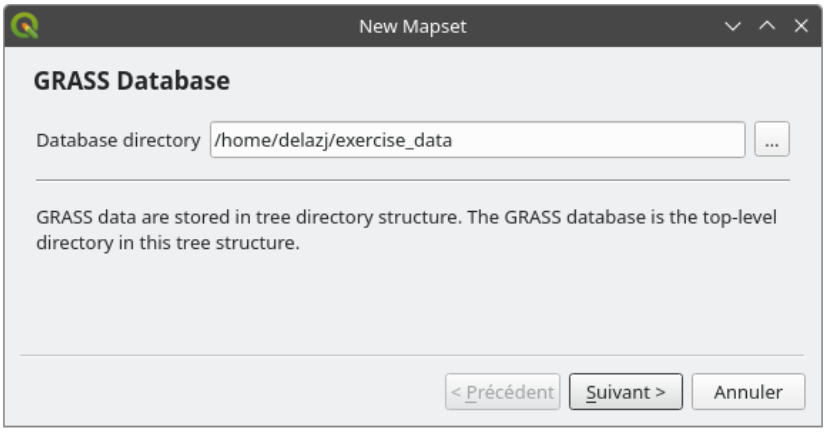
12.1.2 ??? Follow Along: Start a New GRASS Project

1. Click on the *Plugins* ► *GRASS* ► *New Mapset* menu:



   You'll be asked to choose the location of the GRASS database.

2. Set it as the directory that will be used by GRASS to set up its database:



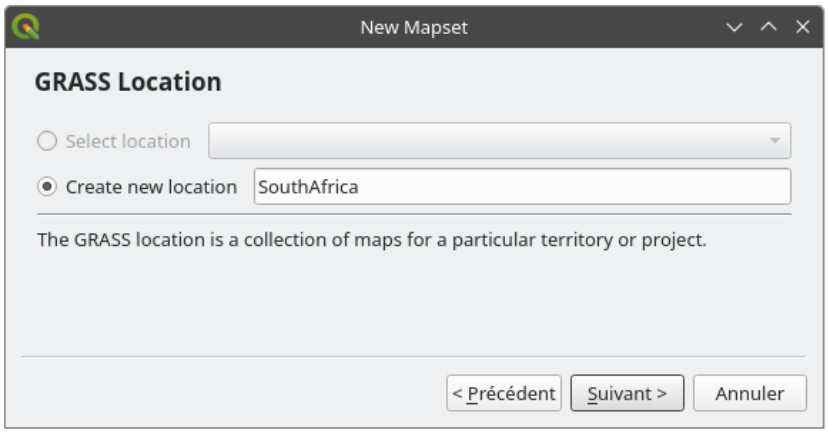
3. Click *Next*.

GRASS needs to create a `Location`, which describes the maximum extent of the geographic area you'll be working in, also known as Grass `Region`.

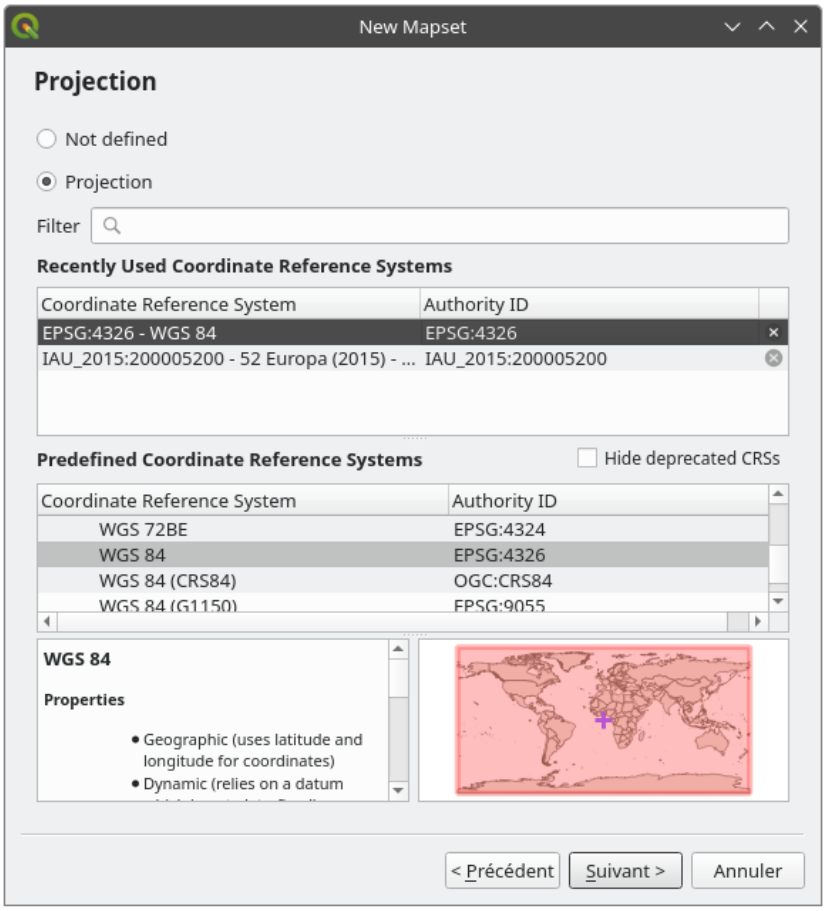
Nota: the Region is extremely important for GRASS because it describes the area in which all layers will be taken into account for GRASS. Everything that is outside will not be considered. Don't worry, you can always change the extent of the GRASS Region after the Location has been created

---

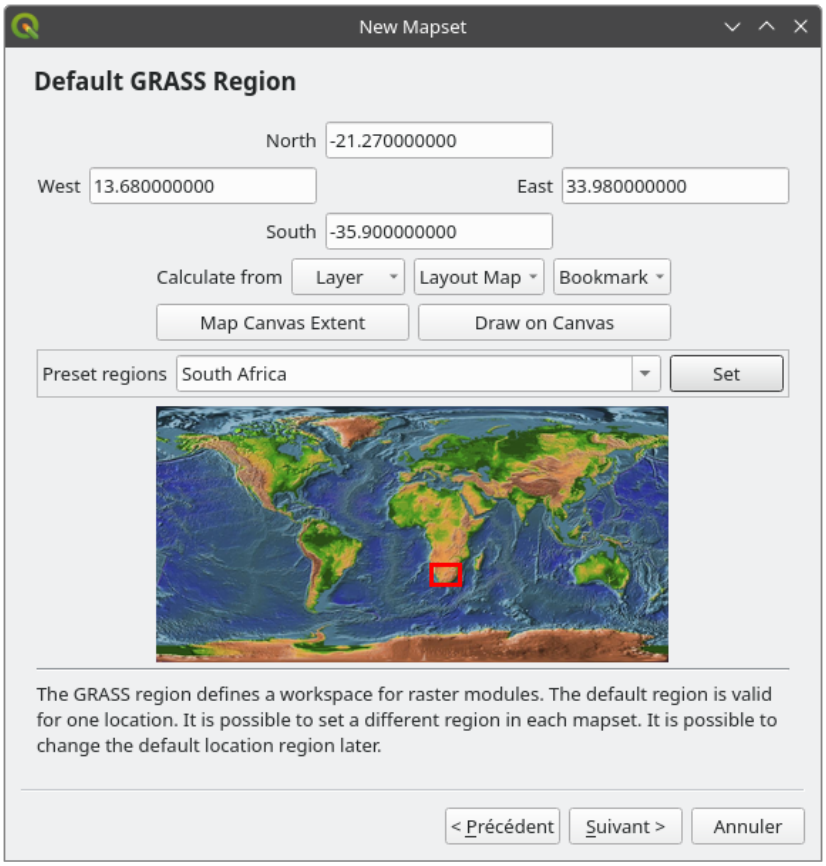
1. Call the new location `SouthAfrica`:



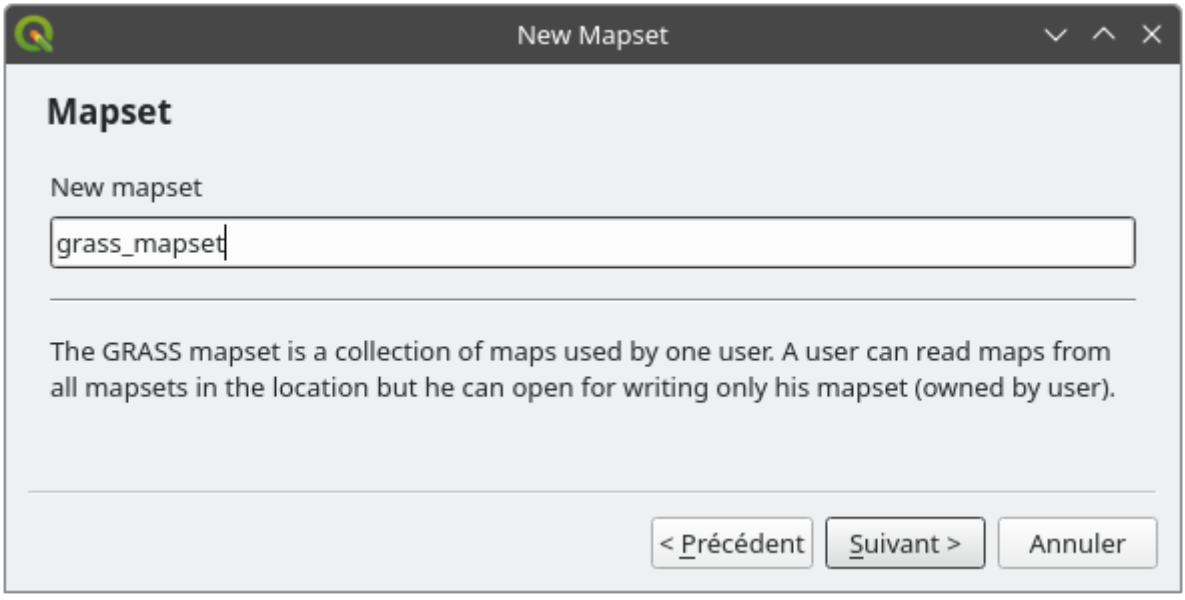
2. Click *Next*.
3. We'll be working with `WGS 84`, so search for and select this CRS:



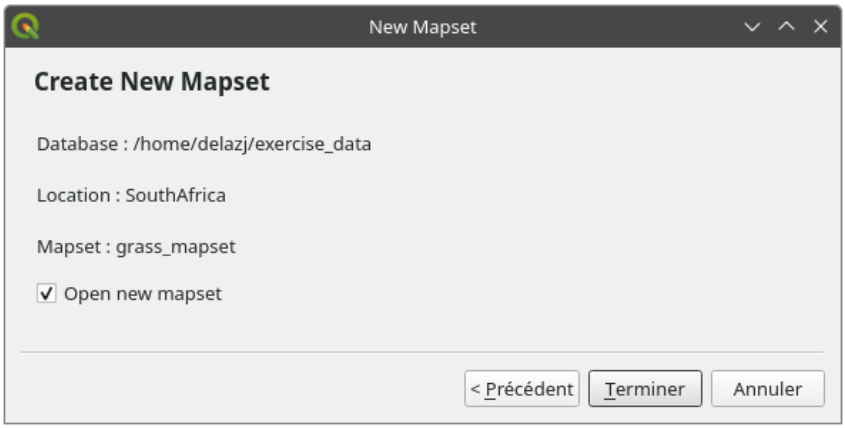
4. Click *Next*.
5. Now select the region *South Africa* from the dropdown and click *Set*:



6. Click *Next*.
7. Create a mapset, which is the map file that you'll be working with.



Once you're done, you'll see a dialog asking with a summary of all the information entered.



8. Click *Finish*.
9. Click *OK* on the success dialog.

You will see that the GRASS Panel will become active and you can start to use all GRASS tools.

12.1.3 ??? Follow Along: Loading Vector Data into GRASS

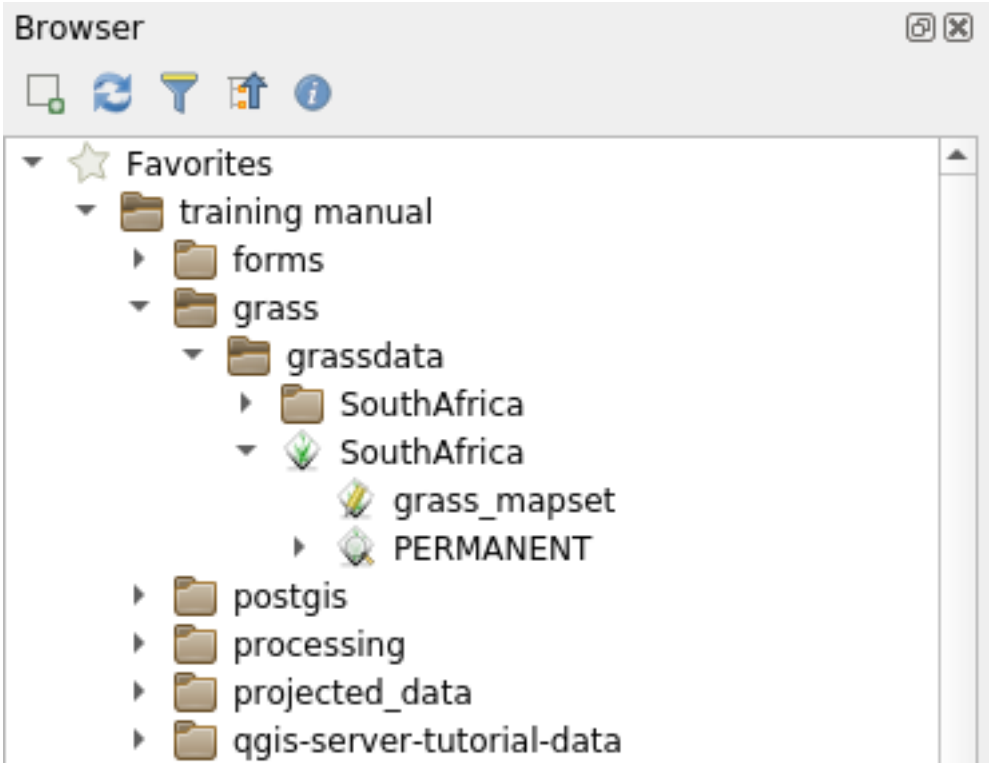
You have now a blank map and before you can start to use all the GRASS tools you have to load data into the GRASS database, specifically into the `Mapset`. You cannot use GRASS tools with layer that are not loaded into a GRASS `Mapset`.

There are many different ways to load data in the GRASS database. Let's start with the first one.

??? Follow Along: Load data using the QGIS Browser

In section *??? The Browser Panel* we saw that the easiest and quickest way to load the data in QGIS is the Browser Panel.

GRASS data are recognized from the QGIS Browser as *real* GRASS data and you can notice it because you will see the GRASS icon next to the GRASS Mapset. Moreover you will see the icon next to the Mapset that is opened.

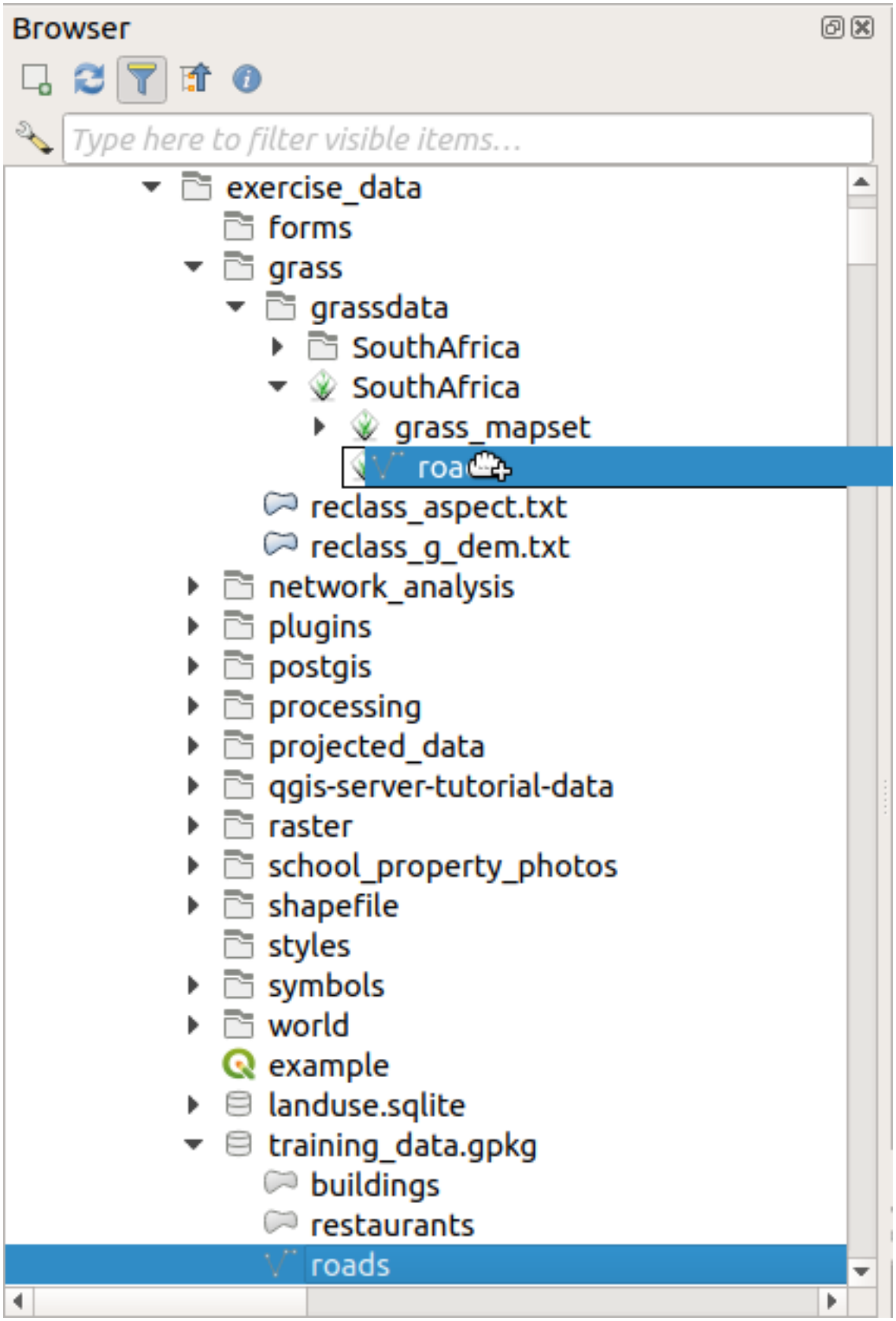


Nota: You will see a replication of the GRASS Location as normal folder: GRASS `Mapset` data are those within the folder

You can easily **drag and drop** layers from a folder to the GRASS Mapset.

Let's try to import the `roads` layer into the `grass_mapset` Mapset of the `SouthAfrica` Location.

Go to the Browser, and simply drag the `roads` layer from the `training_data.gpkg` GeoPackage file into the `grass_mapset` Mapset.



That's it! If you expand the Mapset you will see the imported `roads` layer. You can now load in QGIS the imported layer like all the other layers.

Dica: You can also load layers from the Layer Legend Panel to Mapset in the Browser Panel. This will speed up

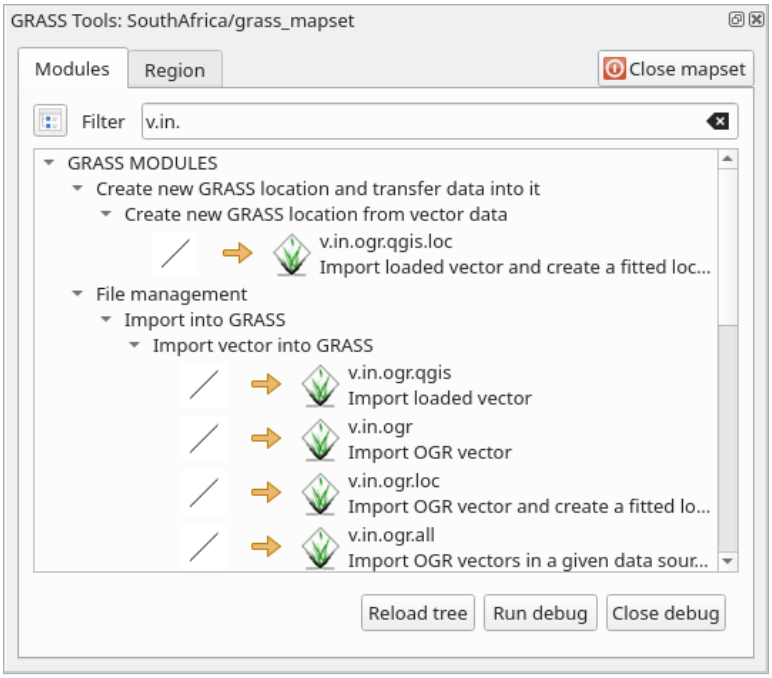
incredibly your workflow!

Follow Along: Load data using the GRASS Panel

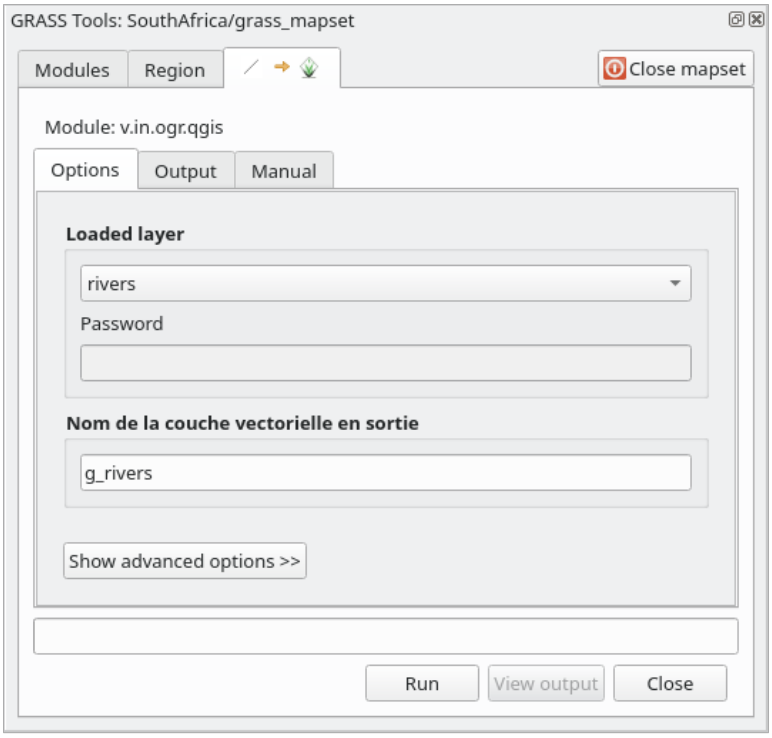
We will use now the *long* method to load the `rivers.shp` layer into the same Mapset.

1. Load data into QGIS as usual. Use the `rivers.shp` dataset (found in the `exercise_data/shapefile/` folder)
2. As soon as it is loaded, click on the *Filter* box of the `GRASS Panel` and find the vector import tool by entering the term `v.in.ogr.qgis` (available under *File management* ► *Import into GRASS* ► *Import vector into GRASS*):

   Aviso: There are 2 similar tools: `v.in.ogr.qgis` and `v.in.ogr.qgis.loc`. We are looking for the **first** one.

   The `v` stands for *vector*, `in` means a function to import data into the GRASS database, `ogr` is the software library used to read vector data, and `qgis` means that the tool will look for a vector from among the vectors already loaded into QGIS.
3. Once you've found this tool, click on it to bring up the tool itself. Choose the *rivers* layer in the *Loaded Layer* box and type and name it `g_rivers` to prevent confusion:



Nota: ★★★ Note the extra import options provided under *Advanced Options*. These include the ability to add a WHERE clause for the SQL query used for importing the data.

4. Click *Run* to begin the import.
5. When it's done, click the *View output* button to see the newly imported GRASS layer in the map.
6. Close first the import tool (click the *Close* button to the immediate right of *View output*), then close the *GRASS Tools* window.
7. Remove the original *rivers* layer.

Now you are left with only the imported GRASS layer as displayed in your QGIS map.

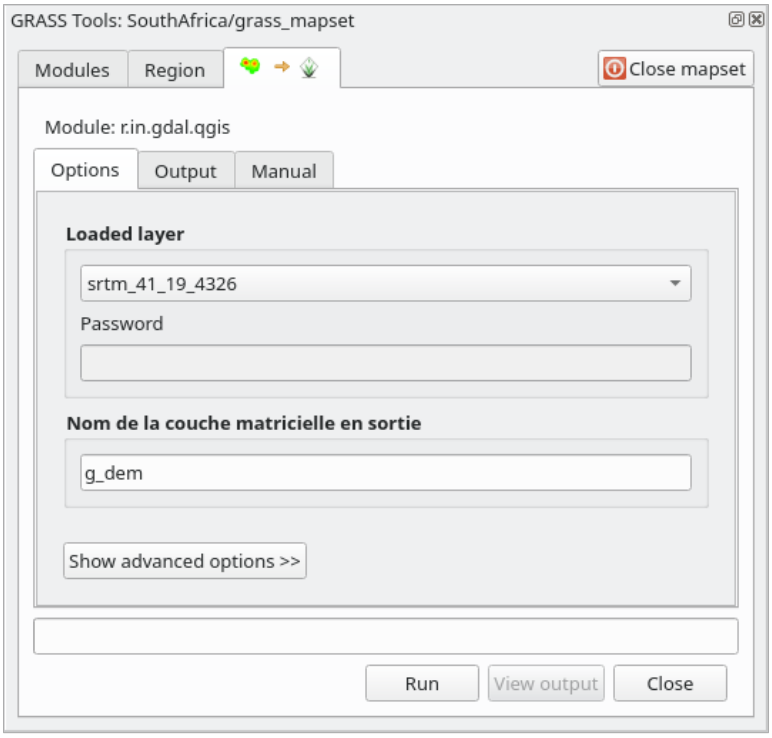
12.1.4 ??? Follow Along: Loading Raster Data into GRASS

You can import a raster layer in the same ways we imported vector layers.

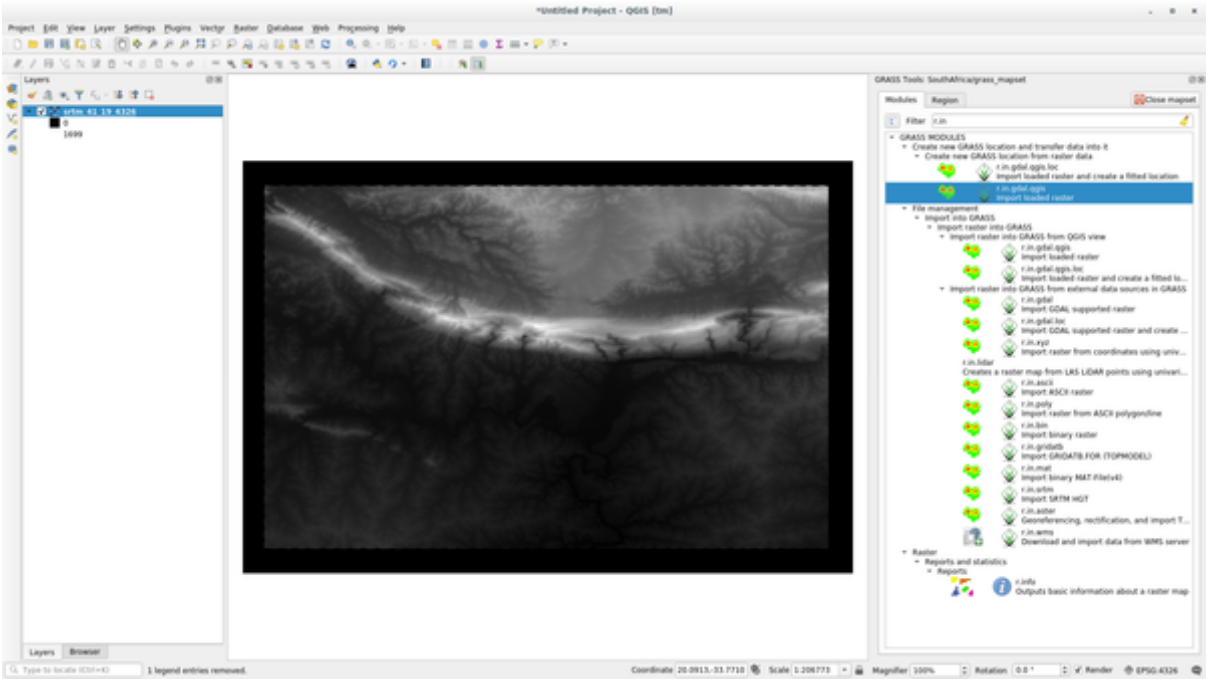
We are going to import in the GRASS Mapset the layer `srtm_41_19_4326.tif`.

Nota: the raster layer is already in the correct CRS, `WGS 84`. If you have layers in different CRS you must reproject them in the same CRS of the GRASS Mapset

1. Load the `srtm_41_19_4326.tif` layer in QGIS
2. Open the *GRASS Tools* dialog again.
3. Click on the *Modules* tab.
4. Search for `r.in.gdal.qgis` (available under *File management* ► *Import into GRASS* ► *Import raster into GRASS* ► *Import raster into GRASS from QGIS view*) and double click the tool to open the tool's dialog.
5. Set it up so that the input layer is `srtm_41_19_4326.tif` and the output is `g_dem`.



6. Click *Run*.
7. When the process is done, click *View output*.
8. *Close* the current tab, and then *Close* the whole dialog.



9. You may now remove the original `srtm_41_19_4326.tif` layer.

12.1.5 ??? Try Yourself: Add Layers to Mapset

Try to import in the GRASS Mapset the vector layers `water.shp` and `places.shp` from the `exercise_data/shapefile/` folder. As we did for `rivers` rename the imported layer as `g_water` and `g_places` to avoid confusion

---

Answer

You can add layers (both vector and raster) into a GRASS Mapset by drag and drop them in the Browser (see *??? Follow Along: Load data using the QGIS Browser*) or by using the `v.in.gdal.qgis` for vector and `r.in.gdal.qgis` for raster layers.

12.1.6 ??? Open an existing GRASS Mapset

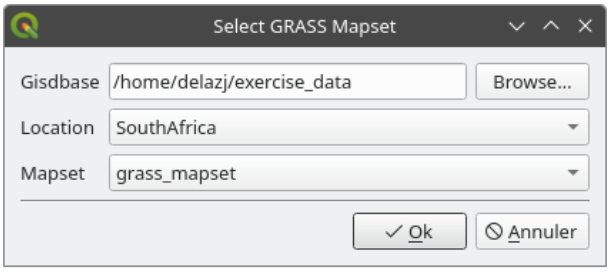
If you have an existing GRASS Mapset you can easily reopen it in another session of QGIS.

You have several method to open a GRASS Mapset, let's explore some of them.

Let's close the Mapset by clicking on the *Close Mapset* button of the *GRASS Tools* window.

??? Follow Along: Using the GRASS plugin

1. Click on the *Plugins* ► *GRASS* ► *Open Mapset* menu.
2. Browse to the GRASS database folder: be careful! You must choose the parent folder, not the GRASS Mapset one. Indeed GRASS will read all the `Locations` of the database and all the `Mapsets` of each `Location`:



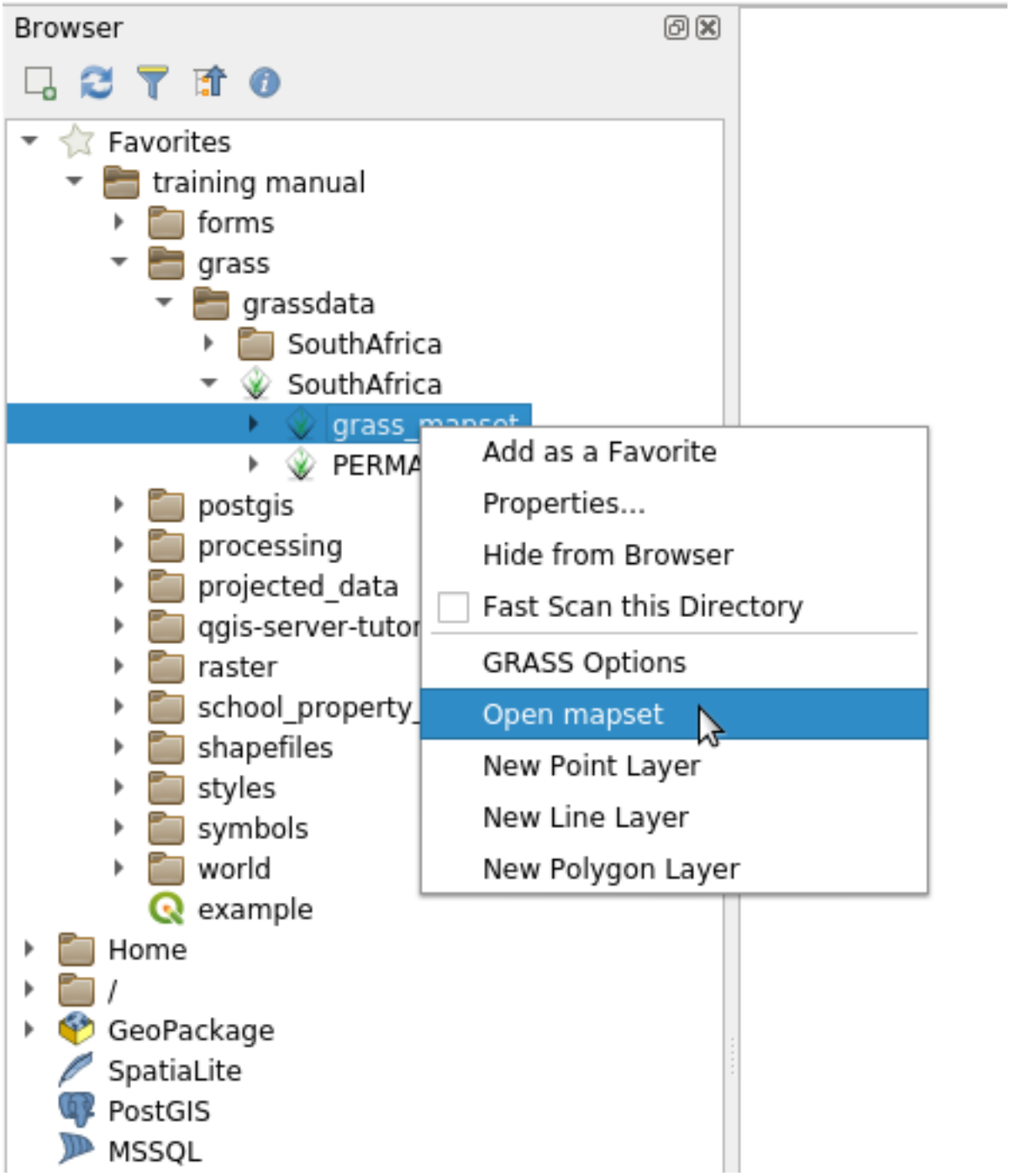
3. Choose the `Location` *SouthAfrica* and the `Mapset` *grass_mapset* that we have created before.

That's it! The GRASS Panel will become active meaning that the Mapset has been correctly opened.

??? Follow Along: Using the QGIS Browser

Even faster and easier is opening a `Mapset` using the QGIS Browser:

1. Close the Mapset (if it is open) by clicking on the *Close Mapset* button of the *GRASS Tools* window.
2. In the QGIS Browser, browse to the folder of the GRASS database.
3. Right click on the Mapset (remember, the Mapset has the GRASS icon next to it). You will see some options.
4. Click on *Open mapset*:



The Mapset is now open and ready to use!

Dica: Right click on a GRASS Mapset offers you a lot of different settings. Try to explore them and see all the useful options.

12.1.7 In Conclusion

The GRASS workflow for ingesting data is somewhat different from the QGIS method because GRASS loads its data into a spatial database structure. However, by using QGIS as a frontend, you can make the setup of a GRASS mapset easier by using existing layers in QGIS as data sources for GRASS.

12.1.8 What's Next?

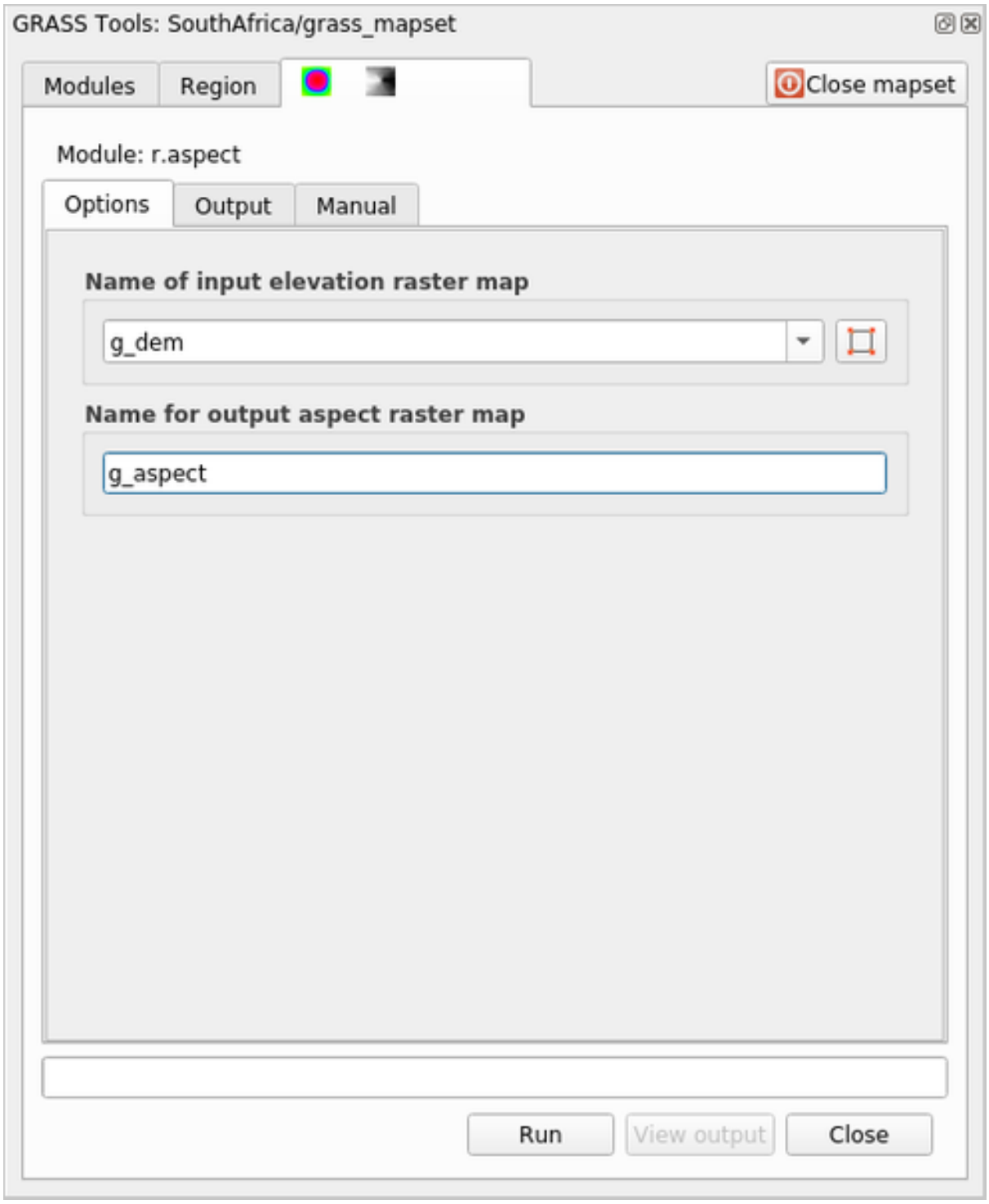
Now that the data is imported into GRASS, we can look at the advanced analysis operations that GRASS offers.

12.2 Lesson: GRASS Tools

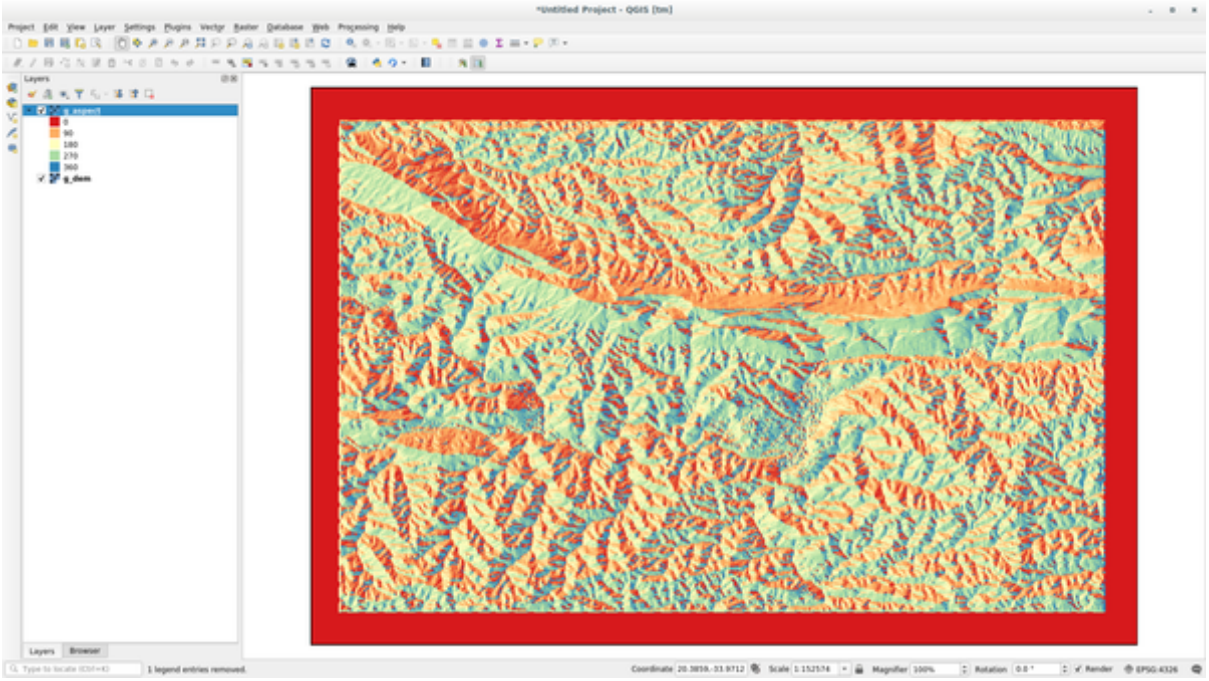
In this lesson we will present a selection of tools to give you an idea of the capabilities of GRASS.

12.2.1 ??? Follow Along: Create an aspect map

1. Open the *GRASS Tools* tab
2. Load the `g_dem` raster layer from the *grass_mapset* Mapset
3. Look for the *r.aspect* module by searching for it in the *Filter* field of the *Modules List* tab
4. Open the tool and set it up like this and click on the *Run* button:



5. When the process is finished click on *View Output* to load the resulting layer in the canvas:

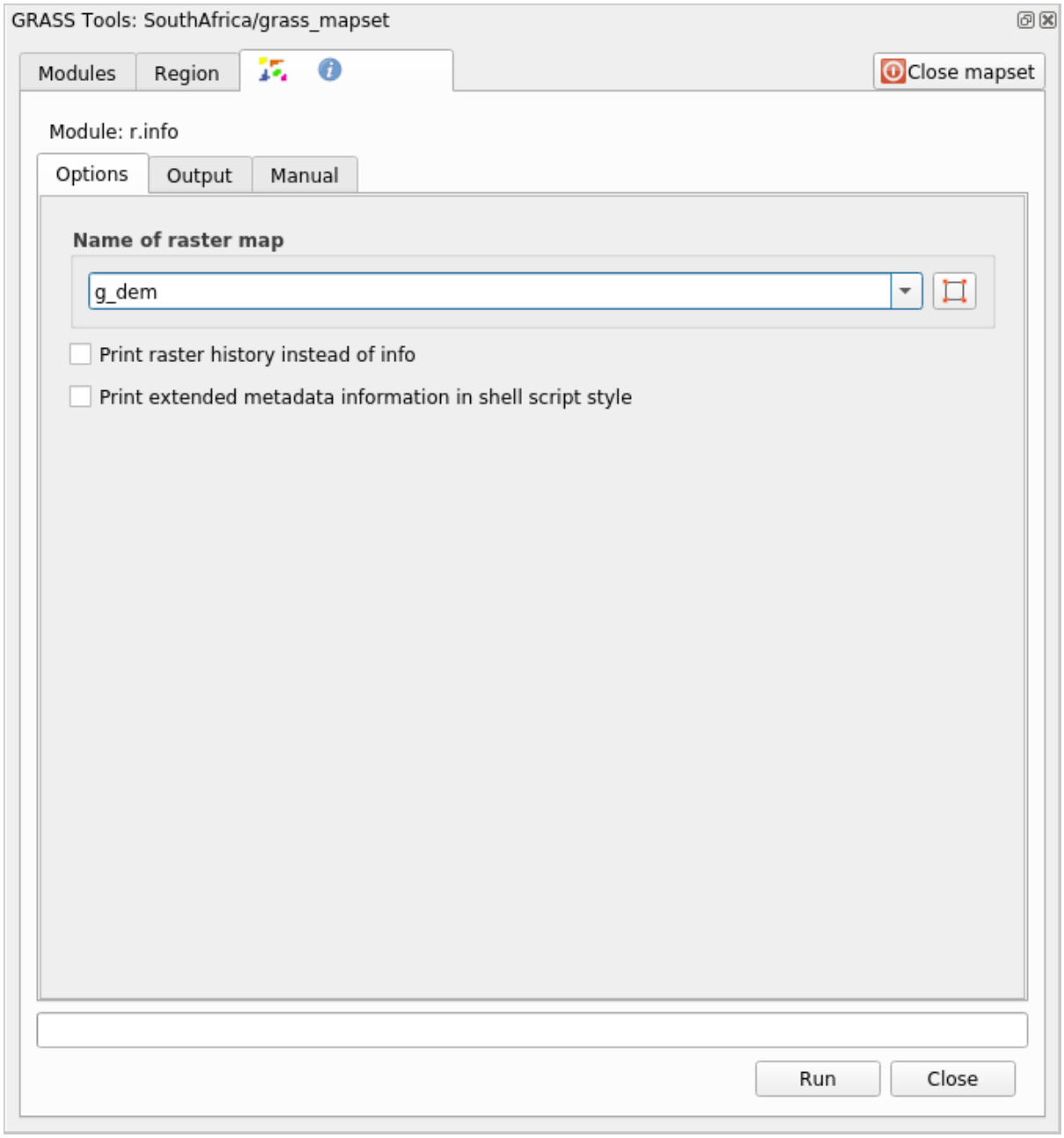


The `g_aspect` layer is stored within the *grass_mapset* Mapset so you can remove the layer from the canvas and reload it whenever you want.

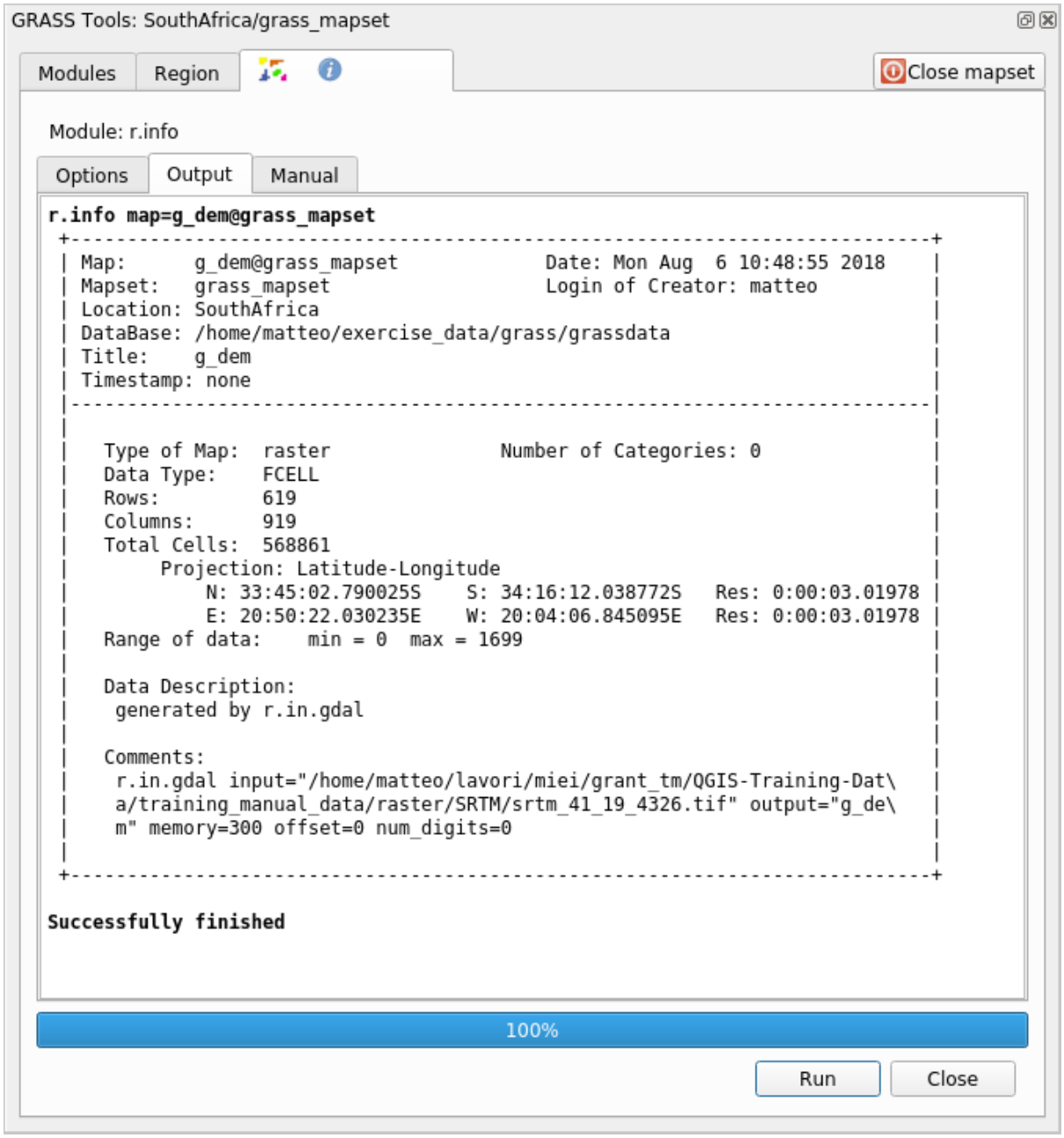
12.2.2 ??? Follow Along: Get basic statistic of raster layer

We want to know some basic statistics of the `g_dem` raster layer.

1. Open the *GRASS Tools* tab
2. Load the `g_dem` raster layer from the *grass_mapset* Mapset
3. Look for the *r.info* module by searching for it in the *Filter* field of the *Modules List* tab
4. Set up the tool like this and click on *Run*:



5. Within the Output tab you will see some raster information printed, like the path of the file, the number of rows and columns and other useful information:



12.2.3 Follow Along: The Reclass Tool

Reclassifying a raster layer is a very useful task. We just created the `g_aspect` layer from the `g_dem` one. The value range gets from 0 (North) passing through 90 (East), 180 (South), 270 (West) and finally to 360 (North again). We can reclassify the `g_aspect` layer to have just 4 **categories** following specific *rules* (North = 1, East = 2, South = 3 and West = 4).

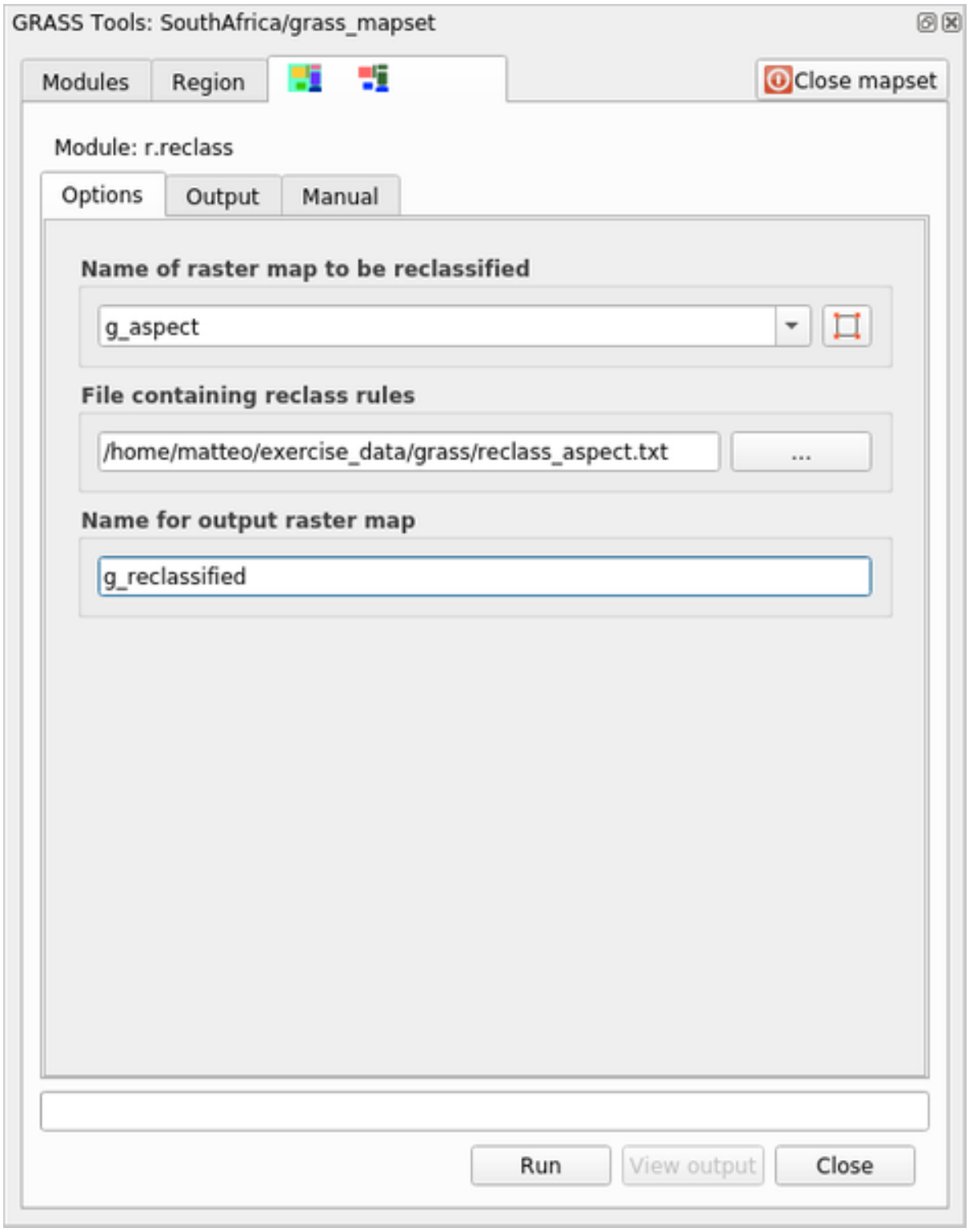
Grass reclassify tool accepts a `txt` file with the defined rules. Writing the rules is very simple and the GRASS Manual contains very good description.

Dica: Each GRASS tool has its own Manual tab. Take the time to read the description of the tool you are using to don't miss some useful parameters

---

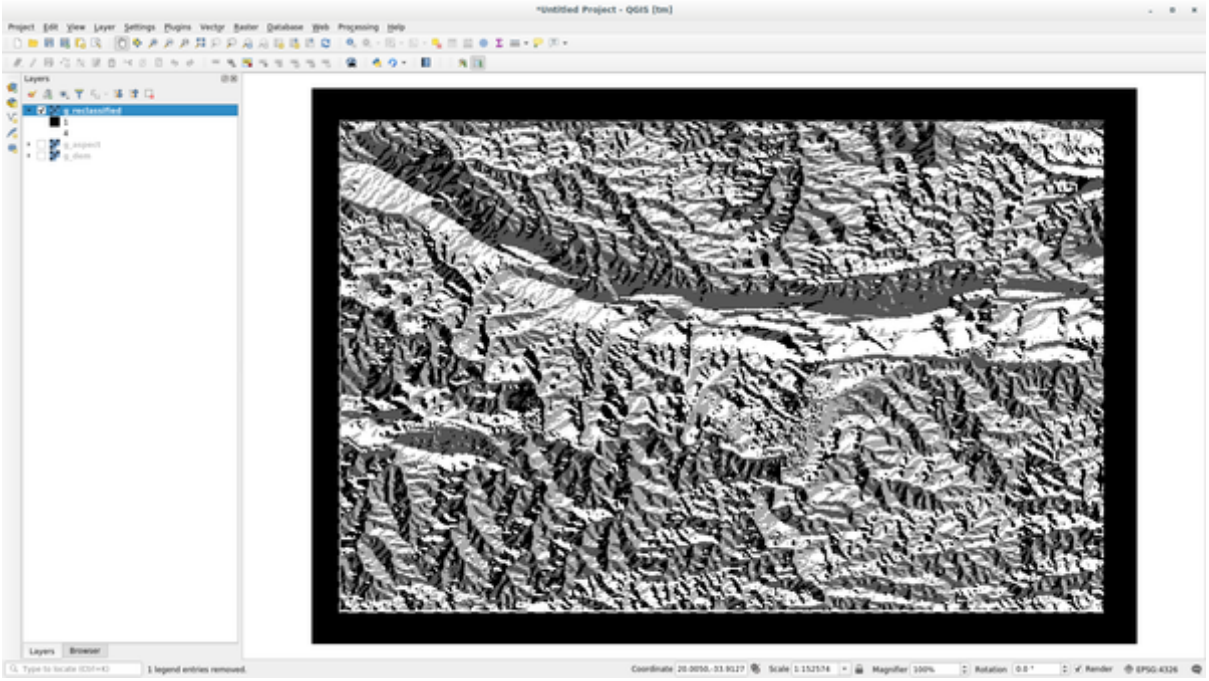
1. Load the `g_aspect` layer or, if you don't have create it, go back to the *Follow Along: Create an aspect map* section.

2. Look for the *r.reclass* module by searching for it in the *Filter* field of the *Modules List* tab
3. Open the tool and set it up like the following picture. The file containing the rules is in the `exercise_data/grass/` folder, named `reclass_aspect.txt`.
4. Click on *Run* and wait until the process is finished:



5. Click on *View Output* to load the reclassified raster in the canvas

   The new layer is made up by just 4 values (1, 2, 3, and 4) and it is easier to manage and to process.



Dica: Open the `reclass_aspect.txt` with a text editor to see the rules and to start becoming used to them. Moreover, take a deep look at the GRASS manual: a lot of different examples are pointed out.

12.2.4 ??? Try Yourself: Reclassify with your rules

Try to reclassify the `g_dem` layer into 3 new categories:

- from 0 to 1000, new value = 1
- from 1000 to 1400, new value = 2
- from 1400 to the maximum raster value, new value = 3

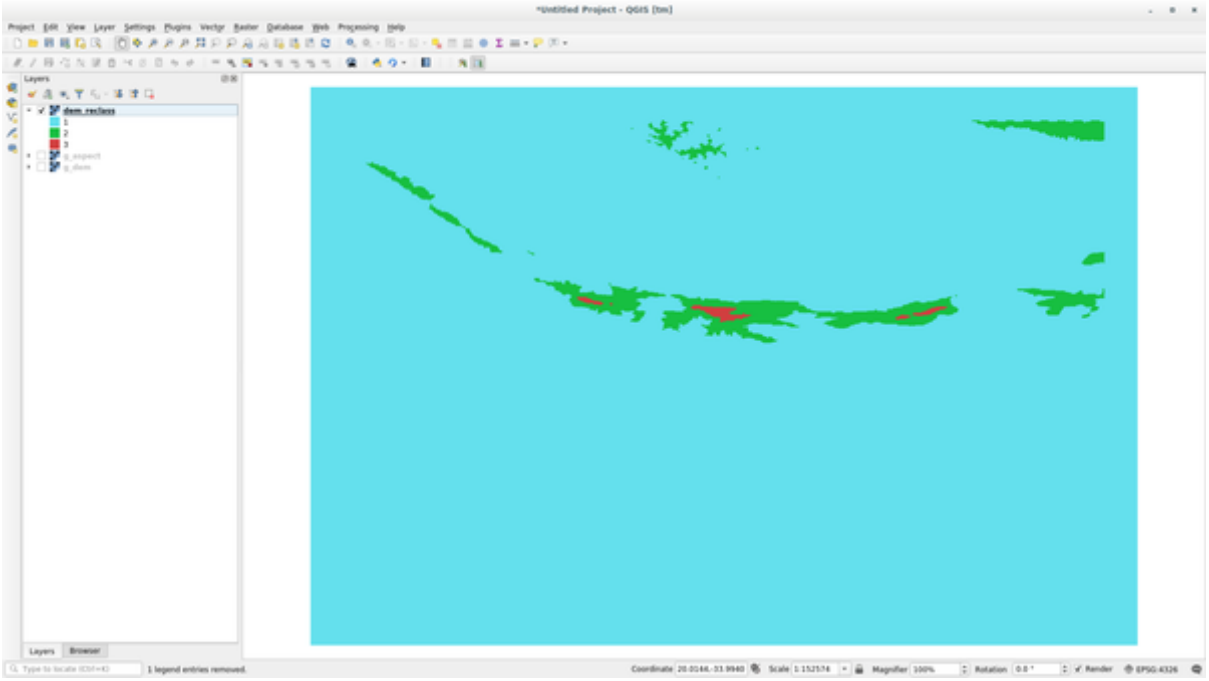
Answer

To discover the maximum value of the raster run the `r.info` tool: in the console you will see that the maximum value is 1699. You are now ready to write the rules.

1. Open a text editor and add the following rules:

```
0 thru 1000 = 1
1000 thru 1400 = 2
1400 thru 1699 = 3
```

2. Save the file as `my_rules.txt` file and close the text editor.
3. Run the `r.reclass` tool: choose the `g_dem` layer and load the file containing the rules you just have saved.
4. Click on *Run* and then on *View Output*. You can change the colors and the final result should look like the following picture:



12.2.5 [?][?][?] Follow Along: The Mapcalc Tool

The Mapcalc tools is similar to the Raster Calculator of QGIS. You can perform mathematical operation on one or more raster layers and the final result will be a new layer with the calculated values.

The aim of the next exercise is to extract the values greater than 1000 from the `g_dem` raster layer.

1. Look for the *r.mapcalc* module by searching for it in the *Filter* field of the *Modules List* tab.
2. Start the tool.

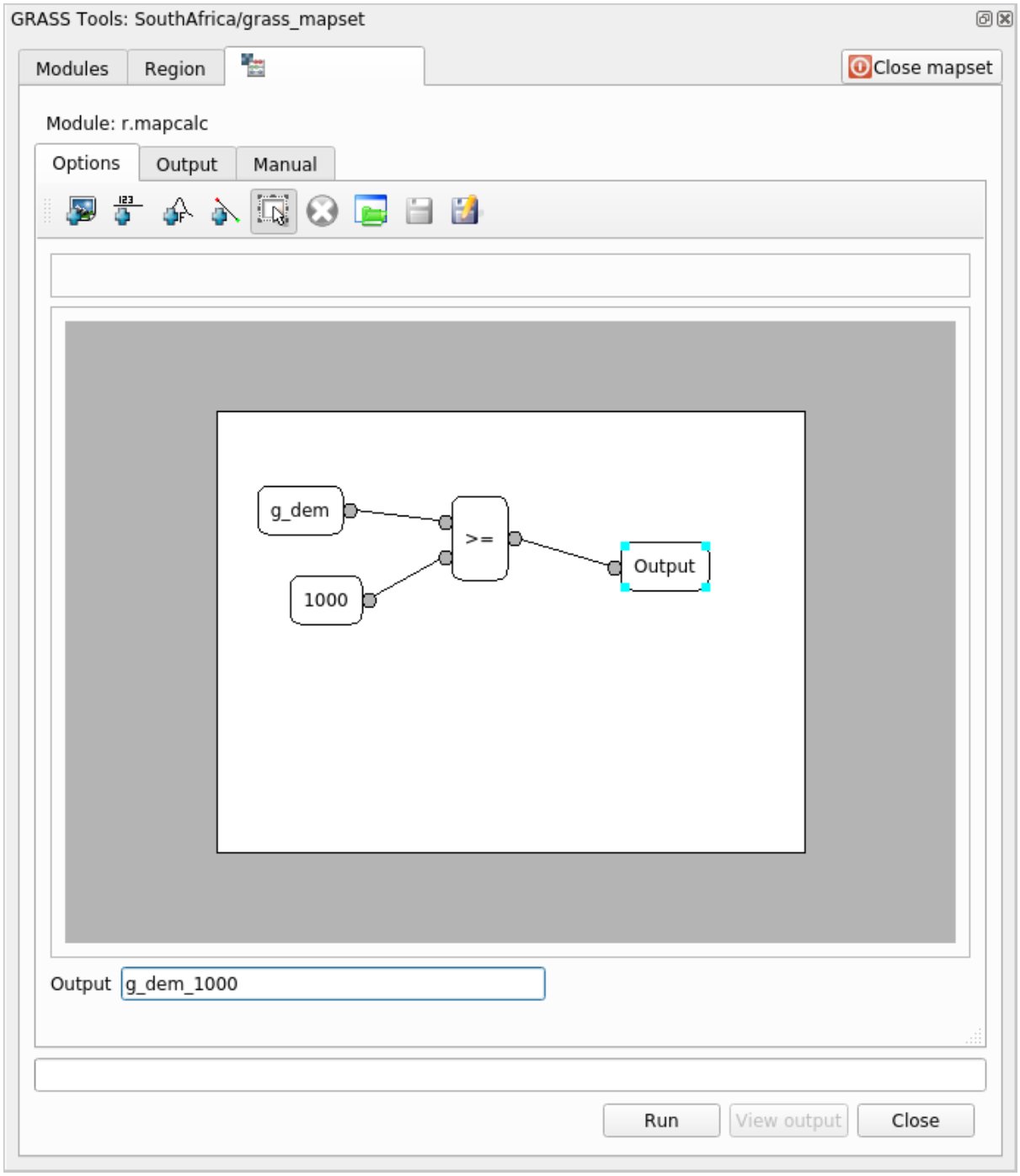
   The *Mapcalc* dialog allows you to construct a sequence of analyses to be performed on a raster, or collection of rasters. You will use these tools to do so:

   In order, they are:

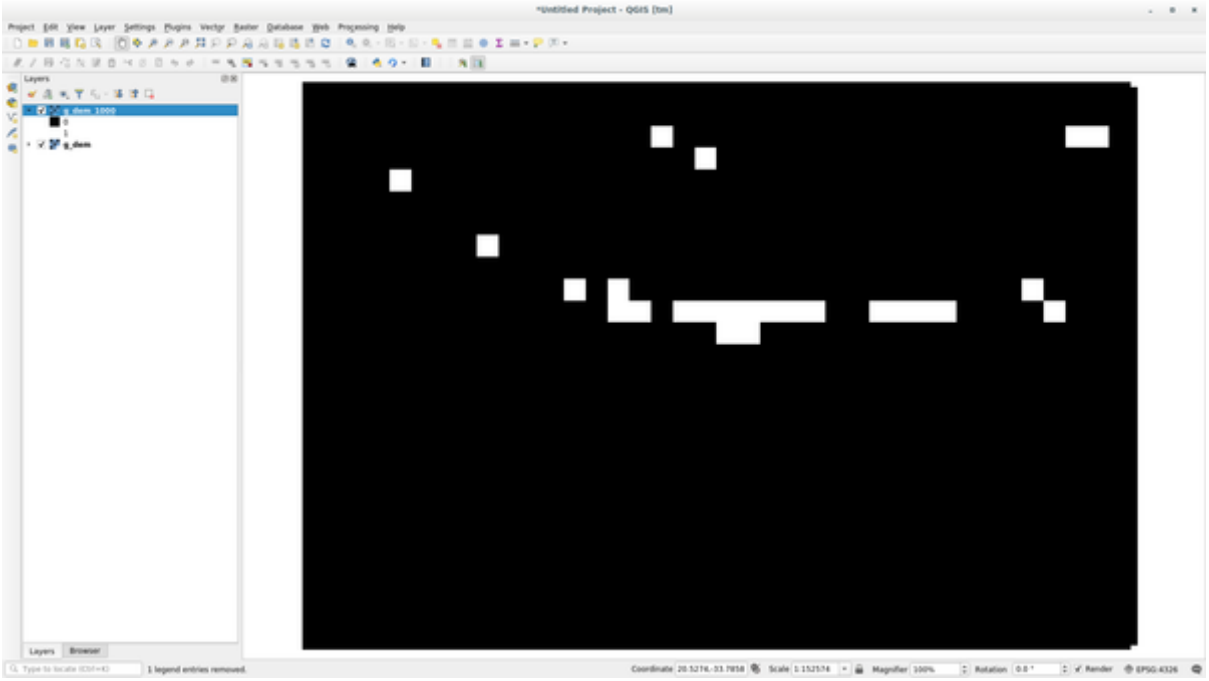
   - *Add map*: Add a raster file from your current GRASS mapset.
   - *Add constant value*: Add a constant value to be used in functions, 1000 in this case
   - *Add operator or function*: Add an operator or function to be connected to inputs and outputs, we will use the operator `greater equals than`
   - *Add connection*: Connect elements. Using this tool, click and drag from the red dot on one item to the red dot on another item. Dots that are correctly connected to a connector line will turn gray. If the line or dot is red, it is not properly connected!
   - *Select item*: Select an item and move selected items.
   - *Delete selected item*: Removes the selected item from the current mapcalc sheet, but not from the mapset (if it is an existing raster)
   - *Open*: Open an existing file with the operation defined

- *Save*: Save all the operation in a file
- *Save as*: Save all the operations as a new file on the disk.

3. Using these tools, construct the following algorithm:



4. Click on *Run* and then on *View output* to see the output displayed in your map:



This shows all the areas where the terrain is higher than 1000 meters.

Dica: You can also save the formula you have created and load it in another QGIS project by clicking on the last button on the GRASS Mapcalc toolbar.

12.2.6 In Conclusion

In this lesson, we have covered only a few of the many tools GRASS offers. To explore the capabilities of GRASS for yourself, open the *GRASS Tools* dialog and scroll down the *Modules List*. Or for a more structured approach, look under the *Modules Tree* tab, which organizes tools by type.

CAPÍTULO 13

Module: Assessment

Use your own data for this section. You will need:

- a point vector dataset of points of interest, with point names and multiple categories
- a line vector dataset of roads
- a polygon vector dataset of land use (using property boundaries)
- a visual-spectrum image (such as an aerial photograph)
- a DEM (downloadable from the CGIAR-CSI if you don't have your own)

13.1 Create a base map

Before doing any data analysis, you will need a base map, which will provide your analysis result with context.

13.1.1 Add the point layer

- Add in the point layer. Based on the level that you're doing the course at, do only what is listed in the appropriate section below:

???

- Label the points according to a unique attribute, such as place names. Use a small font and keep the labels inconspicuous. The information should be available, but shouldn't be a main feature of the map.
- Classify the points themselves into different colors based on a category. For example, categories could include «tourist destination», «police station», and «town center».

???

- Do the same as the ★☆☆ section.
- Classify the point size by importance: the more significant a feature, the larger its point. However, don't exceed the point size of `2.00`.
- For features that aren't located at a single point (for example, provincial/regional names, or town names at a large scale), don't assign any point at all.

???

- Don't use point symbols to symbolize the layer at all. Instead, use labels centered over the points; the point symbols themselves should have no size.
- Use *Data defined settings* to style the labels into meaningful categories.
- Add appropriate columns to the attribute data if necessary. When doing so, don't create fictional data - rather, use the *Field Calculator* to populate the new columns, based on appropriate existing values in the dataset.

13.1.2 Add the line layer

- Add the road layer and then change its symbology. Don't label the roads.

???

- Change the road symbology to a light color with a broad line. Make it somewhat transparent as well.

???

- Create a symbol with multiple symbol layers. The resulting symbol should look like a real road. You can use a simple symbol for this; for example, a black line with a thin white solid line running down the center. It can be more elaborate as well, but the resulting map should not look too busy.
- If your dataset has a high density of roads at the scale you want to show the map at, you should have two road layers: the elaborate road-like symbol, and a simpler symbol at smaller scales. (Use scale-based visibility to make them switch out at appropriate scales.)
- All symbols should have multiple symbol layers. Use symbols to make them display correctly.

???

- Do the same as in the ★★☆ section above.
- In addition, roads should be classified. When using realistic road-like symbols, each type of road should have an appropriate symbol; for example, a highway should appear to have two lanes in either direction.

13.1.3 Add the polygon layer

- Add the land use layer and change its symbology.

???

- Classify the layer according to land use. Use solid colors.

???

- Classify the layer according to land use. Where appropriate, incorporate symbol layers, different symbol types, etc. Keep the results looking subdued and uniform, however. Keep in mind that this will be part of a backdrop!

???

- Use rule-based classification to classify the land use into general categories, such as «urban», «rural», «nature reserve», etc.

13.1.4 Create the raster backdrop

- Create a hillshade from the DEM, and use it as an overlay for a classified version of the DEM itself. You could also use the *Relief* plugin (as shown in the lesson on plugins).

13.1.5 Finalize the base map

- Using the resources above, create a base map using some or all of the layers. This map should include all the basic information needed to orient the user, as well as being visually unified / «simple».

13.2 Analyze the data

- You are looking for a property that satisfies certain criteria.
- You can decide on your own criteria, which you must document.
- There are some guidelines for these criteria:
  - the target property should be of (a) certain type(s) of land use
  - it should be within a given distance from roads, or be crossed by a road
  - it should be within a given distance from some category of points, like a hospital for example

13.2.1 ??? / ???

- Include raster analysis in your results. Consider at least one derived property of the raster, such as its aspect or slope.

13.3 Final Map

- Use the *Print Layout* to create a final map, which incorporates your analysis results.
- Include this map in a document along with your documented criteria. If the map has become too visually busy due to the added layer(s), deselect the layers which you feel are the least necessary.
- Your map must include a title and a legend.

CAPÍTULO 14

Module: Forestry Application

In modules 1 through 13, you have already learned quite a lot about QGIS and how to work with it. If you are interested in learning about some basic forestry applications of GIS, following this module will give you the ability to apply what you have learned and will show you some new useful tools.

The development of this module has been sponsored by the European Union.

14.1 Lesson: Forestry Module Presentation

Following this module about a forestry application requires the knowledge you have learned through the modules 1 to 11 of this training manual. The exercises in the following lessons assume you are already capable of doing many of the basic operations in QGIS and only tools that have not been used before are presented in more detail.

Nevertheless, the module follows a basic level throughout the lessons so that if you have previous experience with QGIS, you can probably follow the instructions without problems.

Note that you need to download an additional data package for this module.

14.1.1 Forestry Sample Data

Nota: The sample data used in this module is part of the training manual data set and is available in the `exercise_data\forestry\` folder.

The forestry related sample data (forestry map, forest data), has been provided by the EVO-HAMK forestry school. The datasets have been modified to adapt to the lessons needs.

The general sample data (aerial images, LiDAR data, basic maps) has been obtained from the National Land Survey of Finland open data service, and adapted for the purposes of the exercises. The open data file download service can be accessed in English here.

Aviso: As for the rest of the training manual, this module includes instructions on adding, deleting and altering GIS datasets. We have provided training datasets for this purpose. Before using the techniques described here on your own data, always ensure you have proper backups!

14.2 Lesson: Georeferencing a Map

A common forestry task would be the update of the information for a forestry area. It is possible that the previous information for that area dates several years back and was collected analogically (that is, in paper) or perhaps it was digitized but all you have left is the paper version of that inventory data.

Most likely you would like to use that information in your GIS to, for example, compare later with later inventories. This means that you will need to digitize the information at hand using your GIS software. But before you can start the digitizing, there is an important first step to be done, scanning and georeferencing your paper map.

The goal for this lesson: To learn to use the Georeferencer tool in QGIS.

14.2.1 ??? Scan the map

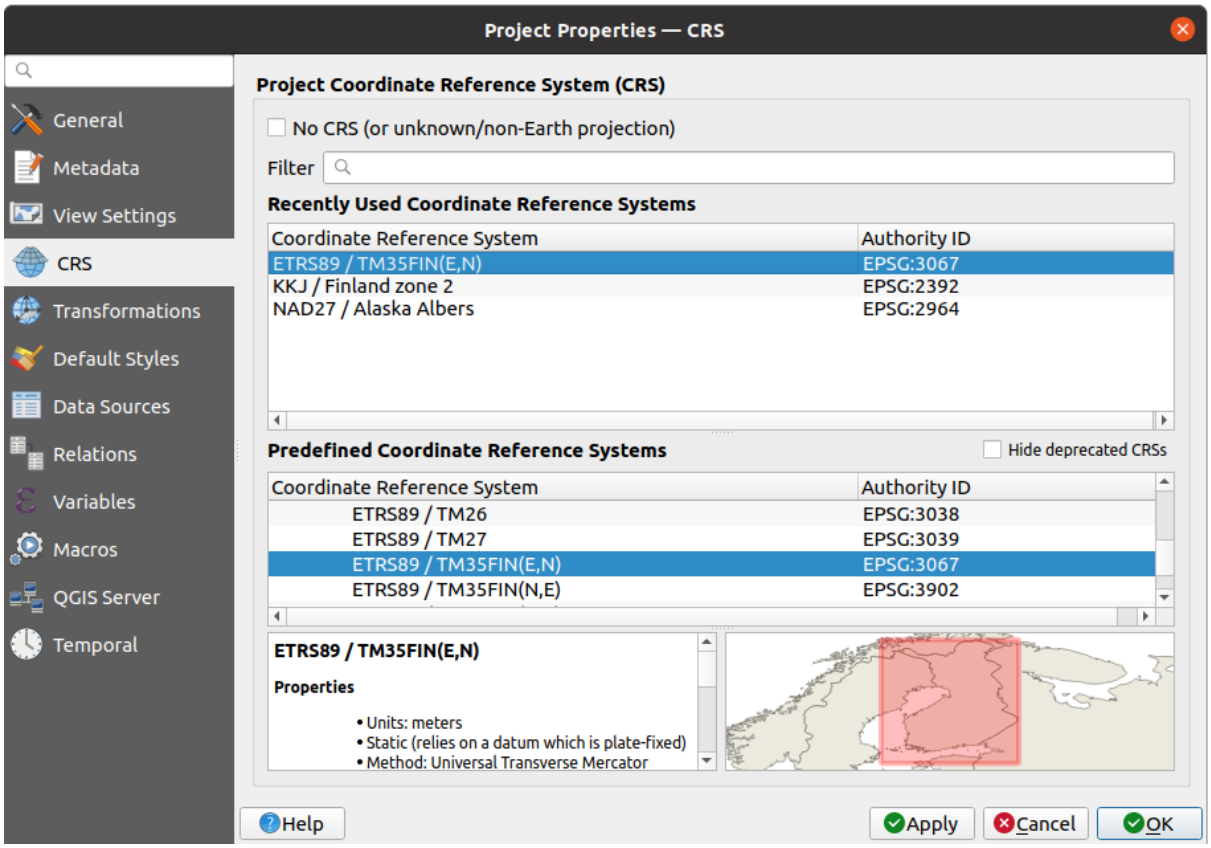
The first task you will have to do is to scan your map. If your map is too big, then you can scan it in different parts but keep in mind that you will have to repeat preprocessing and georeferencing tasks for each part. So if possible, scan the map in as few parts as possible.

If you are going to use a different map that the one provided with this manual, use your own scanner to scan the map as an image file, a resolution of 300 DPI will do. If your map has colors, scan the image in color so that you can later use those colors to separate information from your map into different layers (for ex., forest stands, contour lines, roads…).

For this exercise you will use a previously scanned map, you can find it as `rautjarvi_map.tif` in the data folder `exercise_data/forestry`

14.2.2 ??? Follow Along: Georeferencing the scanned map

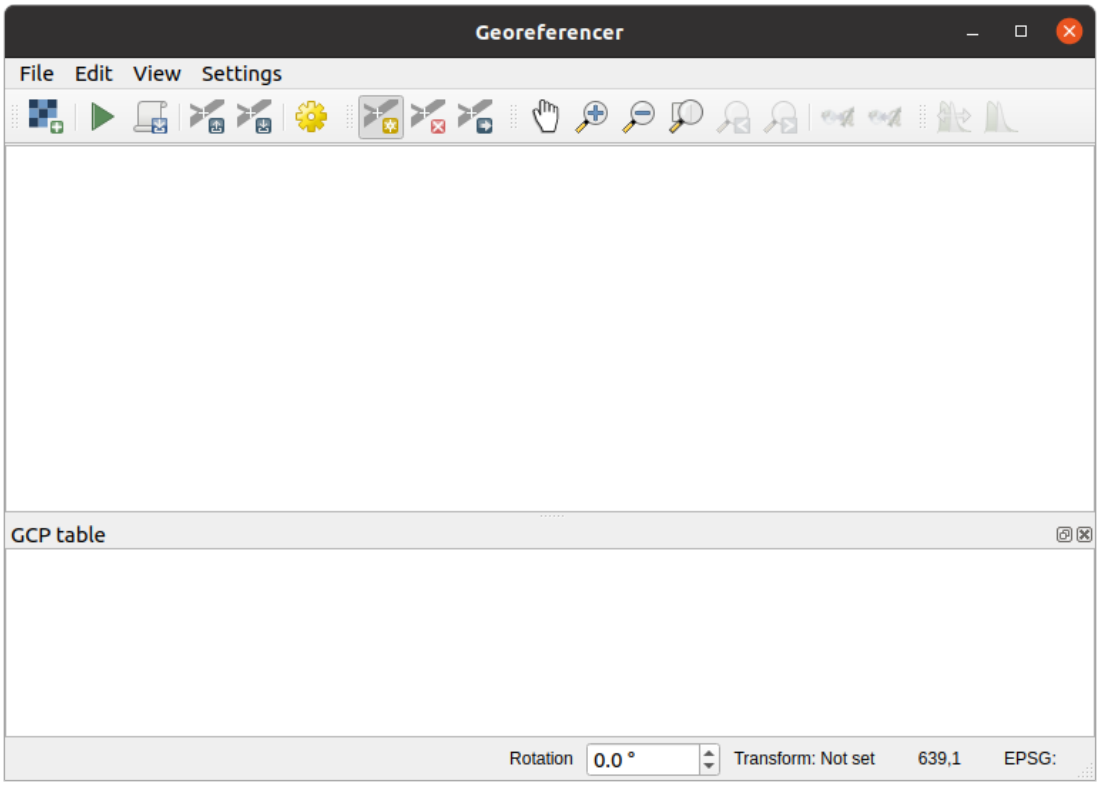
1. Open QGIS and set the project's CRS to `EPSG:3067 - ETRS89 / TM35FIN(E,N)` in *Project* ► *Properties* ► *CRS*, which is the currently used CRS in Finland.



2. Save the QGIS project as `map_digitizing.qgs`.

You will use the georeferencing tool from QGIS named *Georeferencer*. To georeference the map:

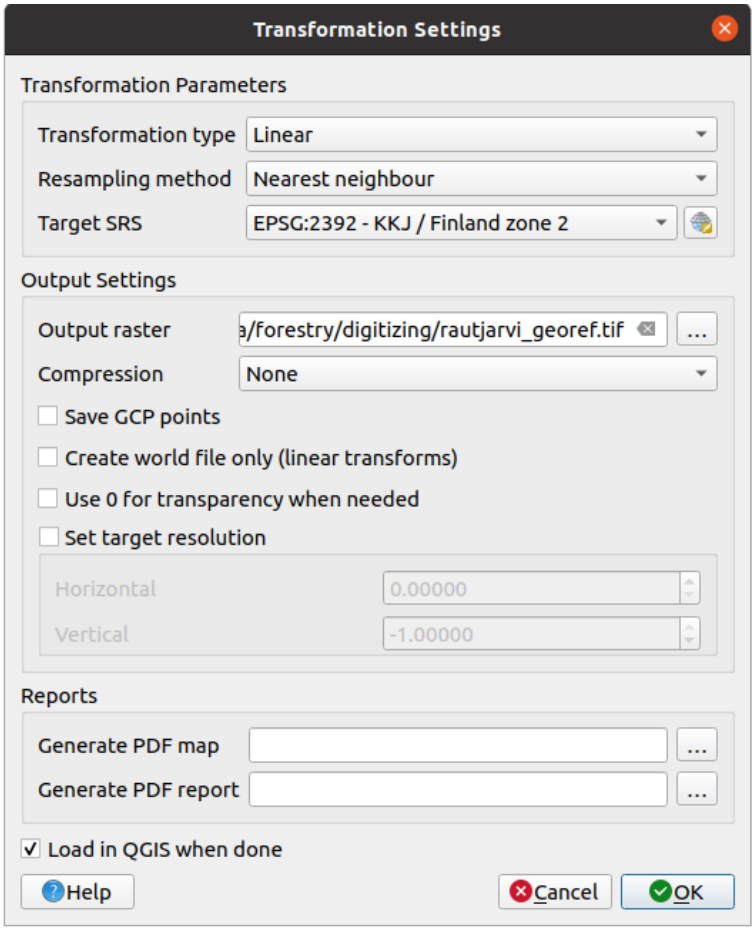
1. Open the georeference tool, *Layer* ► *Georeferencer*....



2. Add the map image file, `rautjarvi_map.tif`, as the image to georeference, *File* ► *Open raster*.
3. Click *OK*.

Next you should define the transformation settings for georeferencing the map:

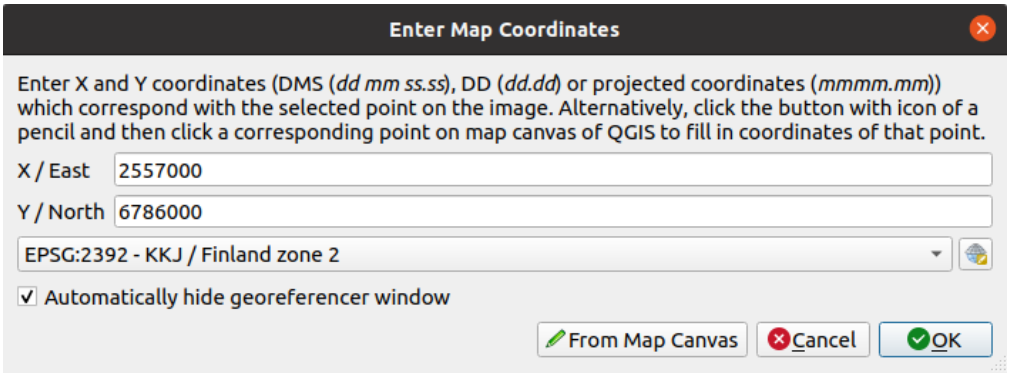
1. Open *Settings* ► *Transformation settings*.
2. Set the *Transformation type* to `Linear` and the *Resampling method* to `Nearest neighbour`.
3. Press the Select CRS button next to the *Target SRS* option and select the `EPSG:2392 - KKJ / Finland zone 2` CRS; it is the CRS that was used in Finland back in 1994 when this map was created.
4. Click the icon next to the *Output raster* box, go to the folder and create the folder `exercise_data\forestry\digitizing` and name the file as `rautjarvi_georef.tif`.
5. Check *Load in QGIS when done*
6. Leave the rest of parameters as default.



7. Click *OK*.

The map contains several cross-hairs marking the coordinates in the map, we will use those to georeference this image. You can use the zooming and panning tools as you usually do in QGIS to inspect the image in the Georeferencer's window.

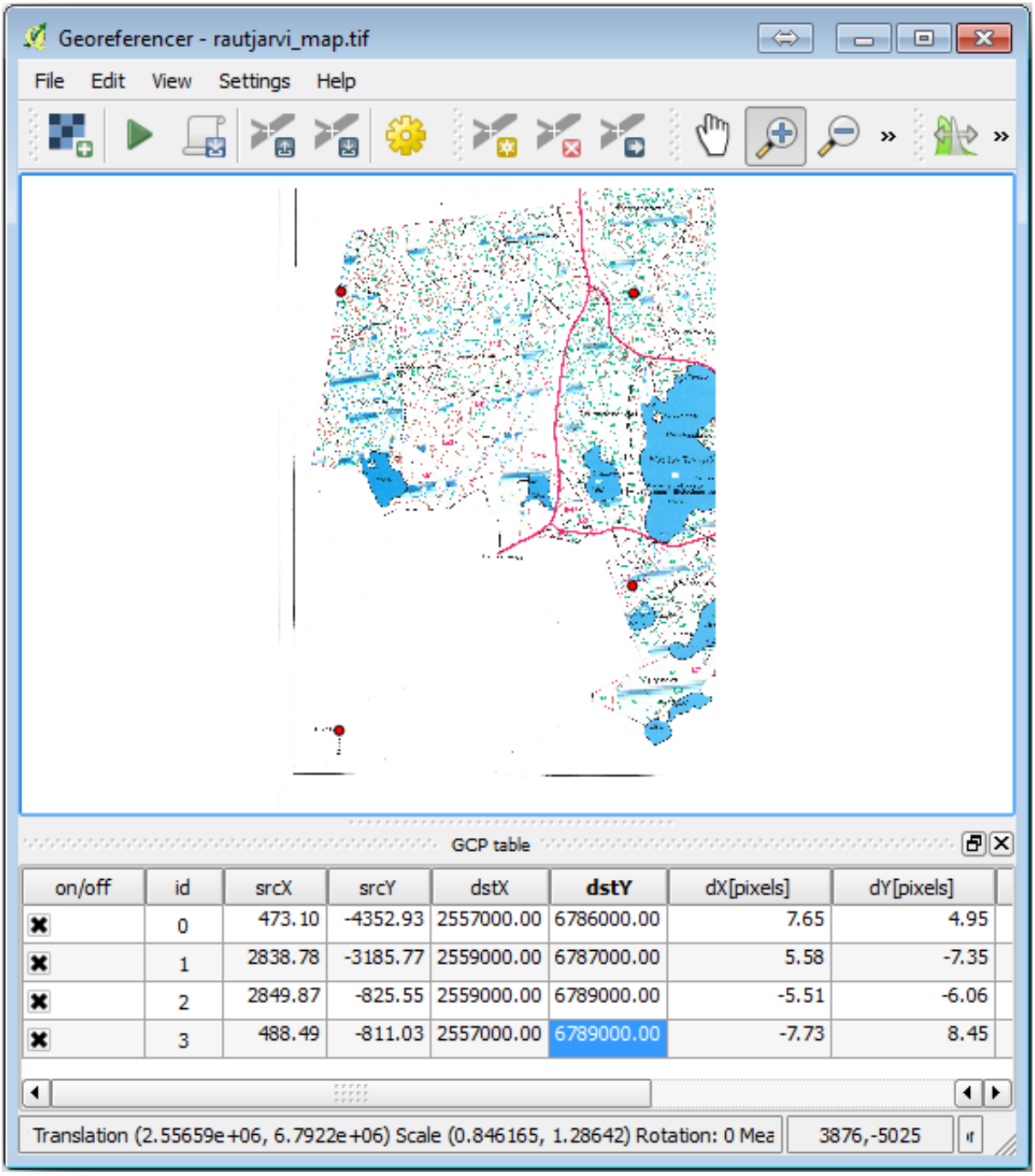
1. Zoom in to the left lower corner of the map and note that there is a cross-hair with a coordinate pair, X and Y, that as mentioned before are in `KKJ / Finland zone 2` CRS. You will use this point as the first ground control point for the georeferencing your map.
2. Select the *Add point* tool and click in the intersection of the cross-hairs (pan and zoom as needed).
3. In the *Enter map coordinates* dialogue write the coordinates that appear in the map (X: 2557000 and Y: 6786000) and their CRS (`EPSG:2392 - KKJ / Finland zone 2`)



4. Click *OK*.

The first coordinate for the georeferencing is now ready.

5. Look for other black cross-hairs in the image, they are separated 1000 meters from each other both in North and East direction. You should be able to calculate the coordinates of those points in relation to the first one.
6. Zoom out in the image and move to the right or the top until you find other cross-hair, and estimate how many kilometres you have moved. Try to get ground control points as far from each other as possible.
7. Digitize at least three more ground control points in the same way you did the first one. You should end up with something similar to this:



With already three digitized ground control points you will be able to see the georeferencing error as a red line coming out of the points. The error in pixels can be seen also in the *GCP table* in the *dX[pixels]* and *dY[pixels]* columns. The error in pixels should not be higher than 10 pixels. If it is, you should review the points you have digitized and the coordinates you have entered to find what the problem is. You can use the image above as a guide.

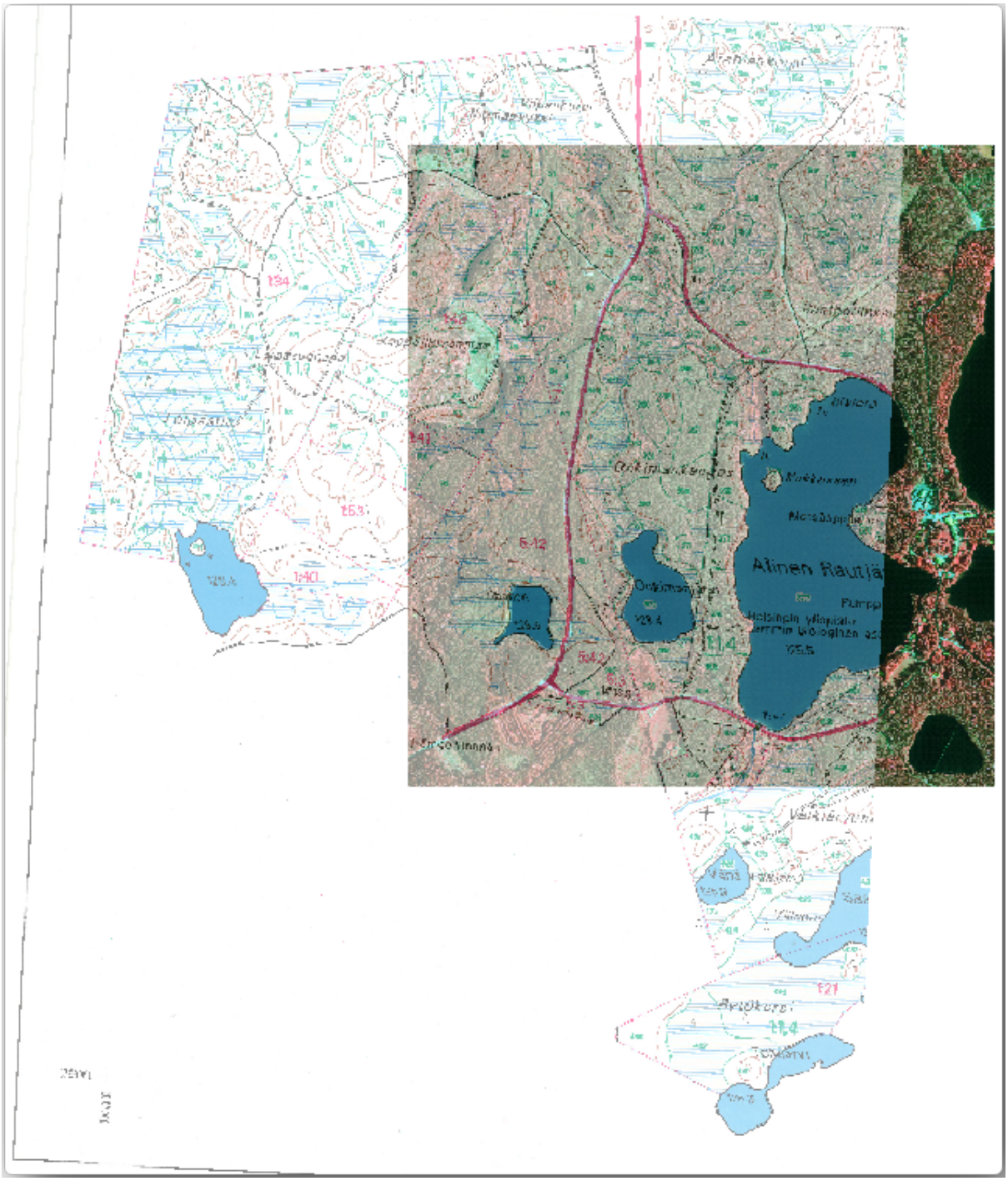
Once you are happy with your control points, you can save them for later use:

1. Go to *File* ► *Save GCP points as…*
2. In the folder `exercise_data\forestry\digitizing`, name the file `rautjarvi_map.tif.points`.

Finally, georeference your map:

1. Go to *File* ► *Start georeferencing*.
2. Note that you named the file already as `rautjarvi_georef.tif` when you edited the Georeferencer settings.

   Now you can see the map in QGIS project as a georeferenced raster. Note that the raster seems to be slightly rotated, but that is simply because the data is in `KKJ / Finland zone 2` and your project is in `ETRS89 / TM35FIN(E,N)`.
3. To check that your data is properly georeferenced, you can
   1. Open the aerial image in the `exercise_data\forestry` folder, named `rautjarvi_aerial.tif`.
   2. Your map and this image should match quite well: set the map transparency to 50% and compare it to the aerial image.



4. Save the changes to your QGIS project, you will continue from this point for the next lesson.

14.2.3 In Conclusion

You have now georeferenced a paper map, making it possible to use it as a map layer in QGIS.

14.2.4 What's Next?

In the next lesson, you will digitize the forest stands in your map as polygons and add the inventory data to them.

14.3 Lesson: Digitizing Forest Stands

Unless you are going to use your georeferenced map as a simple background image, the next natural step is to digitize elements from it. You have already done so in the exercises about creating vector data in *Lesson: Creating a New Vector Dataset*, when you digitized the school fields. In this lesson, you are going to digitize the forest stands" borders that appear in the map as green lines but instead of doing it using an aerial image, you will use your georeferenced map.

The goal for this lesson: Learn a technique to help the digitizing task, digitizing forest stands and finally adding the inventory data to them.

14.3.1 ??? Follow Along: Extracting the Forest Stands Borders

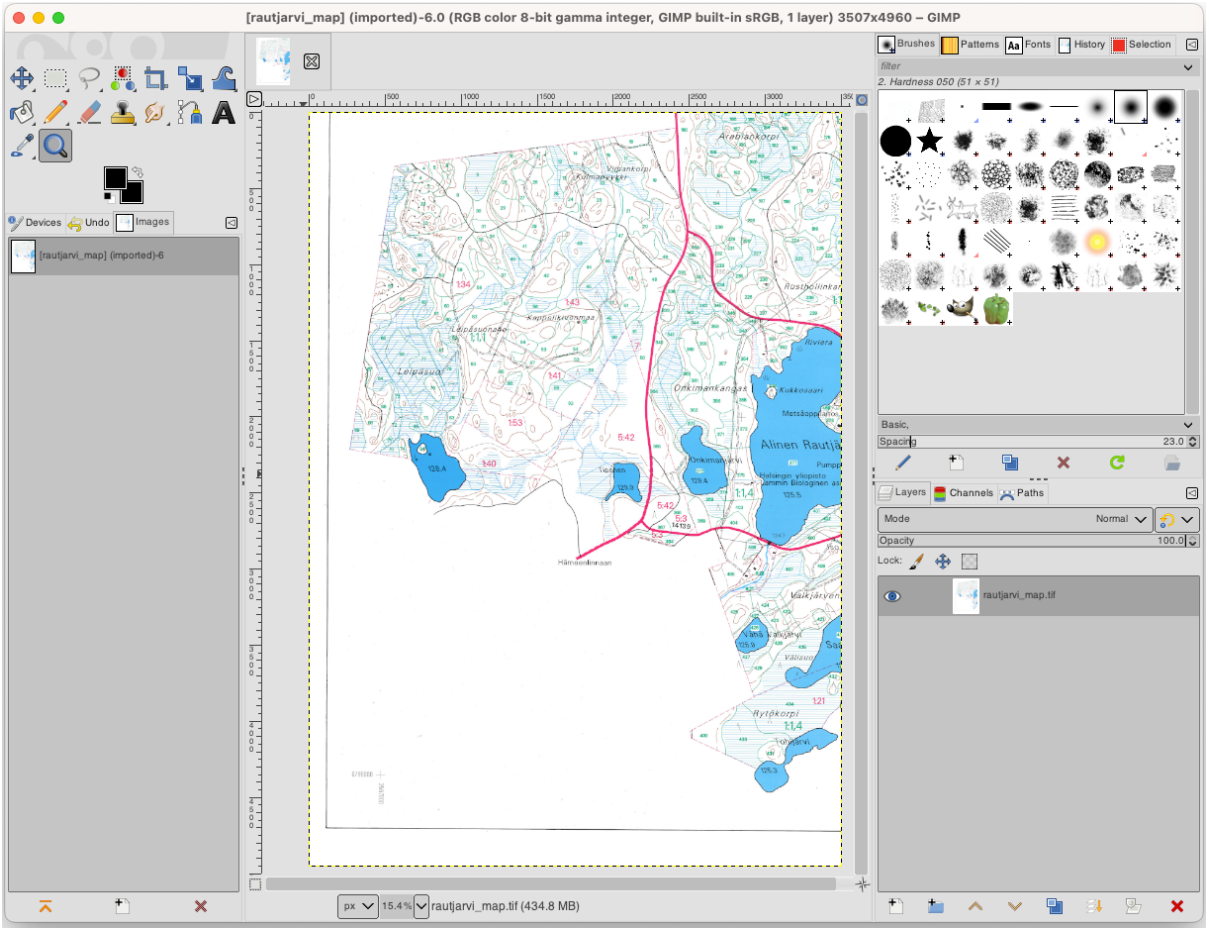
Open your `map_digitizing.qgs` project in QGIS, that you saved from the previous lesson.

Once you have scanned and georeferenced your map you could start to digitize directly by looking at the image as a guide. That would most likely be the way to go if the image you are going to digitize from is, for example, an aerial photograph.

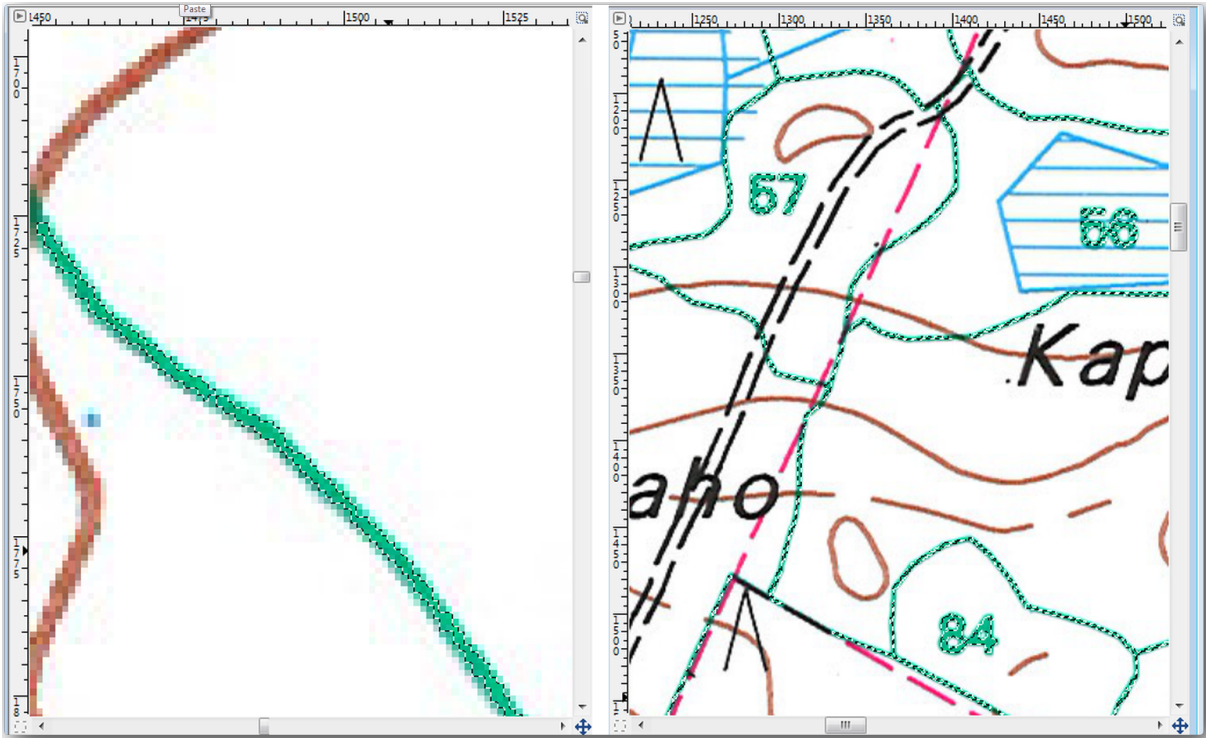
If what you are using to digitize is a good map, as it is in our case, it is likely that the information is clearly displayed as lines with different colors for each type of element. Those colors can be relatively easy extracted as individual images using an image processing software like GIMP. Such separate images can be used to assist the digitizing, as you will see below.

The first step will be to use GIMP to obtain an image that contains only the forest stands, that is, all those greenish lines that you could see in the original scanned map:

1. Open GIMP (if you don't have it installed yet, download it from the internet or ask your teacher).
2. Open the original map image, *File* ► *Open*, `rautjarvi_map.tif` in the `exercise_data/forestry` folder. Note that the forest stands are represented as green lines (with the number of the stand also in green inside each polygon).

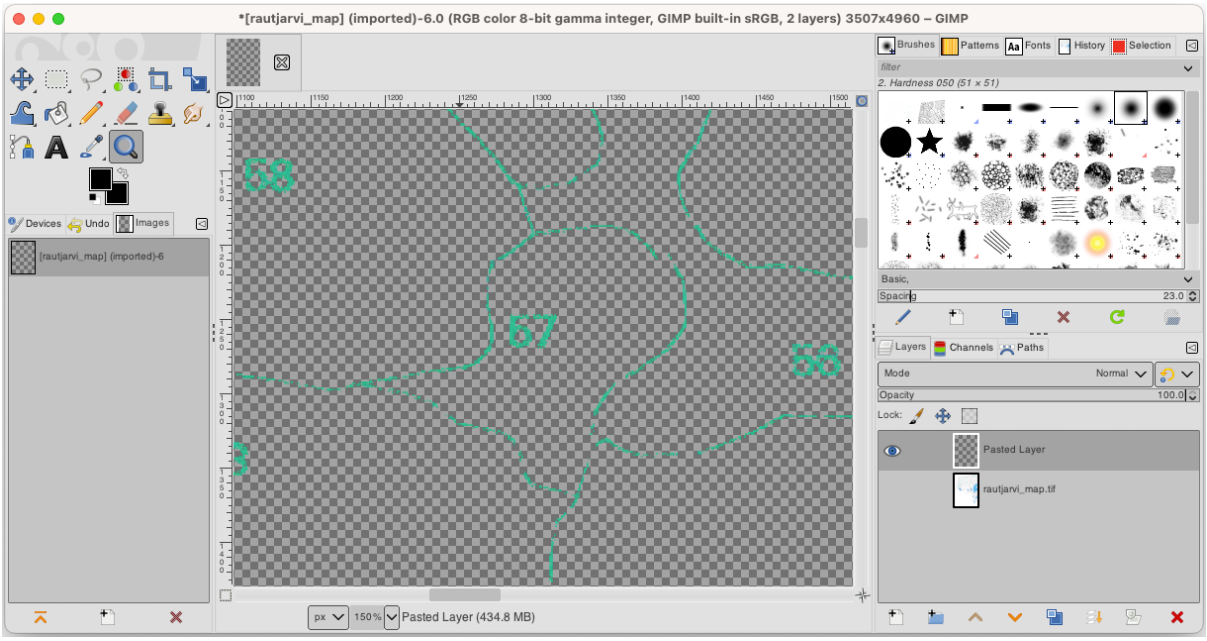


3. Now you can select the pixels in the image that are making up the forest stands" borders (the greenish pixels):
    1. Open the tool *Select* ► *By color*.
    2. With the tool active, zoom into the image (`Ctrl` + mouse wheel) so that a forest stand line is close enough to differentiate the pixels forming the line. See the left image below.
    3. Click and drag the mouse cursor in the middle of the line so that the tool will collect several pixel color values.
    4. Release the mouse click and wait a few seconds. The pixels matching the colors collected by the tool will be selected through the whole image.
    5. Reduza o zoom para ver como os pixeis esverdeados foram seleccionadas em toda a imagem.
    6. Se você não está feliz com o resultado, repita o clique e operação de arrastar.
    7. O pixel que seleccionou deve ser algo como a imagem mostra abaixo.



4. Se não está feliz com o resultado, repita o clique e operação de arrastar.

    1. Copy (`Ctrl+C` or *Edit* ► *Copy*) the selected pixels.
    2. And paste the pixels directly as a new layer (*Edit* ► *Paste as* ► *New Layer*). GIMP will display the pasted pixels as a new layer (`Pasted Layer`) in the *Layers* panel.
    3. Click the «eye» icon next to the original image layer to switch it off, so that only the *Pasted Layer* is visible:



5. Finally, select *File* ► *Export As…*, set *Select File Type (By Extension)* as a *TIFF image*, select the `digitizing` folder and name it `rautjarvi_map_green.tif`. Select no compression when asked.

You could do the same process with other elements in the image, for example extracting the black lines that represent roads or the brown ones that represent the terrain" contour lines. But for us, the forest stands is enough.

14.3.2 ??? Try Yourself: Georeference the Green Pixels Image

As you did in the previous lesson, you need to georeference this new image to be able to use it with the rest of your data.

Note that you don't need to digitize the ground control points anymore because this image is basically the same image as the original map image, as far as the Georeferencer tool is concerned. Here are some things you should remember:

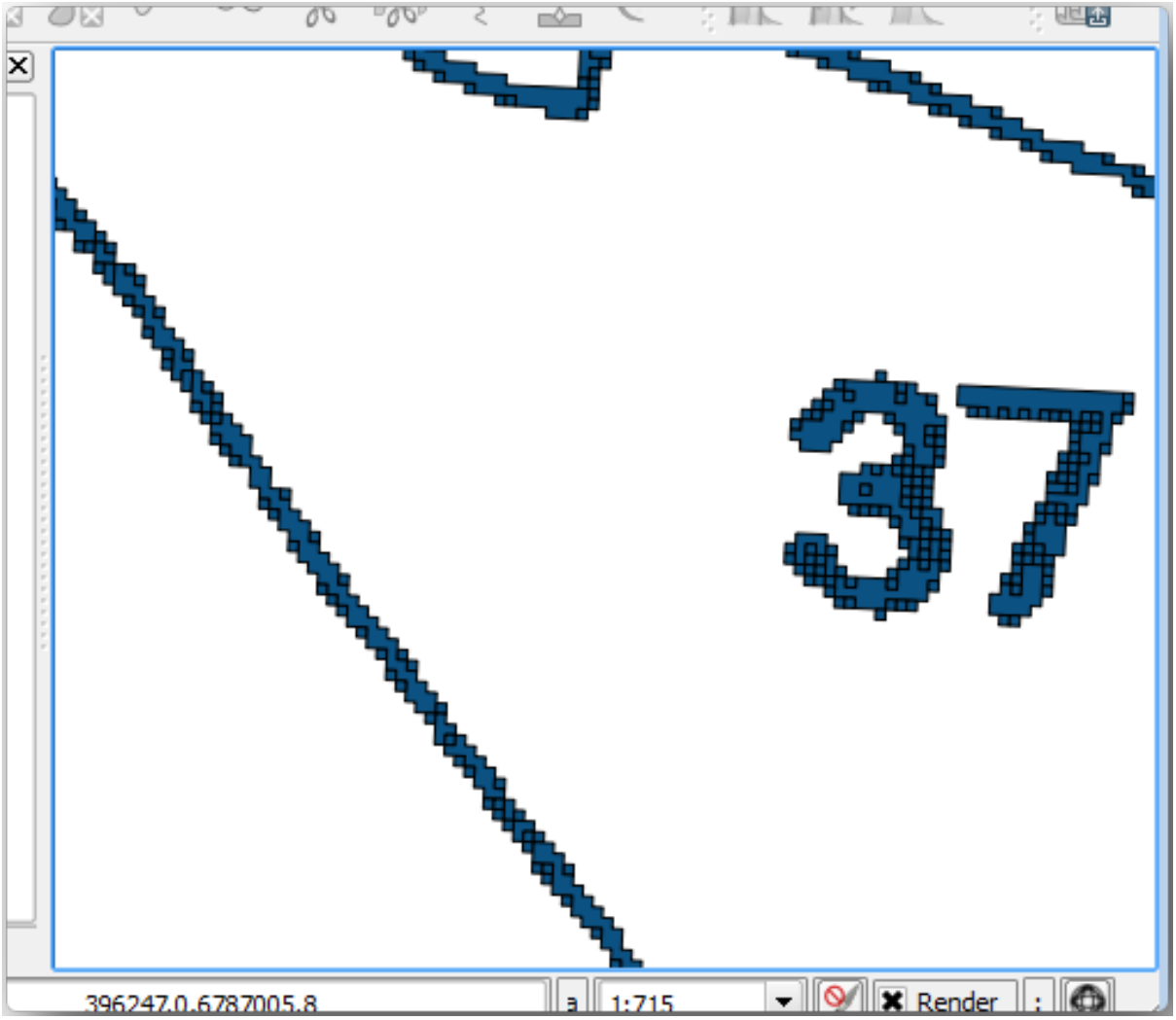
- This image is also, of course, in `KKJ / Finland zone 2` CRS.
- You should use the ground control points you saved, *File* ► *Load GCP points*.
- Remember to review the *Transformation settings*.
- Name the output raster as `rautjarvi_green_georef.tif` in the `digitizing` folder.

Check that the new raster is fitting nicely with the original map.

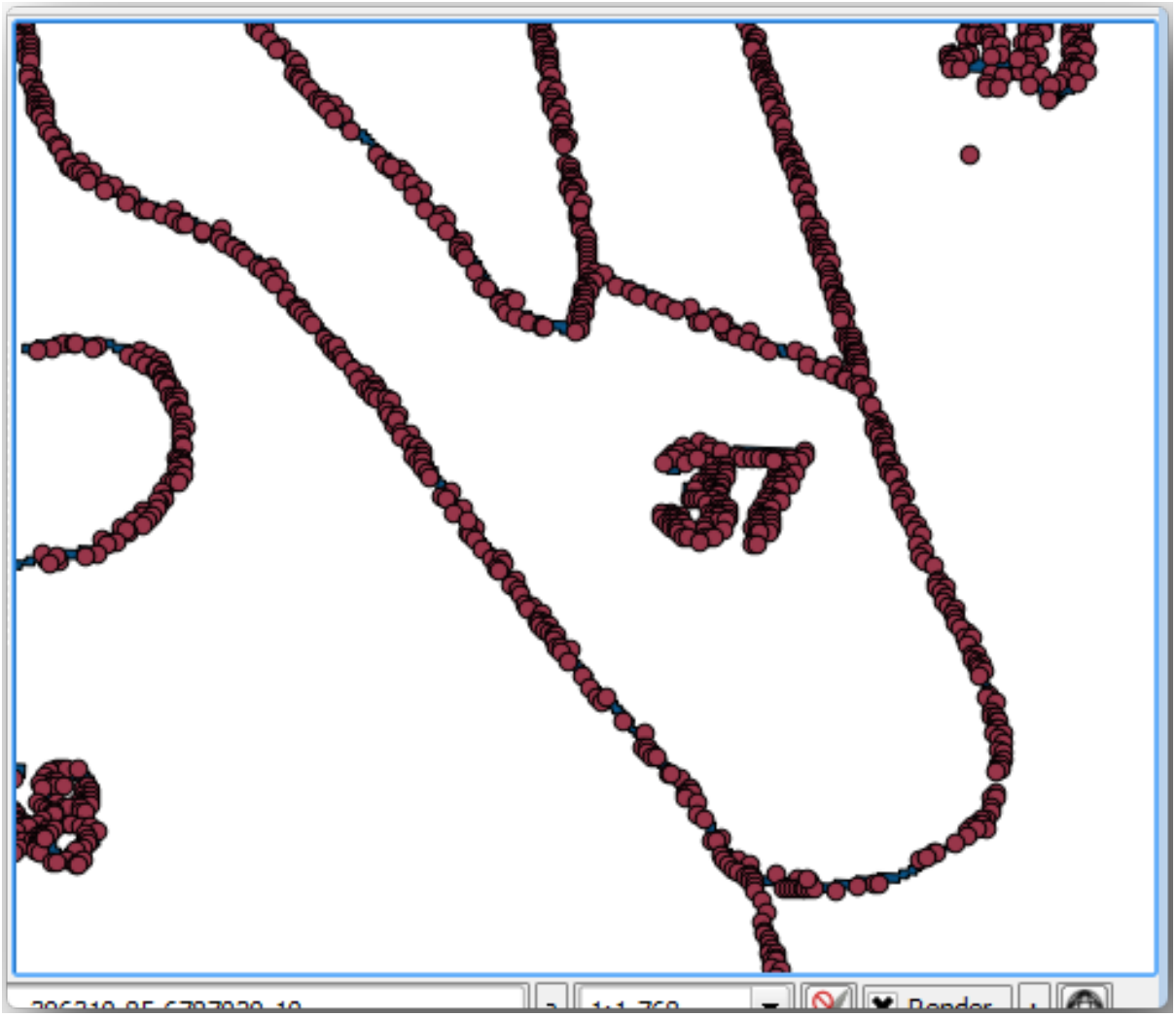
14.3.3 ??? Follow Along: Creating Supporting Points for Digitizing

Having in mind the digitizing tools in QGIS, you might already be thinking that it would be helpful to snap to those green pixels while digitizing. That is precisely what you are going to do next create points from those pixels to use them later to help you follow the forest stands" borders when digitizing, by using the snapping tools available in QGIS.

1. Use the *Raster* ► *Conversion* ► *Polygonize (Raster to Vector)* tool to vectorize your green lines to polygons. If you don't remember how, you can review it in *Lesson: Raster to Vector Conversion*.
2. Save as `rautjarvi_green_polygon.shp` inside the `digitizing` folder.
3. Zoom in and see what the polygons look like. You will get something like this:



4. Next option to get points out of those polygons is to get their centroids:
   1. Open *Vector* ► *Geometry Tools* ► *Centroids…*.
   2. Set *Input Layer* to *rautjarvi_green_polygon* (the polygon layer you have just created)
   3. Set *Centroids* output to `green_centroids.shp` file within the folder `digitizing`
   4. Check *Open output file after running algorithm*
   5. Press *Run*. This will calculate the centroids for the polygons as a new layer and add it to the project.

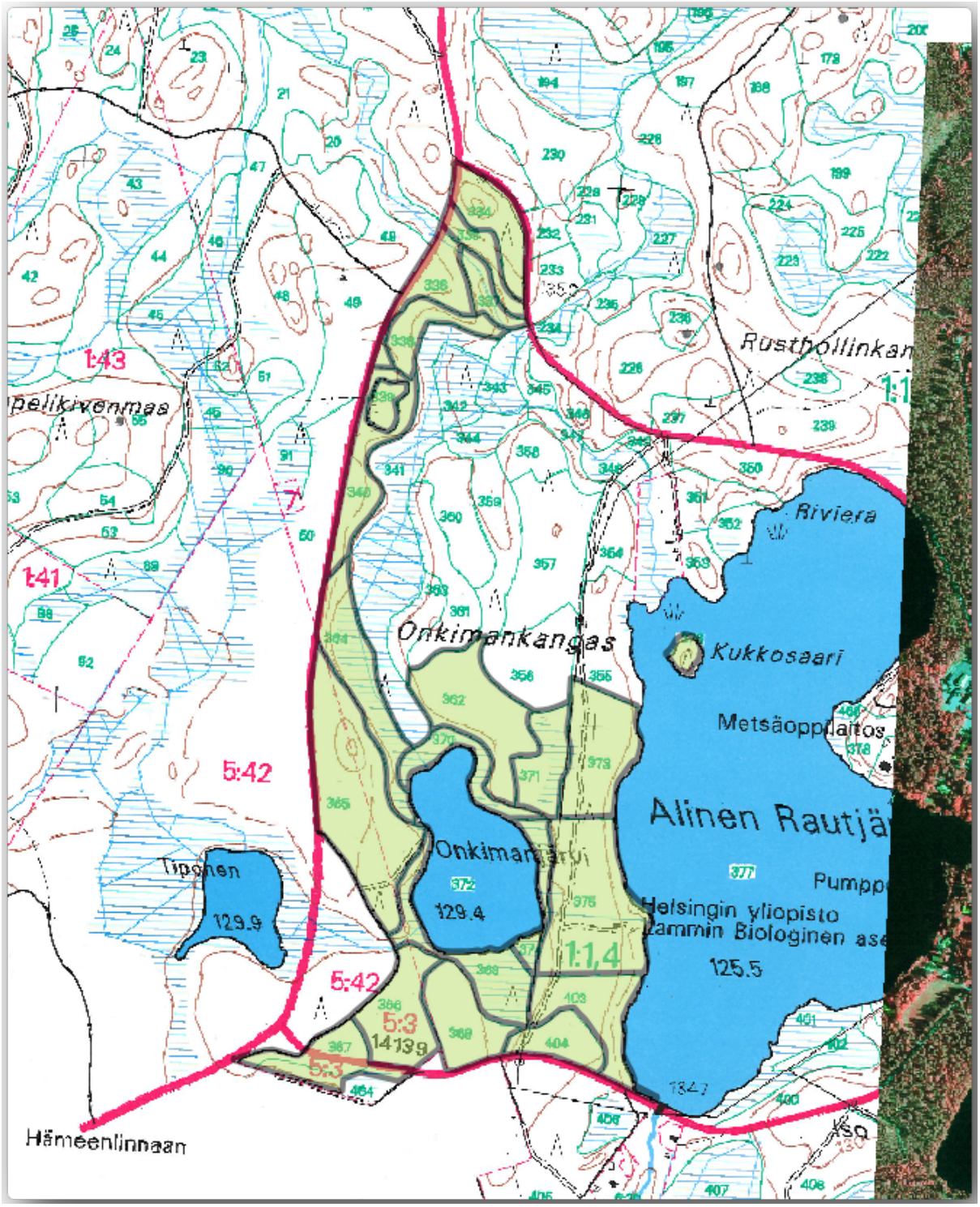


5. Now you can remove the `rautjarvi_green_polygon` layer from the TOC.
6. Change the symbology of the centroids layer as follows:
    1. Open the *Layer Properties* for `green_centroids`.
    2. Go to the *Symbology* tab.
    3. Set *Size* to `1.00` and choose *Map Units*

It is not necessary to differentiate points from each other, you just need them to be there for the snapping tools to use them. You can use those points now to follow the original lines much easily than without them.

14.3.4 ??? Follow Along: Digitize the Forest Stands

Now you are ready to start with the actual digitizing work. You would start by creating a vector file of *polygon type*, but for this exercise, there is a shapefile with part of the area of interest already digitized. You will just finish digitizing the half of the forest stands that are left between the main roads (wide pink lines) and the lake:

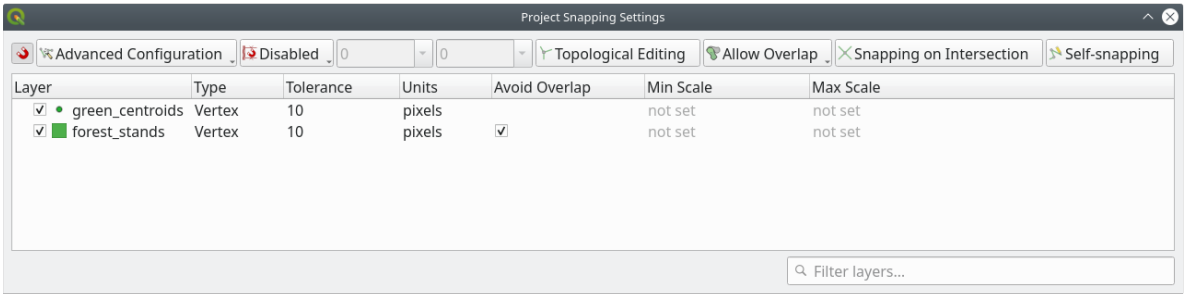


1. Go to the `digitizing` folder using your file manager browser.

2. Drag and drop the `forest_stands.shp` vector file to your map.
3. Change the new layer's symbology so that it will be easier to see the polygons that have already been digitized.
    1. Set *Fill color* to green - and change the *Opacity* to `50%`.
    2. Select *Simple Fill* and set *Stroke width* to `1.00 mm`.

Now, if you remember past modules, we have to set up and activate the snapping options:

1. Go to *Project* ► *Snapping options…*
2. Press Enable Snapping and select *Advanced Configuration*
3. Check the *green_centroids* and *forest_stands* layers
4. Set *Type* for each layer to *Vertex*
5. Set *Tolerance* for each layer to `10`
6. Set *Units* for each layer to *pixels*
7. Check *Avoid Overlap* for the *forest_stands* layer
8. Press *Topological editing*
9. Choose *Follow Advanced Configuration*
10. Close the pop-up



    With these snapping settings, whenever you are digitizing and get close enough to one of the points in the centroids layer or any vertex of your digitized polygons, a pink square will appear on the point that will be snapped to.

11. Finally, turn off the visibility of all the layers except *forest_stands* and *rautjarvi_georef*. Make sure that the map image has not transparency any more.

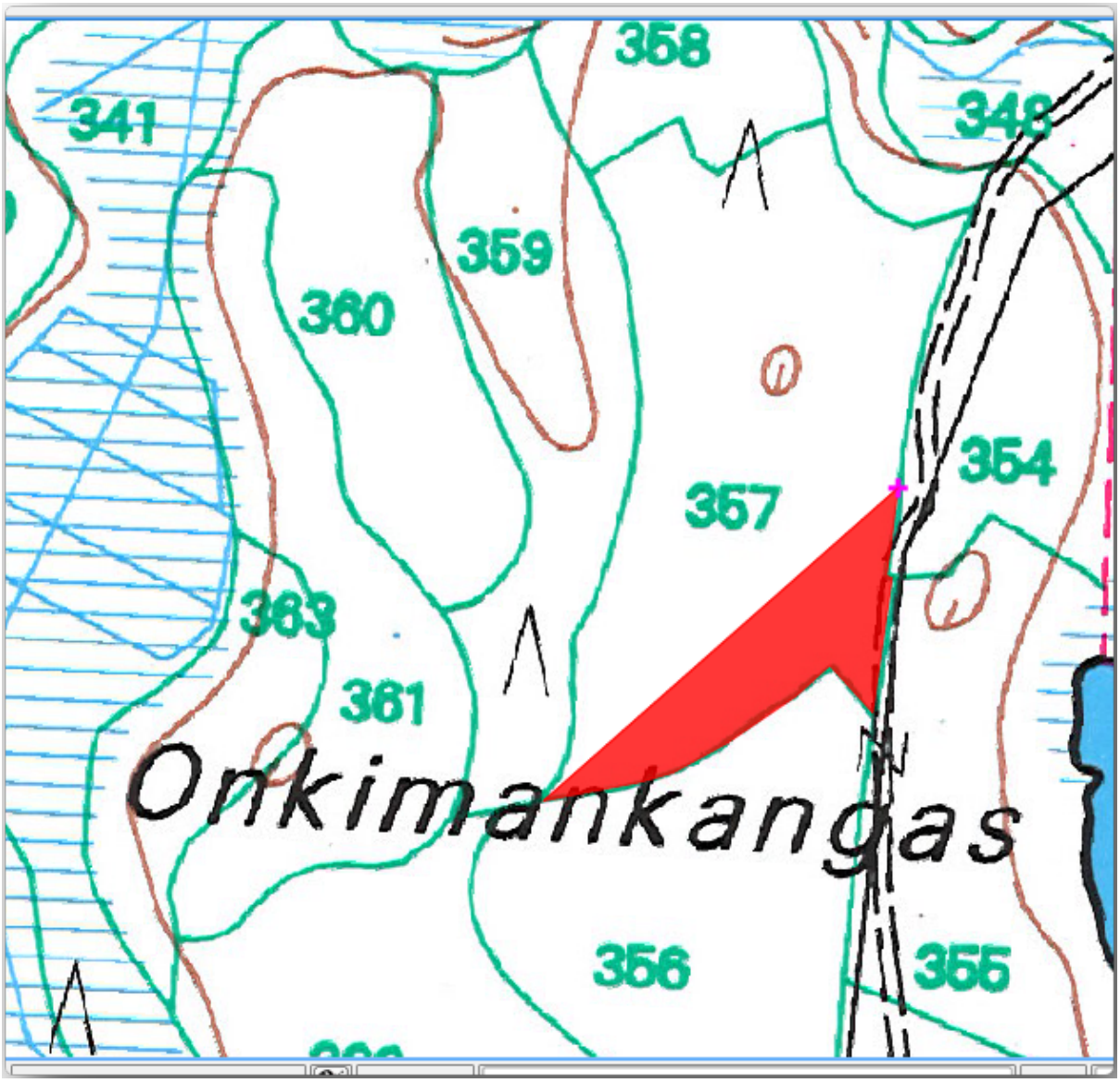
A few important things to note before you start digitizing:

- Don't try to be too accurate with the digitizing of the borders.
- If a border is a straight line, digitize it with just two nodes. In general, digitize using as few nodes as possible.
- Zoom in to close ranges only if you feel that you need to be accurate, for example, at some corners or when you want a polygon to connect with another polygon at a certain node.
- Use the mouse's middle button to zoom in/out and to pan as you digitize.
- Digitize only one polygon at a time.
- After digitizing one polygon, write the forest stand id that you can see from the map.

Now you can start digitizing:

1. Locate the forest stand number `357` in the map window.
2. Select the `forest_stands` layer.

3. Click the Toggle Editing button to enable editing
4. Select Add Polygon Feature tool.
5. Start digitizing the stand `357` by connecting some of the dots. Note the pink crosses indicating the snapping.

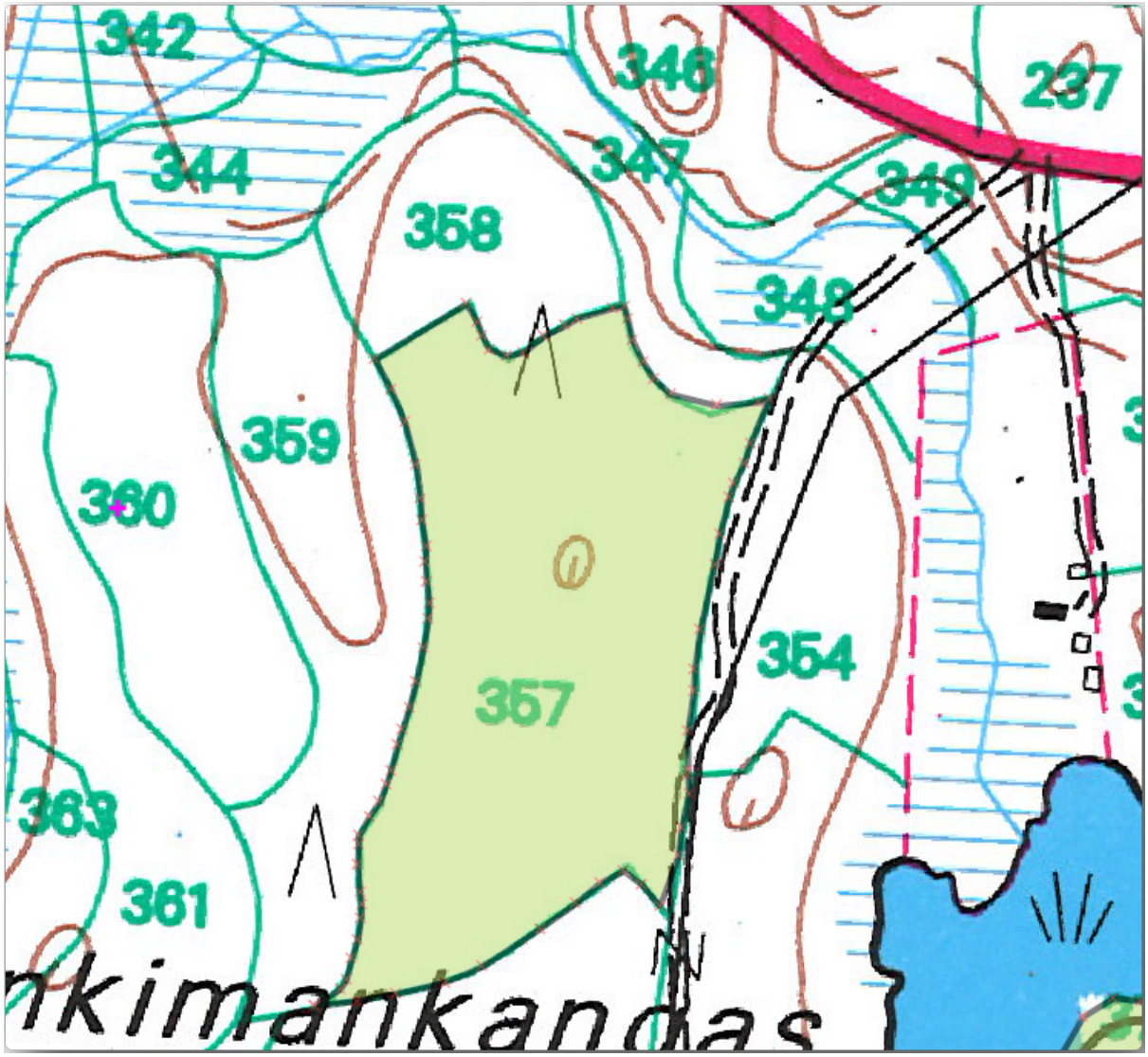


When you are done:

1. Right click to end digitizing that polygon.
2. Enter the forest stand *ID* within the form (in this case `357`).
3. Click *OK*.

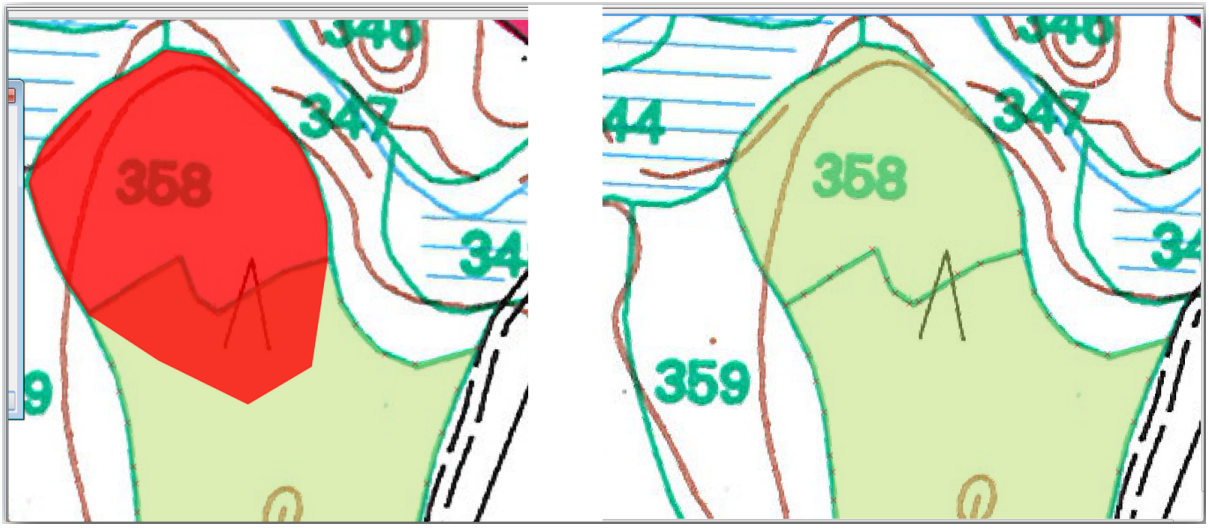
If a form did not appear when you finished digitizing the polygon, go to *Settings* ► *Options* ► *Digitizing* and make sure that the *Suppress attribute form pop-up after feature creation* is not checked.

Your digitized polygon will look like this:



Now for the second polygon, pick up the stand number 358. Make sure that *Avoid Overlap* is checked for the *forest_stands* layer (as shown above). This option ensures polygons do not overlap. So, if you digitize over an existing polygon, the new polygon will be trimmed to meet the border of the existing polygons. You can use this option to automatically obtain a common border.

1. Begin digitizing the stand 358 at one of the common corners with the stand 357.
2. Continue normally until you get to the other common corner for both stands.
3. Finally, digitize a few points inside polygon 357 making sure that the common border is not intersected. See left image below.
4. Right click to finish editing the forest stand 358.
5. Enter the *ID* as `358`.
6. Click *OK*. Your new polygon should have a common border with the stand 357 as you can see in the image below.



The part of the polygon that was overlapping the existing polygon has been automatically trimmed and you are left with a common border - as you intended it to be.

