Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Subscribe to my mailing list

Get occasional emails when I post new material or announce training events.

email address

Subscribe

Change Language

English ▼

QGIS Tutorials and Tips

- **Overview**
  - *Introduction* (docs/introduction.html)

- **Basic GIS operations**
  - *Making a Map (QGIS3)* (docs/3/making_a_map.html)
  - *Working with Attributes (QGIS3)* (docs/3/working_with_attributes.html)
  - *Importing Spreadsheets or CSV files (QGIS3)* (docs/3/importing_spreadsheets_csv.html)
  - *Basic Vector Styling (QGIS3)* (docs/3/basic_vector_styling.html)
  - *Calculating Line Lengths and Statistics (QGIS3)* (docs/3/calculating_line_lengths.html)
  - *Basic Raster Styling and Analysis (QGIS3)* (docs/3/raster_styling_and_analysis.html)
  - *Raster Mosaicing and Clipping (QGIS3)* (docs/3/raster_mosaicing_and_clipping.html)
  - *Working with Terrain Data* (docs/working_with_terrain.html)
  - *Working with WMS Data* (docs/working_with_wms.html)
  - *Working with Projections* (docs/working_with_projections.html)
  - *Georeferencing Topo Sheets and Scanned Maps (QGIS3)* (docs/3/georeferencing_basics.html)
  - *Georeferencing Aerial Imagery (QGIS3)* (docs/3/advanced_georeferencing.html)
  - *Digitizing Map Data* (docs/digitizing_basics.html)
  - *Searching and Downloading OpenStreetMap Data (QGIS3)* (docs/3/downloading_osm_data.html)

- Intermediate GIS operations
  - *Performing Table Joins (QGIS3)* (docs/3/performing_table_joins.html)
  - *Performing Spatial Joins (QGIS3)* (docs/3/performing_spatial_joins.html)
  - *Performing Spatial Queries (QGIS3)* (docs/3/performing_spatial_queries.html)
  - *Creating Heatmaps (QGIS3)* (docs/3/creating_heatmaps.html)
  - *Animating Time Series Data (QGIS3)* (docs/3/animating_time_series.html)

- Advanced GIS operations
  - *Nearest Neighbor Analysis (QGIS3)* (docs/3/nearest_neighbor_analysis.html)
  - *Sampling Raster Data using Points or Polygons (QGIS3)* (docs/3/sampling_raster_data.html)
  - *Interpolating Point Data* (docs/interpolating_point_data.html)
  - *Batch Processing using Processing Framework (QGIS3)* (docs/3/batch_processing.html)
  - *Automating Complex Workflows using Processing Modeler (QGIS3)* (docs/3/processing_graphical_modeler.html)
  - *Automating Map Creation with Print Layout Atlas (QGIS3)* (docs/3/automating_map_creation.html)

- Network Analysis
  - *Basic Network Visualization and Routing (QGIS3)* (docs/3/basic_network_analysis.html)
  - *Locating Nearest Facility with Origin-Destination Matrix (QGIS3)* (docs/3/origin_destination_matrix.html)
  - *Service Area Analysis using Openrouteservice (QGIS3)* (docs/3/service_area_analysis.html)

- Python Scripting (PyQGIS)
  - **new!** PyQGIS in a Day - Course Material (https://courses.spatialthoughts.com/pyqgis-in-a-day.html) ↗
  - *Getting Started With Python Programming (QGIS3)* (docs/3/getting_started_with_pyqgis.html)
  - *Running Processing Algorithms via Python (QGIS3)* (docs/3/processing_algorithms_pyqgis.html)
  - *Building a Python Plugin (QGIS3)* (docs/3/building_a_python_plugin.html)
  - *Building a Processing Plugin (QGIS3)* (docs/3/processing_python_plugin.html)
  - *Using Custom Python Expression Functions (QGIS3)* (docs/3/custom_python_functions.html)
  - *Writing Python Scripts for Processing Framework (QGIS3)* (docs/3/processing_python_scripts.html)
  - *Running and Scheduling QGIS Processing Jobs* (docs/running_qgis_jobs.html)
  - *Performing Table Joins (PyQGIS)* (docs/performing_table_joins_pyqgis.html)

- Web Mapping
  - *Web Mapping with QGIS2Web* (docs/web_mapping_with_qgis2web.html)
  - *Creating Basemaps with QTiles* (docs/creating_basemaps_with_qtiles.html)

- Appendix
  - *Using Plugins* (docs/using_plugins.html)
  - *QGIS Learning Resources* (docs/learning_resources.html)
  - *Data Credits* (docs/credits.html)

Want more QGIS Tips and Tricks? See Spatial Thoughts Blog (https://spatialthoughts.com/category/qgis/) ↗.

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Making a Map (QGIS3)

Often one needs to create a map that can be printed or published. QGIS has a powerful tool called Print Layout that allows you to take your GIS layers and package them to create maps.

Overview of the task

The tutorial shows how to create a map of Japan with standard map elements like map inset, grids, north arrow, scale bar and labels.

Other skills you will learn

- How to view and change QGIS Project Variables
- How to use QGIS expressions

Get the data

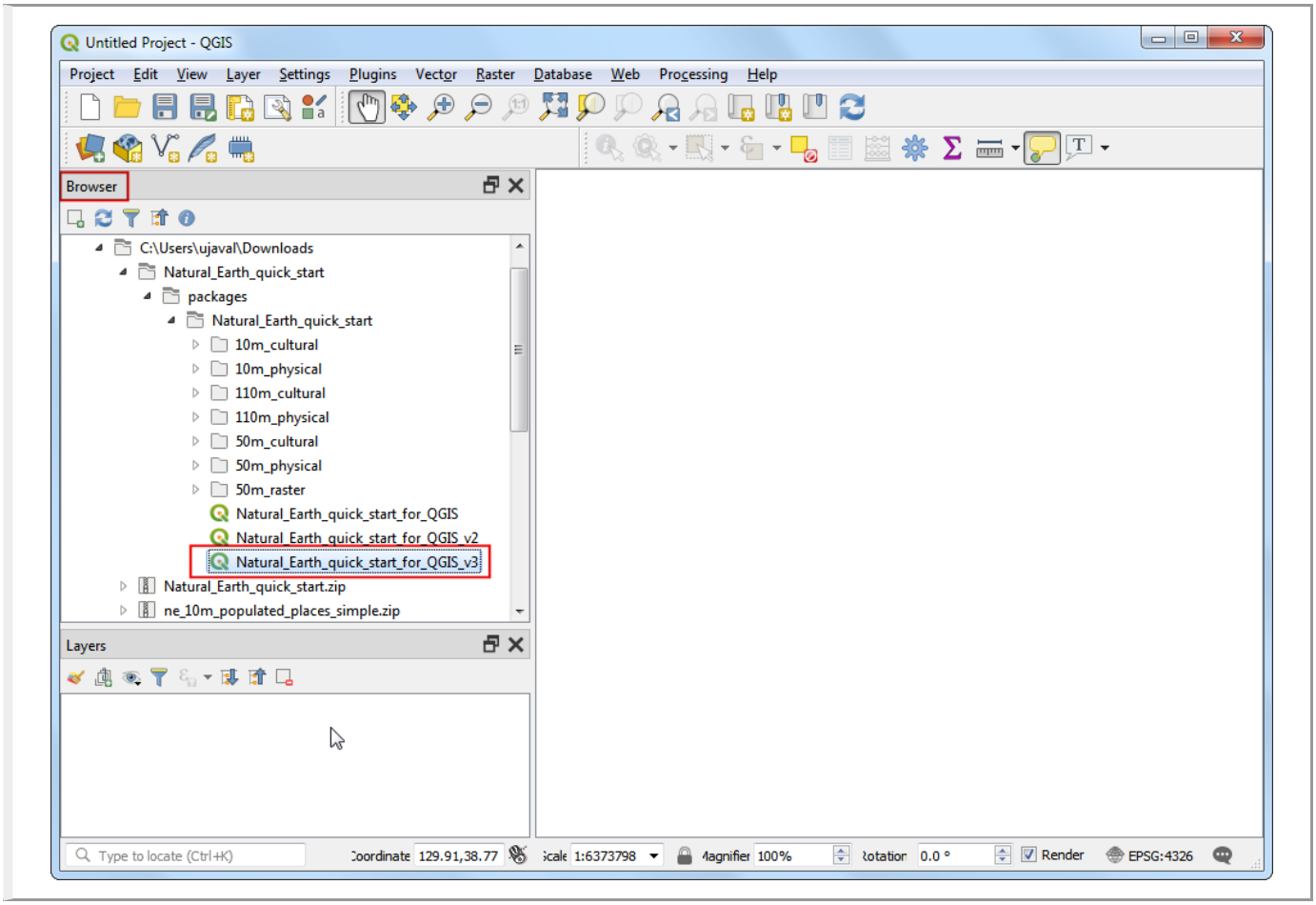
We will use the Natural Earth dataset - specifically the Natural Earth Quick Start Kit that comes with beautifully styled global layers that can be loaded directly to QGIS.

Download the Natural Earth Quickstart Kit (http://naciscdn.org/naturalearth/packages/Natural_Earth_quick_start.zip).

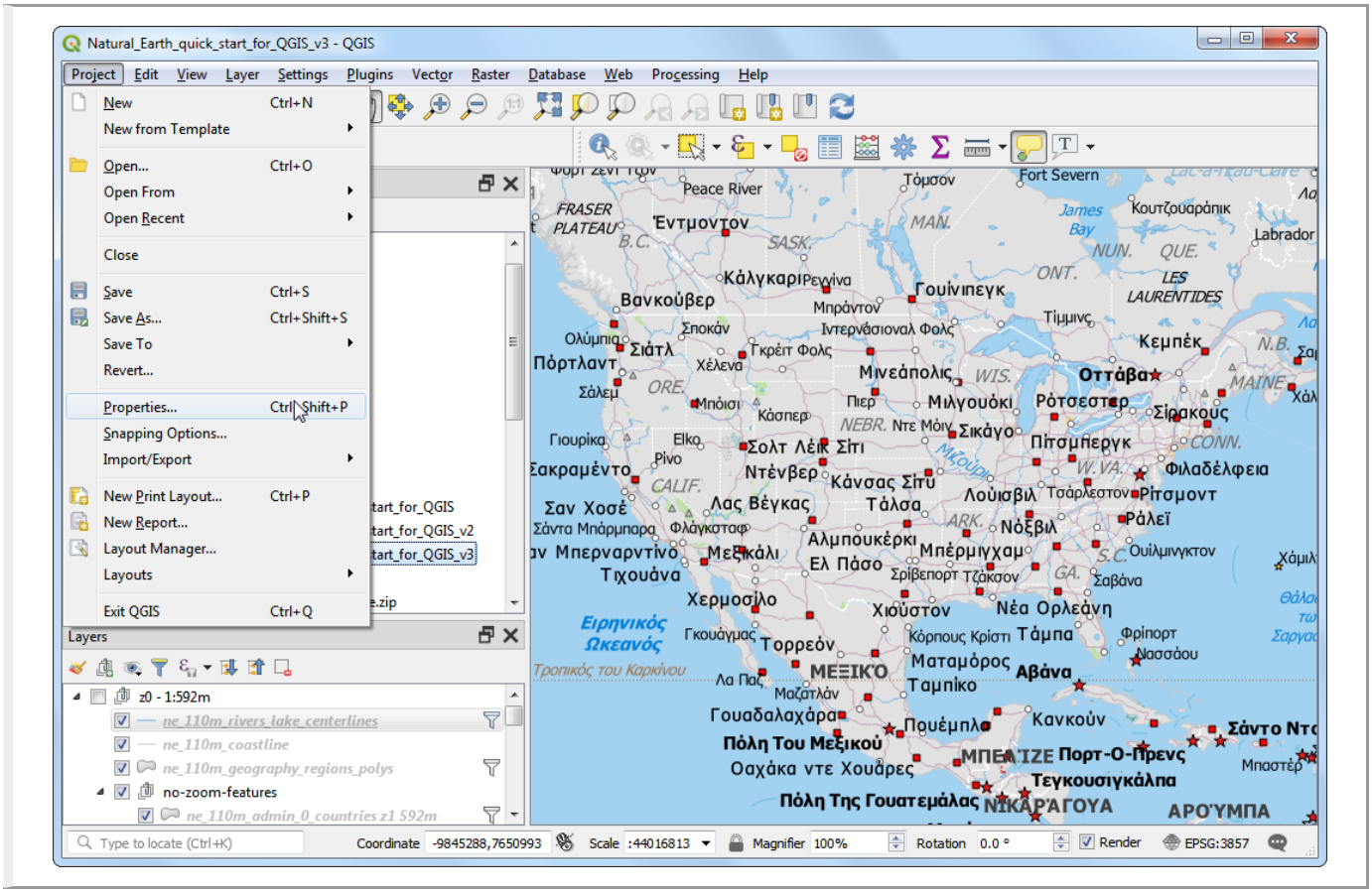
Data Source [NATURALEARTH] (../credits.html#naturalearth)

Procedure

1. Download and extract the Natural Earth Quick Start Kit data. Open QGIS. Locate the `Natural Earth quick start` folder in the Browser panel. Expand the folder to locate the `Natural_Earth_quick_start_for_QGIS_v3` project. This is the project file that contains styled layers in QGIS Document format. Double-click the project to open it.



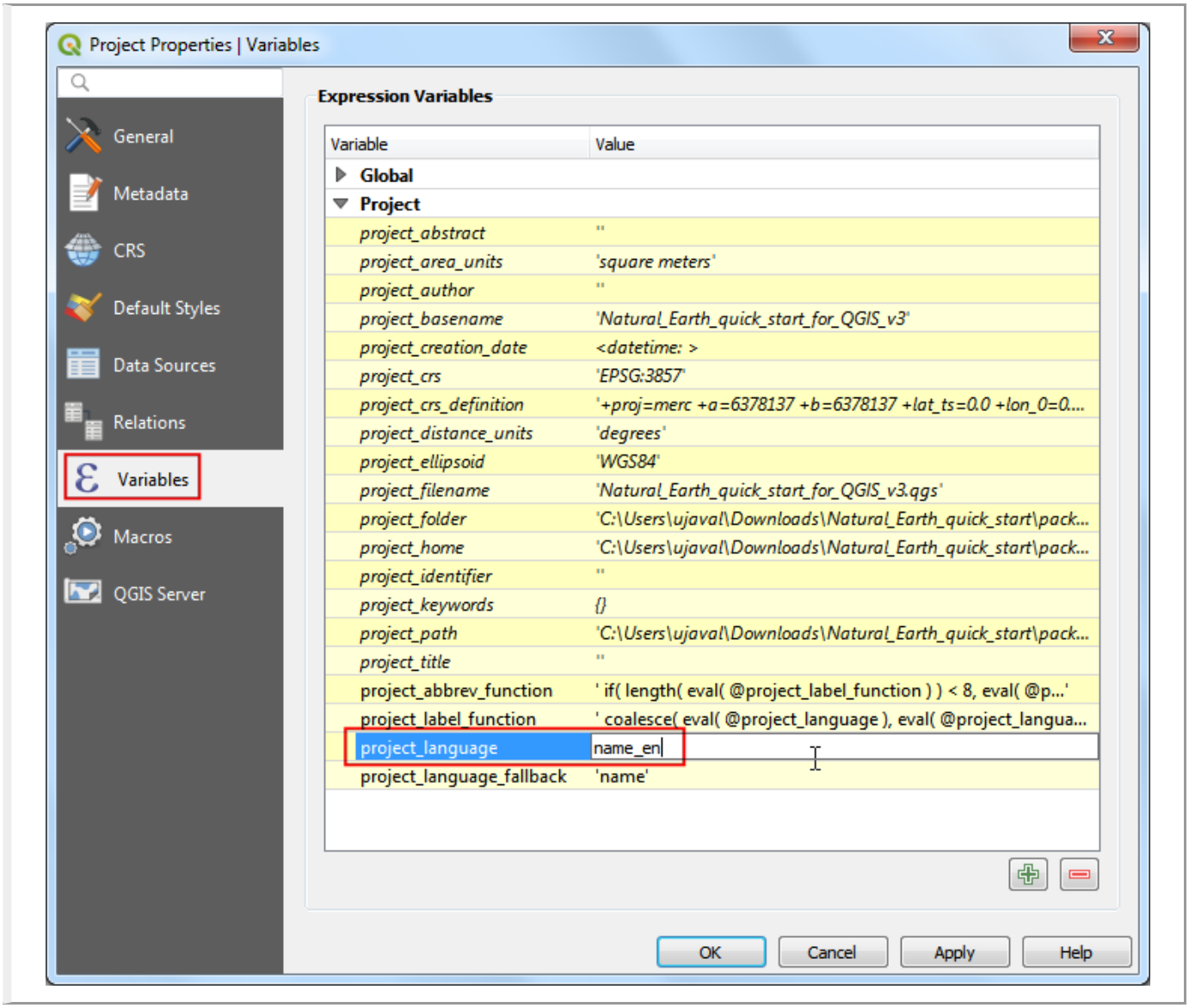
2. You may notice that the map has labels in Greek. This project uses variables to set the language. We can change the variables by going to Project ‣ Properties.



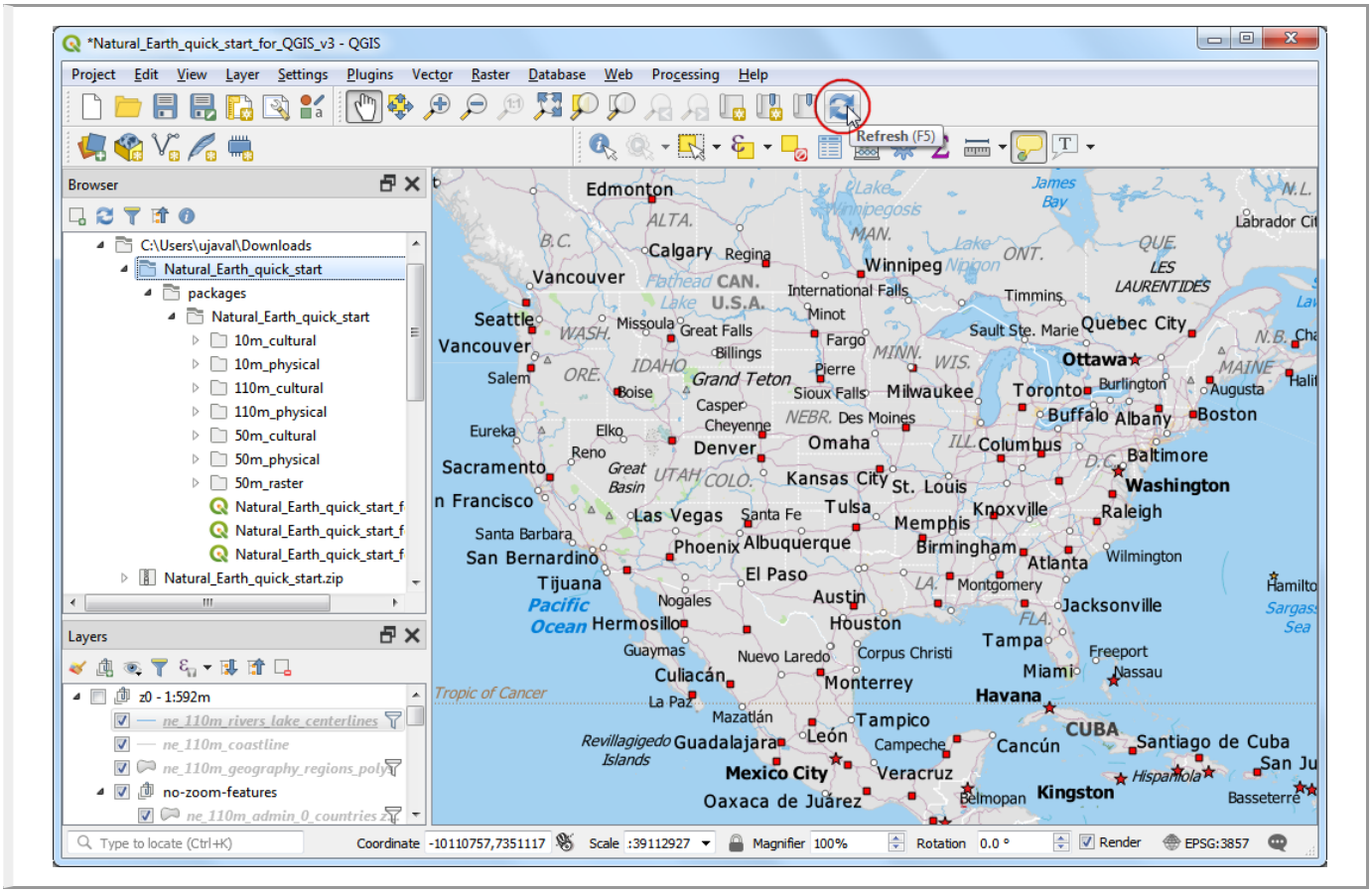
Note

Project variables are a great way to store project-specific values for use anywhere you can use an expression in QGIS. The `Natural_Earth_quick_start_for_QGIS_v3` project comes with many preset variables that are used for styling within that project.

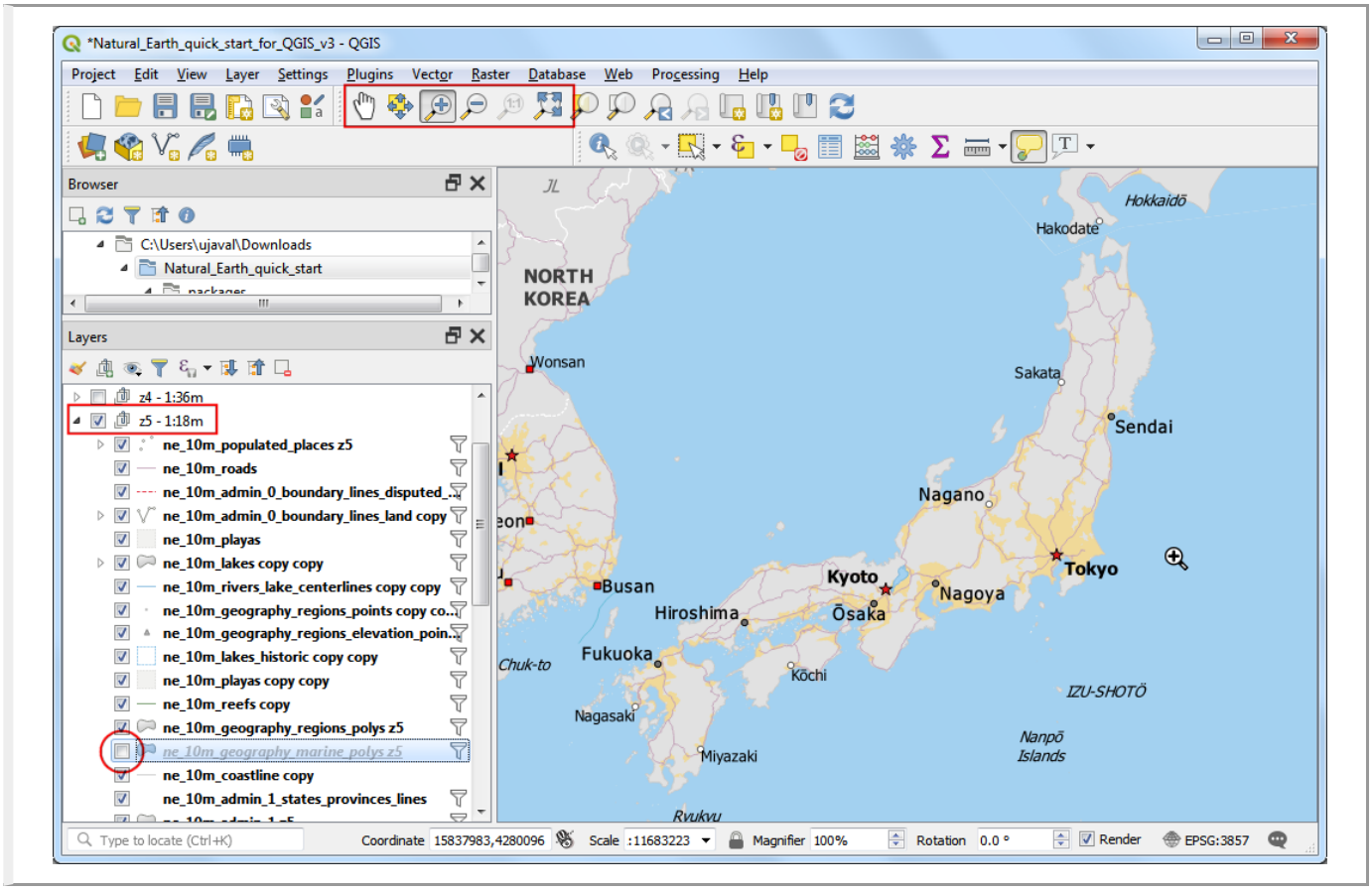
3. Switch to the Variables tab in the Project Properties dialog. Locate the `project_language` variable and click on the Value column to edit it. Change the language to `name_en` and click OK.



4. Back in the main QGIS window, click the Refresh button in the Map Navigation Toolbar. You will now see the map rendered with English labels.

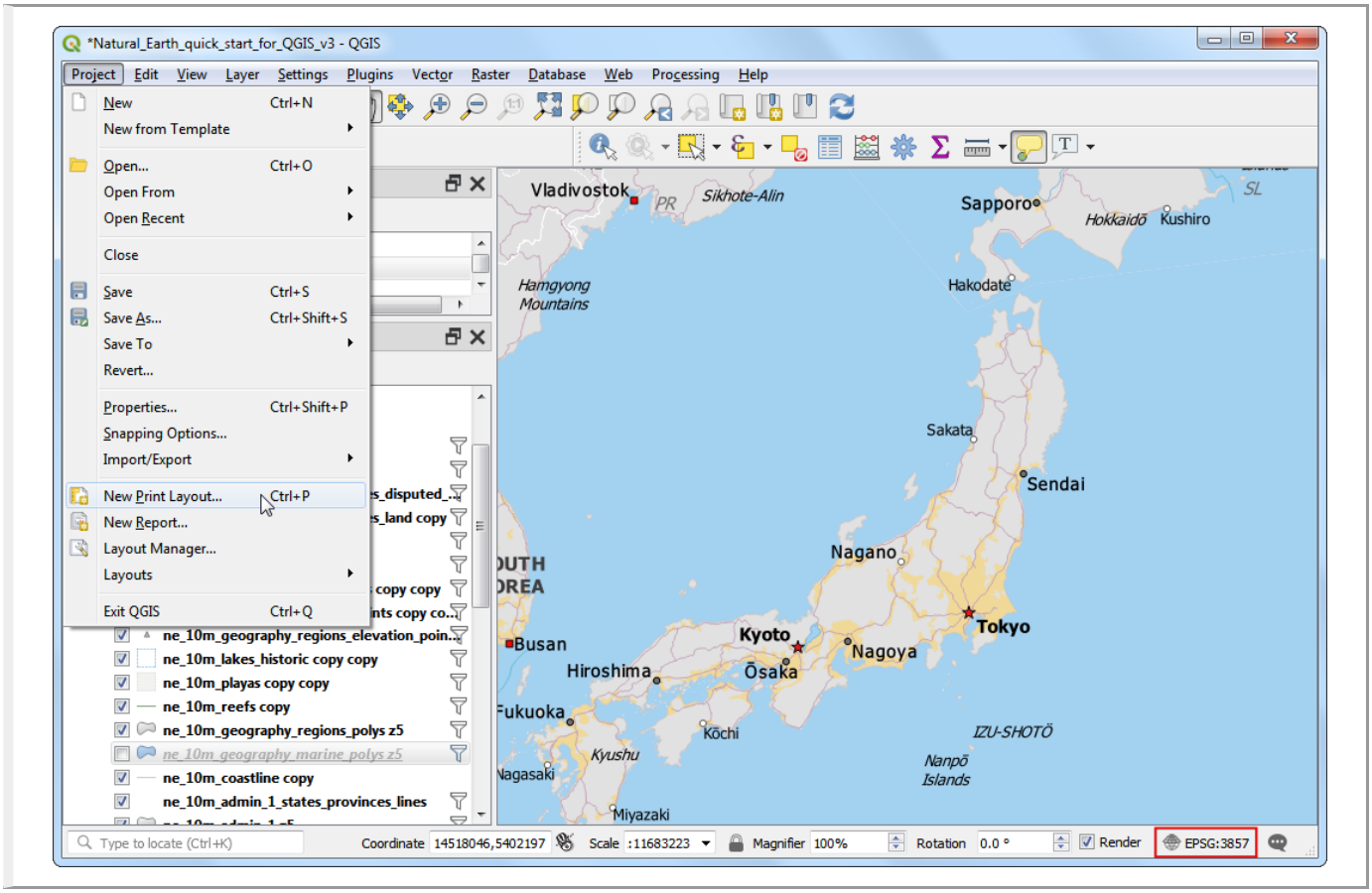


5. Use the pan and zoom controls in the Map Navigation Toolbar and to zoom to Japan.



6. You can turn off some map layers for data that we do not need for this map. Expand the `z5 - 1:18m` folder and uncheck the box next to `ne_10m_geography_marine_polys` and `ne_10m_admin_0_disputed_areas` layers. Before we make a map suitable for printing, we need to choose an appropriate projection. The default CRS for the project is set to `EPSG:3857 Pseudo-`

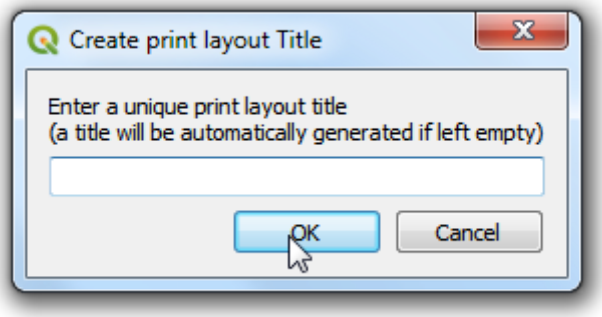
`Mercator` . This is a CRS popularly used for web mapping and is a decent choice for our purpose, so we can leave it to its defalt value. Go to Project ▸ New Print Layout.



Note

For Japan, Japan Plane Rectangular CS is a projected coordinate reference system (CRS) that is designed for minimum distortions. It is divided in 18 zones and if you are working for a smaller region in Japan, using this CRS will be better.

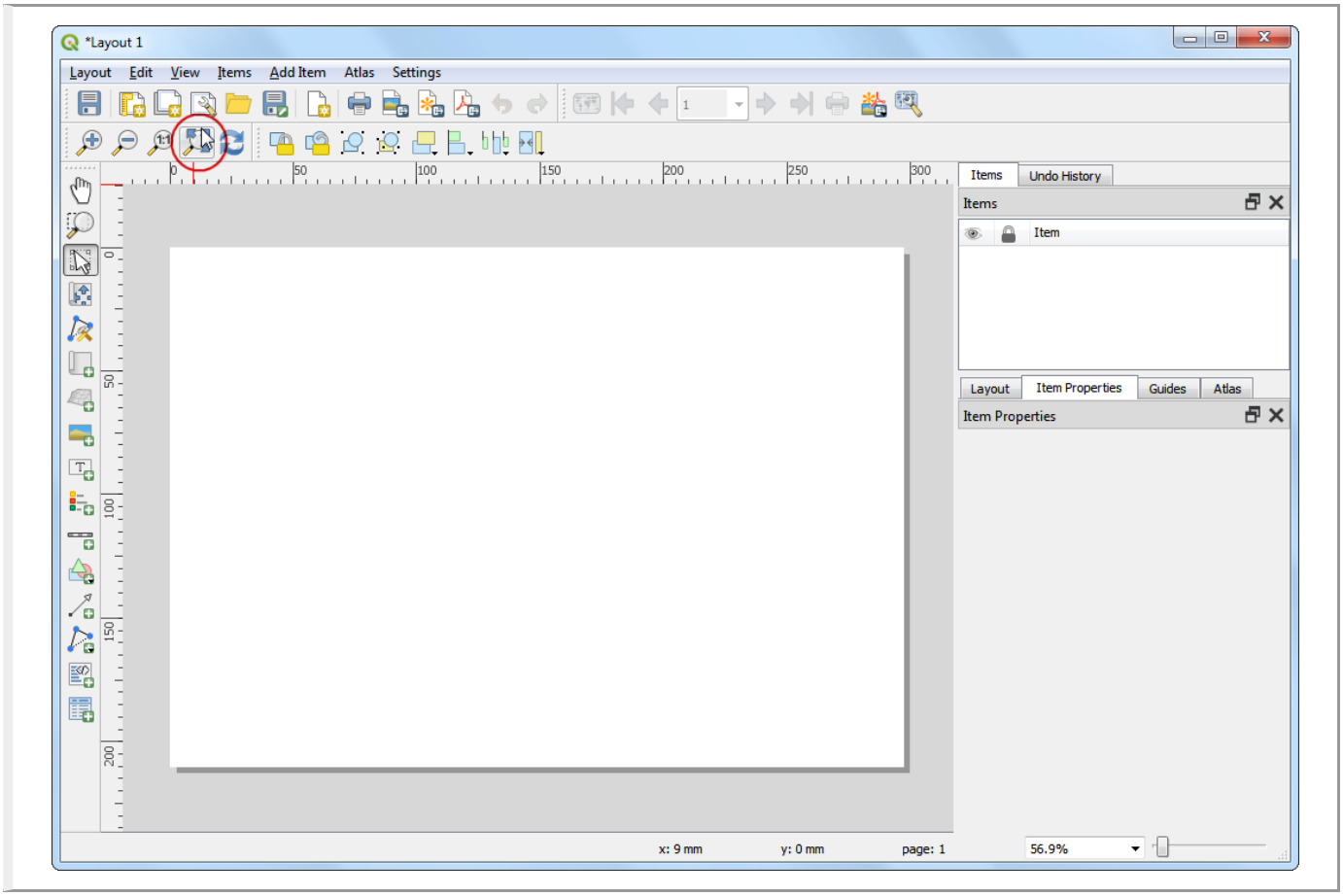
7. You will be prompted to enter a title for the layout. You can leave it empty and click Ok.



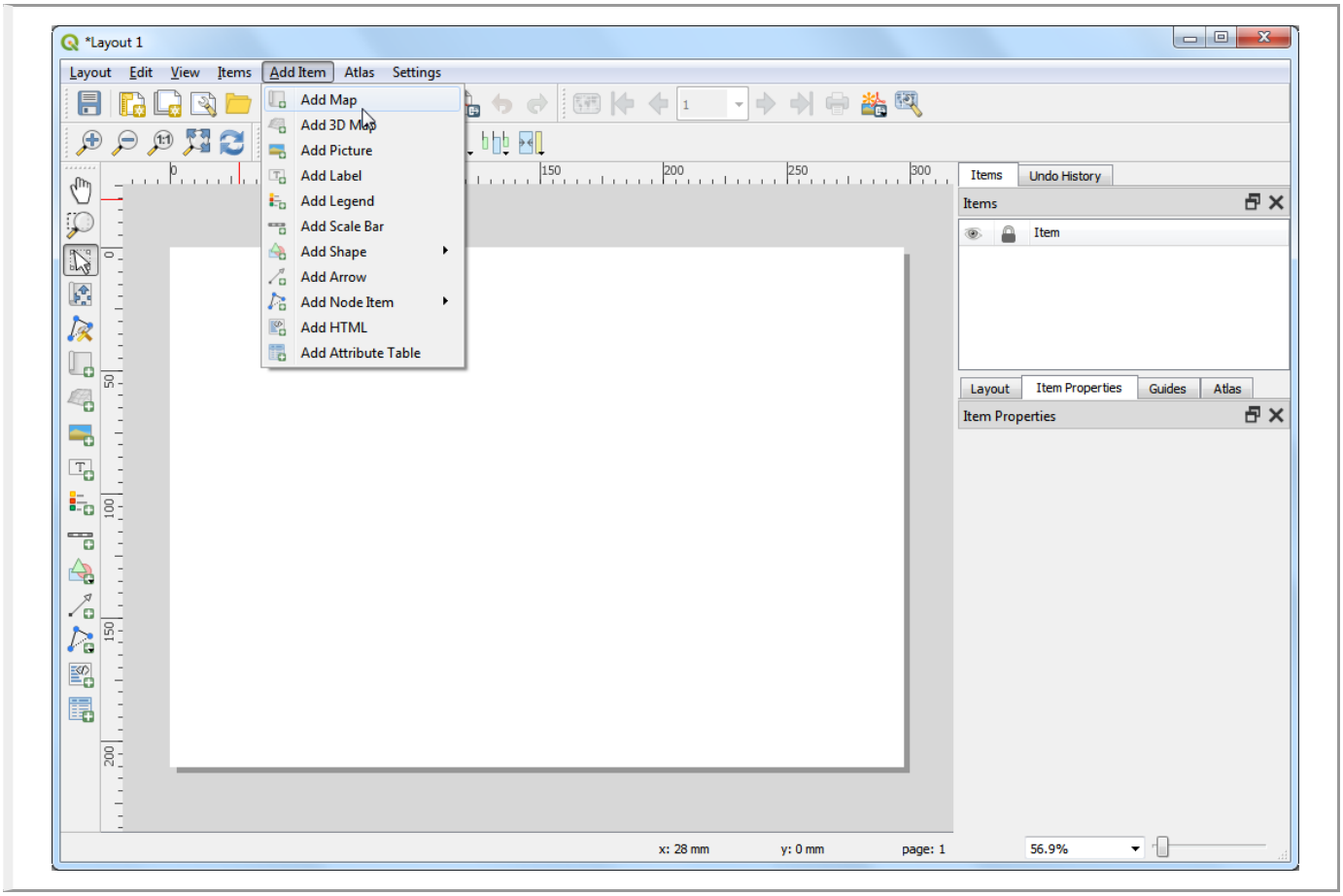
Note

Leaving the layout name empty will assign a default name such as `Layout 1` .

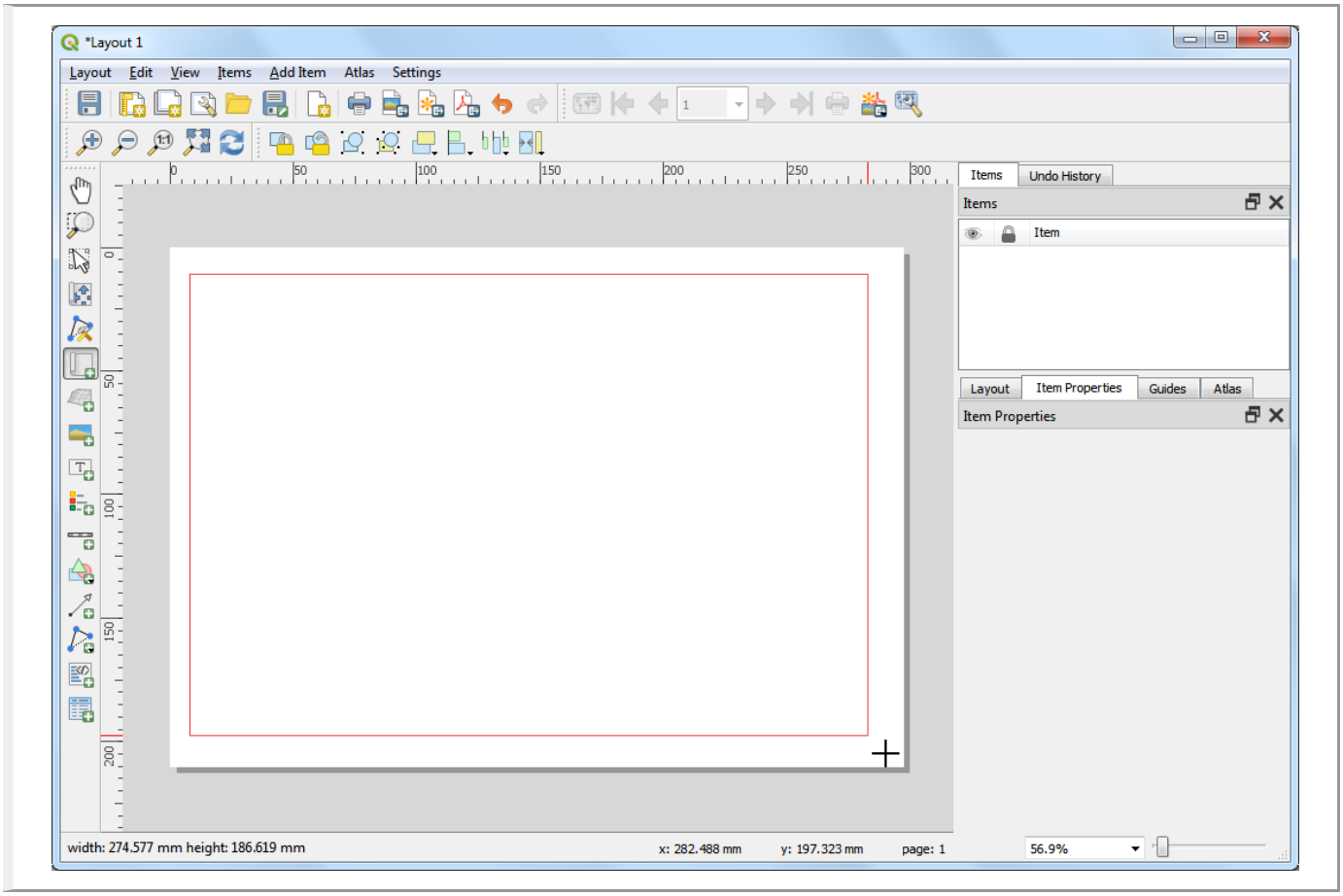
8. In the Print Layout window, click on Zoom full button to display the full extent of the Layout.



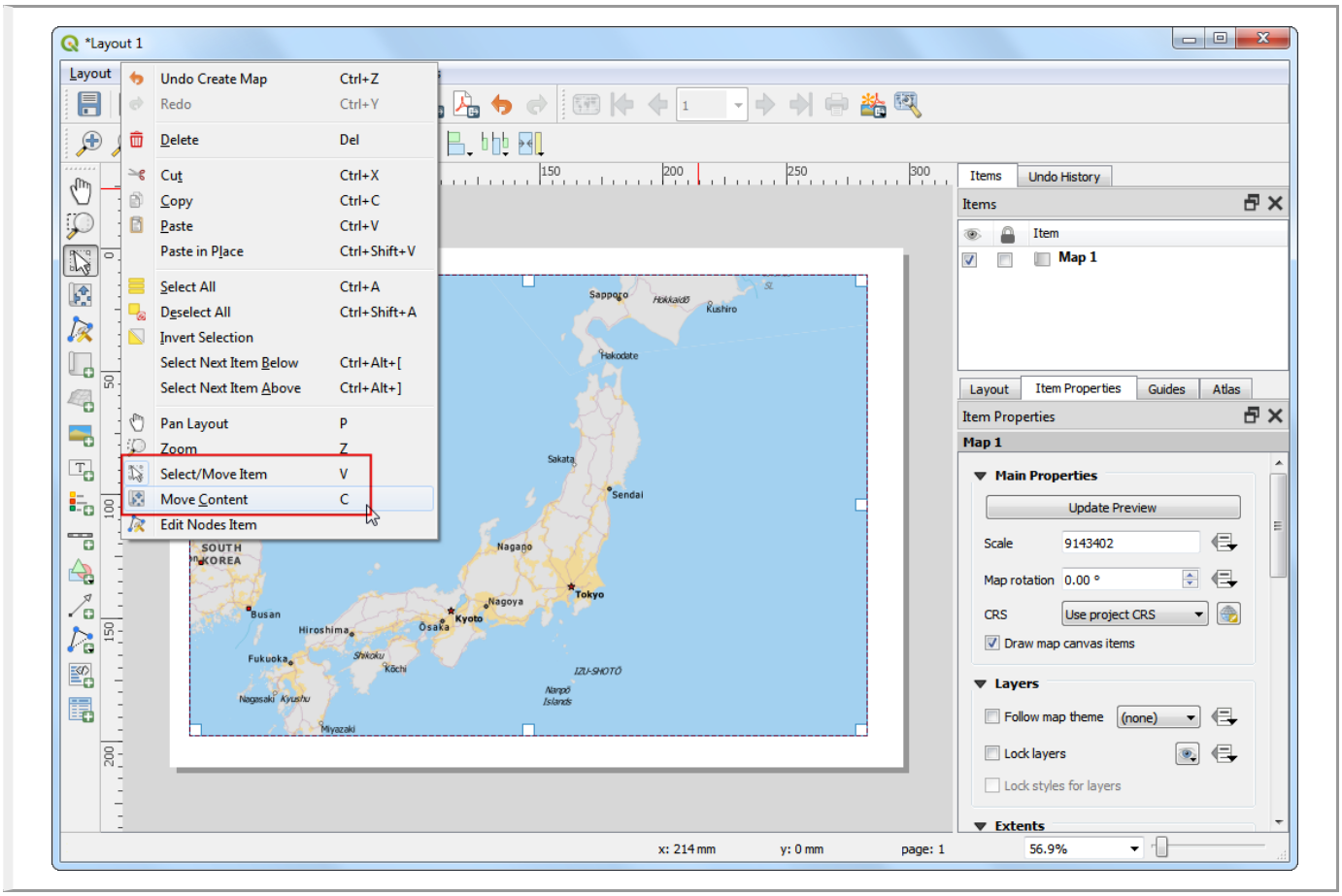
9. Now we would have to bring the map view that we see in the QGIS Canvas to the layout. Go to Add Item ‣ Add Map.



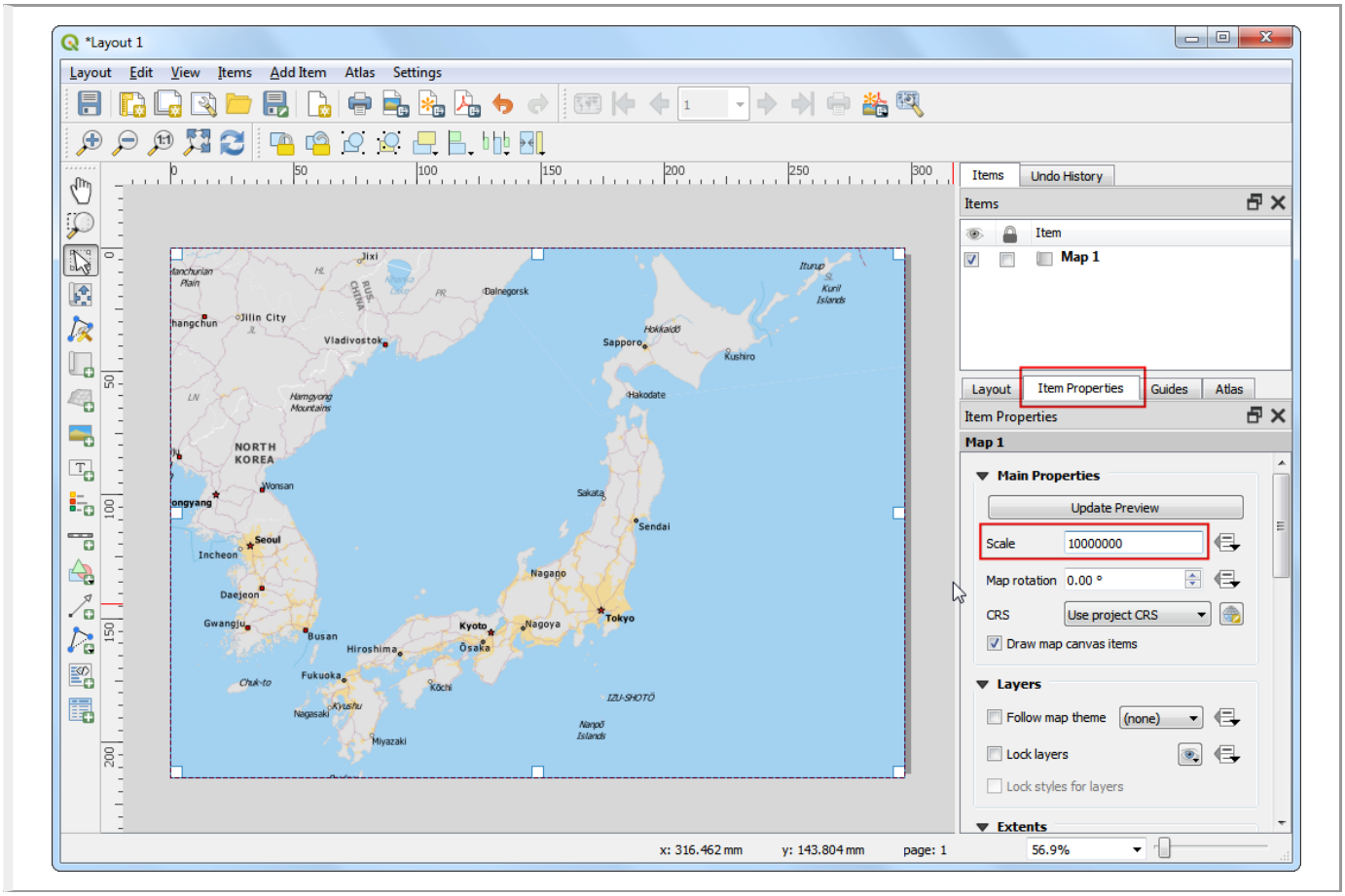
10. Once the Add Map mode is active, hold the left mouse button and drag a rectangle where you want to insert the map.



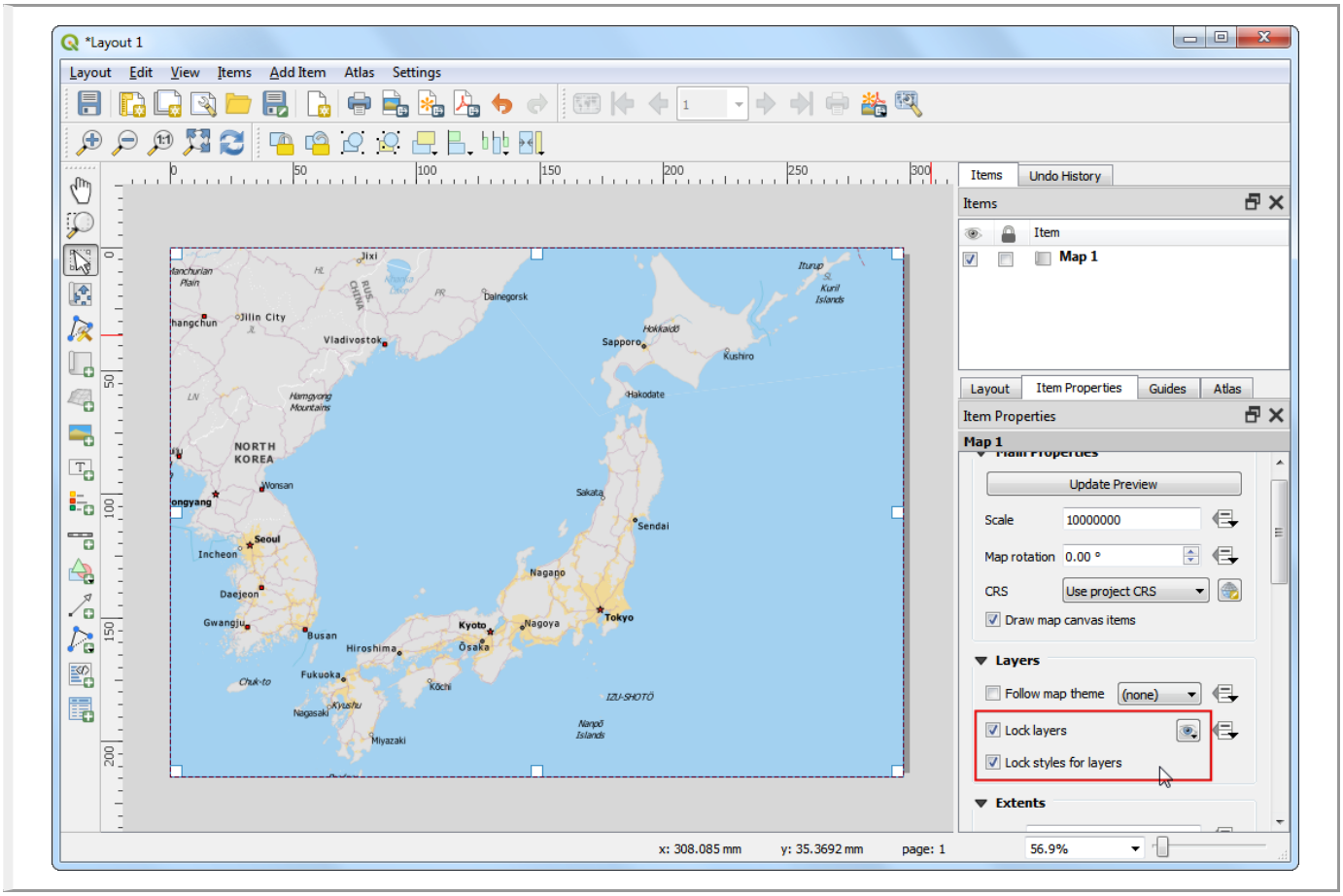
11. You will see that the rectangle window will be rendered with the map from the main QGIS canvas. The rendered map may not be covering the full extent of our interest area. Use Edit ‣ Select/Move item and Edit ‣ Move Content options to pan the map in the window and center it in the composer.



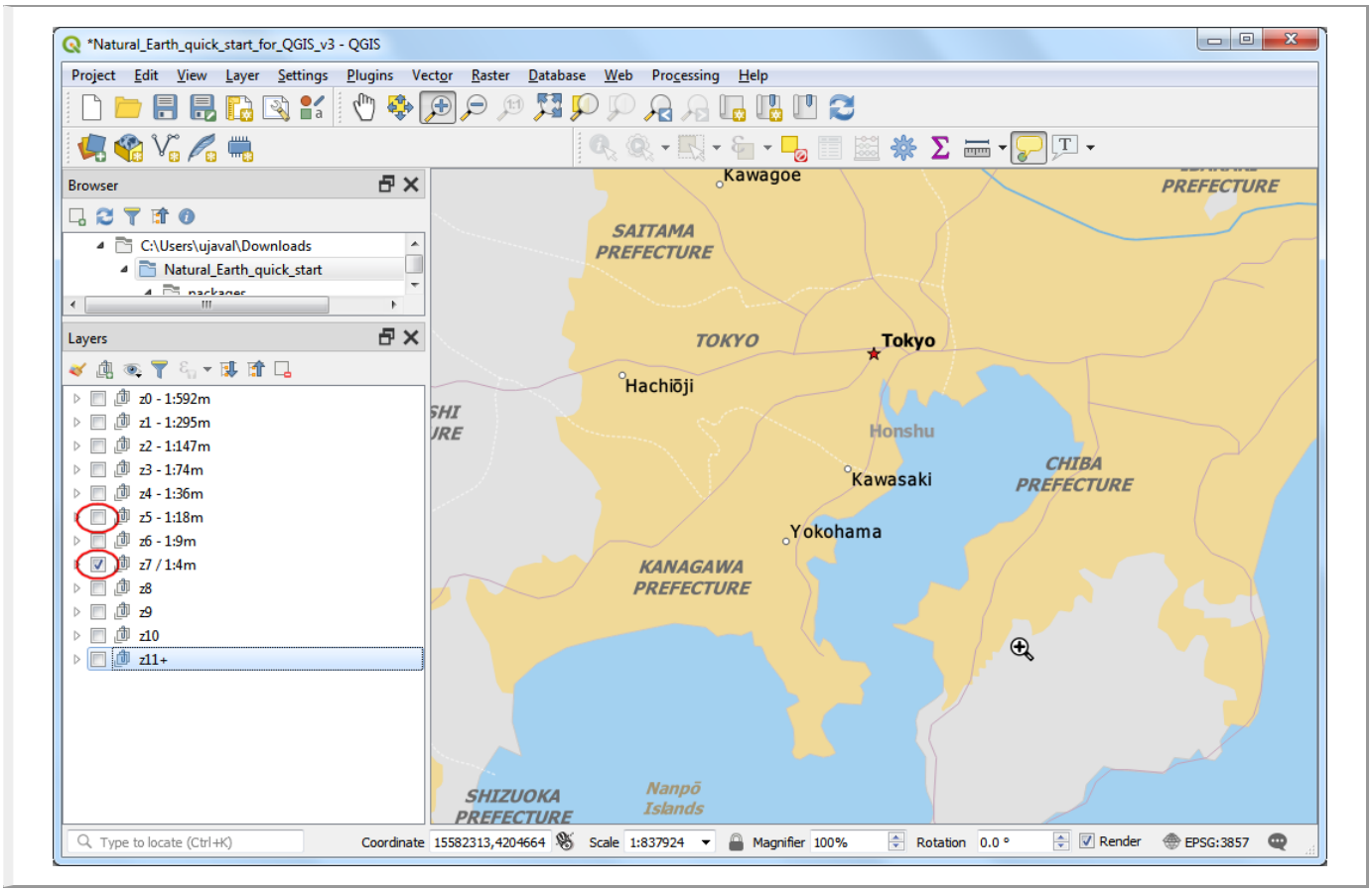
12. Let us also adjust the zoom level for the map. Click on the Item Properties tab and enter `10000000` as the Scale value.



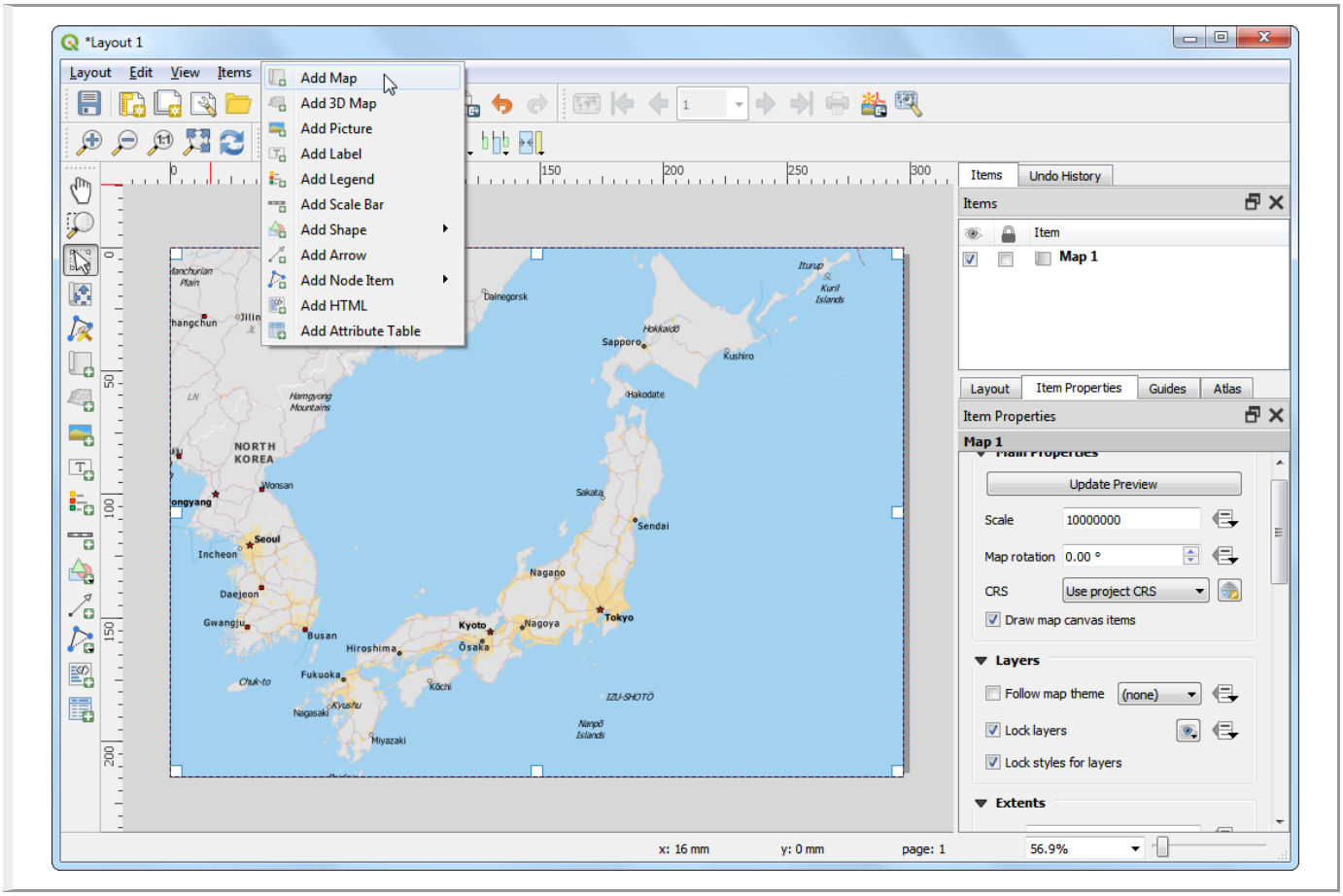
13. Now we will add a map inset that shows a zoomed in view for the Tokyo area. Before we make any changes to the layers in the main QGIS window, check the Lock layers and Lock styles for layers boxes. This will ensure that if we turn off some layers or change their styles, this view will not change.



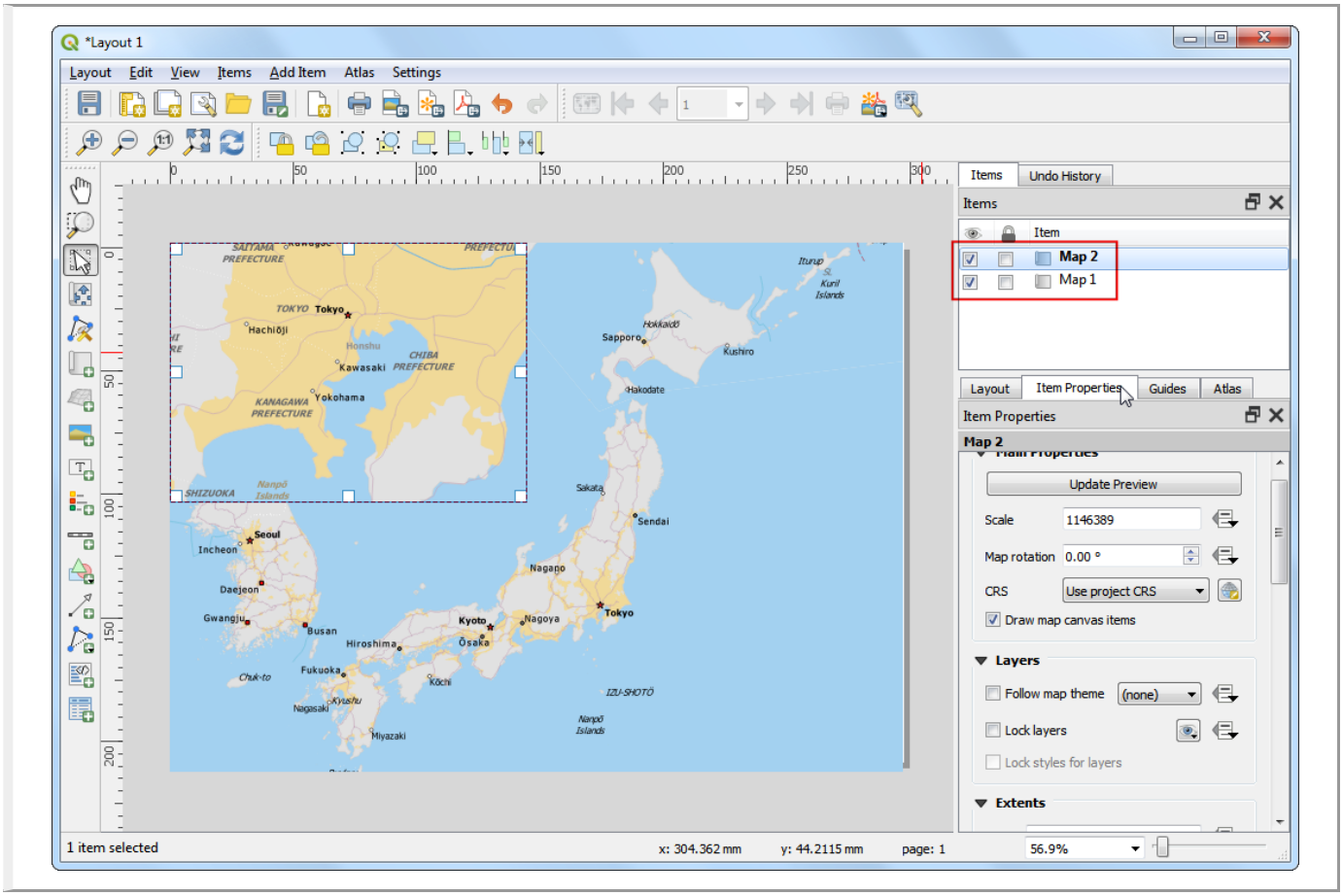
14. Switch to the main QGIS window. Turn off the layer group `z5 - 1:18m` and activate the `z7 - 1: 4m` group. This layer group has styling that is more appropriate for a zoomed-in view. Use the pan and zoom controls in the Map Navigation Toolbar and to zoom around Tokyo.



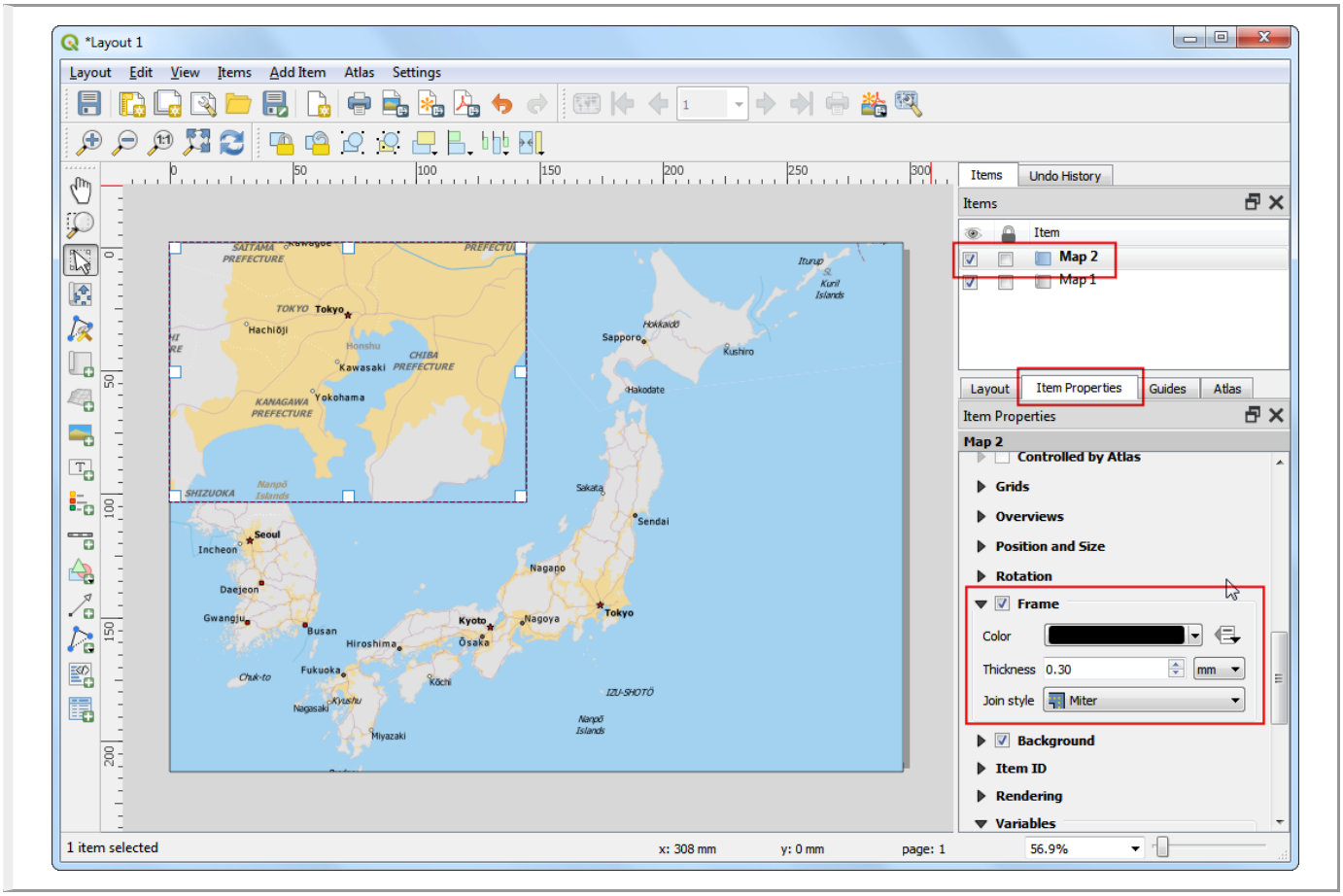
15. We are now ready to add the map inset. Switch the the Print Layout window. Go to Add Item ‣ Add Map.



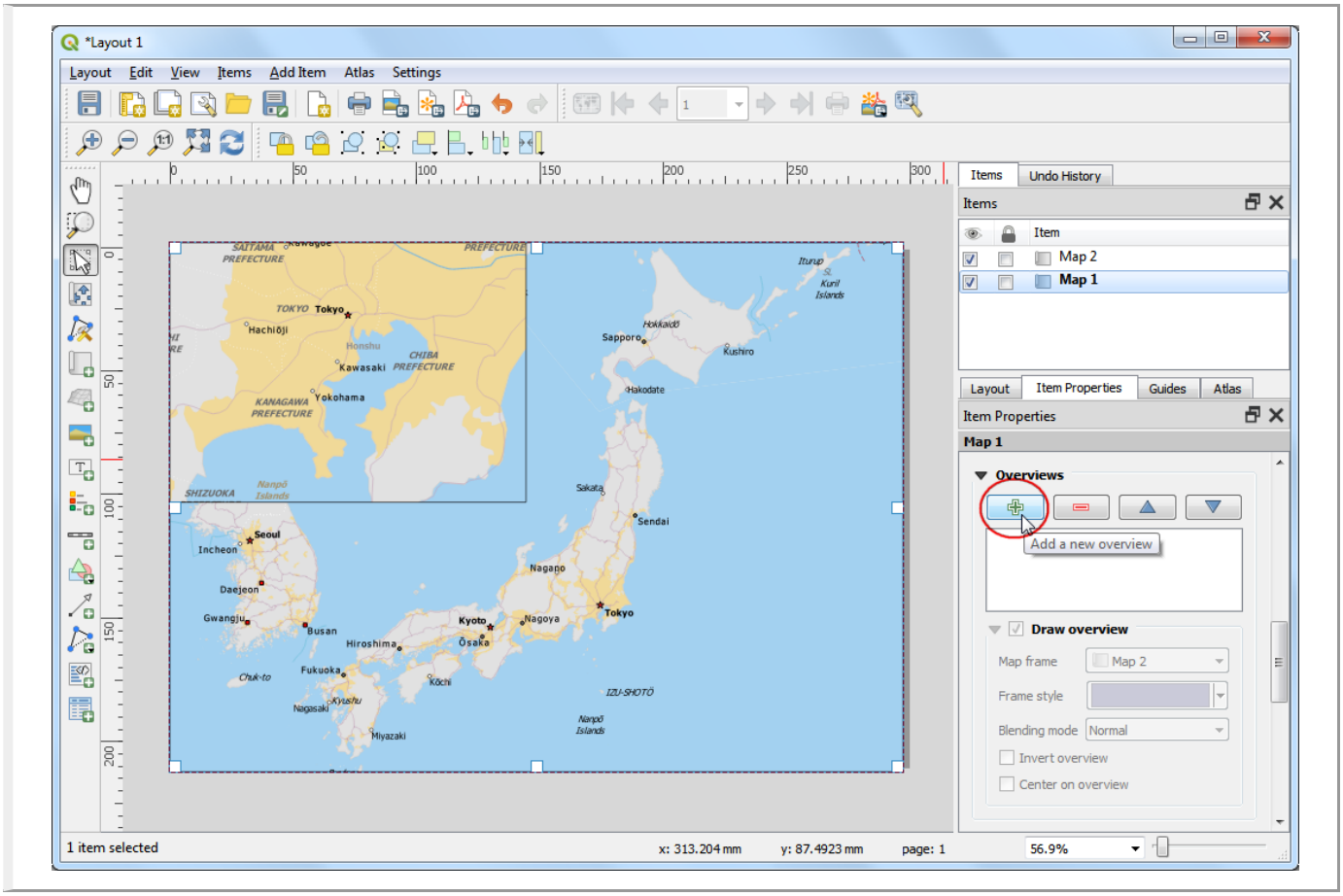
16. Drag a rectangle at the place where you want to add the map inset. You will now notice that we have 2 map objects in the Print Layout. When making changes, make sure you have the correct map selected.



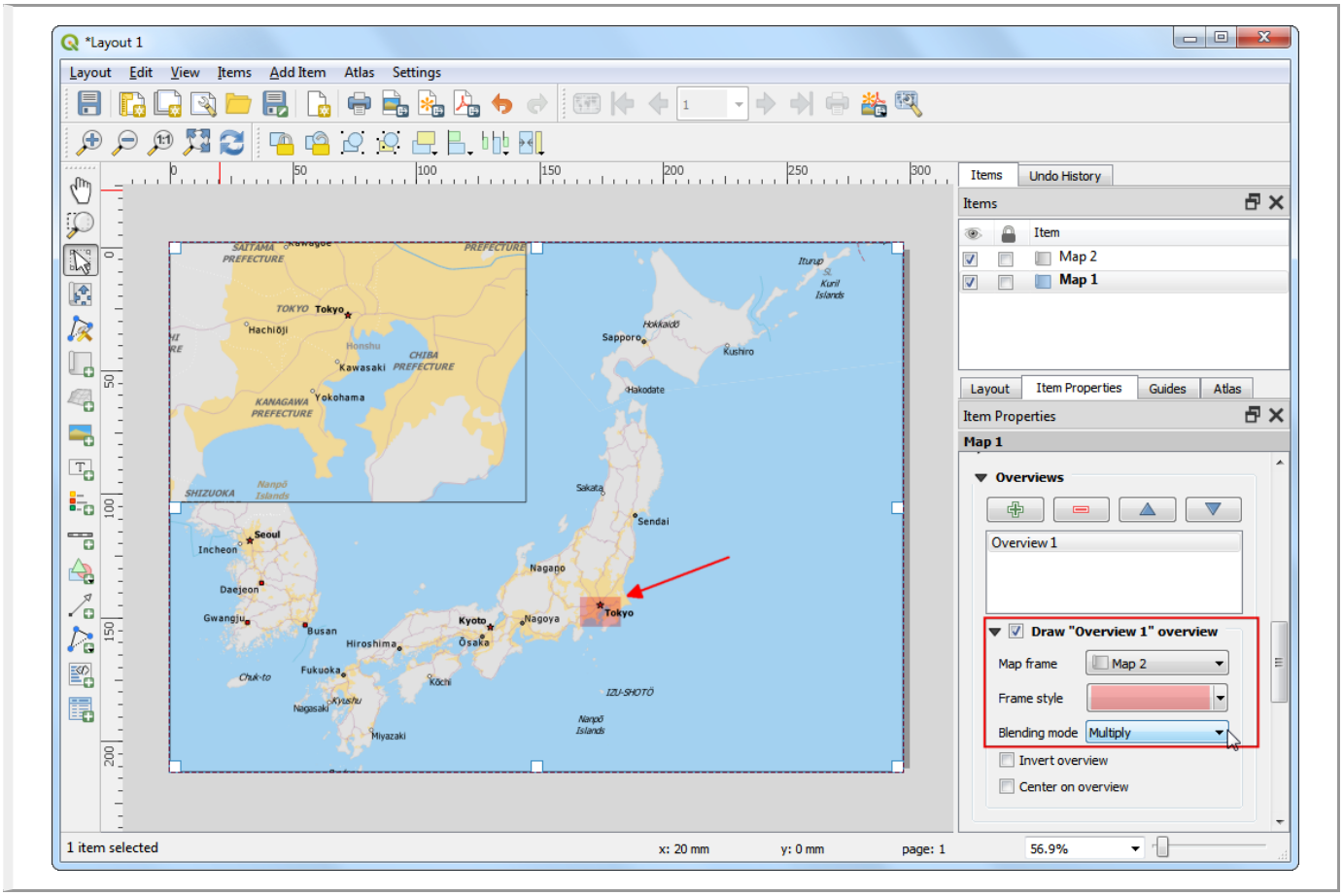
17. Select the `Map 2` object that we just added from the Items panel. Select the Item properties tab. Scroll down to the Frame panel and check the box next to it. You can change the color and thickness of the frame border so it is easy to distinguish against the map background.



18. One neat feature of the Print Layout is that it can automatically highlight the area from the main map which is represented in the inset. Select the `Map 1` object from the Items panel. In the Item properties tab, scroll down to the Overviews section. Click the Add a new overview button.

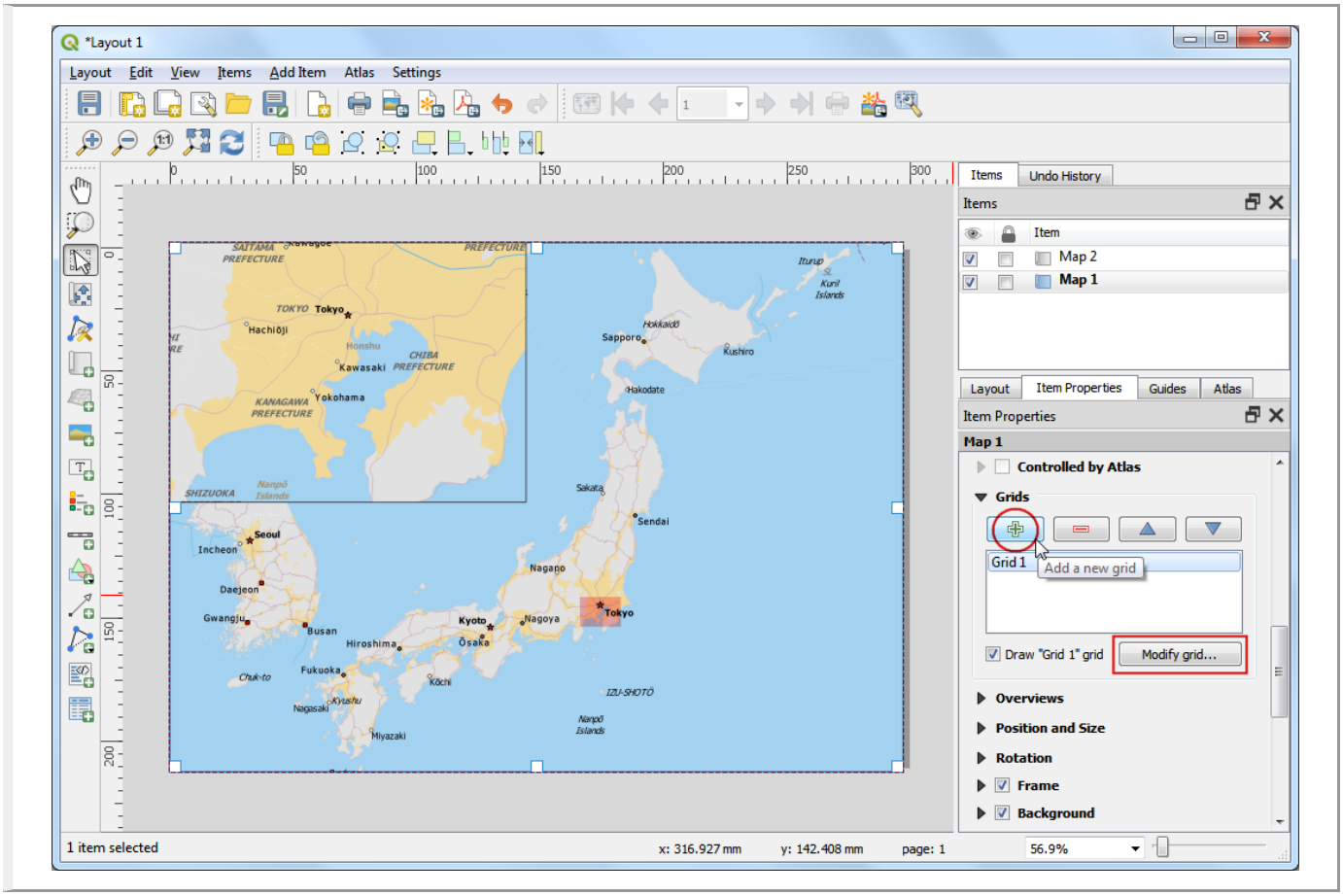


19. Select `Map 2` as the Map Frame. This tells the Print Layout to highlight the current object `Map 1` with the extent of the map shown in the `Map 2` object.

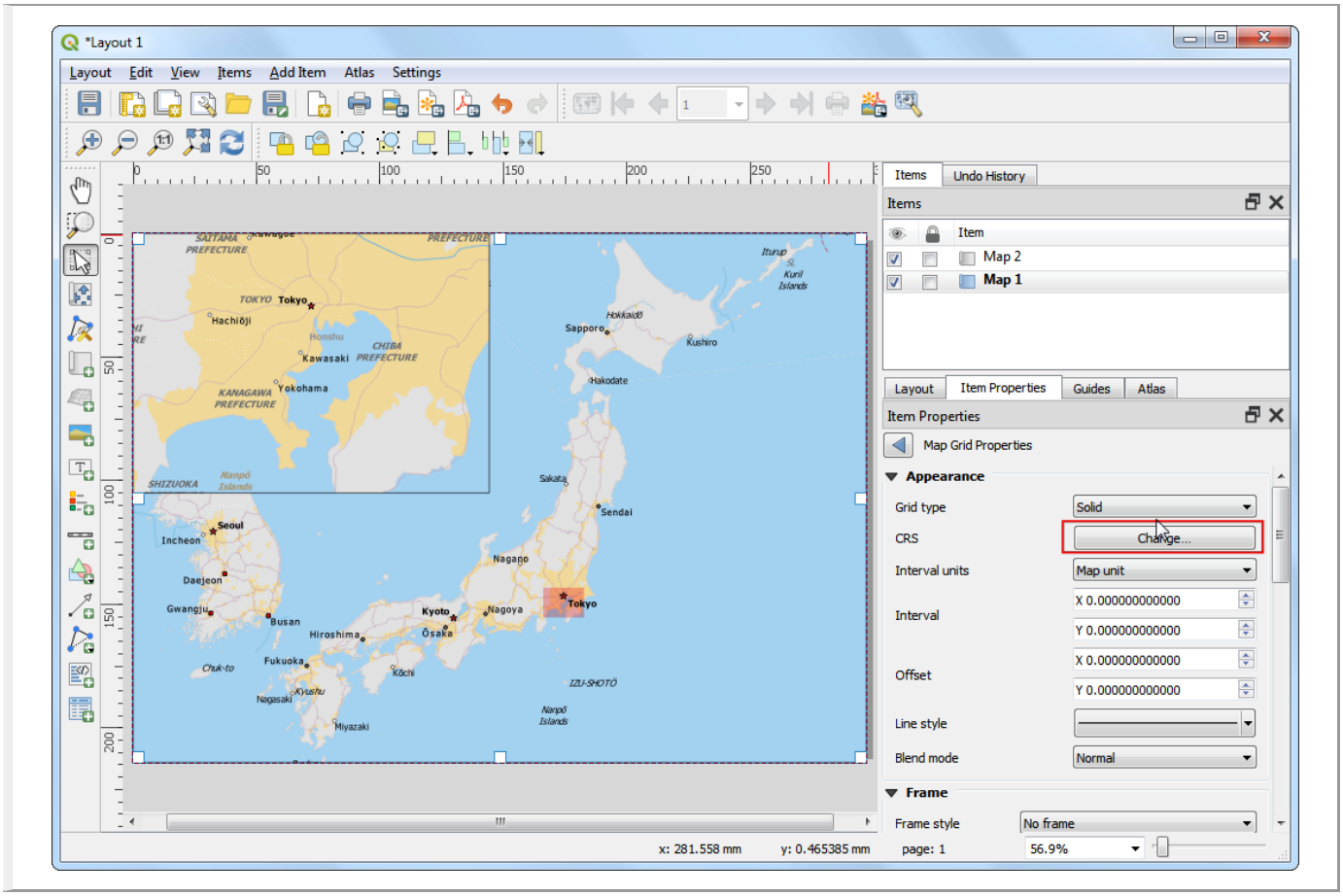


20. Now that we have the map inset ready, we will add a grid to the main map. Select the `Map 1` object from the Items panel. In the Item properties tab, scroll down to the Grids section. Click the Add a new

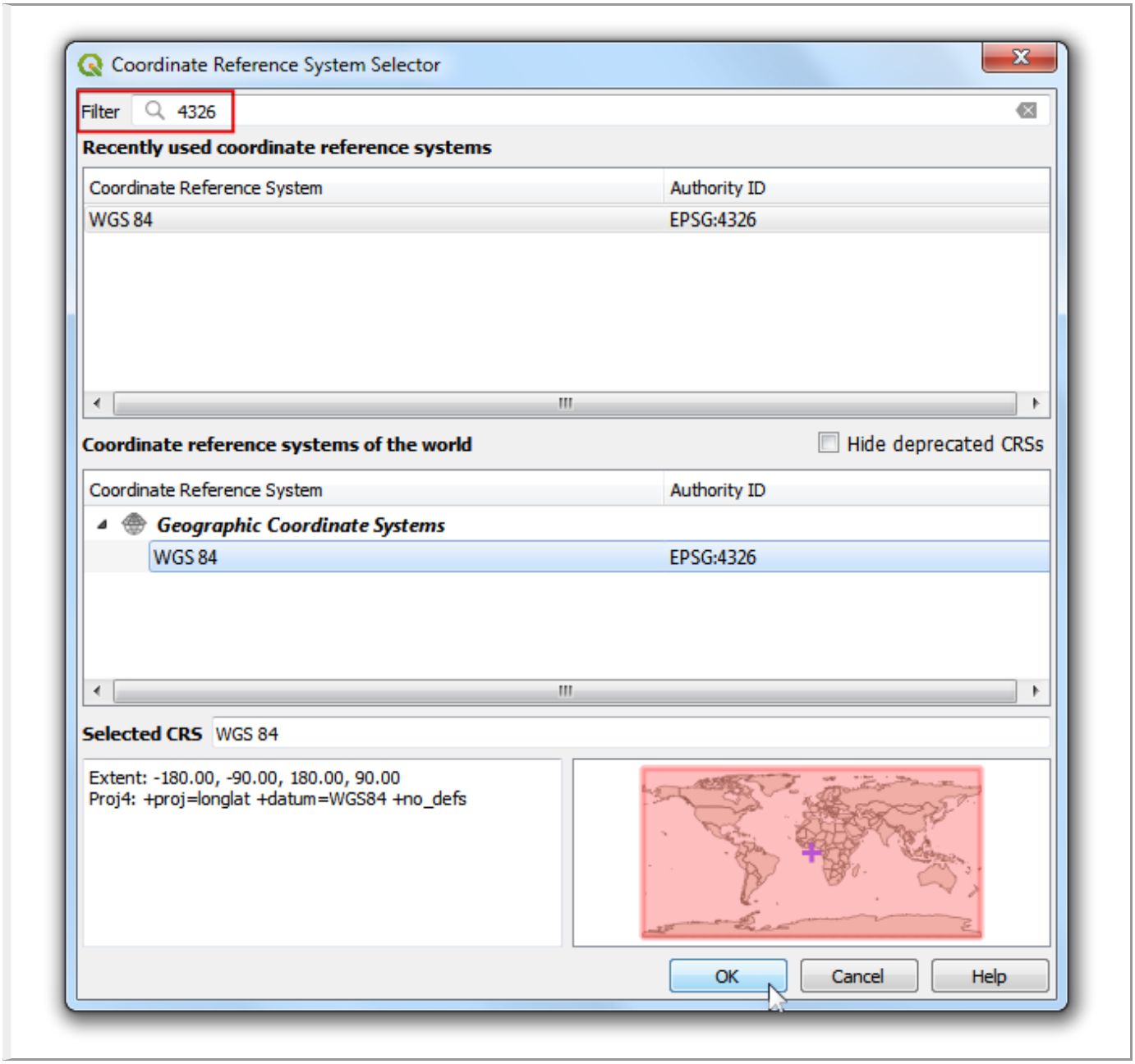
grid button, followed by Modify grid....



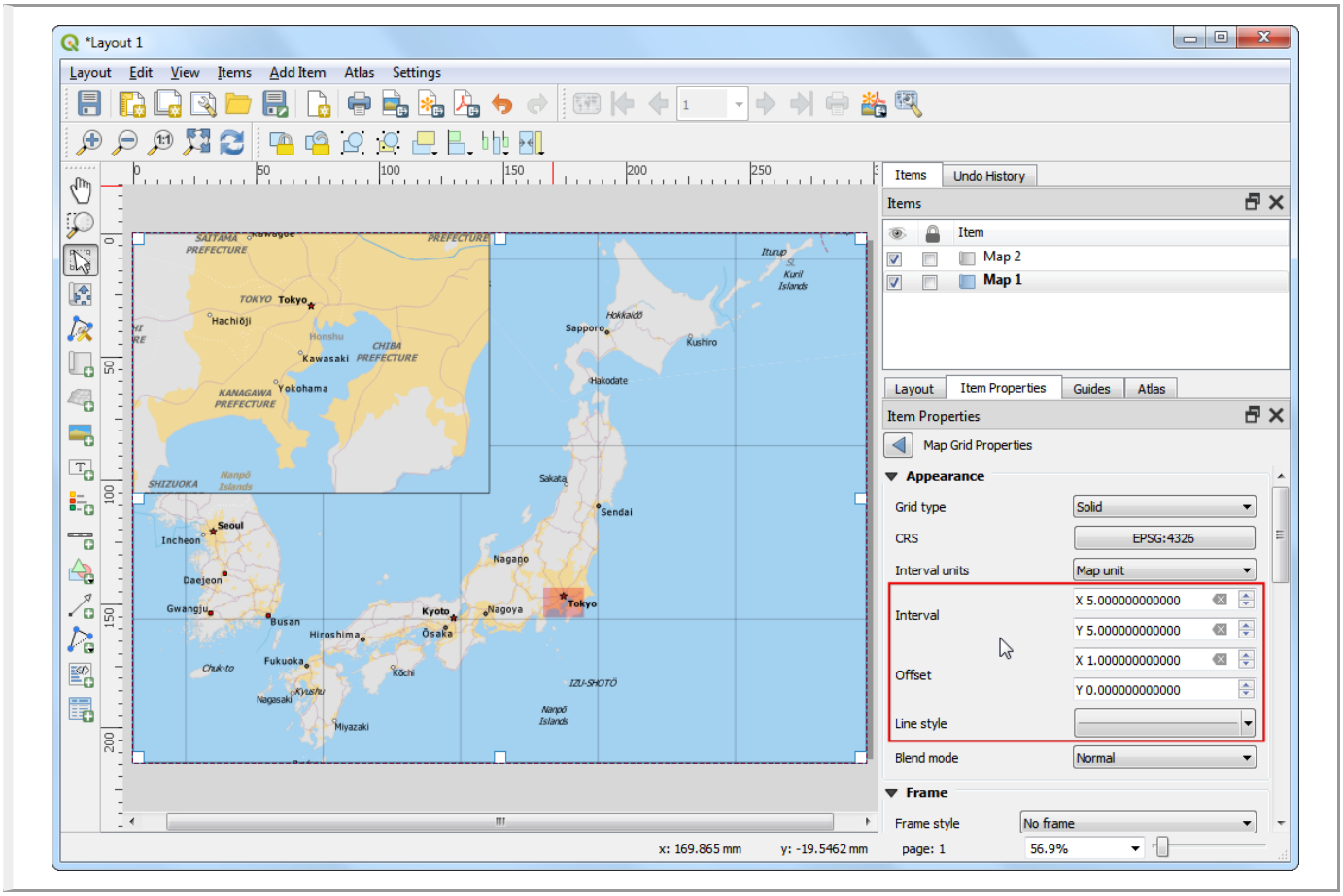
21. By default, the grid lines use the same units and projections as the currently selected map projections. However, it is more common and useful to display grid lines in degrees. We can select a different CRS for the grid. Click on the Change... button next to CRS.



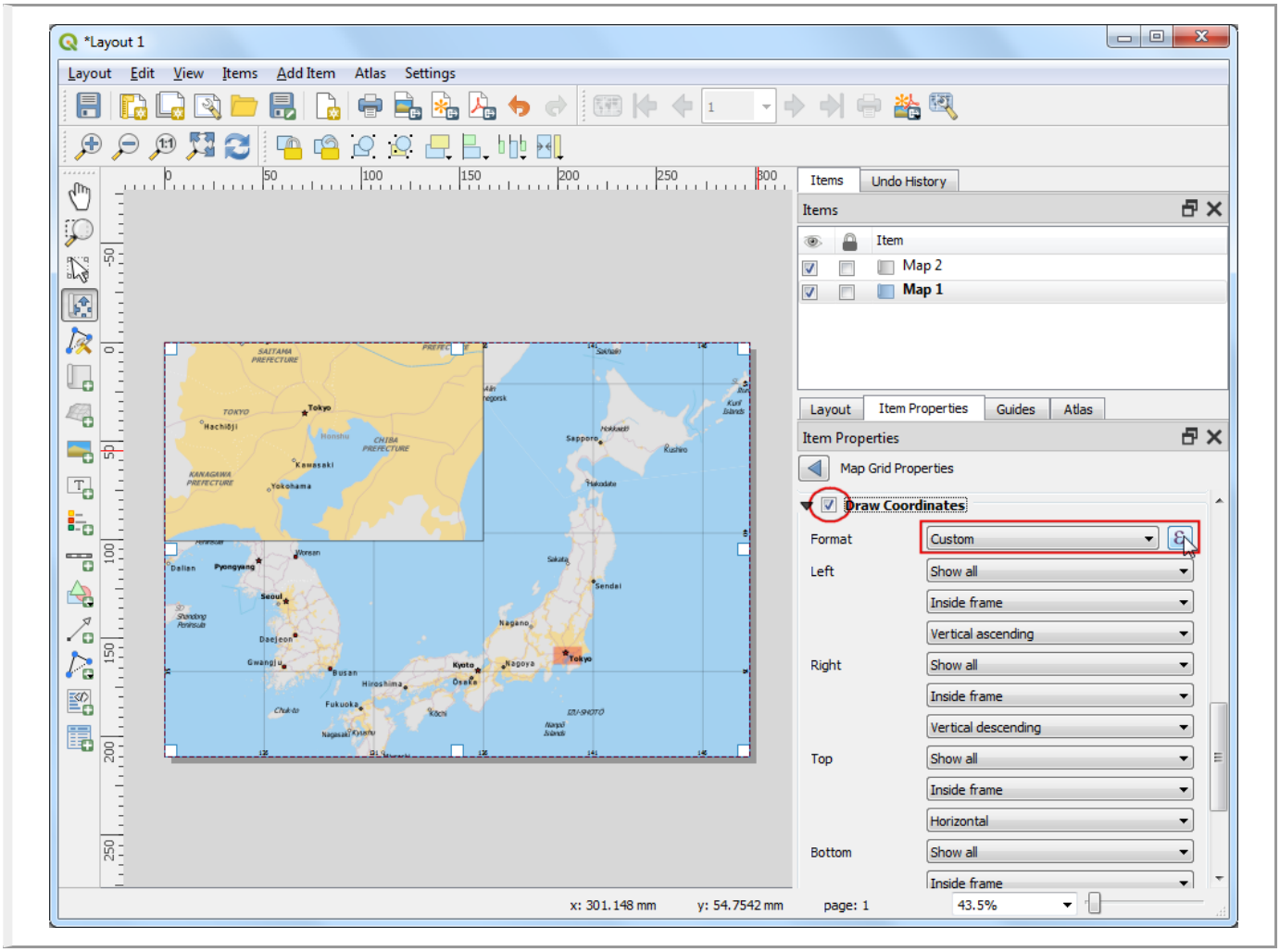
22. In the Coordinate Reference System Selector dialog, enter `4326` in the Filter box. From the results, select the `WGS84 EPSG:4326` as the CRS. Click OK.



23. Select the Interval values as `5` degrees in both X and Y direction. You can adjust the Offset to change where the grid lines appear.

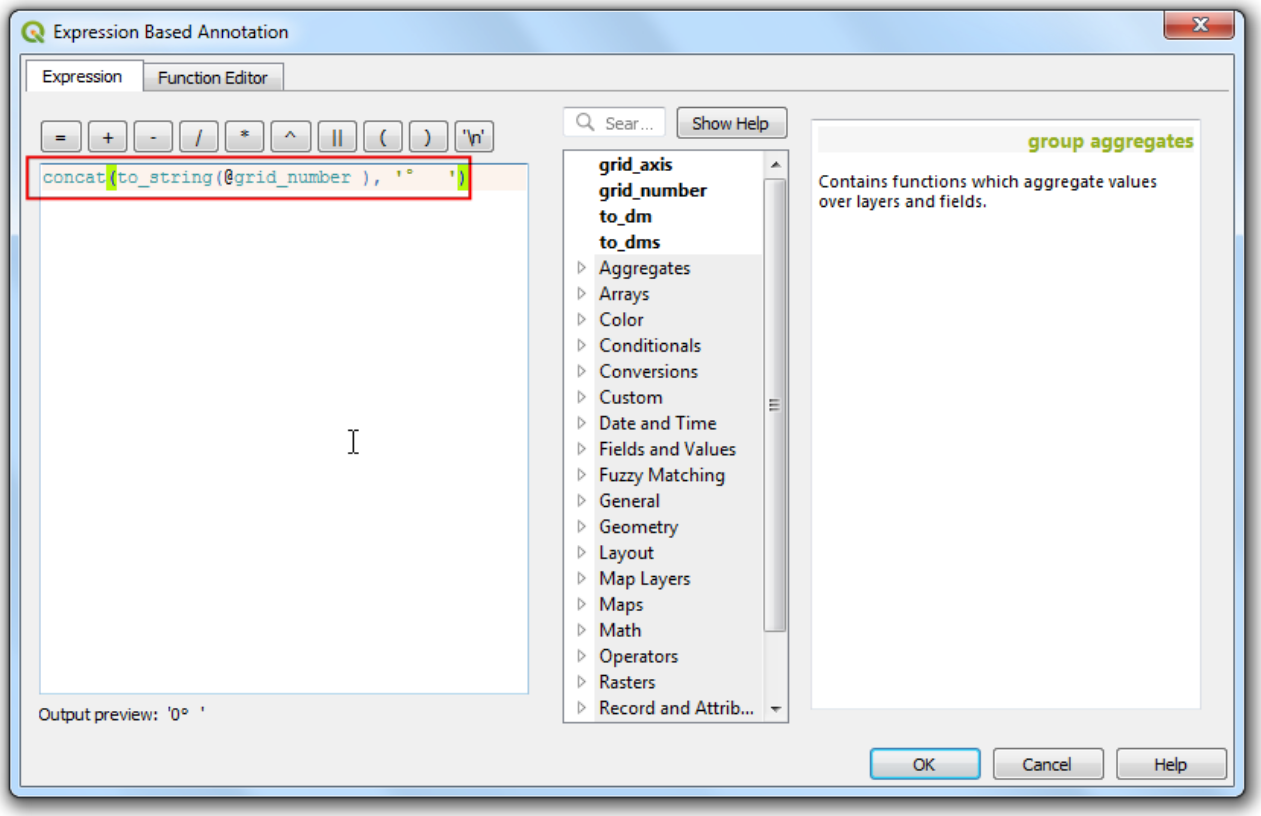


24. Scroll down to the Grid frame section and check the Draw coordinates box. The default format is `Degrees` but it appears as a number. We can customize is to append a ° symbol. Choose `Custom` and click the Expression button next to it.

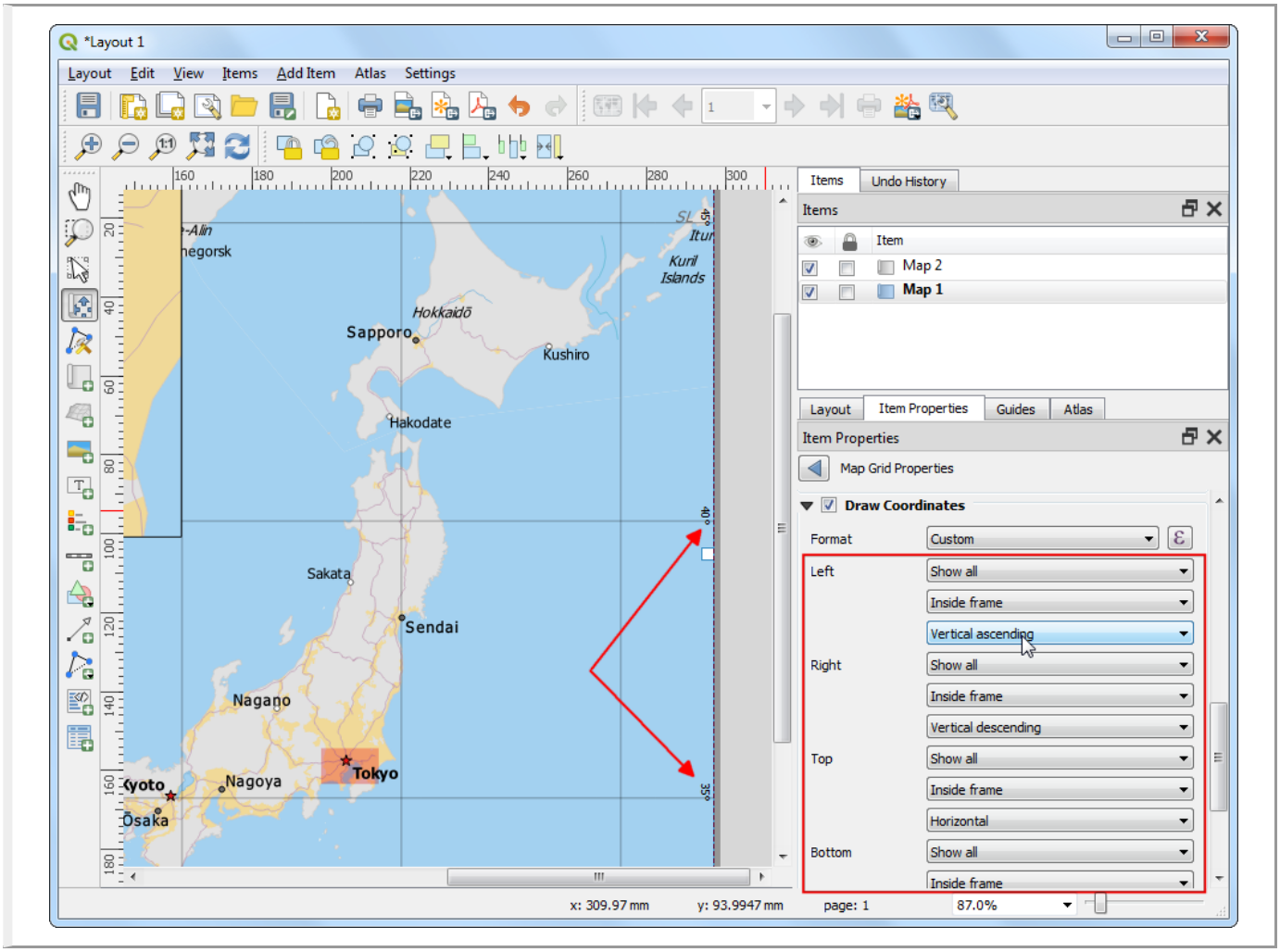


25. Enter the following expression to create a string that takes the grid number and appends ° symbol to it.

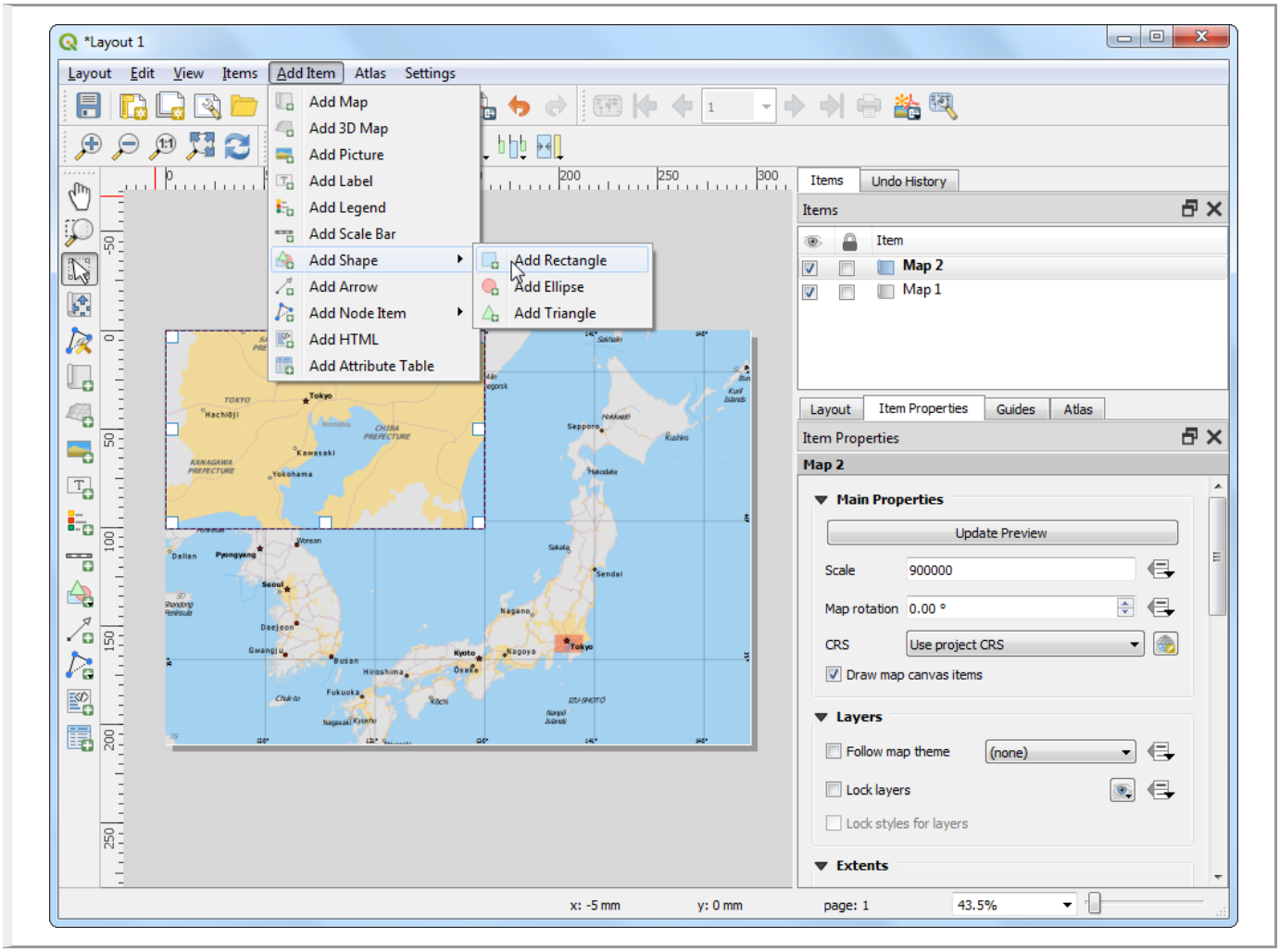
```
concat(to_string(@grid_number), '°    ')
```



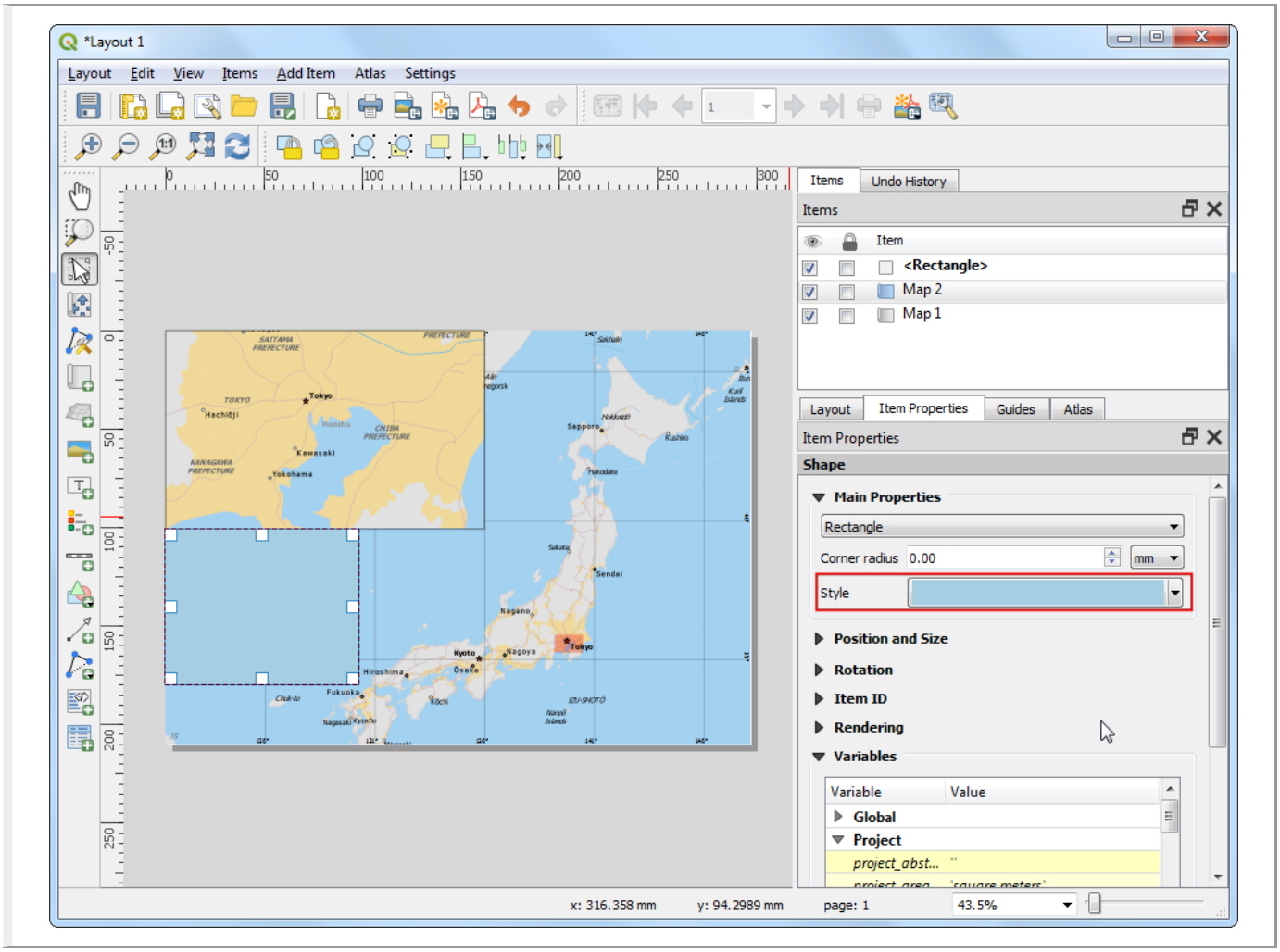
26. Notice that the grids now have a custom label from the expression. Adjust the position settings for Left, Right, Top and Bottom as per your liking.



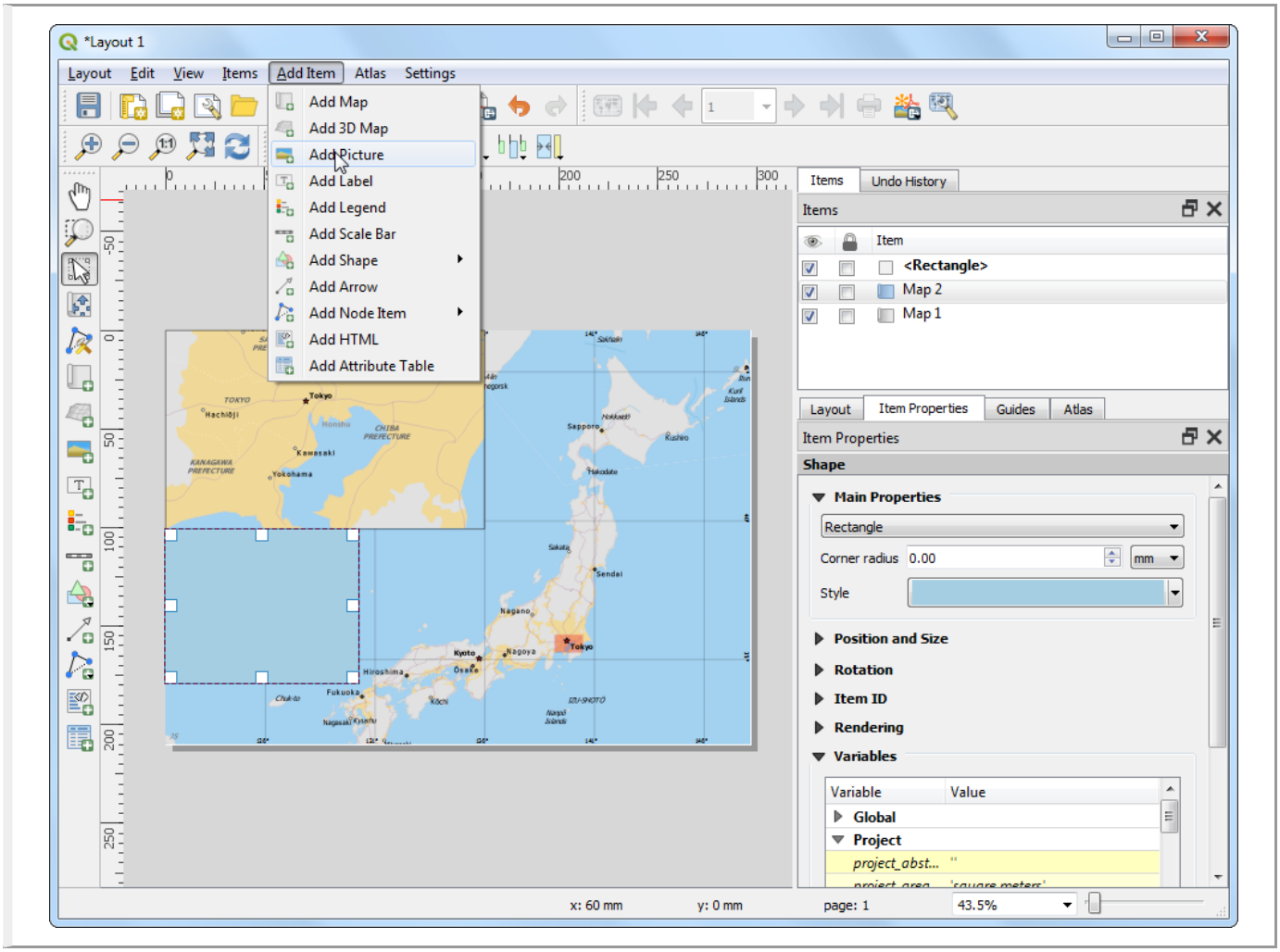
27. Now we will add a Rectangluar frame to hold other map elements like north arrow, scale and label. Go to Add Item ▸ Add Shape ▸ Add Rectangle.



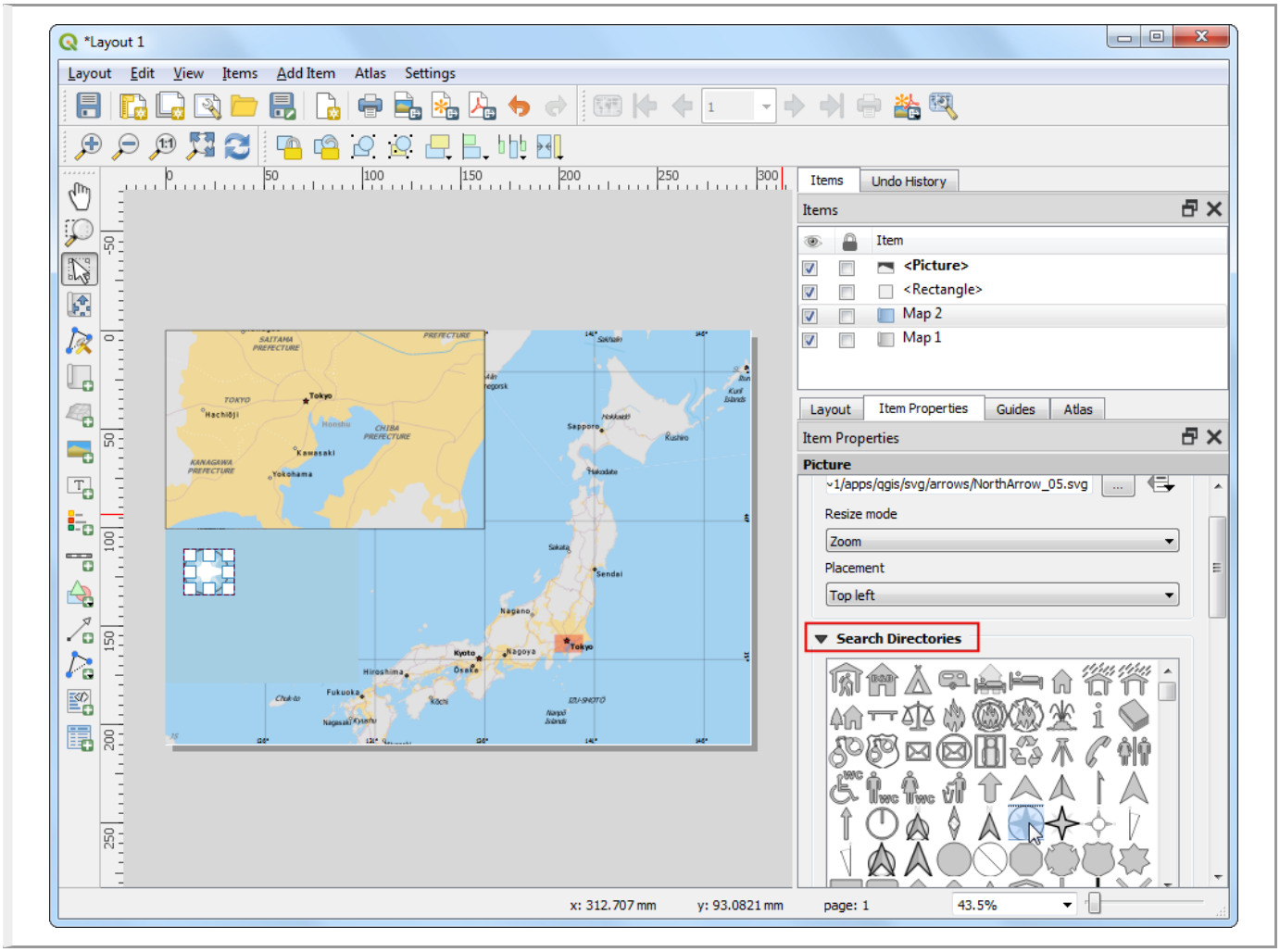
28. You can change the Style of the rectangle to match the map background.



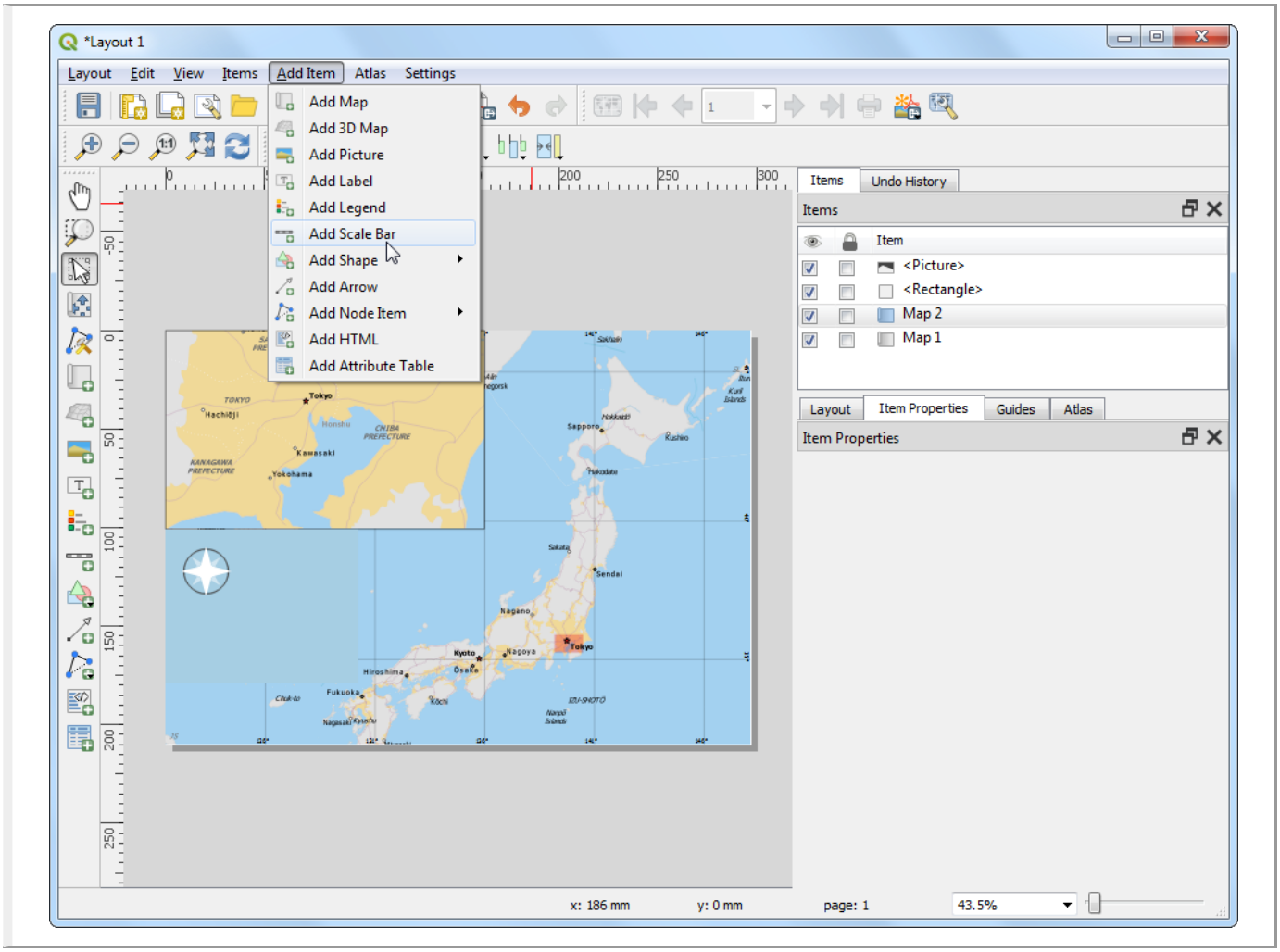
29. Now we will add a North Arrow to the map. QGIS comes with a nice collection of map-related images - including many types of North Arrows. Click Add Item ‣ Add Picture.



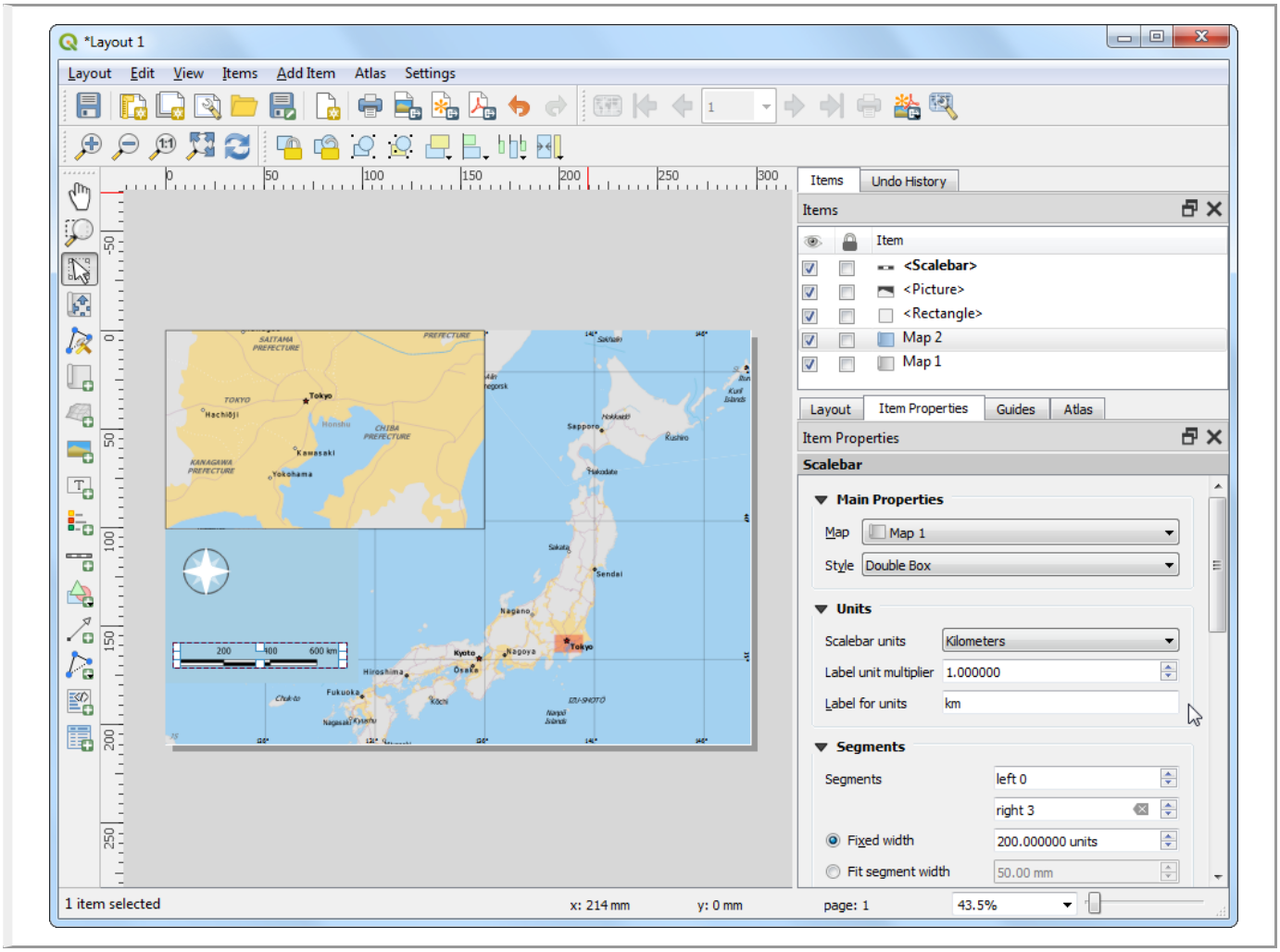
30. Holding your left mouse button, draw a rectangle. On the right-hand panel, click on the Item Properties tab and expand the Search directories section and select the image of your liking.



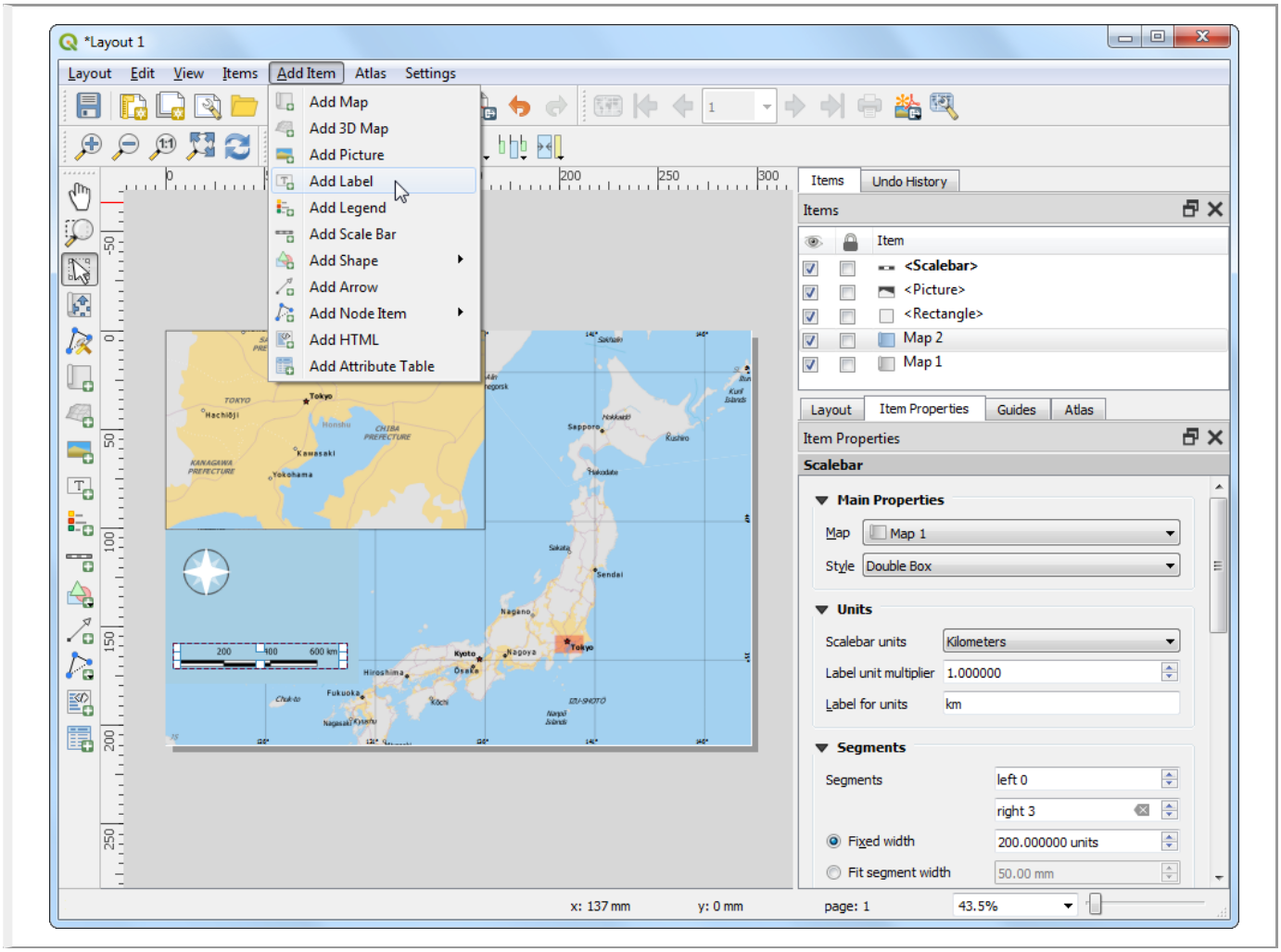
31. Now we will add a scale bar. Click on Add Item ‣ Add Scalebar.



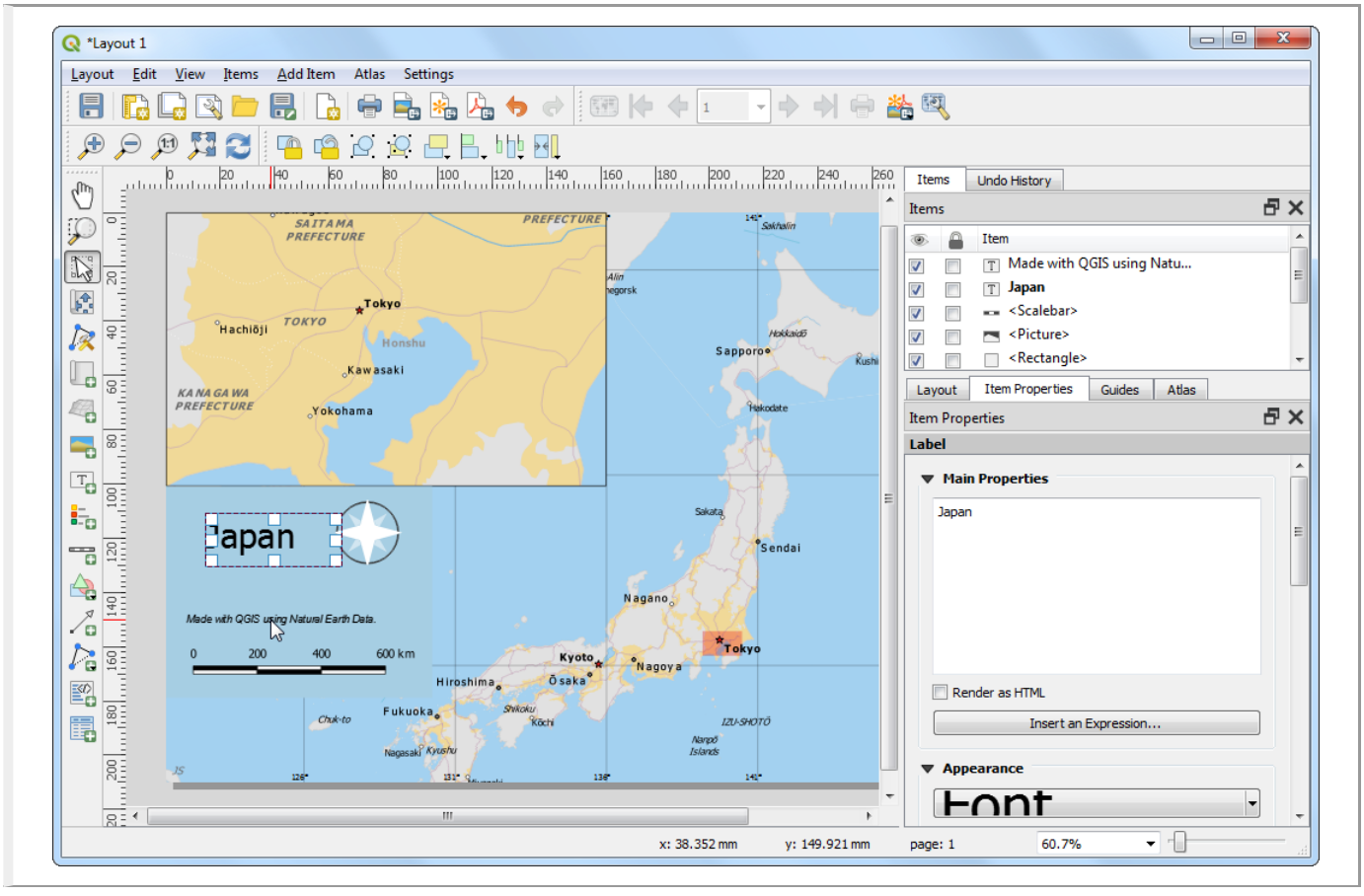
32. Click on the layout where you want the scalebar to appear. In the Item Properties tab, make sure you have chosen the correct map element `Map 1` for which to display the scalebar. Choose the Style that fit your requirement. In the Segments panel, change the Fixed width to `200` units and adjust the segments to your liking.



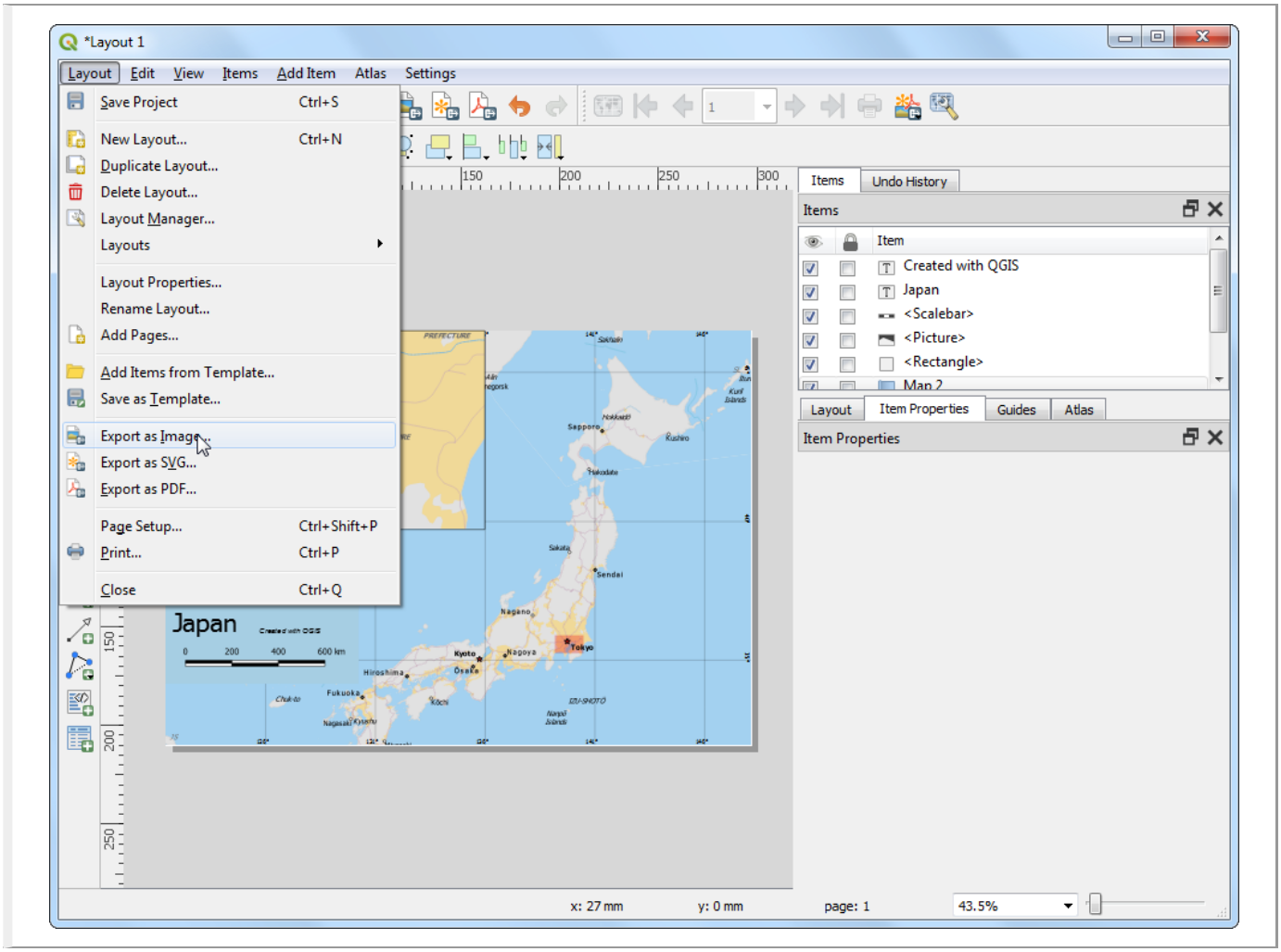
33. It is time to label our map. Click on Add Item ‣ Add Label.



34. Click on the map and draw a box where the label should be. In the Item Properties tab, expand the Label section and enter a label for the map. Similarly add another labels for data and software credits.



35. Once you are satisfied with the map, you can export it as an Image, PDF or SVG. For this tutorial, let's export it as an image. Click Layout ‣ Export as Image.



35. Save the image in the format of your liking. Below is the exported PNG image.

Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Working with Attributes (QGIS3)

GIS data has two parts - features and attributes. Attributes are structured data about each feature. This tutorial shows how to view the attributes of a GIS vector layer and do basic queries on them in QGIS.

Overview of the task

The dataset for this tutorial contains information about populated places of the world. The task is to query and find all the capital cities in the world that have a population greater than 1 million and save the resulting subset as a GeoJSON file.

Other skills you will learn

- Select features from a layer using expressions.
- Using the Attributes toolbar.
- Exporting selected features in a layer

Get the data

Natural Earth provides a Populated Places (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-populated-places/) dataset. Download the simple (less columns) dataset (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_populated_places_simple.zip)

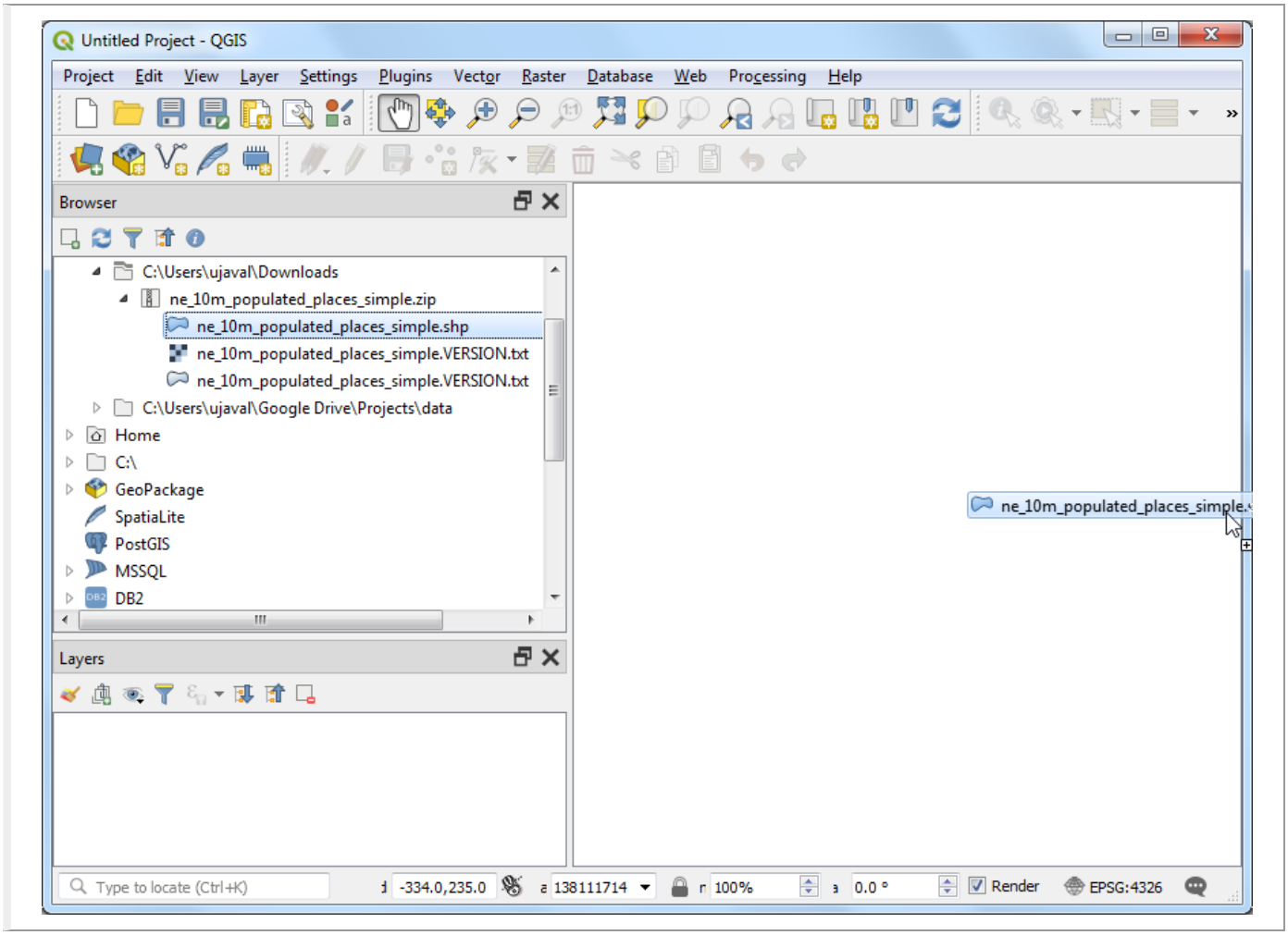
For convenience, you may directly download a copy of datasets from the link below:

ne_10m_populated_places_simple.zip (http://www.qgistutorials.com/downloads/ne_10m_populated_places_simple.zip)

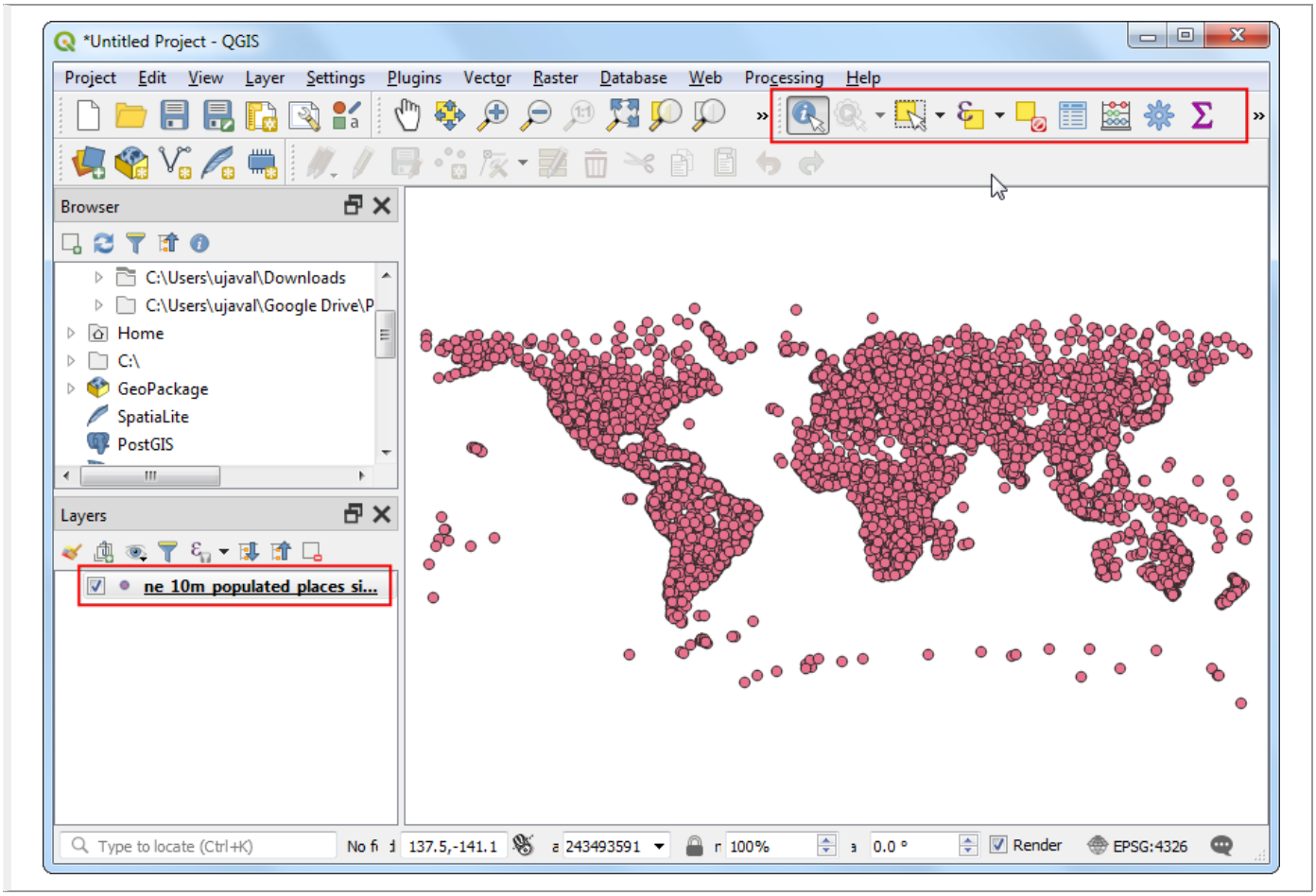
Data Source [NATURALEARTH] (../credits.html#naturalearth)

Procedure

1. Locate the `ne_10m_populated_places_simple.zip` file in the QGIS Browser and expand it. Select the `ne_10m_populated_places_simple.shp` file and drag it to the canvas.



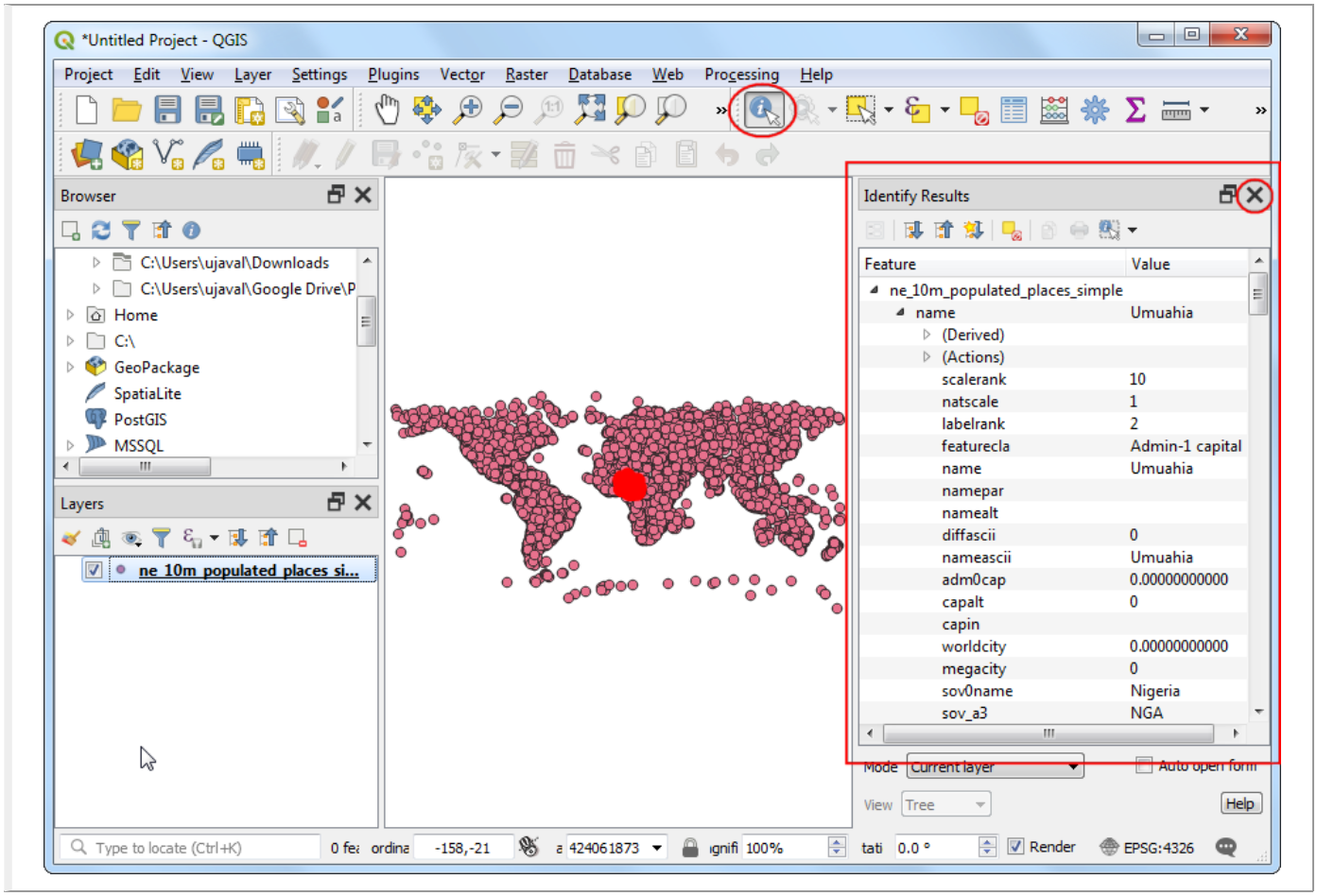
2. A new layer `ne_10m_populated_places_simple` will now be loaded in QGIS and you will see many points representing the populated places of the world. The default view in the QGIS canvas shows the geometry of the GIS layer. Each point also has associated attributes. Let's view them. Locate the Attributes Toolbar. This toolbar contains many useful tools to inspect, view, select, and modify attributes of a layer.



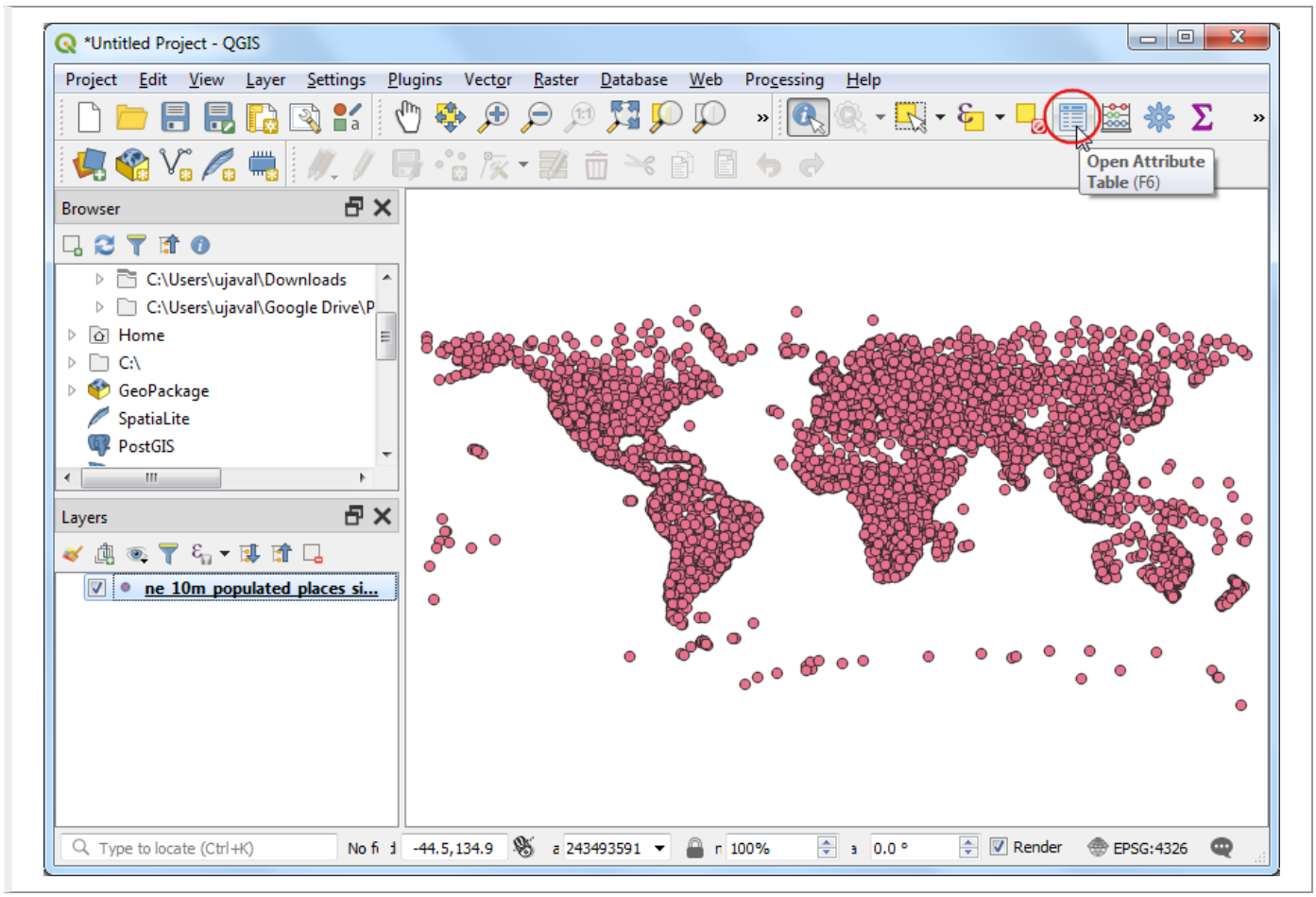
Note

If you do not see the toolbar, you can enable it from View ▸ Toolbars ▸ Attributes Toolbar

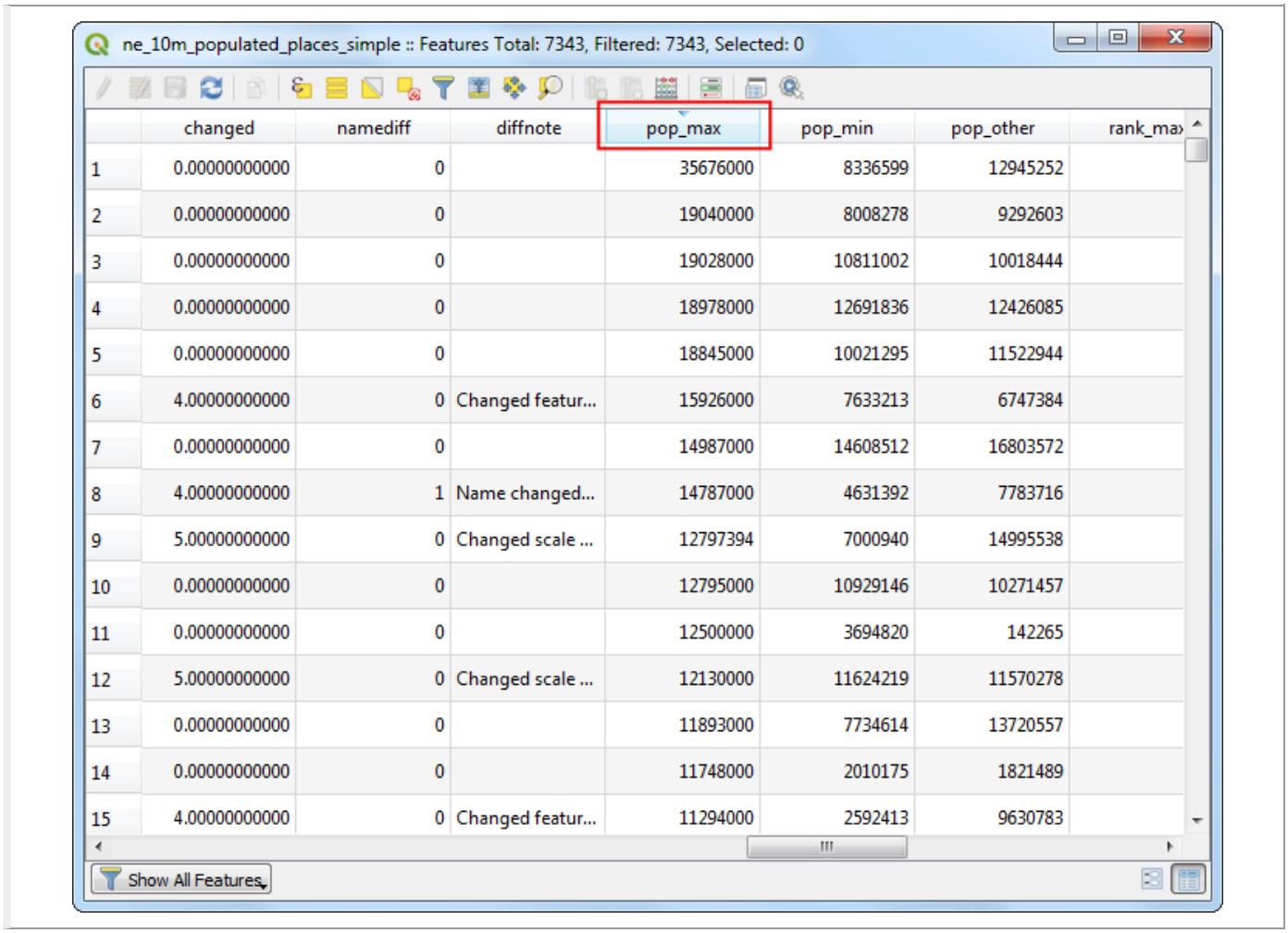
3. Click the Identify button on the Attributes Toolbar. Once the tool is selected, click on any point on the canvas. The associated attributes of that point will be displayed in a new Identify Results panel. Once you are done exploring attributes of different points, you can click the Close button.



4. Rather than viewing the attribute one feature at a time, we can view them all together as a table. Click the Open Attribute Table button on the Attributes Toolbar. You can also right-click the `ne_10m_populated_places_simple` layer and select Open Attribute Table.

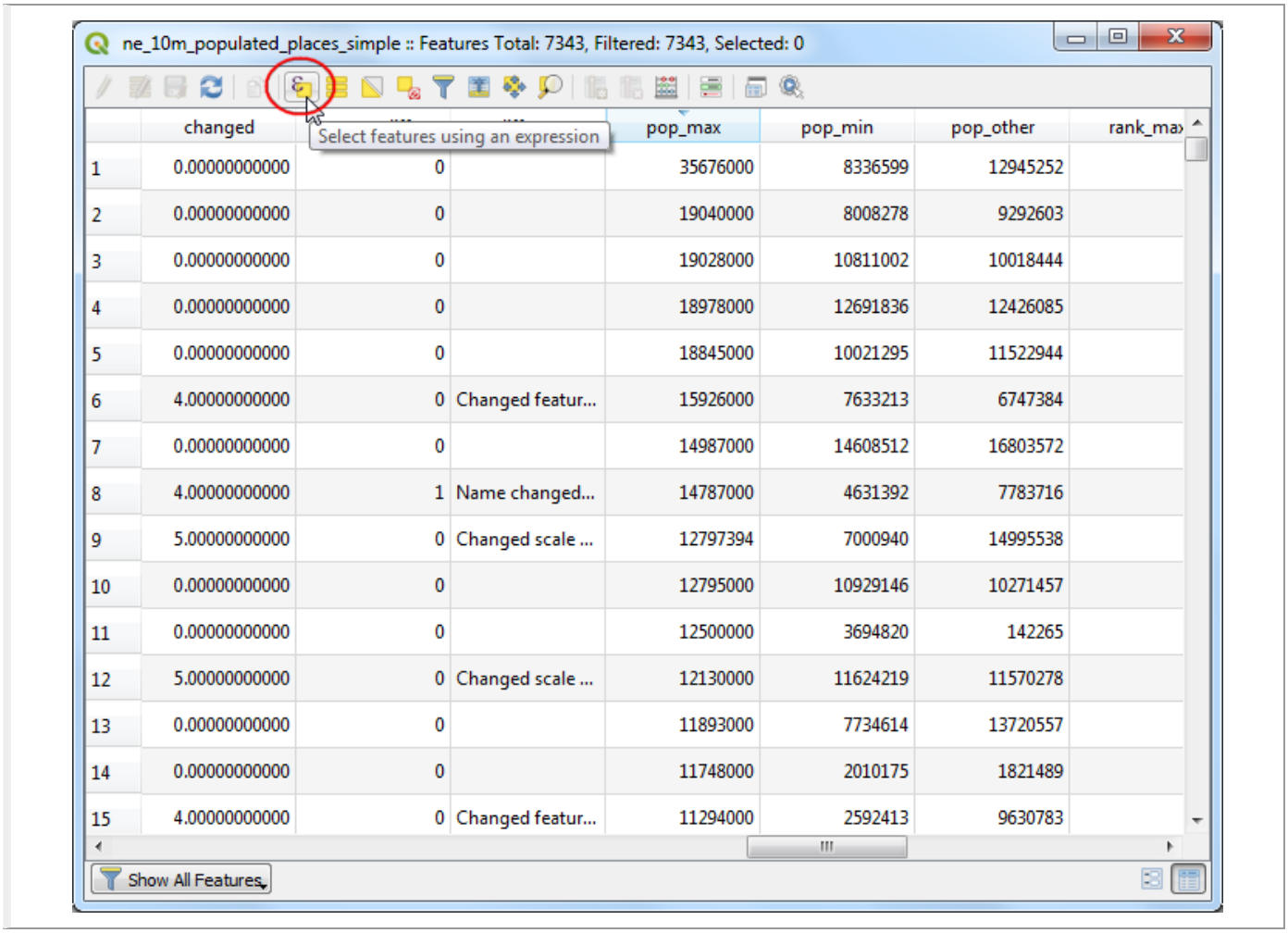


5. You can scroll horizontally and locate the **pop_max** column. This field contains the population of the associated place. You can click twice on the field header to sort the column in descending order.



	changed	namediff	diffnote	pop_max	pop_min	pop_other	rank_ma
1	0.00000000000	0		35676000	8336599	12945252	
2	0.00000000000	0		19040000	8008278	9292603	
3	0.00000000000	0		19028000	10811002	10018444	
4	0.00000000000	0		18978000	12691836	12426085	
5	0.00000000000	0		18845000	10021295	11522944	
6	4.00000000000	0	Changed featur...	15926000	7633213	6747384	
7	0.00000000000	0		14987000	14608512	16803572	
8	4.00000000000	1	Name changed...	14787000	4631392	7783716	
9	5.00000000000	0	Changed scale ...	12797394	7000940	14995538	
10	0.00000000000	0		12795000	10929146	10271457	
11	0.00000000000	0		12500000	3694820	142265	
12	5.00000000000	0	Changed scale ...	12130000	11624219	11570278	
13	0.00000000000	0		11893000	7734614	13720557	
14	0.00000000000	0		11748000	2010175	1821489	
15	4.00000000000	0	Changed featur...	11294000	2592413	9630783	

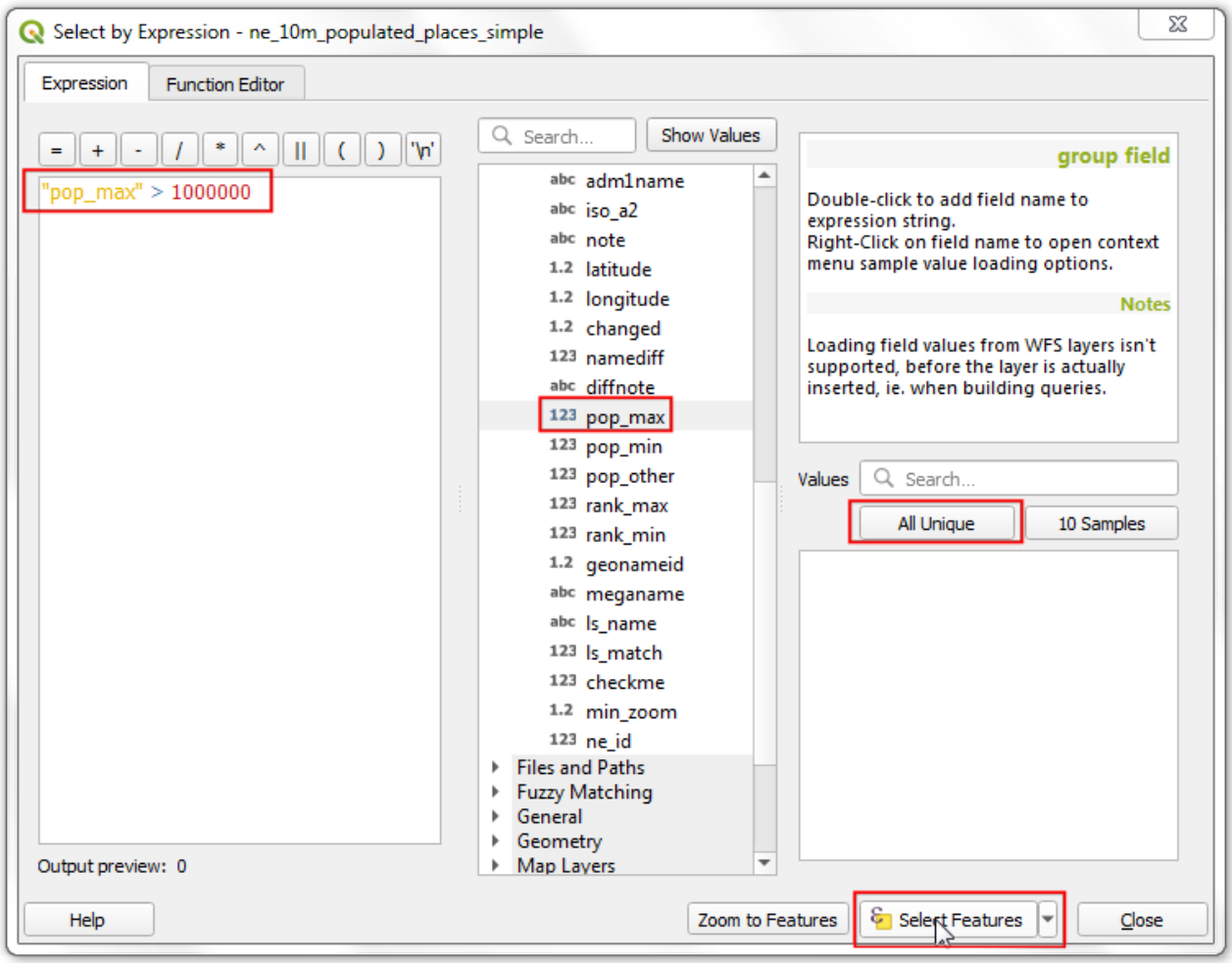
6. Now we are ready to perform our query on these attributes. QGIS uses SQL-like expressions to perform queries. Click Select features using an expression button.



	changed			pop_max	pop_min	pop_other	rank_max
1	0.00000000000	0		35676000	8336599	12945252	
2	0.00000000000	0		19040000	8008278	9292603	
3	0.00000000000	0		19028000	10811002	10018444	
4	0.00000000000	0		18978000	12691836	12426085	
5	0.00000000000	0		18845000	10021295	11522944	
6	4.00000000000	0	Changed featur...	15926000	7633213	6747384	
7	0.00000000000	0		14987000	14608512	16803572	
8	4.00000000000	1	Name changed...	14787000	4631392	7783716	
9	5.00000000000	0	Changed scale ...	12797394	7000940	14995538	
10	0.00000000000	0		12795000	10929146	10271457	
11	0.00000000000	0		12500000	3694820	142265	
12	5.00000000000	0	Changed scale ...	12130000	11624219	11570278	
13	0.00000000000	0		11893000	7734614	13720557	
14	0.00000000000	0		11748000	2010175	1821489	
15	4.00000000000	0	Changed featur...	11294000	2592413	9630783	

7. In the Select By Expression window, expand the Fields and Values section and double-click the `pop_max` label. You will notice that it is added to the expression section at the bottom. If you aren't sure about the field values, you can click the All Unique button to see what the attribute values are present in the dataset. For this exercise, we are looking to find all features that have a population greater than 1 million. So complete the expression as below and click Select Features and then Close.

```
"pop_max" > 1000000
```



Note

In the QGIS Expression engine, text with double-quotes refers to a field and text with single-quotes refer to a string value.

