
QGIS Training Manual

Release 2.0

QGIS Project

16.05.2014

| | | |
|----------|--|------------|
| 1 | Course Introduction | 1 |
| 1.1 | Foreword | 1 |
| 1.2 | Preparing Exercise Data | 3 |
| 2 | Module: The Interface | 11 |
| 2.1 | Lesson: A Brief Introduction | 11 |
| 2.2 | Lesson: Adding your first layer | 12 |
| 2.3 | Lesson: An Overview of the Interface | 14 |
| 3 | Module: Creating a Basic Map | 17 |
| 3.1 | Lesson: Working with Vector Data | 17 |
| 3.2 | Lesson: Symbology | 21 |
| 4 | Module: Classifying Vector Data | 51 |
| 4.1 | Lesson: Attribute Data | 51 |
| 4.2 | Lesson: The Label Tool | 52 |
| 4.3 | Lesson: Classification | 71 |
| 5 | Module: Creating Maps | 91 |
| 5.1 | Lesson: Using Map Composer | 91 |
| 5.2 | Assignment 1 | 100 |
| 6 | Module: Creating Vector Data | 103 |
| 6.1 | Lesson: Creating a New Vector Dataset | 103 |
| 6.2 | Lesson: Feature Topology | 113 |
| 6.3 | Lesson: Forms | 124 |
| 6.4 | Lesson: Actions | 136 |
| 7 | Module: Vector Analysis | 149 |
| 7.1 | Lesson: Reprojecting and Transforming Data | 149 |
| 7.2 | Lesson: Vector Analysis | 158 |
| 7.3 | Lesson: Network Analysis | 176 |
| 7.4 | Lesson: Spatial Statistics | 187 |
| 8 | Module: Rasters | 207 |
| 8.1 | Lesson: Working with Raster Data | 207 |
| 8.2 | Lesson: Changing Raster Symbology | 213 |
| 8.3 | Lesson: Terrain Analysis | 222 |
| 9 | Module: Completing the Analysis | 235 |
| 9.1 | Lesson: Raster to Vector Conversion | 235 |
| 9.2 | Lesson: Combining the Analyses | 238 |
| 9.3 | Assignment | 239 |

| | | |
|-----------|---|------------|
| 9.4 | Lesson: Supplementary Exercise | 239 |
| 10 | Module: Plugins | 253 |
| 10.1 | Lesson: Installing and Managing Plugins | 253 |
| 10.2 | Lesson: Useful QGIS Plugins | 257 |
| 11 | Module: Online Resources | 267 |
| 11.1 | Lesson: Web Mapping Services | 267 |
| 11.2 | Lesson: Web Feature Services | 276 |
| 12 | Module: GRASS | 285 |
| 12.1 | Lesson: GRASS Setup | 285 |
| 12.2 | Lesson: GRASS Tools | 296 |
| 13 | Module: Assessment | 305 |
| 13.1 | Create a base map | 305 |
| 13.2 | Analyze the data | 307 |
| 13.3 | Final Map | 307 |
| 14 | Module: Database Concepts with PostgreSQL | 309 |
| 14.1 | Lesson: Introduction to Databases | 309 |
| 14.2 | Lesson: Implementing the Data Model | 314 |
| 14.3 | Lesson: Adding Data to the Model | 319 |
| 14.4 | Lesson: Queries | 321 |
| 14.5 | Lesson: Views | 325 |
| 14.6 | Lesson: Rules | 326 |
| 15 | Module: Spatial Database Concepts with PostGIS | 329 |
| 15.1 | Lesson: PostGIS Setup | 329 |
| 15.2 | Lesson: Simple Feature Model | 332 |
| 15.3 | Lesson: Import and Export | 337 |
| 15.4 | Lesson: Spatial Queries | 339 |
| 15.5 | Lesson: Geometry Construction | 346 |
| 16 | Module: Using Spatial Databases in QGIS | 353 |
| 16.1 | Lesson: Working with Databases in the QGIS Browser | 353 |
| 16.2 | Lesson: Using DB Manager to work with Spatial Databases in QGIS | 356 |
| 16.3 | Lesson: Working with spatialite databases in QGIS | 369 |
| 17 | Appendix: Contributing To This Manual | 373 |
| 17.1 | Downloading Resources | 373 |
| 17.2 | Manual Format | 373 |
| 17.3 | Adding a Module | 373 |
| 17.4 | Adding a Lesson | 374 |
| 17.5 | Adding a Section | 375 |
| 17.6 | Add a Conclusion | 376 |
| 17.7 | Add a Further Reading Section | 376 |
| 17.8 | Add a What's Next Section | 376 |
| 17.9 | Using Markup | 376 |
| 17.10 | Thank You! | 378 |
| 18 | Answer Sheet | 379 |
| 18.1 | Results For <i>Adding Your First Layer</i> | 379 |
| 18.2 | Results For <i>An Overview of the Interface</i> | 379 |
| 18.3 | Results For <i>Working with Vector Data</i> | 379 |
| 18.4 | Results For <i>Symbology</i> | 380 |
| 18.5 | Results For <i>Attribute Data</i> | 385 |
| 18.6 | Results For <i>The Label Tool</i> | 386 |
| 18.7 | Results For <i>Classification</i> | 390 |
| 18.8 | Results For <i>Creating a New Vector Dataset</i> | 391 |

| | |
|--|------------|
| 18.9 Results For <i>Vector Analysis</i> | 395 |
| 18.10 Results For <i>Raster Analysis</i> | 405 |
| 18.11 Results For <i>Completing the Analysis</i> | 410 |
| 18.12 Results For <i>WMS</i> | 415 |
| 18.13 Results For <i>Database Concepts</i> | 418 |
| 18.14 Results For <i>Spatial Queries</i> | 421 |
| 18.15 Results For <i>Geometry Construction</i> | 421 |
| 18.16 Results For <i>Simple Feature Model</i> | 423 |
| 19 Indices and tables | 425 |

Course Introduction

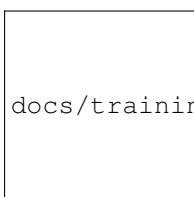
1.1 Foreword

1.1.1 Background

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.

1.1.2 License



docs/training_manual/foreword/../../_static/license.png

The Free Quantum GIS Training Manual by Linfiniti Consulting CC. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. Based on a work at <https://github.com/qgis/QGIS-Training-Manual>. Permissions beyond the scope of this license may be available at <https://github.com/qgis/QGIS-Training-Manual/blob/master/index.rst>.

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.:

Bemerkung: This lesson was sponsored by MegaCorp.

- If you are unsure about what you may do under this license, please contact us at office@linfiniti.com and we will advise you if what you intend doing is acceptable.
- If you publish this work under a self publishing site such as <http://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.

1.1.3 Sponsoring Chapters

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.

1.1.4 Authors

- Rüdiger Thiede (rudi@linfiniti.com) - Rudi has written the QGIS instructional materials and parts of the PostGIS materials.
- Tim Sutton (tim@linfiniti.com) - Tim has overseen and guided the project and co-authored the PostgreSQL and PostGIS parts. Tim also authored the custom sphinx theme used for this manual.
- Horst Düster (horst.duester@kappasys.ch) - Horst co-authored the PostgreSQL and PostGIS parts
- Marcelle Sutton (marcelle@linfiniti.com) - Marcelle provided proof-reading and editorial advice during the creation of this work.

1.1.5 Individual Contributors

Your name here!

1.1.6 Sponsors

- Cape Peninsula University of Technology

1.1.7 Data

Bemerkung: The sample data used throughout the manual can be downloaded here: http://qgis.org/downloads/data/training_manual_exercise_data.zip

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

- Streets and Places datasets from OpenStreetMap (<http://www.openstreetmap.org/>)
- Property boundaries (urban and rural), water bodies from NGI (<http://www.ngi.gov.za/>)
- SRTM DEM from the CGIAR-CGI (<http://srtm.csi.cgiar.org/>)

1.1.8 Latest Version

You can always obtain the latest version of this document by visiting our [home page](http://readthedocs.org) which is kindly hosted by <http://readthedocs.org>.

Bemerkung: There are links to PDF and epub versions of the documentation in the lower right hand corner of the above mentioned home page.

Tim Sutton, May 2012

1.2 Preparing Exercise Data

The sample data provided with the Training Manual refers to the town of Swellendam and its surroundings. Swellendam is located about 2 hours' east of Cape Town in the Western Cape of South Africa. The dataset contains feature names in both English and Afrikaans.

Anyone can use this dataset without difficulty, but you may prefer to use data from your own country or home town. If you choose to do so, your localised data will be used in all lessons from Module 3 to Module 7.2. Later modules use more complex data sources which may or may not be available for your region.

Bemerkung: This process is intended for course conveners, or more experienced QGIS users who wish to create localised sample data sets for their course. Default data sets are provided with the Training Manual, but you may follow these instructions if you wish to replace the default data sets.

Bemerkung: The sample data used throughout the manual can be downloaded here: http://qgis.org/downloads/data/training_manual_exercise_data.zip

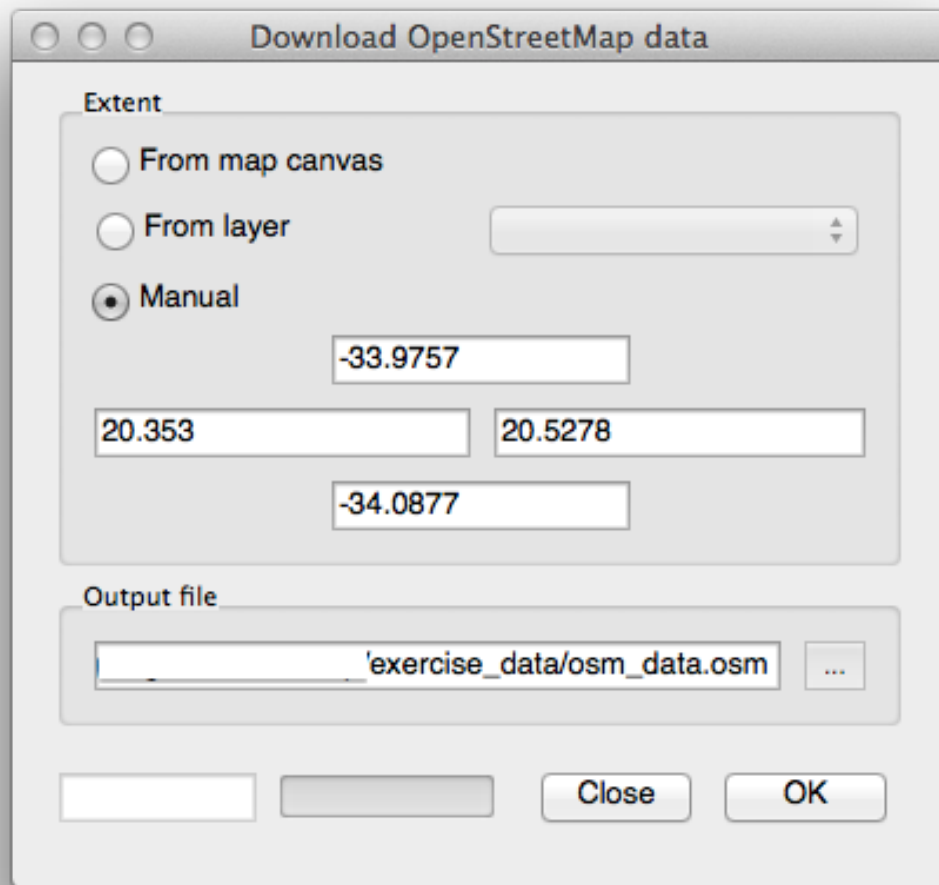
1.2.1 Try Yourself

Bemerkung: These instructions assume you have a good knowledge of QGIS and are not intended to be used as teaching material.

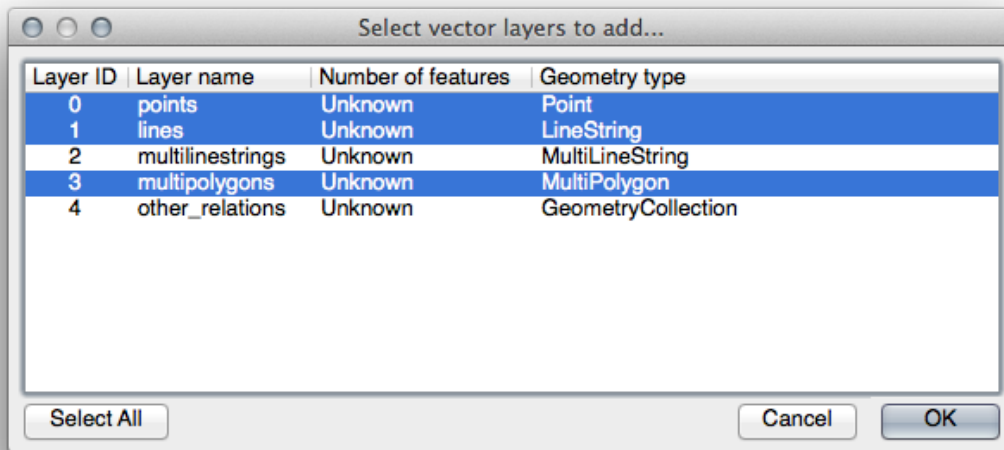
If you wish to replace the default data set with localised data for your course, this can easily be done with tools built into QGIS. The region you choose to use should have a good mix of urban and rural areas, containing roads of differing significance, area boundaries (such as nature reserves or farms) and surface water, such as streams and rivers.

- Open a new QGIS project
- In the *Vector* menu dropdown, select *OpenStreetMap -> Download Data*. You can then manually enter the co-ordinates of the region you wish to use, or you can use an existing layer to set the co-ordinates.

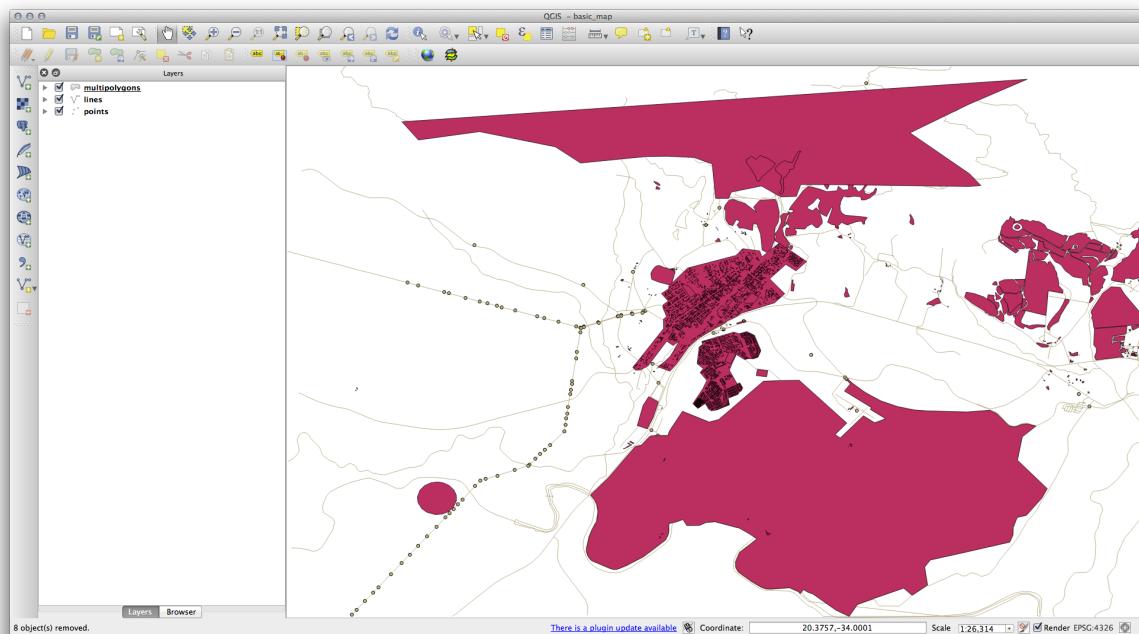
- Choose a location to save the resulting .osm file and click *Ok*:



- You can then open the .osm file using the *Add Vector Layer* button. You may need to select *All files* in the browser window. Alternatively, you can drag and drop the file into the QGIS window.
- In the dialog which opens, select all the layers, *except* the *other_relations* and *multilinestrings* layer:



This will load four layers into your map which relate to OSM's naming conventions (you may need to zoom in/out to see the vector data).



We need to extract the useful data from these layers, rename them and create corresponding shape files:

- First, double-click the `multipolygons` layer to open the *Layer properties* dialog.
- In the *General* tab, click *Query Builder* to open the *Query builder* window.

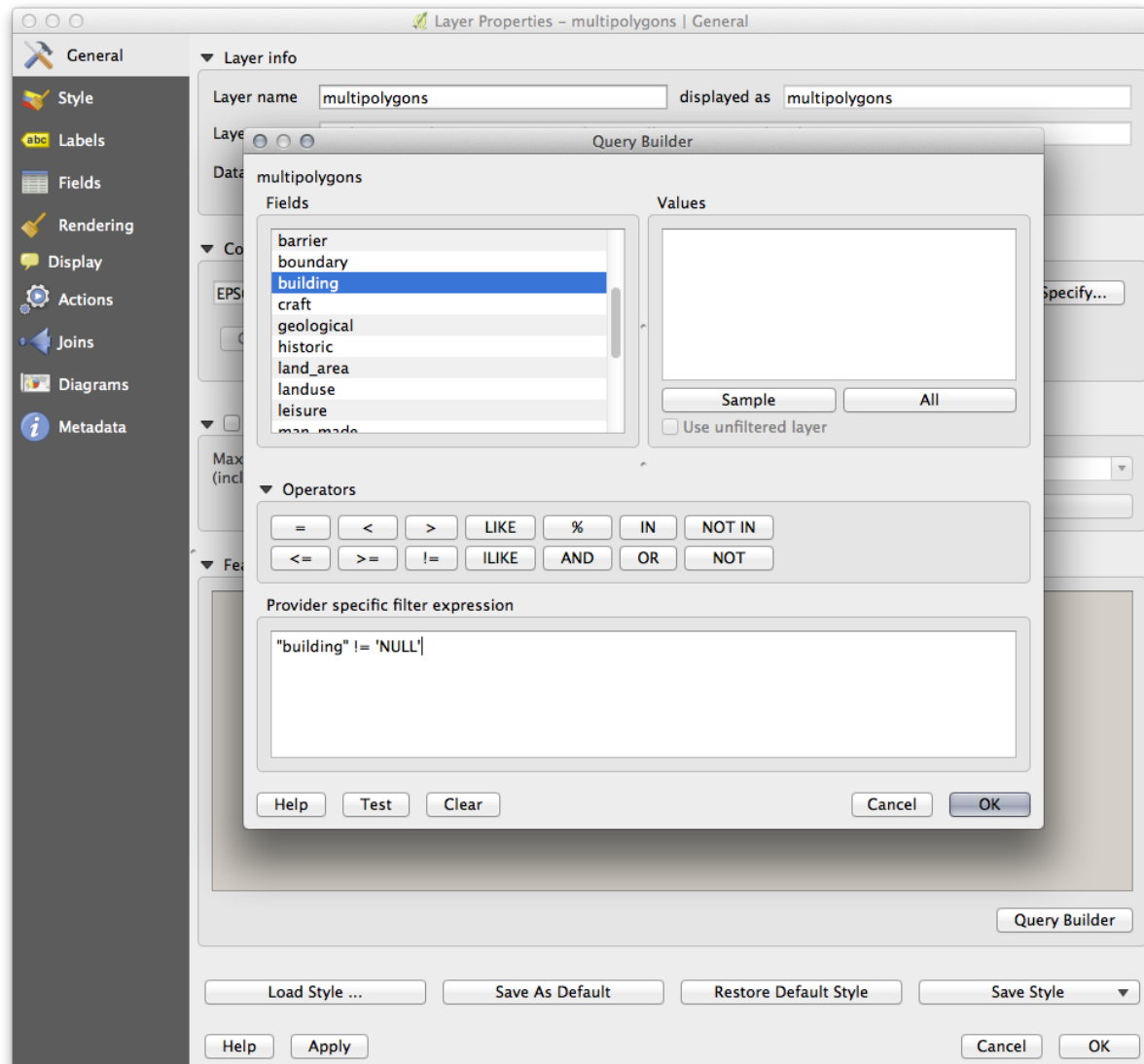
This layer contains three fields whose data we will need to extract for use throughout the Training Manual:

- `building`
- `natural` (specifically, `water`)
- `landuse`

You can sample the data your region contains in order to see what kind of results your region will yield. If you find that “`landuse`” returns no results, then feel free to exclude it.

You'll need to write filter expressions for each field to extract the data we need. We'll use the "building" field as an example here:

- Enter the following expression into the text area: `building != "NULL"` and click *Test* to see how many results the query will return. If the number of results is small, you may wish to have a look at the layer's *Attribute Table* to see what data OSM has returned for your region:



- Click *Ok* and you'll see that the layer elements which are not buildings have been removed from the map.

We now need to save the resulting data as a shapefile for you to use during your course:

- Right-click the *multipolygons* layer and select *Save As...*
- Make sure the file type is *ESRI Shapefile* and save the file in your new *exercise_data* directory, under a directory called "epsg4326".
- Make sure *No Symbolology* is selected (we'll add symbolology as part of the course later on).
- You can also select *Add saved file to map*.

Once the *buildings* layer has been added to the map, you can repeat the process for the *natural* and *landuse* fields using the following expressions:

Bemerkung: Make sure you clear the previous filter (via the *Layer properties* dialog) from the *multipolygons* layer before proceeding with the next filter expression!

- natural: “natural = ‘water’”
- landuse: “landuse != ‘NULL’”

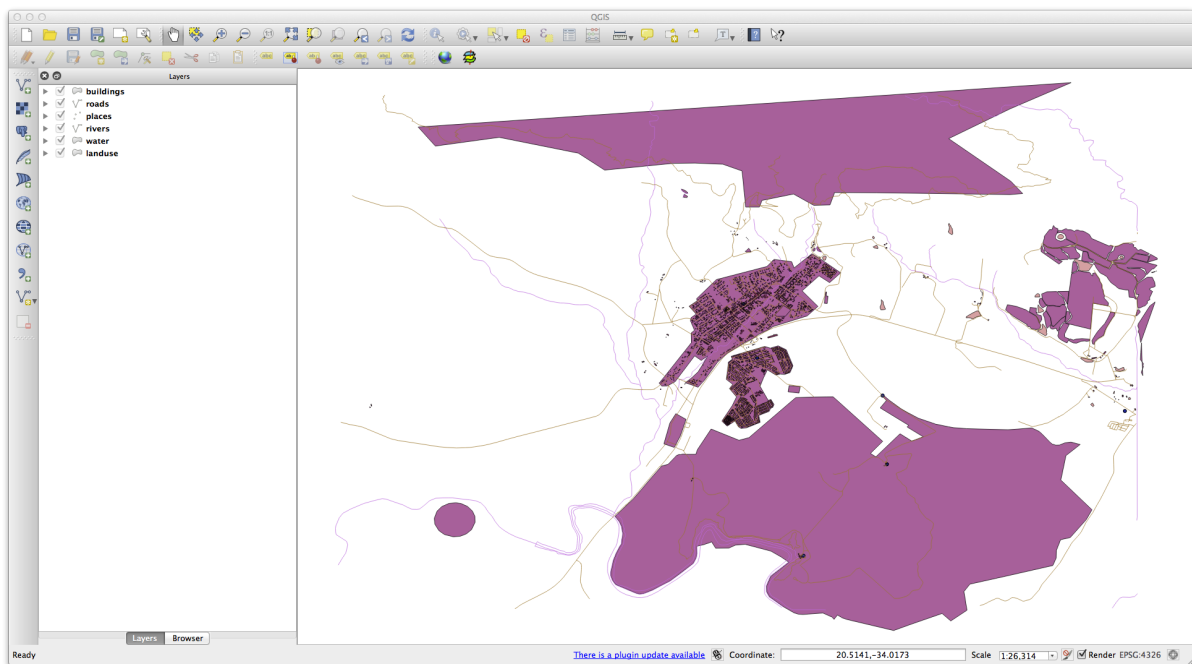
Each resulting data set should be saved in the “epsg4326” directory in your new `exercise_data` directory (i.e. “water”, “landuse”).

You should then extract and save the following fields from the `lines` and `points` layers to their corresponding directories:

- `lines`: “highway != ‘NULL’” to roads, and “waterway != ‘NULL’” to rivers
- `points`: “place != ‘NULL’” to places

Once you have finished extracting the above data, you can remove the *multipolygons*, *lines* and *points* layers.

You should now have a map which looks something like this (the symbology will certainly be very different, but that is fine):



The important thing is that you have 6 layers matching those shown above and that all those layers have some data.

The last step is to create a *spatialite* file from the `landuse` layer for use during the course:

- Right-click the `landuse` layer and select *Save as...*
- Select *Spatialite* as the format and save the file as `landuse` under the “epsg4326” directory.
- Click *Ok*.
- Delete the `landuse.shp` and its related files (if created).

1.2.2 Try Yourself Create SRTM DEM tiff Files

For Module 6 (Creating Vector Data) and Module 8 (Rasters), you’ll also need raster images (SRTM DEM) which cover the region you have selected for your course.

SRTM DEM can be downloaded from the CGIAR-CGI: <http://srtm.csi.cgiar.org/>

You’ll need images which cover the entire region you have chosen to use.

Once you have downloaded the required files, they should be saved in the “exercise_data” directory under “raster/SRTM/”.

In Module 6, Lesson 1.2 shows close-up images of three school sports fields which students are asked to digitize. You’ll therefore need to reproduce these images using your new SRTM DEM tiff files. There is no obligation to use school sports fields: any three school land-use types can be used (e.g. different school buildings, playgrounds or car parks).

For reference, the images in the example data are:





1.2.3 Try Yourself Replace Tokens

Having created your localised dataset, the final step is to replace the tokens in the `conf.py` file so that the appropriate names will appear in your localised version of the Training Manual.

The tokens you need to replace are as follows:

- `majorUrbanName`: this defaults to “Swellendam”. Replace with the name of the major town in your region.
- `schoolAreaType1`: this defaults to “athletics field”. Replace with the name of the largest school area type in your region.
- `largeLandUseArea`: this defaults to “Bontebok National Park”. Replace with the name of a large land-use polygon in your region.

Module: The Interface

2.1 Lesson: A Brief Introduction

Welcome to our course! Over the next few days, we'll be showing you how to use QGIS easily and efficiently. If you're new to GIS, we'll tell you what you need to get started. If you're an experienced user, you'll see how QGIS fulfills all the functions you expect from a GIS program, and more!

In this module we introduce the QGIS project itself, as well as explaining the user interface.

After completing this section, you will be able to correctly identify the main elements of the screen in QGIS and know what each of them does, and load a shapefile into QGIS.

Warnung: 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!

2.1.1 How to use this tutorial

Any text *that looks like this* refers to something on the screen that you can click on.

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

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

2.1.2 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.



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

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.



Intermediate

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

Following 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 from a GIS for everyday use.



Advanced

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

Following 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 QGIS' plugin system, database access system, 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.

2.1.3 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, we can show you how. The first lesson will guide you in creating your first QGIS map.

2.2 Lesson: Adding your first layer

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.


Bemerkung: Before starting this exercise, QGIS must be installed on your computer.

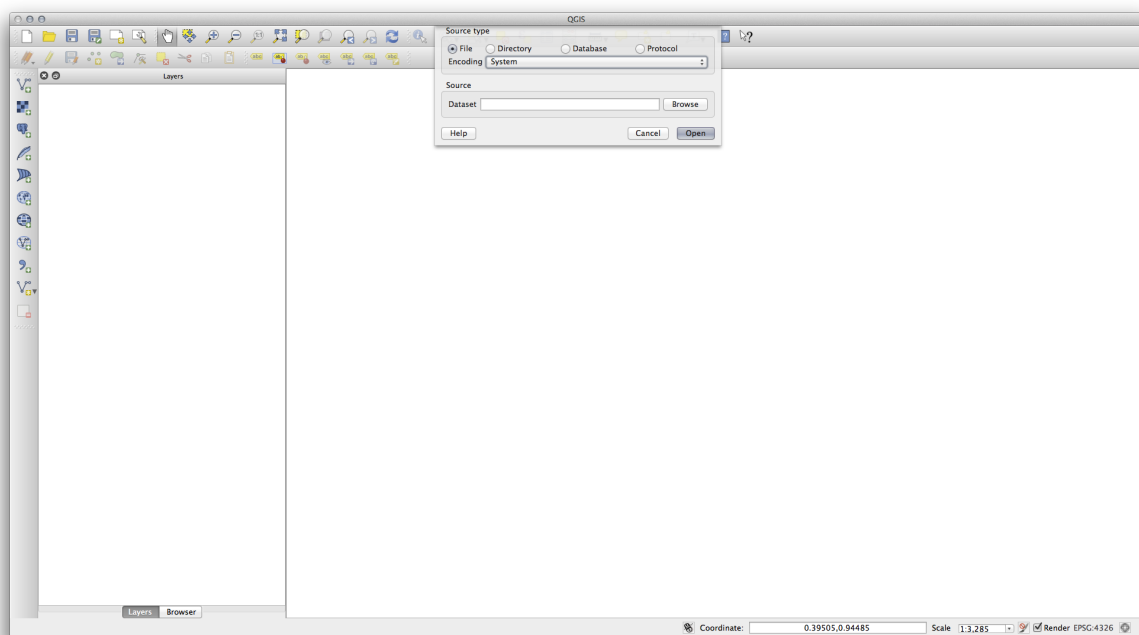
Launch QGIS from its desktop shortcut, menu item, etc., depending on how you configured its installation.

Bemerkung: The screenshots for this course were taken in QGIS 2.0 running on MacOS. 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 2.0 (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

- Open QGIS. You will have a new, blank map.
- Look for the *Add Vector Layer* button: 
- Click on it to open the following dialog:



- Click on the *Browse* button and navigate to the file `exercise_data/epsg4326/roads.shp` (in your course directory). With this file selected, click *Open*. You will see the original dialog, but with the file path filled in. Click *Open* here as well. The data you specified will now load.

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

- Click on the *Save As* button: 
- Save the map under `exercise_data/` and call it `basic_map.qgs`.

Check your results

2.2.2 In Conclusion

You've learned how to add a layer and create a basic map!

2.2.3 What's Next?

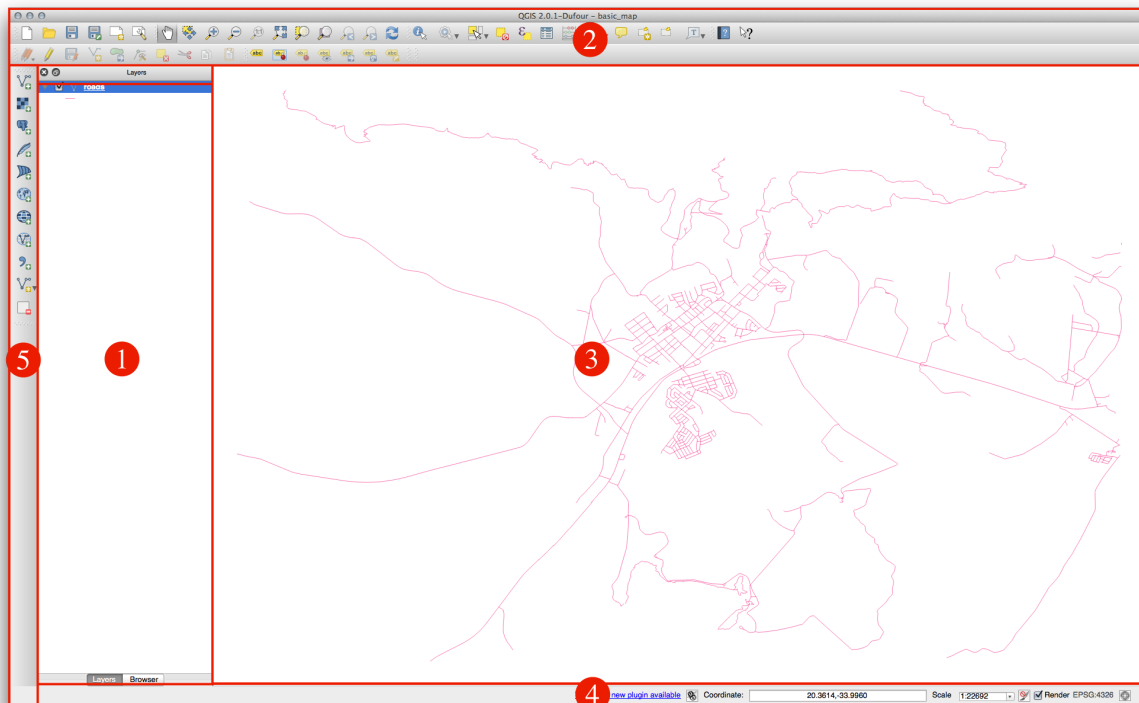
Now you're familiar with the function of the *Add Vector Layer* button, but what about all the others? How does this interface work? Before we go on with the more involved stuff, let's first take a good look at the general layout of the QGIS interface. This is the topic of the next lesson.

2.3 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.

2.3.1 Try Yourself: The Basics



The elements identified in the figure above are:

1. Layers List / Browser Panel
2. Toolbars
3. Map canvas
4. Status bar
5. Side Toolbar

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.

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!

Some versions of QGIS have a separate *Control rendering order* checkbox just underneath the Layers list. Don't worry if you can't see it. If it is present, ensure that it's checked for now.

Bemerkung: 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 or MSSQL Spatial) and WMS/WFS connections. You can also view your GRASS data.



Toolbars

Your most oft-used sets of tools can be turned into toolbars for basic access. For example, the File 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 *File* toolbar (which contains the *Save* button), you can still save your map by clicking on the *File* menu and then clicking on *Save*.



The Map Canvas

This is where the map itself is displayed.



The Status Bar

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

2.3.2



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.




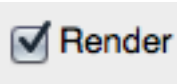

Check your results

2.3.3



Try Yourself 2

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

1. 
2. 
3. 
4. 
5. 

Bemerkung: 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.

Check your results

2.3.4 What's Next?

Now you've seen how the QGIS interface works, you can use the tools available to you and start improving on your map! This is the topic of the next lesson.

Module: Creating a Basic Map

In this module, you will create a basic map which will be used later as a basis for further demonstrations of QGIS functionality.

3.1 Lesson: Working with Vector Data

Vector data is arguably the most common kind of data you will find in the daily use of GIS. It describes geographic data in terms of points, that may be connected into lines and polygons. Every object in a vector dataset is called a **feature**, and is associated with data that describes that feature.


The goal for this lesson: To learn about the structure of vector data, and how to load vector datasets into a map.

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 *roads* layer loaded in your map. What you can see right now is merely the position of the roads.

To see all the data available to you, with the *roads* layer selected in the Layers panel:

- Click on this button: 

It will show you a table with more data about the *roads* layer. This extra data is called *attribute data*. The lines that you can see on your map represent where the roads go; this is the *spatial data*.

These definitions are commonly used in GIS, so it's essential to remember them!

- You may now close the attribute table.

Vector data represents features in terms of points, lines and polygons on a coordinate plane. It is usually used to store discrete features, like roads and city blocks.

3.1.2 Follow Along: Loading Vector Data From Shapefiles

The Shapefile is a specific file format that allows you to store GIS data in an associated group of files. Each layer consists of several files with the same name, but different file types. Shapefiles are easy to send back and forth, and most GIS software can read them.

Refer back to the introductory exercise in the previous section for instructions on how to add vector layers.


Load the data sets into your map following the same method:

- “places”
- “water”
- “rivers”
- “buildings”

Check your results

3.1.3 Follow Along: Loading Vector Data From a 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 Microsoft 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.

- Click on this icon: 

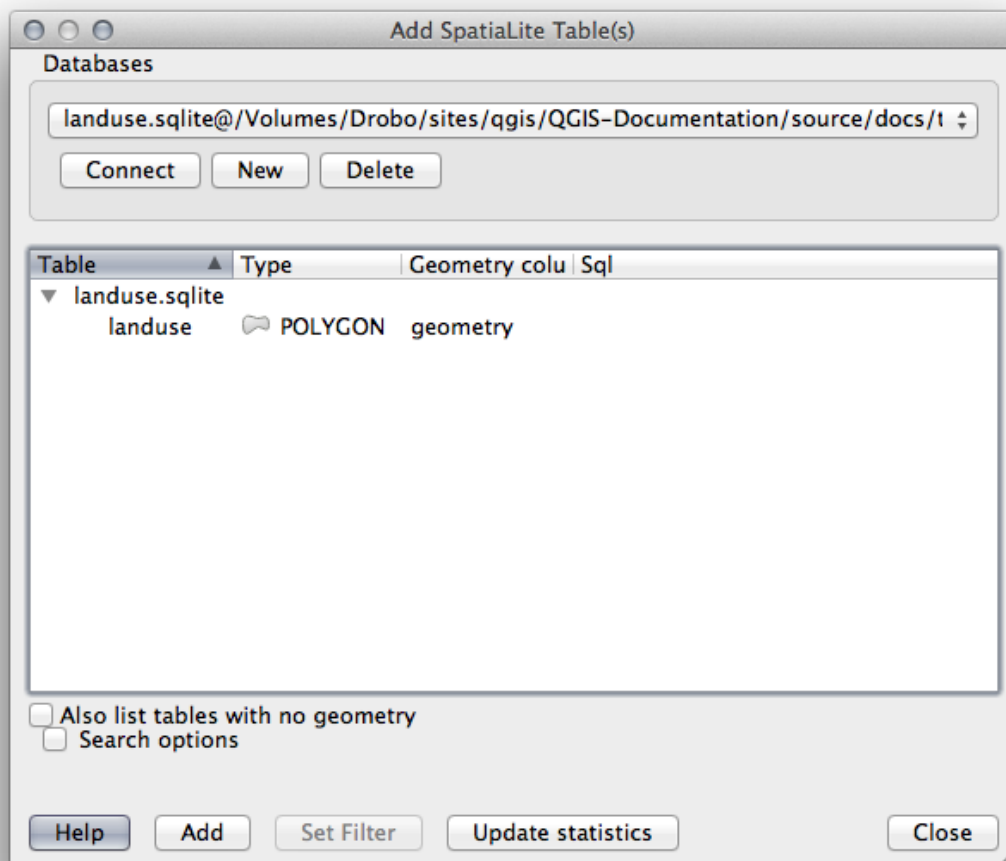
(If you're sure you can't see it at all, check that the *Manage Layers* toolbar is enabled.)

It will give you a new dialog. In this dialog:

- Click the *New* button.
- In the same folder as the other data, you should find the file *landuse.sqlite*. Select it and click *Open*.

You will now see the first dialog again. Notice that the dropdown select above the three buttons now reads “land_use.db@...”, followed by the path of the database file on your computer.

- Click the *Connect* button. You should see this in the previously empty box:



- Click on the `landuse` layer to select it, then click *Add*

Bemerkung: Remember to save the map often! The map file doesn't contain any of the data directly, but it remembers which layers you loaded into your map.

Check your results

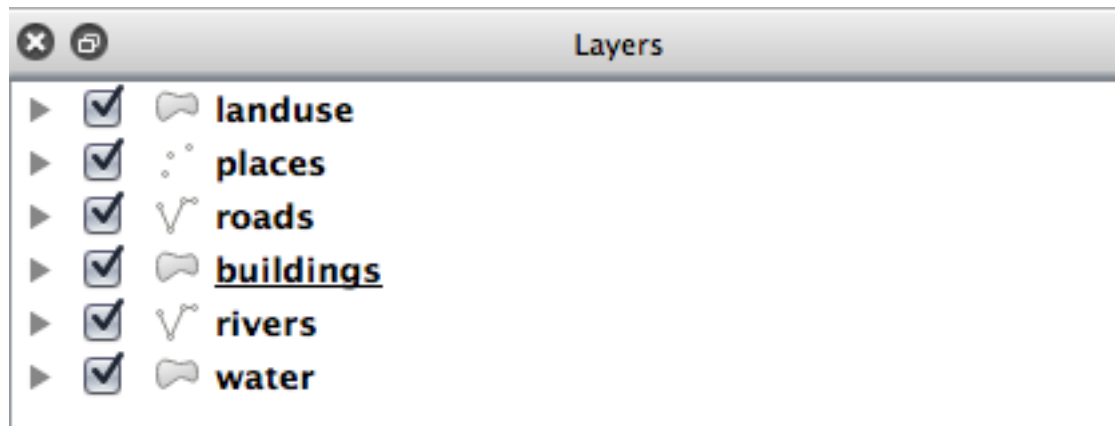
3.1.4 Follow Along: Reordering the Layers

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.

Bemerkung: Depending on the version of QGIS that you are using, you may have a checkbox beneath your Layers list reading *Control rendering order*. This must be checked (switched on) so that moving the layers up and down in the Layers list will bring them to the front or send them to the back in the map. If your version of QGIS doesn't have this option, then it is switched on by default and you don't need to worry about it.

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.

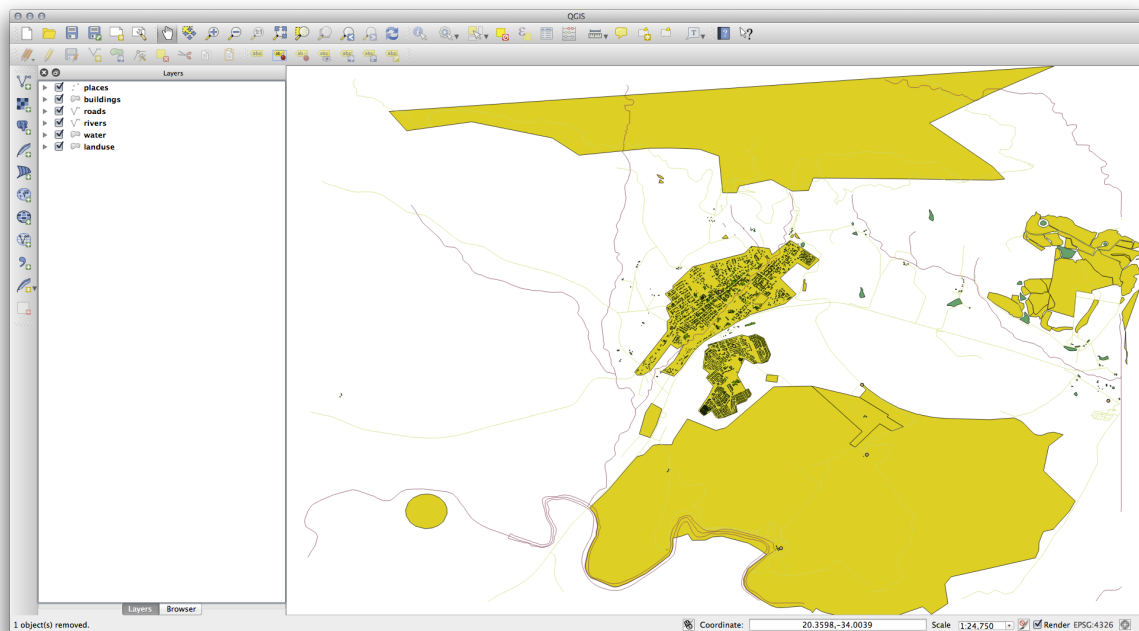
For example, this layer order...



... would result in roads and places being hidden as they run *underneath* urban areas.

To resolve this problem:

- Click and drag on a layer in the Layers list.
- 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.

3.1.5 In Conclusion

Now you've added all the layers you need from several different sources.

3.1.6 What's Next?

Using the random palette automatically assigned when loading the layers, your current map is probably not easy to read. It would be preferable to assign your own choice of colors and symbols. This is what you'll learn to do in the next lesson.

3.2 Lesson: Symbolology

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.

3.2.1 Follow Along: Changing Colors

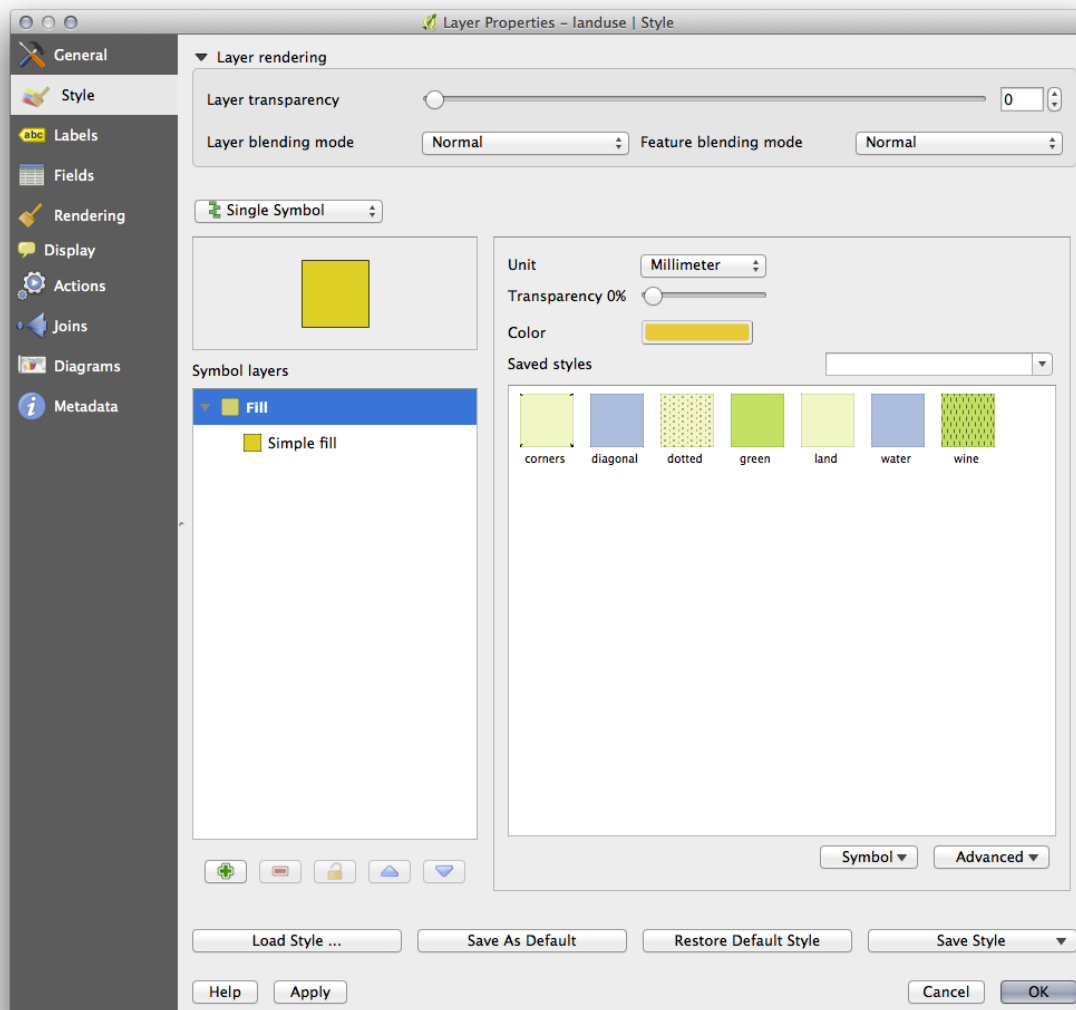
To change a layer's symbology, open its *Layer Properties*. Let's begin by changing the color of the *landuse* layer.

- Right-click on the *landuse* layer in the Layers list.
- Select the menu item *Properties* in the menu that appears.

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

In the *Properties* window:

- Select the *Style* tab at the extreme left:



- Click the color select button next to the *Color* label.

A standard color dialog will appear.

- Choose a gray color and click *OK*.
- Click *OK* again in the *Layer Properties* window, and you will see the color change being applied to the layer.

3.2.2 Try Yourself

Change the *water* layer to a light blue color.

Check your results

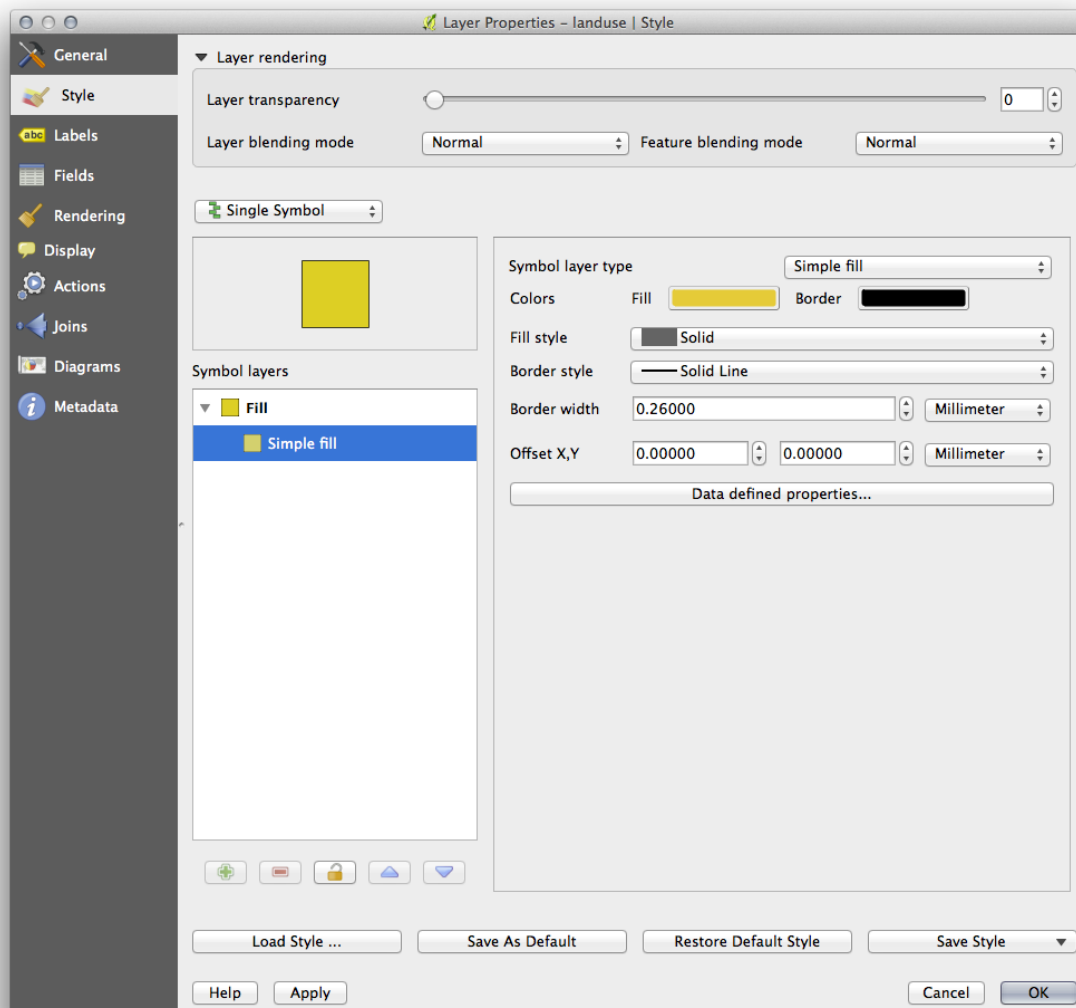
3.2.3 Follow Along: Changing Symbol Structure

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.

- Open the *Layer Properties* window for the *landuse* layer.

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

- In the *Symbol Layers* panel, expand the *Fill* dropdown (if necessary) and select the *Simple fill* option:



- Click on the *Border style* dropdown. At the moment, it should be showing a short line and the words *Solid Line*.
- Change this to *No Pen*.
- Click *OK*.

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

3.2.4 Try Yourself

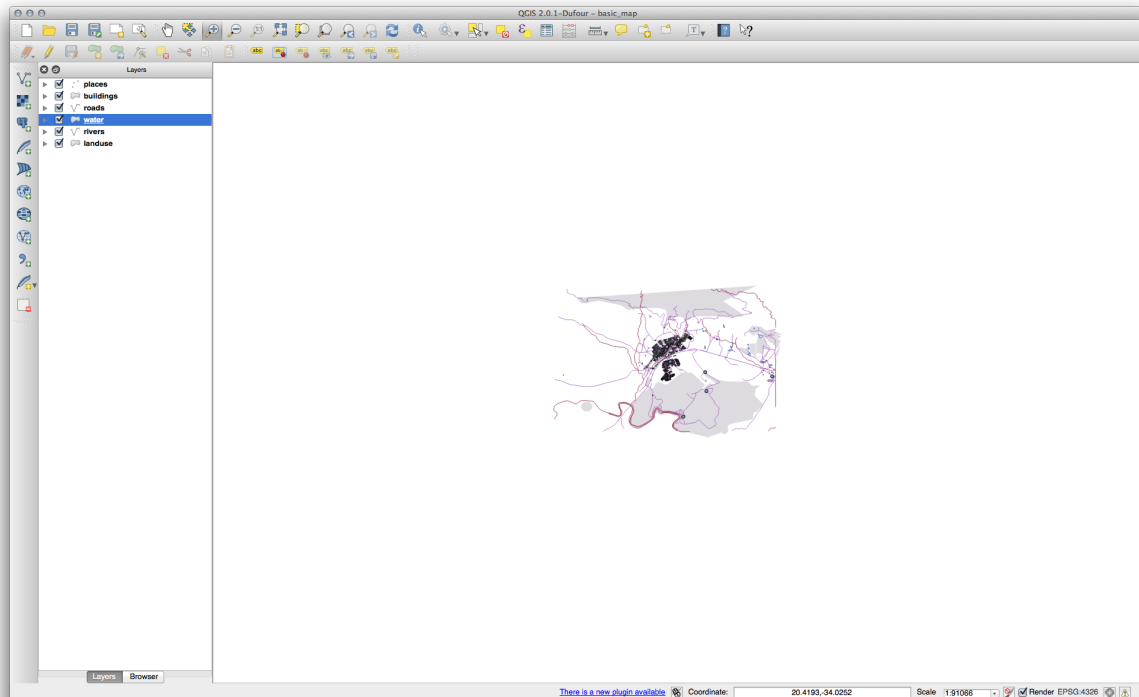
- Change the *water* layer's symbology again so that it has a darker blue outline.
- Change the *rivers* layer's symbology to a sensible representation of waterways.

Check your results

3.2.5 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.

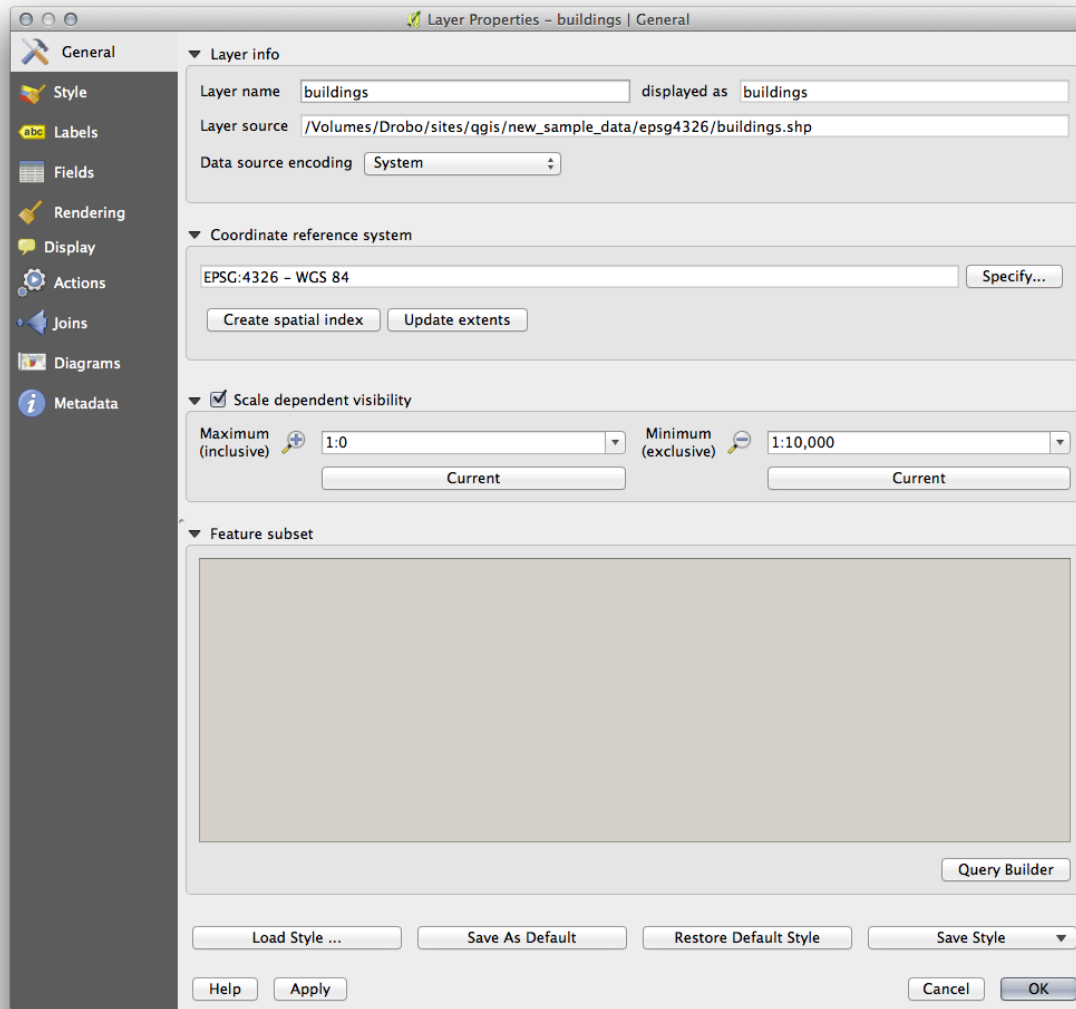
In our case, we may decide to hide the buildings from view at small scales. This map, for example ...



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

To enable scale-based rendering:

- Open the *Layer Properties* dialog for the *buildings* layer.
- Activate the *General* tab.
- Enable scale-based rendering by clicking on the checkbox labeled *Scale dependent visibility*:



- Change the *Maximum* value to 1 : 10 , 000.
- Click *OK*.

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

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



3.2.6 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.

- Go back to the *landuse* layer's symbol properties panel (by clicking

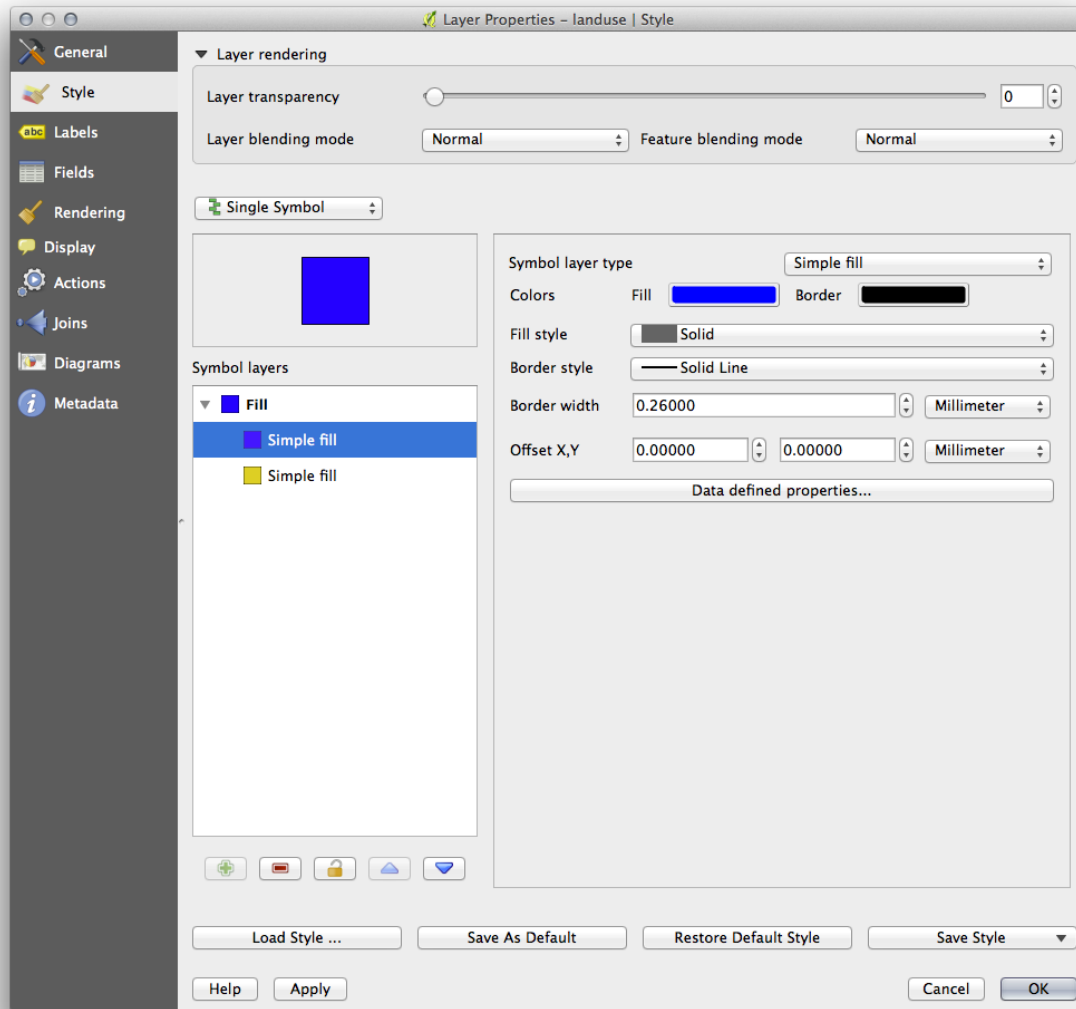
Simple fill in the *Symbol layers* panel).

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

Select the *Fill* in the *Symbol layers* panel. Then click the *Add symbol layer* button:



- Click on it and the dialog will change to look somewhat like this:



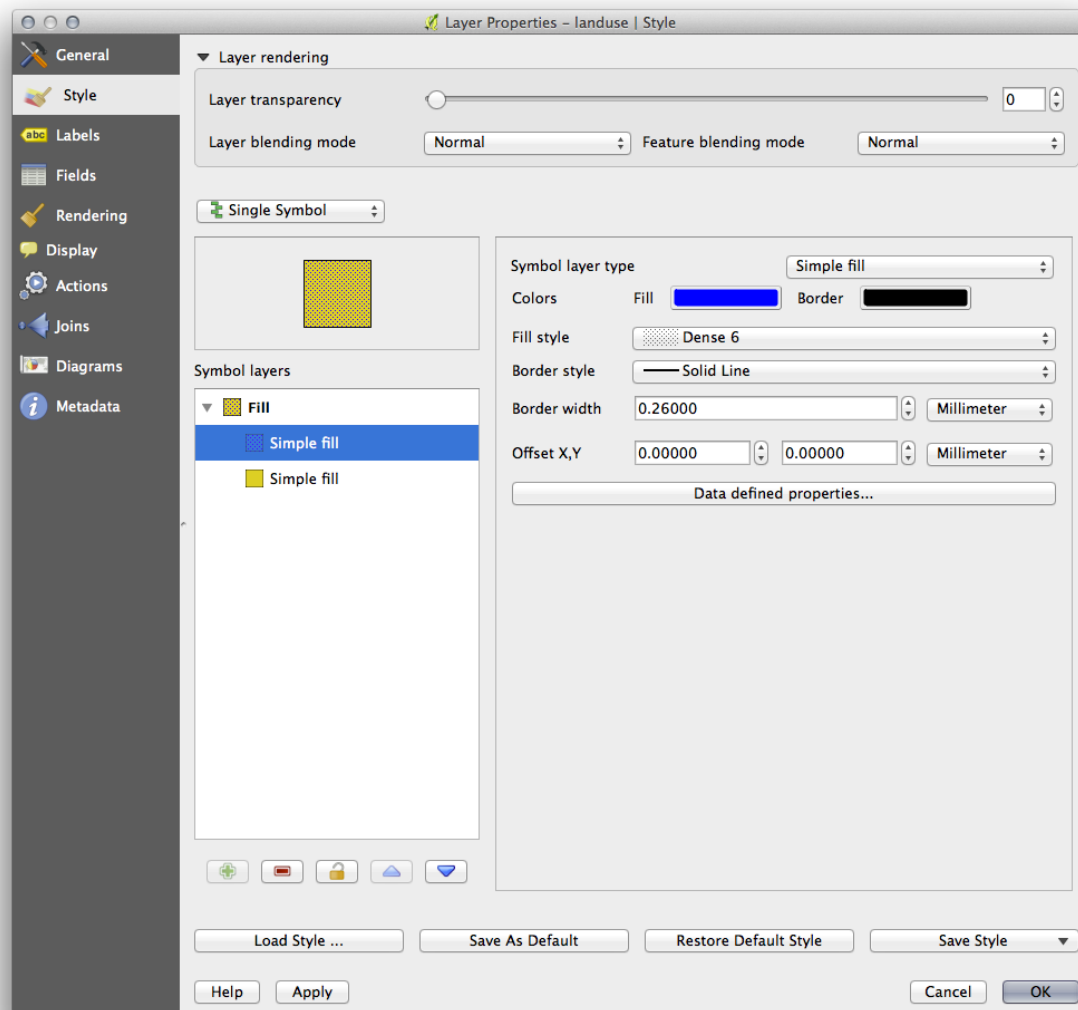
(It may appear somewhat different in color, for example, but you're going to change that anyway.)

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.

Bemerkung: 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.

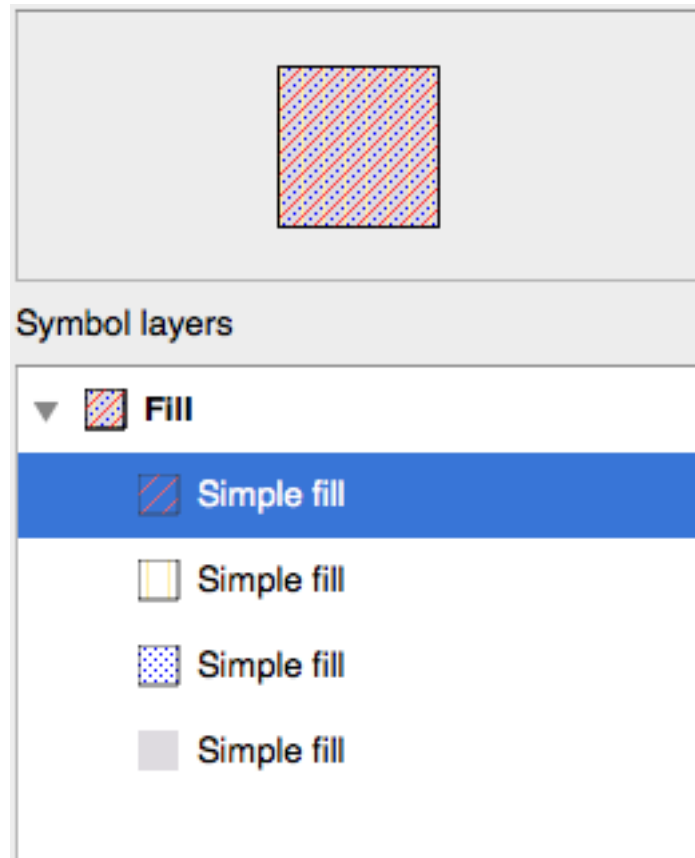
With the new *Simple Fill* layer selected:

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



- 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...

3.2.7 Try Yourself

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

Check your results

3.2.8 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.

- Give the *roads* layer an extra symbol layer (using the method for adding symbol layers demonstrated above).
- Give the base line a *Pen width* of 0.3, a white color and select *Dashed Line* from the *Pen Style* dropdown.
- Give the new, uppermost layer a thickness of 1.3 and ensure that it is a *Solid Line*.

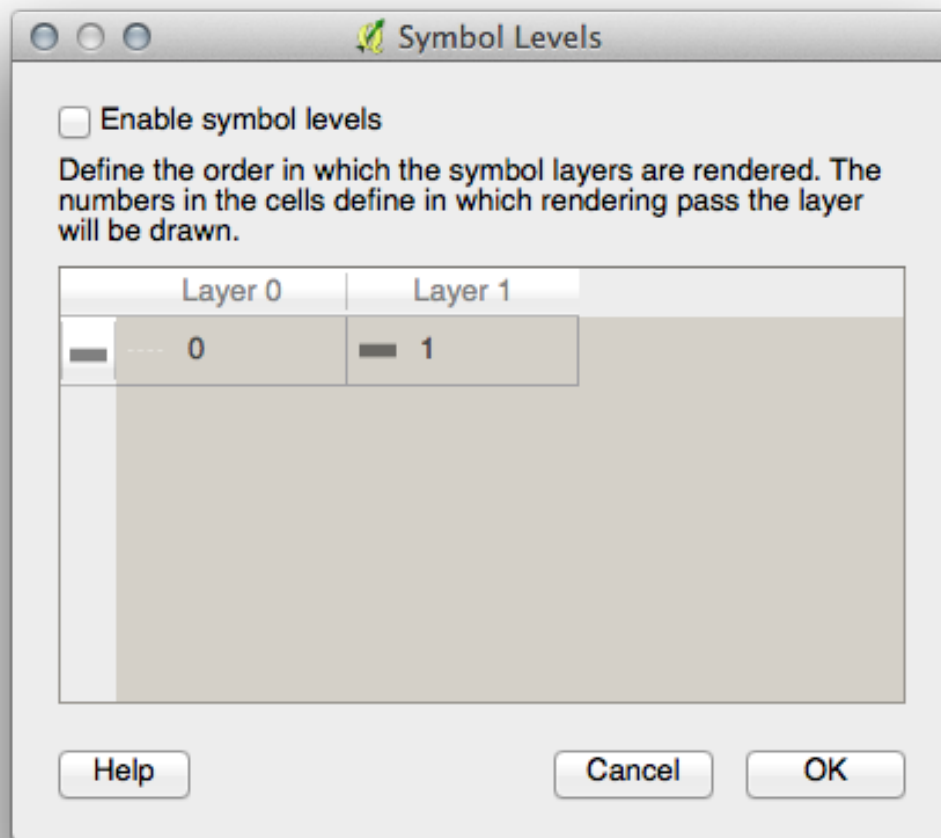
You'll notice that this happens:



Well that's not what we want at all!

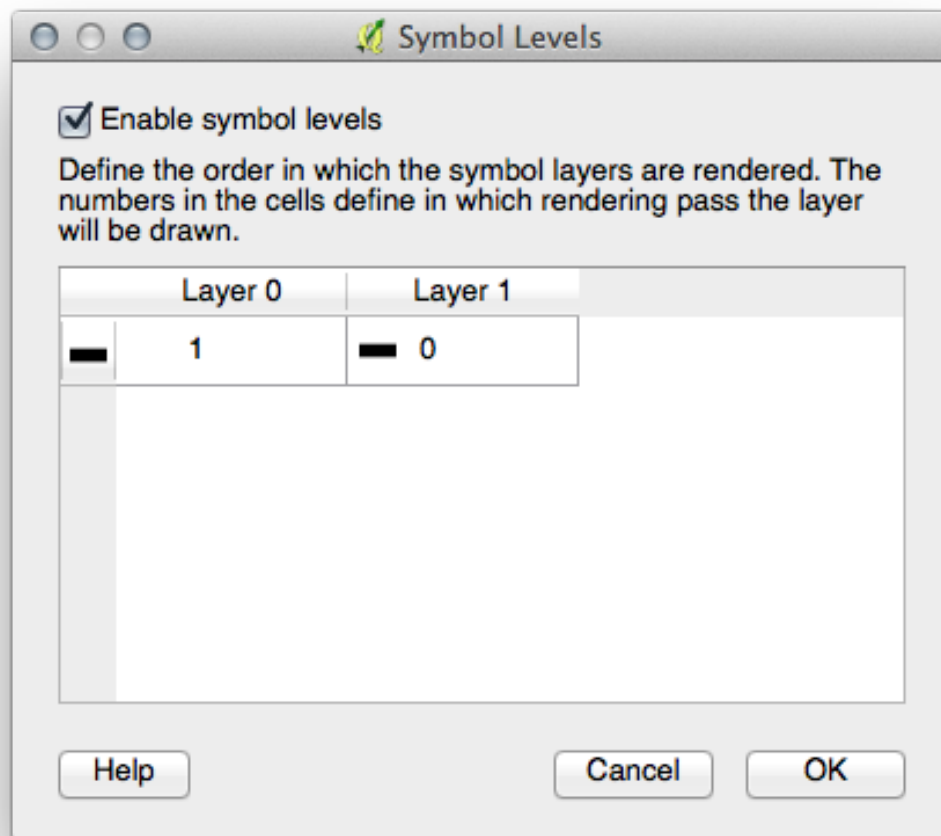
To prevent this from happening, you can sort the symbol levels and thereby control the order in which the different symbol layers are rendered.

To change the order of the symbol layers, select the *Line* layer in the *Symbol layers* panel, then click *Advanced* -> *Symbol levels...* in the bottom right-hand corner of the window. This will open a dialog like this:



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

In our case, we want to reverse the ordering, like this:



This will render the dashed, white line above the thick black line.

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

The map will now look like this:



Also note that the meeting points of roads are now “merged”, so that one road is not rendered above another.

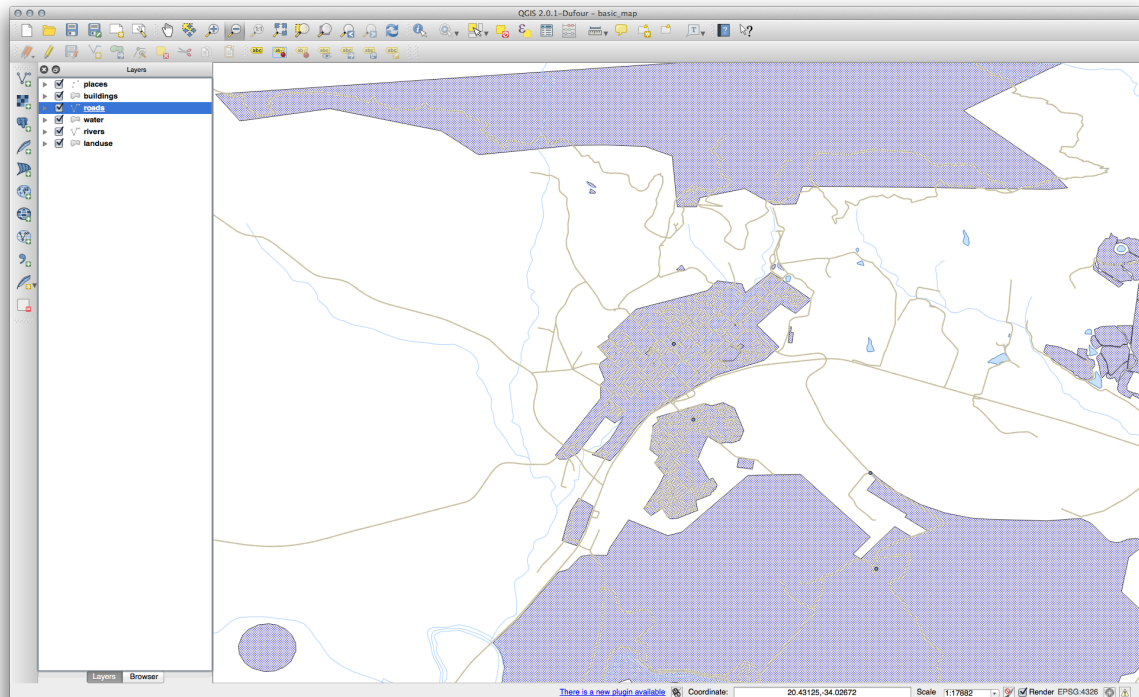
When you’re done, remember to save the symbol itself so as not to lose your work if you change the symbol again in the future. You can save your current symbol style by clicking the *Save Style ...* button under the *Style* tab of the *Layer Properties* dialog. Generally, you should save as *QGIS Layer Style File*.

Save your style under `exercise_data/styles`. 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.

3.2.9 Try Yourself

- Change the appearance of the *roads* layer again.

The roads must be narrow and mid-gray, with a thin, pale yellow outline. Remember that you may need to change the layer rendering order via the *Advanced -> Symbol levels...* dialog.

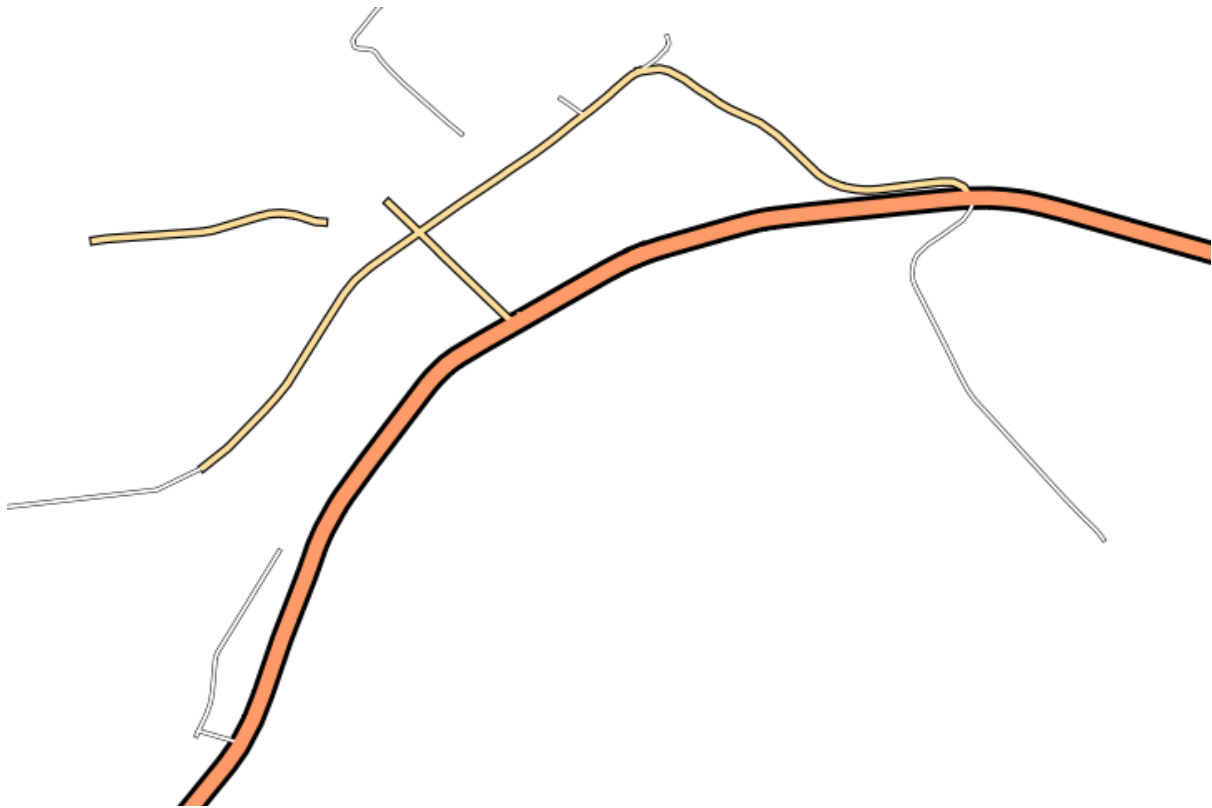


Check your results

3.2.10 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.

- Create a new map and add only the *roads* dataset.
- Apply the style `advanced_levels_demo.qml` provided in `exercise_data/styles`.
- Zoom in to the Swellendam area.
- Using symbol layers, ensure that the outlines of layers flow into one another as per the image below:



Check your results

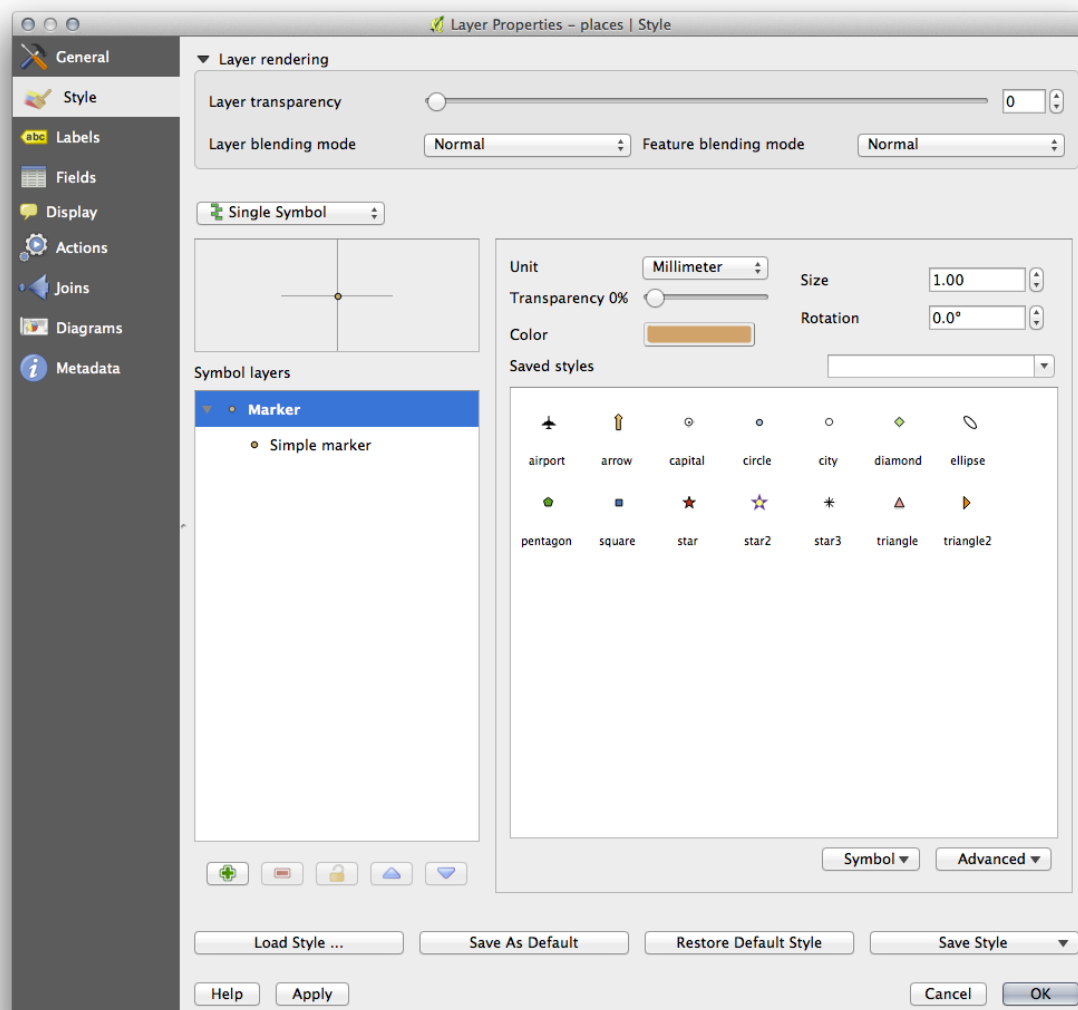
3.2.11 Follow Along: Symbol layer types

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.

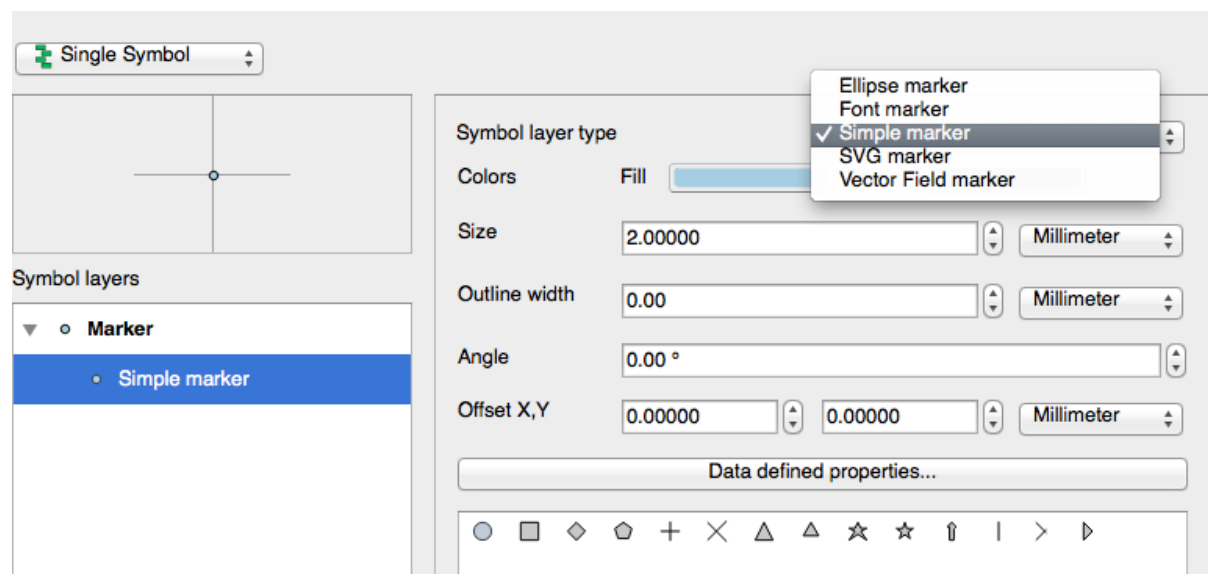
Each type of vector (point, line and polygon) has its own set of symbol layer types. First we will look at the types available for points.

Point Symbol Layer Types

- Open your *basic_map* project.
- Change the symbol properties for the *places* layer:



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

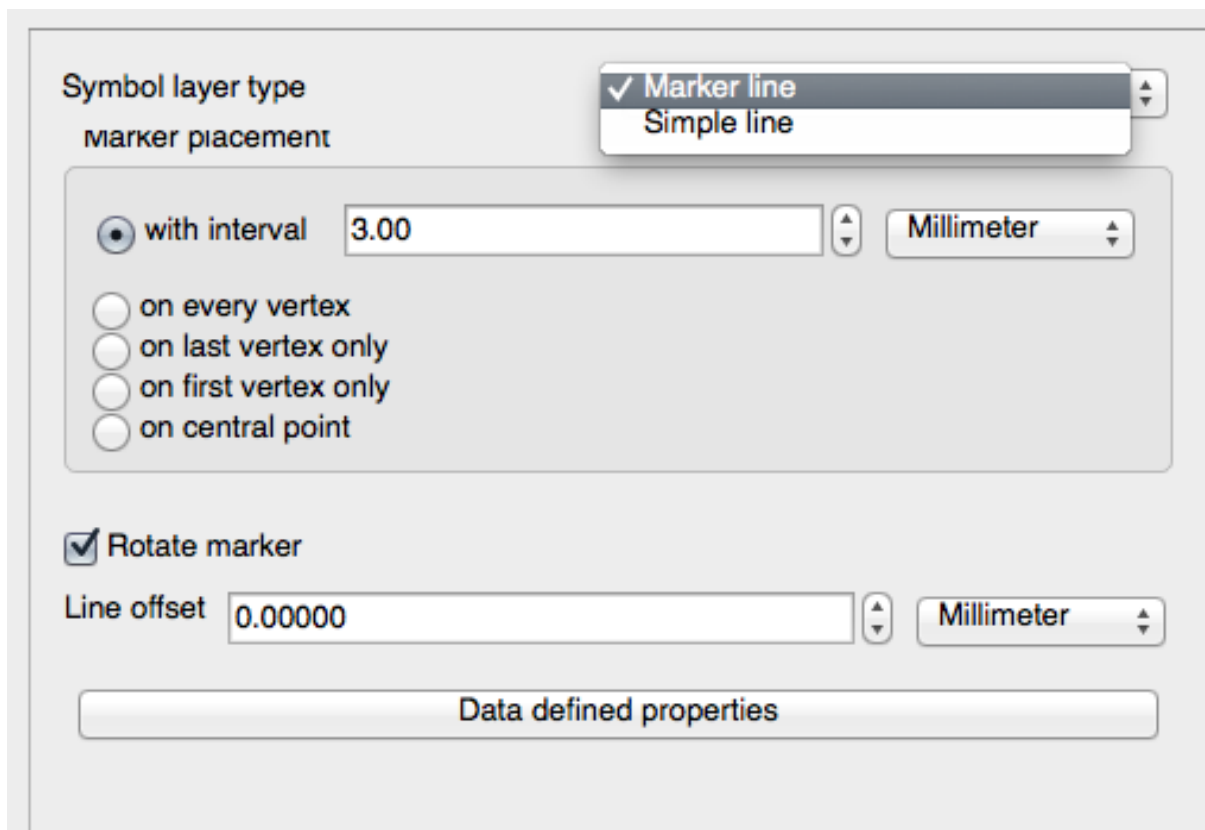


- Investigate the various options available to you, and choose a symbol with styling you think is appropriate.
- If in doubt, use a round *Simple marker* with a white border and pale green fill, with a *size* of 3,00 and an *Outline width* of 0.5.

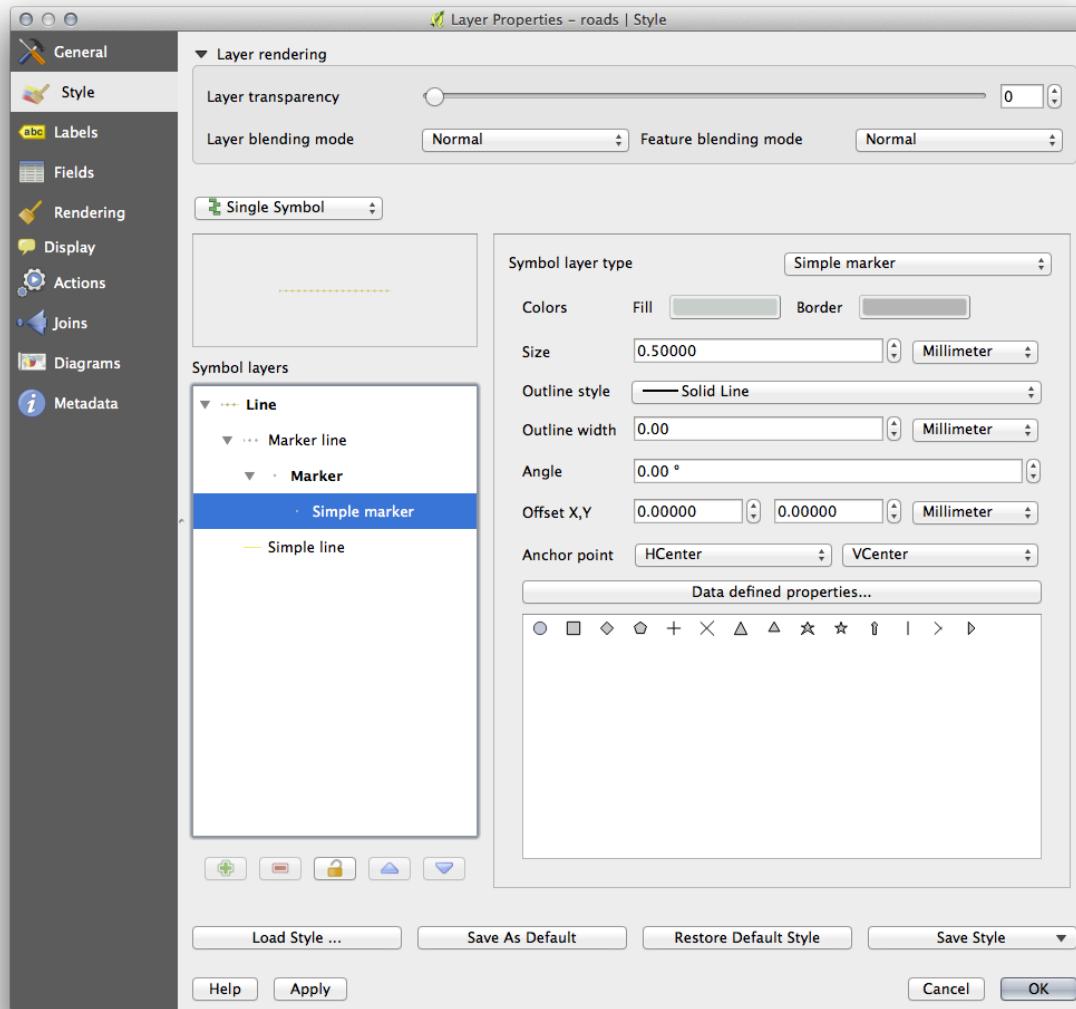
Line Symbol Layer Types

To see the various options available for line data:

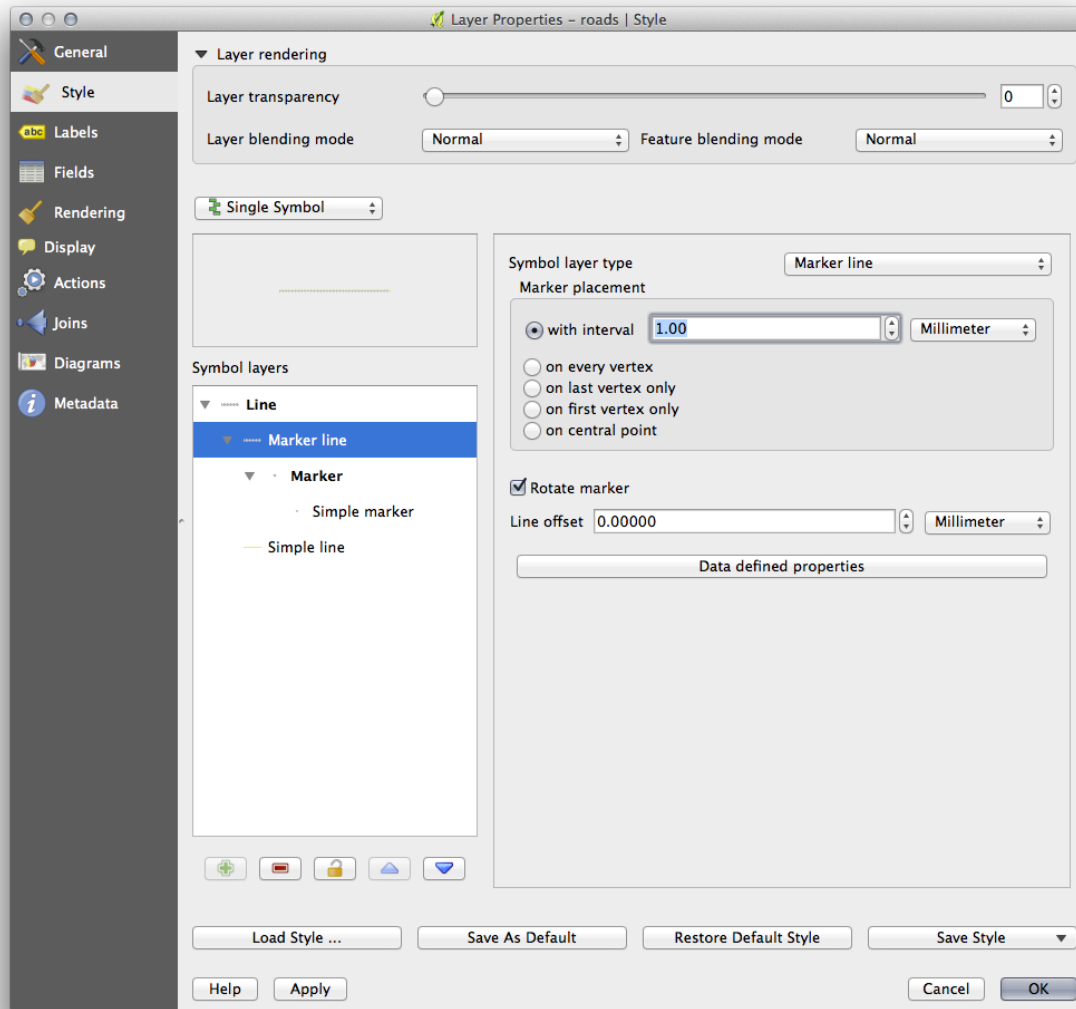
- Change the symbol layer type for the *roads* layer's topmost symbol layer to *Marker line*:



- Select the *Simple marker* layer in the *Symbol layers* panel. Change the symbol properties to match this dialog:



- Change the interval to 1,00:



- Ensure that the symbol levels are correct (via the

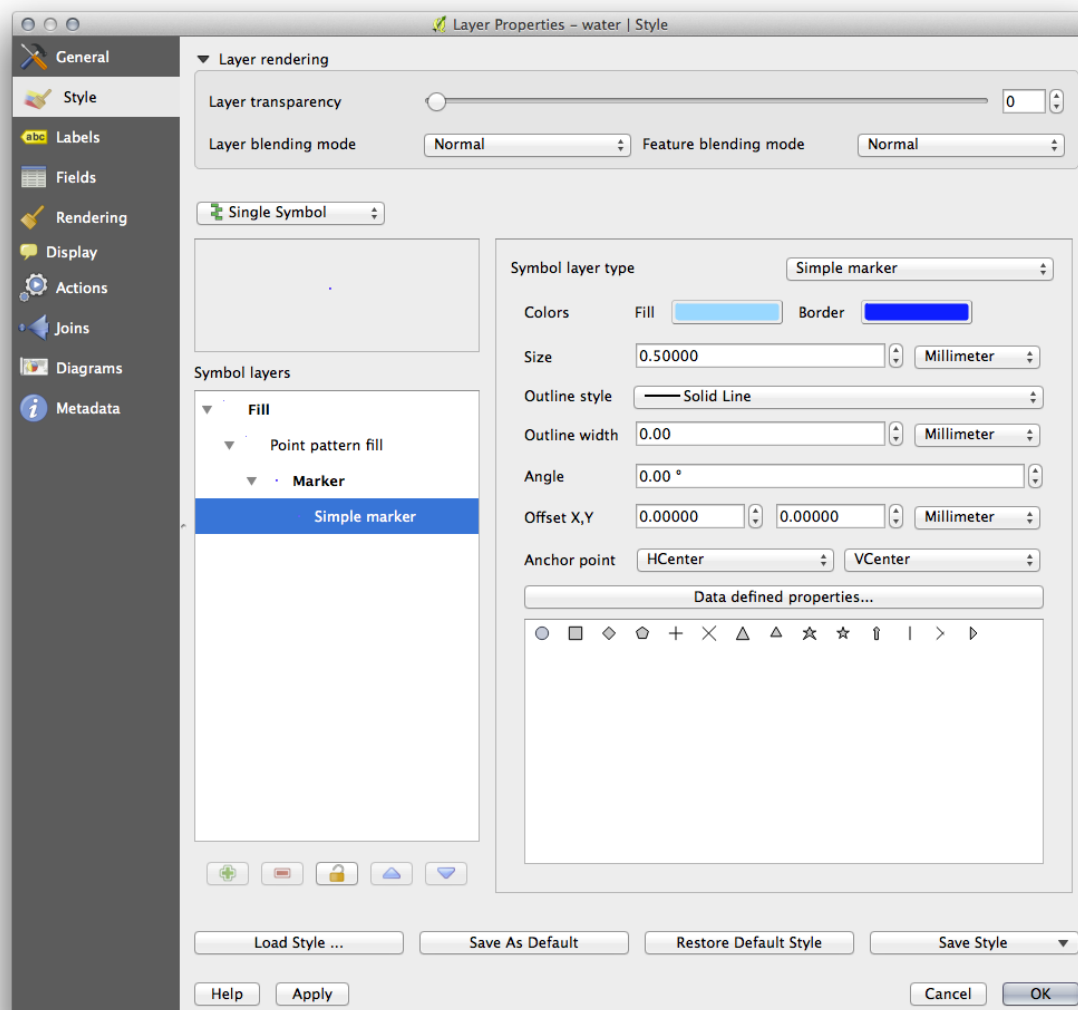
Advanced -> Symbol levels dialog we used earlier) before applying the style.

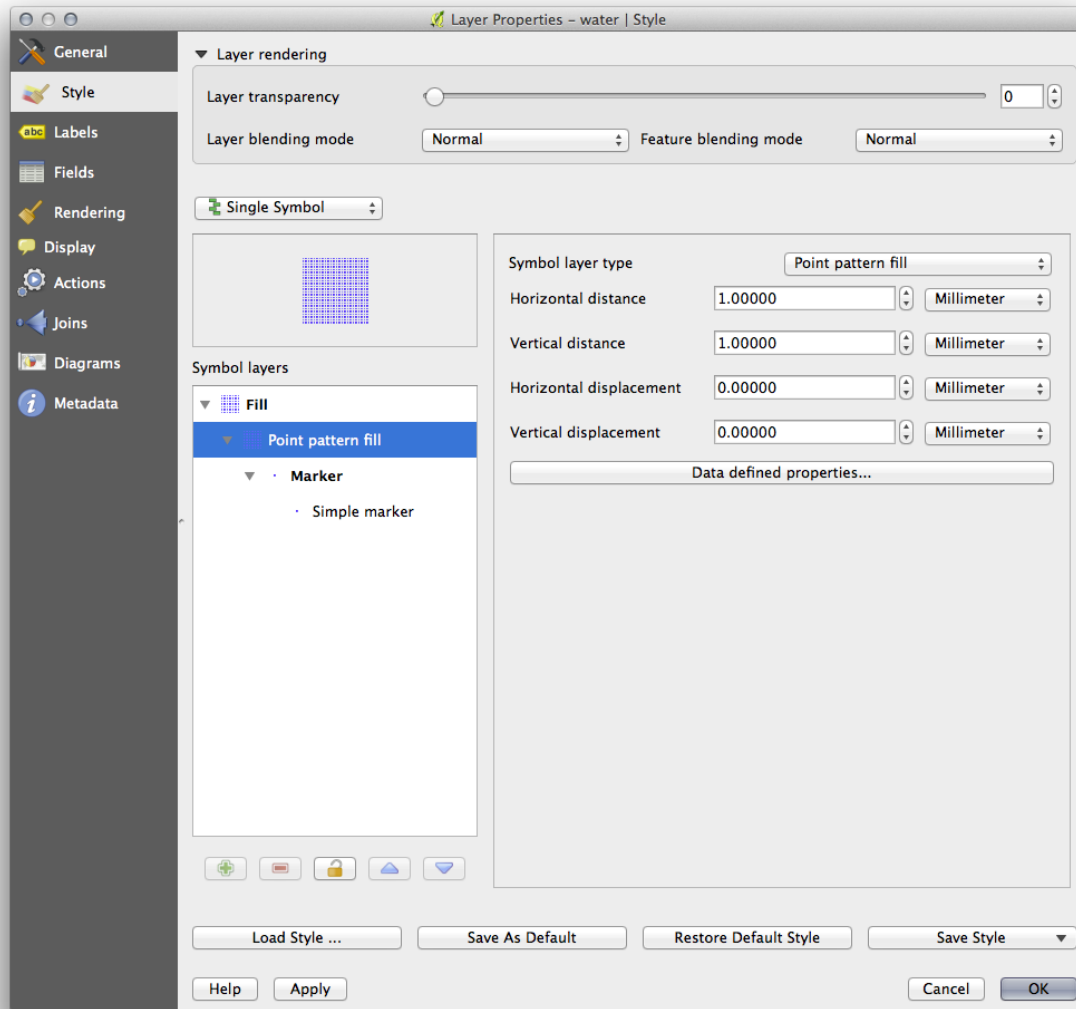
Once you have applied the style, take a look at its results on the map. As you can see, these symbols change direction along with the road but don't always bend along with it. This is useful for some purposes, but not for others. If you prefer, you can change the symbol layer in question back to the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

Polygon Symbol Layer Types

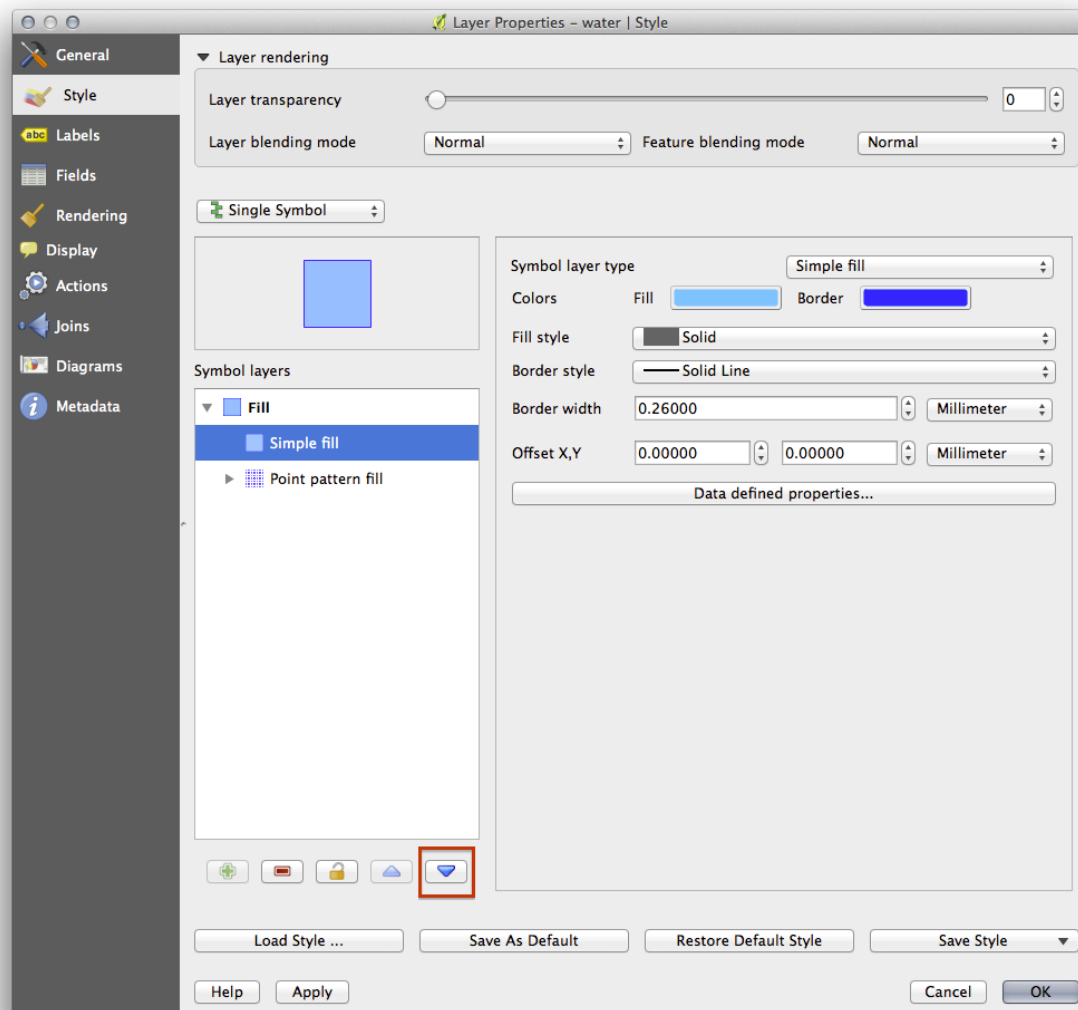
To see the various options available for polygon data:

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





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



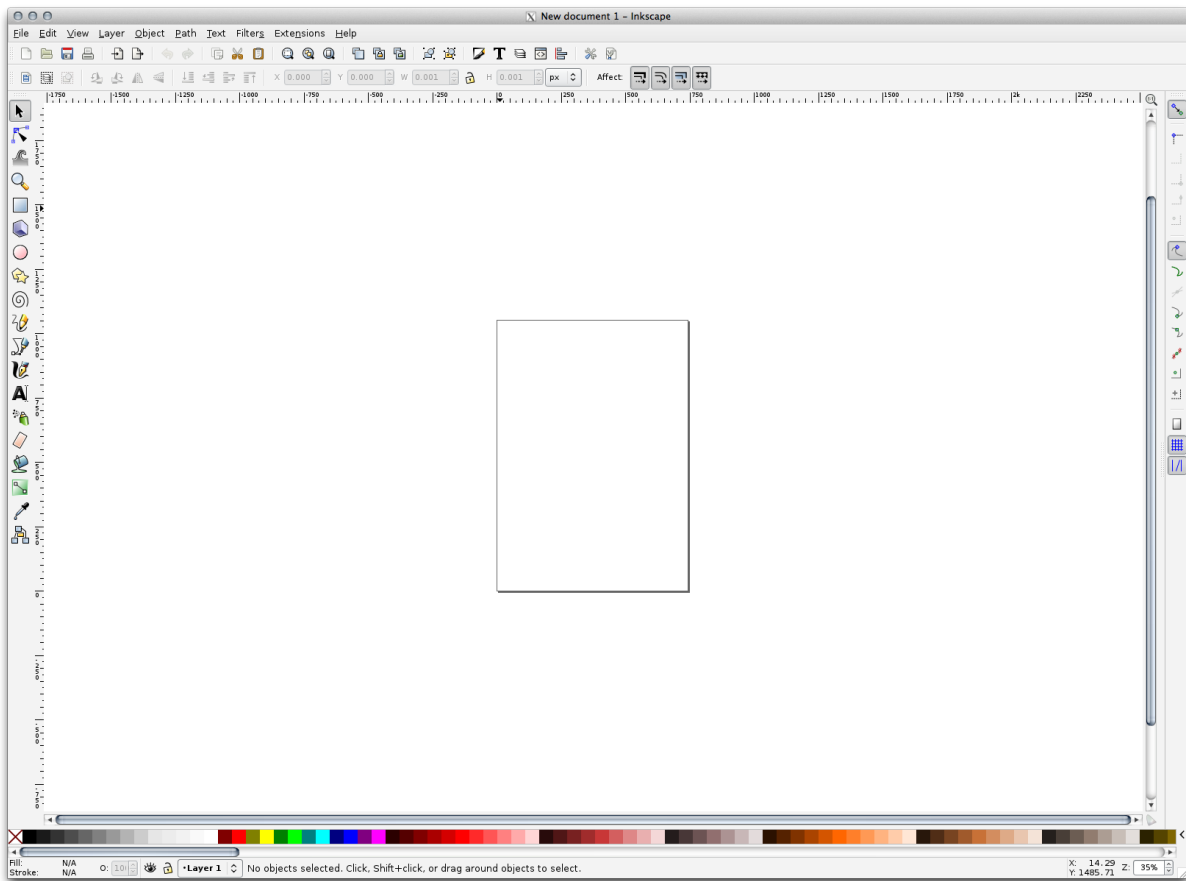
As a result, you have a textured symbol for the water layer, with the added benefit that you can change the size, shape and distance of the individual dots that make up the texture.

3.2.12 Follow Along: Creating a Custom SVG Fill

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

- Start the Inkscape program.

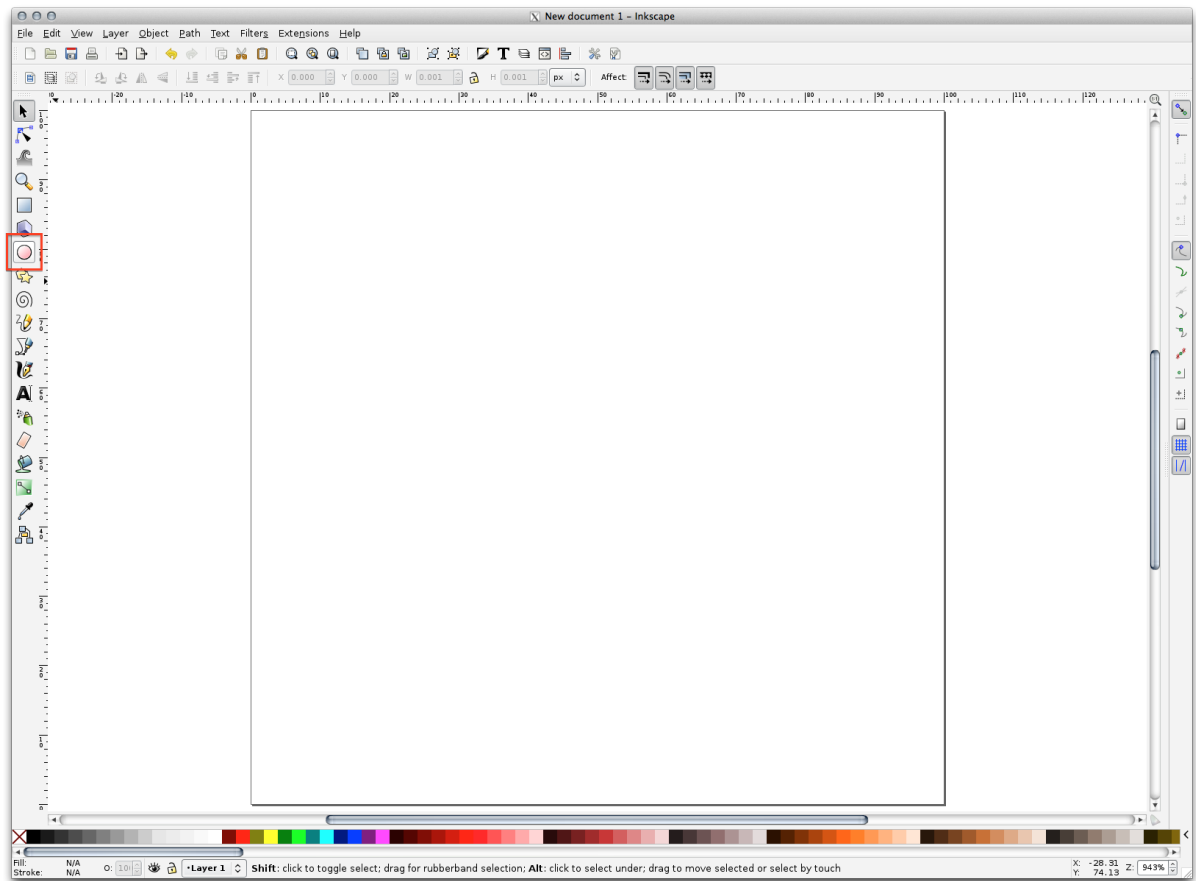
You will see the following interface:



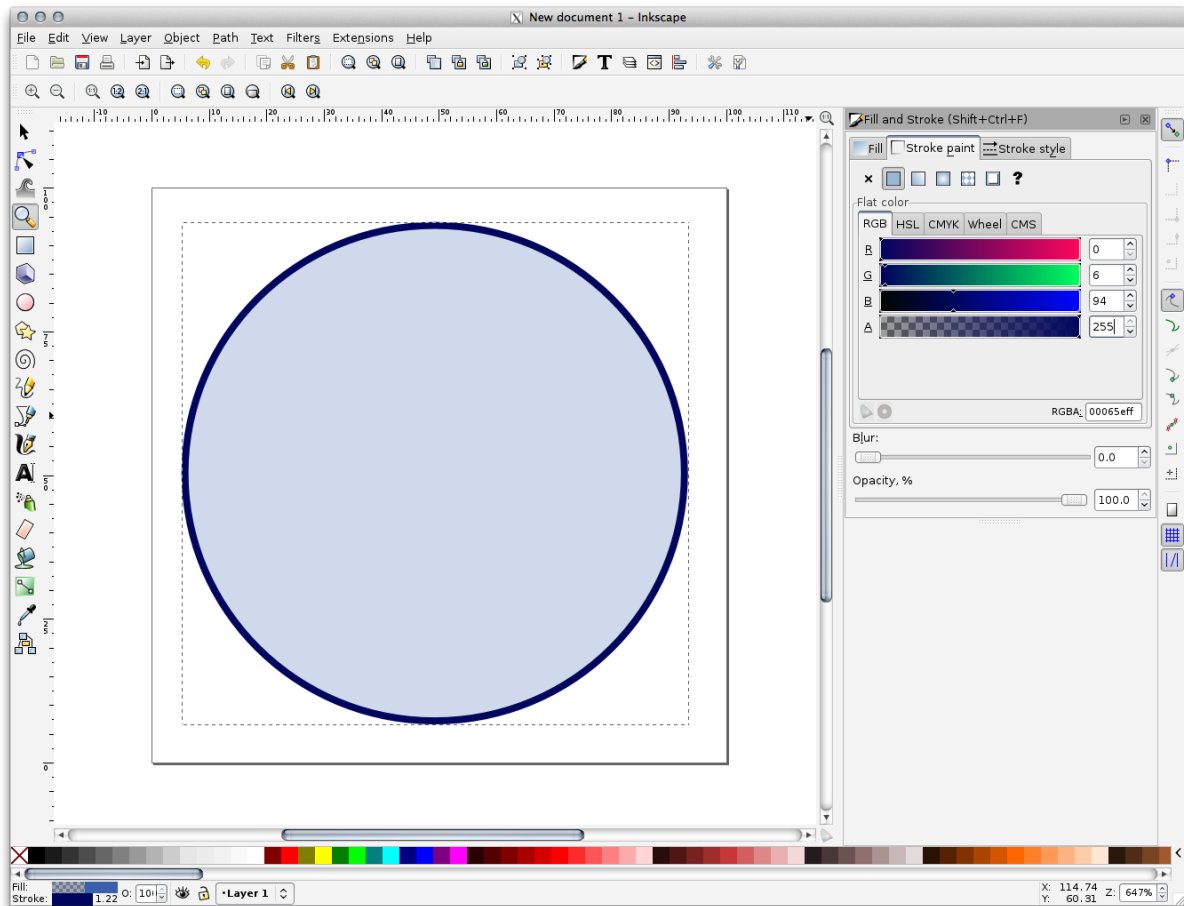
You should find this familiar if you have used other vector image editing programs, like Corel.

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

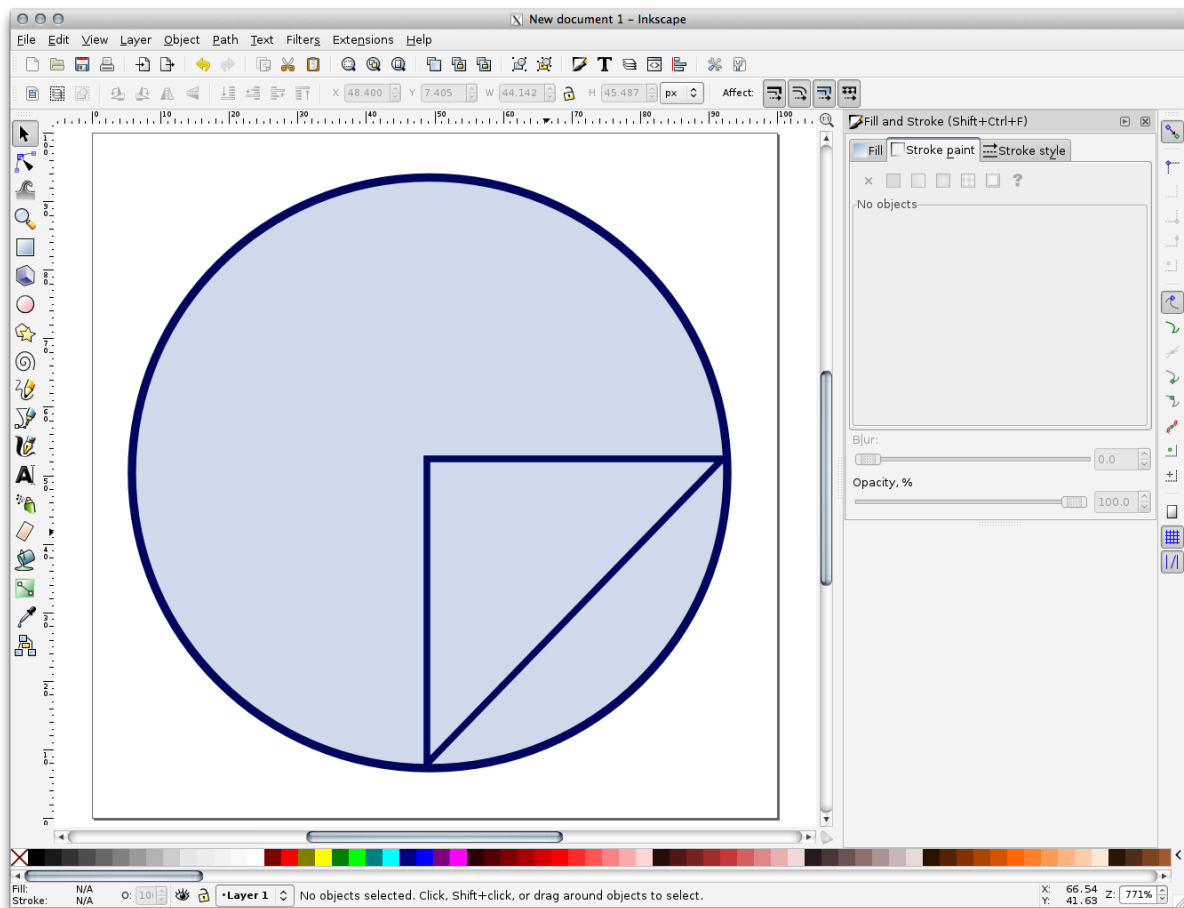
- Click on the menu item *File* → *Document Properties*. This will give you the *Document Properties* dialog.
- Change the *Units* to *px*.
- Change the *Width* and *Height* to 100.
- Close the dialog when you are done.
- Click on the menu item *View* → *Zoom* → *Page* to see the page you are working with.
- Select the *Circle* tool:



- 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.
- Right-click on the circle you just created and open its *Fill and Stroke*:
- Change the *Stroke paint* to a pale grey-blue and the *Stroke style* to a darker color with thin stroke:



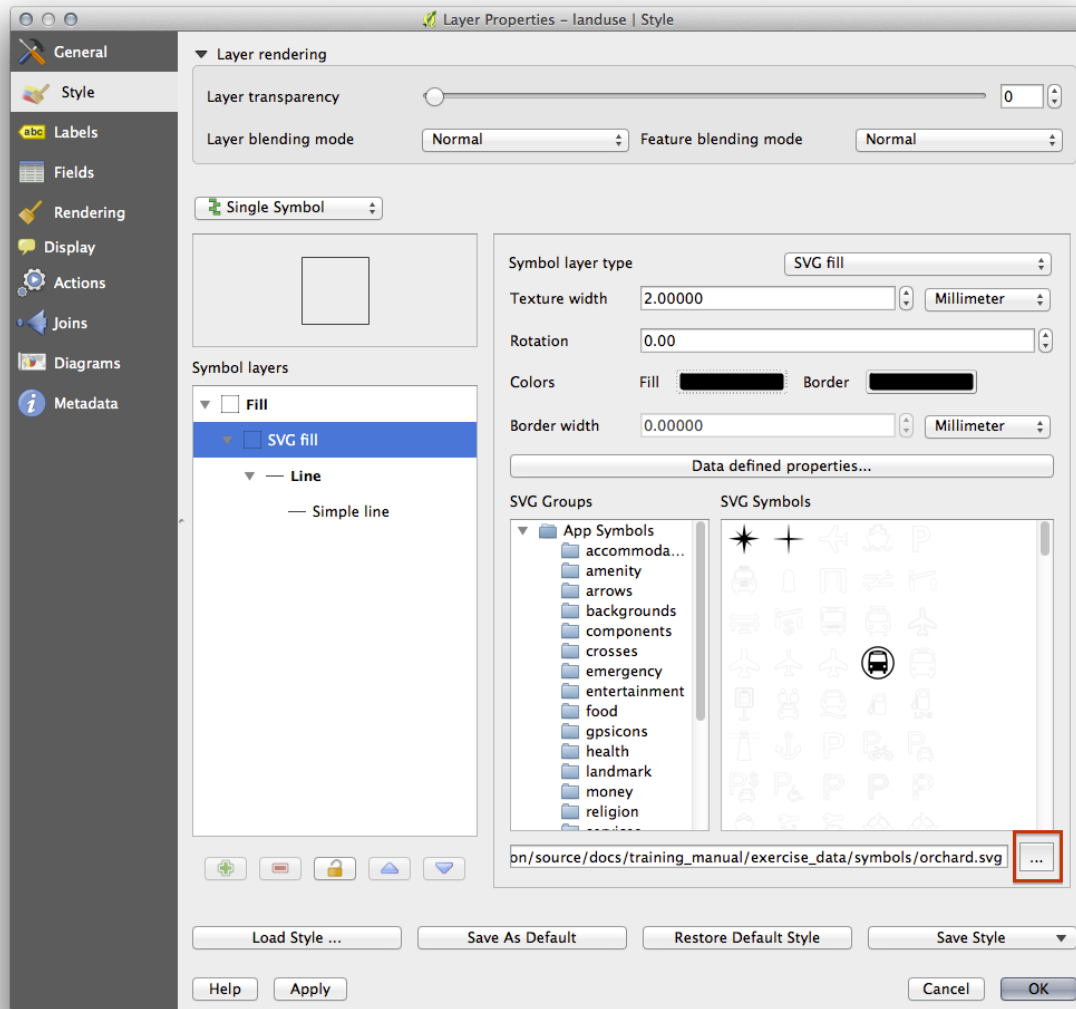
- Draw a line using the *Line* tool:
- Click once to start the line. Hold `ctrl` to make it snap to increments of 15 degrees.
- Click once to end the line segment, then right-click to finalize the line.
- Change its color and width to match the circle's stroke and move it around as necessary, so that you end up with a symbol like this one:



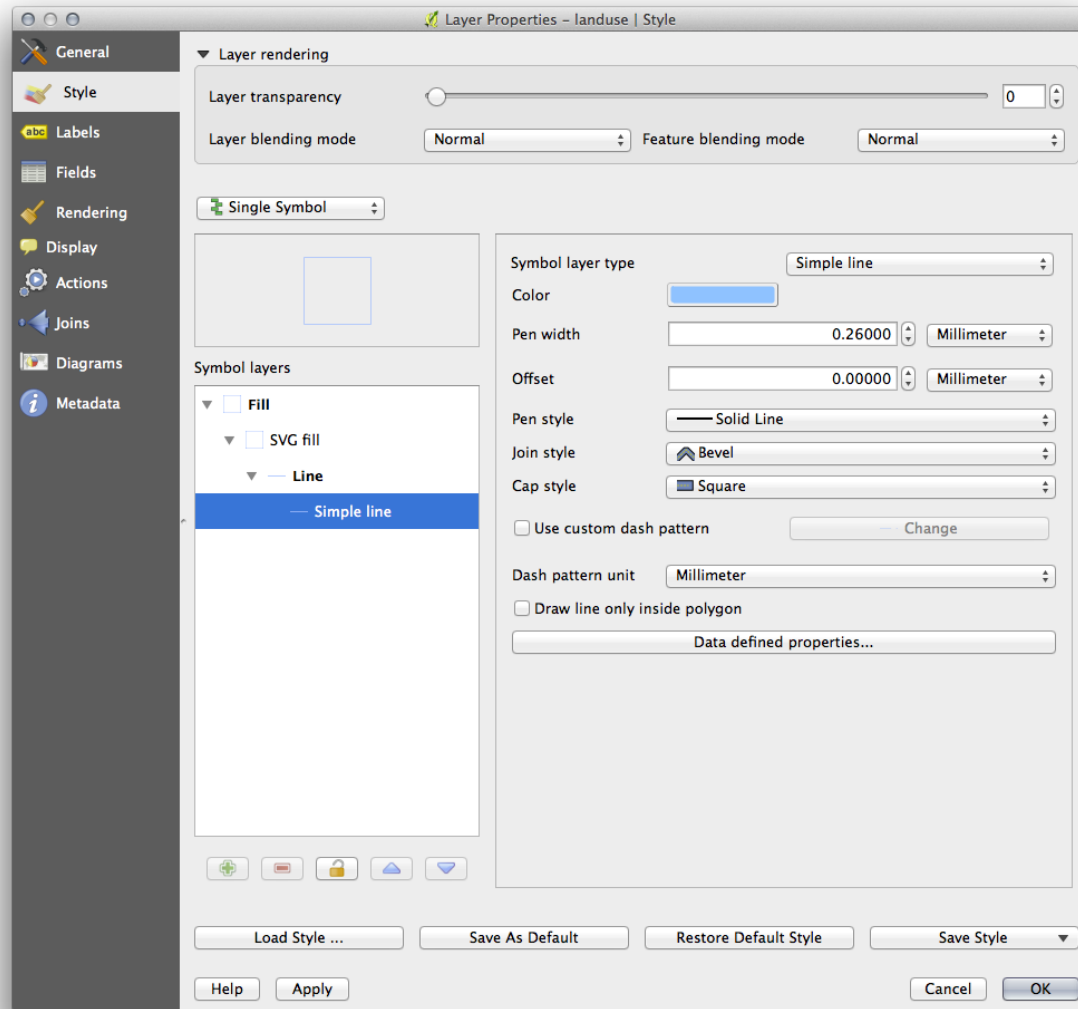
- Save it as *landuse_symbol* under the directory that the course is in, under *exercise_data/symbols*, as an SVG file.

In QGIS:

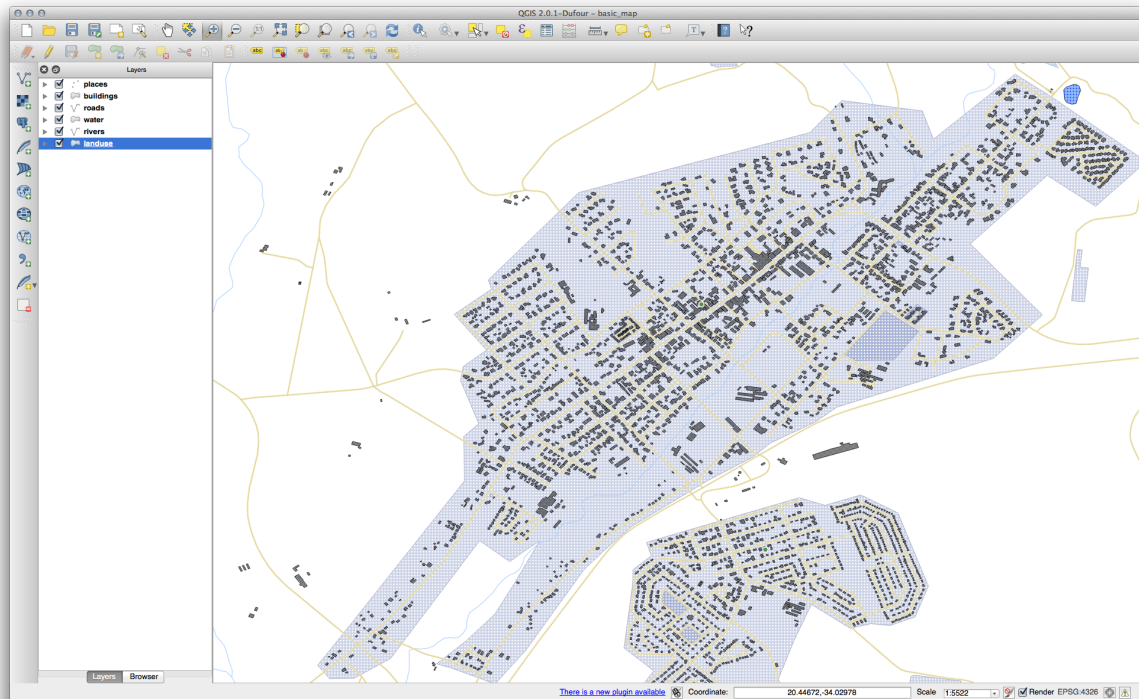
- Open the *Layer Properties* for the *landuse* layer.
- Change the symbol structure to the following and find your SVG image via the *Browse* button:



You may also wish to update the svg layer's border:



Your landuse layer should now have a texture like the one on this map:



3.2.13 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!

3.2.14 Further Reading

Examples of Beautiful Maps

3.2.15 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.

Bemerkung: Did you remember to save your map recently?

Module: Classifying Vector Data

Classifying vector data allows you to assign different symbols to features (different objects in the same layer), depending on their attributes. This allows someone who uses the map to easily see the attributes of various features.

4.1 Lesson: Attribute Data

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 of this lesson: To explore the attribute data of an object and understand what the various data can be useful for.

4.1.1 Follow Along: Attribute data

Open the attribute table for the *places* layer (refer back to the section “*Working with Vector Data*” if necessary). Which field would be the most useful to represent in label form, and why?

Check your results

4.1.2 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.

4.1.3 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.

4.2 Lesson: The Label Tool


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.

Bemerkung: The *Layer Properties* dialog does have a *Labels* tab, which now offers the same functionality, but for this example we'll use the *Label tool*, accessed via a toolbar button.

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

4.2.1 Follow Along: Using Labels

Before being able to access the Label tool, you will need to ensure that it has been activated.

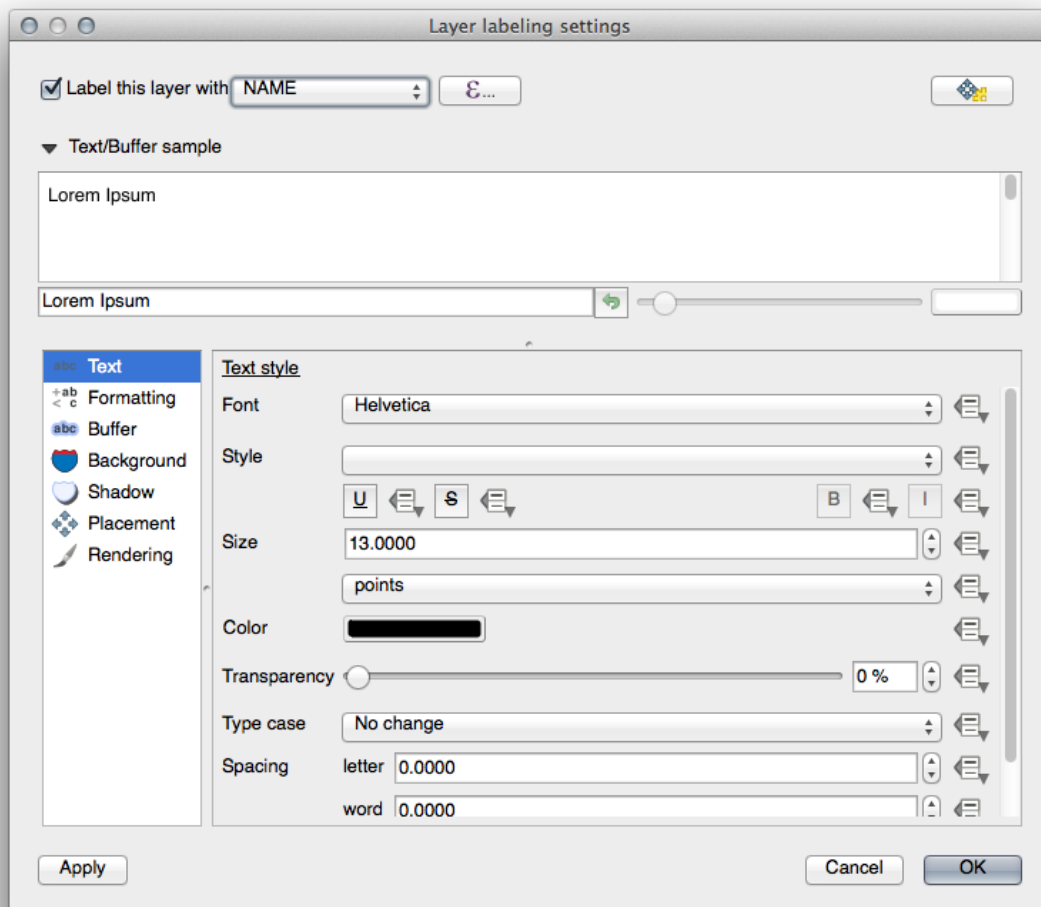
- Go to the menu item *View* → *Toolbars*.
- Ensure that the *Label* item has a check mark next to it. If it doesn't, click on the *Label* item, and it will be activated.
- Click on the *places* layer in the *Layers list*, so that it is highlighted.
- Click on the following toolbar button: 

This gives you the *Layer labeling settings* dialog.

- Check the box next to *Label this layer with...*

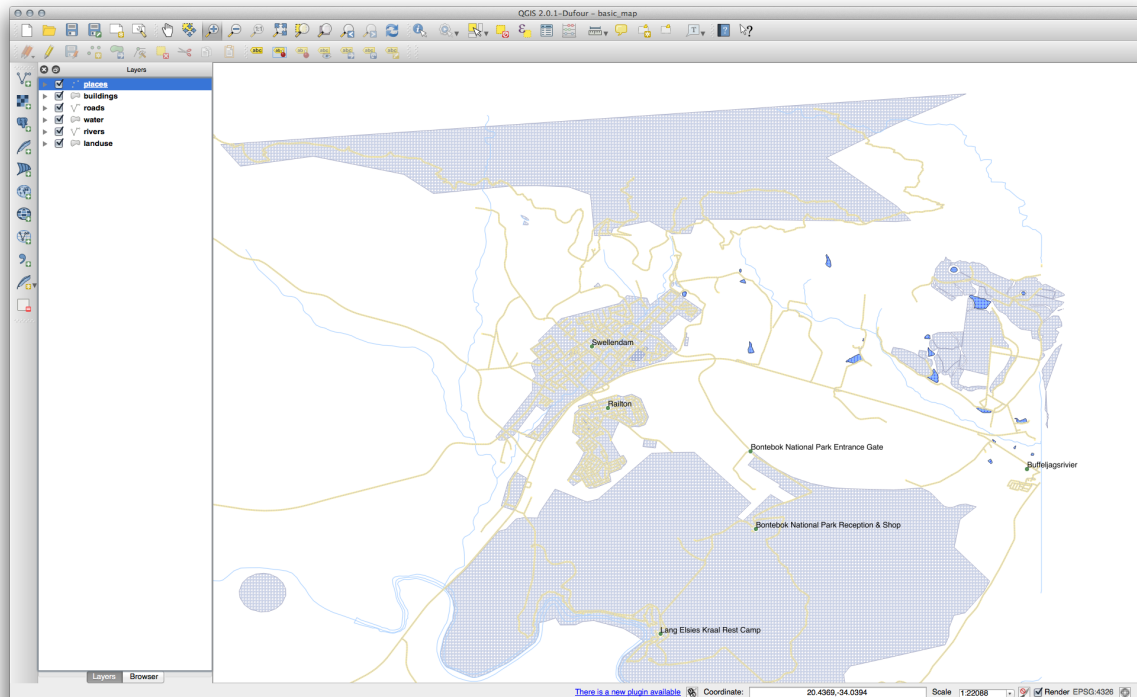
You'll need to choose which field in the attributes will be used for the labels. In the previous lesson, you decided that the `NAME` field was the most suitable one for this purpose.

- Select *name* from the list:



- Click *OK*.