14.3.5 ??? Try Yourself: Finish Digitizing the Forest Stands

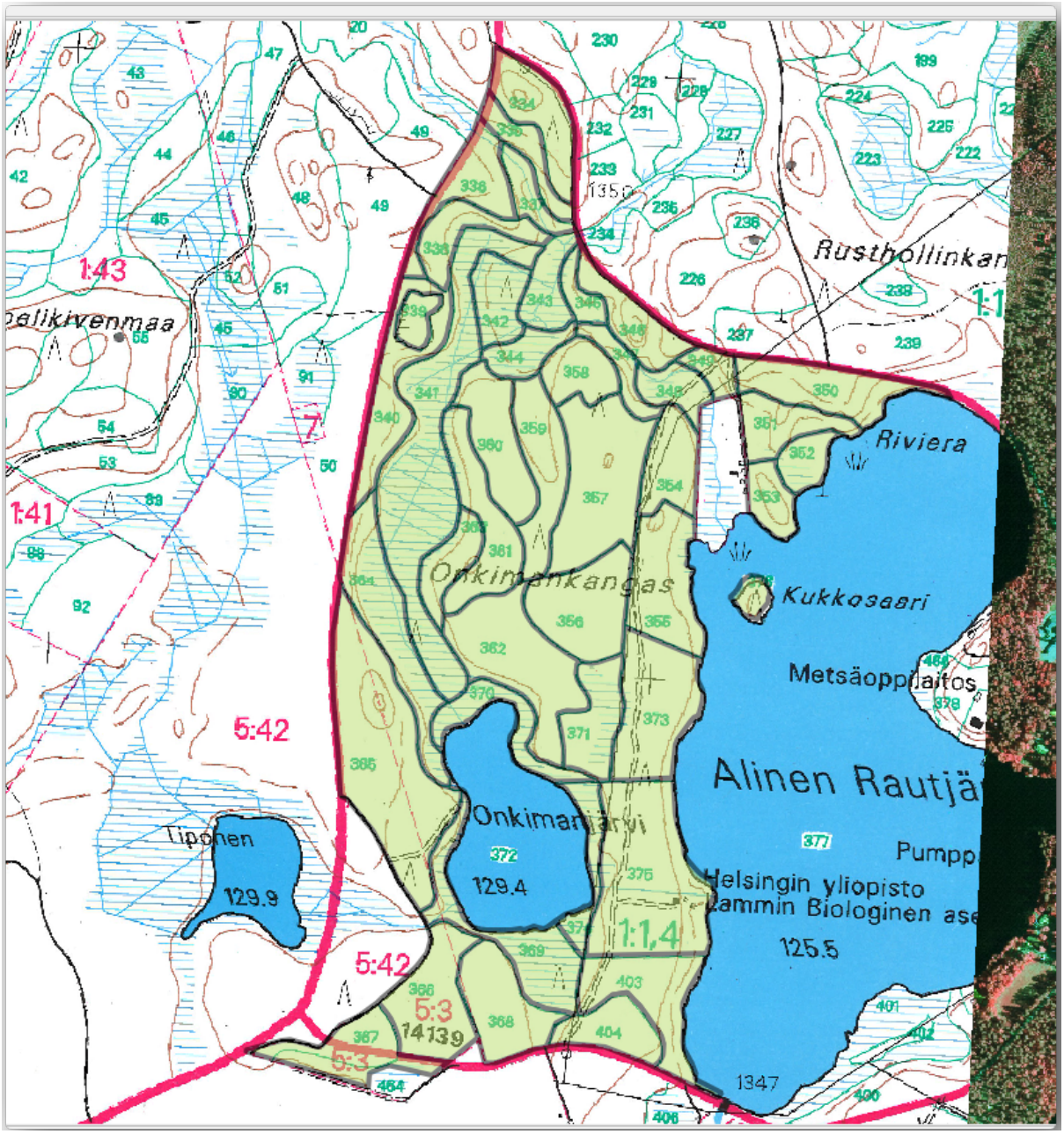
Now you have two forest stands ready. And a good idea on how to proceed. Continue digitizing on your own until you have digitized all the forest stands that are limited by the main road and the lake.

It might look like a lot of work, but you will soon get used to digitizing the forest stands. It should take you about 15 minutes.

During the digitizing you might need to edit or delete nodes, split or merge polygons. You learned about the necessary tools in *Lesson: Feature Topology*, now is probably a good moment to go read about them again.

Remember that having *Enable topological editing* activated, allows you to move nodes common to two polygons so that the common border is edited at the same time for both polygons.

Your result will look like this:

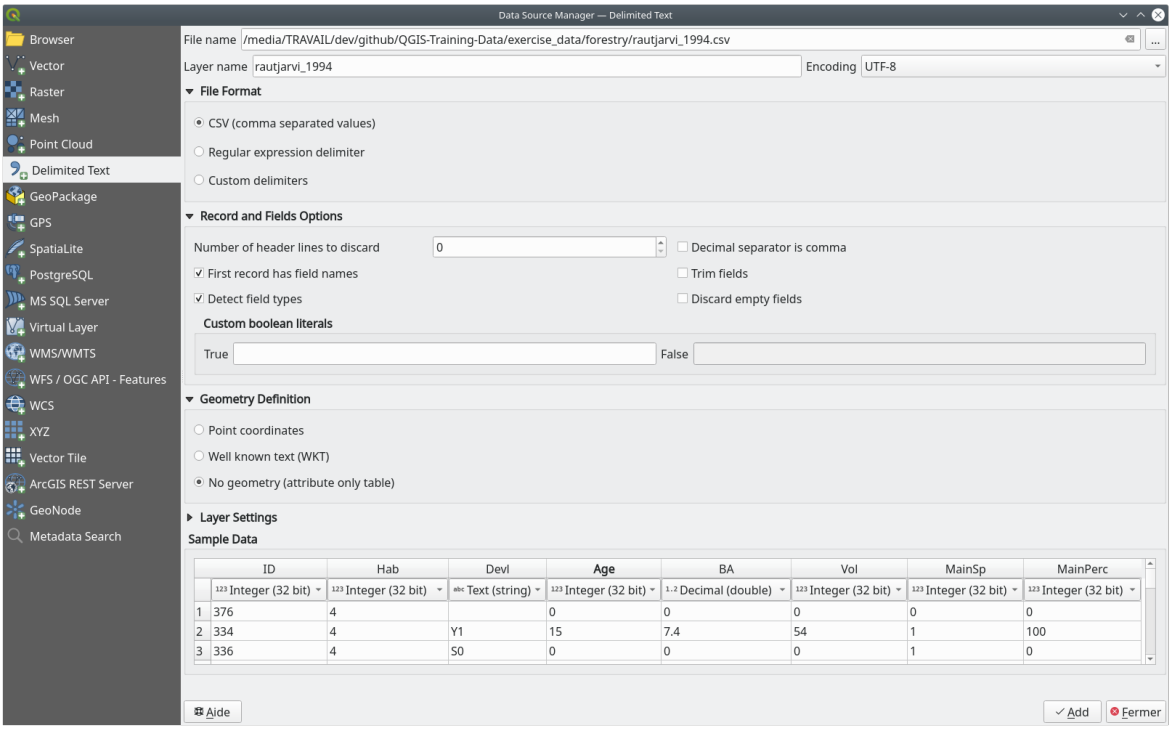


14.3.6 ??? Follow Along: Joining the Forest Stand Data

It is possible that the forest inventory data you have for you map is also written in paper. In that case, you would have to first write that data to a text file or a spreadsheet. For this exercise, the information from the inventory for 1994 (the same inventory as the map) is ready as a comma separated text (csv) file.

1. Open the `rautjarvi_1994.csv` file from the `exercise_data\forestry` directory in a text editor and note that the inventory data file has an attribute called *ID* that has the numbers of the forest stands. Those numbers are the same as the forest stands ids you have entered for your polygons and can be used to link the data from the text file to your vector file. You can see the metadata for this inventory data in the file `rautjarvi_1994_legend.txt` in the same folder.
2. Now add this file into the project:

1. Use the Add Delimited Text Layer tool. This is accessed via *Layer* ► *Add Layer* ► *Add Delimited Text Layer*....
2. Set details in the dialog as follows:



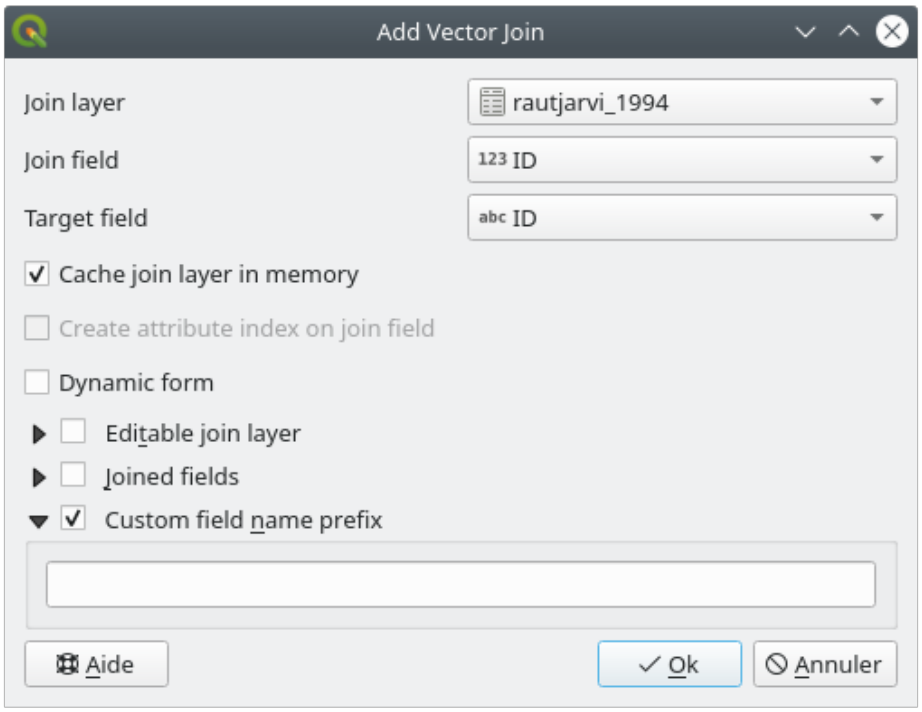
3. Press *Add* to load the formatted `csv` file in the project.

3. To link the data from the `.csv` file with the digitized polygons, create a join between the two layers:
   1. Open the Layer Properties for the `forest_stands` layer.
   2. Go to the *Joins* tab.
   3. Click Add new join on the bottom of the dialog box.
   4. Select *rautjarvi_1994.csv* as the *Join layer*
   5. Set the *Join* field to *ID*
   6. Set the *Target* field to *ID*
   7. Click *OK* two times.

The data from the text file should be now linked to your vector file. To see what has happened, select the `forest_stands` layer and use Open Attribute Table. You can see that all the attributes from the inventory data file are now linked to your digitized vector layer.

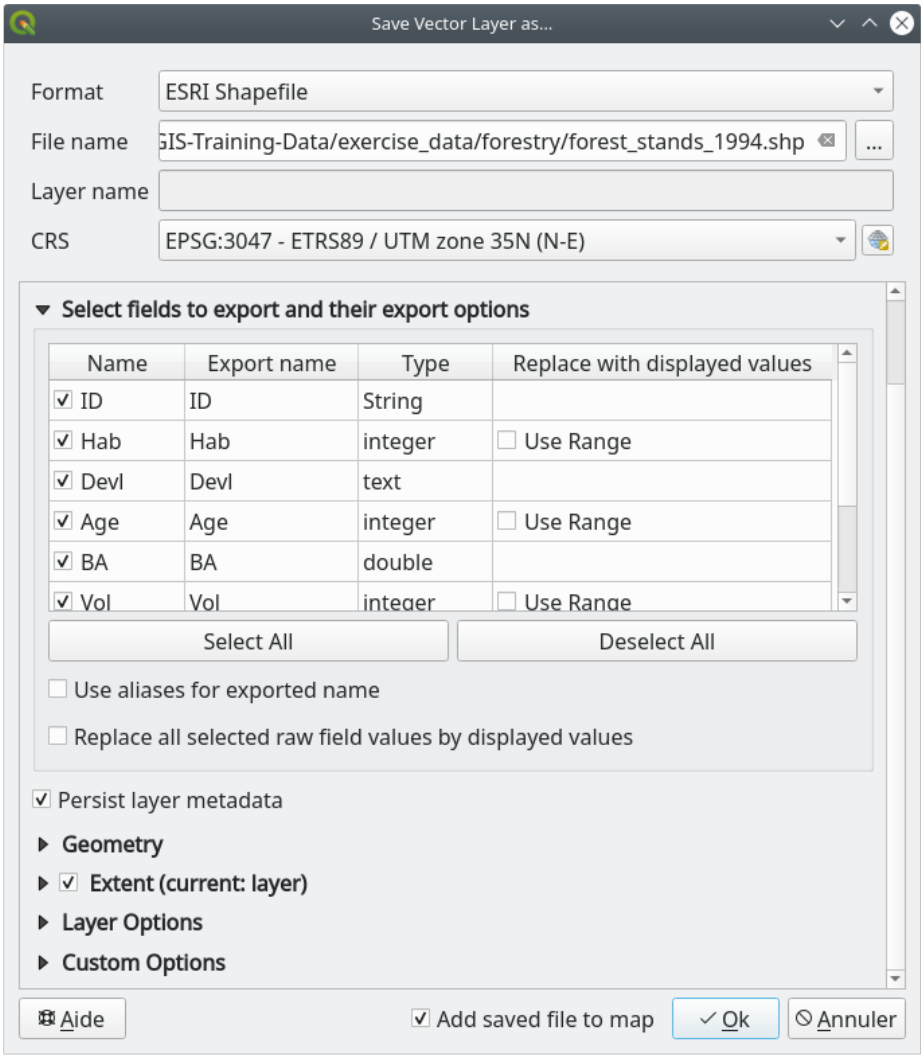
You will see that the field names are prefixed with `rautjarvi_1994_`. To change this:

1. Open the Layer Properties for the `forest_stands` layer.
2. Go to the *Joins* tab.
3. Select *Join Layer rautjarvi_1994*
4. Click the Edit selected join button to enable editing
5. Under *Custom field name prefix* remove the prefix name



The data from the `.csv` file is just linked to your vector file. To make this link permanent, so that the data is actually recorded to the vector file you need to save the `forest_stands` layer as a new vector file. To do this:

1. Right click on `forest_stands` layer
2. Choose *Export* ► *Save Features As...*
3. Set *Format* to *ESRI Shapefile*
4. Set file name to `forest_stands_1994.shp` under the `forestry` folder
5. To include the new file as a layer in the project, check *Add saved file to map*



14.3.7 ??? Try Yourself: Adding Area and Perimeter

To finish gathering the information related to these forest stands, you might calculate the area and the perimeter of the stands. You calculated areas for polygons in *Lesson: Supplementary Exercise*. Go back to that lesson if you need to and calculate the areas for the forest stands. Name the new attribute `Area` and make sure that the values calculated are in hectares. You could also do the same for the perimeter.

Now your `forest_stands_1994` layer is ready and packed with all the available information.

Save your project to keep the current map layers in case you need to come back later to it.

14.3.8 In Conclusion

It has taken a few clicks of the mouse but you now have your old inventory data in digital format and ready for use in QGIS.

14.3.9 What's Next?

You could start doing different analysis with your brand new dataset, but you might be more interested in performing analysis in a dataset more up to date. The topic of the next lesson will be the creation of forest stands using current aerial photos and the addition of some relevant information to your dataset.

14.4 Lesson: Updating Forest Stands

Now that you have digitized the information from the old inventory maps and added the corresponding information to the forest stands, the next step is to create the inventory of the current state of the forest.

You will digitize new forest stands using an aerial photo. As with the previous lesson, you will use an aerial Color Infrared (CIR) photograph. This type of imagery, where the infrared light is recorded instead of the blue light, is widely used to study vegetated areas.

After digitizing the forest stands, you will add information such as new constraints given by conservation regulations.

The goal for this lesson: To digitize a new set of forest stands from CIR aerial photographs and add information from other datasets.

14.4.1 ??? Comparing the Old Forest Stands to Current Aerial Photographs

The National Land Survey of Finland has an open data policy that allows you downloading a variety of geographical data like aerial imagery, traditional topographic maps, DEM, LiDAR data, etc. The service can be accessed in English here. The aerial image used in this exercise has been created from two orthorectified CIR images downloaded from that service (M4134F_21062012 and M4143E_21062012).

1. Open QGIS and set the project's CRS to *ETRS89 / ETRS-TM35FIN* in *Project* ► *Properties…* ► *CRS*
2. Add the CIR image `rautjarvi_aerial.tif` to the project:
    1. Go to the `exercise_data\forestry\` folder using your file manager browser
    2. Drag and drop the file `rautjarvi_aerial.tif` onto your project
3. Save the QGIS project as `digitizing_2012.qgs`

The CIR images are from 2012. You can compare the stands that were created in 1994 with the situation almost 20 years later.

1. Add the `forest_stands_1994.shp` layer created in the previous lesson:
    1. Go to the `exercise_data\forestry\` folder using your file manager browser
    2. Drag and drop the file `forest_stands_1994.shp` onto your project
2. Set the symbology for the layer so that you can see through your polygons:
    1. Right click *forest_stands_1994*
    2. Select *Properties*
    3. Go to the *Symbology* tab
    4. Set *Fill color* to transparent fill

5. Set *Stroke color* to purple
6. Set *Stroke width* to `0.50 mm`

3. Review how the old forest stands follow (or not) what you might visually interpret as an homogeneous forest.
4. Zoom and pan around the area. You probably will notice that some of the old forest stands might be still corresponding with the image but others are not.

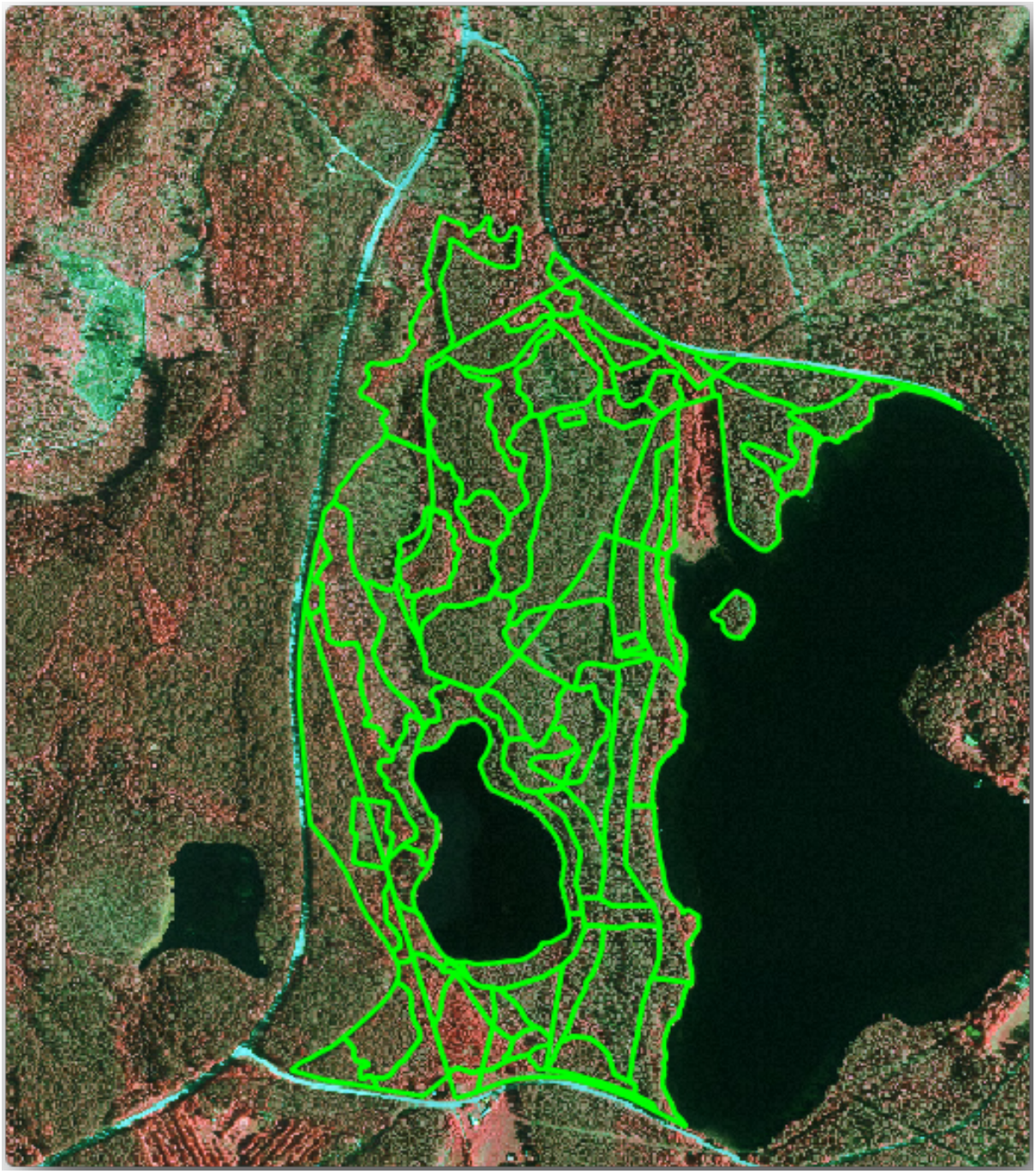
   This is a normal situation, as some 20 years have passed and different forest operations have been carried out (harvesting, thinning…). It is also possible that the forest stands looked homogeneous back in 1992 to the person who digitized them but as time has passed some forest has developed in different ways. It is also possible that that forest inventory priorities back then were different from those of today.

Next, you will create new forest stands for this image without using the old ones. Later you can compare them to see the differences.

14.4.2 ??? Interpreting the CIR Image

Let's digitize the same area that was covered by the old inventory, limited by the roads and the lake. You don't have to digitize the whole area, as in the previous exercise you can start with a vector file that already contains most of the forest stands.

1. Remove the layer *forest_stands_1994*
2. Add the file `exercise_data\forestry\forest_stands_2012.shp` to the project
3. Set the styling of this layer so that the polygons have no fill and the borders are visible
   1. Open *Properties* dialog of the `forest_stands_2012` layer
   2. Go to the *Symbology* tab
   3. Set *Fill color* to transparent fill
   4. Set *Stroke color* to green
   5. Set *Stroke width* to `0.50 mm`



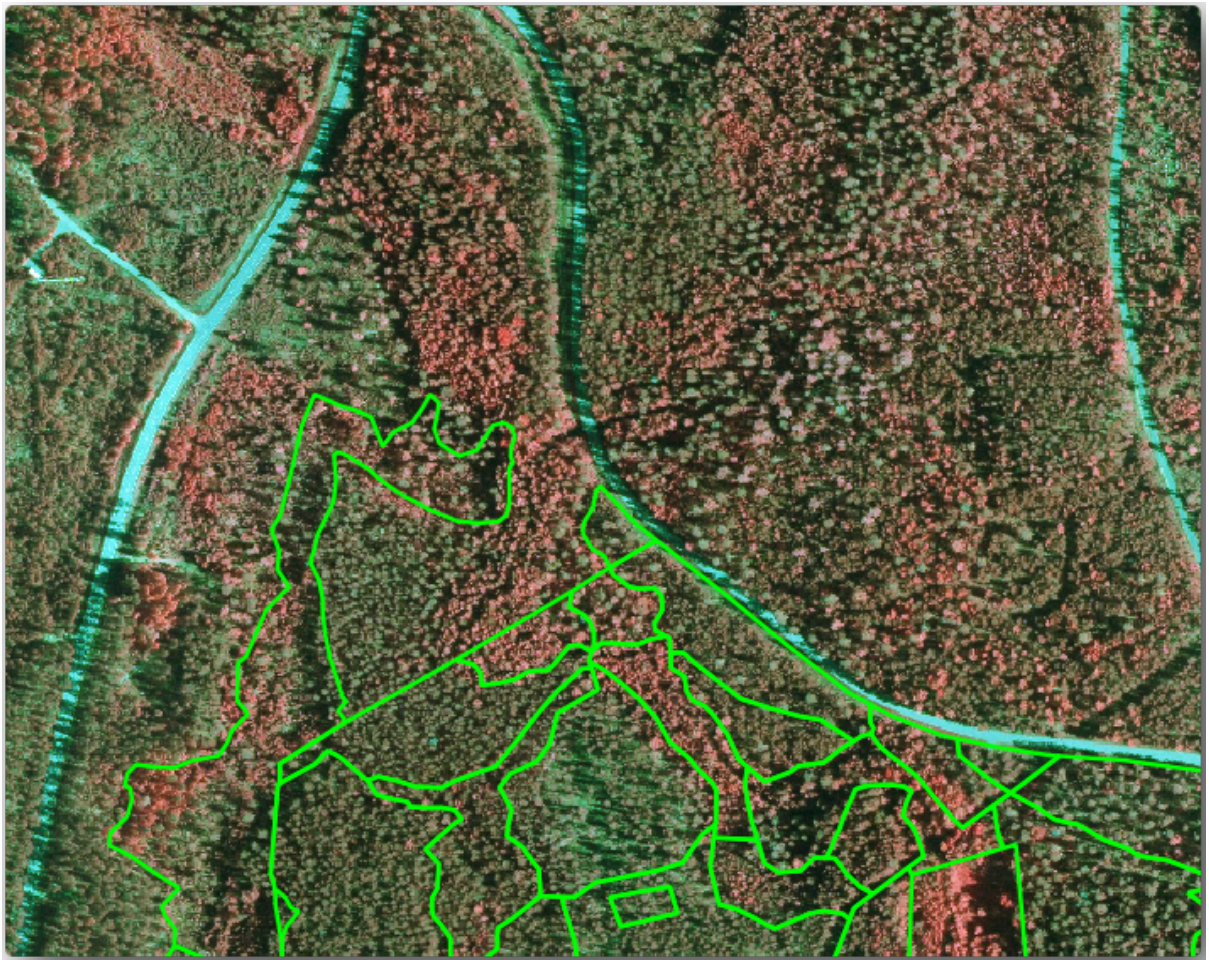
You can see that the northern section of the inventory area is still missing. Your task is to digitize the missing forest stands.

Before you start, spend some time reviewing the forest stands already digitized and the corresponding forest in the image. Try to get an idea about how the stands borders are decided, it helps if you have some forestry knowledge.

Some points to consider:

- Which forests have deciduous species (in Finland these are mostly birch forests) and which ones have conifers (in this area these are pine or spruce)? In CIR images, deciduous species usually show up as a bright red color whereas conifers show as a dark green color.

- How old is the forest? The size of the tree crowns can be identified in the imagery.
- How dense are the different forest stands? A forest stand where a thinning operation has recently been done would show spaces between the tree crowns and should be easy to differentiate from other forest stands around it.
- Blueish areas indicate barren terrain, roads and urban areas, crops that have not started to grow etc.
- Don't use zooms too close to the image when trying to identify forest stands. A scale between 1:3 000 and 1:5 000 should be enough for this imagery. See the image below (1:4000 scale):



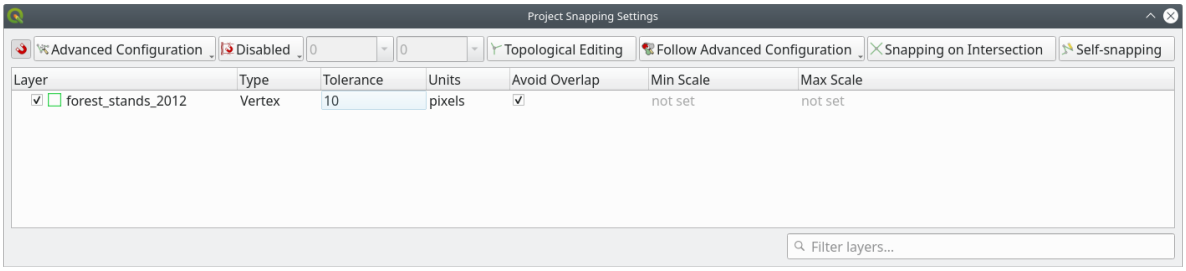
14.4.3 ??? Try Yourself: Digitizing Forest Stands from CIR Imagery

When digitizing the forest stands, you should try to get forest areas that are as homogeneous as possible in terms of tree species, forest age, stand density… Don't be too detailed though, or you will end up making hundreds of small forest stands - and that would not be useful at all. You should try to get stands that are meaningful in the context of forestry, not too small (at least 0.5 ha) but not too big either (no more than 3 ha).

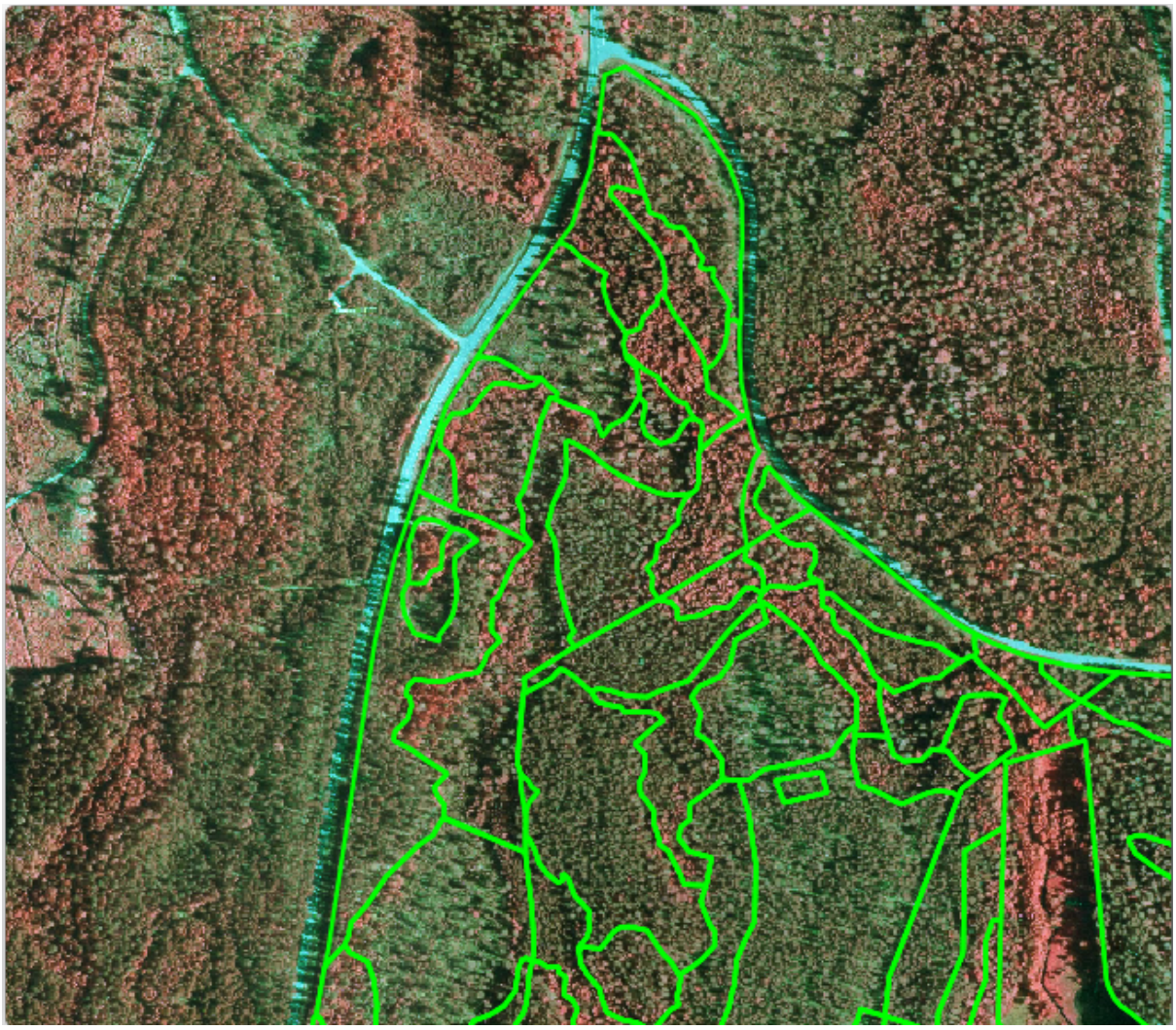
With these points in mind, you can now digitize the missing forest stands.

1. Set up the snapping and topology options:
    1. Go to *Project* ► *Snapping options…*
    2. Press Enable Snapping and select *Advanced Configuration*
    3. Check the *forest_stands_2012* layer
        1. Set *Type* to *Vertex*

2. Set *Tolerance* to `10`
3. Set *Units* to *pixels*
4. Check the box under *Avoid Overlap*
5. Press *Topological editing*
6. Choose *Follow Advanced Configuration*
7. Close the pop-up



2. Select the *forest_stands_2012* layer on the Layers list
3. Click the Toggle Editing button to enable editing
4. Start digitizing using the same techniques as in the previous lesson. The only difference is that you don't have any point layer that you are snapping to. For this area you should get around 14 new forest stands. While digitizing, fill in the `StandID` field with numbers starting at `901`.
5. When you are finished your layer should look something like:



Now you have a new set of polygons showing the different forest stands in 2012 - as interpreted from the CIR images. However, you are missing the forest inventory data. For that you will need to visit the forest and get some sample data that you will use to estimate the forest attributes for each of the forest stands. You will see how to do that in the next lesson.

You can add some extra information about conservation regulations that need to be taken into account for this area.

14.4.4 ??? Follow Along: Updating Forest Stands with Conservation Information

For the area you are working in, there are some conservation regulations that must be taken into account when doing the forest planning:

- Two locations of a protected species of Siberian flying squirrel (Pteromys volans) have been identified. According to the regulation, an area of 15 meters around the spots must be left untouched.
- A riparian forest of special interest that is growing along a stream in the area must be protected. In a visit to the field, it was found that 20 meters to both sides of the stream must be protected.

You have a vector file containing the information about the squirrel locations and another containing the digitized stream running from the North area towards the lake.

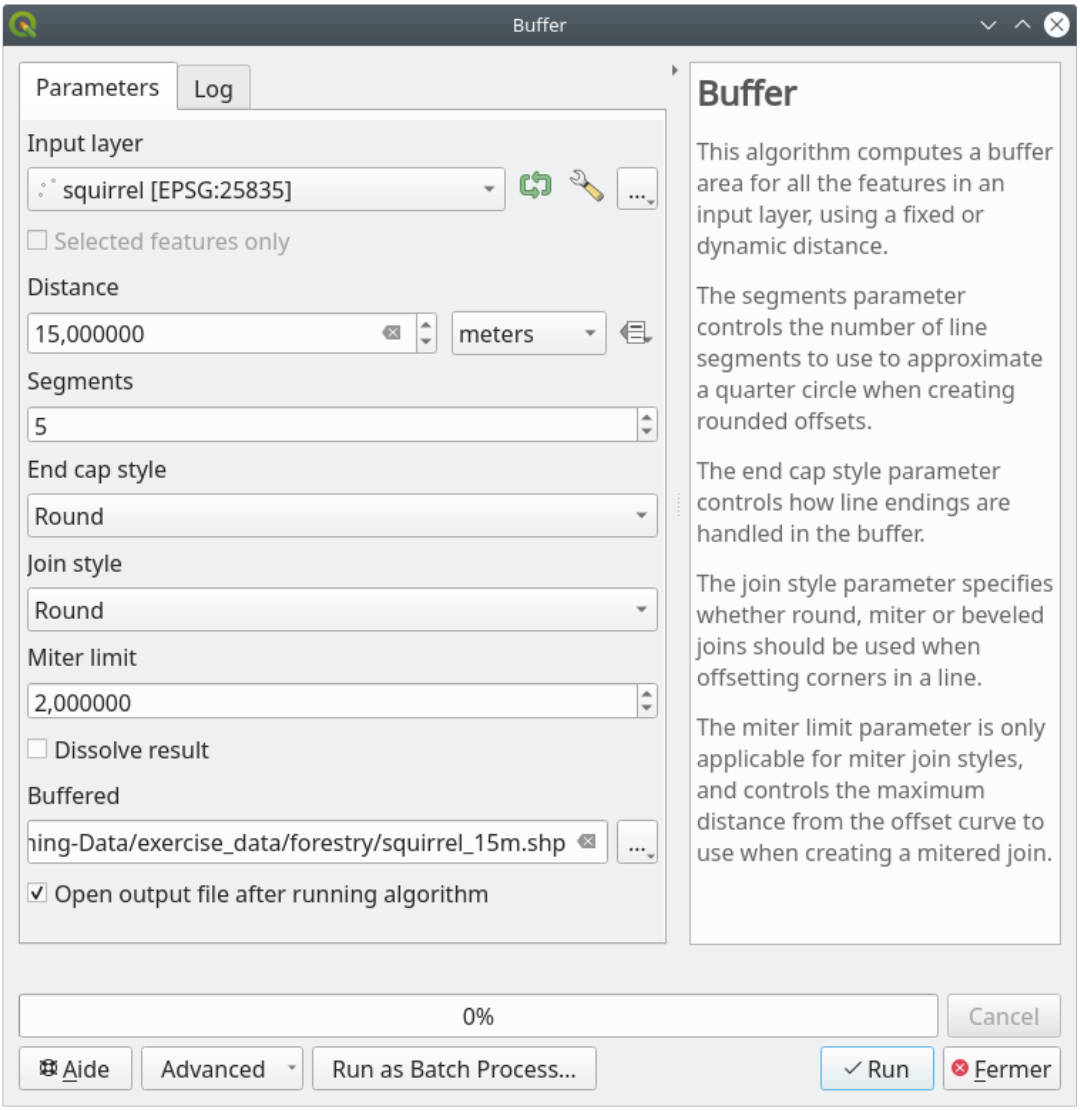
1. From the `exercise_data\forestry\` folder, add the `squirrel.shp` and `stream.shp` files to the project.

2. Use the Open Attribute Table tool to view the `squirrel` layer

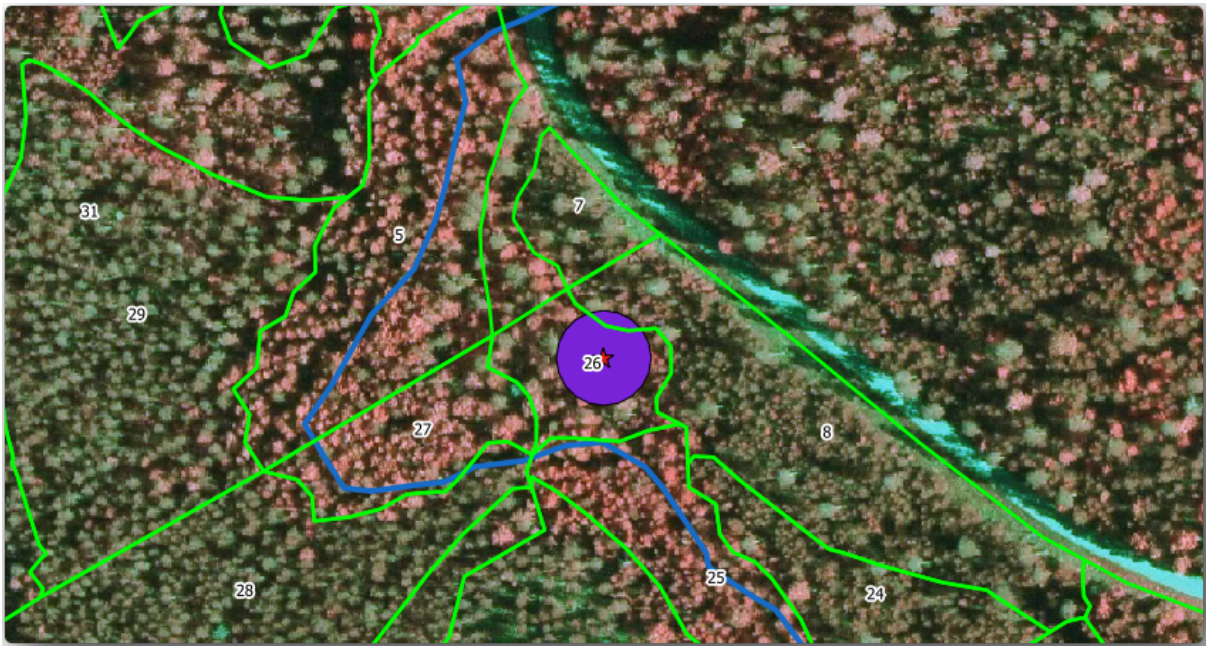
   You can see that there are two locations that are defined as Siberian flying squirrel, and that the area to be protected is indicated by a distance of 15 meters from the locations.

Let's more accurately delimitate that area to protect. We will create a buffer around the point locations, using the protection distance.

1. Open *Vector* ► *Geoprocessing Tools* ► *Buffer*.
2. Set *Input layer* to *squirrel*
3. Set *Distance* to `15 meters`
4. Set *Buffered* to `exercise_data\forestry\squirrel_15m.shp`
5. Check *Open output file afer running algorithm*
6. Click *Run*
7. Once the process is completed, click *Close*



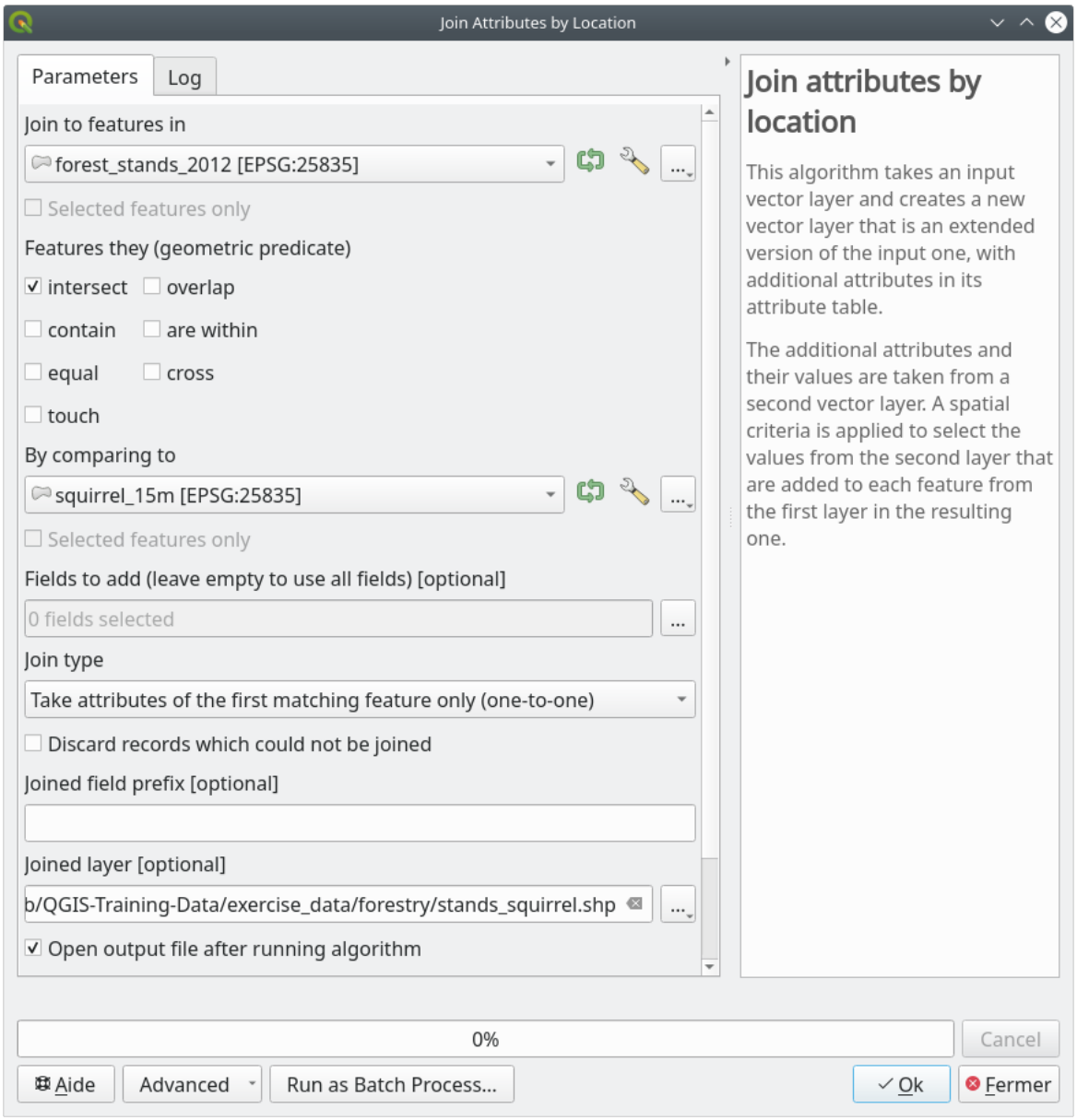
If you zoom in to the location in the northern part of the area, you will notice that the buffer area extends over two neighbouring stands. This means that whenever a forest operation takes place in that stand, the protected location should also be taken into account.



For the protection of the squirrels locations, you are going to add a new attribute (column) to your new forest stands that will contain information about locations that have to be protected. This information will then be available whenever a forest operation is planned, and the field team will be able to mark the area that has to be left untouched before the work starts.

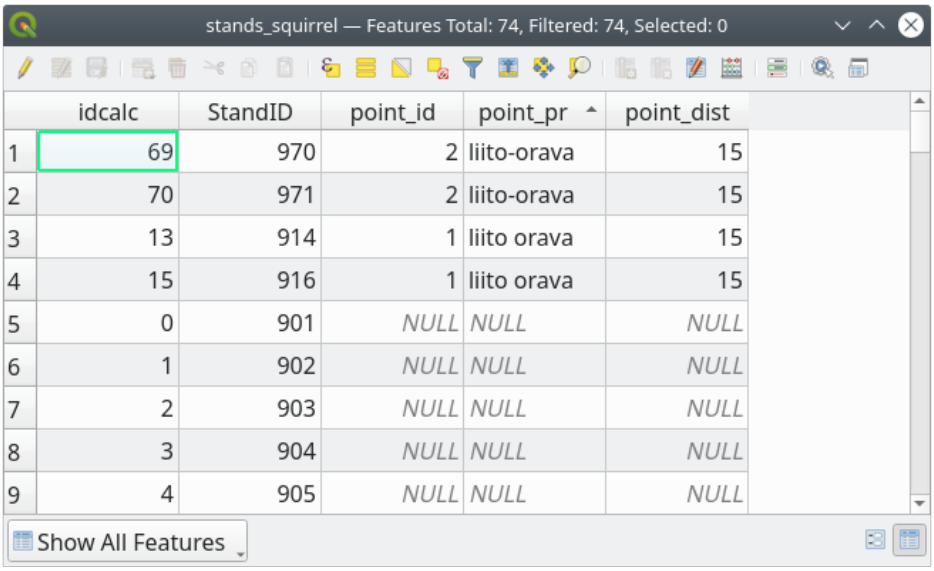
To join the information about the squirrels to your forest stands, you can use the *Join attributes by location* algorithm:

1. Open *Vector* ► *Data Management Tools* ► *Join attributes by location*.
2. Set *Join to features in* to *forest_stands_2012*
3. In *Geometric predicate*, check *intersect*
4. Set *By comparing to* to *squirrel_15m*
5. Set *Join type* as *Take attributes of the first matching feature only (one-to-one)*
6. Leave unchecked *Discard records which could not be joined*
7. Set *Joined layer* to `exercise_data\forestry\stands_squirrel.shp`
8. Check *Open output file afer running algorithm*
9. Click *Run*
10. Once the process is completed, you can *Close* the dialog.



Now you have a new forest stands layer, `stands_squirrel.shp` showing the protection information for the Siberian flying squirrel.

1. Open the attribute table of the `stands_squirrel` layer
2. Sort the table by clicking on *point_pr* field in the table header.



	idcalc	StandID	point_id	point_pr	point_dist
1	69	970	2	liito-orava	15
2	70	971	2	liito-orava	15
3	13	914	1	liito orava	15
4	15	916	1	liito orava	15
5	0	901	NULL	NULL	NULL
6	1	902	NULL	NULL	NULL
7	2	903	NULL	NULL	NULL
8	3	904	NULL	NULL	NULL
9	4	905	NULL	NULL	NULL

You can see that there are some forest stands that have the information about the protection locations. The information in the forest stands data will indicate to the forest manager that there are protection considerations to be taken into account. Then he or she can get the location from the *squirrel* dataset, and visit the area to mark the corresponding buffer around the location so that the operators in the field can avoid disturbing the squirrels environment.

14.4.5 ??? Try Yourself: Updating Forest Stands with Distance to the Stream

Following the same approach as for the protected squirrel locations you can now update your forest stands with protection information related to the stream. A few points:

- Remember the buffer is `20` meters around the stream
- You want to have all the protection information in the same vector file, so use `stands_squirrel.shp` as the base layer
- Name your output as `forest_stands_2012_protect.shp`

Once the process is completed, open the attribute table of the output layer and confirm that you have all the protection information for the riparian forest stands associated with the stream.

When you are happy with the results, save your QGIS project.

14.4.6 In Conclusion

You have seen how to interpret CIR images to digitize forest stands. Of course it would take some practice to make more accurate stands and usually using other information like soil maps would give better results, but you know now the basis for this type of task. And adding information from other datasets resulted to be quite a trivial task.

14.4.7 What's Next?

The forest stands you digitized will be used for planning forestry operations in the future, but you still need to get more information about the forest. In the next lesson, you will see how to plan a set of sampling plots to inventory the forest area you just digitized, and get the overall estimate of forest parameters.

14.5 Lesson: Systematic Sampling Design

Já digitalizou um conjunto de polígonos que representam povoamentos florestais, mas ainda não tem informações sobre a floresta. Para esse efeito, pode projetar uma pesquisa para o inventário da área de floresta e, em seguida, estimar os seus parâmetros. Nesta lição, vai criar um conjunto de pontos de amostragem sistemática.

Quando começamos a planear um inventário florestal é importante definir claramente os objetivos, os tipos de parcelas que vão ser utilizados, e os dados que serão recolhidos para atingir os objectivos. Para cada caso individual, irá depender do tipo de gestão da floresta proposta; e deve ser cuidadosamente planeado por alguém com conhecimento na área florestal. Nesta lição, vai implementar um inventário teórico baseado numa amostragem sistemática de parcelas.

O objetivo desta lição: Criar um esquema de parcelas de amostragem sistemática com o intuito de fazer o levantamento da área florestal.

14.5.1 Inventariar a Floresta

Existem diversos métodos de inventariar a floresta, cada um com diferentes condições e propósitos. Por exemplo, de uma forma muito precisa para inventariar uma floresta (se considerar apenas espécies de árvores) seria visitar a floresta e fazer uma lista de todas as árvores e as suas características. Como pode imaginar este não é geralmente aplicável excepto para áreas pequenas ou situações especiais.

The most common way to find out about a forest is by sampling it, that is, taking measurements in different locations at the forest and generalizing that information to the whole forest. These measurements are often made in *sample plots* that are smaller forest areas that can be easily measured. The sample plots can be of any size (for ex. 50 m2, 0.5 ha) and form (for ex. circular, rectangular, variable size), and can be located in the forest in different ways (for ex. randomly, systematically, along lines). The size, form and location of the sample plots are usually decided following statistical, economical and practical considerations. If you have no forestry knowledge, you might be interested in reading this Wikipedia article.

14.5.2 ??? Follow Along: Implementing a Systematic Sampling Plot Design

Para a floresta que está a trabalhar, o gestor decidiu que o design da amostragem sistemática como o mais apropriado para esta floresta decidindo que a distância entre as parcelas seria fixa, com 80 metros entre as linhas de amostragem visto que iria produzir resultados confiáveis (para este caso, + - 5% de erro médio a uma probabilidade de 68%). Como o método mais eficaz para este inventário decidiu-se que as parcelas teriam tamanho variável, para os povoamentos em crescimento e maduros, para povoamentos jovens seriam usadas parcelas de raio fixo com 4 metros de raio.

Na prática, necessita de simplesmente representar as parcelas de amostra como pontos que serão usados pelas equipas de campo posteriormente:

1. Open your `digitizing_2012.qgs` project from the previous lesson.
2. Remove all the layers except for *forest_stands_2012*.
3. Save your project now as `forest_inventory.qgs`

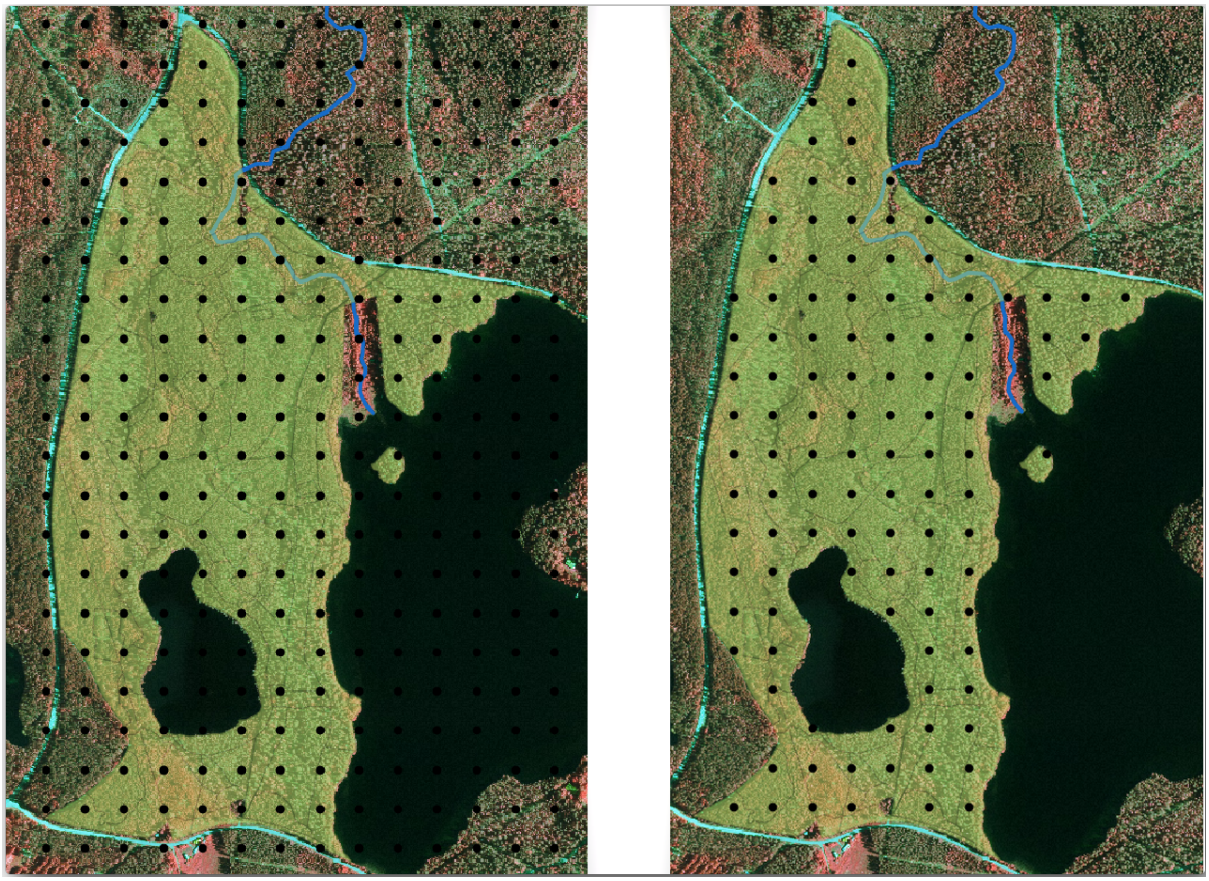
Agora necessita de criar uma grelha retangular de pontos separados por 80 metros um do outro:

1. Open *Vector* ► *Research Tools* *Regular points*.

2. Press the drop-down button next to the *Input extent* field and from the *Calculate from Layer* menu, select *forest_stands_2012*.
3. In the *Point spacing/count* settings, enter `80 meters`.
4. Check the *Use point spacing* box to indicate that the value represents the distance between the points.
5. Under *Regular points*, save the output as `systematic_plots.shp` in the `forestry\sampling\` folder.
6. Check *Open output file after running algorithm*.
7. Press *Run*.

Nota: The suggested *Regular points* creates the systematic points starting in the upper-left corner of the extent of the selected polygon layer. If you want to add some randomness to this regular points, you could use a randomly calculated number between 0 and 80 (80 is the distance between our points), and then write it as the *Initial inset from corner (LH side)* parameter in the tool's dialog.

Percebe que a ferramenta usa toda a extensão da sua camada de povoamentos para gerar uma grelha retangular de pontos. Mas interessa-nos apenas esses pontos que estão, na verdade, dentro da sua área de floresta (veja as imagens abaixo):



1. From the Processing toolbox, open *GDAL* ► *Vector geoprocessing* ► *Clip vector by mask layer*.
2. Select *systematic_plots* as the *Input layer*.
3. Set *forest_stands_2012* as the *Mask layer*.
4. Save the *Clipped (mask)* result as `systematic_plots_clip.shp` in the `forestry\sampling\` folder.
5. Check *Open output file after running algorithm*.

6. Press *Run*.

Tem agora os pontos que as equipas de campo vão usar para navegar até os locais das parcelas planeadas. Pode ainda preparar estes pontos de modo a que eles sejam mais úteis para o trabalho de campo. Pelo menos terá que adicionar nomes significativos para os pontos e exportá-los para um formato que pode ser usado no seu aparelho de GPS.

Let's start with the naming of the sample plots. If you check the *Attribute table* for the plots inside the forest area, you can see that you have the default *id* field automatically generated by the *Regular points* tool. Label the points to see them in the map and consider if you could use those numbers as part of your sample plot naming:

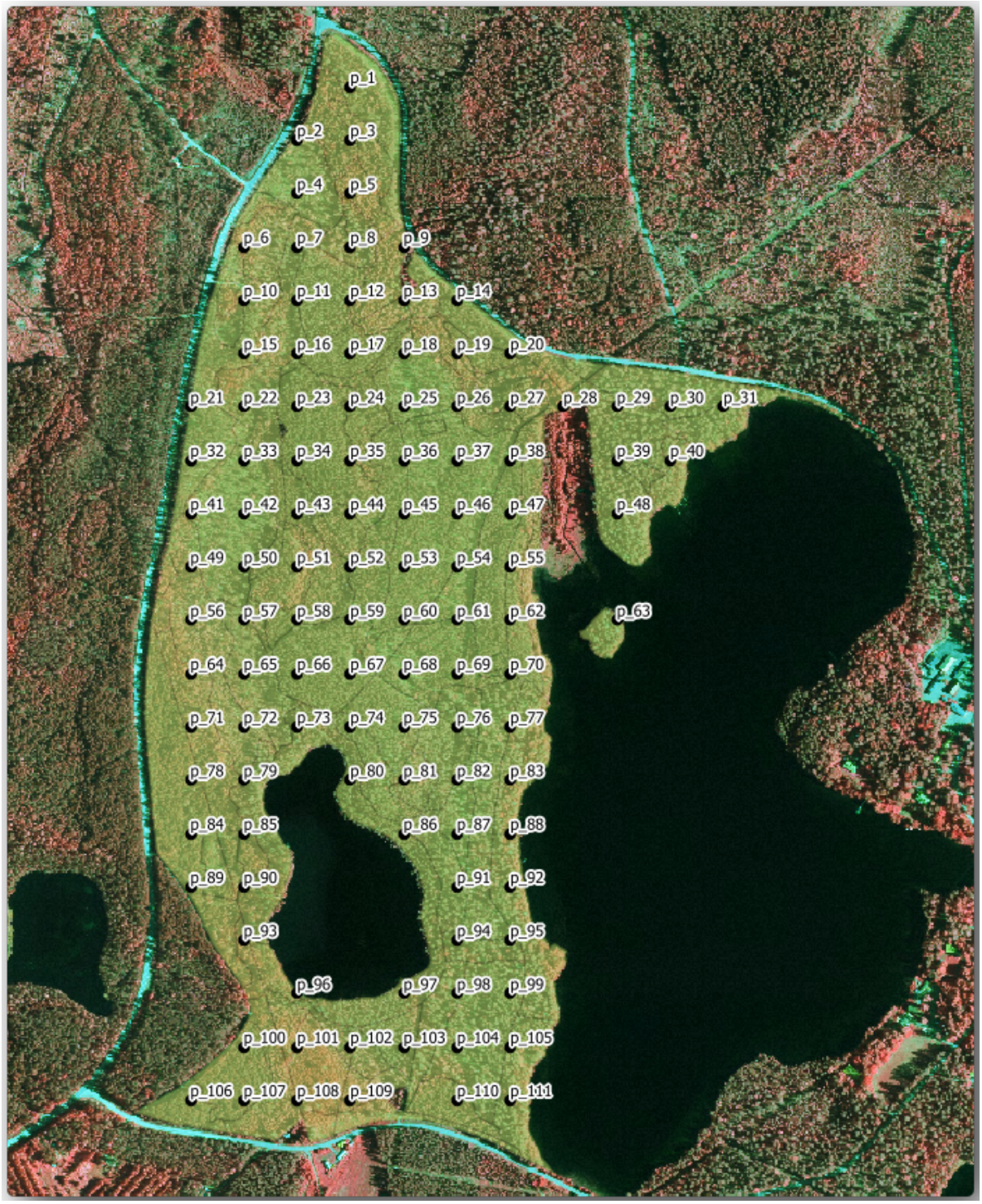
1. Open the *Layer Properties* ► *Labels* for the `systematic_plots_clip` layer.
2. Turn the top menu into *Single Labels*.
3. For the *Value* entry, select the field `id`.
4. Go to the *Buffer* tab, check the *Draw text buffer* and set the buffer *Size* to `1`.
5. Click *OK*.

Agora olhe para os rótulos do seu mapa. Pode verificar que os pontos foram criadas e numeradas primeiro de Ocidente para o Oriente e, em seguida, do Norte para o Sul. Se olhar para a tabela de atributos de novo, vai notar que a ordem na tabela também segue esse padrão. Se pretender nomear por outras razões as parcelas de uma forma diferente, nomeando-os de uma forma Leste-Oeste / Norte-Sul segue uma ordem lógica e é uma boa opção.

Nevertheless, the number values in the `id` field are not so good. It would be better if the naming would be something like `p_1, p_2...`. You can create a new column for the `systematic_plots_clip` layer:

1. Go to the *Attribute table* for `systematic_plots_clip`.
2. Enable the edit mode.
3. Open the Field calculator:
    1. Check *Create a new field*
    2. Enter `Plot_id` as *Output field name*
    3. Set the *Output field type* to *Text (string)*.
    4. In the *Expression* field, write, copy or construct this formula `concat('P_', @row_number )`. Remember that you can also double click on the elements inside the *Function list*. The `concat` function can be found under *String* and `@row_number` is under the *Variables* group.
4. Click *OK*.
5. Desactive o modo de edição e guarde as alterações.

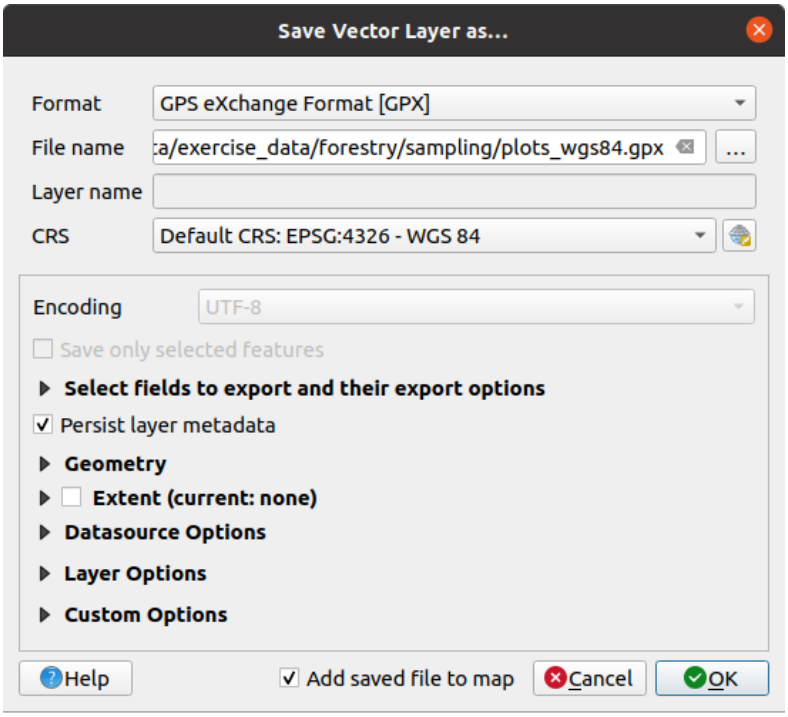
Now you have a new column with plot names that are meaningful to you. For the `systematic_plots_clip` layer, change the field used for labeling to your new `Plot_id` field.


p_1
p_2 p_3
p_4 p_5
p_6 p_7 p_8 p_9
p_10 p_11 p_12 p_13 p_14
p_15 p_16 p_17 p_18 p_19 p_20
p_21 p_22 p_23 p_24 p_25 p_26 p_27 p_28 p_29 p_30 p_31
p_32 p_33 p_34 p_35 p_36 p_37 p_38 p_39 p_40
p_41 p_42 p_43 p_44 p_45 p_46 p_47 p_48
p_49 p_50 p_51 p_52 p_53 p_54 p_55
p_56 p_57 p_58 p_59 p_60 p_61 p_62 p_63
p_64 p_65 p_66 p_67 p_68 p_69 p_70
p_71 p_72 p_73 p_74 p_75 p_76 p_77
p_78 p_79 p_80 p_81 p_82 p_83
p_84 p_85 p_86 p_87 p_88
p_89 p_90 p_91 p_92
p_93 p_94 p_95
p_96 p_97 p_98 p_99
p_100 p_101 p_102 p_103 p_104 p_105
p_106 p_107 p_108 p_109 p_110 p_111

14.5.3 ??? Follow Along: Exporting Sample Plots as GPX format

The field teams will be probably using a GPS device to locate the sample plots you planned. The next step is to export the points you created to a format that your GPS can read. QGIS allows you to save your point and line vector data in GPS eXchange Format (GPX), which is an standard GPS data format that can be read by most of the specialized software. You need to be careful with selecting the CRS when you save your data:

1. Right-click `systematic_plots_clip` layer and select *Export* ► *Save features as...*.



2. Em *Formato* seleccione *GPS eXchange Format [GPX]*.
3. Save the output *File name* as `plots_wgs84.gpx` in the `forestry\sampling\` folder.
4. Em *SRC* selecionar *Seleccionar SRC*.
5. Browse for *EPSG:4326 - WGS 84*.

> **Nota:** The GPX format accepts only this CRS, if you select a different one, QGIS will give no error but you will get an empty file.

6. Click *OK*.
7. In the dialog that opens, select only the `waypoints` layer (the rest of the layers are empty).

The inventory sample plots are now in a standard format that can be managed by most of the GPS software. The field teams can now upload the locations of the sample plots to their devices. That would be done by using the specific devices own software and the `plots_wgs84.gpx` file you just saved. Other option would be to use the *GPS Tools* plugin but it would most likely involve setting the tool to work with your specific GPS device. If you are working with your own data and want to see how the tool works you can find out information about it in the section working_gps in the **QGIS User Manual**.

Guarde o seu projecto QGIS

14.5.4 In Conclusion

Acabou de ver como poderá criar facilmente um projeto de amostragem sistemática para ser usado num inventário florestal. A criação de outros tipos de planos de amostragem envolve o uso de diferentes ferramentas dentro do QGIS, folhas de cálculo ou scripts para calcular as coordenadas das parcelas, mas a ideia geral permanece a mesma.

14.5.5 What's Next?

Na próxima lição verá como usar os recursos do atlas no QGIS para gerar automaticamente mapas detalhados que as equipas de campo usam para navegar para as parcelas que lhes foram atribuídas.

14.6 Lesson: Creating Detailed Maps with the Atlas Tool

The systematic sampling design is ready and the field teams have loaded the GPS coordinates in their navigation devices. They also have a field data form where they will collect the information measured at every sample plot. To easier find their way to every sample plot, they have requested a number of detail maps where some ground information can be clearly seen along with a smaller subset of sample plots and some information about the map area. You can use the Atlas tool to automatically generate a number of maps with a common format.

The goal for this lesson: Learn to use the Atlas tool in QGIS to generate detailed printable maps to assist in the field inventory work.

14.6.1 ??? Follow Along: Preparing the Print Layout

Before we can automate the detailed maps of the forest area and our sampling plots, we need to create a map template with all the elements we consider useful for the field work. Of course the most important will be a properly styled but, as you have seen before, you will also need to add lots of other elements that complete the printed map.

1. Open the QGIS project from the previous lesson `forest_inventory.qgs`. You should have at least the following layers:
   - *forest_stands_2012* (with a 50% transparency, green fill and darker green border lines)
   - *systematic_plots_clip*
   - *rautjarvi_aerial*
2. Save the project with a new name, `map_creation.qgs`.

To create a printable map, remember that you use the *Layout Manager*:

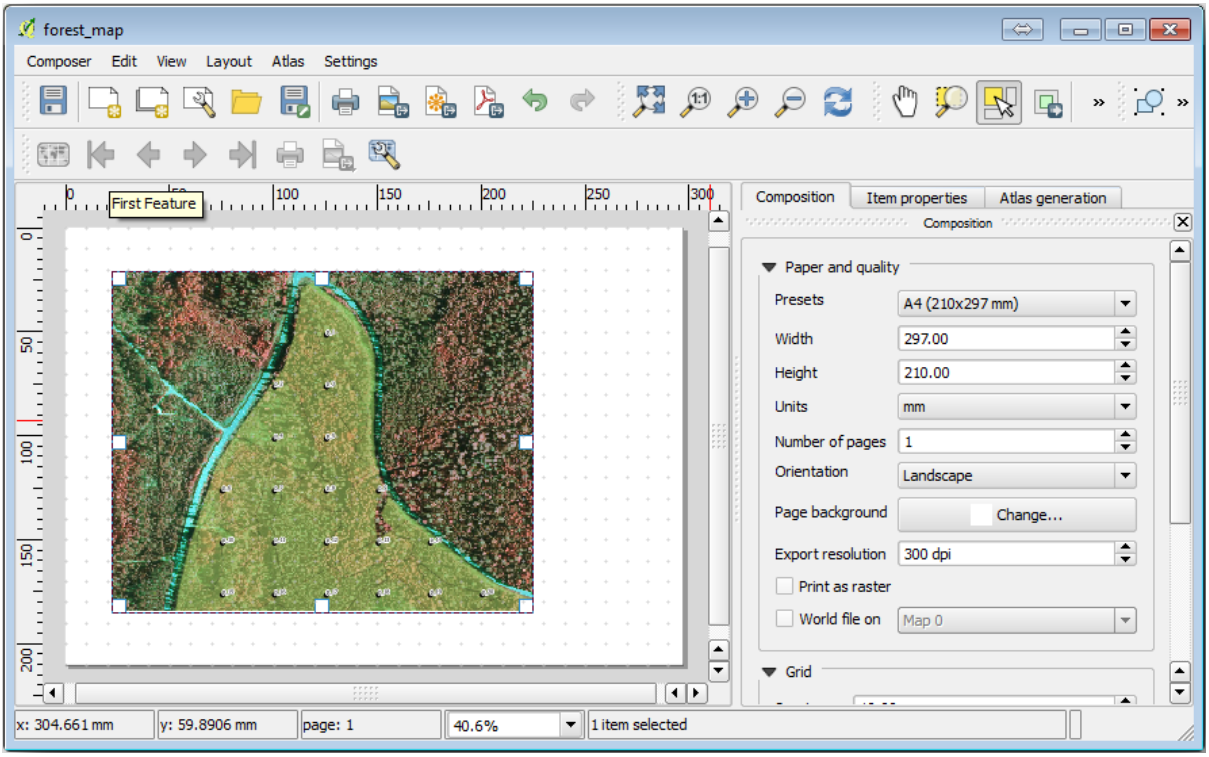
1. Open *Project* ► *Layout Manager…*.
2. In the *Layout manager* dialog:
   1. Under *New from template*, press the *Create…* button next to the *Empty layout* entry
   2. Name your print layout `forest_map`.
   3. Press *OK*. A new print layout is created and opened, showing a blank paper.
3. In the print Layout window, ensure properties are set for an A4 paper:
   1. Right-click over the paper and select *Page properties*. The *Page properties* panel is open on the right side of the layout.
   2. Check *Size* is *A4*.
   3. Check *Orientation* is *Landscape*.
4. Enable the *Layout* tab next to the *Page properties* panel and set *Export resolution* to `300 dpi`.

Composing a map is easier if you make use of the canvas grid to position the different elements. Review the settings for the layout grid:

1. In the *Layout* tab expand the *Guides and Grid* region.
2. Check that *Grid spacing* is set to `10 mm` and that *Snap tolerance* is set to `5 px`.

You need to activate the use of the grid:

1. Open the *View* menu.
2. Check *Show grid*.
3. Check *Snap to grid*.
4. Notice that options for using *Guides* are checked by default, which allows you to see guiding lines when you are moving elements in the layout.
5. Now you can start to add elements to your layout. Add first a map element so you can review how the map canvas looks as you will be making changes in the layers symbology:
    1. Click on the Add Map button.
    2. Click and drag a box on the canvas so that the map occupies most of it.



Notice how the mouse cursor snaps to the canvas grid. Use this function when you add other elements. If you want to have more accuracy, change the grid *Spacing* setting. If for some reason you don't want to snap to the grid at some point, you can always check or uncheck it in the *View* menu.

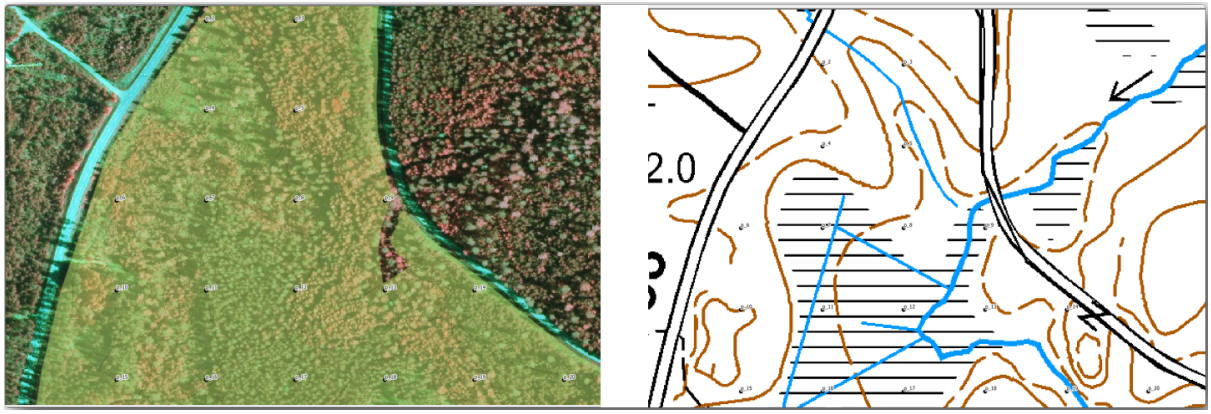
14.6.2 ??? Follow Along: Adding Background Map

Leave the layout open but go back to the map. Let's add some background data and create some styling so that the map content is as clear as possible.

1. Add the background raster `basic_map.tif` that you can find in the `exercise_data\forestry\` folder.
2. If prompted select the *ETRS89 / ETRS-TM35FIN* CRS for the raster.

   As you can see the background map is already styled. This type of ready to use cartography raster is very common. It is created from vector data, styled in a standard format and stored as a raster so that you don't have to bother styling several vector layers and worrying about getting a good result.
3. Now zoom to your sample plots, so that you can see only about four or five lines of plots.

The current styling of the sample plots is not the best:



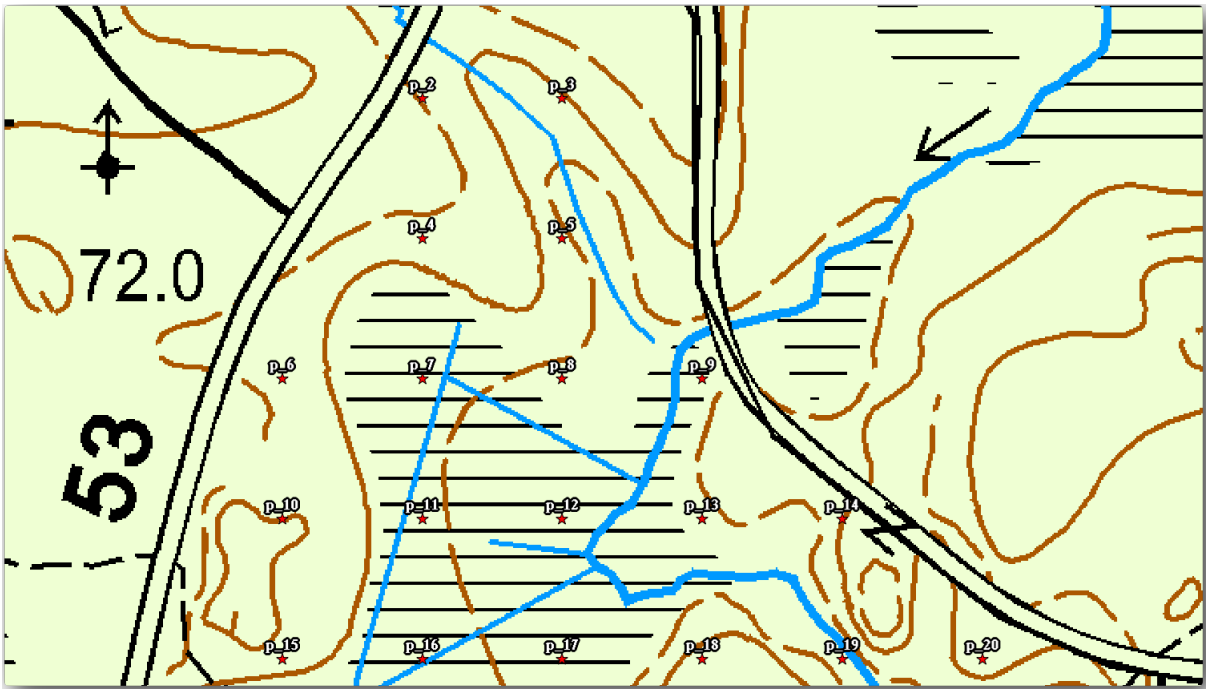
While during the last exercises, the white buffer was OK on top of the aerial image, now that the background image is mostly white you barely can see the labels. But how does it look in the print layout? Let's check it:

1. Go to the print layout window.
2. Use the Select/Move item button to select the map element in the layout.
3. Go to the *Item properties* tab.
4. Click on *Set map extent to match main canvas extent*.
5. If you need to refresh the element, click on *Update map preview*.

Obviously this is not good enough; you want to make the plot numbers as clearly visible as possible for the field teams.

14.6.3 ??? Try Yourself: Changing the Symbology of the Layers

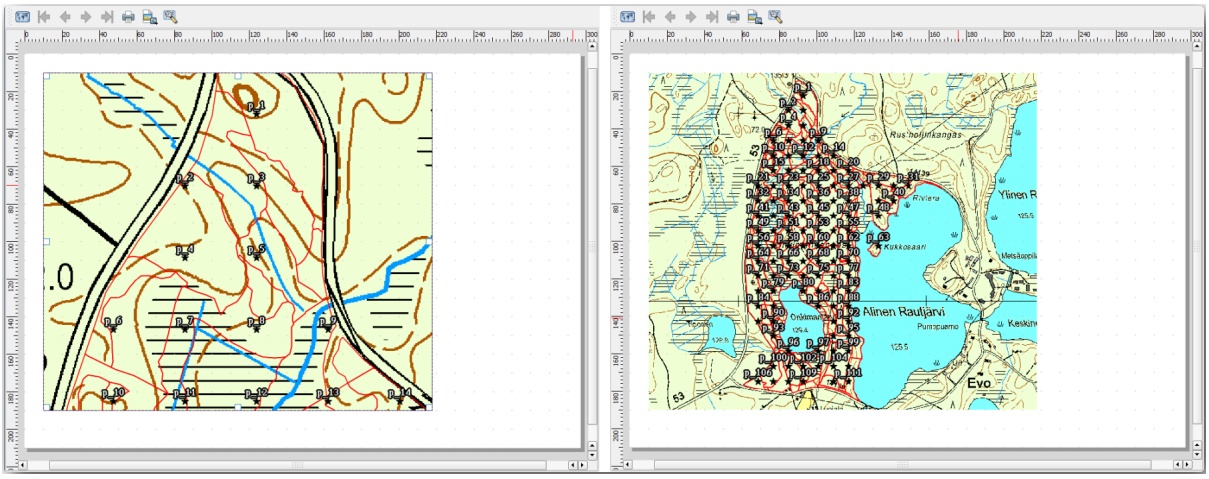
You have been working in *Module: Creating and Exploring a Basic Map* with symbology and in *Module: Classifying Vector Data* with labeling. Go back to those modules if you need to refresh about some of the available options and tools. Your goal is to get the plots locations and their name to be as clearly visible as possible but always allowing to see the background map elements. You can take some guidance from this image:



You will use later the the green styling of the `forest_stands_2012` layer. In order to keep it, and have a visualization of it that shows only the stand borders:

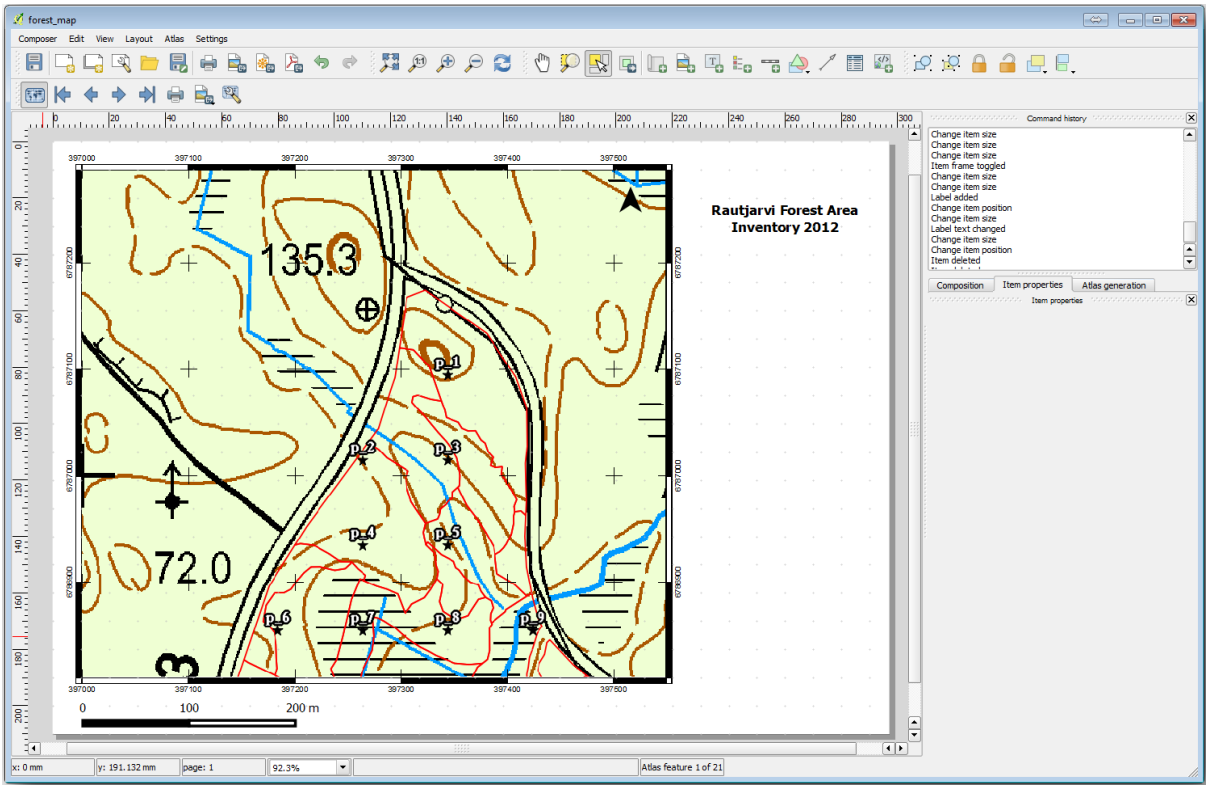
1. Right click on *forest_stands_2012* and select *Duplicate*
2. You get a new layer named `forest_stands_2012 copy` that you can use to define a different style, for example with no filling and red borders.

   Now you have two different visualizations of the forest stands and you can decide which one to display for your detail map.
3. Go back to the print layout window often to see what the map would look like. For the purposes of creating detailed maps, you are looking for a symbology that looks good not at the scale of the whole forest area (left image below) but at a closer scale (right image below). Remember to use Update map preview and *Set map extent to match main canvas extent* whenever you change the zoom in your map or the layout.



14.6.4 ??? Try Yourself: Create a Basic Map Template

1. Once you have a symbology you are happy with, you are ready to add some more information to your printed map. Add at least the following elements:
   - Title.
   - A scale bar.
   - Grid frame for your map.
   - Coordinates on the sides of the grid.
2. You have created a similar layout already in *Module: Laying out the Maps*. Go back to that module as you need. You can look at this example image for reference:



3. Export your map as an image and look at it.
   1. *Layout* ► *Export as Image…*.
   2. Use for example the *JPG format*.

That is what it will look like when printed.

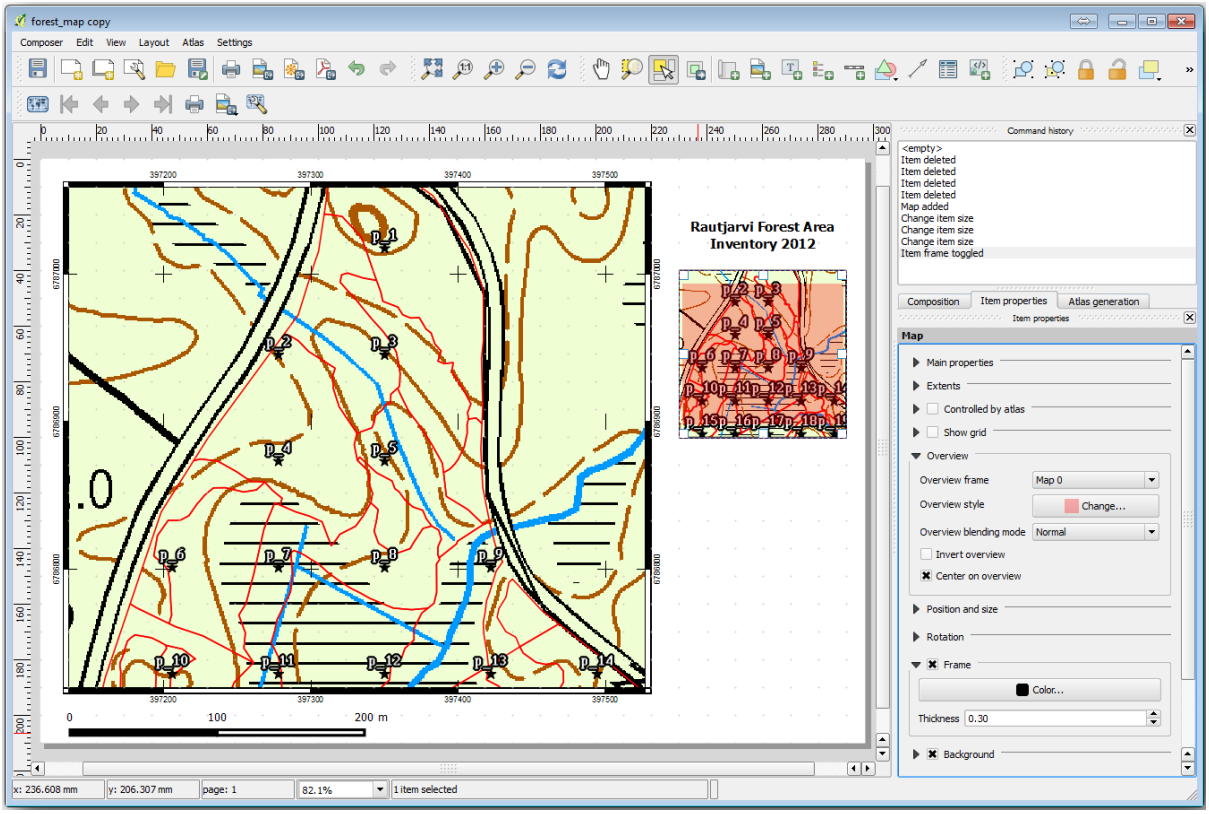
14.6.5 ??? Follow Along: Adding More Elements to the Print Layout

As you probably noticed in the suggested map template images, there are plenty of room on the right side of the canvas. Lets see what else could go in there. For the purposes of our map, a legend is not really necessary, but an overview map and some text boxes could add value to the map.

The overview map will help the field teams place the detail map inside the general forest area:

1. Add another map element to the canvas, right under the title text.
2. In the *Item properties* tab, open the *Overview* dropdown.

3. Set the *Overview frame* to *Map 0*. This creates a shadowed rectangle over the smaller map representing the extent visible in the bigger map.
4. Check also the *Frame* option with a black color and a *Thickness* of `0.30`.



Notice that your overview map is not really giving an overview of the forest area which is what you want. You want this map to represent the whole forest area and you want it to show only the background map and the `forest_stands_2012` layer, and not display the sample plots. And also you want to lock its view so it does not change anymore whenever you change the visibility or order of the layers.

1. Go back to the map, but don't close the *Print Layout*.
2. Right click the *forest_stands_2012* layer and click on *Zoom to Layer Extent*.
3. Deactivate all layers except for *basic_map* and *forest_stands_2012*.
4. Expand the Manage map themes tool on the *Layers* panel and select *Add theme*.
5. Name it `basic_overview`.
6. Go back to the print layout.
7. With the small map selected, click the *Set map extent to match main canvas extent* to set its extents to what you can see in the map window.
8. Lock the view for the overview map by checking *Follow map theme* and selecting `basic_overview` under *Main properties*.

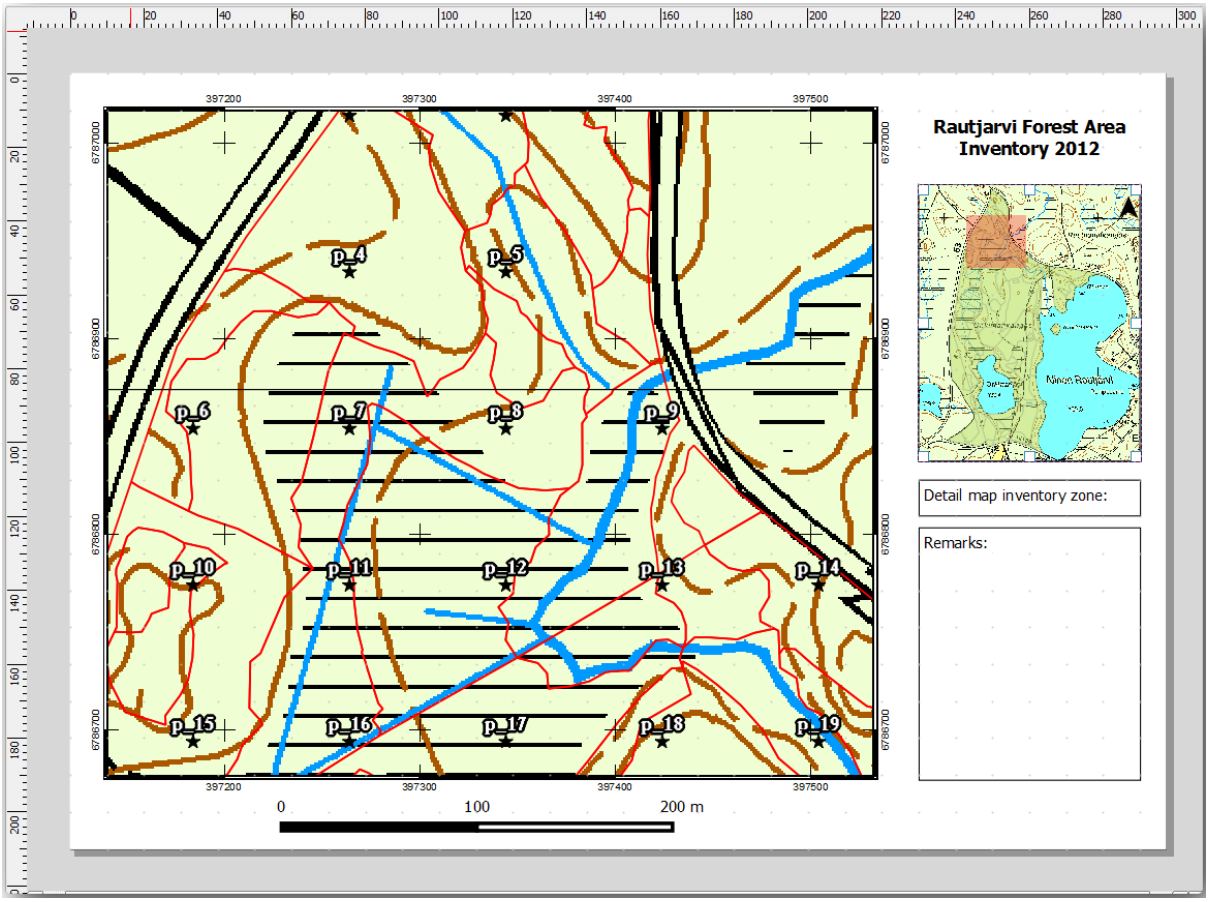
Now your overview map is more what you expected and its view will not change anymore. But, of course, now your detail map is not showing anymore the stand borders nor the sample plots. Let's fix that:

1. Go to the map window again and select the layers you want to be visible (`systematic_plots_clip`, `forest_stands_2012 copy` and `Basic_map`).
2. Zoom again to have only a few lines of sample plots visible.
3. Go back to the *Print Layout* window.

4. Select the bigger map in your layout.
5. In *Item properties* click on *Update preview* and *Set map extent to match main canvas extent*.

Notice that only the bigger map is displaying the current map view, and the small overview map is keeping the same view you had when you locked it.

Note also that the overview is showing a shaded frame for the extent shown in the detail map.



Your template map is almost ready. Add now two text boxes below the map, one containing the text “Detailed map zone: “ and the other one “Remarks: “. Place them as you can see in the image above.

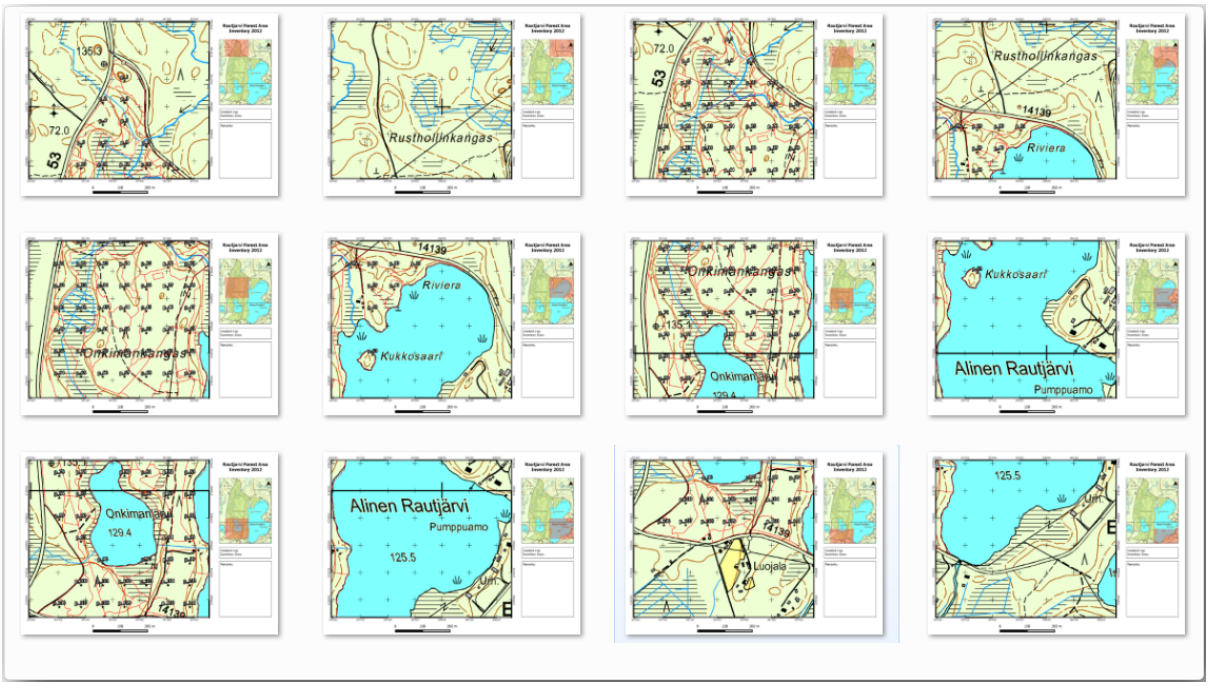
You can also add a North arrow to the overview map:

1. Select the Add North Arrow tool.
2. Click and drag a rectangle at the upper right corner of the overview map.
3. In *Item properties* Check *SVG image*
4. Browse *SVG Browser* ► *SVG Groups* for an arrow image.
5. Under *Image rotation*, check the *Sync with map* and select *Map 1* (the overview map).
6. Resize the arrow image to a size that looks good on the small map.

The basic map layout is ready, now you want to make use of the Atlas tool to generate as many detail maps in this format as you consider necessary.

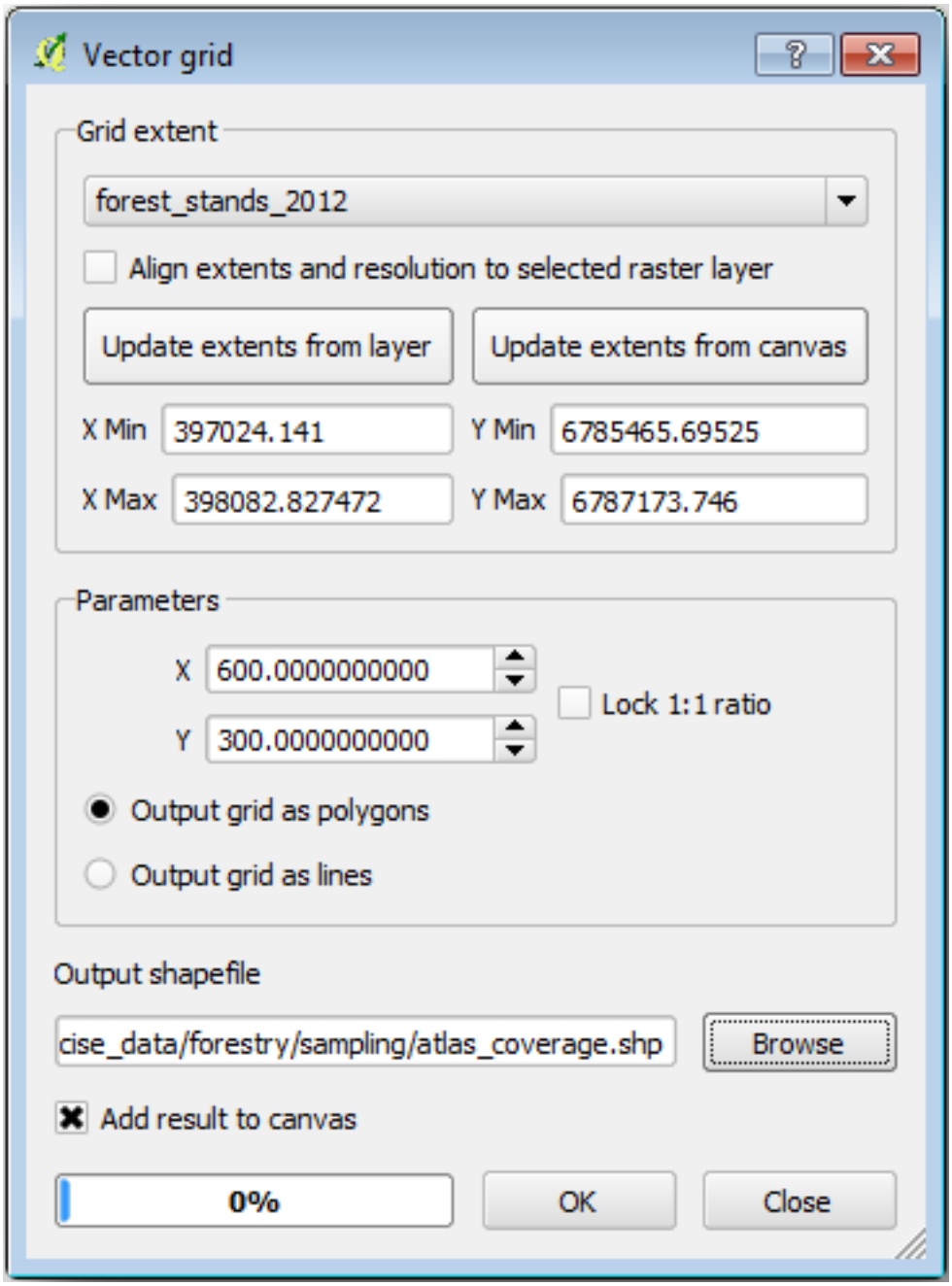
14.6.6 ??? Follow Along: Creating an Atlas Coverage

The Atlas coverage is just a vector layer that will be used to generate the detail maps, one map for every feature in the coverage. To get an idea of what you will do next, here is a full set of detail maps for the forest area:



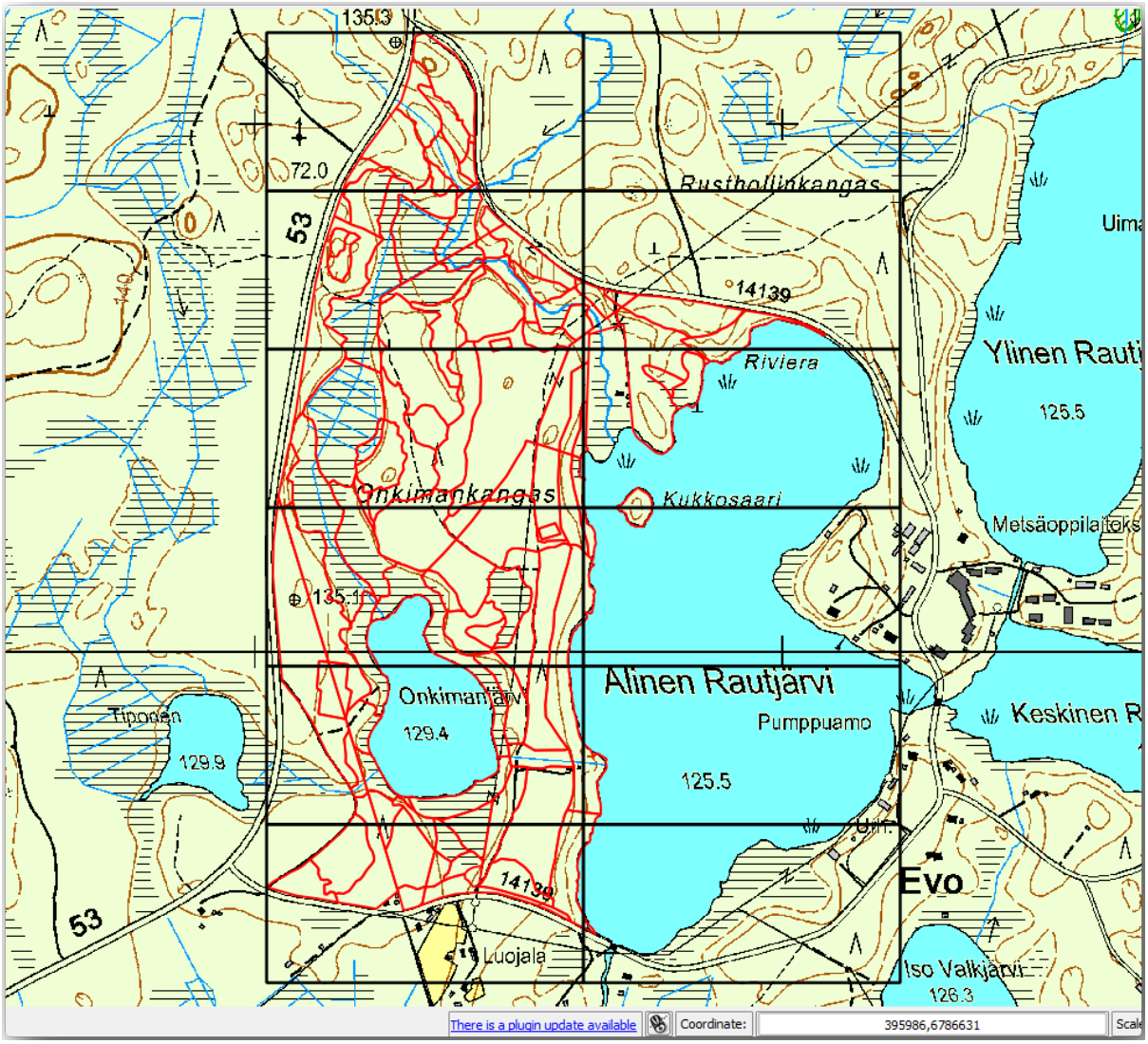
The coverage could be any existing layer, but usually it makes more sense to create one for the specific purpose. Let's create a grid of polygons covering the forest area:

1. In the QGIS map view, open *Vector* ► *Research Tools* ► *Create grid*.
2. Set the tool as shown in this image:



3. Save the output as `atlas_coverage.shp`.
4. Style the new `atlas_coverage` layer so that the polygons have no filling.

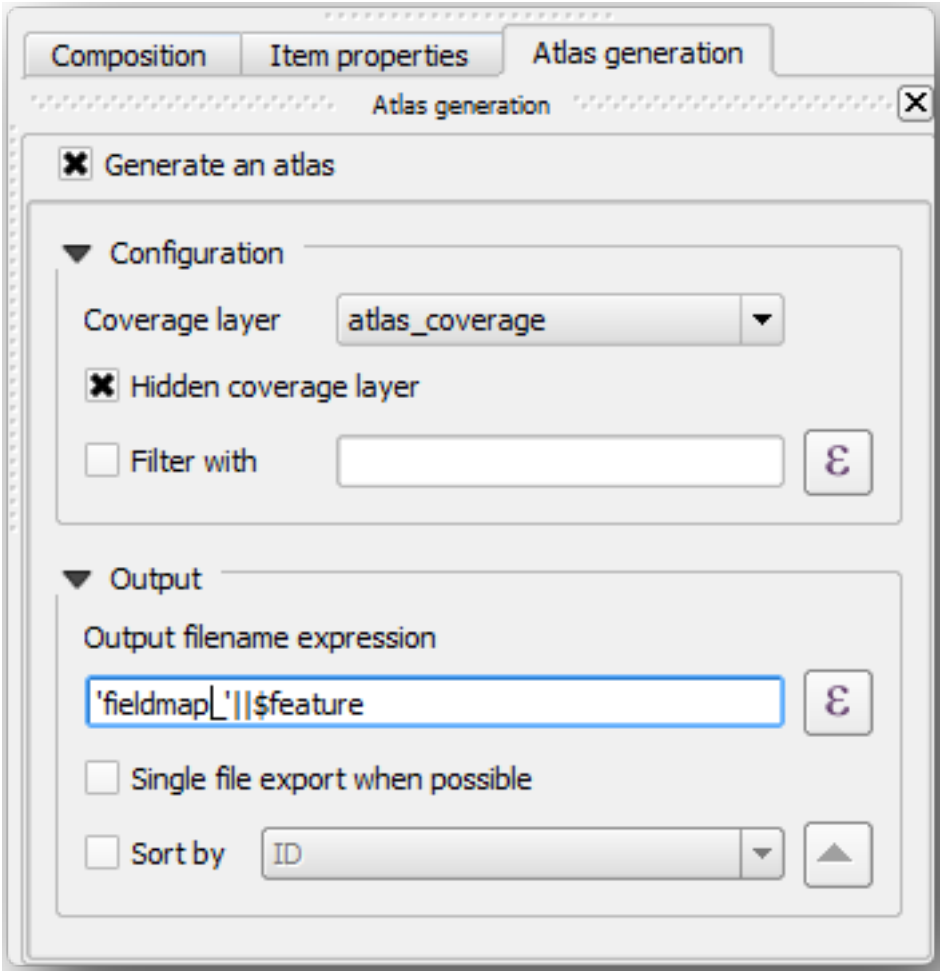
The new polygons are covering the whole forest area and they give you an idea of what each map (created from each polygon) will contain.



14.6.7 ??? Follow Along: Setting Up the Atlas Tool

The last step is to set up the Atlas tool:

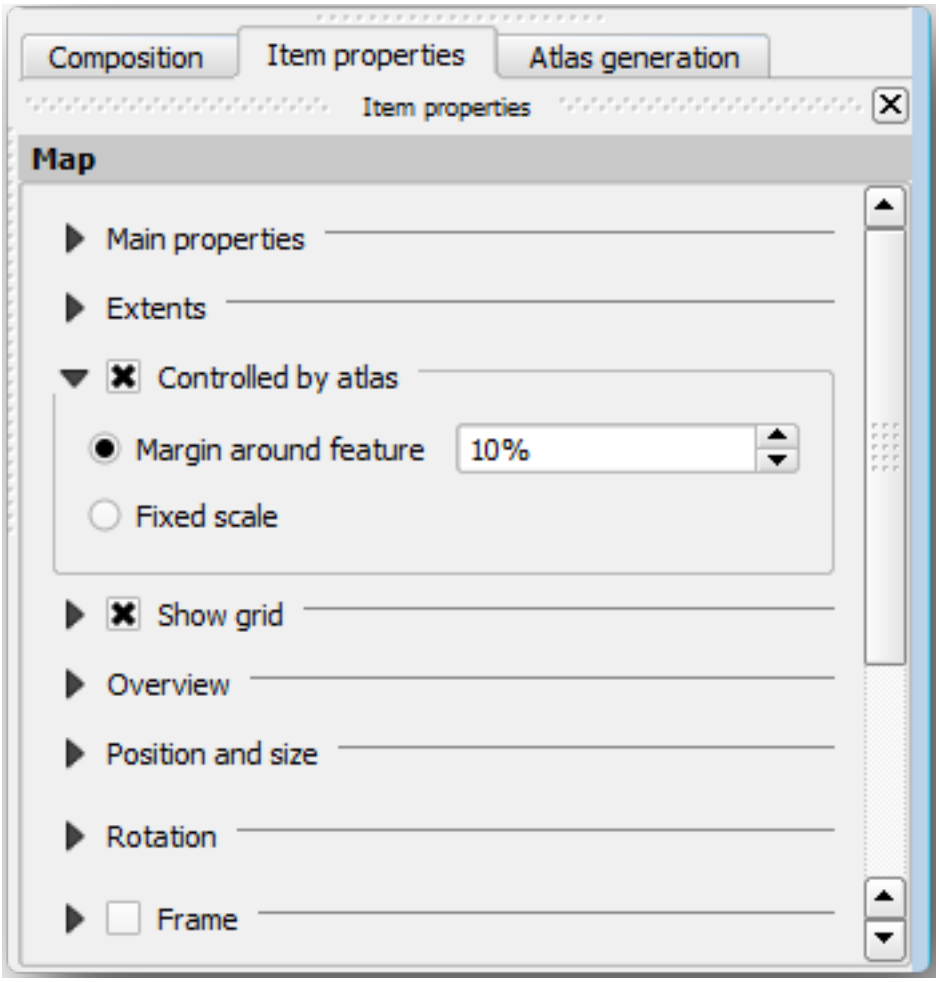
1. Go back to the print layout.
2. In the panel on the right, go to the *Atlas generation* tab.
3. Set the options as follows:



That tells the Atlas tool to use the features (polygons) inside `atlas_coverage` as the focus for every detail map. It will output one map for every feature in the layer. The *Hidden coverage layer* tells the Atlas to not show the polygons in the output maps.

One more thing needs to be done. You need to tell the Atlas tool what map element is going to be updated for every output map. By now, you probably can guess that the map to be changed for every feature is the one you have prepared to contain detail views of the sample plots, that is the bigger map element in your canvas:

1. Select the bigger map element (`Map 0`).
2. Go to the *Item properties* tab.
3. In the list, check *Controlled by atlas*.
4. And set the *Marging around feature* to `10%`. The view extent will be 10% bigger than the polygons, which means that your detail maps will have a 10% overlap.



Now you can use the preview tool for Atlas maps to review what your maps will look like:

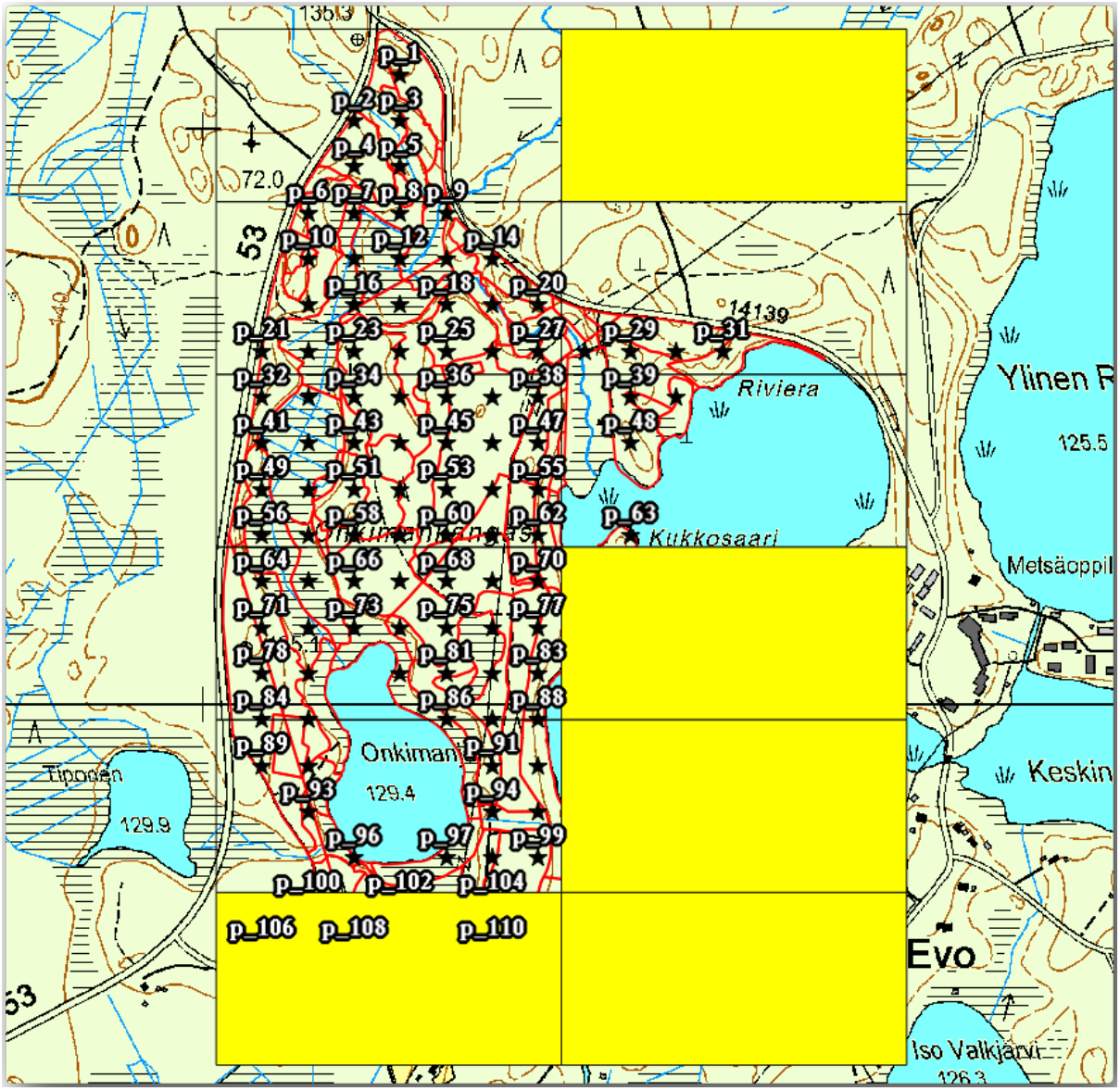
1. Activate the Atlas previews using the button or if your Atlas toolbar is not visible, via *Atlas* ► *Preview Atlas*.
2. You can use the arrows in the Atlas tool bar or in the *Atlas* menu to move through maps that will be created.

Note that some of them cover areas that are not interesting. Let's do something about it and save some trees by not printing those useless maps.

14.6.8 ??? Follow Along: Editing the Coverage Layer

Besides removing the polygons for those areas that are not interesting, you can also customize the text labels in your map to be generated with content from the *Attribute table* of your coverage layer:

1. Go back to the map view.
2. Enable editing for the *atlas_coverage* layer.
3. Select the polygons that are selected (in yellow) in the image below.
4. Remove the selected polygons.
5. Disable editing and save the edits.



You can go back to the print layout and check that the previews of the Atlas use only the polygons you left in the layer.

The coverage layer you are using does not yet have useful information that you could use to customize the content of the labels in your map. The first step is to create them, you can add for example a zone code for the polygon areas and a field with some remarks for the field teams to have into account:

1. Open the *Attribute table* for the *atlas_coverage* layer.
2. Enable editing.
3. Use the calculator to create and populate the following two fields.
4. Create a field named `Zone` of type *Whole number (integer)*.
5. In the *Expression* box write/copy/construct `@row_number`.
6. Create another field named `Remarks`, of type *Text (string)* and a width of `255`.
7. In the *Expression* box write `'No remarks.'`. This will set all the default value for all the polygons.

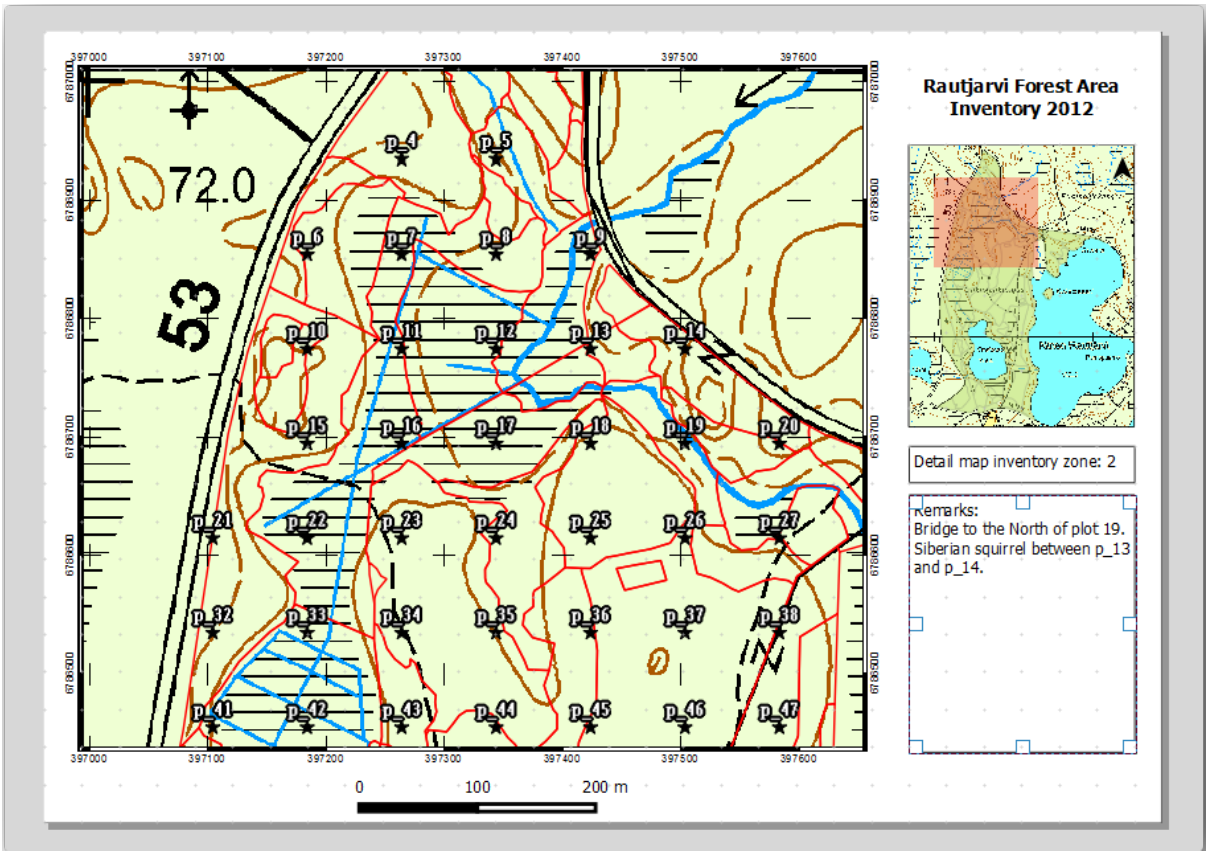
The forest manager will have some information about the area that might be useful when visiting the area. For example, the existence of a bridge, a swamp or the location of a protected species. The *atlas_coverage* layer is

probably in edit mode still, add the following text in the *Remarks* field to the corresponding polygons (double click the cell to edit it):

- For the *Zone* 2: `Bridge to the North of plot 19. Siberian squirrel between p_13 and p_14.`
- For the *Zone* 6: *Difficult to transit in swamp to the North of the lake.`*
- For the *Zone* 7: `Siberian squirrel to the South East of p_94.`
- Disable editing and save your edits.

Almost ready, now you have to tell the Atlas tool that you want some of the text labels to use the information from the *atlas_coverage* layer's attribute table.

1. Go back to the *Print Layout*.
2. Select the text label containing `Detailed map...`.
3. Set the *Font* size to `12`.
4. Set the cursor at the end of the text in the label.
5. In the *Item properties* tab, inside the *Main properties* click on *Insert or Edit an Expression…*.
6. In the *Function list* double-click on the field *Zone* under *Field and Values*.
7. Click *OK*.
8. The text inside the box in the *Item properties* should show `Detail map inventory zone: [% "Zone" %]`. Note that the `[% "Zone" %]` will be substituted by the value of the field *Zone* for the corresponding feature from the layer *atlas_coverage*.
9. Test the contents of the label by looking at the different Atlas preview maps.
10. Do the same for the labels with the text *Remarks:* using the field with the zone information. You can leave a break line before you enter the expression. You can see the result for the preview of zone 2 in the image below:



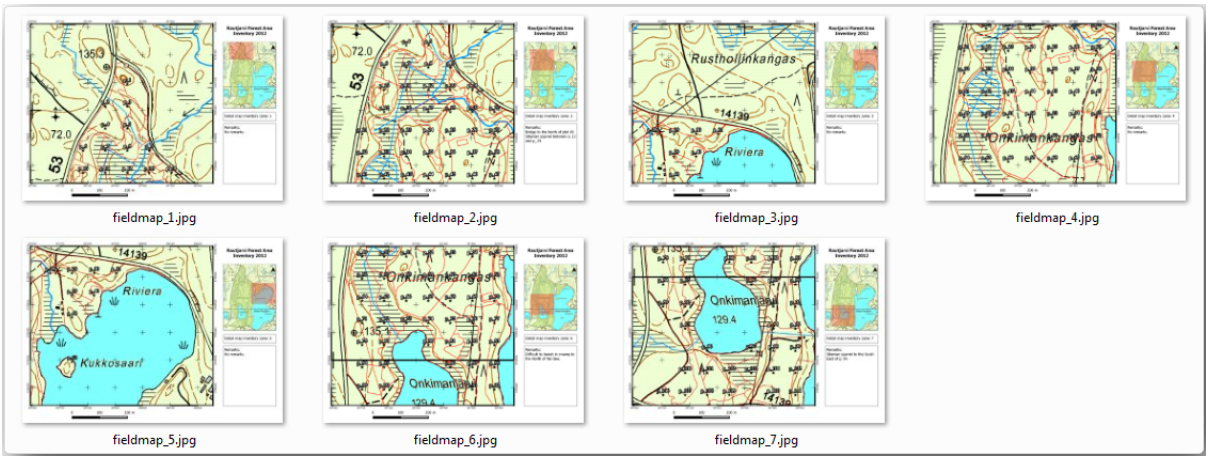
11. Use the Atlas preview to browse through all the maps you will be creating soon and enjoy!

14.6.9 ??? Follow Along: Printing the Maps

Last but not least, printing or exporting your maps to image files or PDF files. You can use the *Atlas* ► *Export Atlas as Images...* or *Atlas* ► *Export Atlas as PDF....* Currently the SVG export format is not working properly and will give a poor result.

Lets print the maps as a single PDF that you can send to the field office for printing:

1. Go to the *Atlas generation* tab on the right panel.
2. Under the *Output* check the *Single file export when possible*. This will put all the maps together into a PDF file, if this option is not checked you will get one file for every map.
3. Open *Layout* ► *Export as PDF....*
4. Save the PDF file as `inventory_2012_maps.pdf` in your `exercise_data\forestry\samplig\map_creation\` folder.
5. Open the PDF file to check that everything went as expected.
6. You could just as easily create separate images for every map (remember to uncheck the single file creation), here you can see the thumbnails of the images that would be created:



7. In the *Print Layout*, press Save in order to save your print layout changes in the project. This also saves the project file. You can reopen the project at any time and run or edit the atlas.

   It is also possible to save your map as a layout template as `forestry_atlas.qpt` in your `exercise_data\forestry\map_creation\` folder. Use *Layout* ► *Save as Template*. You will be able to use this template again and again in other projects.

8. Close the *Print Layout* and the project.

14.6.10 In Conclusion

You have managed to create a template map that can be used to automatically generate detail maps to be used in the field to help navigate to the different plots. As you noticed, this was not an easy task but the benefit will come when you need to create similar maps for other regions and you can use the template you just saved.

14.6.11 What's Next?

In the next lesson, you will see how you can use LiDAR data to create a DEM and then use it to enhance your data and maps visibility.

14.7 Lesson: Calculating the Forest Parameters

Estimating the parameters of the forest is the goal of the forest inventory. Continuing the example from previous lesson, you will use the inventory information gathered in the field to calculate the forest parameters, for the whole forest first, and then for the stands you digitized before.

The goal for this lesson: Calculate forest parameters at general and stand level.

14.7.1 ??? Follow Along: Adding the Inventory Results

The field teams visited the forest and with the help of the information you provided, gathered information about the forest at every sample plot.

Most often the information will be collected into paper forms in the field, then typed to a spreadsheet. The sample plots information has been condensed into a `.csv` file that can be easily open in QGIS.

Continue with the QGIS project from the lesson about designing the inventory, you probably named it `forest_inventory.qgs`.

First, add the sample plots measurements to your QGIS project:

1. Go to *Layer* ► *Add Layer* ► *Add Delimited Text Layer...*.
2. Browse to the file `systematic_inventory_results.csv` located in `exercise_data/forestry/results/`.
3. Garanta que a opção *Coordenadas de Pontos* está activada .
4. Set the fields for the coordinates to the *X* and *Y* fields.
5. Click *OK*.
6. When prompted, select *ETRS89 / ETRS-TM35FIN* as the CRS.
7. Abra a nova *Tabela de atributos* da camada e dê uma vista de olhos nos dados.

You can read the type of data that is contained in the sample plots measurements in the text file `legend_2012_inventorydata.txt` located in the `exercise_data/forestry/results/` folder.

The `systematic_inventory_results` layer you just added is actually just a virtual representation of the text information in the `.csv` file. Before you continue, convert the inventory results to a real spatial dataset:

1. Right click on the `systematic_inventory_results` layer.
2. Browse to `exercise_data/forestry/results/` folder.
3. Name the file `sample_plots_results.shp`.
4. Active a opção *Adicionar ficheiro guardado ao mapa*.
5. Remove the `systematic_inventory_results` layer from your project.

14.7.2 ??? Follow Along: Whole Forest Parameters Estimation

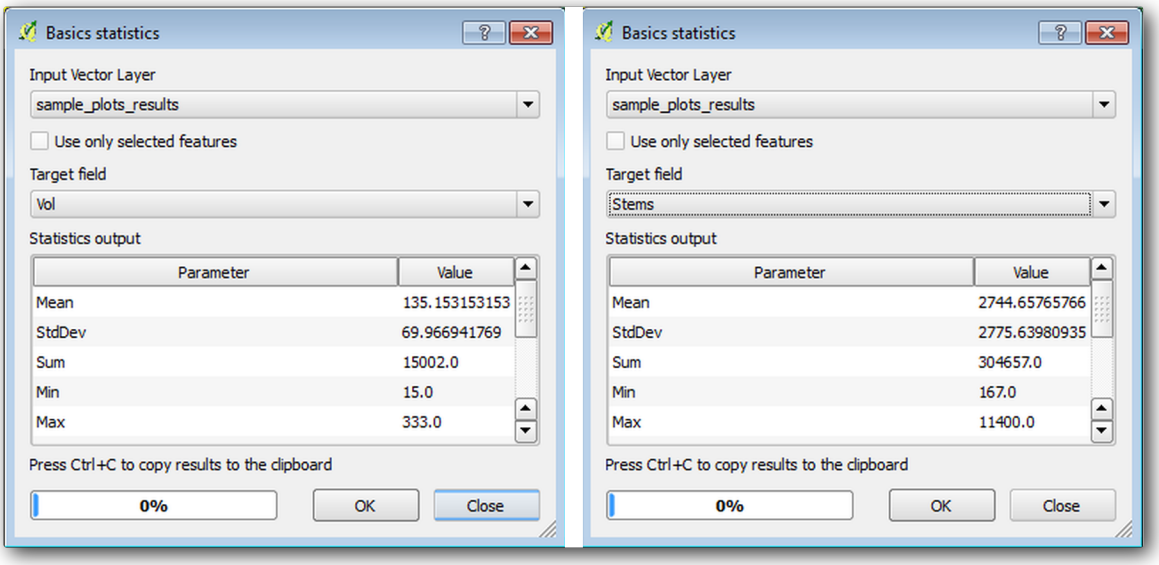
You can calculate the averages for this whole forest area from the inventory results for the some interesting parameters, like the volume and the number of stems per hectare. Since the systematic sample plots represent equal areas, you can directly calculate the averages of the volumes and number of stems per hectare from the `sample_plots_results` layer.

You can calculate the average of a field in a vector layer using the *Basic statistics* tool:

1. Open *Vector* ► *Analysis Tools* ► *Basic statistics for Fields*.
2. Select `sample_plots_results` as the *Input Vector Layer*.
3. Select `Vol` as *Target field*.
4. Click *OK*.

The average volume in the forest is `135.2 m3/ha`.

You can calculate the average for the number of stems in the same way, `2745 stems/ha`.



14.7.3 ??? Follow Along: Estimating Stand Parameters

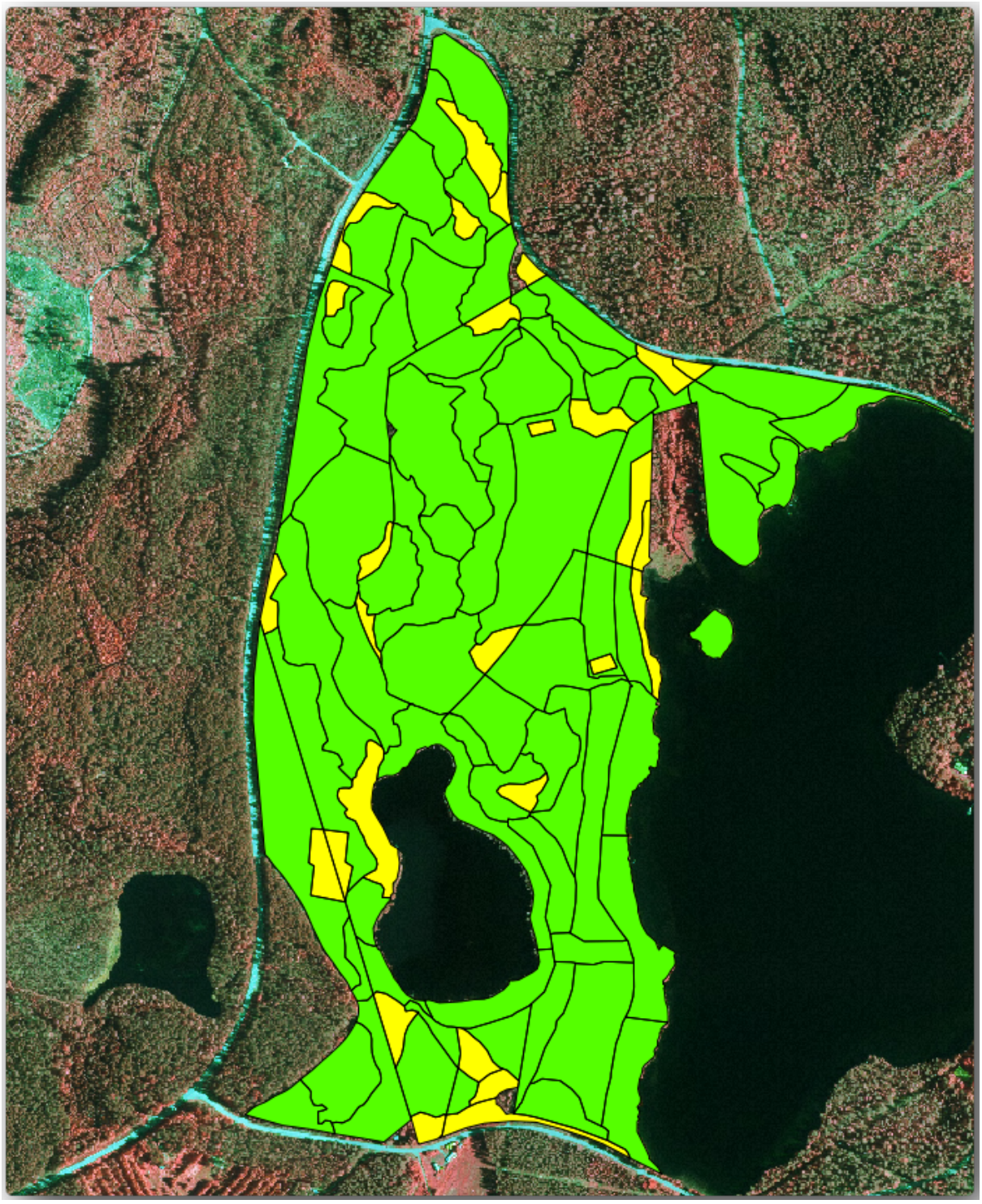
You can make use of those same systematic sample plots to calculate estimates for the different forest stands you digitized previously. Some of the forest stands did not get any sample plot and for those you will not get information. You could have planned some extra sample plots when you planned the systematic inventory, so that the field teams would have measured a few extra sample plots for this purpose. Or you could send a field team later to get estimates of the missing forest stands to complete the stand inventory. Nevertheless, you will get information for a good number of stands just using the planned plots.

What you need is to get the averages of the sample plots that are falling within each of the forest stands. When you want to combine information based on their relative locations, you perform a spatial join:

1. Open the *Vector* ► *Data Management* ► *Join attributes by location* tool.
2. Set `forest_stands_2012` as the *Target vector layer*. The layer you want the results for.
3. Set `sample_plots_results` as the *Join vector layer*. The layer you want to calculate estimates from.
4. Check *Take summary of intersecting features*.
5. Active apenas a opção para calcular a *Média*.

6. Name the result as `forest_stands_2012_results.shp` and save it in the `exercise_data/forestry/results/` folder.
7. Finalmente seleccione *Manter todos os registos…*, para que possa verificar mais tarde os povoamentos que não tinham informação.
8. Click *OK*.
9. Aceite adicionando a nova camada ao seu projecto assim que solicitado.
10. Feche a ferramenta *Juntar atributos por localização* .

Open the *Attribute table* for *forest_stands_2012_results* and review the results you got. Note that a number of forest stands have `NULL` as the value for the calculations, those are the ones having no sample plots. Select them all and view them in the map, they are some of the smaller stands:



Let's calculate now the same averages for the whole forest as you did before, only this time you will use the averages you got for the stands as the bases for the calculation. Remember that in the previous situation, each sample plot represented a theoretical stand of `80x80 m`. Now you have to consider the area of each of the stands individually instead. That way, again, the average values of the parameters that are in, for example, m3/ha for the volumes are converted to total volumes for the stands.