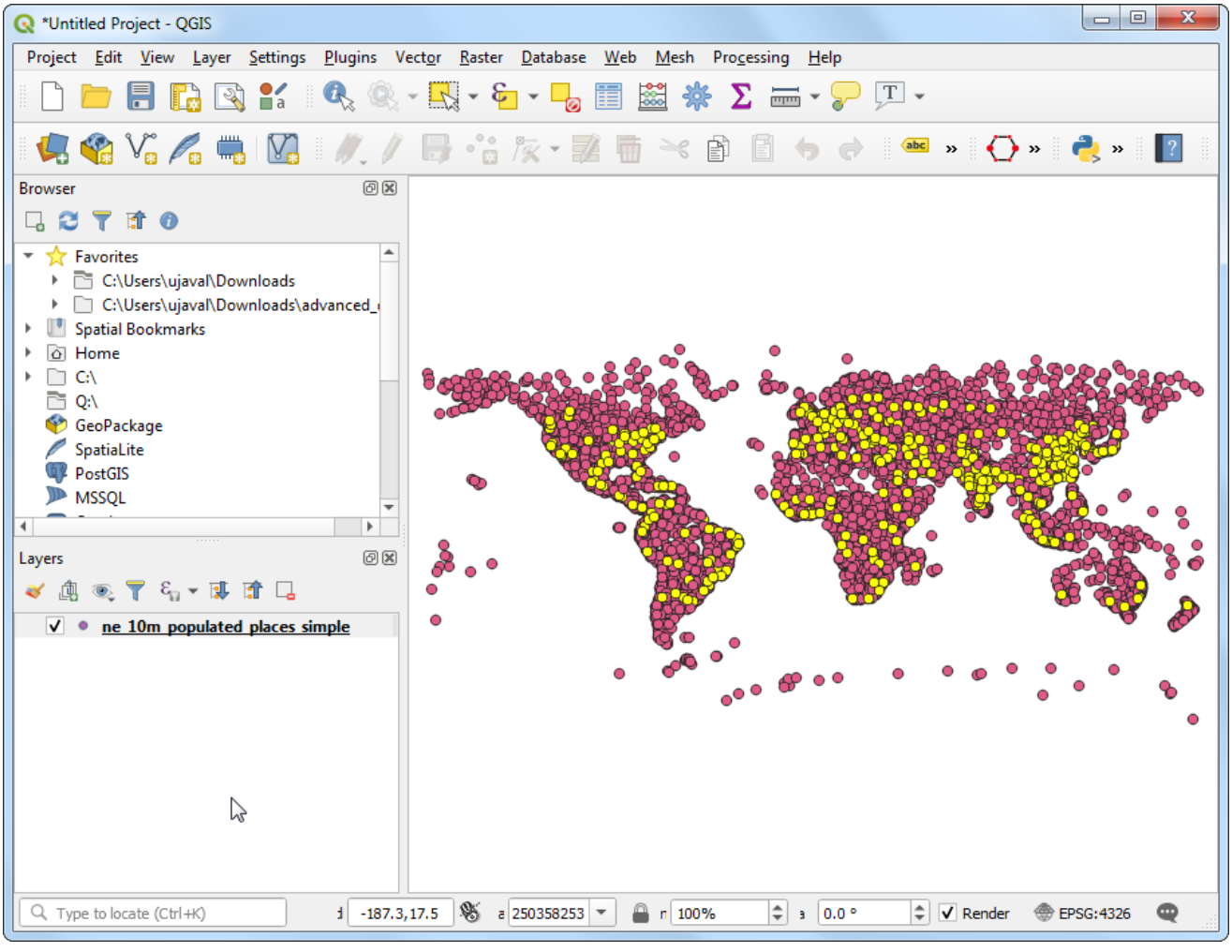
8. You will notice that some rows in the attribute table are now selected. The label window also changes and shows the count of selected features.

ne_10m_populated_places_simple :: Features Total: 7343, Filtered: 7343, Selected: 500

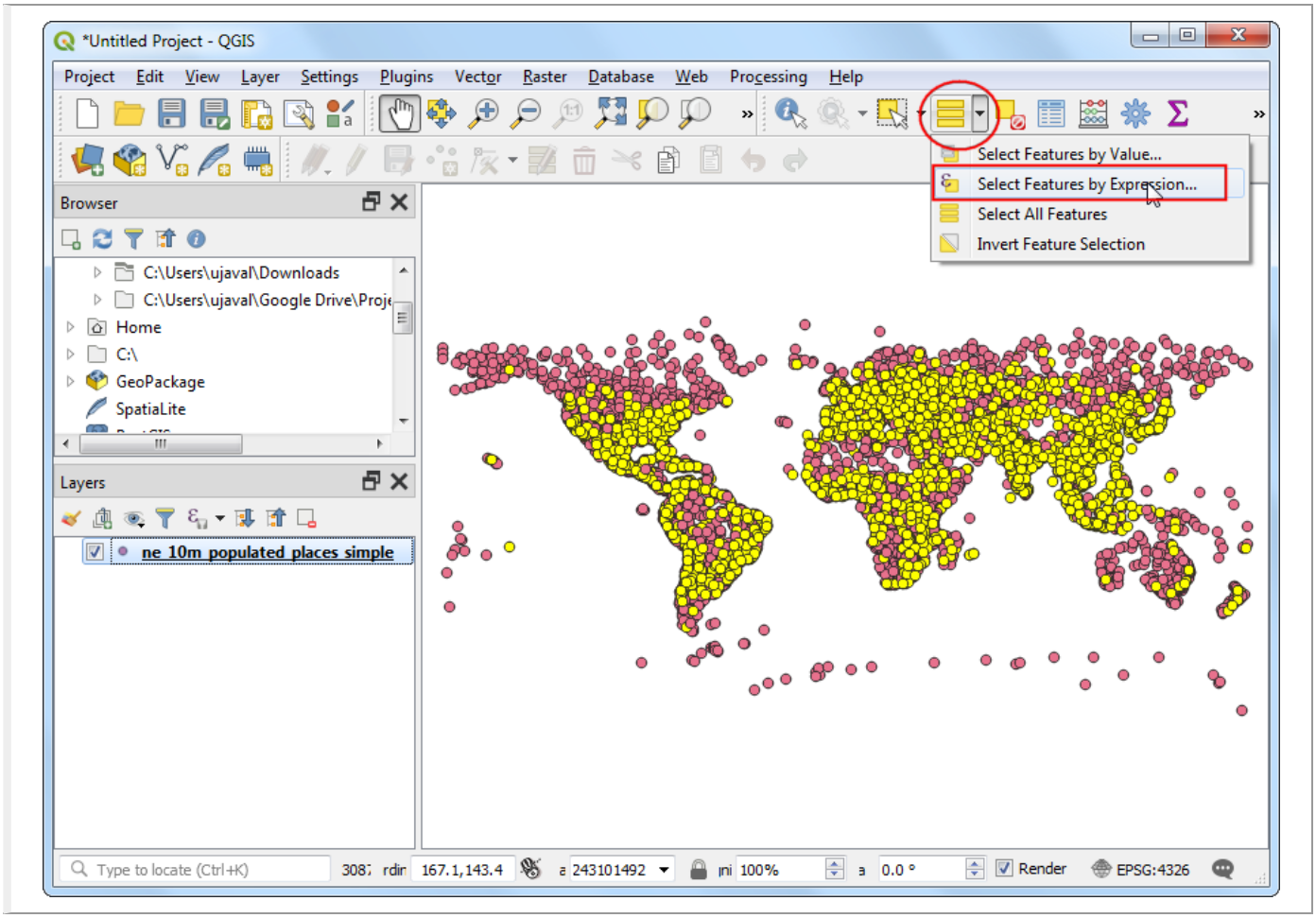
	scalerank	natscale	labelrank	featurecla	name	namepar	namealt
1	8	10	1	Populated place	Glendale		
2	8	10	1	Populated place	Centralia		
3	8	10	1	Populated place	Kennewick		
4	8	10	1	Populated place	Wallace		
5	8	10	1	Populated place	Lake Havasu City		
6	8	10	1	Populated place	Mesa		
7	8	10	1	Populated place	Casa Grande		
8	8	10	1	Populated place	Safford		
9	8	10	1	Populated place	Glendive		
10	8	10	1	Populated place	Hardin		
11	8	10	1	Populated place	Crookston		
12	8	10	1	Populated place	Brainerd		
13	8	10	1	Populated place	Burley		

Show All Features

9. Close the attribute table window and return to the main QGIS window. You will notice that a subset of points is now rendered in yellow. This is the result of our query and the selected points are the ones having `pop_max` attribute value greater than `1000000` .

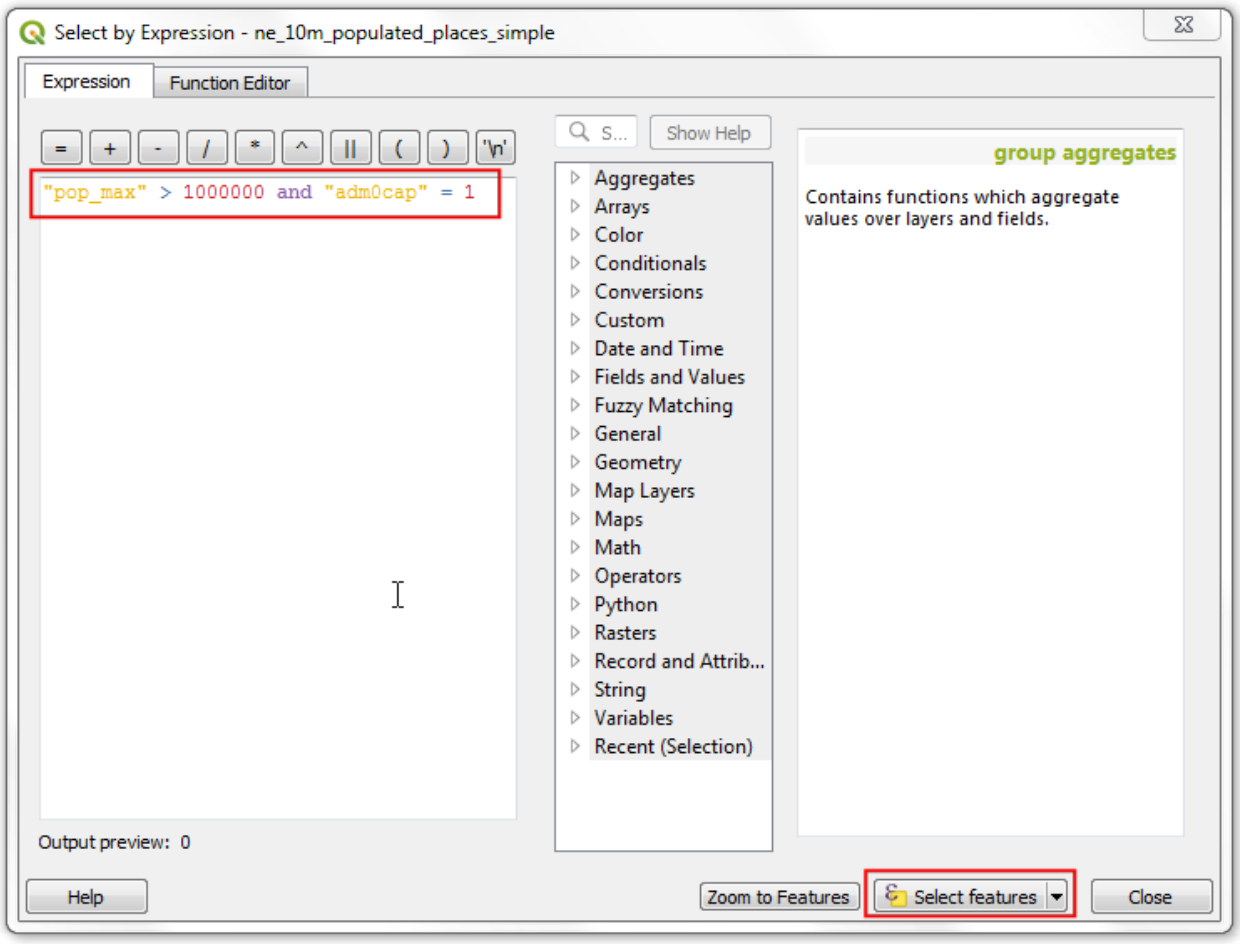


10. Let's update our query to include a condition that the place should also be a capital in addition to having a population greater than 1 million. To quickly get to the expression editor, you can use the Select Features by Expression button in the Attributes Toolbar.

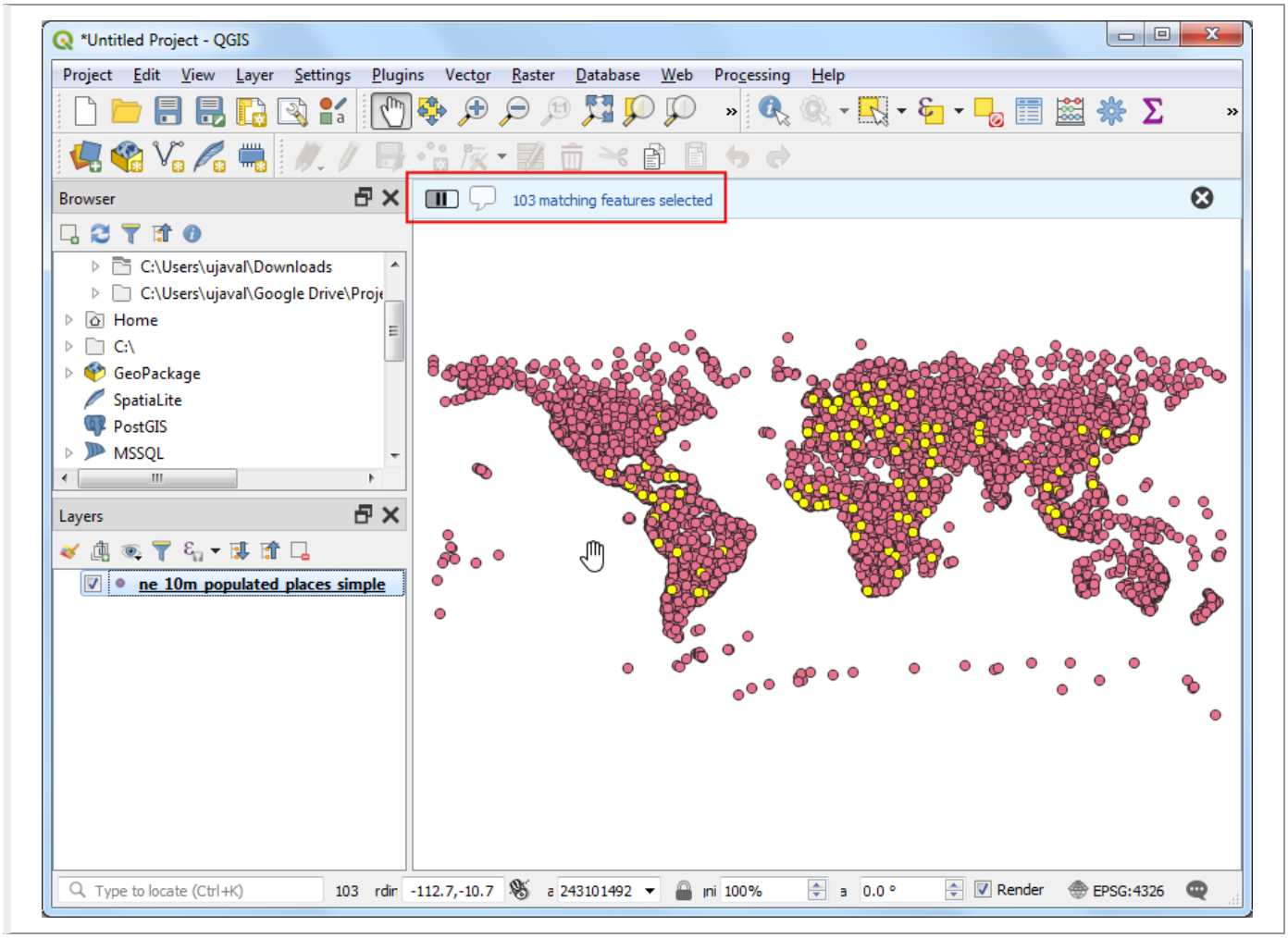


11. The field containing data about capitals is **adm0cap**. The value `1` indicates that the place is a capital. We can add this criteria to our previous expression using the *and* operator. Enter the expression as below and click Select features and then Close.

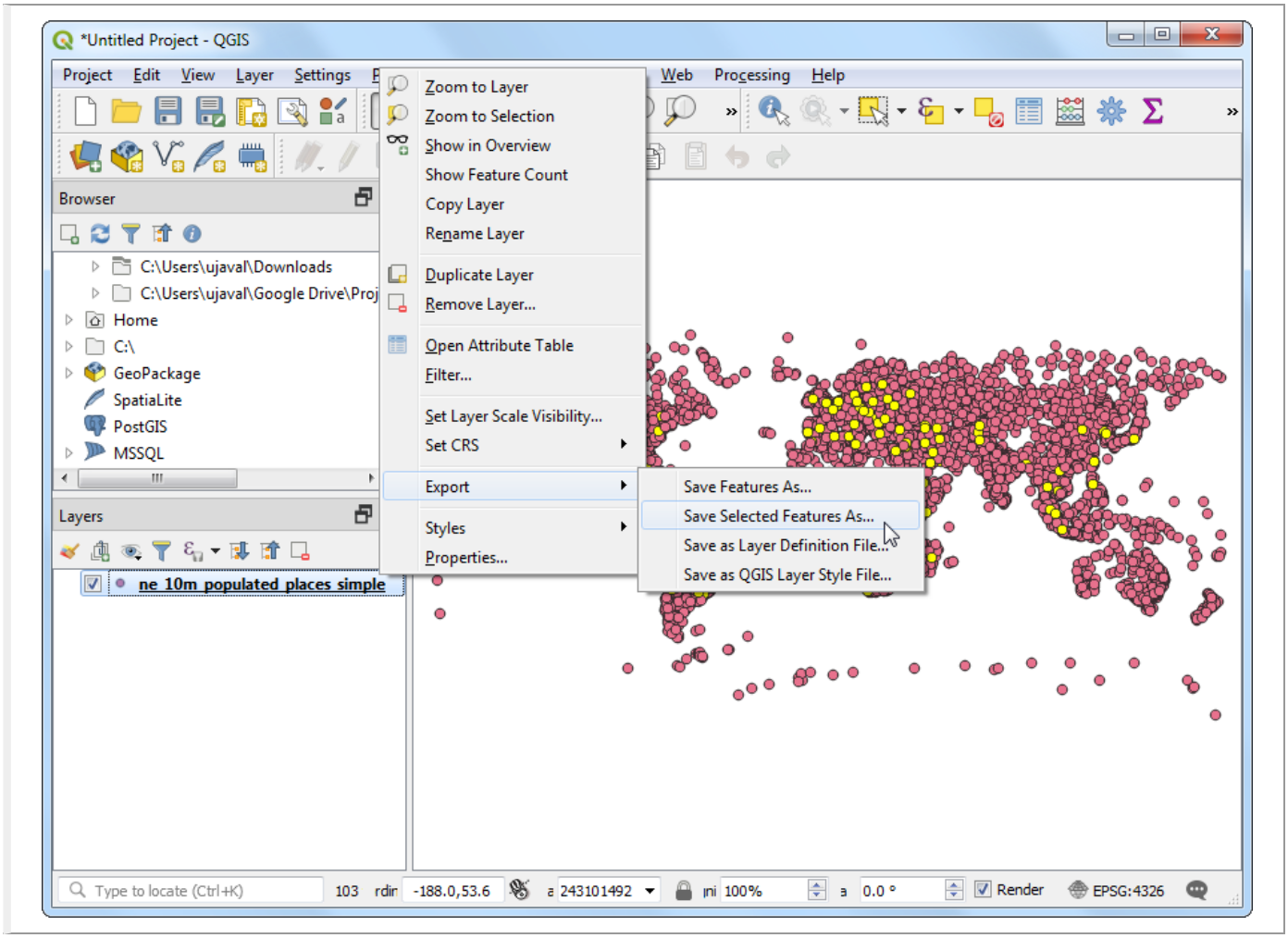
```
"pop_max" > 1000000 and "adm0cap" = 1
```



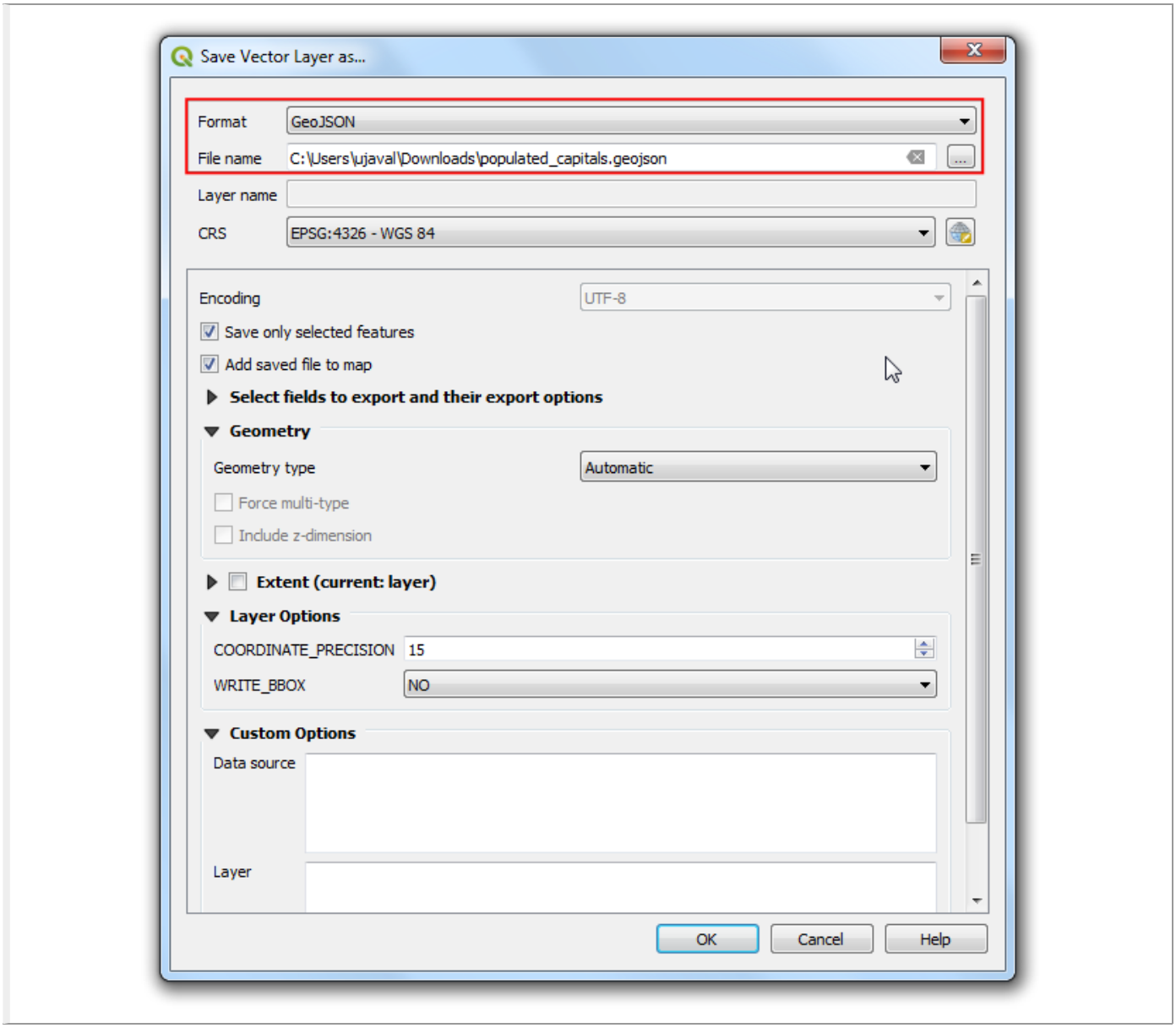
12. Return to the main QGIS window. Now you will see a smaller subset of the points selected. This is the result of the second query and shows all places from the dataset that are country capitals as well as have population greater than 1 million.



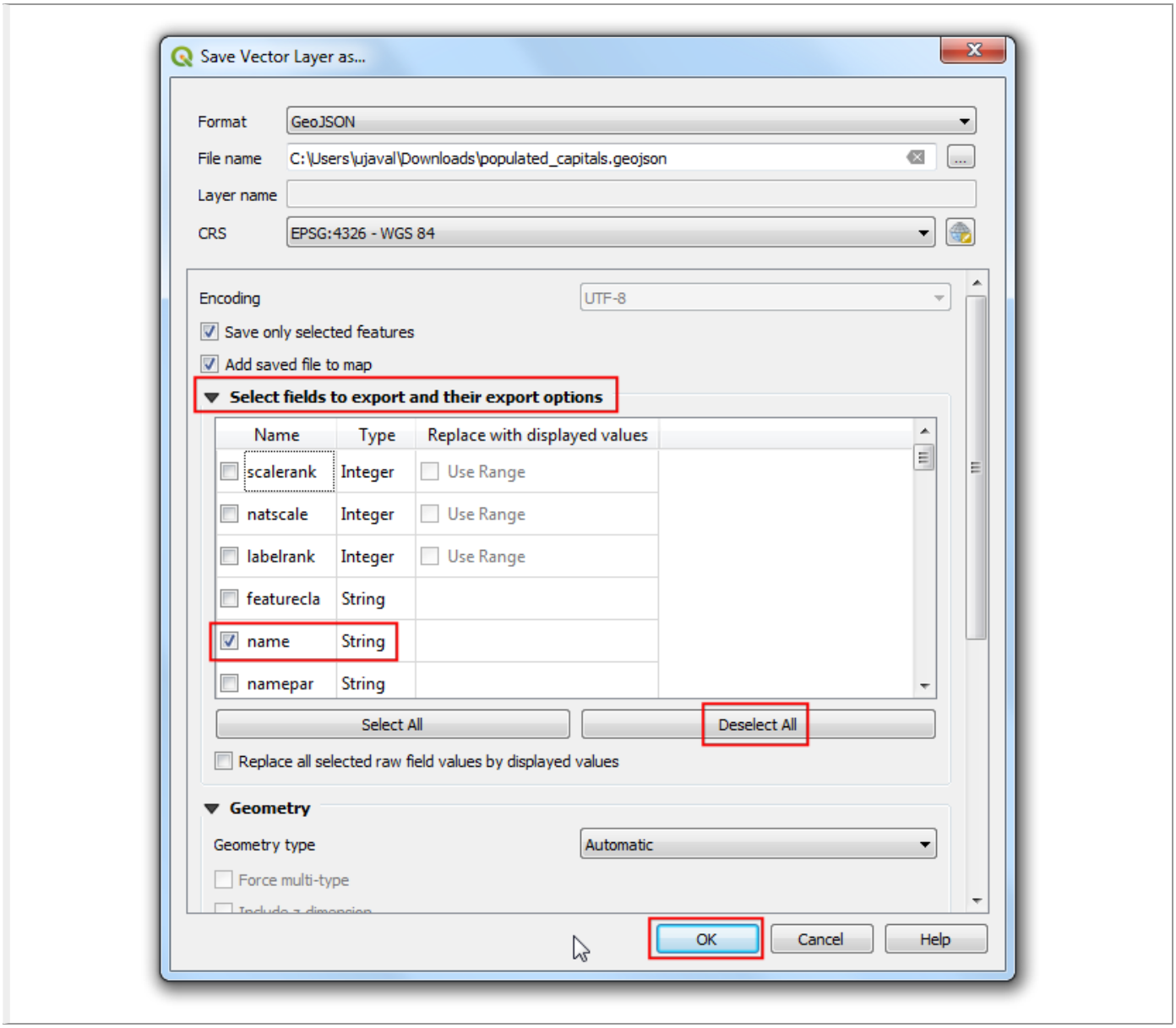
13. Now we will export the selected features as a new layer. Right-click the `ne_10m_populated_places_simple` layer and go to Export ▸ Save Selected Features As...



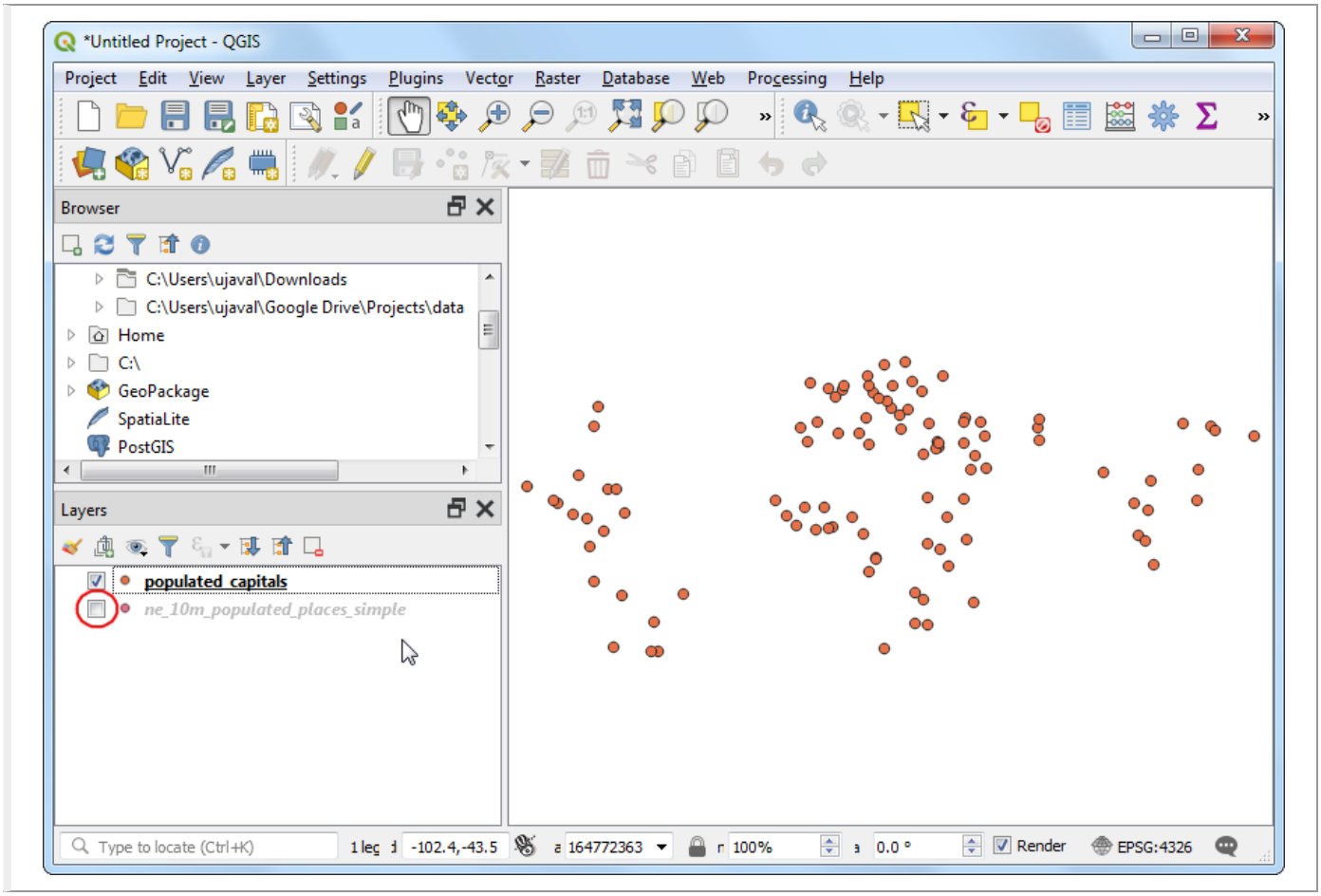
14. You may choose any format of your liking as the Format. For this exercise, we will choose `GeoJSON` . GeoJSON is a text-based format that is used widely in web mapping. Click the ... button next to File name and enter `populated_capitals.geojson` as the output file.



15. The input data has many columns. You are able to choose only a subset of the original columns for export. Expand the Select fields to export and their export options section. Click Deselect All and check the `name` and `pop_max` columns. Click OK.



16. A new layer `populated_capitals` will be loaded in QGIS. You can un-check the `ne_10m_populated_places_simple` layer to hide it and view the points from the newly exported layer.



Back to top

Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Importing Spreadsheets or CSV files (QGIS3)

Many times the GIS data comes in a table or an Excel spreadsheet. Also, if you have a list lat/long coordinates, you can easily import this data in your GIS project.

Overview of the task

We will be importing a text file of earthquake data to QGIS.

Get the data

NOAA's National Geophysical Data Center produces a great dataset of all significant earthquakes since 2150 BC. Learn more. (http://www.ngdc.noaa.gov/nndc/struts/form?t=101650&s=1&d=1)

Download Significant Earthquake Database (http://www.ngdc.noaa.gov/nndc/struts/results?type_0=Exact&query_0=$ID&t=101650&s=13&d=189&dfn=signif.txt) text file.

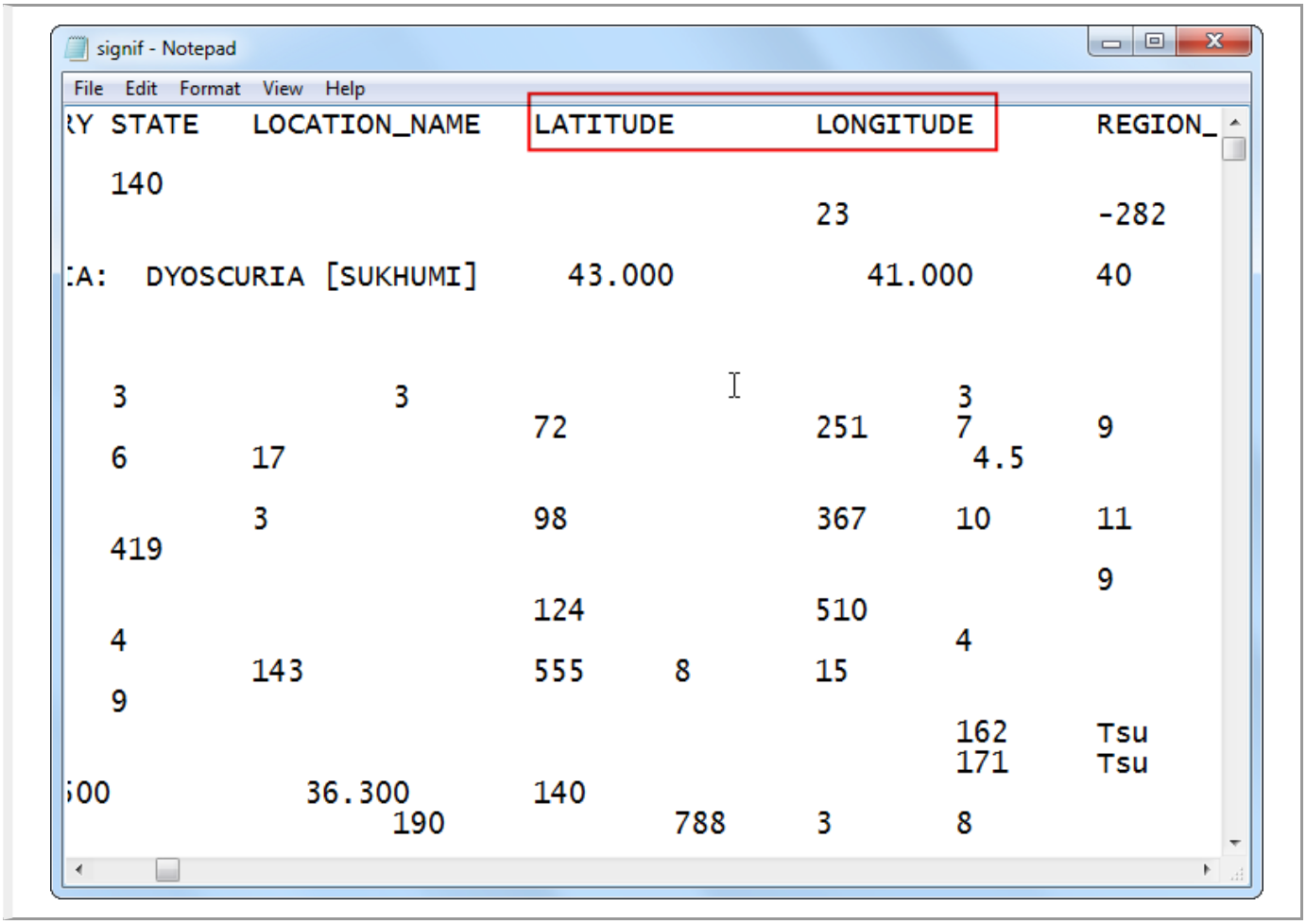
For convenience, you may directly download a copy of both the datasets from the links below:

signif.txt (http://www.qgistutorials.com/downloads/signif.txt)

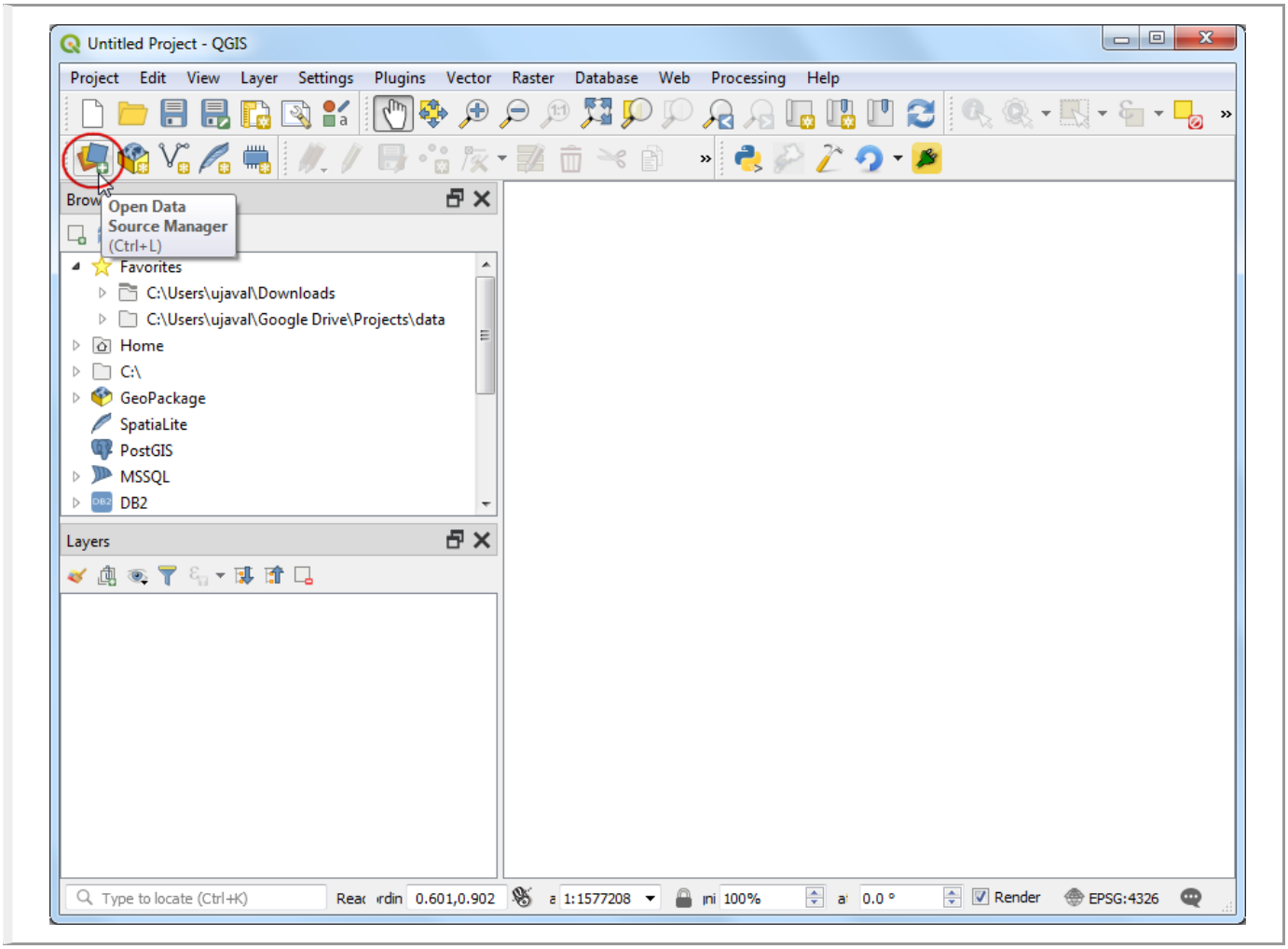
Data Source [NGDC] (../credits.html#ngdc)

Procedure

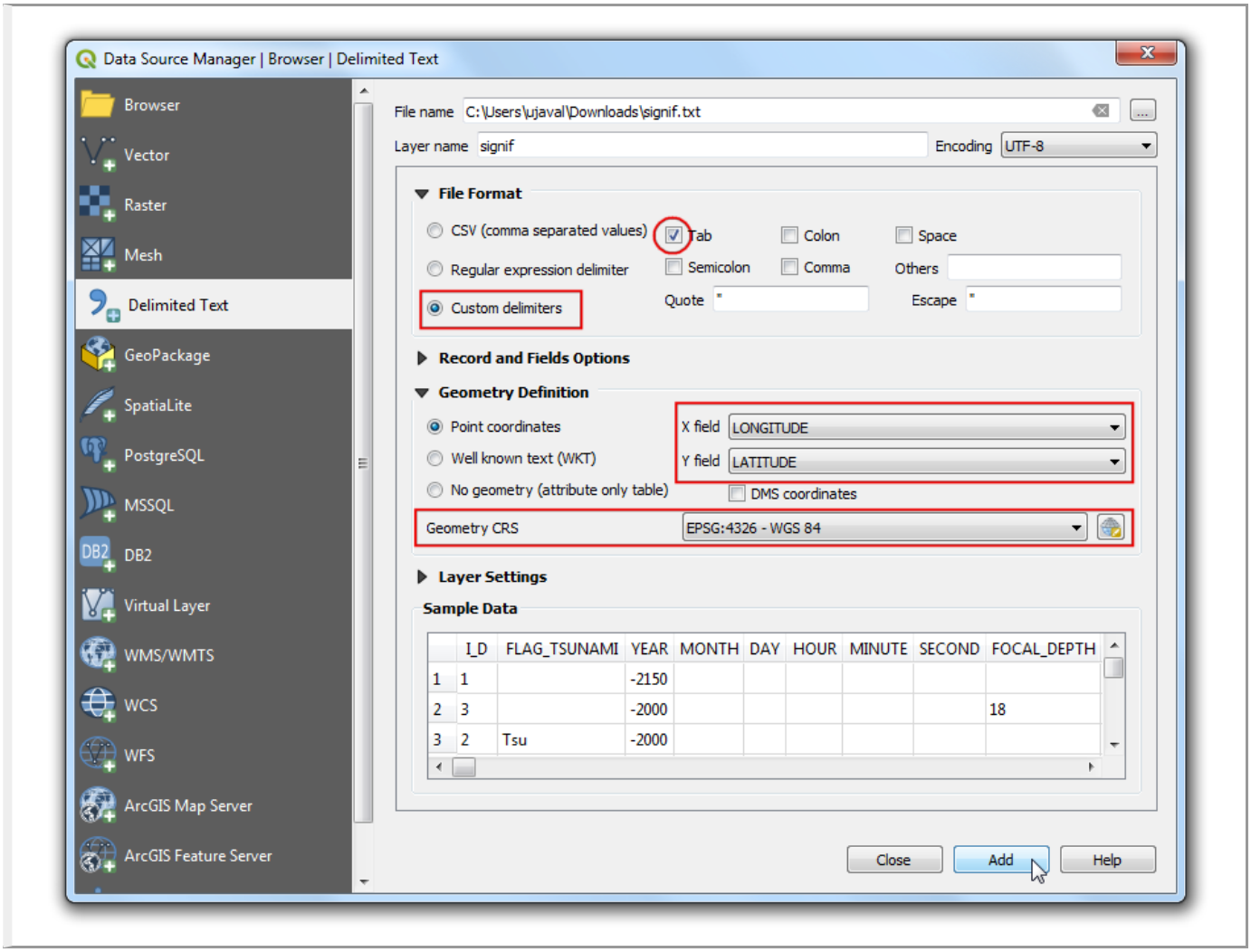
1. Examine your tabular data source. To import this data to QGIS, you will have to save it as a text file and need at least 2 columns which contain the X and Y coordinates. If you have a spreadsheet, use Save As function in your program to save it as a Tab Delimited File or a Comma Separated Values (CSV) file. Once you have the data exported this way, you can open it in a text editor such as Notepad to view the contents. In case of the Significant Earthquake Database, the data already comes as a text file which contains latitude and longitude of the earthquake centers along with other related attributes. You will see that each field is separated by a TAB.



2. QGIS3 comes with a unified data manager that allows you to load all the various supported data format. Click the Open Data Source Manager button on the Data Source Toolbar. You can also use `Ctrl + L` keyboard shortcut.



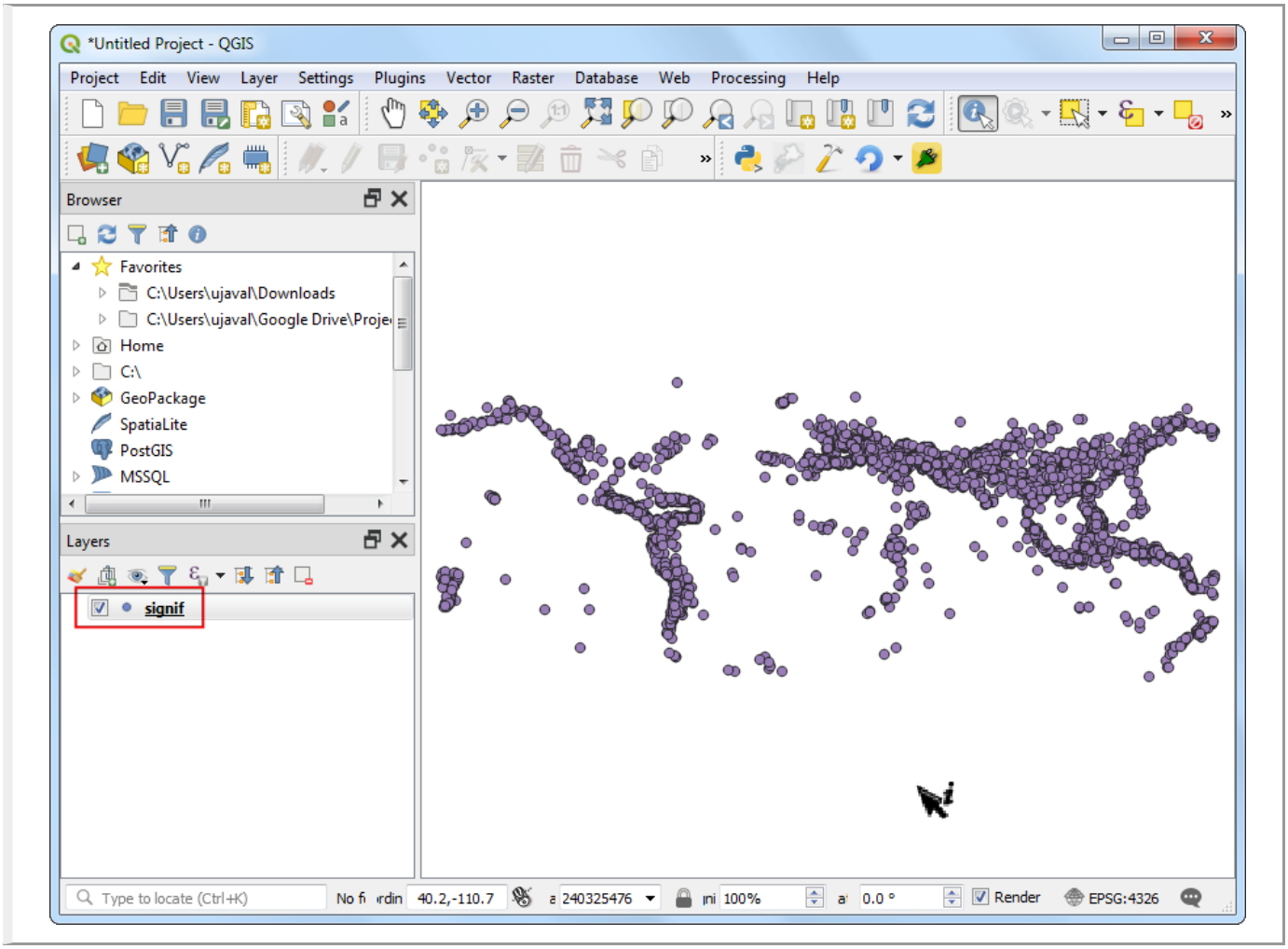
3. Switch to the Delimited Text tab. Click the ... button next to File name and specify the path to the text file you downloaded. In the File format section, select Custom delimiters and check Tab. The Geometry definition secction will be auto-populated if it finds a suitable X and Y coordinate fields. In our case they are `LONGITUDE` and `LATITUDE`. You may change it if the import selects the wrong fields. You can leave the Geometry CRS to the default `EPSG:4326 WGS84` CRS. If your file contained coordinates in a different CRS, you can select the appropriate CRS here. Click OK.



Note

It is easy to confuse X and Y coordinates. Latitude specifies the north-south position of a point and hence it is a **Y** coordinate. Similarly Longitude specifies the east-west position of a point and it is a **X** coordinate.

4. You will now see that the data will be imported and displayed in the QGIS canvas as a new layer called `signif`.



Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Basic Vector Styling (QGIS3)

To create a map, one has to style the GIS data and present it in a form that is visually informative. There are a large number of options available in QGIS to apply different types of symbology to the underlying data. In this tutorial, we will take a text file and apply different data visualization techniques to highlight spatial patterns in the data.

Overview of the task

We will take a CSV file containing the location of all power plants in the world and create a visualization showing distribution of renewable and non-renewable fuels used in these power plants.

Other skills you will learn

- Use expressions to group multiple attribute values into a single category

Get the data

World Resources Institute (https://www.wri.org) has compiled a comprehensive, open source database of power plants around the world covering over 30000 plants. Download the The Global Power Plant Database (http://datasets.wri.org/dataset/globalpowerplantdatabase) from the WRI Open Data Portal.

Natural Earth (http://naturalearthdata.com) has several global vector layers. Download the 10m Physical Vectors - Land (https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/physical/ne_10m_land.zip) containing Land polygons.

For convenience, you may directly download a copy of the above layers from below:

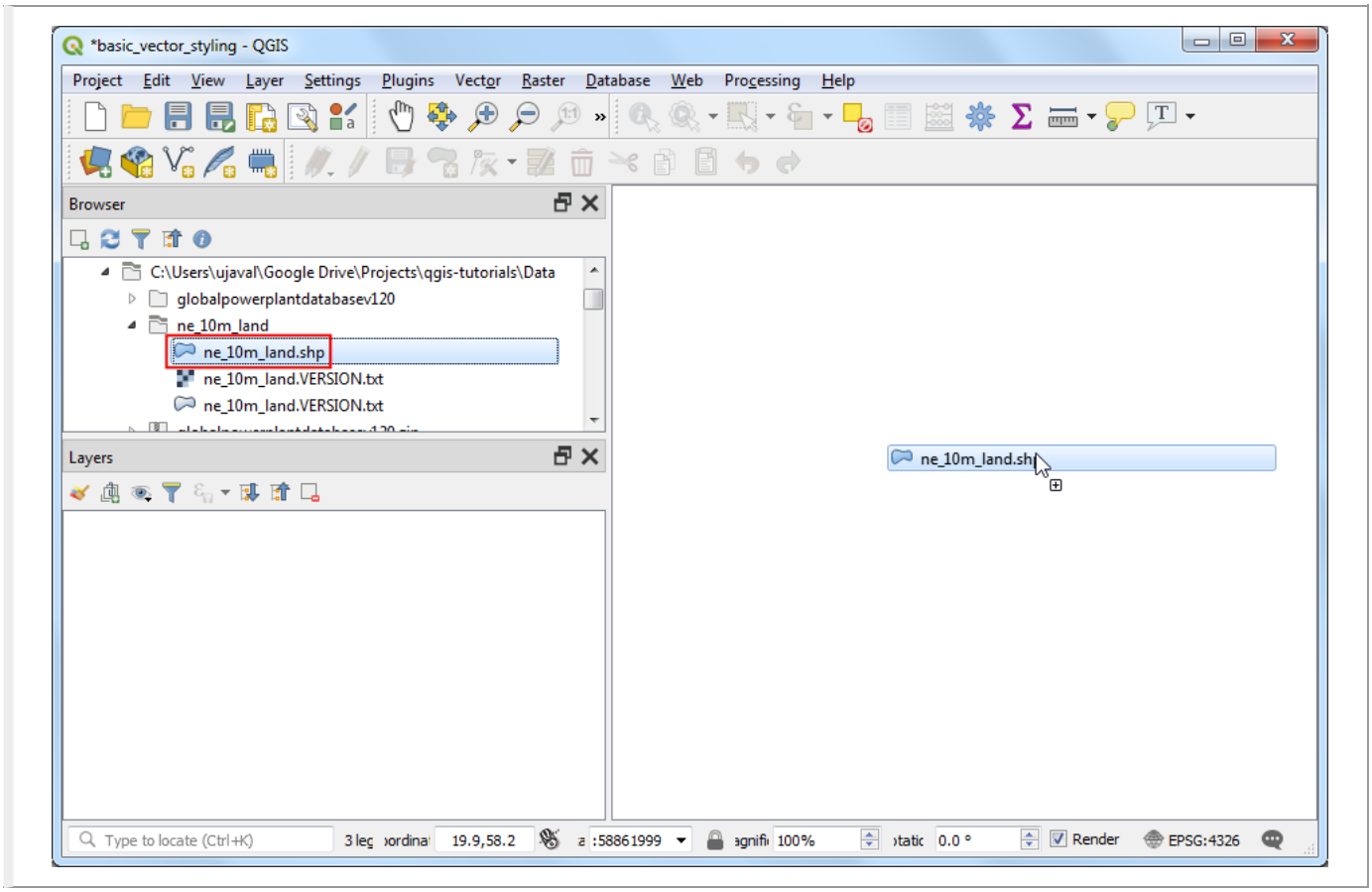
globalpowerplantdatabasev120.zip (http://www.qgistutorials.com/downloads/globalpowerplantdatabasev120.zip)

ne_10m_land.zip (http://www.qgistutorials.com/downloads/ne_10m_land.zip)

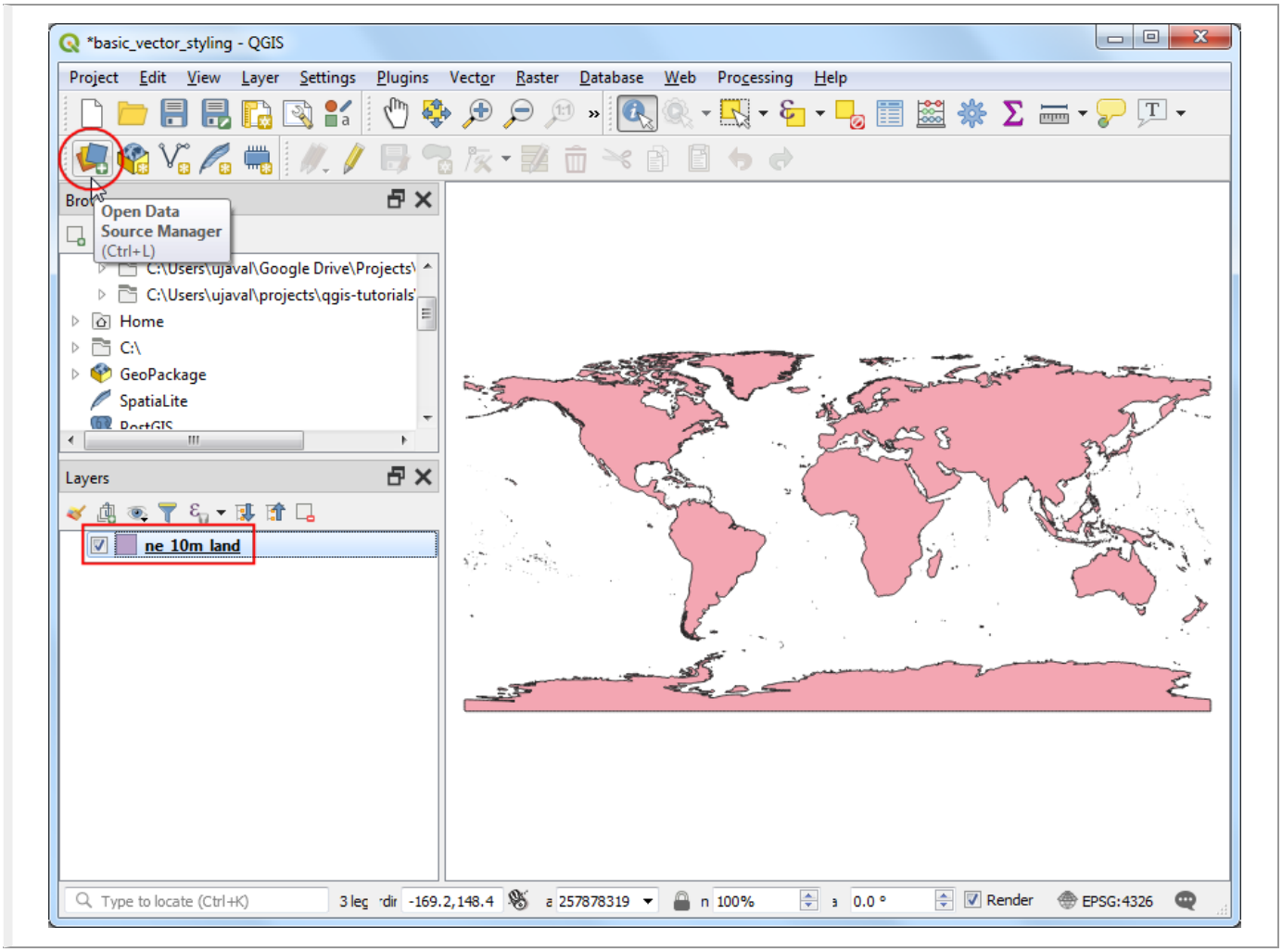
Data Source [WRI] (../credits.html#wri) [NATURALEARTH] (../credits.html#naturalearth)

Procedure

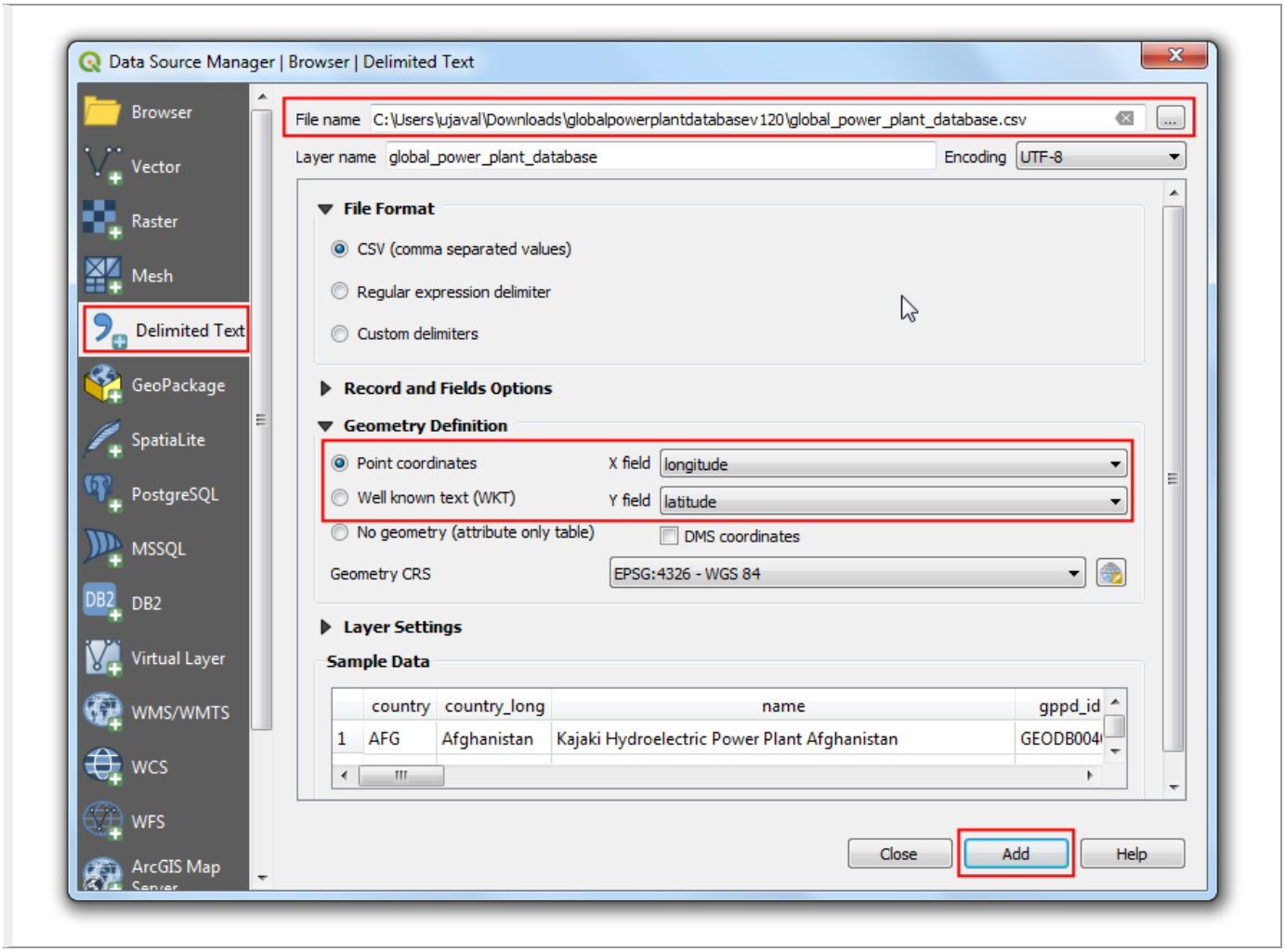
1. Unzip both the datasets to a folder on your computer. In the QGIS Browser Panel, locate the directory where you extracted the data. Expand the `ne_10m_land` folder and select the `ne_10m_land.shp` layer. Drag the layer to the canvas.



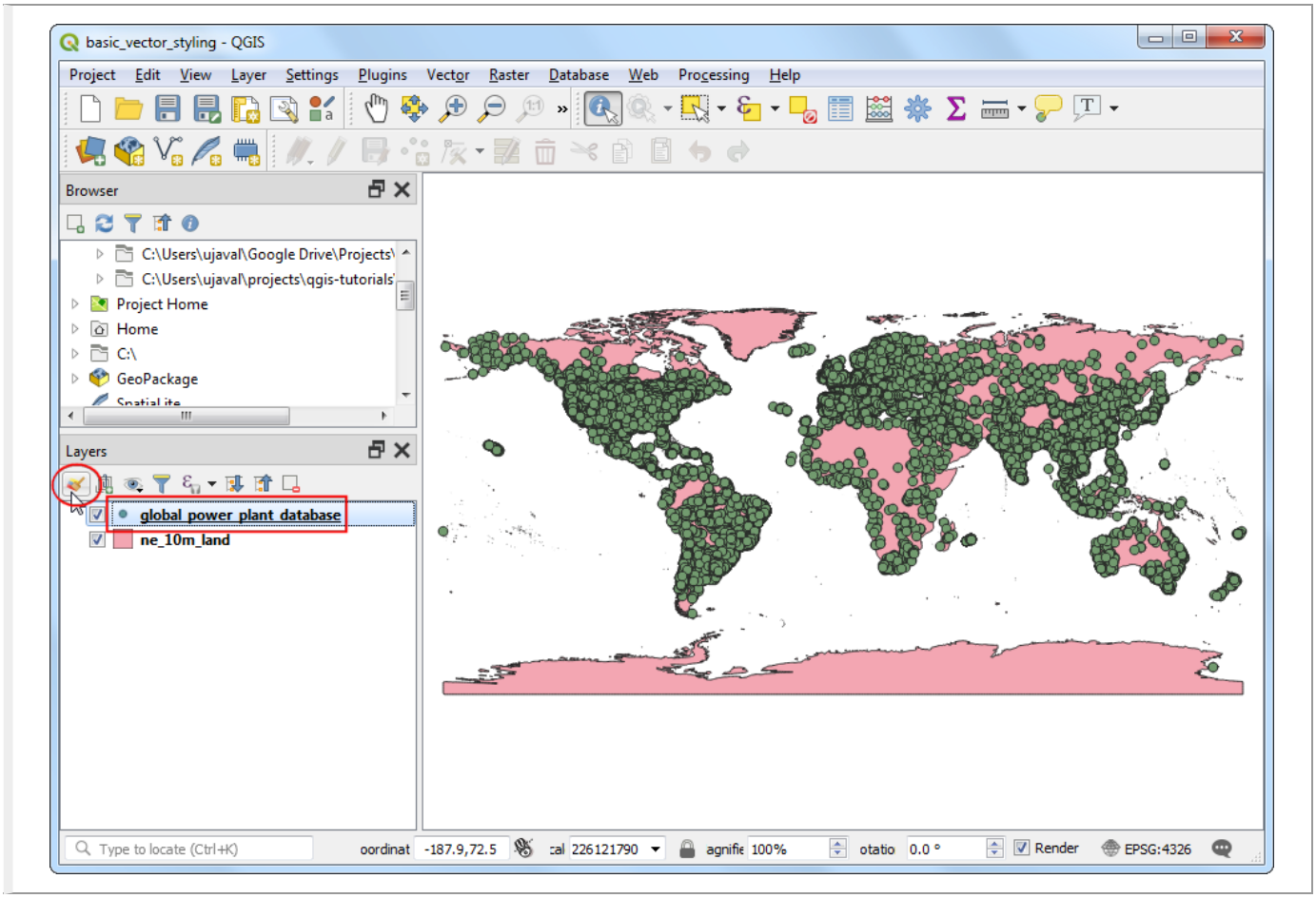
2. You will a new layer `ne_10m_land` added to the Layers panel. The global power plant database comes as a CSV file, so we will need to import it. Click the Open Data Source Manager button on the Data Source Toolbar. You can also use `Ctrl + L` keyboard shortcut.



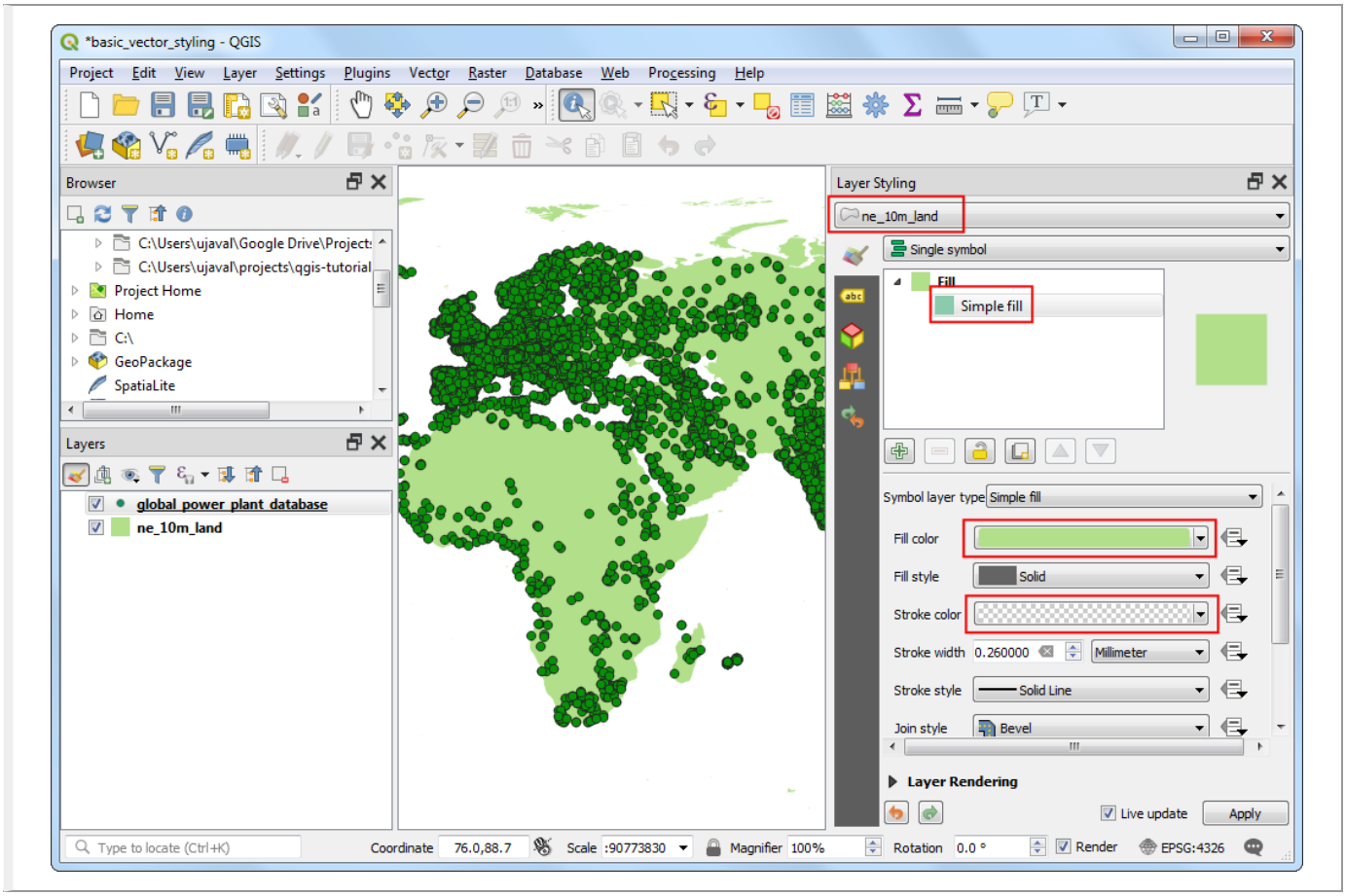
3. In the Data Source Manager window, switch to the Delimited Text tab. Click the ... button next to File name and browse to the directory where you extracted the `globalpowerplantdatabasev120.zip` file. Select the `global_power_plant_database.csv`. QGIS will auto detect the delimiter and geometry fields. Leave the Geometry CRS to the default value of `EPSG:4326 - WGS84`. Click Add followed by Close.



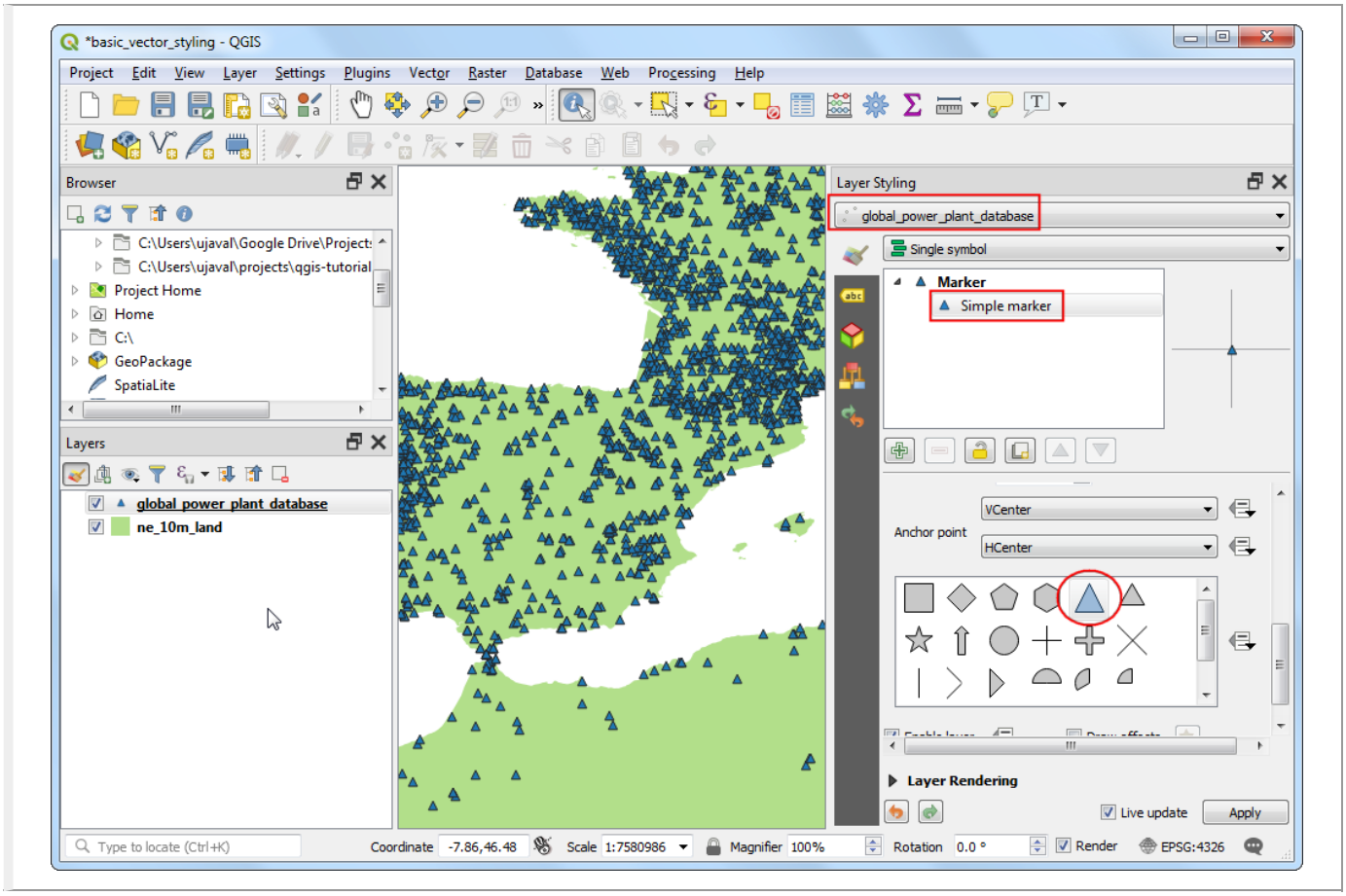
4. A new layer `global_power_plant_database` will be added to the Layers panel and you will see the points representing the power plants in the canvas. Now we are ready to style both these layers. Click the Open the Layer Styling panel button at the top of the Layers panel.



5. The Layer Styling panel will open on the right. Select the `ne_10m_land` layer first. This will be our base layer so we can keep the styling minimalistic so it is not distracting. Click `Simple fill` and scroll down. Select a Fill color as per your liking. Click the drop-down next to Stroke color and select `Transparent Stroke` . This will set the outlines of the land polygons to be transparent. You will see the result of your selection applied immediately to the layer.

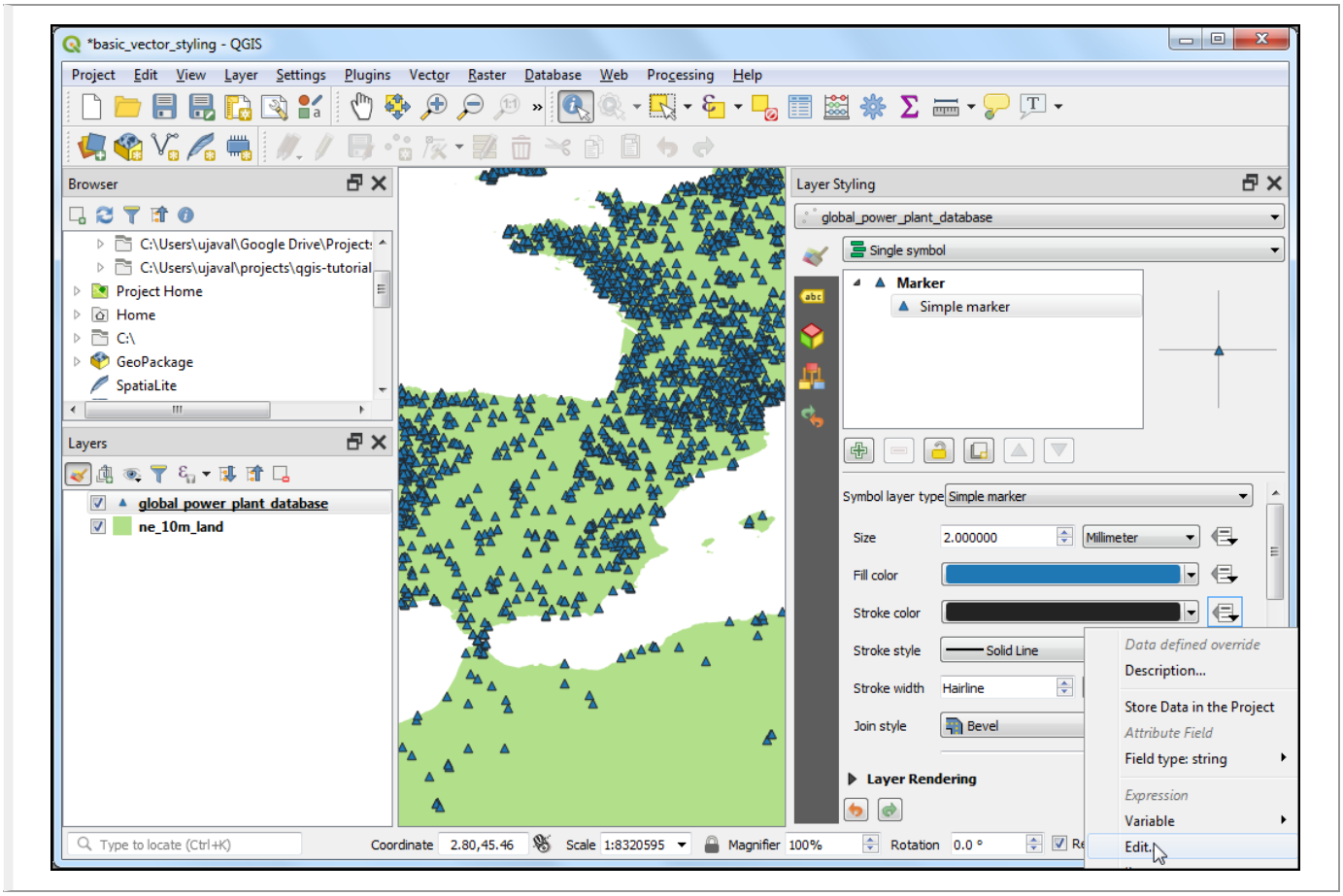


6. Next select the `global_power_plant_database` layer. Click on `Simple marker` and scroll down. Pick a triangle marker.



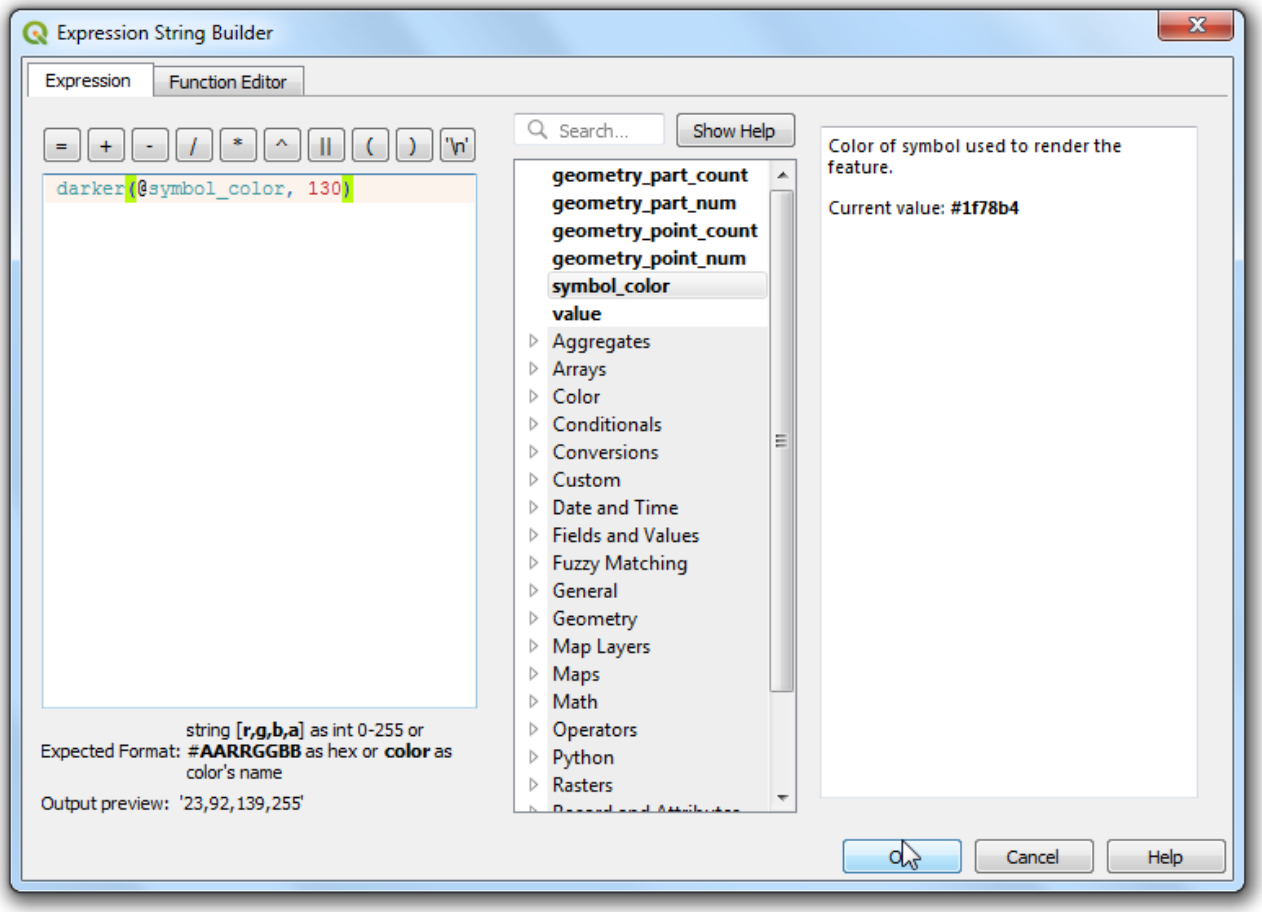
7. Scroll up and select a Fill color of your liking. A useful cartographic technique is to choose a slightly darker version of the fill-color as the Stroke color. Rather than trying to pick that manually, QGIS provides

an expression to control this more precisely. Click the Data defined override button and choose Edit.



8. Enter the following expression to set the color to be 30% darker shade than the fill color and click OK.

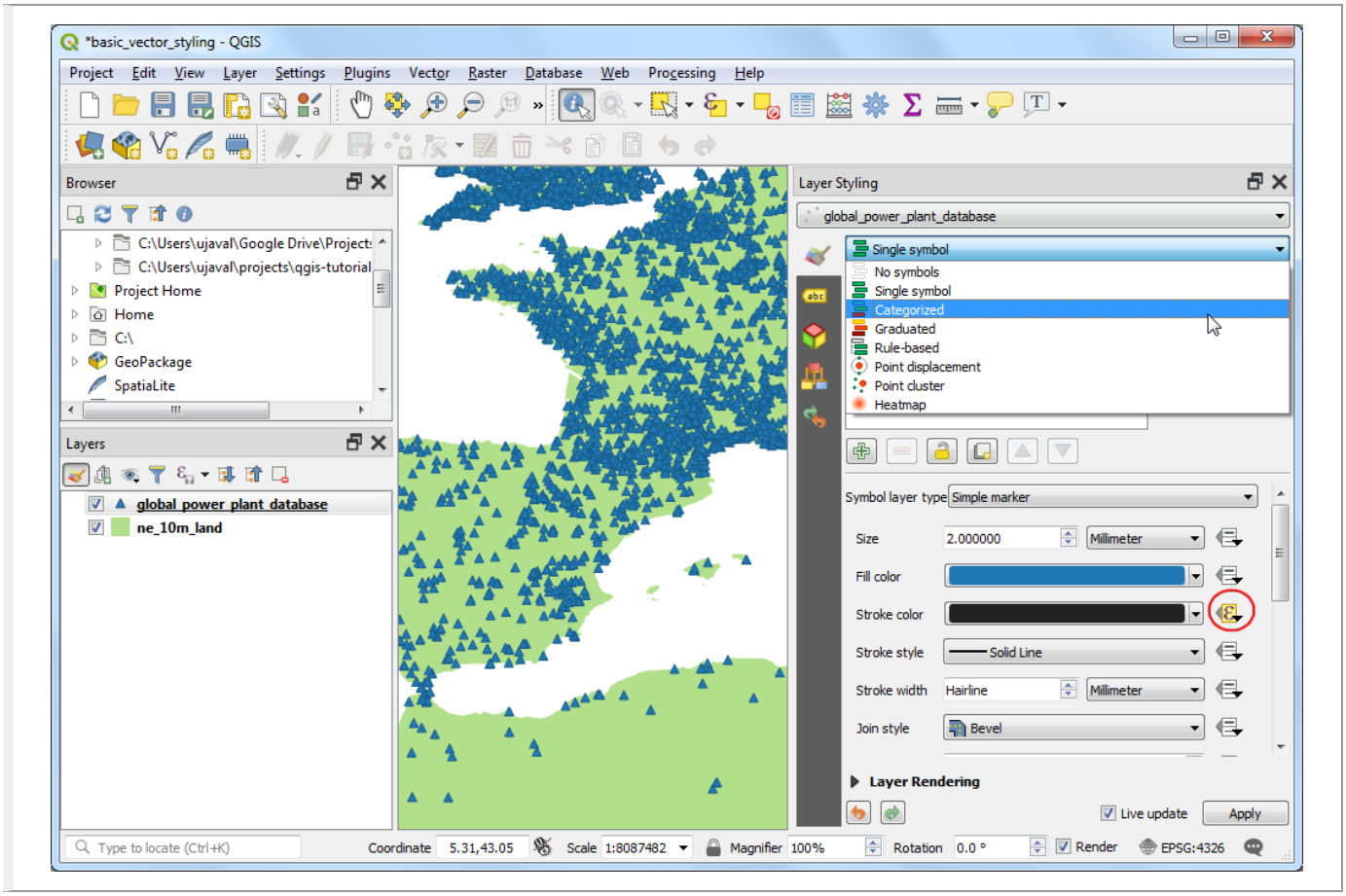
```
darker(@symbol_color, 130)
```



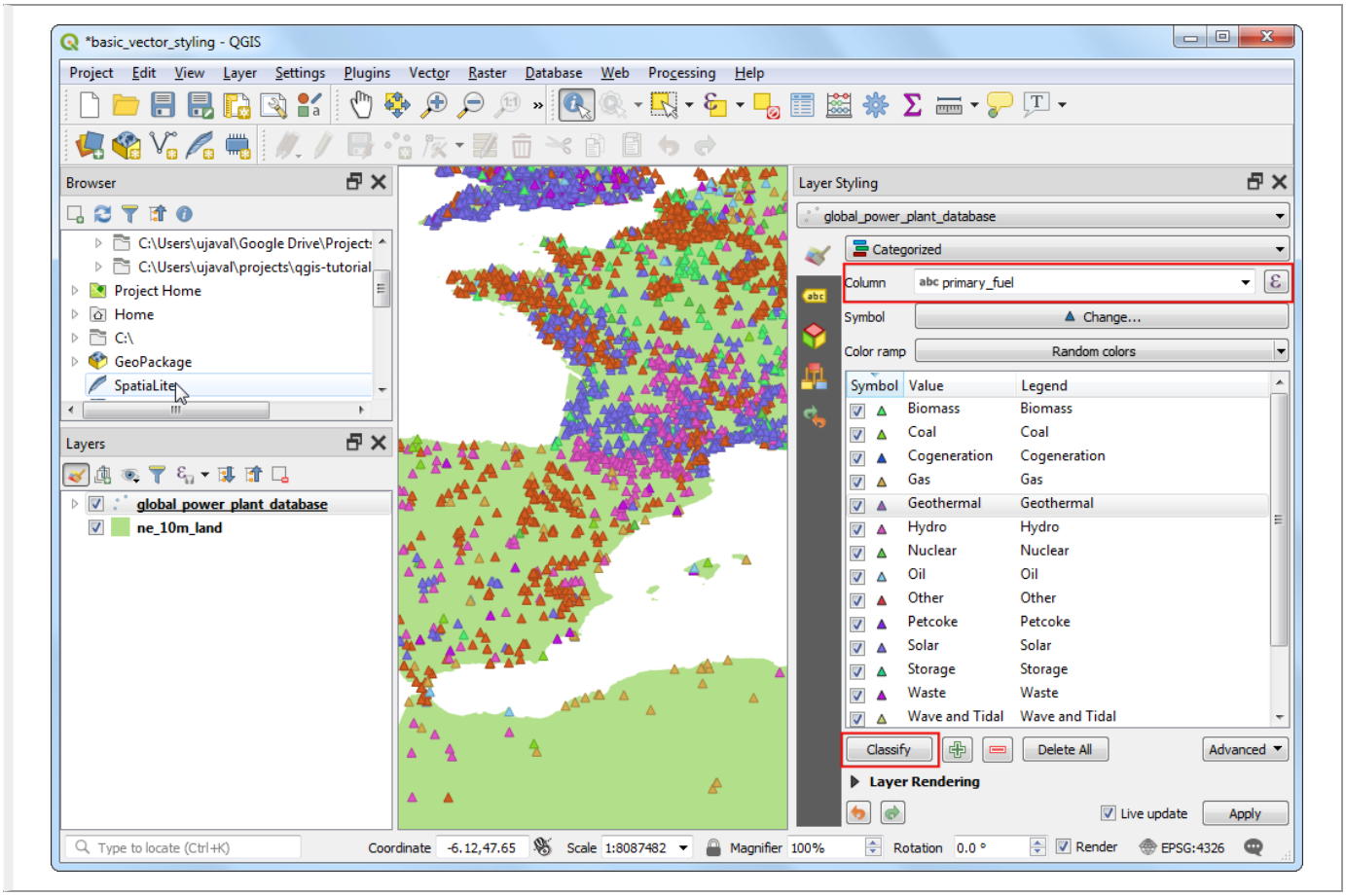
Note

Note that this expression is independent of the fill color you have chosen. You will see that this is immensely useful in the following steps where it automatically sets the border color based on the fill color provided.

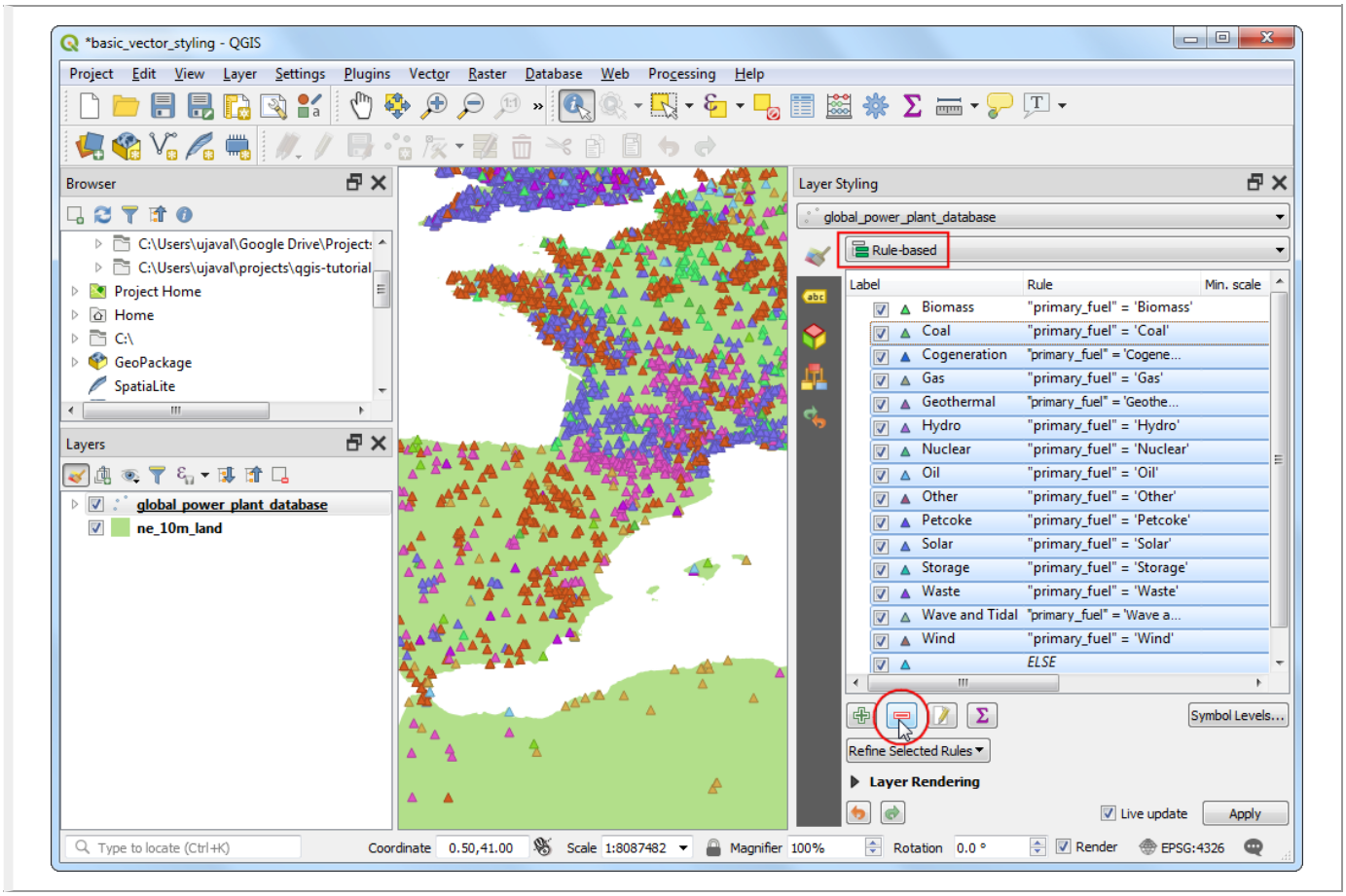
9. You will notice that the Data defined override button next to Stroke color has turned yellow - indicating than this property is controlled by an override. A single symbol rendering of the power plants layer is not very useful. It doesn't convey much information except the locations of the power plants. Let's use a different renderer to make it more useful. Click the Symbology drop-down and select `Categorized` renderer.



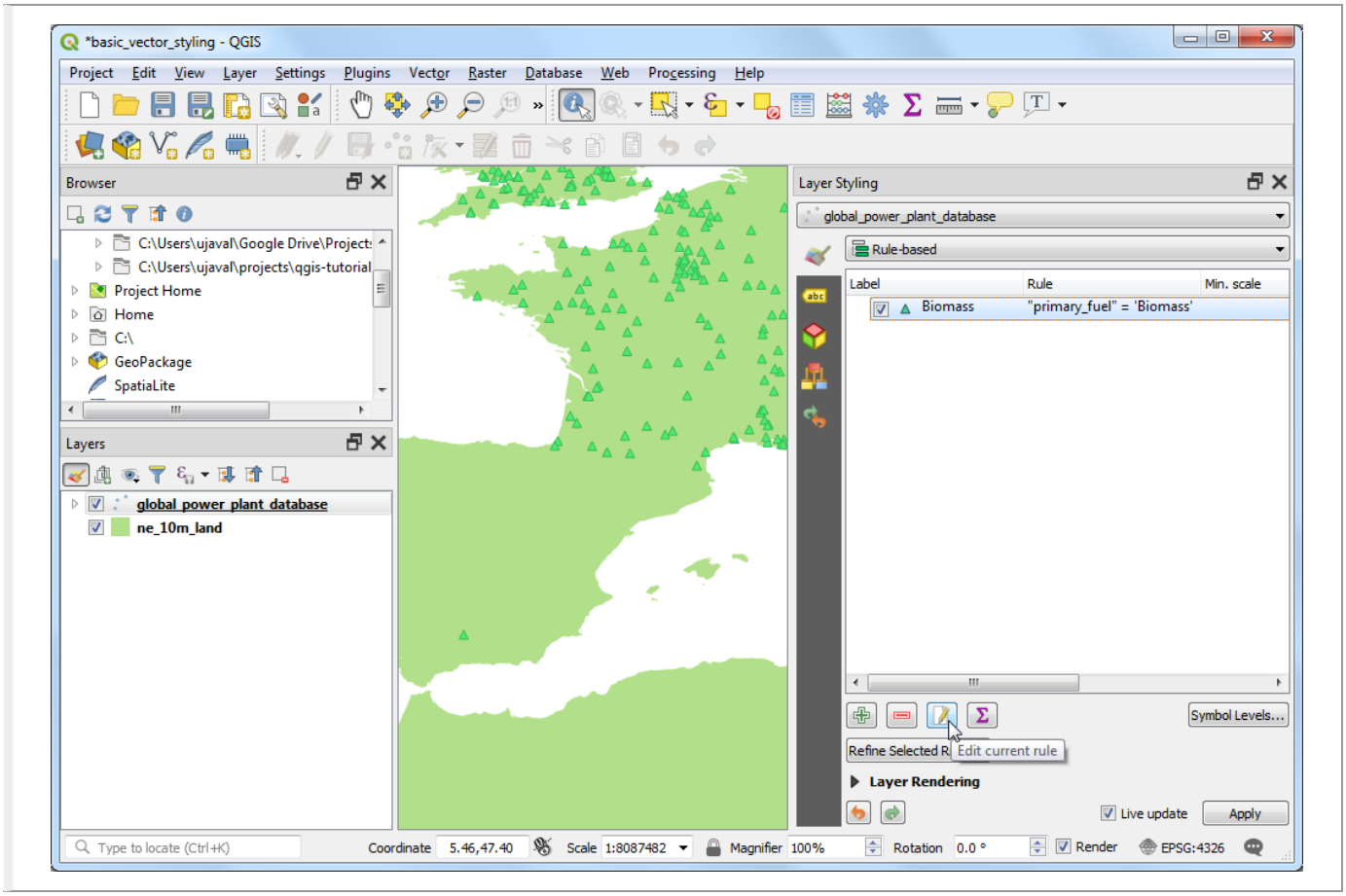
10. The `global_power_plant_database` layer contains an attribute indicating the primary fuel used in each power plant. We can create a style where each unique fuel type is shown in a different color. Select `primary_fuel` as the Column. Click Classify. You will multiple categories appear and the map rendering change accordingly.



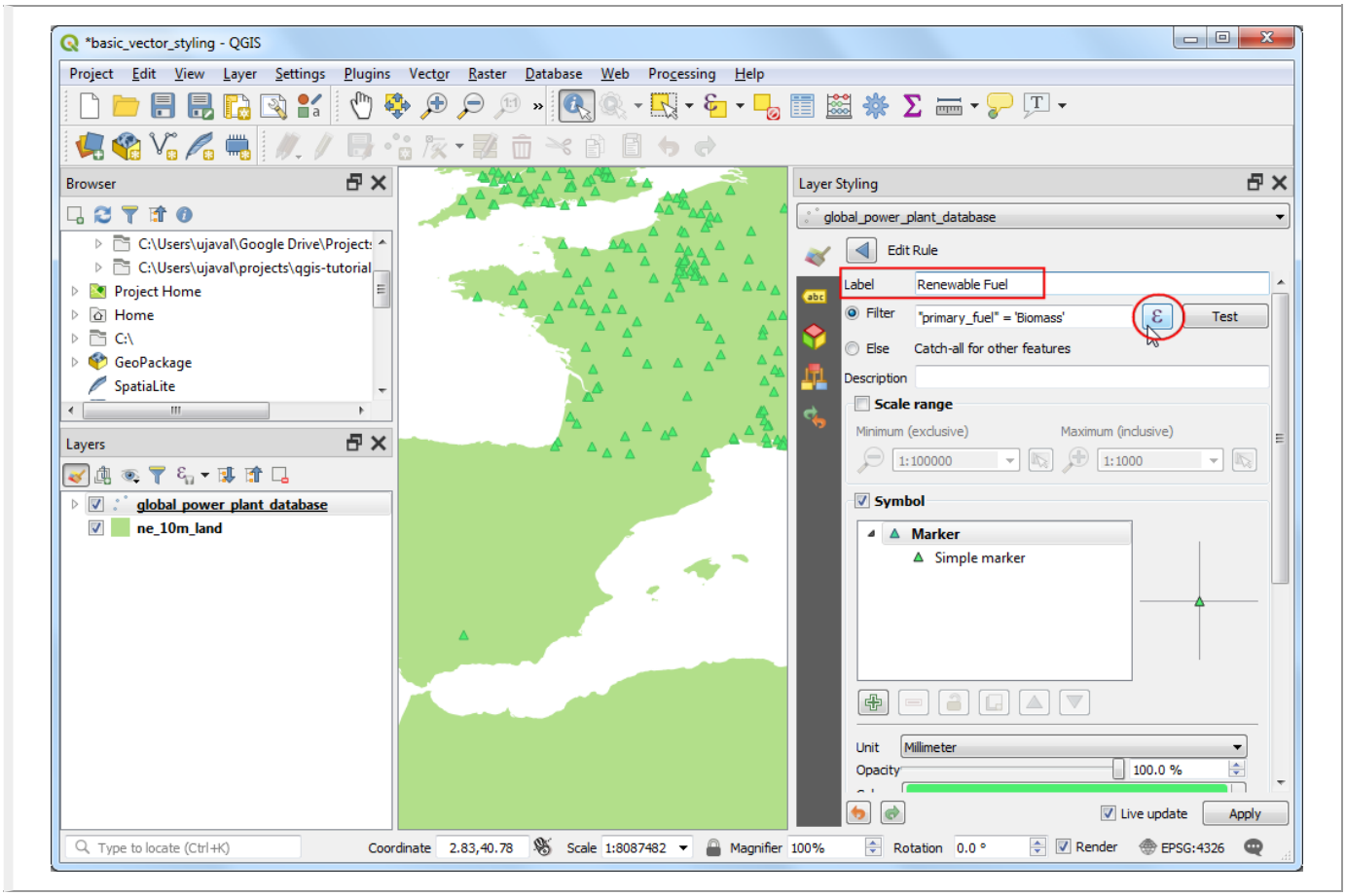
11. While a Categorized view is useful, this layer contains too-many categories for one to meaningfully interpret the map. A better approach would be to group certain type of fuel categories and reduce the number of classes. Let's try to create 3 categories - **Renewable fuel**, **Non-renewable fuel** and **Other**. Select `Rule-based` renderer. Select all but one rules by holding the `Ctrl` key and clicking on each row. Once selected, click the Remove selected rules button to delete them.



12. Select the remaining rule and click Edit current rule.

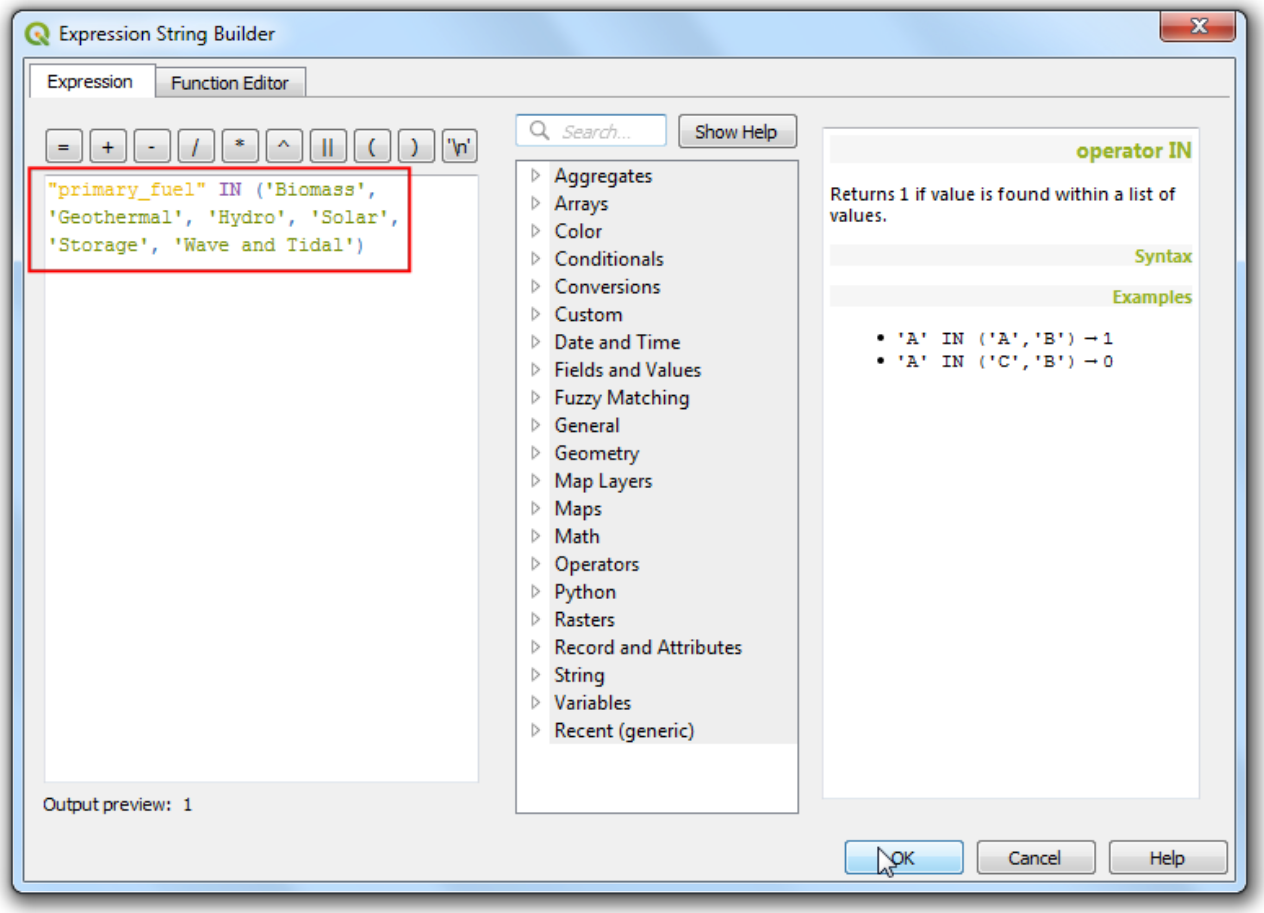


13. Enter `Renewable fuel` as the Label. Click the Expression button next to Filter.



14. In the Expression String Builder dialog, enter the following expression and click OK. Here we are grouping multiple renewable energy categories into a single category.

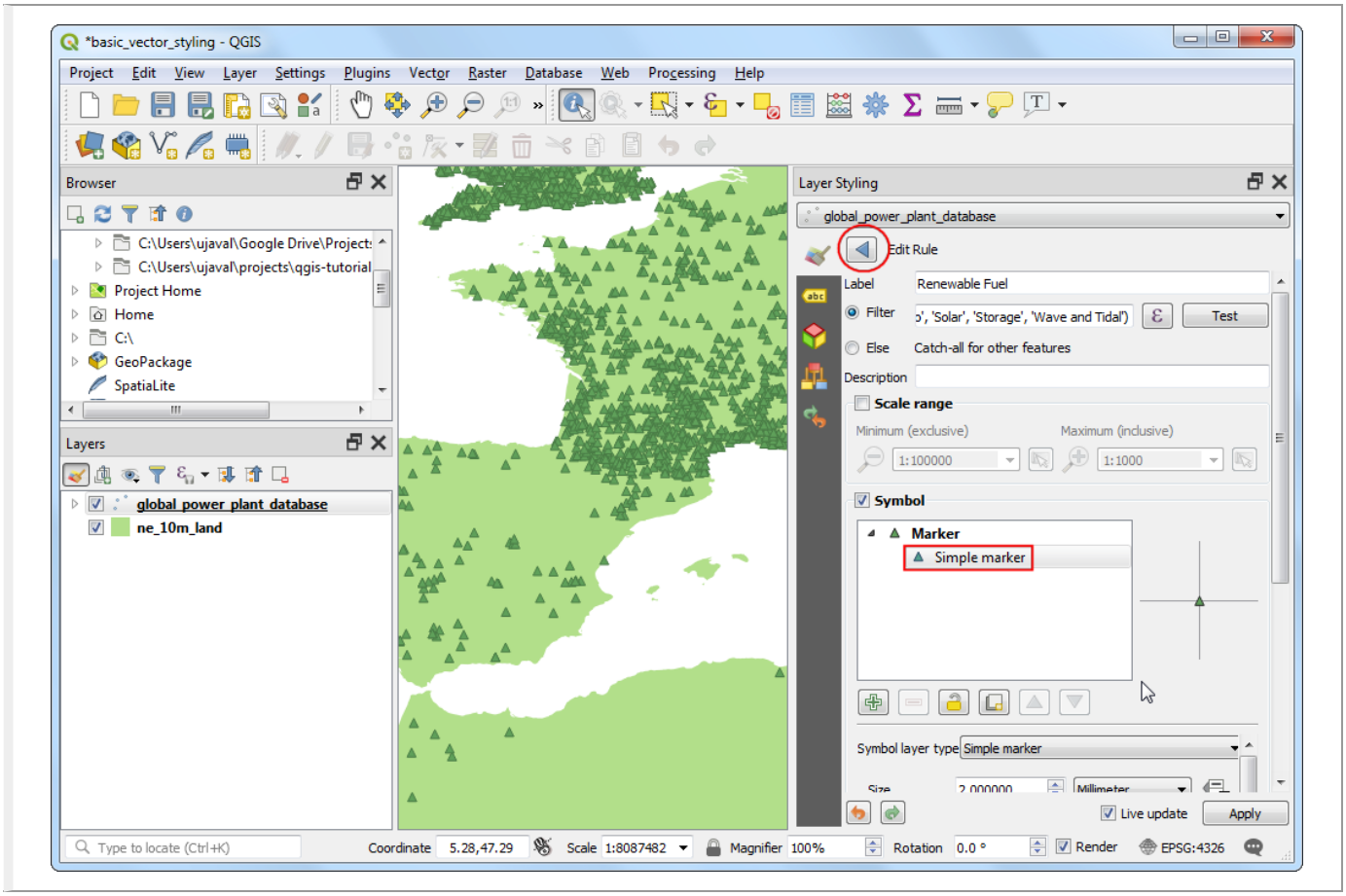
```
"primary_fuel" IN ('Biomass', 'Geothermal', 'Hydro', 'Solar', 'Wind', 'Storage', 'Wave and
```



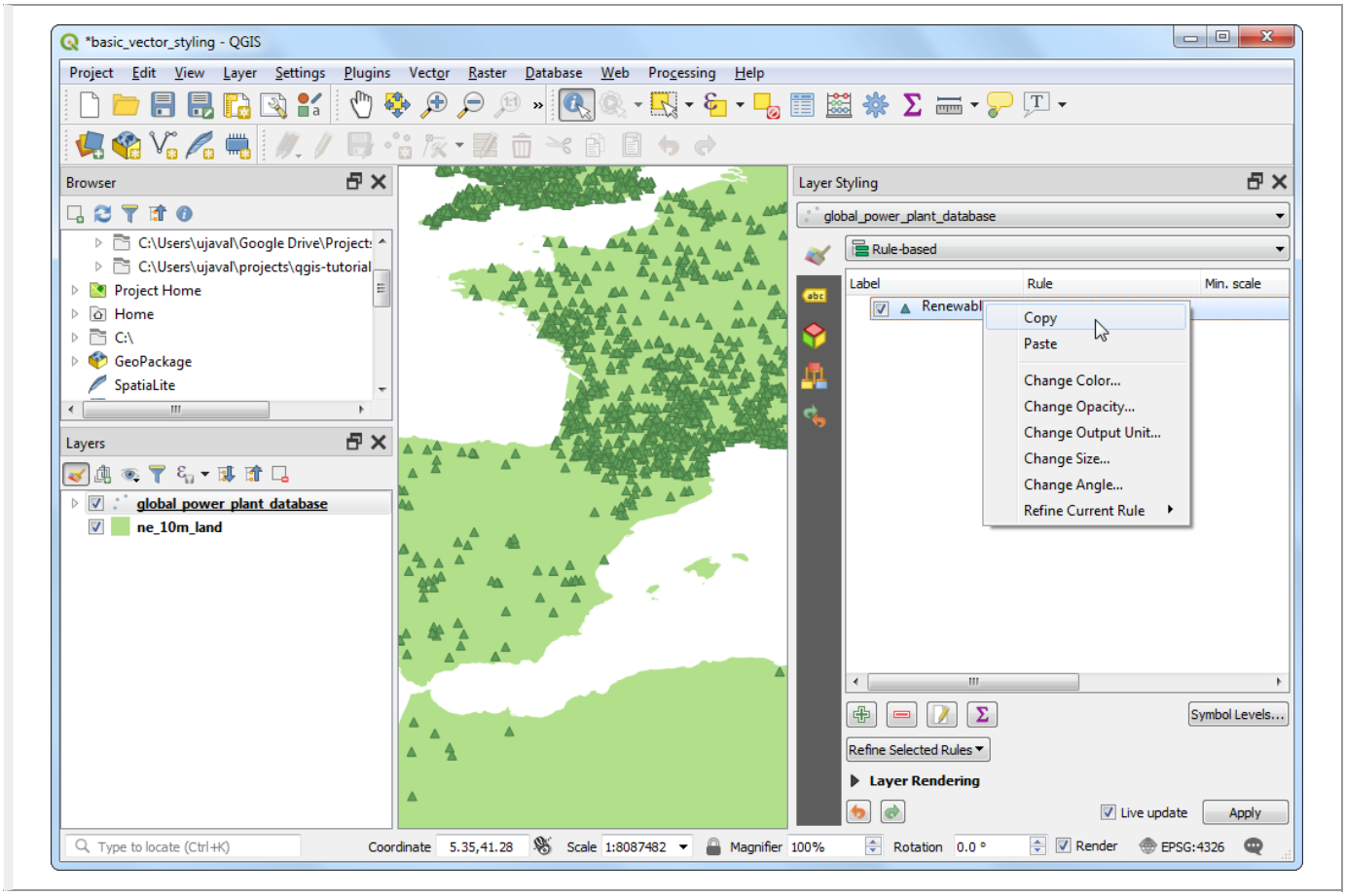
Note

The types of fuel chosen for renewable vs. non-renewable categories is based on Wikipedia (https://en.wikipedia.org/wiki/Renewable_energy). There are alternate definitions and classifications that may not match what is chosen here.

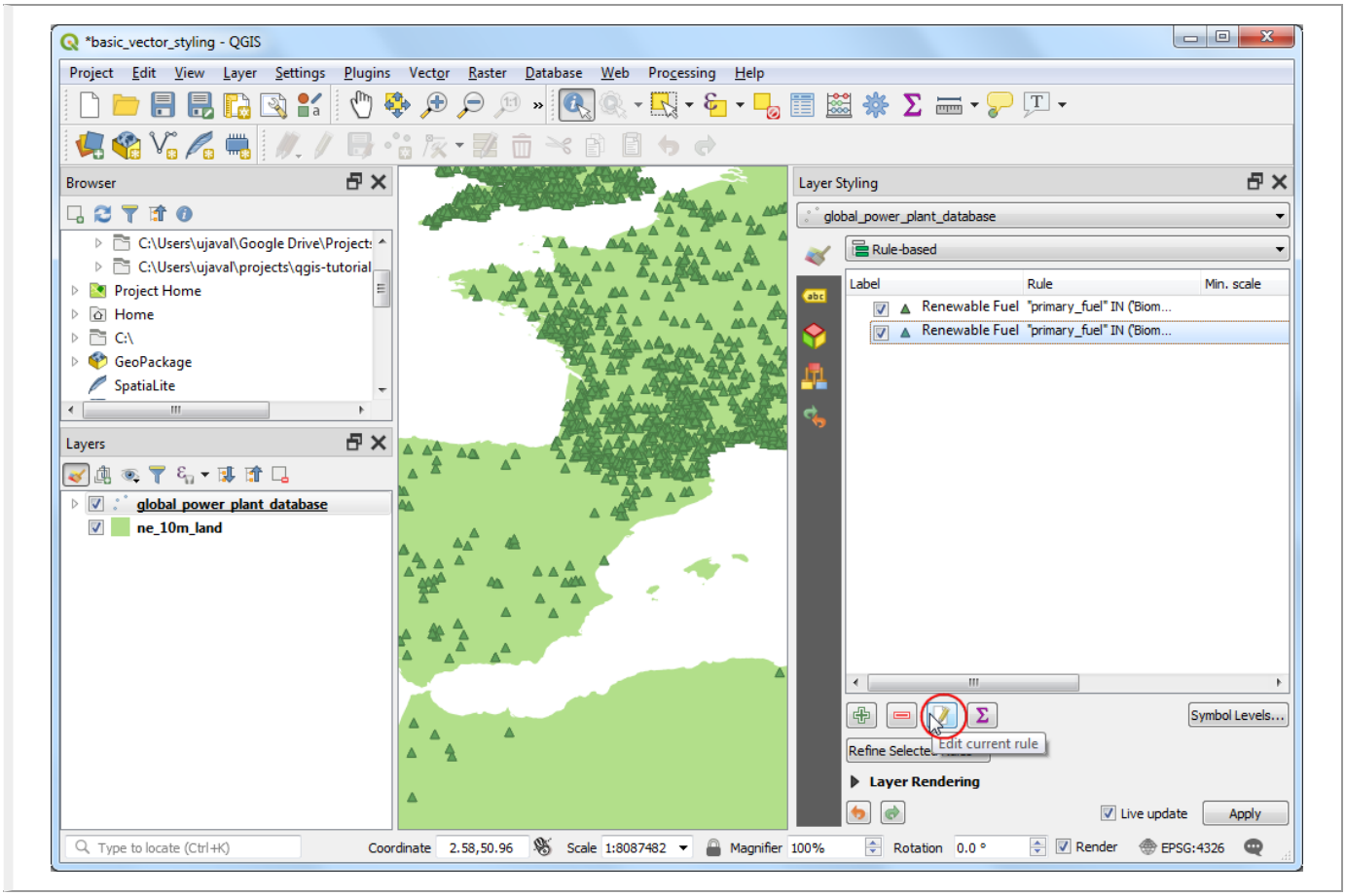
15. Scroll down and click Simple marker. Choose an appropriate Fill color. Once done, click the Back button.



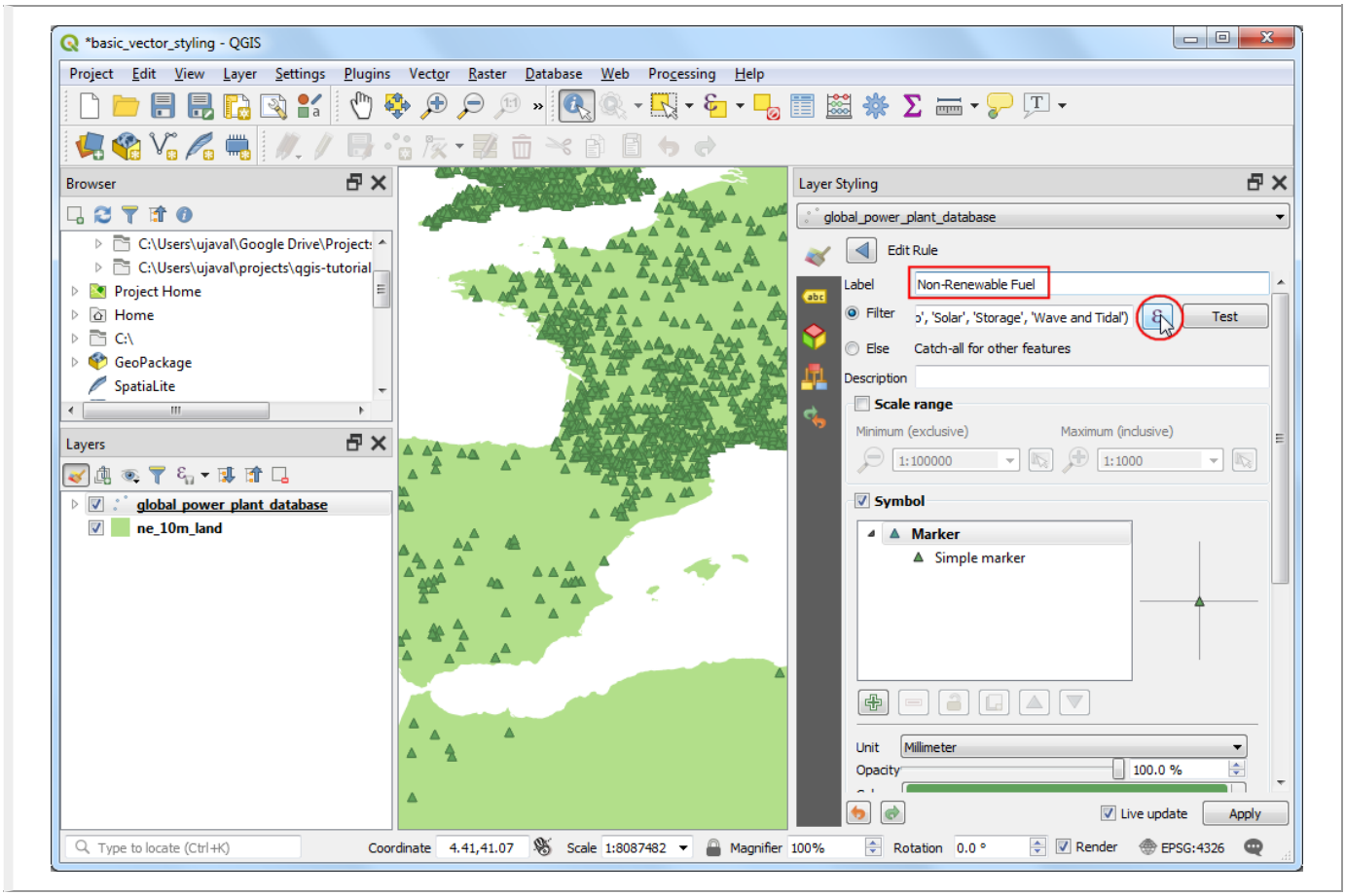
16. You will see a single rule being applied to the layer for the Renewable fuel category. Right-click the row and choose Copy. Right-click again and choose Paste.



17. A copy of the existing rule will be added. Select the newly added row and click Edit current rule.

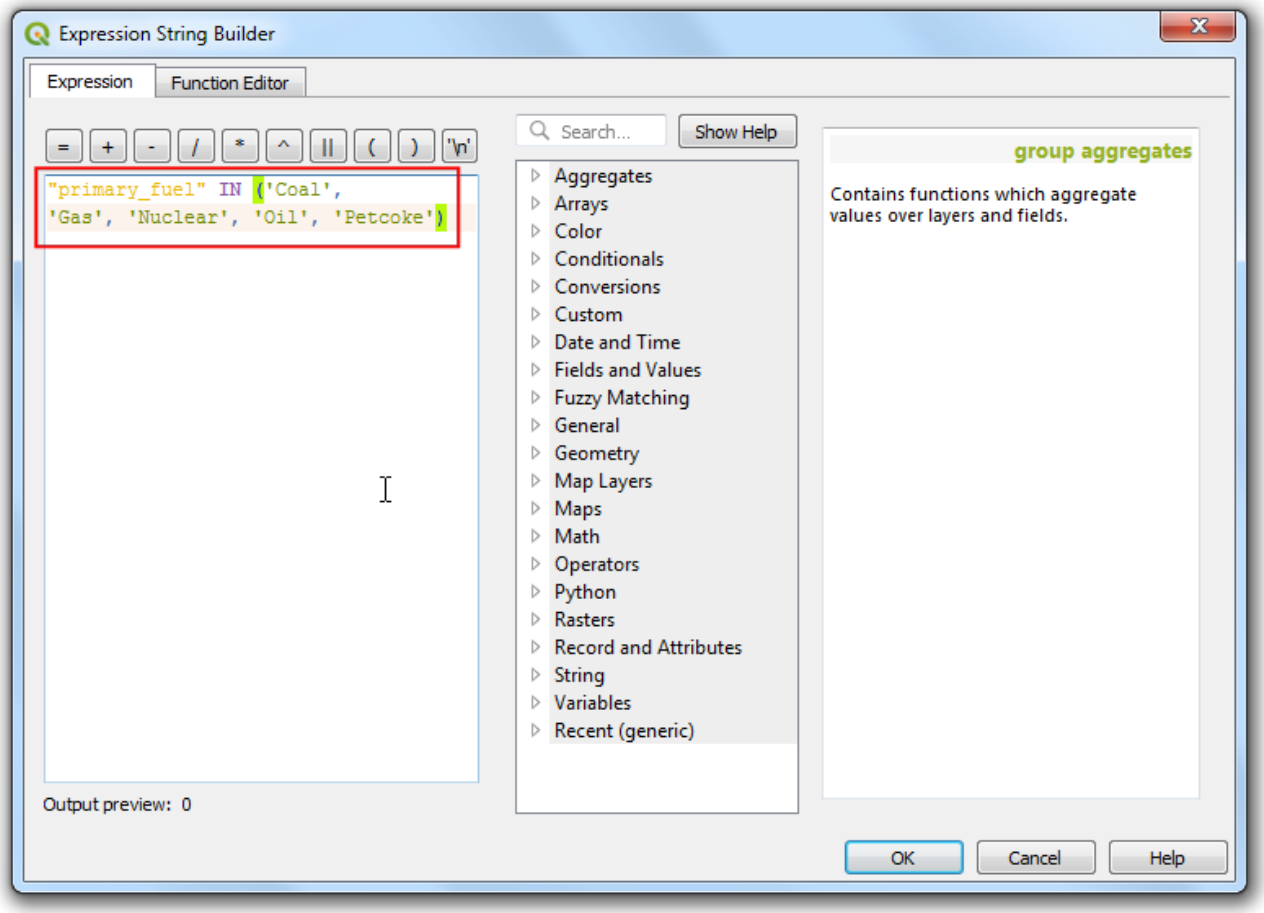


18. Enter `Non-renewable fuel` as the Label. Click the Expression button next to Filter.

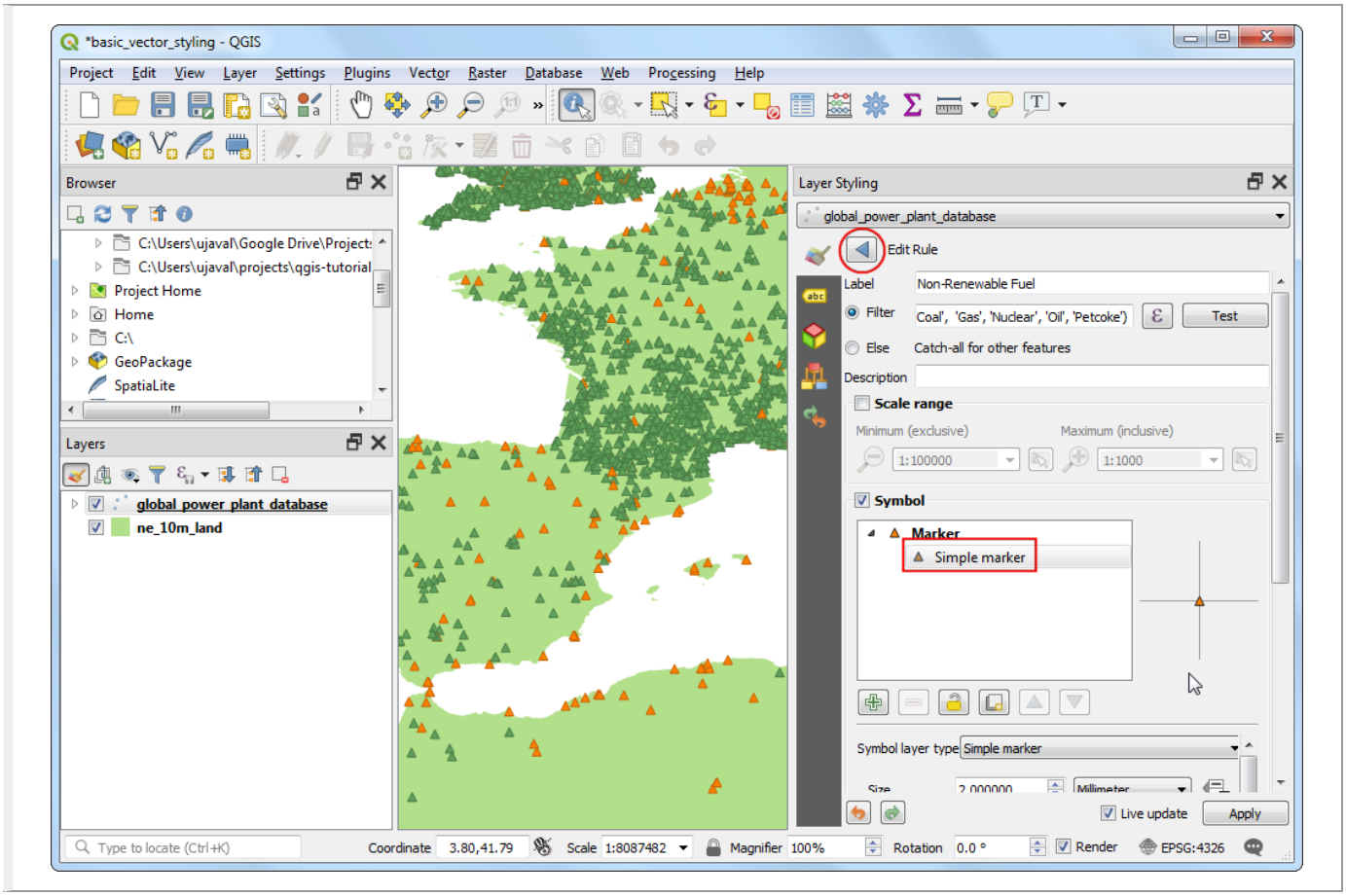


19. In the Expression String Builder dialog, enter the following expression and click OK.

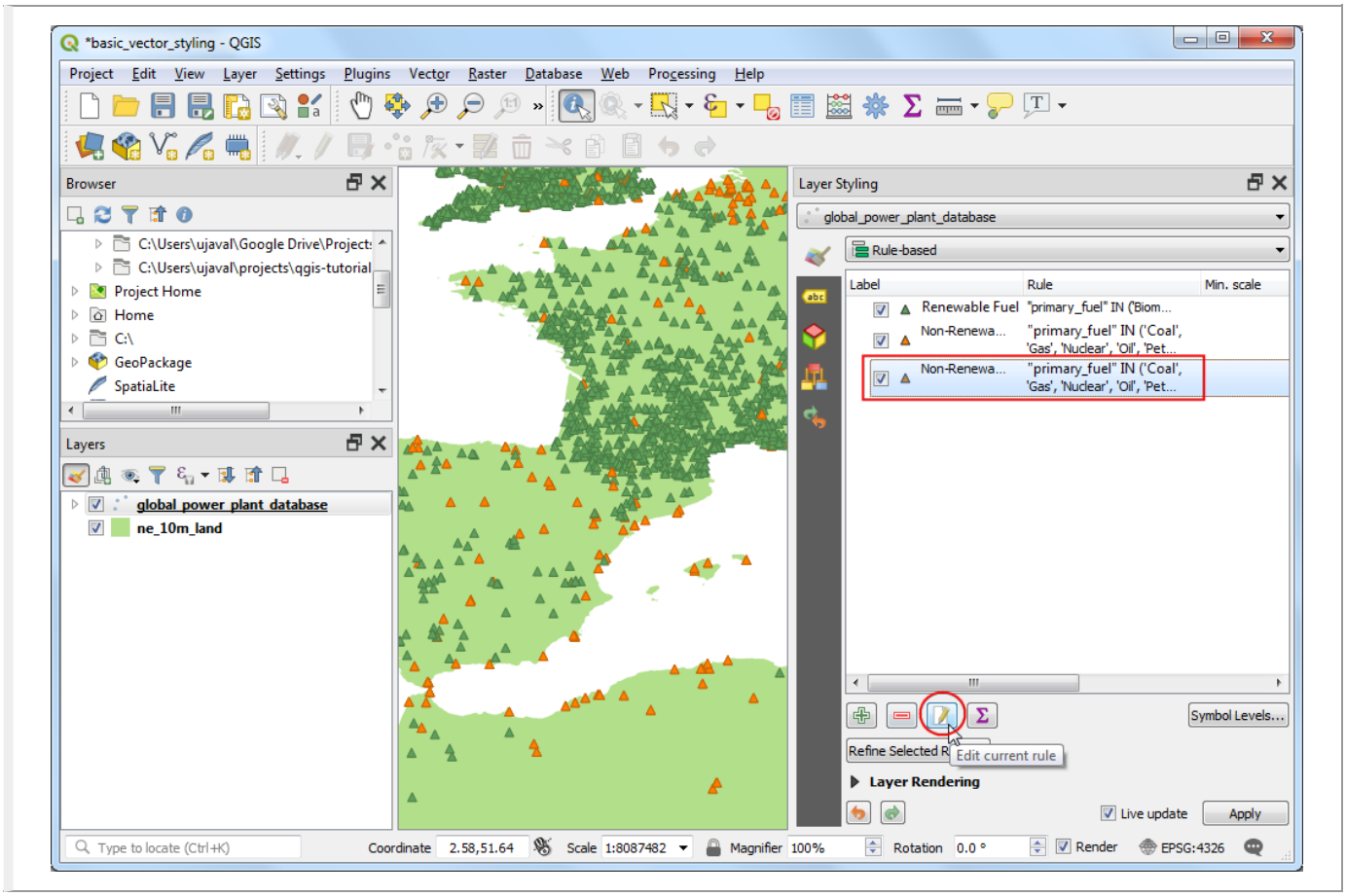
```
"primary_fuel" IN ('Coal', 'Gas', 'Nuclear', 'Oil', 'Petcoke')
```



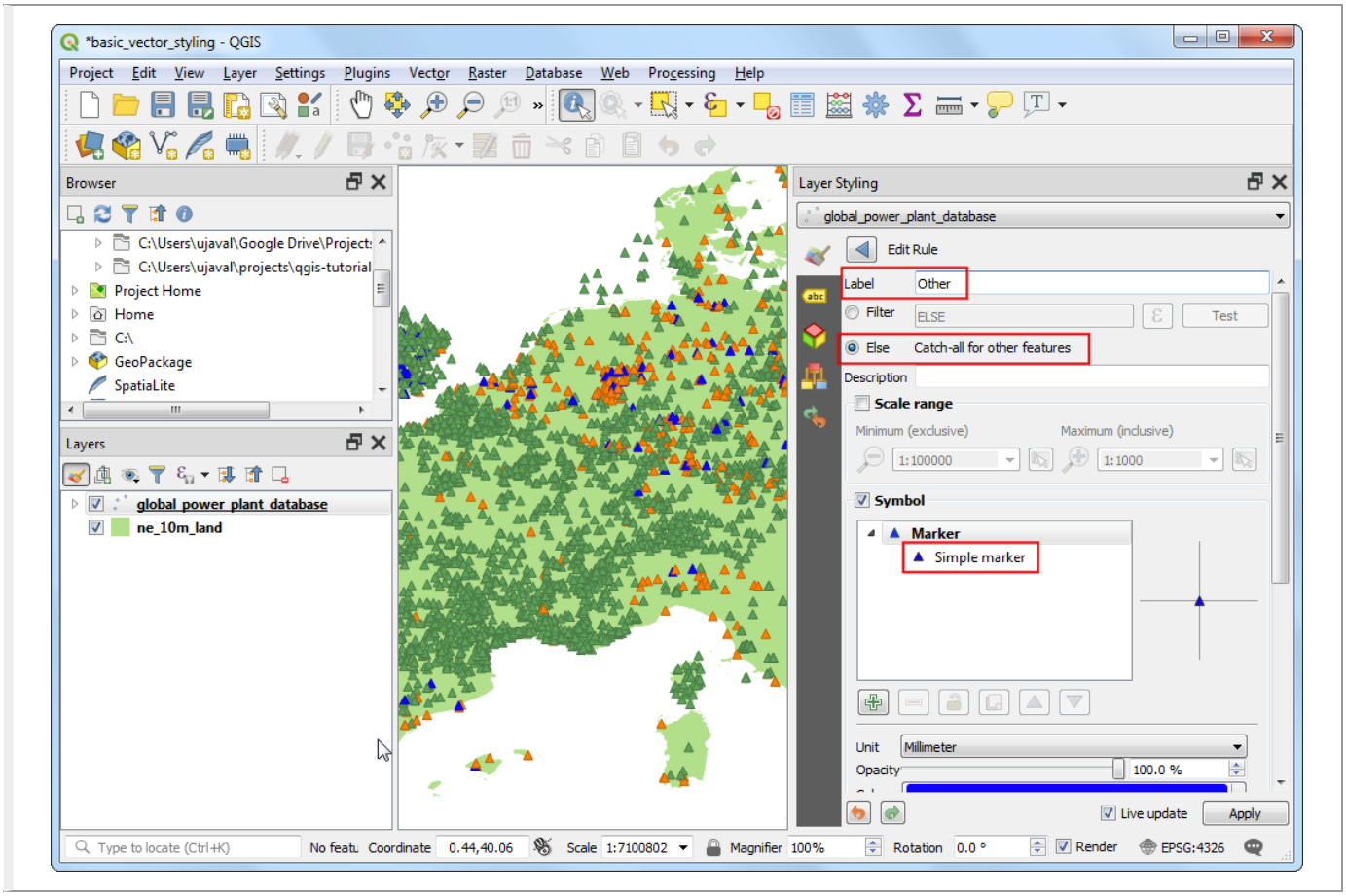
20. Scroll down and click Simple marker. Choose an appropriate Fill color. Once done, click the Back button.



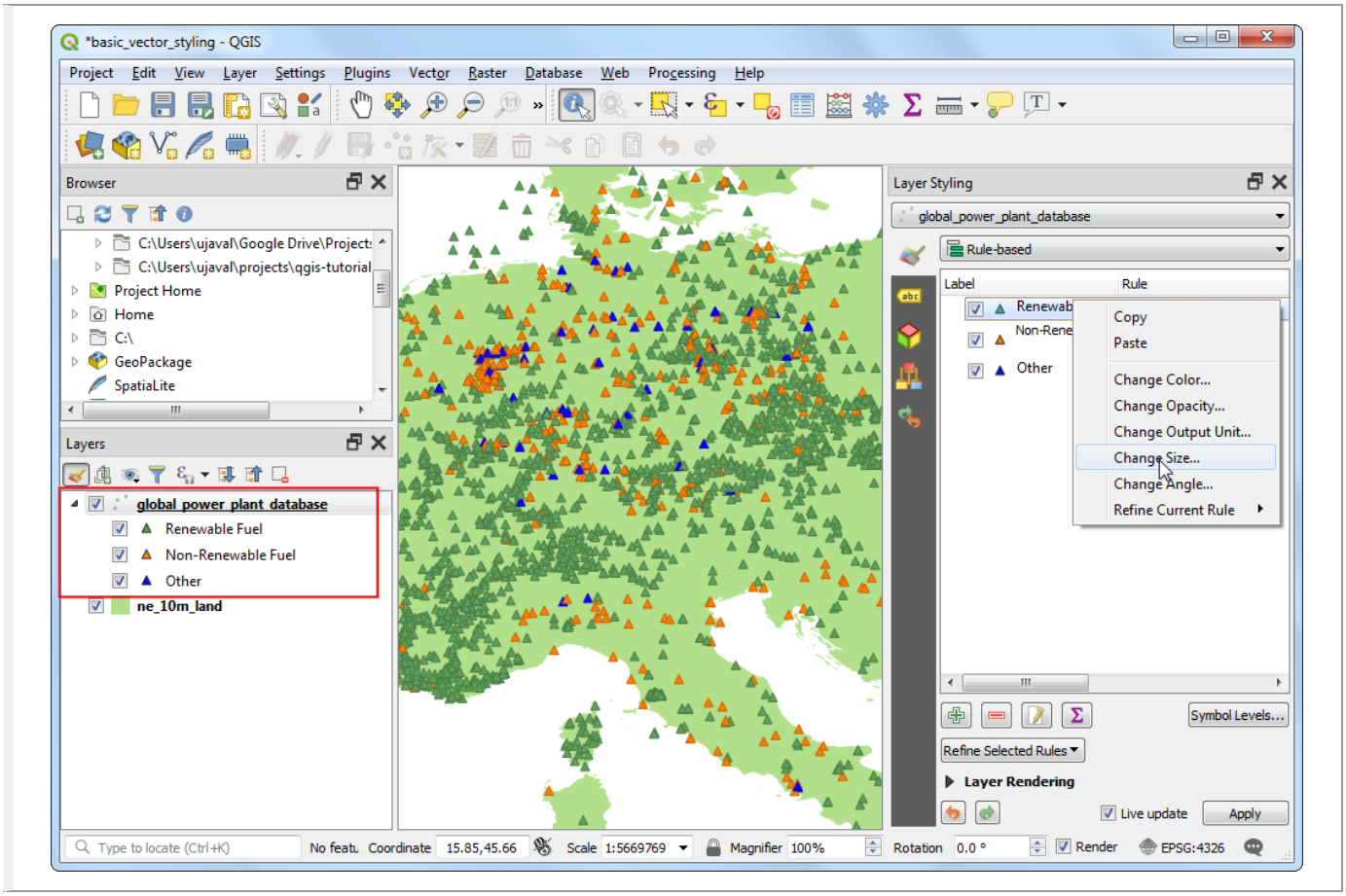
21. Repeat the Copy/Paste process to add a third rule. Select it and click Edit current rule.



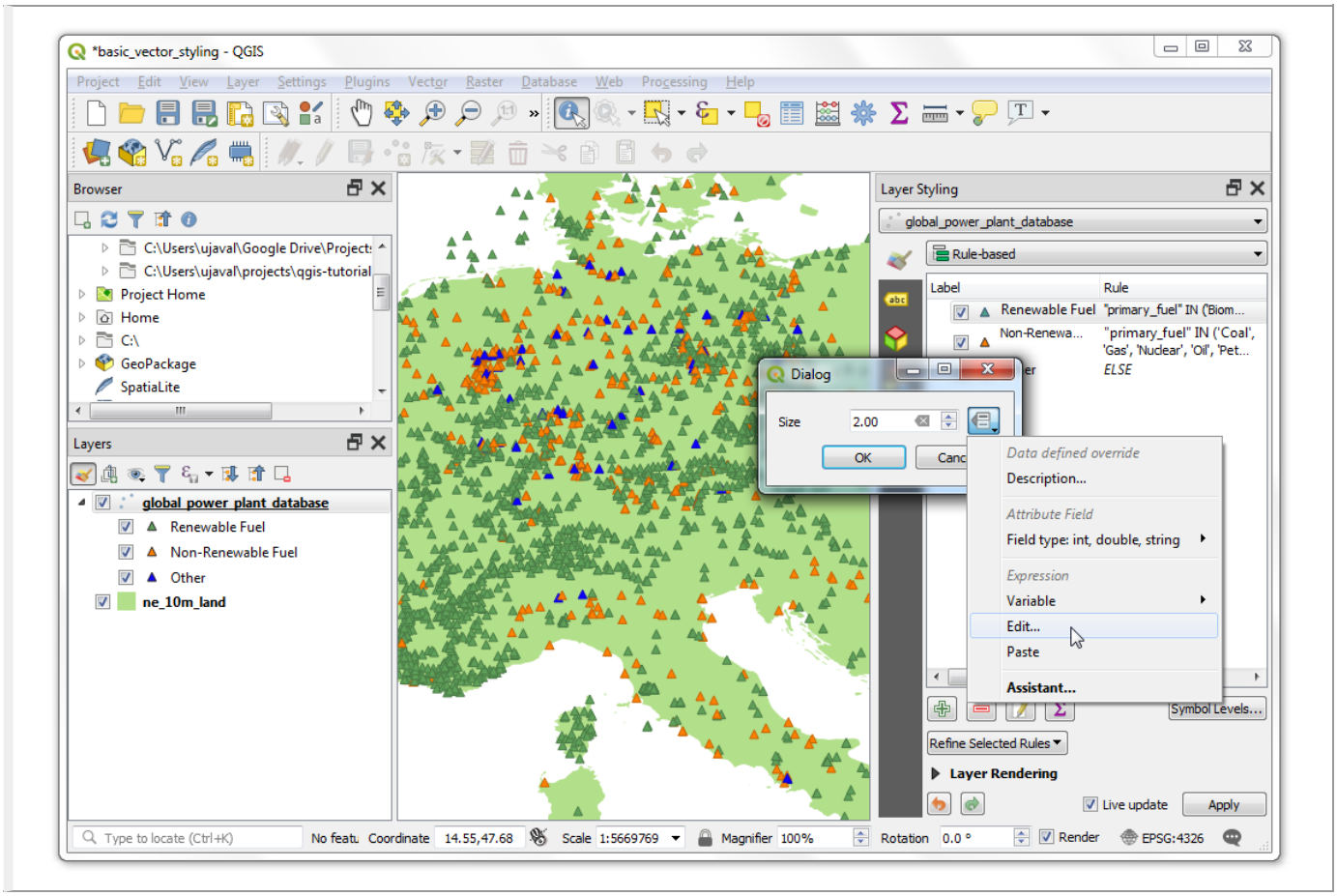
22. Enter `Other` as the Label. Choose Else - Catch all for other features instead of a Filter. This will ensure that any category missed in the previous 2 rules, will be styled by this rule. Scroll down and click Simple marker. Choose an appropriate Fill color. Once done, click the Back button.



23. The re-categorization is complete now. You will see a much cleaner view that shows the distribution of renewable vs. non-renewable fuel sources used by power plants and their distribution across countries. This however doesn't give a complete picture. We can add another variable to the styling. Rather than displaying all markers with uniform size, we can show the sizes proportional to the power generation capacity of each plant. This cartography technique is called *Multivariate mapping*. Right-click the `Renewable fuel` rule and select Change Size.

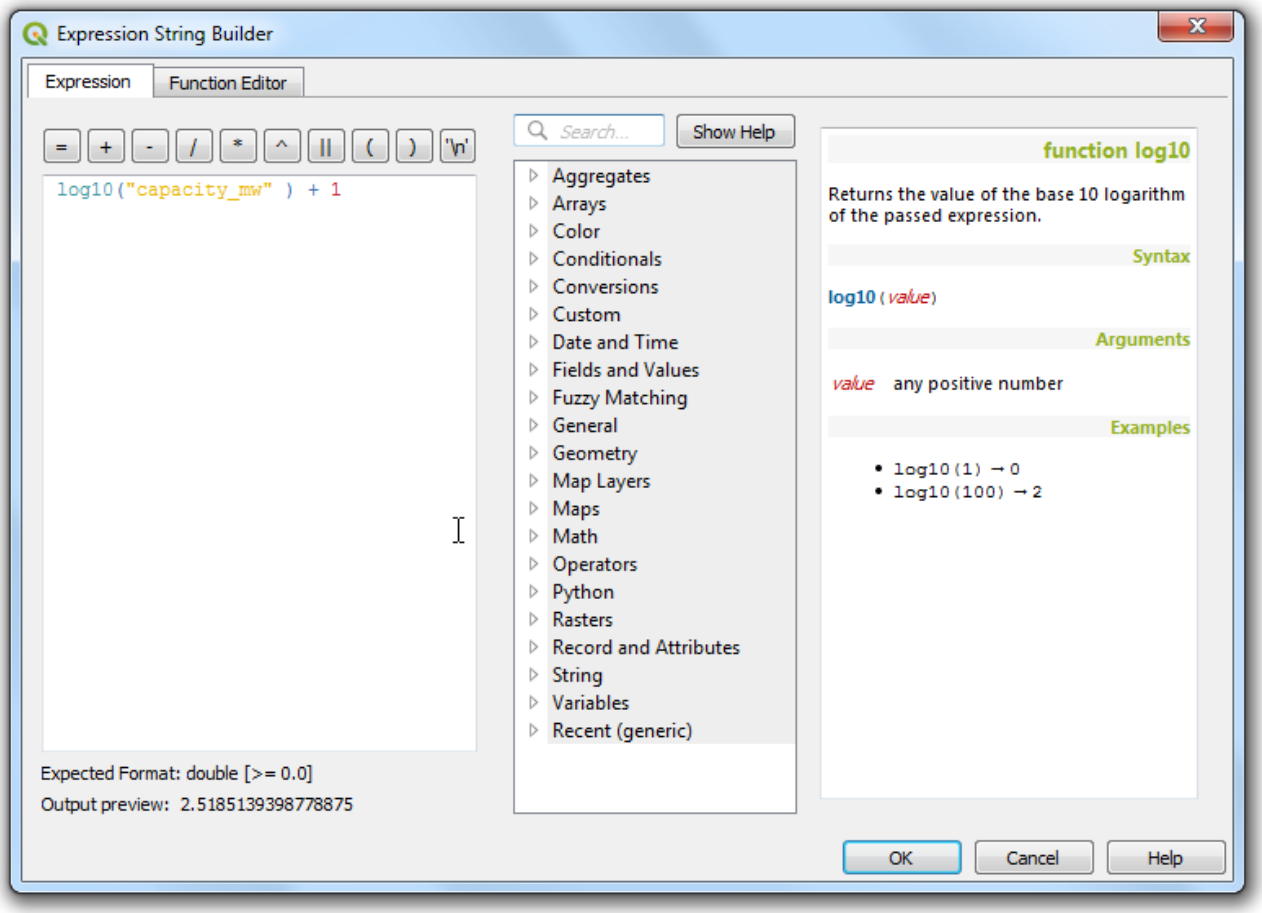


24. Click the Data defined override button next to Size. Select Edit.

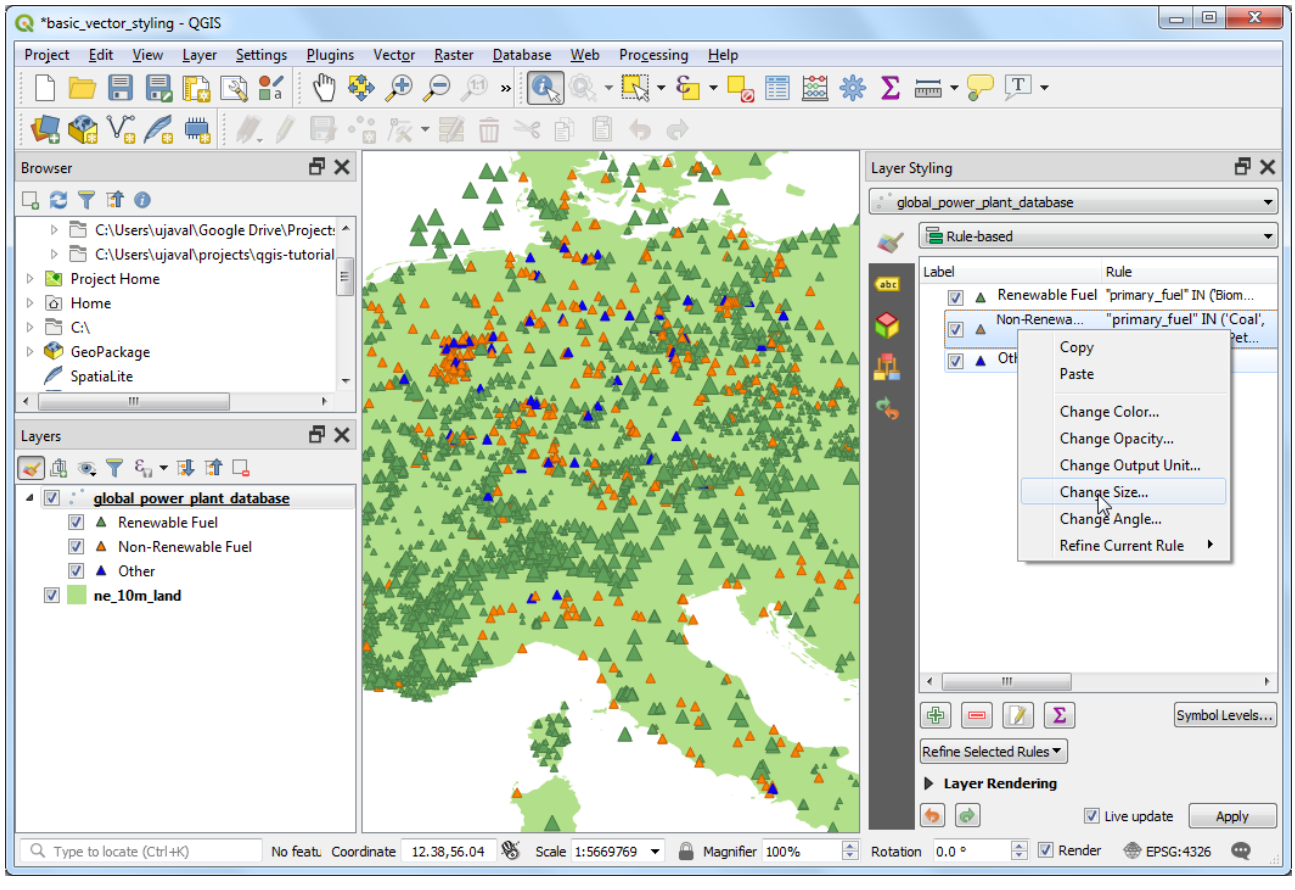


25. As the power generation capacity varies a lot among our dataset, an effective way to get a a small range for size is using the `log10` function. You can experiment with different expressions to arrive at what works best for your preferred visualization. Enter the following expression and click OK.

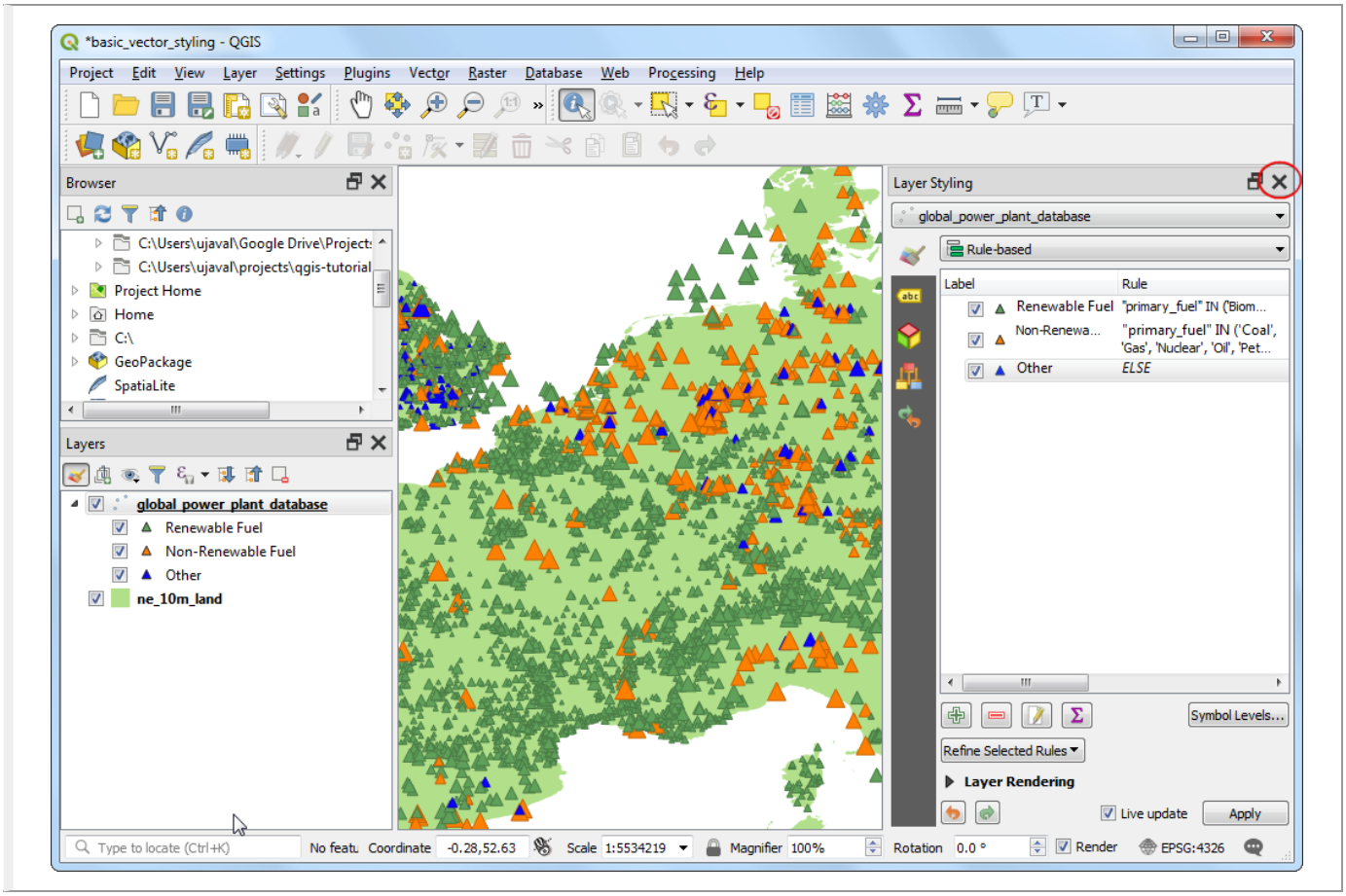
```
log10("capacity_mw") + 1
```



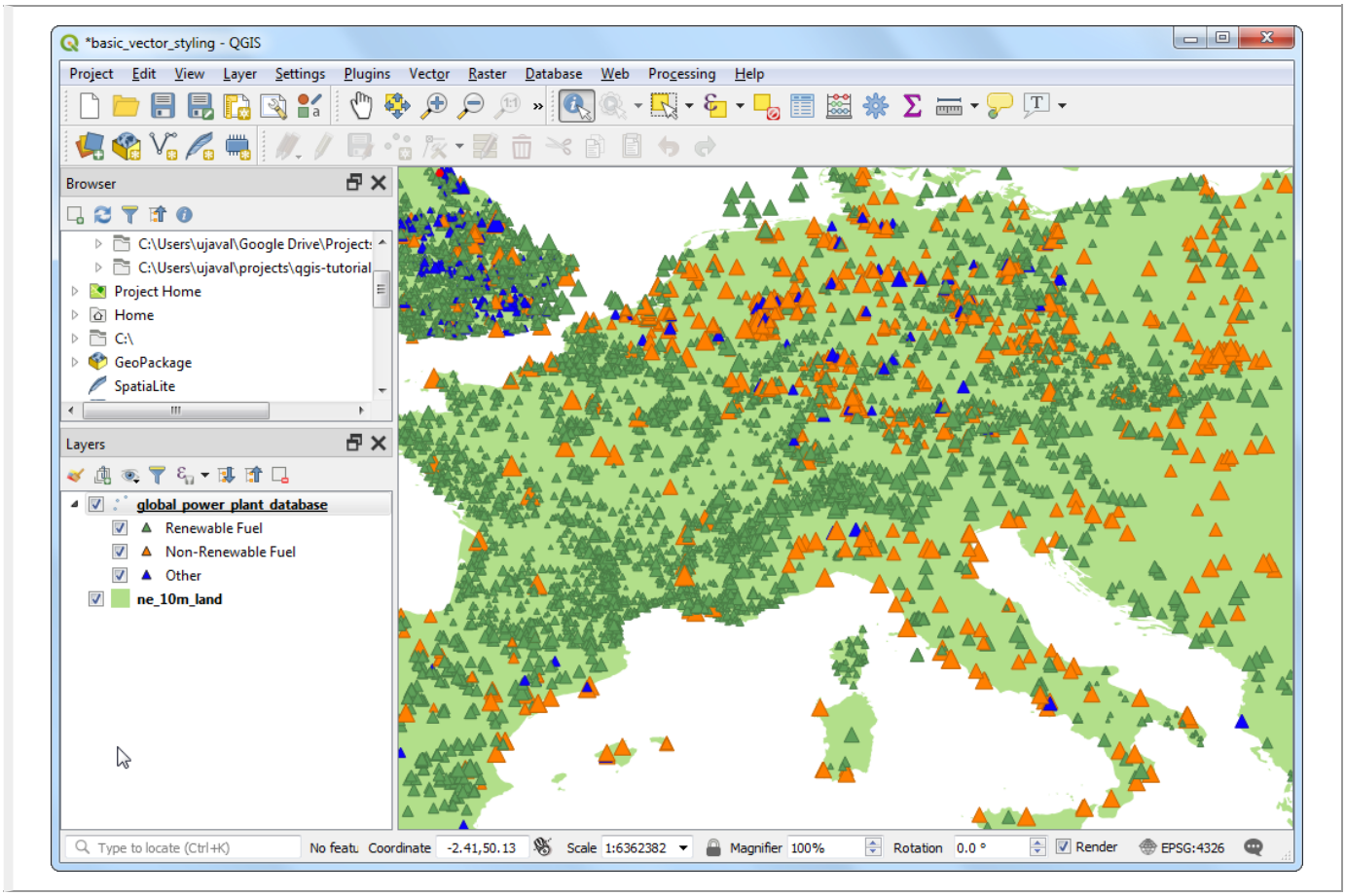
26. Repeat the same process for other rules.



27. Once satisfied, you can close the Layer Styling panel.



28. Looking at our final visualization, you can immediately see the patterns in the dataset. For example, over Europe there are more power plants that use renewable energy source, but they are lower capacity than the plants that use non-renewable energy source.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Calculating Line Lengths and Statistics (QGIS3)

QGIS has built-in functions and algorithms to calculate various properties based on the geometry of the feature - such as length, area, perimeter etc. This tutorial will show how to use the **Add geometry attributess** tool to add a column with a value representing length of each feature.

Overview of the task

Given a polyline layer of railroads in North America, we will determine the total length of railroads in the United States.

Other skills you will learn

- Using expressions to filter features.
- Using the Statistics panel to compute and view statistics on columns.

Get the data

Natural Earth (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/railroads/) has a public domain railroads dataset.

Download the North America supplement (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_railroads_north_america.zip) zip file from the portal.