The map should now have labels like this:

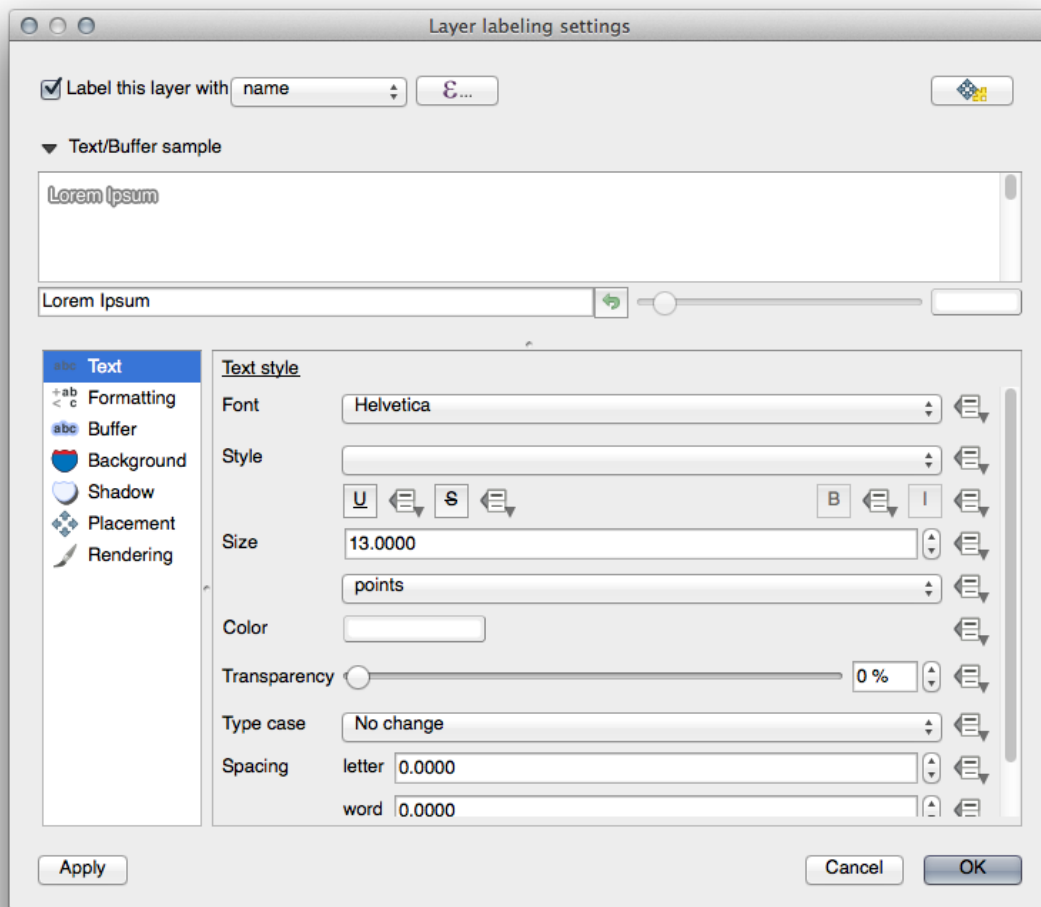


4.2.2 Follow Along: Changing Label Options

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

- Open the *Label tool* again by clicking on its button as before.
- Make sure *Text* is selected in the left-hand options list, then

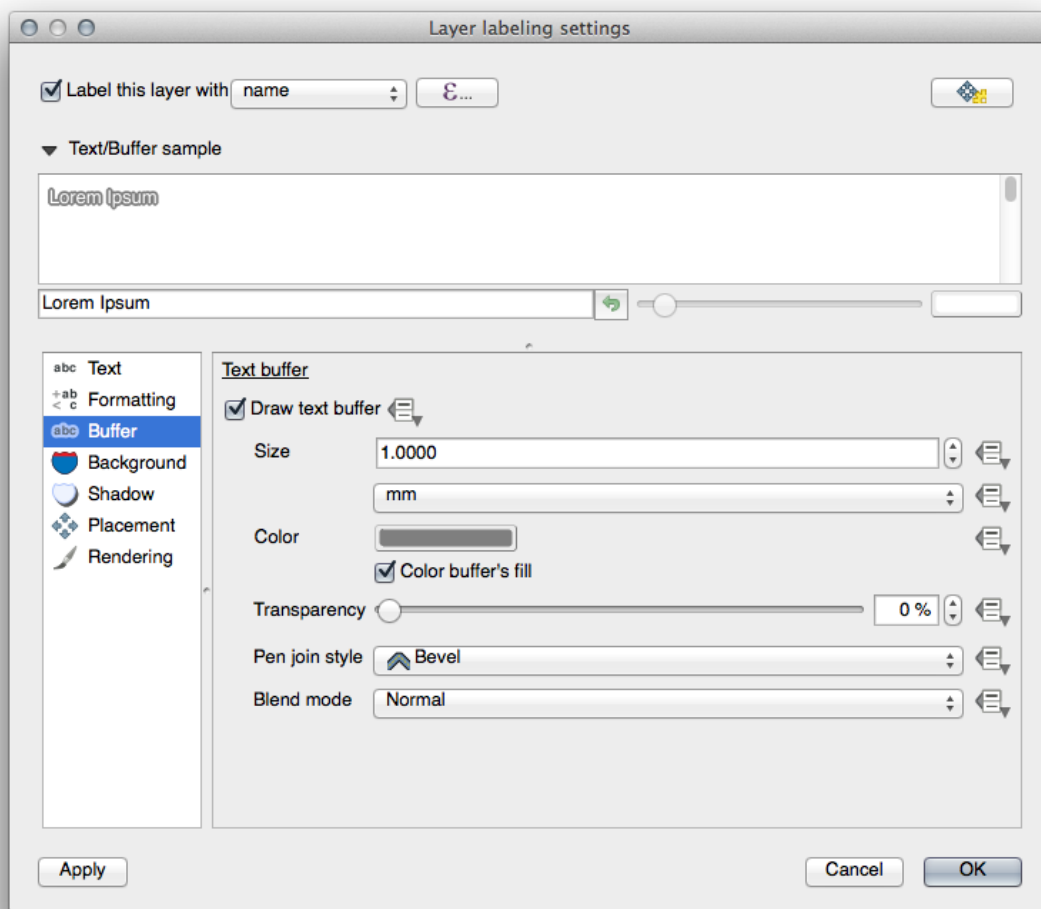
update the text formatting options to match those shown here:



That's the font problem solved! Now let's look at the problem of the labels overlapping the points, but before we do that, let's take a look at the *Buffer* option.

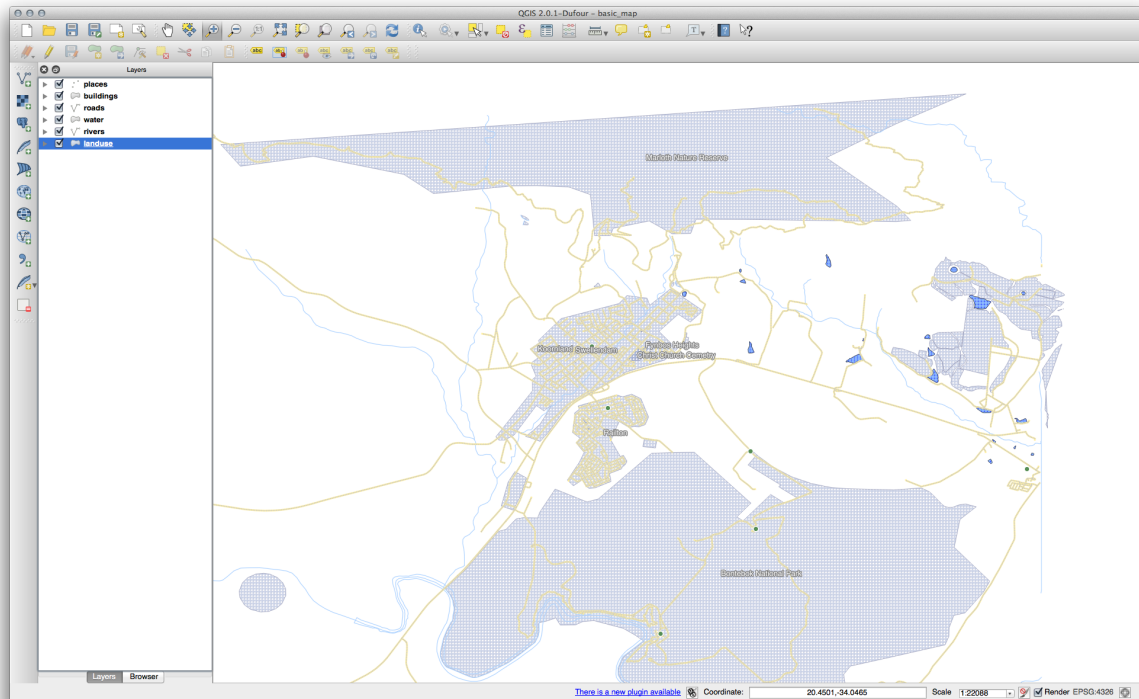
- Open the *Label tool* dialog.
- Select *Buffer* from the left-hand options list.
- Select the checkbox next to *Draw text buffer*, then choose options

to match those shown here:



- Click *Apply*.

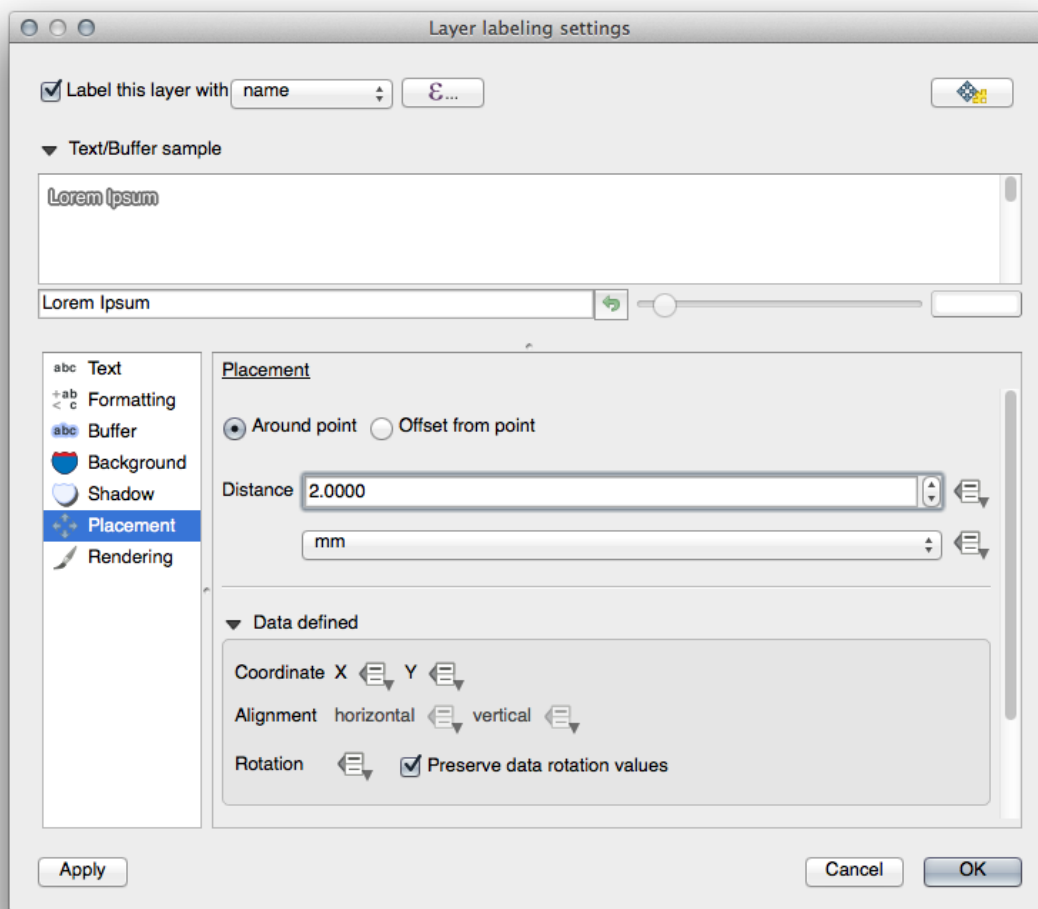
You'll see that this adds a colored buffer or border to the place labels, making them easier to pick out on the map:



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

- In the *Label tool* dialog, go to the *Placement* tab.
- Change the value of *Distance* to 2mm and make sure that

Around point is selected:



- Click *Apply*.

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

4.2.3 Follow Along: Using Labels Instead of Layer Symbolology

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.

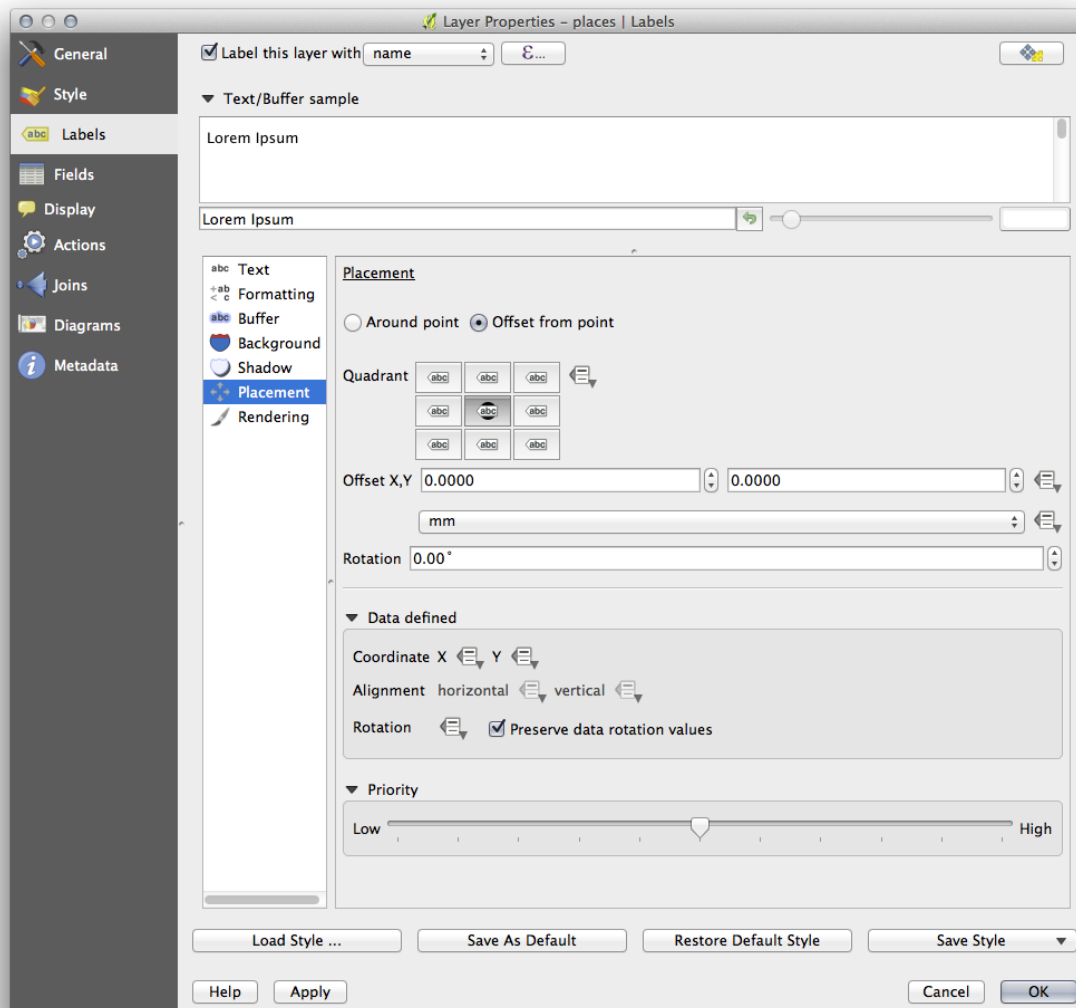
To name an example: on a map of the world, the point given for the European Union may be somewhere in Poland, for instance. To someone reading the map, seeing a point labeled *European Union* in Poland, it may seem that the capital of the European Union is therefore in Poland.

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

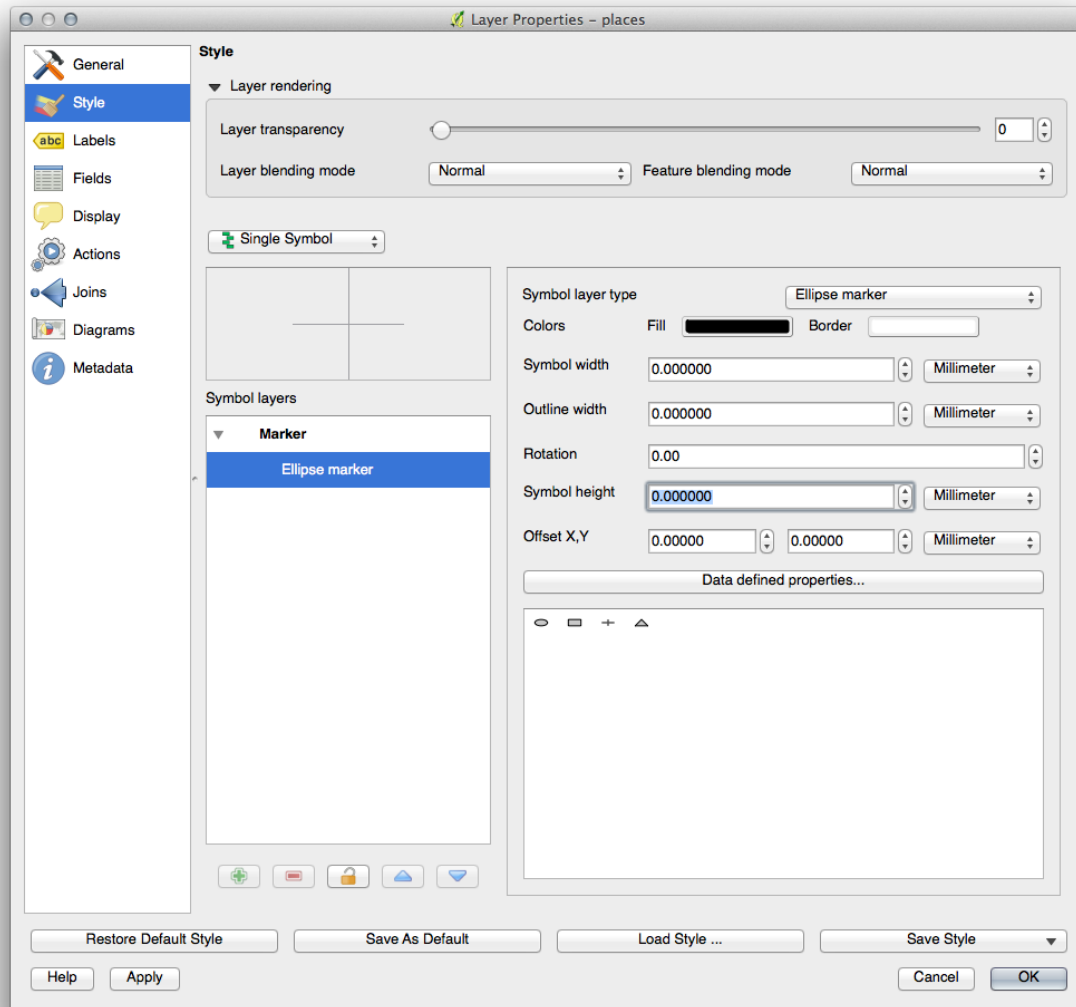
In QGIS, you can do this by changing the position of the labels to be rendered directly over the points they refer to.

- Open the *Layer labeling settings* dialog for the *places* layer.
- Select the *Placement* option from the options list.
- Click on the *Offset from point* button.

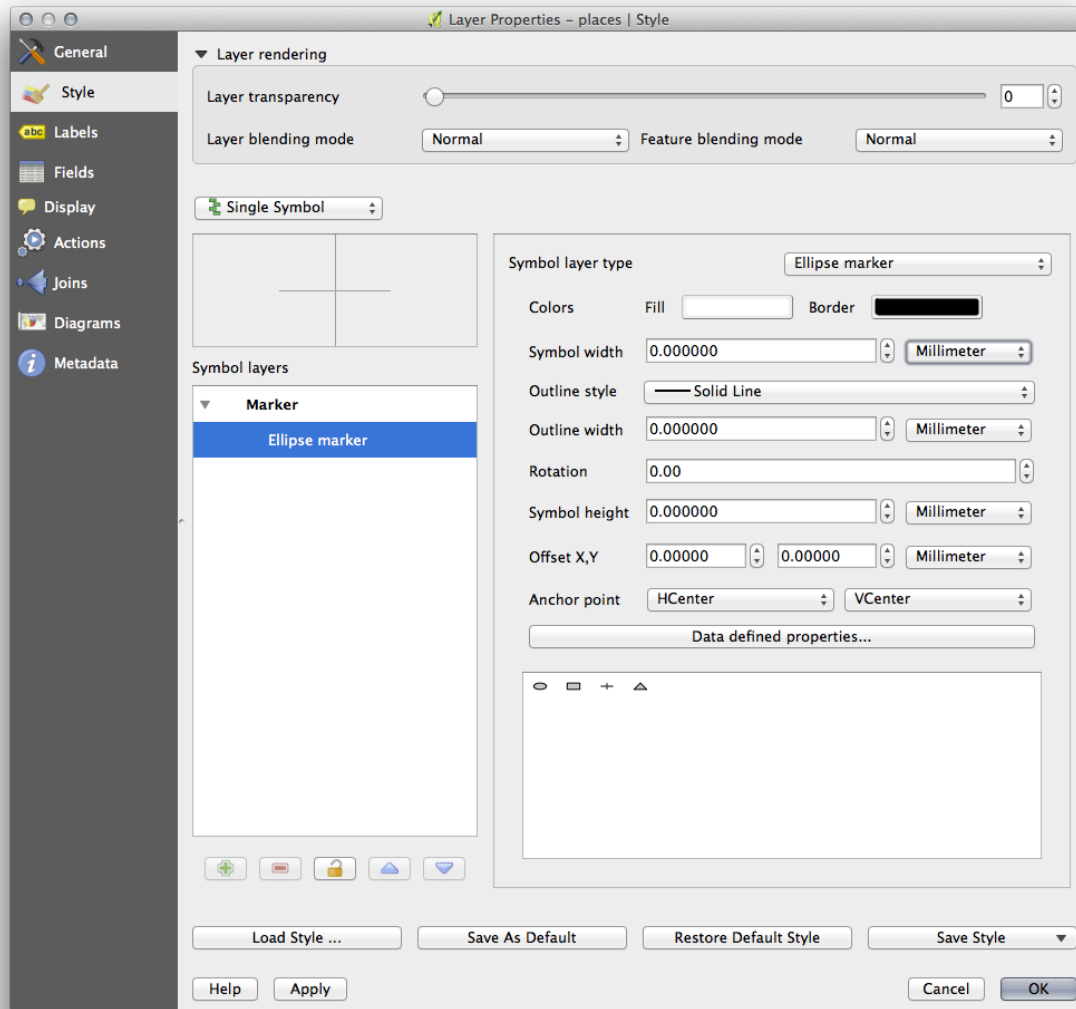
This will reveal the *Quadrant* options which you can use to set the position of the label in relation to the point marker. In this case, we want the label to be centered on the point, so choose the center quadrant:



- Hide the point symbols by editing the layer style as usual, and setting the size of the *Ellipse marker* width and height to 0:



- Click *OK* 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.

4.2.4 Try Yourself Customize the Labels

- Return the label and symbol settings to have a point marker and a label offset of 2.00mm. You may like to adjust the styling of the point marker or labels at this stage.

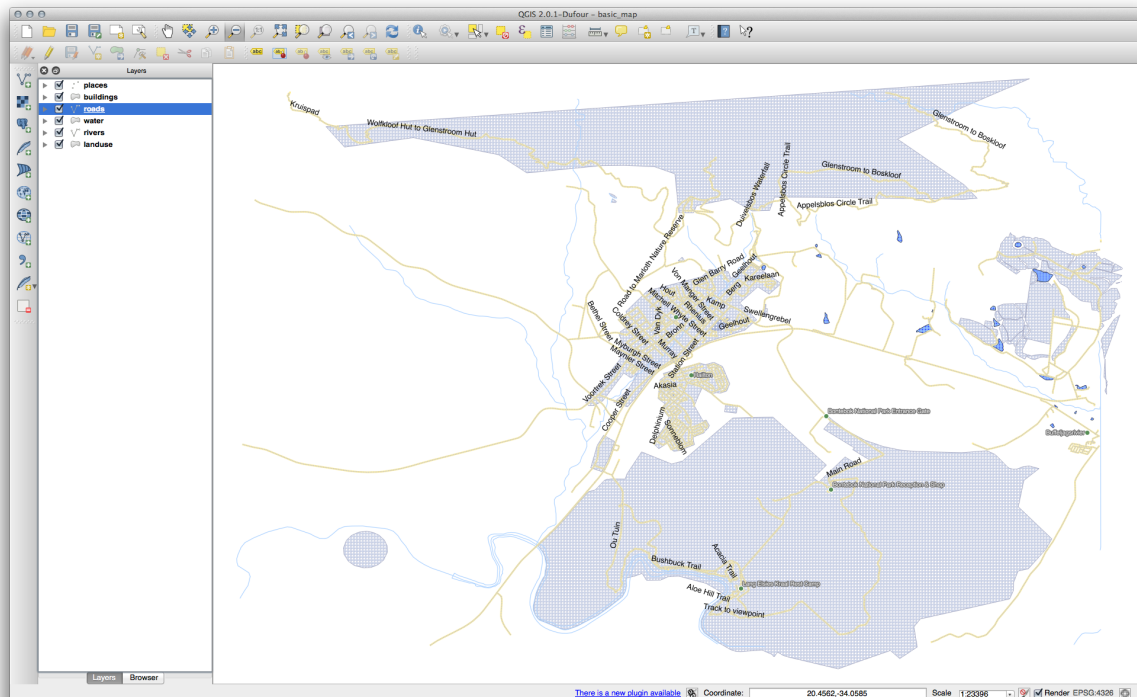
Check your results

- 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.

Check your results

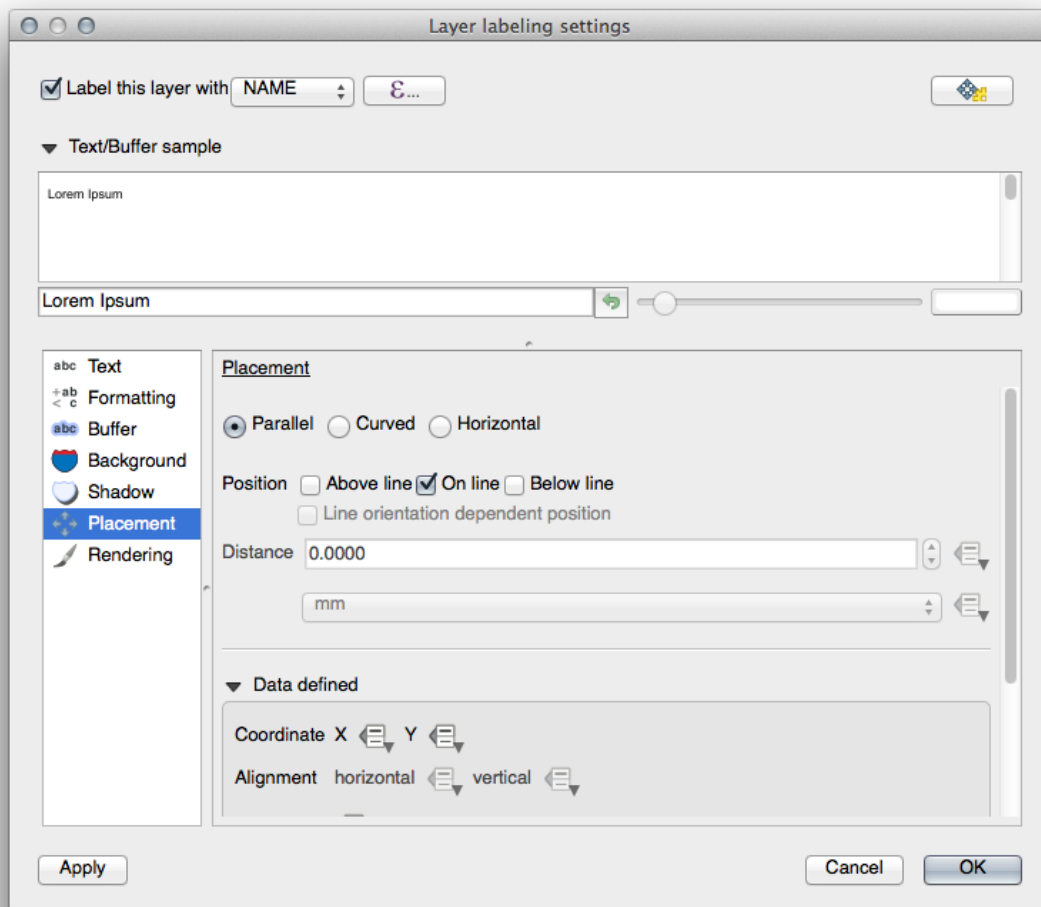
4.2.5 Follow Along: Labeling Lines

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:



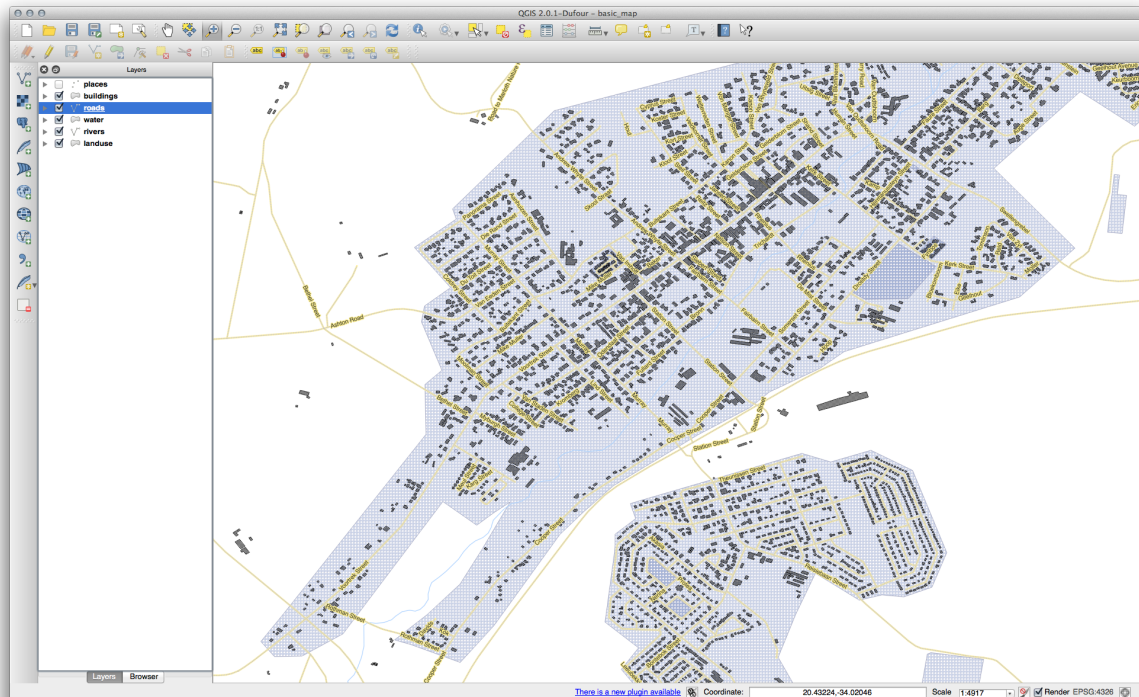
We will now reformat the *roads* layer labels so that they are easy to understand.

- Hide the *Places* layer so that it doesn't distract you.
- Activate labels for the *streets* layer as before.
- Set the font *Size* to 10 so that you can see more labels.
- Zoom in on the Swellendam town area.
- In the *Label tool* dialog's *Advanced* tab, choose the following settings:



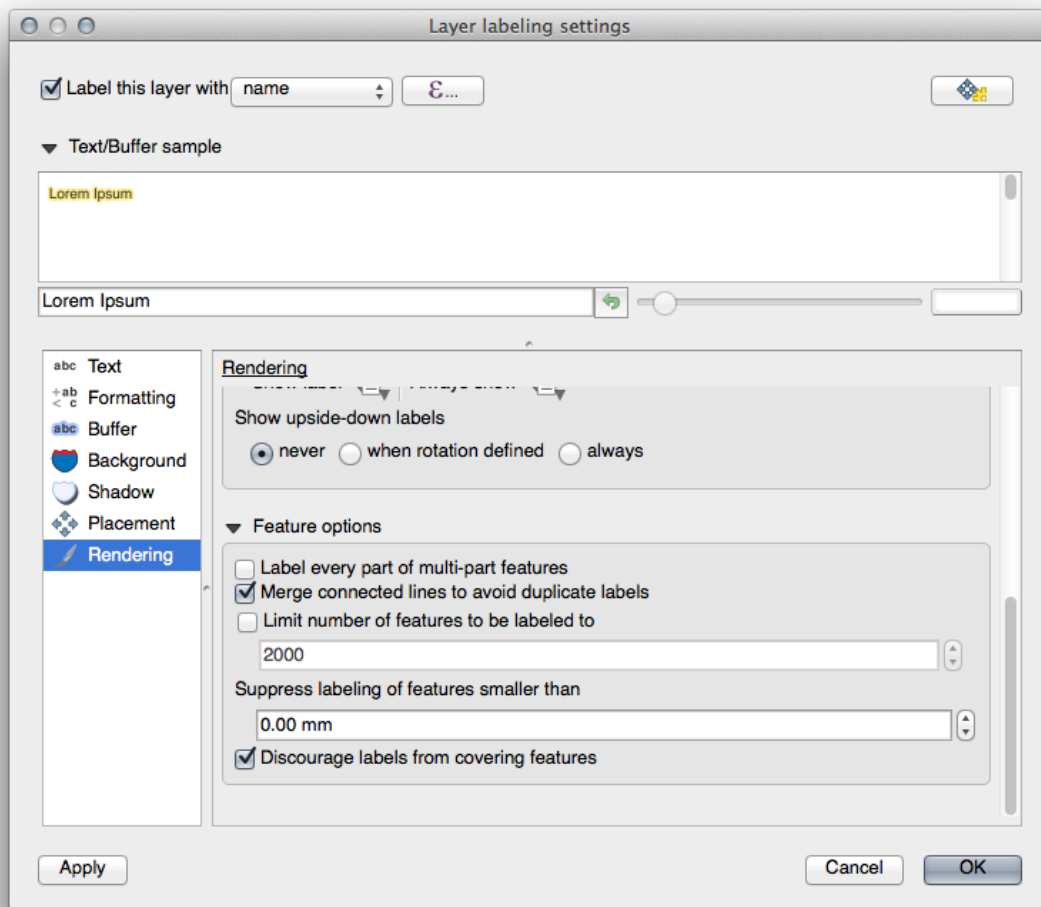
You'll probably find that the text styling has used default values and the labels are consequently very hard to read. Set the label text format to have a dark-grey or black Color and a light-yellow buffer.

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:

- In the *Label labelling settings* dialog, choose the *Rendering* option and select the *Merge connected lines* to avoid duplicate labels:



- Click *OK*

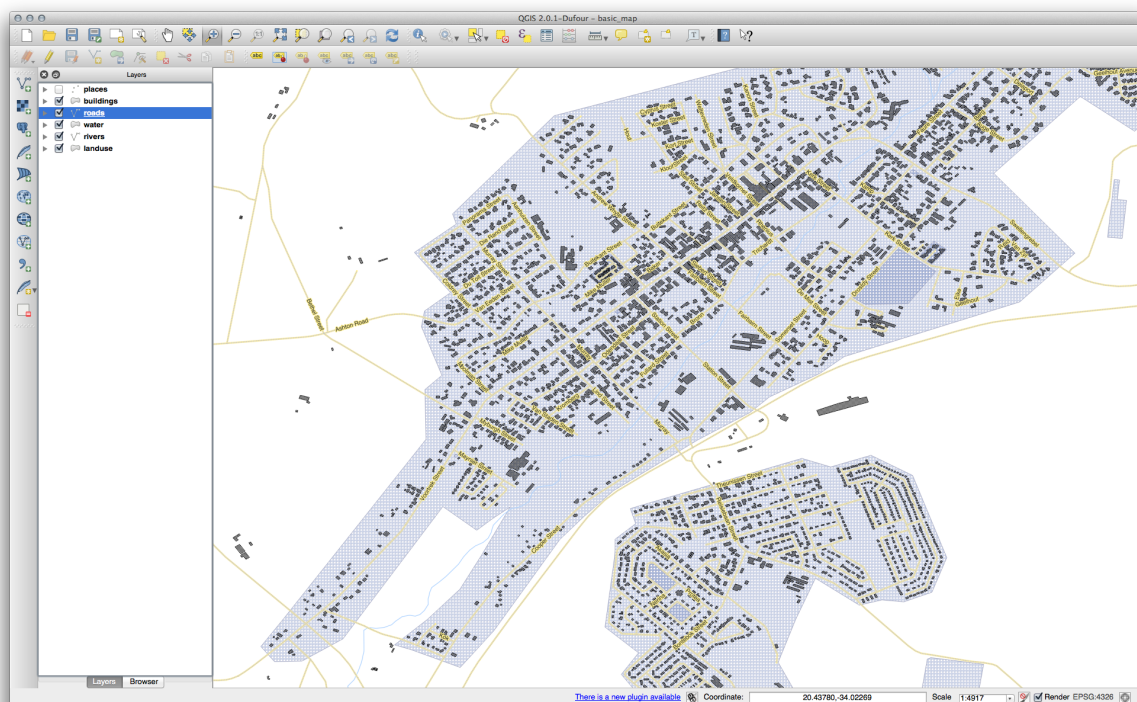
Another useful function is to prevent labels being drawn for features too short to be of notice.

- In the same *Rendering* panel, set the value of *Suppress labeling of features smaller than ...* to 5mm 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.


- Select the *Curved* option in the *Placement* panel of the *Layer labeling settings* dialog.

Here's the result:



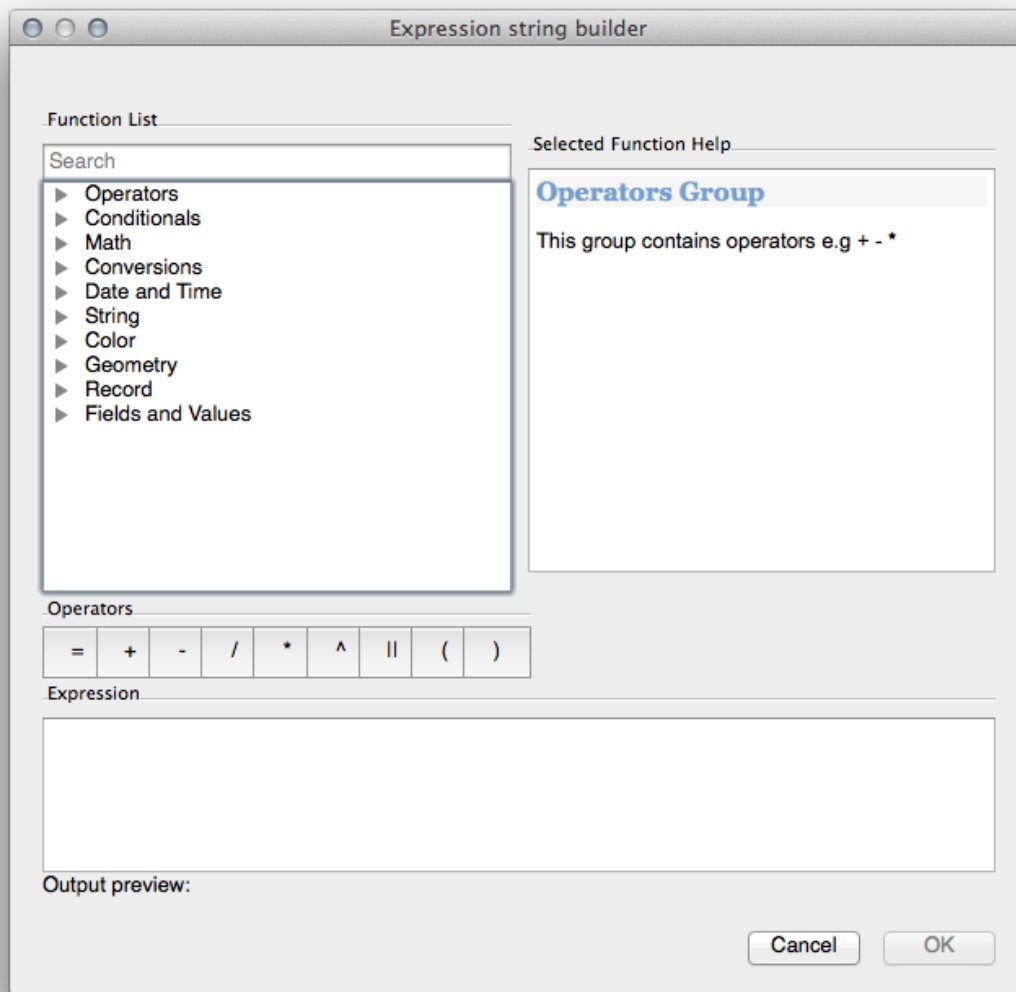
As you can see, this hides a lot of the labels that were previously visible, because of the difficulty of making some of them follow twisting street lines and still be legible. You can decide which of these options to use, depending on what you think seems more useful or what looks better.

4.2.6 Follow Along: Data Defined Settings

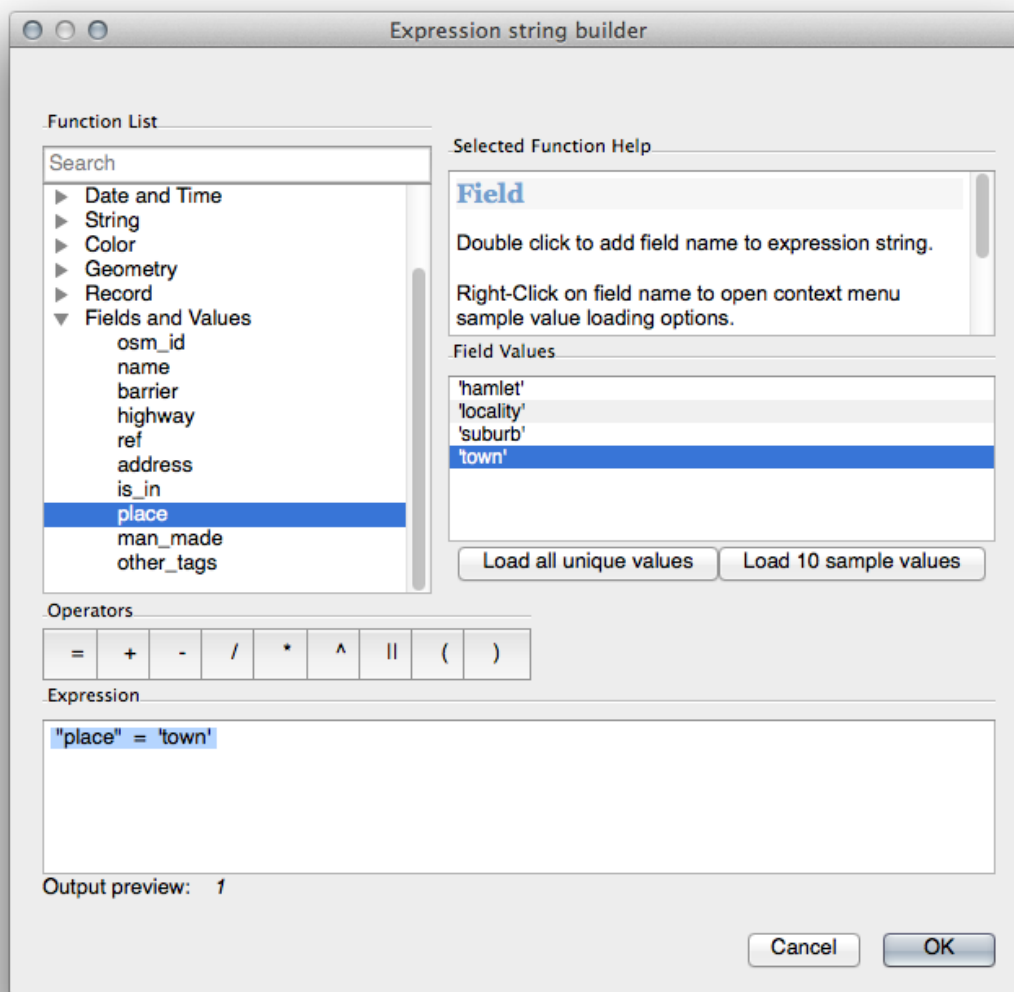
- Deactivate labeling for the *Streets* layer.
- Reactivate labeling for the *Places* layer.
- Open the attribute table for *Places* via the  button.

It has one fields which is of interest to us now: *place* which defines the type of urban area for each object. We can use this data to influence the label styles.

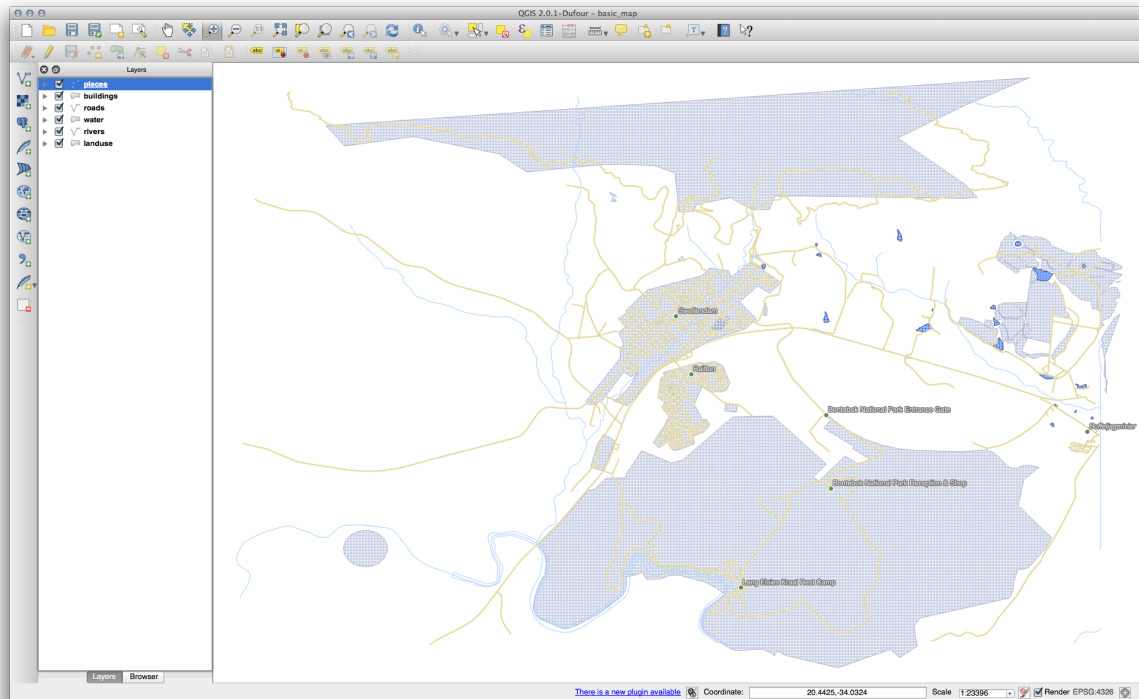
- Navigate to the *Text* panel in the *places Labels* panel.
- In the *Italic* dropdown, select *Edit . . .* to open the *Expression string builder*:



In the text input, type: "place" = 'town' and click *Ok* twice:




Notice its effects:



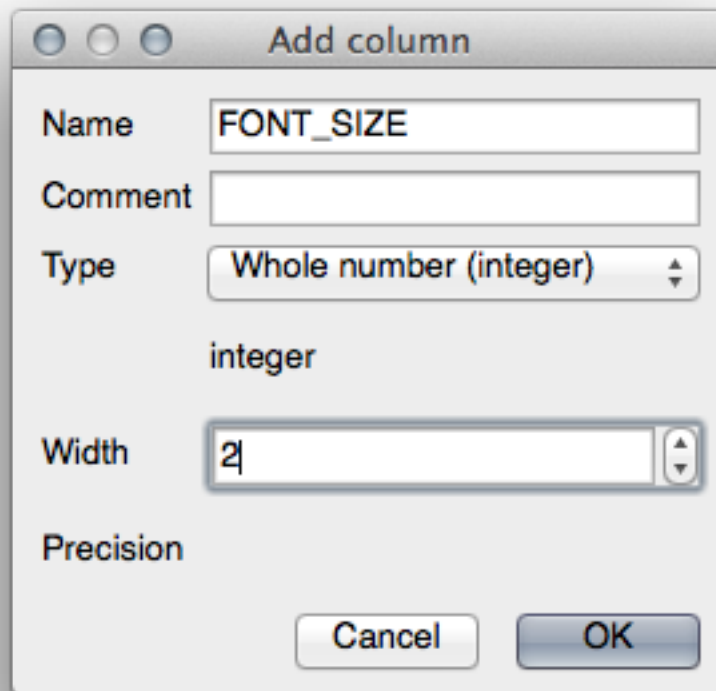
4.2.7 Try Yourself Using Data Defined Settings

Bemerkung: 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.

- Open the Attribute Table for *places*.
- Enter edit mode by clicking this button: 
- Add a new column:



- Configure it like this:



- Use this to set custom font sizes for each different type of place (i.e., each key in the `PLACE` field).

Check your results

4.2.8 Further Possibilities With Labeling

We can't cover every option in this course, but be aware that the *Label tool* 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 *Label tool*:



(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.

4.2.9 In Conclusion

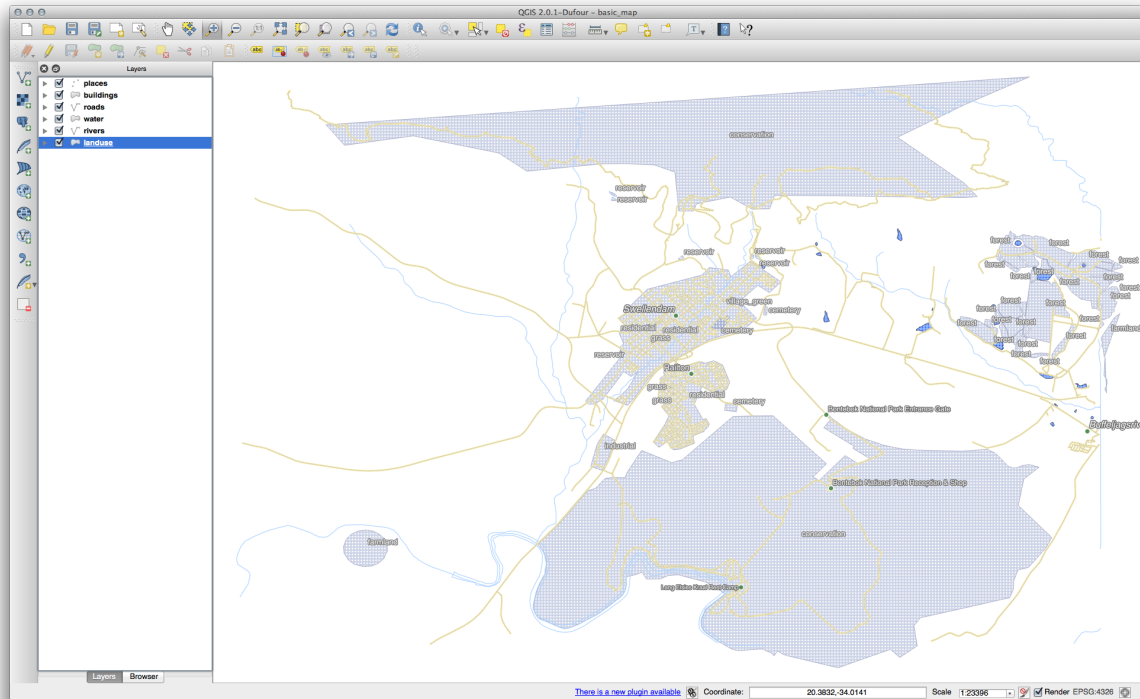
You've learned how to use layer attributes to create dynamic labels. This can make your map a lot more informative and stylish!

4.2.10 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!

4.3 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's say that someone wants to know what each *landuse* area is used for. Using labels, you'd get this:



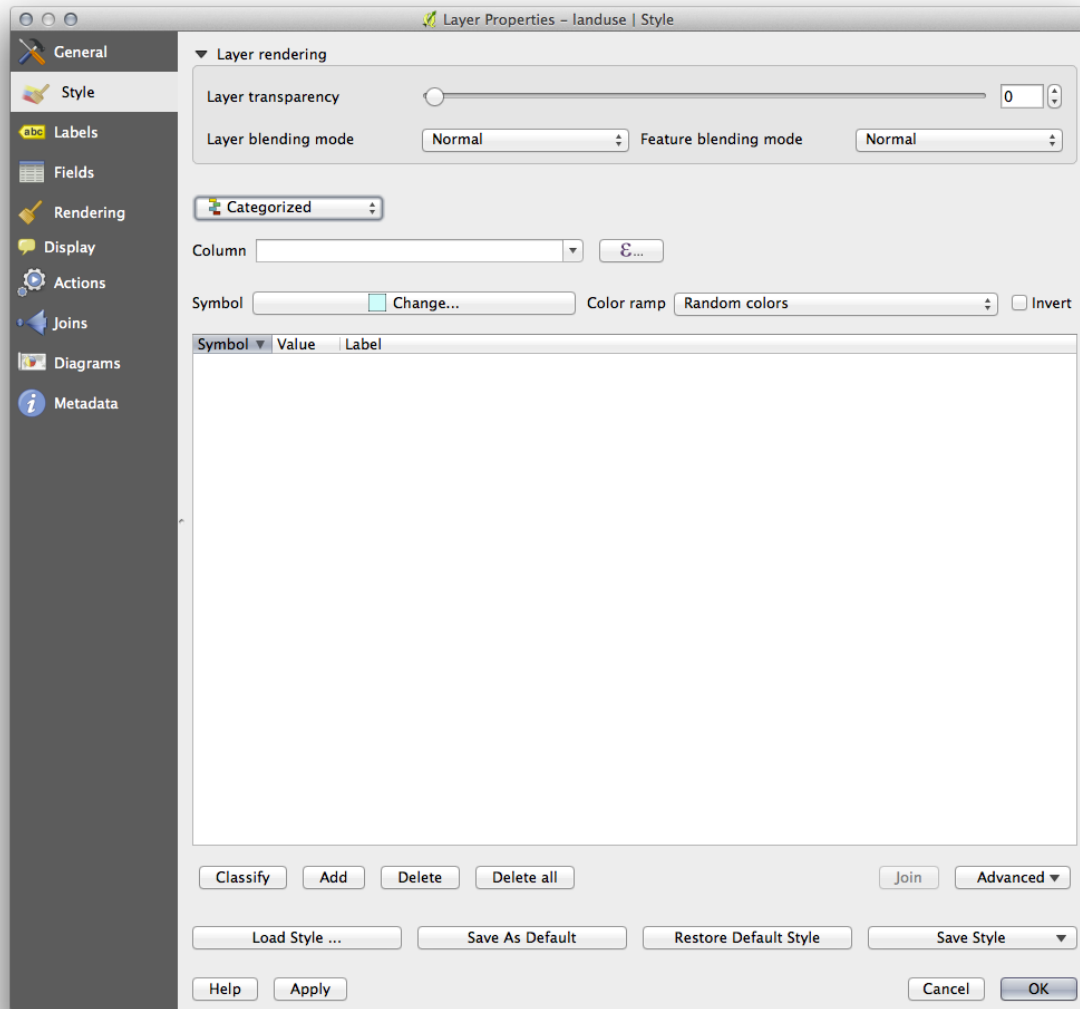
This makes the map's labeling difficult to read and even overwhelming if there are numerous different landuse areas on the map.

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

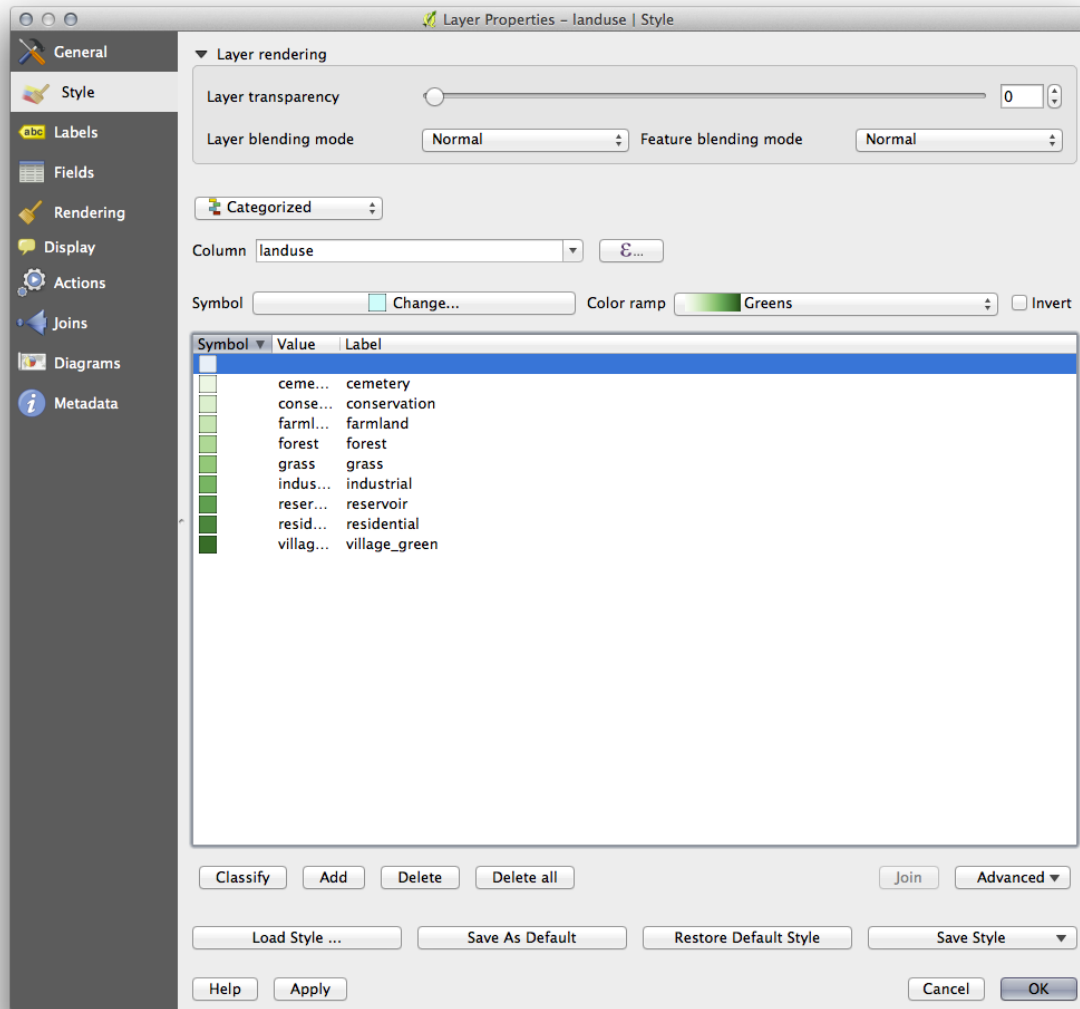
4.3.1 Follow Along: Classifying Nominal Data

- Open the *Layer Properties* dialog for the *landuse* layer.
- Go to the *Style* tab.
- Click on the dropdown that says *Single Symbol* and change it to

Categorized:

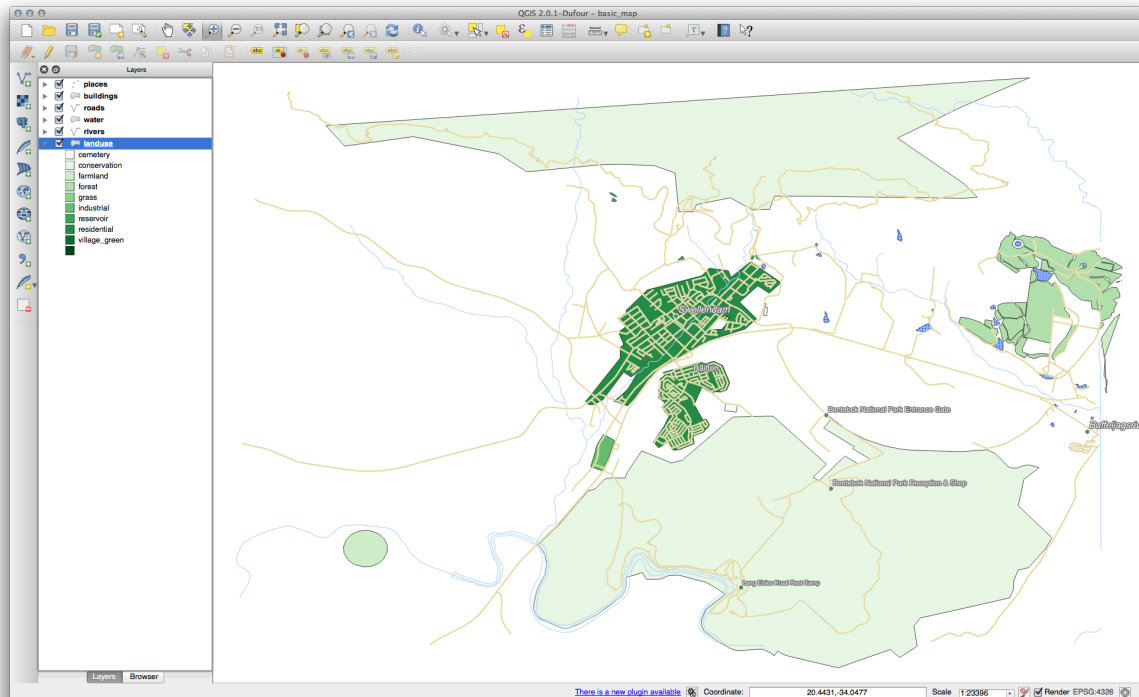


- In the new panel, change the *Column* to *landuse* and the *Color ramp* to *Greens*.
- Click the button labeled *Classify*:

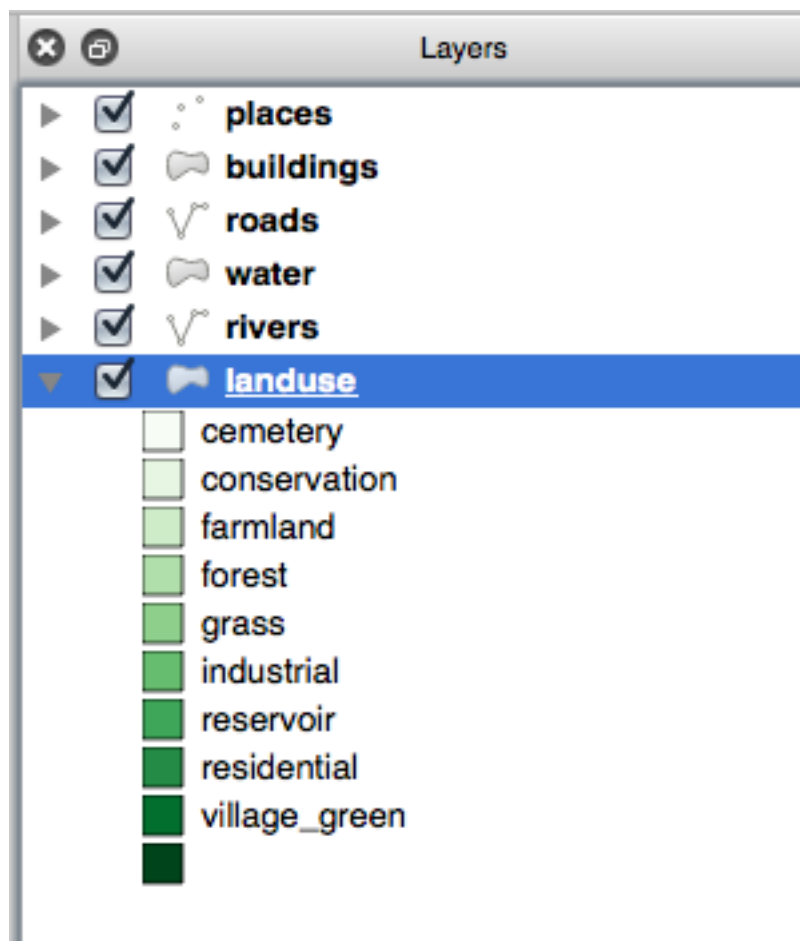


- Click *OK*.

You'll see something like this:



- Click the arrow (or plus sign) next to *landuse* in the *Layer list*, you'll see the categories explained:

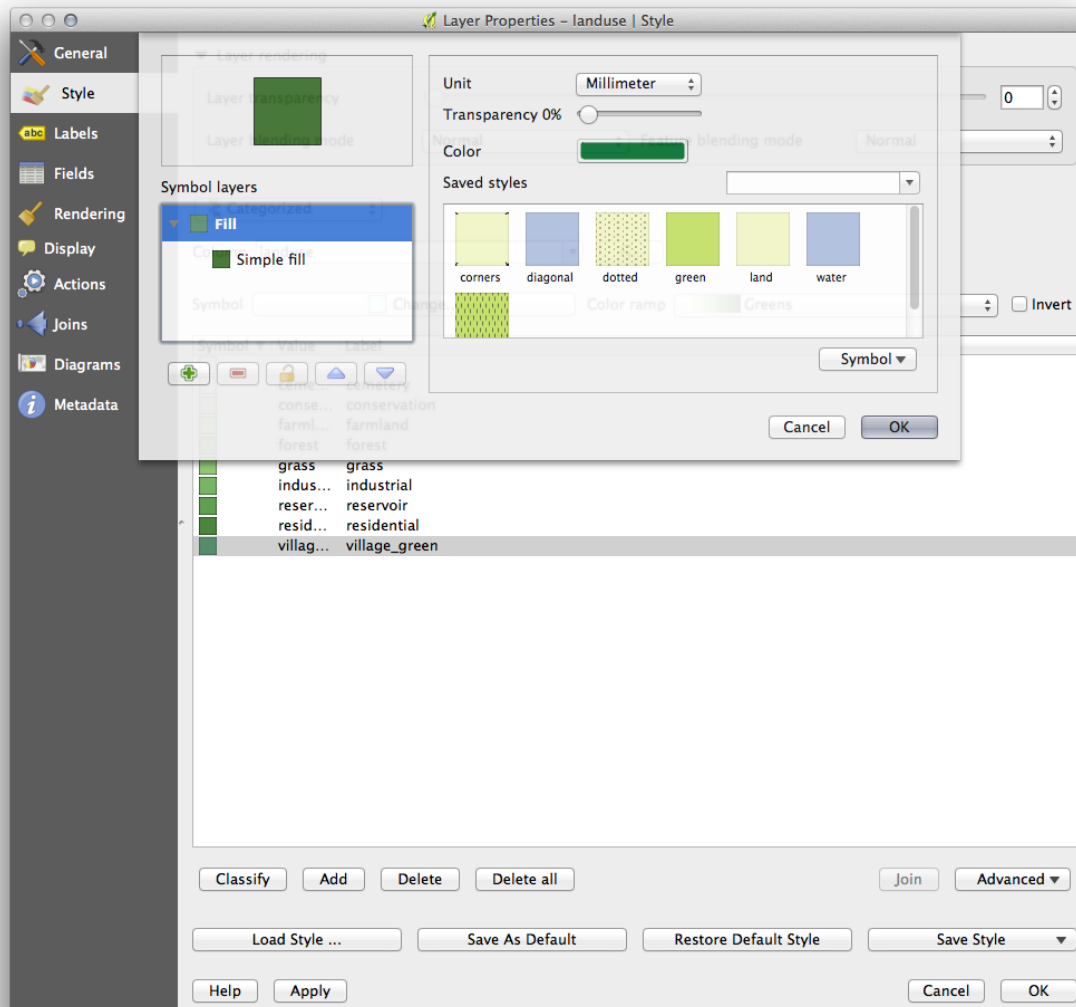


Now our landuse polygons are appropriately colored and are classified so that areas with the same land use are the same color. You may wish to remove the black border from the *landuse* layer:

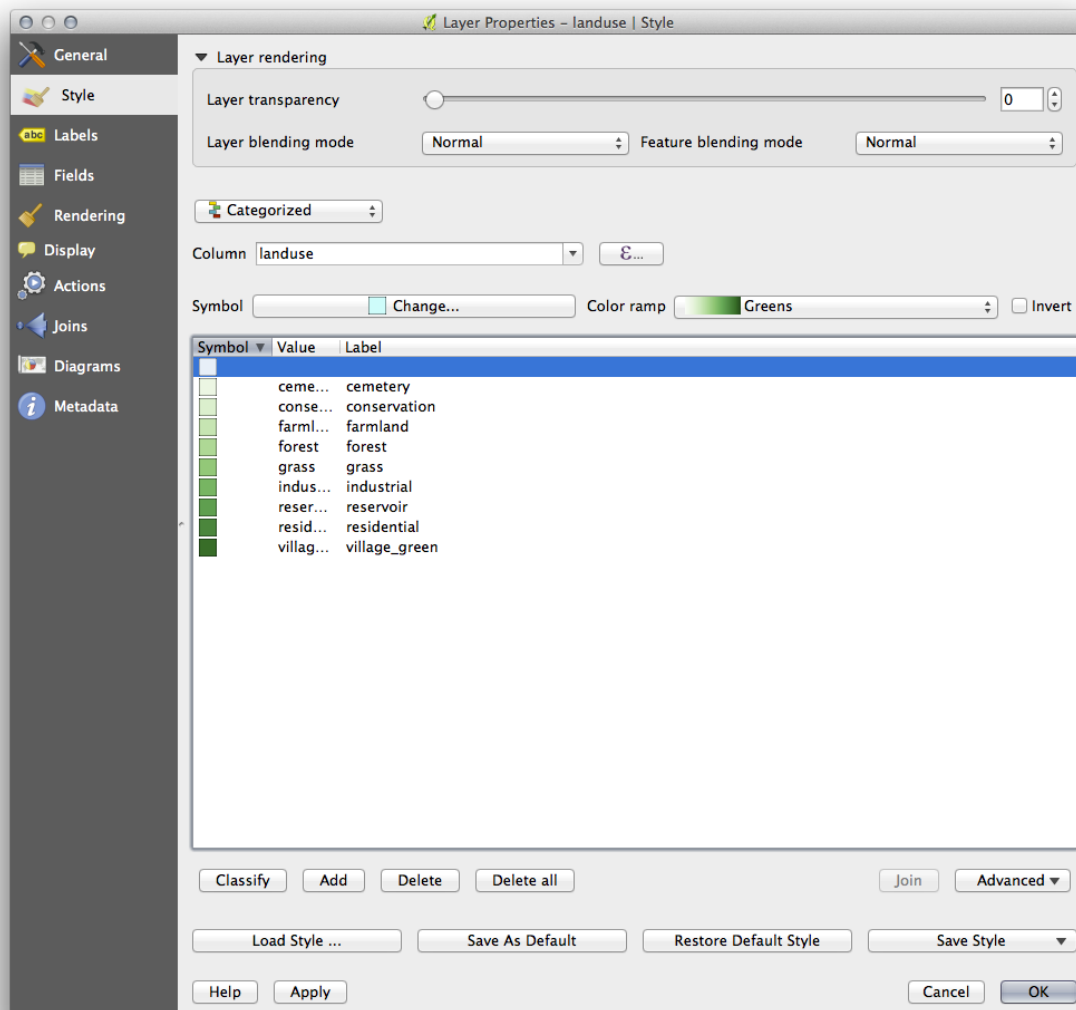
- Open *Layer Properties*, go to the *Style* tab and select *Symbol*.
- Change the symbol by removing the border from the *Simple Fill* layer and click *OK*.

You'll see that the landuse polygon outlines have been removed, leaving just our new fill colours for each categorisation.

- If you wish to, you can change the fill color for each landuse area by double-clicking the relevant color block:



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 is important 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!

4.3.2 Try Yourself More Classification

If you're only following the basic-level content, use the knowledge you gained above to classify the *buildings* layer. Set the categorisation against the *building* column and use the *Spectral* color ramp.

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

4.3.3 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.

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.

In interval classification, the numbers are on a scale with positive, negative and zero values. For example: height above/below sea level, temperature above/below freezing (0 degrees Celsius), etc.

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.

In the example above, we used nominal classification to assign each farm to the town that it is administered by. Now we will use ratio classification to classify the farms by area.

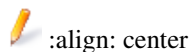
- Save your landuse symbology (if you want to keep it) by clicking on the *Save Style ...* button in the *Style* dialog.

We're going to reclassify the layer, so existing classes will be lost if not saved.

- Close the *Style* dialog.
- Open the Attributes Table for the *landuse* layer.

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

- Enter edit mode by clicking this button:

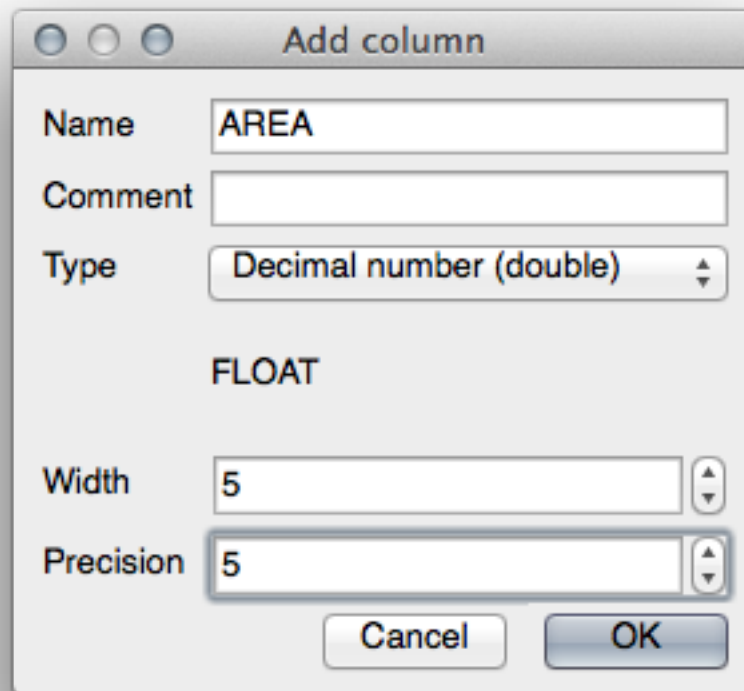


:align: center

- Add a new column with this button:



- Set up the dialog that appears, like this:



- 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.

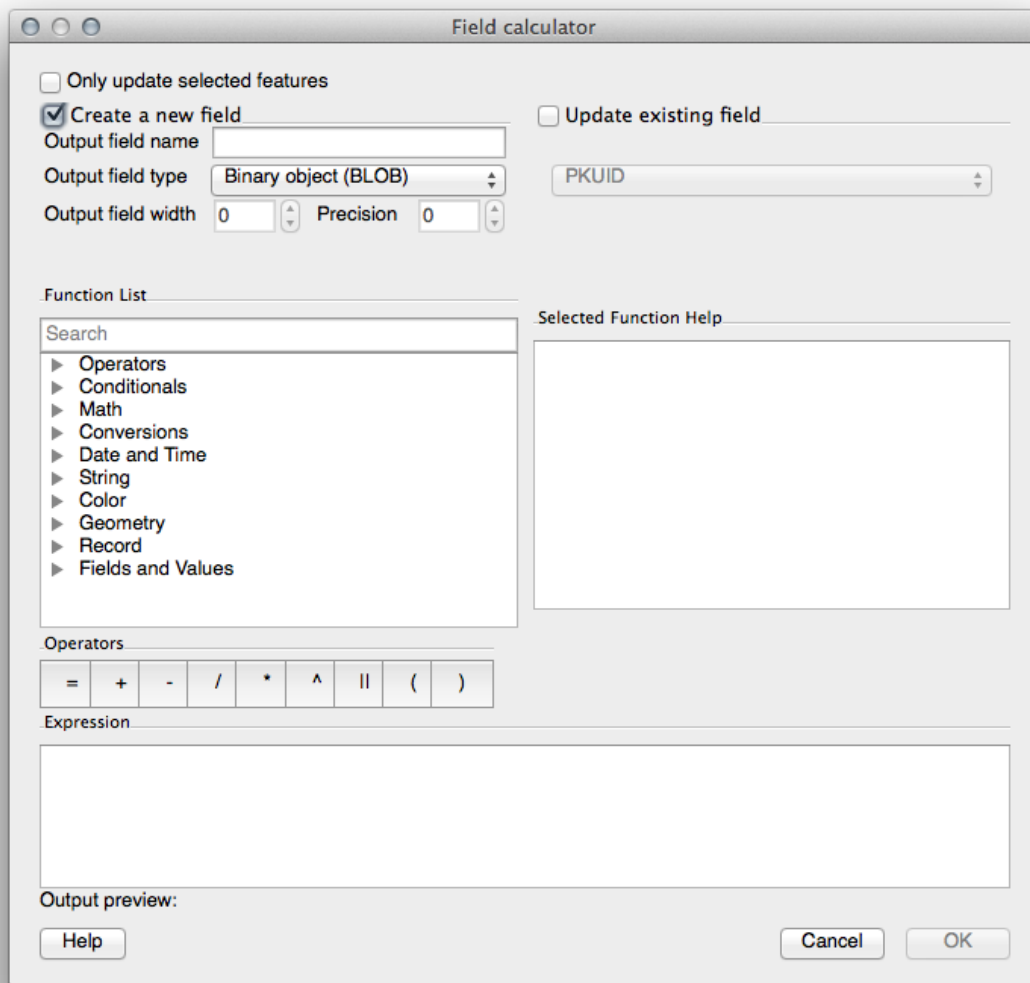
To solve this problem, we'll need to calculate the areas.

- Open the field calculator:

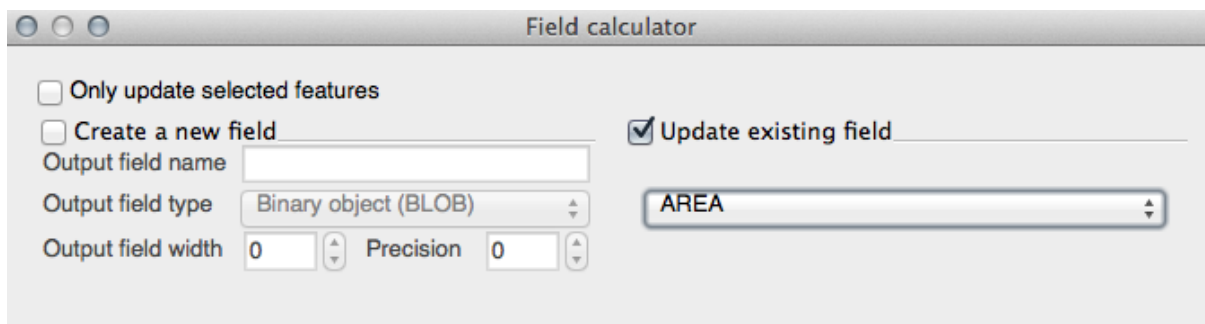


align center

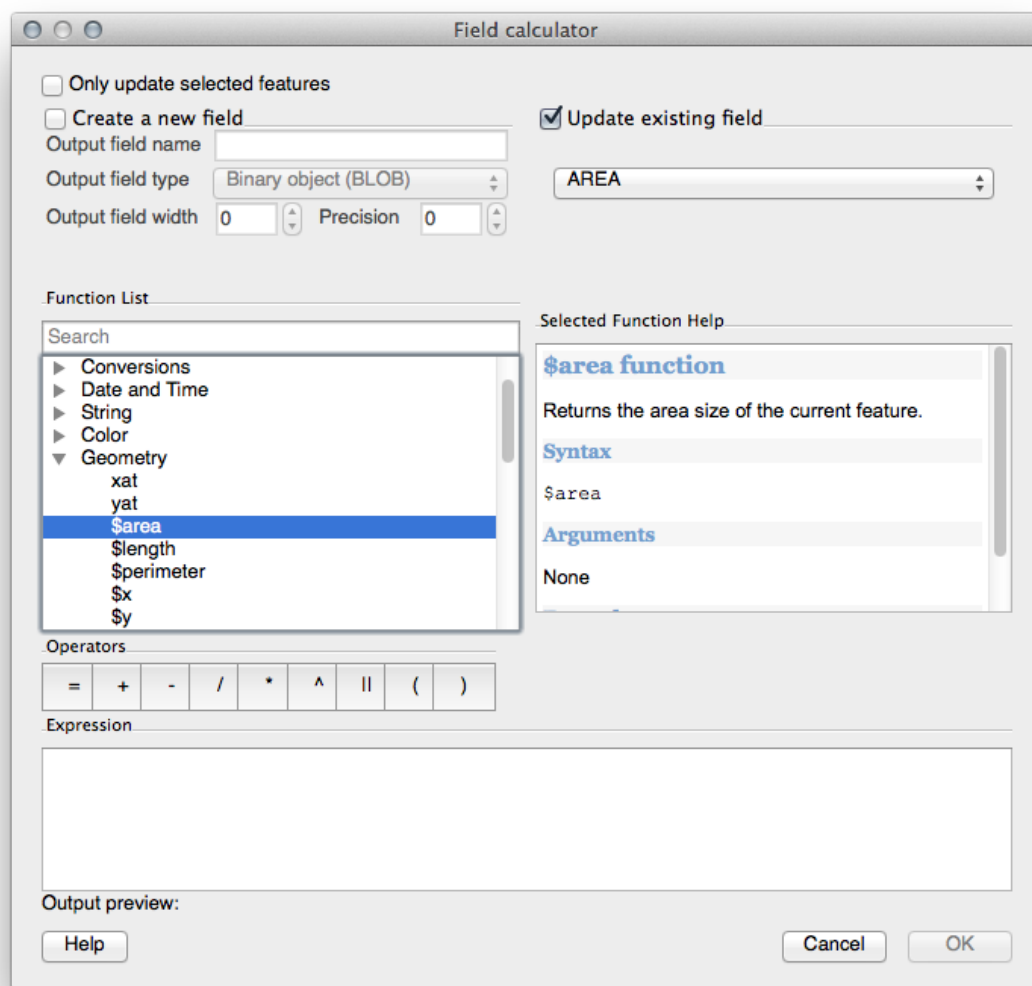
You'll get this dialog:



- Change the values at the top of the dialog to look like this:



- In the *Function List*, select *Geometry* → *\$area*:

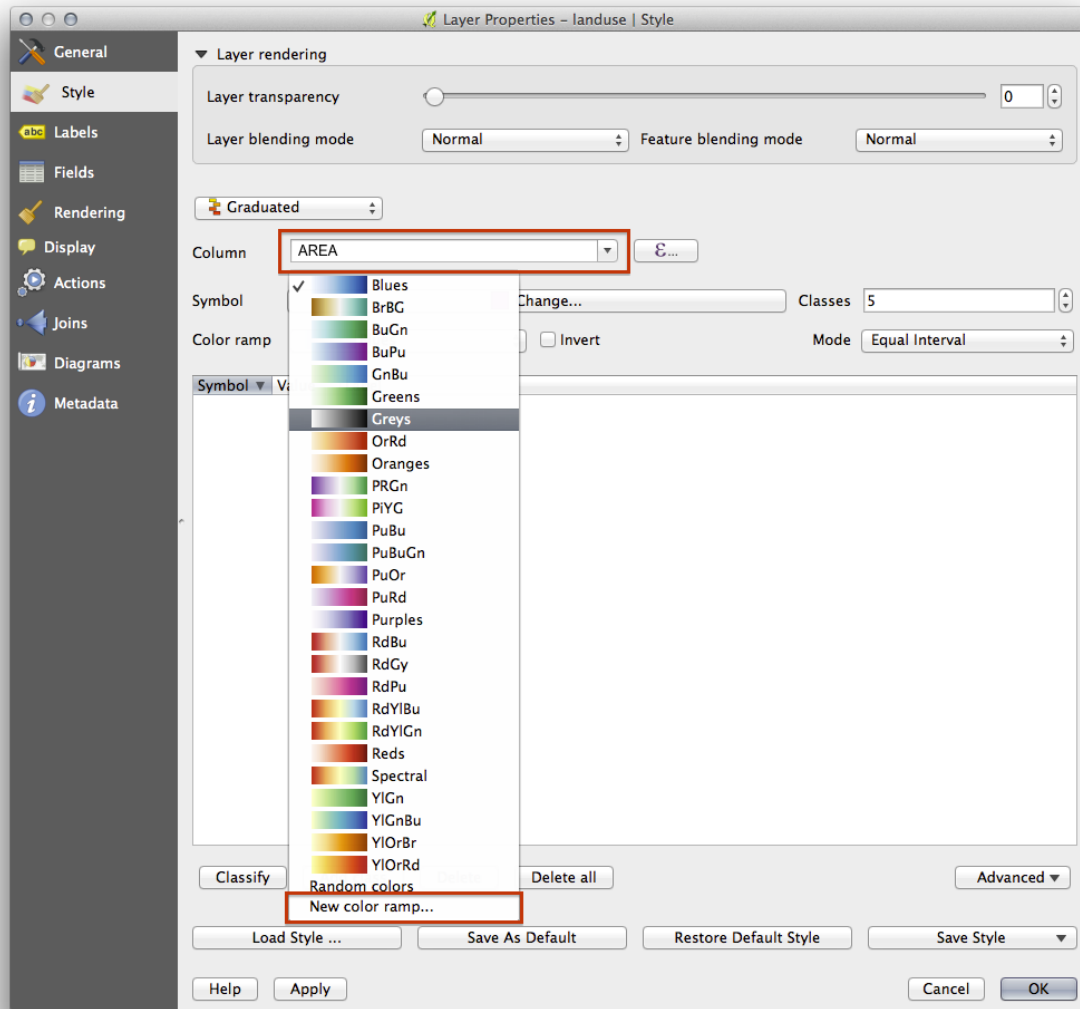


- Double-click on it so that it appears in the *Expression* field.
- Click *OK*.

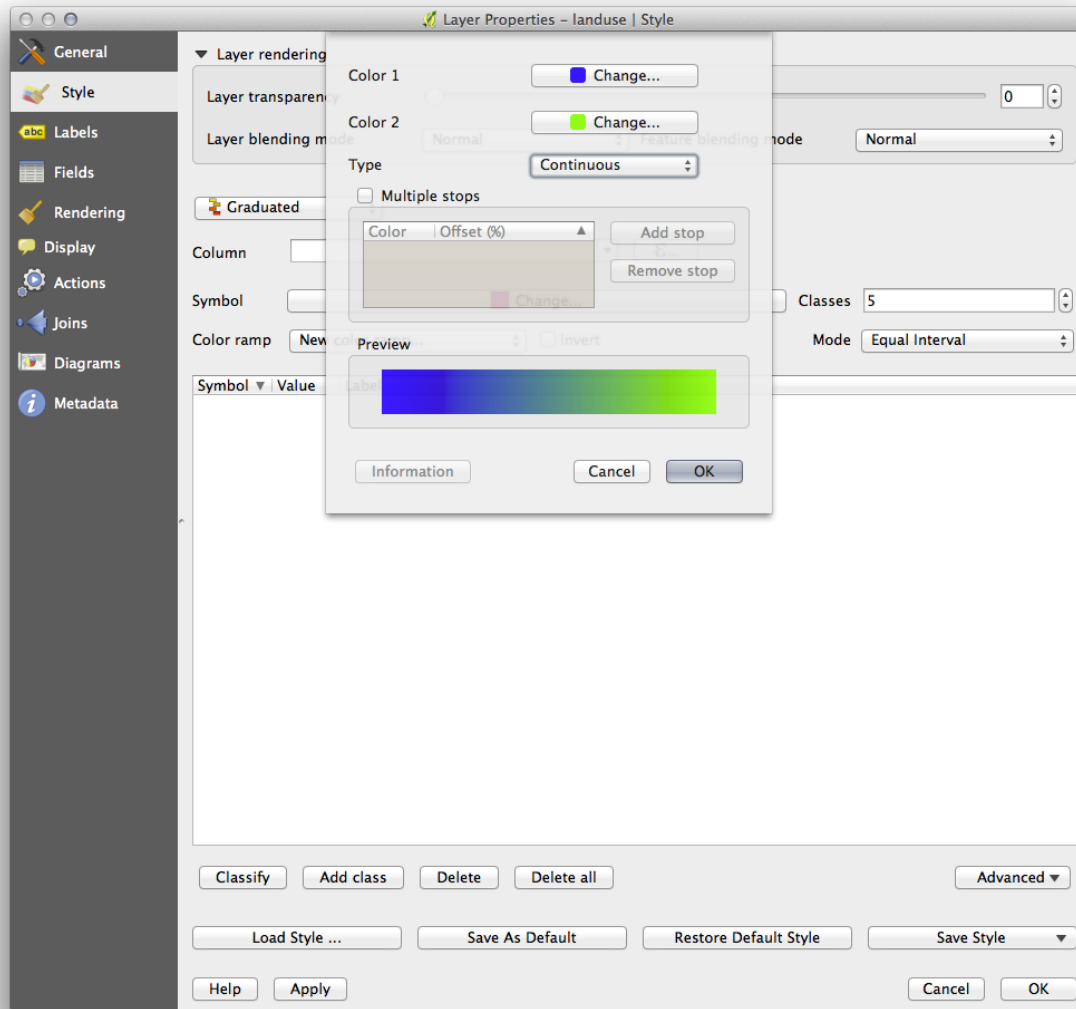
Now your AREA field is populated with values (you may need to click the column header to refresh the data). Save the edits and click *Ok*.

Bemerkung: These areas are in degrees. Later, we will compute them in square meters.

- Open the *Layer properties* dialog's *Style* tab.
- Change the classification style from *Categorized* to *Graduated*.
- Change the *Column* to *AREA*:
- Under *Color ramp*, choose the option *New color ramp...* to get this dialog:



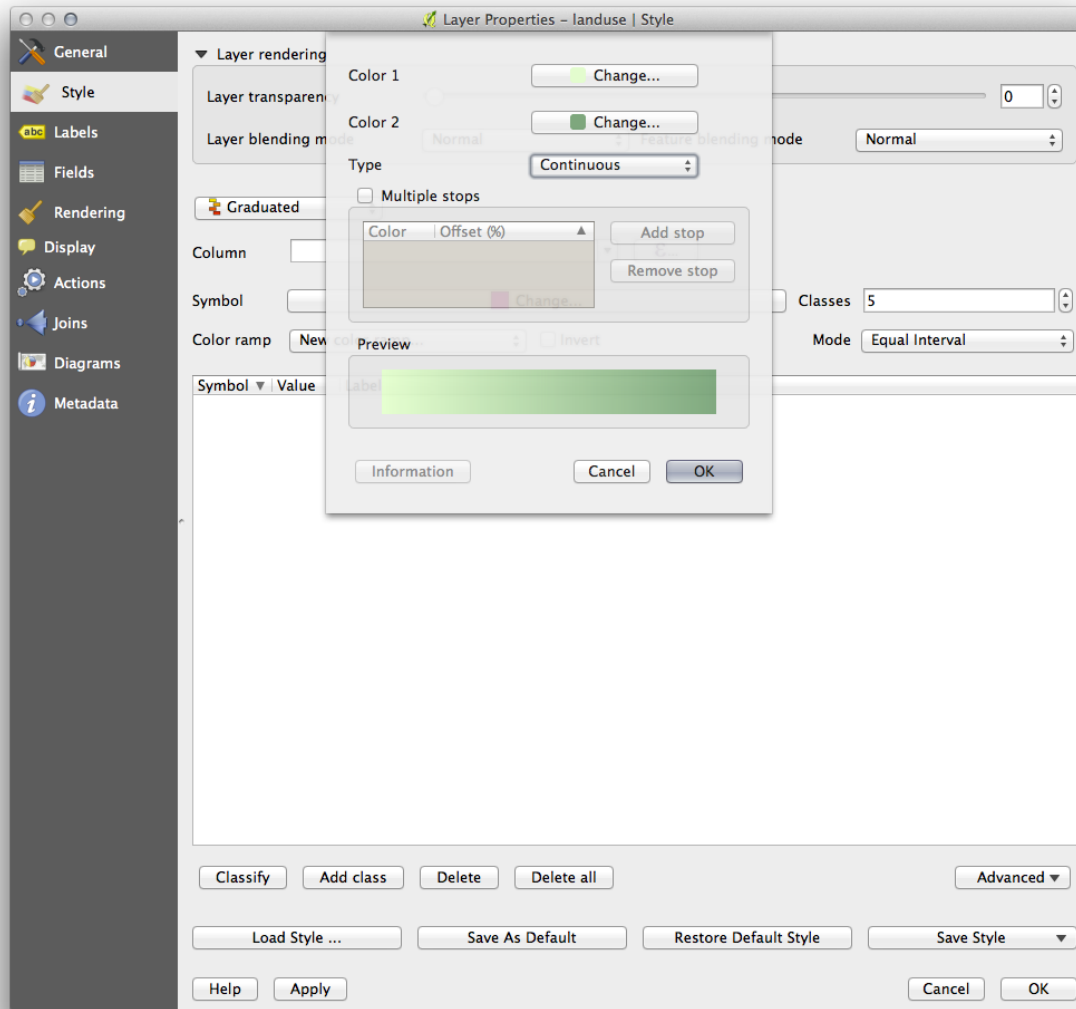
- Choose *Gradient* (if it's not selected already) and click *OK*. You'll see this:



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

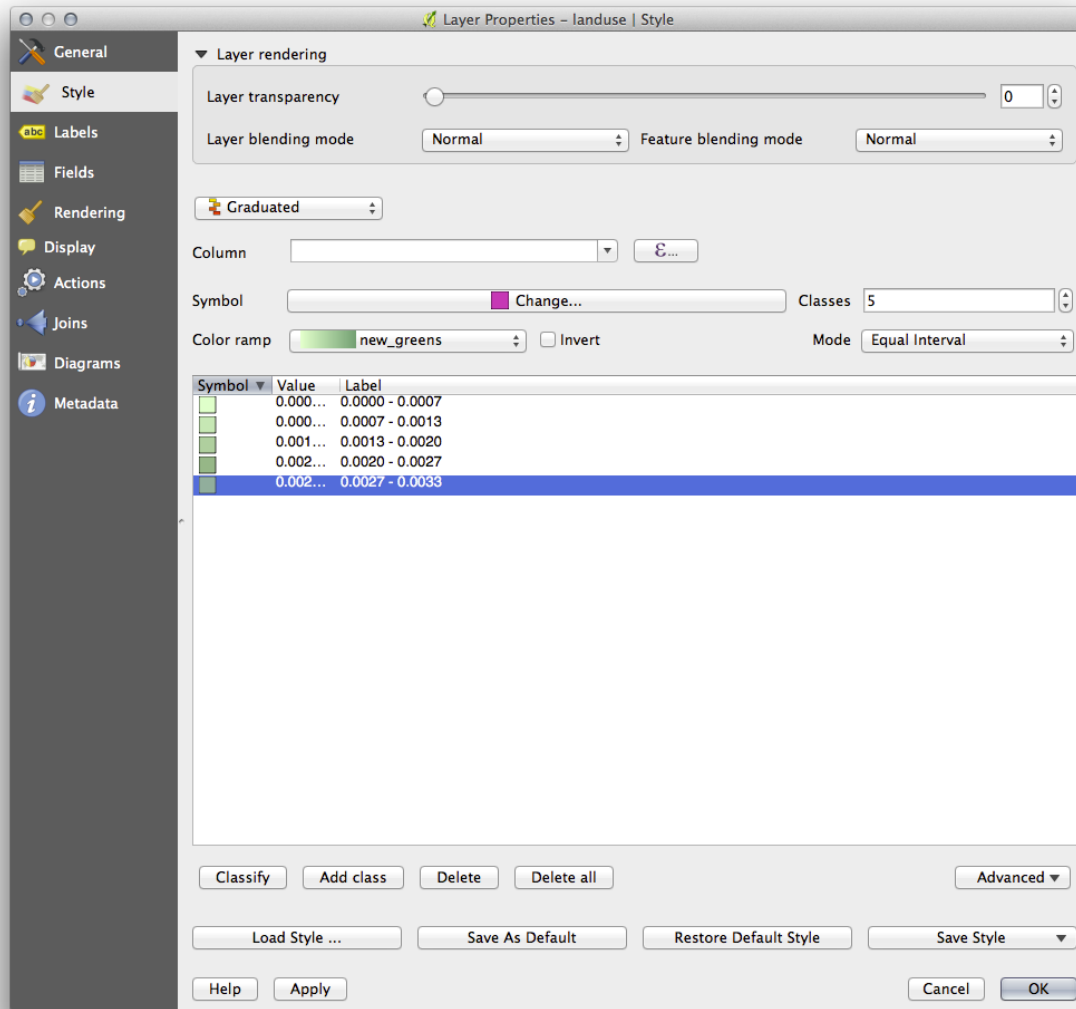
- Choose appropriate colors.

In the example, the result looks like this:



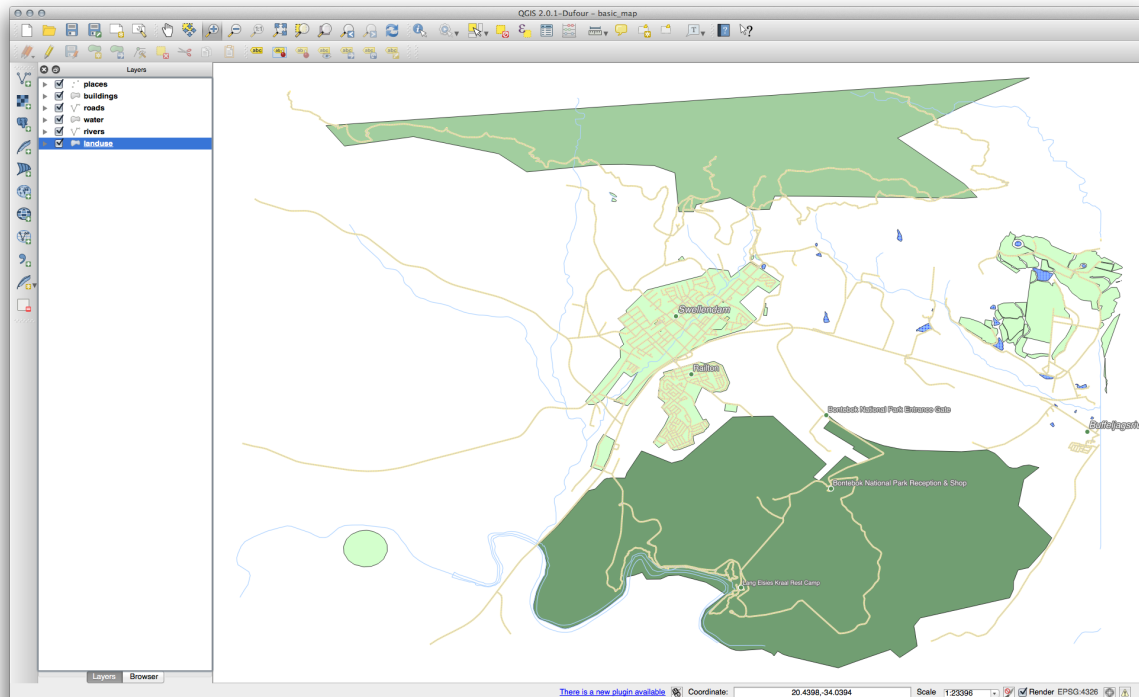
- Click *OK*.
- Choose a suitable name for the new color ramp.
- Click *OK* after filling in the name.

Now you'll have something like this:



Leave everything else as-is.

- Click *Ok*:



4.3.4 Try Yourself Refine the Classification

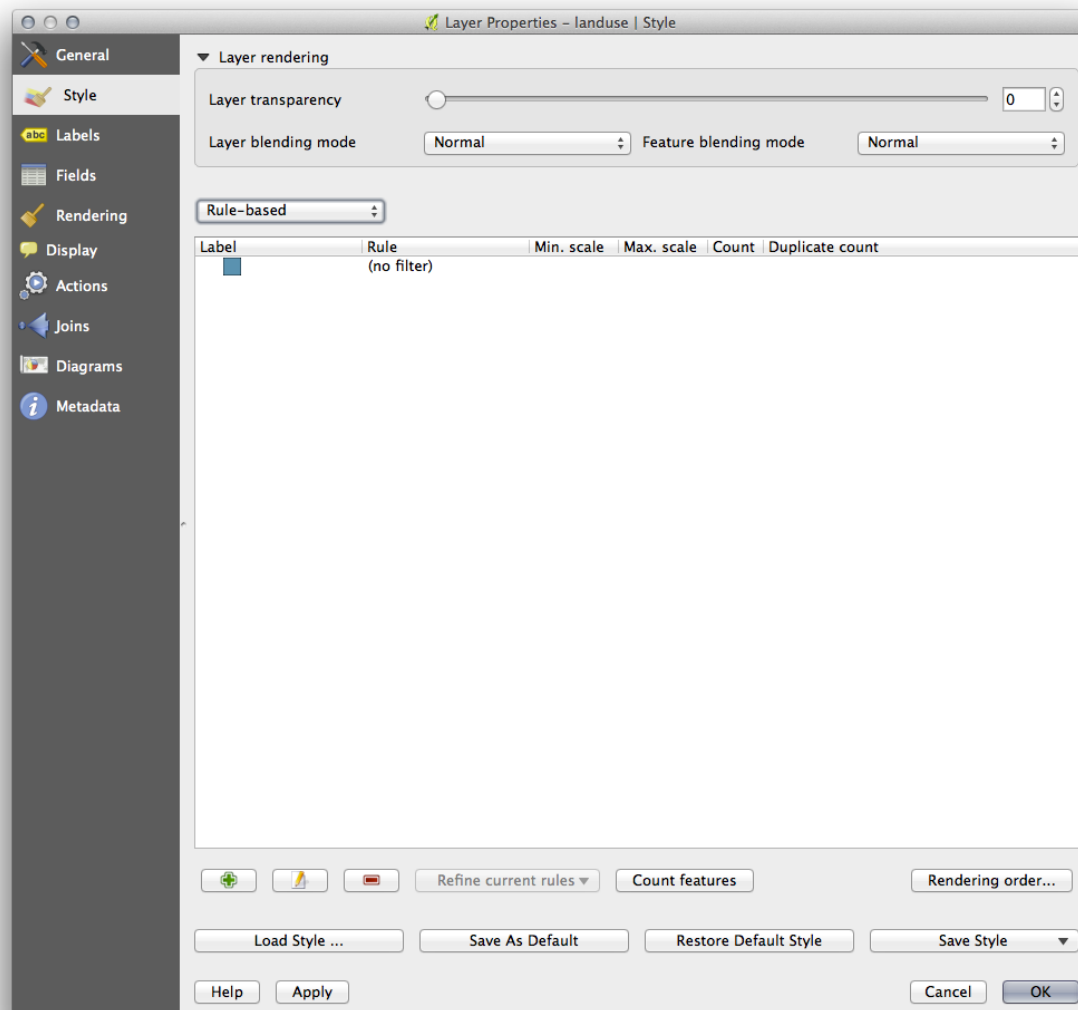
- Get rid of the lines between the classes.
- Change the values of *Mode* and *Classes* until you get a classification that makes sense.


Check your results

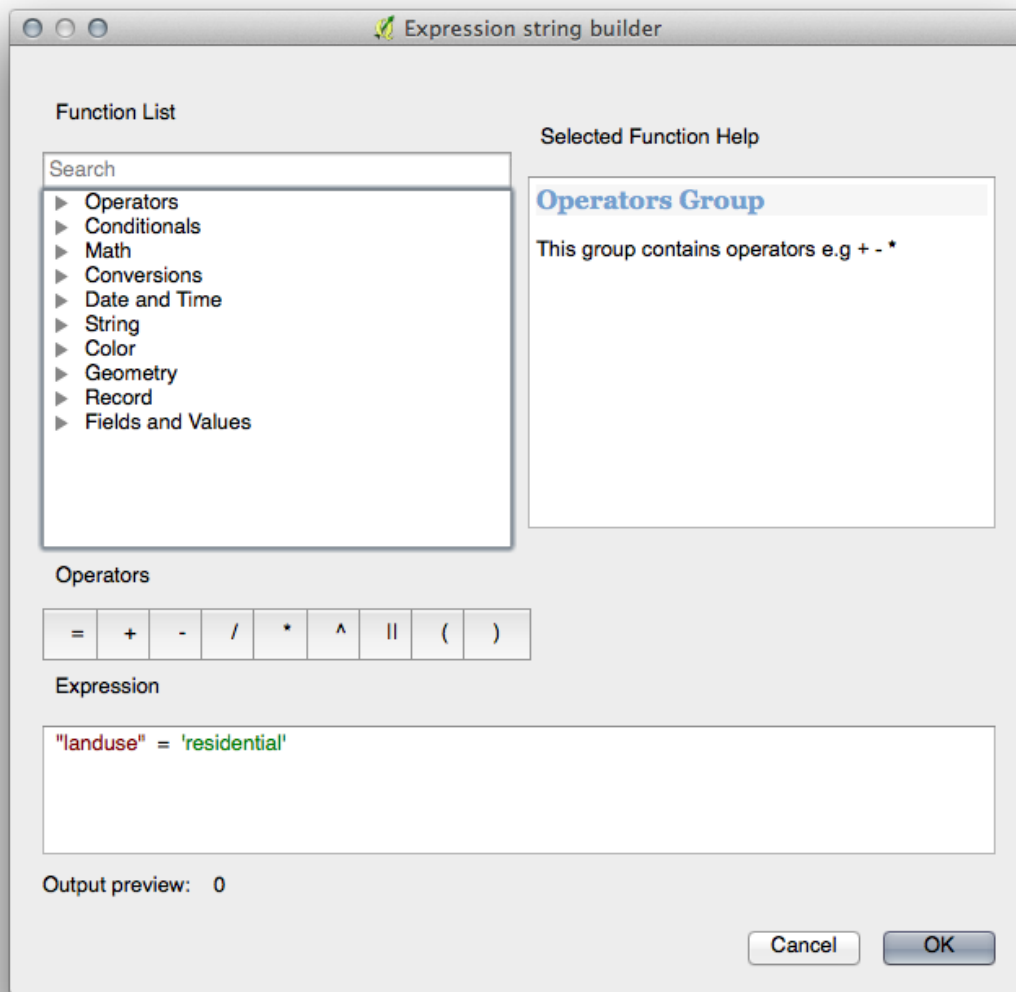
4.3.5 Follow Along: Rule-based Classification

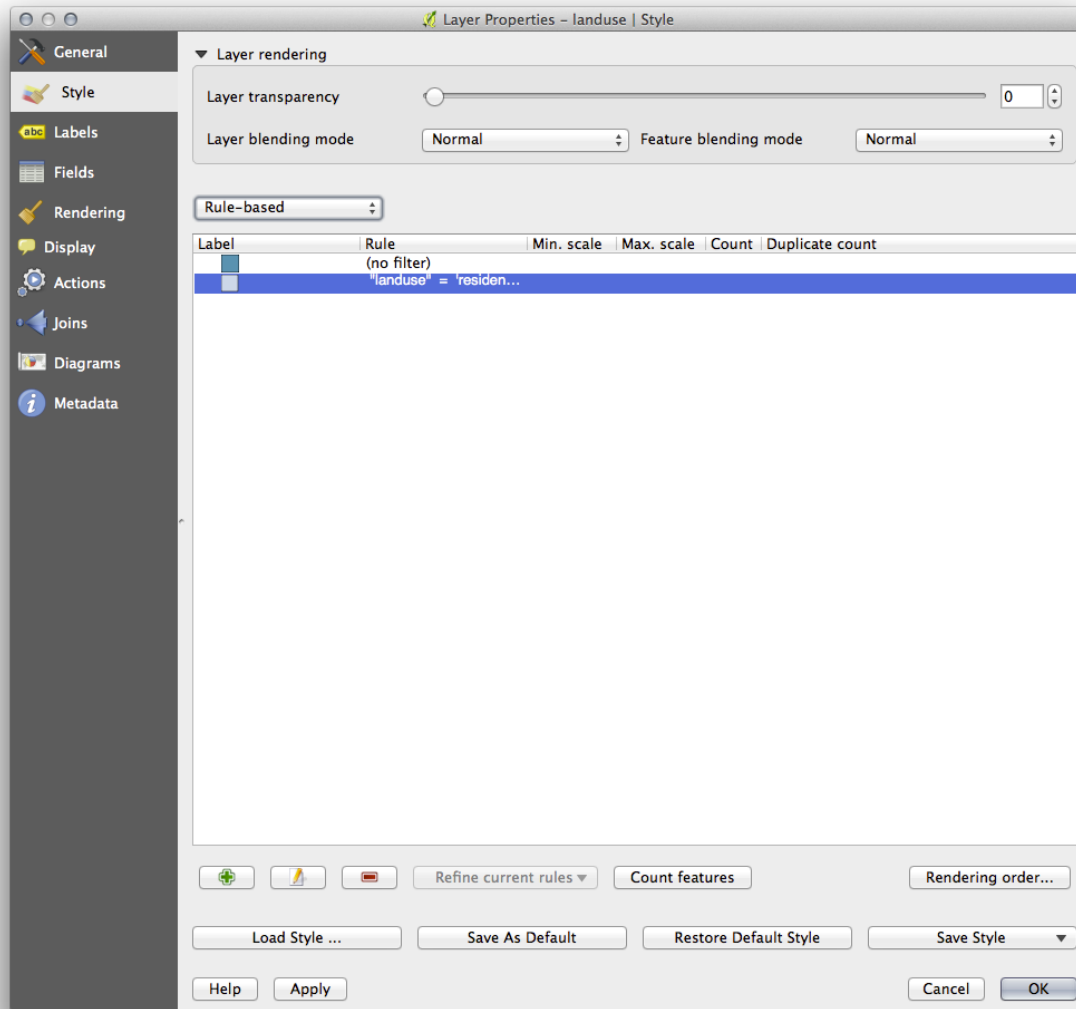
It's often useful to combine multiple criteria for a classification, but unfortunately normal classification only takes one attribute into account. That's where rule-based classification comes in handy.

- Open the *Layer Properties* dialog for the *landuse* layer.
- Switch to the *Style* tab.
- Switch the classification style to *Rule-based*. You'll get this:



- Click the *Add rule* button: .
- A new dialog then appears.
- Click the ellipsis ... button next to the *Filter* text area.
- Using the query builder that appears, enter the criterion "landuse" = 'residential' AND "name" != ' |majorUrbanName| ', click *Ok* and choose a pale blue-grey for it and remove the border:



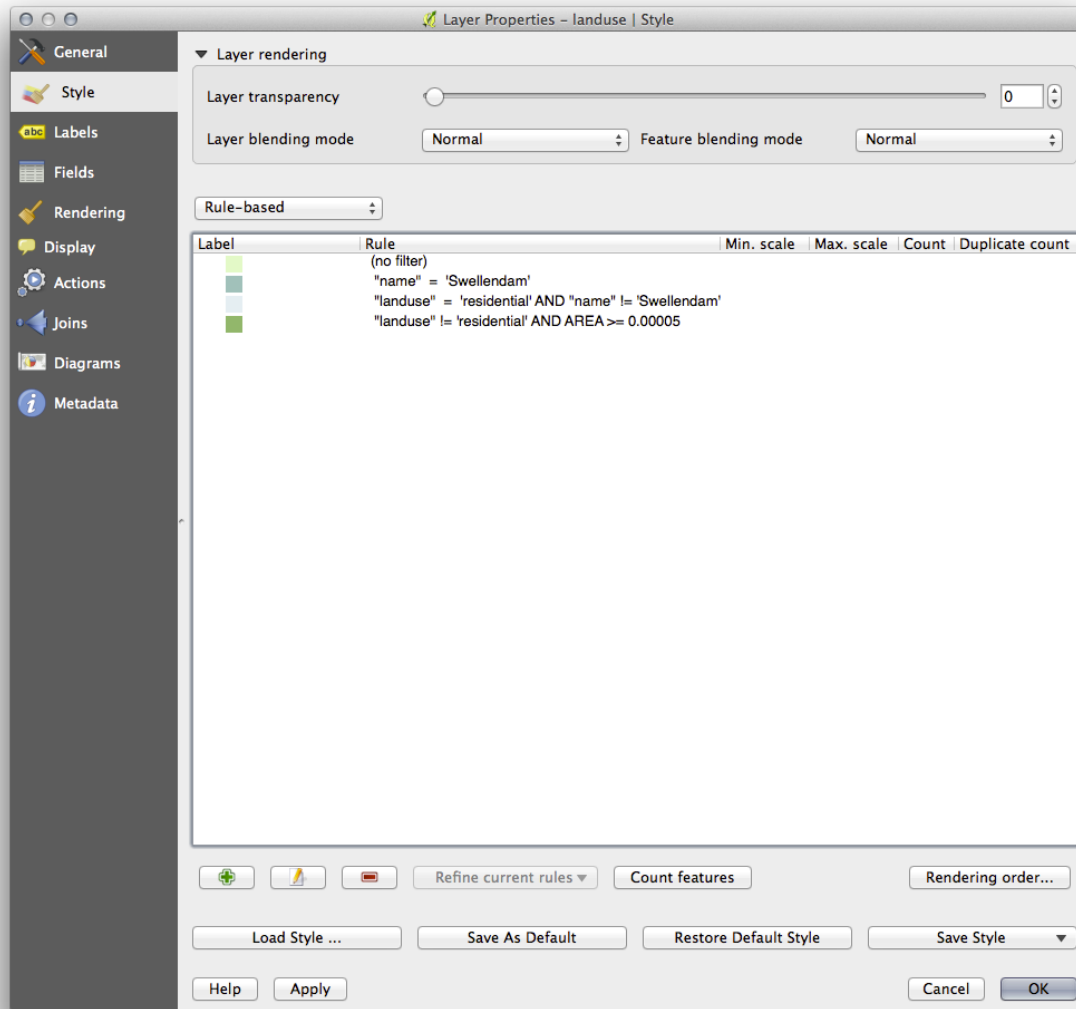


- Add a new criterion `"landuse" != 'residential' AND AREA >= 0.00005` and choose a mid-green color.
- Add another new criterion `"name" = ' |majorUrbanName| '` and assign it a darker grey-blue color in order to indicate the town's importance in the region.
- Click and drag this criterion to the top of the list.

These filters are exclusive, in that they collectively exclude some areas on the map (i.e. those which are smaller than 0.00005, are not residential and are not 'Swellendam'). This means that the excluded polygons take the style of the default (*no filter*) category.

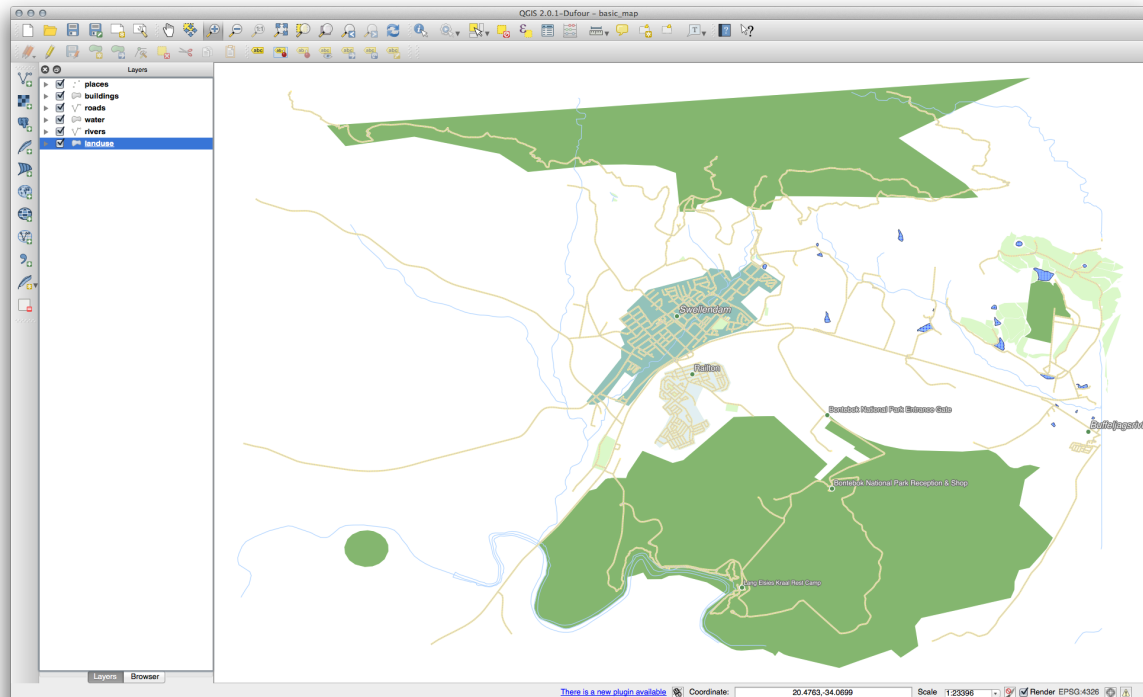
We know that the excluded polygons on our map cannot be residential areas, so give the default category a suitable pale green color.

Your dialog should now look like this:



- 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.

4.3.6 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.

4.3.7 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!

Module: Creating Maps

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

5.1 Lesson: Using Map Composer

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 Map Composer.

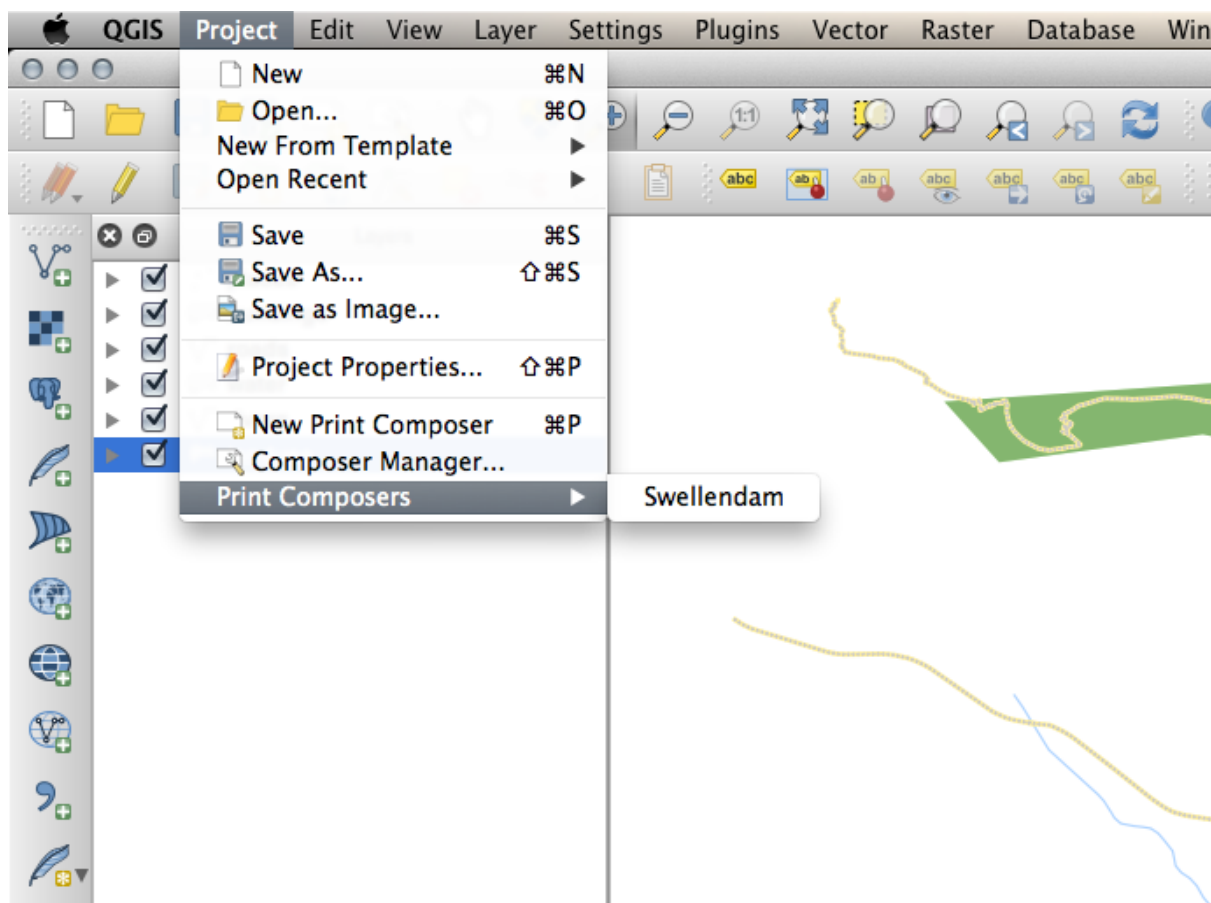
The goal for this lesson: To use the QGIS Map Composer to create a basic map with all the required settings.

5.1.1 Follow Along: The Composer Manager

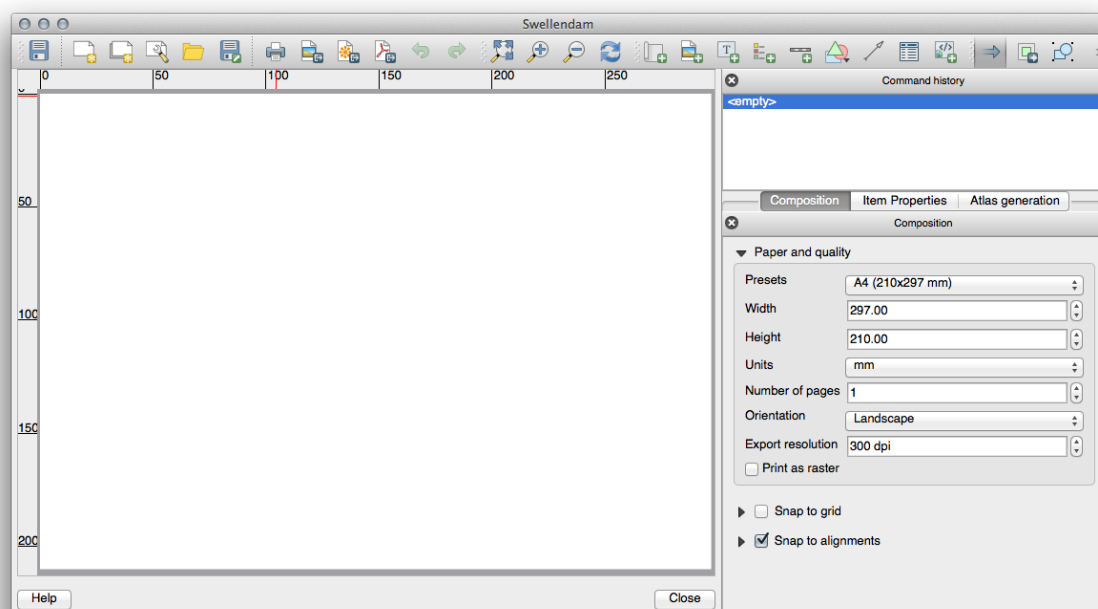
QGIS allows you to create multiple maps using the same map file. For this reason, it has a tool called the *Composer Manager*.

- Click on the *Project → Composer Manager* menu entry to open this tool. You'll see a blank *Composer manager* dialog appear.
- Click the *Add* button and give the new composer the name of Swellendam.
- Click *OK*.
- Click the *Show* button.

(You could also close the dialog and navigate to a composer via the *File → Print Composers* menus, as in the image below.)



Whichever route you take to get there, you will now see the *Print Composer* window:



5.1.2 Follow Along: Basic Map Composition

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

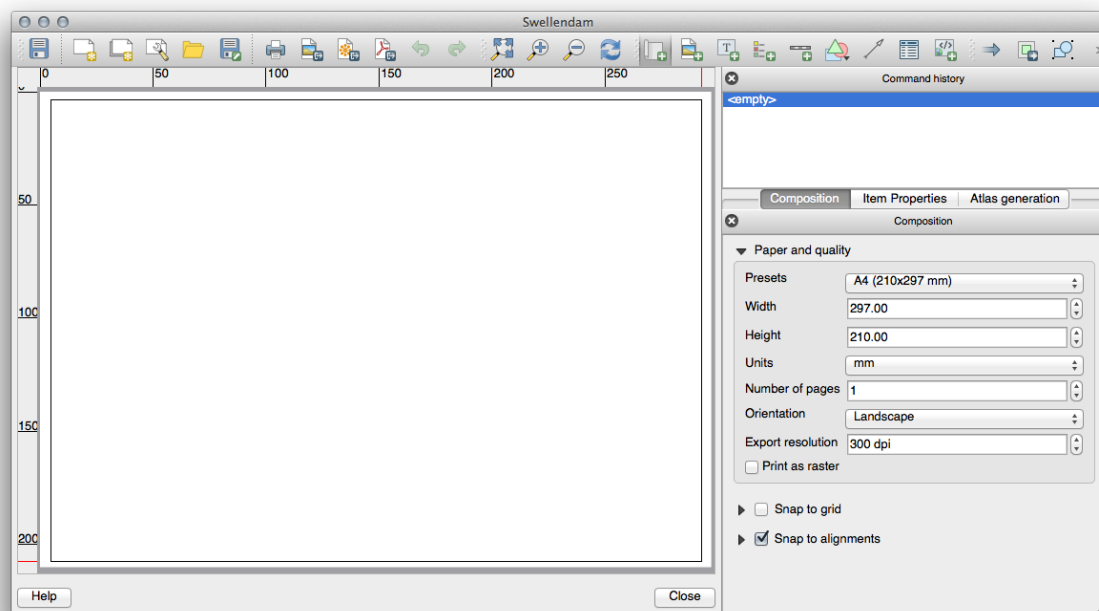
- In the *Print Composer* window, check that the values under *Composition* → *Paper and Quality* are set to the following:
- *Size*: A4 (210x297mm)
- *Orientation*: Landscape
- *Quality*: 300dpi

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!

- Click on the *Add New Map* button: 

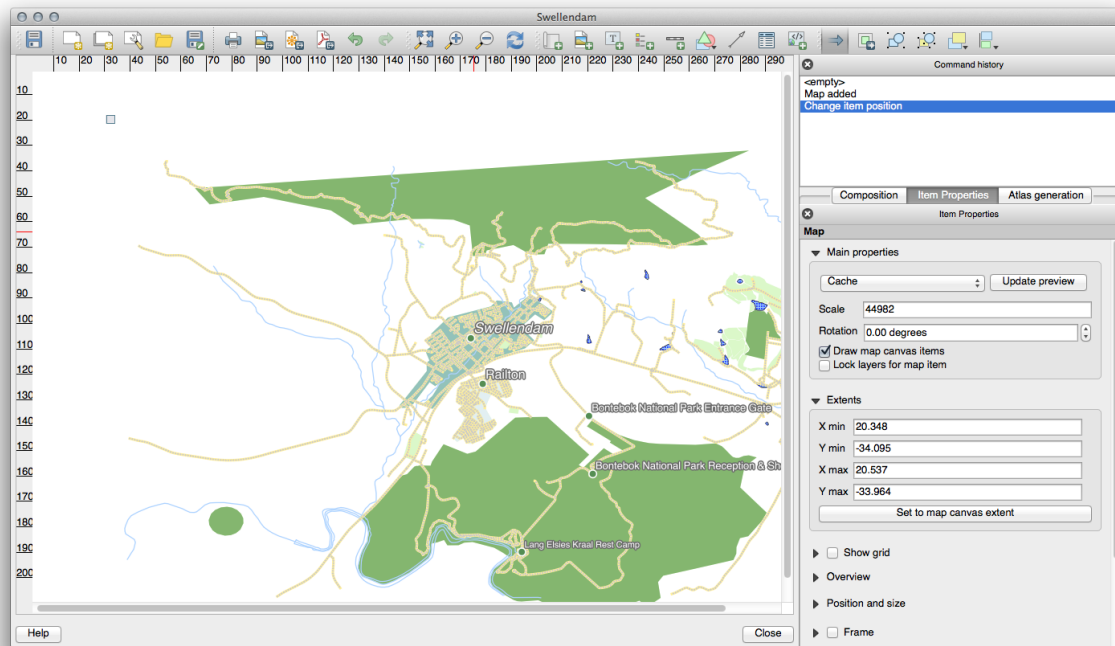
With this tool activated, you'll be able to place a map on the page.

- Click and drag a box on the blank page:

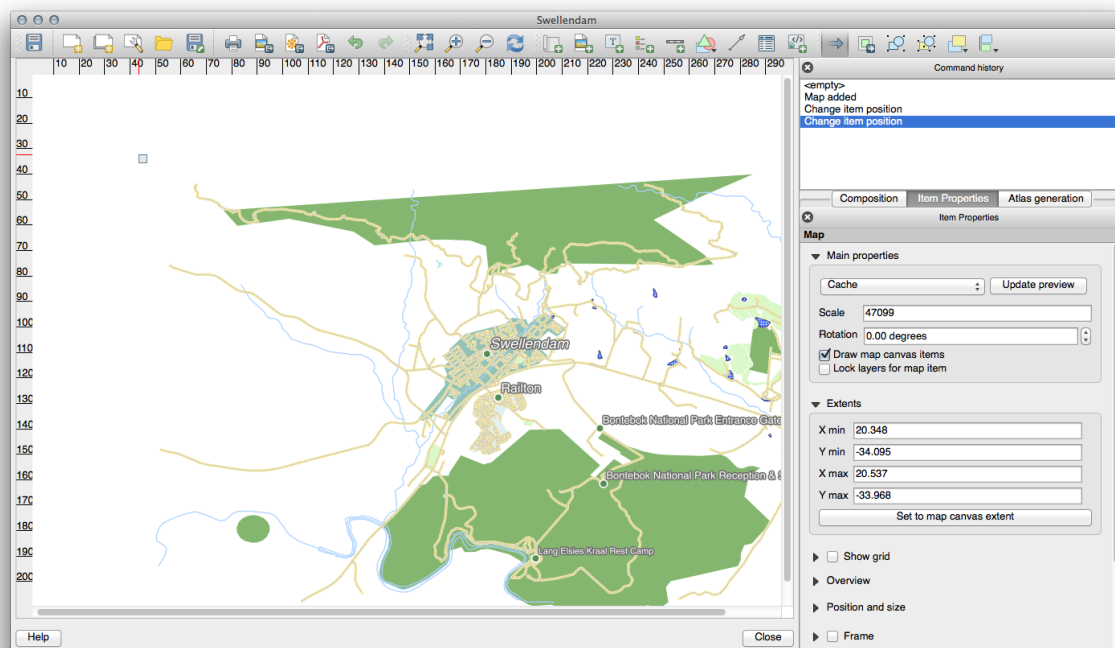


The map will appear on the page.

- Move the map by clicking and dragging it around:




- Resize it by clicking and dragging the boxes in the corners:



Bemerkung: 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.

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



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

When zooming in, the map view will not refresh by itself. This is so that it doesn't waste your time redrawing the map while you're zooming the page to where you want it, but it also means that if you zoom in or out, the map will be at the wrong resolution and will look ugly or unreadable.


- Force the map to refresh by clicking this 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 *Composer* in QGIS is part of the main map file, you'll need to save your main project. Go to the main QGIS window (the one with the *Layers list* and all the other familiar elements you were working with before), and save your project from there as usual.

5.1.3 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's add a title.

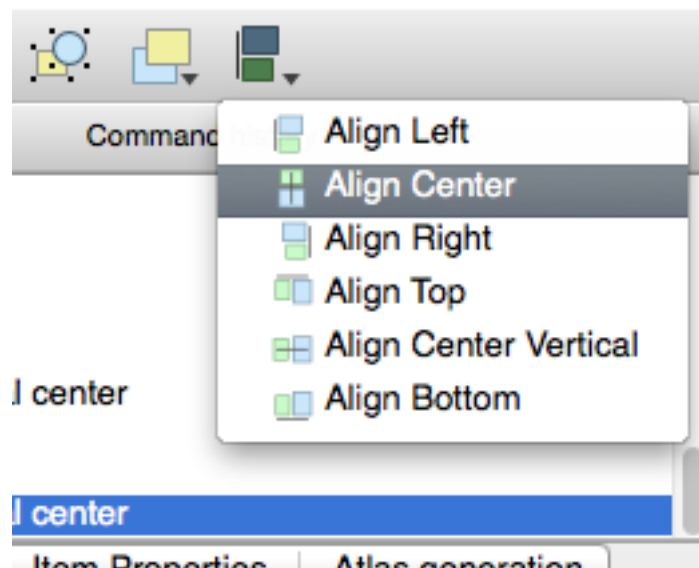
- Click on this button: 
- Click on the page, above the map, and a label will appear at the top of the map.
- Resize it and place it in the top center of the page. It can be resized and moved in the same way that you resized and moved the map.

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 to help position the title relative to the map (not the page):



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



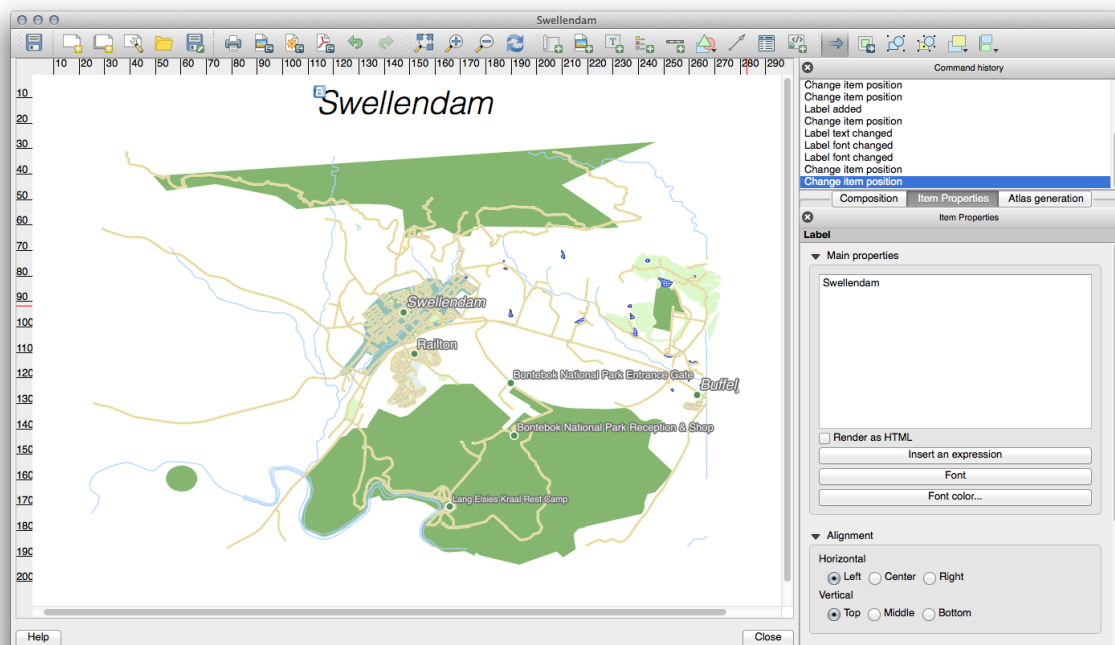
To make sure that you don't accidentally move these elements around now that you've aligned them:

- Right-click on both the map and the label.

A small lock icon will appear in the corner to tell you that an element can't be dragged right now. You can always right-click on an element again to unlock it, though.

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

- Select the label by clicking on it.
- Click on the *Item Properties* tab in the side panel of the *Composer* window.
- Change the text of the label to "Swellendam":
- Use this interface to set the font and alignment options:



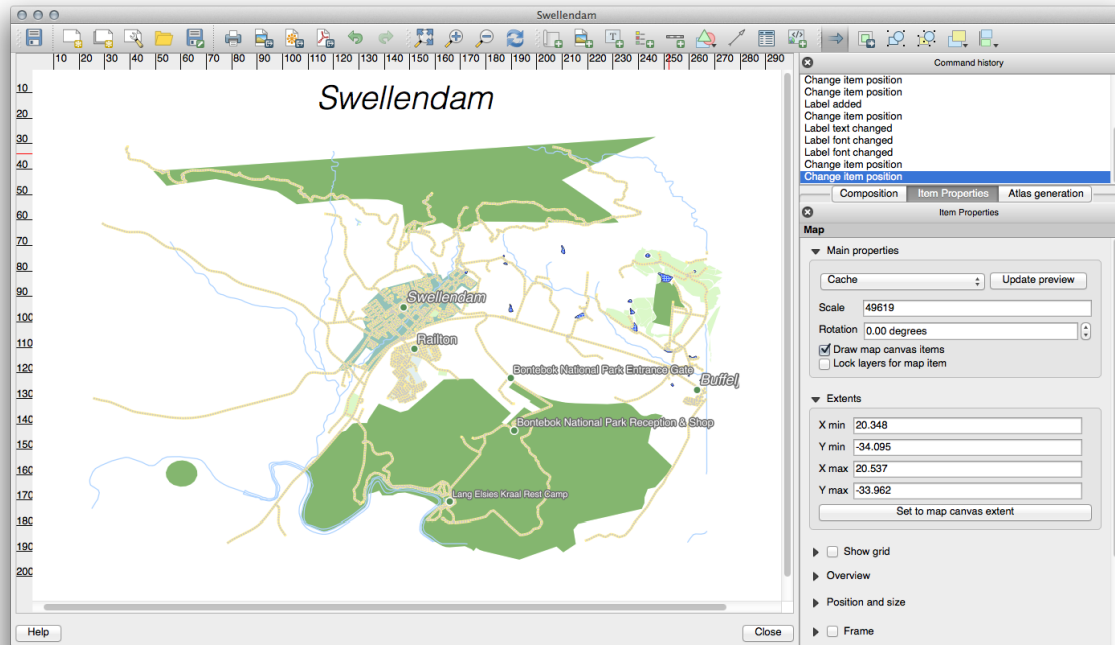
- Choose a large but sensible font (the example will use the default font with a size of 36) and set the *Horizontal Alignment* to *Center*.

You can also change the font color, but it's probably best to keep it black as per the default.

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:


- In the *Item Properties* tab, scroll down until you see the *Frame* option.
- Click the *Frame* checkbox to enable the frame. You can also change the frame's color and width.