You need to first calculate the areas for the stands and then calculate total volumes and stem numbers for each of them:

1. Na *Tabela de atributos* active a edição.
2. Abra a *Calculadora de campos*.

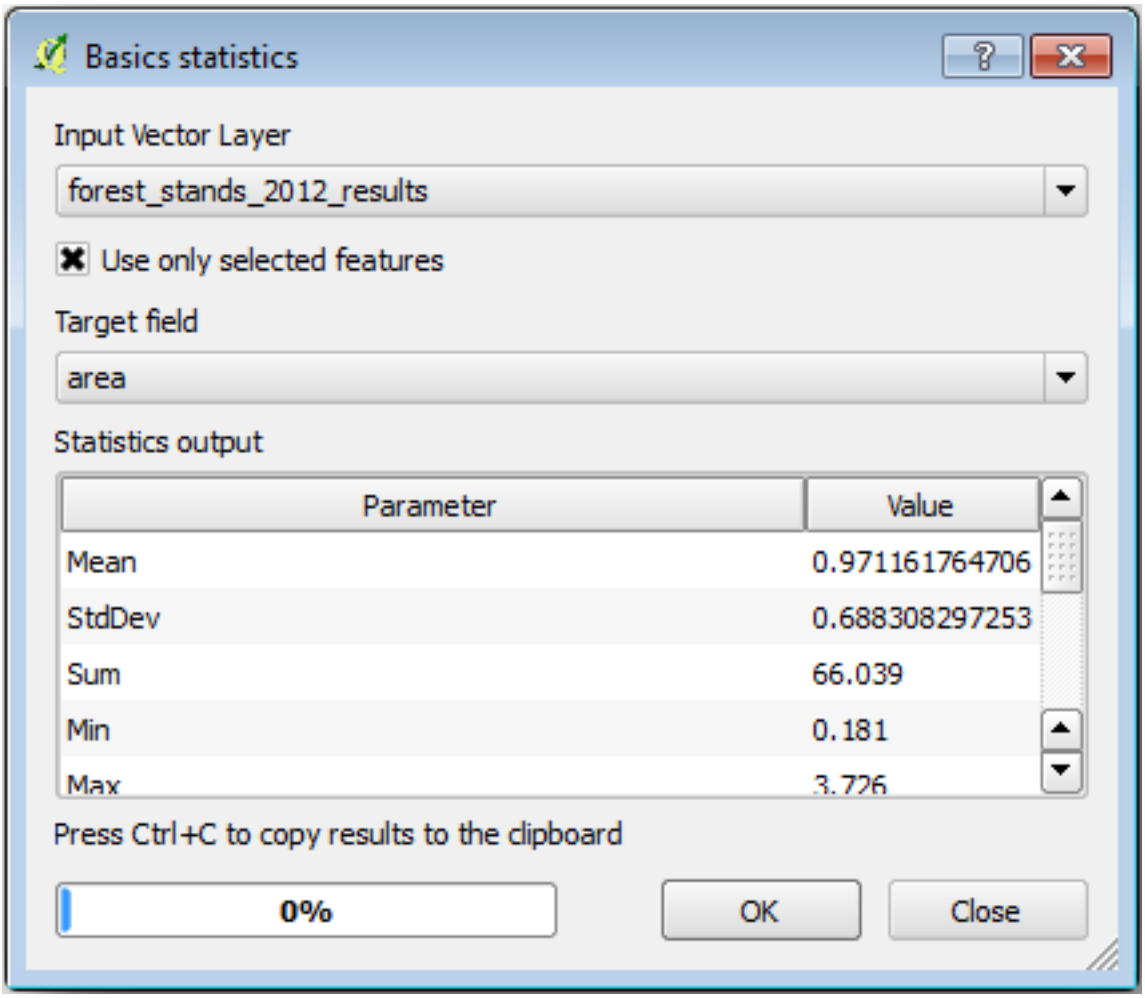
3. Create a new field called `area`.
4. Set the *Output field type* to `Decimal number (real)`.
5. Set the *Precision* to `2`.
6. In the *Expression* box, write `$area / 10000`. This will calculate the area of the forest stands in ha.
7. Click *OK*.

Agora calcule o campo com o volume total e o números de árvores para cada povoamento:

1. Name the fields `s_vol` and `s_stem`.
2. Os campos podem ser números inteiros mas pode usar igualmente os números reais.
3. Use the expressions `"area" * "MEANVol"` and `"area" * "MEANStems"` for total volumes and total stems respectively.
4. Guarde as edições quando terminar.
5. Desactive a edição.

Na situação anterior, as áreas representadas por cada parcela de amostra eram as mesmas, portanto, era suficiente para calcular a média das parcelas de amostra. Agora, para calcular as estimativas, precisa de dividir a soma dos volumes dos povoamentos ou o número de árvores pela soma das áreas dos povoamentos que contêm informações.

1. In the *Attribute table* for the `forest_stands_2012_results` layer, select all the stands containing information.
2. Open *Vector* ► *Analysis Tools* ► *Basic statistics for fields*.
3. Select the `forest_stands_2012_results` as the *Input layer*.
4. Select `area` as *Field to calculate statistics on*.
5. Check the *Selected features only*
6. Click *OK*.



As you can see, the total sum of the stands" areas is `66.04 ha`. Note that the area of the missing forest stands is only about `7 ha`.

In the same way, you can calculate that the total volume for these stands is `8908 m3/ha` and the total number of stems is `179594 stems`.

Usando a informação dos povoamentos florestais, em vez de usar directamente das parcelas de amostragem iremos ter as seguintes médias estimadas:

- `184.9 m3/ha` and
- `2719 stems/ha`.

Save your QGIS project, `forest_inventory.qgs`.

14.7.4 In Conclusion

You managed to calculate forest estimates for the whole forest using the information from your systematic sample plots, first without considering the forest characteristics and also using the interpretation of the aerial image into forest stands. And you also got some valuable information about the particular stands, which could be used to plan the management of the forest in the coming years.

14.7.5 What's Next?

Na lição seguinte, irá primeiro criar um fundo com um mapa de sombras a partir de um conjunto de dados LiDAR que irá usar para preparar a apresentação do mapa com os resultados da floresta que acabou de calcular.

14.8 Lesson: DEM from LiDAR Data

You can improve the look of your maps by using different background images. You could use the basic map or the aerial image you have been using before, but a hillshade raster of the terrain will look nicer in some situations.

You will use LAStools to extract a DEM from a LiDAR dataset and then create a hillshade raster to use in your map presentation later.

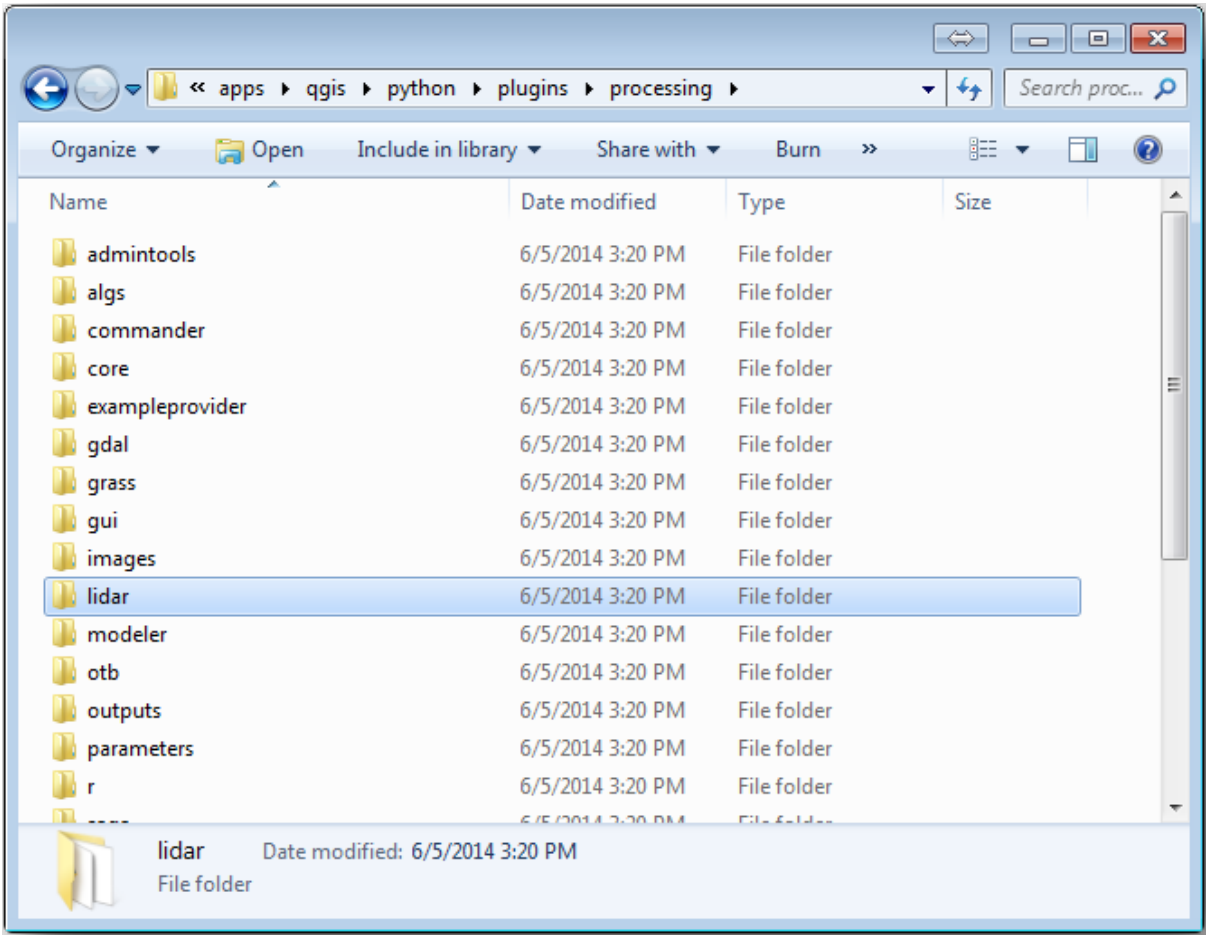
The goal for this lesson: Install LAStools and calculate a DEM from LiDAR data and a hillshade raster.

14.8.1 ??? Follow Along: Installing Lastools

Managing LiDAR data within QGIS is possible using the Processing framework and the algorithms provided by LAStools.

You can obtain a digital elevation model (DEM) from a LiDAR point cloud and then create a hillshade raster that is visually more intuitive for presentation purposes. First you will have to set up the *Processing* framework settings to properly work with LAStools:

- Close QGIS, if you have already started it.
- An old lidar plugin might be installed by default in your system in the folder `C:/Program Files/QGIS Valmiera/apps/qgis/python/plugins/processing/`.
- If you have a folder named `lidar`, delete it. This is valid for some installations of QGIS 2.2 and 2.4.



- Go to the `exercise_data\forestry\lidar\` folder, there you can find the file `QGIS_2_2_toolbox.zip`. Open it and extract the `lidar` folder to replace the one you just deleted.
- If you are using a different QGIS version, you can see more installation instructions in this tutorial.

Now you need to install the LAStools to your computer. Get the newest *lastools* version here and extract the content of the `lastools.zip` file into a folder in your system, for example, `C:\lastools\`. The path to the `lastools` folder cannot have spaces or special characters.

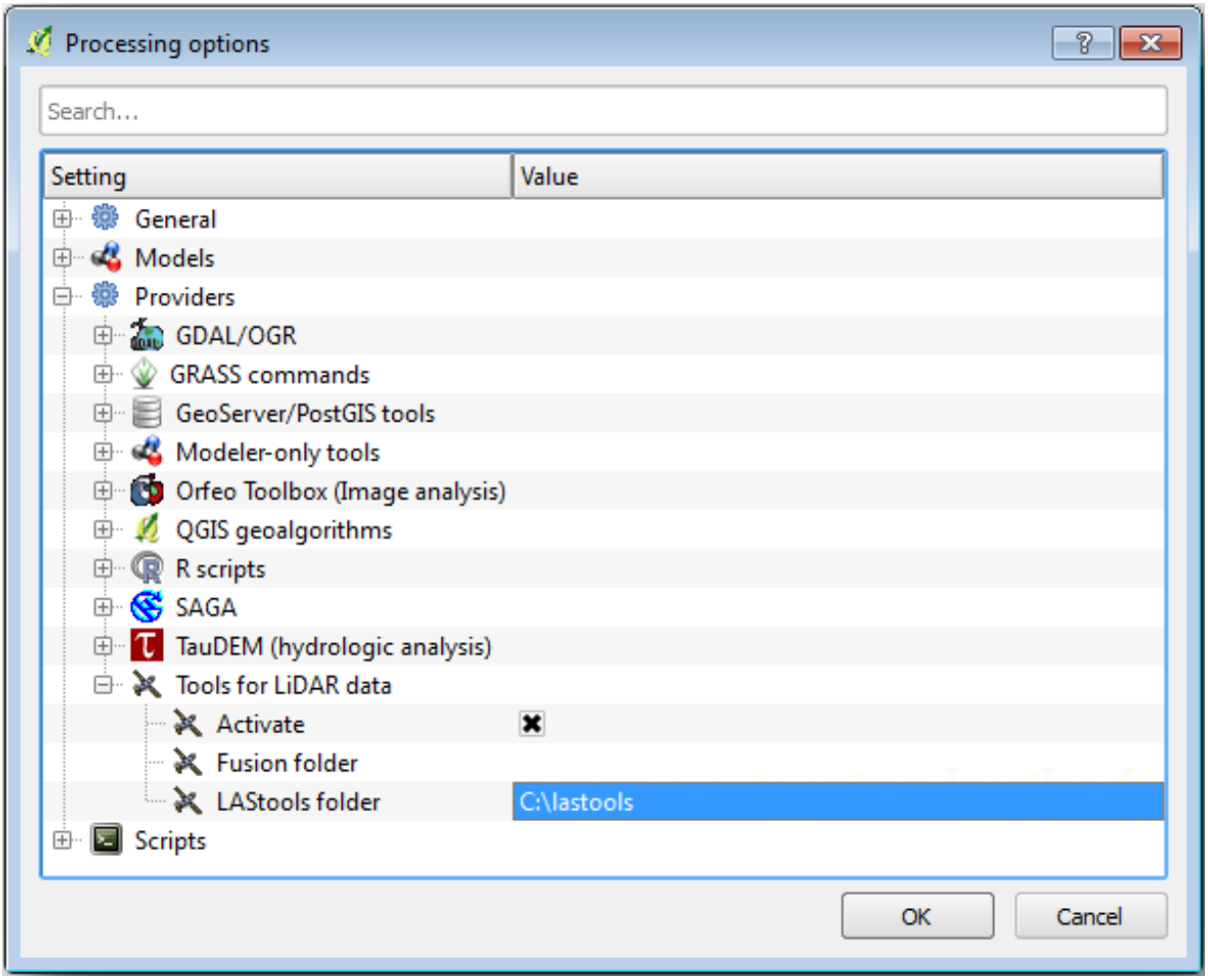
Nota: Read the `LICENSE.txt` file inside the `lastools` folder. Some of the LAStools are open source and other are closed source and require licensing for most commercial and governmental use. For education and evaluation purposes you can use and test LAStools as much as you need to.

The plugin and the actual algorithms are now installed in your computer and almost ready to use, you just need to set up the Processing framework to start using them:

- Open a new project in QGIS.
- Set the project's CRS to `ETRS89 / ETRS-TM35FIN`.
- Save the project as `forest_lidar.qgs`.

To setup the LAStools in QGIS:

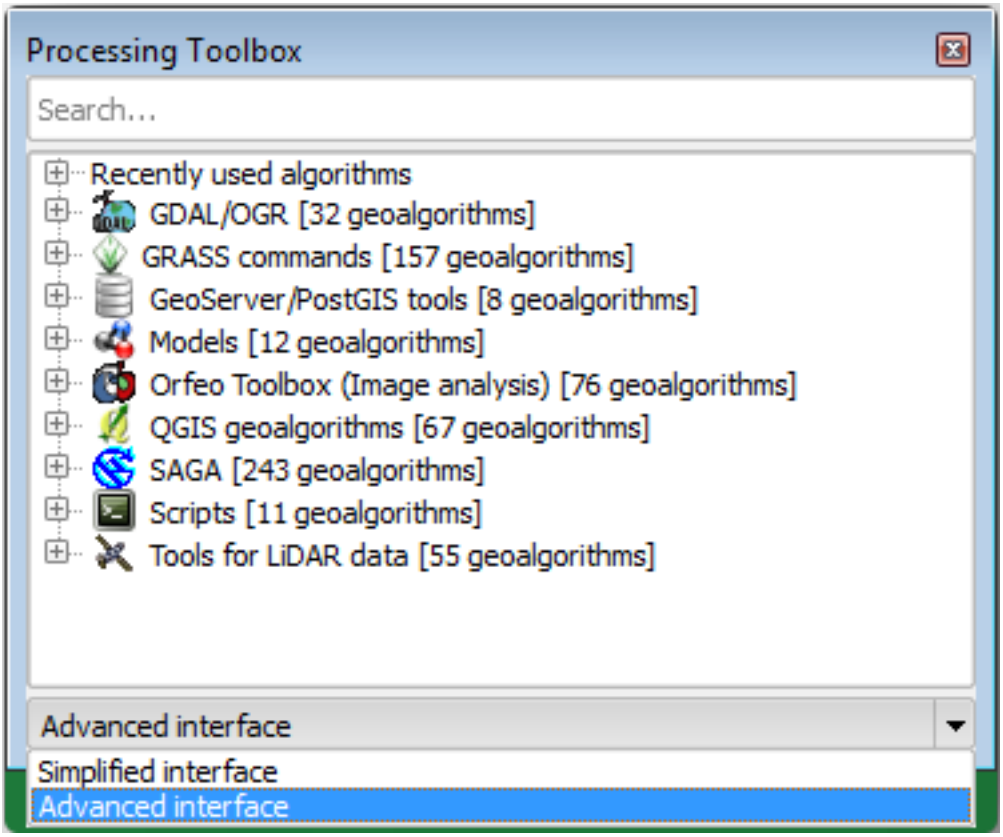
- Go to *Processing* ► *Options and configuration*.
- In the *Processing options* dialog, go to *Providers* and then to *Tools for LiDAR data*.
- Check *Activate*.
- For *LAStools folder* set `c:\lastools\` (or the folder you extracted LAStools to).



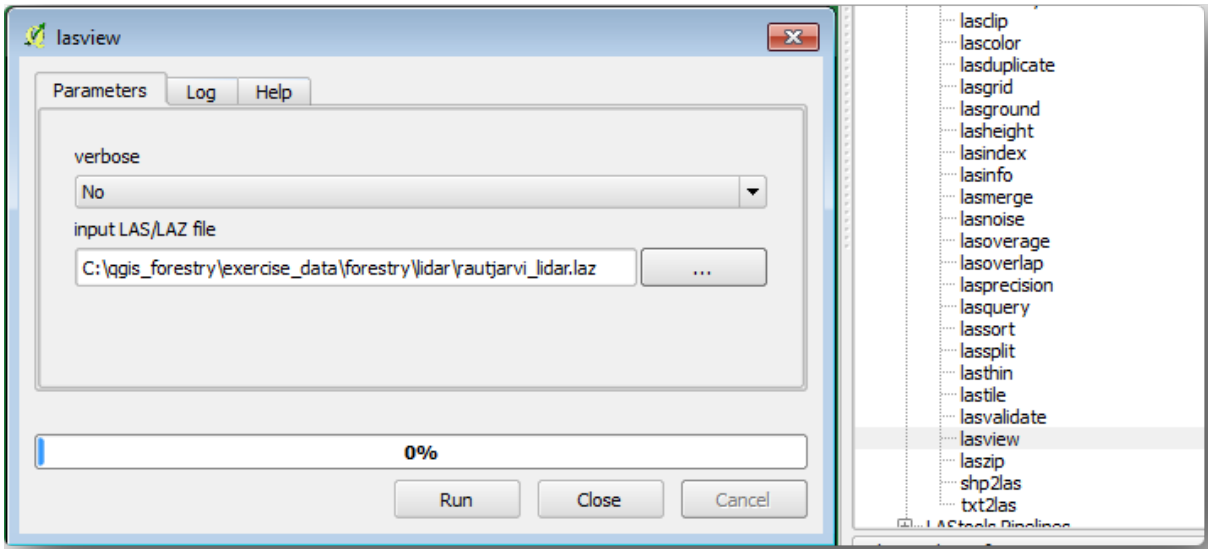
14.8.2 ??? Follow Along: Calculating a DEM with LAStools

You have already used the *Processing* toolbox in *Lesson: Spatial Statistics* to run some SAGA algorithms. Now you are going to use it to run LAStools programs:

- Open *Processing* ► *Toolbox*.
- In the dropdown menu at the bottom, select *Advanced interface*.
- You should see the *Tools for LiDAR data* category.

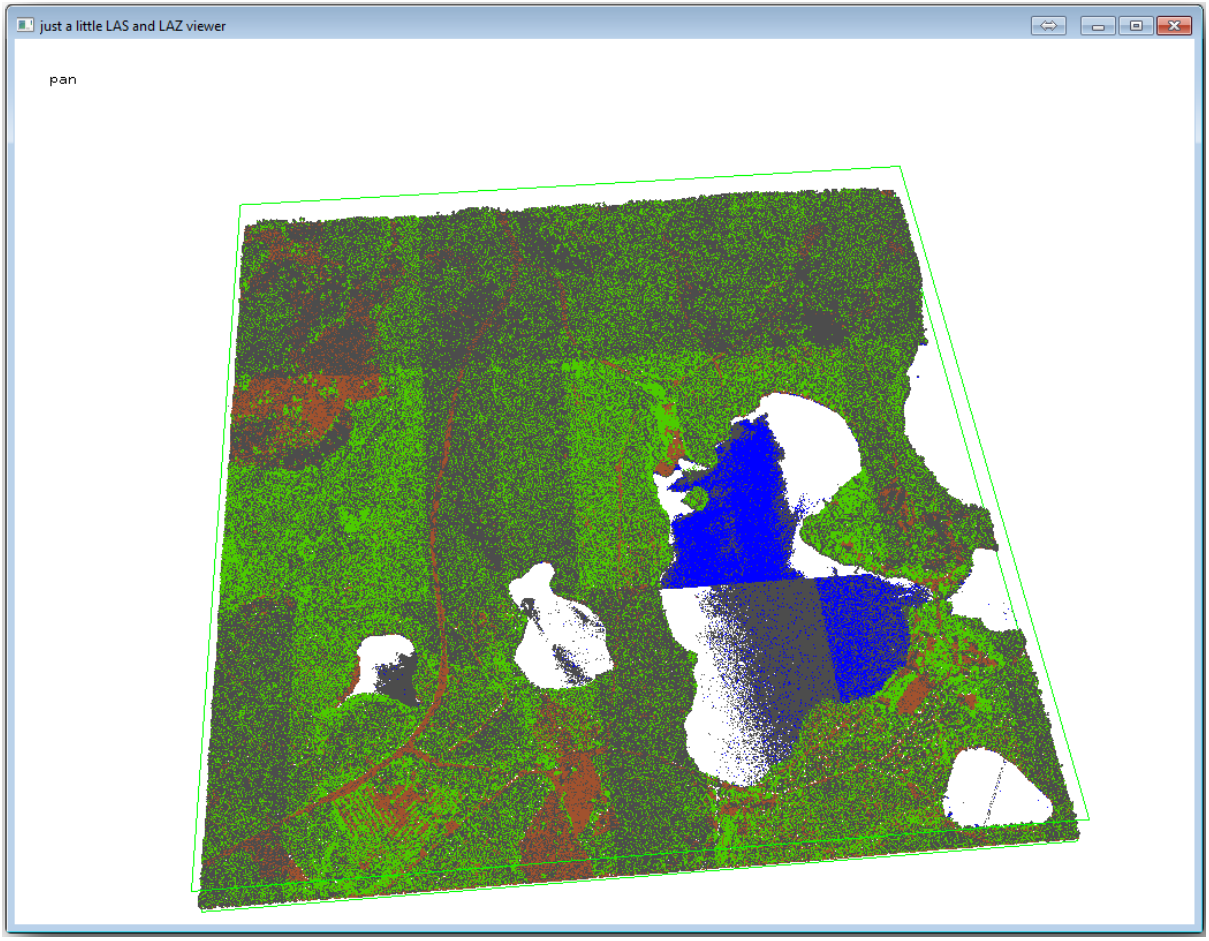


- Expand it to see the tools available, and expand also the *LAStools* category (the number of algorithms may vary).
- Scroll down until you find the *lasview* algorithm, double click it to open.
- At *Input LAS/LAZ file*, browse to `exercise_data\forestry\lidar\` and select the `rautjarvi_lidar.laz` file.



- Click *Run*.

Now you can see the LiDAR data in the *just a little LAS and LAZ viewer* dialog window:



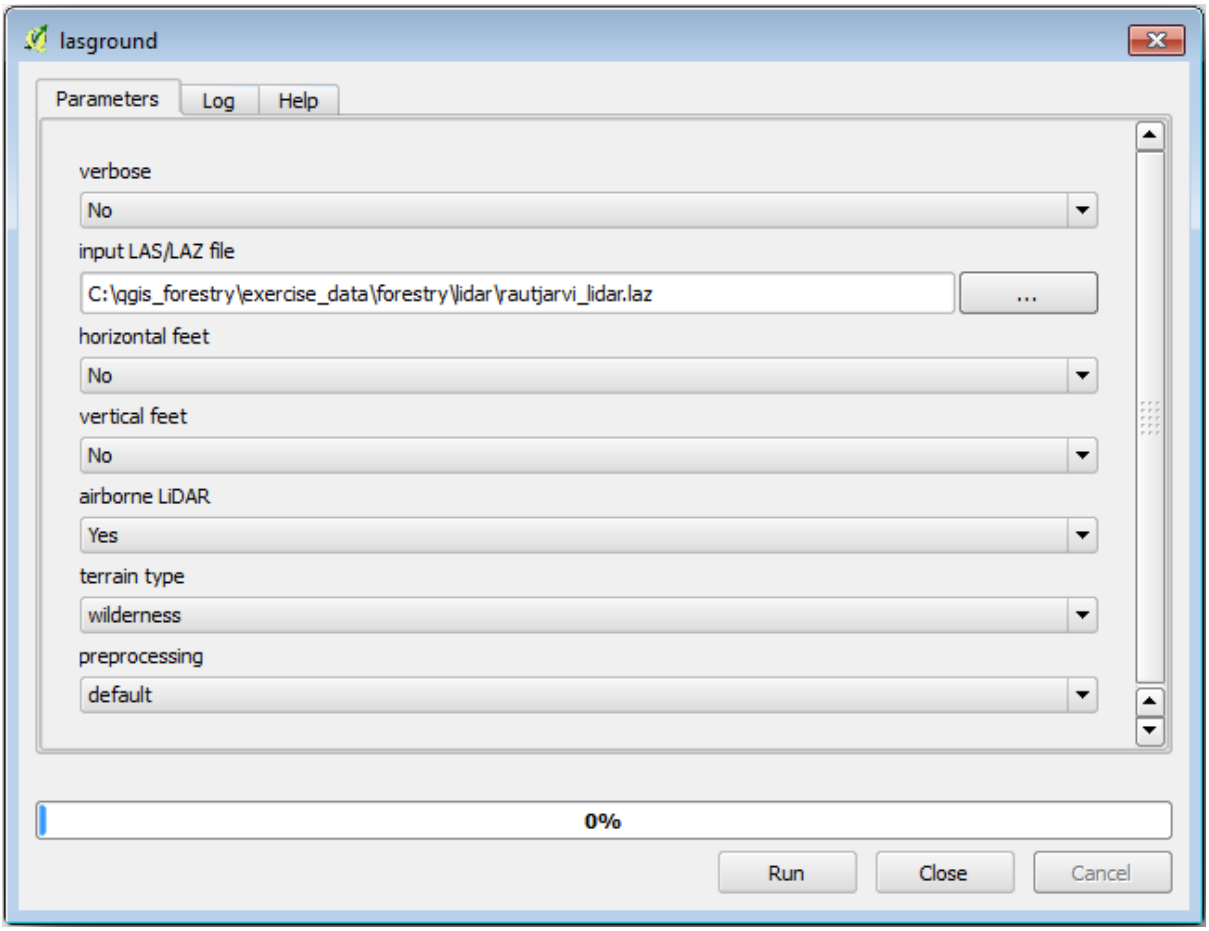
There are many things you can do within this viewer, but for now you can just click and drag on the viewer to pan the LiDAR point cloud to see what it looks like.

Nota: If you want to know further details on how the LAStools work, you can read the `README` text files about each of the tools, in the `C:\lastools\bin\` folder. Tutorials and other materials are available at the Rapidlasso webpage.

- Close the viewer when you are ready.

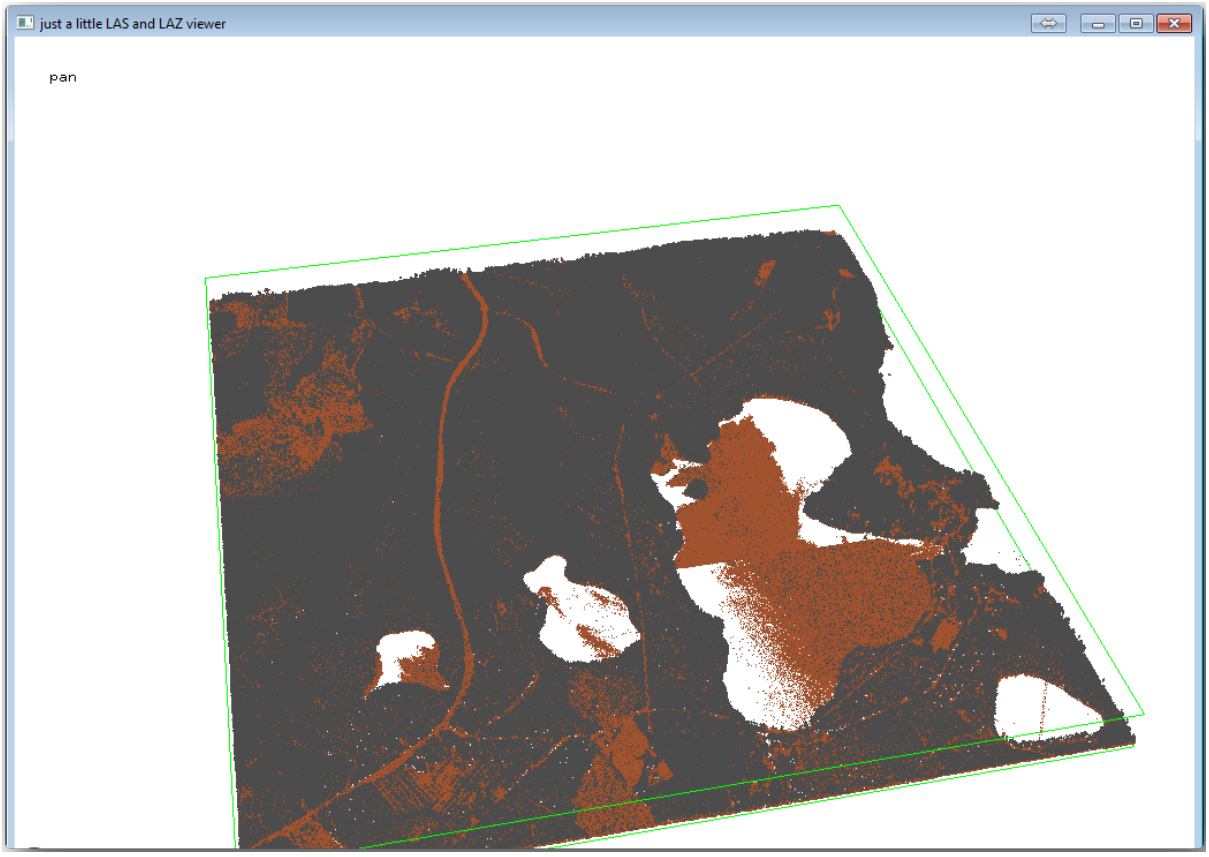
Creating a DEM with LAStools can be done in two steps, first one to classify the point cloud into `ground` and `no ground` points and then calculating a DEM using only the `ground` points.

- Go back to the *Processing Toolbox*.
- Note the *Search...* box, write `lasground`.
- Double click to open the *lasground* tool and set it as shown in this image:



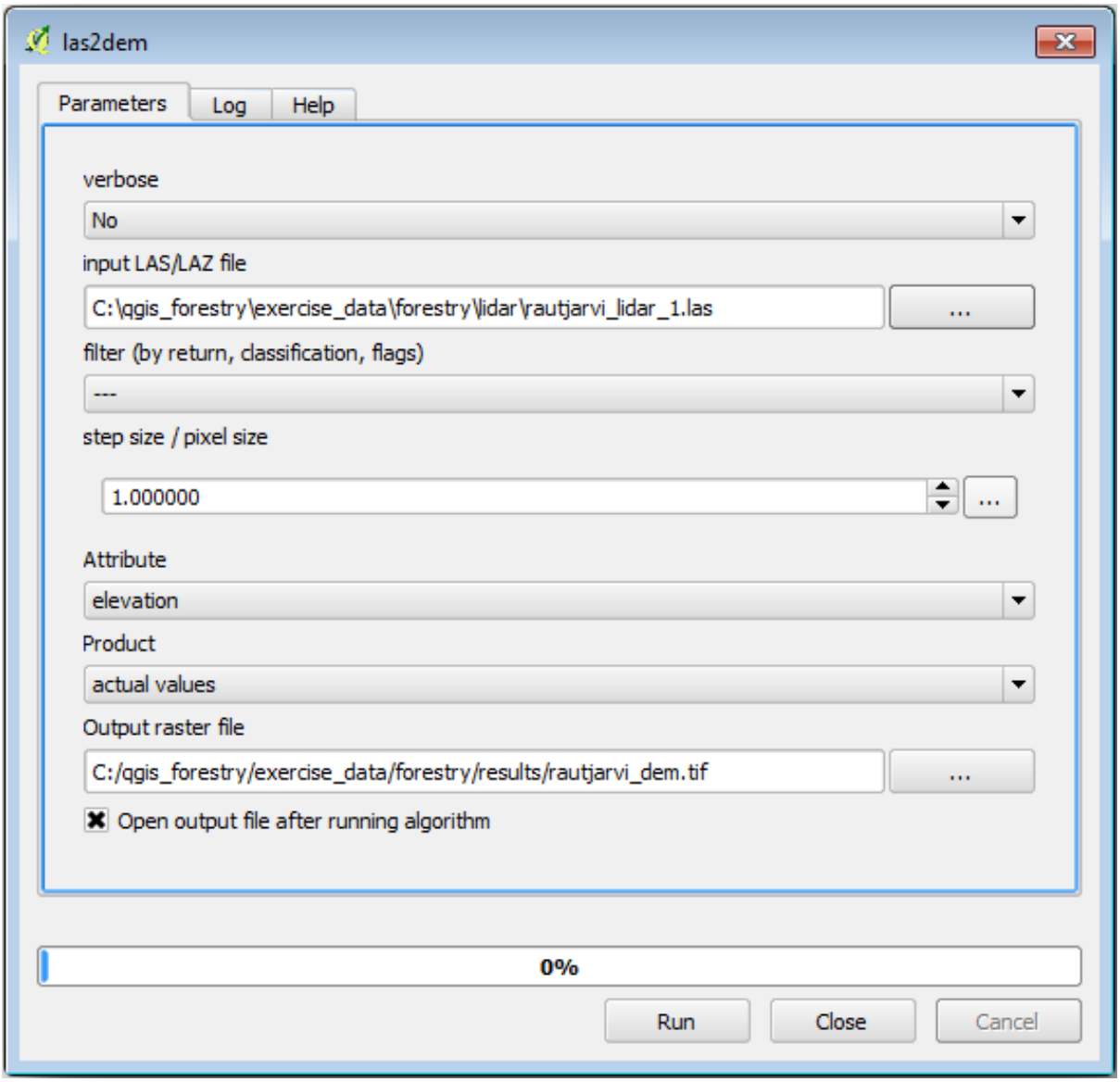
- The output file is saved to the same folder where the `rautjarvi_lidar.laz` is located and it is named `rautjarvi_lidar_1.las`.

You can open it with *lasview* if you want to check it.



The brown points are the points classified as ground and the gray ones are the rest, you can click the letter `g` to visualize only the ground points or the letter `u` to see only the unclassified points. Click the letter `a` to see all the points again. Check the `lasview_README.txt` file for more commands. If you are interested, also this tutorial about editing LiDAR points manually will show you different operations within the viewer.

- Close the viewer again.
- In the *Processing Toolbox*, search for `las2dem`.
- Open the *las2dem* tool and set it as shown in this image:



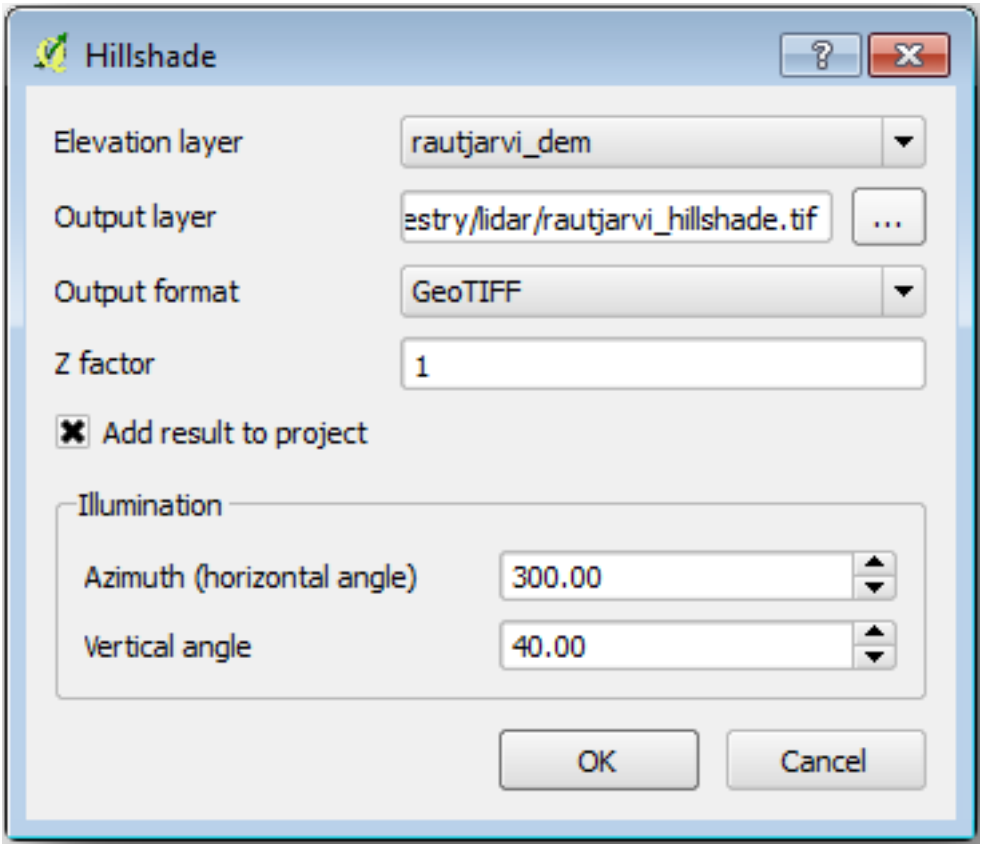
The result DEM is added to your map with the generic name `Output raster file`.

Nota: The *lasground* and *las2dem* tools require licensing. You can use the unlicensed tool as indicated in the license file, but you get the diagonals you can appreciate in the image results.

14.8.3 ??? Follow Along: Creating a Terrain Hillshade

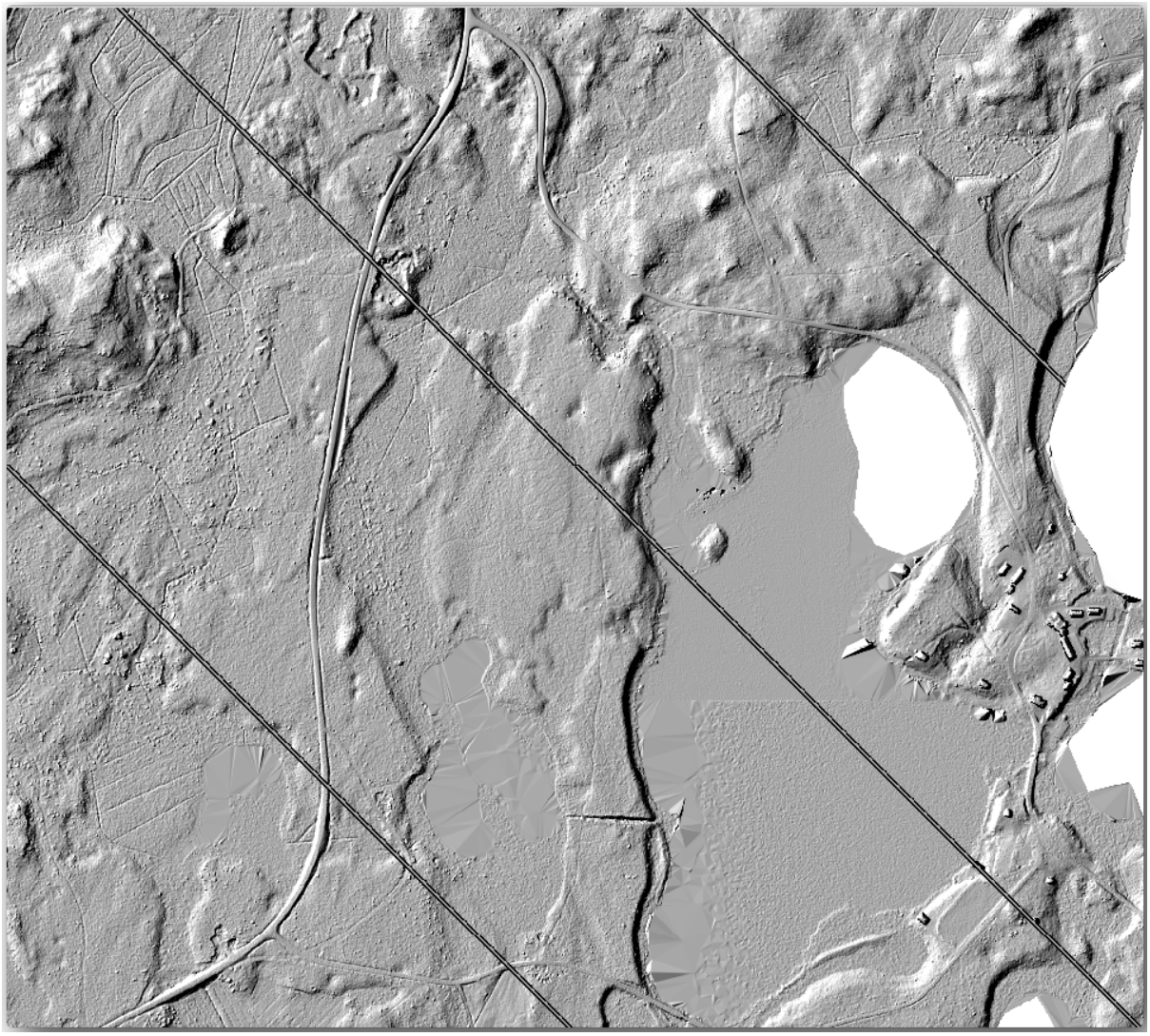
For visualization purposes, a hillshade generated from a DEM gives a better visualization of the terrain:

- Open *Raster* ► *Terrain analysis* ► *Hillshade*.
- As the *Output layer*, browse to `exercise_data\forestry\lidar\` and name the file `hillshade.tif`.
- Leave the rest of parameters with the default settings.



- Select `ETRS89 / ETRS-TM35FIN` as the CRS when prompted.

Despite the diagonal lines remaining in the hillshade raster result, you can clearly see an accurate relief of the area. You can even see the different soil drains that have been dug in the forests.



14.8.4 In Conclusion

Using LiDAR data to get a DEM, specially in forested areas, gives good results with not much effort. You could also use ready LiDAR derived DEMs or other sources like the SRTM 9m resolution DEMs. Either way, you can use them to create a hillshade raster to use in your map presentations.

14.8.5 What's Next?

In the next, and final step in this module, lesson you will use the hillshade raster and the forest inventory results to create a map presentation of the results.

14.9 Lesson: Map Presentation

In the previous lessons you have imported an old forest inventor as a GIS project, updated it to the current situation, designed a forest inventory, created maps for the field work and calculated forest parameters from the field measurements.

It is often important to create maps with the results of a GIS project. A map presenting the results of the forest inventory will make it easier for anyone to have a good idea of what the results are in a quick glance, without looking at the specific numbers.

The goal for this lesson: Create a map to present the inventory results using a hillshade raster as background.

14.9.1 ??? Follow Along: Preparing the Map Data

Open the QGIS project from the parameters calculations lesson, `forest_inventory.qgs`. Keep at least the following layers:

- `forest_stands_2012_results`.
- `basic_map`.
- `rautjarvi_aerial`.
- `lakes` (if you don't have it, add it from the `exercise_data\forestry\` folder).

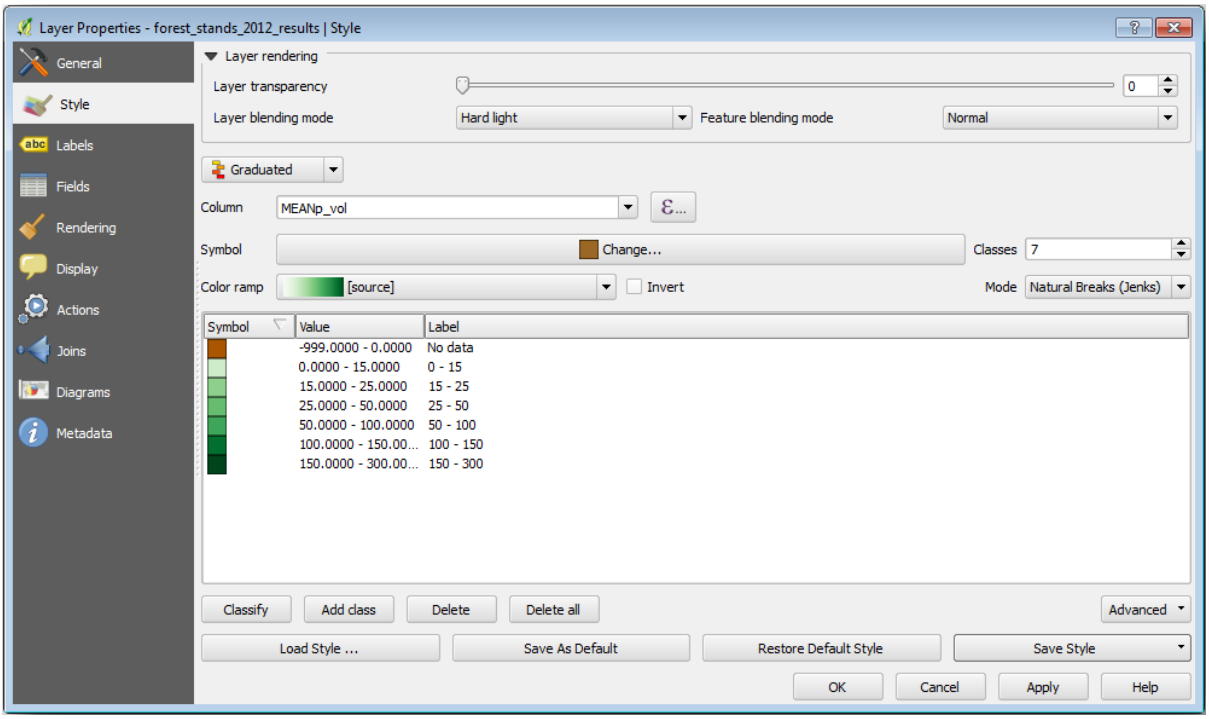
You are going to present the average volumes of your forest stands in a map. If you open the *Attribute table* for the `forest_stands_2012_results` layer, you can see the `NULL` values for the stands without information. To be able to get also those stands into your symbology you should change the `NULL` values to, for example, `-999`, knowing that those negative numbers mean there is no data for those polygons.

For the `forest_stands_2012_results` layer:

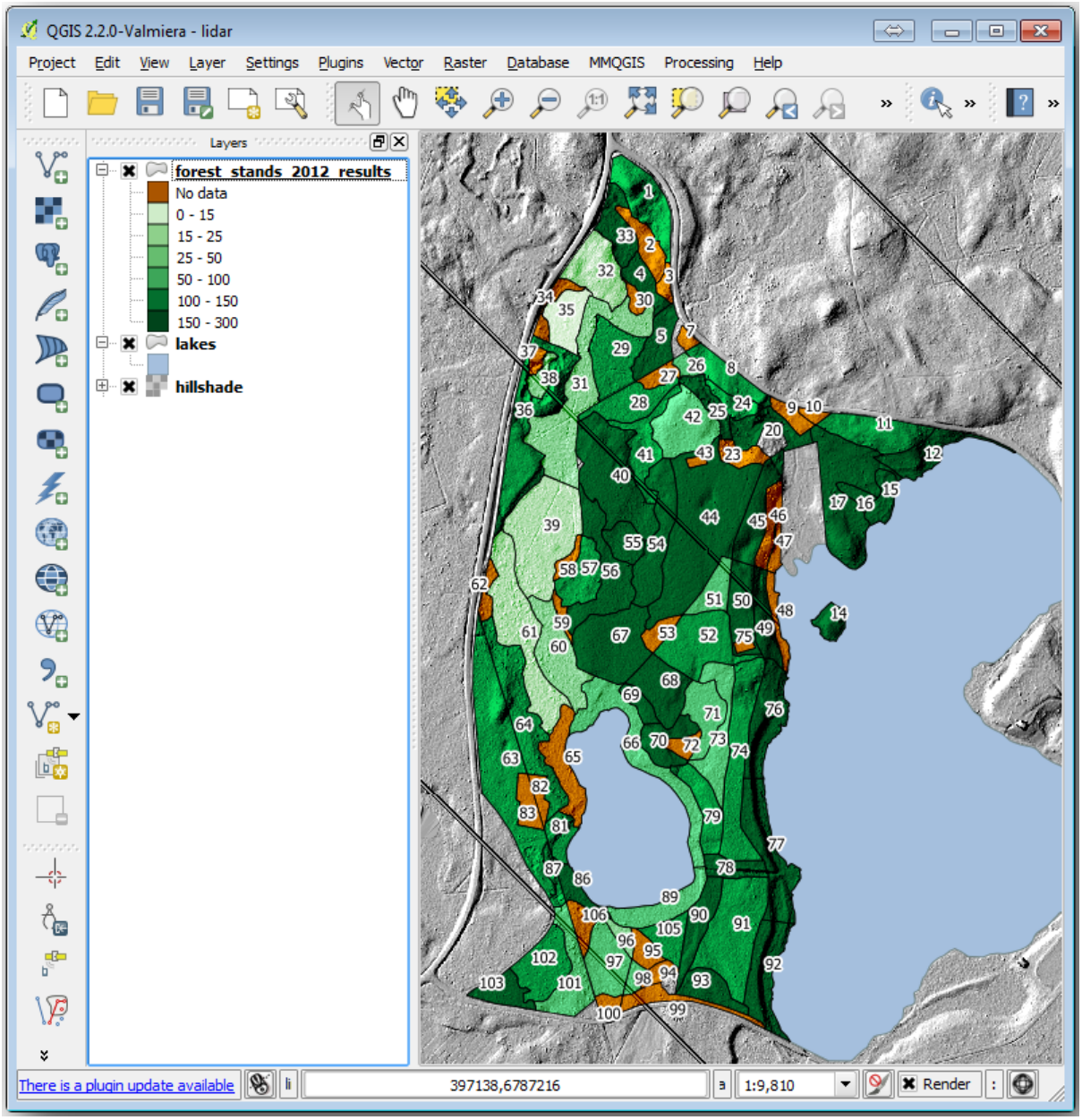
- Open the *Attribute table* and enable editing.
- Select the polygons with `NULL` values.
- Use the calculator to update the values of the `MEANVol` field to `-999` only for the selected features.
- Disable editing and save the changes.

Now you can use a saved style for this layer:

- Go to the *Symbology* tab.
- Click on *Style* ► *Load Style…*
- Select the `forest_stands_2012_results.qml` from the `exercise_data\forestry\results\` folder.
- Click *OK*.

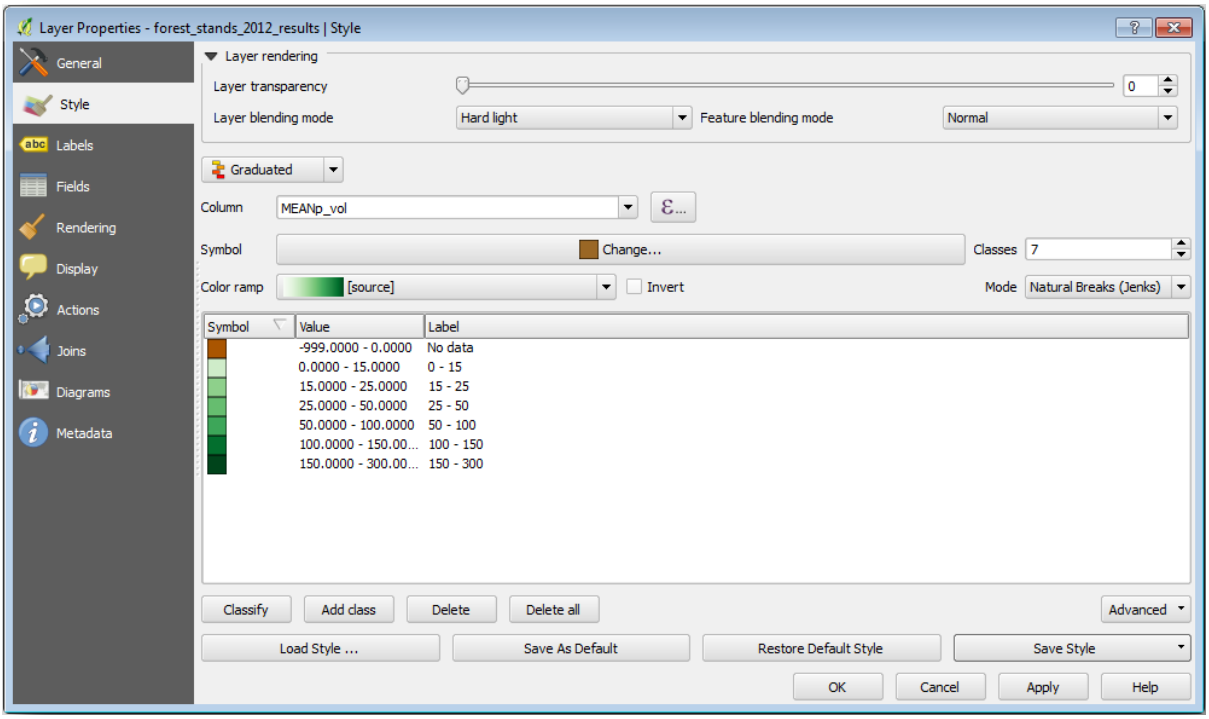


Your map will look something like this:



14.9.2 ??? Try Yourself: Try Different Blending Modes

The style you loaded:

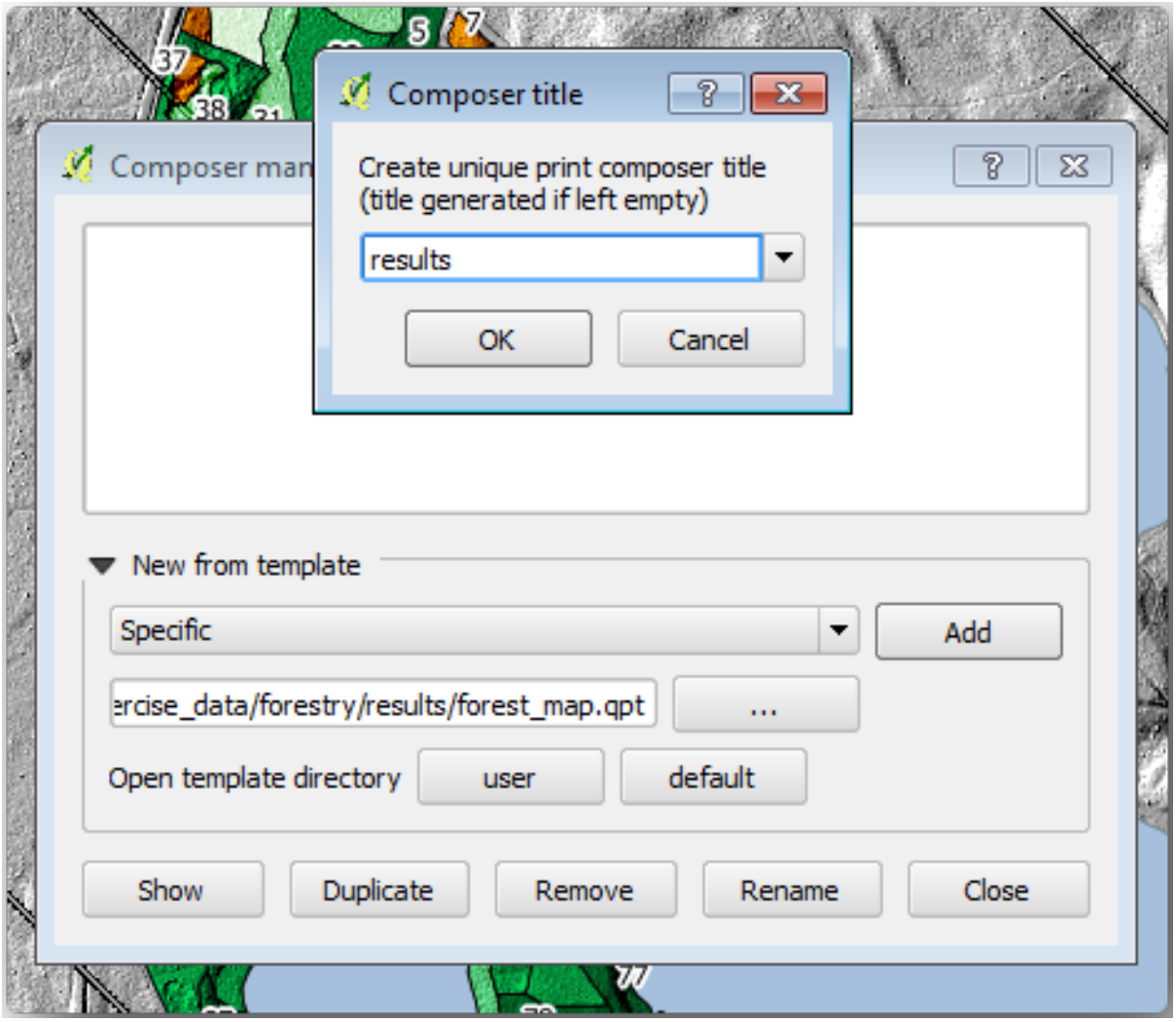


is using the `Hard light` mode for the *Layer blending mode*. Note that the different modes apply different filters combining the underlying and overlying layers, in this case the hillshade raster and your forest stands are used. You can read about these modes in the User Guide.

Try with different modes and see the differences in your map. Then choose the one you like better for your final map.

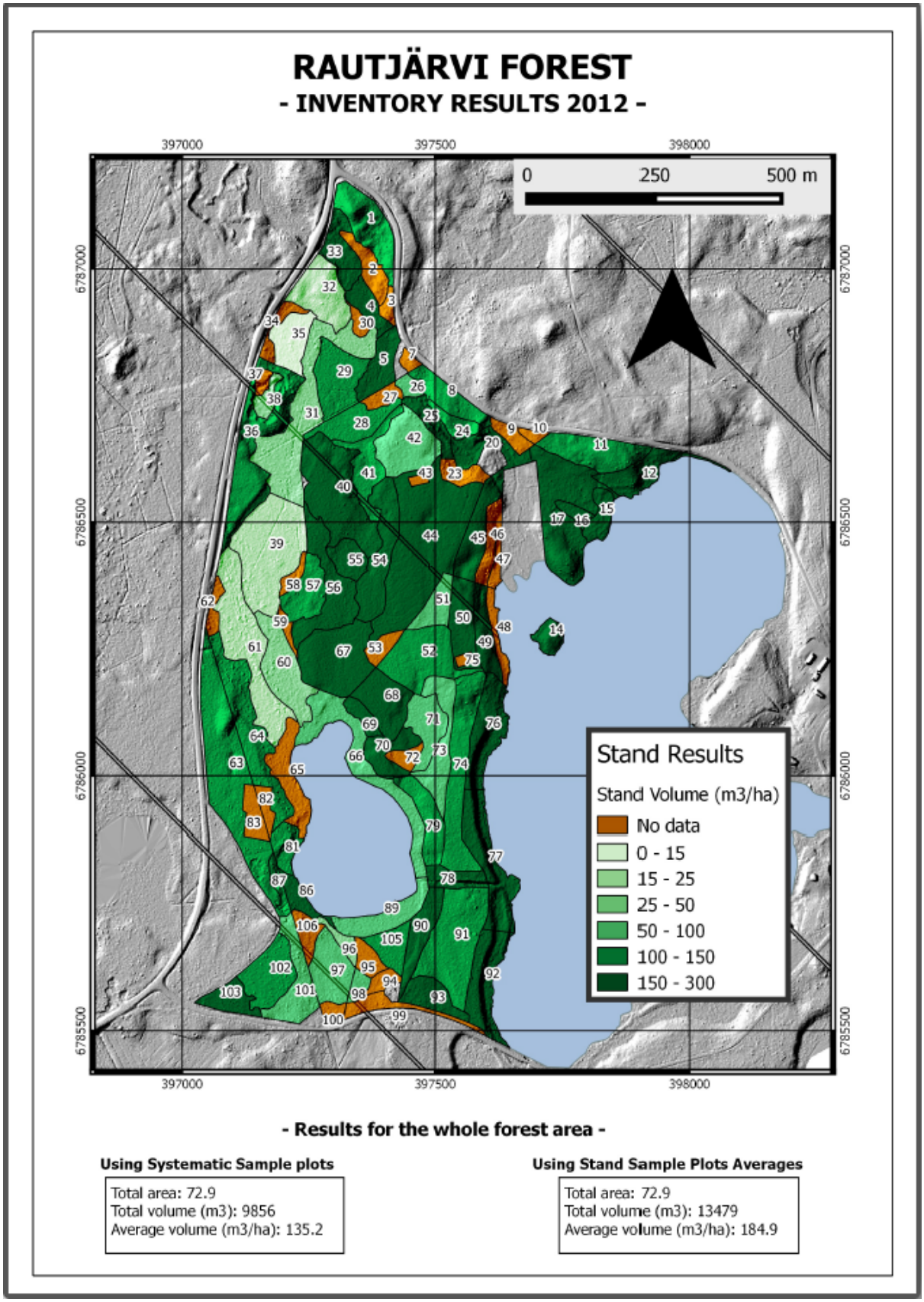
14.9.3 ??? Try Yourself: Using a Layout Template to Create the Map result

Use a template prepared in advanced to present the results. The template `forest_map.qpt` is located in the `exercise_data\forestry\results\` folder. Load it using the *Project* ► *Layout Manager…* dialog.



Open the print layout and edit the final map to get a result you are happy with.

The map template you are using will give a map similar to this one:



Save your QGIS project for future references.

14.9.4 In Conclusion

Through this module you have seen how a basic forest inventory can be planned and presented with QGIS. Many more forest analysis are possible with the variety of tools that you can access, but hopefully this manual has given you a good starting point to explore how you could achieve the specific results you need.

CAPÍTULO 15

Module: Database Concepts with PostgreSQL

Bases de dados relacionais são uma parte importante de qualquer sistema SIG. Neste módulo, irá aprender sobre conceitos de Sistema de Gestão de Bases de Dados Relacionais (RDBMS) e vai usar o PostgreSQL para criar uma nova base de dados para armazenar dados, bem como aprender sobre outras funções típicas de RDBMS.

15.1 Lesson: Introduction to Databases

Before using PostgreSQL, let's make sure of our ground by covering general database theory. You will not need to enter any of the example code; it's only there for illustration purposes.

The goal for this lesson: To understand fundamental database concepts.

15.1.1 What is a Database?

A database consists of an organized collection of data for one or more uses, typically in digital form. - *Wikipedia*

A database management system (DBMS) consists of software that operates databases, providing storage, access, security, backup and other facilities. - *Wikipedia*

15.1.2 Tabelas

In relational databases and flat file databases, a table is a set of data elements (values) that is organized using a model of vertical columns (which are identified by their name) and horizontal rows. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a candidate key. - *Wikipedia*

```
 id | name  | age
----+-------+-----
  1 | Tim   |  20
  2 | Horst |  88
(2 rows)
```

In SQL databases a table is also known as a **relation.**

15.1.3 Columns / Fields

A column is a set of data values of a particular simple type, one for each row of the table. The columns provide the structure according to which the rows are composed. The term field is often used interchangeably with column, although many consider it more correct to use field (or field value) to refer specifically to the single item that exists at the intersection between one row and one column. - *Wikipedia*

A column:

```
| name  |
+-------+
| Tim   |
| Horst |
```

A field:

```
| Horst |
```

15.1.4 Records

A record is the information stored in a table row. Each record will have a field for each of the columns in the table.

```
2 | Horst |  88  <-- one record
```

15.1.5 Datatypes

Datatypes restrict the kind of information that can be stored in a column. - *Tim and Horst*

There are many kinds of datatypes. Let's focus on the most common:

- `String` - to store free-form text data
- `Integer` - to store whole numbers
- `Real` - to store decimal numbers
- `Date` - to store Horst's birthday so no one forgets
- `Boolean` - to store simple true/false values

You can tell the database to allow you to also store nothing in a field. If there is nothing in a field, then the field content is referred to as a **"null" value**:

```
insert into person (age) values (40);

select * from person;
```

Result:

```
id | name  | age
---+-------+-----
 1 | Tim   |  20
 2 | Horst |  88
 4 |       |  40  <-- null for name
(3 rows)
```

There are many more datatypes you can use - check the PostgreSQL manual!

15.1.6 Modelling an Address Database

Let's use a simple case study to see how a database is constructed. We want to create an address database.

??? Try Yourself:

Write down the properties which make up a simple address and which we would want to store in our database.

Answer

For our theoretical address table, we might want to store the following properties:

```
House Number
Street Name
Suburb Name
City Name
Postcode
Country
```

When creating the table to represent an address object, we would create columns to represent each of these properties and we would name them with SQL-compliant and possibly shortened names:

```
house_number
street_name
suburb
city
postcode
country
```

Address Structure

The properties that describe an address are the columns. The type of information stored in each column is its datatype. In the next section we will analyse our conceptual address table to see how we can make it better!

15.1.7 Database Theory

The process of creating a database involves creating a model of the real world; taking real world concepts and representing them in the database as entities.

15.1.8 Normalisation

One of the main ideas in a database is to avoid data duplication / redundancy. The process of removing redundancy from a database is called Normalisation.

Normalization is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics - insertion, update, and deletion anomalies - that could lead to a loss of data integrity. - *Wikipedia*

There are different kinds of normalisation "forms".

Let's take a look at a simple example:

```
Table "public.people"

Column     |           Type           |                 Modifiers
-----------+--------------------------+-----------------------------------
```

(continues on next page)

(continuação da página anterior)

```
 id       | integer                | not null default
          |                        | nextval('people_id_seq'::regclass)
          |                        |
 name     | character varying(50)  |
 address  | character varying(200) | not null
 phone_no | character varying      |
Indexes:
 "people_pkey" PRIMARY KEY, btree (id)
```

```
select * from people;

id |     name      |           address            |  phone_no
---+---------------+------------------------------+-------------
 1 | Tim Sutton    | 3 Buirski Plein, Swellendam | 071 123 123
 2 | Horst Duester | 4 Avenue du Roix, Geneva     | 072 121 122
(2 rows)
```

Imagine you have many friends with the same street name or city. Every time this data is duplicated, it consumes space. Worse still, if a city name changes, you have to do a lot of work to update your database.

15.1.9 ??? Try Yourself:

Redesign the theoretical *people* table above to reduce duplication and to normalise the data structure.

You can read more about database normalisation here

Answer

The major problem with the *people* table is that there is a single address field which contains a person's entire address. Thinking about our theoretical *address* table earlier in this lesson, we know that an address is made up of many different properties. By storing all these properties in one field, we make it much harder to update and query our data. We therefore need to split the address field into the various properties. This would give us a table which has the following structure:

```
id |     name      | house_no |  street_name   |    city    |   phone_no
 --+---------------+----------+----------------+------------+-----------------
 1 | Tim Sutton    |     3    | Buirski Plein  | Swellendam | 071 123 123
 2 | Horst Duester |     4    | Avenue du Roix | Geneva     | 072 121 122
```

In the next section, you will learn about Foreign Key relationships which could be used in this example to further improve our database's structure.

15.1.10 Indexes

A database index is a data structure that improves the speed of data retrieval operations on a database table. - *Wikipedia*

Imagine you are reading a textbook and looking for the explanation of a concept - and the textbook has no index! You will have to start reading at one cover and work your way through the entire book until you find the information you need. The index at the back of a book helps you to jump quickly to the page with the relevant information:

```
create index person_name_idx on people (name);
```

Now searches on name will be faster:

```
Table "public.people"

Column    |          Type          |                Modifiers
----------+------------------------+-----------------------------------------
 id       | integer                | not null default
          |                        | nextval('people_id_seq'::regclass)
          |                        |
 name     | character varying(50)  |
 address  | character varying(200) | not null
 phone_no | character varying      |
Indexes:
 "people_pkey" PRIMARY KEY, btree (id)
 "person_name_idx" btree (name)
```

15.1.11 Sequences

A sequence is a unique number generator. It is normally used to create a unique identifier for a column in a table.

In this example, id is a sequence - the number is incremented each time a record is added to the table:

```
id |     name     |           address           |  phone_no
---+--------------+-----------------------------+-------------
 1 | Tim Sutton   | 3 Buirski Plein, Swellendam | 071 123 123
 2 | Horst Duster | 4 Avenue du Roix, Geneva    | 072 121 122
```

15.1.12 Entity Relationship Diagramming

In a normalised database, you typically have many relations (tables). The entity-relationship diagram (ER Diagram) is used to design the logical dependencies between the relations. Consider our non-normalised *people* table from earlier in the lesson:

```
select * from people;

 id |     name     |           address           |  phone_no
----+--------------+-----------------------------+-------------
 1  | Tim Sutton   | 3 Buirski Plein, Swellendam | 071 123 123
 2  | Horst Duster | 4 Avenue du Roix, Geneva    | 072 121 122
(2 rows)
```

With a little work we can split it into two tables, removing the need to repeat the street name for individuals who live in the same street:

```
select * from streets;

 id |     name
----+--------------
 1  | Plein Street
(1 row)
```

and:

```
select * from people;

 id |     name     | house_no | street_id |  phone_no
----+--------------+----------+-----------+-------------
  1 | Horst Duster |        4 |         1 | 072 121 122
(1 row)
```

We can then link the two tables using the "keys" `streets.id` and `people.streets_id`.

If we draw an ER Diagram for these two tables it would look something like this:

The ER Diagram helps us to express "one to many" relationships. In this case the arrow symbol show that one street can have many people living on it.

??? Try Yourself:

Our *people* model still has some normalisation issues - try to see if you can normalise it further and show your thoughts by means of an ER Diagram.

Answer

Our *people* table currently looks like this:

```
id |     name     | house_no | street_id |  phone_no
---+--------------+----------+-----------+-------------
 1 | Horst Duster |        4 |         1 | 072 121 122
```

The *street_id* column represents a "one to many" relationship between the people object and the related street object, which is in the *streets* table.

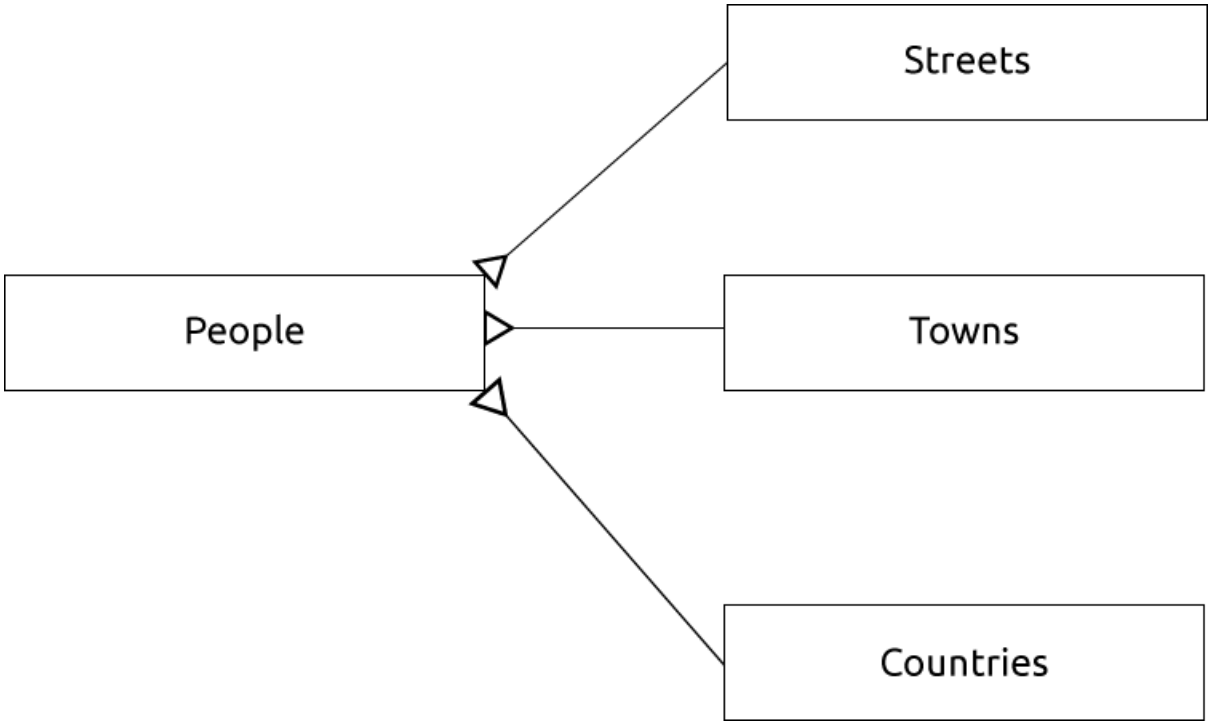
One way to further normalise the table is to split the name field into *first_name* and *last_name*:

```
id | first_name | last_name  | house_no | street_id |  phone_no
---+------------+------------+----------+-----------+------------
 1 |    Horst   |   Duster   |     4    |     1     | 072 121 122
```

We can also create separate tables for the town or city name and country, linking them to our *people* table via "one to many" relationships:

```
id | first_name | last_name | house_no | street_id | town_id | country_id
---+------------+-----------+----------+-----------+---------+------------
 1 |    Horst   |   Duster  |     4    |     1     |    2    |     1
```

An ER Diagram to represent this would look like this:



15.1.13 Constraints, Primary Keys and Foreign Keys

A database constraint is used to ensure that data in a relation matches the modeller's view of how that data should be stored. For example a constraint on your postal code could ensure that the number falls between `1000` and `9999`.

A Primary key is one or more field values that make a record unique. Usually the primary key is called id and is a sequence.

A Foreign key is used to refer to a unique record on another table (using that other table's primary key).

In ER Diagramming, the linkage between tables is normally based on Foreign keys linking to Primary keys.

If we look at our people example, the table definition shows that the street column is a foreign key that references the primary key on the streets table:

```
Table "public.people"

 Column    |         Type          |  Modifiers
-----------+-----------------------+-------------------------------------
 id        | integer               | not null default
           |                       | nextval('people_id_seq'::regclass)
 name      | character varying(50) |
 house_no  | integer               | not null
 street_id | integer               | not null
 phone_no  | character varying     |
Indexes:
"people_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
"people_street_id_fkey" FOREIGN KEY (street_id) REFERENCES streets(id)
```

15.1.14 Transactions

When adding, changing, or deleting data in a database, it is always important that the database is left in a good state if something goes wrong. Most databases provide a feature called transaction support. Transactions allow you to create a rollback position that you can return to if your modifications to the database did not run as planned.

Take a scenario where you have an accounting system. You need to transfer funds from one account and add them to another. The sequence of steps would go like this:

- remove R20 from Joe
- add R20 to Anne

If something goes wrong during the process (e.g. power failure), the transaction will be rolled back.

15.1.15 In Conclusion

Databases allow you to manage data in a structured way using simple code structures.

15.1.16 What's Next?

Now that we've looked at how databases work in theory, let's create a new database to implement the theory we've covered.

15.2 Lesson: Implementing the Data Model

Now that we've covered all the theory, let's create a new database. This database will be used for our exercises for the lessons that will follow afterwards.

The goal for this lesson: To install the required software and use it to implement our example database.

15.2.1 Install PostgreSQL

Nota: You can find PostgreSQL packages and installation instructions for your operating system at https://www.postgresql.org/download/. Please note that the documentation will assume users are running QGIS under Ubuntu.

Under Ubuntu:

```
sudo apt install postgresql-9.1
```

You should get a message like this:

```
[sudo] password for qgis:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
postgresql-client-9.1 postgresql-client-common postgresql-common
Suggested packages:
oidentd ident-server postgresql-doc-9.1
The following NEW packages will be installed:
postgresql-9.1 postgresql-client-9.1 postgresql-client-common postgresql-common
0 upgraded, 4 newly installed, 0 to remove and 5 not upgraded.
Need to get 5,012kB of archives.
```

(continues on next page)

(continuação da página anterior)

```
After this operation, 19.0MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Press `Y` and `Enter` and wait for the download and installation to finish.

15.2.2 Ajuda

PostgreSQL has very good online documentation.

15.2.3 Create a database user

Under Ubuntu:

After the installation is complete, run this command to become the postgres user and then create a new database user:

```
sudo su - postgres
```

Type in your normal log in password when prompted (you need to have sudo rights).

Now, at the postgres user's bash prompt, create the database user. Make sure the user name matches your unix login name: it will make your life much easier, as postgres will automatically authenticate you when you are logged in as that user:

```
createuser -d -E -i -l -P -r -s qgis
```

Enter a password when prompted. You should use a different password to your login password.

What do those options mean?

```
-d, --createdb     role can create new databases
-E, --encrypted    encrypt stored password
-i, --inherit      role inherits privileges of roles it is a member of (default)
-l, --login        role can login (default)
-P, --pwprompt     assign a password to new role
-r, --createrole   role can create new roles
-s, --superuser    role will be superuser
```

Now you should leave the postgres user's bash shell environment by typing:

```
exit
```

15.2.4 Verify the new account

```
psql -l
```

Should return something like this:

```
Name      |  Owner   | Encoding | Collation  |   Ctype    |
----------+----------+----------+------------+------------+
postgres  | postgres | UTF8     | en_ZA.utf8 | en_ZA.utf8 |
template0 | postgres | UTF8     | en_ZA.utf8 | en_ZA.utf8 |
template1 | postgres | UTF8     | en_ZA.utf8 | en_ZA.utf8 |
(3 rows)
```

Type `Q` to exit.

15.2.5 Create a database

The `createdb` command is used to create a new database. It should be run from the bash shell prompt:

```
createdb address -O qgis
```

You can verify the existence of your new database by using this command:

```
psql -l
```

Which should return something like this:

```
Name      |  Owner   | Encoding | Collation  |   Ctype    |   Access privileges
----------+----------+----------+------------+------------+-----------------------
address   | qgis     | UTF8     | en_ZA.utf8 | en_ZA.utf8 |
postgres  | postgres | UTF8     | en_ZA.utf8 | en_ZA.utf8 |
template0 | postgres | UTF8     | en_ZA.utf8 | en_ZA.utf8 | =c/postgres:␣
↪postgres=CTc/postgres
template1 | postgres | UTF8     | en_ZA.utf8 | en_ZA.utf8 | =c/postgres:␣
↪postgres=CTc/postgres
(4 rows)
```

Type `Q` to exit.

15.2.6 Starting a database shell session

You can connect to your database easily like this:

```
psql address
```

To exit out of the psql database shell, type:

```
\q
```

For help in using the shell, type:

```
\?
```

For help in using sql commands, type:

```
\help
```

To get help on a specific command, type (for example):

```
\help create table
```

See also the Psql cheat sheet.

15.2.7 Make Tables in SQL

Let's start making some tables! We will use our ER Diagram as a guide. First, connect to the address db:

```
psql address
```

Then create a `streets` table:

```
create table streets (id serial not null primary key, name varchar(50));
```

`serial` and `varchar` are **data types**. `serial` tells PostgreSQL to start an integer sequence (auto-number) to populate the `id` automatically for every new record. `varchar(50)` tells PostgreSQL to create a character field of 50 characters in length.

You will notice that the command ends with a `;` - all SQL commands should be terminated this way. When you press `Enter`, psql will report something like this:

```
NOTICE:  CREATE TABLE will create implicit sequence "streets_id_seq"
         for serial column "streets.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
         "streets_pkey" for table "streets"
CREATE TABLE
```

That means your table was created successfully, with a primary key `streets_pkey` using `streets.id`.

Note: If you hit return without entering a `;`, then you will get a prompt like this: `address-#`. This is because PG is expecting you to enter more. Enter `;` to run your command.

To view your table schema, you can do this:

```
\d streets
```

Which should show something like this:

```
Table "public.streets"
Column  |          Type          |             Modifiers
--------+------------------------+-----------------------------------
 id     | integer                | not null default
        |                        | nextval('streets_id_seq'::regclass)
 name   | character varying(50)  |
Indexes:
  "streets_pkey" PRIMARY KEY, btree (id)
```

To view your table contents, you can do this:

```
select * from streets;
```

Which should show something like this:

```
id | name
---+------
(0 rows)
```

As you can see, our table is currently empty.

Try Yourself: ???

Use the approach shown above to make a table called people:

Add fields such as phone number, home address, name, etc. (these aren't all valid names: change them to make them valid). Make sure you give the table an ID column with the same data-type as above.

Answer

The SQL required to create the correct people table is:

```
create table people (id serial not null primary key,
                     name varchar(50),
                     house_no int not null,
                     street_id int not null,
                     phone_no varchar null );
```

The schema for the table (enter `\\d people`) looks like this:

```
Table "public.people"

Column     |         Type          |                       Modifiers
-----------+-----------------------+--------------------------------------
id         | integer               | not null default
           |                       | nextval('people_id_seq'::regclass)
name       | character varying(50) |
house_no   | integer               | not null
street_id  | integer               | not null
phone_no   | character varying     |
Indexes:
  "people_pkey" PRIMARY KEY, btree (id)
```

For illustration purposes, we have purposely omitted the fkey constraint.

15.2.8 Create Keys in SQL

The problem with our solution above is that the database doesn't know that people and streets have a logical relationship. To express this relationship, we have to define a foreign key that points to the primary key of the streets table.

There are two ways to do this:

- Add the key after the table has been created
- Define the key at time of table creation

Our table has already been created, so let's do it the first way:

```
alter table people
  add constraint people_streets_fk foreign key (street_id) references streets(id);
```

That tells the `people` table that its `street_id` fields must match a valid street `id` from the `streets` table.

The more usual way to create a constraint is to do it when you create the table:

```
create table people (id serial not null primary key,
                     name varchar(50),
                     house_no int not null,
                     street_id int references streets(id) not null,
                     phone_no varchar null);

\d people
```

After adding the constraint, our table schema looks like this now:

```
Table "public.people"

  Column    |         Type          |            Modifiers
------------+-----------------------+---------------------------------
 id         | integer               | not null default
            |                       | nextval('people_id_seq'::regclass)
 name       | character varying(50) |
```

(continues on next page)

(continuação da página anterior)

```
 house_no   | integer                | not null
 street_id  | integer                | not null
 phone_no   | character varying      |
Indexes:
  "people_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
  "people_streets_fk" FOREIGN KEY (id) REFERENCES streets(id)
```

15.2.9 Create Indexes in SQL

We want lightning fast searches on peoples names. To provide for this, we can create an index on the name column of our people table:

```
create index people_name_idx on people(name);

\d people
```

Which results in:

```
Table "public.people"

  Column    |         Type           |                       Modifiers
------------+------------------------+-------------------------------------
 id         | integer                | not null default nextval
            |                        | ('people_id_seq'::regclass)
 name       | character varying(50)  |
 house_no   | integer                | not null
 street_id  | integer                | not null
 phone_no   | character varying      |
Indexes:
 "people_pkey" PRIMARY KEY, btree (id)
 "people_name_idx" btree (name)     <-- new index added!
Foreign-key constraints:
 "people_streets_fk" FOREIGN KEY (id) REFERENCES streets(id)
```

15.2.10 Dropping Tables in SQL

If you want to get rid of a table you can use the `drop` command:

```
drop table streets;
```

In our current example, the above command would not work. Why not?

Answer

The reason the DROP command would not work in this case is because the *people* table has a Foreign Key constraint to the *streets* table. This means that dropping (or deleting) the *streets* table would leave the *people* table with references to non-existent *streets* data.

It is possible to "force" the *streets* table to be deleted by using the `CASCADE` command, but this would also delete the *people* and any other table which had a relationship to the *streets* table. Use with caution!

If you used the same `drop table` command on the *people* table, it would be successful:

```
drop table people;
```

Nota: If you actually did enter that command and dropped the `people` table, now would be a good time to rebuild it, as you will need it in the next exercises.

15.2.11 A word on pgAdmin III

We are showing you the SQL commands from the *psql* prompt because it's a very useful way to learn about databases. However, there are quicker and easier ways to do a lot of what we are showing you. Install pgAdmin III and you can create, drop, alter etc tables using "point and click" operations in a GUI.

Under Ubuntu, you can install it like this:

```
sudo apt install pgadmin3
```

pgAdmin III will be covered in more detail in another module.

15.2.12 In Conclusion

You have now seen how to create a brand new database, starting completely from scratch.

15.2.13 What's Next?

Next you'll learn how to use the DBMS to add new data.

15.3 Lesson: Adding Data to the Model

The models we've created will now need to be populated with the data they're intended to contain.

The goal for this lesson: To learn how to insert new data into the database models.

15.3.1 Insert statement

How do you add data to a table? The sql `INSERT` statement provides the functionality for this:

```
insert into streets (name) values ('High street');
```

A couple of things to note:

- After the table name (`streets`), you list the column names that you will be populating (in this case only the `name` column).
- After the `values` keyword, place the list of field values.
- Strings should be quoted using single quotes.
- Note that we did not insert a value for the `id` column; this is because it is a sequence and will be auto-generated.
- If you do manually set the `id`, you may cause serious problems with the integrity of your database.

You should see `INSERT 0 1` if it is successful.

You can see the result of your insert action by selecting all the data in the table:

```
select * from streets;
```

Result:

```
select * from streets;
 id |     name
----+-------------
  1 | High street
(1 row)
```

Try Yourself: ???

Use the `INSERT` command to add a new street to the `streets` table.

Answer

The SQL command you should use looks like this (you can replace the street name with a name of your choice):

```
insert into streets (name) values ('Low Road');
```

15.3.2 Sequencing Data Addition According to Constraints

15.3.3 Try Yourself: ???

Try to add a person object to the `people` table with the following details:

```
Name: Joe Smith
House Number: 55
Street: Main Street
Phone: 072 882 33 21
```

Nota: Recall that in this example, we defined phone numbers as strings, not integers.

At this point, you should have an error report if you try to do this without first creating a record for Main Street in the `streets` table.

You should have also noticed that:

- You can't add the street using its name
- You can't add a street using a street `id` before first creating the street record on the streets table

Remember that our two tables are linked via a Primary/Foreign Key pair. This means that no valid person can be created without there also being a valid corresponding street record.

Using the above knowledge, add the new person to the database.

Answer

Here is the correct SQL statement:

```
insert into streets (name) values('Main Road');
insert into people (name,house_no, street_id, phone_no)
  values ('Joe Smith',55,2,'072 882 33 21');
```

If you look at the streets table again (using a select statement as before), you'll see that the *id* for the `Main Road` entry is *2*.

That's why we could merely enter the number `2` above. Even though we're not seeing `Main Road` written out fully in the entry above, the database will be able to associate that with the *street_id* value of *2*.

If you have already added a new street object, you might find that the new `Main Road` has an *id* of *3* not *2*.

15.3.4 Select data

We have already shown you the syntax for selecting records. Let's look at a few more examples:

```
select name from streets;
```

```
select * from streets;
```

```
select * from streets where name='Main Road';
```

In later sessions we will go into more detail on how to select and filter data.

15.3.5 Update data

What if you want to make a change to some existing data? For example, a street name is changed:

```
update streets set name='New Main Road' where name='Main Road';
```

Be very careful using such update statements - if more than one record matches your `WHERE` clause, they will all be updated!

A better solution is to use the primary key of the table to reference the record to be changed:

```
update streets set name='New Main Road' where id=2;
```

It should return `UPDATE 1`.

Nota: the `WHERE` statement criteria are case sensitive `Main Road` is not the same as `Main road`

15.3.6 Delete Data

In order to delete an object from a table, use the `DELETE` command:

```
delete from people where name = 'Joe Smith';
```

Let's look at our people table now:

```
address=# select * from people;

 id | name | house_no | street_id | phone_no
----+------+----------+-----------+----------
(0 rows)
```

15.3.7 Try Yourself: ???

Use the skills you have learned to add some new friends to your database:

```
      name       | house_no | street_id |   phone_no
-----------------+----------+-----------+--------------
Joe Bloggs       |        3 |         2 | 072 887 23 45
Jane Smith       |       55 |         3 | 072 837 33 35
Roger Jones      |       33 |         1 | 072 832 31 38
Sally Norman     |       83 |         1 | 072 932 31 32
```

15.3.8 In Conclusion

Now you know how to add new data to the existing models you created previously. Remember that if you want to add new kinds of data, you may want to modify and/or create new models to contain that data.

15.3.9 What's Next?

Now that you've added some data, you'll learn how to use queries to access this data in various ways.

15.4 Lesson: Queries

When you write a `SELECT ...` command it is commonly known as a query - you are interrogating the database for information.

The goal of this lesson: To learn how to create queries that will return useful information.

Nota: If you did not do so in the previous lesson, add the following people objects to your `people` table. If you receive any errors related to foreign key constraints, you will need to add the "Main Road" object to your streets table first

---

```
insert into people (name,house_no, street_id, phone_no)
          values ('Joe Bloggs',3,2,'072 887 23 45');
insert into people (name,house_no, street_id, phone_no)
          values ('Jane Smith',55,3,'072 837 33 35');
insert into people (name,house_no, street_id, phone_no)
          values ('Roger Jones',33,1,'072 832 31 38');
insert into people (name,house_no, street_id, phone_no)
          values ('Sally Norman',83,1,'072 932 31 32');
```

15.4.1 Ordering Results

Let's retrieve a list of people ordered by their house numbers:

```
select name, house_no from people order by house_no;
```

Result:

```
     name      | house_no
---------------+----------
 Joe Bloggs    |        3
 Roger Jones   |       33
```

(continues on next page)

(continuação da página anterior)

```
 Jane Smith   |       55
 Sally Norman |       83
(4 rows)
```

You can sort the results by the values of more than one column:

```
select name, house_no from people order by name, house_no;
```

Result:

```
     name     | house_no
--------------+----------
 Jane Smith   |       55
 Joe Bloggs   |        3
 Roger Jones  |       33
 Sally Norman |       83
(4 rows)
```

15.4.2 Filtering

Often you won't want to see every single record in the database - especially if there are thousands of records and you are only interested in seeing one or two.

Here is an example of a numerical filter which only returns objects whose `house_no` is less than 50:

```
select name, house_no from people where house_no < 50;

     name     | house_no
  ------------+----------
   Joe Bloggs  |        3
   Roger Jones |       33
  (2 rows)
```

You can combine filters (defined using the `WHERE` clause) with sorting (defined using the `ORDER BY` clause):

```
select name, house_no from people where house_no < 50 order by house_no;

     name     | house_no
  ------------+----------
   Joe Bloggs  |        3
   Roger Jones |       33
  (2 rows)
```

You can also filter based on text data:

```
select name, house_no from people where name like '%s%';

     name     | house_no
  ------------+----------
   Joe Bloggs  |        3
   Roger Jones |       33
  (2 rows)
```

Here we used the `LIKE` clause to find all names with an `s` in them. You'll notice that this query is case-sensitive, so the `Sally Norman` entry has not been returned.

If you want to search for a string of letters regardless of case, you can do a case in-sensitive search using the `ILIKE` clause:

```
select name, house_no from people where name ilike '%r%';

      name      | house_no
  --------------+----------
   Roger Jones  |       33
   Sally Norman |       83
  (2 rows)
```

That query returned every **people** object with an `r` or `R` in their name.

15.4.3 Joins

What if you want to see the person's details and their street's name instead of the ID? In order to do that, you need to join the two tables together in a single query. Lets look at an example:

```
select people.name, house_no, streets.name
from people,streets
where people.street_id=streets.id;
```

Nota: With joins, you will always state the two tables the information is coming from, in this case people and streets. You also need to specify which two keys must match (foreign key & primary key). If you don't specify that, you will get a list of all possible combinations of people and streets, but no way to know who actually lives on which street!

Here is what the correct output will look like:

```
     name      | house_no |    name
---------------+----------+-------------
 Joe Bloggs    |        3 | Low Street
 Roger Jones   |       33 | High street
 Sally Norman  |       83 | High street
 Jane Smith    |       55 | Main Road
(4 rows)
```

We will revisit joins as we create more complex queries later. Just remember they provide a simple way to combine the information from two or more tables.

15.4.4 Sub-Select

Sub-selections allow you to select objects from one table based on the data from another table which is linked via a foreign key relationship. In our case, we want to find people who live on a specific street.

First, let's do a little tweaking of our data:

```
insert into streets (name) values('QGIS Road');
insert into streets (name) values('OGR Corner');
insert into streets (name) values('Goodle Square');
update people set street_id = 2 where id=2;
update people set street_id = 3 where id=3;
```

Let's take a quick look at our data after those changes: we can reuse our query from the previous section:

```
select people.name, house_no, streets.name
from people,streets
where people.street_id=streets.id;
```

Result:

```
     name      | house_no |     name
---------------+----------+-------------
 Roger Jones   |       33 | High street
 Sally Norman  |       83 | High street
 Jane Smith    |       55 | Main Road
 Joe Bloggs    |        3 | Low Street
(4 rows)
```

Now let's show you a sub-selection on this data. We want to show only people who live in `street_id` number `1`:

```
select people.name
from people, (
    select *
    from streets
    where id=1
  ) as streets_subset
where people.street_id = streets_subset.id;
```

Result:

```
     name
--------------
 Roger Jones
 Sally Norman
(2 rows)
```

Although this is a very simple example and unnecessary with our small data-sets, it illustrates how useful and important sub-selections can be when querying large and complex data-sets.

15.4.5 Aggregate Queries

One of the powerful features of a database is its ability to summarise the data in its tables. These summaries are called aggregate queries. Here is a typical example which tells us how many people objects are in our people table:

```
select count(*) from people;
```

Result:

```
 count
-------
     4
(1 row)
```

If we want the counts to be summarised by street name we can do this:

```
select count(name), street_id
from people
group by street_id;
```

Result:

```
 count | street_id
-------+-----------
     2 |         1
     1 |         3
     1 |         2
(3 rows)
```

Nota: Because we have not used an `ORDER BY` clause, the order of your results may not match what is shown here.

Try Yourself: ???

Summarise the people by street name and show the actual street names instead of the street_ids.

Answer

Here is the correct SQL statement you should use:

```
select count(people.name), streets.name
from people, streets
where people.street_id=streets.id
group by streets.name;
```

Result:

```
count |    name
------+-------------
    1 | Low Street
    2 | High street
    1 | Main Road
(3 rows)
```

You will notice that we have prefixed field names with table names (e.g. people.name and streets.name). This needs to be done whenever the field name is ambiguous (i.e. not unique across all tables in the database).

15.4.6 In Conclusion

You've seen how to use queries to return the data in your database in a way that allows you to extract useful information from it.

15.4.7 What's Next?

Next you'll see how to create views from the queries that you've written.

15.5 Lesson: Views

When you write a query, you need to spend a lot of time and effort formulating it. With views, you can save the definition of an SQL query in a reusable "virtual table".

The goal for this lesson: To save a query as a view.

15.5.1 Creating a View

You can treat a view just like a table, but its data is sourced from a query. Let's make a simple view based on the above:

```
create view roads_count_v as
  select count(people.name), streets.name
  from people, streets where people.street_id=streets.id
  group by people.street_id, streets.name;
```

As you can see the only change is the `create view roads_count_v as` part at the beginning. We can now select data from that view:

```
select * from roads_count_v;
```

Result:

```
 count |    name
-------+-------------
     1 | Main Road
     2 | High street
     1 | Low Street
(3 rows)
```

15.5.2 Modifying a View

A view is not fixed, and it contains no "real data". This means you can easily change it without impacting on any data in your database:

```
CREATE OR REPLACE VIEW roads_count_v AS
  SELECT count(people.name), streets.name
  FROM people, streets WHERE people.street_id=streets.id
  GROUP BY people.street_id, streets.name
  ORDER BY streets.name;
```

(This example also shows the best practice convention of using UPPER CASE for all SQL keywords.)

You will see that we have added an `ORDER BY` clause so that our view rows are nicely sorted:

```
select * from roads_count_v;

 count |    name
-------+-------------
     2 | High street
     1 | Low Street
     1 | Main Road
(3 rows)
```

15.5.3 Dropping a View

If you no longer need a view, you can delete it like this:

```
drop view roads_count_v;
```

15.5.4 In Conclusion

Using views, you can save a query and access its results as if it were a table.

15.5.5 What's Next?

Sometimes, when changing data, you want your changes to have effects elsewhere in the database. The next lesson will show you how to do this.

15.6 Lesson: Rules

Rules allow the «query tree» of an incoming query to be rewritten. One common usage is to implement views, including updatable view. - *Wikipedia*

The goal for this lesson: To learn how to create new rules for the database.

15.6.1 Creating a logging rule

Say you want to log every change of phone_no in your people table in to a people_log table. So you set up a new table:

```
create table people_log (name text, time timestamp default NOW());
```

In the next step, create a rule that logs every change of a phone_no in the people table into the people_log table:

```
create rule people_log as on update to people
  where NEW.phone_no <> OLD.phone_no
  do insert into people_log values (OLD.name);
```

To test that the rule works, let's modify a phone number:

```
update people set phone_no = '082 555 1234' where id = 2;
```

Check that the `people` table was updated correctly:

```
select * from people where id=2;

 id |    name     | house_no | street_id |   phone_no
----+-------------+----------+-----------+--------------
  2 | Joe Bloggs |        3 |         2 | 082 555 1234
(1 row)
```

Now, thanks to the rule we created, the `people_log` table will look like this:

```
select * from people_log;

    name     |            time
-------------+----------------------------
 Joe Bloggs | 2014-01-11 14:15:11.953141
(1 row)
```

Nota: The value of the `time` field will depend on the current date and time.

15.6.2 In Conclusion

Rules allow you to automatically add or change data in your database to reflect changes in other parts of the database.

15.6.3 What's Next?

The next module will introduce you to Spatial Database using PostGIS, which takes these database concepts and applies them to GIS data.

CAPÍTULO 16

Module: Spatial Database Concepts with PostGIS

Spatial Databases allow the storage of the geometries of records inside a Database as well as providing functionality for querying and retrieving the records using these Geometries. In this module we will use PostGIS, an extension to PostgreSQL, to learn how to setup a spatial database, import data into the database and make use of the geographic functions that PostGIS offers.

While working through this section, you may want to keep a copy of the **PostGIS cheat sheet** available from Boston GIS user group. Another useful resource is the online PostGIS documentation.

There are also some more extensive tutorials on PostGIS and Spatial Databases created by Boundless that are now hosted on the PostGIS's website:

- Introduction to PostGIS
- PostGIS Database Tips and Tricks

See also PostGIS In Action.

16.1 Lesson: PostGIS Setup

Setting up PostGIS functions will allow you to access spatial functions from within PostgreSQL.

The goal for this lesson: To install spatial functions and briefly demo their effects.

Nota: We will assume the use of PostGIS version 2.1 or newer in this exercise. The installation and database configuration are different for older versions, but the rest of this material in this module will still work. Consult the documentation for your platform for help with installation and database configuration.

16.1.1 Installing under Ubuntu

Postgis is easily installed from apt.

```
$ sudo apt install postgresql
$ sudo apt install postgis
```

Really, it's that easy…

Nota: The exact versions that will be installed depend on which version of Ubuntu you are using and which repositories you have configured. After installing you can check the version by issuing a `select PostGIS_full_version();` query with psql or another tool.

To install a specific version (eg, PostgreSQL version 13 and PostGIS 3), you can use the following commands.

```
$ sudo apt install wget ca-certificates
$ sudo lsb_release -a
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-
↪key add -
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-
↪pgdg main" >> /etc/apt/sources.list.d/pgdg.list'
$ sudo apt-get update
$ sudo apt install postgis postgresql-13-postgis-3
```

16.1.2 Installing under Windows

Installing on Windows can be done from binary packages using a normal Windows installation dialogs.

First Visit the download page. Then follow this guide.

More information about installing on Windows can be found on the PostGIS website.

16.1.3 Installing on Other Platforms

The PostGIS website download has information about installing on other platforms including macOS and on other Linux distributions

16.1.4 Configuring Databases to use PostGIS

Once PostGIS is installed, you will need to configure your database to use the extensions. If you have installed PostGIS version > 2.0, this is as simple as issuing the following command with psql using the address database from our previous exercise.

```
$ psql -d address -c "CREATE EXTENSION postgis;"
```

Nota: Depending on your version, you could find more instructions on how to spatially enable a database at https://postgis.net/docs/postgis_administration.html#create_spatial_db.

16.1.5 Looking at the installed PostGIS functions

PostGIS can be thought of as a collection of in-database functions that extend the core capabilities of PostgreSQL so that it can deal with spatial data. By "deal with", we mean store, retrieve, query and manipulate. In order to do this, a number of functions are installed into the database.

Our PostgreSQL `address` database is now geospatially enabled, thanks to PostGIS. We are going to delve a lot deeper into this in the coming sections, but let's give you a quick little taster. Let's say we want to create a point from text. First we use the psql command to find functions relating to point. If you are not already connected to the `address` database, do so now. Then run:

```
\df *point*
```

This is the command we're looking for: `st_pointfromtext`. To page through the list, use the down arrow, then press `Q` to quit back to the psql shell.

Try running this command:

```
select st_pointfromtext('POINT(1 1)');
```

Result:

```
st_pointfromtext
----------------------------------------------
0101000000000000000000F03F000000000000F03F
(1 row)
```

Three things to note:

- We defined a point at position 1,1 (EPSG:4326 is assumed) using `POINT(1 1)`,
- We ran an sql statement, but not on any table, just on data entered from the SQL prompt,
- The resulting row does not make much sense.

The resulting row is in the OGC format called "Well Known Binary" (WKB). We will look at this format in detail in the next section.

To get the results back as text, we can do a quick scan through the function list for something that returns text:

```
\df *text
```

The query we're looking for now is `st_astext`. Let's combine it with the previous query:

```
select st_astext(st_pointfromtext('POINT(1 1)'));
```

Result:

```
 st_astext
------------
  POINT(1 1)
   (1 row)
```

Here, we entered the string `POINT(1,1)`, turned it into a point using `st_pointfromtext()`, and turned it back into a human-readable form with `st_astext()`, which gave us back our original string.

One last example before we really get into the detail of using PostGIS:

```
select st_astext(st_buffer(st_pointfromtext('POINT(1 1)'),1.0));
```

What did that do? It created a buffer of 1 degree around our point, and returned the result as text.

16.1.6 Spatial Reference Systems

In addition to the PostGIS functions, the extension contains a collection of spatial reference system (SRS) definitions as defined by the European Petroleum Survey Group (EPSG). These are used during operations such as coordinate reference system (CRS) conversions.

We can inspect these SRS definitions in our database as they are stored in normal database tables.

First, let's look at the schema of the table by entering the following command in the psql prompt:

```
\d spatial_ref_sys
```

The result should be this:

```
Table "public.spatial_ref_sys"
   Column    |          Type           | Modifiers
 ------------+-------------------------+-----------
  srid       | integer                 | not null
  auth_name  | character varying(256)  |
  auth_srid  | integer                 |
  srtext     | character varying(2048) |
  proj4text  | character varying(2048) |
  Indexes:
"spatial_ref_sys_pkey" PRIMARY KEY, btree (srid)
```

You can use standard SQL queries (as we have learned from our introductory sections), to view and manipulate this table - though its not a good idea to update or delete any records unless you know what you are doing.

One SRID you may be interested in is EPSG:4326 - the geographic / lat lon reference system using the WGS 84 ellipsoid. Let's take a look at it:

```
select * from spatial_ref_sys where srid=4326;
```

Result:

```
srid      | 4326
auth_name | EPSG
auth_srid | 4326
srtext    | GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],TOWGS84[0,
0,0,0,0,0,0],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,
AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,
AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]
proj4text | +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs
```

The `srtext` is the projection definition in well known text (you may recognise this from .prj files in your shapefile collection).

16.1.7 In Conclusion

You now have PostGIS functions installed in your copy of PostgreSQL. With this you'll be able to make use of PostGIS" extensive spatial functions.

16.1.8 What's Next?

Next you'll learn how spatial features are represented in a database.

16.2 Lesson: Simple Feature Model

How can we store and represent geographic features in a database? In this lesson we'll cover one approach, the Simple Feature Model as defined by the OGC.

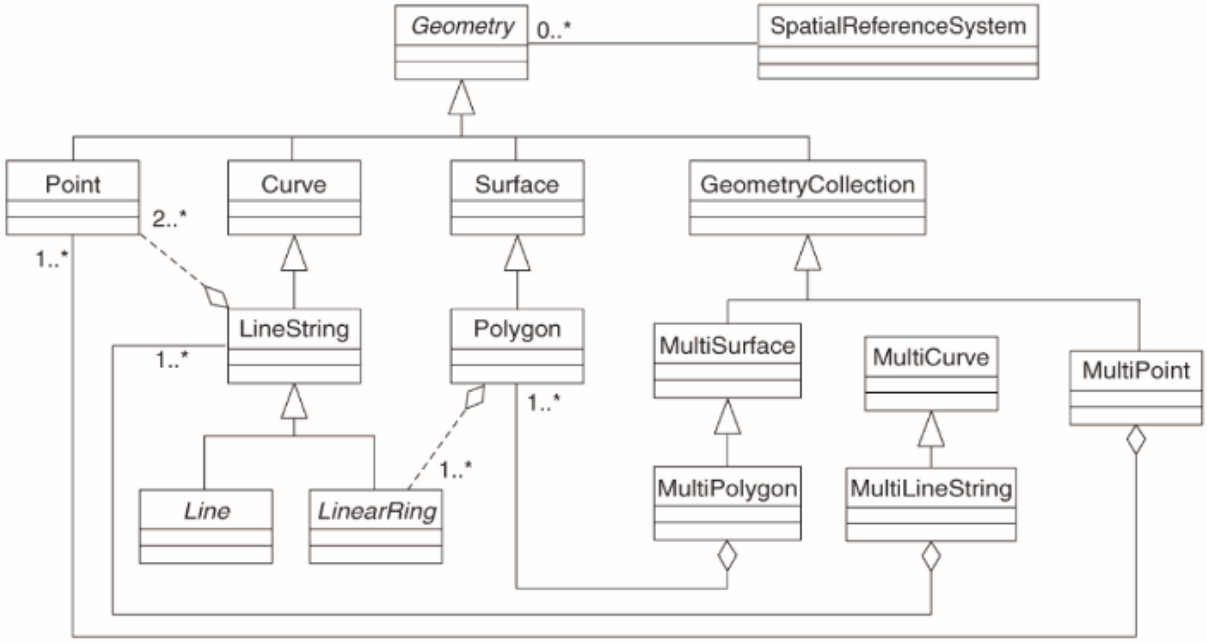
The goal for this lesson: To learn what the SFS Model is and how to use it.

16.2.1 What is OGC

The Open Geospatial Consortium (OGC), an international voluntary consensus standards organization, originated in 1994. In the OGC, more than 370+ commercial, governmental, nonprofit and research organizations worldwide collaborate in an open consensus process encouraging development and implementation of standards for geospatial content and services, GIS data processing and data sharing. - *Wikipedia*

16.2.2 What is the SFS Model

The Simple Feature for SQL (SFS) Model is a *non-topological* way to store geospatial data in a database and defines functions for accessing, operating, and constructing these data.



The model defines geospatial data from Point, Linestring, and Polygon types (and aggregations of them to Multi objects).

For further information, have a look at the OGC Simple Feature for SQL standard.

16.2.3 Add a geometry field to table

Let's add a point field to our people table:

```
alter table people add column geom geometry;
```

16.2.4 Add a constraint based on geometry type

You will notice that the geometry field type does not implicitly specify what *type* of geometry for the field - for that we need a constraint:

```
alter table people
add constraint people_geom_point_chk
    check(st_geometrytype(geom) = 'ST_Point'::text
          OR geom IS NULL);
```

This adds a constraint to the table so that it will only accept a point geometry or a null value.

16.2.5 ??? Try Yourself:

Create a new table called cities and give it some appropriate columns, including a geometry field for storing polygons (the city boundaries). Make sure it has a constraint enforcing geometries to be polygons.

Answer

16.2.6 Populate geometry_columns table

At this point you should also add an entry into the `geometry_columns` table:

```
insert into geometry_columns values
  ('','public','people','geom',2,4326,'POINT');
```

Why? `geometry_columns` is used by certain applications to be aware of which tables in the database contain geometry data.

Nota: If the above `INSERT` statement causes an error, run this query first:

```
select * from geometry_columns;
```

If the column `f_table_name` contains the value `people`, then this table has already been registered and you don't need to do anything more.

The value `2` refers to the number of dimensions; in this case, two: **X** and **Y**.

The value `4326` refers to the projection we are using; in this case, WGS 84, which is referred to by the number 4326 (refer to the earlier discussion about the EPSG).

??? Try Yourself:

Add an appropriate *geometry_columns* entry for your new cities layer

Answer

```
insert into geometry_columns values
      ('','public','cities','geom',2,4326,'POLYGON');
```

16.2.7 Add geometry record to table using SQL

Now that our tables are geo-enabled, we can store geometries in them:

```
insert into people (name,house_no, street_id, phone_no, geom)
        values ('Fault Towers',
                  34,
                  3,
                  '072 812 31 28',
                  'SRID=4326;POINT(33 -33)');
```

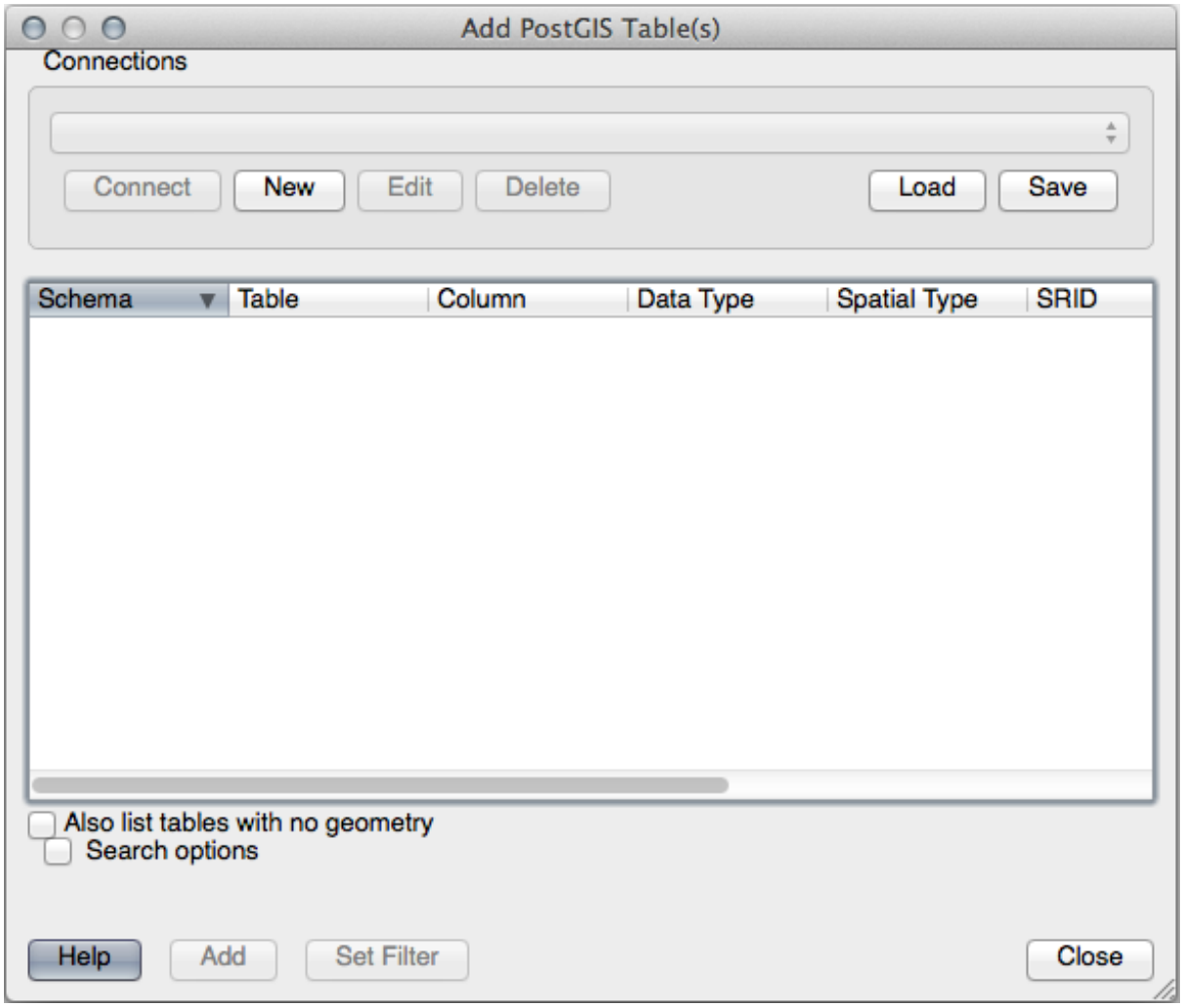
Nota: In the new entry above, you will need to specify which projection (SRID) you want to use. This is because you entered the geometry of the new point using a plain string of text, which does not automatically add the correct projection information. Obviously, the new point needs to use the same SRID as the data-set it is being added to, so you need to specify it.

If at this point you were using a graphical interface, for example, specifying the projection for each point would be automatic. In other words, you usually won't need to worry about using the correct projection for every point you want to add if you've already specified it for that data-set, as we did earlier.

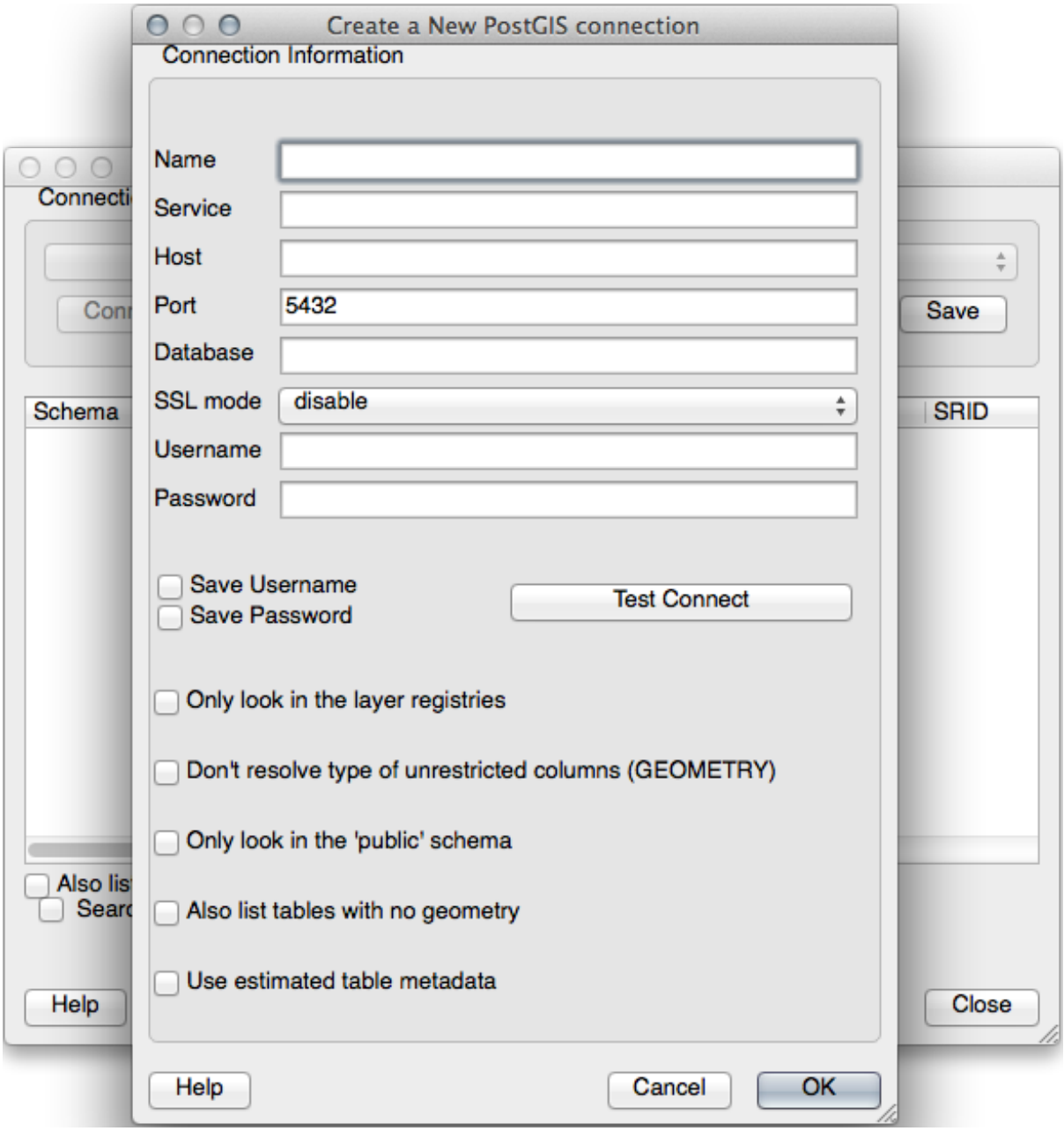
Now is probably a good time to open QGIS and try to view your `people` table. Also, we should try editing / adding / deleting records and then performing select queries in the database to see how the data has changed.

To load a PostGIS layer in QGIS, use the *Layer* ► *Add PostGIS Layers* menu option or toolbar button:

This will open the dialog:



Click on the *New* button to open this dialog:



Then define a new connection, e.g.:

```
Name: myPG
Service:
Host: localhost
Port: 5432
Database: address
User:
Password:
```

To see whether QGIS has found the `address` database and that your username and password are correct, click *Test Connect*. If it works, check the boxes next to *Save Username* and *Save Password*. Then click *OK* to create this connection.

Back in the *Add PostGIS Layers* dialog, click *Connect* and add layers to your project as usual.

??? Try Yourself:

Formulate a query that shows a person's name, street name and position (from the geom column) as plain text.

Answer

```
select people.name,
       streets.name as street_name,
       st_astext(people.geom) as geometry
from   streets, people
where  people.street_id=streets.id;
```

Result:

```
     name      | street_name |   geometry
---------------+-------------+---------------
 Roger Jones   | High street |
 Sally Norman  | High street |
 Jane Smith    | Main Road   |
 Joe Bloggs    | Low Street  |
 Fault Towers  | Main Road   | POINT(33 -33)
(5 rows)
```

As you can see, our constraint allows nulls to be added into the database.

16.2.8 In Conclusion

You have seen how to add spatial objects to your database and view them in GIS software.

16.2.9 What's Next?

Next you'll see how to import data into, and export data from, your database.

16.3 Lesson: Import and Export

Of course, a database with no easy way to migrate data into it and out of it would not be of much use. Fortunately, there are a number of tools that will let you easily move data into and out of PostGIS.

16.3.1 shp2pgsql

shp2pgsql is a commandline tool to import ESRI Shapefile to the database. Under Unix, you can use the following command for importing a new PostGIS table:

```
shp2pgsql -s <SRID> -c -D -I <path to shapefile> <schema>.<table> | \
  psql -d <databasename> -h <hostname> -U <username>
```

Under Windows, you have to perform the import process in two steps:

```
shp2pgsql -s <SRID> -c -D -I <path to shapefile> <schema>.<table> > import.sql
psql psql -d <databasename> -h <hostname> -U <username> -f import.sql
```

You may encounter this error:

```
ERROR:  operator class "gist_geometry_ops" does not exist for access method
"gist"
```

This is a known issue regarding the creation *in situ* of a spatial index for the data you're importing. To avoid the error, exclude the `-I` parameter. This will mean that no spatial index is being created directly, and you'll need to create it in the database after the data have been imported. (The creation of a spatial index will be covered in the next lesson.)

16.3.2 pgsql2shp

pgsql2shp is a commandline tool to export PostGIS Tables, Views or SQL select queries. To do this under Unix:

```
pgsql2shp -f <path to new shapefile> -g <geometry column name> \
  -h <hostname> -U <username> <databasename> <table | view>
```

To export the data using a query:

```
pgsql2shp -f <path to new shapefile> -g <geometry column name> \
  -h <hostname> -U <username> "<query>"
```

16.3.3 ogr2ogr

ogr2ogr is a very powerful tool to convert data into and from postgis to many data formats. ogr2ogr is part of the GDAL library and has to be installed separately. To export a table from PostGIS to GML, you can use this command:

```
ogr2ogr -f GML export.gml PG:'dbname=<databasename> user=<username>
        host=<hostname>' <Name of PostGIS-Table>
```

16.3.4 Gestor BD

You may have noticed another option in the *Database* menu labeled *DB Manager*. This is a tool that provides a unified interface for interacting with spatial databases including PostGIS. It also allows you to import and export from databases to other formats. Since the next module is largely devoted to using this tool, we will only briefly mention it here.

16.3.5 In Conclusion

Importing and exporting data to and from the database can be done in many various ways. Especially when using disparate data sources, you will probably use these functions (or others like them) on a regular basis.

16.3.6 What's Next?

Next we'll look at how to query the data we've created before.

16.4 Lesson: Spatial Queries

Spatial queries are no different from other database queries. You can use the geometry column like any other database column. With the installation of PostGIS in our database, we have additional functions to query our database.

The goal for this lesson: To see how spatial functions are implemented similarly to «normal» non-spatial functions.

16.4.1 Spatial Operators

When you want to know which points are within a distance of 2 degrees to a point(X,Y) you can do this with:

```
select *
from people
where st_distance(geom,'SRID=4326;POINT(33 -34)') < 2;
```

Result:

```
 id |      name      | house_no | street_id |    phone_no     |    geom
----+----------------+----------+-----------+-----------------+-----------------
  6 | Fault Towers |        34 |          3 | 072 812 31 28 | 01010008040C0
(1 row)
```

Nota: geom value above was truncated for space on this page. If you want to see the point in human-readable coordinates, try something similar to what you did in the section «View a point as WKT», above.

How do we know that the query above returns all the points within 2 *degrees*? Why not 2 *meters*? Or any other unit, for that matter?

Answer

The units being used by the example query are degrees, because the CRS that the layer is using is WGS 84. This is a Geographic CRS, which means that its units are in degrees. A Projected CRS, like the UTM projections, is in meters.

Remember that when you write a query, you need to know which units the layer's CRS is in. This will allow you to write a query that will return the results that you expect.

16.4.2 Spatial Indexes

We also can define spatial indexes. A spatial index makes your spatial queries much faster. To create a spatial index on the geometry column use:

```
CREATE INDEX people_geo_idx
  ON people
  USING gist
  (geom);

\d people
```

Result:

```
Table "public.people"
   Column    |          Type           |                 Modifiers
 -----------+-------------------------+-----------------------------------------
  id         | integer                 | not null default
             |                         | nextval('people_id_seq'::regclass)
```

(continues on next page)

(continuação da página anterior)

```
  name       | character varying(50) |
  house_no   | integer               | not null
  street_id  | integer               | not null
  phone_no   | character varying     |
  geom       | geometry              |
Indexes:
  "people_pkey" PRIMARY KEY, btree (id)
  "people_geo_idx" gist (geom)  <-- new spatial key added
  "people_name_idx" btree (name)
Check constraints:
  "people_geom_point_chk" CHECK (st_geometrytype(geom) = 'ST_Point'::text
  OR geom IS NULL)
Foreign-key constraints:
  "people_street_id_fkey" FOREIGN KEY (street_id) REFERENCES streets(id)
```

16.4.3 Try Yourself: ???

Modify the cities table so its geometry column is spatially indexed.

Answer

```
CREATE INDEX cities_geo_idx
  ON cities
  USING gist (geom);
```

16.4.4 PostGIS Spatial Functions Demo

In order to demo PostGIS spatial functions, we'll create a new database containing some (fictional) data.

To start, create a new database (exit the psql shell first):

```
createdb postgis_demo
```

Remember to install the postgis extensions:

```
psql -d postgis_demo -c "CREATE EXTENSION postgis;"
```

Next, import the data provided in the `exercise_data/postgis/` directory. Refer back to the previous lesson for instructions, but remember that you'll need to create a new PostGIS connection to the new database. You can import from the terminal or via DB Manager. Import the files into the following database tables:

- `points.shp` into `building`
- `lines.shp` into `road`
- `polygons.shp` into `region`

Load these three database layers into QGIS via the *Add PostGIS Layers* dialog, as usual. When you open their attribute tables, you'll note that they have both an `id` field and a `gid` field created by the PostGIS import.

Now that the tables are imported, we can use PostGIS to query the data. Go back to your terminal (command line) and enter the psql prompt by running:

```
psql postgis_demo
```

We'll demo some of these select statements by creating views from them, so that you can open them in QGIS and see the results.

Select by location

Get all the buildings in the KwaZulu region:

```
SELECT a.id, a.name, st_astext(a.geom) as point
  FROM building a, region b
    WHERE st_within(a.geom, b.geom)
    AND b.name = 'KwaZulu';
```

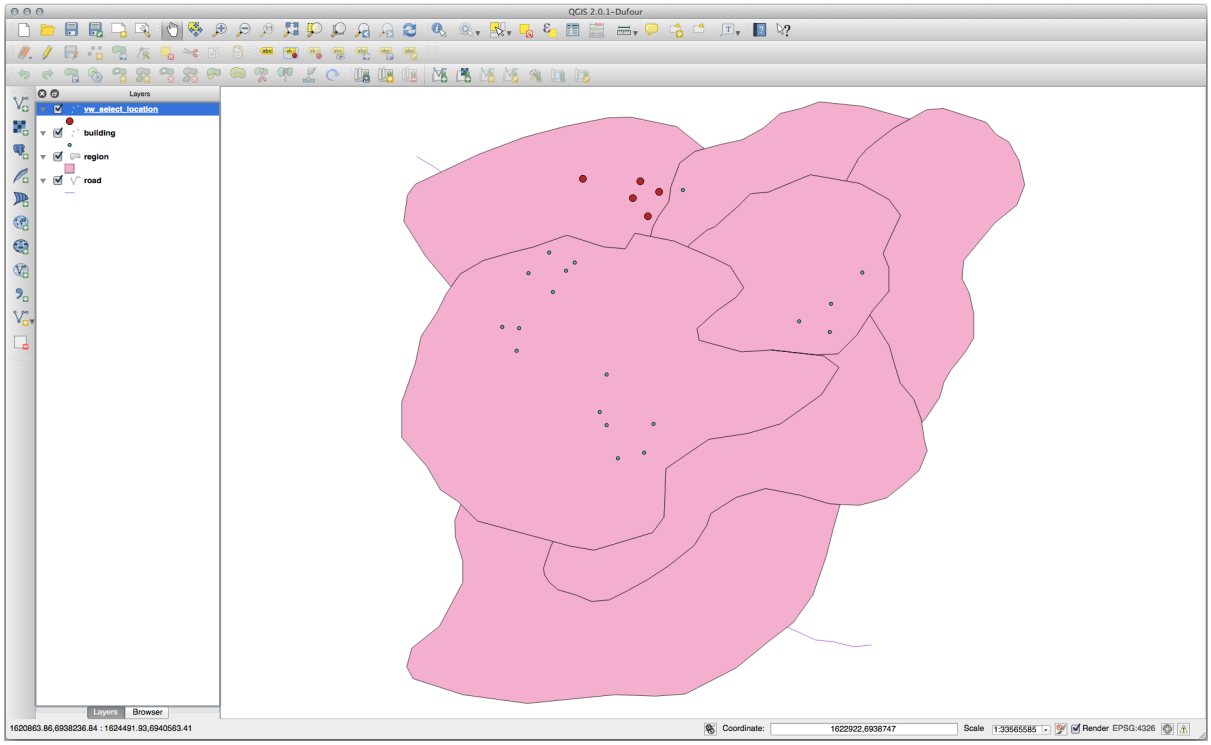
Result:

```
 id | name |                    point
----+------+-------------------------------------------
 30 | York | POINT(1622345.23785063 6940490.65844485)
 33 | York | POINT(1622495.65620524 6940403.87862489)
 35 | York | POINT(1622403.09106394 6940212.96302097)
 36 | York | POINT(1622287.38463732 6940357.59605424)
 40 | York | POINT(1621888.19746548 6940508.01440885)
(5 rows)
```

Or, if we create a view from it:

```
CREATE VIEW vw_select_location AS
  SELECT a.gid, a.name, a.geom
    FROM building a, region b
      WHERE st_within(a.geom, b.geom)
      AND b.name = 'KwaZulu';
```

Add the view as a layer and view it in QGIS:



Select neighbors

Show a list of all the names of regions adjoining the Hokkaido region:

```
SELECT b.name
  FROM region a, region b
    WHERE st_touches(a.geom, b.geom)
    AND a.name = 'Hokkaido';
```

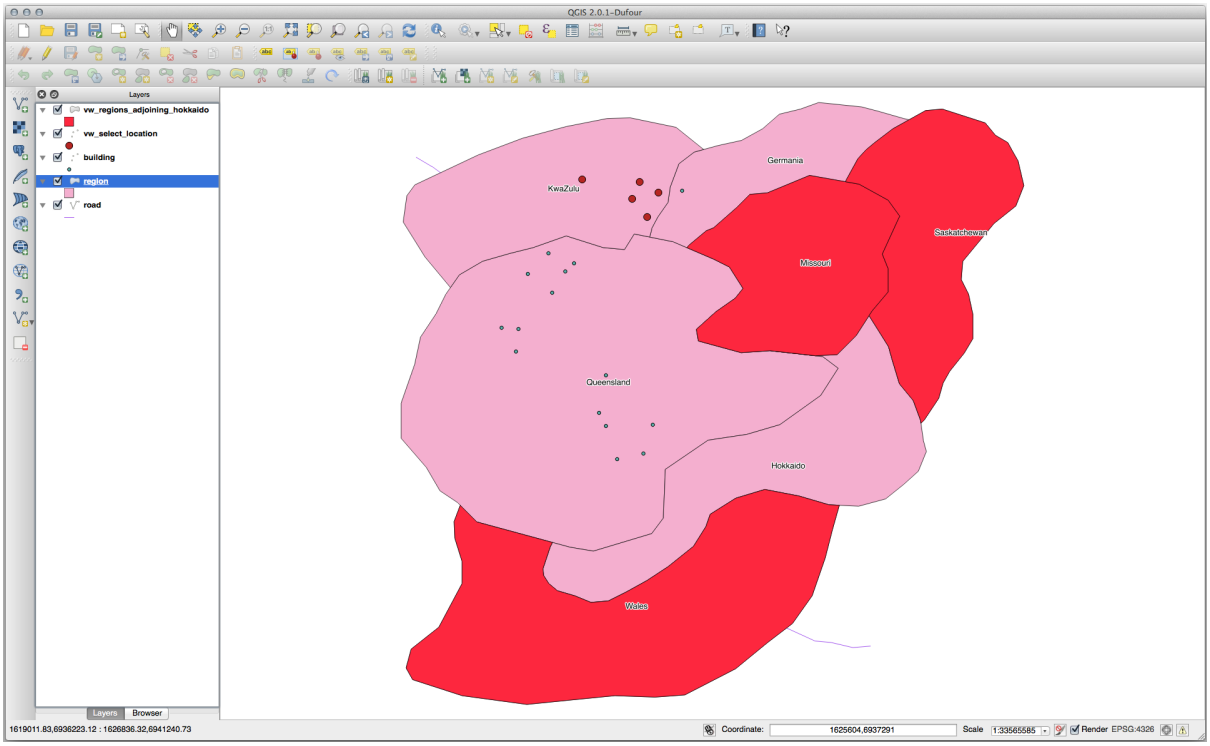
Result:

```
    name
--------------
 Missouri
 Saskatchewan
 Wales
(3 rows)
```

As a view:

```
CREATE VIEW vw_regions_adjoining_hokkaido AS
  SELECT b.gid, b.name, b.geom
    FROM region a, region b
      WHERE st_touches(a.geom, b.geom)
      AND a.name = 'Hokkaido';
```

In QGIS:



Note the missing region (Queensland). This may be due to a topology error. Artifacts such as this can alert us to potential problems in the data. To solve this enigma without getting caught up in the anomalies the data may have, we could use a buffer intersect instead:

```
CREATE VIEW vw_hokkaido_buffer AS
  SELECT gid, ST_BUFFER(geom, 100) as geom
```

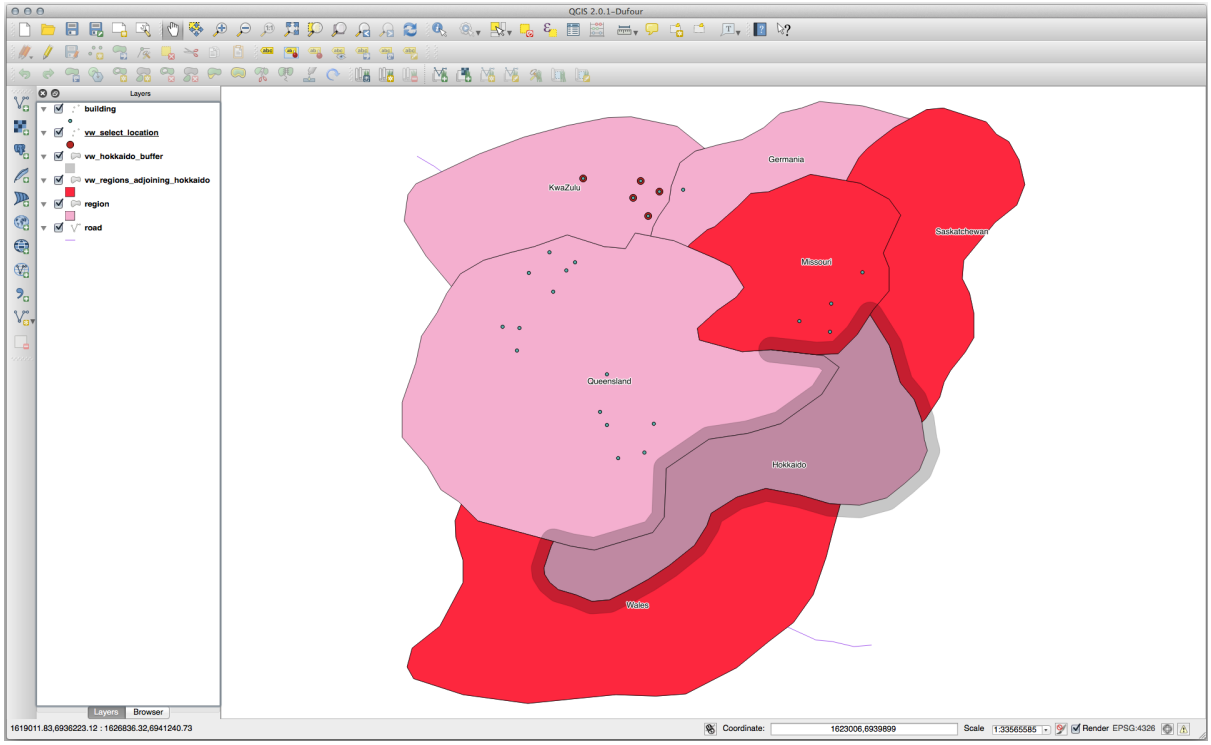
(continues on next page)

(continuação da página anterior)

```
    FROM region
      WHERE name = 'Hokkaido';
```

This creates a buffer of 100 meters around the region Hokkaido.

The darker area is the buffer:

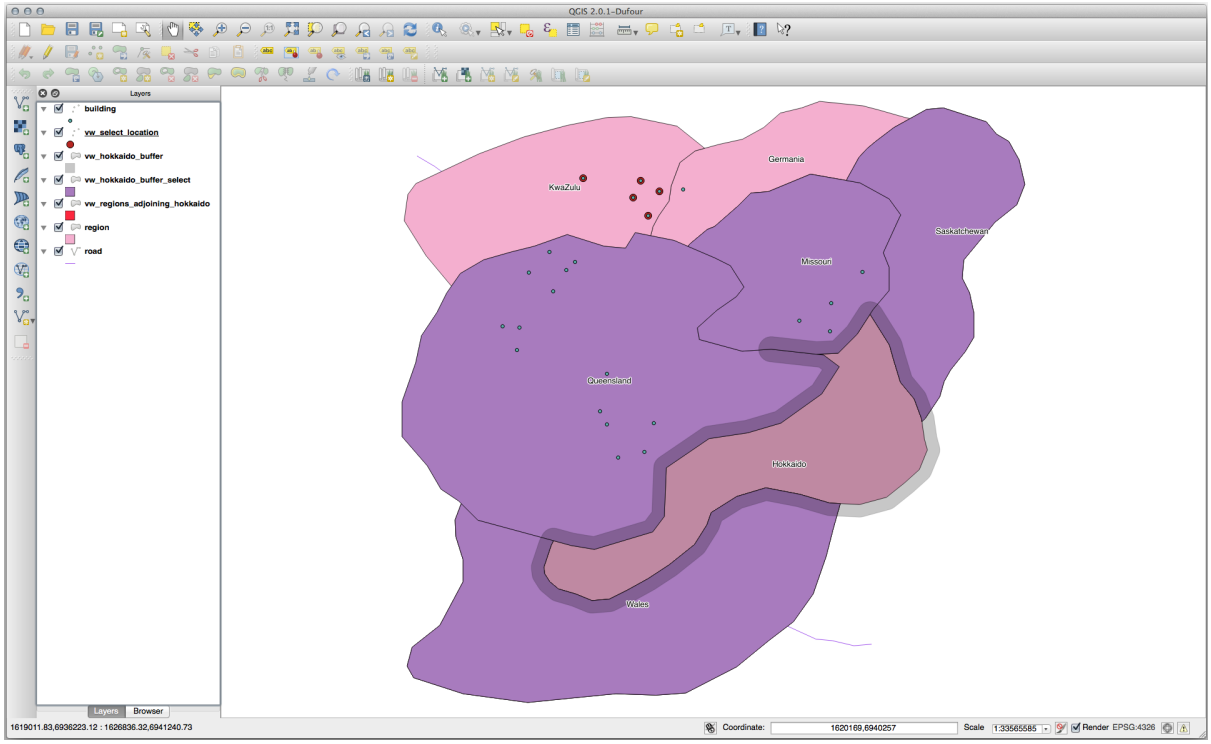


Select using the buffer:

```
CREATE VIEW vw_hokkaido_buffer_select AS
  SELECT b.gid, b.name, b.geom
    FROM
    (
      SELECT * FROM
        vw_hokkaido_buffer
    ) a,
    region b
    WHERE ST_INTERSECTS(a.geom, b.geom)
    AND b.name != 'Hokkaido';
```

In this query, the original buffer view is used as any other table would be. It is given the alias `a`, and its geometry field, `a.geom`, is used to select any polygon in the `region` table (alias `b`) that intersects it. However, Hokkaido itself is excluded from this select statement, because we don't want it; we only want the regions adjoining it.

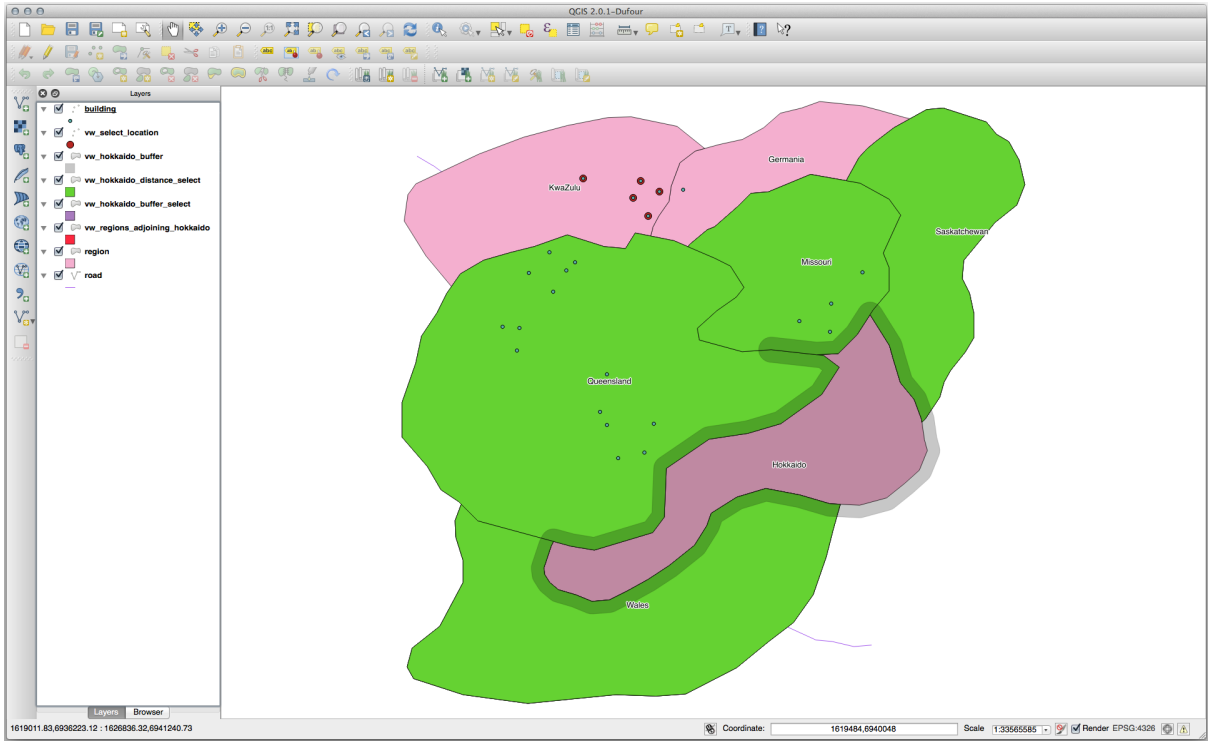
In QGIS:



It is also possible to select all objects within a given distance, without the extra step of creating a buffer:

```
CREATE VIEW vw_hokkaido_distance_select AS
  SELECT b.gid, b.name, b.geom
    FROM region a, region b
      WHERE ST_DISTANCE (a.geom, b.geom) < 100
      AND a.name = 'Hokkaido'
      AND b.name != 'Hokkaido';
```

This achieves the same result, without need for the interim buffer step:



Select unique values

Show a list of unique town names for all buildings in the Queensland region:

```
SELECT DISTINCT a.name
  FROM building a, region b
    WHERE st_within(a.geom, b.geom)
    AND b.name = 'Queensland';
```

Result:

```
  name
---------
 Beijing
 Berlin
 Atlanta
(3 rows)
```

Further examples ...