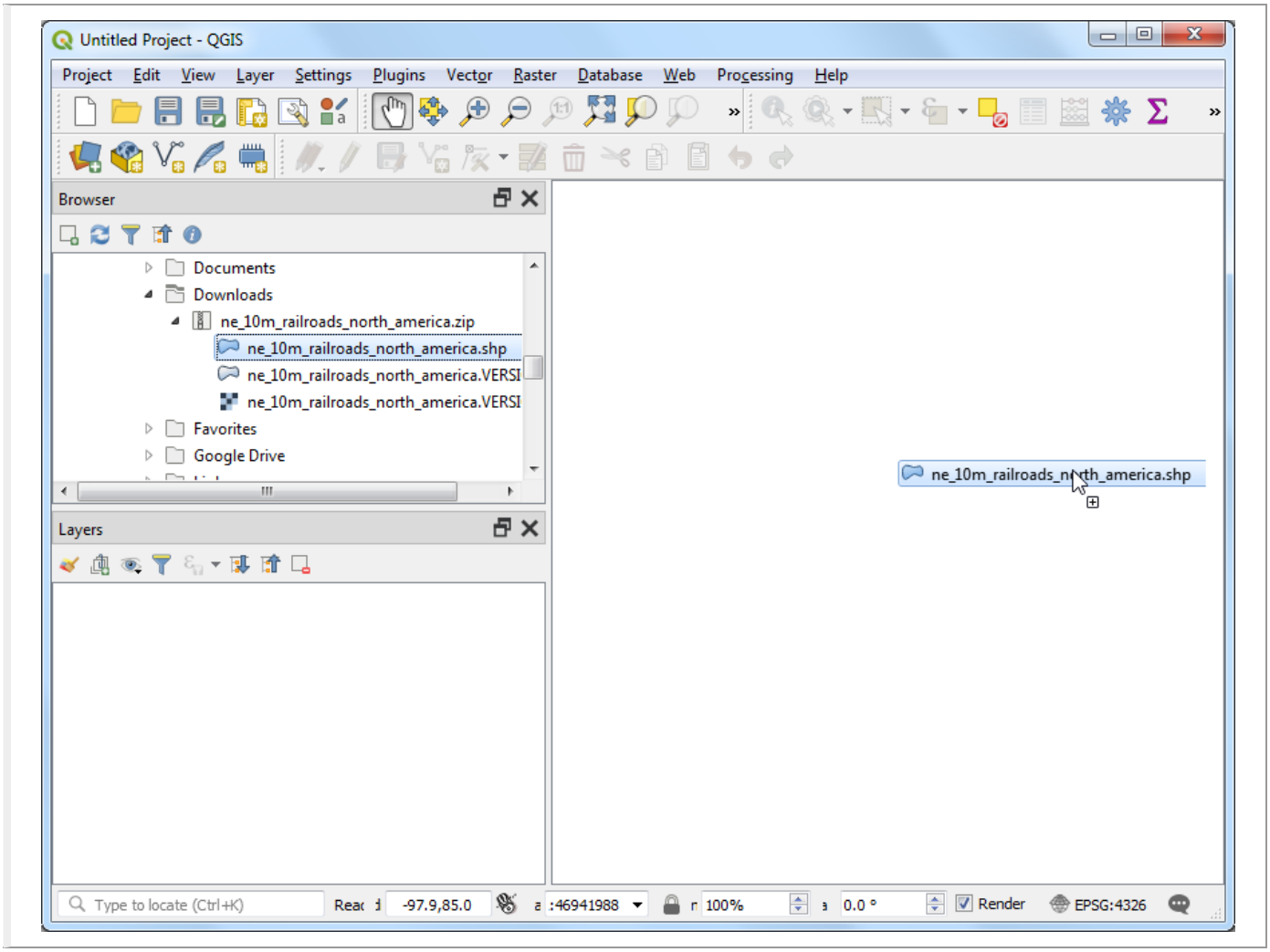
For convenience, you may directly download a copy of the dataset from the link below:

ne_10m_railroads_north_america..zip (http://www.qgistutorials.com/downloads/ne_10m_railroads_north_america.zip)

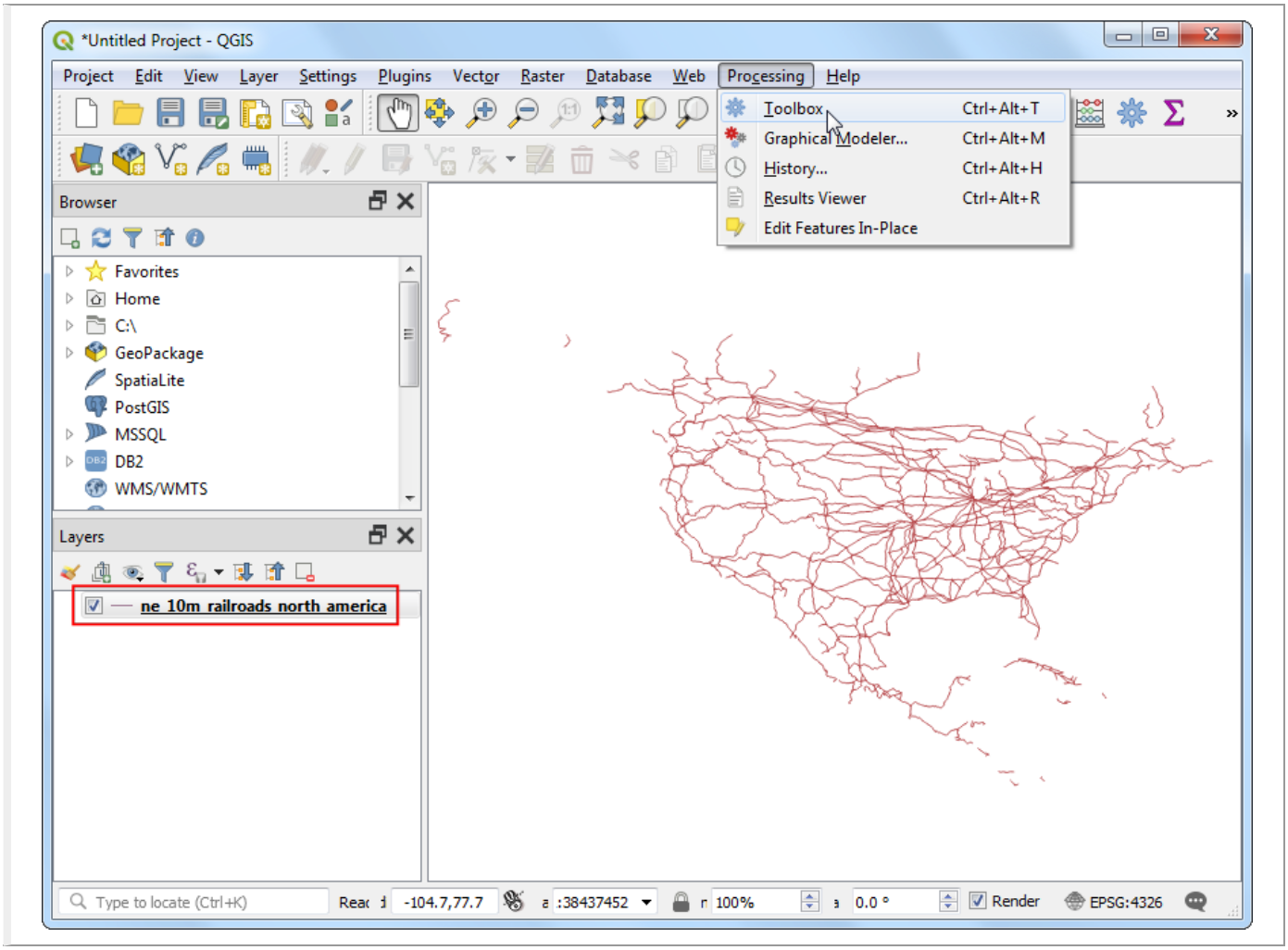
Data Source [NATURALEARTH] (../credits.html#naturalearth)

Procedure

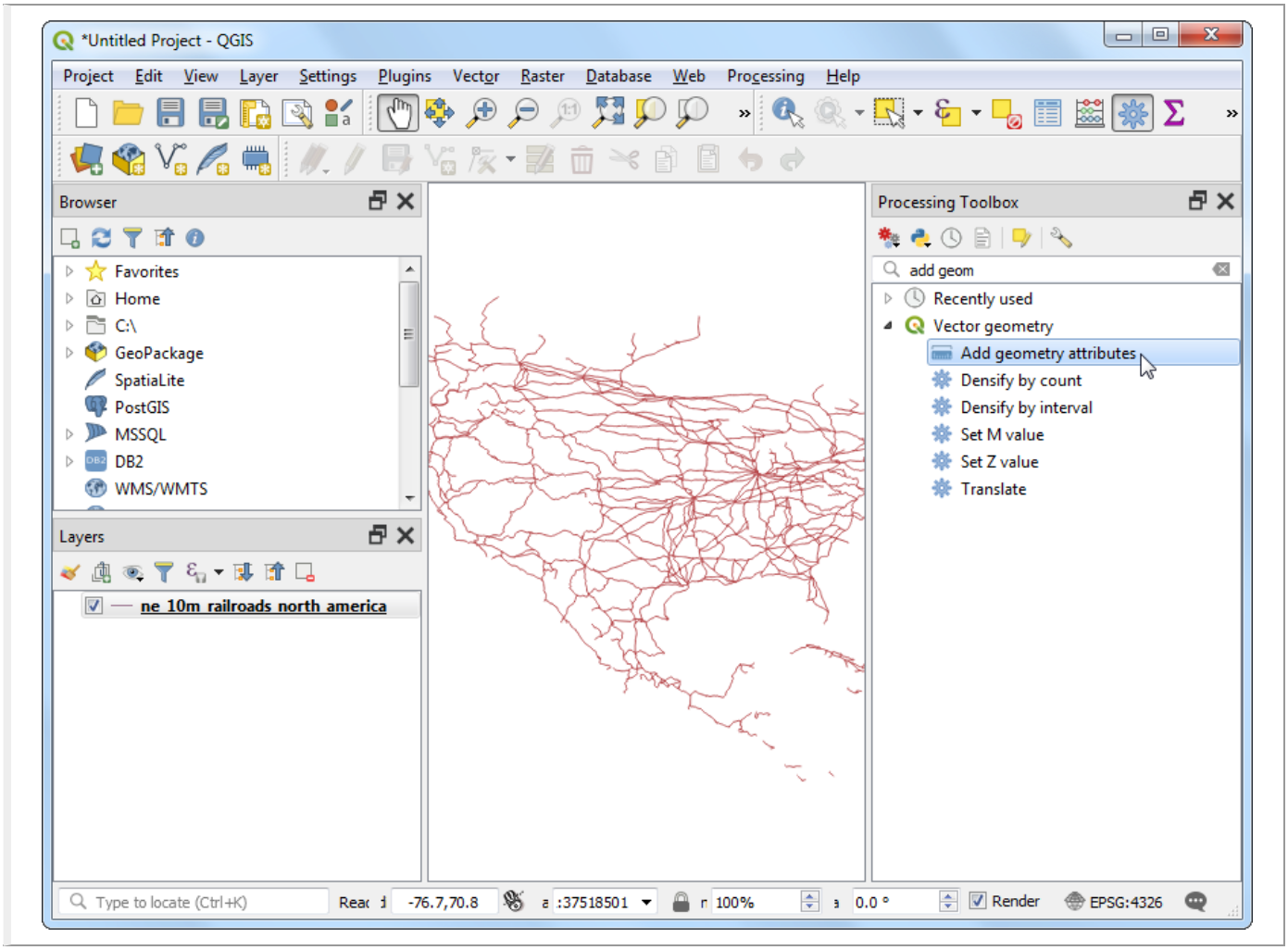
1. Locate the downloaded `ne_10m_railroads_north_america.zip` file in the Browser panel and expand it. Drag the `ne_10m_railroads_north_america.shp` file to the canvas.



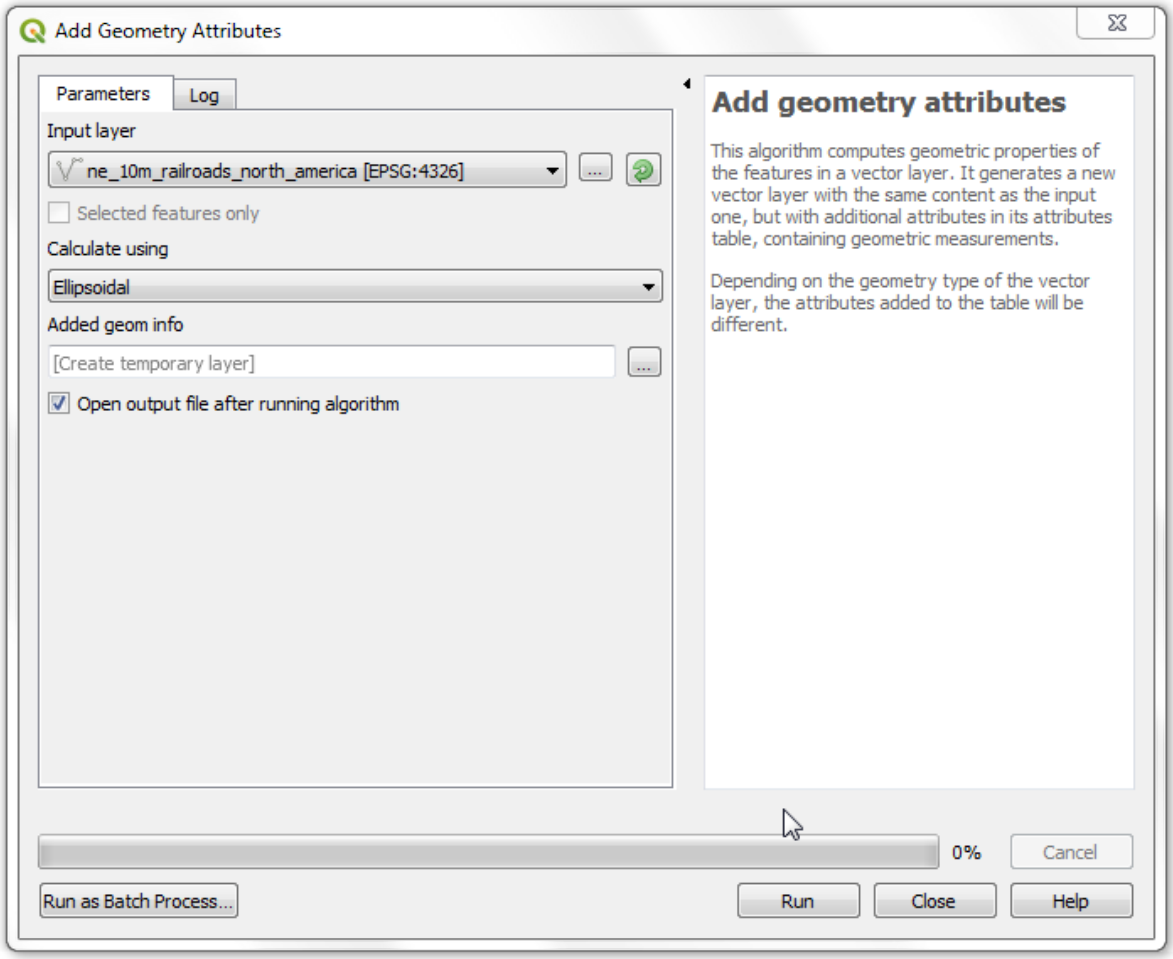
2. You will see a new layer `ne_10m_railroads_north_america` loaded in the Layers panel. You will see that the layer has lines representing railroads for all of North America. Now, let's calculate the lengths of each line feature. Go to Processing ‣ Toolbox.



3. Search for and locate the Vector geometry ‣ Add geometry attributes algorithm. Double-click to launch it.



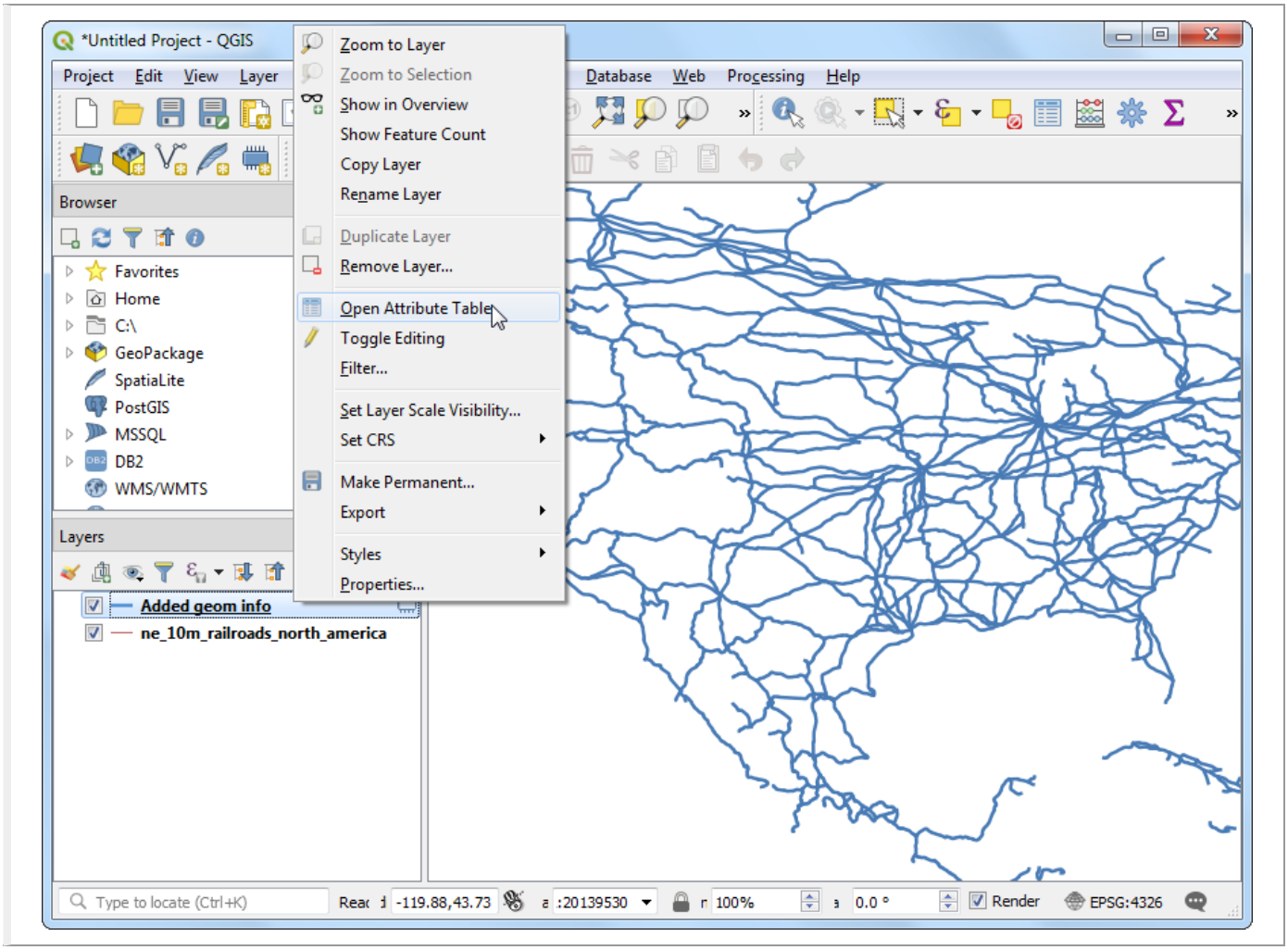
4. In the Add Geometry Attributes dialog, select `ne_10m_railroads_north_america` as the Input layer. The input layer's Coordinate Reference System (CRS) is *EPSG:4326 WGS84*. This is a *Geographic* CRS with Latitude and Longitude as coordinates, WGS84 as ellipsoid and degrees as units. Because latitude and longitude don't have a standard length, you can't measure distances or areas accurately using planar geometry functions. Fortunately, QGIS provides a better way to compute distances using ellipsoidal geometry, which is the most accurate method for layers spanning large areas such as this. Pick `Ellipsoidal` as the Calculate using option. Click Run. Once the process finishes, click Close.



Note

If your input layer is in a *Projected CRS*, you may choose `Layer CRS` option for calculation. Local or Regional projected coordinate systems are designed to minimize distortions over their region of interest, so are more accurate for such computation.

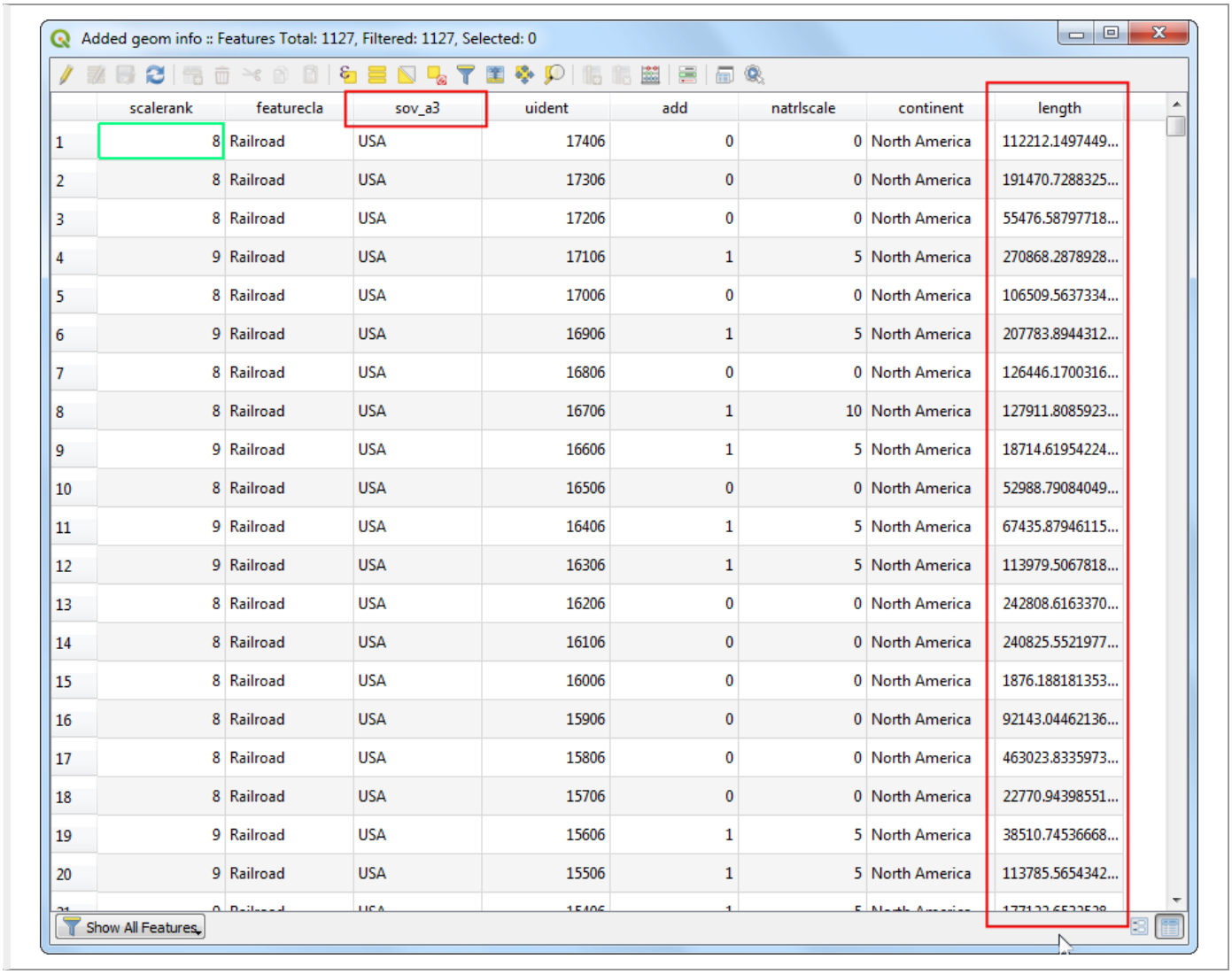
5. You will see a new layer `Added geom info` loaded in the Layers panel. This is a copy of the input layer with a new column added for distance. Right-click the `Added geom info` layer and select Open Attribute Table.



Note

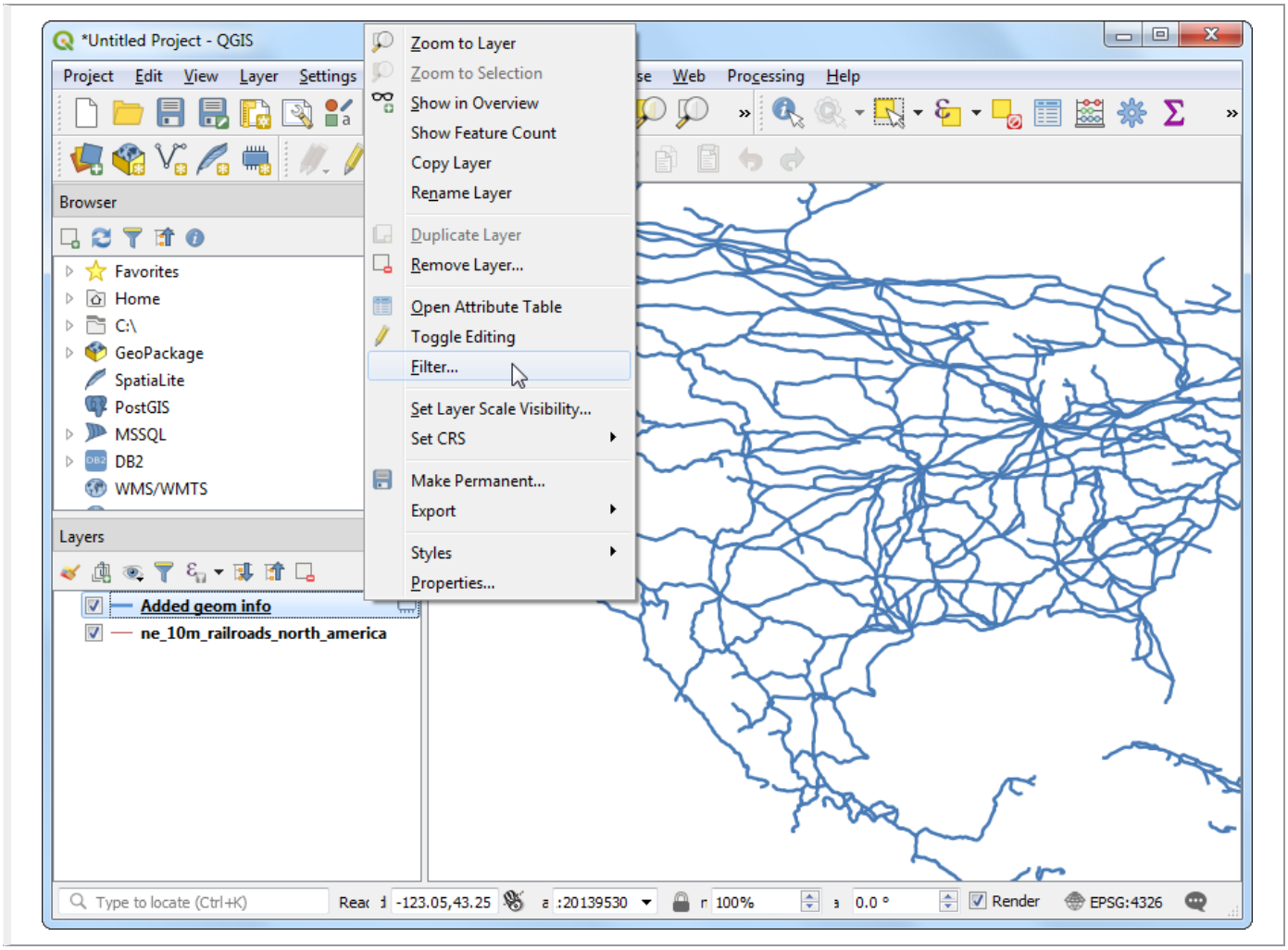
The *Add Geometry Attribute* tool adds different set of attributes depending on whether the input layer is points, lines or polygons. See QGIS documentation (https://docs.qgis.org/testing/en/docs/user_manual/processing_algs/qgis/vectorgeometry.html#add-geometry-attributes) for more details.

6. In the Attribute Table, you will see a new column called **distance**. This contains the length of each line feature in *meters*. Also note that the **sov_a3** attribute which contains the contry code for each feature. Close the Attribute Table window.



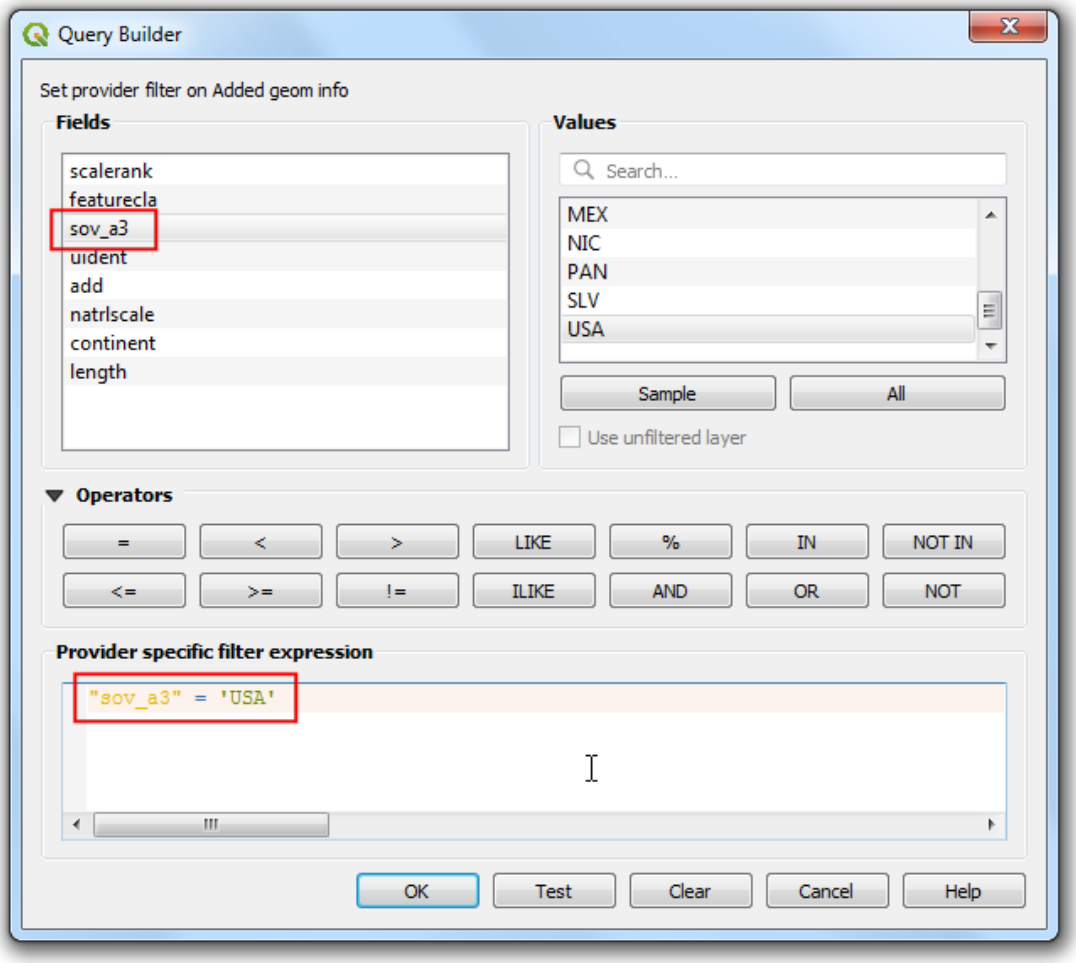
	scalerank	featurecla	sov_a3	uident	add	natrlscale	continent	length
1	8	Railroad	USA	17406	0	0	North America	112212.1497449...
2	8	Railroad	USA	17306	0	0	North America	191470.7288325...
3	8	Railroad	USA	17206	0	0	North America	55476.58797718...
4	9	Railroad	USA	17106	1	5	North America	270868.2878928...
5	8	Railroad	USA	17006	0	0	North America	106509.5637334...
6	9	Railroad	USA	16906	1	5	North America	207783.8944312...
7	8	Railroad	USA	16806	0	0	North America	126446.1700316...
8	8	Railroad	USA	16706	1	10	North America	127911.8085923...
9	9	Railroad	USA	16606	1	5	North America	18714.61954224...
10	8	Railroad	USA	16506	0	0	North America	52988.79084049...
11	9	Railroad	USA	16406	1	5	North America	67435.87946115...
12	9	Railroad	USA	16306	1	5	North America	113979.5067818...
13	8	Railroad	USA	16206	0	0	North America	242808.6163370...
14	8	Railroad	USA	16106	0	0	North America	240825.5521977...
15	8	Railroad	USA	16006	0	0	North America	1876.188181353...
16	8	Railroad	USA	15906	0	0	North America	92143.04462136...
17	8	Railroad	USA	15806	0	0	North America	463023.8335973...
18	8	Railroad	USA	15706	0	0	North America	22770.94398551...
19	9	Railroad	USA	15606	1	5	North America	38510.74536668...
20	9	Railroad	USA	15506	1	5	North America	113785.5654342...

7. Now that we have lengths of individual railroad line segments, we can add them up to find the total length of railroads. But as the problem statement demands we need total railroad length in the United States, we must use only the segments contained within USA. We can use the country code value in the **sov_a3** column to filter the layer. Right-click the `Added geom info` layer and select Filter.

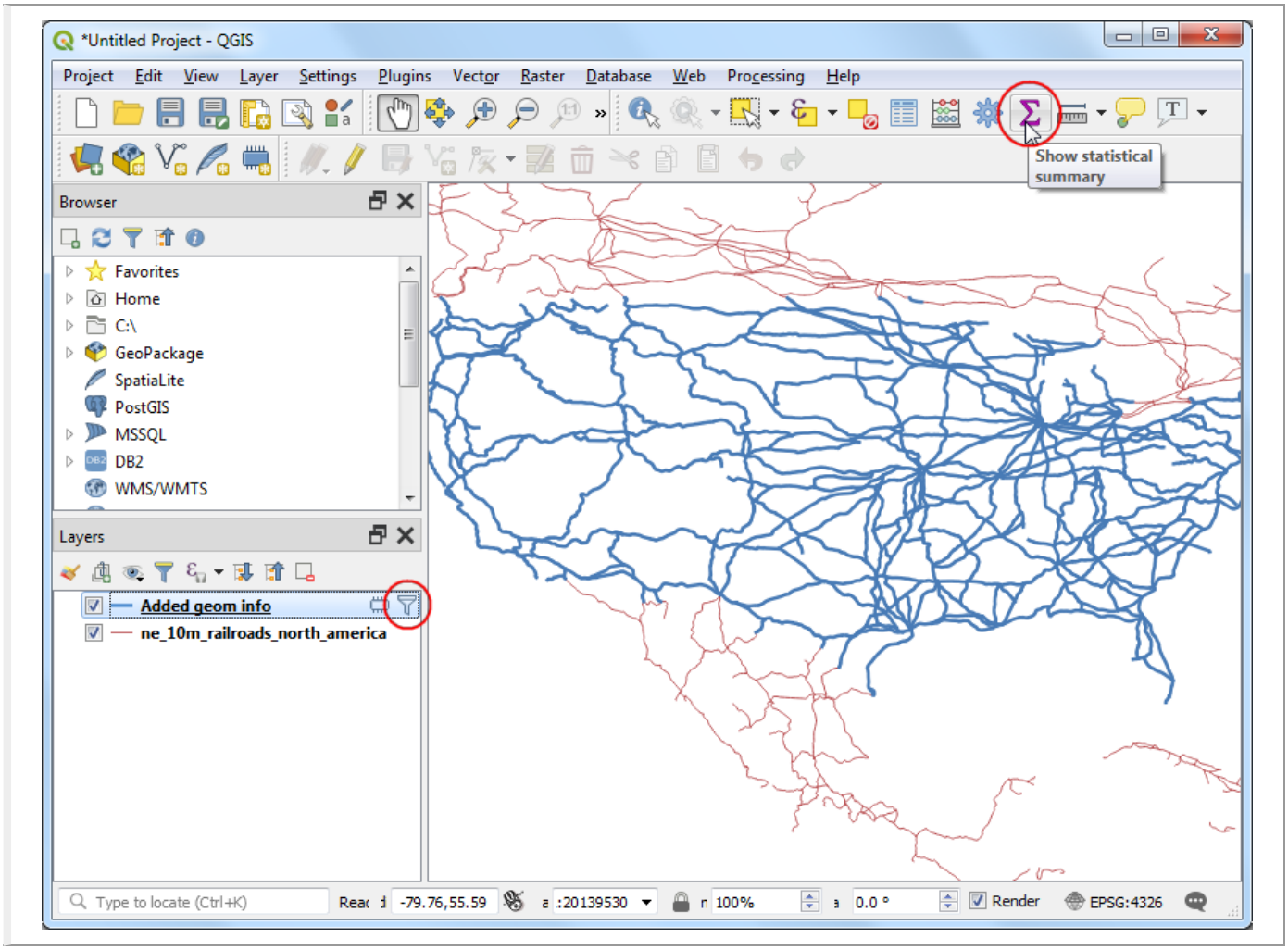


8. In the Query Builder dialog, enter the following expression and click OK.

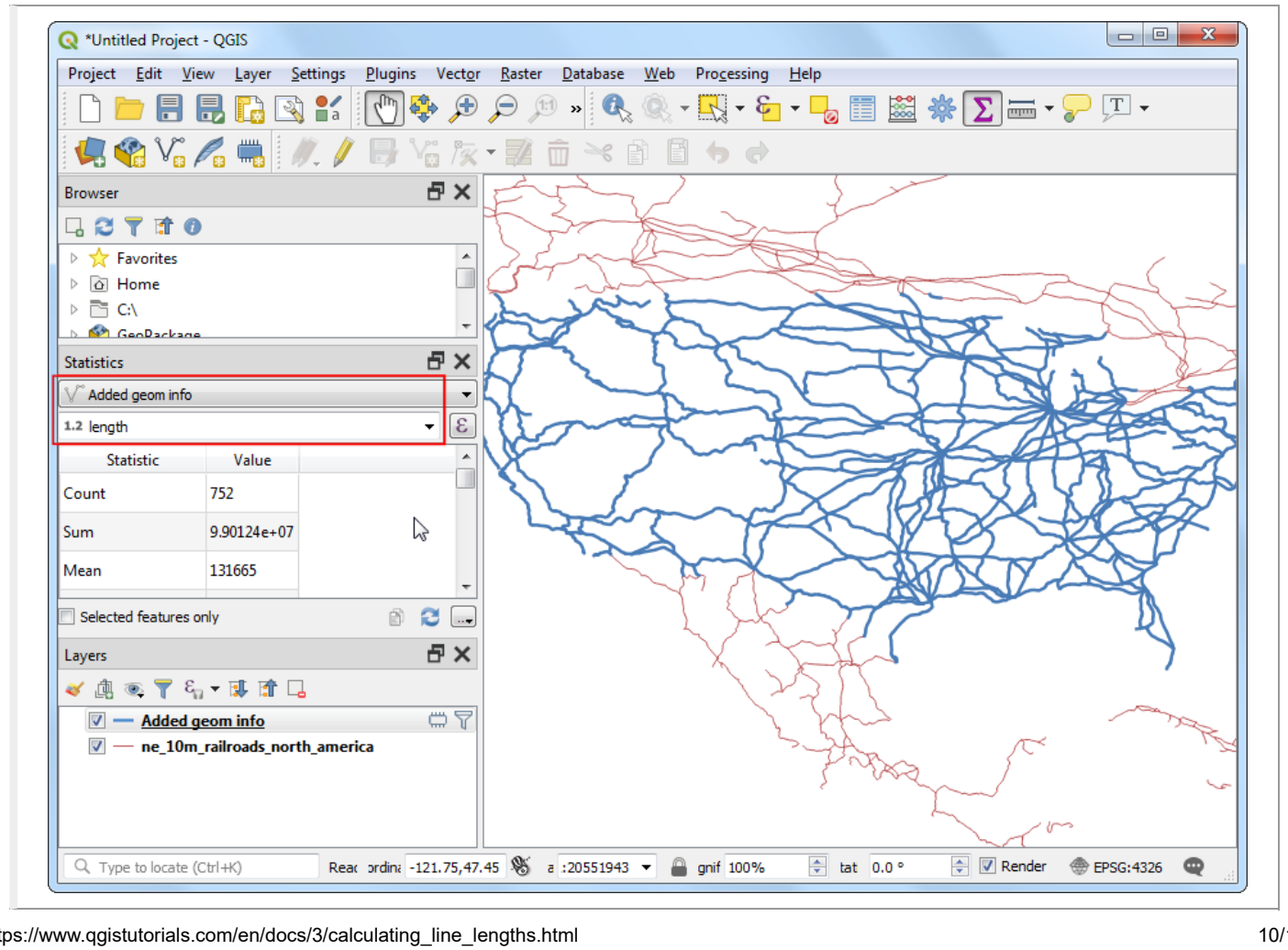
```
"sov_a3" = 'USA'
```



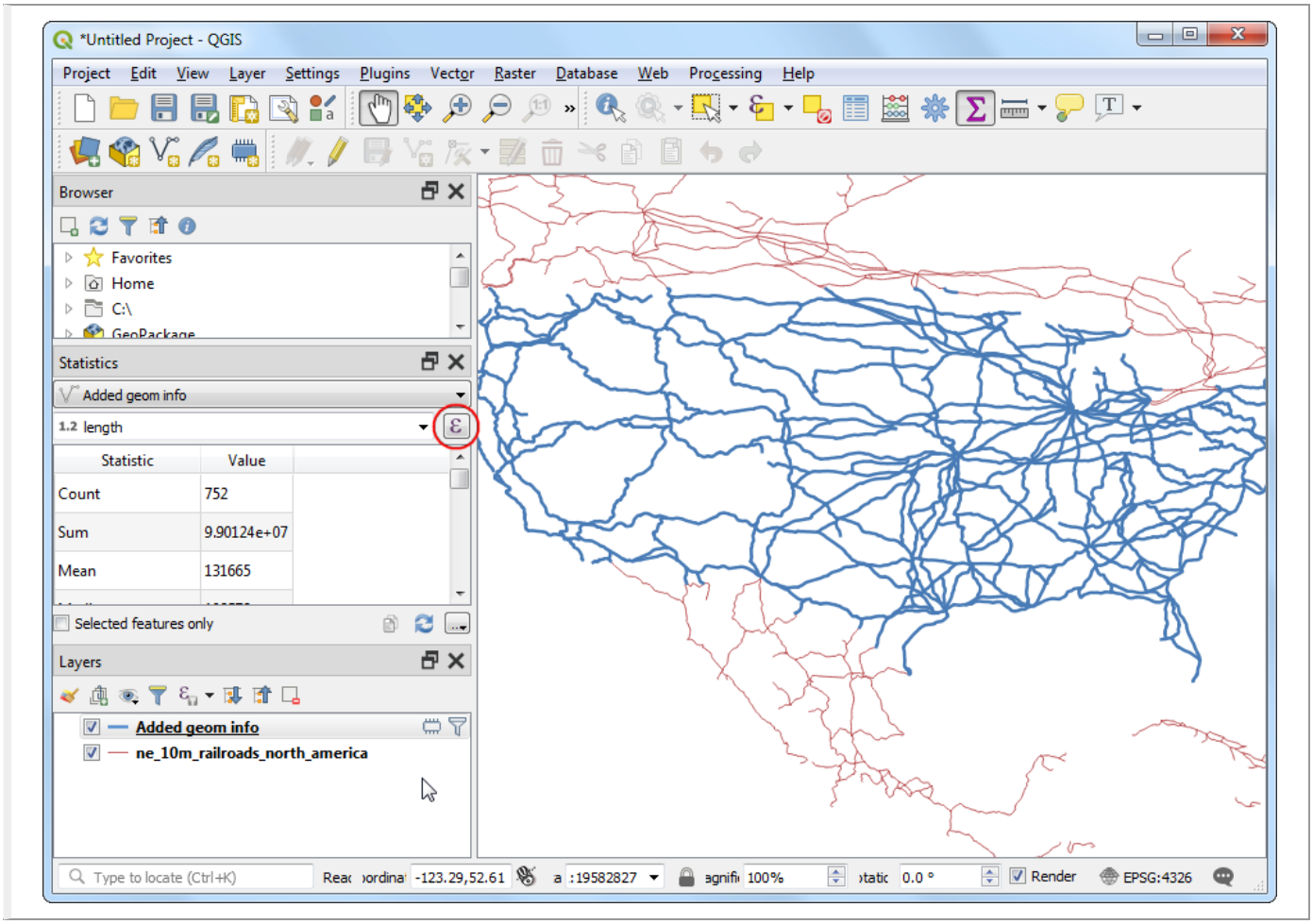
9. You will see a Filter icon appear next to the `Added geom info` layer in the Layers panel indicating that a filter is applied to the layer. You can also visually confirm that the layer now contains line segments only for United States. Now we are ready to calculate the sum. Click the Show statistical summary button on the Attributes Toolbar.



10. A new Statistics panel will open. Select `Added geom info` layer and `length` column.

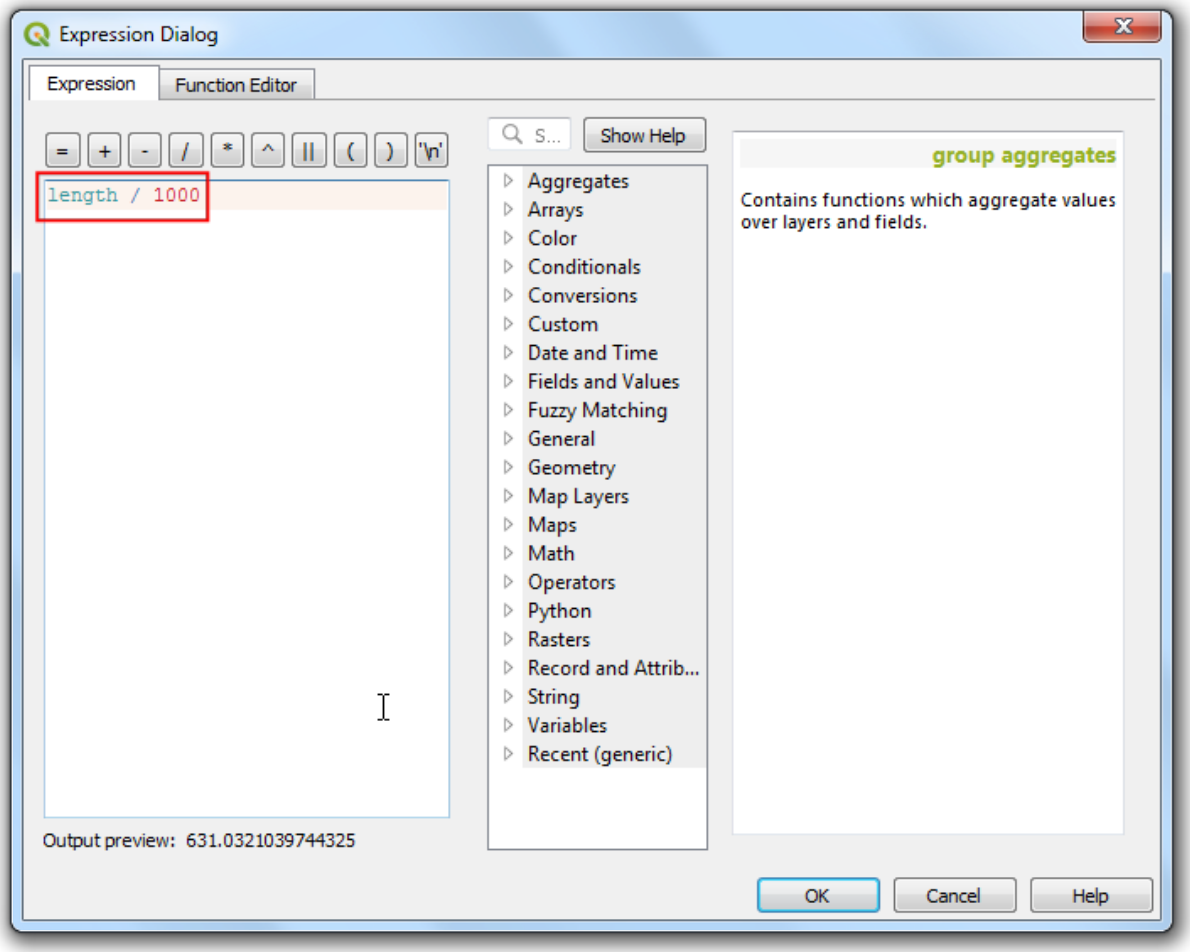


11. You will see various statistics displayed in the panel. The unit of the statistics is the same as the units of `length` column - **meters**. Let's change the computation to use **kilometers** instead. Click the Expression icon next to the fields drop-down menu in the Statistics panel.

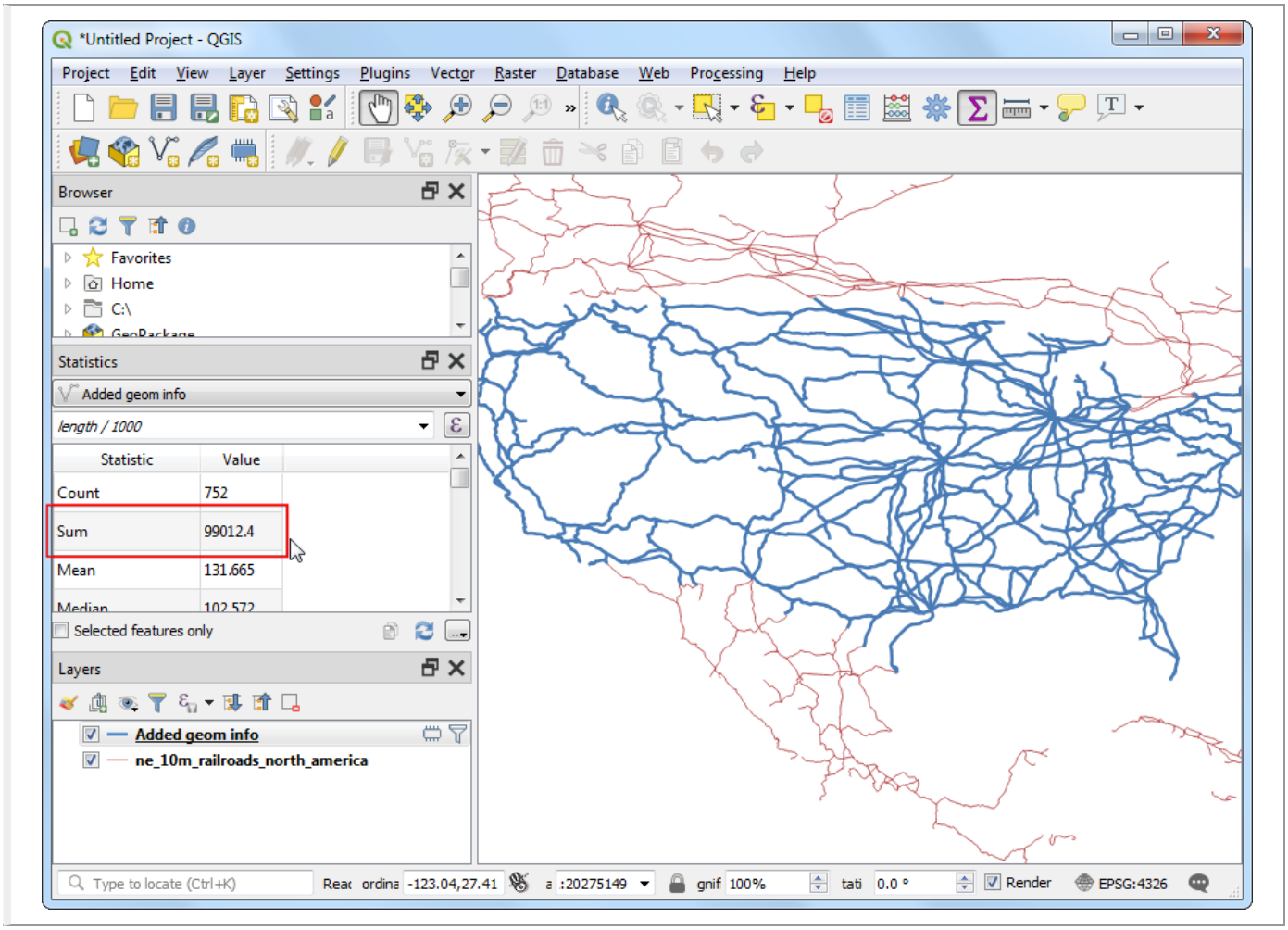


12. Enter the following expression in the Expression Dialog that converts the length to kilometers.

```
length / 1000
```



13. The Sum value displayed is the total length of railroads in USA.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Basic Raster Styling and Analysis (QGIS3)

A lot of scientific observations and research produces raster datasets. Rasters are grids of pixels that have a specific value assigned to them. By doing mathematical operations on these values, one can do some interesting analysis. QGIS has some basic analysis capabilities built-in via **Raster Calculator**. In this tutorial, we will explore the options available for styling rasters and functionality provided by the raster calculator.

Overview of the task

We will use population grid data to create a thematic map of the global population change between year 2000 and 2010.

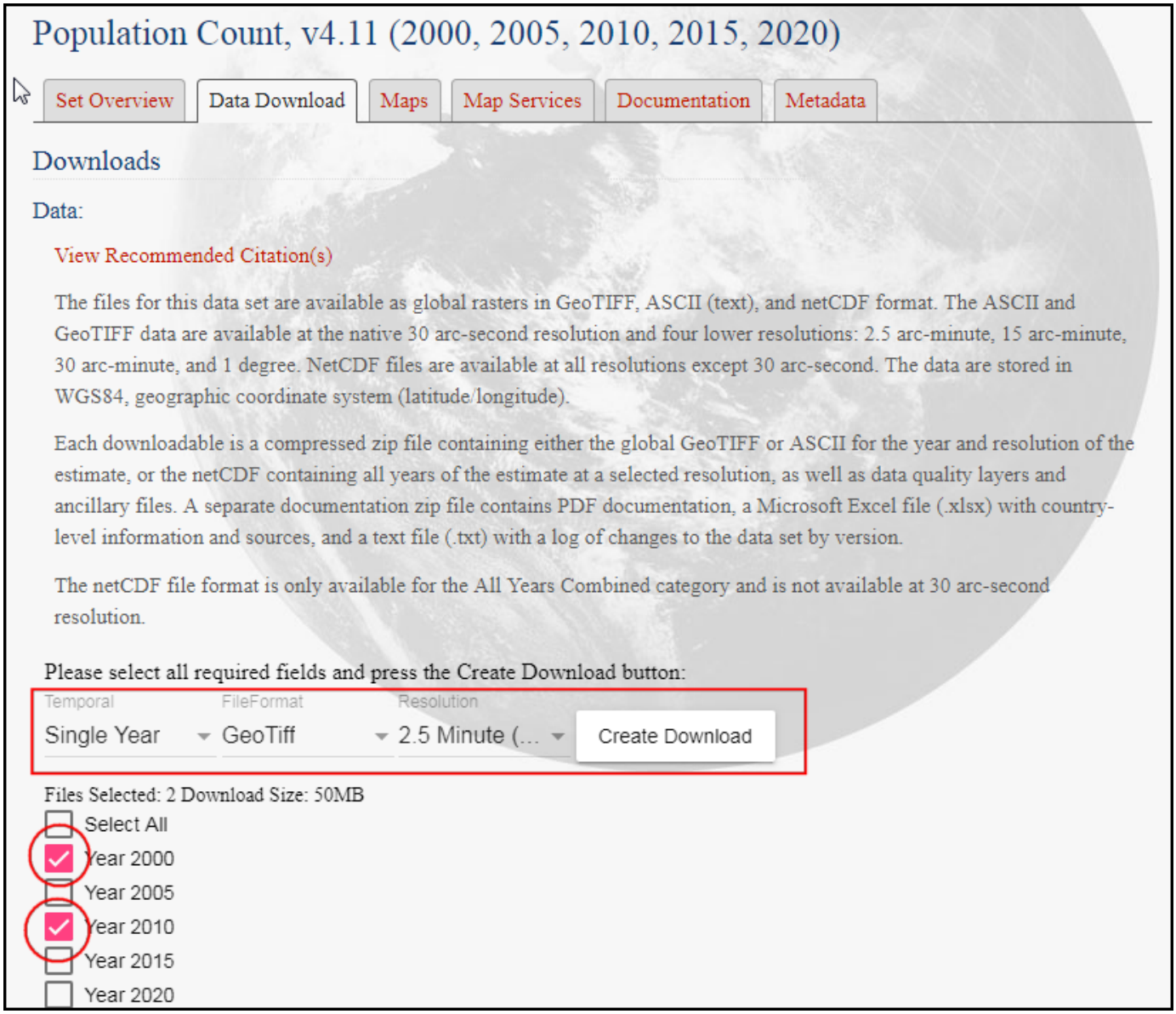
Other skills you will learn

- How to copy/paste styles between layers

Get the data

We will use the Gridded Population of the World (GPW) v4 (https://sedac.ciesin.columbia.edu/data/collection/gpw-v4) dataset from Columbia University. Specifically, we need the Population Count (https://sedac.ciesin.columbia.edu/data/set/gpw-v4-population-count-

rev11/data-download) for the entire globe at 2.5 Degree Minute resolution in GeoTiFF format and for the year 2000 and 2010. You will need a free Earth Data account (https://urs.earthdata.nasa.gov/home) to download the data.



For convenience, you may directly download a copy of the datasets from the links below:

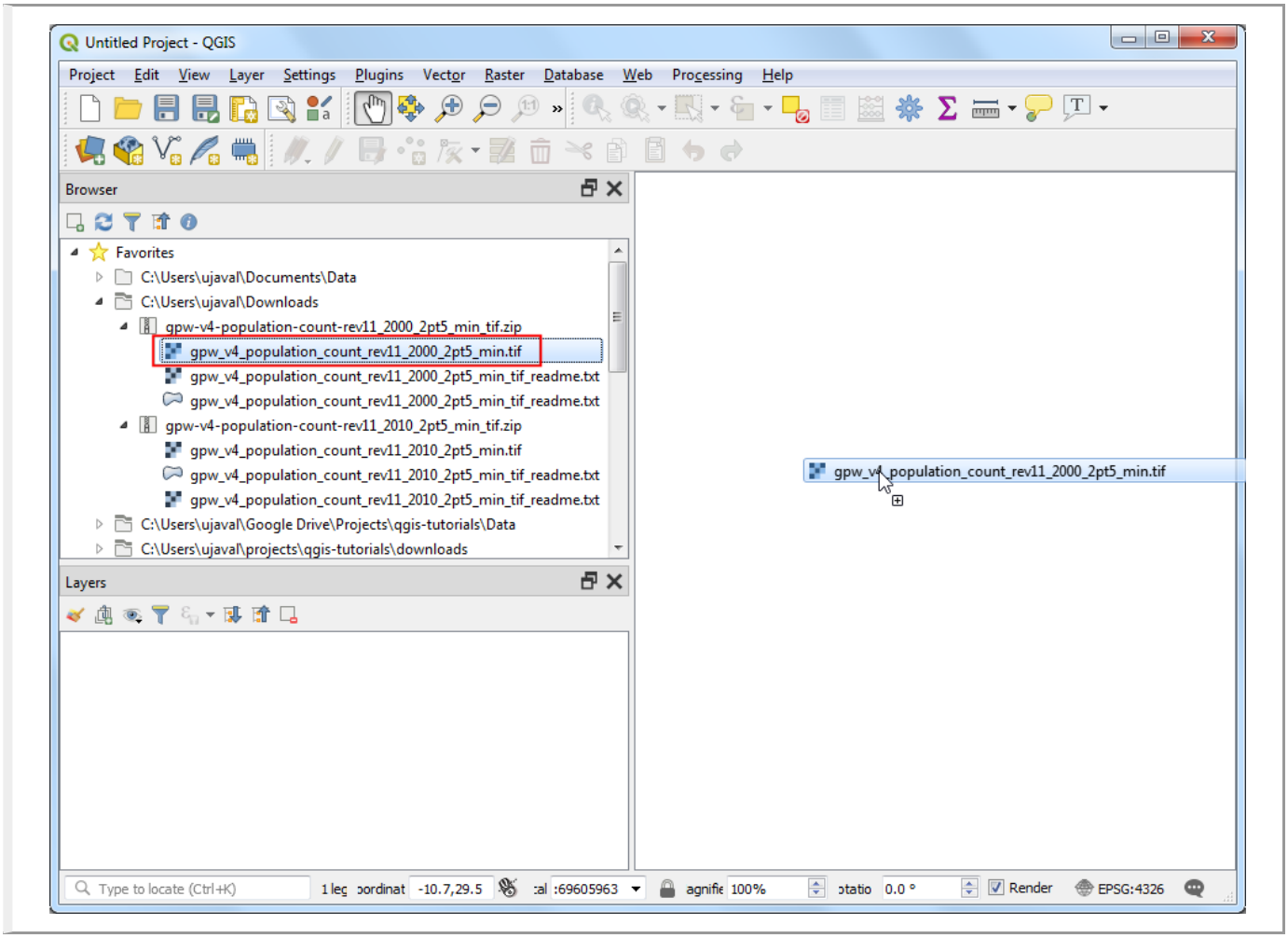
gpw-v4-population-count-rev11_2000_2pt5_min_tif.zip (http://www.qgistutorials.com/downloads/gpw-v4-population-count-rev11_2000_2pt5_min_tif.zip)

gpw-v4-population-count-rev11_2010_2pt5_min_tif.zip (http://www.qgistutorials.com/downloads/gpw-v4-population-count-rev11_2010_2pt5_min_tif.zip)

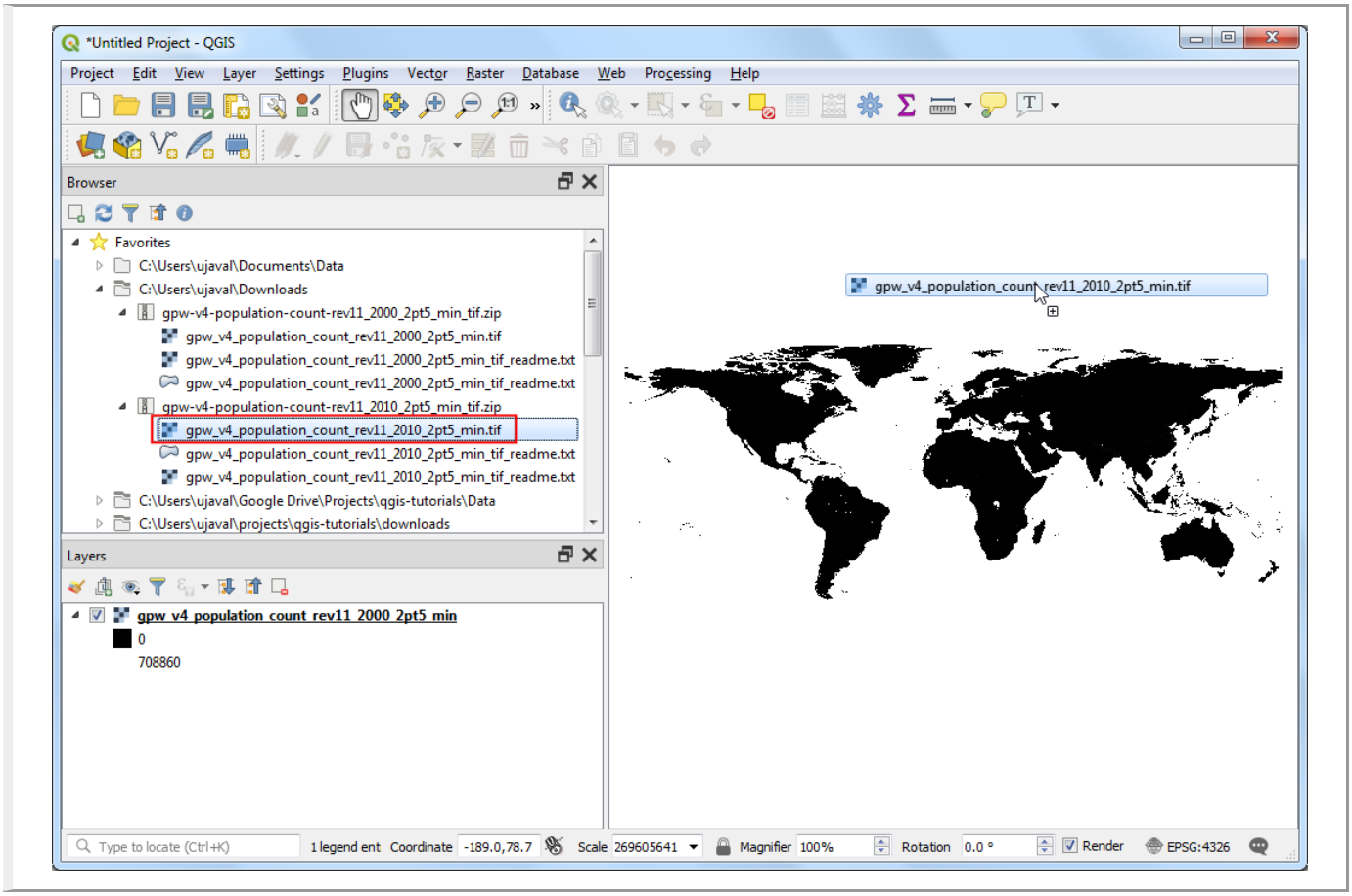
Data Source [GPW4] (../credits.html#gpw4)

Procedure

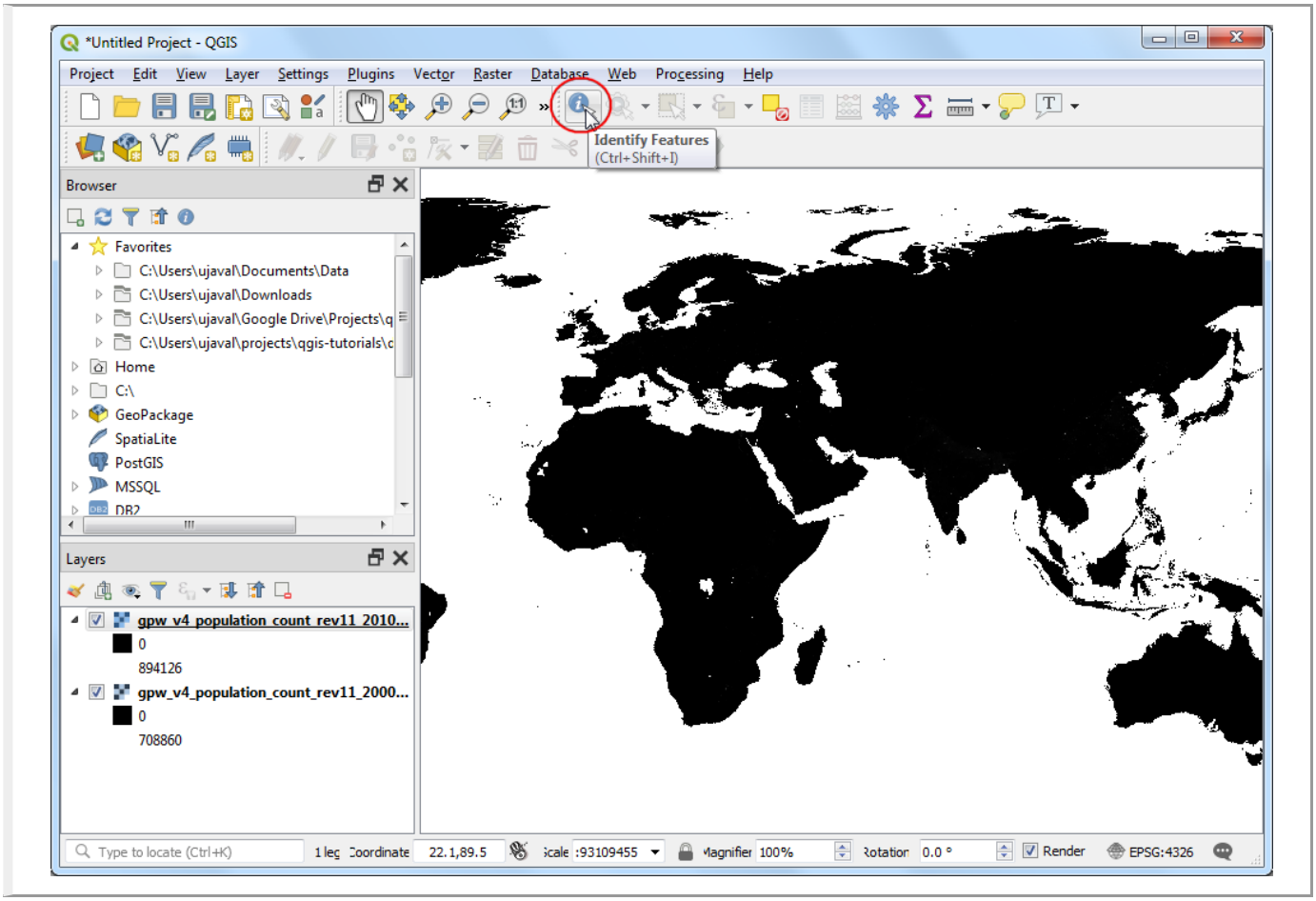
1. Open QGIS and locate the downloaded files in the Browser panel. Expand the `gpw-v4-population-count-rev11_2000_2pt5_min_tif.zip` file and drag the `gpw-v4-population-count-rev11_2000_2pt5_min.tif` file to the canvas.



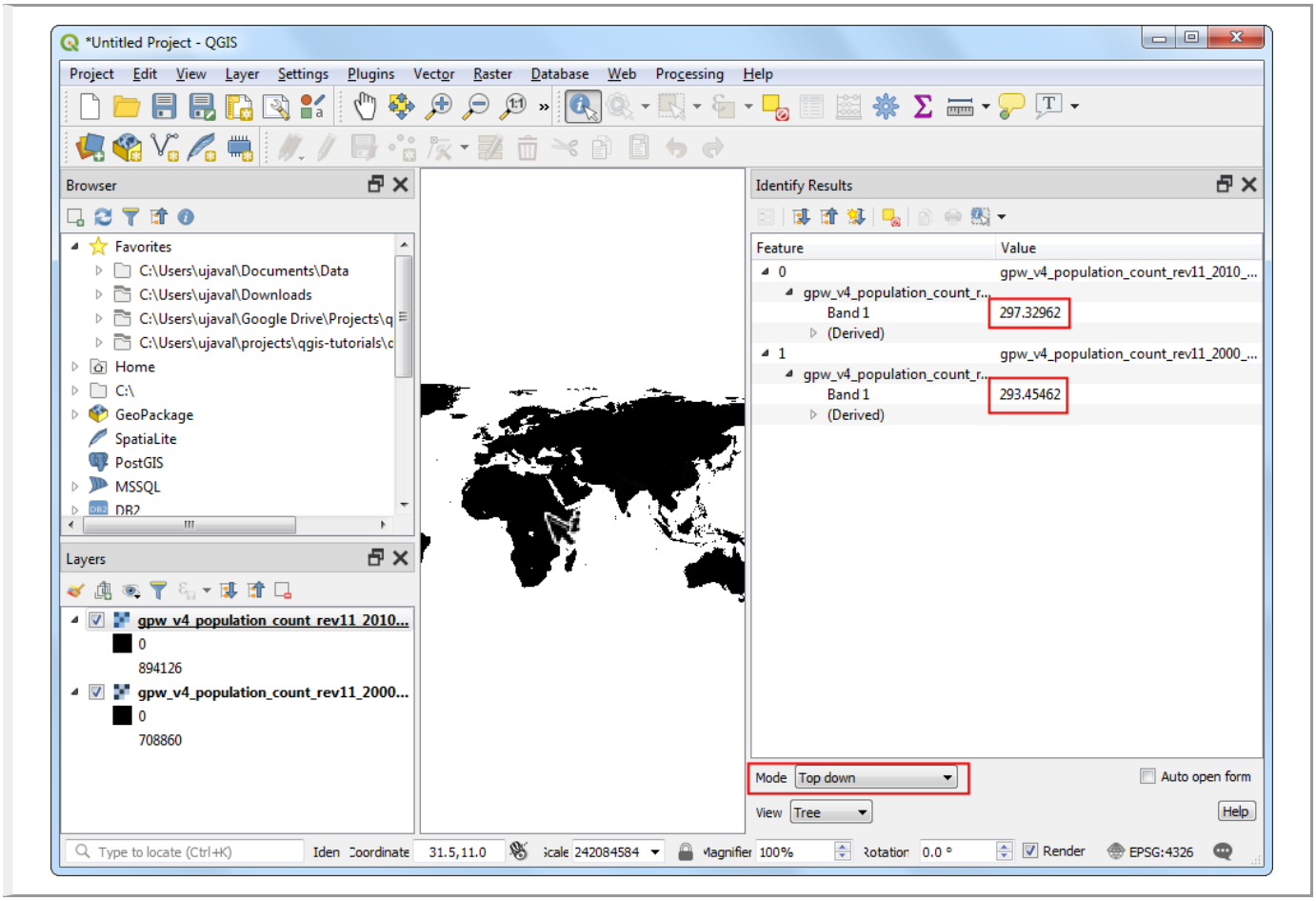
2. A new layer `gpw-v4-population-count-rev11_2000_2pt5_min` will be added to the Layers panel. Similarly, locate the `gpw-v4-population-count-rev11_2010_2pt5_min_tif.zip` file and drag the `gpw-v4-population-count-rev11_2010_2pt5_min.tif` file to the canvas.



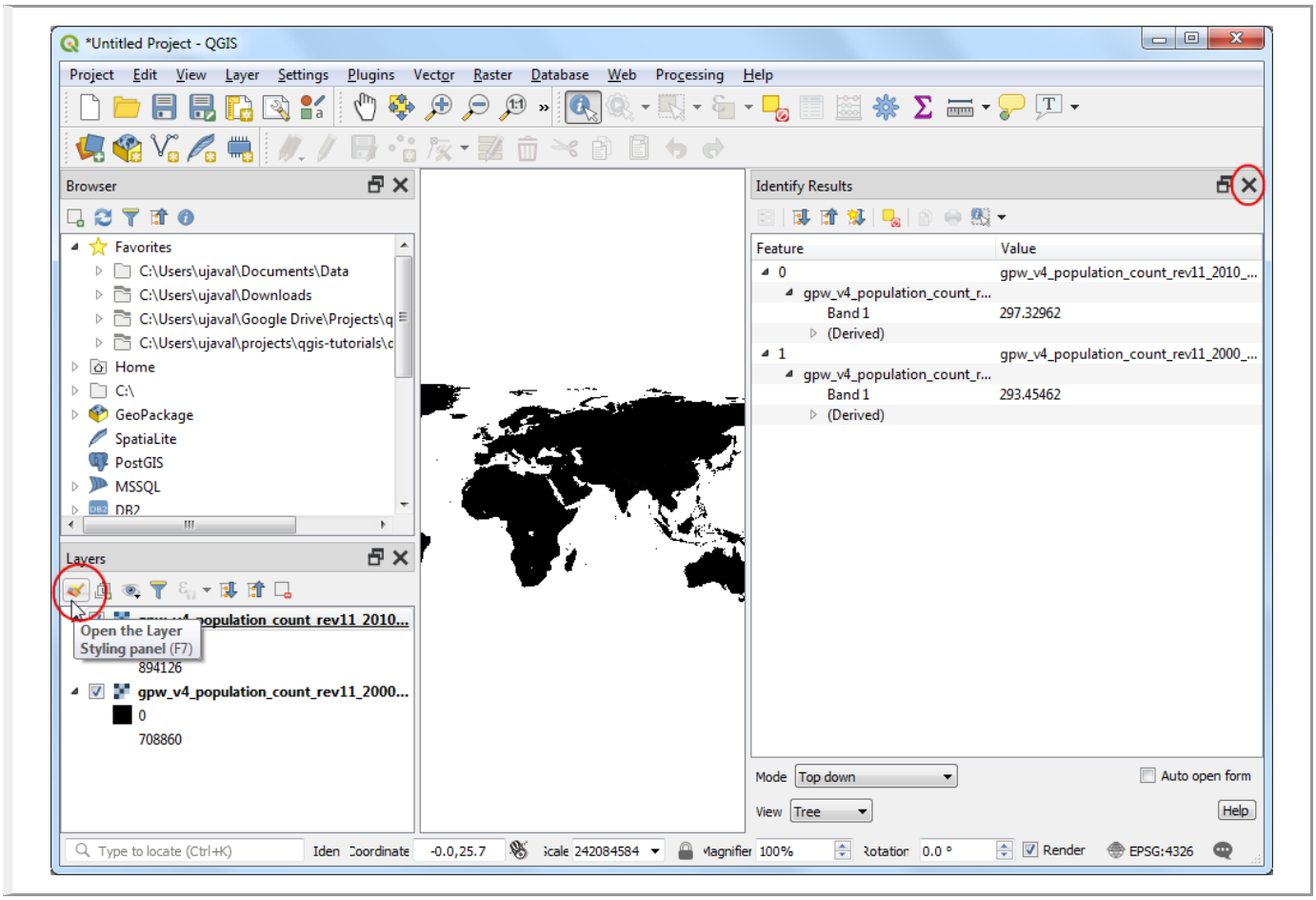
3. Let's explore these layers. Click the Identify button on the Attributes Toolbar. Once the tool is selected, click on any point on the canvas.



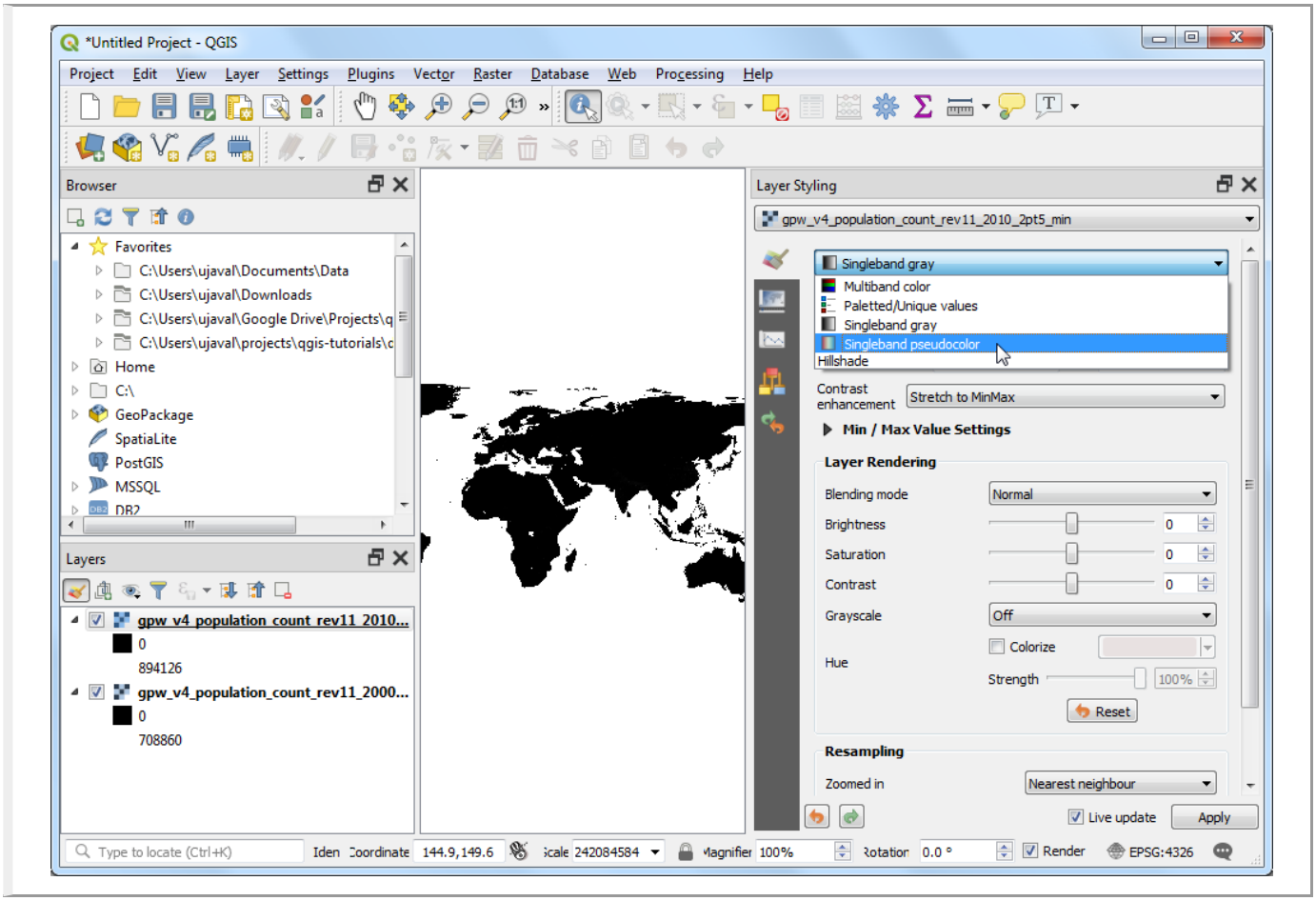
4. The value associated with that pixel will be displayed in a new Identify Results panel. In the Identify Results panel, change the Mode to `Top down`. This will display pixel values of all rasters instead of just the topmost layer. Compare the values from both the layers. As the resolution of the rasters is approximately 5km x 5km, the pixel values represent the total population in the area (25 sq. km) represented by the pixel.



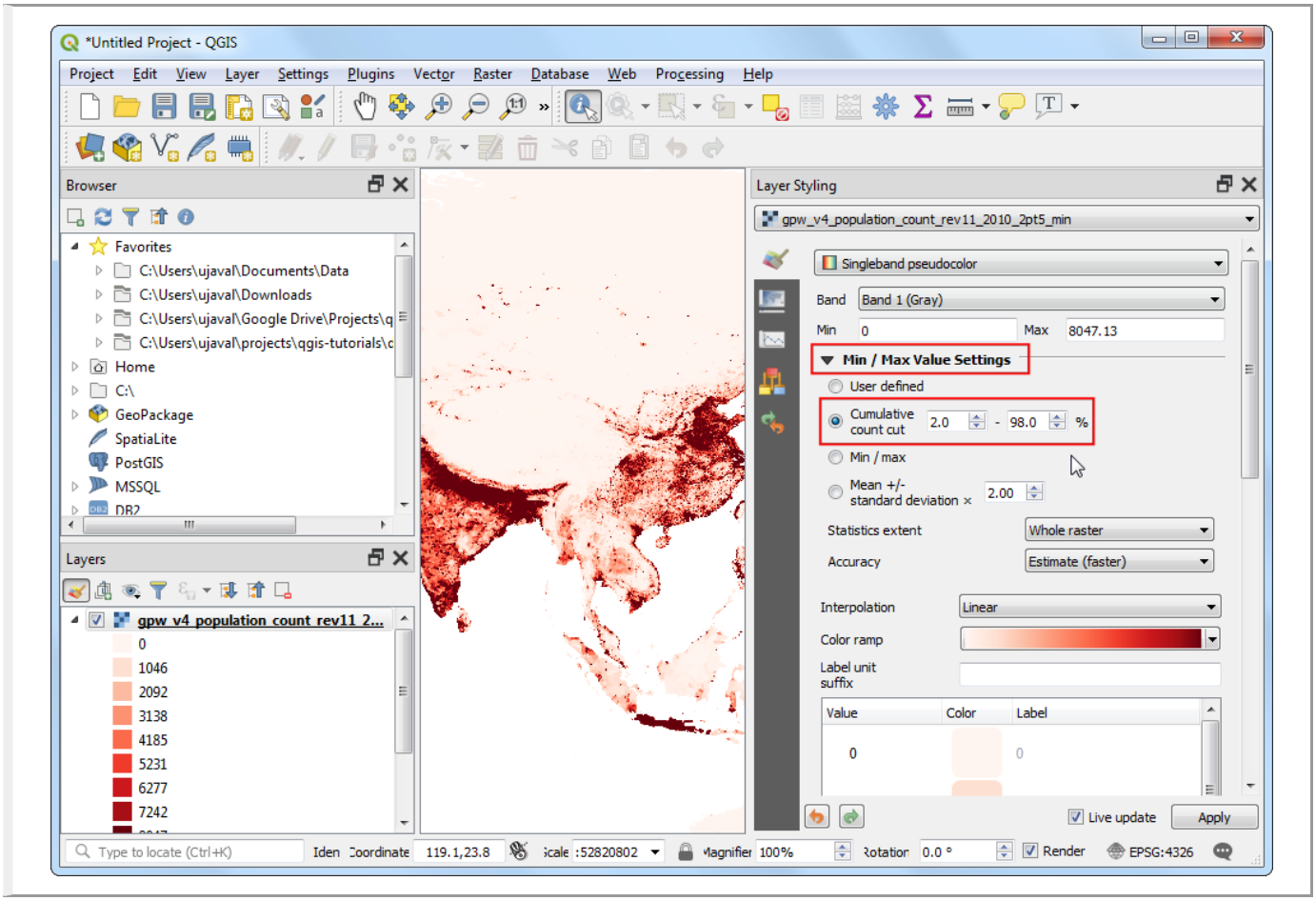
5. Close the Identify Results panel. Let's create a better visualization of the layers. Click the Open the layer Styling panel button in the Layers panel.



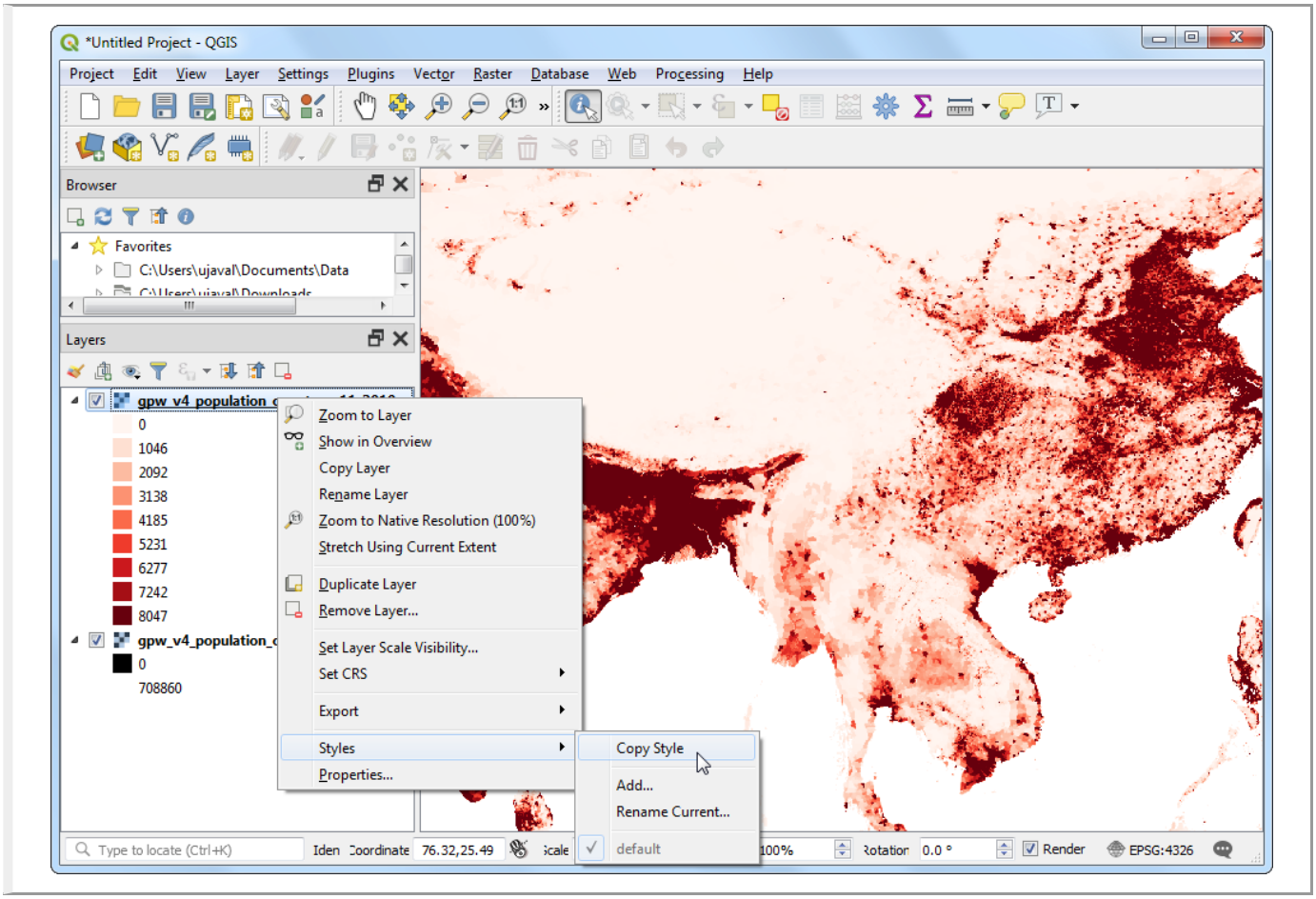
6. In the Layer Styling panel, click the Render type dropdown and select `Singleband pseudocolor` renderer.



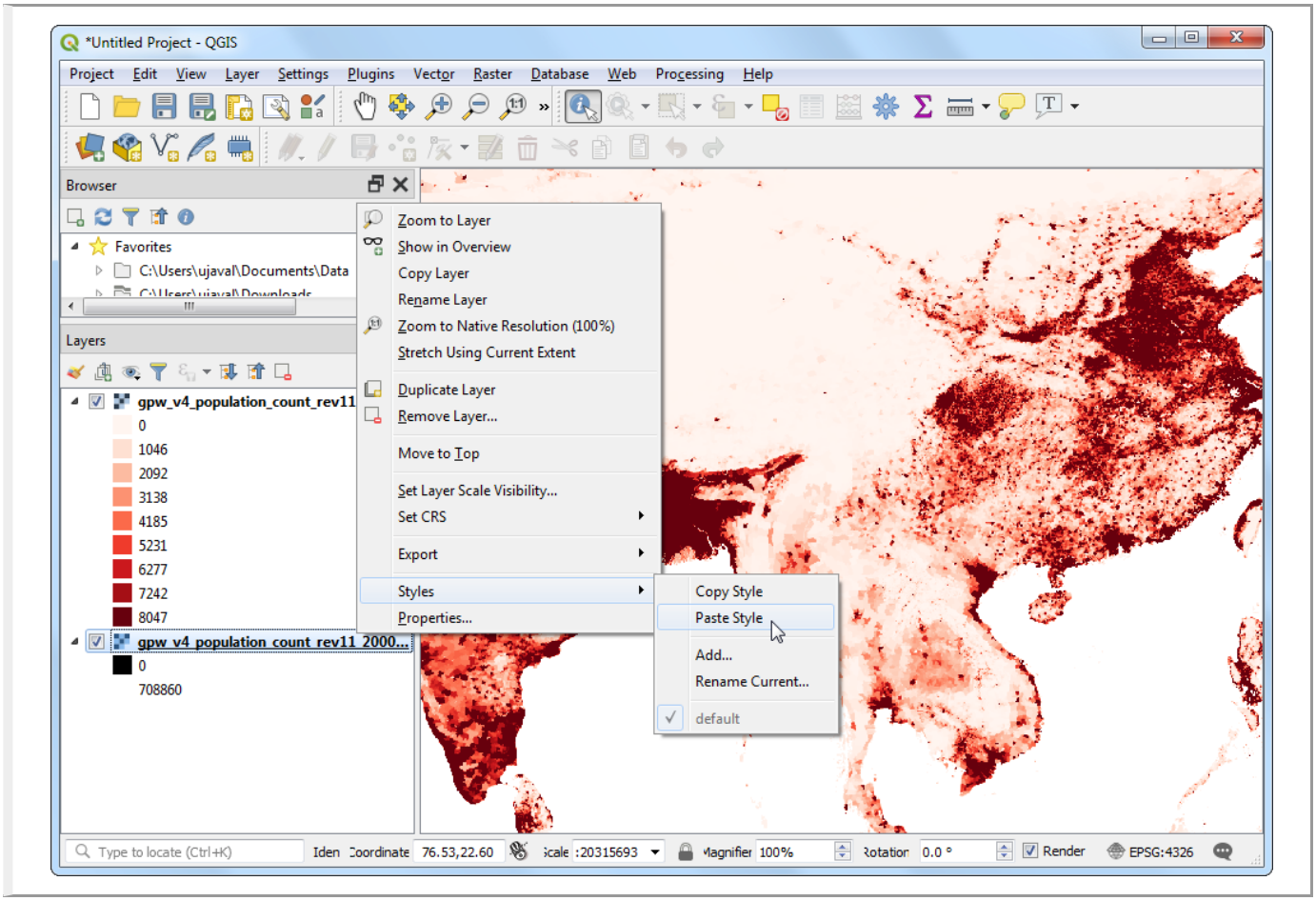
7. This renderer will style the layer using a color ramp. The default color ramp is white-red where the minimum value will be assigned the white color and the maximum value in the layer will be assigned the red color. The intermediate values will be assigned a shade of red linear interpolation. Expand the Min / Max Value Settings and choose `Cumulative count cut` option. You will see that the map visualization is much better now. The The standard data range is set from 2% to 98% of the data values, meaning that the outliers will not be used to set the minimum and maximum values, resulting in a much more representative visualization.



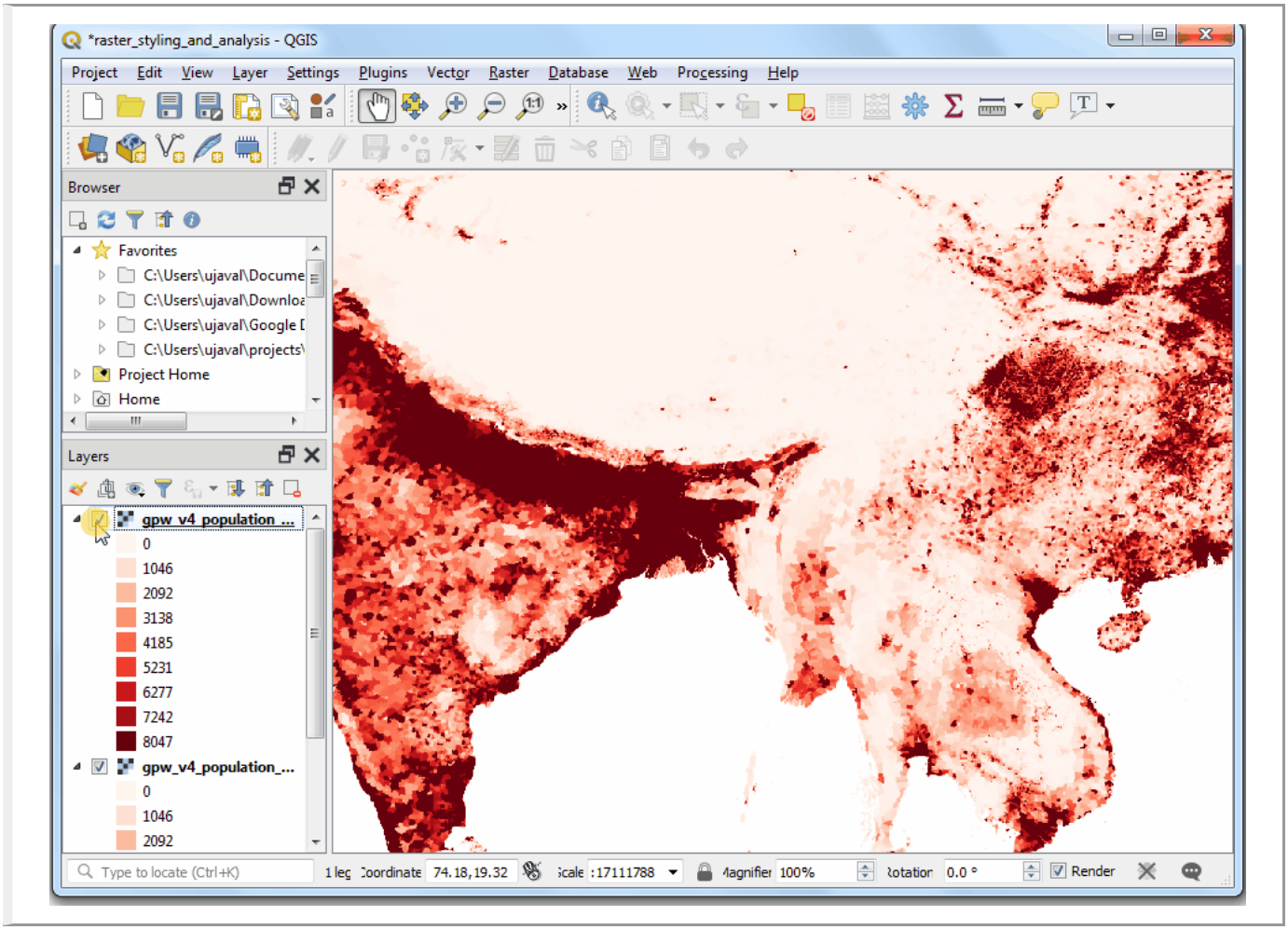
8. Close the Layer Styling panel. We can apply the similar styling to the other layer as well. But there is an easier way to transfer the styles from one layer to the other. Right-click the `gpw-v4-population-count-rev11_2010_2pt5_min` layer and select Styles ‣ Copy Style.



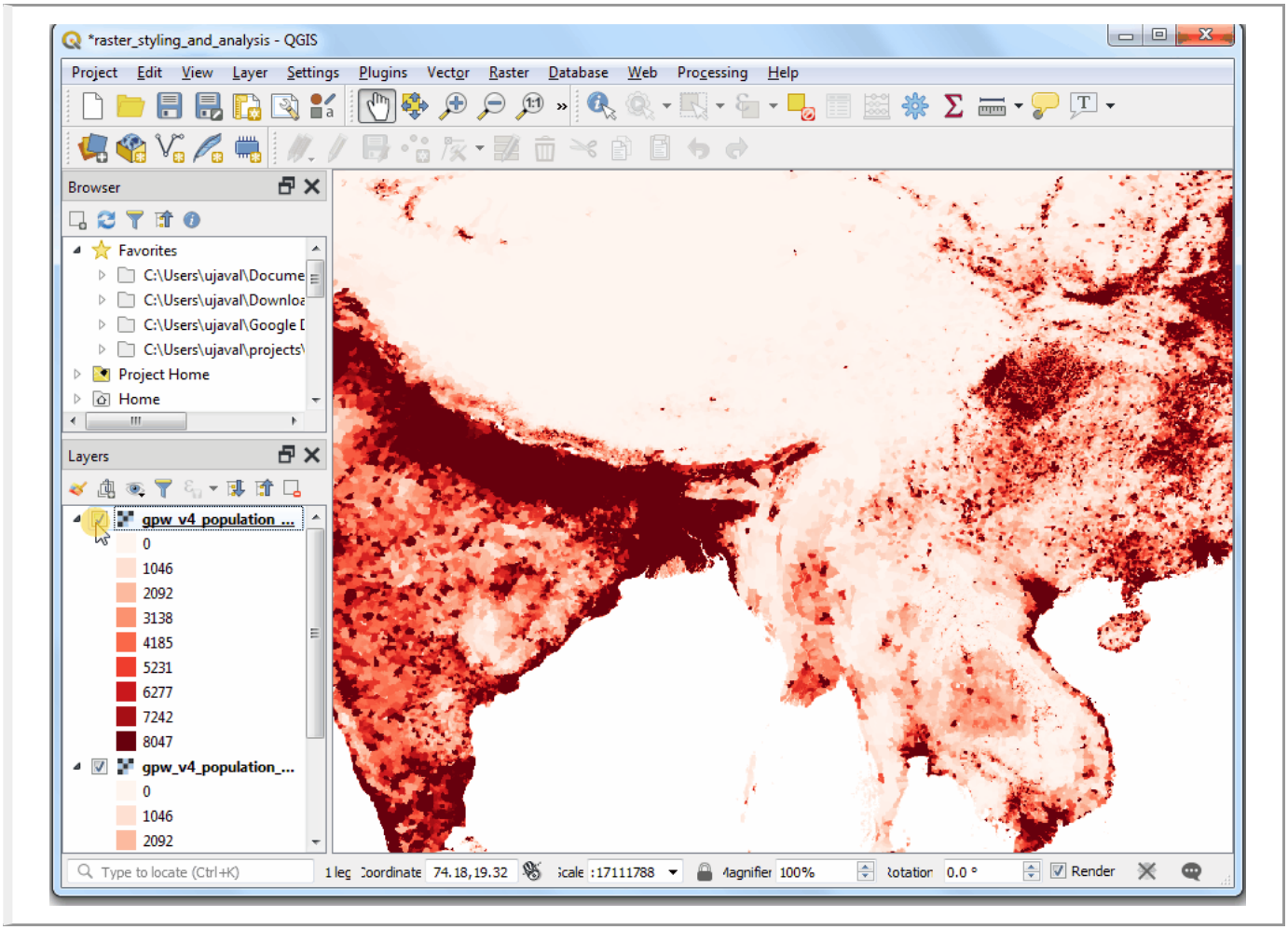
9. Now right-click the un-styled `gpw-v4-population-count-rev11_2000_2pt5_min` layer and select Styles ▸ Paste Style.



10. The same styling parameters will be applied to the other layer. This feature is particularly useful when you want to compare different layers using the same categorization. As you toggle the visibility of the top layer, you can see the changes in population visually.

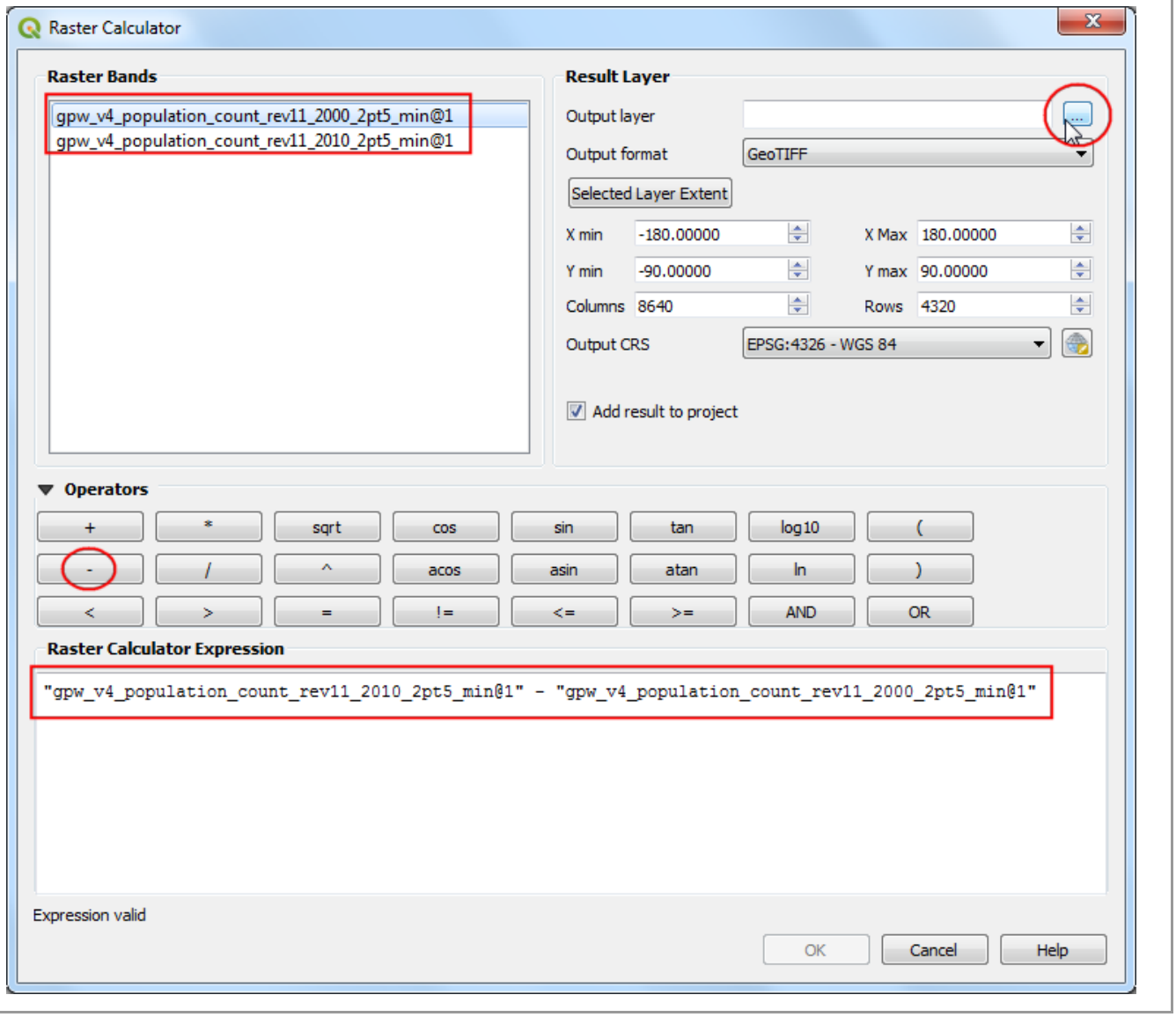


11. Our task is to create a thematic map of the changes in population. Let's compute the difference between the 2 layers and create another raster where each pixel represents the change in the population. Go to Raster ▸ Raster calculator.

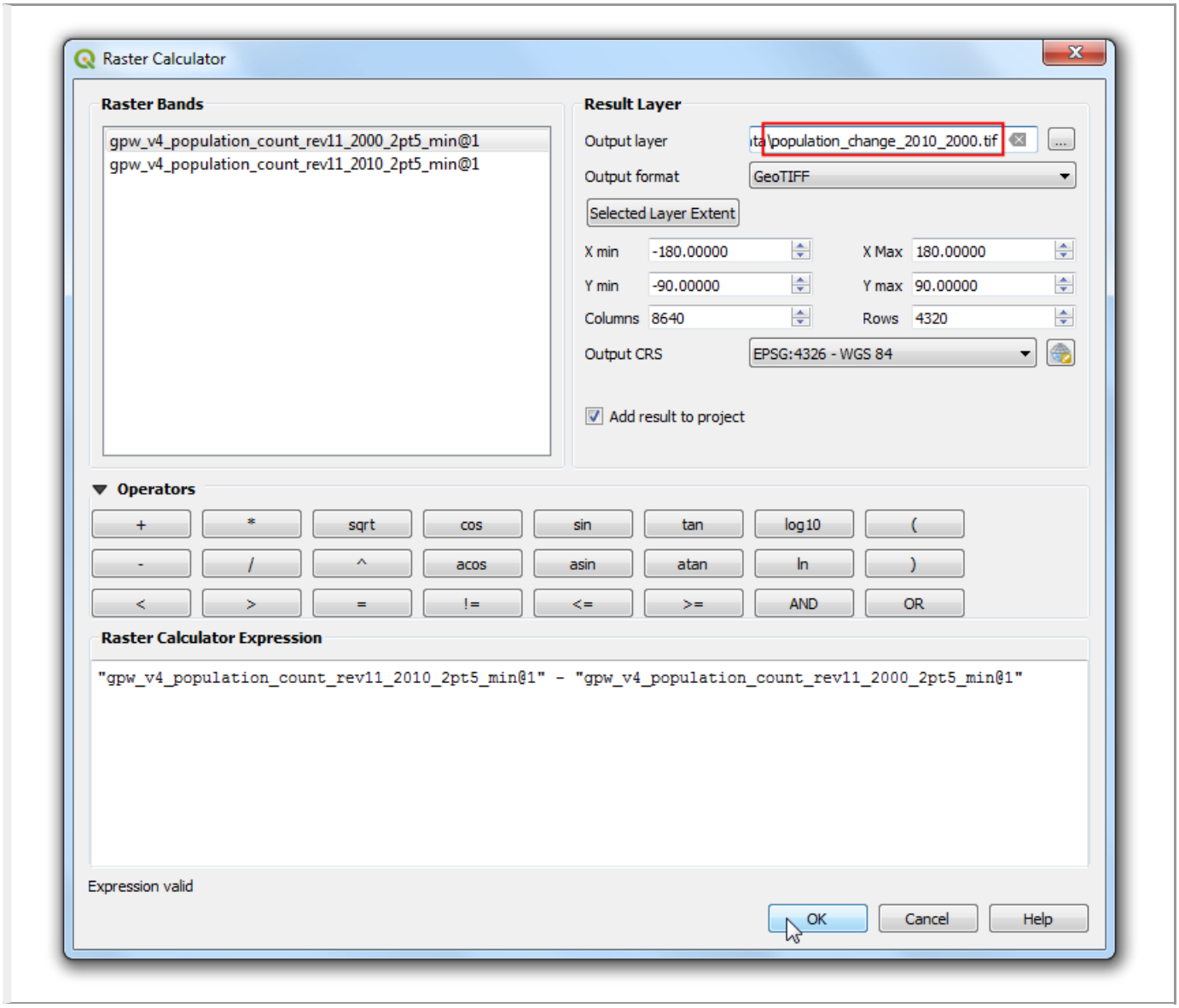


12. In the Raster bands section, you can select the layer by double-clicking on them. The bands are named after the raster name followed by `@` and band number. Since each of our rasters have only 1 band, you will the names with `@1` appended to the layer name. The raster calculator can apply mathematical operations on the raster pixels. In this case we want to enter a simple formula to subtract the 2010 population from 2000. Enter the following expression. Next, click the ... button next to Output layer.

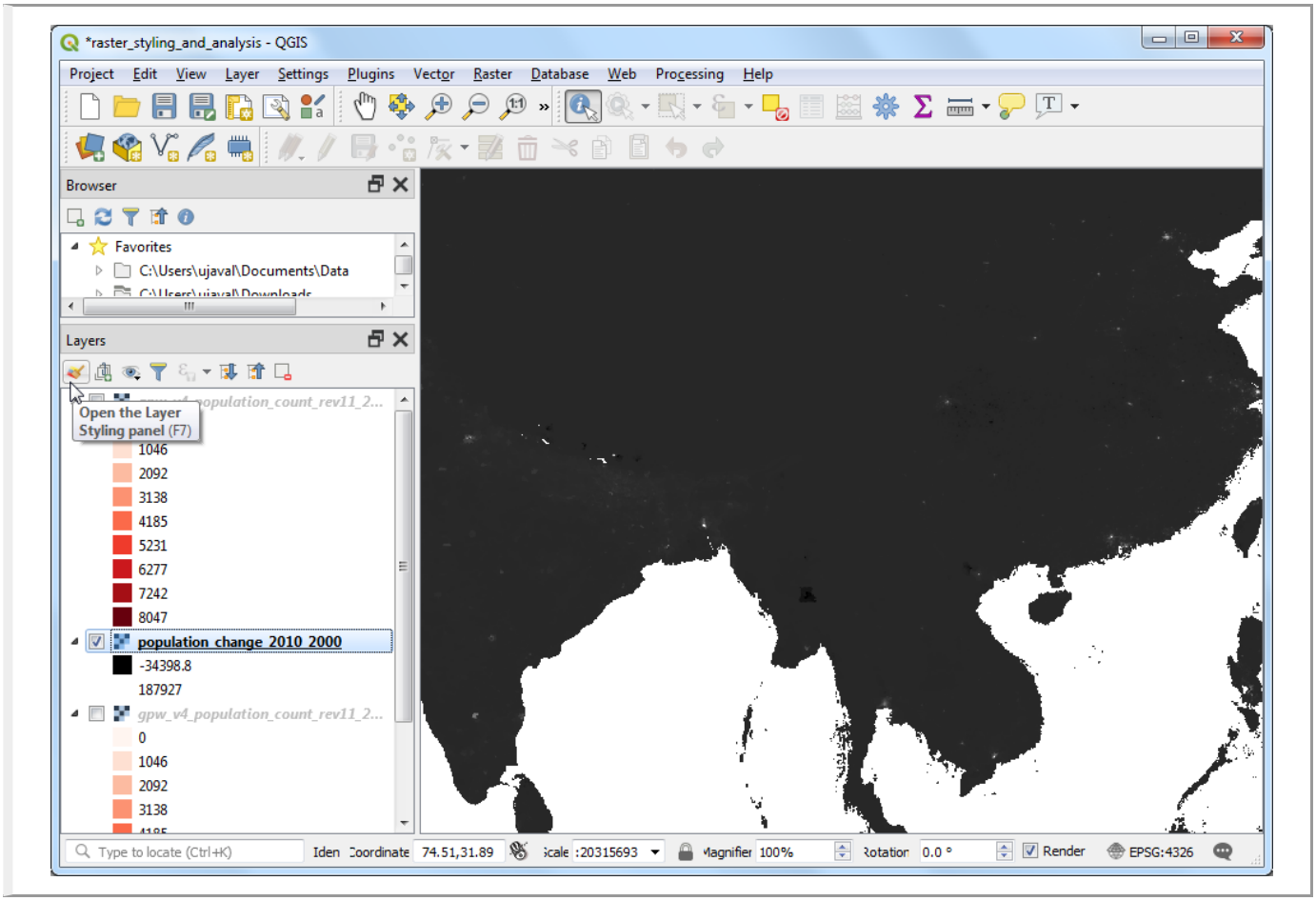
```
"gpw-v4-population-count-rev11_2010_2pt5_min@1" - "gpw-v4-population-count-rev11_2000_2p
```



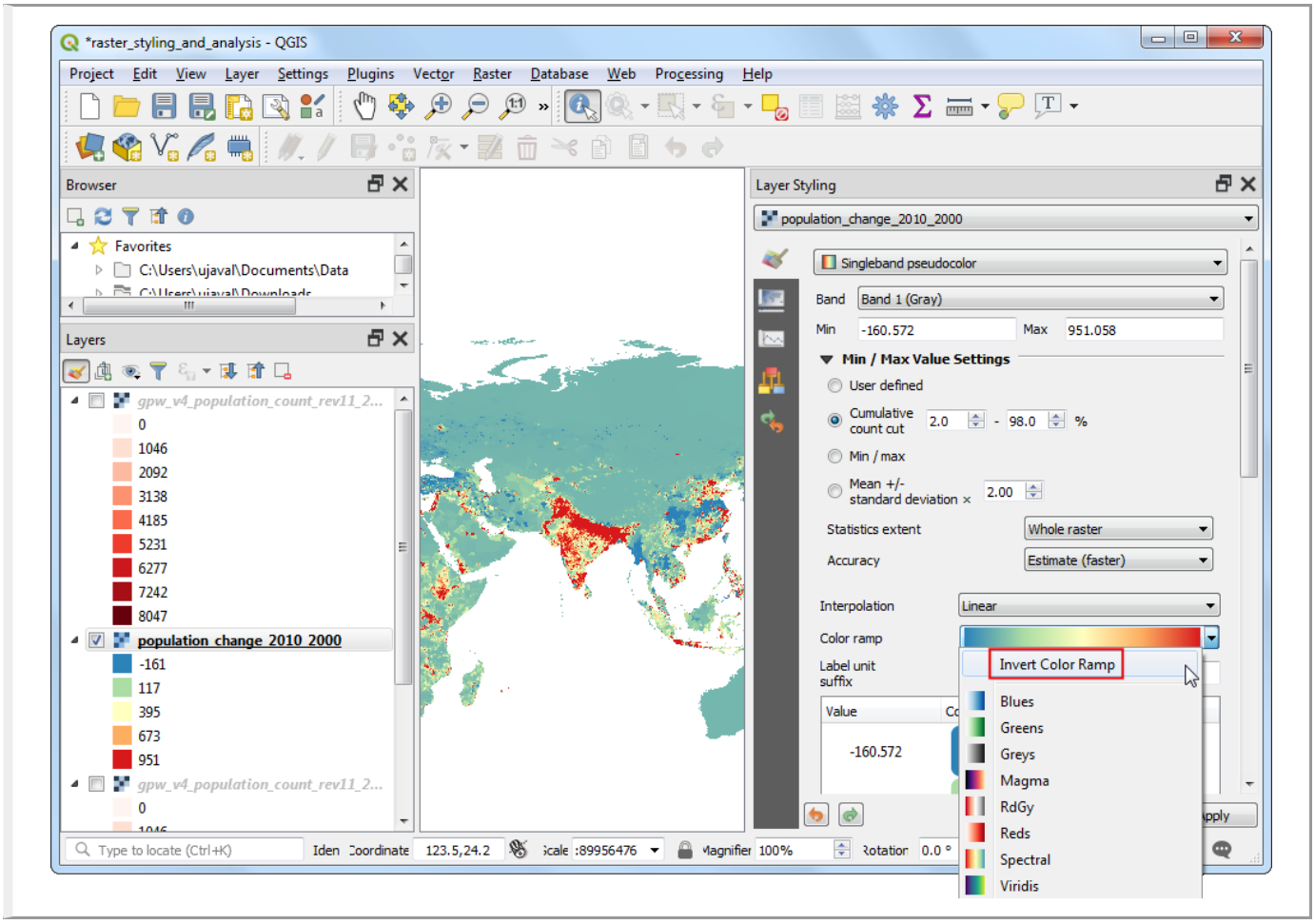
13. Enter `population_change_2010_2000.tif` as the Output file. Click OK to start the computation.



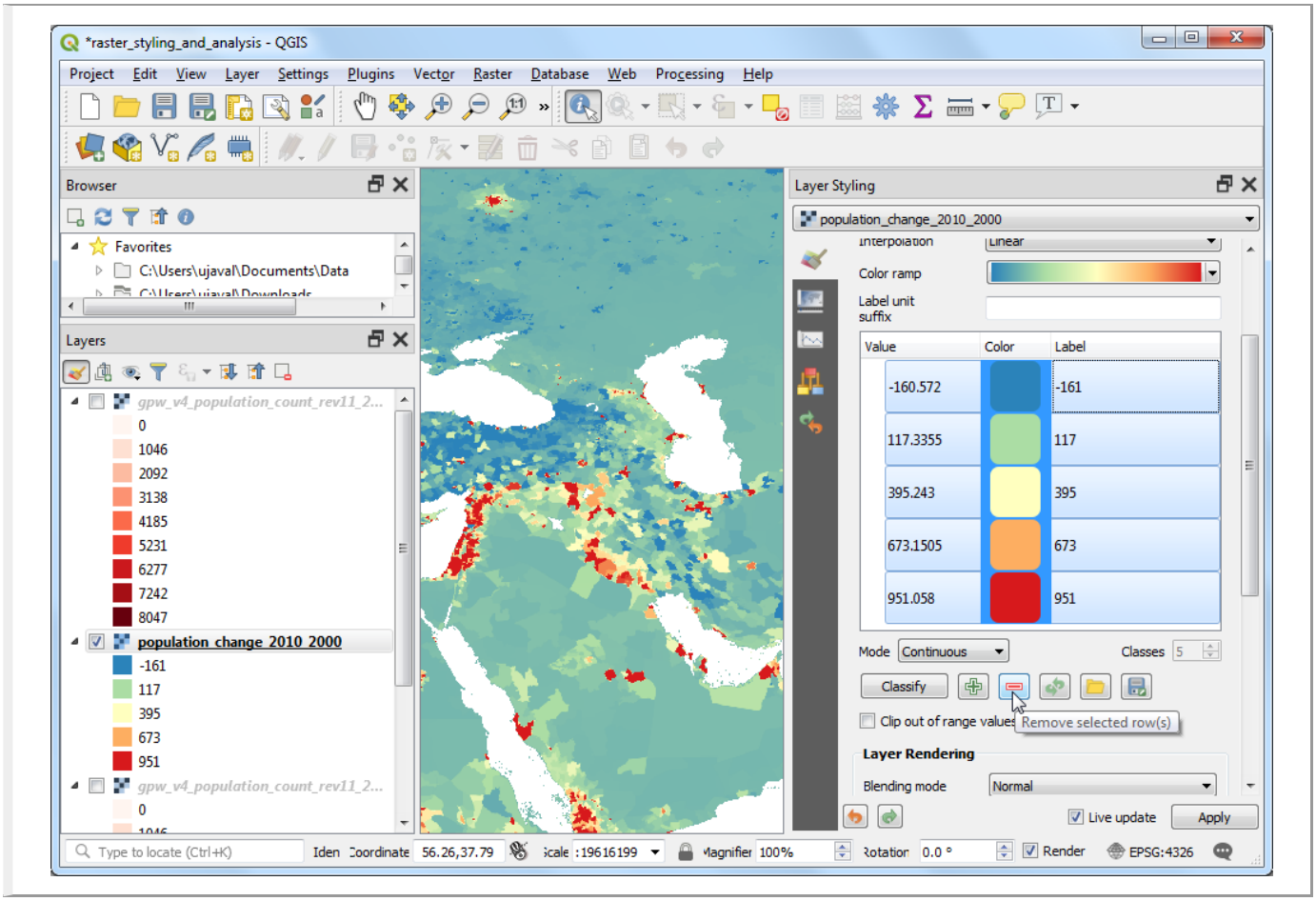
14. Once completed a new layer `population_change_2010_2000` will be added to the Layers panel. Let's change the styling so that the negative and positive population changes are better visualized. Click the Open the layer Styling panel button in the Layers panel.



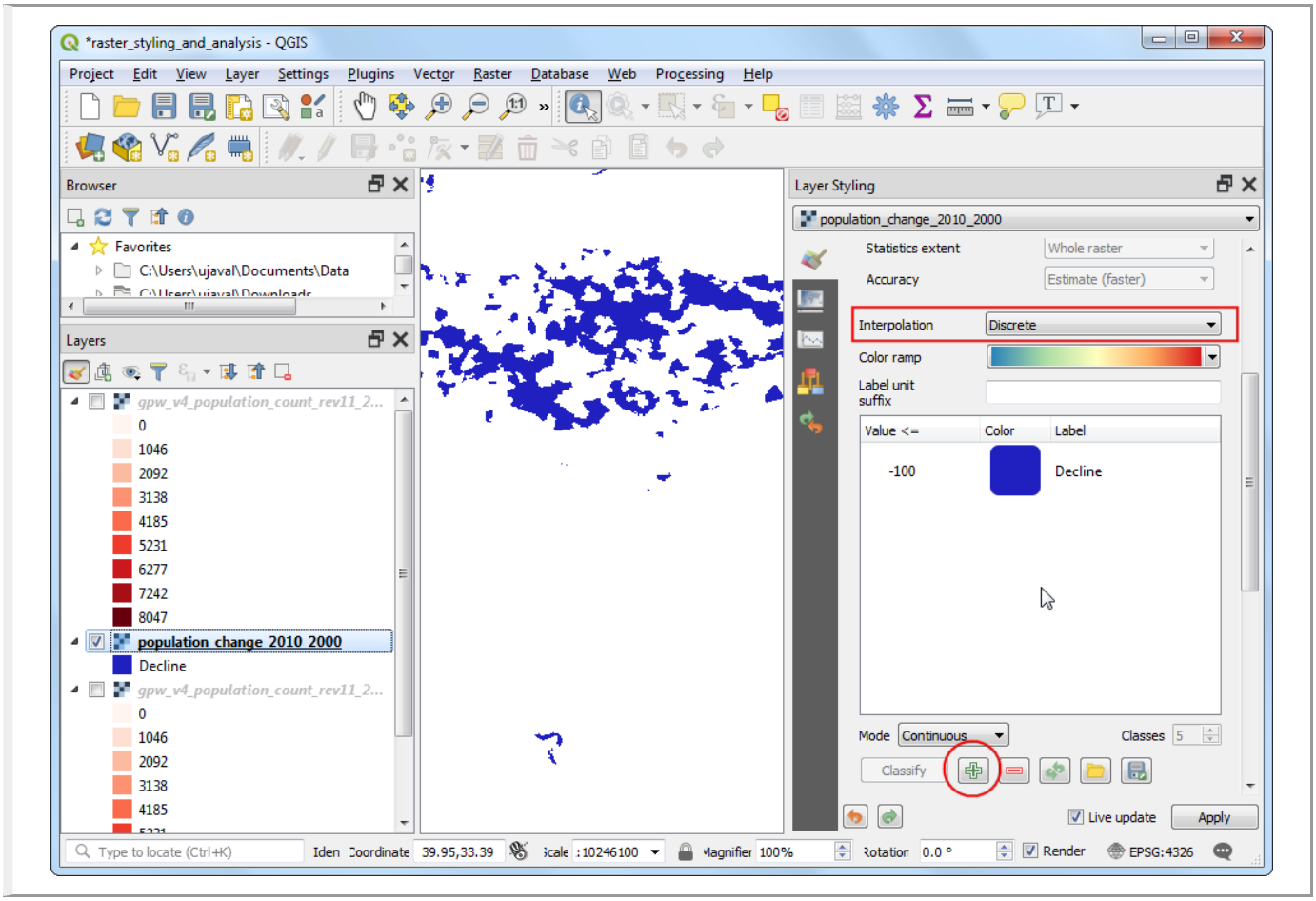
15. One option is to use the similar styling technique as earlier and choose a diverging color ramp. Click the Color ramp drop-down and select `Spectral` ramp. Click the dro-down again and choose `Invert Color Ramp` to assign blues to low values and reds to high values.



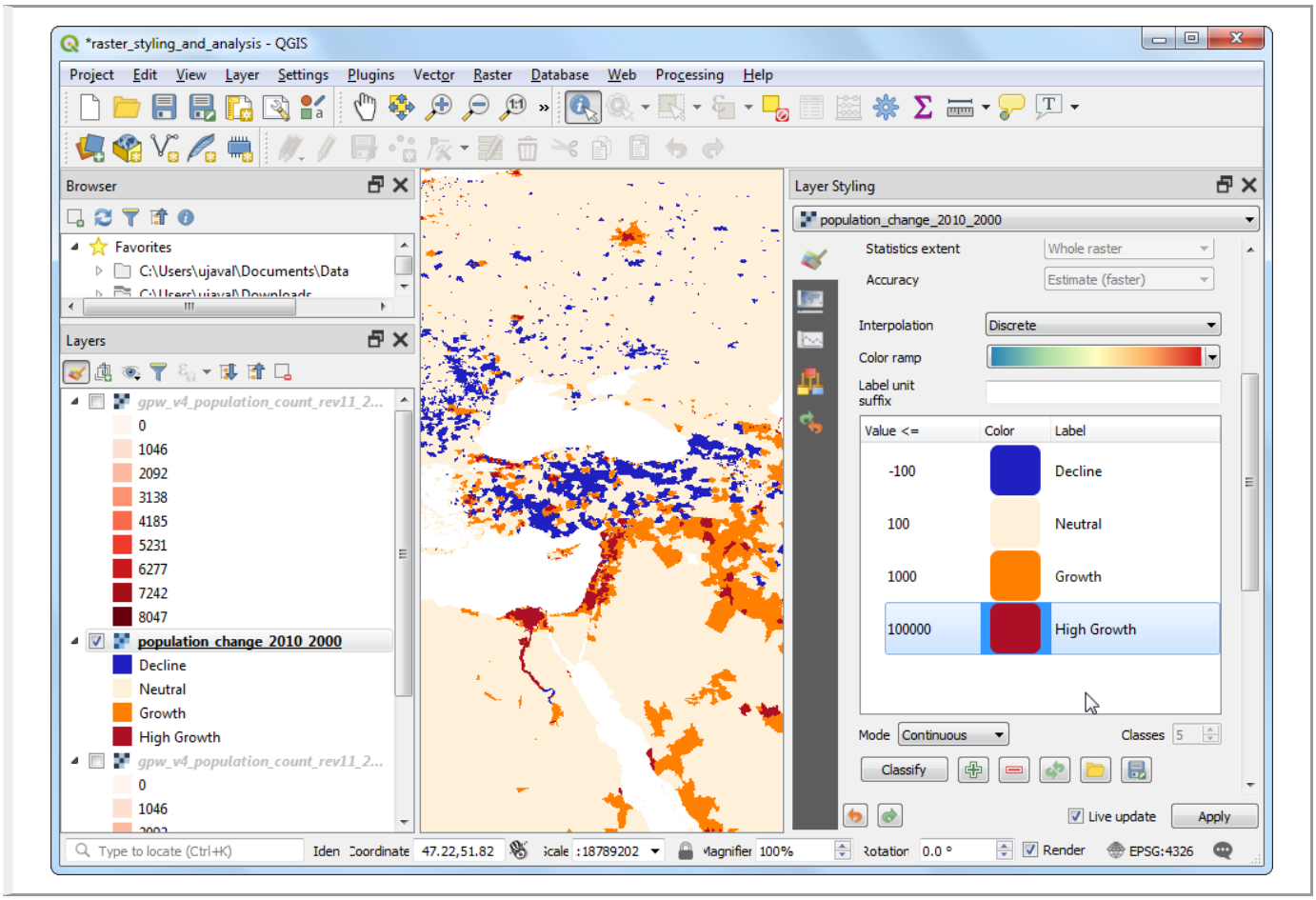
16. This is a good visualization, but not easy to interpret. Let's create a better map with 4 discrete categories, `Decline`, `Neutral`, `Growth` and `High Growth`. Scroll down to the tables with classes. Hold the `Shift` key and select all the rows. Click the Remove selected row(s) button.



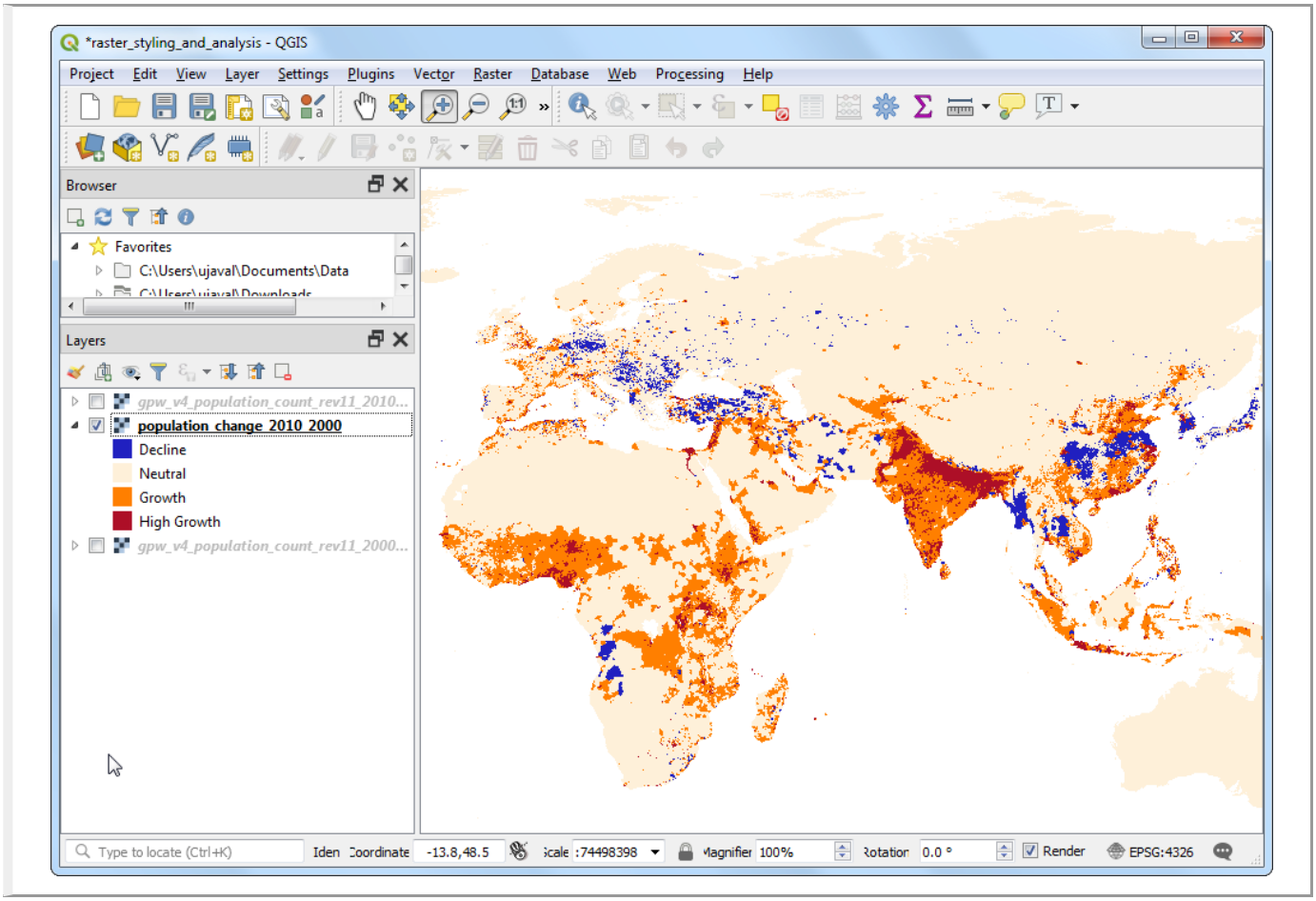
17. Change the Interpolation mode to `Discrete`. We will now create a color map manually. Click the Add values manually button. Enter `-100` as the Value and `Decline` as the Label. Assign blue color to this category. The way color map works is that all values lower than the value entered will be given the color of that entry. You will notice the canvas will show only those areas with negative population change.



18. Complete the color-map with suitable values. I chose `100`, `1000` and `100000` as the upper-bounds for the `Neutral`, `Growth` and `High Growth` categories respectively.



19. Once you are satisfied with the visualization, close the Layer Styling panel. You now have a global thematic map of population change.



Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Raster Mosaicing and Clipping (QGIS3)

This tutorial explores basic techniques for working with rasters in QGIS such as mosaicing and subsetting.

Overview of the task

We will download elevation data for Sri Lanka in form of SRTM tiles, merge them and clip the resulting mosaic to the country boundary.

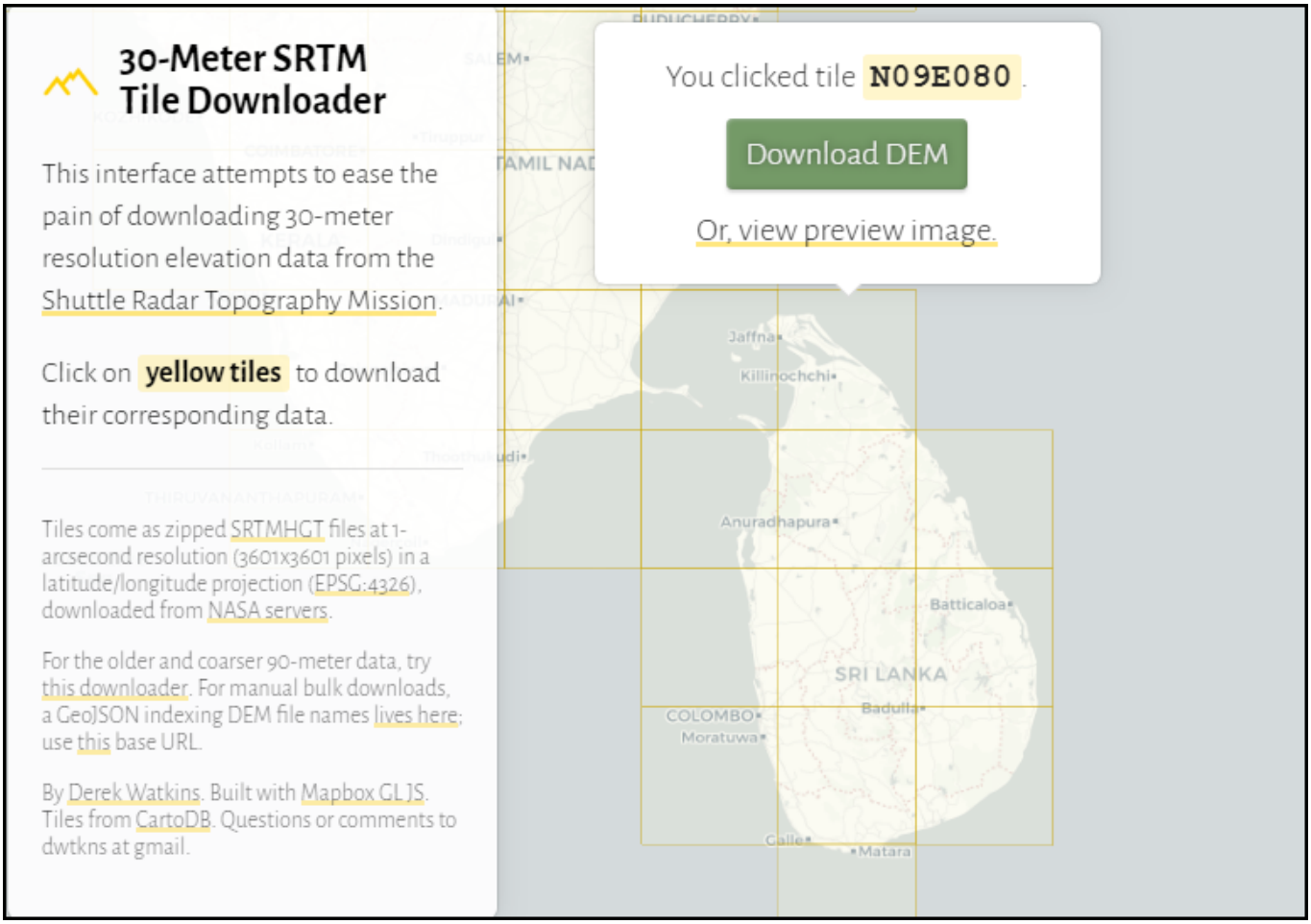
Other skills you will learn

- Using the Hillshade renderer to visualize elevation data.

Get the data

Land Processes Distributed Active Archive Center (LP DAAC) provides NASA Shuttle Radar Topography Mission (SRTM) Global 1 arc second (https://lpdaac.usgs.gov/products/srtmgl1v003/) dataset as elevation tiles.

An easy interface to download tiles for a given area is the 30-Meter SRTM Tile Downloader (https://dwtkns.com/srtm30m/) by By Derek Watkins. Download the individual SRTM tiles covering Sri Lanka. Note that you will need a free Earth Data account (https://urs.earthdata.nasa.gov/home) to download the data.



We will also need the Admin 0 - Countries (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_admin_0_countries.zip) shapefile from Natural Earth.

For convenience, you may directly download a copy of the datasets from the links below:

N05E080.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N05E080.SRTMGL1.hgt.zip)

N06E079.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N06E079.SRTMGL1.hgt.zip)

N06E080.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N06E080.SRTMGL1.hgt.zip)

N06E081.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N06E081.SRTMGL1.hgt.zip)

N07E079.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N07E079.SRTMGL1.hgt.zip)

N07E080.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N07E080.SRTMGL1.hgt.zip)

N07E081.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N07E081.SRTMGL1.hgt.zip)

N08E079.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N08E079.SRTMGL1.hgt.zip)

N08E080.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N08E080.SRTMGL1.hgt.zip)

N08E081.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N08E081.SRTMGL1.hgt.zip)

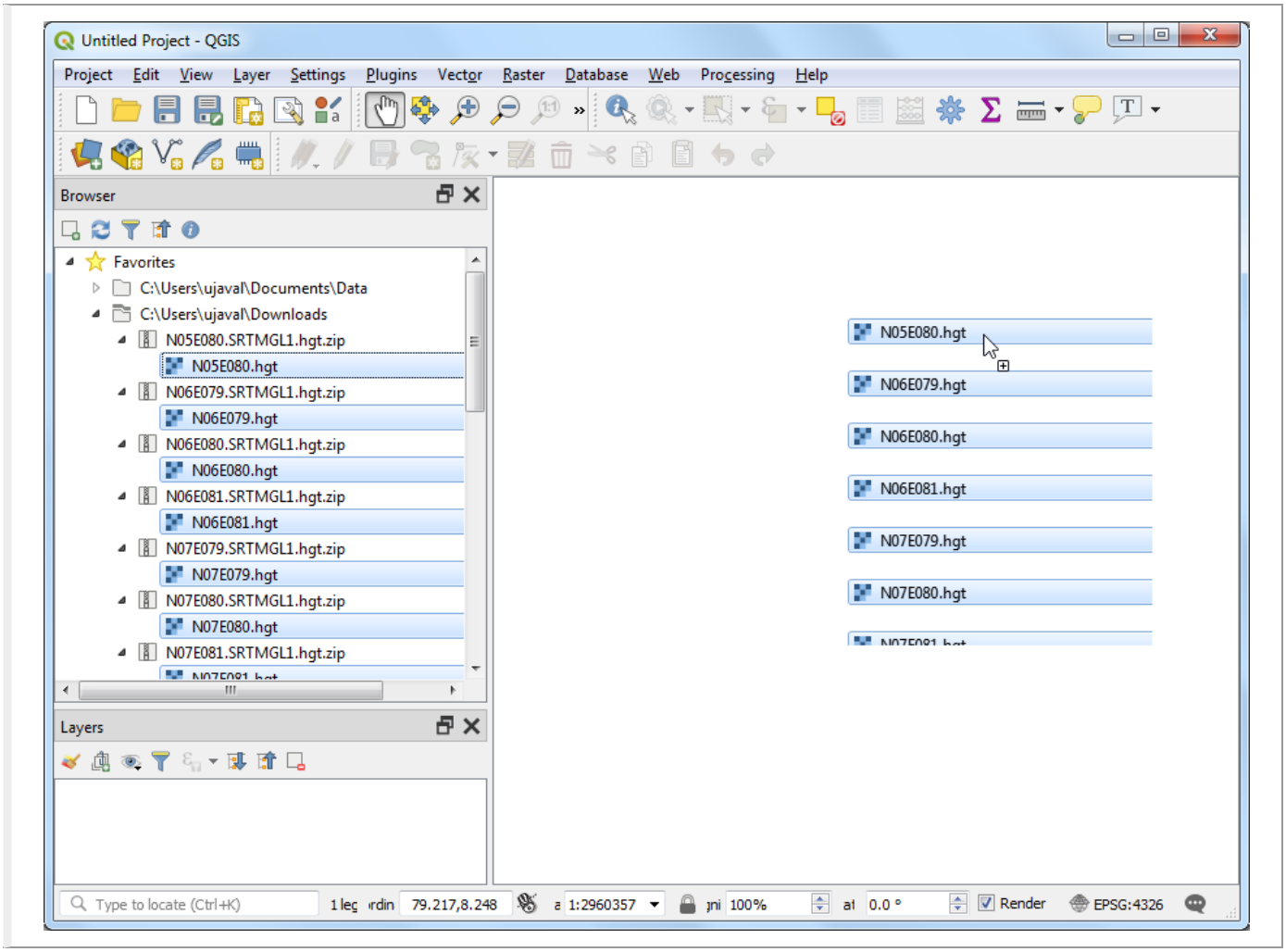
N09E080.SRTMGL1.hgt.zip (http://www.qgistutorials.com/downloads/N09E080.SRTMGL1.hgt.zip)

ne_10m_admin_0_countries.zip (http://www.qgistutorials.com/downloads/ne_10m_admin_0_countries.zip)

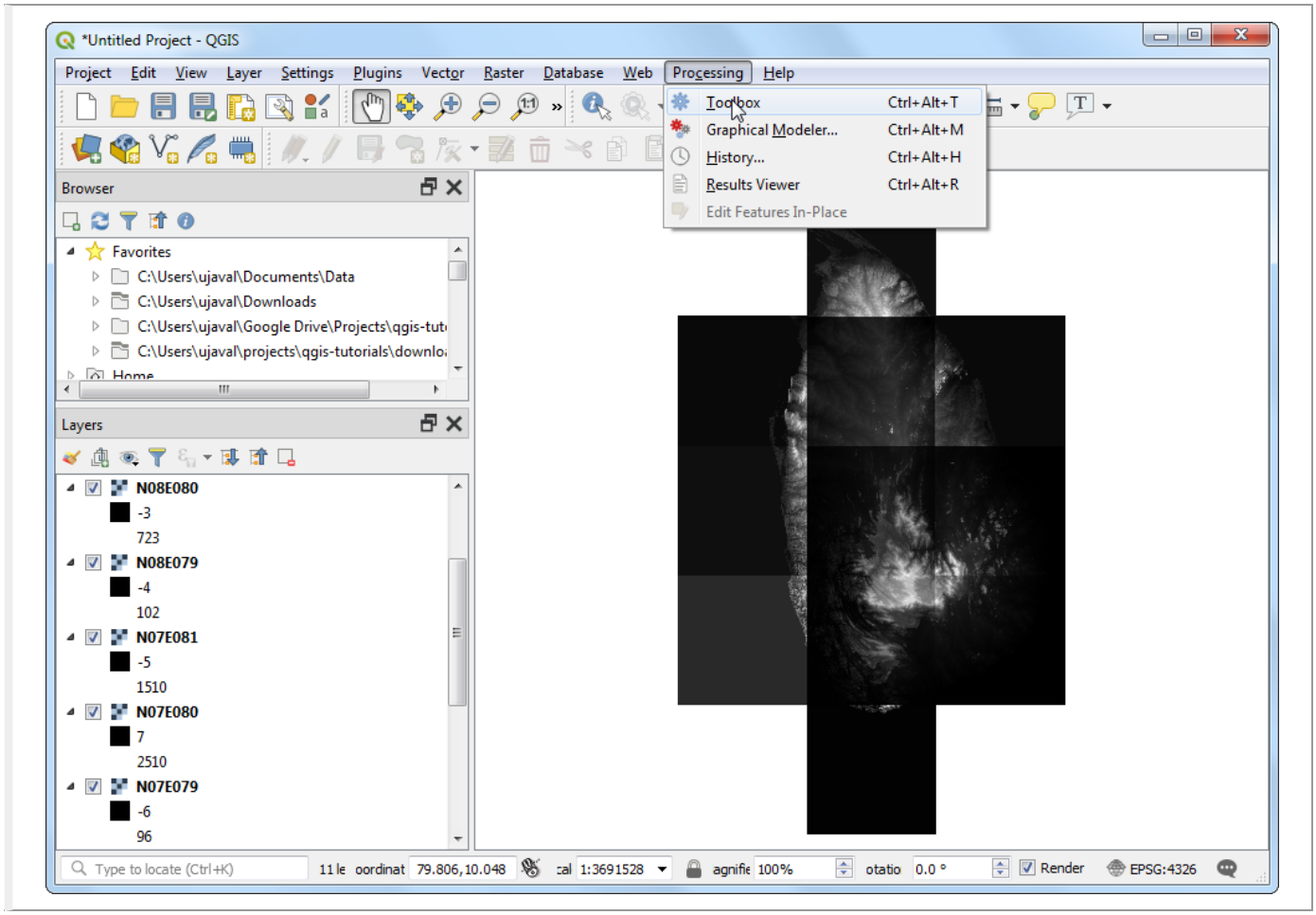
Data Source [SRTM] (../credits.html#srtm) , [NATURALEARTH] (../credits.html#naturalearth)

Procedure

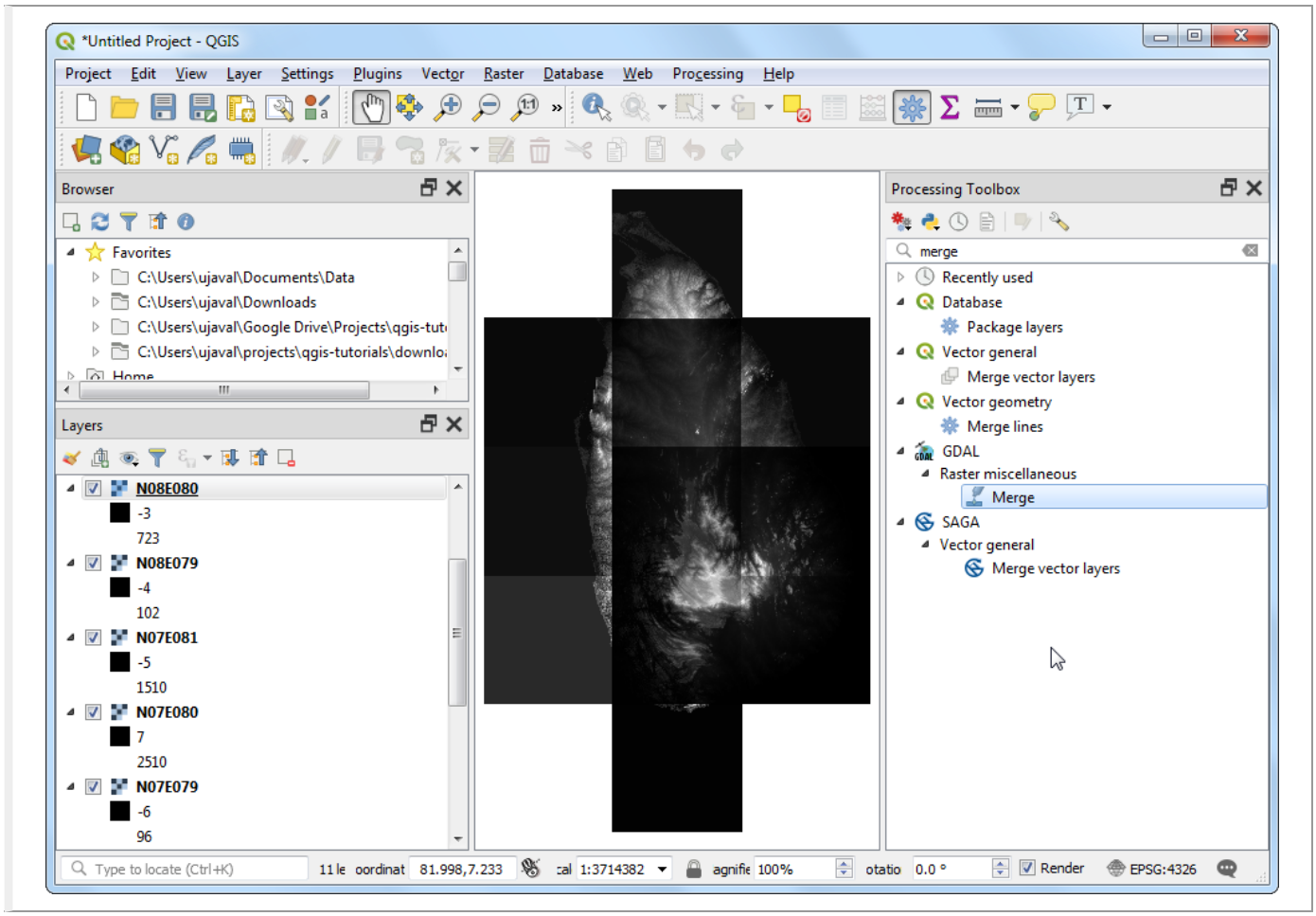
1. Open QGIS and locate the downloaded files in the Browser panel. Expand individual zip files to show the `.hgt` files. Hold the `Ctrl` key and select all individual files. Once selected, drag them to the canvas.



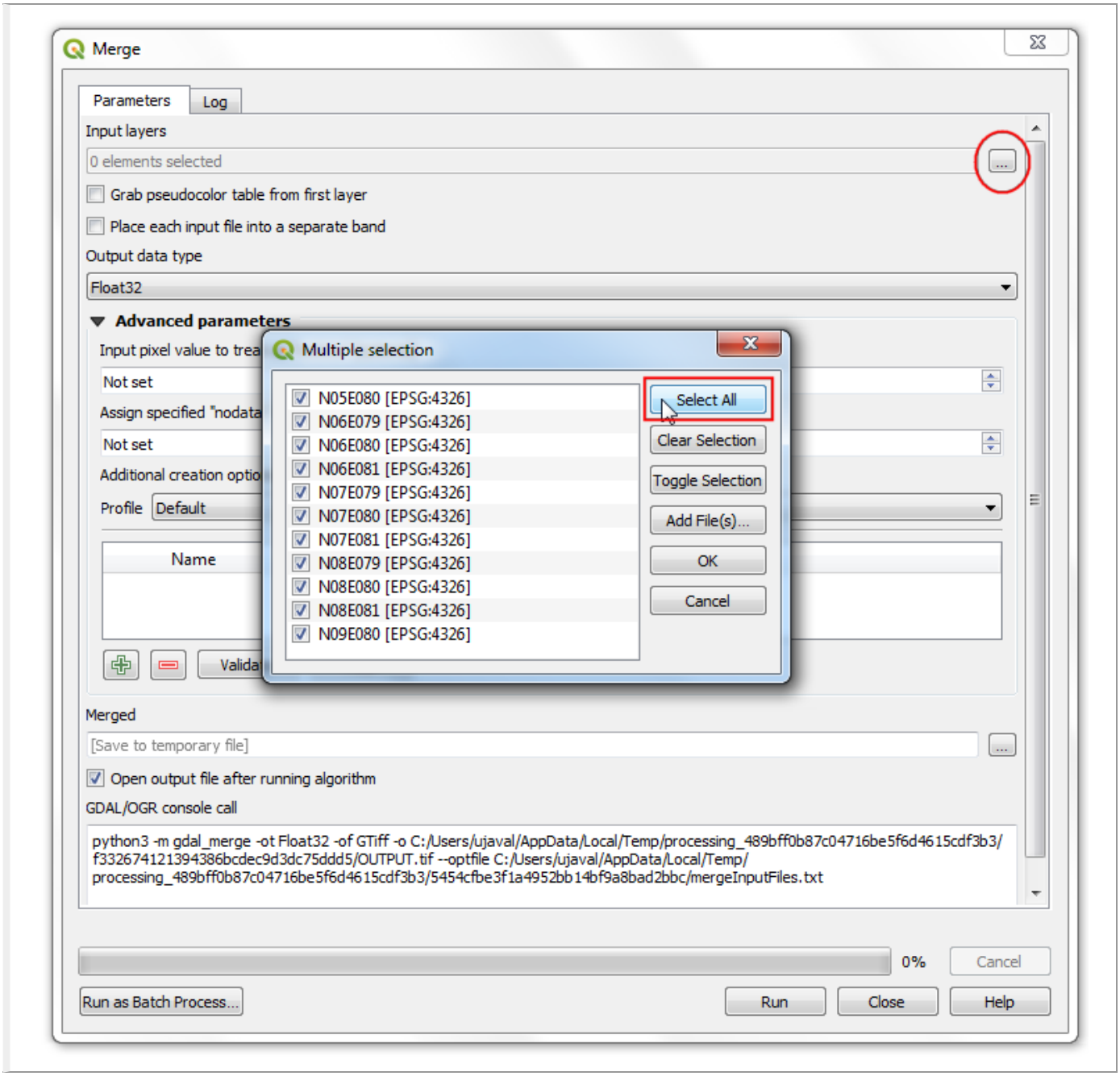
2. You will see 11 individual layers loaded in the Layers panel and displayed in the canvas. We will merge these individual tiles into a single mosaic. Go to Processing ‣ Toolbox.



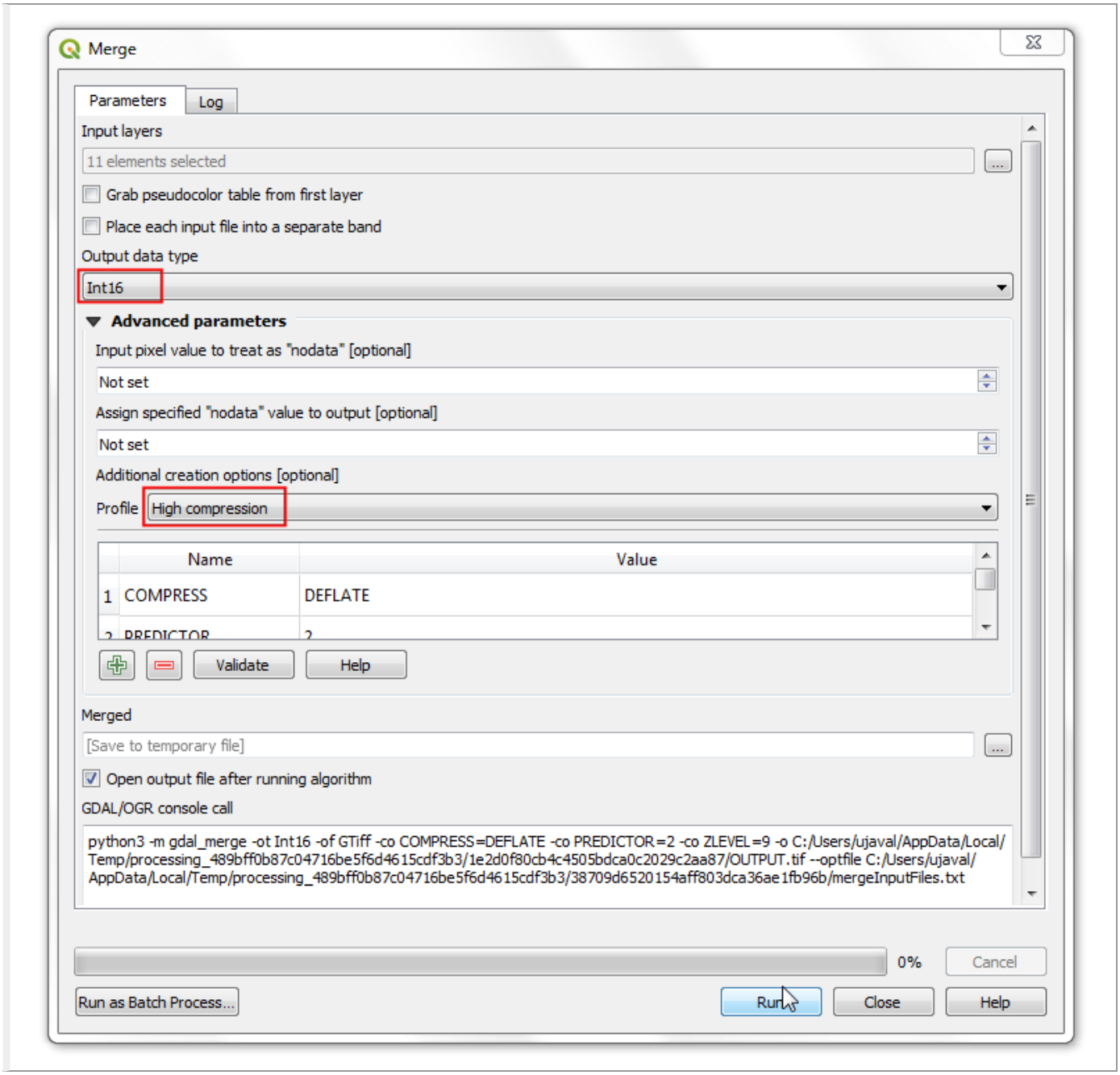
3. Search and locate the GDAL ‣ Raster miscellaneous ‣ Merge tool. Double-click to launch it.



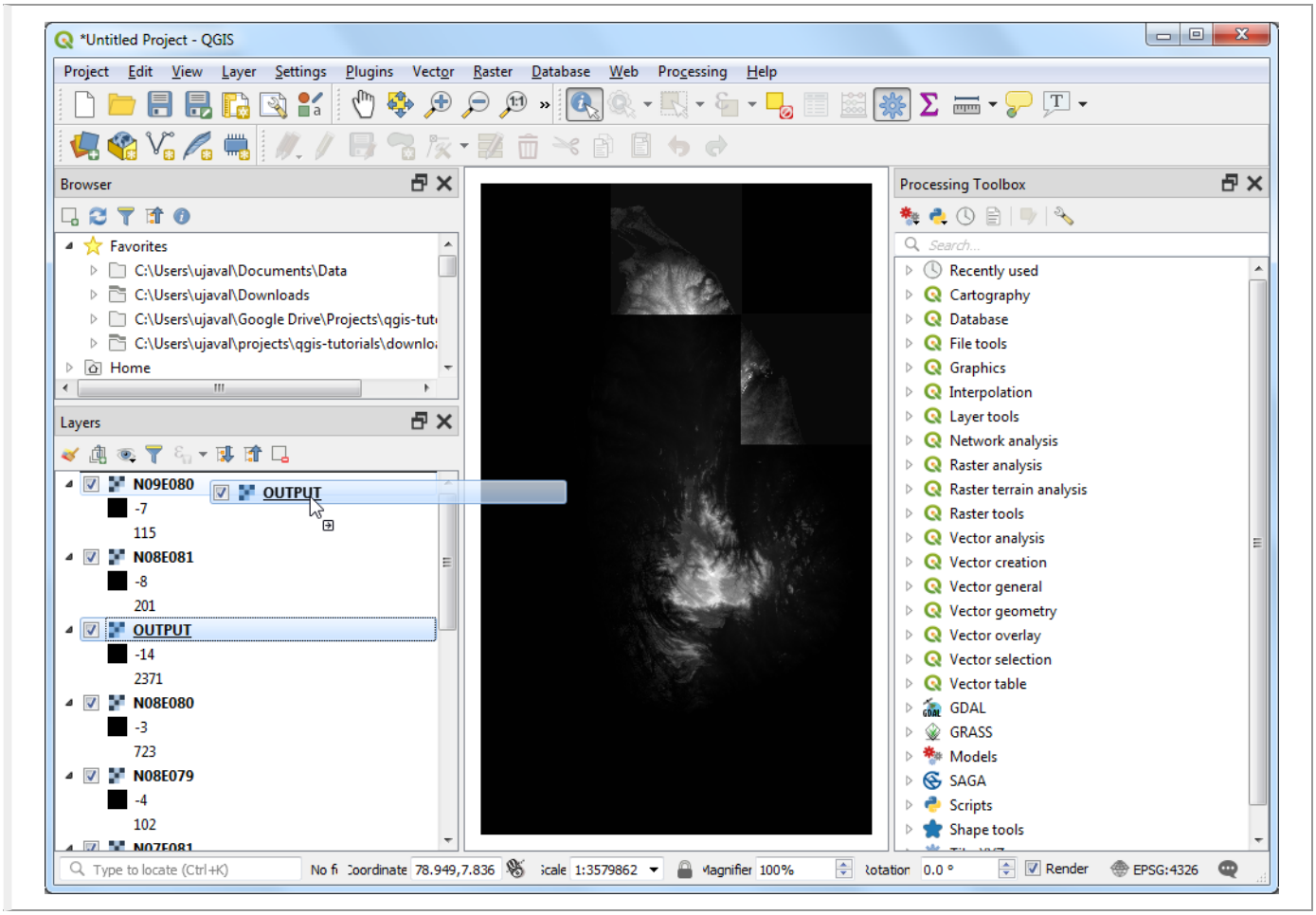
4. In the Merge dialog, click the ... button next to Input layers. Click Select All to select all individual layers.



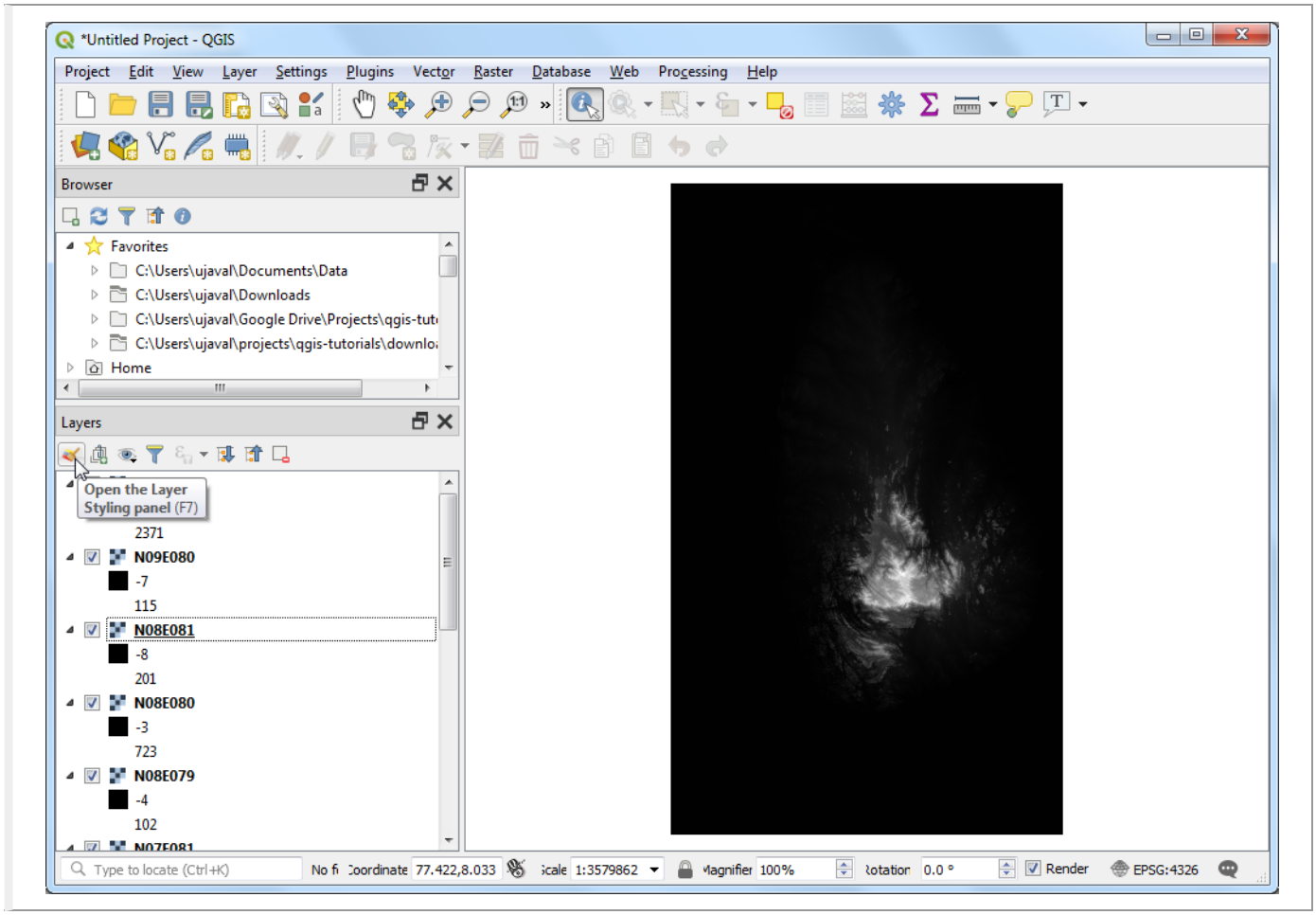
5. As mentioned in the dataset layer details (https://lpdaac.usgs.gov/products/srtmgl1v003/), the input data type is *16-bit signed integer*. To maintain data integrity, we should keep the same data type for the merged layer. Select `Int16` as the Output data type. Also the default output data format is GeoTiff. GeoTiff files can get very large if not compressed. Choose `High Compression` as the Profile. Click Run.



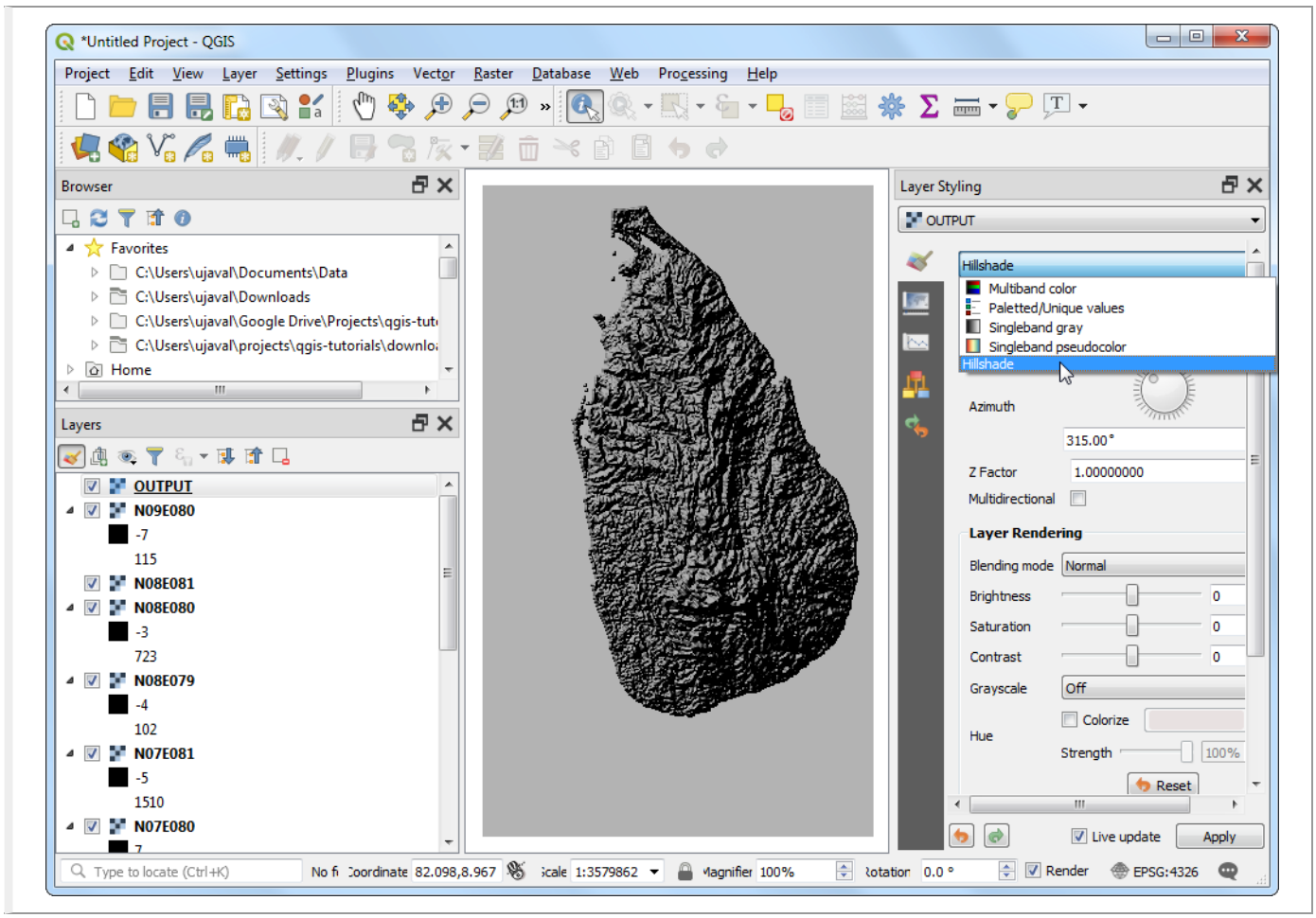
6. Once the processing finishes, the a new layer `OUTPUT` will be added to the Layers panel. In case the layer is not at the top of the stack, select it and drag it to the top of the Layers panel.



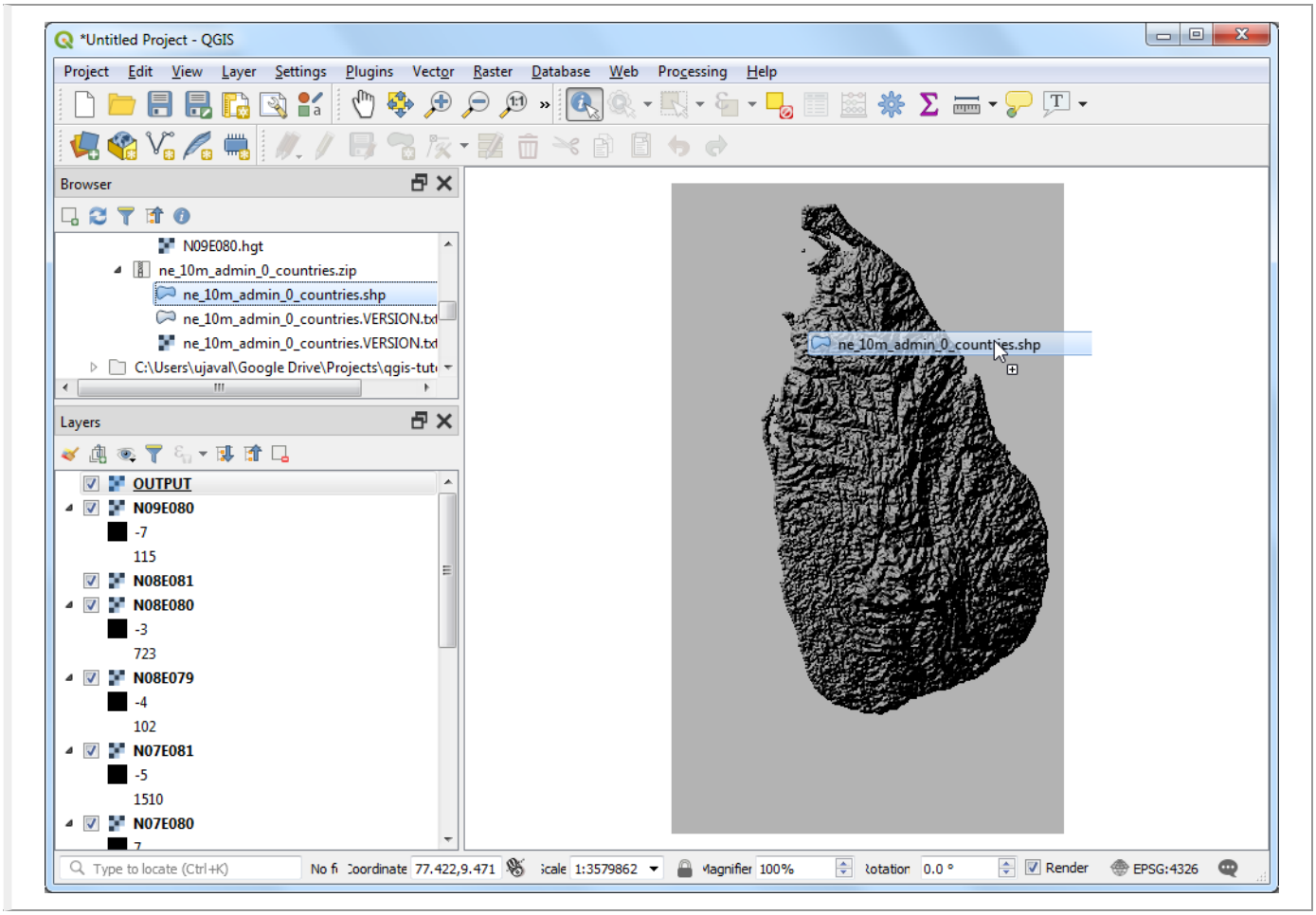
7. You will see that the `OUTPUT` layer contains the merged elevation data from the individual input tiles. The default visualization only shows the pixel values in the range from 0-255. But our data contains pixels with values -14 to 2371, resulting in a low contrast rendering. Let's change it a better visualization. Click the Open the layer Styling panel button in the Layers panel.