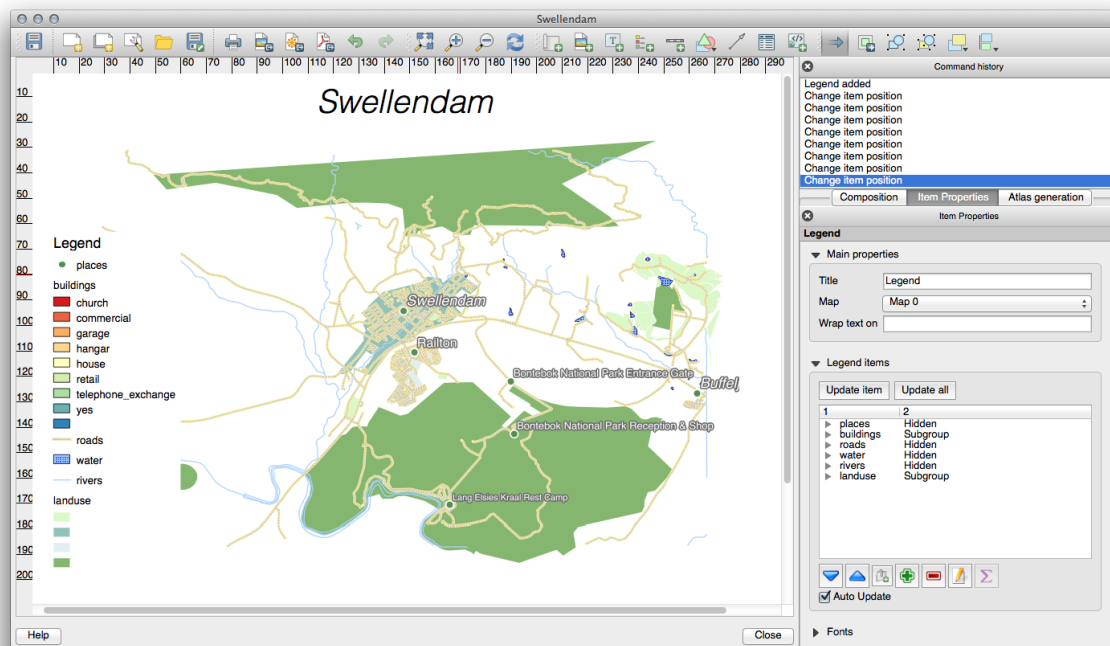
In this example, we won't enable the frame, so here is our page so far:



5.1.4 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 farms. Let's add a new legend.

- Click on this button: 
- Click on the page to place the legend, and move it to where you want it:




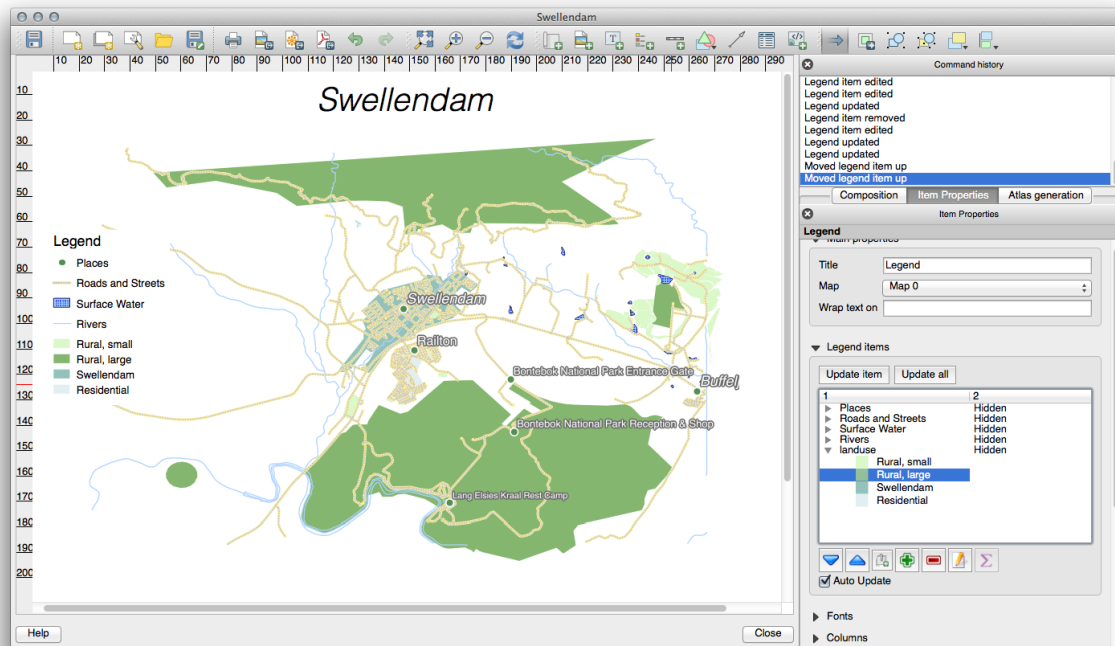
5.1.5 Follow Along: Customizing Legend Items

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

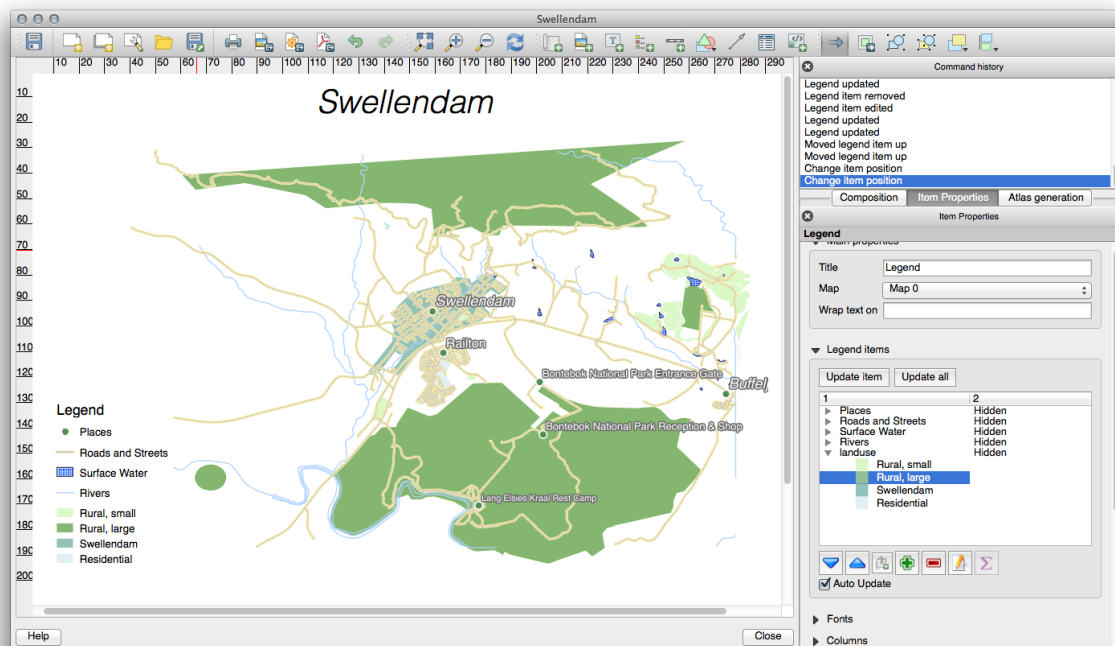
- In the *Item Properties* tab, you'll find the *Legend items* panel.
- Select the *buildings* entry.
- Delete it from the legend by clicking the *minus* button: 

You can also rename items.

- Select a layer from the same list.
- Click the *Edit* button: 
- Rename the layers to Places, Roads and Streets, Surface Water, and Rivers.
- Set landuse to *Hidden*, then click the down arrow and edit each category to name them on the legend. 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:



5.1.6 Follow Along: Exporting Your Map

Bemerkung: 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 *Composer* window:



The button on the left is the *Print* button, which 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 three buttons allow you to export the map page to a file. There are three export formats to choose from:

- *Export as Image*
- *Export as SVG*
- *Export as PDF*


Exporting as an image will give 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.

The other two options are more common.

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.

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.

- Click the *Export as PDF* button: 
- Choose a save location and a file name as usual.
- Click *Save*.

5.1.7 In Conclusion

- Close the *Composer* window.
- Save your map.
- Find your exported PDF using your operating system's file manager.
- Open it.
- Bask in its glory.

Congratulations on your first completed QGIS map project!

5.1.8 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.

5.2 Assignment 1

Open your existing map project and revise it thoroughly. If you have noticed small errors or things you'd have liked to fix earlier, do so now.

While customizing your map, keep asking yourself questions. Is this map easy to read and understand for someone who's unfamiliar with the data? If I saw this map on the Internet, or on a poster, or in a magazine, would it capture my attention? Would I want to read this map if it wasn't mine?



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!

5.2.1 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.

Module: Creating Vector Data

Creating maps using existing data is just the beginning. In this module, you'll learn how to modify existing vector data and create new datasets entirely.

6.1 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.

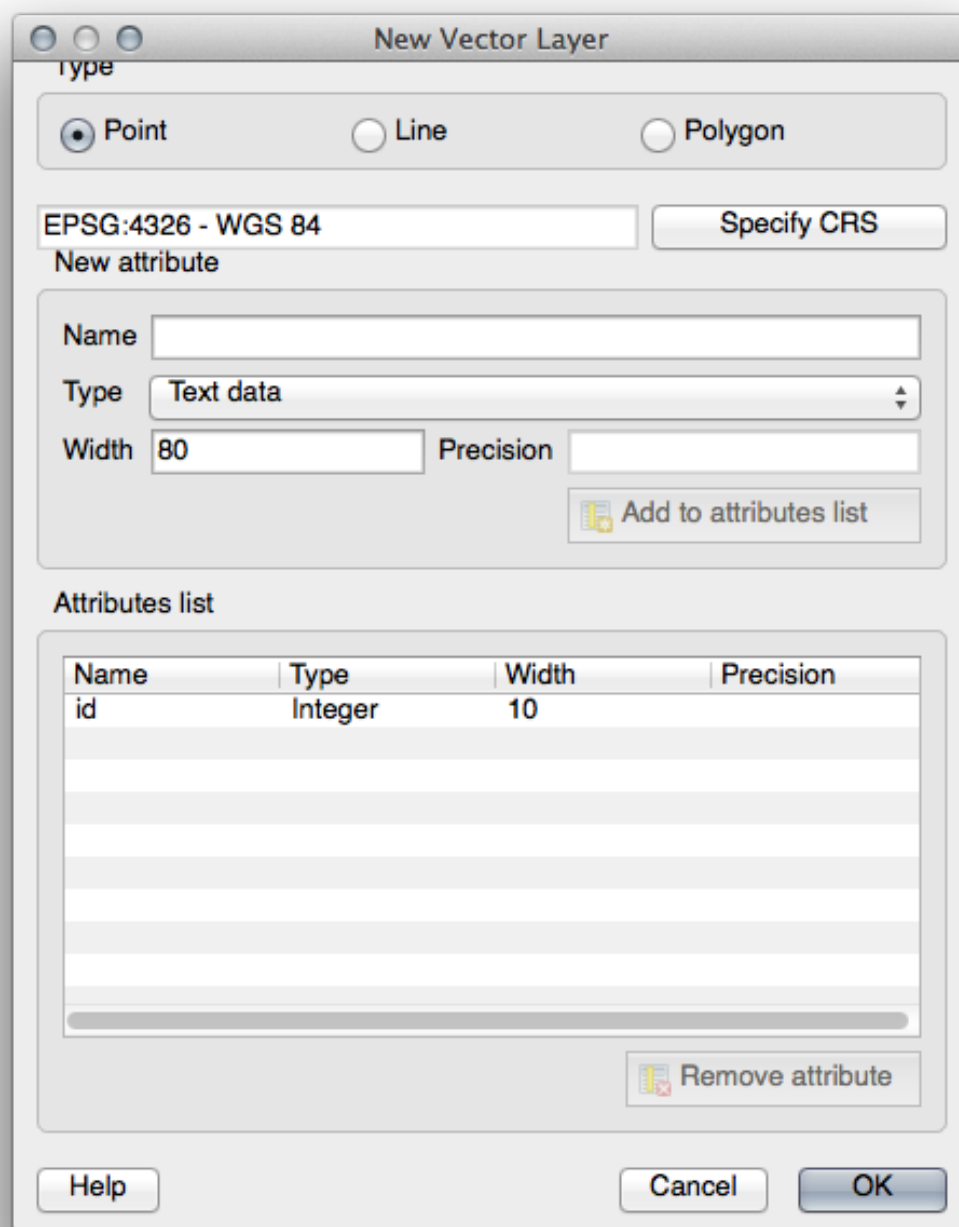
6.1.1 Follow Along: The Layer Creation Dialog

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.

You'll need to open the *New Vector Layer* dialog that will allow you to define a new layer.

- Navigate to and click on the menu entry *Layer → New → New Shapefile Layer*.

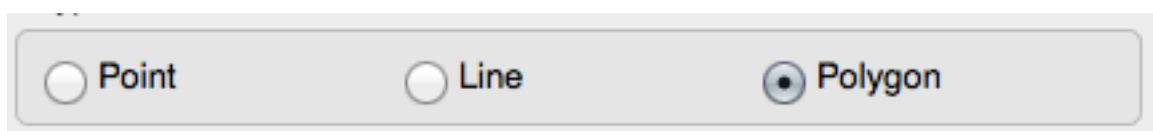
You'll be presented with the following dialog:



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 be creating new features which describe areas. For such features, you'll need to create a polygon dataset.

- Click on the *Polygon* radio button:



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. A CRS specifies how to describe a point on Earth in terms of coordinates, and because there are many different ways to do this, there are many different CRSs. The CRS of this project is WGS84, so it's already correct by default:

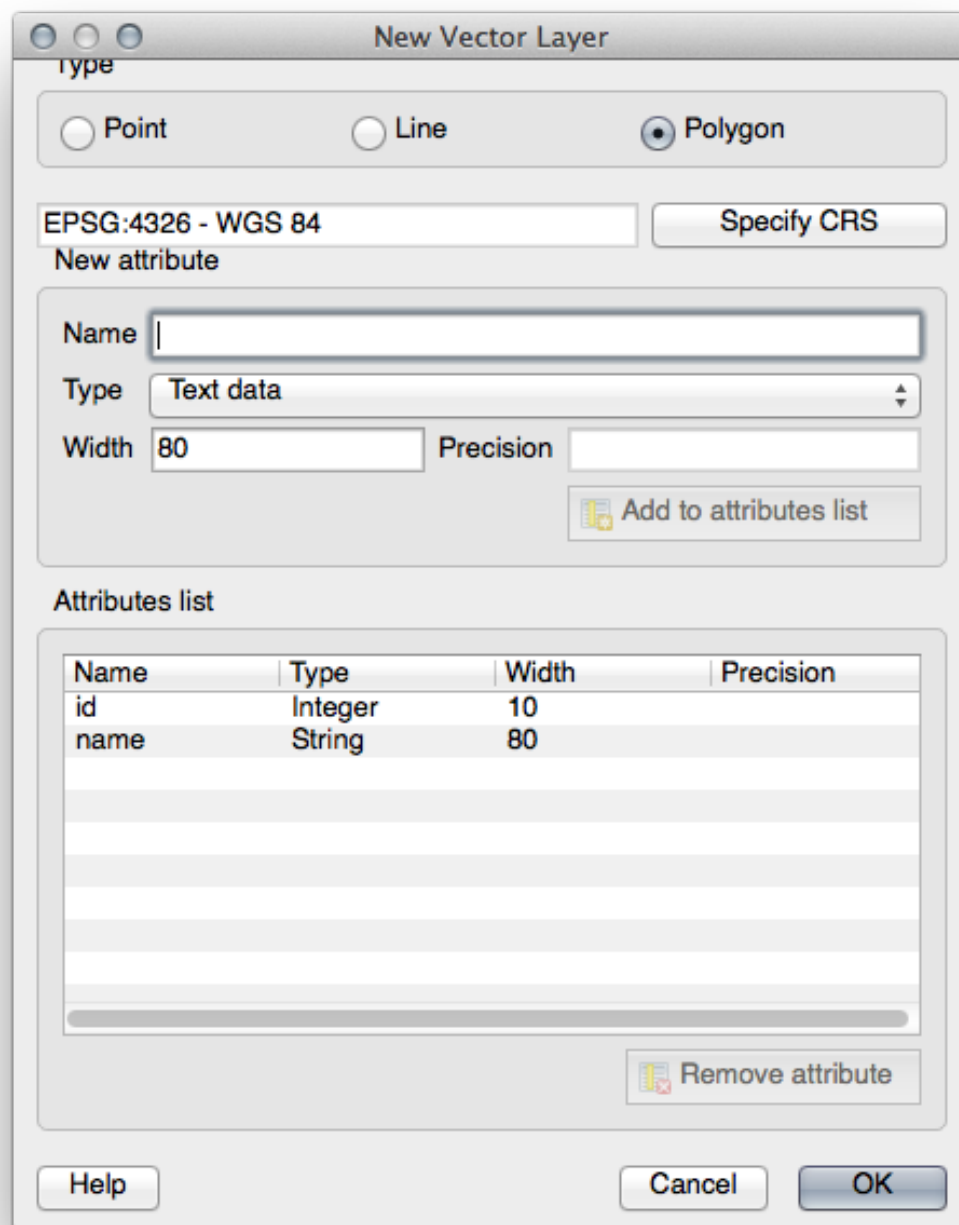


Next there is a collection of fields grouped under *New attribute*. By default, a new layer has only one attribute, the `id` field (which you should see in the *Attributes list*) below. However, in order for the data you create to be useful, you actually need to say something about the features you'll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

- Replicate the setup below, then click the *Add to attributes list* button:

A screenshot of the 'New attribute' section of a QGIS dialog. It contains three input fields: 'Name' with the value 'name', 'Type' with a dropdown menu showing 'Text data', and 'Width' with the value '80'. There is also an empty 'Precision' field. At the bottom right of this section is a button with a plus icon and the text 'Add to attributes list'.

- Check that your dialog now looks like this:



- Click *OK*. A save dialog will appear.
- Navigate to the `exercise_data` directory.
- Save your new layer as `school_property.shp`.


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

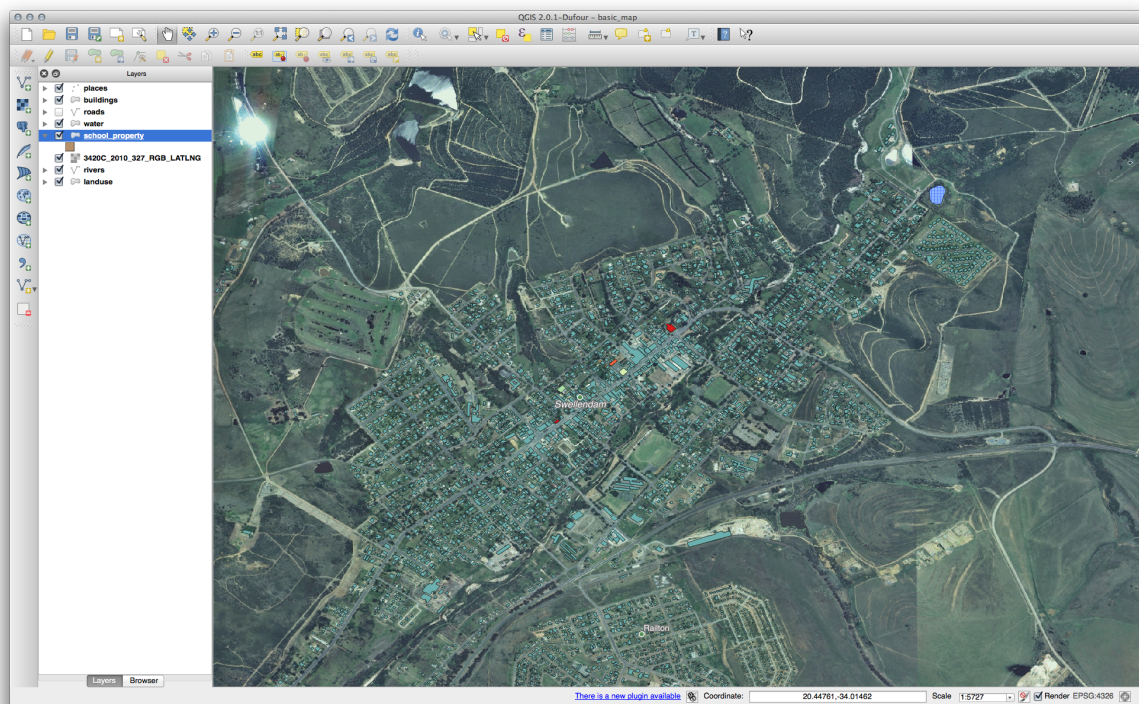
6.1.2 Follow Along: Data Sources

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.

- Click on the *Add Raster Layer* button: 
- Navigate to `exercise_data/raster/`.
- Select the file `3420C_2010_327_RGB_LATLNG.tif`.
- Click *Open*. An image will load into your map.
- Find the new image in the *Layers list*.
- Click and drag it to the bottom of the list so that you can still see your other layers.
- Find and zoom to this area:




Bemerkung: If your *buildings* layer symbology is covering part or all of the raster layer, you can temporarily disable the layer by deselecting it in the *Layers panel*. You may also wish to hide the *roads* symbology if you find it distracting.

You'll be digitizing these three fields:



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:

- Click on the layer in the *Layer list* to select it. (Make very sure that the correct layer is selected, otherwise you'll edit the wrong layer!)
- 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 the digitizing tools are now active:



Four other relevant buttons are still inactive, but will become active when we start interacting with our new data:



From left to right on the toolbar, they are:

- *Save Edits*: saves changes made to the layer.
- *Add Feature*: start digitizing a new feature.

- *Move Feature(s)*: move an entire feature around.
- *Node Tool*: move only one part of a feature.
- *Delete Selected*: delete the selected feature.
- *Cut Features*: cut the selected feature.
- *Copy Features*: copy the selected feature.
- *Paste Features*: paste a cut or copied feature back into the map.

You want to add a new feature.

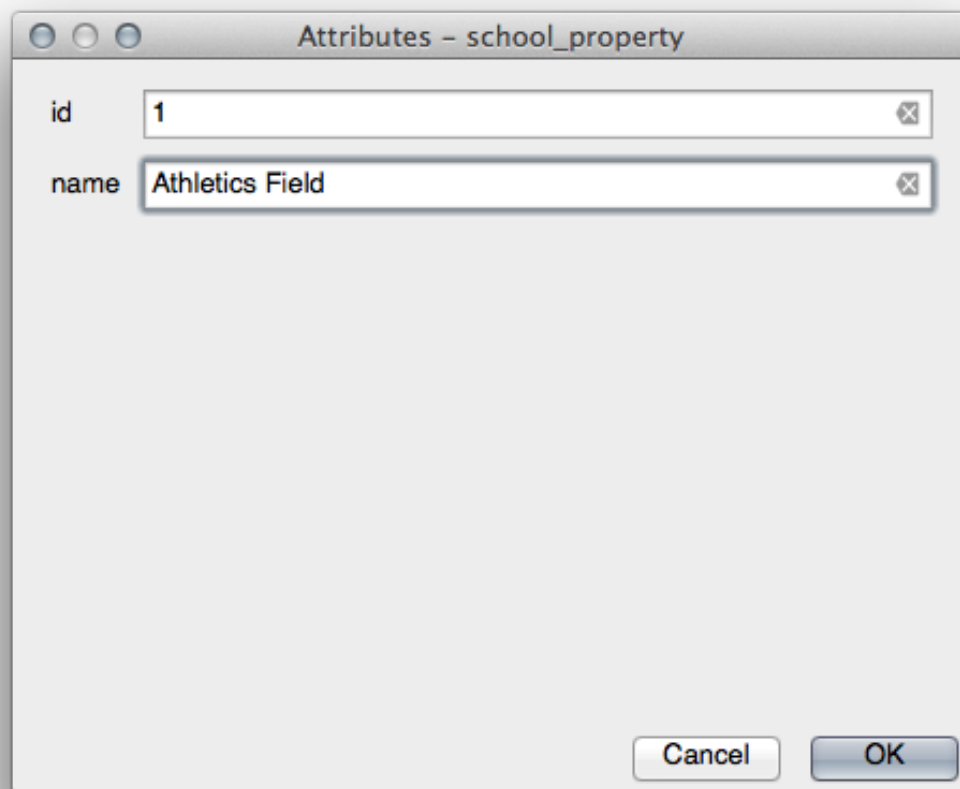
- Click on the *Add Feature* button now to begin digitizing our school fields.

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 as 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:



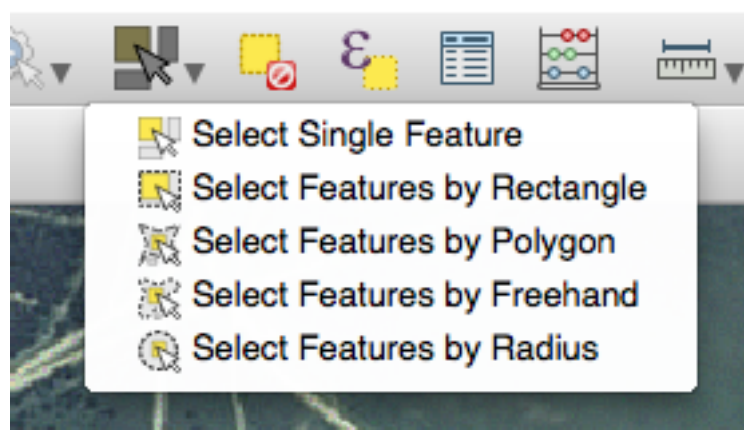
- Start digitizing by clicking on a point somewhere along the edge of the field.
- Place more points by clicking further along the edge, until the shape you're drawing completely covers the field.
- After placing your last point, *right-click* to finish drawing the polygon. This will finalize the feature and show you the *Attributes* dialog.
- Fill in the values as below:



- Click *OK* and you've created a new feature!

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:

- Select the feature with the *Select Single Feature* tool:



You can use:

- the *Move Feature(s)* tool to move the entire feature,
- the *Node Tool* to move only one point where you may have miss-clicked,
- *Delete Selected* to get rid of the feature entirely so you can try again, and

- the *Edit* → *Undo* menu item or the `ctrl + z` keyboard shortcut to undo mistakes.

6.1.3 Try Yourself

- 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!

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

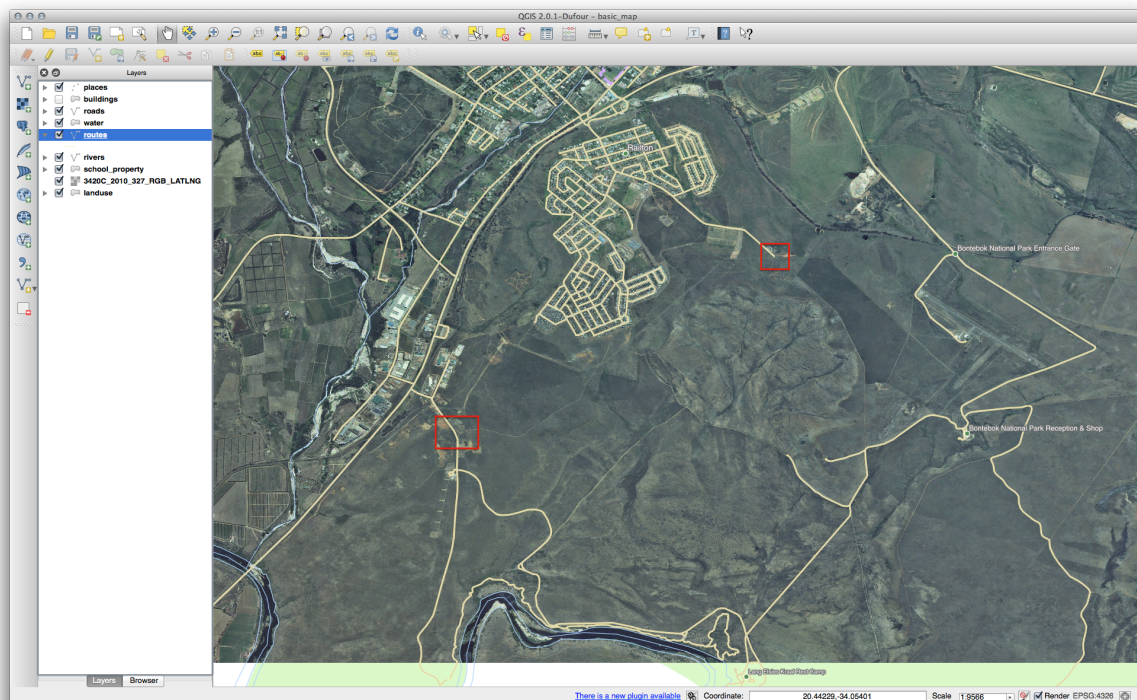
Bemerkung: You can style the fill, outline and label placement and formatting of the *school_property* using techniques learnt in earlier lessons. In our example, we will use a dashed outline of light purple color with no fill.

6.1.4 Try Yourself

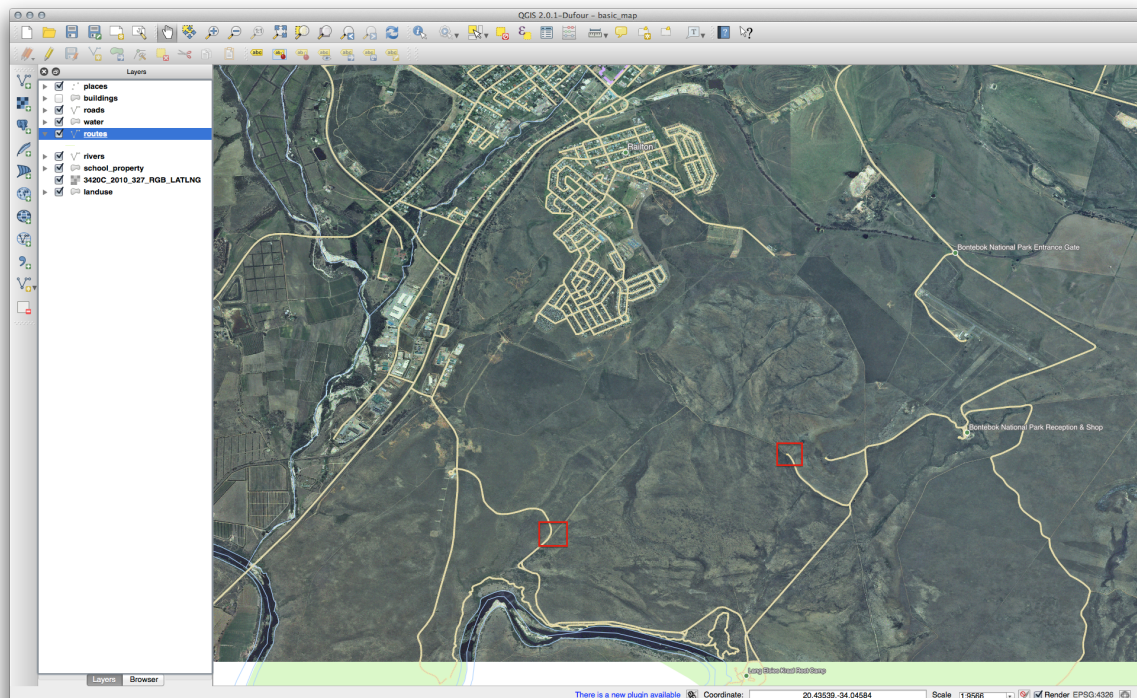
- Create a new line feature called `routes.shp` with attributes `id` and `type`. (Use the approach above to guide you.)

- We're 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:



One at a time, digitize the path and the track on the *routes* layer. Try to follow the routes as accurately as possible, using points (left-click) at any corners or turns.

When creating each route, give them the *type* attribute value of *path* or *track*.

You'll probably find that only the points are marked; use the *Layer Properties* dialog to add styling to your routes. Feel free to give different styles to the path and track.

Save your edits and toggle *Edit* mode.

Check your results

6.1.5 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.

6.1.6 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.

6.2 Lesson: Feature Topology

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

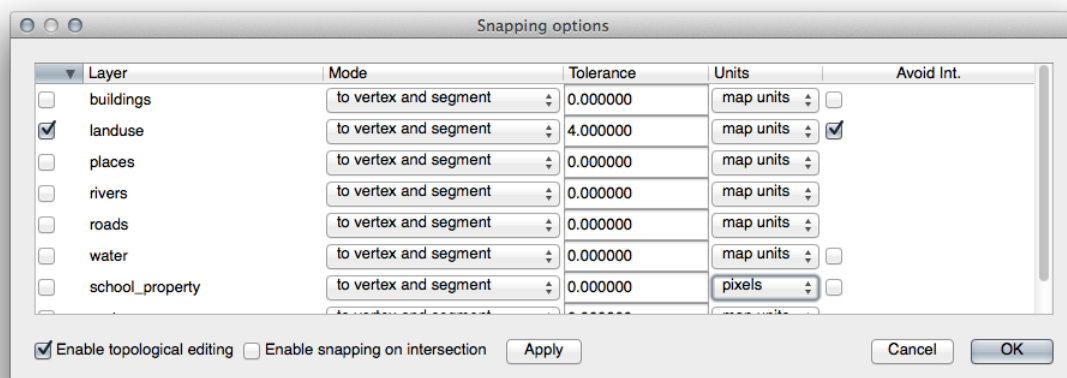
For example: if two features share a border, and you edit the border using topology, then you won't need to edit first one feature, then another, and carefully line up the borders so that they match. Instead, you can edit their shared border and both features will change at the same time.

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

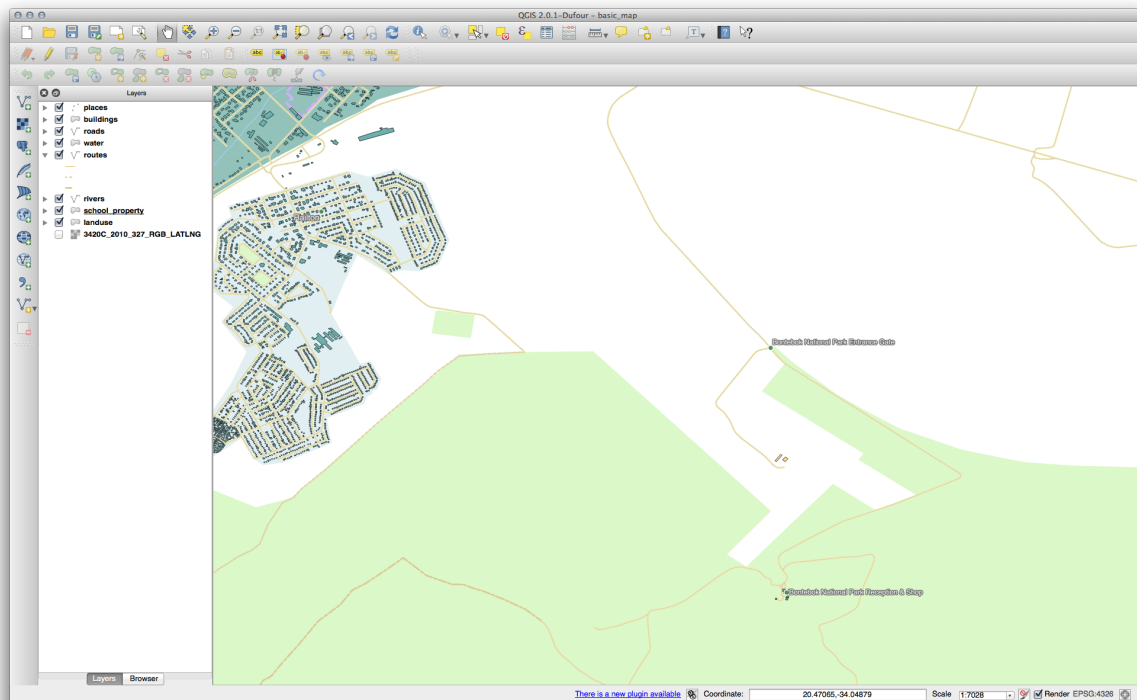
6.2.1 Follow Along: Snapping

To make topological editing easier, it's best if you enable snapping. This will allow your mouse cursor to snap to other objects while you digitize. To set snapping options:

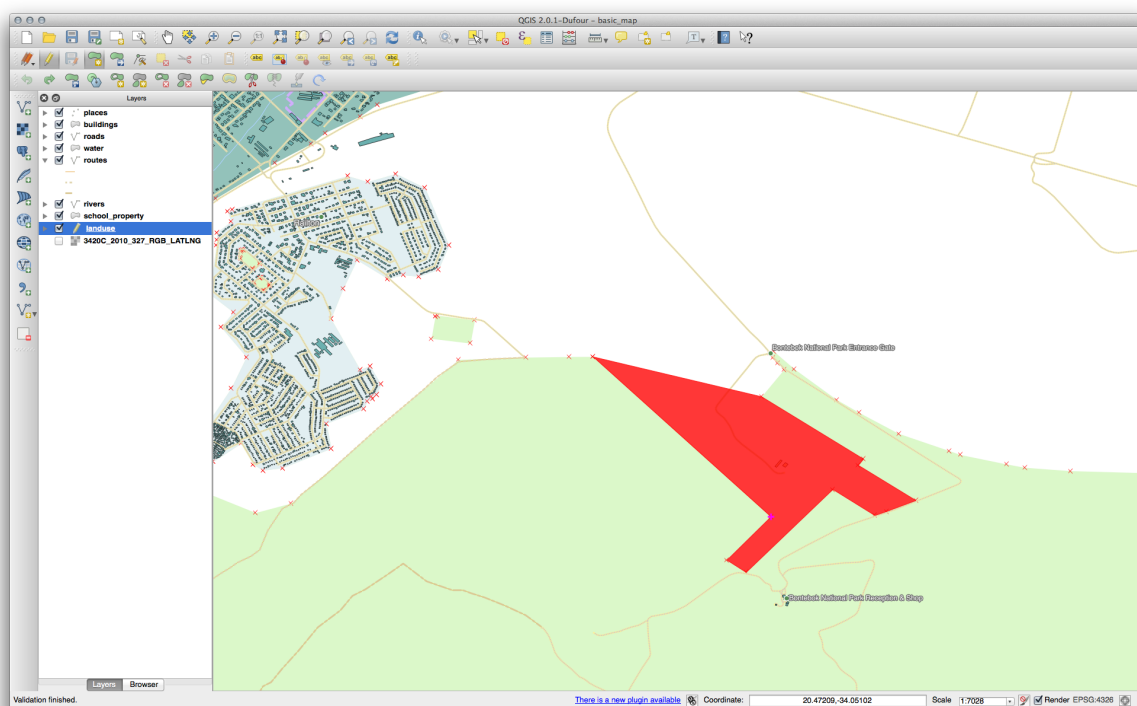
- Navigate to the menu entry *Settings* → *Snapping Options*....
- Set up your *Snapping options* dialog as shown:



- Ensure that the box in the *Avoid Int.* column is checked (set to true).
- Click *OK* to save your changes and leave the dialog.
- Enter edit mode with the *landuse* layer selected.
- Check under *View* → *Toolbars* to make sure that your *Advanced Digitizing* toolbar is enabled.
- Zoom to this area (enable layers and labels if necessary):



- Digitize this new (fictional) area of the Bontebok National Part:



- When prompted, give it a *OGC_FID* of 999, but feel free to leave the other values unchanged.

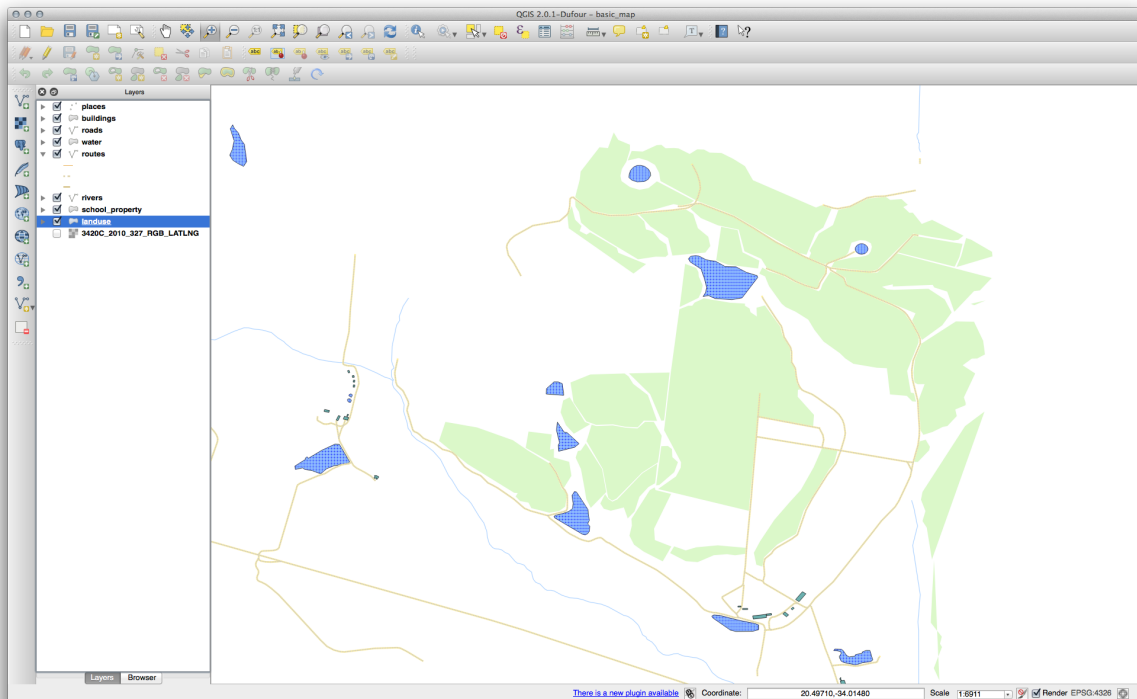
If you're careful while digitizing and allow the cursor to snap to the vertices of adjoining farms, you'll notice that there won't be any gaps between your new farm and the existing farms adjacent to it.

- Note the undo/redo tools in the *Advanced Digitizing* toolbar:



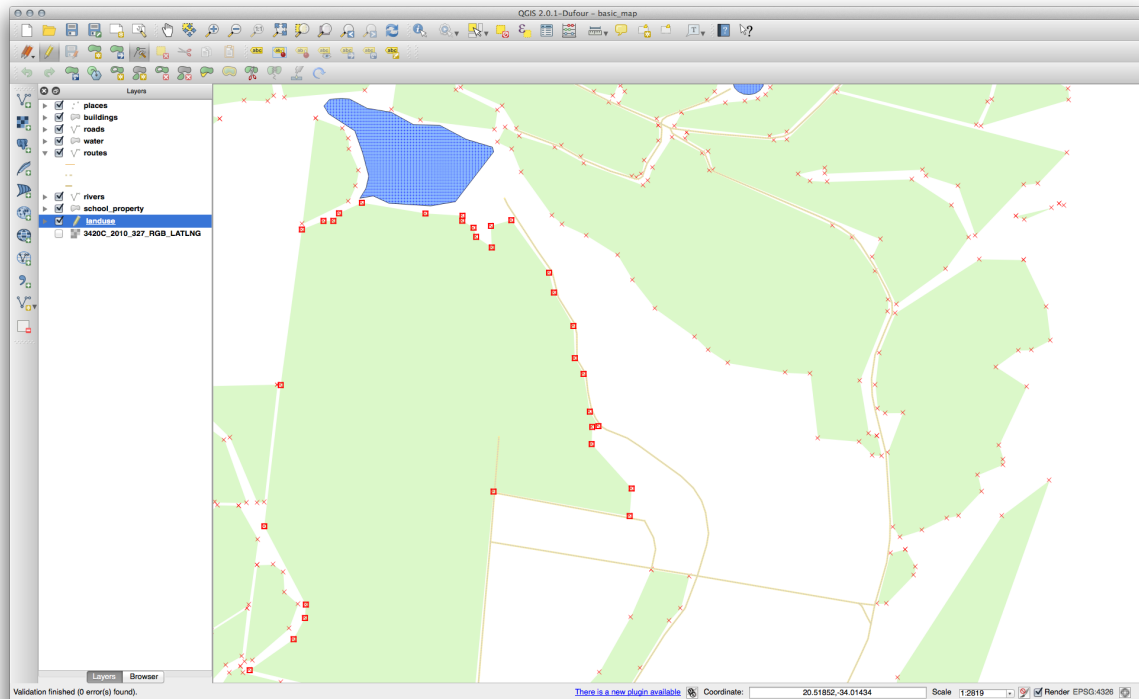
6.2.2 Follow Along: Correct Topological Features

Topology features can sometimes need to be updated. In our example, the *landuse* layer has some complex forest areas which have recently been joined to form one area:

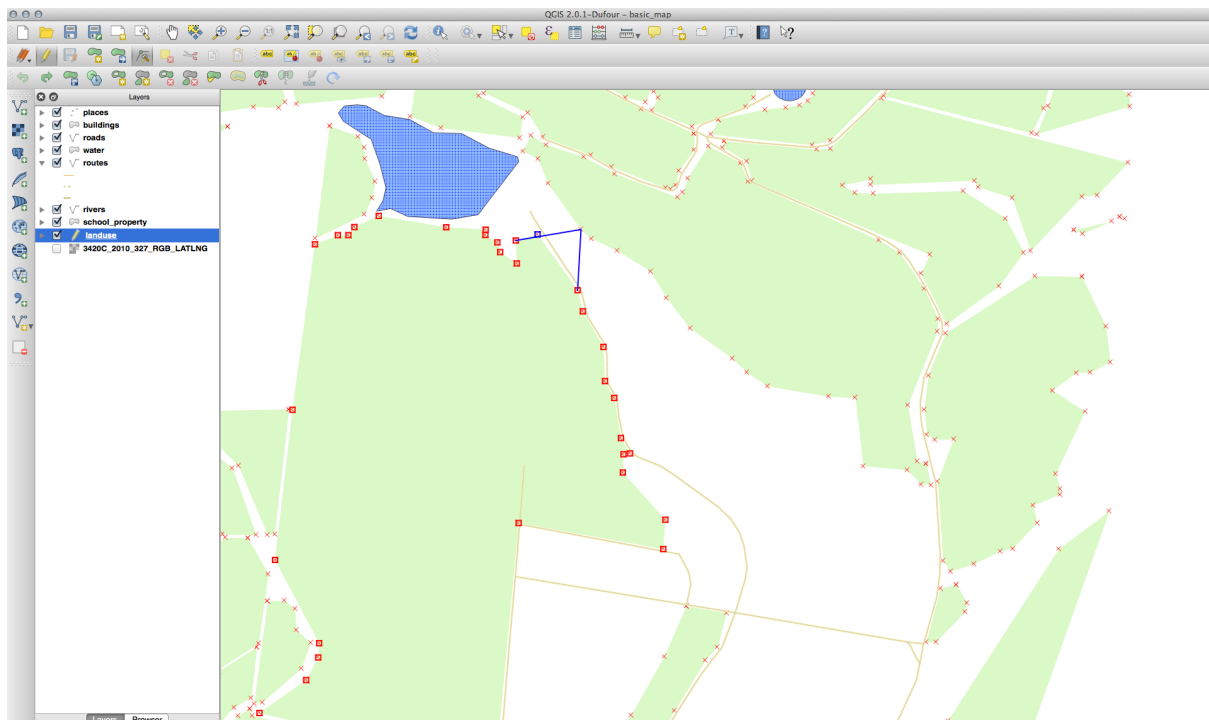


Instead of creating new polygons to join the forest areas, we're going to use the *Node Tool* to edit the existing polygons and join them.

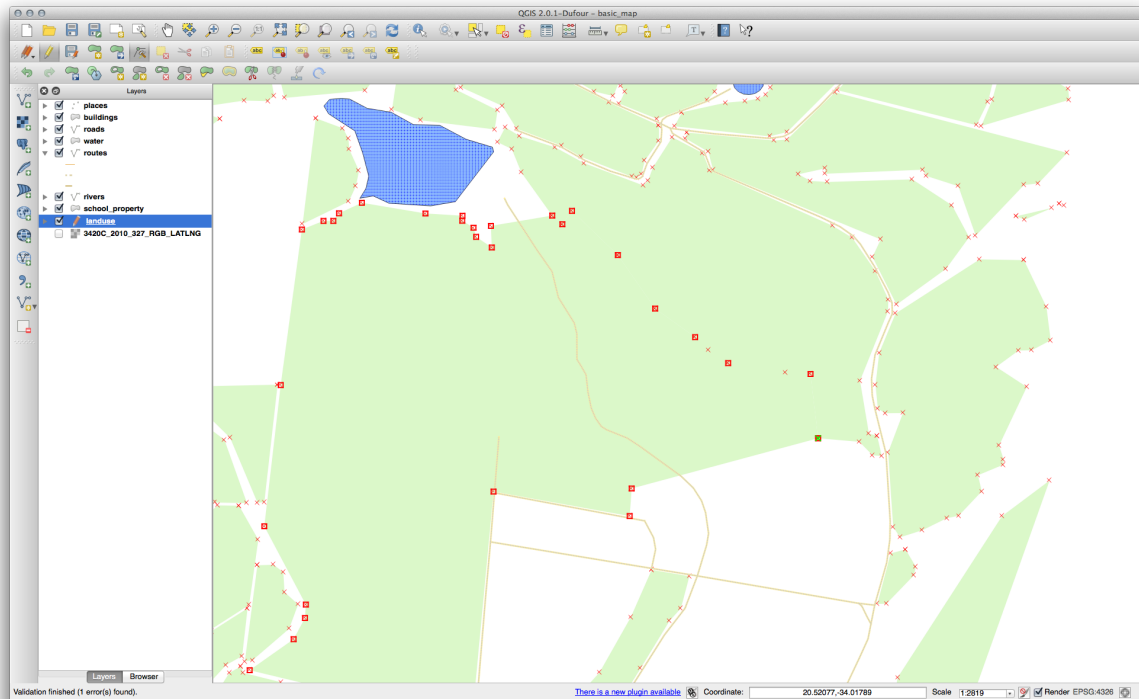
- Enter edit mode, if it isn't active already.
- Select the *Node Tool*.
- Pick an area of forest, select a corner and move it to an adjoining corner so two forest sections meet:



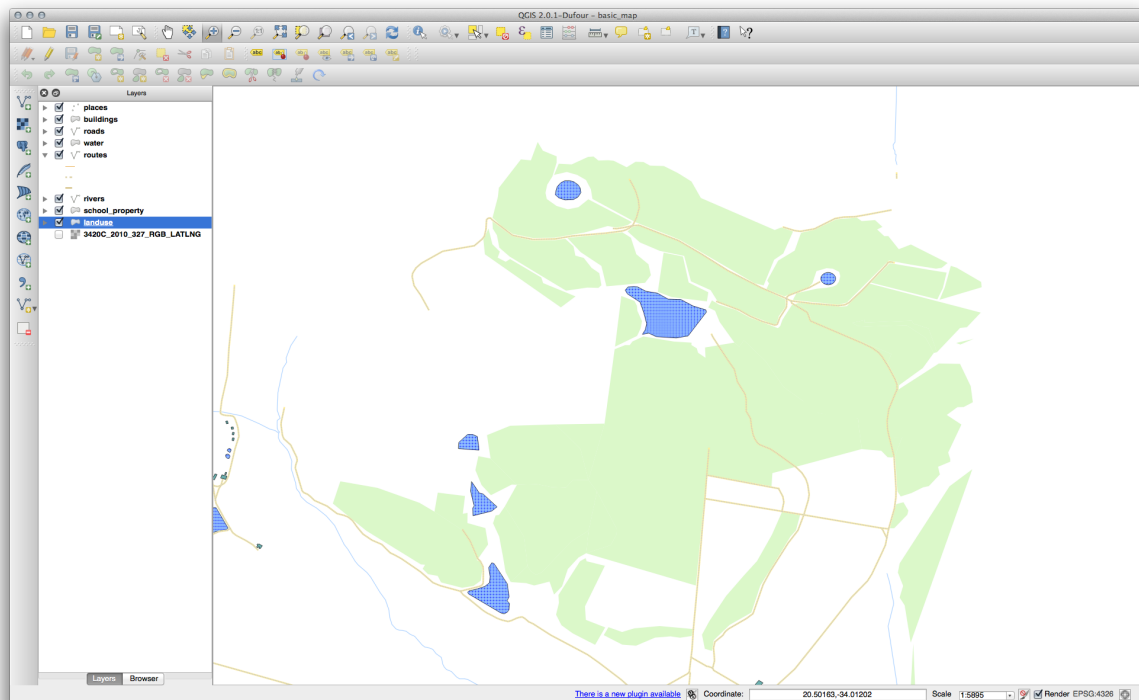
- Click and drag the nodes until they snap into place.



The topologically correct border looks like this:



Go ahead and join a few more areas using the *Node Tool*. You can also use the *Add Feature* tool if it is appropriate. If you are using our example data, you should have a forest area looking something like this:



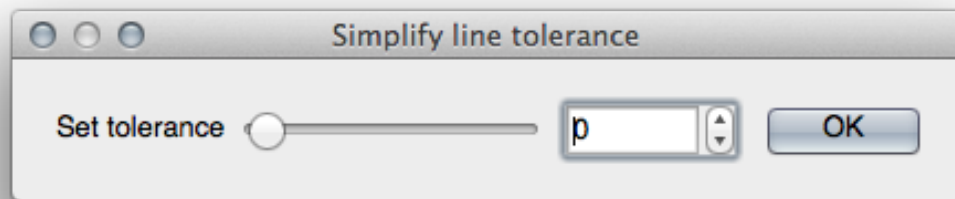
Don't worry if you have joined more, less or different areas of forest.

6.2.3 Follow Along: Tool: Simplify Feature

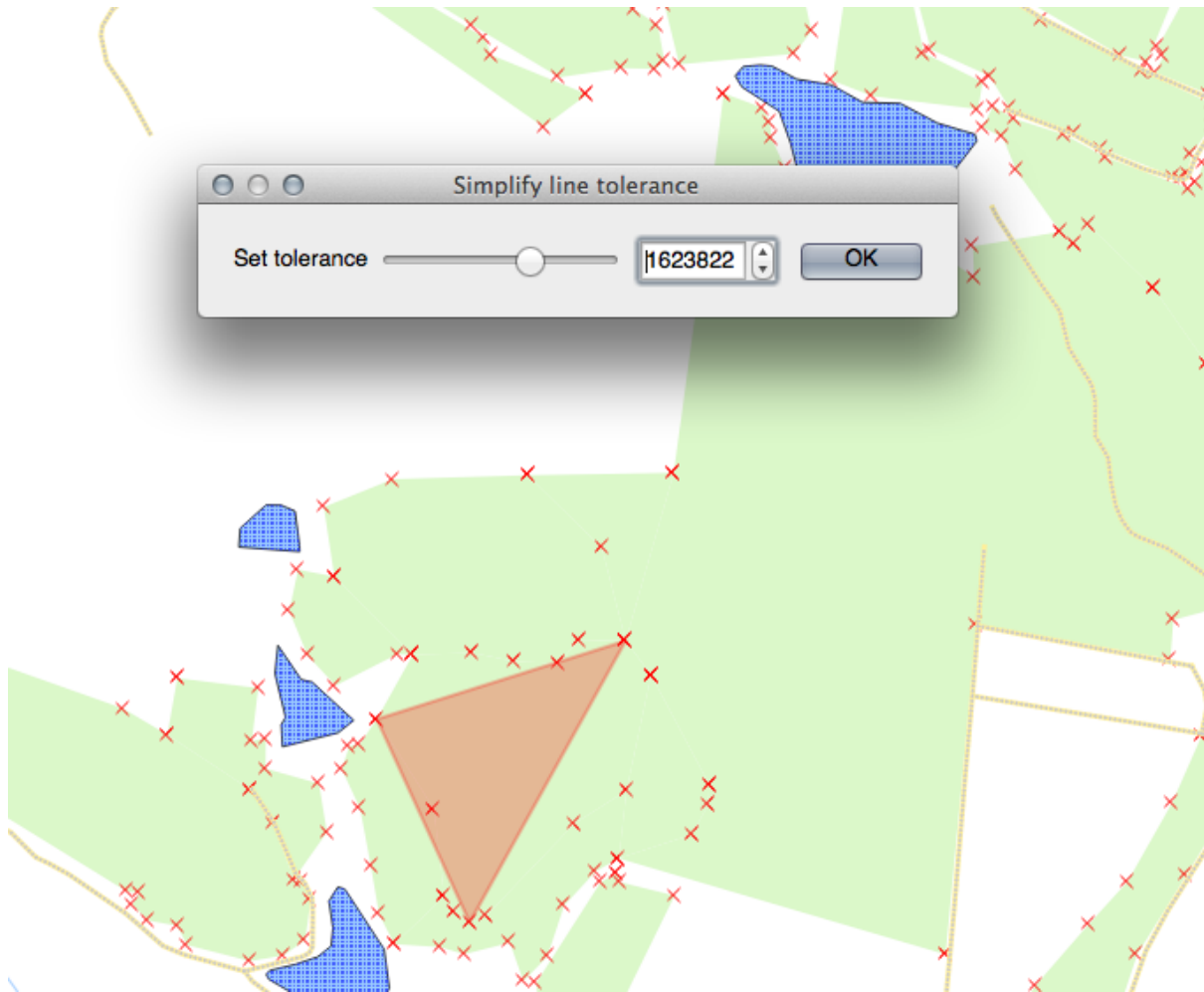
This is the *Simplify Feature* tool:



- Click on it to activate it.
- Click on one of the areas which you joined using either the *Node Tool* or *Add Feature* tool. You'll see this dialog:



- Move the slider from side to side and watch what happens:



This allows you to reduce the amount of nodes in complex features.

- Click *Ok*

Notice what the tool does to the topology. The simplified polygon is now no longer touching the adjacent polygons as it should. This shows that this tool is better suited to generalizing stand-alone features. The advantage is that it provides you with a simple, intuitive interface for generalization.

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

6.2.4 Try Yourself Tool: Add Ring

This is the *Add Ring* tool:



It allows you to take a hole out of a feature, as long as the hole is bounded on all side by the feature. For example, if you've digitized the outer boundaries of South Africa and you need to add a hole for Lesotho, you'd use this tool.

If you experiment with this tool, you'll notice that the current snapping options prevent you from creating a ring in the middle of the polygon. This would be fine if the area you wished to exclude linked to the polygon's boundaries.

- Disable snapping for the landuse layer via the dialog you used earlier.
- Now try using the *Add Ring* tool to create a gap in the middle of the Bontebok National Park.
- Delete your new feature by using the *Delete Ring* tool:



Bemerkung: You need to select a corner of the ring in order to delete it.

Check your results

6.2.5 Try Yourself Tool: Add Part

This is the *Add Part* tool:



It allows you to create an extra part of the feature, not directly connected to the main feature. For example, if you've digitized the boundaries of mainland South Africa but you haven't yet added the Prince Edward Islands, you'd use this tool to create them.

- To use this tool, you must first select the polygon to which you wish to add the part by using the *Select Single Feature* tool:



- Now try using the *Add Part* tool to add an outlying area to the Bontebok National Part.
- Delete your new feature by using the *Delete Part* tool:



Bemerkung: You need to select a corner of the part in order to delete it.

Check your results

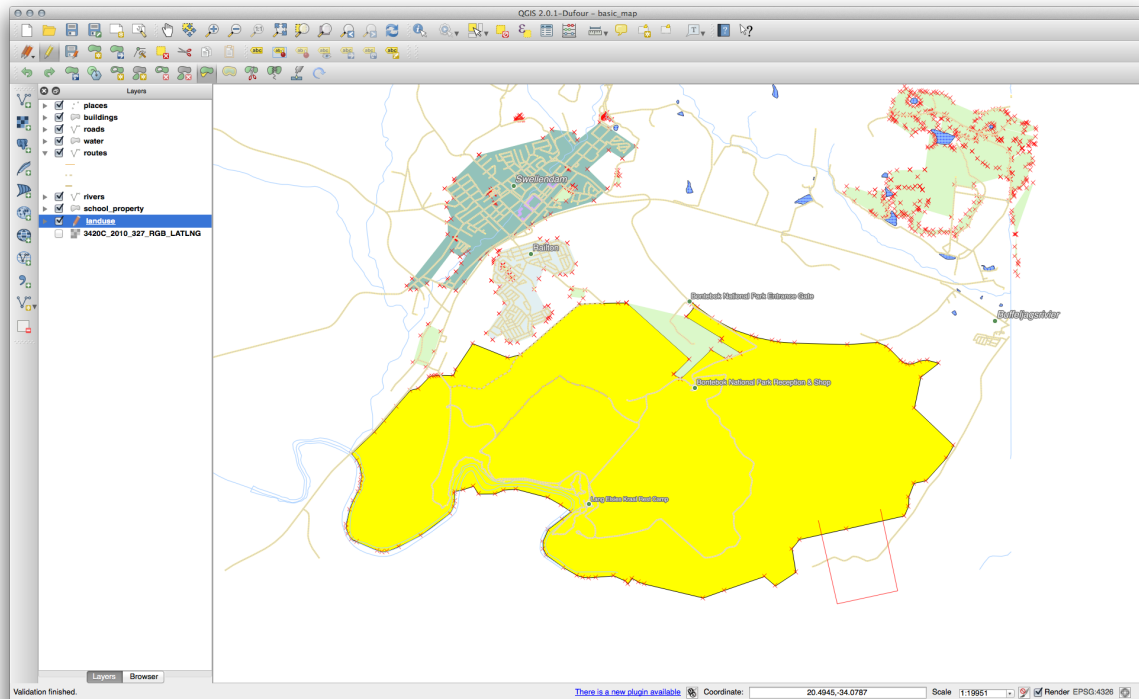
6.2.6 Follow Along: Tool: Reshape Features

This is the *Reshape Features* tool:

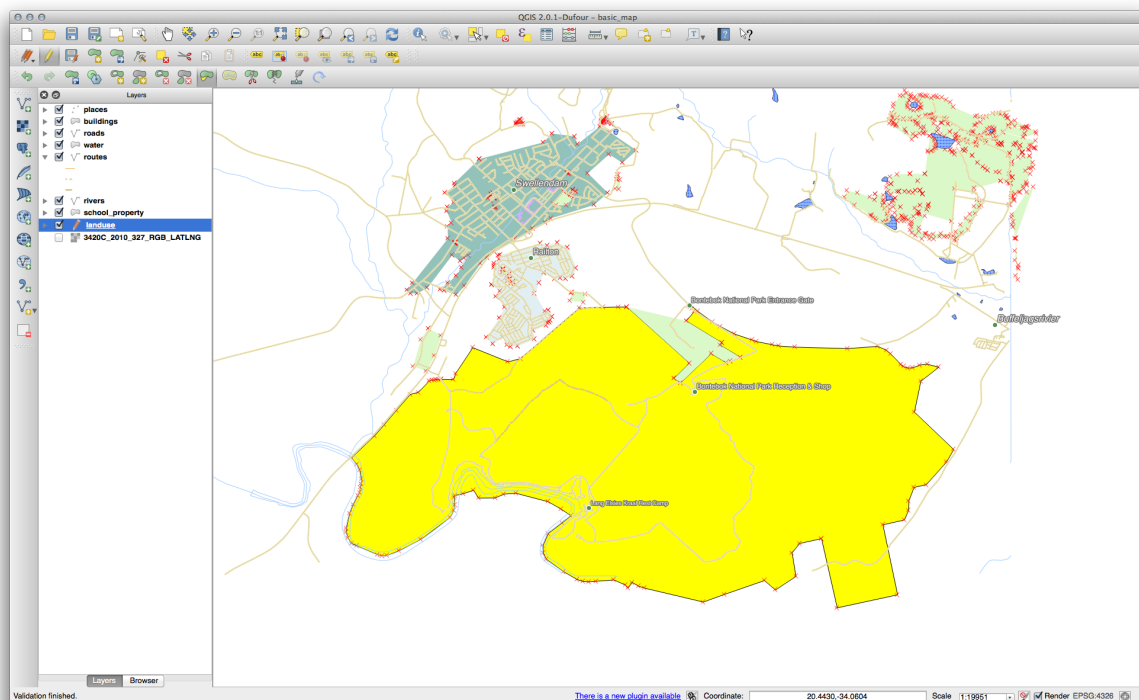


It can add a bump to an existing feature. With this tool selected:

- Left-click inside the Bontebok National Part to start drawing a polygon.
- Draw a polygon with three corners, the last of which should be back inside the original polygon, forming an open-sided rectangle.
- Right-click to finish marking points:

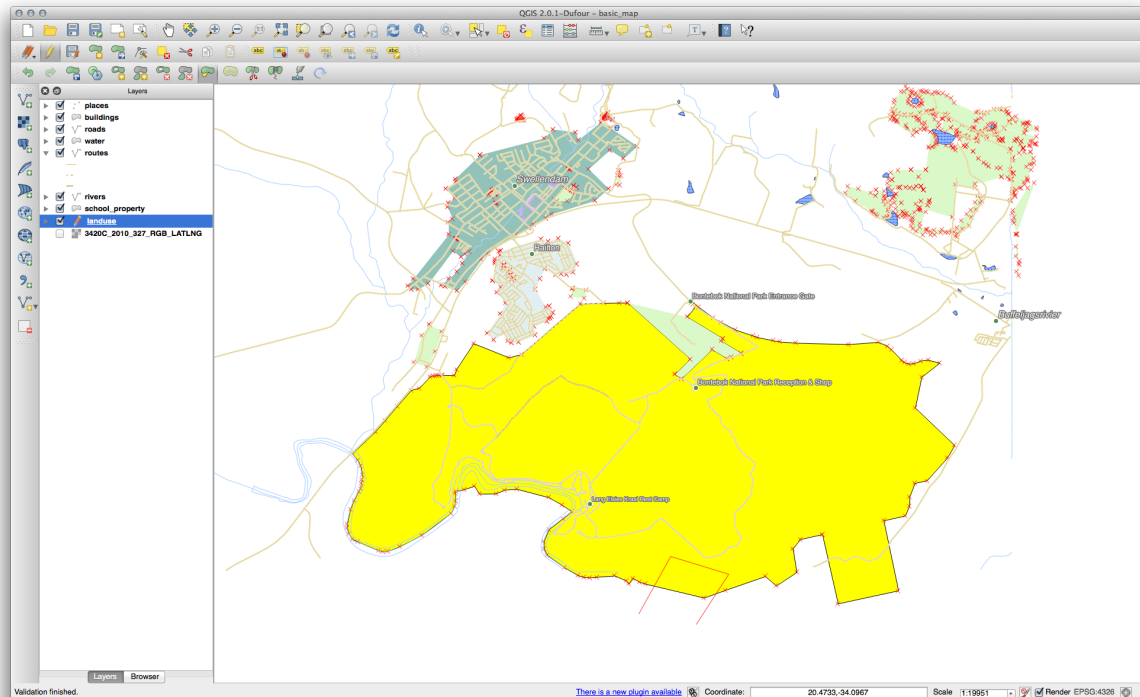


This will give a result similar to:

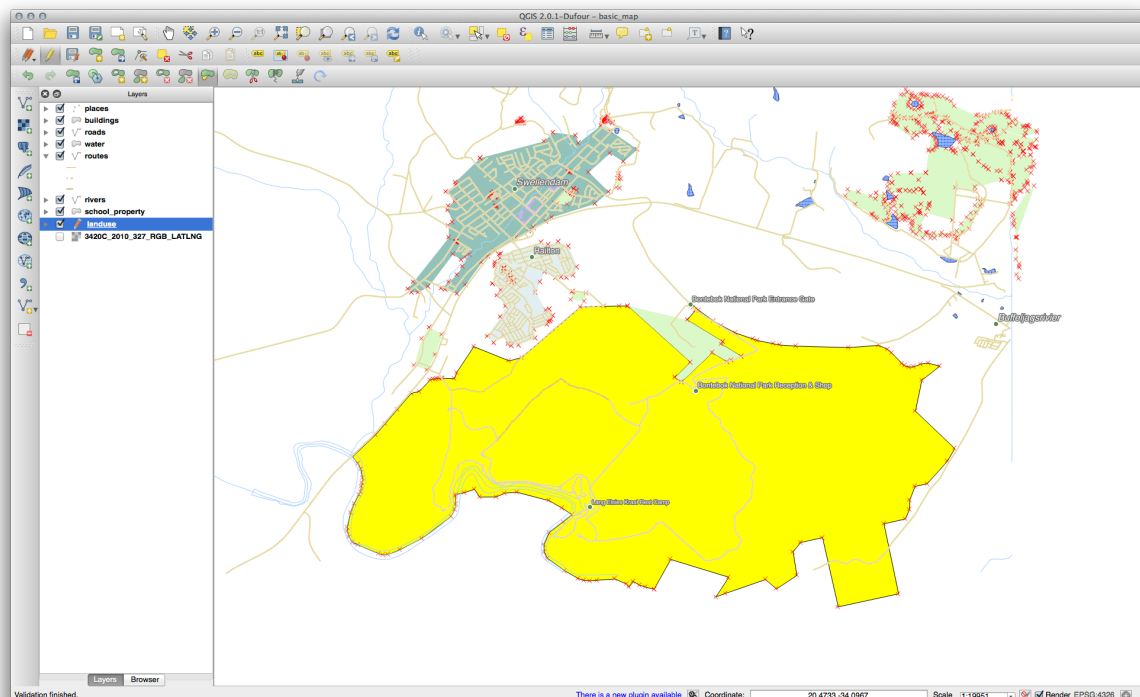


You can do the opposite, too:

- Click outside the polygon.
- Draw a rectangle into the polygon.
- Right-click outside the polygon again:



The result of the above:



6.2.7 Try Yourself Tool: Split Features

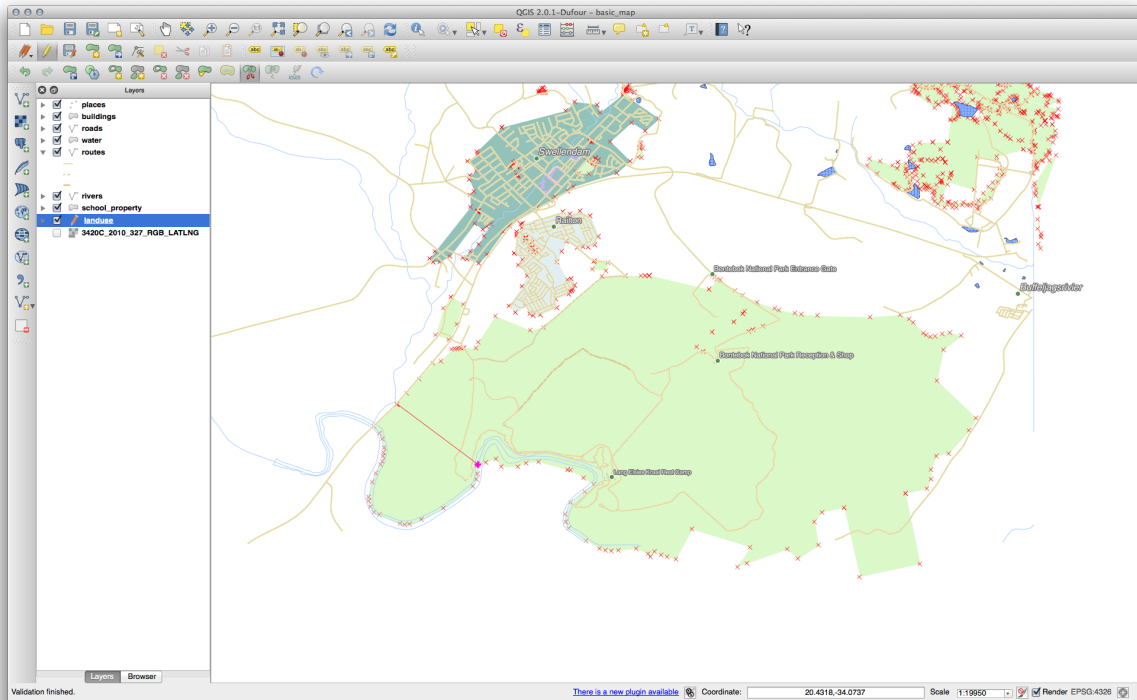
The *Split Features* tool is similar to how you took part of the farm away, except that it doesn't delete either of the two parts. Instead, it keeps them both.



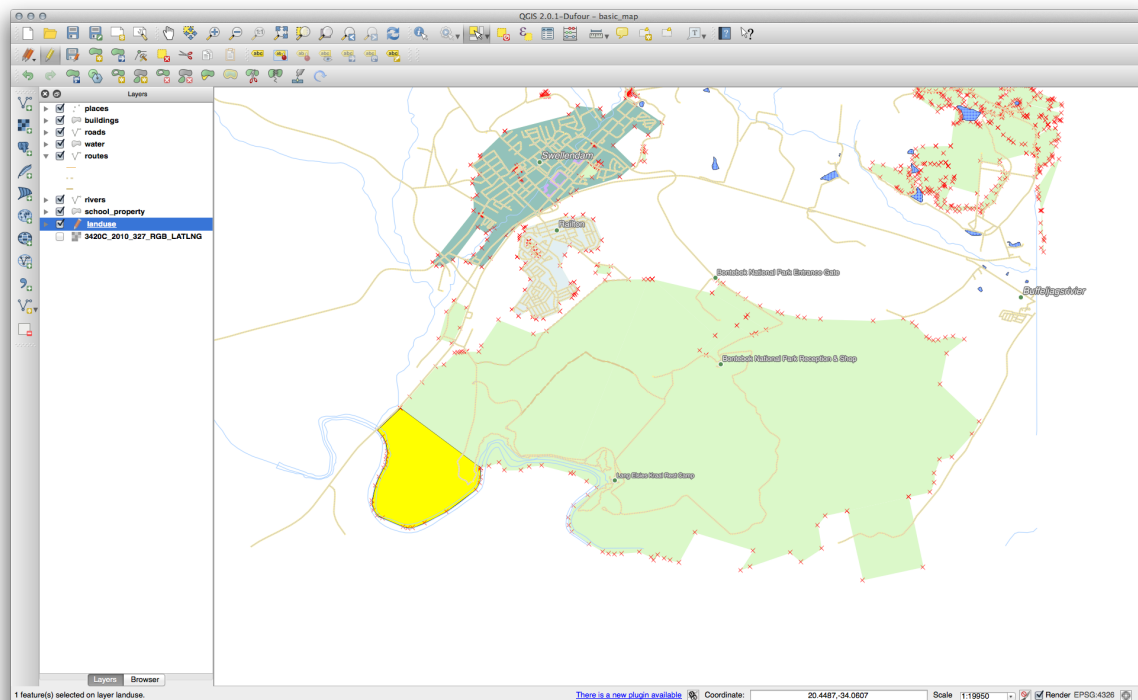
- First, re-enable snapping for the *landuse* layer.

We will use the tool to split a corner from the Bontebok National Park.

- Select the *Split Features* tool and click on a vertex to begin drawing a line. Click the vertex on the opposite side of the corner you wish to split and right-click to complete the line:



- At this point, it may seem as if nothing has happened. But remember that your symbology for the *landuse* layer does not have any border, so the new division line will not be shown.
- Use the *Select Single Feature* tool to select the corner you just split; the new feature will now be highlighted:



6.2.8 Try Yourself Tool: Merge Features

Now we will re-join the feature you just created to the original polygon:

- Experiment with the *Merge Selected Features* and *Merge Attributes of Selected Features* tools.
- Note the differences.

Check your results

6.2.9 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.

6.2.10 What's Next?

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

6.3 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.

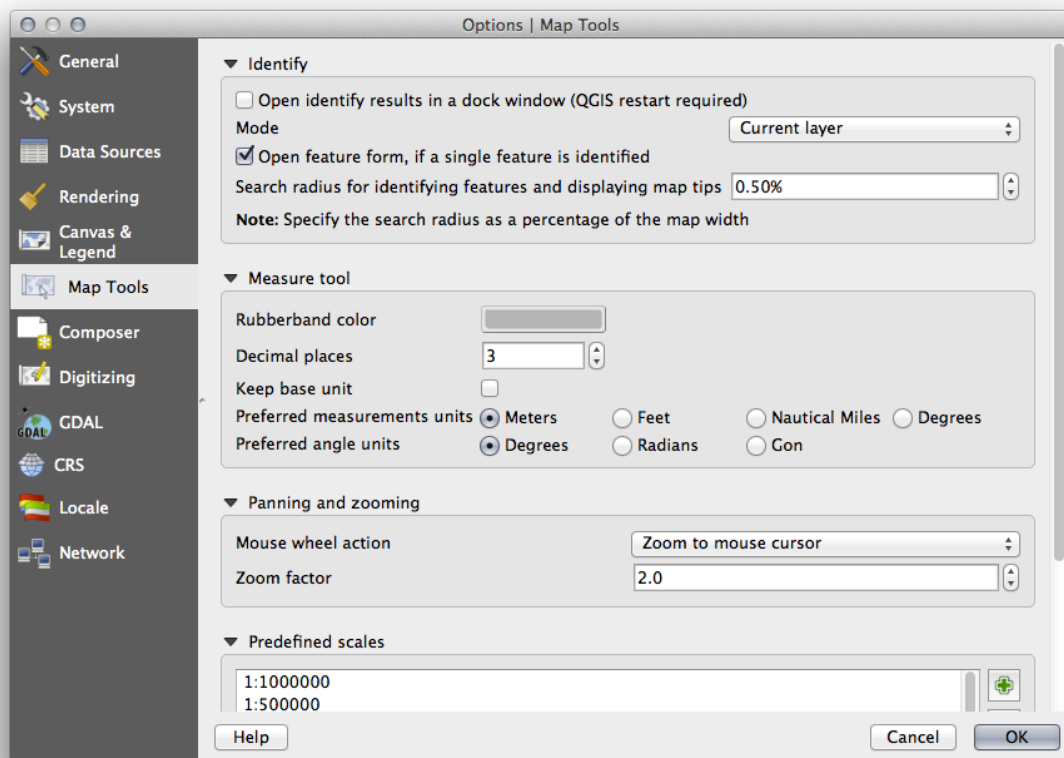
The goal for this lesson: To create a form for a layer.

6.3.1 Follow Along: Using QGIS' Form Design Functionality

- Select the *roads* layer in the *Layers list*.
- Enter *Edit Mode* as before.
- Open its *Attribute Table*.
- Right-click on any cell in the table. A short menu will appear, with the only entry being *Open form*.
- 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.

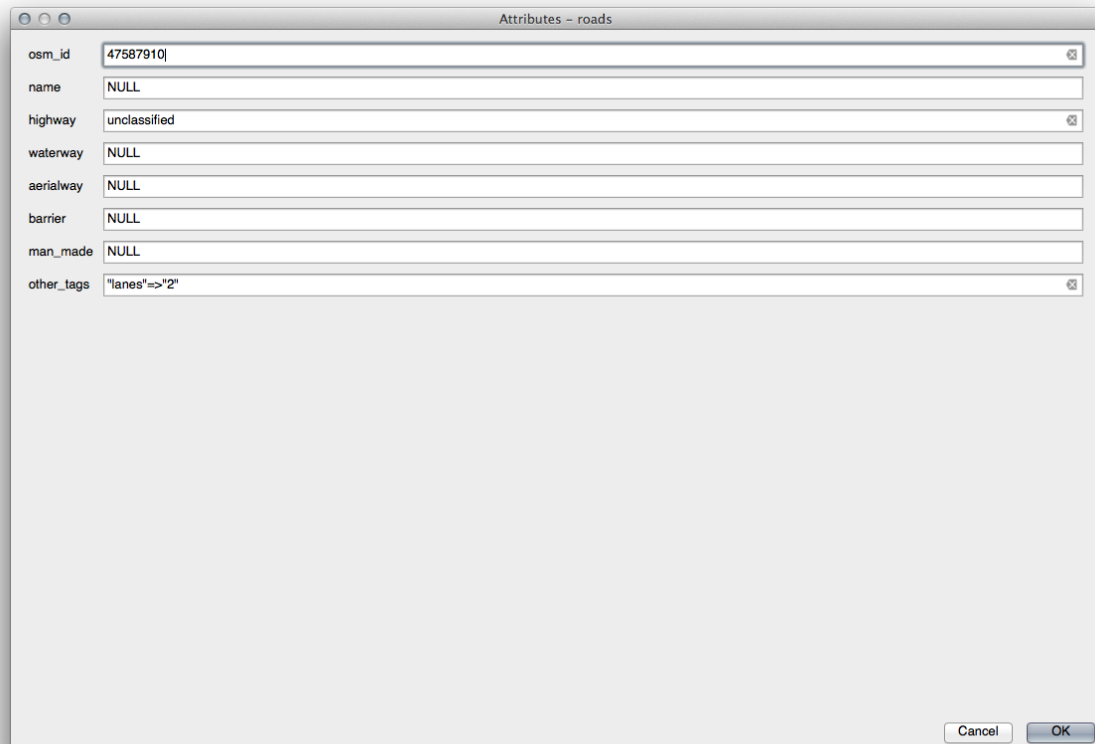
- Go to the *Settings* → *Options* menu.
- In the dialog that appears, select the *Map Tools* tab.
- Check the *Open feature form ...* checkbox:



- Click *Ok*.
- Select the *roads* layer in the *Layers list*.
- Using the *Identify* tool, click on any street in the map.



Instead of the normal *Identify* dialog, you'll see the now-familiar form instead:

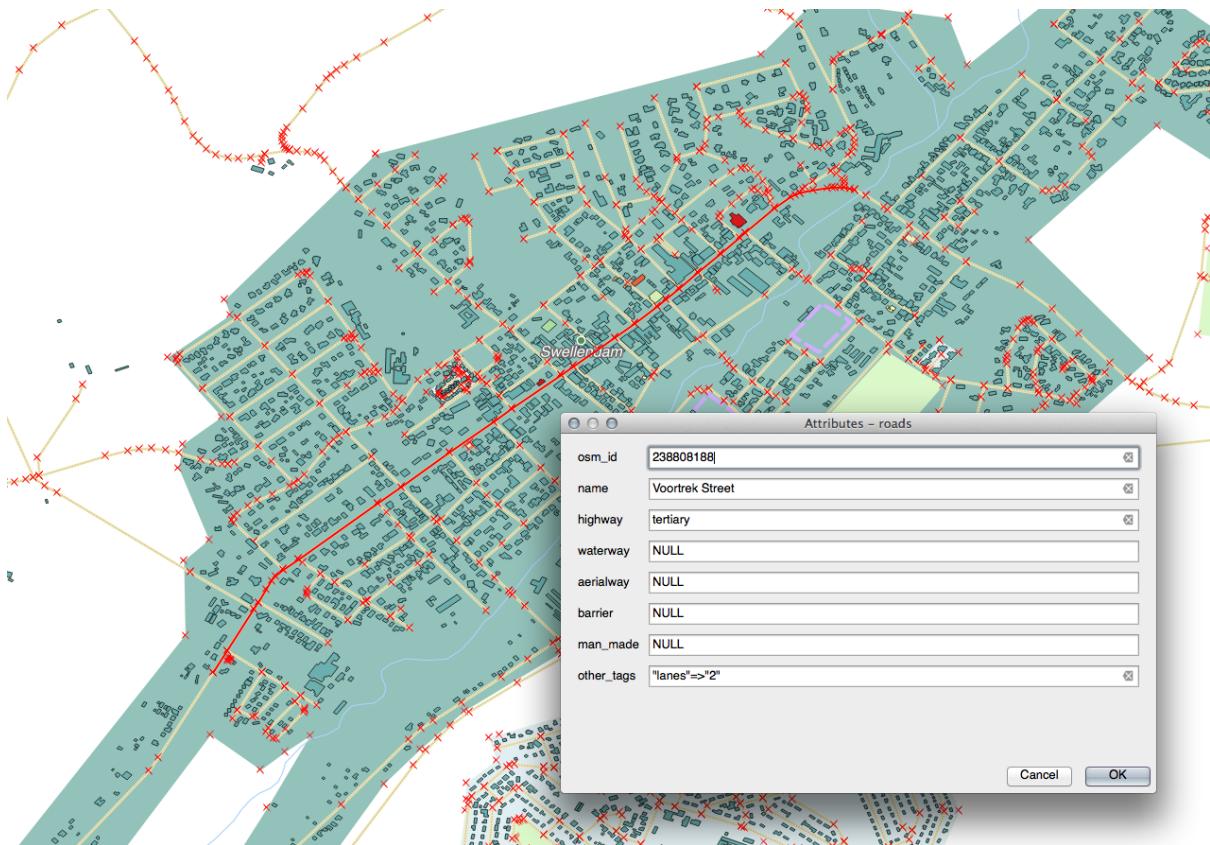


| Attribute | Value |
|------------|--------------|
| osm_id | 47587910 |
| name | NULL |
| highway | unclassified |
| waterway | NULL |
| aerialway | NULL |
| barrier | NULL |
| man_made | NULL |
| other_tags | "lanes"=>2" |

6.3.2 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.

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



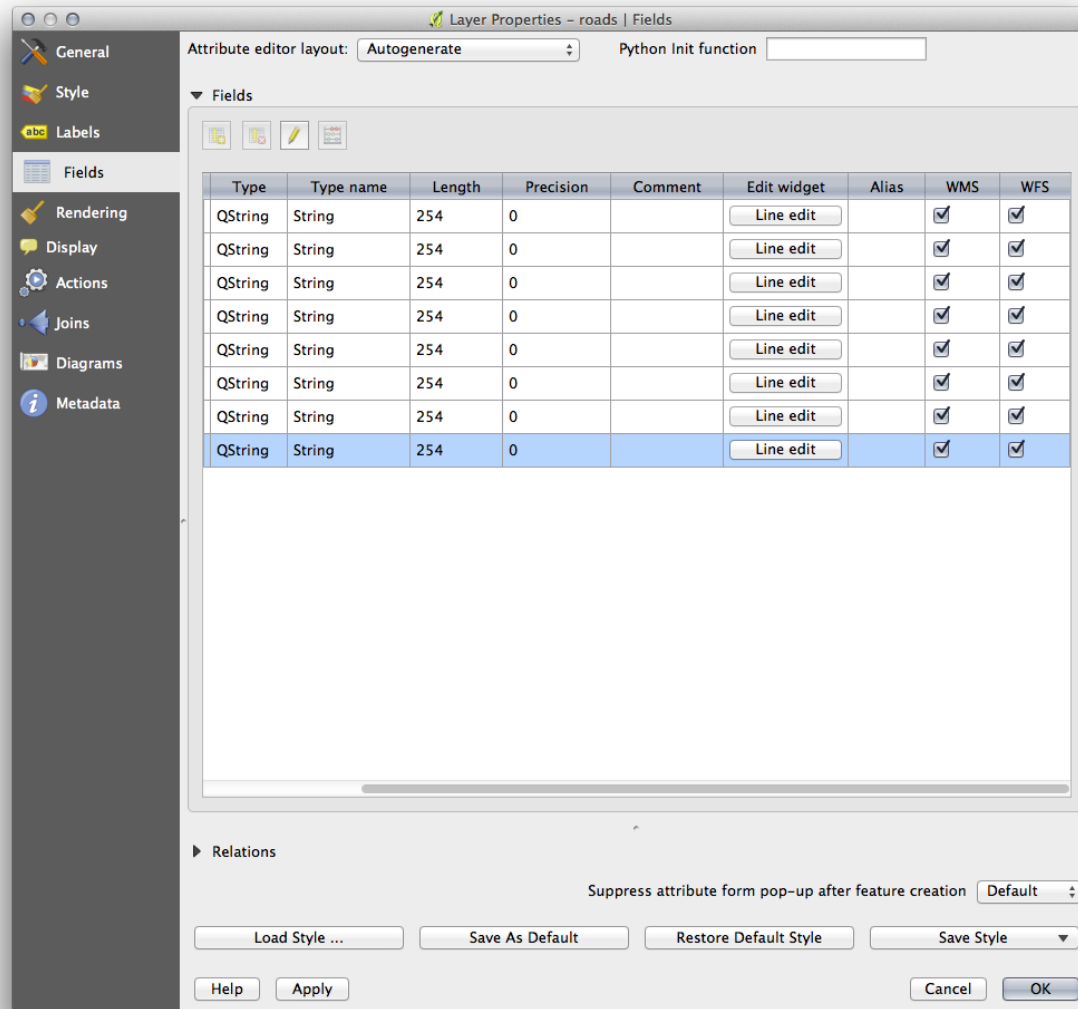
- Edit its *highway* value to be *secondary*.
- Save your edits.
- Exit edit mode.
- Open the *Attribute Table* and note that the value has been updated in the attributes table and therefore in the source data.

Bemerkung: If you're using the default dataset, you'll find that there is more than one road on this map called Voortrek Street.

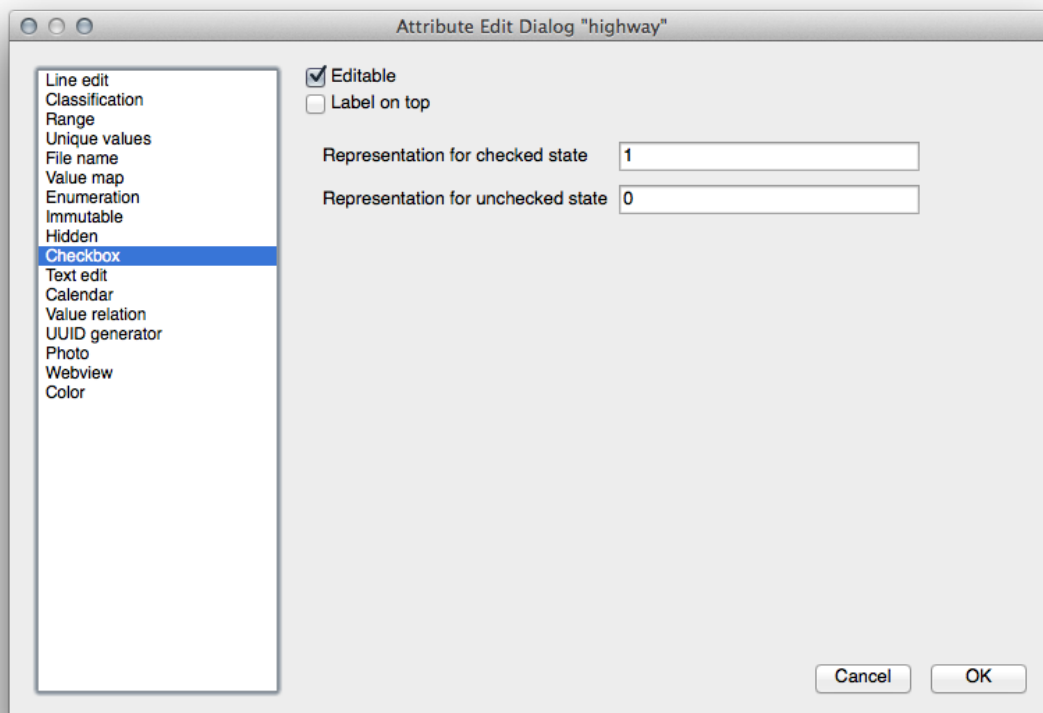
6.3.3 Follow Along: Setting Form Field Types

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.

- Open the *roads* layer's *Layer Properties*.
- Switch to the *Fields* tab. You'll see this:



- Click on the *Line edit* button in the same row as *man_made* and you'll be given a new dialog.
- Select *Checkbox* in the list of options:



- Click *OK*.
- Enter edit mode (if the *roads* layer is not already in edit mode).
- Click on the *Identify* tool.
- Click on the same main road you chose earlier.

You'll now see that the *man_made* attribute has a checkbox next to it denoting *True* (checked) or *False* (unchecked).

6.3.4 Try Yourself

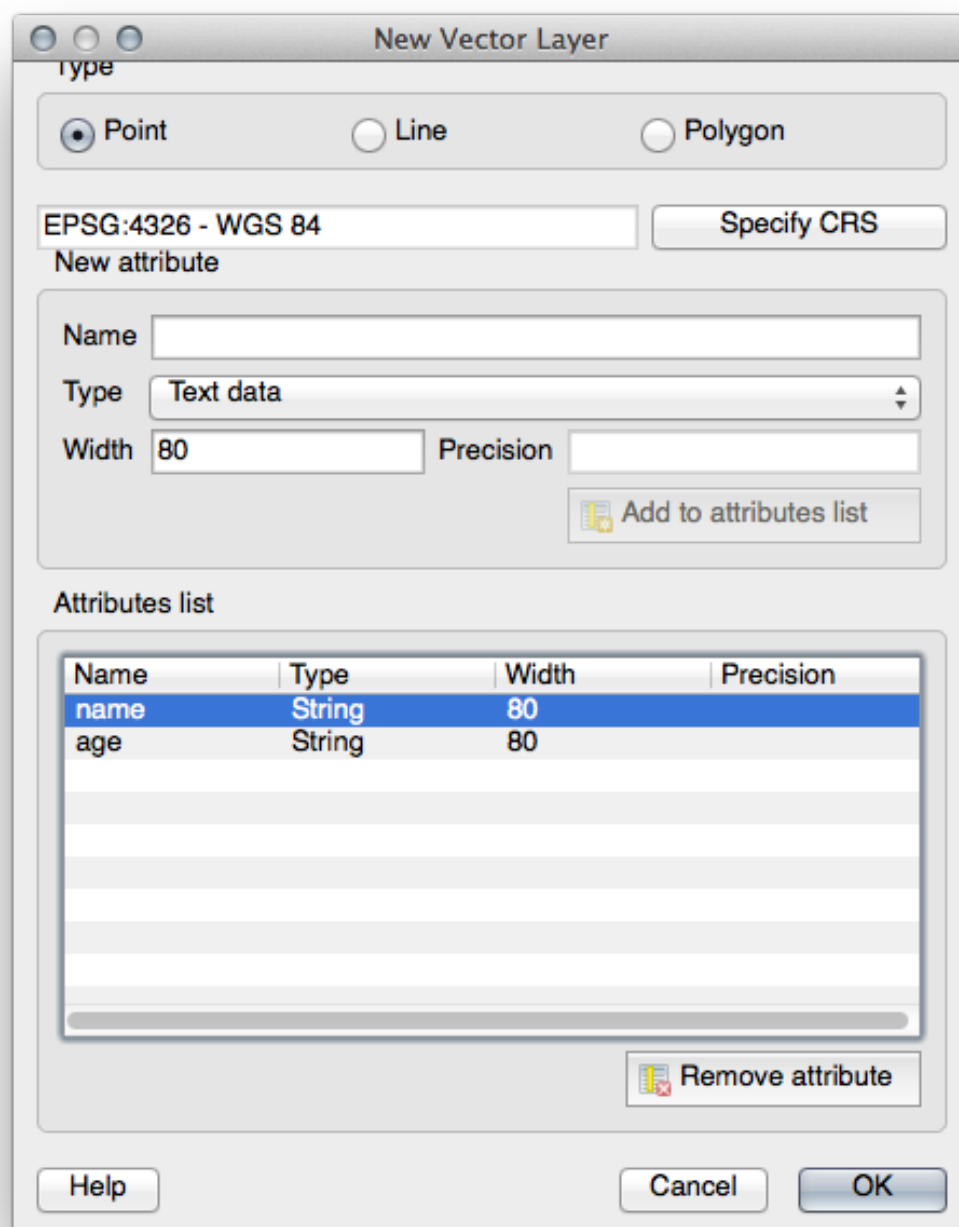
Set a more appropriate form widget for the *highway* field.

Check your results

6.3.5 Try Yourself Creating Test Data

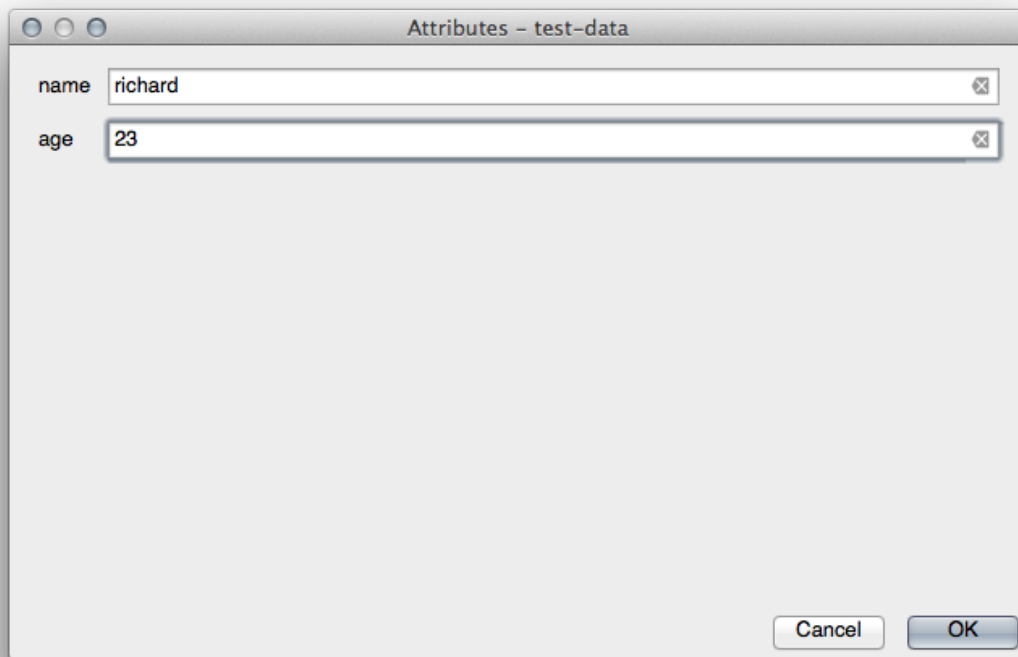
You can also design your own custom form completely from scratch.

- Create a simple point layer named `test-data` with two attributes:
 - Name (text)
 - Age (text)



- 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.

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



6.3.6 Follow Along: Creating a New Form

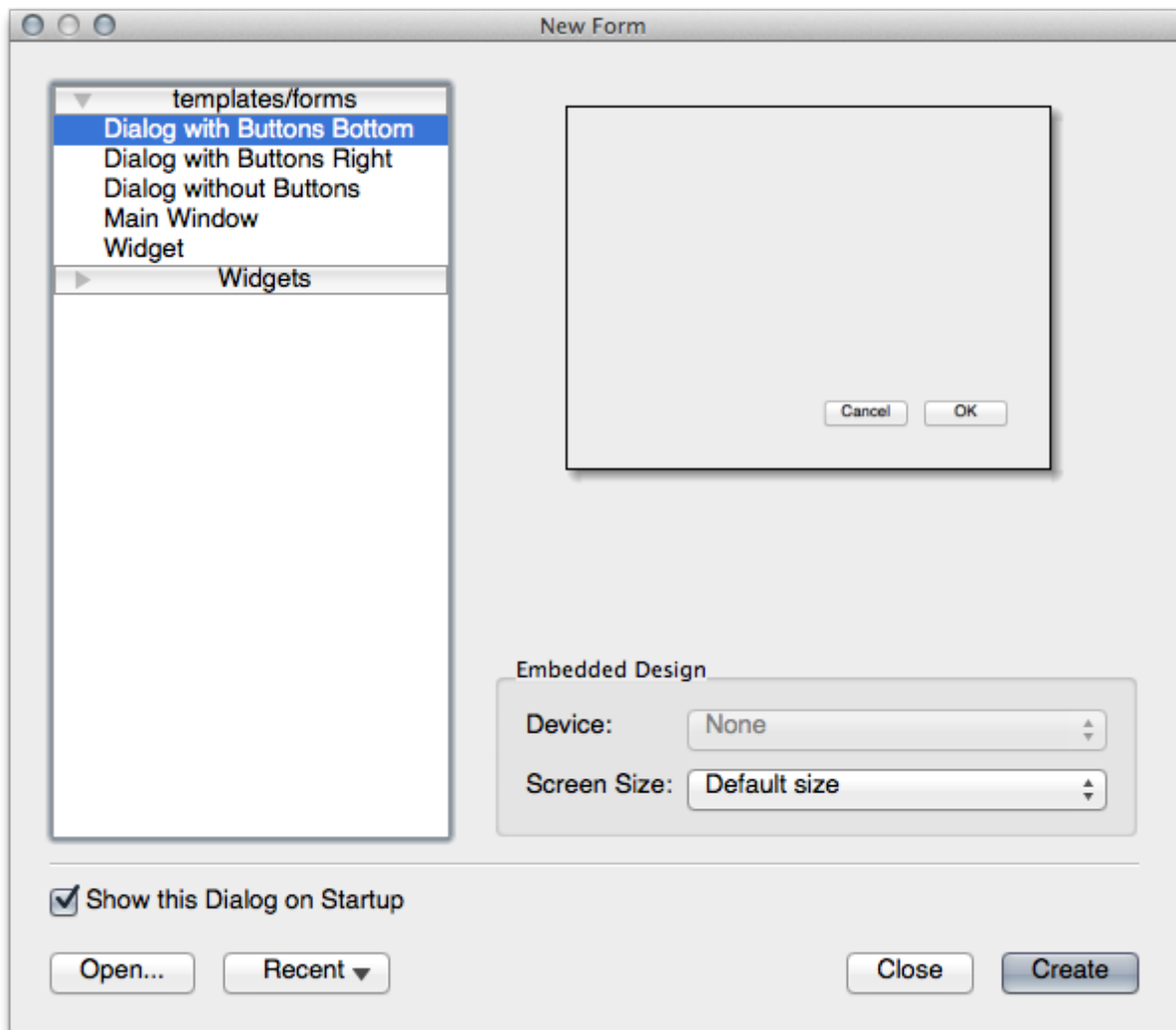
Now we want to create our own custom form for the attribute data capture phase. To do this, you need to have *Qt4 Designer* installed (only needed for the person who creates the forms). It should be provided as part of your course materials, if you're using Windows. You may need to look for it if you're using another OS. In Ubuntu, do the following in the terminal:

Bemerkung: At the time of writing, Qt5 is the latest version available. However, this process specifically requires Qt4 and is not necessarily compatible with Qt5.

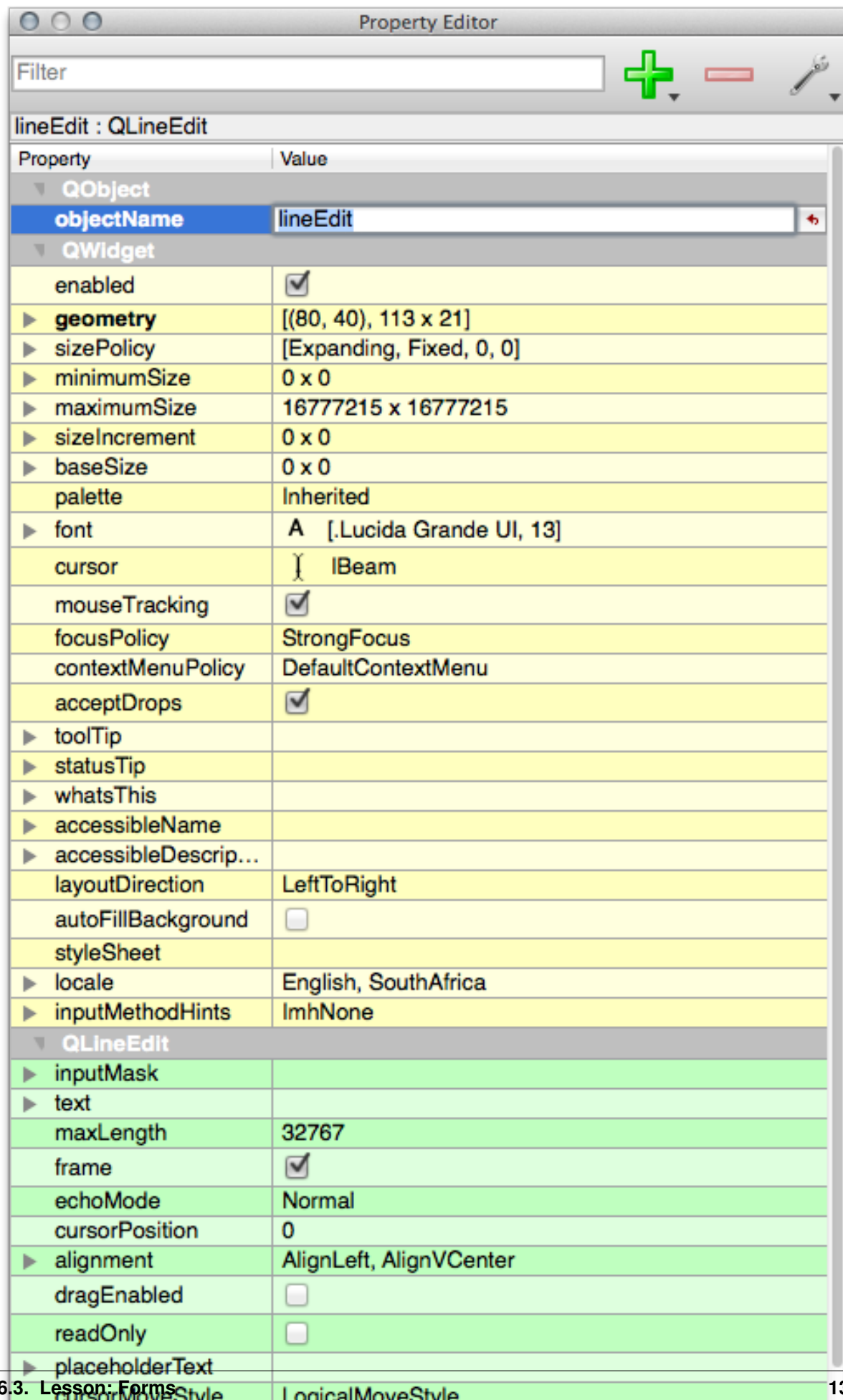
```
sudo apt-get install qt4-designer
```

... and it should install automatically. Otherwise, look for it in the *Software Center*.

- Start *Designer* by opening its *Start Menu* entry in Windows (or whatever approach is appropriate in your OS).
- In the dialog that appears, create a new dialog:



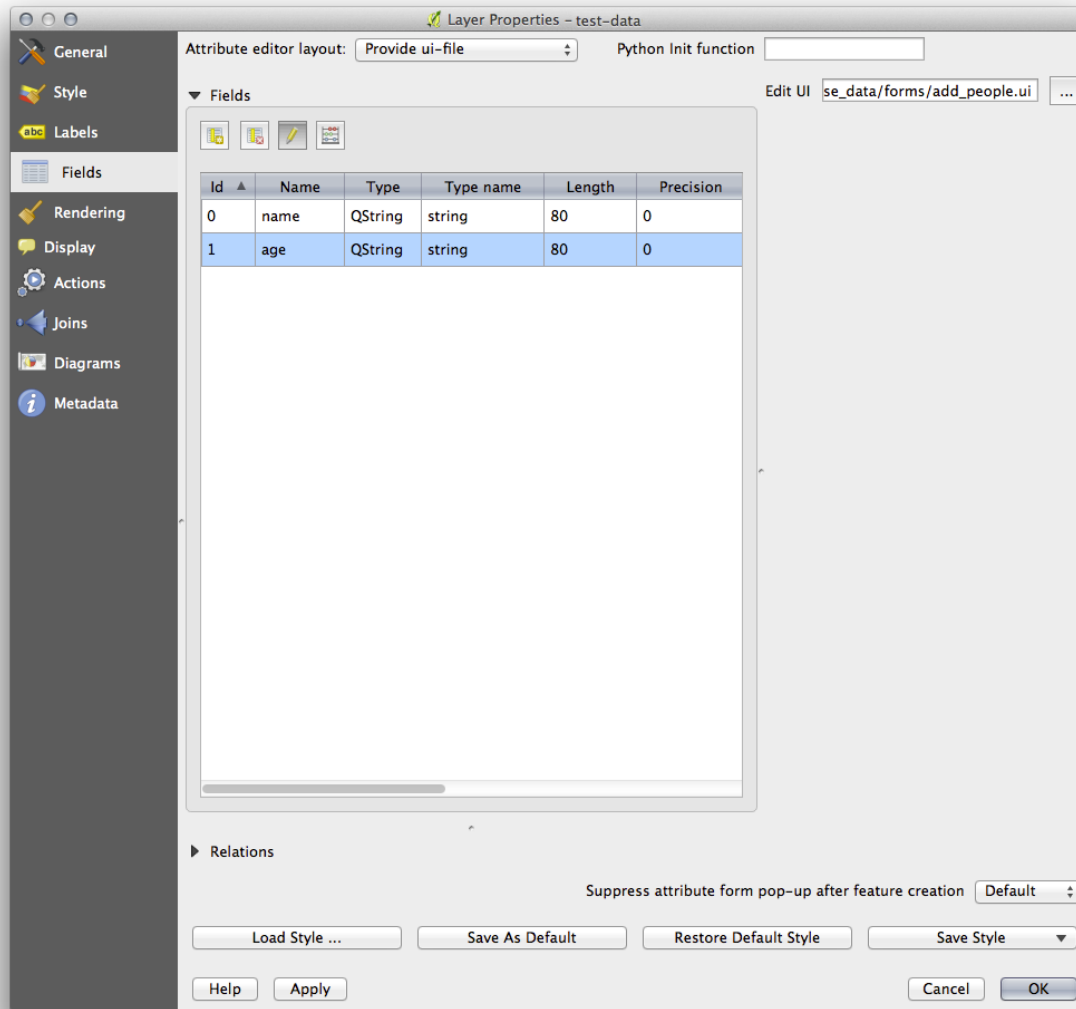
- Look for the *Widget Box* along the left of your screen (default). It contains an item called *Line Edit*.
- Click and drag this item into your form. This creates a new *Line Edit* in the form.
- With the new line edit element selected, you'll see its *properties* along the side of your screen (on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



- Set its name to `Name`.
- Using the same approach, create a new spinbox and set its name to `Age`.
- 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).
- Click anywhere in your dialog.
- Find the *Lay Out Vertically* button (in a toolbar along the top edge of the screen, by default). This lays out your dialog automatically.
- Set the dialog's maximum size (in its properties) to 200 (width) by 100 (height).
- Save your new form as `exercise_data/forms/add_people.ui`.
- When it's done saving, you can close the *Qt4 Designer* program.

6.3.7 Follow Along: Associating the Form with Your Layer

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



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

6.3.8 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.

6.3.9 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.

6.3.10 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.

6.4 Lesson: Actions

Now that you've seen a default action in the previous lesson, it's 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.

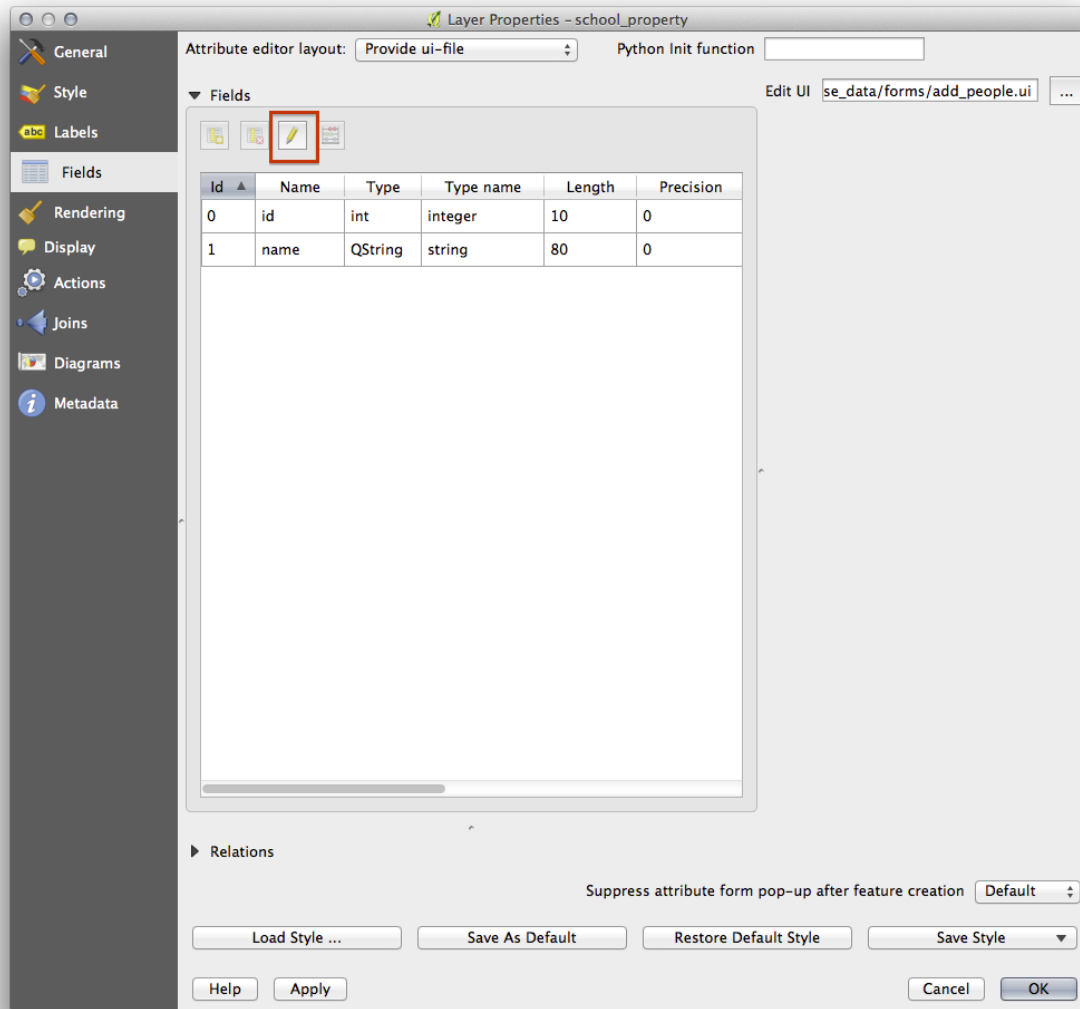
6.4.1 Follow Along: Open an Image

Use the *school_property* layer you created previously. The course materials include photos of each of the three properties you digitized. What we're going to do next is to associate each property with its image. Then we'll create an action that will open the image for a property when clicking on the property.

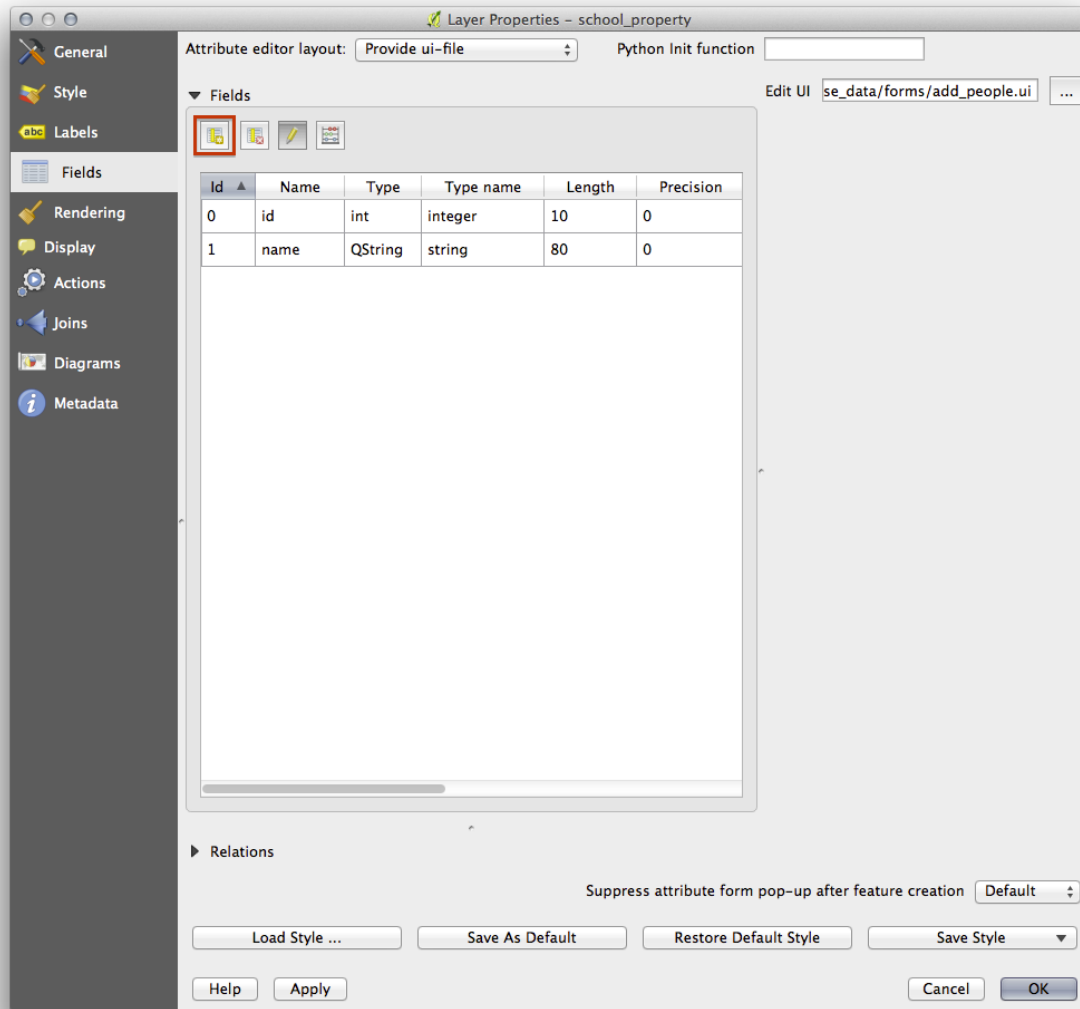
6.4.2 Follow Along: Add a Field for Images

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

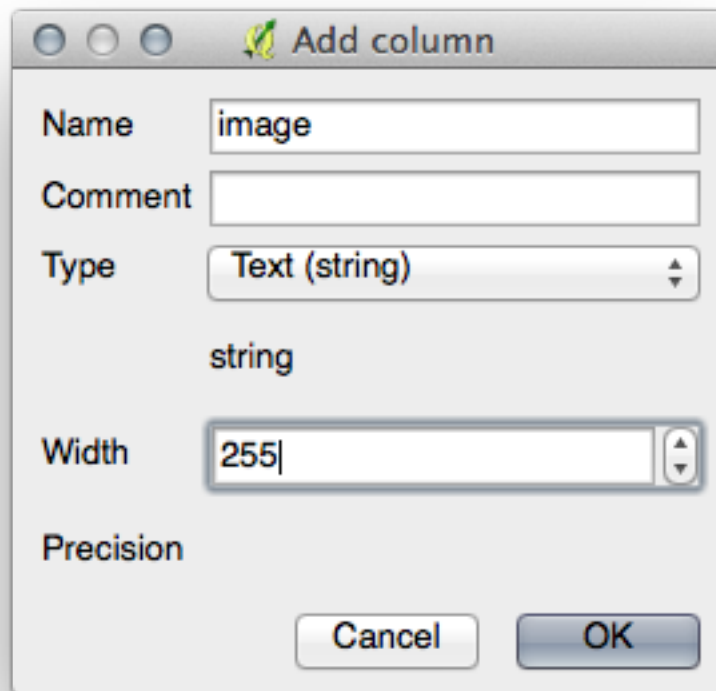
- Open the *Layer Properties* dialog.
- Click on the *Fields* tab.
- Toggle editing mode:



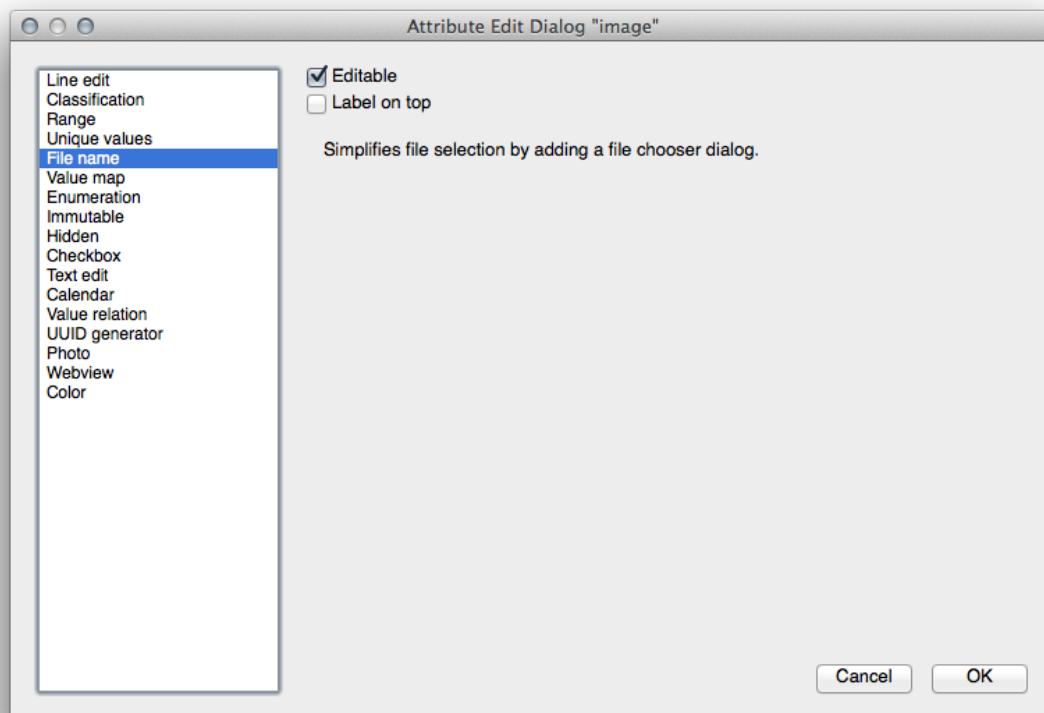
- Add a new column:



- Enter the values below:

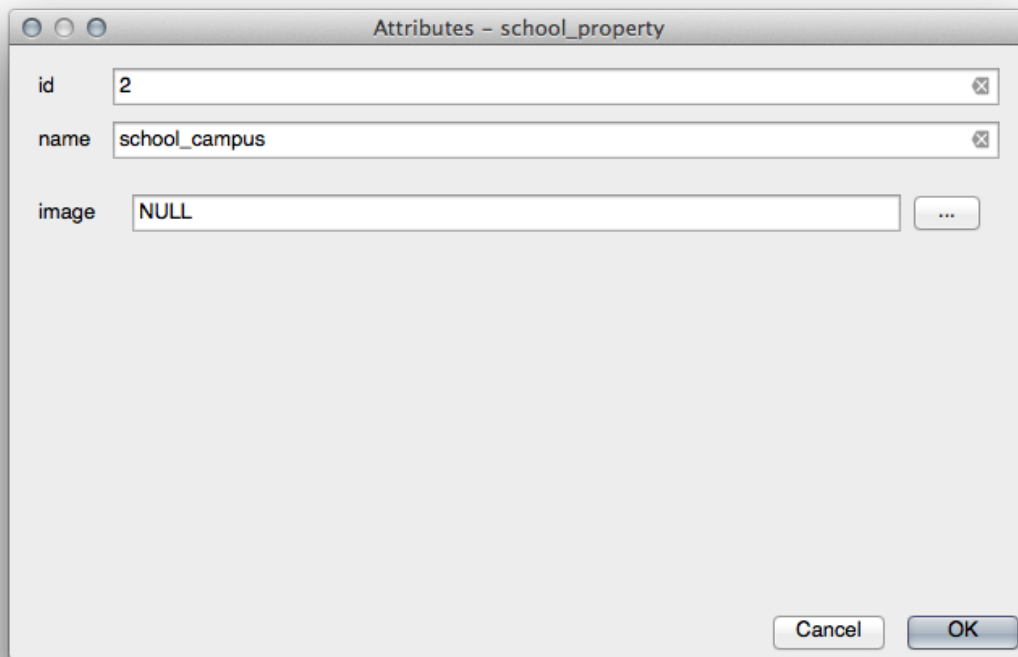


- After the field has been created, click on the *Line edit* button next to the new field.
- Set it up for a *File name*:



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

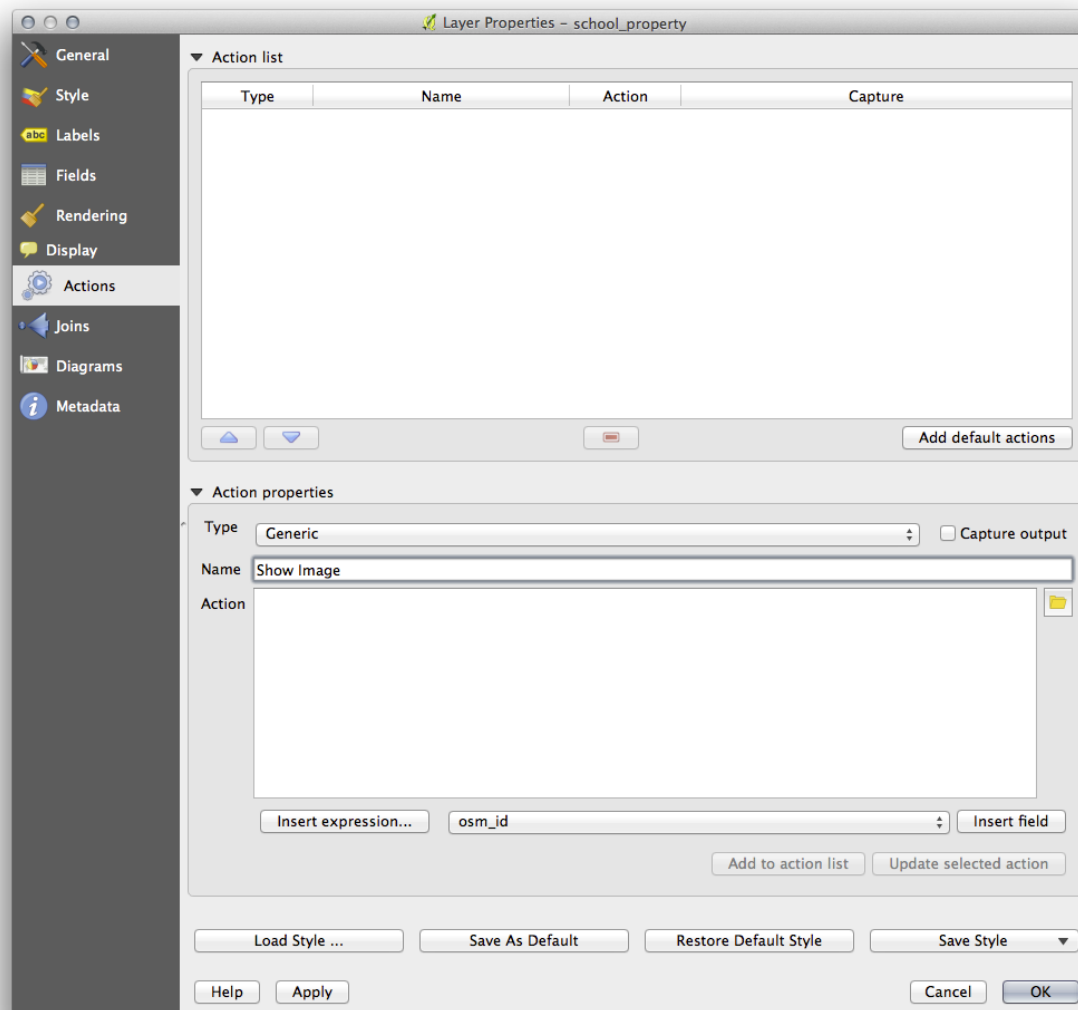
Since you're still in edit mode, the dialog should be active and look like this:



- Click on the browse button (the ... next to the *image* field).
- 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.
- Click *OK*.
- Associate all of the images with the correct features using this method.
- Save your edits and exit edit mode.

6.4.3 Follow Along: Creating an Action

- Open the *Actions* form for the *school_property* layer.
- In the *Action properties* panel, enter the words *Show Image* into the *Name* 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!

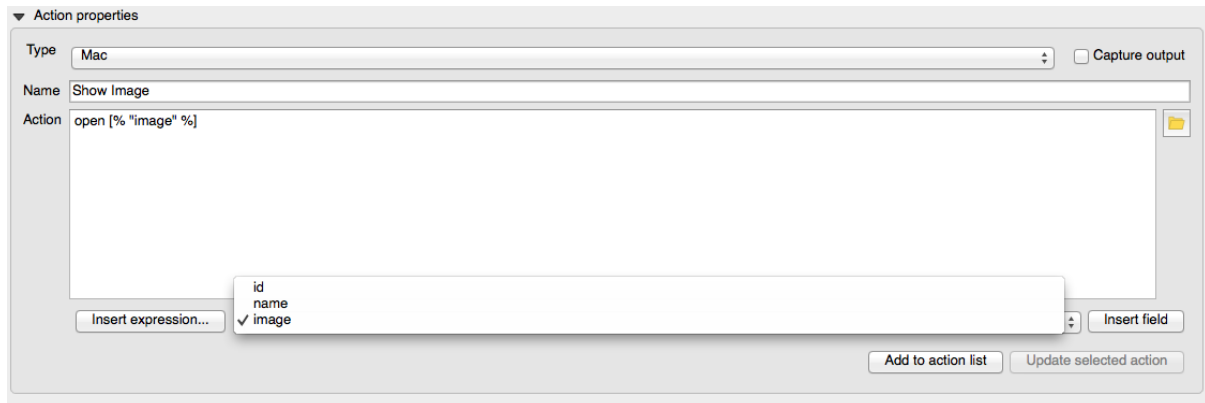
MacOS

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

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.

- Select *image* from the list:



- Click the *Insert field* button. QGIS will add the phrase [% "image" %] in the *Action* field.
- Click the *Add to action list* button.
- Click *OK* on the *Layer Properties* dialog.

Now we will test the new Action:

- Click on the *school_property* layer in the *Layers list* so that it is highlighted.
- Find the *Run feature action* button (on the same toolbar as the *Open Attribute Table* button):



- Click on the down arrow to the right of this button. There's only one action defined for this layer so far, which is the one you just created.
- Click the button itself to activate the tool.
- Using this tool, click on any of the three school properties.
- The image for that property will now open.

6.4.4 Follow Along: Searching the Internet

Let's say we're 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!

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

We'll be using the *name* field for each of our landuse areas to search Google.

- Close the attribute table.
- Go back to *Actions* in *Layer Properties*.
- In the field *Action Properties* → *Name*, write *Google Search*.

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.

Continue writing the command

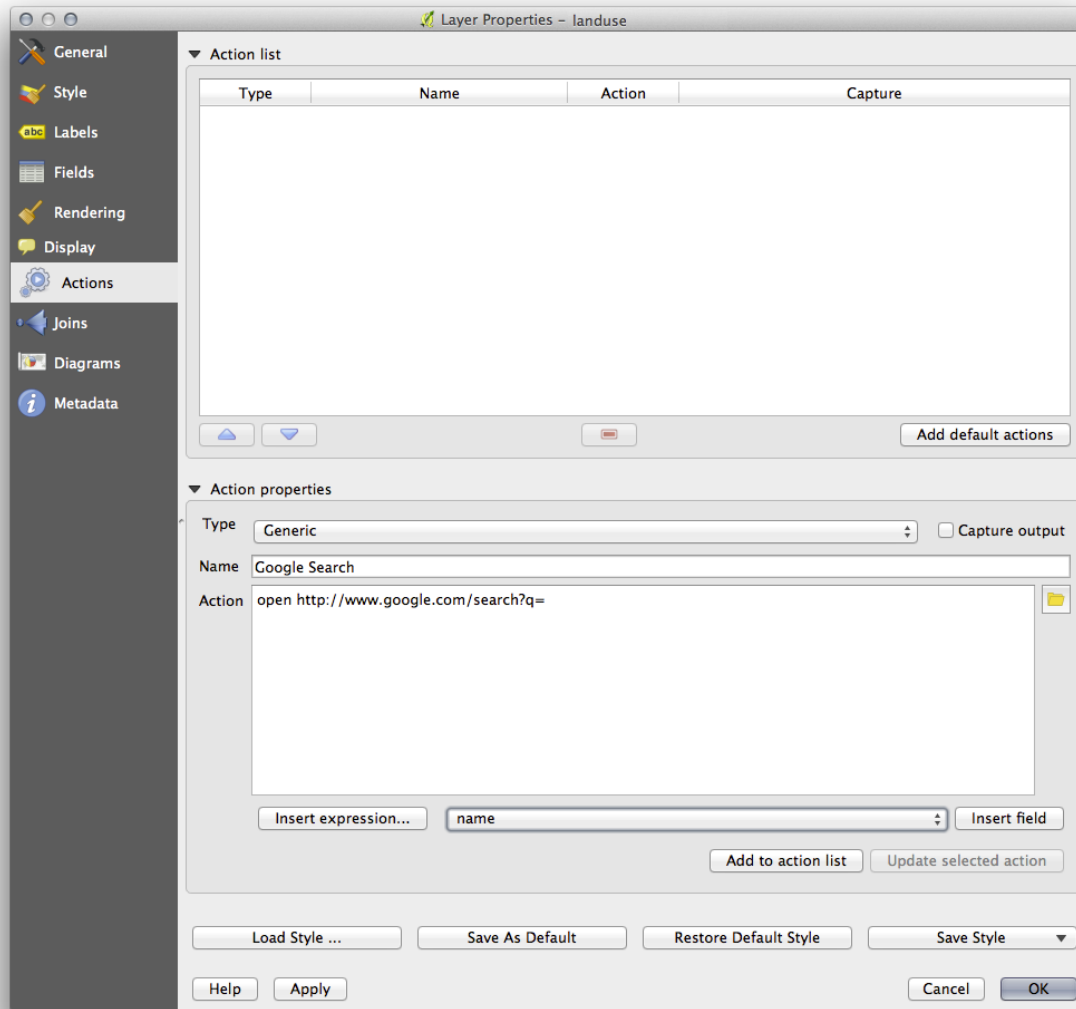
Whichever command you used above, you need to tell it which Internet address to open next. You want it to visit Google, and to search for a phrase automatically.

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 `http://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'll just enter the first part (without the search phrase).

- In the *Action* field, write `http://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.

- Select the *name* field.
- Click *Insert field*:



This will tell QGIS to add the phrase next:

| | |
|--------|--|
| Type | Mac |
| Name | Google Search |
| Action | open http://www.google.com/search?q=[% "name" %] |

What this means is that QGIS is going to open the browser and send it to the address `http://www.google.com/search?q=[% "name" %]`. But `[% "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, then QGIS is going to send the browser to `http://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”.

- If you haven't done so already, set everything up as explained above.
- Click the *Add to action list* button. The new action will appear in the list above.
- Click *OK* on the *Layer Properties* dialog.

Now to test the new action.

- With the *landuse* layer active in the *Layers list*, click on the *Run feature action* button.
- Click on any landuse area you can see on the map. Your browser will now open, and will automatically start a Google search for the town that is recorded as that area's *name* value.

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

6.4.5 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'll 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 Qt4 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'll see how to use a python action to show a web page. It's 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 Qt4 QWebView class (which is a webkit based html widget) to display the content in a pop up window.

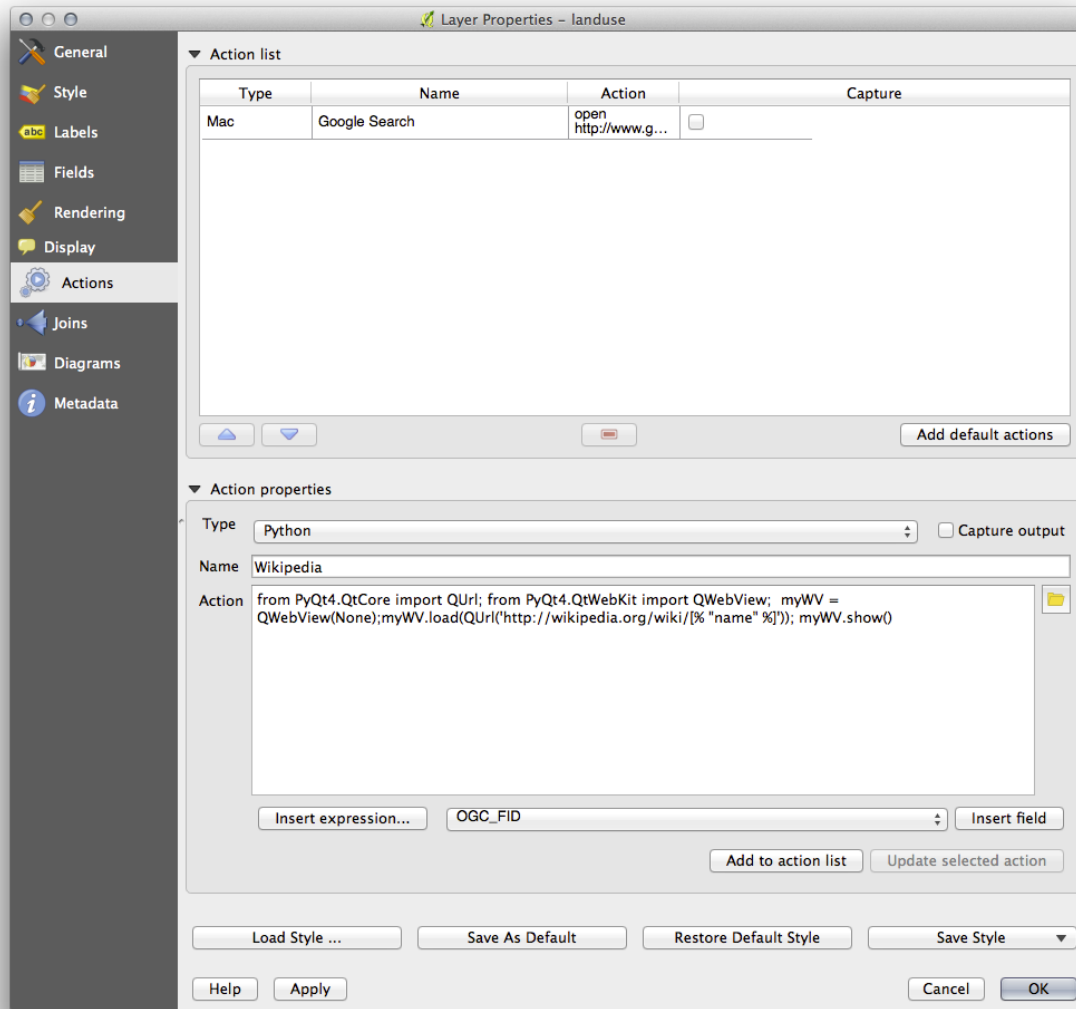
Instead of Google, let's use Wikipedia this time. So the URL you request will look like this:

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

To create the layer action:

- Open the *Layer Properties* dialog and head over to the *Actions* tab.
- Set up a new action using the following properties for the action: * *Type:* Python * *Name:* Wikipedia * *Action* (all on one line):

```
from PyQt4.QtCore import QUrl; from PyQt4.QtWebKit import
QWebView; myWV = QWebView(None); myWV.load(QUrl('http://wikipedia.org/wiki/[%
"name" %]')); myWV.show()
```



There are a couple of things going on here:

- All the python code is in a single line with semi-colons separating commands (instead of newlines, the usual way of separating Python commands).
- [% "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.

Note that this is a somewhat contrived example. Python works with semantically significant indentation, so separating things with semicolons isn't the best way to write it. So, in the real world, you'd be more likely to import your logic from a Python module and then call a function with a field attribute as parameter.

You could equally use the approach to display an image without requiring that the user has a particular image viewer on their system.

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

6.4.6 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 you can use shell commands for any operating system, as well as Python, the sky's the limit in terms of the functions you could incorporate!