```
CREATE VIEW vw_shortestline AS
  SELECT b.gid AS gid,
        ST_ASTEXT(ST_SHORTESTLINE(a.geom, b.geom)) as text,
        ST_SHORTESTLINE(a.geom, b.geom) AS geom
    FROM road a, building b
      WHERE a.id=5 AND b.id=22;

CREATE VIEW vw_longestline AS
  SELECT b.gid AS gid,
         ST_ASTEXT(ST_LONGESTLINE(a.geom, b.geom)) as text,
         ST_LONGESTLINE(a.geom, b.geom) AS geom
    FROM road a, building b
      WHERE a.id=5 AND b.id=22;
```

```
CREATE VIEW vw_road_centroid AS
  SELECT a.gid as gid, ST_CENTROID(a.geom) as geom
    FROM road a
      WHERE a.id = 1;

CREATE VIEW vw_region_centroid AS
  SELECT a.gid as gid, ST_CENTROID(a.geom) as geom
    FROM region a
      WHERE a.name = 'Saskatchewan';
```

```
SELECT ST_PERIMETER(a.geom)
  FROM region a
    WHERE a.name='Queensland';

SELECT ST_AREA(a.geom)
  FROM region a
    WHERE a.name='Queensland';
```

```
CREATE VIEW vw_simplify AS
  SELECT gid, ST_Simplify(geom, 20) AS geom
    FROM road;

CREATE VIEW vw_simplify_more AS
  SELECT gid, ST_Simplify(geom, 50) AS geom
    FROM road;
```

```
CREATE VIEW vw_convex_hull AS
  SELECT
    ROW_NUMBER() over (order by a.name) as id,
    a.name as town,
    ST_CONVEXHULL(ST_COLLECT(a.geom)) AS geom
    FROM building a
    GROUP BY a.name;
```

16.4.5 In Conclusion

You have seen how to query spatial objects using the new database functions from PostGIS.

16.4.6 What's Next?

Next we're going to investigate the structures of more complex geometries and how to create them using PostGIS.

16.5 Lesson: Geometry Construction

In this section we are going to delve a little deeper into how simple geometries are constructed in SQL. In reality, you will probably use a GIS like QGIS to create complex geometries using their digitising tools; however, understanding how they are formulated can be handy for writing queries and understanding how the database is assembled.

The goal of this lesson: To better understand how to create spatial entities directly in PostgreSQL/PostGIS.

16.5.1 Creating Linestrings

Going back to our `address` database, let's get our streets table matching the others; i.e., having a constraint on the geometry, an index and an entry in the geometry_columns table.

16.5.2 Try Yourself: ???

- Modify the `streets` table so that it has a geometry column of type ST_LineString.
- Don't forget to do the accompanying update to the geometry columns table!
- Also add a constraint to prevent any geometries being added that are not LINESTRINGS or null.
- Create a spatial index on the new geometry column

Answer

```
alter table streets add column geom geometry;
alter table streets add constraint streets_geom_point_chk check
     (st_geometrytype(geom) = 'ST_LineString'::text OR geom IS NULL);
insert into geometry_columns values ('','public','streets','geom',2,4326,
     'LINESTRING');
create index streets_geo_idx
  on streets
  using gist
  (geom);
```

Now let's insert a linestring into our streets table. In this case we will update an existing street record:

```
update streets
set geom = 'SRID=4326;LINESTRING(20 -33, 21 -34, 24 -33)'
where streets.id=2;
```

Take a look at the results in QGIS. (You may need to right-click on the streets layer in the "Layers" panel, and choose "Zoom to layer extent".)

Now create some more streets entries - some in QGIS and some from the command line.

16.5.3 Creating Polygons

Creating polygons is just as easy. One thing to remember is that by definition, polygons have at least four vertices, with the last and first being co-located:

```
insert into cities (name, geom)
values ('Tokyo', 'SRID=4326;POLYGON((10 -10, 5 -32, 30 -27, 10 -10))');
```

Nota: A polygon requires double brackets around its coordinate list; this is to allow you to add complex polygons with multiple unconnected areas. For instance

```
insert into cities (name, geom)
values ('Tokyo Outer Wards',
        'SRID=4326;POLYGON((20 10, 20 20, 35 20, 20 10),
                           (-10 -30, -5 0, -15 -15, -10 -30))'
        );
```

If you followed this step, you can check what it did by loading the cities dataset into QGIS, opening its attribute table, and selecting the new entry. Note how the two new polygons behave like one polygon.

16.5.4 Exercise: Linking Cities to People

For this exercise you should do the following:

- Delete all data from your people table.
- Add a foreign key column to people that references the primary key of the cities table.
- Use QGIS to capture some cities.
- Use SQL to insert some new people records, ensuring that each has an associated street and city.

Your updated people schema should look something like this:

```
\d people

Table "public.people"
   Column   |         Type          |                      Modifiers
 -----------+-----------------------+-----------------------------------------------
  id        | integer               | not null
            |                       | default nextval('people_id_seq'::regclass)
  name      | character varying(50) |
  house_no  | integer               | not null
  street_id | integer               | not null
  phone_no  | character varying     |
  geom      | geometry              |
  city_id   | integer               | not null
Indexes:
  "people_pkey" PRIMARY KEY, btree (id)
  "people_name_idx" btree (name)
```

(continues on next page)

(continuação da página anterior)

```
Check constraints:
  "people_geom_point_chk" CHECK (st_geometrytype(geom) =
                        'ST_Point'::text OR geom IS NULL)
Foreign-key constraints:
  "people_city_id_fkey" FOREIGN KEY (city_id) REFERENCES cities(id)
  "people_street_id_fkey" FOREIGN KEY (street_id) REFERENCES streets(id)
```

Answer

```
delete from people;
alter table people add column city_id int not null references cities(id);
```

(capture cities in QGIS)

```
insert into people (name,house_no, street_id, phone_no, city_id, geom)
   values ('Faulty Towers',
            34,
            3,
            '072 812 31 28',
            1,
            'SRID=4326;POINT(13 -15)');

insert into people (name,house_no, street_id, phone_no, city_id, geom)
   values ('IP Knightly',
            32,
            1,
            '071 812 31 28',
            1,
            'SRID=4326;POINT(18 -24)');

insert into people (name,house_no, street_id, phone_no, city_id, geom)
   values ('Rusty Bedsprings',
            39,
            1,
            '071 822 31 28',
            1,
            'SRID=4326;POINT(22 -25)');
```

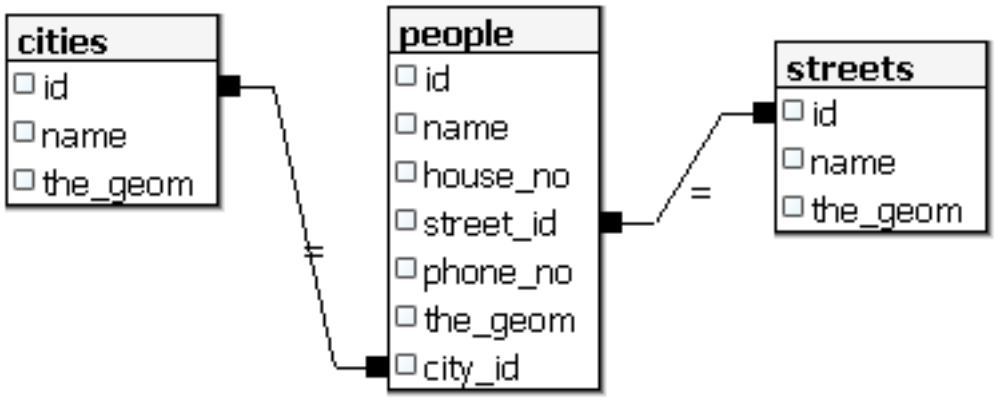
If you're getting the following error message:

```
ERROR:  insert or update on table "people" violates foreign key constraint
        "people_city_id_fkey"
DETAIL: Key (city_id)=(1) is not present in table "cities".
```

then it means that while experimenting with creating polygons for the cities table, you must have deleted some of them and started over. Just check the entries in your cities table and use any *id* which exists.

16.5.5 Looking at Our Schema

By now our schema should be looking like this:



16.5.6 Try Yourself: ???

Create city boundaries by computing the minimum convex hull of all addresses for that city and computing a buffer around that area.

Answer

- Add some people in "Tokyo Outer Wards" city

```
INSERT INTO people (name, house_no, street_id, phone_no, city_id, geom)
   VALUES ('Bad Aboum',
           57,
           2,
           '073 712 31 21',
           2,
           'SRID=4326;POINT(22 18)');

INSERT INTO people (name, house_no, street_id, phone_no, city_id, geom)
   VALUES ('Pat Atra',
           59,
           2,
           '074 712 31 25',
           2,
           'SRID=4326;POINT(23 14)');

INSERT INTO people (name, house_no, street_id, phone_no, city_id, geom)
   VALUES ('Kat Herin',
           65,
           2,
           '074 722 31 28',
           2,
           'SRID=4326;POINT(29 18)');
```

- Create myPolygonTable table

```
CREATE TABLE myPolygonTable (
  id serial NOT NULL PRIMARY KEY,
  city_id int NOT NULL REFERENCES cities(id),
  geom geometry NOT NULL
);
```

(continues on next page)

(continuação da página anterior)

```
ALTER TABLE myPolygonTable
  ADD CONSTRAINT myPolygonTable_geom_polygon_chk
  CHECK (st_geometrytype(geom) = 'ST_Polygon'::text );
```

- Create and load the convex hulls

```
INSERT INTO myPolygonTable (city_id, geom)
  SELECT * FROM
  (
    SELECT
      ROW_NUMBER() over (order by city_id)::integer AS city_id,
      ST_CONVEXHULL(ST_COLLECT(geom)) AS geom
        FROM people
        GROUP BY city_id
  ) convexHulls;
```

16.5.7 Access Sub-Objects

With the SFS-Model functions, you have a wide variety of options to access sub-objects of SFS Geometries. When you want to select the first vertex point of every polygon geometry in the table myPolygonTable, you have to do this in this way:

- Transform the polygon boundary to a linestring:

```
select st_boundary(geom) from myPolygonTable;
```

- Select the first vertex point of the resultant linestring:

```
select st_startpoint(myGeometry)
from (
    select st_boundary(geom) as myGeometry
    from myPolygonTable) as foo;
```

16.5.8 Data Processing

PostGIS supports all OGC SFS/MM standard conform functions. All these functions start with `ST_`.

16.5.9 Clipping

To clip a subpart of your data you can use the `ST_INTERSECT()` function. To avoid empty geometries, use:

```
where not st_isempty(st_intersection(a.geom, b.geom))
```

```
select st_intersection(a.geom, b.geom), b.*
from clip as a, road_lines as b
where not st_isempty(st_intersection(st_setsrid(a.geom,32734),
  b.geom));
```

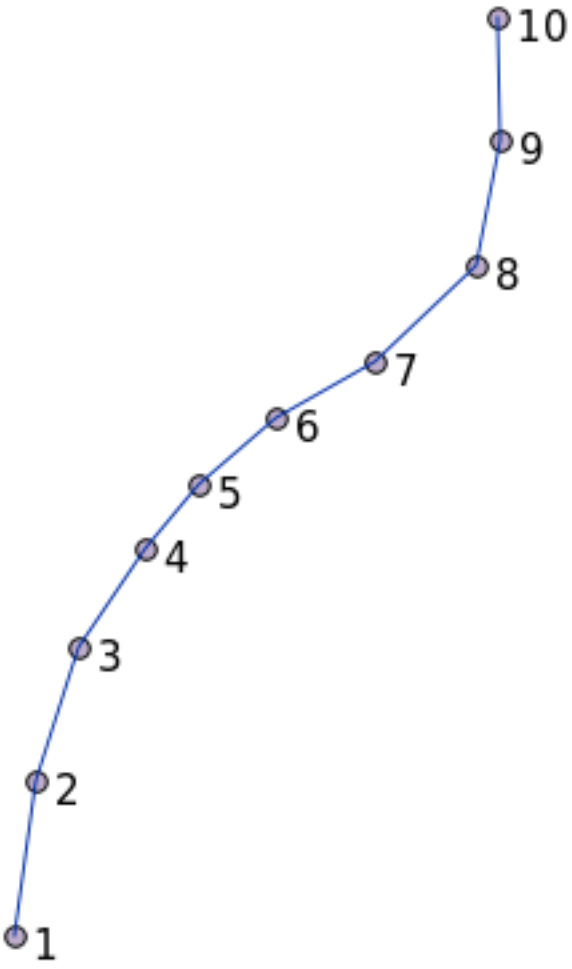
16.5.10 Building Geometries from Other Geometries

From a given point table, you want to generate a linestring. The order of the points is defined by their `id`. Another ordering method could be a timestamp, such as the one you get when you capture waypoints with a GPS receiver.

To create a linestring from a new point layer called “points”, you can run the following command:

```
select ST_LineFromMultiPoint(st_collect(geom)), 1 as id
from (
  select geom
  from points
  order by id
) as foo;
```

To see how it works without creating a new layer, you could also run this command on the “people” layer, although of course it would make little real-world sense to do this.



16.5.11 Geometry Cleaning

You can get more information for this topic in this blog entry.

16.5.12 Differences between tables

To detect the difference between two tables with the same structure, you can use the PostgreSQL keyword `EXCEPT`:

```
select * from table_a
except
select * from table_b;
```

As the result, you will get all records from table_a which are not stored in table_b.

16.5.13 Tablespaces

You can define where postgres should store its data on disk by creating tablespaces:

```
CREATE TABLESPACE homespace LOCATION '/home/pg';
```

When you create a database, you can then specify which tablespace to use e.g.:

```
createdb --tablespace=homespace t4a
```

16.5.14 In Conclusion

You've learned how to create more complex geometries using PostGIS statements. Keep in mind that this is mostly to improve your tacit knowledge when working with geo-enabled databases through a GIS frontend. You usually won't need to actually enter these statements manually, but having a general idea of their structure will help you when using a GIS, especially if you encounter errors that would otherwise seem cryptic.

CAPÍTULO 17

Guia de Processamento QGIS

Este módulo teve os contributos de Victor Olaya e Paolo Cavallini.

Conteúdos:

17.1 Introdução

This guide describes how to use the QGIS processing framework. It assumes no previous knowledge of the Processing framework or any of the applications that it rely on. It assumes basic knowledge of QGIS. The chapters about scripting assume you have some basic knowledge of Python and maybe the QGIS Python API.

This guide is designed for self-study or to be used for running a processing workshop.

Examples in this guide use QGIS 3.4. They might not work or not be available in versions other than that one.

This guide is comprised of a set of small exercises of progressive complexity. If you have never used the processing framework, you should start from the very beginning. If you have some previous experience, feel free to skip lessons. They are more or less independent of each other, and each one introduces some new concept or some new element, as indicated in the chapter title and the short introduction at the beginning of each chapter. That should make it easy to locate lessons dealing with a particular topic.

For a more systematic description of all the framework components and their usage, it is recommended to check the corresponding chapter in the user manual. Use it as a support text along with this guide.

All the exercises in this guide use the same free dataset used throughout the training manual and referenced at section *Data*. The zip file to download contains several folders corresponding to each one of the lessons in this guide. In each of them you will find a QGIS project file. Just open it and you will be ready to start the lesson.

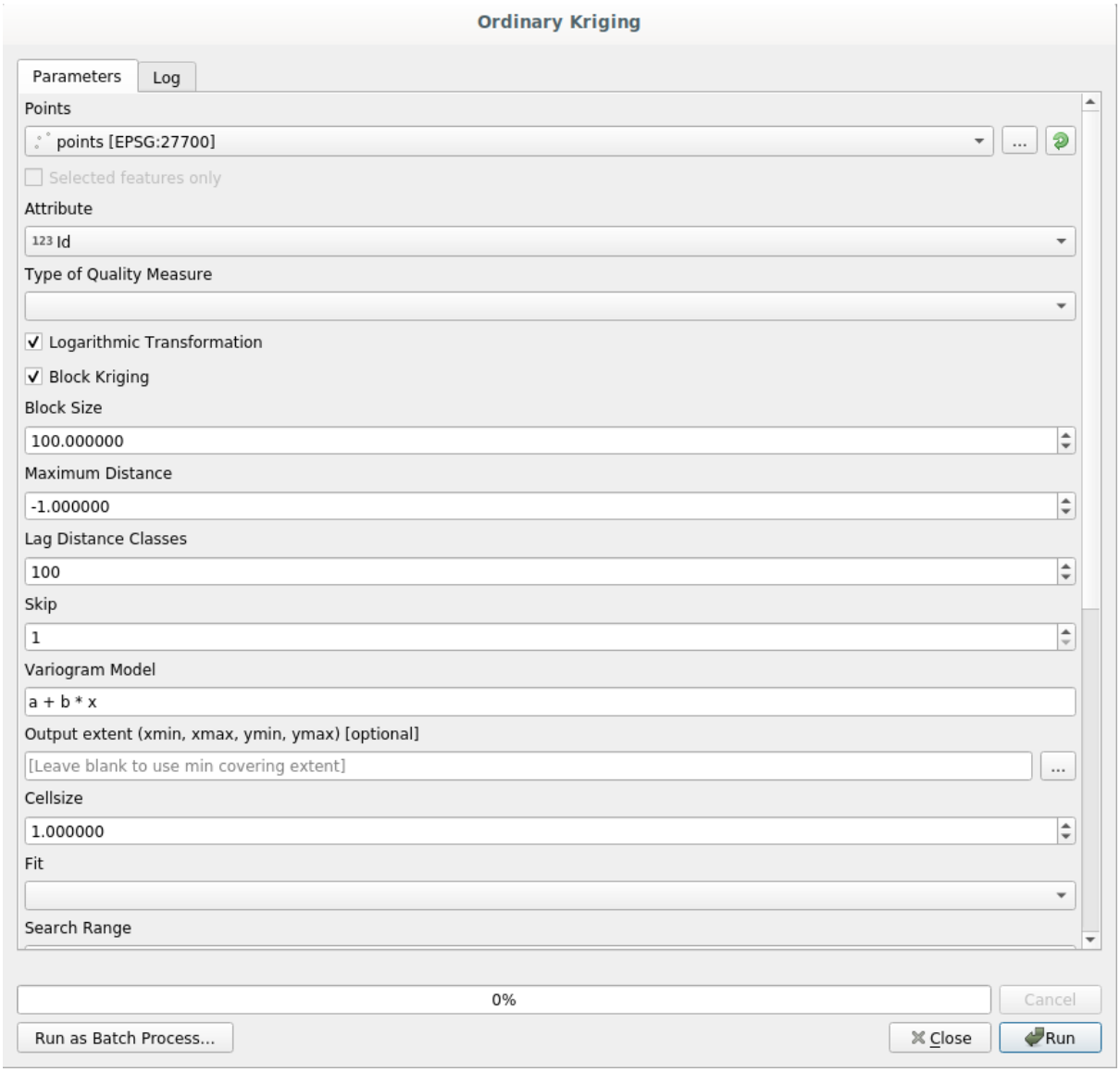
Enjoy!

17.2 Aviso importante antes de começar

Just like the manual of a word processor doesn't teach you how to write a novel or a poem, or a CAD tutorial doesn't show you how to calculate the size of a beam for a building, this guide will not teach you spatial analysis. Instead, it will show you how to use the QGIS Processing framework, a powerful tool for performing spatial analysis. It is up to you to learn the required concepts that are needed to understand that type of analysis. Without them, there is no point in using the framework and its algorithms, although you might be tempted to try.

Let's show this more clearly with an example.

Given a set of points and a value of a given variable value at each point, you can calculate a raster layer from them using the *Kriging* geoalgorithm. The parameters dialog for that module is like the following one.



It looks complex, right?

By reading this manual, you will learn things such as how to use that module, how to run it in a batch process to create raster layers from hundreds of points layers in a single run, or what happens if the input layer has some points selected. However, the parameters themselves are not explained. A seasoned analyst with a good knowledge of geostatistics will have no problem understanding those parameters. If you are not one of them and *sill*, *range*, or *nugget* are not familiar concepts to you, then you should not use the *Kriging* module. More than that, you are far from being ready to use the *Kriging* module, since it requires learning about concepts such as spatial autocorrelation or semivariograms,

which probably you also haven't heard before, or at least haven't studied long enough. You should first study and understand them, and then come back to QGIS to actually run it and perform the analysis. Ignoring this will result in wrong results and poor (and most likely useless) analysis.

Although not all algorithms are as complex as kriging (but some of them are even more complex!), almost all of them require understanding the fundamental analysis ideas that they are based on. Without that knowledge, using them will most likely lead to poor results.

Using geoalgorithms without having a good foundation of spatial analysis is like trying to write a novel without knowing anything about grammar or syntax, and having no knowledge about storytelling. You might get a result, but it is likely to have no value at all. Please, don't fool yourself and think that after reading this guide you are already capable of performing spatial analysis and get sound results. You need to study spatial analysis as well.

Here is a good reference that you can read to learn more about spatial data analysis.

Geospatial Analysis (3rd Edition): A Comprehensive Guide to Principles, Techniques and Software Tools Michael John De Smith, Michael F. Goodchild, Paul A. Longley

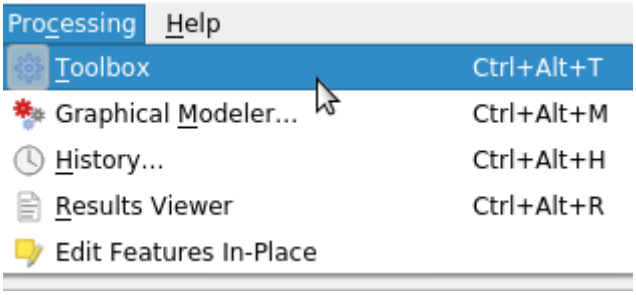
It is available online here

17.3 Setting-up the processing framework

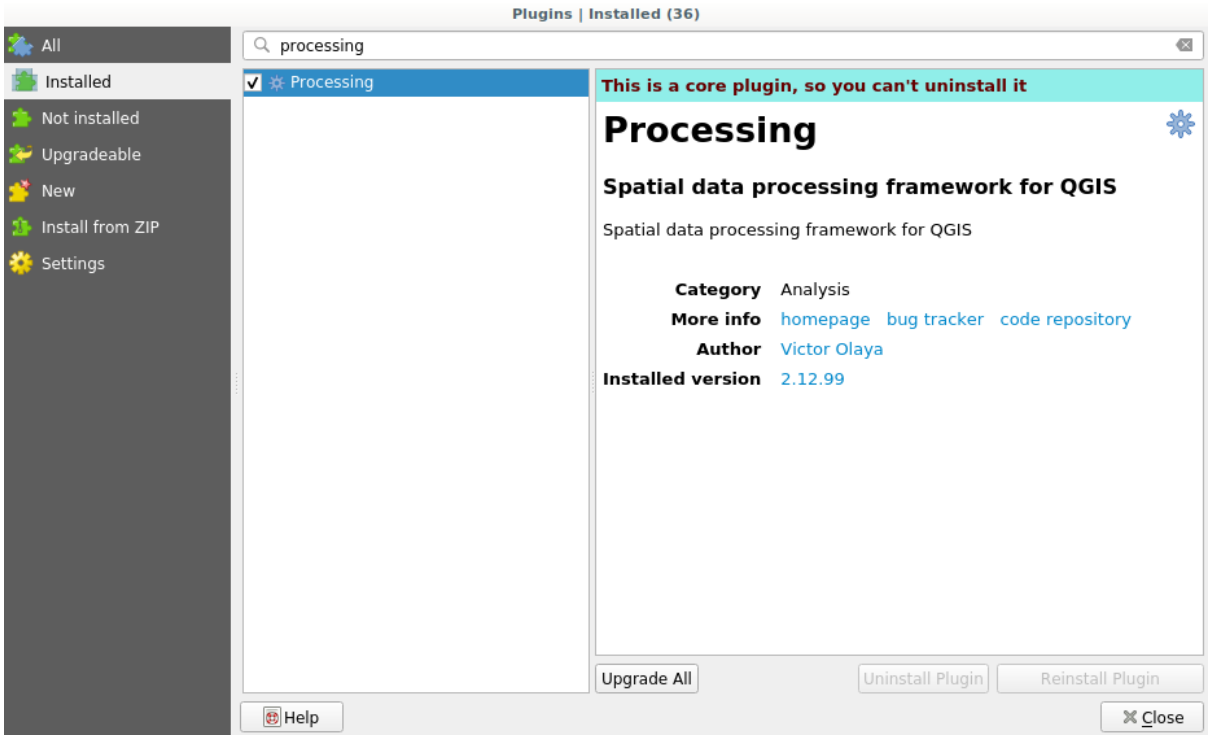
The first thing to do before using the processing framework is to configure it. There is not much to set-up, so this is an easy task.

Later on we will show how to configure the external applications that are used for extending the list of available algorithms, but for now we are just going to work with the framework itself.

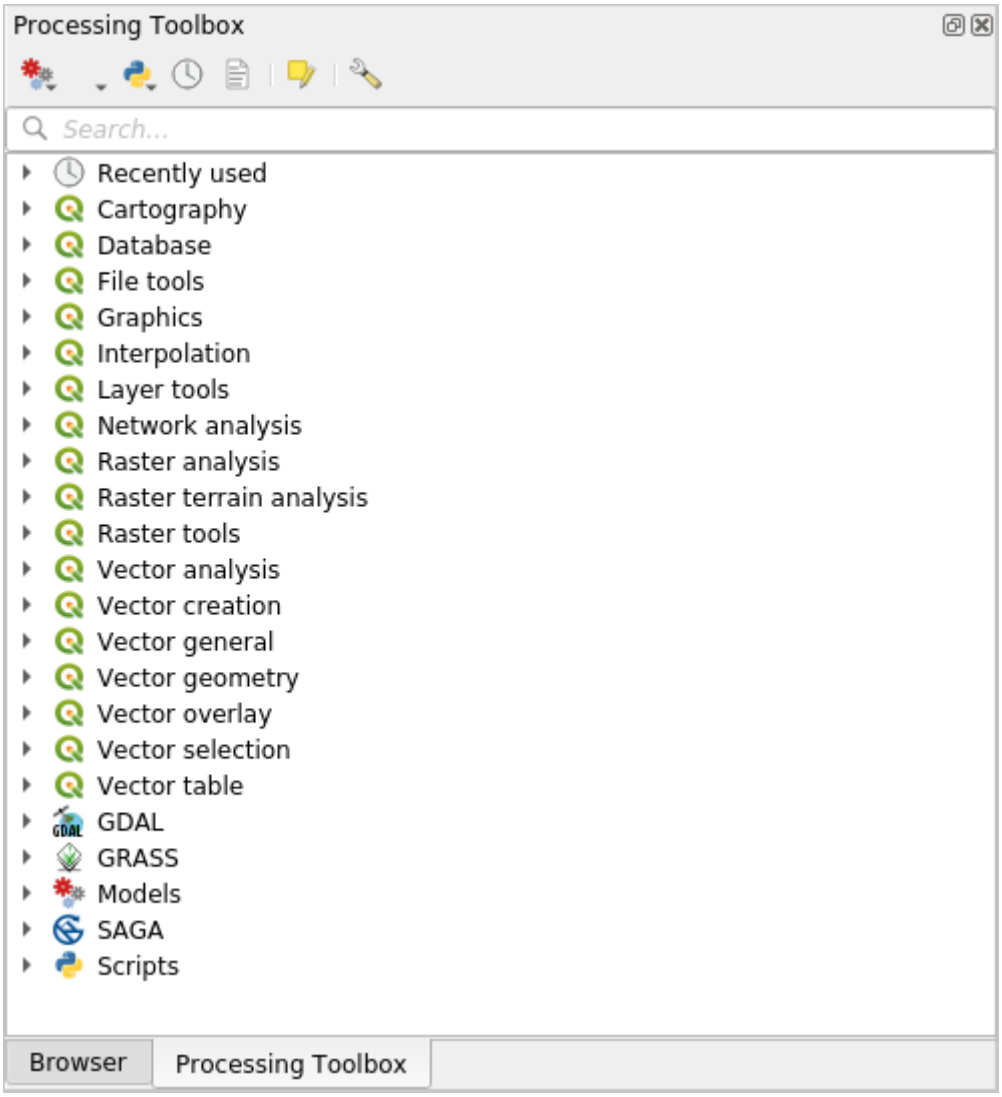
The processing framework is a core QGIS plugin, which means that it should already be installed in your system, since it is included with QGIS. In case it is active, you should see a menu called *Processing* in your menu bar. There you can access all the framework components.



If you cannot find that menu, you have to enable the plugin by going to the plugin manager and activating it.



The main element that we are going to work with is the toolbox. Click on the corresponding menu entry and you will see the toolbox docked at the right side of the QGIS window.



The toolbox contains a list of all the available algorithms, divided in groups called *Providers*. Providers can be (de)activated in the *Settings* ► *Options* ► *Processing*. We will discuss that dialog later in this manual.

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.

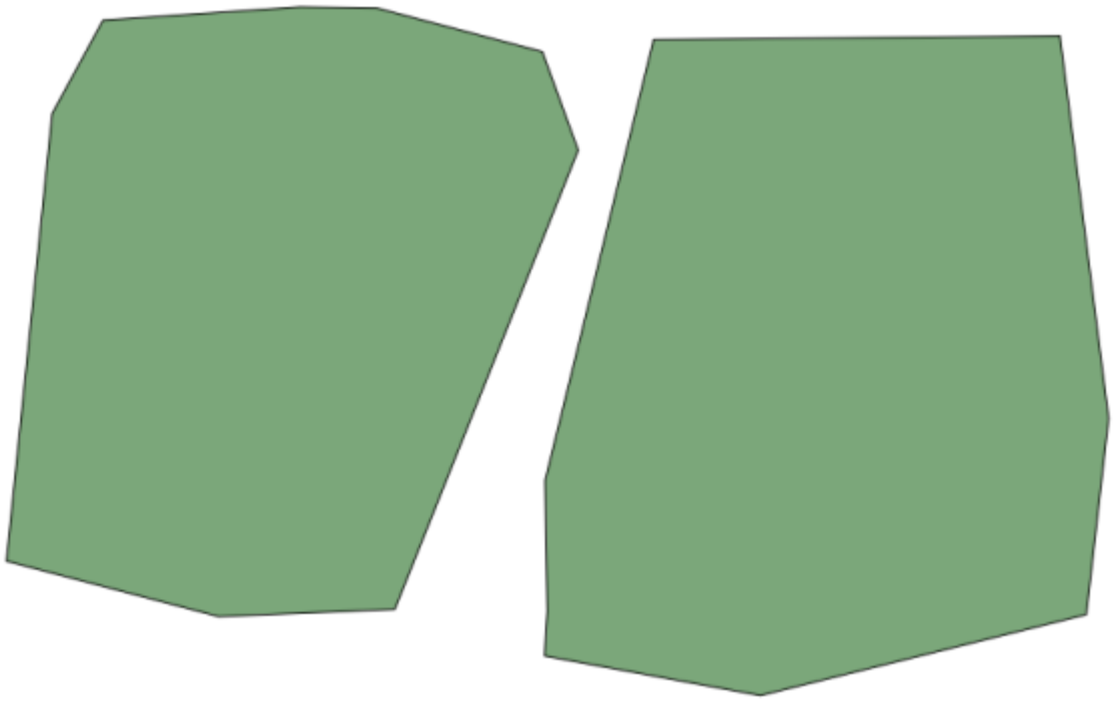
If you have reached this point, now you are ready to use geoalgorithms. There is no need to configure anything else by now. We can already run our first algorithm, which we will do in the next lesson.

17.4 Running our first algorithm. The toolbox

Nota: In this lesson we will run our first algorithm, and get our first result from it.

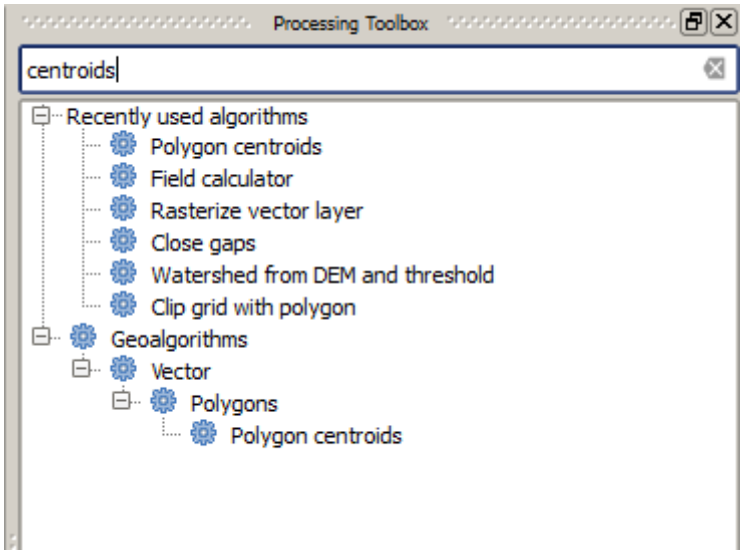
As we have already mentioned, the processing framework can run algorithms from other applications, but it also contains native algorithms that need no external software to be run. To start exploring the processing framework, we are going to run one of those native algorithms. In particular, we are going to calculate the centroids of set of polygons.

First, open the QGIS project corresponding to this lesson. It contains just a single layer with two polygons

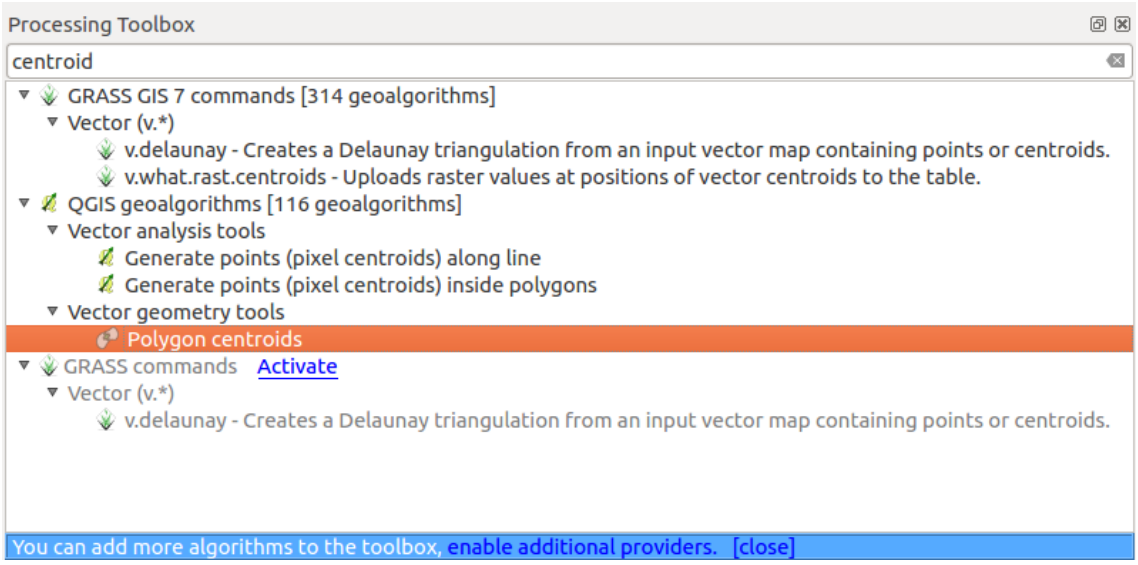


Now go to the text box at the top of the toolbox. That is the search box, and if you type text in it, it will filter the list of algorithms so just those ones containing the entered text are shown. If there are algorithms that match your search but belong to a provider that is not active, an additional label will be shown in the lower part of the toolbox.

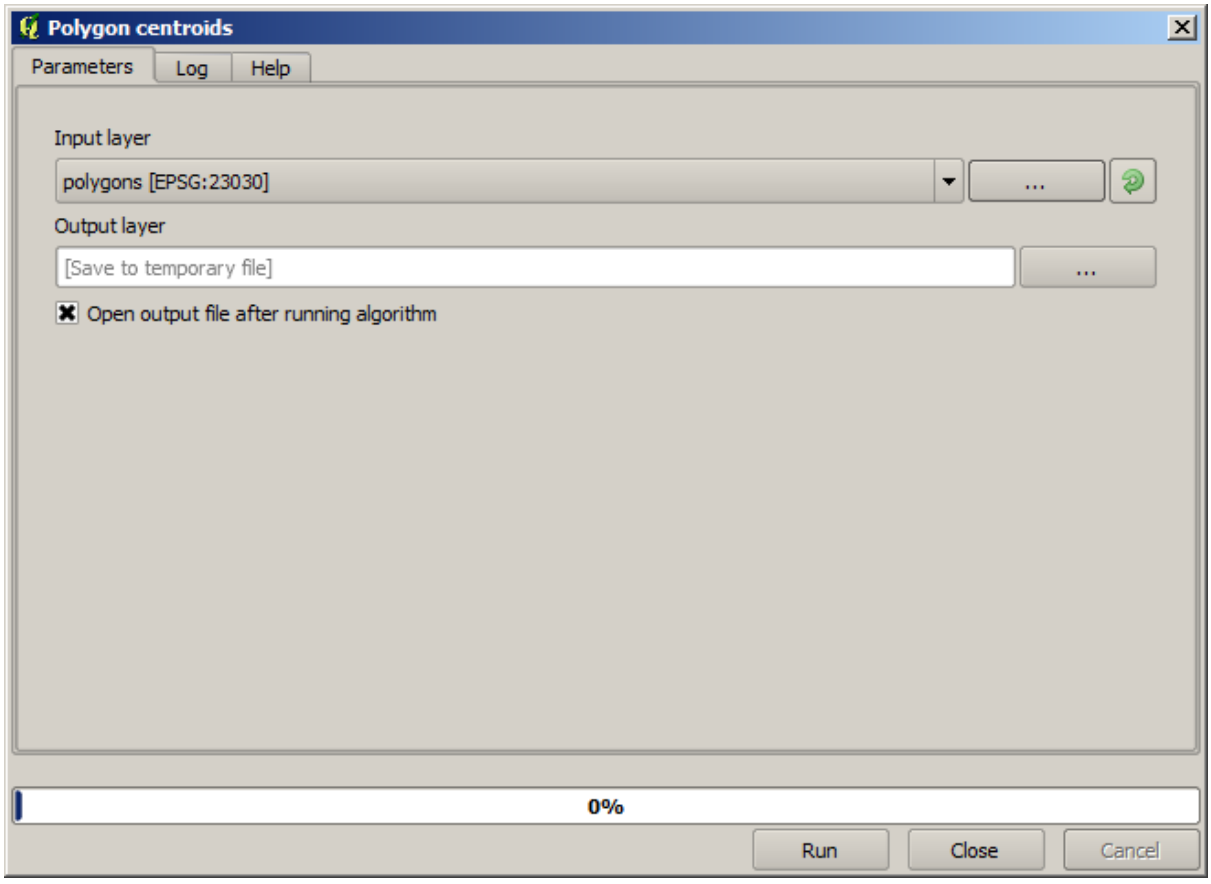
Type `centroids` and you should see something like this.



The search box is a very practical way of finding the algorithm you are looking for. At the bottom of the dialog, an additional label shows that there are algorithms that match your search but belong to a provider that is not active. If you click on the link in that label, the list of algorithms will also include results from those inactive providers, which will be shown in light gray. A link to activate each inactive provider is also shown. We'll see later how to activate other providers.



To execute an algorithm, you just have to double-click on its name in the toolbox. When you double-click on the *Polygon centroids* algorithm, you will see the following dialog.



All algorithms have a similar interface, which basically contains input parameters that you have to fill, and outputs that you have to select where to store. In this case, the only input we have is a vector layer with polygons.

Select the *Polygons* layer as input. The algorithm has a single output, which is the centroids layer. There are two options to define where a data output is saved: enter a filepath or save it to a temporary filename

In case you want to set a destination and not save the result in a temporary file, the format of the output is defined by the filename extension. To select a format, just select the corresponding file extension (or add it if you are directly

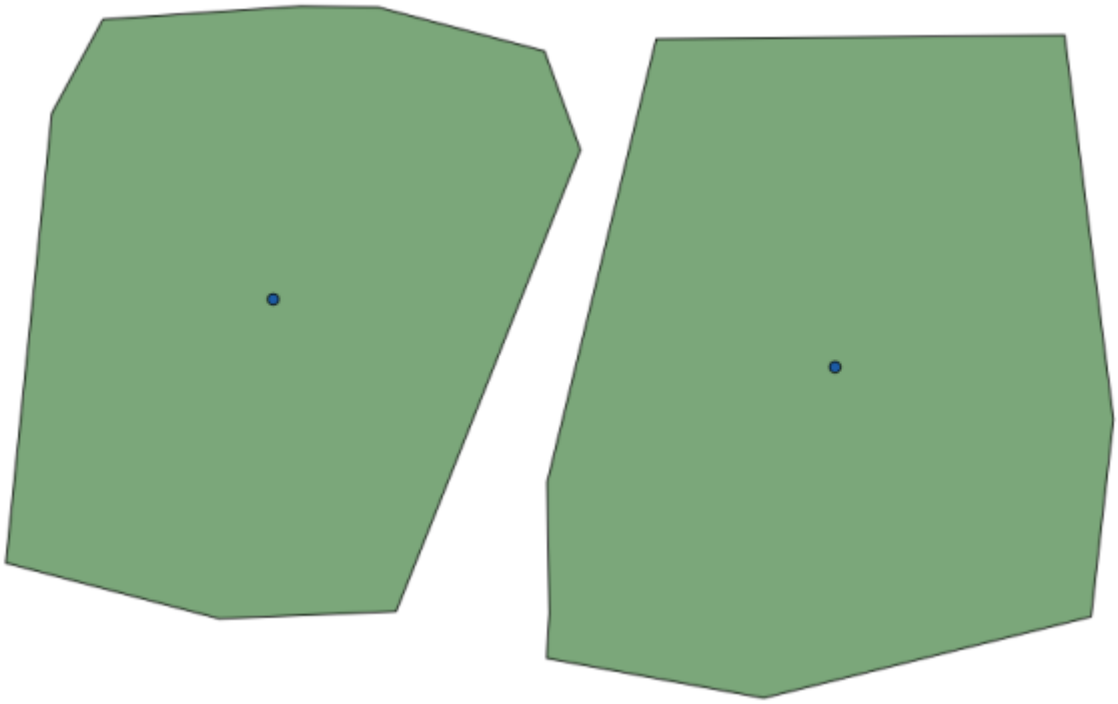
typing the filepath instead). If the extension of the filepath you entered does not match any of the supported ones, a default extension (usually `.dbf` for tables, `.tif` for raster layers and `.shp` for vector ones) will be appended to the filepath and the file format corresponding to that extension will be used to save the layer or table.

In all the exercises in this guide, we will be saving results to a temporary file, since there is no need to save them for a later use. Feel free to save them to a permanent location if you want to.

Aviso: Temporary files are deleted once you close QGIS. If you create a project with an output that was saved as a temporary output, QGIS will complain when you try to open back the project later, since that output file will not exist.

Once you have configured the algorithm dialog, press *Run* to run the algorithm.

You will get the following output.



The output has the same CRS as the input. Geoalgorithms assumes all input layers share the same CRS and do not perform any reprojection. Except in the case of some special algorithms (for instance, reprojection ones), the outputs will also have that same CRS. We will see more about this soon.

Try yourself saving it using different file formats (use, for instance, `shp` and `geojson` as extensions). Also, if you do not want the layer to be loaded in QGIS after it is generated, you can check off the checkbox that is found below the output path box.

17.5 Mais algoritmos e tipo de dados

Nota: In this lesson we will run three more algorithms, learn how to use other input types, and configure outputs to be saved to a given folder automatically.

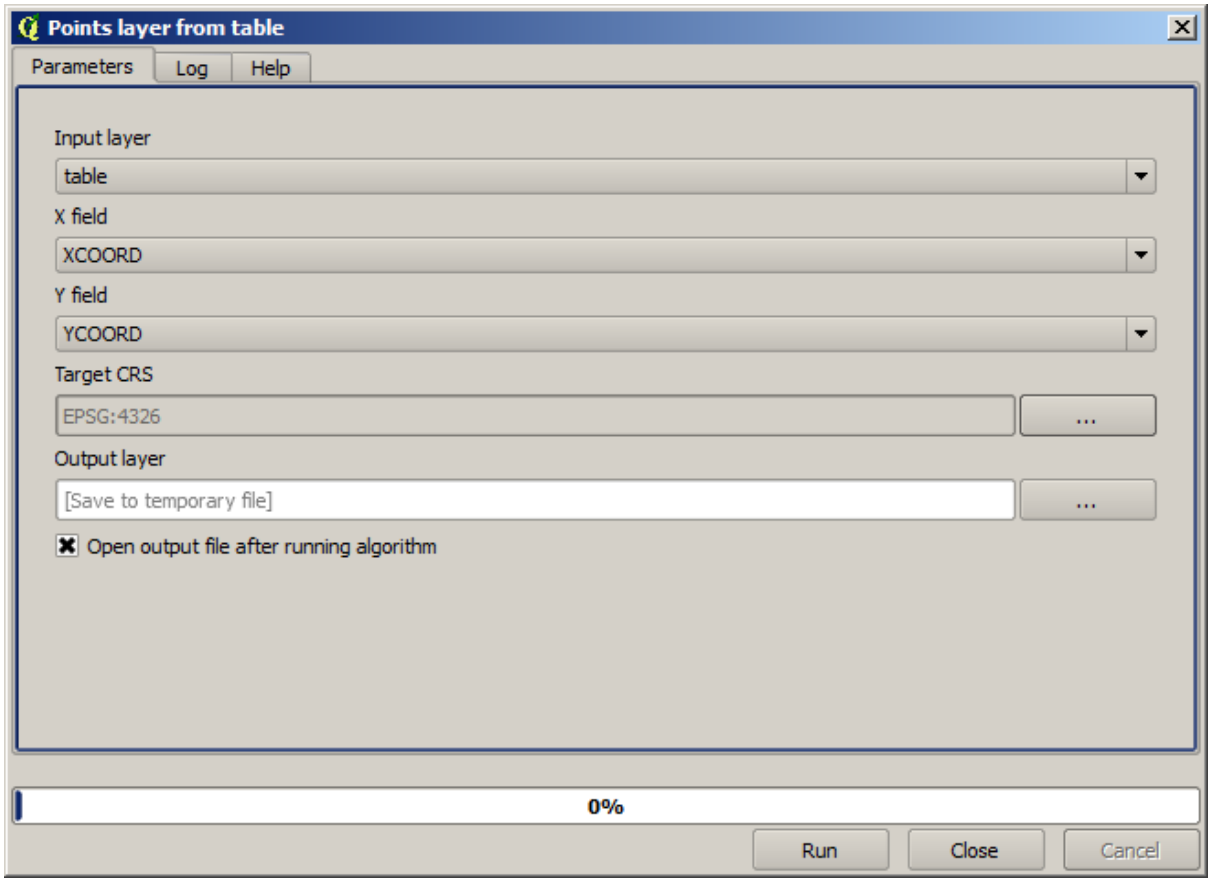
For this lessons we will need a table and a polygons layer. We are going to create a points layer based on coordinates in the table, and then count the number of points in each polygon. If you open the QGIS project corresponding to this lesson, you will find a table with X and Y coordinates, but you will find no polygons layer. Don't worry, we will create it using a processing geoalgorithm.

The first thing we are going to do is to create a points layer from the coordinates in the table, using the *Points layer from table* algorithm. You now know how to use the search box, so it should not be hard for you to find it. Double–click on it to run it and get to its following dialog.

This algorithm, like the one from the previous lesson, just generates a single output, and it has three inputs:

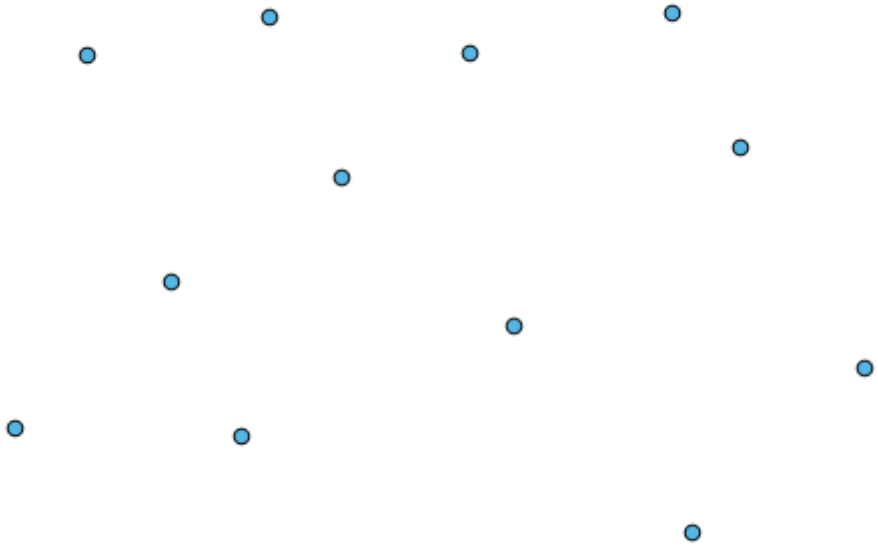
- *Table*: the table with the coordinates. You should select here the table from the lesson data.
- *X and Y fields*: these two parameters are linked to the first one. The corresponding selector will show the name of those fields that are available in the selected table. Select the *XCOORD* field for the *X* parameter, and the *YYCOORD* field for the *Y* parameter.
- *CRS*: Since this algorithm takes no input layers, it cannot assign a CRS to the output layer based on them. Instead, it asks you to manually select the CRS that the coordinates in the table use. Click on the button on the left–hand side to open the QGIS CRS selector, and select EPSG:4326 as the output CRS. We are using this CRS because the coordinates in the table are in that CRS.

Your dialog should look like this.

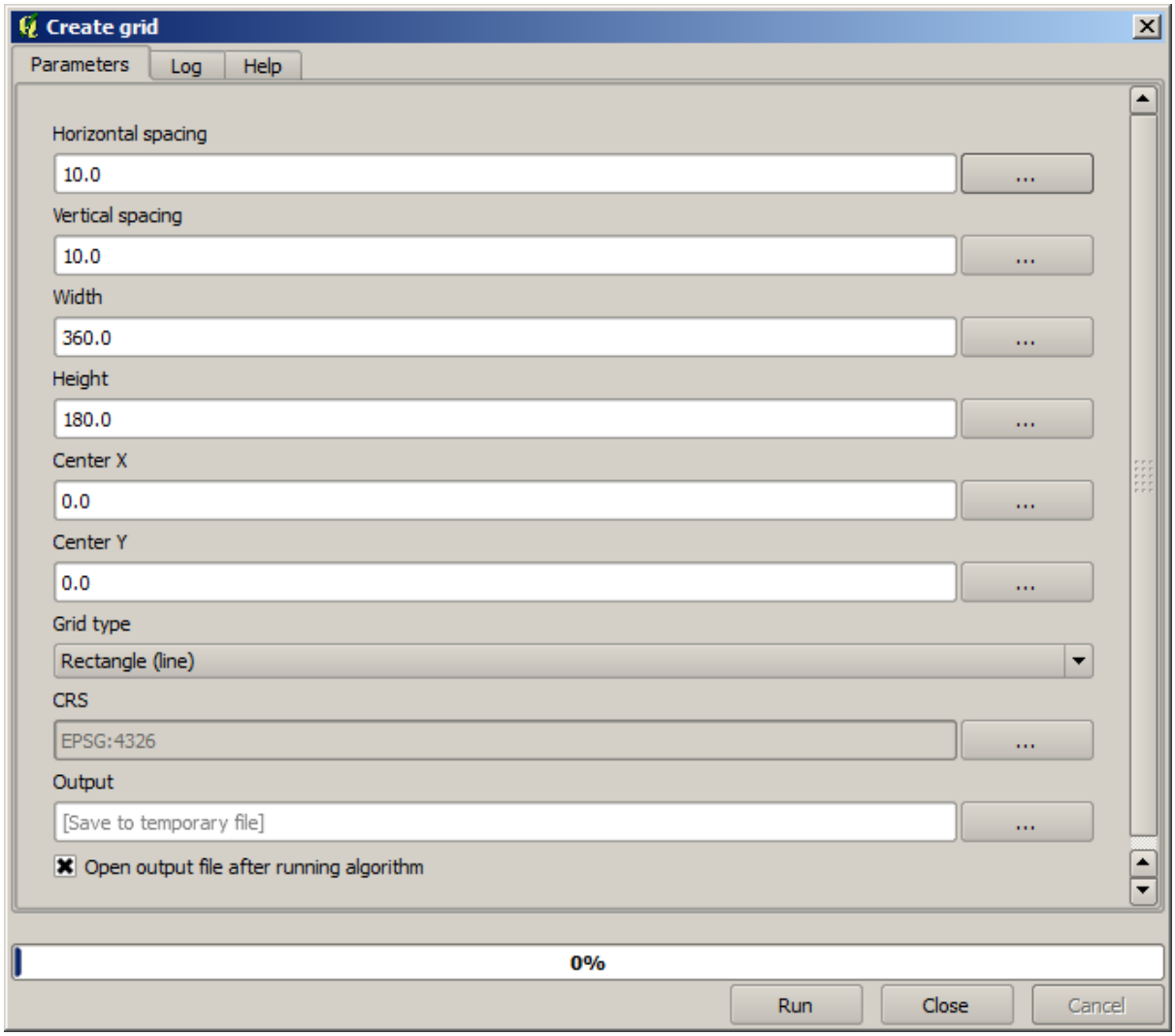


Now press the *Run* button to get the following layer (you may need to zoom full to reenter the map around the newly

created points):

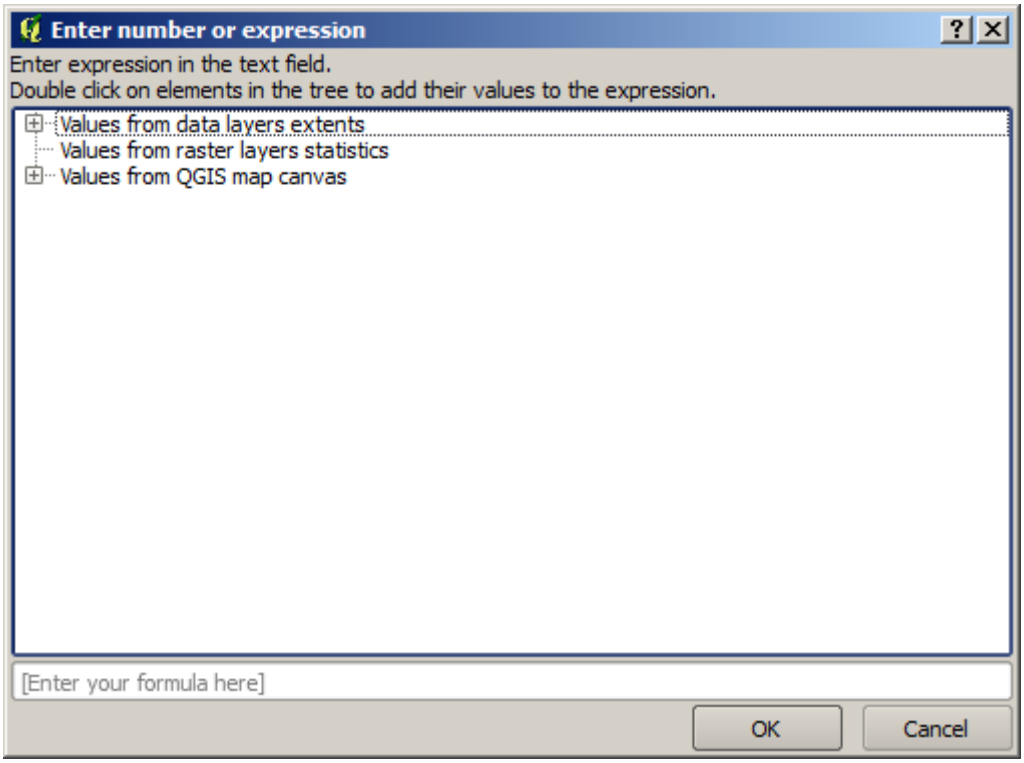


The next thing we need is the polygon layer. We are going to create a regular grid of polygons using the *Create grid* algorithm, which has the following parameters dialog.



Aviso: The options are simpler in recent versions of QGIS; you just need to enter min and max for X and Y (suggested values: -5.696226,-5.695122,40.24742,40.248171)

The inputs required to create the grid are all numbers. When you have to enter a numerical value, you have two options: typing it directly on the corresponding box or clicking the button on the right–hand side to get to a dialog like the one shown next.



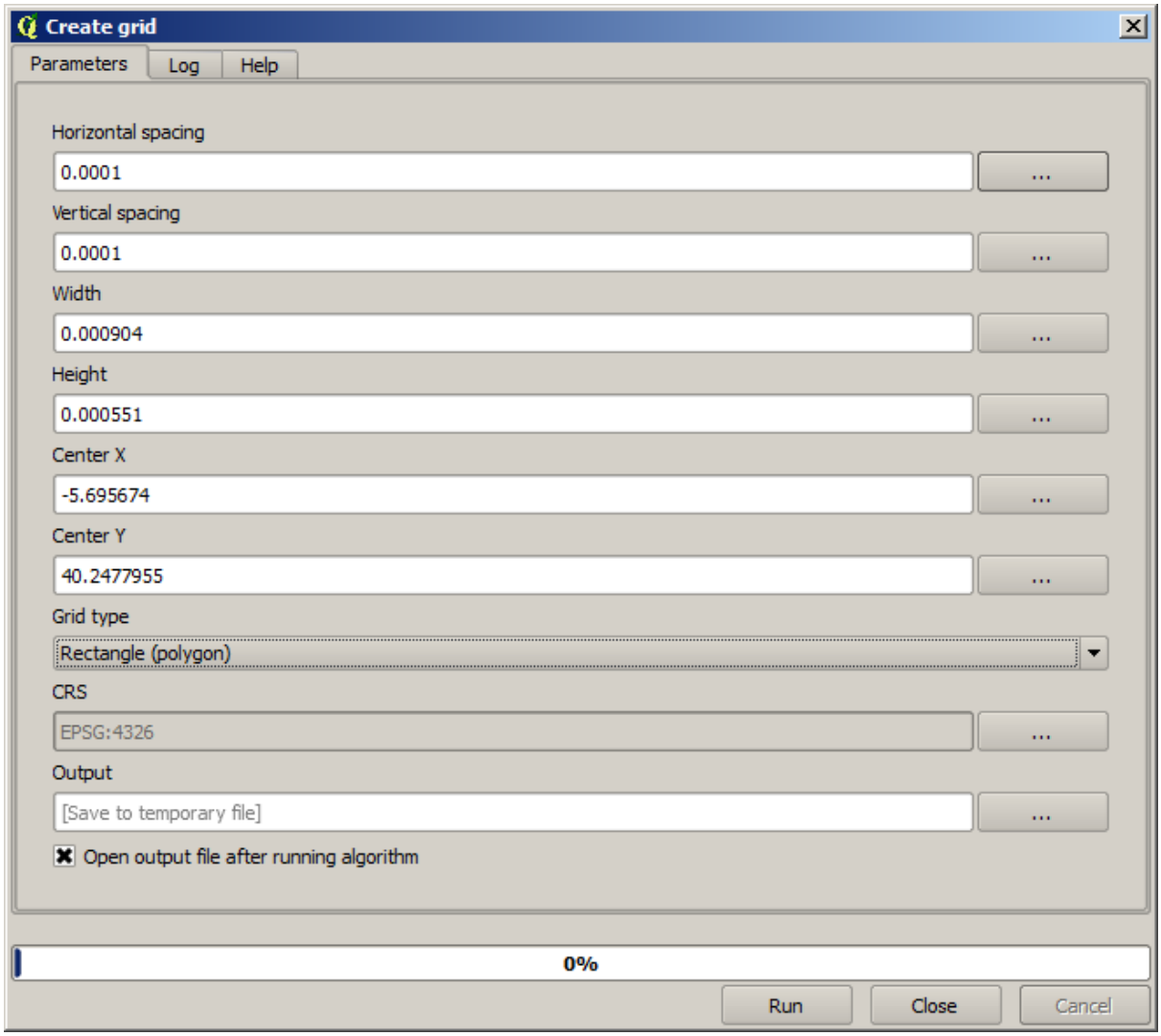
The dialog contains a simple calculator, so you can type expressions such as `11 * 34.7 + 4.6`, and the result will be computed and put in the corresponding text box in the parameters dialog. Also, it contains constants that you can use, and values from other layers available.

In this case, we want to create a grid that covers the extent of the input points layer, so we should use its coordinates to calculate the center coordinate of the grid and its width and height, since those are the parameters that the algorithm takes to create the grid. With a little bit of math, try to do that yourself using the calculator dialog and the constants from the input points layer.

Select *Rectangles (polygons)* in the *Type* field.

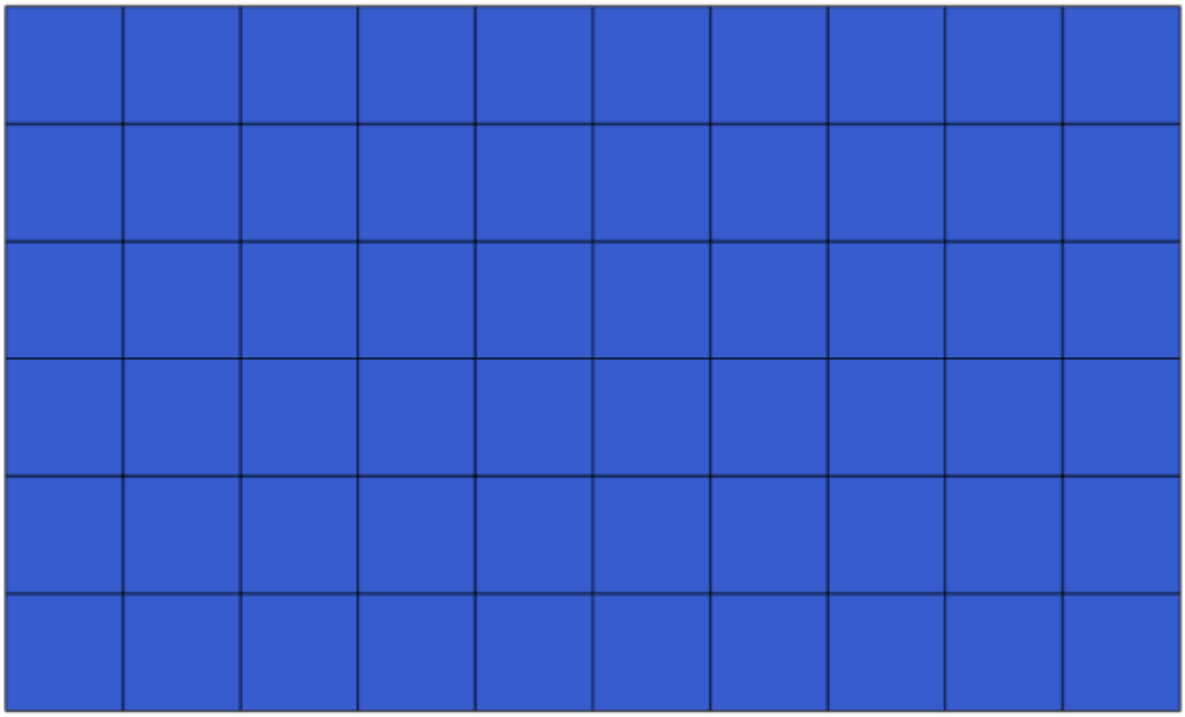
As in the case of the last algorithm, we have to enter the CRS here as well. Select EPSG:4326 as the target CRS, as we did before.

In the end, you should have a parameters dialog like this:

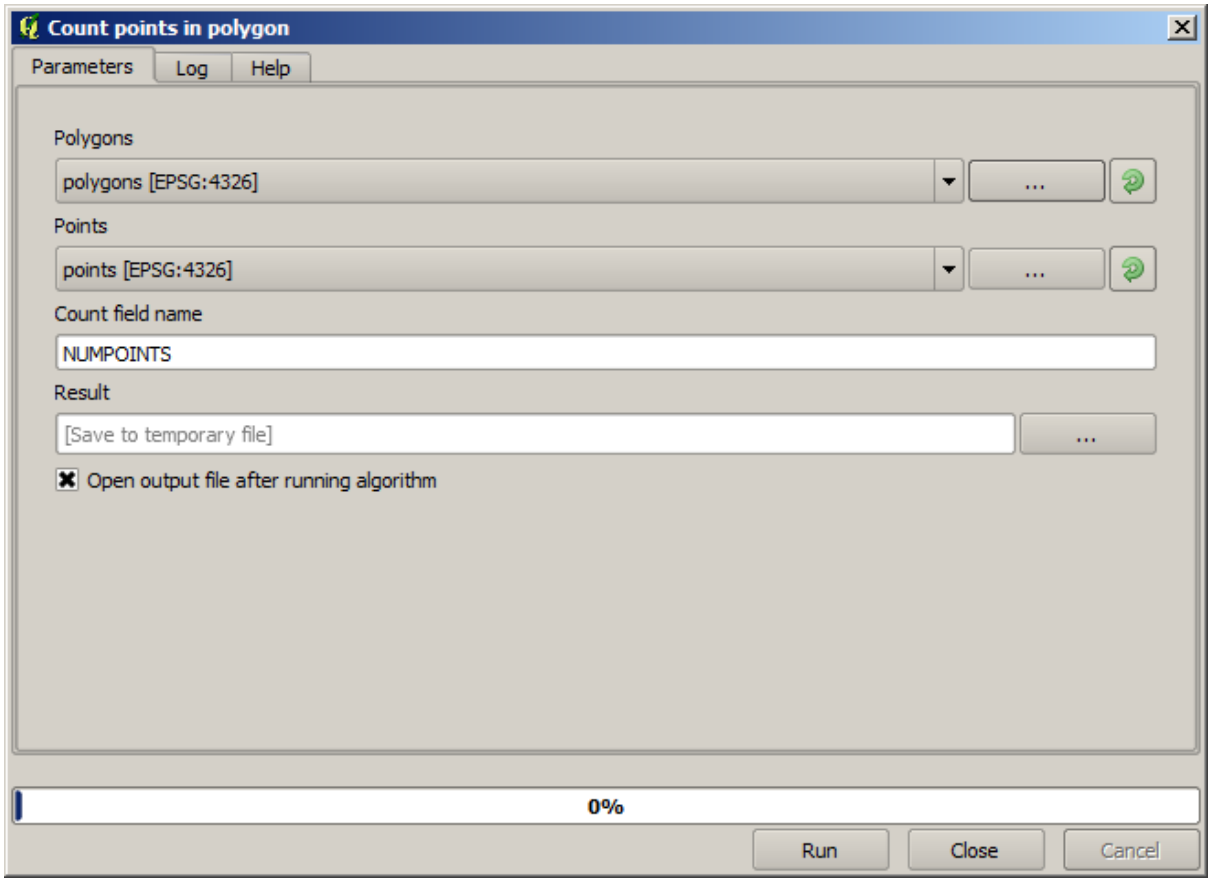


(Better add one spacing on the width and height: Horizontal spacing: 0.0001, Vertical spacing: 0.0001, Width: 0.001004, Height: 0.000651, Center X: -5.695674, Center Y: 40.2477955) The case of X center is a bit tricky, see: -5.696126+((-5.695222+ 5.696126)/2)

Press *Run* and you will get the graticule layer.

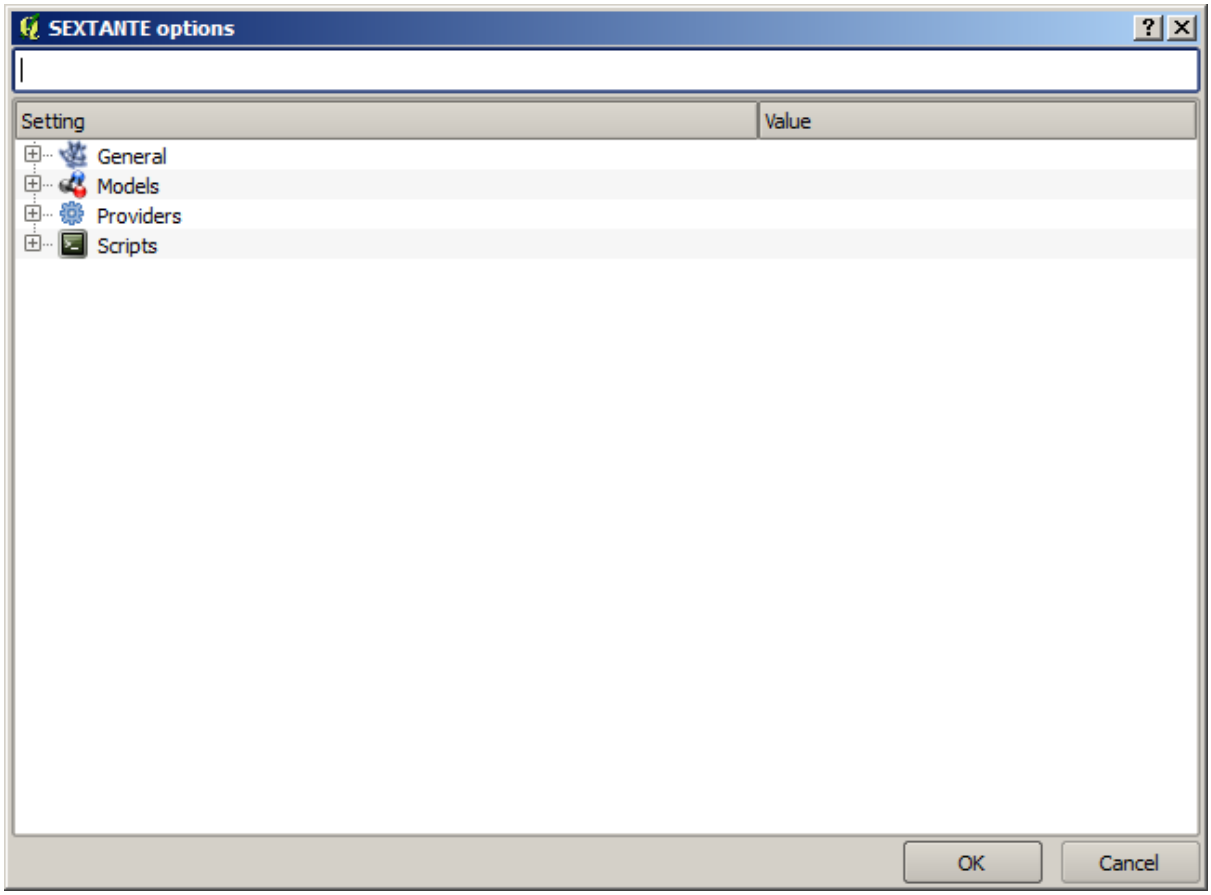


The last step is to count the points in each one of the rectangles of that graticule. We will use the *Count points in polygons* algorithm.

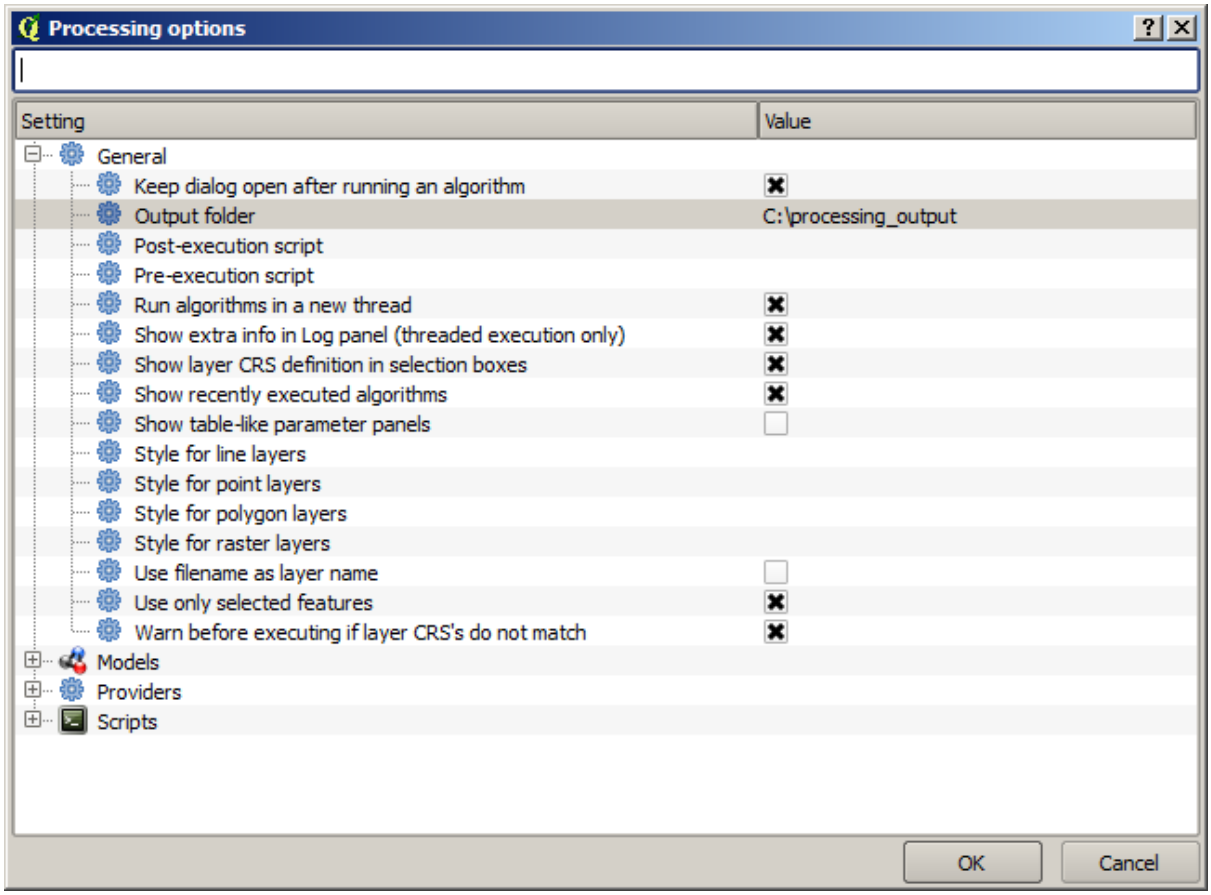


Now we have the result we were looking for.

Before finishing this lesson, here is a quick tip to make your life easier in case you want to persistently save your data. If you want all your output files to be saved in a given folder, you do not have to type the folder name each time. Instead, go to the processing menu and select the *Options and configuration* item. It will open the configuration dialog.



In the *Output folder* entry that you will find in the *General* group, type the path to your destination folder.



Now when you run an algorithm, just use the filename instead of the full path. For instance, with the configuration shown above, if you enter `graticule.shp` as the output path for the algorithm that we have just used, the result will be saved in `D:\processing_output\graticule.shp`. You can still enter a full path in case you want a result to be saved in a different folder.

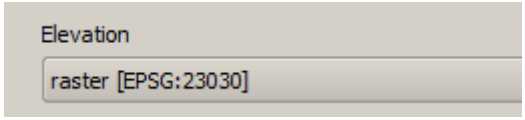
Try yourself the *Create grid* algorithm with different grid sizes, and also with different types of grids.

17.6 SRC. Reprojetar

Nota: In this lesson we will discuss how Processing uses CRSs. We will also see a very useful algorithm: reprojecting.

CRS's are a great source of confusion for QGIS Processing users, so here are some general rules about how they are handled by geoalgorithms when creating a new layer.

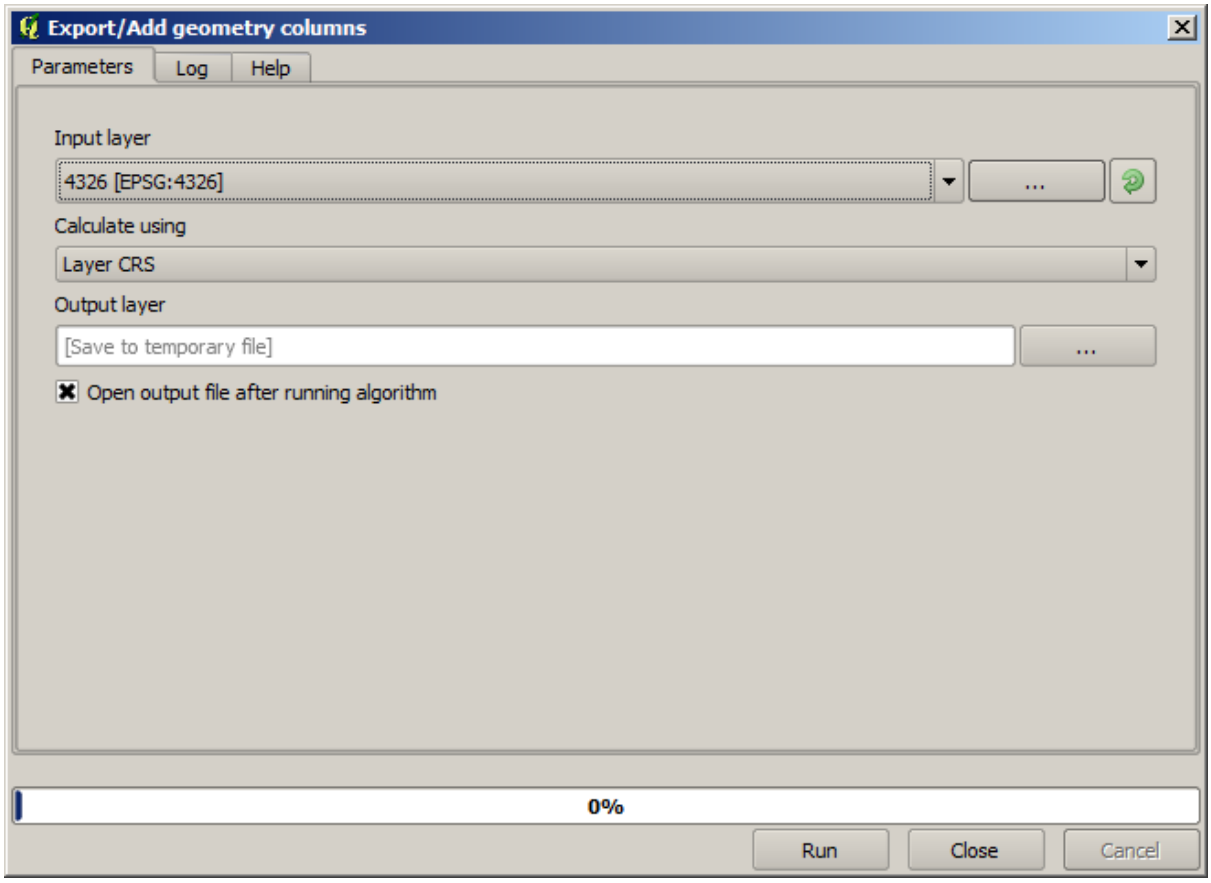
- If there are input layers, it will use the CRS of the first layer. This is assumed to be the CRS of all input layers, since they should have the same one. If you use layers with unmatching CRS's, QGIS will warn you about it. Notice that the CRS of input layers is shown along with its name in the parameters dialog.



- If there are no input layer, it will use the project CRS, unless the algorithm contains a specific CRS field (as it happened in the last lesson with the graticule algorithm)

Open the project corresponding to this lesson and you will see two layers named 23030 and 4326. They both contain the same points, but in different CRSs (EPSG:23030 and EPSG:4326). They appear in the same place because QGIS is reprojecting on the fly to the project CRS (EPSG:4326), but they are not actually the same layer.

Open the *Export/Add geometry columns* algorithm.



This algorithm add new columns to the attributes table of a vector layer. The content of the columns depend on the type of geometry of the layer. In the case of points, it adds new columns with the X and Y coordinates of each point.

In the list of available layers that you will find in the input layer field, you will see each one with its corresponding CRS. That means that, although they appear in the same place in your canvas, they will be treated differently. Select the 4326 layer.

The other parameter of the algorithm allows to set how the algorithm uses coordinates to calculate the new value that it will add to the resulting layers. Most algorithms do not have an option like that, and just use the coordinates directly. Select the *Layer CRS* option to just use coordinates as they are. This is how almost all geoalgorithms work.

You should get a new layer with exactly the same points as the other two layers. If you right click on the name of the layer and open its properties, you will see that it shares the same CRS of the input layer, that is, EPSG:4326. When the layer is loaded into QGIS, you will not be asked to enter the CRS of the layer, since QGIS already knows about it.

If you open the attributes table of the new layer you will see that it contains two new fields with the X and Y coordinates of each point.

	ID	PT_NUM_A	PT_ST_A	xcoord	ycoord
0	1	1.100000	a	-5.695426	40.248071
1	2	2.200000	b	-5.695885	40.247622
2	3	3.300000	c	-5.695406	40.247520
3	4	4.400000	a	-5.695222	40.247694
4	5	5.500000	b	-5.695642	40.248030
5	6	6.600000	a	-5.695855	40.248067
6	7	7.700000	b	-5.696049	40.248028
7	8	8.800000	c	-5.696126	40.247629
8	9	9.900000	a	-5.695961	40.247786
9	10	11.000000	b	-5.695353	40.247929
10	11	12.100000	a	-5.695595	40.247739
11	12	13.200000	b	-5.695779	40.247896

Those coordinate values are given in the layer CRS, since we chose that option. However, even if you choose another option, the output CRS of the layer would have been the same, since the input CRS is used to set the CRS of the output layer. Choosing another option will cause the values to be different, but not the resulting point to change or the CRS of the output layer to be different to the CRS of the input one.

Now do the same calculation using the other layer. You should find the resulting layer rendered exactly in the same place as the other ones, and it will have the EPSG:23030 CRS, since that was the one of the input layer.

If you go to its attribute table, you will see values that are different to the ones in the first layer that we created.

	ID	PT_NUM_A	PT_ST_A	xcoord	ycoord
0	1	1.100000	a	270839.655869	4458983.162670
1	2	2.200000	b	270799.116425	4458934.552874
2	3	3.300000	c	270839.468187	4458921.978139
3	4	4.400000	a	270855.745301	4458940.799487
4	5	5.500000	b	270821.164389	4458979.173980
5	6	6.600000	a	270803.157564	4458983.848803
6	7	7.700000	b	270786.542791	4458980.047841
7	8	8.800000	c	270778.601980	4458935.968837
8	9	9.900000	a	270793.142411	4458952.931700
9	10	11.000000	b	270845.414756	4458967.311298
10	11	12.100000	a	270824.166376	4458946.784250
11	12	13.200000	b	270809.035643	4458964.649799

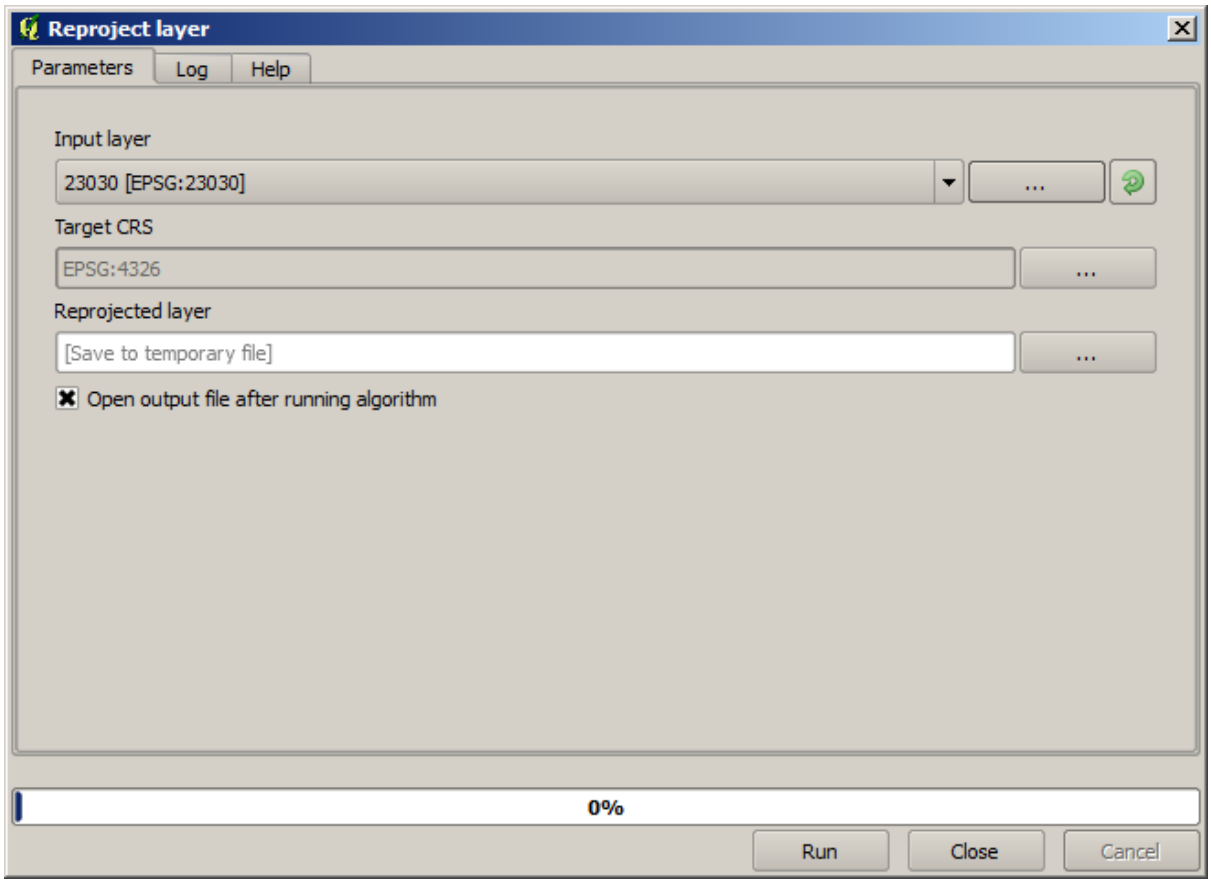
This is because the original data is different (it uses a different CRS), and those coordinates are taken from it.

What should you learn from this? The main idea behind these examples is that geoalgorithms use the layer as it is in its original data source, and completely ignore the reprojections that QGIS might be doing before rendering. In other words, do not trust what you see in the canvas, but always have in mind that the original data will be used. That is not so important in this case, since we are just using one single layer at a time, but in an algorithm that needs several of them (such as a clip algorithm), layers that appear to match or overlay might be very far one from each other, since they might have different CRSs.

Algorithms performs no reprojection (except in the reprojection algorithm that we will see next), so it is up to you to make sure that layers have matching CRS's.

An interesting module that deals with CRS's is the reprojection one. It represents a particular case, since it has an input layer (the one to reproject), but it will not use its CRS for the output one.

Open the *Reproject layer* algorithm.



Select any of the layers as input, and select EPSG:23029 as the destination CRS. Run the algorithm and you will get a new layer, identical to the input one, but with a different CRS. It will appear on the same region of the canvas, like the other ones, since QGIS will reproject it on the fly, but its original coordinates are different. You can see that by running the *Export/Add geometry columns* algorithm using this new layer as input, and veryfing that the added coordinates are different to the ones in the attribute tables of both of the two layers that we had computed before.

17.7 Selecção

Nota: In this lesson we will see how processing algorithms handle selections in vector layers that are used as inputs, and how to create a selection using a particular type of algorithm.

Unlike other analysis plugins in QGIS, you will not find in processing geoalgorithms any «Use only selected features» checkbox or similar. The behaviour regarding selection is set for the whole plugin and all its algorithms, and not for each algorithm execution. Algorithms follow the following simple rules when using a vector layer.

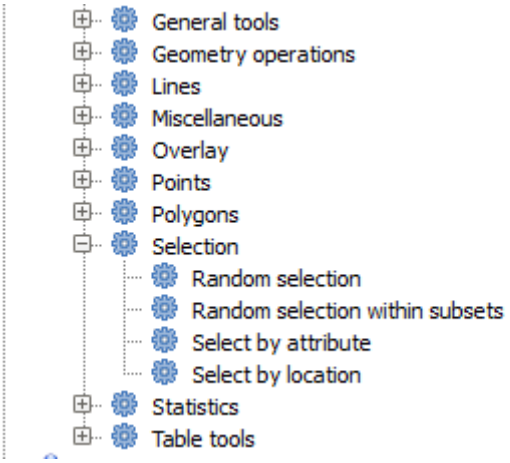
- If the layer has a selection, only selected features are used.
- If there is no selection, all features are used.

Please note that you can change this behaviour by unselecting the relevant option in the *Processing* ► *Options* ► *General* menu.

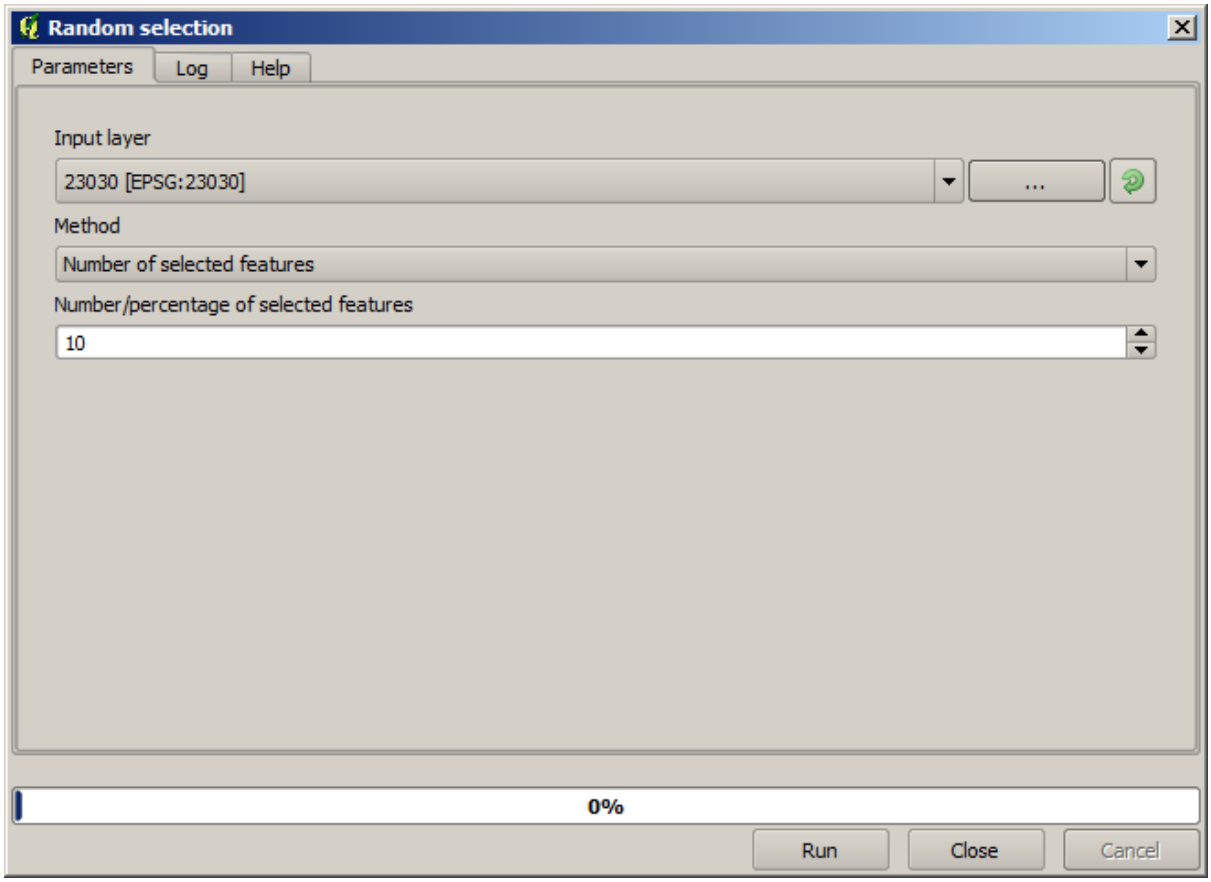
You can test that yourself by selecting a few points in any of the layers that we used in the last chapter, and running the reprojection algorithm on them. The reprojected layer that you will obtain will contain only those points that

were selected, unless there was no selection, which will cause the resulting layer to contain all points from the original layer.

To make a selection, you can use any of the available methods and tools in QGIS. However, you can also use a geoalgorithm to do so. Algorithms for creating a selection are found in the toolbox under *Vector/Selection*



Open the *Random selection* algorithm.



Leaving the default values, it will select 10 points from the current layer.

You will notice that this algorithm does not produce any output, but modifies the input layer (not the layer itself, but its selection). This is an uncommon behaviour, since all the other algorithms will produce new layers and not alter the input layers.

Since the selection is not part of the data itself, but something that only exist within QGIS, these selection algorithms only must be used selecting a layer that is open in QGIS, and not with the file selection option that you can find in the corresponding parameter value box.

The selection we have just made, like most of the ones created by the rest of the selection algorithms, can also be done manually from QGIS, so you might be wondering what is the point on using an algorithm for that. Although now this might not make much sense to you, we will later see how to create models and scripts. If you want to make a selection in the middle of a model (which defines a processing workflow), only a geoalgorithm can be added to a model, and other QGIS elements and operations cannot be added. That is the reason why some processing algorithms duplicate functionality that is also available in other QGIS elements.

By now, just remember that selections can be made using processing geoalgorithms, and that algorithms will only use the selected features if a selection exists, or all features otherwise.

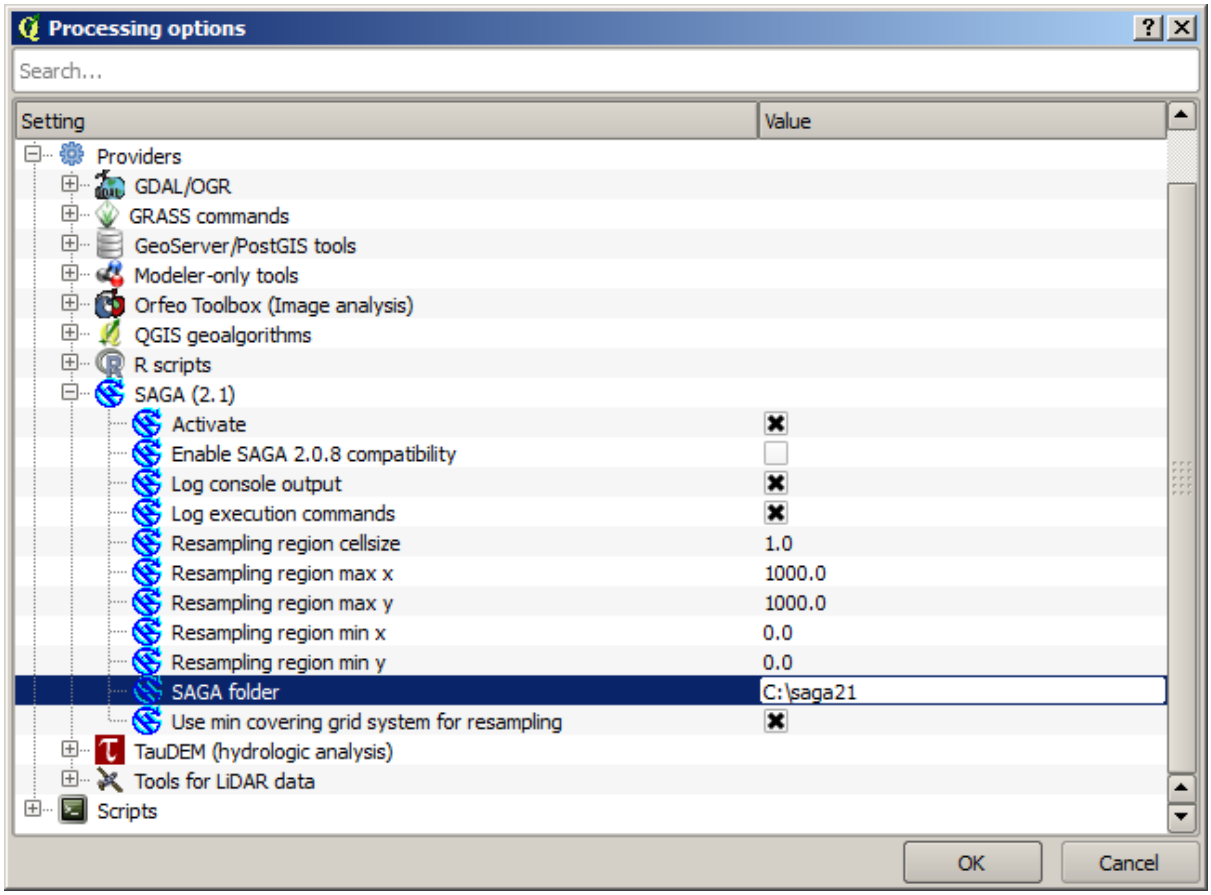
17.8 A correr algoritmo externo

Nota: In this lesson we will see how to use algorithms that depend on a third-party application, particularly SAGA, which is one of the main algorithm providers.

All the algorithms that we have run so far are part of processing framework. That is, they are *native* algorithms implemented in the plugin and run by QGIS just like the plugin itself is run. However, one of the greatest features of the processing framework is that it can use algorithms from external applications and extend the possibilities of those applications. Such algorithms are wrapped and included in the toolbox, so you can easily use them from QGIS, and use QGIS data to run them.

Some of the algorithms that you see in the simplified view require third party applications to be installed in your system. One algorithm provider of special interest is SAGA (System for Automated Geospatial Analysis). First, we need to configure everything so QGIS can correctly call SAGA. This is not difficult, but it's important to understand how it works. Each external application has its own configuration, and later in this same manual we will talk about some of the other ones, but SAGA is going to be our main backend, so we will discuss it here.

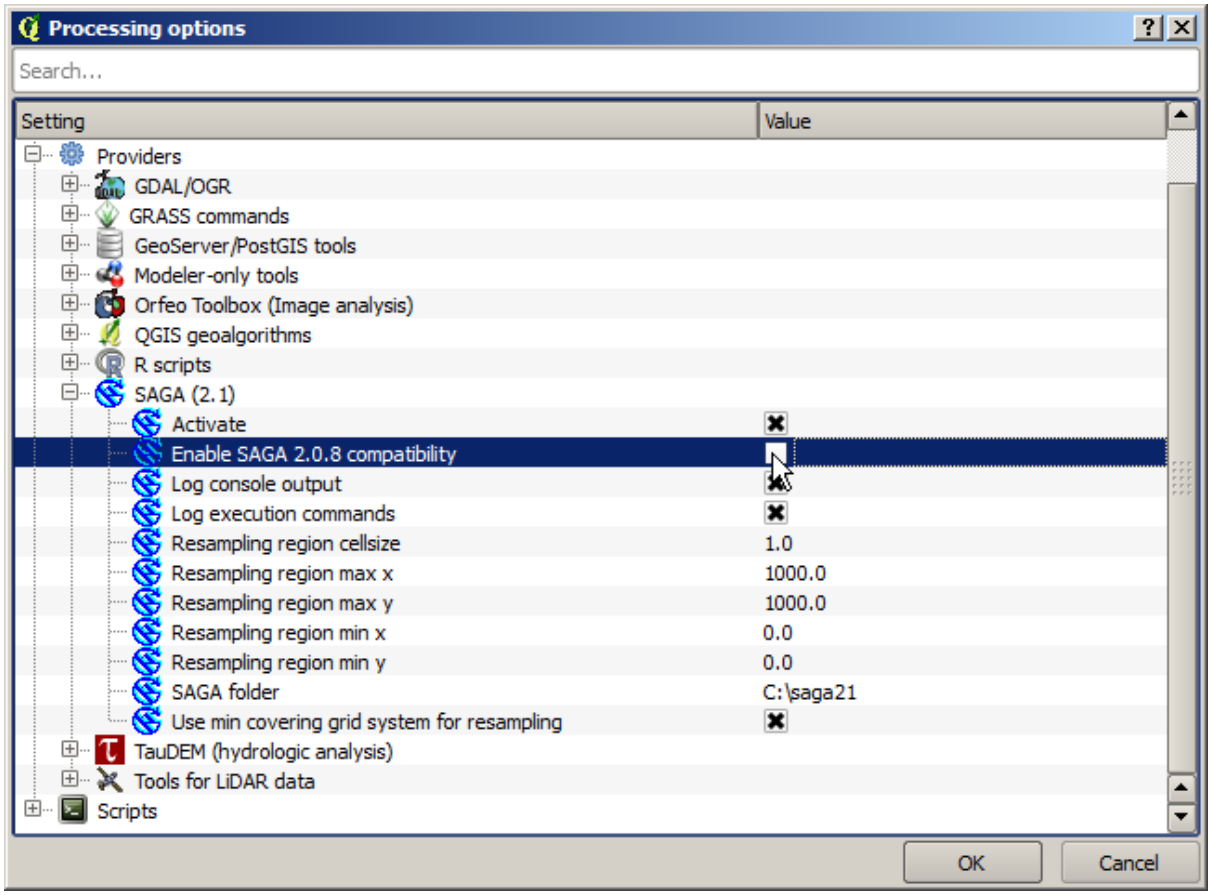
If you are on Windows, the best way to work with external algorithms is to install QGIS using the standalone installer. It will take care of installing all the needed dependencies, including SAGA, so if you have used it, there is nothing else to do. You can open the settings dialog and go to the *Providers/SAGA* group.



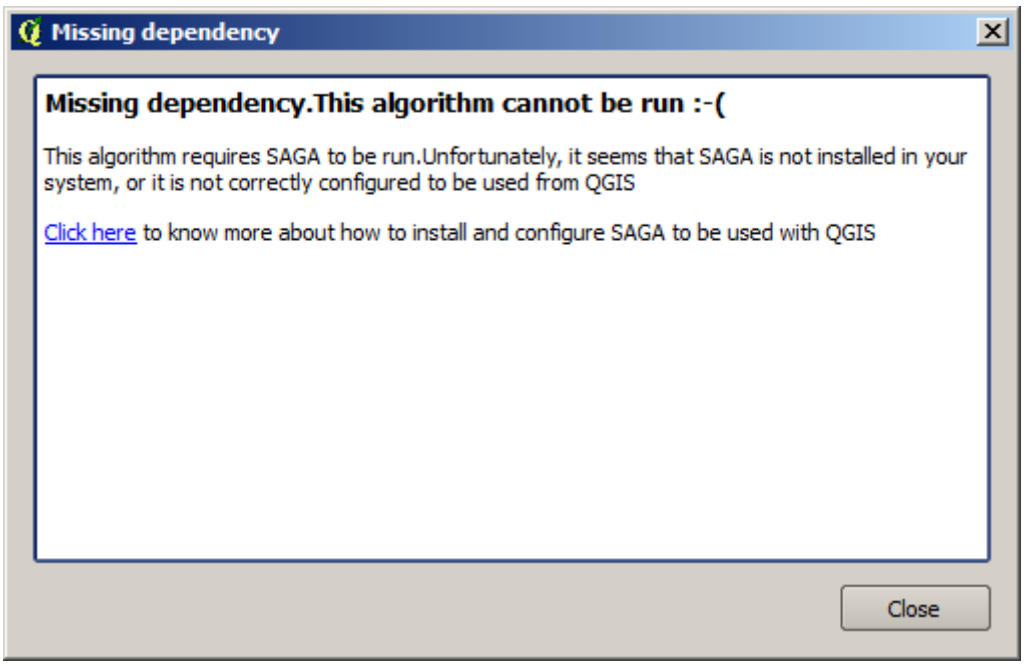
The SAGA path should already be configured and pointing to the folder where SAGA is installed.

If you have installed QGIS not using the standalone installer, then you must enter the path to your SAGA installation (which you must have installed separately) there. The required version is SAGA 2.1 [this is changing according to the releases of SAGA].

In case you are using Linux, you do not have to set the path to your SAGA installation in the processing configuration. Instead, you must install SAGA and make sure that the SAGA folder is in PATH, so it can be called from the console (just open a console and type `saga_cmd` to check it). Under Linux, the target version for SAGA is also 2.1, but in some installations (such as the OSGeo Live DVD) you might have just 2.0.8 available. There are some 2.1 packages available, but they are not commonly installed and might have some issues, so if you prefer to use the more common and stable 2.0.8, you can do it by enabling 2.0.8 compatibility in the configuration dialog, under the *SAGA* group.

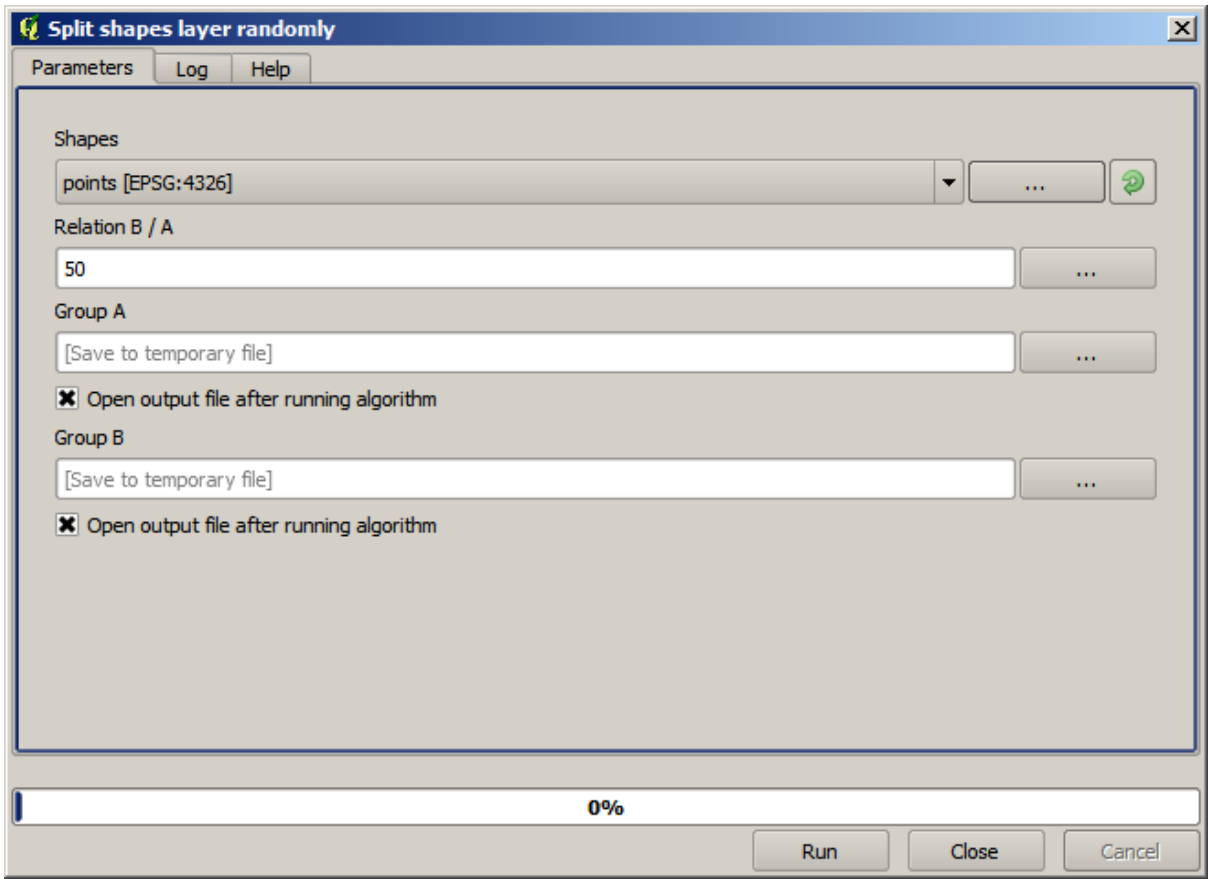


Once SAGA is installed, you can launch a SAGA algorithm double clicking on its name, as with any other algorithm. Since we are using the simplified interface, you do not know which algorithms are based on SAGA or in another external application, but if you happen to double–click on one of them and the corresponding application is not installed, you will see something like this.

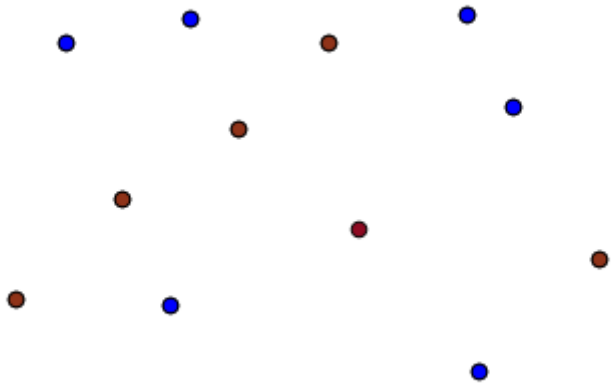


In our case, and assuming that SAGA is correctly installed and configured, you should not see this window, and you will get to the parameters dialog instead.

Let's try with a SAGA–based algorithm, the one called *Split shapes layer randomly*.



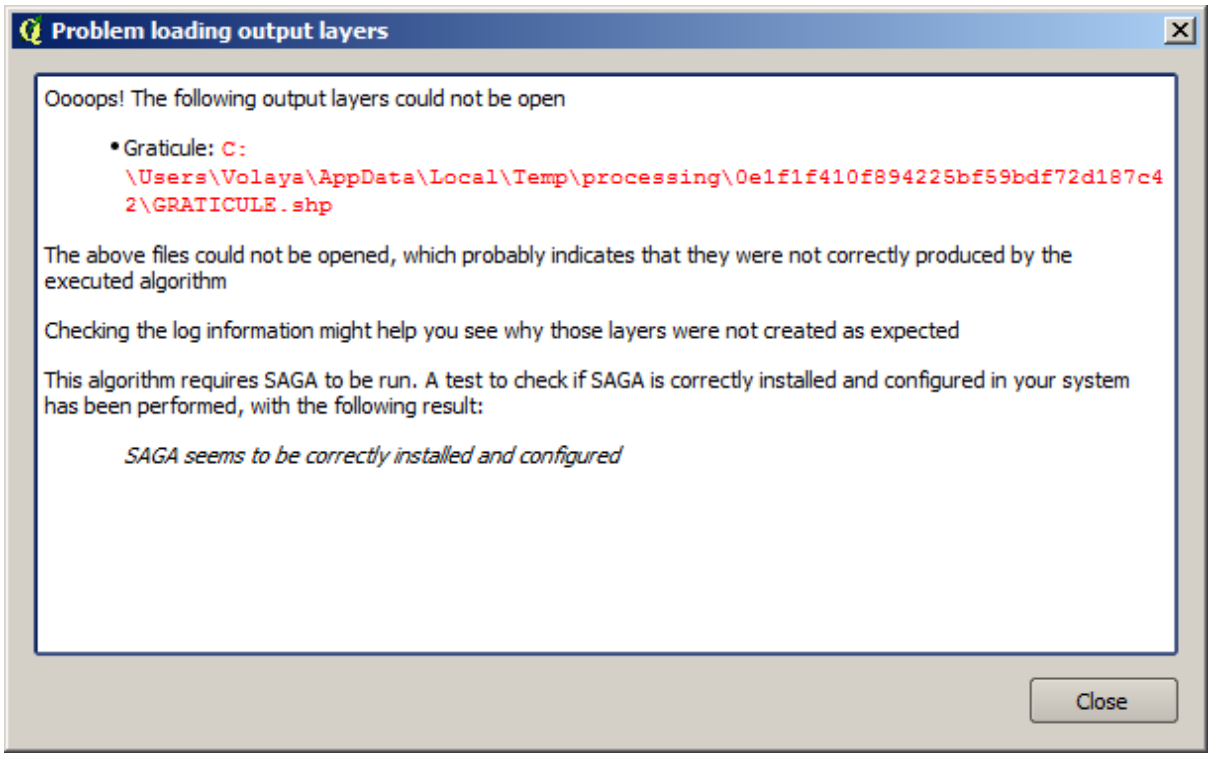
Use the points layer in the project corresponding to this lesson as input, and the default parameter values, and you will get something like this (the split is random, so your result might be different).



The input layer has been split in two layers, each one with the same number of points. This result has been computed by SAGA, and later taken by QGIS and added to the QGIS project.

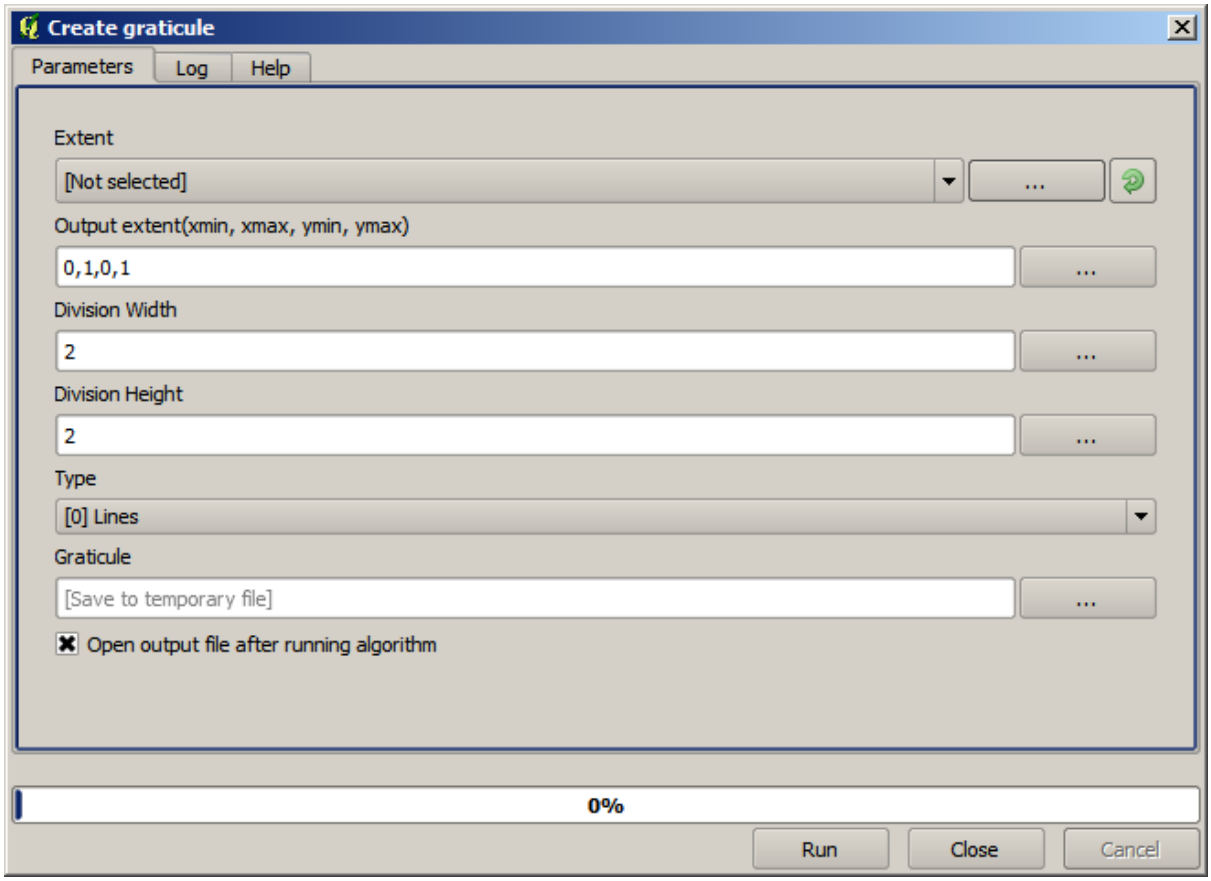
If all goes fine, you will not notice any difference between this SAGA–based algorithm and one of the others that we have previously run. However, SAGA might, for some reason, not be able to produce a result and not generate the

file that QGIS is expecting. In that case, there will be problems adding the result to the QGIS project, and an error message like this will be shown.



This kind of problems might happen, even if SAGA (or any other application that we are calling from the processing framework) is correctly installed, and it is important to know how to deal with them. Let's produce one of those error messages.

Open the *Create graticule* algorithm and use the following values.



We are using width and height values that is larger than the specified extent, so SAGA cannot produce any output. In other words, the parameter values are wrong, but they are not checked until SAGA gets them and tries to create the graticule. Since it cannot create it, it will not produce the expected layer, and you will see the error message shown above.

Nota: In SAGA >= 2.2.3, the command will adjust automatically wrong input data, so you'll not get an error. To provoke an error, use negative values for division.

Understanding this kind of problems will help you solve them and find an explanation to what is happening. As you can see in the error message, a test is performed to check that the connection with SAGA is working correctly, indicating you that there might be a problem in how the algorithm was executed. This applies not only to SAGA, but also to other external applications as well.

In the next lesson we will introduce the processing log, where information about commands run by geoalgorithms is kept, and you will see how to get more detail when issues like this appear.

17.9 Registo de processamento

Nota: This lesson describes the processing log.

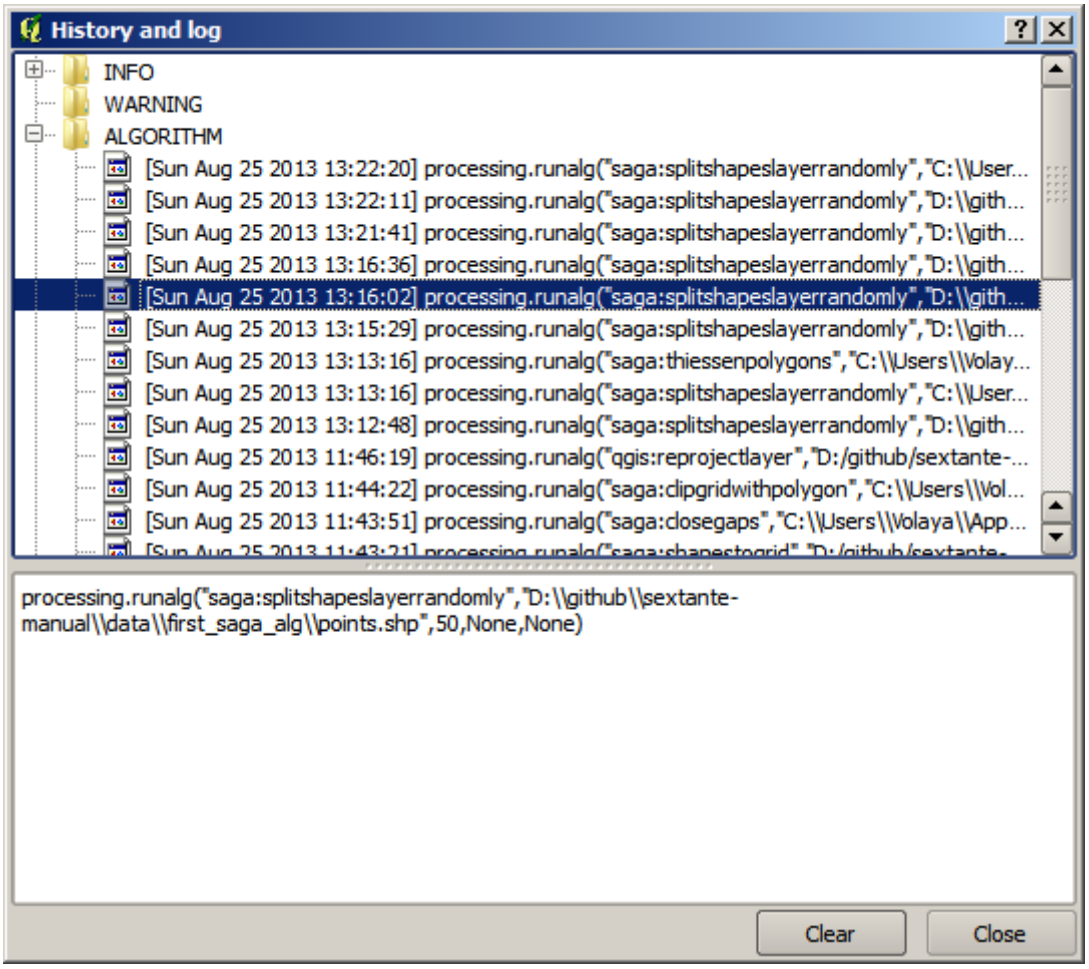
All the analysis performed with the processing framework is logged in QGIS logging system. This allows you to know more about what has been done with the processing tools, to solve problems when they happen, and also to re–run previous operations, since the logging system also implements some interactivity.

To open the log, click on the balloon at the bottom right, on the QGIS status bar. Some algorithms might leave here information about their execution. For instance, those algorithms that call an external application usually log the console output of that application to this entry. If you have a look at it, you will see that the output of the SAGA algorithm that we just run (and that fails to execute because input data was not correct) is stored here.

This is helpful to understand what is going on. Advanced users will be able to analyze that output to find out why the algorithm failed. If you are not an advanced user, this will be useful for others to help you diagnose the problem you are having, which might be a problem in the installation of the external software or an issue with the data you provided.

Even if the algorithm could be executed, some algorithms might leave warnings in case the result might not be right. For instance, when executing an interpolation algorithm with a very small amount of points, the algorithm can run and will produce a result, but it is likely that it will not be correct, since more points should be used. It's a good idea to regularly check for this type of warnings if you are not sure about some aspect of a given algorithm.

From the *Processing* ► *History…* menu, you'll find *algorithms* that are executed, regardless of they are executed from the GUI or from the console (which will be explained later in this manual). The execution is stored in this dialog as a console call. That means that everytime you run an algorithm, a console command is added to the log, and you have the full history of your working session. Here is how that history looks like:



This can be very useful when starting working with the console, to learn about the syntax of algorithms. We will use it when we discuss how to run analysis commands from the console.

17.9.1 ??? Follow Along

The history is also interactive, and you can run any previous algorithm just by double-clicking on its entry. This is an easy way of replicating the work we already did before.

For instance, try the following:

1. Open the data corresponding to the first chapter of this manual and run the algorithm explained there.
2. Now go to the History dialog and locate the last algorithm in the list, which corresponds to the algorithm you have just run.
3. Double-click on it and a new result should be produced, just like when you run it using the normal dialog and calling it from the toolbox.

17.9.2 ??? Follow Along

You can also modify the algorithm.

1. Copy the algorithm call
2. Open the *Plugins* ► *Python console*
3. Paste your copy to run the analysis; change the parameters at will.
4. To display the resulting file, you may want to replace `processing.run` in the pasted command with `processing.runAndLoadResults`.

17.10 The raster calculator. No-data values

Nota: In this lesson we will see how to use the raster calculator to perform some operations on raster layers. We will also explain what are no–data values and how the calculator and other algorithms deal with them

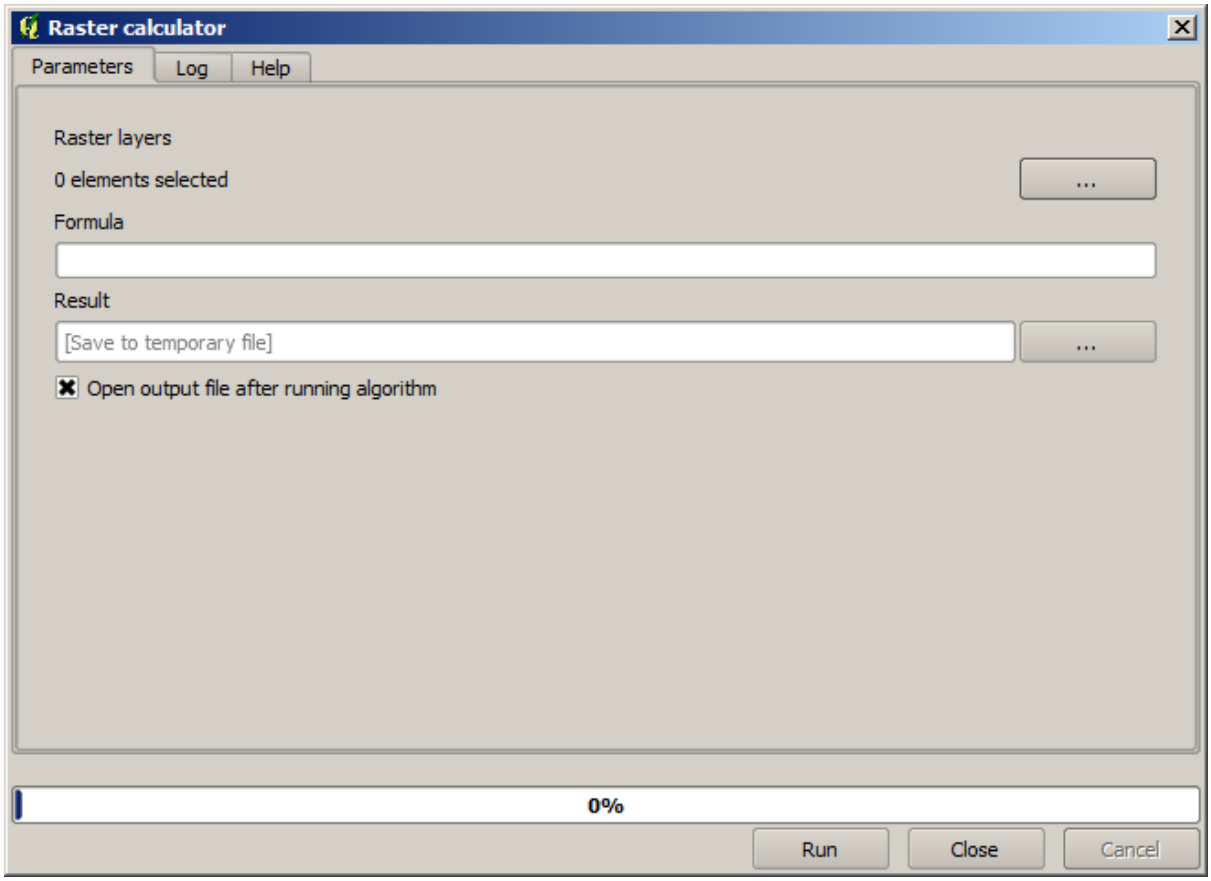
---

The raster calculator is one of the most powerful algorithms that you will find. It's a very flexible and versatile algorithm that can be used for many different calculations, and one that will soon become an important part of your toolbox.

In this lesson we will be performing some calculation with the raster calculator, most of them rather simple. This will let us see how it is used and how it deals with some particular situations that it might find. Understanding that is important to later get the expected results when using the calculator, and also to understand certain techniques that are commonly applied with it.

Open the QGIS project corresponding to this lesson and you will see that it contains several raster layers.

Now open the toolbox and open the dialog corresponding to the raster calculator.



Nota: The interface is different in recent versions.

The dialog contains 2 parameters.

- The layers to use for the analysis. This is a multiple input, that meaning that you can select as many layers as you want. Click on the button on the right–hand side and then select the layers that you want to use in the dialog that will appear.
- The formula to apply. The formula uses the layers selected in the above parameter, which are named using alphabet letters (`a, b, c...`) or `g1, g2, g3...` as variable names. That is, the formula `a + 2 * b` is the same as `g1 + 2 * g2` and will compute the sum of the value in the first layer plus two times the value in the second layer. The ordering of the layers is the same ordering that you see in the selection dialog.

Aviso: The calculator is case sensitive.

To start with, we will change the units of the DEM from meters to feet. The formula we need is the following one:

```
h' = h * 3.28084
```

Select the DEM in the layers field and type `a * 3.28084` in the formula field.

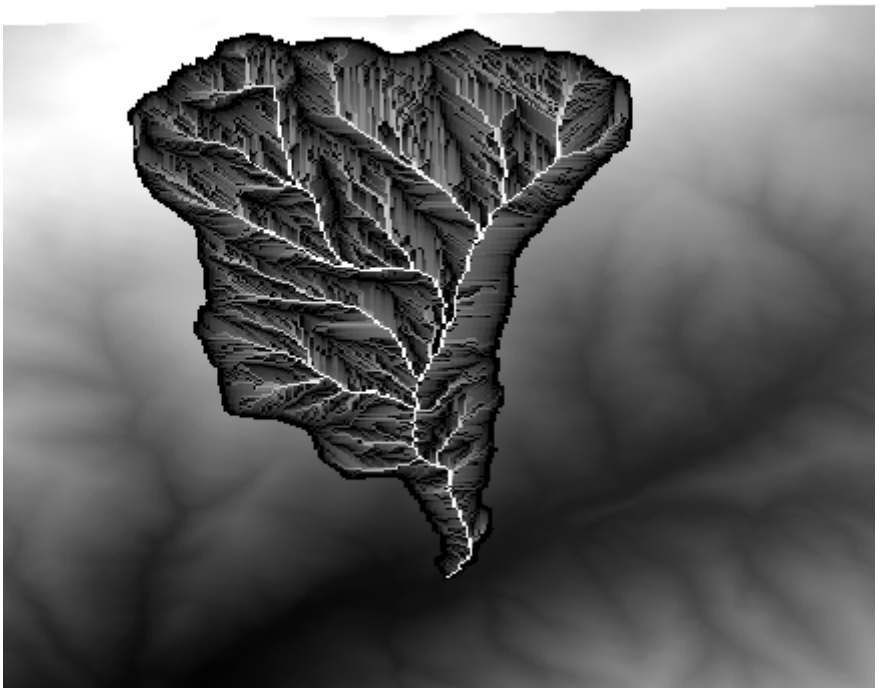
Aviso: For non English users: use always «.», not «,».

Click *Run* to run the algorithm. You will get a layer that has the same appearance of the input layer, but with different values. The input layer that we used has valid values in all its cells, so the last parameter has no effect at all.

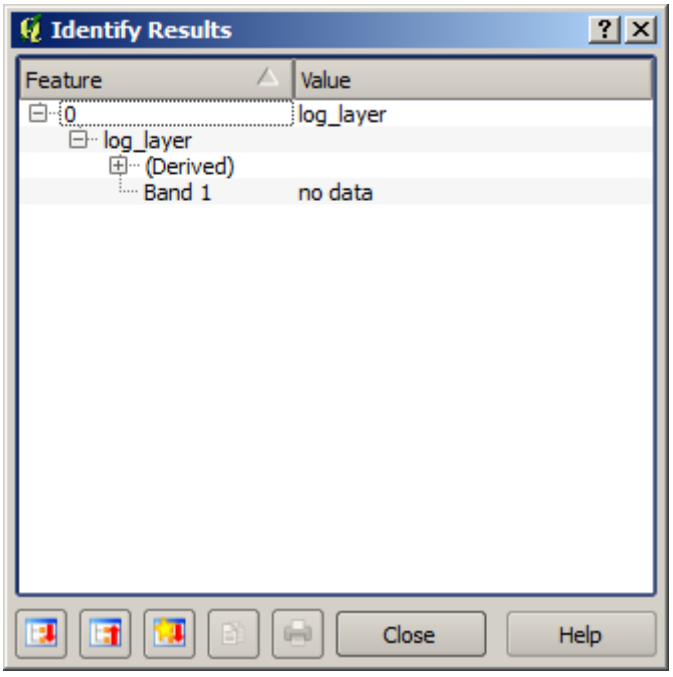
Let's now perform another calculation, this time on the *accflow* layer. This layer contains values of accumulated flow, a hydrological parameter. It contains those values only within the area of a given watershed, with no–data values outside of it. As you can see, the rendering is not very informative, due to the way values are distributed. Using the logarithm of that flow accumulation will yield a much more informative representation. We can calculate that using the raster calculator.

Open the algorithm dialog again, select the *accflow* layer as the only input layer, and enter the following formula: `log(a)`.

Here is the layer that you will get.



If you select the *Identify* tool to know the value of a layer at a given point, select the layer that we have just created, and click on a point outside of the basin, you will see that it contains a no–data value.



For the next exercise we are going to use two layers instead of one, and we are going to get a DEM with valid elevation values only within the basin defined in the second layer. Open the calculator dialog and select both layers of the project in the input layers field. Enter the following formula in the corresponding field:

```
a/a * b
```

`a` refers to the accumulated flow layer (since it is the first one to appear in the list) and `b` refers to the DEM. What we are doing in the first part of the formula here is to divide the accumulated flow layer by itself, which will result in a value of 1 inside the basin, and a no–data value outside. Then we multiply by the DEM, to get the elevation value in those cells inside the basin (`DEM * 1 = DEM`) and the no–data value outside (`DEM * no_data = no_data`)

Here is the resulting layer.

This technique is used frequently to *mask* values in a raster layer, and is useful whenever you want to perform calculations for a region other that the arbitrary rectangular region that is used by raster layer. For instance, an

elevation histogram of a raster layer doesn't have much meaning. If it is instead computed using only values corresponding to a basin (as in he case above), the result that we obtain is a meaningful one that actually gives information about the configuration of the basin.

There are other interesting things about this algorithm that we have just run, apart from the no–data values and how they are handled. If you have a look at the extents of the layers that we have multiplied (you can do it double–clicking on their names of the layer in the table of contents and looking at their properties), you will see that they are not the same, since the extent covered by the flow accumulation layer is smaller that the extent of the full DEM.

That means that those layers do not match, and that they cannot be multiplied directly without homogenizing those sizes and extents by resampling one or both layers. However, we did not do anything. QGIS takes care of this situation and automatically resamples input layers when needed. The output extent is the minimum covering extent calculated from the input layers, and the minimum cell size of their cellsizes.

In this case (and in most cases), this produces the desired results, but you should always be aware of the additional operations that are taking place, since they might affect the result. In cases when this behaviour might not be the desired, manual resampling should be applied in advance. In later chapters, we will see more about the behaviour of algorithms when using multiple raster layers.

Let's finish this lesson with another masking exercise. We are going to calculate the slope in all areas with an elevation between 1000 and 1500 meters.

In this case, we do not have a layer to use as a mask, but we can create it using the calculator.

Run the calculator using the DEM as only input layer and the following formula

```
ifelse(abs(a-1250) < 250, 1, 0/0)
```

As you can see, we can use the calculator not only to do simple algebraic operations, but also to run more complex calculation involving conditional sentences, like the one above.

The result has a value of 1 inside the range we want to work with, and no-data in cells outside of it.

The no-data value comes from the 0/0 expression. Since that is an undetermined value, SAGA will add a NaN (Not a Number) value, which is actually handled as a no-data value. With this little trick you can set a no-data value without needing to know what the no–data value of the cell is.

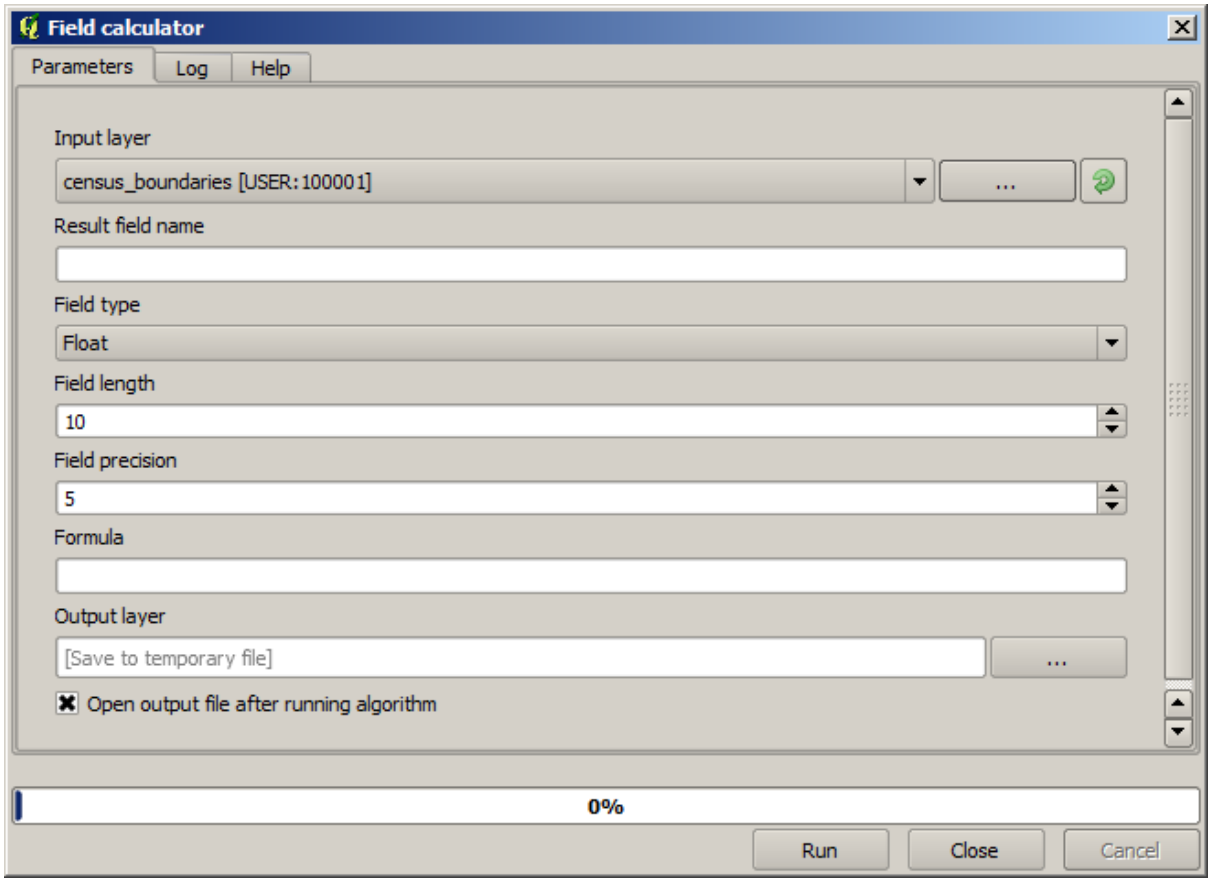
Now you just have to multiply it by the slope layer included in the project, and you will get the desired result.

All that can be done in a single operation with the calculator. We leave that as an exercise for the reader.

17.11 Calculadora vetorial

Nota: In this lesson we will see how to add new attributes to a vector layer based on a mathematical expression, using the vector calculator.

We already know how to use the raster calculator to create new raster layers using mathematical expressions. A similar algorithm is available for vector layers, and generates a new layer with the same attributes of the input layer, plus an additional one with the result of the expression entered. The algorithm is called *Field calculator* and has the following parameters dialog.



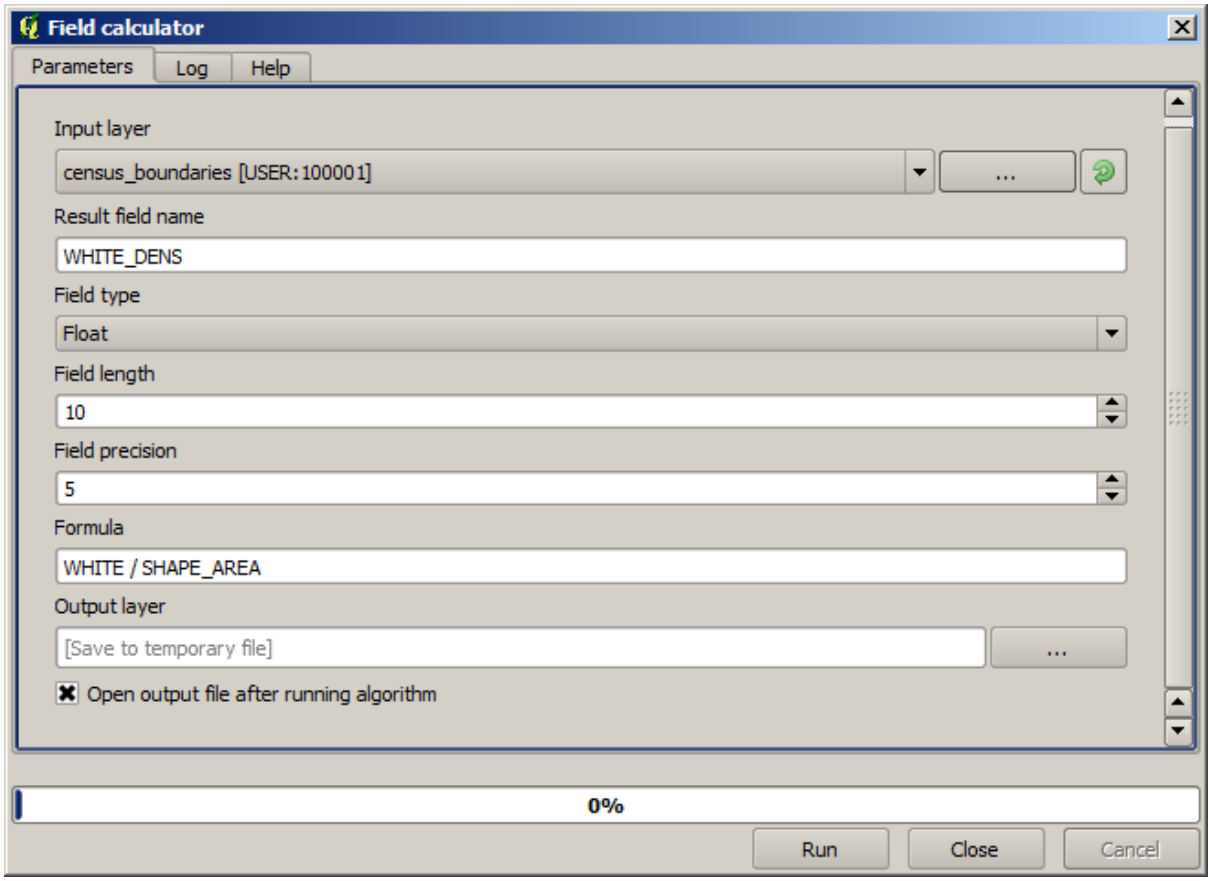
Nota: In newer versions of Processing the interface has changed considerably, it's more powerful and easier to use.

Here are a few examples of using that algorithm.

First, let's calculate the population density of white people in each polygon, which represents a census. We have two fields in the attributes table that we can use for that, namely `WHITE` and `SHAPE_AREA`. We just have to divide them and multiply by one million (to have density per square km), so we can use the following formula in the corresponding field

```
( "WHITE" / "SHAPE_AREA" ) * 1000000
```

The parameters dialog should be filled as shown below.



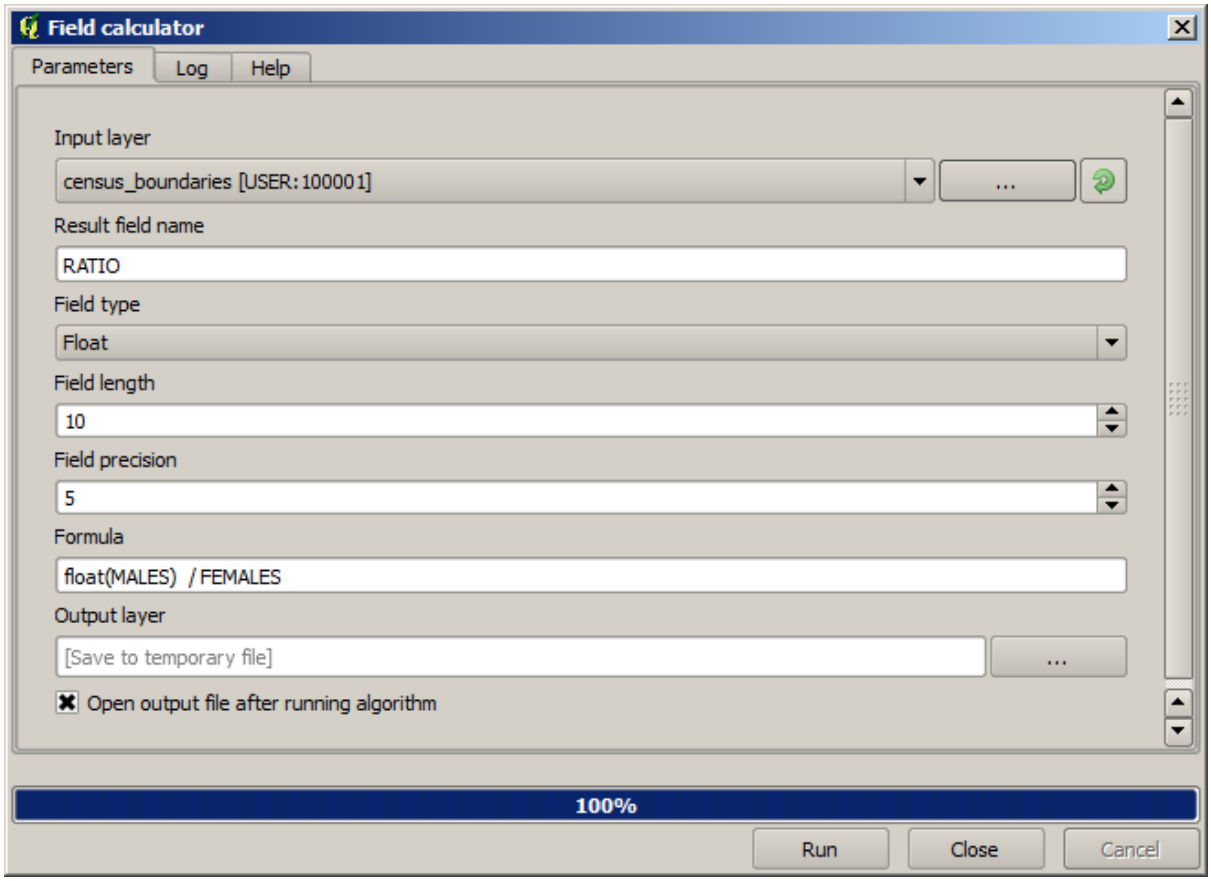
This will generate a new field named `WHITE_DENS`

Now let's calculate the ratio between the `MALES` and `FEMALES` fields to create a new one that indicates if male population is numerically predominant over female population.

Enter the following formula

```
"MALES" / "FEMALES"
```

This time the parameters window should look like this before pressing the *OK* button.

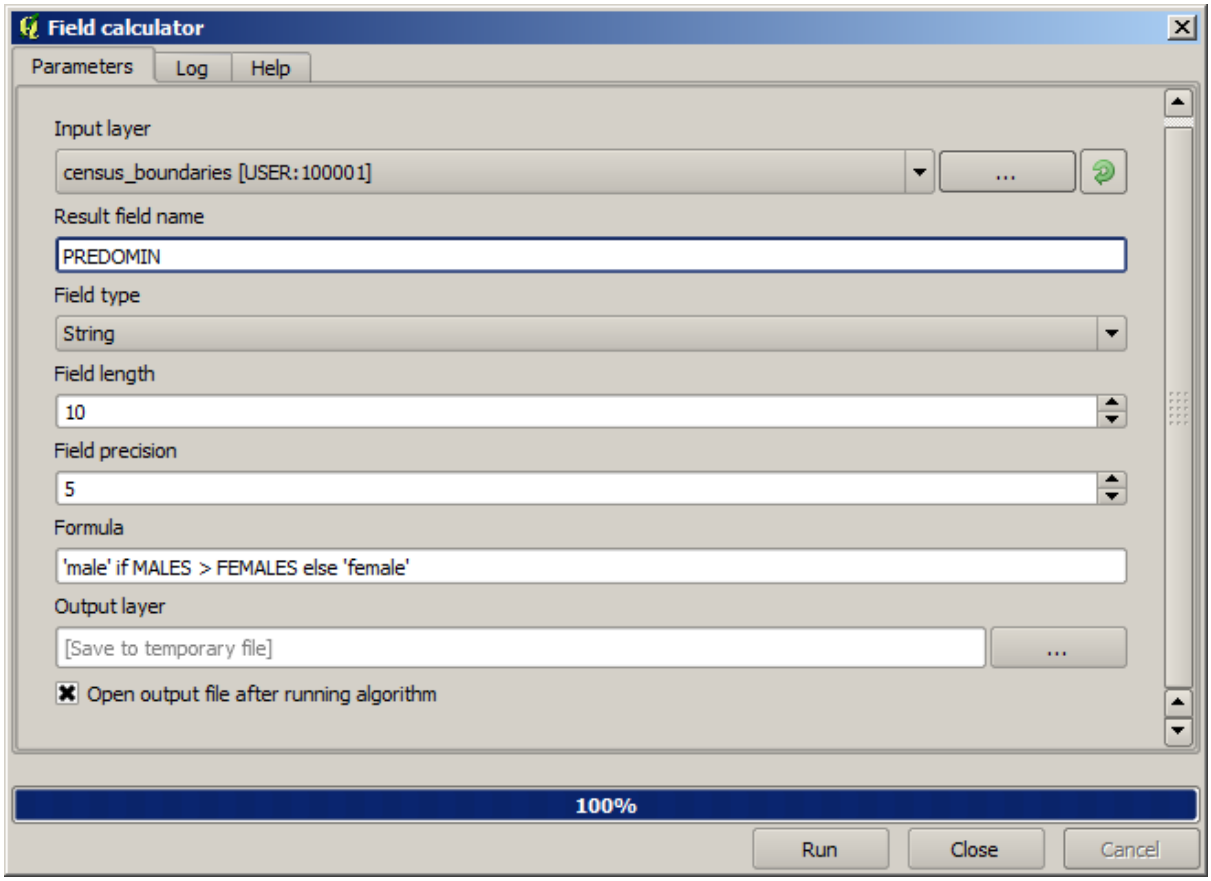


In earlier version, since both fields are of type integer, the result would be truncated to an integer. In this case the formula should be: `1.0 * "MALES" / "FEMALES"`, to indicate that we want floating point number a result.

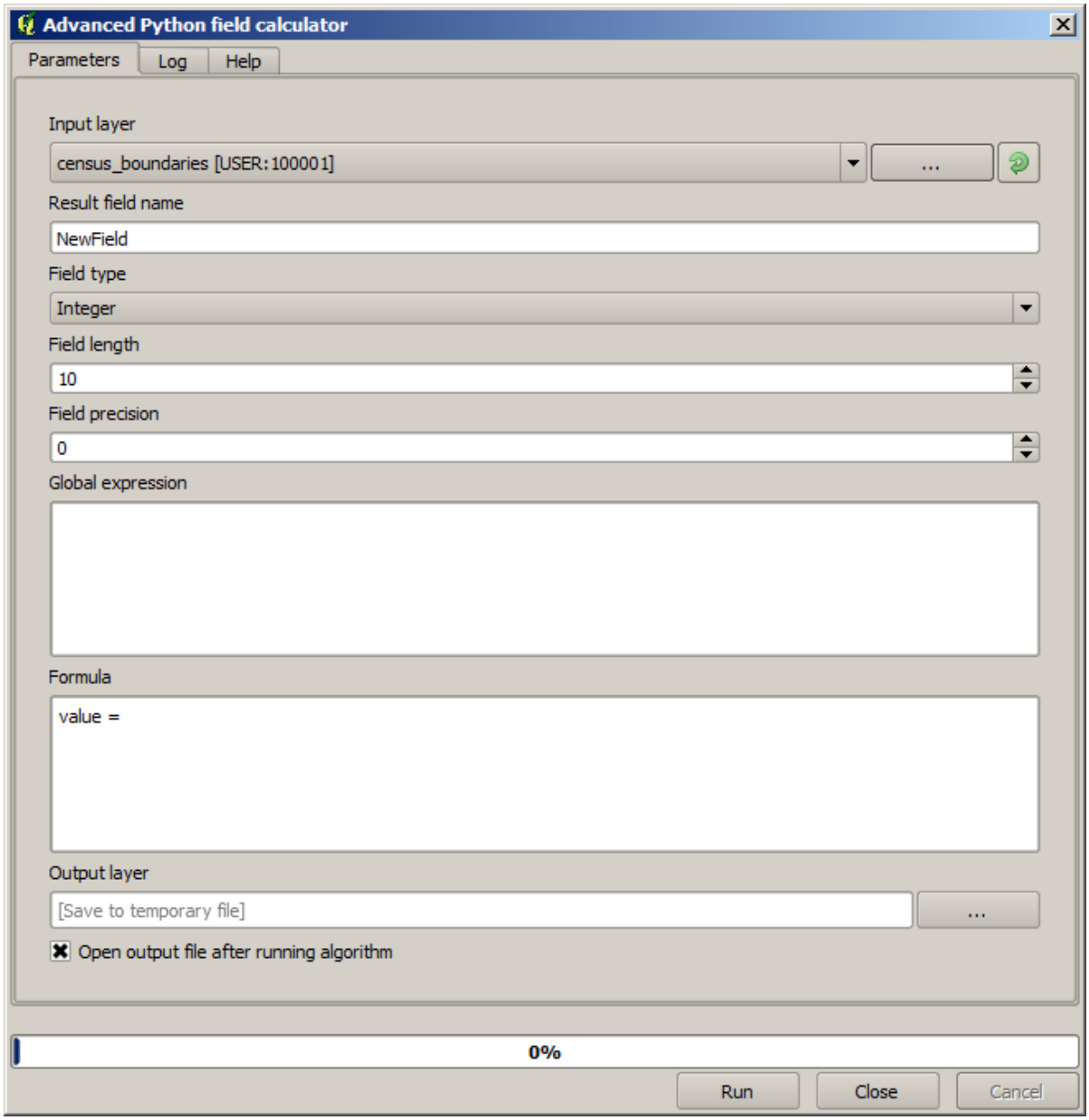
We can use conditional functions to have a new field with `male` or `female` text strings instead of those ratio value, using the following formula:

```
CASE WHEN  "MALES" > "FEMALES"  THEN 'male' ELSE 'female' END
```

The parameters window should look like this.



A python field calculator is available in the *Advanced Python field calculator*, which will not be detailed here



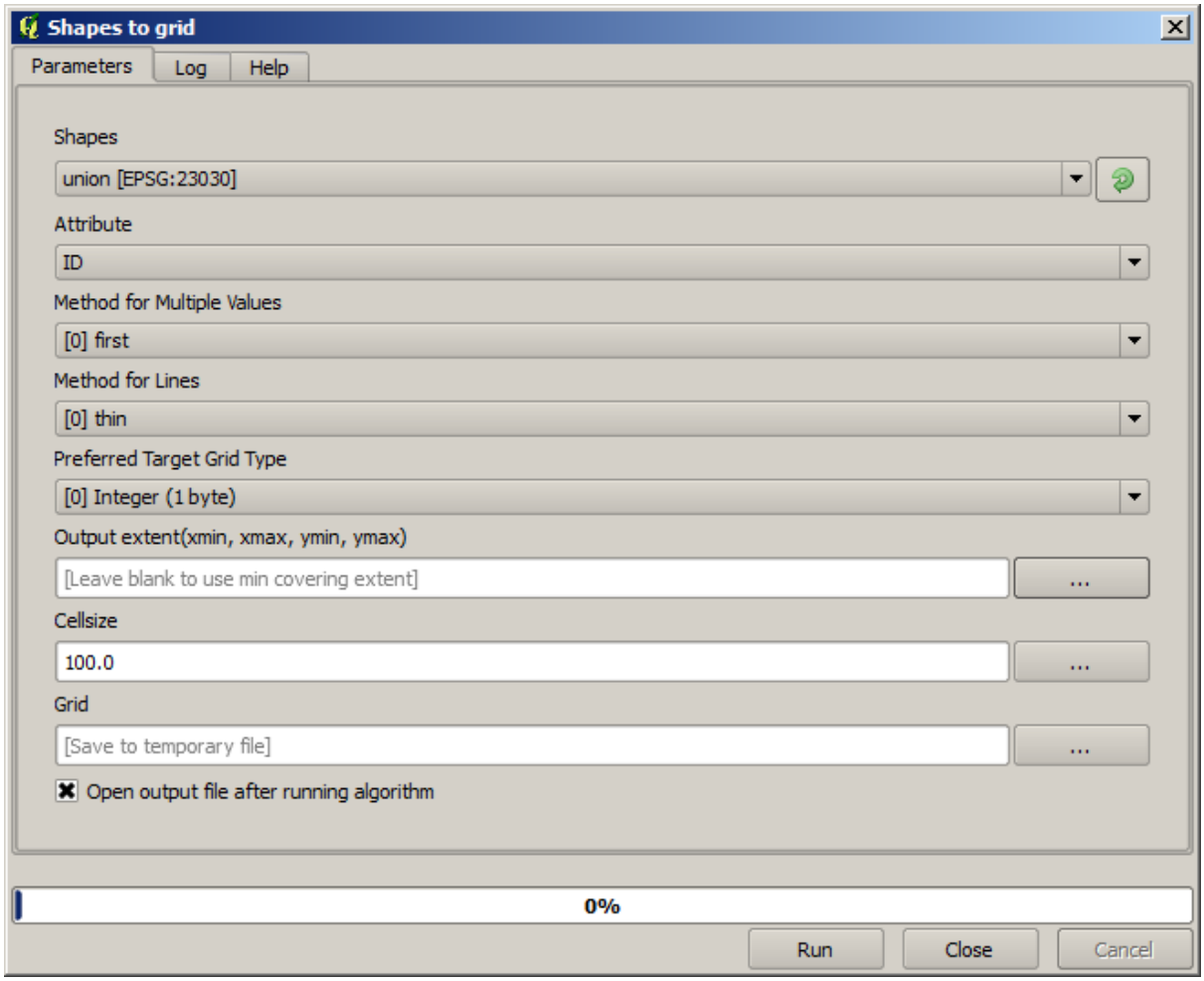
17.12 Defining extents

Nota: In this lesson we will see how to define extents, which are needed by some algorithms, especially raster ones.

Some algorithms require an extent to define the area to be covered by the analysis they perform, and usually to define the extent of the resulting layer.

When an extent is required, it can be defined manually by entering the four values that define it (min X, min Y, max X, max Y), but there are other more practical and more interesting ways of doing it as well. We will see all of them in this lesson.

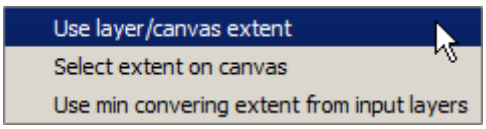
First, let's open an algorithm that requires an extent to be defined. Open the *Rasterize* algorithm, which creates a raster layer from a vector layer.



All the parameters, except for the last two ones, are used to define which layer is to be rasterized, and configure how the rasterization process should work. The two last parameters, on the other hand, define the characteristics of the output layer. That means that they define the area that is covered (which is not necessarily the same area covered by the input vector layer), and the resolution/cellsize (which cannot be inferred from the vector layer, since vector layers do not have a cellsize).

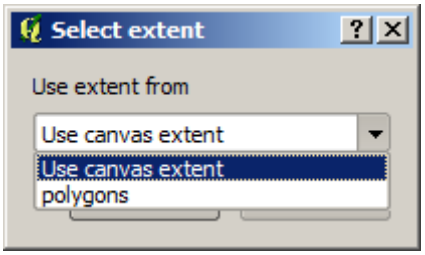
The first thing you can do is to type the 4 defining values explained before, separated by commas.

That doesn't need any extra explanation. While this is the most flexible option, it is also the less practical in some cases, and that's why other options are implemented. To access them, you have to click on the button on the right–hand side of the extent text box.



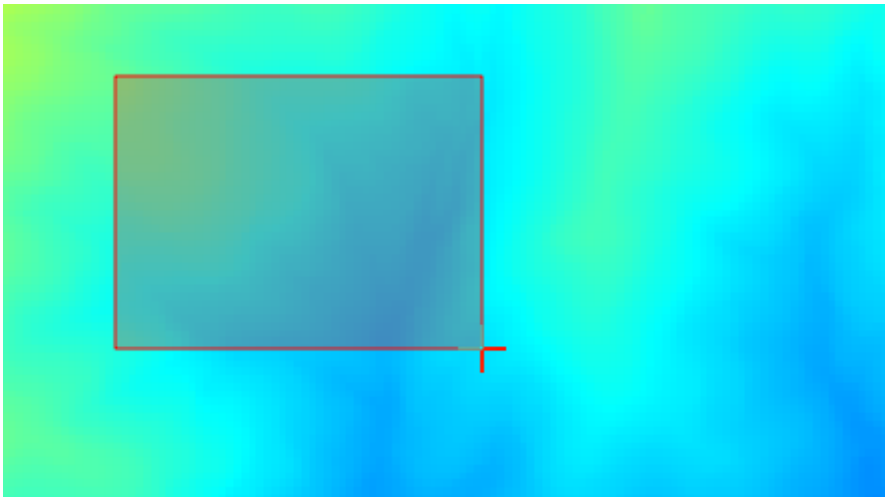
Let's see what each one of them does.

The first option is *Use layer/canvas extent*, which will show the selection dialog shown below.



Here you can select the extent of the canvas (the extent covered by the current zoom), or the extension any of the available layers. Select it and click on *OK*, and the text box will be automatically filled with the corresponding values.

The second option is *Select extent on canvas*. In this case, the algorithm dialog disappears and you can click and drag on the QGIS canvas to define the desired extent.

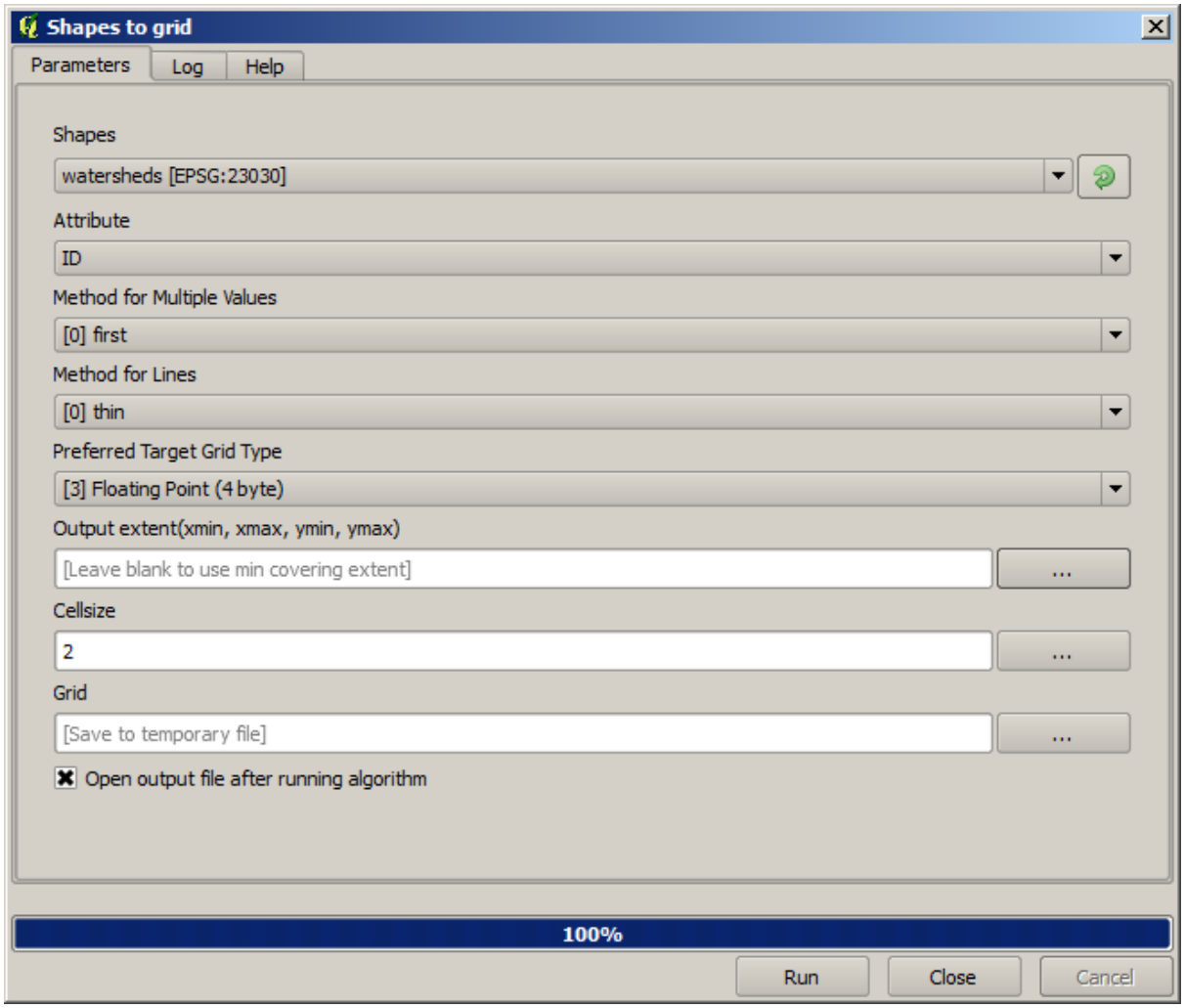


Once you release the mouse button, the dialog will reappear and the text box will already have the values corresponding to the defined extent.

The last option is *Use min covering extent from input layers*, which is the default option. This will compute the min covering extent of all layers used to run the algorithm, and there is no need to enter any value in the text box. In the case of a single input layer, as in the algorithm we are running, the same extent can be obtained by selecting that same input layer in the *Use layer/canvas extent* that we already saw. However, when there are several input layers, the min covering extent does not correspond to any of the input layer extent, since it is computed from all of them together.

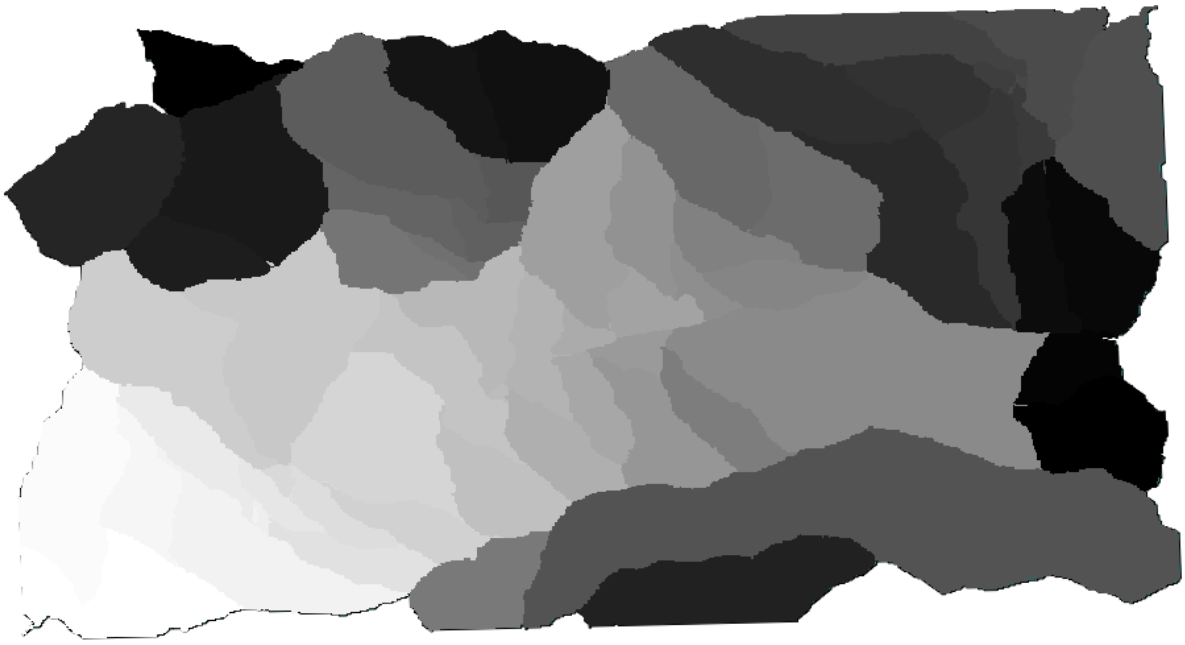
We will use this last method to execute our rasterization algorithm.

Fill the parameters dialog as shown next, and press *OK*.



Nota: In this case, better use an *Integer (1 byte)* instead of a *Floating point (4 byte)*, since the *NAME* is an integer with maximum value=64. This will result in a smaller file size and faster computations.

You will get a rasterized layer that covers exactly the area covered by the original vector layer.



In some cases, the last option, *Use min covering extent from input layers*, might not be available. This will happen in those algorithm that do not have input layers, but just parameters of other types. In that case, you will have to enter the value manually or use any of the other options.

Notice that, when a selection exist, the extent of the layer is that of the whole set of features, and the selection is not used to compute the extent, even though the rasterization is executed on the selected items only. In that case, you might want to actually create a new layer from the selection, and then use it as input.

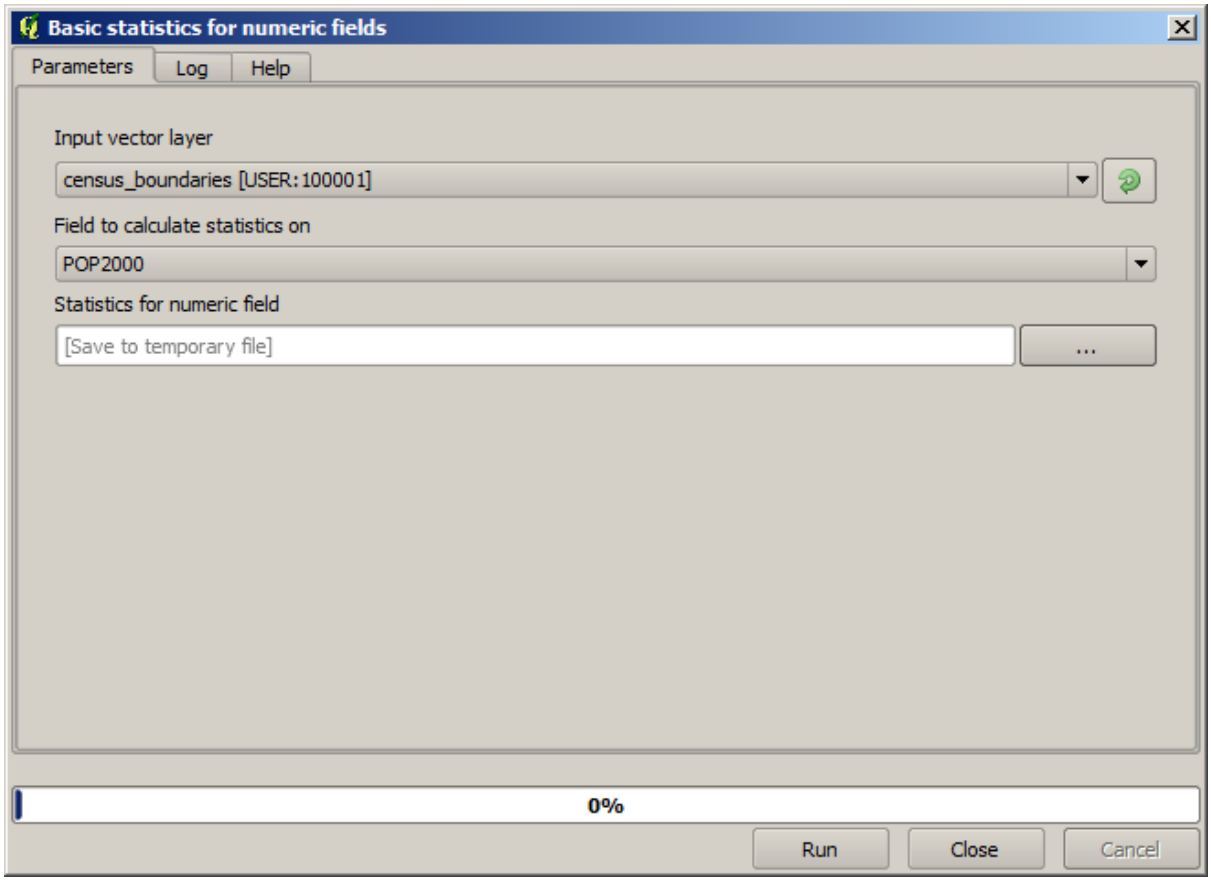
17.13 HTML outputs

Nota: In this lesson we learn how QGIS handles outputs in HTML format, which are used to produce text outputs and graphs.

All the outputs we have produced so far were layers (whether raster or vector). However, some algorithms generate outputs in the form of text and graphics. All this outputs are wrapped in HTML files and displayed in the so–called *Results viewer*, which is another element of the processing framework.

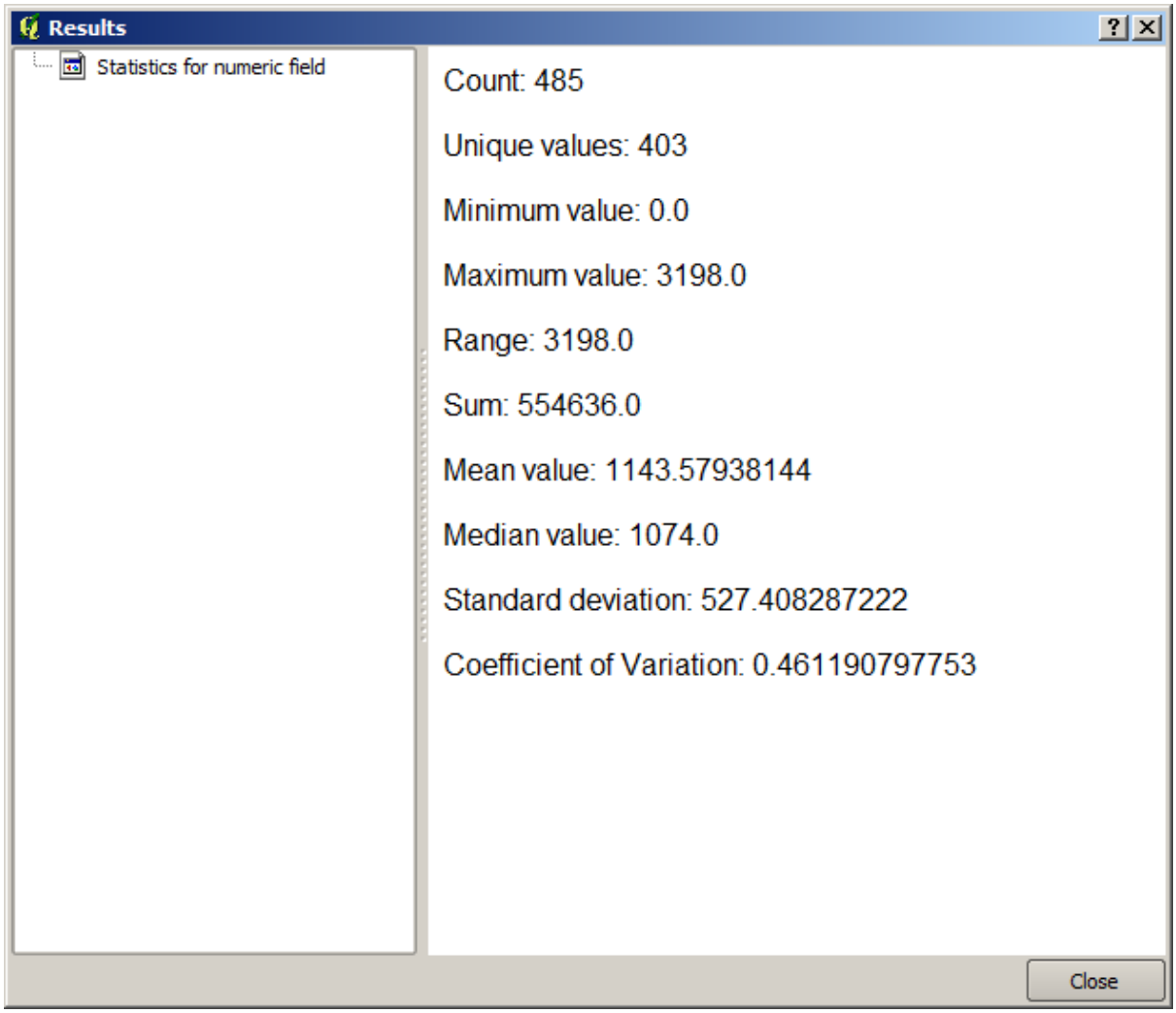
Let's see one of those algorithms to understand how they work.

Open the project with the data to be used in this lesson and then open the *Basic statistics for numeric fields* algorithm.



The algorithm is rather simple, and you just have to select the layer to use and one of its field (a numeric one). The output is of type HTML, but the corresponding box works exactly like the one that you can find in the case of a raster or vector output. You can enter a filepath or leave it blank to save to a temporary file. In this case, however, only the `html` and `htm` extensions are allowed, so there is no way of altering the output format by using a different one.

Run the algorithm selecting the only layer in the project as input, and the *POP2000* field, and a new dialog like the one shown next will appear once the algorithm is executed and the parameters dialog is closed.



This is the *Results viewer*. It keeps all the HTML result generated during the current session, easily accessible, so you can check them quickly whenever you need it. As it happens with layers, if you have saved the output to a temporary file, it will be deleted once you close QGIS. If you have saved to a non-temporary path, the file will remain, but it will not appear in the *Results viewer* the next time you open QGIS.

Some algorithms generate text that cannot be divided into other more detailed outputs. That is the case if, for instance, the algorithm captures the text output from an external process. In other cases, the output is presented as text, but internally is divided into several smaller outputs, usually in the form of numeric values. The algorithm that we have just executed is one of them. Each one of those values is handled as a single output, and stored in a variable. This has no importance at all now, but once we move to the model designer, you will see that it will allow us to use those values as numeric inputs for other algorithms.

17.14 First analysis example

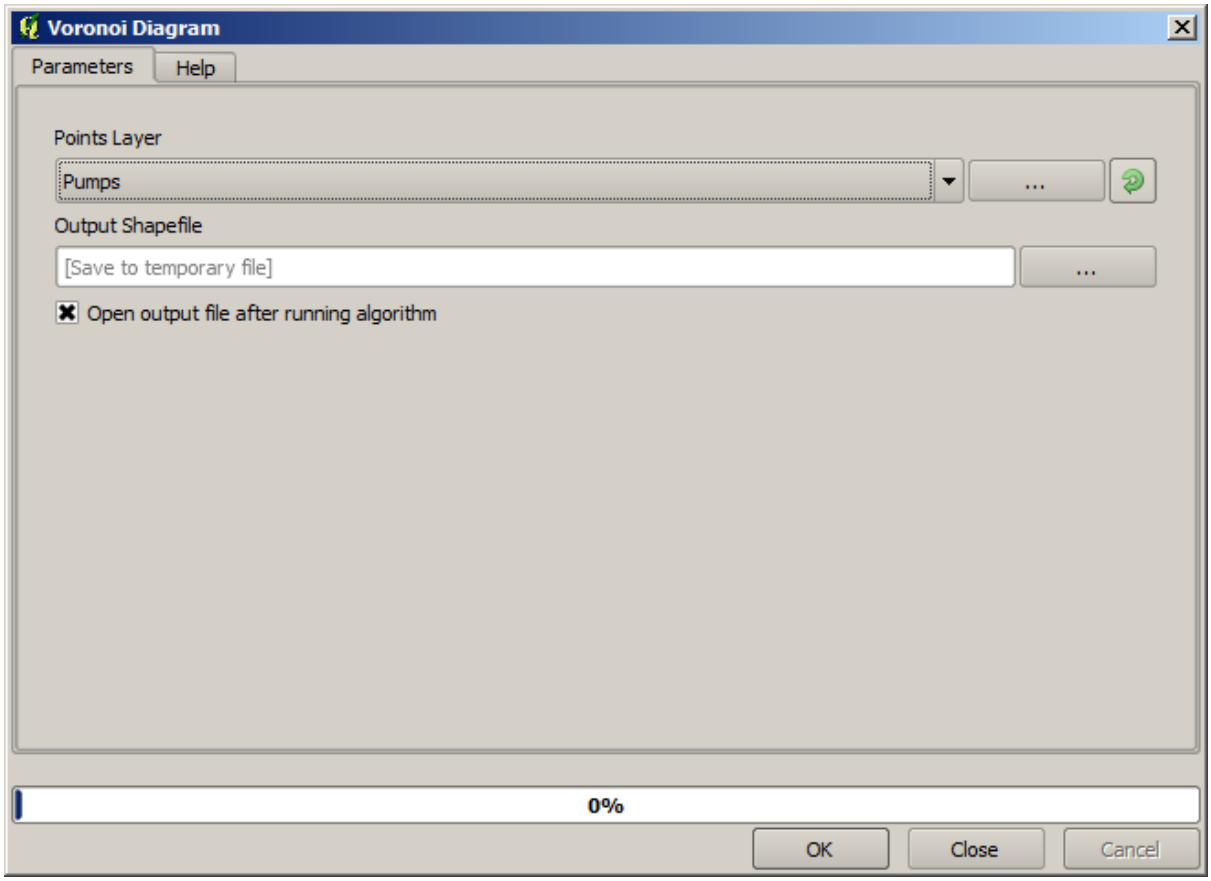
Nota: In this lesson we will perform some real analysis using just the toolbox, so you can get more familiar with the processing framework elements.

Now that everything is configured and we can use external algorithms, we have a very powerful tool to perform spatial analysis. It is time to work out a larger exercise with some real–world data.

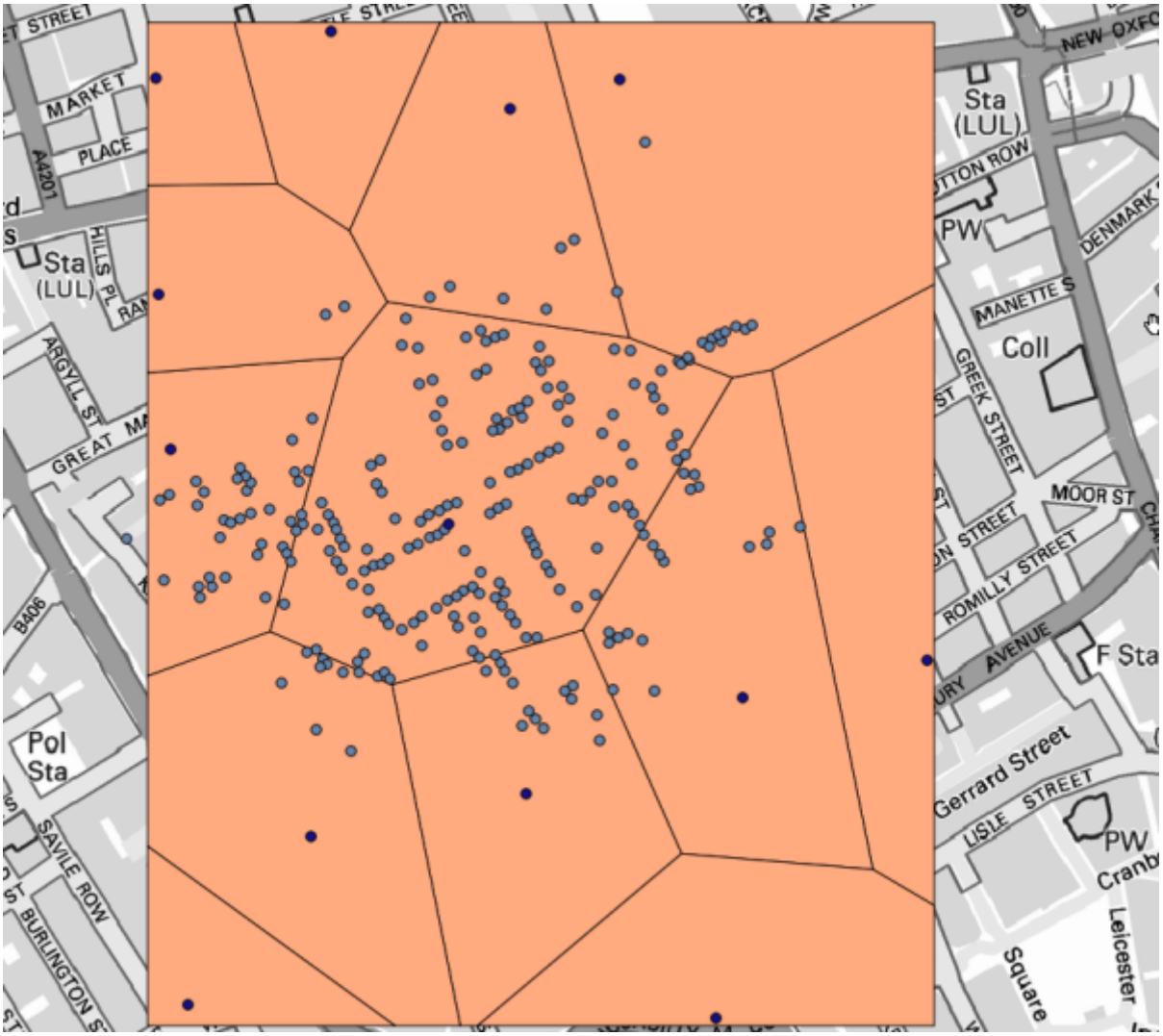
We will be using the well-known dataset that John Snow used in 1854, in his groundbreaking work (https://en.wikipedia.org/wiki/John_Snow_%28physician%29), and we will get some interesting results. The analysis of this dataset is pretty obvious and there is no need for sophisticated GIS techniques to end up with good results and conclusions, but it is a good way of showing how these spatial problems can be analyzed and solved by using different processing tools.

The dataset contains shapefiles with cholera deaths and pump locations, and an OSM rendered map in TIFF format. Open the corresponding QGIS project for this lesson.

The first thing to do is to calculate the Voronoi diagram (a.k.a. Thiessen polygons) of the pumps layer, to get the influence zone of each pump. The *Voronoi Diagram* algorithm can be used for that.

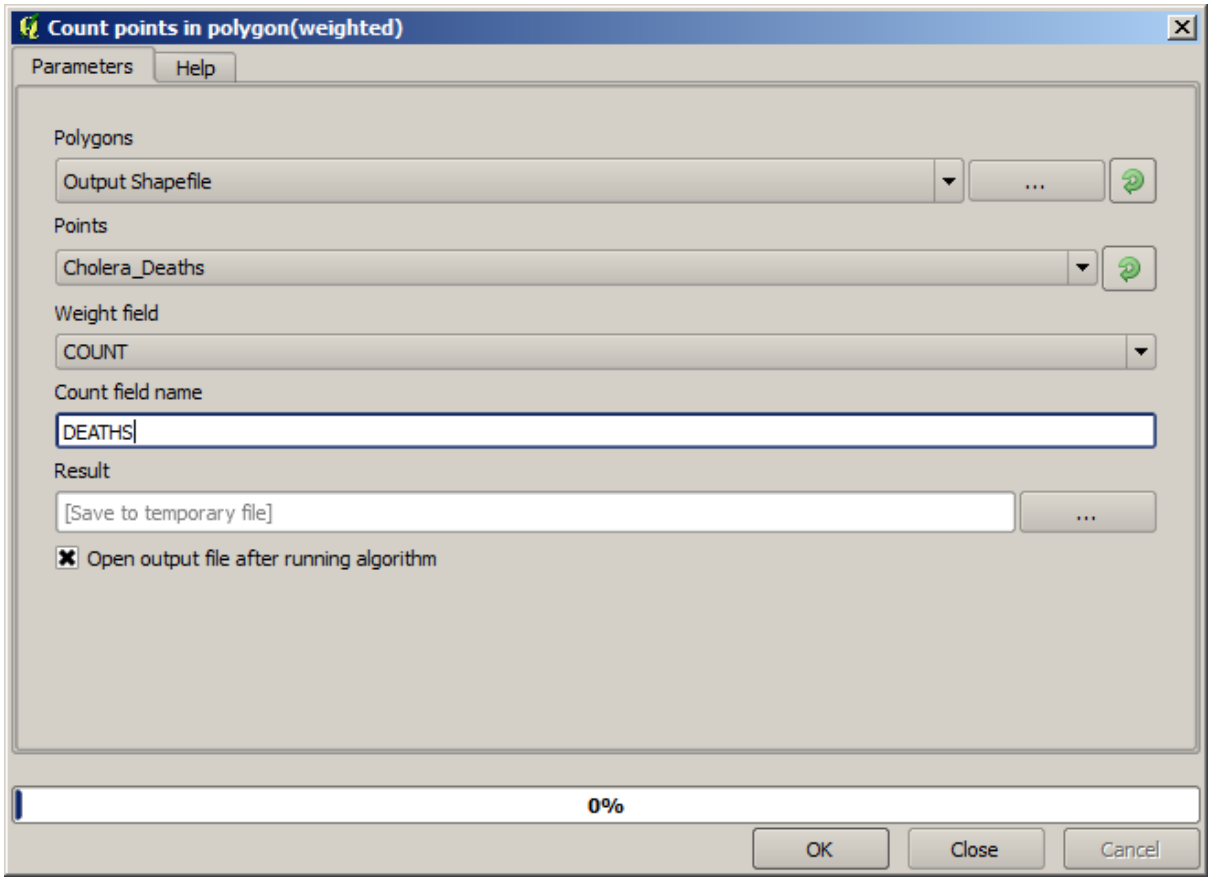


Pretty easy, but it will already give us interesting information.

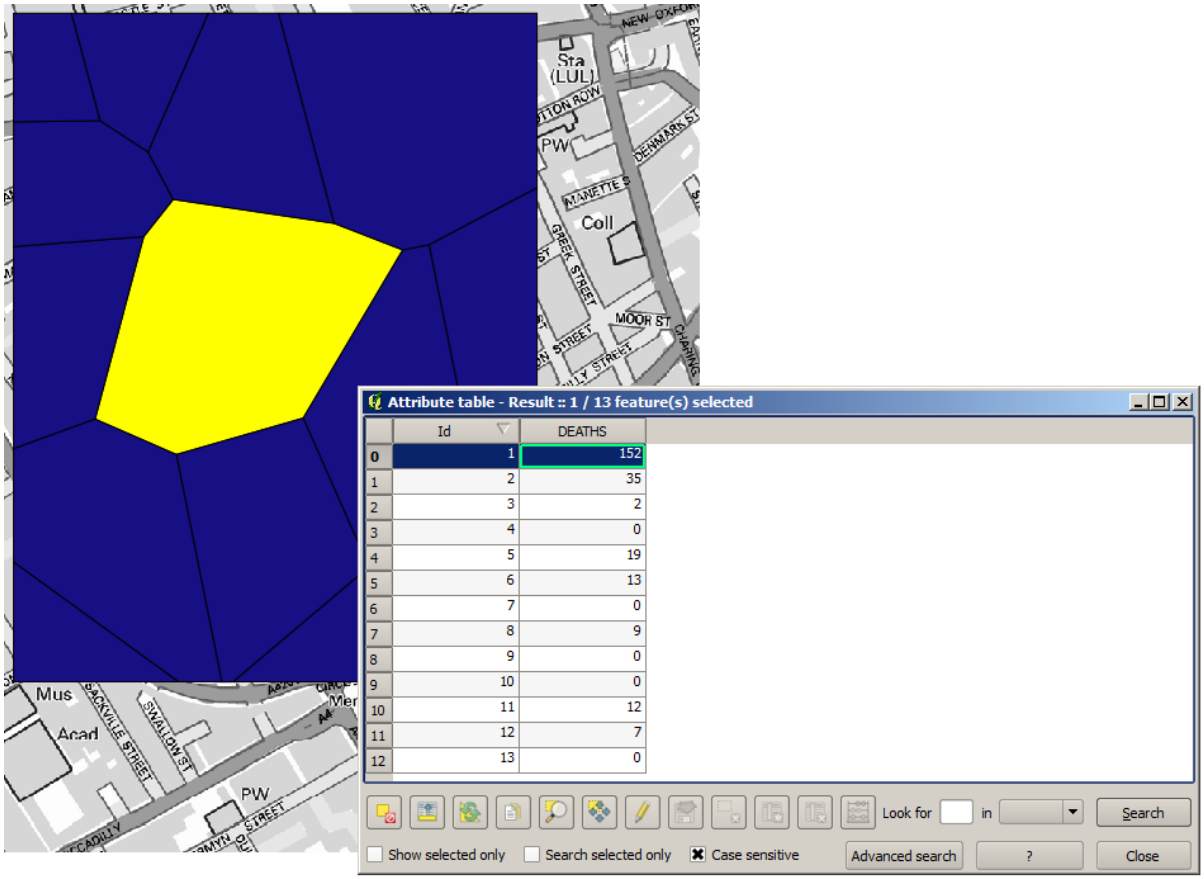


Clearly, most cases are within one of the polygons

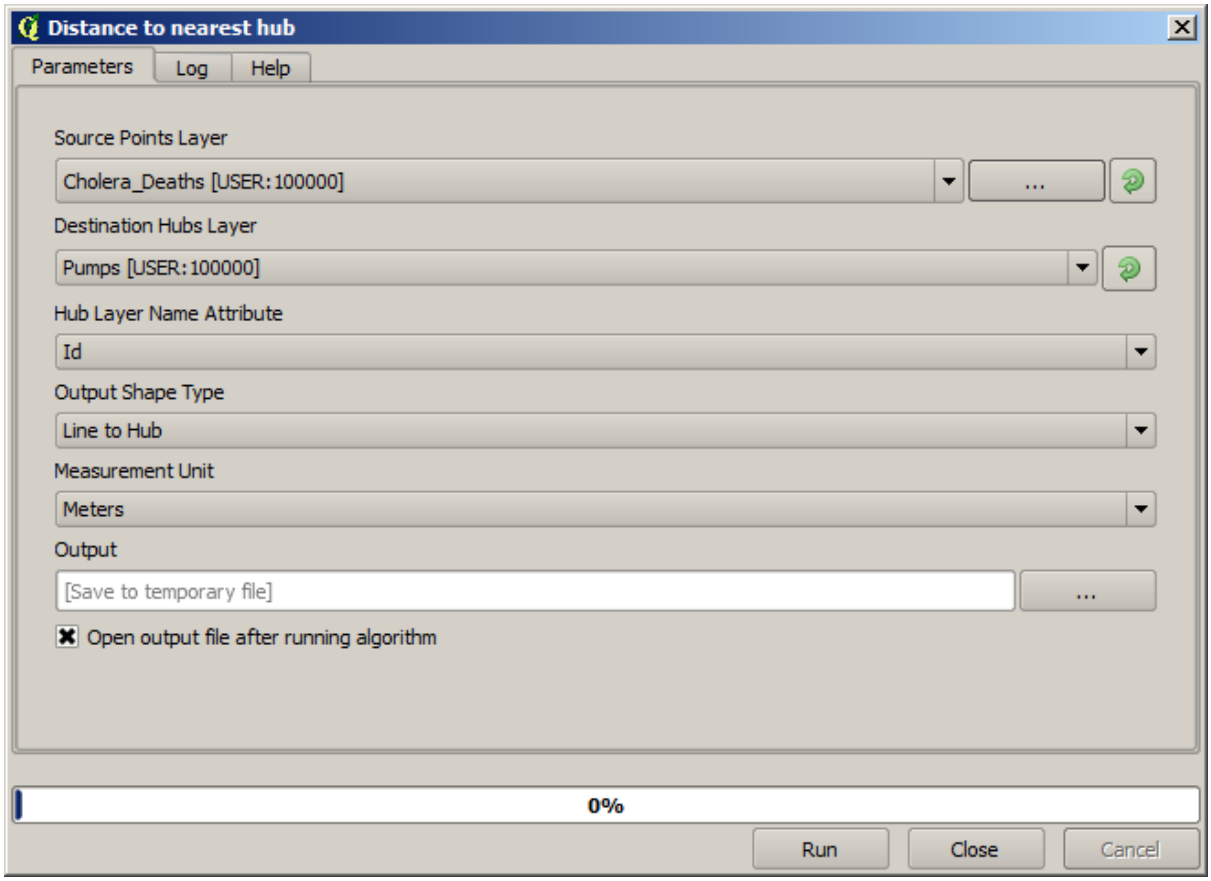
To get a more quantitative result, we can count the number of deaths in each polygon. Since each point represents a building where deaths occured, and the number of deaths is stored in an attribute, we cannot just count the points. We need a weighted count, so we will use the *Count points in polygon (weighted)* tool.



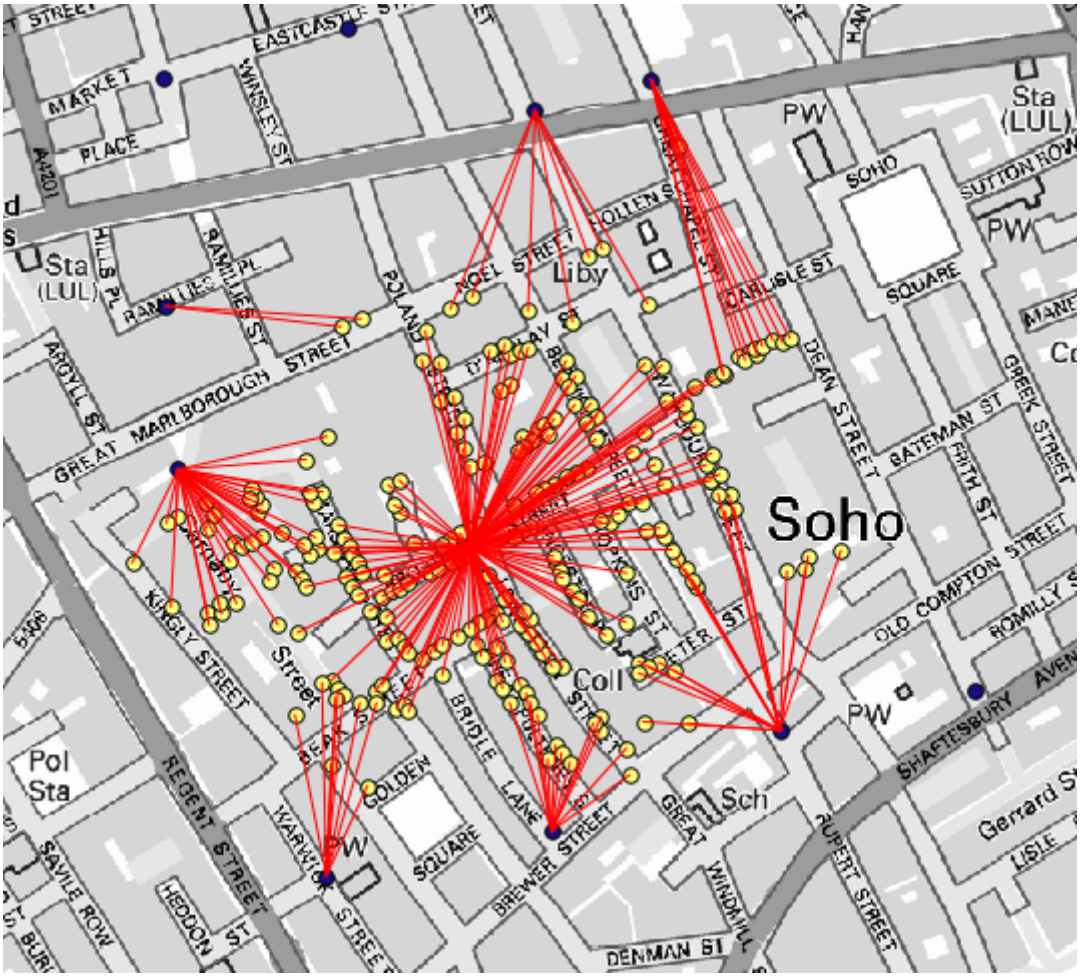
The new field will be called *DEATHS*, and we use the *COUNT* field as weighting field. The resulting table clearly reflects that the number of deaths in the polygon corresponding to the first pump is much larger than the other ones.



Another good way of visualizing the dependence of each point in the `Cholera_deaths` layer with a point in the `Pumps` layer is to draw a line to the closest one. This can be done with the *Distance to nearest hub* tool, and using the configuration shown next.

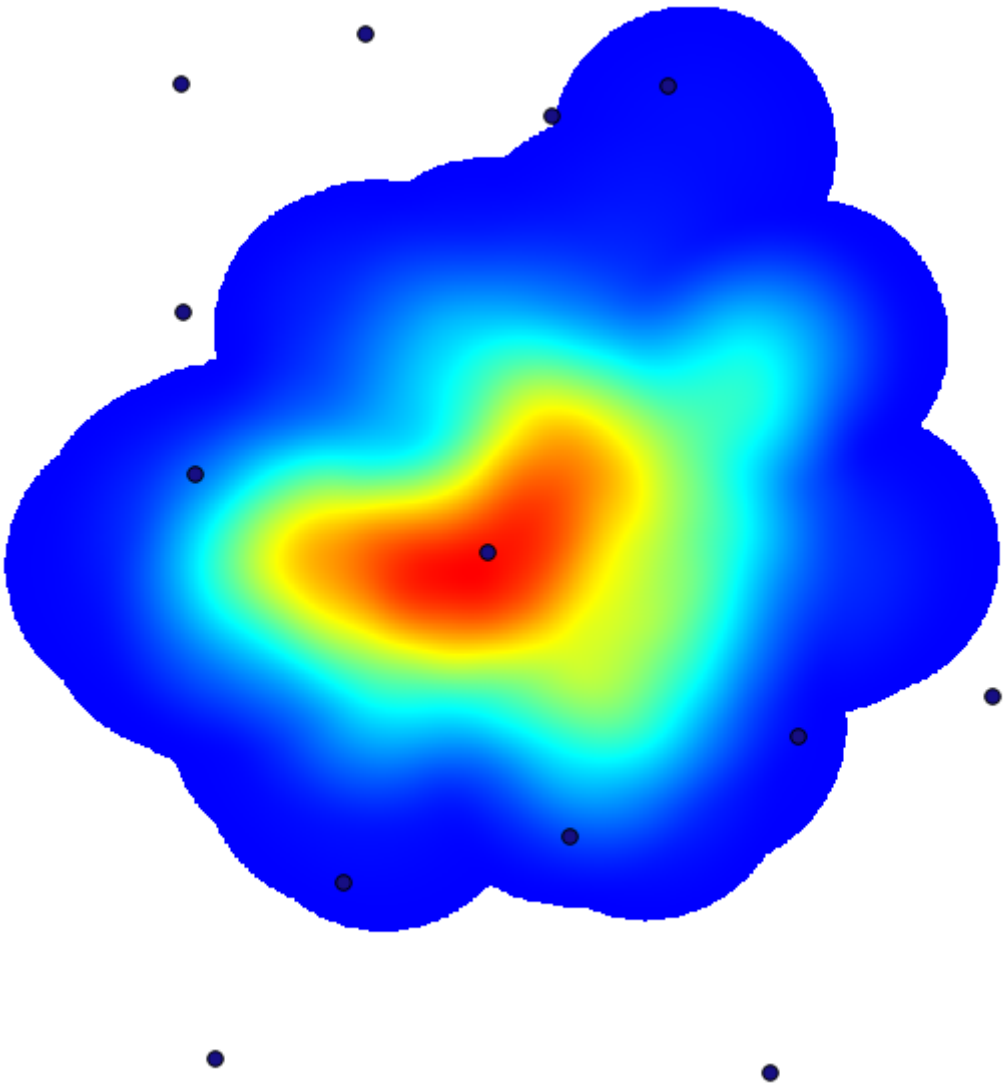


O resultado é parecido com isto:

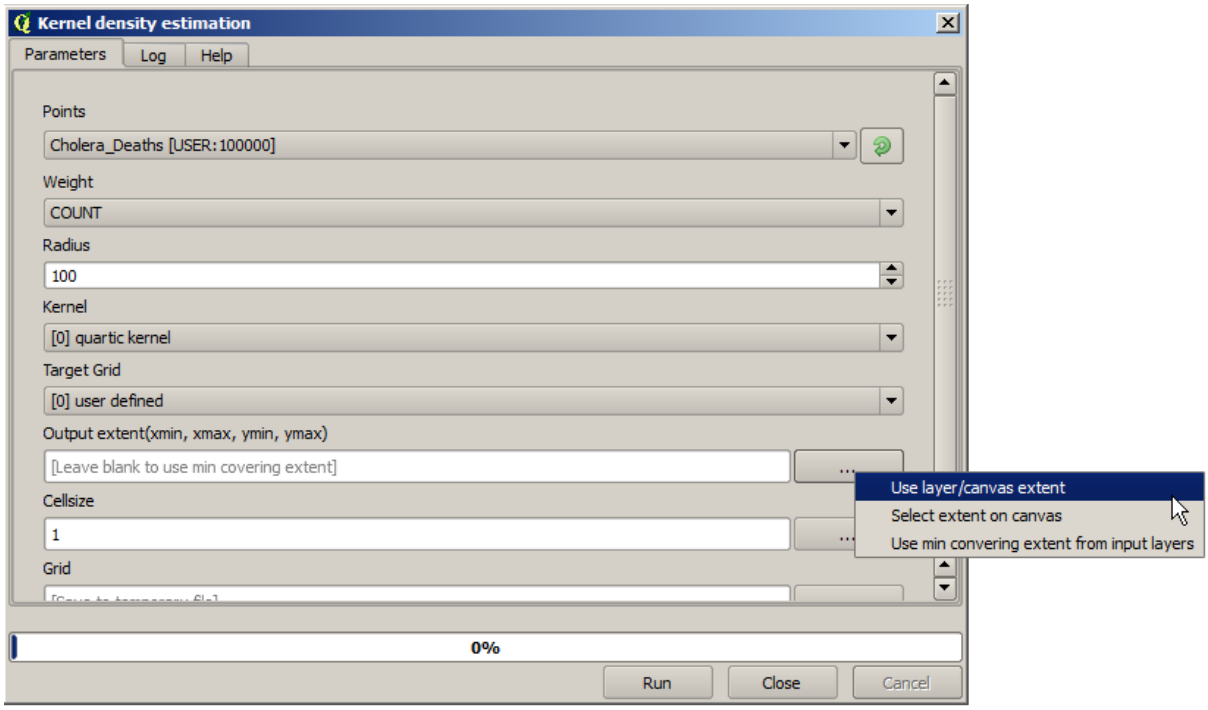


Although the number of lines is larger in the case of the central pump, do not forget that this does not represent the number of deaths, but the number of locations where cholera cases were found. It is a representative parameter, but it is not considering that some locations might have more cases than other.

A density layer will also give us a very clear view of what is happening. We can create it with the *Kernel density* algorithm. Using the *Cholera_deaths* layer, its *COUNT* field as weight field, with a radius of 100, the extent and cellsize of the streets raster layer, we get something like this.



Remember that, to get the output extent, you do not have to type it. Click on the button on the right-hand side and select *Use layer/canvas extent*.



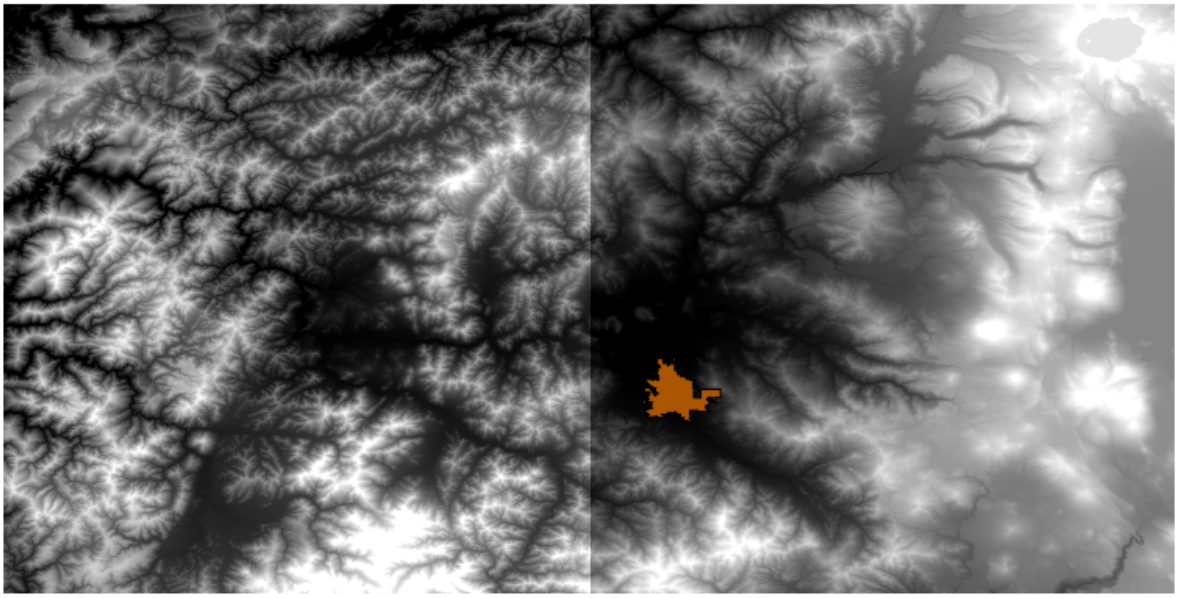
Select the streets raster layer and its extent will be automatically added to the text field. You must do the same with the cellsize, selecting the cellsize of that layer as well.

Combining with the pumps layer, we see that there is one pump clearly in the hotspot where the maximum density of death cases is found.

17.15 Clipping and merging raster layers

Nota: In this lesson we will see another example of spatial data preparation, to continue using geoalgorithms in real-world scenarios.

For this lesson, we are going to calculate a slope layer for an area surrounding a city area, which is given in a vector layer with a single polygon. The base DEM is divided in two raster layers that, together, cover an area much larger than that around the city that we want to work with. If you open the project corresponding to this lesson, you will see something like this.



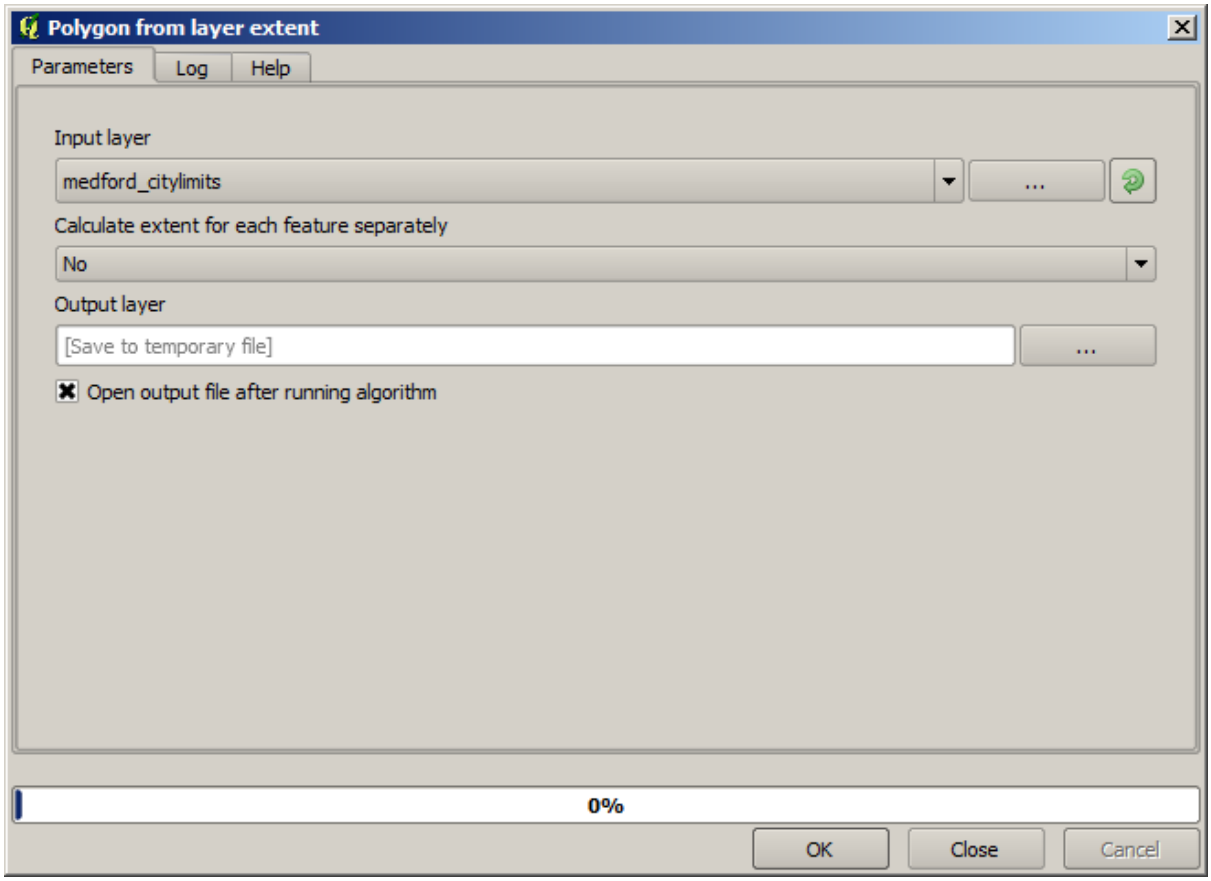
These layers have two problems:

- They cover an area that is too large for what we want (we are interested in a smaller region around the city center)
- They are in two different files (the city limits fall into just one single raster layer, but, as it's been said, we want some extra area around it).

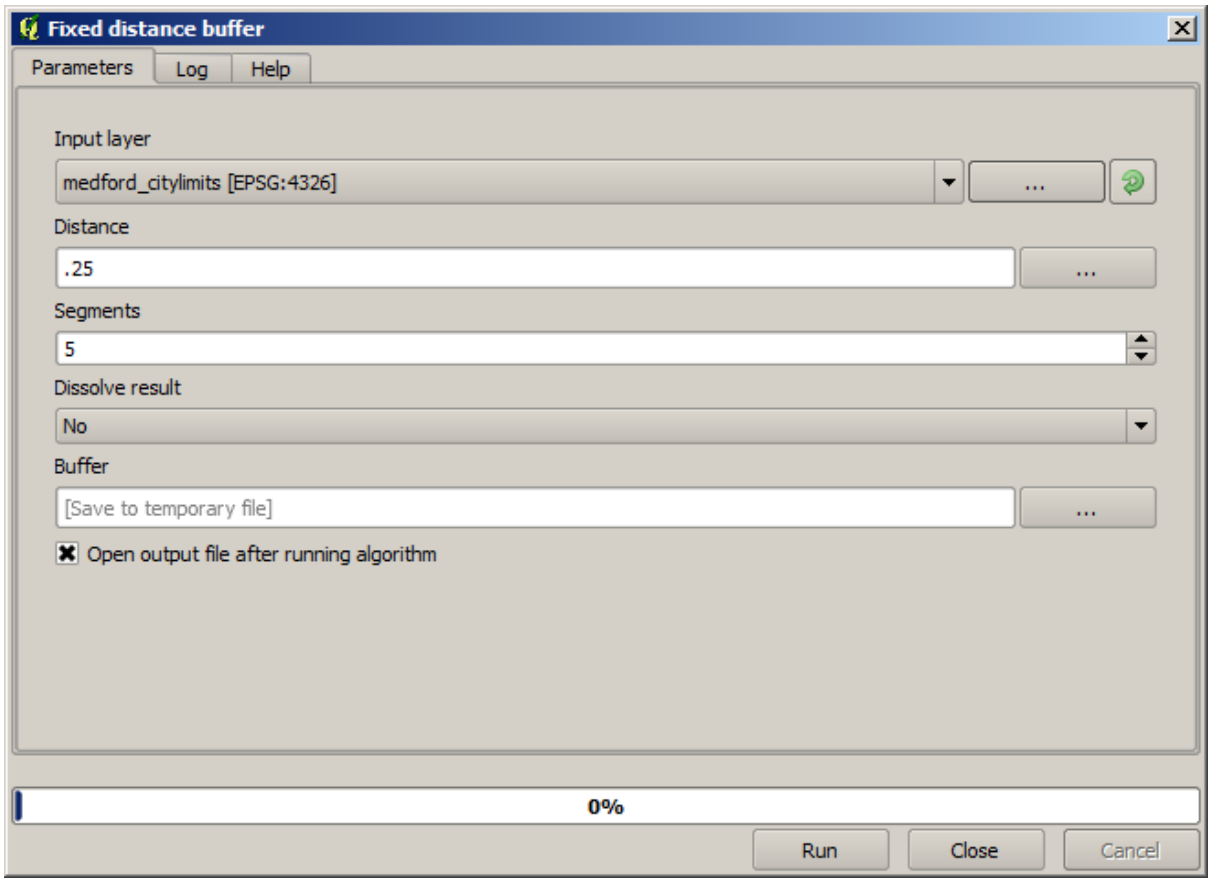
Both of them are easily solvable with the appropriate geoalgorithms.

First, we create a rectangle defining the area that we want. To do it, we create a layer containing the bounding box of the layer with the limits of the city area, and then we buffer it, so as to have a raster layer that covers a bit more that the strictly necessary.

To calculate the bounding box , we can use the *Polygon from layer extent* algorithm

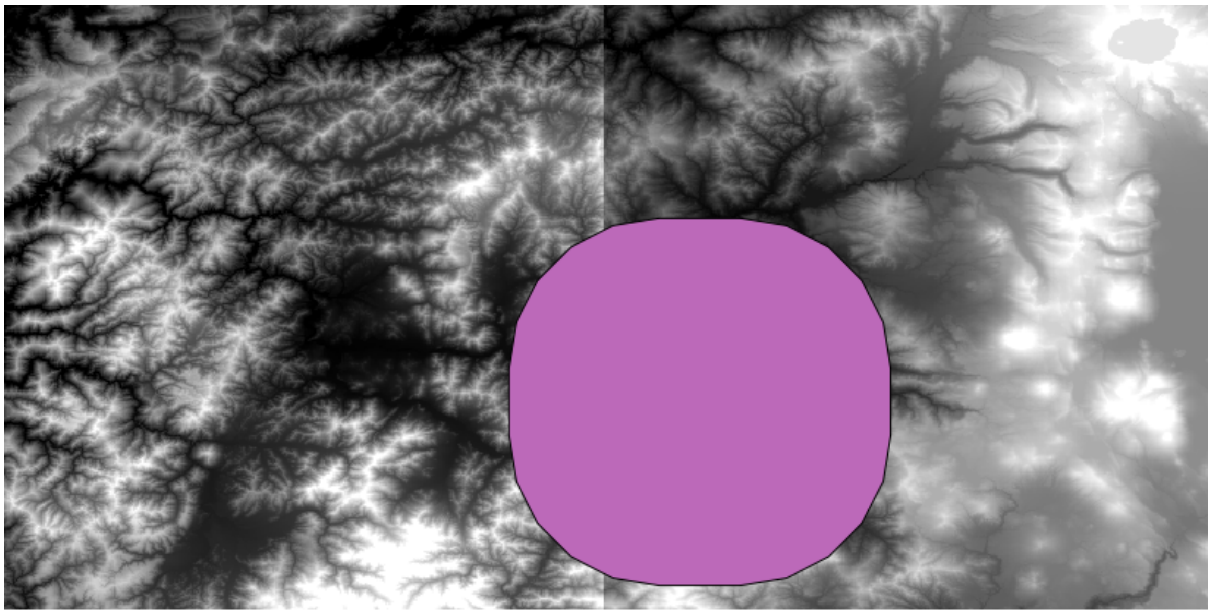


To buffer it, we use the *Fixed distance buffer* algorithm, with the following parameter values.

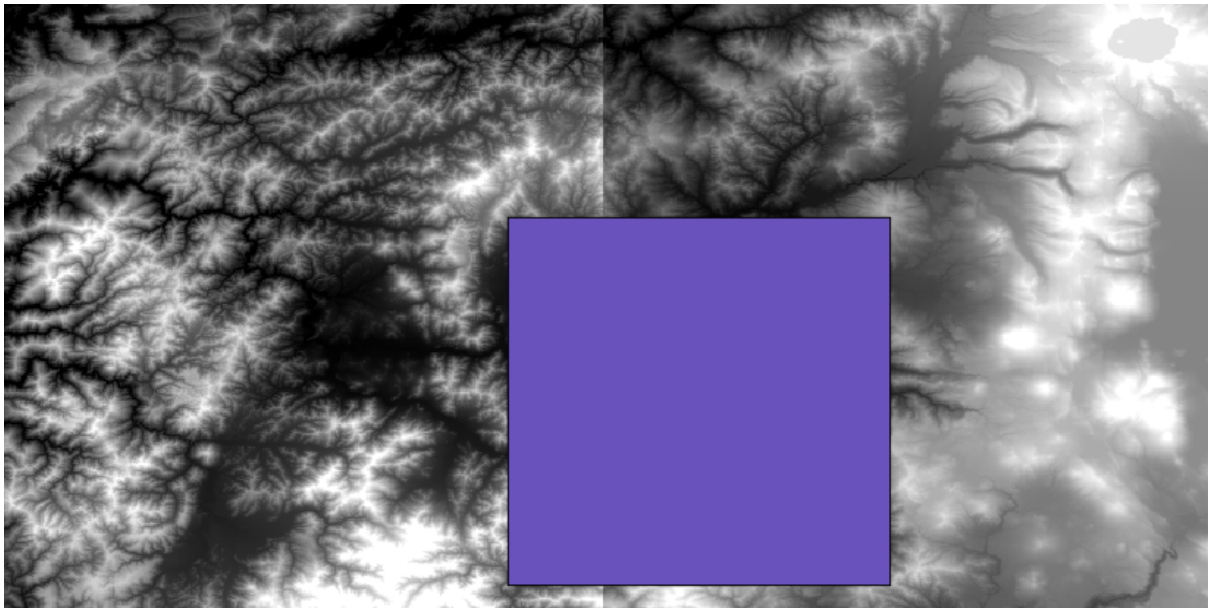


Aviso: Syntax changed in recent versions; set both Distance and Arc vertex to .25

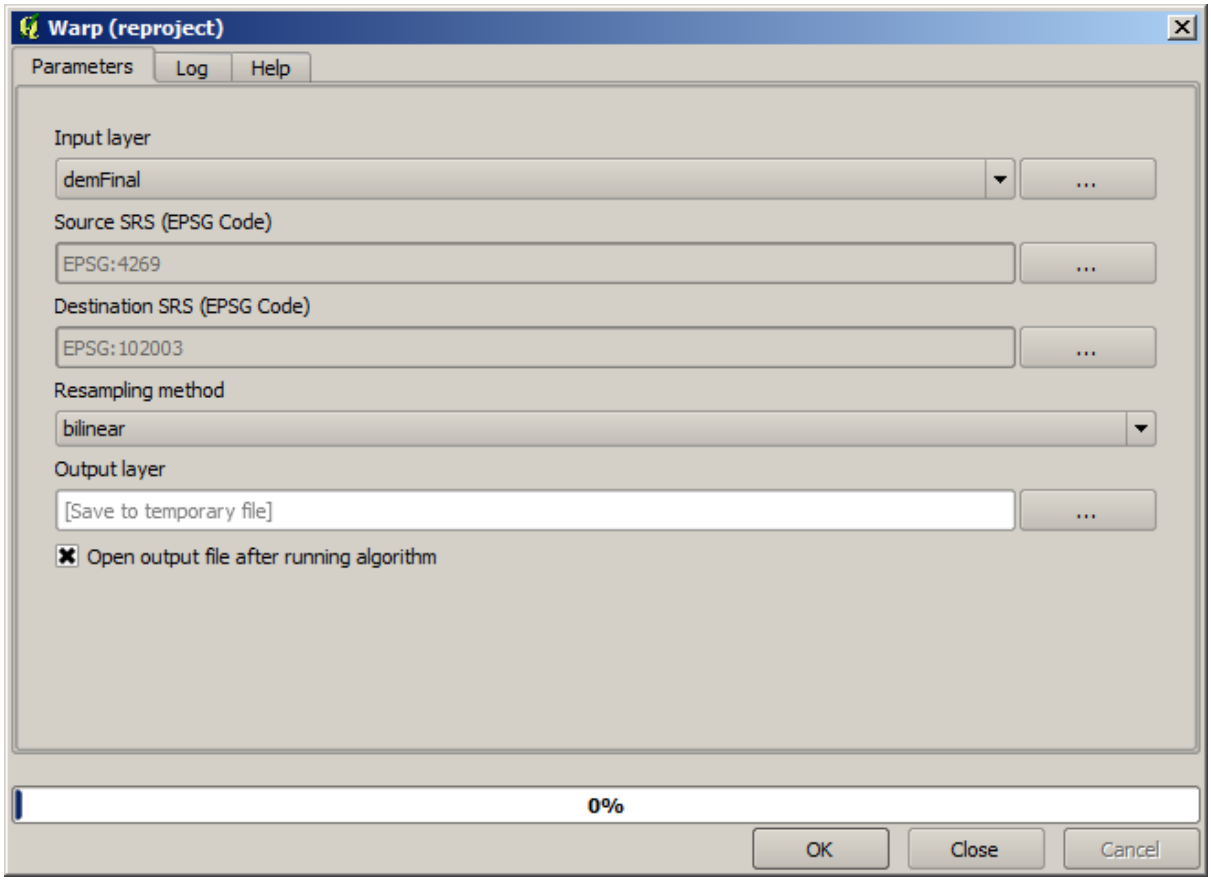
Here is the resulting bounding box obtained using the parameters shown above



It is a rounded box, but we can easily get the equivalent box with square angles, by running the *Polygon from layer extent* algorithm on it. We could have buffered the city limits first, and then calculate the extent rectangle, saving one step.

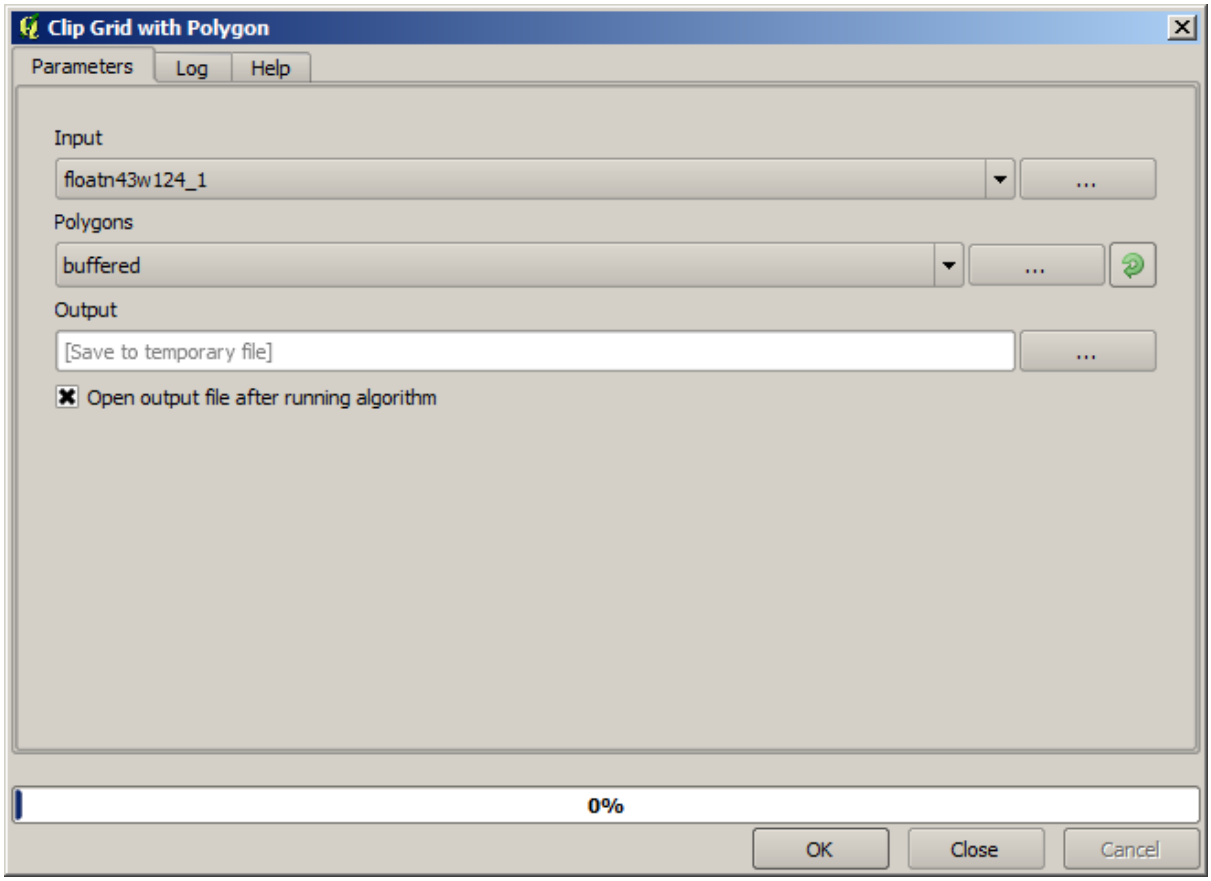


You will notice that the rasters has a different projection from the vector. We should therefore reproject them before proceeding further, using the *Warp (reproject)* tool.

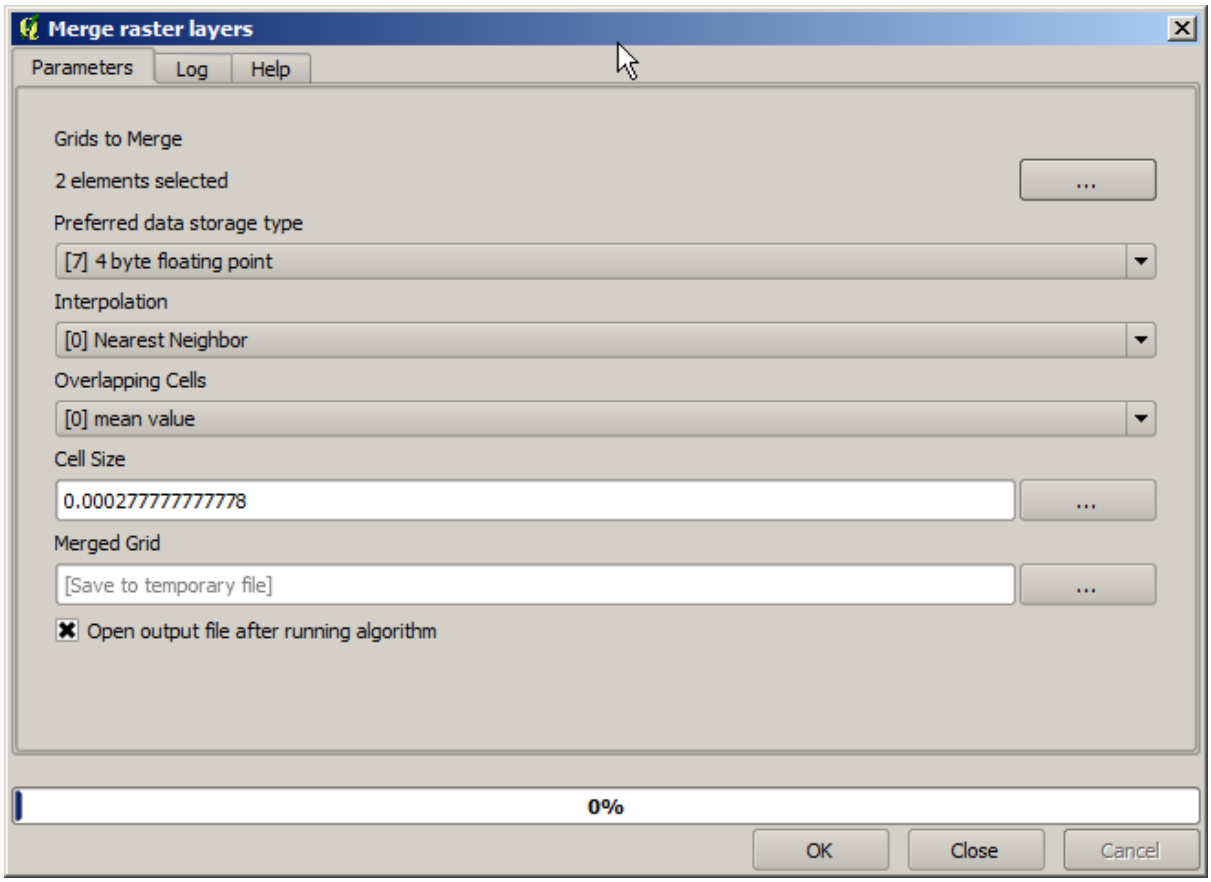


Nota: Recent versions have a more complex interface. Make sure at least one compression method is selected.

With this layer that contains the bounding box of the raster layer that we want to obtain, we can crop both of the raster layers, using the *Clip raster with polygon* algorithm.

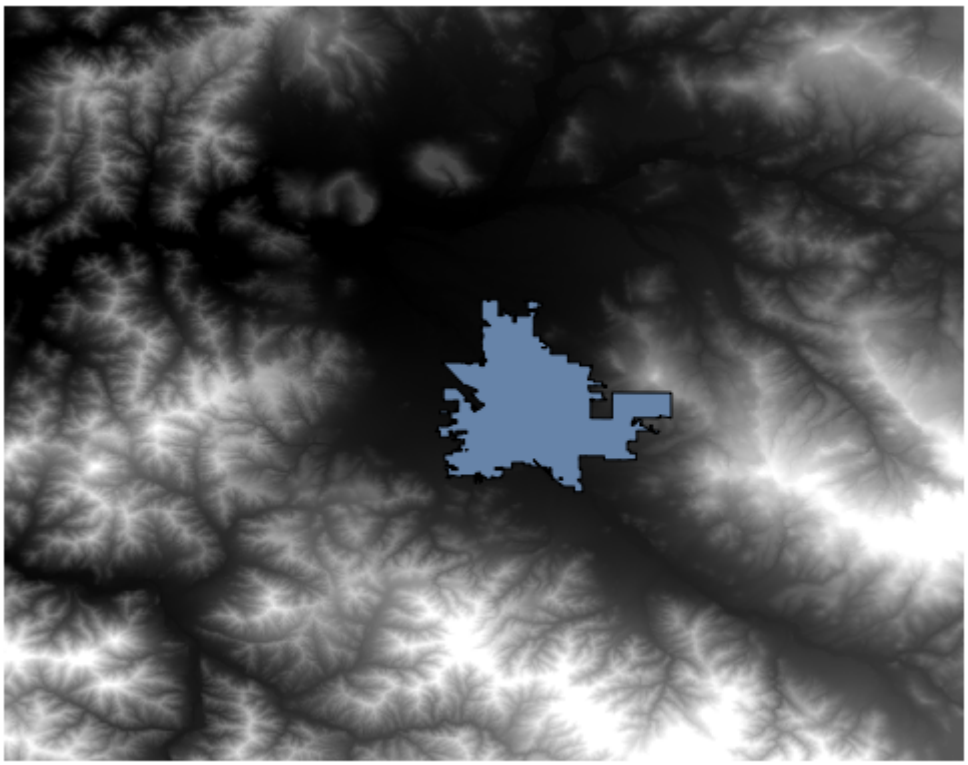


Once the layers have been cropped, they can be merged using the SAGA *Mosaic raster layers* algorithm.



Nota: You can save time merging first and then cropping, and you will avoid calling the clipping algorithm twice. However, if there are several layers to merge and they have a rather big size, you will end up with a large layer than it can later be difficult to process. In that case, you might have to call the clipping algorithm several times, which might be time consuming, but don't worry, we will soon see that there are some additional tools to automate that operation. In this example, we just have two layers, so you shouldn't worry about that now.

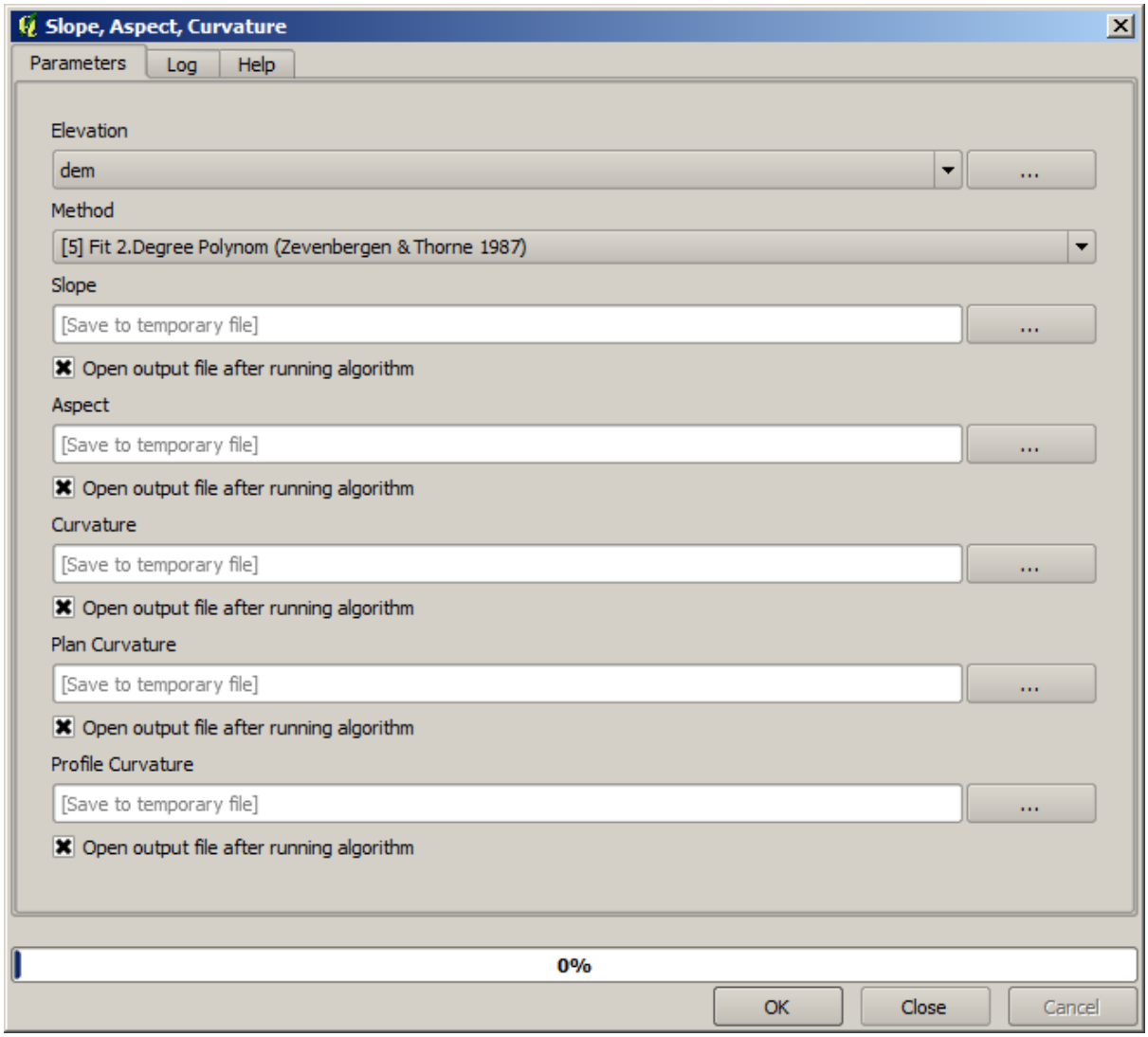
With that, we get the final DEM we want.



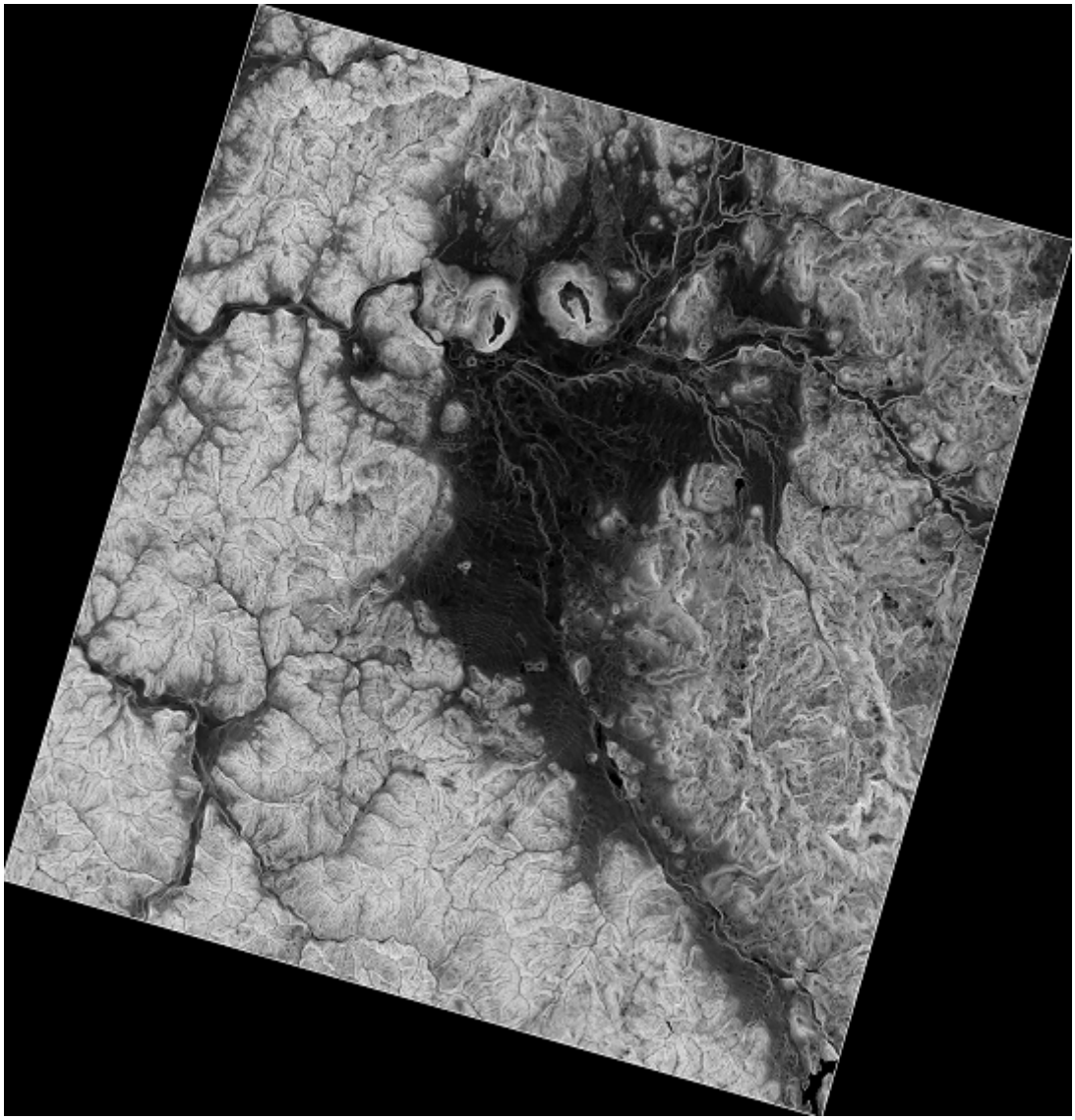
Now it is time to compute the slope layer.

A slope layer can be computed with the *Slope, Aspect, Curvature* algorithm, but the DEM obtained in the last step is not suitable as input, since elevation values are in meters but cellsize is not expressed in meters (the layer uses a CRS with geographic coordinates). A reprojection is needed. To reproject a raster layer, the *Warp (reproject)* algorithm can be used again. We reproject into a CRS with meters as units (e.g. 3857), so we can then correctly calculate the slope, with either SAGA or GDAL.

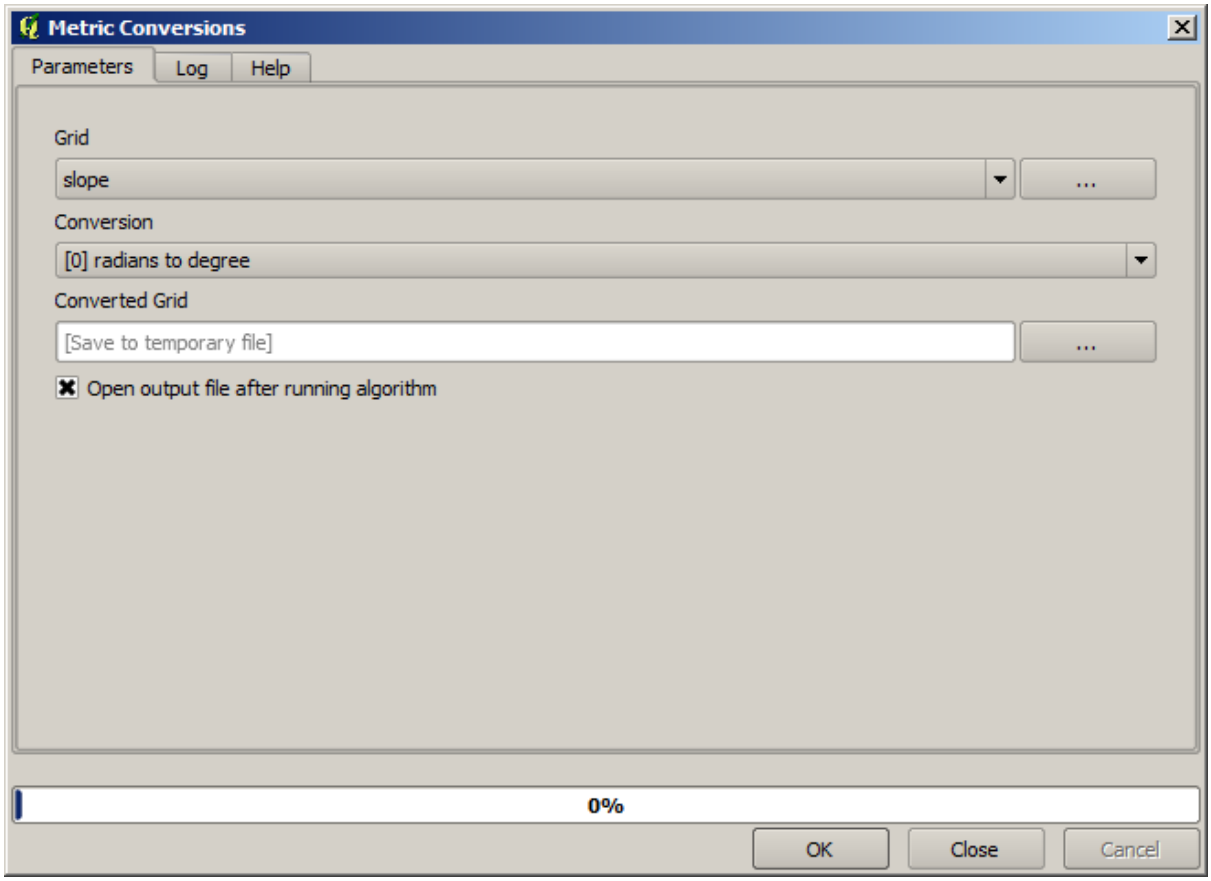
With the new DEM, slope can now be computed.



And here is the resulting slope layer.



The slope produced by the *Slope, Aspect, Curvature* algorithm can be expressed in degrees or radians; degrees are a more practical and common unit. In case you calculated it in radians, the *Metric conversions* algorithm will help us to do the conversion (but in case you didn't know that algorithm existed, you could use the raster calculator that we have already used).



Reprojecting the converted slope layer back with the *Reproject raster layer*, we get the final layer we wanted.

Aviso: todo: Add image

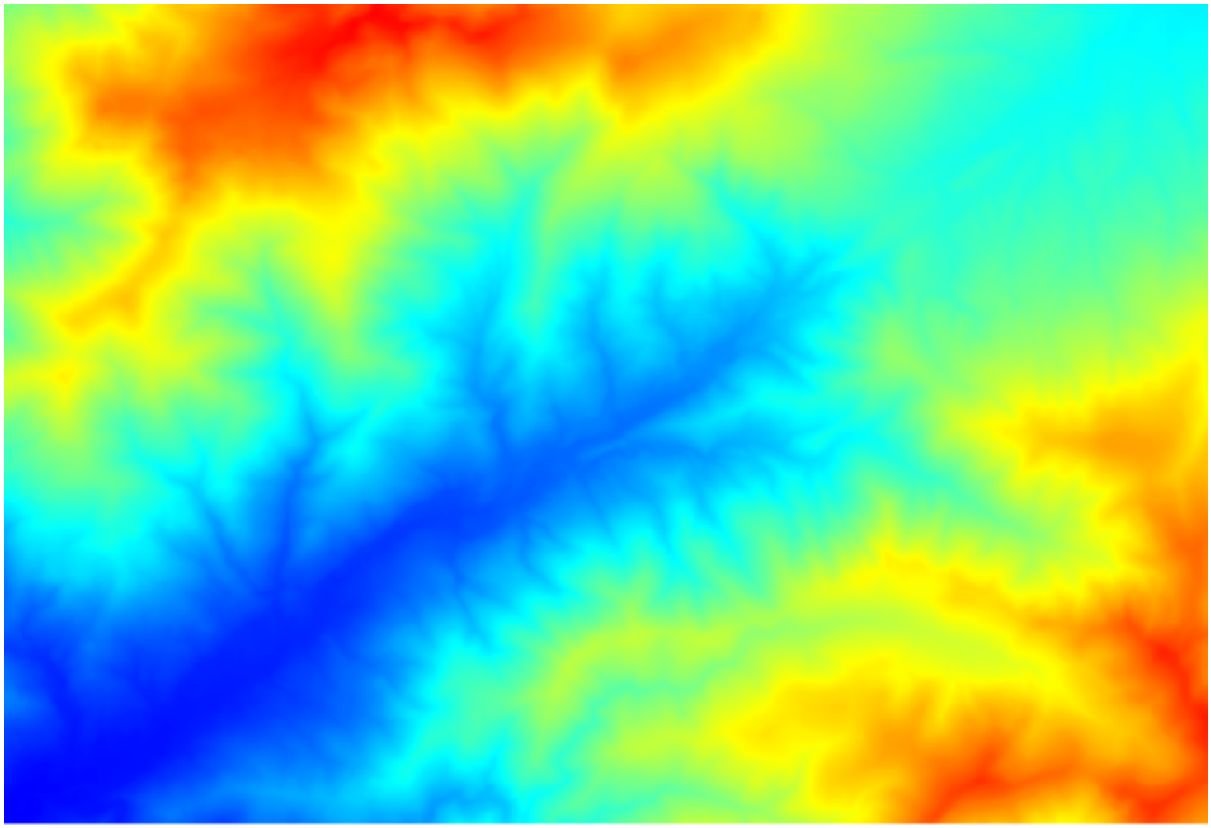
The reprojection processes might have caused the final layer to contain data outside the bounding box that we calculated in one of the first steps. This can be solved by clipping it again, as we did to obtain the base DEM.

17.16 Análise hidrológica

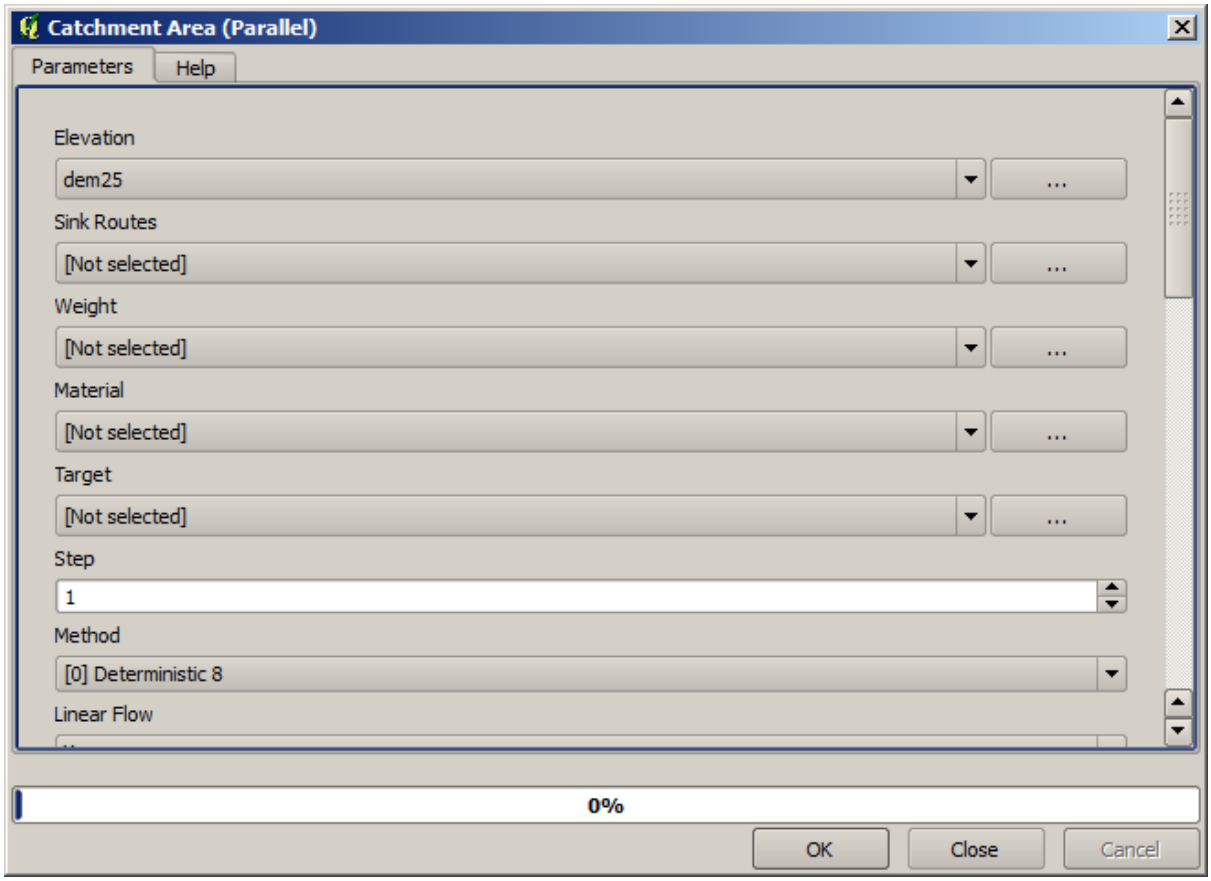
Nota: In this lesson we will perform some hydrological analysis. This analysis will be used in some of the following lessons, as it constitutes a very good example of an analysis workflow, and we will use it to demonstrate some advanced features.

Objectives: Starting with a DEM, we are going to extract a channel network, delineate watersheds and calculate some statistics.

1. The first thing is to load the project with the lesson data, which just contains a DEM.

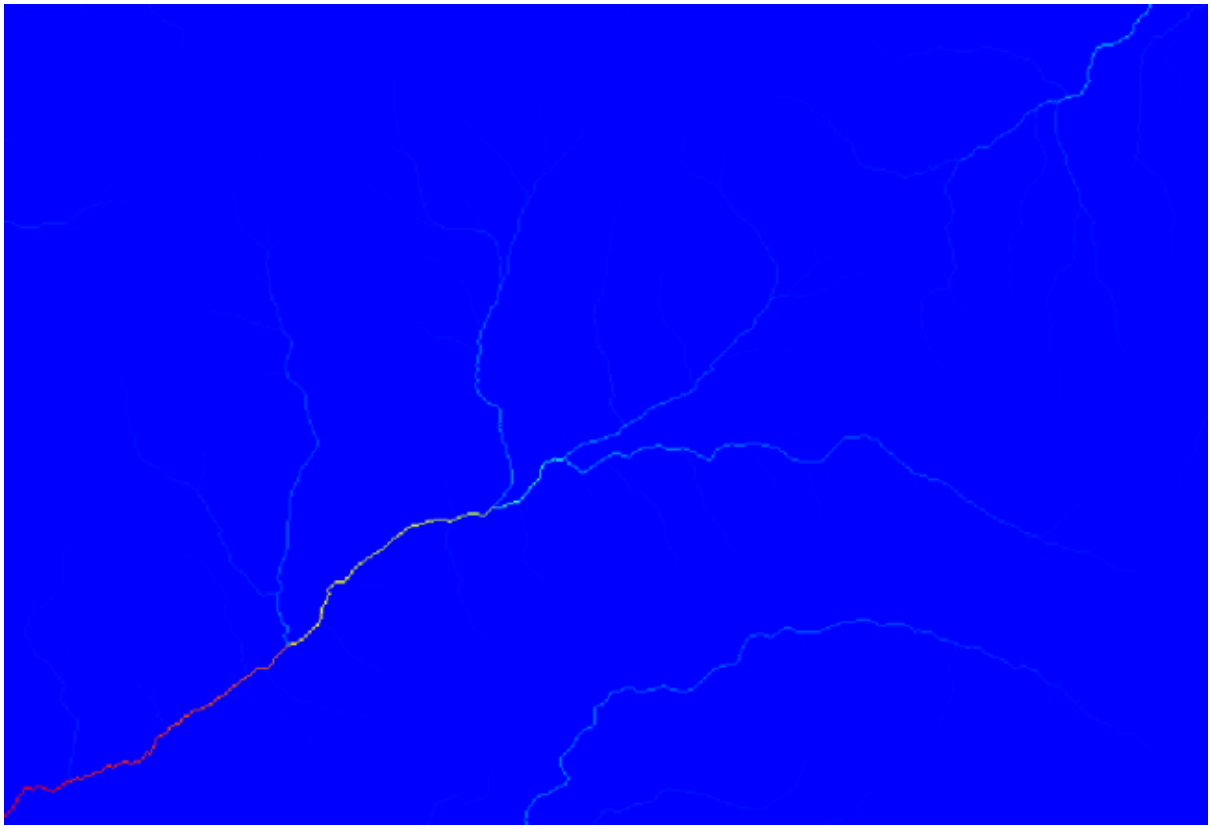


2. The first module to execute is *Catchment area* (in some SAGA versions it is called *Flow accumulation (Top Down)*). You can use any of the others named *Catchment area*. They have different algorithms underneath, but the results are basically the same.
3. Select the DEM in the *Elevation* field, and leave the default values for the rest of the parameters.

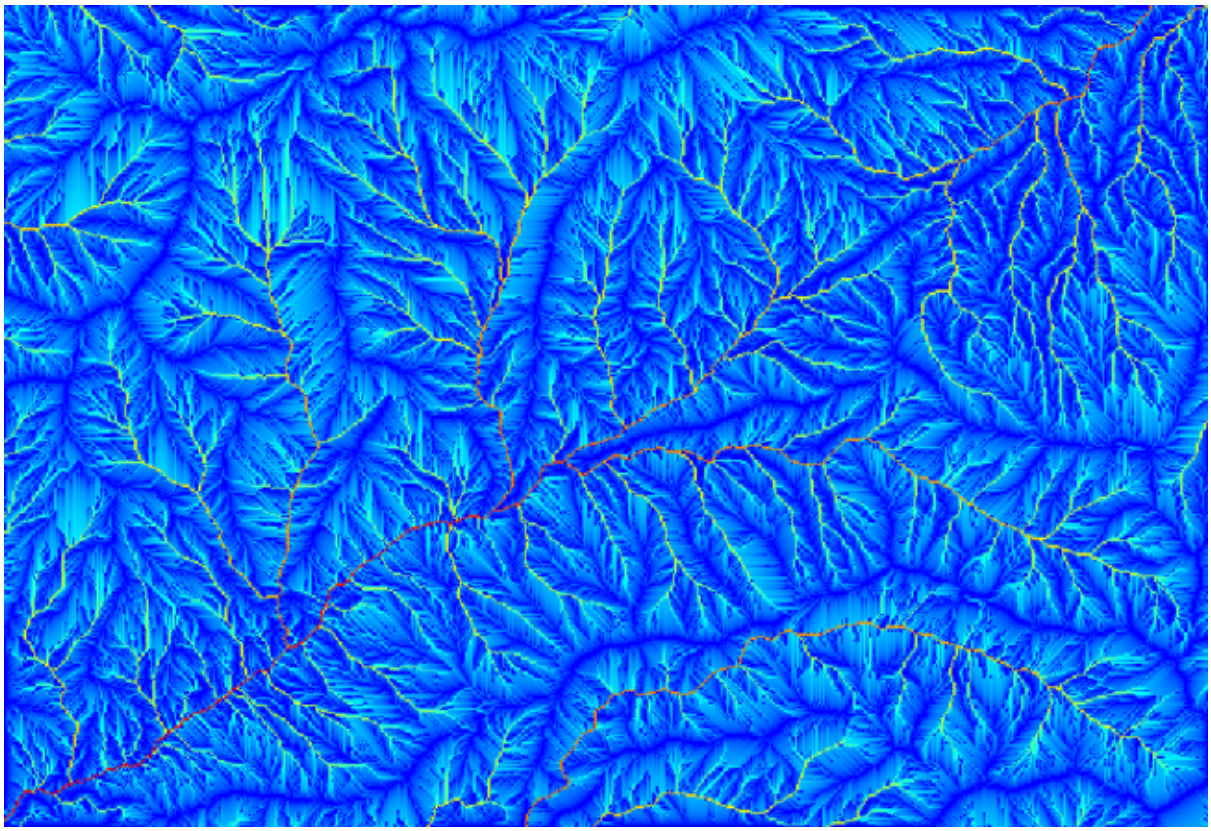


Some algorithms calculate many layers, but the *Catchment Area* layer is the only one we will be using. You can get rid of the other ones if you want.

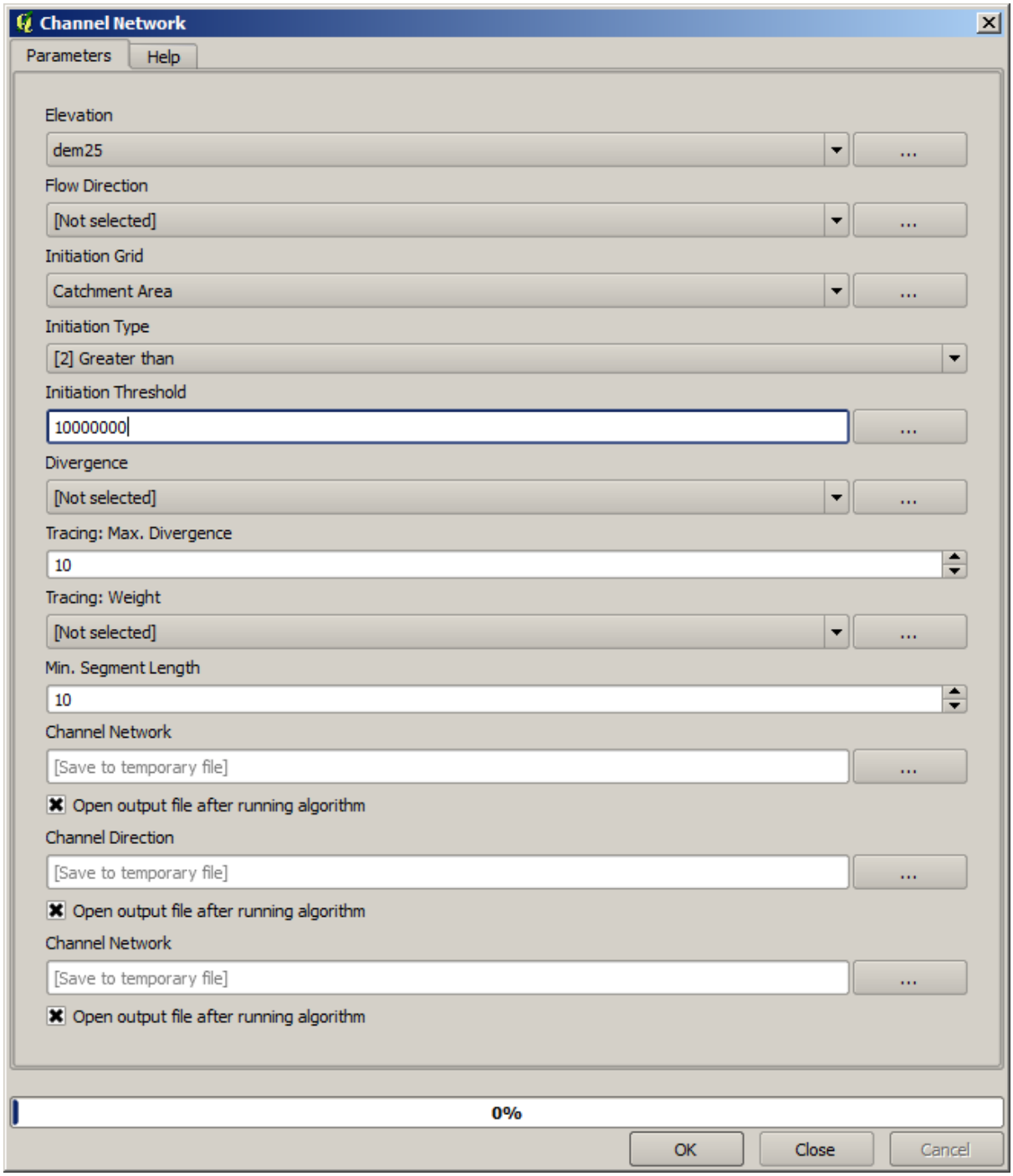
The rendering of the layer is not very informative.



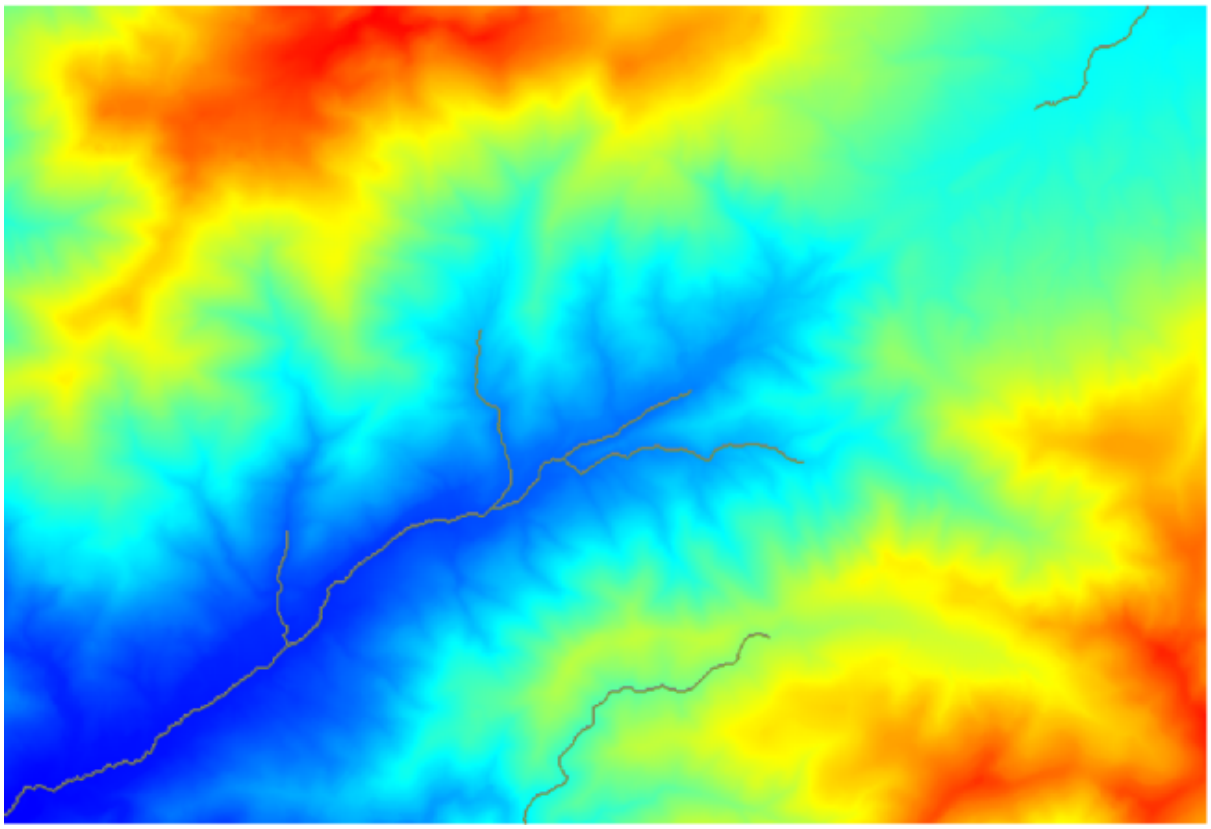
To know why, you can have a look at the histogram and you will see that values are not evenly distributed (there are a few cells with very high value, those corresponding to the channel network). Use the *Raster calculator* algorithm to calculate the logarithm of the catchment value area and you will get a layer with much more information



4. The catchment area (also known as flow accumulation) can be used to set a threshold for channel initiation. This can be done using the *Channel network* algorithm.
   - *Initiation grid*: use the catchment area layer and not the logarithm one.
   - *Initiation threshold*: `10.000.000`
   - *Initiation type*: *Greater than*

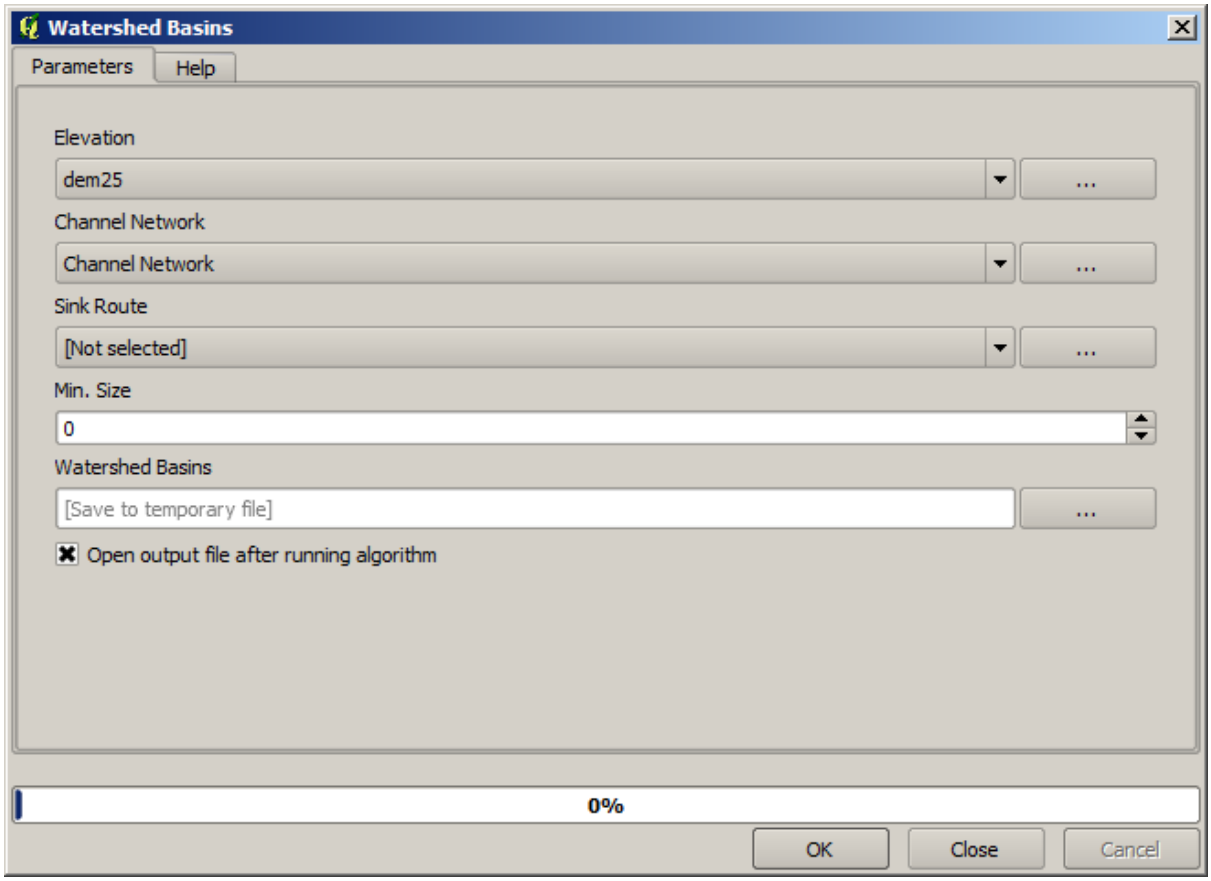


If you increase the *Initiation threshold* value, you will get a more sparse channel network. If you decrease it, you will get a denser one. With the proposed value, this is what you get.

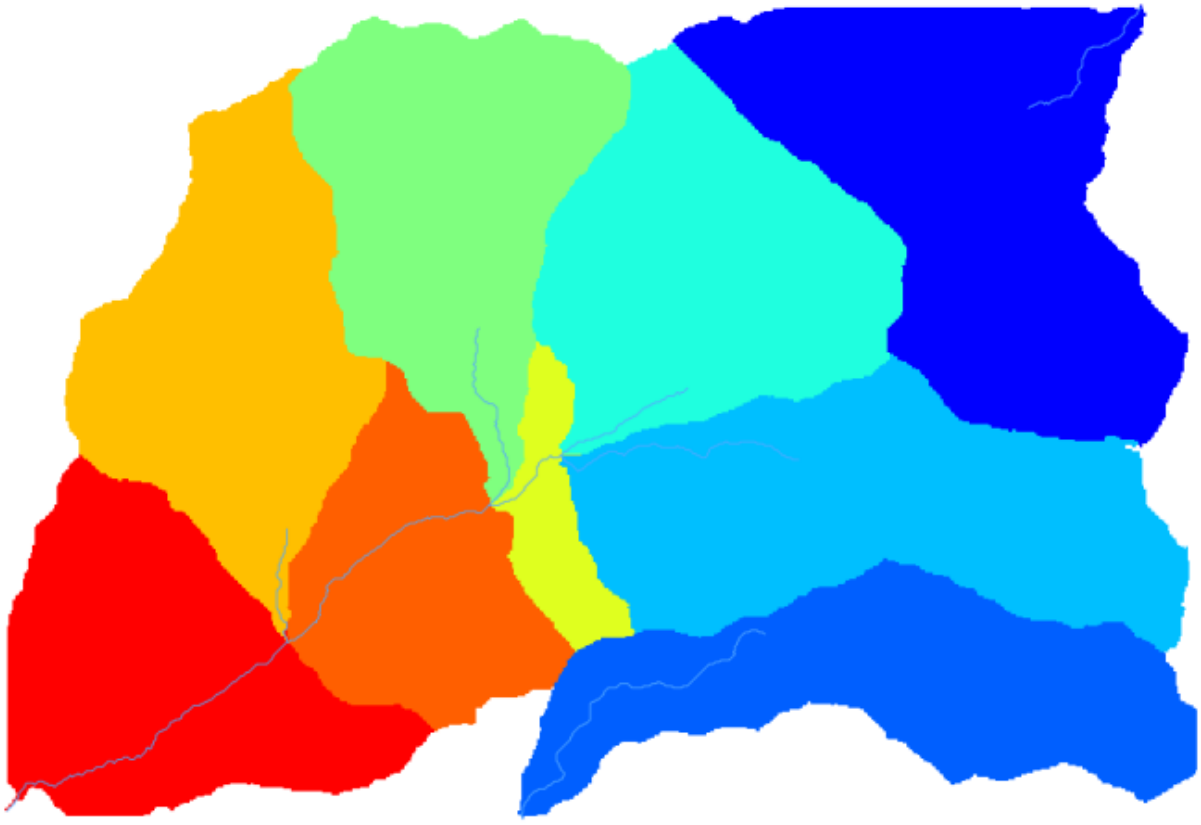


The image above shows just the resulting vector layer and the DEM, but there should be also a raster layer with the same channel network. That raster will be, in fact, the layer we will be using.

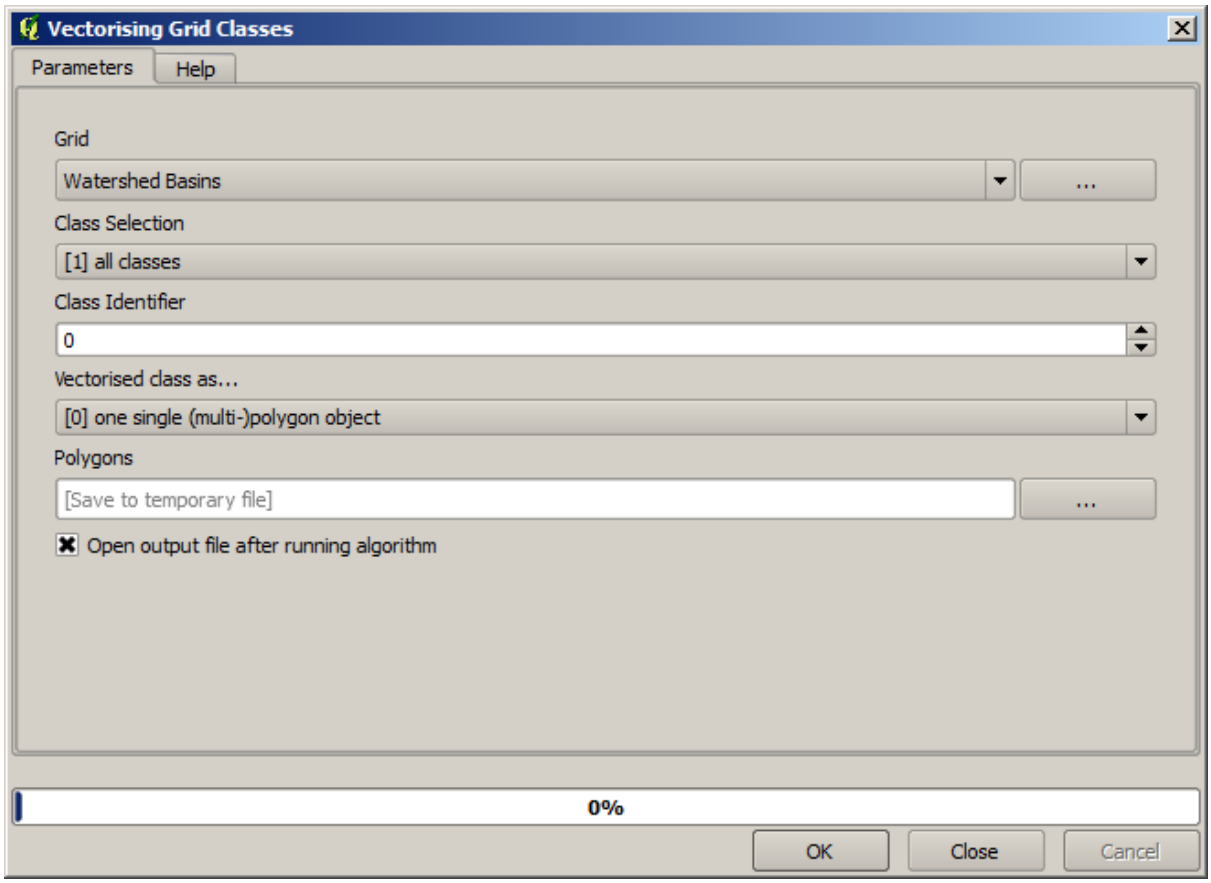
5. Now, we will use the *Watersheds basins* algorithm to delineate the subbasins corresponding to that channel network, using as outlet points all the junctions in it. Here is how you have to set the corresponding parameters dialog.

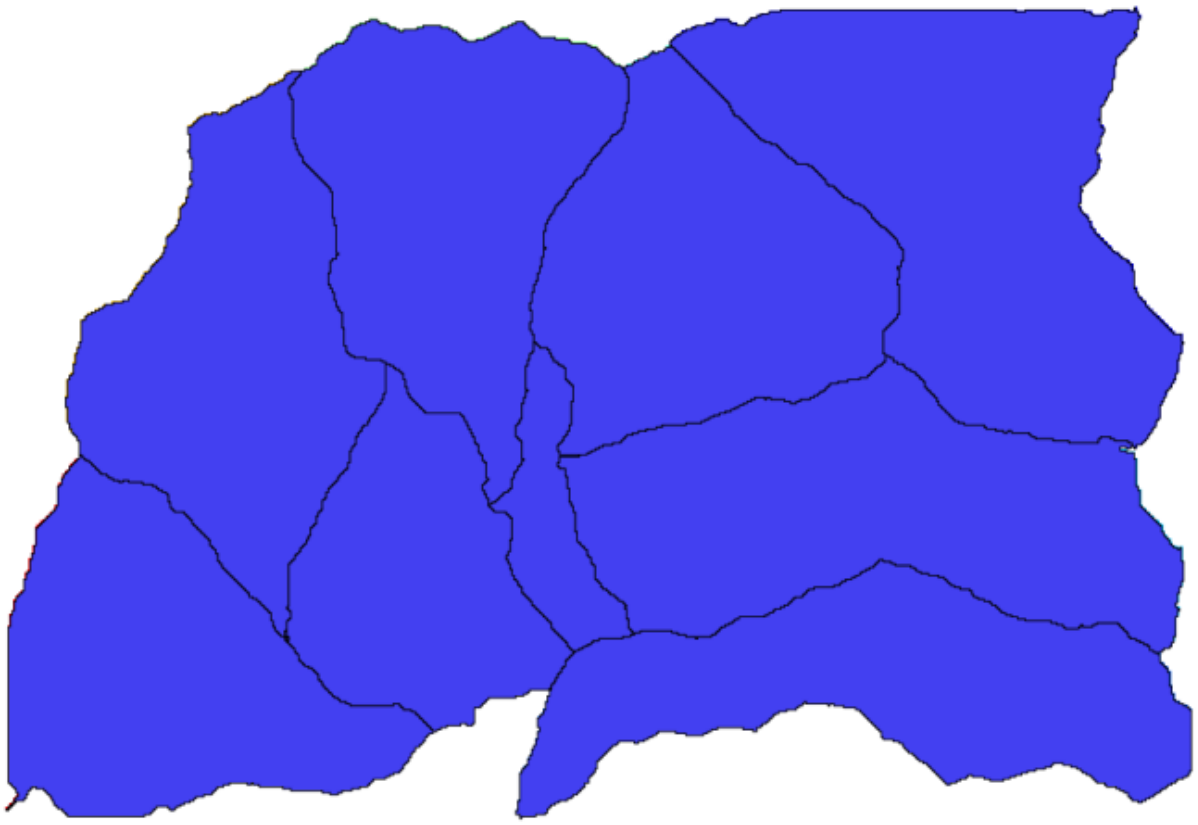


And this is what you will get.



6. This is a raster result. You can vectorise it using the *Vectorising grid classes* algorithm.

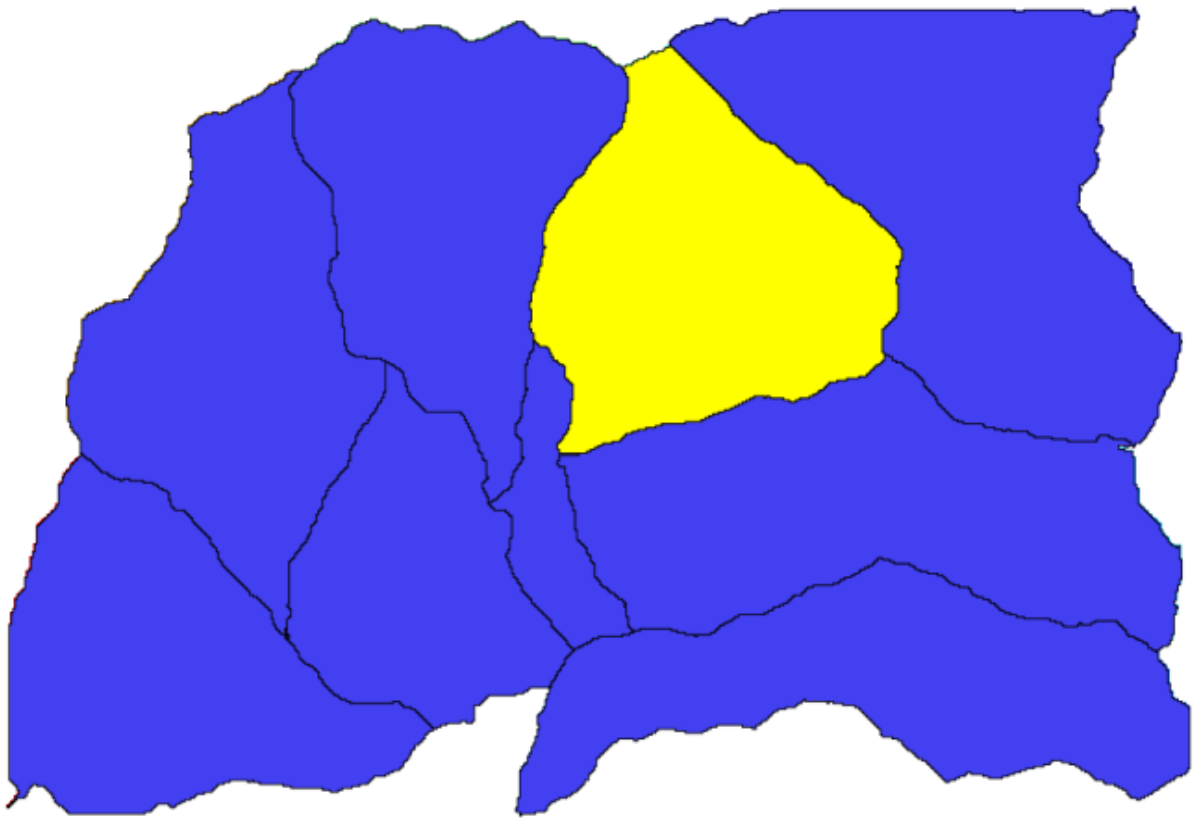




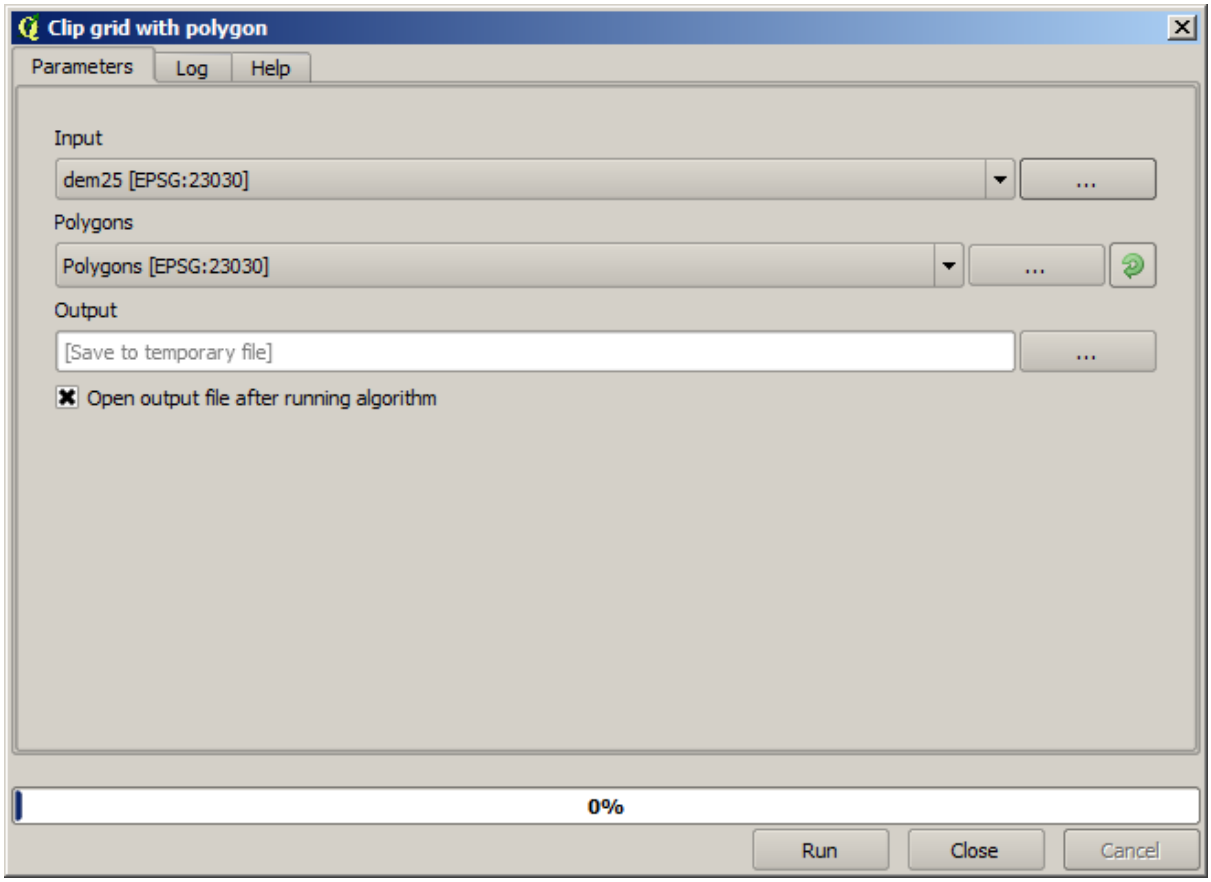
Now, let's try to compute statistics about the elevation values in one of the subbasins. The idea is to have a layer that just represents the elevation within that subbasin and then pass it to the module that calculates those statistics.

1. First, let's clip the original DEM with the polygon representing a subbasin. We will use the *Clip raster with polygon* algorithm. If we select a single subbasin polygon and then call the clipping algorithm, we can clip the DEM to the area covered by that polygon, since the algorithm is aware of the selection.

    1. Select a polygon

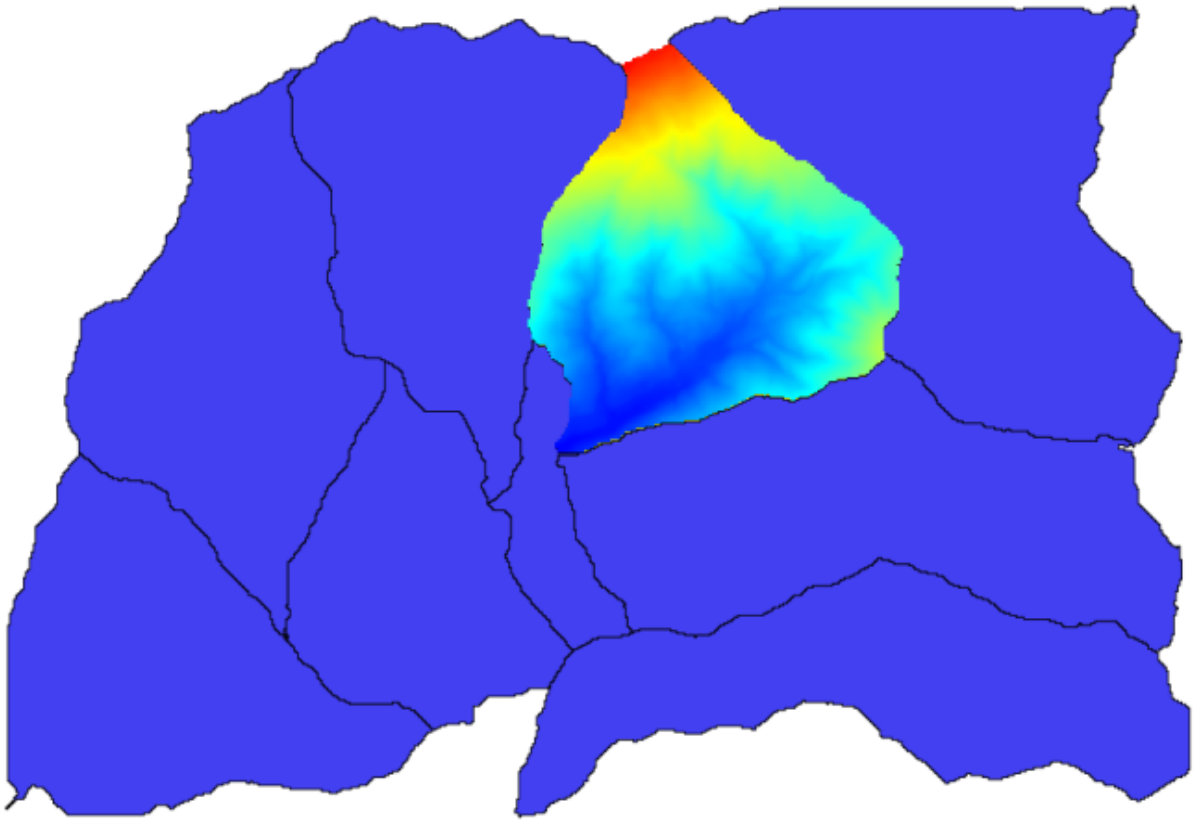


2. Call the clipping algorithm with the following parameters:

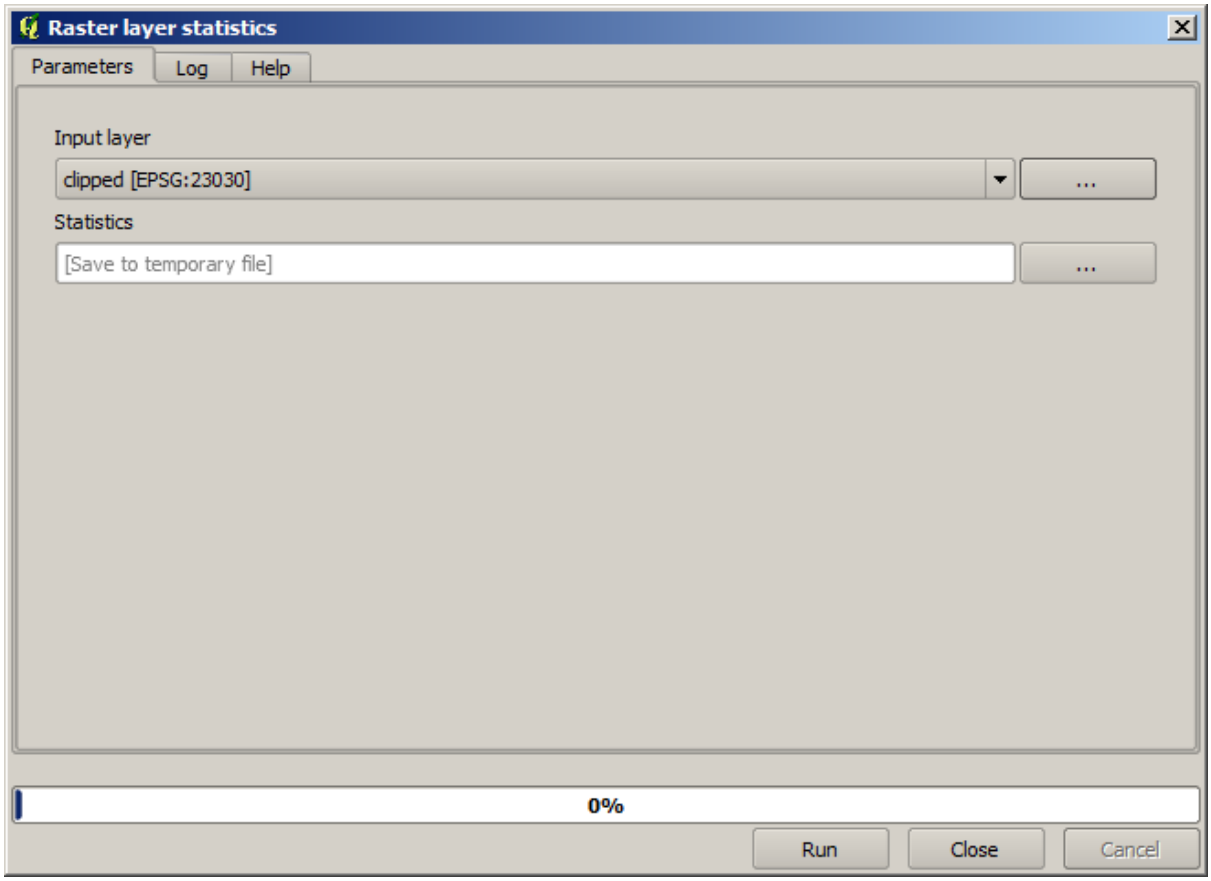


The element selected in the input field is, of course, the DEM we want to clip.

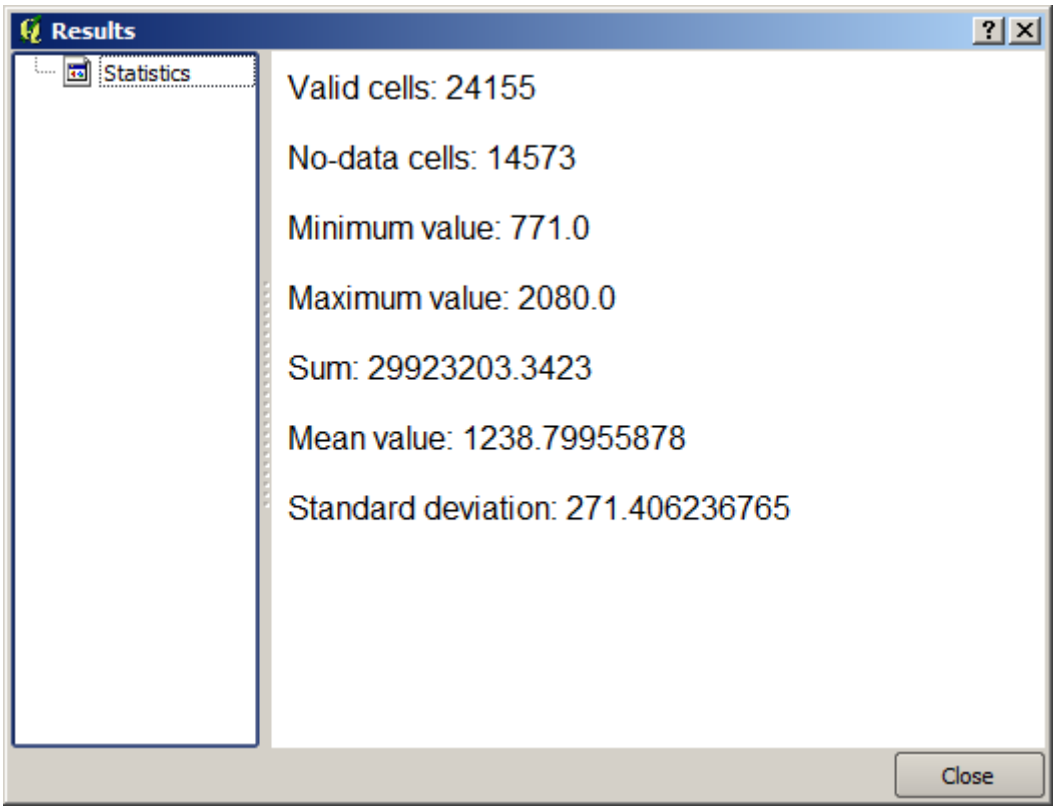
You will get something like this.



2. This layer is ready to be used in the *Raster layer statistics* algorithm.



The resulting statistics are the following ones.



We will use both the basin calculations procedure and the statistics calculation in other lessons, to find out how other

elements can help us automate both of them and work more effectively.

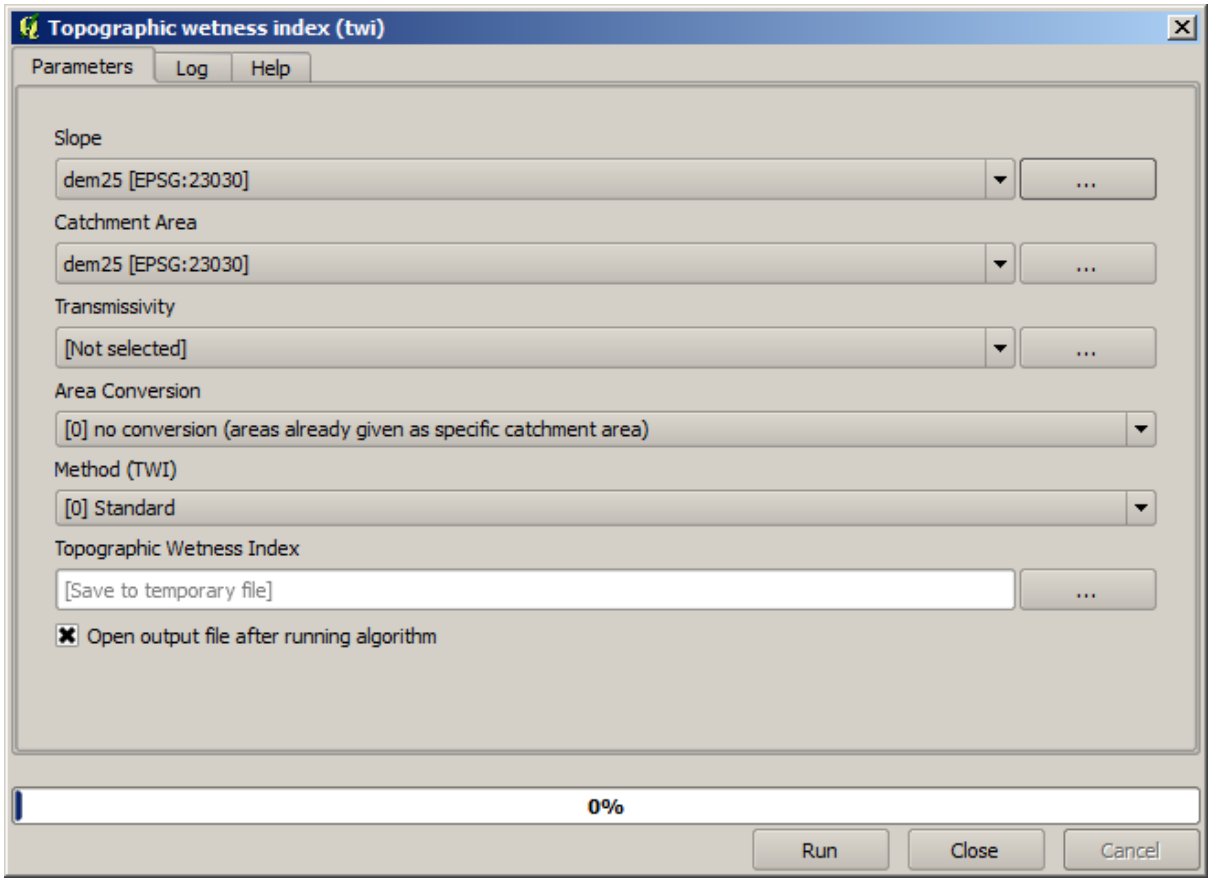
17.17 Starting with the model designer

Nota: In this lesson we will use the model designer, a powerful component that we can use to define a workflow and run a chain of algorithms.

A normal session with the processing tools includes more than running a single algorithm. Usually several of them are run to obtain a result, and the outputs of some of those algorithms are used as input for some of the other ones.

Using the model designer, that workflow can be put into a model, which will run all the necessary algorithms in a single run, thus simplifying the whole process and automating it.

To start this lesson, we are going to calculate a parameter named Topographic Wetness Index. The algorithm that computes it is called *Topographic wetness index (twi)*.

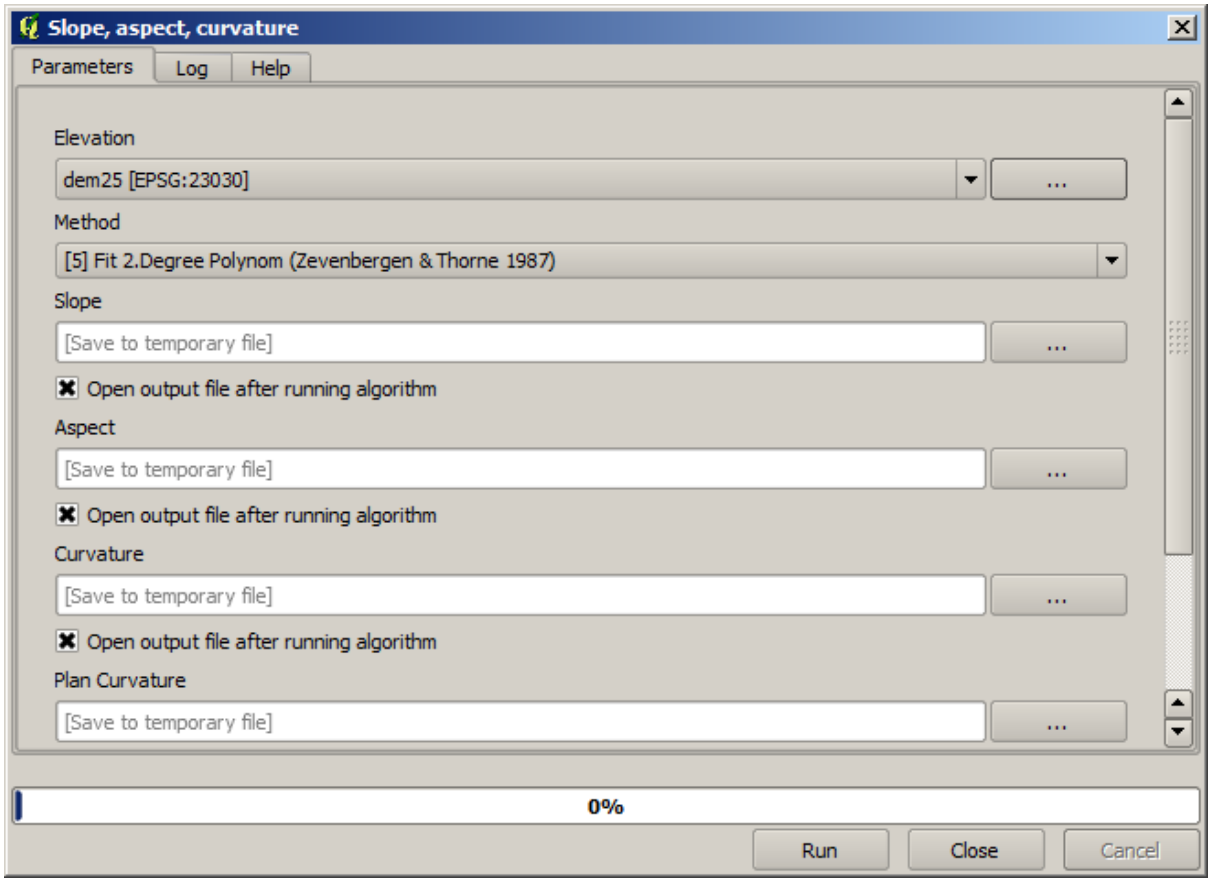


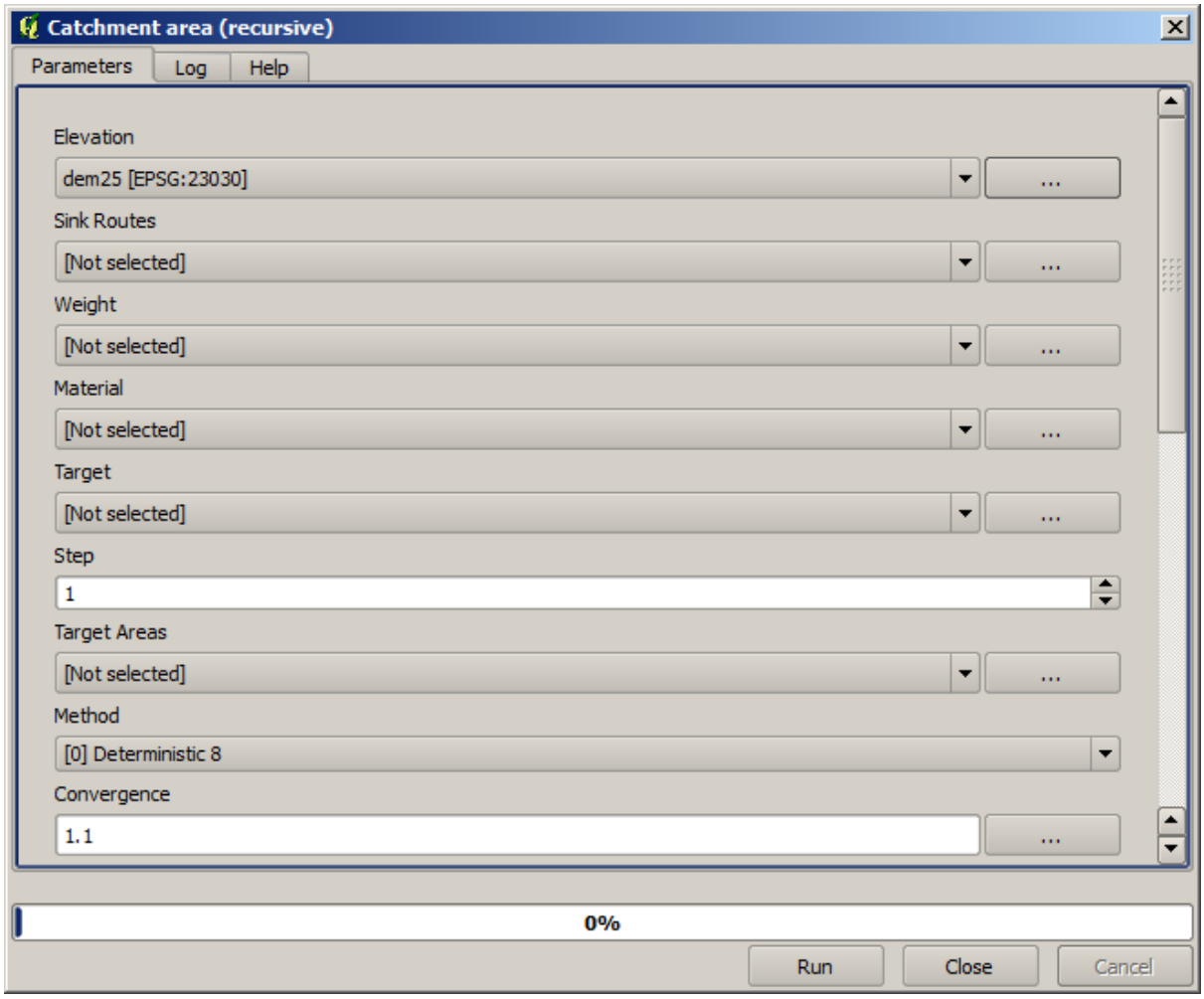
As you can see, there are two mandatory inputs: *Slope* and *Catchment area*. There is also an optional input, but we will not be using it, so we can ignore it.

The data for this lesson contains just a DEM, so we do not have any of the required inputs. However, we know how to calculate both of them from that DEM, since we have already seen the algorithms to compute slope and catchment area. So we can first compute those layers and then use them for the TWI algorithm.

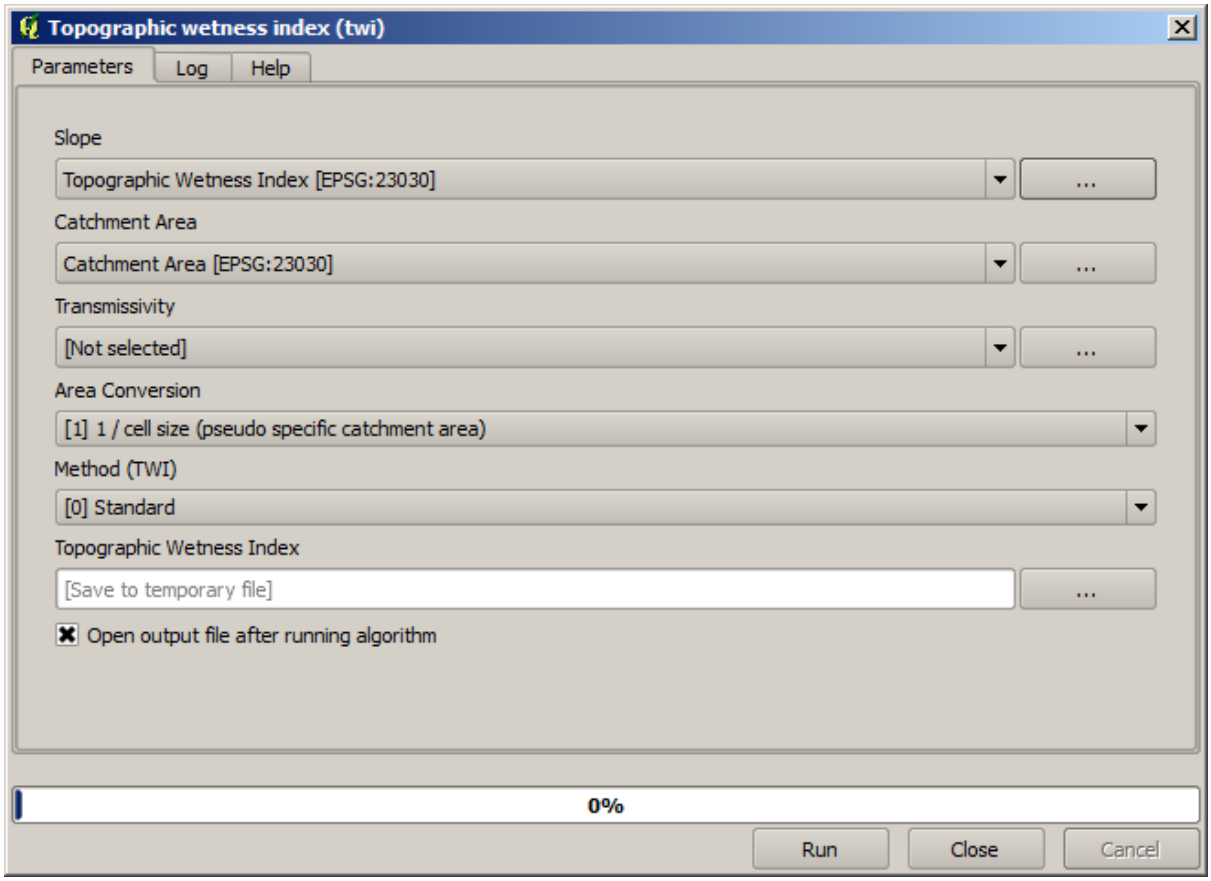
Here are the parameter dialogs that you should use to calculate the 2 intermediate layers.

Nota: Slope must be calculated in radians, not in degrees.

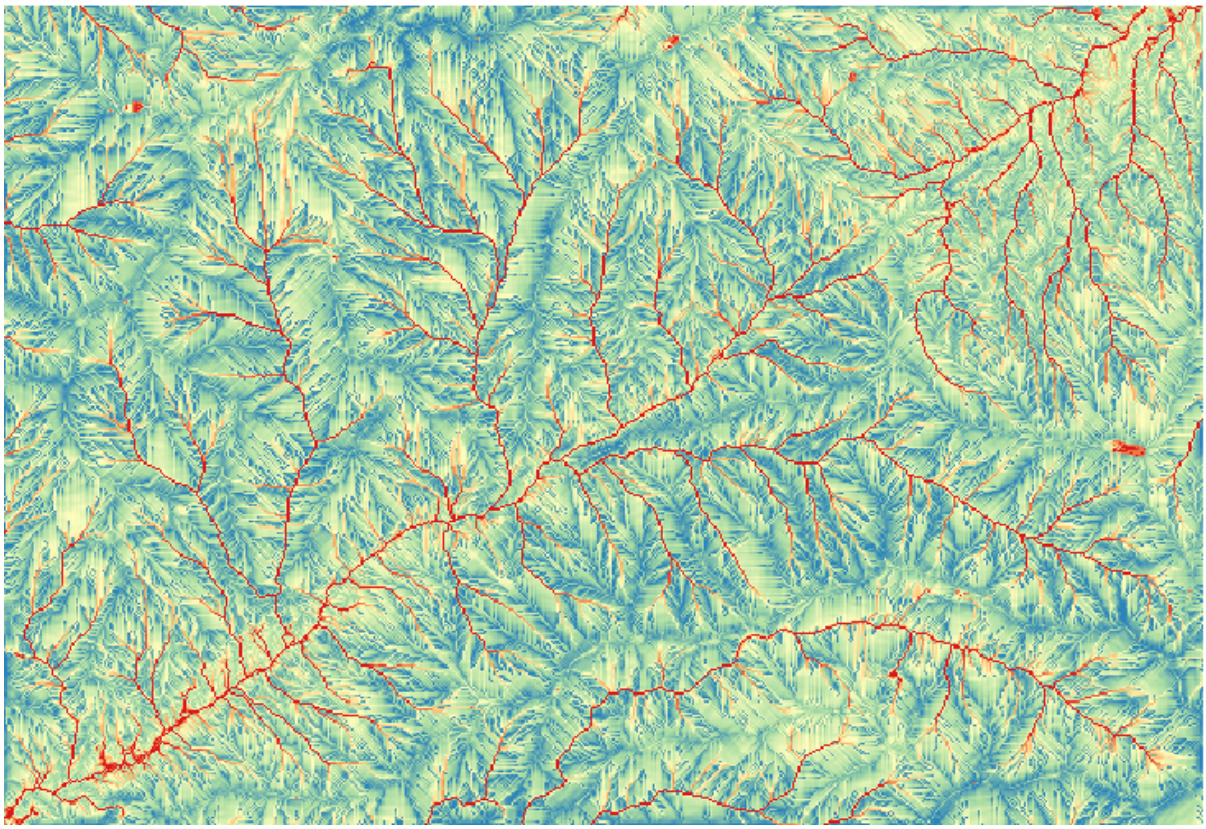




And this is how you have to set the parameters dialog of the TWI algorithm.

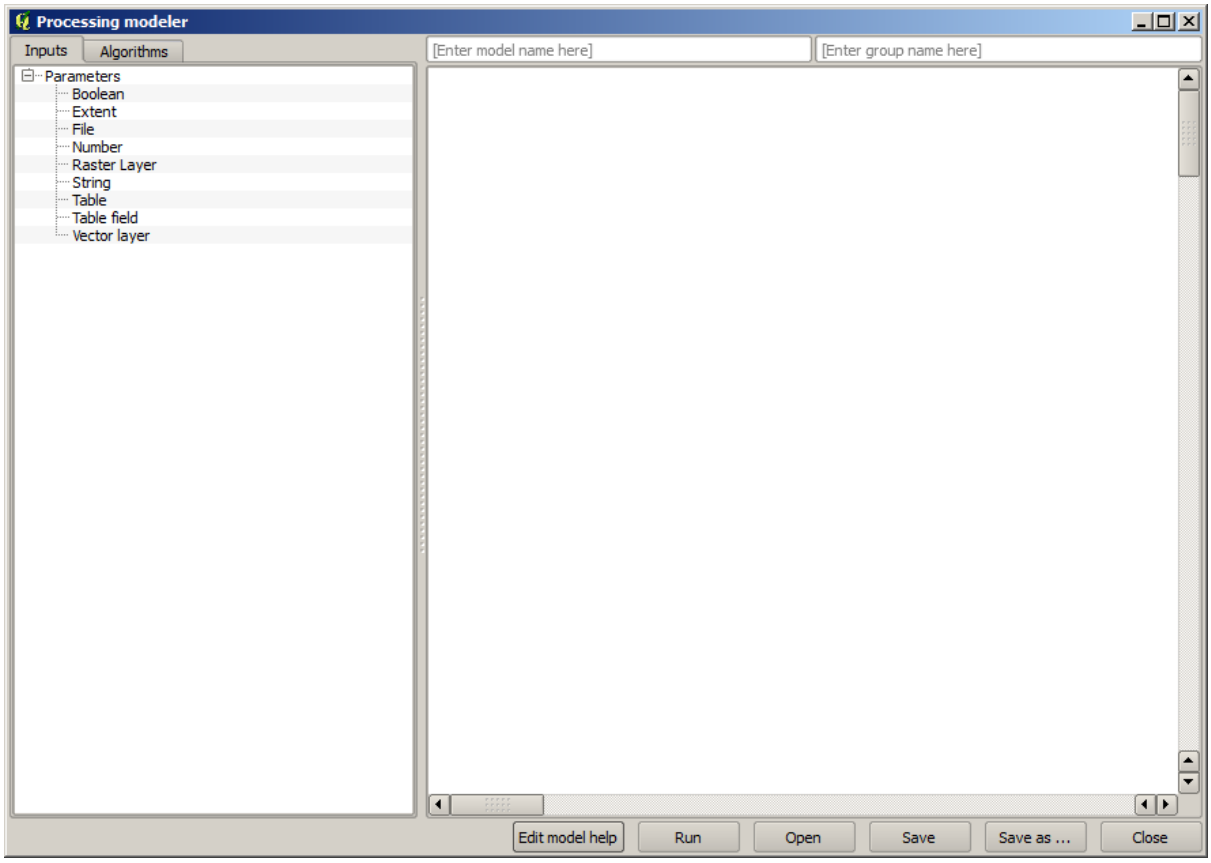


This is the result that you will obtain (the default singleband pseudocolor inverted palette has been used for rendering). You can use the `twi.qml` style provided.



What we will try to do now is to create an algorithm that calculates the TWI from a DEM in just one single step. That will save us work in case we later have to compute a TWI layer from another DEM, since we will need just one single step to do it instead of the three above. All the processes that we need are found in the toolbox, so what we have to do is to define the workflow to wrap them. This is where the model designer comes in.

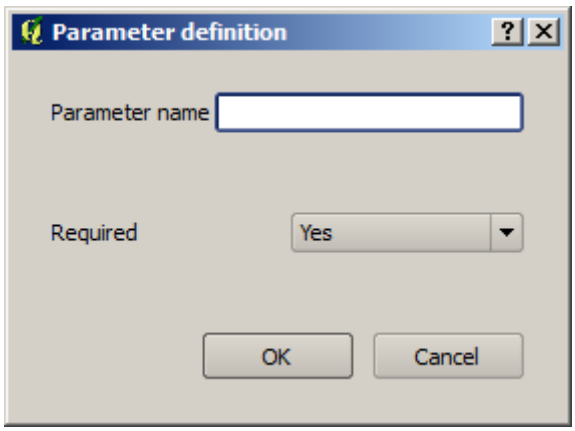
1. Open the modeler by selecting its menu entry in the processing menu.



   Two things are needed to create a model: setting the inputs that it will need, and defining the algorithm that it contains. Both of them are done by adding elements from the two tabs in the left-hand side of the modeler window: *Inputs* and *Algorithms*.

2. Let's start with the inputs. In this case we do not have much to add. We just need a raster layer with the DEM, and that will be our only input data.

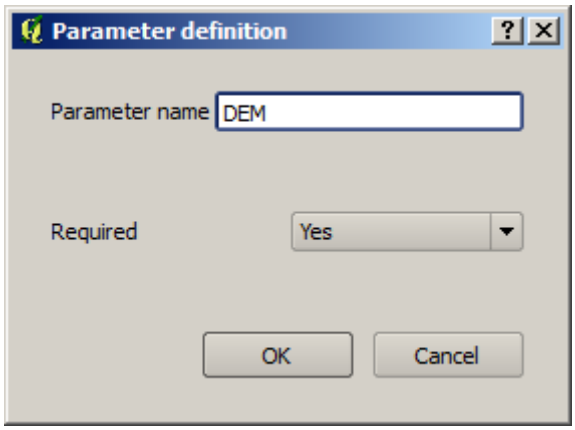
3. Double click on the *Raster layer* input and you will see the following dialog.



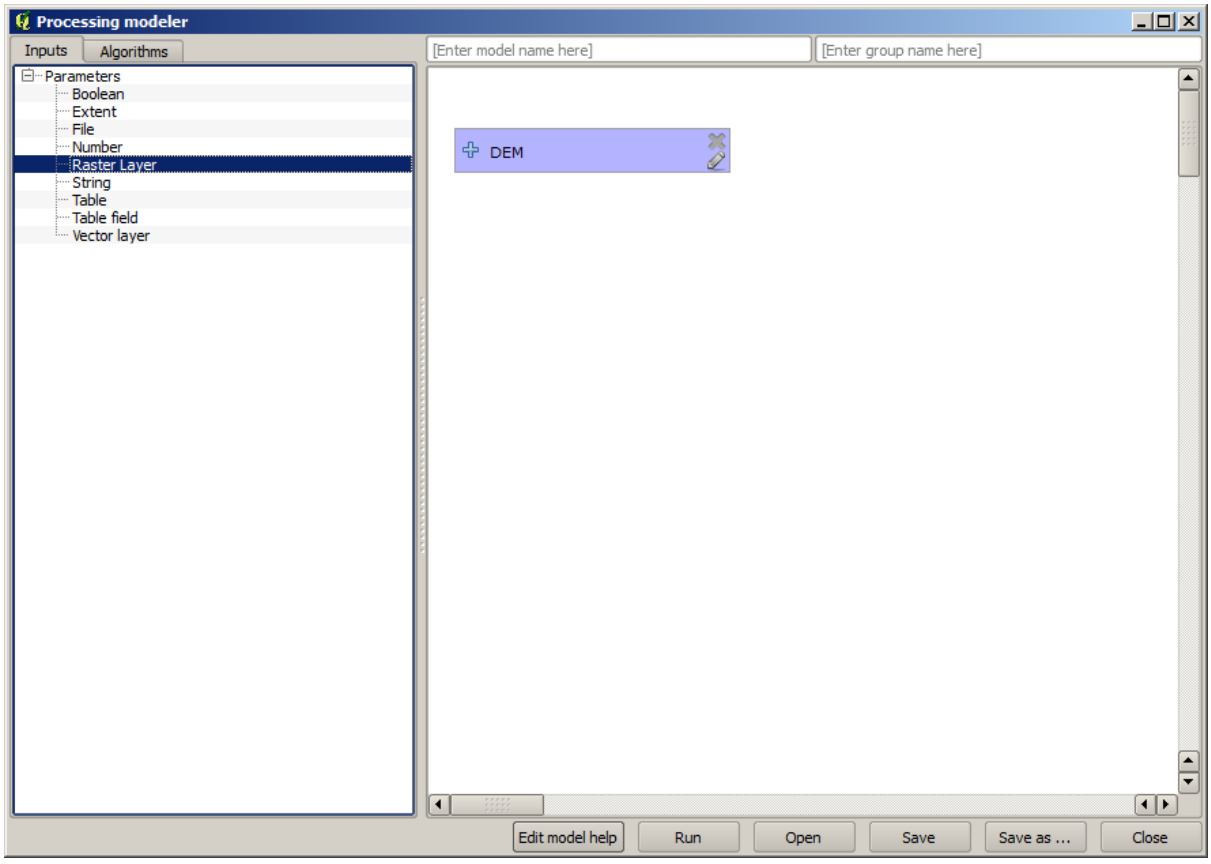
4. Here we will have to define the input we want:

1. Since we expect this raster layer to be a DEM, we will call it `DEM`. That's the name that the user of the model will see when running it.
2. Since we need that layer to work, we will define it as a mandatory layer.

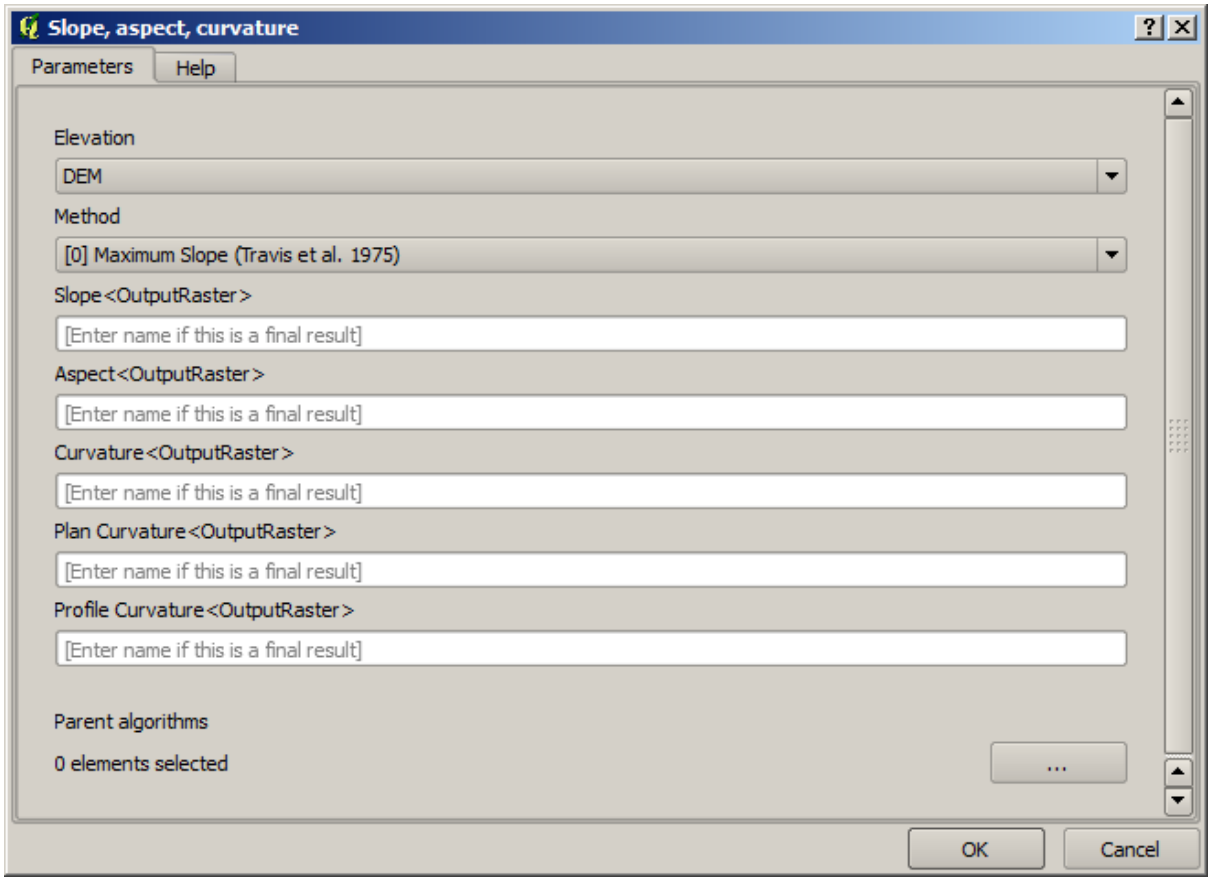
Here is how the dialog should be configured.



5. Click on *OK* and the input will appear in the modeler canvas.



6. Now let's move to the *Algorithms* tab.
7. The first algorithm we have to run is the *Slope, aspect, curvature* algorithm. Locate it in the algorithm list, double-click on it and you will see the dialog shown below.

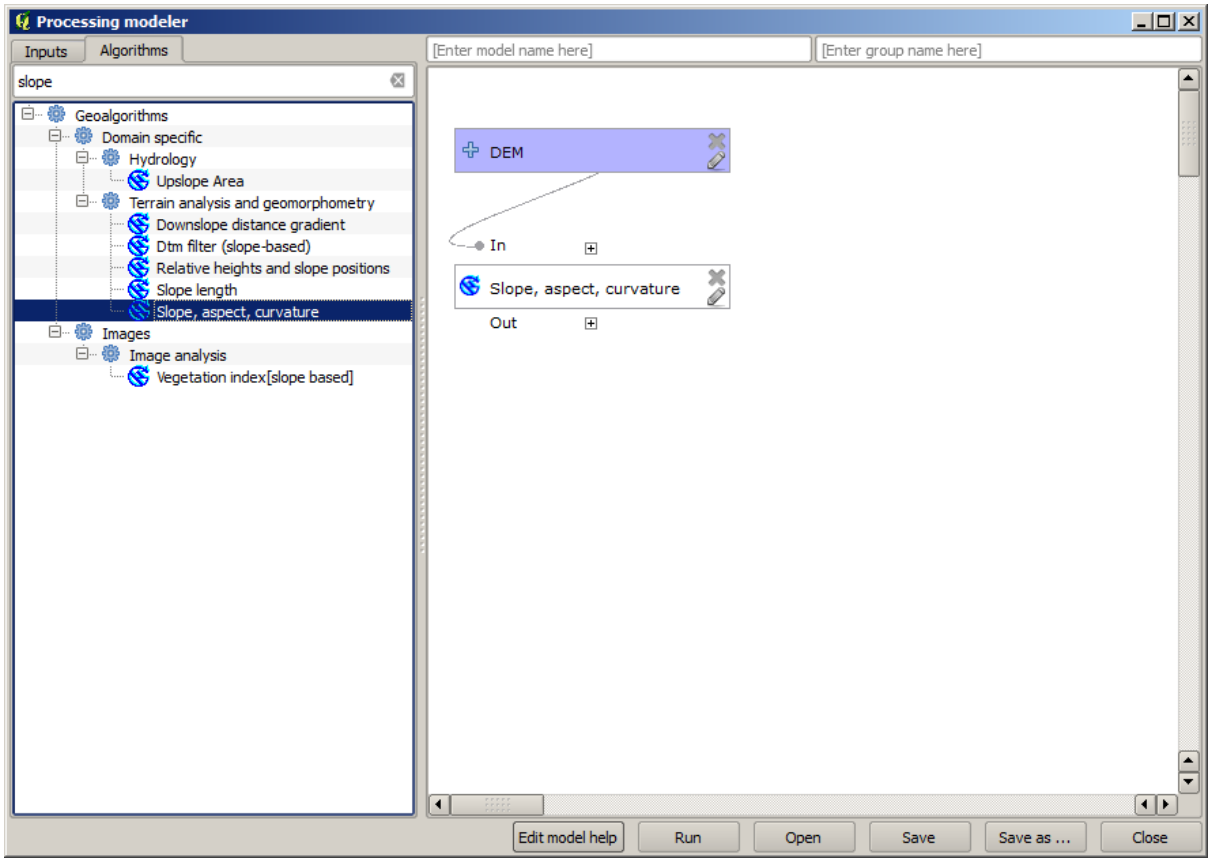


This dialog is very similar to the one that you can find when running the algorithm from the toolbox, but the element that you can use as parameter values are not taken from the current QGIS project, but from the model itself. That means that, in this case, we will not have all the raster layers of our project available for the *Elevation* field, but just the ones defined in our model. Since we have added just one single raster input named `DEM`, that will be the only raster layer that we will see in the list corresponding to the *Elevation* parameter.

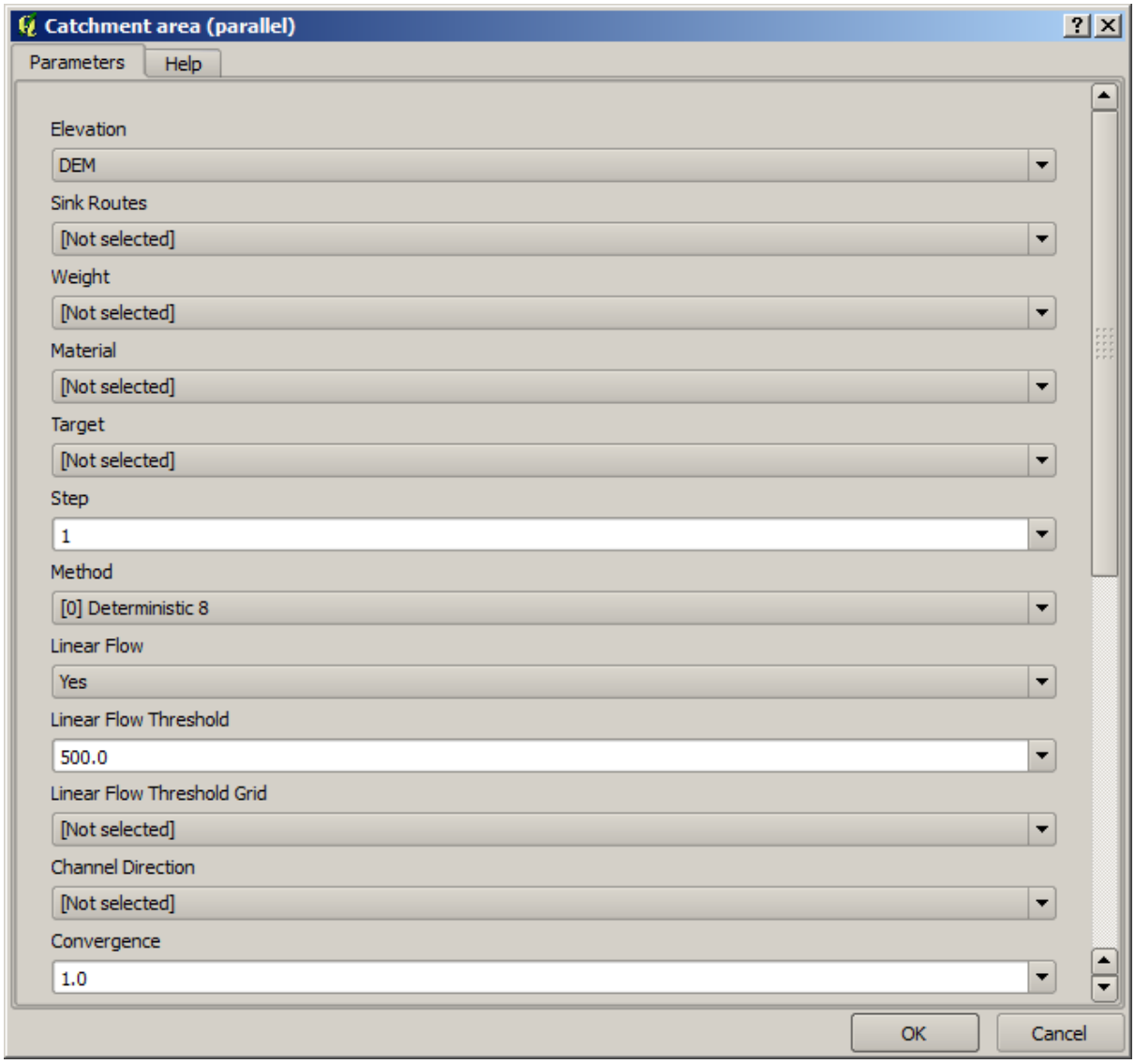
Output generated by an algorithm are handled a bit differently when the algorithm is used as a part of a model. Instead of selecting the filepath where you want to save each output, you just have to specify if that output is an intermediate layer (and you do not want it to be preserved after the model has been executed), or it is a final one. In this case, all layers produced by this algorithm are intermediate. We will only use one of them (the slope layer), but we do not want to keep it, since we just need it to calculate the TWI layer, which is the final result that we want to obtain.

When layers are not a final result, you should just leave the corresponding field. Otherwise, you have to enter a name that will be used to identify the layer in the parameters dialog that will be shown when you run the model later.

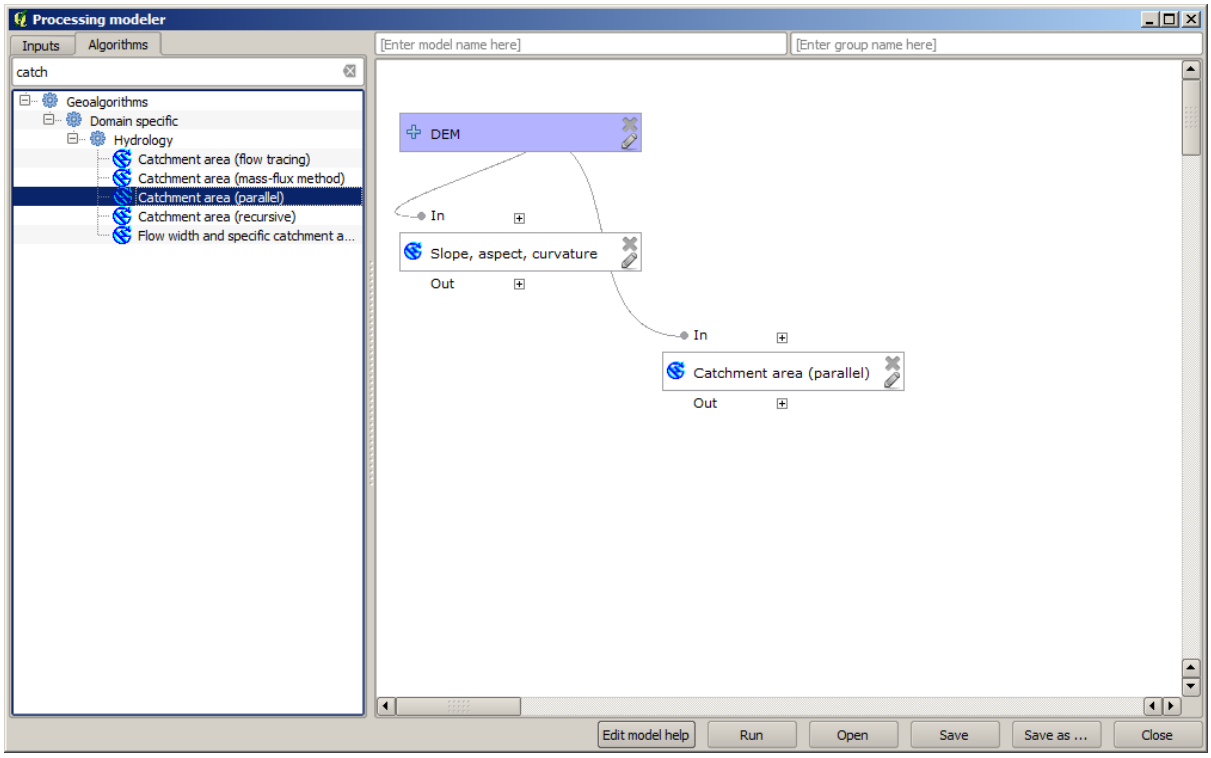
8. There is not much to select in this first dialog, since we do not have but just one layer in our model (The DEM input that we created). Actually, the default configuration of the dialog is the correct one in this case, so you just have to press *OK*. This is what you will now have in the modeler canvas.



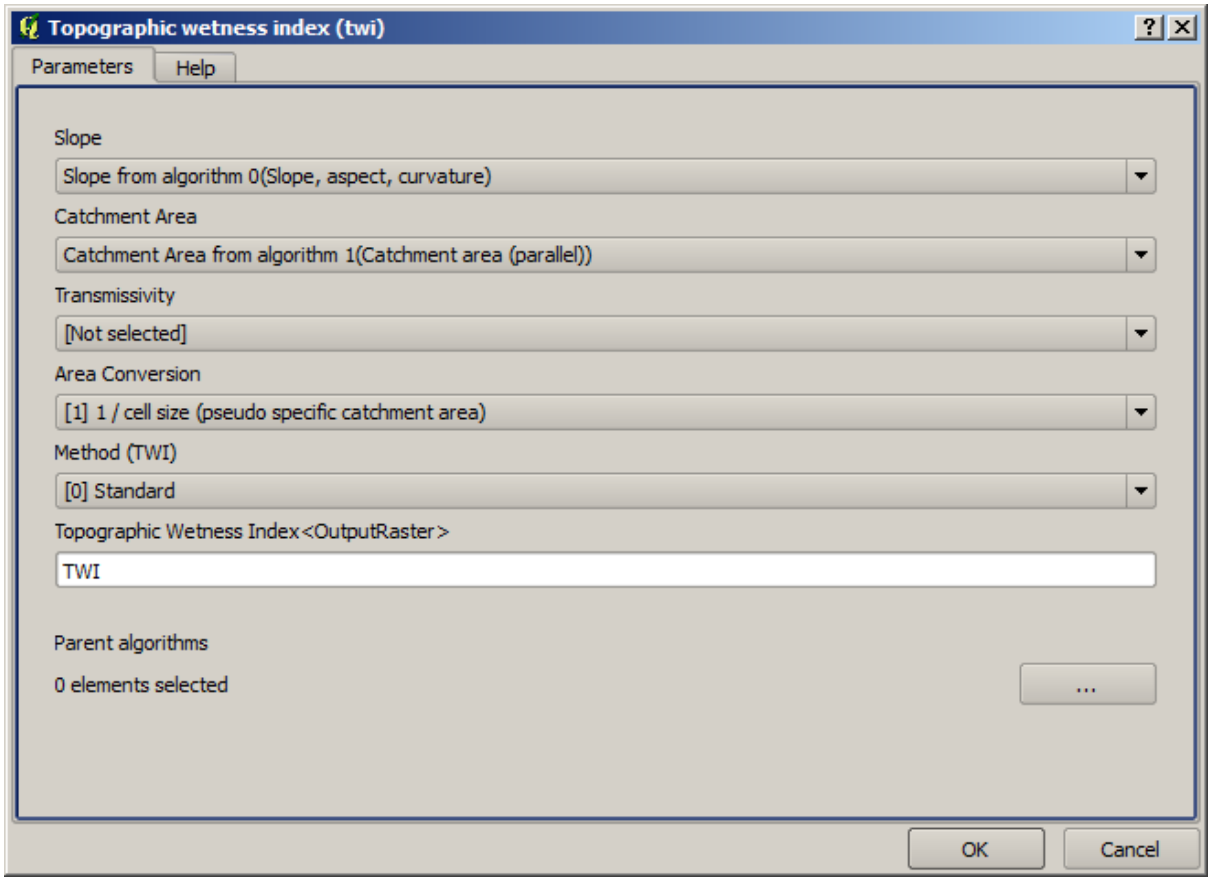
9. The second algorithm we have to add to our model is the catchment area algorithm. We will use the algorithm named *Catchment area (Paralell)*. We will use the DEM layer again as input, and none of the ouputs it produces are final, so here is how you have to fill the corresponding dialog.



Now your model should look like this.



10. The last step is to add the *Topographic wetness index* algorithm, with the following configuration.

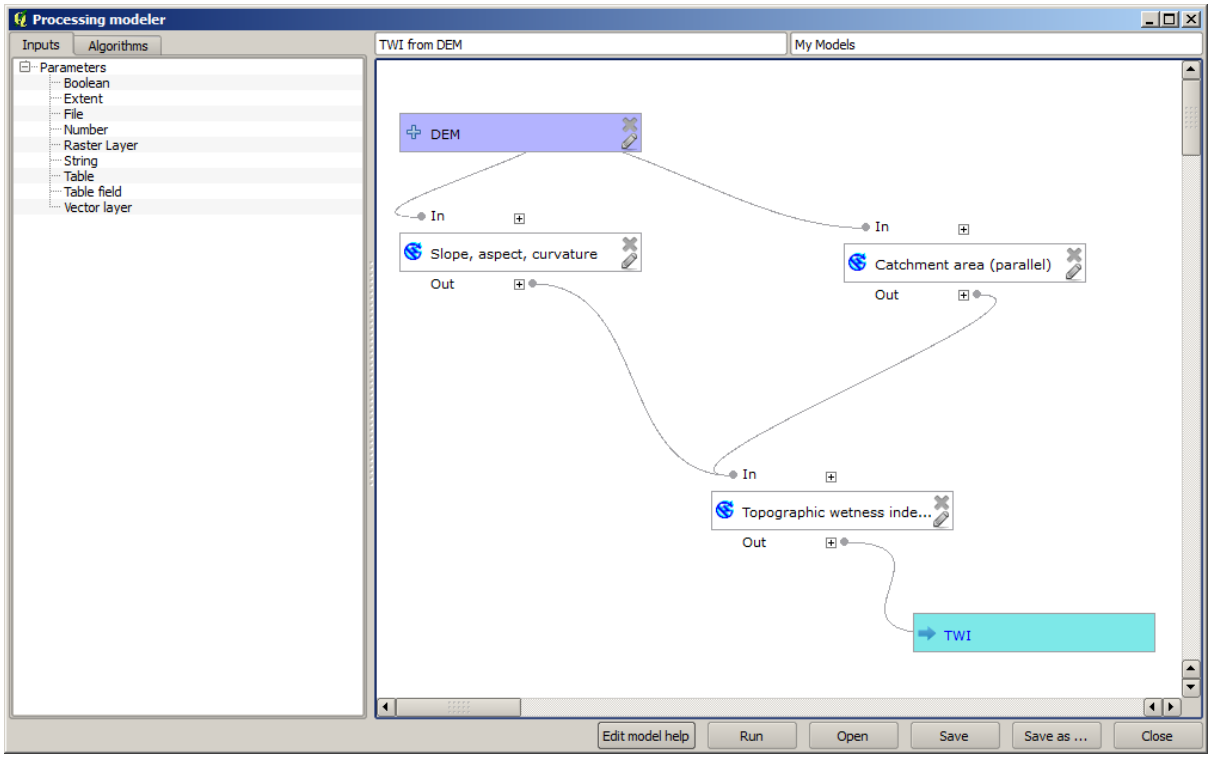


In this case, we will not be using the DEM as input, but instead, we will use the slope and catchment area layers that are calculated by the algorithms that we previously added. As you add new algorithms, the outputs they

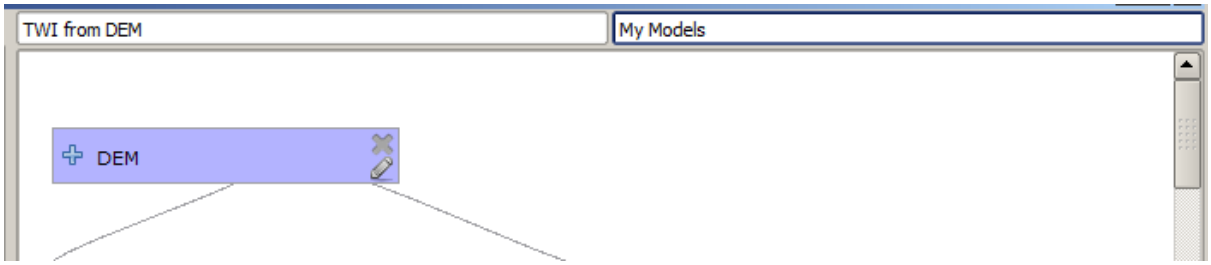
produce become available for other algorithms, and using them you link the algorithms, creating the workflow.

11. In this case, the output `TWI` layer is a final layer, so we have to indicate so. In the corresponding textbox, enter the name that you want to be shown for this output.

    Now our model is finished and it should look like this.

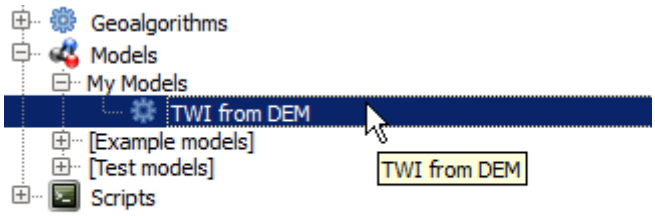


12. Enter a name and a group name in the upper part of the model window.

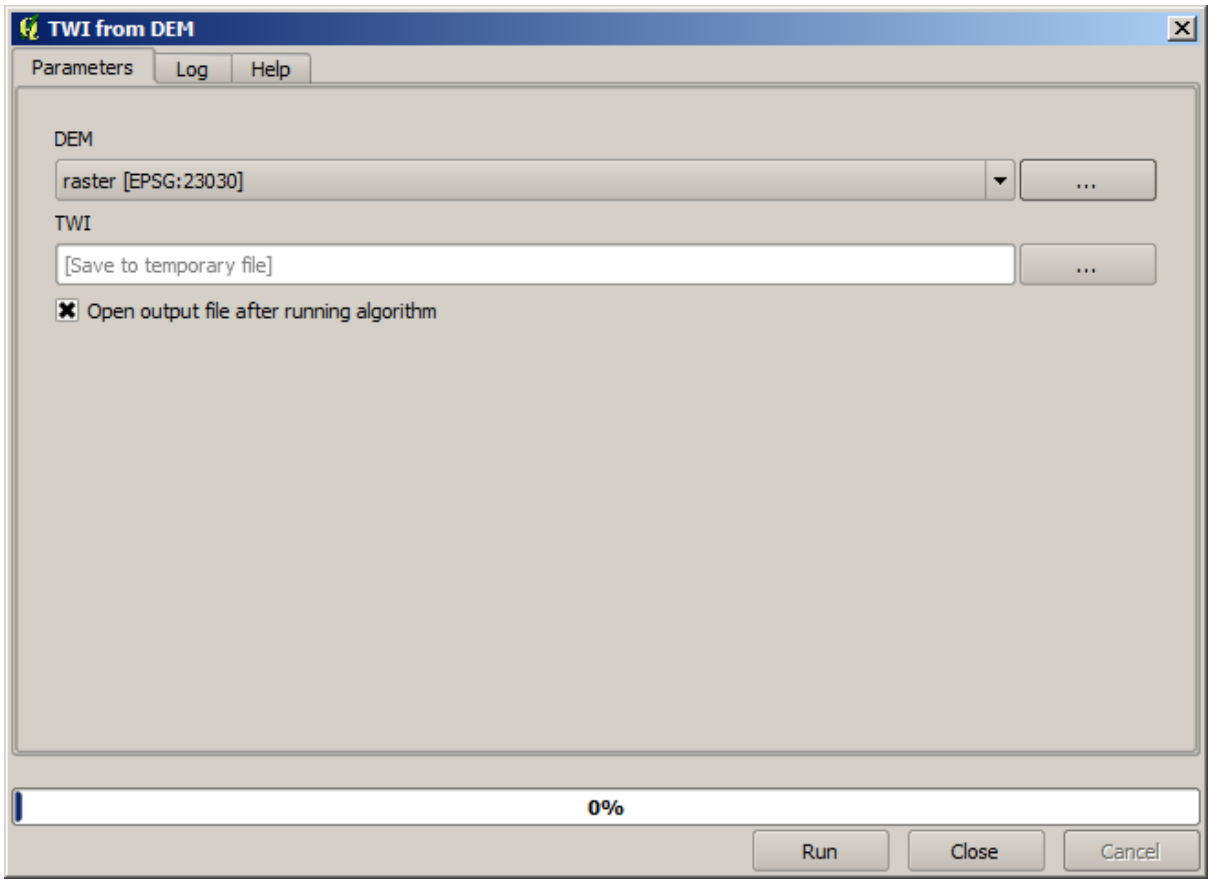


13. Save it clicking on the *Save* button. You can save it anywhere you want and open it later, but if you save it in the models folder (which is the folder that you will see when the save file dialog appears), your model will also be available in the toolbox as well. So stay on that folder and save the model with the filename that you prefer.

14. Now close the modeler dialog and go to the toolbox. In the *Models* entry you will find your model.



15. You can run it just like any normal algorithm, double-clicking on it.



As you can see, the parameters dialog, contain the input that you added to the model, along with the outputs that you set as final when adding the corresponding algorithms.

16. Run it using the DEM as input and you will get the TWI layer in just one single step.

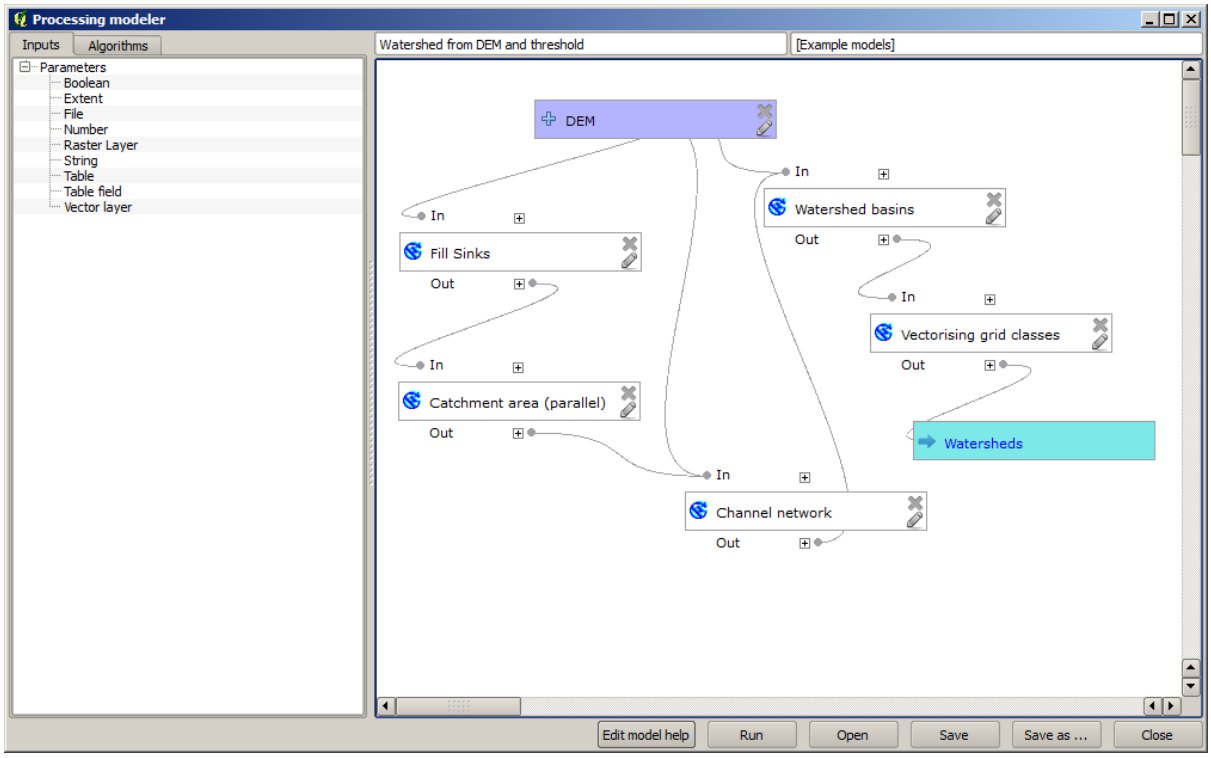
17.18 Modelos mais complexos

Nota: In this lesson we will work with a more complex model in the model designer.

The first model that we created in the previous chapter was a very simple one, with just one input and three algorithms. More complex models can be created, with different types of inputs and containing more steps. For this chapter we will work with a model that creates a vector layer with watersheds, based on a DEM and a threshold value. That will be very useful for calculating several vector layers corresponding to different thresholds, without having to repeat each single step each time.

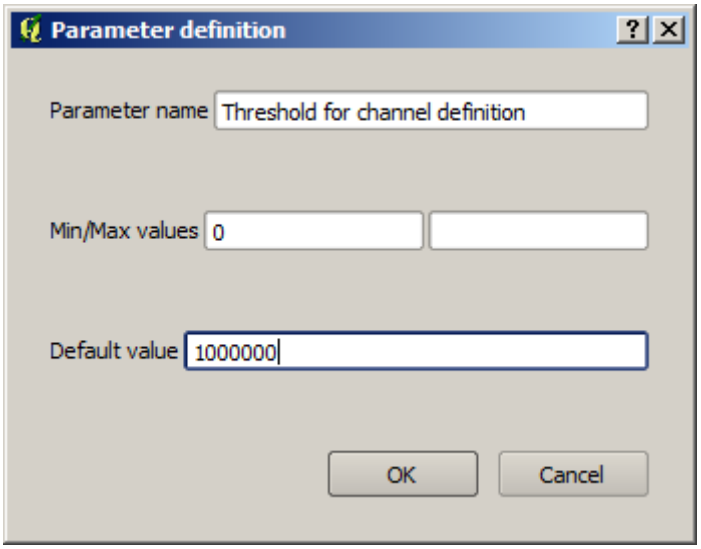
This lesson does not contain instructions about how to create your model. You already know the necessary steps (we saw them in a *previous lesson*) and you have already seen the basic ideas about the modeler, so you should try it yourself. Spend a few minutes trying to create your model, and don't worry about making mistakes. Remember: first add the inputs and then add the algorithms that use them to create the workflow.

Nota: In case you could not create the full model yourself and you need some extra help, the data folder corresponding to this lesson contains an "almost" finished version of it. Open the modeler and then open the model file that you will find in the data folder. You should see something like this.

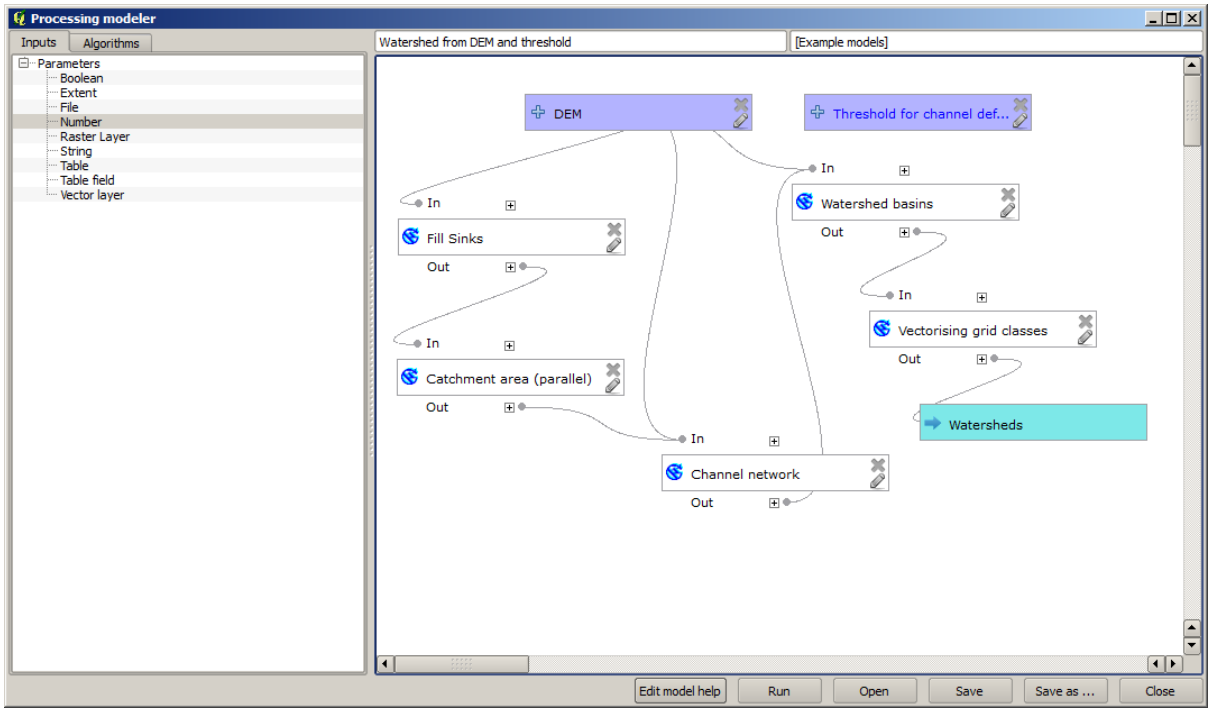


This model contains all the steps needed to complete the calculation, but it just has one input: the DEM. That means that the threshold for channel definition uses a fixed value, which makes the model not as useful as it could be. That is not a problem, since we can edit the model, and that is exactly what we will do.

1. First, let's add a numerical input. That will ask the user for a numerical input that we can use when such a value is needed in any of the algorithms included in our model.
2. Click on the *Number* entry in the *Inputs* tree, and you will see the corresponding dialog.
3. Fill it with the following values.
   - *Parameter name*: `Threshold for channel definition`
   - *Default value*: `1,000,000`

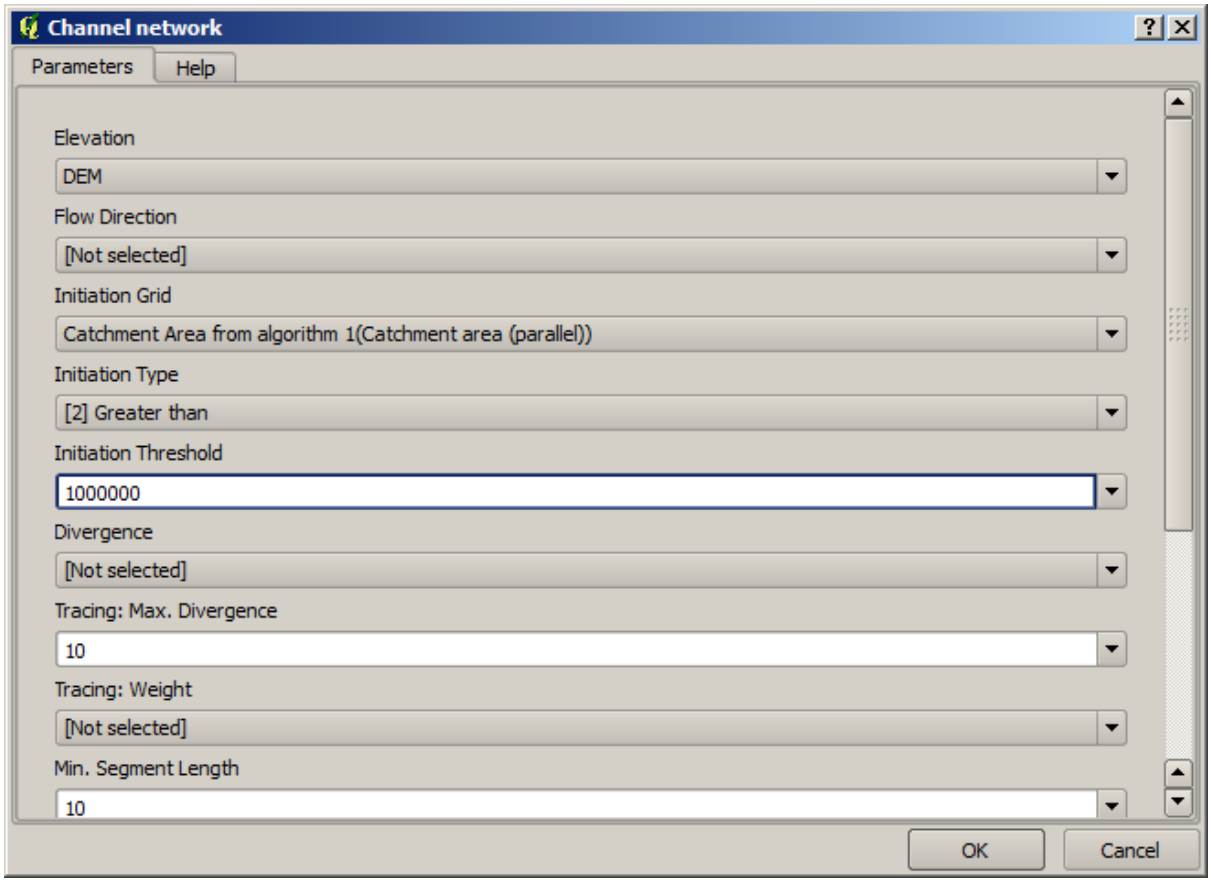


Now your model should look like this.



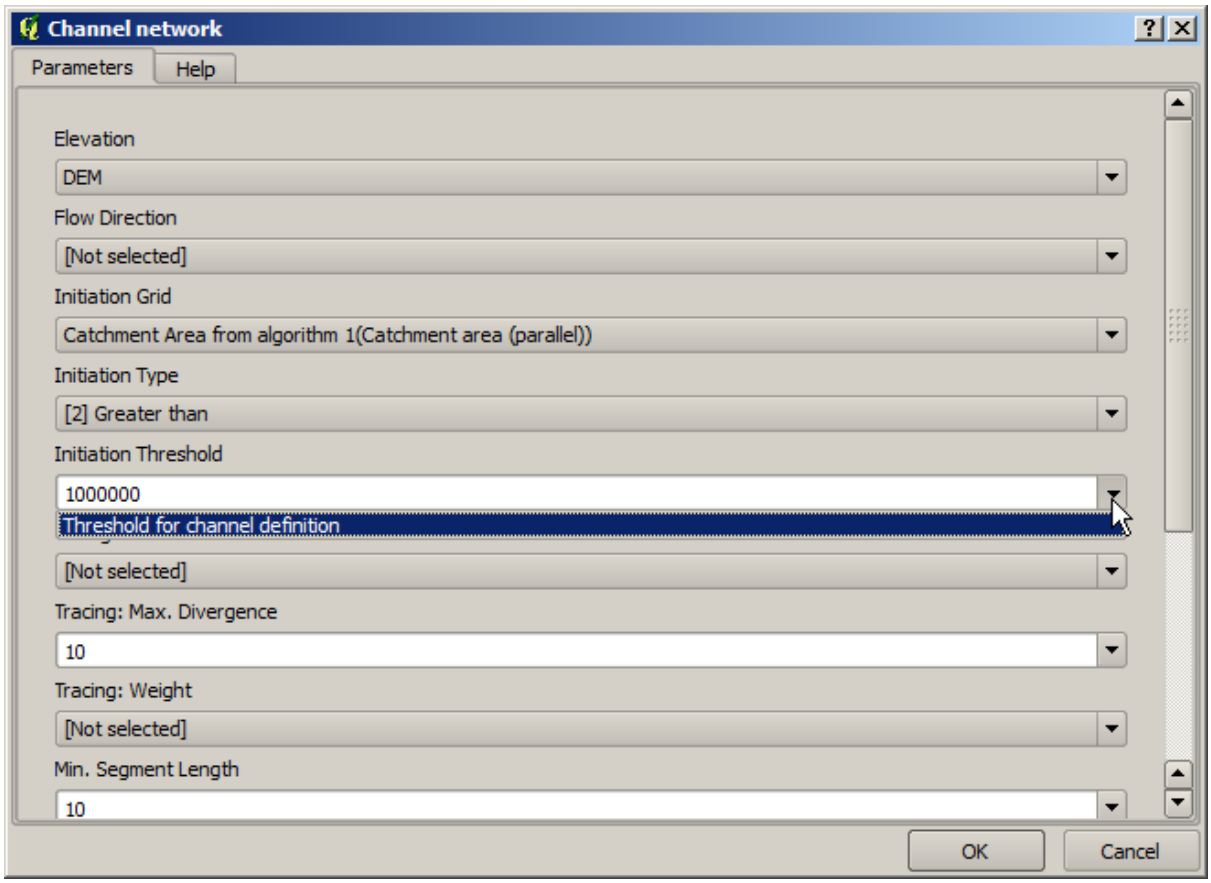
The input that we have just added is not used, so the model hasn't actually changed. We have to link that input to the algorithm that uses it, in this case the *Channel network* one. To edit an algorithm that already exists in the modeler, just click on the pen icon on the corresponding box in the canvas.

4. Click on the *Channel network* algorithm and you will see something like this.



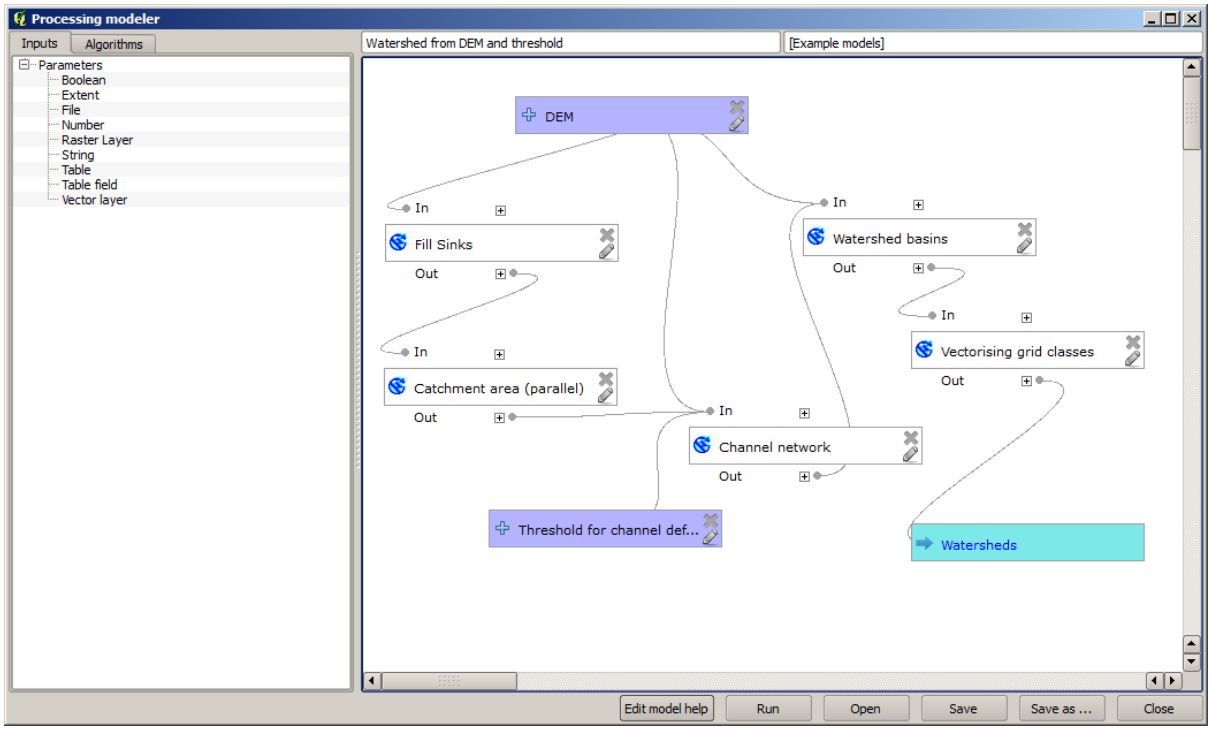
The dialog is filled with the current values used by the algorithm. You can see that the *Initiation threshold* parameter has a fixed value of 1,000,000 (this is also the default value of the algorithm, but any other value could be put in there). However, you might notice that the parameter is not entered in a common text box, but in a drop-down menu.

5. Unfold the threshold parameter menu and you will see something like this.



The input that we added is there and we can select it. Whenever an algorithm in a model requires a numerical value, you can hardcode it and directly type it, or you can use any of the available inputs and values (remember that some algorithms generate single numerical values. We will see more about this soon). In the case of a string parameter, you will also see string inputs and you will be able to select one of them or type the desired fixed value.