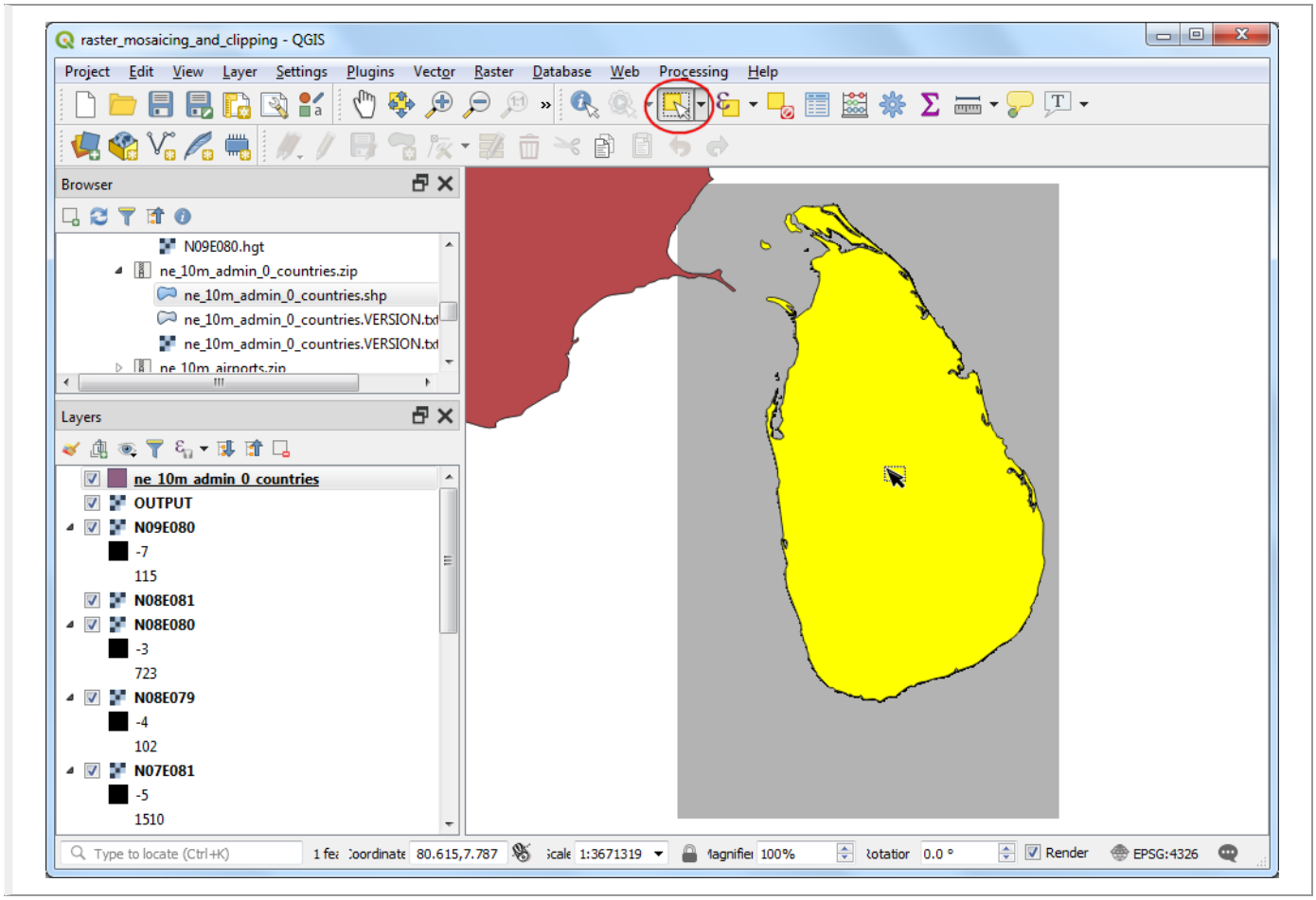
8. In the Layer Styling panel, click the Render type dropdown and select `Hillshade` renderer. This rendering option is particularly well-suited for elevation data.



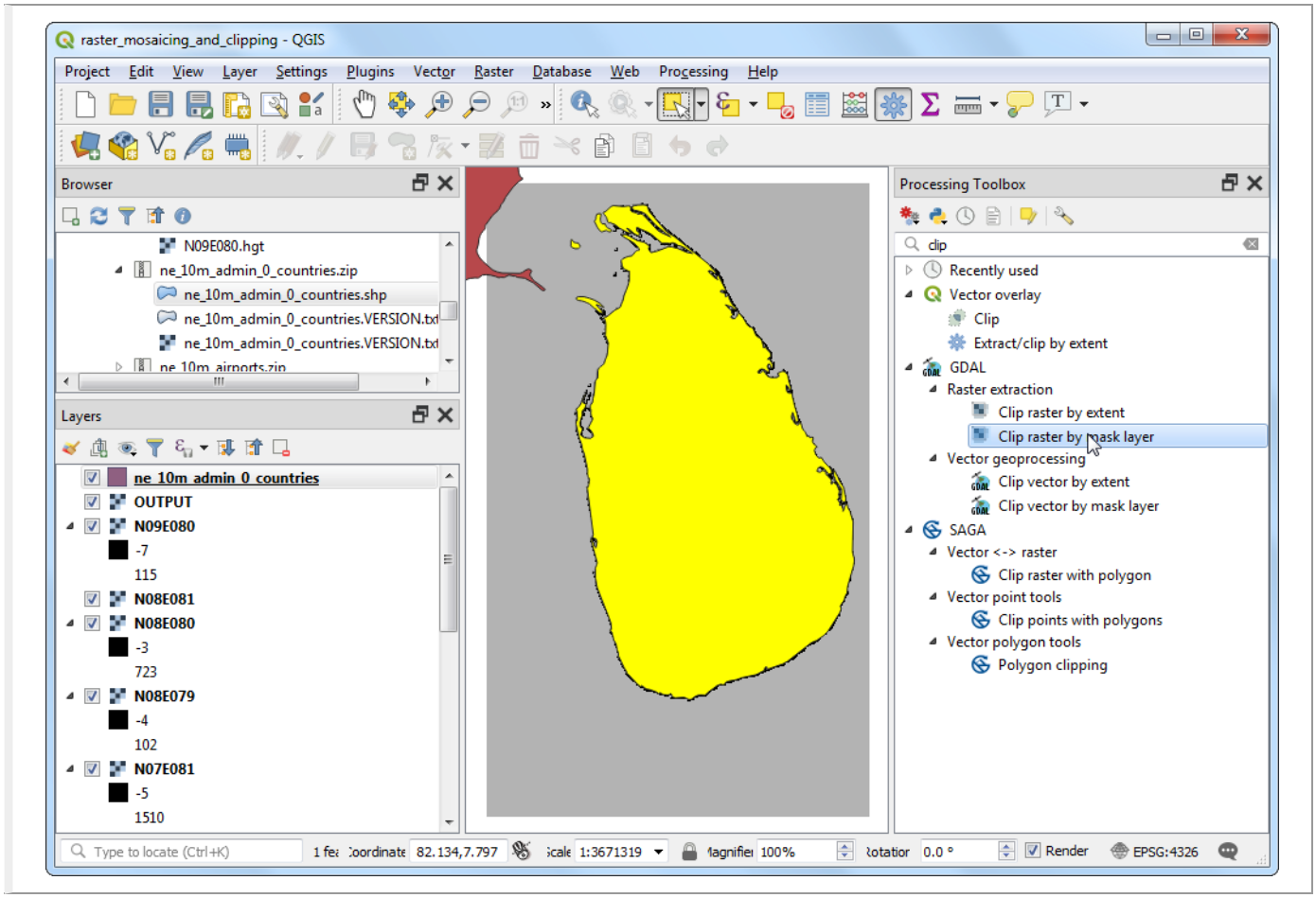
9. Another common operation when working with rasters is to clip a raster to your area of interest. For this tutorial, we will clip the merged layer to the country boundary for Sri Lanka. Locate the downloaded `ne_10m_admin_0_countries.zip` file and expand it. Drag the `ne_10m_admin_0_countries.shp` file to the canvas.



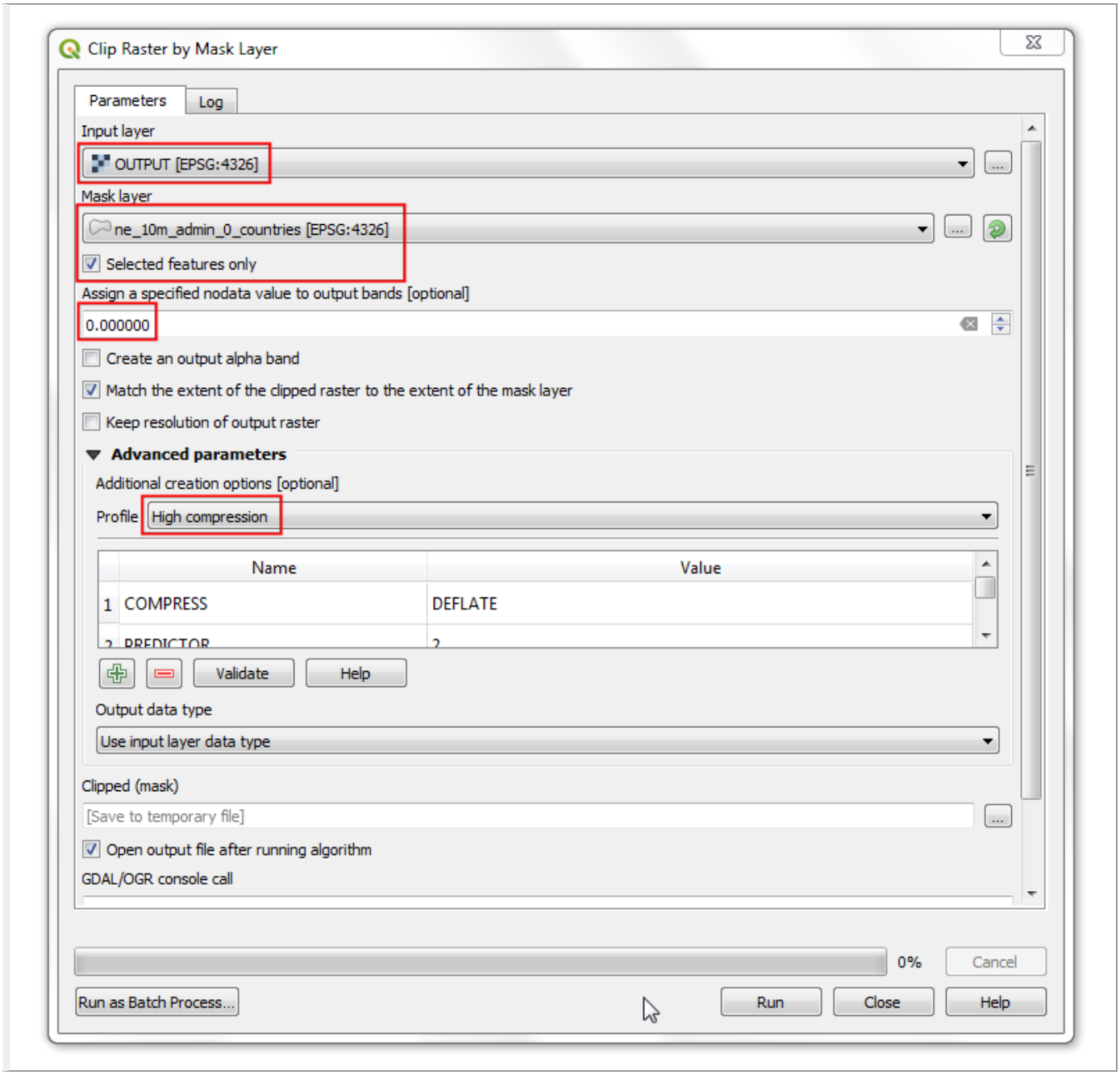
10. Select the newly added `ne_10m_admin_0_countries` layers in the Layers panel. Click the Select Features by area of single click button on the Attributes Toolbar. Once selected, click the polygon for Sri Lanka to select it.



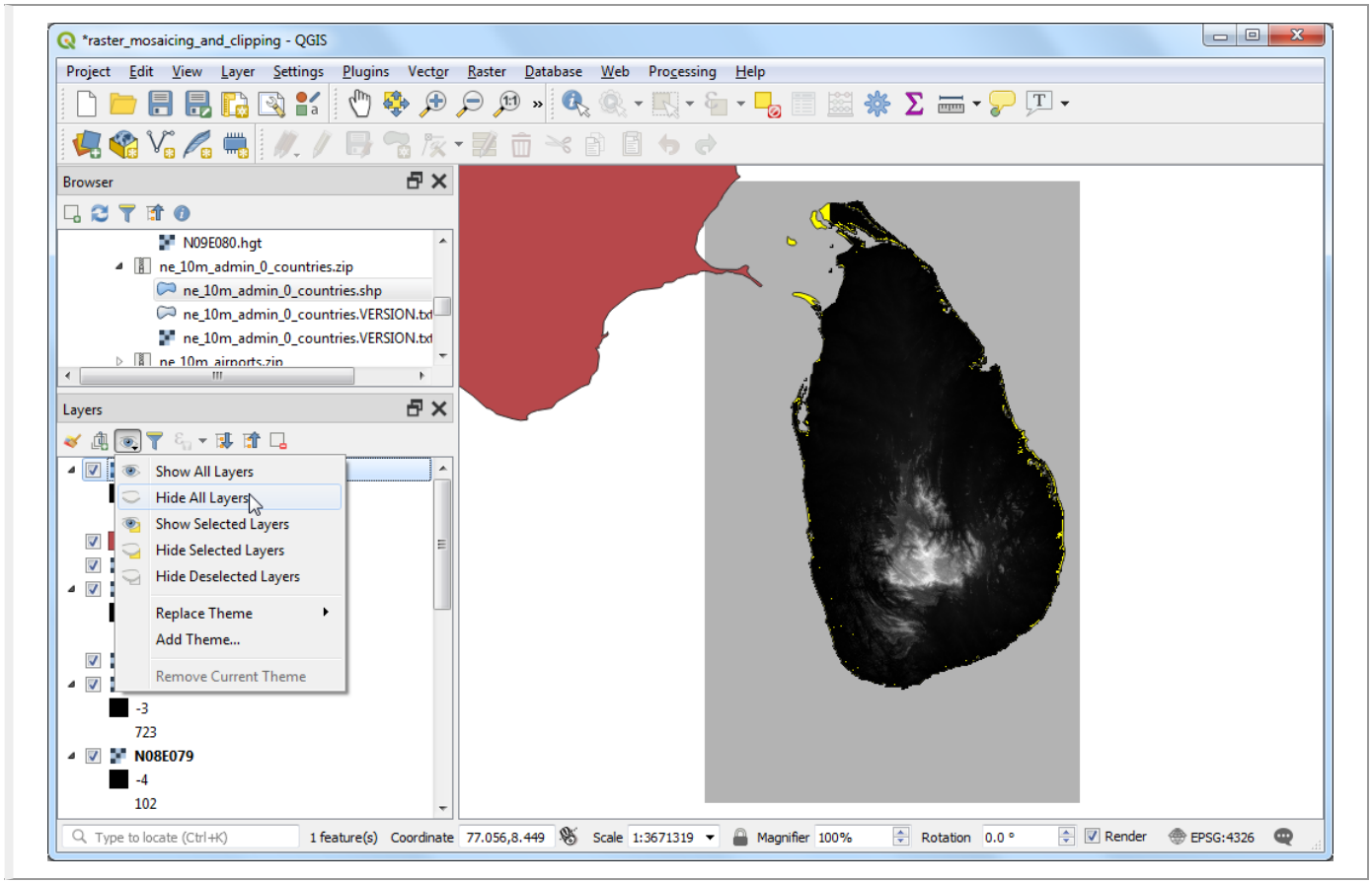
11. Keep the selection as it is and open Processing ▸ Toolbox. Search and locate the GDAL ▸ Raster extraction ▸ Clip raster by mask layer tool. Double-click to launch it.



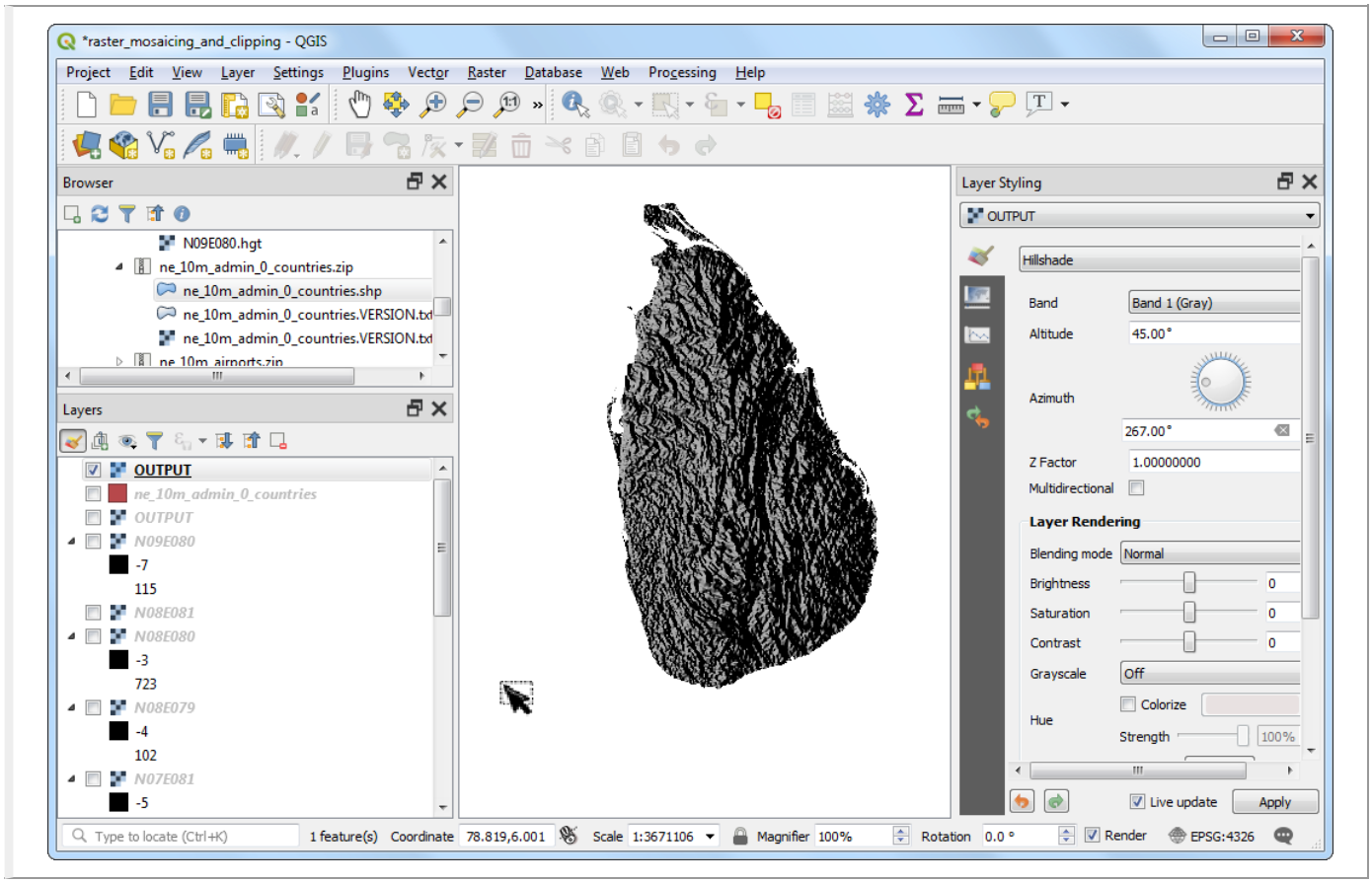
12. In the Clip Raster by Mask Layer dialog, set `OUTPUT` as the Input Layer. Select `ne_10m_admin_0_countries` as the Mask layer, and check the Selected features only checkbox. Enter `0.0000` as the Assign a specified nodata value to output bands. As before, choose `High compression` as the Profile. Click Run.



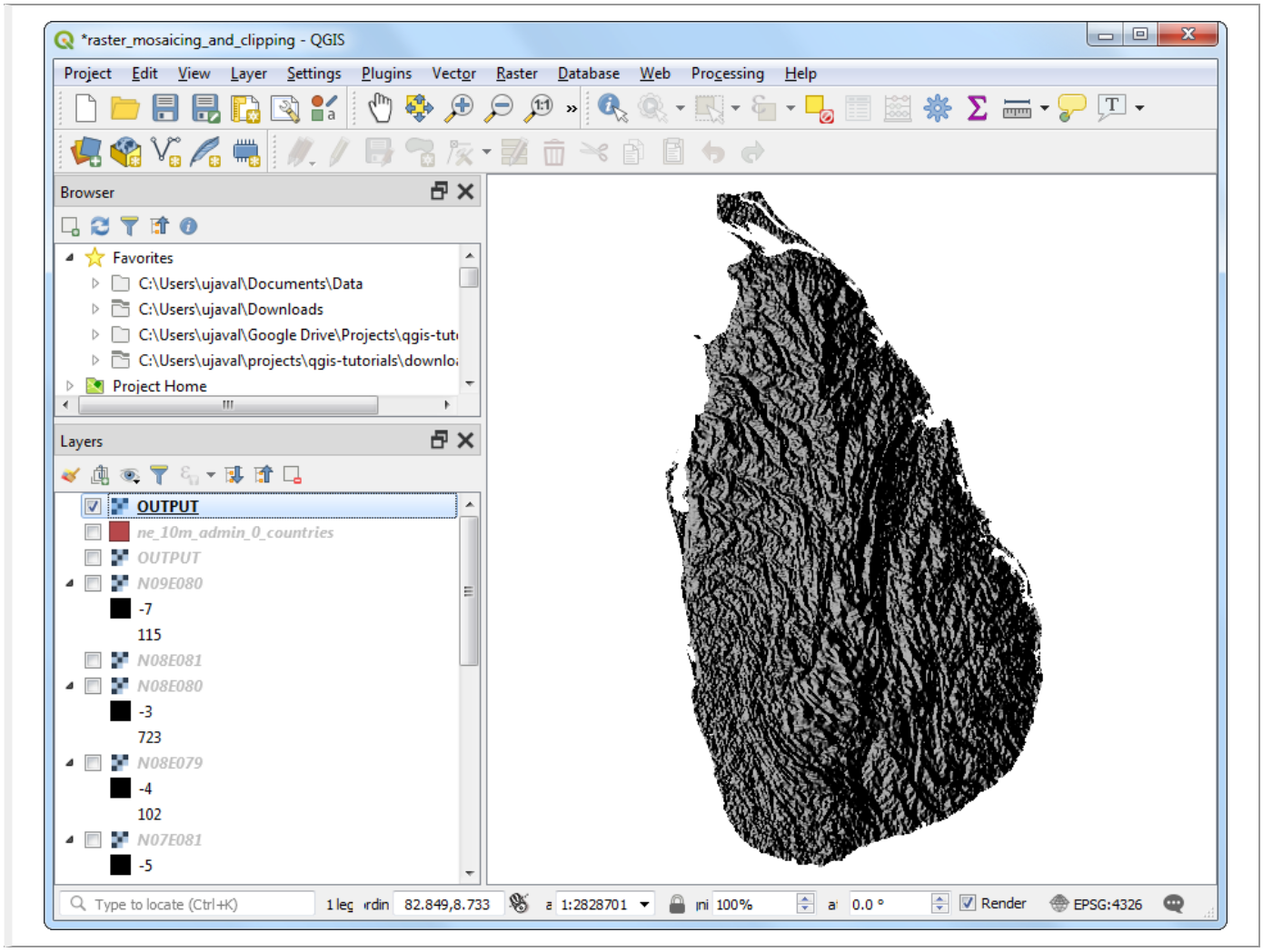
13. A new layer `OUTPUT` will be added to the Layers panel. At this point, it may be hard to see the output because we have too many overlapping layers visible. Click the Manage Map Themes button in the Layers panel and choose `Hide All Layers`.



14. Turn on only the latest `OUTPUT` layer and style it with the `Hilshade` renderer as done before.



15. The merged and subsetted output elevation layer for Sri Lanka is ready.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Working with Terrain Data

Terrain or elevation data is useful for many GIS Analysis and it is often used in maps. QGIS has good terrain processing capabilities built-in. In this tutorial, we will work through the steps to generate various products from elevation data such as contours, hillshade etc.

Overview of the task

The task is to create contours and hillshade map for area around Mt. Everest.

Other skills you will learn

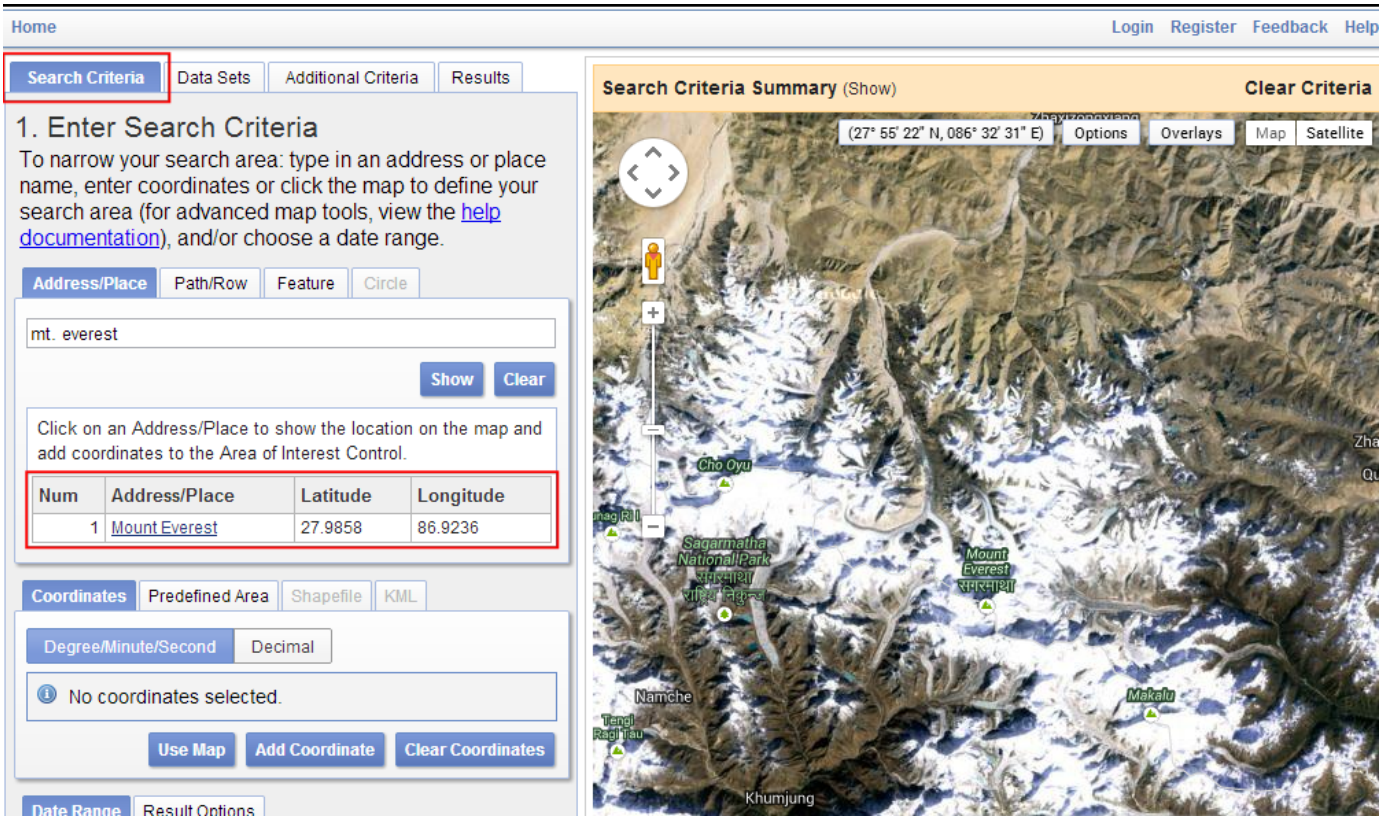
- Searching and downloading freely available terrain data.
- Exporting a vector layer as KML and viewing it in Google Earth.

Get the data

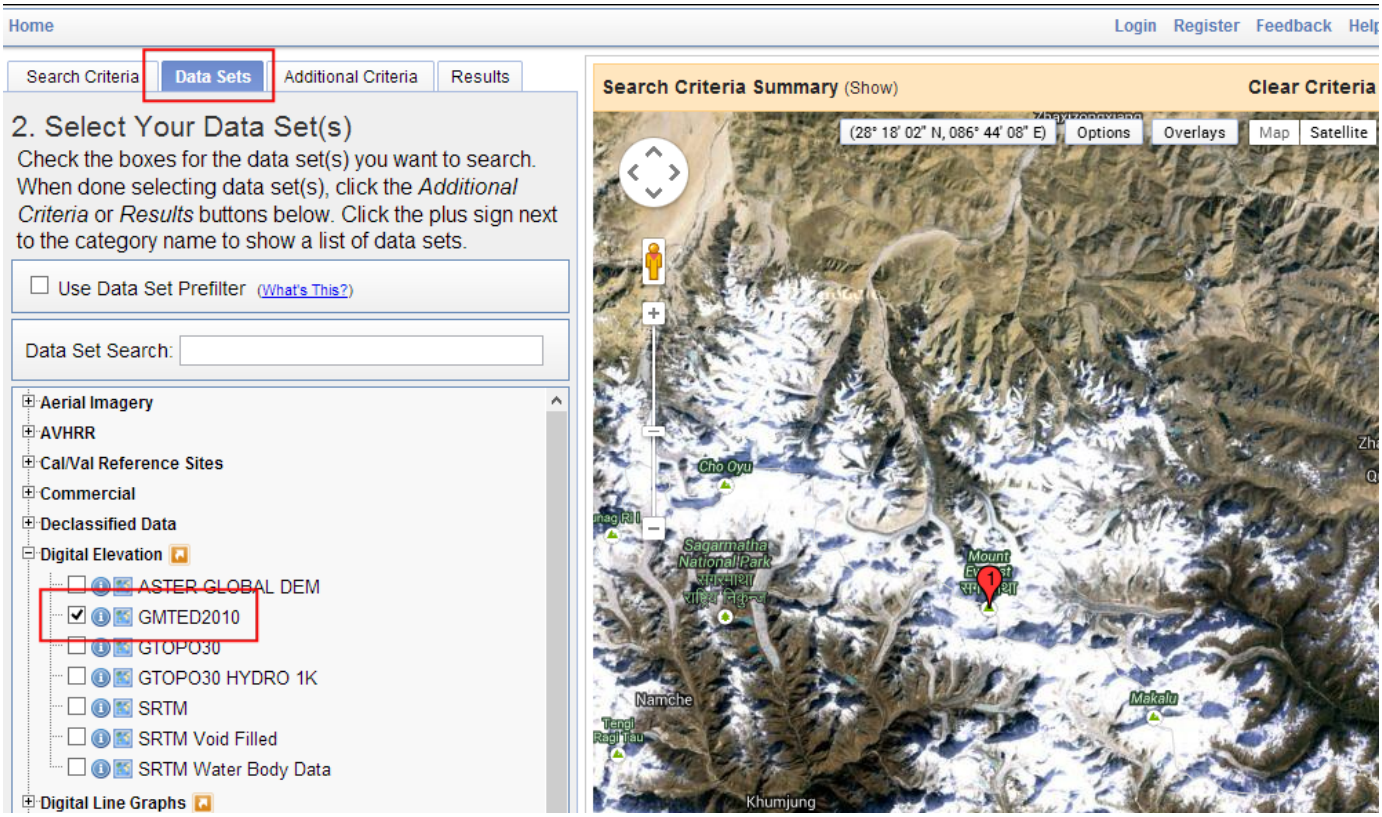
We will be working with GMTED2010 dataset from USGS. This data can be downloaded from the USGS Earthexplorer (http://earthexplorer.usgs.gov/) site. GMTED (Global Multi-resolution Terrain Elevation Data) (http://eros.usgs.gov/#/Find_Data/Products_and_Data_Available/GMTED2010) is a global terrain dataset that is the newer version of GTOPO30 dataset.

Here is how to search and download the revelant data from USGS Earthexplorer.

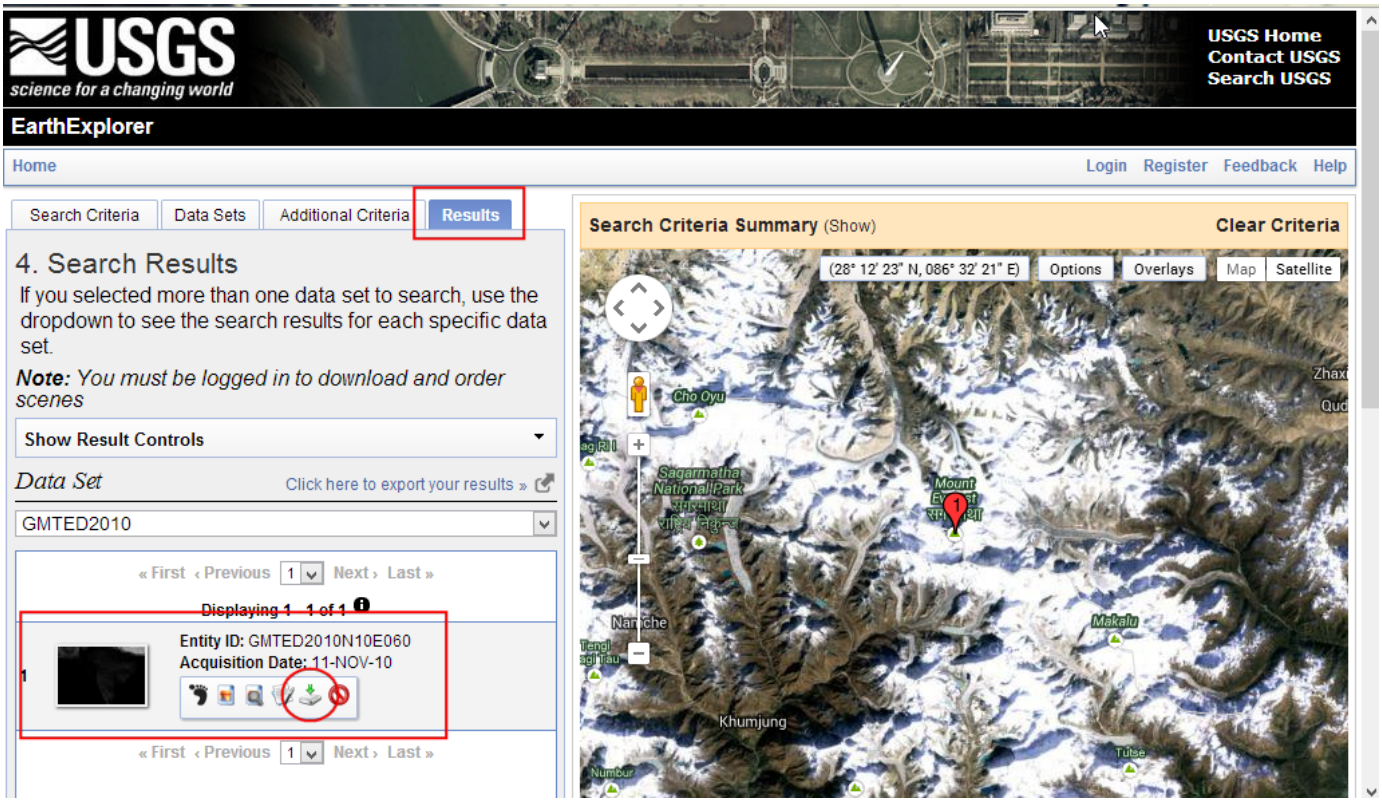
1. Go to the USGS Earthexplorer (http://earthexplorer.usgs.gov/) . In the Search Criteria tab, search for the place name Mt. Everest. Click on the result to select the location.



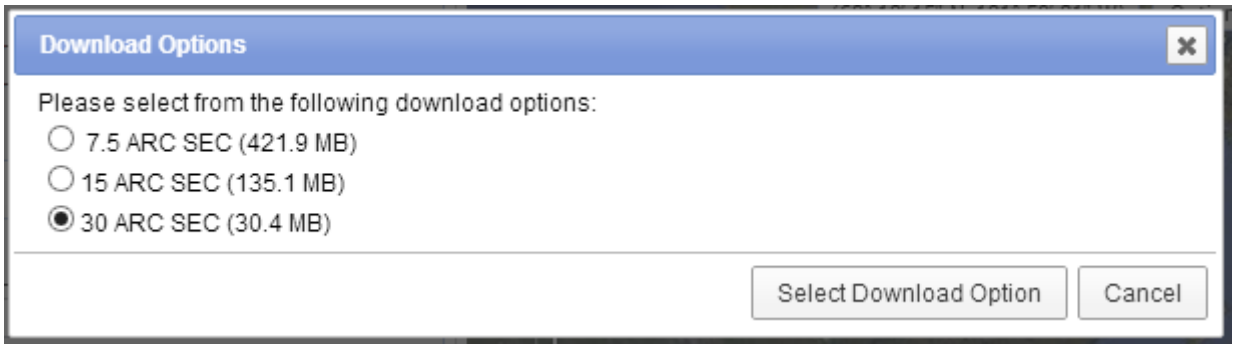
2. In the Data Sets tab, expand the Digital Elevation group, and check GMTED2010.



3. You can now skip to the Results tab and see the part of the dataset intersecting your search criteria. Click the Download Options button. You will have to log in to the site at this point. You can create a free account if you do not have one.



4. Select the 30 ARC SEC option and click Select Download Option.



You will now have a file named GMTED2010N10E060_300.zip. Elevation data is distributed in various raster formats such as ASC, BIL, GeoTiff etc. QGIS supports a wide variety of raster formats (http://www.gdal.org/formats_list.html) via the GDAL library. The GMTED data comes as GeoTiff files which are contained in this zip archive.

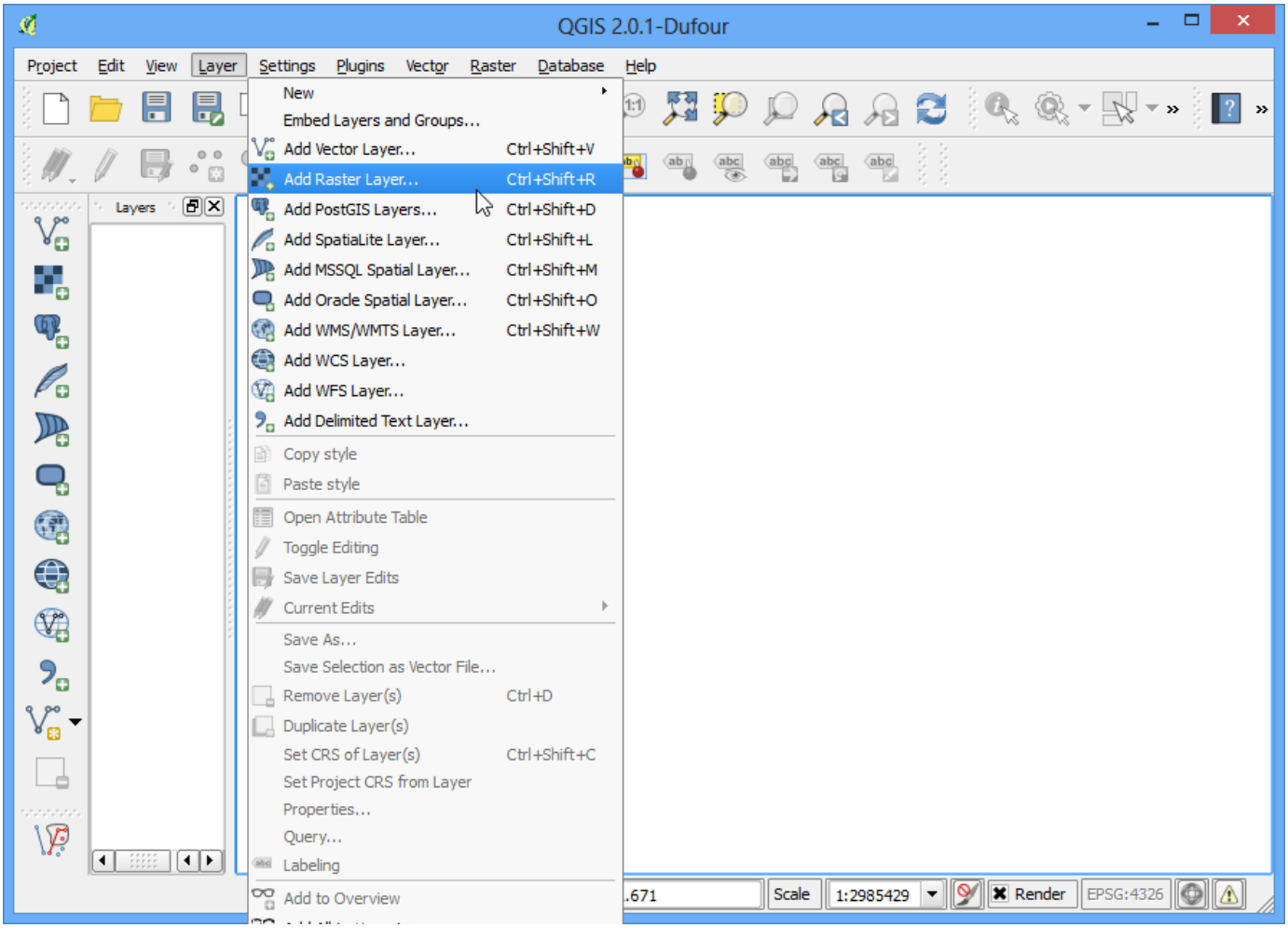
For convenience, you can download a copy of the data directly from below.

GMTED2010N10E060_300.zip (http://www.qgistutorials.com/downloads/GMTED2010N10E060_300.zip)

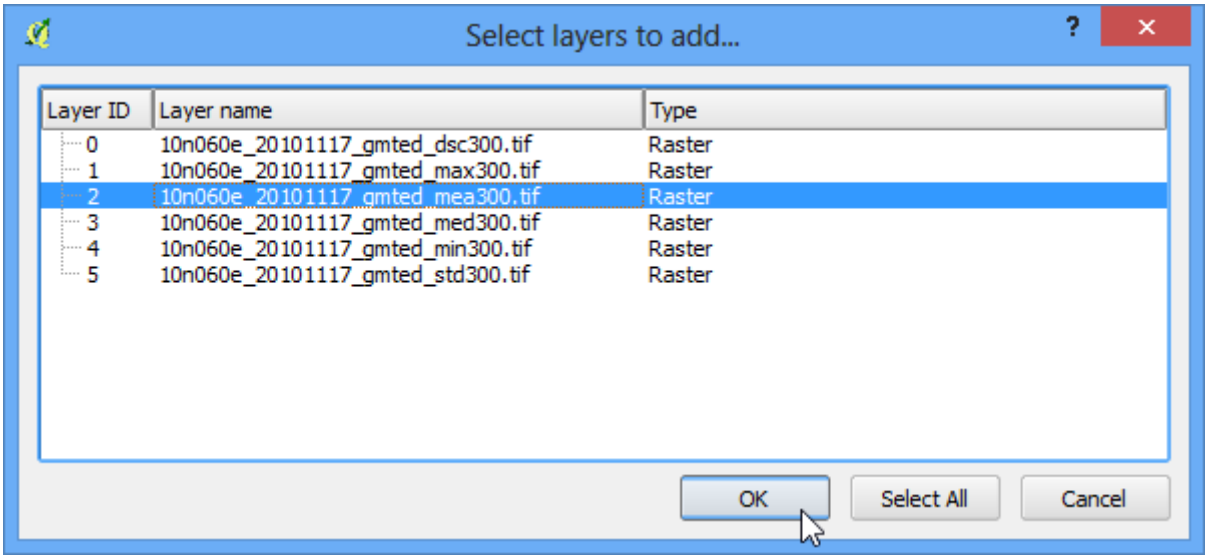
Data Source: [GMTED2010] (credits.html#gmted2010)

Procedure

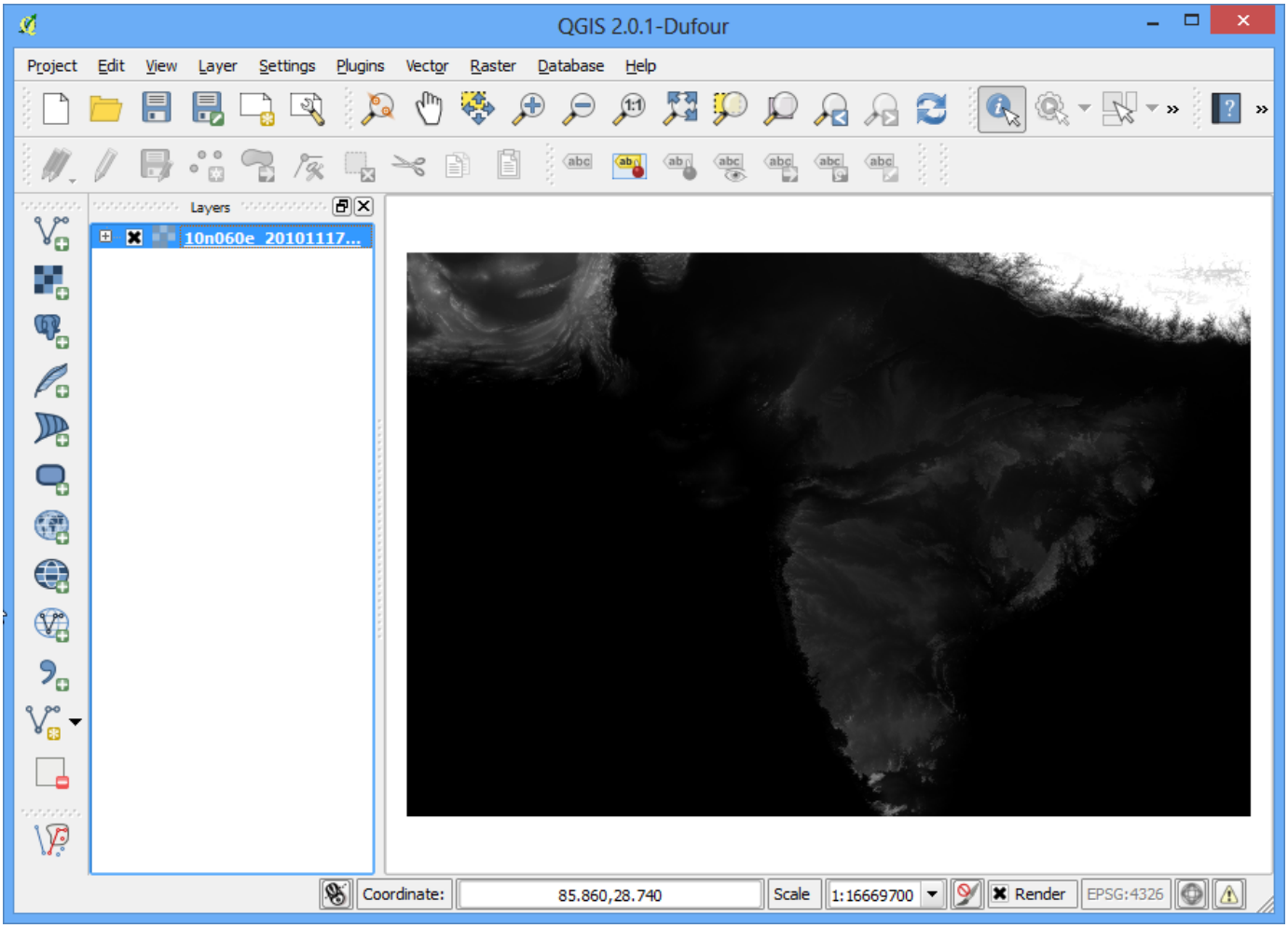
5. Open Layer ▸ Add Raster Layer and browse to the downloaded zip file.



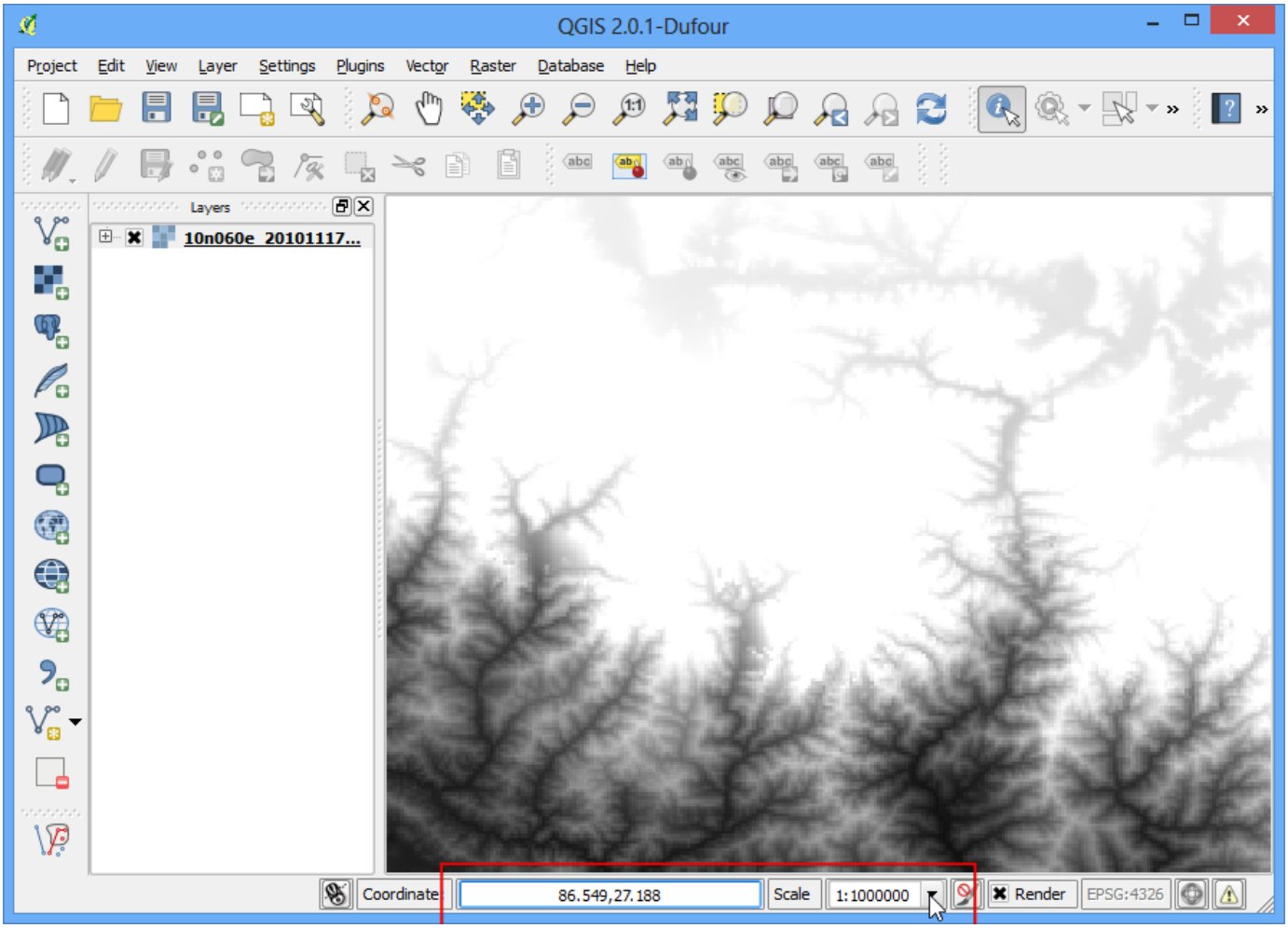
6. There are many different files generated from different algorithms. For this tutorial, we will use the file named 10n060e_20101117_gmted_mea300.tif.



7. You will see the terrain data rendered in the QGIS Canvas. Each pixel in the terrain raster represents the average elevation in meters at that location. The dark pixels represent areas with low altitude and lighter pixels represent areas with high altitude.



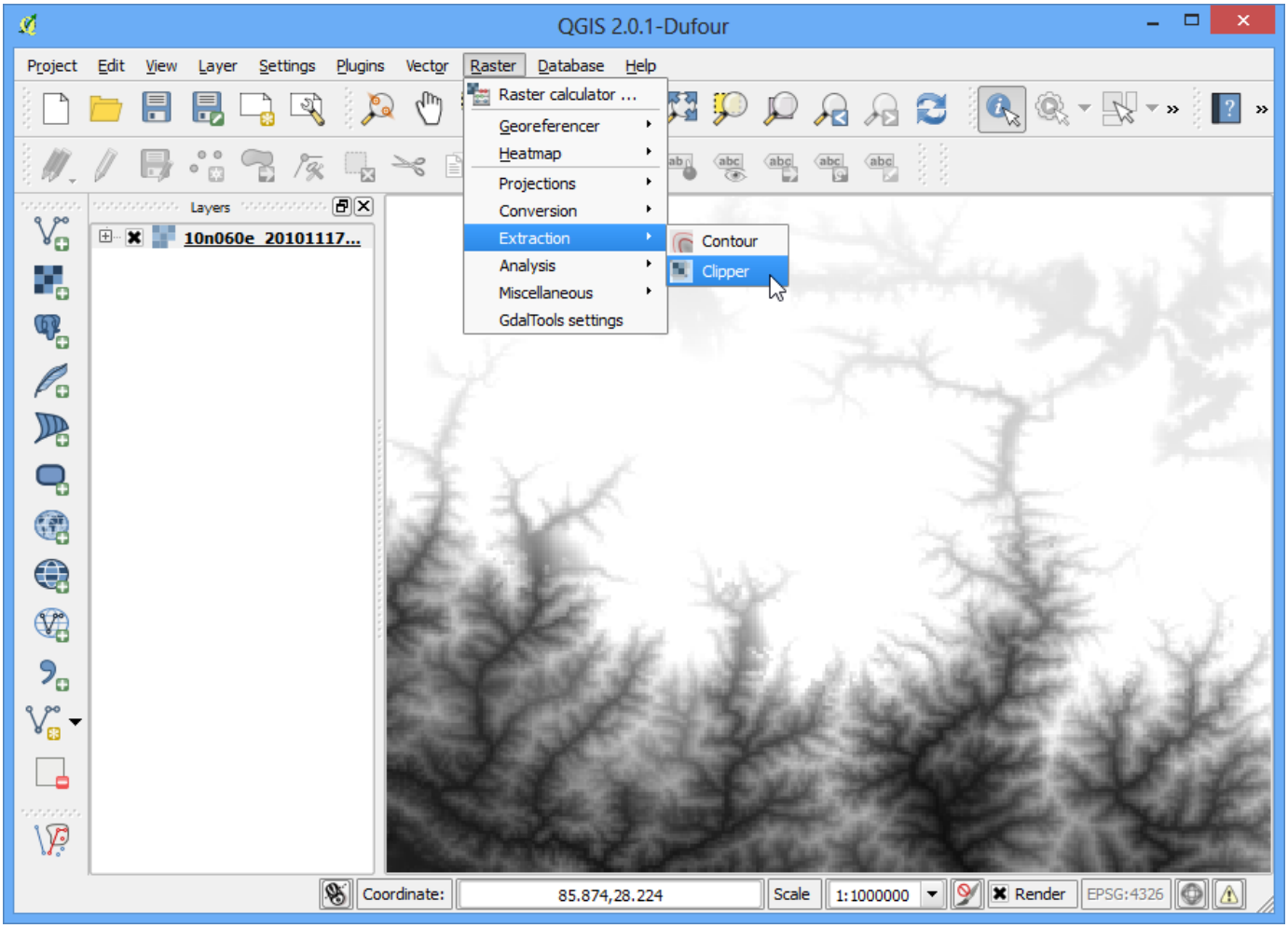
8. Let's find our area of interest. From Wikipedia (http://en.wikipedia.org/wiki/Mount_Everest), we find that the coordinates for our area of interest - Mt. Everest - is located at the coordinates 27.9881° N, 86.9253° E. Note that QGIS uses the coordinates in (X,Y) format , so you must use the coordinates as (Longitude, Latitude). Paste 86.9253,27.9881 these at the bottom of QGIS window where it says Coordinate and press Enter. The viewport will be centered at this coordinate. To zoom in, Enter 1:1000000 in the Scale field and press Enter. You will see the viewport zoom to the area around the Himalayas.



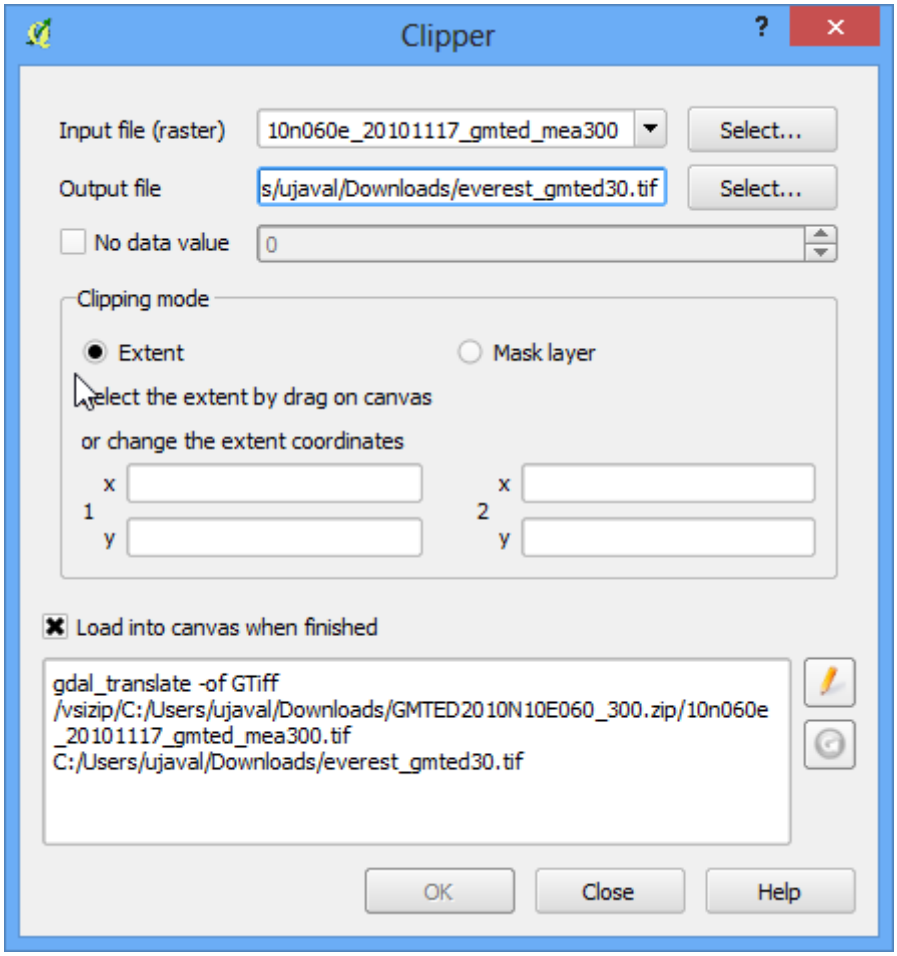
9. We will now crop the raster to this area of interest. Select the Clipper tool from Raster ‣ Extraction ‣ Clipper.

Note

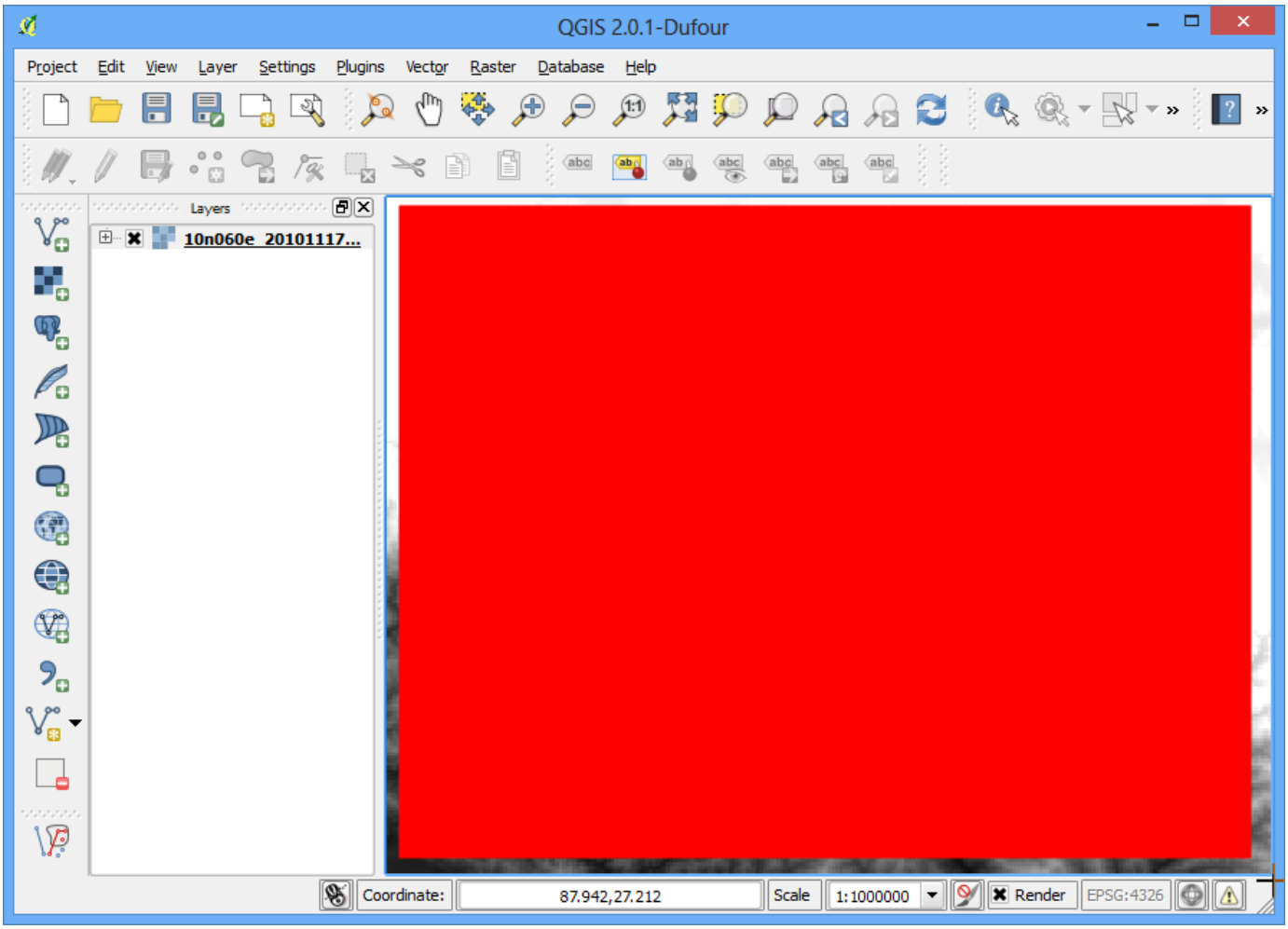
The Raster menu in QGIS comes from a core plugin called GdalTools. If you do not see the Raster menu, enable the GdalTools plugin from Plugins ‣ Manage and install plugins ‣ Installed. See *Using Plugins* (using_plugins.html) for more details.



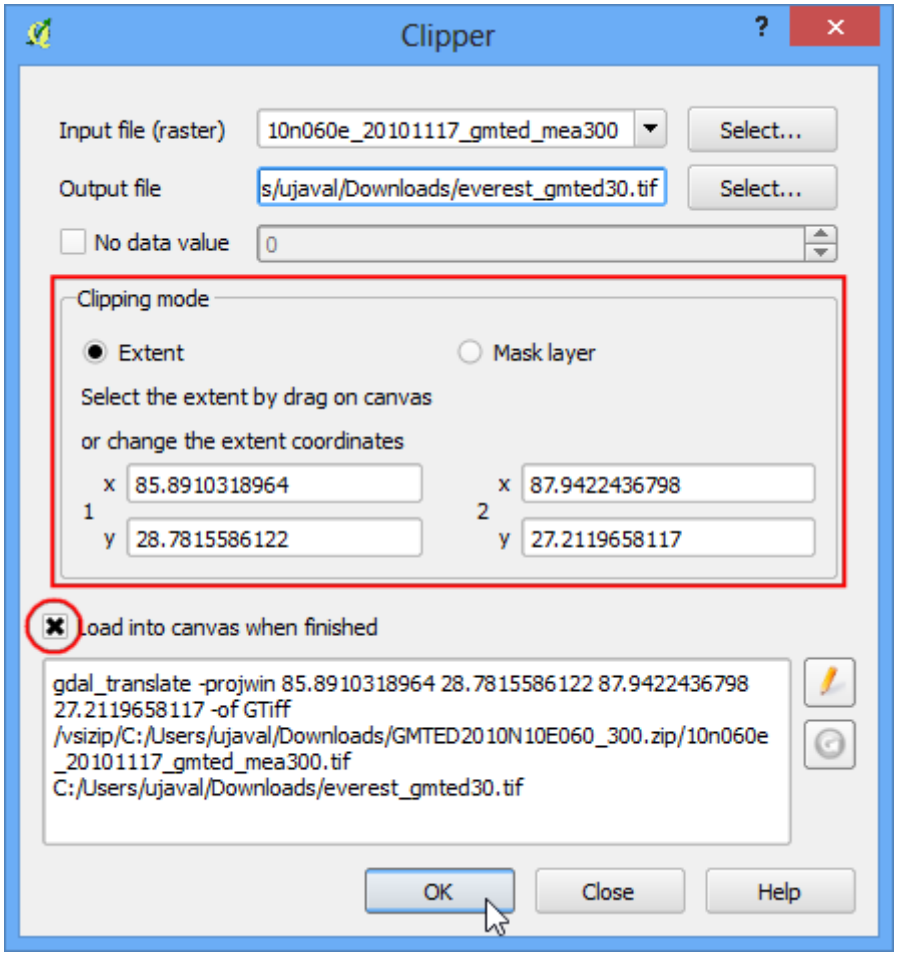
10. In the Clipper window, name your output file as everest_gmted30.tif. Select the Clipping mode as Extent.



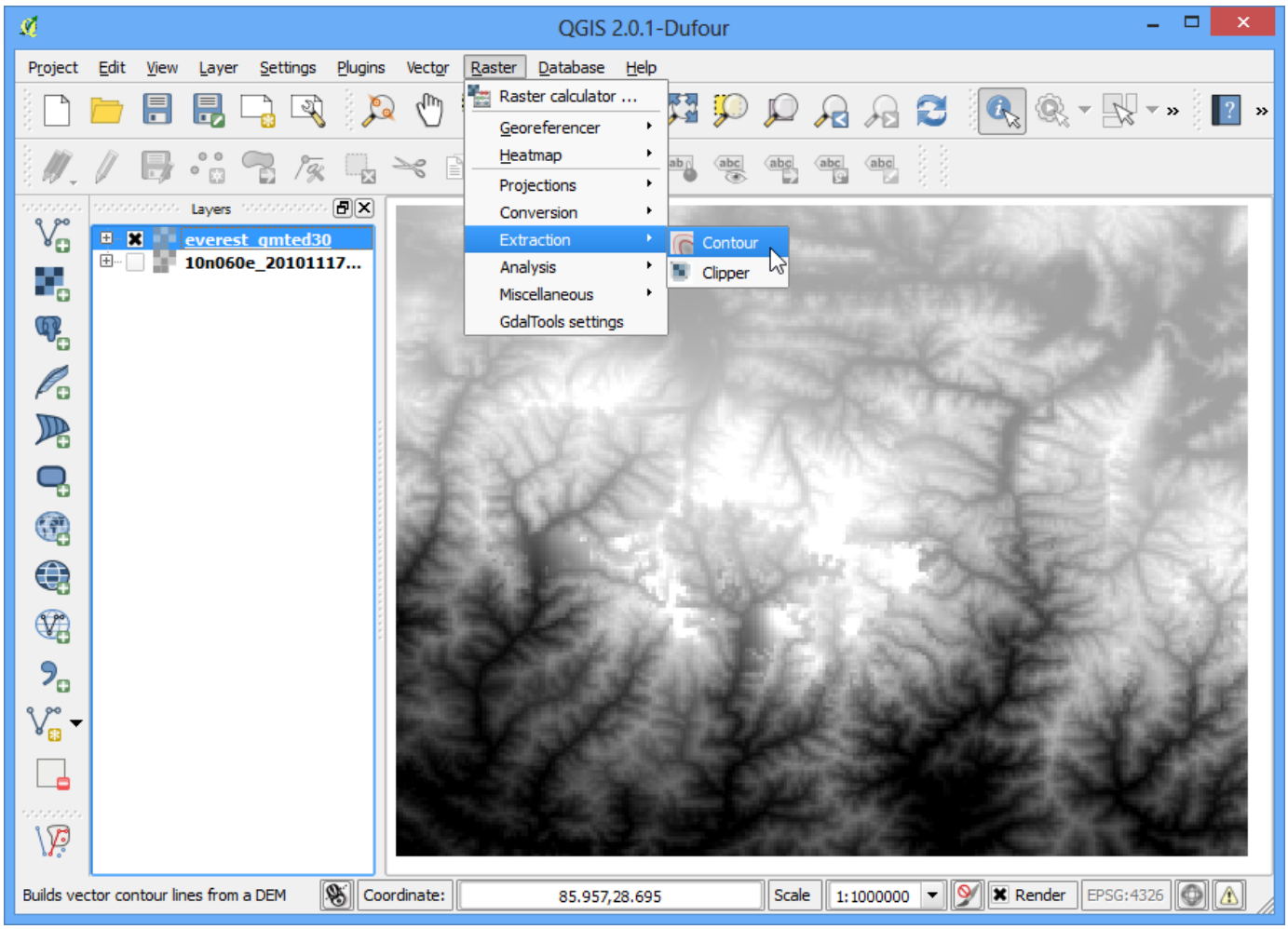
11. Keep the Clipper window open and switch to the main QGIS window. Hold your left mouse button and draw a rectangle covering the full canvas.



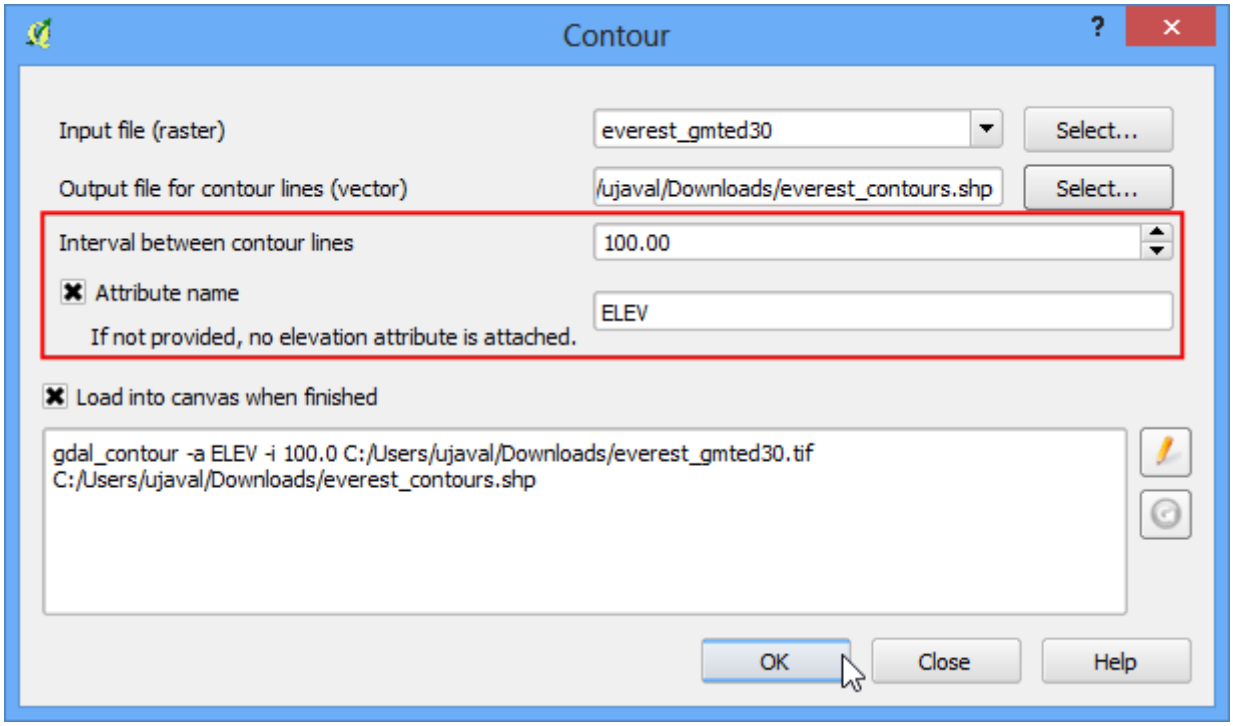
12. Now back in the Clipper window, you will see the coordinates auto-populated from your selection. Make sure the Load into canvas when finished option is checked, and click OK.



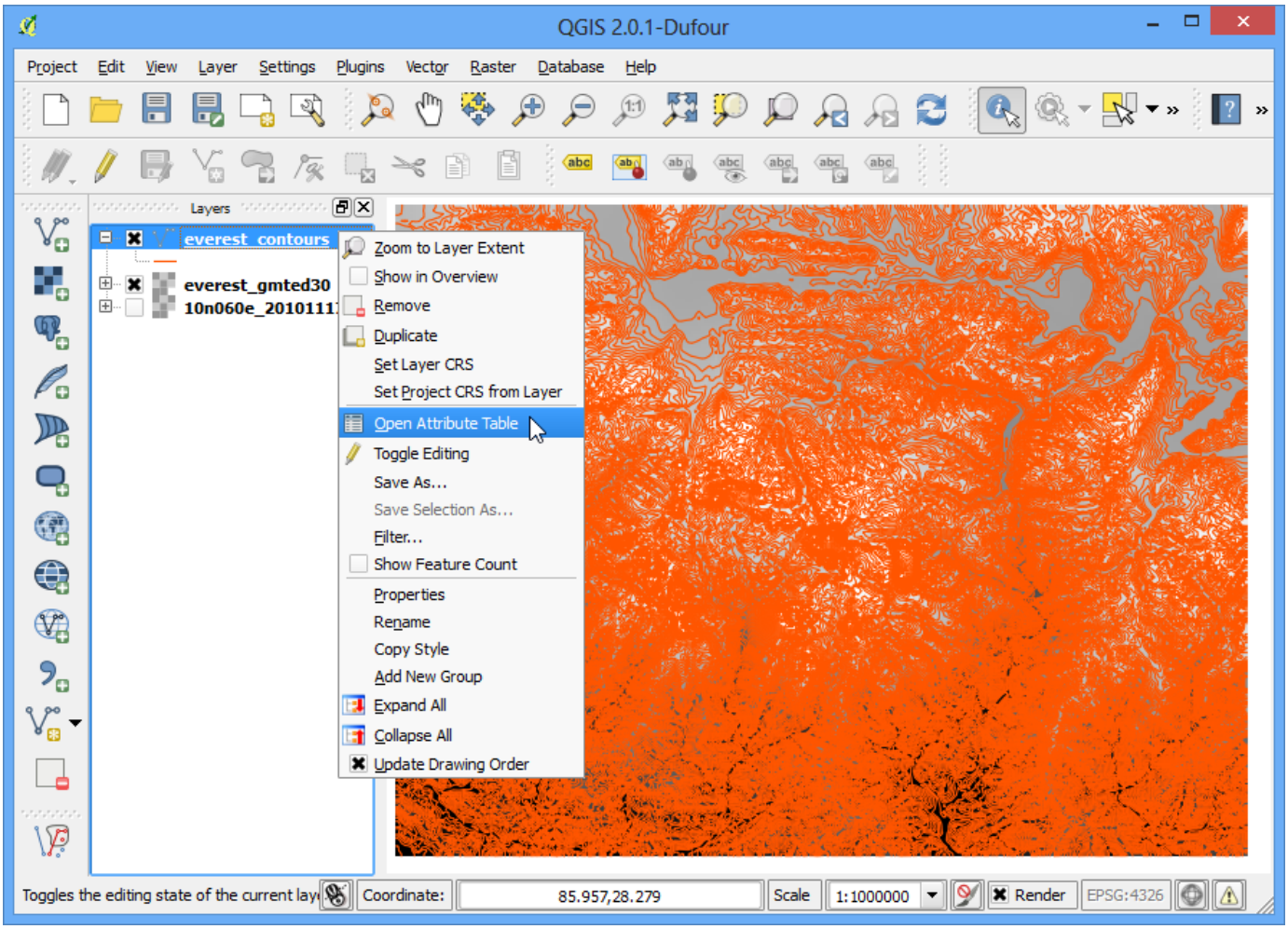
13. Once the process finishes, you will see a new layer loaded in QGIS. This layer covers only the area around Mt. Everest.Now we are ready to generate contours. Select the contour tool from Raster ‣ Extraction ‣ Contour.



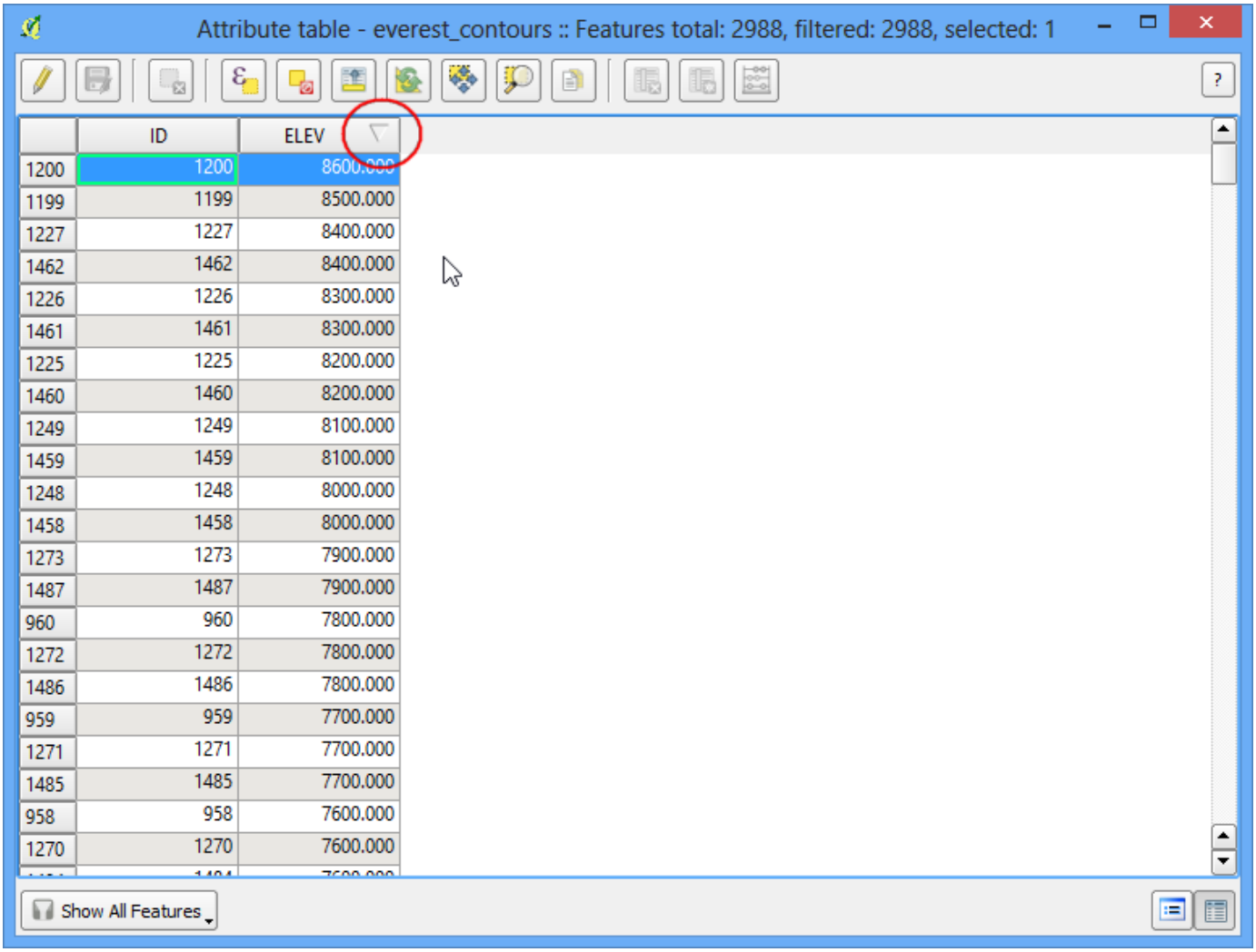
14. In the Contour dialog, select everest_gmted30 as the Input file. Name the Output file for contour lines as everest_countours.shp. We will generate contour lines for 100m intervals, so put 100 as the Interval between contour lines. Also check the Attribute name option so elevation value will be recorded as attribute of each contour line. Click OK.



15. Once the processing is complete, you will see contour lines loaded into the canvas. Each line in this layer represents a particular elevation. All points along a countour line in the underlying raster would be at the same elevation. The closer the lines, the steeper the slope. Let's inspect the contours a bit more. Right click on the contours layer and choose Open Attribute Table.

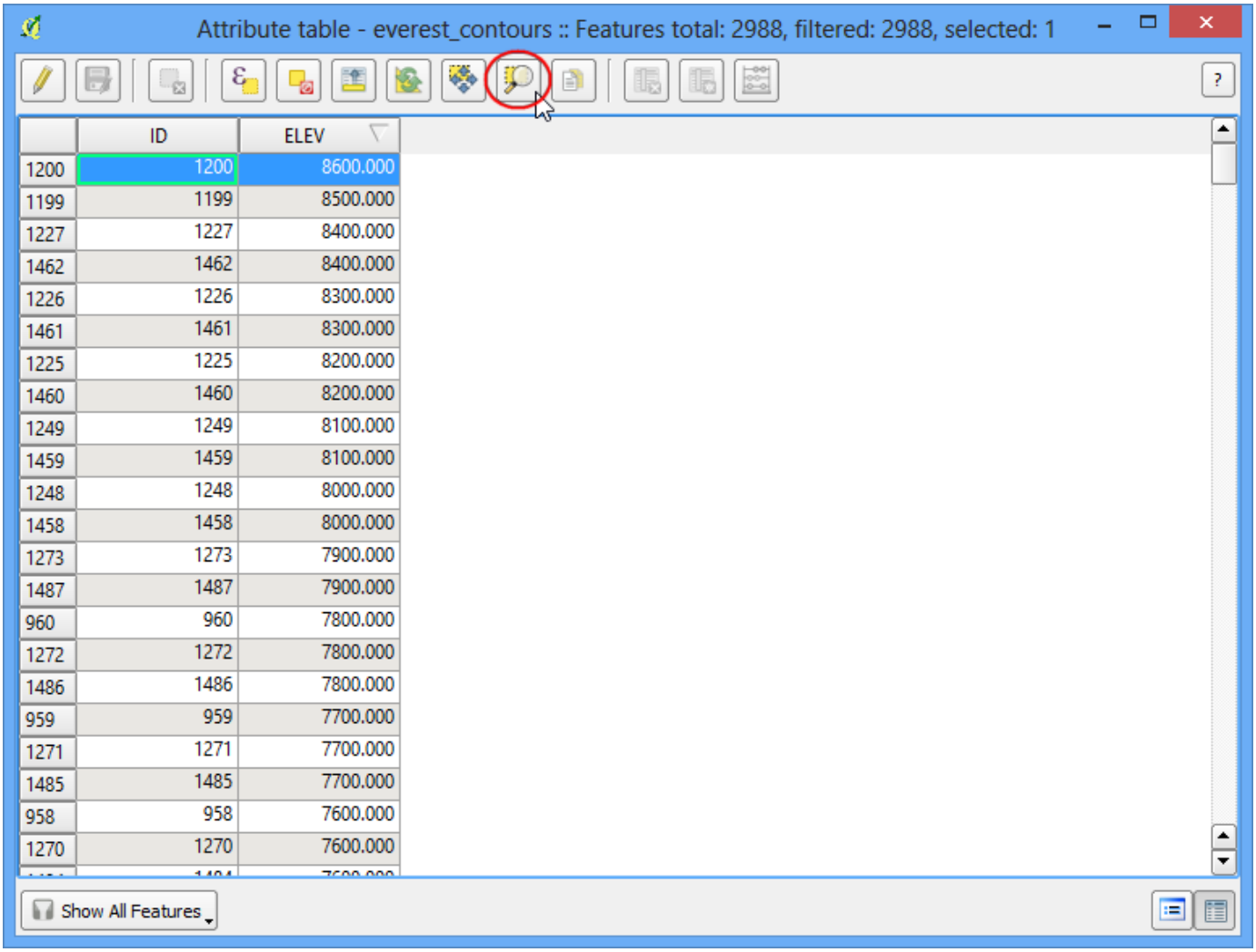


16. You will see that each line feature has an attribute named ELEV. This is the height in metres that each line represents. Click on the column header a couple of times to sort the values in descending order. Here you will find the line representing the highest elevation in our data, i.e. Mt. Everest.



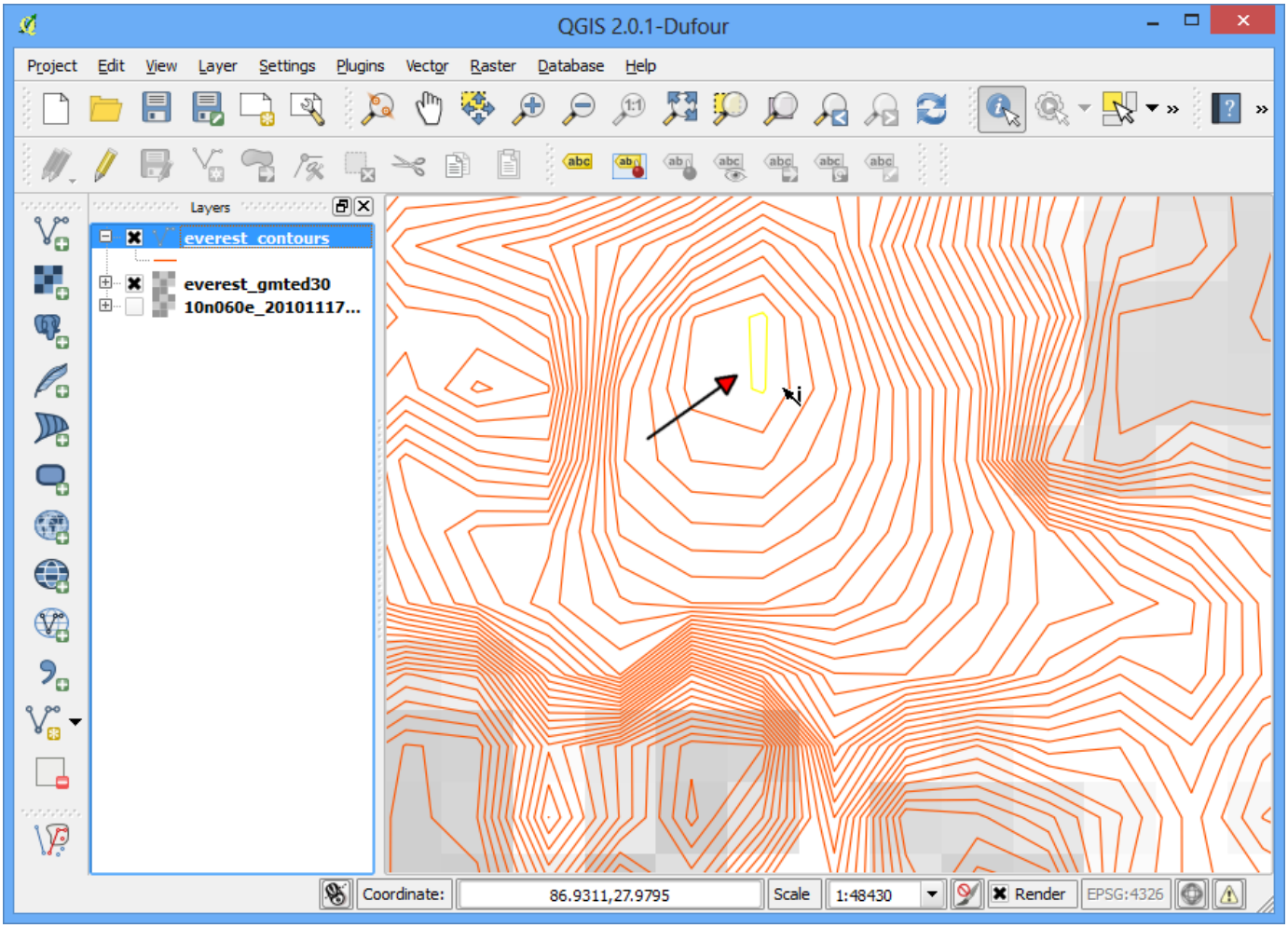
	ID	ELEV
1200	1200	8600.000
1199	1199	8500.000
1227	1227	8400.000
1462	1462	8400.000
1226	1226	8300.000
1461	1461	8300.000
1225	1225	8200.000
1460	1460	8200.000
1249	1249	8100.000
1459	1459	8100.000
1248	1248	8000.000
1458	1458	8000.000
1273	1273	7900.000
1487	1487	7900.000
960	960	7800.000
1272	1272	7800.000
1486	1486	7800.000
959	959	7700.000
1271	1271	7700.000
1485	1485	7700.000
958	958	7600.000
1270	1270	7600.000

17. Select the top row, and click on the Zoom to selection button.

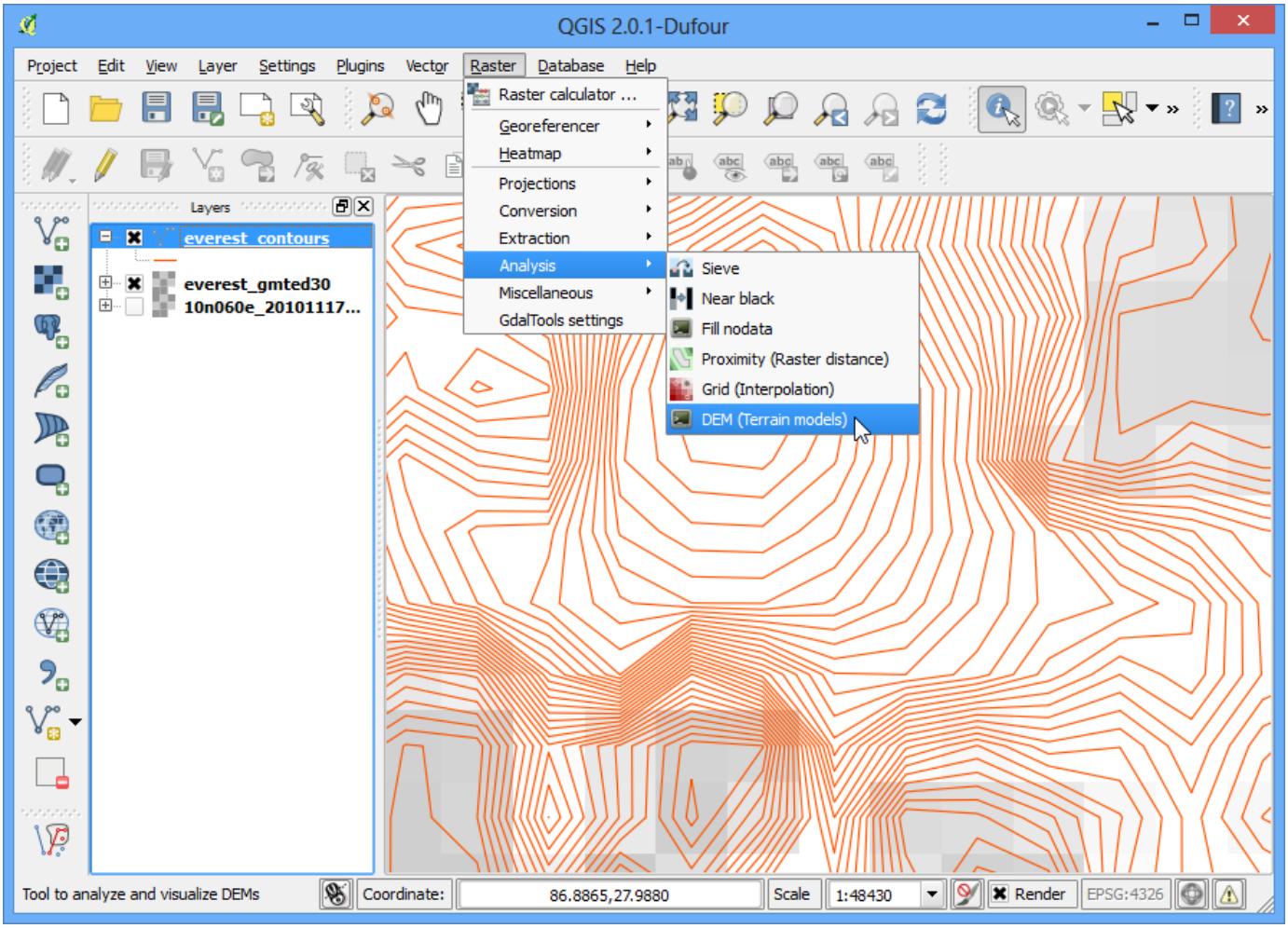


	ID	ELEV
1200	1200	8600.000
1199	1199	8500.000
1227	1227	8400.000
1462	1462	8400.000
1226	1226	8300.000
1461	1461	8300.000
1225	1225	8200.000
1460	1460	8200.000
1249	1249	8100.000
1459	1459	8100.000
1248	1248	8000.000
1458	1458	8000.000
1273	1273	7900.000
1487	1487	7900.000
960	960	7800.000
1272	1272	7800.000
1486	1486	7800.000
959	959	7700.000
1271	1271	7700.000
1485	1485	7700.000
958	958	7600.000
1270	1270	7600.000

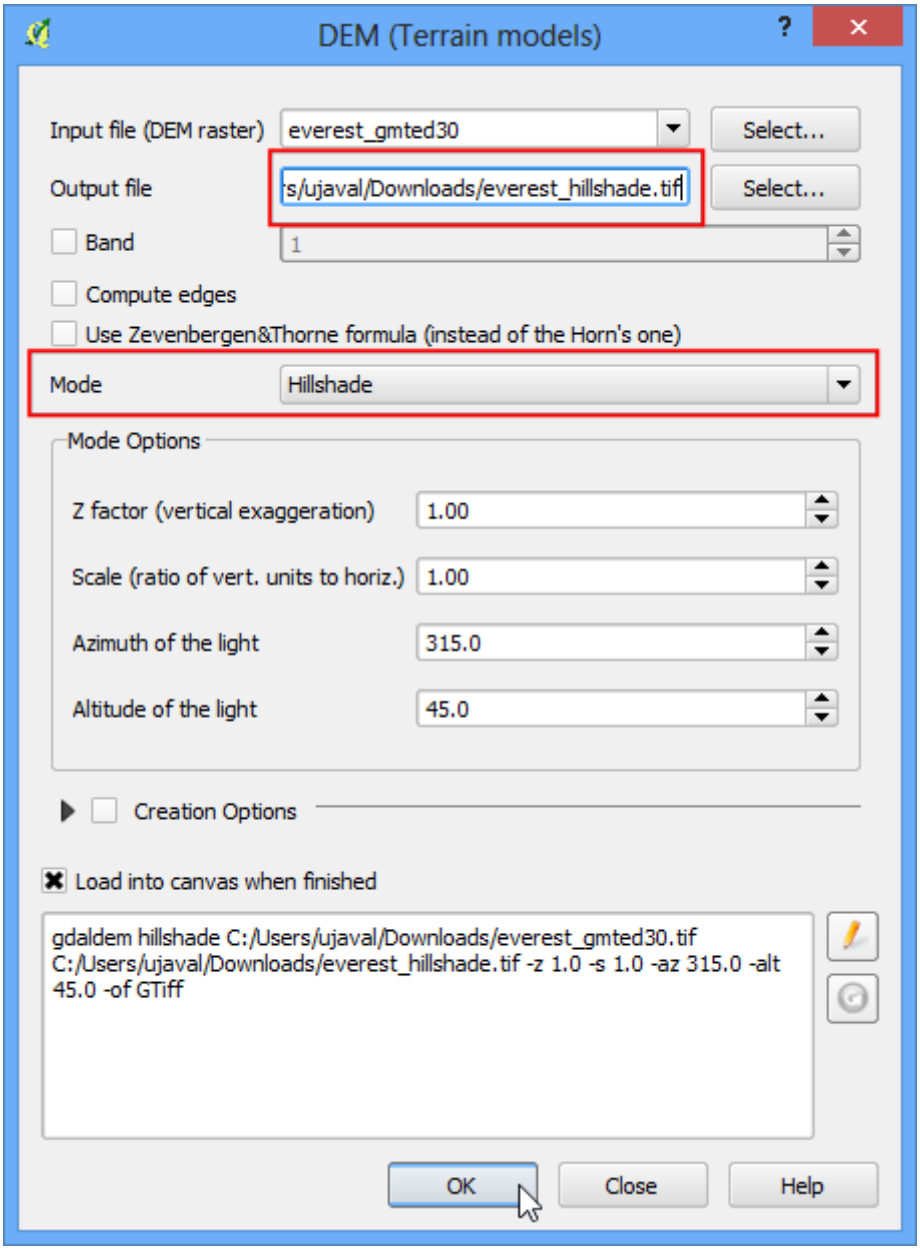
18. Switch to the main QGIS window. You will see the selected contour line highlighted in yellow. This is the area of the highest elevation in our dataset.



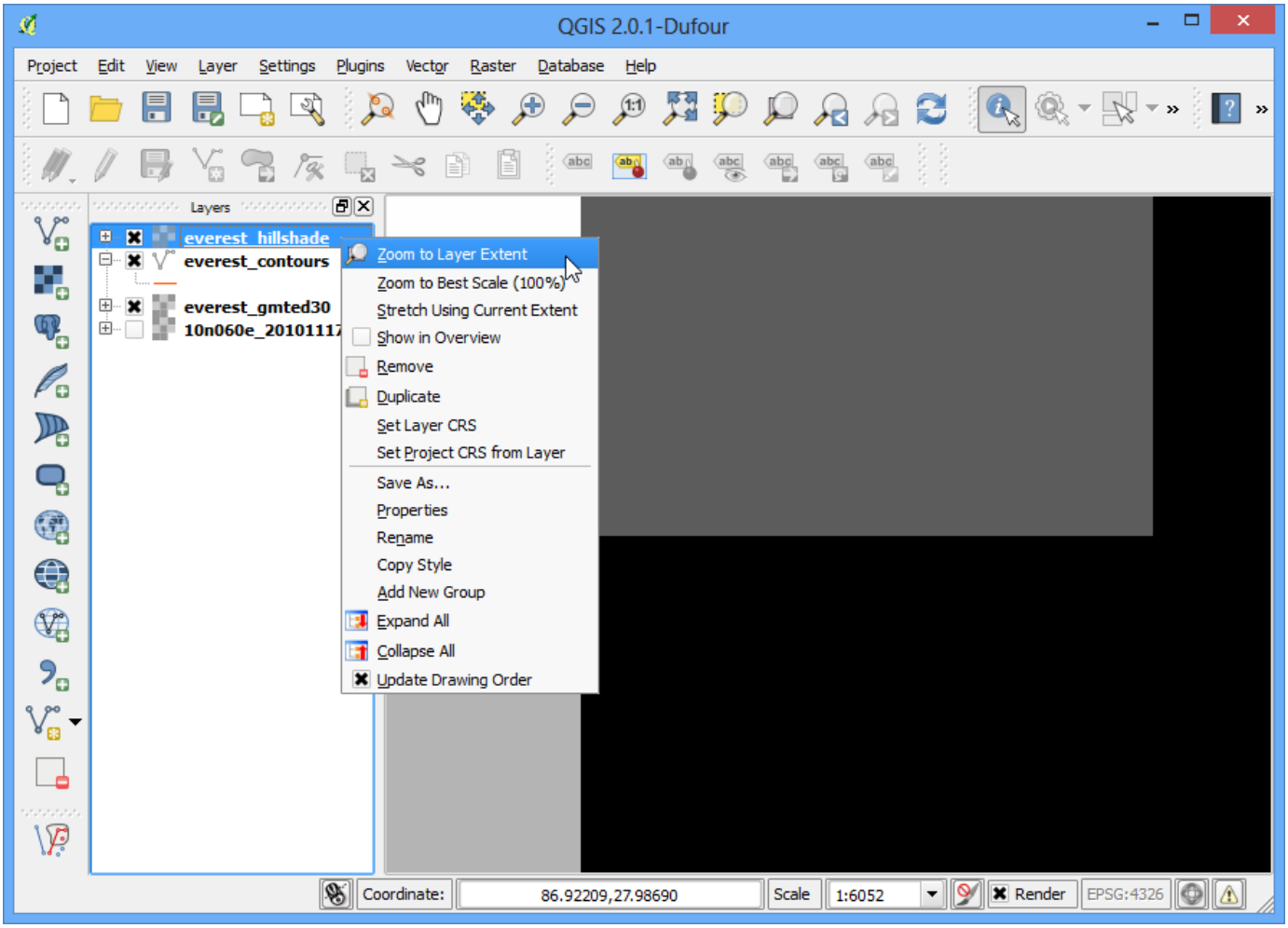
19. Now let us create a hillshade map from the raster. Select Raster ▸ Analysis ▸ DEM (Terrain Models).



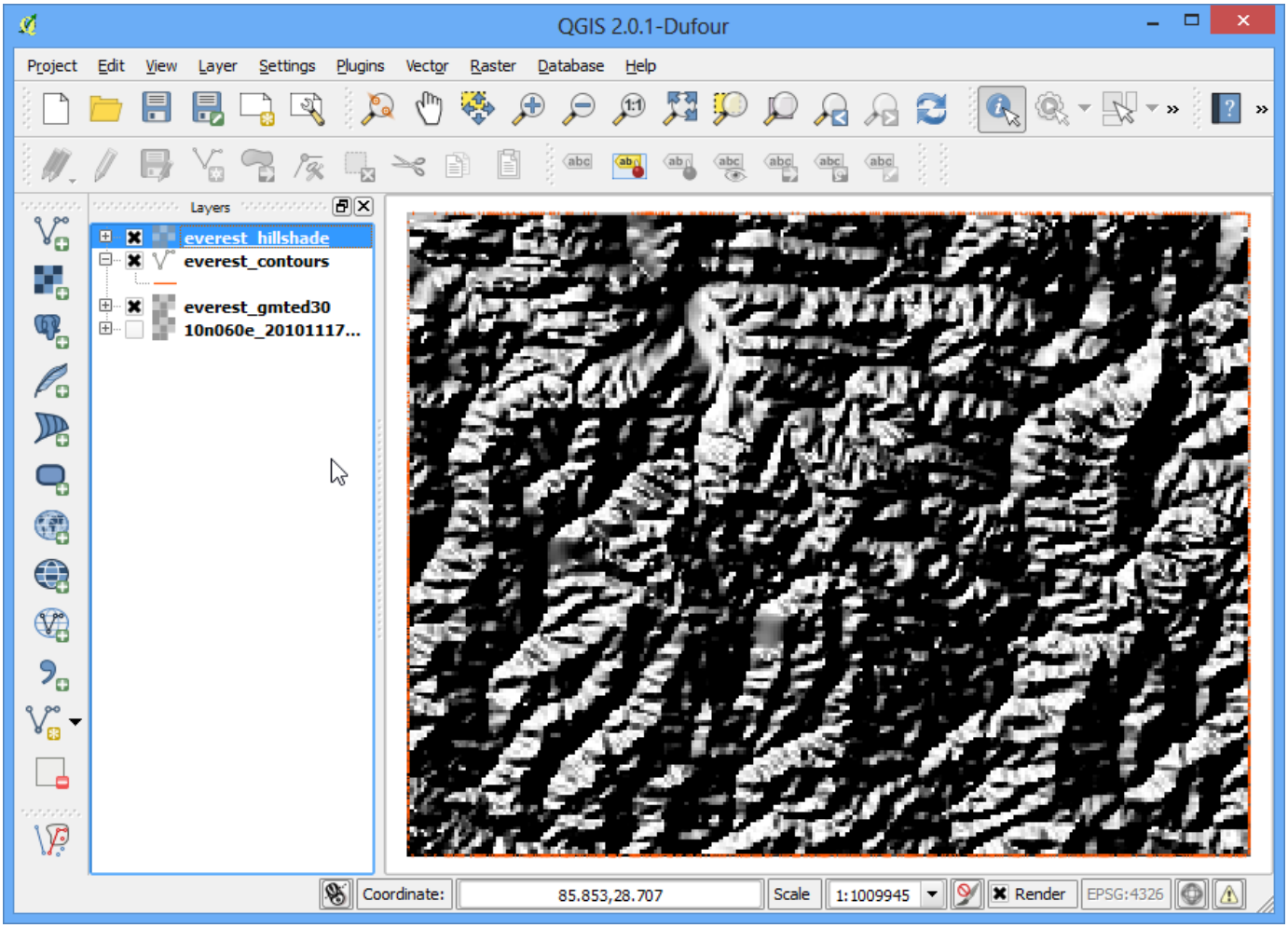
20. In the DEM (Terrain Models) dialog, choose everest_gmted30 as the Input file. Name the Output file as everest_hillshade.tif. Choose Hillshade as the Mode. Leave all other options as is. Make sure the Load into canvas when finished option is checked, and click OK.



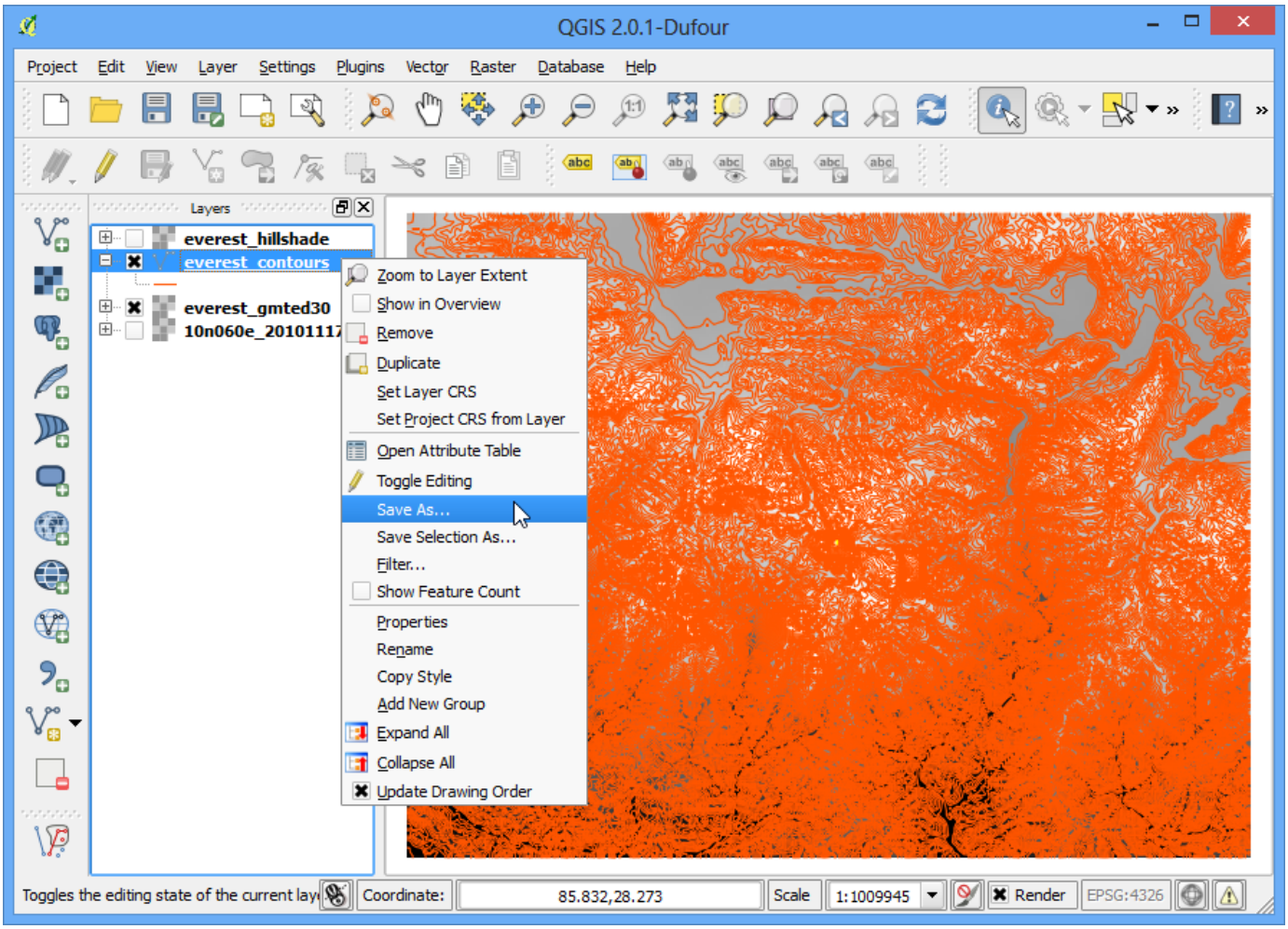
21. Once the process finishes, you will see yet another raster loaded into QGIS canvas. Since you maybe zoomed-in near the Mt.Everest region, right click on the everest_hillshade layer and choose Zoom to Layer Extent.



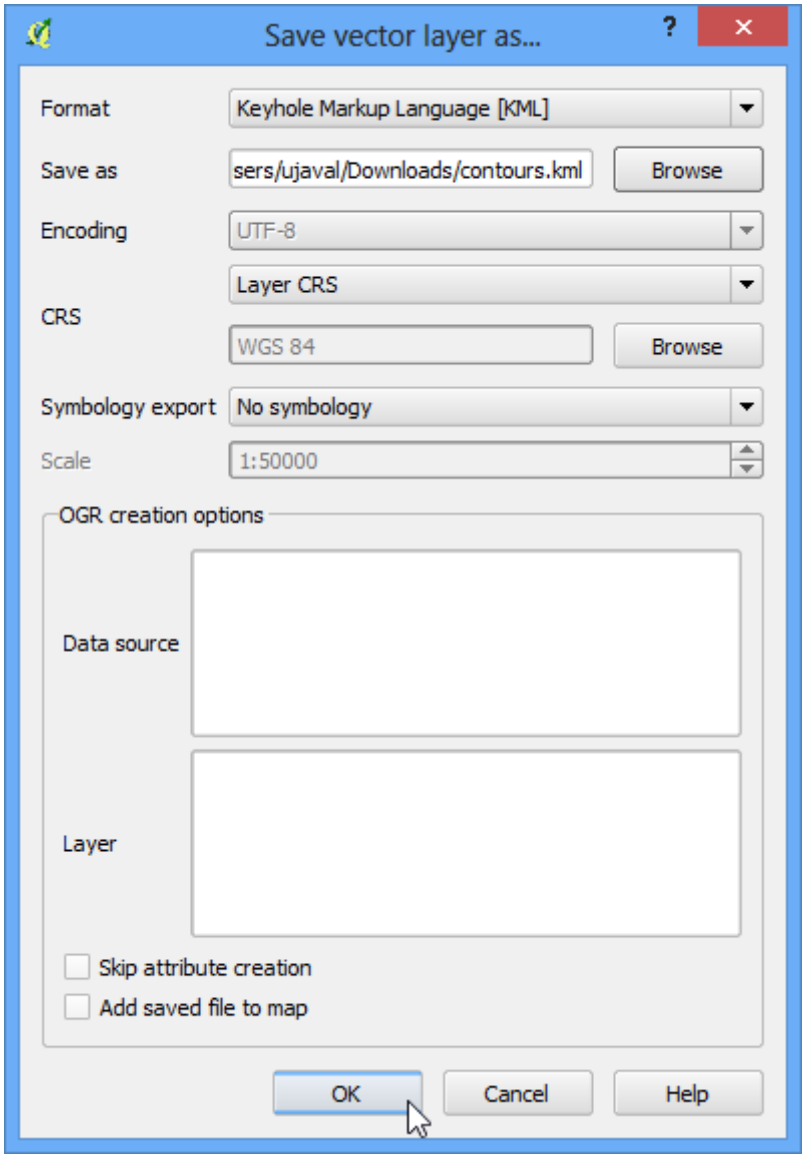
22. Now you will see the full extent of the hillshade raster.



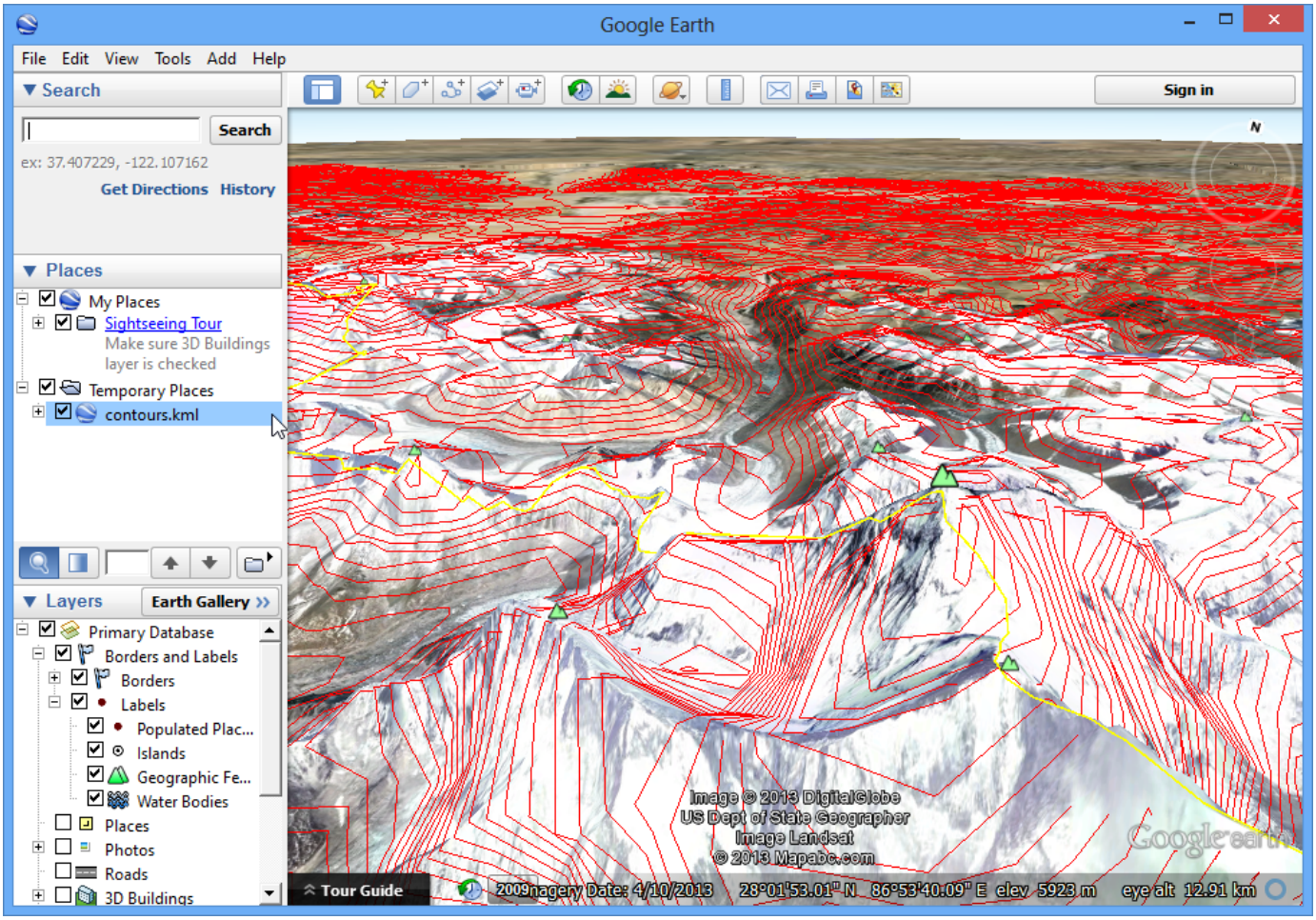
23. You can also visualize your contour layer and verify your analysis by exporting the contours layer as KML and viewing it in Google Earth. Right click on the contours layer, select Save as...



24. Select Keyhole Markup Language [KML] as the Format. Name your output as contours.kml and click OK.



25. Browse to the output file on your disk and double-click on it to open Google Earth.



Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Working with WMS Data

Often you need reference data layers for your basemap or to display your results in the context of other datasets. Many organizations publish datasets online that can be readily used in GIS. A popular standard for publishing maps online is called **WMS (Web Map Service)**. This is a better choice for using reference layers as you get access to rich datasets in your GIS without the hassle of downloading or styling the data.

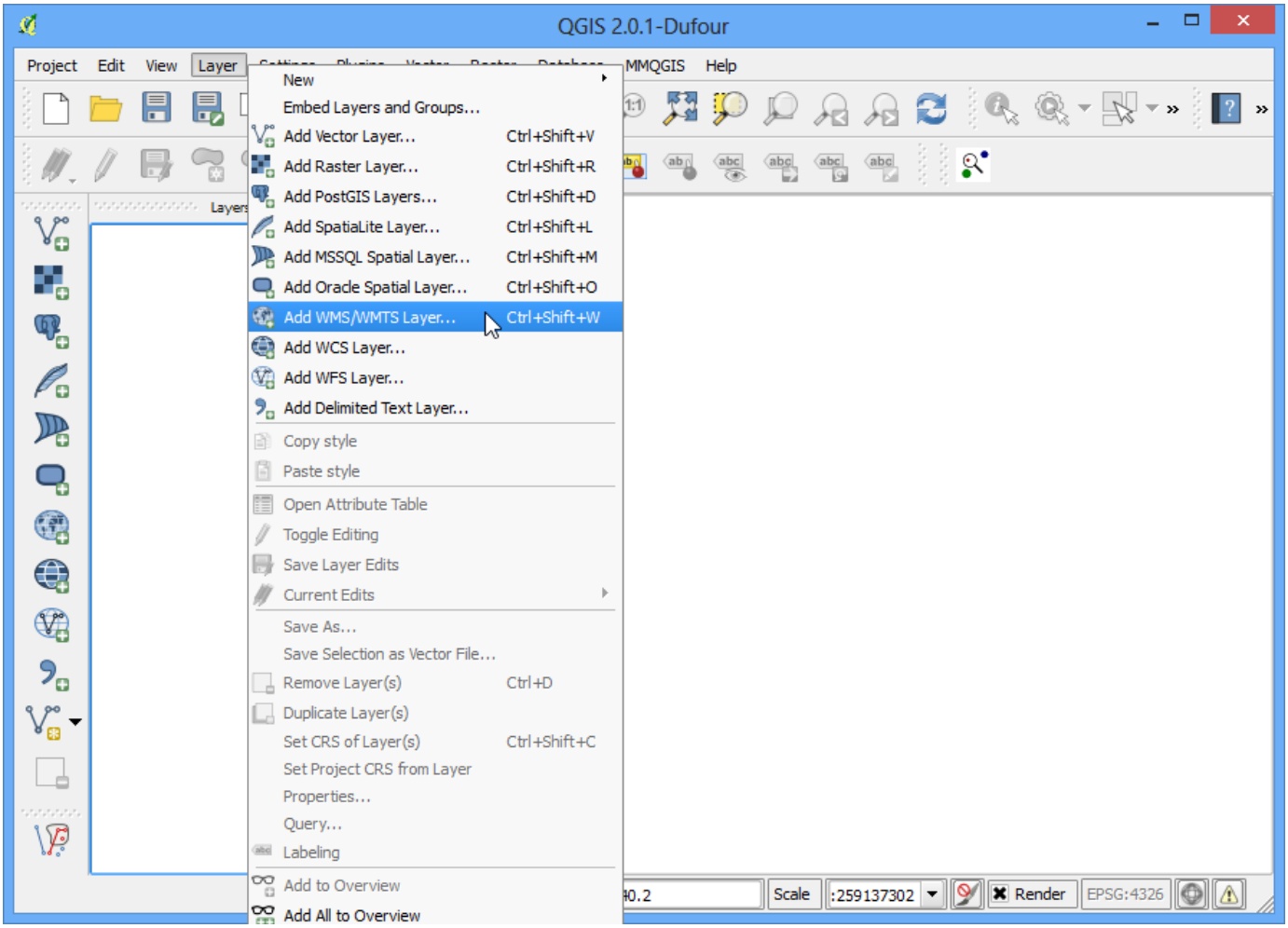
Overview of the task

In this tutorial, we will load layers of Mineral Resources (http://mrdata.usgs.gov/wms.html) published by USGS.

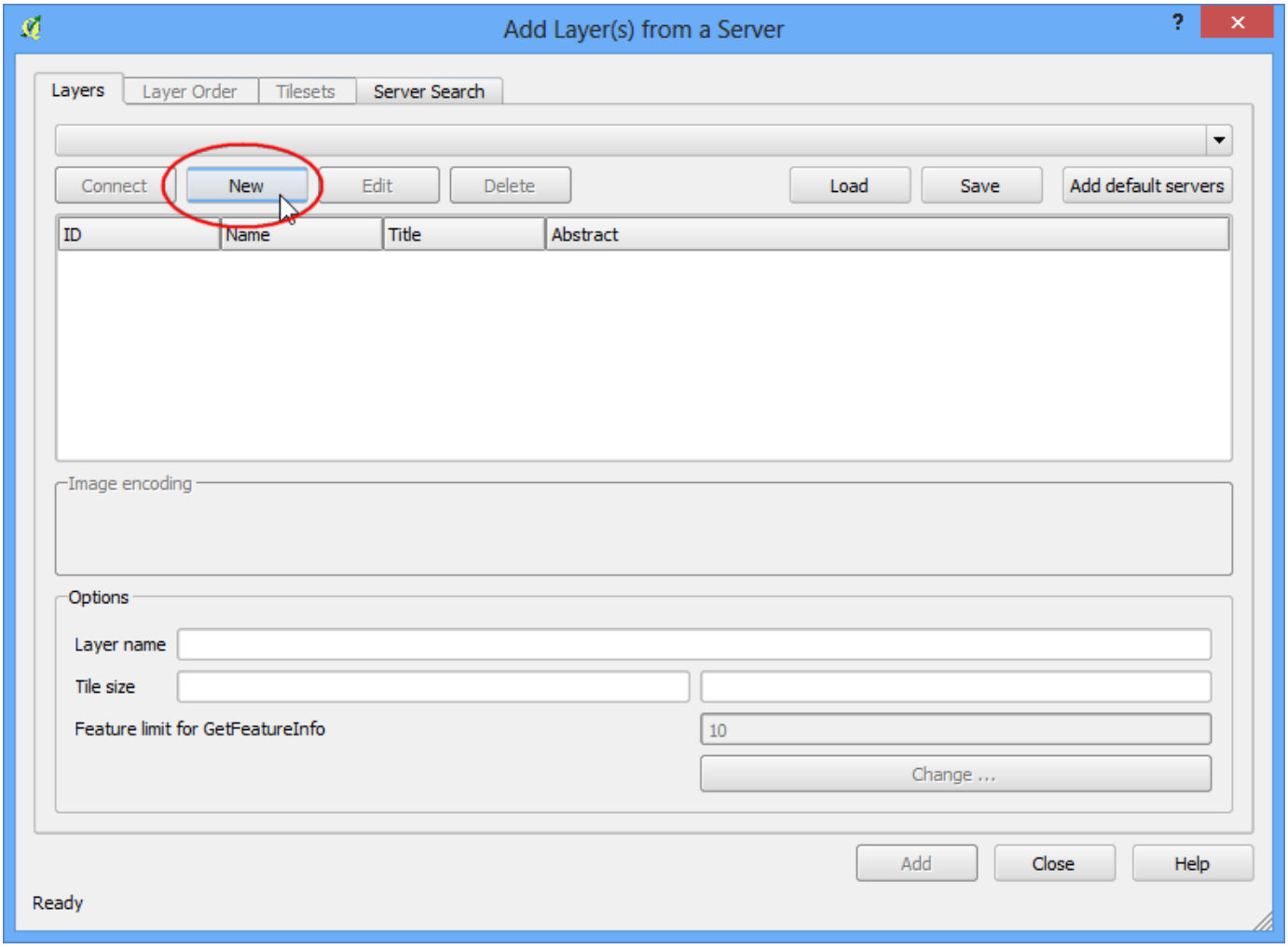
Data Source: [MRDATA] (credits.html#mrdata)

Procedure

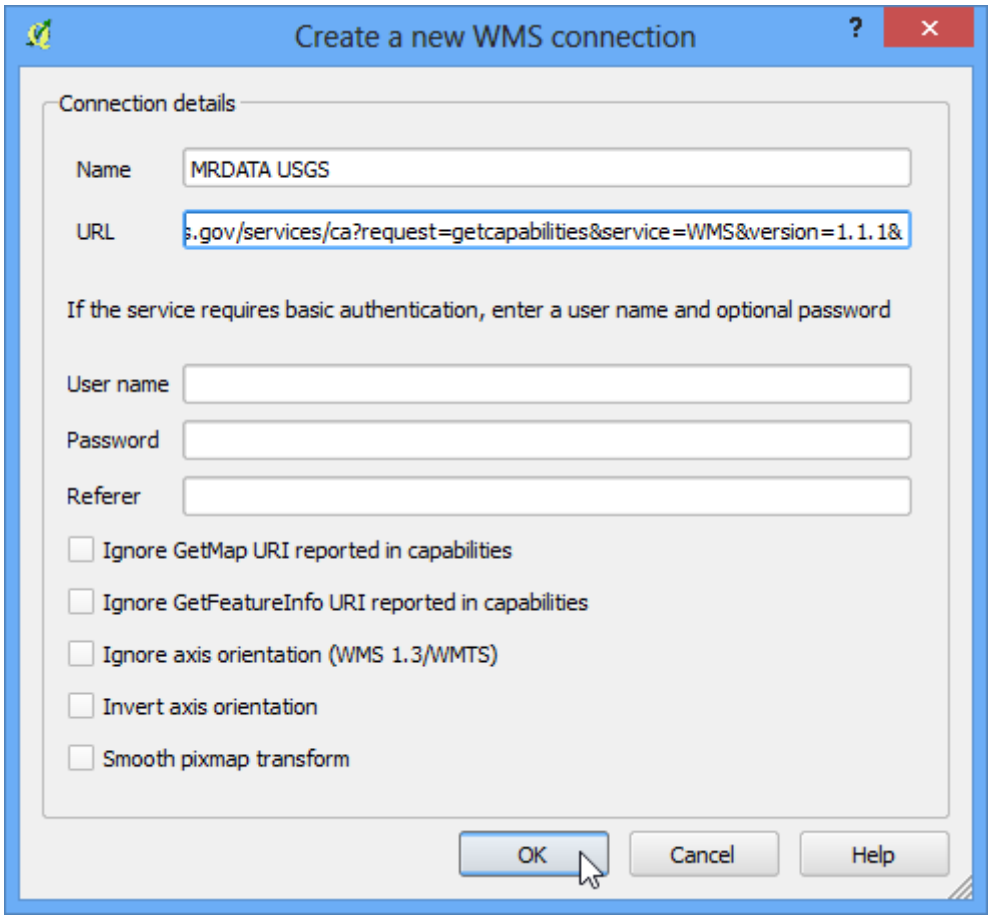
1. Open QGIS and go to on Layer ‣ Add WMS Layer....



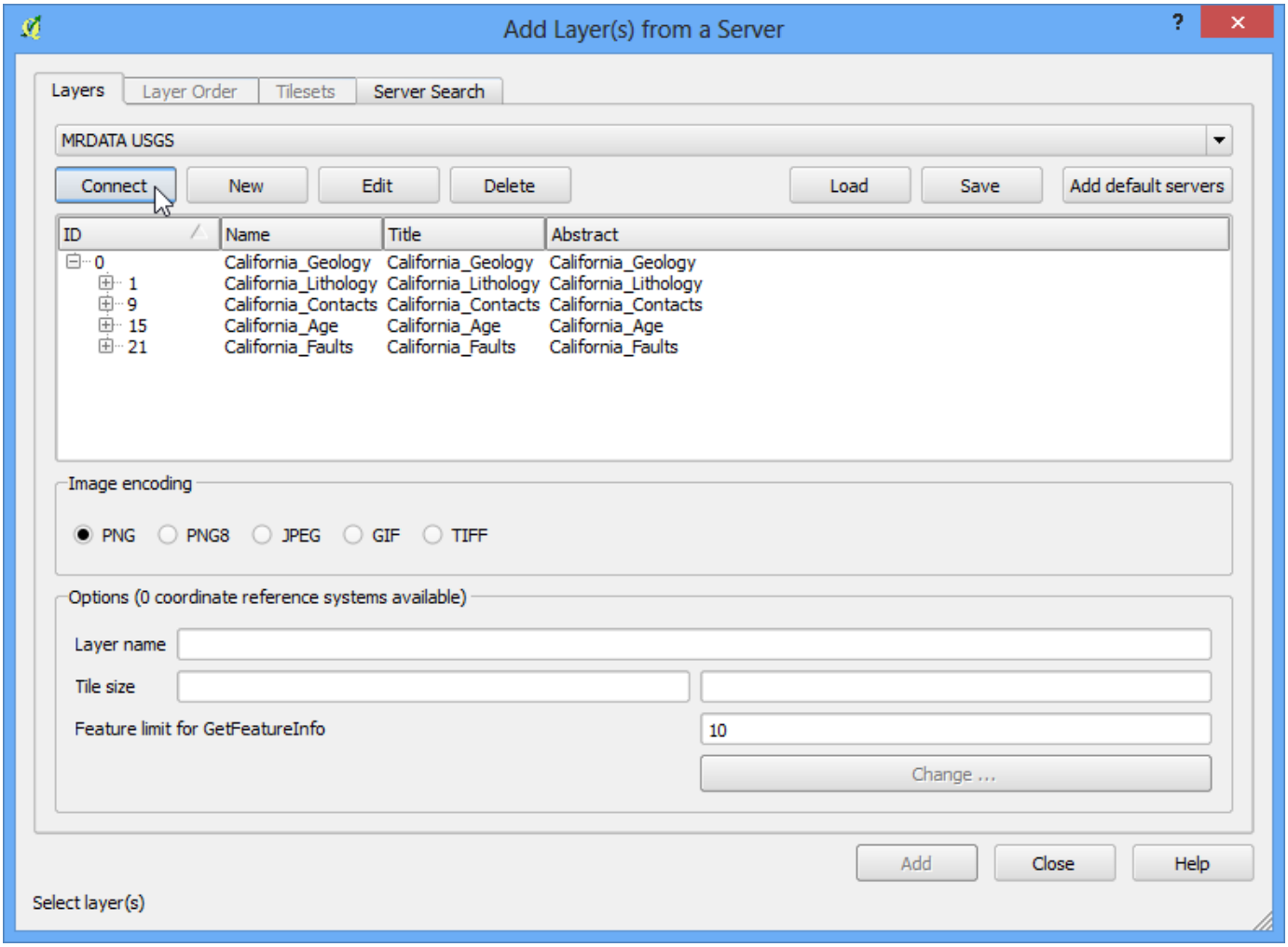
2. In the Layers tab, click on New.



3. Name your connection. This is not the name of the layer but the name of service which is offering the WMS layer. A single service usually offers multiple layers that can be added to your project. The URL that you need to access a WMS layer is called *GetCapabilities*. When you access a WMS server with this parameter in the URL, it returns a list of layers available along with various metadata. In this case, name the connection as `MRDATA USGS` and the GetCapabilities URL as `http://mrdata.usgs.gov/services/ca?request=getcapabilities&service=WMS&version=1.1.1&`. Click OK.



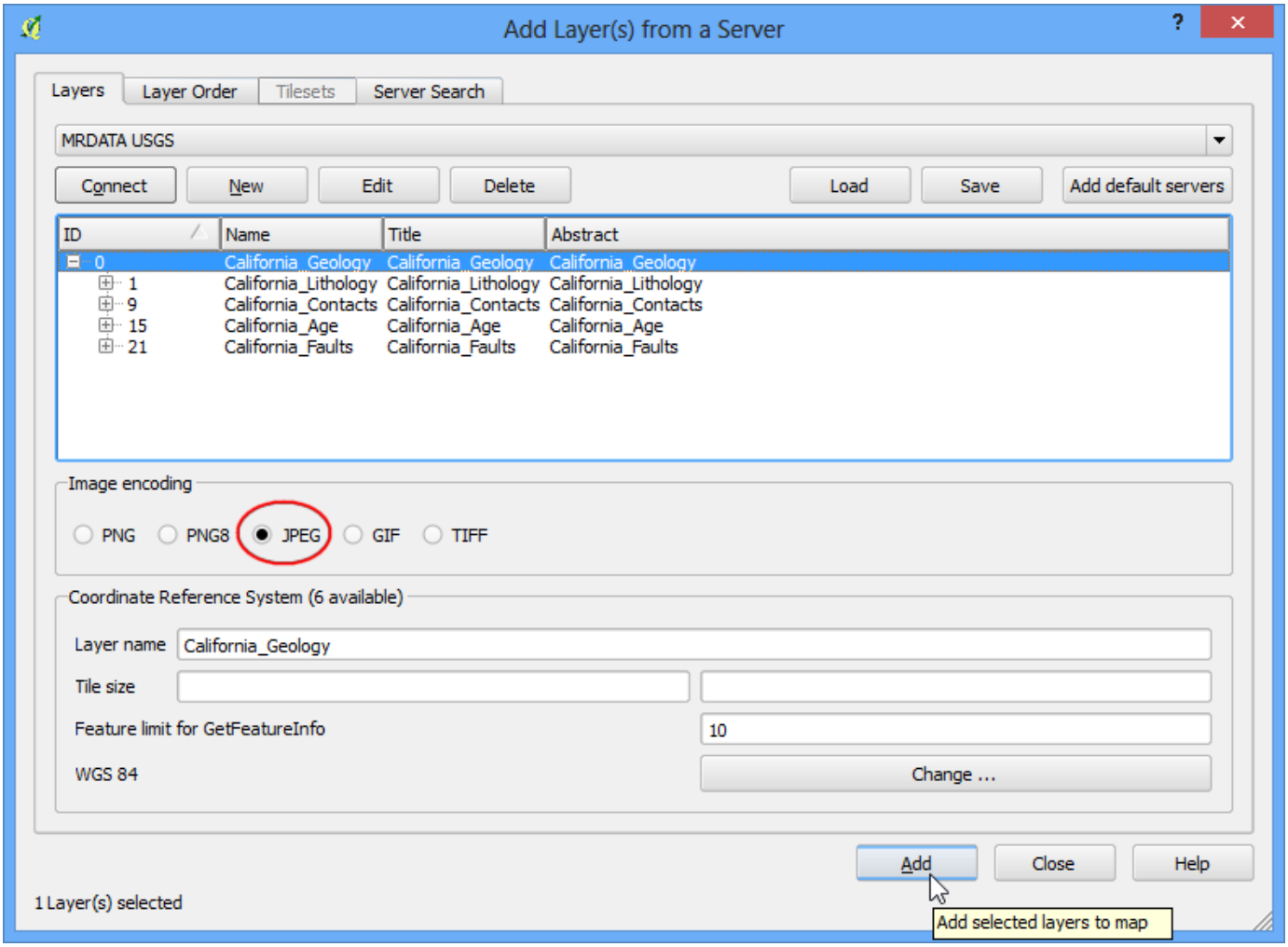
4. Next, click on the Connect button to fetch the list of layers available. You will notice different IDs listed next to the layers. ID `0` means you get a map of all the layers. If you do not want all the layers, you can expand the list by clicking on + icon and selecting the layer of interest. Select the layer `0` for this tutorial.



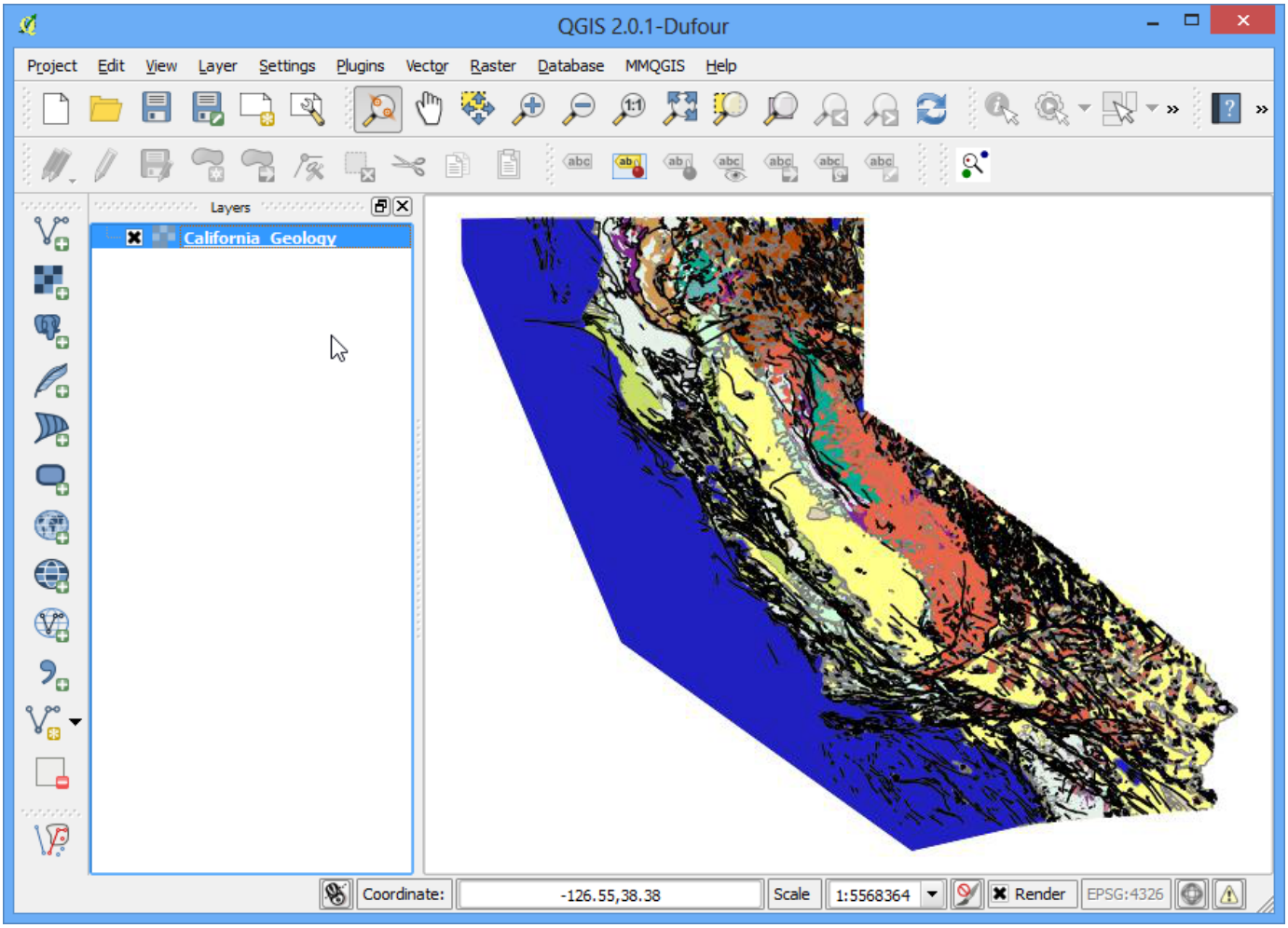
5. In the Image encoding section, you need to choose an image format. Image formats matter a great deal and which one you choose depends on your use case. Here are some pointers

    - Quality: PNG is a lossless compressed image format. JPEG is lossy compressed format. TIFF can be either. That means the quality of PNG images will be better compared to JPEG. If your main purpose is to print a map, use PNG.
    - Speed: Since PNG images are uncompressed and thus larger in size, they will take longer to load. If you are using the layer in your project as a reference layer and need to zoom/pan a lot, use JPEG.
    - Client Support: QGIS supports most of the formats, but if you are developing web applications, browsers usually do not support TIFF, so you should choose another format.
    - Type of data: If your layers are primarily vector, PNG will give better results. For imagery layers, JPEG is usually a better choice.

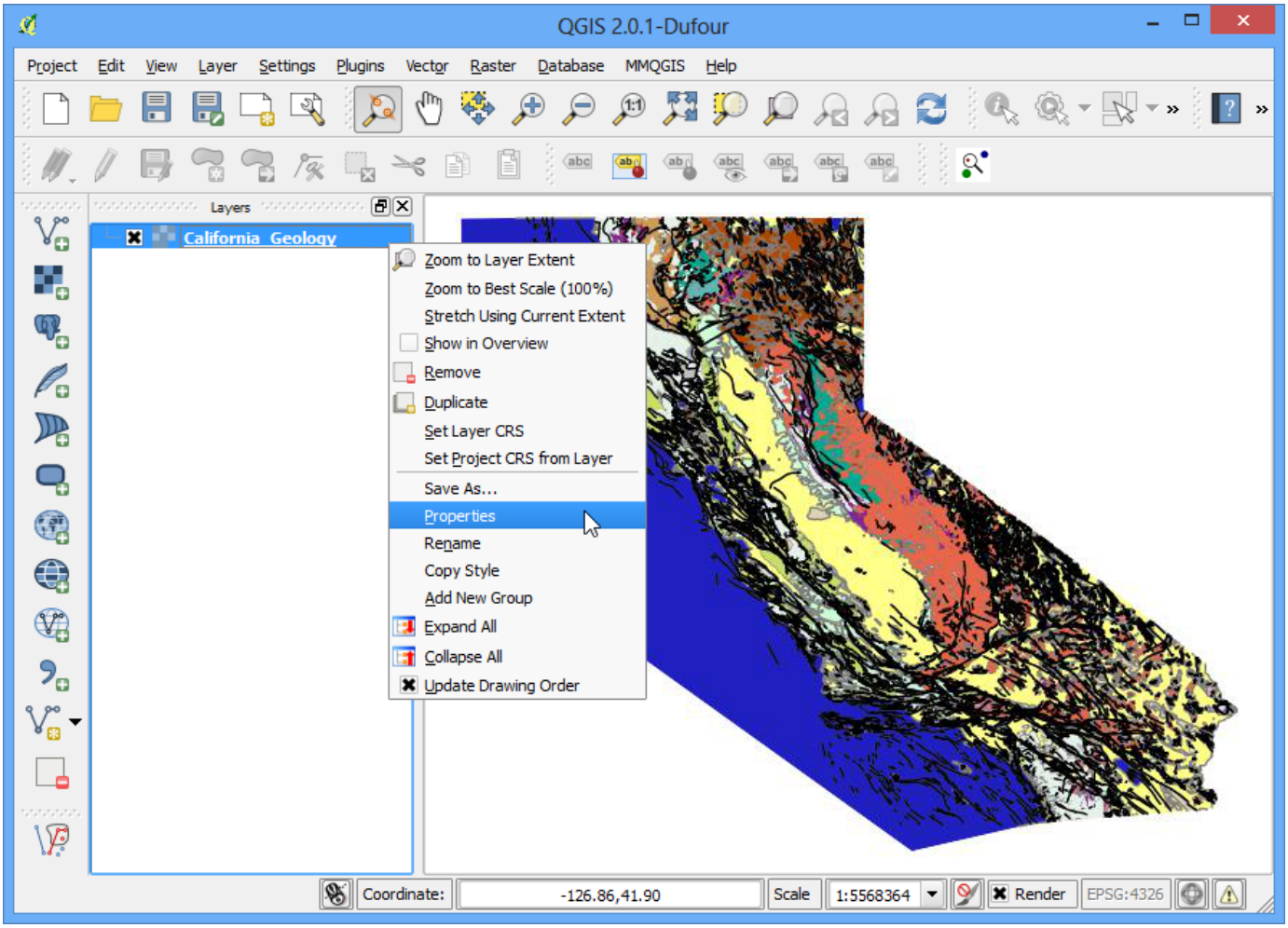
    For this tutorial, choose JPEG as the format. Change the Layer name if you wish and click Add.



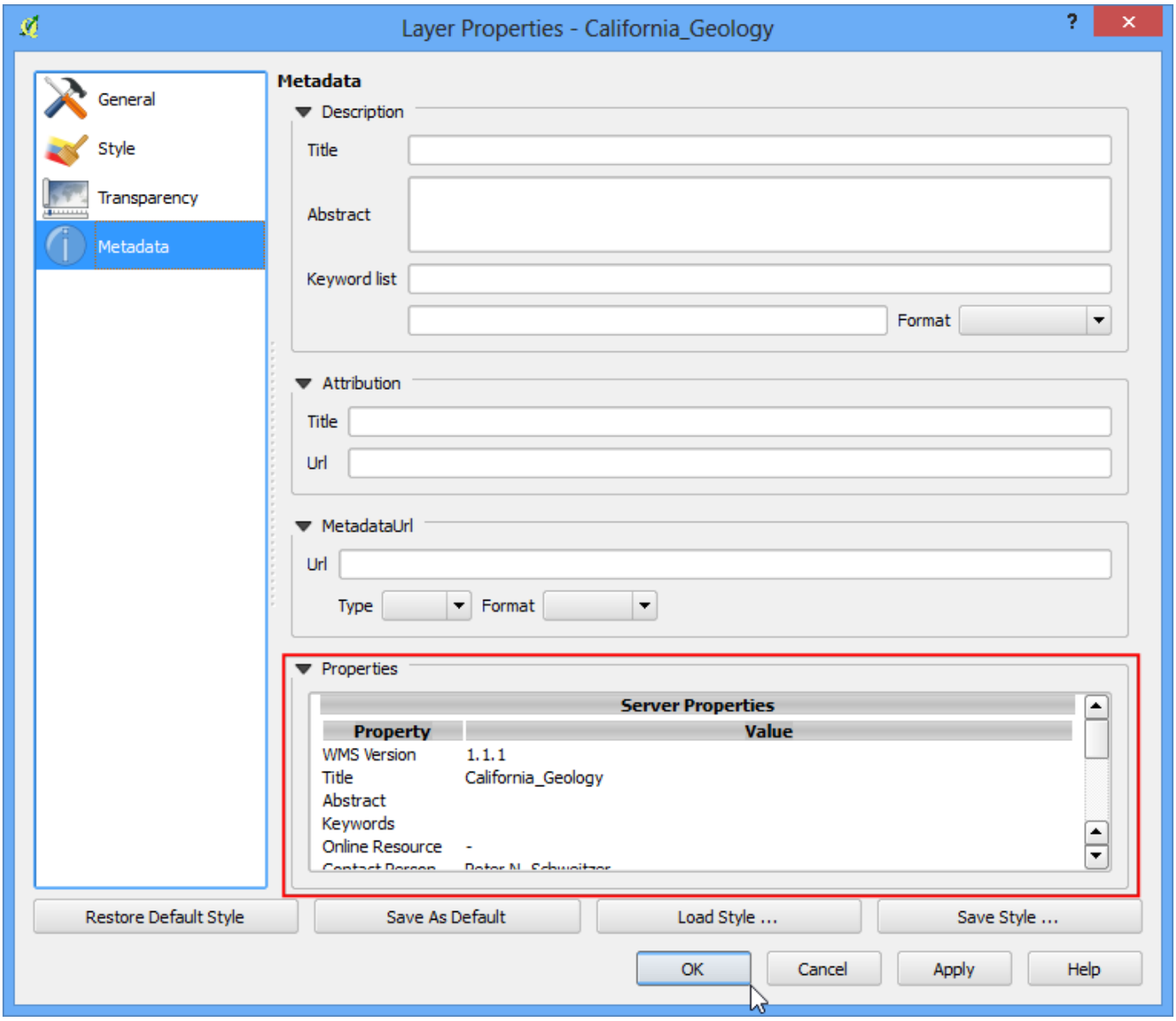
6. You will see the layer loaded in the QGIS canvas. You can zoom/pan around just like any other layer. The way WMS service works is that every time you zoom/pan, it sends your viewport coordinates to the server and the server creates an image for that viewport and return it to the client. So there will be some delay before you see the image for the area after you have zoomed in. Also, since the data you see is an image, there is no way to query for attributes like in a regular vector/imagery layer.



7. You can, however, see some metadata about the layer. Right-click the layer and choose Properties.



8. You will notice that the Properties dialog looks different and has fewer tabs. You can go to the Metadata tab to learn more about the WMS service and the layers.



Back to top

Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Working with Projections

Map projections - or Coordinate Reference System (CRS) - often cause a lot of frustration when working with GIS data. But proper understanding of the concepts and access to the right tools will make it much easier to deal with projections. In this tutorial, we will explore how projections work in QGIS and learn about tools available for vector and rasters - particularly re-projecting vector and raster data, enabling on-the-fly re-projection and assigning projection to data without projection.

Overview of the task

The task is to re-project and overlay data layers of difference projections together in QGIS.

Other skills you will learn

- Use `.tfw` files to georeference to rasters.
- How to save selected features from a layer to a new layer.
- How to view metadata information for layers in QGIS.

Get the data

Natural Earth has Admin 0 - Countries (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/) dataset. Download the countries (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_admin_0_countries.zip)

UK's Ordnance Survey (https://www.ordnancesurvey.co.uk/) provides open data for download. Download the MiniScale raster product (https://www.ordnancesurvey.co.uk/opendatadownload/products.html) for Great Britain and extract it to a folder on your computer.

Note

You will need to enter your personal details to be able to download the Ordnance Survey dataset.

For convenience, you may directly download a copy of the dataset from the link below:

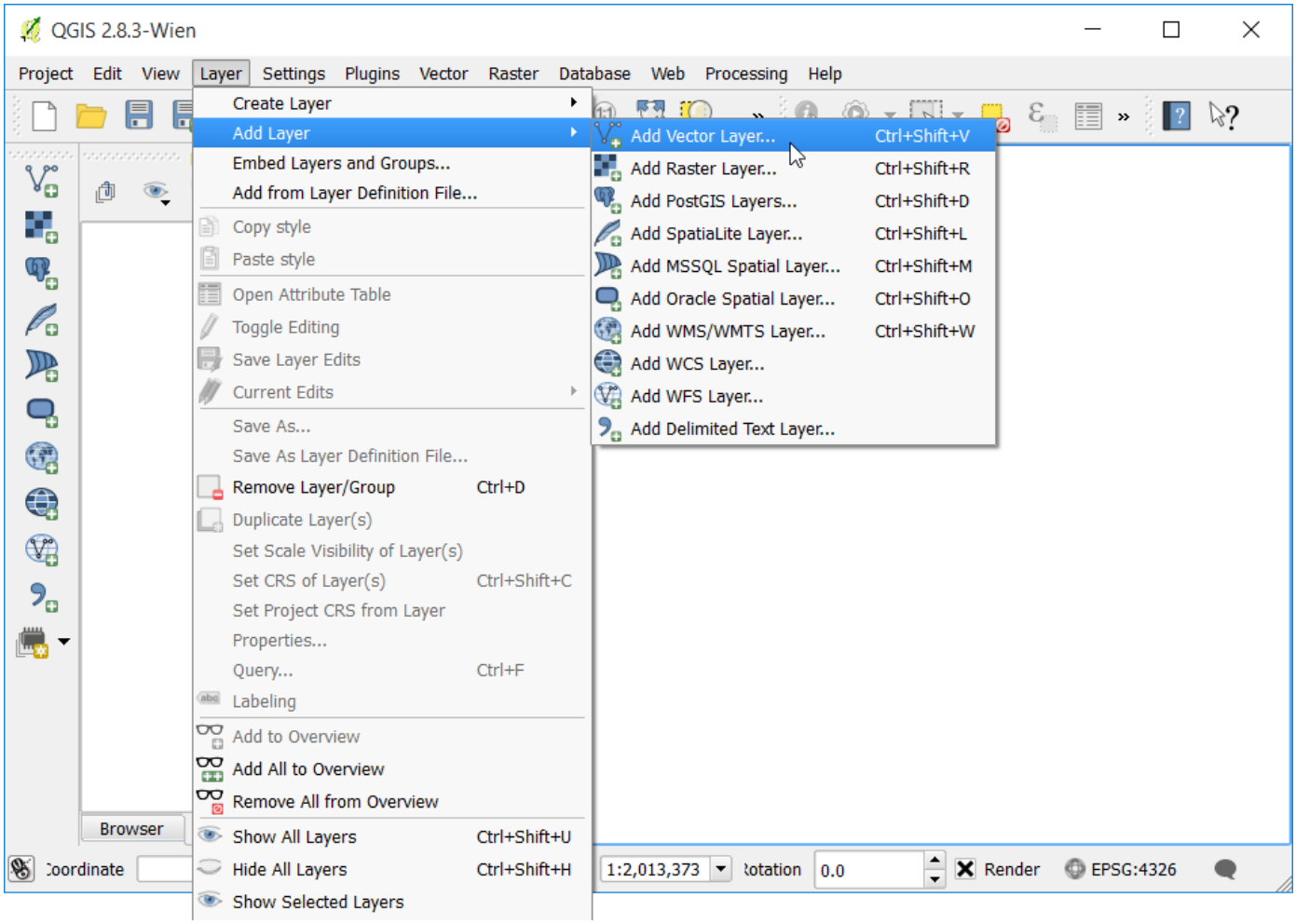
ne_10m_admin_0_countries.zip (http://www.qgistutorials.com/downloads/ne_10m_admin_0_countries.zip)

minisc_gb.zip (http://www.qgistutorials.com/downloads/minisc_gb.zip) (Contains only the files required for this tutorial)

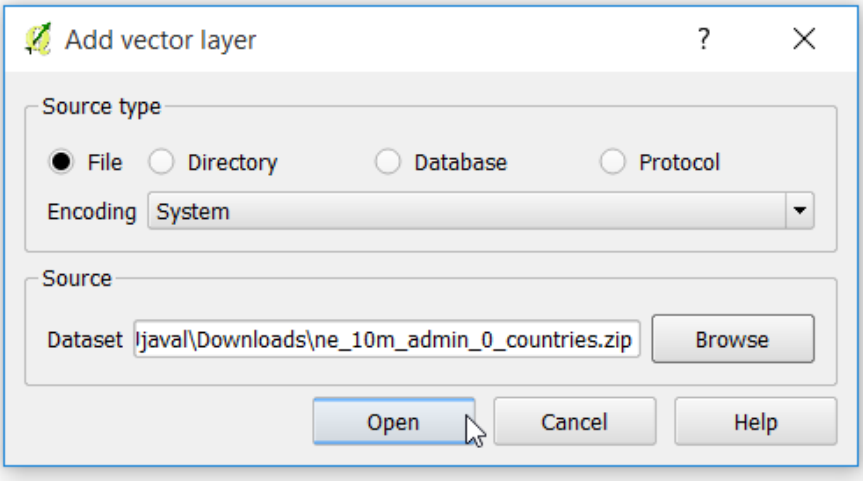
Data Sources: [NATURALEARTH] (credits.html#naturalearth) [OSOPENDATA] (credits.html#osopendata)

Procedure

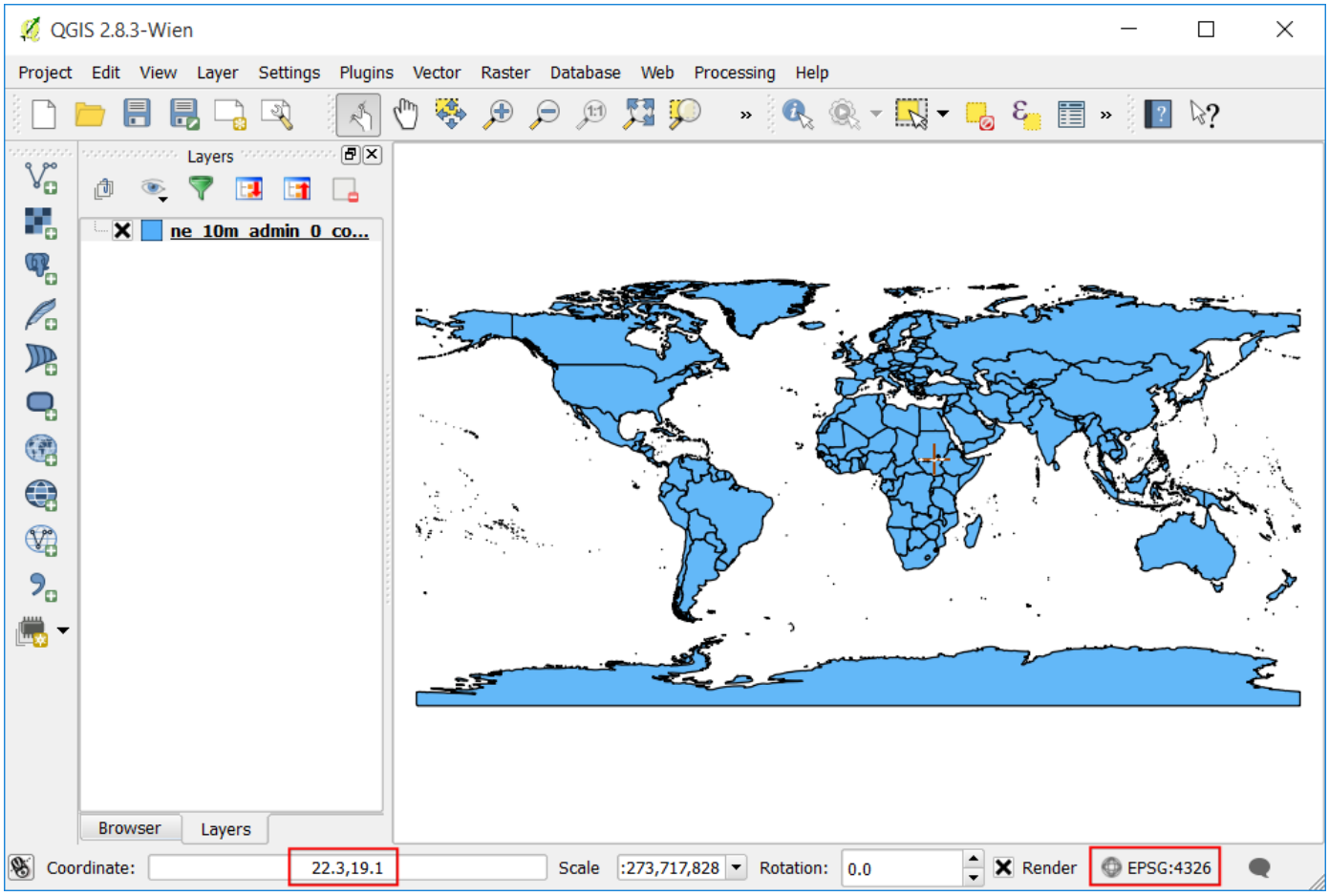
1. Open QGIS. Go to Layer ▸ Add Layer ▸ Add Vector Layer....



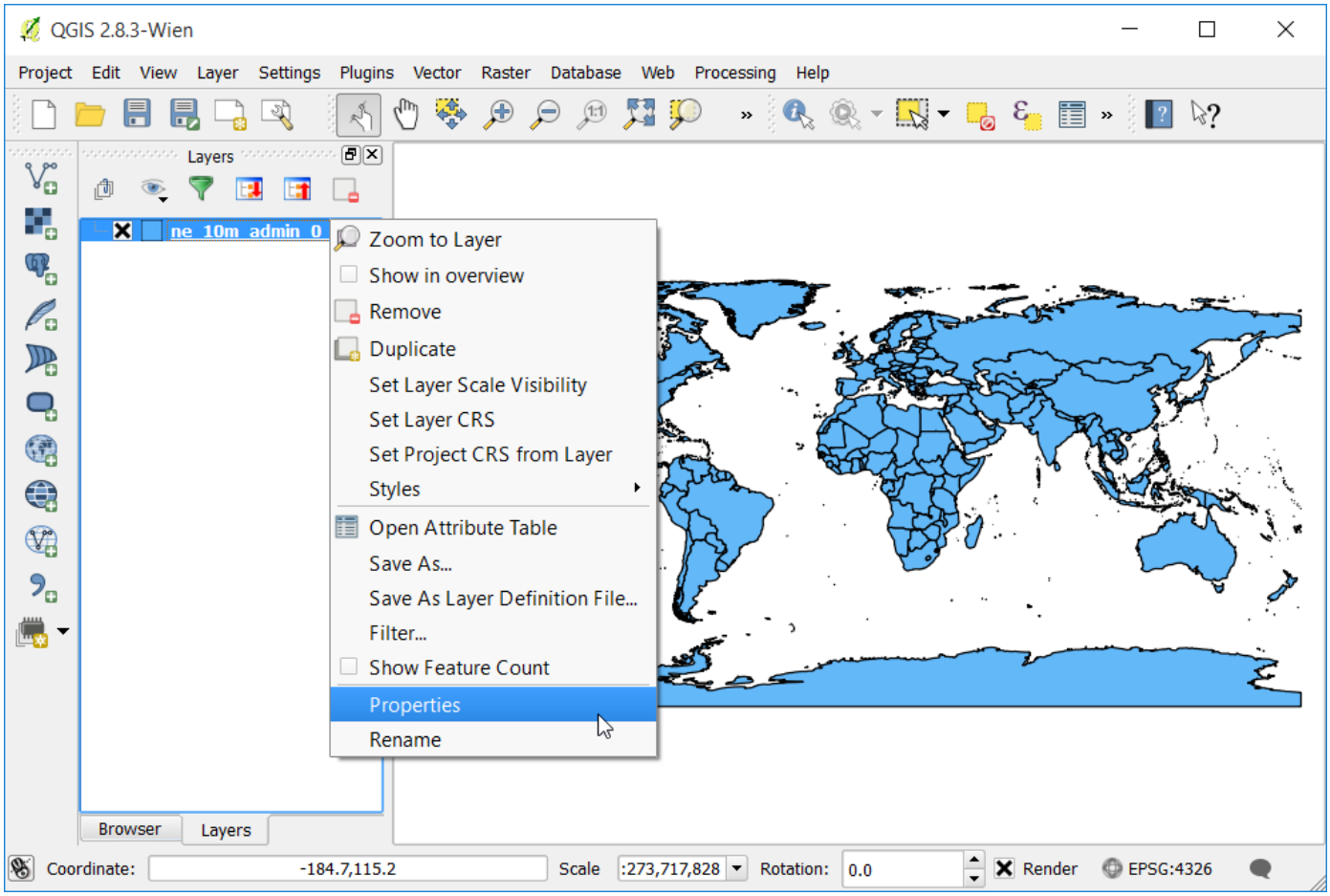
2. Browse to the downloaded `ne_10m_admin_0_countries.zip` file and click Open.



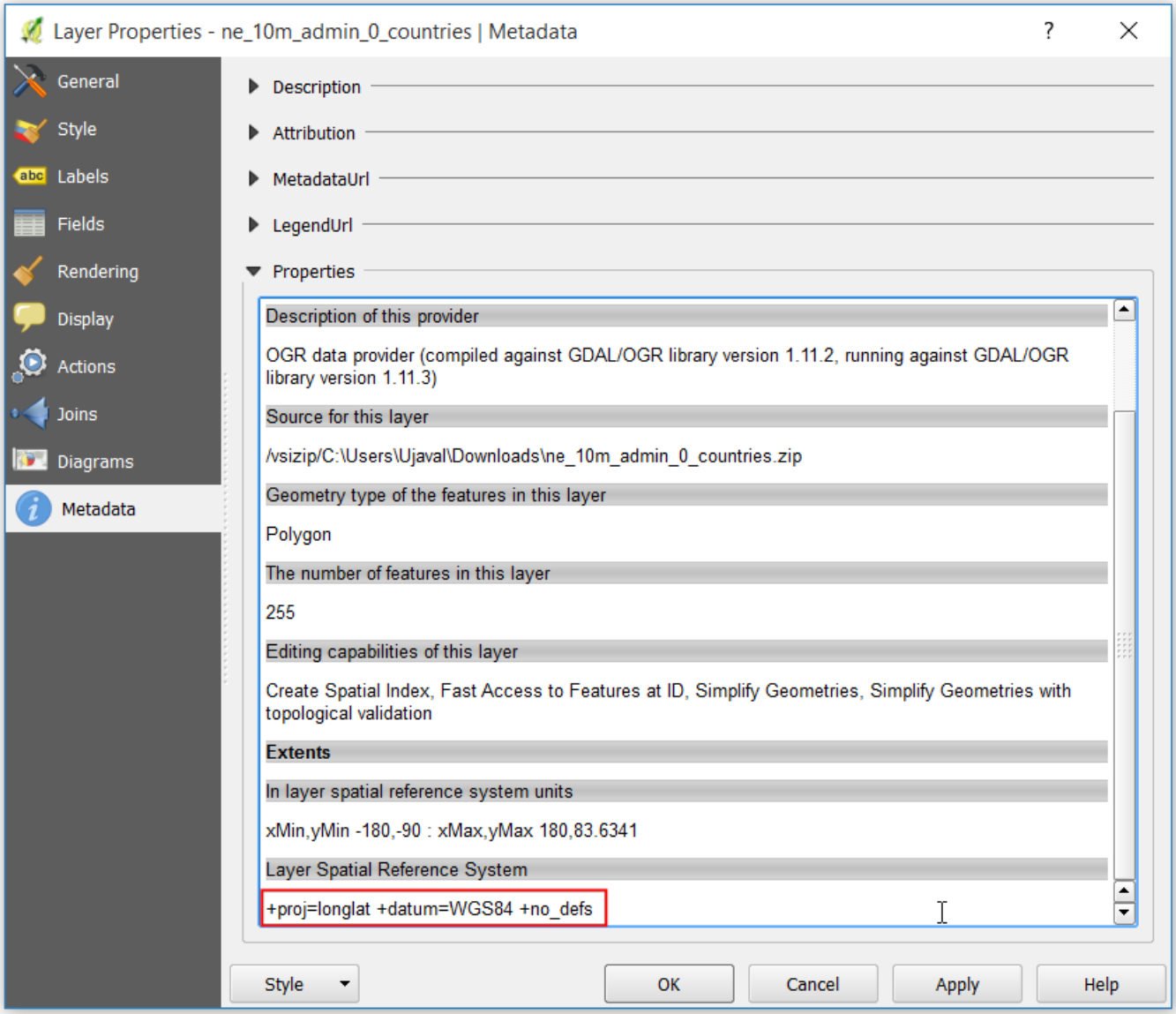
3. At the bottom of QGIS window, you will notice the label Coordinate. As you move your cursor over the map, it will show you the X and Y coordinates at that location. At the bottom-right corner you will see EPSG:4326. This is the code for the current CRS (Projection) for the project.



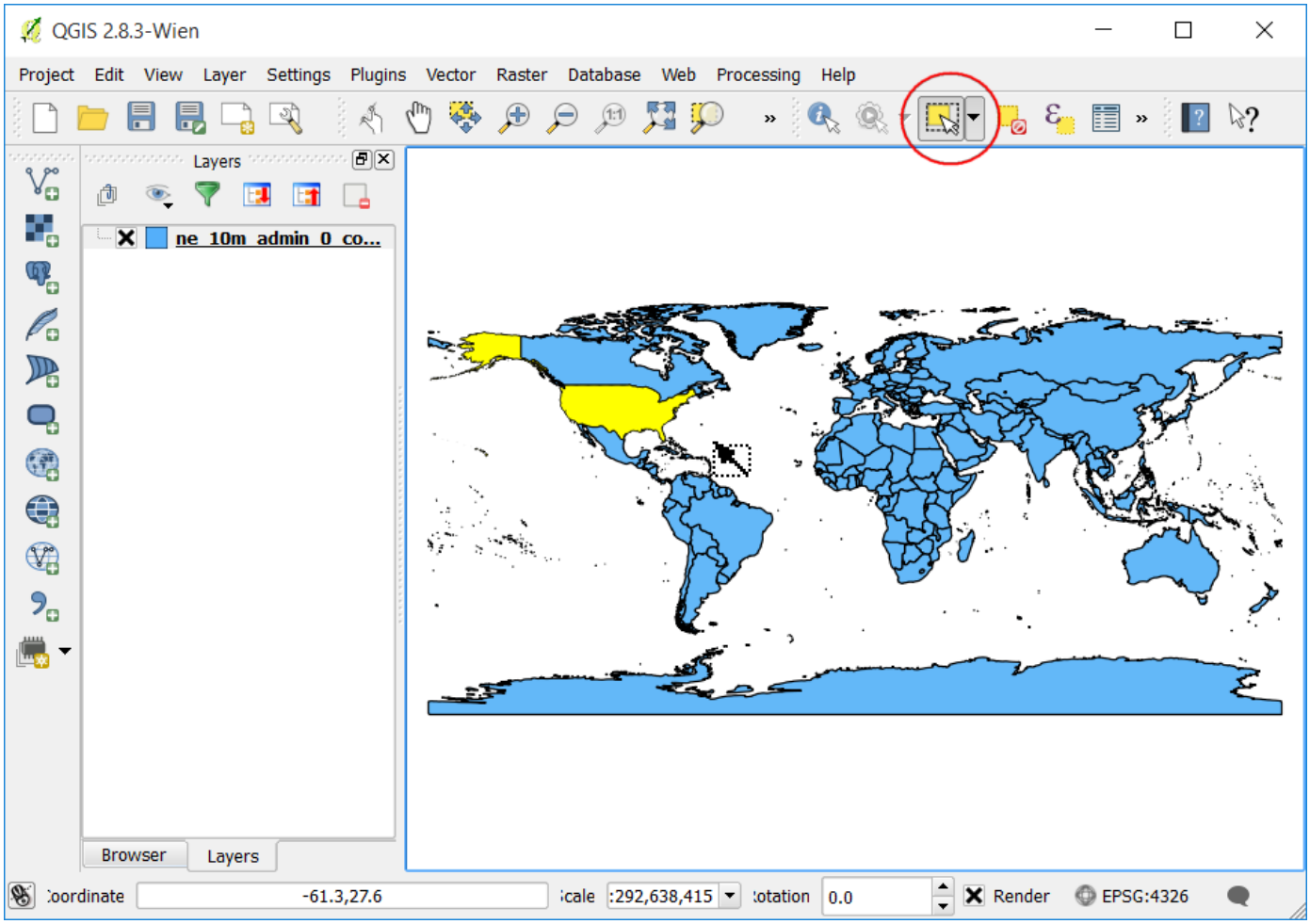
4. As you will later see, the project's CRS may not match the layer's CRS. To determine a layer's projection, we can look into the metadata. Right click on `ne_10m_admin_0_countries` layer and select Properties.