6.4.7 What's Next?

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

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.

The key advantage of a GIS is this: *a GIS can answer questions.*

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.

7.1 Lesson: Reprojecting and Transforming Data

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

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

7.1.1 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.

- Save your current map.
- Then open the map of the world which you'll find under `exercise_data/world/world.qgs`.
- Zoom in to South Africa by using the *Zoom In* tool.
- Try setting a scale in the *Scale* field, which is in the *Status Bar* along the bottom of the screen. While over South Africa, set this value to 1 : 5000000 (one to five million).
- Pan around the map while keeping an eye on the *Scale* field.

Notice the scale changing? That's because you're moving away from the one point that you zoomed into at 1 : 5000000, 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're 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.

7.1.2 Follow Along: "On the Fly" Reprojection

QGIS allows you to reproject 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.

- To enable "on the fly" projection, click on the *CRS Status* button in the *Status Bar* along the bottom of the QGIS window:



- In the dialog that appears, check the box next to *Enable 'on the fly' CRS transformation*.
- Type the word `global` into the *Filter* field. One CRS (*NSIDC EASE-Grid Global*) should appear in the list below.
- Click on the *NSIDC EASE-Grid Global* to select it, then click *OK*.
- Notice how the shape of South Africa changes. All projections work by changing the apparent shapes of objects on Earth.
- Zoom in to a scale of 1 : 5000000 again, as before.
- 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.

- Deactivate "on the fly" re-projection again:
 - Click on the *CRS Status* button again.
 - Un-check the *Enable 'on the fly' CRS transformation* box.
 - Clicking *OK*.
- In QGIS 2.0, the 'on the fly' reprojection is automatically activated when

layers with different CRSs are loaded in the map. To understand what 'on the fly' reprojection does, deactivate this automatic setting:

- Go to *Settings* → *Options...*
- On the left panel of the dialog, select *CRS*.
- Un-check :guilabel:'Automatically enable 'on the fly' reprojection if layers have different CRS'. * Click *OK*.

- Add another vector layer to your map which has the data for South Africa only. You'll find it as `exercise_data/world/RSA.shp`.

What do you notice?

The layer isn't visible! But that's easy to fix, right?

- Right-click on the *RSA* layer in the *Layers list*.
- Select *Zoom to Layer Extent*.

OK, so now we see South Africa... but where is the rest of the world?

It turns out that we can zoom between these two layers, but we can't ever see them at the same time. That's because their Coordinate Reference Systems are so different. The *continents* dataset is in *degrees*, but the *RSA* dataset is in *meters*. So, let's say that a given point in Cape Town in the *RSA* dataset is about 4 100 000 meters away from the equator. But in the *continents* dataset, that same point is about 33 . 9 degrees away from the equator.

This is the same distance - but QGIS doesn't know that. You haven't told it to reproject the data. So as far as it's concerned, the version of South Africa that we see in the *RSA* dataset has Cape Town at the correct distance of 4 100 000 meters from the equator. But in the *continents* dataset, Cape Town is only 33 . 9 *meters* away from the equator! You can see why this is a problem.

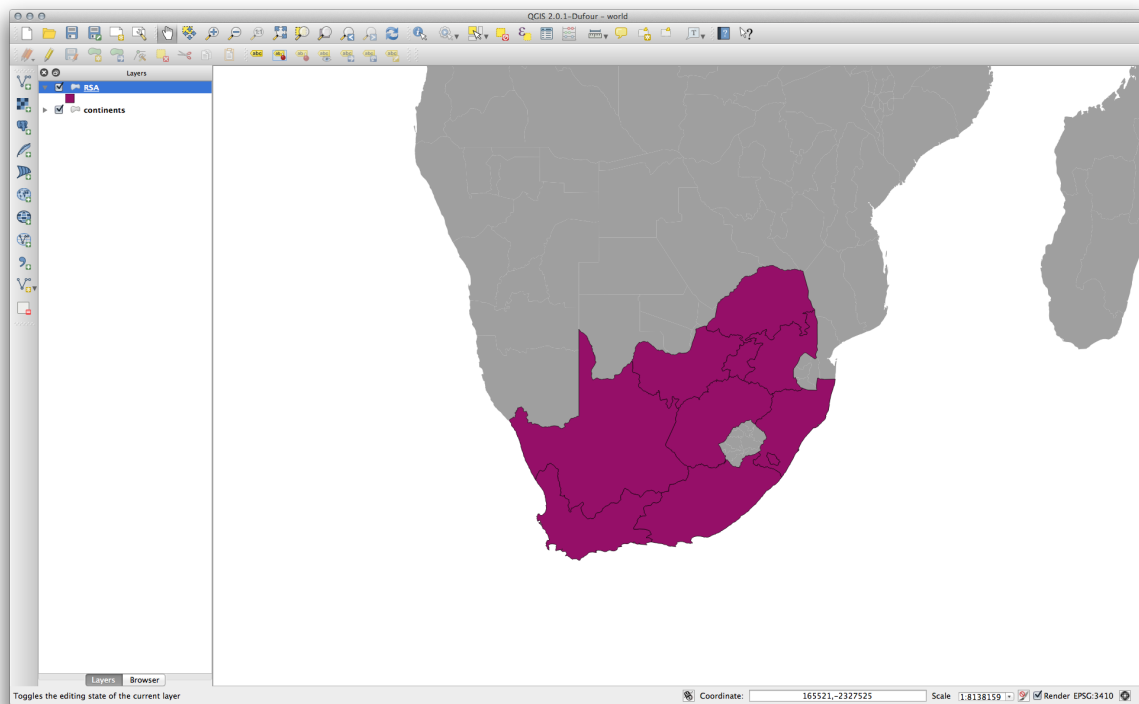
QGIS doesn't know where Cape Town is *supposed* to be - that's what the data should be telling it. If the data tells QGIS that Cape Town is 34 meters away from the equator and that South Africa is only about 12 meters from north to south, then that is what QGIS will draw.

To correct this:

- Click on the *CRS Status* button again and switch

Enable 'on the fly' CRS transformation on again as before. * Zoom to the extents of the *RSA* dataset.

Now, because they're made to project in the same CRS, the two datasets fit perfectly:



When combining data from different sources, it's important to remember that they might not be in the same CRS. "On the fly" reprojection helps you to display them together.

Before you go on, you probably want to have the 'on the fly' reprojection to be automatically activated whenever you open datasets having different CRS:

- Open again *Settings* → *Options...* and select *CRS*.
- Activate *Automatically enable 'on the fly' reprojection if layers have different CRS*.

7.1.3 Follow Along: Saving a Dataset to Another CRS

Remember when you calculated areas for the buildings in the *Classification* lesson? You did it so that you could classify the buildings according to area.

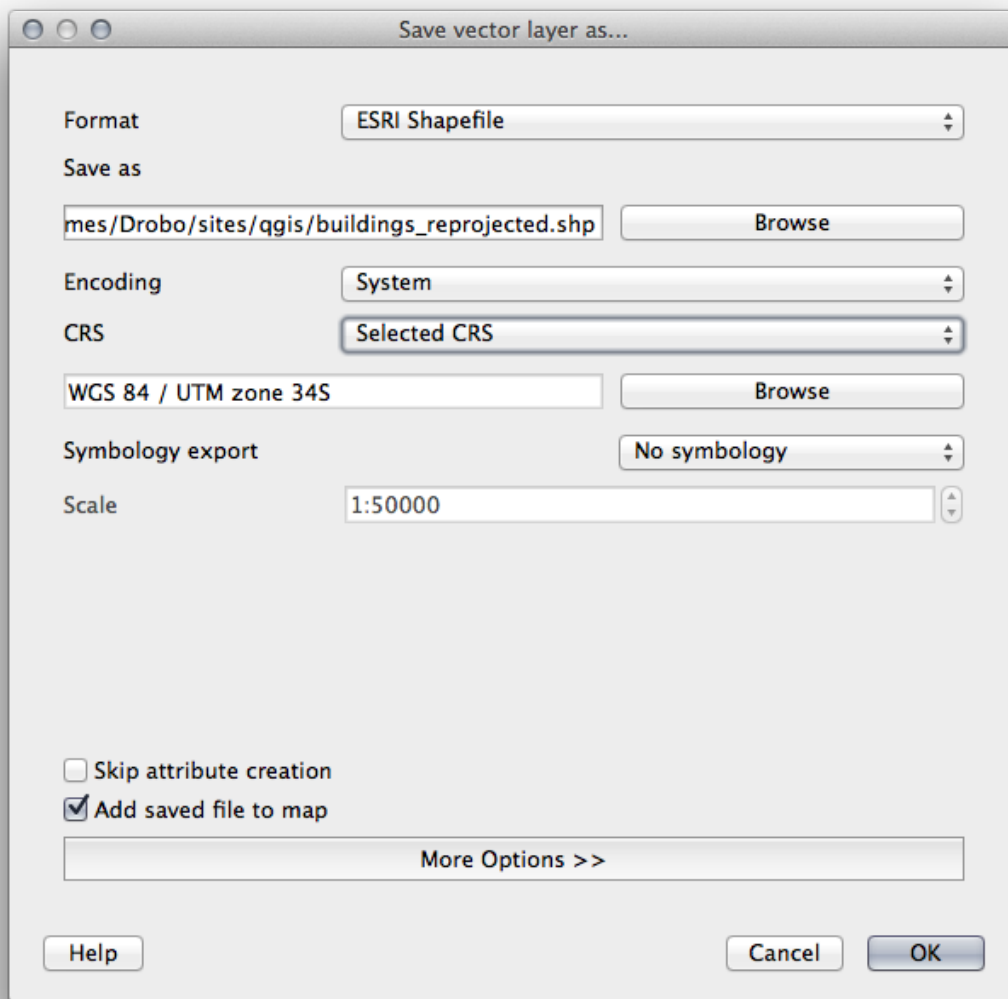
- Open your usual map again (containing the Swellendam data).
- Open the attribute table for the *buildings* layer.
- Scroll to the right until you see the *AREA* field.

Notice how the areas are all very small; probably zero. This is because these areas are given in degrees - the data isn't in a Projected Coordinate System. In order to calculate the area for the farms in square meters, the data has to be in square meters as well. So, we'll need to reproject it.

But it won't help to just use 'on the fly' reprojection. 'On the fly' does what it says - it doesn't change the data, it just reprojects the layers as they appear on the map. To truly reproject the data itself, you need to export it to a new file using a new projection.

- Right-click on the *buildings* layer in the *Layers list*.
- Select *Save As...* in the menu that appears. You will be shown the *Save vector layer as...* dialog.
- Click on the *Browse* button next to the *Save as* field.
- Navigate to `exercise_data/` and specify the name of the new layer as `buildings_reprojected.shp`.
- Leave the *Encoding* unchanged.
- Change the value of the *Layer CRS* dropdown to *Selected CRS*.
- Click the *Browse* button beneath the dropdown.
- The *CRS Selector* dialog will now appear.
- In its *Filter* field, search for *34S*.
- Choose *WGS 84 / UTM zone 34S* from the list.
- Leave the *Symbolology export* unchanged.

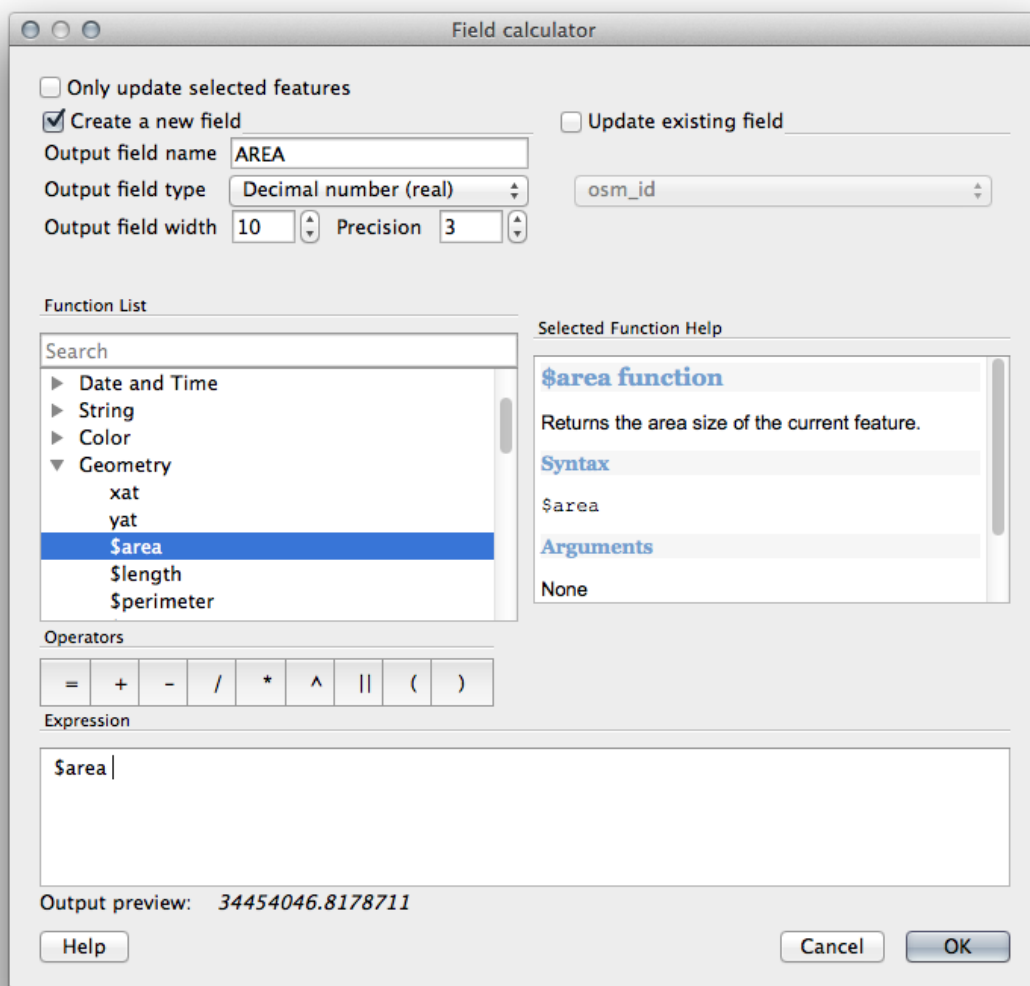
The *Save vector layer as...* dialog now looks like this:



- Click *OK*.
- Start a new map and load the reprojected layer you just created.

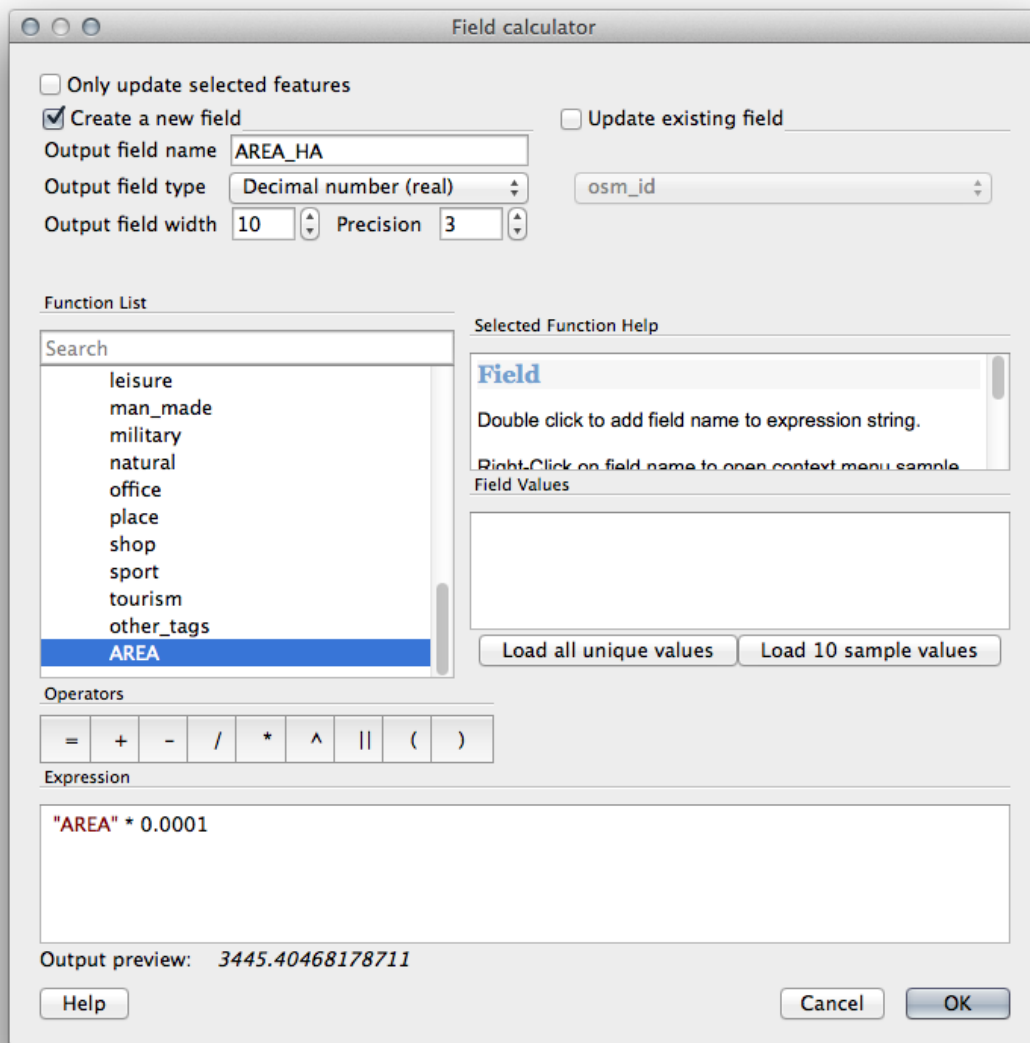
Refer back to the lesson on *Classification* to remember how you calculated areas.

- Update (or add) the AREA field by running the same expression as before:



This will add an AREA field with the size of each building in square meters

- To calculate the area in another unit of measurement, for example hectares, use the AREA field to create a second column:

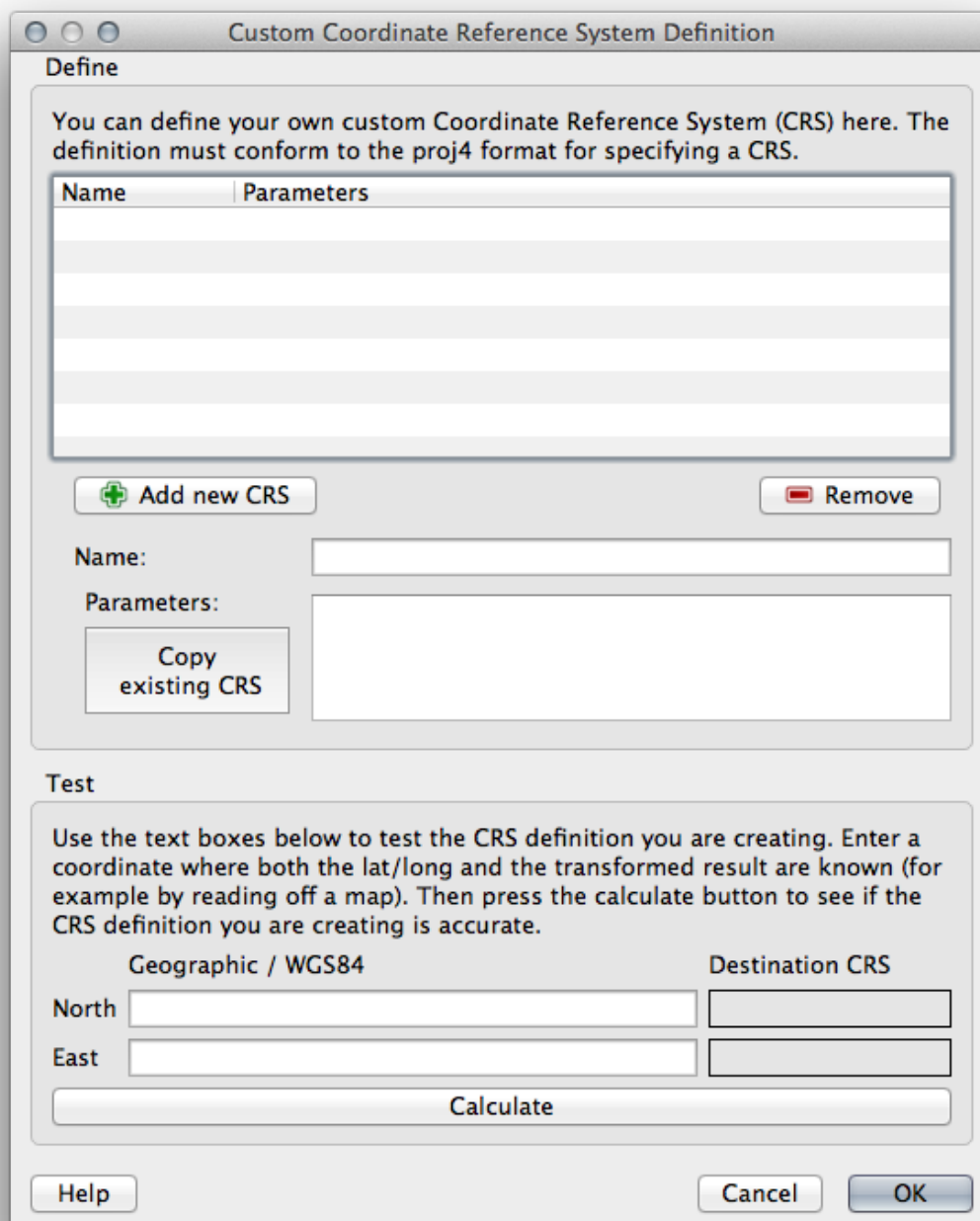


Look at the new values in your attribute table. This is much more useful, as people actually quote building size in metres, not in degrees. This is why it's a good idea to reproject your data, if necessary, before calculating areas, distances, and other values that are dependent on the spatial properties of the layer.

7.1.4 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.

- Start a new map.
- Load the `world/oceans.shp` dataset.
- Go to *Settings* → *Custom CRS...* and you'll see this dialog:



- Click on the *Add new CRS* button to create a new projection.

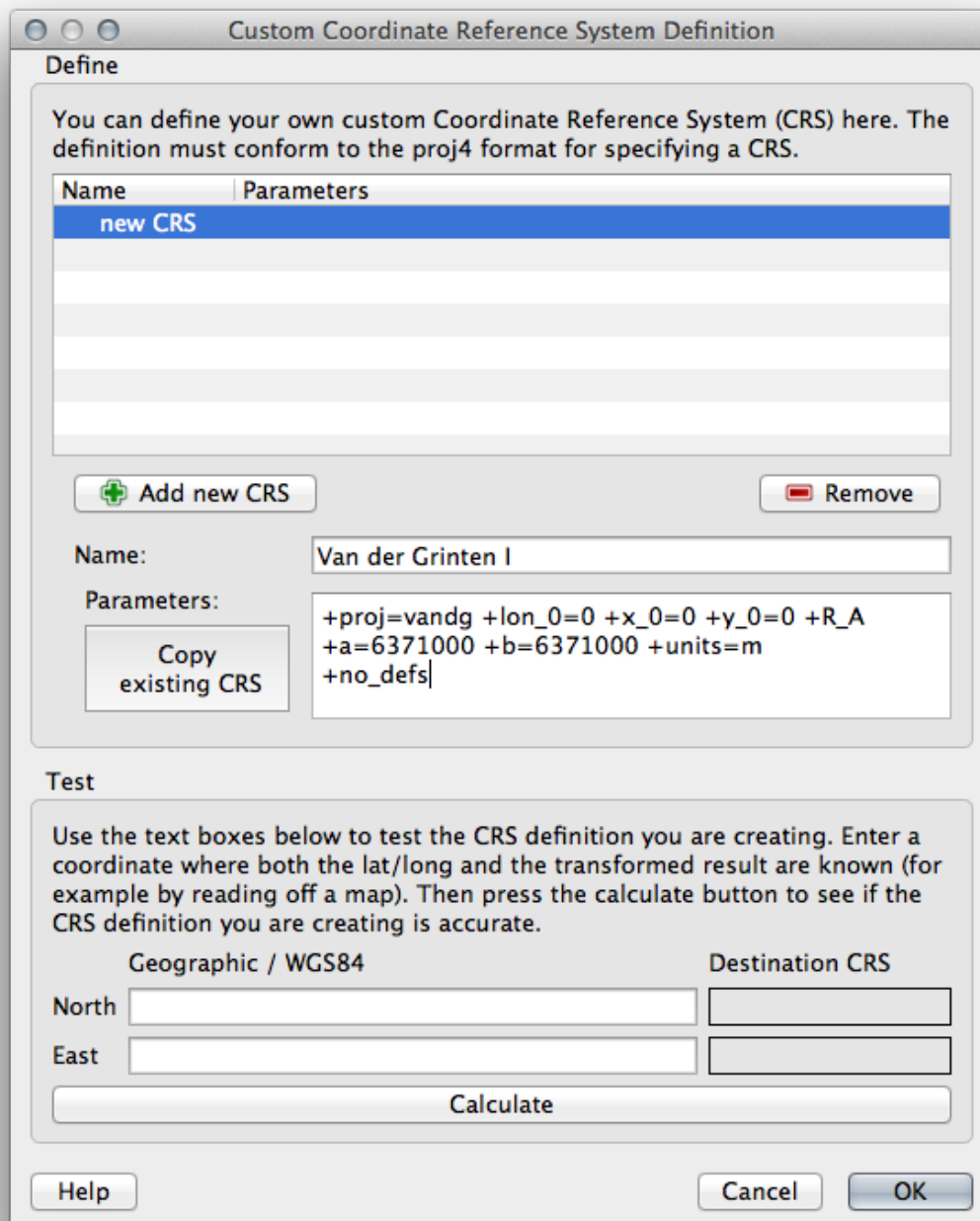
An interesting projection to use is called Van der Grinten I.

- Enter its name in the *Name* field.

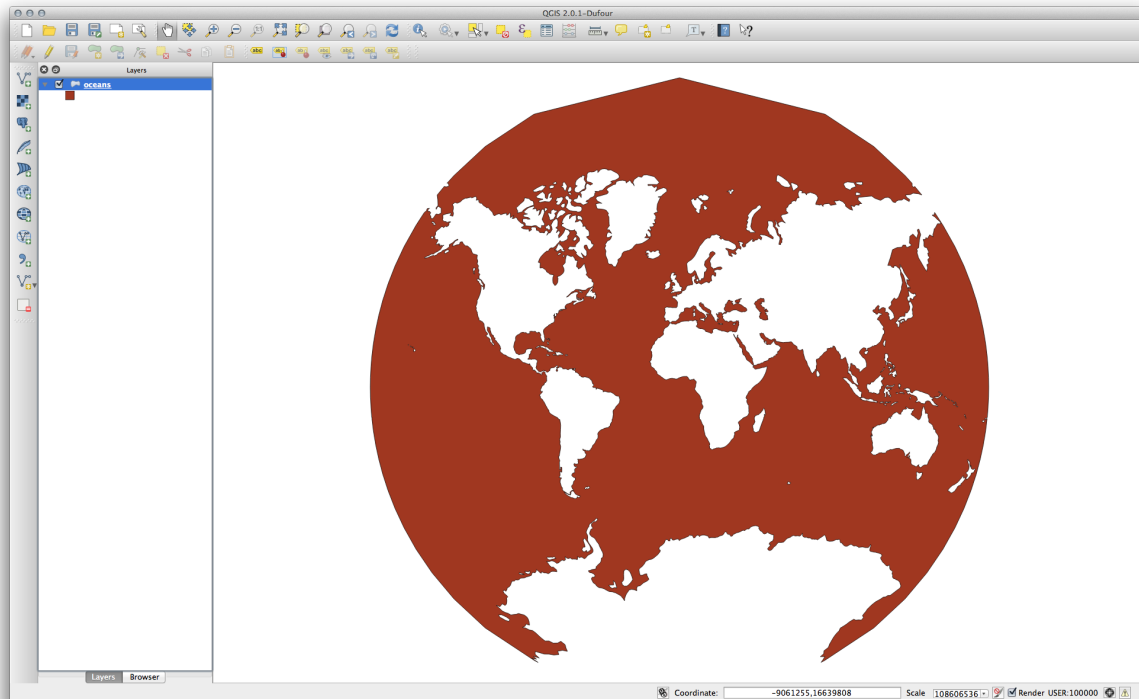
This projection represents the Earth on a circular field instead of a rectangular one, as most other projections do.

- For its parameters, use the following string:

```
+proj=vandg +lon_0=0 +x_0=0 +y_0=0 +R_A +a=6371000 +b=6371000 +units=m
+no_defs
```

- Click *OK*.
- Enable “on the fly” reprojection.
- Choose your newly defined projection (search for its name in the *Filter* field).
- On applying this projection, the map will be reprojected thus:



7.1.5 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.

7.1.6 Further Reading

Materials for the *Advanced* section of this lesson were taken from [this article](#).

Further information on Coordinate Reference Systems is available [here](#).

7.1.7 What's Next?

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

7.2 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 in GIS, so we won't go through them all. Rather, we'll 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.

7.2.1 The GIS Process

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

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

7.2.2 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.

7.2.3 The data

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

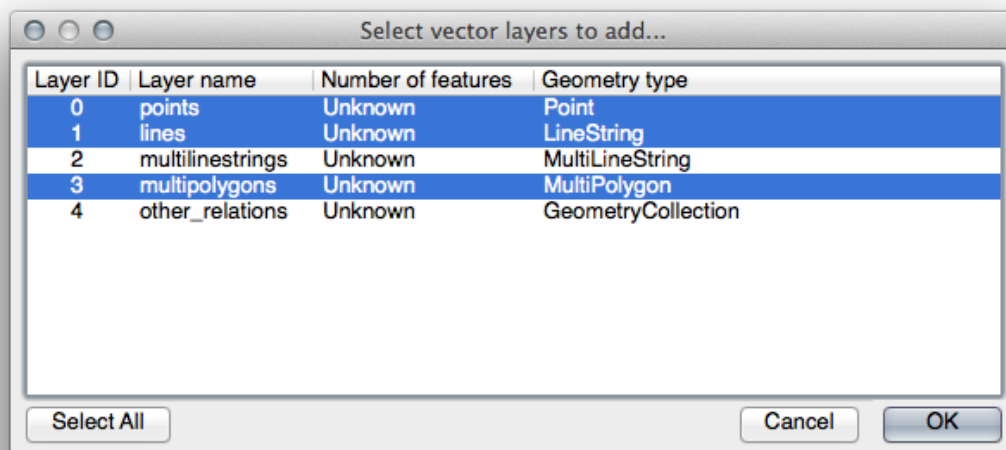
1. The residential properties (buildings) in the area.
2. The roads in and around the town.
3. The location of schools and restaurants.
4. The size of buildings.

All of this data is available through OSM and you should find that the dataset you have been using throughout this manual can also be used for this lesson. However, in order to ensure we have the complete data, we will re-download the data from OSM using QGIS' built-in OSM download tool.

Bemerkung: 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.

7.2.4 Follow Along: Start a Project

- Start a new QGIS project.
- Use the OpenStreetMap data download tool found in the *Vector -> OpenStreetMap* menu to download the data for your chosen region.
- Save the data as `osm_data.osm` in your `exercise_data` folder.
- Note that the *osm* format is a type of vector data. Add this data as a vector layer as usually *Layer -> Add vector layer...*, browse to the new `osm_data.osm` file you just downloaded. You may need to select *Show All Files* as the file format.
- Select `osm_data.osm` and click *Open*
- In the dialog which opens, select all the layers, *except* the `other_relations` and `multilinestrings` layer:



This will import the OSM data as separate layers into your map.

The data you just downloaded from OSM is in a geographic coordinate system, WGS84, which uses latitude and longitude coordinates, as you know from the previous lesson. You also learnt that to calculate distances in meters, we need to work with a projected coordinate system. Start by setting your project's coordinate system to a suitable CRS for your data, in the case of Swellendam, *WGS 84 / UTM zone 34S*:

- Open the **Project Properties** dialog, select *CRS* and filter the list to find *WGS 84 / UTM zone 34S*.
- Click **OK**.

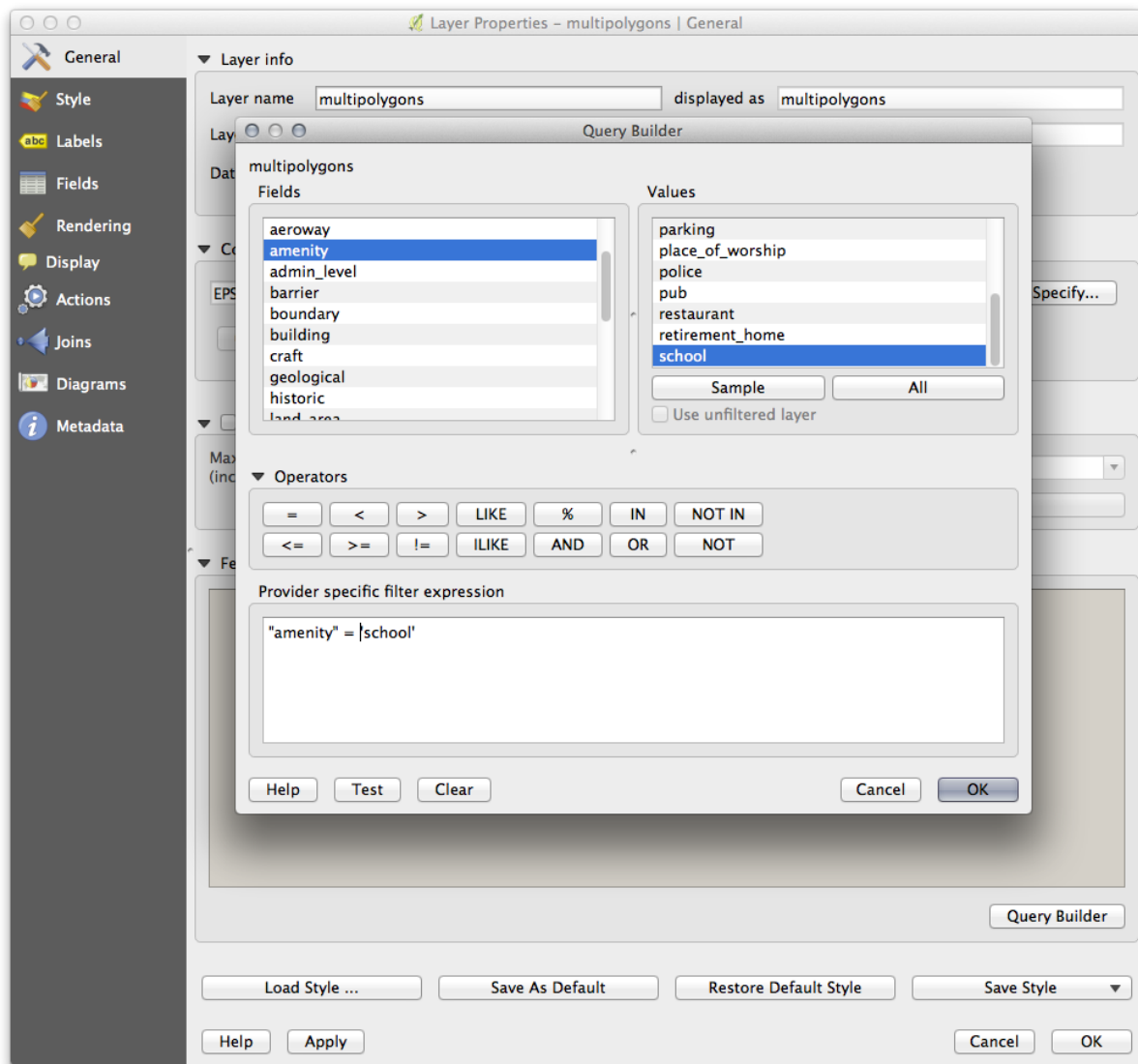
We now need to extract the information we need from the OSM dataset. We need to end up with layers representing all the houses, schools, restaurants and roads in the region. That information is inside the *multipolygons* layer and can be extracted using the information in its *Attribute Table*. We'll start with the *schools* layer:

- Right-click on the *multipolygons* layer in the *Layers list* and open the *Layer Properties*.
- Go to the *General* menu.
- Under *Feature subset* click on the **[Query Builder]** button to open the *Query builder* dialog.
- In the *Fields* list on the left of this dialog until you see the field *amenity*.
- Click on it once.
- Click the *All* button underneath the *Values* list:

Now we need to tell QGIS to only show us the polygons where the value of *amenity* is equal to *school*.

- Double-click the word *amenity* in the *Fields* list.
- Watch what happens in the *Provider specific filter expression*

field below:



The word "amenity" has appeared. To build the rest of the query:

- Click the = button (under *Operators*).
- Double-click the value school in the *Values* list.
- Click OK twice.

This will filter OSM's multipolygon layer to only show the schools in your region. You can now either:

- Rename the filtered OSM layer to schools and re-import the multipolygons layer from `osm_data.osm`, OR
- Duplicate the filtered layer, rename the copy, clear the Query Builder and create your new query in the *Query Builder*.

7.2.5 Try Yourself Extract Required Layers from OSM

Using the above technique, use the Query Builder tool to extract the remaining data from OSM to create the following layers:

- roads (from OSM's lines layer)
- restaurants (from OSM's multipolygons layer)

- houses (from OSM's multipolygons layer)

You may wish to re-use the `roads.shp` layer you created in earlier lessons.

Check your results

- Save your map under *exercise_data*, as *analysis.qgs*.
- In your operating system's file manager, create a new folder under *exercise_data* and call it *residential_development*. This is where you'll save the datasets that will be the results of the analysis functions.

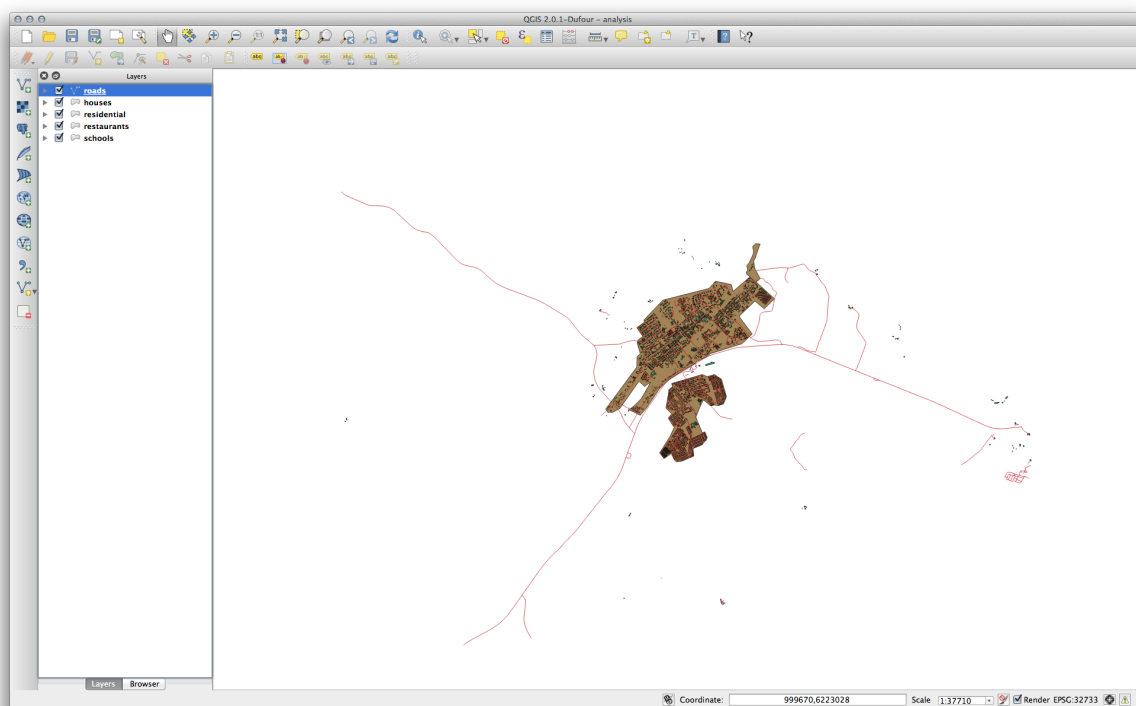
7.2.6 Try Yourself Find important roads

Some of the roads in OSM's dataset are listed as *unclassified*, *tracks*, *path* and *footway*. We want to exclude these from our roads dataset.

- Open the Query Builder for the *roads* layer, click *Clear* and build the following query:
`"highway" != 'NULL' AND "highway" != 'unclassified' AND "highway" != 'track' AND "highway" != 'path' AND "highway" != 'footway'`

You can either use the approach above, where you double-clicked values and clicked buttons, or you can copy and paste the command above.

This should immediately reduce the number of roads on your map:



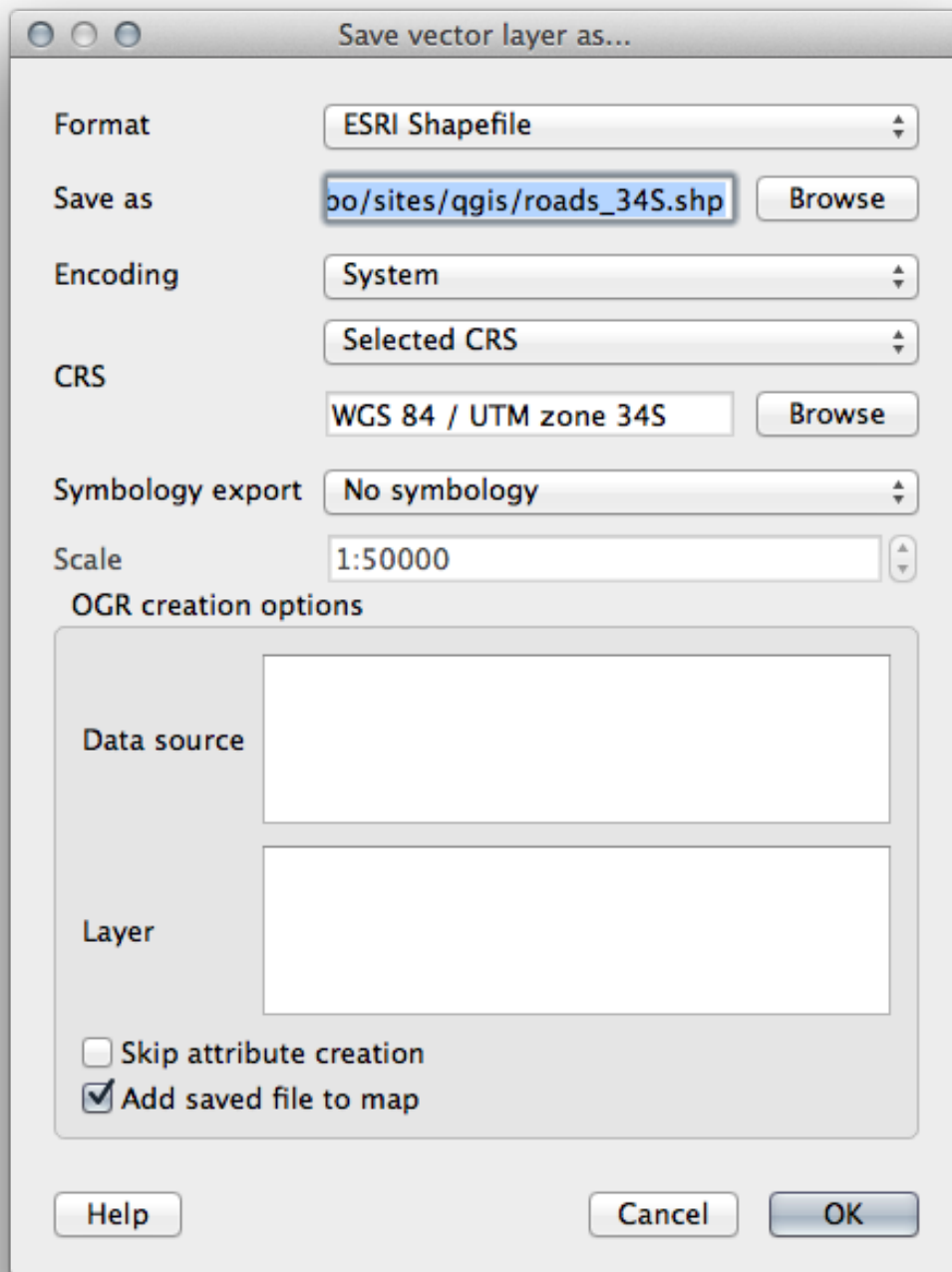
7.2.7 Try Yourself Convert Layers' CRS

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 shapefile with our new projection, then import that new layer into our map.

Bemerkung: In this example, we are using the *WGS 84 / UTM zone 34S* CRS, but you may use a UTM CRS

which is more appropriate for your region.

- Right click the roads layer in the Layers panel.
- Click Save as...
- In the Save Vector As dialog, choose the following settings and click *Ok* (making sure you select Add saved file to map):



The new shapefile will be created and the resulting layer added to your map.

Bemerkung: If you don't have activated *Enable 'on the fly' CRS transformation* or the *Automatically enable 'on the fly' reprojection if layers have different CRS* settings (see previous lesson), you might not be able to see the new layers you just added to the map. In this case, you can focus the map on any of the layers by right click on any layer and click *Zoom to layer extent*, or just enable any of the mentioned 'on the fly' options.

- Remove the old `roads` layer.

Repeat this process for each layer, creating a new shapefile and layer with “_34S” appended to the original name and removing each of the old layers.

Once you have completed the process for each layer, 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's data to a UTM projection, we can begin our calculations.

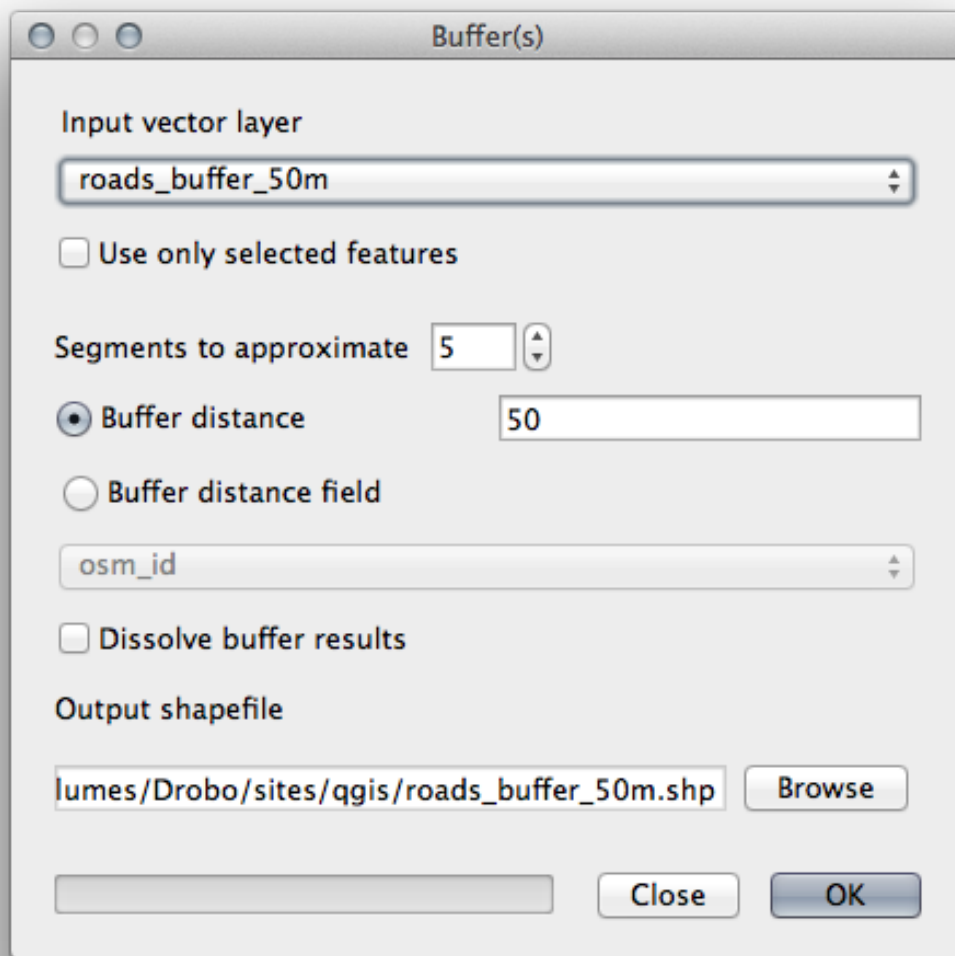
7.2.8 Follow Along: Analyzing the Problem: Distances From Schools and Roads

QGIS allows you to calculate distances from any vector object.

- Make sure that only the `roads_34S` and `houses_34S` layers are visible, to simplify the map while you're working.
- Click on the *Vector* → *Geoprocessing Tools* → *Buffer(s)* tool:

This gives you a new dialog.

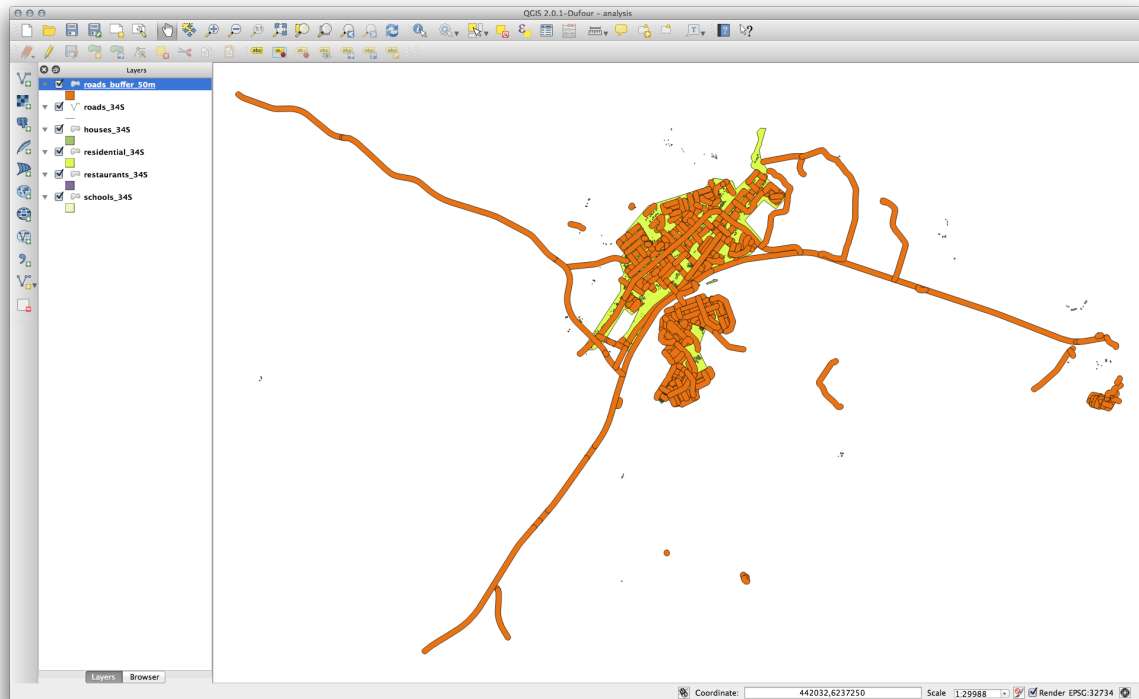
- Set it up like this:



The *Buffer distance* is in meters because our input dataset is in a Projected Coordinate System that uses meter as its basic measurement unit. This is why we needed to use projected data.

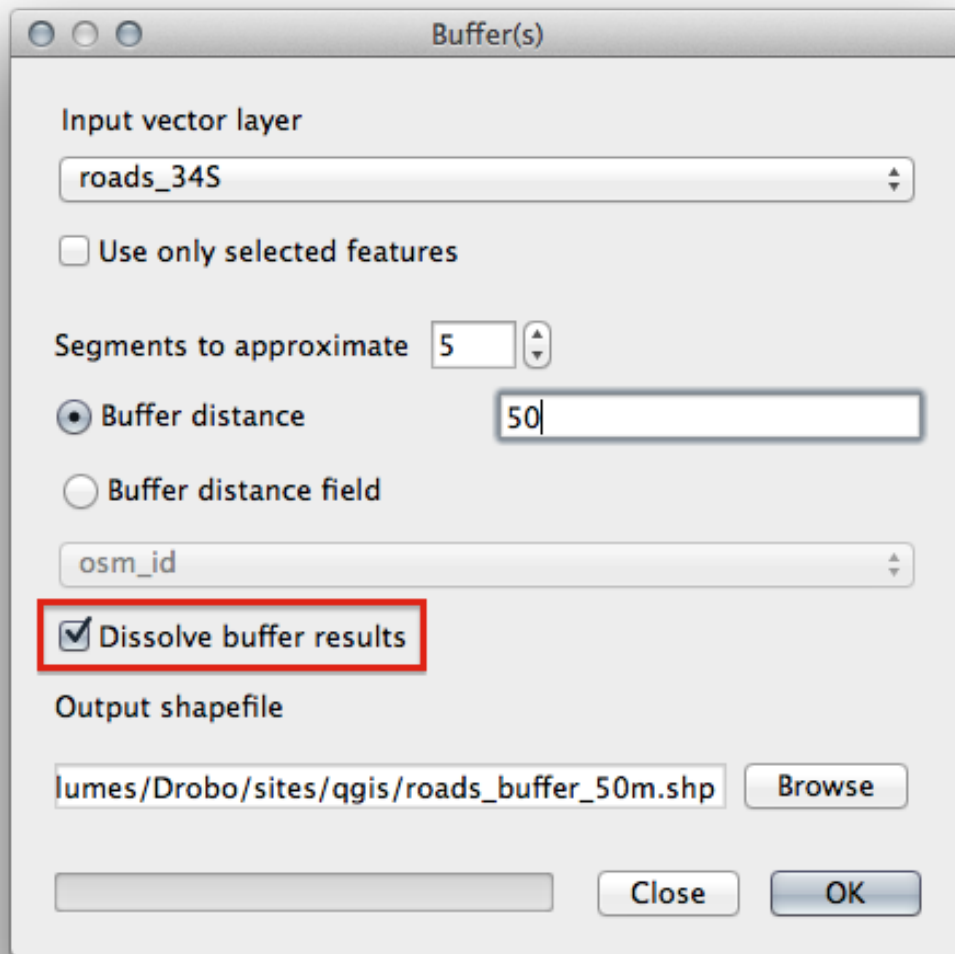
- Save the resulting layer under `exercise_data/residential_development/` as `roads_buffer_50m.shp`.
- Click *OK* and it will create the buffer.
- When it asks you if it should “add the new layer to the TOC”, click *Yes*. (“TOC” stands for “Table of Contents”, by which it means the *Layers list*).
- Close the *Buffer(s)* dialog.

Now your map will look something like this:



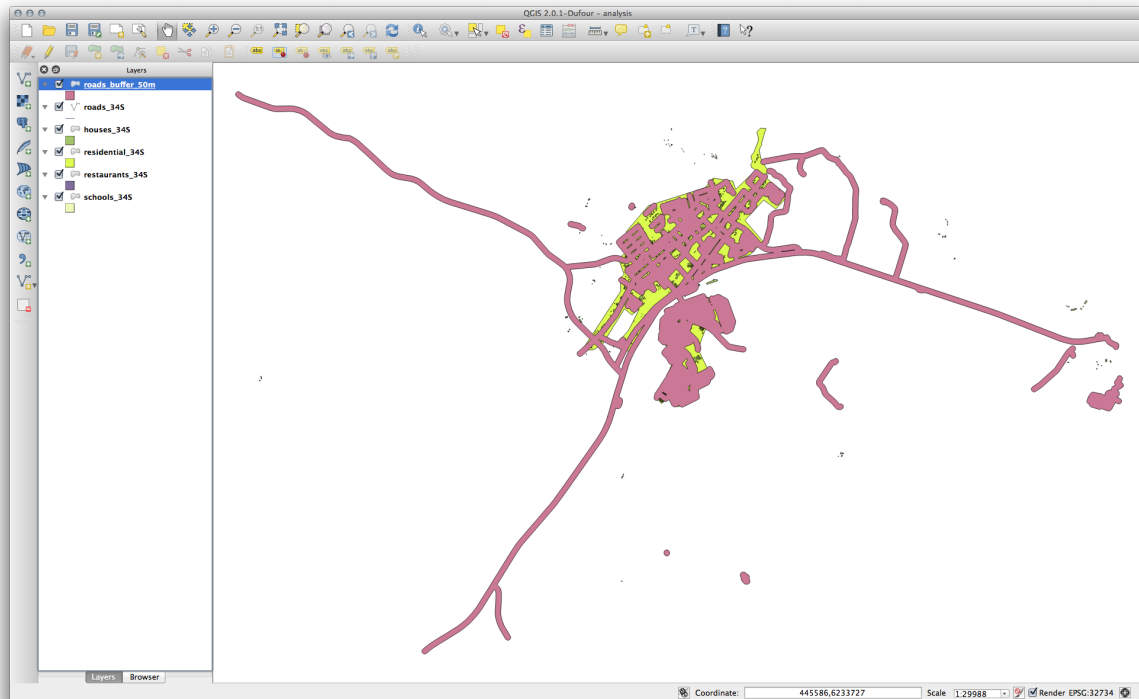
If your new layer is at the top of the `Layers` list, it will probably obscure much of your map, but this gives us all the areas in your region which are within 50m of a road.

However, you'll notice that there are distinct areas within our buffer, which correspond to all the individual roads. To get rid of this problem, remove the layer and re-create the buffer using the settings shown here:



- Note that we're now checking the *Dissolve buffer results* box.
- Save the output under the same name as before (click *Yes* when it asks your permission to overwrite the old one).
- Click *OK* and close the *Buffer(s)* dialog again.

Once you've added the layer to the *Layers list*, it will look like this:



Now there are no unnecessary subdivisions.

7.2.9 Try Yourself Distance from schools

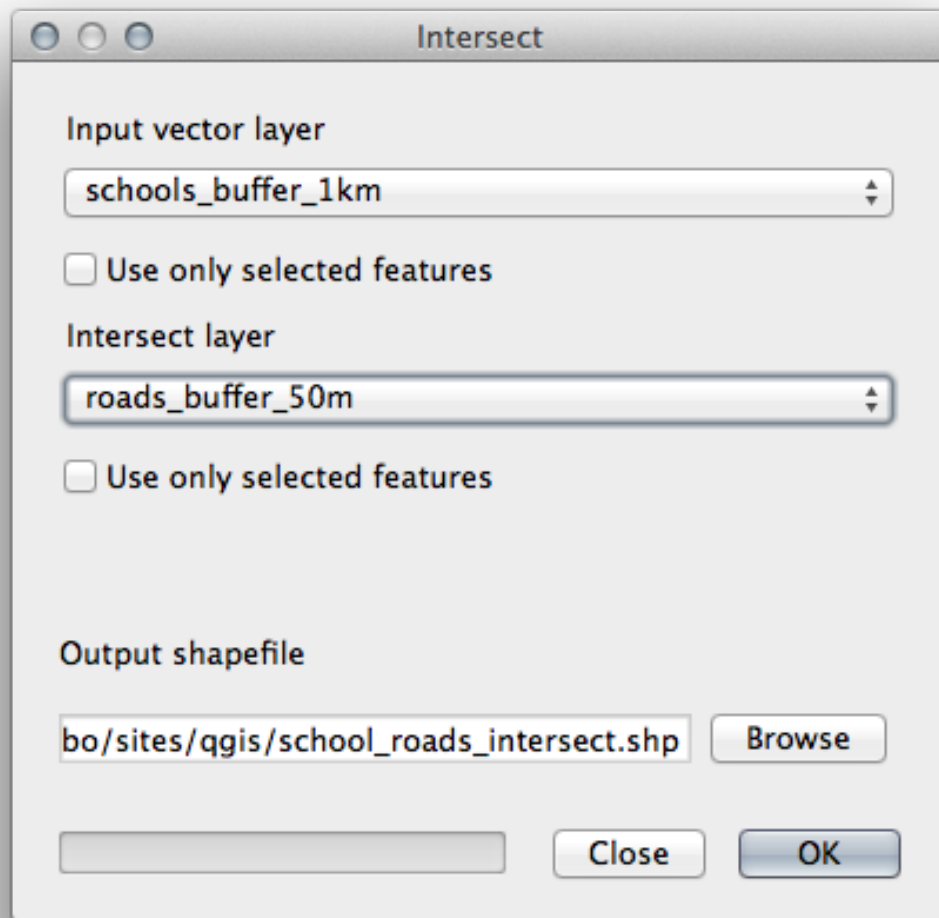
- Use the same approach as above and create a buffer for your schools.

It needs to be 1 km in radius, and saved under the usual directory as `schools_buffer_1km.shp`.

Check your results

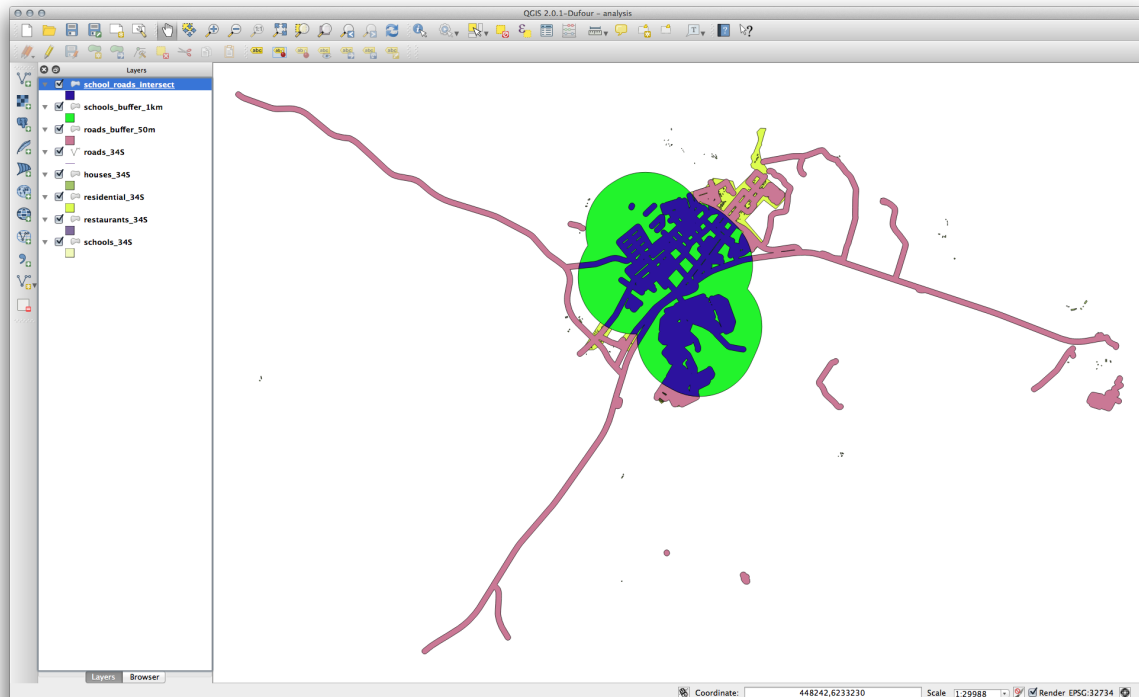
7.2.10 Follow Along: Overlapping Areas

Now we have areas where the road is 50 meters away and there's 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'll need to use the *Intersect* tool. Find it under *Vector* → *Geoprocessing Tools* → *Intersect*. Set it up like this:

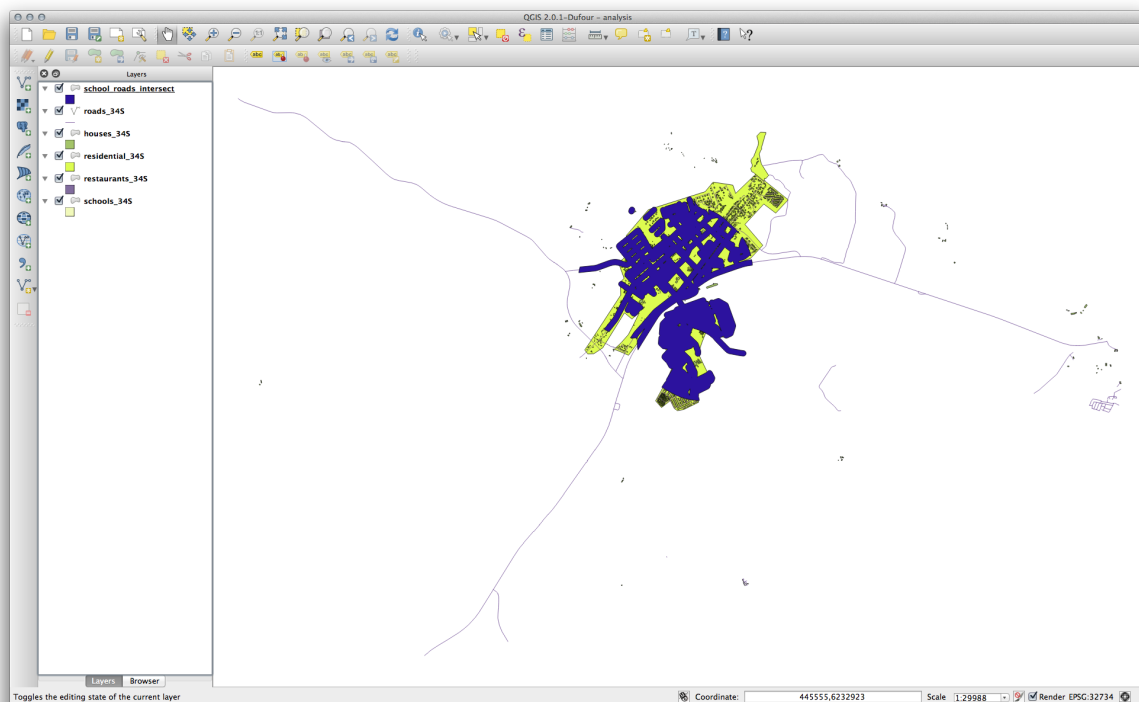


The two input layers are the two buffers; the save location is as usual; and the file name is `road_school_buffers_intersect.shp`. Once it's set up like this, click *OK* and add the layer to the *Layers list* when prompted.

In the image below, the blue areas show us where both distance criteria are satisfied at once!



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:

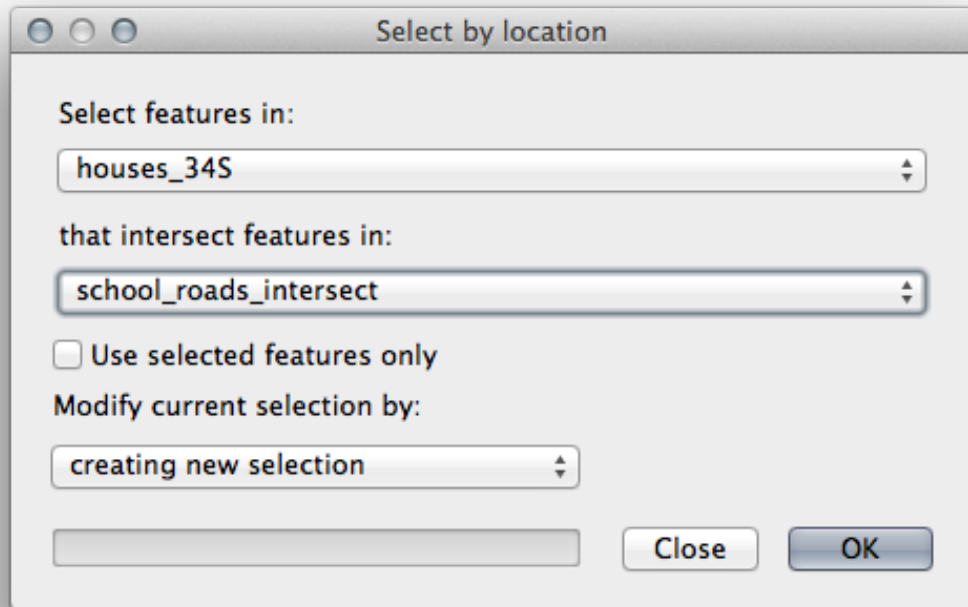


7.2.11 Follow Along: Select the Buildings

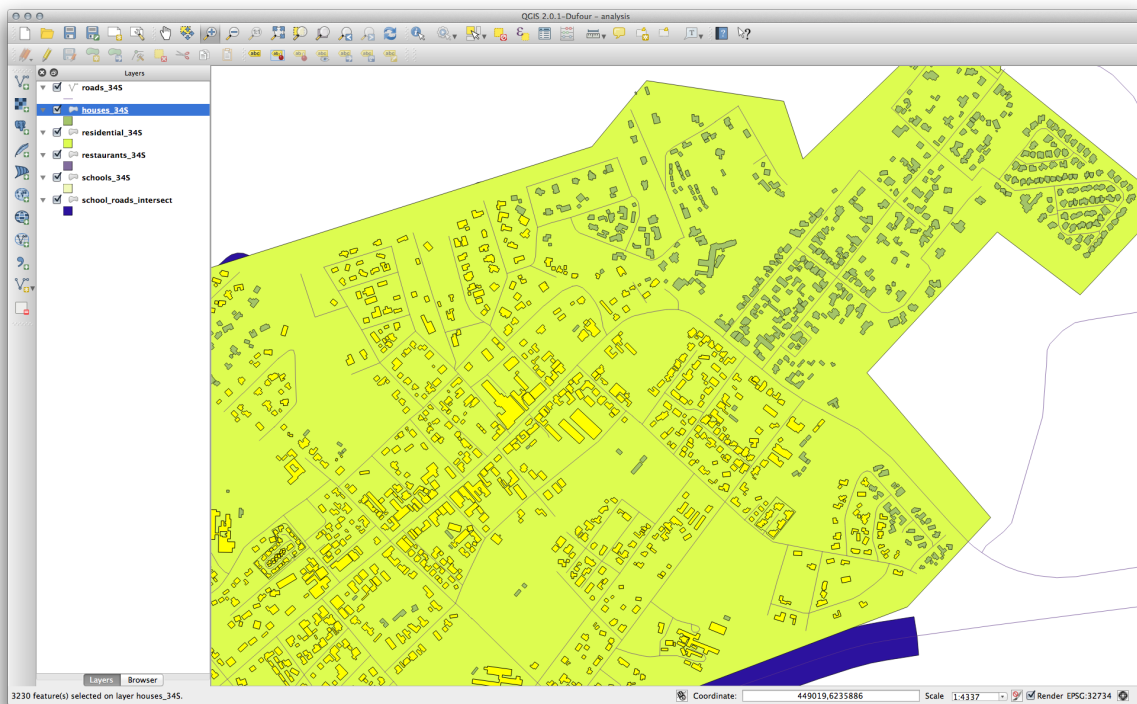
Now you've got the area that the buildings must overlap. Next, you want to select the buildings in that area.

- Click on the menu entry *Vector* → *Research Tools* → *Select by location*. A dialog will appear.

- Set it up like this:

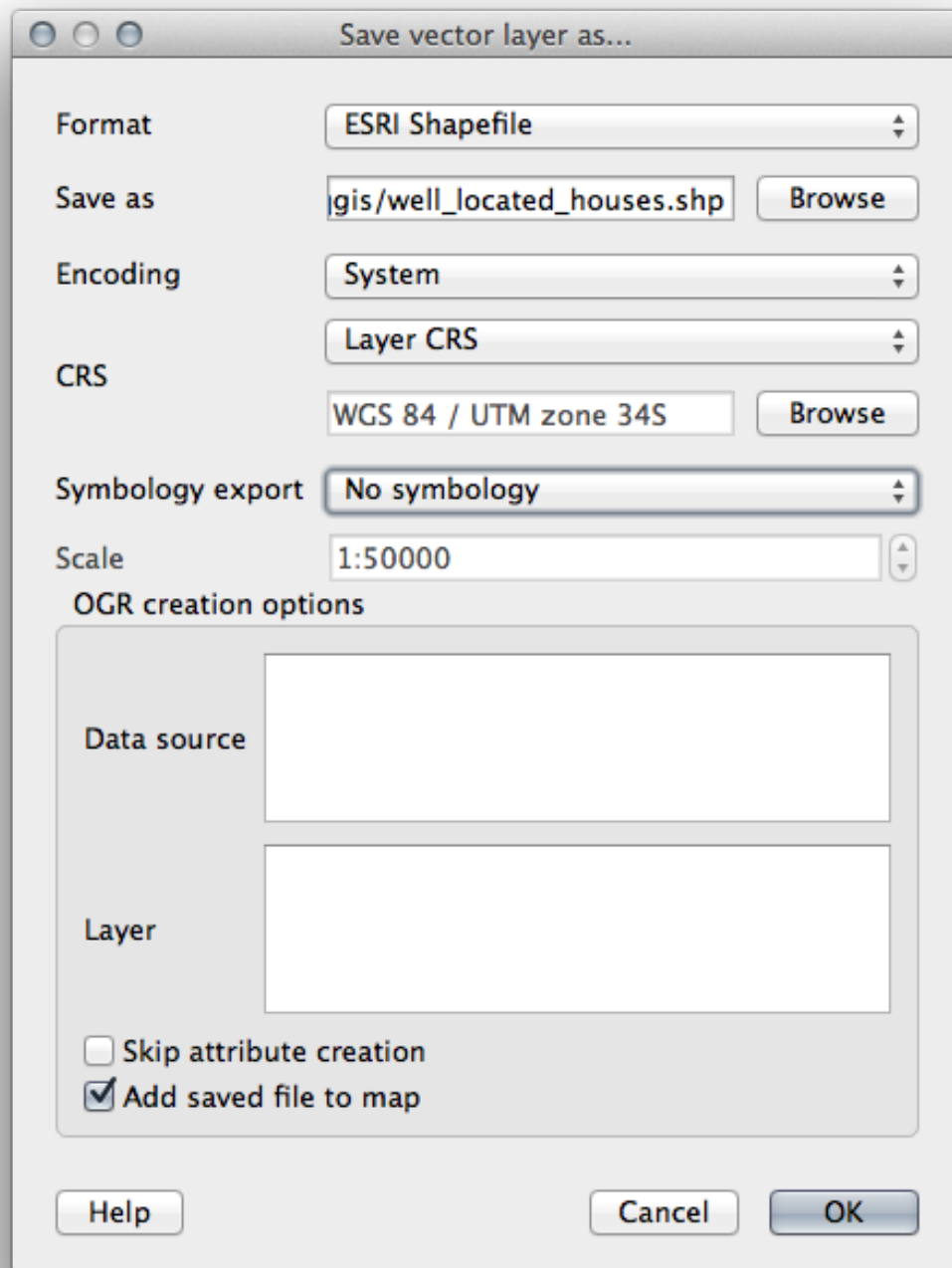


- Click *OK*, then *Close*.
- You'll probably find that not much seems to have changed. If so, move the `school_roads_intersect` layer to the bottom of the layers list, then zoom in:



The buildings highlighted in yellow are those which match our criteria and are selected, while the buildings in green are those which do not. We can now save the selected buildings as a new layer.

- Right-click on the *houses_34S* layer in the *Layers list*.
- Select *Save Selection As...*
- Set the dialog up like this:



- The file name is `well_located_houses.shp`.
- Click *OK*.

Now you have the selection as a separate layer and can remove the `houses_34S` layer.

7.2.12 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.

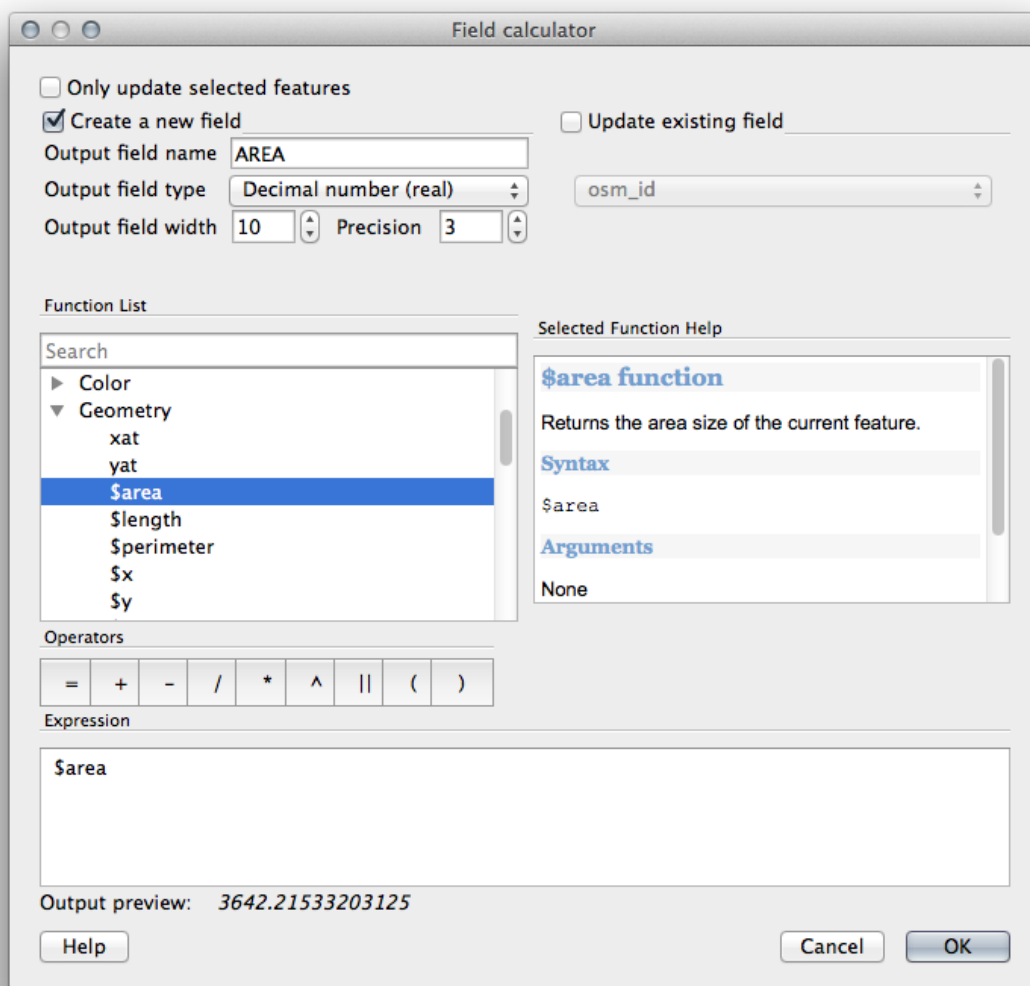
Using the processes described above, create a new layer called `houses_restaurants_500m` which further filters your `well_located_houses` layer to show only those which are within 500m of a restaurant.

Check your results

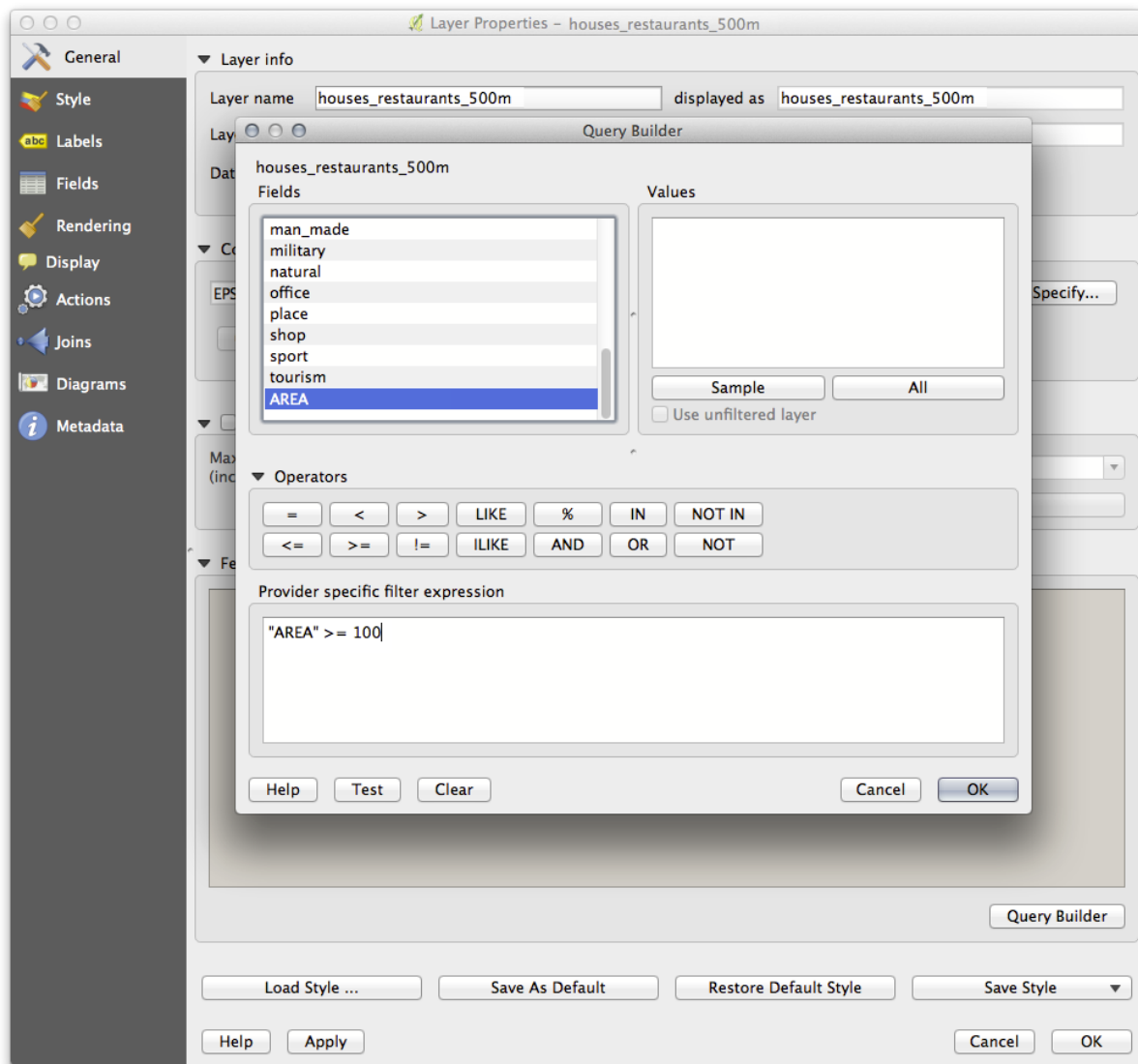
7.2.13 Follow Along: Select Buildings of the Right Size

To see which buildings are the correct size (more than 100 square metres), we first need to calculate their size.

- Open the attribute table for the `houses_restaurants_500m` layer.
- Enter edit mode and open the field calculator.
- Set it up like this:



- If you can't find *AREA* in the list, try creating a new field as you did in the previous lesson of this module.
- Click *OK*.
- Scroll to the right of the attribute table; your *AREA* field now has areas in metres for all the buildings in your *houses_restaurants_500m* layer.
- Click the edit mode button again to finish editing, and save your edits when prompted.
- Build a query as earlier in this lesson:



- Click **OK**. Your map should now only show you those buildings which match our starting criteria and which are more than 100m squared in size.

7.2.14 Try Yourself

- Save your solution as a new layer, using the approach you learned above for doing so. The file should be saved under the usual directory, with the name `solution.shp`.

7.2.15 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.

7.2.16 What's Next?

In the next lesson, we'll look at how to calculate the shortest distance along the road from one point to another.

7.3 Lesson: Network Analysis

Calculating the shortest distance between two points is a commonly cited use for GIS. QGIS ships with this tool, but it's not visible by default. In this brief lesson, we'll show you what you need to get started.

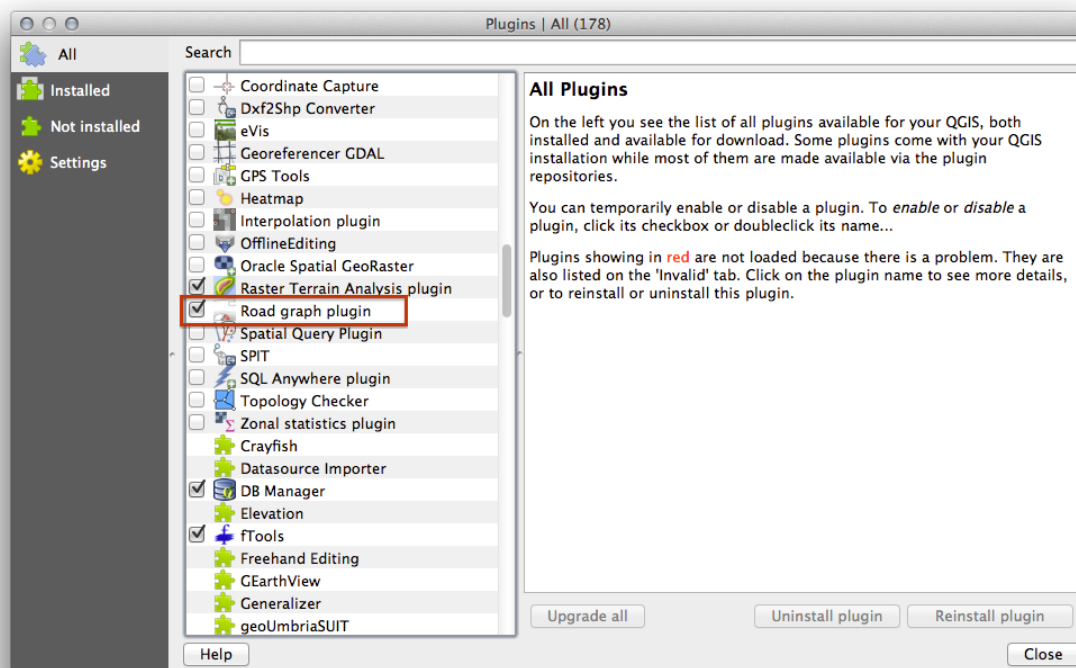
The goal for this lesson: To activate, configure and use the *Road Graph* plugin.

7.3.1 Follow Along: Activate the Tool

QGIS has many plugins that add to its basic functions. Many of these plugins are so useful that they ship along with the program straight out of the box. They're still hidden by default, though. So in order to use them, you need to activate them first.

To activate the *Road Graph* plugin:

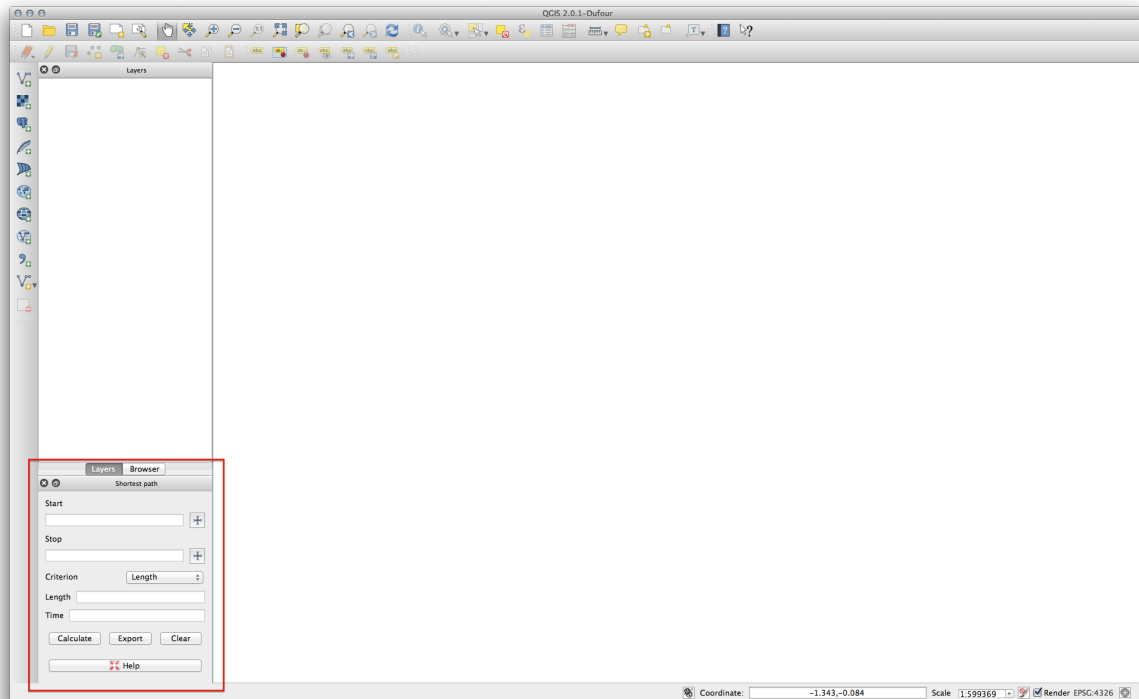
- Start the *Plugin Manager* by clicking on the QGIS main window's menu item *Plugins → Manage and Install Plugins....* A dialog appears.
- Select the plugin like this:



- Click *Close* on the *Plugin Manager* dialog.

Bemerkung: If you do not see the the plugin in your interface, go to *View → Panels* and ensure that *Shortest path* has a check mark next to it.

This panel will appear in your interface:

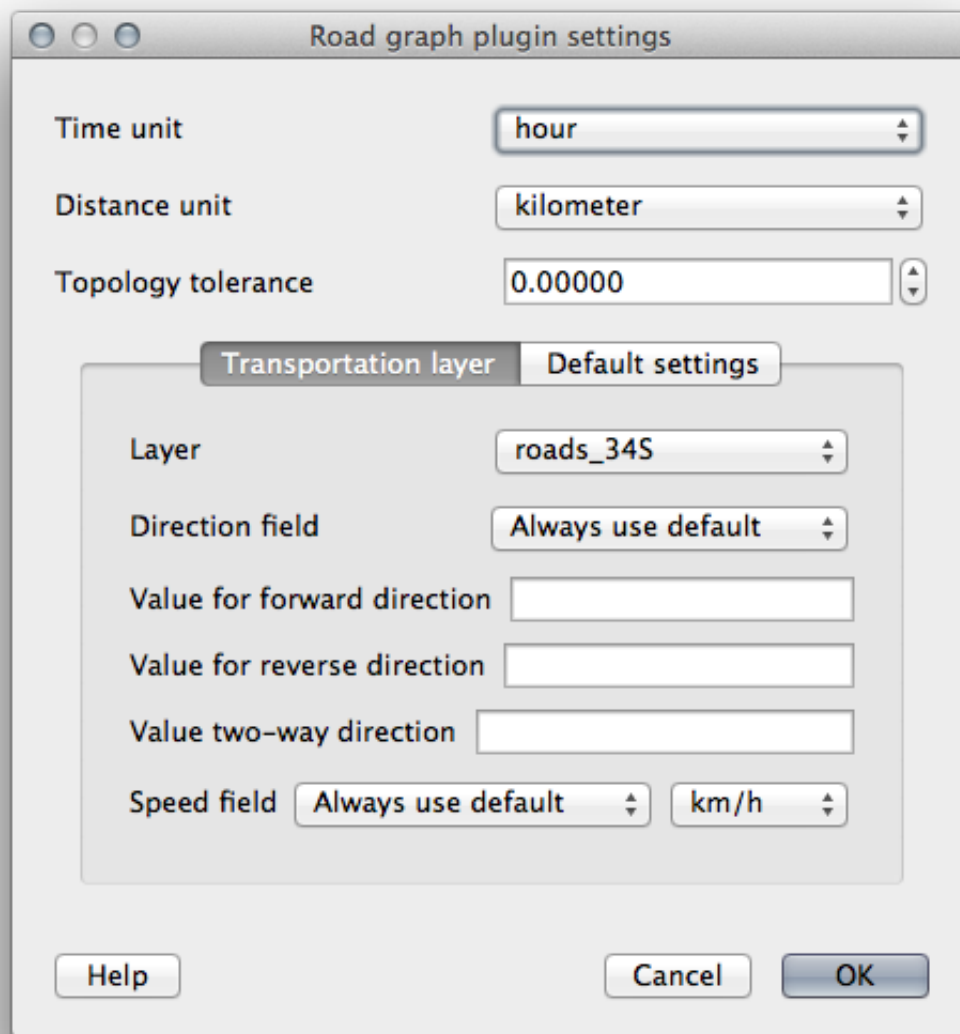


7.3.2 Follow Along: Configure the Tool

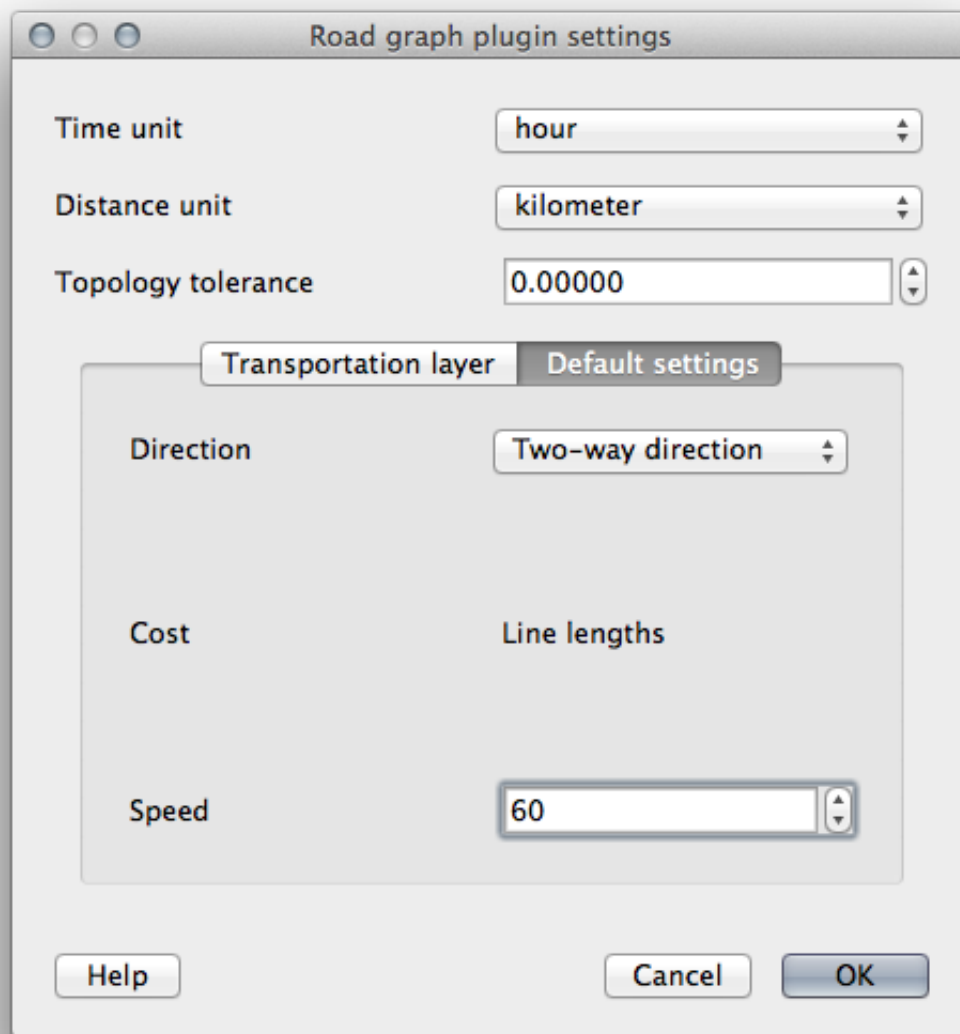
To have a layer to calculate on, first save your current map. If you haven't already done so, save your `roads_34S` layer to a shapefile by right-clicking the layer and selecting *Save as....* Create a new map and load this layer into it.

Since so many different configurations are possible when analyzing networks, the plugin doesn't assume anything before you've set it up. This means that it won't do anything at all if you don't set it up first.

- Click on the menu item *Vector* → *Road graph* → *settings*. A dialog will appear.
- Make sure it's set up like this (use defaults unless otherwise specified):



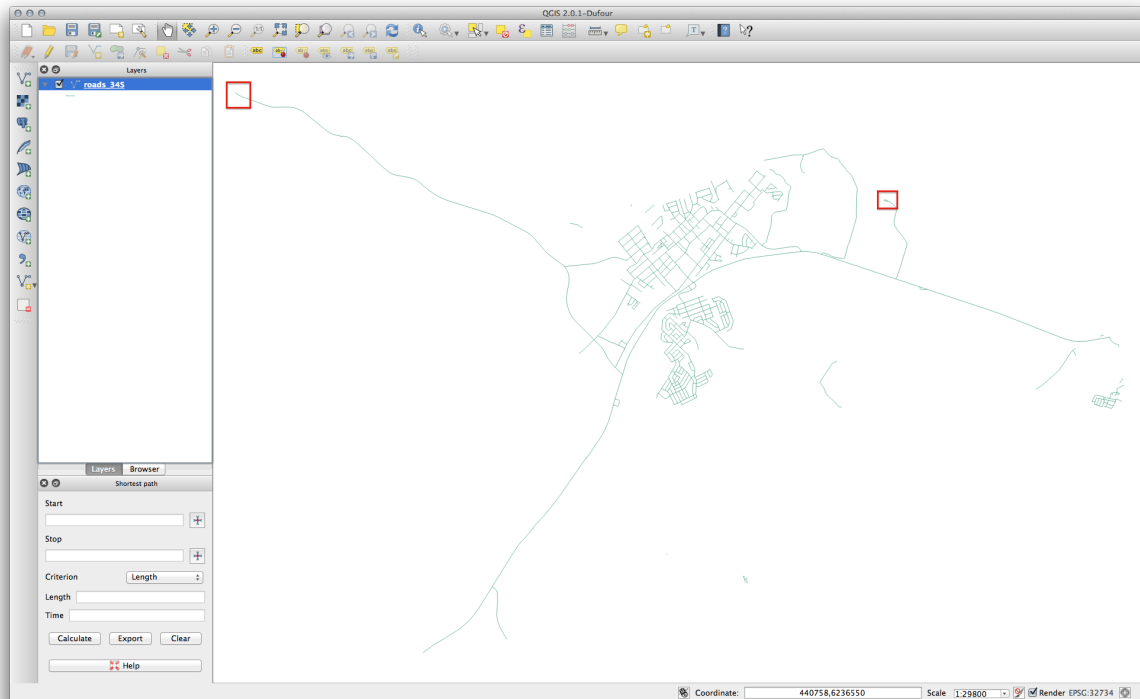
- *Time unit: hour*
- *Distance unit: kilometer*
- *Layer: roads_34S*
- *Speed field: Always use default / km/h*



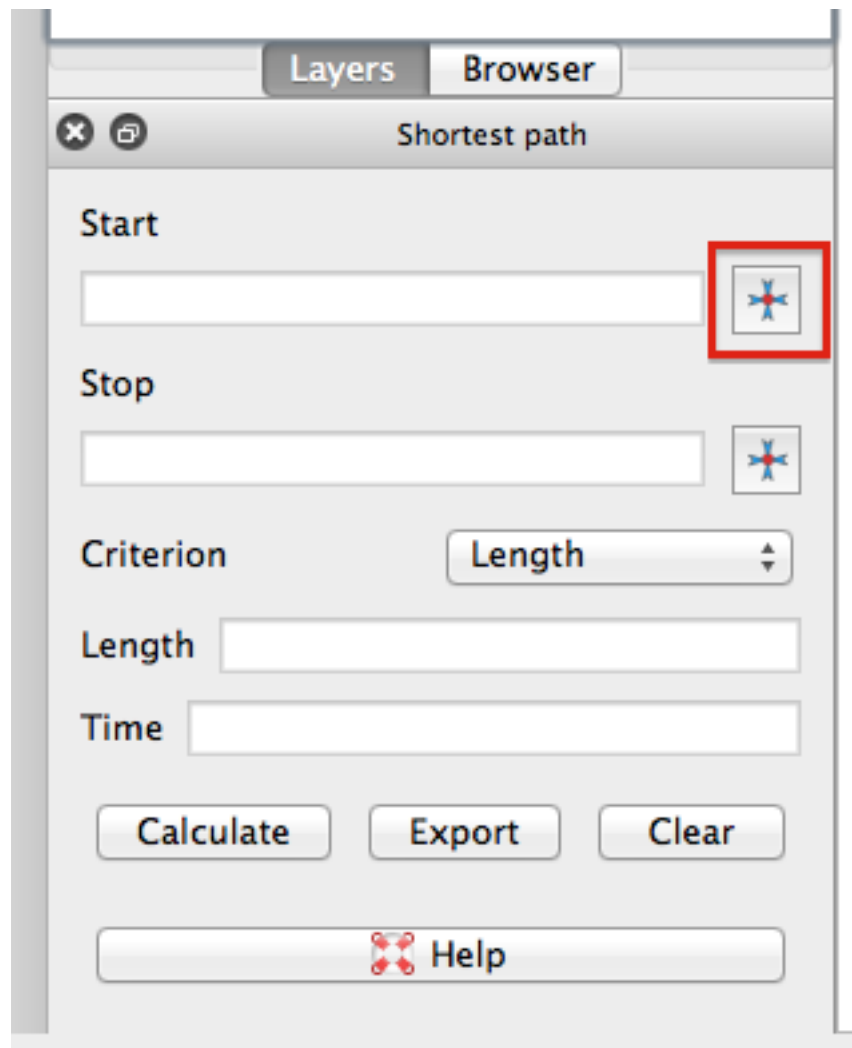
- *Direction: Two-way direction*
- *Speed: 60*

7.3.3 Follow Along: Use the Tool

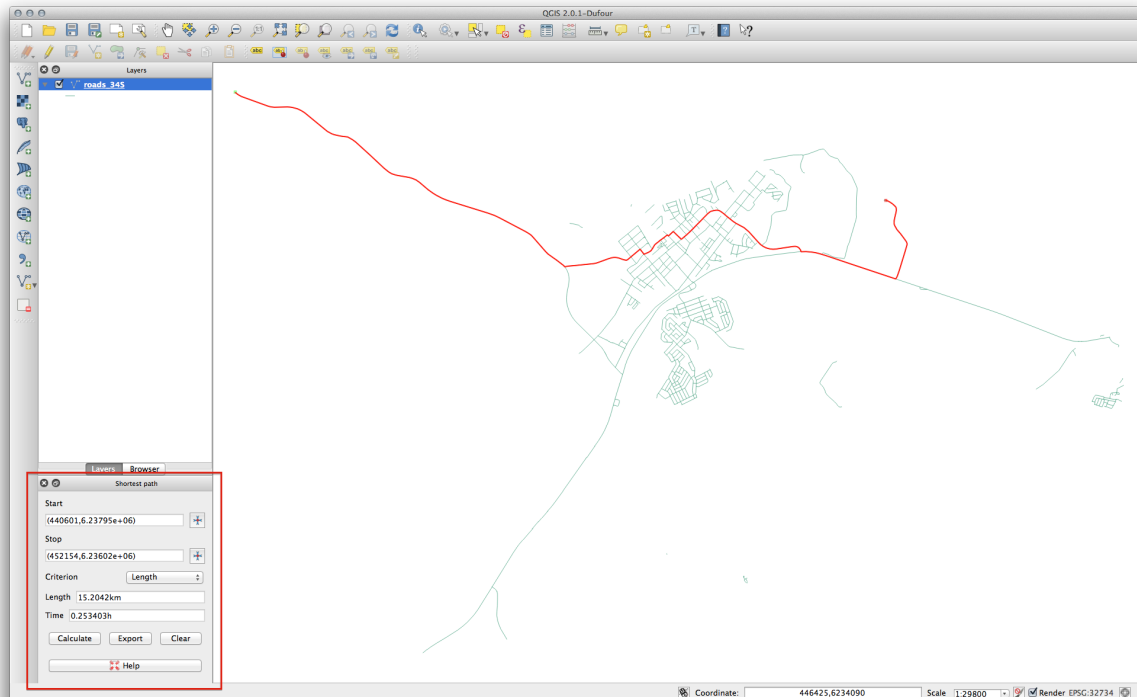
Find two points, on roads, on your map. They do not need to have any significance, but they should be connected by roads and separated by a reasonable distance:



- In the plugin panel, click on the *Capture Point* button next to the *Start* field:



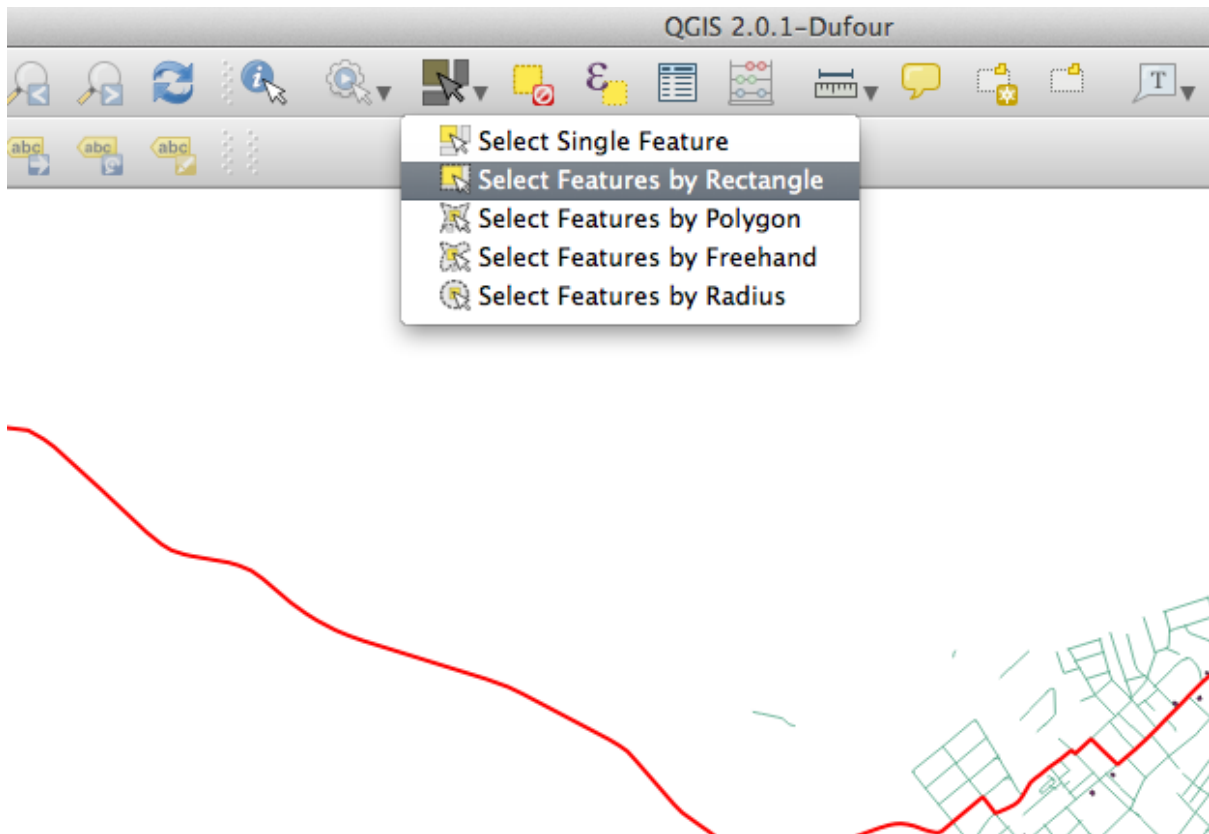
- Click on your chosen start point.
- Use the *Capture Point* button next to the *Stop* field and capture your chosen end point.
- Click on the *Calculate* button to see the solution:



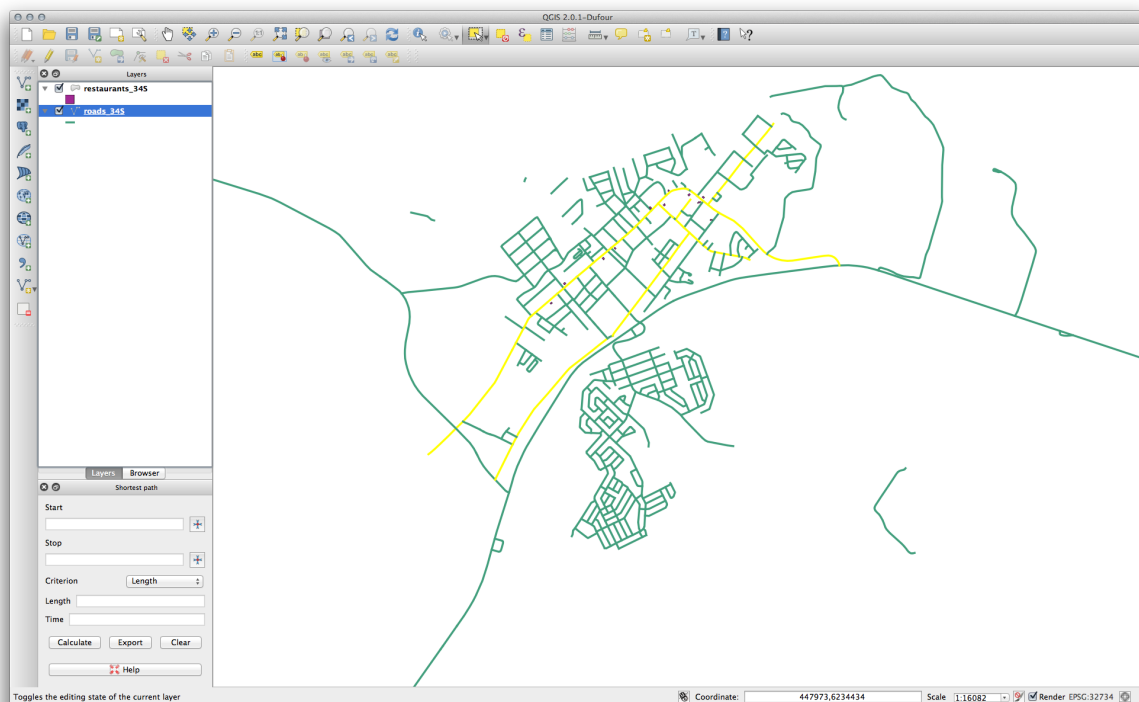
7.3.4 Follow Along: Using Criteria

Bemerkung: Section developed by Linfinity and S Motala (Cape Peninsula University of Technology)

- Add your `restaurants_34S` layer to the map (extract it from your analysis map if necessary).
- Open the attribute table for the `roads_34S` layer and enter edit mode.
- Add a new column with the name `SPEED`, and give it the type *Whole number (integer)* with a width of 3.
- In the main window, activate the *Select Features by Rectangle* tool:



- Select any main roads in urban - but not residential - areas:



(To select more than one road, hold the `ctrl` button and drag a box across any road that you want to include in the selection.)

- In the attribute table, select *Show selected features*.

Attribute table - roads_345 :: Features total: 302, filtered: 302, selected:

| | osm_id | name | highway | waterway | aerialway | barrier | man_made | other_tags | SPEED |
|-----|-----------|-----------------|-------------|----------|-----------|---------|----------|----------------|-------|
| 222 | 183010770 | Cooper Street | residential | NULL | NULL | NULL | NULL | "lanes"=>... | NULL |
| 226 | 183080638 | Kerk Street | residential | NULL | NULL | NULL | NULL | "name:af"=... | NULL |
| 229 | 183080642 | Swellengrebel | tertiary | NULL | NULL | NULL | NULL | "lanes"=>"2" | NULL |
| 277 | 238808188 | Voortrek St... | secondary | NULL | NULL | NULL | 0 | "lanes"=>"2" | NULL |
| 281 | 238992238 | Cooper Street | residential | NULL | NULL | NULL | NULL | "surface"=... | NULL |
| 286 | 238992244 | Somerset St... | residential | NULL | NULL | NULL | NULL | "lanes"=>... | NULL |
| 4 | 37608761 | Berg | residential | NULL | NULL | NULL | NULL | "maxspeed"=... | NULL |
| 1 | 4937372 | Voortrek St... | tertiary | NULL | NULL | NULL | NULL | "lanes"=>... | NULL |
| 57 | 55692509 | Cooper Street | residential | NULL | NULL | NULL | NULL | NULL | NULL |
| 64 | 55692517 | Kerk Street | residential | NULL | NULL | NULL | NULL | "surface"=... | NULL |
| 70 | 59406669 | NULL | residential | NULL | NULL | NULL | NULL | NULL | NULL |
| 91 | 100187237 | NULL | service | NULL | NULL | NULL | NULL | NULL | NULL |
| 92 | 101398165 | NULL | trunk | NULL | NULL | NULL | NULL | "bridge"=>... | NULL |
| 93 | 101398171 | NULL | trunk | NULL | NULL | NULL | NULL | "maxspeed"=... | NULL |
| 94 | 169449294 | Rhenius | residential | NULL | NULL | NULL | NULL | NULL | NULL |
| 95 | 169451631 | De Mist Street | residential | NULL | NULL | NULL | NULL | "oneway"=... | NULL |
| 96 | 170968185 | Bonteboek St... | residential | NULL | NULL | NULL | NULL | "access"=>... | NULL |
| 97 | 172750608 | NULL | service | NULL | NULL | NULL | NULL | "service"=>... | NULL |
| 98 | 177153014 | NULL | residential | NULL | NULL | NULL | NULL | NULL | NULL |
| 99 | 177153015 | NULL | service | NULL | NULL | NULL | NULL | NULL | NULL |
| 100 | 177153016 | Akasia | residential | NULL | NULL | NULL | NULL | NULL | NULL |
| 101 | 177153017 | NULL | residential | NULL | NULL | NULL | NULL | NULL | NULL |
| 102 | 177153018 | NULL | residential | NULL | NULL | NULL | NULL | NULL | NULL |

Context menu options:

- Show All Features
- Show Selected Features
- Show Features Visible On Map
- Show Edited and New Features
- Column Filter
- Advanced Filter (Expression)

- Set the SPEED value for all the selected streets to 60:

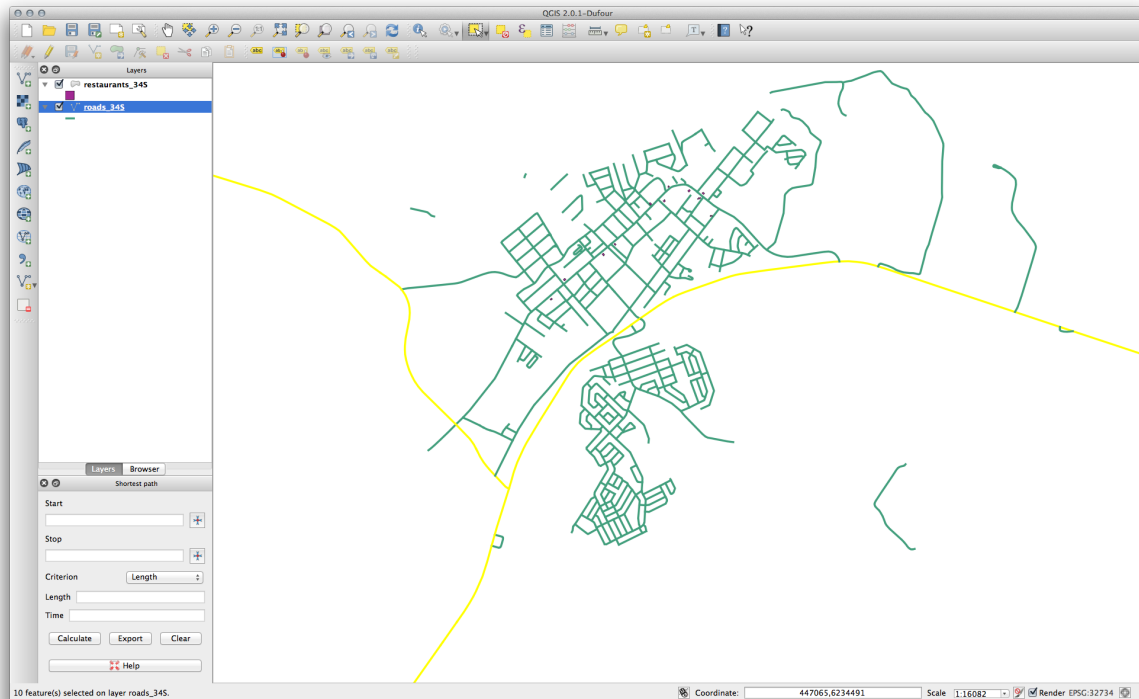
Attribute table - roads_345 :: Features total: 302, filtered: 302, selected: 11

| | osm_id | name | highway | waterway | aerialway | barrier | man_made | other_tags | SPEED |
|-----|-----------|----------------|-------------|----------|-----------|---------|----------|----------------|-------|
| 222 | 183010770 | Cooper Street | residential | NULL | NULL | NULL | NULL | "lanes"=>... | 60 |
| 226 | 183080638 | Kerk Street | residential | NULL | NULL | NULL | NULL | "name:af"=... | 60 |
| 229 | 183080642 | Swellengrebel | tertiary | NULL | NULL | NULL | NULL | "lanes"=>"2" | 60 |
| 277 | 238808188 | Voortrek St... | secondary | NULL | NULL | NULL | 0 | "lanes"=>"2" | 60 |
| 281 | 238992238 | Cooper Street | residential | NULL | NULL | NULL | NULL | "surface"=... | 60 |
| 286 | 238992244 | Somerset St... | residential | NULL | NULL | NULL | NULL | "lanes"=>... | 60 |
| 4 | 37608761 | Berg | residential | NULL | NULL | NULL | NULL | "maxspeed"=... | 60 |
| 1 | 4937372 | Voortrek St... | tertiary | NULL | NULL | NULL | NULL | "lanes"=>... | 60 |
| 57 | 55692509 | Cooper Street | residential | NULL | NULL | NULL | NULL | NULL | 60 |
| 64 | 55692517 | Kerk Street | residential | NULL | NULL | NULL | NULL | "surface"=... | 60 |
| 70 | 59406669 | NULL | residential | NULL | NULL | NULL | NULL | NULL | 60 |

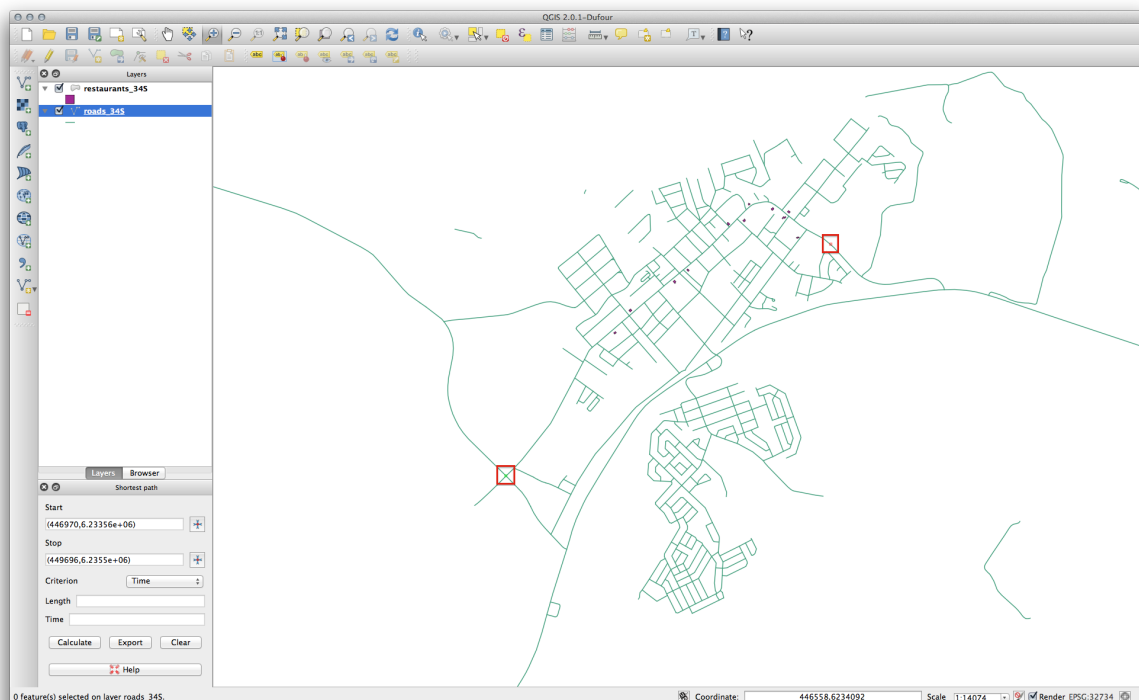
Show Selected Features

In context, this means that you're setting the speed limit on those roads to 60 km/h.

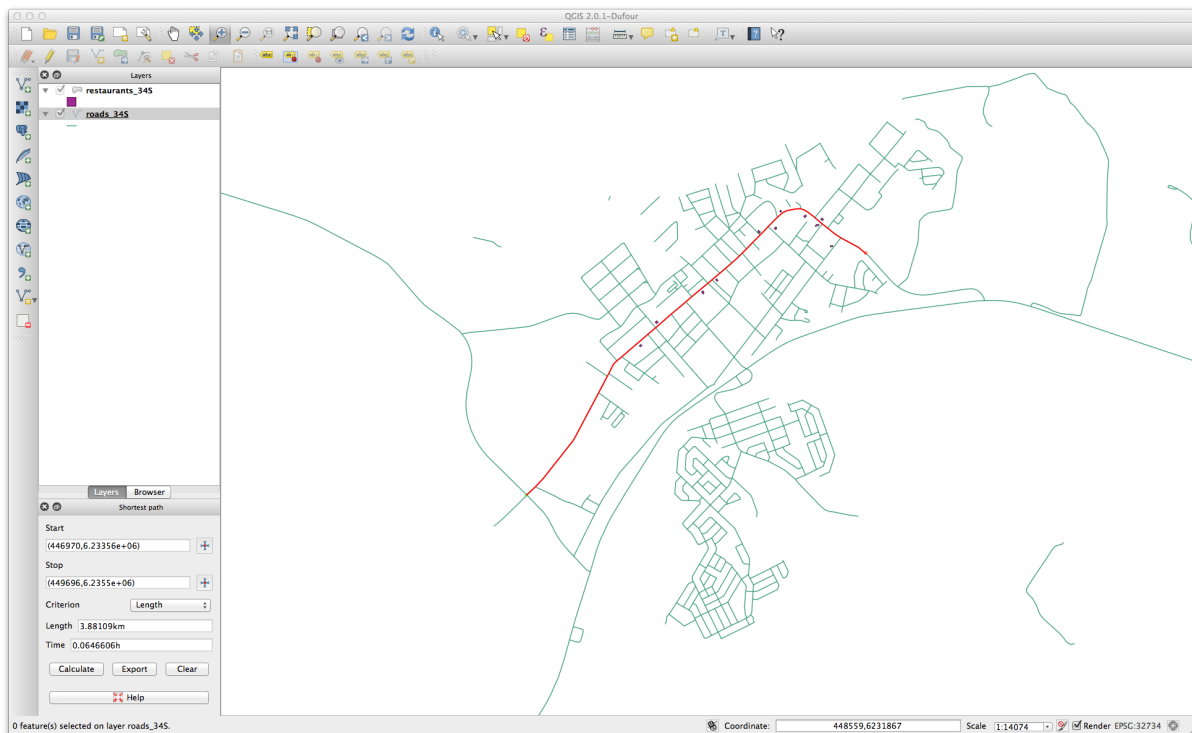
- Select any highways or major roads outside urban areas:



- Set the **SPEED** value for all the selected streets to 120.
- Close the attribute table, save your edits, and exit edit mode.
- Check the *Vector* → *Road graph* → *Road graph settings* to ensure that it's set up as explained previously in this lesson, but with the *Speed* value set to the *SPEED* field you just created.
- In the *Shortest path* panel, click the *Start point* button.
- Set the starting point on a minor road on one side of Swellendam and the end point on a major road on the other side of town:

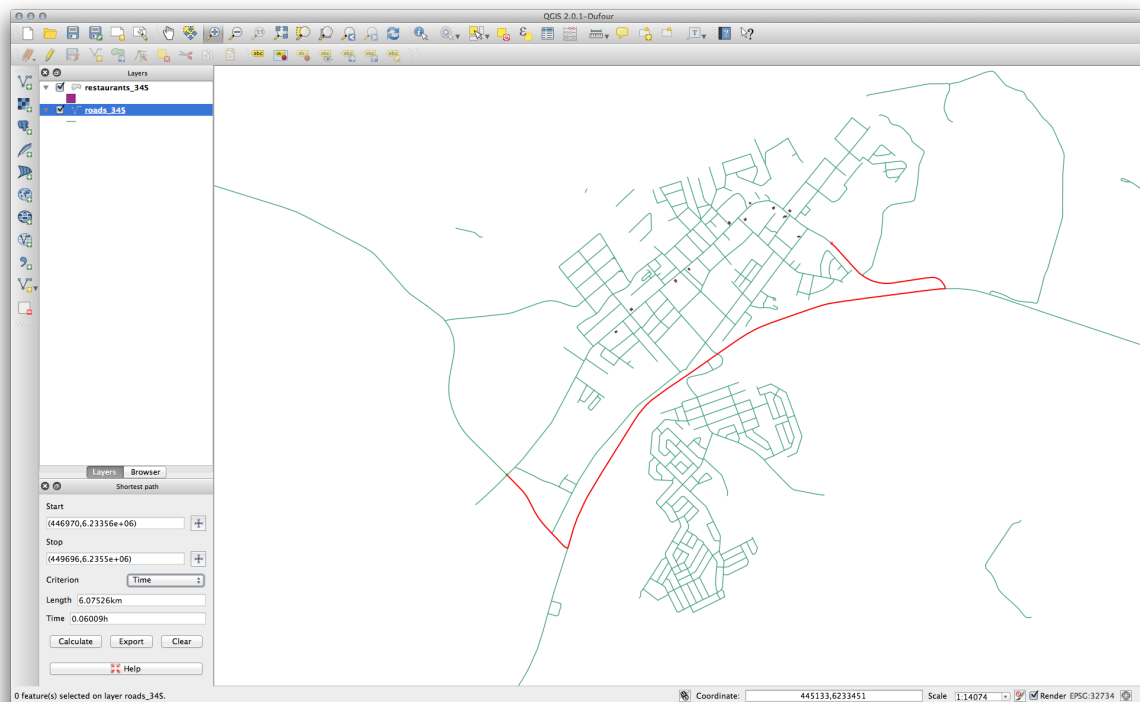


- In the *Criterion* drop-down list in the *Shortest path* panel, select *Length*.
- Click *Calculate*. The route will be calculated for the shortest distance:



Notice the values of *Length* and *Time* in the *Shortest path* panel.

- Set the *Criterion* to *Time*.
- Click *Calculate* again. The route will be calculated for the shortest time:



You can switch back and forth between these criteria, recalculating each time, and note the changes in the *Length* and *Time* taken. Remember that the assumption being made to arrive at the time taken to travel a route does not

account for acceleration, and assumes that you will be traveling at the speed limit at all times. In a real situation, you may want to split roads into smaller sections and note the average or expected speed in each section, rather than the speed limit.

If, on clicking *Calculate*, you see an error stating that a path could not be found, make sure that the roads you digitized actually meet each other. If they're not quite touching, either fix them by modifying the features, or set the *Topology tolerance* in the plugin's settings. If they're passing over each other without intersecting, use the *Split features* tool to "split" roads at their intersections:



Remember that the *Split features* tool only works in edit mode on selected features, though!

You might also find that the shortest route is also the quickest if this error is returned.

7.3.5 In Conclusion

Now you know how to use the *Road Graph* plugin to solve shortest-path problems.

7.3.6 What's Next?

Next you'll see how to run spatial statistics algorithms on vector datasets.

7.4 Lesson: Spatial Statistics

Bemerkung: Lesson developed by Linfiniti and S Motala (Cape Peninsula University of Technology)

Spatial statistics allow you to analyze and understand what is going on in a given vector dataset. QGIS includes several standard tools for statistical analysis which prove useful in this regard.

The goal for this lesson: To know how to use QGIS' spatial statistics tools.

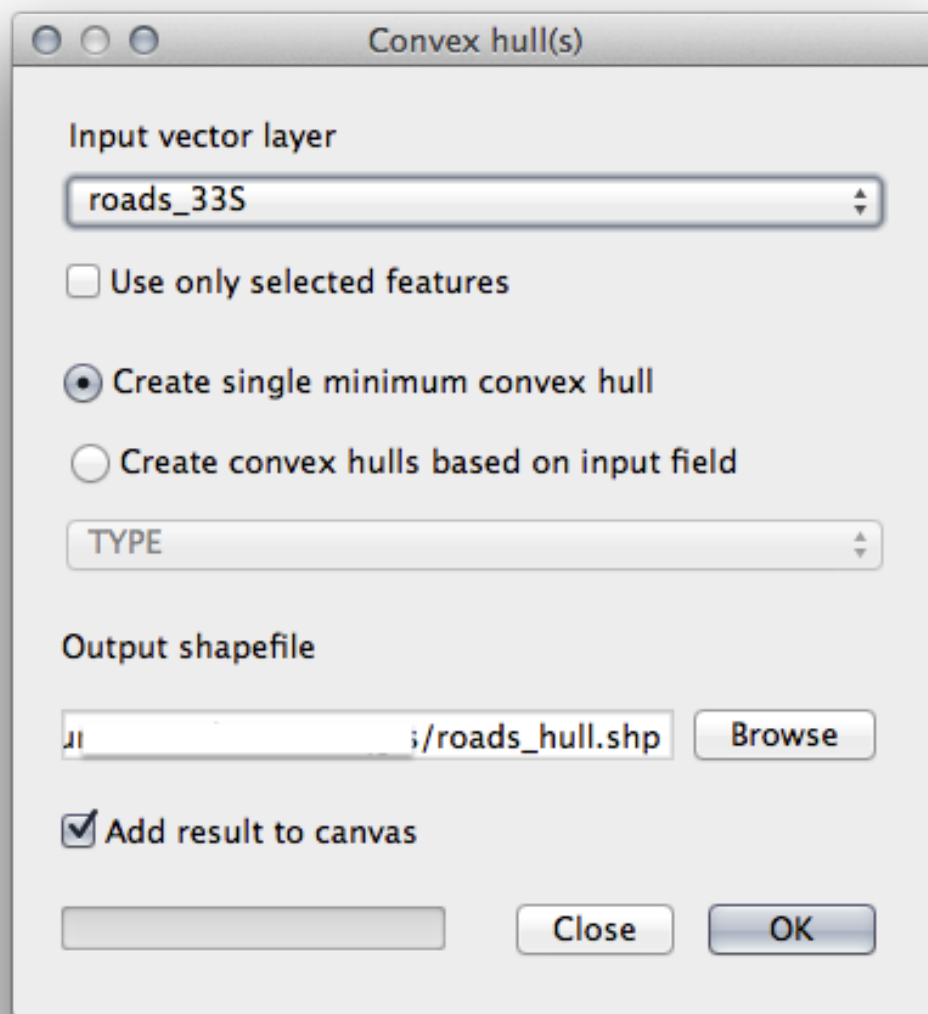
7.4.1 Follow Along: Create a Test Dataset

In order to get a point dataset to work with, we'll create a random set of points.

To do so, you'll need a polygon dataset defining the extents of the area you want to create the points in.

We'll use the area covered by streets.

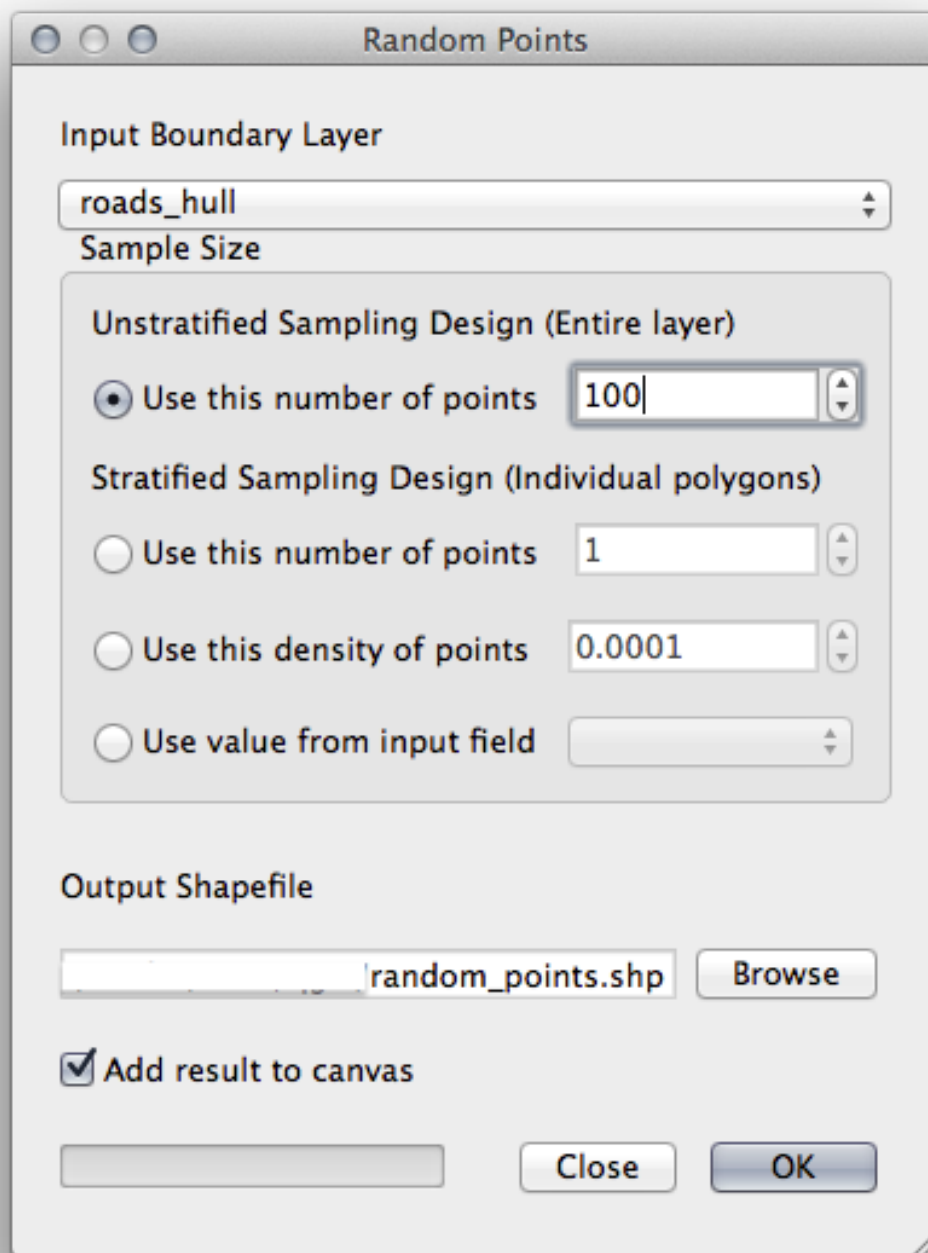
- Create a new empty map.
- Add the `roads_33S` layer from the `exercise_data/projected_data` directory, as well as the `srtm_41_19.tif` raster (elevation data) found in `exercise_data/raster/SRTM/`.
- Use the *Convex hull(s)* tool (available under *Vector* → *Geoprocessing Tools*) to generate an area enclosing all the roads:



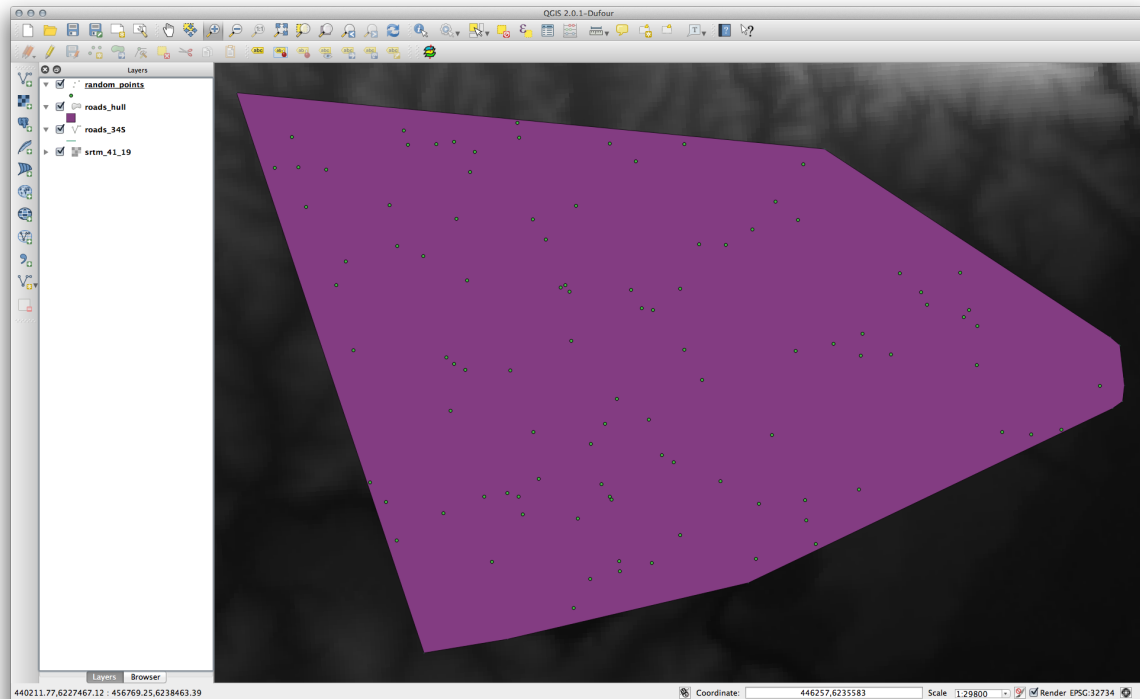
- Save the output under `exercise_data/spatial_statistics/` as `roads_hull.shp`.
- Add it to the TOC (*Layers list*) when prompted.