6. Select the `Threshold for channel definition` input in the *Initiation threshold* parameter.
7. Click on *OK* to apply the changes to your model. Now the design of the model should look like this.



8. The model is now complete. Run it using the DEM that we have used in previous lessons, and with different threshold values.

   Below you have a sample of the result obtained for different values. You can compare with the result for the default value, which is the one we obtained in the *hydrological analysis lesson*.

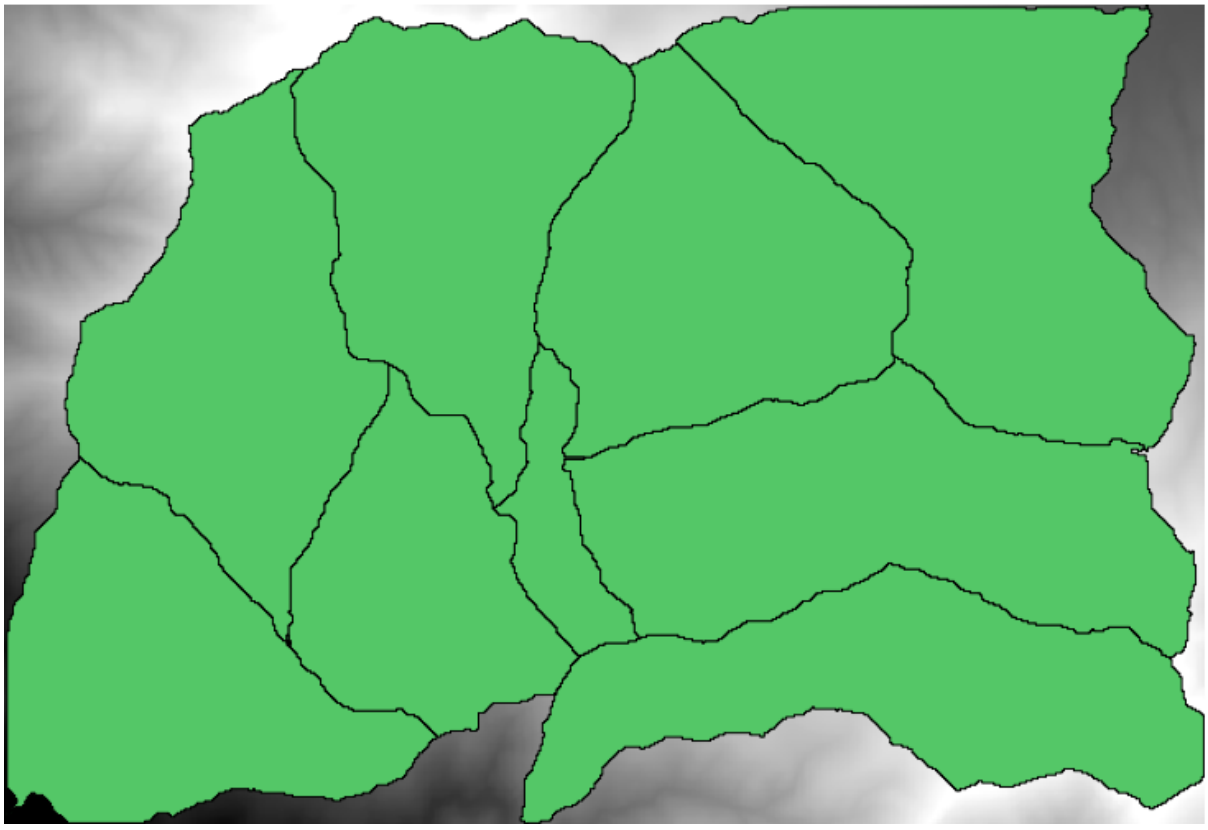


Figura17.1: Threshold = 100,000

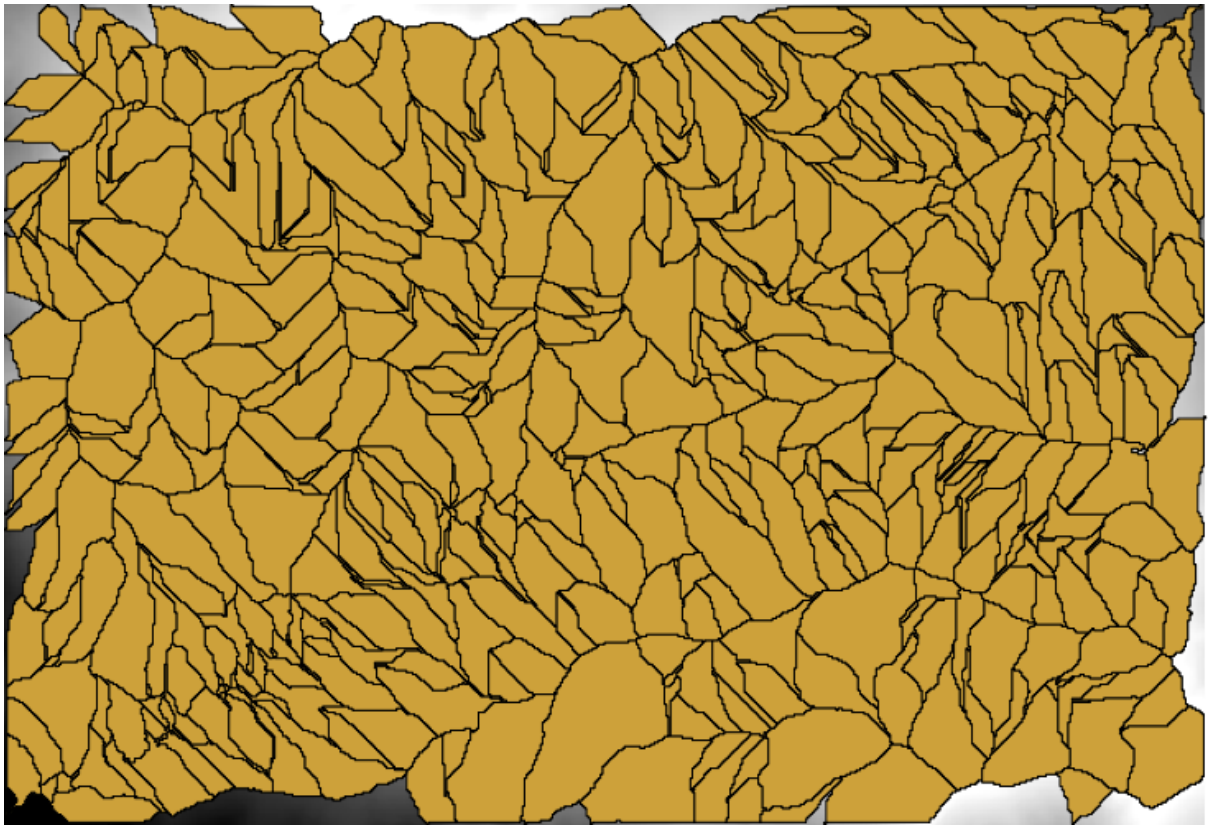


Figura17.2: Threshold = 1,0000,000

17.19 Numeric calculations in the modeler

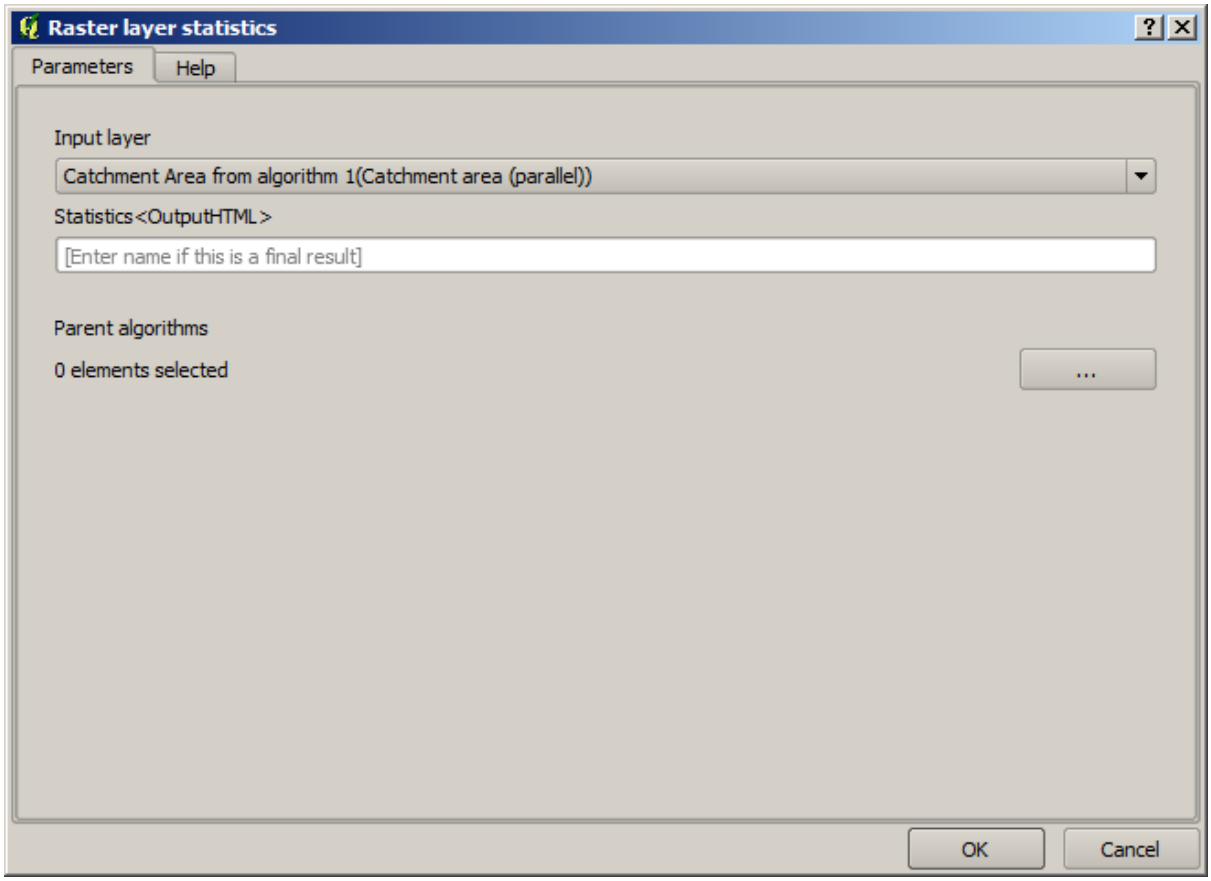
Aviso: Beware, this chapter is not well tested, please report any issue; images are missing

Nota: In this lesson we will see how to use numeric outputs in the modeler

For this lesson, we are going to modify the hydrological model that we created in the last chapter (open it in the modeler before starting), so we can automate the calculation of a valid threshold value and we do not have to ask the user to enter it. Since that value refers to the variable in the threshold raster layer, we will extract it from that layer, based on some simple statistical analysis.

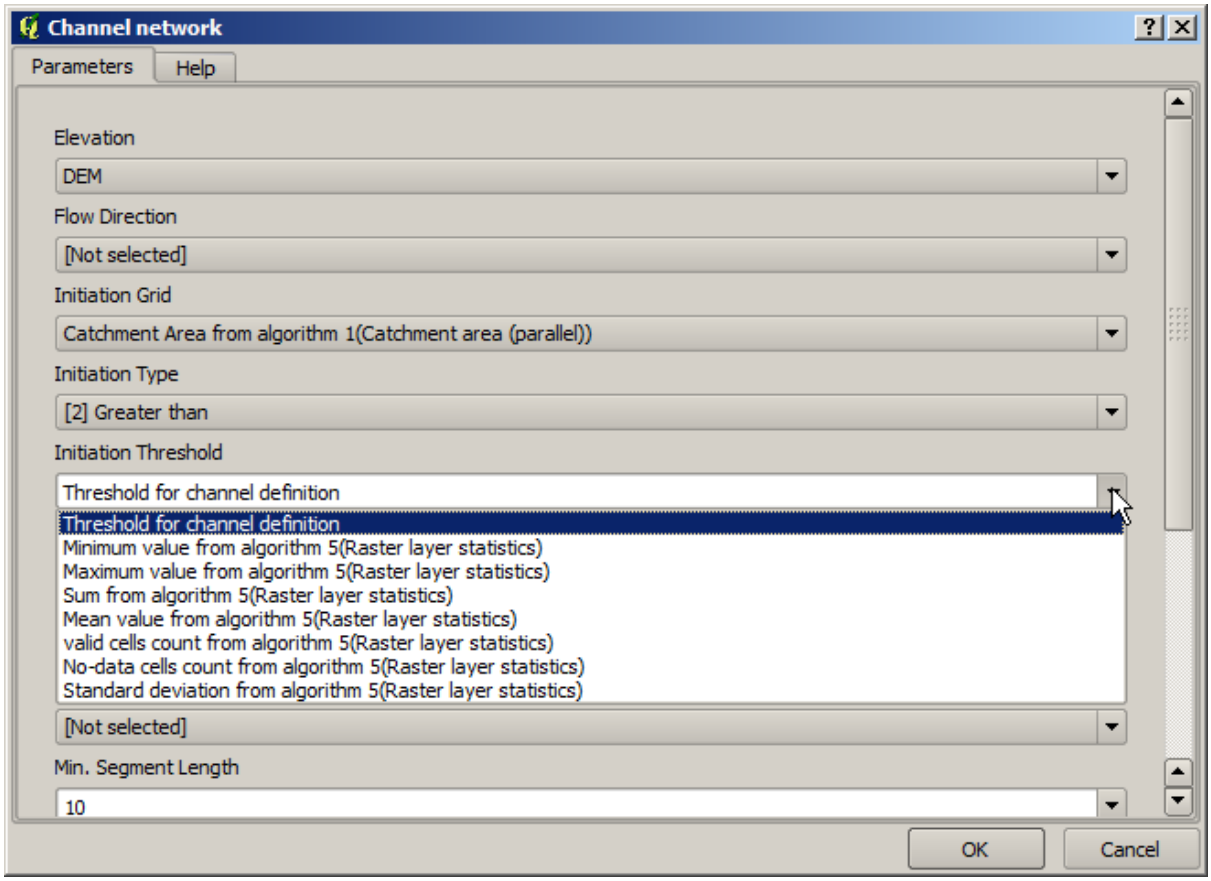
Starting with the aforementioned model, let's do the following modifications:

First, calculate statistics of the flow accumulation layer using the *Raster layer statistics* algorithm.



This will generate a set of statistical values that will now be available for all numeric fields in other algorithms.

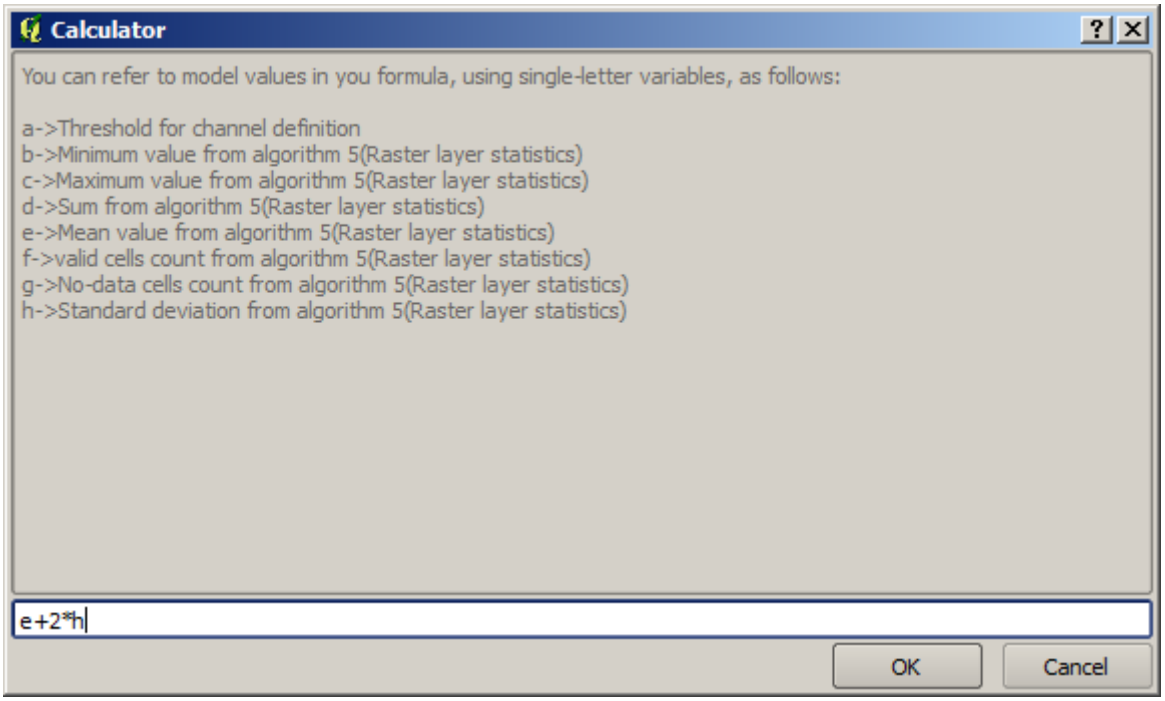
If you edit the *Channel network* algorithm, as we did in the last lesson, you will see now that you have other options apart from the numeric input that you added.



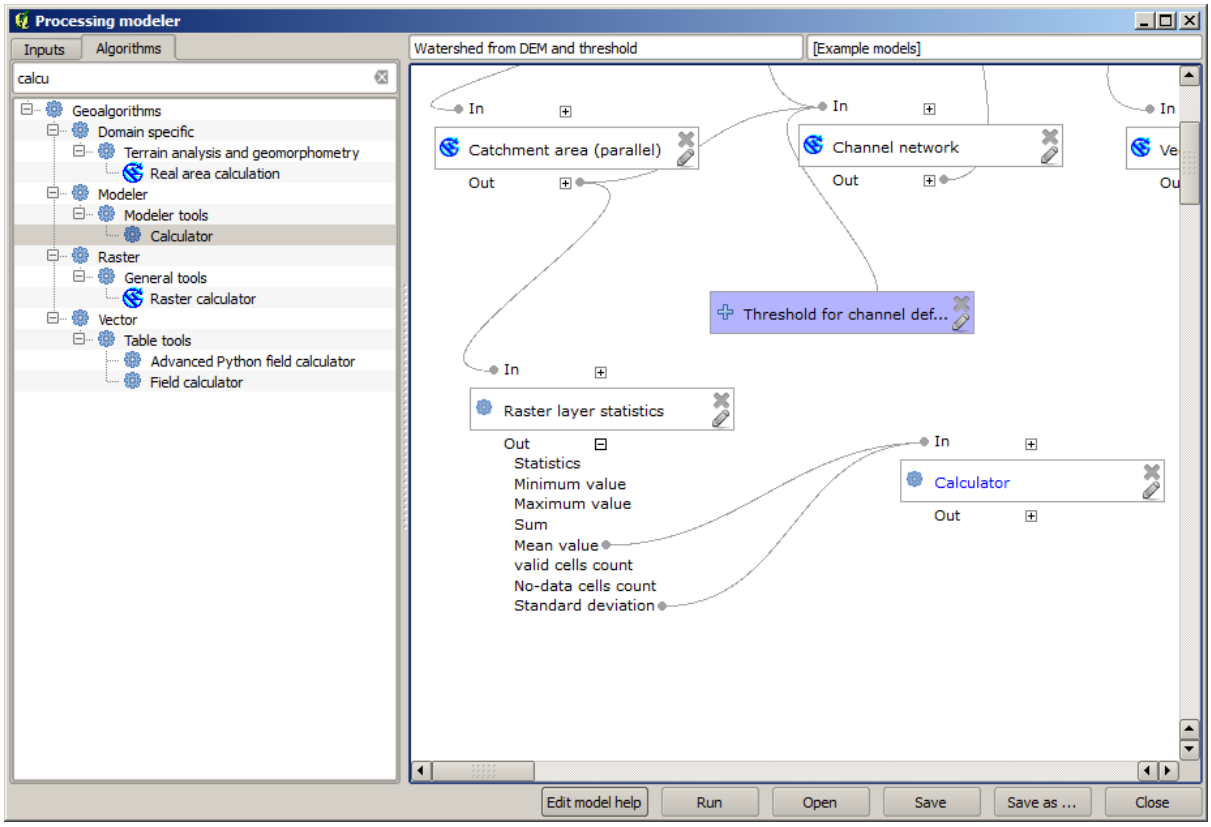
However, none of this values is suitable for being used as a valid threshold, since they will result in channel networks that will not be very realistic. We can, instead, derive some new parameter based on them, to get a better result. For instance, we can use the mean plus 2 times the standard deviation.

To add that arithmetical operation, we can use the calculator that you will find in the *Geoalgorithms/modeler/modeler-tools* group. This group contains algorithms that are not very useful outside of the modeler, but that provide some useful functionality when creating a model.

The parameters dialog of the calculator algorithm looks like this:



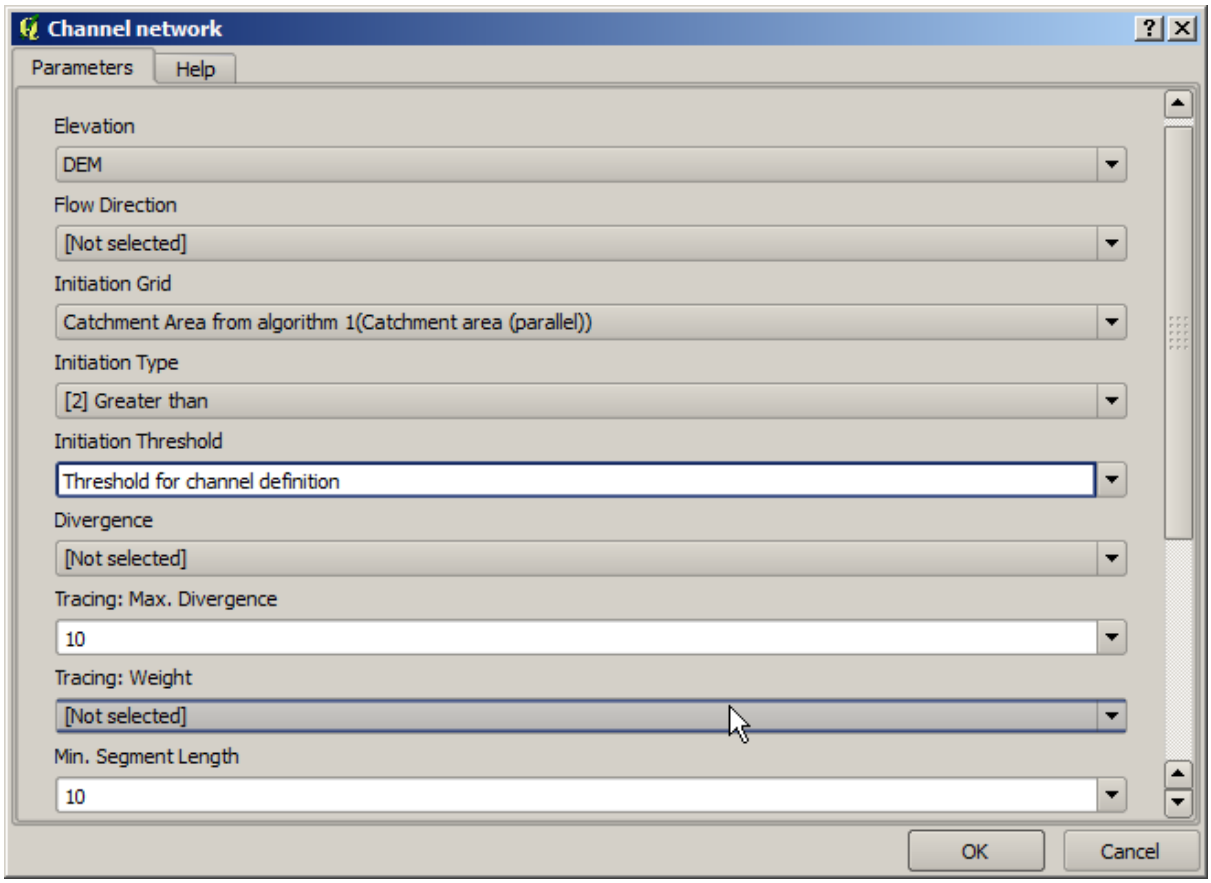
As you can see, the dialog is different to the other ones we have seen, but you have in there the same variables that were available in the *Threshold* field in the *Channel network* algorithm. Enter the above formula and click on *OK* to add the algorithm.



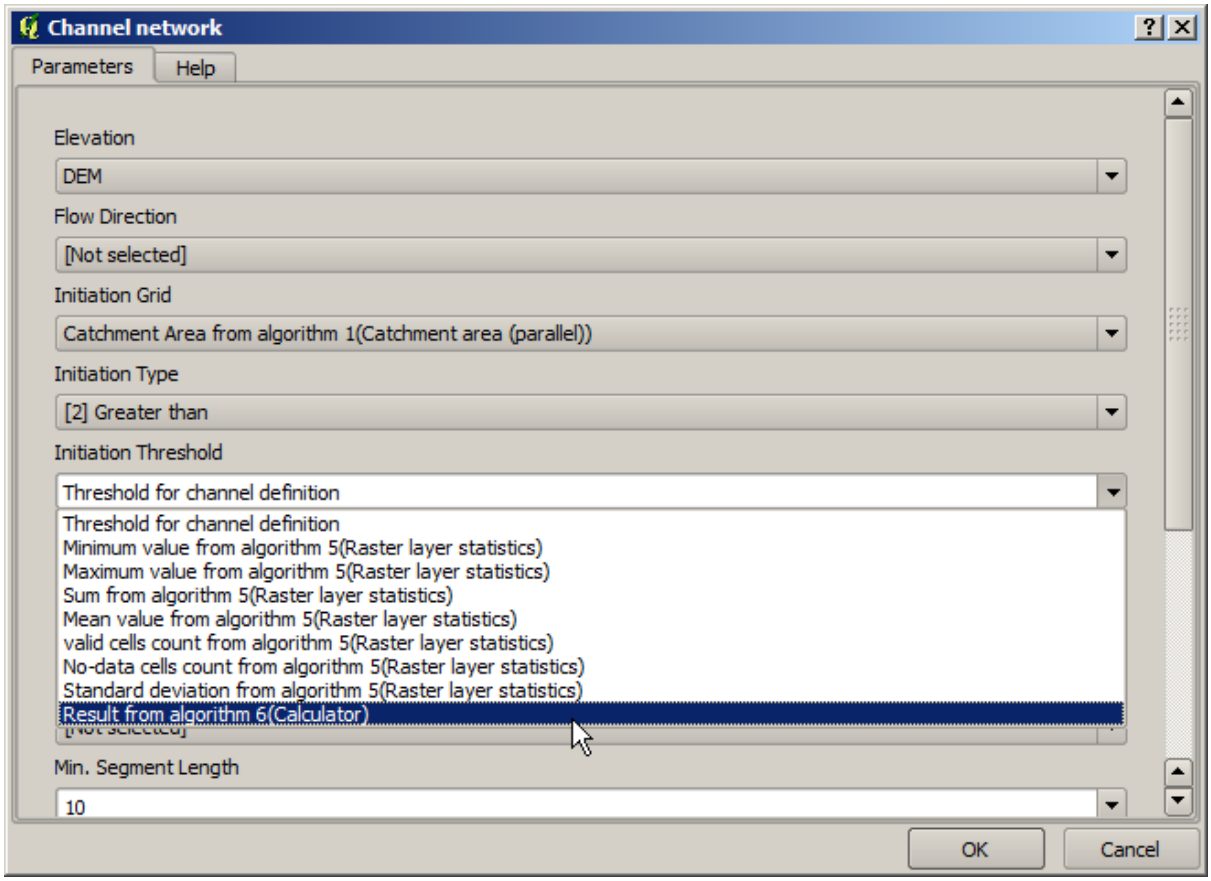
If you expand the outputs entry, as shown above, you will see that the model is connected to two of the values, namely the mean and the standard deviation, which are the ones that we have used in the formula.

Adding this new algorithm will add a new numeric value. If you go again to the *Channel network* algorithm, you can

now select that value in the *Threshold* parameter.



Click on *OK* and your model should look like this.



We are not using the numeric input that we added to the model, so it can be removed. Right–click on it and select *Remove*

Aviso: todo: Add image

Our new model is now finished.

17.20 A model within a model

Aviso: Beware, this chapter is not well tested, please report any issue; images are missing

Nota: In this lesson we will see how to use a model within a bigger model.

We have already created a few models, and in this lesson we are going to see how we can combine them on a single bigger one. A model behaves like any other algorithm, which means that you can add a model that you have already created as part of another one that you create after that.

In this case, we are going to expand our hydrological model, by adding the mean TWI value in each of the basins that it generates as result. To do that, we need to calculate the TWI, and to compute the statistics. Since we have already created a model to calculate TWI from a DEM, it is a good idea to reuse that model instead of adding the algorithms it contains individually.

Let's start with the model we used as starting point for the last lesson.

Aviso: todo: Add image

First, we will add the TWI model. For it to be available, it should have been saved on the models folder, since otherwise it will not be shown in the toolbox or the algorithms list in the modeler. Make sure you have it available.

Add it to the current model and use the input DEM as its input. The output is a temporary one, since we just want the TWI layer to compute the statistics. The only output of this model we are creating will still be the vector layer with the watersheds.

Here is the corresponding parameters dialog:

Aviso: todo: Add image

Now we have a TWI layer that we can use along with the watersheds vector layer, to generate a new one which contains the values of the TWI corresponding to each watershed.

This calculation is done using the *Grid statistics in polygons* algorithm. Use the layers mentioned above as input, to create the final result.

Aviso: todo: Add image

The output of the *Vectorize grid classes* algorithm was originally our final output, but now we just want it as an intermediate result. To change that, we have to edit the algorithm. Just double–click on it to see its parameters dialog, and delete the name of the output. That will make it a temporary output, as it is by default.

Aviso: todo: Add image

This is how the final model should look like:

Aviso: todo: Add image

As you see, using a model in another model is nothing special, and you can add it just like you add another algorithm, as long as the model is saved in the models folder and is available in the toolbox.

17.21 Using modeler-only tools for creating a model

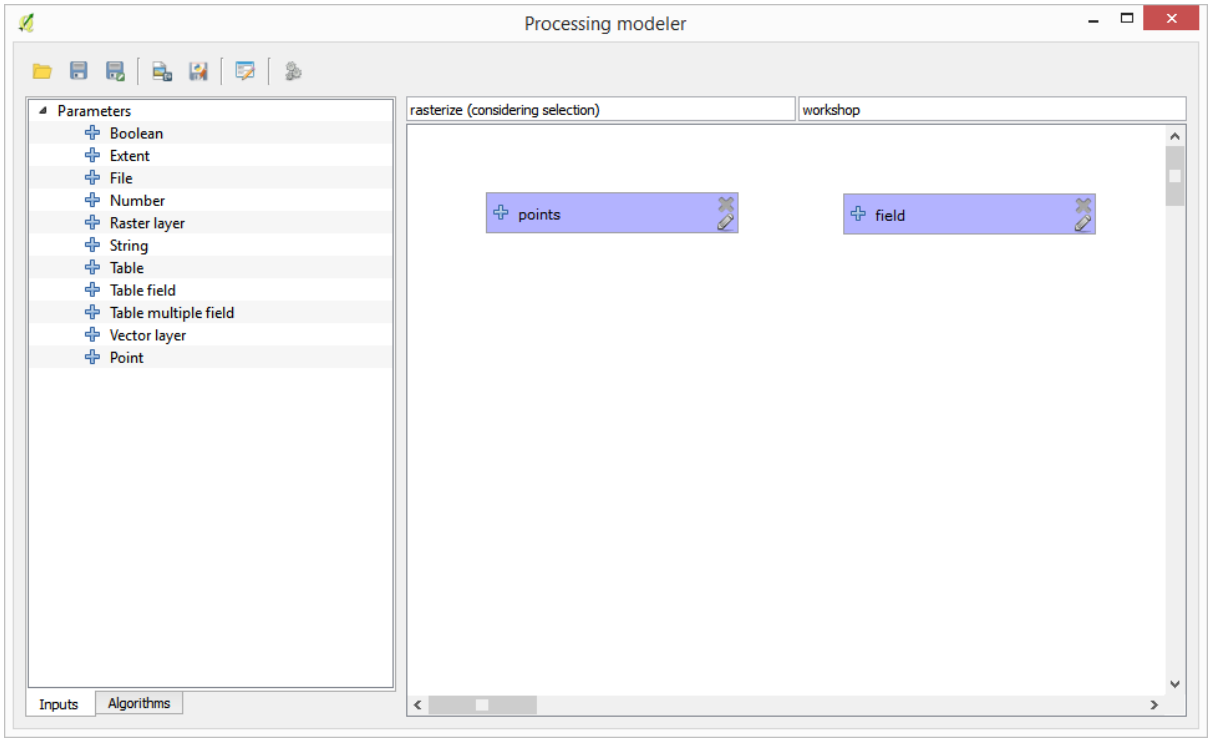
Nota: This lesson shows how to use some algorithms that are only available in the modeler, to provide additional functionality to models.

The goal of this lesson is to use the modeler to create an interpolation algorithm that takes into account the current selection, not just to use only selected features, but to use the extent of that selection to create the interpolated raster layer.

The interpolation process involves two steps, as it has been already explained in previous lessons: rasterizing the points layer and fill the no-data values that appear in the rasterized layer. In case the points layer has a selection, only selected points will be used, but if the output extent is set to be automatically adjusted, the full extent of the layer will

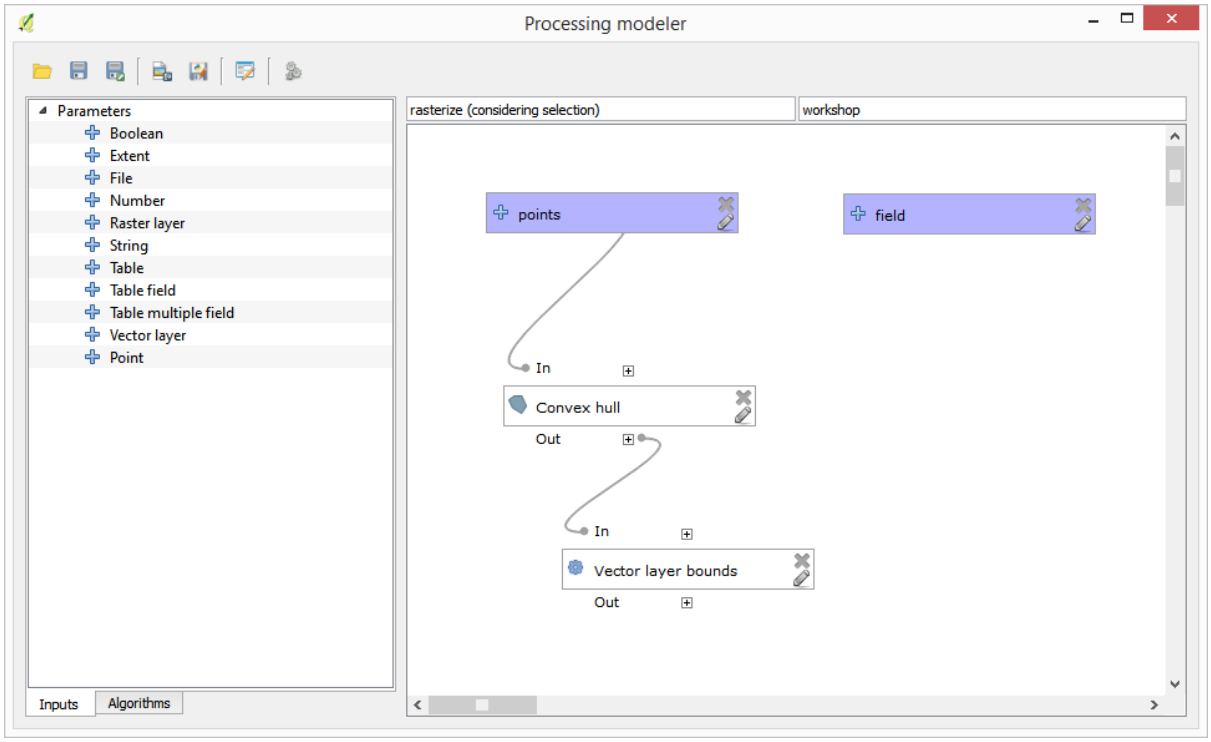
be used. That is, the extent of the layer is always considered to be the full extent of all features, not the one computed from just the selected ones. We will try to fix that by using some additional tools into our model.

Open the modeler and start the model by adding the required inputs. In this case we need a vector layer (restricted to points) and an attribute from it, with the values that we will use for rasterizing.

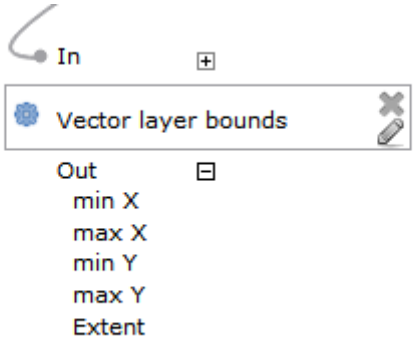


The next step is to compute the extent of the selected features. That's where we can use the model-only tool called *Vector layer bounds*. First, we will have to create a layer that has the extent of those selected features. Then, we can use this tool on that layer.

An easy way of creating a layer with the extent of the selected features is to compute a convex hull of the input points layer. It will use only the selected point, so the convex hull will have the same bounding box as the selection. Then we can add the *Vector layer bounds* algorithm, and use the convex hull layer as input. It should look this in the modeler canvas:

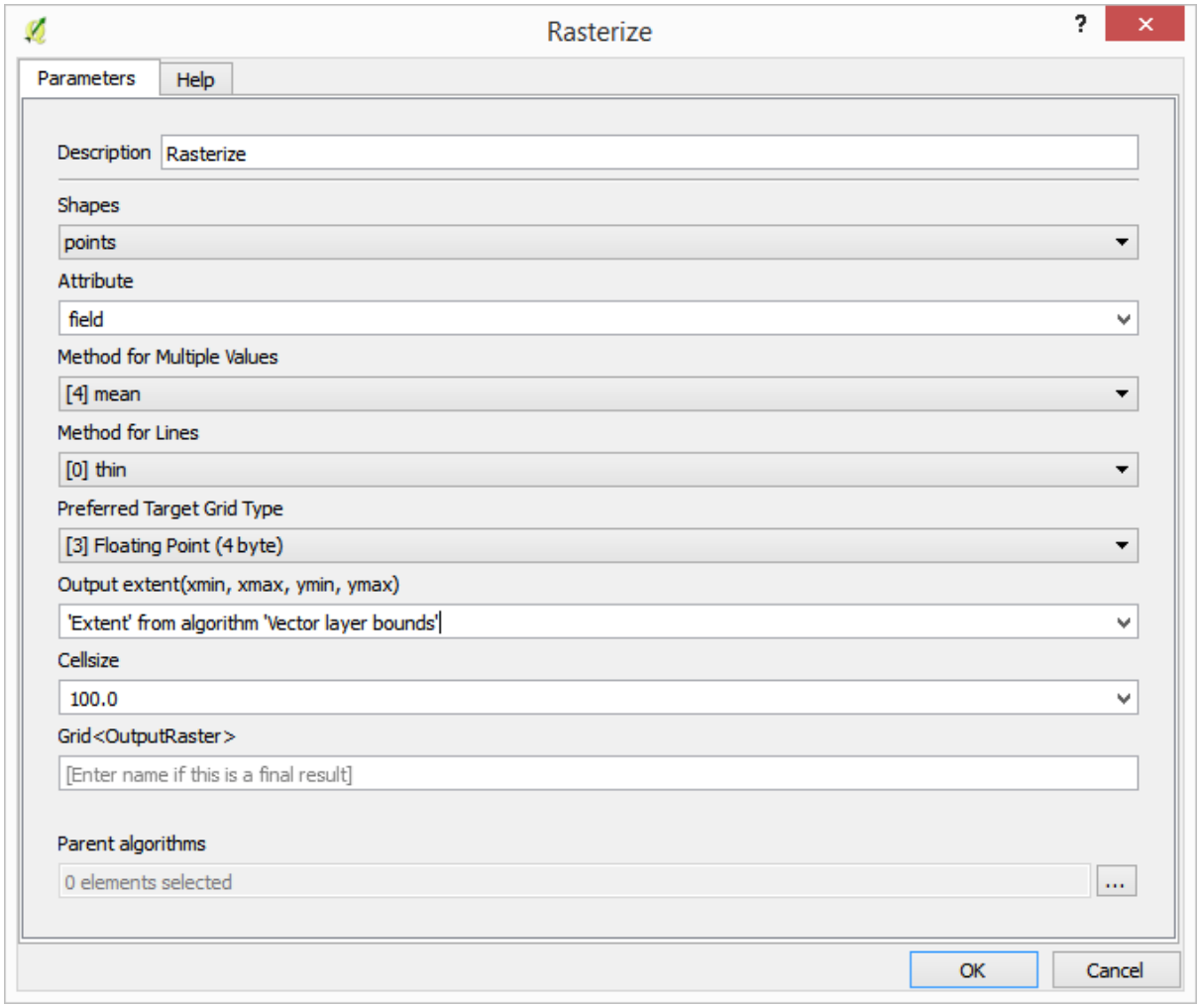


The result from the *Vector layer bounds* is a set of four numeric values and a extent object. We will use both the numeric outputs and the extent for this exercise.

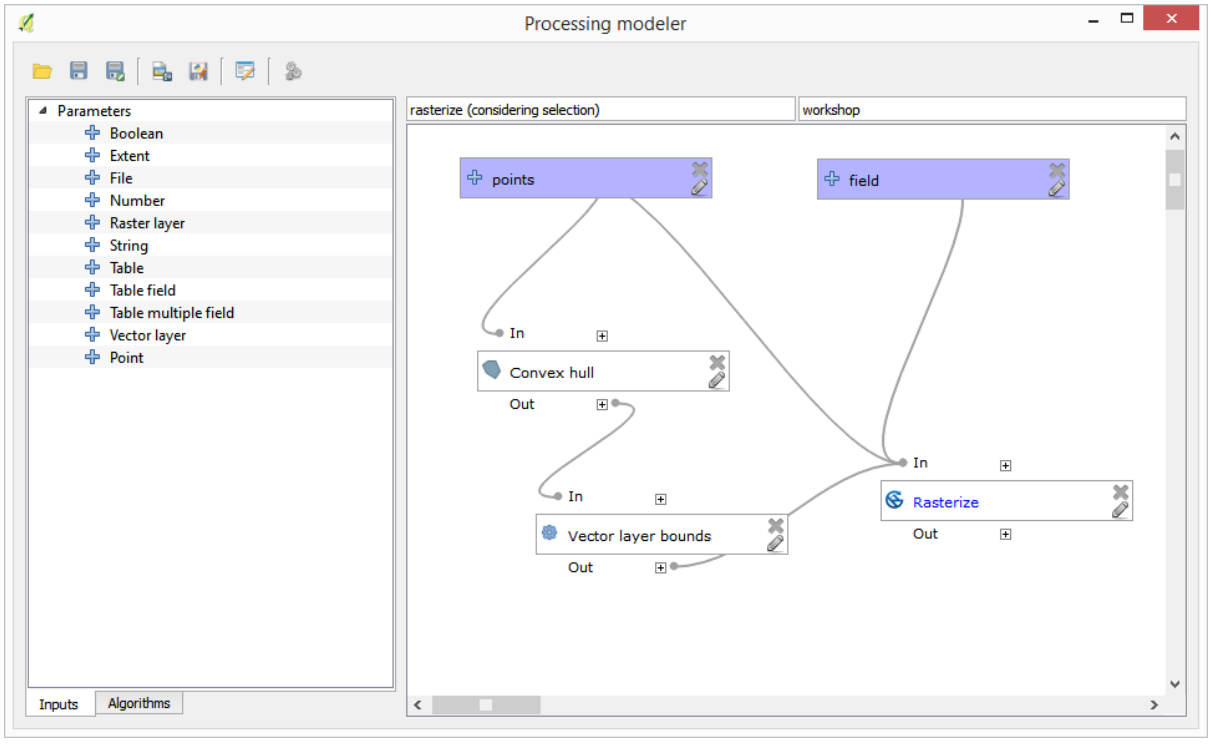


We can now add the algorithm that rasterizes the vector layer, using the extent from the *Vector layer bounds* algorithm as input.

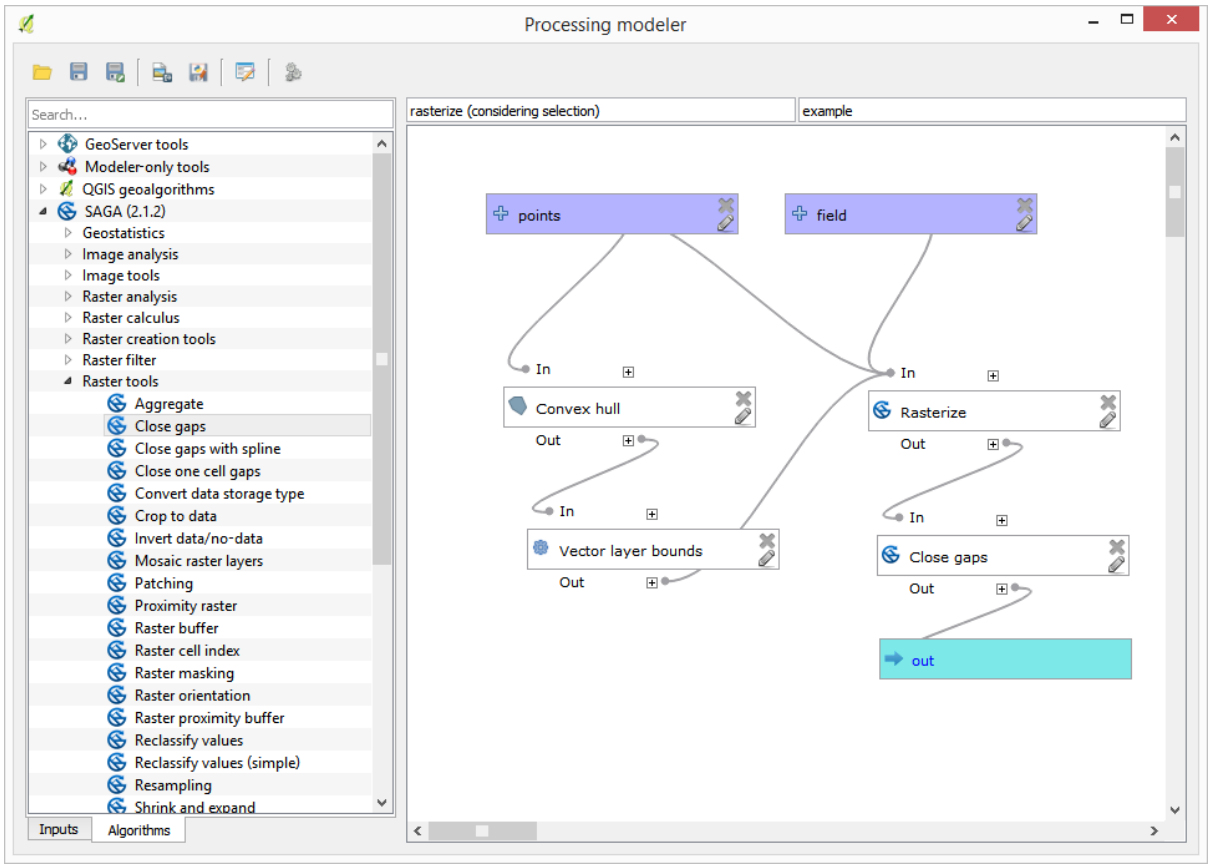
Fill the parameters of the algorithm as shown next:



The canvas should now look like.



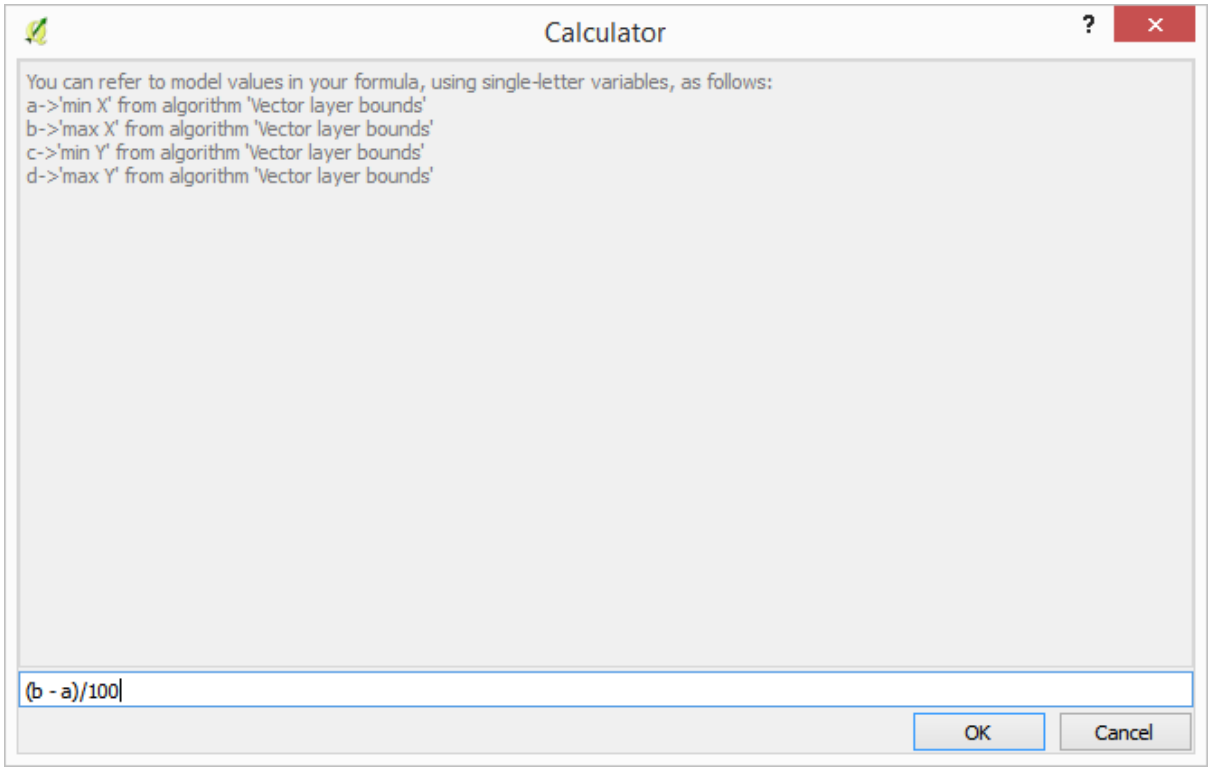
Finally, fill the no-data values of the raster layer using the *Close gaps* algorithm.



The algorithm is now ready to be saved and added to the toolbox. You can run it and it will generate a raster layer from interpolating the selected points in the input layer, and the layer will have the same extent as the selection.

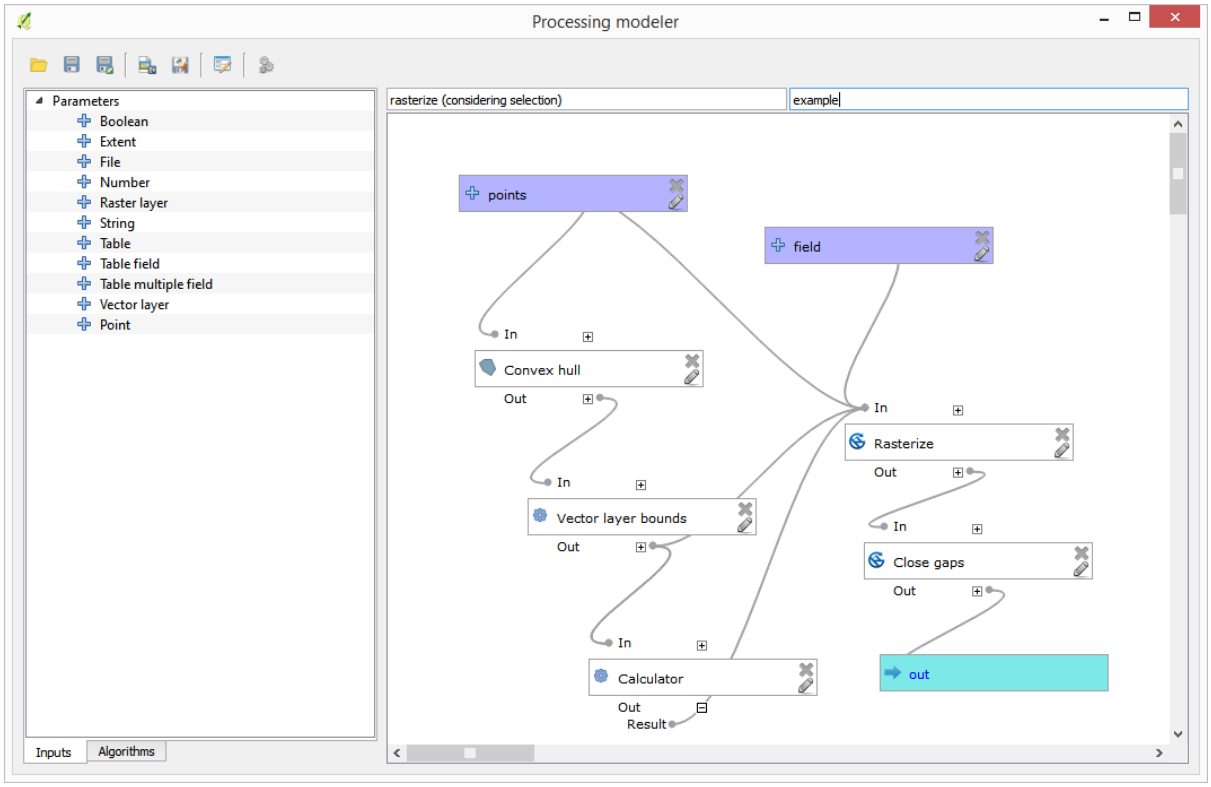
Here's an improvement to the algorithm. We have used a harcoded value for the cellsize when rasterizing. This value is fine for our test input layer, but might not be for other cases. We could add a new parameter, so the user enters the desired value, but a much better approach would be to have that value automatically computed.

We can use the modeler-only calculator, and compute that value from the extent coordinates. For instance, to create a layer with a fixed width of 100 pixels, we can use the following formula in the calculator.



Now we have to edit the rasterize algorithm, so it uses the output of the calculator instead of the hardcoded value.

The final algorithm should look like this:

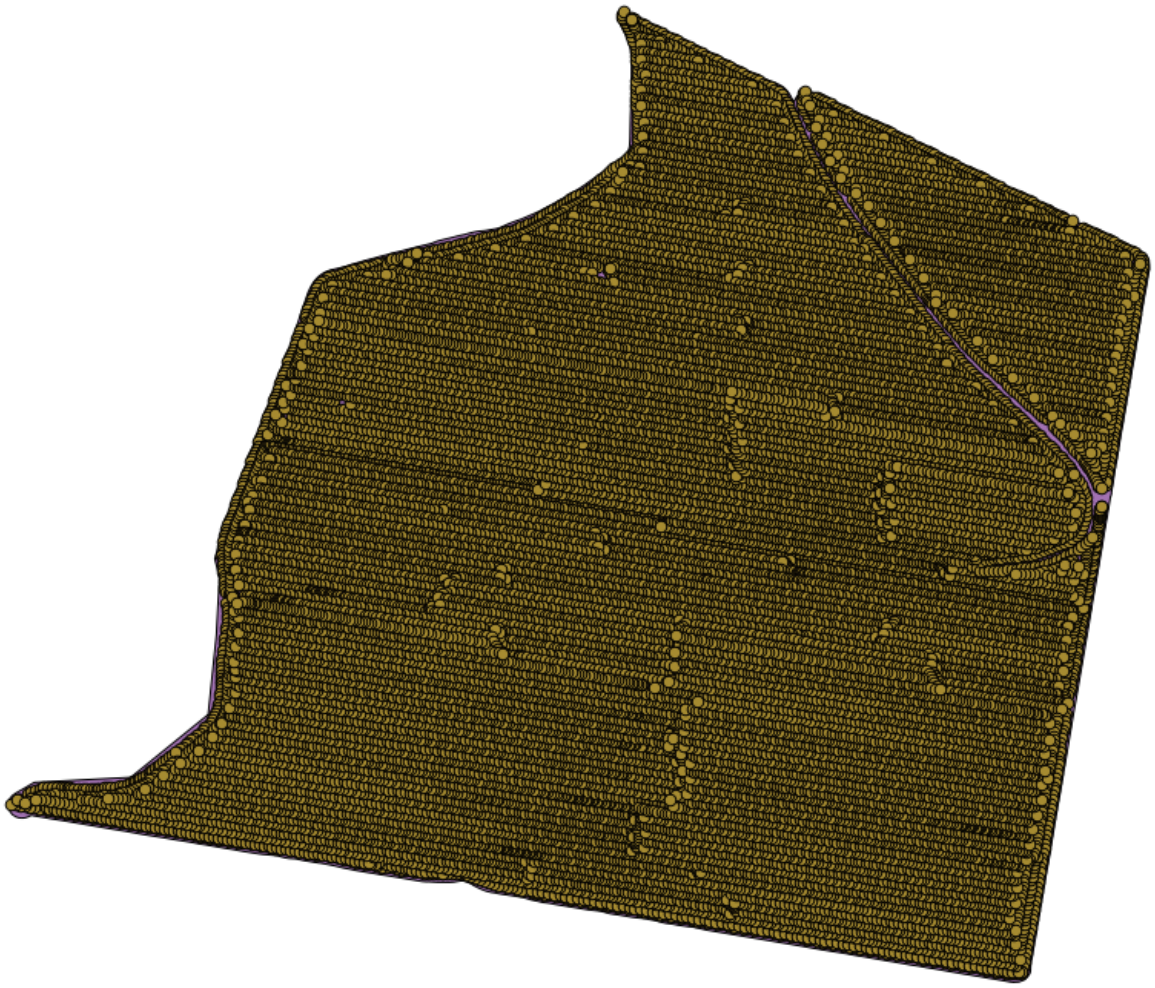


17.22 Interpolação

Nota: This chapter shows how to interpolate point data, and will show you another real example of performing spatial analysis

In this lesson, we are going to interpolate points data to obtain a raster layer. Before doing it, we will have to do some data preparation, and after interpolating we will add some extra processing to modify the resulting layer, so we will have a complete analysis routine.

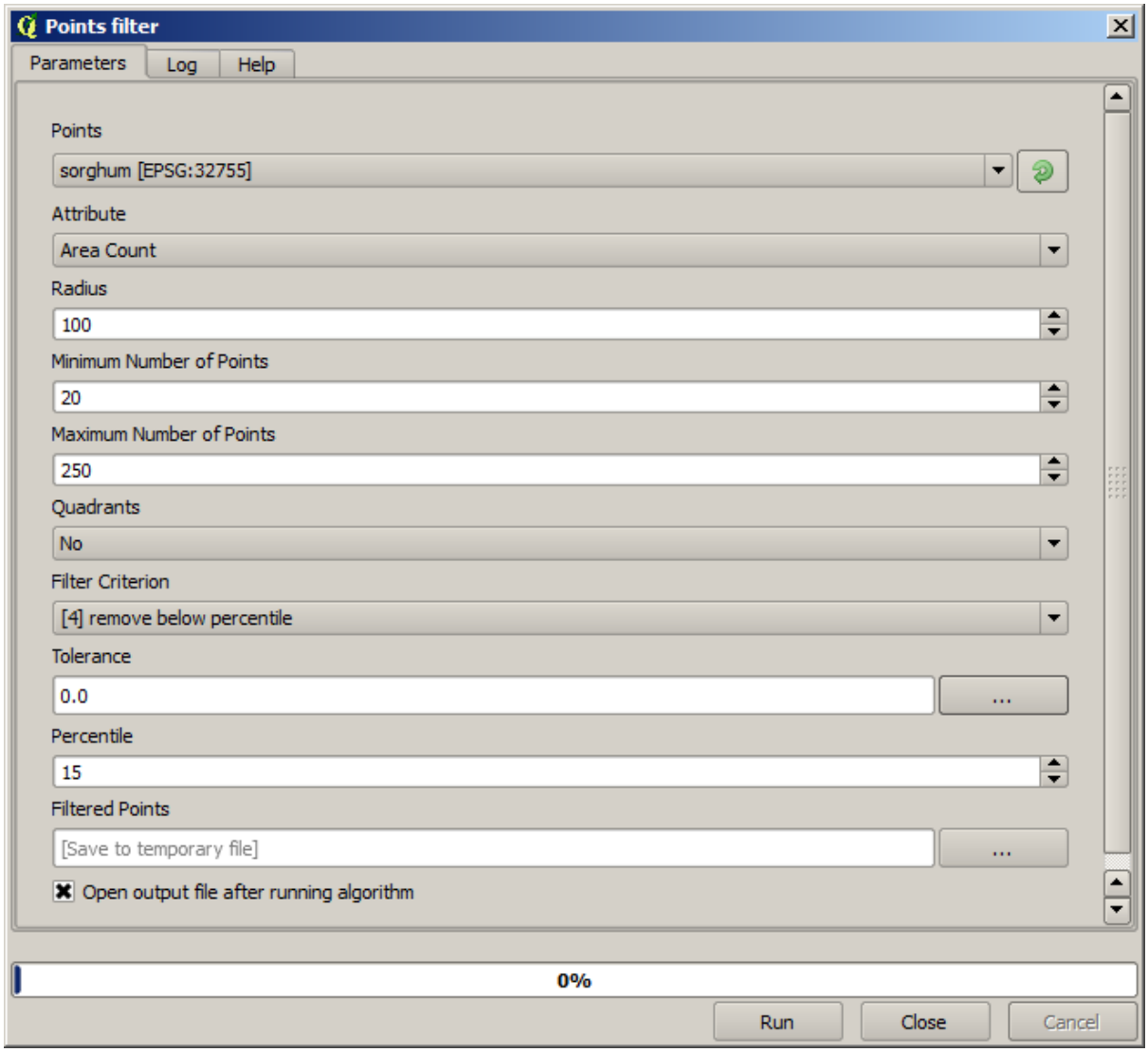
Open the example data for this lesson, which should look like this.



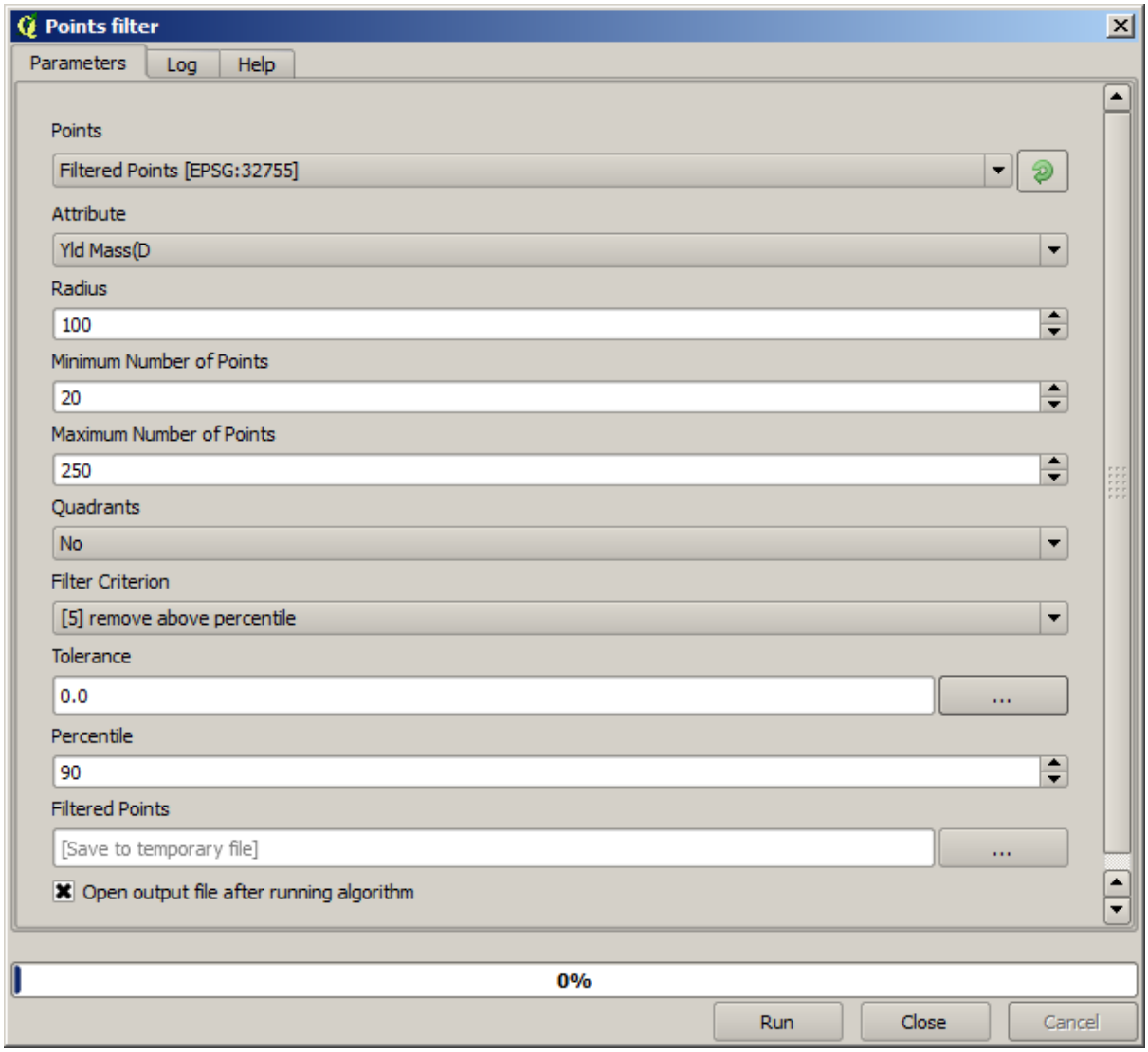
The data correspond to crop yield data, as produced by a modern harvester, and we will use it to get a raster layer of crop yield. We do not plan to do any further analysis with that layer, but just to use it as a background layer for easily identifying the most productive areas and also those where productivity can be improved.

The first thing to do is to clean–up the layer, since it contains redundant points. These are caused by the movement of the harvester, in places where it has to do a turn or it changes its speed for some reason. The *Points filter* algorithm will be useful for this. We will use it twice, to remove points that can be considered outliers both in the upper and lower part of the distribution.

For the first execution, use the following parameter values.



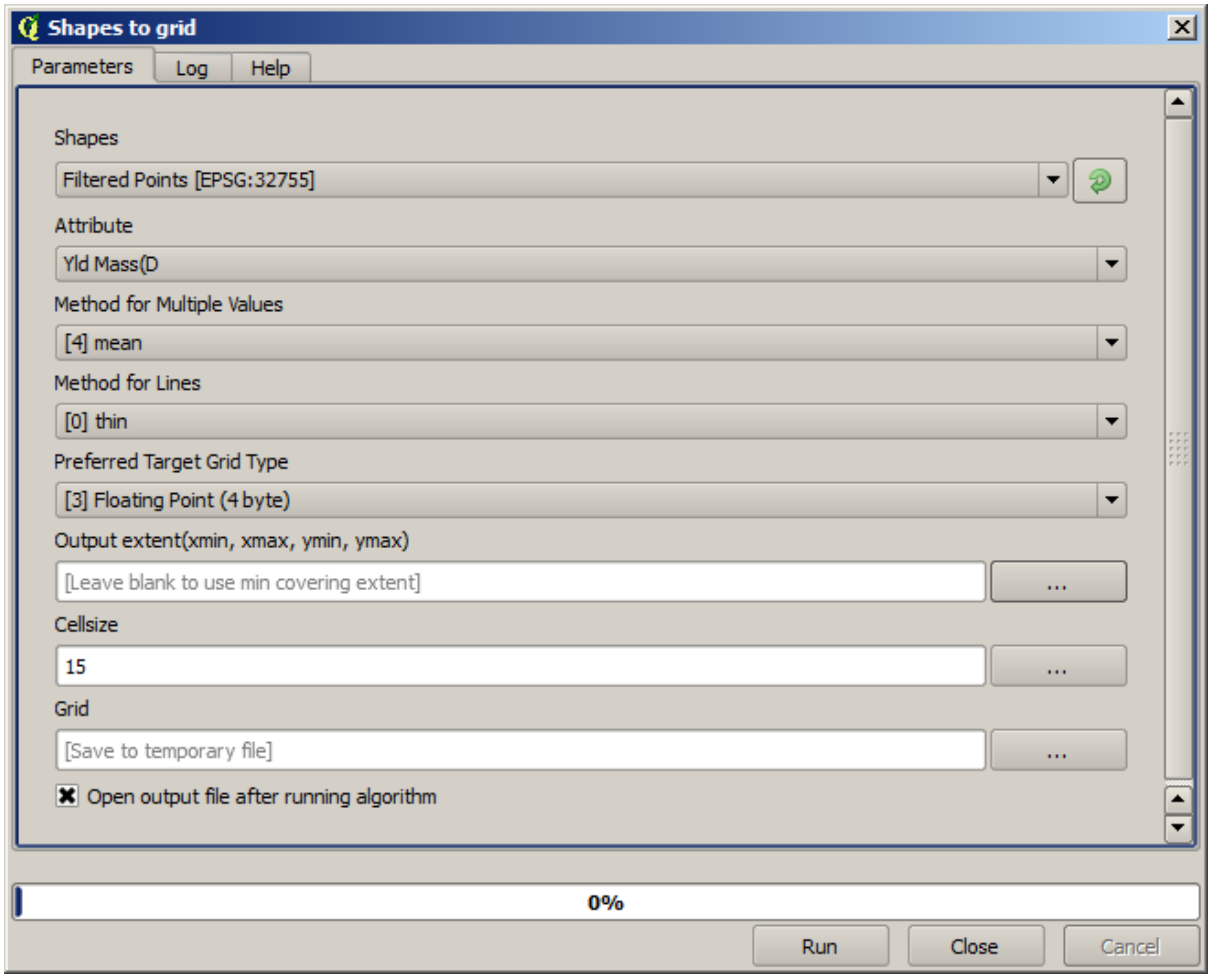
Now for the next one, use the configuration shown below.



Notice that we are not using the original layer as input, but the output of the previous run instead.

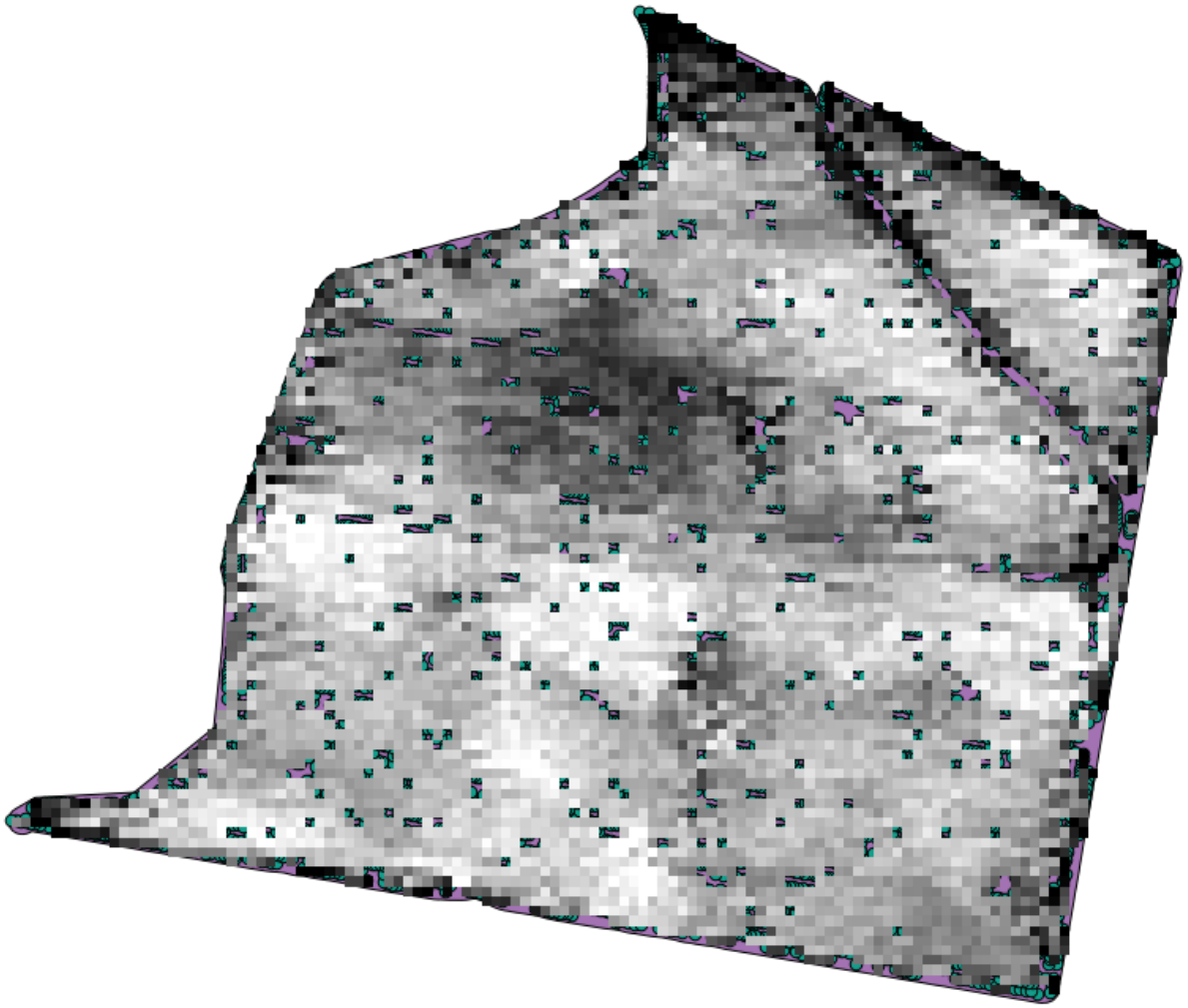
The final filter layer, with a reduced set of points, should look similar to the original one, but it contains a smaller number of points. You can check that by comparing their attribute tables.

Now let's rasterize the layer using the *Rasterize* algorithm.

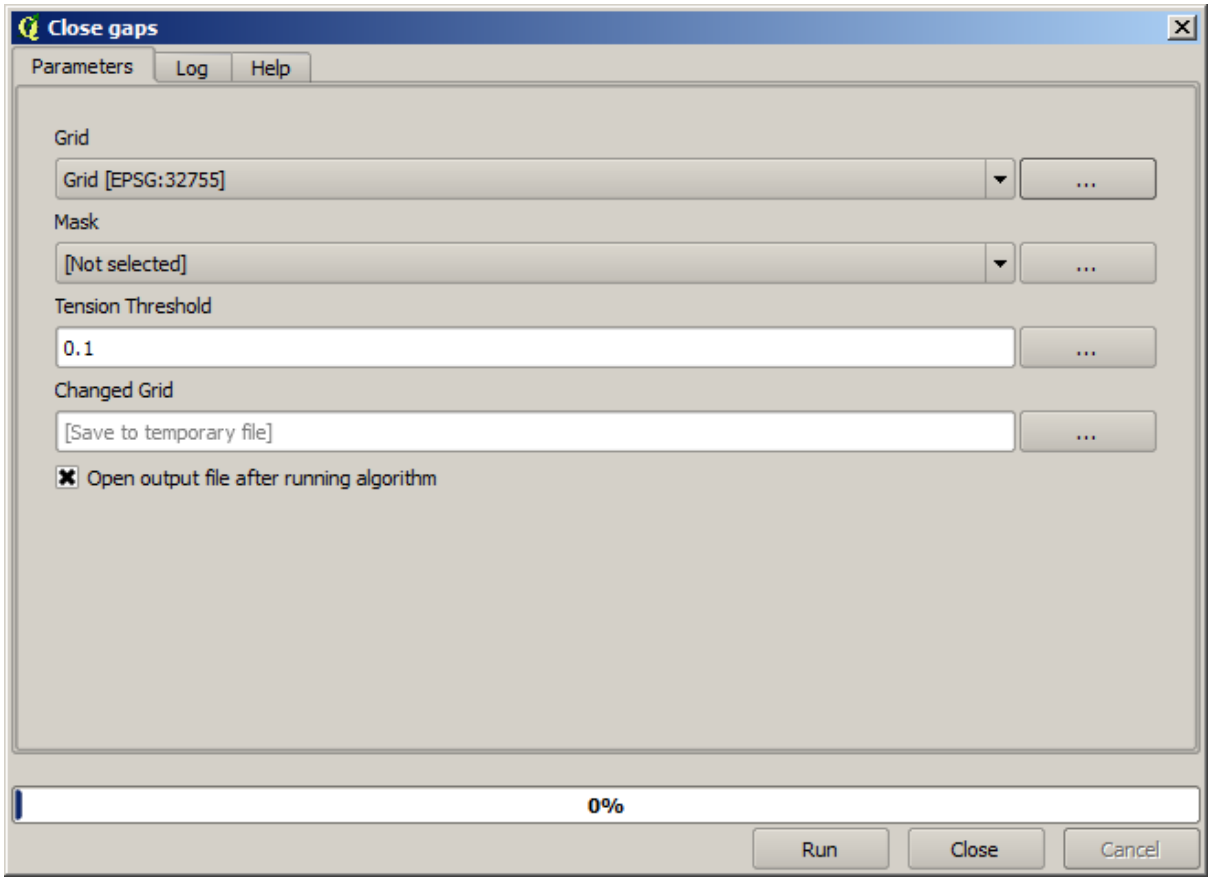


The *Filtered points* layer refers to the resulting one of the second filter. It has the same name as the one produced by the first filter, since the name is assigned by the algorithm, but you should not use the first one. Since we will not be using it for anything else, you can safely remove it from your project to avoid confusion, and leave just the last filtered layer.

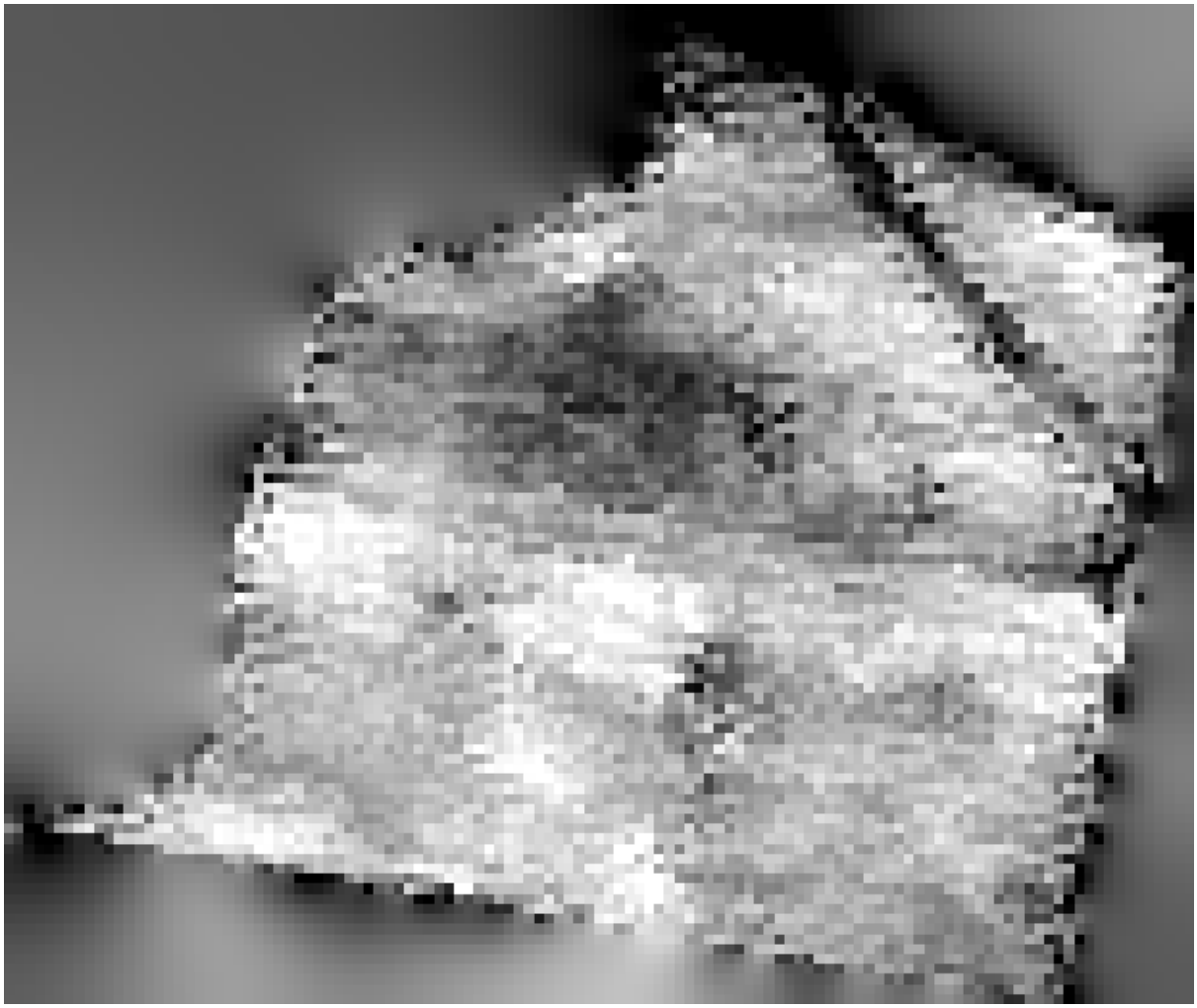
The resulting raster layer looks like this.



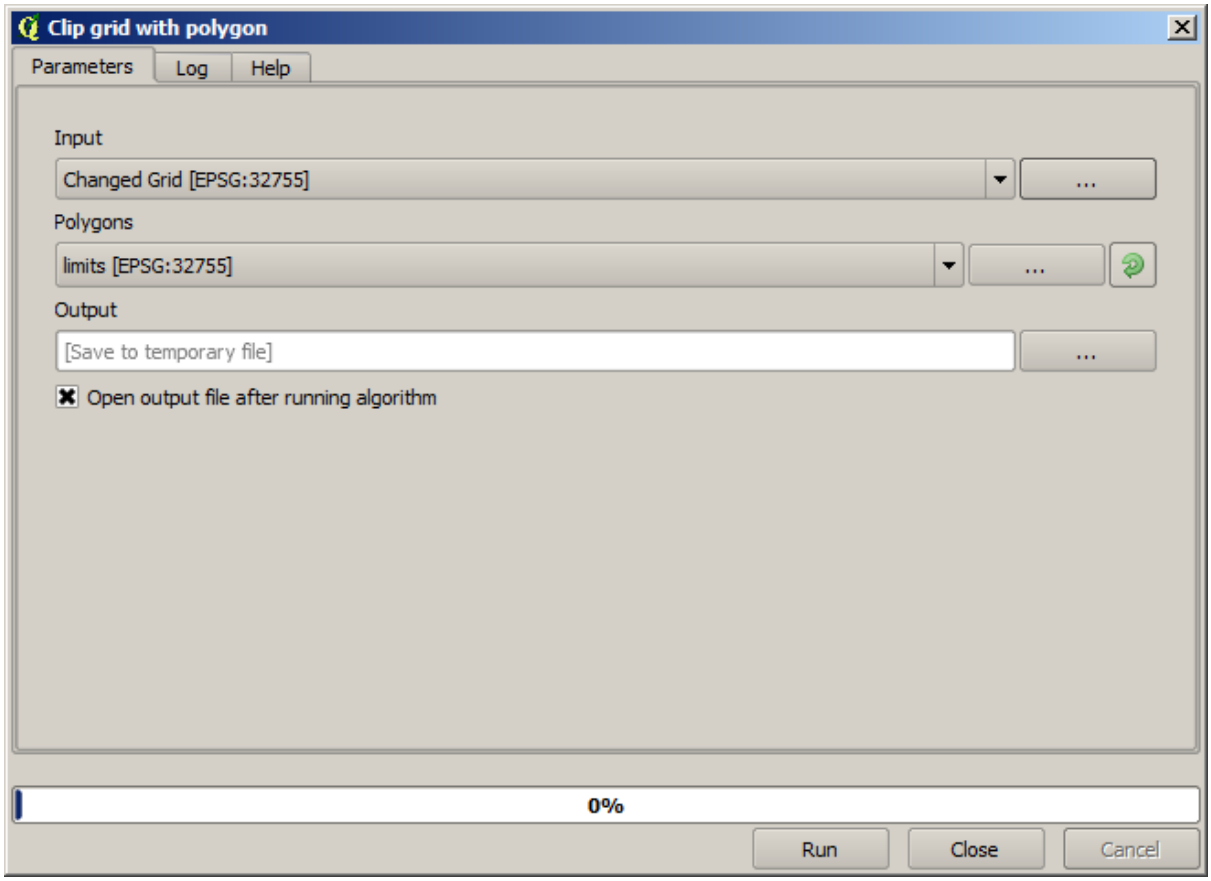
It is already a raster layer, but it is missing data in some of its cells. It only contain valid values in those cells that contained a point from the vector layer that we have just rasterized, and a no–data value in all the other ones. To fill the missing values, we can use the *Close gaps* algorithm.



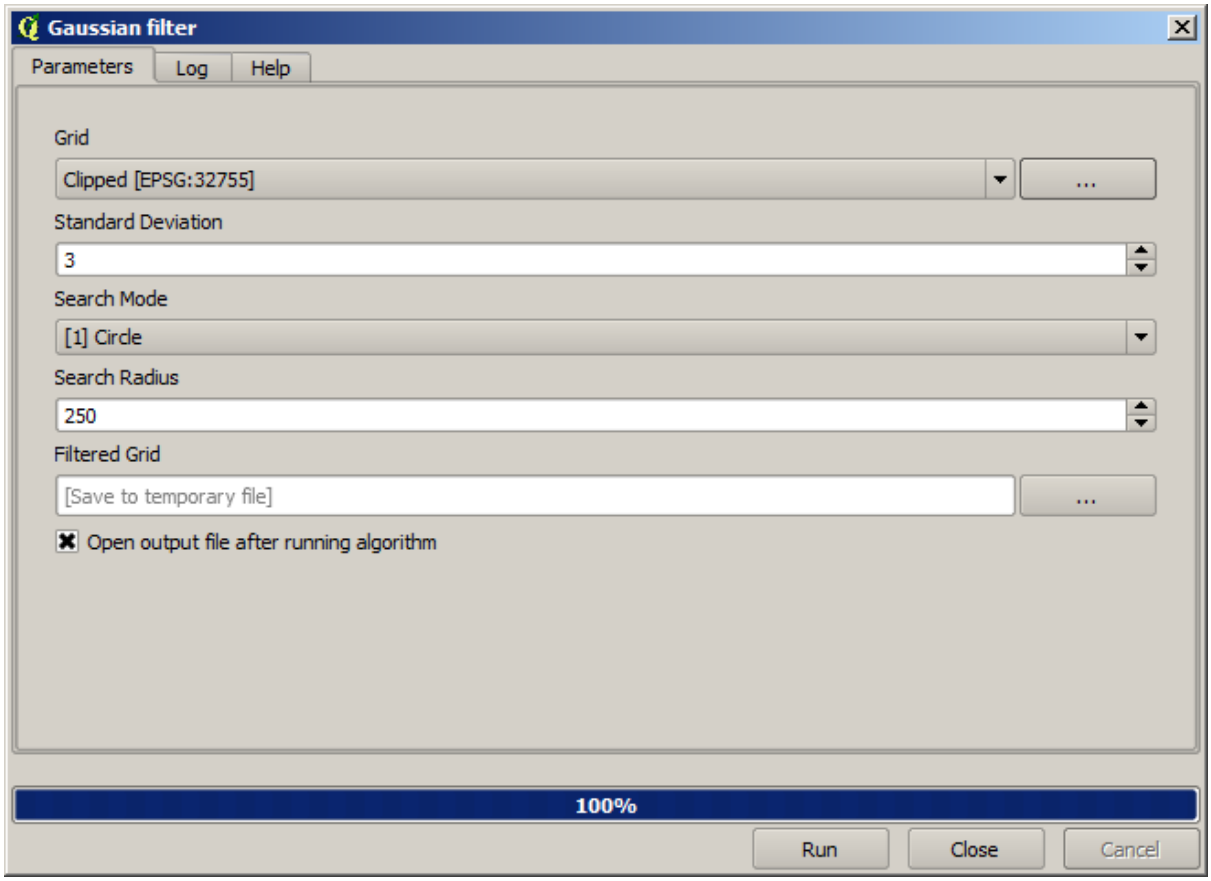
The layer without no–data values looks like this.



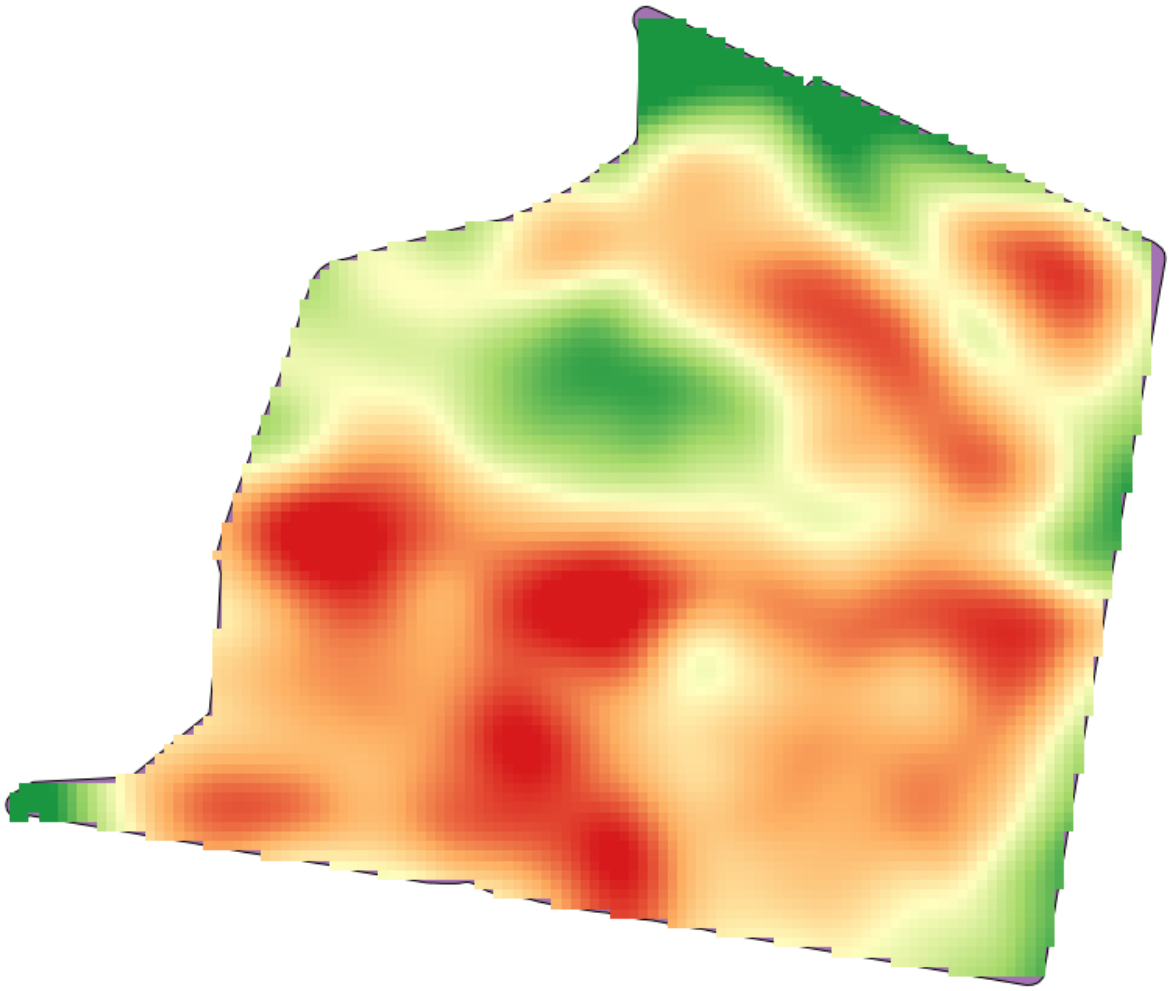
To restrict the area covered by the data to just the region where crop yield was measured, we can clip the raster layer with the provided limits layer.



And for a smoother result (less accurate but better for rendering in the background as a support layer), we can apply a *Gaussian filter* to the layer.



With the above parameters you will get the following result



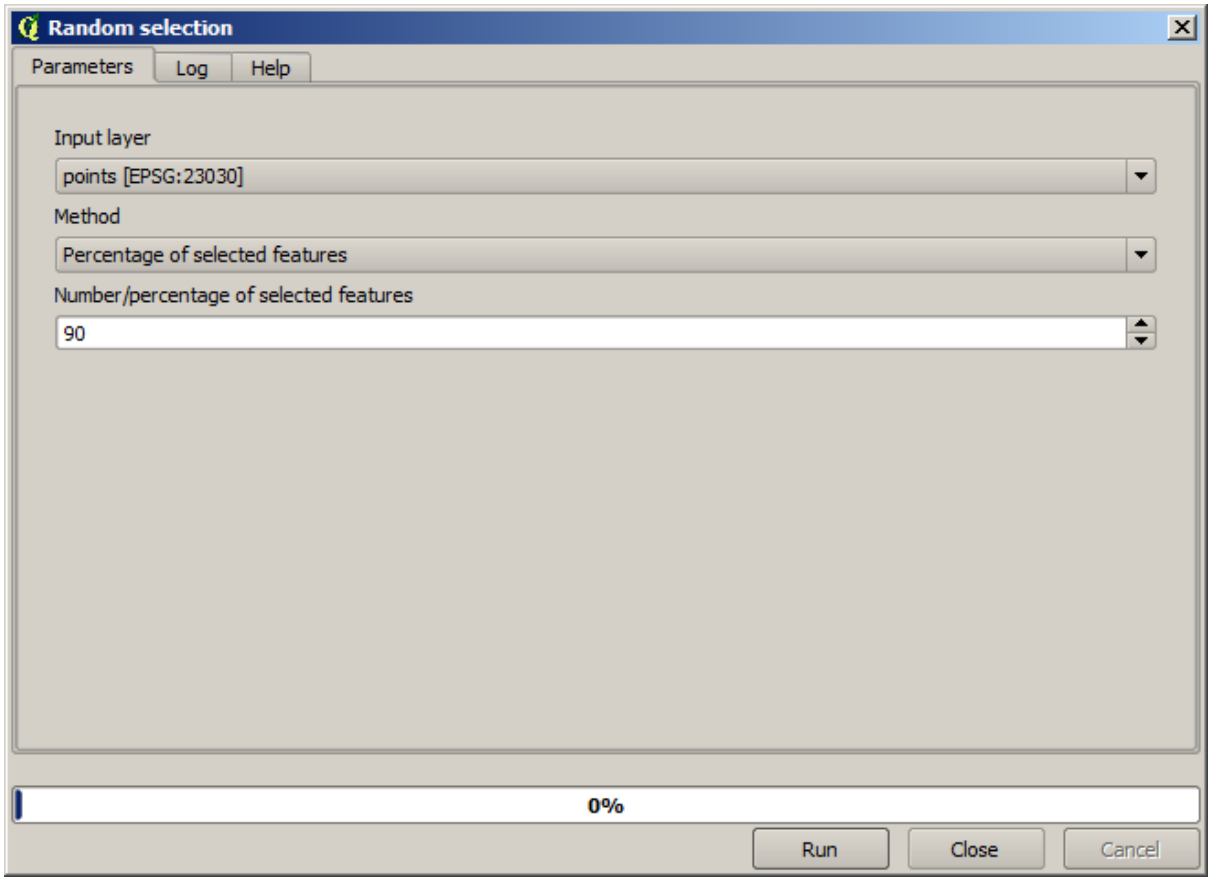
17.23 More interpolation

Nota: This chapter shows another practical case where interpolation algorithms are used.

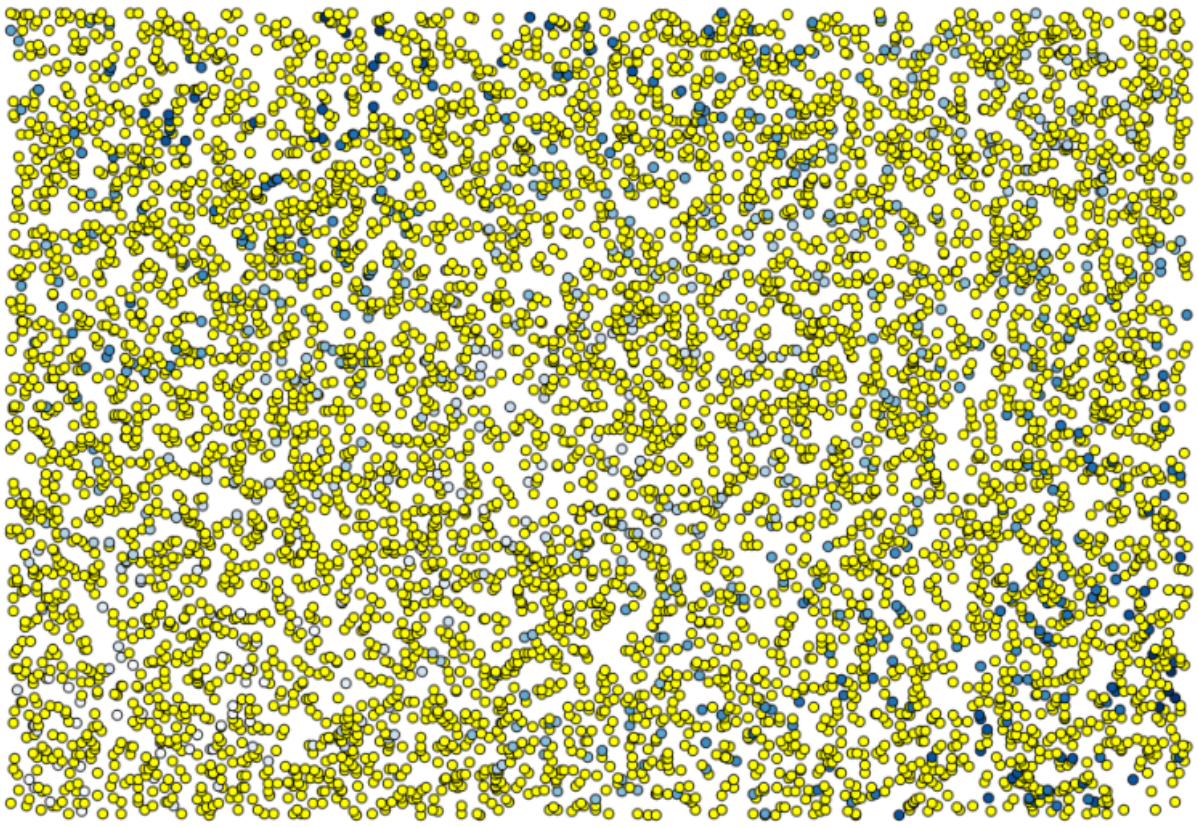
Interpolation is a common technique, and it can be used to demonstrate several techniques that can be applied using the QGIS processing framework. This lesson uses some interpolation algorithms that were already introduced, but has a different approach.

The data for this lesson contains also a points layer, in this case with elevation data. We are going to interpolate it much in the same way as we did in the previous lesson, but this time we will save part of the original data to use it for assessing the quality of the interpolation process.

First, we have to rasterize the points layer and fill the resulting no–data cells, but using just a fraction of the points in the layer. We will save 10% of the points for a later check, so we need to have 90% of the points ready for the interpolation. To do so, we could use the *Split shapes layer randomly* algorithm, which we have already used in a previous lesson, but there is a better way to do that, without having to create any new intermediate layer. Instead of that, we can just select the points we want to use for the interpolation (the 90% fraction), and then run the algorithm. As we have already seen, the rasterizing algorithm will use only those selected points and ignore the rest. The selection can be done using the *Random selection* algorithm. Run it with the following parameters.

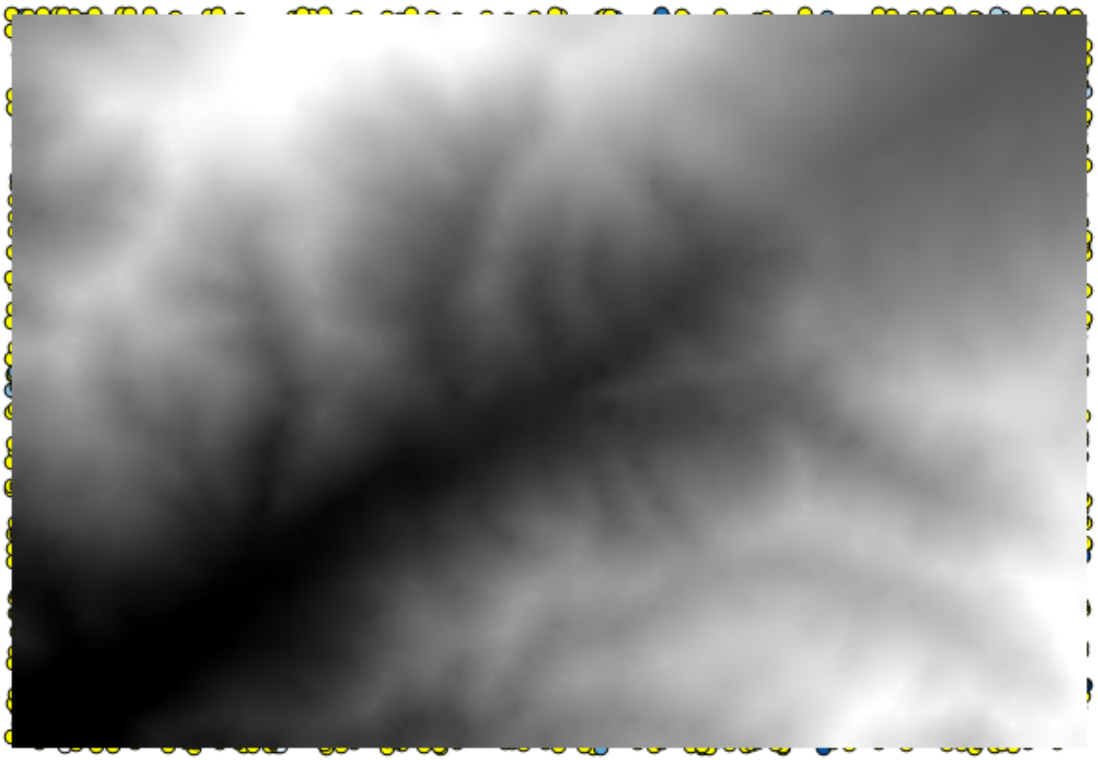


That will select 90% of the points in the layer to rasterize



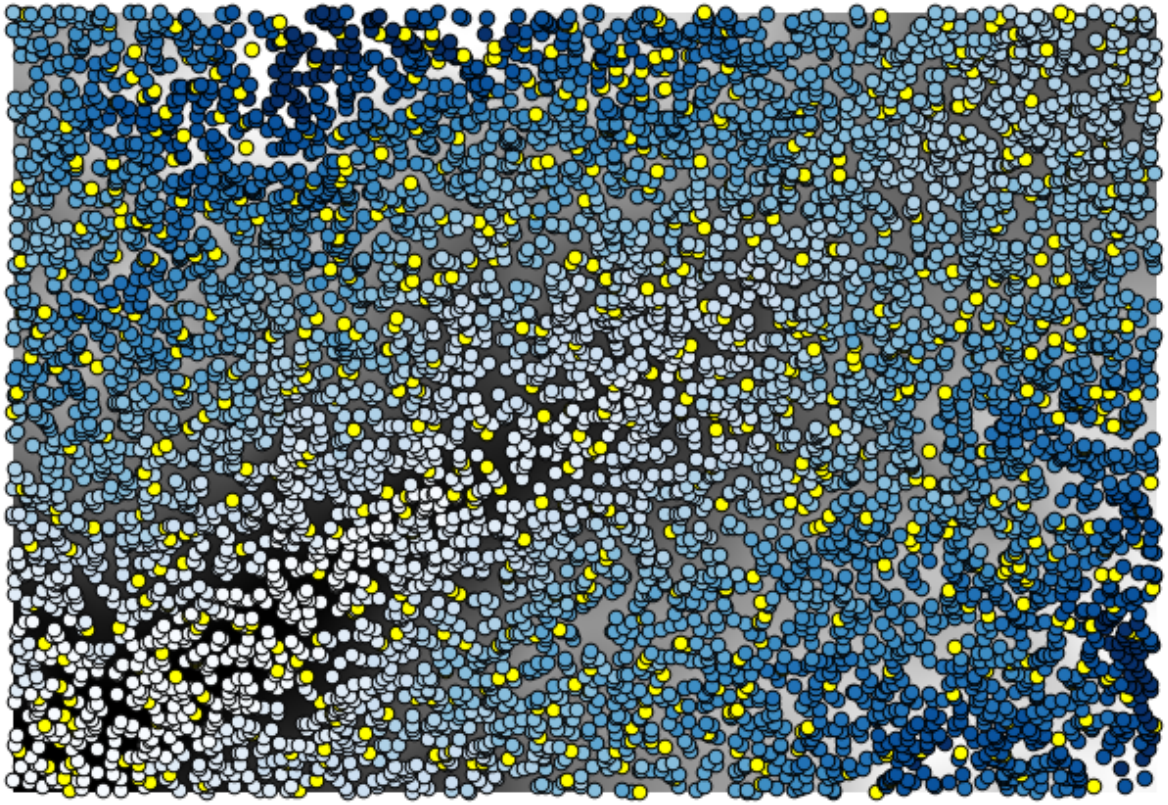
The selection is random, so your selection might differ from the selection shown in the above image.

Now run the *Rasterize* algorithm to get the first raster layer, and then run the *Close gaps* algorithm to fill the no–data cells [Cell resolution: 100 m].

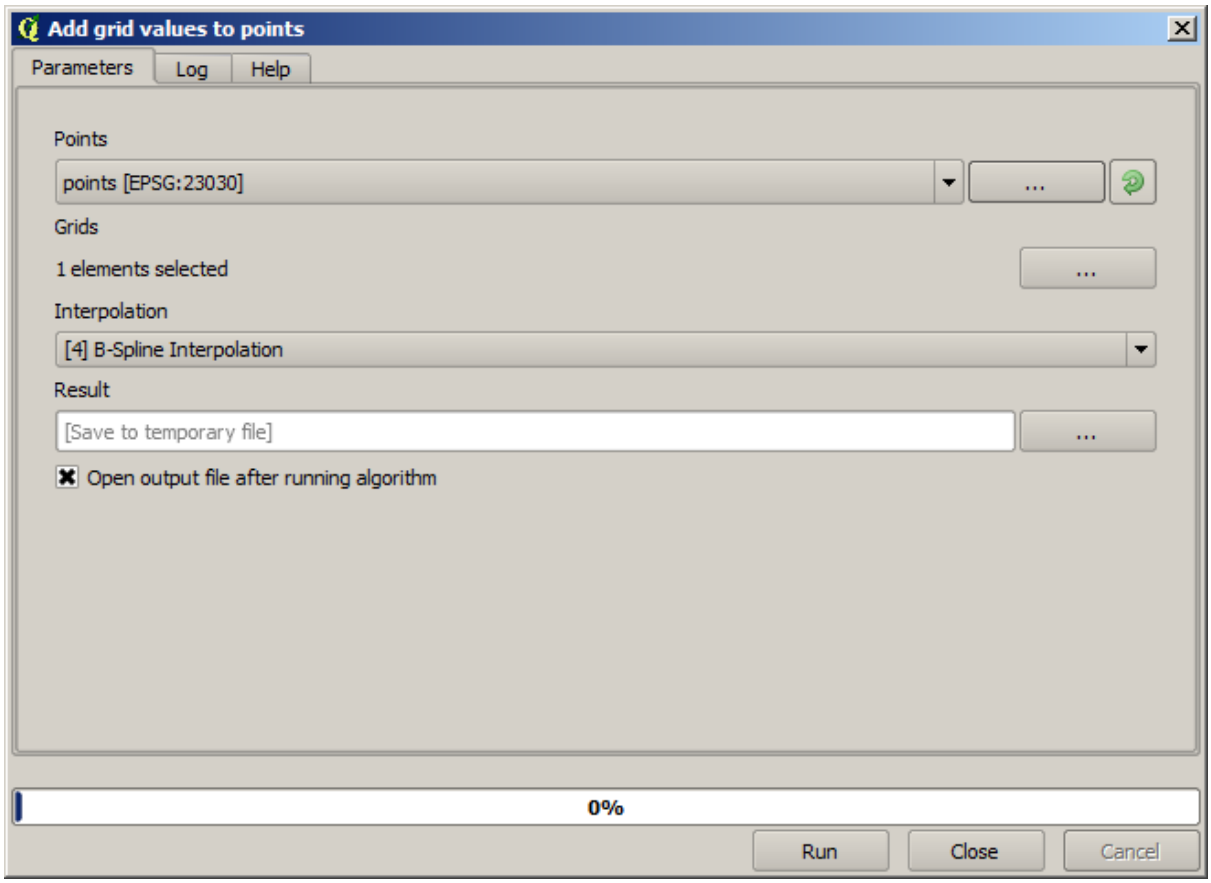


To check the quality of the interpolation, we can now use the points that are not selected. At this point, we know the real elevation (the value in the points layer) and the interpolated elevation (the value in the interpolated raster layer). We can compare the two by computing the differences between those values.

Since we are going to use the points that are not selected, first, let's invert the selection.

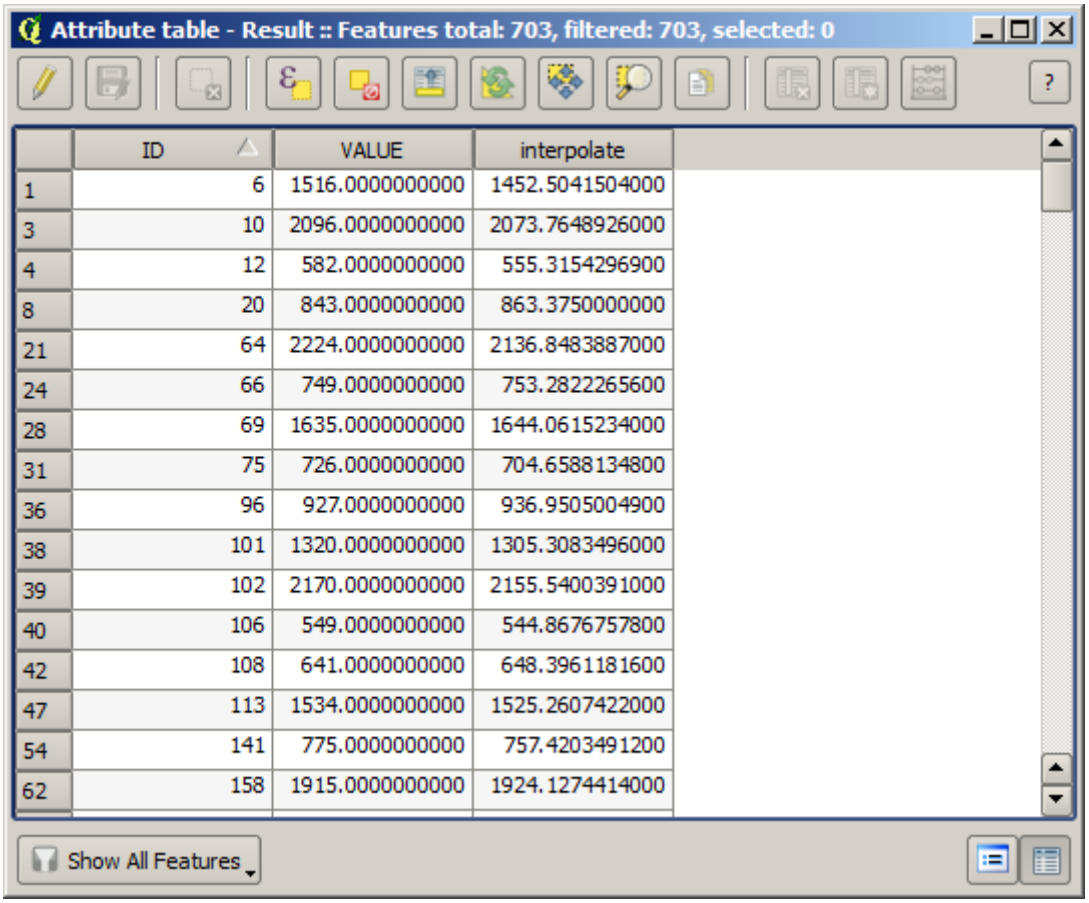


The points contain the original values, but not the interpolated ones. To add them in a new field, we can use the *Add raster values to points* algorithm



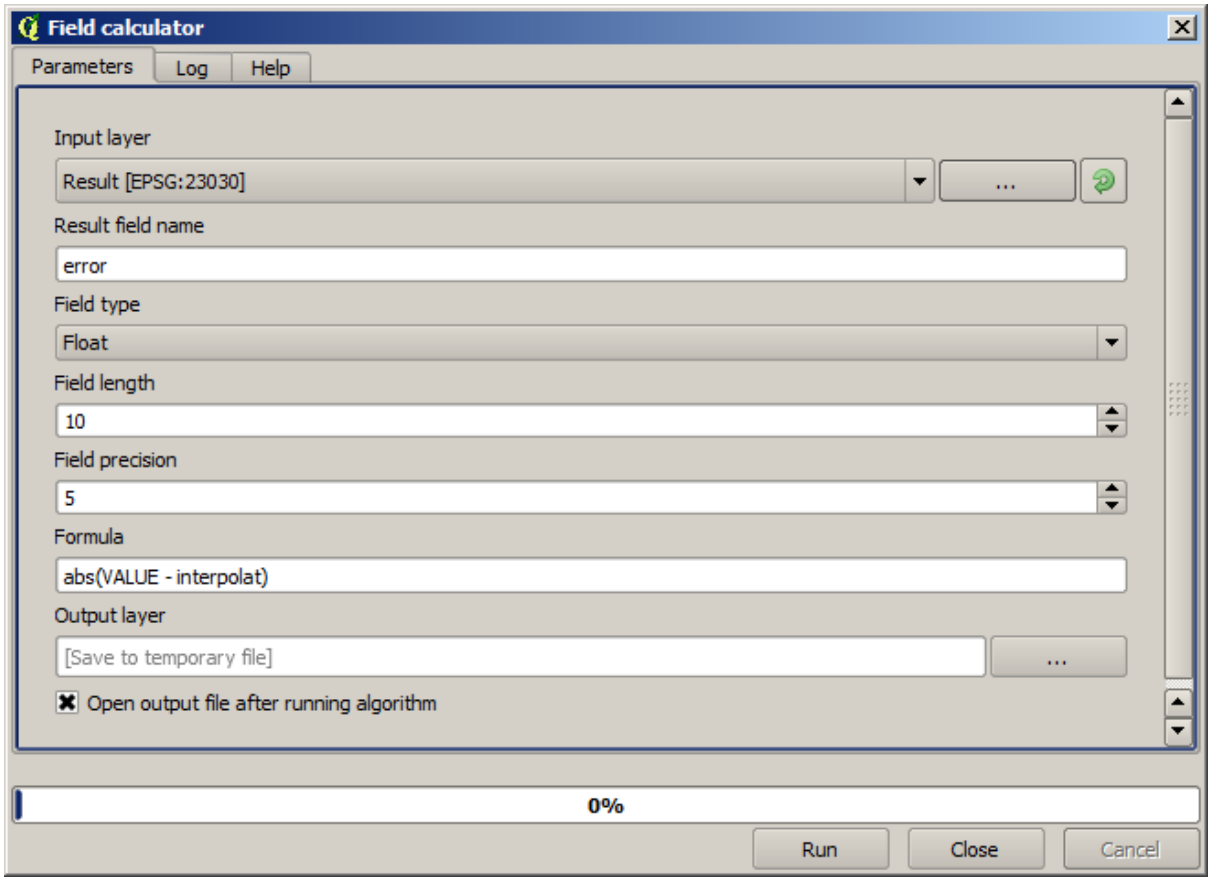
The raster layer to select (the algorithm supports multiple raster, but we just need one) is the resulting one from the interpolation. We have renamed it to *interpolate* and that layer name is the one that will be used for the name of the field to add.

Now we have a vector layer that contains both values, with points that were not used for the interpolation.



	ID	VALUE	interpolate
1	6	1516.0000000000	1452.5041504000
3	10	2096.0000000000	2073.7648926000
4	12	582.0000000000	555.3154296900
8	20	843.0000000000	863.3750000000
21	64	2224.0000000000	2136.8483887000
24	66	749.0000000000	753.2822265600
28	69	1635.0000000000	1644.0615234000
31	75	726.0000000000	704.6588134800
36	96	927.0000000000	936.9505004900
38	101	1320.0000000000	1305.3083496000
39	102	2170.0000000000	2155.5400391000
40	106	549.0000000000	544.8676757800
42	108	641.0000000000	648.3961181600
47	113	1534.0000000000	1525.2607422000
54	141	775.0000000000	757.4203491200
62	158	1915.0000000000	1924.1274414000

Now, we will use the fields calculator for this task. Open the *Field calculator* algorithm and run it with the following parameters.



If your field with the values from the raster layer has a different name, you should modify the above formula accordingly. Running this algorithm, you will get a new layer with just the points that we haven't used for the interpolation, each of them containing the difference between the two elevation values.

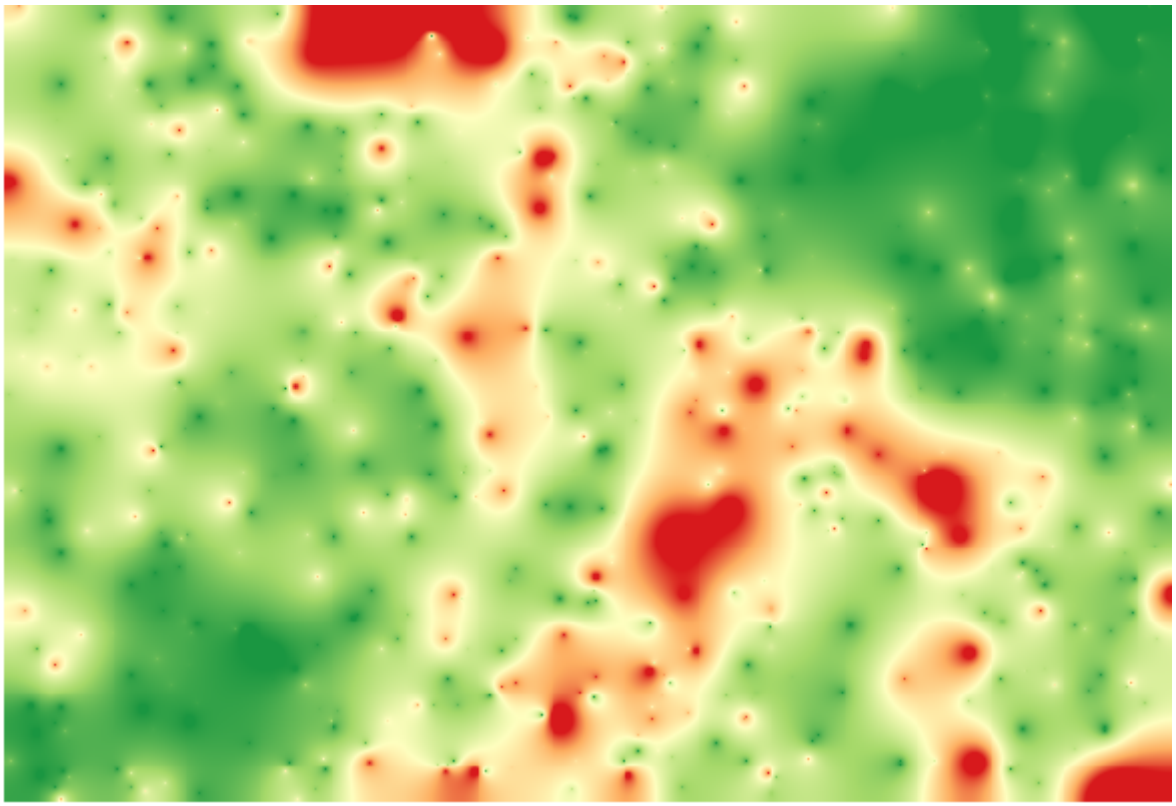
Representing that layer according to that value will give us a first idea of where the largest discrepancies are found.

Attribute table - Output layer :: Features total: 703, filtered: 703, selected: 0

	ID	VALUE	interpolat	error
0	4107	1243.0000000000	1199.6501465000	43.34985
1	6	1516.0000000000	1452.5041504000	63.49585
2	4112	1594.0000000000	1590.4835205000	3.51648
3	10	2096.0000000000	2073.7648926000	22.23511
4	12	582.0000000000	555.3154296900	26.68457
5	4121	1101.0000000000	1103.0323486000	2.03235
6	6176	1258.0000000000	1260.9846191000	2.98462
7	4125	1241.0000000000	1225.0878906000	15.91211
8	20	843.0000000000	863.3750000000	20.37500
9	6179	1195.0000000000	1198.4991455000	3.49915
10	2075	1786.0000000000	1799.5468750000	13.54688
11	4133	1196.0000000000	1156.2314453000	39.76855
12	6188	1720.0000000000	1724.4638672000	4.46387
13	6189	1497.0000000000	1498.2706299000	1.27063
14	6191	1349.0000000000	1347.5555420000	1.44446
15	2086	1277.0000000000	1296.1885986000	19.18860

Show All Features

Interpolating that layer will get you a raster layer with the estimated error in all points of the interpolated area.



You can also get the same information (difference between original point values and interpolated ones) directly with *GRASS ► v.sample*.

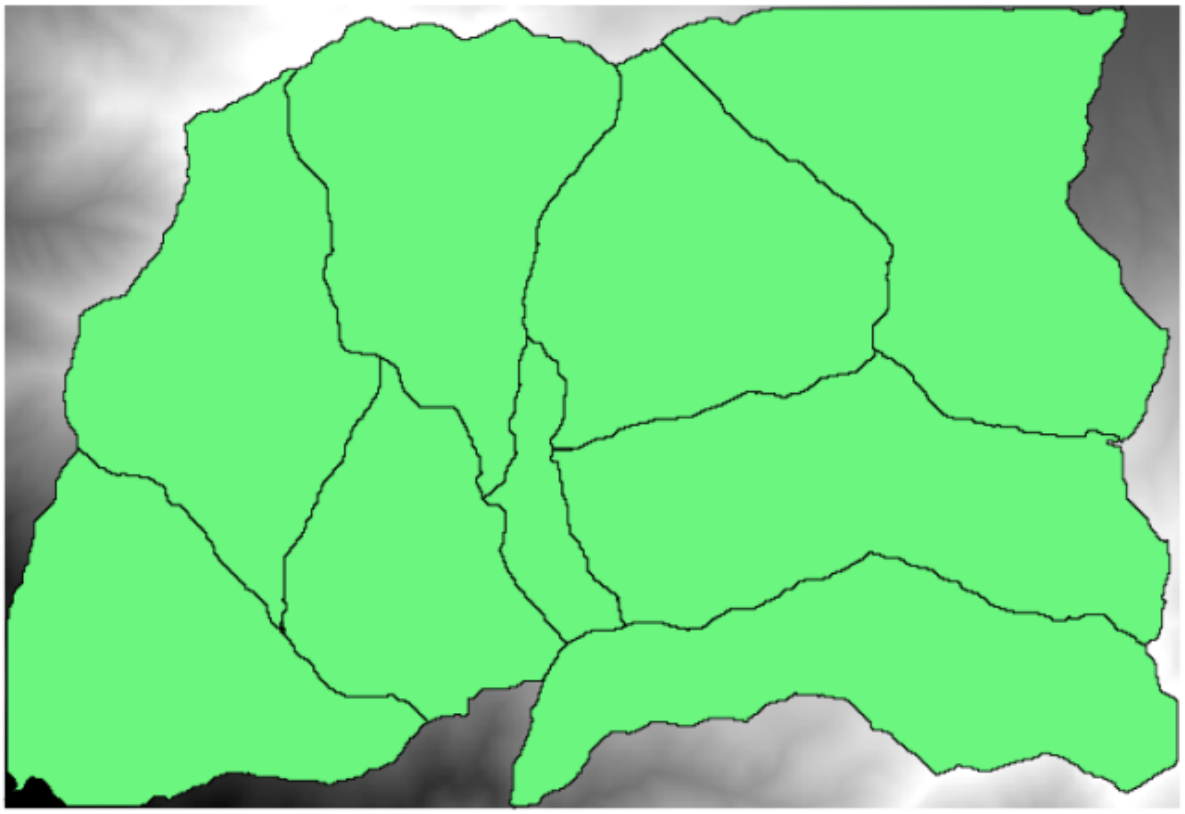
Your results might differ from these ones, since there is a random component introduced when running the random selection, at the beginning of this lesson.

17.24 Iterative execution of algorithms

Nota: This lesson shows a different way of executing algorithms that use vector layers, by running them repeatedly, iterating over the features in an input vector layer

We already know the model designer, which is one way of automating processing tasks. However, in some situations, the modeler might not be what we need to automate a given task. Let's see one of those situations and how to easily solve it using a different functionality: the iterative execution of algorithms.

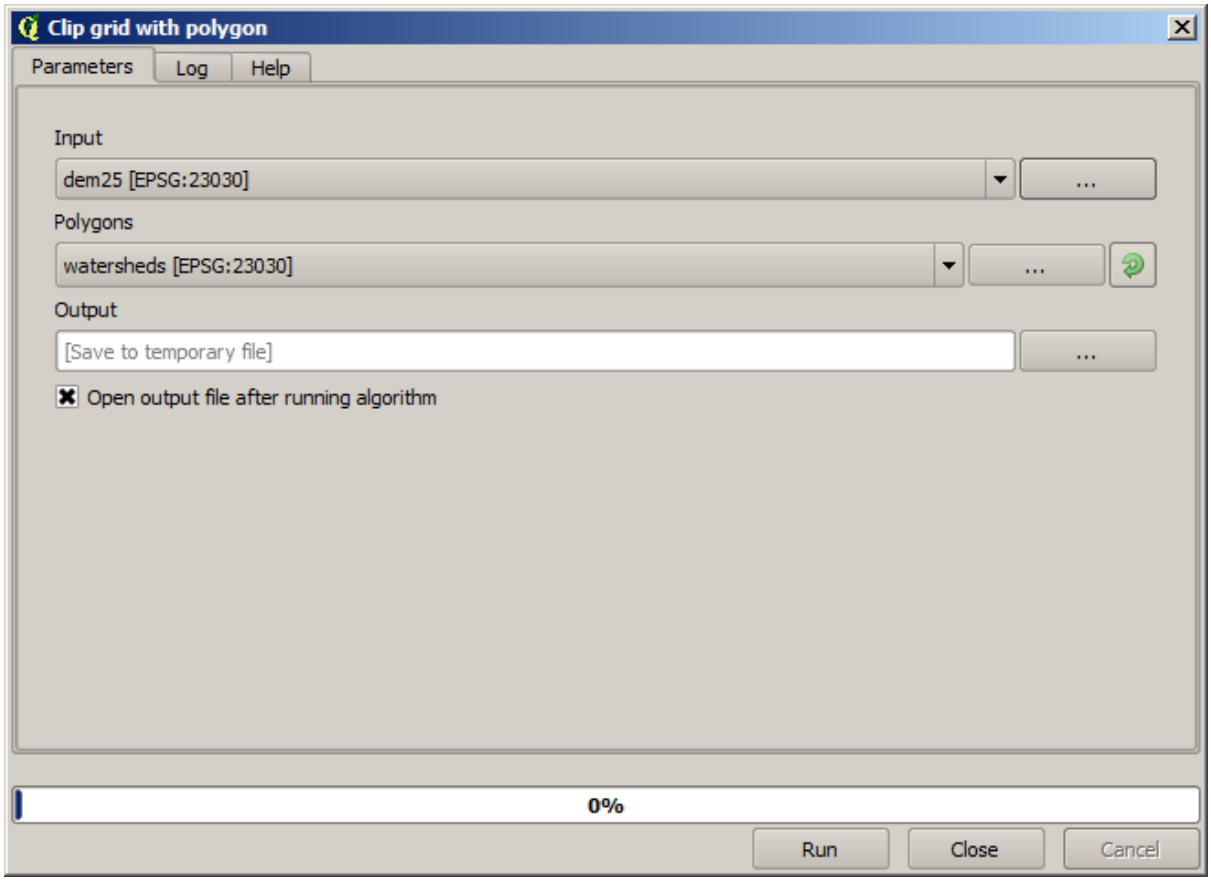
Open the data corresponding to this chapter. It should look like this.



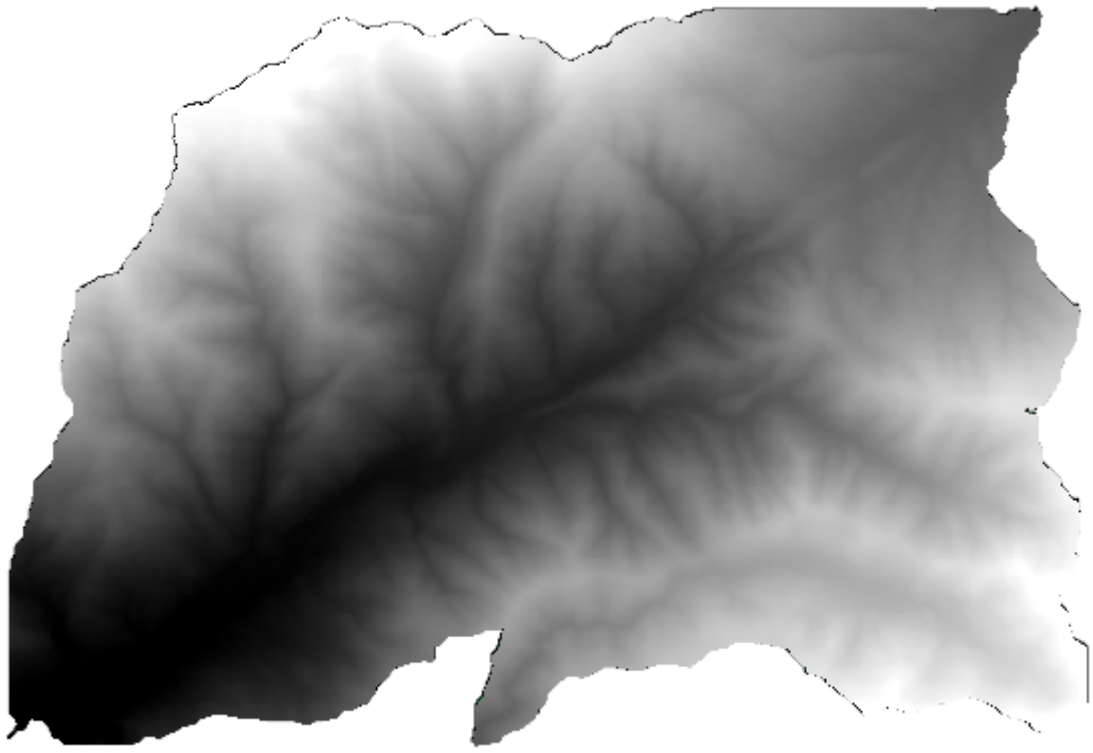
You will recognize our well-known DEM from previous chapters and a set of watersheds extracted from it. Imagine that you need to cut the DEM into several smaller layers, each of them containing just the elevation data corresponding to a single watershed. That will be useful if you later want to calculate some parameters related to each watershed, such as its mean elevation or it hypsographic curve.

This can be a lengthy and tedious task, especially if the number of watersheds is large. However, it is a task that can be easily automated, as we will see.

The algorithm to use for clipping a raster layer with a polygon layer is called *Clip raster with polygons*, and has the following parameters dialog.

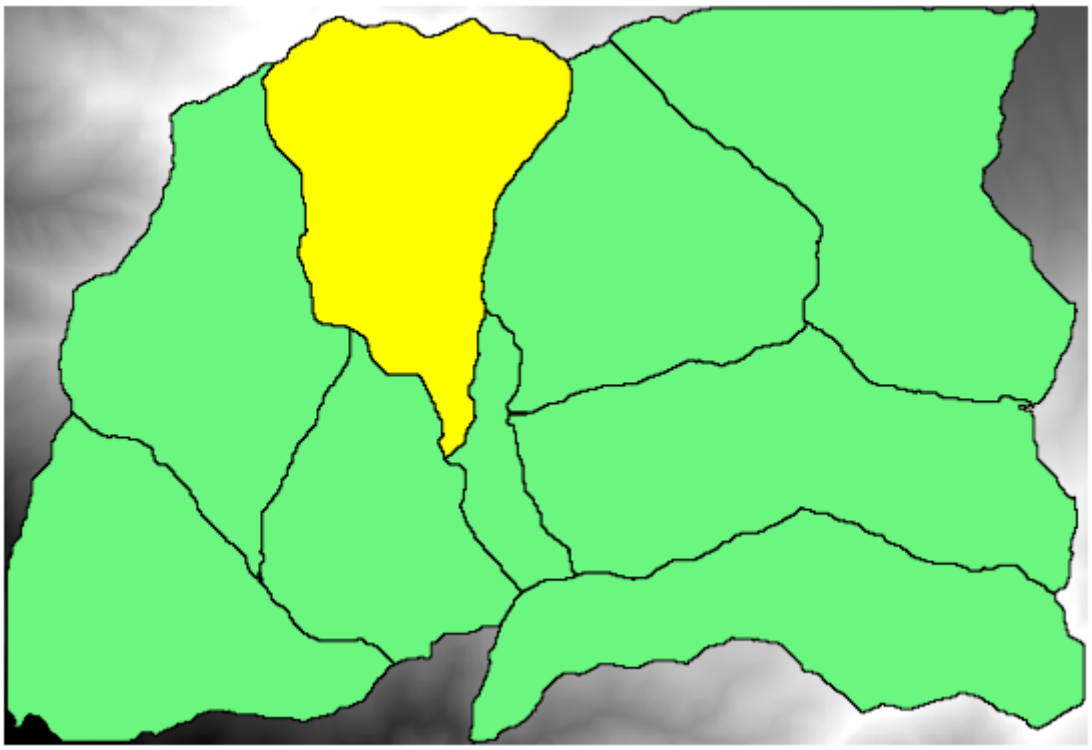


You can run it using the watersheds layer and the DEM as input, and you will get the following result.

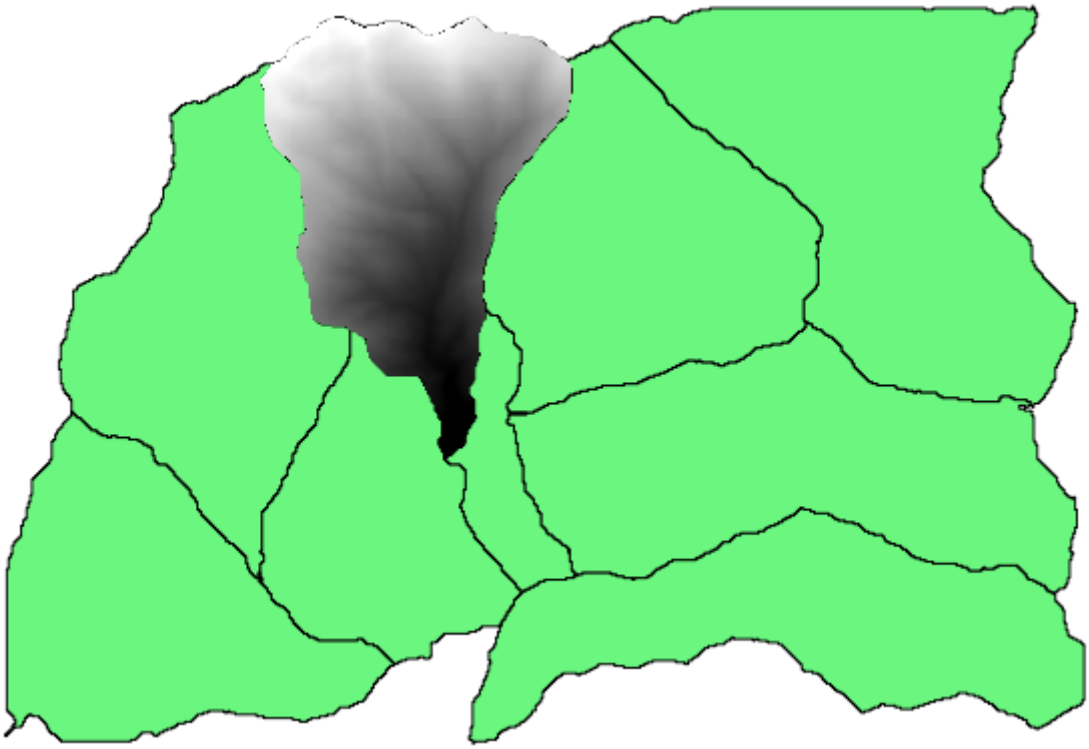


As you can see, the area covered by all the watershed polygons is used.

You can have the DEM clipped with just a single watershed by selecting the desired watershed and then running the algorithm as we did before.

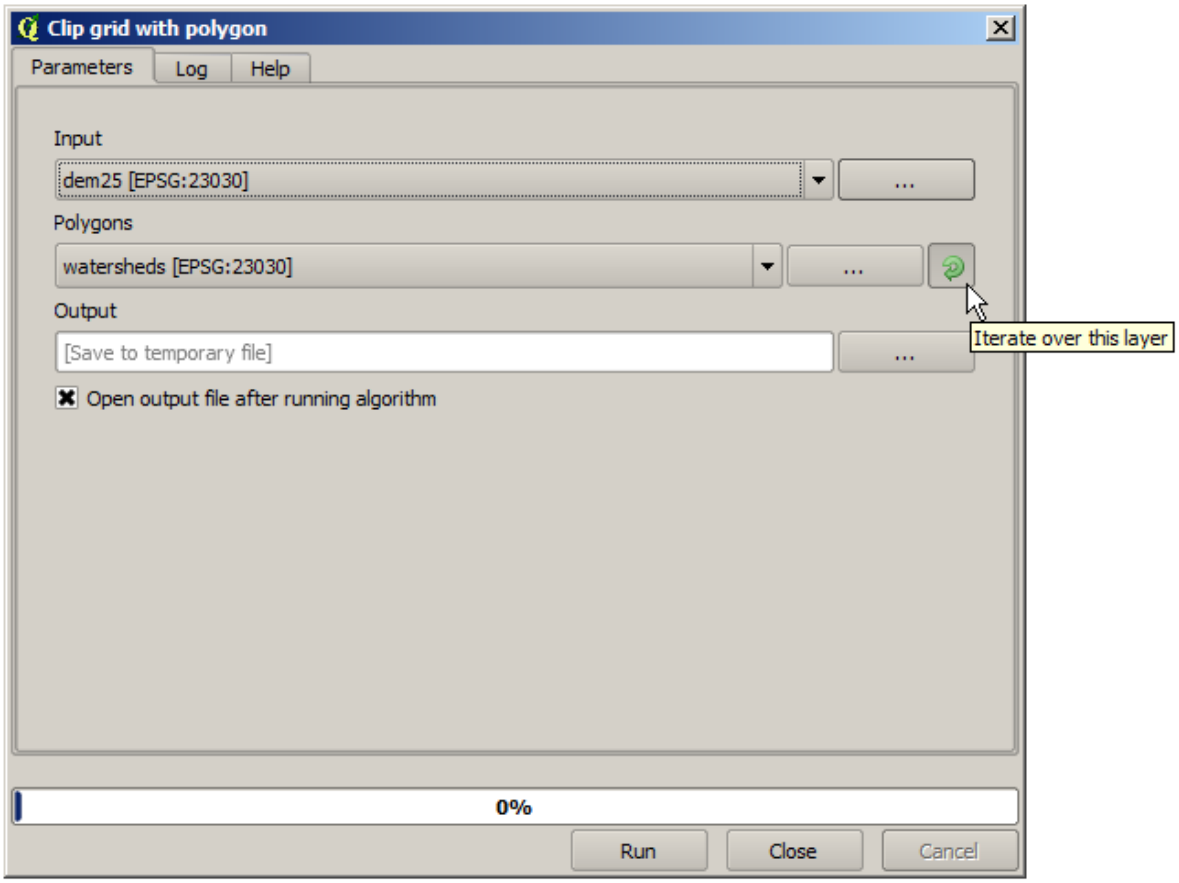


Since only selected features are used, only the selected polygon will be used to crop the raster layer.



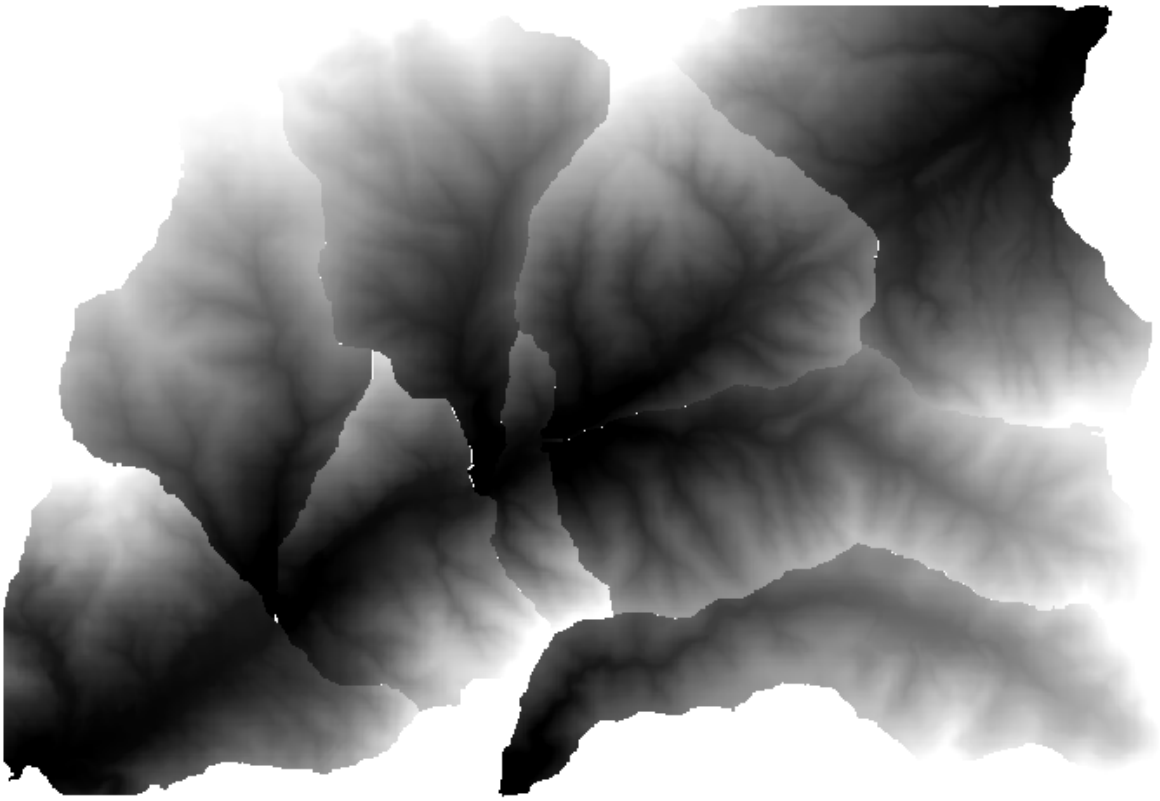
Doing this for all the watersheds will produce the result we are looking for, but it doesn't look like a very practical way of doing it. Instead, let's see how to automate that *select and crop* routine.

First of all, remove the previous selection, so all polygons will be used again. Now open the *Clip raster with polygon* algorithm and select the same inputs as before, but this time click on the button that you will find in the right–hand side of the vector layer input where you have selected the watersheds layer.



This button will cause the selected input layer to be split into as many layer as feature are found in it, each of them containing a single polygon. With that, the algorithm will be called repeatedly, one for each one of those single-polygon layers. The result, instead of just one raster layer in the case of this algorithm, will be a set of raster layers, each one of them corresponding to one of the executions of the algorithm.

Here's the result that you will get if you run the clipping algorithm as explained.



For each layer, the black and white color palette, (or whatever palette you are using), is adjusted differently, from its minimum to its maximum values. That's the reason why you can see the different pieces and the colors do not seem to match in the border between layers. Values, however, do match.

If you enter an output filename, resulting files will be named using that filename and a number corresponding to each iteration as suffix.

17.25 More iterative execution of algorithms

Nota: This lesson shows how to combine the iterative execution of algorithms with the modeler to get more automation.

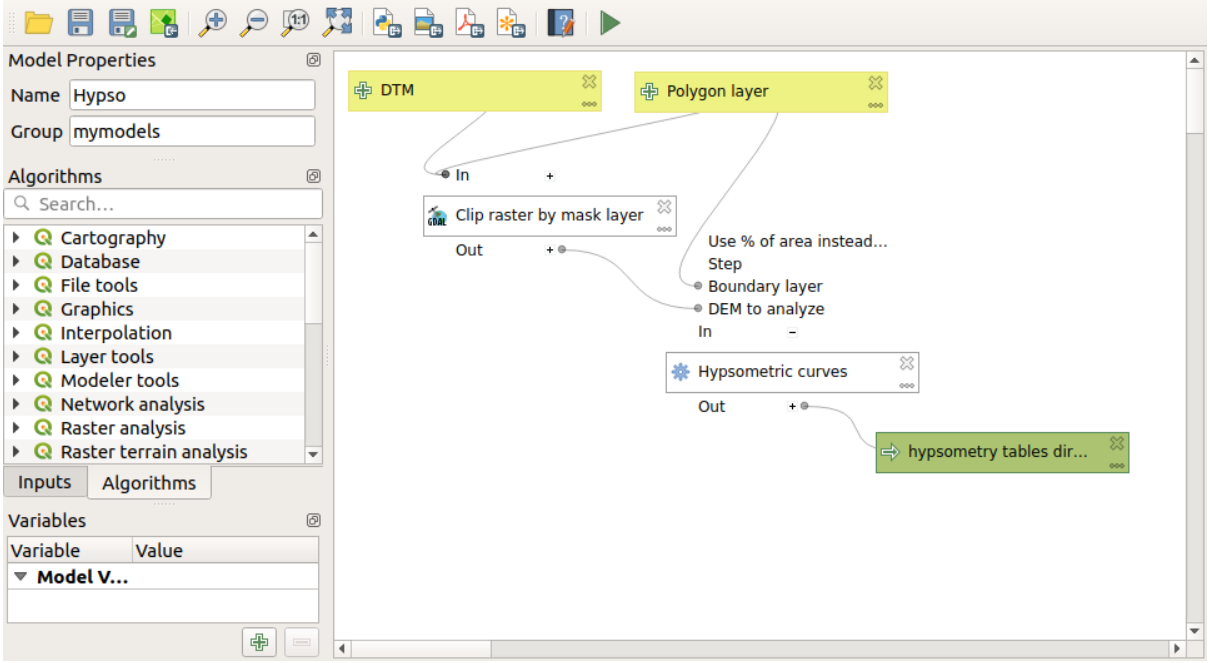
The iterative execution of algorithms is available not just for built-in algorithms, but also for the algorithms that you can create yourself, such as models. We are going to see how to combine a model and the iterative execution of algorithms, so we can obtain more complex results with ease.

The data the we are going to use for this lesson is the same one that we already used for the last one. In this case, instead of just clipping the DEM with each watershed polygon, we will add some extra steps and calculate a hypsometric curve for each of them, to study how elevation is distributed within the watershed.

Since we have a workflow that involves several steps (clipping + computing the hypsometric curve), we should go to the modeler and create the corresponding model for that workflow.

You can find the model already created in the data folder for this lesson, but it would be good if you first try to create it yourself. The clipped layer is not a final result in this case, since we are just interested in the curves, so this model will not generated any layers, but just a table with the curve data.

The model should look like this:

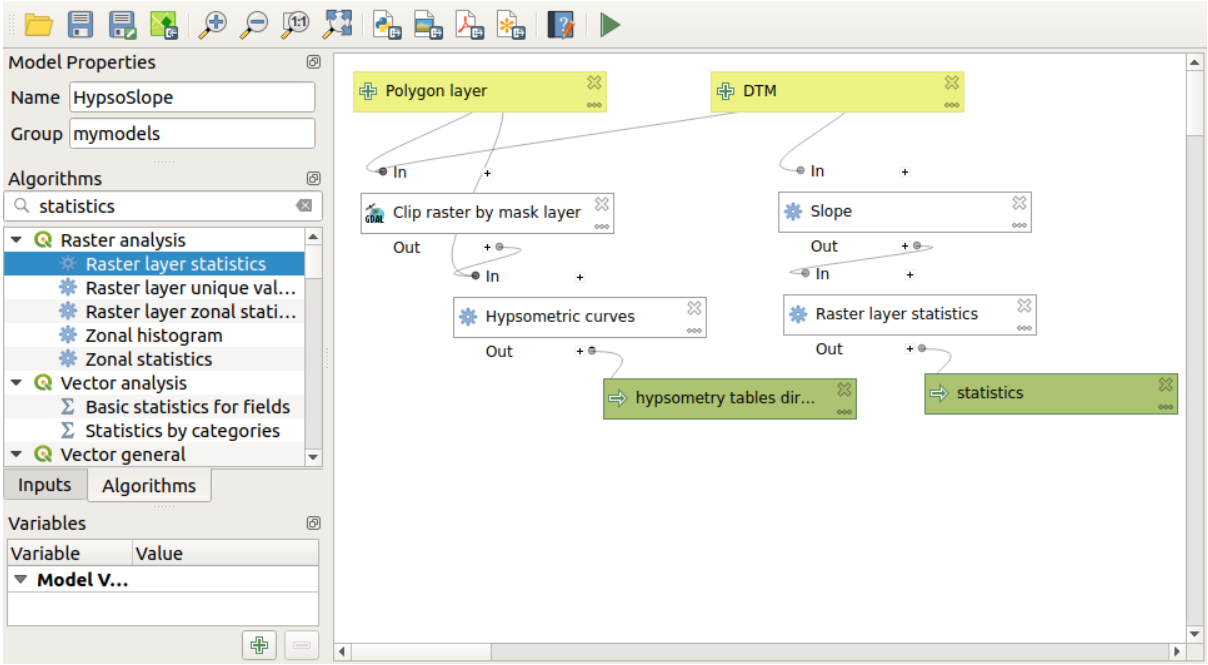


Add the model to you models folder, so it is available in the toolbox, and execute it.

Select the DEM and watersheds basins.

The algorithm will generate tables for all the basins and place them in the output directory.

We can make this example more complex by extending the model and computing some slope statistics. Add the *Slope* algorithm to the model, and then the *Raster statistics* algorithm, which should use the slope output as its only input.



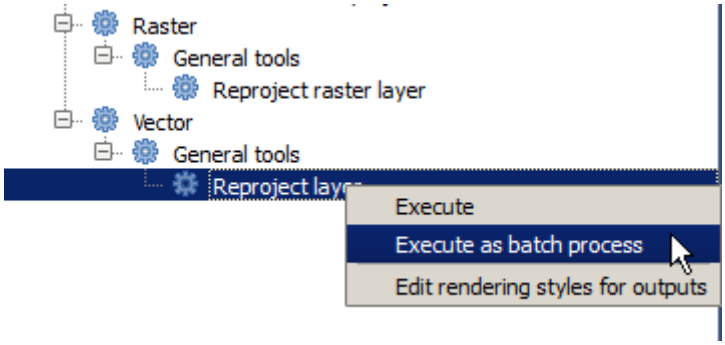
If you now run the model, apart from the tables you will get a set of pages with statistics. These pages will be available in the results dialog.

17.26 O interface do processamento em lote

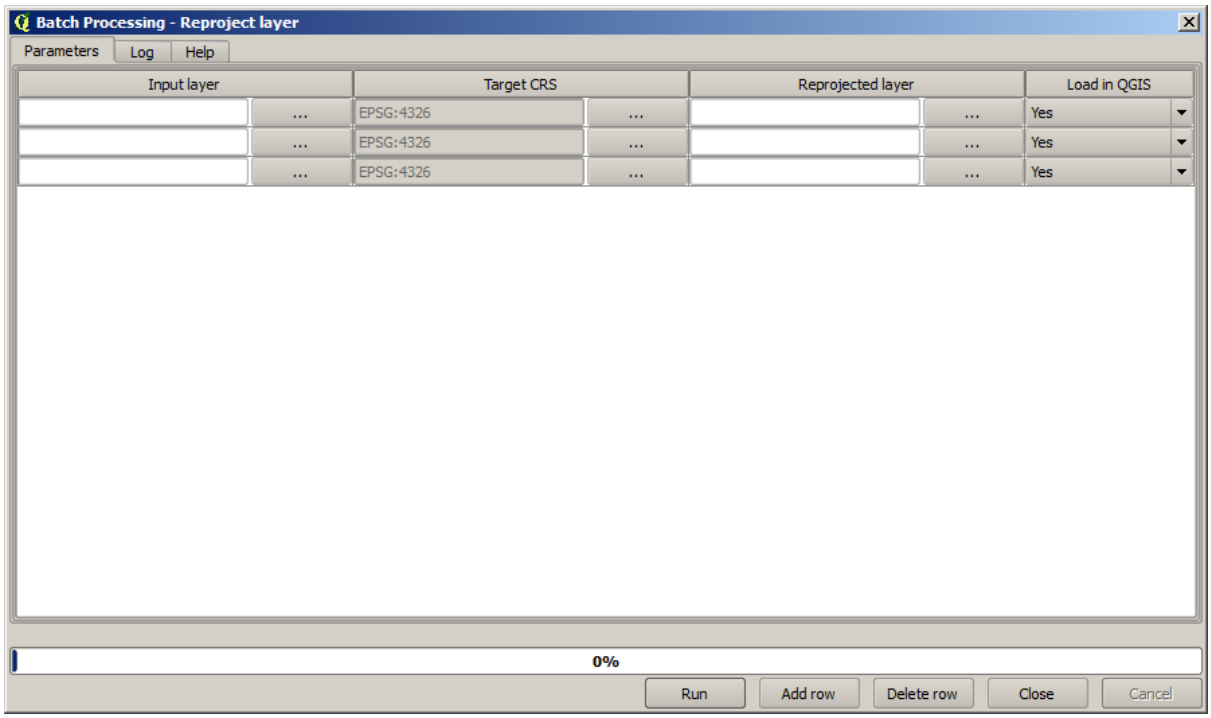
Nota: This lesson introduces the batch processing interface, which allows to execute a single algorithm with a set of different input values.

Sometimes a given algorithm has to be executed repeatedly with different inputs. This is, for instance, the case when a set of input files have to be converted from one format to another, or when several layers in a given projection must be converted into another projection.

In that case, calling the algorithm repeatedly on the toolbox is not the best option. Instead, the batch processing interface should be used, which greatly simplifies performing a multiple execution of a given algorithm. To run an algorithm as a batch process, find it in the toolbox, and instead of double–clicking on it, right–click on it and select *Run as batch process*.



For this example, we will use the *Reproject algorithm*, so find it and do as described above. You will get to the following dialog.

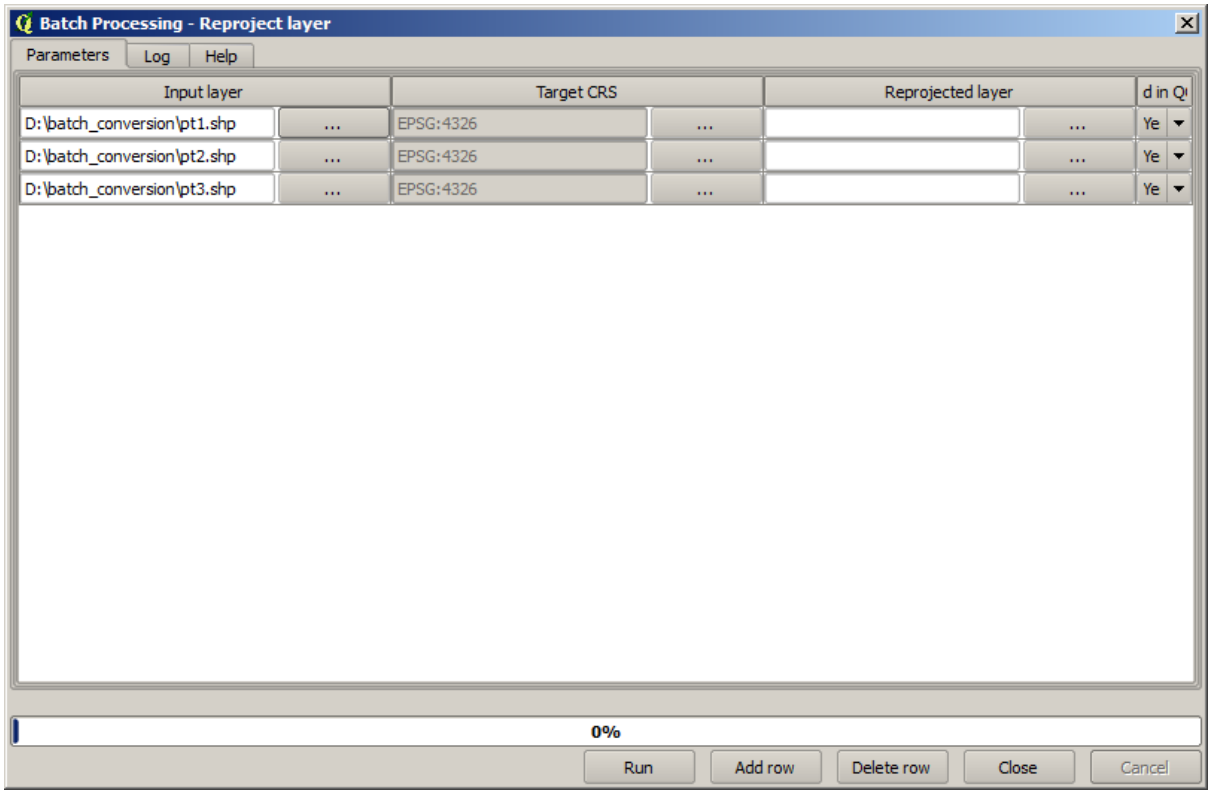


If you have a look at the data for this lesson, you will see that it contains a set of three shapefiles, but no QGIS project file. This is because, when an algorithm is run as a batch process, layer inputs can be selected either from the current QGIS project or from files. That makes it easier to process large amounts of layers, such as, for instance, all the layers in a given folder.

Each row in the table of the batch processing dialog represents a single execution of the algorithm. Cells in a row correspond to the parameter needed by the algorithm, which are not arranged one above the other, as in the normal single–execution dialog, but horizontally in that row.

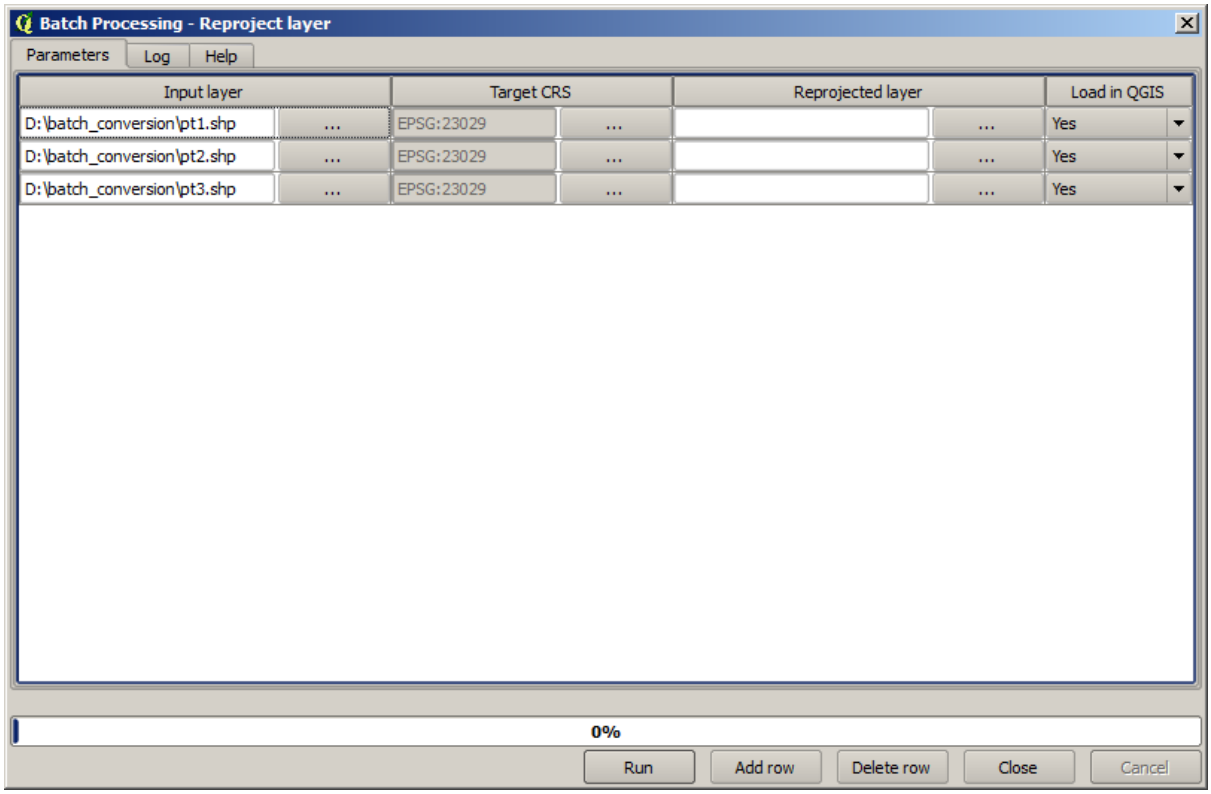
Defining the batch process to run is done by filling the table with the corresponding values, and the dialog itself contains several tools to make this task easier.

Let's start filling the fields one by one. The first column to fill is the *Input layer* one. Instead of entering the names of each one of the layers we want to process, you can select all of them and let the dialog put one in each row. Click on the button in the upper-left cell, and in the file selection dialog that will popup, select the three files to reproject. Since only one of them is needed for each row, the remaining ones will be used to fill the rows underneath.



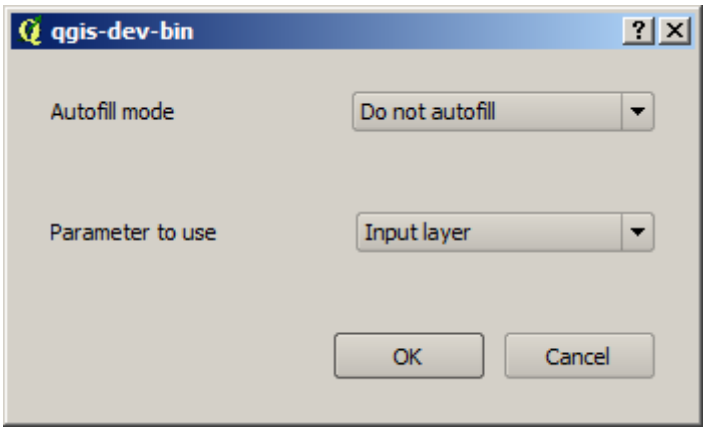
The default number of rows is 3, which is exactly the number of layers we have to convert, but if you select more layers, new rows will be added automatically. If you want to fill the entries manually, you can add more rows using the *Add row* button.

We are going to convert all those layers to the EPSG:23029 CRS, so we have to select that CRS in the second field. We want the same on for all rows, but we do not have to do it for every single row. Instead, set that CRS for the first row (the one at the top) using the button in the corresponding cell, and then double click on the column header. That causes all the cells in the column to be filled using the value of the top cell.

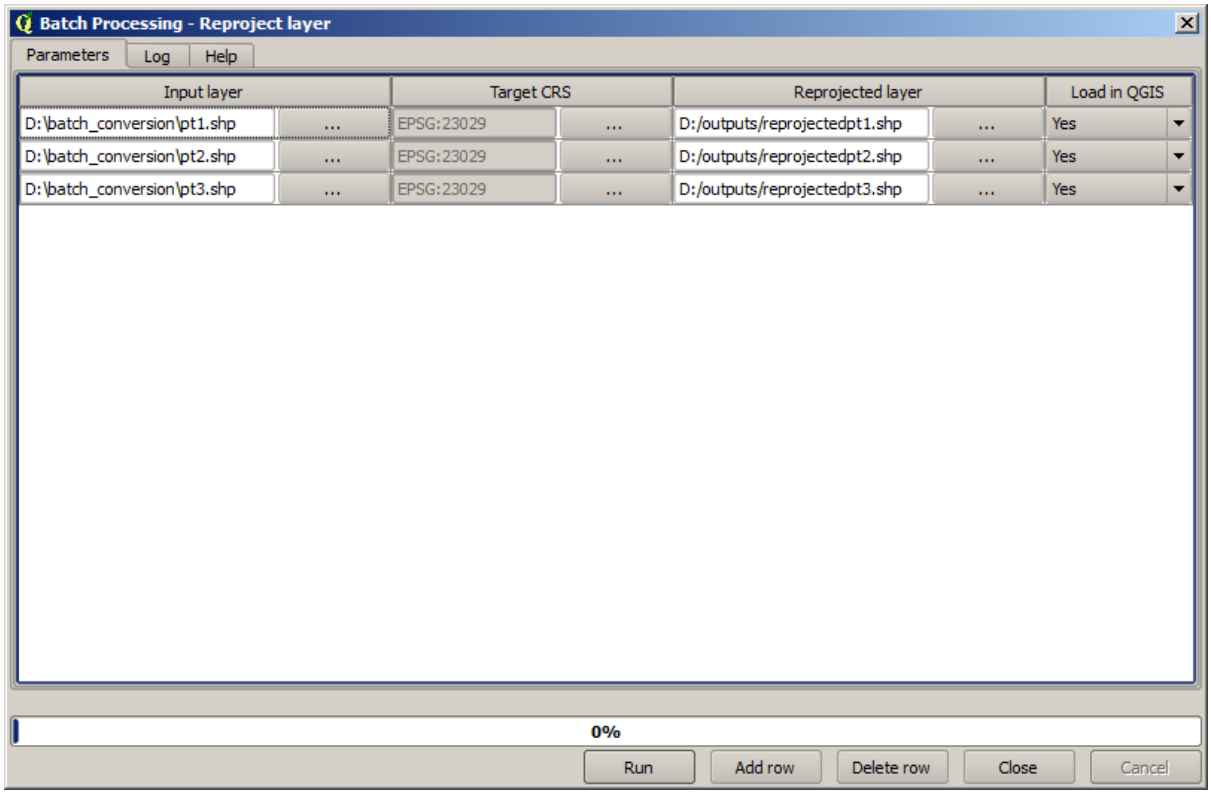


Finally, we have to select an output file for each execution, which will contain the corresponding reprojected layer. Once again, let's do it just for the first row. Click on the button in the upper cell and, in a folder where you want to put your output files, enter a filename (for instance, `reprojected.shp`).

Now, when you click *OK* on the file selection dialog, the file does not automatically gets written to the cell, but an input box like the following one is shown instead.



If you select the first option, only the current cell will be filled. If you select any of the other ones, all the rows below will be filled with a given pattern. In this case, we are going to select the *Fill with parameter value* option, and then the *Input Layer* value in the drop down menu below. That will cause the value in the *Input Layer* (that is, the layer name) to be added to the filename we have added, making each output filename different. The batch processing table should now look like this.



The last column sets whether or not to add the resulting layers to the current QGIS project. Leave the default *Yes* option, so you can see your results in this case.

Click on *OK* and the batch process will be run. If everything went fine, all your layers will have been processed, and 3 new layers would have been created.

17.27 Models in the batch processing interface

Aviso: Beware, this chapter is not well tested, please report any issue; images are missing

Nota: This lesson shows another example of the batch processing interface, but this time using a model instead of a built-in algorithm

Models are just like any other algorithm, and they can be used in the batch processing interface. To demonstrate that, here is a brief example that we can do using our already well-known hydrological model.

Make sure you have the model added to your toolbox, and then run it in batch mode. This is what the batch processing dialog should look like.

Aviso: todo: Add image

Add rows up to a total of 5. Select the DEM file corresponding to this lesson as the input for all of them. Then enter 5 different threshold values as shown next.

Aviso: todo: Add image

As you see the batch processing interface can be run not just to run the same process on different datasets but also on the same dataset with different parameters.

Click on *OK* and you should get 5 new layers with watersheds corresponding to the specified 5 threshold values.

17.28 Pré- e pós-execução de encaixe da script

Nota: This lesson shows how to use pre- and post-execution hooks, which allow to perform additional operations before and after actual processing.

Pre- and post-execution hooks are Processing scripts that run before and after actual data processing is performed. This can be used to automate tasks that should be performed whenever an algorithm is executed.

The syntax of the hooks is identical to the syntax of Processing scripts, see the corresponding chapter in the QGIS User Guide for more details.

In addition to all scripts features, in hooks you can use a special global variable named `alg`, which represents the algorithm that has just been (or is about to be) executed.

Here is an example post-execution script. By default, Processing stores analysis results in temporary files. This script will copy outputs to a specific directory, so they won't be deleted after closing QGIS.