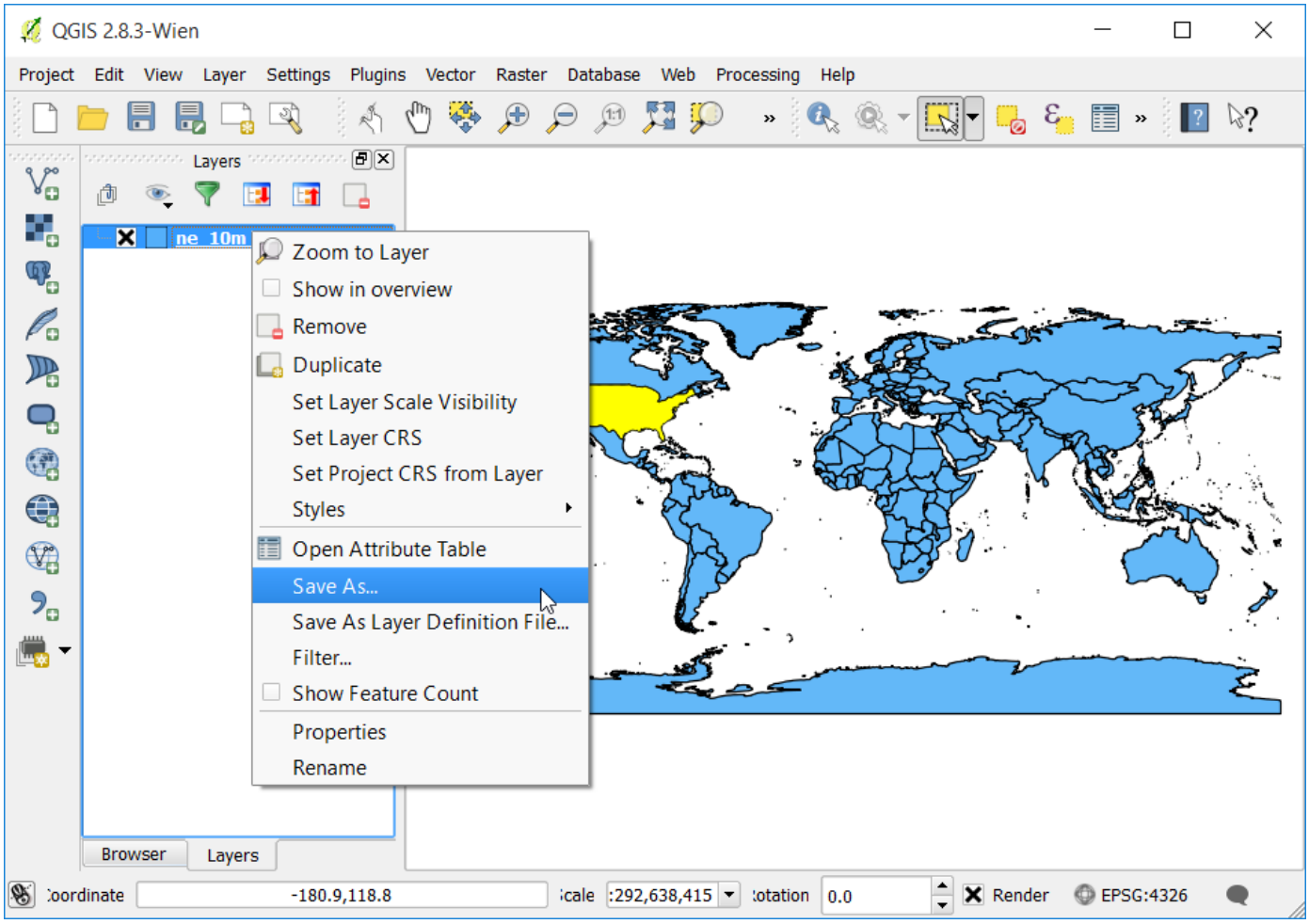
5. Switch to the Metadata tab in the Layer Properties dialog. Expand the Properties section. At the bottom, you will see the definition for the projection under Layer Spatial Reference System. This definition is in the PROJ.4 format (https://en.wikipedia.org/wiki/PROJ.4).



6. Now let's see how we can change the layer's projection. This operation is called **Re-Projection**. Rather than re-projecting the entire layer, we can also re-project some features from the layer. Use the Select features by area or single click tool and click on United States feature to select it.

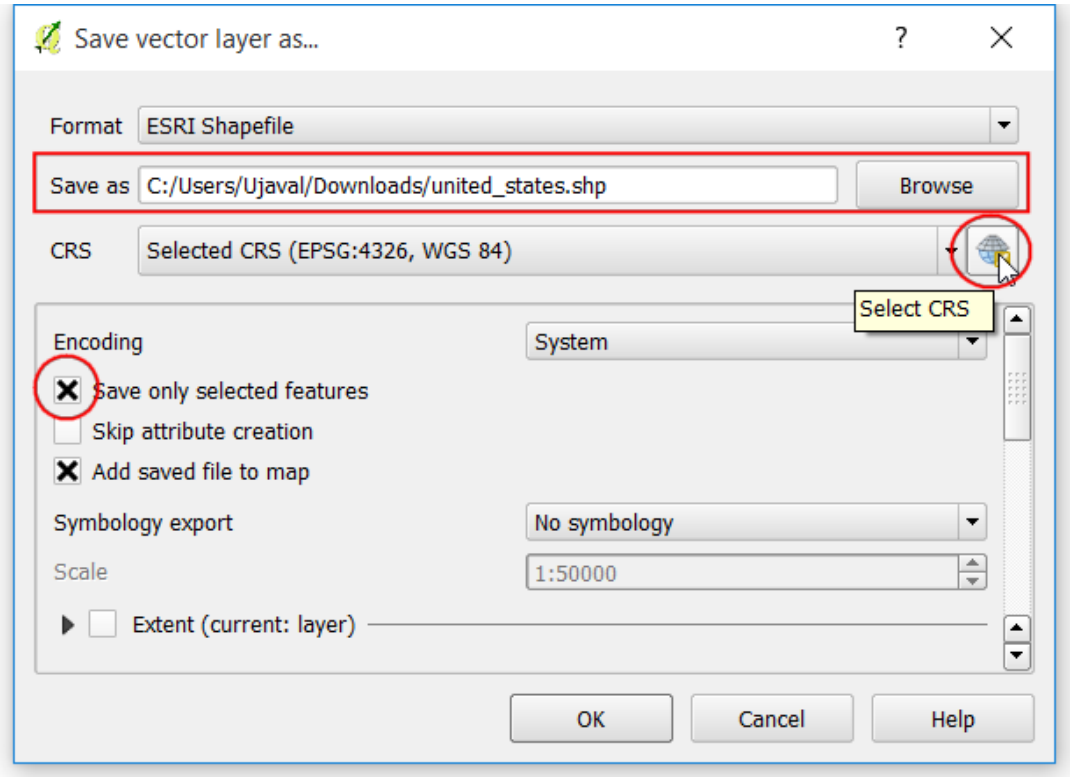


7. Right-click the `ne_10m_admin_0_countries` layer and select Save As.



8. In the Save vector layer as... dialog, name the output layer as `united_states.shp` . Also check the Save only selected features box. This will ensure that only the selected feature gets re-projected and exported. Next, we choose the new

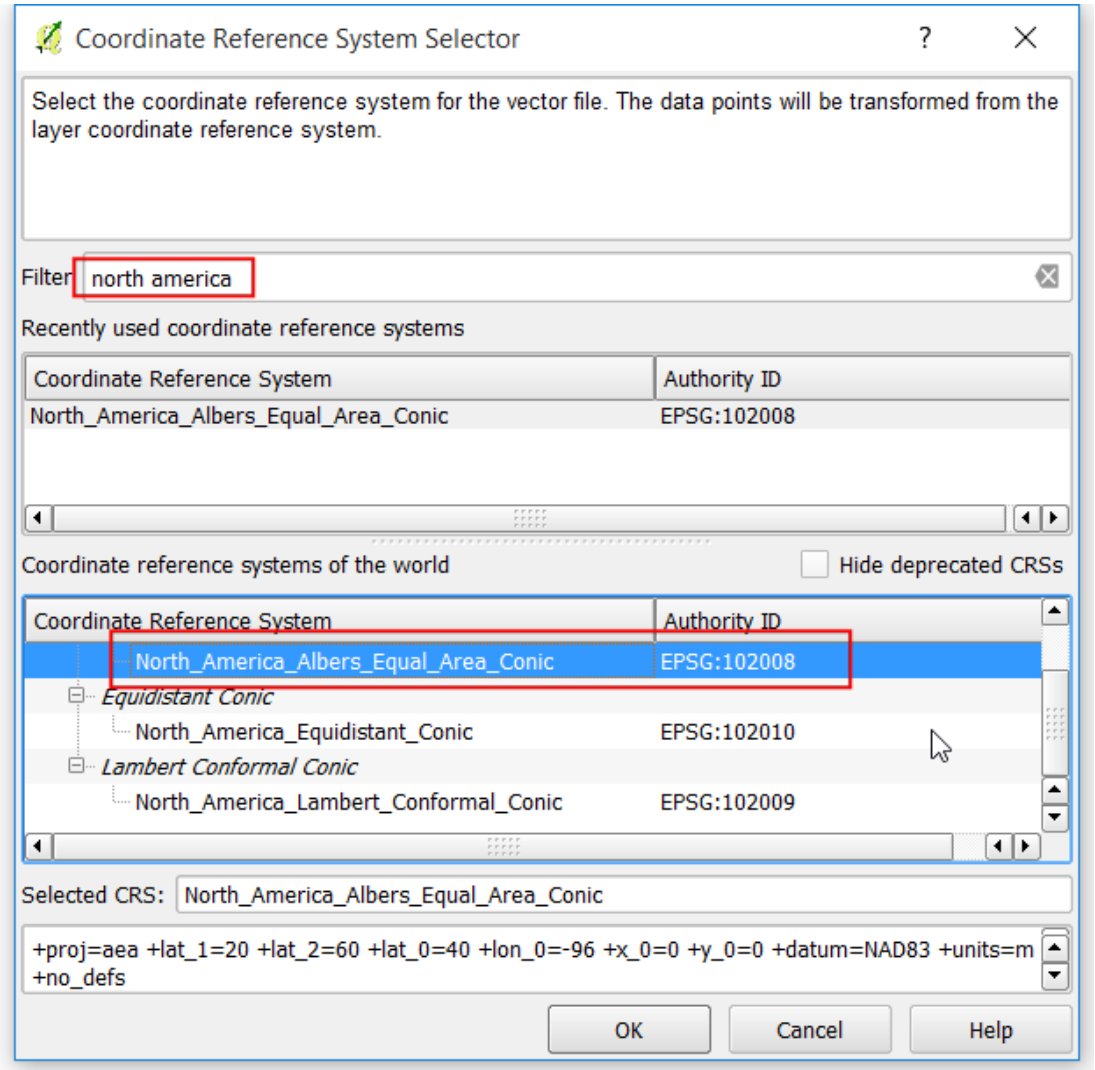
projection for the layer. Click on the Select CRS button.



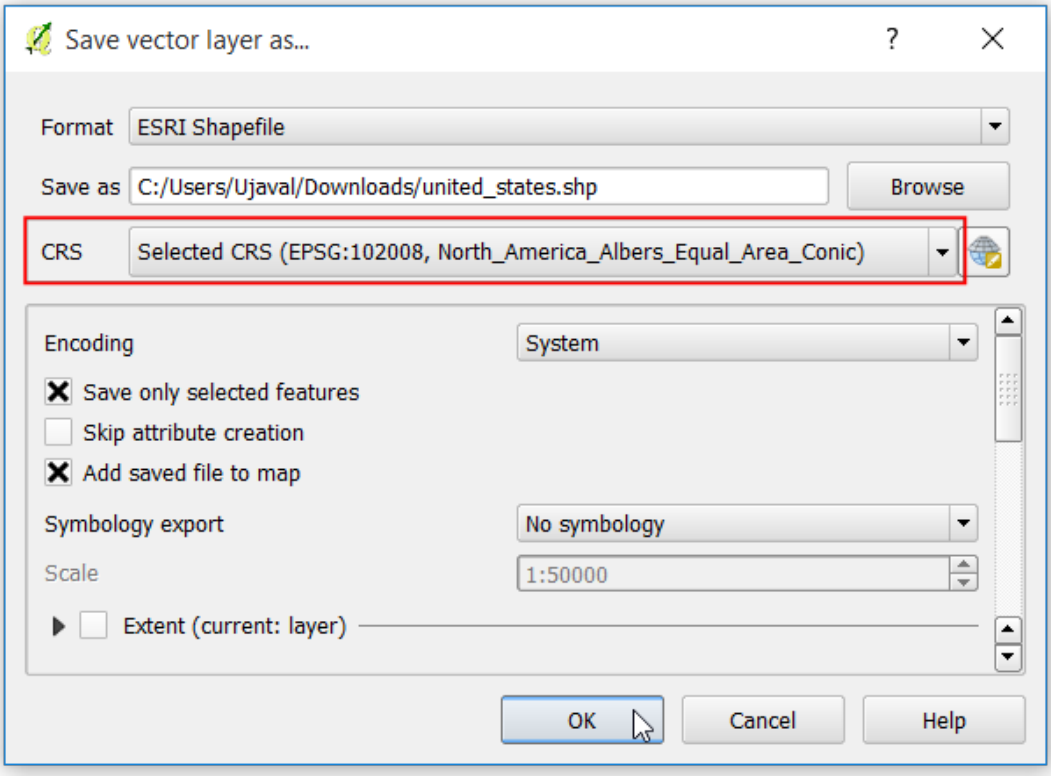
9. In the Coordinate Reference System Selector dialog, enter `north america` in the Filter search box. Scroll through the results and select `North_America_Albers_Equal_Area_Conic (EPSG:102008)` projection and click OK.

Note

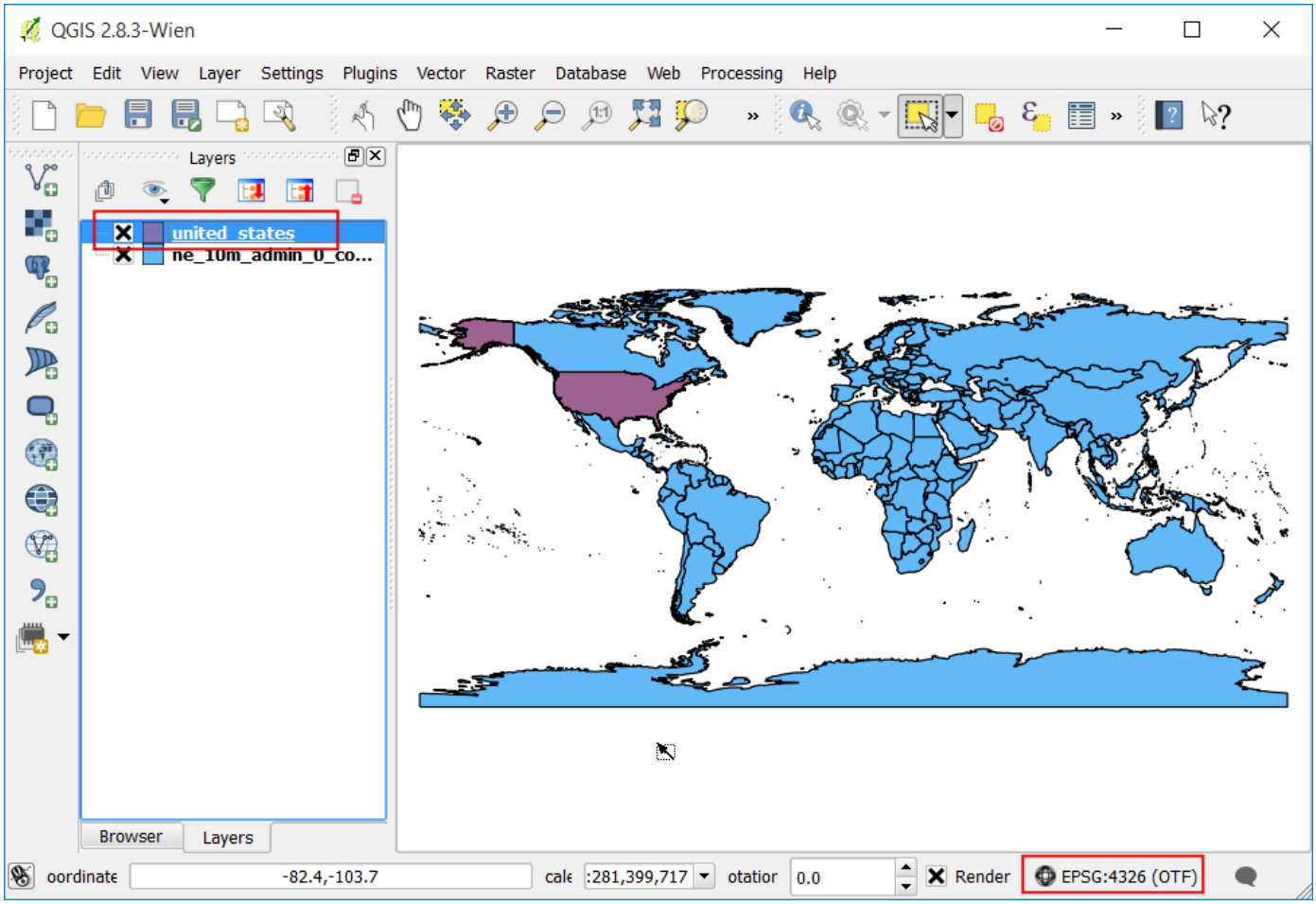
We choose Albers Equal Area Conic projection for this tutorial as it is a popular projection choice for thematic maps of the US. The choice of projection for your particular use-case will depend on a lot of factors. See this guide (http://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/coordinate_reference_systems.html) for a good overview of Projections.



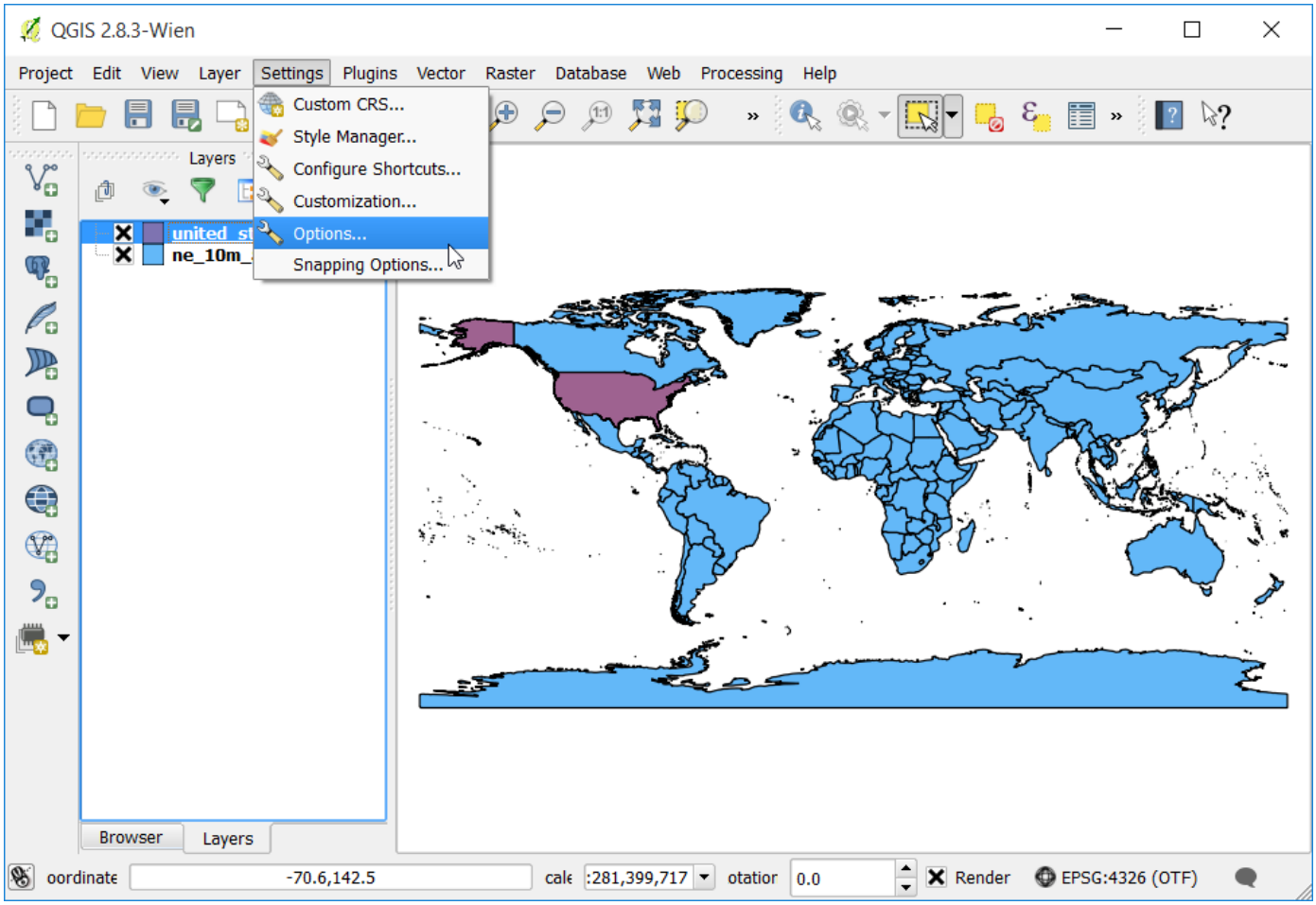
10. You will see the new CRS selected in the Save vector layer as... dialog. Click OK.



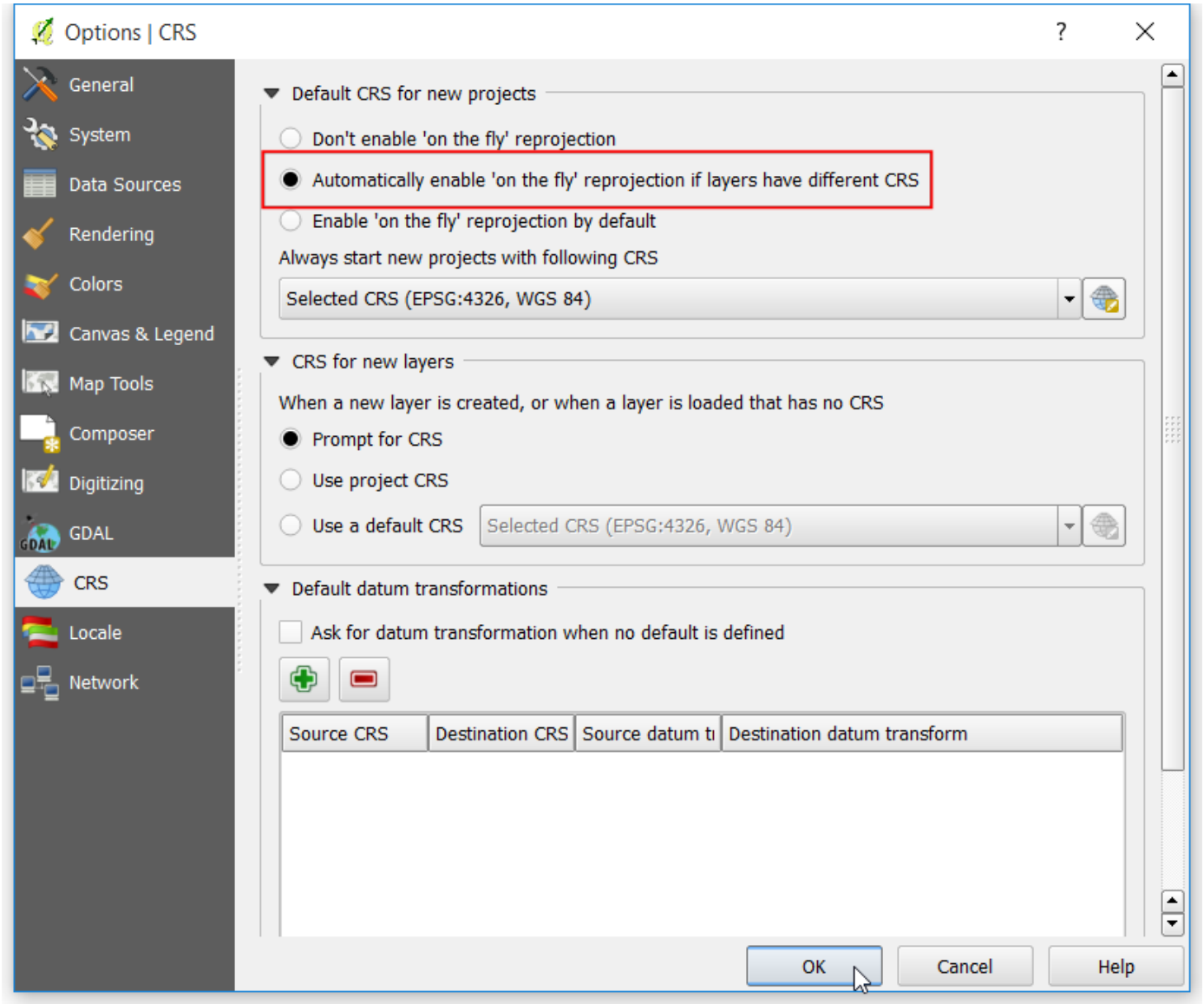
11. Once the re-projected layer gets loaded, you will notice that the new `united_states` layer overlays perfectly on top of `ne_10m_admin_0_countries` layer - even though they are in different projections. This is because QGIS has a feature called **On-the-fly CRS transformation**. The projection text at the bottom-right of QGIS now has the words `OTF` next the EPSG:4326`. To learn more, let's explore the CRS option in QGIS.



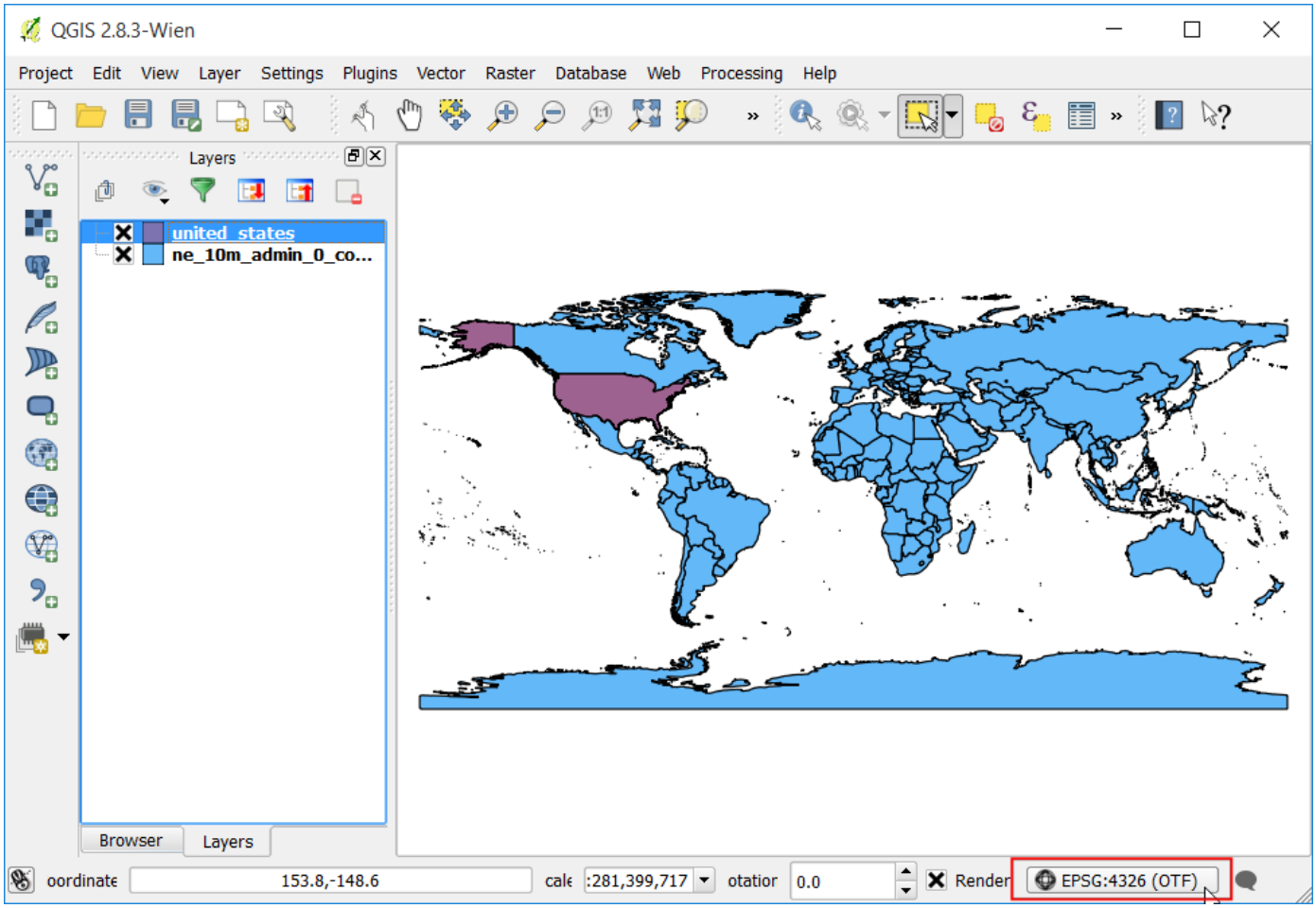
12. Go to Settings ▸ Options....



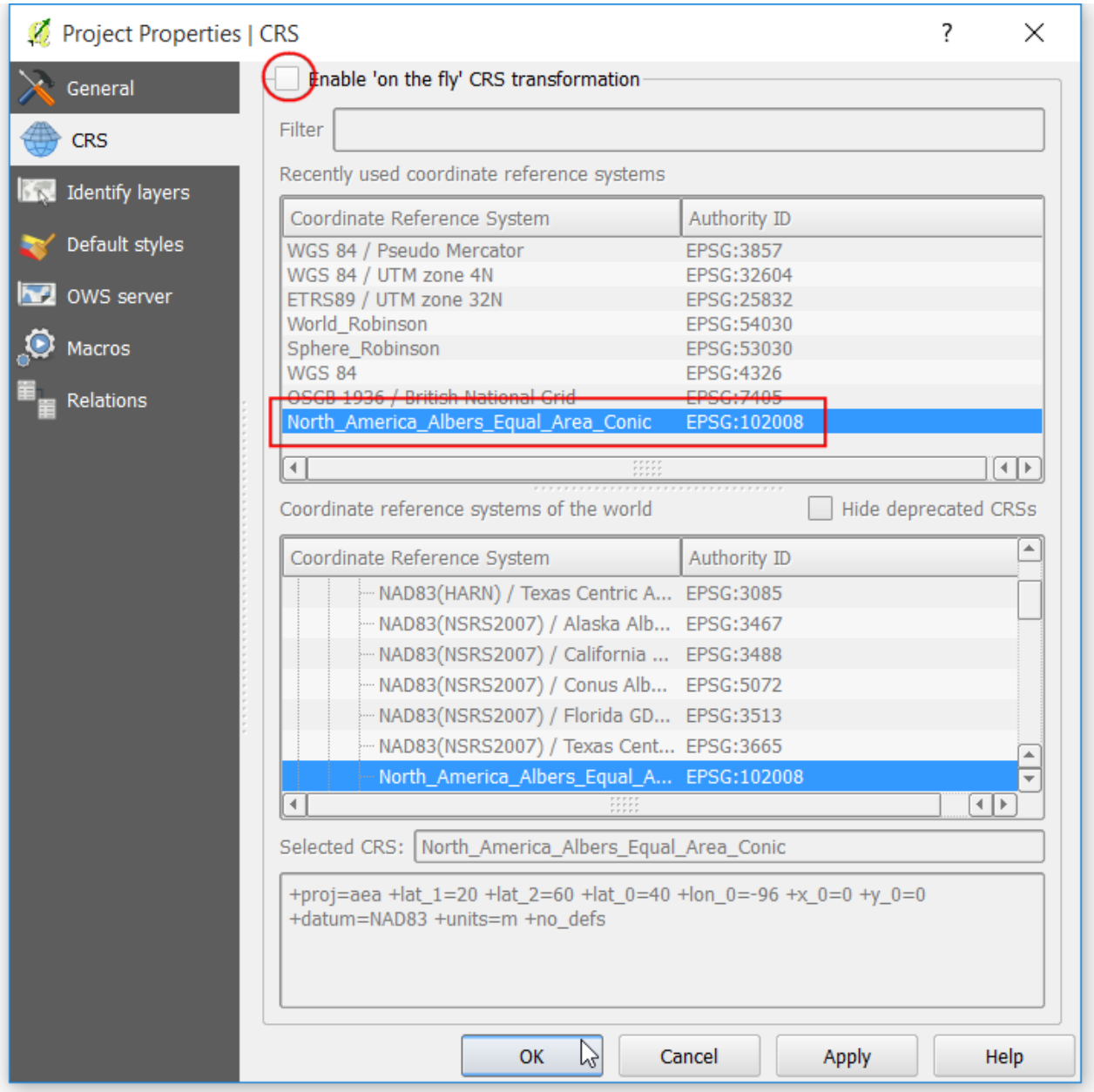
13. Switch to the CRS tab in the Options dialog. You will see that the default is Automatically enable 'on the fly' reprojection if the layers have different CRS. This means that when QGIS detects that you have loaded layers with different CRS, it will automatically re-project them back to a common CRS so they line up with each other. Click OK.



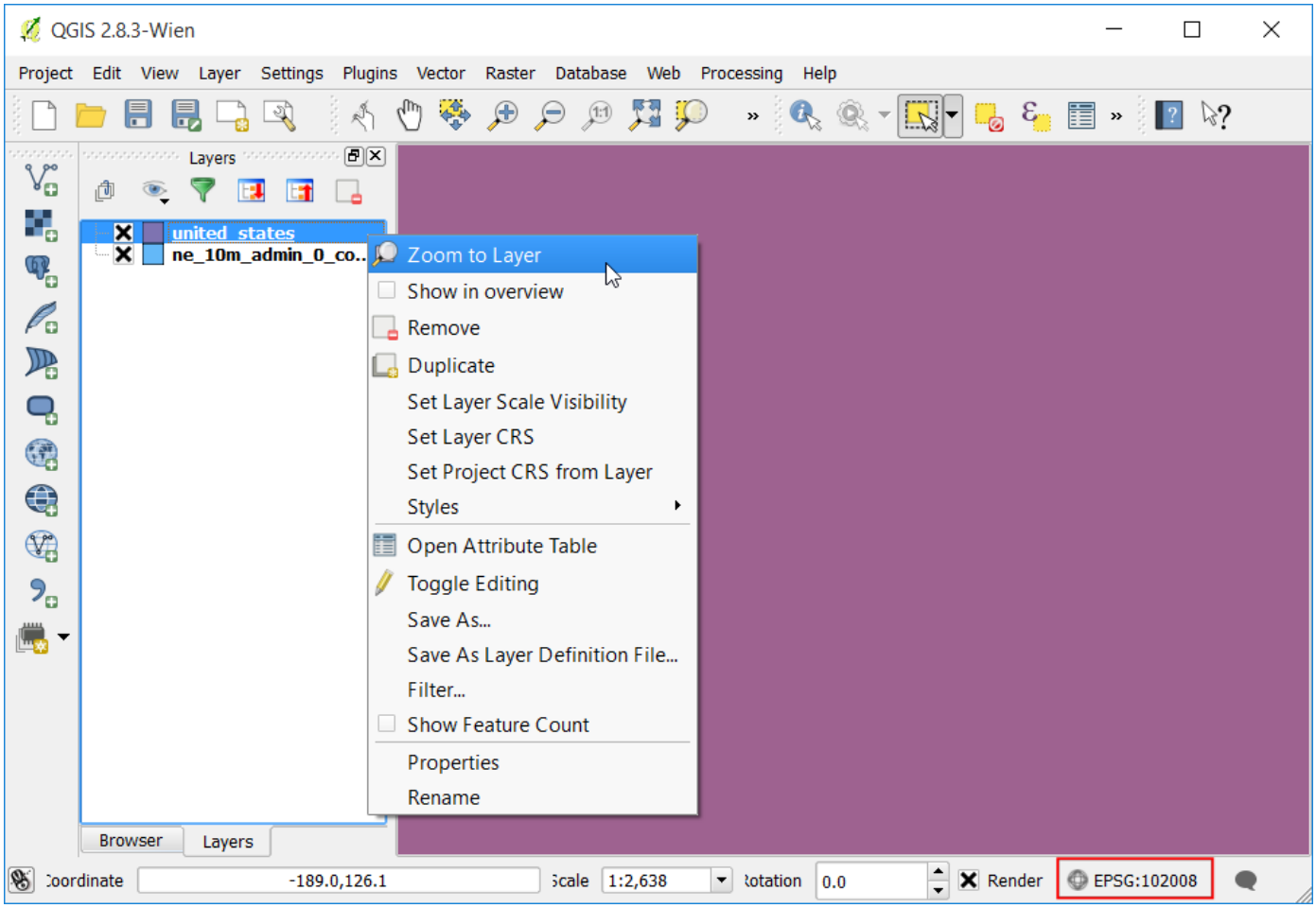
14. Let's turn-off the **On-the-fly CRS transformation** and see what happens. Click on the Current CRS text at the bottom-right corner.



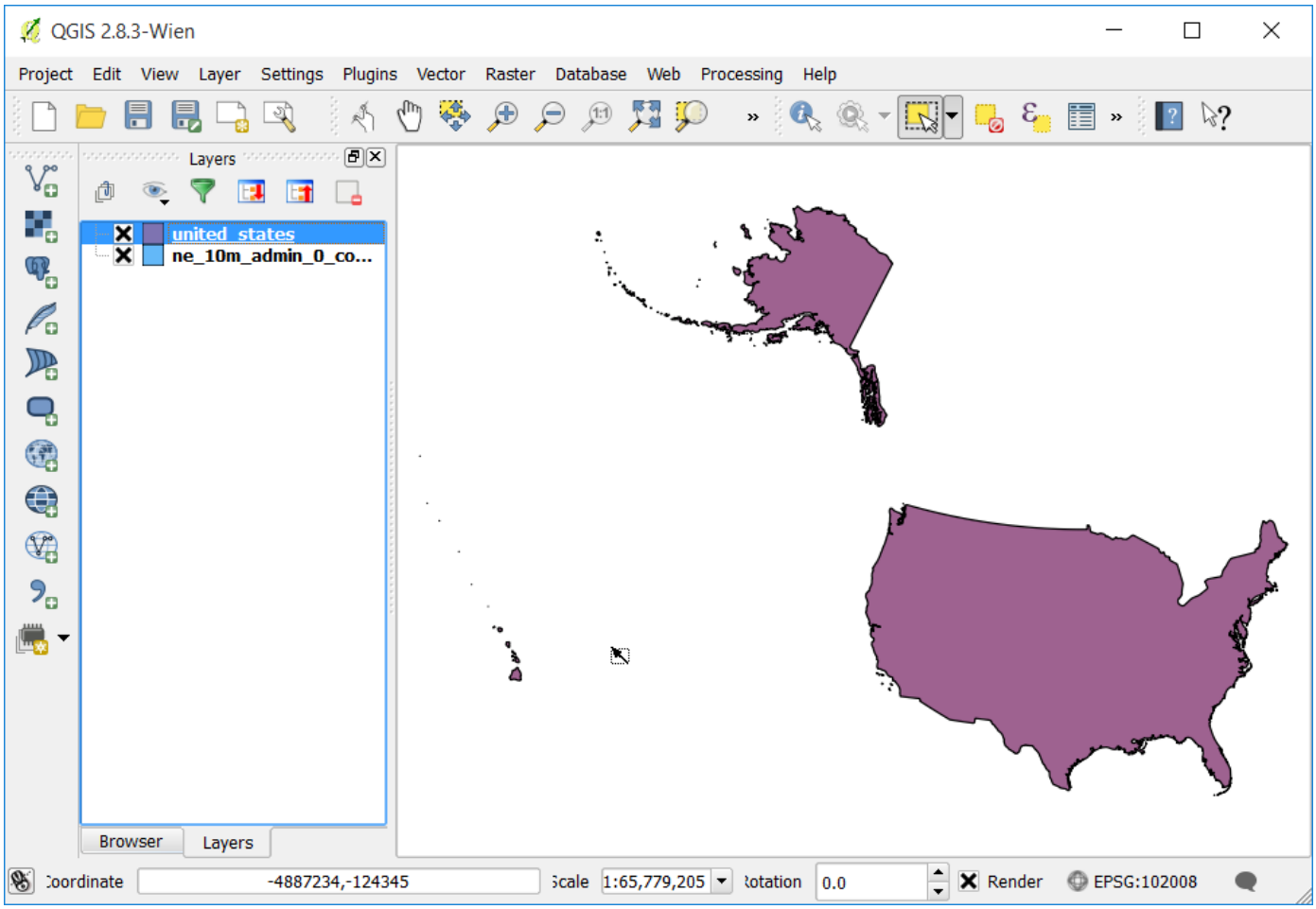
15. In the Project Properties dialog, un-check the Enable 'on the fly' CRS transformation box and click OK.



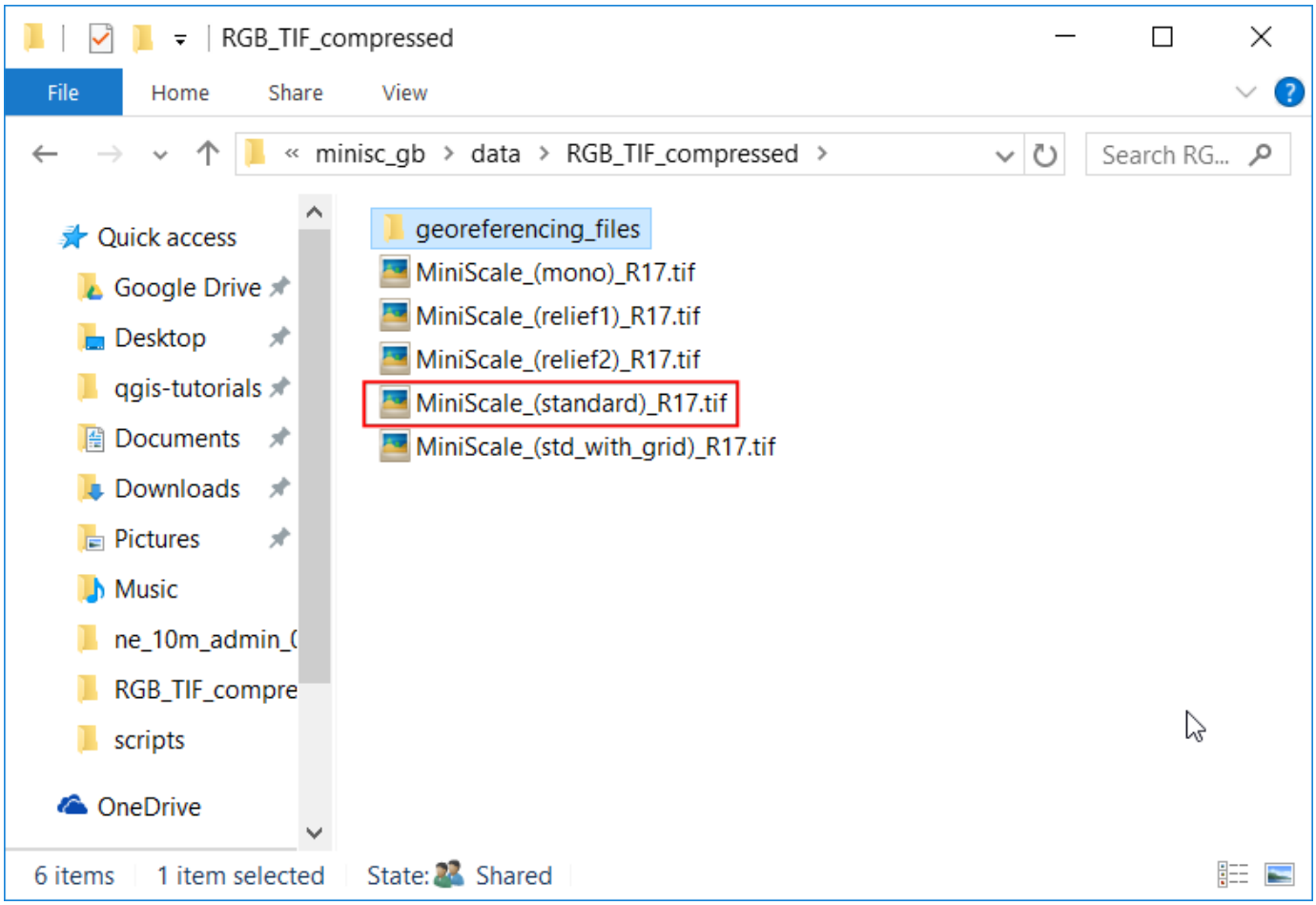
16. Back in the main QGIS window, you will see the nice world map disappear. This is because the Project CRS changed to `North_America_Albers_Equal_Area_Conic` and the coordinates and scale are different now. Right-click the `united_states` layer and select Zoom to Layer.



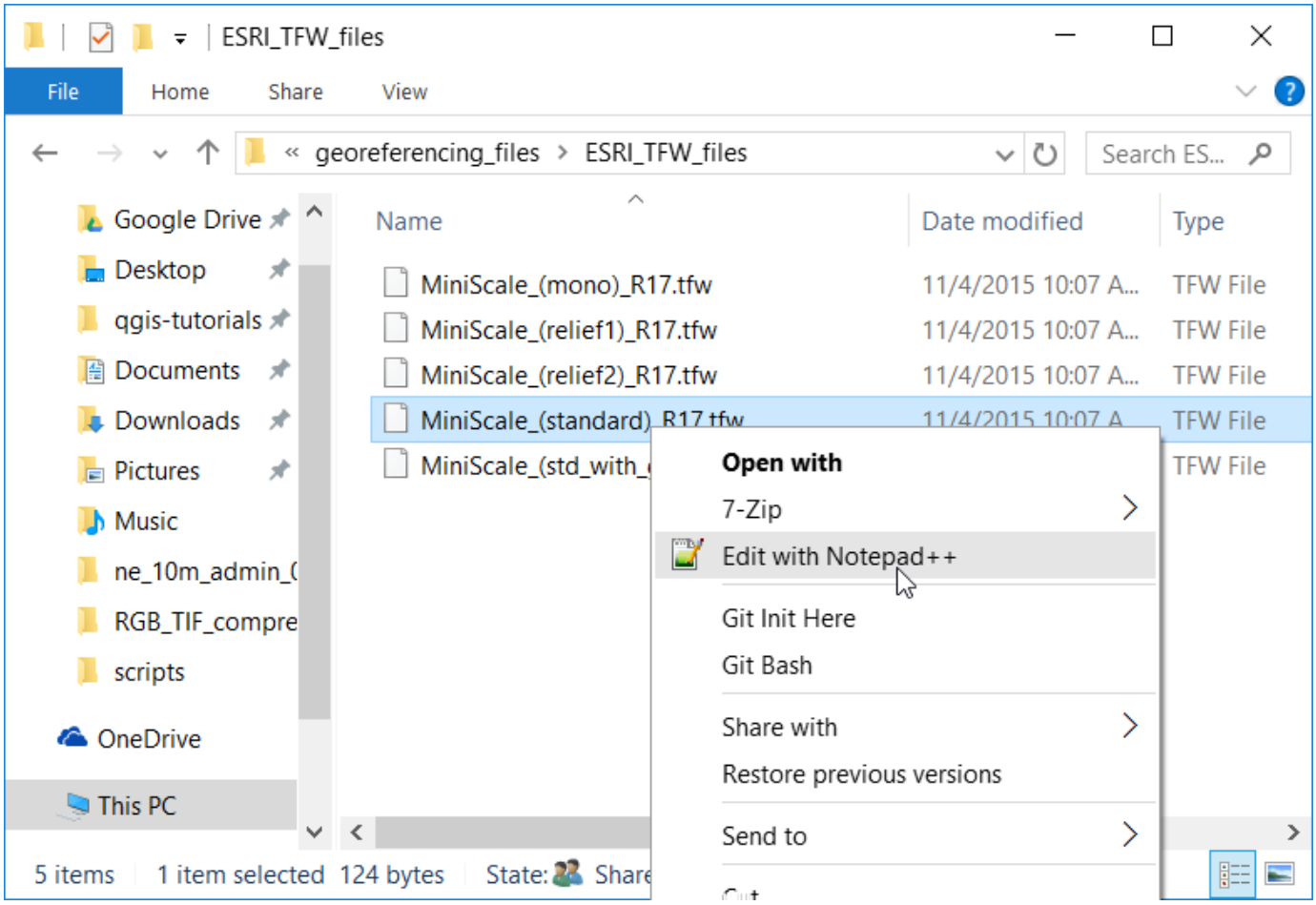
17. Now you will see the United States in the selected projection. Notice that the features from `ne_10m_admin_0_countries` do not appear on the canvas as they are in a different coordinate space than the `united_states` layer. Go back to the Project Properties dialog and turn-on the Enable 'on the fly' CRS transformation option for the remainder of the tutorial.



18. Now let's switch gears and add a raster layer to our project. Browse to the directory where you had extracted the `minisc_gb.zip` file. Locate the `RGB_TIF_COMPRESSED` folder containing tif files. You will notice that the .tif image files are plain TIF files, not GeoTIFF files. That means they do not have any projection information. To use these images in a GIS, you need to georeference them. A georeference contains 2 types of information - image extents and projection. Typically, the extents are stored in a file known as World file (https://en.wikipedia.org/wiki/World_file) and they have extensions like `.tfw` or `.jgw`. Most GIS software, including QGIS would be able to use information stored in the world files as long as they are stored in the same directory as the original image and has the same name. The `.tfw` files for the MiniScale raster files are in a separate folder named `georeferencing_files`.

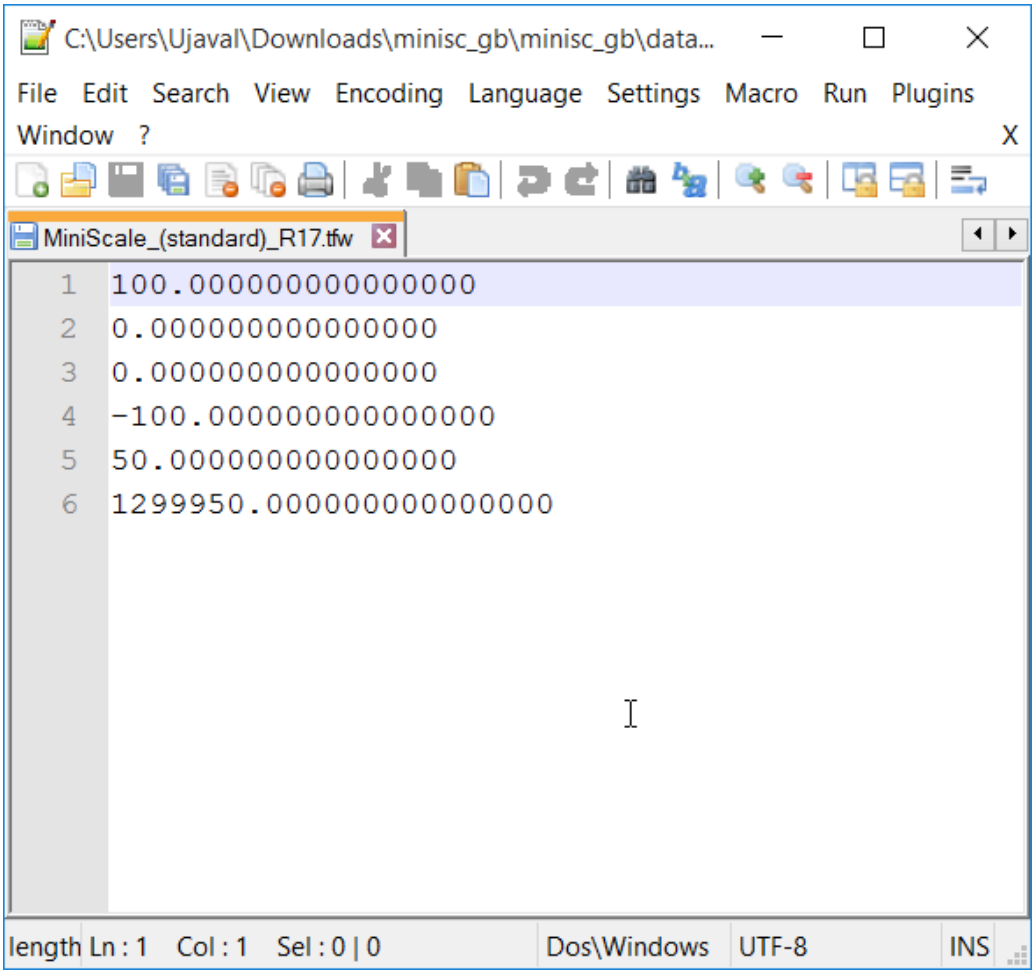


19. Go to the `ESRI_TFW_FILES` folder within `georeferencing_files`. The `.tfw` files are plain text files. Open one of the `.tfw` files in a text editor.

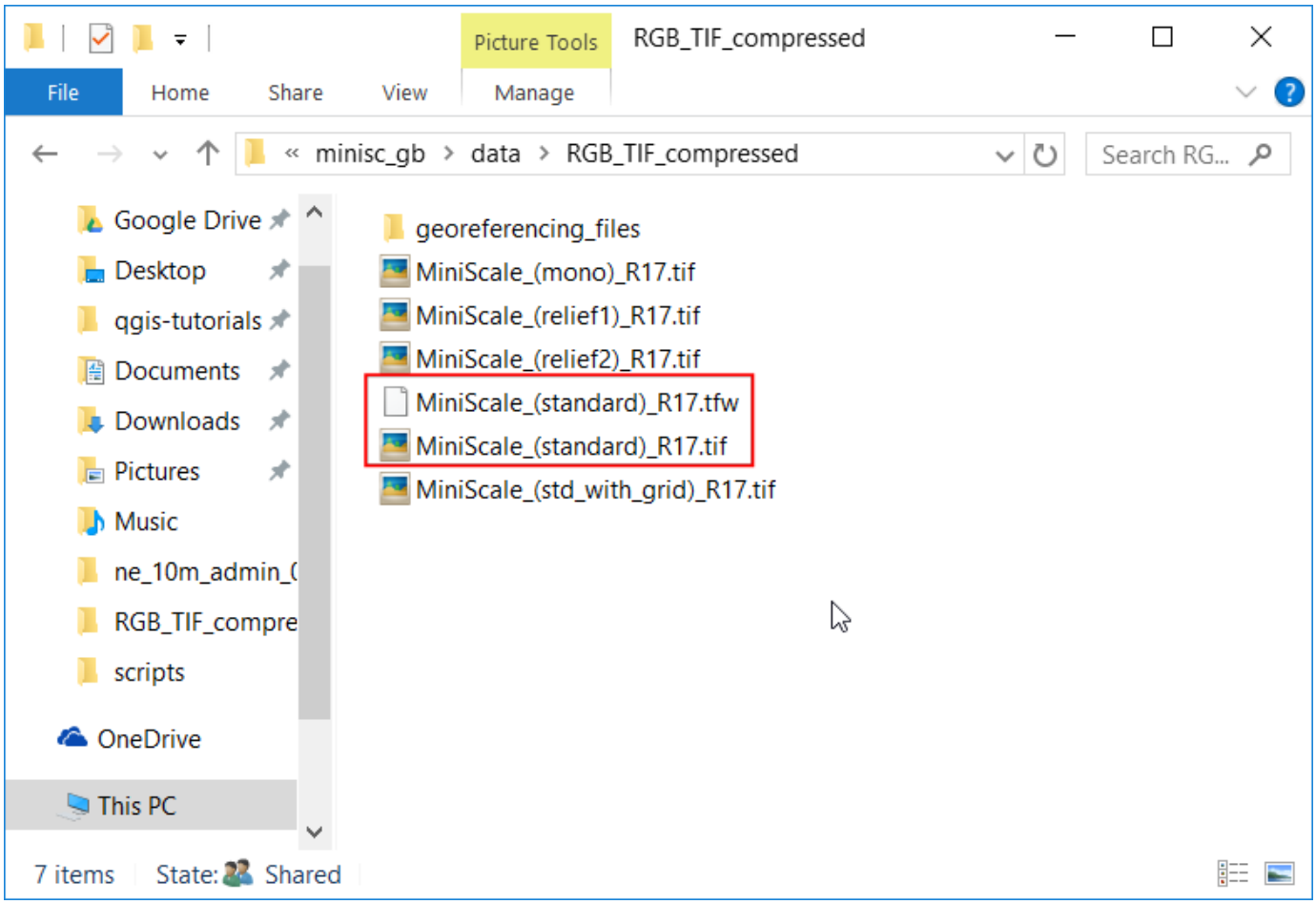


20. The world files contain 6 lines with some numbers. As explained below, each line signifies some information about the raster file. Knowing this format is useful because some data do not come with the world files and you may have to create these by hand using the supplied information.

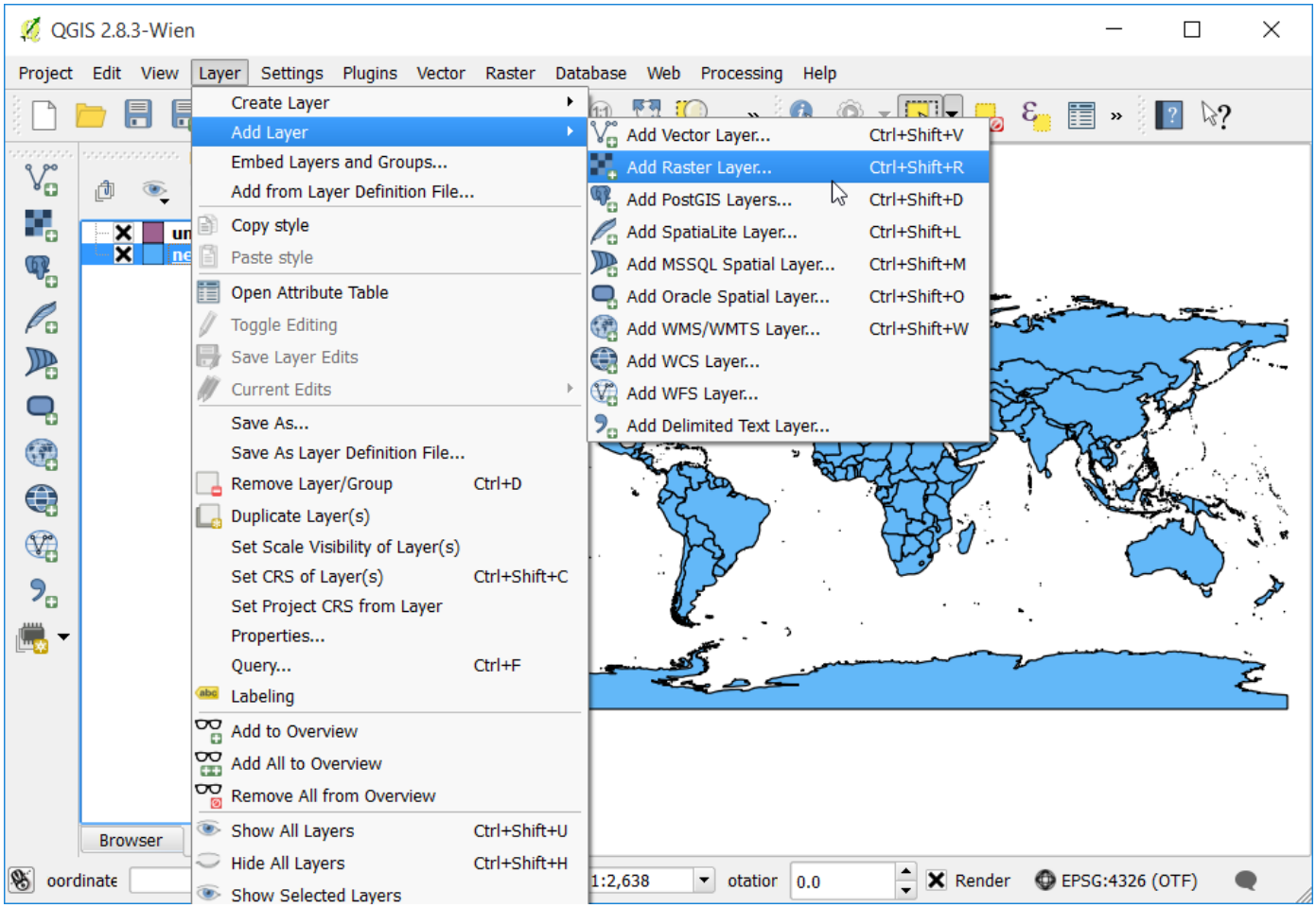
```
Line 1: A: pixel size in the x-direction in map units/pixel
Line 2: D: rotation about y-axis
Line 3: B: rotation about x-axis
Line 4: E: pixel size in the y-direction in map units
Line 5: C: x-coordinate of the center of the upper left pixel
Line 6: F: y-coordinate of the center of the upper left pixel
```



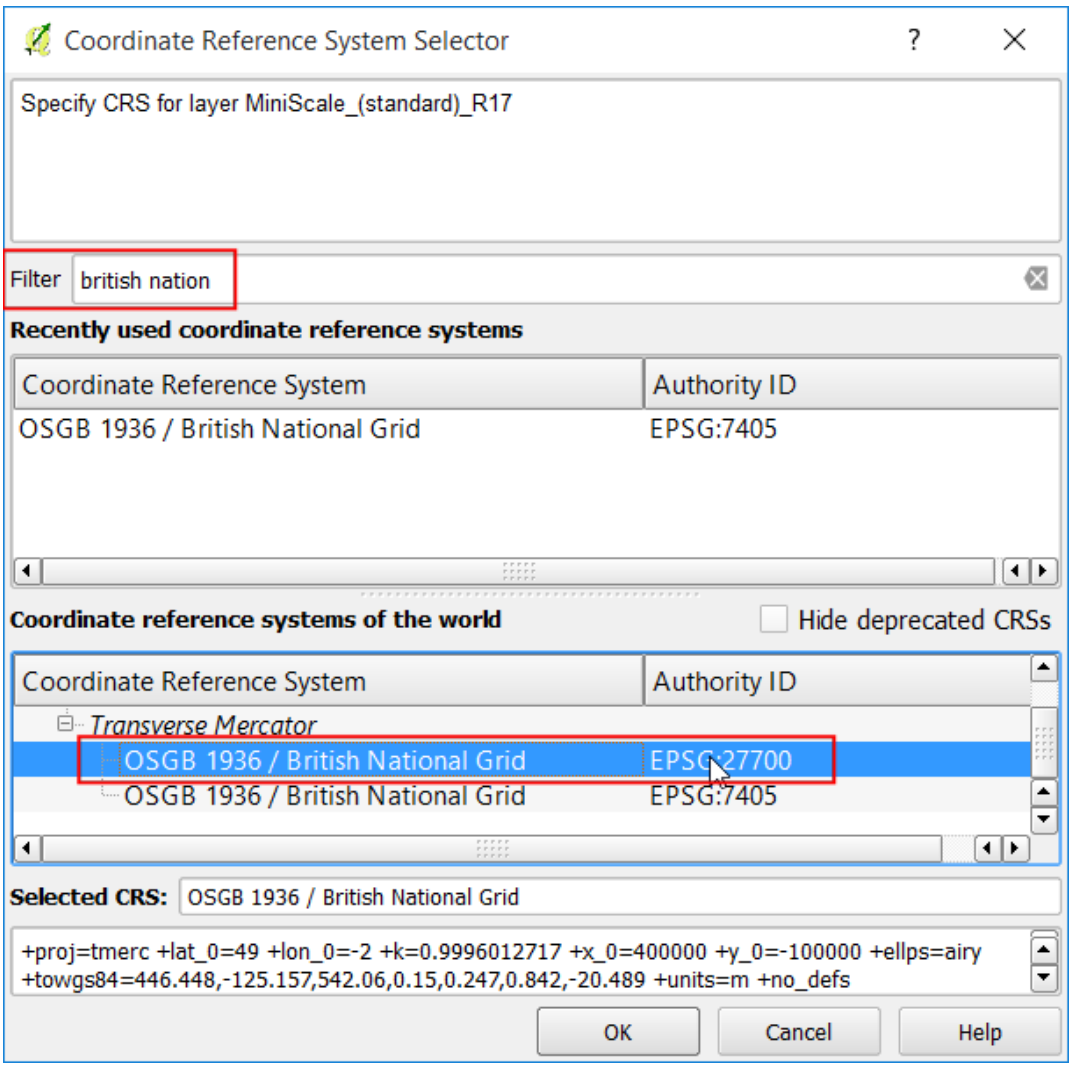
21. Copy the `MiniScale_(standard)_R17.tfw` file from the `georeferencing_files` folder to the `RGB_TIF_COMPRESSED` folder. This way the `.tfw` and the `.tif` files are in the same directory and QGIS can use the information.



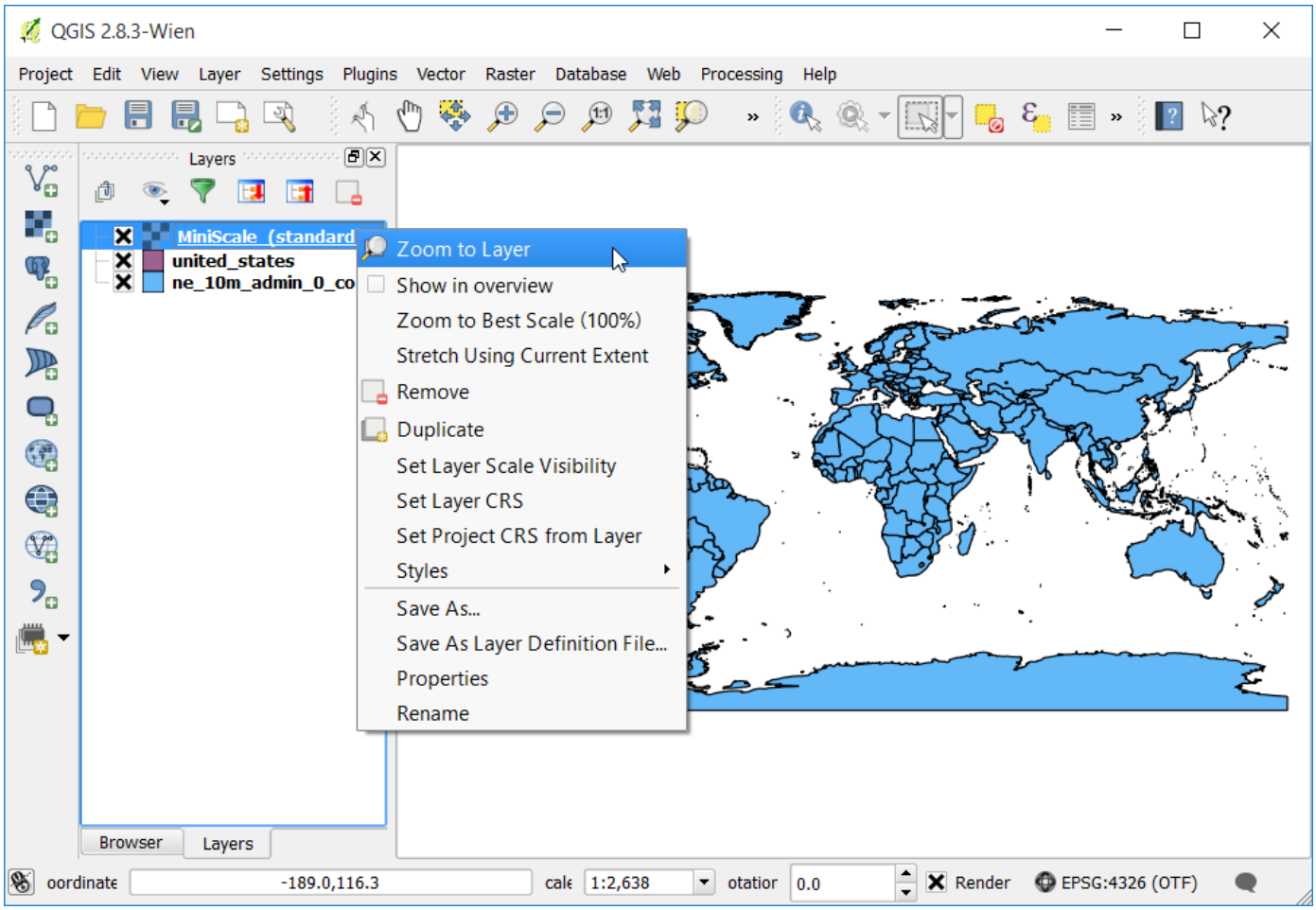
22. In the QGIS main windows, go to Layer ‣ Add Layer ‣ Add Raster Layer.... Browse to the `MiniScale_(standard)_R17.tif` file and click Open.



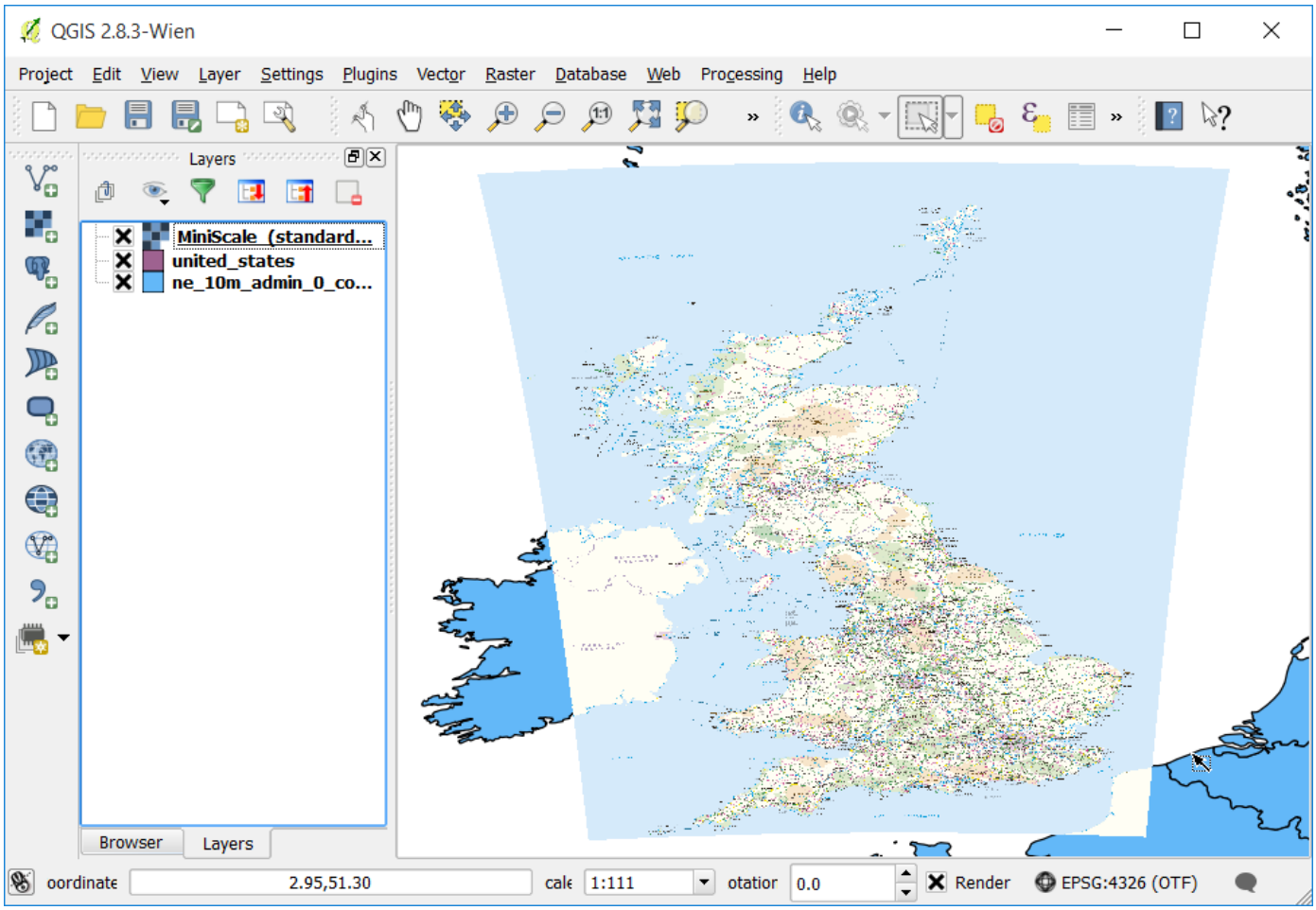
23. The Ordnance Survey files are in the British National Grid projection. In the Coordinate Reference System Selector dialog, search for `british national` and pick the `OSGB 1936 / British National Grid (EPSG:27700)` CRS. Click OK.



24. Once the `MiniScale_(standard)_R17` layer is loaded, right-click on it and select Zoom to layer.



25. You will see the raster layer overlaid on top of the `ne_10m_admin_0_countries` vector layer. Since we have the `OTF` enabled with EPSG:4326, the `MiniScale_(standard)_R17` layer gets dynamically reprojected to EPSG:4326 and shown in the same coordinate space as the other layer.



Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Georeferencing Topo Sheets and Scanned Maps (QGIS3)

Most GIS projects require georeferencing some raster data. *Georeferencing* is the process of assigning real-world coordinates to each pixel of the raster. Many times these coordinates are obtained by doing field surveys - collecting coordinates with a GPS device for few easily identifiable features in the image or map. In some cases, where you are looking to digitize scanned maps, you can obtain the coordinates from the markings on the map image itself. Using these sample coordinates or GCPs (Ground Control Points), the image is warped and made to fit within the chosen coordinate system. In this tutorial I will discuss the concepts, strategies and tools within QGIS to achieve a high accuracy georeferencing.

This tutorial is to geo-reference an image which has coordinates information available on the map image itself (i.e. grids with labels). If your source image does not have such information, you can use the method outlined in *Georeferencing Aerial Imagery (QGIS3)* (advanced_georeferencing.html)

Overview of the task

We will use a scanned map of southern India from 1870 and geo-reference it using QGIS.

Other skills you will learn

- How to determine datum and coordinate system for old maps.

Get the data

Hipkiss's Scanned Old Maps (http://www.hipkiss.org/data/maps.html) website has an excellent collection out-of-copyright scanned maps that one can use for research.

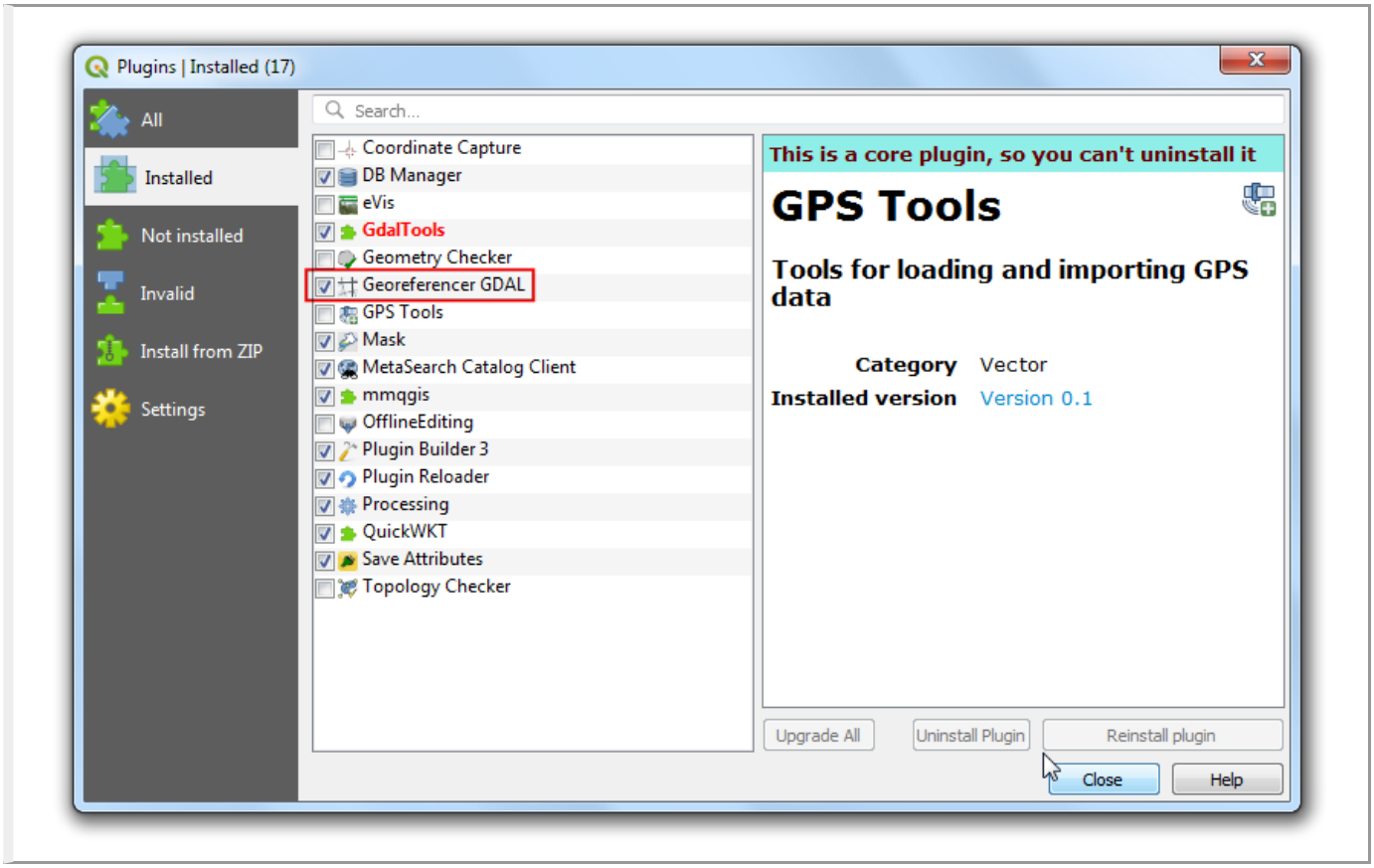
Download the 1870 map of southern India (http://www.hipkiss.org/data/maps/william-mackenzie_gallery-of-geography_1870_southern-india_3975_3071_600.jpg) and save it as a JPG image on your hard drive.

For convenience, you may directly download a copy of the dataset from the link below:

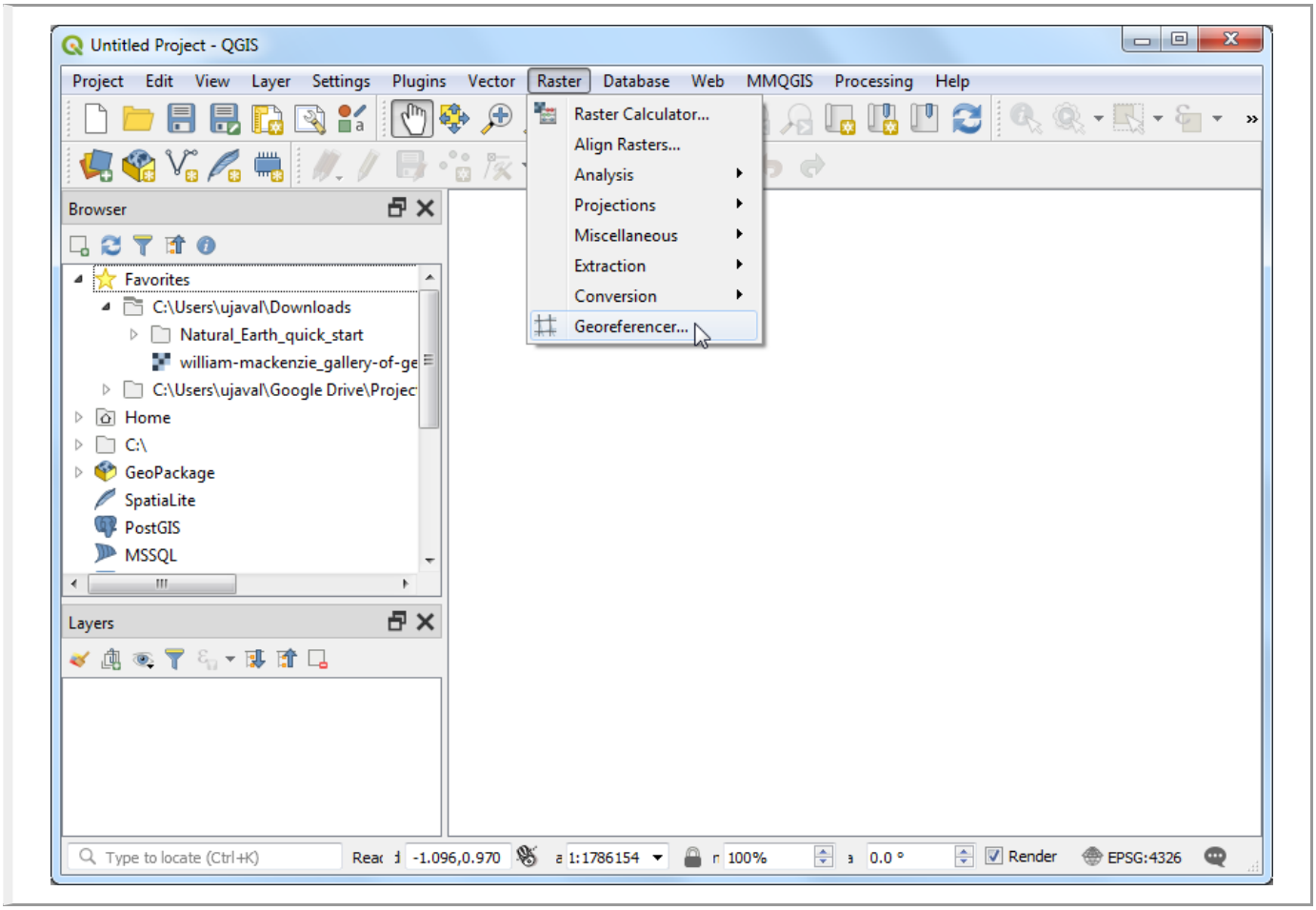
1870_southern_india.jpg (http://www.qgistutorials.com/downloads/1870_southern-india.jpg)

Procedure

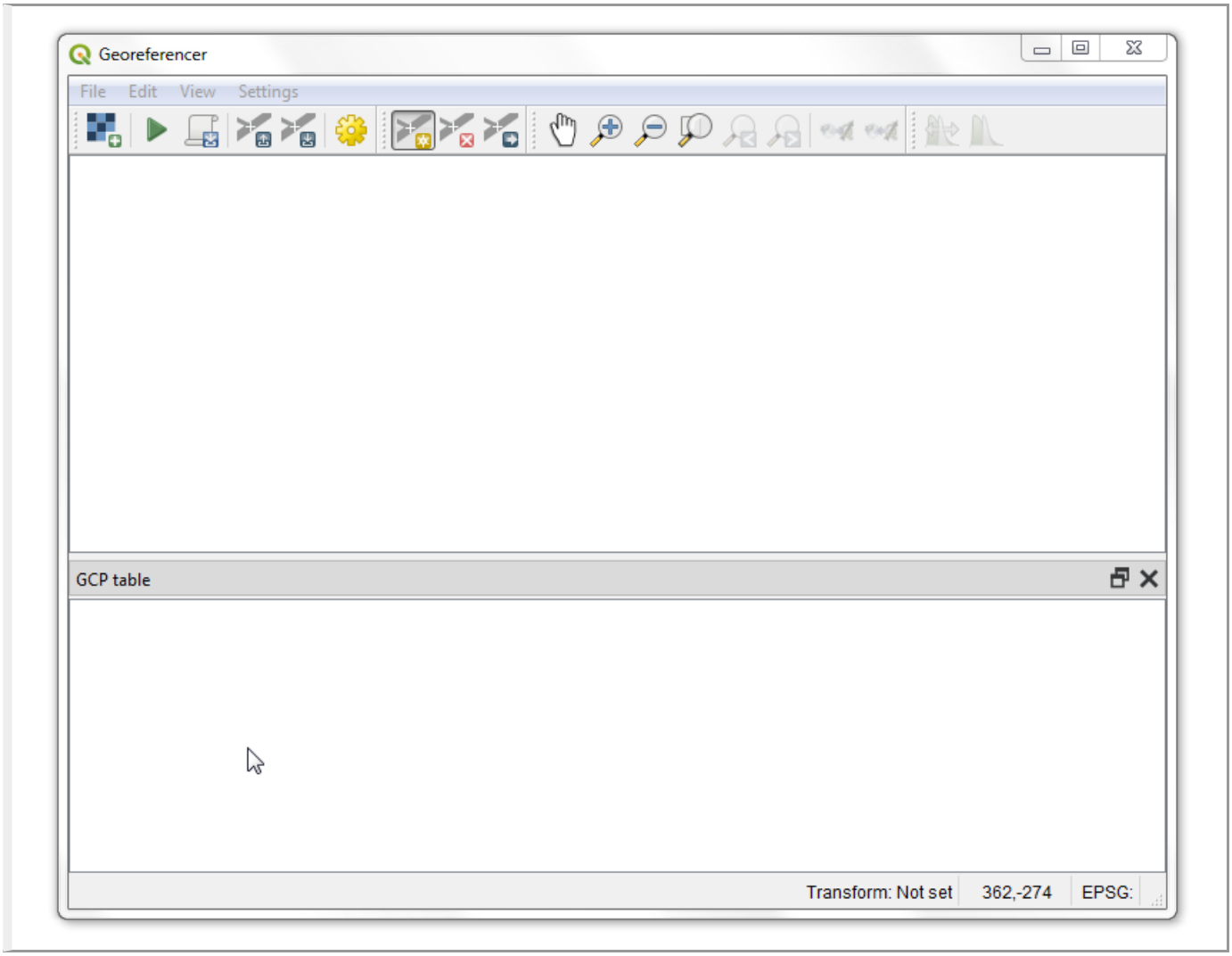
1.Georeferencing in QGIS is done via the **Georeferencer GDAL** plugin. This is a core plugin - meaning it is already part of your QGIS installation. You just need to enable it. Go to Plugins ▸ Manage and Install Plugins and enable the Georeferencer GDAL plugin in the Installed tab. See *Using Plugins* (../using_plugins.html) for more details on how to work with plugins.



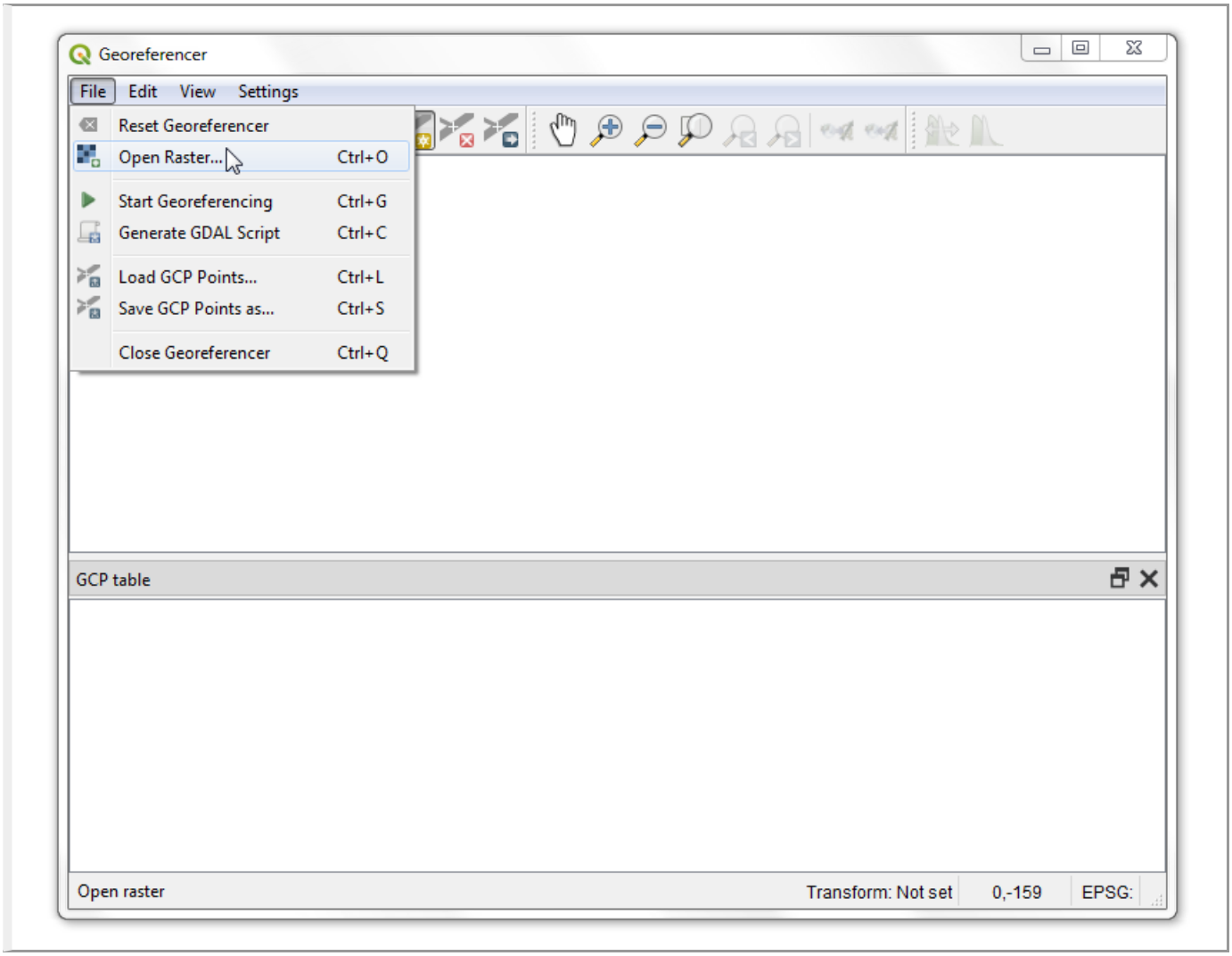
2. The plugin is installed in the Raster menu. Click on Raster ▸ Georeferencer to open the plugin.



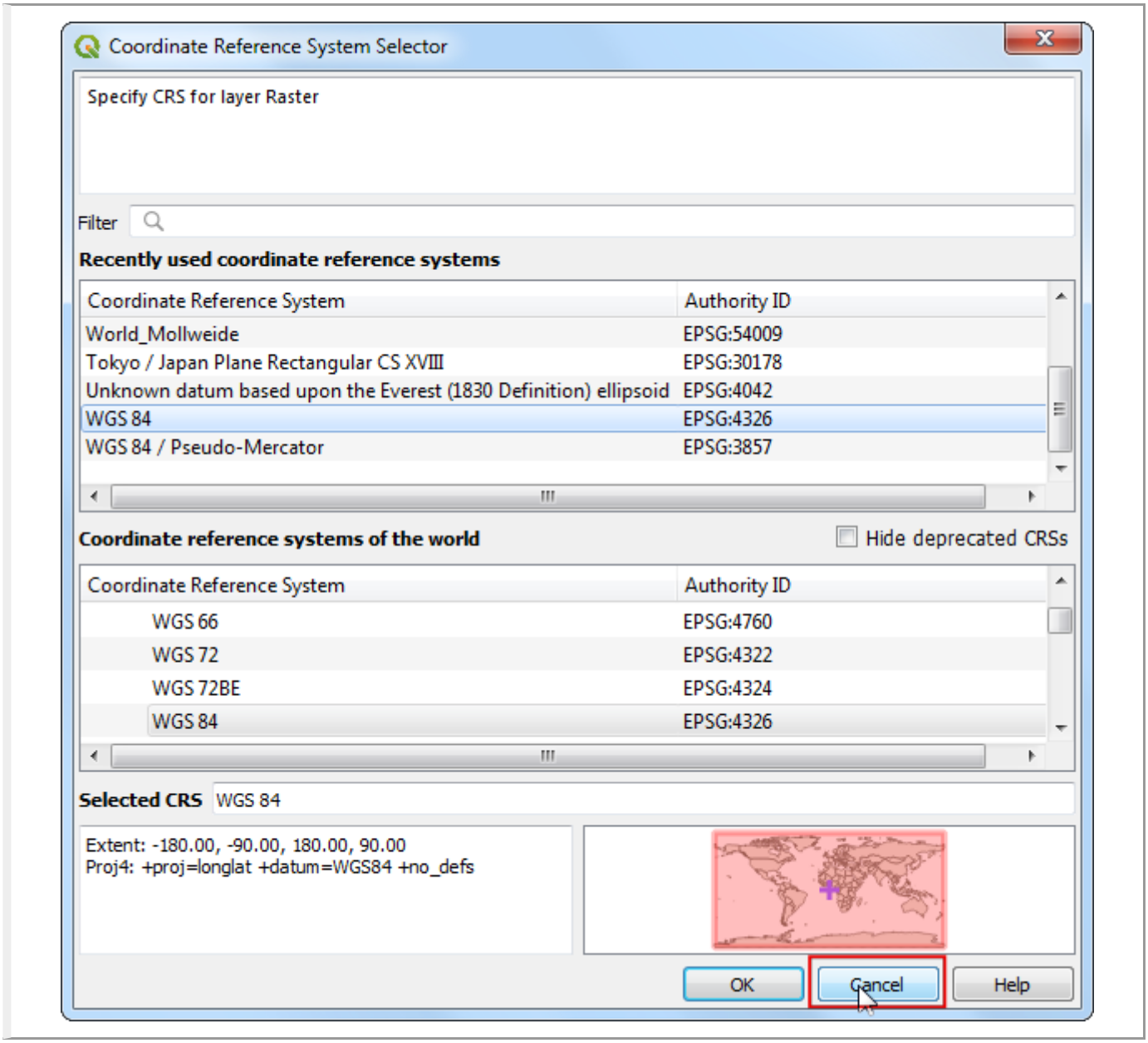
3. The plugin window is divided into 2 sections. The top section where the image will be displayed and the bottom section where a table showing your GCPs will appear.



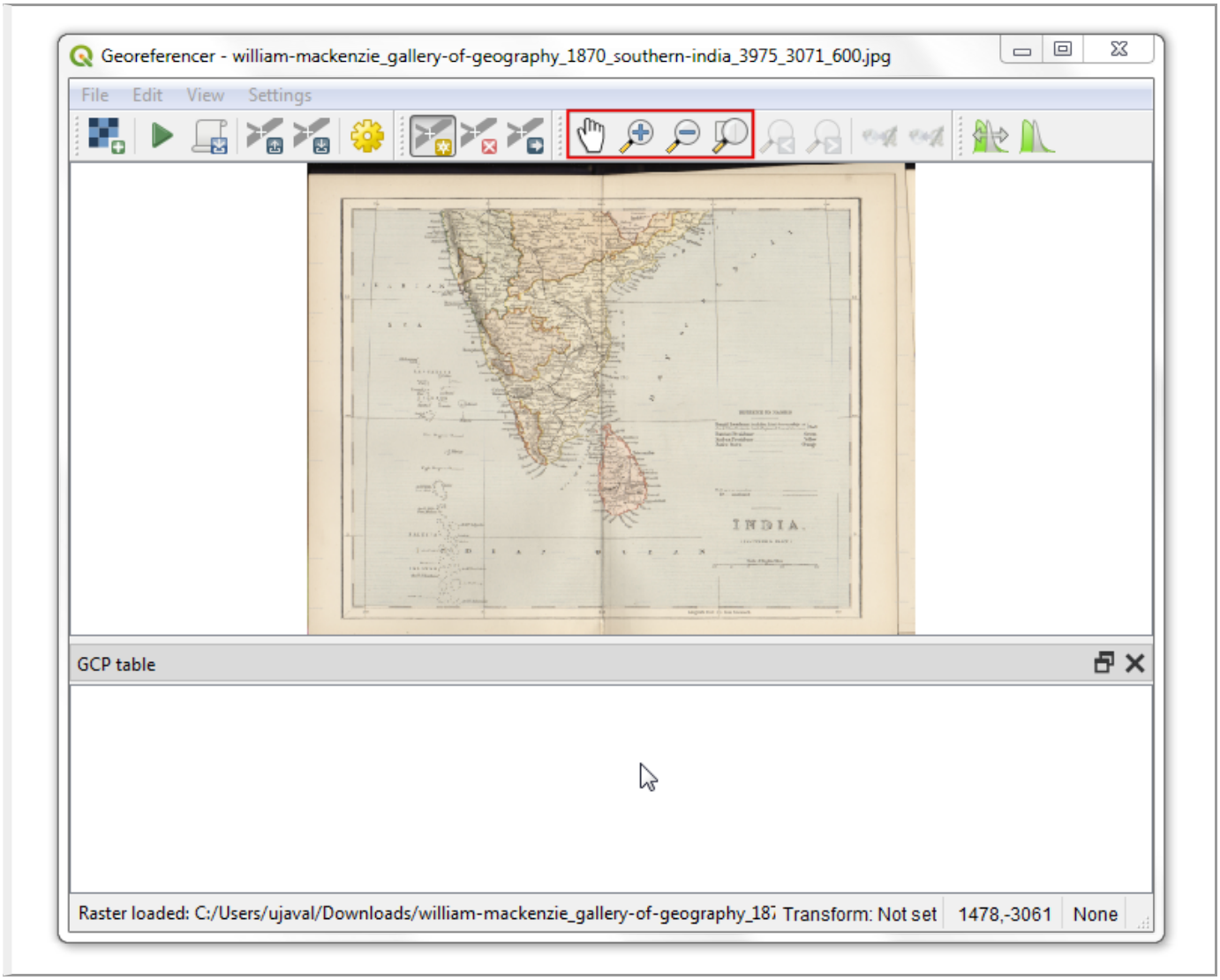
4. Now we will open our JPG image. Go to File ‣ Open Raster. Browse to the downloaded image of the scanned map and click Open.



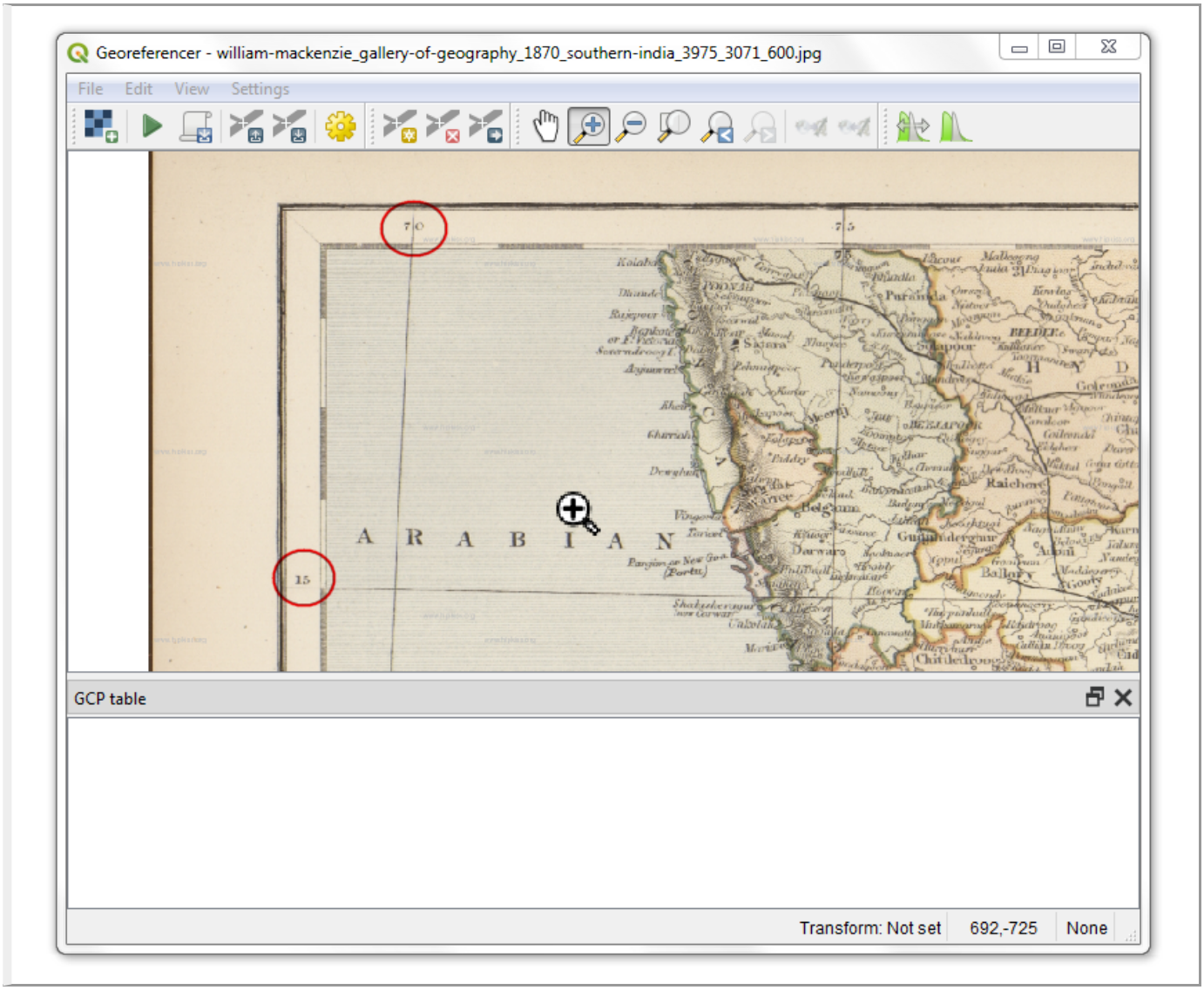
5. In the next screen, you will asked to choose the raster's coordinate reference system (CRS). Our source image is a plain JPEG file and doesn't have any coordinate reference system atached to it, so you can click Cancel.



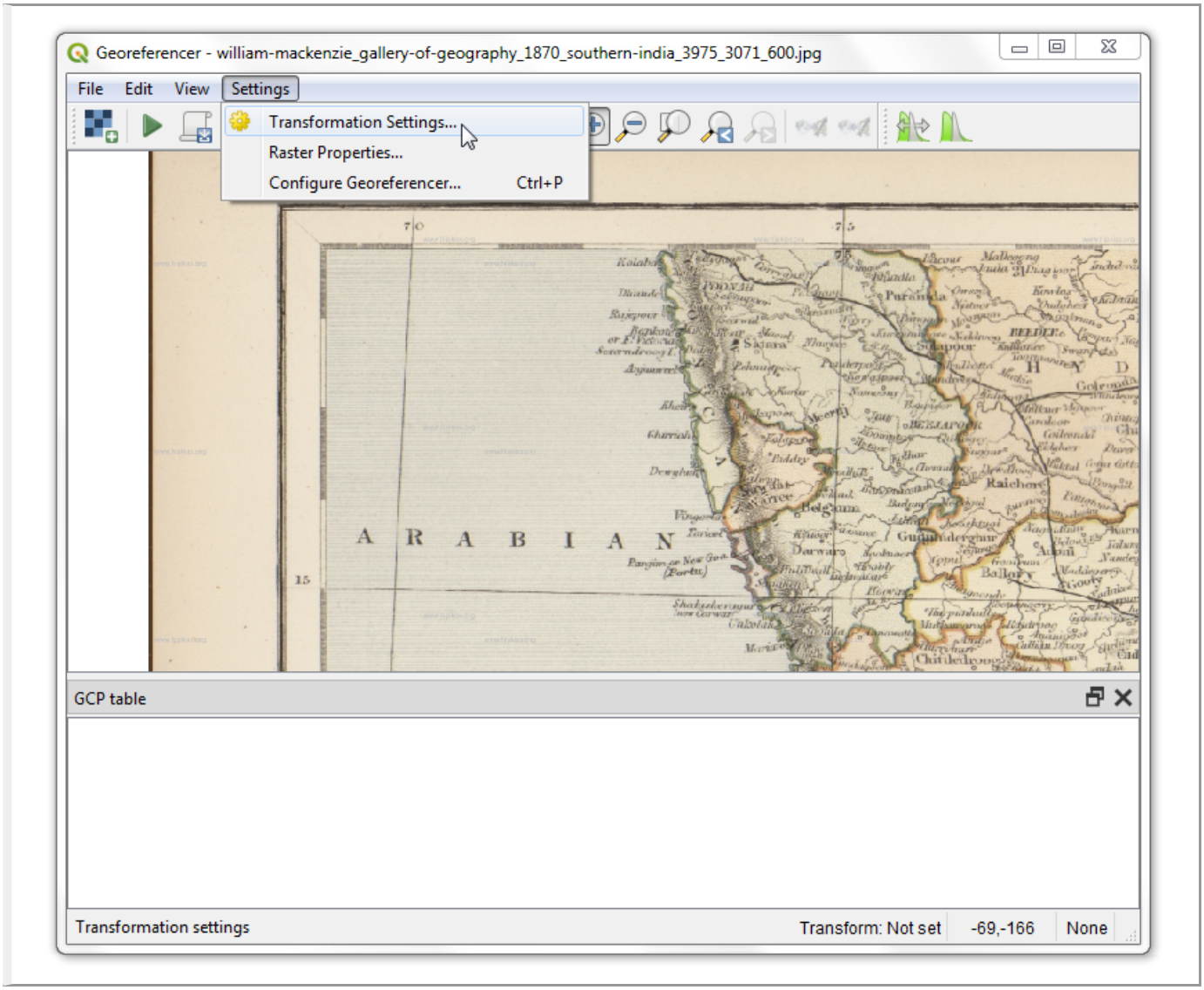
6. You will see the image will be loaded on the top section. You can use the zoom/pan controls in the toolbar to learn more about the map.



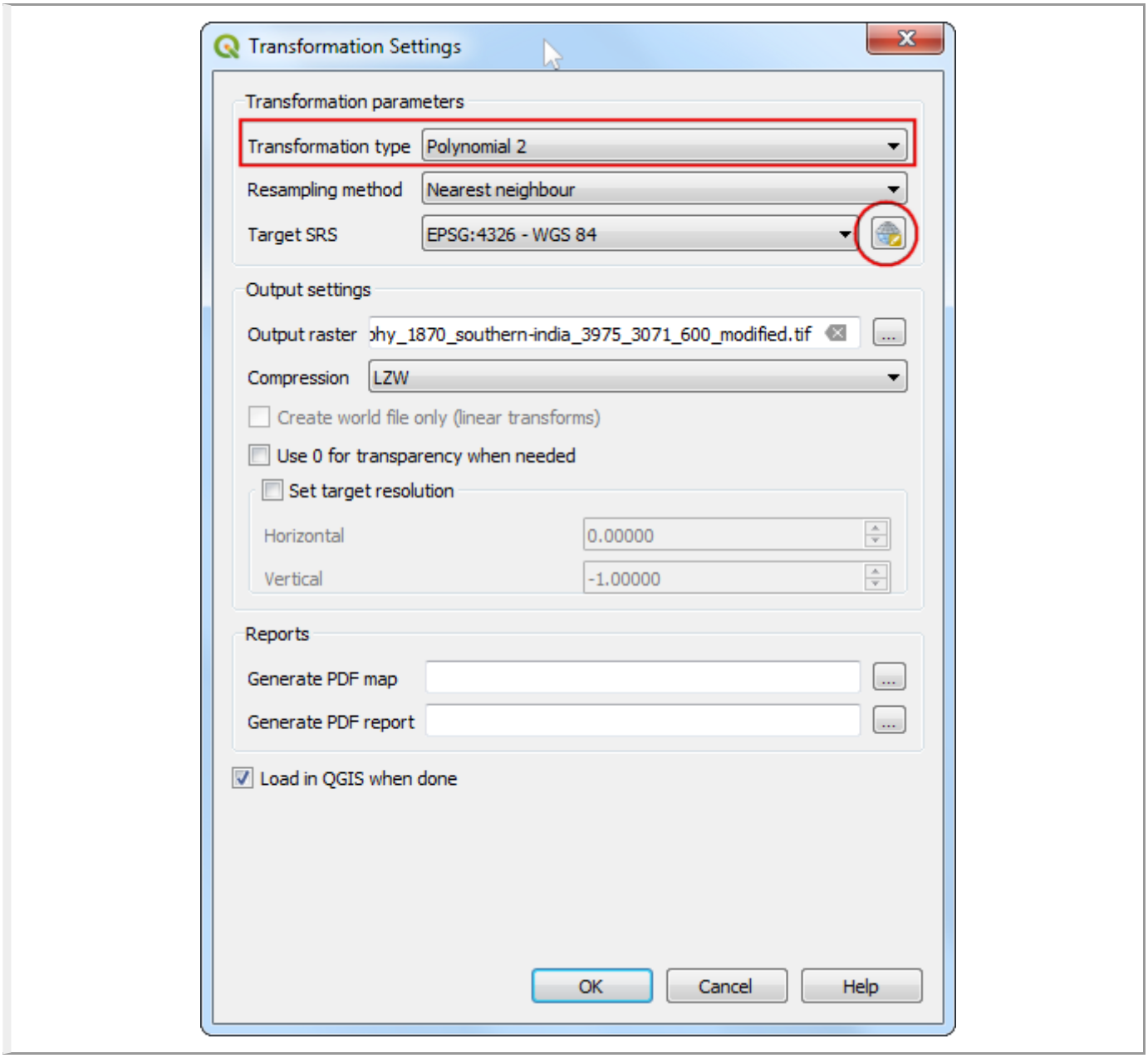
7. Now we need to assign coordinates to some points on this map. If you look closely, you will see coordinate grid with markings. These are Latitude and Longitude grid lines.



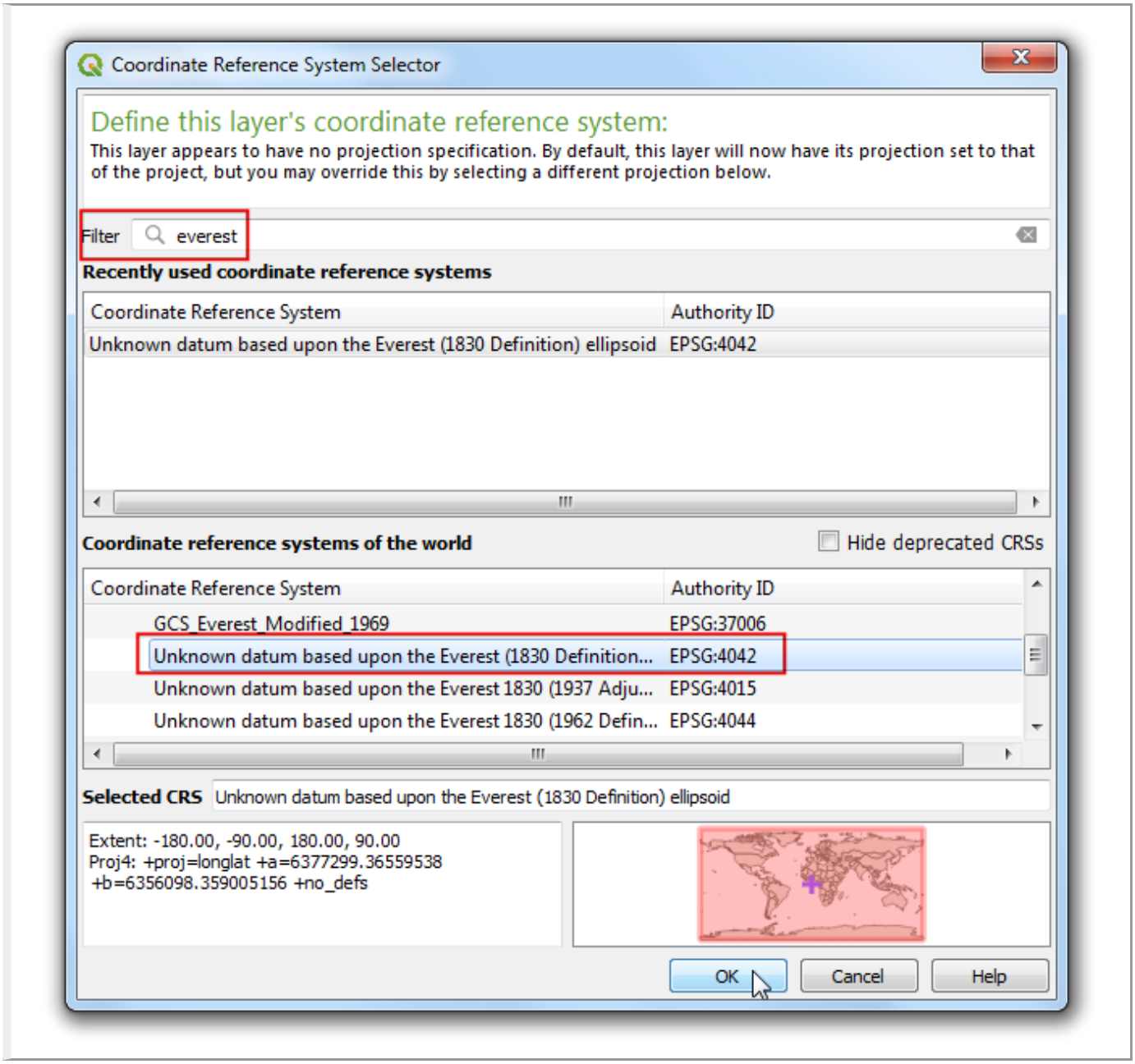
8. Before we start adding Ground Control Points (GCP), we need to define the Transformation Settings. Go to Settings ‣ Transformation settings.



9. In the Transformation settings dialog, choose the Transformation type as `Polynomial 2`. See QGIS Documentation (https://docs.qgis.org/testing/en/docs/user_manual/plugins/plugins_georeferencer.html#available-transformation-algorithms) to learn about different transofrmation types and their uses. Click Select CRS button next to Target SRS.



10. If you are geo-referencing a scanned map like this, you can obtain the CRS information from the map itself. Looking at our map image, the coordinates are in Latitude/Longitude. There is no datum information given, so we have to assume an appropriate one. Since it is India and the map is quite old, we can bet the Everest 1830 datum would give us good results. Search for `everest` and select the CRS with oldest definition of the Everest datum (EPSG:4042). Click OK.



Note

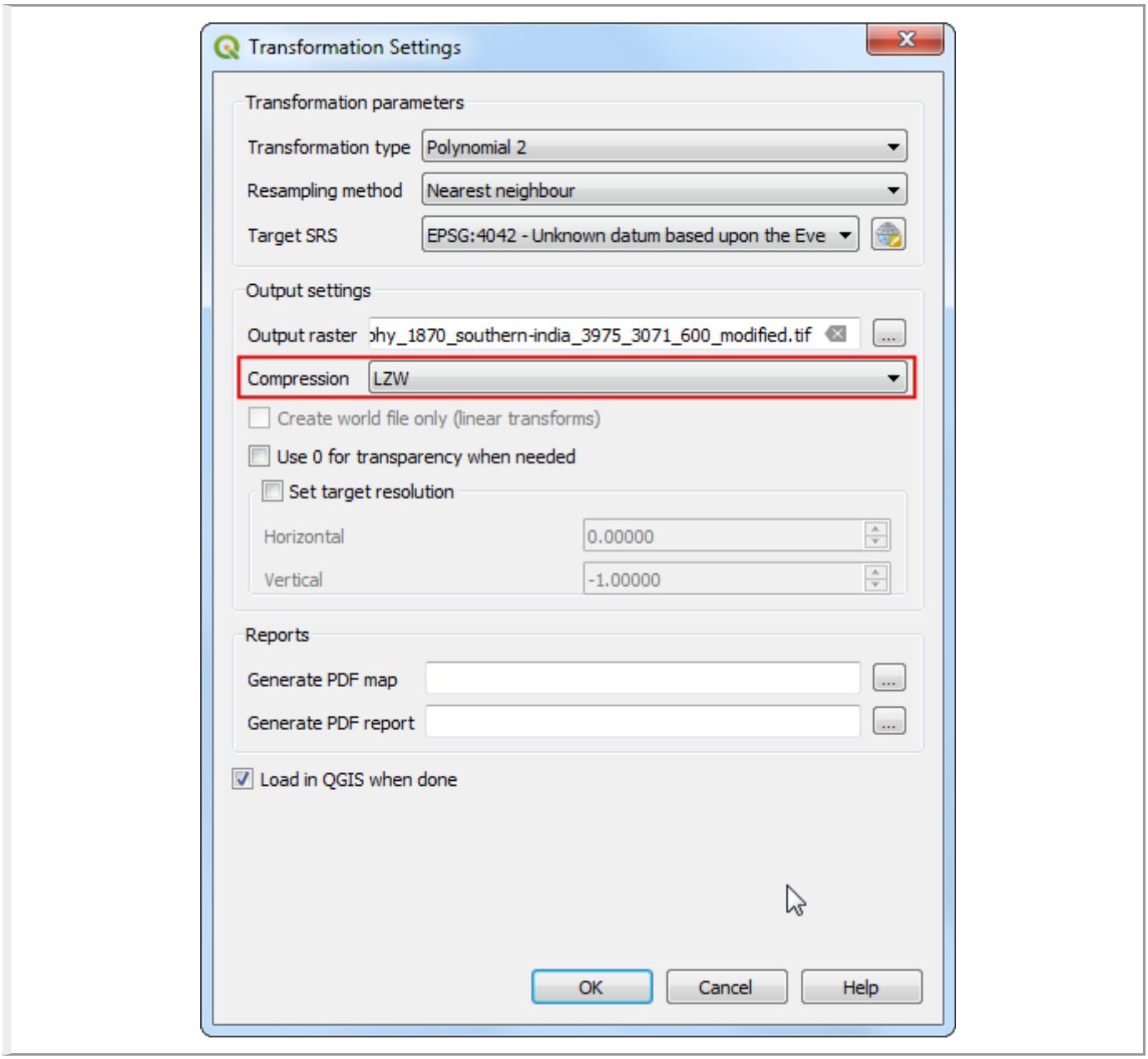
Survey of India Topo Sheets created between the year 1960 and 2000 use the Everest 1956 spheroid and India_nepal datum. If you are georeferencing SOI Topo Sheets, , you can define a Custom CRS in QGIS with the following paramters and use it in this step. This definition includes a delta_x, delta_y and delta_z parameters for transforming this datum to WGS84. See this page for more information on the Indian Grid System (https://deeppradhan.heliohost.org/gis/indian-grid/).

```
+proj=longlat +a=6377301.243 +b=6356100.2284 +towgs84=295,736,257,0,0,0,0 +no_defs
```

Note

Most maps are created using a Projected CRS. If the map you are trying to georeference uses a projected CRS that you know of, but the graticules labels are in a Geographic CRS (latitude/longitude), you may use an alternate workflow to minimize distortion. Instead of using a Geographic CRS like we are using here, you can create a vector grid in QGIS and transform it to the projected CRS to be used as a reference for accurate coordinate capture. See this page (https://raisedbeaches.net/2018/02/01/georeferencing-in-qgis/) for more details.

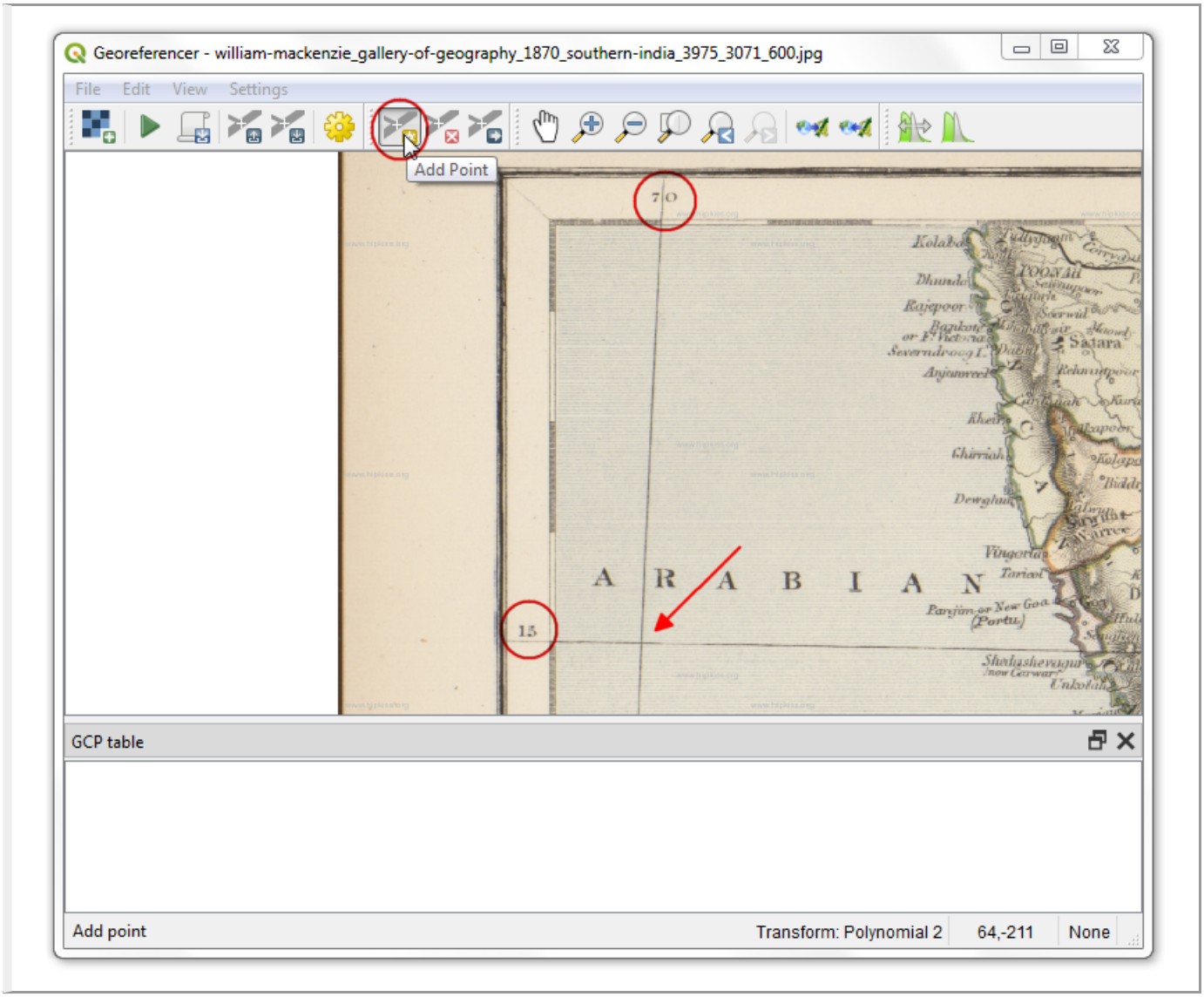
11. Name your output raster as `1870_southern_india_modified.tif`. Choose `LZW` as the Compression. Make sure the Load in QGIS when done option is checked. CLick OK.



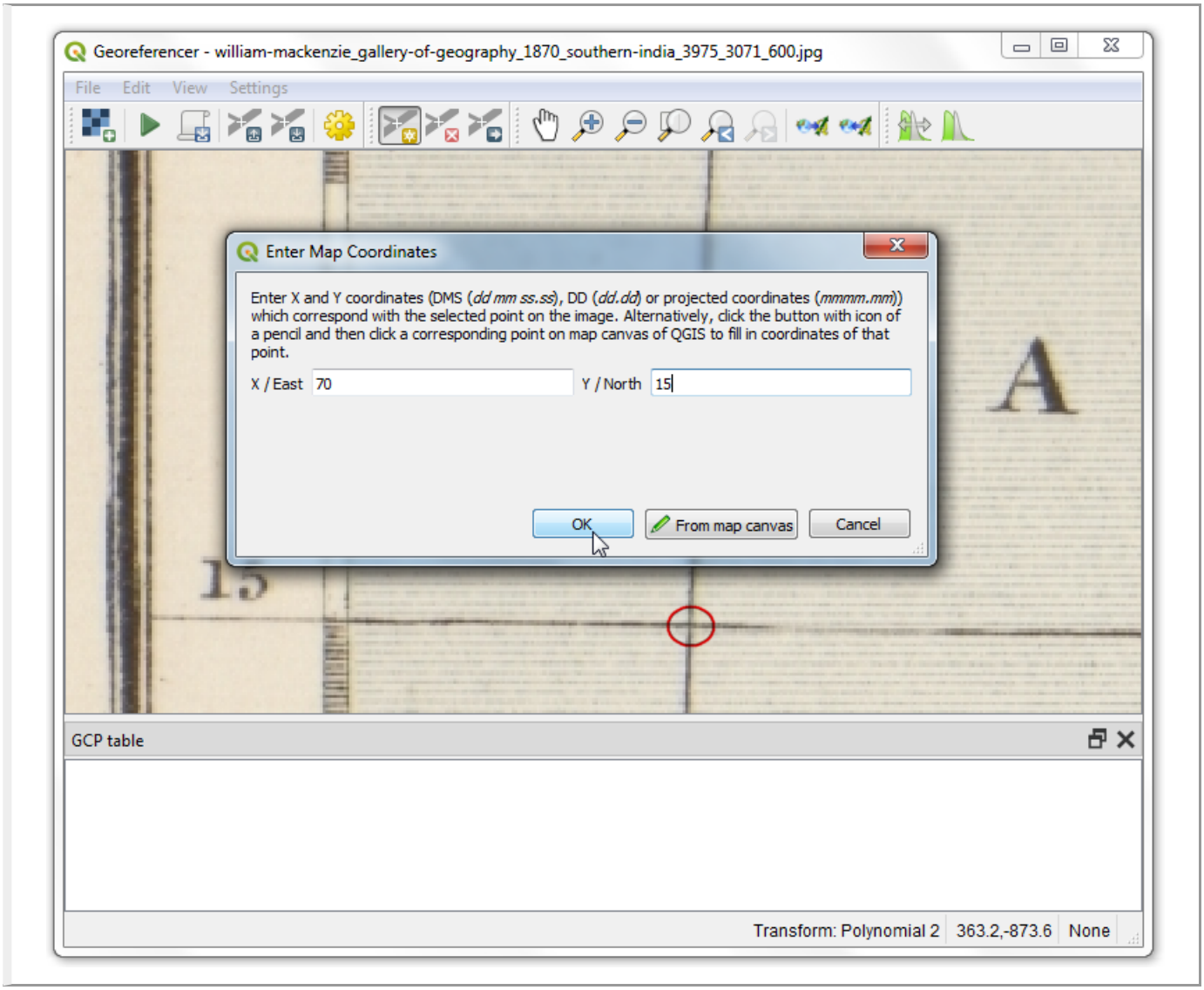
Note

Uncompressed GeoTIFF files can be very large in size. So compressing them is always a good idea. You can learn more about different TIFF compression options (LZW, PACKBITS or DEFLATE) in this article (https://www.accusoft.com/faqs/differences-compressions-used-tiff-files/).

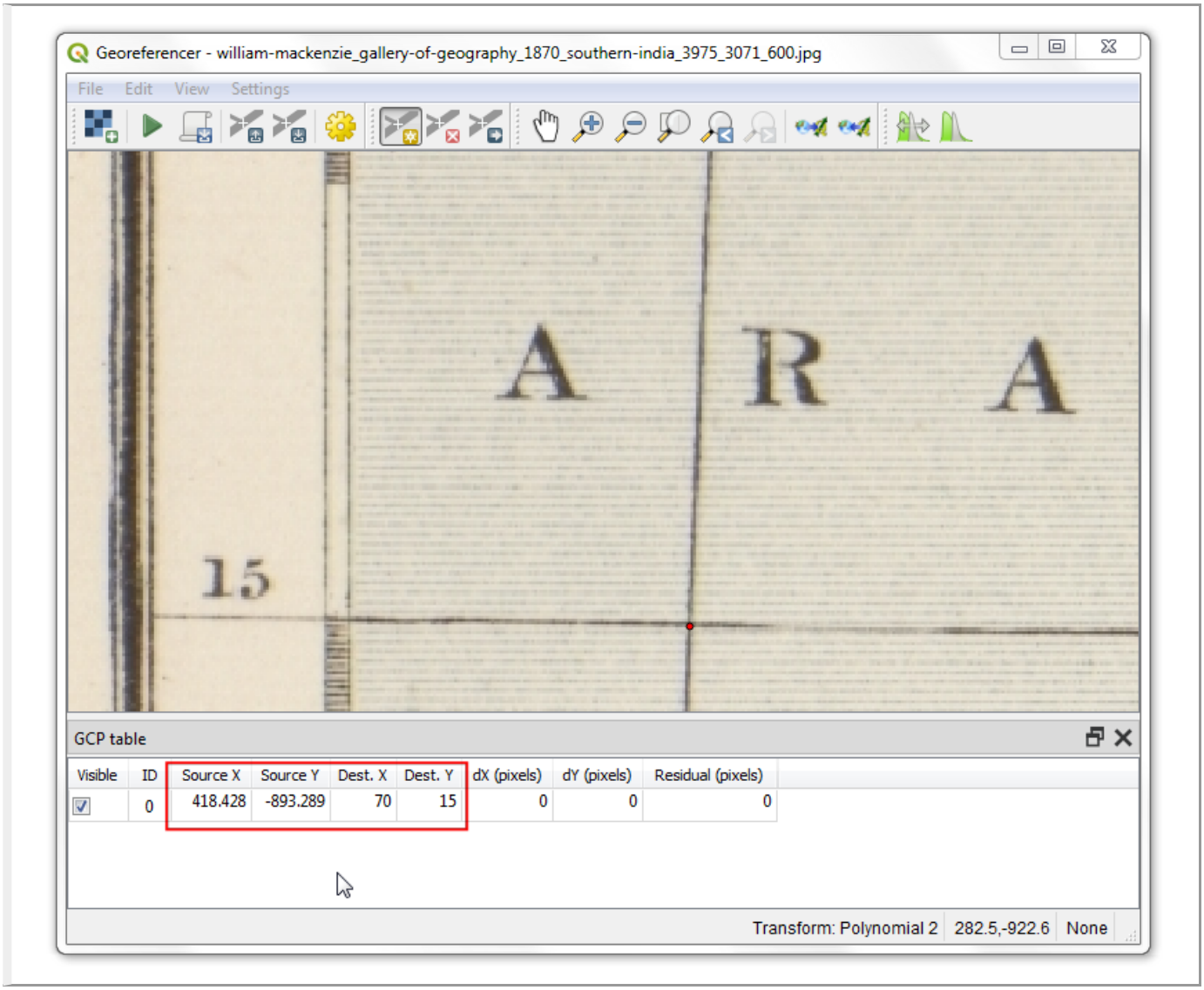
12. Now we can start adding the Ground Control Points (GCP). The intersections of the grid lines will serve as the *ground-truth* in our case. As the grid lines are labeled, we can determine the X and Y coordinates of the points using them. Click Add Point.



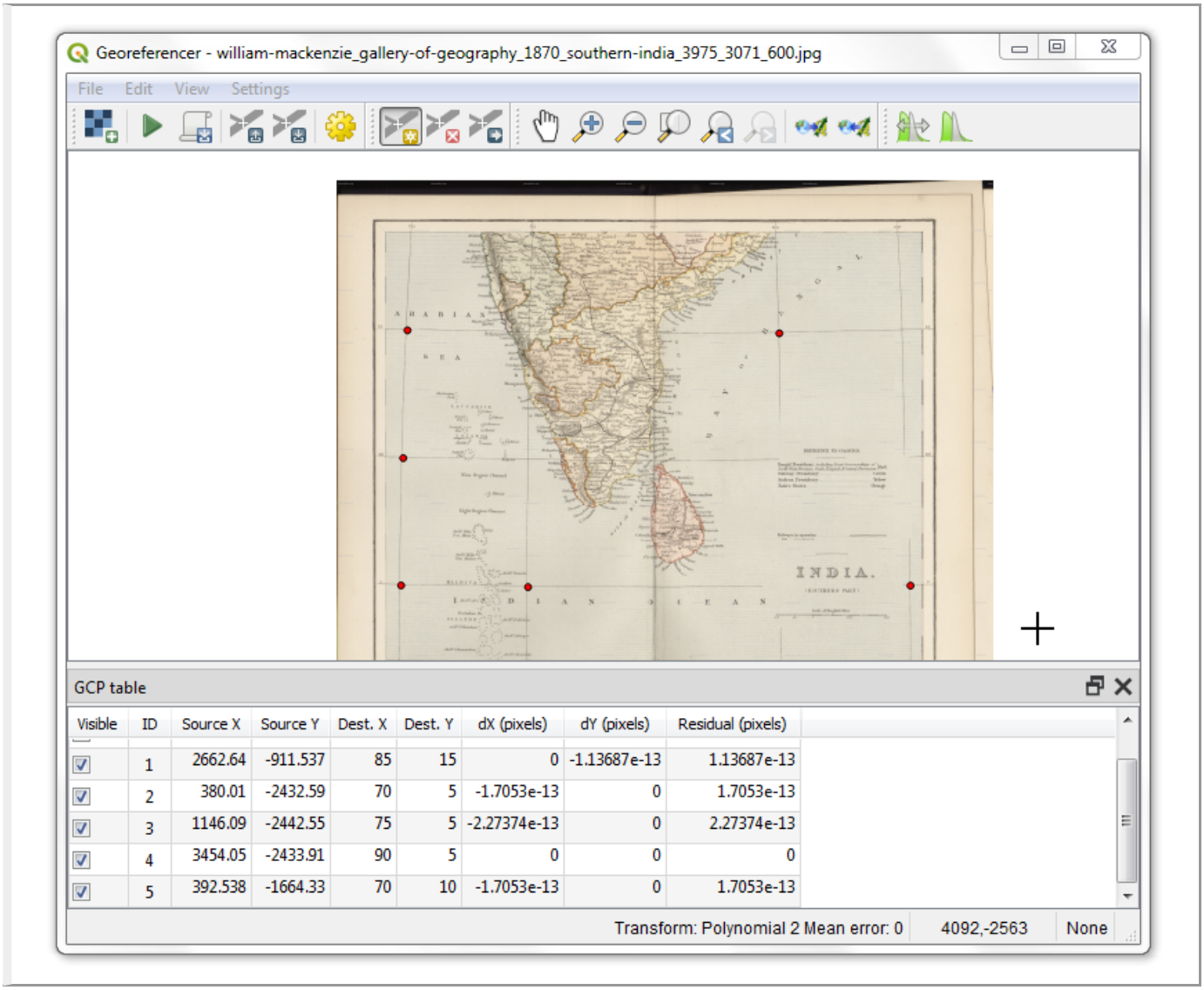
13 . In the pop-up window, enter the coordinates. Remember that X=longitude and Y=latitude. Click OK.



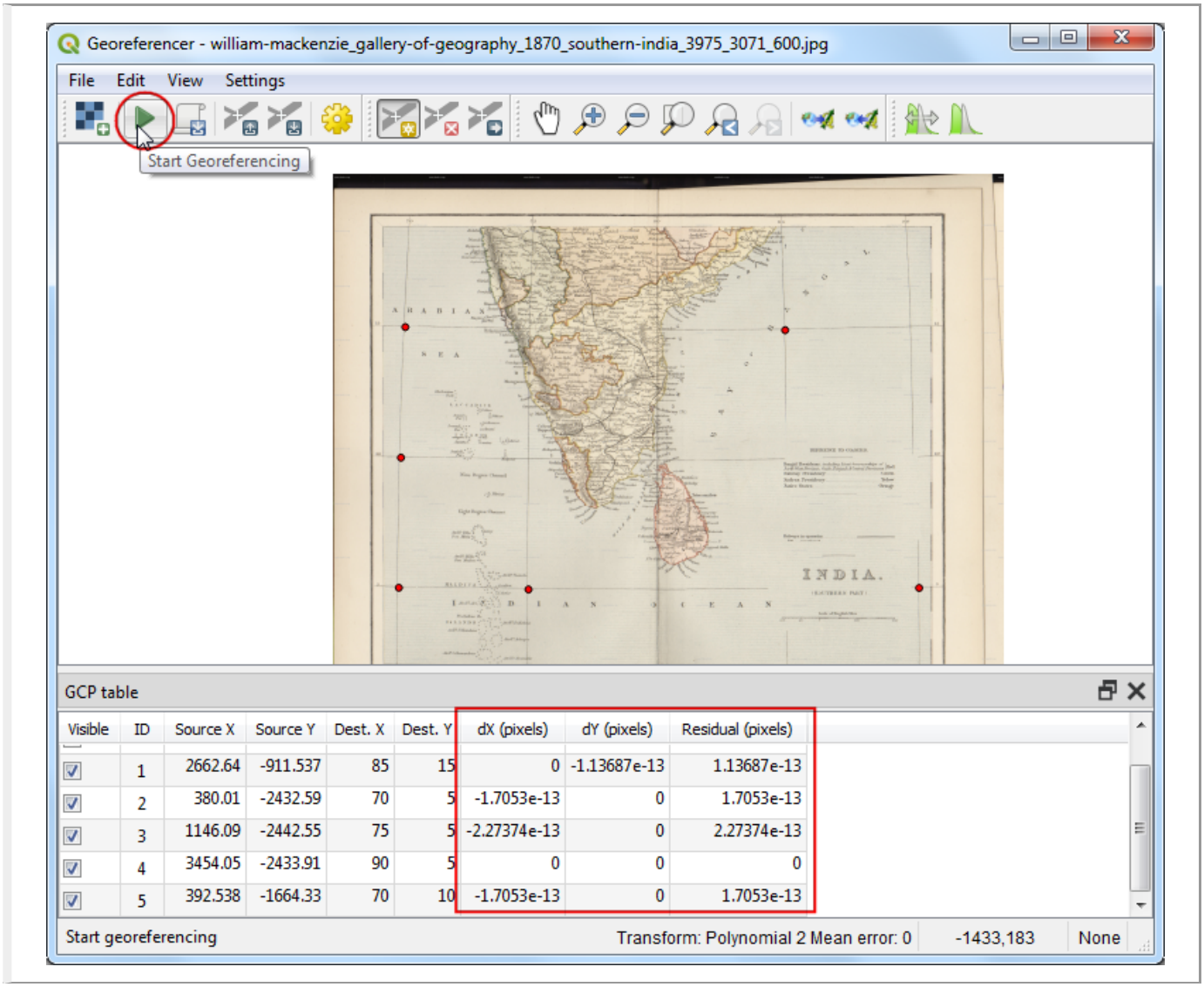
14. You will notice the GCP table now has a row with details of your first GCP.



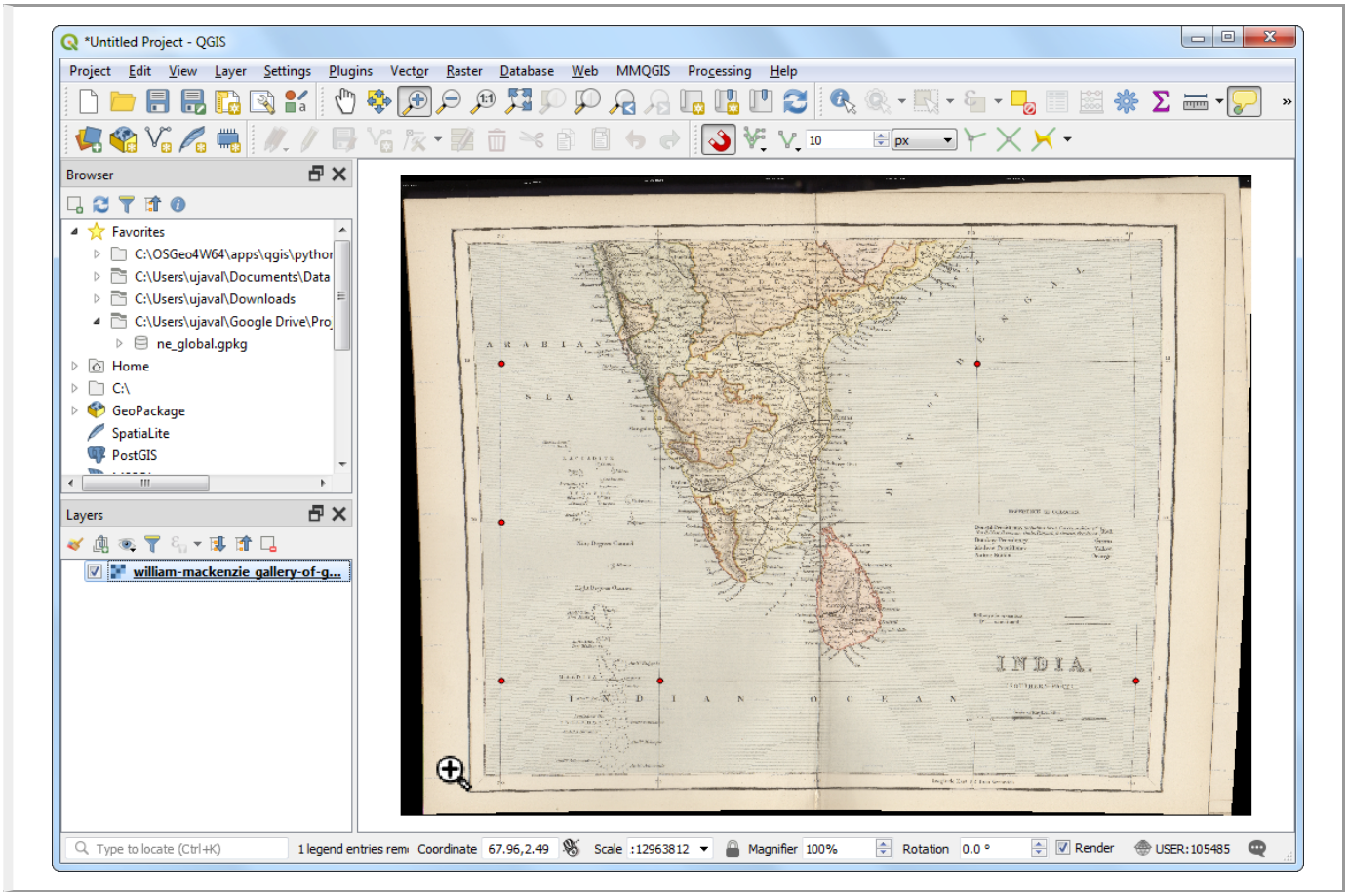
15. Similarly, add at least more GCPs covering the entire image. The more points you have, the more accurate your image is registered to the target coordinates. The `Polynomial 2` transform requires at least 6 GCPs.



16. Once you have added the minimum number of points required for the transform, you will notice that the GCPs now have a non-zero `dX`, `dY` and `Residual` error values. If a particular GCP has unusually high error values, that usually means a human-error in entering the coordinate values. So you can delete that GCP and capture it again. You can also edit the coordinate values in the GCP Table by clicking the cell in either Dest. X or Dest. Y columns. Once you are satisfied with the GCPs, go to File ▸ Start georeferencing. This will start the process of warping the image using the GCPs and creating the target raster.



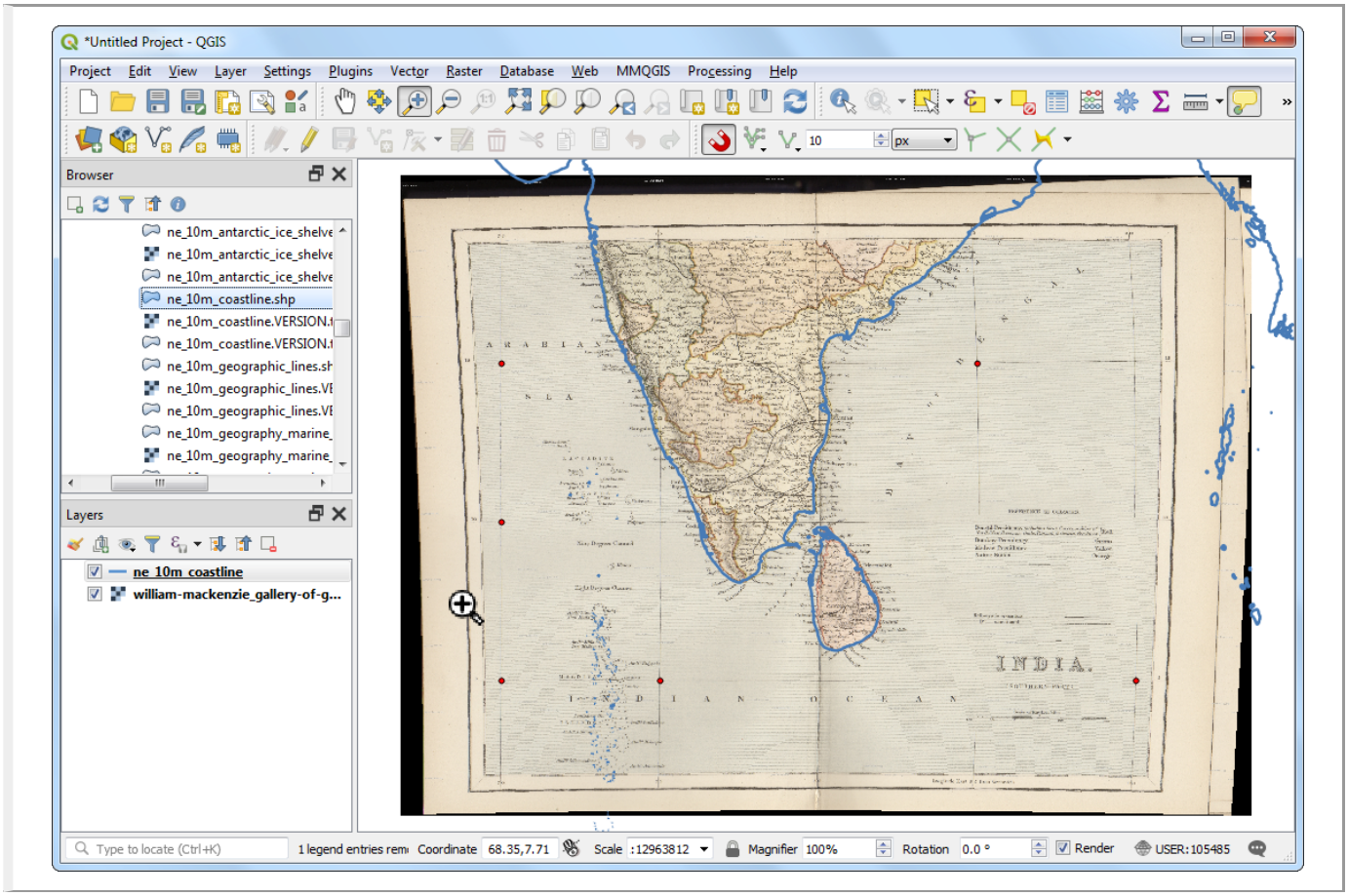
17. Once the process finishes, you will see the georeferenced layer loaded in QGIS. The georeferencing is now complete.



Note

The GCPs will also be displayed in the main QGIS Canvas. If you wish to remove them, you can switch to the Georeferencer window, and choose File ▸ Reset Georeferencer.

18. It is a good practice to verify your work. How do we check if our georeferencing is accurate? In this case, you can load the boundary shapefile from a trusted source like the Natural Earth dataset and compare them. You will notice they match up pretty nicely. There is some error and it can be further improved by taking more control points, changing transformation parameters and trying a different datum.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Georeferencing Aerial Imagery (QGIS3)

In the tutorial *Georeferencing Topo Sheets and Scanned Maps (QGIS3)* (georeferencing_basics.html) we covered the basic process of georeferencing in QGIS. That method involved reading the coordinates from your scanned map and entering them manually as control points. Many times though you may not have the coordinates printed on your map, or you are trying to georeference an image. In that case, you can use another georeferenced data-source as your input. In this tutorial, you will learn how to use existing open data sources in your georeferencing process.

Overview of the task

We will georeference high resolution balloon-imagery using reference coordinates from OpenStreetMap.

Other skills you will learn

- Downloading super high-resolution public domain imagery.
- Using XYZ Tile Layers as basemap.
- Using the OSM Place Search plugin in QGIS.
- Setting a custom no-data value for a layer.

Get the data

In this tutorial, we will be using kite and balloon imagery collected by The Public Laboratory (http://publiclaboratory.org/archive). They make the georeferenced versions of the images also available, but we will download a non-georeferenced JPG image and go through the process of georeferencing it in QGIS.

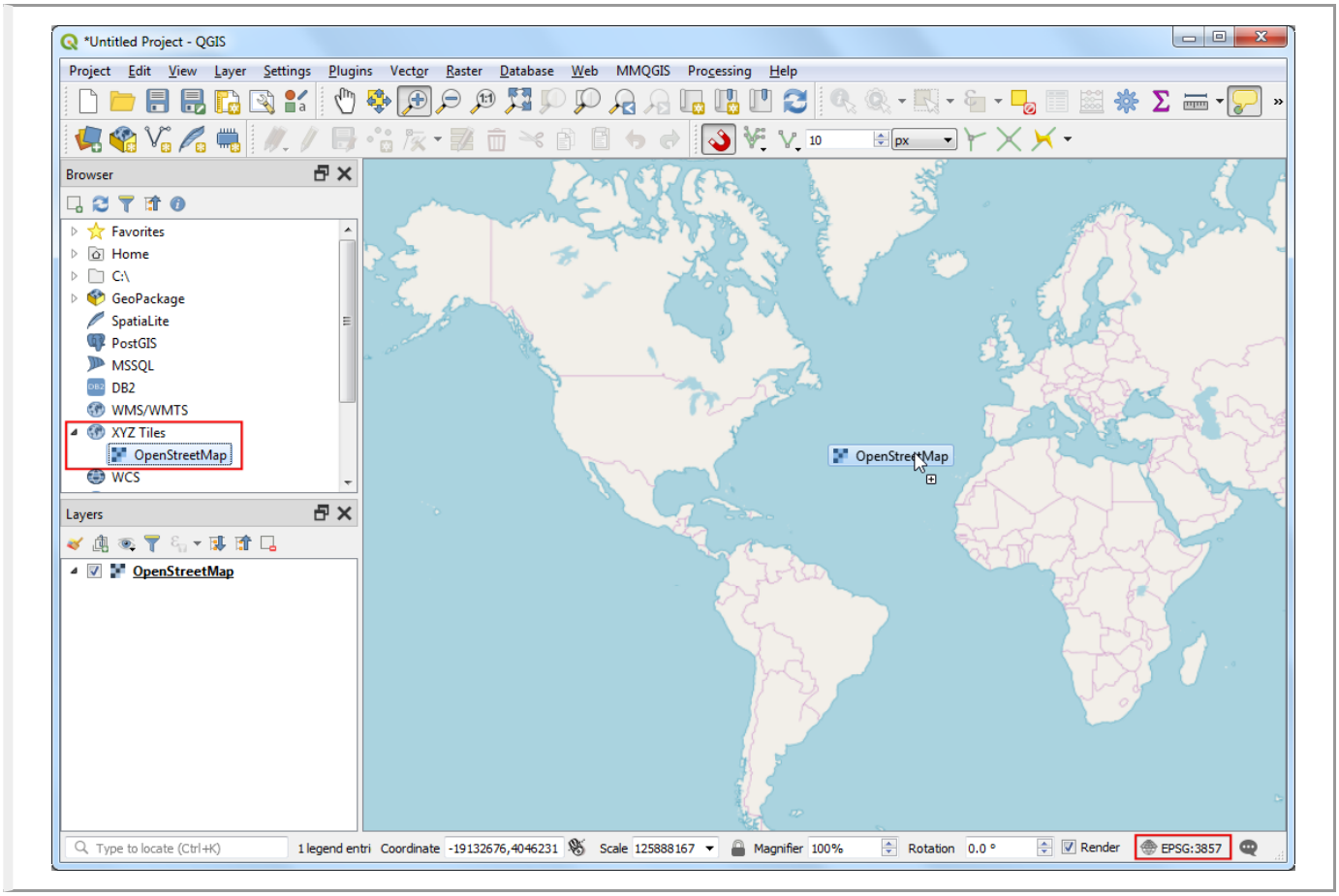
Download the JPG image of Washington Square Park, New York (http://publiclaboratory.org/map/washington-square-park-new-york-new-york/2012-10-01). You can right-click the JPG button and choose Save link as....

For convenience, you may directly download a copy of the dataset from the link below:

newyorkcity-washingtonsquarepark.jpg (http://www.qgistutorials.com/downloads/newyorkcity-washingtonsquarepark.jpg)

Procedure

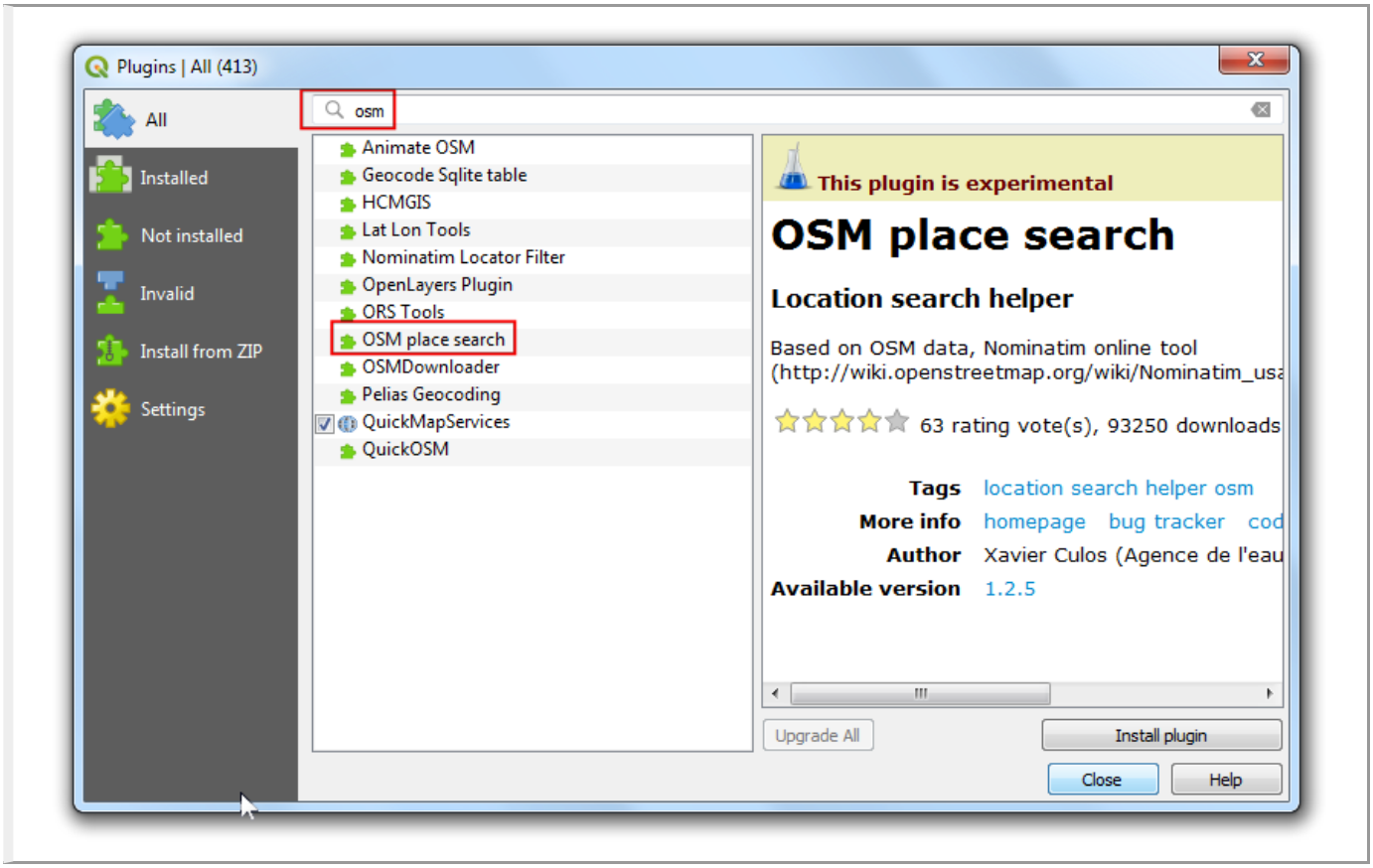
1. We will use a basemap from OpenStreetMap to capture the coordinates for georeferencing. QGIS3 comes with built-in support for tile layers. These are commonly known as 'XYZ' layers since they are made using individual map tiles for each zoom level (z) on a x,y coordinate grid. You can find the `OpenStreetMap` layer under XYZ Tiles in the Browser Panel. Drag the layer to the main canvas. Once loaded, note the Coordinate Reference System (CRS) for this layer in the bottom-right corder. It is set as `EPSG 3857 Pseudo Mercator`. This is important because the coordinates we infer from this layer during georeferencing will be in this CRS.



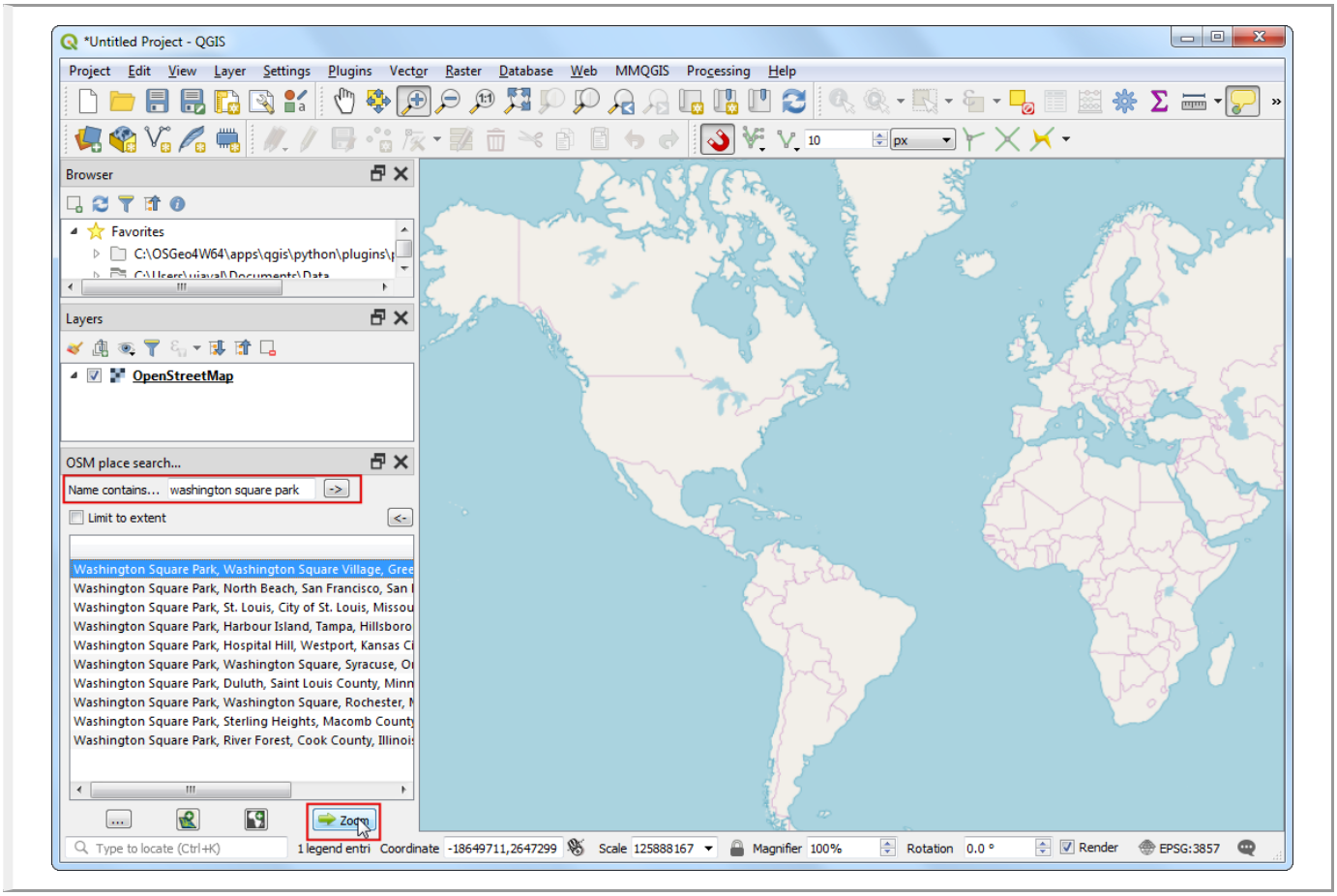
Note

See this page (https://www.spatialbias.com/2018/02/qgis-3.0-xyz-tile-layers/) for more details on XYZ layers and how to add other basemaps in QGIS.

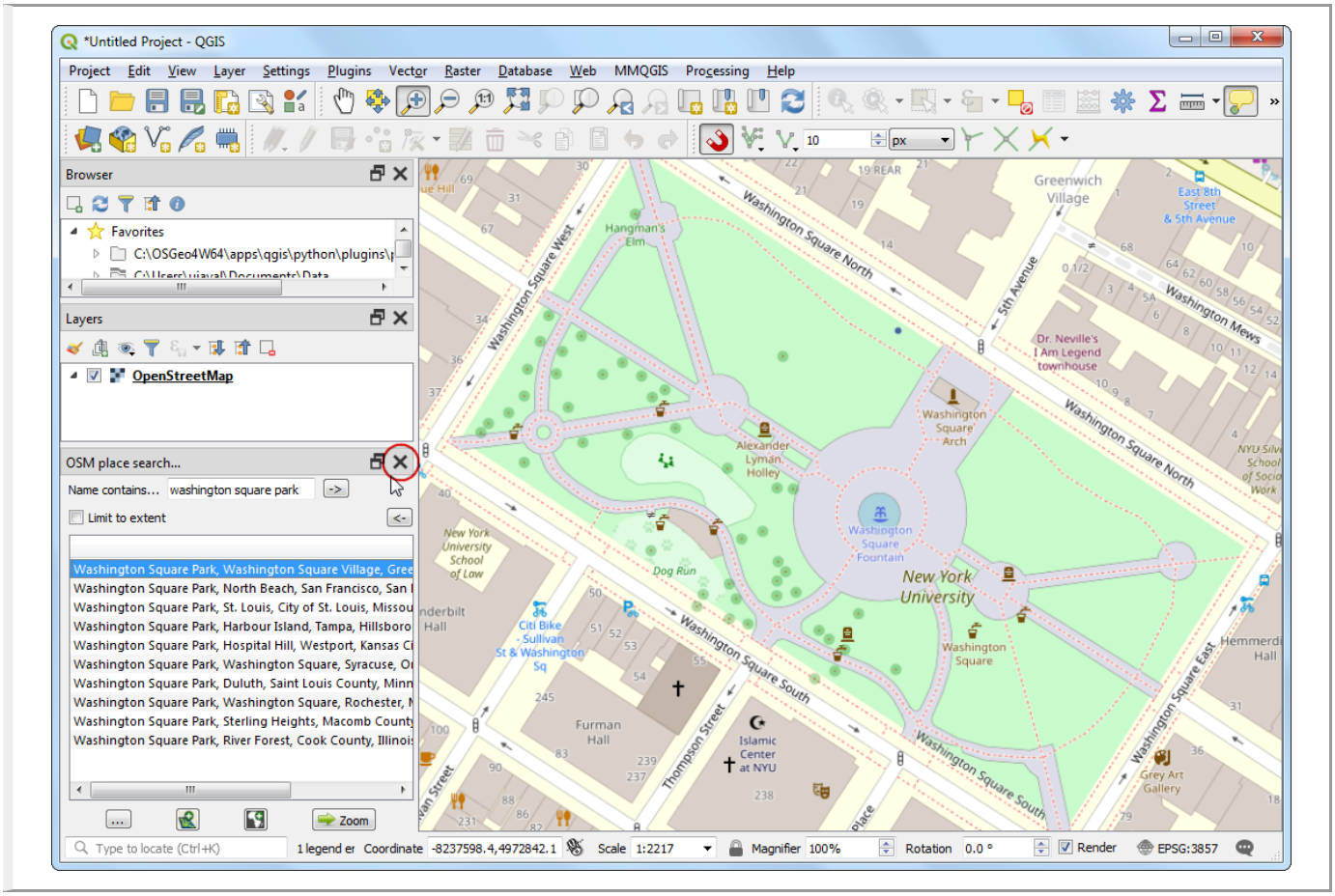
2. The image we are georeferencing is for `Washington Square Park, New York`. You can zoom/pan try to locate this park in the map. But that is cumbersome and may not be practical. An easier way is to use the OpenStreetMap (OSM) Place Search plugin to search for the exact location. Install the `OSM Place Search` plugin from Plugins ▸ Manage and install plugins ▸ All. If you do not see this plugin in the search results, make sure you have checked Also show experimental plugins under Settings. See *Using Plugins* (../using_plugins.html) for more information on using plugins in QGIS.



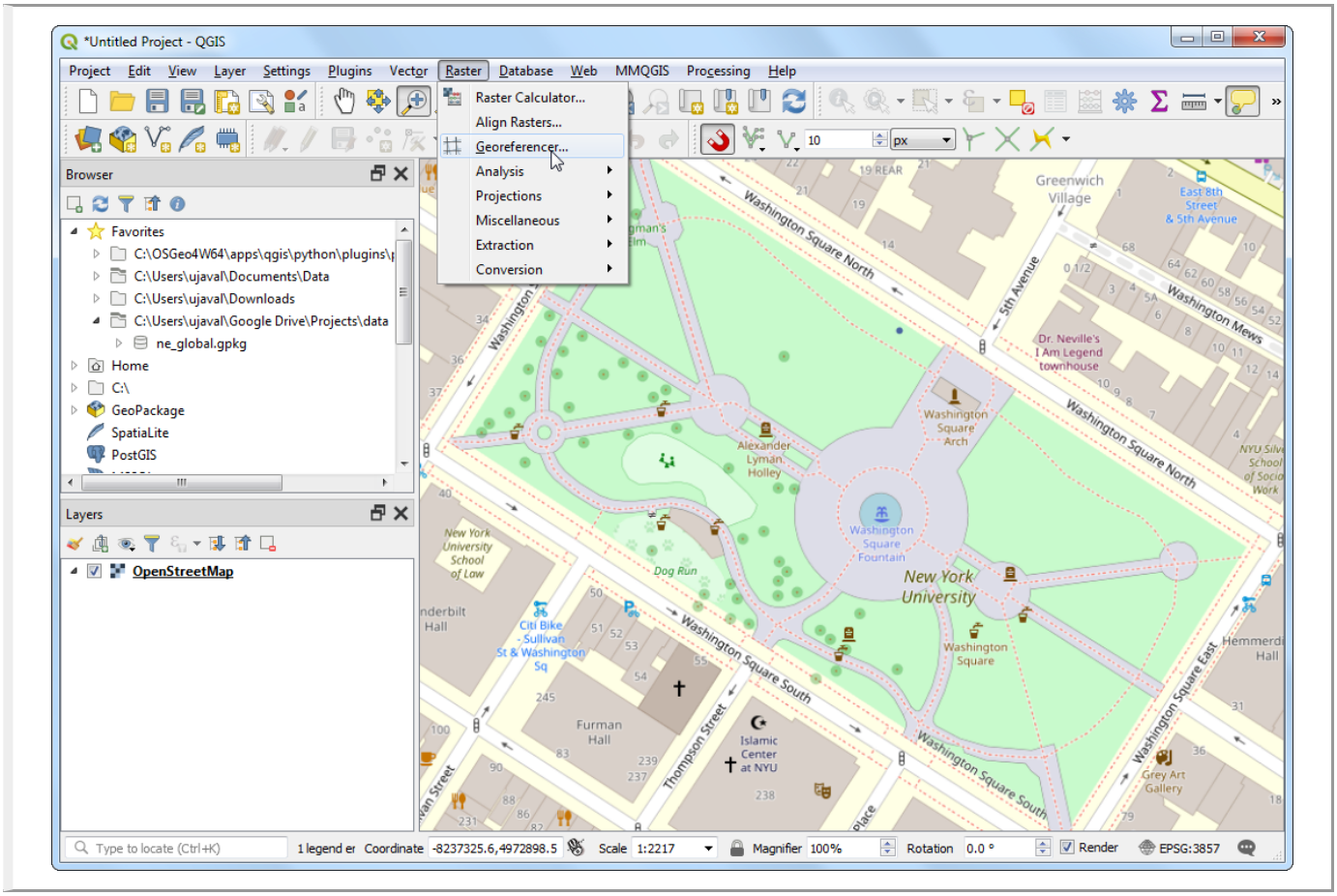
3. Once the plugin is installed, you will see a new panel called OSM Place Search.... Search for `Washington Square Park` in the Name contains.. box and click ->. You will see the matching place names appear in the results panel. Select the correct result and click the Zoom button.