Creating random points

- Create random points in this area using the tool at *Vector → Research Tools → Random points*:

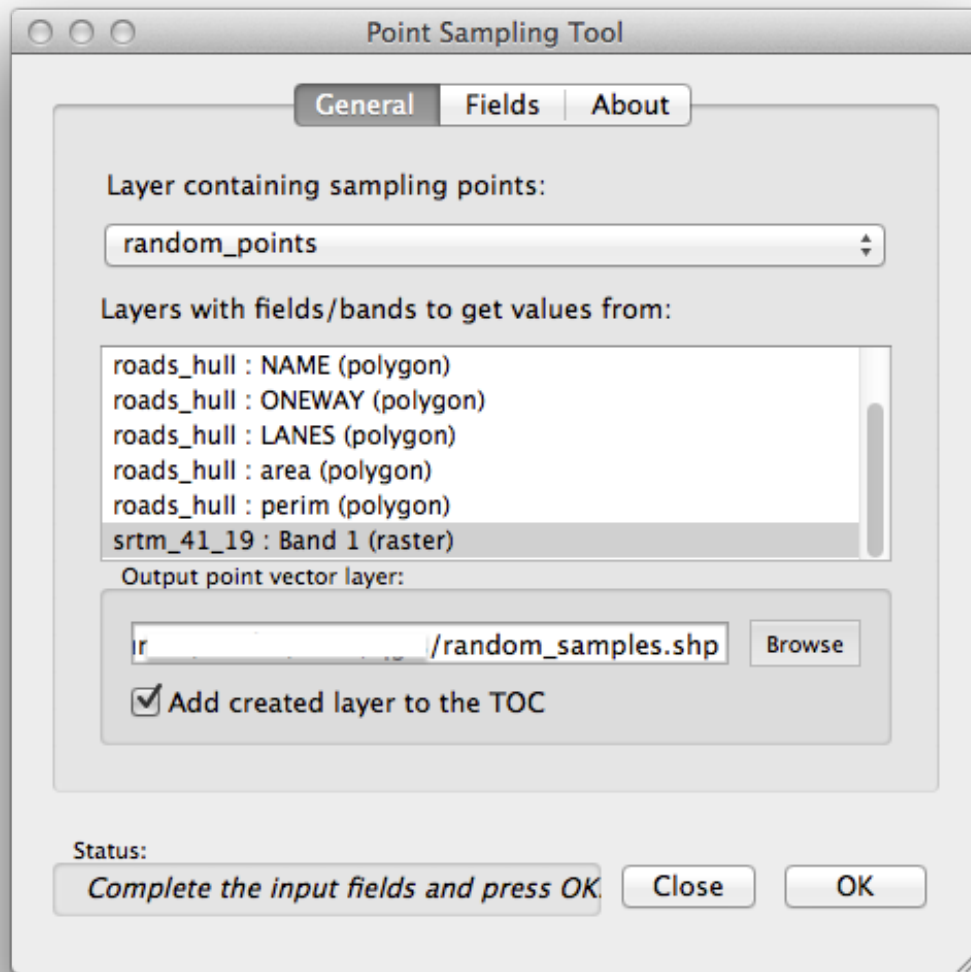


- Save the output under `exercise_data/spatial_statistics/` as `random_points.shp`.
- Add it to the TOC (*Layers list*) when prompted:



Sampling the data

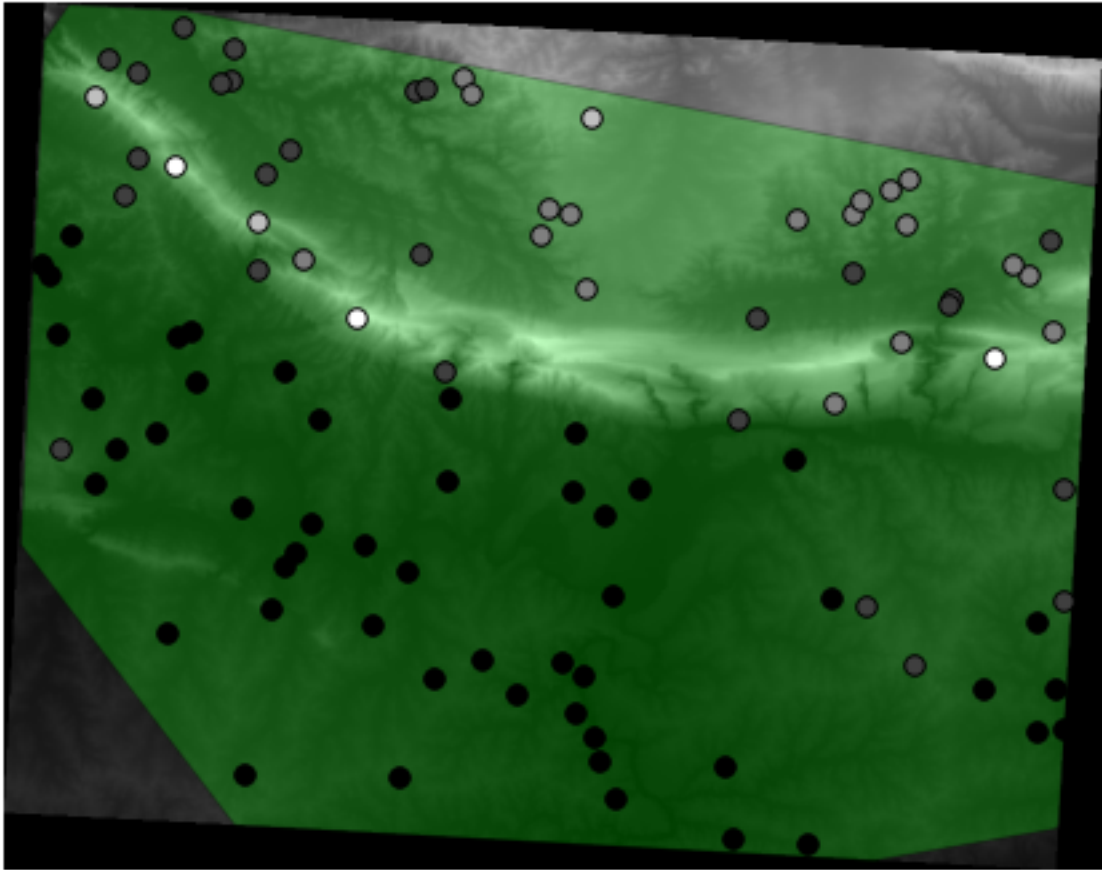
- To create a sample dataset from the raster, you'll need to use the *Point sampling tool* plugin.
- Refer ahead to the module on plugins if necessary.
- Search for the phrase `point sampling` in the *Plugin → Manage and Install Plugins...* and you will find the plugin.
- As soon as it has been activated with the *Plugin Manager*, you will find the tool under *Plugins → Analyses → Point sampling tool*:



- Select *random_points* as the layer containing sampling points, and the SRTM raster as the band to get values from.
- Make sure that “Add created layer to the TOC” is checked.
- Save the output under `exercise_data/spatial_statistics/` as `random_samples.shp`.

Now you can check the sampled data from the raster file in the attributes table of the *random_samples* layer, they will be in a column named `srtm_41_19`.

A possible sample layer is shown here:

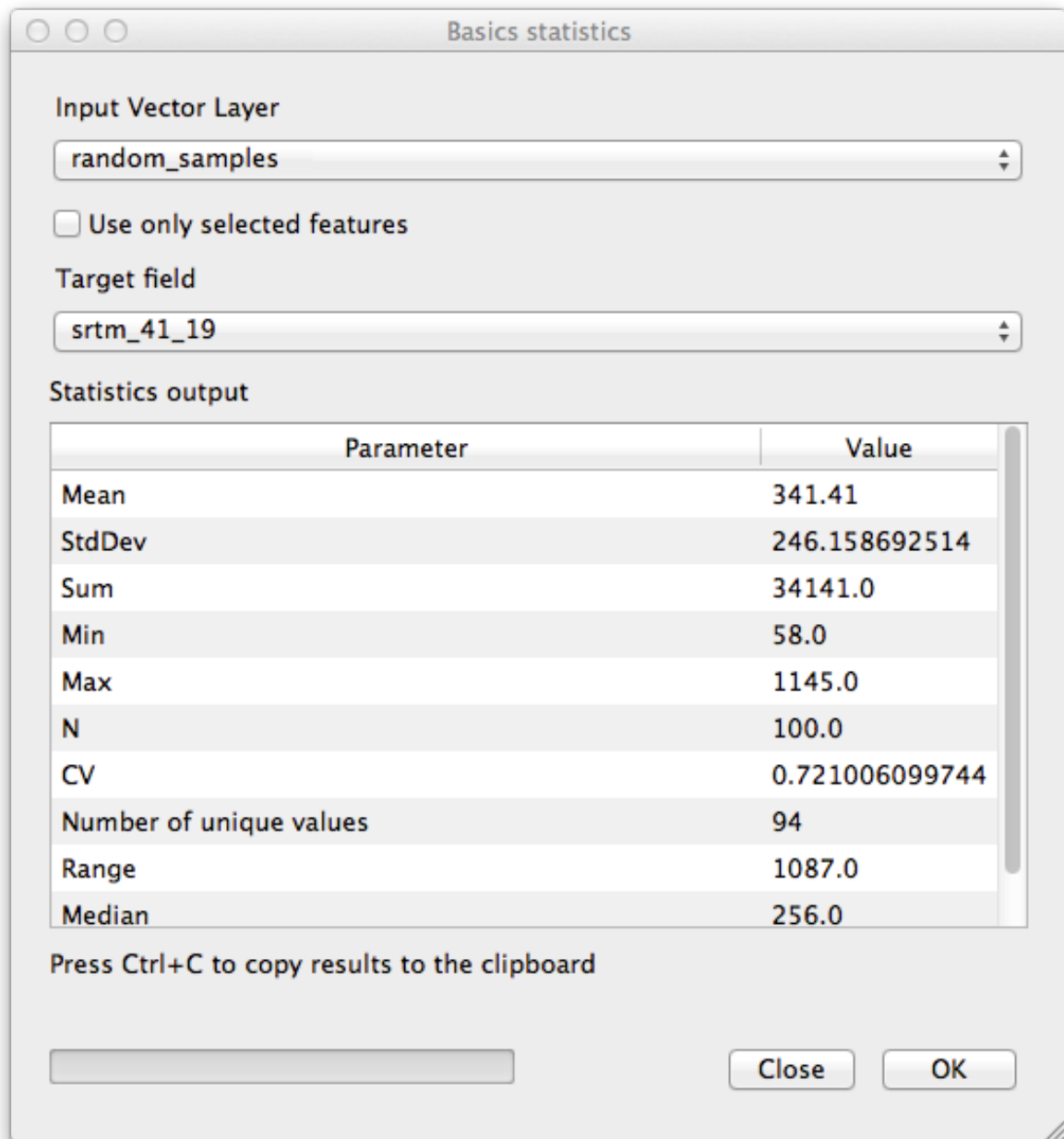


The sample points are classified by their value such that darker points are at a lower altitude. You'll be using this sample layer for the rest of the statistical exercises.

7.4.2 Follow Along: Basic Statistics

Now get the basic statistics for this layer.

- Click on the *Vector* → *Analysis Tools* → *Basic statistics* menu entry.
- In the dialog that appears, specify the *random_samples* layer as the source.
- Make sure that the *Target field* is set to *srtm_41_19* which is the field you will calculate statistics for.
- Click *OK*. You'll get results like this:



Bemerkung: You can copy and paste the results into a spreadsheet. The data uses a (colon :) separator.

| | A | B |
|----|-------------------------|-------------|
| 1 | Mean | 343.9 |
| 2 | StdDev | 254.4824748 |
| 3 | Sum | 34390 |
| 4 | Min | 34 |
| 5 | Max | 1226 |
| 6 | N | 100 |
| 7 | CV | 0.739989749 |
| 8 | Number of unique values | 91 |
| 9 | Range | 1192 |
| 10 | Median | 269 |

- Close the plugin dialog when done.

To understand the statistics above, refer to this definition list:

Mean The mean (average) value is simply the sum of the values divided by the amount of values.

StdDev 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.

Sum All the values added together.

Min The minimum value.

Max The maximum value.

N The amount of samples/values.

CV The spatial covariance of the dataset.

Number of unique values The number of values that are unique across this dataset. If there are 90 unique values in a dataset with N=100, then the 10 remaining values are the same as one or more of each other.

Range The difference between the minimum and maximum values.

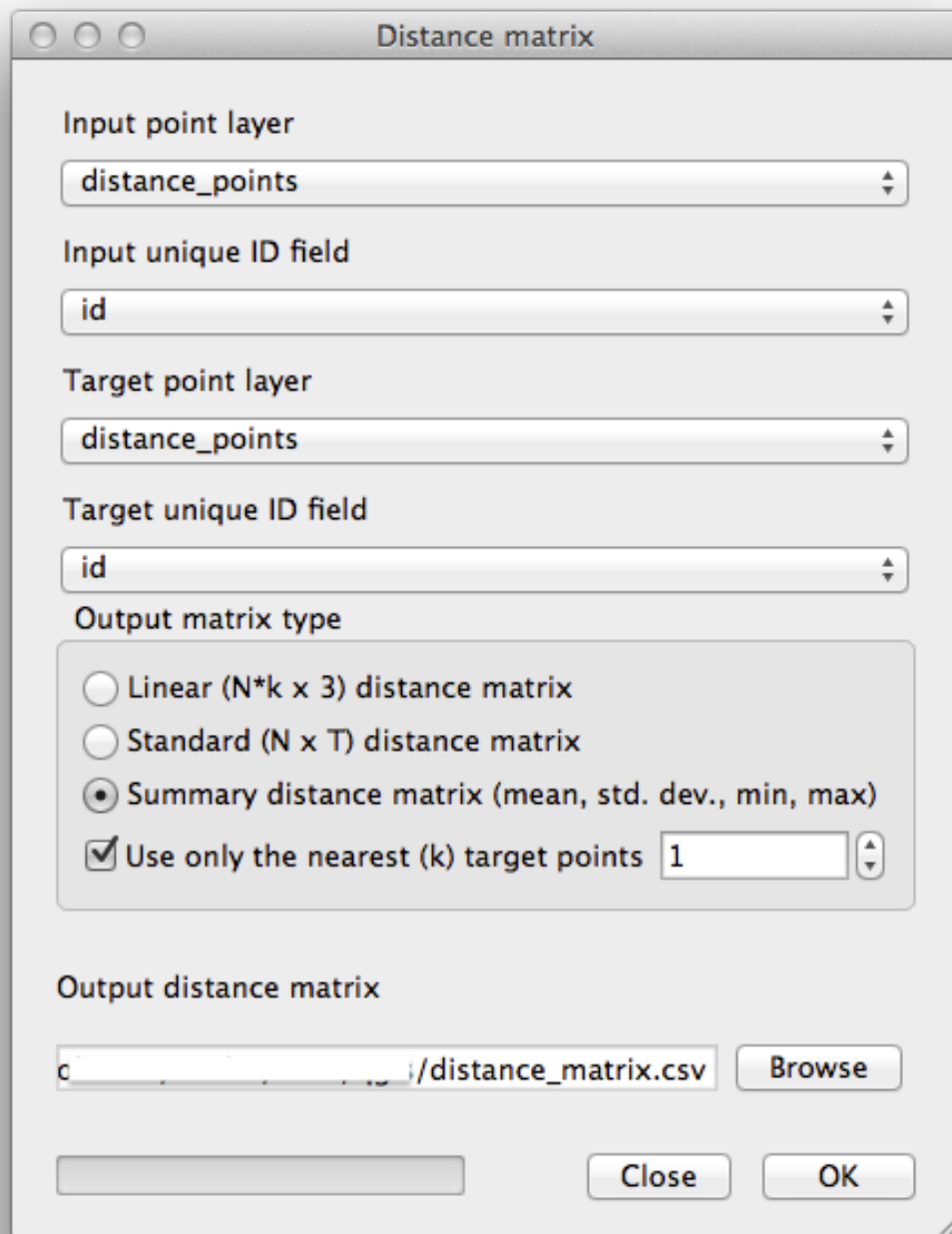
Median If you arrange all the values from least to greatest, the middle value (or the average of the two middle values, if N is an even number) is the median of the values.

7.4.3 Follow Along: Compute a Distance Matrix

- Create a new point layer in the same projection as the other datasets (WGS 84 / UTM 33S).
- Enter edit mode and digitize three point somewhere among the other points.
- Alternatively, use the same random point generation method as before, but specify only three points.
- Save your new layer as `distance_points.shp`.

To generate a distance matrix using these points:

- Open the tool *Vector* → *Analysis Tools* → *Distance matrix*.
- Select the *distance_points* layer as the input layer, and the *random_samples* layer as the target layer.
- Set it up like this:



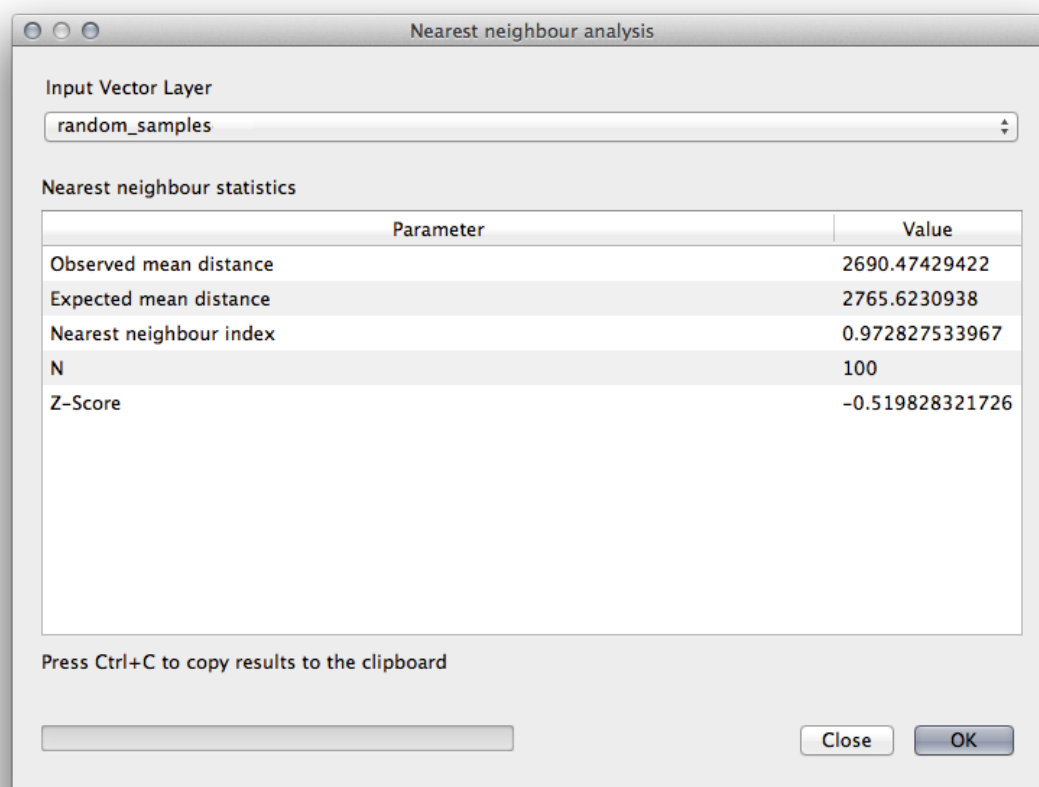
- Save the result as `distance_matrix.csv`.
- Click *OK* to generate the distance matrix.
- Open it in a spreadsheet program to see the results. Here is an example:

| InputID | MEAN | STDDEV | MIN | MAX |
|---------|----------------|--------|----------------|----------------|
| 3 | 0.195448627921 | 0 | 0.195448627921 | 0.195448627921 |
| 2 | 0.174928758638 | 0 | 0.174928758638 | 0.174928758638 |
| 1 | 0.174928758638 | 0 | 0.174928758638 | 0.174928758638 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

7.4.4 Follow Along: Nearest Neighbor Analysis

To do a nearest neighbor analysis:

- Click on the menu item *Vector* → *Analysis Tools* → *Nearest neighbor analysis*.
- In the dialog that appears, select the *random_samples* layer and click *OK*.
- The results will appear in the dialog's text window, for example:



Bemerkung: You can copy and paste the results into a spreadsheet. The data uses a (colon :) separator.

7.4.5 Follow Along: Mean Coordinates

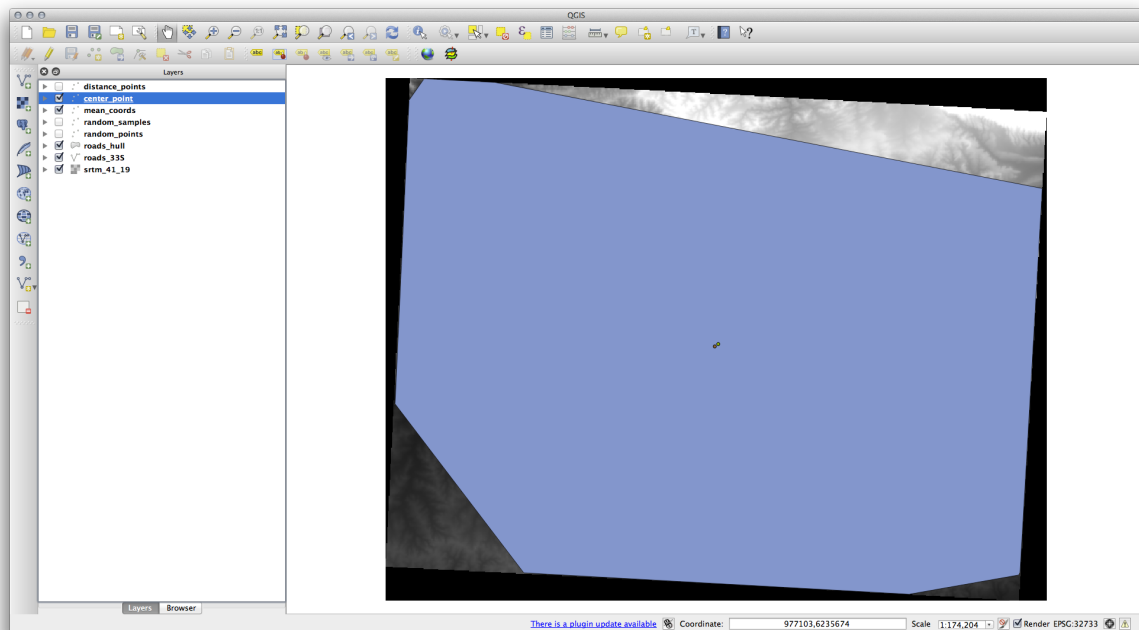
To get the mean coordinates of a dataset:

- Click on the *Vector* → *Analysis Tools* → *Mean coordinate(s)* menu item.
- In the dialog that appears, specify *random_samples* as the input layer, but leave the optional choices unchanged.
- Specify the output layer as *mean_coords.shp*.
- Click *OK*.
- Add the layer to the *Layers list* when prompted.

Let's compare this to the central coordinate of the polygon that was used to create the random sample.

- Click on the *Vector* → *Geometry Tools* → *Polygon centroids* menu item.
- In the dialog that appears, select *roads_hull* as the input layer.
- Save the result as *center_point*.
- Add it to the *Layers list* when prompted.

As you can see from the example below, the mean coordinates and the center of the study area (in orange) don't necessarily coincide:

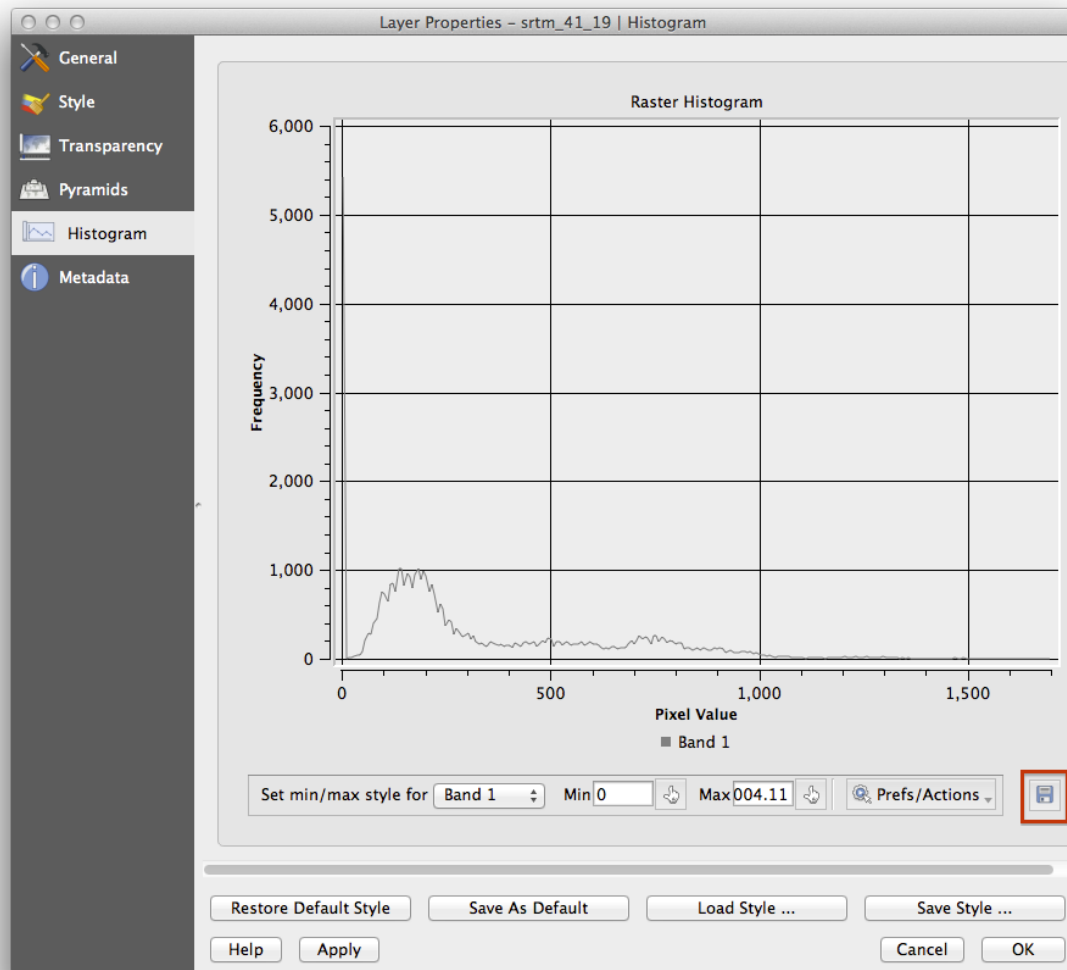


7.4.6 Follow Along: Image Histograms

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.

- In your *Layers list*, right-click on the SRTM DEM layer.
- Select *Properties*.
- Choose the tab *Histogram*. You may need to click on the *Compute Histogram* button to generate the graphic. You will see a graph describing the frequency of values in the image.

- You can export it as an image:



- Select the *Metadata* tab, you can see more detailed information inside the *Properties* box.

The mean value is 332.8, and the maximum value is 1699! But those values don't show up on the histogram. Why not? It's because there are so few of them, compared to the abundance of pixels with values below the mean. That's also why the histogram extends so far to the right, even though there is no visible red line marking the frequency of values higher than about 250.

Therefore, keep in mind that a histogram shows you the distribution of values, and not all values are necessarily visible on the graph.

- (You may now close *Layer Properties*.)

7.4.7 Follow Along: Spatial Interpolation

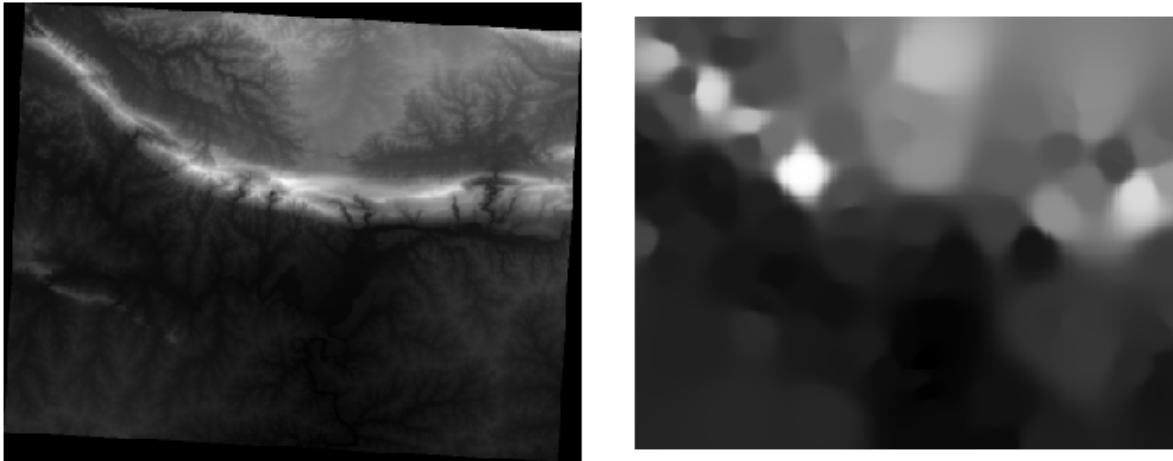
Let's say you have a collection of sample points from which you would like to extrapolate data. For example, you might have access to the *random_samples* dataset we created earlier, and would like to have some idea of what the terrain looks like.

To start, launch the *Grid (Interpolation)* tool by clicking on the *Raster* → *Analysis* → *Grid (Interpolation)* menu item.

- In the *Input file* field, select *random_samples*.

- Check the *Z Field* box, and select the field `srtm_41_19`.
- Set the *Output file* location to `exercise_data/spatial_statistics/interpolation.tif`.
- Check the *Algorithm* box and select *Inverse distance to a power*.
- Set the *Power* to `5.0` and the *Smoothing* to `2.0`. Leave the other values as-is.
- Check the *Load into canvas when finished* box and click *OK*.
- When it's done, click *OK* on the dialog that says `Process completed`, click *OK* on the dialog showing feedback information (if it has appeared), and click *Close* on the *Grid (Interpolation)* dialog.

Here's 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.

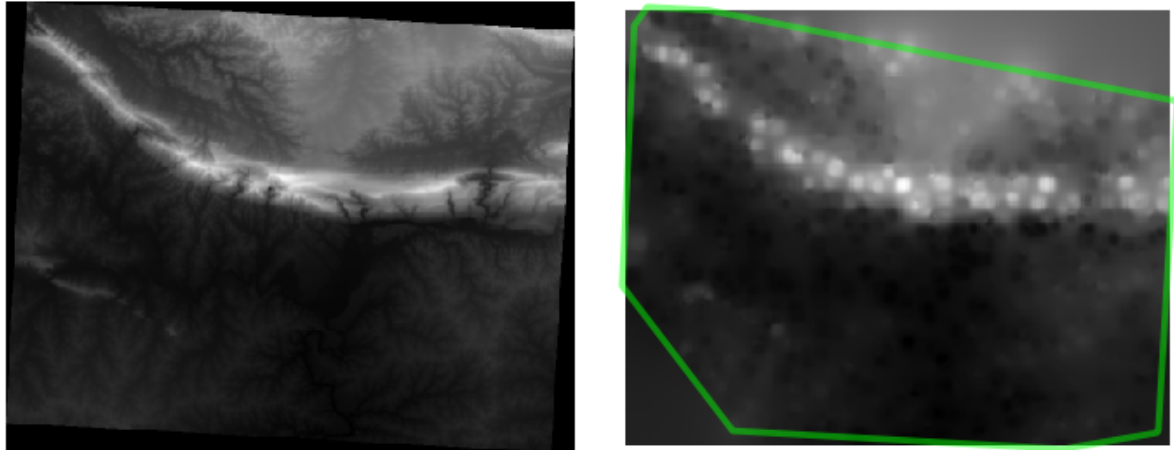


As you can see, 100 sample points aren't really enough to get a detailed impression of the terrain. It gives a very general idea, but it can be misleading as well. For example, in the image above, it is not clear that there is a high, unbroken mountain running from east to west; rather, the image seems to show a valley, with high peaks to the west. Just using visual inspection, we can see that the sample dataset is not representative of the terrain.

7.4.8 Try Yourself

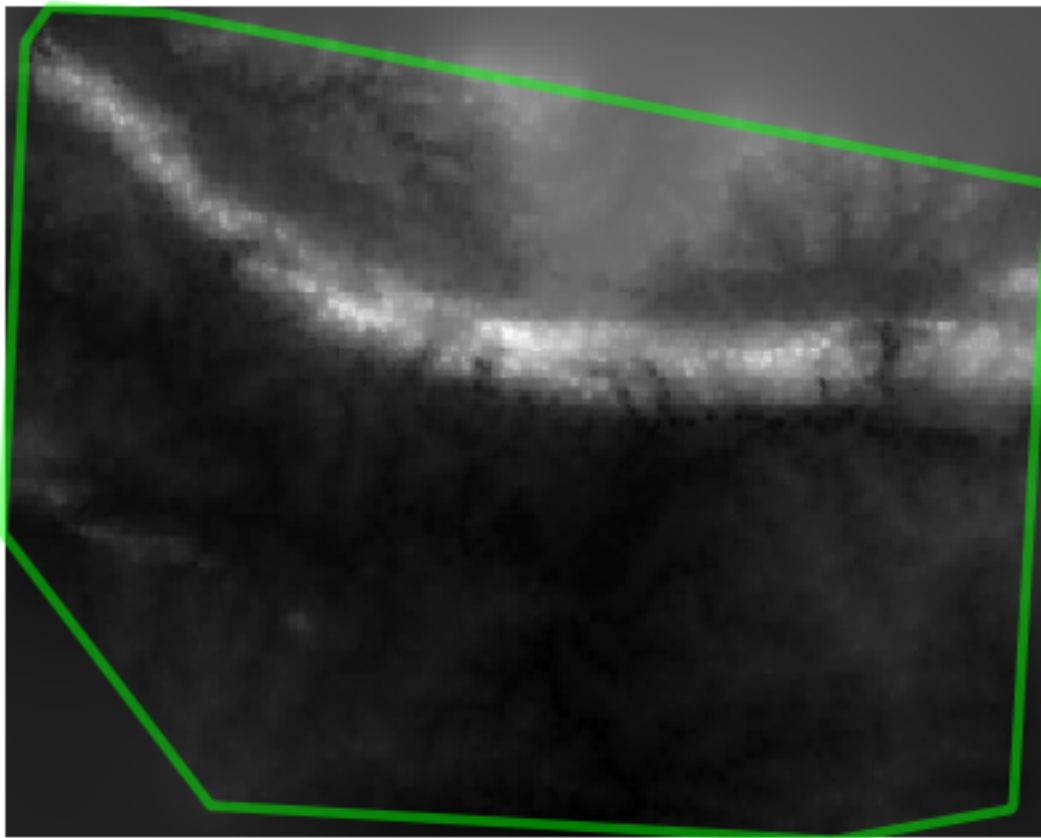
- Use the processes shown above to create a new set of 1000 random points.
- Use these points to sample the original DEM.
- Use the *Grid (Interpolation)* tool on this new dataset as above.
- Set the output filename to `interpolation_1000.tif`, with *Power* and *Smoothing* set to `5.0` and `2.0`, respectively.

The results (depending on the positioning of your random points) will look more or less like this:



The border shows the *roads_hull* layer (which represents the boundary of the random sample points) to explain the sudden lack of detail beyond its edges. This is a much better representation of the terrain, due to the much greater density of sample points.

Here is an example of what it looks like with 10 000 sample points:

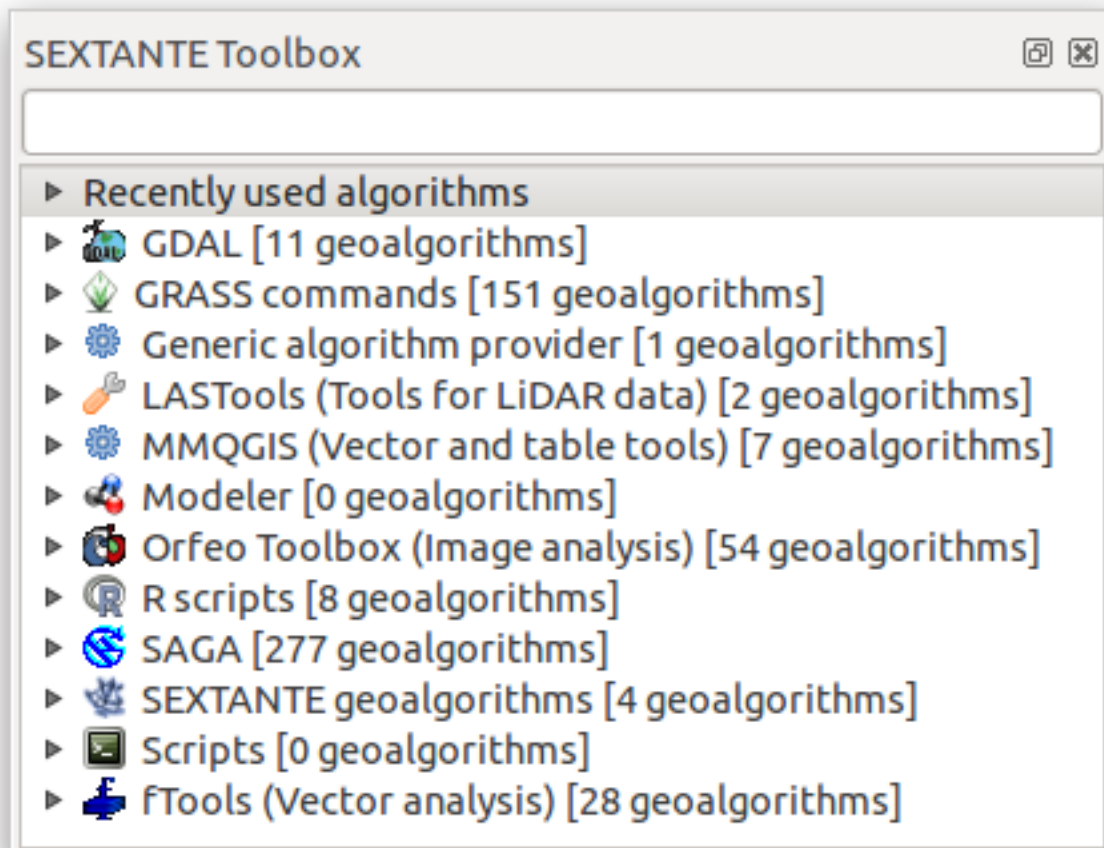


Bemerkung: It's not recommended that you try doing this with 10 000 sample points if you are not working on a fast computer, as the size of the sample dataset requires a lot of processing time.

7.4.9 Follow Along: Additional Spatial Analysis Tools

Originally a separate project and then accessible as a plugin, the SEXTANTE software has been added to QGIS as a core function from version 2.0. You can find it as a new QGIS menu with its new name *Processing* from where you can access a rich toolbox of spatial analysis tools allows you to access various plugin tools from within a single interface.

- Activate this set of tools by enabling the *Processing* → *Toolbox* menu entry. The toolbox looks like this:



You will probably see it docked in QGIS to the right of the map. Note that the tools listed here are links to the actual tools. Some of them are SEXTANTE's own algorithms and others are links to tools that are accessed from external applications such as GRASS, SAGA or the Orfeo Toolbox. This external applications are installed with QGIS so you are already able to make use of them. In case you need to change the configuration of the Processing tools or, for example, you need to update to a new version of one of the external applications, you can access its setting from *Processing* → *Options and configurations*.

7.4.10 Follow Along: Spatial Point Pattern Analysis

For a simple indication of the spatial distribution of points in the *random_samples* dataset, we can make use of SAGA's *Spatial Point Pattern Analysis* tool via the *Processing Toolbox* you just opened.

- In the *Processing Toolbox*, search for this tool *Spatial Point Pattern Analysis*.
- Double-click on it to open its dialog.

Installing SAGA

Bemerkung: If SAGA is not installed on your system, the plugin's dialog will inform you that the dependency is missing. If this is not the case, you can skip these steps.

On Windows

Included in your course materials you will find the SAGA installer for Windows.

- Start the program and follow its instructions to install SAGA on your Windows system. Take note of the path you are installing it under!

Once you have installed SAGA, you'll need to configure SEXTANTE to find the path it was installed under.

- Click on the menu entry *Analysis → SAGA options and configuration*.
- In the dialog that appears, expand the *SAGA* item and look for *SAGA folder*. Its value will be blank.
- In this space, insert the path where you installed SAGA.

On Ubuntu

- Search for *SAGA GIS* in the *Software Center*, or enter the phrase `sudo apt-get install saga-gis` in your terminal. (You may first need to add a SAGA repository to your sources.)
- QGIS will find SAGA automatically, although you may need to restart QGIS if it doesn't work straight away.

On Mac

Homebrew users can install SAGA with this command:

- `brew install saga-core`

If you do not use Homebrew, please follow the instructions here:

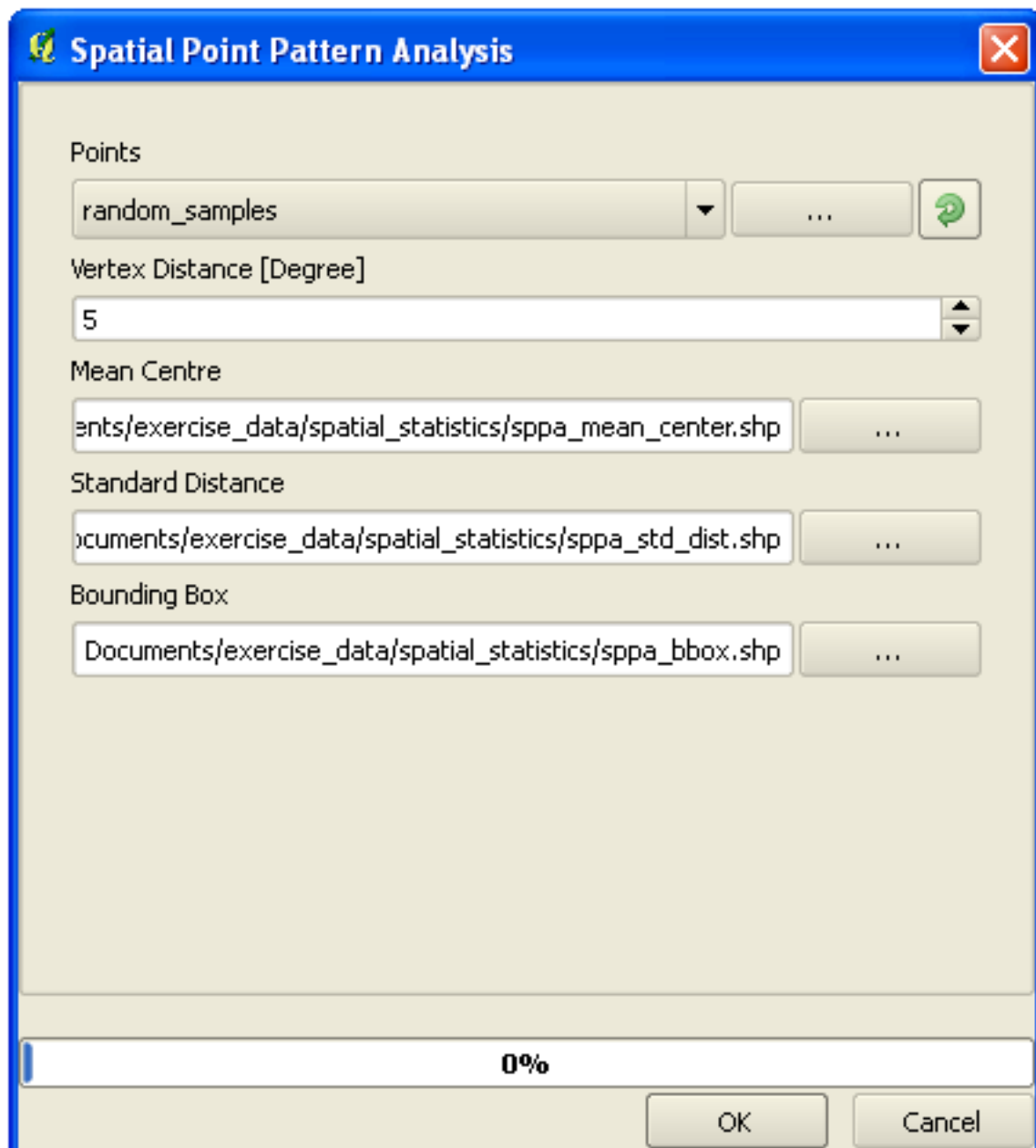
<http://sourceforge.net/apps/trac/saga-gis/wiki/Compiling%20SAGA%20on%20Mac%20OS%20X>

After installing

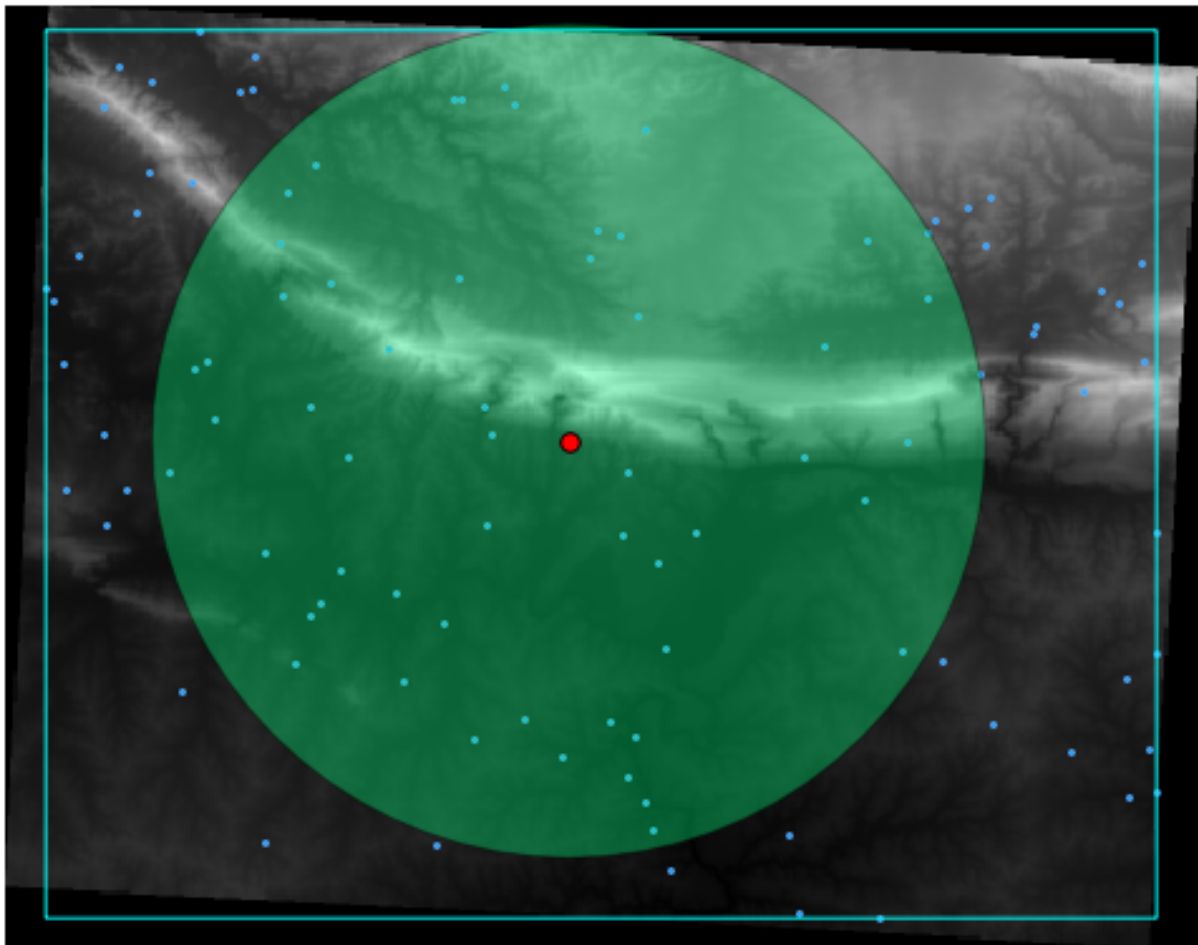
Now that you have installed and configured SAGA, its functions will become accessible to you.

Using SAGA

- Open the SAGA dialog.
- SAGA produces three outputs, and so will require three output paths.
- Save these three outputs under `exercise_data/spatial_statistics/`, using whatever file names you find appropriate.



The output will look like this (the symbology was changed for this example):



The red dot is the mean center; the large circle is the standard distance, which gives an indication of how closely the points are distributed around the mean center; and the rectangle is the bounding box, describing the smallest possible rectangle which will still enclose all the points.

7.4.11 Follow Along: Minimum Distance Analysis

Often, the output of an algorithm will not be a shapefile, but rather a table summarizing the statistical properties of a dataset. One of these is the *Minimum Distance Analysis* tool.

- Find this tool in the *Processing Toolbox* as :guilabel:‘Minimum

Distance Analysis‘.

It does not require any other input besides specifying the vector point dataset to be analyzed.

- Choose the *random_points* dataset.
- Click *OK*. On completion, a DBF table will appear in the *Layers list*.
- Select it, then open its attribute table. Although the figures may vary, your results will be in this format:

| | NAME ▾ | VALUE |
|---|--------------------|---------------|
| 0 | Mean Average | 2823.45817848 |
| 1 | Minimum | 424.0860061 |
| 2 | Maximum | 9773.35250512 |
| 3 | Standard Deviation | 1662.40681133 |
| 4 | Duplicates | 0 |

7.4.12 In Conclusion

QGIS allows many possibilities for analyzing the spatial statistical properties of datasets.

7.4.13 What's Next?

Now that we've covered vector analysis, why not see what can be done with rasters? That's what we'll do in the next module!

Module: Rasters

We've used rasters for digitizing before, but raster data can also be used directly. In this module, you'll see how it's done in QGIS.

8.1 Lesson: Working with Raster Data

Raster data is quite different from vector data. Vector data has discrete features 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 of various different color values.

During this module you're 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 the QGIS environment.

8.1.1 Follow Along: Loading Raster Data

- Open your `analysis.qgs` map.
- Deactivate all the layers except the *solution* and *important_roads* layers.
- Click on the *Load Raster Layer* button:



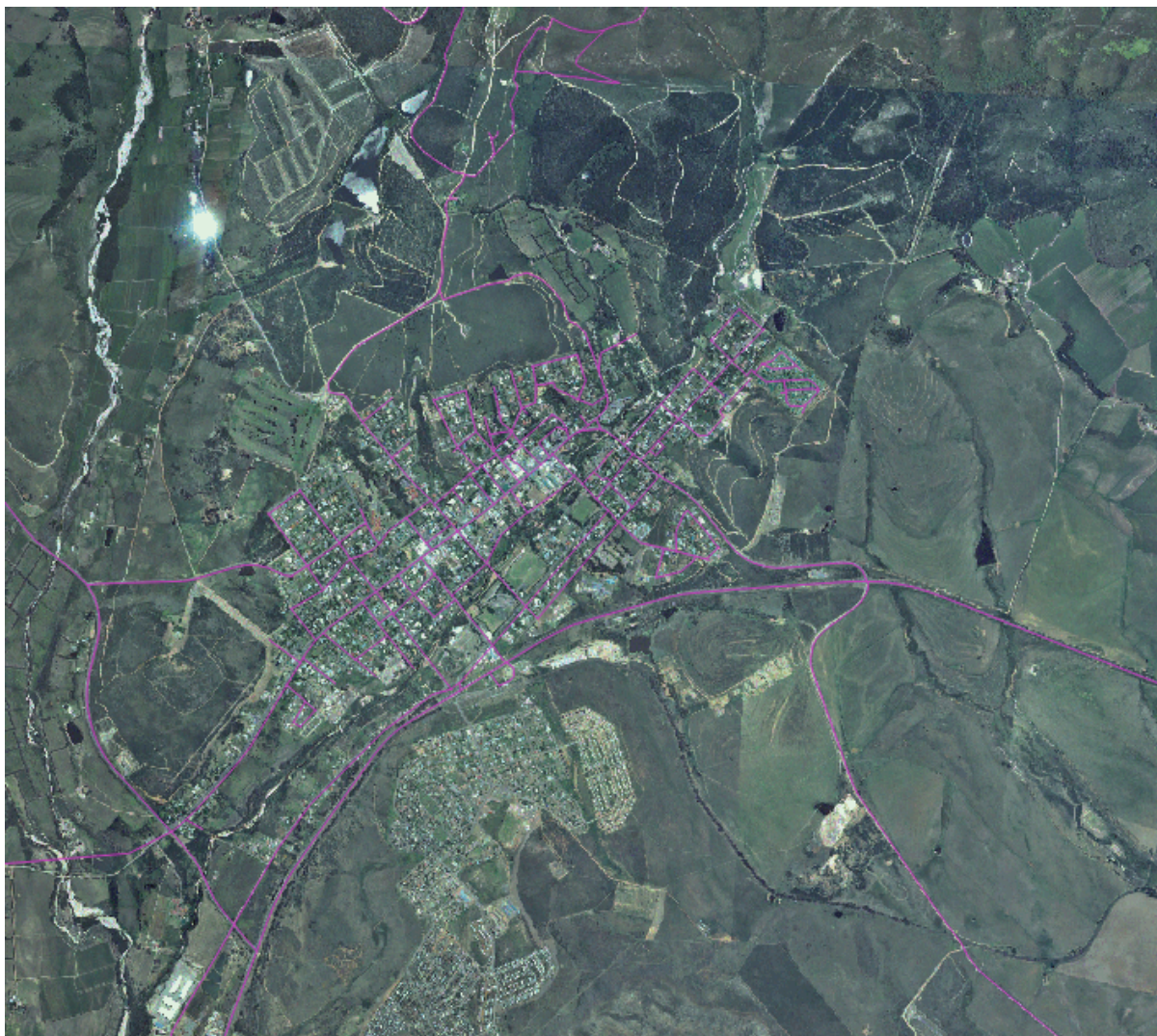
The *Load Raster Layer* dialog will open. The data for this project is in `exercise_data/raster`.

- Either load them all in separately, or hold down `ctrl` and click on all four of them in turn, then open them at the same time.

The first thing you'll notice is that nothing seems to be happening in your map. Are the rasters not loading? Well, there they are in the *Layers list*, so obviously they did load. The problem is that they're not in the same projection. Luckily, we've already seen what to do in this situation.

- Select *Project* → *Project Properties* in the menu:
- Select *CRS* tab in the menu:
- Enable “on the fly” reprojection.
- Set it to the same projection as the rest of your data (WGS 84 / UTM zone 33S).
- Click *OK*.

The rasters should fit nicely:



There we have it - four aerial photographs covering our whole study area.

8.1.2 Follow Along: Create a Virtual Raster


Now as you can see from this, your solution layer lies across all four photographs. What this means is that you're going to have to work with four rasters all the time. That's not ideal; it would be better to have one file for one (composite) image, right?

Luckily, QGIS allows you to do exactly this, and without needing to actually create a new raster file, which could take up a lot of space. Instead, 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's a way to organize your existing rasters into one catalog: one file for easy access.

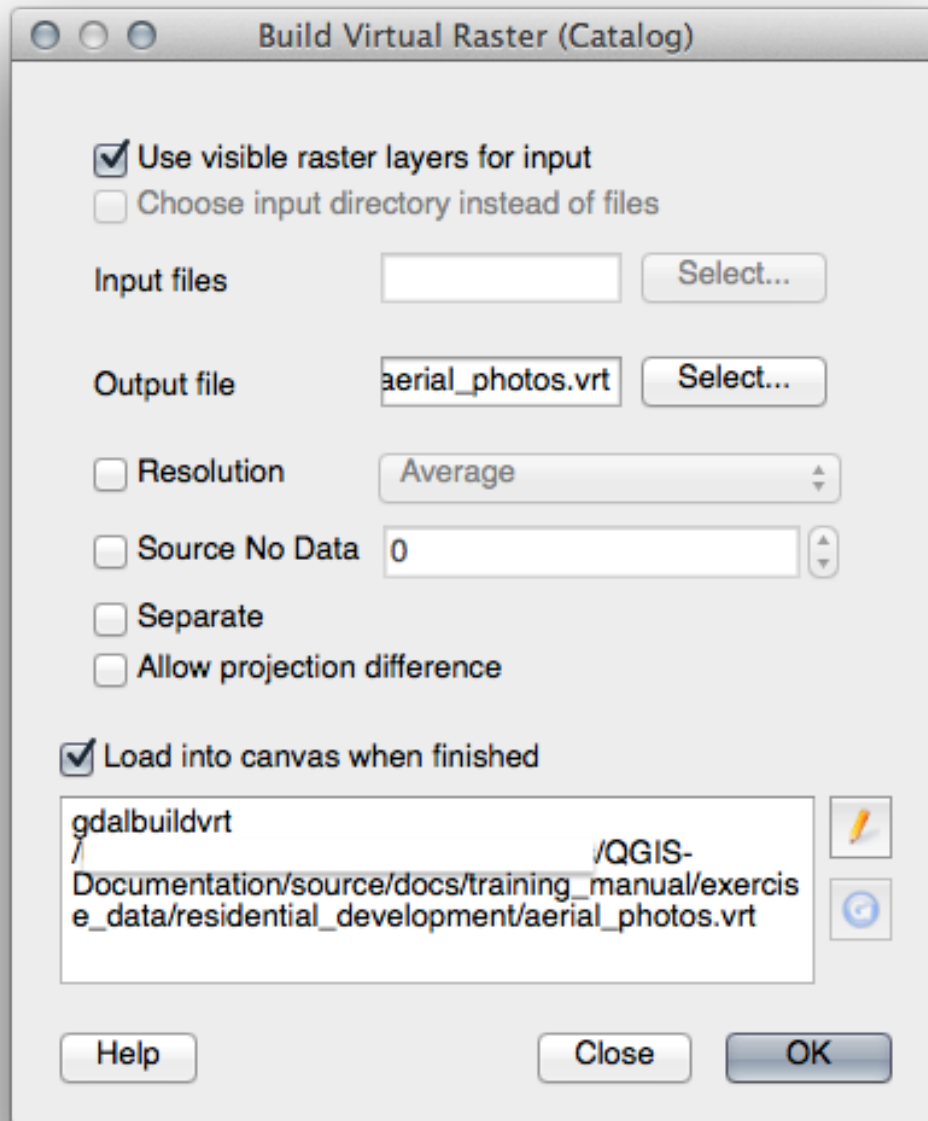
To make a catalog:

- Click on the menu item *Raster* → *Miscellaneous* → *Build Virtual Raster (Catalog)*.
- In the dialog that appears, check the box next to *Use visible raster layers for input*.
- Enter `exercise_data/residential_development` as the output location.
- Enter `aerial_photos.vrt` as the file name.
- Check the *Load into canvas when finished* button.

Notice the text field below. What this dialog is actually doing is that it's writing that text for you. It's a long command that QGIS is going to run.

Bemerkung:  Keep in mind that the command text is editable, so you can customize the command further if preferred. Search online for the initial command (in this case, `gdalbuildvrt`) for help on the syntax.

- Click *OK* to run the command.



It may take a while to complete. When it's done, it will tell you so with a message box.

- Click *OK* to chase the message away.
- Click *Close* on the *Build Virtual Raster (Catalog)* dialog. (Don't click *OK* again, otherwise it's going to start running that command again.)

- You can now remove the original four rasters from the *Layers list*.
- If necessary, click and drag the new *aerial_photos* raster catalog layer to the bottom of the *Layers list* so that the other activated layers become visible.

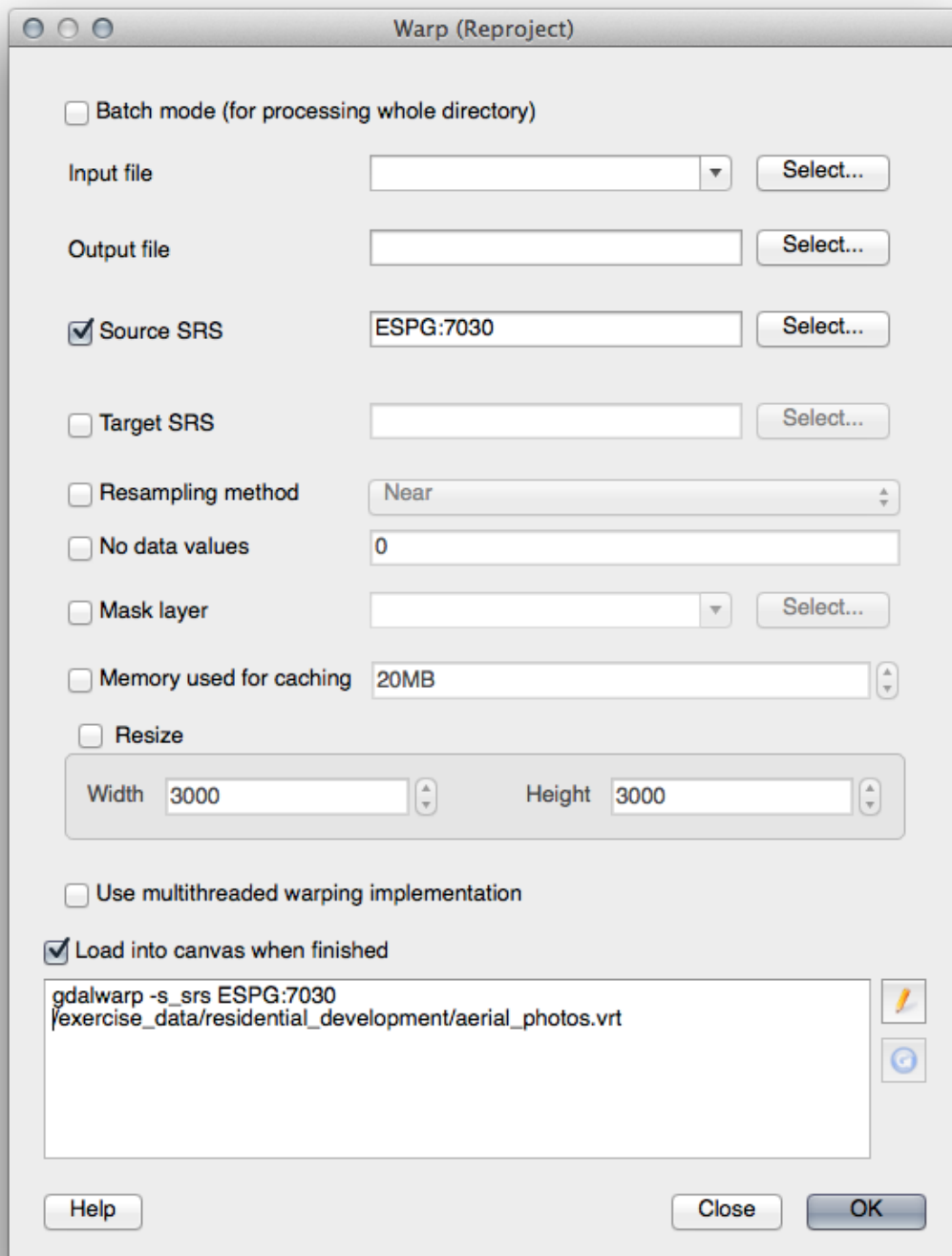
8.1.3 Transforming Raster Data

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

- Click on the menu item *Raster* → *Projections* → *Warp (Reproject)*.

Note that this tool features a handy batch option for reprojecting the contents of whole directories. You can also reproject virtual rasters (catalogs), as well as enabling a multithreaded processing mode.

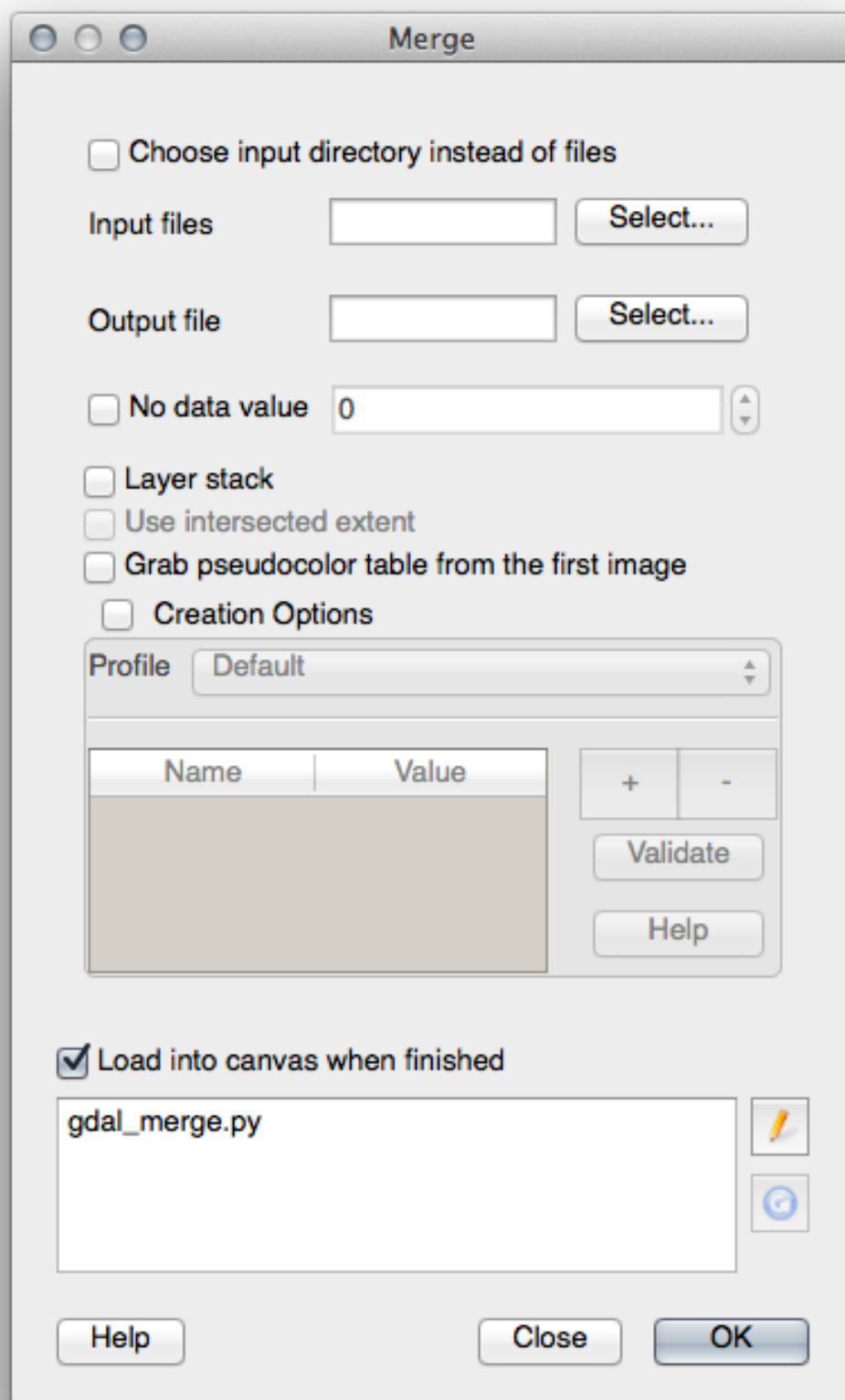


Merging rasters

- Click on the menu item *Raster* → *Miscellaneous* → *Merge*.

You can choose to process entire directories instead of single files, giving you a very useful built-in batch processing capability. You can specify a virtual raster as input file, too, and all of the rasters that it consists of will be processed.

You can also add your own command line options using the *Creation Options* checkbox and list. This only applies if you have knowledge of the GDAL library's operation.



8.1.4 In Conclusion

QGIS makes it easy to include raster data into your existing projects.

8.1.5 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.

8.2 Lesson: Changing Raster Symbolology

Not all raster data consists of aerial photographs. There are many other forms of raster data, and in many of those cases, it's essential to symbolize the data properly so that it becomes properly visible and useful.

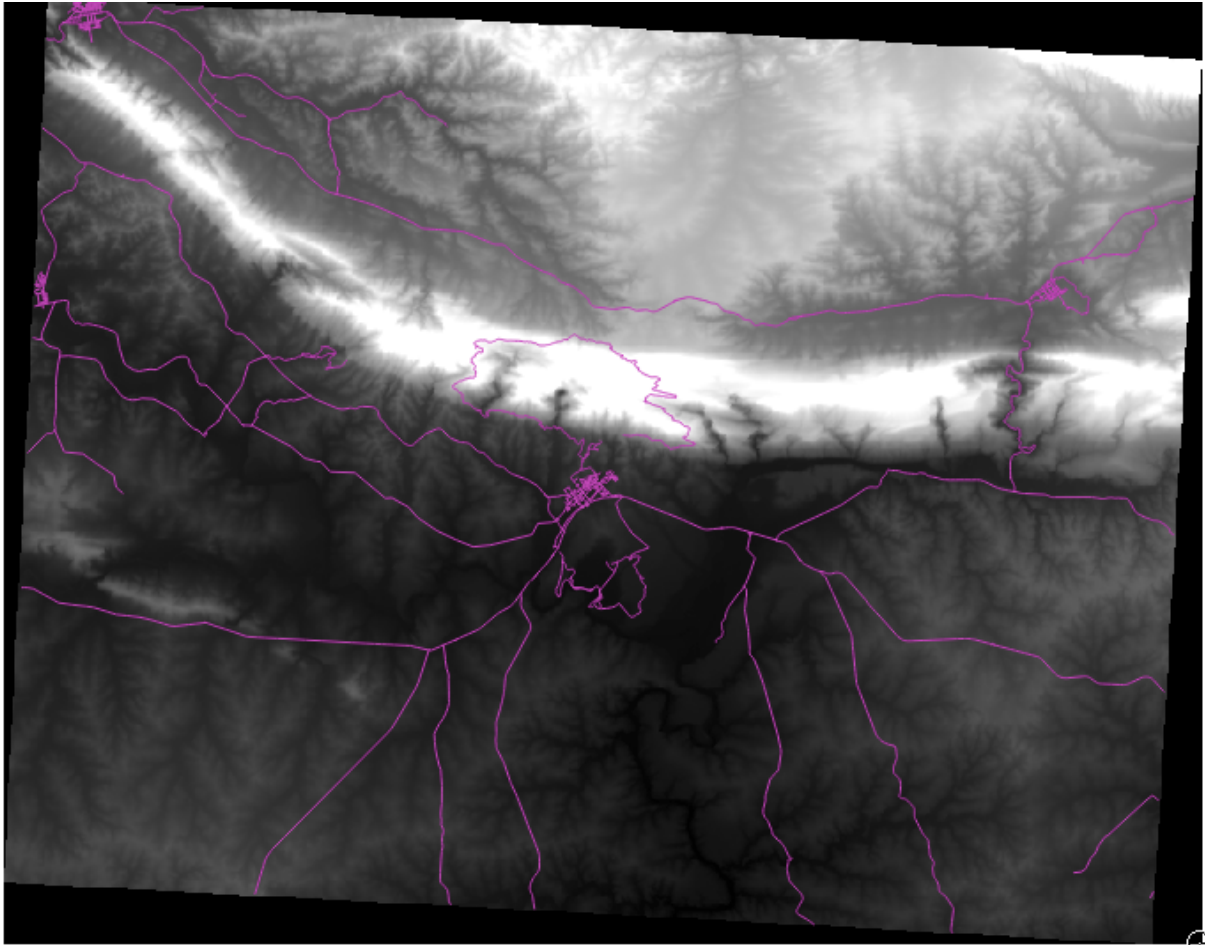
The goal for this lesson: To change the symbology for a raster layer.

8.2.1 Try Yourself

- Start with the current map from the previous exercise `analysis.qgs`.
- Use the *Add Raster Layer* button to load the new raster dataset.
- Load the dataset `srtm_41_19.tif`, found under the directory `exercise_data/raster/SRTM/`.
- Once it appears in the *Layers list*, rename it to `DEM`.
- Zoom to the extent of this layer by right-clicking on it in the Layer List and selecting *Zoom to Layer Extent*.

This dataset is a *Digital Elevation Model (DEM)*. It's a map of the elevation (altitude) of the terrain, allowing us to see where the mountains and valleys are, for example.

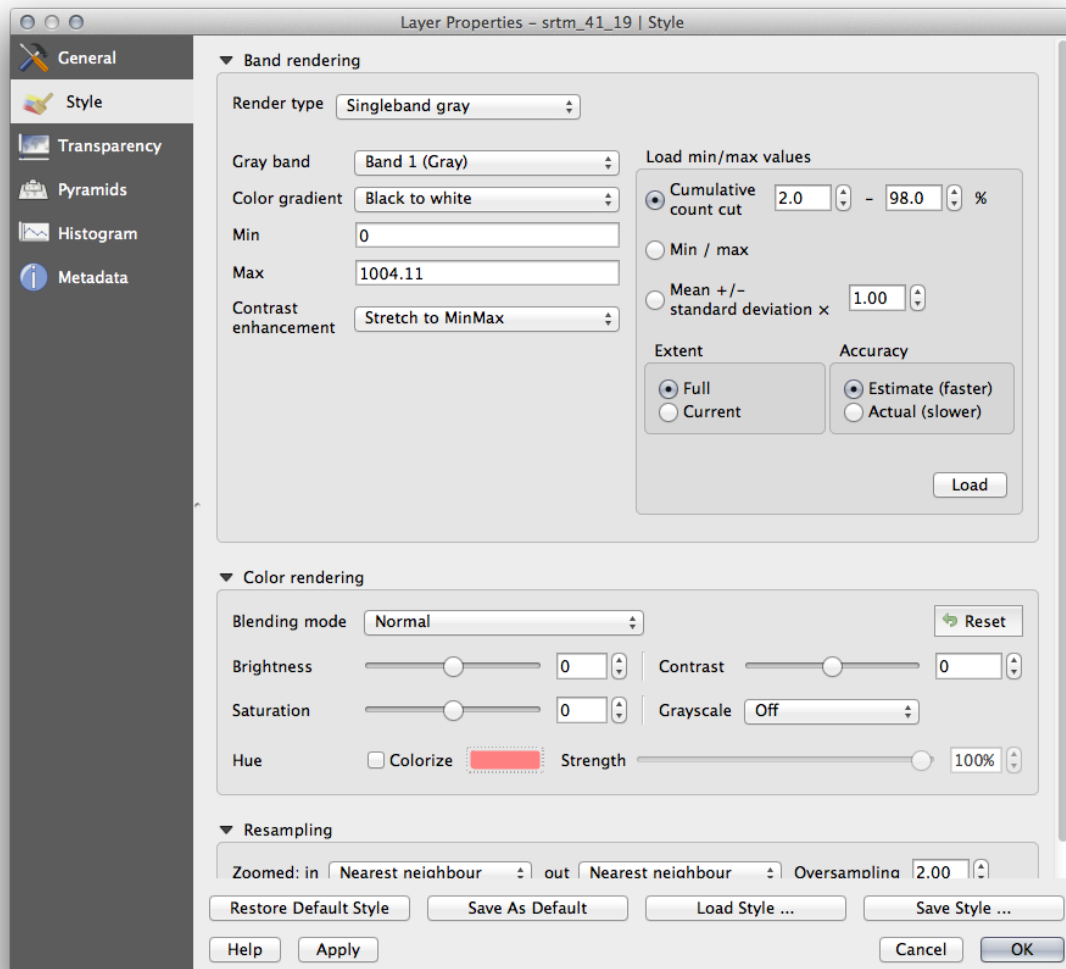
Once it's loaded, you'll notice that it's a basic stretched grayscale representation of the DEM. It's seen here with the vector layers on top:



QGIS has automatically applied a stretch to the image for visualization purposes, and we will learn more about how this works as we continue.

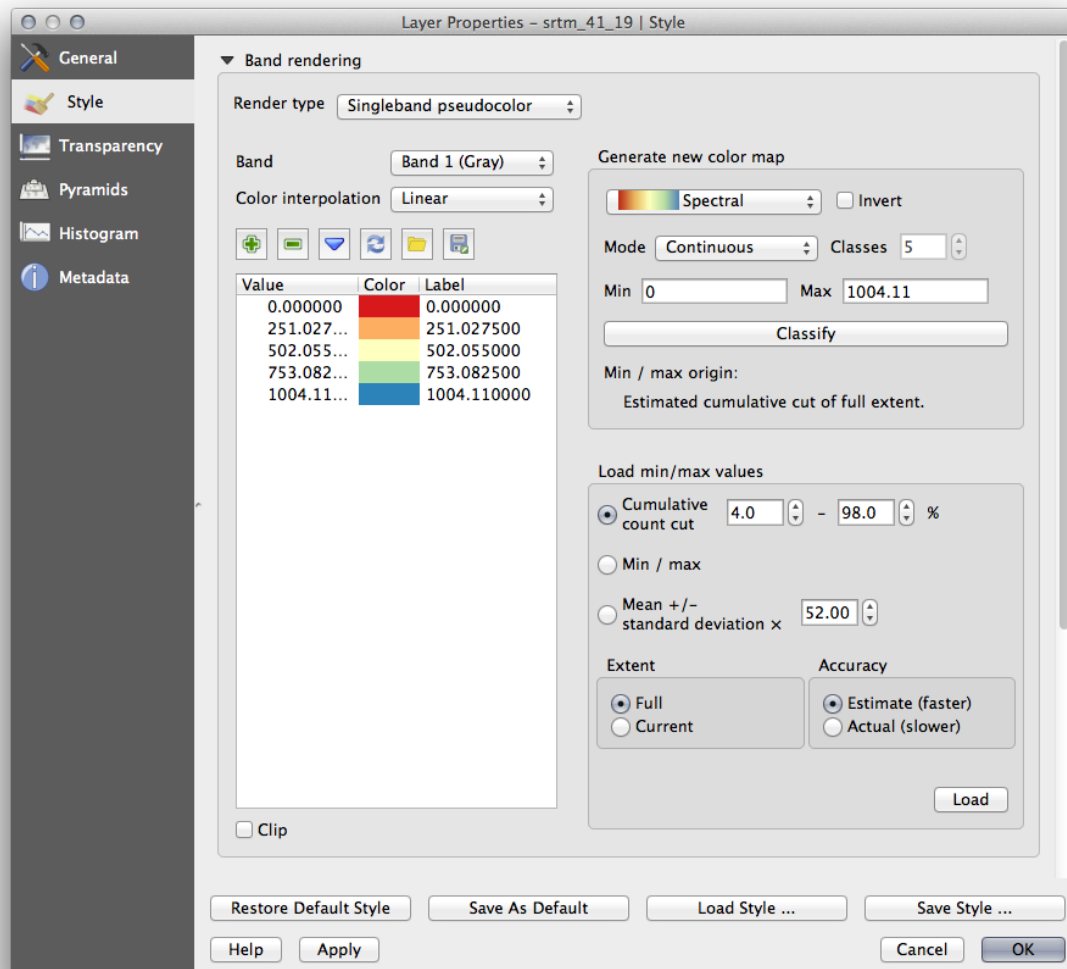
8.2.2 Follow Along: Changing Raster Layer Symbology

- Open the *Layer Properties* dialog for the *SRTM* layer by right-clicking on the layer in the Layer tree and selecting *Properties* option.
- Switch to the *Style* tab.

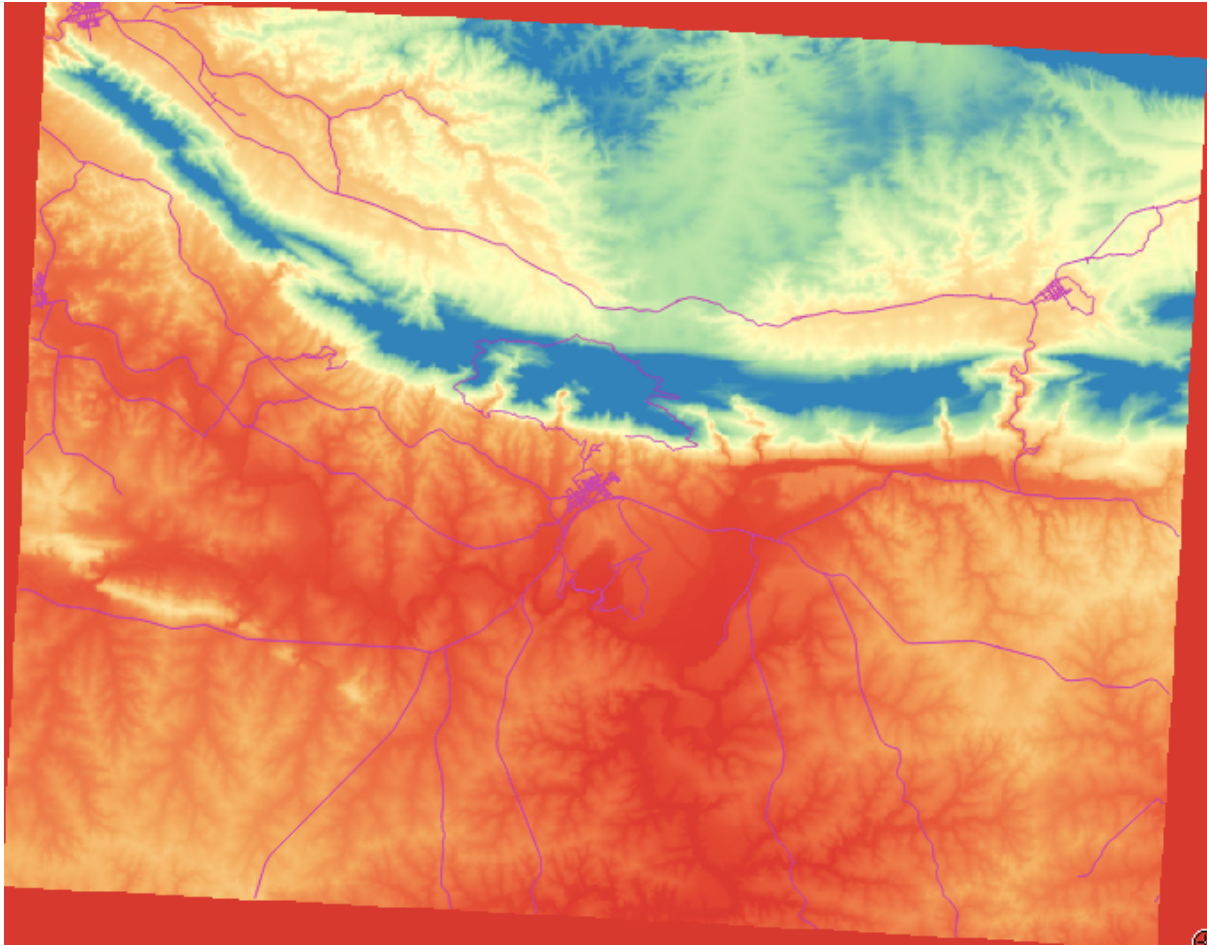


These are the current settings that QGIS applied for us by default. Its just one way to look at a DEM, so lets explore some others.

- Change the *Render type* to *Singleband pseudocolor*, and use the default options presented.
- Click the *Classify* button to generate a new color classification, and click *OK* 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, but maybe we don't want to symbolize it using these colors.

- Open *Layer Properties* dialog again.
- Switch the *Render Type* back to *Singleband gray*.
- Click *OK* to apply this setting to the raster.

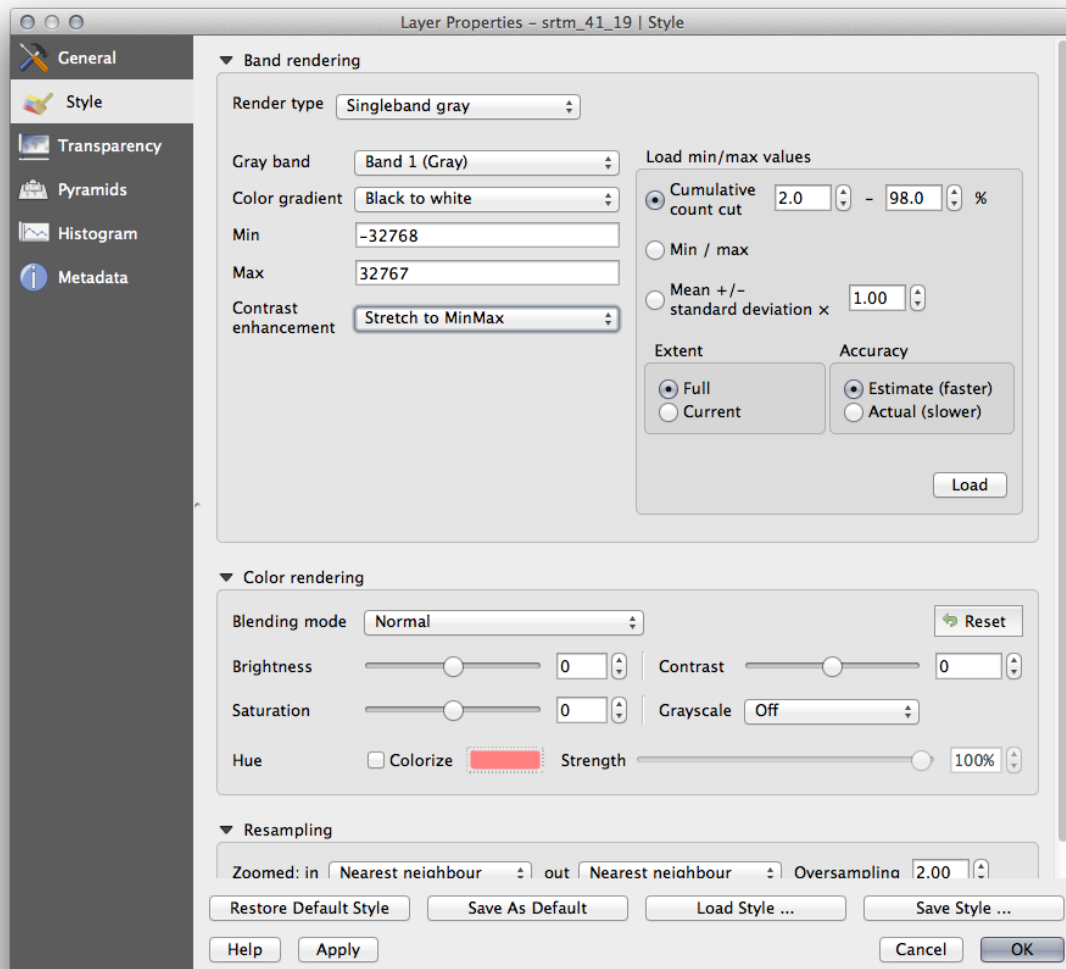
You will now see a totally gray rectangle that isn't very useful at all.



This is because we have lost the default settings which “stretch” the color values to show them contrast.

Let’s tell QGIS to again “stretch” the color values based on the range of data in the DEM. This will make QGIS use all of the available colors (in *Grayscale*, this is black, white and all shades of gray in between).

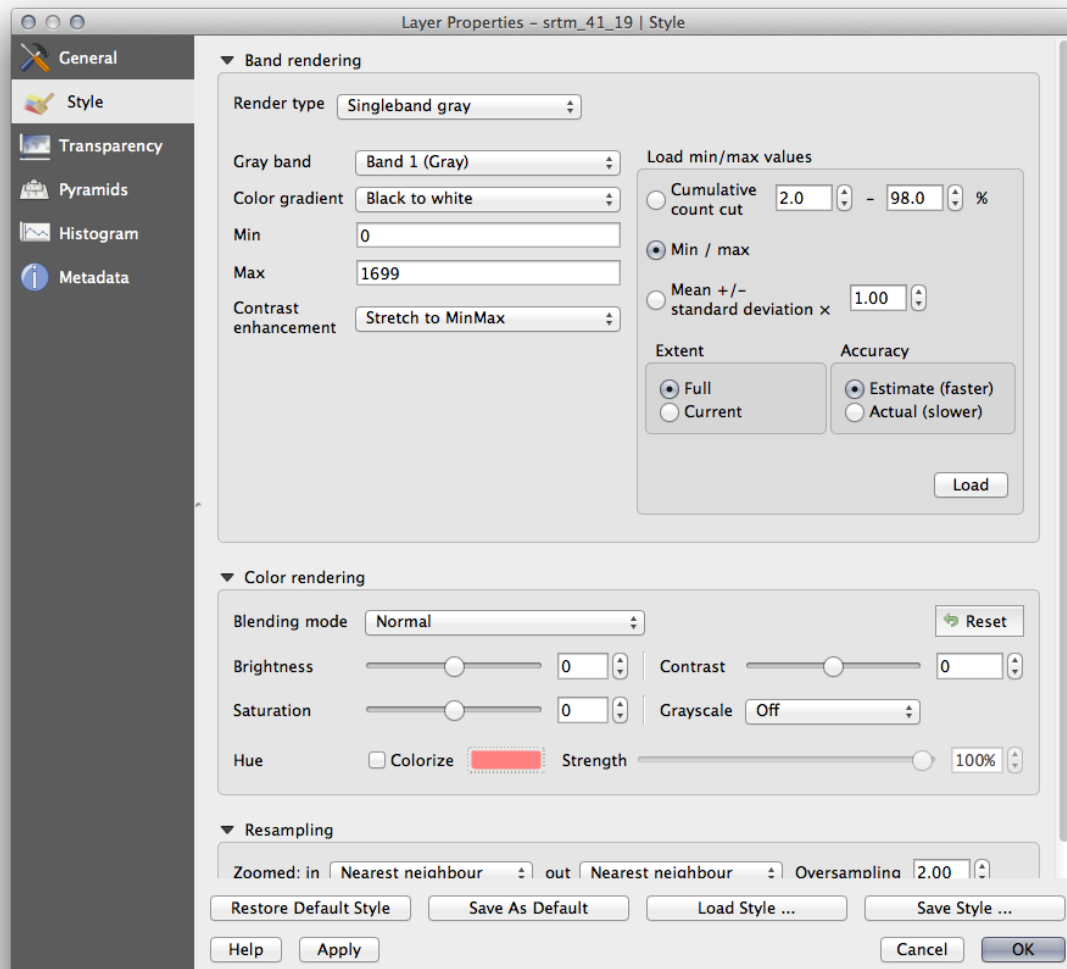
- Specify the *Min* and *Max* values as shown below.
- Set the value *Contrast enhancement* to *Stretch To MinMax*:



But what are the minimum and maximum values that should be used for the stretch? The ones that are currently under *Min* and *Max* values are the same values that just gave us a gray rectangle before. Instead, we should be using the minimum and maximum values that are actually in the image, right? Fortunately, you can determine those values easily by loading the minimum and maximum values of the raster.

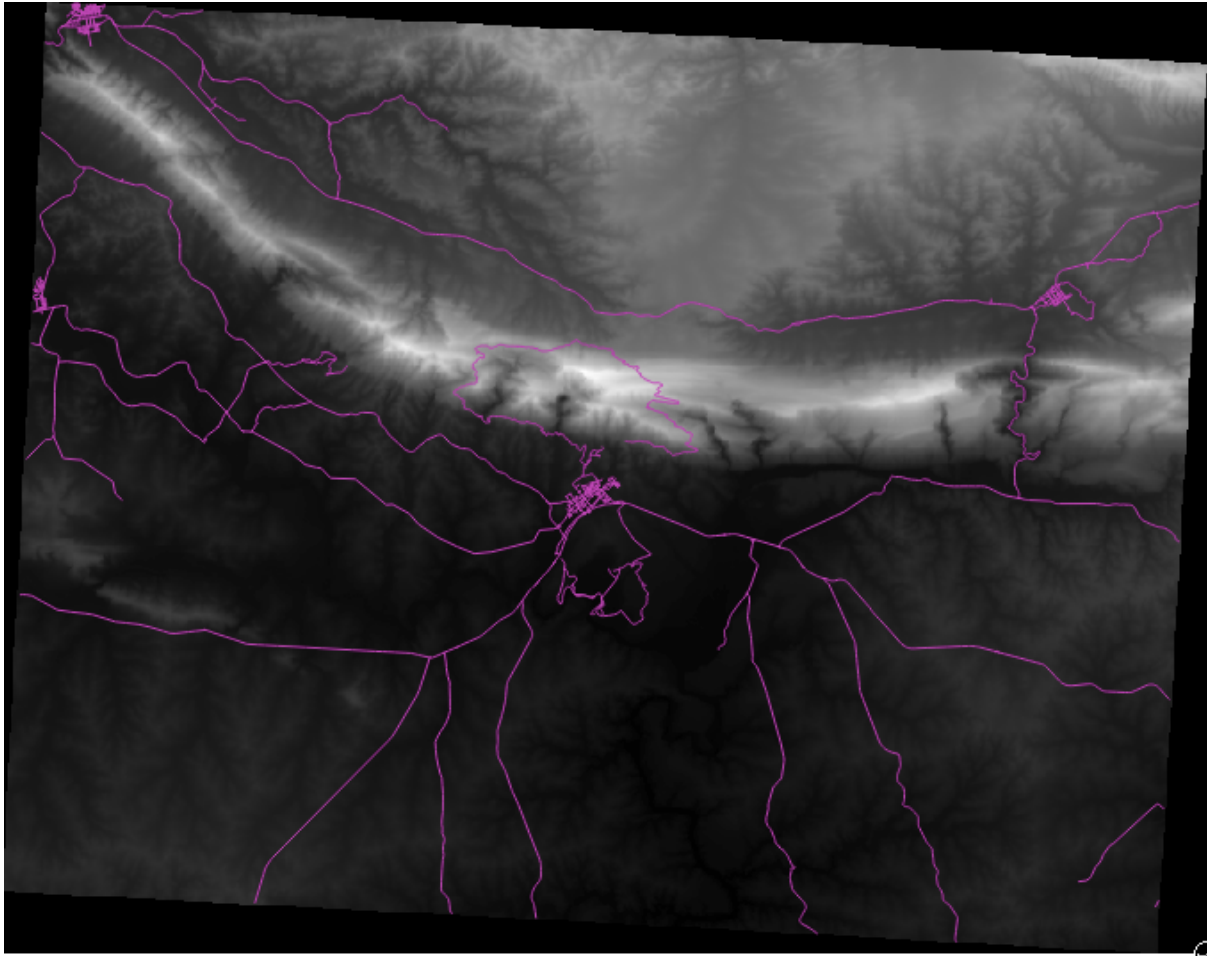
- Under *Load min / max values*, select *Min / Max* option.
- Click the *Load* button:

Notice how the *Custom min / max values* have changed to reflect the actual values in our DEM:



- Click **OK** to apply these settings to the image.

You'll now see that the values of the raster are again properly displayed, with the darker colors representing valleys and the lighter ones, mountains:



But isn't there a better or easier way?

Yes, there is. Now that you understand what needs to be done, you'll be glad to know that there's a tool for doing all of this easily.

- Remove the current DEM from the *Layers list*.
- Load the raster in again, renaming it to DEM as before. It's a gray rectangle again...
- Enable the tool you'll need by enabling *View → Toolbars → Raster*. These icons will appear in the interface:



The third button from the left *Local Histogram Stretch* will automatically stretch the minimum and maximum values to give you the best contrast in the local area that you're zoomed into. It's useful for large datasets. The button on the left *Local Cumulative Cut Stretch ...* will stretch the minimum and maximum values to constant values across the whole image.

- Click the fourth button from the left (*Stretch Histogram to Full Dataset*). You'll see the data is now correctly represented as before.

You can try the other buttons in this toolbar and see how they alter the stretch of the image when zoomed in to local areas or when fully zoomed out.

8.2.3 In Conclusion

These are only the basic functions to get you started with raster symbology. QGIS also allows you many other options, such as symbolizing a layer using standard deviations, or representing different bands with different colors in a multispectral image.

8.2.4 Reference

The SRTM dataset was obtained from <http://srtm.csi.cgiar.org/>

8.2.5 What's Next?

Now that we can see our data displayed properly, let's investigate how we can analyze it further.

8.3 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.

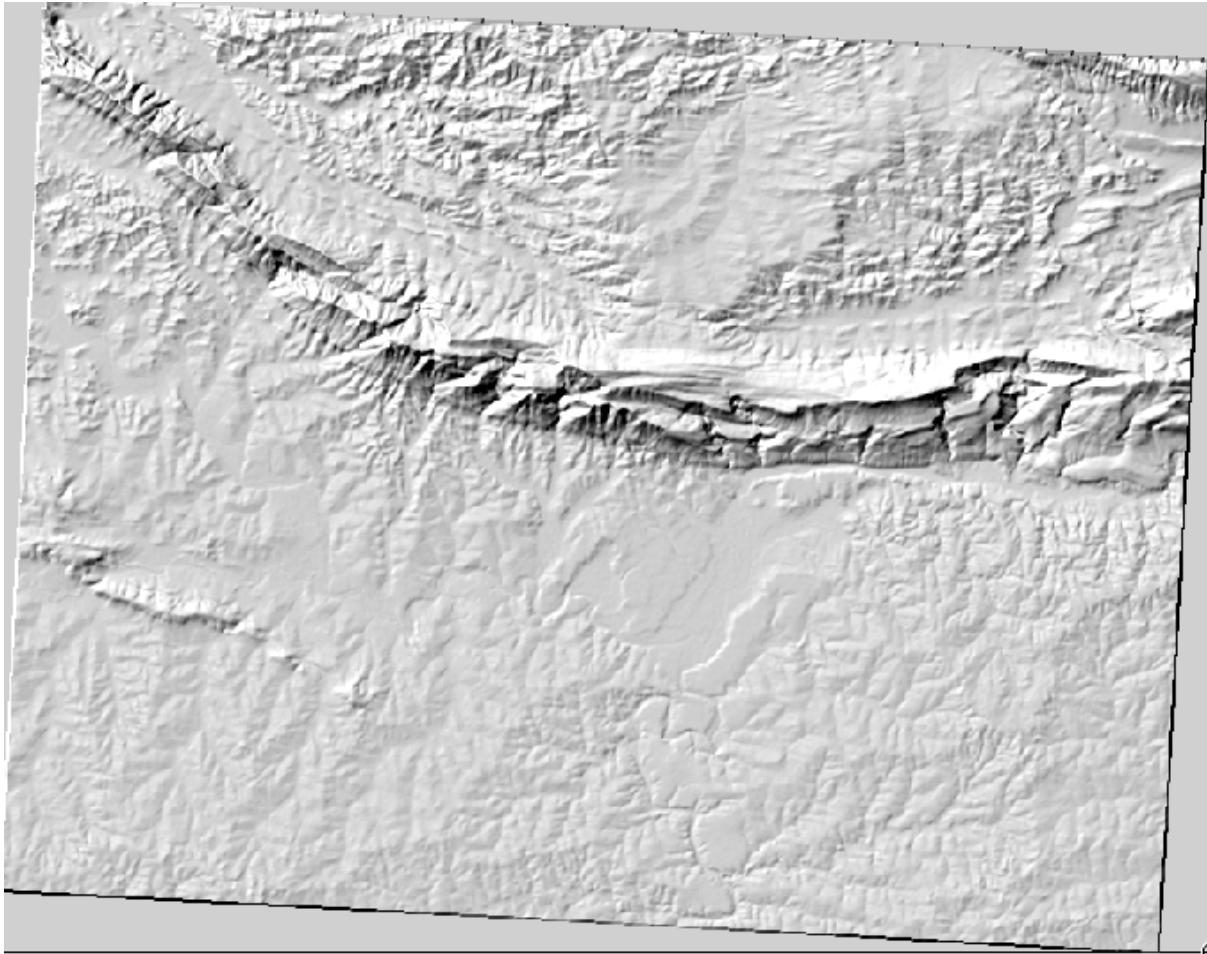
8.3.1 Follow Along: Calculating a Hillshade

The DEM you have on your map right now does show 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 look at 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.

To work with DEMs, you should use QGIS' all-in-one *DEM (Terrain models)* analysis tool.

- Click on the menu item *Raster → Analysis → DEM (Terrain models)*.
- In the dialog that appears, ensure that the *Input file* is the *DEM* layer.
- Set the *Output file* to `hillshade.tif` in the directory `exercise_data/residential_development`.
- Also make sure that the *Mode* option has *Hillshade* selected.
- Check the box next to *Load into canvas when finished*.
- You may leave all the other options unchanged.
- Click *OK* to generate the hillshade.
- When it tells you that processing is completed, click *OK* on the message to get rid of it.
- Click *Close* on the main *DEM (Terrain models)* dialog.

You will now have a new layer called *hillshade* that looks like this:

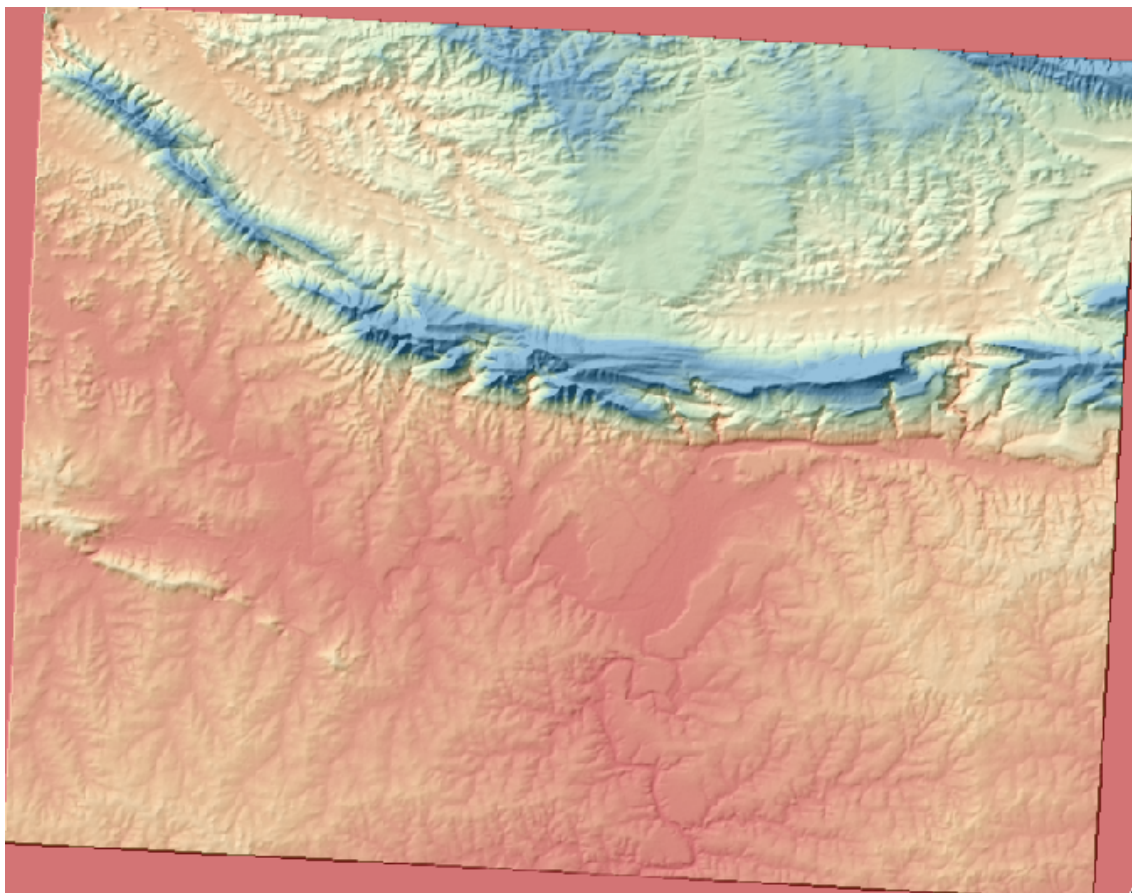


That looks nice and 3D, but can we improve on this? On its own, the hillshade looks like a plaster cast. Can't we use it together with our other, more colorful rasters somehow? Of course we can, by using the hillshade as an overlay.

8.3.2 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 be mostly transparent.

- Change the symbology of the original *DEM* to use the *Pseudocolor* scheme as in the previous exercise.
- Hide all the layers except the *DEM* and *hillshade* layers.
- Click and drag the *DEM* to be beneath the *hillshade* layer in the *Layers list*.
- Set the *hillshade* layer to be transparent by opening its *Layer Properties* and go to the *Transparency* tab.
- Set the *Global transparency* to 50%:
- Click *OK* on the *Layer Properties* dialog. You'll get a result like this:



- Switch the *hillshade* layer off and back on in the *Layers list* 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 your map when you are done.

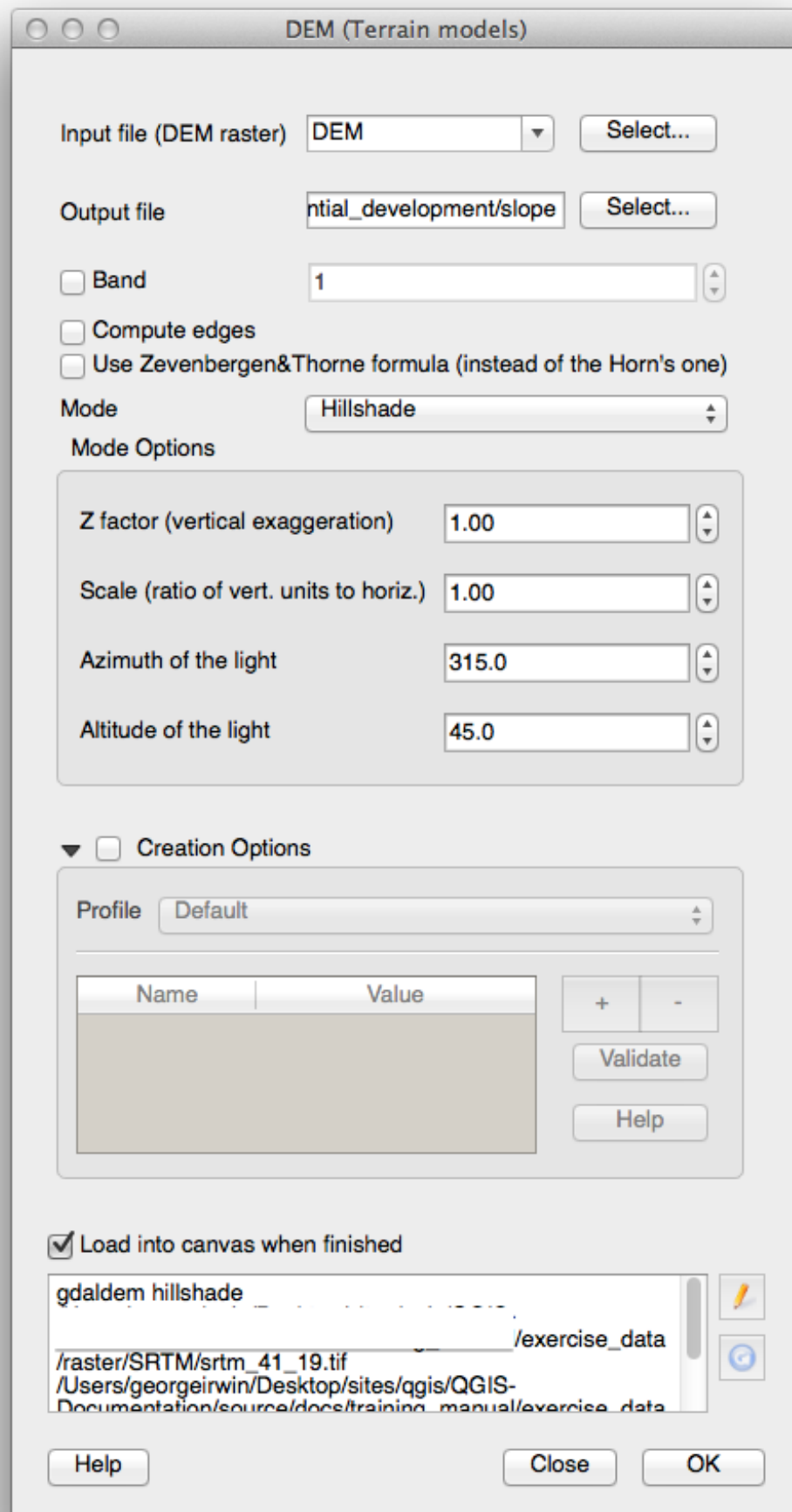
Bemerkung: For the next two exercises, please use a new map. Load only the DEM raster dataset into it (`exercise_data/raster/SRTM/srtm_41_19.tif`). This is to simplify matters while you're working with the raster analysis tools. Save the map as `exercise_data/raster_analysis.qgs`.

8.3.3 Follow Along: Calculating the Slope

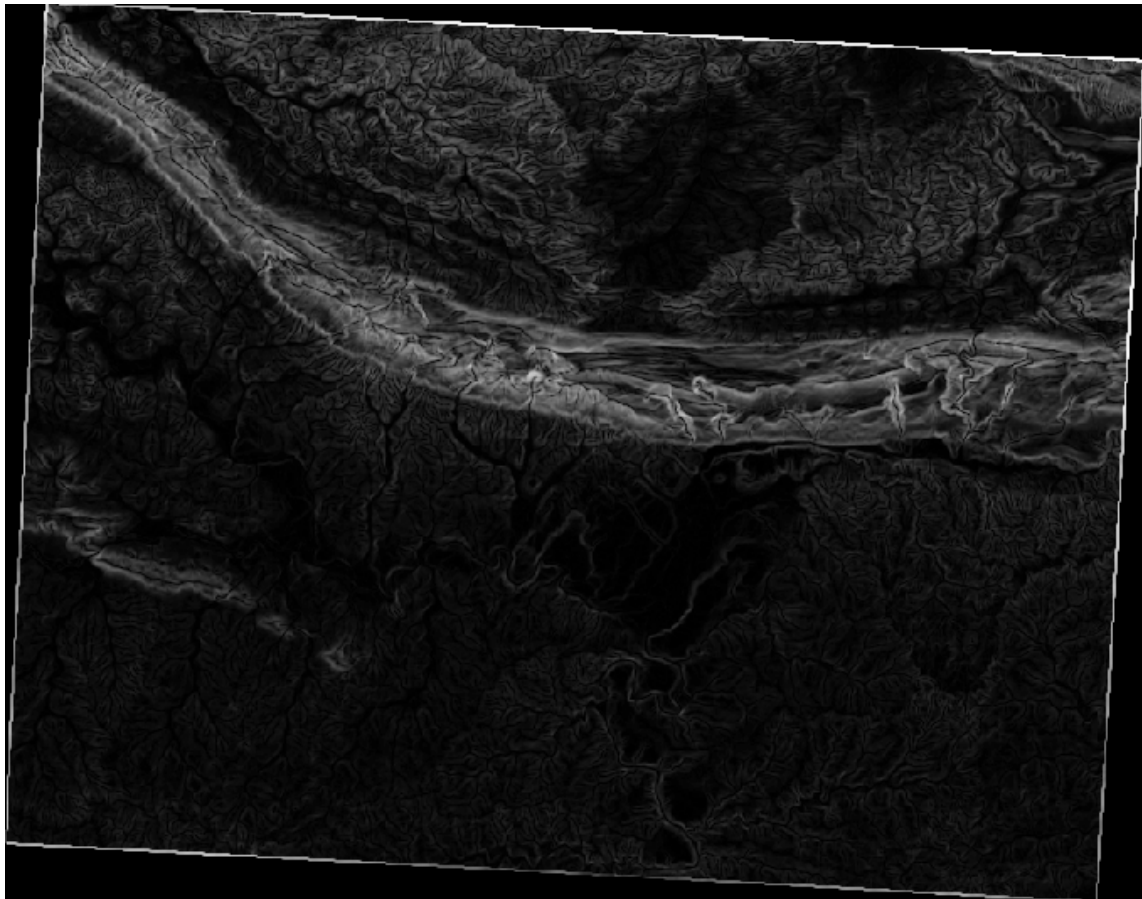
Another useful thing to know about the terrain is how steep it is. If, for example, you want to build houses on the land there, then you need land that is relatively flat.

To do this, you need to use the *Slope* mode of the *DEM (Terrain models)* tool.

- Open the tool as before.
- Select the *Mode* option *Slope*:



- Set the save location to `exercise_data/residential_development/slope.tif`
- Enable the *Load into canvas...* checkbox.
- Click *OK* and close the dialogs when processing is complete, and click *Close* to close the dialog. You'll see a new raster loaded into your map.
- With the new raster selected in the *Layers list*, click the *Stretch Histogram to Full Dataset* button. Now you'll see the slope of the terrain, with black pixels being flat terrain and white pixels, steep terrain:



8.3.4 Try Yourself calculating the aspect

The *aspect* of terrain refers to the direction it's facing in. 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* mode of the *DEM (Terrain models)* tool to calculate the aspect of the terrain.

Check your results

8.3.5 Follow Along: Using the Raster Calculator

Think back to the estate agent problem, which we last addressed in the *Vector Analysis* lesson. Let's 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, and with a slope of less than five degrees. But if the slope is less than 2 degrees, then the aspect doesn't matter.

Fortunately, you already have rasters showing you the slope as well as the aspect, but you have no way of knowing where both conditions are satisfied at once. How could this analysis be done?

The answer lies with the *Raster calculator*.

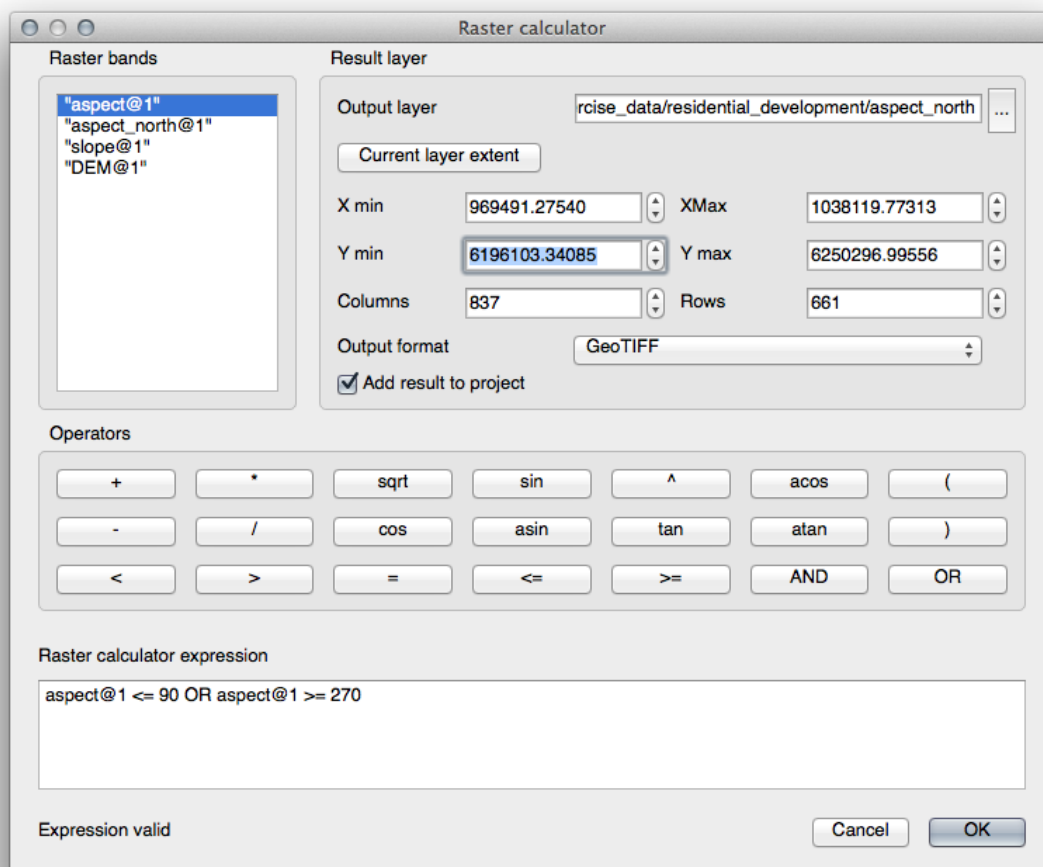
- Click on *Raster > Raster calculator...* to start this tool.
- To make use of the *aspect* dataset, double-click on the item *aspect@1* in the *Raster bands* list on the left. It will appear in the *Raster calculator expression* text field below.

North is at 0 (zero) degrees, so for the terrain to face north, its aspect needs to be greater than 270 degrees and less than 90 degrees.

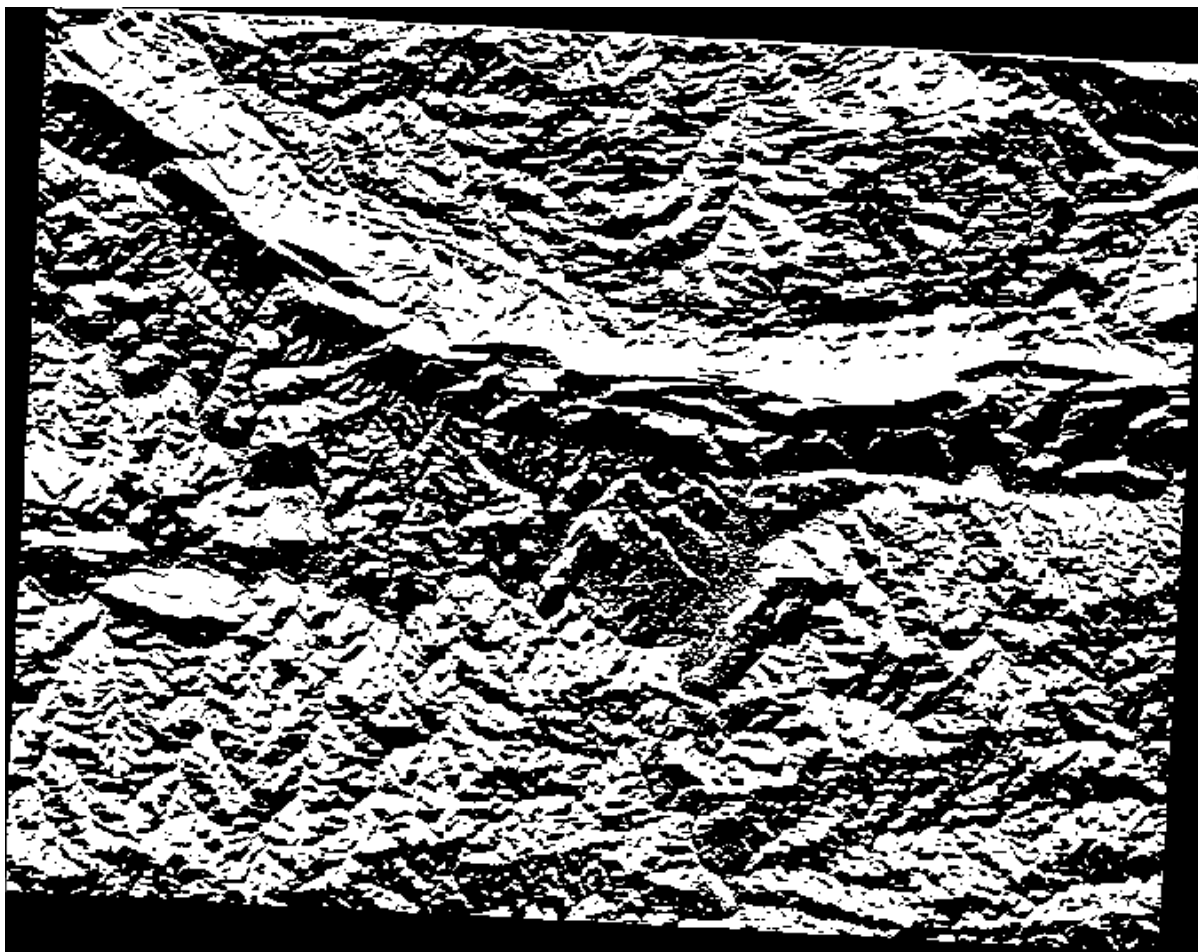
- In the *Raster calculator expression* field, enter this expression:

```
aspect@1 <= 90 OR aspect@1 >= 270
```

- Set the output file to *aspect_north.tif* in the directory *exercise_data/residential_development/*.
- Ensure that the box *Add result to project* is checked.
- Click *OK* to begin processing.



Your result will be this:



8.3.6 Try Yourself

Now that you've done the aspect, create two separate new analyses of the *DEM* layer.

- The first will be to identify all 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/residential_development/` as `slope_lte2.tif` and `slope_lte5.tif`.

Check your results

8.3.7 Follow Along: Combining Raster Analysis Results

Now you have three new analysis rasters of the *DEM* layer:

- *aspect_north*: the terrain faces north
- *slope_lte2*: the slope is at or below 2 degrees
- *slope_lte5*: the slope is at or below 5 degrees

Where the conditions of these layers are met, they are equal to 1. Elsewhere, they are equal to 0. Therefore, if you multiply one of these rasters by another one, you will get the areas where both of them are equal to 1.

The conditions to be met are: at or below 5 degrees of slope, the terrain must face north; but at or below 2 degrees of slope, the direction that the terrain faces in does not matter.

Therefore, you need to find areas where the slope is at or below 5 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:

- Open your *Raster calculator* again.
- Use the *Raster bands* list, the *Operators* buttons, and your keyboard to build this expression in the *Raster calculator expression* text area:

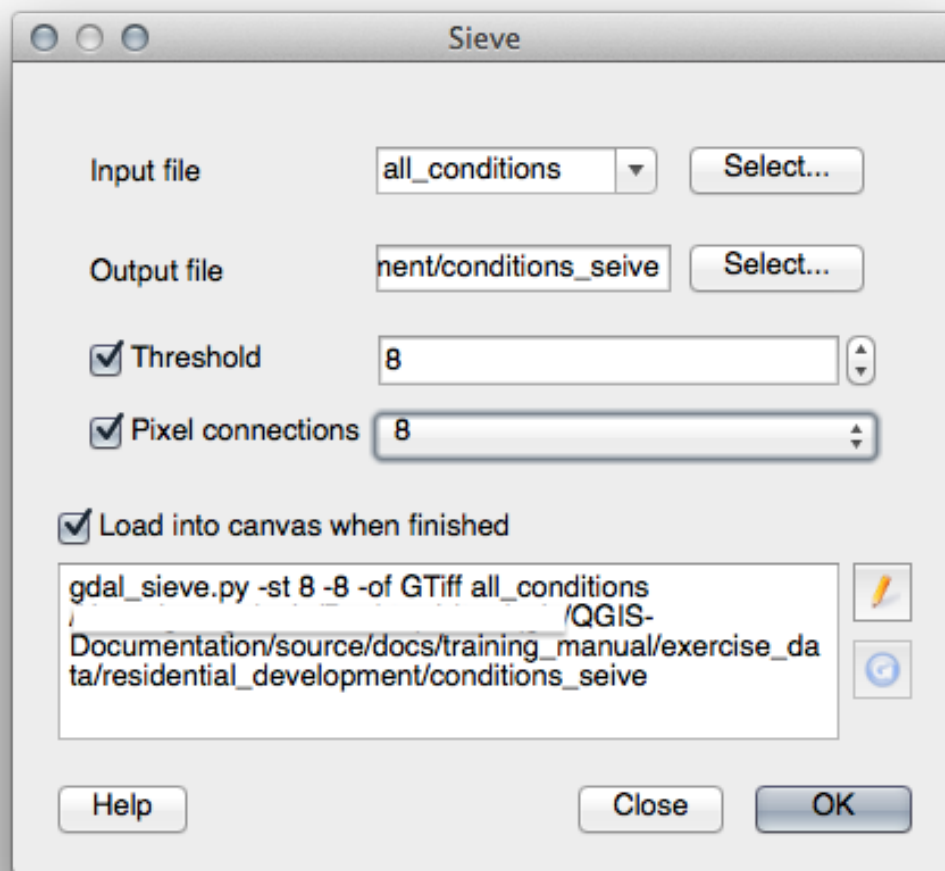
```
( aspect_north@1 = 1 AND slope_lte5@1 = 1 ) OR slope_lte2@1 = 1
```
- Save the output under `exercise_data/residential_development/` as `all_conditions.tif`.
- Click *OK* on the *Raster calculator*. Your results:



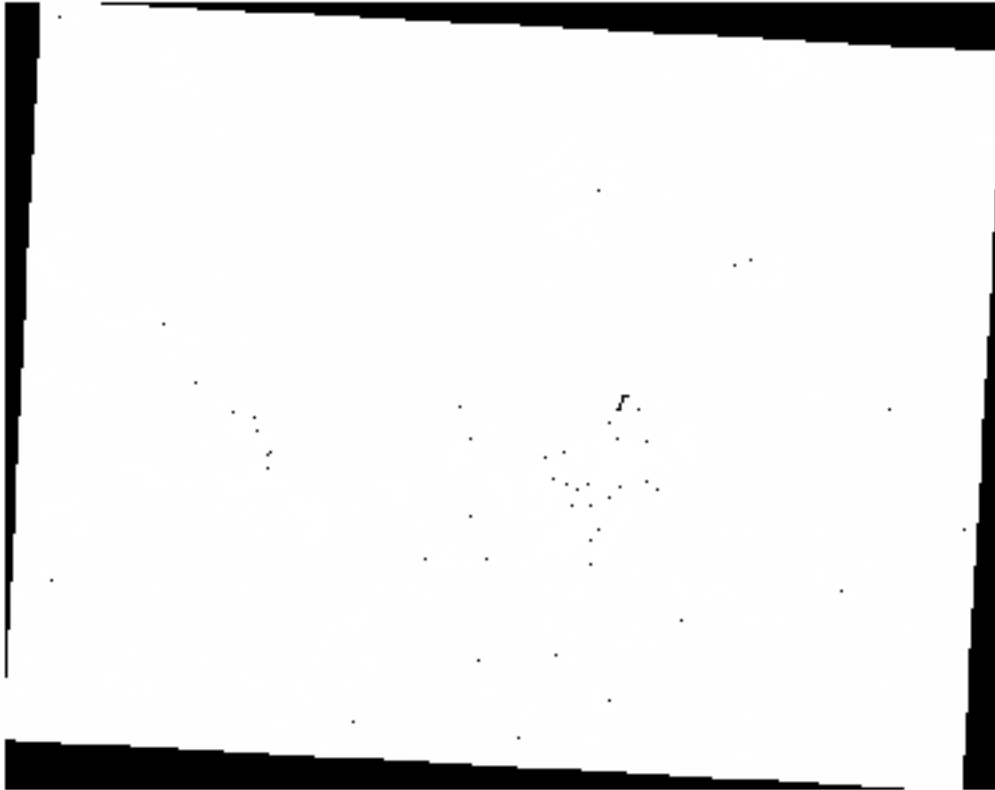
8.3.8 Follow Along: Simplifying the Raster

As you can see from the image above, the combined analysis has left us with many, very small areas where the conditions are met. But these aren't really useful for our analysis, since they're too small to build anything on. Let's get rid of all these tiny unusable areas.

- Open the *Sieve* tool (*Raster* → *Analysis* → *Sieve*).
- Set the *Input file* to `all_conditions`, and the *Output file* to `all_conditions_sieve.tif` (under `exercise_data/residential_development/`).
- Set both the *Threshold* and *Pixel connections* values to 8, then run the tool.

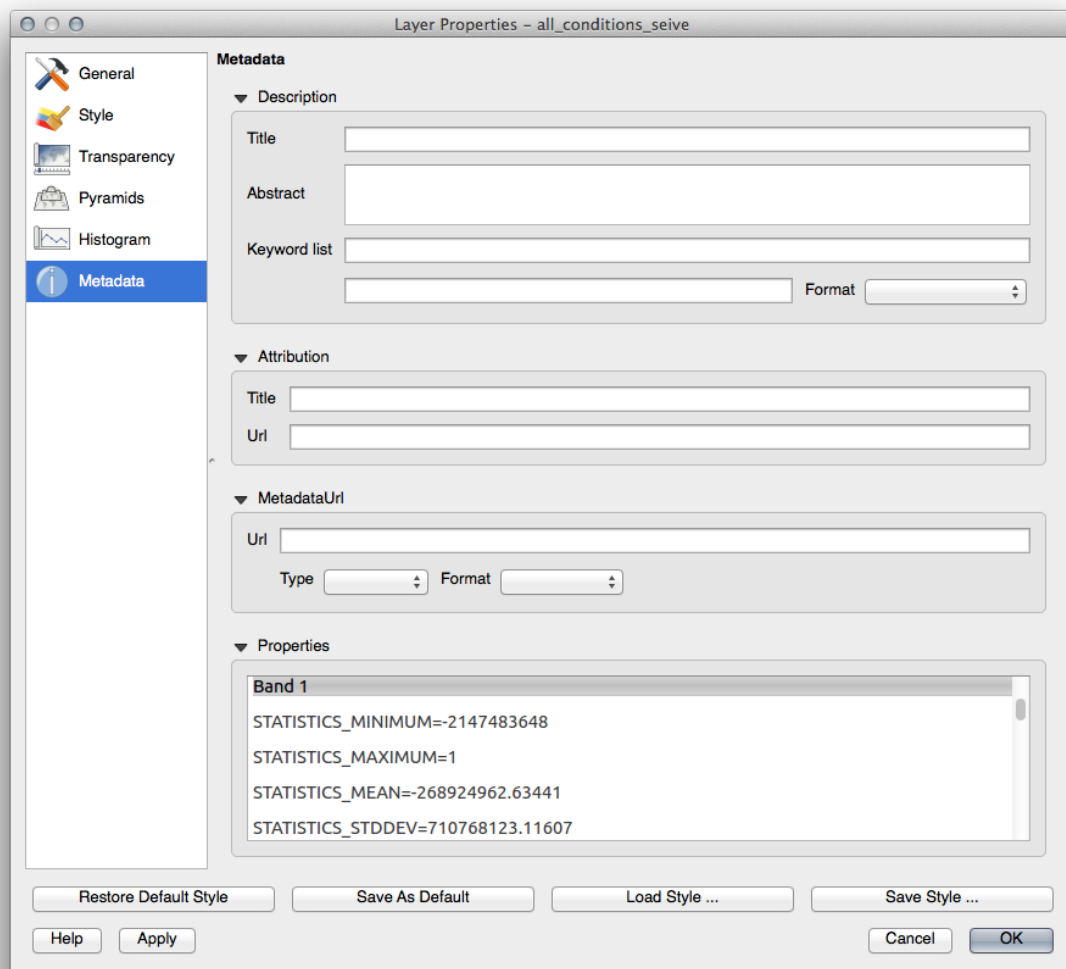


Once processing is done, the new layer will load into the canvas. But when you try to use the histogram stretch tool to view the data, this happens:



What's going on? The answer lies in the new raster file's metadata.

- View the metadata under the *Metadata* tab of the *Layer Properties* dialog. Look in the *Properties* section at the bottom.



Whereas this raster, like the one it's derived from, should only feature the values 1 and 0, it has the `STATISTICS_MINIMUM` value of a very large negative number. Investigation of the data shows that this number acts as a null value. Since we're only after areas that weren't filtered out, let's set these null values to zero.

- Open the *Raster Calculator* again, and build this expression:

```
(all_conditions_sieve@1 <= 0) = 0
```

This will maintain all existing zero values, while also setting the negative numbers to zero; which will leave all the areas with value 1 intact.

- Save the output under `exercise_data/residential_development/` 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.

8.3.9 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.

8.3.10 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.

Module: Completing the Analysis

You now have two halves of an analysis: a vector and a raster part. In this module, you'll see how to combine them. You will conclude the analysis and present the final results.

9.1 Lesson: Raster to Vector Conversion

Converting between raster and vector formats allows you to make use of both raster and vector data when solving a GIS problem, as well as using the various analysis methods unique to these two forms of geographic data. This increases the flexibility you have when considering data sources and processing methods for solving a GIS problem.

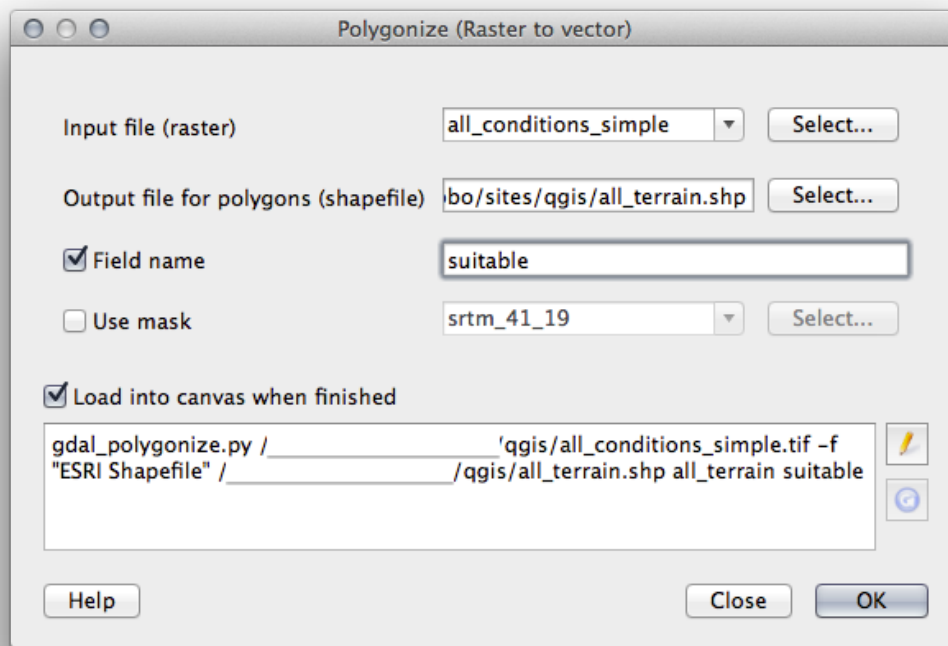
To combine a raster and vector analysis, you need to convert the one type of data to the other. Let's convert the raster result of the previous lesson to a vector.

The goal for this lesson: To get the raster result into a vector that can be used to complete the analysis.

9.1.1 Follow Along: The *Raster to Vector* Tool

Start with the map from the last module, `raster_analysis.qgs`. There you should have the `all_conditions_simple.tif` calculated during the previous exercises.

- Click on *Raster* → *Conversion* → *Polygonize (Raster to Vector)*. The tool dialog will appear.
- Set it up like this:



- Change the field name (describing the values of the raster) to `suitable`.
- Save the shapefile 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.

9.1.2 Try Yourself

Refer back to the module on vector analysis.

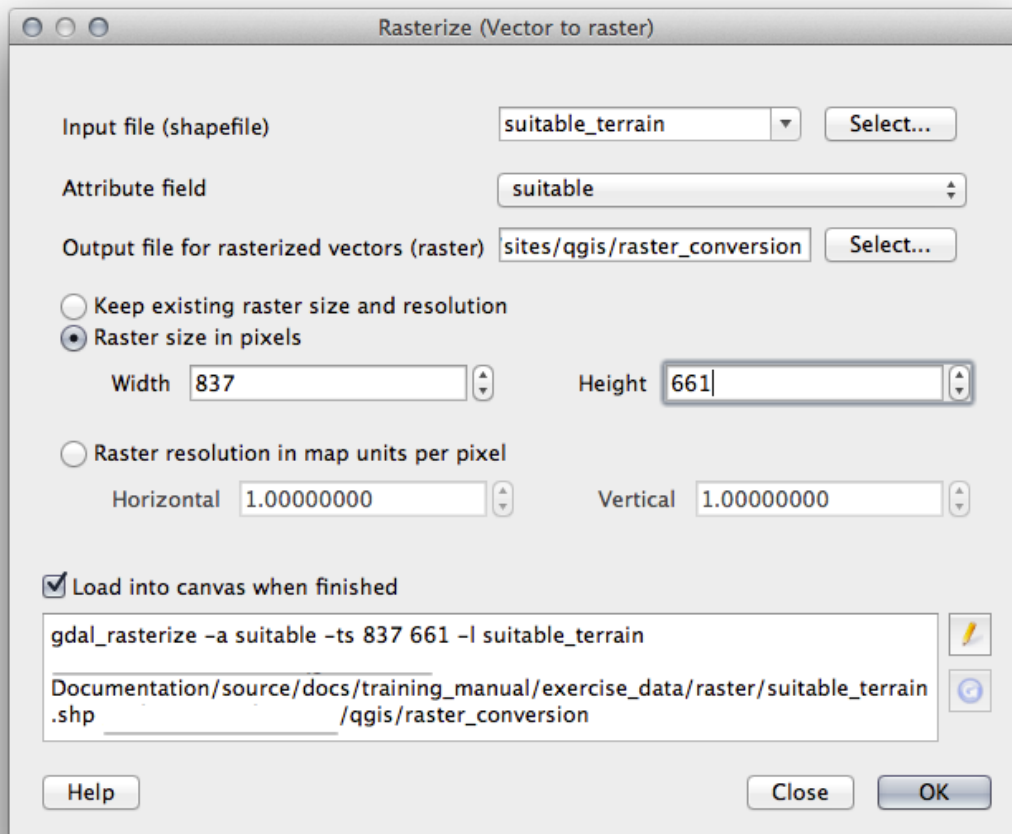
- Create a new vector file that contains only the polygons where `suitable` has the value of 1.
- Save the new file as `exercise_data/residential_development/` as `suitable_terrain.shp`.

Check your results

9.1.3 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.

- Click on *Raster* → *Conversion* → *Rasterize (Vector to Raster)* to start this tool, then set it up as in the screenshot below:



- *Input file* is *all_terrain*;
- *Output file...* is *exercise_data/residential_development/raster_conversion.tif*;
- *Width* and *Height* are 837 and 661, respectively.

Bemerkung: The size of the output image is specified here to be the same as the original raster which was vectorized. To view the dimensions of an image, open its metadata (*Metadata* tab in the *Layer Properties*).

- Click *OK* on the dialog to begin the conversion process.
- When it is complete, gauge its success by comparing the new raster with the original one. They should match up exactly, pixel for pixel.

9.1.4 In Conclusion

Converting between raster and vector formats allows you to widen the applicability of data, and need not lead to data degradation.

9.1.5 What's Next?

Now that we have the results of the terrain analysis available in vector format, they can be used to solve the problem of which buildings we should consider for the residential development.

9.2 Lesson: Combining the Analyses

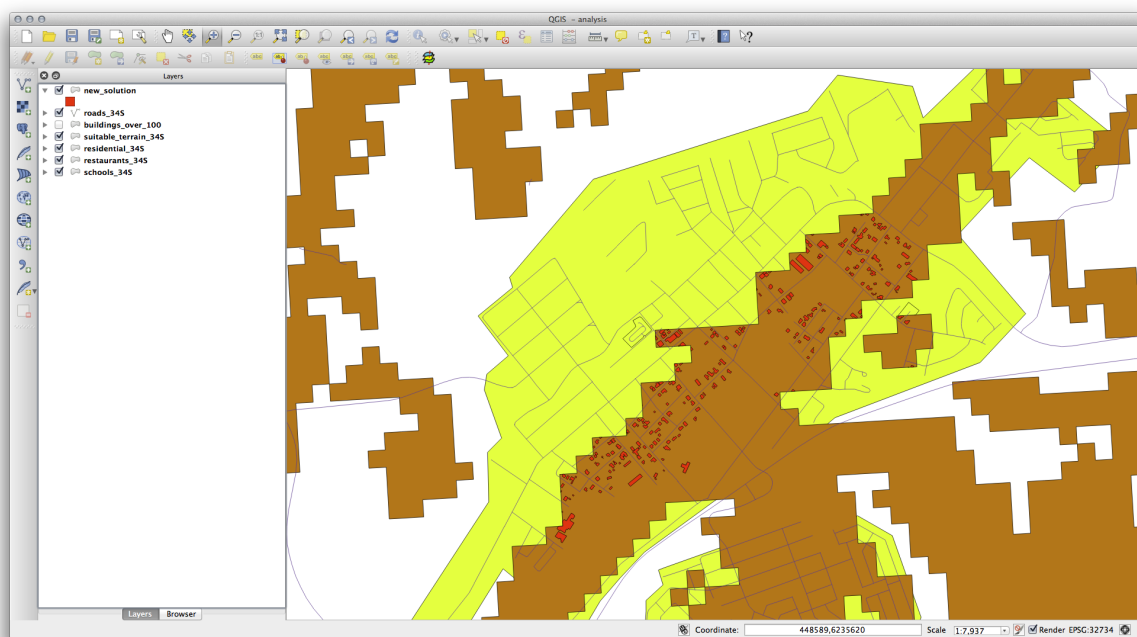
Using the vectorized results of the raster analysis will allow you to select only those buildings on suitable terrain.