```
import os
import shutil
from processing.core.outputs import OutputVector, OutputRaster, OutputFile

MY_DIRECTORY = '/home/alex/outputs'

for output in alg.outputs:
    if isinstance(output, (OutputVector, OutputRaster, OutputFile)):
        dirname = os.path.split(output.value)[0]
        shutil.copytree(dirname, MY_DIRECTORY)
```

In the first two lines we import the required Python packages: `os` — for path manipulations, e.g. extracting file name, and `shutil` — for various filesystem operations like copying files. In the third line we import Processing outputs. This will be explained in more detail later in this lesson.

Then we define a `MY_DIRECTORY` constant, which is the path to the directory where we want to copy analysis results.

At the end of the script, we have the main hook code. In the loop we iterate over all algorithm outputs and check if this output is a file-based output and can be copied. If so, we determine top-level directory in which output files are located and then copy all files to our directory.

To activate this hook we need to open the Processing options, find the entry named *Post-execution script file* in the *General* group, and specify the filename of the hook script there. the specified hook will be executed after each Processing algorithm.

In a similar way, we can implement pre-execution hooks. For example, let's create a hook to check input vectors for geometry errors.

```
from qgis.core import QgsGeometry, QgsFeatureRequest
from processing.core.parameters import ParameterVector

for param in alg.parameters:
```

(continues on next page)

(continuação da página anterior)

```
    if isinstance(param, ParameterVector):
        layer = processing.getObject(param.value)
        for f in layer.getFeatures(QgsFeatureRequest().setSubsetOfAttributes([])):
            errors = f.geometry().validateGeometry()
            if len(errors) > 0:
                progress.setInfo('One of the input vectors contains invalid␣
↪geometries!')
```

As in the previous example, first we import required QGIS and Processing packages.

Then we iterate over all the algorithm parameters and if a ParameterVector parameter is found, we get the corresponding vector layer object from it. We loop over all the features of the layer and check them for geometry errors. If at least one feature contains an invalid geometry, we print a warning message.

To activate this hook we need enter its filename in the *Pre-execution script file* option in the Processing configuration dialog. The hook will be executed before running any Processing algorithm.

17.29 Outros programas

Module contributed by Paolo Cavallini - Faunalia

Nota: This chapter shows how to use additional programs from inside Processing. To complete it, you must have installed, with the tools of your operating system, the relevant packages.

17.29.1 GRASS

GRASS is a free and open source GIS software suite for geospatial data management and analysis, image processing, graphics and maps production, spatial modeling, and visualization.

It is installed by default on Windows through the OSGeo4W standalone installer (32 and 64 bit), and it is packaged for all major Linux distributions.

17.29.2 R

R is a free and open source software environment for statistical computing and graphics.

It has to be installed separately, together with a few necessary libraries (**LIST**). To enable the use of R in QGIS, the *Processing R Provider* plugin must also be installed.

The beauty of Processing implementation is that you can add your own scripts, simple or complex ones, and they may then be used as any other module, piped into more complex workflows, etc.

Test some of the preinstalled examples, if you have R already installed (remember to activate R modules from the General configuration of Processing).

17.29.3 Others

LASTools is a set of mixed, free and proprietary commands to process and analyze LiDAR data. Availability in various operating system varies.

More tools are available through additional plugins, e.g.:

- LecoS: a suite for land cover statistics and landscape ecology
- lwgeom: formerly part of PostGIS, this library brings a few useful tools for geometry cleanup
- Animove: tools to analyse the home range of animals.

More will come.

17.29.4 Comparison among backends

Buffers and distances

Let's load `points.shp` and type `buf` in the filter of the Toolbox, then double click on:

- *Fixed distance buffer*: Distance 10000
- *Variable distance buffer*: Distance field SIZE
- *v.buffer.distance*: distance 10000
- *v.buffer.column*: bufcolumn SIZE
- *Shapes Buffer*: fixed value 10000 (dissolve and not), attribute field (with scaling)

See how speed is quite different, and different options are available.

Exercise for the reader: find the differences in geometry output between different methods.

Now, raster buffers and distances:

- first, load and rasterize the vector `rivers.shp` with *GRASS* ► *v.to.rast.value*; **beware:** cell size must be set to 100 m, otherwise the computation time will be enormous; resulting map will have 1 and NULLs
- same, with *SAGA* ► *Shapes to Grid* ► *COUNT* (resulting map: 6 to 60)
- then, *proximity* (value= 1 for GRASS, a list of rivers ID for SAGA), *r.buffer* with parameters 1000,2000,3000, *r.grow.distance* (the first of the two maps; the second will show the areas pertaining to each river, if done on the SAGA raster).

Dissolve

Dissolve features based on a common attribute:

- *GRASS* ► *v.dissolve* `municipalities.shp` on PROVINCIA
- *QGIS* ► *Dissolve* `municipalities.shp` on PROVINCIA
- *OGR* ► *Dissolve* `municipalities.shp` on PROVINCIA
- *SAGA* ► *Polygon Dissolve* `municipalities.shp` on PROVINCIA (**NB:** *Keep inner boundaries* must be unselected)

Nota: The last one is broken in SAGA <=2.10

Exercise for the reader: find the differences (geometry and attributes) between different methods.

17.30 Interpolation and contouring

Module contributed by Paolo Cavallini - Faunalia

Nota: This chapter shows how to use different backends to calculate different interpolations.

17.30.1 Interpolação

The project shows a gradient in rainfall, from south to north. Let's use different methods for interpolation, all based on vector `points.shp`, parameter `RAIN`:

Aviso: Set cell size to `500` for all analyses.

- *GRASS* ► *v.surf.rst*
- *SAGA* ► *Multilevel B-Spline Interpolation*
- *SAGA* ► *Inverse Distance Weighted* [Inverse distance to a power; Power: 4; Search radius: Global; Search range: all points]
- *GDAL* ► *Grid (Inverse Distance to a power)* [Power:4]
- *GDAL* ► *Grid (Moving average)* [Radius1&2: 50000]

Then measure variation among methods and correlate it with distance to points:

- *GRASS* ► *r.series* [Unselect Propagate NULLs, Aggregate operation: stddev]
- *GRASS* ► *v.to.rast.value* on `points.shp`
- *GDAL* ► *Proximity*
- *GRASS* ► *r.covar* to show the correlation matrix; check the significance of the correlation e.g. with http://vassarstats.net/rsig.html.

Thus, areas far from points will have less accurate interpolation.

17.30.2 Contour

Various methods to draw contour lines [always step= 10] on the *stddev* raster:

- *GRASS* ► *r.contour.step*
- *GDAL* ► *Contour*
- *SAGA* ► *Contour lines from grid* [**NB:** in some older SAGA versions, output shp is not valid, known bug]

17.31 Vector simplification and smoothing

Module contributed by Paolo Cavallini - Faunalia

Nota: This chapter shows how simplify vectors, and smooth out sharp corners.

Sometimes we need a simplified version of a vector, to have a smaller file size and get rid of unnecessary details. Many tools do this in a very rough way, and miss the adjacency and sometimes the topological correctness of polygons. GRASS is the ideal tool for this: being a topological GIS, adjacency and correctness are preserved even at very high simplification levels. In our case, we have a vector resulting from a raster, thus showing a «saw» pattern at borders. Applying a simplification results in straight lines:

- *GRASS* ► *v.generalize* [Maximal tolerance value: 30 m]

We can also do the reverse, and make a layer more complex, smoothing out sharp corners:

- *GRASS* ► *v.generalize* [method: chaiken]

Try to apply this second command both to original vector and to the one from the first analysis, and see the difference. Note that adjacency is not lost.

This second option can be applied e.g. to contour lines resulting from a coarse raster, to GPS tracks with sparse vertices, etc.

17.32 Planning a solar farm

Module contributed by Paolo Cavallini - Faunalia

Nota: This chapter shows how to use several criteria to locate the areas suitable for installing a photovoltaic power station

First of all, create an aspect map from DTM:

- *GRASS* ► *r.aspect* [Data type: int; cell size:100]

In GRASS, aspect is calculated in degrees, counterclockwise starting from East. To extract only South facing slopes (270 degrees +- 45), we can reclassify it:

- *GRASS* ► *r.reclass*

with the following rules:

```
225 thru 315 = 1 south
* = NULL
```

You can use the text file `reclass_south.txt` provided. Note that with these simple text files we can create also very complex reclassifications.

We want to build a large farm, so we select only large (> 100 ha) contiguous areas:

- *GRASS* ► *r.reclass.greater*

Finally, we convert to a vector:

- *GRASS* ► *r.to.vect* [Feature type: area; Smooth corners: yes]

Exercise for the reader: repeat the analysis, replacing GRASS commands with analogous from other programs.

17.33 Utilize as scripts R no Processing

Module contributed by Matteo Ghetta - funded by Scuola Superiore Sant'Anna

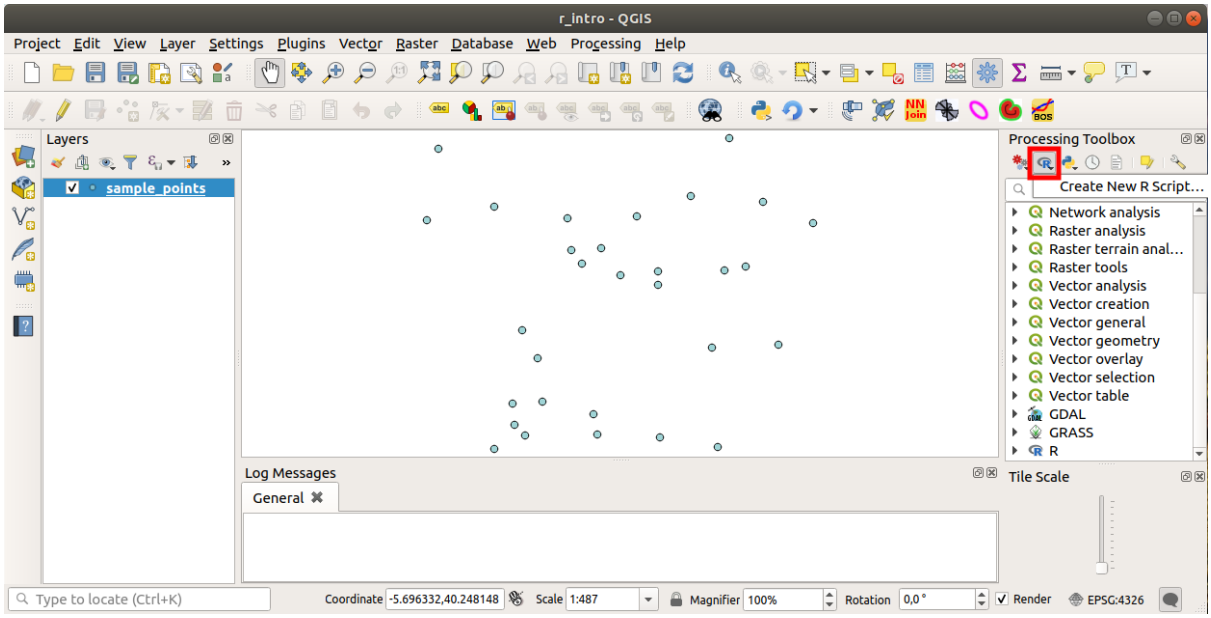
Processing (with the `Processing R Provider` plugin) makes it possible to write and run R scripts inside QGIS.

Aviso: R has to be installed on your computer and the PATH has to be correctly set up. Moreover Processing just calls the external R packages, it is not able to install them. So be sure to install external packages directly in R. See the related chapter in the user manual.

Nota: If you have *package* problems, it may be related to missing *mandatory* packages required by Processing, like `sp`, `rgdal` and `raster`.

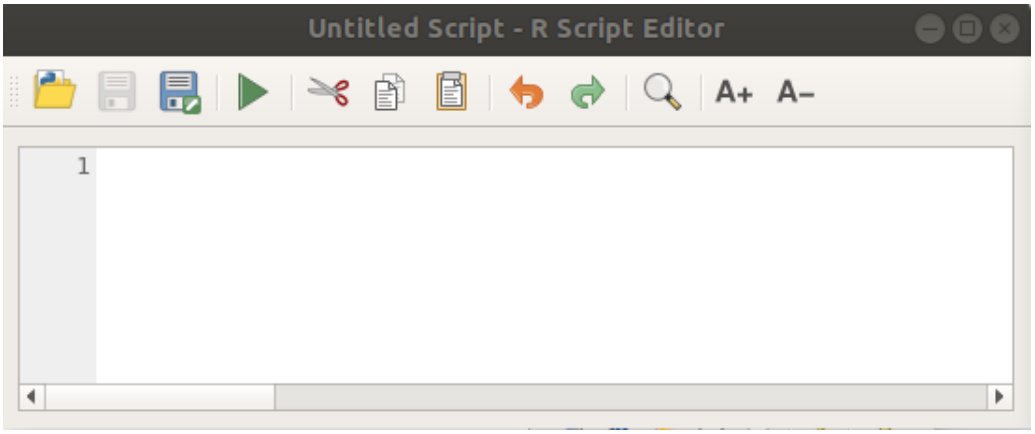
17.33.1 Adicionar scripts

Adding a script is simple. The easiest way is to open the Processing toolbox and choose *Create new R script…* from the R menu (labelled with an R icon) at the top of the Processing Toolbox. You can also create the script in for instance a text editor and save it in your R scripts folder (`processing/rscripts`). When it has been saved there, it will be available for editing by right-clicking on the script name in the processing toolbox and then choose *Edit Script…*).



Nota: If you cannot see R in Processing, you have to activate it in *Processing* ► *Options* ► *Providers*

It opens a *script editor window* in which you have to specify some parameters before you can add the script body.



17.33.2 Creating plots

In this tutorial we are going to create a **boxplot** of a vector layer field.

Open the `r_intro.qgs` QGIS project under the `exercise_data/processing/r_intro/` folder.

Script parameters

Open the editor and start writing at the beginning of it.

You **must** specify some parameters **before** the script body:

1. The name of the group (*plots* in this case) in which you want to put your script (if the group does not exist, it will be created):

   ```
   ##plots=group
   ```

   You will find your script in the **plots** R group in the Processing toolbox.

2. You have to tell Processing that you want to display a plot (in this example):

   ```
   ##showplots
   ```

   You will then find a link to the plot in the **Result Viewer** panel (can be turned on / off in *View* ► *Panels* and with *Processing* ► *Results Viewer*).

3. You also need to tell Processing about your input data. In this example we want to create a plot from a field of a vector layer:

   ```
   ##Layer=vector
   ```

   Processing now knows that the input is a vector. The name *Layer* is not important, what matters is the **vector** parameter.

4. Finally, you have to specify the input field of the vector layer (using the name you have provided above - *Layer*):

   ```
   ##X=Field Layer
   ```

   Processing now knows that you need a field of *Layer*, and that you will call it **X**.

5. It is also possible to define the name of your script using `name`:

   ```
   ##My box plot script=name
   ```

   If not defined, the file name will be used as the name of the script.

Script body

Now that you have set up the *heading* of the script you can add the function:

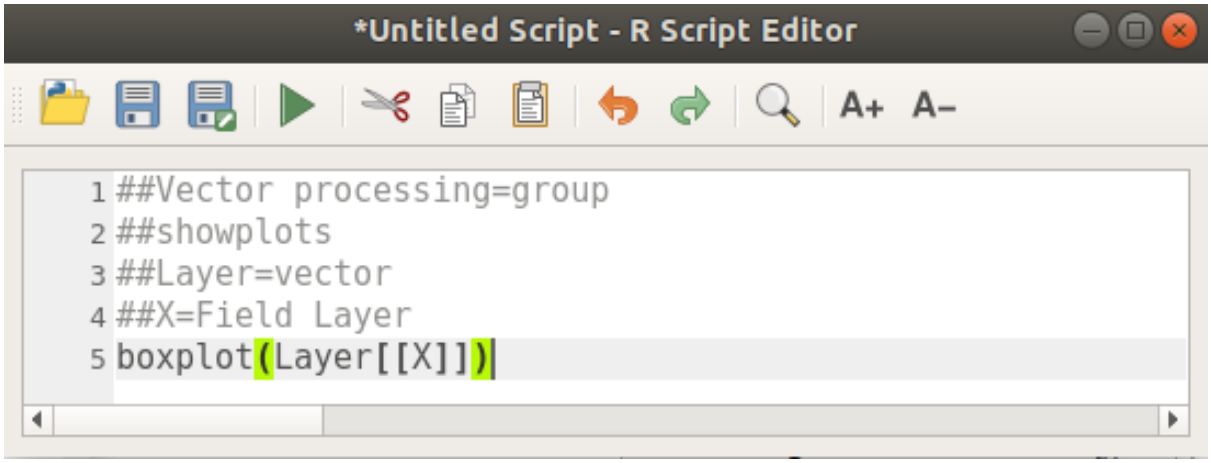
```
boxplot(Layer[[X]])
```

boxplot is the name of the R function, the parameter **Layer** is the name that you have defined for the input dataset and **X** is the name you have defined for the field of that dataset.

> **Aviso:** The parameter **X** has to be within double square brackets (`[[]]`).

The final script looks like this:

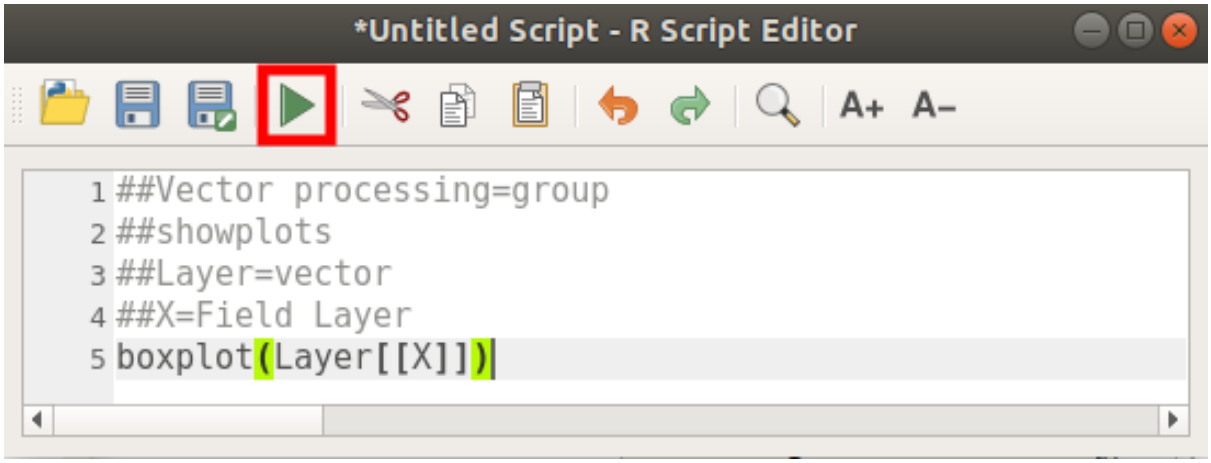
```
##Vector processing=group
##showplots
##Layer=vector
##X=Field Layer
boxplot(Layer[[X]])
```



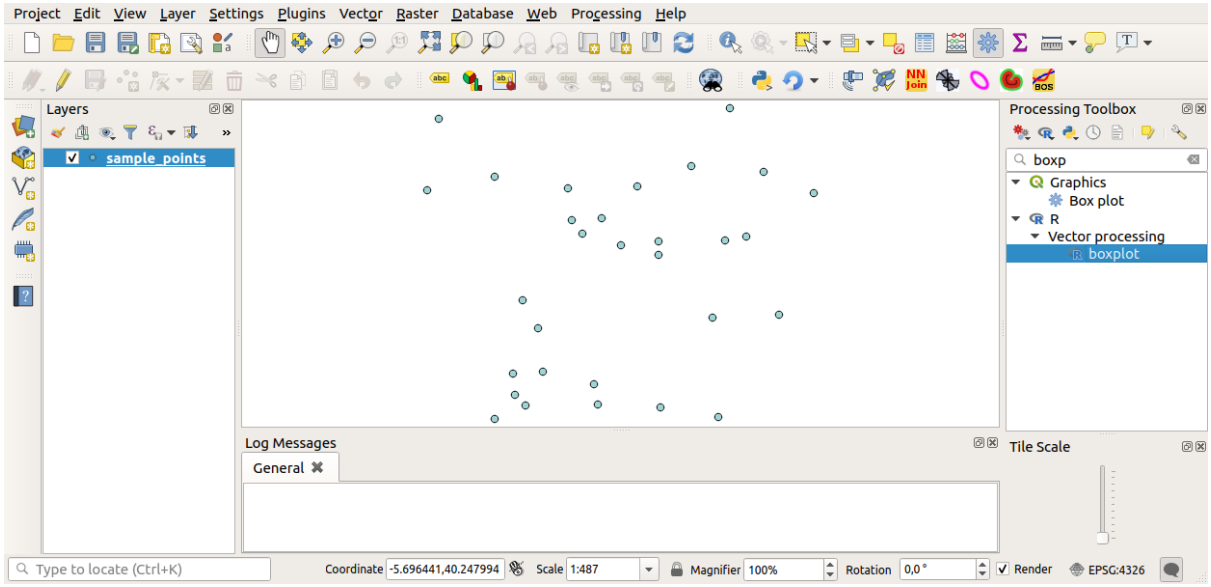
Save the script in the default path suggested by Processing (processing/rscripts). If you have not defined a `name` in the script heading, the file name you choose will become the name of the script in the Processing toolbox.

Nota: You can save the script wherever you like, but Processing will then not be able to include it in the processing toolbox automatically, so you have to upload it manually.

Now just run it using the button on the top of the editor window:



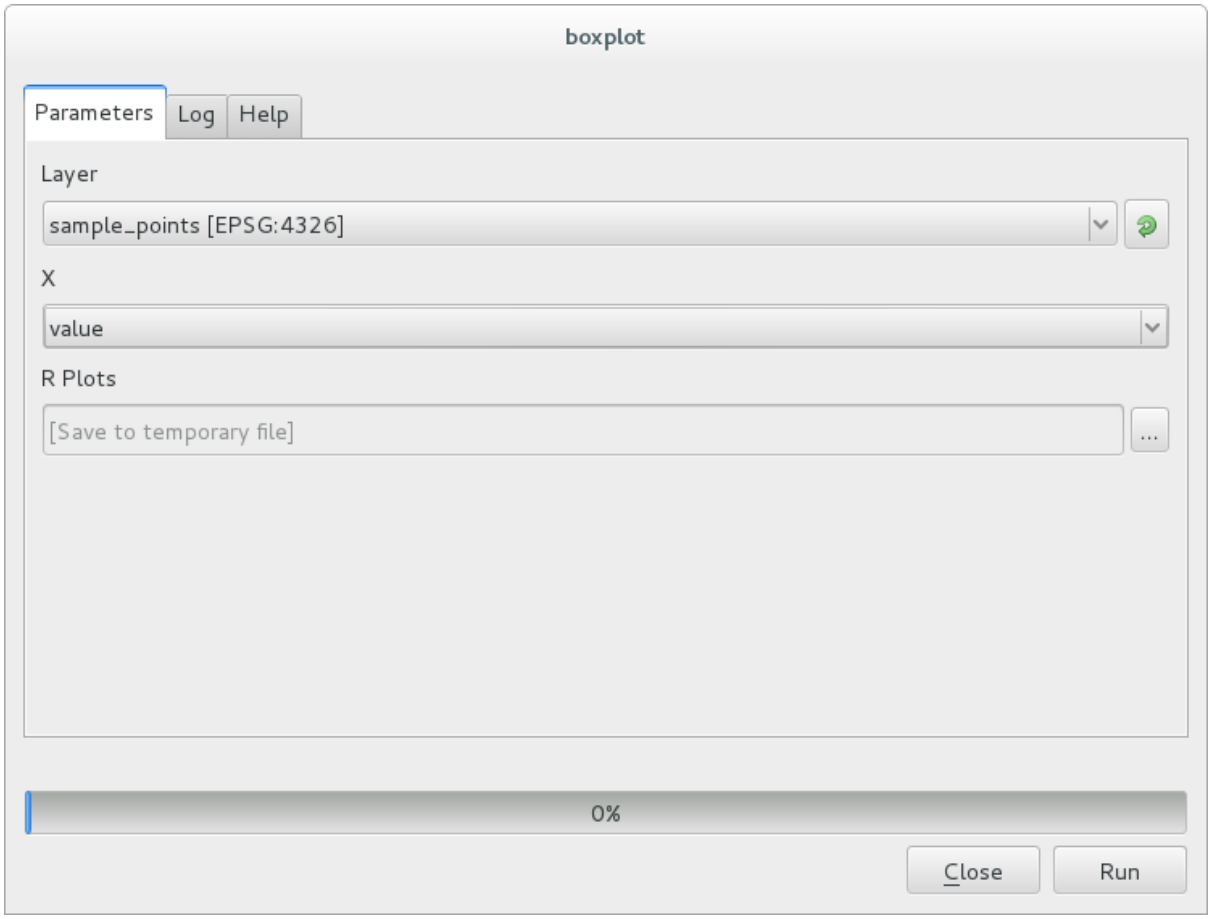
Once the editor window has been closed, use the text box of Processing to find your script:



You can now fill the parameters required in the Processing algorithm window:

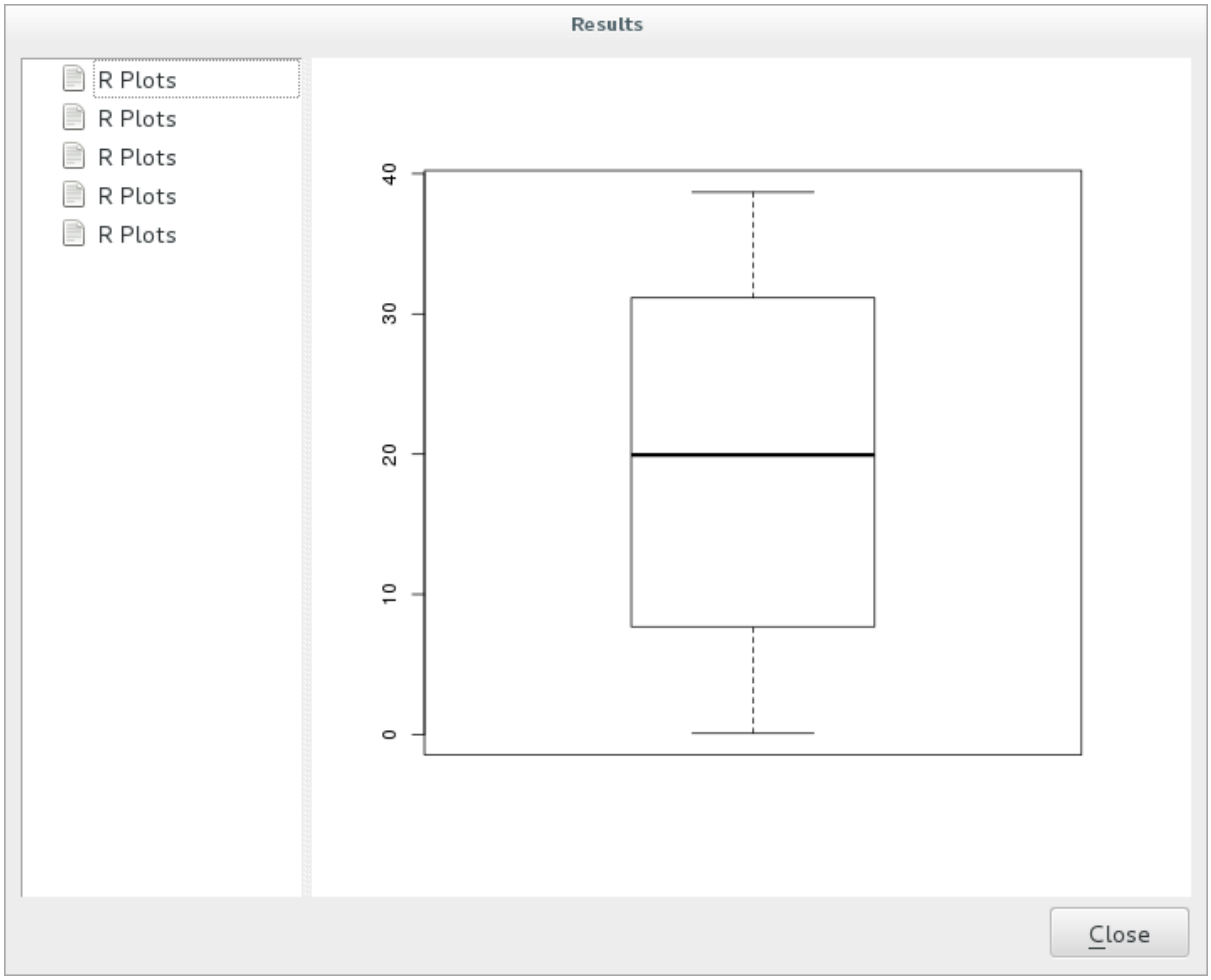
- for **Layer** choose *sample_points*
- for the **X** field choose *value*

Click on **Run**.



The **Result window** should be automatically opened, if not, just click on *Processing* ► *Result Viewer*....

Click on the link in the viewer and you will see:



Nota: You can open, copy and save the image by right clicking on the plot.

17.33.3 Create a vector

You can also create a vector layer and have it automatically loaded into QGIS.

The following example has been taken from the `Random sampling grid` script that can be found in the online collection of R scripts (the scripts in this online collection can be found in https://github.com/qgis/QGIS-Processing/tree/master/rscripts).

The aim of this exercise is to create a random point vector layer using an input vector layer to restrict the extent using the `spsample` function of the `sp` package.

Script parameters

As before we have to set some parameters before the script body:

1. Specify the name of the group in which you want to put your script, in this case *Point pattern analysis*:

```
##Point pattern analysis=group
```

2. Define an input parameter (a vector layer) that will constrain the placement of the random points:

```
##Layer=vector
```

3. Set an input parameter for the number of points that are going to be created (`Size`, with a default value of `10`):

```
##Size=number 10
```

Nota: Since a default value (`10`) is defined, the user can change this number or can leave the parameter without a number.

---

4. Specify that there is an output vector layer (called `Output`):

```
##Output=output vector
```

Script body

Now you can add the body of the function:

1. Use the `spsample` function:

```
pts=spsample(Layer, Size, type="random")
```

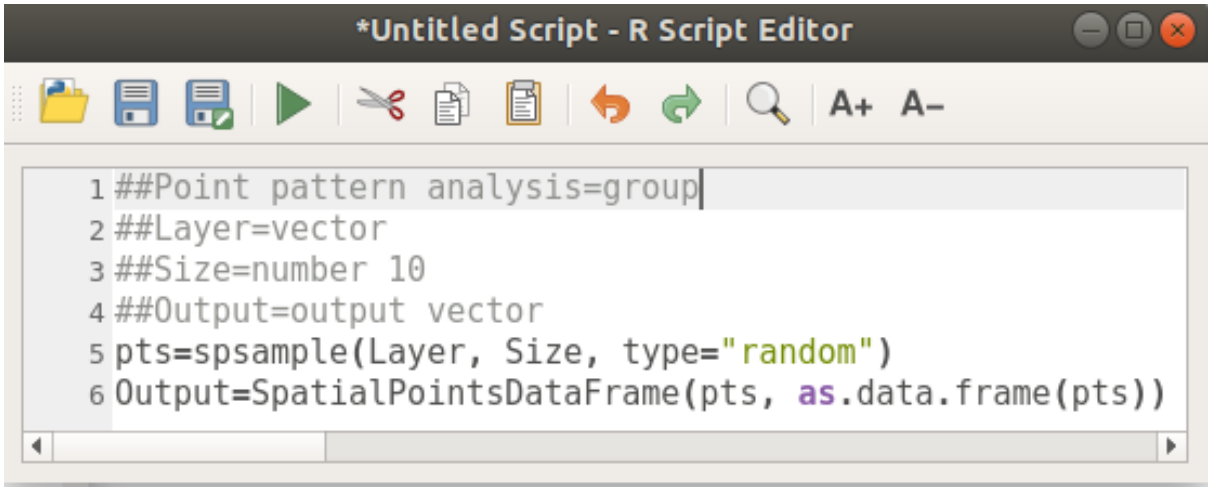
The function uses the *Layer* to constrain the placement of the points (if it is a line layer, a points will have to be on one of the lines in the layer, if it is a polygon layer, a point will have to be within a polygon). The number of points is taken from the *Size* parameter. The sampling method is *random*.

2. Generate the output (the `Output` parameter):

```
Output=SpatialPointsDataFrame(pts, as.data.frame(pts))
```

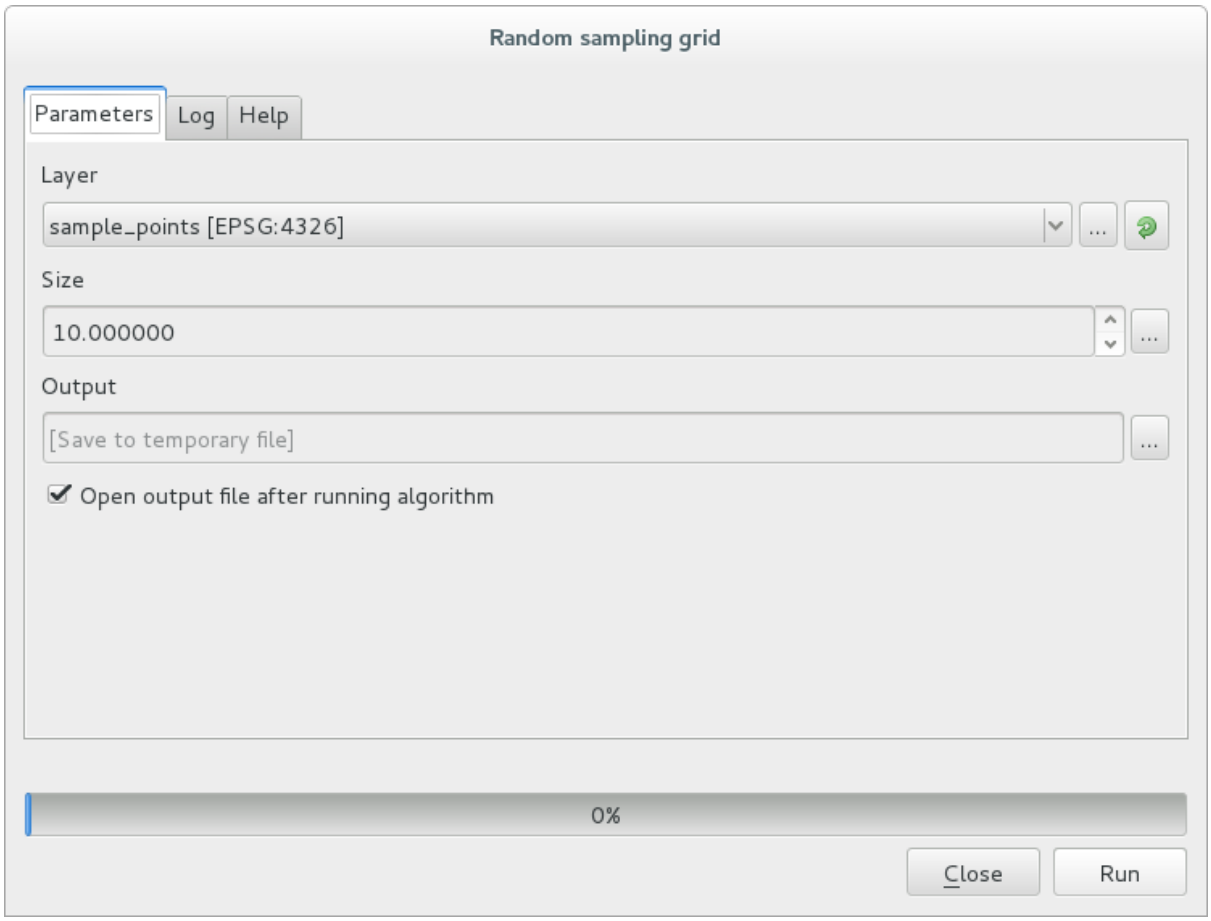
The final script looks like this:

```
##Point pattern analysis=group
##Layer=vector
##Size=number 10
##Output=output vector
pts=spsample(Layer, Size, type="random")
Output=SpatialPointsDataFrame(pts, as.data.frame(pts))
```



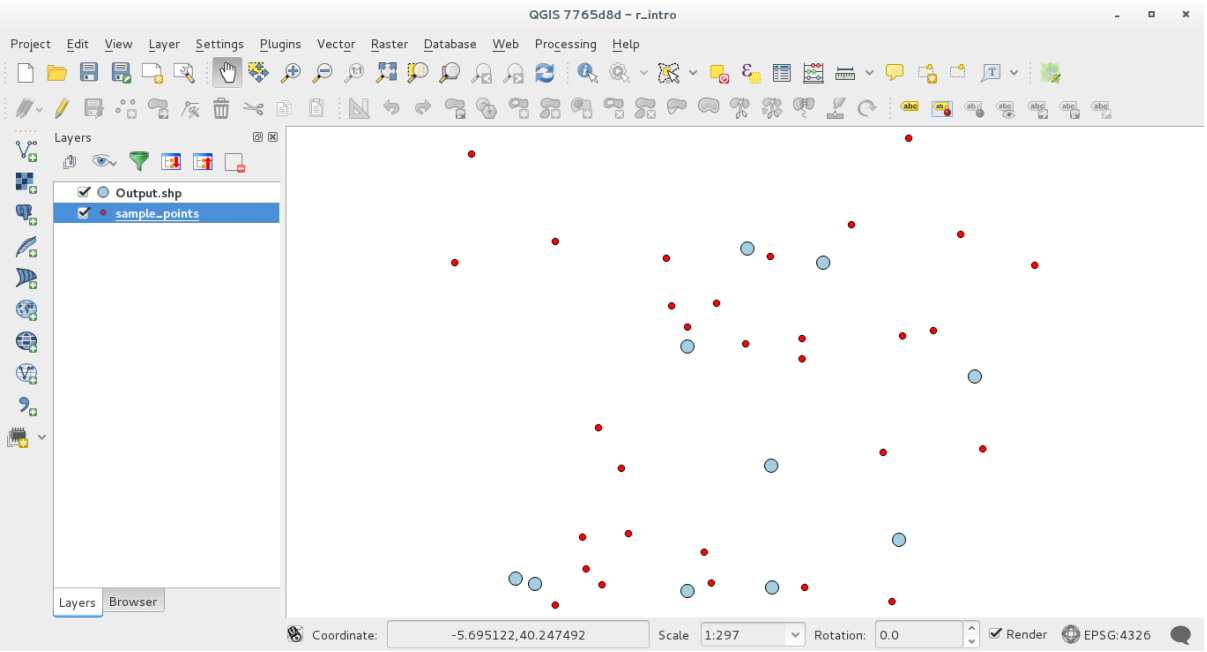
Save it and run it, clicking on the run button.

In the new window type in the right parameters:



and click on run.

The result layer will be added to the table of contents and its points will be displayed on the map canvas:



17.33.4 Text and graph output from R - syntax

Processing (with the `Processing R Provider plugin`) uses special syntax to get the results out of R:

- `>` before your command, as in `>lillie.test(Layer[[Field]])` means the result should be sent to R output (Result viewer)
- `+` after a plot enables overlay plots. For example `plot(Layer[[X]], Layer[[Y]]) + abline(h=mean(Layer[[X]]))`

17.34 Predicting landslides

Module contributed by Paolo Cavallini - Faunalia

Nota: This chapter shows how to create an oversimplified model to predict the probability of landslides.

First, we calculate slope (choose among various backends; the interested reader can calculate the difference between the outputs):

- *GRASS* ► *r.slope*
- *SAGA* ► *Slope, Aspect, Curvature*
- *GDAL Slope*

Then we create a model of predicted rainfall, based on the interpolation of rainfall values at meteo stations:

- *GRASS* ► *v.surf.rst* (resolution: 500 m)

The probability of a landslide will be very roughly related to both rainfall and slope (of course a real model will use more layers, and appropriate parameters), let's say `(rainfall * slope )/100`:

- *SAGA* ► *Raster calculator* rain, slope: `(a*b)/100` (or: *GRASS* ► *r.mapcalc*)
- then let's calculate what are the municipalities with the greatest predicted risk of rainfall: *SAGA* ► *Raster statistics with polygons* (the parameters of interest are *Maximum* and *Mean*)

CAPÍTULO 18

Module: Using Spatial Databases in QGIS

Neste módulo, irá aprender como usar bases de dados espaciais no QGIS para gerir, visualizar e manipular dados na base de dados assim como fazer análise através de pesquisas. Iremos usar principalmente PostgreSQL e PostGIS (que foram falado em secções anteriores), mas os mesmos conceitos aplicam-se a outras implementações de bases de dados incluindo o Spatialite.

18.1 Lesson: Working with Databases in the QGIS Browser

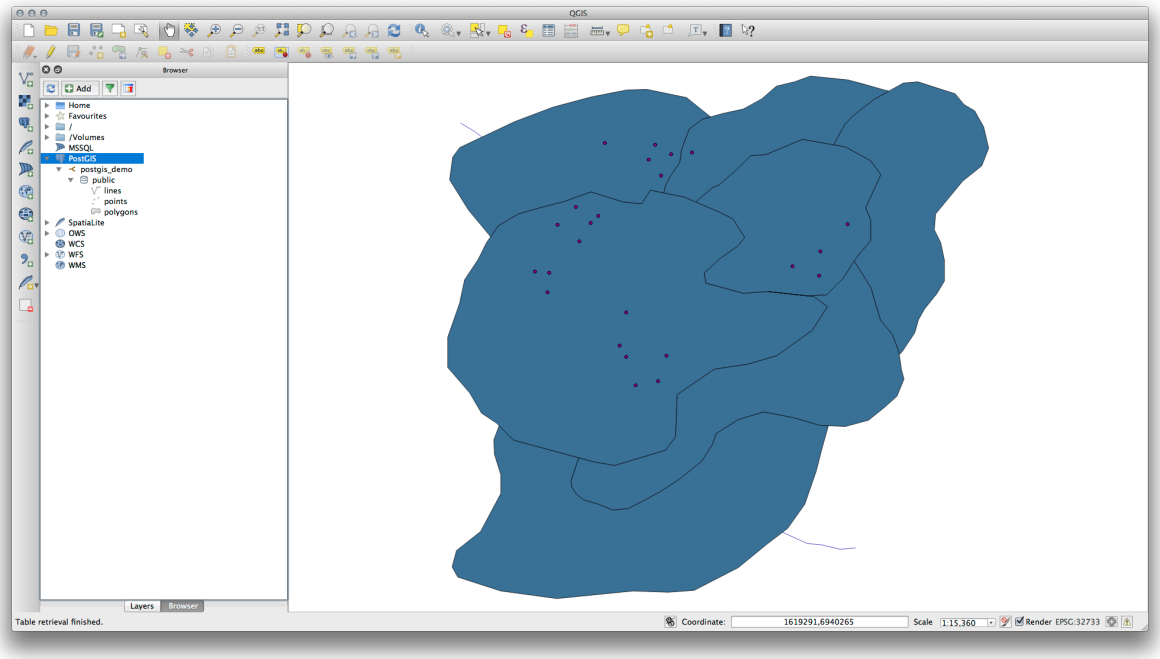
In the previous 2 modules we looked at the basic concepts, features and functions of relational databases as well as extensions that let us store, manage, query and manipulate spatial data in a relational database. This section will dive deeper into how to effectively use spatial databases in QGIS.

The goal for this lesson: To learn how to interact with spatial databases using the QGIS Browser interface.

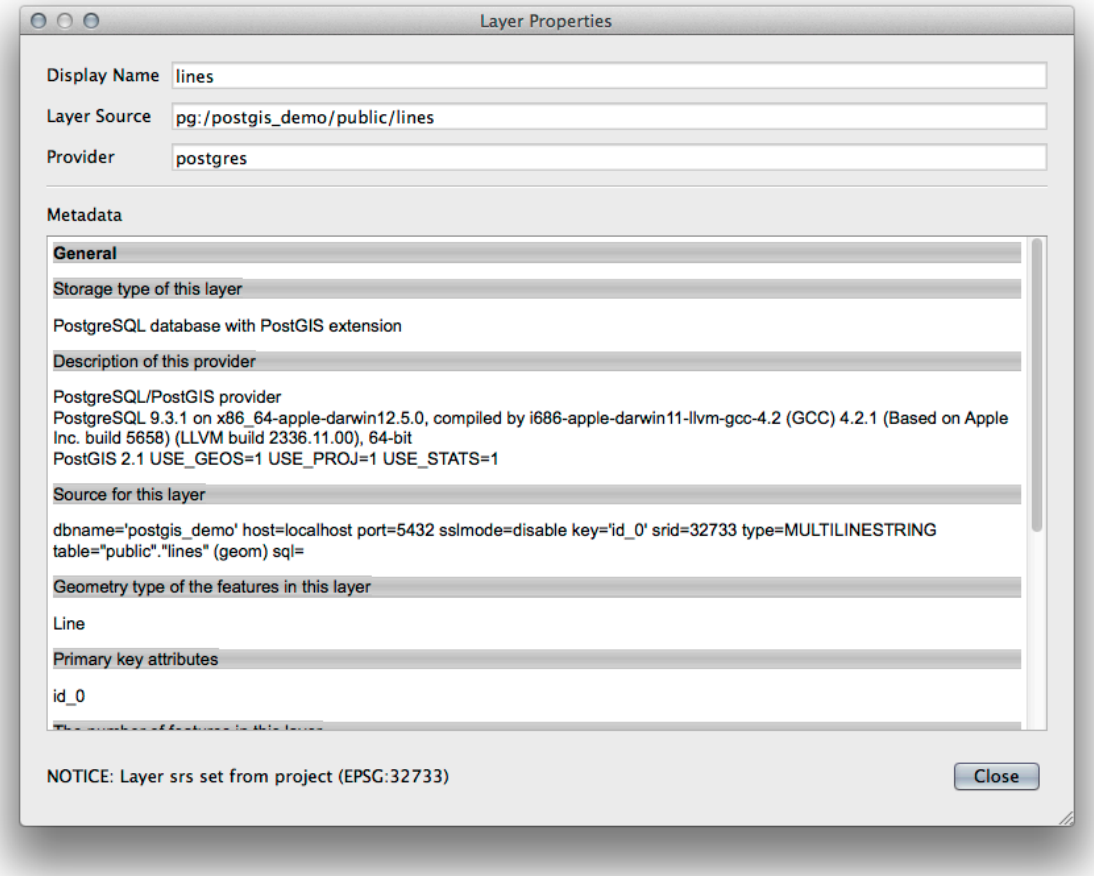
18.1.1 ??? Follow Along: Adding Database Tables to QGIS using the Browser

We have already briefly looked at how to add tables from a database as QGIS layers, now lets look at this in a bit more detail and look at the different ways this can be done in QGIS. Lets start by looking at the new Browser interface.

- Start a new empty map in QGIS.
- Open the Browser by clicking the *Browser* tab at the bottom of the *Layer Panel*
- Open the PostGIS portion of the tree and you should find your previously configured connection available (you may need to click the Refresh button at the top of the browser window).



- Double clicking on any of the table/layers listed here will add it to the Map Canvas.
- Right Clicking on a table/layer in this view will give you a few options. Click on the *Properties* item to look at the properties of the layer.



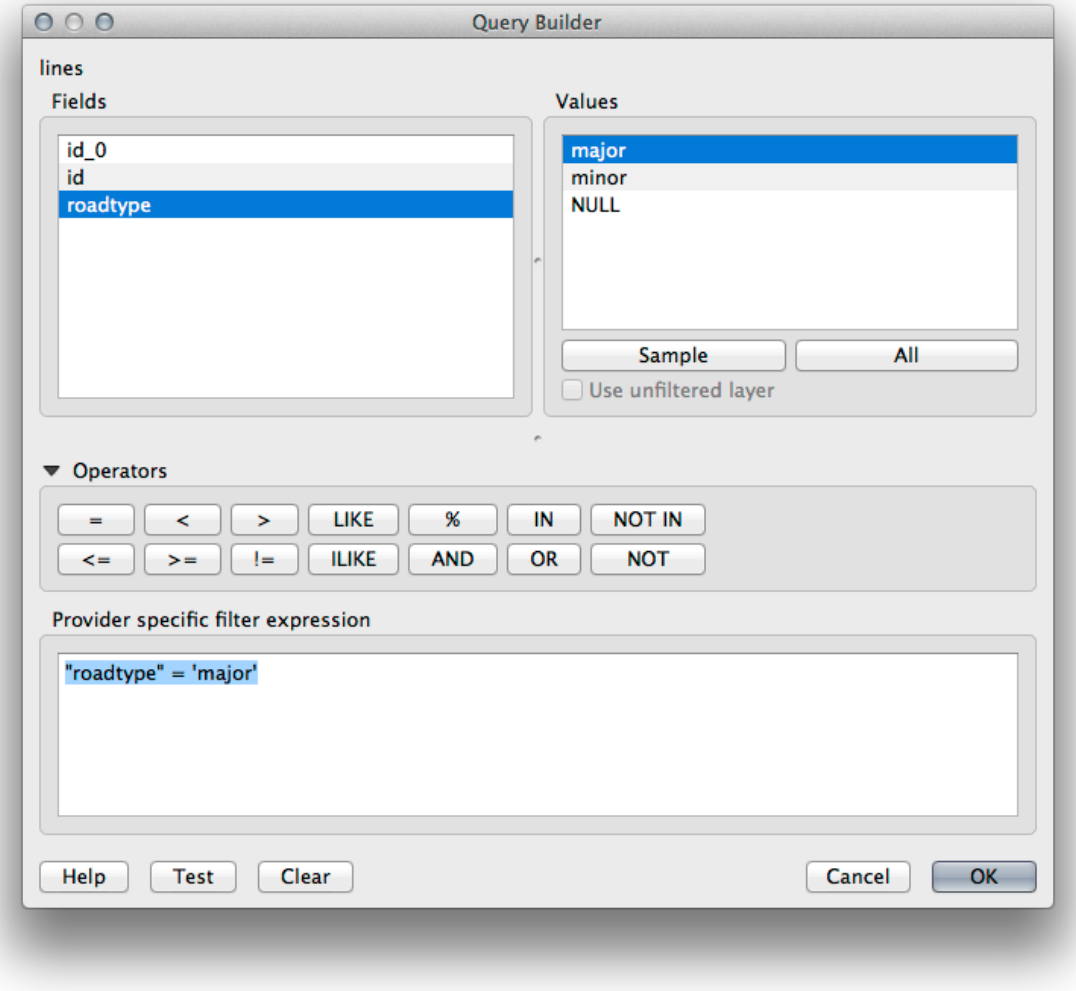
Nota: Of course you can also use this interface to connect to PostGIS databases hosted on a server external to your workstation. Right clicking on the PostGIS entry in the tree will allow you to specify connection parameters for a new connection.

18.1.2 ??? Follow Along: Adding a filtered set of records as a Layer

Now that we have seen how to add an entire table as a QGIS layer it might be nice to learn how to add a filtered set of records from a table as a layer by using queries that we learned about in previous sections.

- Start a new empty map with no layers
- Click the *Add PostGIS Layers* button or select *Layer* ► *Add PostGIS Layers* from the menu.
- In the *Add PostGIS Table(s)* dialog that comes up, connect to the `postgis_demo` connection.
- Expand the `public schema` and you should find the three tables we were working with previously.
- Click the `lines` layer to select it, but instead of adding it, click the *Set Filter* button to bring up the *Query Builder* dialog.
- Construct the following expression using the buttons or by entering it directly:

```
"roadtype" = 'major'
```



- Click *OK* to complete editing the filter and click *Add* to add the filtered layer to your map.
- Rename the `lines` layer in the tree to `roads_primary`.

You will notice that only the Primary Roads have been added to your map rather than the entire layer.

18.1.3 In Conclusion

You have seen how to interact with spatial databases using the QGIS Browser and how to add layers to your map based on a query filter.

18.1.4 What's Next?

Next you'll see how to work with the DB Manager interface in QGIS for a more complete set of database management tasks.

18.2 Lesson: Using DB Manager to work with Spatial Databases in QGIS

We have already seen how to perform many database operations with QGIS as well as with other tools, but now it's time to look at the DB Manager tool which provides much of this same functionality as well as more management oriented tools.

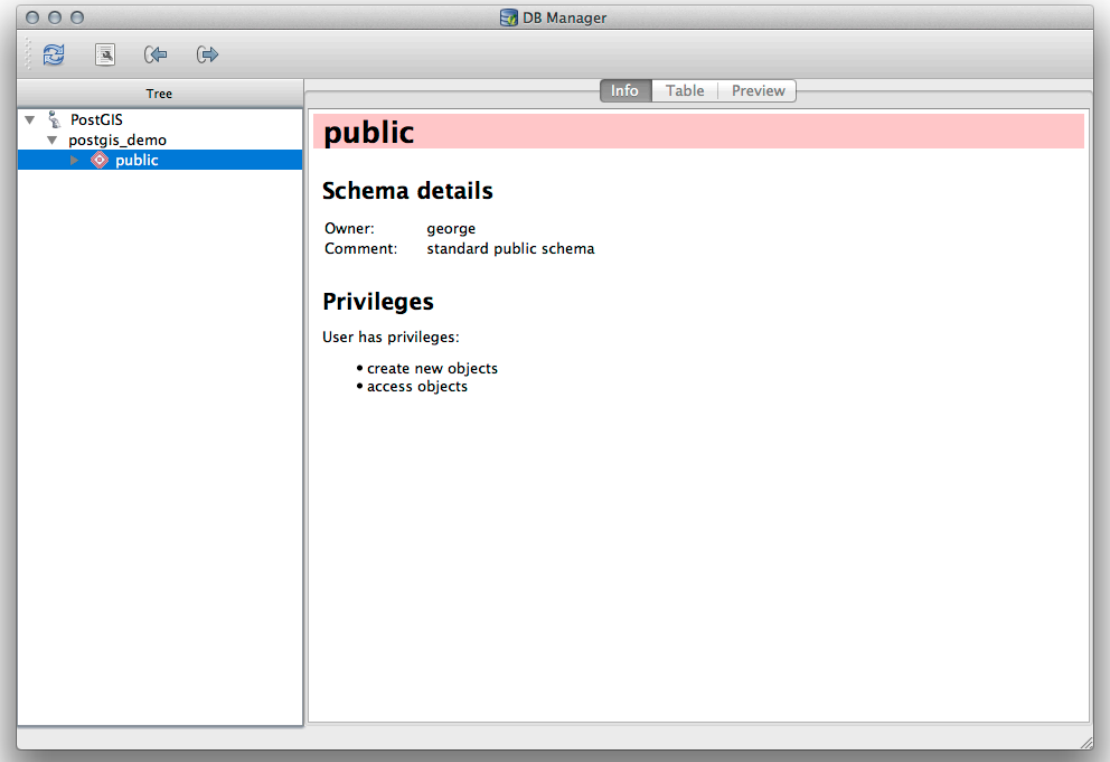
The goal for this lesson: To learn how to interact with spatial databases using the QGIS DB Manager.

18.2.1 ??? Follow Along: Managing PostGIS Databases with DB Manager

You should first open the DB Manager interface by selecting *Database* ► *DB Manager* ► *DB Manager* on the menu or by selecting the DB Manager icon on the toolbar.

You should already see the previous connections we have configured and be able to expand the `myPG` section and its `public` schema to see the tables we have worked with in previous sections.

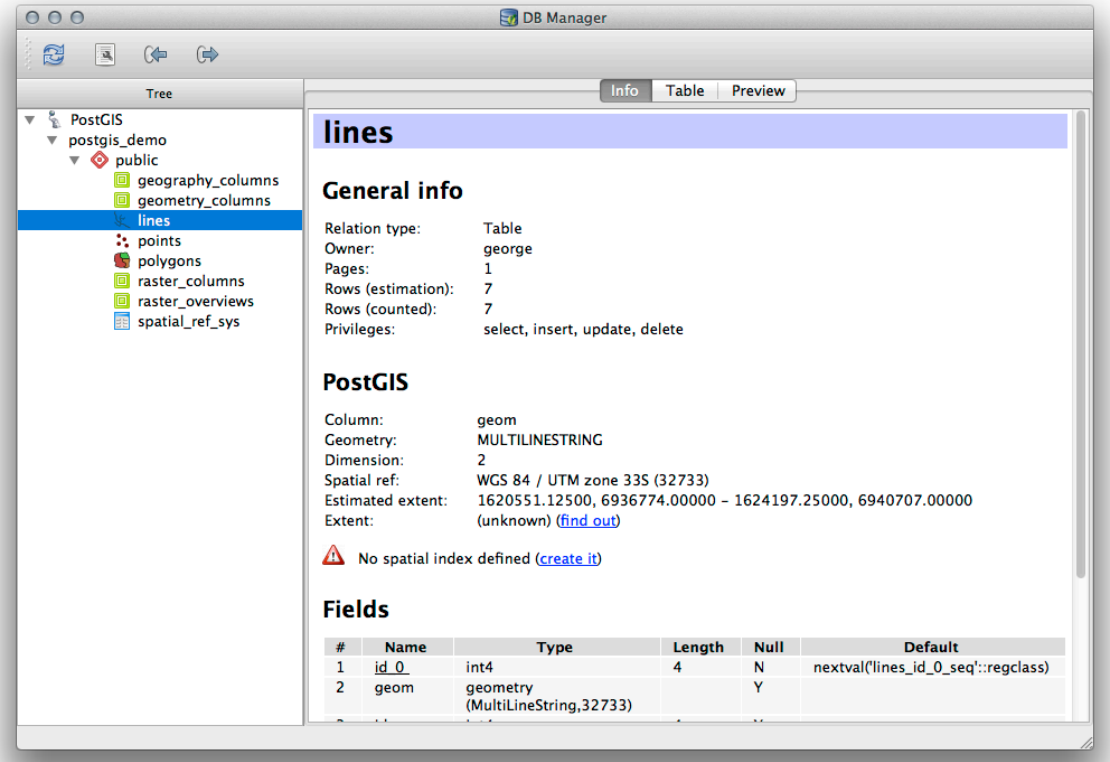
The first thing you may notice is that you can now see some metadata about the Schemas contained in your database.



Schemas are a way of grouping data tables and other objects in a PostgreSQL database and a container for permissions and other constraints. Managing PostgreSQL schemas is beyond the scope of this manual, but you can find more information about them in the PostgreSQL documentation on Schemas. You can use the DB Manager to create new Schemas, but will need to use a tool like pgAdmin III or the command line interface to manage them effectively.

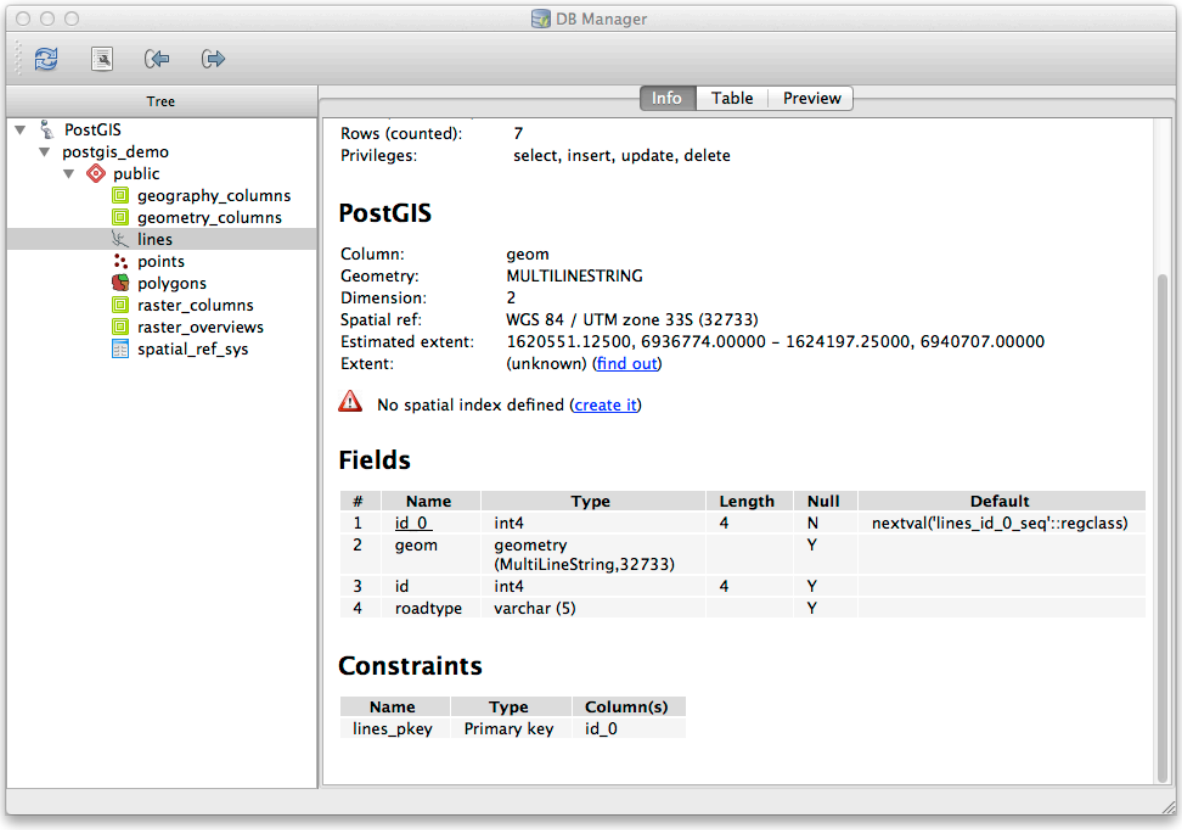
DB Manager can also be used to manage the tables within your database. We have already looked at various ways to create and manage tables on the command line, but now lets look at how to do this in DB Manager.

First, its useful to just look at a table's metadata by clicking on its name in tree and looking in the *Info* tab.

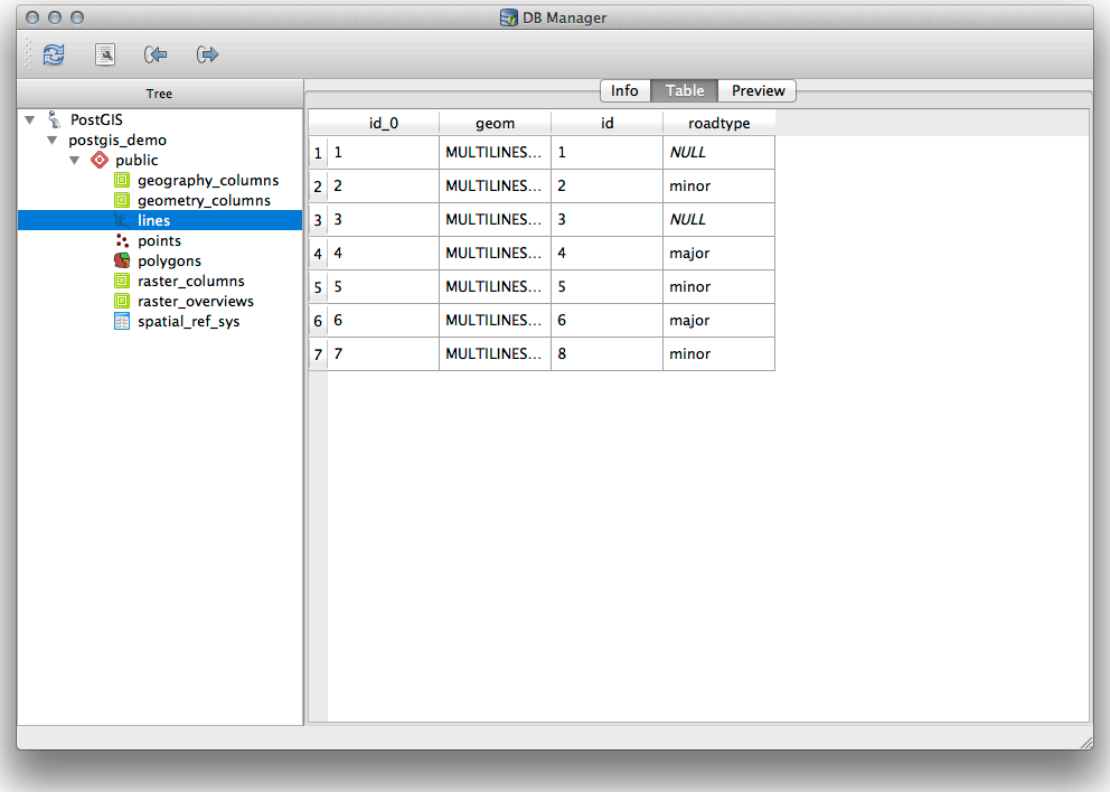


In this panel you can see the *General Info* about the table as well the information that the PostGIS extension maintains about the geometry and spatial reference system.

If you scroll down in the *Info* tab, you can see more information about the *Fields*, *Constraints* and *Indexes* for the table you are viewing.



Its also very useful to use DB Manager to simply look at the records in the database in much the same way you might do this by viewing the attribute table of a layer in the Layer Tree. You can browse the data by selecting the *Table* tab.

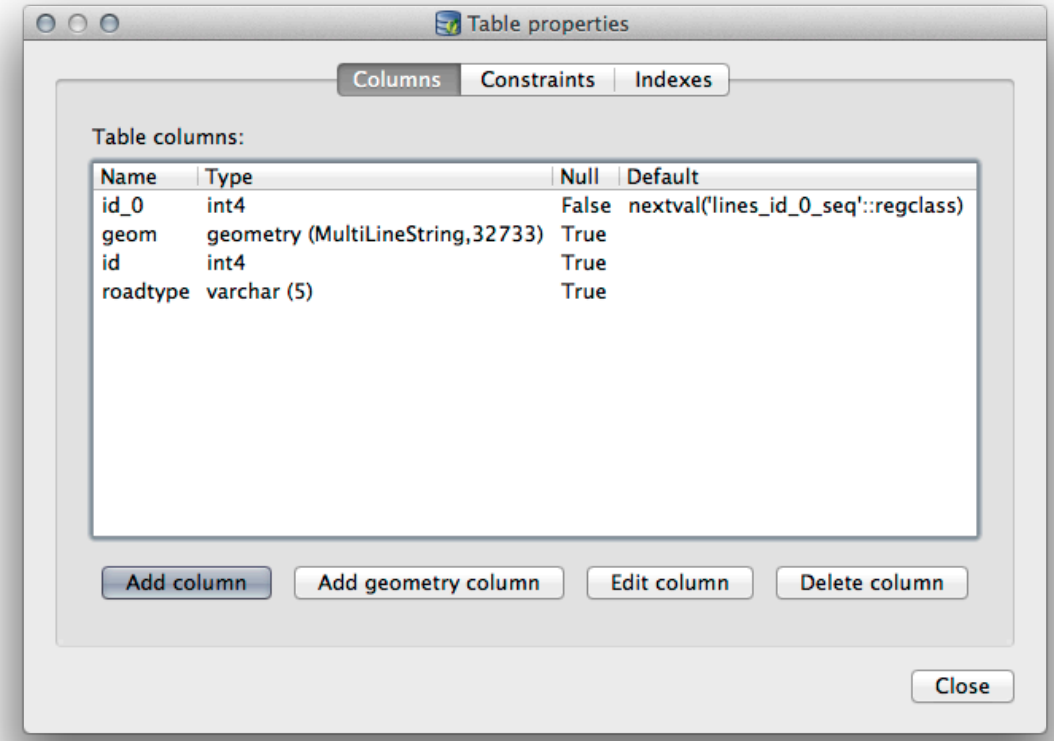


There is also a *Preview* tab which will show you the layer data in a map preview.

Right-clicking on a layer in the tree and clicking *Add to Canvas* will add this layer to your map.

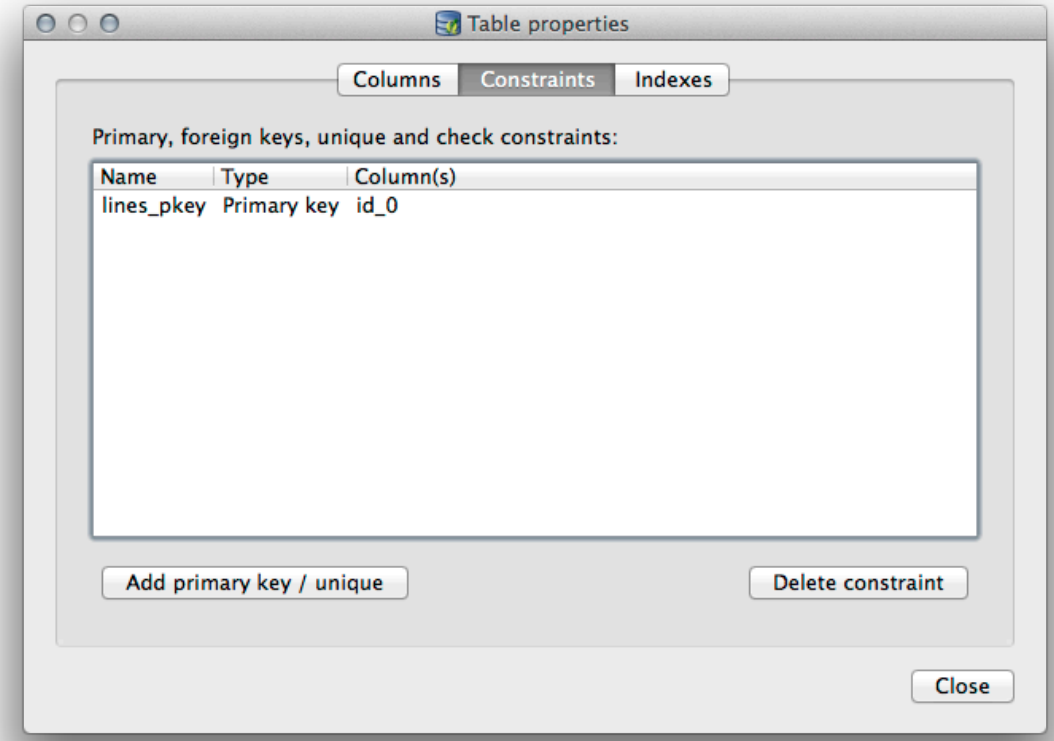
So far we have only been viewing the database its schemas and tables and their metadata, but what if we wanted to alter the table to add an additional column perhaps? DB Manager allows you to do this directly.

1. Select the table you want to edit in the tree
2. Select *Table* ► *Edit Table* from the menu, to open the *Table Properties* dialog.

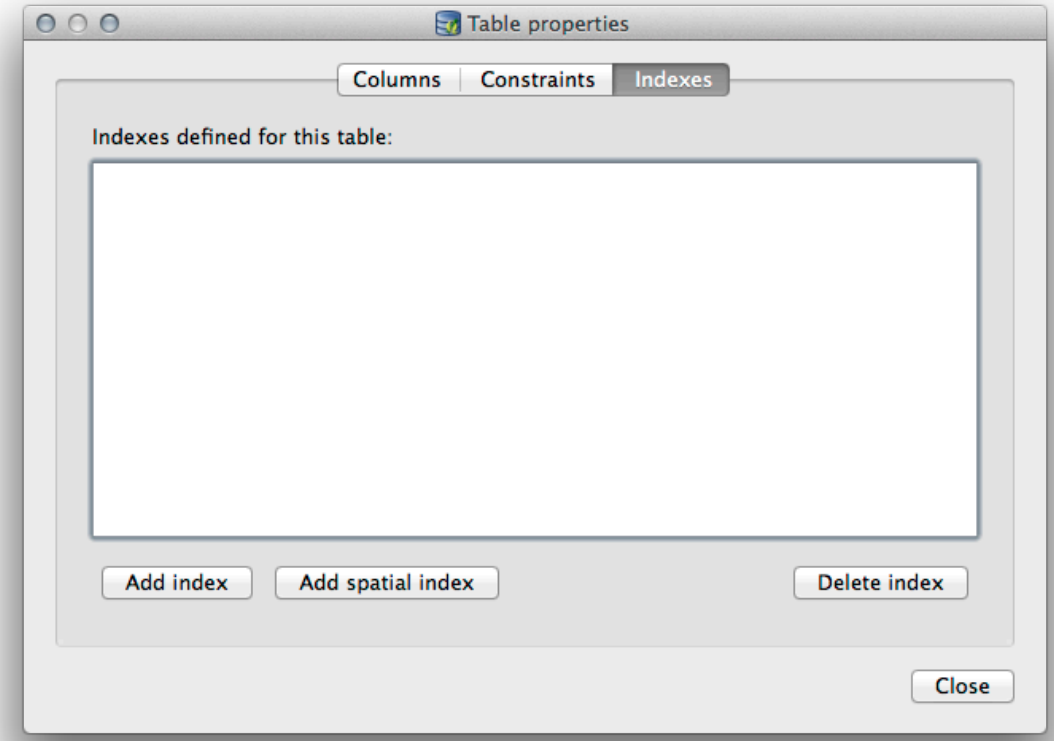


You can use this dialog to Add Columns, Add geometry columns, edit existing columns or to remove a column completely.

Using the *Constraints* tab, you can manage which fields are used as the primary key or to drop existing constraints.



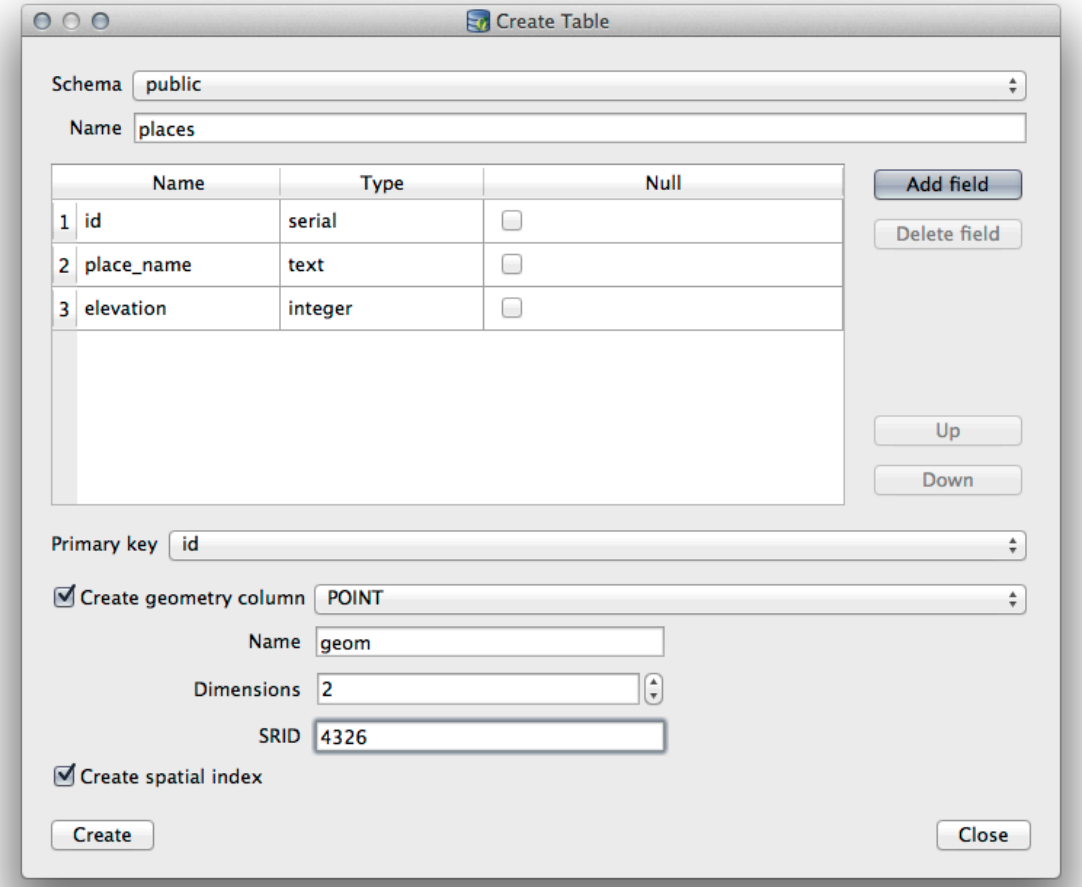
The *Indexes* tab can be used to add and delete both spatial and normal indexes.



18.2.2 Follow Along: Creating a New Table

Now that we have gone through the process of working with existing tables in our database, let's use DB Manager to create a new table.

1. If it is not already open, open the DB Manager window, and expand the tree until you see the list of tables already in your database.
2. From the menu select *Table –> Create Table* to bring up the Create Table dialog.
3. Use the default `Public` schema and name the table `places`.
4. Add the `id`, `place_name`, and `elevation` fields as shown below
5. Make sure the `id` field is set as the primary key.
6. Click the checkbox to *Create geometry column* and make sure it is set to a `POINT` type and leave it named `geom` and specify `4326` as the *SRID*.
7. Click the checkbox to *Create spatial index* and click *Create* to create the table.



8. Dismiss the dialog letting you know that the table was created and click *Close* to close the Create Table Dialog.

You can now inspect your table in the DB Manager and you will of course find that there is no data in it. From here you can *Toggle Editing* on the layer menu and begin to add places to your table.

18.2.3 ??? Follow Along: Basic Database Administration

The DB Manager will also let you do some basic database administration tasks. It is certainly not a substitute for a more complete database administration tool, but it does provide some functionality that you can use to maintain your database.

Database tables can often become quite large and tables which are being modified frequently can end up leaving around remnants of records that are no longer needed by PostgreSQL. The *VACUUM* command takes care of doing a kind of garbage collection to compact and optional analyze your tables for better performance.

Let us take a look at how we can perform a *VACUUM ANALYZE* command from within DB Manager.

1. Select one of your tables in the DB Manager Tree
2. Select *Table* ► *Run Vacuum Analyze* from the menu

PostgreSQL will now perform the operation. Depending on how big your table is, this may take some time to complete.

You can find more information about the VACUUM ANALYZE process in the PostgreSQL Documentation on VACUUM ANALYZE.

18.2.4 ??? Follow Along: Executing SQL Queries with DB Manager

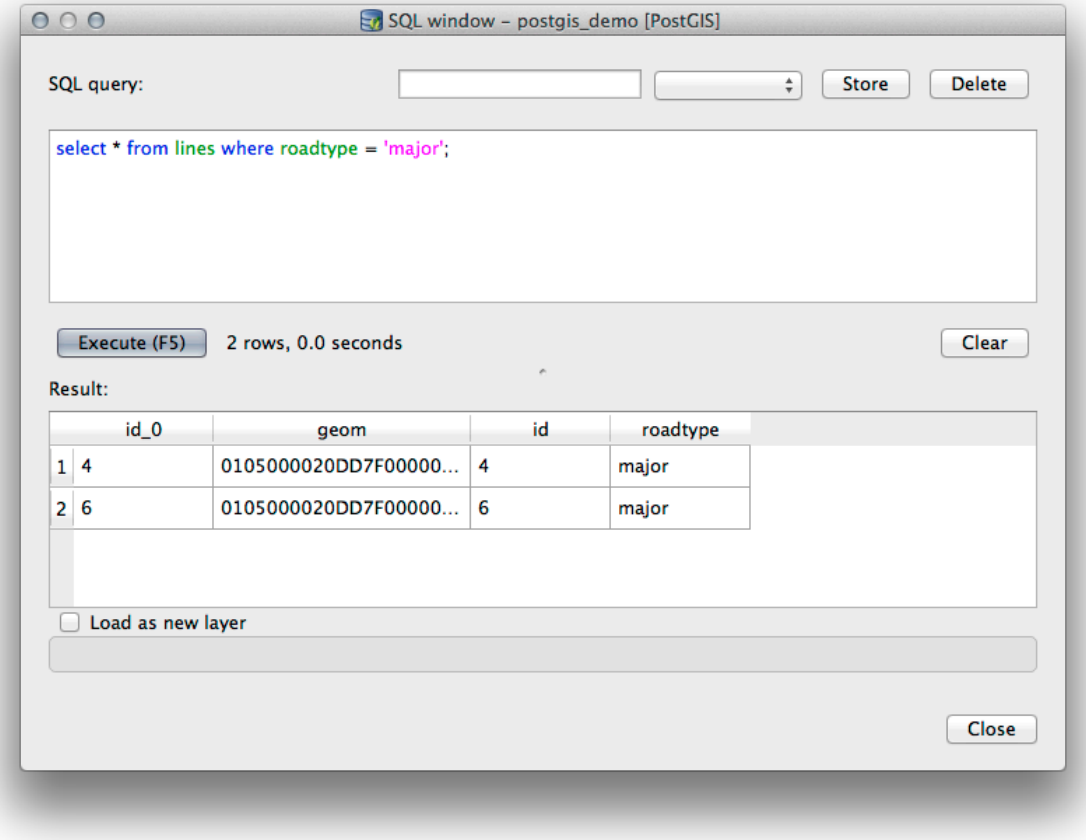
DB Manager also provides a way for you to write queries against your database tables and to view the results. We have already seen this type of functionality in the *Browser* panel, but lets look at it again here with DB Manager.

1. Select the `lines` table in the tree.
2. Select the *SQL window* button in the DB Manager toolbar.

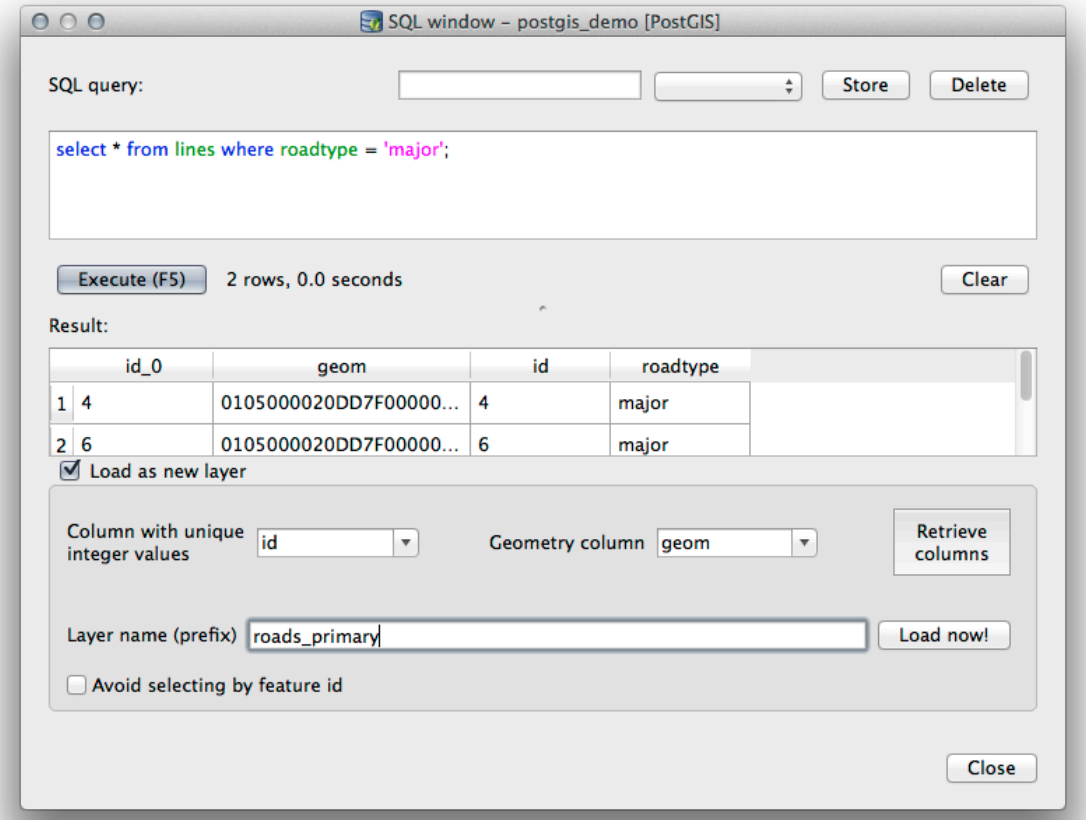
3. Compose the following *SQL query* in the space provided:

```
select * from lines where roadtype = 'major';
```

4. Click the *Execute (F5)* button to run the query.
5. You should now see the records that match in the *Result* panel.



6. Click the checkbox for *Load as new layer* to add the results to your map.
7. Select the `id` column as the *Column with unique integer values* and the `geom` column as the *Geometry column*.
8. Enter `roads_primary` as the *Layer name (prefix)*.
9. Click *Load now!* to load the results as a new layer into your map.



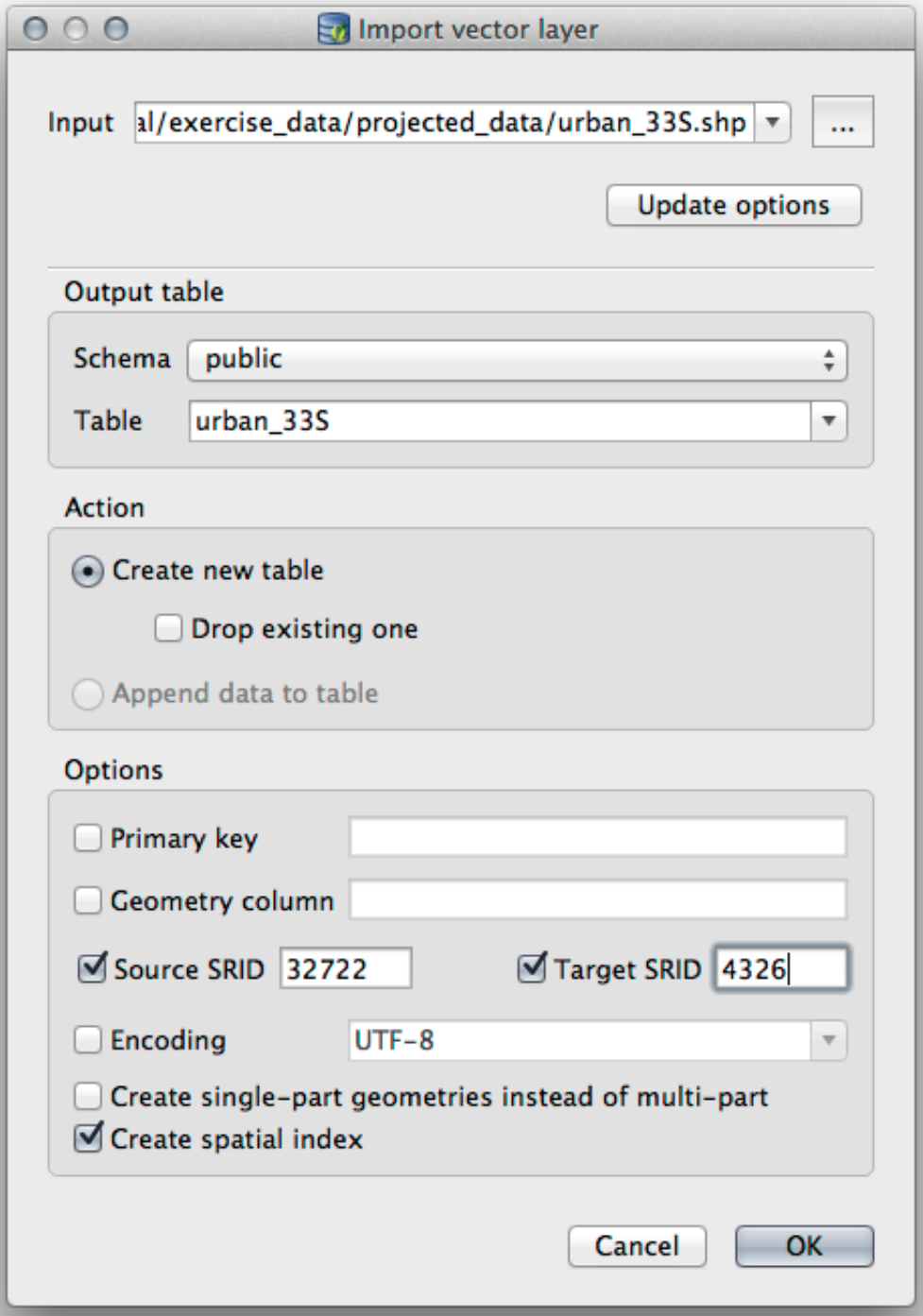
The layers that matched your query are now displayed on your map. You can of course use this query tool to execute any arbitrary SQL command including many of the ones we looked at in previous modules and sections.

18.2.5 Importing Data into a Database with DB Manager

We have already looked at how to import data into a spatial database using command line tools, so now let's learn how to use DB Manager to do imports.

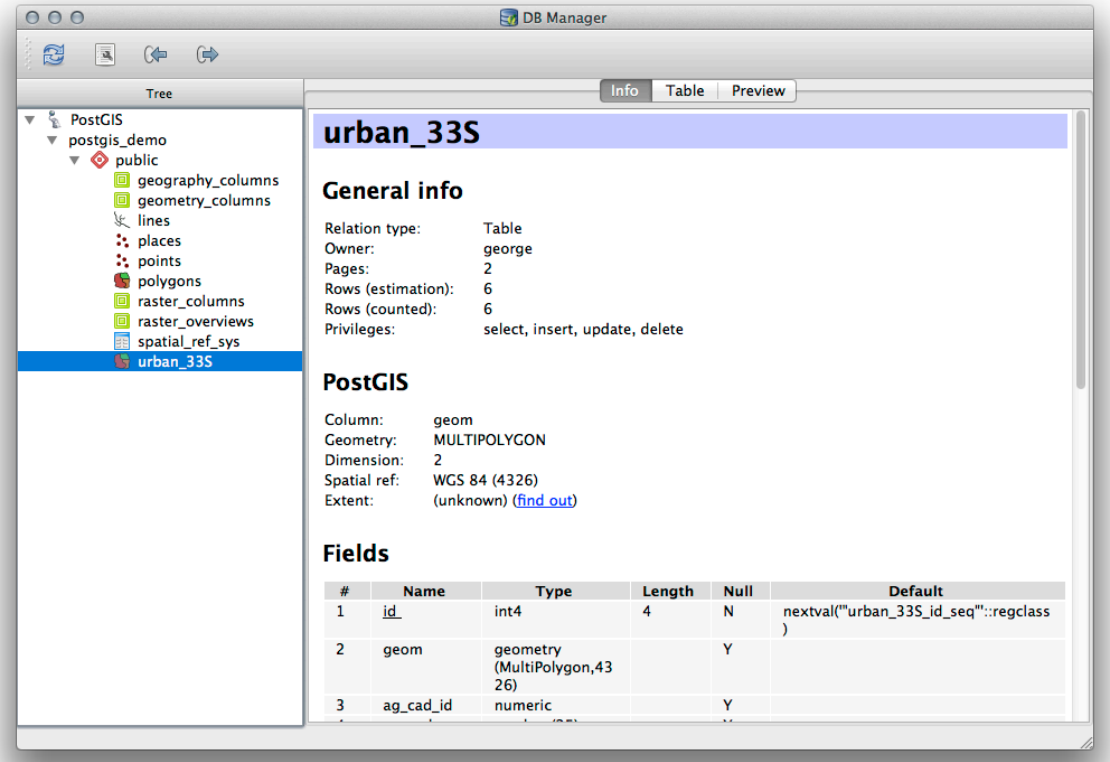
1. Click the *Import layer/file* button on the toolbar in the DB Manager dialog.

2. Select the `urban_33S.shp` file from `exercise_data/projected_data` as the input dataset
3. Click the *Update Options* button to pre-fill some of the form values.
4. Make sure that the *Create new table* option is selected
5. Specify the *Source SRID* as `32722` and the *Target SRID* as `4326`
6. Enable the checkbox to *Create Spatial Index*
7. Click *OK* to perform the import



8. Dismiss the dialog letting you know that the import was successful
9. Click the *Refresh* button on the DB Manager Toolbar

You can now inspect the table in your database by clicking on it in the Tree. Verify that the data has been reprojected by checking that the *Spatial ref:* is listed as `WGS 84 (4326)`.

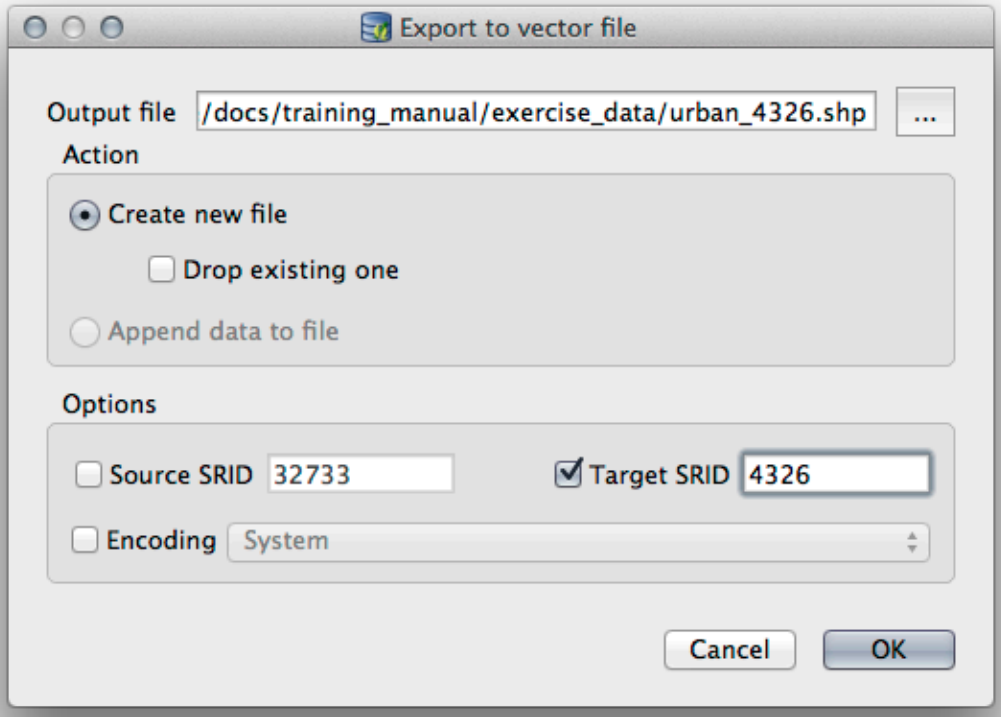


Right clicking on the table in the Tree and a selecting *Add to Canvas* will add the table as a layer in your map.

18.2.6 Exporting Data from a Database with DB Manager

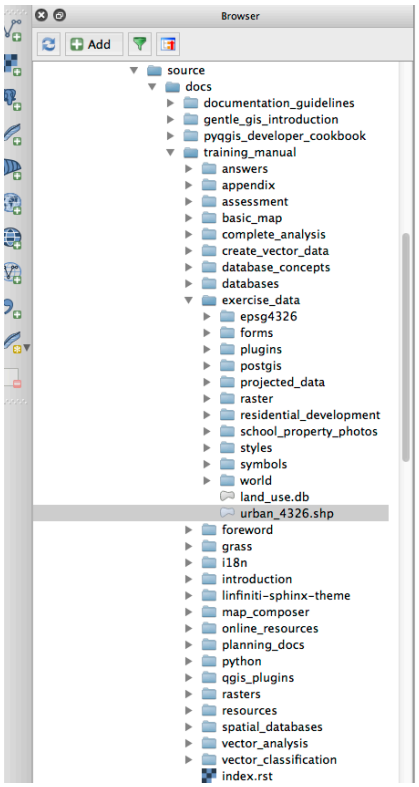
Of course DB Manager can also be used to export data from your spatial databases, so lets take a look at how that is done.

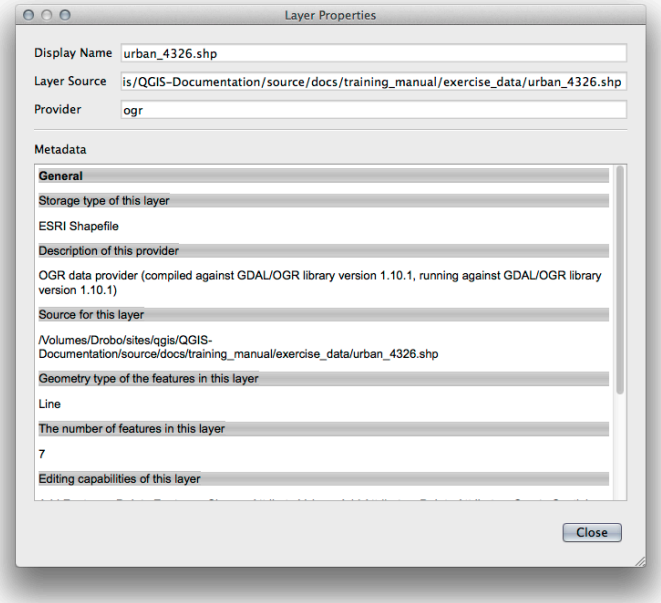
1. Select the `lines` layer in the Tree and click the *Export to File* button on the toolbar to open the *Export to vector file* dialog.
2. Click the … button to select the *Output file* and save the data to your `exercise_data` directory as `urban_4326`.
3. Set the *Target SRID* as `4326`.
4. Click *OK* to initialize the export.



5. Dismiss the dialog letting you know the export was successful and close the DB Manager.

You can now inspect the shapefile you created with the Browser panel.





18.2.7 In Conclusion

You have now seen how to use the DB Manager interface in QGIS to manage your spatial databases, to execute SQL queries against your data and how to import and export data.

18.2.8 What's Next?

Next, we will look at how to use many of these same techniques with *SpatiaLite* databases.

18.3 Lesson: Working with SpatiaLite databases in QGIS

While PostGIS is generally used on a server to provide spatial database capabilities to multiple users at the same time, QGIS also supports the use of a file format called *SpatiaLite* that is a lightweight, portable way to store an entire spatial database in a single file. Obviously, these 2 types of spatial databases should be used for different purposes, but the same basic principles and techniques apply to both. Let's create a new SpatiaLite database and explore the functionality provided to work with these databases in QGIS.

The goal for this lesson: To learn how to interact with SpatiaLite databases using the QGIS Browser interface.

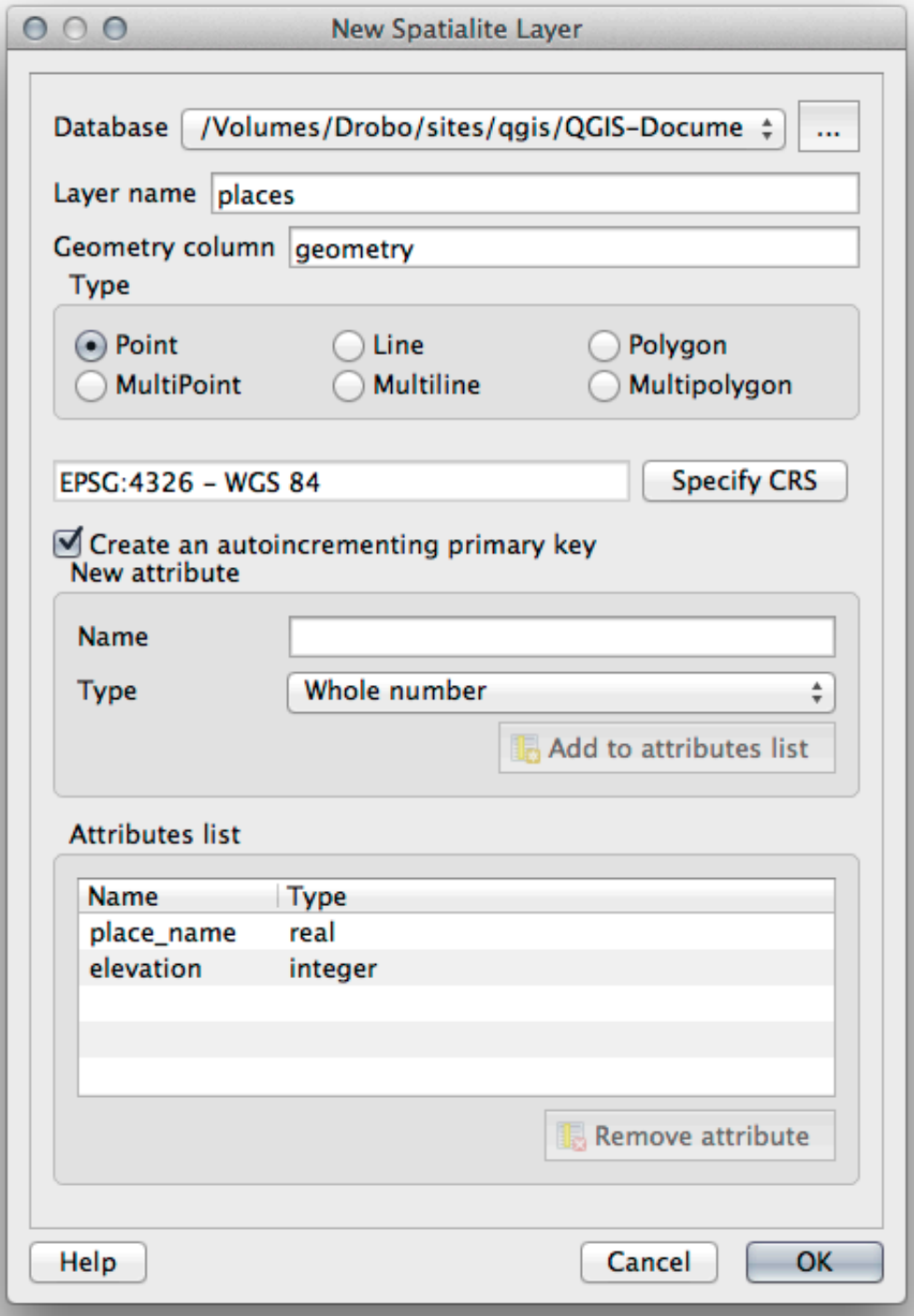
18.3.1 ??? Follow Along: Creating a SpatiaLite database with the Browser

Using the Browser panel, we can create a new SpatiaLite database and get it setup for use in QGIS.

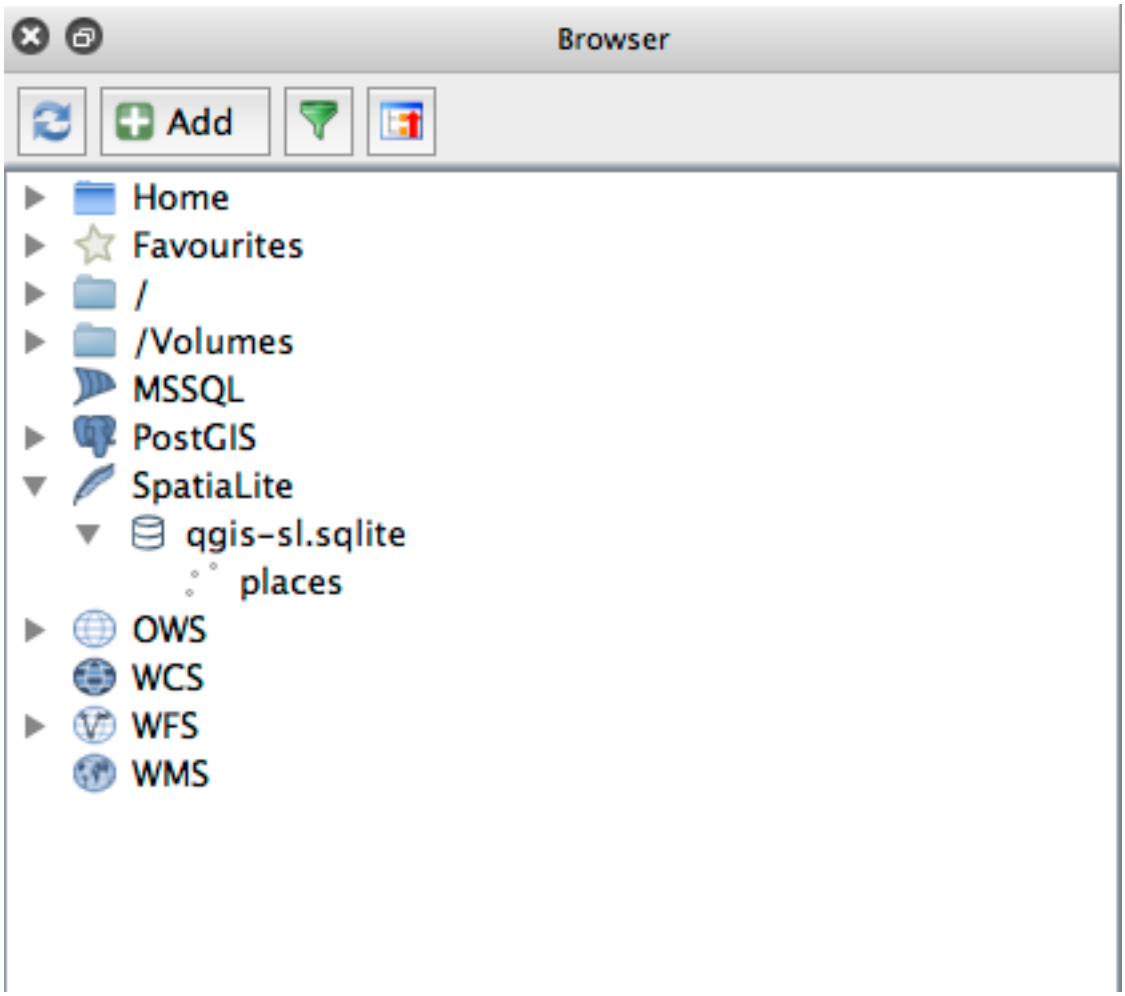
1. Right click on the *SpatiaLite* entry in the Browser tree and select *Create Database*.
2. Specify where on your filesystem you want to store the file and name it `qgis-sl.db`.
3. Again right click on the *SpatiaLite* entry in the Browser tree and now select the *New Connection* item. Find the file you created in the last step and open it.

Now that you have configured your new database you will find that the entry in Browser tree has nothing underneath it and the only thing you can do at this point is to delete the connection. This is of course because we haven't added any tables to this database. Let's go ahead and do that.

1. Find the button to create a new layer and use the dropdown to create a new SpatiaLite layer, or select *Layer* ► *New* ► *New SpatiaLite Layer*.
2. Select the database we created in the previous steps in the drop down.
3. Give the layer the name `places`.
4. Tick the checkbox next to *Create an auto-incrementing primary key*.
5. Add two attributes as shown in below
6. Click *OK* to create the table.



7. Click the refresh button at the top of the Browser and you should now see your `places` table listed.



You can right click on the table and view its properties as we did in the previous exercise.

From here you can start an editing session and start adding data to your new database directly.

We also learned about how to import data into a database using the DB Manager and you can use this same technique to import data into your new SpatiaLite DB.

18.3.2 In Conclusion

You have seen how to create SpatiaLite databases, add tables to them and use these tables as layers in QGIS.

CAPÍTULO 19

Appendix: Contributing To This Manual

To add materials to this course, you must follow the guidelines in this Appendix. You are not allowed to alter the conditions in this Appendix except for clarification. This is to ensure that the quality and consistency of this manual can be maintained.

19.1 Downloading Resources

The source of this document is available at GitHub. Consult GitHub.com for instructions on how to use the git version control system.

19.2 Manual Format

This manual is written using Sphinx, a Python document generator using the reStructuredText markup language. Instructions on how to use these tools are available on their respective sites.

19.3 Adding a Module

To add a new module:

1. First create a new directory (directly under the top-level of the `qgis-training-manual` directory) with the name of the new module.
2. Under this new directory, create a file called `index.rst`. Leave this file blank for now.
3. Open the `index.rst` file under the top-level directory. Its first lines are:

```
.. toctree::
   :maxdepth: 2

   foreword/index
   introduction/index
```

You will note that this is a list of directory names, followed by the name `index`. This directs the top-level index file to the index files in each directory. The order in which they are listed determines the order they will have in the document.

1. Add the name of your new module (i.e., the name you gave the new directory), followed by `/index`, to this list, wherever you want your module to appear.
2. Remember to keep the order of the modules logical, such that later modules build on the knowledge presented in earlier modules.
3. Open your new module's own index file (*`module_name`*`/index.rst`).
4. Along the top of the page, create the module heading:
   1. write a first line of asterisks (`*`).
   2. Follow this with a line containing the markup phrase `Module:`, followed by the name of your module.
   3. End this off with another line of same number of asterisks.

Nota: The underline and the overline should not be shorter than the line containing the module title.

---

5. Leave a line open beneath this.
6. Write a short paragraph explaining the purpose and content of the module.
7. Leave one line open, then add the following text:

```
.. toctree::
   :maxdepth: 2

   lesson1
   lesson2
```

… where `lesson1`, `lesson2`, etc., are the names of your planned lessons.

The module-level index file will look like this:

```
*********************************************************************************
Module: Module Name
*********************************************************************************

Short paragraph describing the module.

.. toctree::
   :maxdepth: 2

   lesson1
   lesson2
```

19.4 Adding a Lesson

To add a lesson to a new or existing module:

1. Open the module directory.
2. Open the `index.rst` file (created above in the case of new modules).
3. Ensure that the name of the planned lesson is listed underneath the `toctree` directive, as shown above.
4. Create a new file under the module directory.
5. Name this file exactly the same as the name you provided in the module's `index.rst` file, and add the extension `.rst`.

Nota: For editing purposes, a `.rst` file works exactly like a normal text file (`.txt`).

1. To begin writing the lesson, write the markup phrase `Lesson`, followed by the lesson name.
2. In the next line, write a line of equal signs (=), not shorter than the lesson title.
3. Leave a line open after this.
4. Write a short description of the lesson's intended purpose.
5. Include a general introduction to the subject matter. See the existing lessons in this manual for examples.
6. Beneath this, start a new paragraph, beginning with this phrase:

```
**The goal for this lesson:**
```

7. Briefly explain the intended outcome of completing this lesson.
8. If you can't describe the goal of the lesson in one or two sentences, consider breaking the subject matter up into multiple lessons.

Each lesson will be subdivided into multiple sections, which will be addressed next.

19.5 Adding a Section

There are two types of sections: «follow along» and «try yourself».

- A «follow along» section is a detailed set of directions intended to teach the reader how to use a given aspect of QGIS. This is typically done by giving click-by-click directions as clearly as possible, interspersed with screenshots.
- A «try yourself» section gives the reader a short assignment to try by themselves. It is usually associated with an entry in the answer box beneath, which will show or explain how to complete the assignment, and will show the expected outcome if possible.

Every section comes with a difficulty level. An easy section is denoted by ★☆☆, moderate by ★★☆, and advanced by ★★★.

19.5.1 Adding a «follow along» section

1. To start this section, write the markup phrase of the intended difficulty level (as shown above).
2. Leave a space and then write `Follow Along:`.
3. Leave another space and write the name of the section (use only an initial capital letter, as well as capitals for proper nouns).
4. In the next line, write a line of minuses/dashes (−), not shorter than the section title.
5. Write a short introduction to the section, explaining its purpose. Then give detailed (click-by-click) instructions on the procedure to be demonstrated.
6. In each section, include internal links, external links and screenshots as needed.
7. Try to end each section with a short paragraph that concludes it and leads naturally to the next section, if possible.

19.5.2 Adding a «try yourself» section

1. To start this section, write the markup phrase of the intended difficulty level (as shown above).
2. Leave a space and then write `Try Yourself:`.
3. In the next line, write a line of minuses/dashes (−), not shorter than the section title.
4. Explain the exercise that you want the reader to complete. Refer to previous sections, lessons or modules if necessary.
5. Include screenshots to clarify the requirements if a plain textual description is not clear.

In most cases, you will want to provide an answer regarding how to complete the assignment given in this section. To do so, you will need to create and feed an answer block beneath the instructions.

1. First, create the custom collapsible widget that contains the answer:

```
.. admonition:: Answer
   :class: dropdown
```

2. Keeping an indentation with regard to the above block, write the instructions on how to complete the assignment, using links and images where needed.

19.6 Add a Conclusion

To end a lesson:

1. Write the phrase `In Conclusion`, followed by a new line of minuses/dashes (−).
2. Write a conclusion for the lesson, explaining which concepts have been covered in the lesson.

19.7 Add a Further Reading Section

This section is optional.

- Write the phrase `Further Reading`, followed by a new line of minuses/dashes (−).
- Include links to appropriate external websites.

19.8 Add a What's Next Section

1. Write the phrase `What's Next?`, followed by a new line of minuses/dashes (−).
2. Explain how this lesson has prepared students for the next lesson or module.
3. Remember to change the «what's next» section of the previous lesson if necessary, so that it refers to your new lesson. This will be necessary if you have inserted a new lesson among existing lessons, or after an existing lesson.

19.9 Using Markup

To adhere to the standards of this document, you will need to add standard markup to your text.

19.9.1 New concepts

If you are explaining a new concept, you will need to write the new concept's name in italics by enclosing it in asterisks (*).

```
This sample text shows how to introduce a *new concept*.
```

19.9.2 Emphasis

- To emphasize a crucial term which is not a new concept, write the term in bold by enclosing it in double asterisks (**).
- Use this sparingly! If used too much, it can seem to the reader that you are shouting or being condescending.

```
This sample text shows how to use **emphasis** in a sentence. Include the
punctuation mark if it is followed by a **comma,** or at the **end of the
sentence.**
```

19.9.3 Images

- When adding an image, save it to an `img` folder next to the lesson file.
- Include it in the document like this:

```
.. figure:: img/image_file.extension
   :align: center
```

- Remember to leave a line open above and below the image markup.

19.9.4 Internal links

- To create an anchor for a link, write the following line above the place where you want the link to point to:

```
.. _link-name:
```

- Remember to leave a line open above and below this line.
- To create a link, refer to it as below:

```
:ref:`Descriptive link text <link-name>`
```

19.9.5 External links

- To create an external link, write it out like this:

```
`Descriptive link text <link-url>`_
```

19.9.6 Using monospaced text

- When you are writing text that the user needs to enter, a path name, or the name of a database element such as a table or column name, you must write it in `monospaced text`. For example:

```
Enter the following path in the text box: ``path/to/file``.
```

19.9.7 Labeling GUI items

- If you are referring to a GUI item, such as a button, you must write its name in *the GUI label format*. For example:

```
To access this tool, click on the :guilabel:`Tool Name` button.
```

- This also applies if you are mentioning the name of a tool without requiring the user to click a button.

19.9.8 Menu selections

- If you are guiding a user through menus, you must use the *menu* ► *selection* ► *format*. For example:

```
To use the :guilabel:`Tool Name` tool, go to :menuselection:`Plugins -->
Tool Type --> Tool Name`.
```

19.9.9 Adding notes

- You might need to add a note in the text, which explains extra details that can't easily be made part of the flow of the lesson. This is the markup:

```
[Normal paragraph.]

.. note:: Note text.
  New line within note.

  New paragraph within note.

[Unindented text resumes normal paragraph.]
```

19.9.10 Adding a sponsorship/authorship note

If you are writing a new module, lesson or section on behalf of a sponsor, you must include a short sponsor message of their choice. This must notify the reader of the name of the sponsor and must appear below the heading of the module, lesson or section that they sponsored. However, it may not be an advertisement for their company.

If you have volunteered to write a module, lesson or section in your own capacity, and not on behalf of a sponsor, you may include an authorship note below the heading of the module, lesson or section that you authored. This must take the form `This [module/lesson/section] contributed by [author name].` Do not add further text, contact details, etc. Such details are to be added in the «Contributors» section of the Foreword, along with the name(s) of the part(s) you added. If you only made enhancements, corrections and/or additions, list yourself as an editor.

19.10 Thank You!

Thank you for contributing to this project! By so doing, you are making QGIS more accessible to users and adding value to the QGIS project as a whole.

CAPÍTULO 20

A preparar os dados do exercício

Importante: Este processo destina-se a coordenadores de curso, ou a utilizadores de QGIS avançados que desejem criar conjuntos de dados de amostra localizados para o seu curso. Os conjuntos de dados prédefinidos são fornecidos com o Manual de Formação, mas pode seguir estas instruções se desejar substituir os conjuntos de dados prédefinidos.

Os *dados de amostra fornecidos* com o Manual de Formação referem-se à cidade Swellendam e suas imediações. A Swellendam localiza-se a 2 horas a oriente da Cidade do Cabo no Cabo Ocidental da África do Sul. O conjunto de dados nomes de elementos em Inglês e Afrikaans.

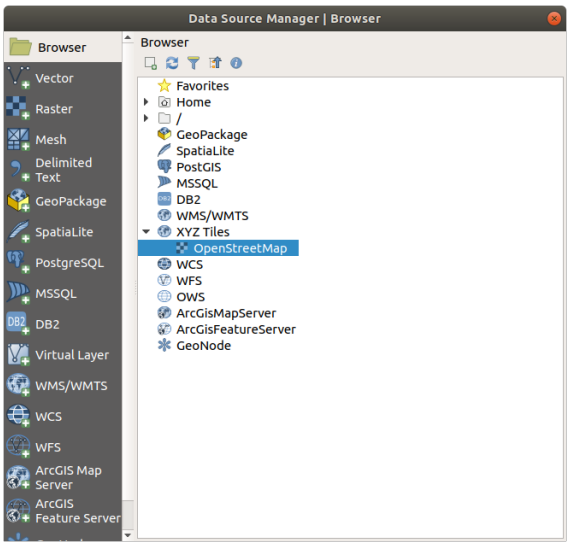
Todos podem utilizar este conjunto de dados sem dificuldade, mas poderá desejar utilizar dados do seu país ou cidade. Se o fizer, os seus dados localizados serão utilizados em todas as aulas do Módulo 3 ao Módulo 7.2. Os módulos posteriores utilizam fontes de dados mais complexas que poderão ou não estar disponíveis na sua região.

Nota: Estas instruções pressupõem que tem bons conhecimentos de QGIS e não se destinam a serem utilizados como material de ensino.

20.1 Create OSM based vector files

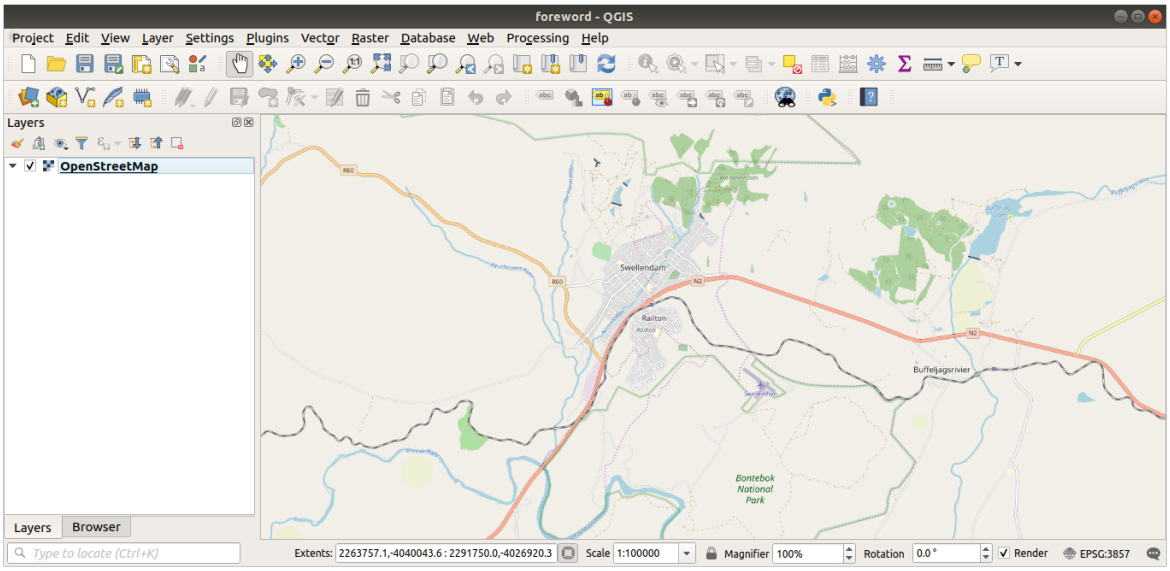
Se desejar substituir os conjuntos de dados prédefinidos por localizados no seu curso, isto pode ser facilmente efetuado com ferramentas existentes no QGIS. A região que escolher utilizador deverá ter uma boa mistura de áreas urbanas e rurais, contendo estradas diferenciadas, fronteiras de área (tais como reservas naturais ou quintas) e águas de superfície, tais como ribeiros e rios.

1. Abrir um novo projeto QGIS
2. Selecione *Layer* ► *Data Source Manager* para abrir a caixa de diálogo *Data Source Manager*
3. No separador *Browser*, expanda o menu pendente *XYZ Tiles* e clique duas vezes no item *OpenStreetMap*.



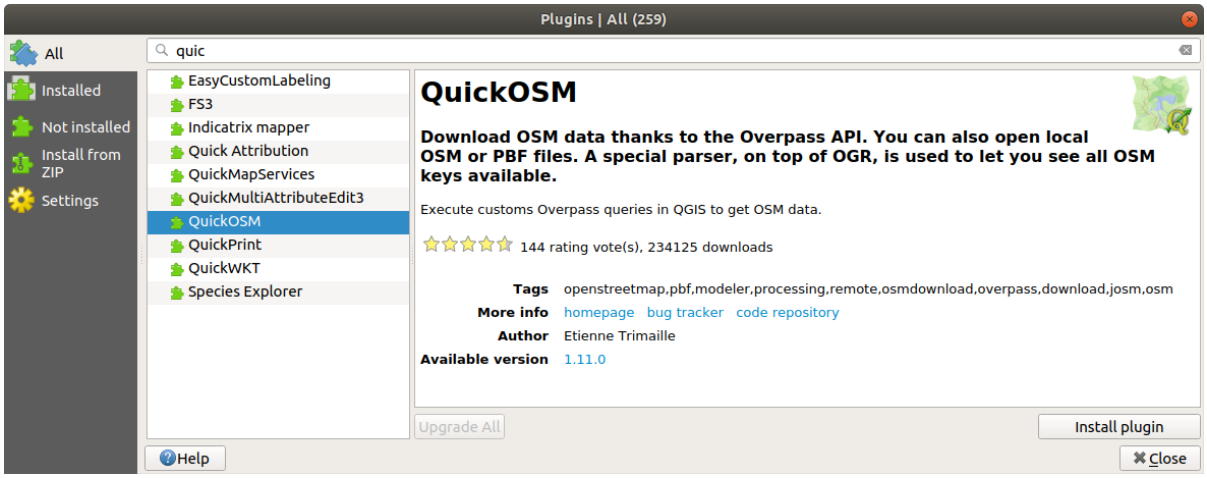
Um mapa mundo está agora visível na janela do mapa.

4. Feche a caixa de diálogo *Data Source Manager*
5. Mova a área que deseja utilizar como área de estudo

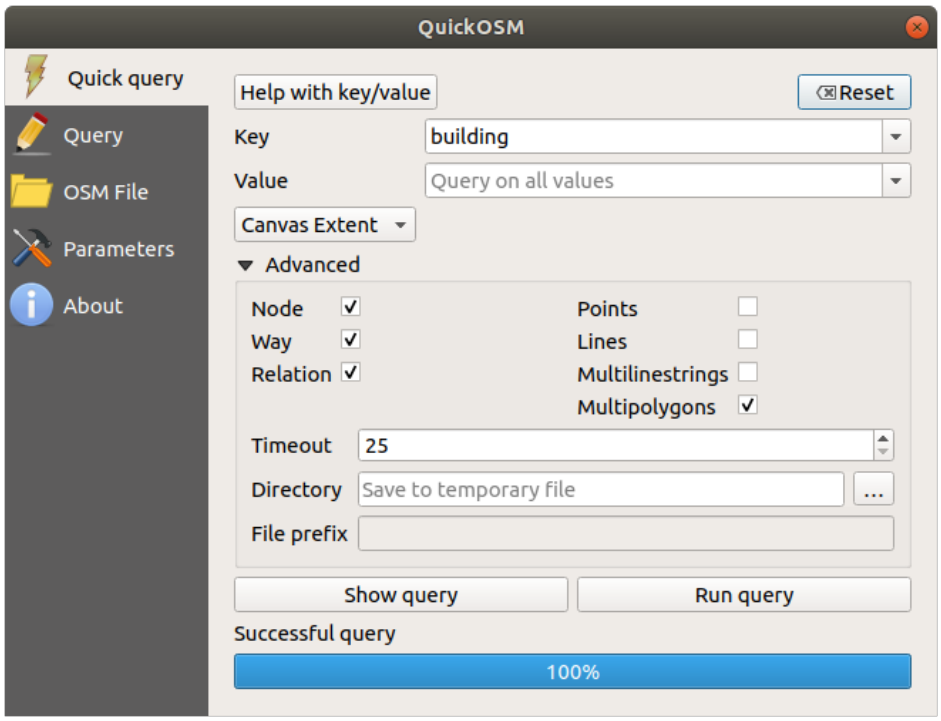


Agora temos a área da qual iremos extrair os dados, vamos ativar as ferramentas de extração.

1. Vá a *Plugins* ► *Manage/Install Plugins...*
2. No separador :*All*, digite `QuickOSM` na caixa de pesquisa
3. Selecione o módulo QuickOSM, prima *Install Plugin* e, de seguida, a caixa de diálogo *Close*.



4. Execute o novo módulo a partir do menu *Vector* ► *QuickOSM* ► *QuickOSM…*
5. No separador *Quick query*, selecione `building` no menu pendente *Key*
6. Deixe o campo *Value* vazio, para pesquisar todos os edifícios.
7. Selecione *Canvas Extent* no menu pendente seguinte
8. Expanda o grupo *Advanced* abaixo e anule a seleção de todos os tipos de geometria à direita exceto *Multipolygons*.
9. Prima *Run query*

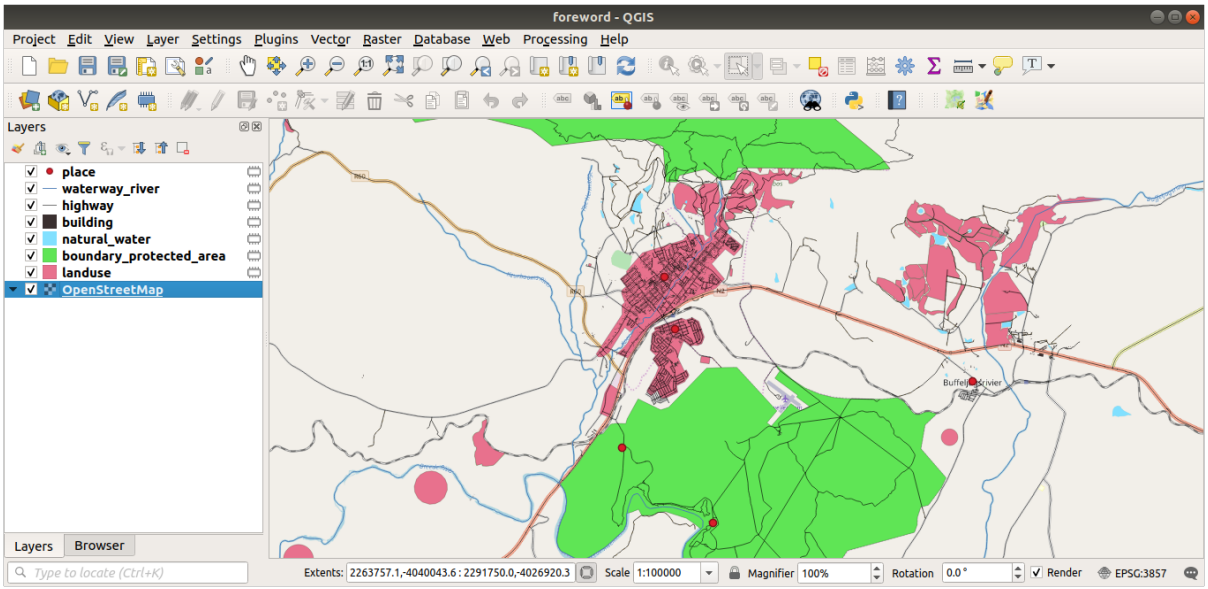


Uma camada `building` nova é adicionada ao painel, mostrando os edifícios na extensão selecionada.

10. Proceda como acima indicado para extrair outros dados:
    1. Tipo de geometria `Key = landuse` e `Multipolygons`.
    2. Tipo de geometria `Key = boundary`, `Value = protected_area` e `Multipolygons`.
    3. Tipo de geometria `Key = natural`, `Value = water` e `Multipolygons`.

4. `Key = highway` e selecione os tipos de geometria `Lines` e `Multilines`.
5. `Key = waterway`, `Value = river` e selecione os tipos de geometria `Lines` `Multilines`.
6. `Key = place` e tipo de geometria `Points`.

Este processo adiciona camadas como ficheiros temporários (indicados pelo ícone ao lado do seu nome).



Pode exemplificar os dados que a sua região contém para ver que tipo de resultados a sua região irá apresentar.

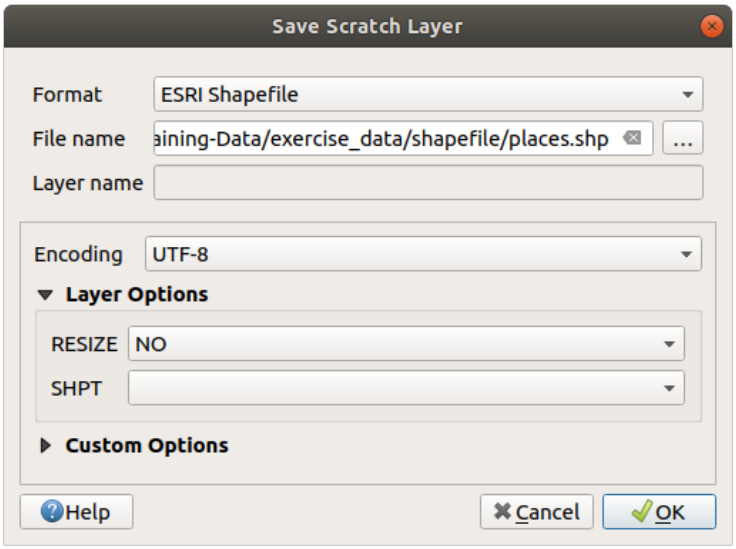
Agora precisamos de guardar os dados apresentados para utilizar durante o curso. Iremos utilizar os formatos ESRI Shapefile, GeoPackage e SpatiaLite dependendo dos dados.

Para converter a camada temporária *place* para outro formato:

1. Clique no ícone ao lado da camada *place* para abrir a caixa de diálogo *Save Scratch Layer*.

   Nota: Se necessitar de alterar quaisquer propriedades da camada temporária (CRS, exyensão, campos…), utilize o menu de contexto *Export* ► *Save Features as…*, e assegure-se que a opção *Add saved file to map* está selecionada. Isto adiciona uma nova camada.

2. Selecione o formato *ESRI Shapefile*
3. Utilize o botão … para navegar para a pasta `exercise_data/shapefile/` e guarde o ficheiro como `places.shp`.



4. Prima *OK*

   No painel *Layers*, a camada temporária *place* é substituída pela camada shapefile *places* e o ícone temporário ao seu lado é removido.

5. Clique duas vezes na camada para abrir o seu separador *Layer Properties* ► *Source* e atualizar a propriedade *Layer name* para corresponder ao nome do ficheiro.

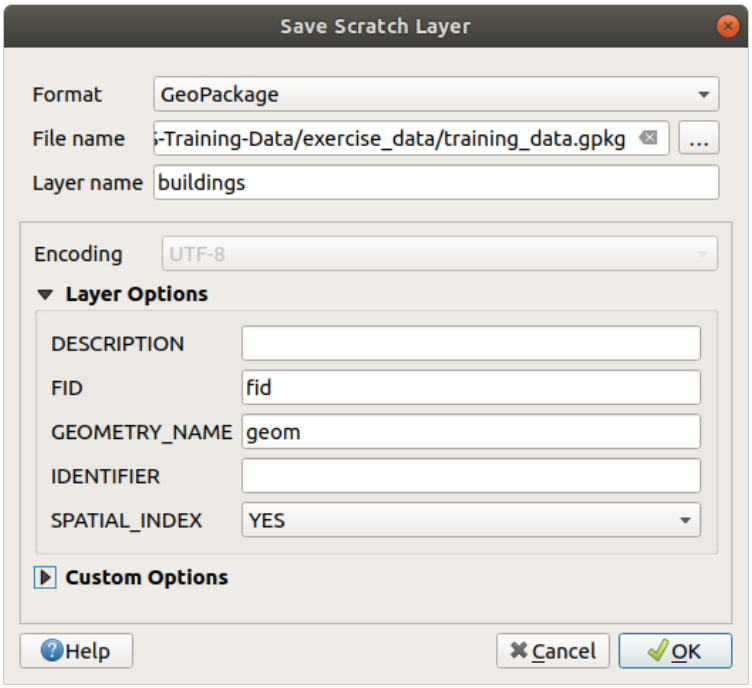
6. Repita o processo para outras camadas, atribuindo os respetivos nomes da seguinte forma:

   - `natural_water` para `water`
   - `waterway_river` para `rivers`
   - `boundary_protected_area` para `protected_areas`

   Cada conjunto de dados resultante deverá ser guardado no diretório `exercise_data/shapefile/`.

O passo seguinte é a criação de um ficheiro GeoPackage a partir da camada *building* para utilizar durante o curso:

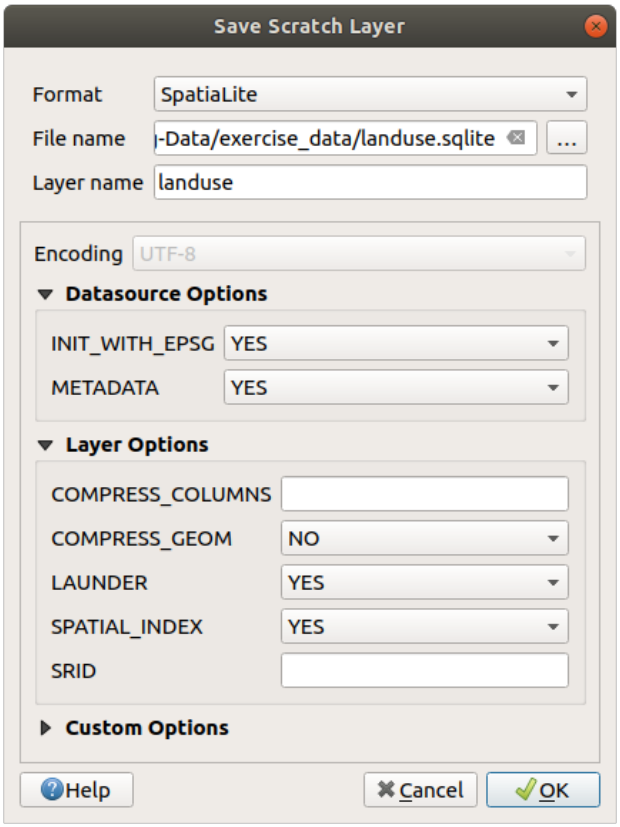
1. Clique no ícone ao lado da camada *building*
2. Selecione o formato *GeoPackage*
3. Guarde o ficheiro como `training_data.gpkg` na pasta `exercise_data/`
4. Por prédefinição, *Layer name* aparece como nome do ficheiro. Substitua-o por``buildings``.



5. Prima *OK*
6. Mude o nome da camada na sua caixa de diálogo de propriedades
7. Repita o prossesso com a camada *highway*, guardando-a como `roads` na mesma base de dados do GeoPackage.

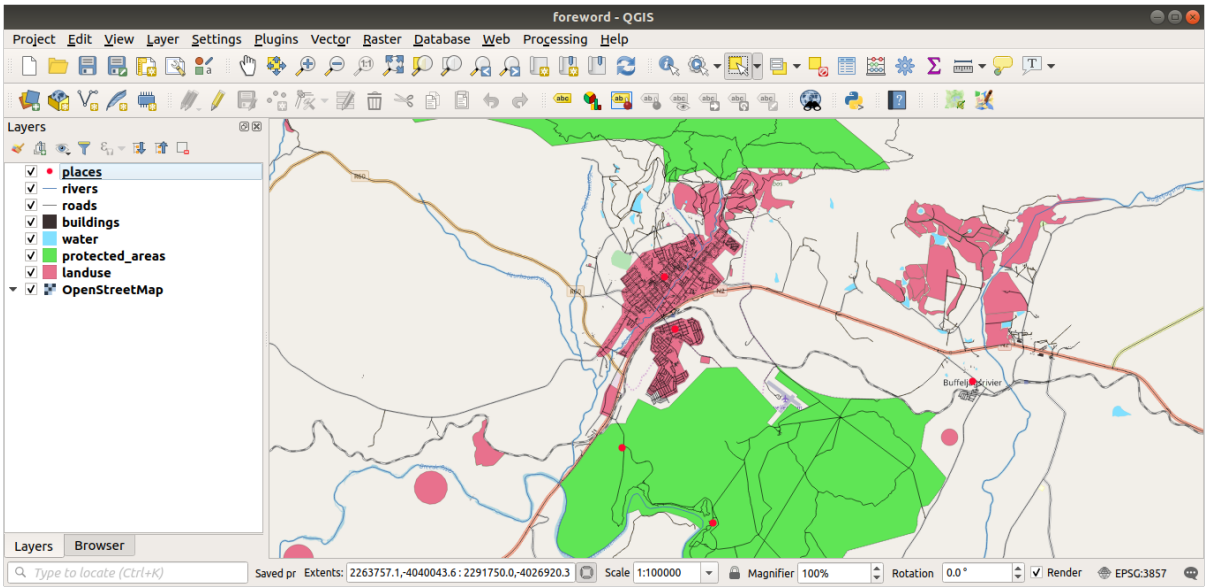
O último passo é mudar o nome do ficheiro temporário como um ficheiro SpatiaLite.

1. Clique no ícone ao lado da camada *landuse*
2. Selecione o formato *SpatiaLite*
3. Guarde o ficheiro como `landuse.sqlite` na pasta `exercise_data/`. Por prédefinição, *Layer name* aparece como nome do ficheiro. Não altere.



4. Prima *OK*

Deverá ter agora um mapa mais ou menos com o seguinte aspeto (a simbologia deverá ser certamente bastante diferente, uma vez que o QGIS atribui aleatoriamente cores sempre que são adicionadas camadas ao mapa):



O importante é que tenha 7 camadas de vetor correspondendo às acima mostradas e que todas essas camadas tenham os mesmos dados.

20.2 Create SRTM DEM tiff files

Para os módulos *Module: Creating Vector Data* e *Module: Rasters*, também irá necessitar de imagens matriciais (SRTM DEM) que abrangem a região que selecionou para o seu curso.

O CGIAR-CGI disponibiliza alguns MDE SRTM que pode fazer download de https://srtm.csi.cgiar.org/srtmdata/.

Irá necessitar de imagens que abrangem toda a região que escolheu. Para encontrar as coordenadas da extensão, no QGIS , em amplie para a extensão da camada maior e escolha os valores na caixa *Extents* da barra de estado. Mantenha o formato `GeoTiff`. Assim que o formulário esteja preenchido, clique no botão *Click here to Begin Search* >> e transfira o(s) ficheiro(s).

Assim que tenha transferido o(s) ficheiro(s) necessários, deverá guardá-lo(s) nas subpastas `exercise_data` directory, under `raster/SRTM`.

20.3 Create imagery tiff files

No módulo *Module: Creating Vector Data*, a lição *Follow Along: Data Sources* mostra imagens aproximadas de três campos de desporto que os alunos devem digitalizar. Deverá então reproduzir estas imagens utilizando o(s) seu(s) novo(s) ficheiro(s) tiff SRTM DEM. Não é obrigação utilizar campos de desporto escolares: podem ser utilizados quaisquer uns dos três tipos de território escolar (por exemplo, edifícios escolares, recreios ou estacionamento automóvel).

Para referência, a imagem nos dados de amostra é:

20.4 Replace tokens

Having created your localized dataset, the final step is to replace the tokens in the `substitutions.txt` file so that the appropriate names will appear in your localized version of the Training Manual.

Os tokens que necessita de substituir são os seguintes:

- `majorUrbanName`: passa para a prédefinição «Swellendam». Substitua pelo nome da principal cidade da sua região.
- `schoolAreaType1`: passa para a prédefinição «athletics field». Substitua pelo nome do tipo de área da maior escola da sua região.
- `largeLandUseArea`: passa para a prédefinição «Bontebok National Park». Substitua pelo nome do maior polígono de território da sua região.
- `srtmFileName`: passa para a prédefinição `srtm_41_19.tif`. Substitua pelo nome do ficheiro do seu ficheiro SRTM DEM.
- `localCRS`: passa para a prédefinição `WGS 84 / UTM 34S`. Deverá substituir pelo CRS correto da sua região.