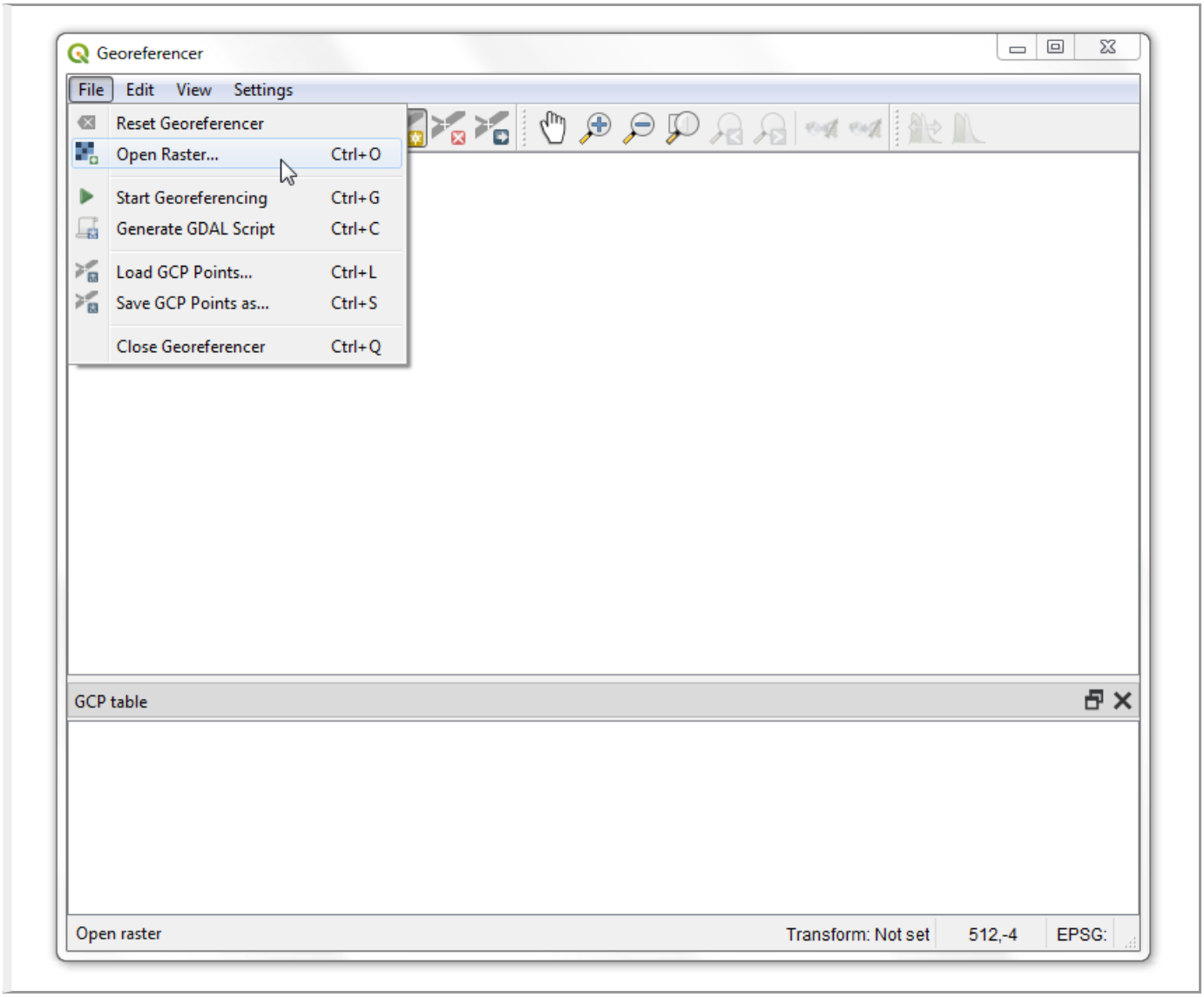
4. You will see the map that is familiar and contains the landmarks that we can identify from the image. You may close the OSM Place Search panel now. If you need it again, you can open it from View ▸ Panels ▸ OSM Place Search.



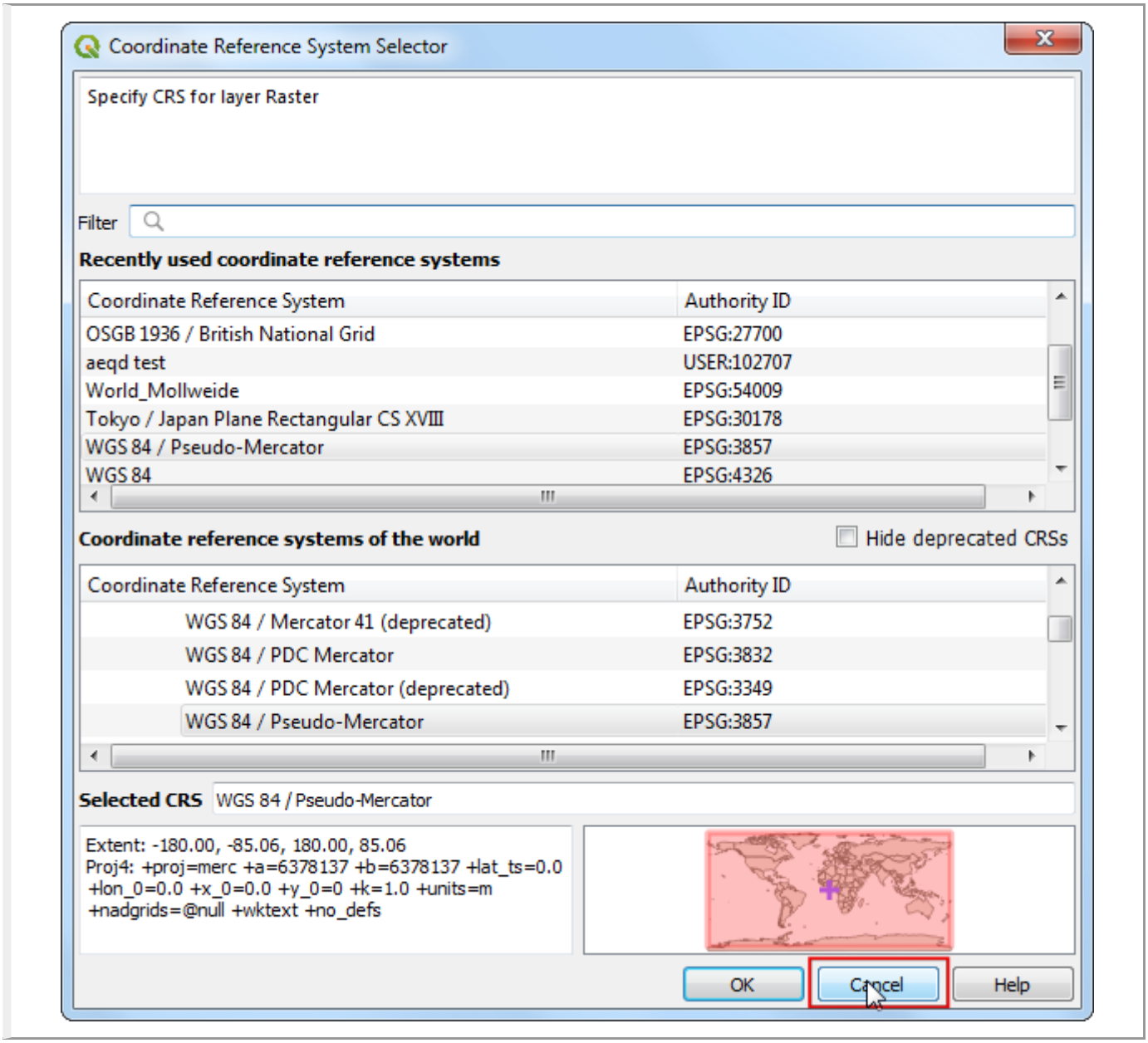
5. Now it is time to start georeferencing. Launch the **Georeferencer** from Raster ▸ Georeferencer ▸ Georeferencer. If you do not see that menu item, you will need to enable the Georeferencer GDAL plugin from Plugins ▸ Manage and install Plugins ▸ Installed.



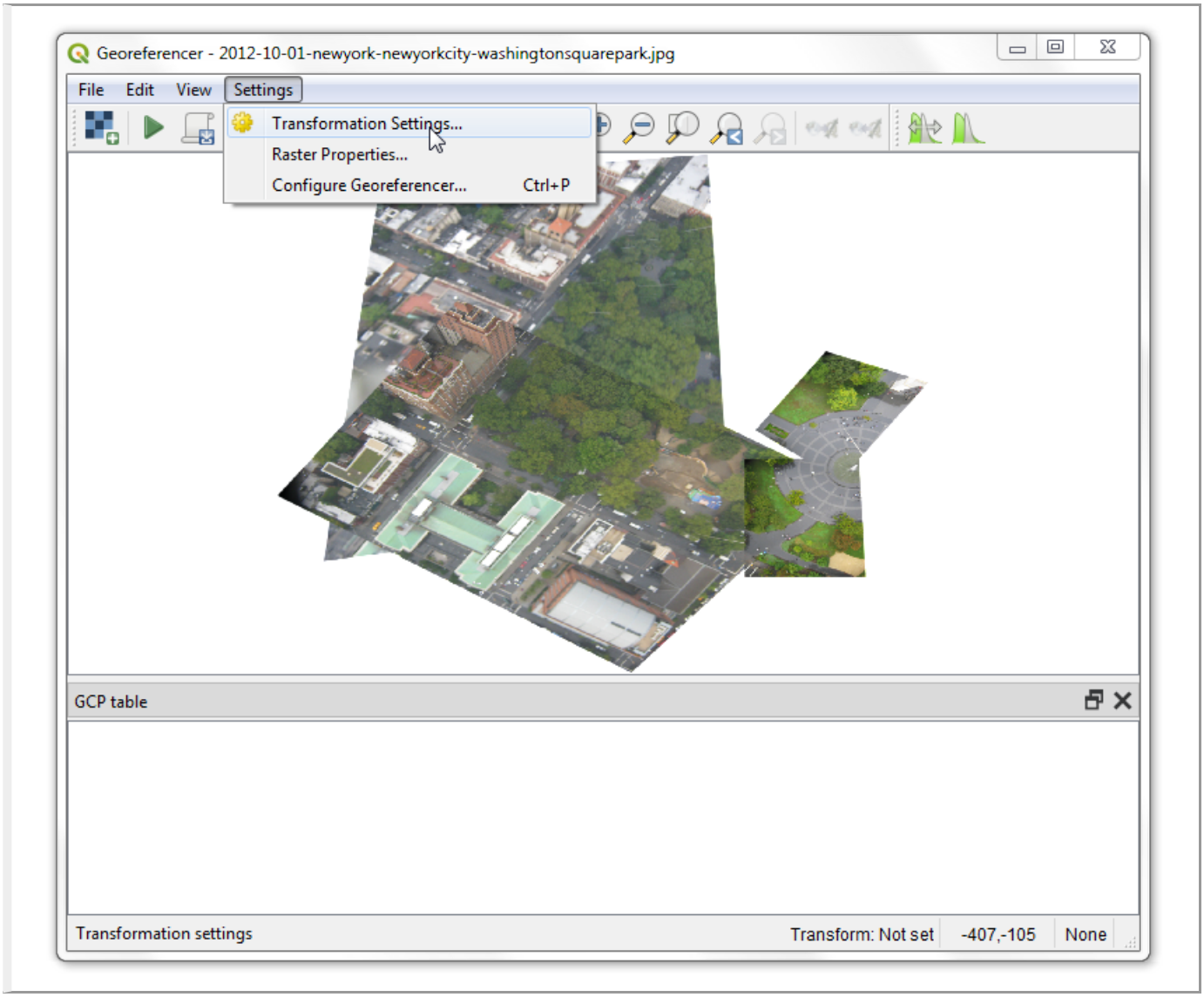
6. In the Georeferencer window, go to File ▸ Open Raster. Navigate to the downloaded JPG file and click Open.



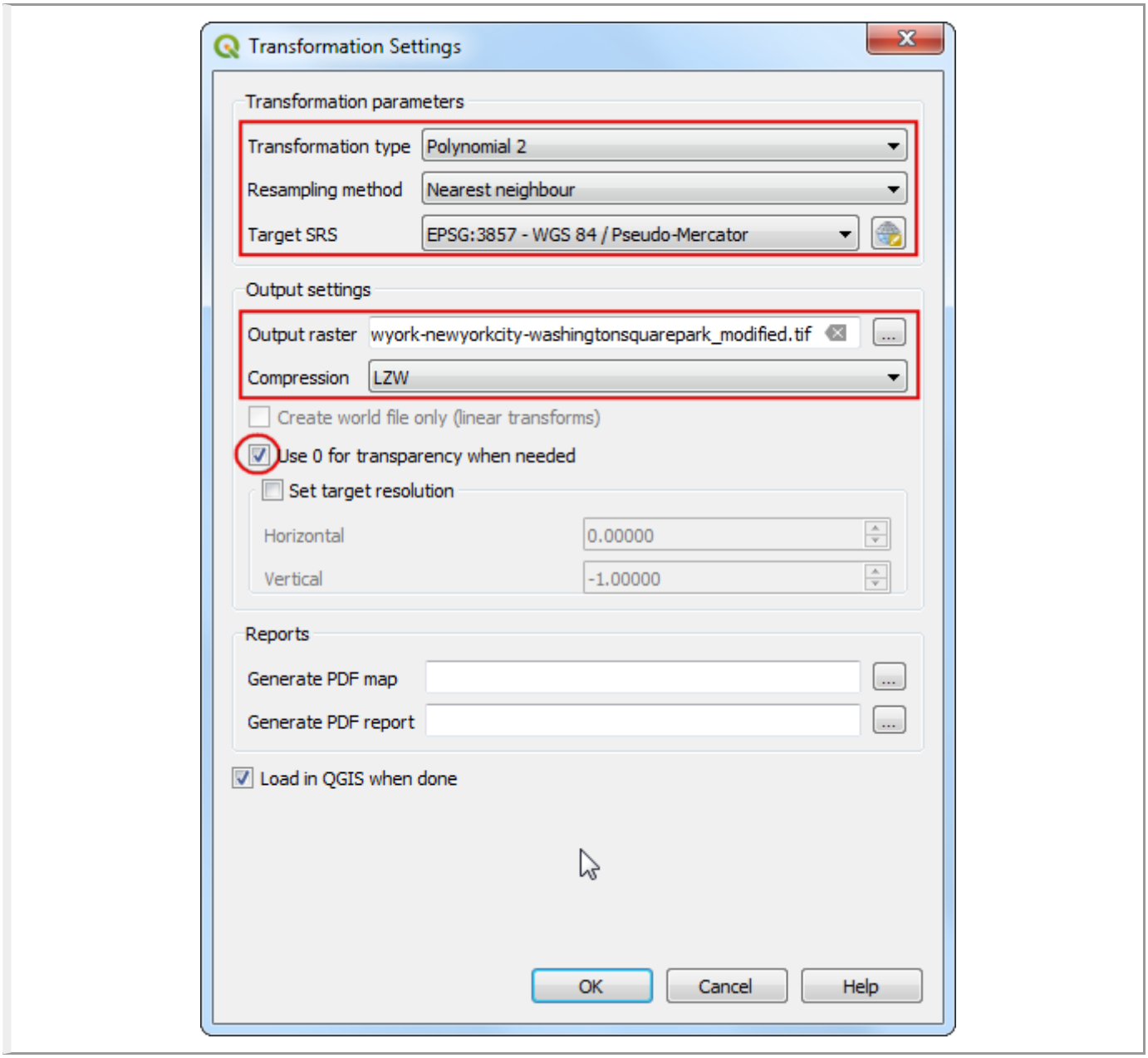
7. In the next screen, you will asked to choose the raster's coordinate reference system (CRS). Our source image is a plain JPEG file and doesn't have any coordinate reference system atached to it, so you can click Cancel.



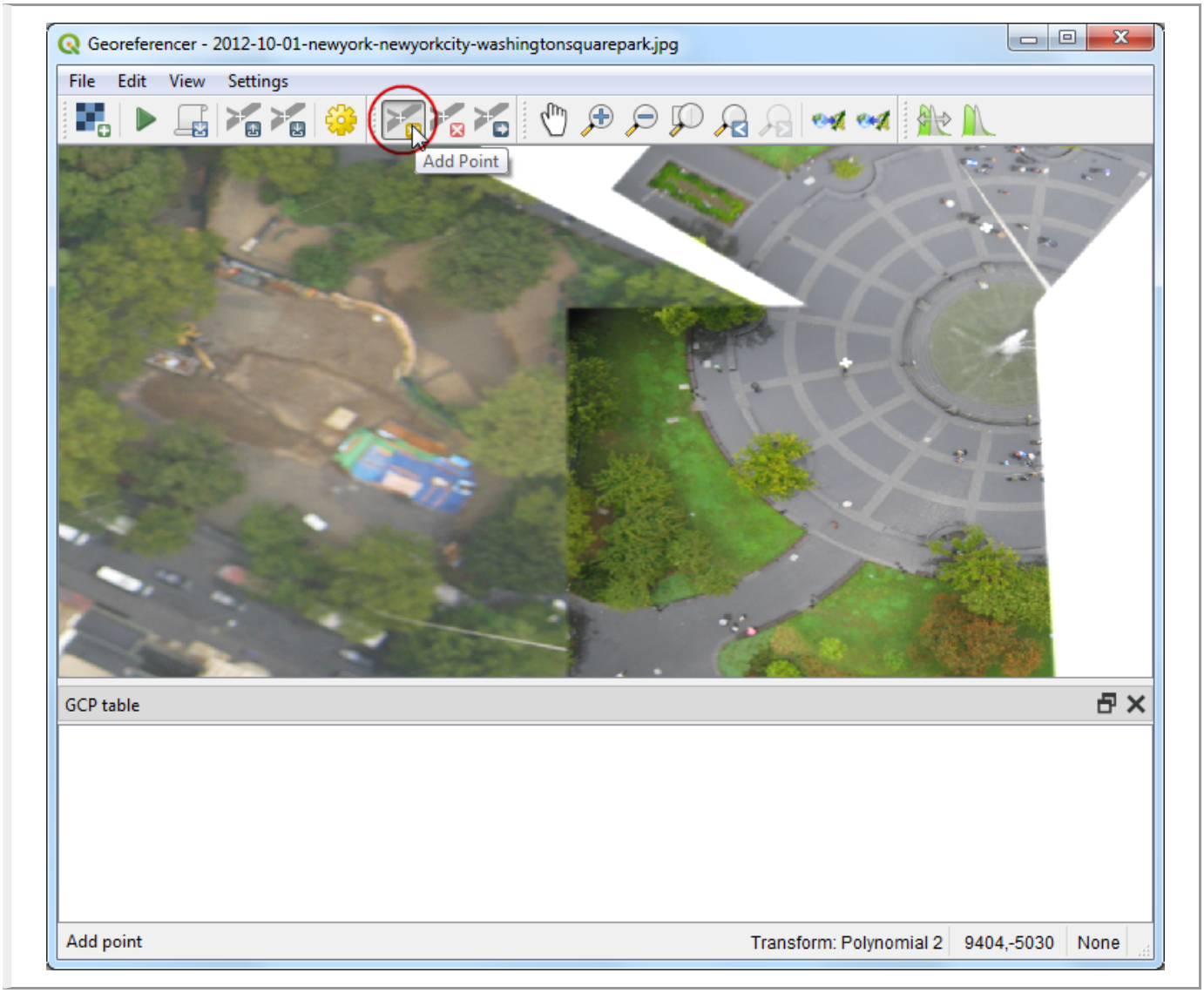
8. Before we start adding Ground Control Points (GCP), we need to define the Transformation Settings. Go to Settings ‣ Transformation settings.



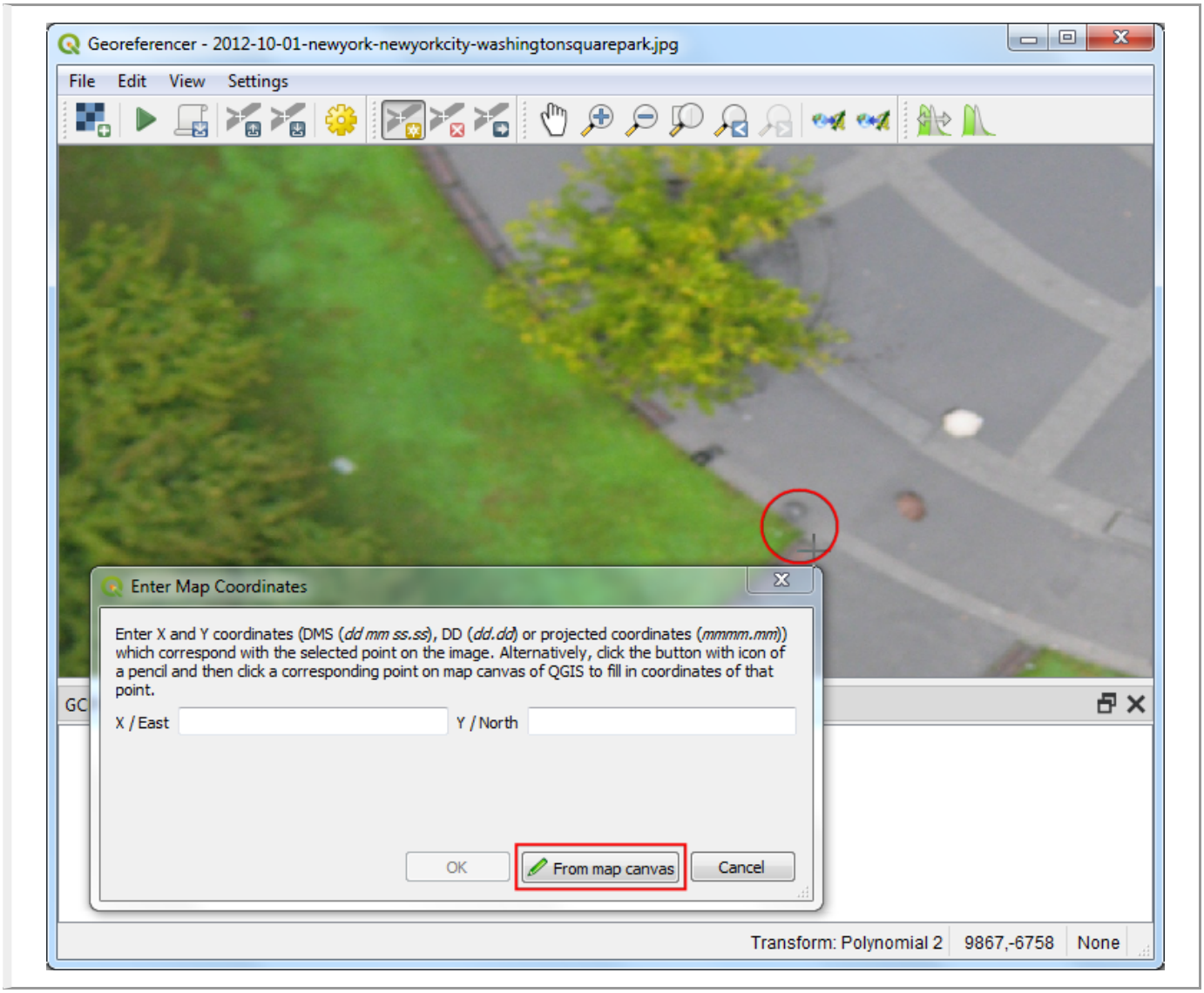
9. In the Transformation settings dialog, choose the Transformation type as `Polynomial 2`. See QGIS Documentation (https://docs.qgis.org/testing/en/docs/user_manual/plugins/plugins_georeferencer.html#available-transformation-algorithms) to learn about different transofrmation types and their uses. As noted earlier, our basemap is in `EPSG 3857 Pseudo Mercator` CRS, so set that as the Target CRS. You can leave the Output raster name to the default and choose `LZW` as the Compression. Check the Use 0 for transparency when needed. Make sure the Load in QGIS when done option is checked. CLick OK.



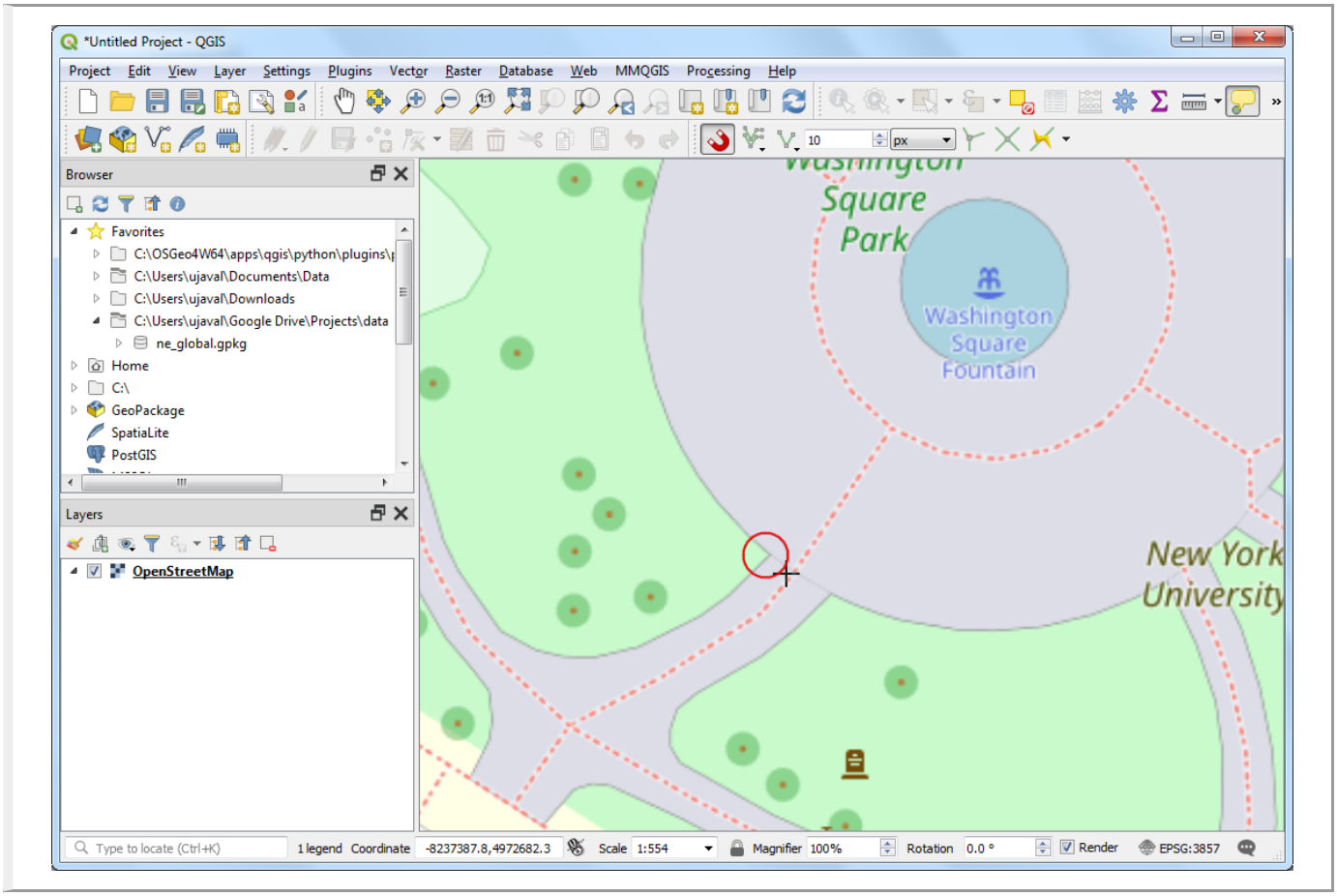
10. Now click on the Add Point button on the toolbar and select an easily identifiable location on the image. Corners, intersections, poles etc. make good control points.



11. Once you click on the image at a control point location, you will see a pop-up asking you to enter map coordinates. Click the button From map canvas.



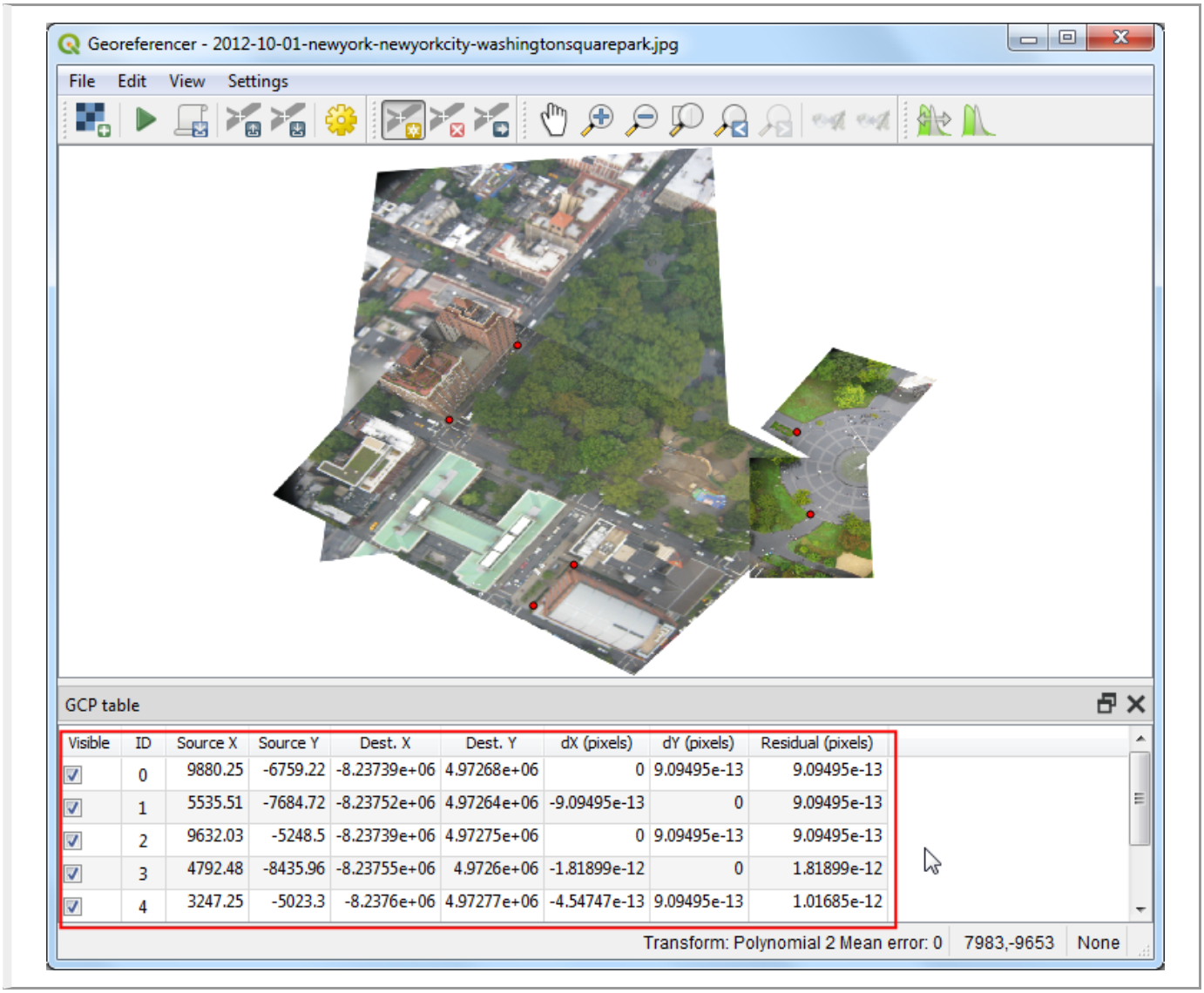
12. Find the same location in the reference layer and click at the precise point. The coordinates are auto-populated from your click on the map canvas. Click Ok. Similarly, choose at least 6 points on the image and add their coordinates from the reference layer.



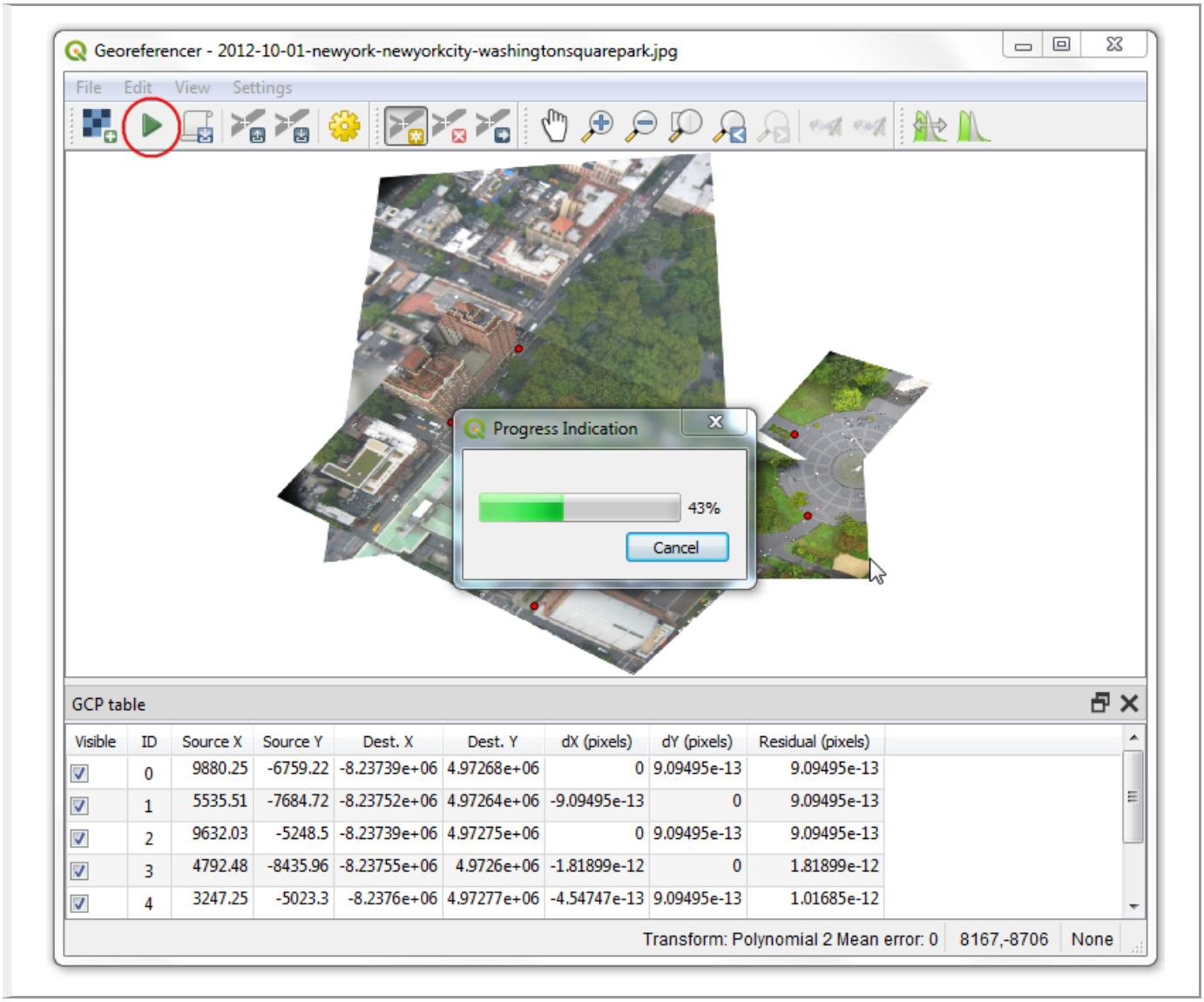
Note

Tip: When selecting a GCP on a building, always choose the bottom of the building. Many aerial and satellite imagery have leaning buildings, so choosing a point on the rooftop will introduce errors.

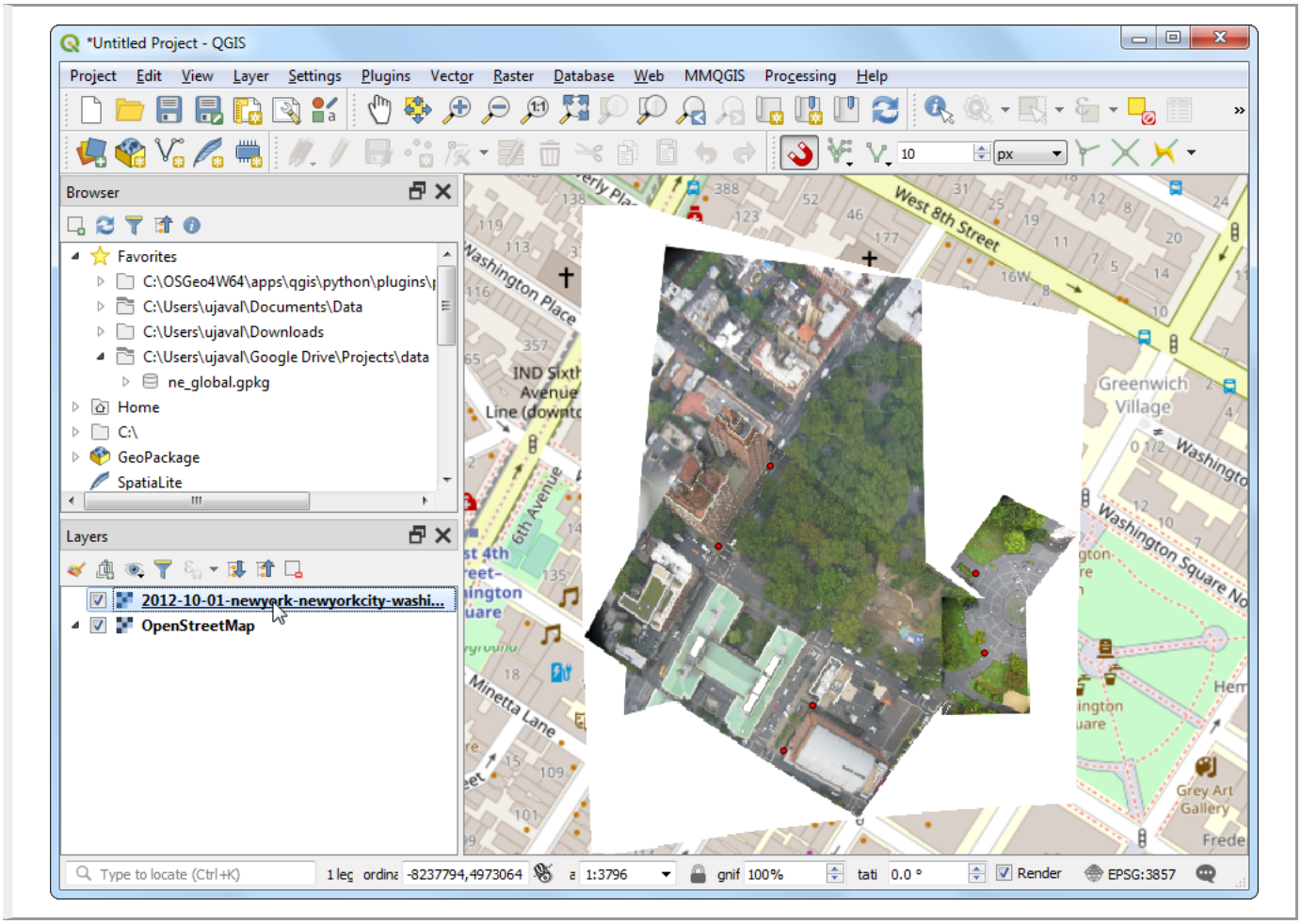
13. Once you have added the minimum number of points required for the transform, you will notice that the GCPs now have a non-zero `dX`, `dY` and `Residual` error values. If a particular GCP has unusually high error values, that usually means a human-error in entering the coordinate values. So you can delete that GCP and capture it again.



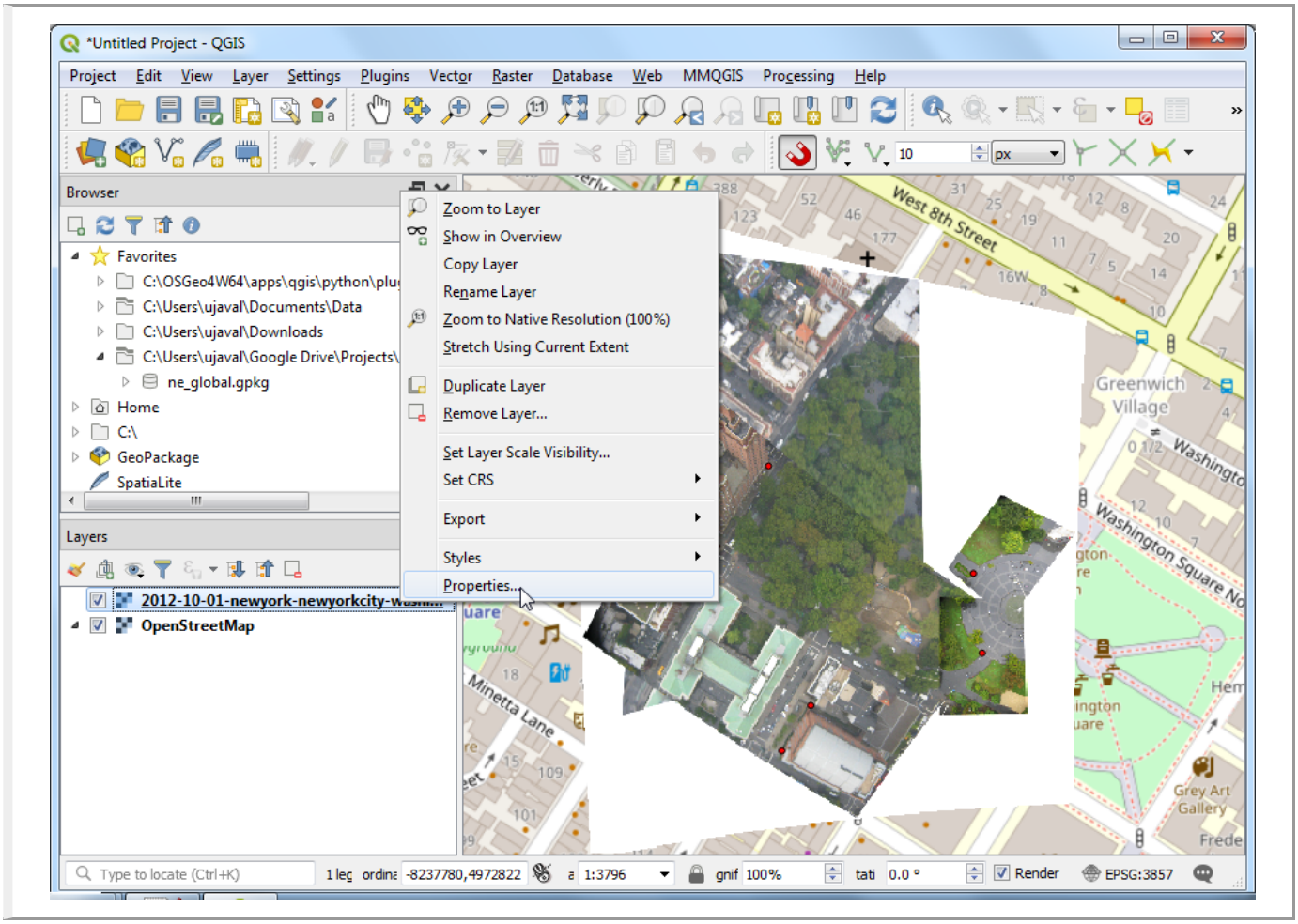
14. Once you are satisfied with the GCPs, go to File ▸ Start georeferencing. This will start the process of warping the image using the GCPs and creating the target raster.



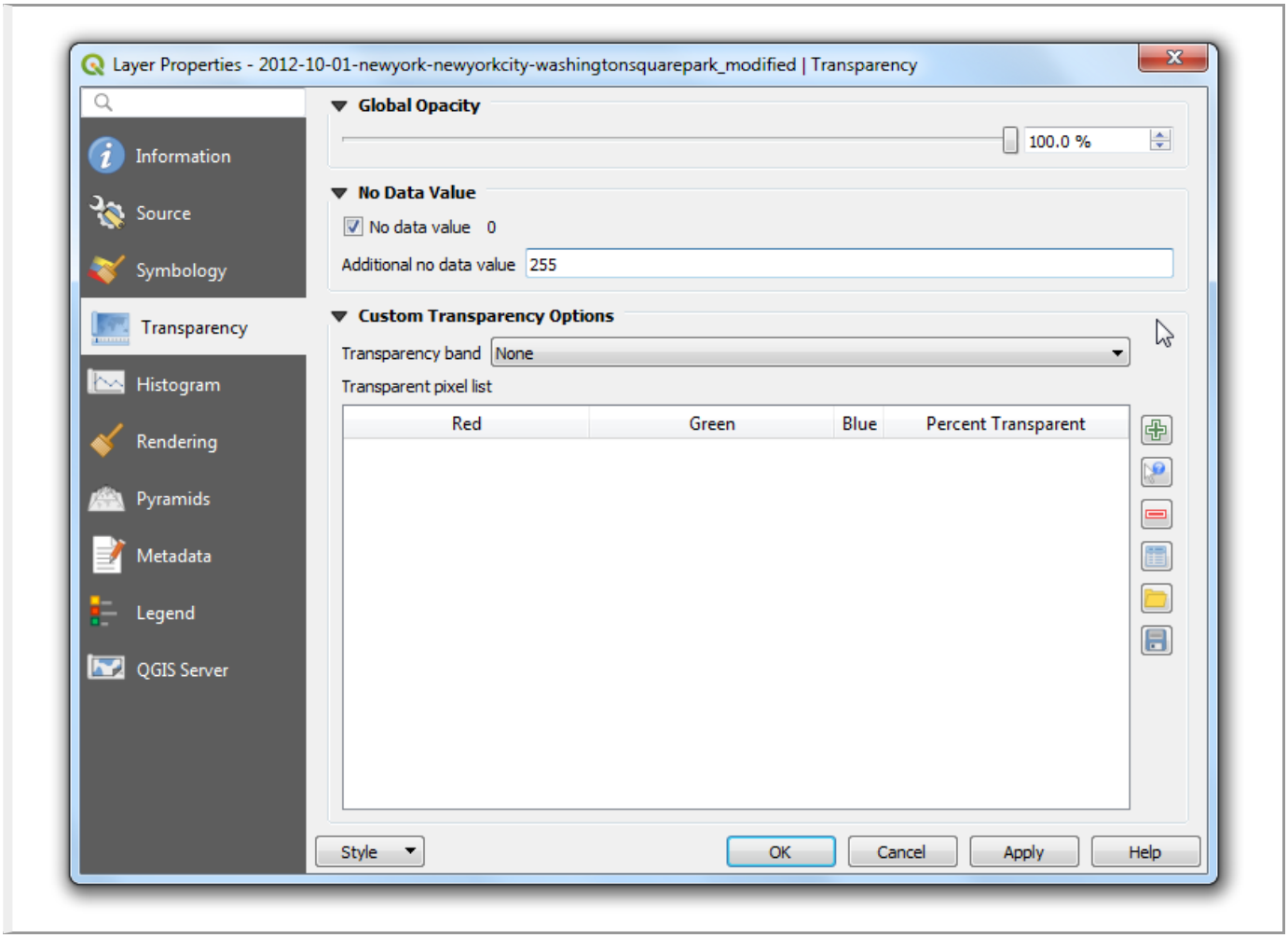
15. Once the process finishes, you will see the georeferenced layer loaded in QGIS. If all went well, you will see it nicely overlay the basemap.



16. To make the output look nicer, let's remove the white border. Right-click on the image layer and choose Properties.



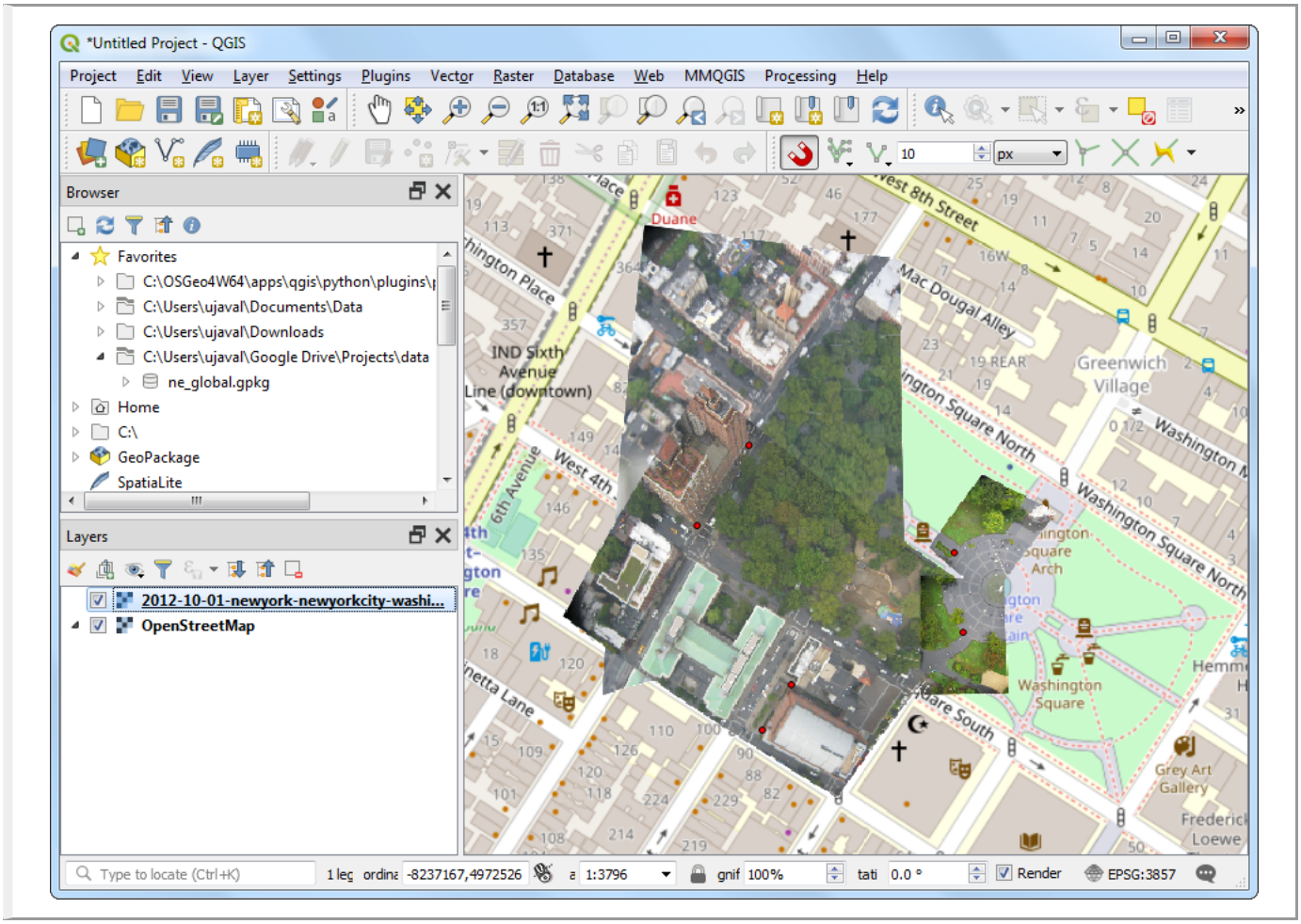
17. Switch to the Transparency tab. Add `255` as the Additional no data value and click OK.



Note

8-bit images have pixel values in the range 0-255. 0 is black and 255 is white.

18. Now you will see your georeferenced image nicely overlaid on the base layer.



Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Digitizing Map Data

Digitizing is one of the most common tasks that a GIS Specialist has to do. Often a large amount of *GIS time* is spent in digitizing raster data to create vector layers that you use in your analysis. QGIS has powerful on-screen digitizing and editing capabilities that we will explore in this tutorial.

Overview of the task

We will use a raster topographic map and create several vector layers representing features around a park.

Other skills you will learn

- Building pyramids for large raster datasets to speed up zoom and pan operations.
- Working with a Spatialite database.

Get the data

Land Information New Zealand (LINZ) (http://www.linz.govt.nz/) provides raster topographic maps at 1:50,000 scale for the New Zealand mainland and Chatham Islands.

Download the GeoTIFF Image file (http://topo.linz.govt.nz/Topo50_raster_images/GeoTIFFTopo50/BX24_GeoTifv1-02.tif) from the Christchurch Topo50 map download page (http://www.linz.govt.nz/topography/topo-maps/map-chooser/christchurch/christchurch#digitalfile).

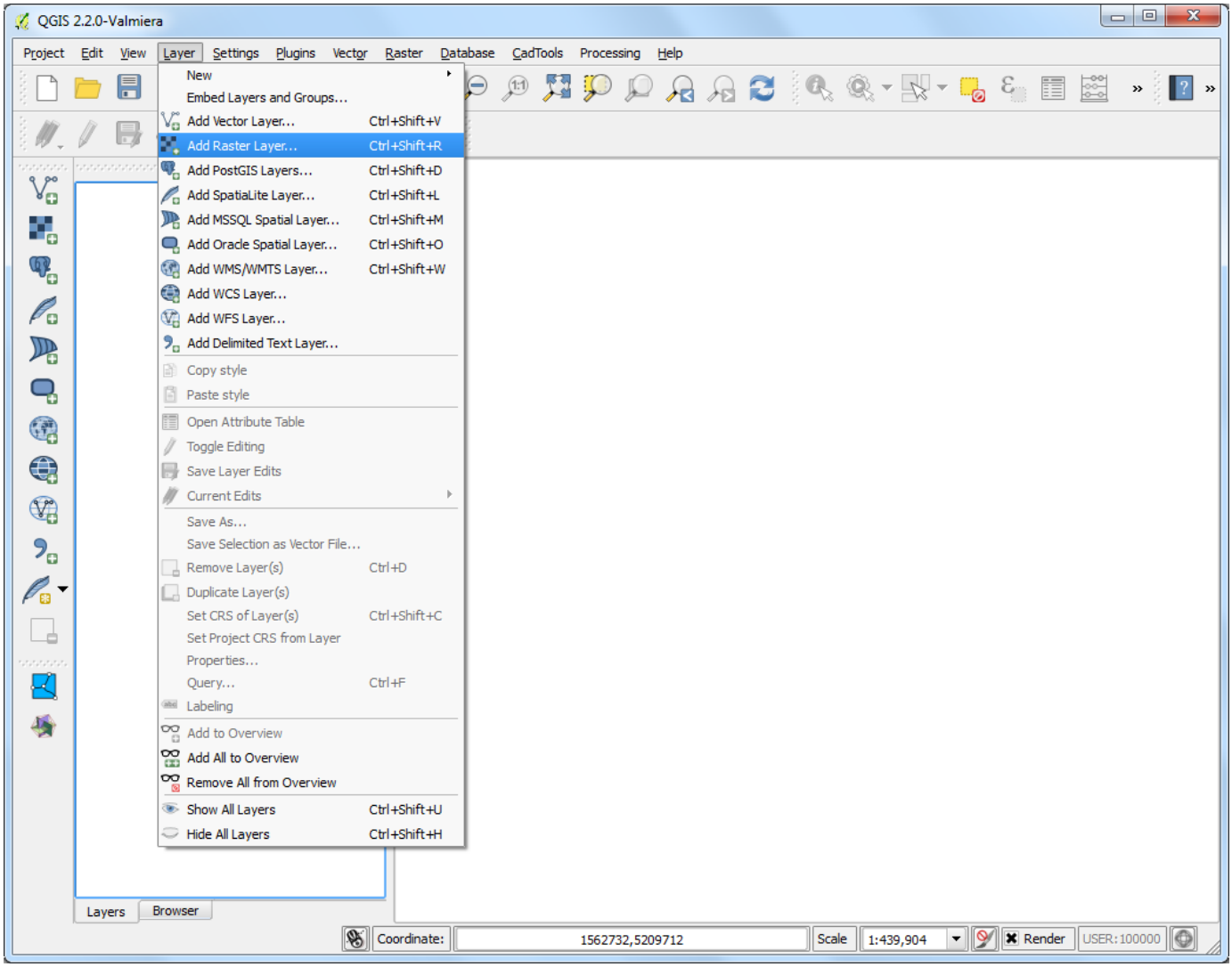
For convenience, you may directly download a copy of the dataset from the link below:

BX24_GeoTifv1-02-clip.tif (http://www.qgistutorials.com/downloads/BX24_GeoTifv1-02-clip.tif)

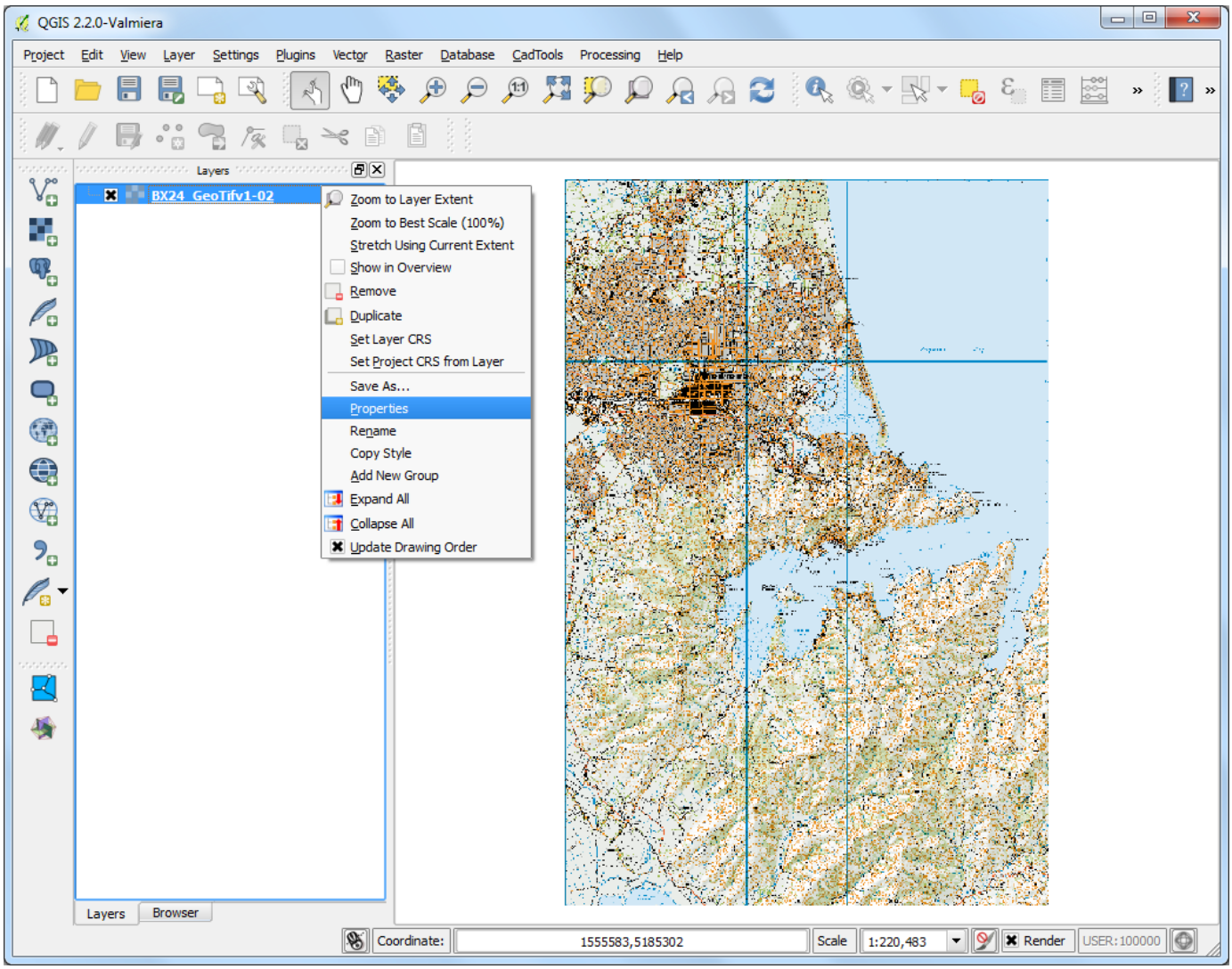
Data Source [LINZ] (credits.html#linz)

Procedure

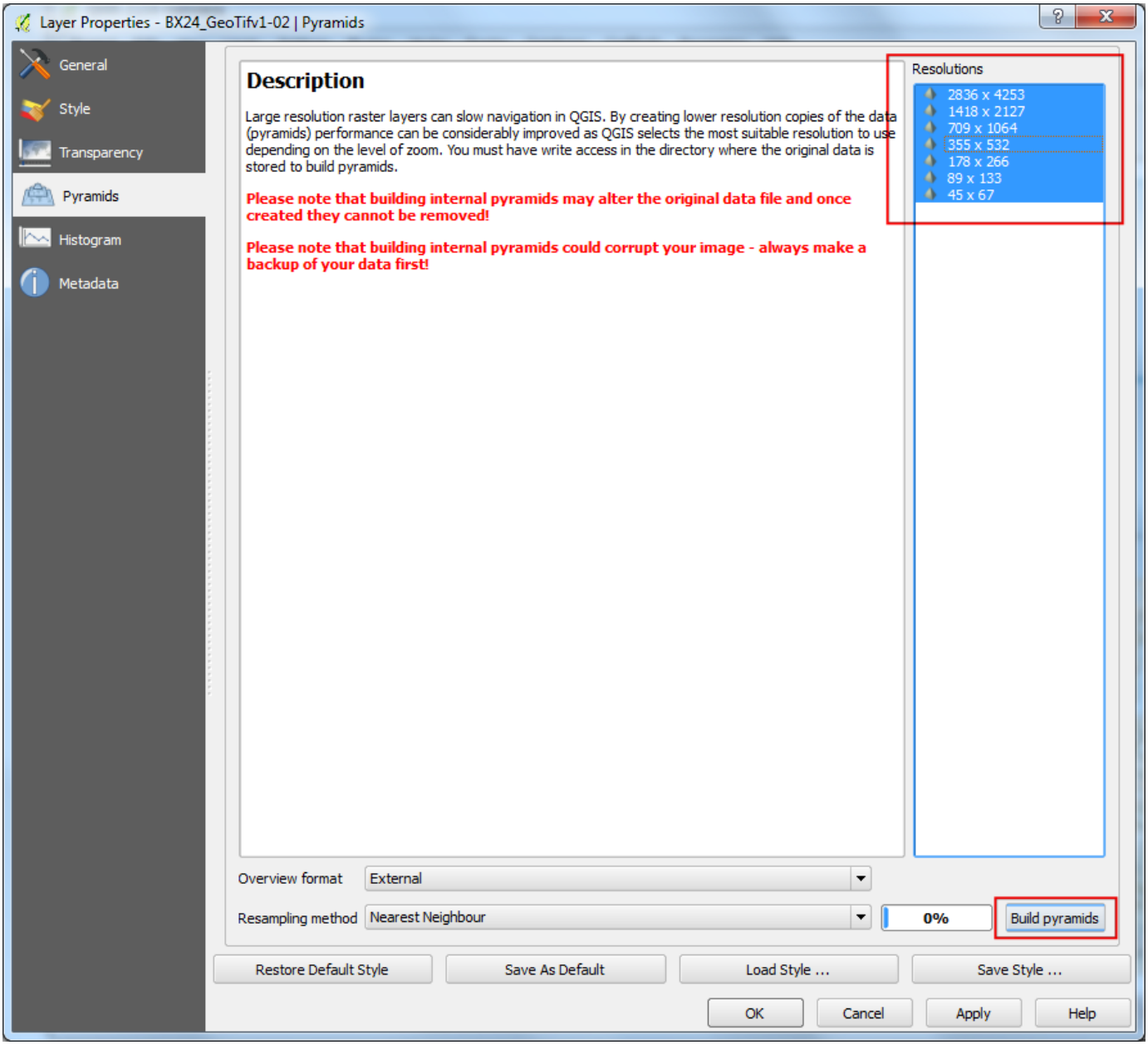
1. Go to Layer ▸ Add Raster Layer. Locate the downloaded `BX24_GeoTifv1-02.tif` and click Open.



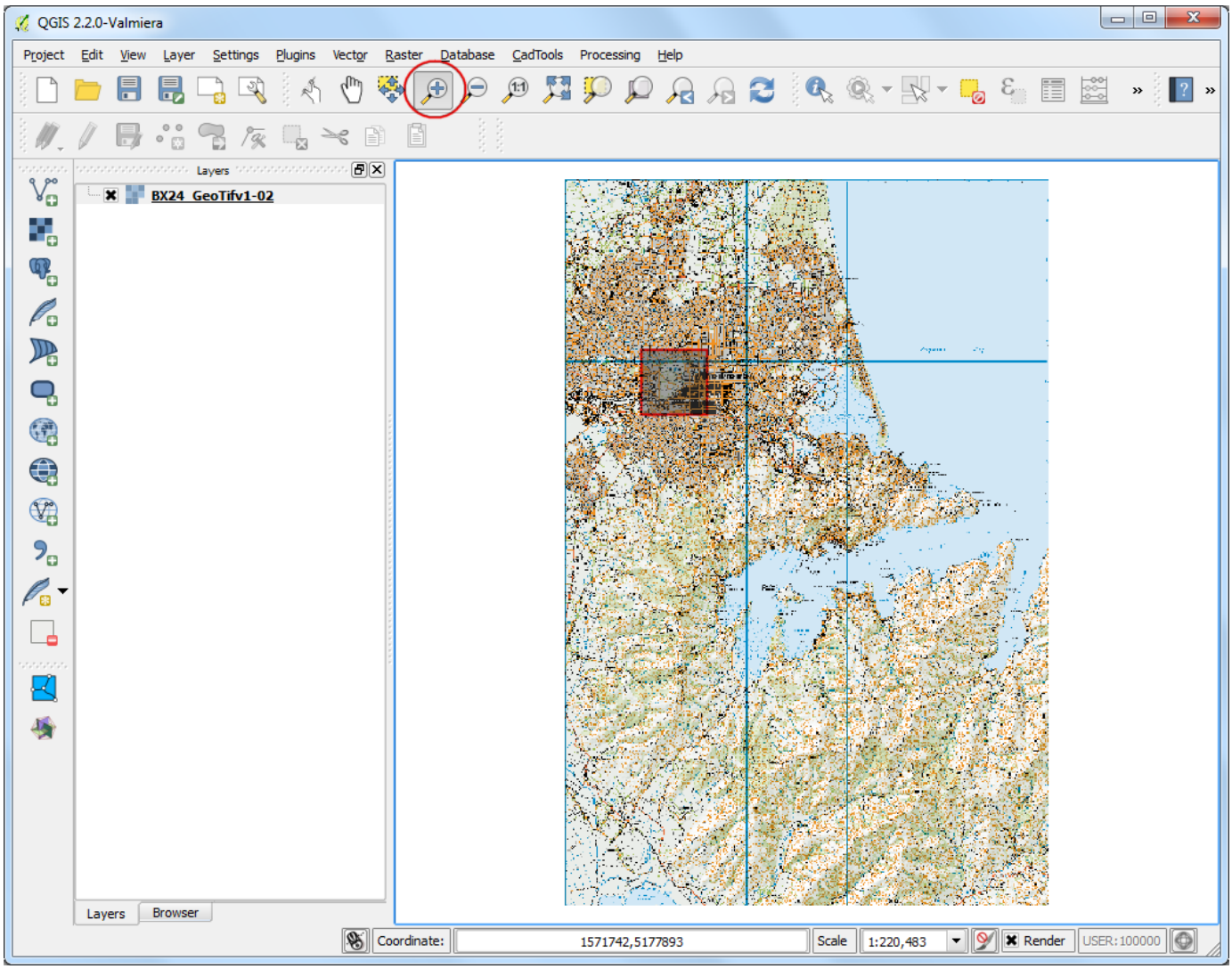
2. This is a large raster file and you may notice that when you zoom or pan around the map, the map takes a little time to render the image. QGIS offers a simple solution to make rasters load much faster by using **Image Pyramids**. QGIS creates pre-rendered tiles at different resolutions and these are presented to you instead of the full raster. This makes map navigation snappy and responsive. Right-click the `BX24_GeoTifv1-02` layer and choose Properties.



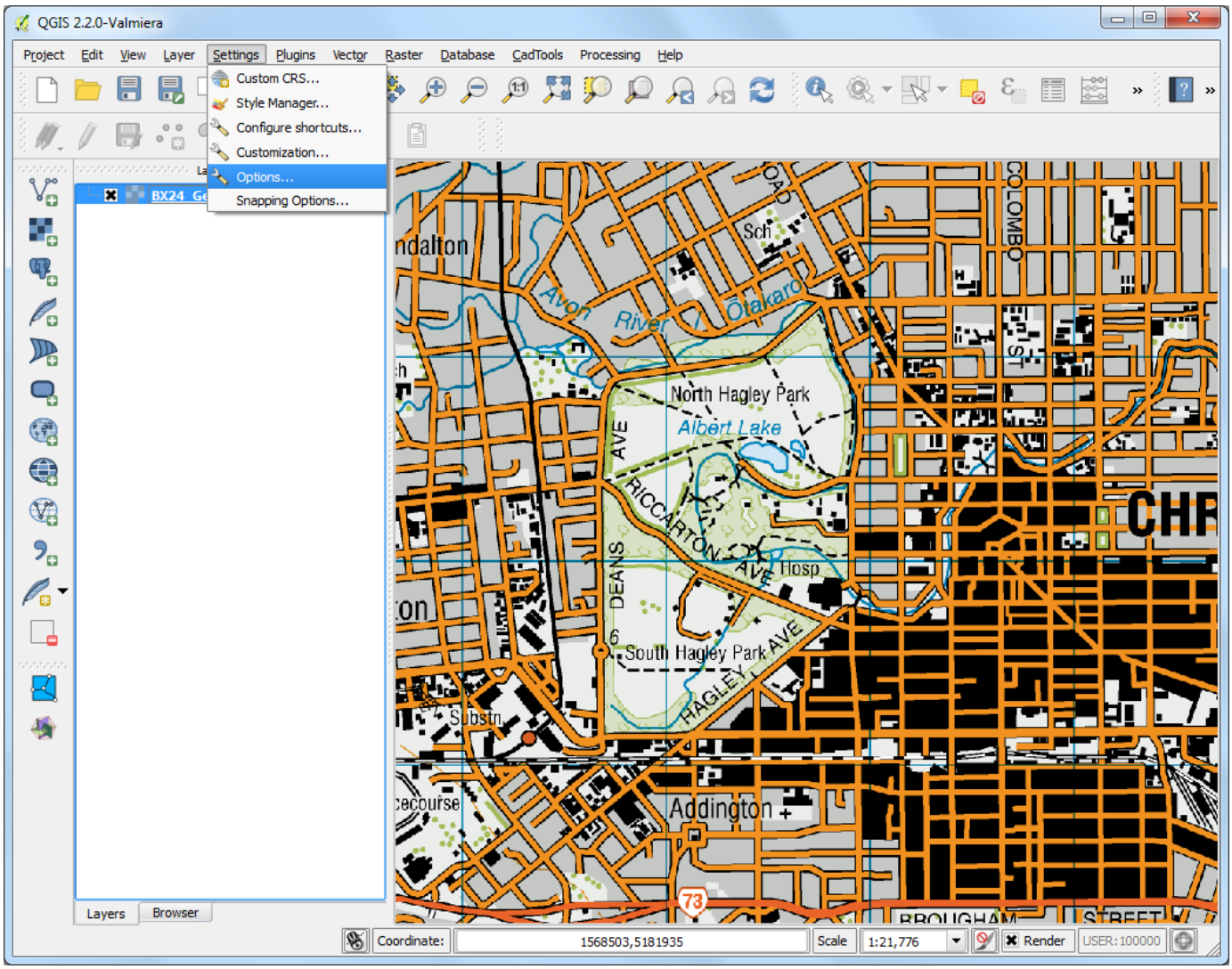
3. Choose the Pyramids tab. Hold the `Ctrl` key and select all the resolutios offered in the Resolutions panel. Leave other options to defaults and click Build pyramids. Once the process finishes, click OK.



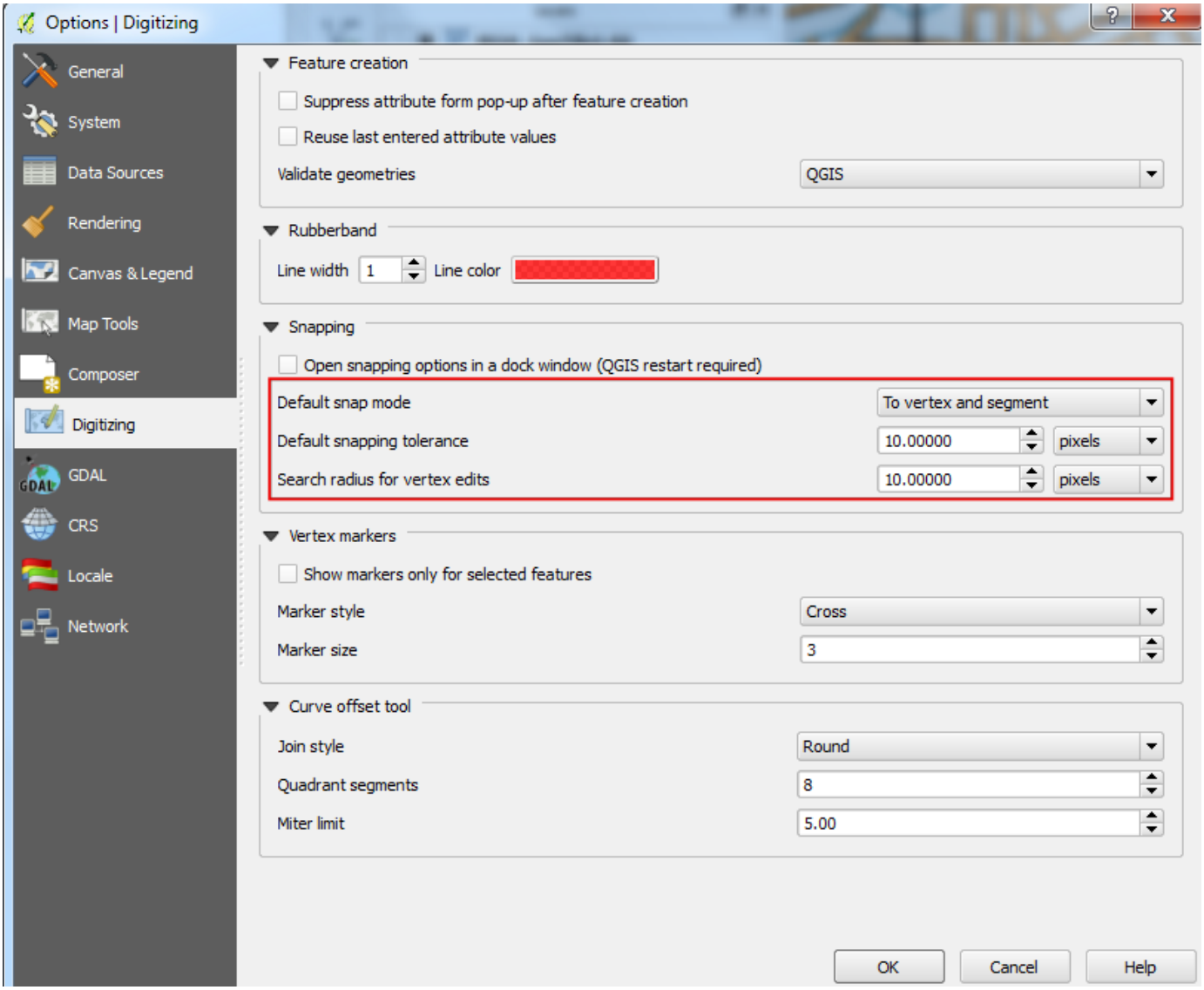
4. Back in the main QGIS window, use the Zoom tool to locate *Hagley Park* area in Christchurch. This is the park that we will be digitizing.



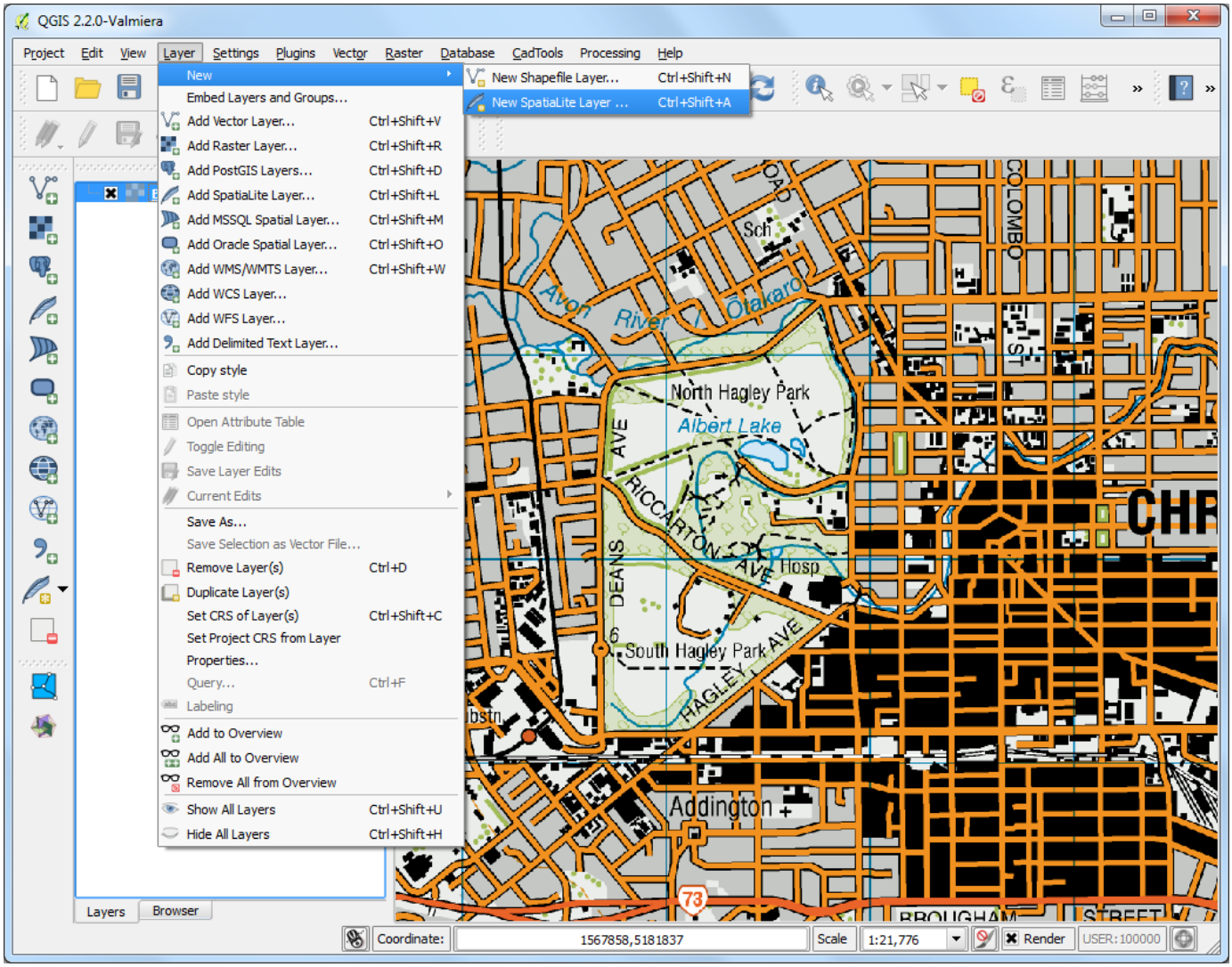
5. Before we start, we need to set default **Digitizing Options**. Go to Settings ‣ Options....



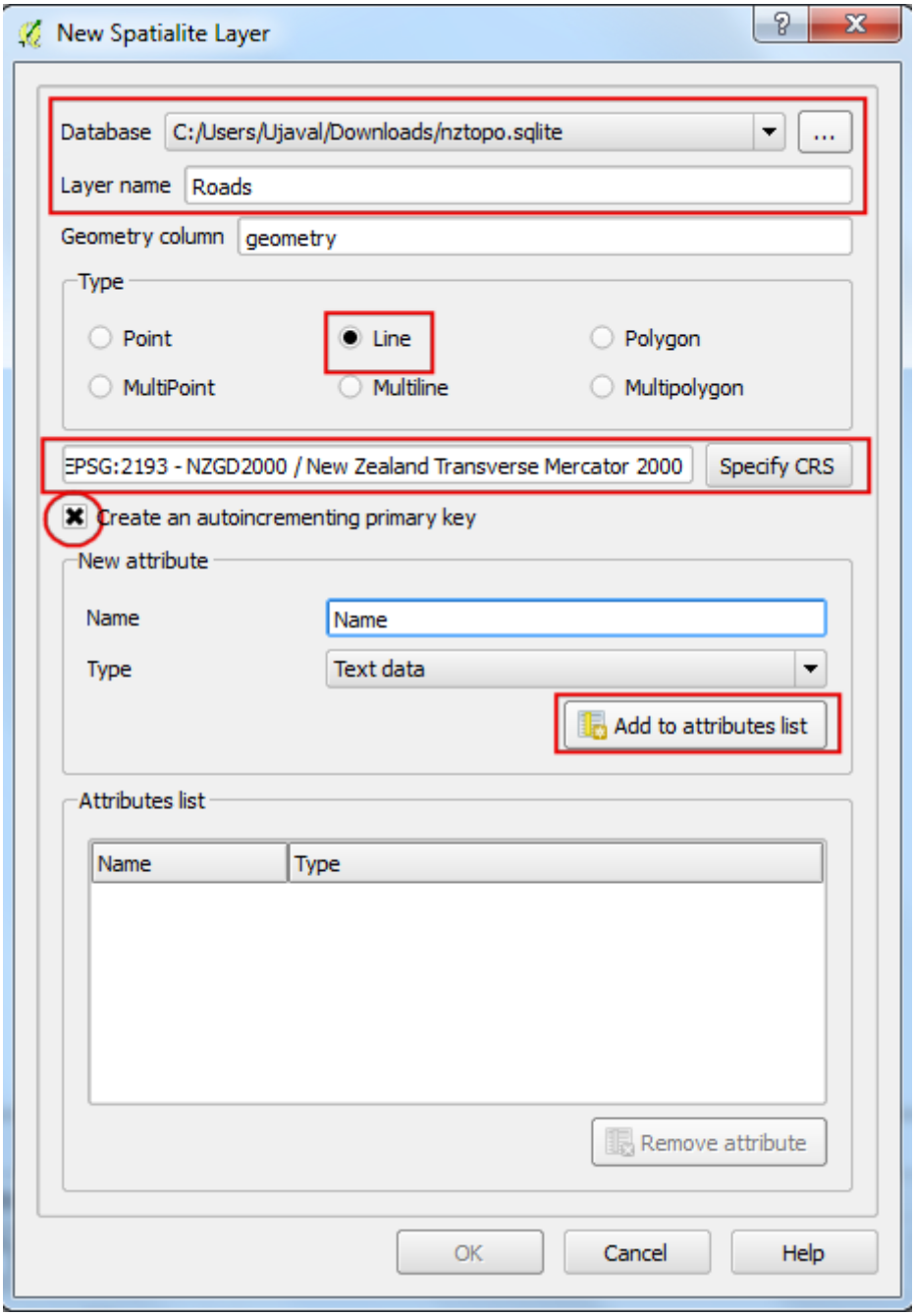
6. Select the Digitizing tab in the Options dialog. Set the Default snap mode to To vertex and segment. This will allow you to snap to the nearest vertex or line segment. I also prefer to set the Default snapping tolerance and Search radius for vertex edits in pixels instead of map units. This will ensure that the snapping distance remains constant regardless of zoom level. Depending on your computer screen resolution, you may choose an appropriate value. Click OK.



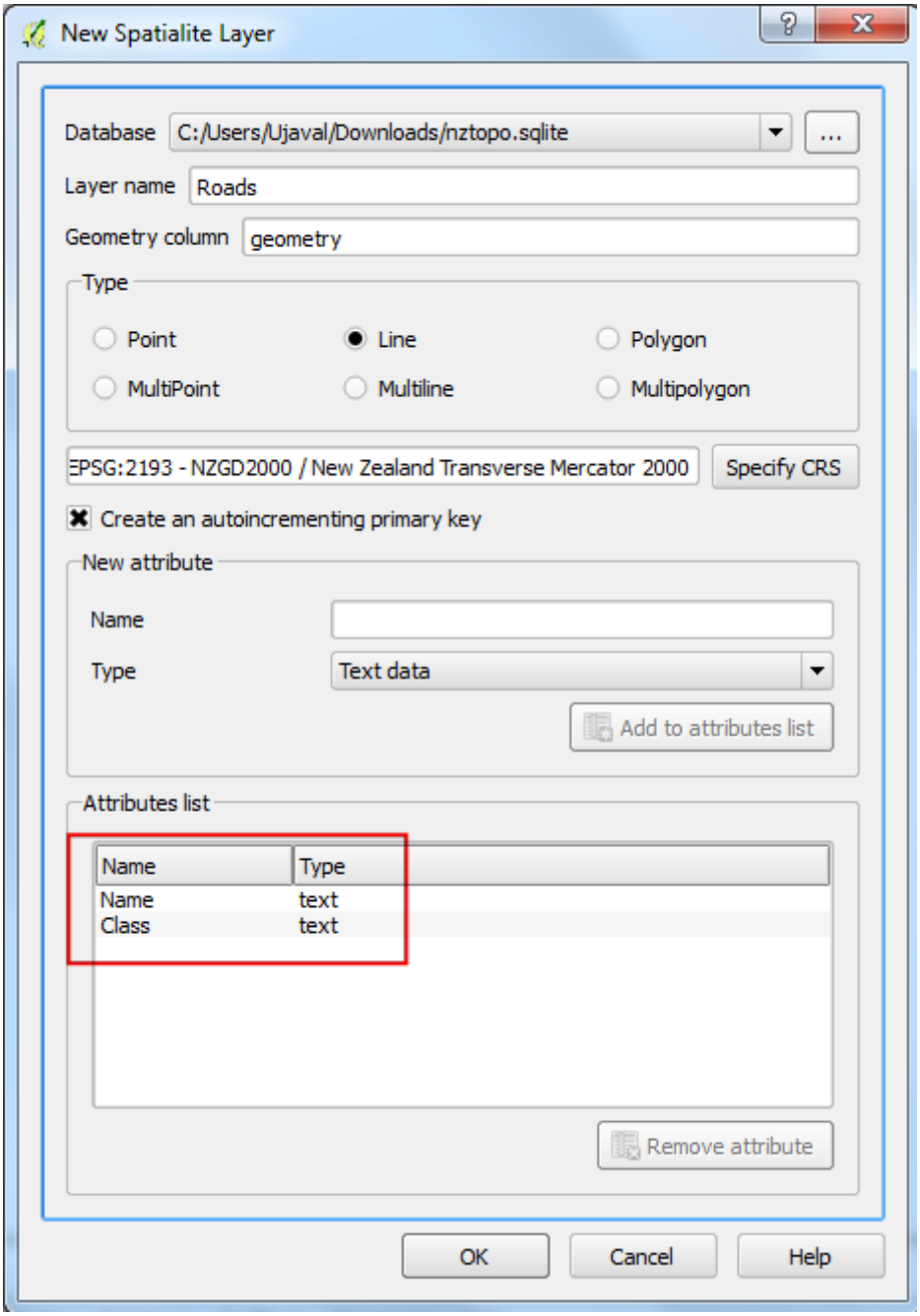
7. Now we are ready to start digitizing. We will first create a roads layer and digitize the roads around the park area. Select Layer ▸ New ▸ New Spatialite Layer.... You may also choose to create a New Shapefile Layer... instead if you prefer. Spatialite is an open database format similar to ESRI's geodatabase format. Spatialite database is contained within a single file on your hard drive and can contain diferent types of spatial (point, line, polygon) as well as non-spatial layers. This makes is much easier to move it around instead of a bunch of shapefiles. In this tutorial, we are creating a couple of polygon layers and a line layer, so a Spatialite database will be better suited. You can always load a spatialite layer and save it as a shapefile or any other format you want.



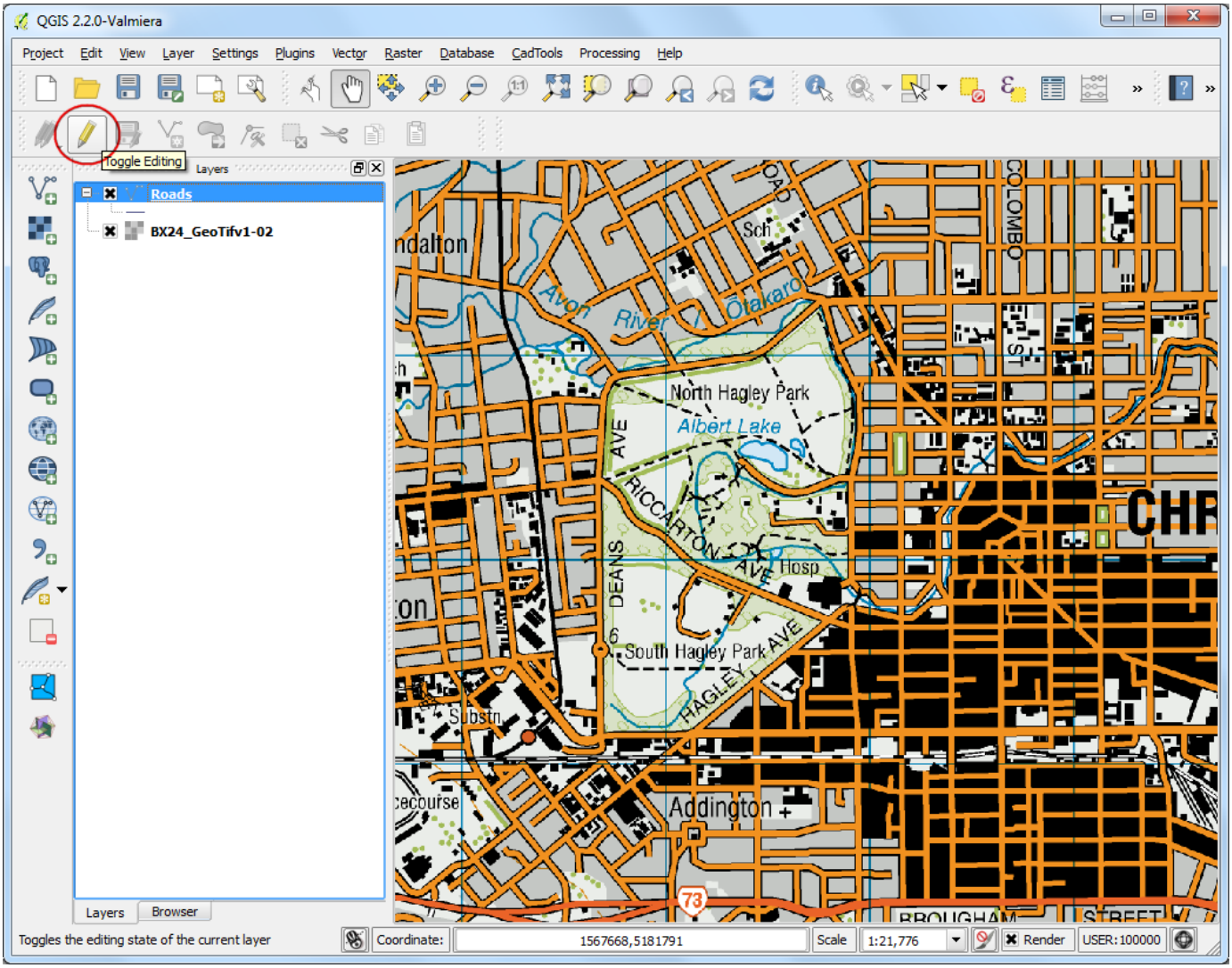
8. In the New Spatialite Layer dialog, click the ... button and save a new spatialite database named `nztopo.sqlite`. Choose the Layer name as `Roads` and select `Line` as the Type. The base topographic map is in the `EPSG:2193 - NZGD 2000` CRS, so we can select the same for our roads layer. Check the Create an autoincrementing primary key box. This will create a field called **pkuid** in the attribute table and assign a unique numeric id automatically to each feature. When creating a GIS layer, you must decide on the attributes that each feature will have. Since this is a roads layer, we will have 2 basic attributes - Name and Class. Enter `Name` as the Name of the attribute in the New attribute section and click Add to attribute list.



9. Similarly create a new attribute `Class` of the type Text data. Click OK.



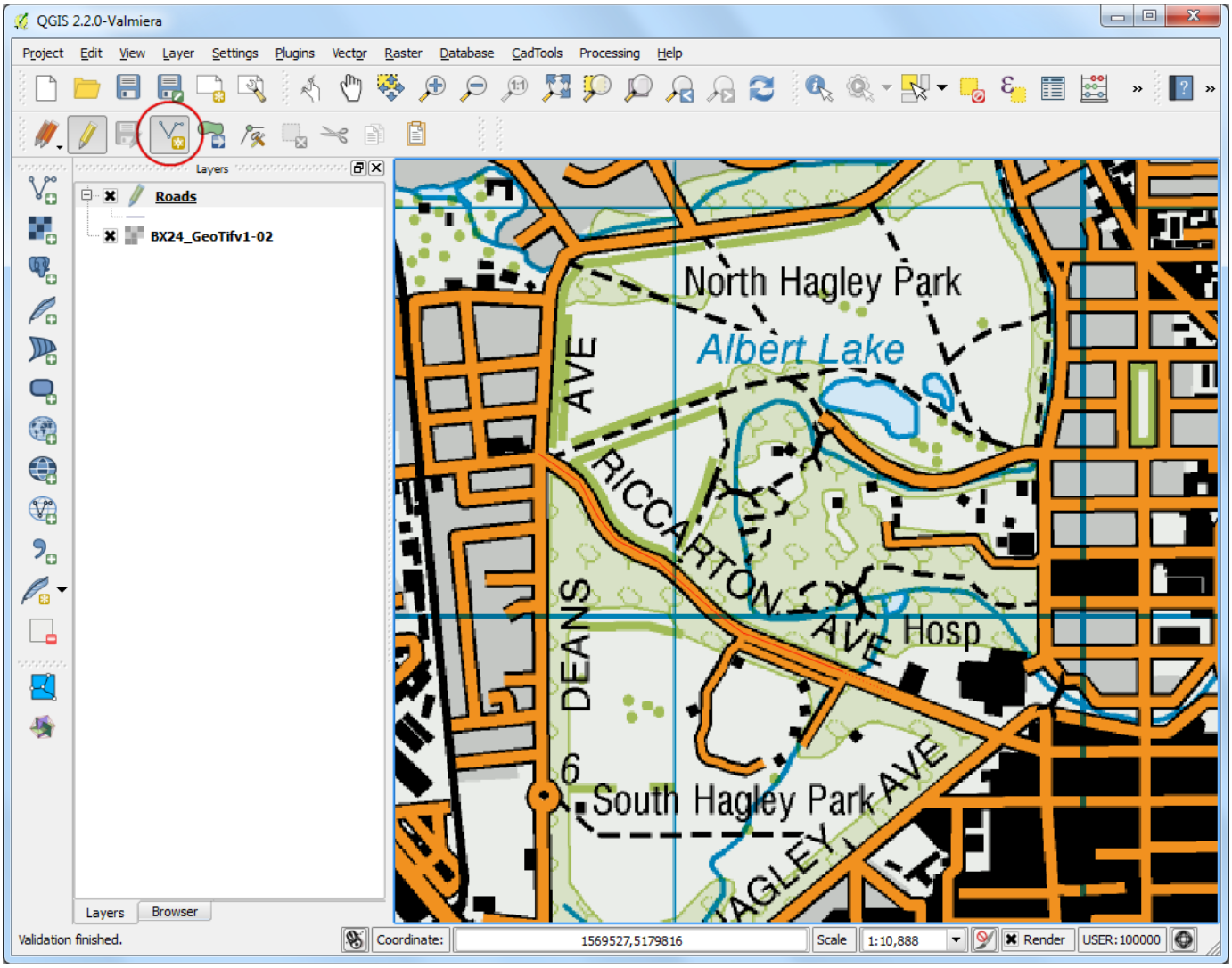
10. Once the layer is loaded, click the Toggle Editing button to put the layer in editing mode.



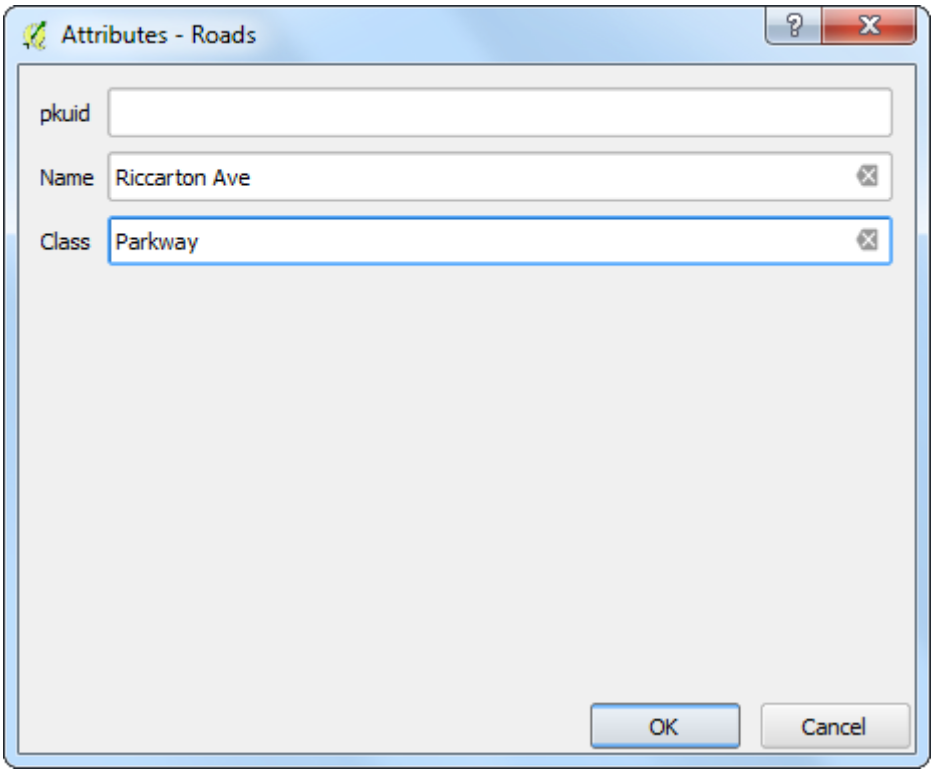
11. Click the Add feature button. Click on the map canvas to add a new vertex. Add new vertices along the road feature. Once you have digitized a road segment, right-click to end the feature.

Note

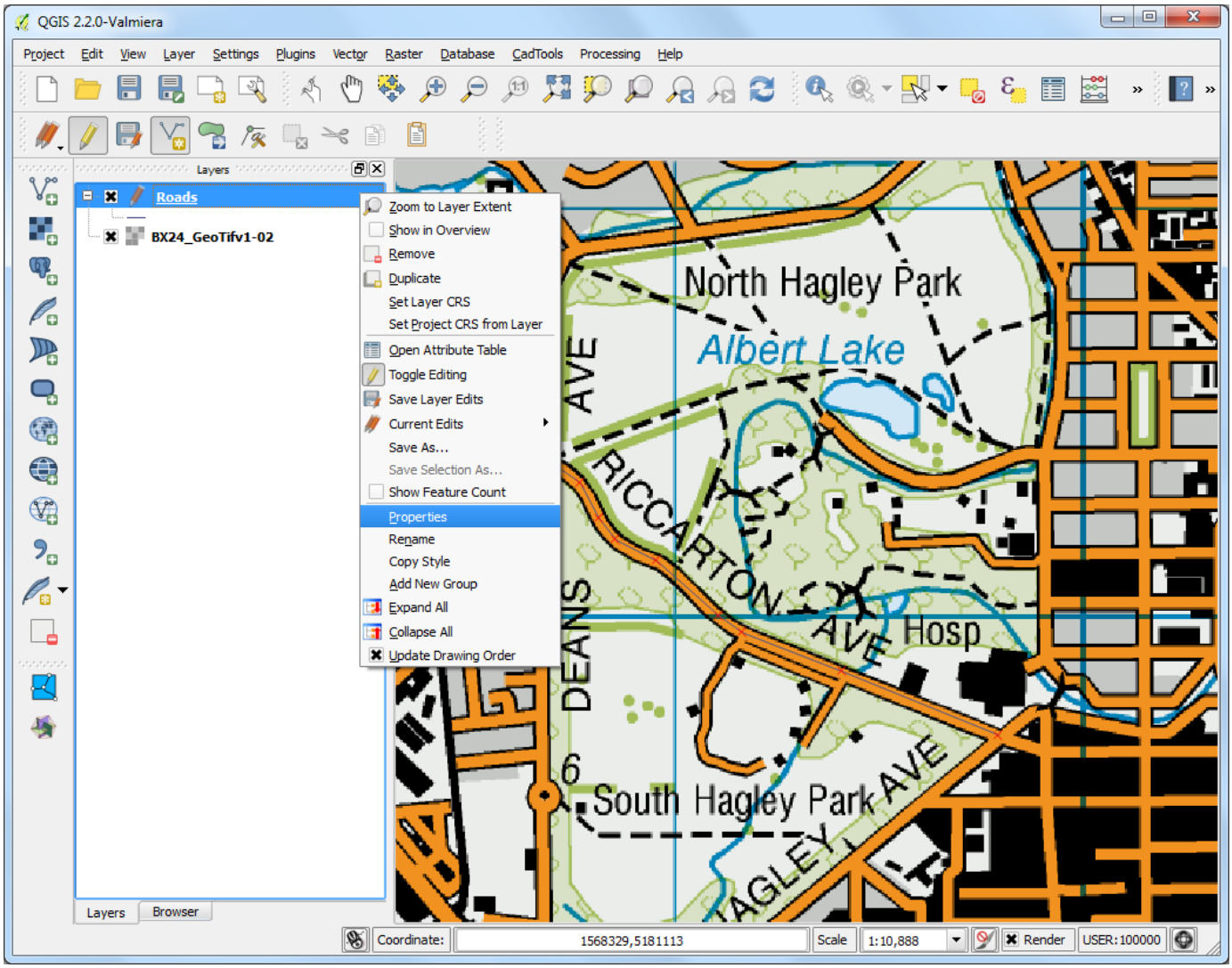
You can use the scroll wheel of the mouse to zoom in or out while digitizing. You can also hold the scroll button and move the mouse to pan around.



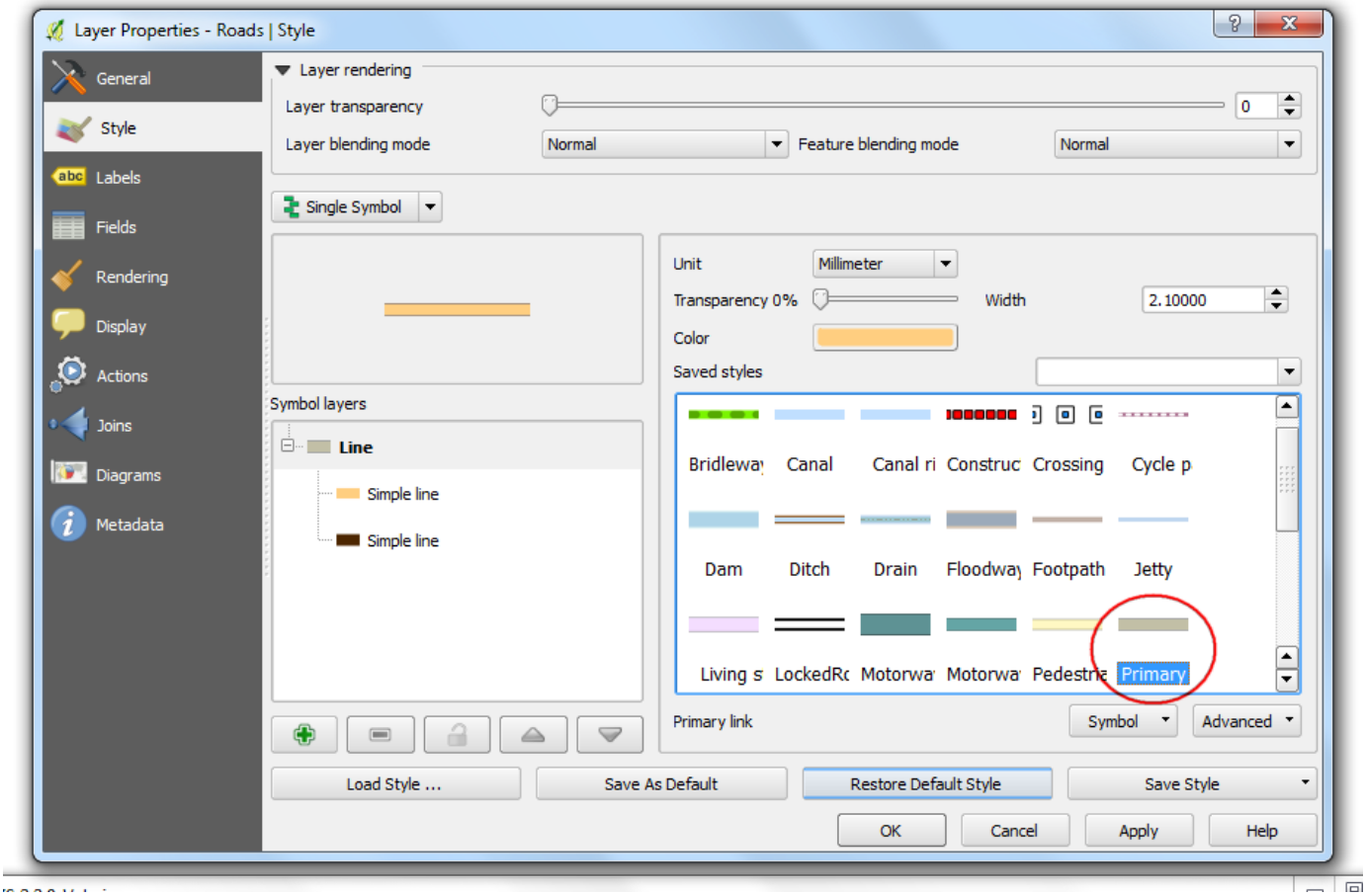
12. After you right-click to end the feature, you will get a pop-up dialog called Attributes. Here you can enter attributes of the newly created feature. Since the **pkuid** is an auto-incrementing field, you will not be able to enter a value manually. Leave it blank and enter the road name as it appears on the topo map. Optionally, assign a Road Class value as well. Click OK.



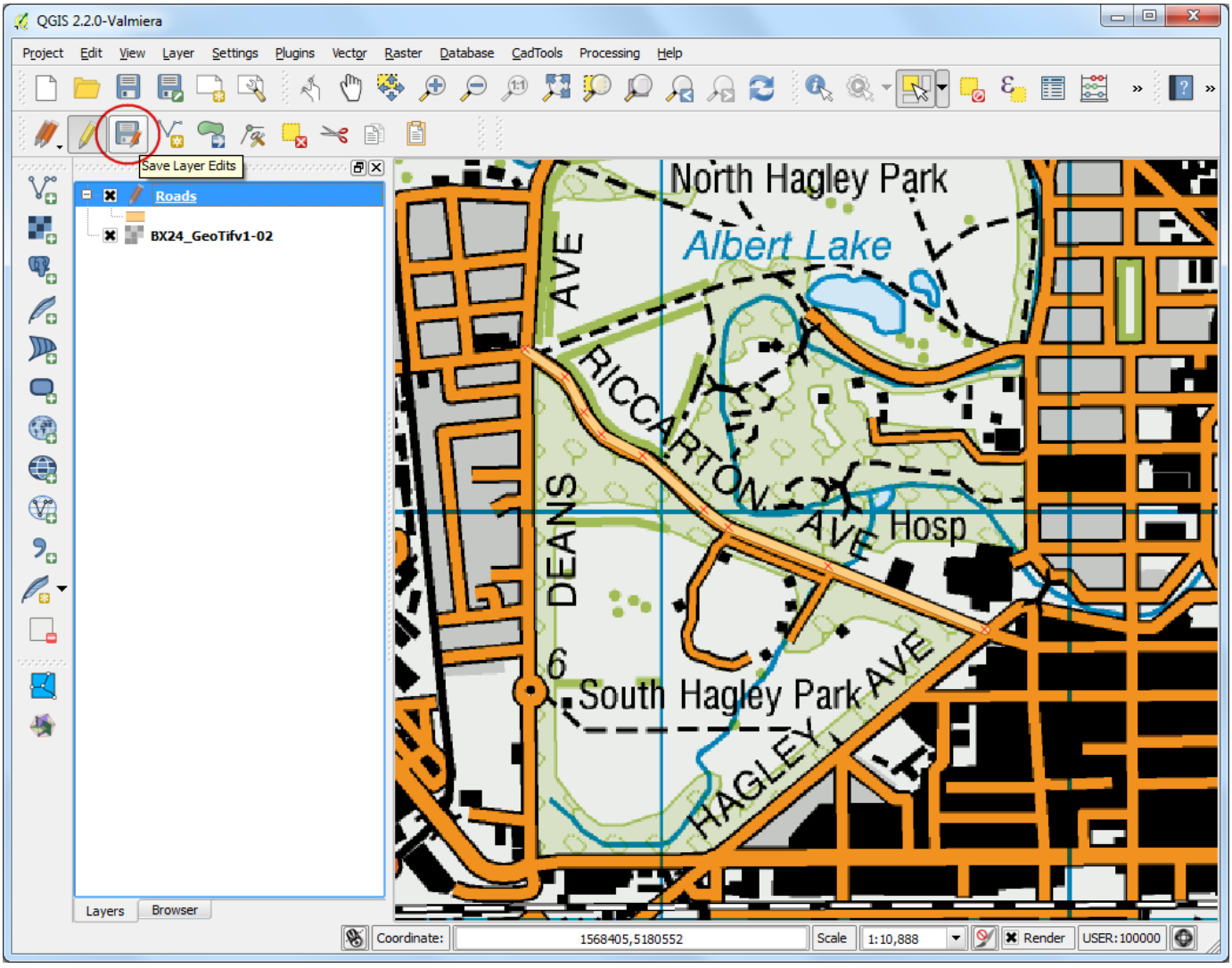
13. The default style of the new line layer is a thin line. Let's change it so we can better see the digitized features on the canvas. Right click the `Roads` layer and select Properties.



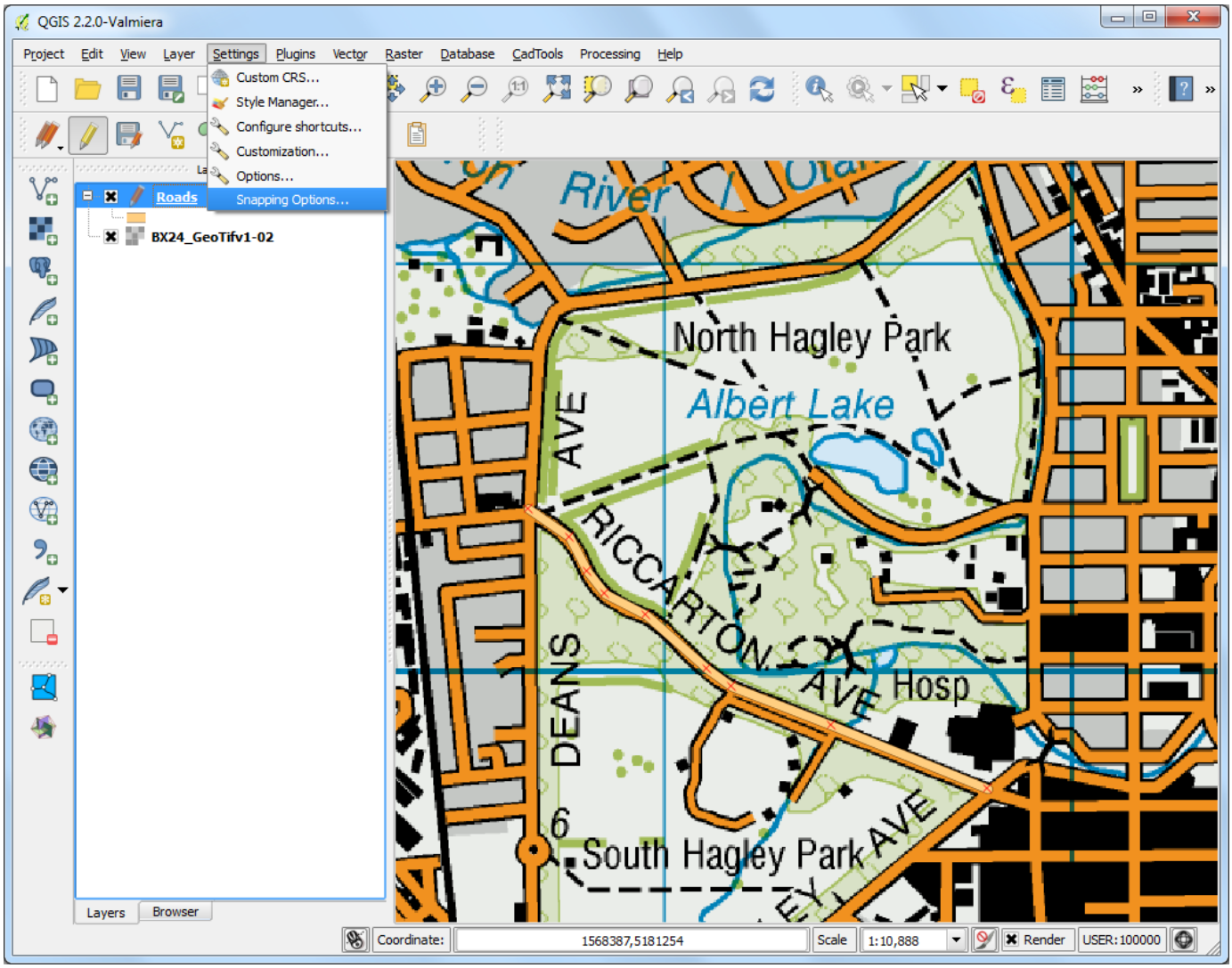
14. Select the Style tab in the Layer Properties dialog. Choose a thicker line style such as Primary from the predefined styles. Click OK.



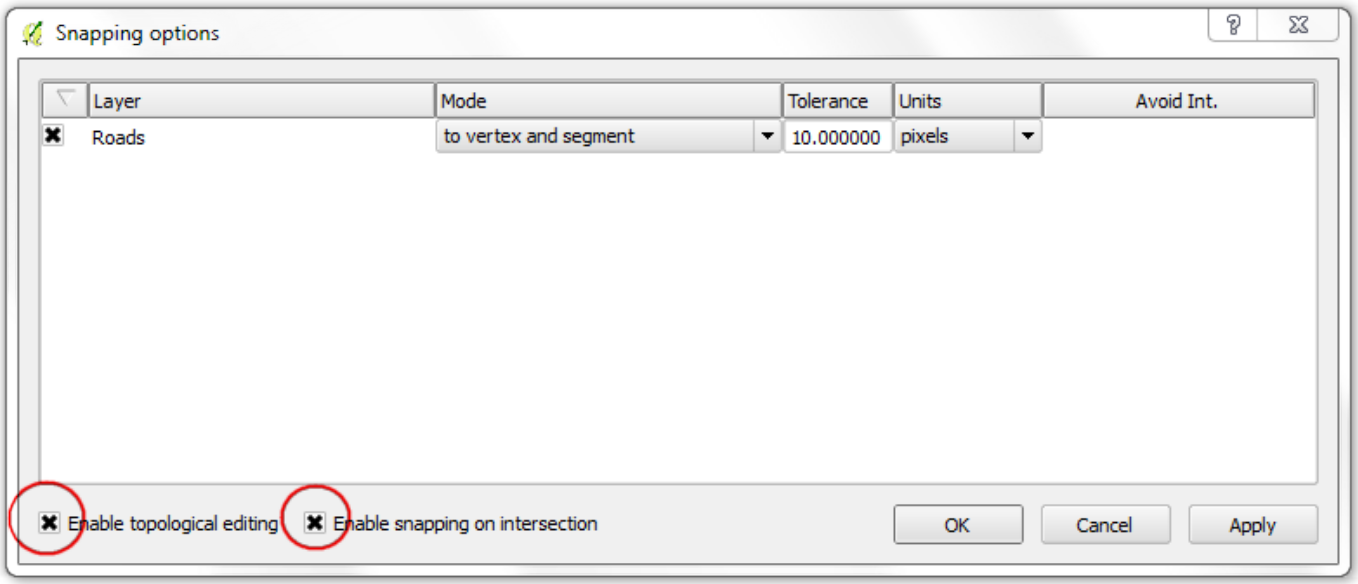
15. Now you will see the digitized road feature clearly. Click Save Layer Edits to commit the new feature to disk.



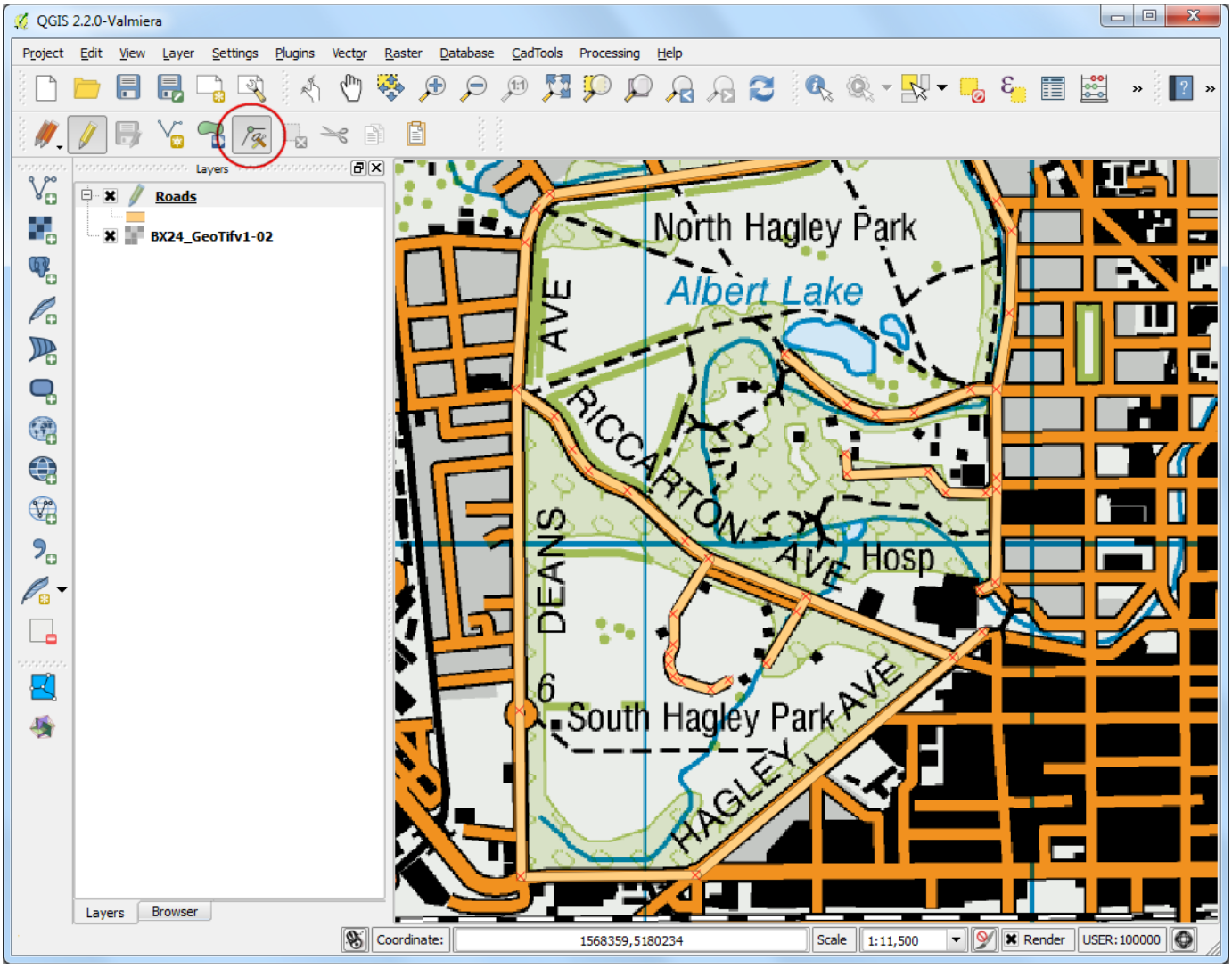
16. Before we digitize remaining roads, it is important to update some other settings that are important to create an error free layer. Go to Settings ‣ Snapping Options....



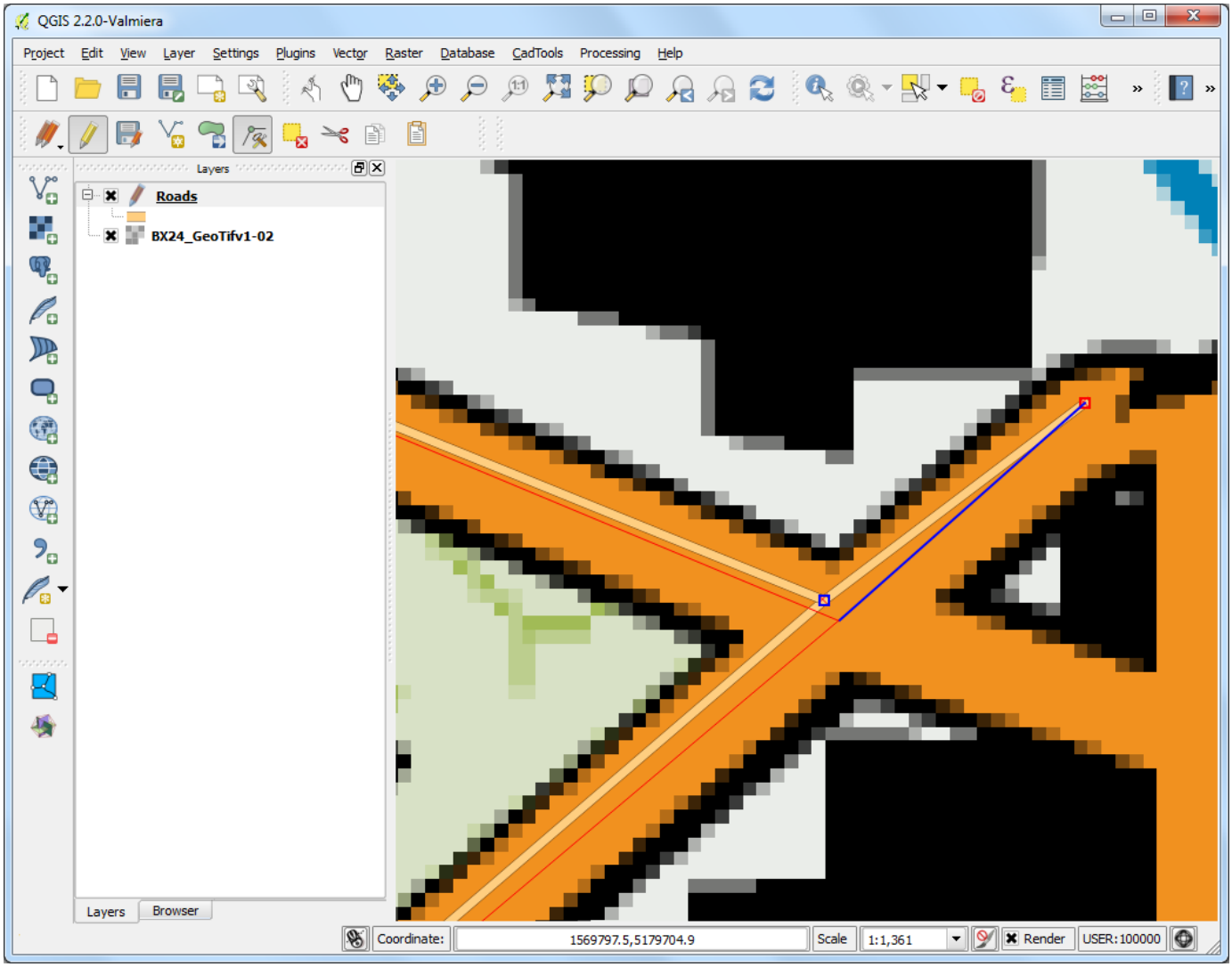
17. In the Snapping Options dialog, check the Enable topological editing. This option will ensure that the common boundaries are maintained correctly in polygon layers. Also check the Enable snapping on intersection which allows you to snap on an intersection of a background layer.



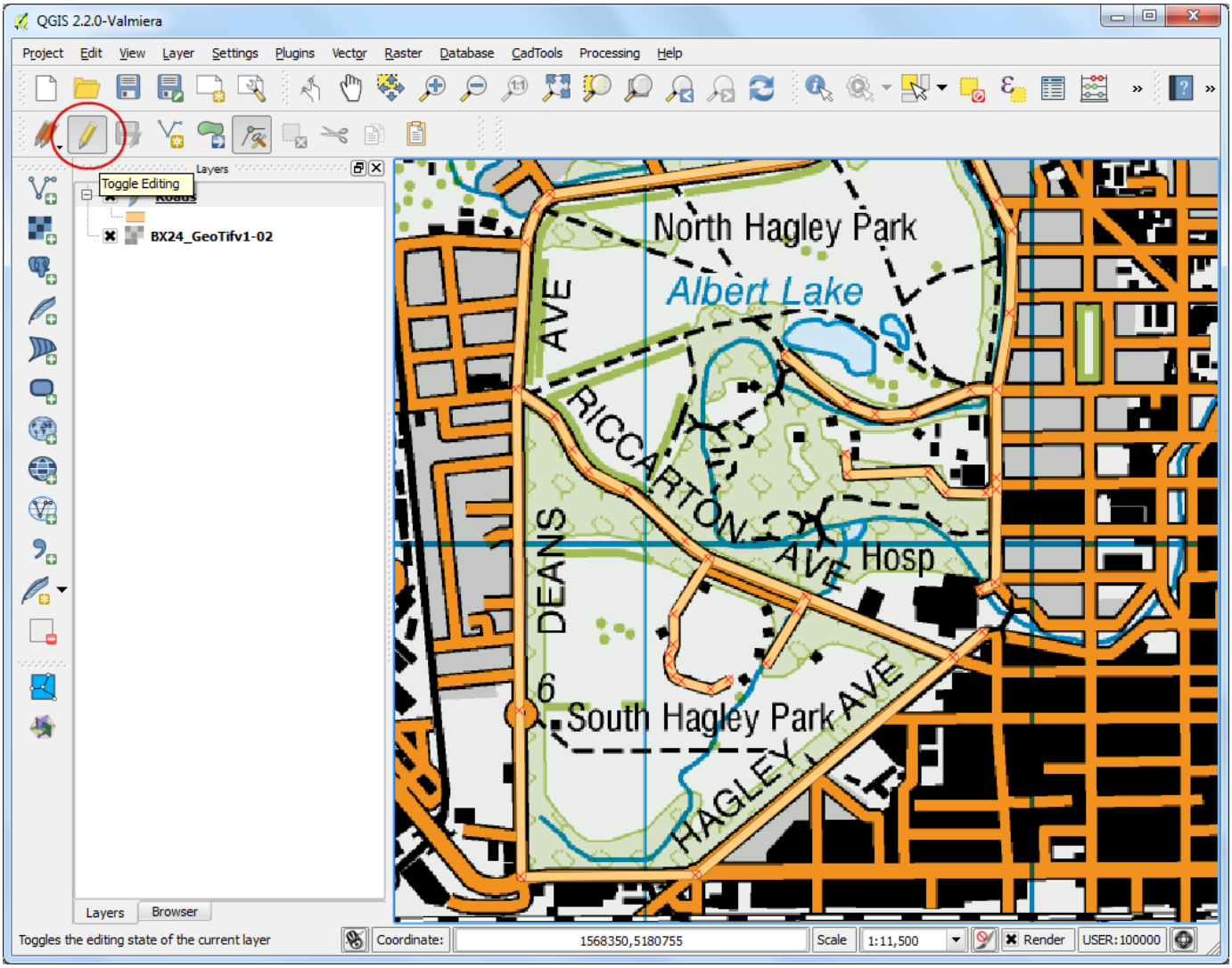
18. Now you can click Add feature button and digitize other roads around the park. Make sure to click Save Edits after you add a new feaure to save your work. A useful tool to help you with digitizing is the **Node Tool**. Click the Node Tool button.



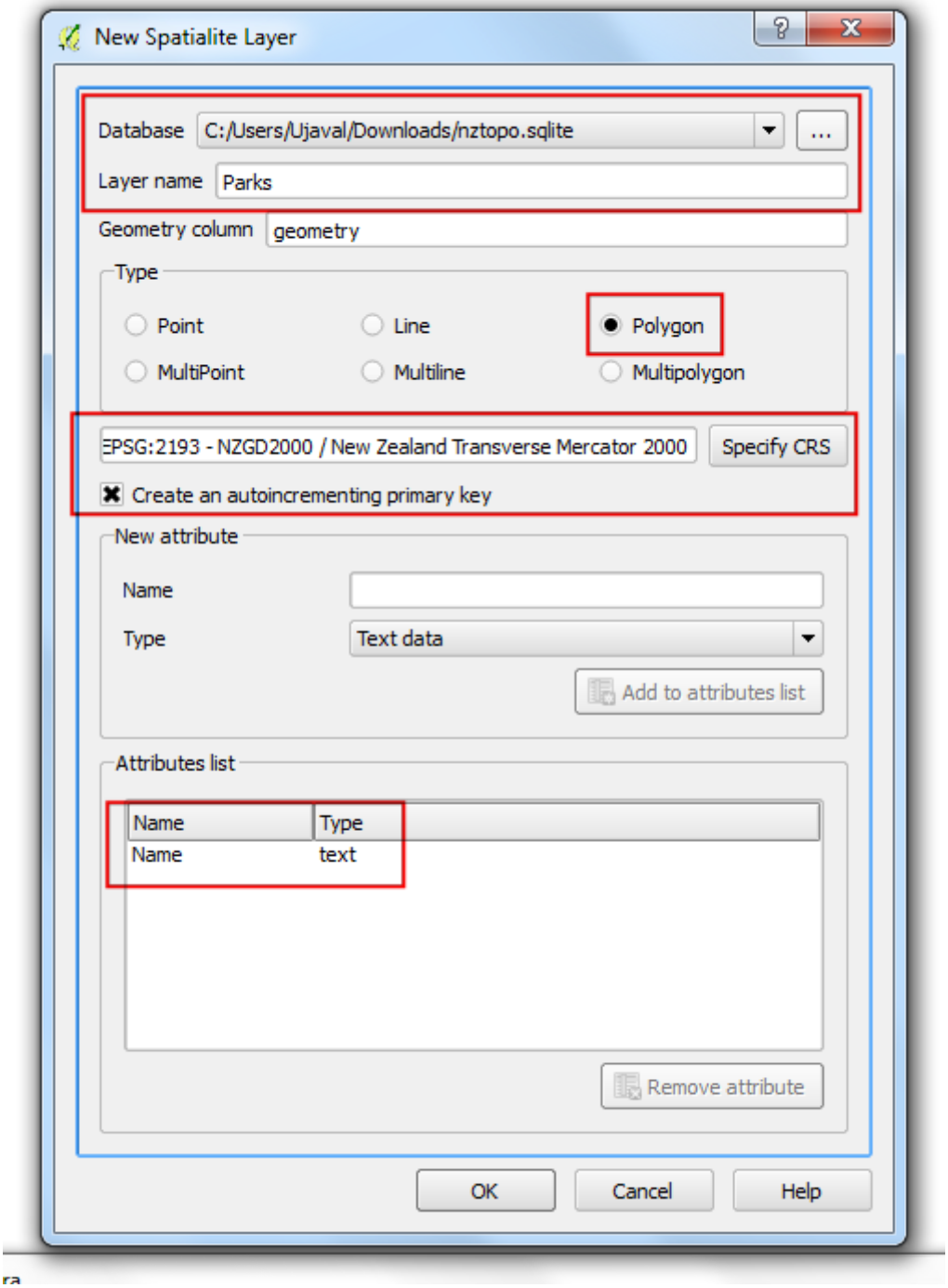
19. Once the node tool is activated, click on any feature to show the vertices. Click on any vertex to select it. The vertex will change the color once it is selected. Now you can click and drag your mouse to move the vertex. This is useful when you want to make adjustments after the feature is created. You can also delete a selected vertex by clicking the `Delete` key. (`Option+Delete` on a mac)



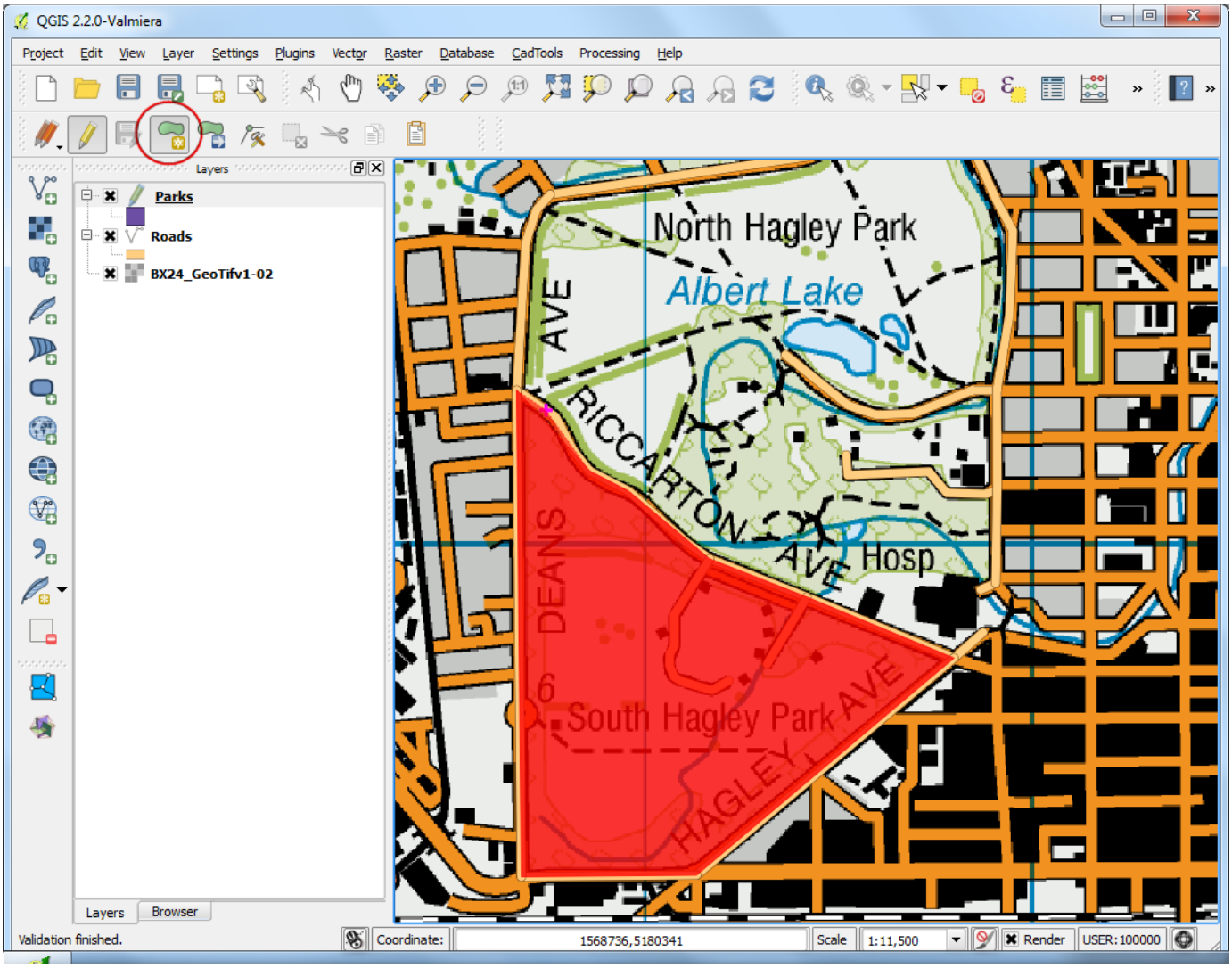
20. Once you have finished digitizing all the roads, click the Toggle Editing button.



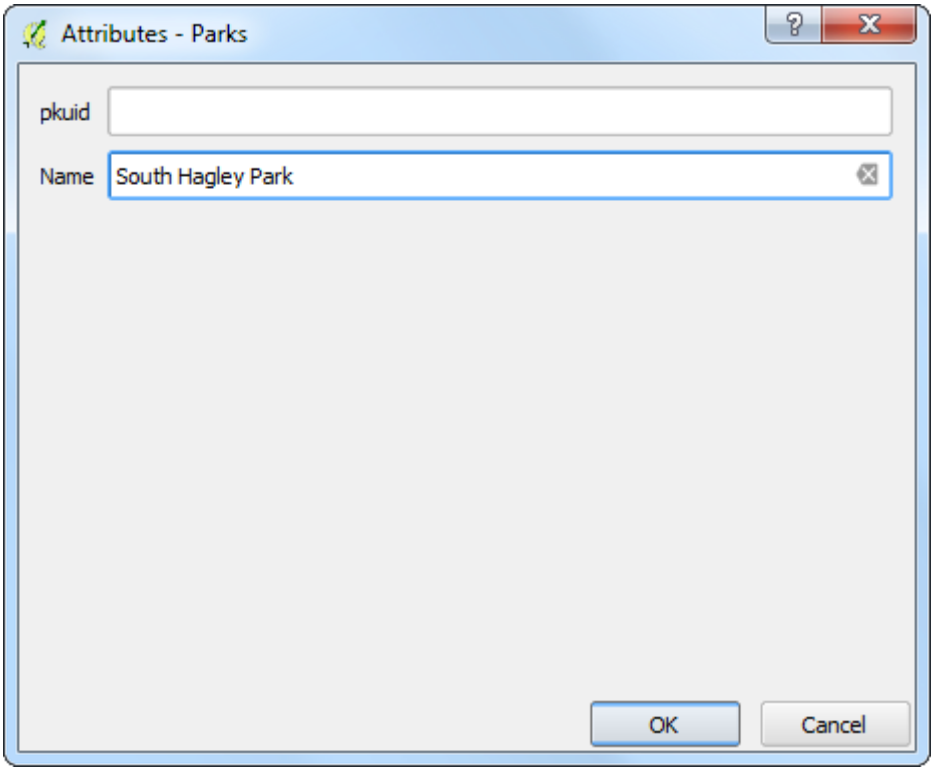
21. Now we will create a polygon layer representing the park boundaries. Go to Layer ▸ New ▸ New Spatialite Layer.... Select the `nztopo.sqlite` database from the dropdown list. Name the new layer as `Parks`. Select `Polygon` as the Type. Create a new attribute called `Name`. Click OK.



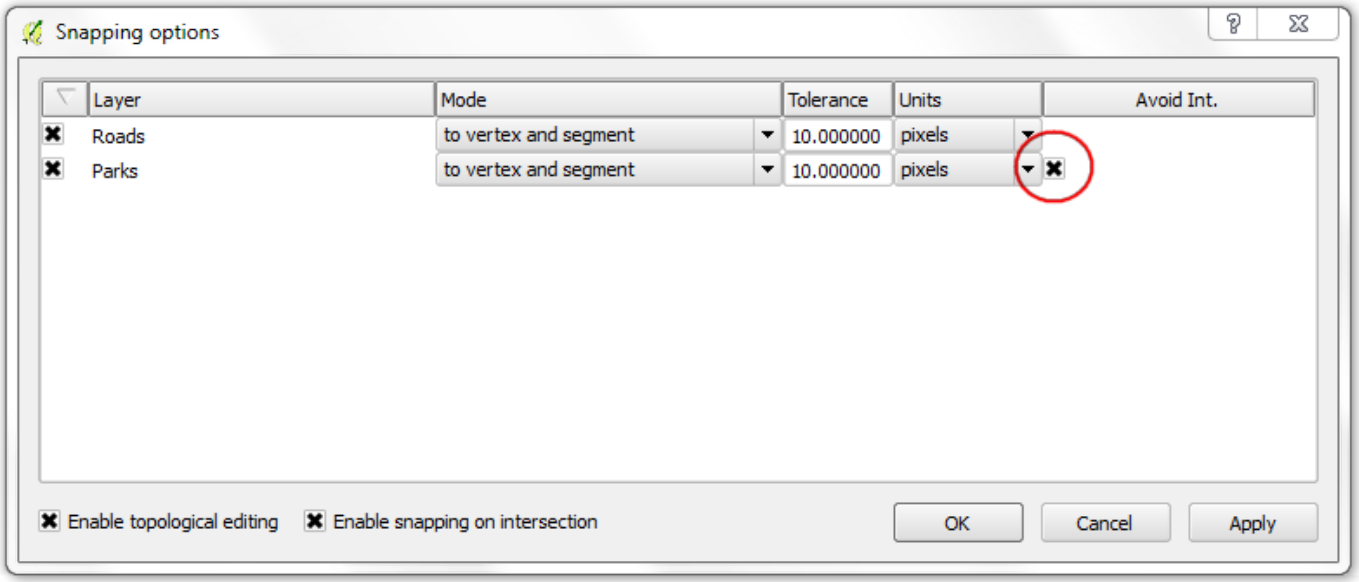
22. Click the Add feature button and click on the map canvas to add a polygon vertex. Digitize the polygon representing the park. Make sure you snap to the roads vertices so there are no gaps between the park polygons and road lines. Right-click to finish the polygon.



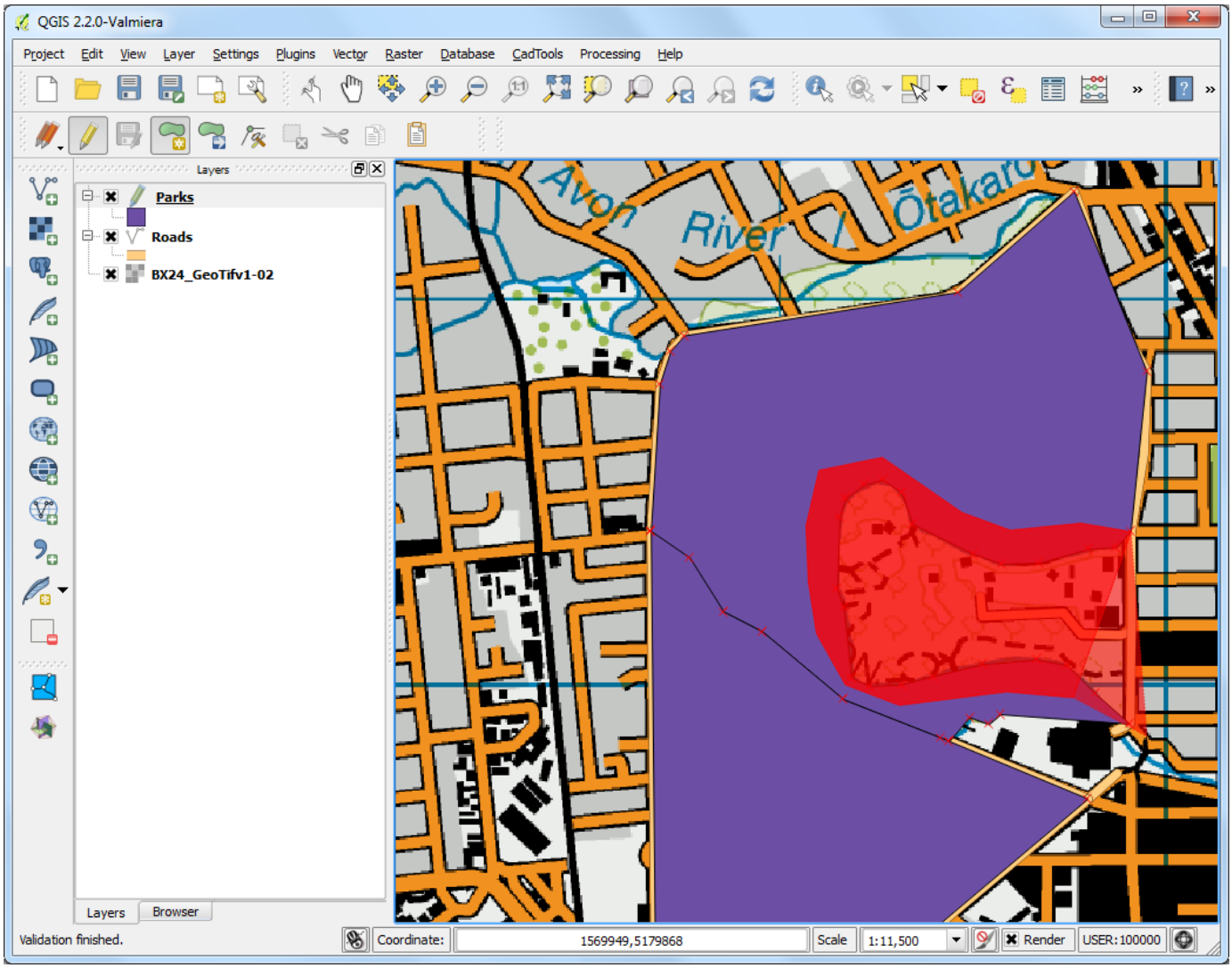
23. Enter the park name in the Attributes pop-up.



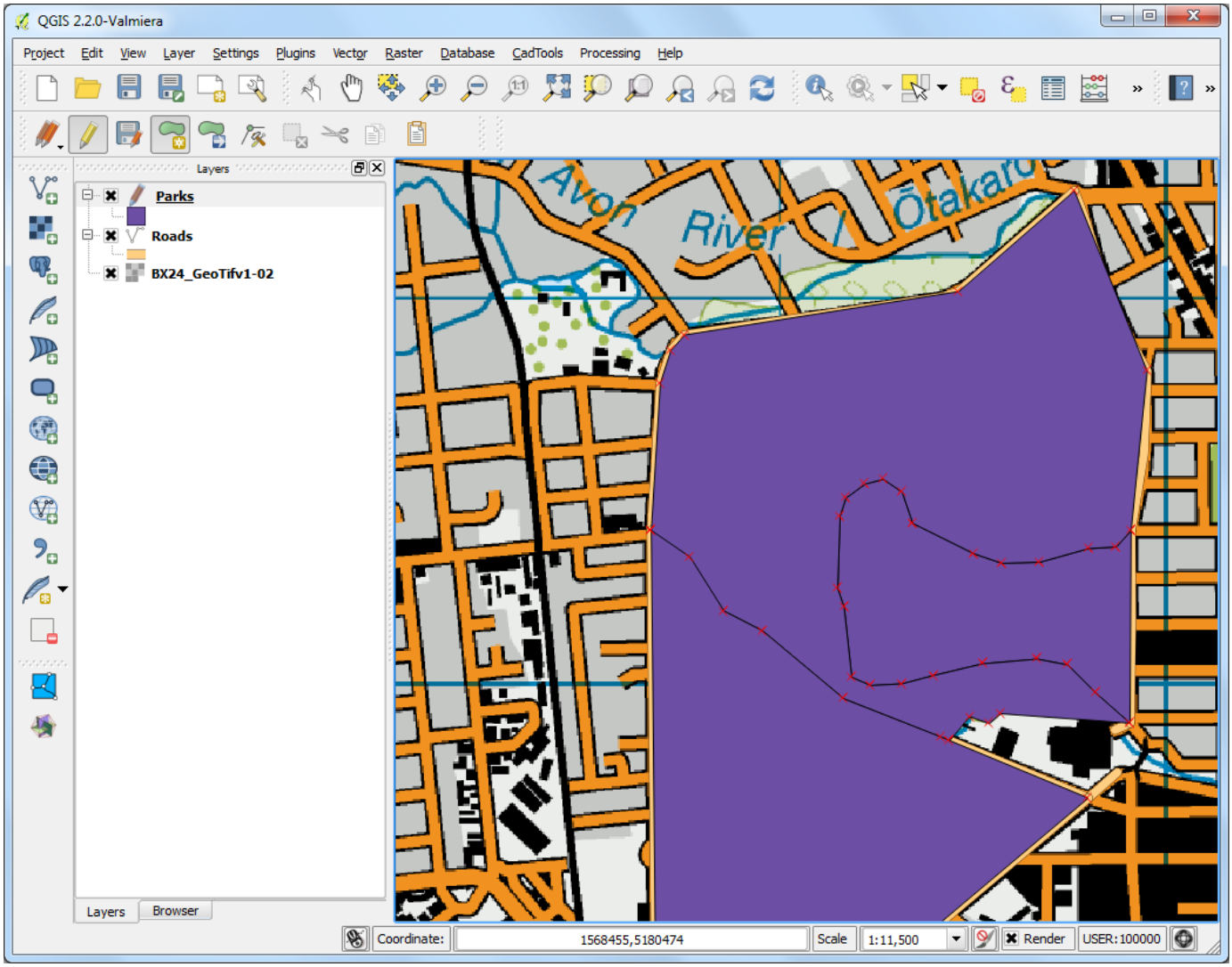
24. Polygon layers offer another very useful setting called **Avoid intersections of new polygons**. Go to Settings ▸ Snapping Options.... Check the box in the Avoid Int column in the row for the `Parks` layer. Click OK.



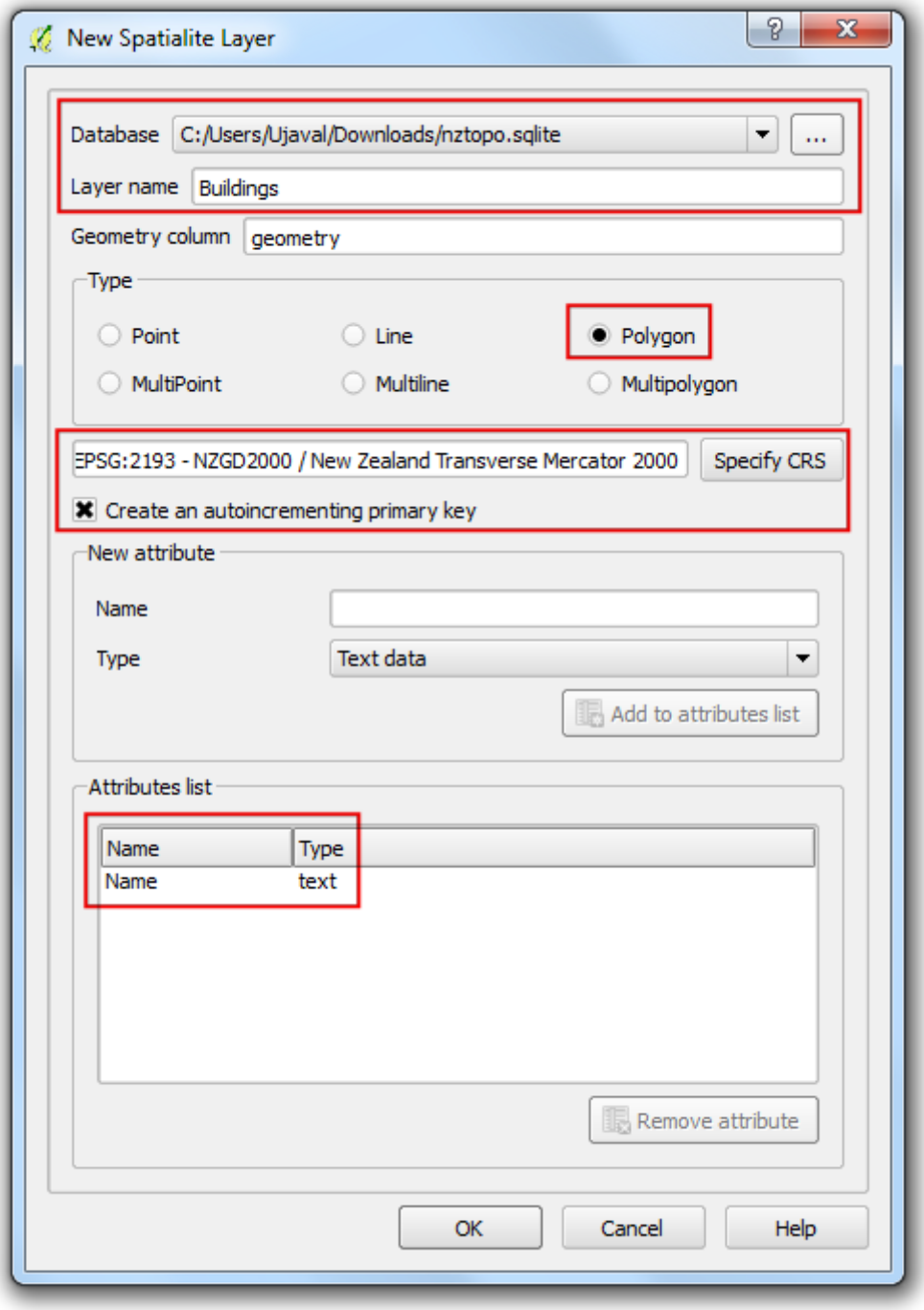
25. Now click on Add feature to add a polygon. With the **Avoid intersections of new polygons**, you will be able quickly digitize a new polygon without worrying about snapping exactly to the neighboring polygons.



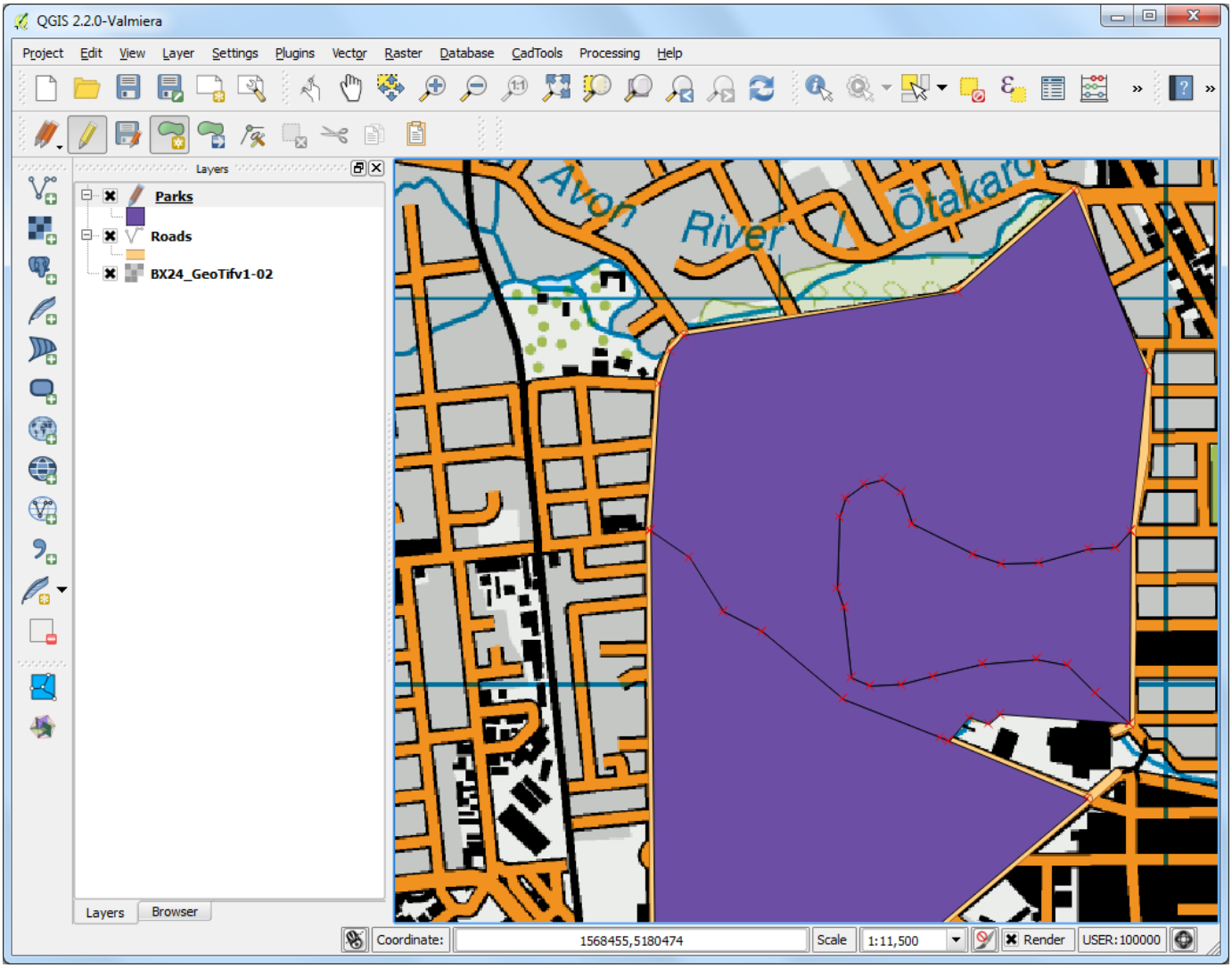
26. Right-click to finish the polygon and enter the attributes. Magically the new polygon is shrunk and snapped exactly to the boundary of the neighboring polygons! This is very useful when digitizing complex boundaries where you need not be very precise and still have topologically correct polygon. Click Toggle Editing to finish editing the `Parks` layer.



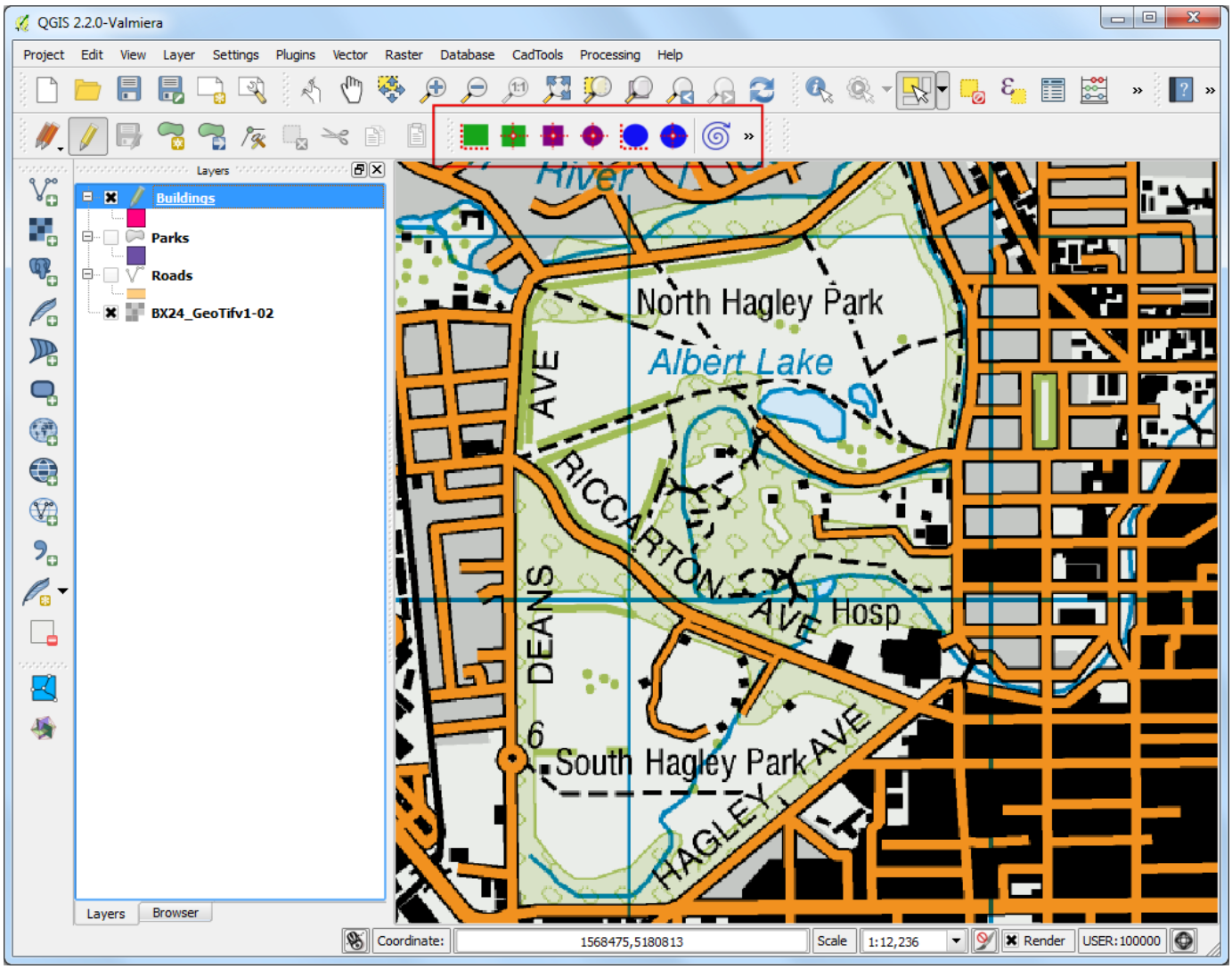
27. Now it is time to digitize a buildings layer. Create a new polygon layer named `Buildings` by going to Layer ▸ New ▸ New Spatialite Layer.



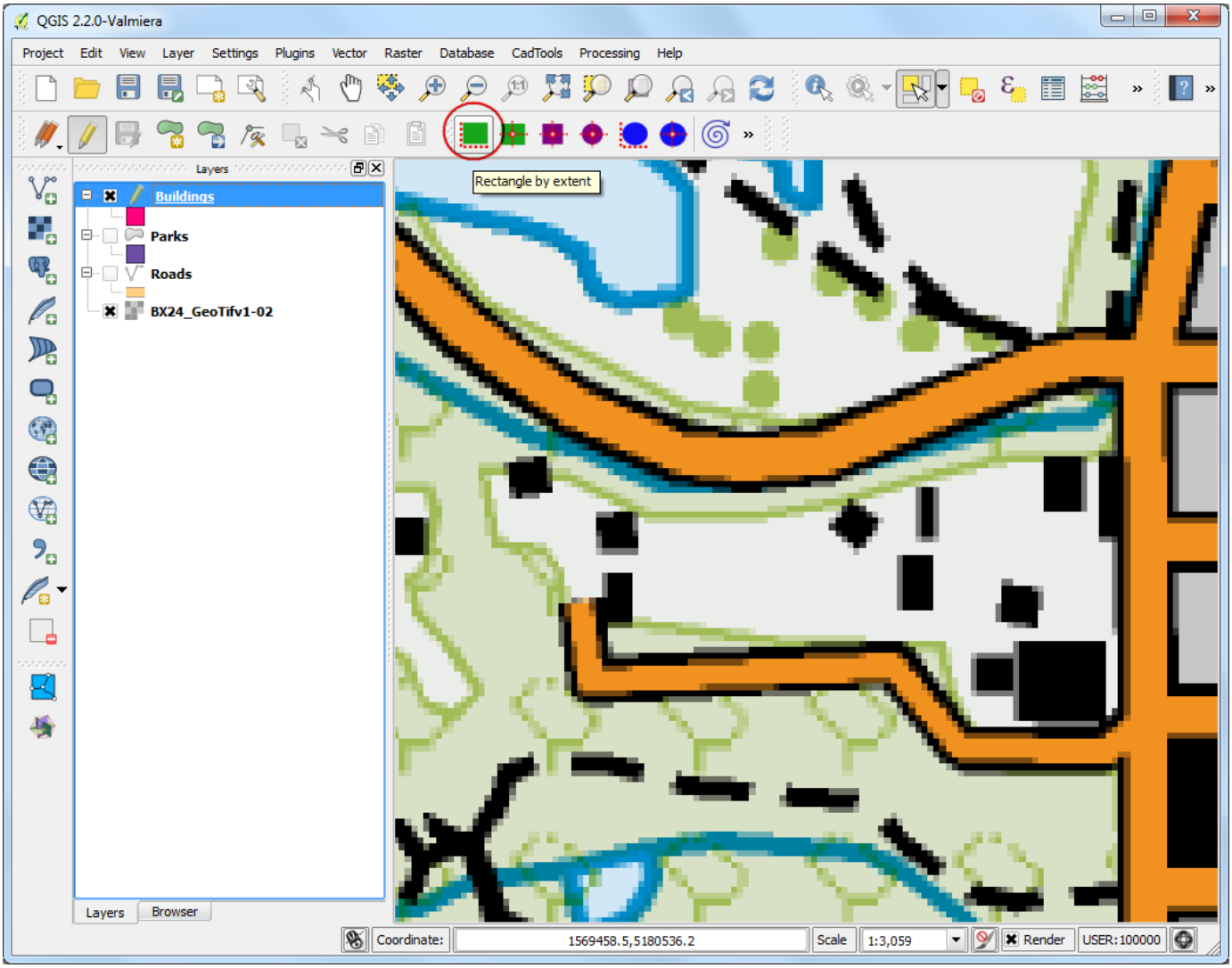
28. Once the `Buildings` layer is added, turn off the `Parks` and `Roads` layer so the base topo map is visible. Select the `Buildings` layer and click Toggle Editing.