The goal for this lesson: To use the vectorized terrain results to select suitable plots.

9.2.1 Try Yourself

- Save your current map (`raster_analysis.qgs`).
- Open the map in which you did the vector analysis earlier (`analysis.qgs`).
- In the *Layers list*, enable these layers:
 - *hillshade*,
 - *solution* (or *buildings_over_100*)
- 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.
- If you are missing some layers, you should find them in `exercise_data/residential_development/`
- Use the *Intersect* 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:



Bemerkung: If you find that the *Intersect* tool does not produce any results, check the CRS settings of each of your layers. The CRS must be the same for both the layers you are comparing. You may need to reproject one layer by saving the layer as a new shapefile with the required CRS. In our example, the `suitable_terrain` layer was reprojected to WGS 84 / UTM 34S and named `suitable_terrain_34S`.

9.2.2 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?

Check your results

9.2.3 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.

Check your results

9.2.4 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.

9.2.5 What's Next?

Next you will present these results as part of your second assignment.

9.3 Assignment

Using the Map Composer, 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.

9.4 Lesson: Supplementary Exercise

In this lesson, you will be guided through a complete GIS analysis in QGIS.

Bemerkung: Lesson developed by Linfiniti and S Motala (Cape Peninsula University of Technology)

9.4.1 Problem Statement

You are tasked with finding areas in and around the Cape Peninsula that are a suitable habitat for a rare fynbos plant species. The extent of your area of investigation in the Cape Peninsula is: south of Melkbosstrand, west of Strand. 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 > 1200 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 6000m² in area.

As a volunteer for Cape Nature, you have agreed to search for the plant on the closest suitable piece of land to your house. Use your GIS skills to determine where you should go to look.

9.4.2 Solution Outline

In order to solve this problem, you will have to use the available data (available in `exercise_data/more_analysis`) to find the candidate area that is closest to your house. If you don't live in Cape Town (where this problem is based) you can choose any house in the Cape Town region. The solution will involve:

- analysing the DEM to find the east facing slopes and the slopes with the correct gradients;
- analysing the rainfall raster to find the areas with the correct amount of rainfall;
- analysing the Zoning vector layer to find areas that are away from human settlement and are of the correct size.

9.4.3 Setting up the Map

- Click on the “CRS status” button in the extreme lower right corner of the screen. Under the *CRS* tab of the screen that appears, you will see the box *Coordinate reference systems of the world*.
- In this box, navigate to *Projected Coordinate Systems* → *Universal Transverse Mercator (UTM)*.
- Select the entry *WGS 84 / UTM zone 33S* (with the EPSG code 32733).
- Click *OK*. The map is now in the UTM33S coordinate reference system.
- Save the map by clicking on the *Save Project As* toolbar button, or use the *File* → *Save Project As...* menu item.
- Save the map in a directory called `Rasterprac` that you should create somewhere on your computer. You will save whatever layers you create in this directory as well.

9.4.4 Loading Data into the Map

In order to process the data, you will need to load the necessary layers (street names, zones, rainfall, DEM) into the map canvas.

For vectors ...

- Click on the *Add Vector Layer* button, or use the *Layer* → *Add Vector Layer...* menu item.
- In the dialog that appears, ensure that the *File* radio button is selected.
- Click on the *Browse* button.

- In the dialog that appears, open the *exercise_data/more_analysis/streets* directory.
- Select the file *Street_Names_UTM33S.shp*.
- Click *Open*.

The dialog closes and shows the original dialog, with the file path specified in the text field next to the *Browse* button. 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.

- Click *Open*. The vector layer will load in your map. Its color is automatically assigned. It will be changed later.
- Rename the layer to *Streets*.
- Right-click on it in the *Layers list* (by default, the pane along the left-hand side of the screen).
- Click *Rename* in the dialog that appears and rename it, pressing the *Enter* key when done.
- Repeat the vector adding process, but this time select the *Generalised_Zoning_Dissolve_UTM33S.shp* file in the *Zoning* directory.
- Rename it to *Zoning*.

For rasters ...

- Click on the *Add Raster Layer* button, or use the *Layer → Add Raster Layer...* menu item.
- Navigate to the appropriate file, select it, and click *Open*.
- Do this for each of the two raster files. The files you want are *DEM/reproject/DEM* and *Rainfall/reprojected/rainfall.tif*.
- Rename the rainfall raster to *Rainfall* (with an initial capital). Initially when you load them, the images will be gray rectangles. Don't worry, this will be changed later.
- Save the map.

In order to properly see what's going on, the symbology for the layers needs to be changed.

9.4.5 Changing the symbology of vector layers

- In the *Layers list*, right-click on the *Streets* layer.
- Select *Properties* from the menu that appears.
- Switch to the *Style* tab in the dialog that appears.
- Click on the button labeled *Change*, with a square showing the current color of the *Streets* layer.
- Select a new color in the dialog that appears.
- Click *OK*.
- Click *OK* again in the *Layer Properties* dialog. This will change the color of the *Streets* layer.
- Follow a similar process for the *Zoning* layer and choose an appropriate color for it.

9.4.6 Changing the symbology of raster layers

Raster layer symbology is somewhat different.

- Open the *Properties* dialog for the *Rainfall* raster.
- Switch to the *Style* tab. You'll notice that this style dialog is very different from the version used for vector layers.
- Ensure that the button *Use standard deviation* is selected.

- Change the value in the associated box to 2.00 (it should be set to 0.00 by default).
- Under the heading *Contrast enhancement*, change the value of the *Current* dropdown list to *Stretch to MinMax*.
- Click *OK*. The “Rainfall” raster, if visible, should change colors, allowing you to see different brightness values for each pixel.
- Repeat this process for the DEM, but set the standard deviations used for stretching to 4.00.

9.4.7 Changing the layer order

- In the *Layers list*, click and drag layers up and down to change the order they appear in on the map.
- Newer versions of QGIS may have a *Control rendering order* checkbox beneath the *Layers list*. Ensure that it is checked.

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 aren’t going to be used anyway.

9.4.8 Find the Correct Districts

- Load the vector layer `admin_boundaries/Western_Cape_UTM33S.shp` into your map.
- Rename it to `Districts`.
- Right-click on the *Districts* layer in the *Layers list*.
- In the menu that appears, select the *Query...* menu item. The *Query Builder* dialog appears.

You will now build a query to select only the following list of districts:

- Bellville,
- Cape,
- Goodwood,
- Kuils River,
- Mitchells Plain,
- Simons Town, and
- Wynberg.
- In the *Fields* list, double-click on the `NAME_2` field. It appears in the *SQL where clause* text field below.
- Click the `=` button; an `=` sign is added to the SQL query.
- 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`).
- Double-click the value *Bellville* in the *Values* list. As before, this will be added to the SQL query.

In order to select more than one district, you’ll need to use the `OR` boolean operator.

- Click the *OR* button and it will be added to the SQL query.
- Using a process similar to the above, add the following to the existing SQL query:

```
"NAME_2" = 'Cape'
```

- Add another `OR` operator, then work your way through the list of districts above in a similar fashion.
- The final query should be

```
"NAME_2" = 'Bellville' OR "NAME_2" = 'Cape' OR "NAME_2" = 'Goodwood' OR
"NAME_2" = 'Kuils River' OR "NAME_2" = 'Mitchells Plain' OR "NAME_2" =
'Simons Town' OR "NAME_2" = 'Wynberg'
```

- Click *OK*. The districts shown in your map are now limited to those in the list above.

9.4.9 Clip the Rasters

Now that you have an area of interest, you can clip the rasters to this area.

- Ensure that the only layers that are visible are the *DEM*, *Rainfall* and *Districts* layers.
- *Districts* must be on top so that they are visible.
- Open the clipping dialog by selecting the menu item *Raster* → *Extraction* → *Clipper*.
- In the *Input file (raster)* dropdown list, select the *DEM* layer.
- Specify an output location in the *Output file* text field by clicking the *Select...* button.
- Navigate to your *Rasterprac* directory.
- Enter a file name.
- Save the file. Leave the *No data value* checkbox unchecked.
- Use the *Extent* clipping mode by ensuring the correct radio button is selected.
- Click and drag an area in the canvas, so that the area which includes the districts is selected.
- Check the *Load into canvas when finished* box.
- Click *OK*.
- After the clipping operation is completed, DO NOT CLOSE the *Clipper* dialog. (Doing so would cause you to lose the clipping area that you have already defined.)
- Select the *Rainfall* raster in the *Input file (raster)* dropdown list and choose a different output file name.
- Do not change any other options. Do not alter the existing clipping area which you drew previously. Leave everything the same and click *OK*.
- After the second clipping operation has completed, you may close the *Clipper* dialog.
- Save the map.

9.4.10 Clean up the map

- Remove the original *Rainfall* and *DEM* layers from the *Layers list*:
- Right-click on these layers and select *Remove*.
 - This will not remove the data from your storage device, it will merely take it out of your map.
- Deactivate the labels on the *Streets* layer:
 - Click the *Labeling* button.
 - Uncheck the *Label this layer with* box.
 - Click *OK*.
- Show all the *Streets* again:
 - Right-click on the layer in the *Layers list*.
 - Select *Query*.
- In the *Query* dialog that appears, click the *Clear* button, then click *OK*.

- Wait while the data is loaded. All the streets will now be visible.
- Change the raster symbology as before (see *Changing the symbology of raster layers*).
- Save the map.
- You can now hide the vector layers by unchecking the box next to them in the *Layers list*. This will make the map render faster and will save you some time.

In order to create the hillshade, you will need to use a plugin that was written for this purpose.

9.4.11 Activating the *Raster Terrain Analysis* plugin

This plugin is included by default in QGIS 1.8. However, it may not be immediately visible. To check if it is accessible on your system:

- Click on the menu item *Plugins → Manage Plugins...*
- Ensure that the box next to *Raster Terrain Analysis plugin* is selected.
- Click *OK*.

You will now have access to this plugin via the *Raster → Terrain analysis* menu item.

Remember that plugins may sometimes depend on certain Python modules being installed on your system. Should a plugin refuse to work while complaining of missing dependencies, please ask your tutor or lecturer for assistance.

9.4.12 Create the hillshade

- In the *Layers list*, ensure that the *DEM* is the active layer (i.e., it is highlighted by having been clicked on).
- Click on the *Raster → Terrain analysis → Hillshade* menu item to open the *Hillshade* dialog.
- Specify an appropriate location for the output layer and call it *hillshade*.
- Check the *Add result to project* box.
- Click *OK*.
- Wait for it to finish processing.

The new *hillshade* layer has appeared in your *Layers list*.

- Right-click on the *hillshade* layer in your *Layers list* and bring up the *Properties* dialog.
- Click on the *Transparency* tab and set the transparency slider to 80%.
- Click *OK* on the dialog.
- Note the effect when the transparent hillshade is superimposed over the clipped DEM.

9.4.13 Slope

- Click on the menu item *Raster → Terrain analysis*.
- Select the *Slope* analysis type, with the clipped DEM as the input layer.
- Specify an appropriate file name and location for output purposes.
- Check the *Add result to project* box.
- Click *OK*.

The slope image has been calculated and added to the map. However, as usual it is just a gray rectangle. To properly see what's going on, change the symbology as follows.

- Open the layer *Properties* dialog (as usual, via the right-click menu of the layer).

- Click on the *Style* tab.
- Where it says *Grayscale* (in the *Color map* dropdown menu), change it to *Pseudocolor*.
- Ensure that the *Use standard deviation* radio button is selected.

9.4.14 Aspect

- Use the same approach as for calculating the slope, but select *Aspect* in the initial dialog box.

Remember to save the map periodically.

9.4.15 Reclassifying rasters

- Click the menu item *Raster* → *Raster calculator*.
- Specify your `Rasterprac` directory as the location for the output layer.
- Ensure that the *Add result to project* box is selected.

In the *Raster bands* list on the left, you will see all the raster layers in your *Layers list*. If your Slope layer is called *slope*, it will be listed as *slope@1*.

The slope needs to be between 15 and 60 degrees. Everything less than 15 or greater than 60 must therefore be excluded.

- Using the list items and buttons in the interface, build the following expression:

```
((slope@1 < 15) OR (slope@1 > 60)) = 0
```

- Set the *Output layer* field to an appropriate location and file name.
- Click *OK*.

Now find the correct aspect (east-facing: between 45 and 135 degrees) using the same approach.

- Build the following expression:

```
((aspect@1 < 45) OR (aspect@1 > 135)) = 0
```

- Find the correct rainfall (greater than 1200mm) the same way. Build the following expression:

```
(rainfall@1 < 1200) = 0
```

Having reclassified all the rasters, you will now see them displayed as gray rectangles in your map (assuming that they have been added to the map correctly). To properly display raster data with only two classes (1 and 0, meaning true or false), you will need to change their symbology.

9.4.16 Setting the style for the reclassified layers

- Open the *Style* tab in the layer's *Properties* dialog as usual.
- Under the heading *Load min / max values from band*, select the *Actual (slower)* radio button.
- Click the *Load* button.

The *Custom min / max values* fields should now populate with 0 and 1, respectively. (If they do not, then there was a mistake with your reclassification of the data, and you will need to go over that part again.)

- Under the heading *Contrast enhancement*, set the *Current* dropdown list to *Stretch To MinMax*.
- Click *OK*.
- Do this for all three reclassified rasters, and remember to save your work!

The only criterion that remains is that the area must be 250m away from urban areas. We will satisfy this requirement by ensuring that the areas we compute are 250m or more from the edge of a rural area. Hence, we need to find all rural areas first.

9.4.17 Finding rural areas

- Hide all layers in the *Layers list*.
- Unhide the *Zoning* vector layer.
- Right-click on it and bring up the *Query* dialog.
- Build the following query:

```
"Gen_Zoning" = 'Rural'
```

See the earlier instructions for building the *Streets* query if you get stuck.

- When you're done, close the *Query* dialog.

You should see a collection of polygons from the *Zoning* layer. You will need to save these to a new layer file.

- On the right-click menu for *Zoning*, select *Save as...*
- Save your layer under the *Zoning* directory.
- Name the output file `rural.shp`.
- Click *OK*.
- Add the layer to your map.
- Click the menu item *Vector* → *Geoprocessing Tools* → *Dissolve*.
- Select the *rural* layer as your input vector layer, while leaving the *Use only selected features* box unchecked.
- Under *Dissolve field*, select — *Dissolve all* —.
- Save your layer under the *Zoning* directory.
- Click *OK*. A dialog will appear asking whether you want to add the new layer to the TOC ("Table of Contents", referring to the *Layers list*).
- Click *Yes*.
- Close the *Dissolve* dialog.
- Remove the *rural* and *Zoning* layers.
- Save the map.

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.

9.4.18 Creating a negative buffer

- Click the menu item *Vector* → *Geoprocessing Tools* → *Buffer(s)*.
- In the dialog that appears, select the *rural_dissolve* layer as your input vector layer (*Use only selected features* should not be checked).
- Select the *Buffer distance* button and enter the value `-250` into the associated field; the negative value means that the buffer must be an internal buffer.
- Check the *Dissolve buffer results* box.
- Set the output file to the same directory as the other rural vector files.
- Name the output file `rural_buffer.shp`.

- Click *Save*.
- Click *OK* and wait for the processing to complete.
- Select *Yes* on the dialog that appears.
- Close the *Buffer* dialog.
- Remove the *rural_dissolve* layer.
- Save the map.

In order to incorporate the rural zones into the same analysis with the three existing rasters, it will need to be rasterized as well. But in order for the rasters to be compatible for analysis, they will need to be the same size. Therefore, before you can rasterize, you'll need to clip the vector to the same area as the three rasters. A vector can only be clipped by another vector, so you will first need to create a bounding box polygon the same size as the rasters.

9.4.19 Creating a bounding box vector

- Click on the menu item *Layer -> New -> New Shapefile Layer...*
- Under the *Type* heading, select the *Polygon* button.
- Click *Specify CRS* and set the coordinate reference system WGS 84 / UTM zone 33S : EPSG:32733.
- Click *OK*.
- Click *OK* on the *New Vector Layer* dialog as well.
- Save the vector in the *Zoning* directory.
- Name the output file *bbox.shp*.
- Hide all layers except the new *bbox* layer and one of the reclassified rasters.
- Ensure that the *bbox* layer is highlighted in the *Layers list*.
- Navigate to the *View > Toolbars* menu item and ensure that *Digitizing* is selected. You should then see a toolbar icon with a pencil or koki on it. This is the *Toggle editing* button.
- Click the *Toggle editing* button to enter *edit mode*. This allows you to edit a vector layer.
- Click the *Add feature* button, which should be nearby the *Toggle editing* button. It may be hidden behind a double arrow button; if so, click the double arrows to show the *Digitizing* toolbar's hidden buttons.
- With the *Add feature* tool activated, left-click on the corners of the raster. You may need to zoom in with the mouse wheel to ensure that it is accurate. To pan across the map in this mode, click and drag in the map with the middle mouse button or mouse wheel.
- For the fourth and final point, right-click to finalize the shape.
- Enter any arbitrary number for the shape ID.
- Click *OK*.
- Click the *Save edits* button.
- Click the *Toggle editing* button to stop your editing session.
- Save the map.

Now that you have a bounding box, you can use it to clip the rural buffer layer.

9.4.20 Clipping a vector layer

- Ensure that only the *bbox* and *rural_buffer* layers are visible, with the latter on top.
- Click the menu item *Vector > Geoprocessing Tools > Clip*.
- In the dialog that appears, set the input vector layer to *rural_buffer* and the clip layer to *bbox*, with both *Use only selected features* boxes unchecked.
- Put the output file under the *Zoning* directory.
- Name the output file `rural_clipped`.
- Click *OK*.
- When prompted to add the layer to the TOC, click *Yes*.
- Close the dialog.
- Compare the three vectors and see the results for yourself.
- Remove the *bbox* and *rural_buffer* layers, then save your map.

Now it's ready to be rasterized.

9.4.21 Rasterizing a vector layer

You'll need to specify a pixel size for a new raster that you create, so first you'll need to know the size of one of your existing rasters.

- Open the *Properties* dialog of any of the three existing rasters.
- Switch to the *Metadata* tab.
- Make a note of the X and Y values under the heading *Dimensions* in the Metadata table.
- Close the *Properties* dialog.
- Click on the *Raster → Conversion → Rasterize* menu item. You may receive a warning about a dataset being unsupported. Click it away and ignore it.
- Select *rural_clipped* as your input layer.
- Set an output file location inside the *Zoning* directory.
- Name the output file `rural_raster.tif`.
- Check the *New size* box and enter the X and Y values you made a note of earlier.
- Check the *Load into canvas* box.
- Click the pencil icon next to the text field which shows the command that will be run. At the end of the existing text, add a space and then the text `-burn 1`. This tells the Rasterize function to “burn” the existing vector into the new raster and give the areas covered by the vector the new value of 1 (as opposed to the rest of the image, which will automatically be 0).
- Click *OK*.
- The new raster should show up in your map once it has been computed.
- The new raster will look like a grey rectangle – you may change the display style as you did for the reclassified rasters.
- Save your map.

Now that you have all four criteria each in a separate raster, 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, but if a pixel has the value of 0 in any of the four rasters, then it will be 0 in the result. In this way, the result will contain only the overlapping areas.

9.4.22 Combining rasters

- Click the *Raster* → *Raster calculator* menu item.
- Build the following expression (with the appropriate names for your layers, depending on what you called them):

```
[Rural raster] * [Reclassified aspect] * [Reclassified slope] *
[Reclassified rainfall]
```

- Set the output location to the `Rasterprac` directory.
- Name the output raster `cross_product.tif`.
- Ensure that the *Add result to project* box is checked.
- Click OK.
- Change the symbology of the new raster in the same way as you set the style for the other reclassified rasters. The new raster now properly displays the areas where all the criteria are satisfied.

To get the final result, you need to select the areas that are greater than 6000m^2 . However, computing these areas accurately is only possible for a vector layer, so you will need to vectorize the raster.

9.4.23 Vectorizing the raster

- Click on the menu item *Raster* → *Conversion* → *Polygonize*.
- Select the `cross_product` raster.
- Set the output location to `Rasterprac`.
- Name the file `candidate_areas.shp`.
- Ensure that *Load into canvas when finished* is checked.
- Click OK.
- Close the dialog when processing is complete.

All areas of the raster have been vectorized, so you need to select only the areas that have a value of 1.

- Open the *Query* dialog for the new vector.
- Build this query:

```
"DN" = 1
```
- Click *OK*.
- Create a new vector file from the results by saving the `candidate_areas` vector after the query is complete (and only the areas with a value of 1 are visible). Use the *Save as...* function in the layer's right-click menu for this.
- Save the file in the `Rasterprac` directory.
- Name the file `candidate_areas_only.shp`.
- Save your map.

9.4.24 Calculating the area for each polygon

- Open the new vector layer's right-click menu.
- Select *Open attribute table*.
- Click the *Toggle editing mode* button along the bottom of the table, or press `Ctrl+E`.
- Click the *Open field calculator* button along the bottom of the table, or press `Ctrl+I`.

- Under the *New field* heading in the dialog that appears, enter the field name `area`. The output field type should be an integer, and the field width should be 10.
- In *Field calculator expression*, 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.

- Click *OK*.
- 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.

- Click *Toggle editing mode* again, and save your edits if prompted to do so.

9.4.25 Selecting areas of a given size

Now that the areas are known:

- Build a query (as usual) to select only the polygons larger than 6000m². The query is:

```
"area" > 6000
```

- Save the selection as a new vector layer called *solution.shp*.

You now have your solution areas, from which you will pick the one nearest to your house.

9.4.26 Digitize your house

- Create a new vector layer as before, but this time, select the *Type* value as being a *Point*.
- Ensure that it is in the correct CRS!
- Name the new layer `house.shp`.
- Finish creating the new layer.
- Enter edit mode (while the new layer is selected).
- Click the point where your house or other current place of residence is, using the streets as a guide. You might have to open other layers to help you find your house. If you don't live anywhere nearby, just click somewhere among the streets where a house could conceivably be.
- Enter any arbitrary number for the shape ID.
- Click *OK*.
- Save your edits and exit edit mode.
- Save the map.

You will need to find the centroids ("centers of mass") for the solution area polygons in order to decide which is closest to your house.

9.4.27 Calculate polygon centroids

- Click on the *Vector* → *Geometry Tools* → *Polygon centroids* menu item.
- Specify the input layer as *solution.shp*.
- Provide the output location as `Rasterprac`.

- Call the destination file `solution_centroids.shp`.
- Click *OK* and add the result to the TOC (*Layers list*), then close the dialog.
- Drag the new layer to the top of the layer order so that you can see it.

9.4.28 Calculate which centroid is closest to your house

- Click on the menu item *Vector -> Analysis Tools -> Distance matrix*.
- The input layer should be your house, and the target layer *solution_centroids*. Both of these should use the `id` field as their unique ID field.
- The output matrix type should be *linear*.
- Set an appropriate output location and name.
- Click *OK*.
- Open the file in a text editor (or import it into a spreadsheet). Note which target ID is associated with the shortest *Distance*. There may be more than one at the same distance.
- Build a query in QGIS to select only the solution areas closest to your house (selecting it using the `id` field).

This is the final answer to the research question.

For your submission, include 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 polygon of the closest solution area(s), as well as your house. Follow all the best practices for cartography in creating your output map.

Module: Plugins

Plugins allow you to extend the functionality QGIS offers. In this module, you'll be shown how to activate and use plugins.

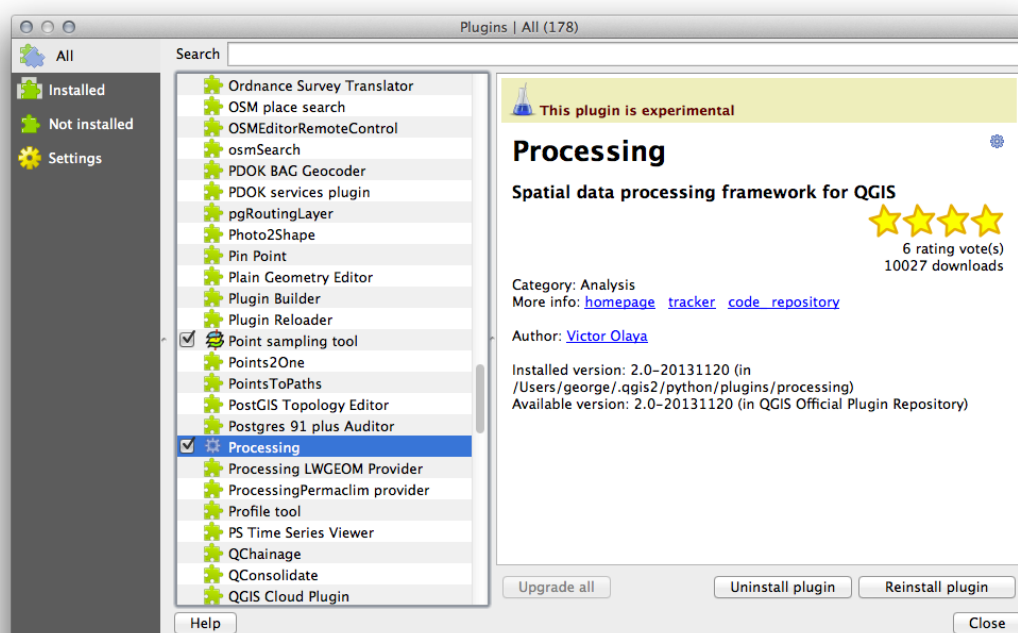
10.1 Lesson: Installing and Managing Plugins

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.

10.1.1 Follow Along: Managing Plugins

- To open the *Plugin Manager*, click on the menu item *Plugins* → *Manage and Install Plugins*.
- In the dialog that opens, find the *Processing* plugin:



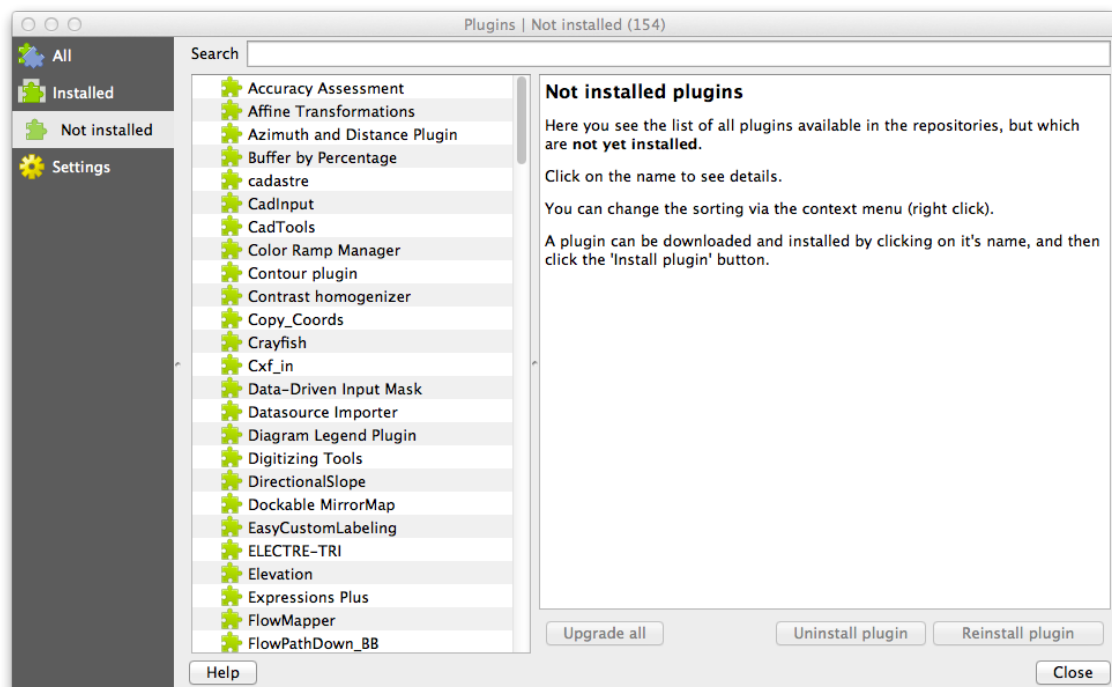
- Click in the box next to this plugin and uncheck it to uninstall it.

- Click *Close*.
- Looking at the menu, you will notice that the *Processing* menu is now gone. This means that many of the processing functions you have been using before have disappeared! This is because they are part of the *Processing* plugin, which needs to be activated for you to use them.
- Open the *Plugin Manager* again and reactivate the *Processing* plugin by clicking in the checkbox next to it and clicking *Close*..
- The *Processing* menu should be available again.

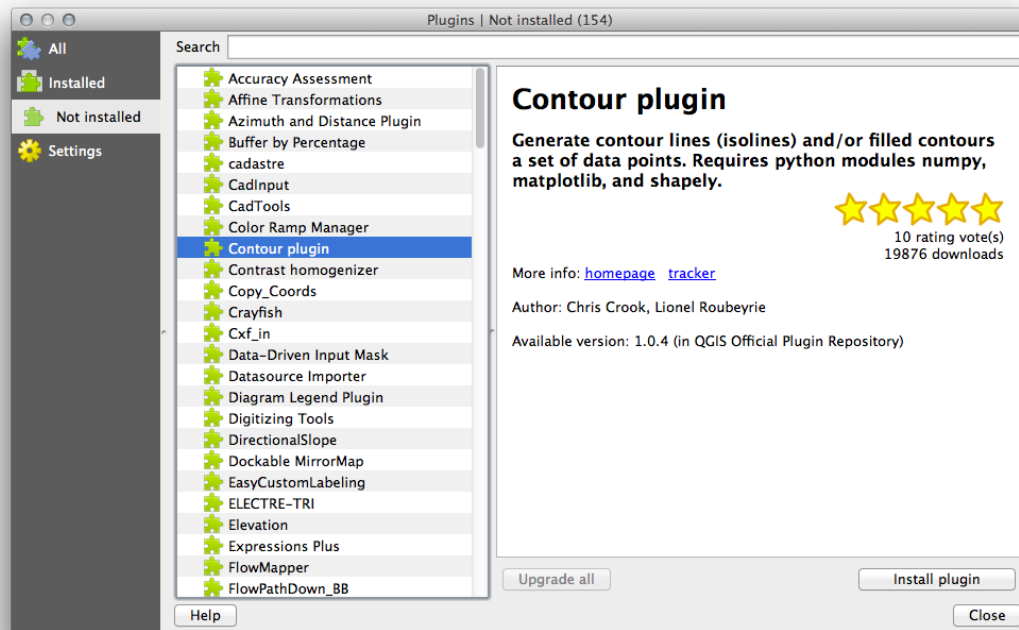
10.1.2 Follow Along: Installing New Plugins

The list of plugins that you can activate and deactivate draws from the plugins that you currently have installed.

- To install new plugins, 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.



- You can find information about each plugin by selecting it in the list of plugins displayed.



- A plugin can be installed by clicking the *Install Plugin* button below the plugin information panel.

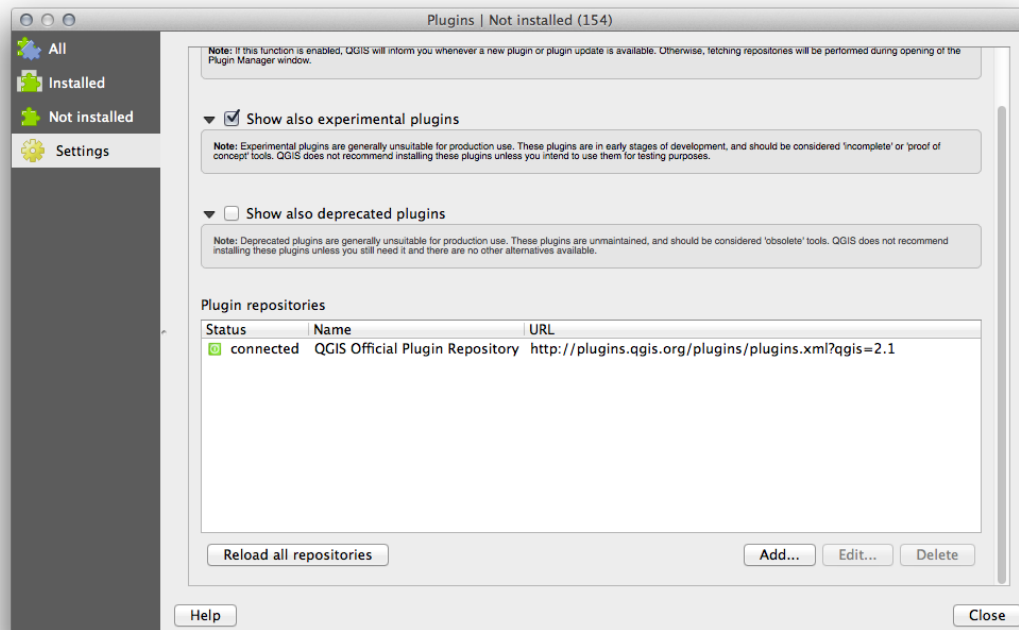
10.1.3 Follow Along: Configuring Additional Plugin Repositories

The plugins that are available to you for installation depend on which plugin *repositories* you are configured to use.

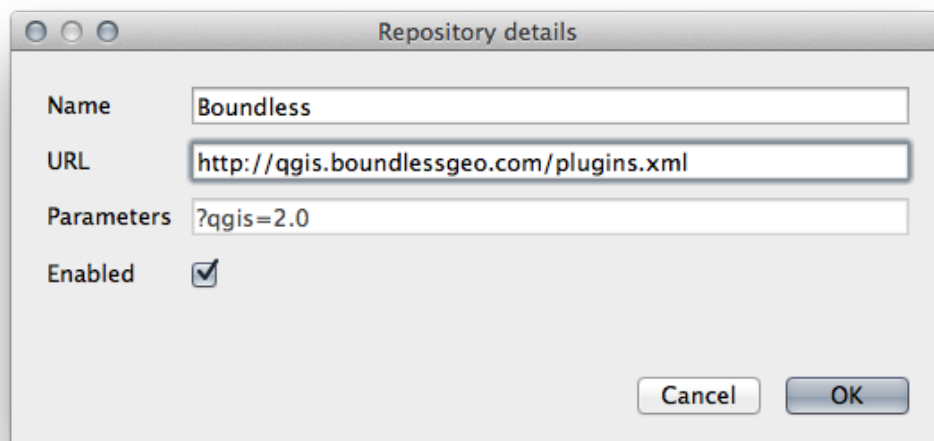
QGIS plugins are stored online in repositories. By default, only the official repositories are active, meaning that you can only access official plugins. These are usually the first plugins you want, because they have been tested thoroughly and are often included in QGIS by default.

It is possible, however, to try out more plugins than the default ones. First, you want to configure additional repositories. To do this:

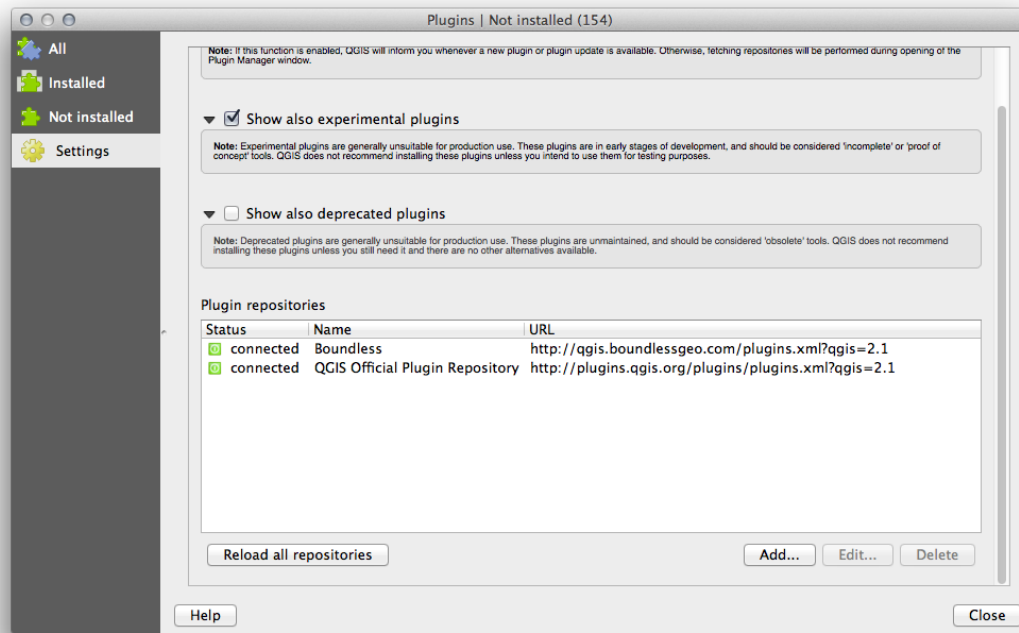
- Open the *Settings* tab in the *Plugin Manager* dialog:



- Click *Add* to find and add a new repository.
- Provide a Name and URL for the new repository you want to configure and make sure the *Enabled* checkbox is selected.



- You will now see the new plugin repo listed in the list of configured Plugin Repositories



- You can also select the option to display Experimental Plugins by selecting the *Show also experimental plugins* checkbox.
- If you now switch back to the *Get More* tab, you will see that additional plugins are now available for installation.
- To install a plugin, simply click on it in the list and then click the *Install plugin* button.

10.1.4 In Conclusion

Installing plugins in QGIS is simple and effective!

10.1.5 What's Next?

Next we'll introduce you to some useful plugins as examples.

10.2 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.

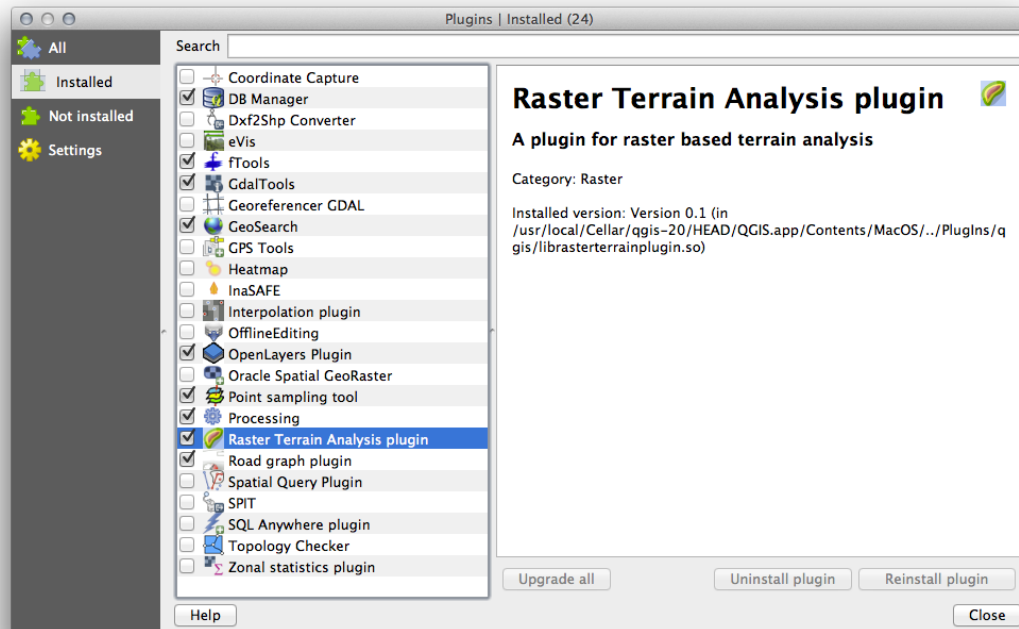
The goal for this lesson: To familiarize yourself with the plugin interface and get acquainted with some useful plugins.

10.2.1 Follow Along: The Raster Terrain Analysis Plugin

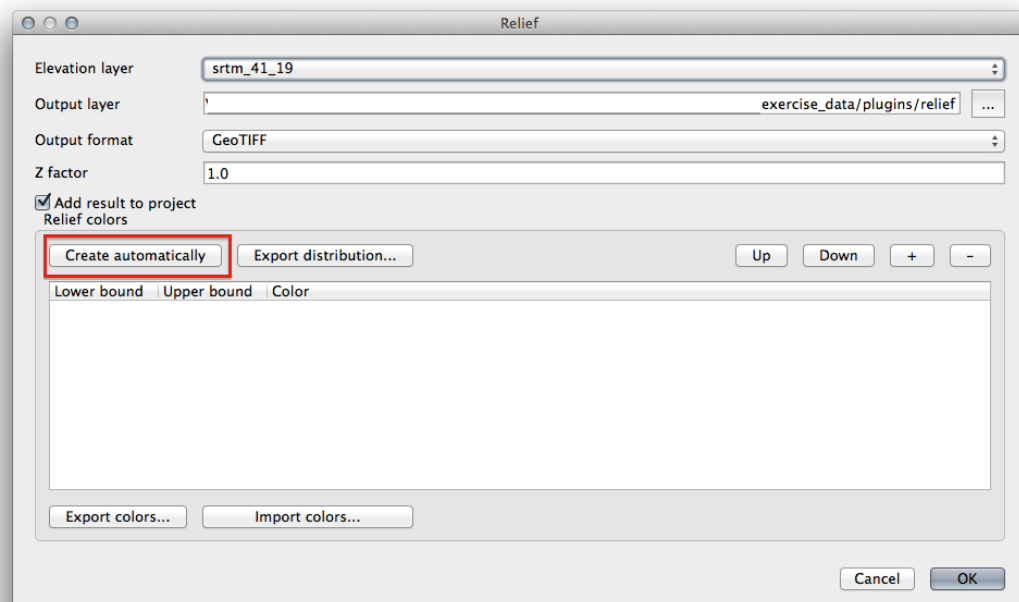
- Start a new map with only the *srtm_41_19.tif* raster dataset in it (look in `exercise_data/raster/SRTM`).

From the lesson on raster analysis, you're already familiar with raster analysis functions. You used GDAL tools (accessible via *Raster* → *Analysis*) for this. However, you should also know about the Raster Terrain Analysis plugin. This ships standard with newer versions of QGIS, and so you don't need to install it separately.

- Open the *Plugin Manager* and check that the Raster Terrain Analysis plugin is enabled:

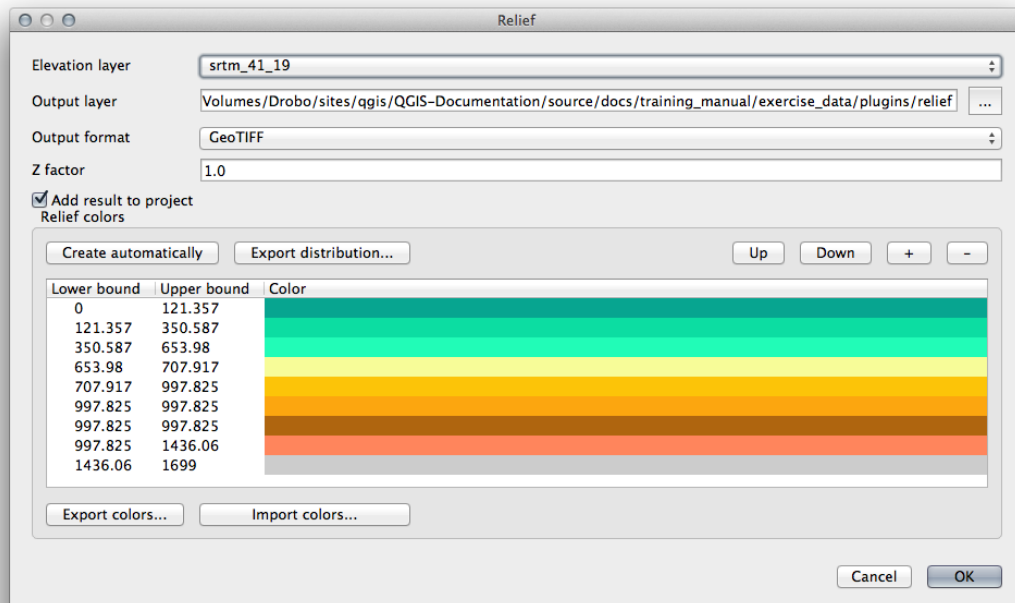


- Open the *Raster* menu. You should see a *Terrain analysis* submenu.
- Click on *Terrain analysis* → *Relief* and input the following options:



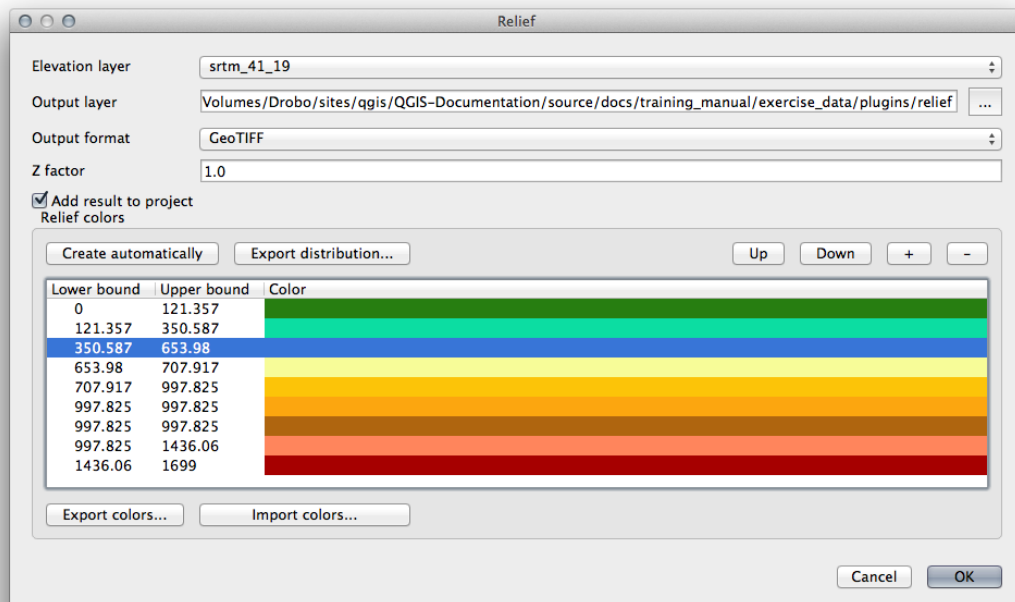
- Save the new file under `exercise_data/plugins/relief.tif` (create a new folder if necessary).
- Leave the *Output format* and *Z factor* unchanged.

- Make sure the *Add result to project* box is checked.
- Click the *Create automatically* button. The list below will be populated:

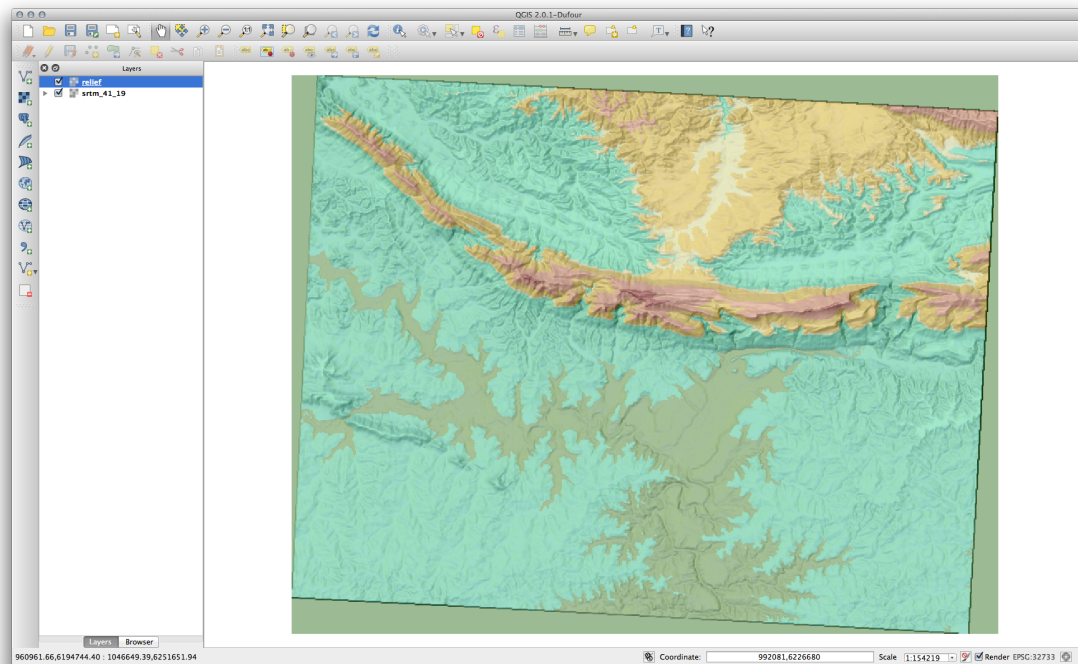


These are the colors that the plugin will use to create the relief.

- If you like, you can change these colors by double-clicking on each row's color bar. For example:



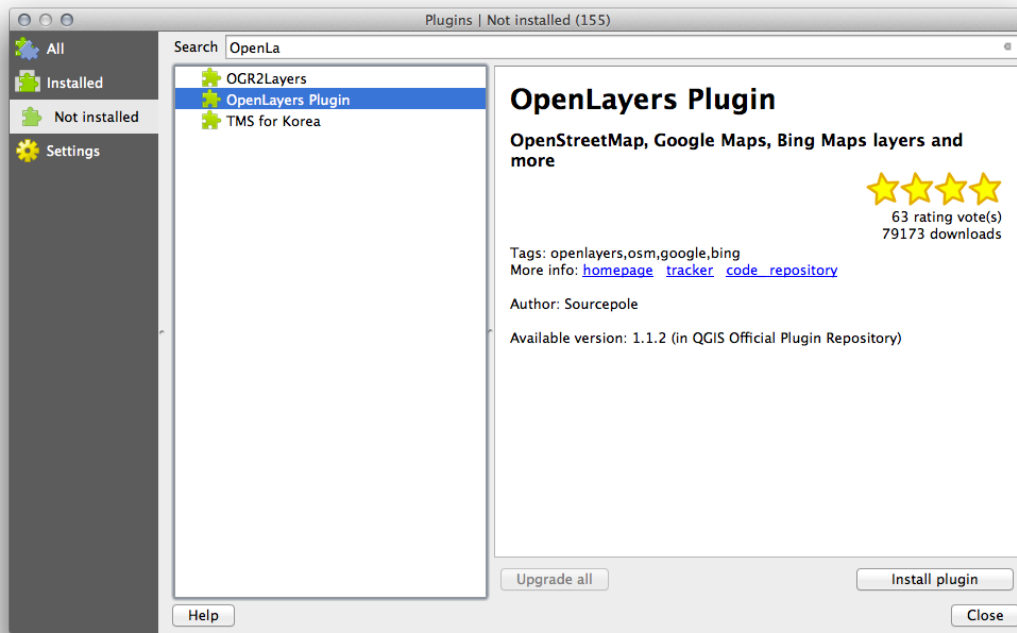
- Click *OK* and the relief will be created:



This achieves a similar effect to when you used the semi-transparent hillshade as an overlay over another raster layer. The advantage of this plugin is that it creates this effect using only one layer.

10.2.2 Follow Along: The OpenLayers Plugin

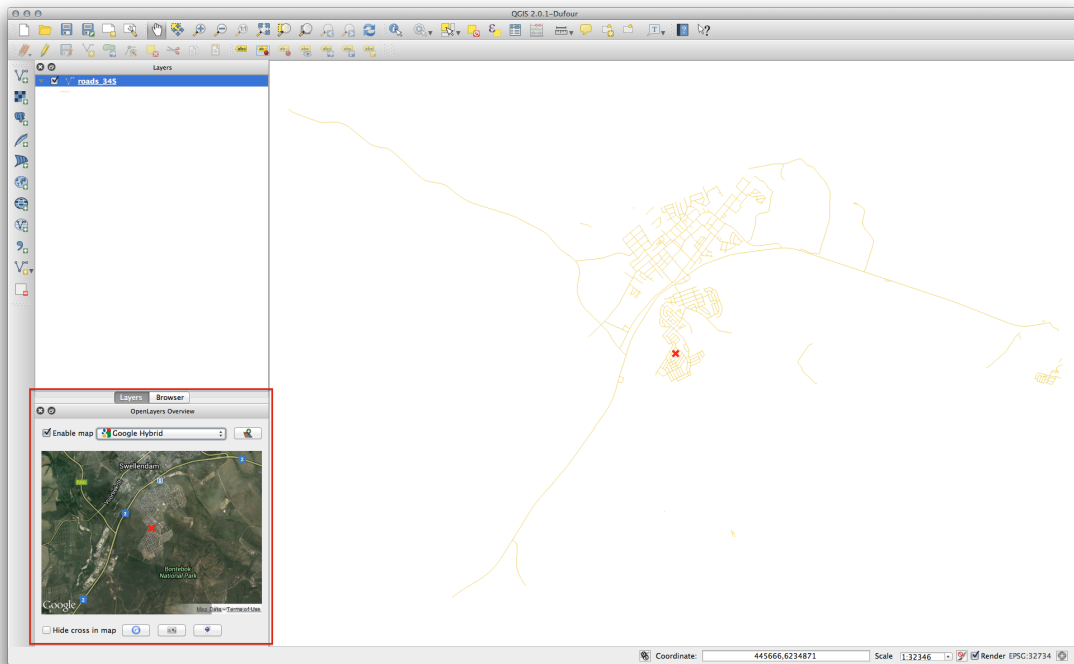
- Start a new map and add the *roads.shp* layer to it.
- Zoom in over the Swellendam area.
- Using the *Plugin Manager*, find a new plugin by entering the word *OpenLayers* in the *Filter* field.
- Select the OpenLayers Layers plugin from the filtered list:



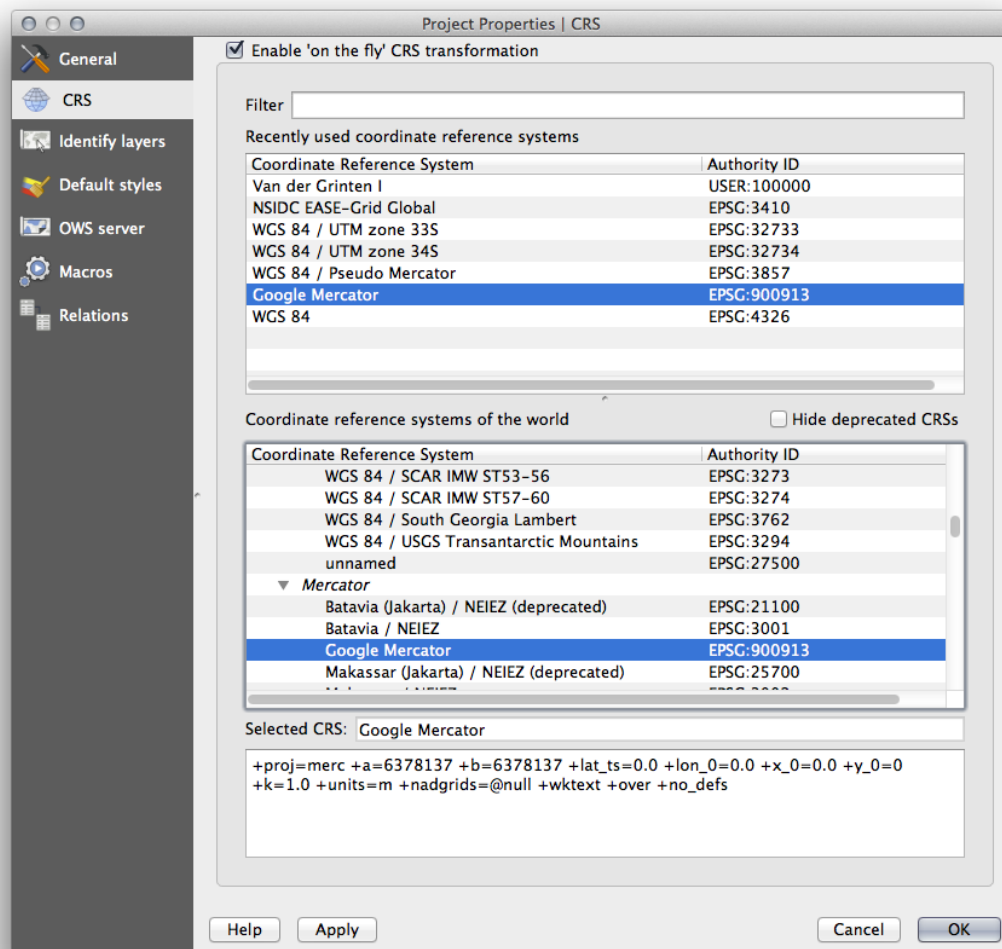
- Click the *Install plugin* button to install.
- When it's done, close the *Plugin Manager*.

Before using it, make sure that both your map and the plugin are configured properly:

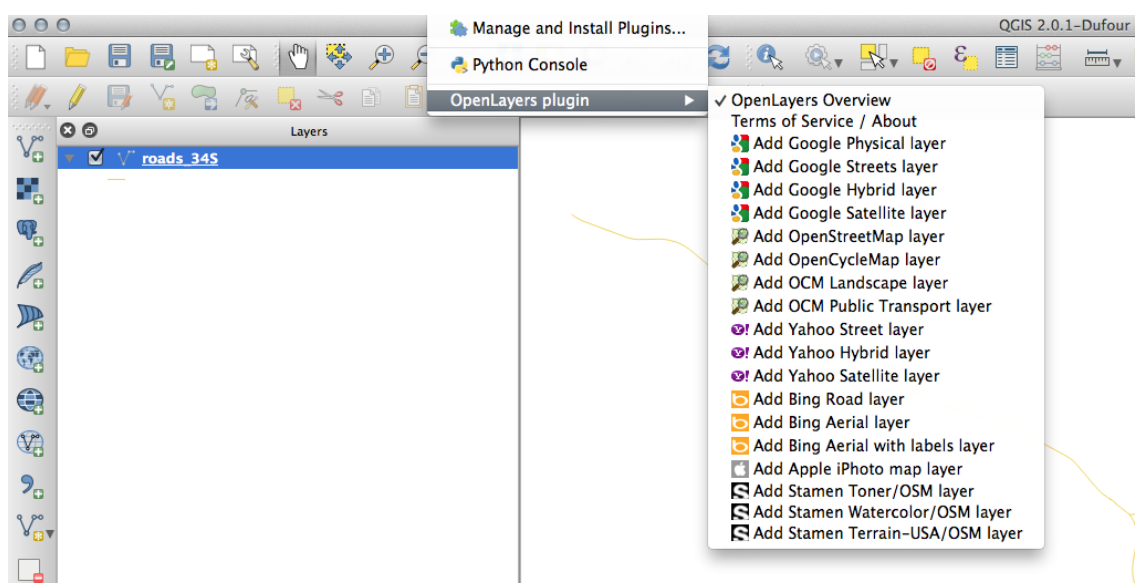
- Open the plugin's settings by clicking on *Plugins* → *OpenLayers plugin* → *OpenLayers Overview*.
- Use the panel to choose a map type you want. In this example, we'll use the "Hybrid" type map, but you can choose any others if you want.



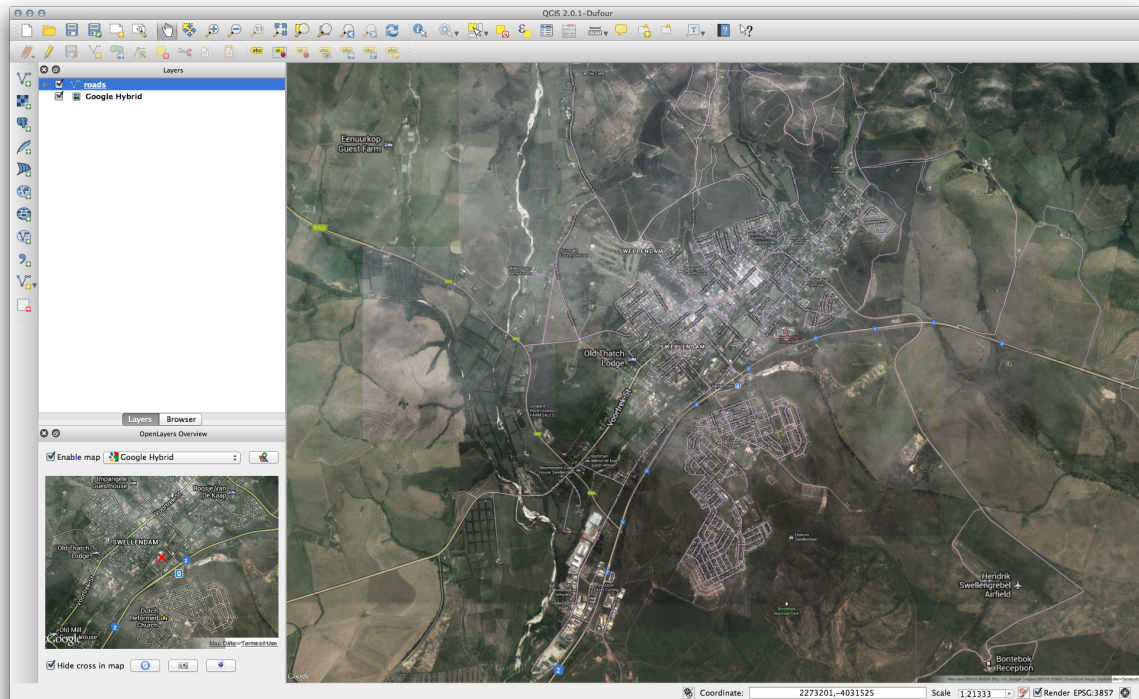
- Open the *Project Properties* Dialog by selecting *Project* → *Project Properties* from the menu.
- Enable "on the fly" projection and use the Google Mercator projection:



- Now use the plugin to give you a Google map of the area. You can click on *Plugins* → *OpenLayers Plugin* → *Add Google Hybrid Layer* to add it:



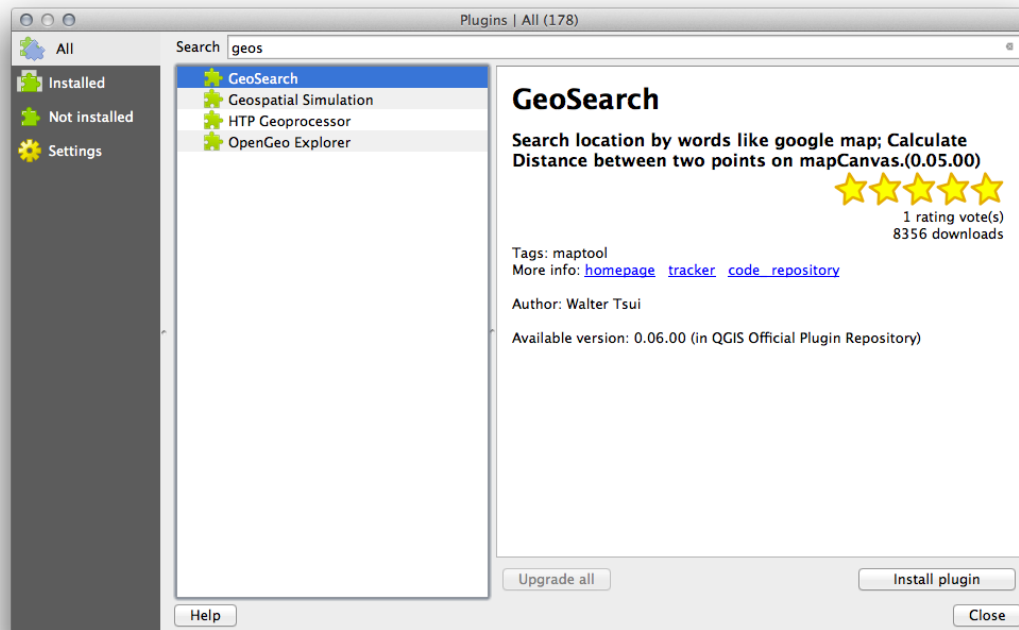
This will load a new raster image in from Google that you can use as a backdrop, or to help you find out where you are on the map. Here is such a layer, with our own vector road layer as overlay:



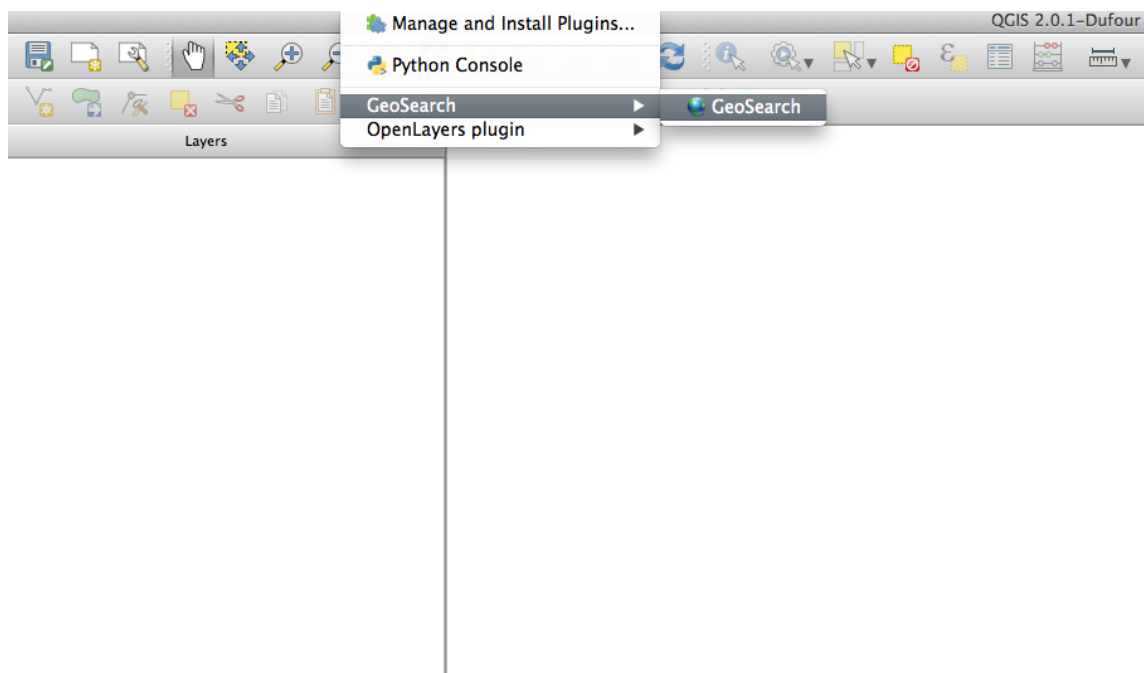
Bemerkung: You may need to drag your roads layer above the Google layer to make it visible above the background layer. It may also be necessary to zoom to the extent of the roads layer to re-center the map.

10.2.3 Follow Along: The GeoSearch Plugin

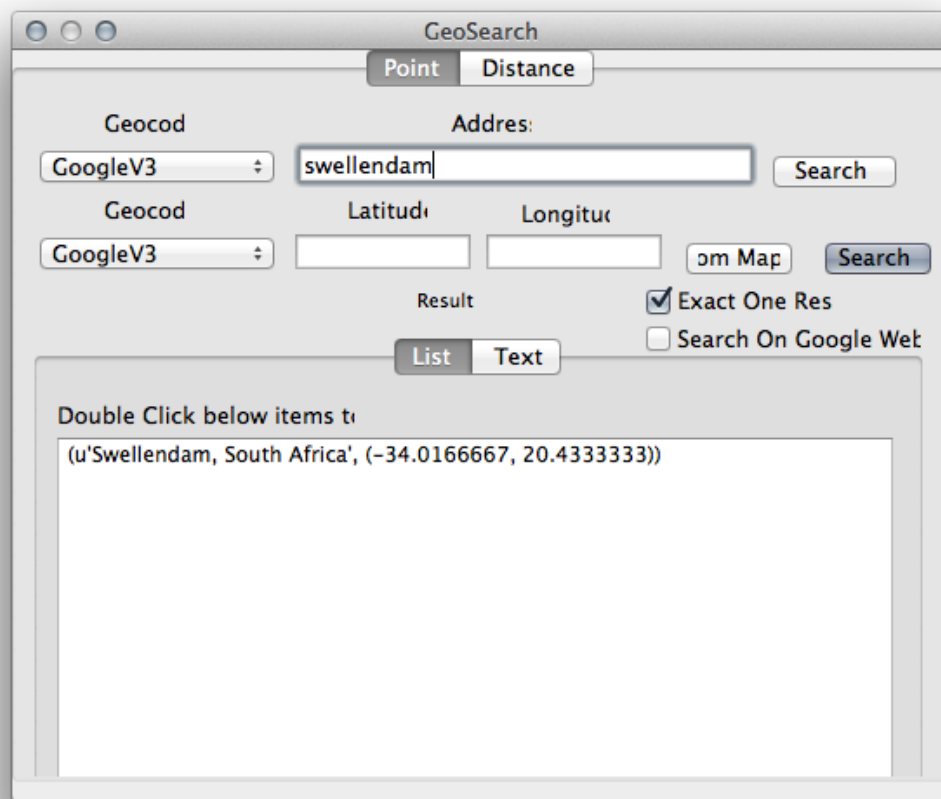
- Start a new map with no datasets.
- Open the *Plugin Manager* and filter for the GeoSearch Plugin and click *Install Plugin* to install it.



- Close the *Plugin Manager*.
- You can now use the GeoSearch plugin to search for placenames. Click on *Plugins* → *GeoSearch Plugin* → *GeoSearch* to open the GeoSearch dialog.



- Search for Swellendam in the GeoSearch Dialog to locate it on your map:



10.2.4 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.

10.2.5 What's Next?

Next we'll look at how to use layers that are hosted on remote servers in real time.

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).

11.1 Lesson: Web Mapping Services

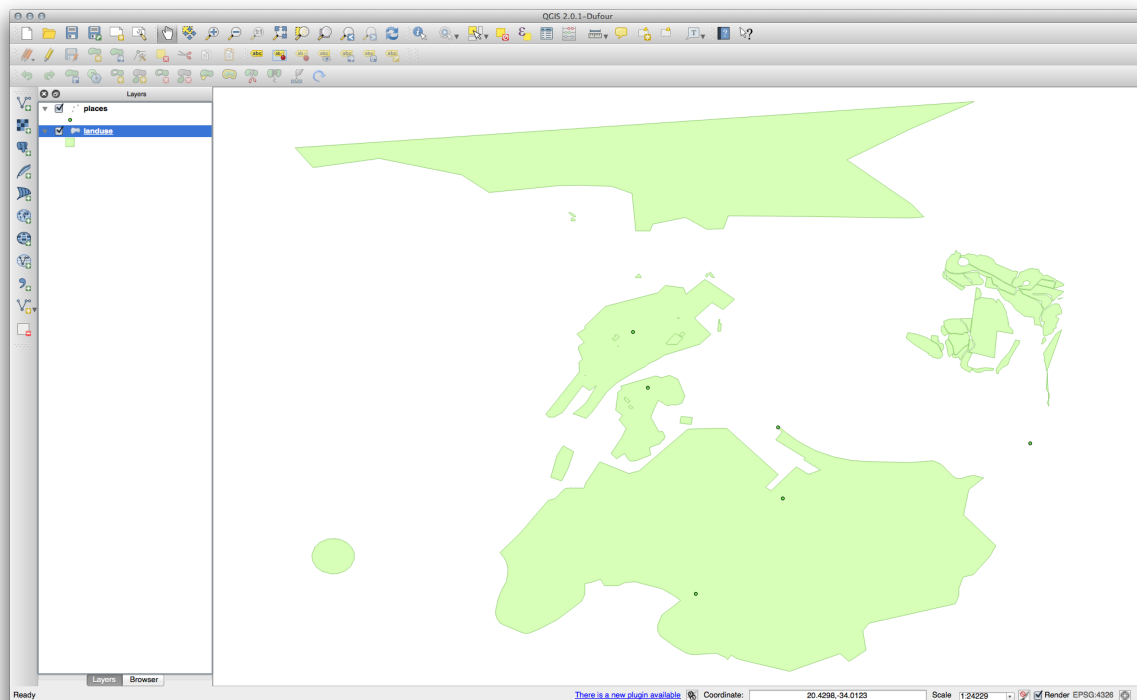
A Web Mapping Service (WMS) is a service hosted on a remote server. Similar to a website, you can access it as long as you have a connection to the server. Using QGIS, you can load a WMS directly into your existing map.

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.

11.1.1 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* and *landuse* layers and adjusted the symbology:



- Load these layers into a new map, or use your original map with only these layers visible.
- Before starting to add the WMS layer, first deactivate “on the fly” projection. This may cause the layers to no longer overlap properly, but don’t worry: we’ll fix that later.
- To add WMS layers, click on the *Add WMS Layer* button:

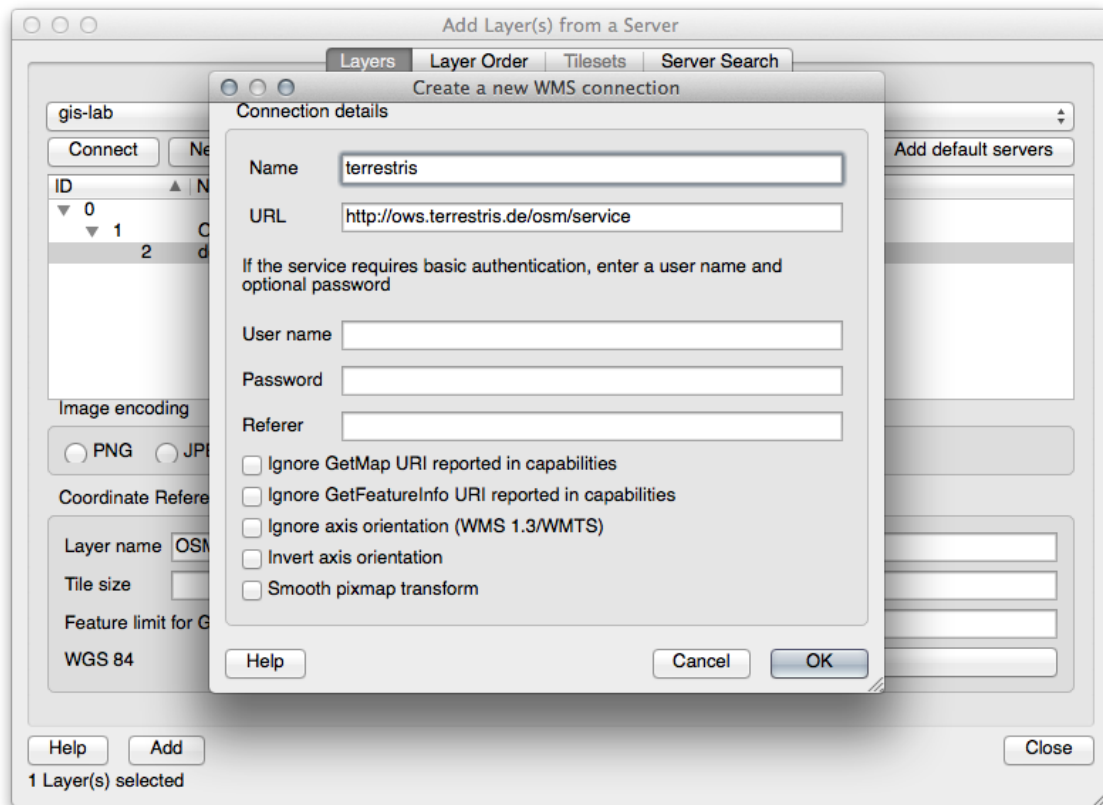


Remember how you connected to a Spatialite database at the beginning of the course. The *landuse*, *places*, and *water* layers are in that database. To use those layers, you first needed to connect to the database. Using a WMS is similar, with the exception that the layers are on a remote server.

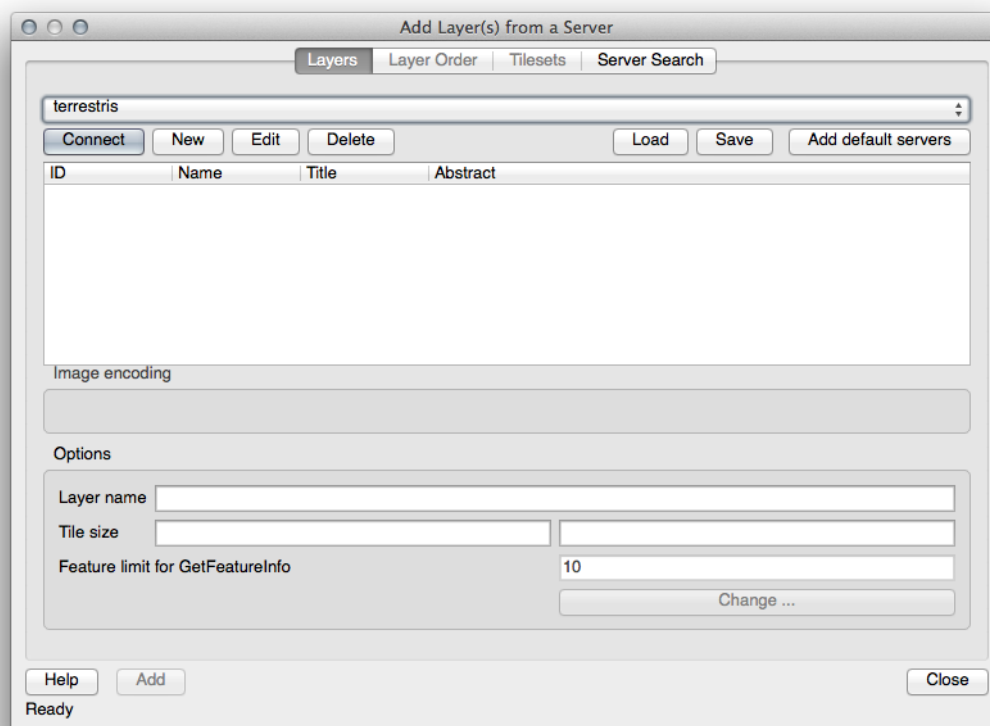
- To create a new connection to a WMS, click on the *New* button.

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.

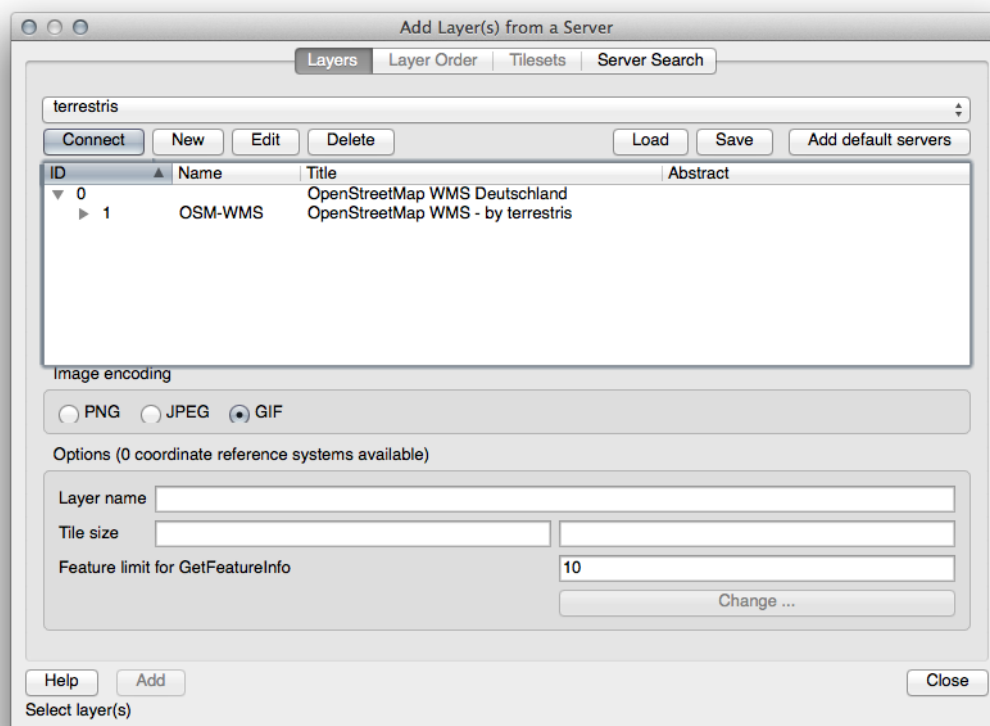
- To make use of this WMS, set it up in your current dialog, like this:



- The value of the *Name* field should be terrestris.
- The value of the *URL* field should be `http://ows.terrestris.de/osm/service`.
- Click *OK*. You should see the new WMS server listed:

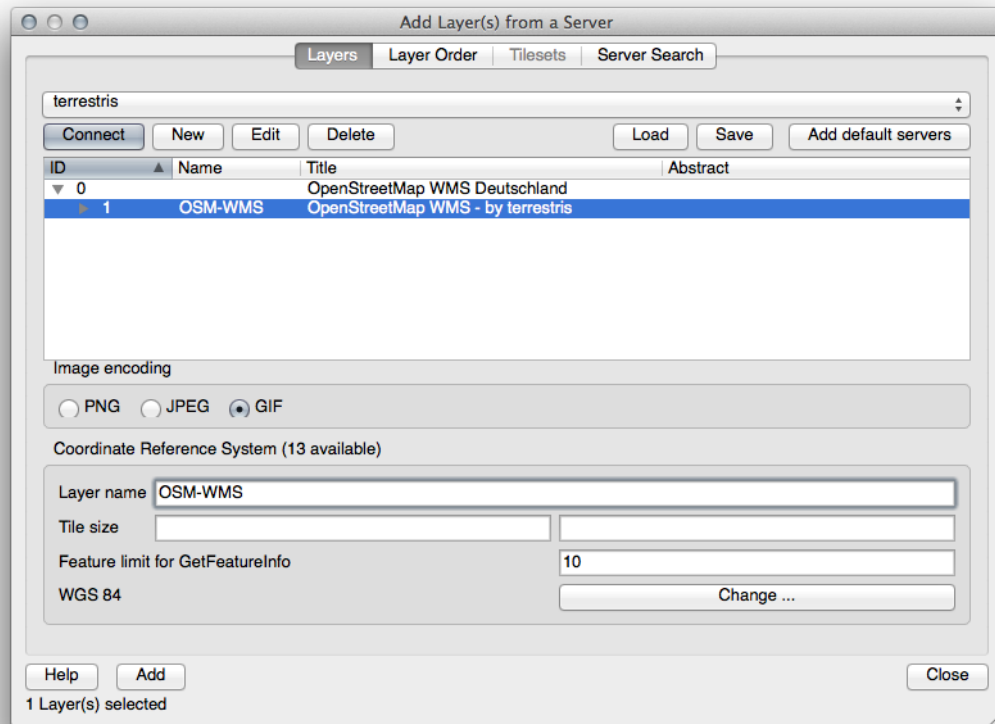


- Click *Connect*. In the list below, you should now see these new entries loaded:



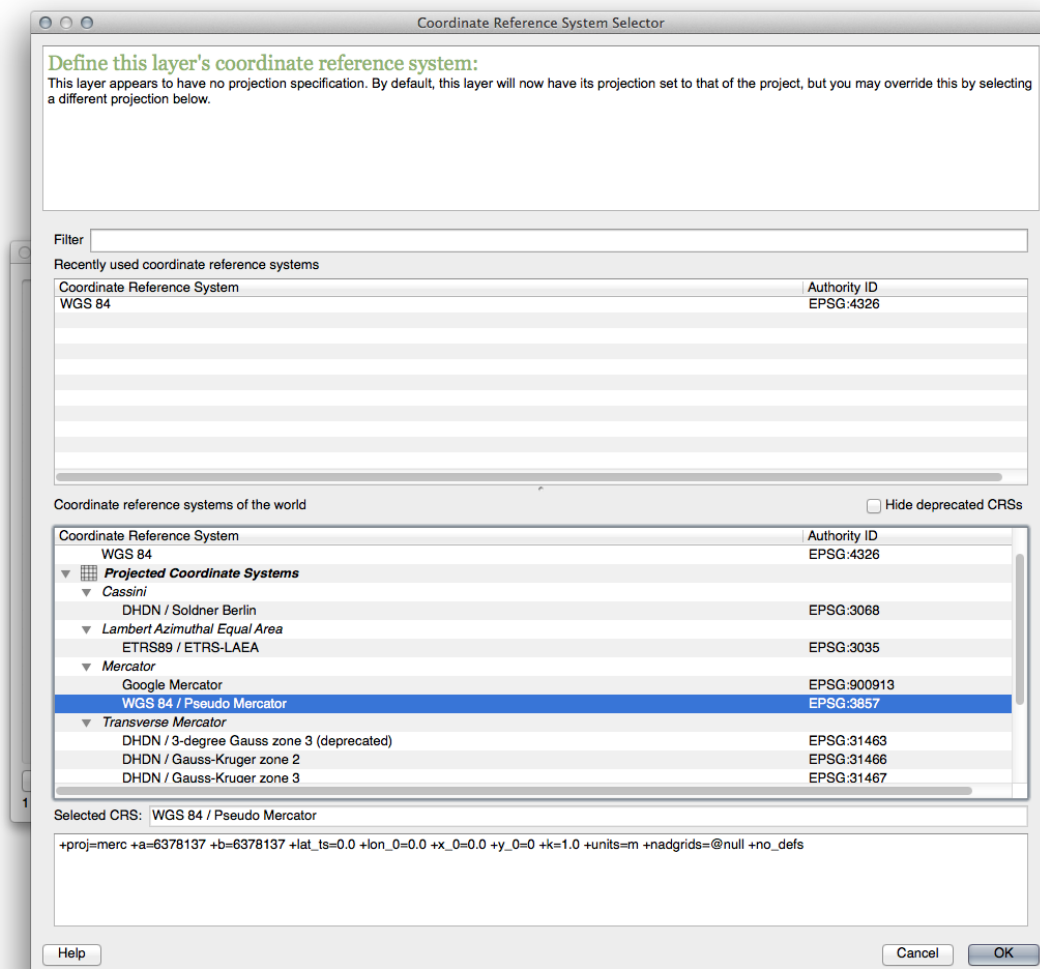
These are all the layers hosted by this WMS server.

- Click once on the *OSM-WMS* layer. This will display its *Coordinate Reference System*:



Since we're not using WGS 84 for our map, let's see all the CRSs we have to choose from.

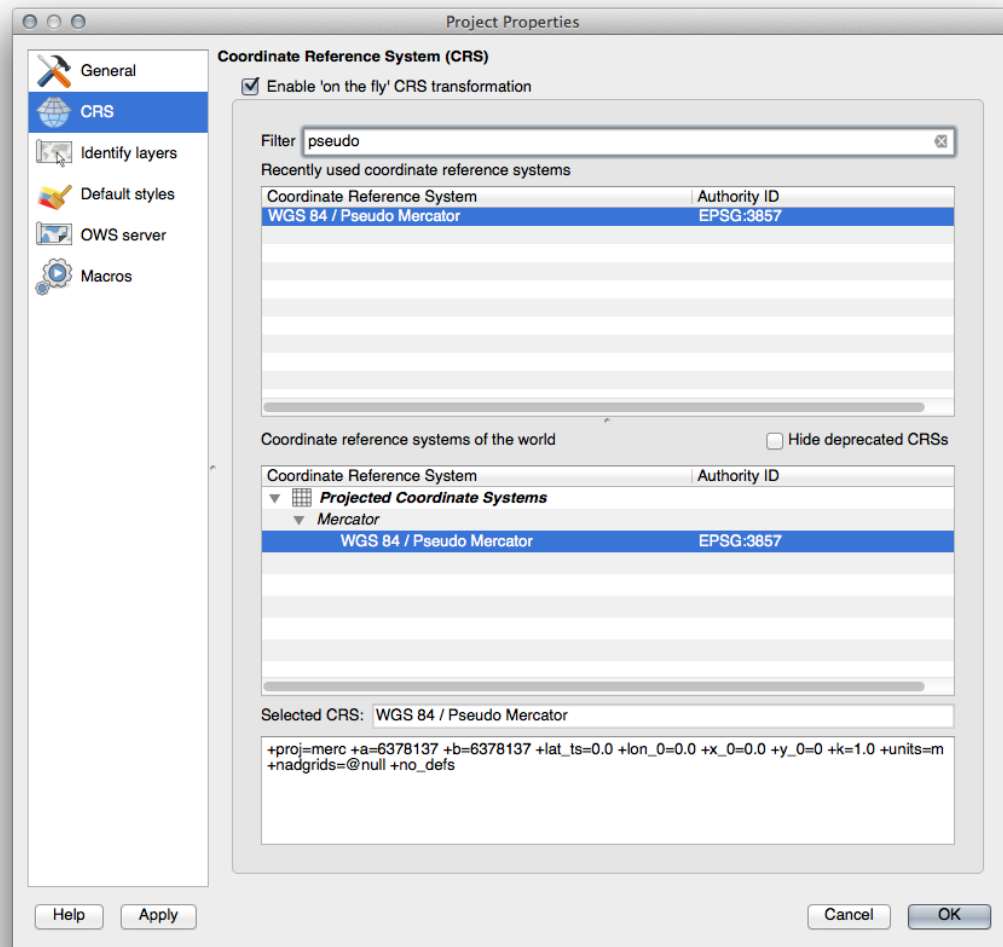
- Click the *Change* button. You will see a standard *Coordinate Reference System Selector* dialog.
- We want a *projected* CRS, so let's choose *WGS 84 / Psuedo Mercator*.



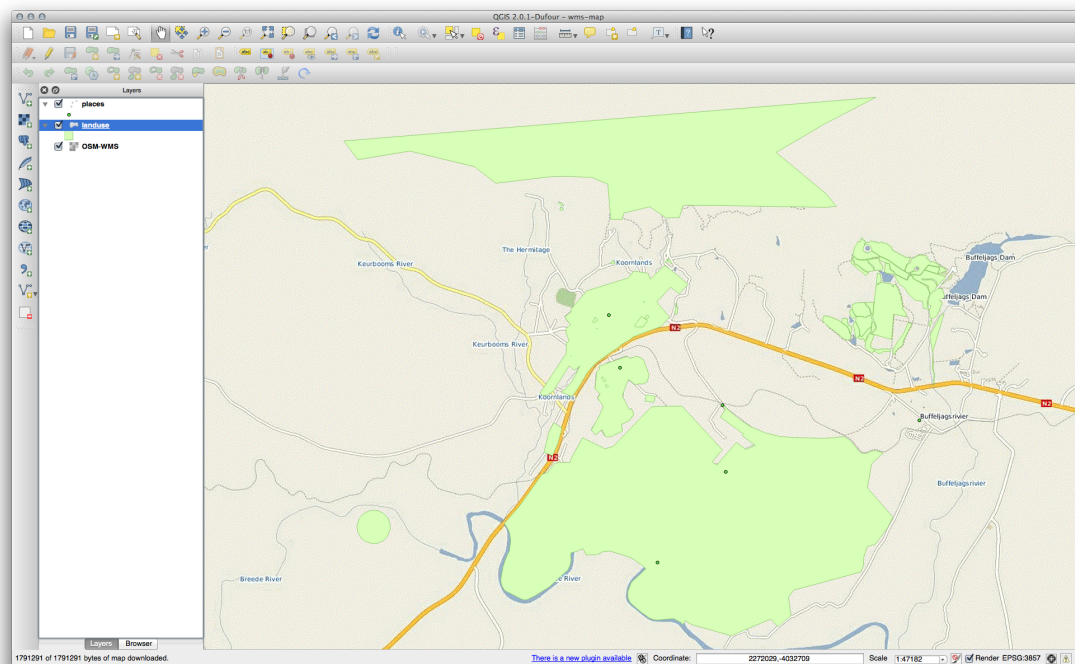
- Click **OK**.
- Click **Add** and the new layer will appear in your map as *OSM-WMS*.
- In the *Layers list*, click and drag it to the bottom of the list.

You will notice that your layers aren't located correctly. This is because "on the fly" projection is disabled. Let's enable it again, but using the same projection as the *OSM-WMS* layer, which is *WGS 84 / Pseudo Mercator*.

- Enable "on the fly" projection.
- In the *CRS* tab (*Project Properties* dialog), enter the value `pseudo` in the *Filter* field:



- Choose *WGS 84 / Pseudo Mercator* from the list.
- Click *OK*.
- Now right-click on one of your own layers in the *Layers list* and click *Zoom to layer extent*. 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.

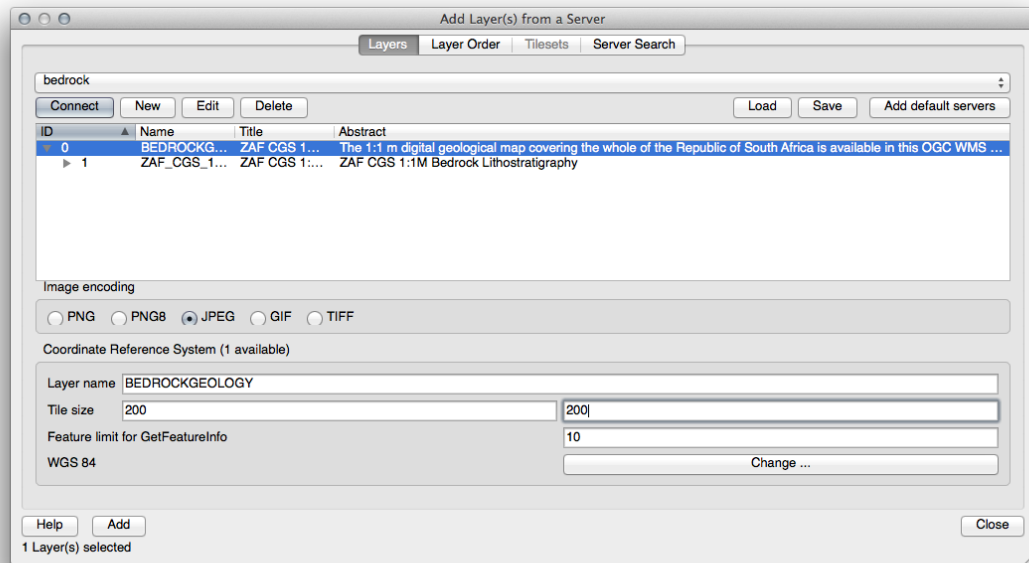
This has several advantages, because you don't need to worry about the symbology. It's already worked out, and should be nice to look at on any competently designed WMS.

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 from the *terrestris* WMS server.

11.1.2 Try Yourself

- Hide the *OSM-WMS* layer in the *Layers list*.
- Add the “ZAF CGS 1M Bedrock Lithostratigraphy” WMS server at this URL:
http://196.33.85.22/cgi-bin/ZAF_CGS_Bedrock_Geology/wms
- Load the *BEDROCKGEOLOGY* layer into the map (use the *Add WMS Layer* button as before). Remember to check that it's in the same *WGS 84 / World Mercator* projection as the rest of your map!
- You might want to set its *Encoding* to *JPEG* and its *Tile size* option to 200 by 200, so that it loads faster:



Check your results

11.1.3 Try Yourself

- Hide all other WMS layers to prevent them rendering unnecessarily in the background.
- Add the “OGC” WMS server at this URL: <http://ogc.gbif.org:80/wms>
- Add the *bluemarble* layer.

Check your results

11.1.4 Try Yourself

Part of the difficulty of using WMS is finding a good (free) server.

- Find a new WMS at 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).

Check your results

11.1.5 In Conclusion

Using a WMS, you can add inactive maps as backdrops for your existing map data.

11.1.6 Further Reading

- spatineo.com
- Geopole.org
- OpenStreetMap.org list of WMS servers

11.1.7 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.

11.2 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.

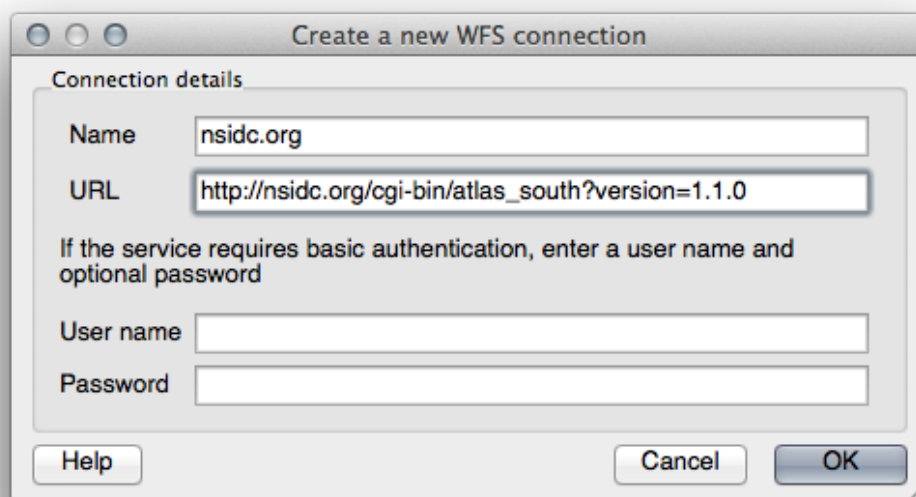
The goal for this lesson: To use a WFS and understand how it differs from a WMS.

11.2.1 Follow Along: Loading a WFS Layer

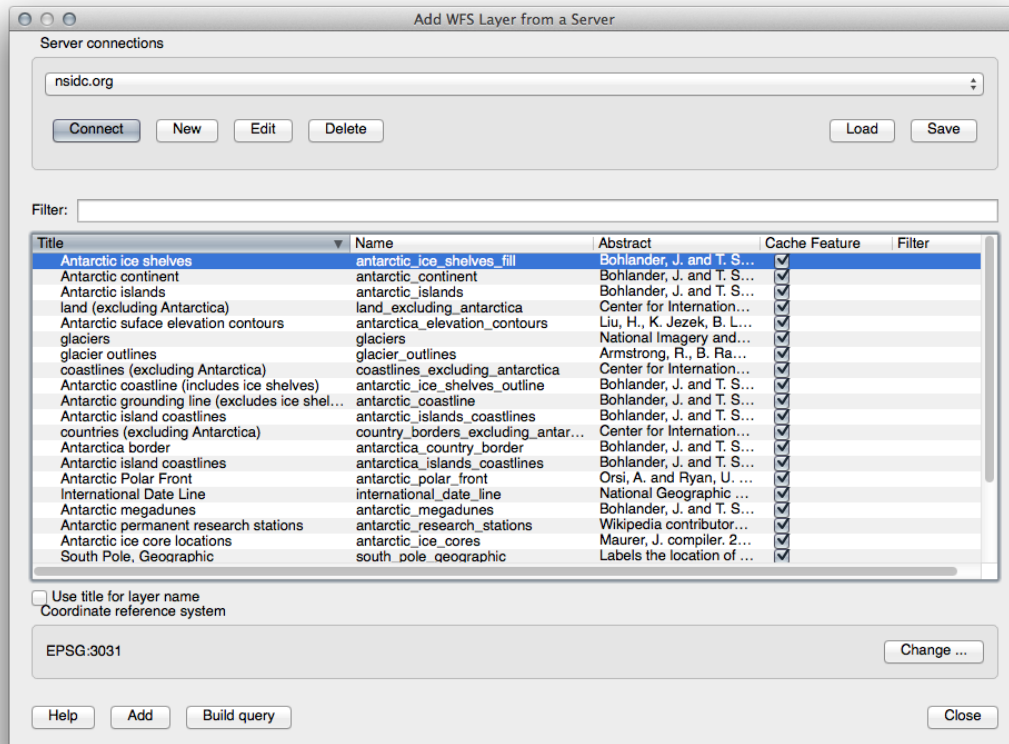
- Start a new map. This is for demo purposes and won't be saved.
- Ensure that “on the fly” re-projection is switched off.
- Click the *Add WFS Layer* button:



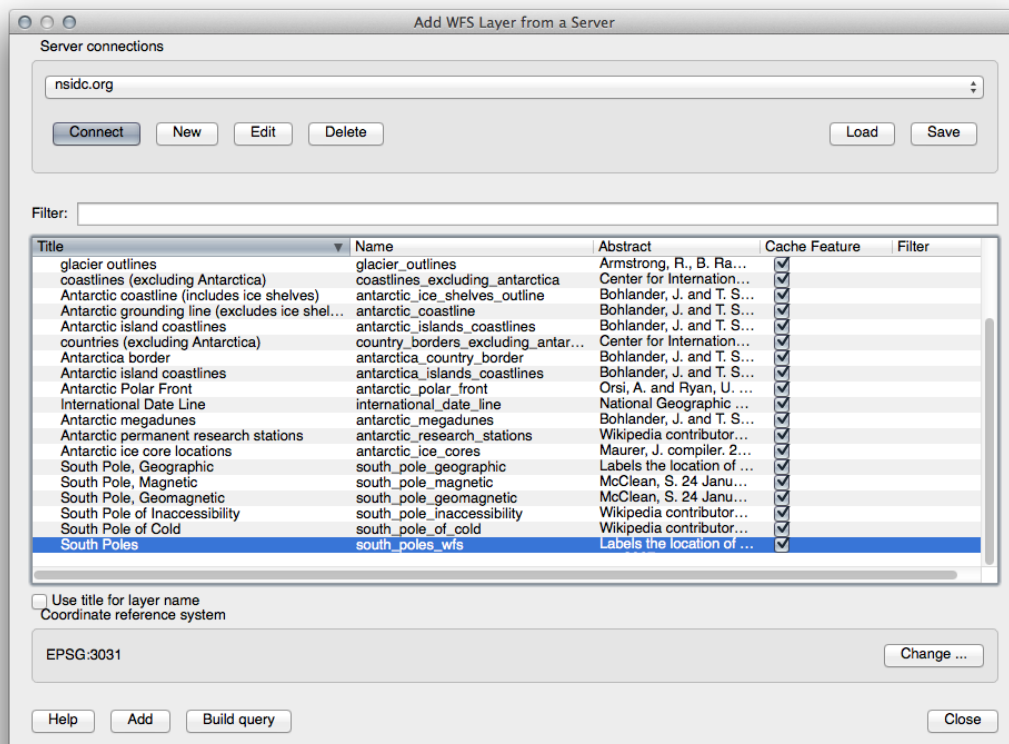
- Click the *New* button.
- In the dialog that appears, enter the *Name* as `nsidc.org` and the *URL* as `http://nsidc.org/cgi-bin/atlas_south?version=1.1.0`.



- Click *OK*, and the new connection will appear in your *Server connections*.
- Click the *Connect*. A list of the available layers will appear:

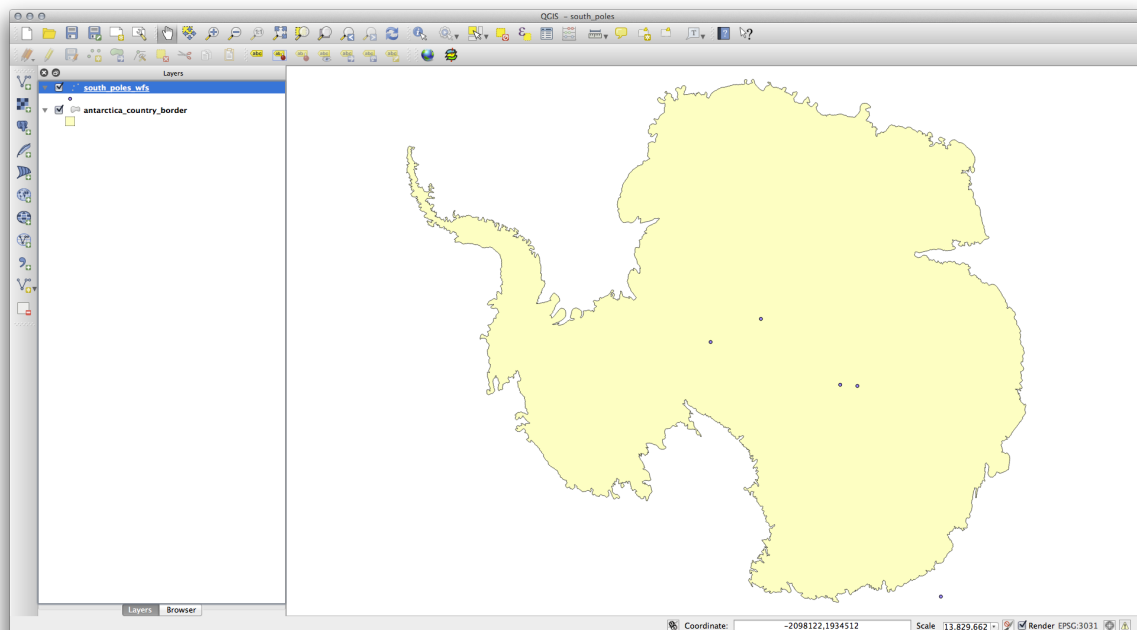


- Find the layer *south_poles_wfs*.
- Click on the layer to select it:



- Click *Add*.

It may take a while to load the layer. When it has loaded, it will appear in the map. Here it is over the outlines of Antarctica (available on the same server, and by the name of *antarctica_country_border*):



How is this different from having a WMS layer? That will become obvious when you see the layers' attributes.

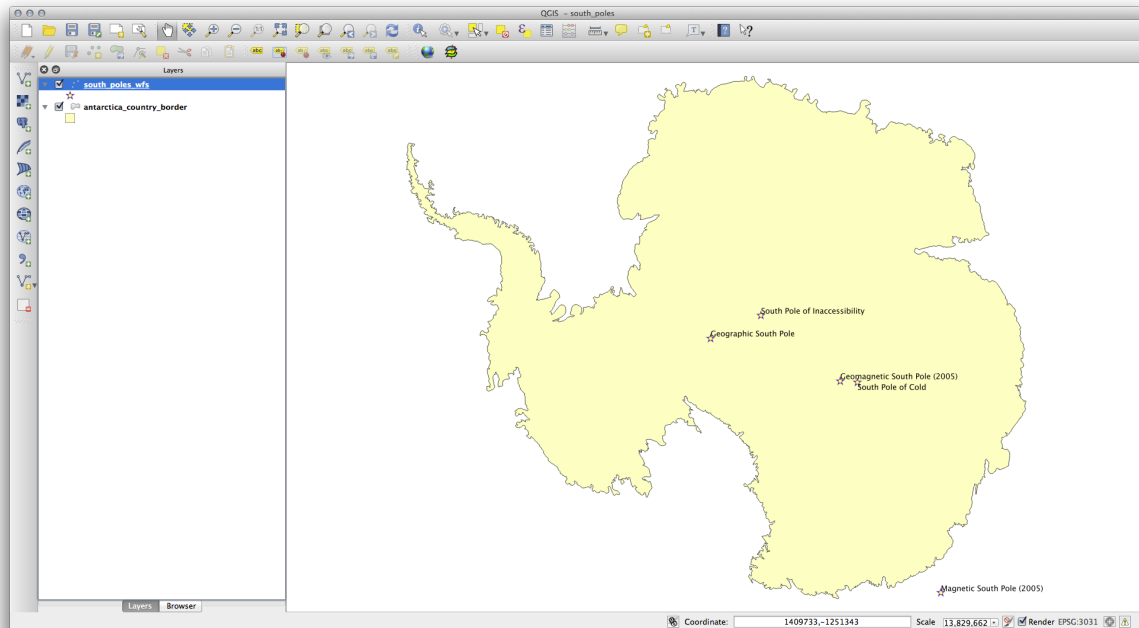
- Open the *south_poles_wfs* layer's attribute table. You should see this:

Attribute table – south_poles_wfs :: Features total: 5, filtered: 5, selected: 0

| | Id | NAME |
|---|----|-------------------------------|
| 0 | 0 | Geographic South Pole |
| 1 | 0 | Magnetic South Pole (2005) |
| 2 | 0 | Geomagnetic South Pole (2005) |
| 3 | 0 | South Pole of Inaccessibility |
| 4 | 0 | South Pole of Cold |

Show All Features

Since the points have attributes, we are able to label them, as well as change their symbology. Here's an example:



- Add labels to your layer to take advantage of the attribute data in this layer.

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.

11.2.2 Follow Along: Querying a WFS Layer

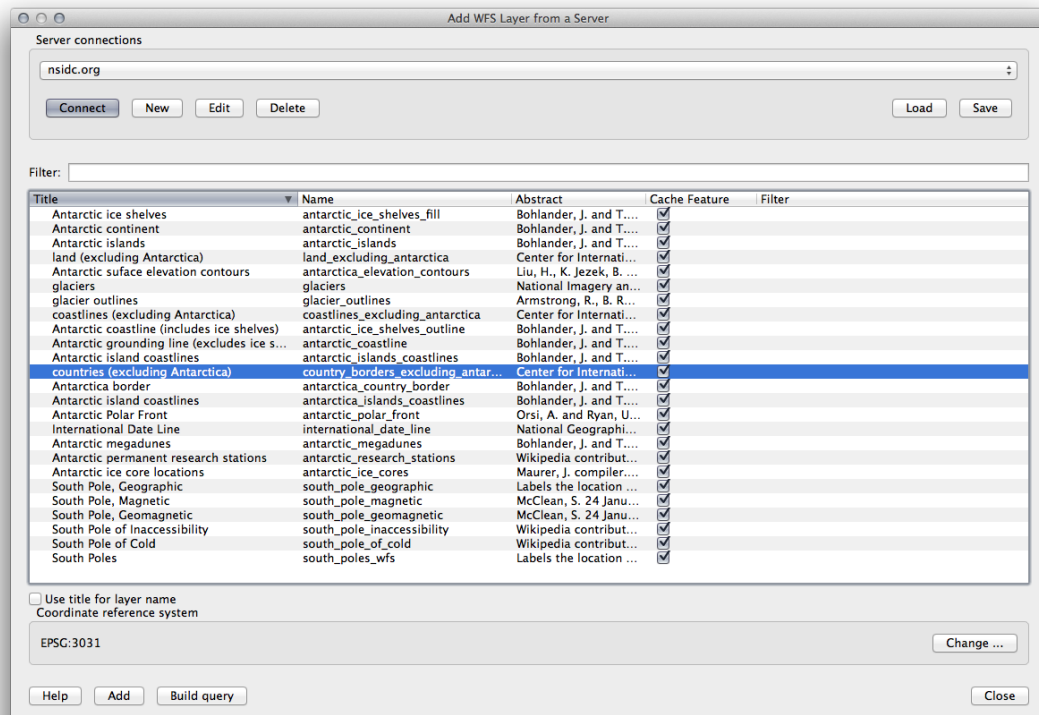
Although it is possible to query a WFS layer after having loaded it, it's often more efficient to query it before you load it. That way, you're only requesting the features you want, meaning that you use far less bandwidth.

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.

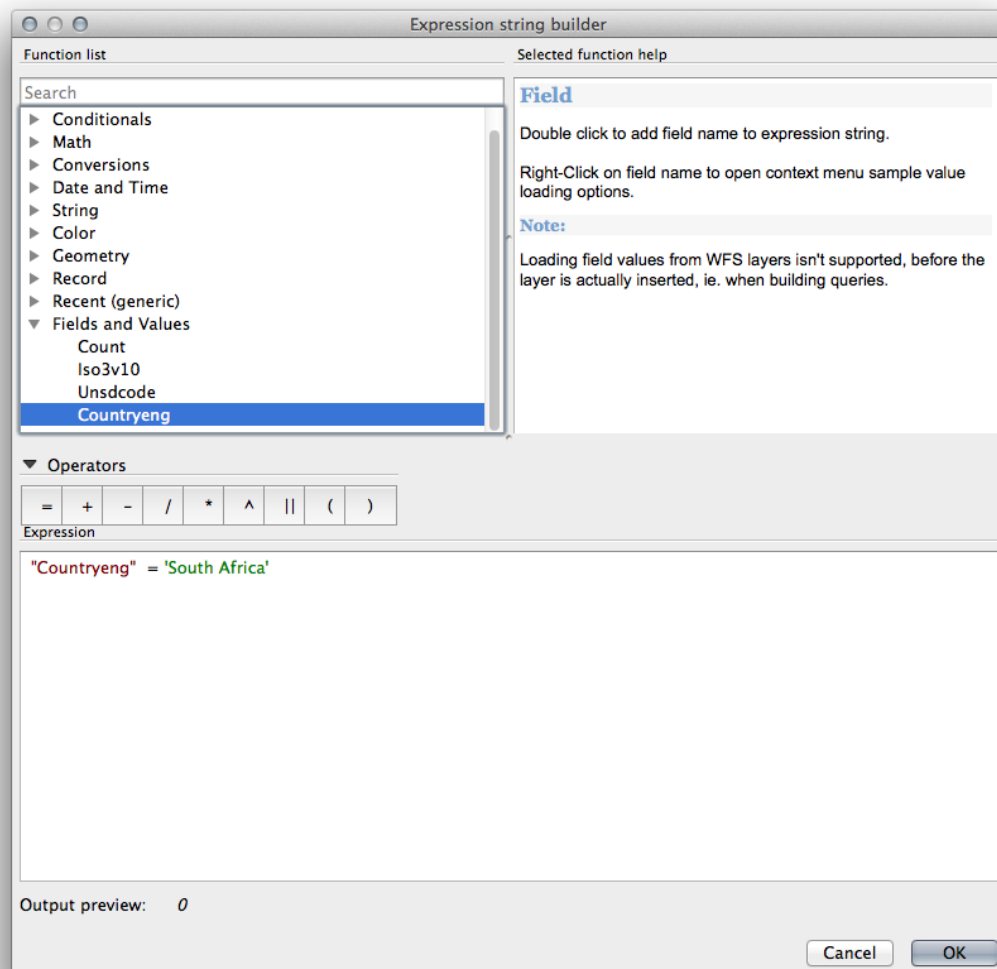
There are two ways to do this. You can load the whole *countries ...* layer, and then build a query as usual once it's loaded. However, transmitting the data for all the countries in the world and then only using the data for South Africa seems a bit wasteful of bandwidth. Depending on your connection, 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.

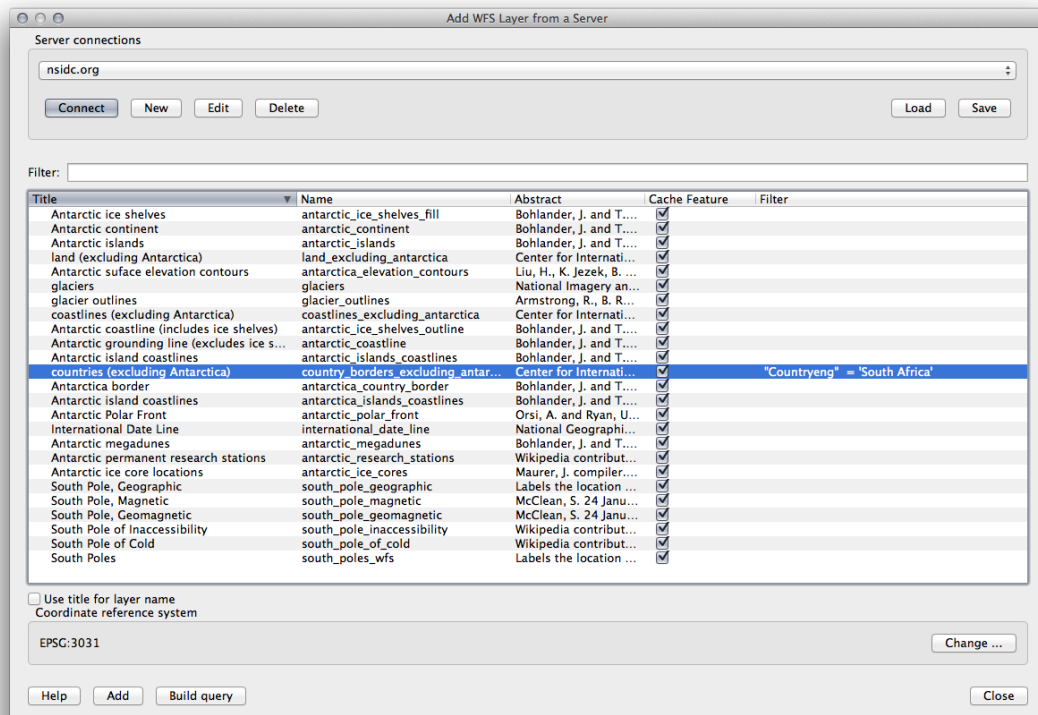
- In the *Add WFS Layer ...* dialog, connect to the server we used before and you should see the list of available layers.
- Double-click next to the *countries ...* layer in the *Filter* field, or click *Build query*:



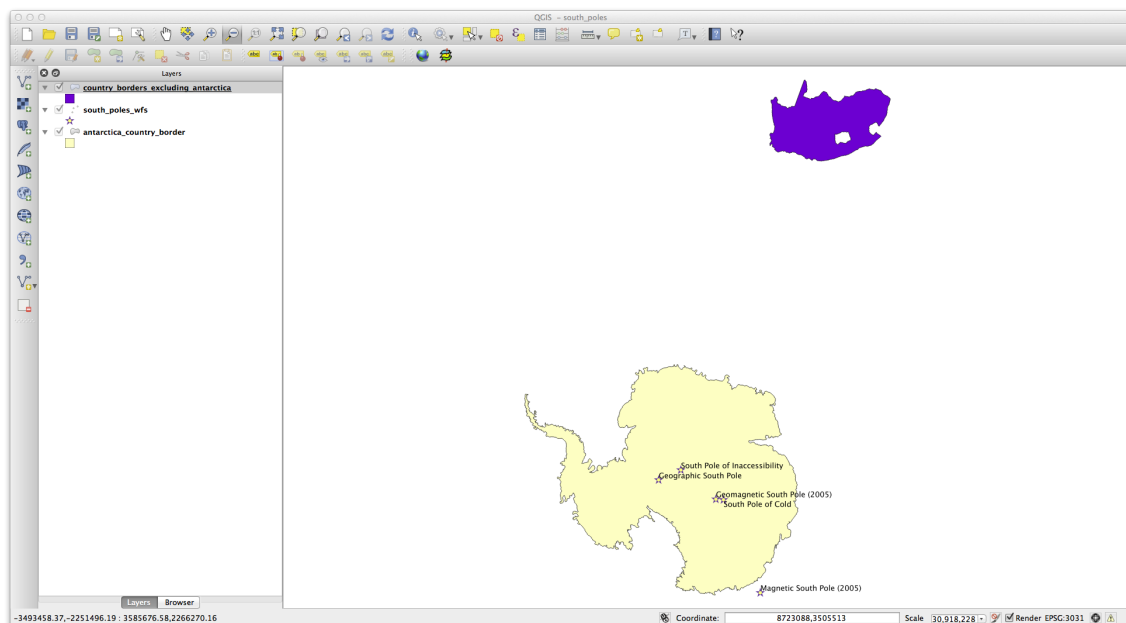
- In the dialog that appears, build the query "Countryeng" = 'South Africa':



- It will appear as the *Filter* value:



- Click *Add* with the *countries* layer selected as above. Only the country with the Countryeng value of South Africa will load from that layer:



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.

11.2.3 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.

11.2.4 What's Next?

Next, you'll see how to use QGIS as a frontend for the famous GRASS GIS.

Module: GRASS

GRASS (Geographic Resources Analysis Support System) is a well-known open source GIS with a wide array of useful GIS functions. It was first released in 1984, and has seen much improvement and additional functionality since then. QGIS allows you to make use of GRASS' powerful GIS tools directly.

12.1 Lesson: GRASS Setup

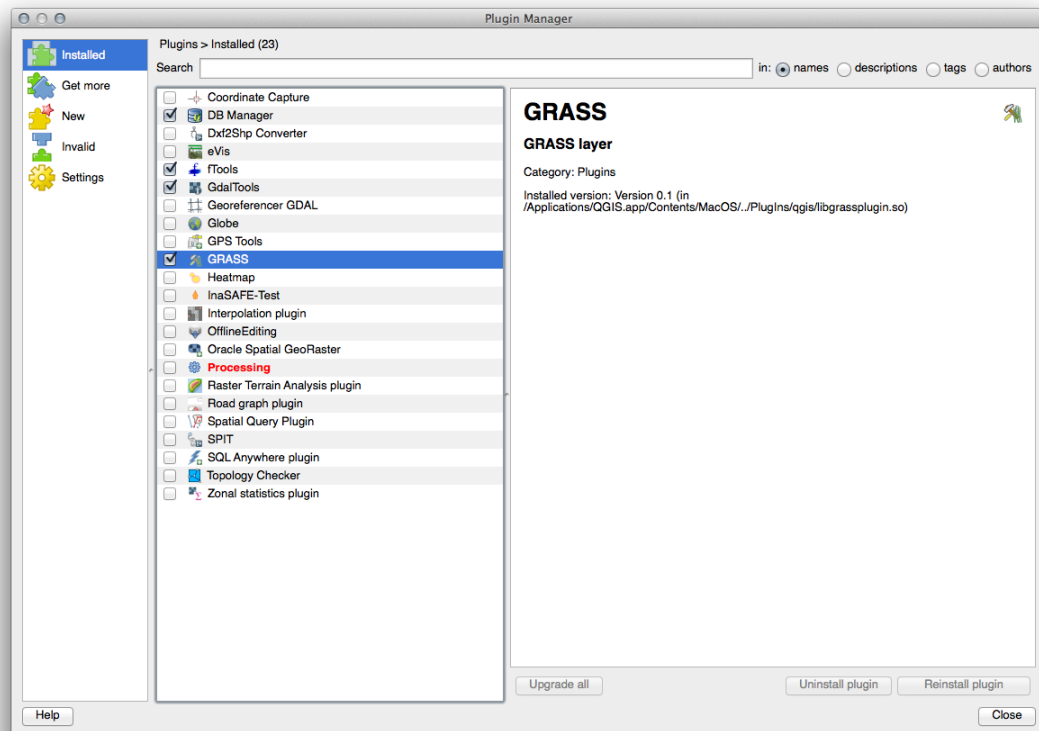
Using GRASS in QGIS requires you to think of the interface in a slightly different way. Remember that you're not working in QGIS directly, but working in GRASS *via* QGIS.

The goal for this lesson: To begin a GRASS project in QGIS.

12.1.1 Follow Along: Start a New GRASS Project

To launch GRASS from within QGIS, you need to activate it as with any other plugin. First, open a new QGIS project.

- In the *Plugin Manager*, enable *GRASS* in the list:



The GRASS toolbar will appear:



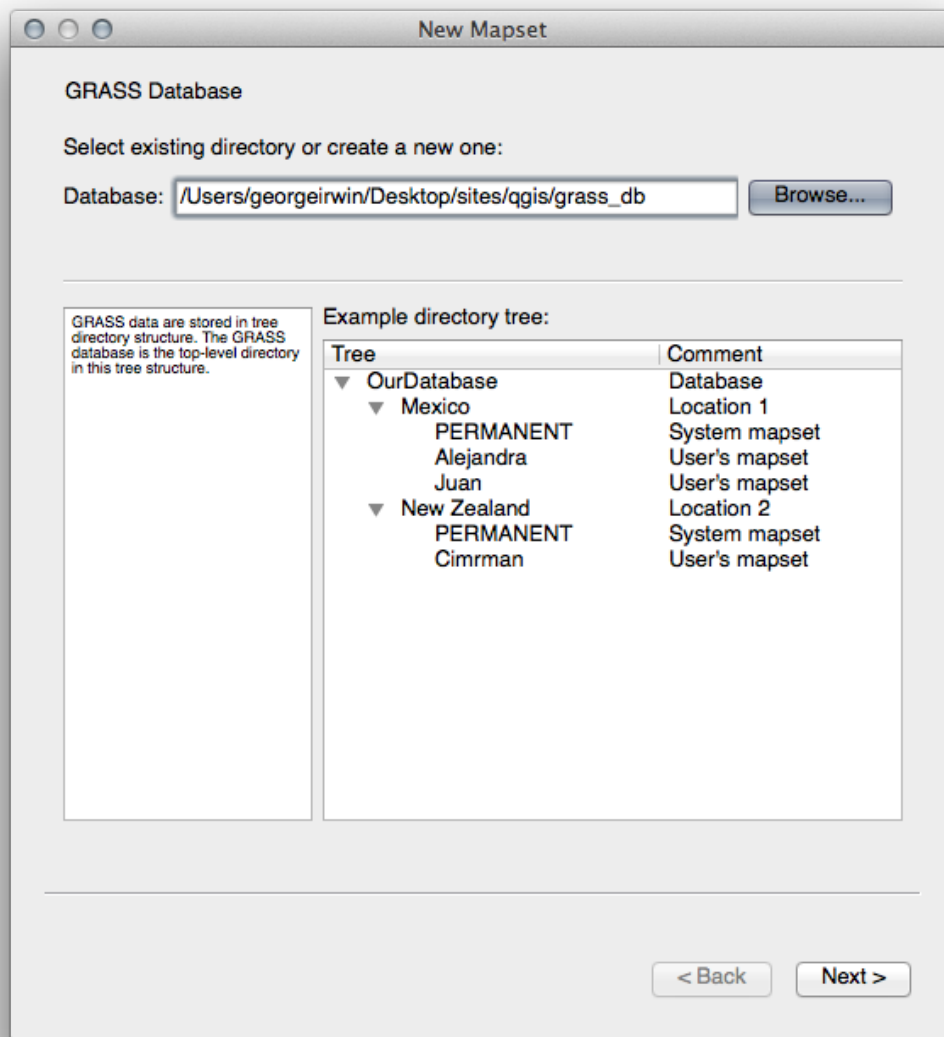
Before you can use GRASS, you need to create a **mapset**. GRASS always works in a database environment, which means that you need to import all the data you want to use into a GRASS database.

- Click on the *New mapset* button:



You'll see a dialog explaining the structure of a GRASS mapset.

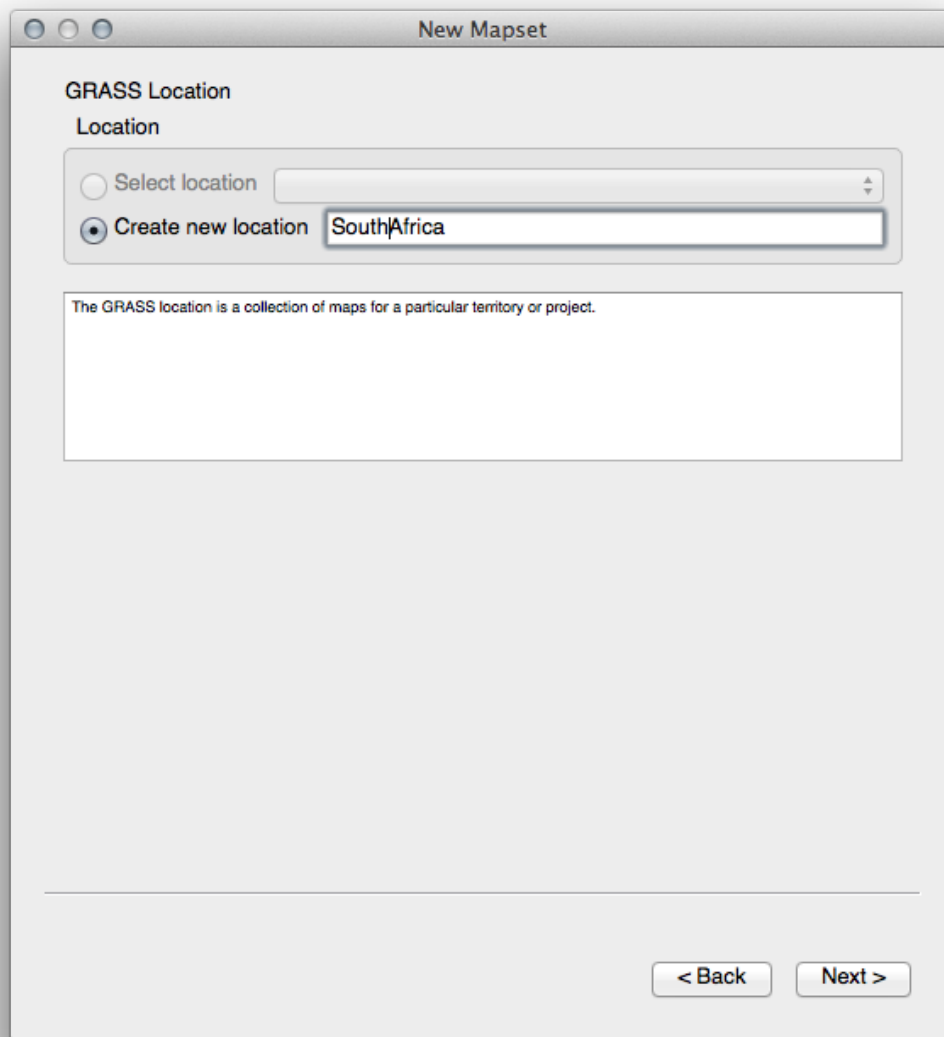
- Create a new directory called `grass_db` in *exercise_data*.
- Set it as the directory that will be used by GRASS to set up its database:



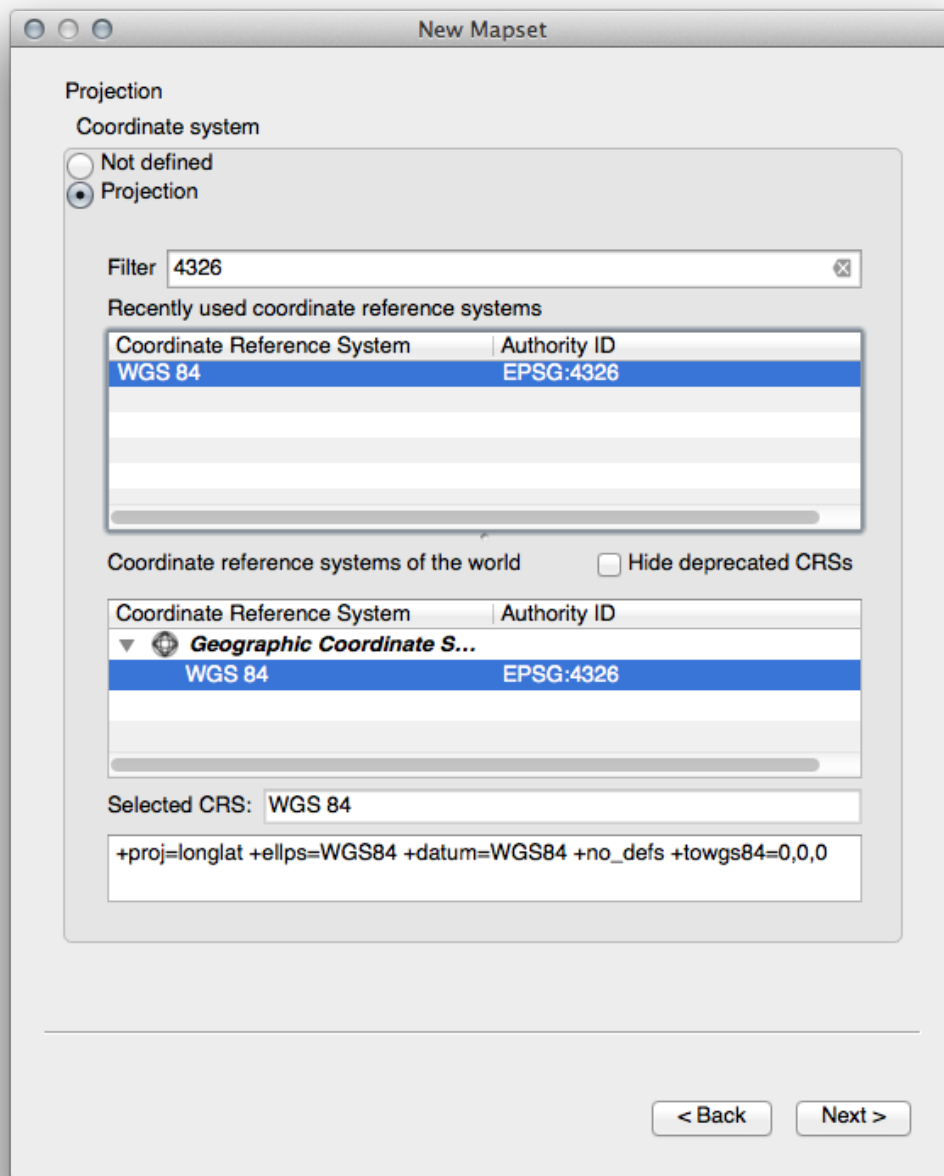
- Click *Next*.

GRASS needs to create a “location”, which describes the maximum extents of the geographic area you’ll be working in.

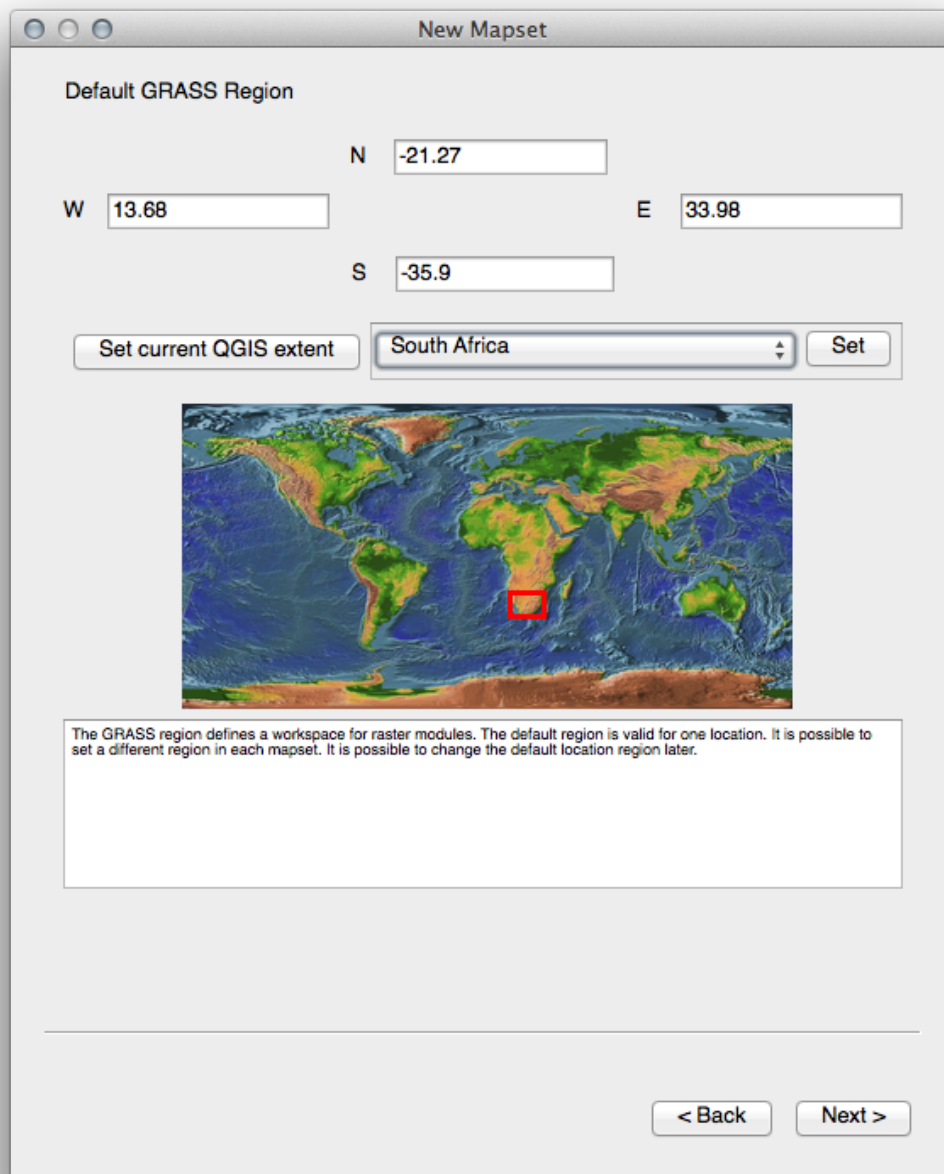
- Call the new location `South_Africa`:



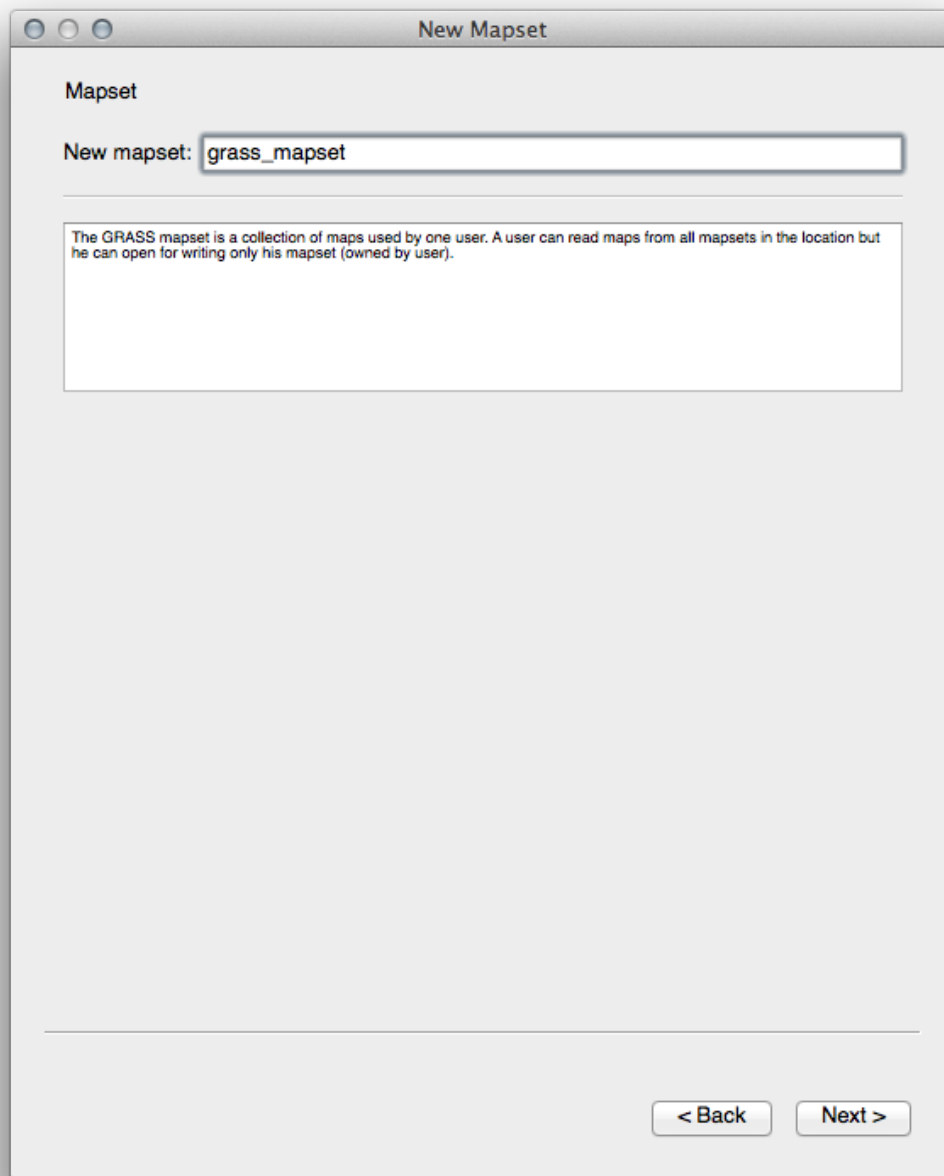
- Click *Next*.
- We'll be working with WGS 84, so search for and select this CRS:



- Click *Next*.
- Now select the region *South Africa* from the dropdown and click *Set*:



- Click *Next*.
- Create a mapset, which is the map file that you'll be working with.



Once you're done, you'll see a dialog asking you to confirm that the settings it displays are correct.

- Click *Finish*.
- Click *OK* on the success dialog.

12.1.2 Follow Along: Loading Vector Data into GRASS

You'll now have a blank map. To load data into GRASS, you need to follow a two-step process.

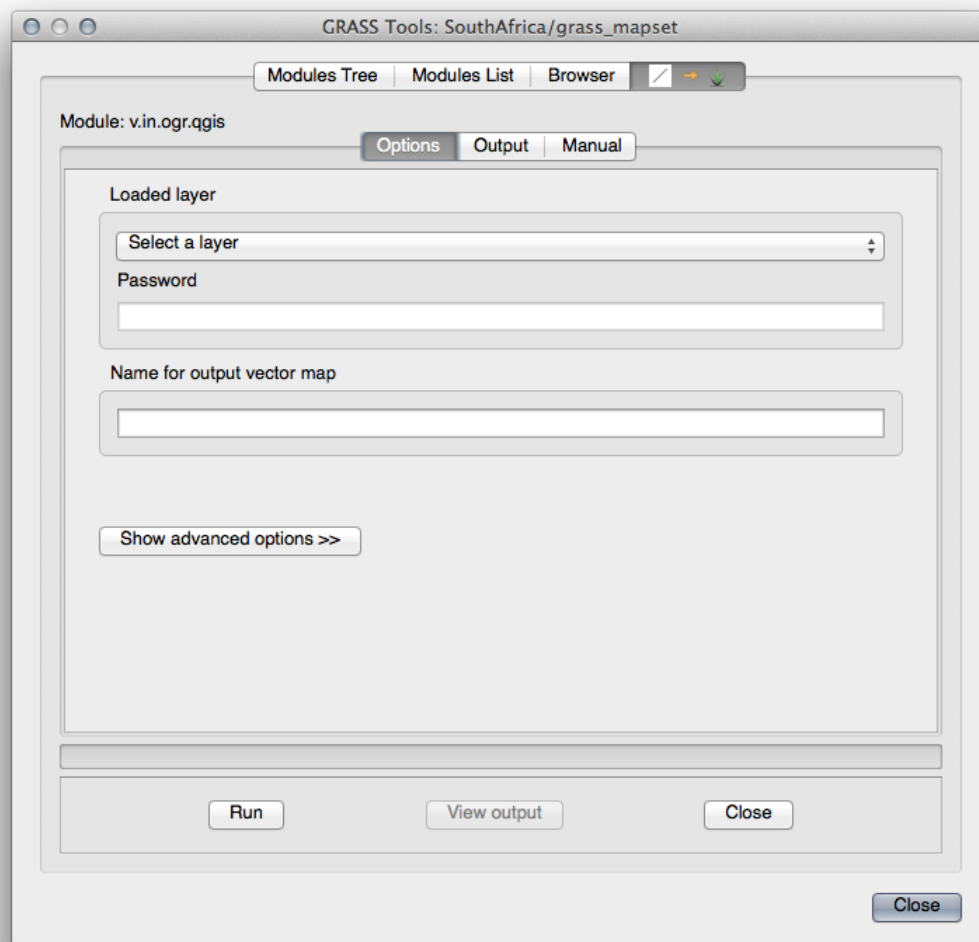
- Load data into QGIS as usual. Use the `roads.shp` dataset (found under `exercise_data/epsg4326/`) for now.
- As soon as it's loaded, click on the *GRASS Tools* button:



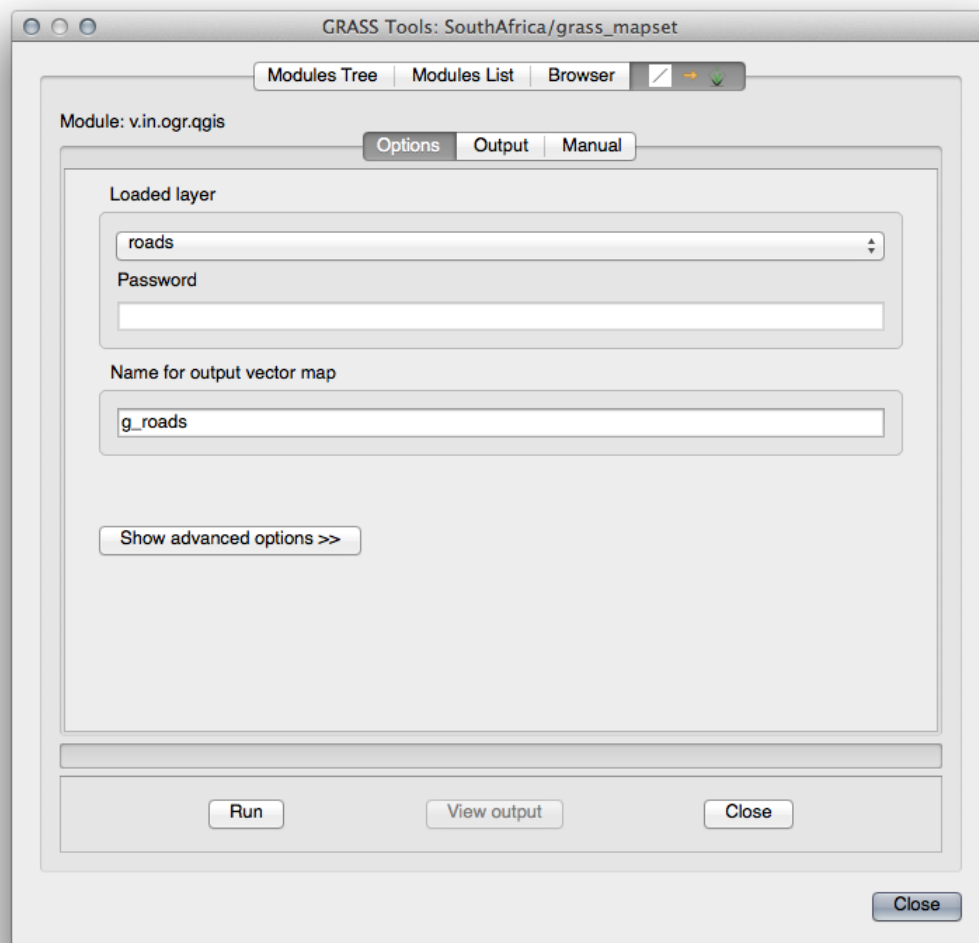
- In the new dialog, select *Modules list*.
- Find the vector import tool by entering the term `v.in.ogr.qgis` in the *Filter* field.


The `v` stands for “vector”, `in` means its 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.

- Once you’ve found this tool, click on it to bring up the tool itself:



- Set the loaded layer to *roads* and its GRASS version’s name to `g_roads` to prevent confusion.



Bemerkung:  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.

- Click *Run* to begin the import.
- When it's done, click the *View output* button to see the newly imported GRASS layer in the map.
- Close first the import tool (click the *Close* button to the immediate right of *View output*), then close the *GDAL Tools* window.
- Remove the original *roads* layer.

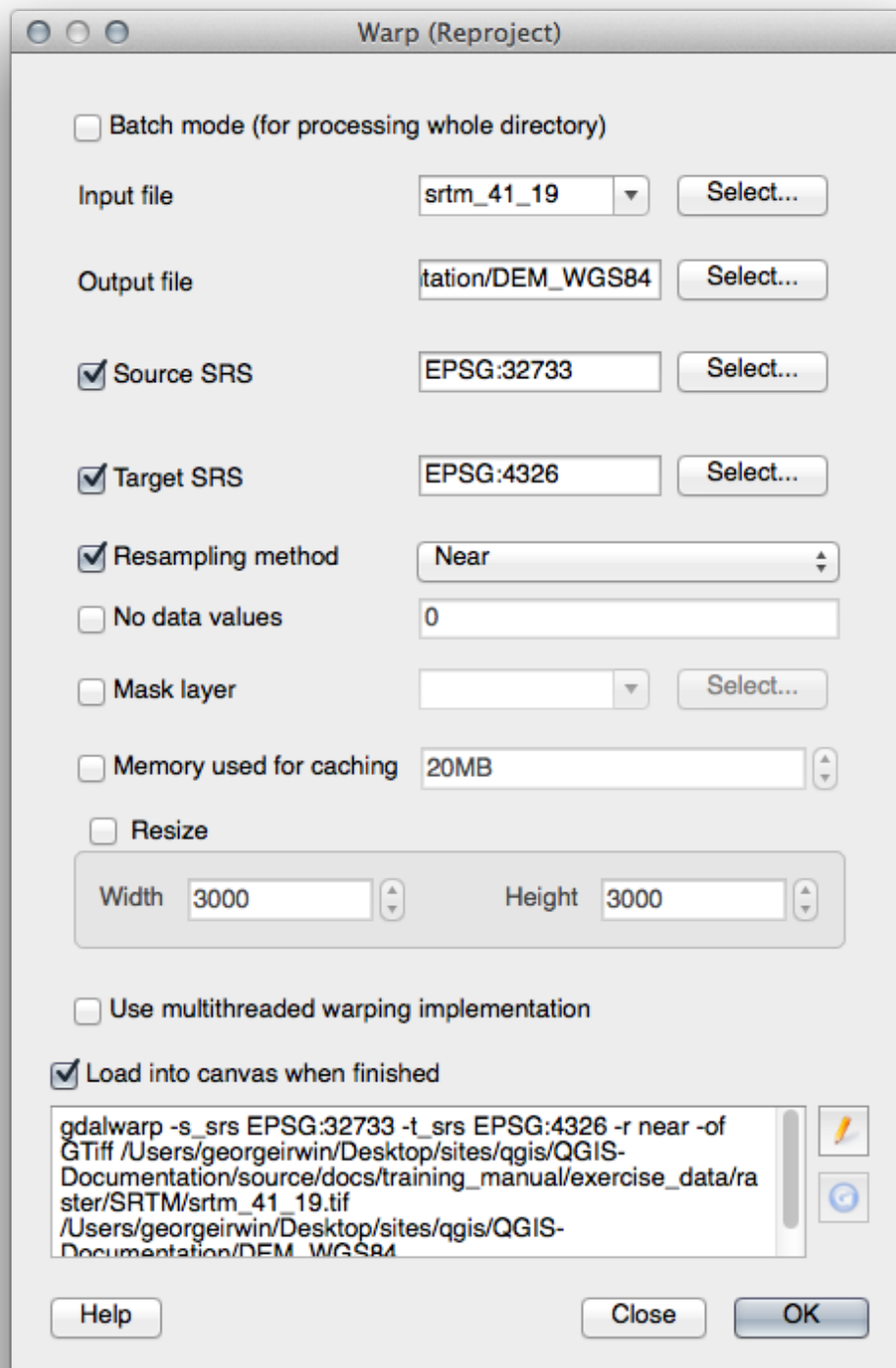
Now you are left with only the imported GRASS layer as displayed in your QGIS map.

12.1.3 Follow Along: Loading Raster Data into GRASS

Recall that our DEM is in the Projected CRS UTM 33S / WGS 84, but our GRASS project is in the Geographic CRS WGS 84. So let's re-project the DEM first.

- Load the `srtm_41_19.tif` dataset (found under `exercise_data/raster/SRTM/`) into the QGIS map as usual, using QGIS' *Add Raster Layer* tool.

- Re-project it using GDAL Warp tool (*Raster → Projections → Warp (Reproject)*), setting it up as shown:

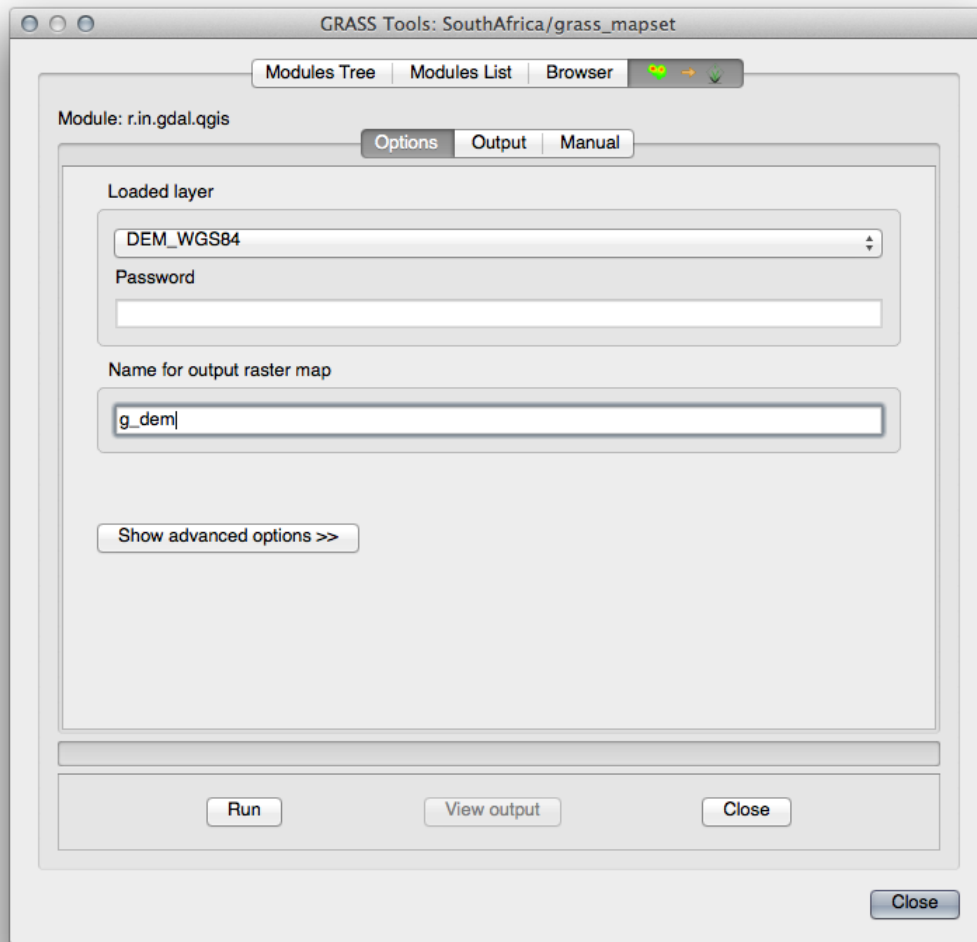


- Save the raster under the same folder as the original, but with the file name `DEM_WGS84.tif`. Once it appears in your map, remove the `srtm_41_19.tif` dataset from your *Layers list*.

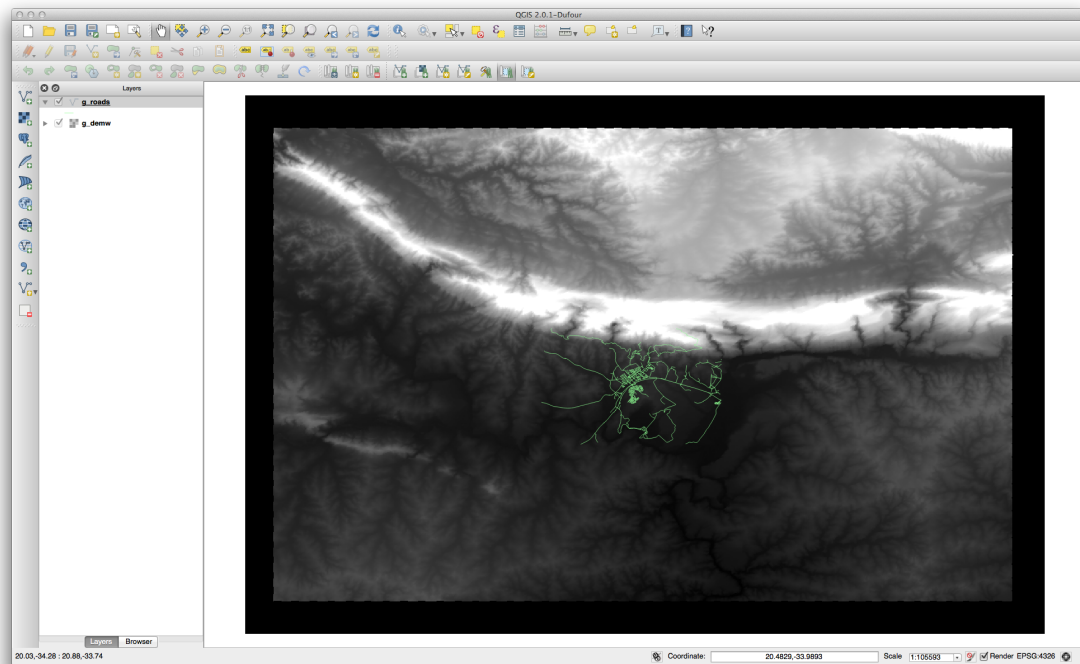
Now that it's reprojected, you can load it into your GRASS database.

- Open the *GRASS Tools* dialog again.

- Click on the *Modules List* tab.
- Search for `r.in.gdal.qgis` and double click the tool to open the tool's dialog.
- Set it up so that the input layer is *DEM_WGS84* and the output is *g_dem*.



- Click *Run*.
- When the process is done, click *View output*.
- *Close* the current tab, and then *Close* the whole dialog.



- You may now remove the original *DEM_WGS84* layer.

12.1.4 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.

12.1.5 What's Next?

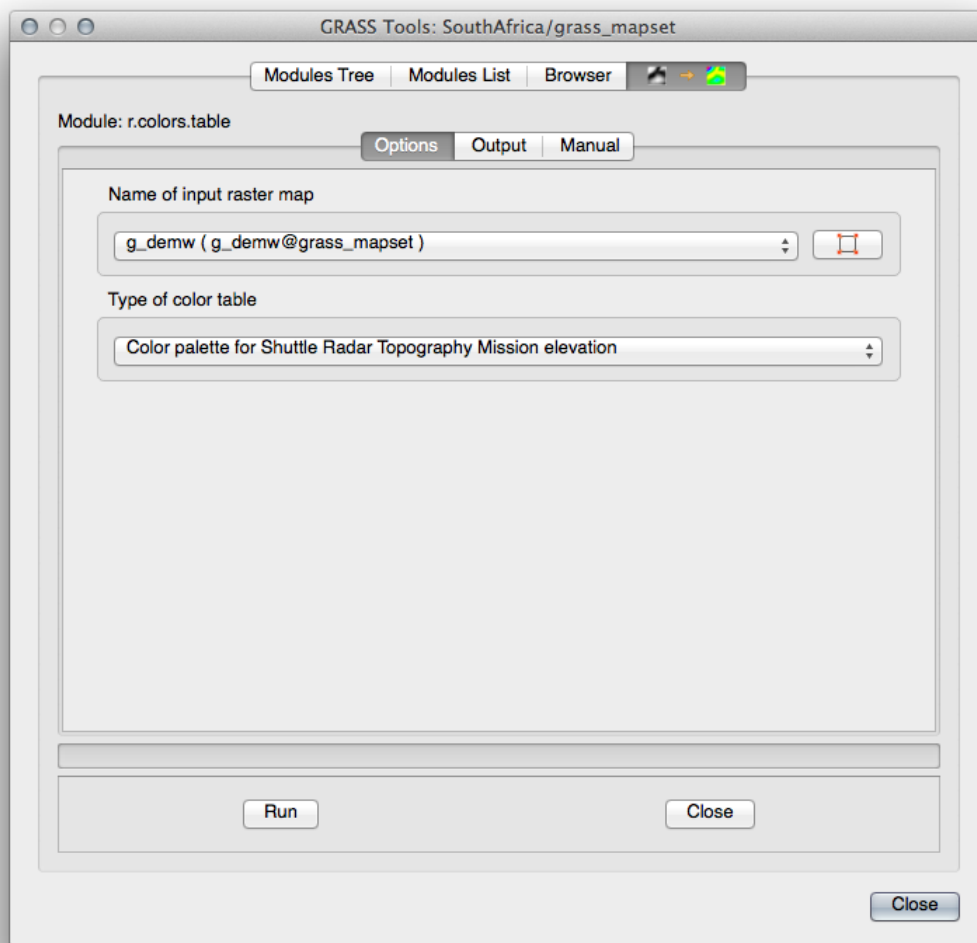
Now that the data is imported into GRASS, we can look at the advanced analysis operations that GRASS offers.

12.2 Lesson: GRASS Tools

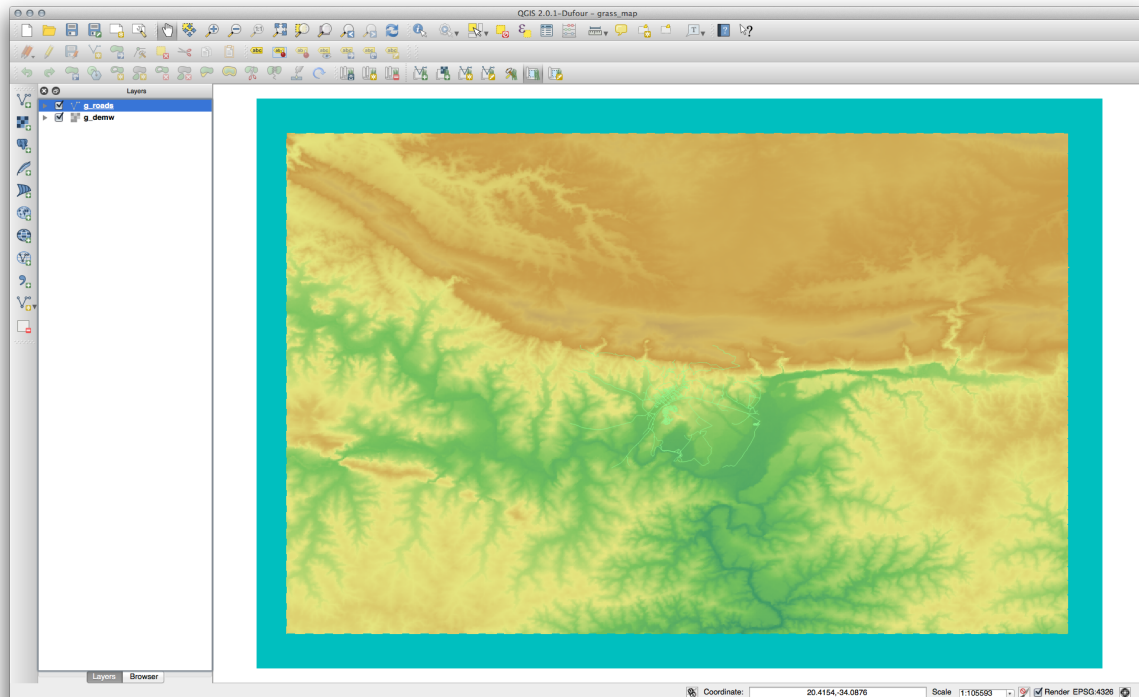
In this lesson we will present a selection of tools to give you an idea of the capabilities of GRASS.

12.2.1 Follow Along: Set Raster Colors

- Open the *GRASS Tools* dialog.
- Look for the *r.colors.table* module by searching for it in the *Filter* field of the *Modules List* tab.
- Open the tool and set it up like this:



When you run the tool, it will recolor your raster:



12.2.2 Follow Along: Visualize Data in 3D

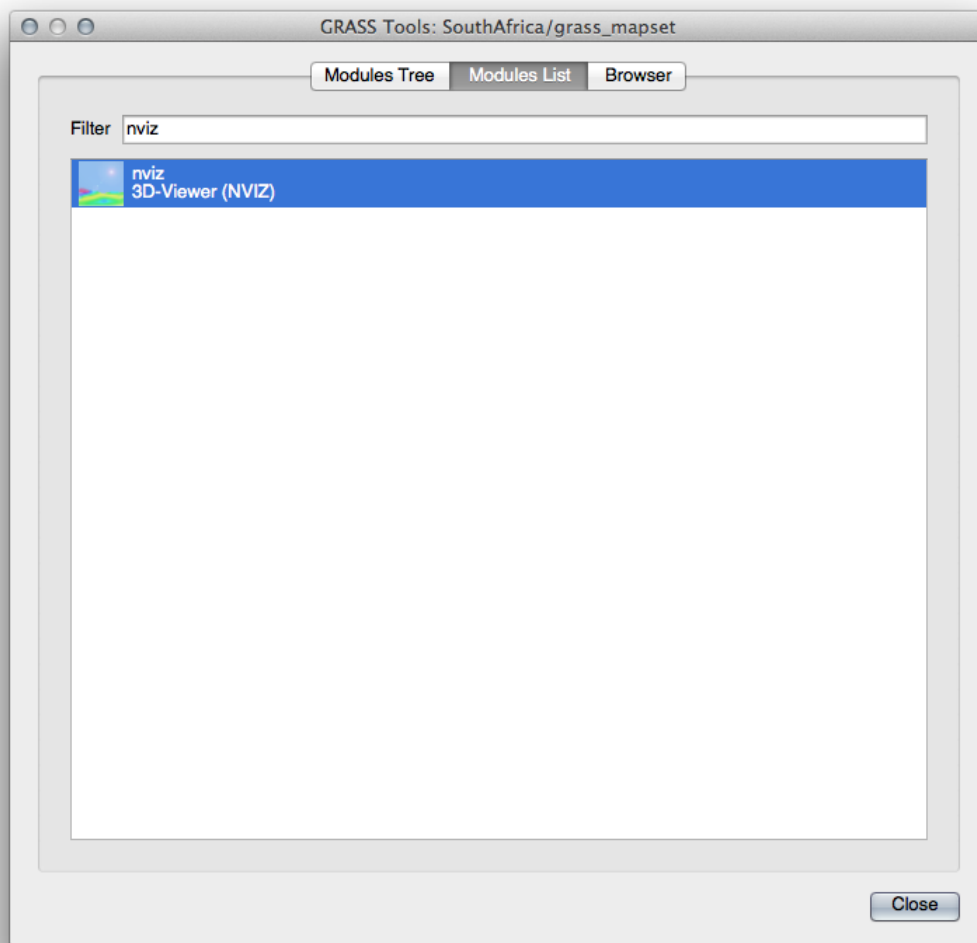
GRASS allows you to use a DEM to visualize your data in three dimensions. The tool you'll use for this operates on the GRASS Region, which at the moment is set to the whole extent of South Africa, as you set it up before.

- To redefine the extent to cover only our raster dataset, click this button:

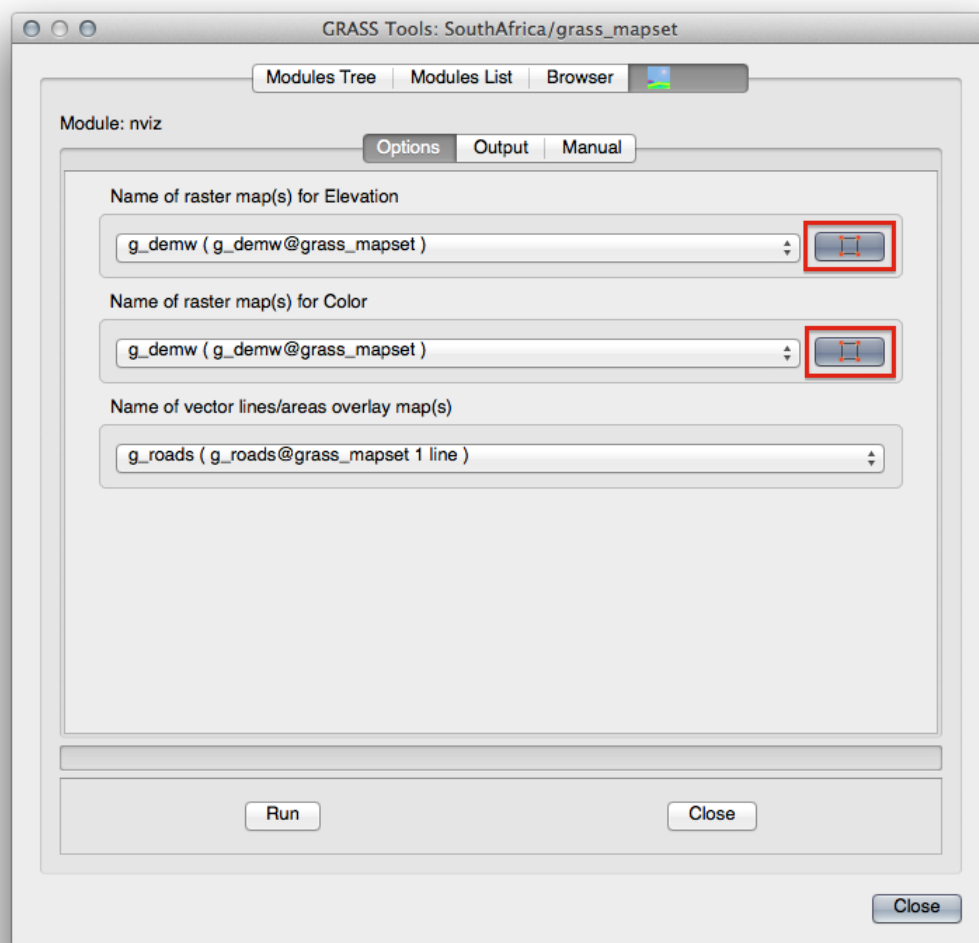


When this tool is activated, your cursor will turn into a cross whe over the QGIS map canvas.

- Using this tool, click and drag a rectangle around the edges of the GRASS raster.
- Click *OK* in the *GRASS Region Settings* dialog when done.
- Search for the `nviz` tool:

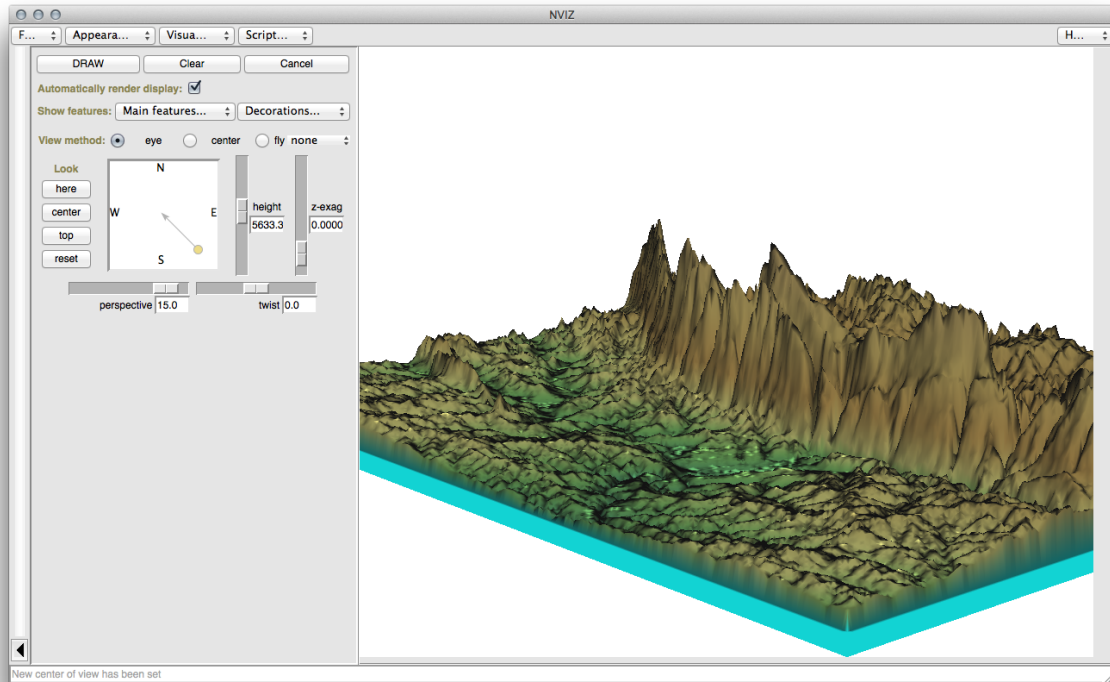


- Set it up as shown:



- Remember to enable both *Use region of this map* buttons to the right of the two raster selection dropdown menus. This will allow NVIZ to correctly assess the resolution of the rasters.
- Click the *Run* button.

NVIZ will set up a 3D environment using the raster and vector selected. This may take some time, depending on your hardware. When it's done, you will see the map rendered in 3D in a new window:



Experiment with the *height*, *z-exag*, and *View method* settings to change your view of the data. The navigation methods may take some getting used to.

After experimenting, close the NVIZ window.

12.2.3 Follow Along: The Mapcalc Tool

- Open the *GRASS Tools* dialog's *Modules List* tab and search for `calc`.
- From the list of modules, select *r.mapcalc* (not *r.mapcalculator*, which is more basic).
- 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:

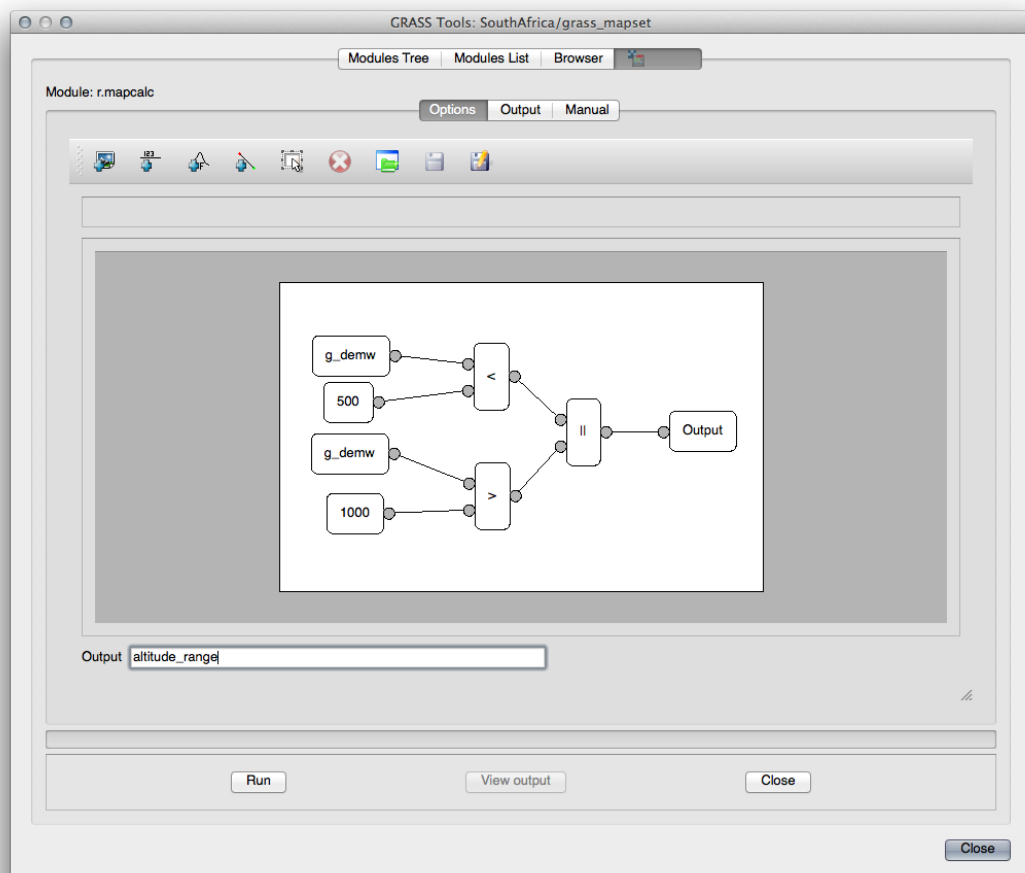


In order, they are:

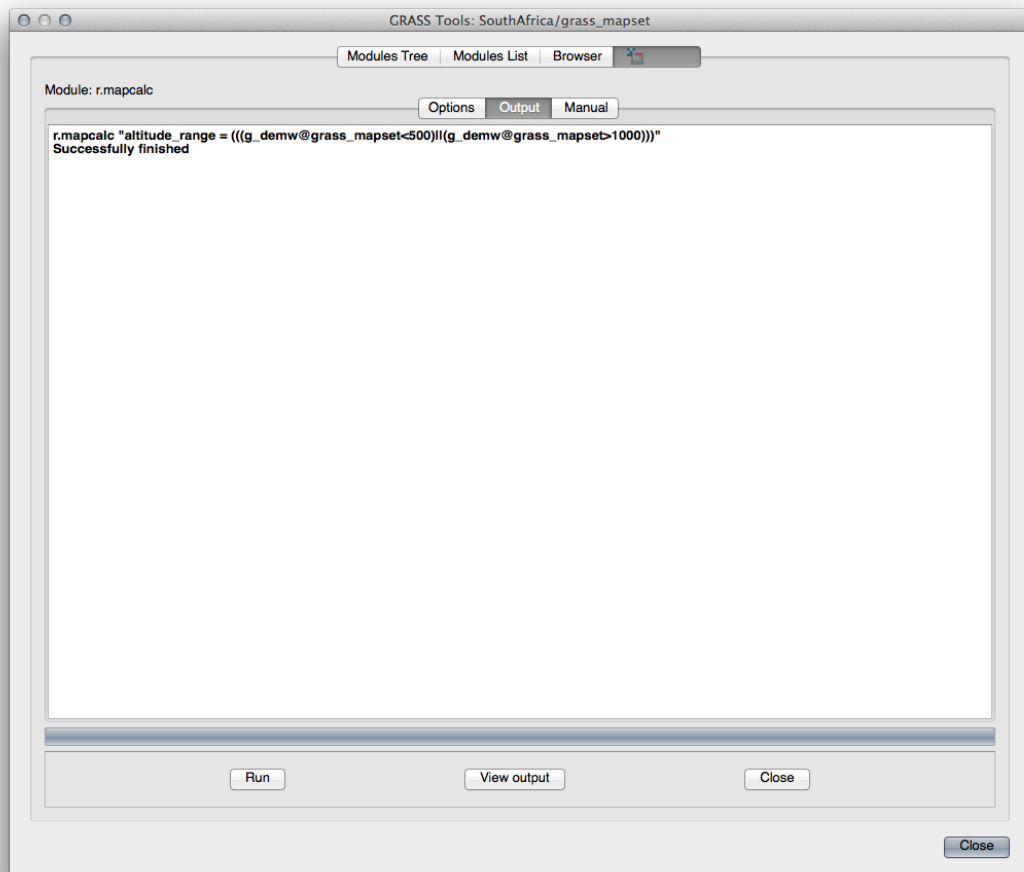
- Add `map`: Add a raster file from your current GRASS mapset.
- Add `constant value`: Add a constant value to be used in functions.
- Add `operator or function`: Add an operator or function to be connected to inputs and outputs.
- Add `connection`: Connect elements. Using this tool, click and drag from 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).

Using these tools:

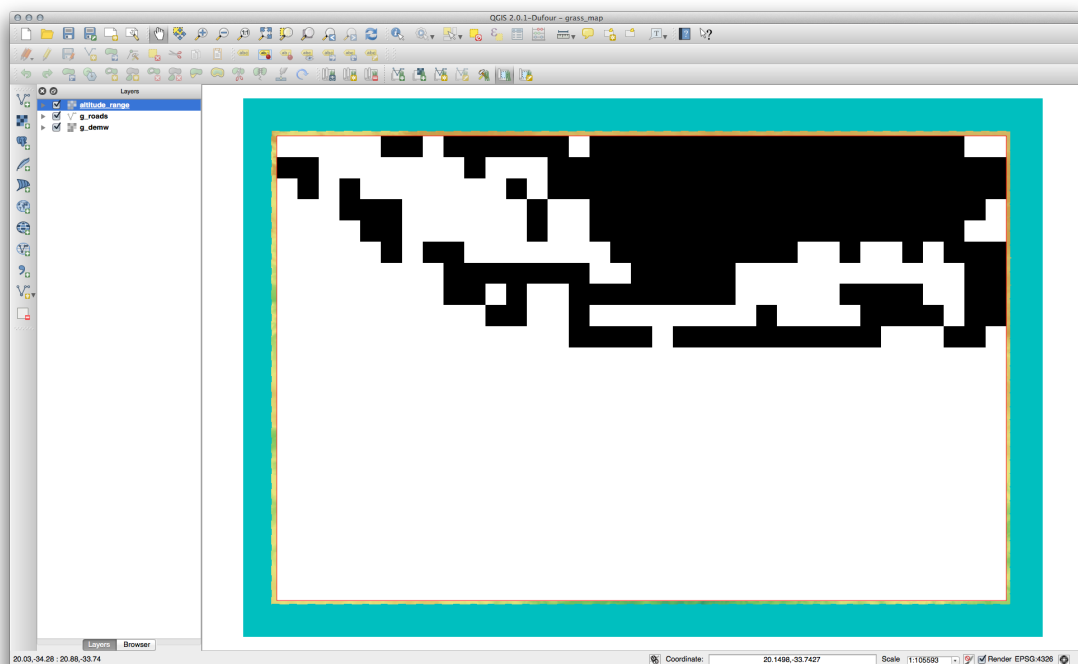
- Construct the following algorithm:



- When you click *Run*, your output should look like this:



- Click *View output* to see the output displayed in your map:



This shows all the areas where the terrain is lower than 500 meters or higher than 1000 meters.

12.2.4 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.

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 [this URL](#) 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 you 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 *Map Composer* 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.

Module: Database Concepts with PostgreSQL

Relational Databases are an important part of any GIS system. In this module, you'll learn about Relational Database Management System (RDBMS) concepts and you will use PostgreSQL to create a new database to store data, as well as learning about other typical RDBMS functions.

14.1 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.

14.1.1 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*

14.1.2 Tables

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**.

14.1.3 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 |
```

14.1.4 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
```

14.1.5 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!](#)

14.1.6 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.

Check your results

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!

14.1.7 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.

14.1.8 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 |
|----------|------------------------|--|
| 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.

14.1.9 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](#)

Check your results

14.1.10 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)
```

14.1.11 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 |

14.1.12 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.

Check your results

14.1.13 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)

14.1.14 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.

14.1.15 In Conclusion

Databases allow you to manage data in a structured way using simple code structures.

14.1.16 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.

14.2 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.

14.2.1 Install PostgreSQL

Bemerkung: Although outside the scope of this document, Mac users can install PostgreSQL using [Homebrew](#). Windows users can use the graphical installer located here: <http://www.postgresql.org/download/windows/>. Please note that the documentation will assume users are running QGIS under Ubuntu.

Under Ubuntu:

```
sudo apt-get 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.
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.

14.2.2 Help

PostgreSQL has very good [online](#) documentation.

14.2.3 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
```

14.2.4 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.

14.2.5 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.

14.2.6 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 - available online [here](#).

14.2.7 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-types as above.

Check your results

14.2.8 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) | |
| 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)

14.2.9 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)

14.2.10 Dropping Tables in SQL

If you want to get rid of a table you can use the drop command:

```
drop table streets;
```

Bemerkung: In our current example, the above command would not work. Why not? *See why*

If you used the same `drop table` command on the *people* table, it would be successful:

```
drop table people;
```

Bemerkung: 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.

14.2.11 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 pgAdminIII 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-get install pgadmin3
```

pgAdmin III will be covered in more detail in another module.

14.2.12 In Conclusion

You have now seen how to create a brand new database, starting completely from scratch.

14.2.13 What's Next?

Next you'll learn how to use the DBMS to add new data.

14.3 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.

14.3.1 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.

Check your results

14.3.2 Sequencing Data Addition According to Constraints

14.3.3 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
```

Bemerkung: 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.

Check your results

14.3.4 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.

14.3.5 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.`

Bemerkung: the `WHERE` statement criteria are case sensitive `Main Road` is not the same as `Main road`

14.3.6 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)
```

14.3.7 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 |

14.3.8 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.

14.3.9 What's Next?

Now that you've added some data, you'll learn how to use queries to access this data in various ways.

14.4 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.

Bemerkung: 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');
```

14.4.1 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 |
| 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)

14.4.2 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`):

```
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.

14.4.3 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;
```

Bemerkung: 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.

14.4.4 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.

14.4.5 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)

Bemerkung: 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.

Check your results

14.4.6 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.

14.4.7 What's Next?

Next you'll see how to create views from the queries that you've written.

14.5 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.

14.5.1 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)
```

14.5.2 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)
```

14.5.3 Dropping a View

If you no longer need a view, you can delete it like this:

```
drop view roads_count_v;
```

14.5.4 In Conclusion

Using views, you can save a query and access its results as if it were a table.

14.5.5 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.

14.6 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.

14.6.1 Materialised Views (Rule based views)

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)

Bemerkung: The value of the time field will depend on the current date and time.

14.6.2 In Conclusion

Rules allow you to automatically add or change data in your database to reflect changes in other parts of the database.

14.6.3 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.

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 from shapefiles 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 available from Boundless Geo:

- [Introduction to PostGIS](#)
- [Spatial Database Tips and Tricks](#)

See also [PostGIS online](#).

15.1 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.

Bemerkung: We will assume the use of PostGIS version 2.1 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.

15.1.1 Installing under Ubuntu

Postgis is easily installed from apt.

```
$ sudo apt-get install postgis
$ sudo apt-get install postgresql-9.1-postgis
```

Really, it's that easy...

Bemerkung: Depending on which version of Ubuntu you are using, and which repositories you have configured, these commands will install PostGIS 1.5, or 2.x. You can find the version installed by issuing a `select PostGIS_full_version();` query with `psql` or another tool.

To install the absolute latest version of PostGIS, you can use the following commands.

```
$ sudo apt-add-repository ppa:sharpie/for-science
$ sudo apt-add-repository ppa:sharpie/postgis-nightly
```

```
$ sudo apt-get update
$ sudo apt-get install postgresql-9.1-postgis-nightly
```

15.1.2 Installing under Windows

Installing on Windows is a little more complicated, but still not hard. Note that you need to be online to install the postgis stack.

First Visit [the download page](#).

Then follow [this guide](#).

More information about installing on Windows can be found on the [PostGIS website](#).

15.1.3 Installing on Other Platforms

The [PostGIS website](#) has information about installing on other platforms including MacOSX and on other linux distributions

15.1.4 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;"
```

Bemerkung: If you are using PostGIS 1.5 and a version of PostgreSQL lower than 9.1, you will need to follow a different set of steps in order to install the postgis extensions for your database. Please consult the [PostGIS Documentation](#) for instructions on how to do this. There are also some instructions in the [previous version](#) of this manual.

15.1.5 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
-----
010100000000000000000000F03F000000000000F03F
(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.

15.1.6 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 it's 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).

15.1.7 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.

15.1.8 What's Next?

Next you'll learn how spatial features are represented in a database.

15.2 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.

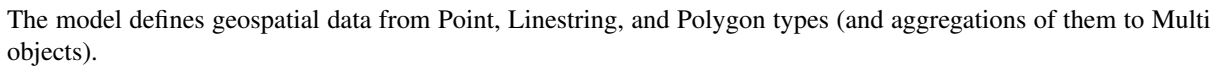
The goal for this lesson: To learn what the SFS Model is and how to use it.

15.2.1 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*

15.2.2 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.



Why? `geometry_columns` is used by certain applications to be aware of which tables in the database contain geometry data.

Bemerkung: If the above `INSERT` statement causes an error, run this query first:

```
select * from geometry_columns;
```

If the column `:kbd:'f_table_name'` contains the value `:kbd:'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

Check your results

15.2.7 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, the_geom)
values ('Fault Towers',
       34,
       3,
       '072 812 31 28',
       'SRID=4326;POINT(33 -33)');
```

Bemerkung: 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.

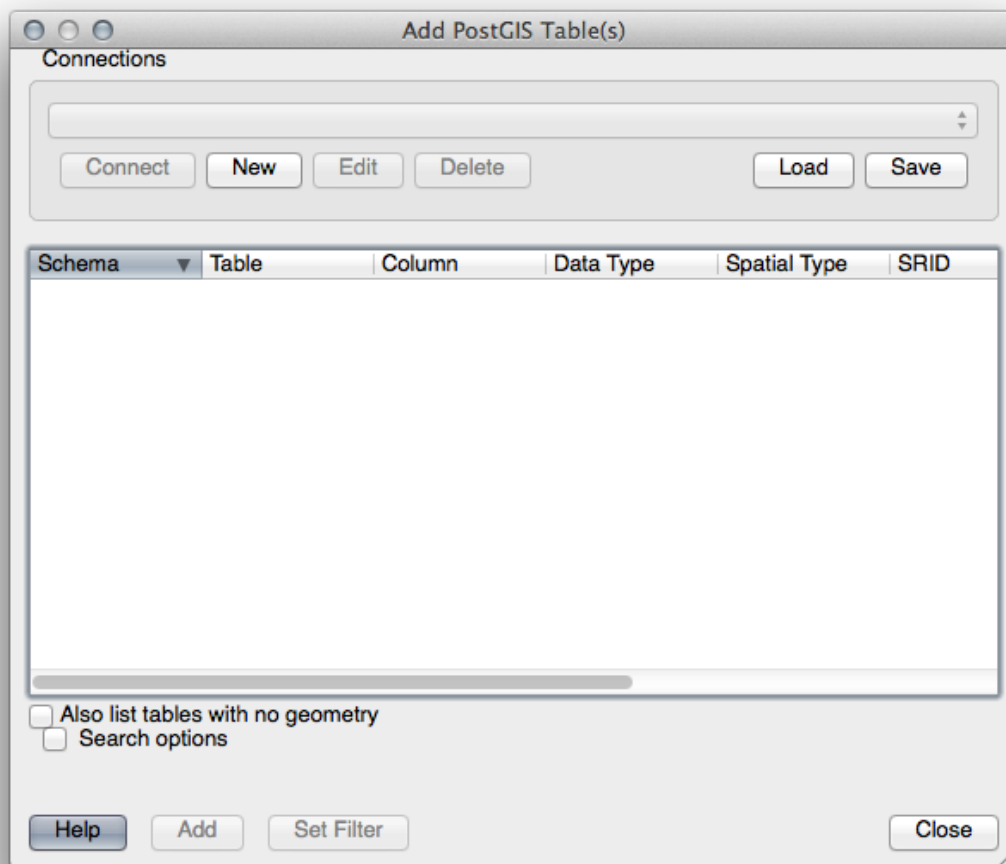
If at this point you were using a graphical interface, for example, specifying the projection for each point would be automatic. In other words, you usually won't need to worry about using the correct projection for every point you want to add if you've already specified it for that data-set, as we did 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.

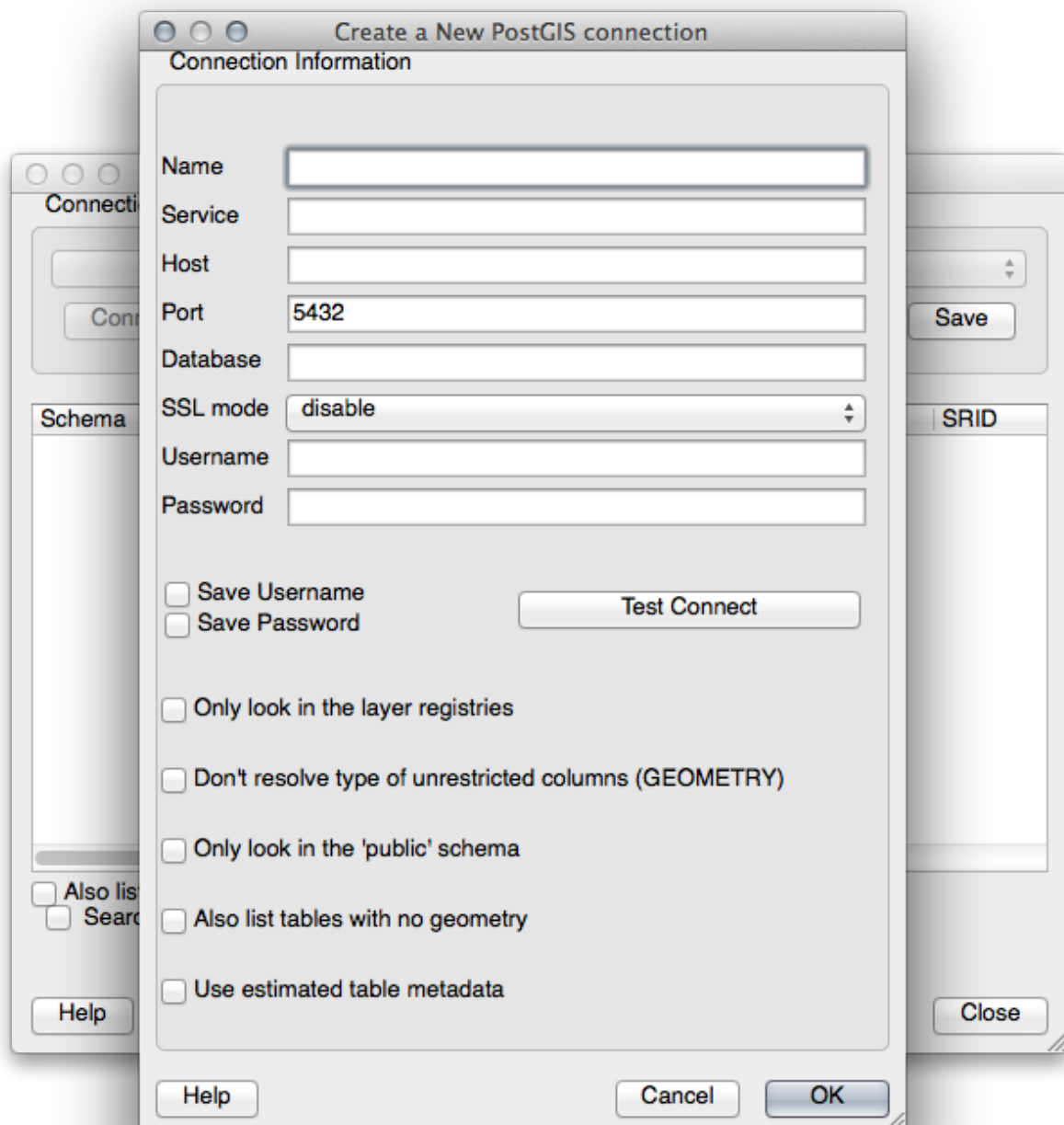
To load a PostGIS layer in QGIS, use the *Layer → Add PostGIS Layers* menu option or toolbar button:



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 `the_geom` column) as plain text.

Check your results

15.2.8 In Conclusion

You have seen how to add spatial objects to your database and view them in GIS software.

15.2.9 What's Next?

Next you'll see how to import data into, and export data from, your database.

15.3 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.

15.3.1 shp2pgsql

`shp2pgsql` is a commandline tool to import ESRI shapefiles 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.)

15.3.2 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>"
```

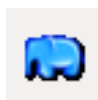
15.3.3 ogr2ogr

ogr2ogr is a very powerful tool to convert data into and from postgres to many data formats. ogr2ogr is part of the GDAL/OGR Software and has to be installed separately. To export a table from PostGIS to GML, you can use this command:

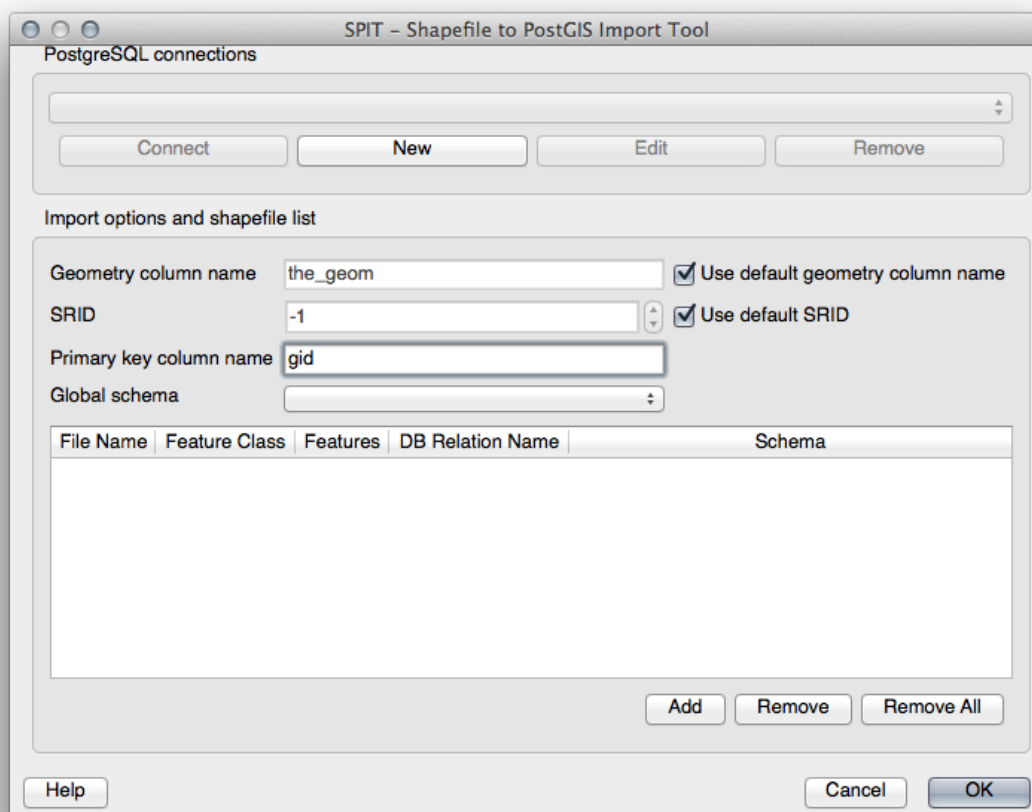
```
ogr2ogr -f GML export.gml PG:'dbname=<databasename> user=<username>
host=<hostname>' <Name of PostGIS-Table>
```

15.3.4 SPIT

SPIT is a QGIS plugin which is delivered with QGIS. You can use SPIT for uploading ESRI shapefiles to PostGIS. Once you've added the SPIT plugin via the *Plugin Manager*, look for this button:



Clicking on it or selecting *Database -> Spit -> Import Shapefiles to PostgreSQL* from the menu will give you the SPIT dialog:



You can add shapefiles to the database by clicking the *Add* button, which will give you a file browser window.

15.3.5 DB Manager

You may have noticed another option in the *Database* menu labeled *DB Manager*. This is a new tool in QGIS 2.0 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.

15.3.6 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.

15.3.7 What's Next?

Next we'll look at how to query the data we've created before.

15.4 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.

15.4.1 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(the_geom, 'SRID=4326;POINT(33 -34)') < 2;
```

Result:

| id | name | house_no | street_id | phone_no | the_geom |
|----|--------------|----------|-----------|---------------|---------------|
| 6 | Fault Towers | 34 | 3 | 072 812 31 28 | 01010008040C0 |

(1 row)

Bemerkung: the_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?

Check your results

15.4.2 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
(the_geom);
```

```
\d people
```

Result:

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 | |
| the_geom | geometry | |

Indexes:

```
"people_pkey" PRIMARY KEY, btree (id)
"people_geo_idx" gist (the_geom) <-- new spatial key added
"people_name_idx" btree (name)
```

Check constraints:

```
"people_geom_point_chk" CHECK (st_geometrytype(the_geom) = 'ST_Point'::text
OR the_geom IS NULL)
```

Foreign-key constraints:

```
"people_street_id_fkey" FOREIGN KEY (street_id) REFERENCES streets(id)
```

15.4.3 Try Yourself

Modify the cities table so its geometry column is spatially indexed.

Check your results

15.4.4 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 SPIT. 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.the_geom) as point
FROM building a, region b
WHERE st_within(a.the_geom, b.the_geom)
AND b.name = 'KwaZulu';
```

Result:

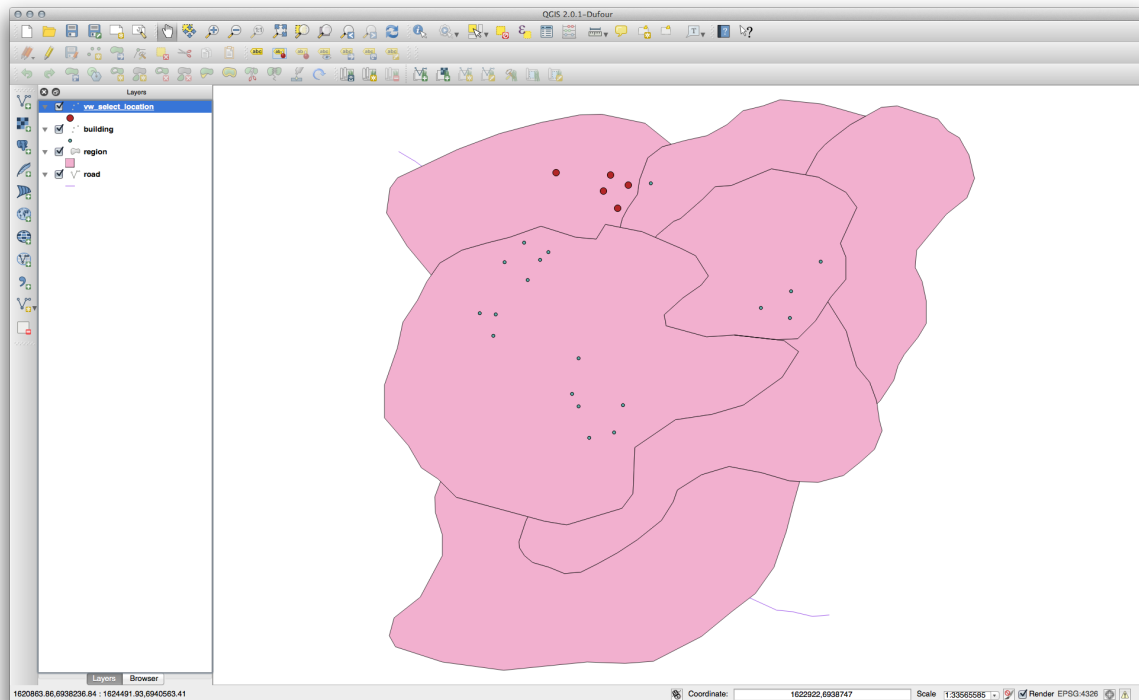
| id | name | point |
|----|------|--|
| 30 | York | POINT(1622345.23785063 6940490.65844485) |
| 33 | York | POINT(1622495.65620524 6940403.87862489) |
| 35 | York | POINT(1622403.09106394 6940212.96302097) |
| 36 | York | POINT(1622287.38463732 6940357.59605424) |
| 40 | York | POINT(1621888.19746548 6940508.01440885) |

(5 rows)

Or, if we create a view from it:

```
CREATE VIEW vw_select_location AS
SELECT a.gid, a.name, a.the_geom
FROM building a, region b
WHERE st_within(a.the_geom, b.the_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.the_geom, b.the_geom)
AND a.name = 'Hokkaido';
```

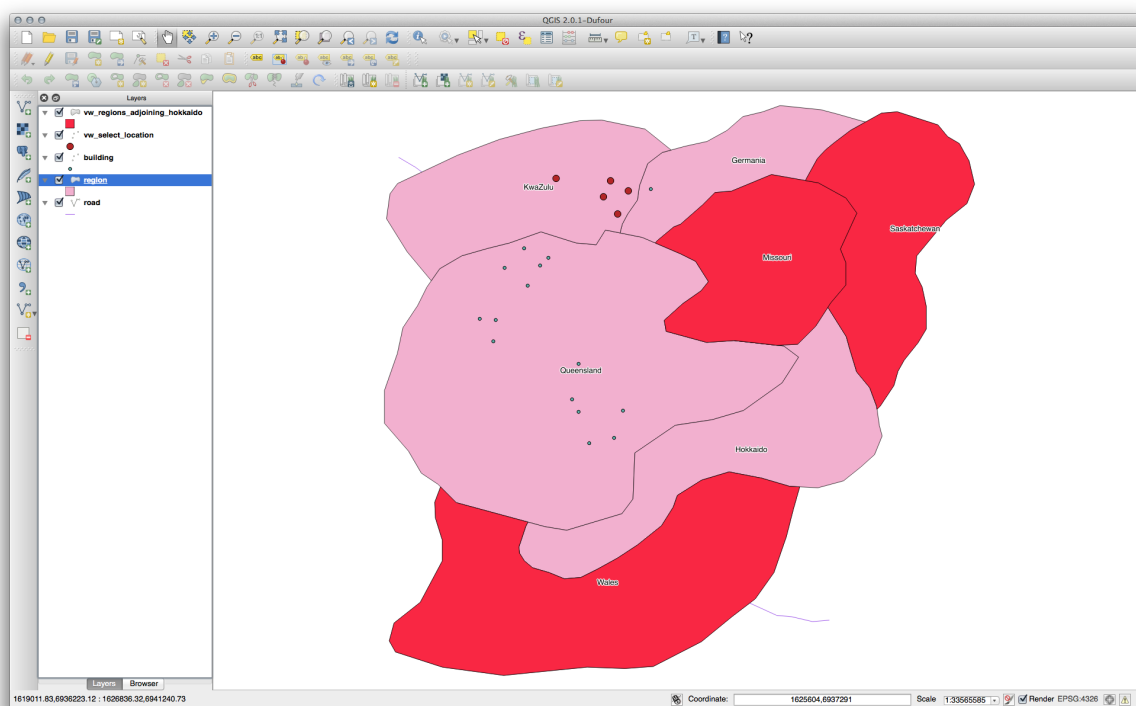
Result:

```
name
-----
Missouri
Saskatchewan
Wales
(3 rows)
```

As a view:

```
CREATE VIEW vw_regions_adjoining_hokkaido AS
SELECT b.gid, b.name, b.the_geom
FROM region a, region b
WHERE TOUCHES(a.the_geom, b.the_geom)
AND a.name = 'Hokkaido';
```

In QGIS:

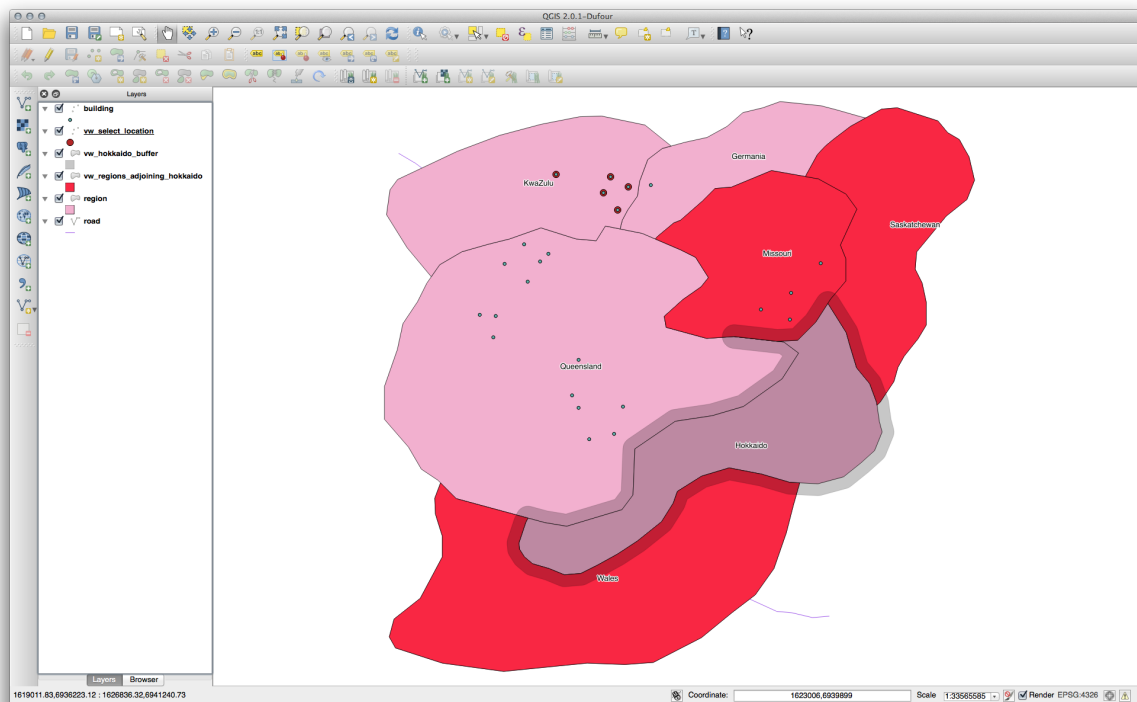


Note the missing region (Queensland). This may be due to a topology error. Artifacts such as this can alert us to potential problems in the data. To solve this enigma without getting caught up in the anomalies the data may have, we could use a buffer intersect instead:

```
CREATE VIEW vw_hokkaido_buffer AS
SELECT gid, ST_BUFFER(the_geom, 100) as the_geom
FROM region
WHERE name = 'Hokkaido';
```

This creates a buffer of 100 meters around the region Hokkaido.

The darker area is the buffer:

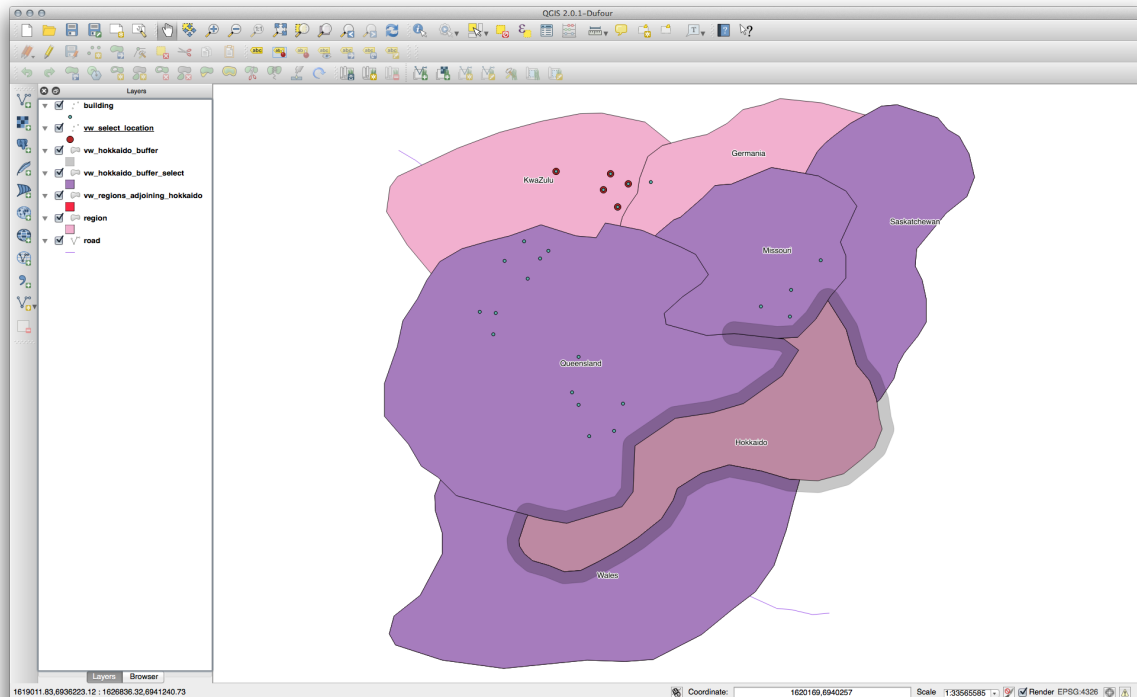


Select using the buffer:

```
CREATE VIEW vw_hokkaido_buffer_select AS
SELECT b.gid, b.name, b.the_geom
FROM
(
  SELECT * FROM
    vw_hokkaido_buffer
) a,
region b
WHERE ST_INTERSECTS(a.the_geom, b.the_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.the_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.

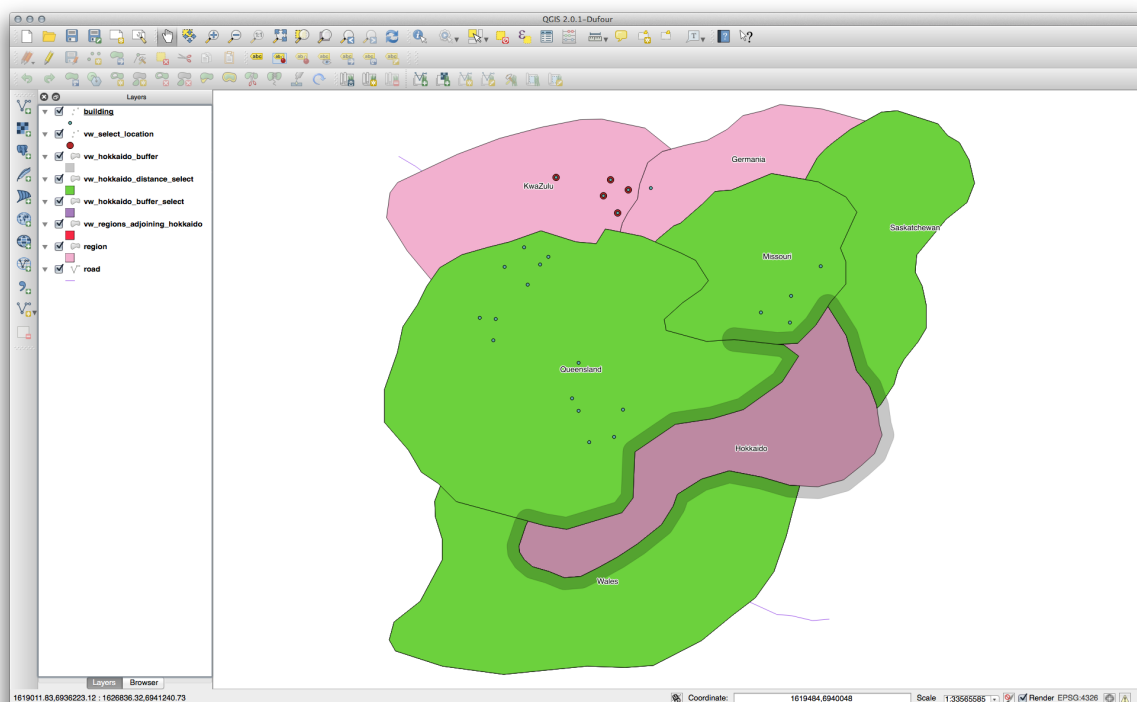
In QGIS:



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.the_geom
FROM region a, region b
WHERE ST_DISTANCE (a.the_geom, b.the_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.the_geom, b.the_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.the_geom, b.the_geom)) as
text, ST_SHORTESTLINE(a.the_geom, b.the_geom) AS the_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.the_geom, b.the_geom)) as
text, ST_LONGESTLINE(a.the_geom, b.the_geom) AS the_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.the_geom) as the_geom
FROM road a
WHERE a.id = 1;
```

```
CREATE VIEW vw_region_centroid AS
SELECT a.gid as gid, ST_CENTROID(a.the_geom) as the_geom
FROM region a
WHERE a.name = 'Saskatchewan';
```

```
SELECT ST_PERIMETER(a.the_geom)
FROM region a
WHERE a.name='Queensland';
```

```
SELECT ST_AREA(a.the_geom)
FROM region a
WHERE a.name='Queensland';
```

```
CREATE VIEW vw_simplify AS
SELECT gid, ST_Simplify(the_geom, 20) AS the_geom
FROM road;
```

```
CREATE VIEW vw_simplify_more AS
SELECT gid, ST_Simplify(the_geom, 50) AS the_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.the_geom)) AS the_geom
FROM building a
GROUP BY a.name;
```

15.4.5 In Conclusion

You have seen how to query spatial objects using the new database functions from PostGIS.

15.4.6 What's Next?

Next we're going to investigate the structures of more complex geometries and how to create them using PostGIS.

15.5 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.

15.5.1 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.

15.5.2 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

Check your results

Now let's insert a linestring into our `streets` table. In this case we will update an existing street record:

```
update streets set the_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.

15.5.3 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, the_geom)
values ('Tokyo', 'SRID=4326;POLYGON((10 -10, 5 -32, 30 -27, 10 -10))');
```

Bemerkung: 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, the_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.

15.5.4 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 | |
| the_geom | geometry | |
| city_id | integer | not null |

Indexes:

```
"people_pkey" PRIMARY KEY, btree (id)
"people_name_idx" btree (name)
```

Check constraints:

```
"people_geom_point_chk" CHECK (st_geometrytype(the_geom) =
                                'ST_Point'::text OR the_geom IS NULL)
```

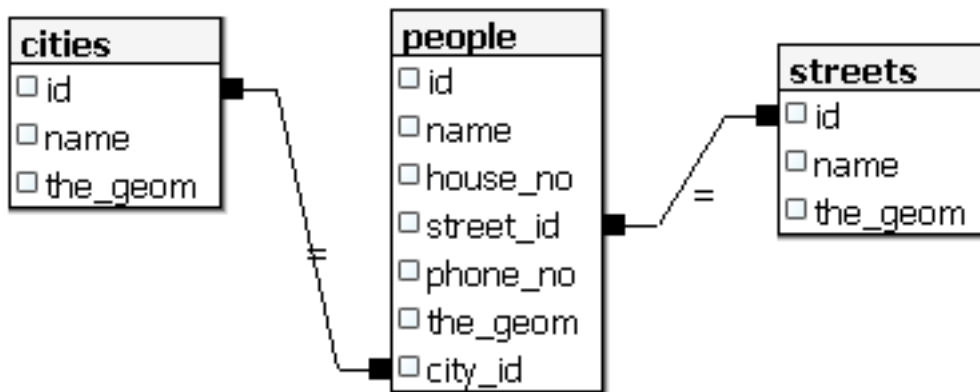
Foreign-key constraints:

```
"people_city_id_fkey" FOREIGN KEY (city_id) REFERENCES cities(id)
"people_street_id_fkey" FOREIGN KEY (street_id) REFERENCES streets(id)
```

Check your results

15.5.5 Looking at Our Schema

By now our schema should be looking like this:



15.5.6 Try Yourself

Create city boundaries by computing the minimum convex hull of all addresses for that city and computing a buffer around that area.

15.5.7 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(geometry) from myPolygonTable;
```

- Select the first vertex point of the resultant linestring:

```
select st_startpoint(myGeometry)
from (
  select st_boundary(geometry) as myGeometry
  from myPolygonTable) as foo;
```

15.5.8 Data Processing

PostGIS supports all OGC SFS/MM standard conform functions. All these functions start with `ST_`.

15.5.9 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.the_geom, b.the_geom))
```




```
select st_intersection(a.the_geom, b.the_geom), b.*
from clip as a, road_lines as b
where not st_isempty(st_intersection(st_setsrid(a.the_geom, 32734),
    b.the_geom));
```



15.5.10 Building Geometries from Other Geometries

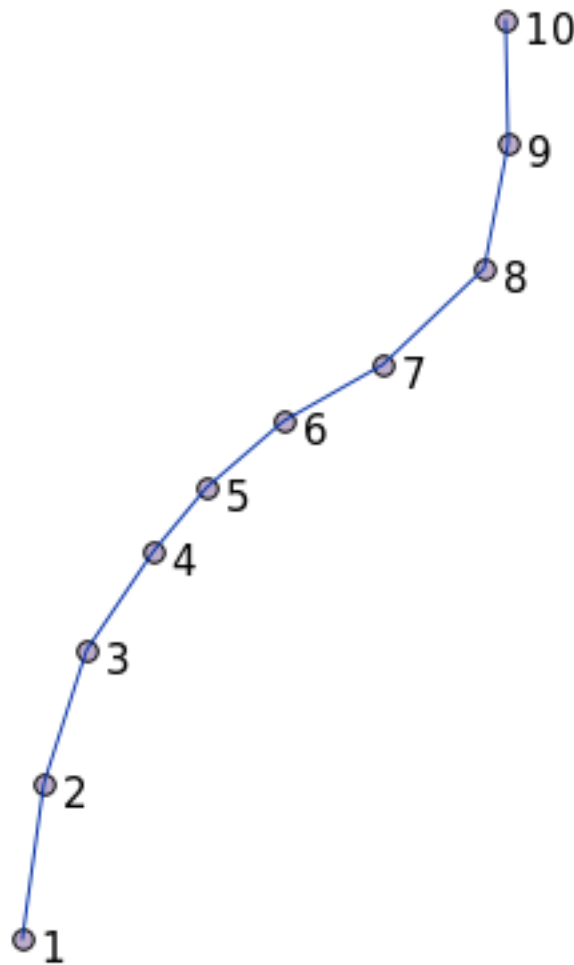
From a given point table, you want to generate a linestring. The order of the points is defined by their `id`. Another ordering method could be a timestamp, such as the one you get when you capture waypoints with a GPS receiver.



To create a linestring from a new point layer called ‘points’, you can run the following command:

```
select ST_LineFromMultiPoint(st_collect(the_geom)), 1 as id
from (
  select the_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.



15.5.11 Geometry Cleaning

You can get more information for this topic in [this blog entry](#).

15.5.12 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`.

15.5.13 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
```

15.5.14 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.

Module: Using Spatial Databases in QGIS

In this module you will learn about how to use Spatial Databases with QGIS to manage, display and manipulate data in the database as well as performing analysis by querying. We will primarily use PostgreSQL and PostGIS (which were covered in previous sections), but the same concepts are applicable to other spatial database implementations including spatialite.

16.1 Lesson: Working with Databases in the QGIS Browser

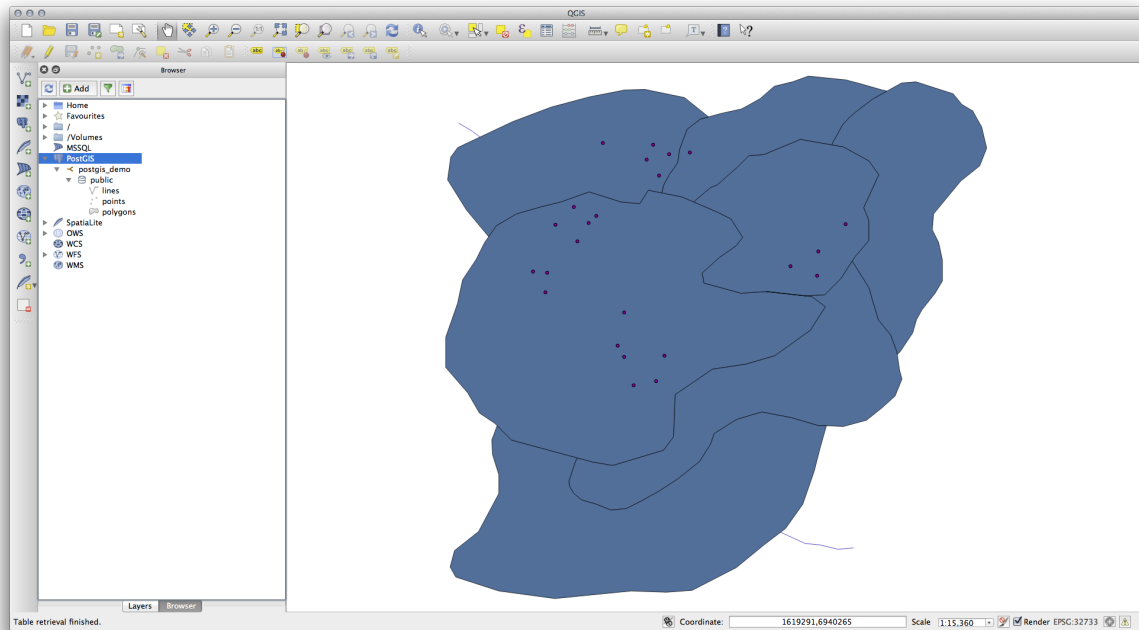
In the previous 2 modules we looked at the basic concepts, features and functions of relational databases as well as extensions that let us store, manage, query and manipulate spatial data in a relational database. This section will dive deeper into how to effectively use spatial databases in QGIS.

The goal for this lesson: To learn how to interact with spatial databases using the QGIS Browser interface.

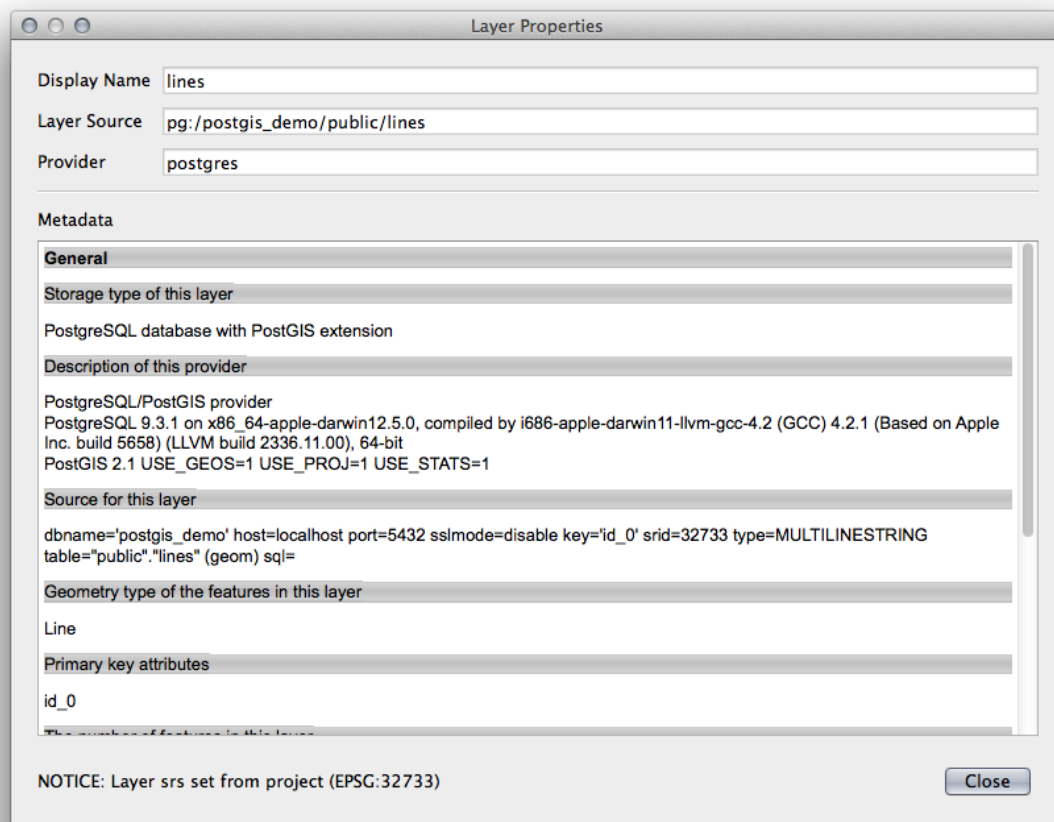
16.1.1 Follow Along: Adding Database Tables to QGIS using the Browser

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.



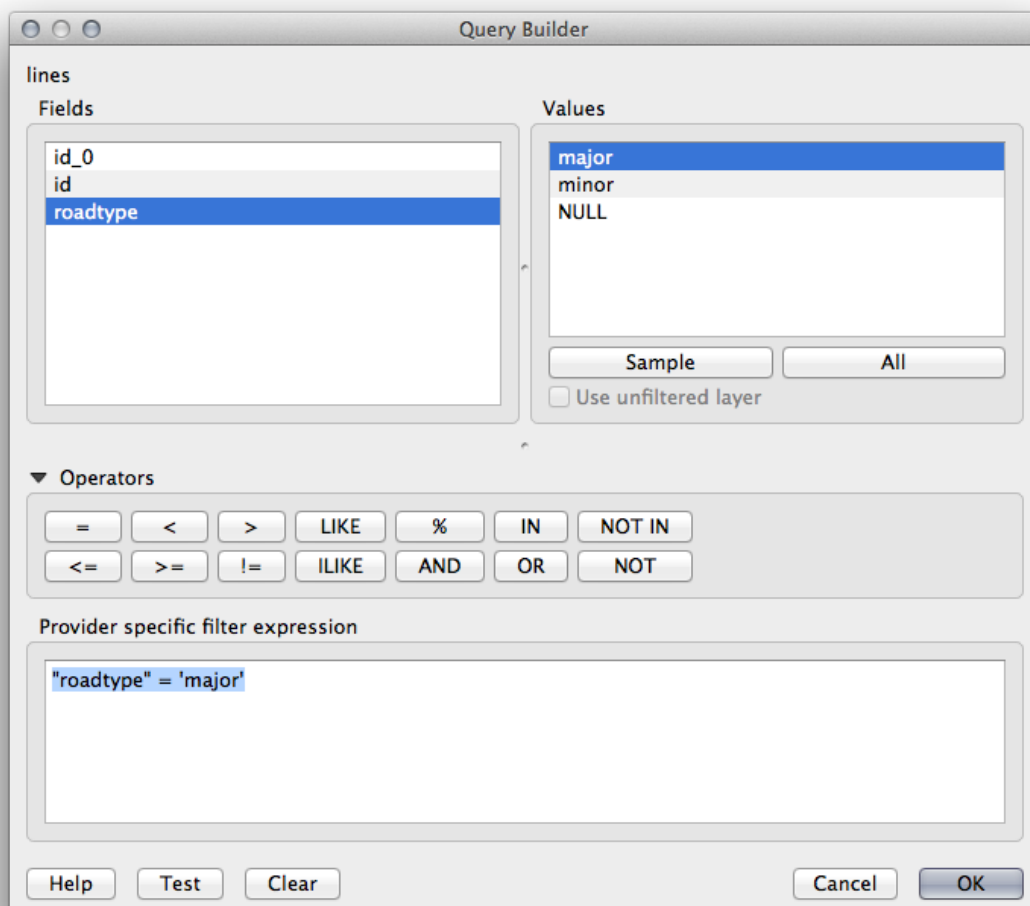
Bemerkung: Of course you can also use this interface to connect to PostGIS databases hosted on a server external to your workstation. Right clicking on the PostGIS entry in the tree will allow you to specify connection parameters for a new connection.

16.1.2 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.

16.1.3 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.

16.1.4 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.

16.2 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.

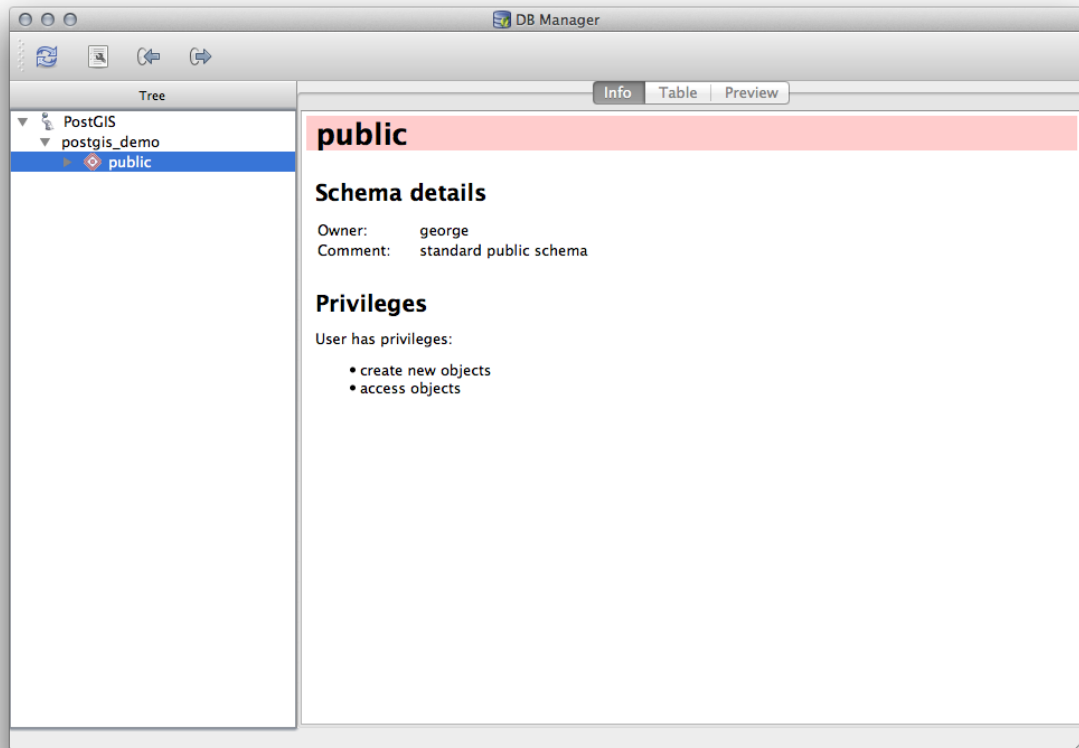
16.2.1 Follow Along: Managing PostGIS Databases with DB Manager

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.

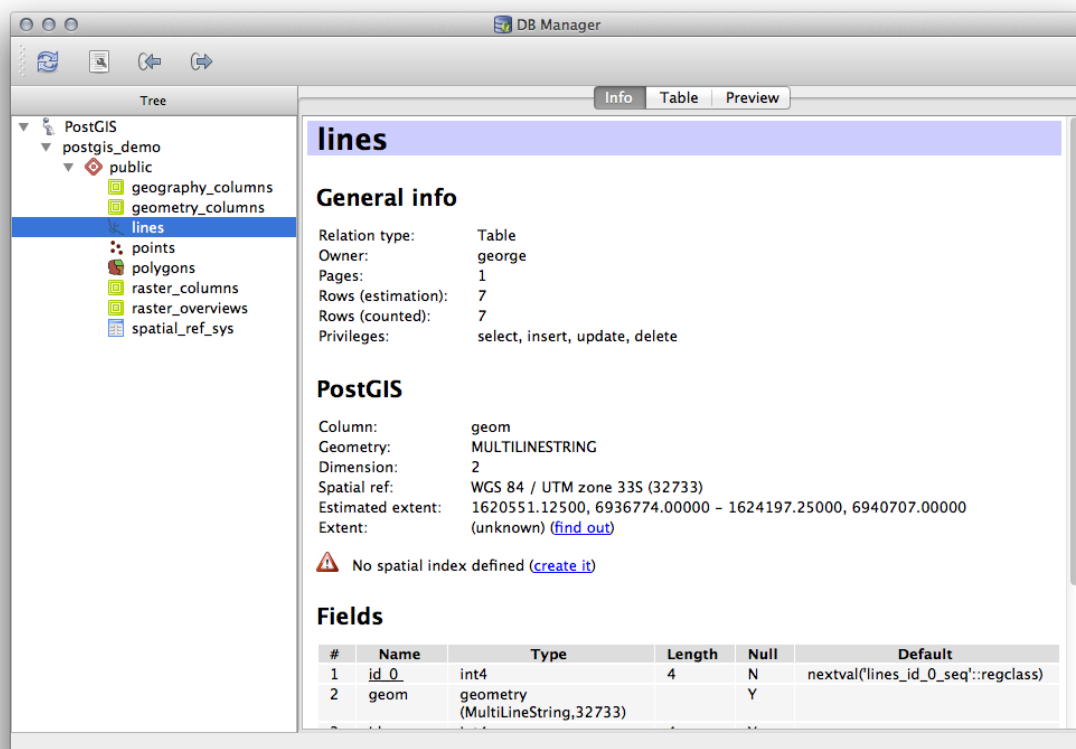
The first thing you may notice is that you can now see some metadata about the Schemas contained in your database.



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](#). 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.

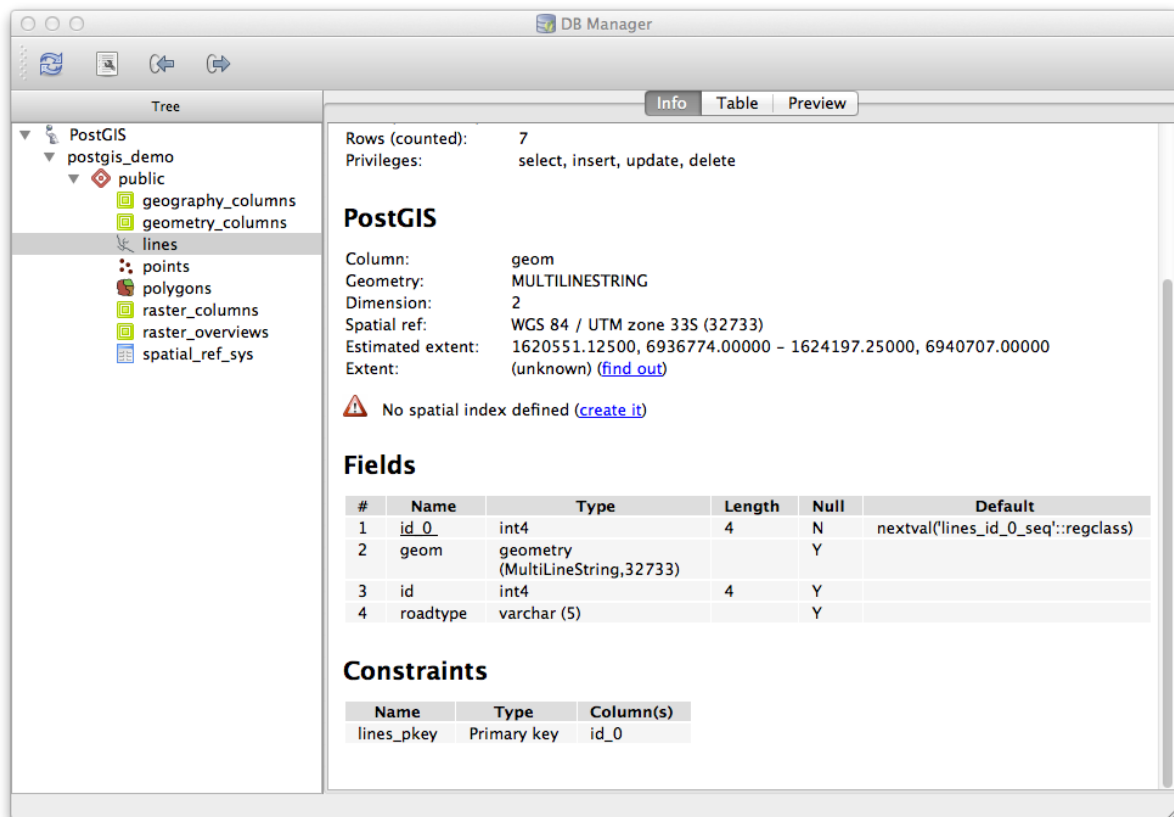
DB Manager can also be used to manage the tables within your database. We have already looked at various ways to create and manage tables on the command line, but now let's look at how to do this in DB Manager.

First, it's useful to just look at a table's metadata by clicking on its name in tree and looking in the *Info* tab.

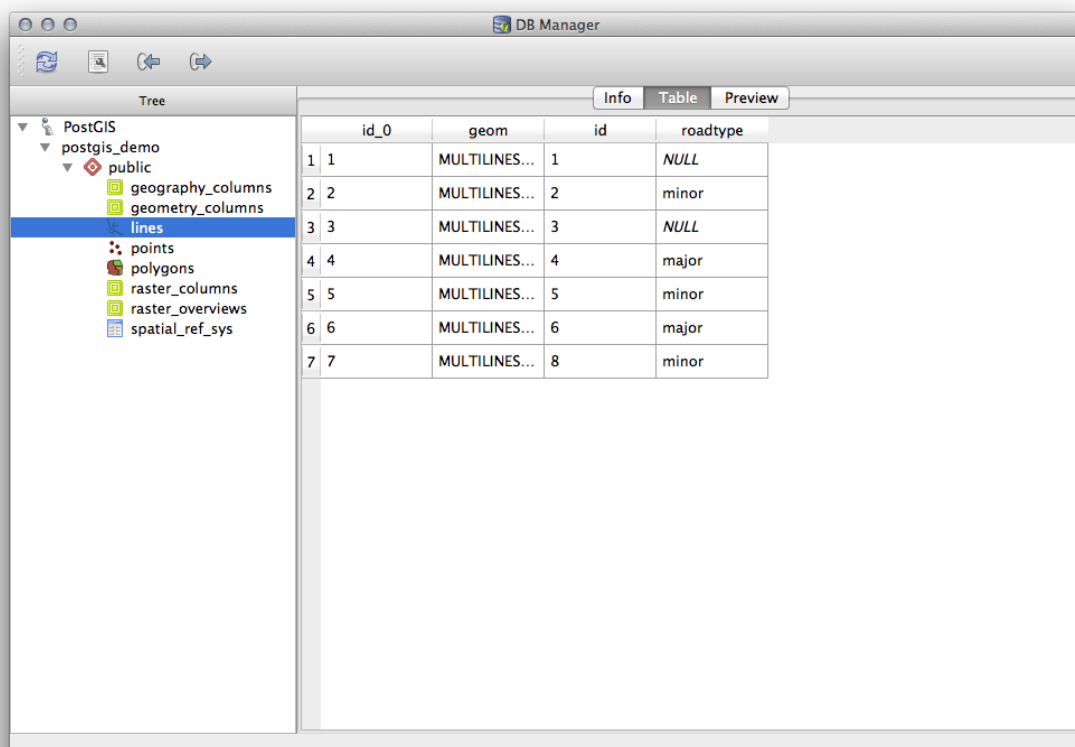


In this panel you can see the *General Info* about the table as well the information that the PostGIS extension maintains about the geometry and spatial reference system.

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.



It's 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.

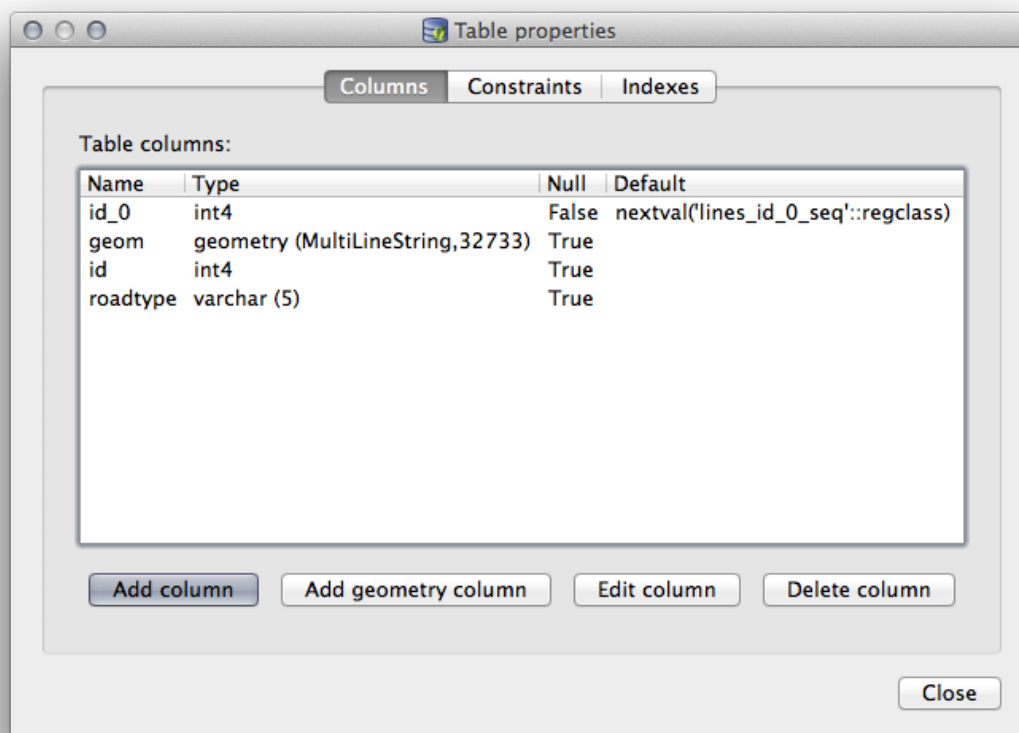


There is also a *Preview* tab which will show you the layer data in a map preview.

Right Clicking on a layer in the tree and clicking *Add to Canvas* will add this layer to your map.

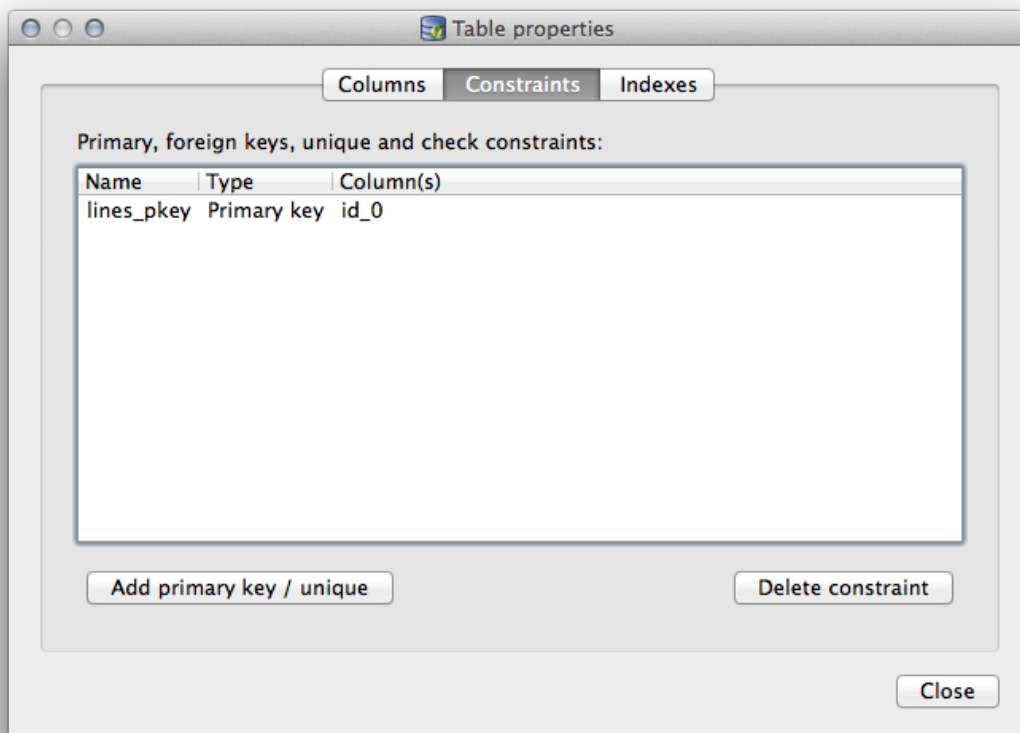
So far we have only been viewing the database its schemas and tables and their metadata, but what if we wanted to alter the table to add an additional column perhaps? DB Manager allows you to do this directly.

- Select the table you want to edit in the tree
- Select *Table* → *Edit Table* from the menu to open the *Table Properties* dialog.

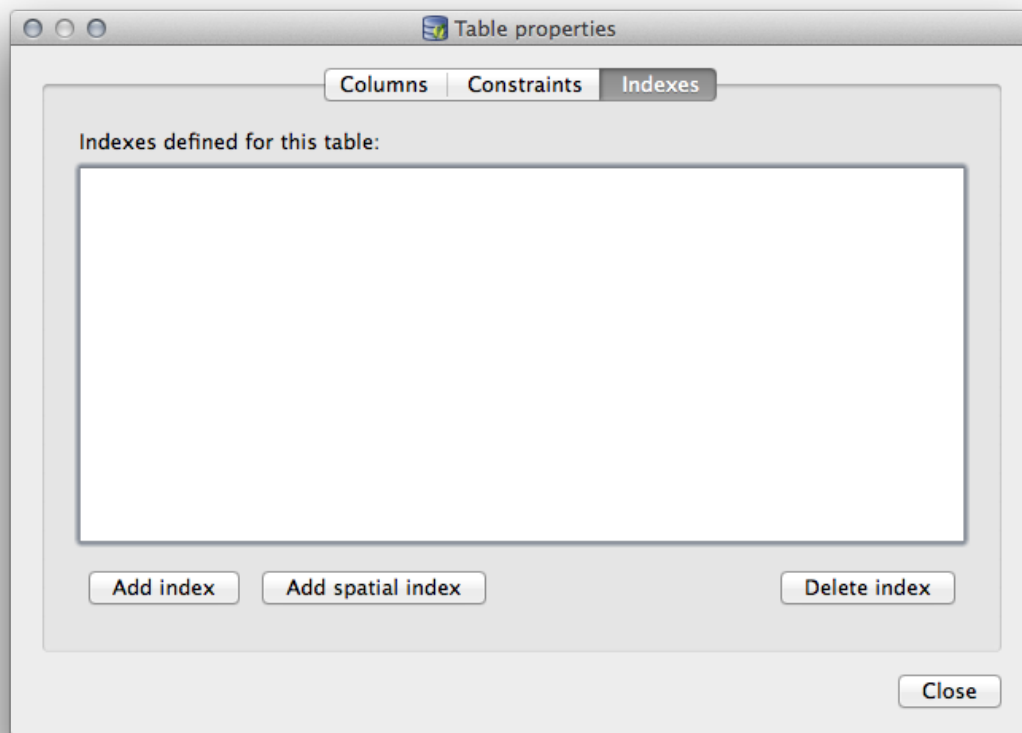


You can use this dialog to Add Columns, Add geometry columns, edit existing columns or to remove a column completely.

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.



16.2.2 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.

- 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.
- From the menu select *Table* → *Create Table* to bring up the Create Table dialog.
- Use the default `Public` schema and name the table `places`.
- Add the `id`, `place_name`, and `elevation` fields as shown below
- Make sure the `id` field is set as the primary key.
- 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*.
- Click the checkbox to *Create spatial index* and click *Create* to create the table.

Schema: public

Name: places

| | Name | Type | Null |
|---|------------|---------|--------------------------|
| 1 | id | serial | <input type="checkbox"/> |
| 2 | place_name | text | <input type="checkbox"/> |
| 3 | elevation | integer | <input type="checkbox"/> |

Primary key: id

☒ Create geometry column: POINT

Name: geom

Dimensions: 2

SRID: 4326

☒ Create spatial index

Buttons: Add field, Delete field, Up, Down, Create, Close

- 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.

16.2.3 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 optionally analyze your tables for better performance.

Lets take a look at how we can perform a *VACUUM ANALYZE* command from within DB Manager.

- Select one of your tables in the DB Manager Tree.
- Select *Table -> Run Vacuum Analyze* from the menu.

Thats it! PostgreSQL will 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](#)

16.2.4 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 let's look at it again here with DB Manager.

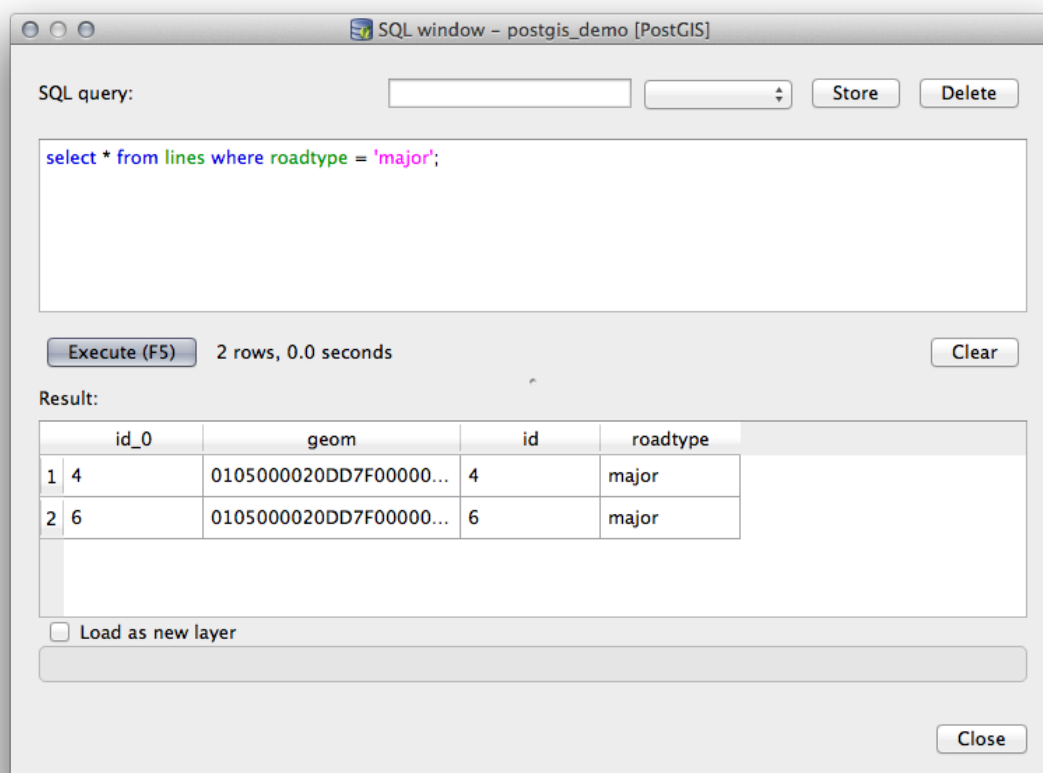
- Select the `lines` table in the tree.
- Select the *SQL window* button in the DB Manager toolbar.



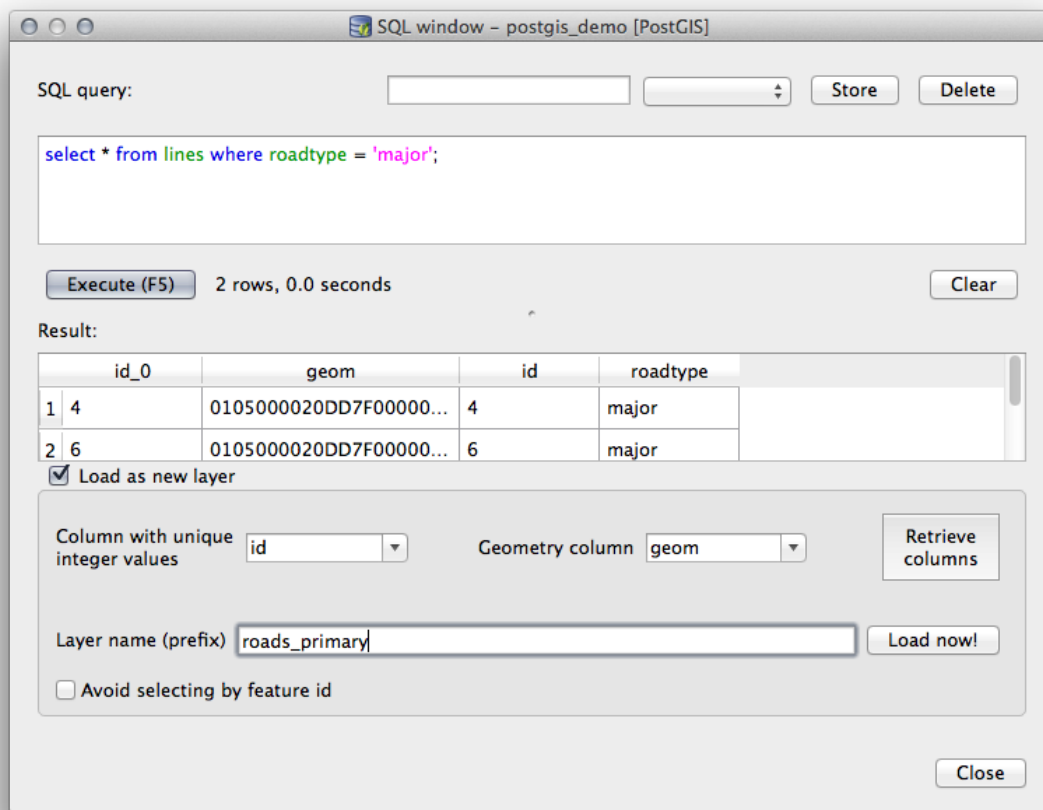
- Compose the following *SQL query* in the space provided:

```
select * from lines where roadtype = 'major';
```

- Click the *Execute (F5)* button to run the query.
- You should now see the records that match in the *Result* panel.



- Click the checkbox for *Load as new layer* to add the results to your map.
- Select the `id` column as the *Column with unique integer values* and the `geom` column as the *Geometry column*.
- Enter `roads_primary` as the *Layer name (prefix)*.
- 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.

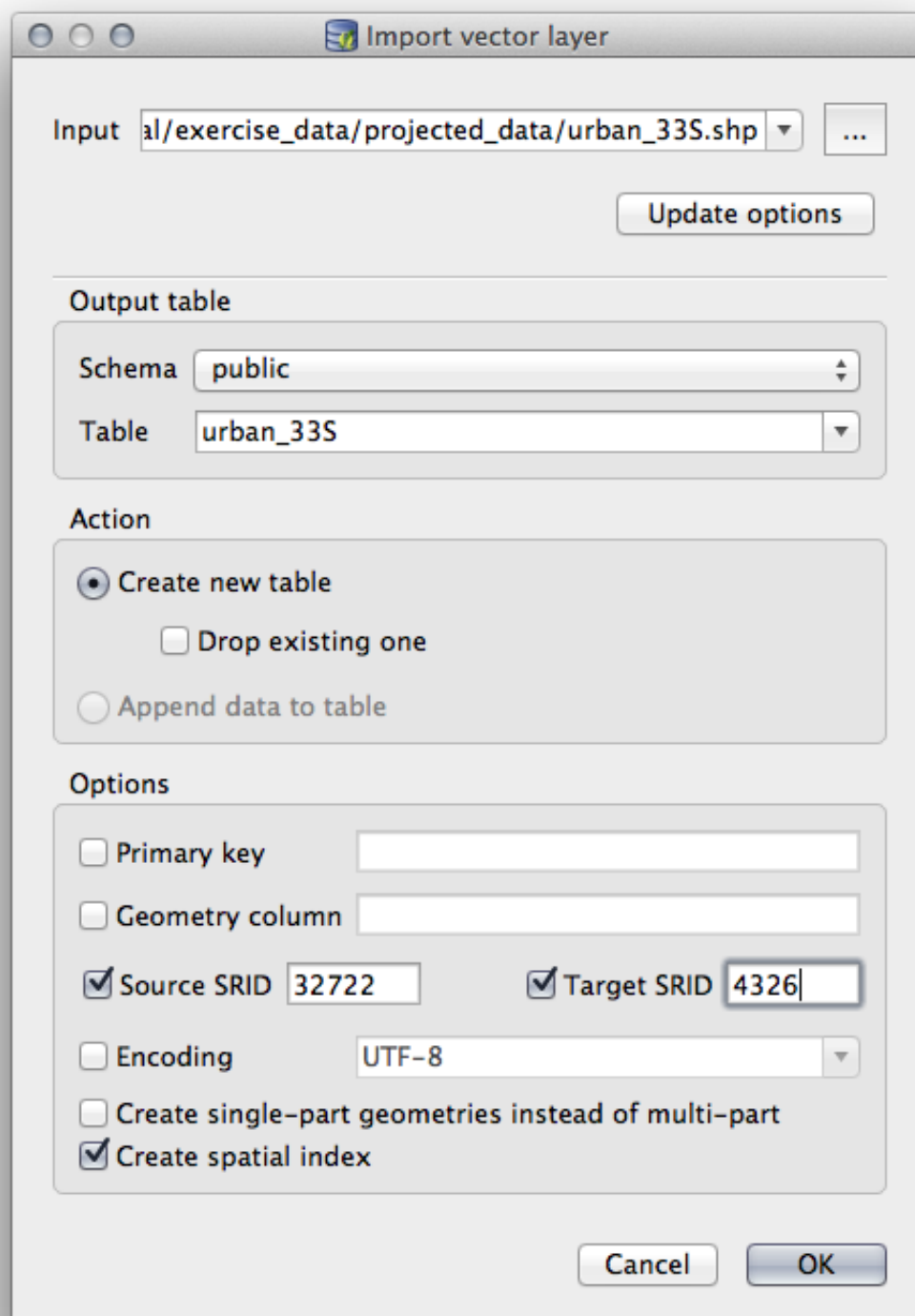
16.2.5 Importing Data into a Database with DB Manager

We have already looked at how to import data into a spatial database using command line tools and also looked at how to use the SPIT plugin, so now lets learn how to use DB Manager to do imports.

- Click the *Import layer/file* button on the toolbar in the DB Manager dialog.

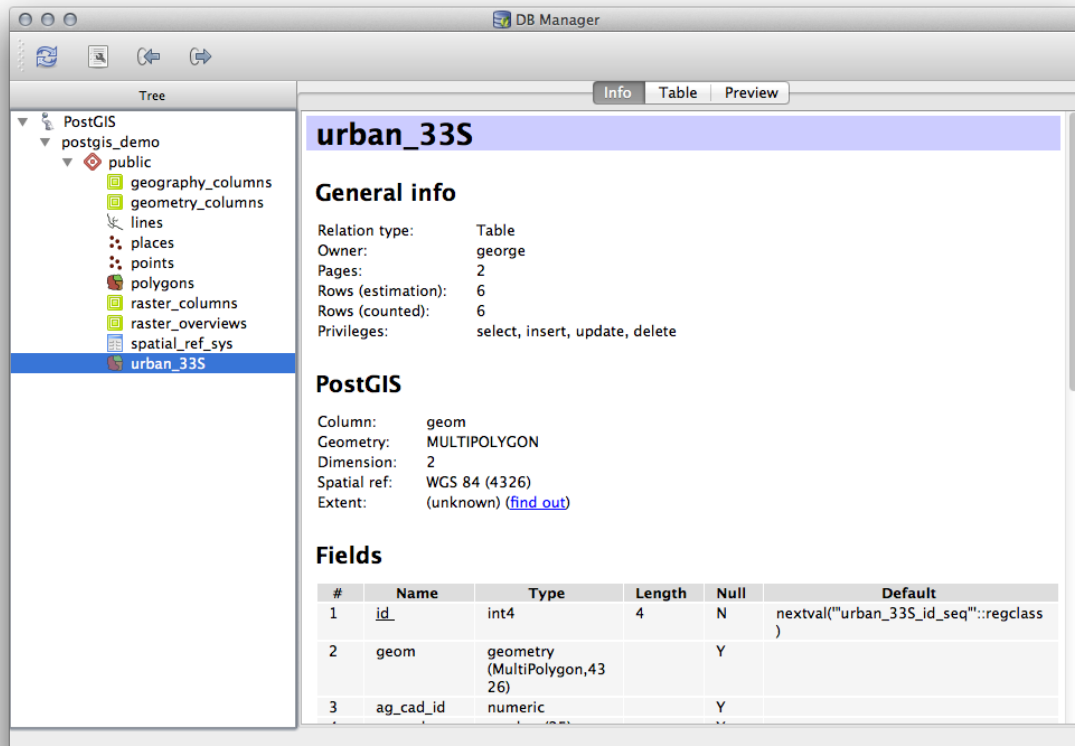


- Select the `urban_33S.shp` file from `exercise_data/projected_data` as the input dataset.
- Click the *Update Options* button to pre-fill some of the form values.
- Make sure that the *Create new table* option is selected
- Specify the *Source SRID* as 32722 and the *Target SRID* as 4326.
- Enable the checkbox to *Create Spatial Index*
- Click *OK* to perform the import.



- Dismiss the dialog letting you know that the import was successful
- 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)

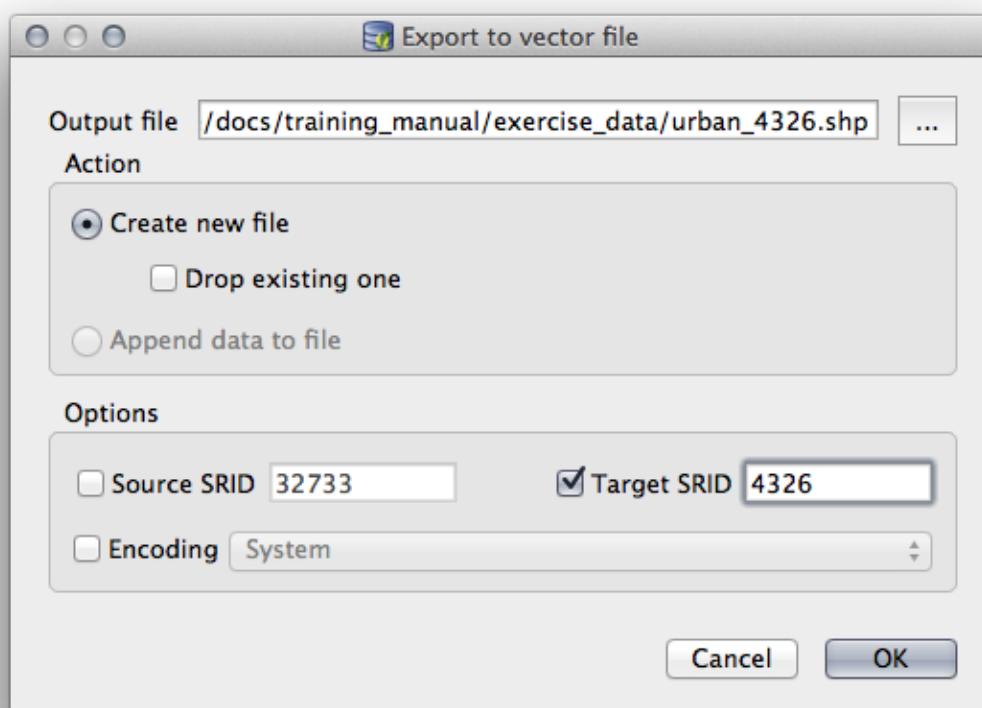


Right clicking on the table in the Tree and selecting *Add to Canvas* will add the table as a layer in your map.

16.2.6 Exporting Data from a Database with DB Manager

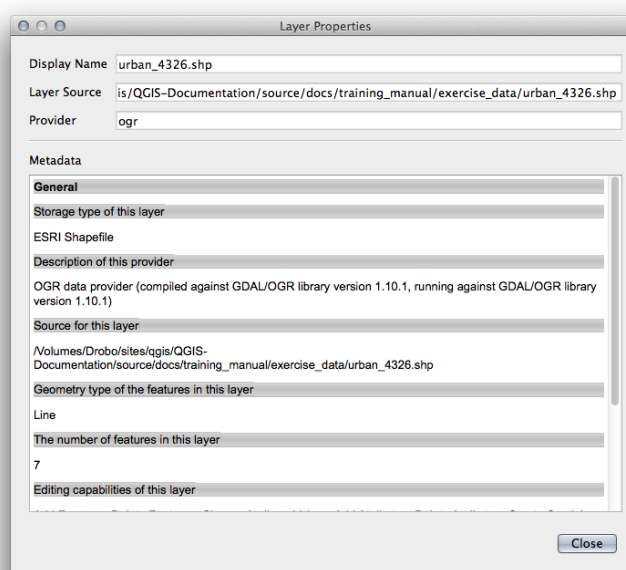
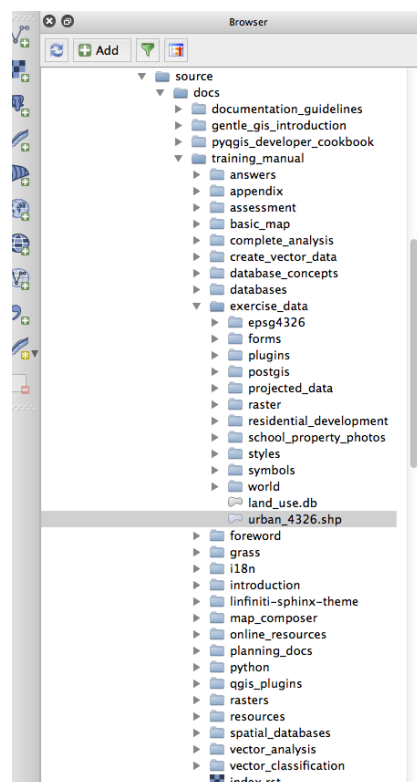
Of course DB Manager can also be used to export data from your spatial databases, so let's take a look at how that is done.

- 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.
- Click the `...` button to select the *Output file* and save the data to your `exercise_data` directory as `urban_4326`.
- Set the *Target SRID* as 4326.
- Click *OK* to initialize the export.



- 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.



16.2.7 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.

16.2.8 What's Next?

Next, we will look at how to use many of these same techniques with *spatialite* databases.

16.3 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.

16.3.1 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.

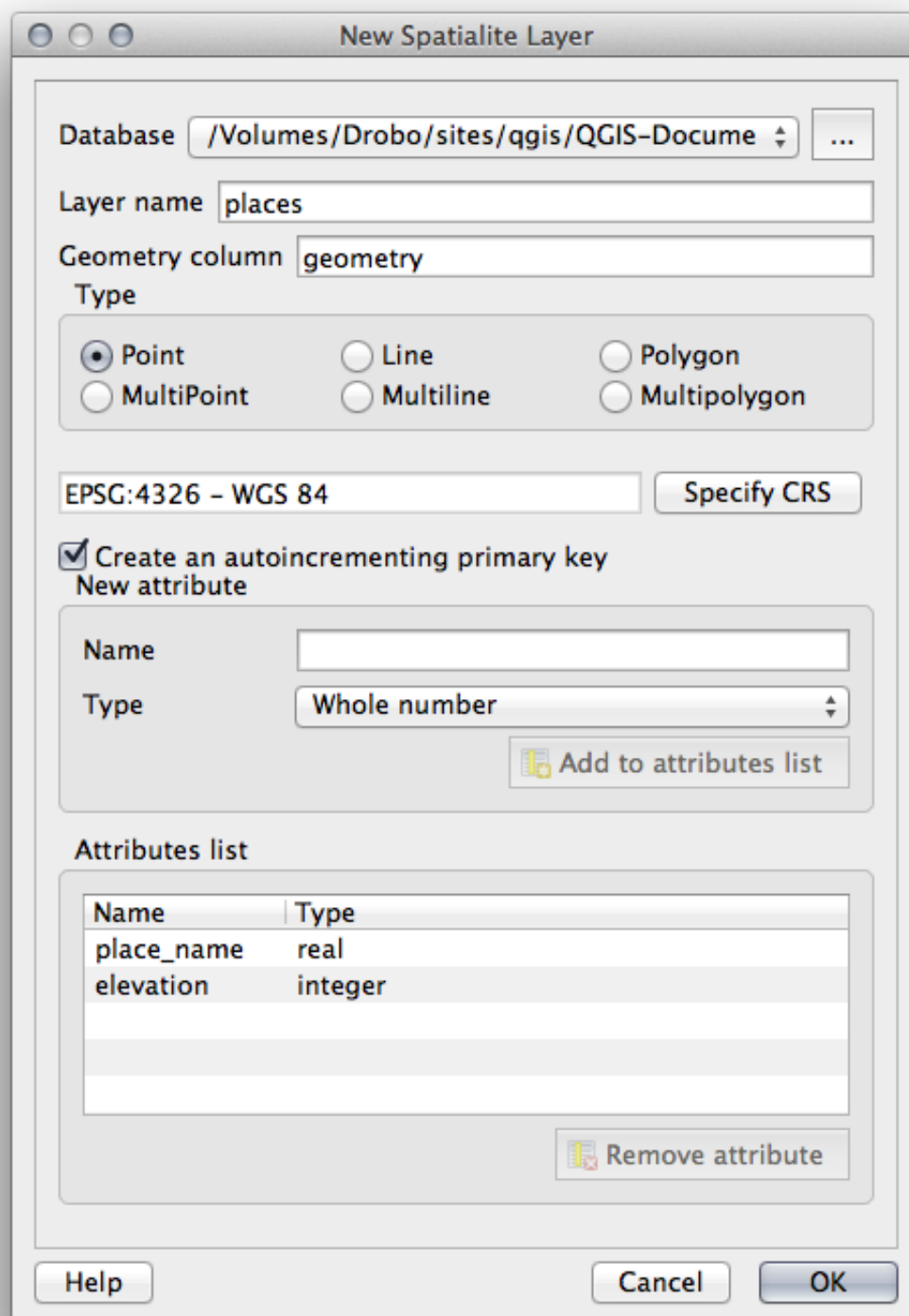
- Right click on the *Spatialite* entry in the Browser tree and select *Create Database*.
- Specify where on your filesystem you want to store the file and name it `qgis-sl.db`.
- Again right click on the *Spatialite* entry in the Browser tree and now select the *New Connection* item. Find the file you 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.

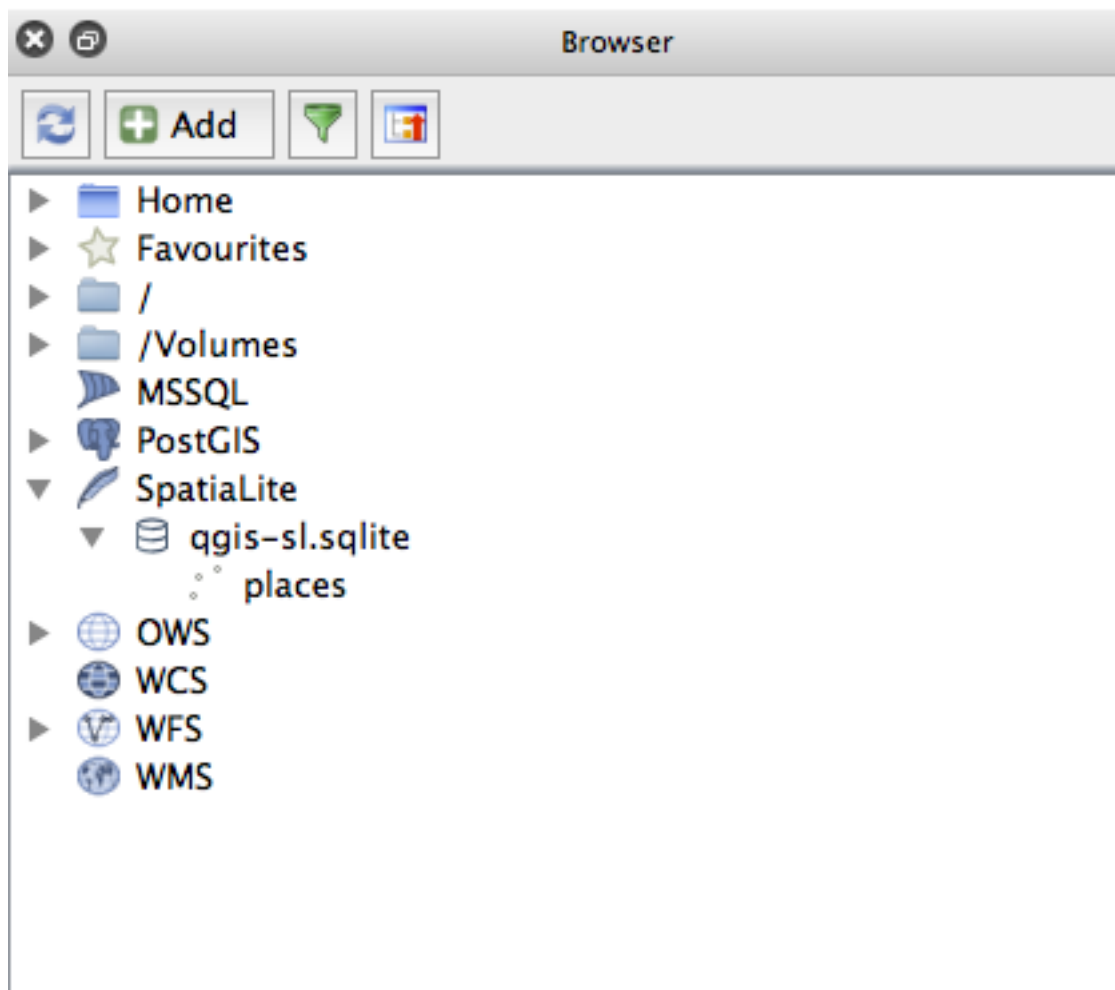
- Find the button to create a new layer and use the dropdown to create a new new Spatialite layer, or select *Layer -> New -> New Spatialite Layer*.



- Select the database we created in the previous steps in the drop down.
- Give the layer the name `places`.
- Tick the checkbox next to *Create an auto-incrementing primary key*.
- Add 2 attributes as shown in below
- Click *OK* to create the table.



- 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.

16.3.2 In Conclusion

You have seen how to create spatialite databases and add tables to them and to use these tables as layers in QGIS.

Appendix: Contributing To This Manual

To add materials to this course, you must follow the guidelines in this Appendix. You are not 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.

17.1 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.

17.2 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.

17.3 Adding a Module

- To add a new module, first create a new directory (directly under the top-level of the `qgis-training-manual` directory) with the name of the new module.
- Under this new directory, create a file called `index.rst`. Leave this file blank for now.
- Open the `index.rst` file under the top-level directory. Its first lines are:

```
.. toctree::
   :maxdepth: 2
   :numbered:

   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.

- 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.
- Remember to keep the order of the modules logical, such that later modules build on the knowledge presented in earlier modules.
- Open your new module's own index file (`[module name]/index.rst`).
- Along the top of the page, write a line of 80 asterisks (*). This represents a module heading.

- Follow this with a line containing the markup phrase `|MOD|` (which stands for “module”), followed by the name of your module.
- End this off with another line of 80 asterisks.
- Leave a line open beneath this.
- Write a short paragraph explaining the purpose and content of the module.
- Leave one line open, then add the following text:

```
.. toctree::
    :maxdepth: 2
    :numbered:

    lesson1
    lesson2
```

... where `lesson1`, `lesson2`, etc., are the names of your planned lessons.

The module-level index file will look like this:

```
*****
|MOD| Module Name
*****
```

Short paragraph describing the module.

```
.. toctree::
    :maxdepth: 2
    :numbered:

    lesson1
    lesson2
```

17.4 Adding a Lesson

To add a lesson to a new or existing module:

- Open the module directory.
- Open the `index.rst` file (created above in the case of new modules).
- Ensure that the name of the planned lesson is listed underneath the `toctree` directive, as shown above.
- Create a new file under the module directory.
- Name this file exactly the same as the name you provided in the module’s `index.rst` file, and add the extension `.rst`.

Bemerkung: For editing purposes, a `.rst` file works exactly like a normal text file (`.txt`).

- To begin writing the lesson, write the markup phrase `|LS|`, followed by the lesson name.
- In the next line, write a line of 80 equal signs (=).
- Leave a line open after this.
- Write a short description of the lesson’s intended purpose.
- Include a general introduction to the subject matter. See the existing lessons in this manual for examples.
- Beneath this, start a new paragraph, beginning with this phrase:

```
**The goal for this lesson:**
```

- Briefly explain the intended outcome of completing this lesson.

- 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.

17.5 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 sheet at the end of the documentation, 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 `|basic|`, moderate by `|moderate|`, and advanced by `|hard|`.

17.5.1 Adding a “follow along” section

- To start this section, write the markup phrase of the intended difficulty level (as shown above).
- Leave a space and then write `|FA|` (for “follow along”).
- Leave another space and write the name of the section (use only an initial capital letter, as well as capitals for proper nouns).
- In the next line, write a line of 80 minuses/dashes (`-`). Ensure that your text editor does not replace the default minus/dash character with a long dash or other character.
- Write a short introduction to the section, explaining its purpose. Then give detailed (click-by-click) instructions on the procedure to be demonstrated.
- In each section, include internal links, external links and screenshots as needed.
- Try to end each section with a short paragraph that concludes it and leads naturally to the next section, if possible.