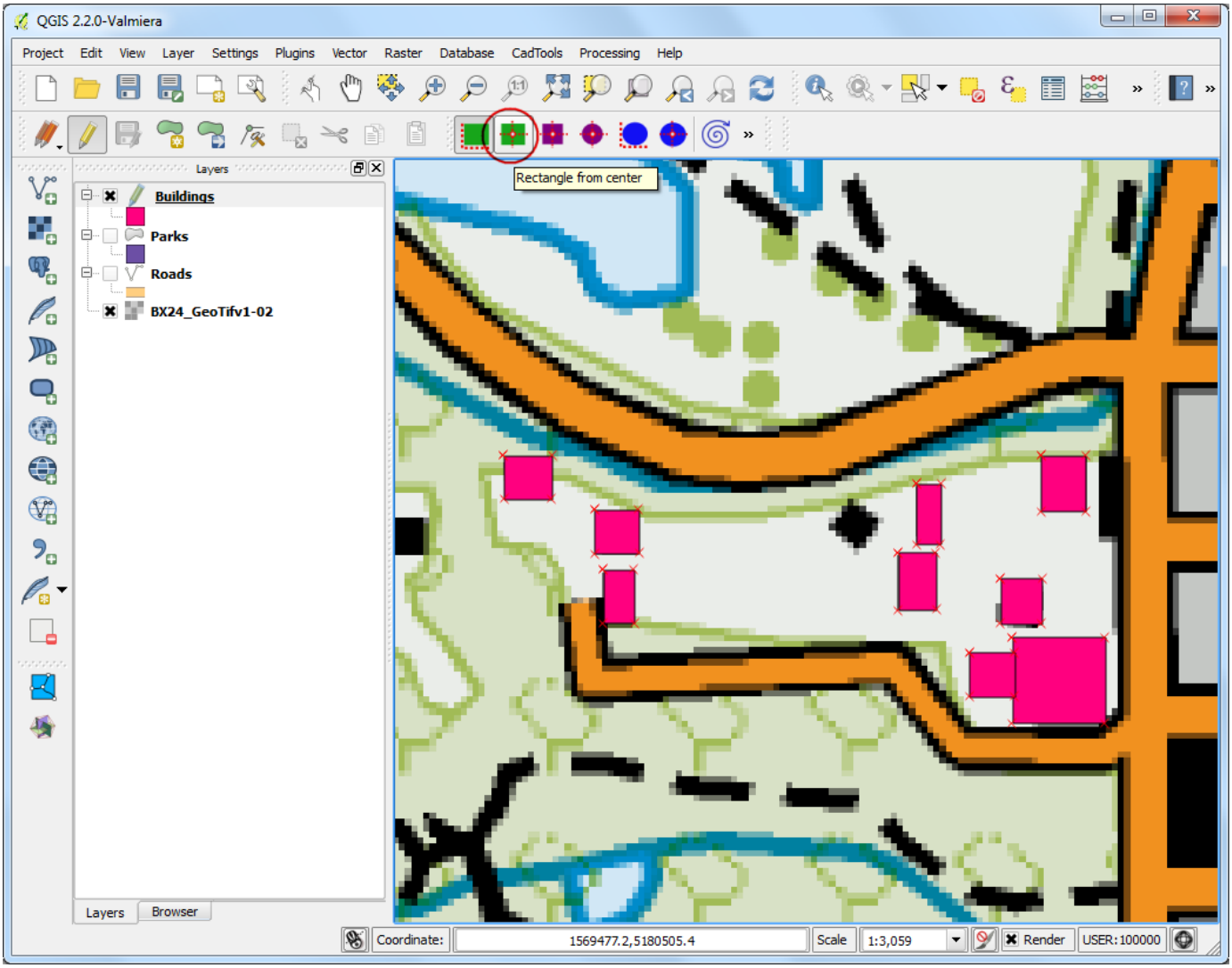
29. Digitizing buildings can be a cumbersome task. Also it is difficult to add vertices manually so that the edges are perpendicular and form a rectangle. We will use a plugin called **Rectangles Ovals Digitizing** to help with this task. See *Using Plugins* (using_plugins.html) to see how to search and install plugins. Once the **Rectangles Ovals Digitizing** plugin is installed, you will see a new toolbar appear above the canvas.



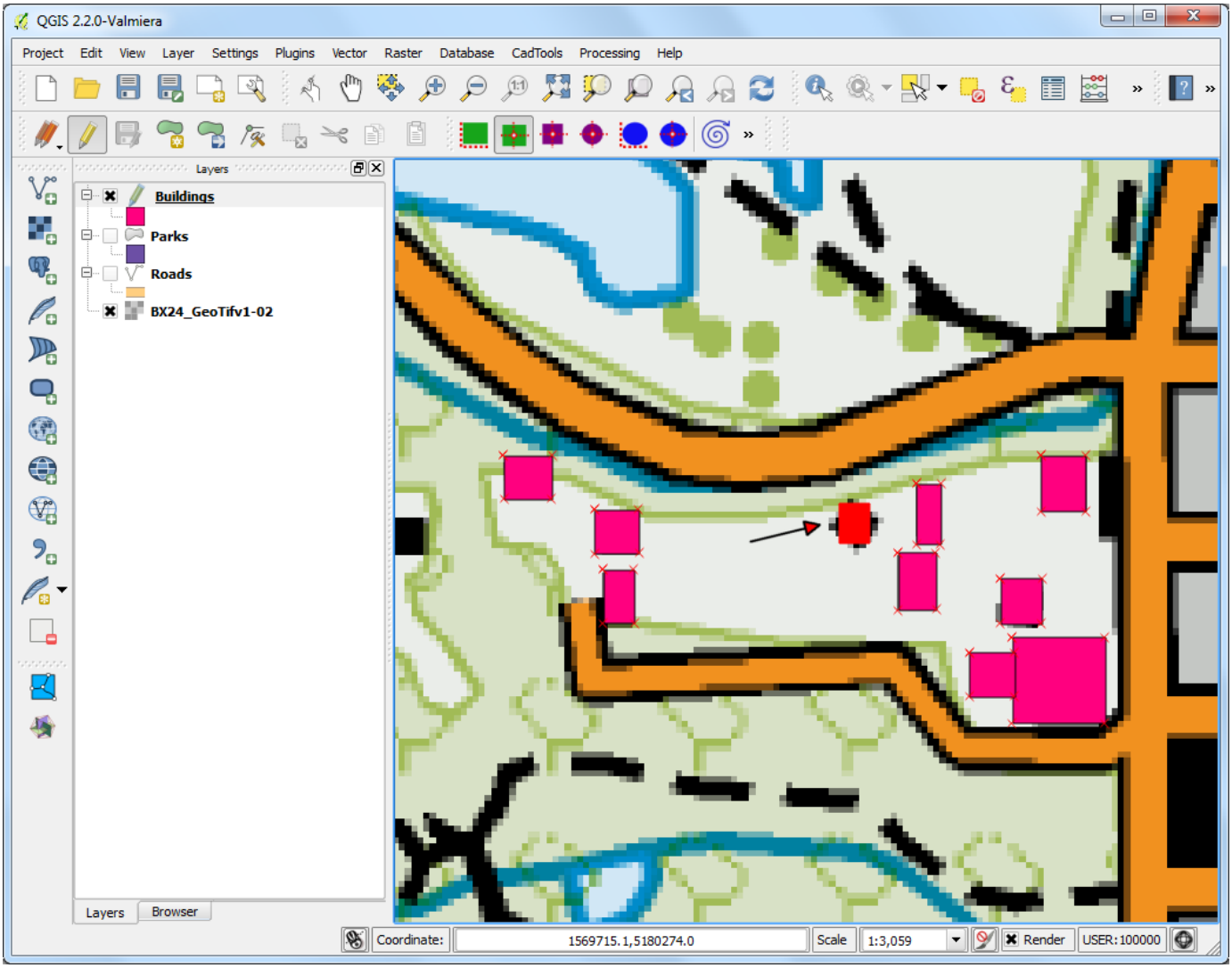
30. Zoom to an area with the buildings and click Rectangle by Extent button. Click and drag the mouse to draw a perfect rectangle. Similarly, add remaining buildings.



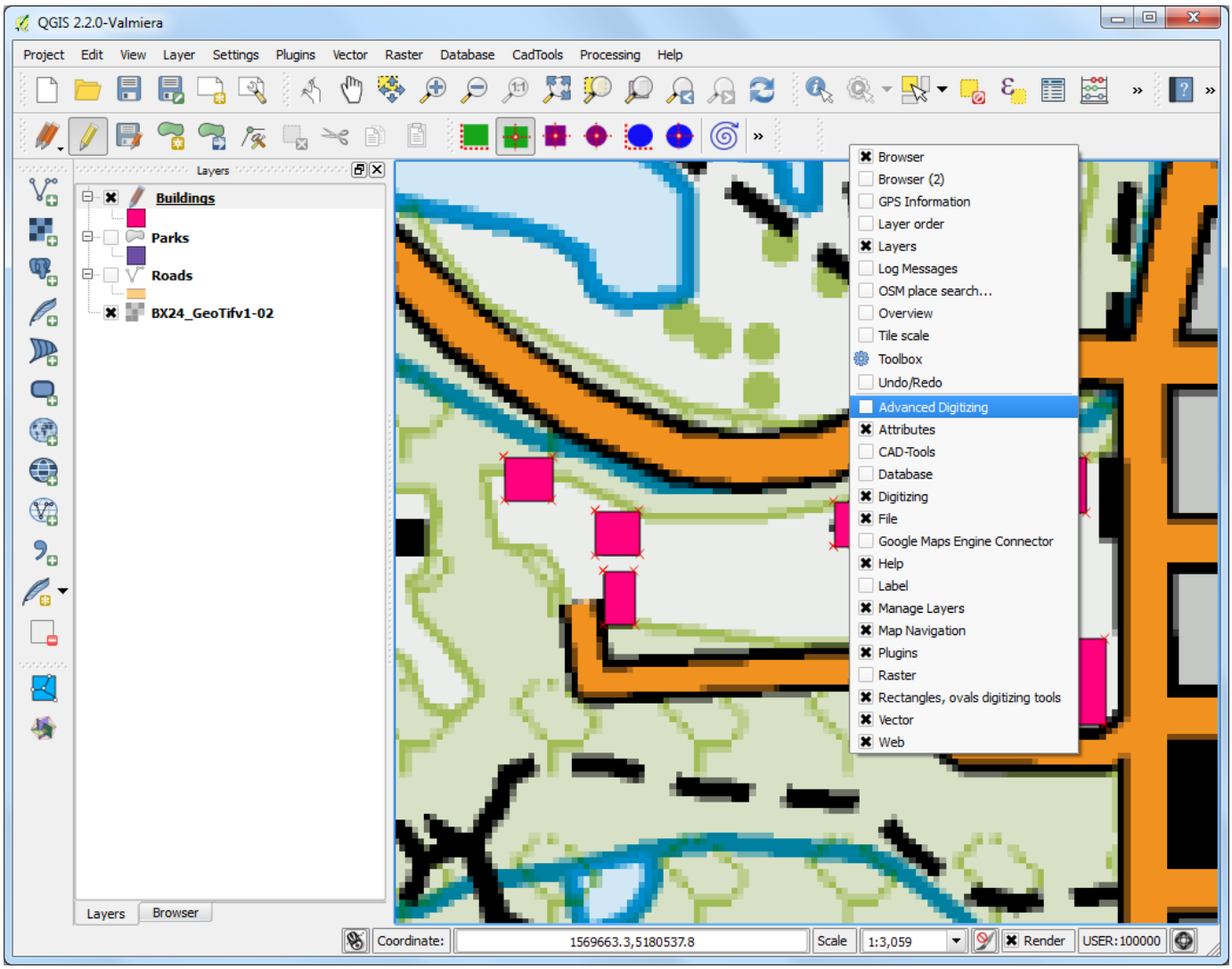
31. You will notice that some buildings are not vertical. We will need to draw a rectangle at an angle to match the building footprint. Click the Rectangle from center.



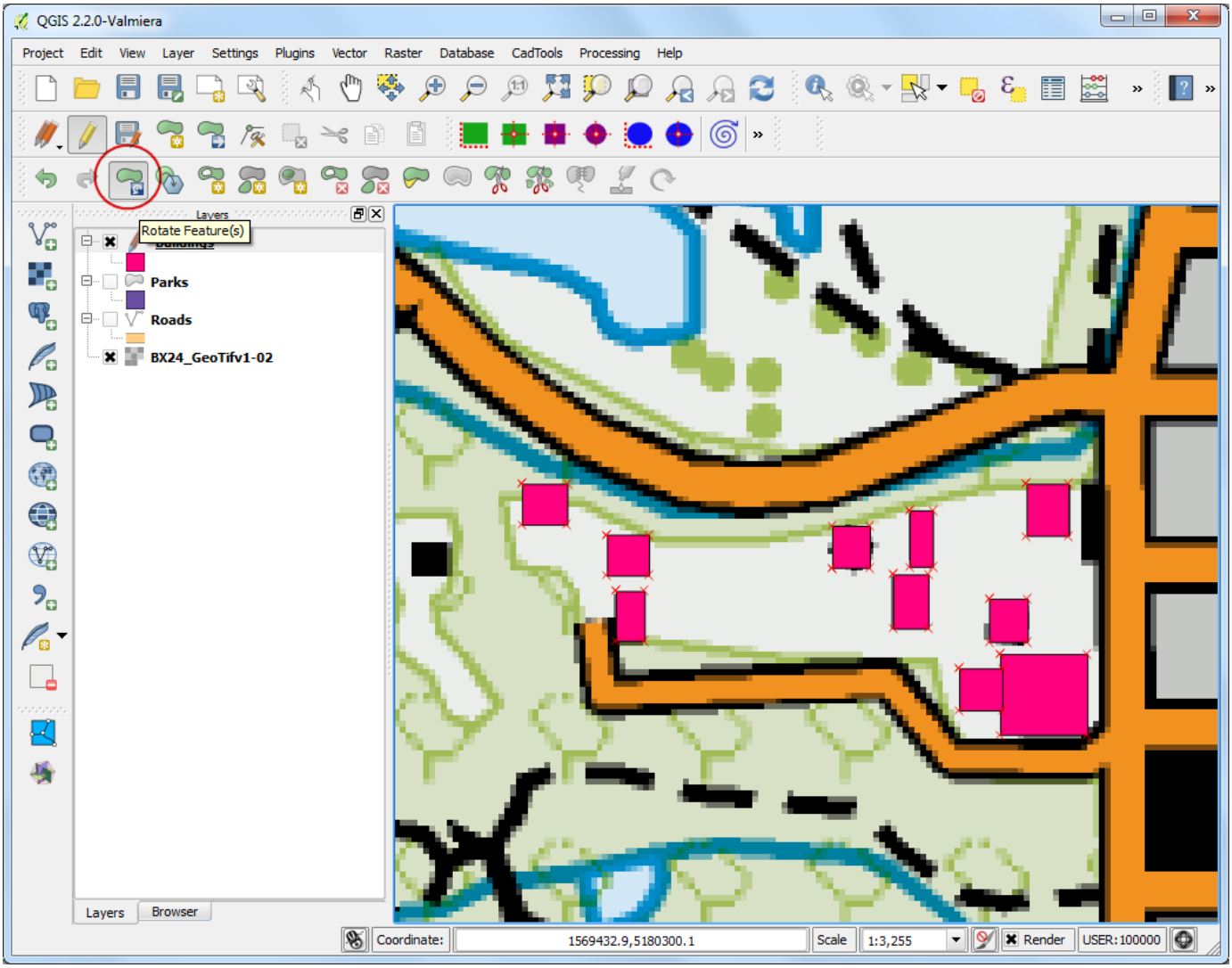
32. Click at the center of the building and drag the mouse to draw a vertical rectangle.



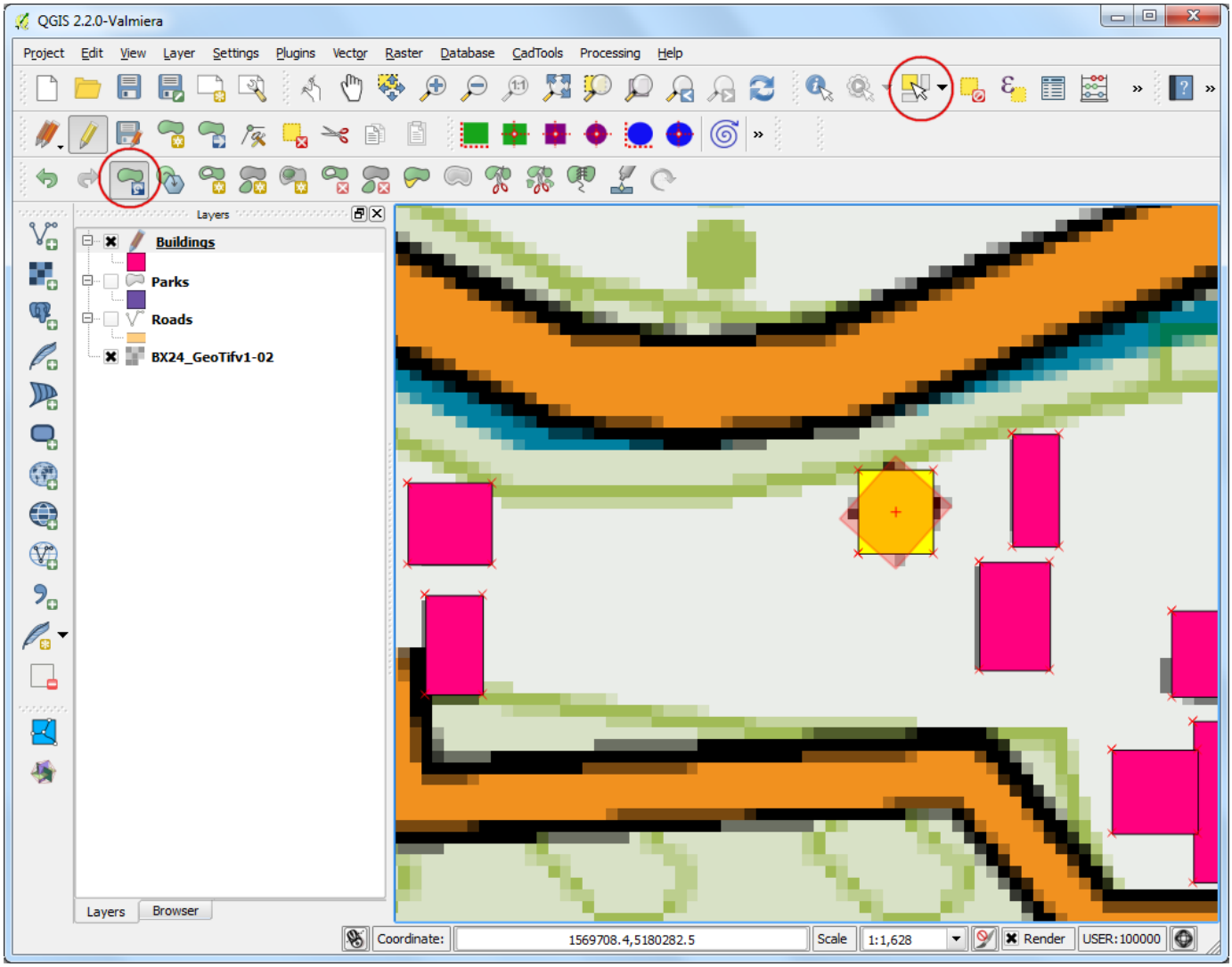
33. We need to rotate this rectangle to match the image on the topo map. The rotate tool is available in the **Advanced Digitizing** toolbar. Right-click on an empty area on the toolbar section and enable the Advanced Digitizing toolbar.



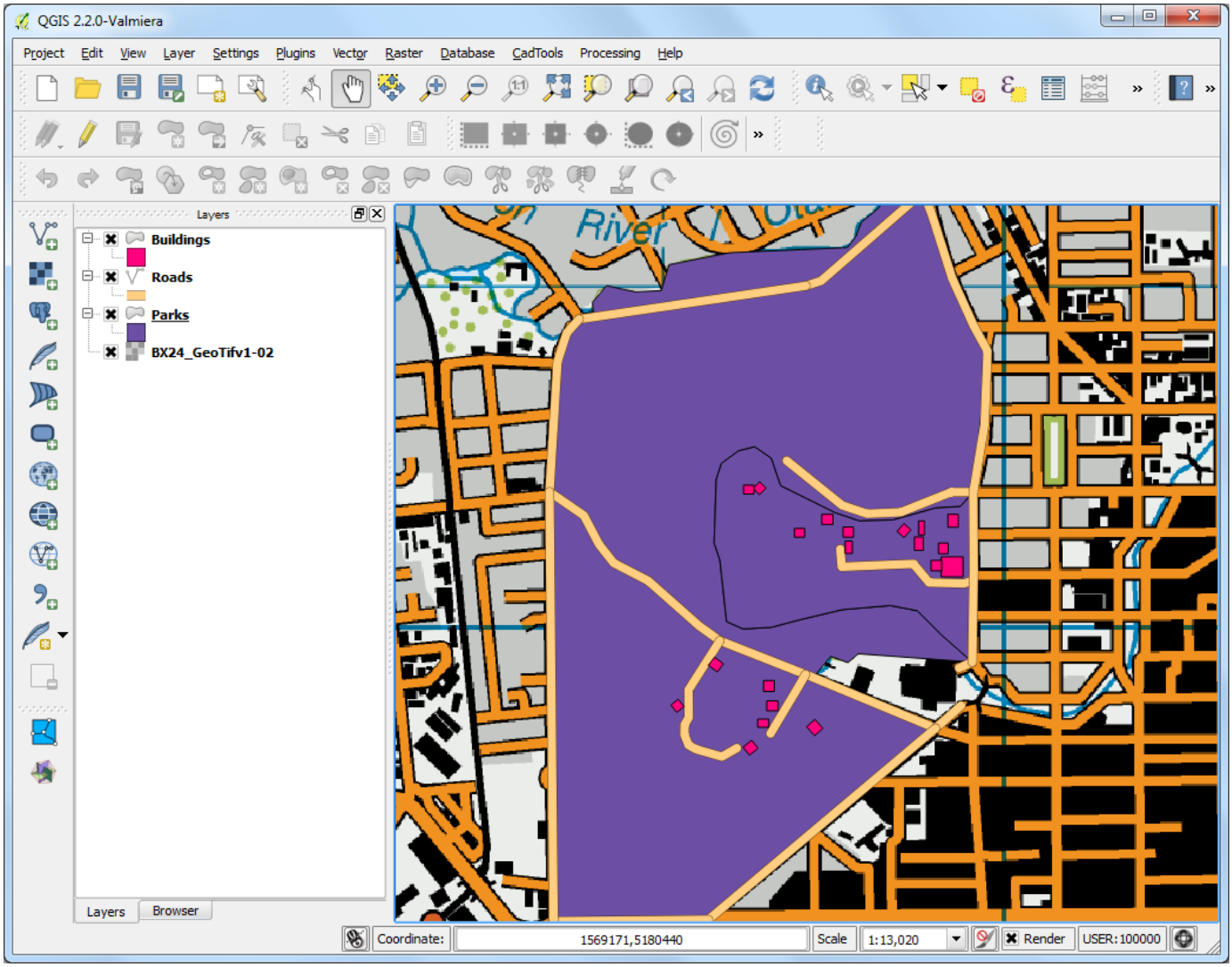
34. Click the Rotate Feature(s) button.



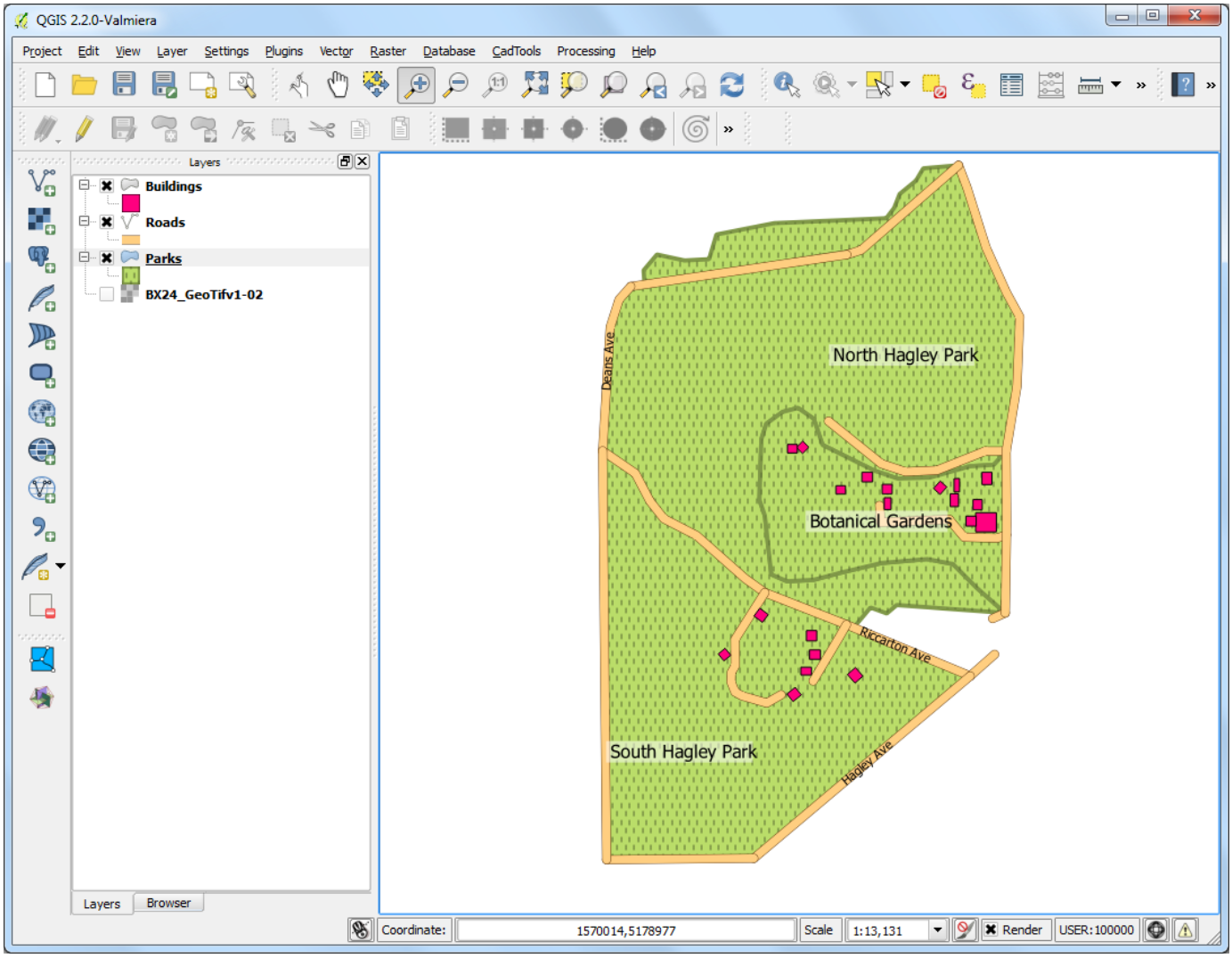
35. Use the Select Single feature tool to select the polygon that you want to rotate. Once the Rotate Feature(s) tool is activated, you will see crosshairs at the center of the polygon. Click exactly on that crosshairs and drag the mouse while holding the left-click button. A preview of the rotated feature will appear. Let go of the mouse button when the polygon aligns with the building footprint.



36. Save the layer edits and click Toggle Editing once you finish digitizing all buildings. You can drag the layers to change their order of appearance.



37. The digitizing task is now complete. You can play with the styling and labelling options in layer properties to create a nice looking map from the data you created.


QGIS 2.2.0-Valmiera
Project Edit View Layer Settings Plugins Vector Raster Database CadTools Processing Help
Layers
Buildings
Roads
Parks
BX24_GeoTifv1-02
Layers Browser
North Hagley Park
Botanical Gardens
South Hagley Park
Deans Ave
Riccarton Ave
Hagley Ave
Coordinate: 1570014,5178977 Scale 1:13,131 Render USER:100000

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Performing Table Joins (QGIS3)

Not every dataset you want to use comes in spatial format. Often the data would come as a table or a spreadsheet and you would need to link it with your existing spatial data for use in your analysis. This operation is known as a **Table Join** and is done using the `Join attributes by field value` Processing algorithm.

Overview of the task

We will use a shapefile of census tracts for California and population data table from US Census Bureau to create a population map for california.

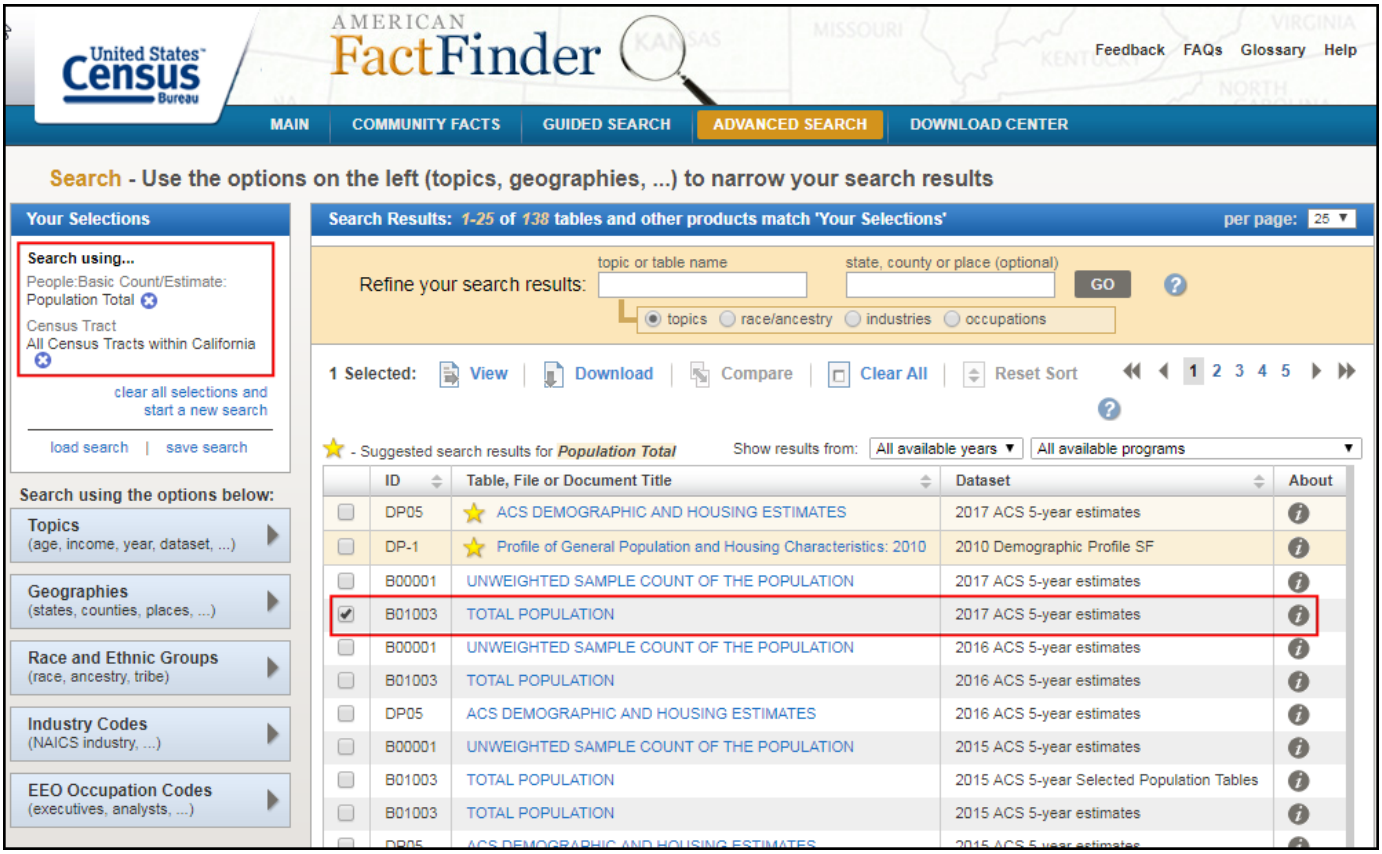
Other skills you will learn

- Loading CSV files that do not contain any geometry in QGIS.
- Using DB Manager to perform SQL queries to calculate group statistics.

Get the data

US Census Bureau (https://www.census.gov/en.html) provides TIGER/Line Shapefiles (https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html). You can visit the FTP site (https://www2.census.gov/geo/tiger/TIGER2018/) and download census tracts shapefile for California. Download Census Tracts for California (https://www2.census.gov/geo/tiger/TIGER2018/TRACT/tl_2018_06_tract.zip) file.

Americal FactFinder (http://factfinder2.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t) is a repository of all census data for the US. You can use Advanced Search and query for the Topic - Basic Count/Estimate and Geographies - All Census Tracts in California to create a custom CSV and download it. This tutorial uses `TOTAL POPULATION | 2017 ACS 5-year estimates` data.



For convenience, you may directly download a copy of both the datasets from the links below:

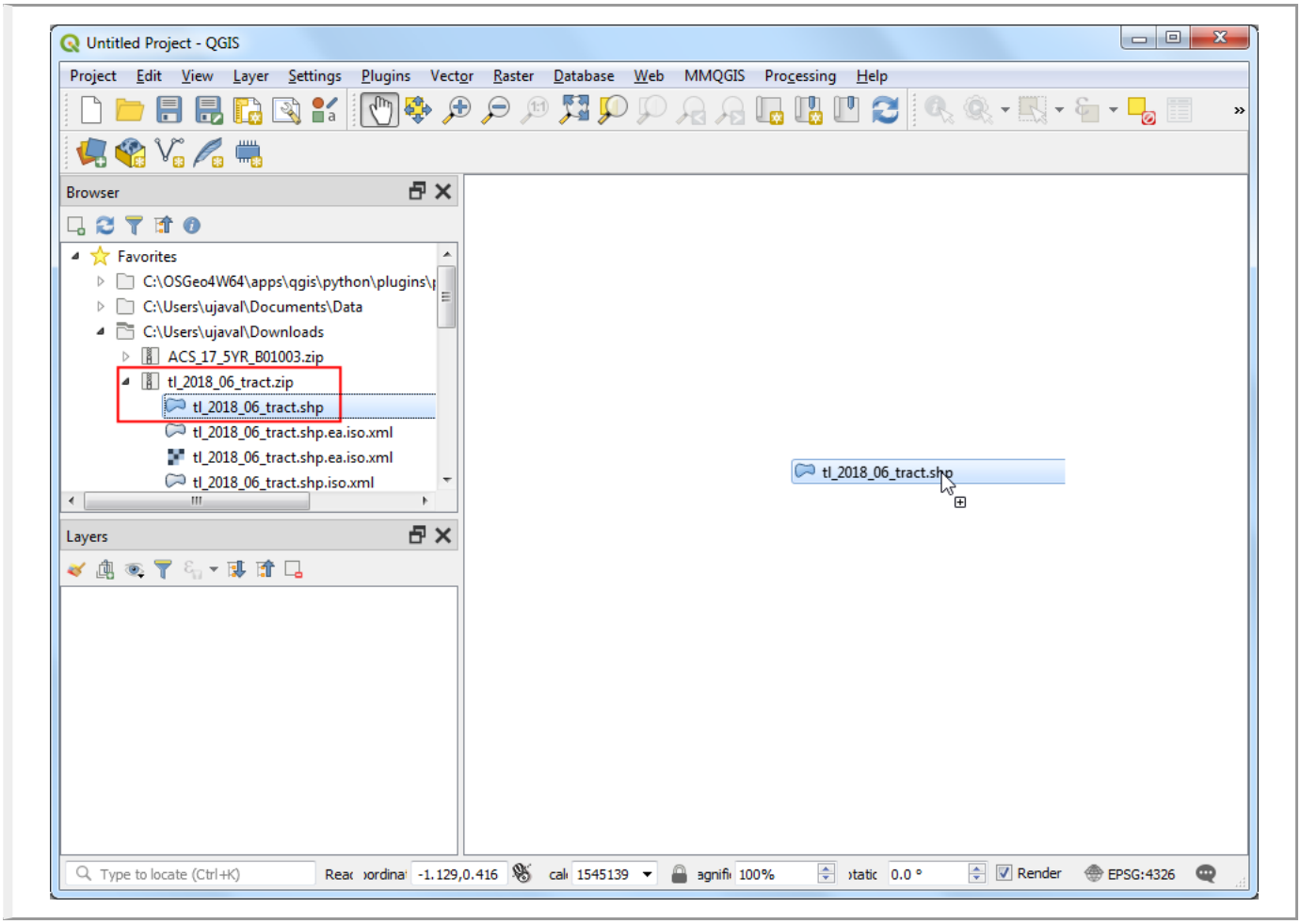
tl_2018_06_tract.zip (http://www.qgistutorials.com/downloads/tl_2018_06_tract.zip)

ACS_17_5YR_B01003.zip (http://www.qgistutorials.com/downloads/ACS_17_5YR_B01003.zip)

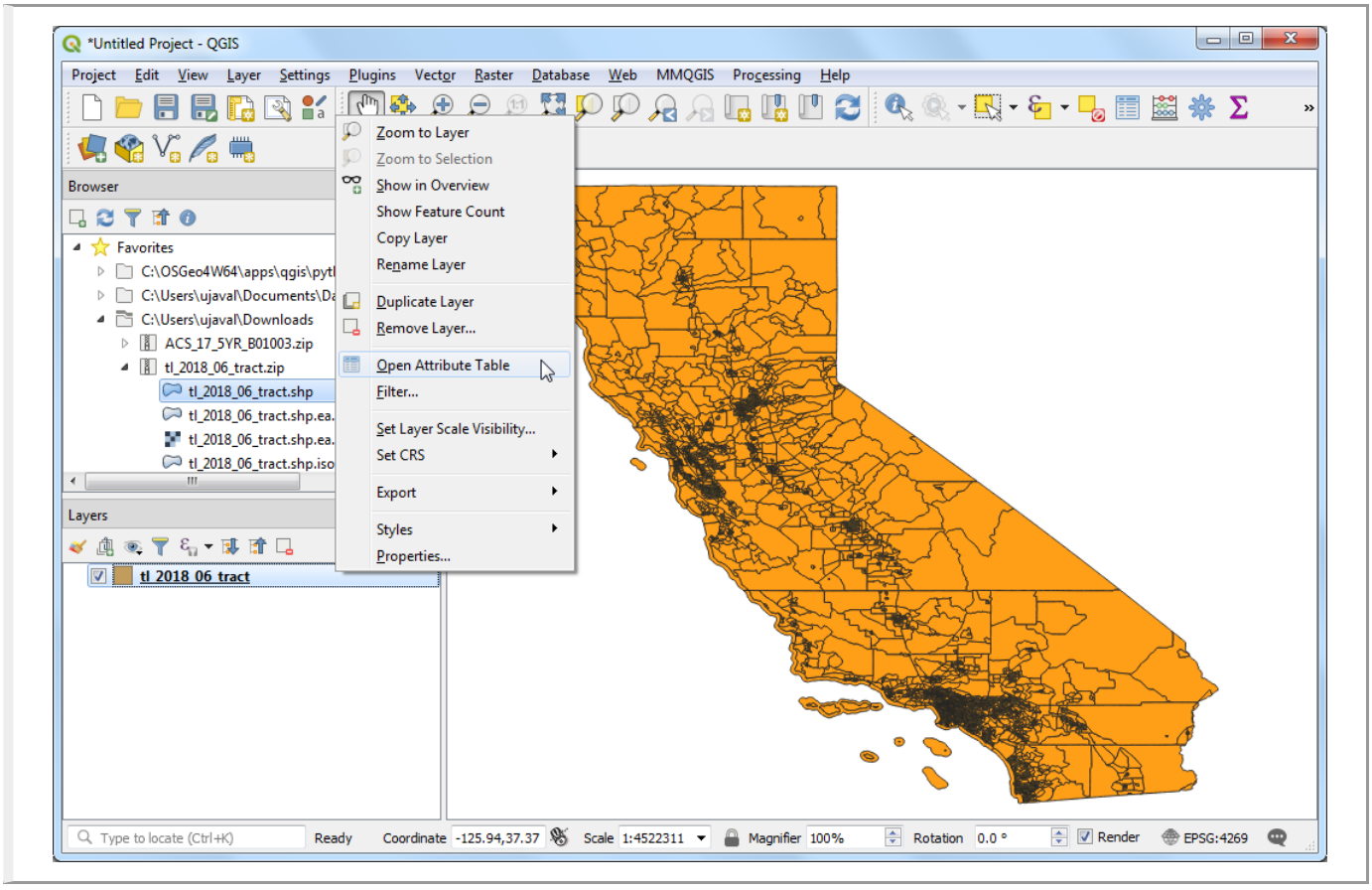
Data Source [TIGER] (../credits.html#tiger) [USCENSUS] (../credits.html#uscensus)

Procedure

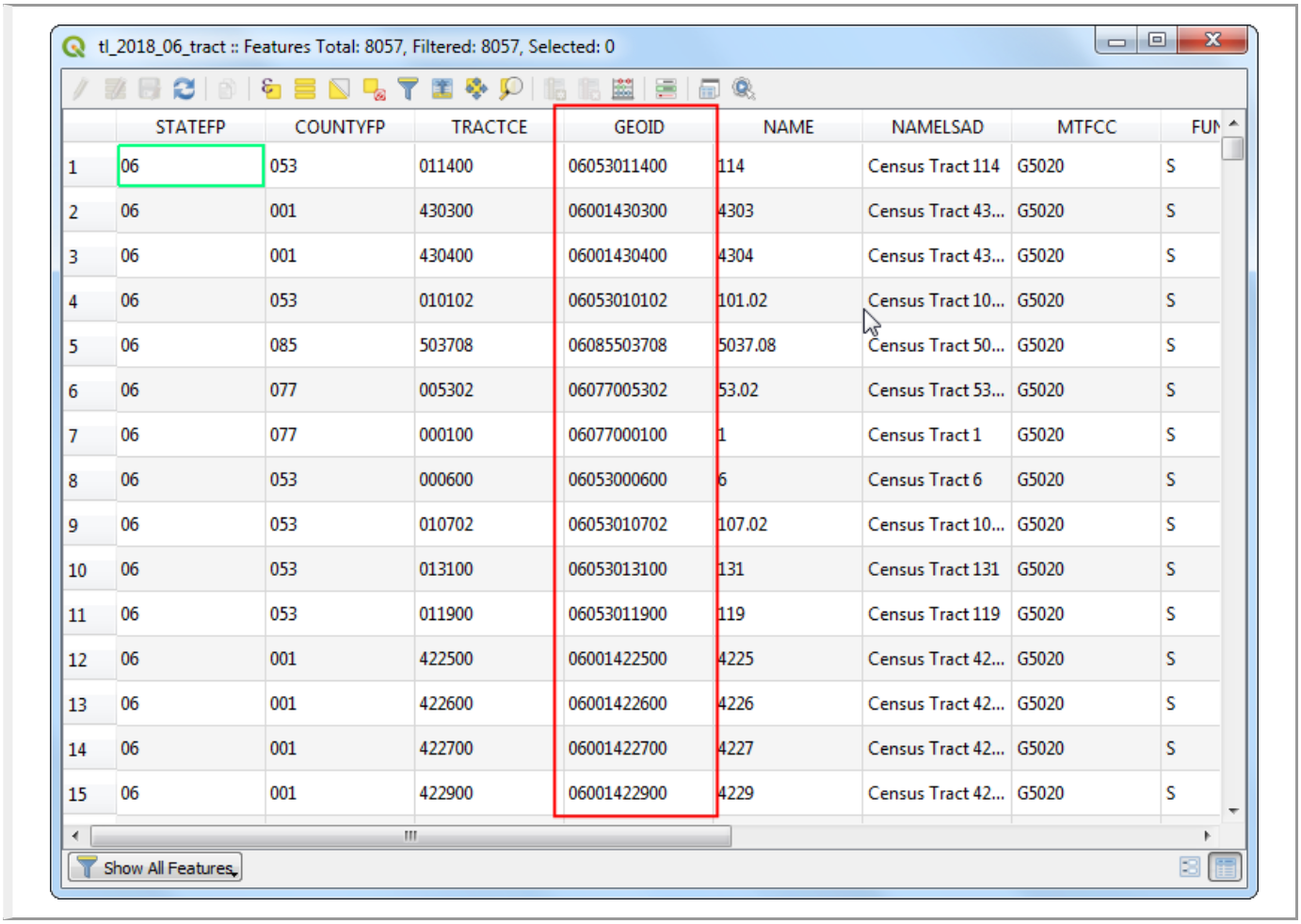
1. Locate the `tl_2018_06_tract.zip` file in the QGIS Browser and expand it. Select the `tl_2018_06_tract.shp` file and drag it to the canvas.



2. You will see the layer `tl_2018_06_tract` loaded in the Layers panel. This layer contains the boundaries of census tracts in California. Right-click on the `tl_2018_06_tract` layer and select Open Attribute Table.



3. Examine the attributes of the layer. To join a table with this layer, we need a unique and common attribute for each feature. In this case, the `GEOID` field is a unique identifier for each tract and can be used to link this layer with any other layer or table containing the same ID.



	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME	NAMELSAD	MTFCC	FUN
1	06	053	011400	06053011400	114	Census Tract 114	G5020	S
2	06	001	430300	06001430300	4303	Census Tract 43...	G5020	S
3	06	001	430400	06001430400	4304	Census Tract 43...	G5020	S
4	06	053	010102	06053010102	101.02	Census Tract 10...	G5020	S
5	06	085	503708	06085503708	5037.08	Census Tract 50...	G5020	S
6	06	077	005302	06077005302	53.02	Census Tract 53...	G5020	S
7	06	077	000100	06077000100	1	Census Tract 1	G5020	S
8	06	053	000600	06053000600	6	Census Tract 6	G5020	S
9	06	053	010702	06053010702	107.02	Census Tract 10...	G5020	S
10	06	053	013100	06053013100	131	Census Tract 131	G5020	S
11	06	053	011900	06053011900	119	Census Tract 119	G5020	S
12	06	001	422500	06001422500	4225	Census Tract 42...	G5020	S
13	06	001	422600	06001422600	4226	Census Tract 42...	G5020	S
14	06	001	422700	06001422700	4227	Census Tract 42...	G5020	S
15	06	001	422900	06001422900	4229	Census Tract 42...	G5020	S

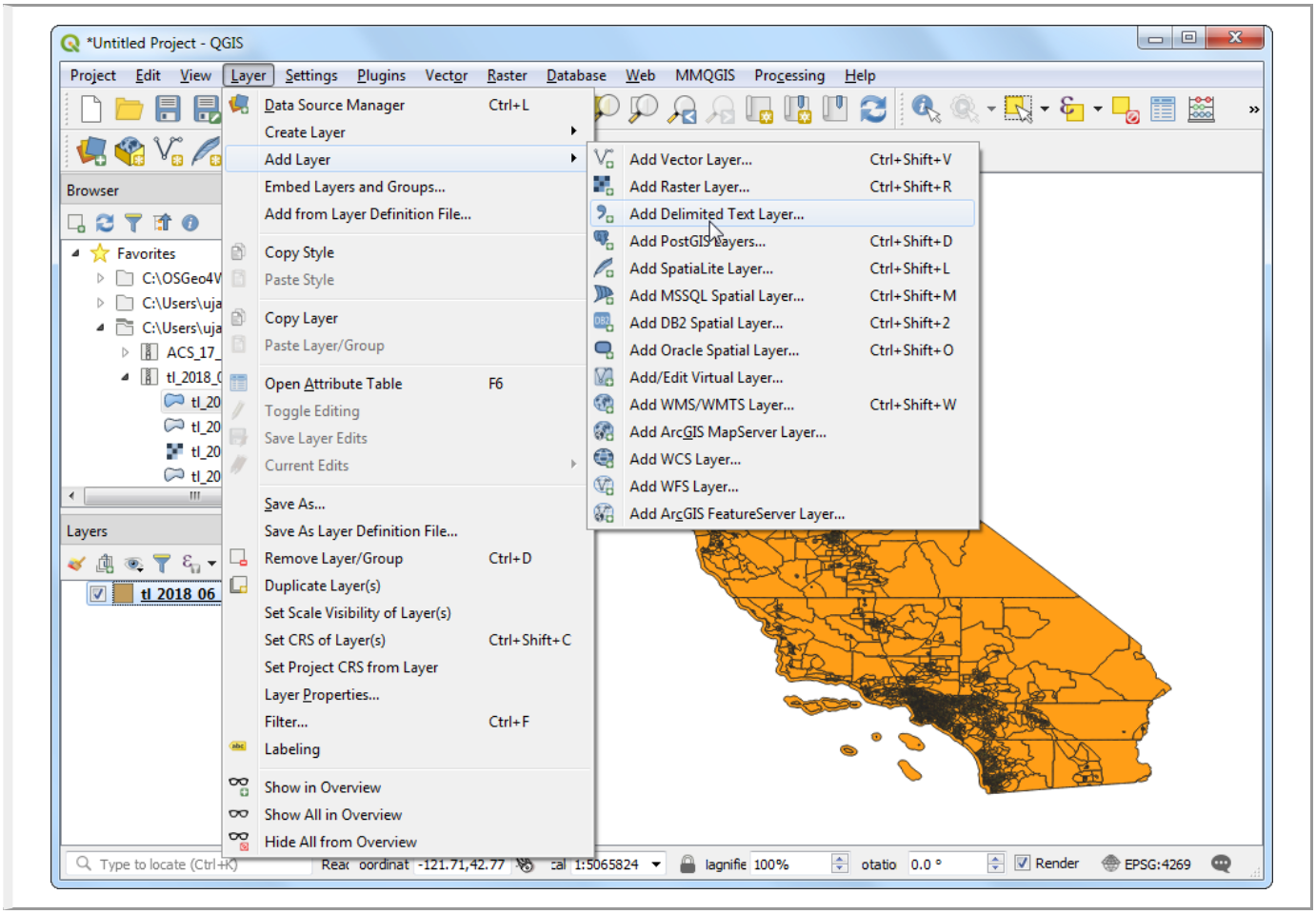
4. Unzip the `ACS_17_5YR_B01003.zip` file and open the `ACS_17_5YR_B01003_with_ann.csv` file in a text editor. You will notice that each row of the file contains information about a tract along with the unique identifier we saw in the previous step. Note that this field is called `GEO.id2` in the CSV. You will also note that the `HD01_VD01` column has population value for each of the census tract.

```
1  GEO.id,GEO.id2,GEO.display-label,HD01_VD01,HD02_VD01
2  Id,Id2,Geography,Estimate; Total,Margin of Error; Total
3  1400000US06001400100,06001400100,"Census Tract 4001, Alameda County, California",2991,194
4  1400000US06001400200,06001400200,"Census Tract 4002, Alameda County, California",1997,96
5  1400000US06001400300,06001400300,"Census Tract 4003, Alameda County, California",5123,312
6  1400000US06001400400,06001400400,"Census Tract 4004, Alameda County, California",3991,250
7  1400000US06001400500,06001400500,"Census Tract 4005, Alameda County, California",3944,313
8  1400000US06001400600,06001400600,"Census Tract 4006, Alameda County, California",1635,177
9  1400000US06001400700,06001400700,"Census Tract 4007, Alameda County, California",4837,523
10 1400000US06001400800,06001400800,"Census Tract 4008, Alameda County, California",3796,418
11 1400000US06001400900,06001400900,"Census Tract 4009, Alameda County, California",2394,186
12 1400000US06001401000,06001401000,"Census Tract 4010, Alameda County, California",6247,473
13 1400000US06001401100,06001401100,"Census Tract 4011, Alameda County, California",4205,334
14 1400000US06001401200,06001401200,"Census Tract 4012, Alameda County, California",2601,184
15 1400000US06001401300,06001401300,"Census Tract 4013, Alameda County, California",4054,649
16 1400000US06001401400,06001401400,"Census Tract 4014, Alameda County, California",4013,423
17 1400000US06001401500,06001401500,"Census Tract 4015, Alameda County, California",2638,298
```

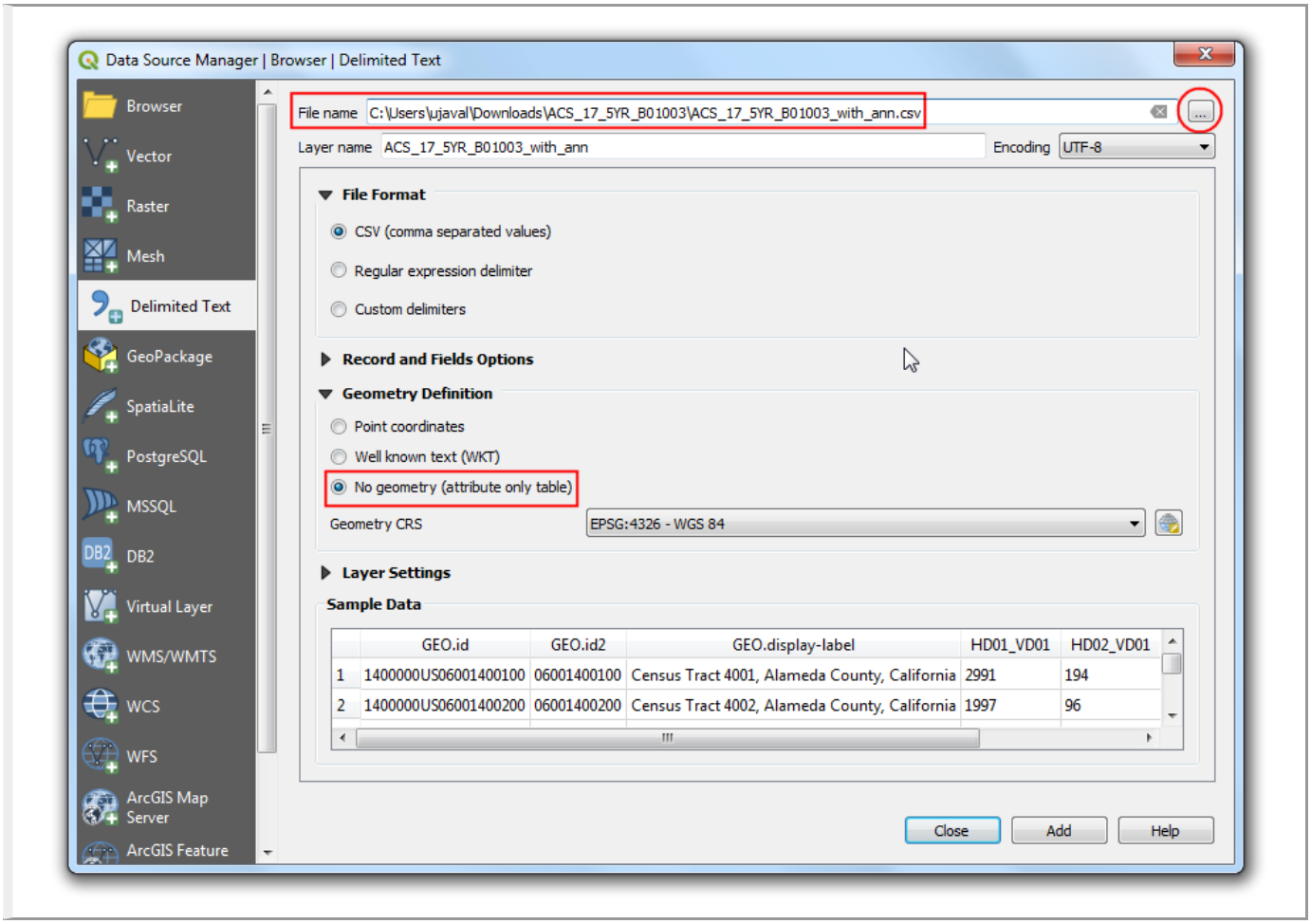
5. Before importing this CSV file, we need to make a minor edit. QGIS CSV importer expects the first row of the file to contain the column headers and all remaining rows to contain the data for these columns. This file contains an extra row 2 with column labels. Delete that row and save the file.

```
GEO.id,GEO.id2,GEO.display-label,HD01_VD01,HD02_VD01
Id,Id2,Geography,Estimate; Total,Margin of Error; Total
1400000US06001400100,06001400100,"Census Tract 4001, Alameda County, California",2991,194
1400000US06001400200,06001400200,"Census Tract 4002, Alameda County, California",1997,96
1400000US06001400300,06001400300,"Census Tract 4003, Alameda County, California",5123,312
1400000US06001400400,06001400400,"Census Tract 4004, Alameda County, California",3991,250
1400000US06001400500,06001400500,"Census Tract 4005, Alameda County, California",3944,313
1400000US06001400600,06001400600,"Census Tract 4006, Alameda County, California",1635,177
1400000US06001400700,06001400700,"Census Tract 4007, Alameda County, California",4837,523
1400000US06001400800,06001400800,"Census Tract 4008, Alameda County, California",3796,418
1400000US06001400900,06001400900,"Census Tract 4009, Alameda County, California",2394,186
1400000US06001401000,06001401000,"Census Tract 4010, Alameda County, California",6247,473
1400000US06001401100,06001401100,"Census Tract 4011, Alameda County, California",4205,334
1400000US06001401200,06001401200,"Census Tract 4012, Alameda County, California",2601,184
1400000US06001401300,06001401300,"Census Tract 4013, Alameda County, California",4054,649
1400000US06001401400,06001401400,"Census Tract 4014, Alameda County, California",4013,423
1400000US06001401500,06001401500,"Census Tract 4015, Alameda County, California",2638,298
```

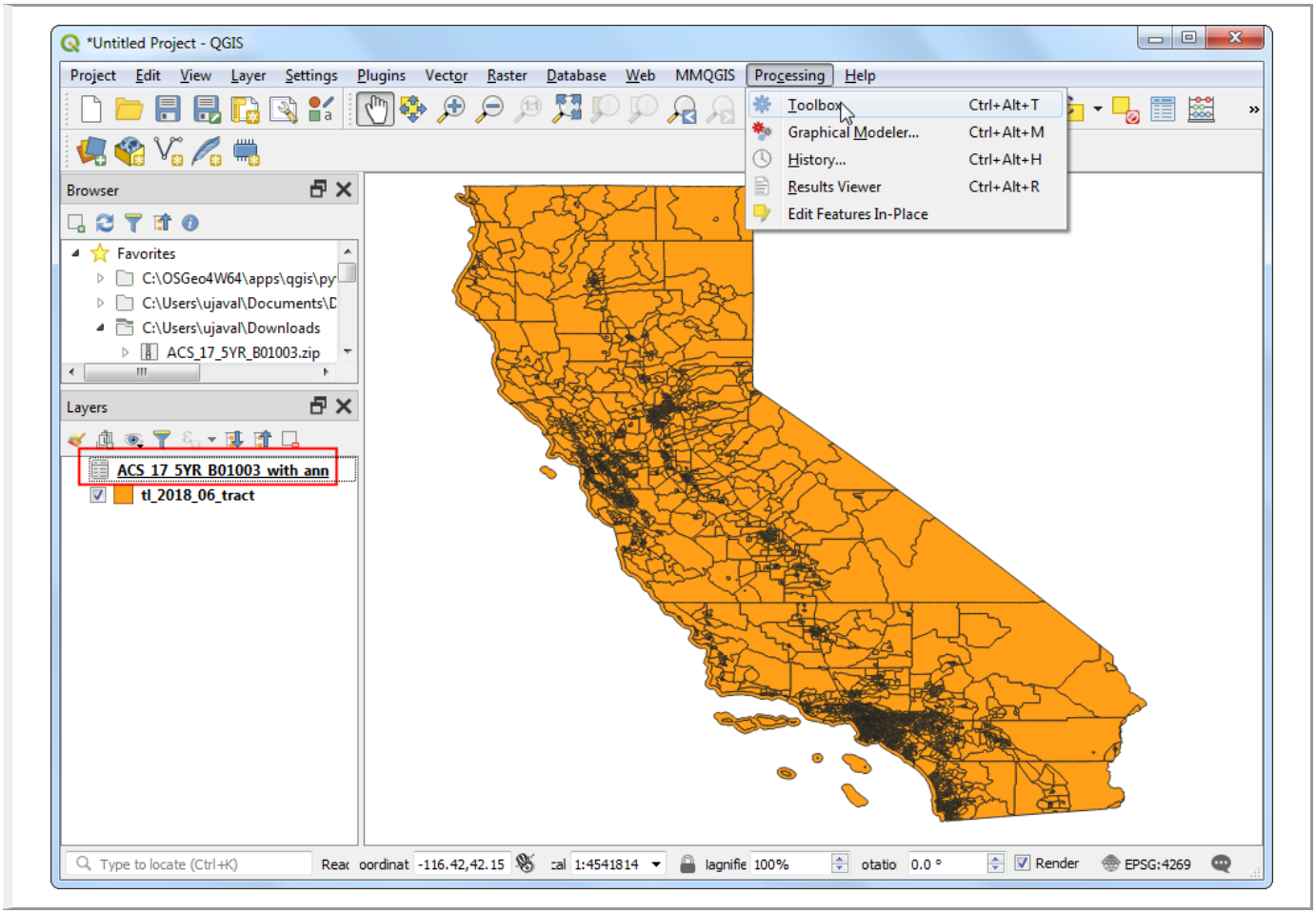
6. Now we are ready to import the CSV file to QGIS. Go to Layer ▸ Add Layer ▸ Add Delimited Text Layer.



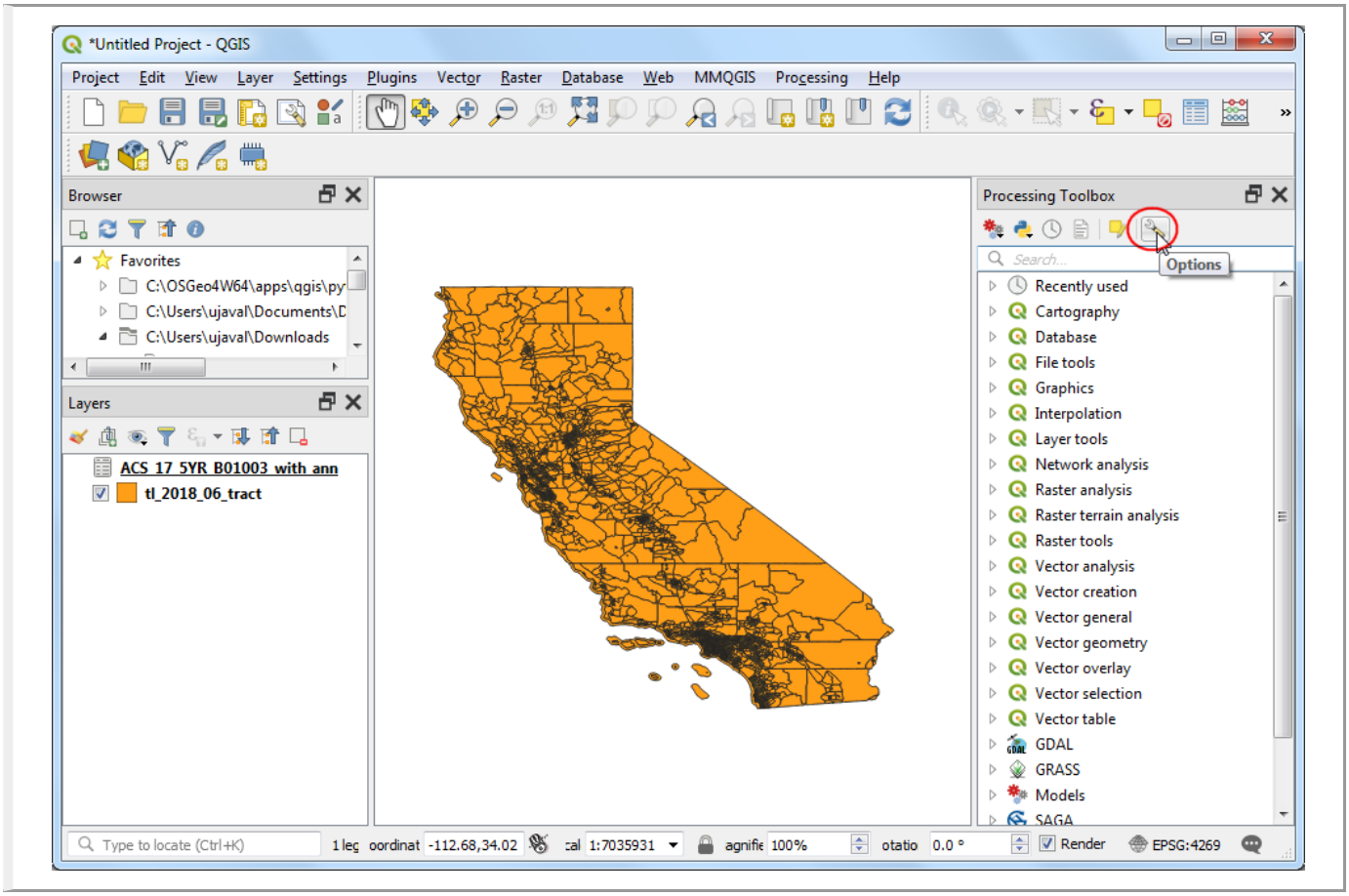
7. In the Data Source Manager window, click the ... button and browse to the CSV file and select it. Make sure you have selected File format as CSV (comma separated values). Since we are importing this as a table, we must specify that our file contains no geometry using the No geometry (attribute table only) option. Verify that Sample Data preview at the bottom looks fine and click Add followed by Close.



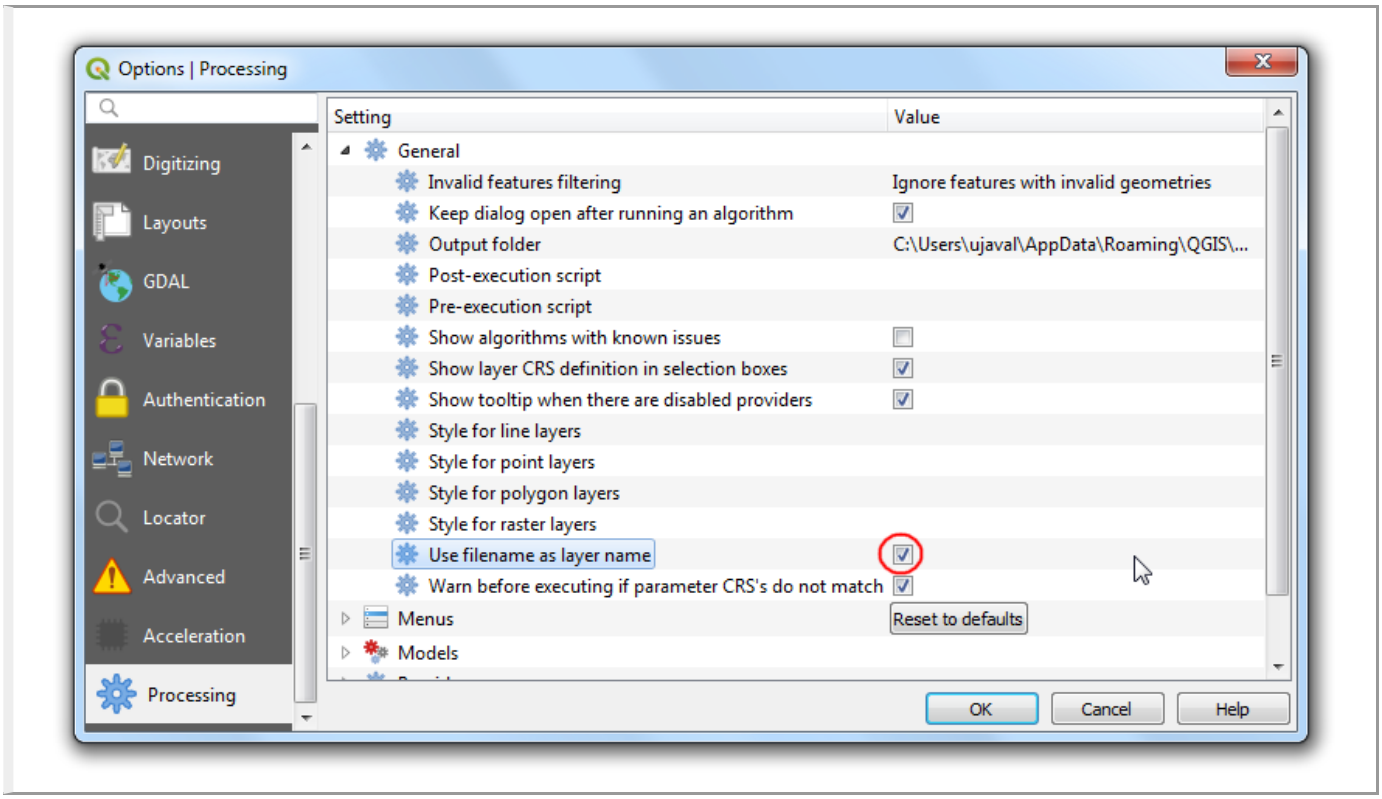
8. The CSV will now be imported as a table to QGIS and will appear as `ACS_17_5YR_B01003_with_ann` in the Layers panel. Now we are ready to create the table join. Go to Processing ▸ Toolbox.



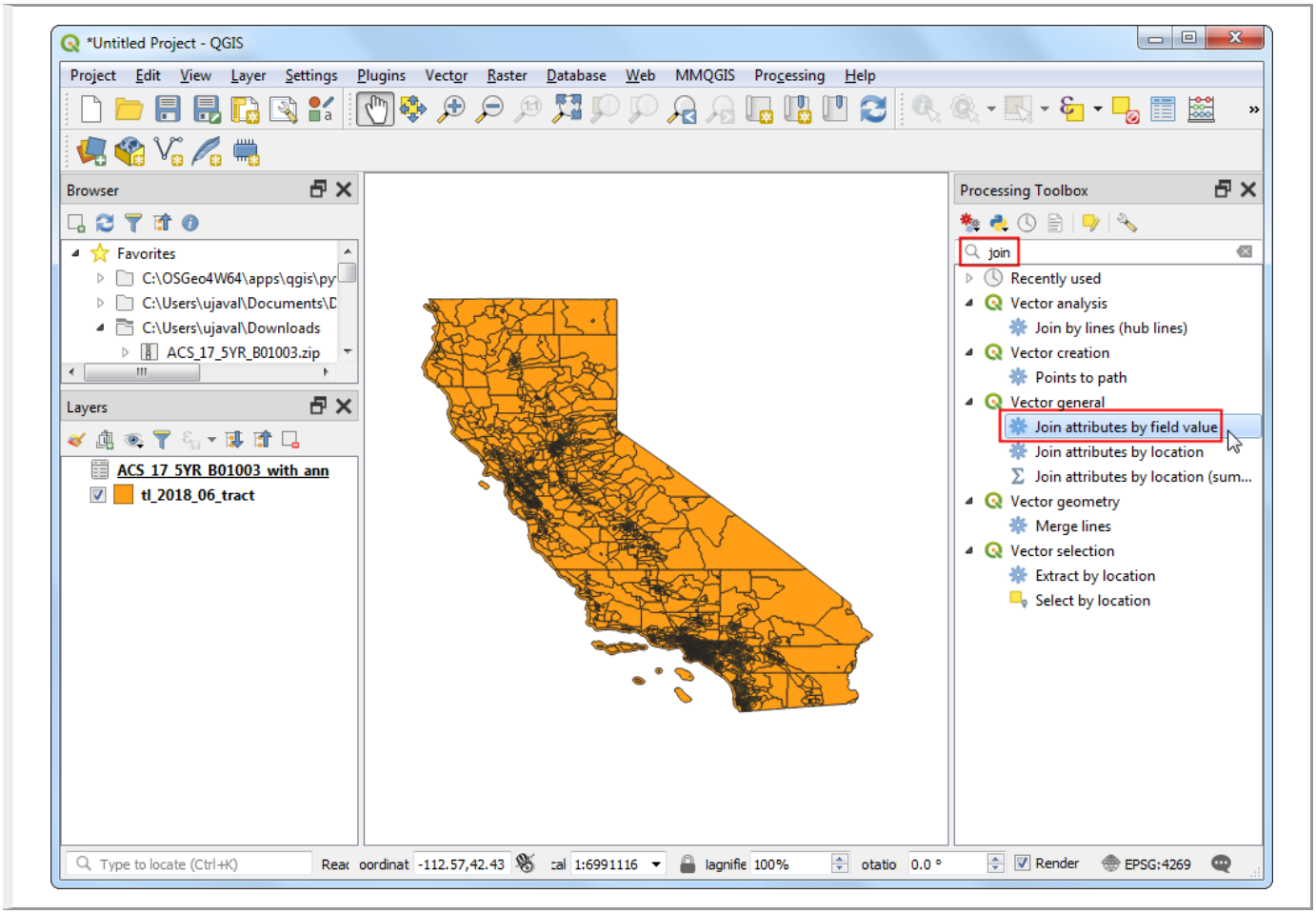
9. First we need to change a default setting in the Processing Toolbox. Click the Options button.



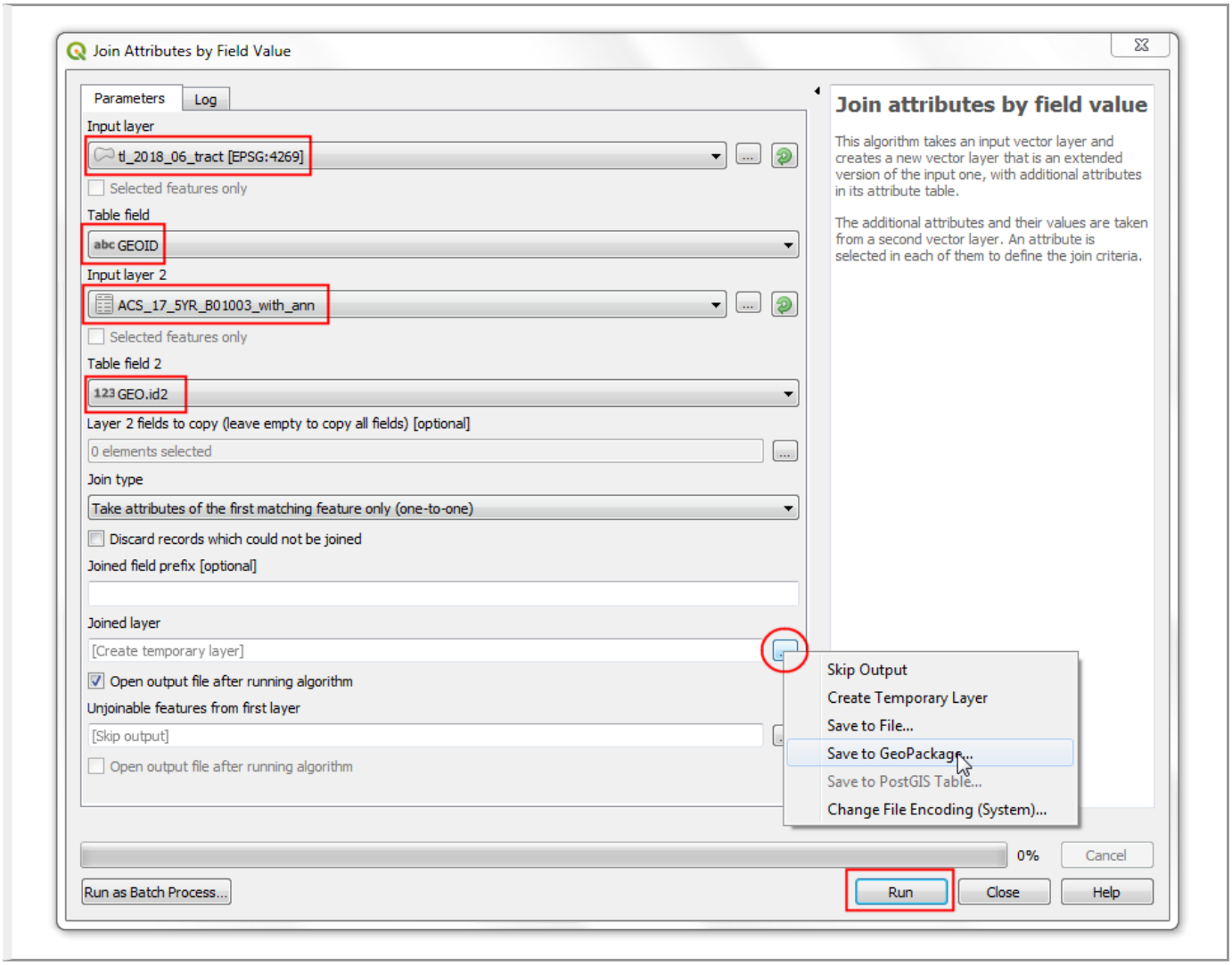
10. In the Processing Options tab, check the Use filename as layer name option. When using algorithms from Processing Toolbox, this option makes the output layer names much more intuitive and useful. Click OK.



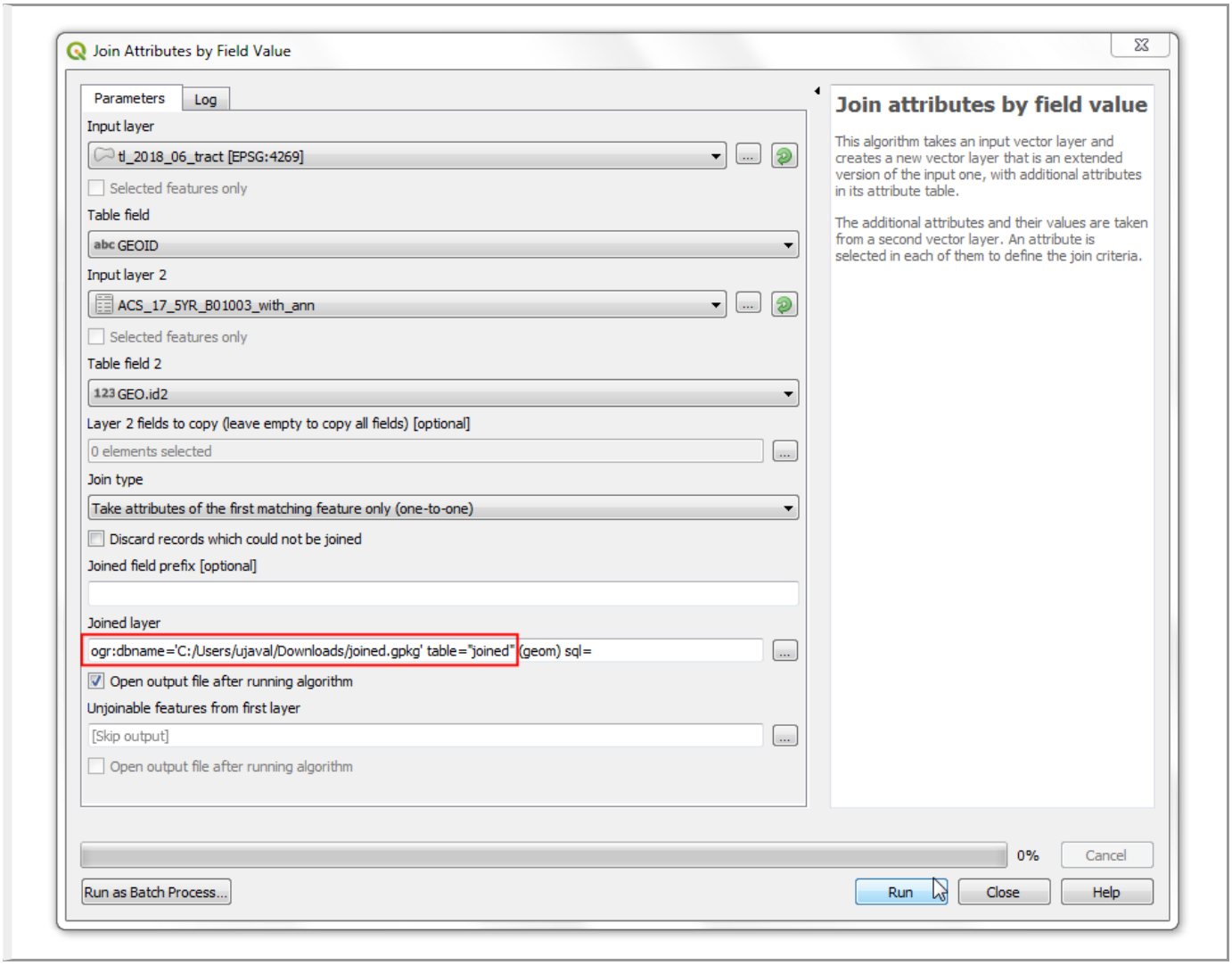
11. Back in the Processing Toolbox, search and locate the Vector General ▸ Join attributes by field value algorithm and double-click it to open it.



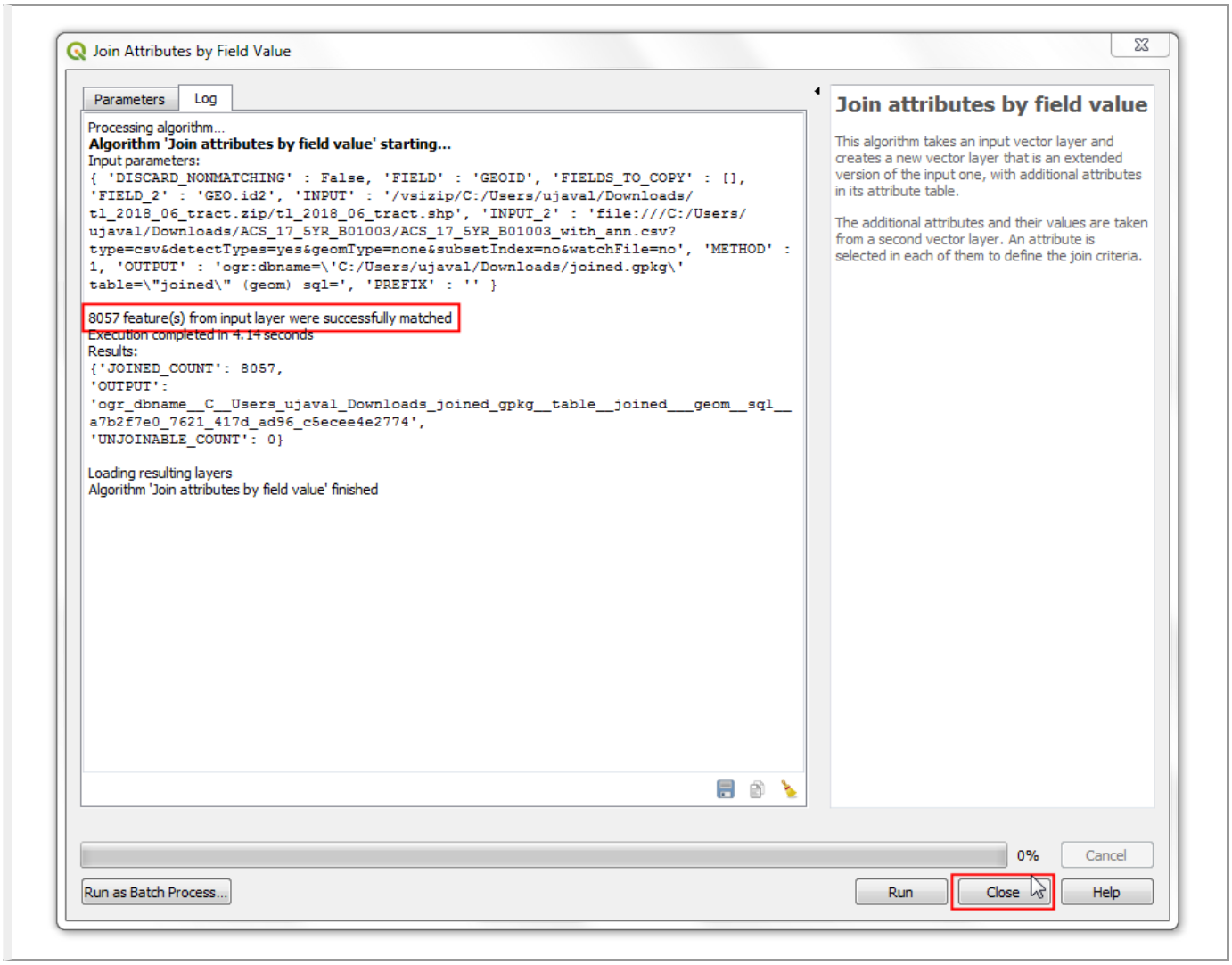
12. In the Join Attributes by Field Values dialog, select `tl_2018_06_tract` as Input layer and `GEOID` as the Table field. Select `ACS_17_5YR_B01003_with_ann` as the Input layer 2 and `GEO.id2` as the Table field 2. Leave other options to their default values and click the ... button to select the output file location and select `Save to GeoPackage...`.



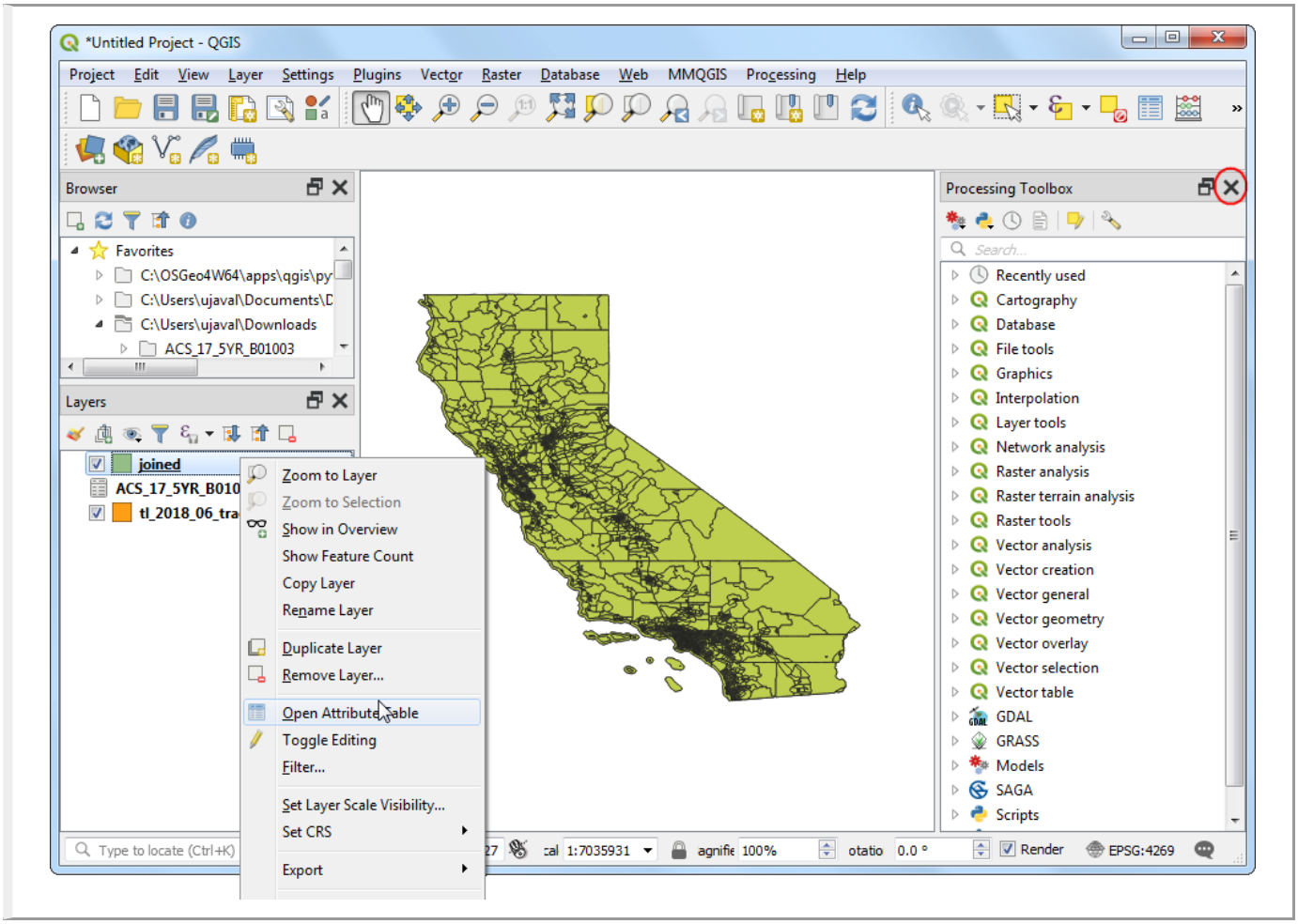
13. Name the output geopackage as `joined.gpkg` and the output layer as `joined`. Click Run.



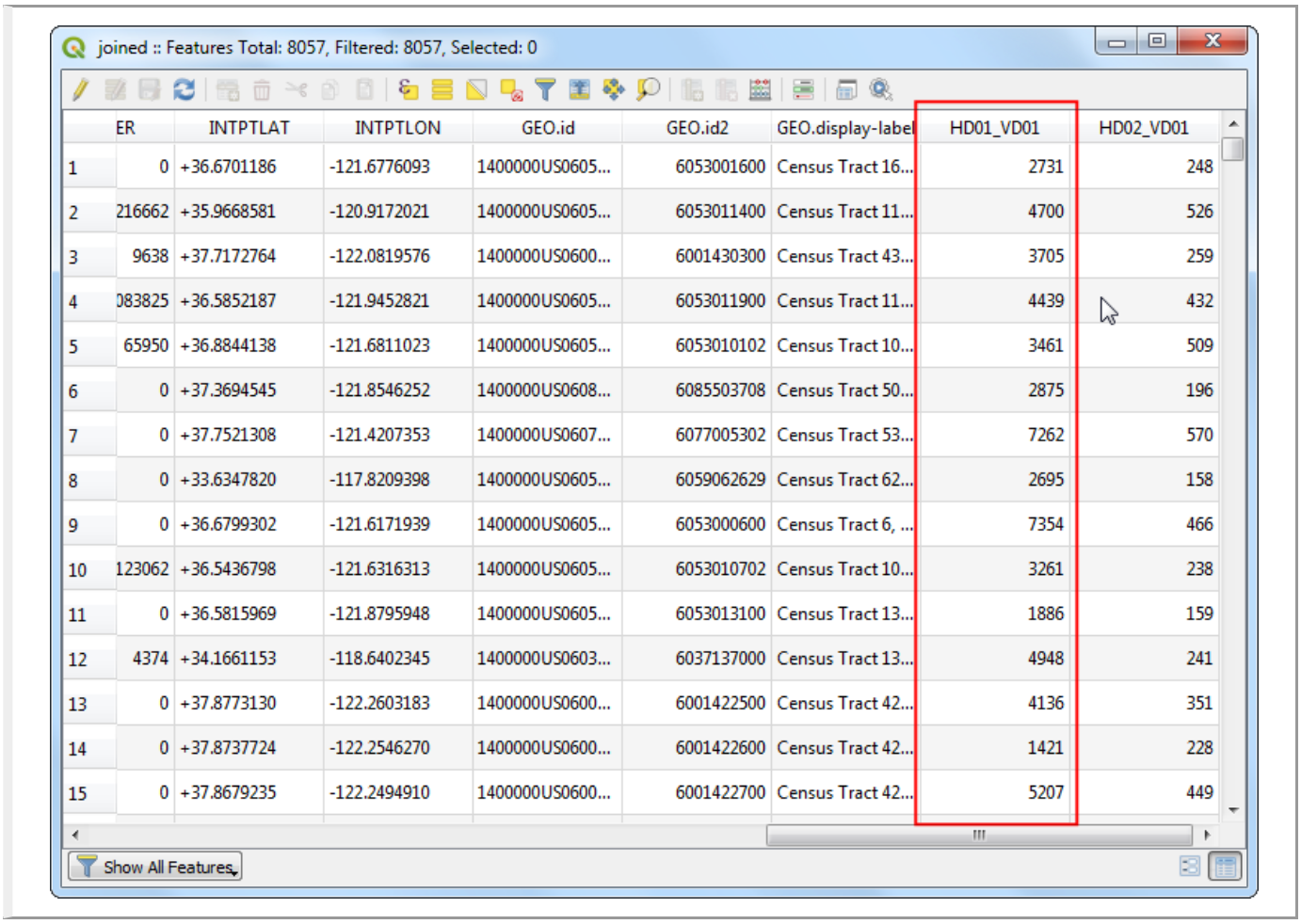
14. Once the processing finishes, verify that the algorithm was successful and click Close.



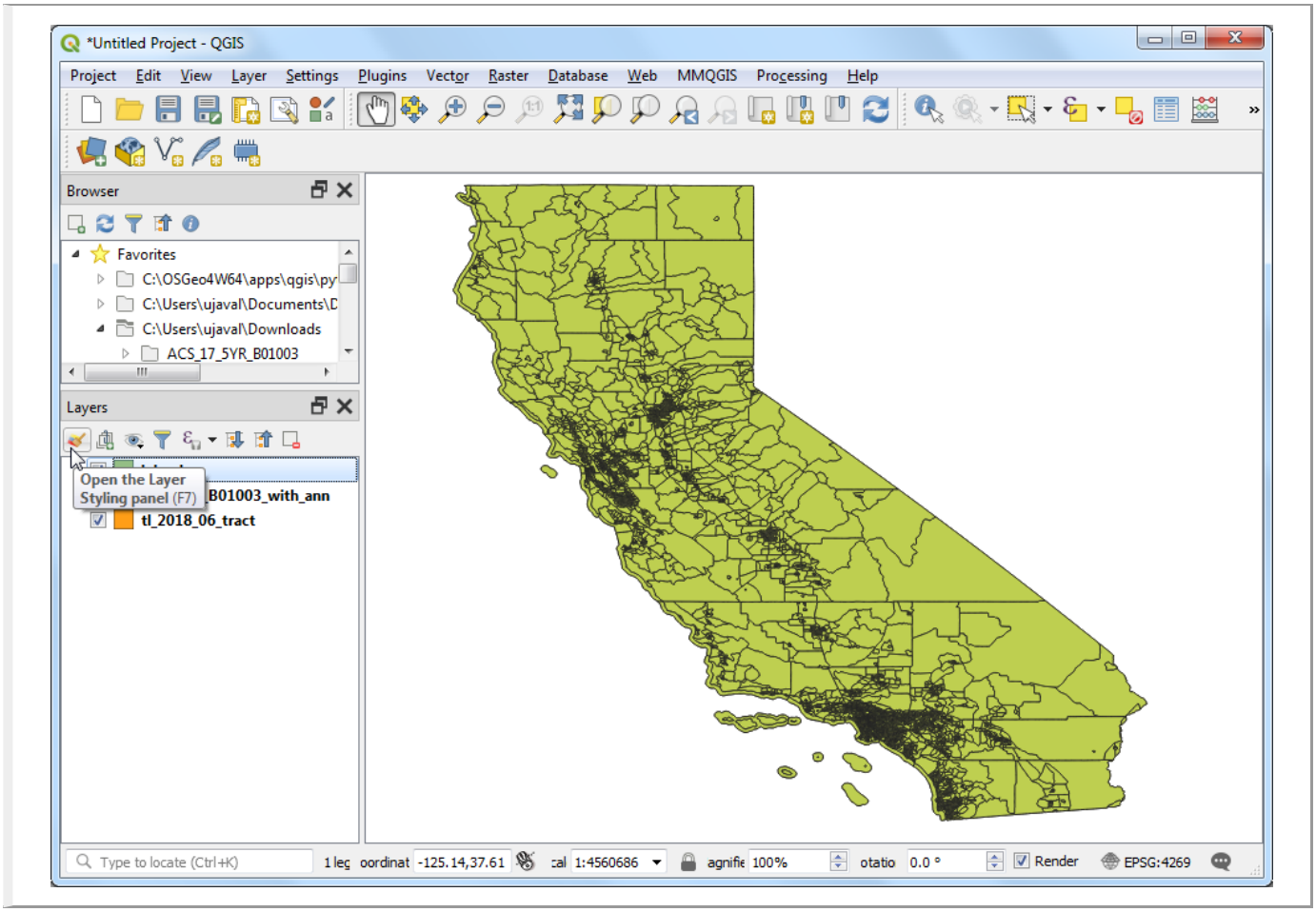
15. You will see a new layer `joined` loaded in the Layers panel. At this point, the fields from the CSV file are joined with the census tracts layer. You can close the Processing Toolbox for now. Right-click on the `joined` layer and select Open Attribute Table.



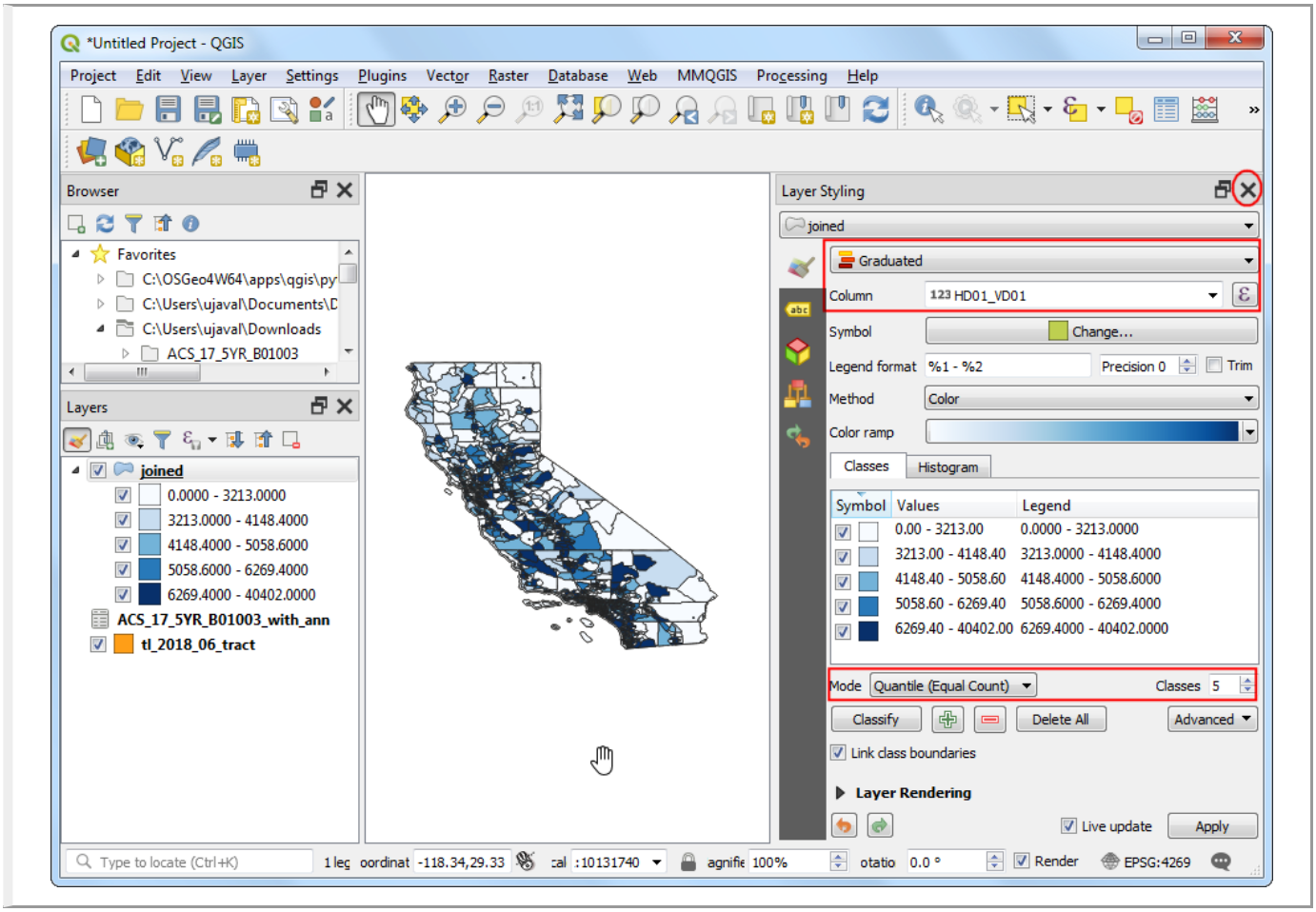
16. You will see a new set of fields, including the `HD01_VD01` field containing population estimates.



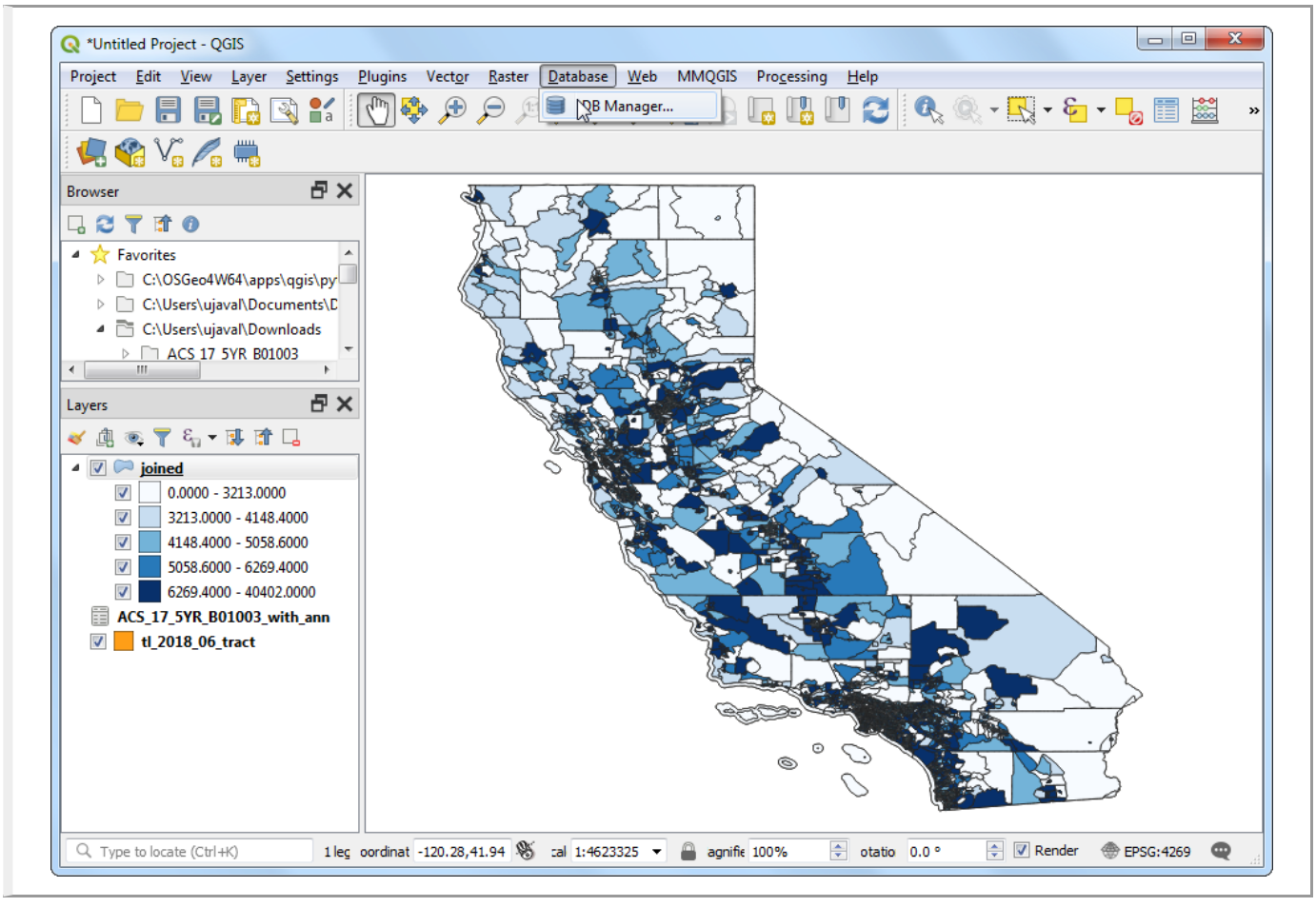
17. Now that we have the population data in the census tracts layer, we can style it to create a visualization of population distribution. Select the `joined` layer and click the Open the Layer Styling Panel button.



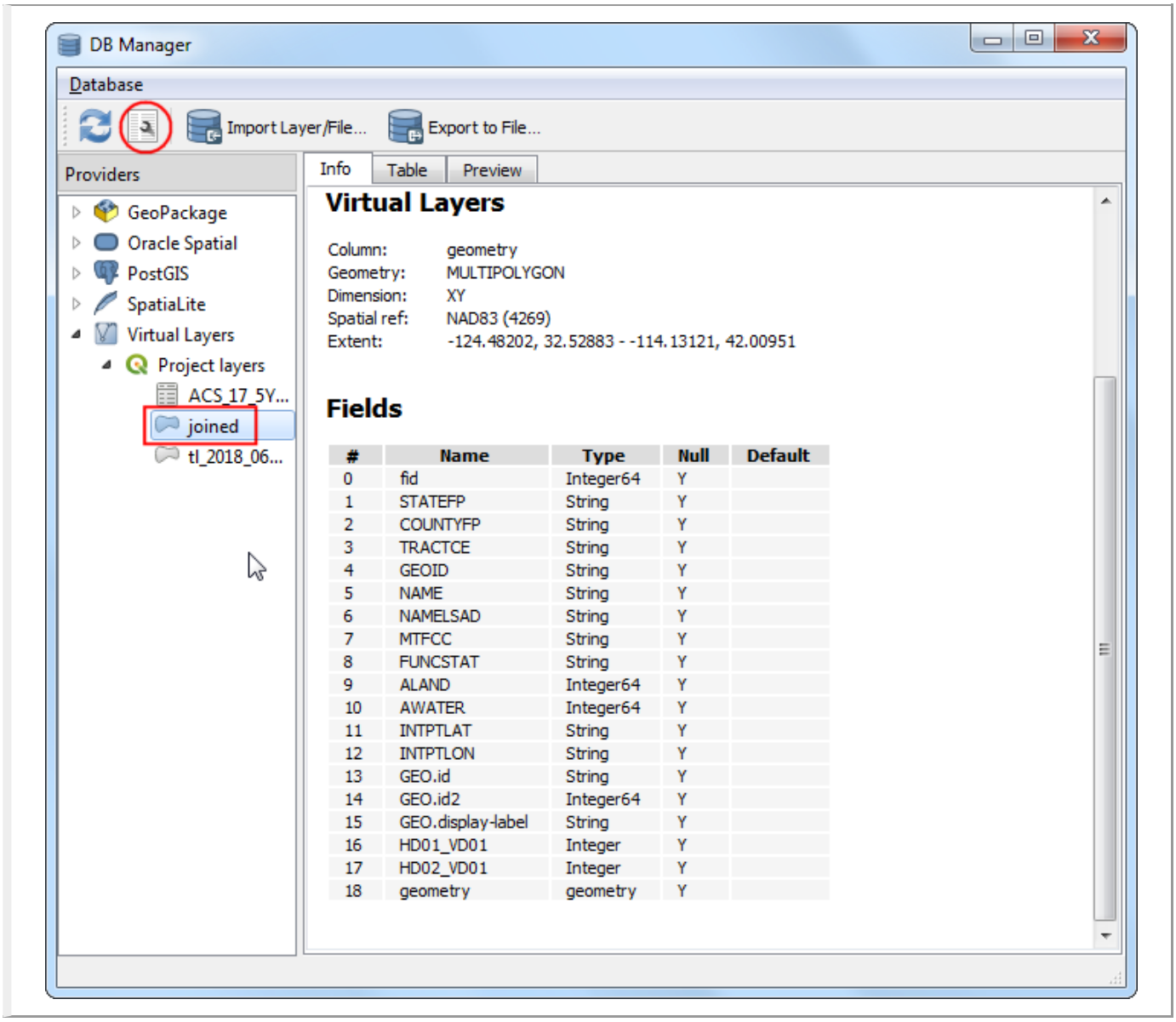
18. In the Layer Styling panel, select `Graduated` from the drop-down menu. As we are looking to create a population map, we want to assign different color to each census tract feature based on the population estimate. Select `HD01_VD01` as the Column. Select a color ramp of your liking from the Color ramp drop-down. In the Mode, select Quantile (Equal Count) with `5` Classes. Click the Classify button and see the map layer update with a color assigned to a population range. You can close the Layer Styling panel once you are satisfied with the map.



19. A good practice in any GIS analysis is to validate your results. To check our work, we can run some simple queries on the output layer to make sure the results are correct. Go to Database ▸ DB Manager....

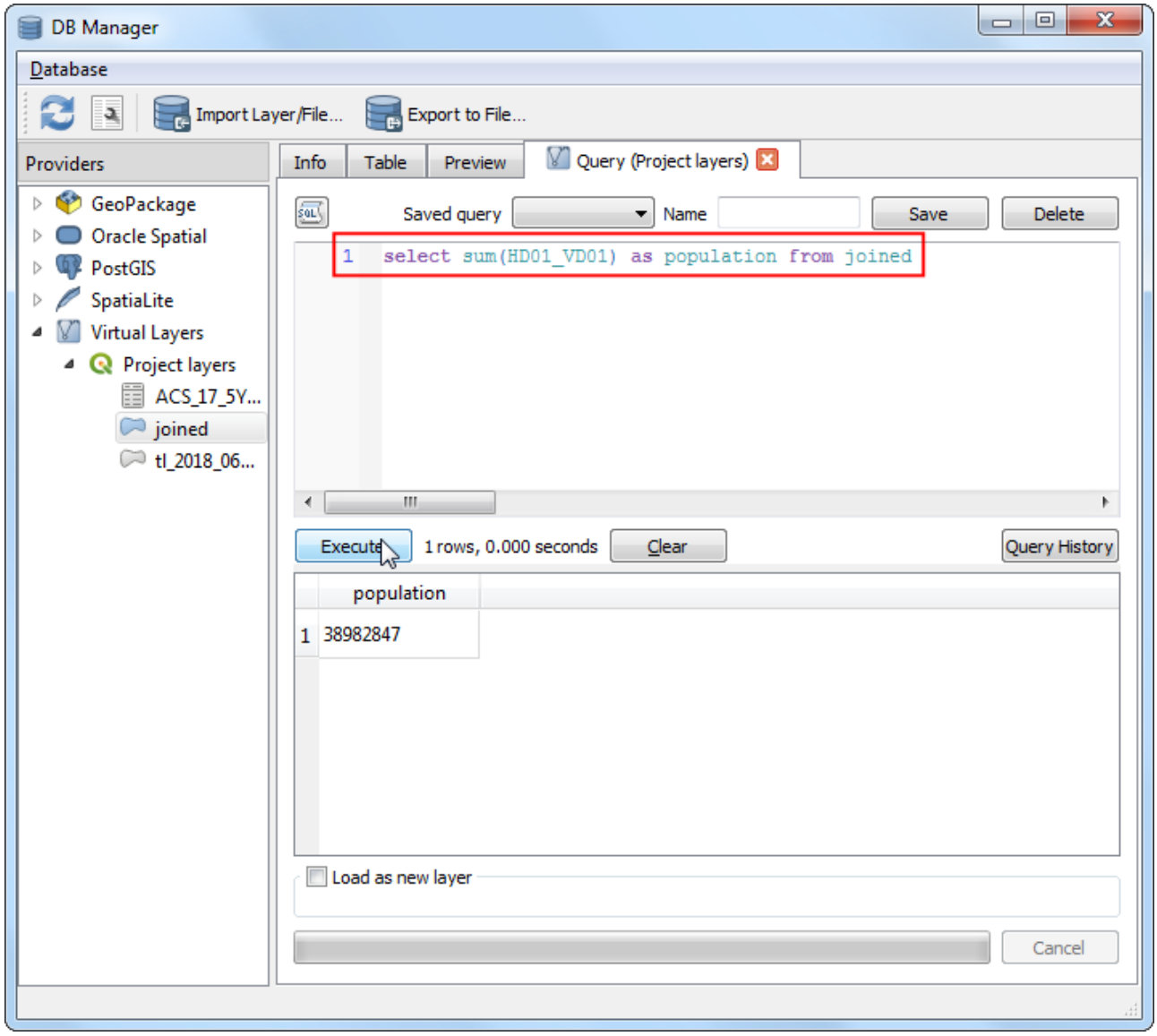


20. All layers loaded in QGIS are available as *Virtual Layers* that can be queried using SQL without loading them into a separate database. This add a lot of useful functionality by enabling spatial and non-spatial SQL queries via SQLite engine and the Spatialite library (https://www.gaia-gis.it/fossil/libspatialite/index). Locate the output layer from Virtual Layers ▸ Project layers ▸ joined` and select it. Click the SQL Window button.



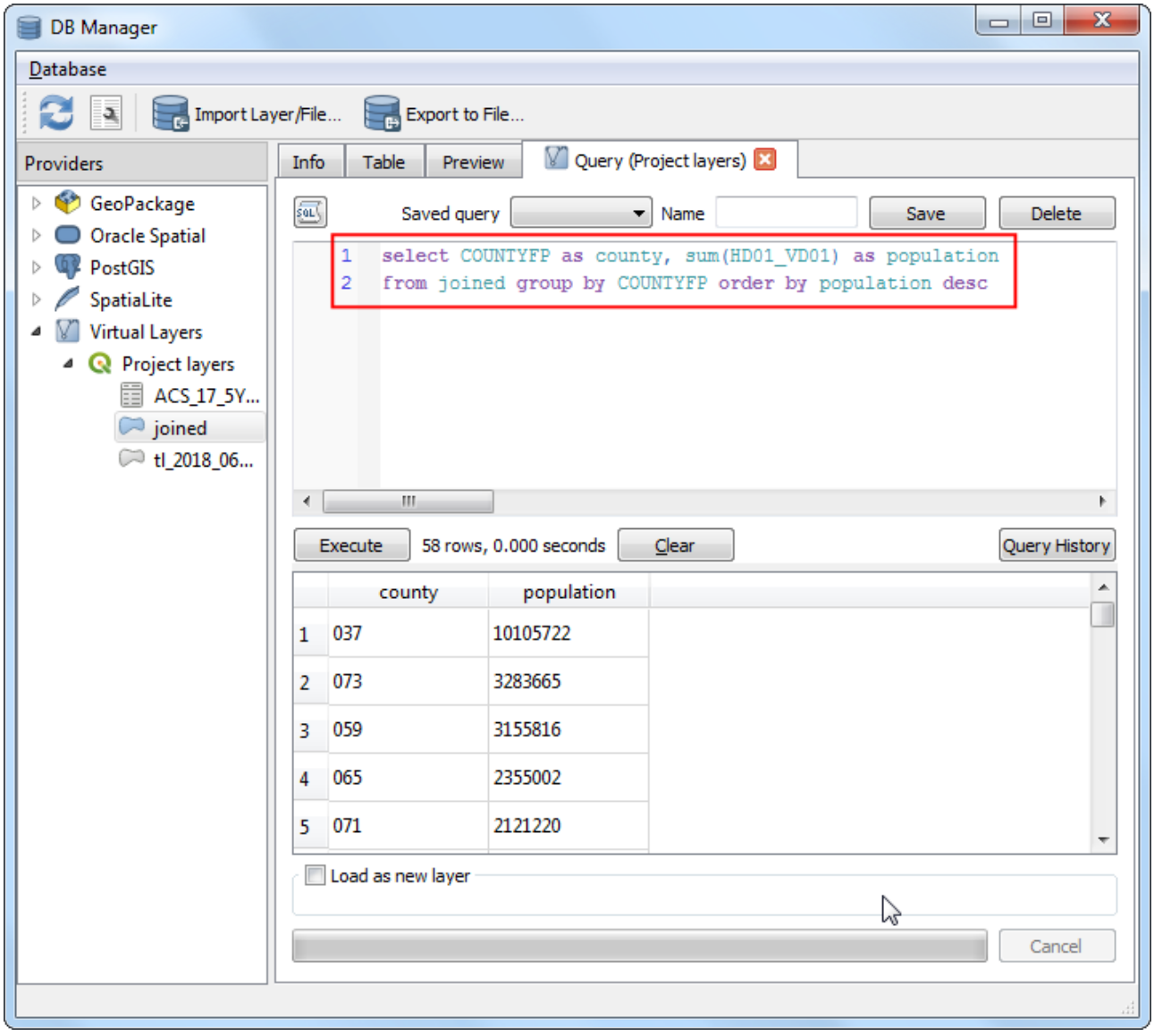
21. Type the following query that sums the `HD01_VD01` field to count the total population of the state. Enter the query in the Query tab and click Execute. The result will appear in the bottom panel. You can verify that the result matches the population of California (https://en.wikipedia.org/wiki/California).

```
select sum(HD01_VD01) as population from joined
```



22. SQL queries are also well-suited to perform group statistics. Here's a query that sums the population field but adds a `group by` clause to group all census tracts by county and create a table of total population by county. The query also sorts the result by population. We can also cross-verify that the county with FIPS id `037` (Los Angeles County) is the most populated county in california (https://en.wikipedia.org/wiki/List_of_counties_in_California) .

```
select COUNTYFP as county, sum(HD01_VD01) as population
from joined group by COUNTYFP order by population desc
```



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Performing Spatial Joins (QGIS3)

Spatial Join is a classic GIS problem - transferring attributes from one layer to another based on their spatial relationship. In QGIS, this functionality is available through the `Join attributes by location` Processing algorithm.

Overview of the task

We will use 2 layers - A shapefile of Borough boundaries of New York city and another shapefile of Street Pavement Rating for all streets in New York city. The first task will be to find the average rating of streets in each of the borough using a spatial join with summary algorithm. The second task will be to add the name of the borough to the street features through a one-to-many spatial join.

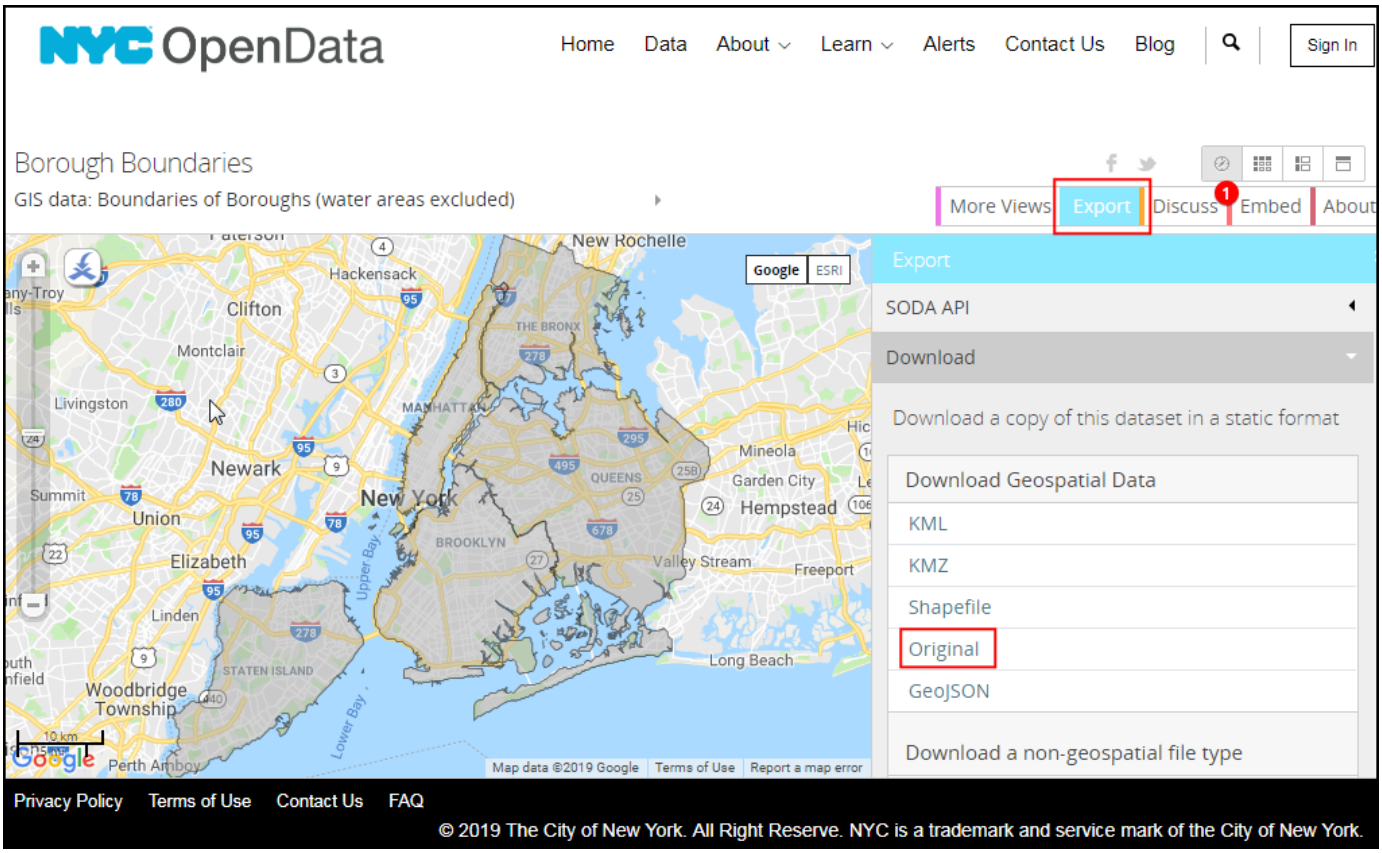
Other skills you will learn

- Creating filters to temporarily exclude certain features from computation.

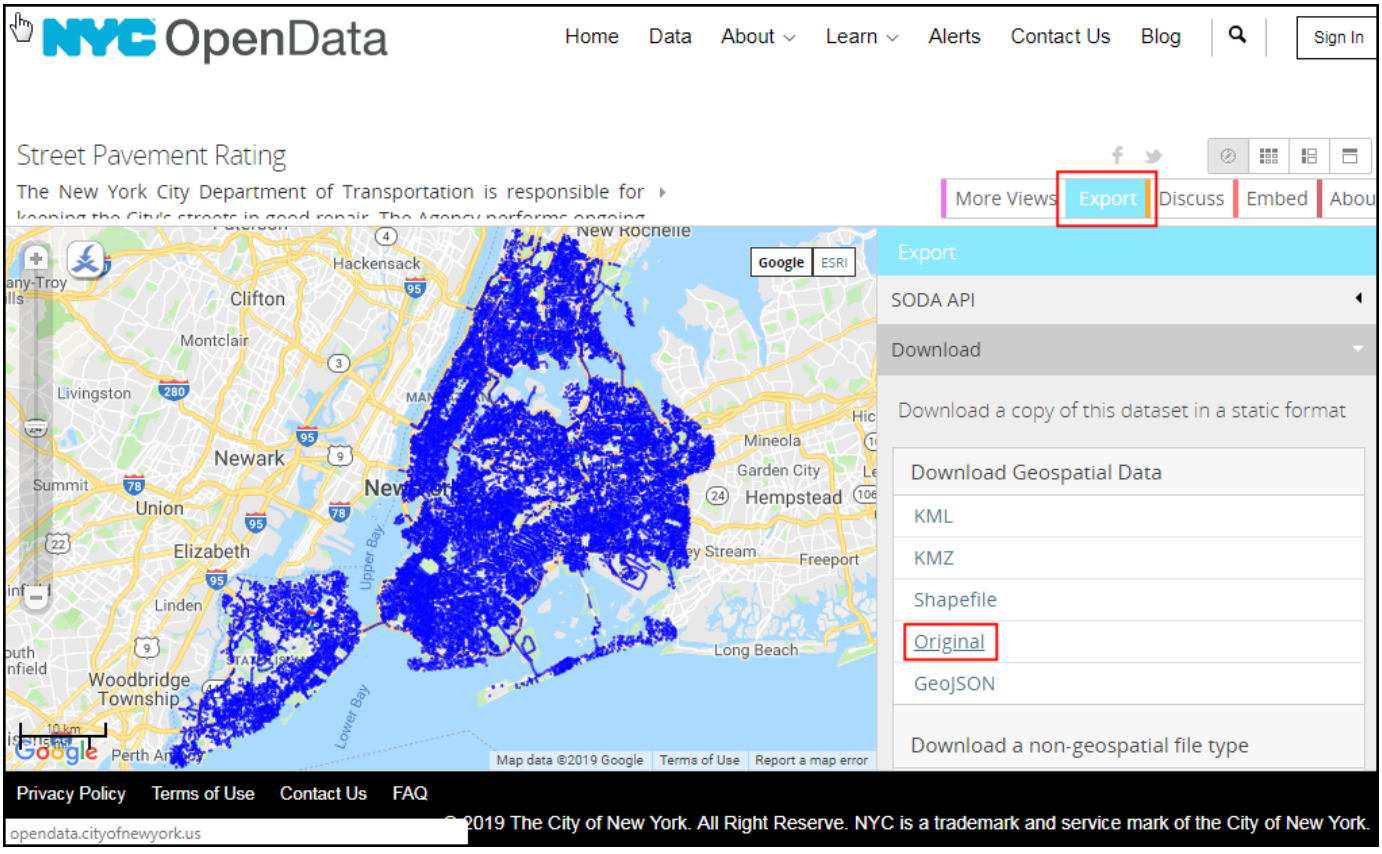
Get the data

NYC Open Data Portal (https://data.cityofnewyork.us/) is an excellent source of free data for New York city.

Download the Borough Boundaries (https://data.cityofnewyork.us/City-Government/Borough-Boundaries/tqmj-j8zm) zip file using the Export option on the portal.



Download the Street Pavement Rating (https://data.cityofnewyork.us/Transportation/Street-Pavement-Rating/2cav-chmn) zip file using the Export option on the portal.



For convenience, you may directly download a copy of the datasets from the links below:

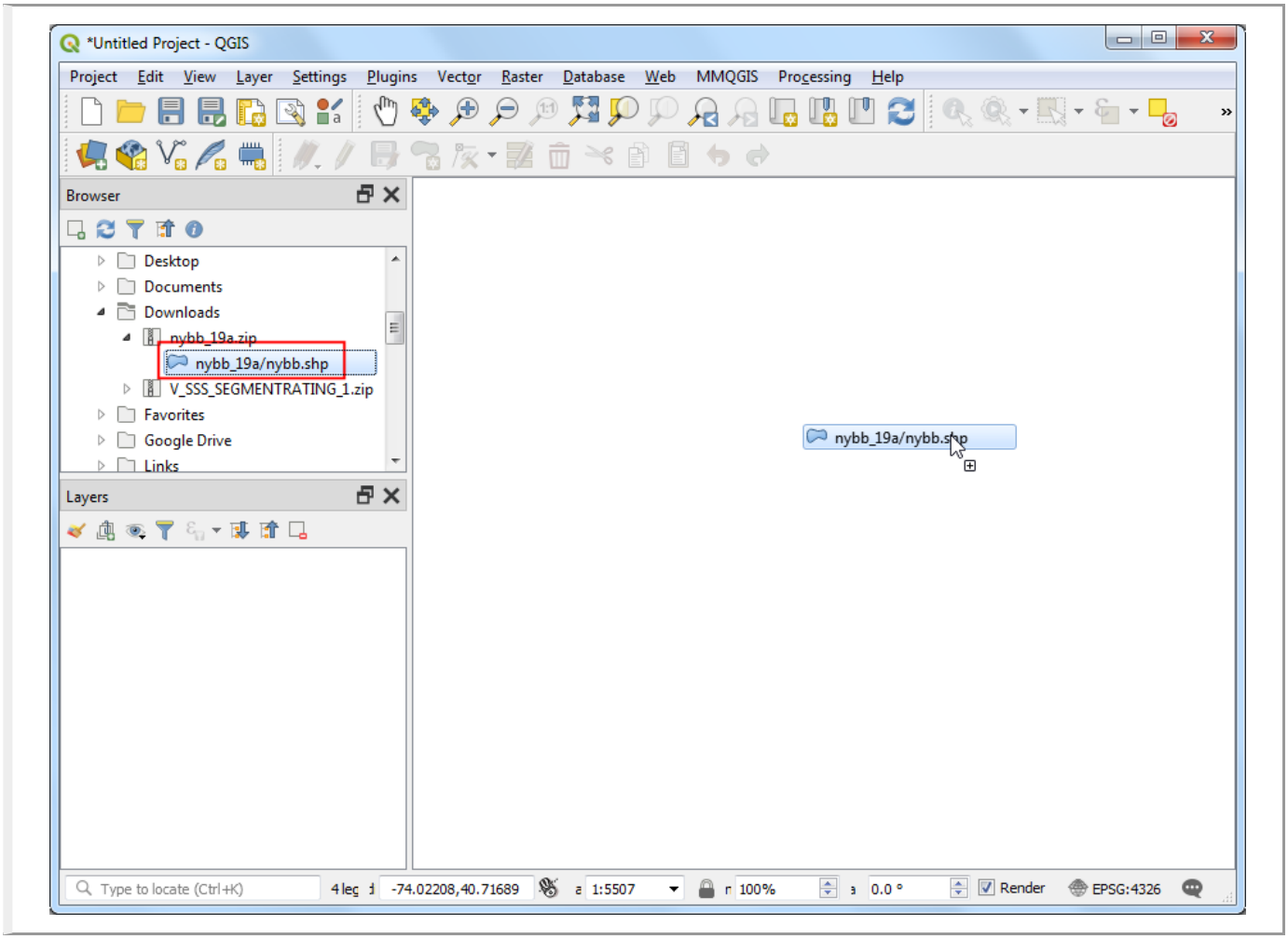
nybb_19a.zip (http://www.qgistutorials.com/downloads/nybb_19a.zip)

V_SSS_SEGMENTRATING_1.zip (http://www.qgistutorials.com/downloads/V_SSS_SEGMENTRATING_1.zip)

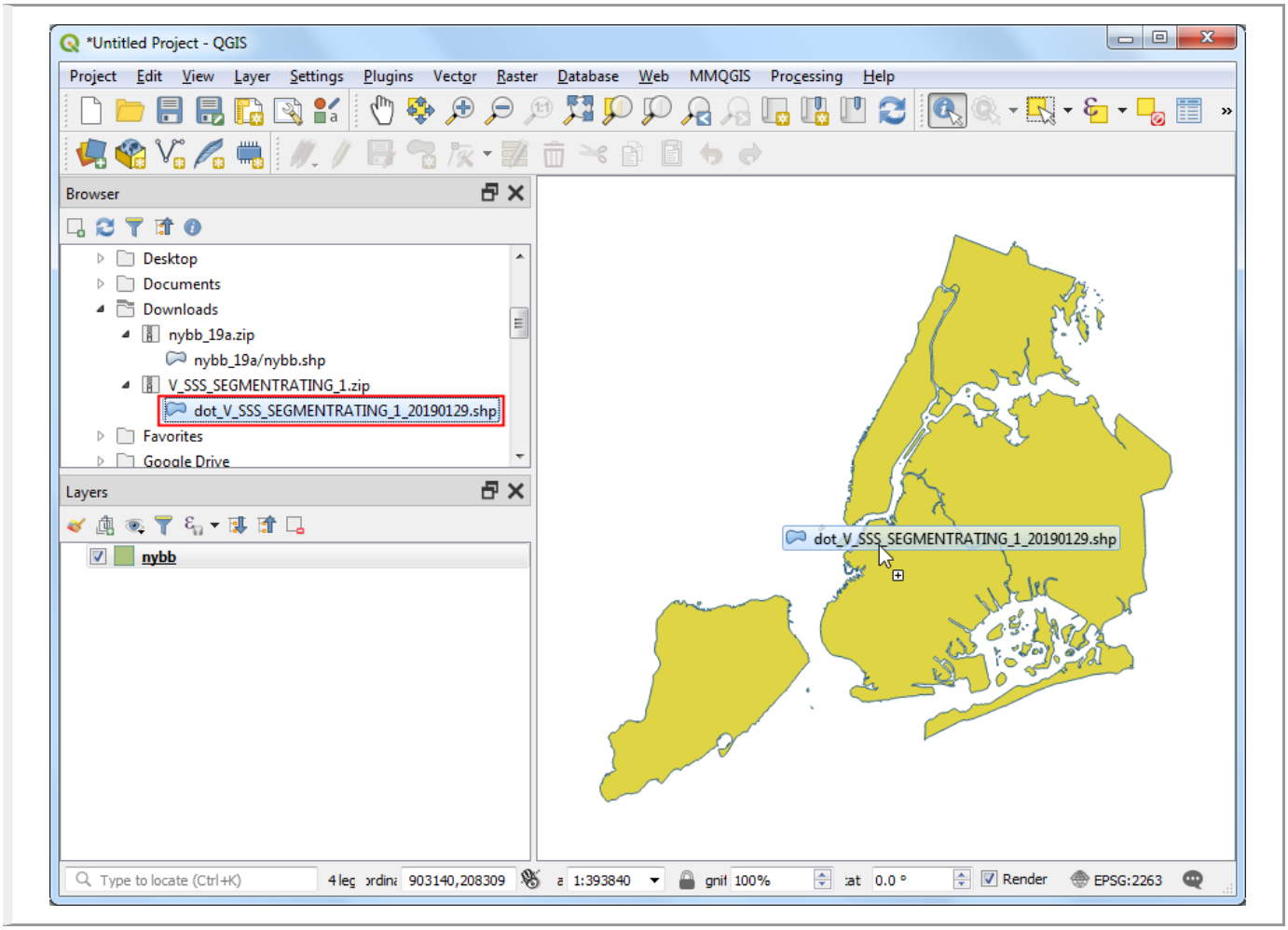
Data Source [CITYOFNY] (../credits.html#cityofny)

Procedure

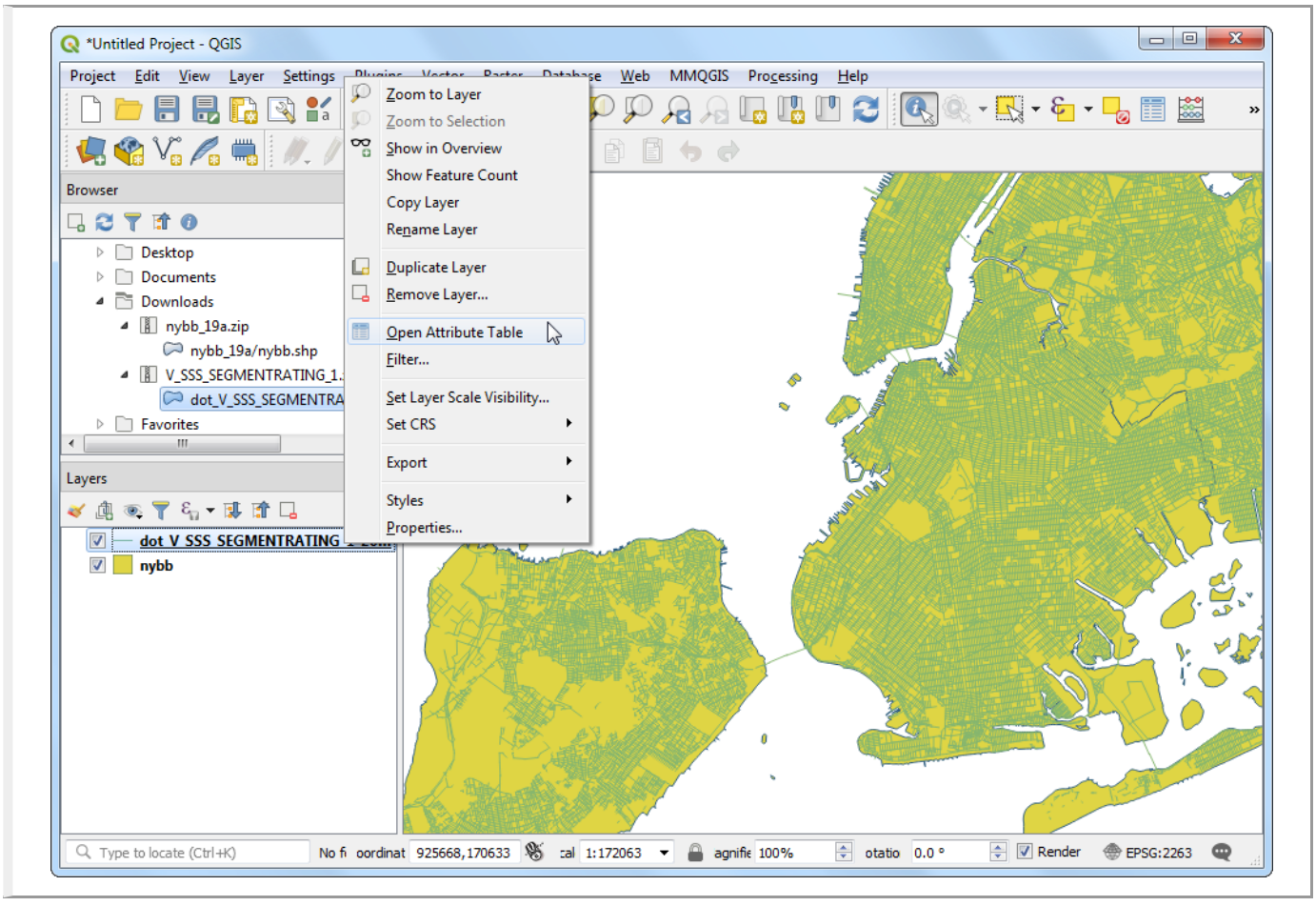
1. Locate the `nybb_19a.zip` file in the QGIS Browser and expand it. Select the `nybb_19a/nybb.shp` layer and drag it to the canvas. This is a polygon layer representing the borough boundaries in New York city.



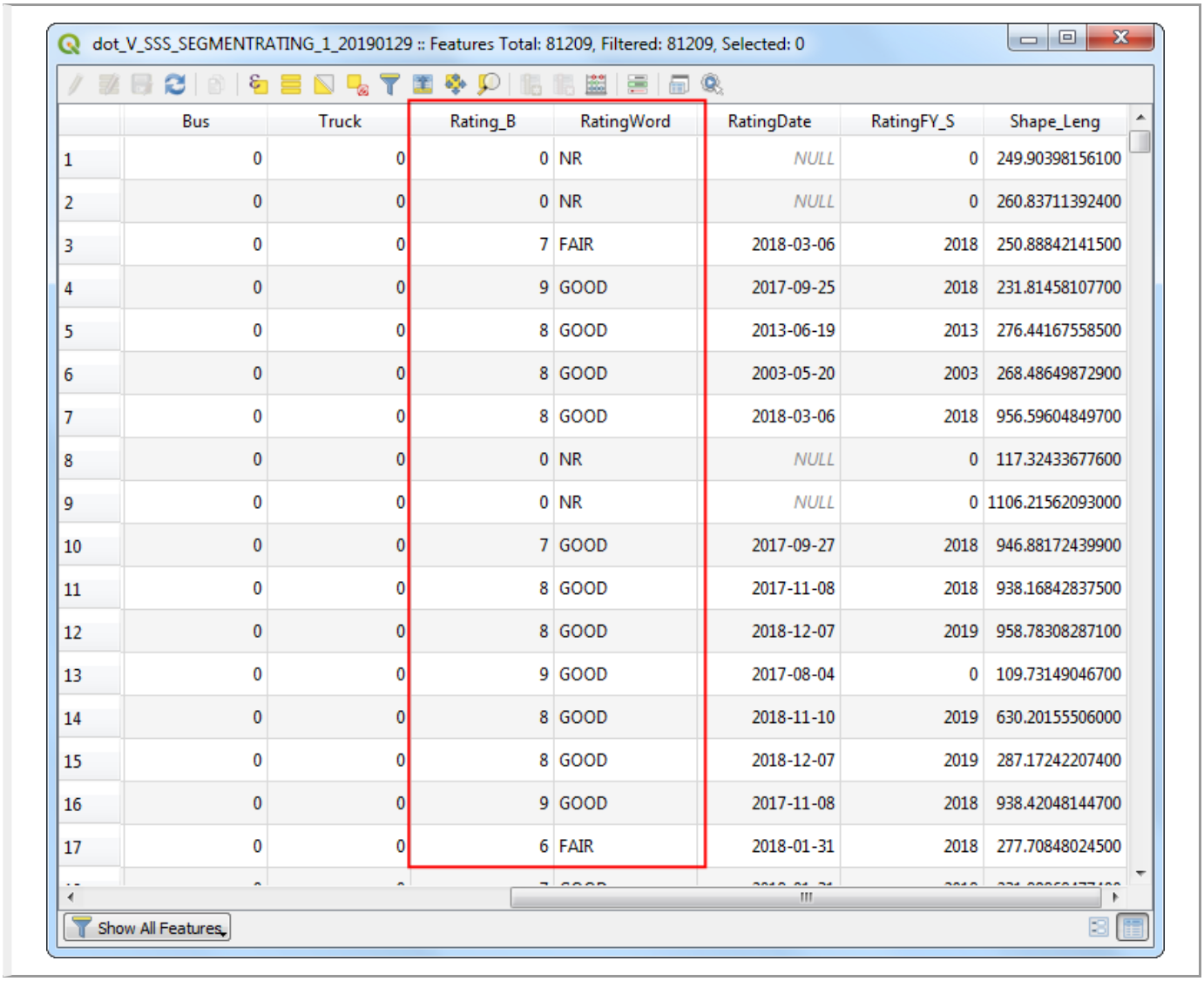
2. Next, locate the `V_SSS_SEGMENTRATING_1.zip` file and expand it. Select the `dot_V_SSS_SEGMENTRATING_1_20190129.shp` layer and add it to the canvas. This is a line layer of all streets in the city.



3. Let's examine the attributes available for each feature of the `dot_V_SSS_SEGMENTRATING_1_20190129` layer. Right-click and select Open Attribute Table.

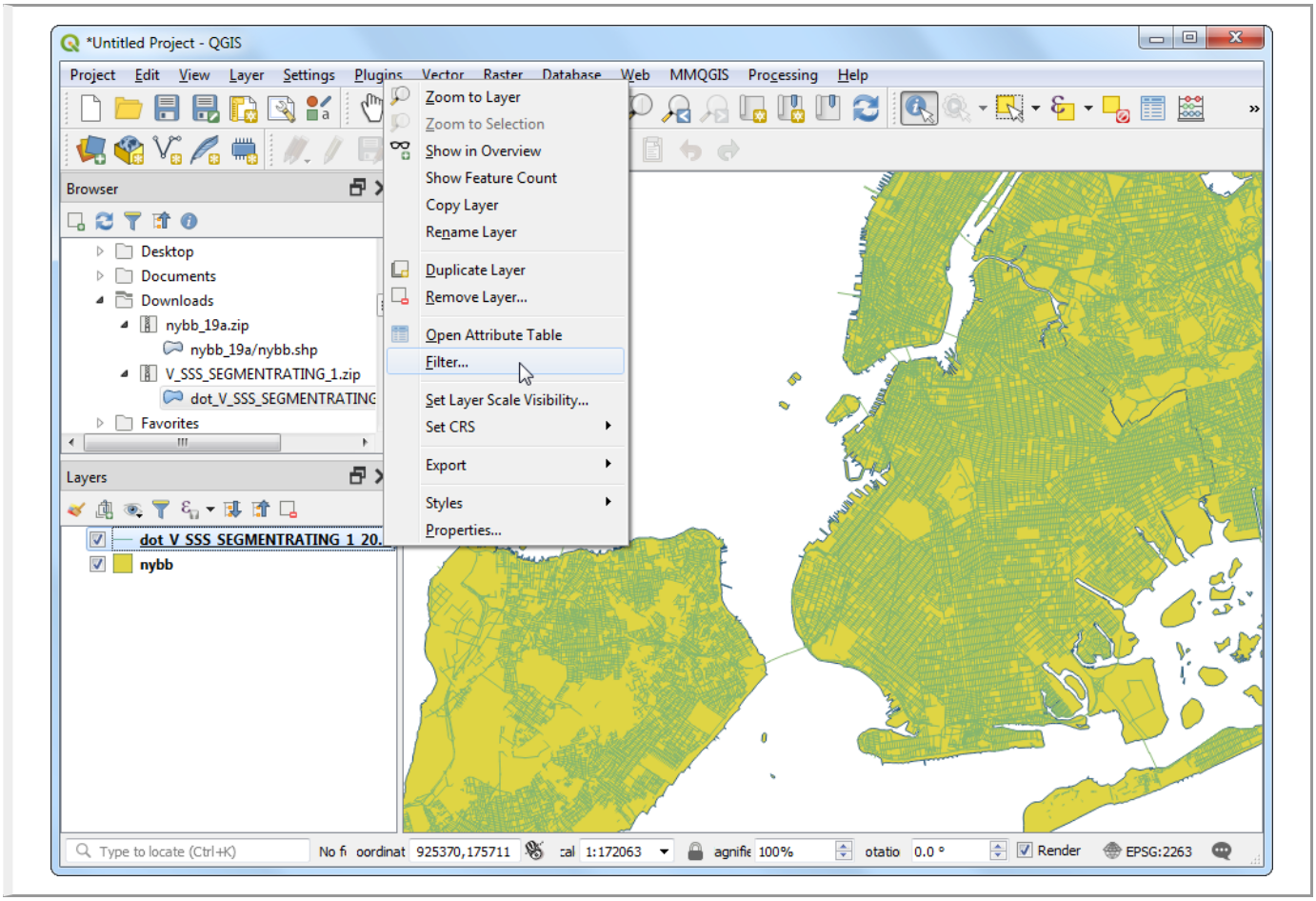


4. You will notice the attribute called `Rating_B` which has values in the range 0-10 representing the street segment's rating. The attribute `RatingWord` has descriptive rating. We can use the `Rating_B` field to calculate the average rating.



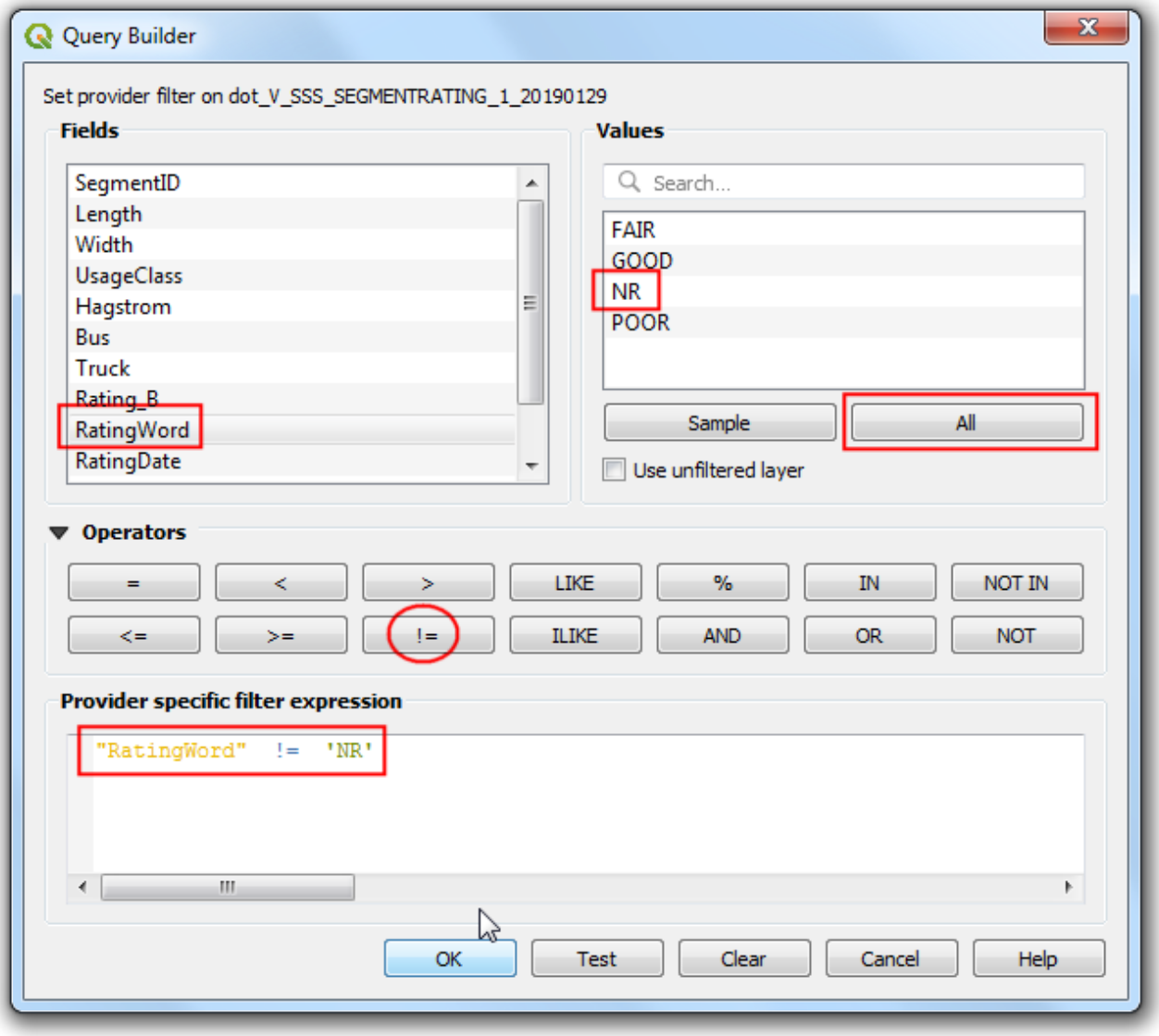
	Bus	Truck	Rating_B	RatingWord	RatingDate	RatingFY_S	Shape_Leng
1	0	0	0	NR	NULL	0	249.90398156100
2	0	0	0	NR	NULL	0	260.83711392400
3	0	0	7	FAIR	2018-03-06	2018	250.88842141500
4	0	0	9	GOOD	2017-09-25	2018	231.81458107700
5	0	0	8	GOOD	2013-06-19	2013	276.44167558500
6	0	0	8	GOOD	2003-05-20	2003	268.48649872900
7	0	0	8	GOOD	2018-03-06	2018	956.59604849700
8	0	0	0	NR	NULL	0	117.32433677600
9	0	0	0	NR	NULL	0	1106.21562093000
10	0	0	7	GOOD	2017-09-27	2018	946.88172439900
11	0	0	8	GOOD	2017-11-08	2018	938.16842837500
12	0	0	8	GOOD	2018-12-07	2019	958.78308287100
13	0	0	9	GOOD	2017-08-04	0	109.73149046700
14	0	0	8	GOOD	2018-11-10	2019	630.20155506000
15	0	0	8	GOOD	2018-12-07	2019	287.17242207400
16	0	0	9	GOOD	2017-11-08	2018	938.42048144700
17	0	0	6	FAIR	2018-01-31	2018	277.70848024500

5. You may have notice some features have a rating of `NR` . These are the segments that were not rated. Including them in our analysis will not be correct. Before we do the spatial join, let's set up a **Filter** to exclude these records. Right-click the `dot_V_SSS_SEGMENTRATING_1_20190129` layer and select Filter.

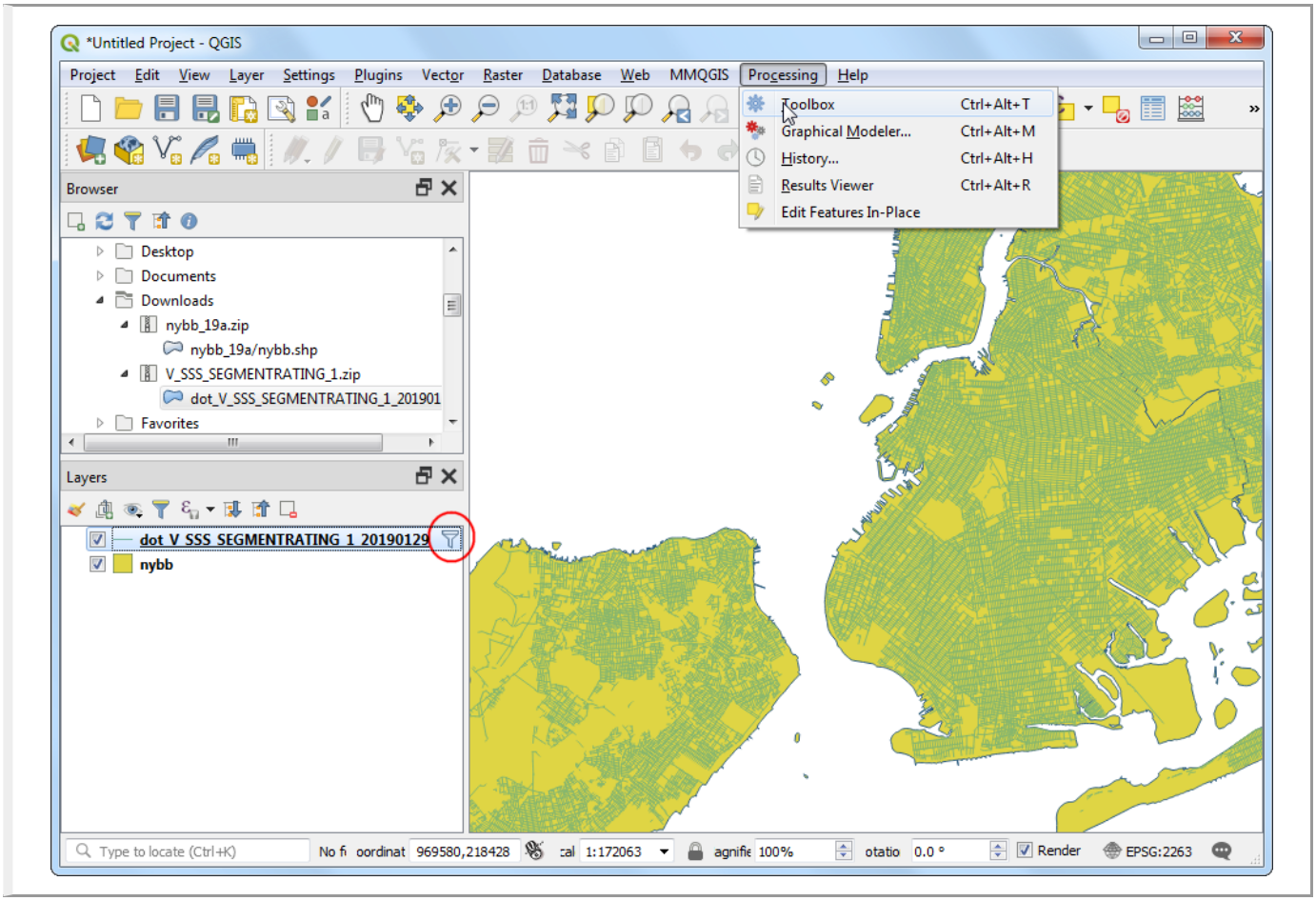


6. In the Query Builder, type the following expression to select all records that are not rated `NR`. You can also build the expression interactively by clicking on Field, Operator and selecting the appropriate Value. Click OK.

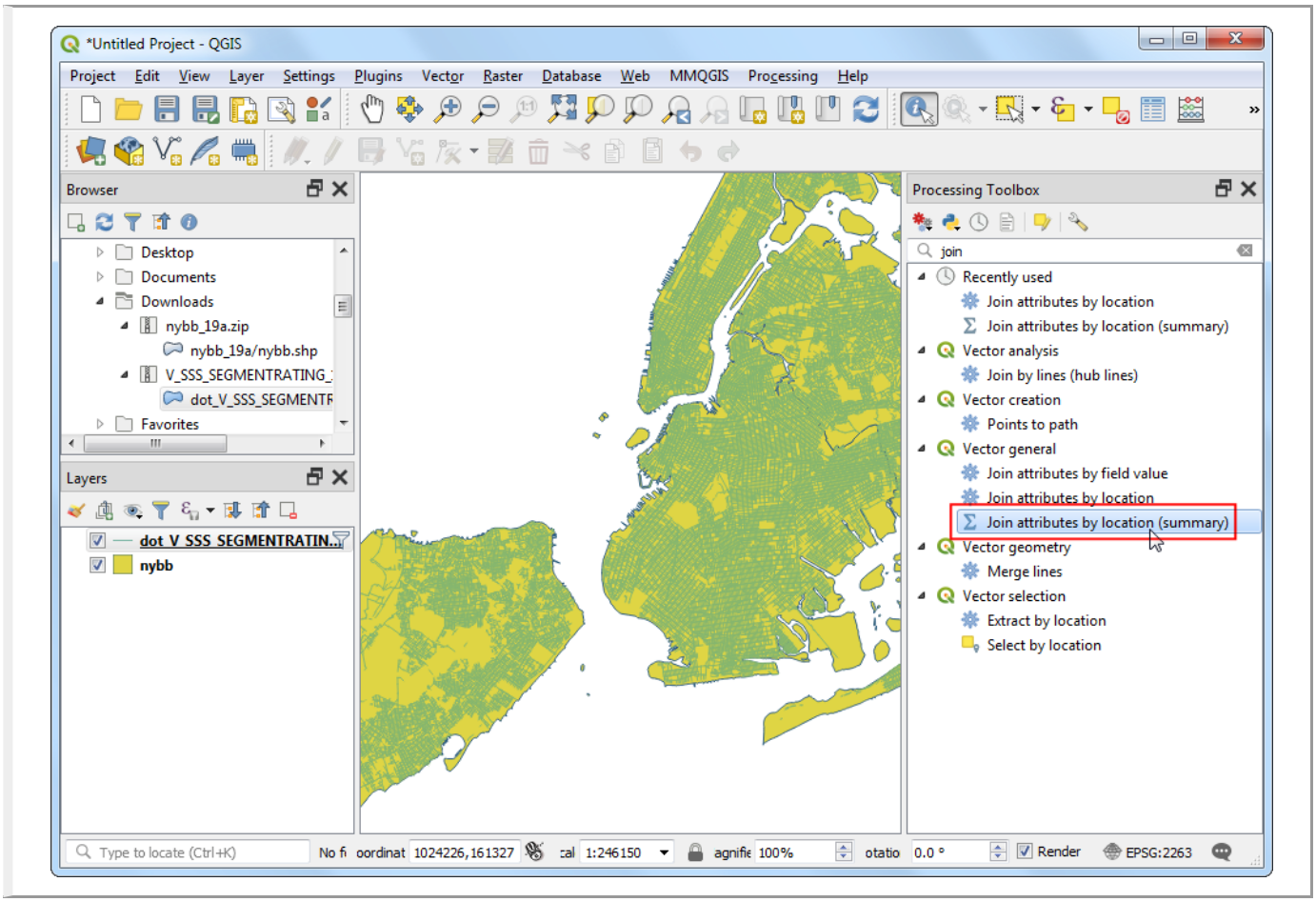
```
"RatingWord" != 'NR'
```



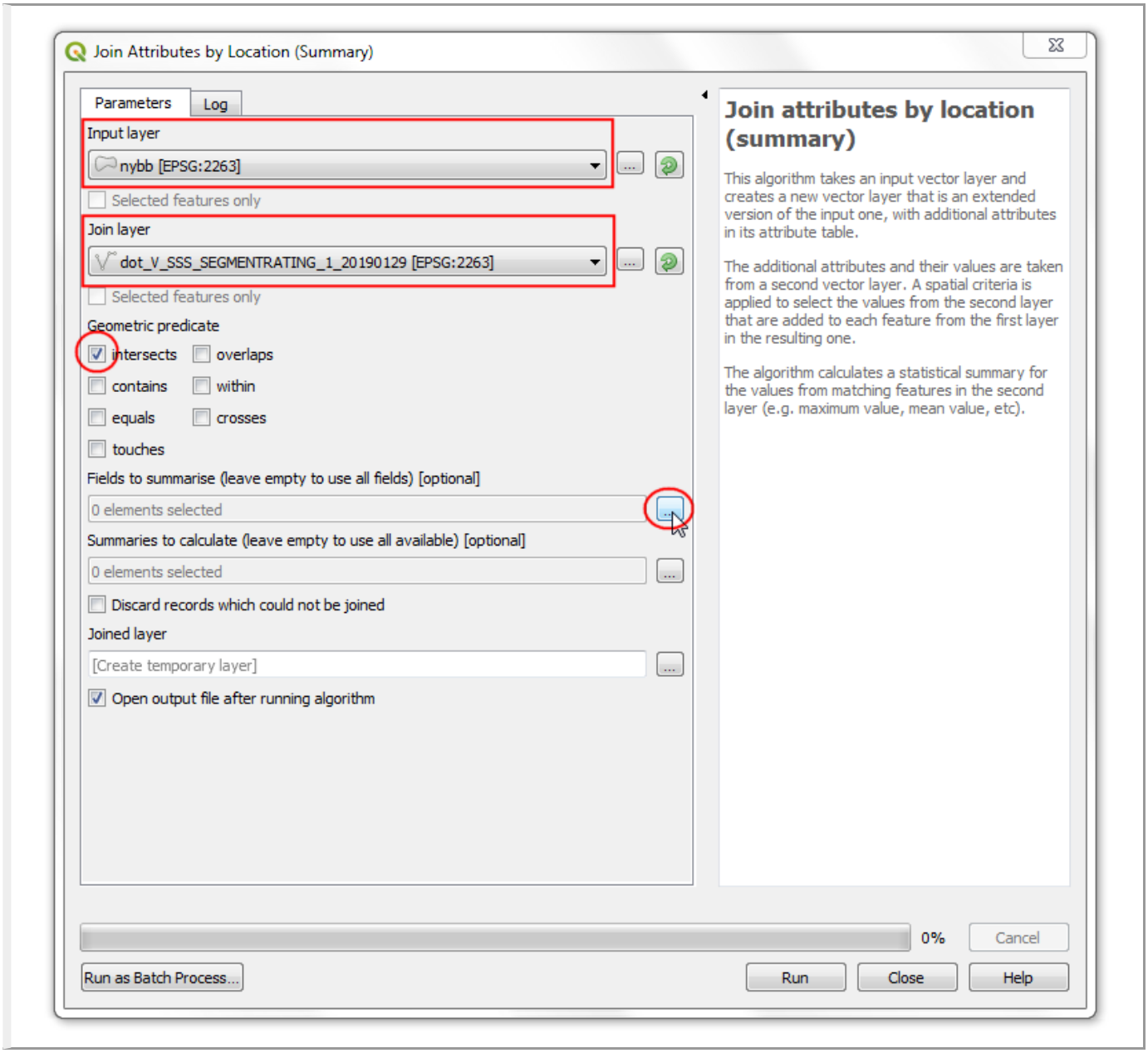
7. You will notice the `dot_V_SSS_SEGMENTRATING_1_20190129` layer now has a filter icon indicating that there is an active filter applied to this layer. Now we can do a spatial join using this layer. Go to Processing ▸ Toolbox.



8. Search and locate the Vector general ‣ Join attribute by location (summary) algorithm. Double-click to launch it.



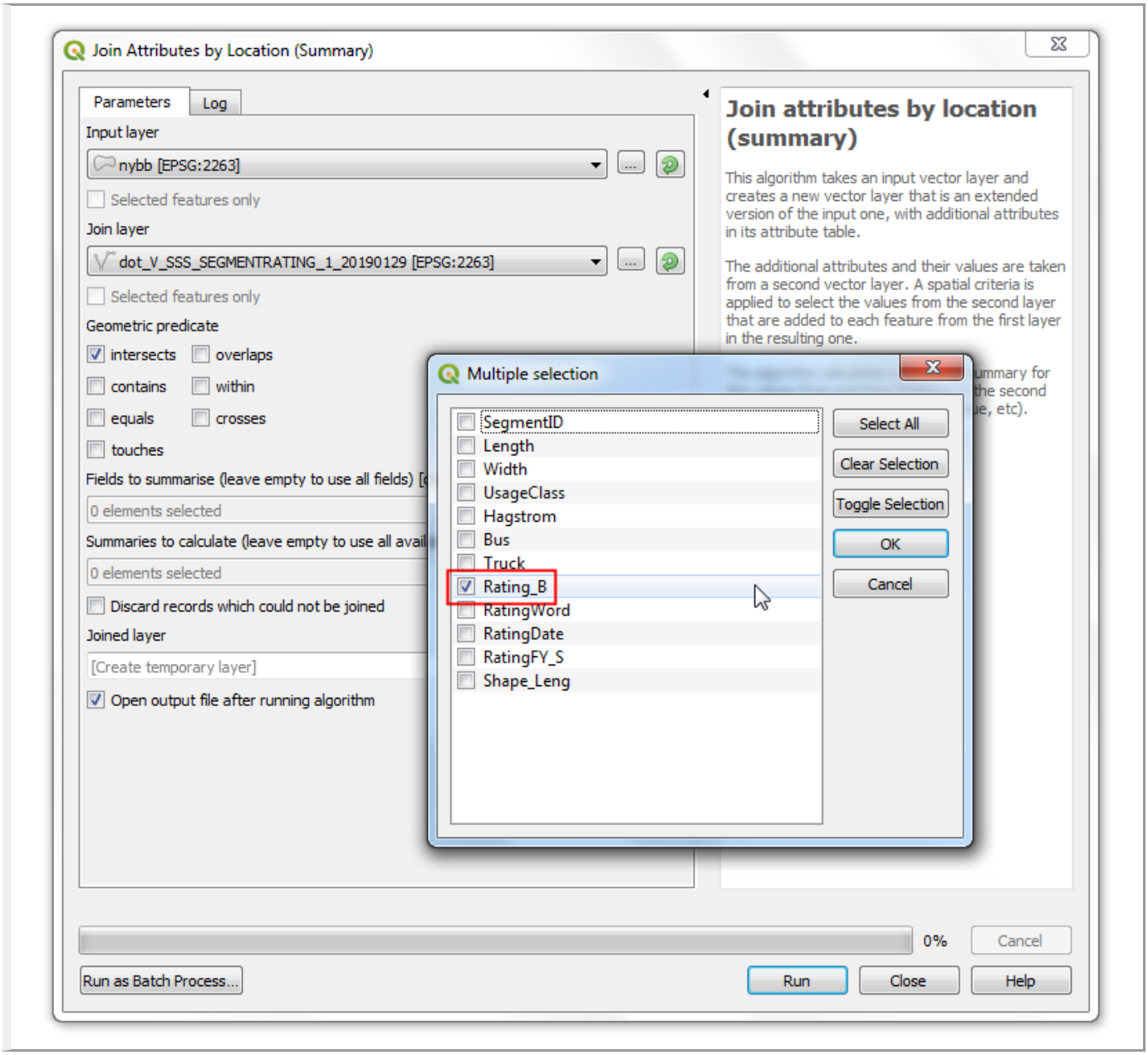
9. In the Join attribute by location (summary) dialog, select `nybb` as the Input layer. The street layer `dot_V_SSS_SEGMENTRATING_1_20190129` will be the Join layer. You can leave the Geometry predicate to the default `Intersects`. Click the ... button next to Fields to sumarize.



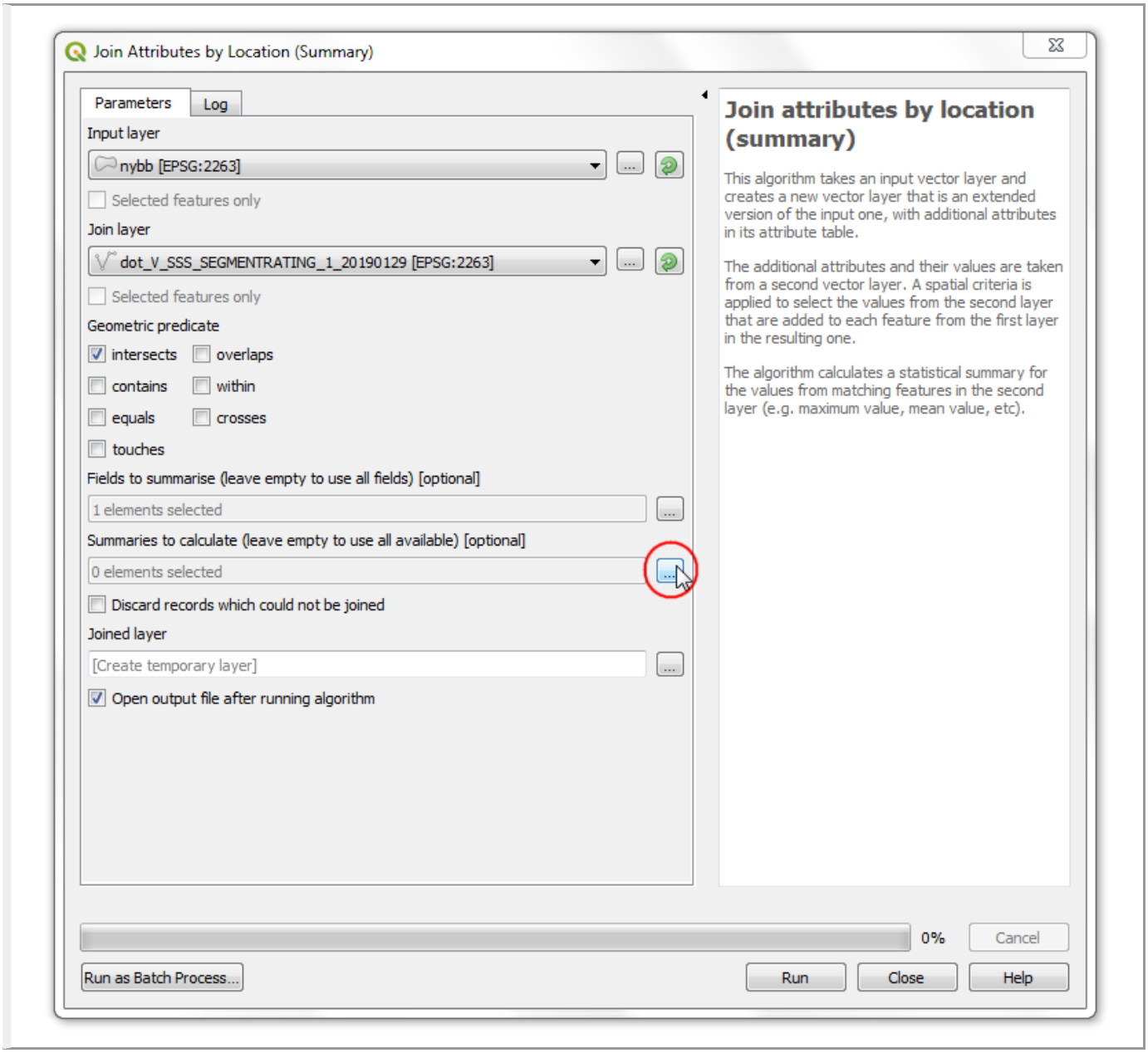
Note

A tip to help you select the correct input and join layers: The input layer is the one that will be modified will new attributes in the spatial join. As we want the average rating field to be added to the borough layer, it will be the input layer.