17.5.2 Adding a “try yourself” section

- To start this section, write the markup phrase of the intended difficulty level (as shown above).
- Leave a space and then write `|TY|` (for “try yourself”).
- In the next line, write a line of 80 minuses/dashes (`-`). Ensure that your text editor does not replace the default minus/dash character with a long dash or other character.
- Explain the exercise that you want the reader to complete. Refer to previous sections, lessons or modules if necessary.
- 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 add an entry in the answer sheet.

- First, decide on a unique name for the answer. Ideally, this name will include the name of the lesson and an incrementing number.
- Create a link for this answer:

```
:ref: 'Check your results <answer-name>'
```

- Open the answer sheet (`answers/answers.rst`).
- Create a link to the “try yourself” section by writing this line:

```
.. _answer-name:
```

- Write the instructions on how to complete the assignment, using links and images where needed.
- To end it off, include a link back to the “try yourself” section by writing this line:

```
:ref: 'Back to text <backlink-answer-name>'
```

- To make this link work, add the following line above the heading to the “try yourself” section:

```
.. _backlink-answer-name:
```

Remember that each of these lines shown above must have a blank line above and below it, otherwise it could cause errors while creating the document.

17.6 Add a Conclusion

- To end a lesson, write the phrase `|IC|` for “in conclusion”, followed by a new line of 80 minuses/dashes (`-`). Write a conclusion for the lesson, explaining which concepts have been covered in the lesson.

17.7 Add a Further Reading Section

- This section is optional.
- Write the phrase `FR` for “further reading”, followed by a new line of 80 minuses/dashes (`-`).
- Include links to appropriate external websites.

17.8 Add a What’s Next Section

- Write the phrase `|WN|` for “what’s next”, followed by a new line of 80 minuses/dashes (`-`).
- Explain how this lesson has prepared students for the next lesson or module.
- 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.

17.9 Using Markup

To adhere to the standards of this document, you will need to add standard markup to your text.

17.9.1 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*.
```

17.9.2 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.****

17.9.3 Images

- When adding an image, save it to the folder `_static/lesson_name/`.
- Include it in the document like this:

```
.. image:: /static/training_manual/lesson_name/image_file.extension
   :align: center
```

- Remember to leave a line open above and below the image markup.

17.9.4 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:
```

- To create a link, add this line:

```
:ref: `Descriptive link text <link-name>`
```

- Remember to leave a line open above and below this line.

17.9.5 External links

- To create an external link, write it out like this:

```
`Descriptive link text <link-url>`_
```

- Remember to leave a line open above and below this line.

17.9.6 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: `:kbd: `path/to/file``.

17.9.7 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 `:guiabel: `Tool Name`` button.

- This also applies if you are mentioning the name of a tool without requiring the user to click a button.

17.9.8 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'.
```

17.9.9 Adding notes

- You might need to 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.]
```

17.9.10 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.

17.10 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.

Answer Sheet

18.1 Results For *Adding Your First Layer*

18.1.1 *Preparation*

You should see a lot of lines, symbolizing roads. All these lines are in the vector layer that you just loaded to create a basic map.

[Back to text](#)

18.2 Results For *An Overview of the Interface*

18.2.1 *Overview (Part 1)*

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

[Back to text](#)

18.2.2 *Overview (Part 2)*

1. *Save as*
2. *Zoom to layer*
3. *Help*
4. *Rendering on/off*
5. *Measure line*

[Back to text](#)

18.3 Results For *Working with Vector Data*

18.3.1 *Shapefiles*

There should be five layers on your map:

- *places*
- *water*
- *buildings*
- *rivers and*
- *roads.*

[Back to text](#)

18.3.2 Databases

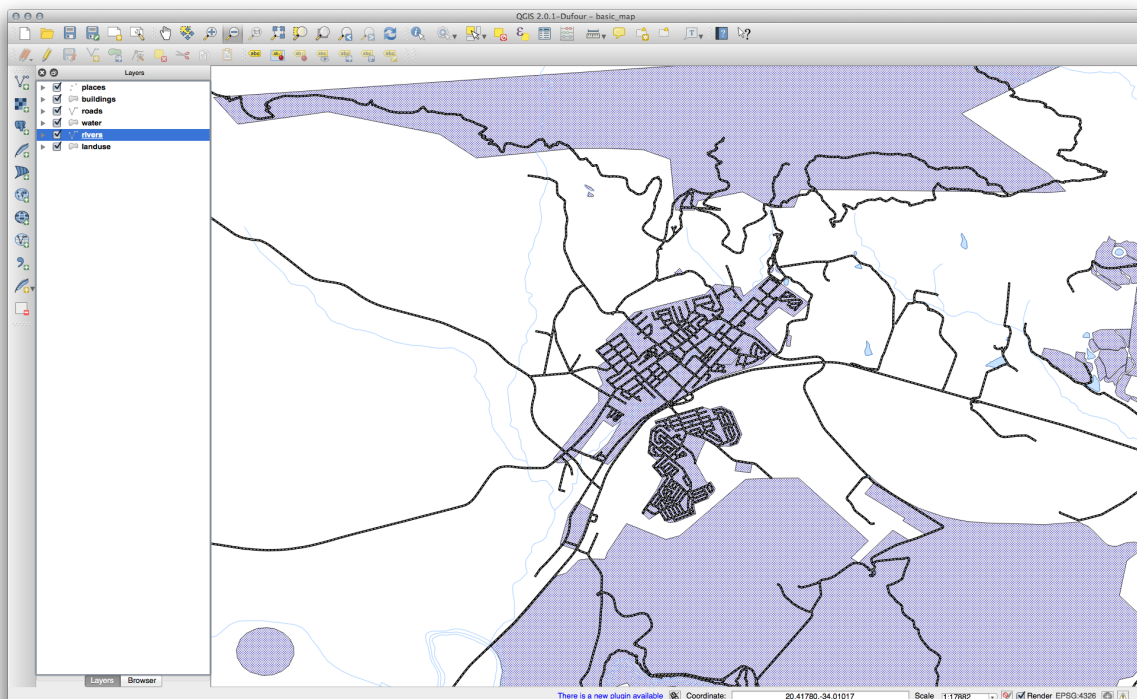
All the vector layers should be loaded into the map. It probably won't look nice yet though (we'll fix the ugly colors later).

[Back to text](#)

18.4 Results For *Symbology*

18.4.1 Colors

- Verify that the colors are changing as you expect them to change.
- It is enough to change only the *water* layer for now. An example is below, but may look different depending on the color you chose.

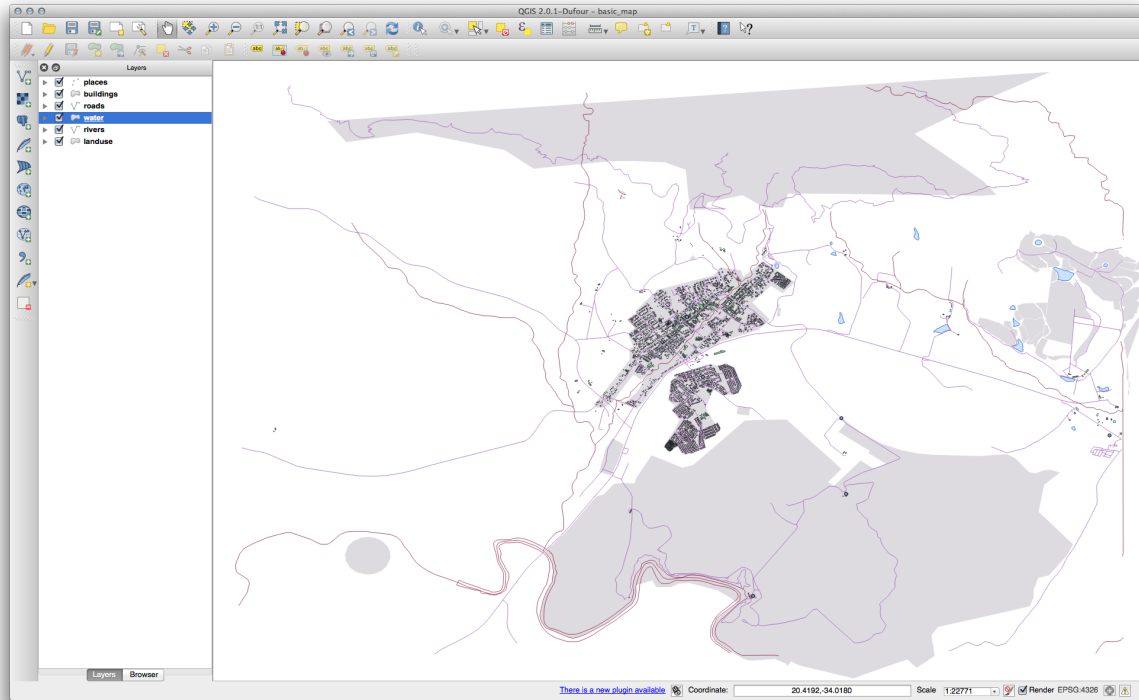


Bemerkung: If you want to work on only one layer at a time and don't want the other layers to distract you, you can hide a layer by clicking in the check box next to its name in the Layers list. If the box is blank, then the layer is hidden.

[Back to text](#)

18.4.2 Symbol Structure

Your map should now look like this:



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

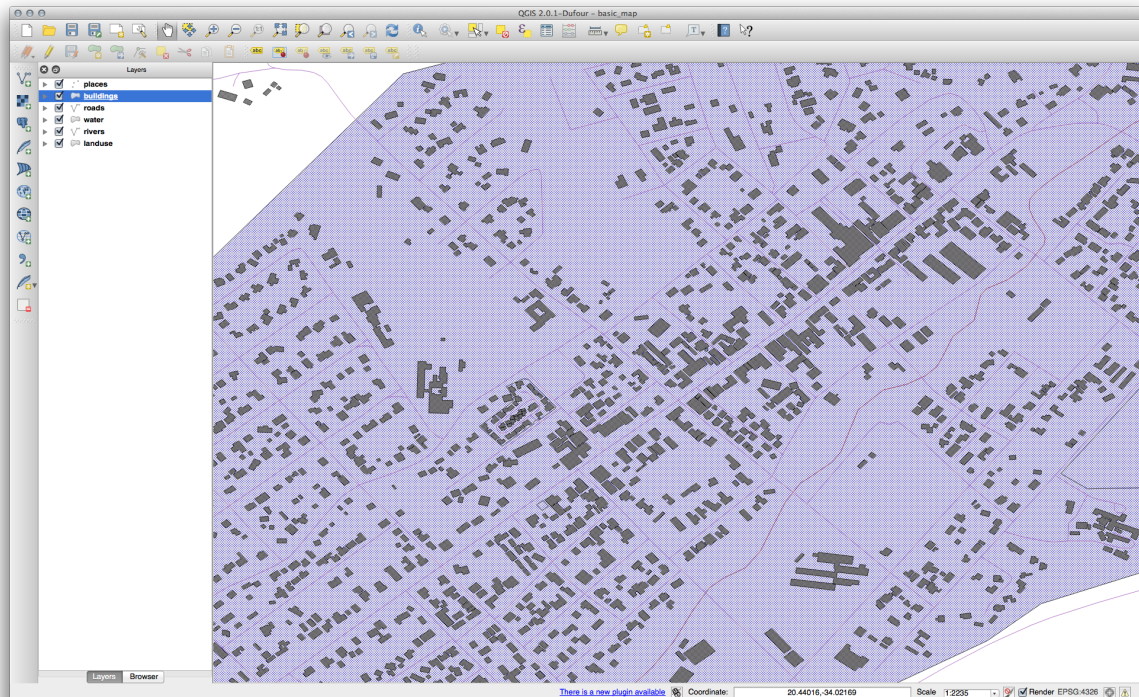
- Use the method above to change the colors and styles for all the remaining layers.
- Try using natural colors for the objects. For example, a road should not be red or blue, but can be gray or black.
- Also feel free to experiment with different *Fill Style* and *Border Style* settings for the polygons.

[Back to text](#)

18.4.3 Symbol Layers

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

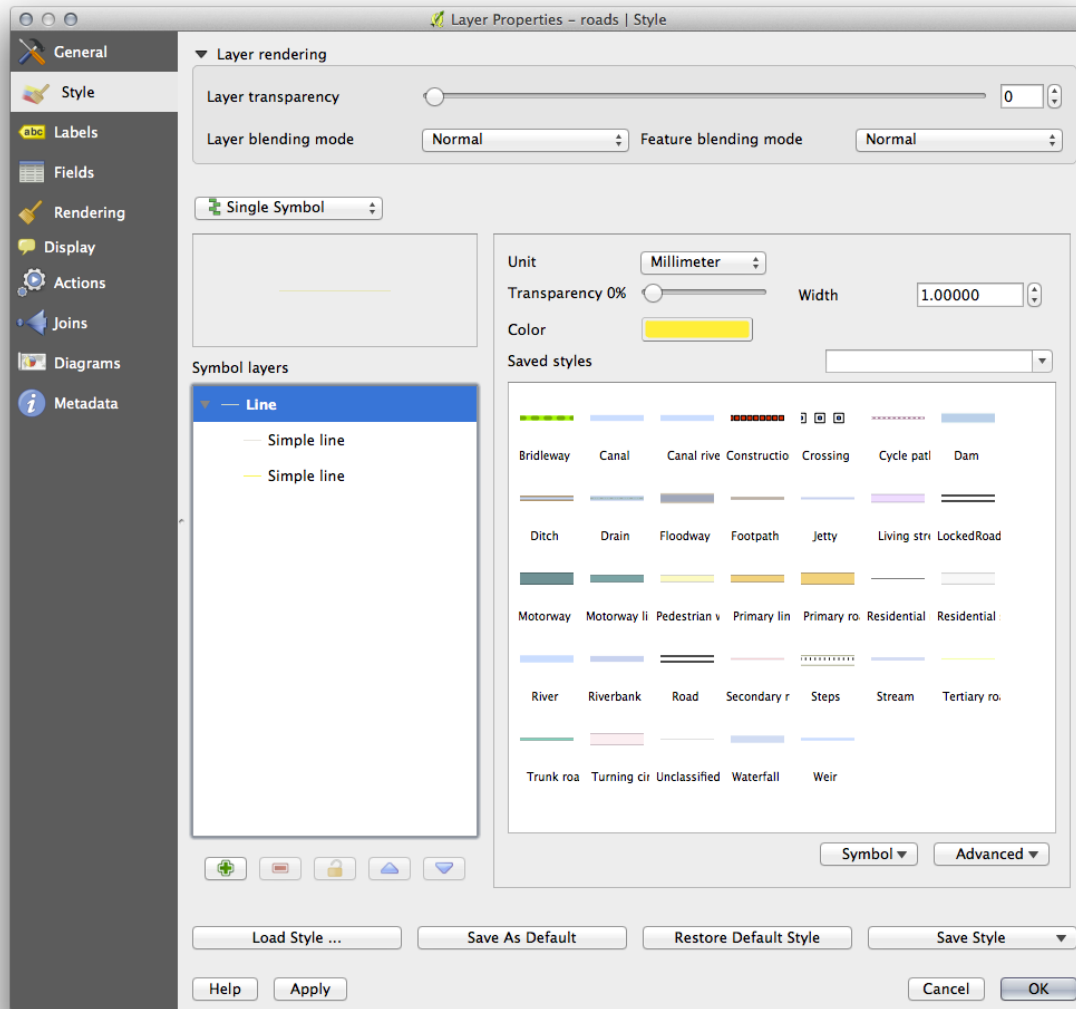
Here's an example:



Back to text

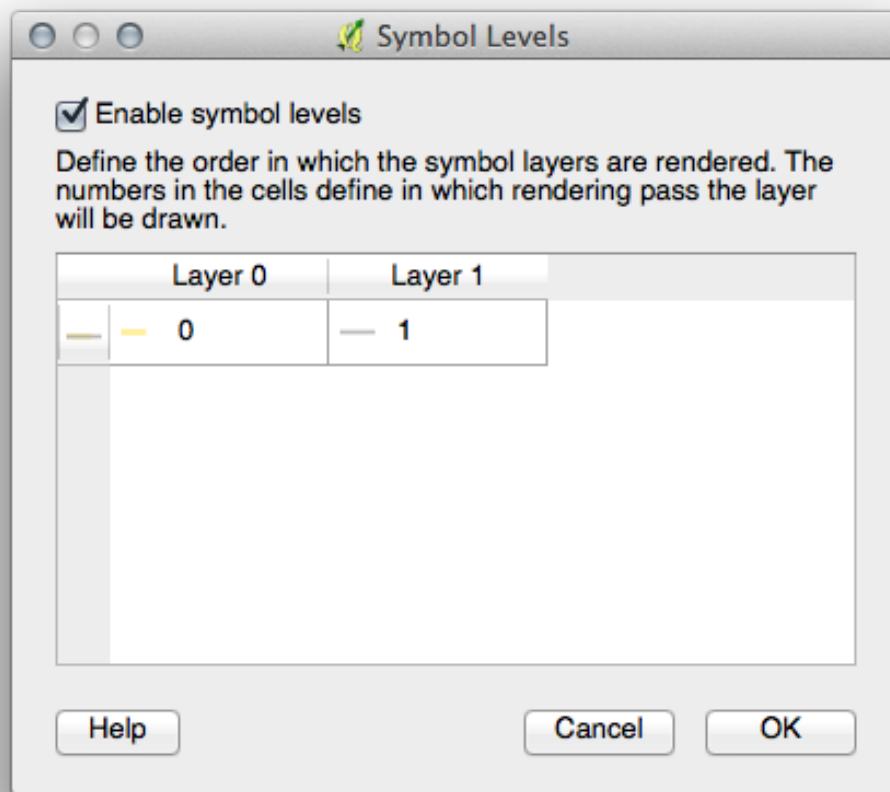
18.4.4 Symbol Levels

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

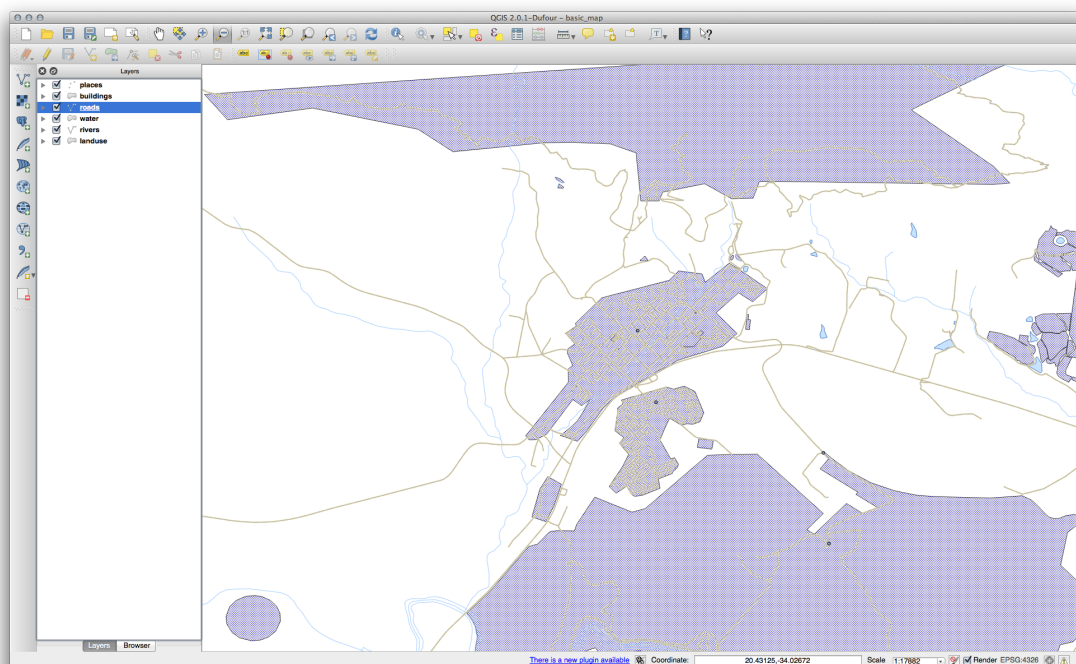


The lowest symbol layer is a broad, solid yellow line. On top of it there is a slightly thinner solid gray line.

- If your symbol layers resemble the above but you're not getting the result you want, check that your symbol levels look something like this:



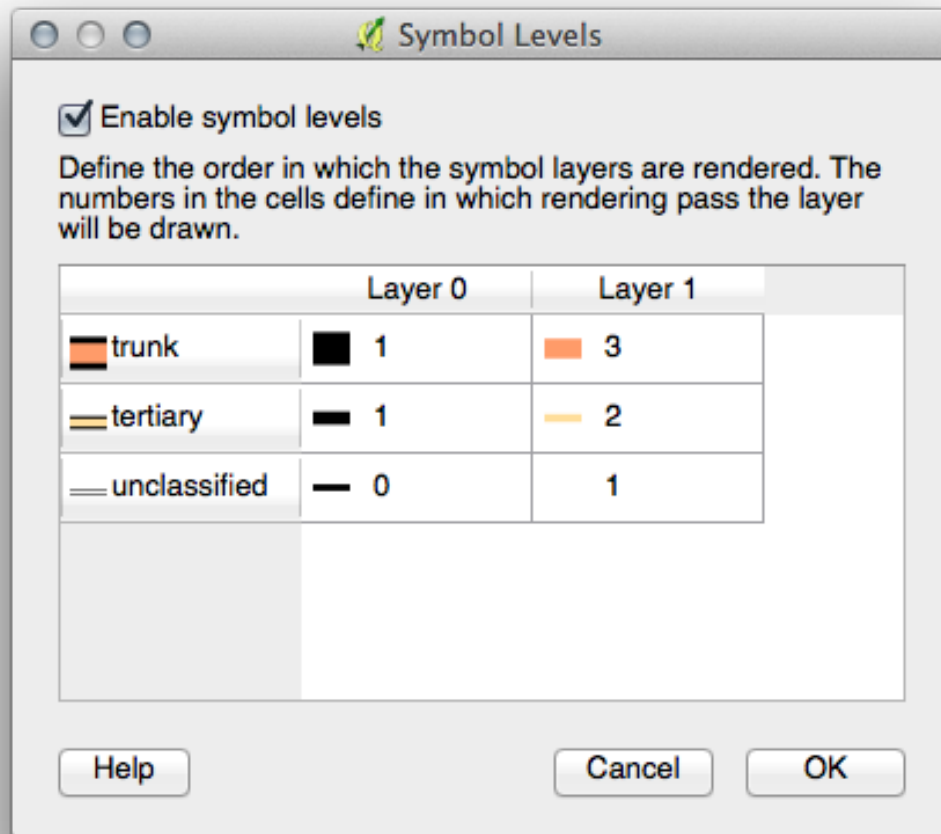
- Now your map should look like this:



[Back to text](#)

18.4.5 Symbol Levels

- Adjust your symbol levels to these values:



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

[Back to text](#)

18.5 Results For Attribute Data

18.5.1 Attribute Data

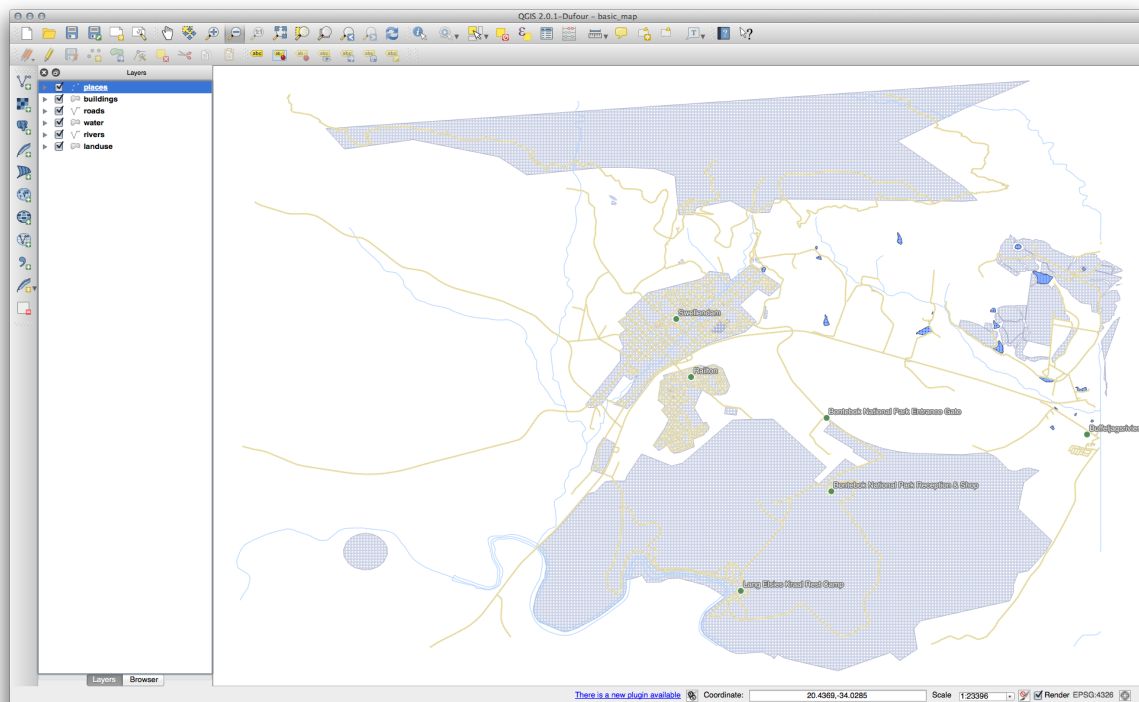
The *NAME* field is the most useful to show as labels. This is because all its values are unique for every object and are very unlikely to contain *NULL* values. If your data contains some *NULL* values, do not worry as long as most of your places have names.

Back to text

18.6 Results For *The Label Tool*

18.6.1 *Label Customization (Part 1)*

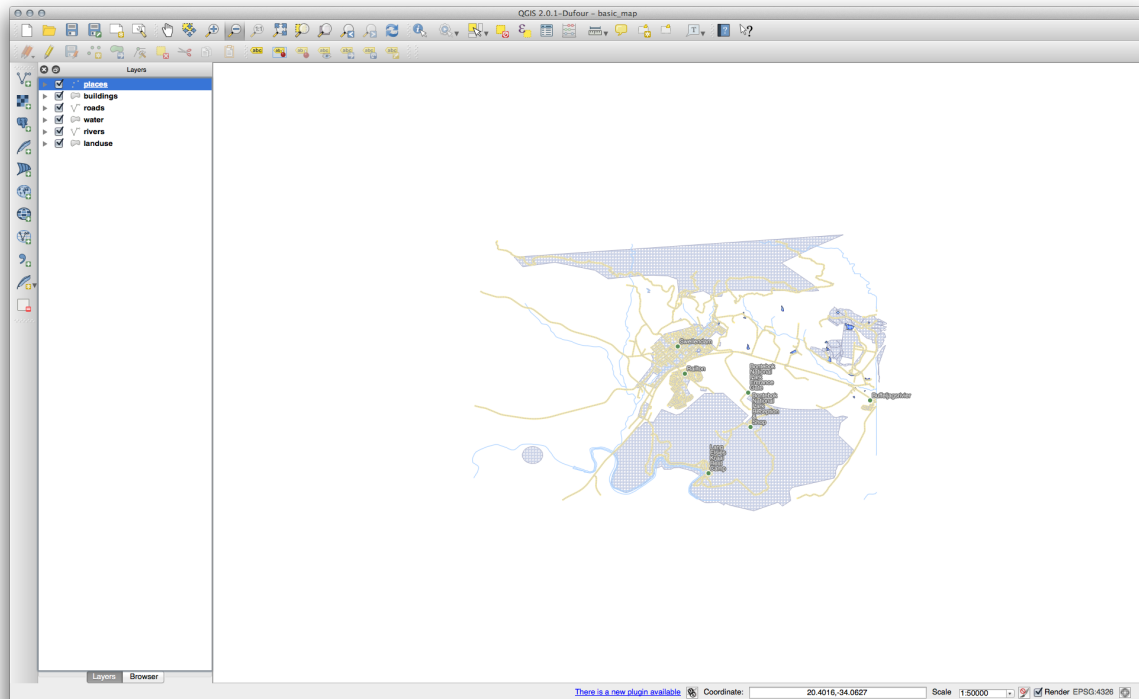
Your map should now show the marker points and the labels should be offset by 2.0 mm: The style of the markers and labels should allow both to be clearly visible on the map:



Back to text

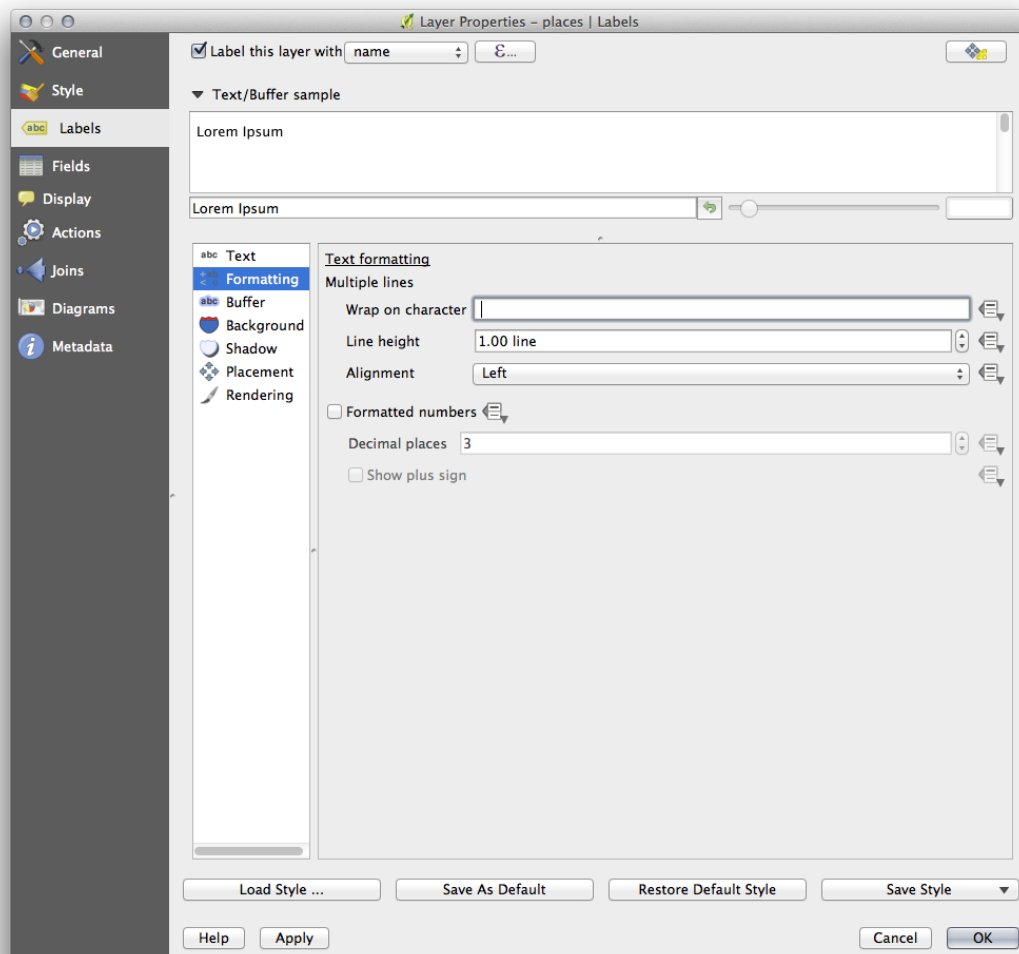
18.6.2 *Label Customization (Part 2)*

One possible solution has this final product:



To arrive at this result:

- Use a font size of 10, a *Label distance* of 1,5 mm, *Symbol width* and *Symbol size* of 3.0 mm.
- In addition, this example uses the *Wrap label on character* option:

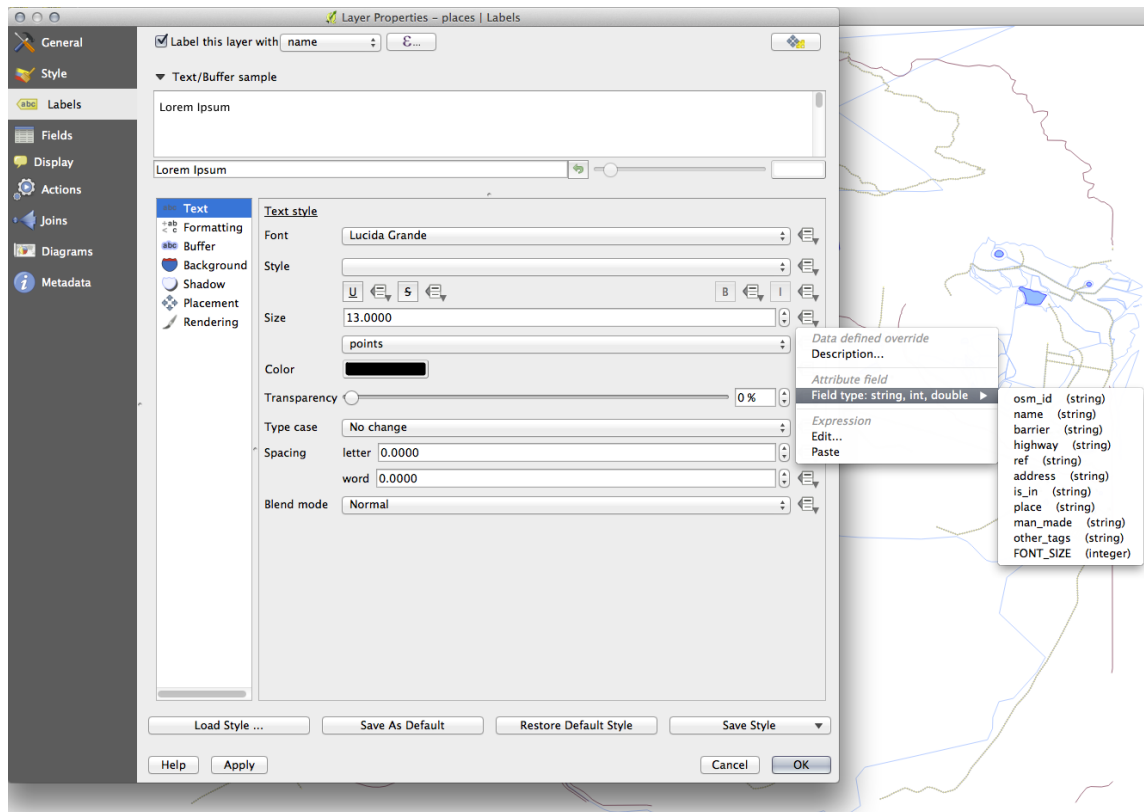


- Enter a space in this field and click *Apply* to achieve the same effect. In our case, some of the place names are very long, resulting in names with multiple lines which is not very user friendly. You might find this setting to be more appropriate for your map.

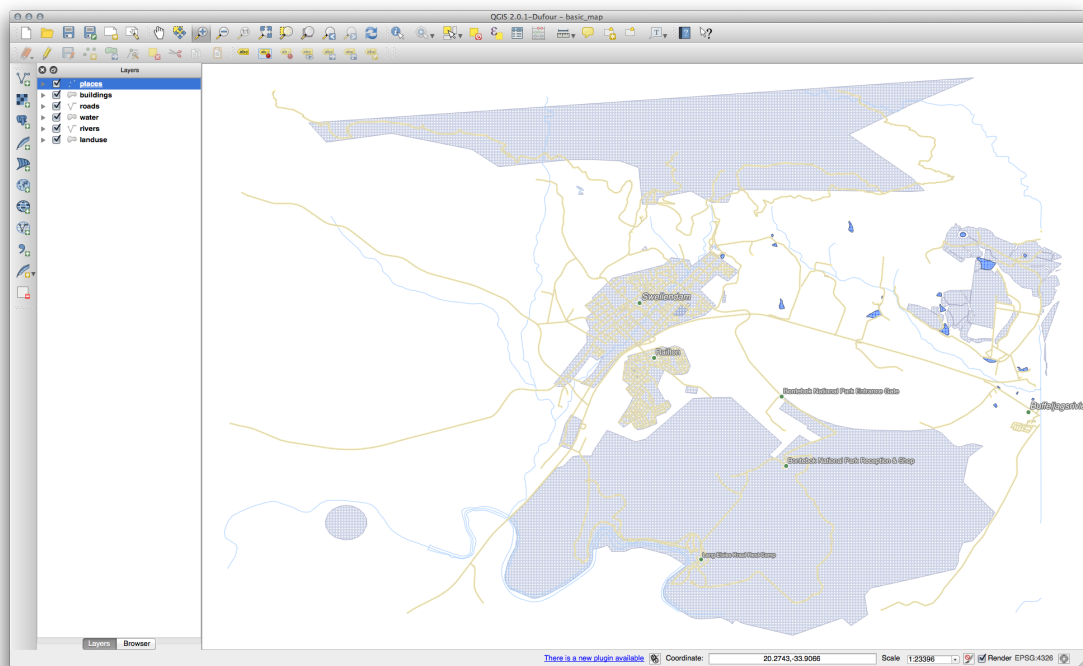
[Back to text](#)

18.6.3 Using Data Defined Settings

- 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.
- Remember to save changes and exit edit mode.
- Return to the *Text* formatting options for the *places* layer and select FONT_SIZE in the *Attribute field* of the font size data override dropdown:



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

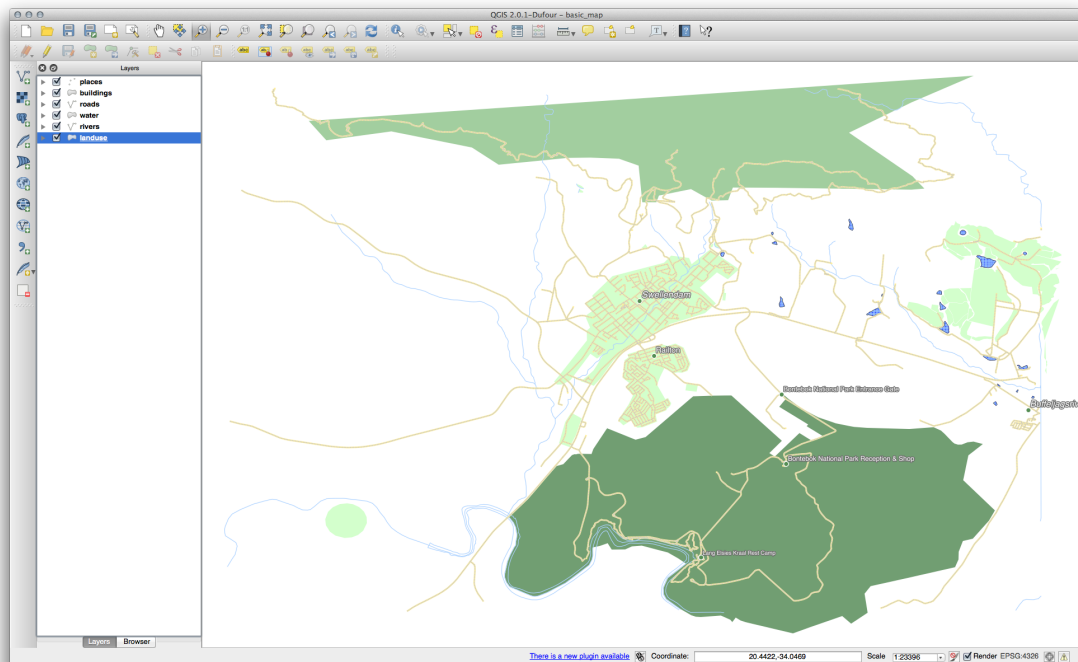


Back to text

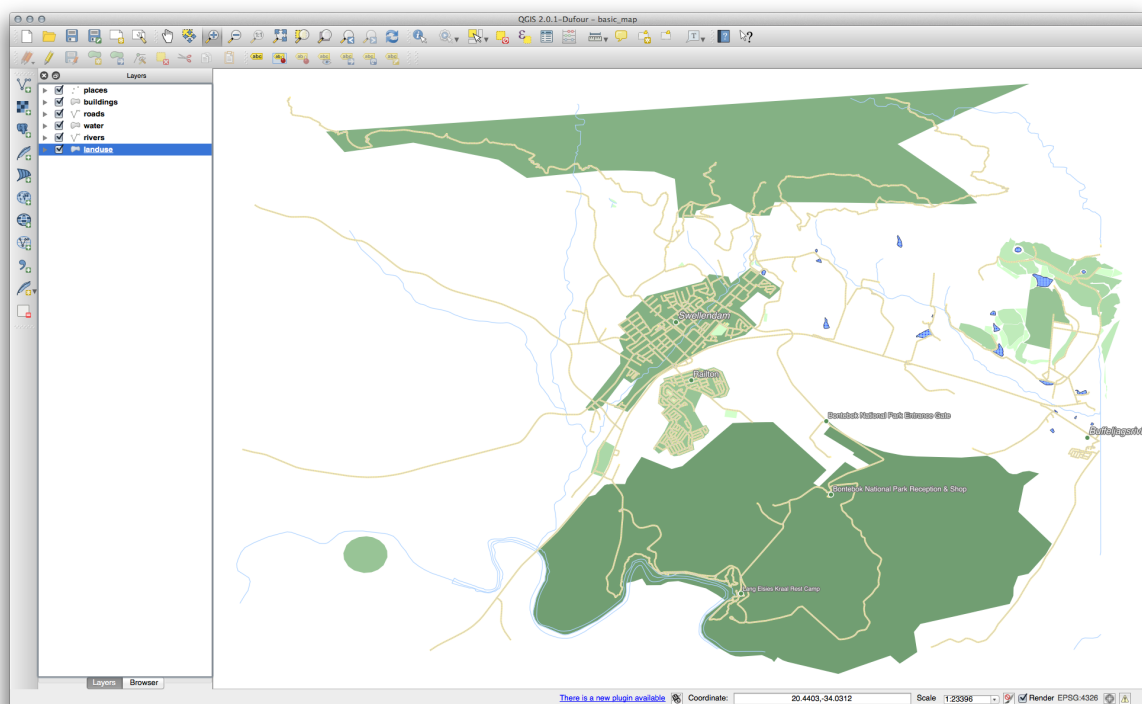
18.7 Results For *Classification*

18.7.1 *Refine the Classification*

- Use the same method as in the first exercise of the lesson to get rid of the borders:



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:

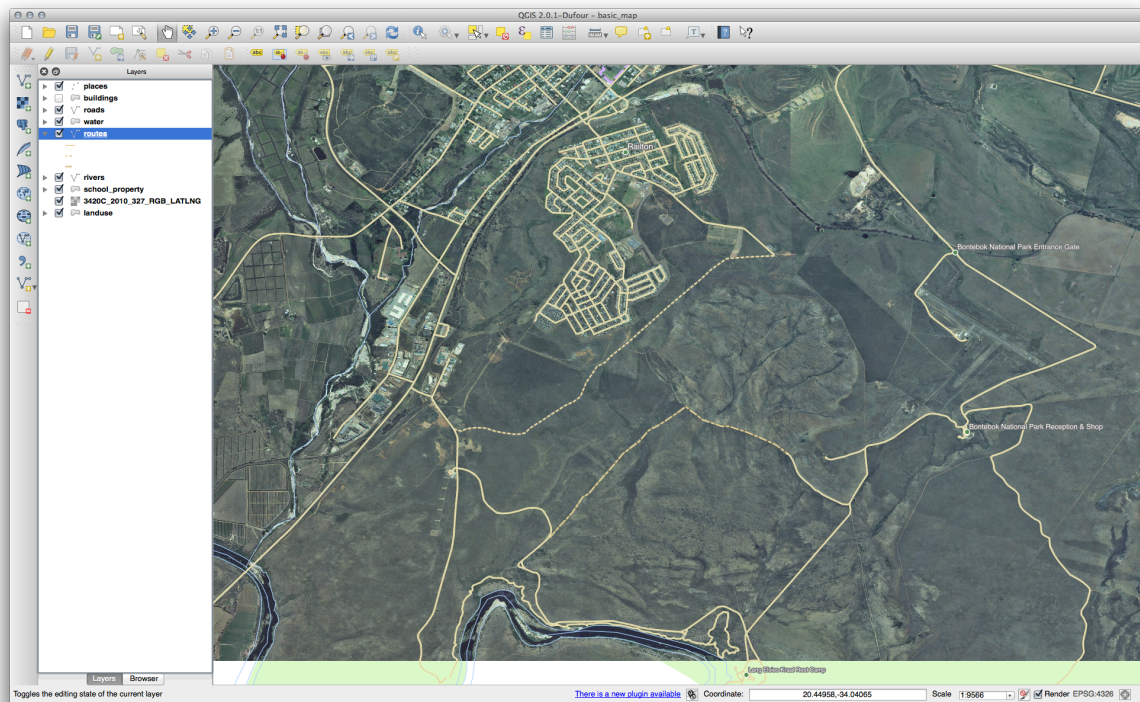


Back to text

18.8 Results For *Creating a New Vector Dataset*

18.8.1 *Digitizing*

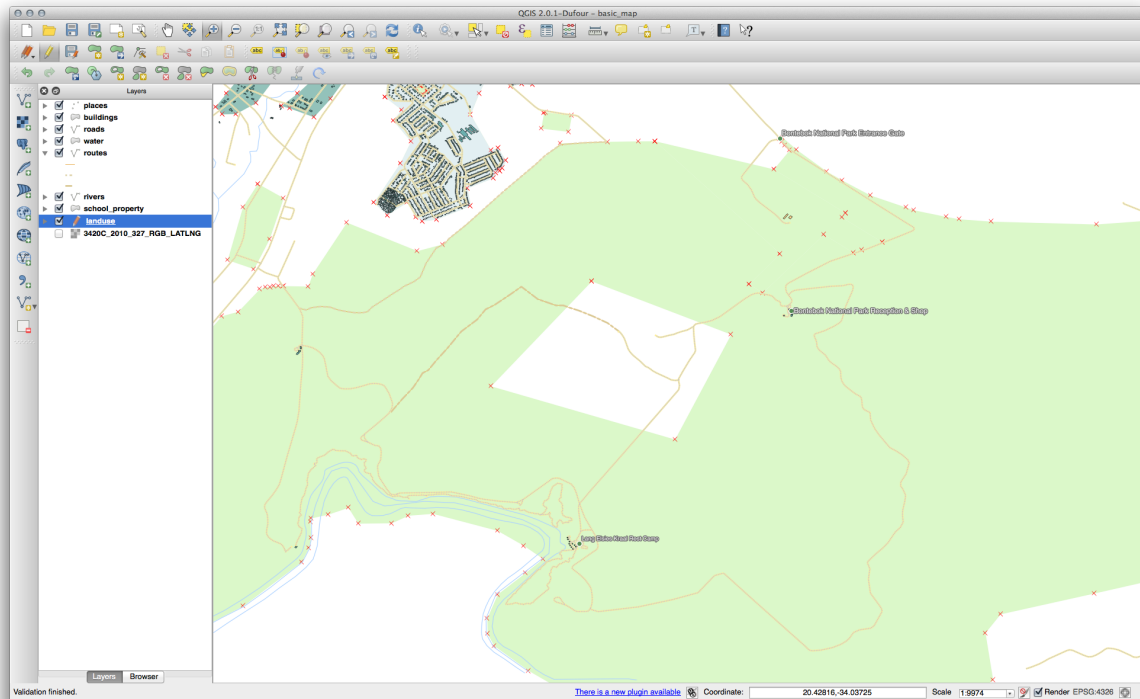
The symbology doesn't matter, but the results should look more or less like this:



Back to text

18.8.2 *Topology: Add Ring Tool*

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

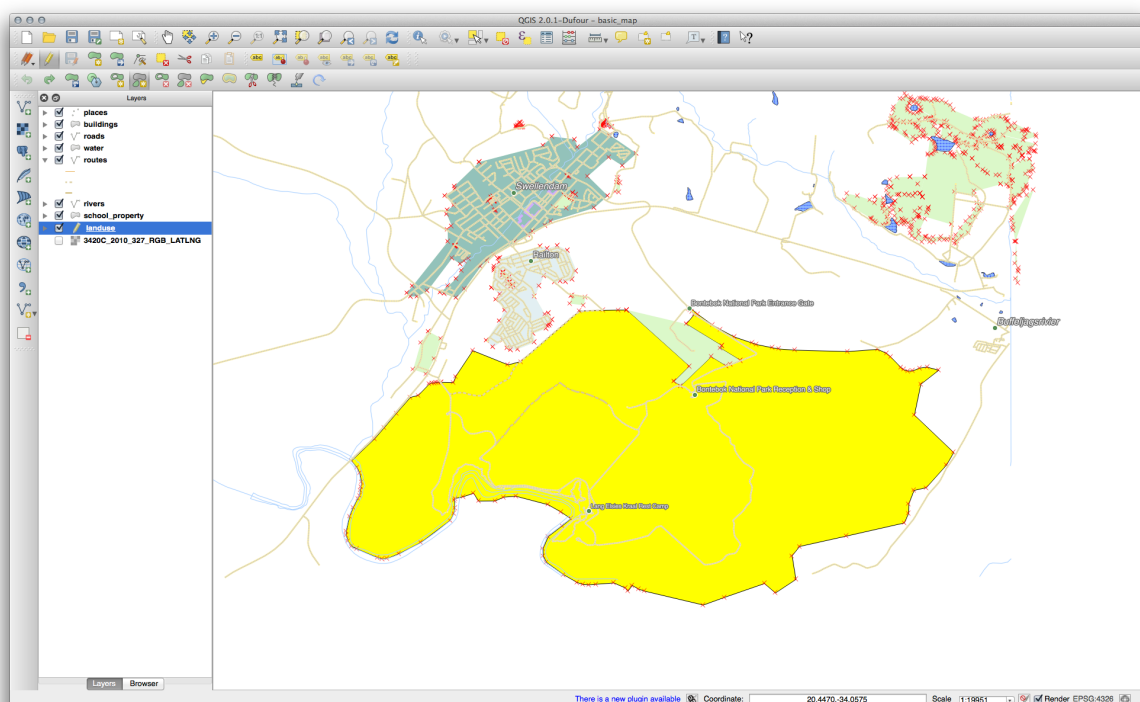


- Undo your edit before continuing with the exercise for the next tool.

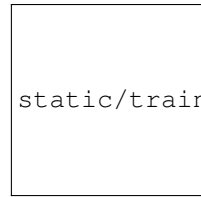
Back to text

18.8.3 Topology: Add Part Tool

- First select the Bontebok National Part:



- Now add your new part:



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

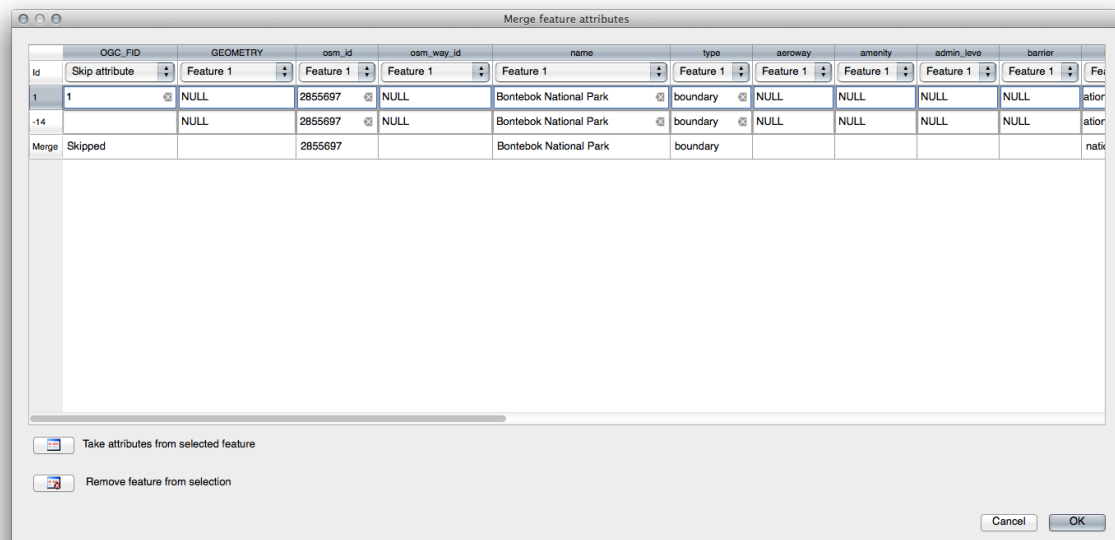
[Back to text](#)

18.8.4 Merge Features

- 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):

Bemerkung:

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*.



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

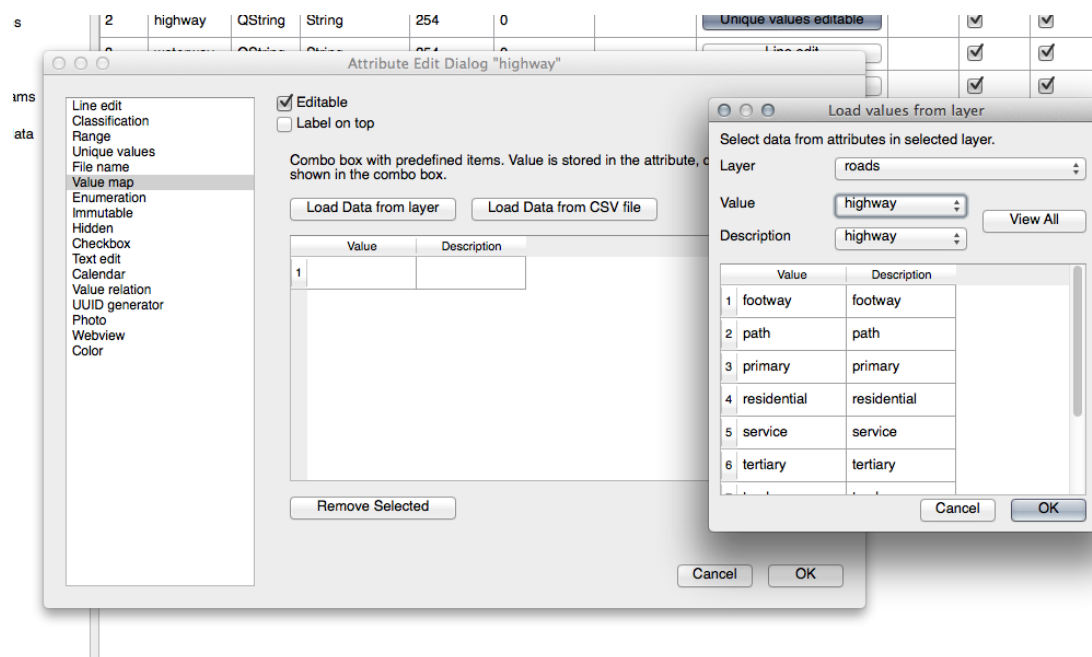
[Back to text](#)

18.8.5 Forms

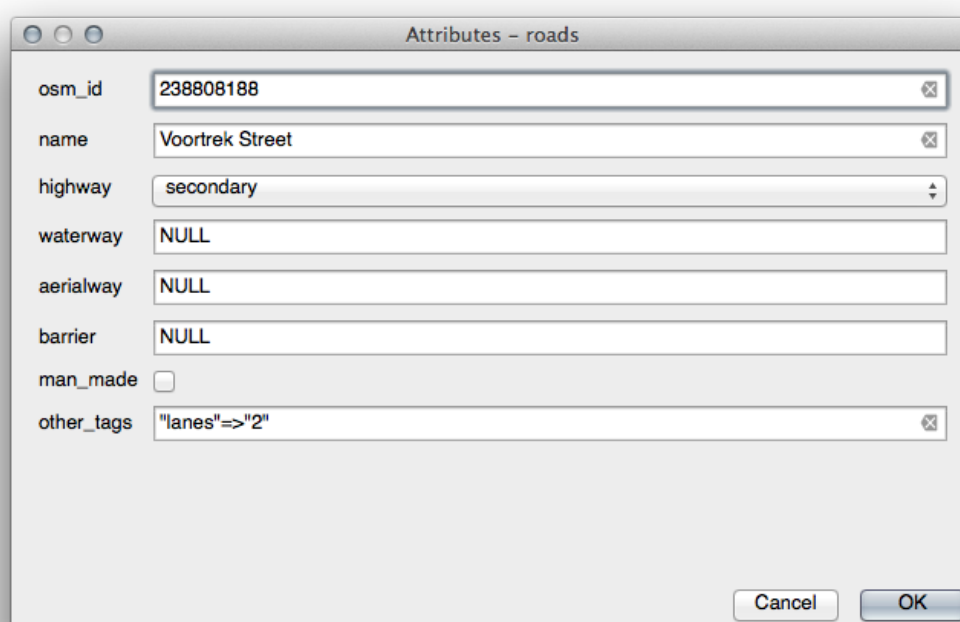
For the *TYPE*, there is obviously a limited amount of types that a road can be, and if you check the attribute table for this layer, you'll see that they are predefined.

- Set the widget to *Value Map* and click *Load Data from Layer*.

- Select *roads* in the *Label* dropdown and *highway* for both the *Value* and *Description* options:



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



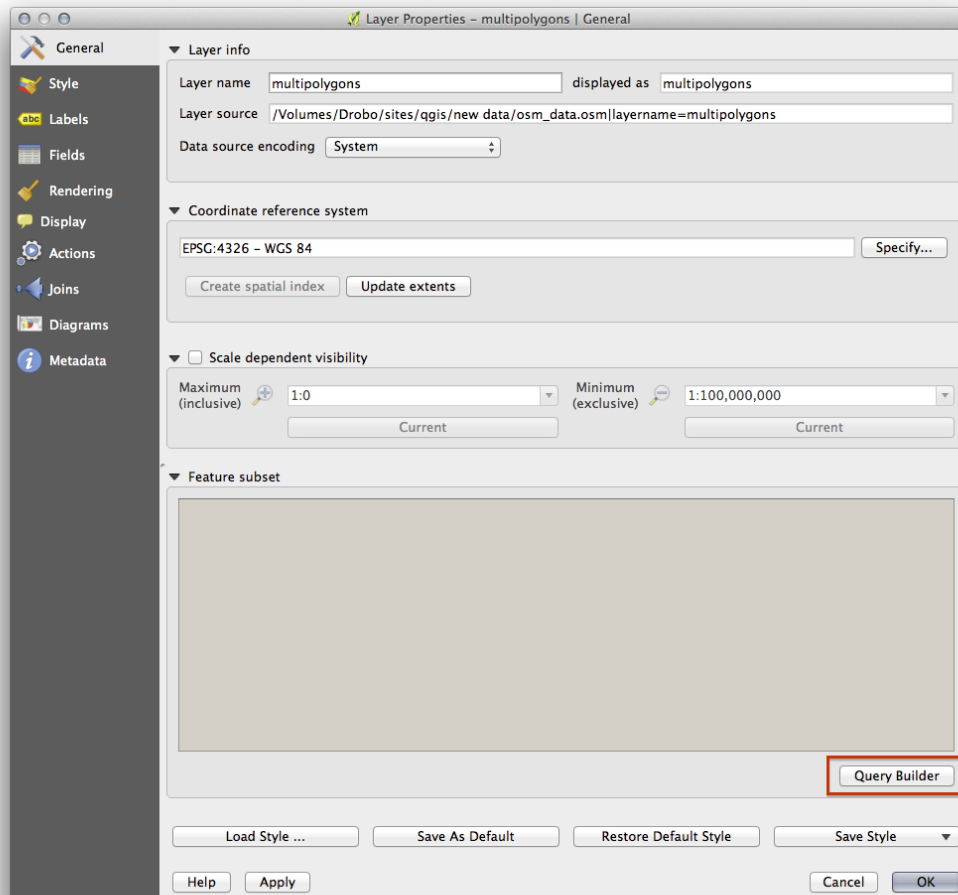
[Back to text](#)

18.9 Results For Vector Analysis

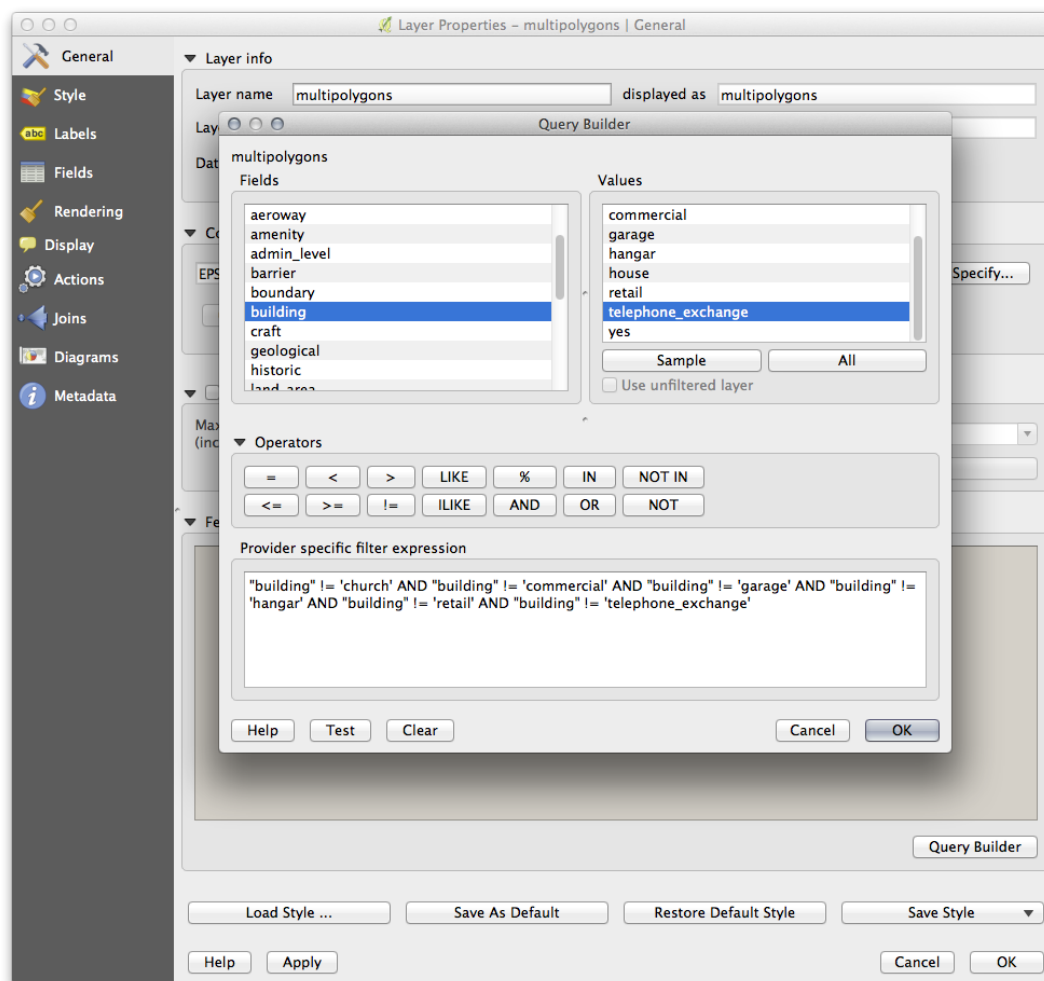
18.9.1 Extract Your Layers from OSM Data

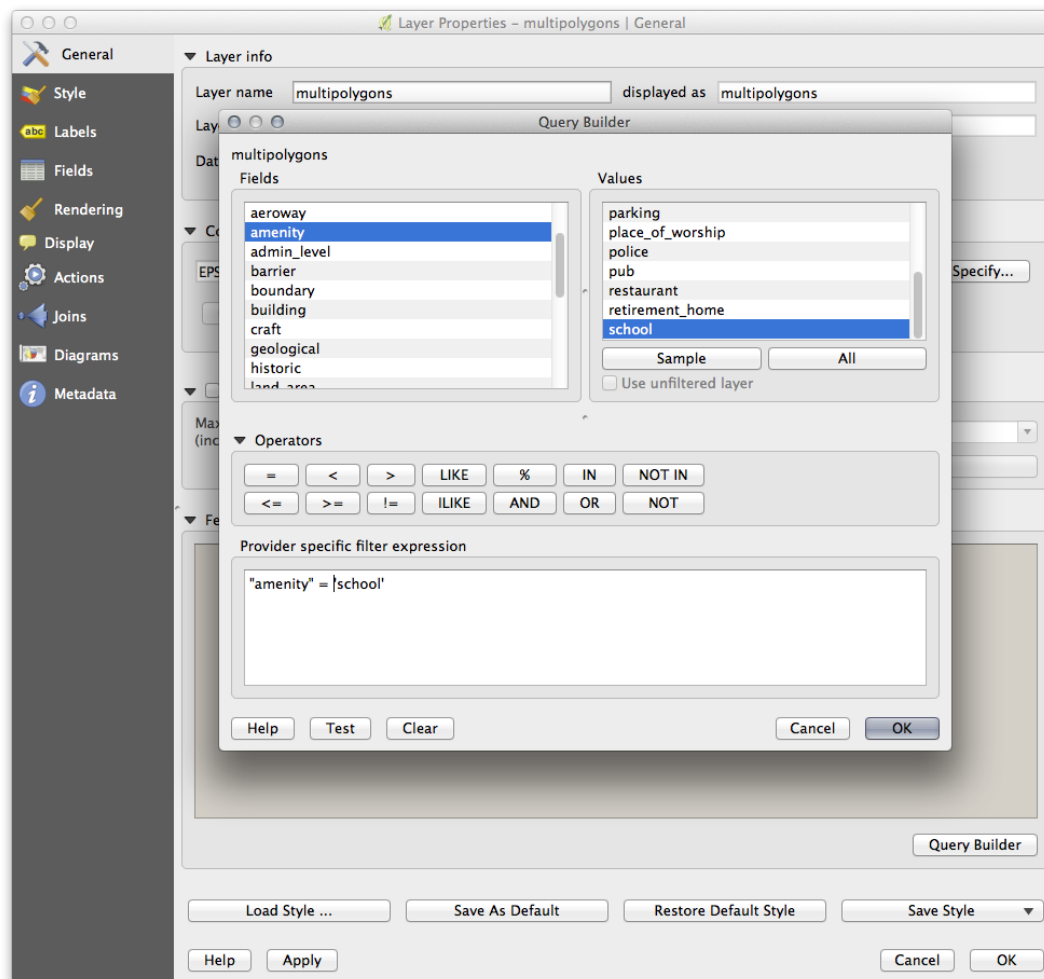
For the purpose of this exercise, the OSM layers which we are interested in are multipolygons and lines. The multipolygons layer contains the data we need in order to produce the houses, schools and restaurants layers. The lines layer contains the roads dataset.

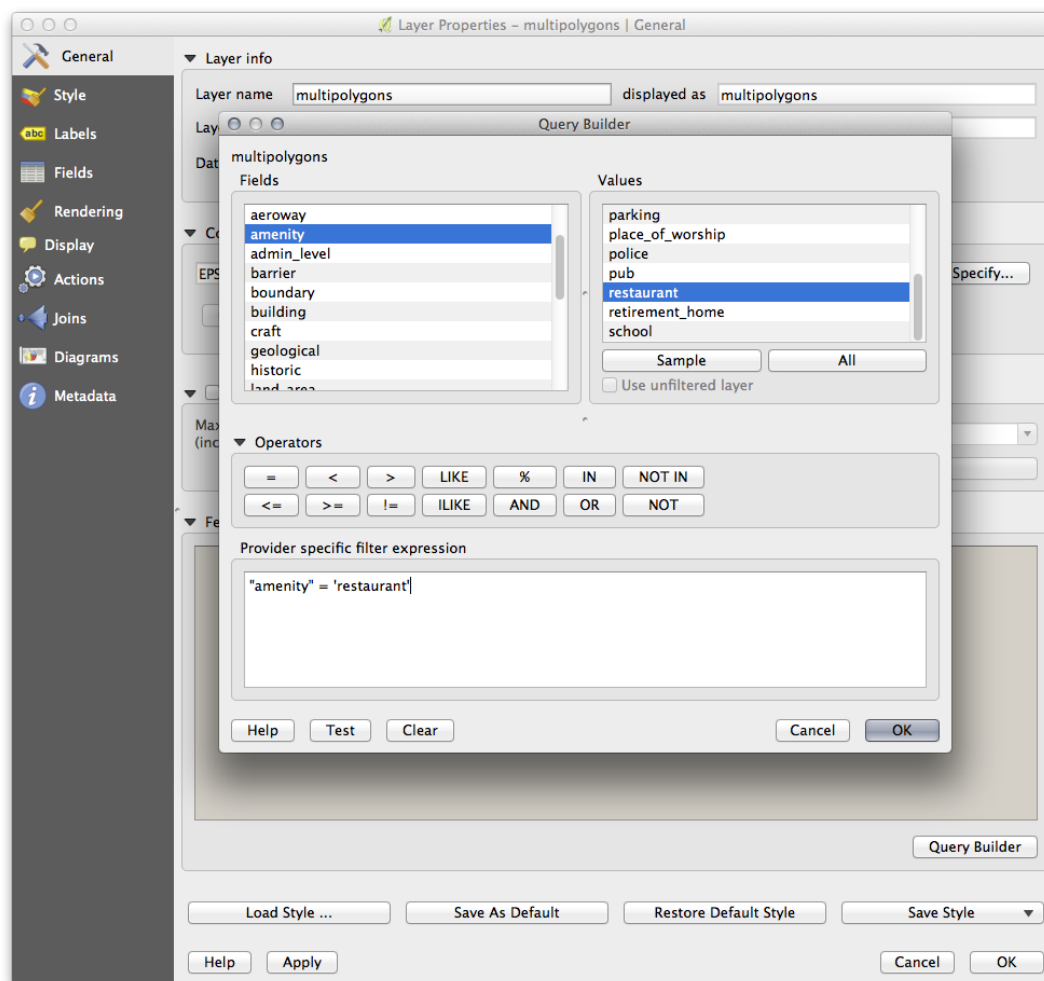
The *Query Builder* is found in the layer properties:



Using the *Query Builder* against the multipolygon layer, create the following queries for the houses, schools, restaurants and residential layers:





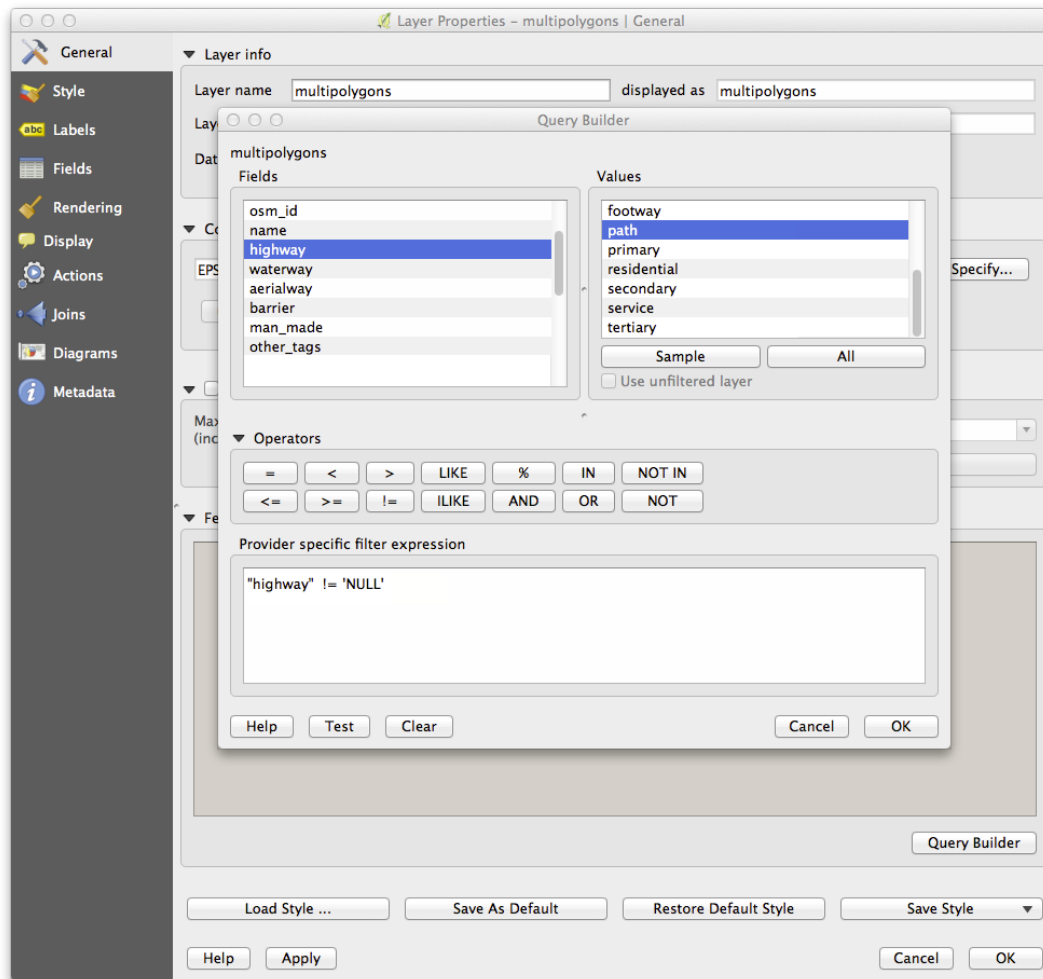


Once you have entered each query, click *OK*. You'll see that the map updates to show only the data you have selected. Since you need to use again the multipolygon data from the OSM dataset, at this point, you can use one of the following methods:

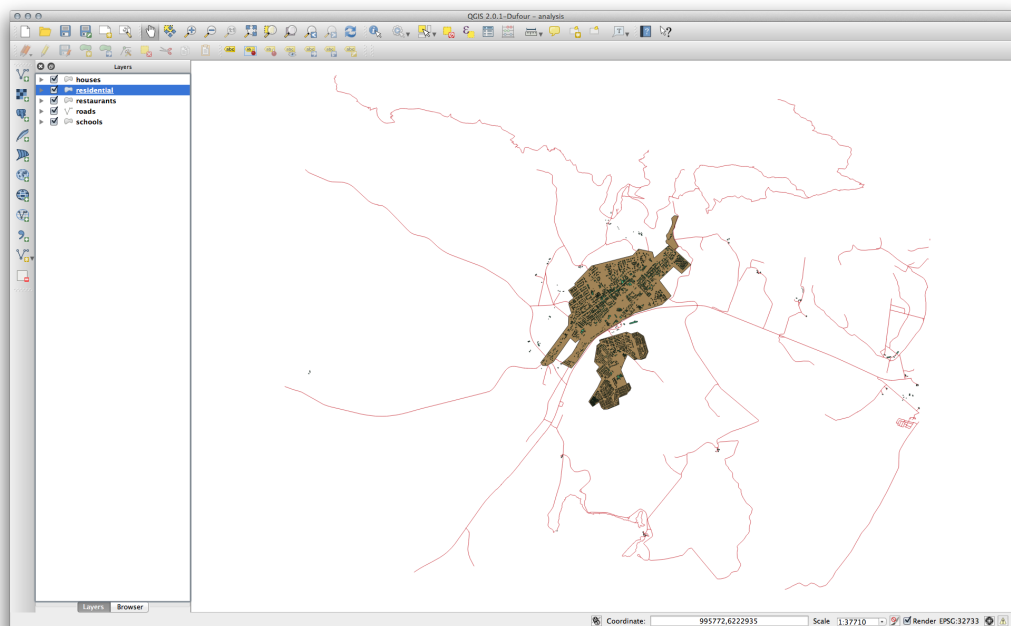
- Rename the filtered OSM layer and re-import the layer from `osm_data.osm`, OR
- Duplicate the filtered layer, rename the copy, clear the query and create your new query in the *Query Builder*.

Bemerkung: Although OSM's building field has a house value, the coverage in your area - as in ours - may not be complete. In our test region, it is therefore more accurate to *exclude* all buildings which are defined as anything other than house. You may decide to simply include buildings which are defined as house and all other values that have not a clear meaning like yes.

To create the roads layer, build this query against OSM's lines layer:



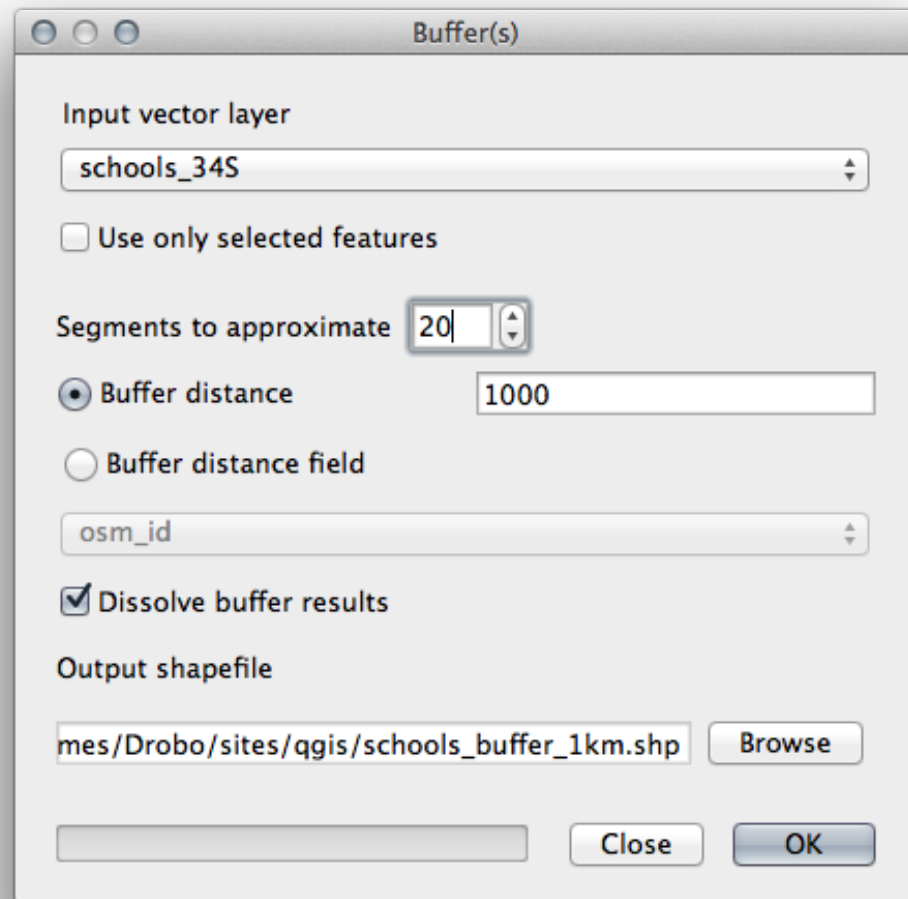
You should end up with a map which looks similar to the following:



[Back to text](#)

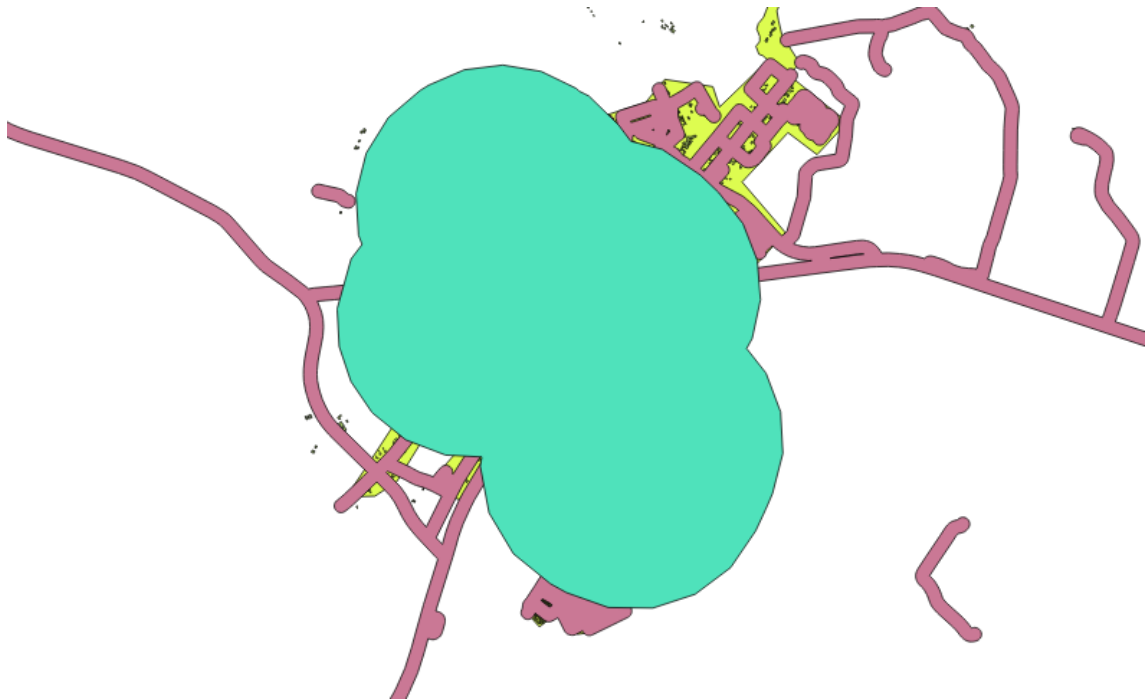
18.9.2 Distance from High Schools

- Your buffer dialog should look like this:



The *Buffer distance* is 1000 meters (i.e., 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:



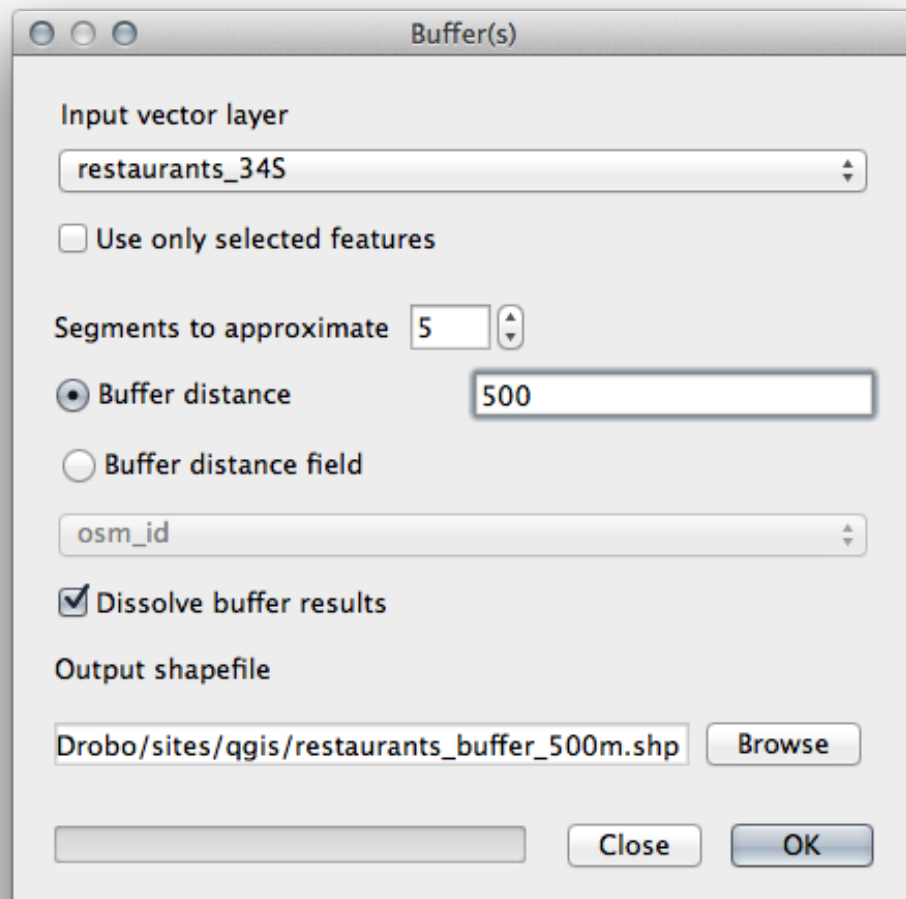
The first image shows the buffer with the *Segments to approximate* value set to 5 and the second shows the value set to 20. In our example, the difference is subtle, but you can see that the buffer's edges are smoother with the higher value.

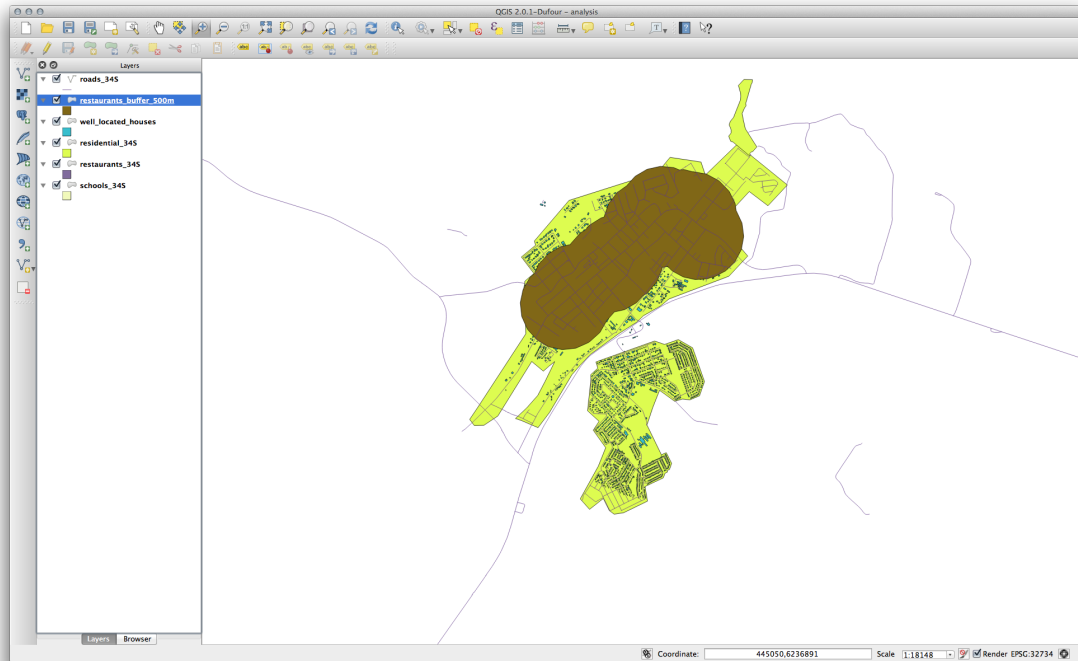
[Back to text](#)

18.9.3 Distance from Restaurants

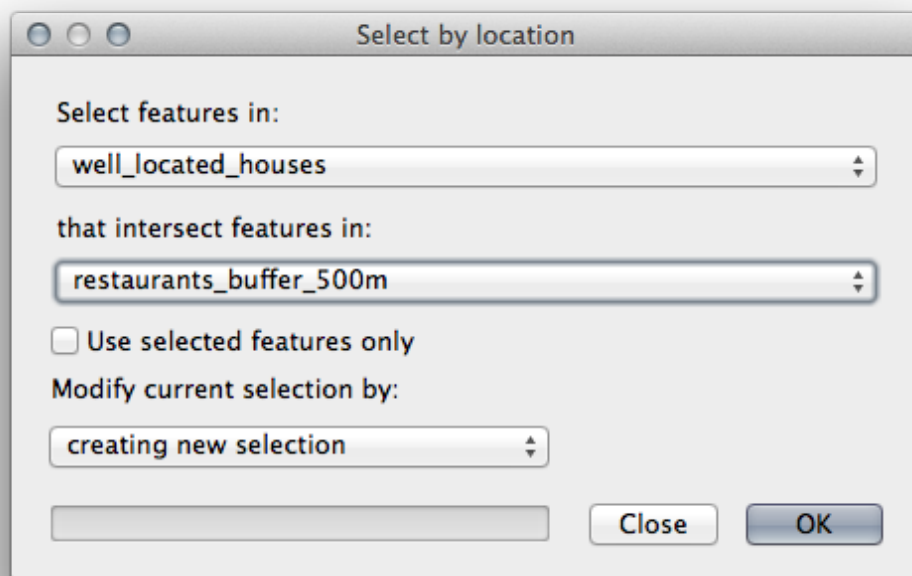
To create the new `houses_restaurants_500m` layer, we go through a two step process:

- First, create a buffer of 500m around the restaurants and add the layer to the map:

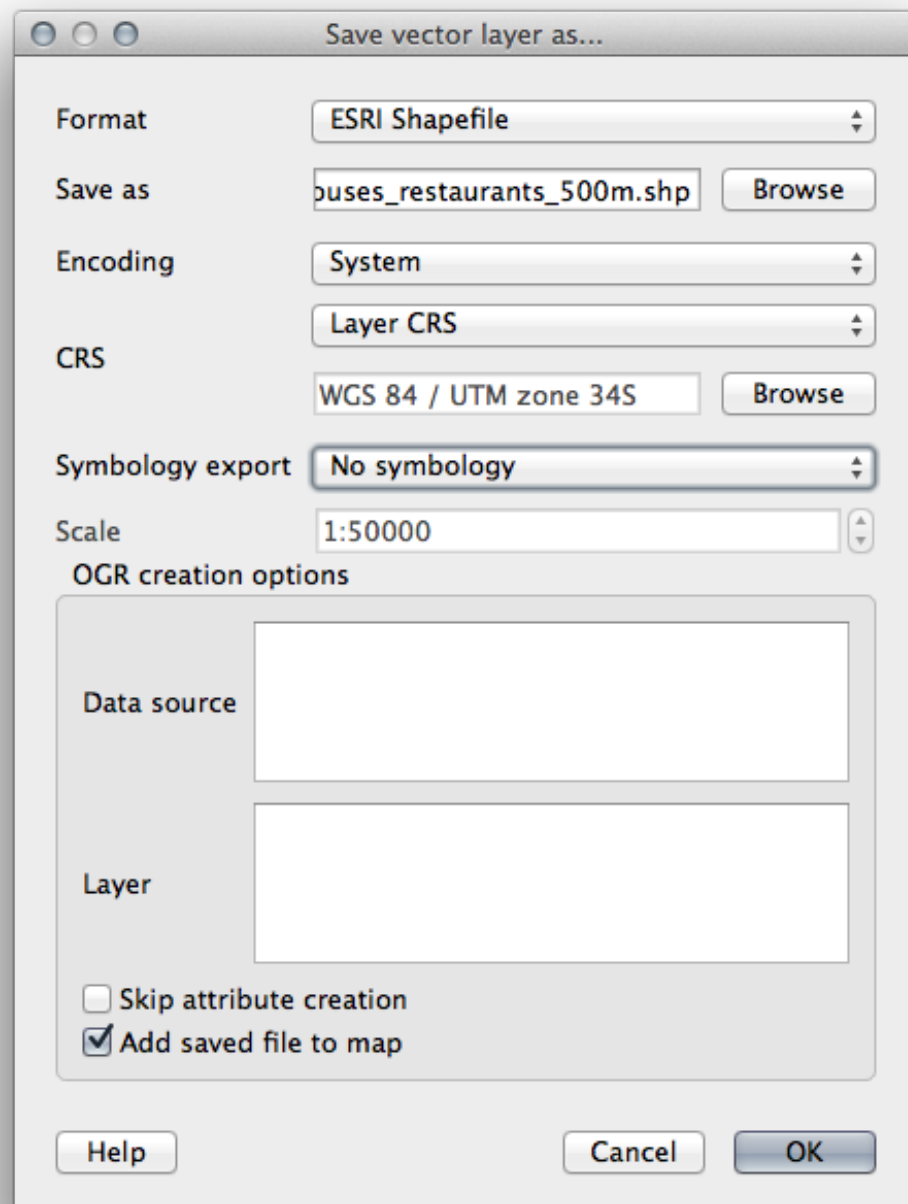




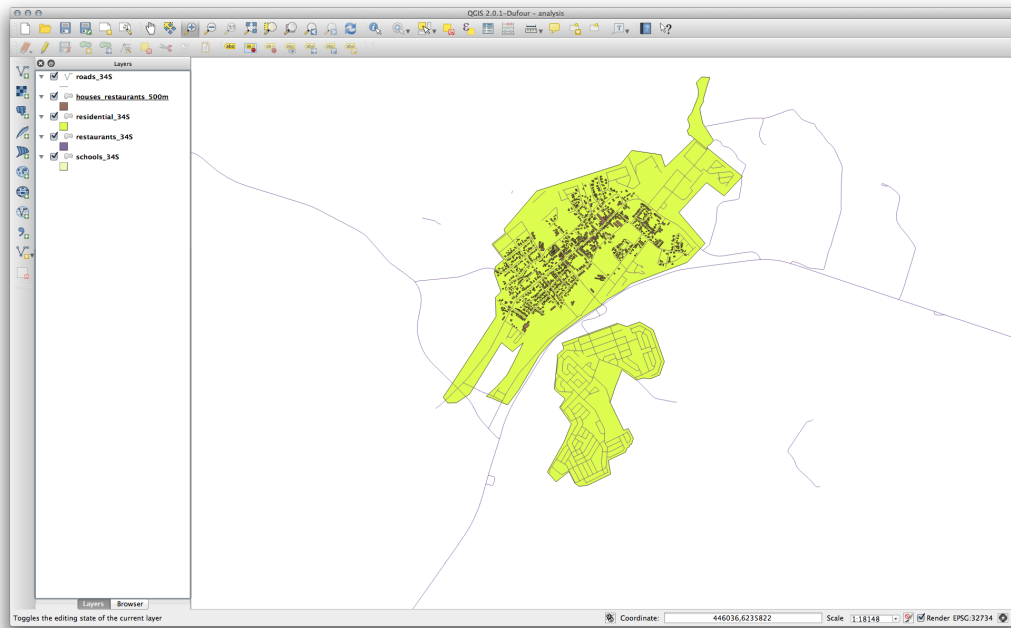
- Next, select buildings within that buffer area:



- Now save that selection to our new houses_restaurants_500m layer:



Your map should now show only those buildings which are within 50m of a road, 1km of a school and 500m of a restaurant:

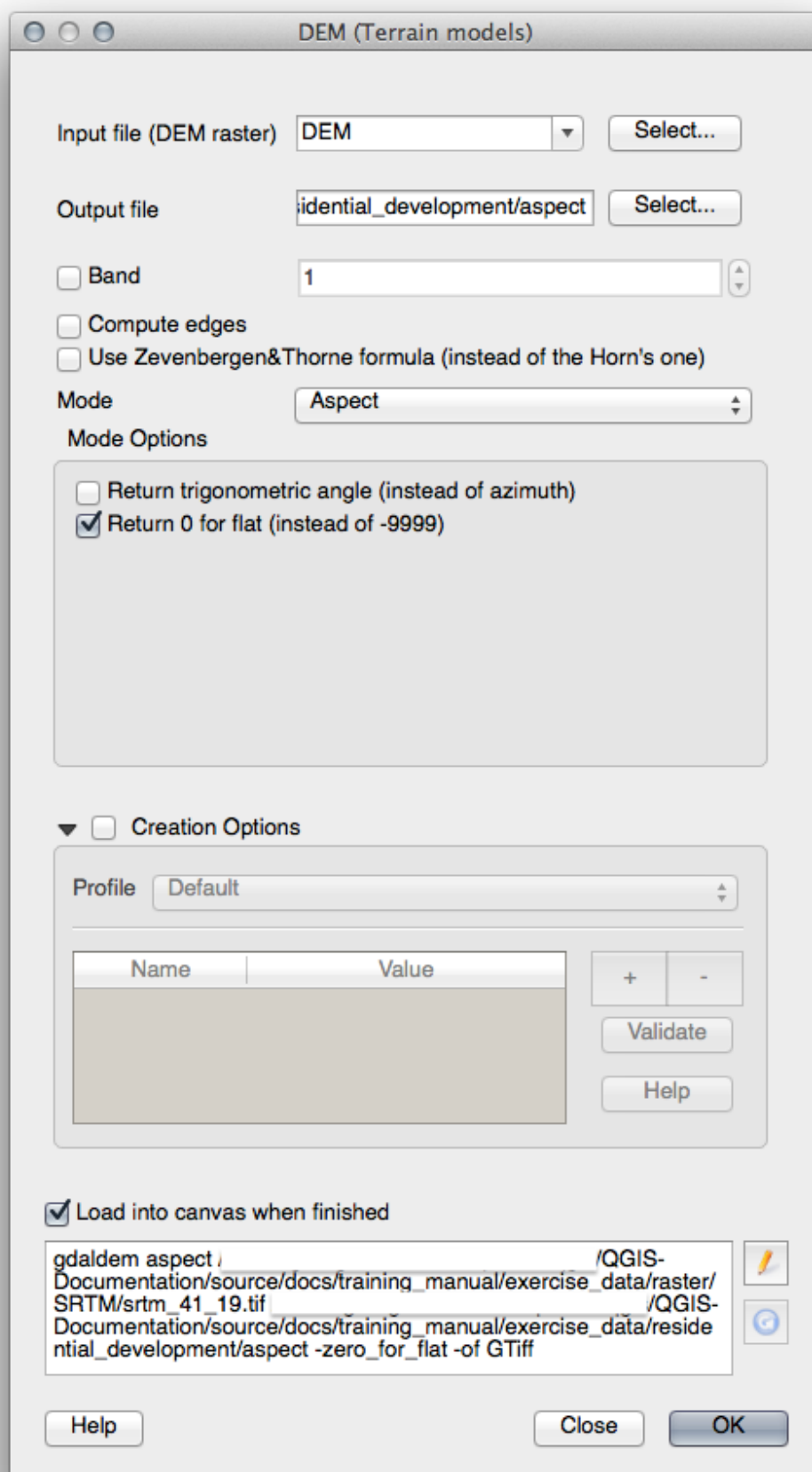


[Back to text](#)

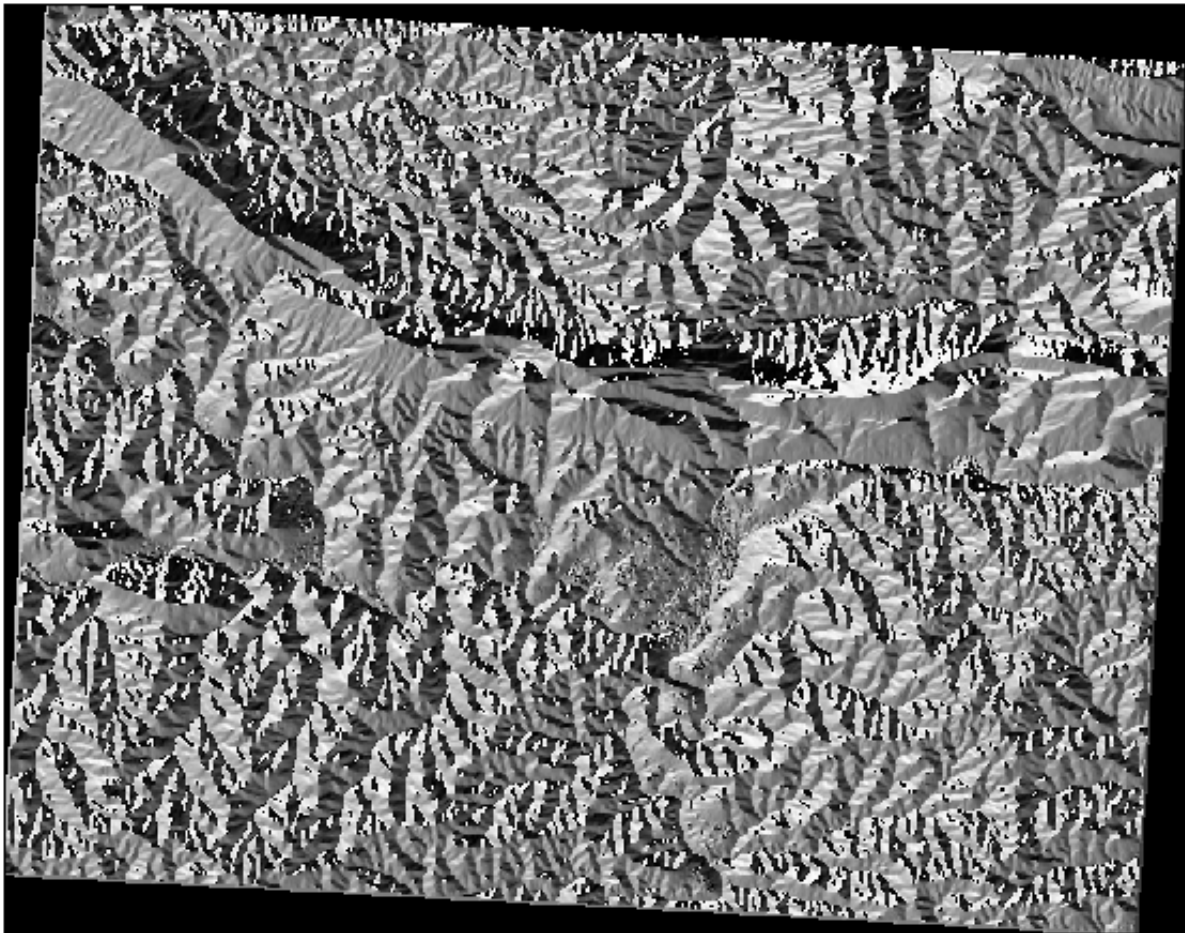
18.10 Results For *Raster Analysis*

18.10.1 *Calculate Aspect*

- Set your *DEM (Terrain analysis)* dialog up like this:



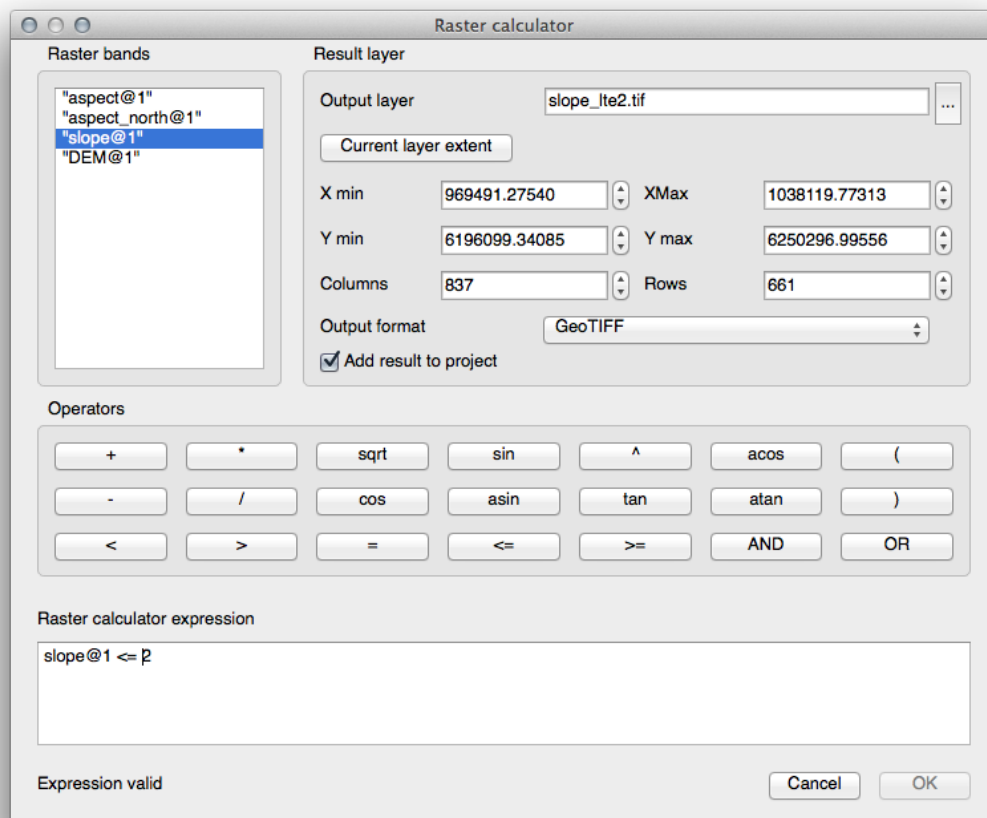
Your result:



[Back to text](#)

18.10.2 Calculate Slope (less than 2 and 5 degrees)

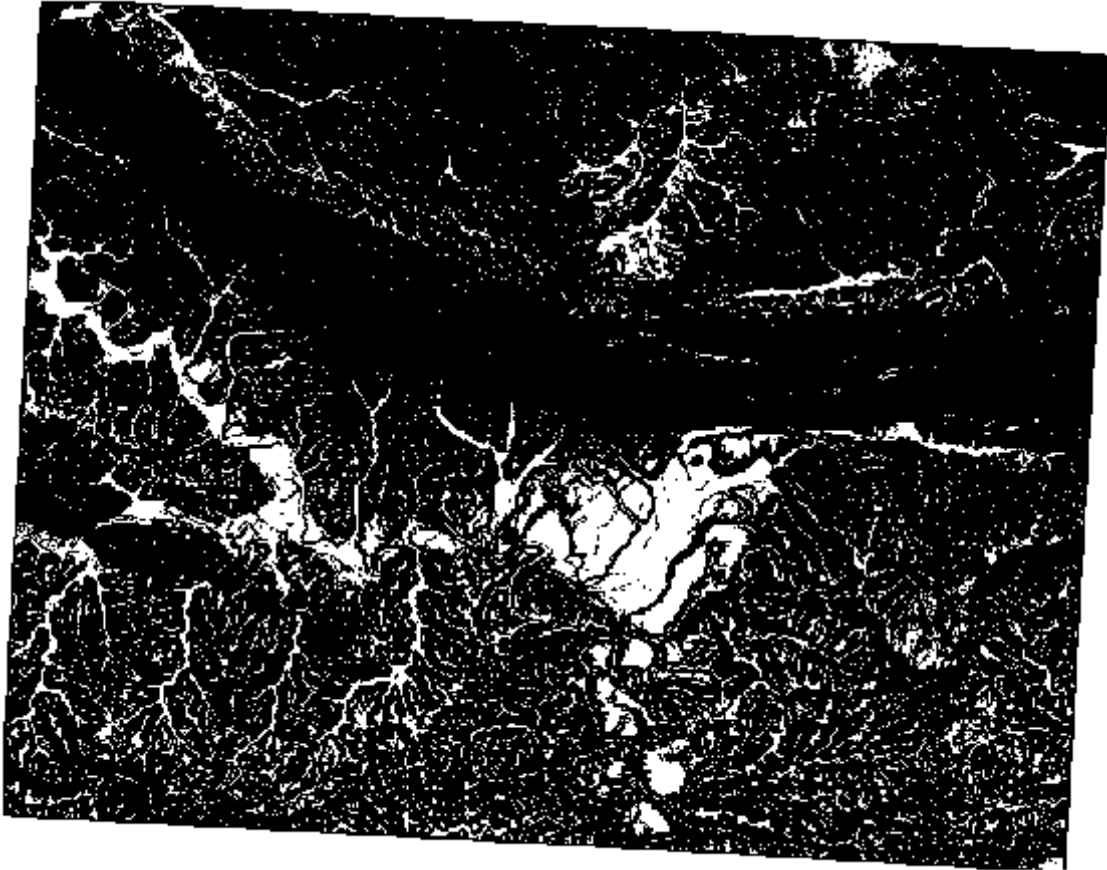
- Set your *Raster calculator* dialog up like this:



- For the 5 degree version, replace the 2 in the expression and file name with 5.

Your results:

- 2 degrees:



- 5 degrees:



[Back to text](#)

18.11 Results For *Completing the Analysis*

18.11.1 *Raster to Vector*

- Open the *Query Builder* by right-clicking on the *all_terrain* layer in the *Layers list*, select the *General* tab.
- Then build the query "suitable" = 1.
- 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:

static/training_manual/c

- You can save this layer by right-clicking on the *all_terrain* layer in the *Layers list* and choosing *Save As...*, then continue as per the instructions.

[Back to text](#)

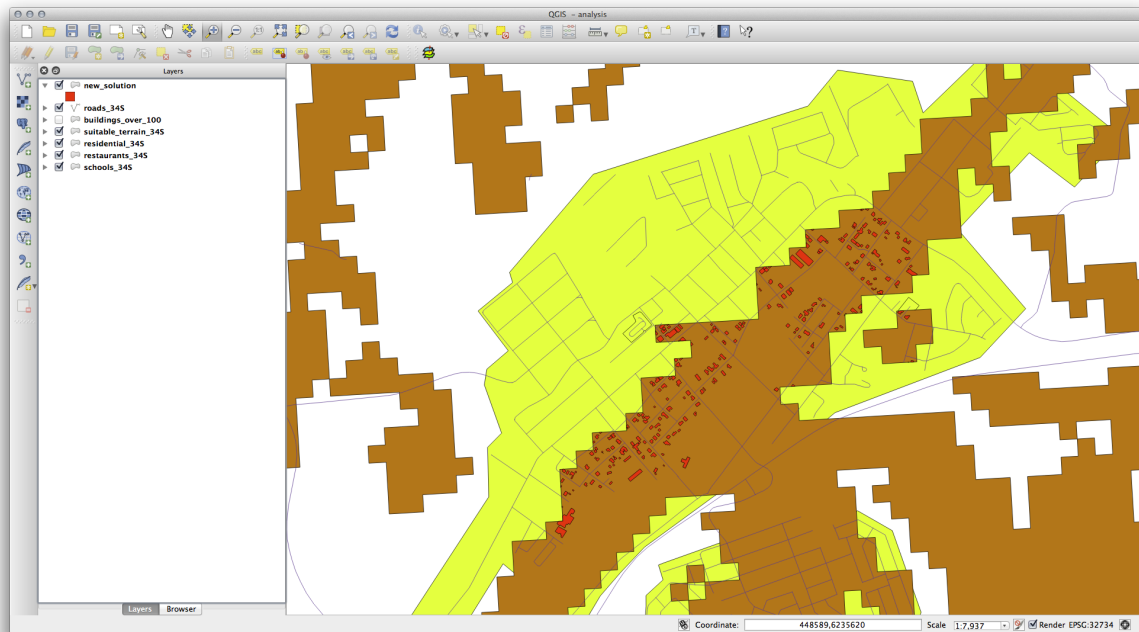
18.11.2 Inspecting the Results

You may notice that some of the buildings in your `new_solution` layer have been “sliced” by the *Intersect* 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

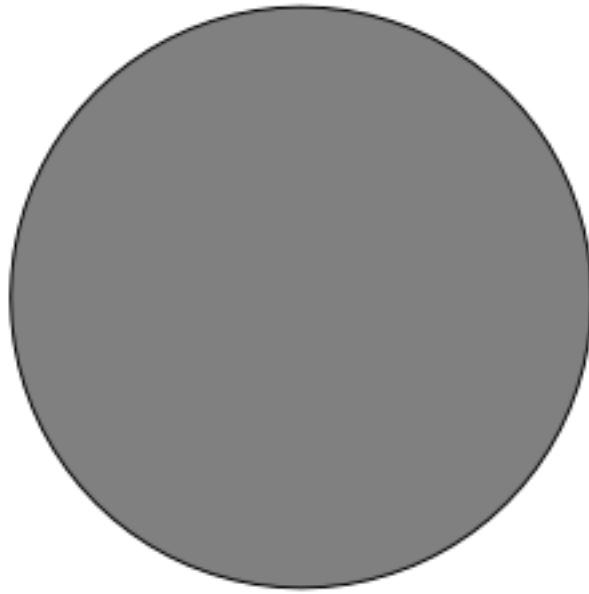
Back to text

18.11.3 Refining the Analysis

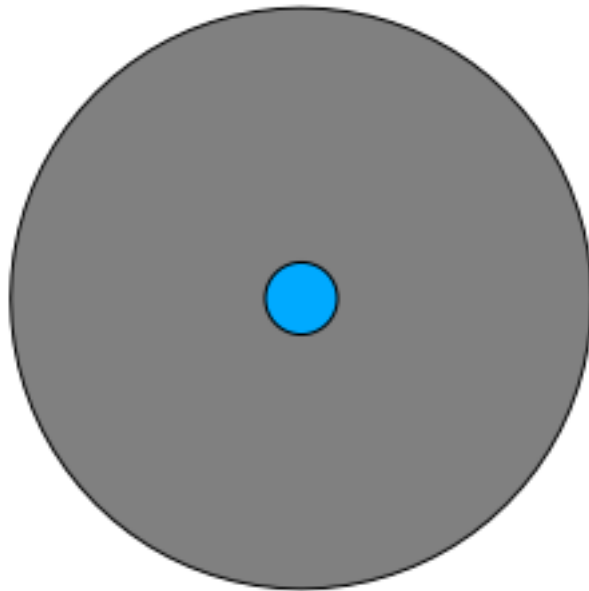
At the moment, your analysis should look something like this:



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.

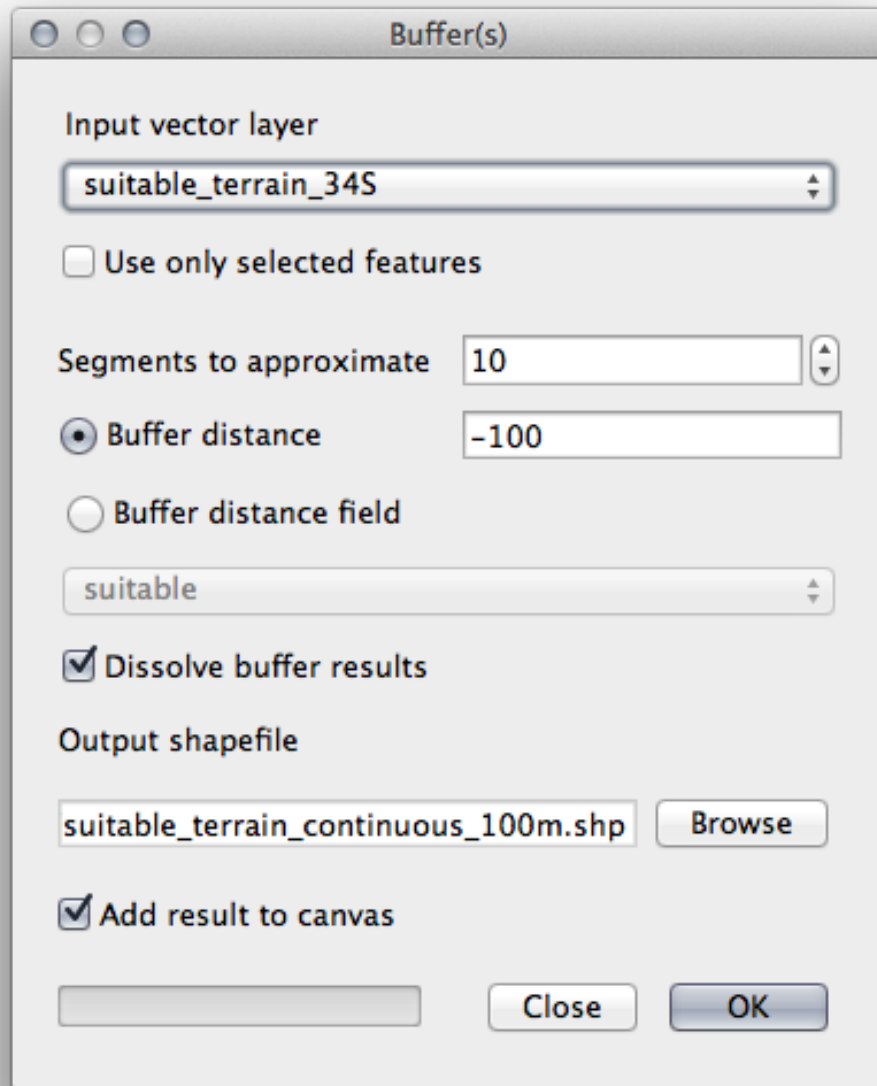


Therefore, you can run an *interior buffer* of 100 meters on your existing *suitable_terrain* vector layer. In the output of the buffer function, whatever remains of the original layer will represent areas where there is suitable terrain for 100 meters beyond.

To demonstrate:

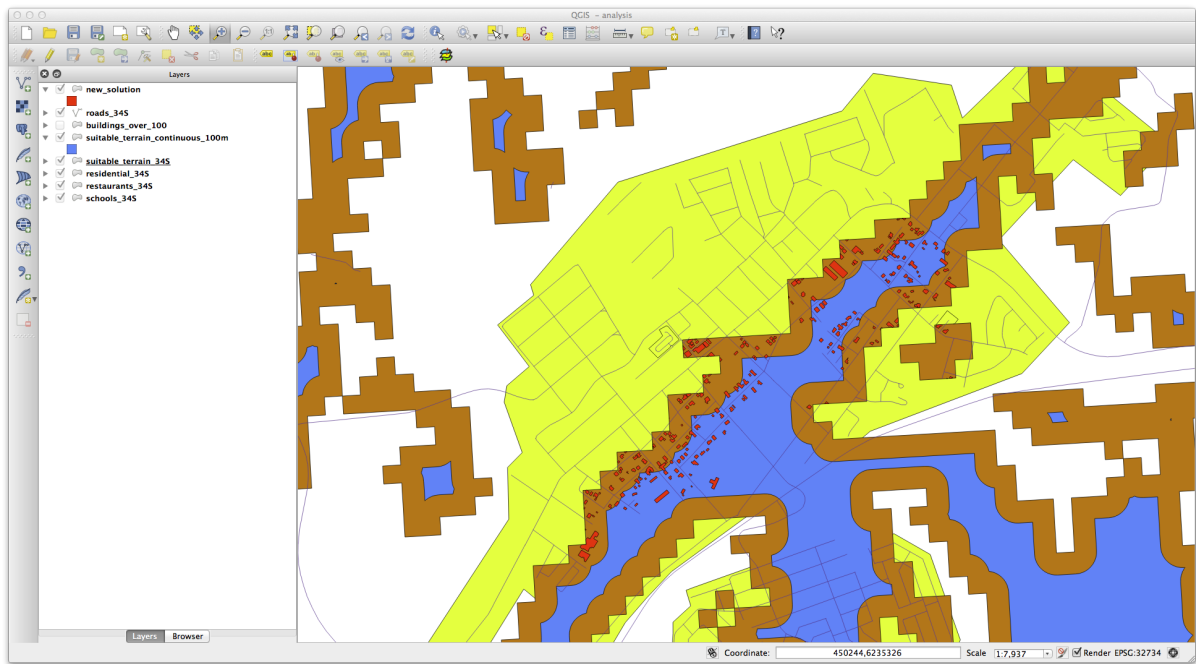
- Go to *Vector* → *Geoprocessing Tools* → *Buffer(s)* to open the Buffer(s) dialog.

- Set it up like this:

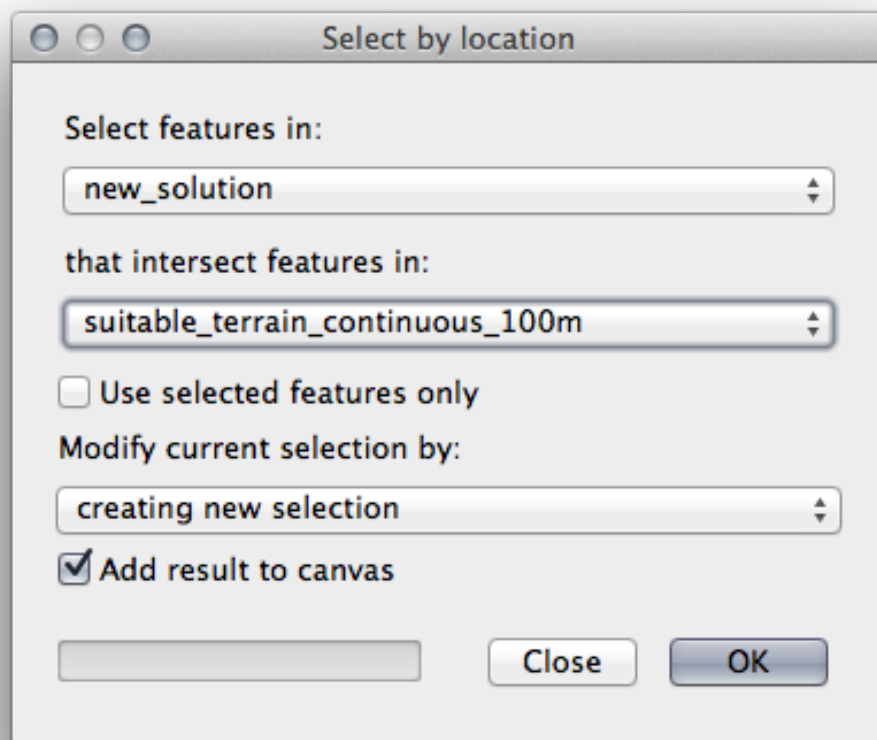


- Use the *suitable_terrain* layer with 10 segments and a buffer distance of -100. (The distance is automatically in meters because your map is using a projected CRS.)
- Save the output in `exercise_data/residential_development/` as `suitable_terrain_continuous100m.shp`.
- If necessary, move the new layer above your original *suitable_terrain* layer.

Your results will look like something like this:

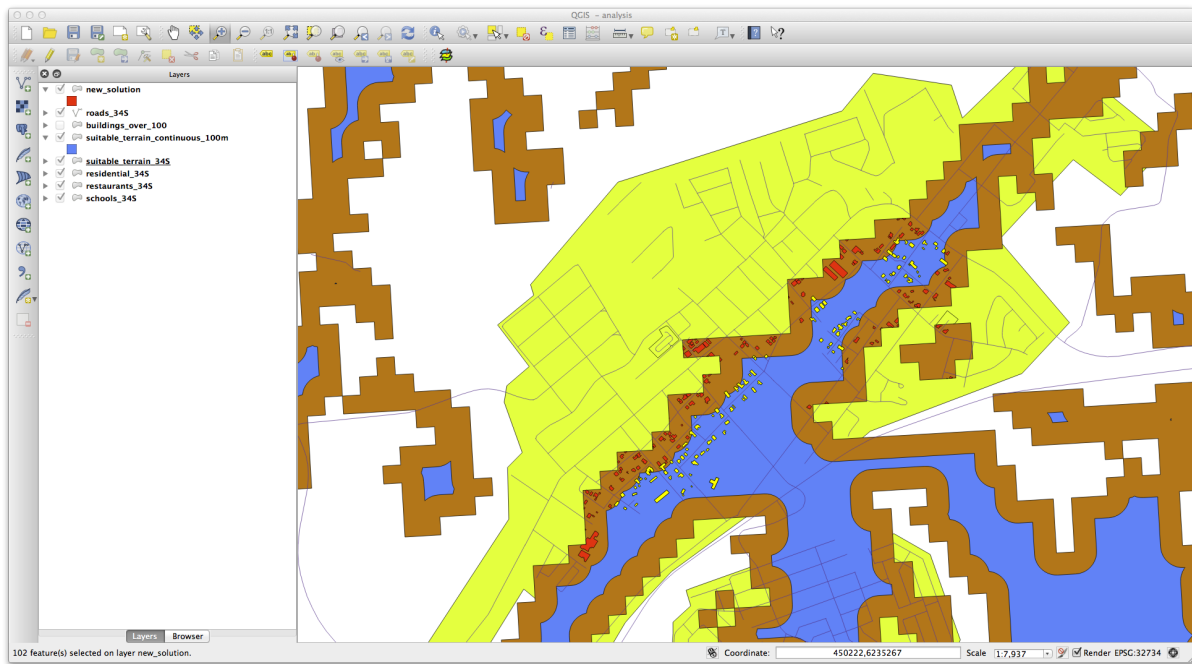


- Now use the *Select by Location* tool (*Vector* → *Research Tools* → *Select by location*).
- Set up like this:



- 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.

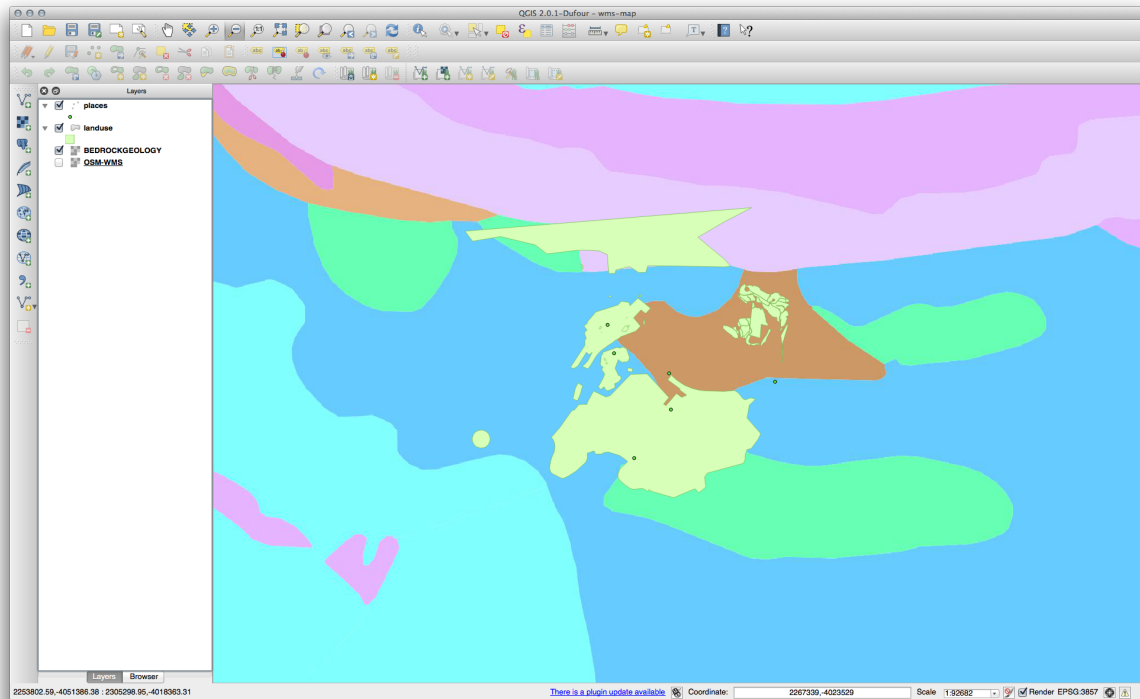
- Save the selection under `exercise_data/residential_development/` as `final_answer.shp`.

Back to text

18.12 Results For WMS

18.12.1 Adding Another WMS Layer

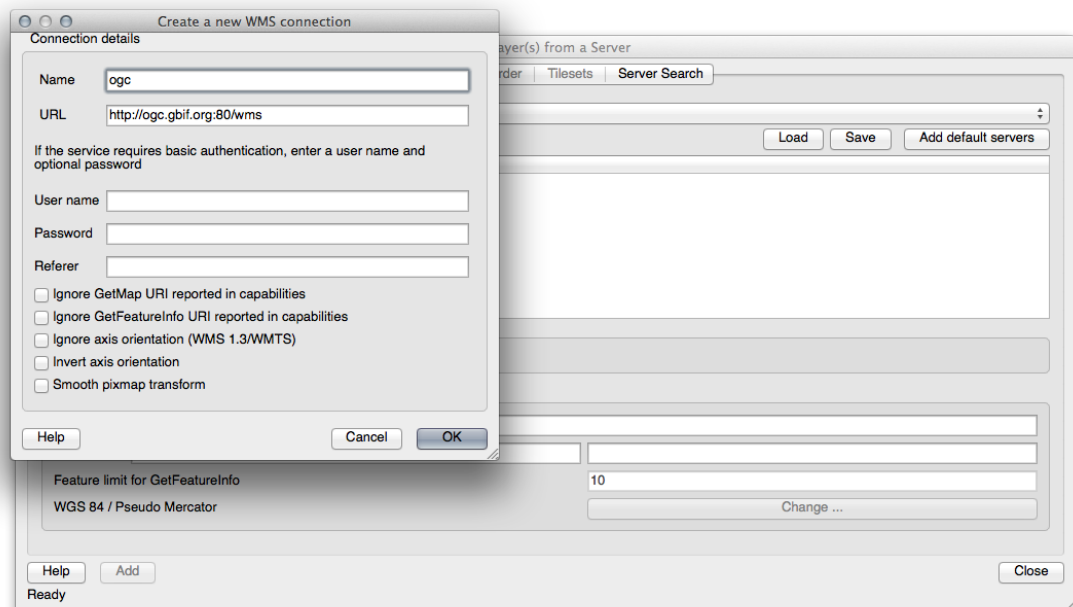
Your map should look like this (you may need to re-order the layers):

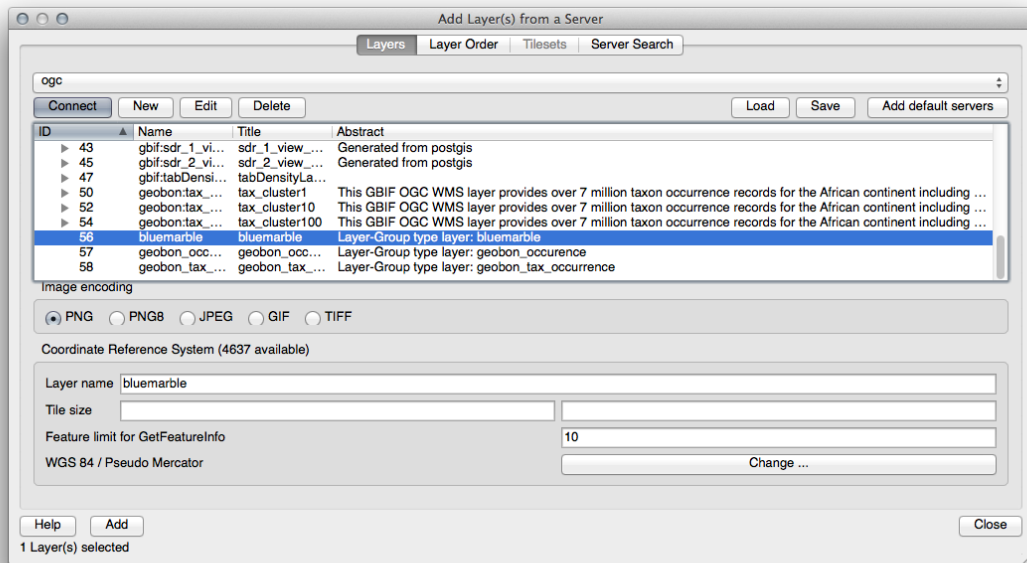


Back to text

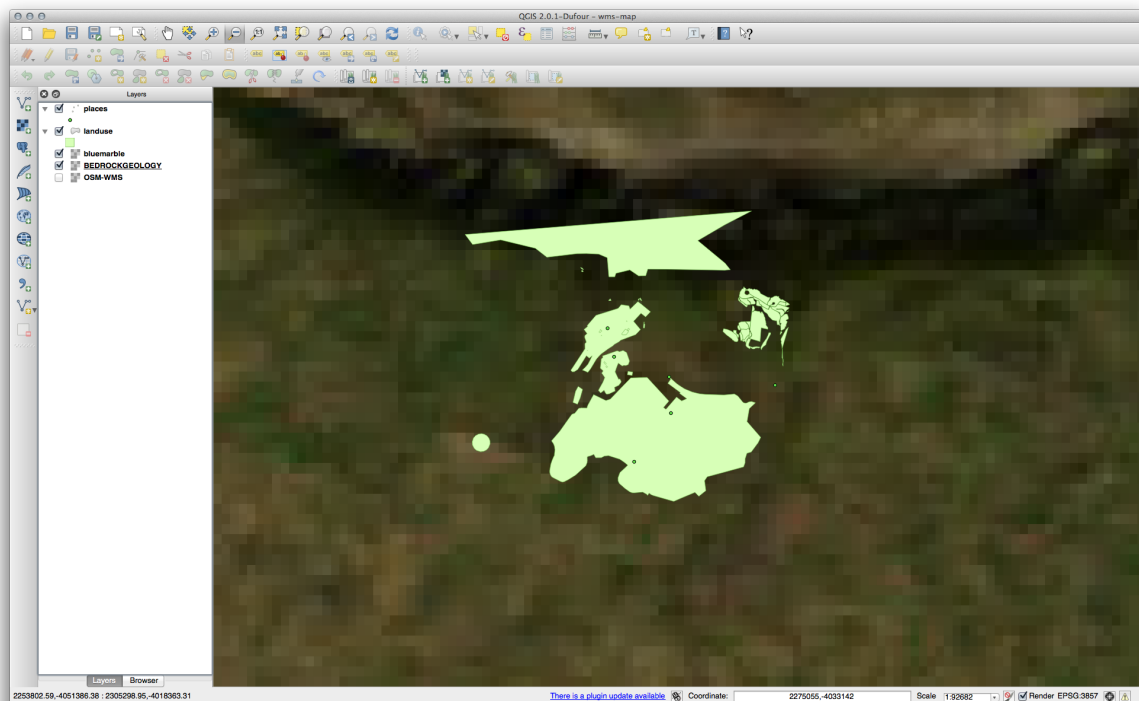
18.12.2 Adding a New WMS Server

- Use the same approach as before to add the new server and the appropriate layer as hosted on that server:





- 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.

Back to text

18.12.3 Finding a WMS Server

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 *World Mineral Deposits* WMS

at http://apps1.gdr.nrcan.gc.ca/cgi-bin/worldmin_en-ca_ows. 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.

[Back to text](#)

18.13 Results For *Database Concepts*

18.13.1 *Address Table Properties*

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
```

[Back to text](#)

18.13.2 *Normalising the People Table*

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 |

Bemerkung: 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.

[Back to text](#)

18.13.3 *Further Normalisation of the People Table*

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.

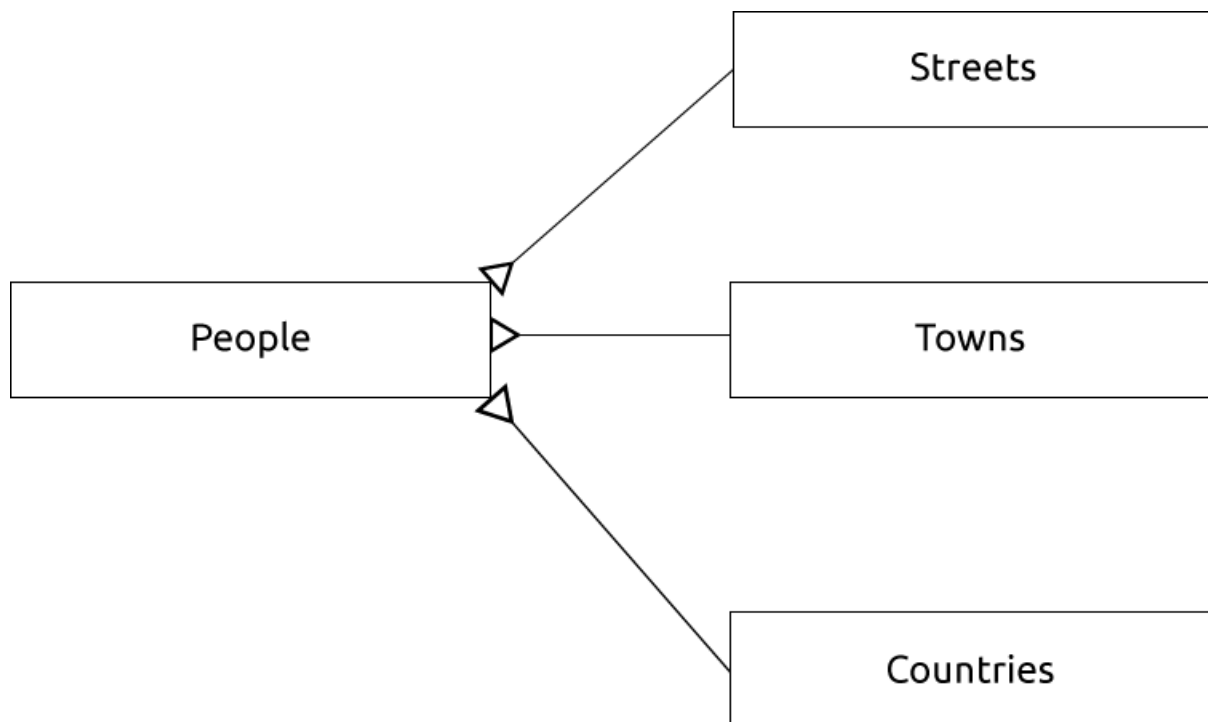
One way to further normalise the table is to split the name field into *first_name* and *last_name*:

| 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:



[Back to text](#)

18.13.4 Create a People Table

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) | | |

Bemerkung: For illustration purposes, we have purposely omitted the fkey constraint.

[Back to text](#)

18.13.5 The DROP Command

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.

Bemerkung: 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!

[Back to text](#)

18.13.6 Insert a New Street

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');
```

[Back to text](#)

18.13.7 Add a New Person With Foreign Key Relationship

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.

Bemerkung: If you have already added a new street object, you might find

that the new Main Road has an ID of 3 not 2.

[Back to text](#)

18.13.8 Return Street Names

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)
```

Bemerkung: 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).

[Back to text](#)

18.14 Results For Spatial Queries

18.14.1 The Units Used in Spatial Queries

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.

[Back to text](#)

18.14.2 Creating a Spatial Index

```
CREATE INDEX cities_geo_idx
ON cities
USING gist (the_geom);
```

[Back to text](#)

18.15 Results For Geometry Construction

18.15.1 Creating Linestrings

```
alter table streets add column the_geom geometry;
alter table streets add constraint streets_geom_point_chk check
    (st_geometrytype(the_geom) = 'ST_LineString'::text OR the_geom IS NULL);
insert into geometry_columns values ('','public','streets','the_geom',2,4326,
    'LINESTRING');
create index streets_geo_idx
    on streets
    using gist
    (the_geom);
```

[Back to text](#)

18.15.2 **Linking Tables**

```
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, the_geom)
    values ('Faulty Towers',
        34,
        3,
        '072 812 31 28',
        1,
        'SRID=4326;POINT(33 33)');

insert into people (name,house_no, street_id, phone_no, city_id, the_geom)
    values ('IP Knightly',
        32,
        1,
        '071 812 31 28',
        1,
        'SRID=4326;POINT(32 -34)');

insert into people (name,house_no, street_id, phone_no, city_id, the_geom)
    values ('Rusty Bedsprings',
        39,
        1,
        '071 822 31 28',
        1,
        'SRID=4326;POINT(34 -34)');
```

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.

[Back to text](#)

18.16 Results For *Simple Feature Model*

18.16.1 *Populating Tables*

```
create table cities (id serial not null primary key,
                    name varchar(50),
                    the_geom geometry not null);
alter table cities
add constraint cities_geom_point_chk
check (st_geometrytype(the_geom) = 'ST_Polygon'::text );
```

[Back to text](#)

18.16.2 *Populate the Geometry_Columns Table*

```
insert into geometry_columns values
('','public','cities','the_geom',2,4326,'POLYGON');
```

[Back to text](#)

18.16.3 *Adding Geometry*

```
select people.name,
       streets.name as street_name,
       st_astext(people.the_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.

[Back to text](#)

Indices and tables

- *genindex*
- *modindex*
- *search*