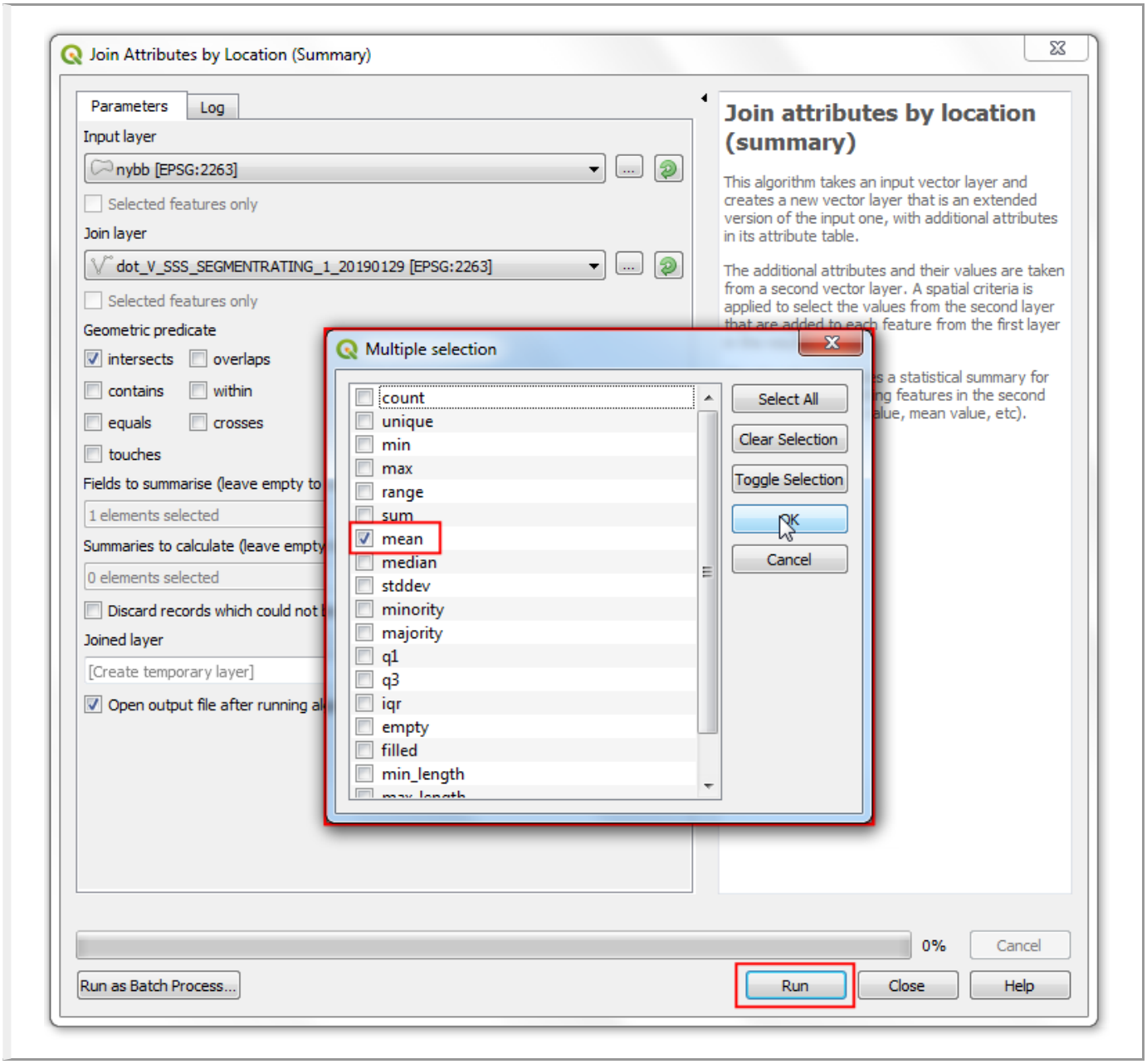
10. Select `Rating_B` and click OK.



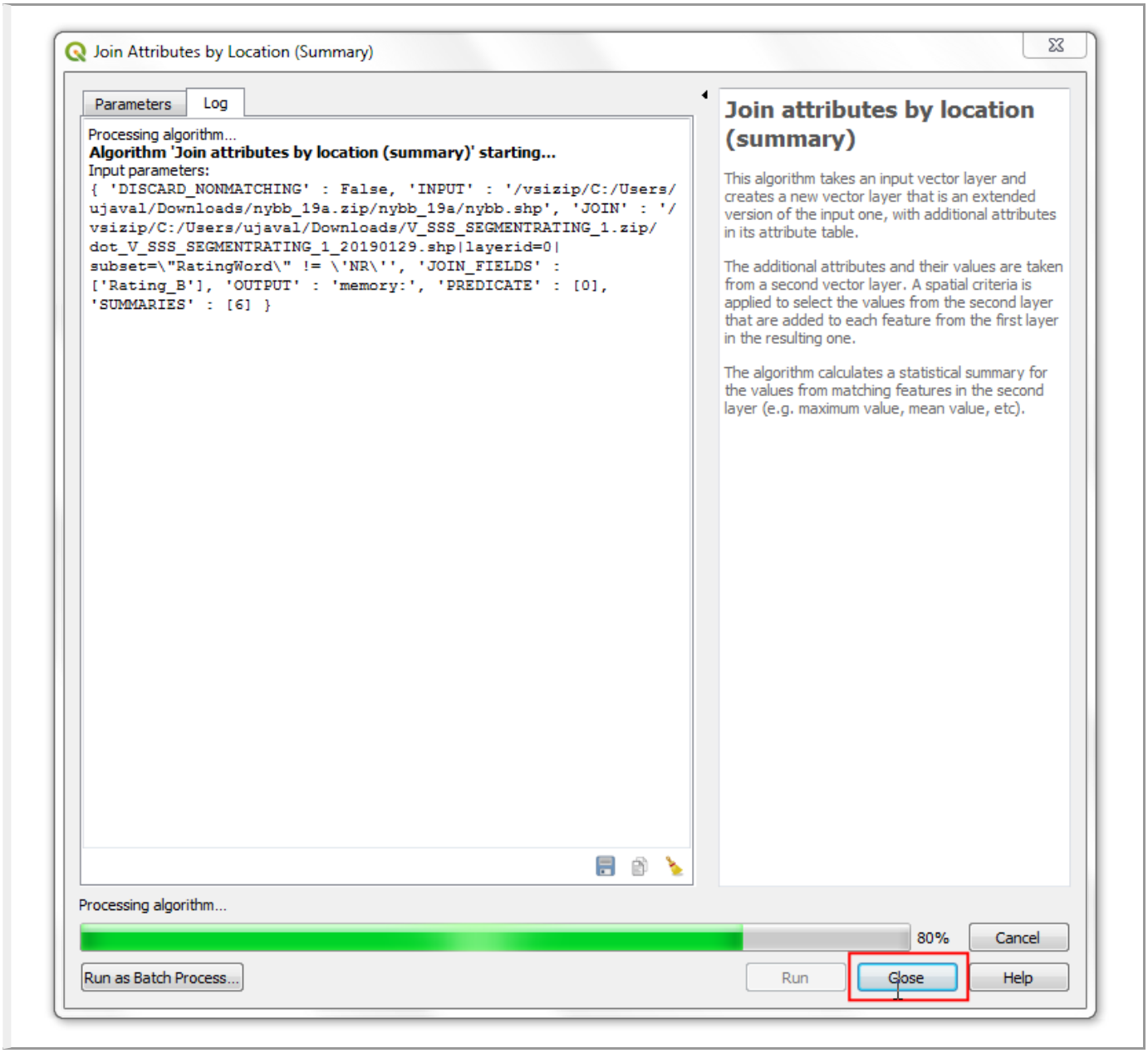
11. Similarly, click the … button next to Summaries to calculate.



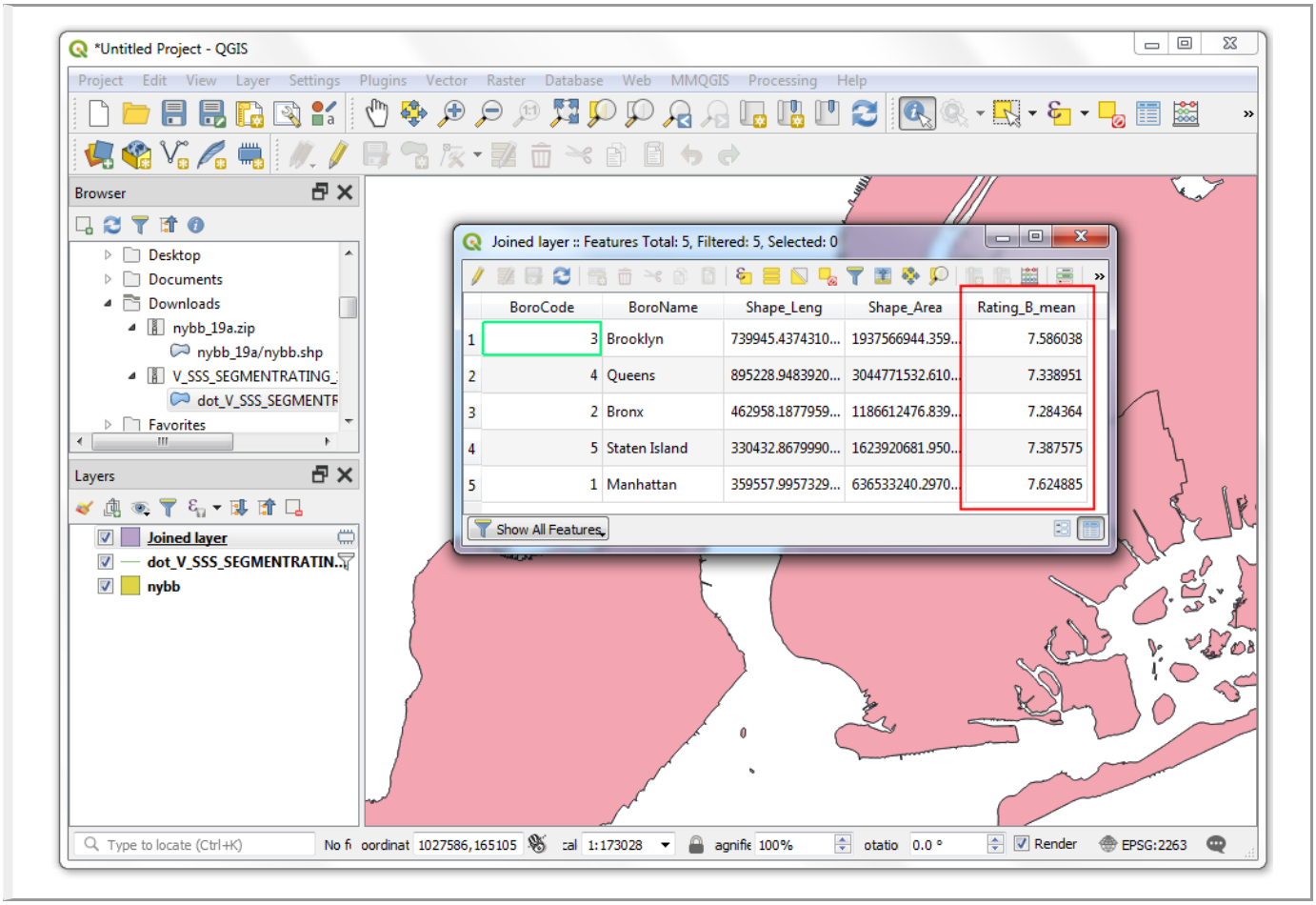
12. Select `mean` as the summary operator and click OK. Now we are ready to start the processing. Click Run.



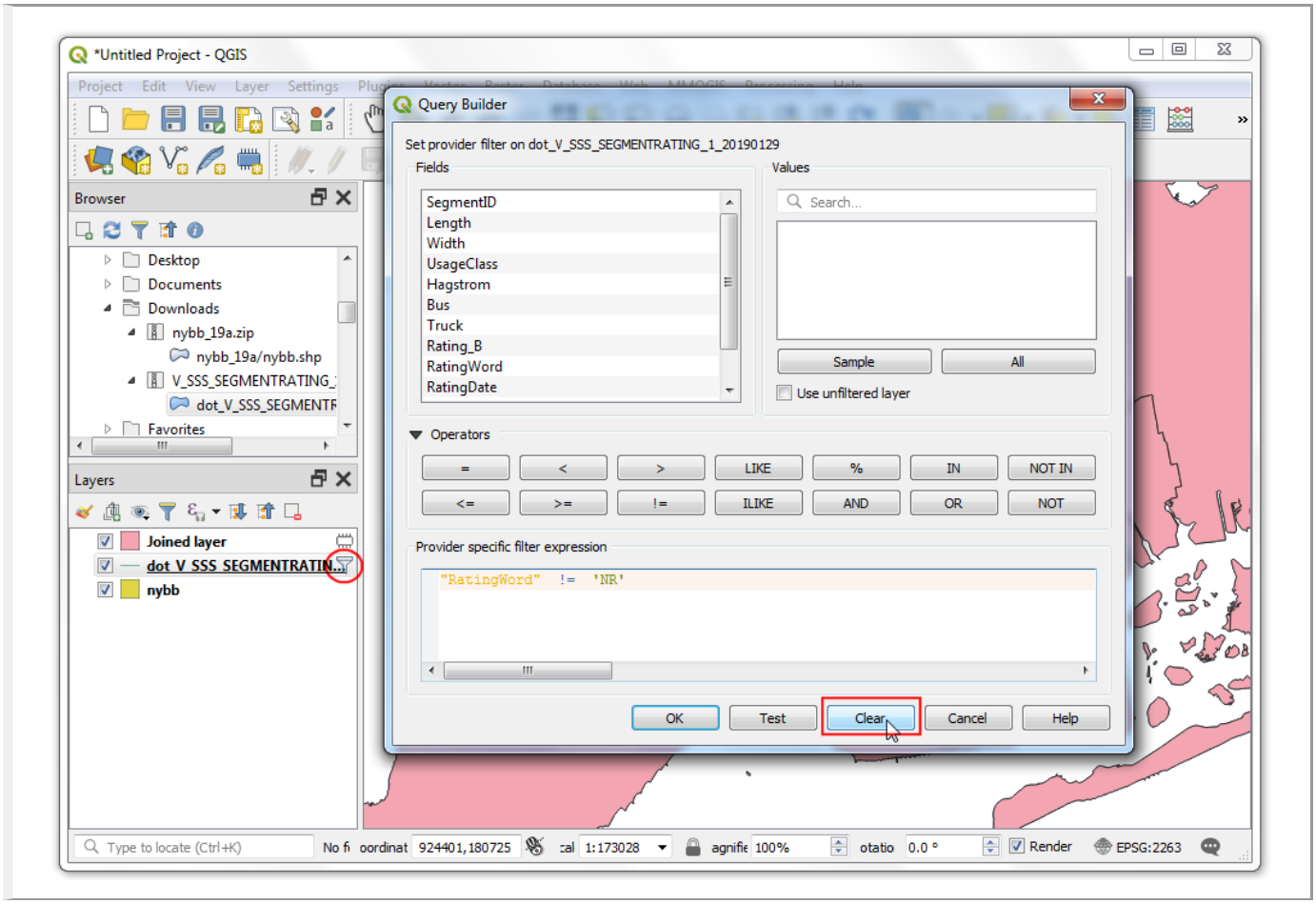
13. The processing algorithm will work through the features and apply the spatial join. Verify that the processing job was successful and click Close.



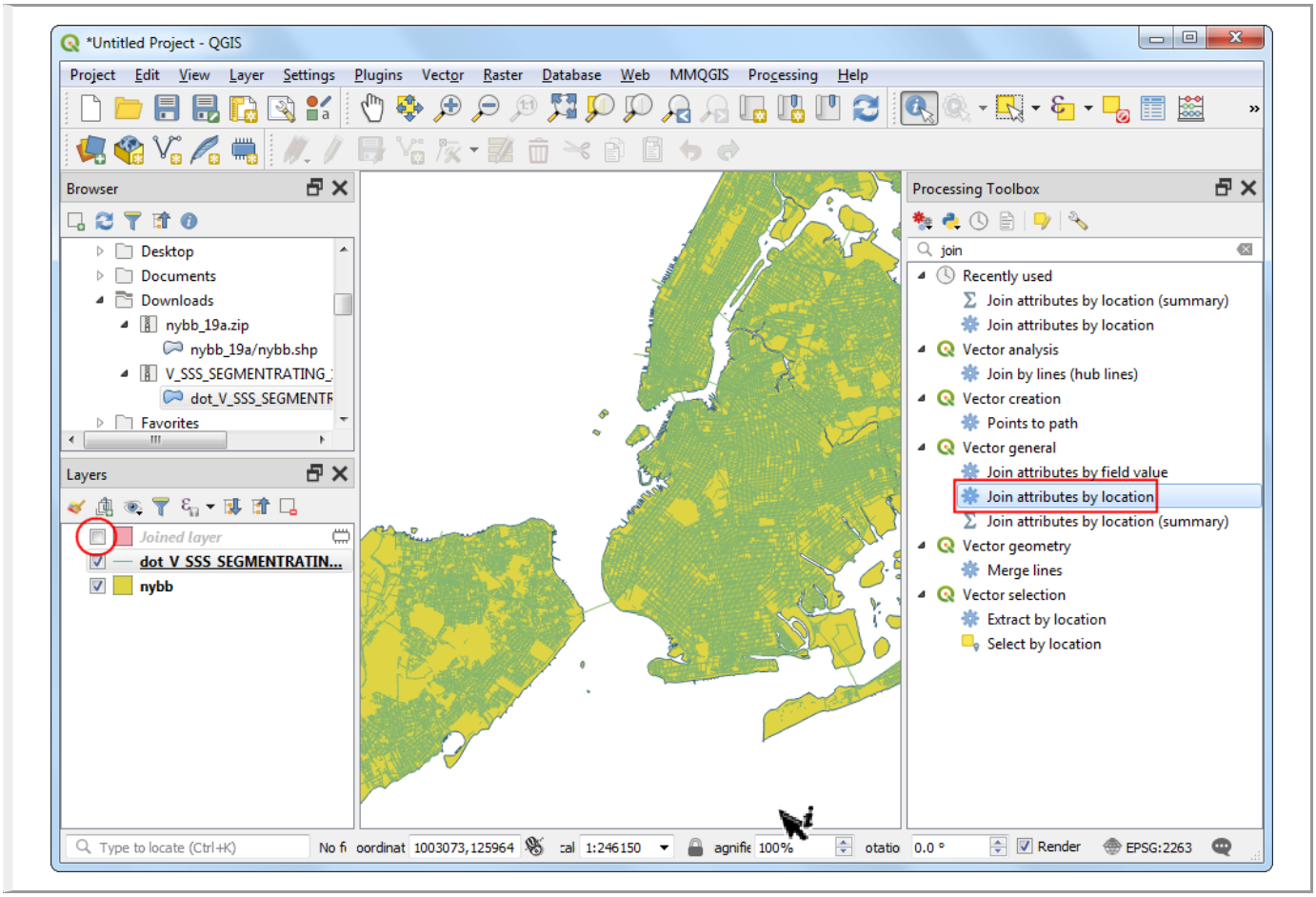
14. Back in the main QGIS window, you will see a new `Joined layer` layer added to canvas. Open the attribute table for this layer. You will see a new column `Rating_B_mean` is added to the input borough layer with the average rating of all streets that are interesecting with that feature.



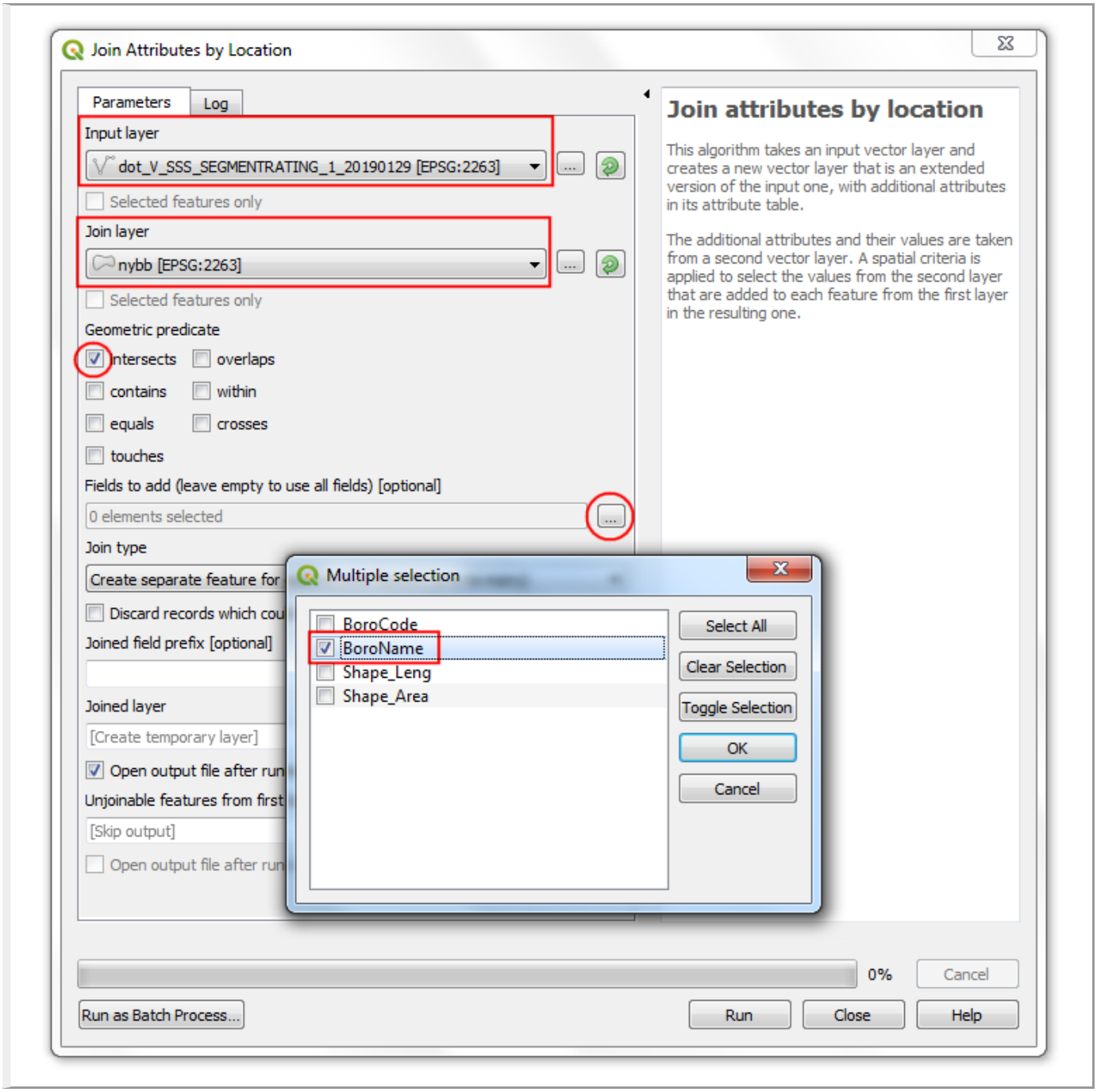
15. Now we can perform a reverse operation. Sometimes your analysis requires getting attributes from another layer based on the spatial relationship but not compute any summary. We can use the `Join attribute by location` algorithm for such analysis. The task is to add the name of the borough to each feature in the streets layer based on which borough polygon it intersects with. Before we run this algorithm, let's remove the filter from the `dot_V_SSS_SEGMENTRATING_1_20190129` layer. Click the filter icon and press the Clear in the Query Builder. Click OK.



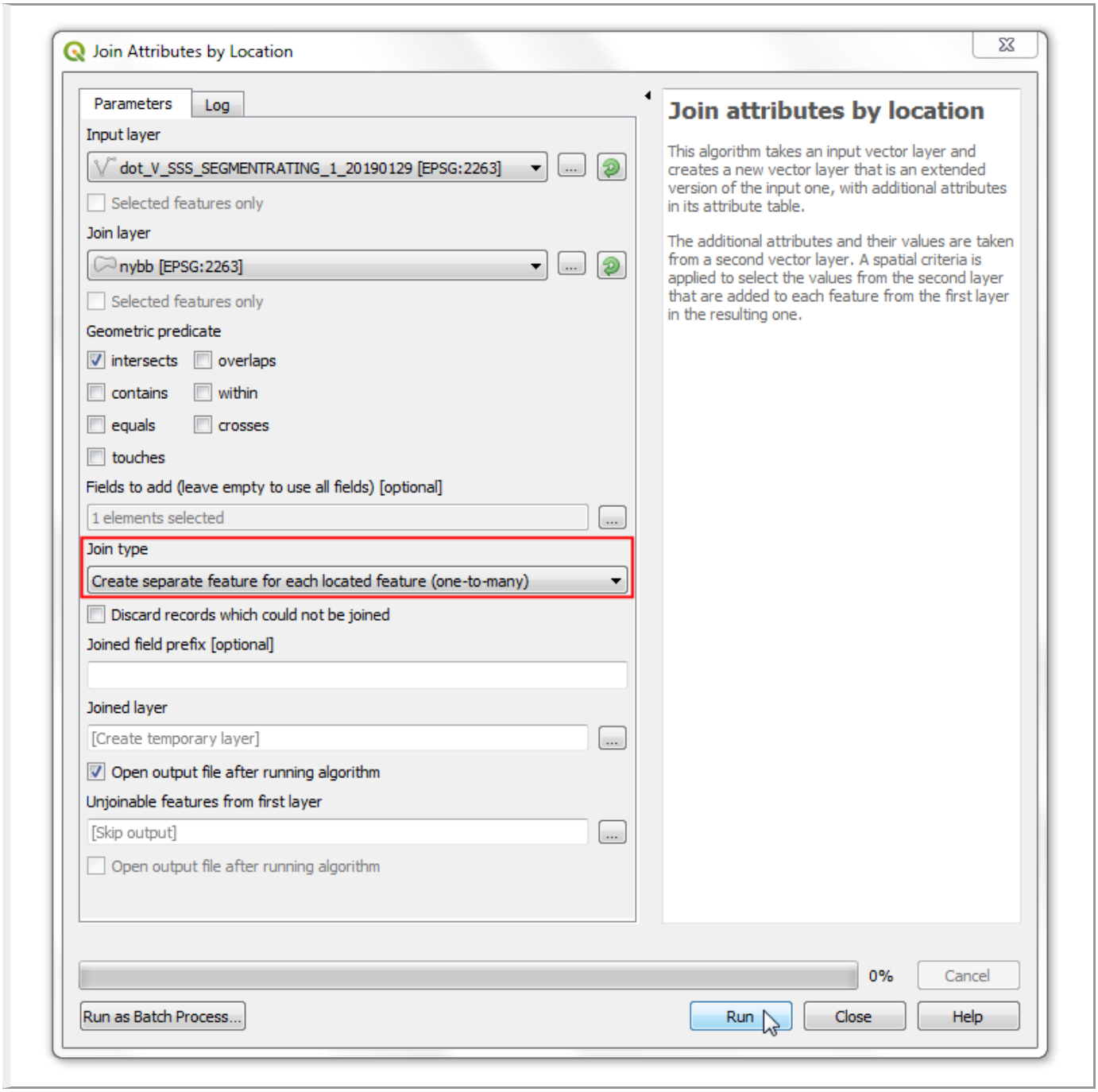
16. Turn of the `Joined layer` in the Layers panel. Find the Vector general ‣ Join attribute by location algorithm in the Processing Toolbox and double-click it to launch.



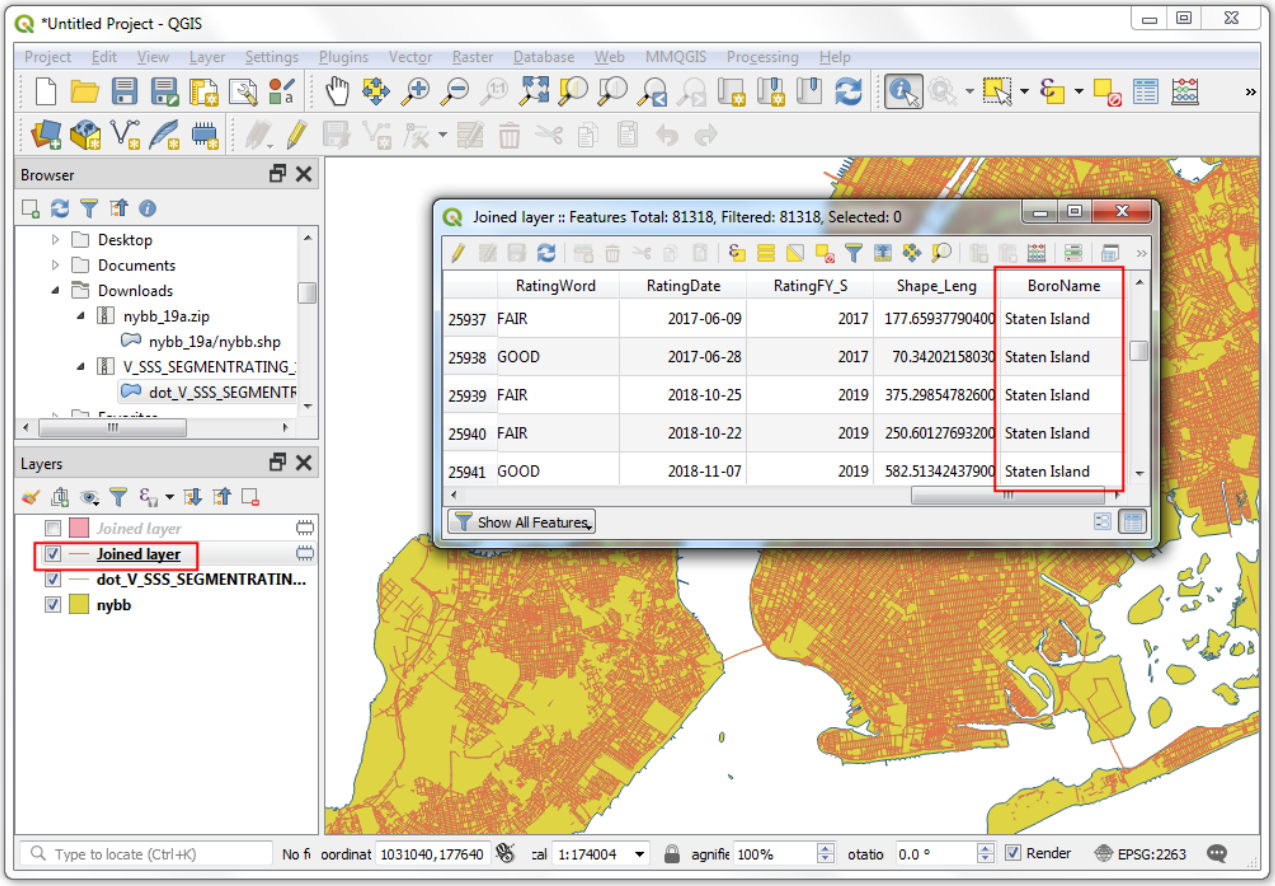
17. Select `dot_V_SSS_SEGMENTRATING_1_20190129` as the Input layer and `nybb` as the Join layer. You can leave the Geometry predicate to the default `Intersects`. Click the ... button next to Fields to add and select `BoroName`. Click OK.



18. The line segment may cross a borough boundary, so we choose the Join type as `Crate separate feature for each located feature (one-to-many)`. Click Run.



19. Once the processing finishes, open the attribute table of the newly added `Joined layer`. You will see that there is a new `BoroName` attribute added to each street feature.



Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Creating Heatmaps (QGIS3)

Heatmaps are one of the best visualization tools for dense point data. Heatmap is an interpolation technique that useful in determining density of input features. Heatmaps are most commonly used to visualize crime data, traffic incidents, housing density etc. QGIS has a heatmap renderer that can be used to style a point layer and a Processing algorithm **Heatmap (Kernel Density Estimation)** that can be used to create an raster from a point layer.

Overview of the task

We will work with a dataset of crime locations in Surrey, UK and create a heatmap to visualize regions with high density of crime.

Other skills you will learn

- Using virtual fields and conditional expressions

Get the data

data.police.uk (https://data.police.uk) provides street-level crime, outcome, and stop and search data in simple CSV format. Download the data for Surrey Police (https://data.police.uk/data/) and unzip the downloaded archive to extract the CSV file.

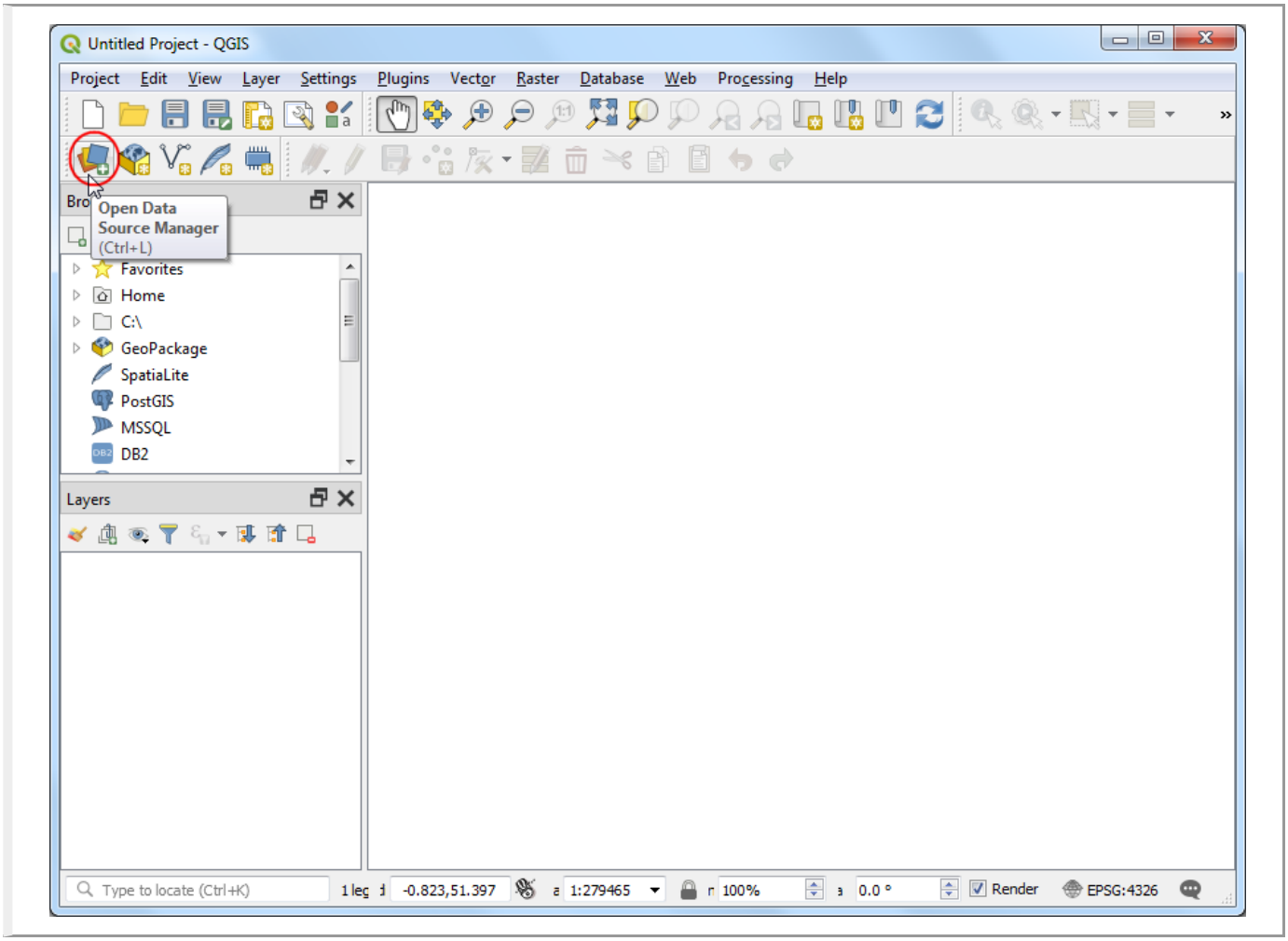
For convenience, you may directly download a copy of the dataset from the link below:

2019-02-surrey-street.csv (http://www.qgistutorials.com/downloads/2019-02-surrey-street.csv)

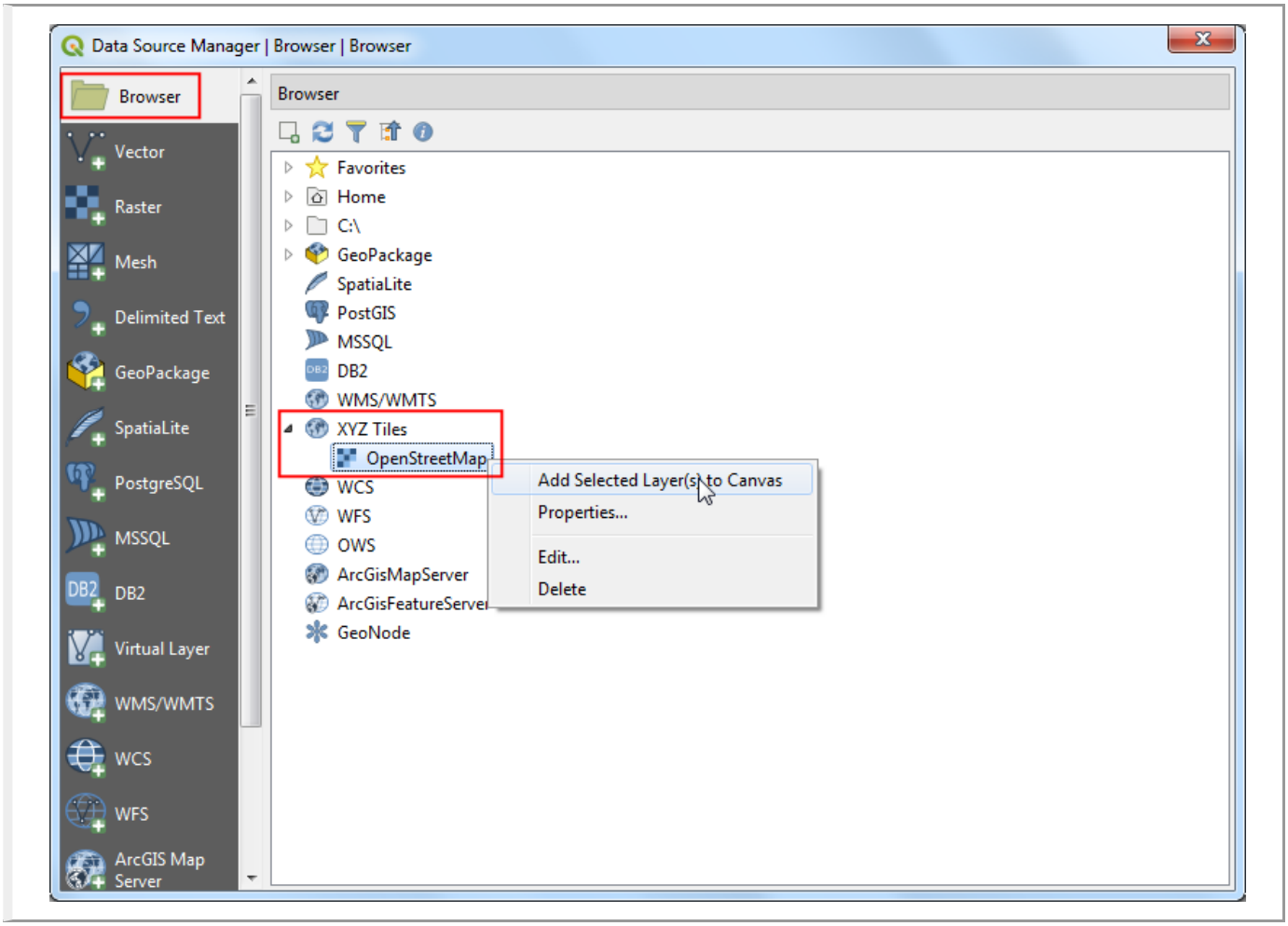
Data Source [POLICEUK] (../credits.html#policeuk)

Procedure

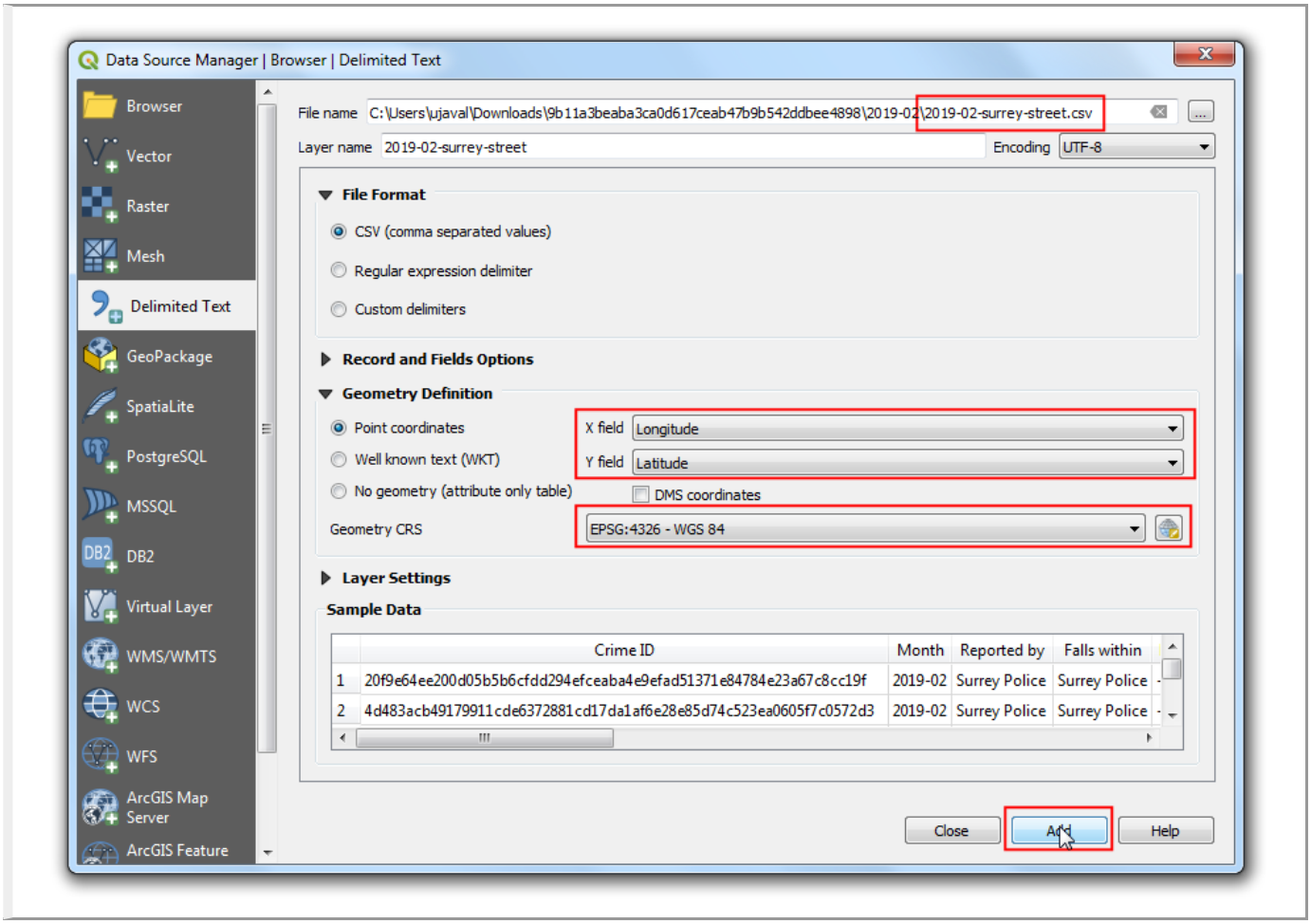
1. We will first load a basemap layer from OpenStreetMap and then import the CSV data. Click the Open Data Source Manager button.



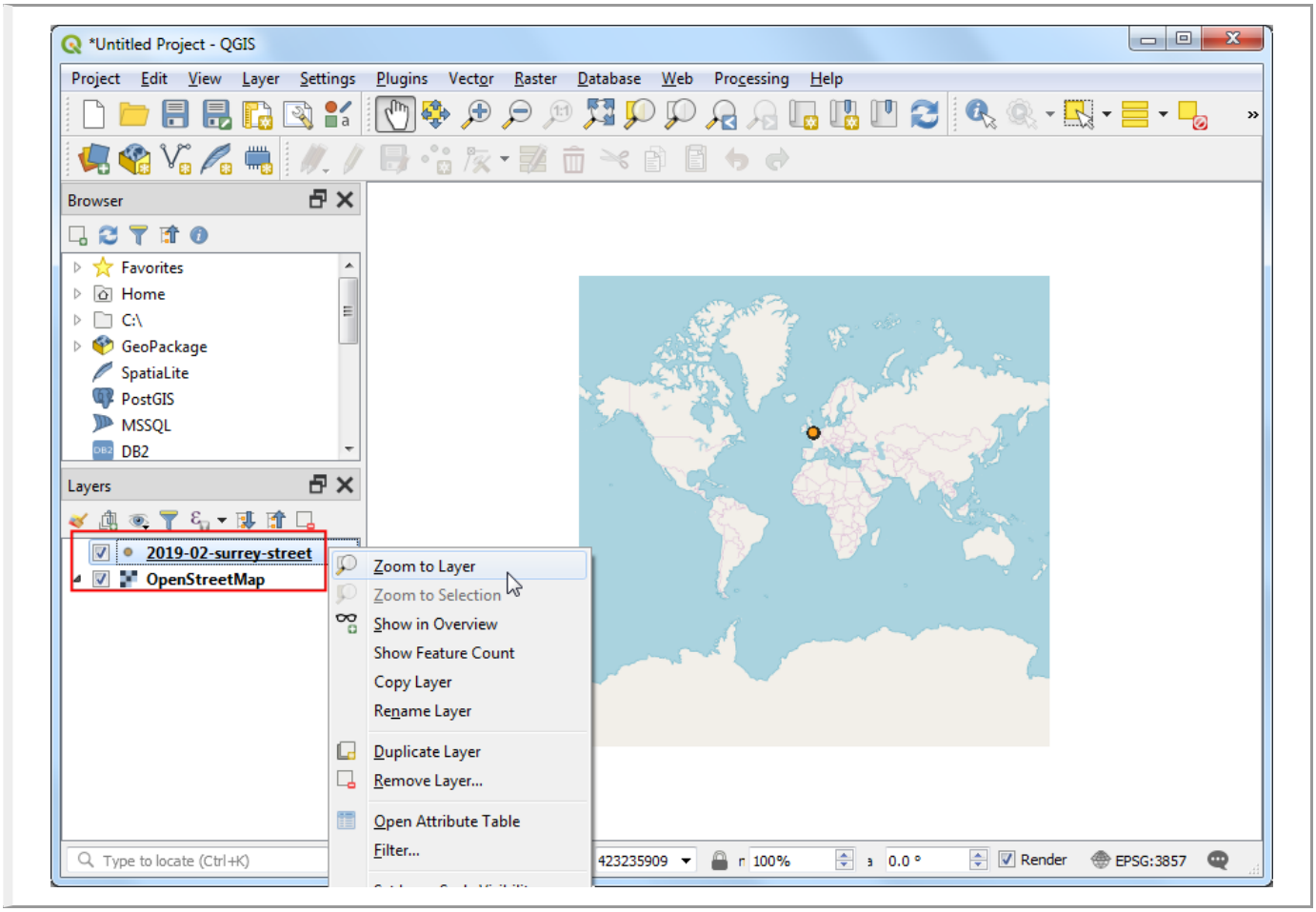
2. Select the Browser tab in the left-hand panel and find the OpenStreetMap layer under XYZ Tiles. Right-click and select Add Selected Layer(s) to Canvas to add this layer in QGIS.



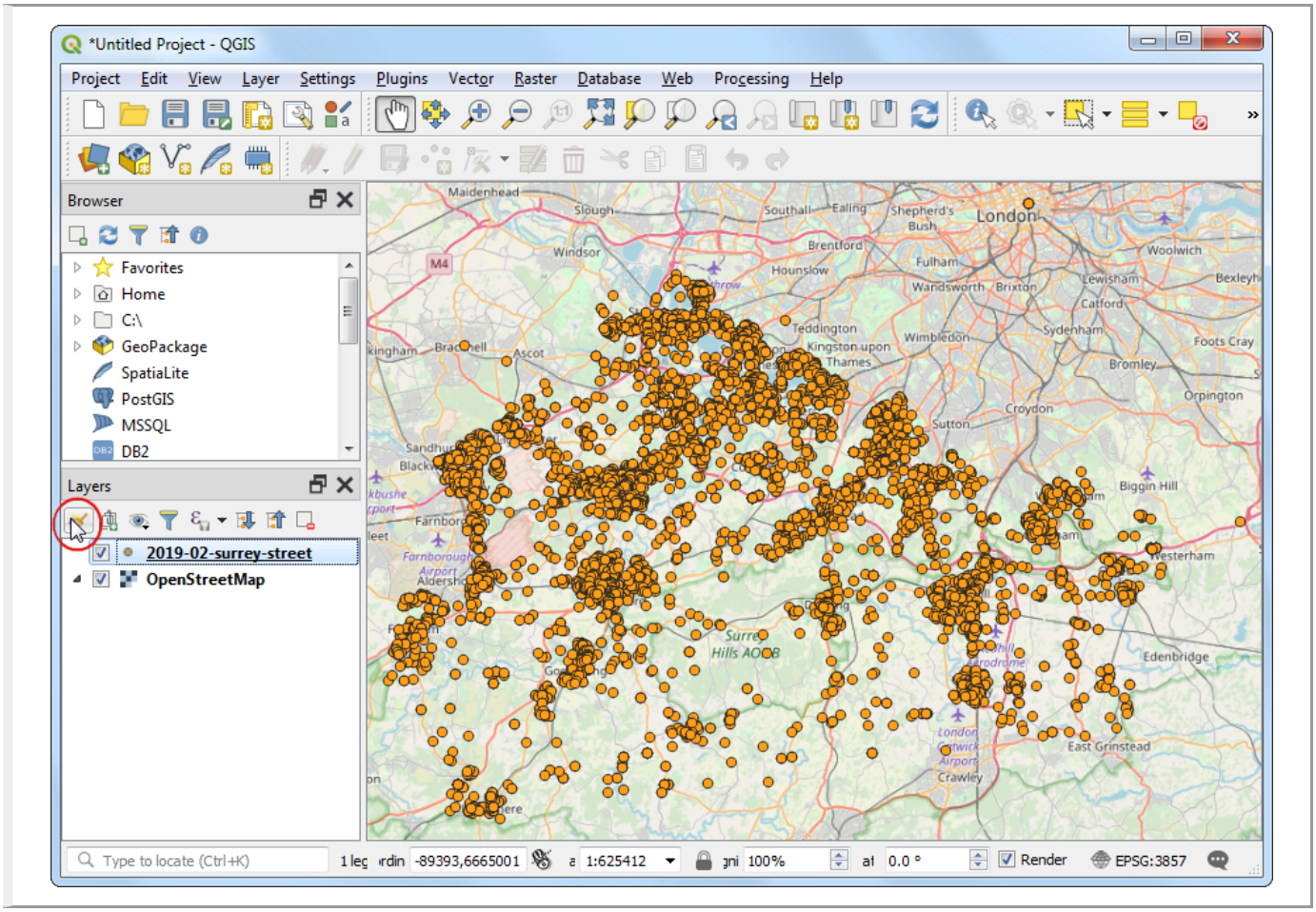
3. Switch to the Delimited Text tab. Here we will import the crime data which comes in a CSV format text file. Click the ... button next to File name and browse to the downloaded `2019-02-surrey-street.csv` file. The X field and Y field in the Geometry Definition section to be auto-populated with the `Longitude` and `Latitude` columns. The Geometry CRS should be left to default `EPSG:4326 - WGS 84` definition. Make sure the data looks correct in the Sample data panel and click Add, followed by Close.



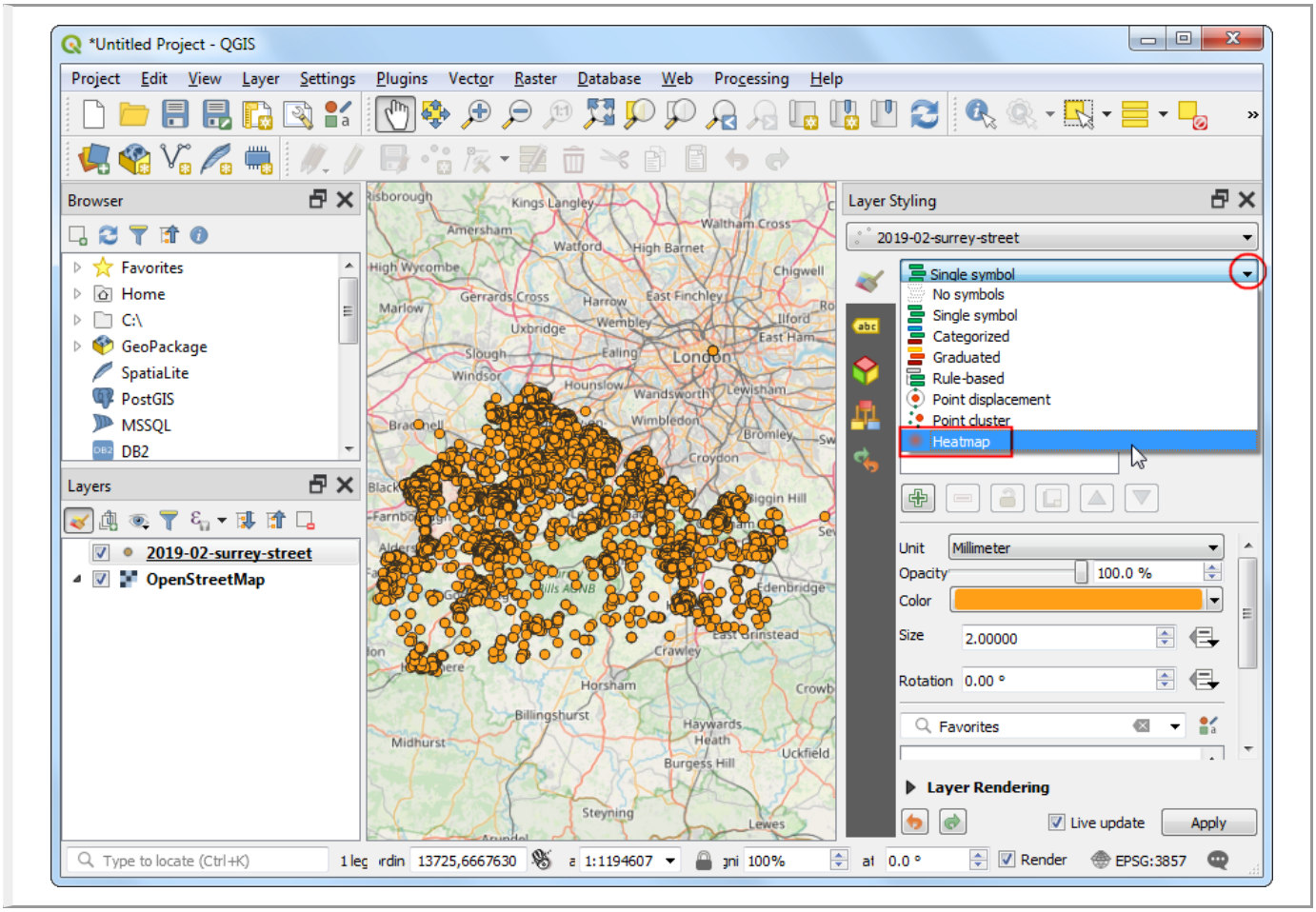
4. You will see 2 layers - `OpenStreetMap` and `2019-02-surrey-street` loaded in the QGIS Layers panel. Right-click the `2019-02-surrey-street` layer and select Zoom to Layer.



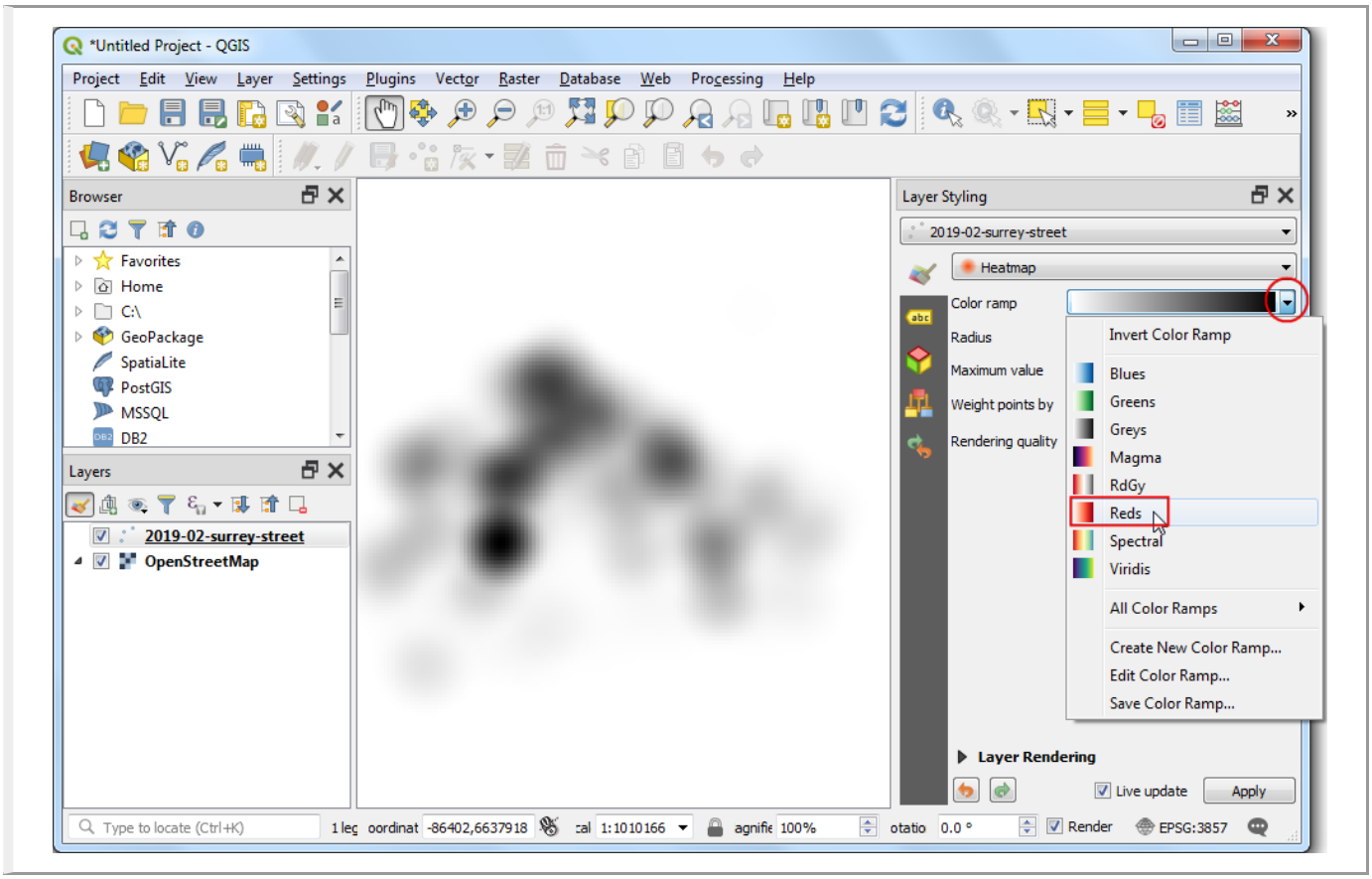
5. You will see the crime incident points layer overlaid on the OpenStreetMap basemap. Zoom and Pan to explore the data. The data is quite dense and it is hard to get an idea of where there is a high concentration of crime. This is where a heatmap visualization will come in handy. Select the `2019-02-surrey-street` layer and click the Open the Layer Styling panel button.



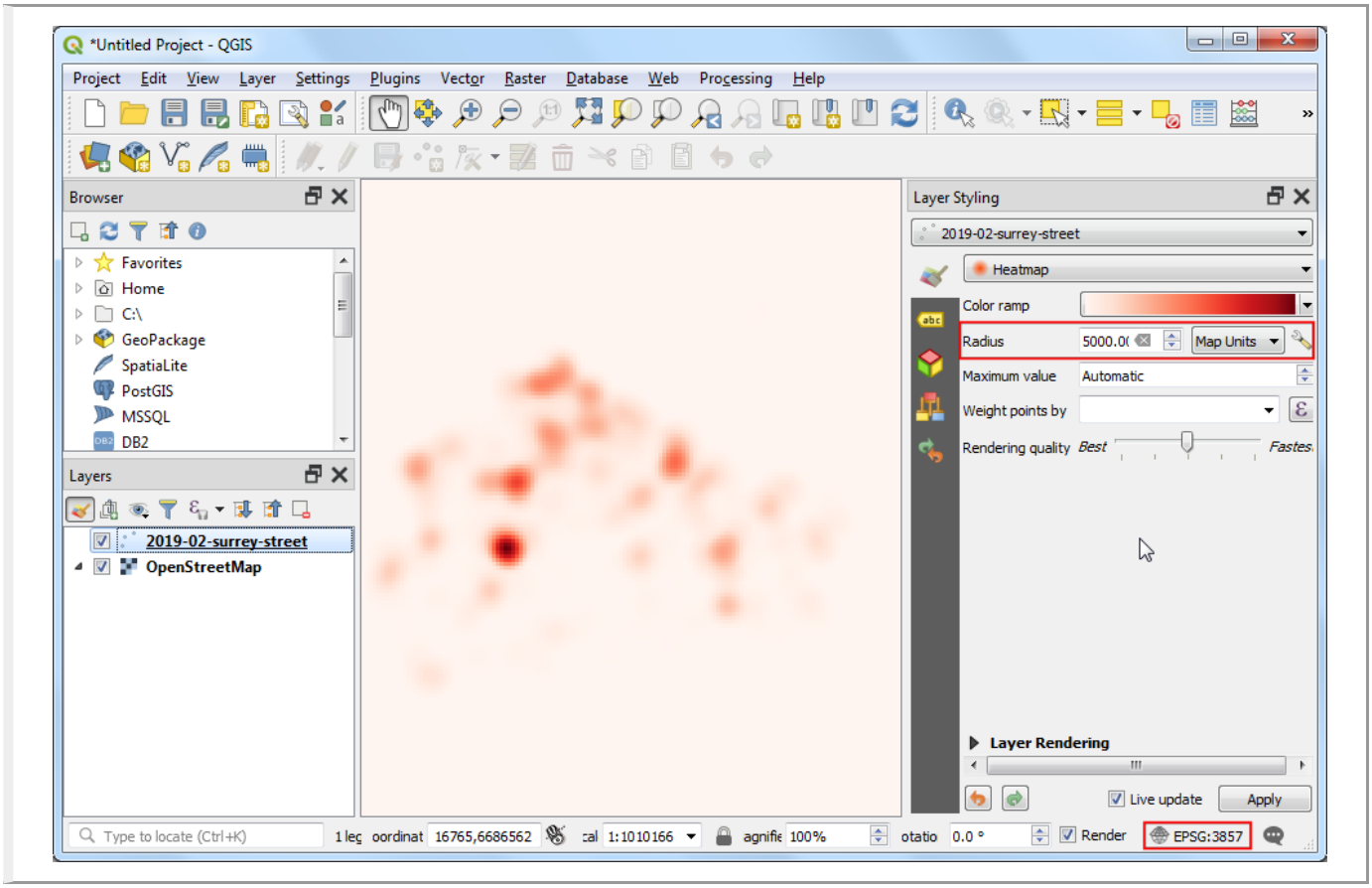
6. Select `Heatmap` as the renderer in the dropbox menu. The Layer Styling panel is interactive and you can see the effect of your changes reflected in the canvas immediately. The layer will now be displayed in the default grayscale color-ramp.



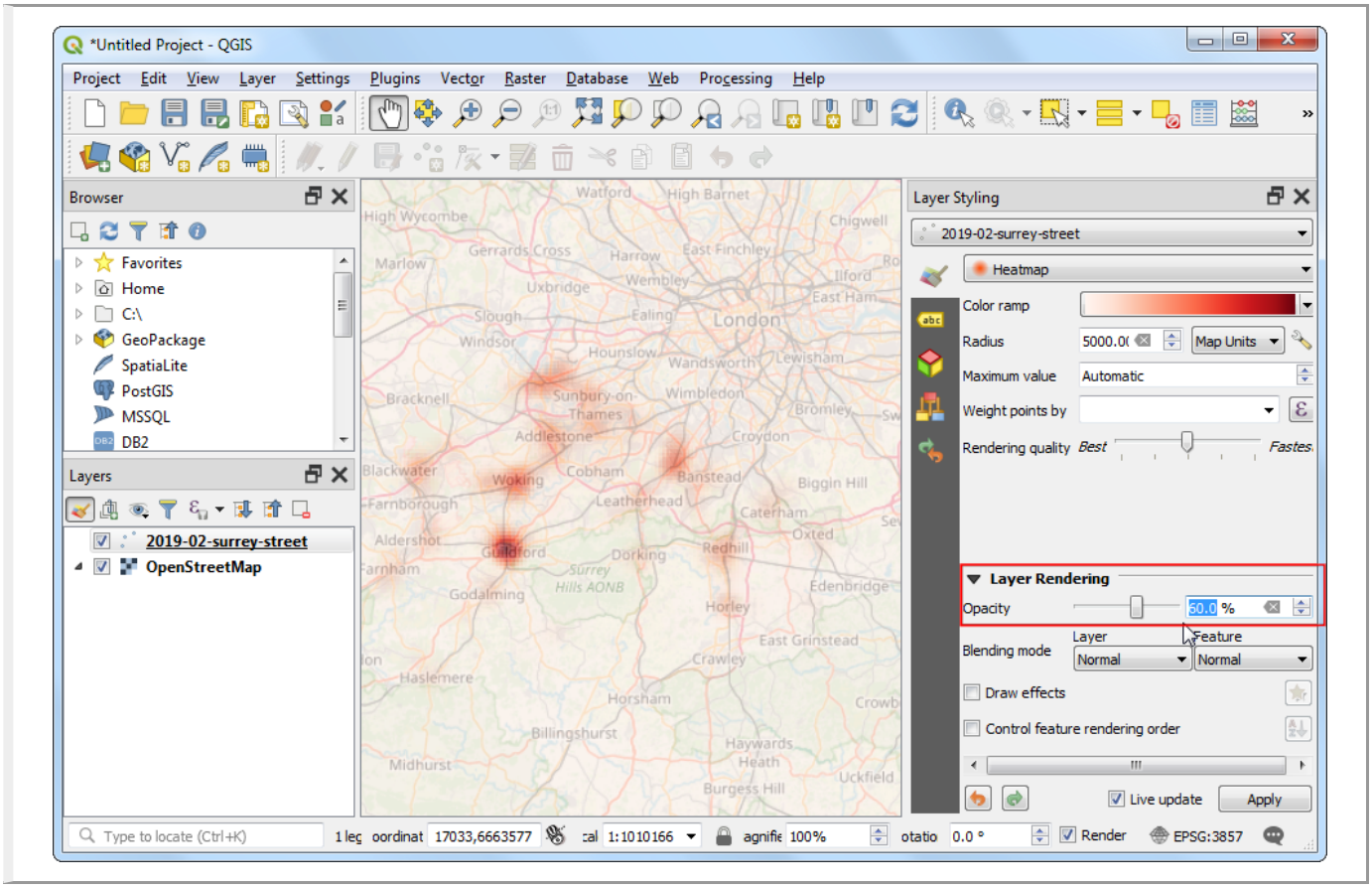
7. A heatmap is typically renderer using a yellow–to-red or white–to-red color ramp where higher concentration of points result in more **heat**. Click the Color ramp dropdown menu and select `Reds` color-ramp.



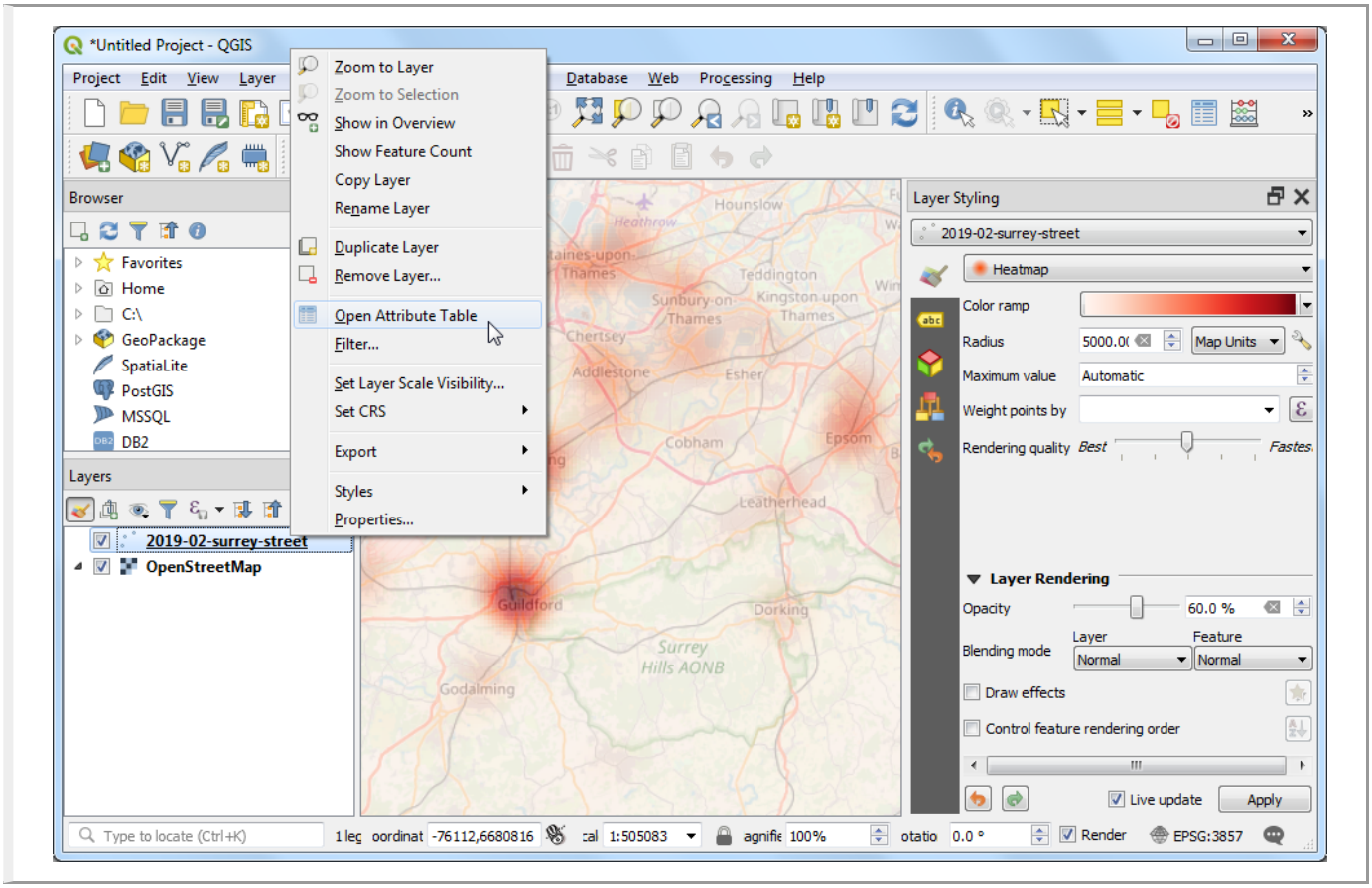
8. Next you need to choose a Radius. This parameter determines the circular neighborhood around each point where that point will have an influence. This value will largely depend on the type of your input data. For our data, let's assume a crime incident will have an influence upto 5 Kilometers from the location. Notice that the current project CRS is set to `EPSG: 3857` in the bottom-right corner. This CRS has a unit of meter, so we should specify `5000` meters as the radius. Another parameter that is hidden from this menu is the Kernel shape. This is a function that determines how the influence of a point should be spread out over the given radius. The Heatmap renderer uses the `Quartic` function for this calculation. There are other types of kernels such as `Triangular`, `Uniform`, `Triweight` and `Epanechnikov` that can be specified in when using a different heatmap creation method described later in this tutorial. See this post (https://www.geodose.com/2017/11/qgis-heatmap-using-kernel-density.html) for a good explanation and guidance for select the right radius and kernel shape.



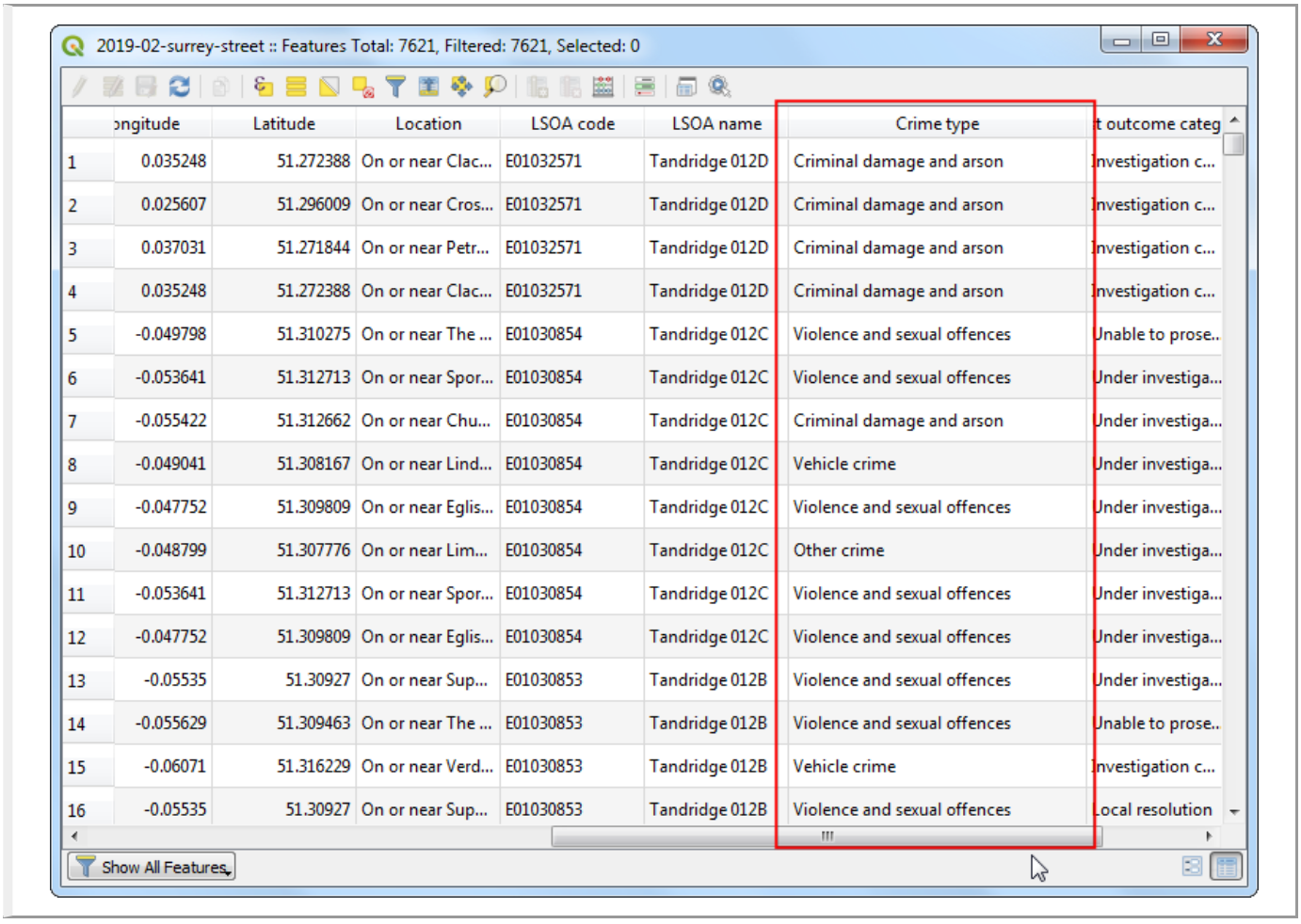
9. The heatmap visualization is ready. We can adjust the Opacity of the heatmap in the Layer Rendering section at the bottom. Set the opacity to `60 %` so you can see the basemap along with the heatmap.



10. For many types of analysis, just considering density of points is good enough. But sometimes, you may want to give different importance to each point. A more violent crime should have more influence on the output heatmap than a robbery. Similarly, sometimes a point may represent multiple observations at a single location which needs to be accounted for in the analysis. To do this, you are able to supply an optional numeric **weight** field which specifies a value for each point. Let's add a weight field and use it to improve the heatmap. Right-click the `2019-02-surrey-street` layer and select Open Attribute Table.

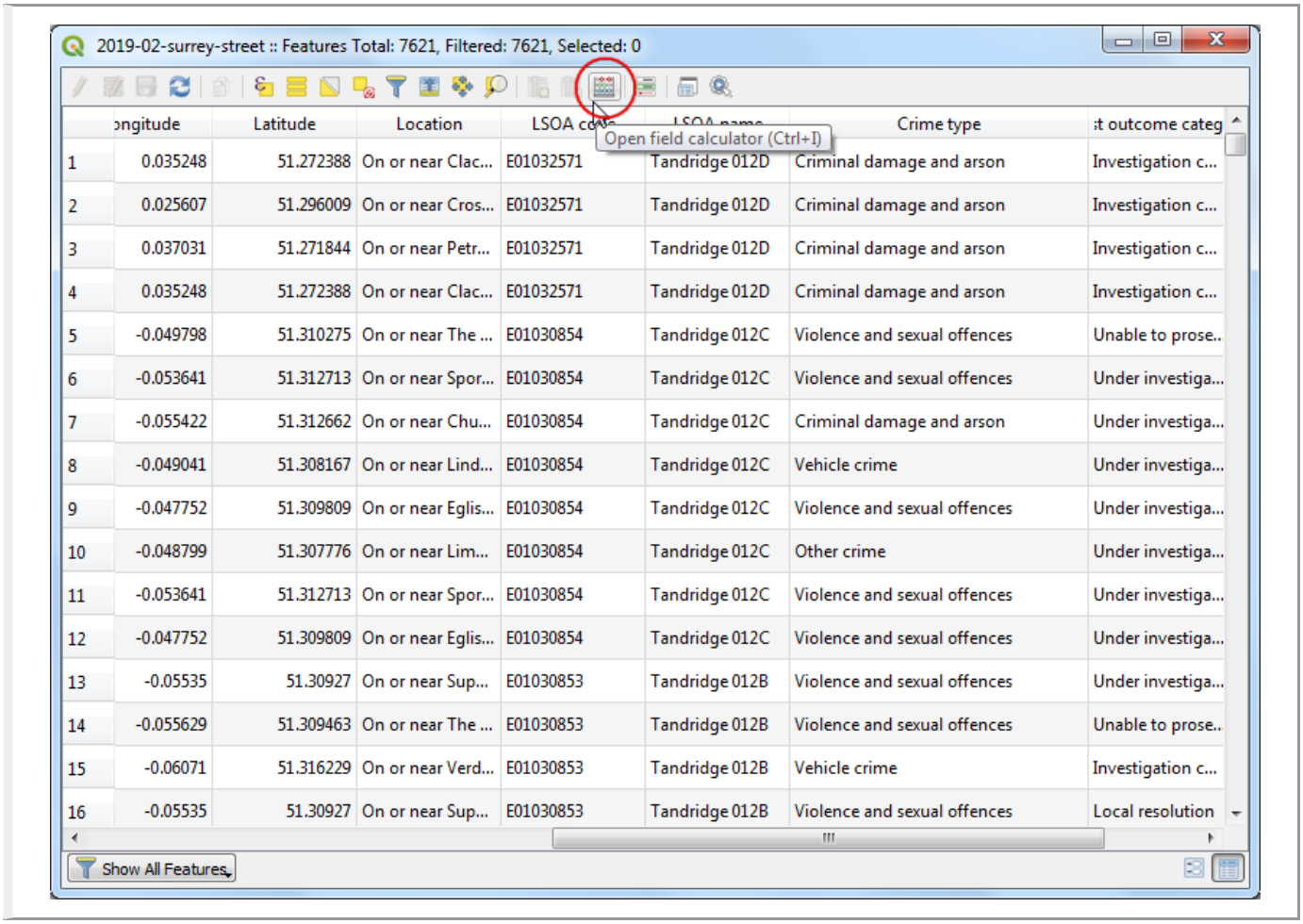


11. You will see a text field called `Crime type` in the input data that describes the type of crime. We can use these to categorize the different types of crimes and assign a higher weight to more violent crimes.



	ongitude	Latitude	Location	LSOA code	LSOA name	Crime type	t outcome categ
1	0.035248	51.272388	On or near Clac...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
2	0.025607	51.296009	On or near Cros...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
3	0.037031	51.271844	On or near Petr...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
4	0.035248	51.272388	On or near Clac...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
5	-0.049798	51.310275	On or near The ...	E01030854	Tandridge 012C	Violence and sexual offences	Unable to prose...
6	-0.053641	51.312713	On or near Spor...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
7	-0.055422	51.312662	On or near Chu...	E01030854	Tandridge 012C	Criminal damage and arson	Under investiga...
8	-0.049041	51.308167	On or near Lind...	E01030854	Tandridge 012C	Vehicle crime	Under investiga...
9	-0.047752	51.309809	On or near Eglis...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
10	-0.048799	51.307776	On or near Lim...	E01030854	Tandridge 012C	Other crime	Under investiga...
11	-0.053641	51.312713	On or near Spor...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
12	-0.047752	51.309809	On or near Eglis...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
13	-0.05535	51.30927	On or near Sup...	E01030853	Tandridge 012B	Violence and sexual offences	Under investiga...
14	-0.055629	51.309463	On or near The ...	E01030853	Tandridge 012B	Violence and sexual offences	Unable to prose...
15	-0.06071	51.316229	On or near Verd...	E01030853	Tandridge 012B	Vehicle crime	Investigation c...
16	-0.05535	51.30927	On or near Sup...	E01030853	Tandridge 012B	Violence and sexual offences	Local resolution

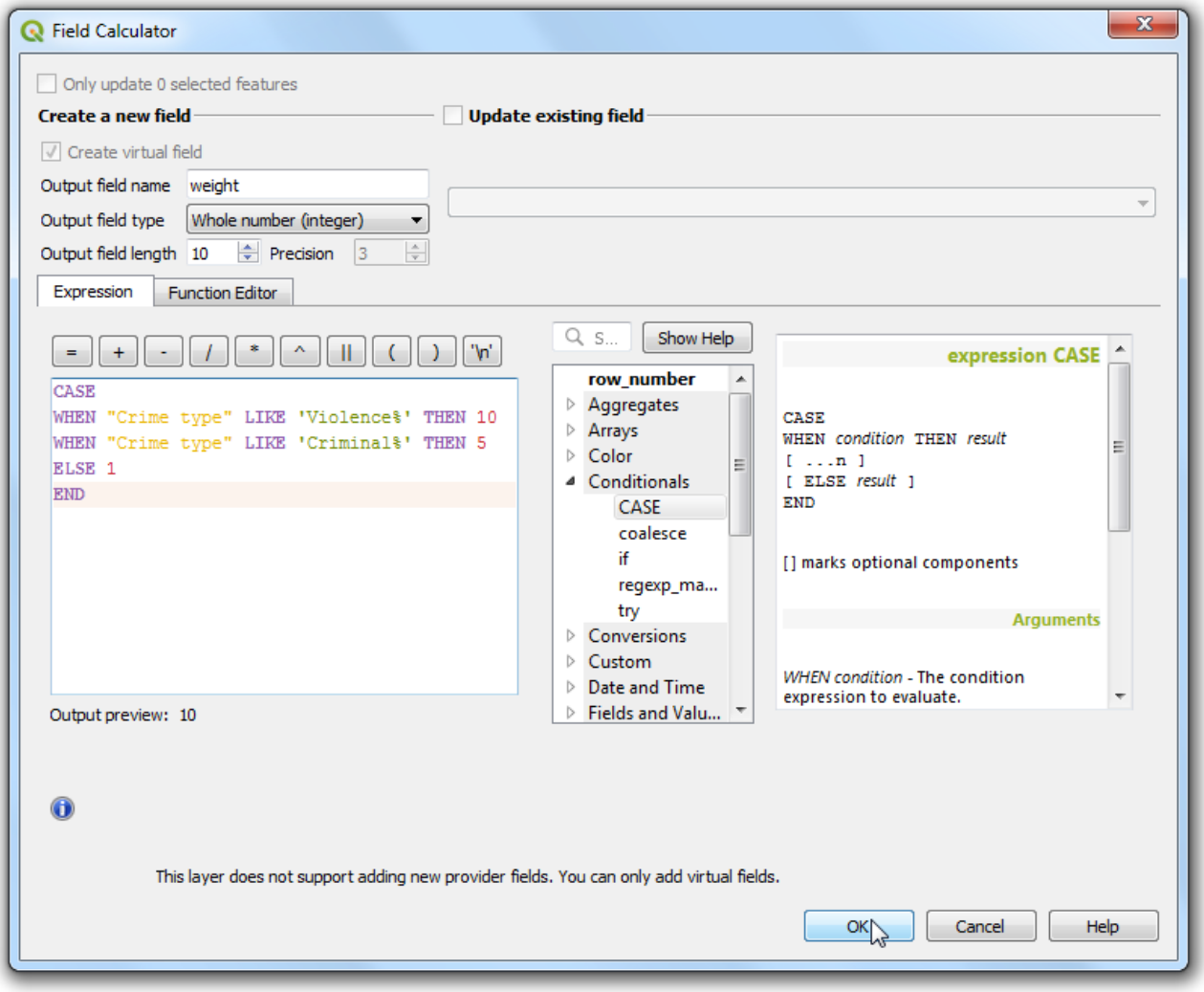
12. Click the Open field calculator.



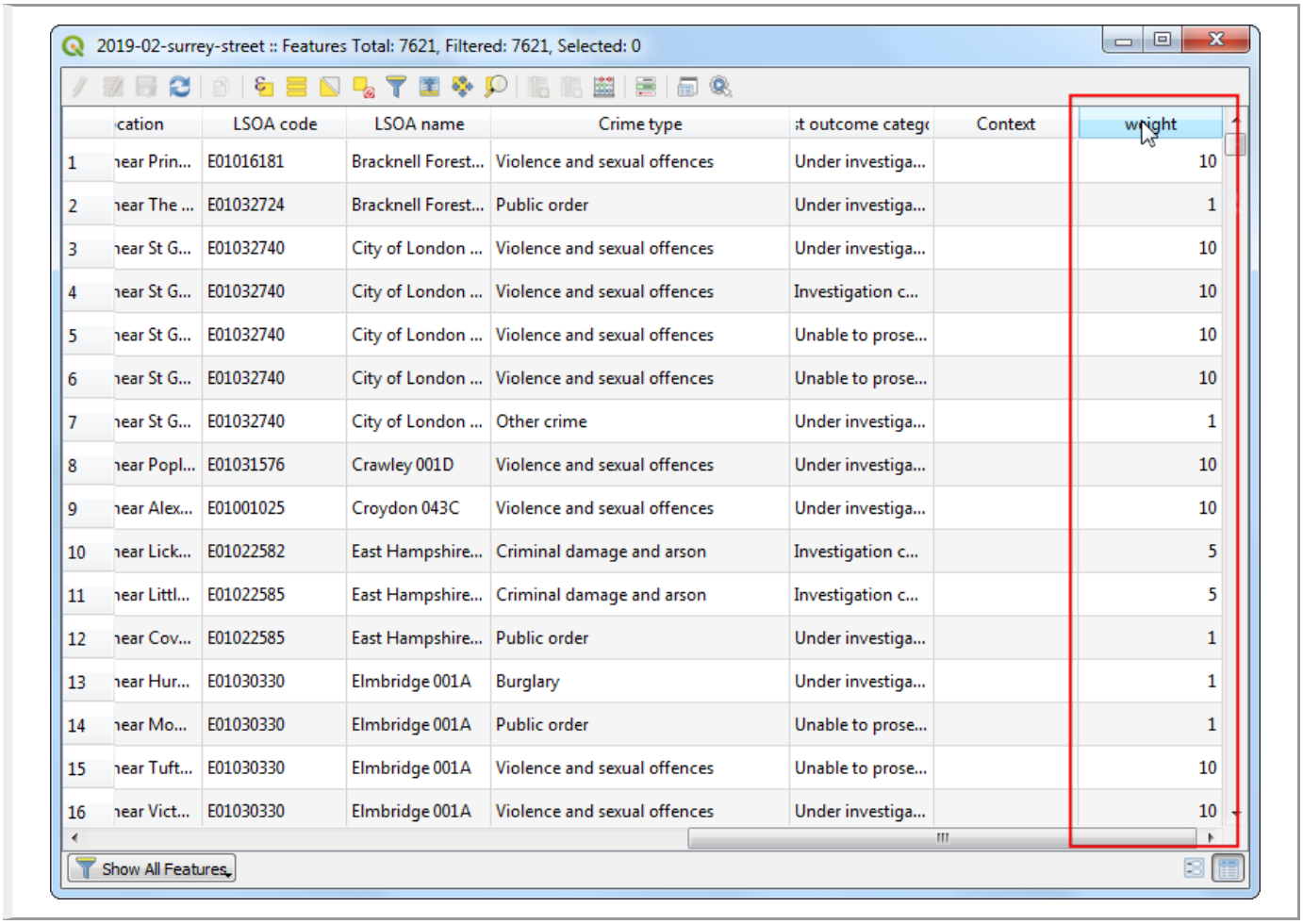
	ongitude	Latitude	Location	LSOA code	LSOA name	Crime type	t outcome categ
1	0.035248	51.272388	On or near Clac...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
2	0.025607	51.296009	On or near Cros...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
3	0.037031	51.271844	On or near Petr...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
4	0.035248	51.272388	On or near Clac...	E01032571	Tandridge 012D	Criminal damage and arson	Investigation c...
5	-0.049798	51.310275	On or near The ...	E01030854	Tandridge 012C	Violence and sexual offences	Unable to prose..
6	-0.053641	51.312713	On or near Spor...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
7	-0.055422	51.312662	On or near Chu...	E01030854	Tandridge 012C	Criminal damage and arson	Under investiga...
8	-0.049041	51.308167	On or near Lind...	E01030854	Tandridge 012C	Vehicle crime	Under investiga...
9	-0.047752	51.309809	On or near Eglis...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
10	-0.048799	51.307776	On or near Lim...	E01030854	Tandridge 012C	Other crime	Under investiga...
11	-0.053641	51.312713	On or near Spor...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
12	-0.047752	51.309809	On or near Eglis...	E01030854	Tandridge 012C	Violence and sexual offences	Under investiga...
13	-0.05535	51.30927	On or near Sup...	E01030853	Tandridge 012B	Violence and sexual offences	Under investiga...
14	-0.055629	51.309463	On or near The ...	E01030853	Tandridge 012B	Violence and sexual offences	Unable to prose..
15	-0.06071	51.316229	On or near Verd...	E01030853	Tandridge 012B	Vehicle crime	Investigation c...
16	-0.05535	51.30927	On or near Sup...	E01030853	Tandridge 012B	Violence and sexual offences	Local resolution

13. We will now input a formula that uses the `Crime type` and determines the weight value. QGIS has a handy way to add such computed fields using *Virtual Fields*. The virtual field is saved in the QGIS project and doesn't modify the source data. It is also dynamically computed and can be used anywhere in QGIS just like any other attribute value. Enter `weight` as the Output field name and set the Output field type to `Whole number (integer)`. Enter the following expression in the Expression editor. Here we are using **CASE** statement to assign different values based on different conditions. Click OK.

```
CASE
WHEN "Crime type" LIKE 'Violence%' THEN 10
WHEN "Crime type" LIKE 'Criminal%' THEN 5
ELSE 1
END
```

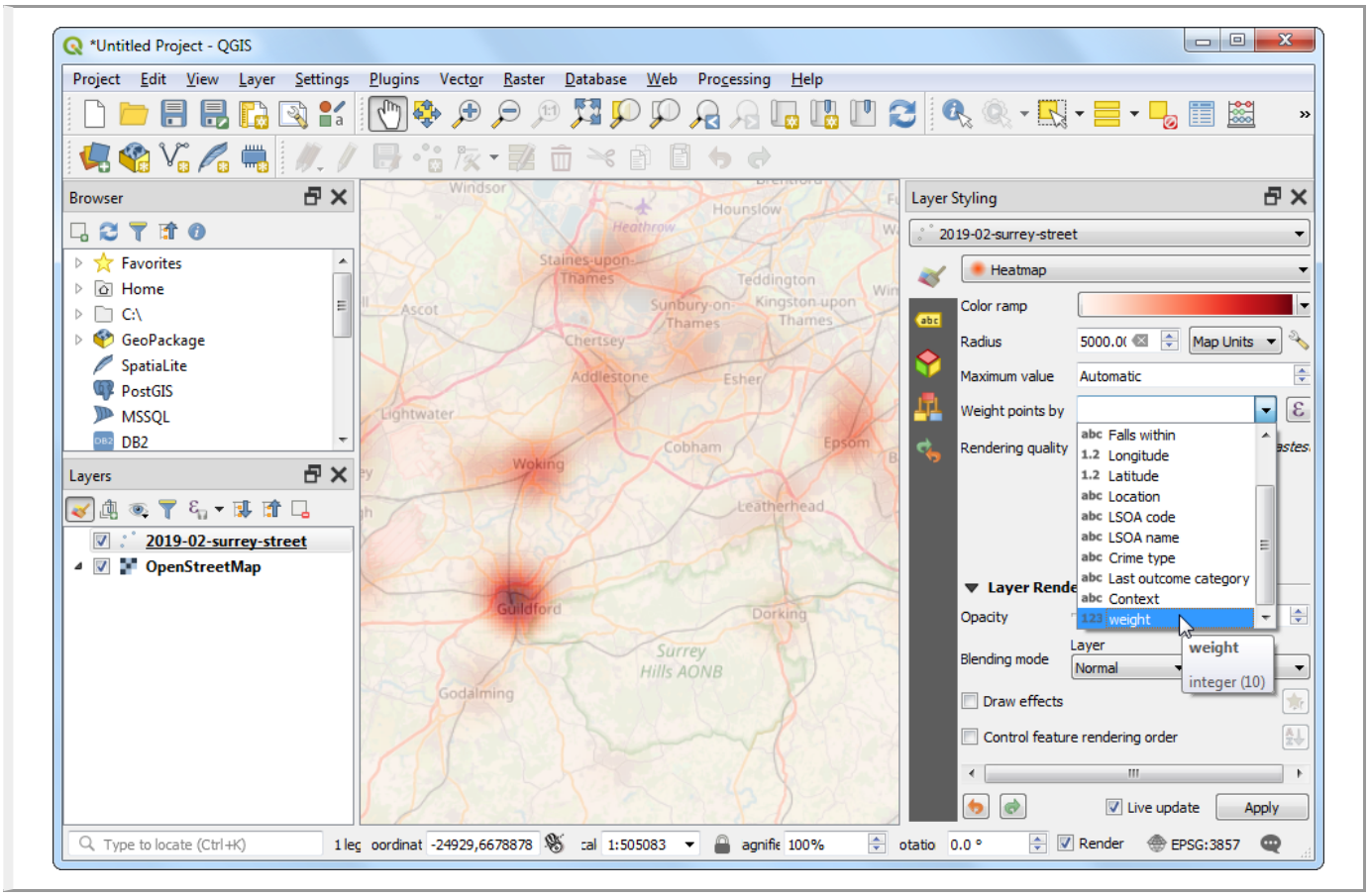


14. A new attribute will be added for each feature with the appropriate weight value.

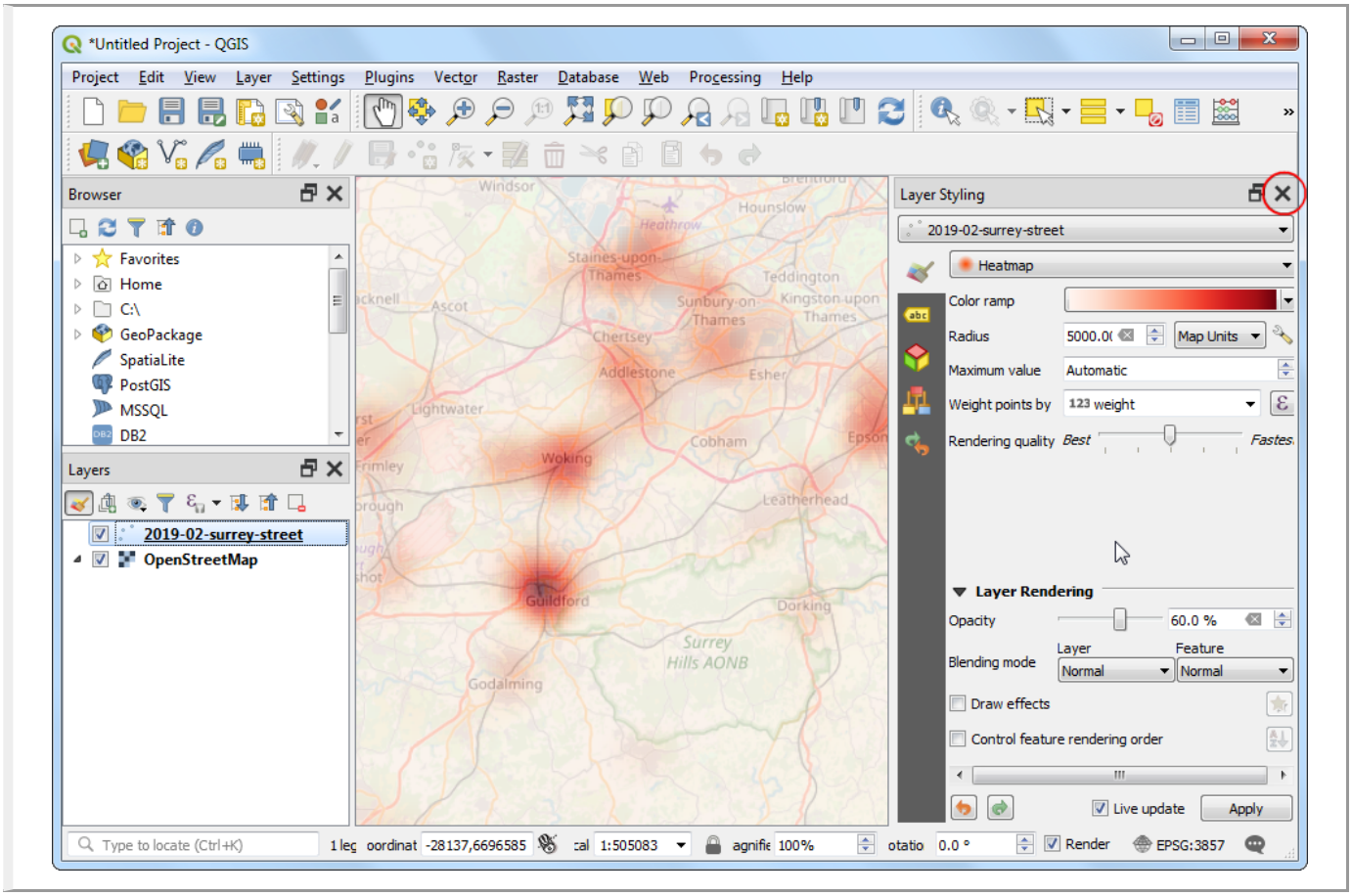


	cation	LSOA code	LSOA name	Crime type	t outcome categ	Context	weight
1	near Prin...	E01016181	Bracknell Forest...	Violence and sexual offences	Under investiga...		10
2	near The ...	E01032724	Bracknell Forest...	Public order	Under investiga...		1
3	near St G...	E01032740	City of London ...	Violence and sexual offences	Under investiga...		10
4	near St G...	E01032740	City of London ...	Violence and sexual offences	Investigation c...		10
5	near St G...	E01032740	City of London ...	Violence and sexual offences	Unable to prose...		10
6	near St G...	E01032740	City of London ...	Violence and sexual offences	Unable to prose...		10
7	near St G...	E01032740	City of London ...	Other crime	Under investiga...		1
8	near Popl...	E01031576	Crawley 001D	Violence and sexual offences	Under investiga...		10
9	near Alex...	E01001025	Croydon 043C	Violence and sexual offences	Under investiga...		10
10	near Lick...	E01022582	East Hampshire...	Criminal damage and arson	Investigation c...		5
11	near Littl...	E01022585	East Hampshire...	Criminal damage and arson	Investigation c...		5
12	near Cov...	E01022585	East Hampshire...	Public order	Under investiga...		1
13	near Hur...	E01030330	Elmbridge 001A	Burglary	Under investiga...		1
14	near Mo...	E01030330	Elmbridge 001A	Public order	Unable to prose...		1
15	near Tuft...	E01030330	Elmbridge 001A	Violence and sexual offences	Unable to prose...		10
16	near Vict...	E01030330	Elmbridge 001A	Violence and sexual offences	Under investiga...		10

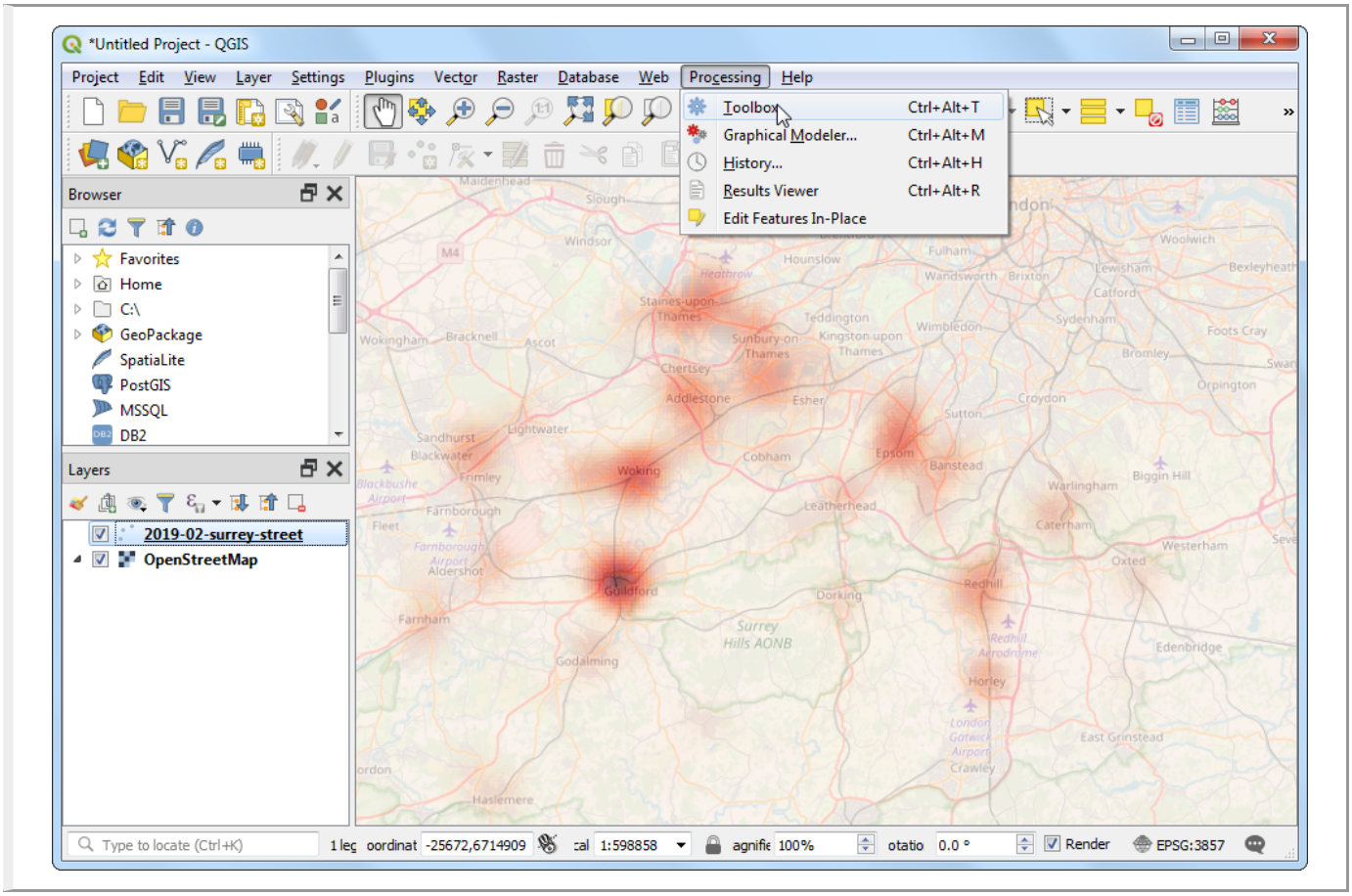
15. Back in the Layer Styling panel, click the drop-down menu for Weight points by and select the newly added `weight` field.



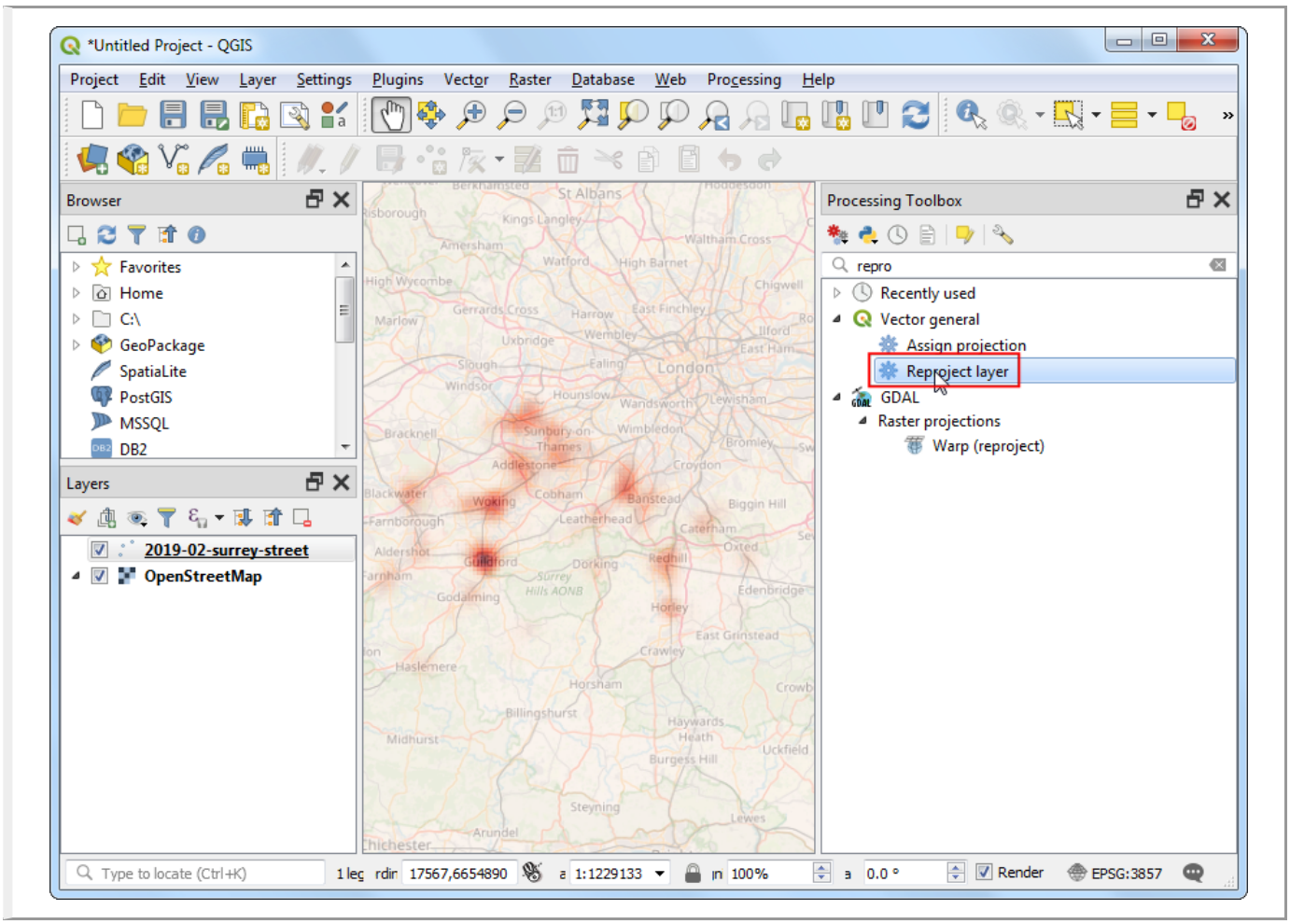
16. You will see the heatmap rendering change to account for the weight parameter. Close the Layer Styling panel.



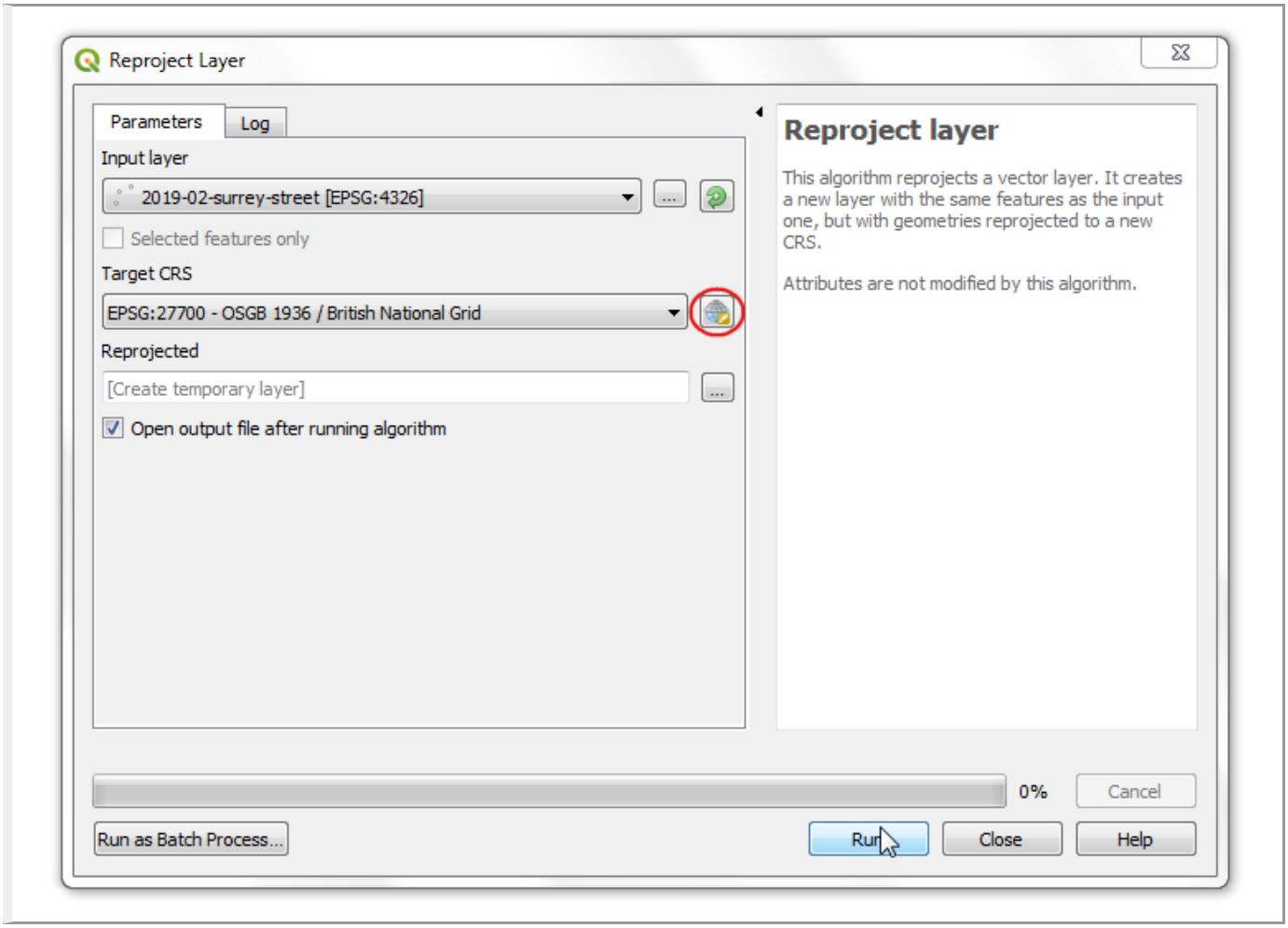
17. If you need the heatmap visualization to be saved as a permanent raster layer or want to customize the heatmap with advanced options such as different kernels or dynamic radius, you can use the **Heatmap (Kernel Density Estimation)** from the Processing Toolbox. We will now use this algorithm. Go to Processing ‣ Toolbox.



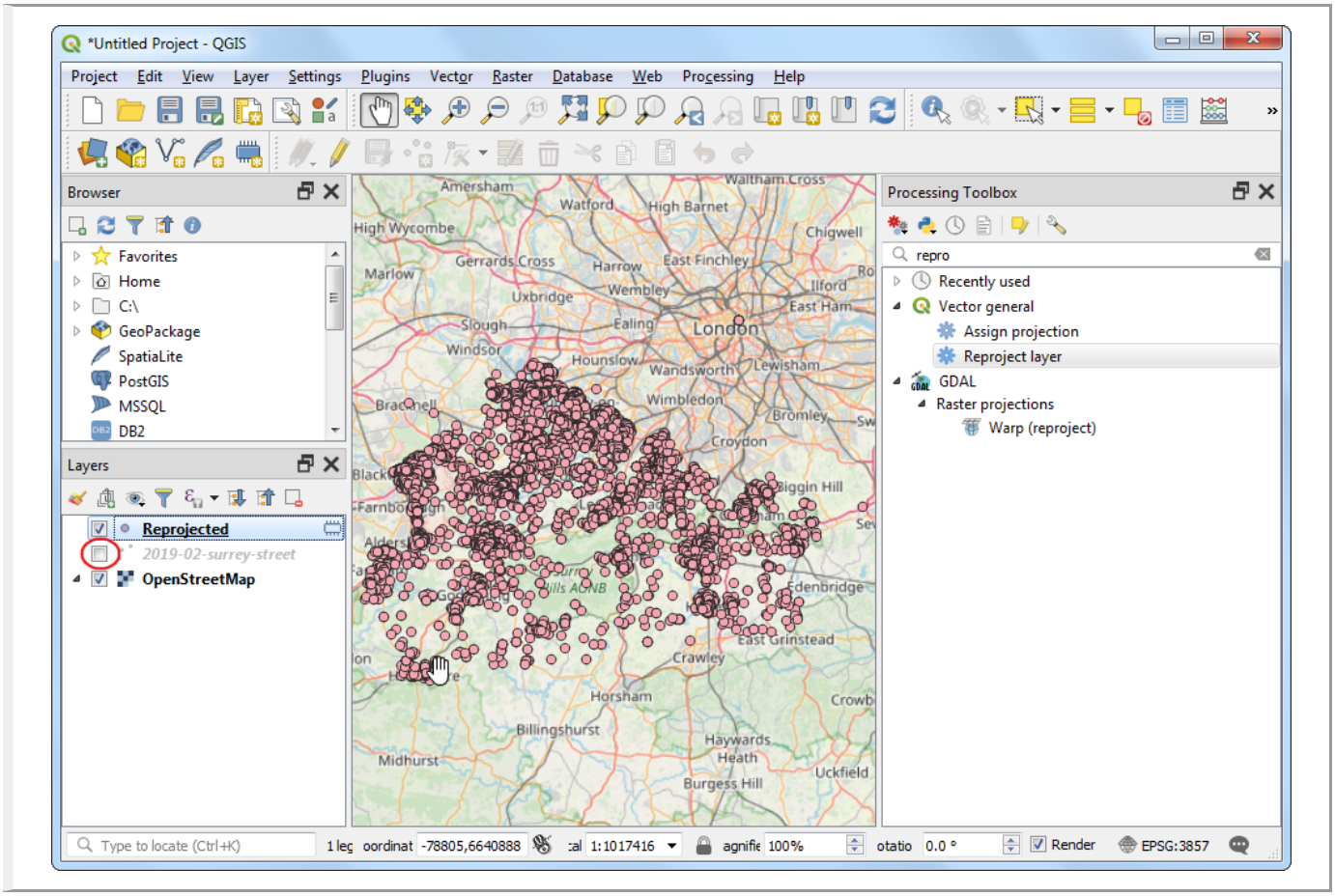
18. Before we can create the heatmap, we need to re-project the source data to a projected CRS. As distance plays an important role in computation of heatmap, it is not correct to use a geographic CRS. Search and find the Vector general ▸ Reproject layer algorithm.



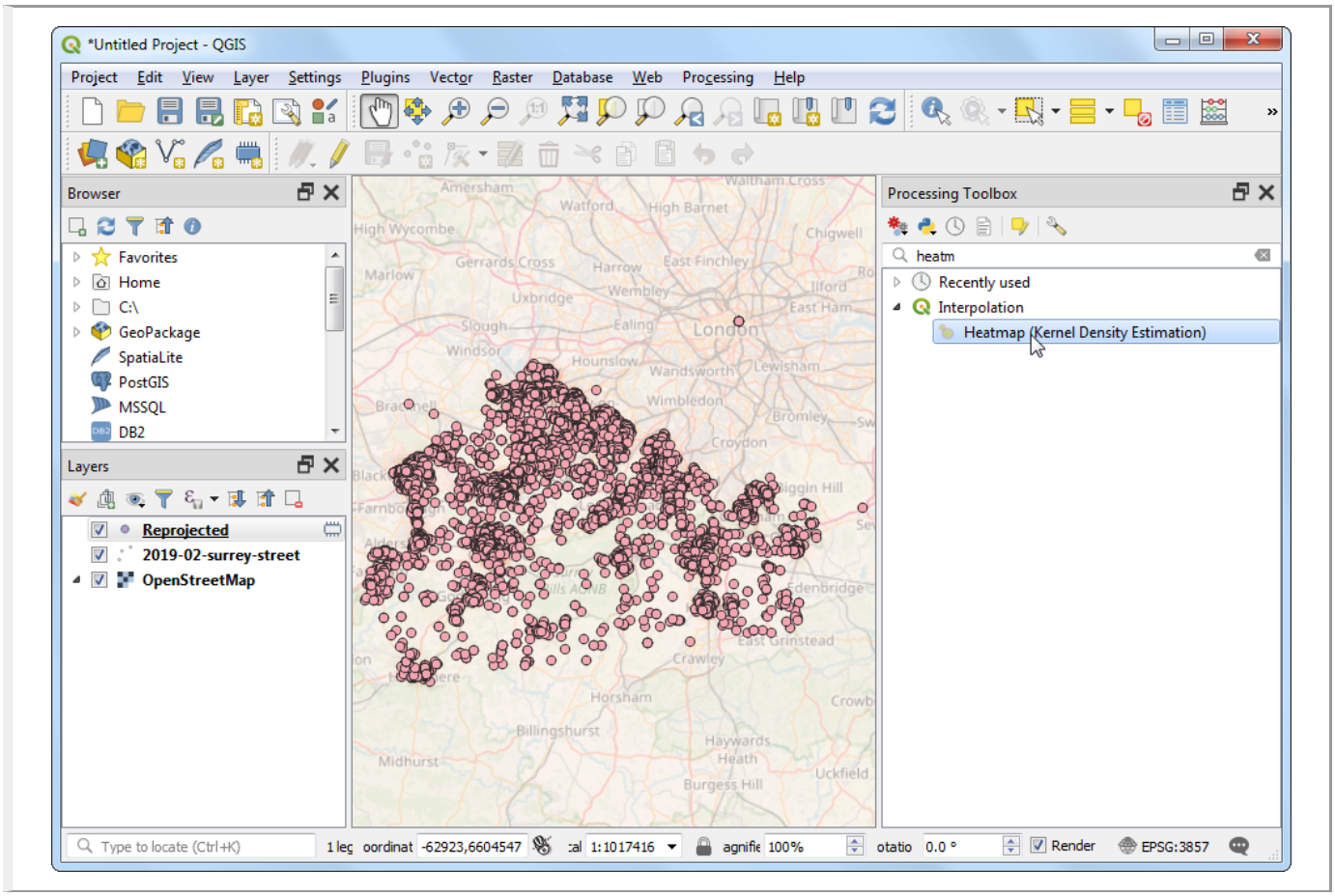
19. In the Reproject layer dialog, click the Select CRS button for Target CRS. Search for and select the `EPSG:27700 OSGB 1936 / British National Grid` CRS. This projected CRS is a good choice for data in the UK. Click Run.



20. A new layer named `Reprojected` will be added to the Layers panel. Un-check the box next to the old `2019-02-surrey-street` layer to hide it.

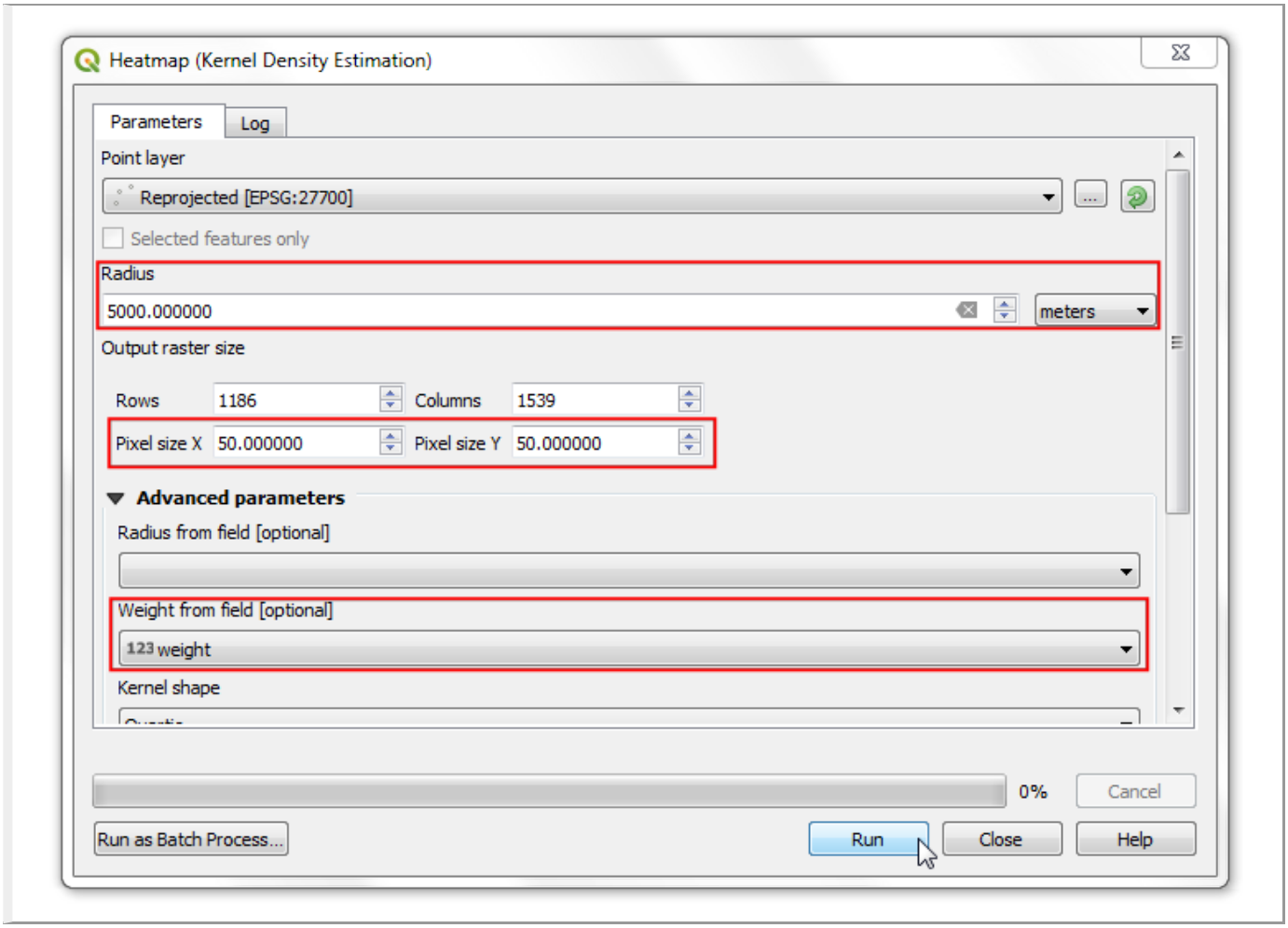


21. Search and find the Interpolation ▸ Heatmap (Kernel Density Estimation) algorithm.



22. In the Heatmap (Kernel Density Estimation) dialog, we will use the same paramters as earlier. Select Radius as `5000` meters and Weight from field as `weight`. Set the Pixel size X and Pixel size Y to `50`

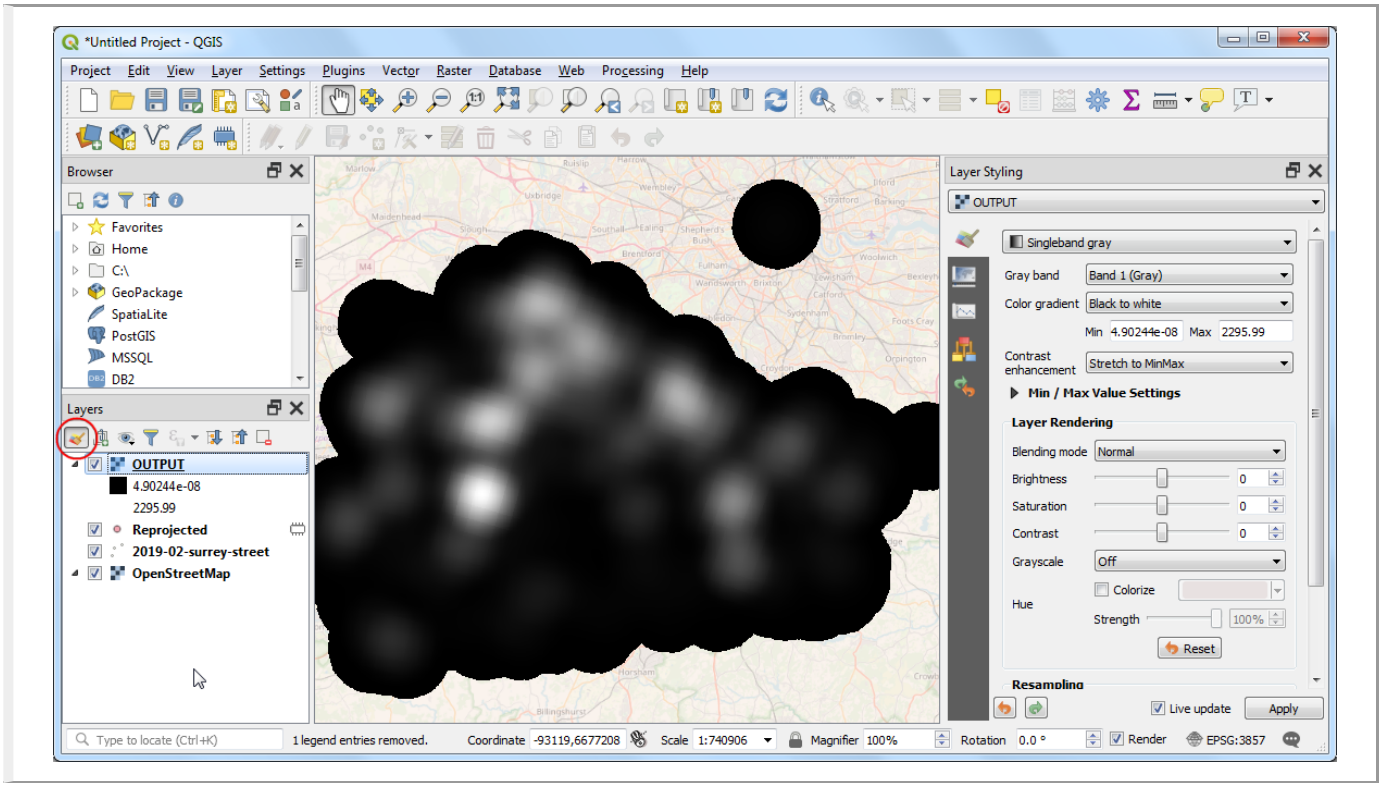
meters. Let the Kernel shape to the default value of `Quartic`. Click Run.



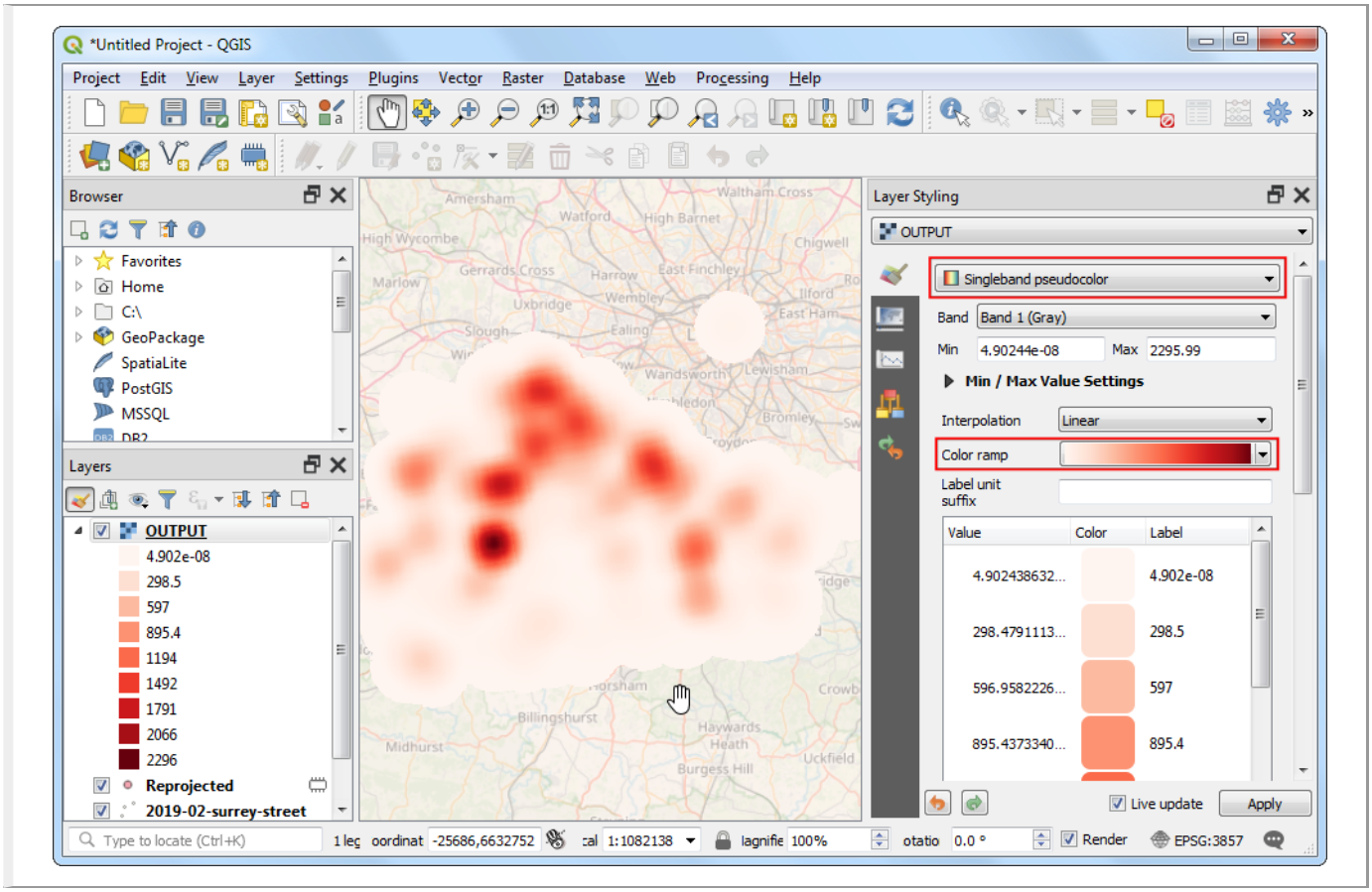
Note

The Radius from field parameter allows you to specify a dynamic search radius for each point. This can be used along with Weight from field to have fine grainer control on how each point's influence is spread.

23. Once the processing finishes, a new raster layer named `OUTPUT` will be loaded. The default visualization is ugly since it uses the `Singleband gray` renderer. Click the Open the Layer Styling panel button.



24. Change the render to `Singleband Pseudocolor` and select the `Reds` color ramp. The layer now looks like the heatmap visualization that we had created earlier.



Note

Notice that `OUTPUT` layer in the Layers panel has a legend but the `2019-02-surrey-street` layer does not. A common problem with using a heatmap layer created with the Heatmap renderer is the lack of a legend. Say you want use the heatmap in the Print Layout and add a legend. A raster heatmap created with the Heatmap processing algorithm method makes this possible.

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Animating Time Series Data (QGIS3)

Time is an important component of many spatial datasets. Along with location information, time providers another dimension for analysis and visualization of data. If you are working with dataset that contains timestamps or have observations recorded at multiple time-steps, you can easily visualize it using the **TimeManager** plugin in QGIS. TimeManager allows you to view and export 'slices' of data between certain time intervals that can be combined into animations.

Overview of the task

We will take a point layer of maritime piracy incidents, create a heatmap visualization and create an animation of how the piracy hot-spots have changed over past 2 decades.

Other skills you will learn

- Using the Heatmap renderer for quick visualization of dense point data
- Creating and using custom map projections

Get the data

National Geospatial-Intelligence Agency's Maritime Safety Information portal (https://msi.nga.mil/NGAPortal/MSI.portal) provides a shapefile of all incidencts of maritine piracy in the form on Anti-shipping Activity Messages (https://msi.nga.mil/Piracy). Download the Arc Shape file (https://msi.nga.mil/api/publications/download?key=16920958/SFH00000/ASAM_shp.zip&type=download) version of the database.

Natural Earth (http://naturalearthdata.com) has several global vector layers. Download the 10m Physical Vectors - Land (https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/physical/ne_10m_land.zip) containing Land polygons.

For convenience, you may directly download a copy of the above layers from below:

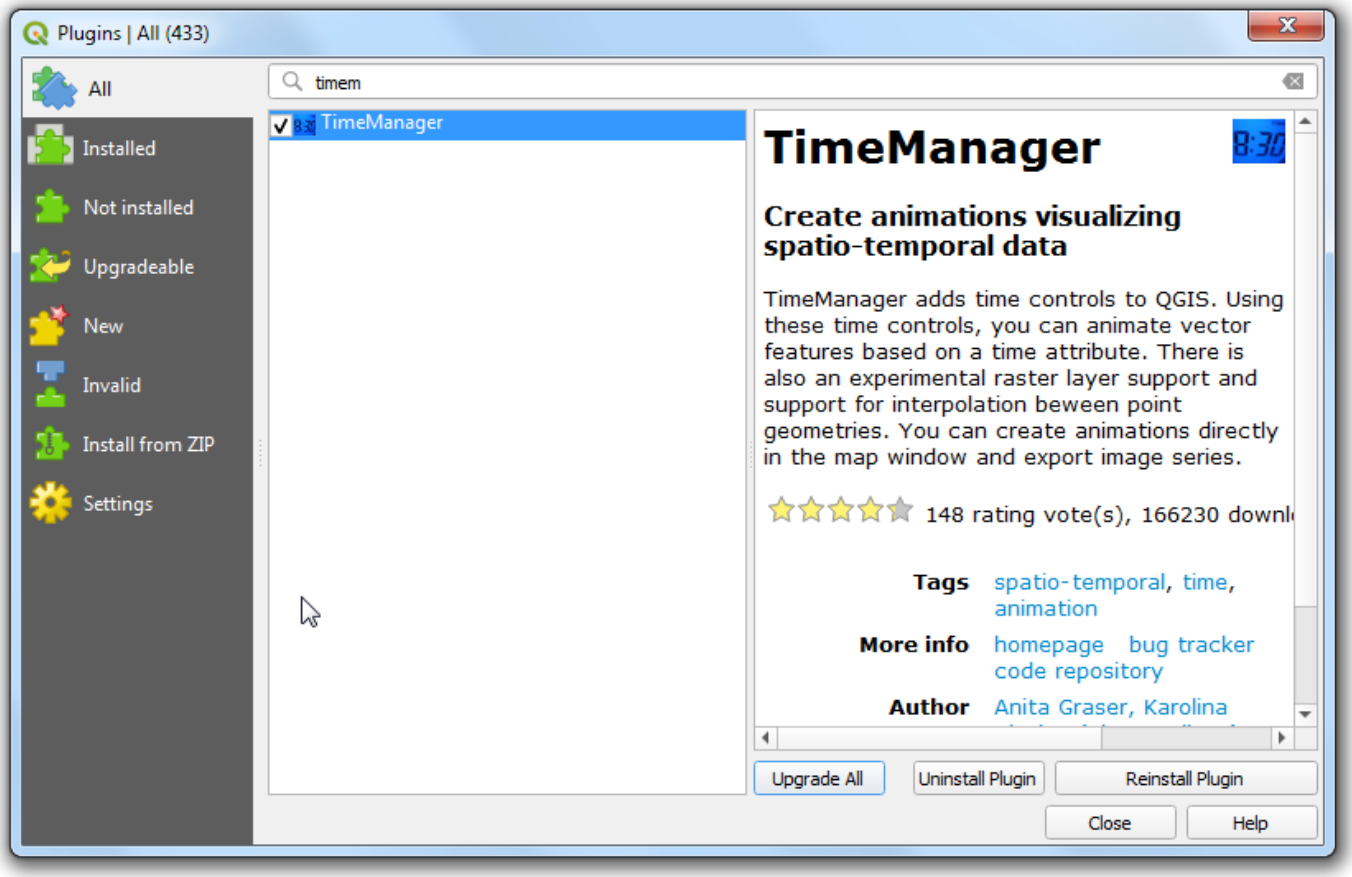
ASAM_shp.zip (http://www.qgistutorials.com/downloads/ASAM_shp.zip)

ne_10m_land.zip (http://www.qgistutorials.com/downloads/ne_10m_land.zip)

Data Source: [NGA_MSI] (../credits.html#nga-msi) [NATURALEARTH] (../credits.html#naturalearth)

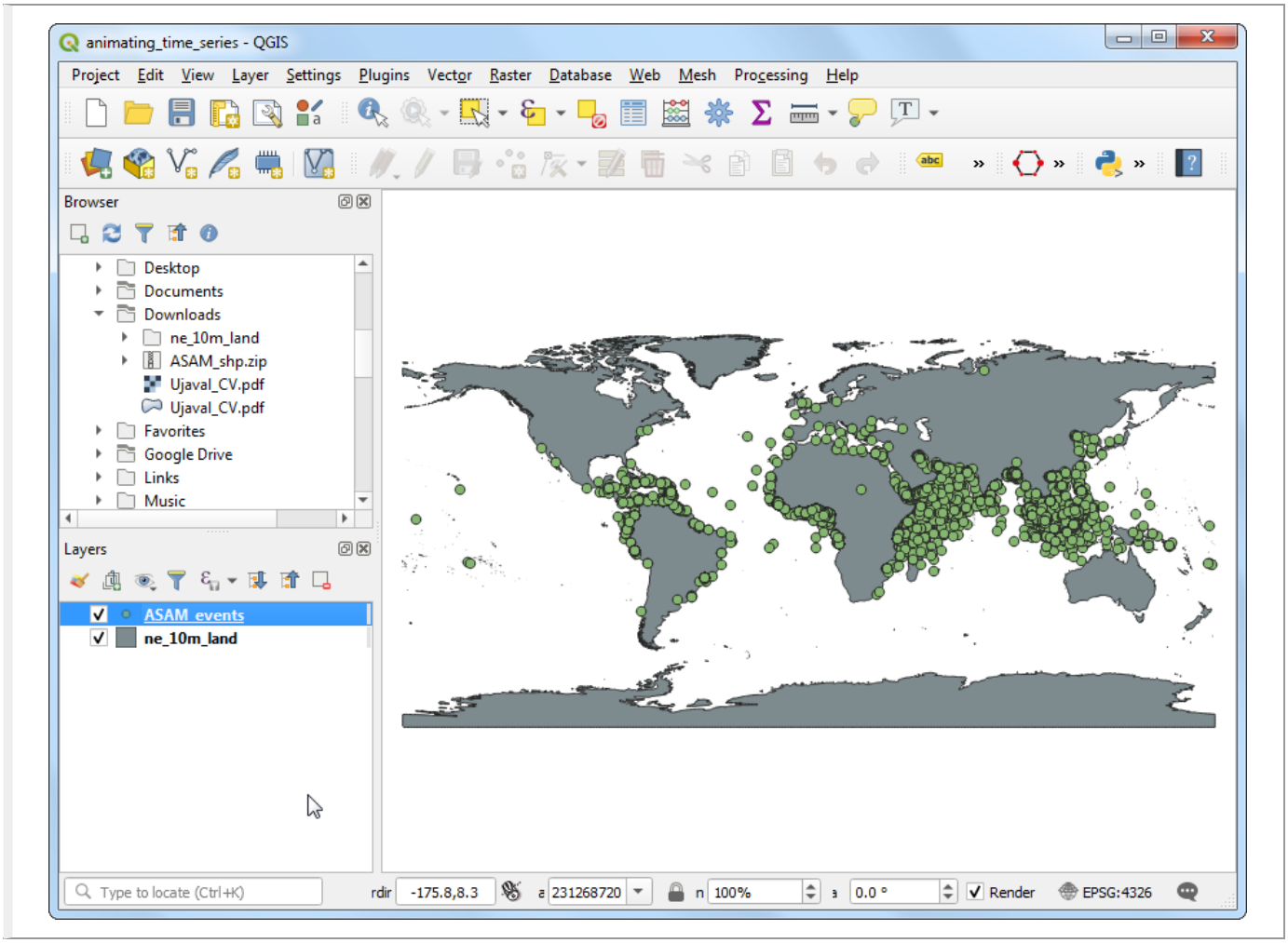
Setup

Go to Plugins ▸ Manage and Install Plugins.... Search for and install the **TimeManager** plugin.

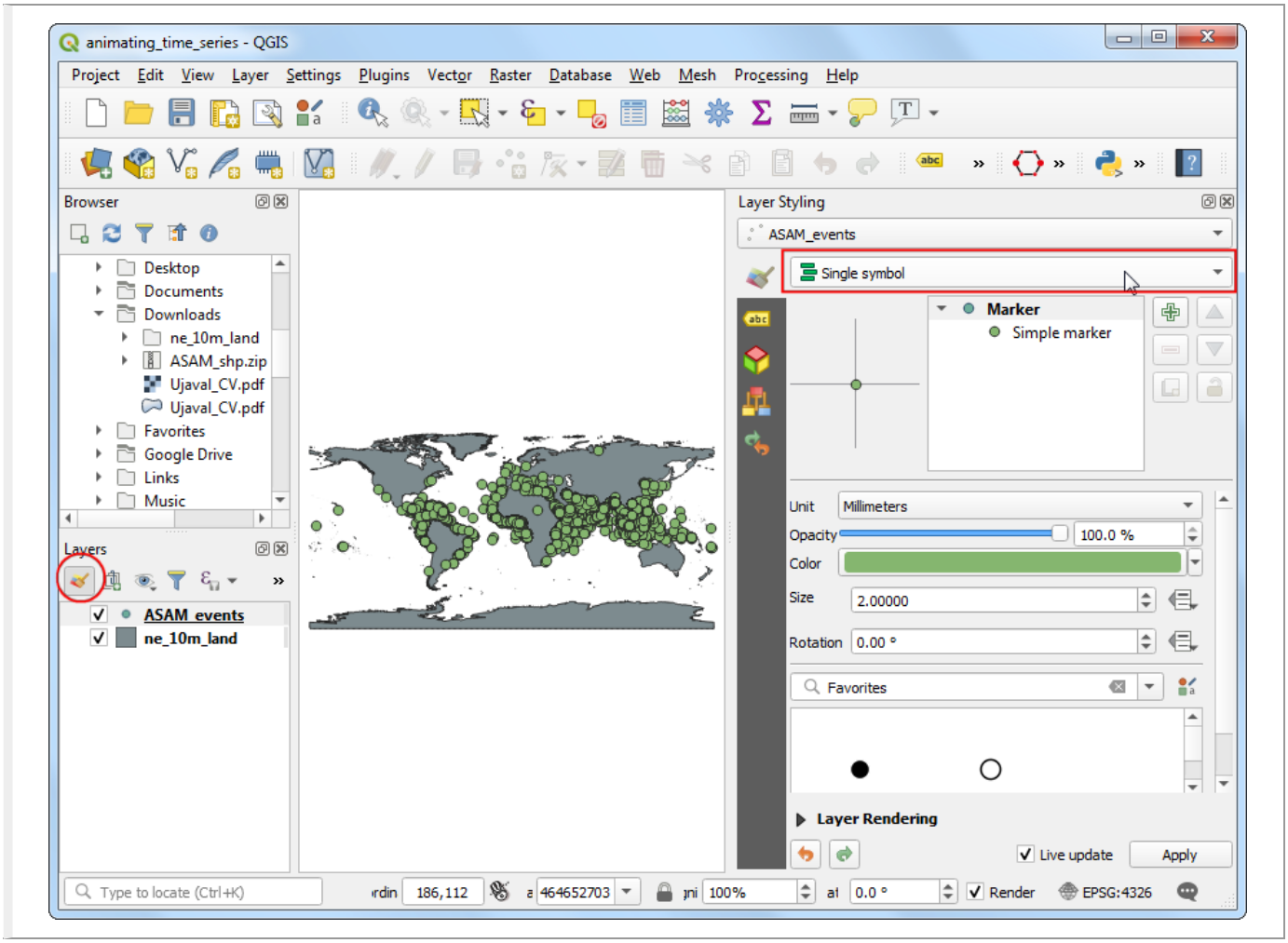


Procedure

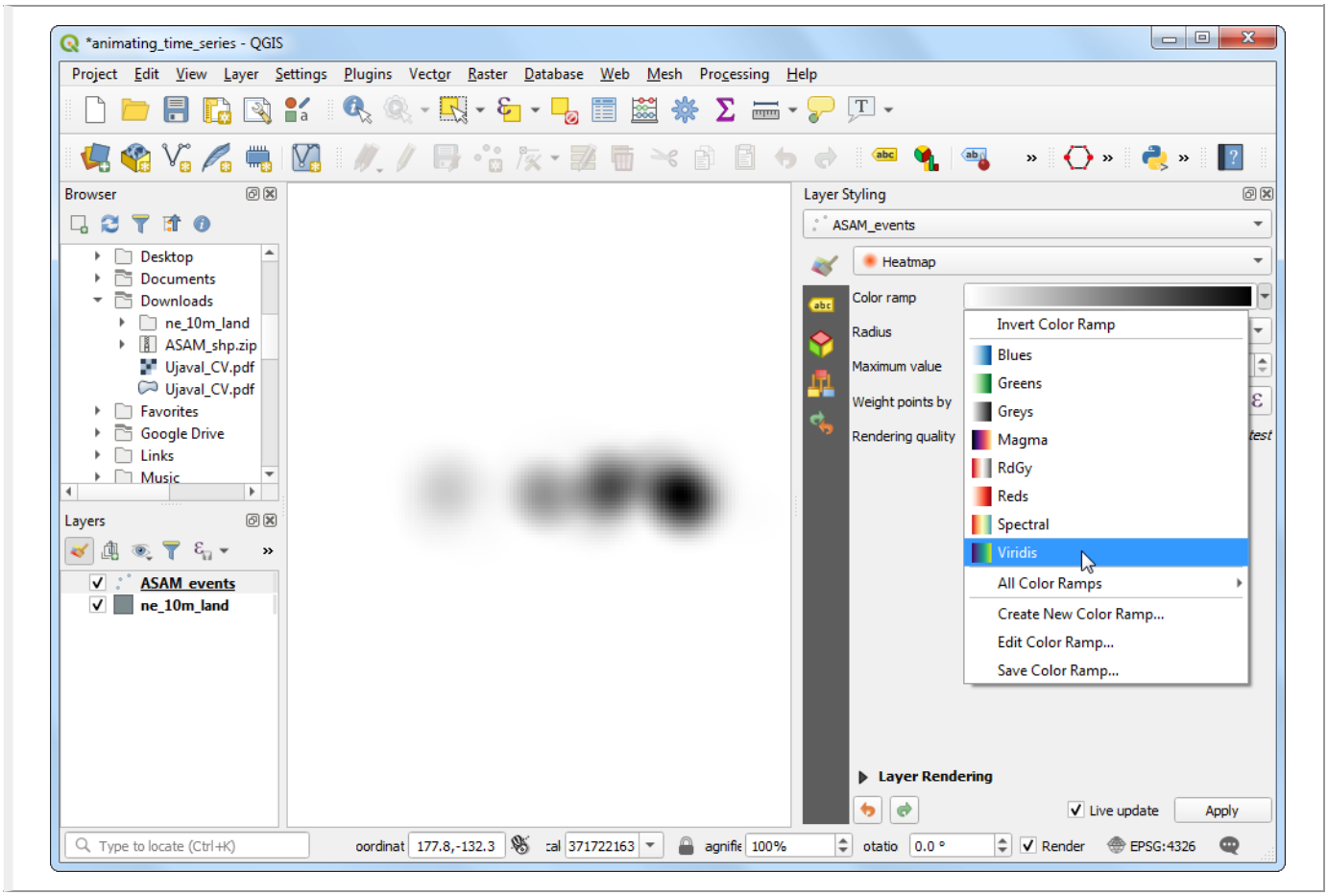
1. In the QGIS Browser Panel, locate the directory where you saved your downloaded data. Expand the `ne_10m_land.zip` and select the `ne_10m_land.shp` layer. Drag the layer to the canvas. Next, locate the `ASAM_shp.zip` file. Expand it and select the `asam_data_download/ASAM_events.shp` layer and drag it on to the canvas.



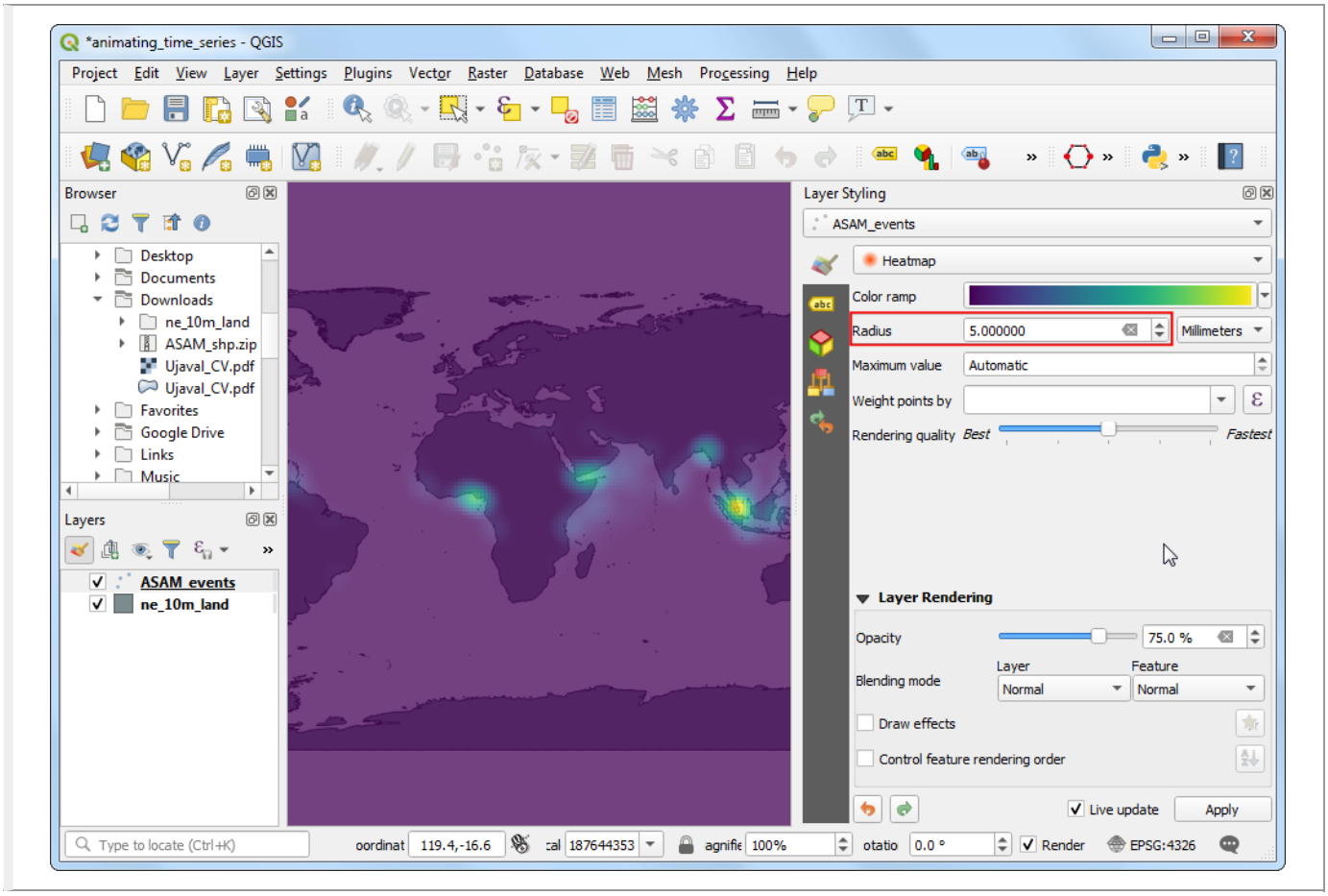
2. Once the layer is loaded, you can see the individual points representing incidents of piracy locations. There are thousands of incidents and it is difficult to determine with more piracy. Rather than individual points, a better way to visualize this data is through a heatmap. Select the `ASAM_events` layers and click the Open the layer Styling Panel button in the Layers panel. Click the `Single symbol` drop-down.



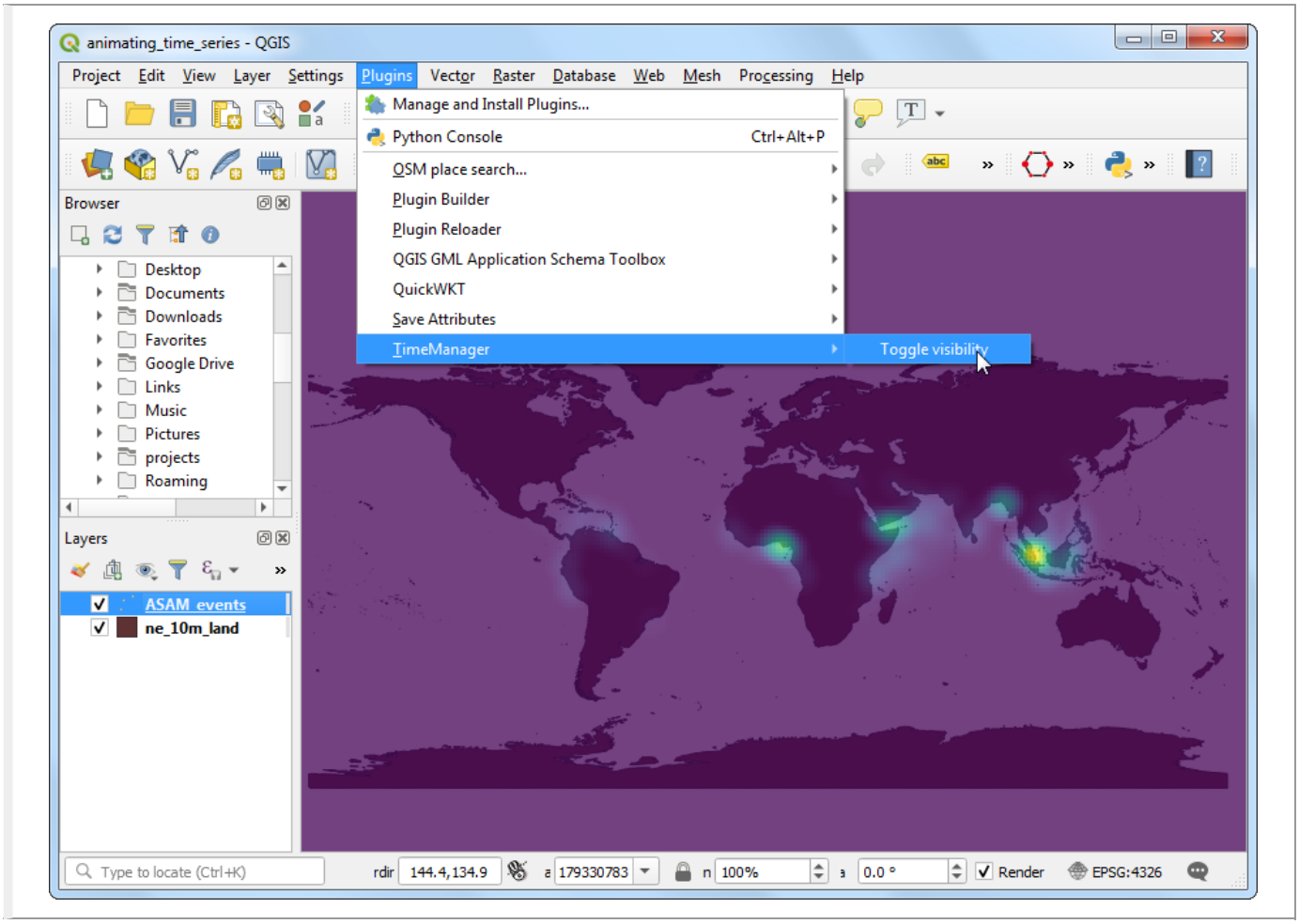
3. In the renderer selection drop-down, select `Heatmap` renderer. Next, select the `Viridis` color ramp from the Color ramp selector.



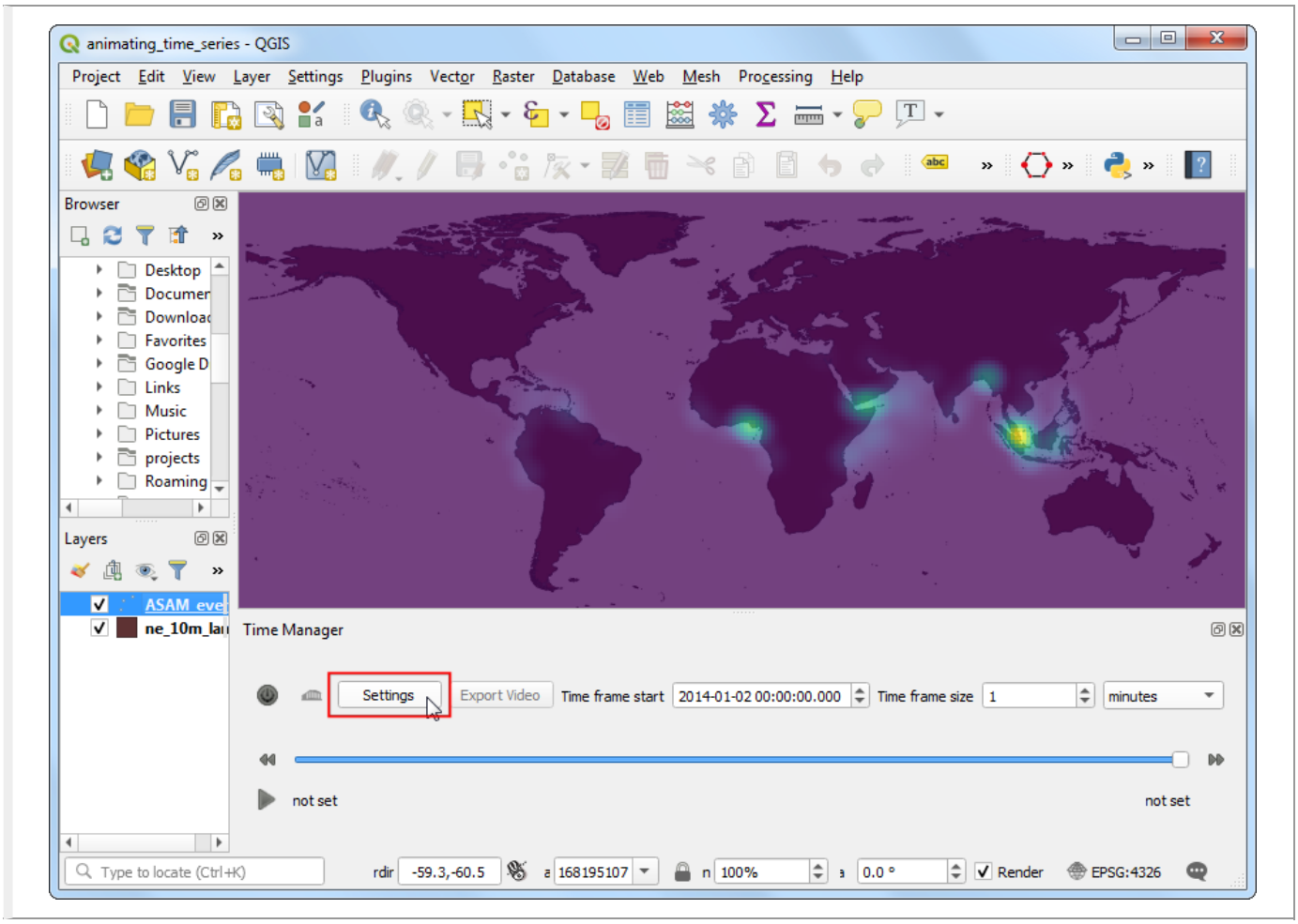
4. Adjust the Radius value to `5.0`. At the bottom, expand the Layer Rendering section and adjust the Opacity to `75.0%`. This gives a nice visual effect of the hotspots with the land layer below.



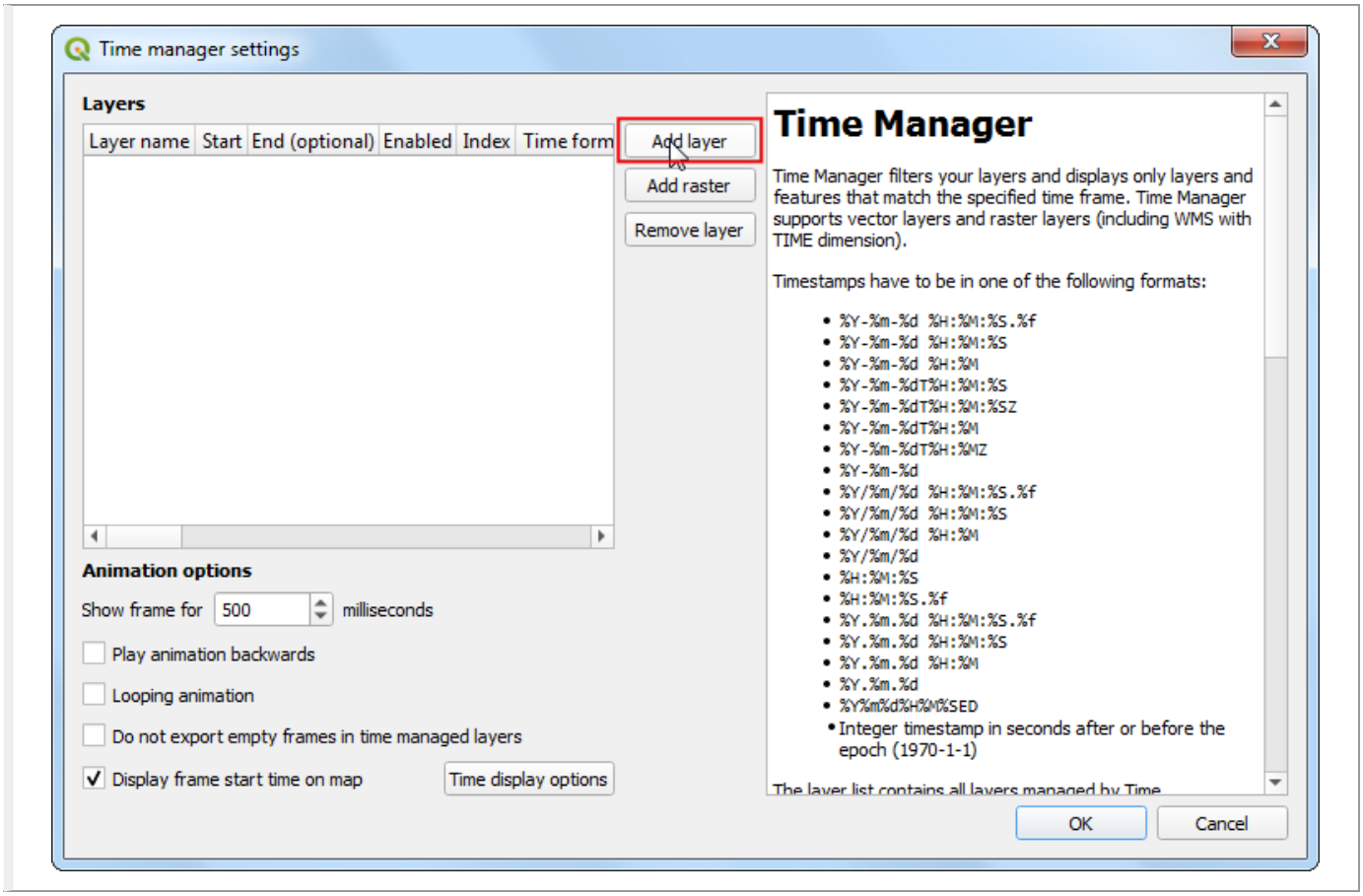
5. Now let's animate this data to show the yearly map of piracy incidents. Go to Plugins ‣ TimeManager ‣ Toggle visibility.



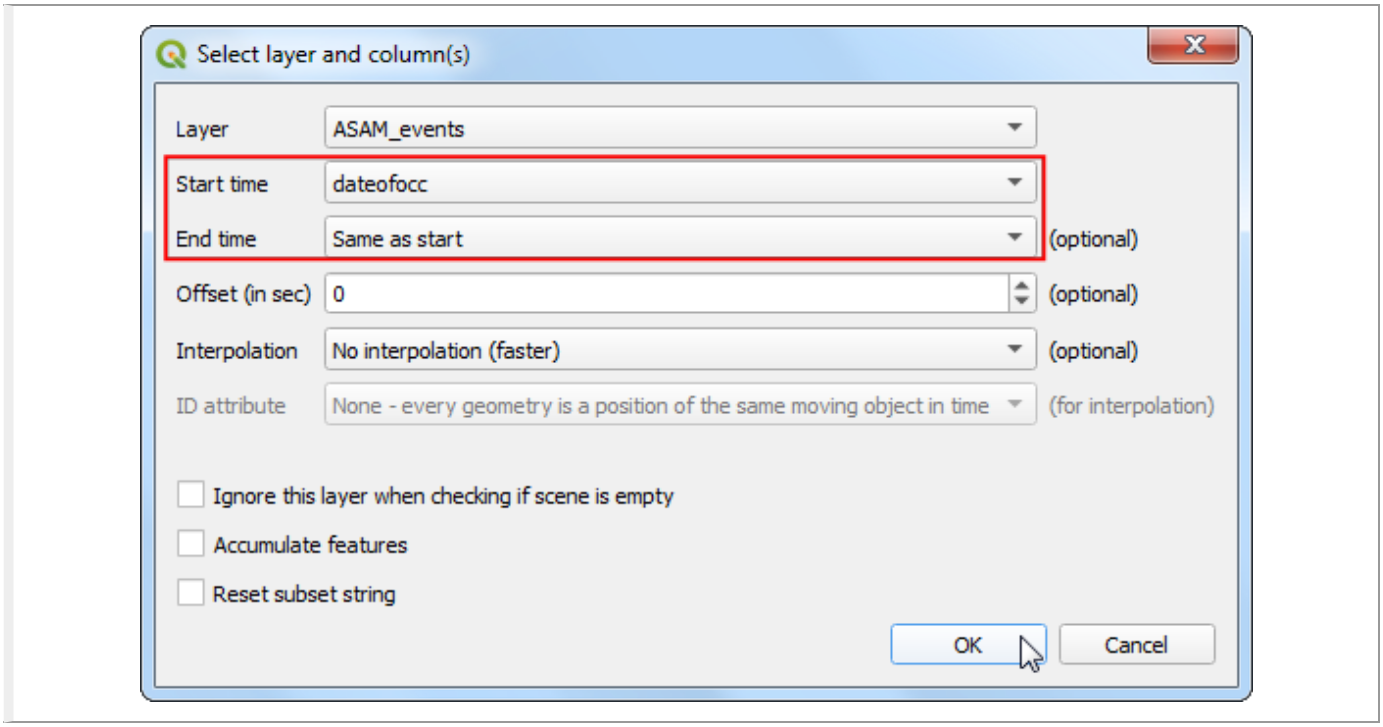
6. In the TimeManager panel, click Settings.



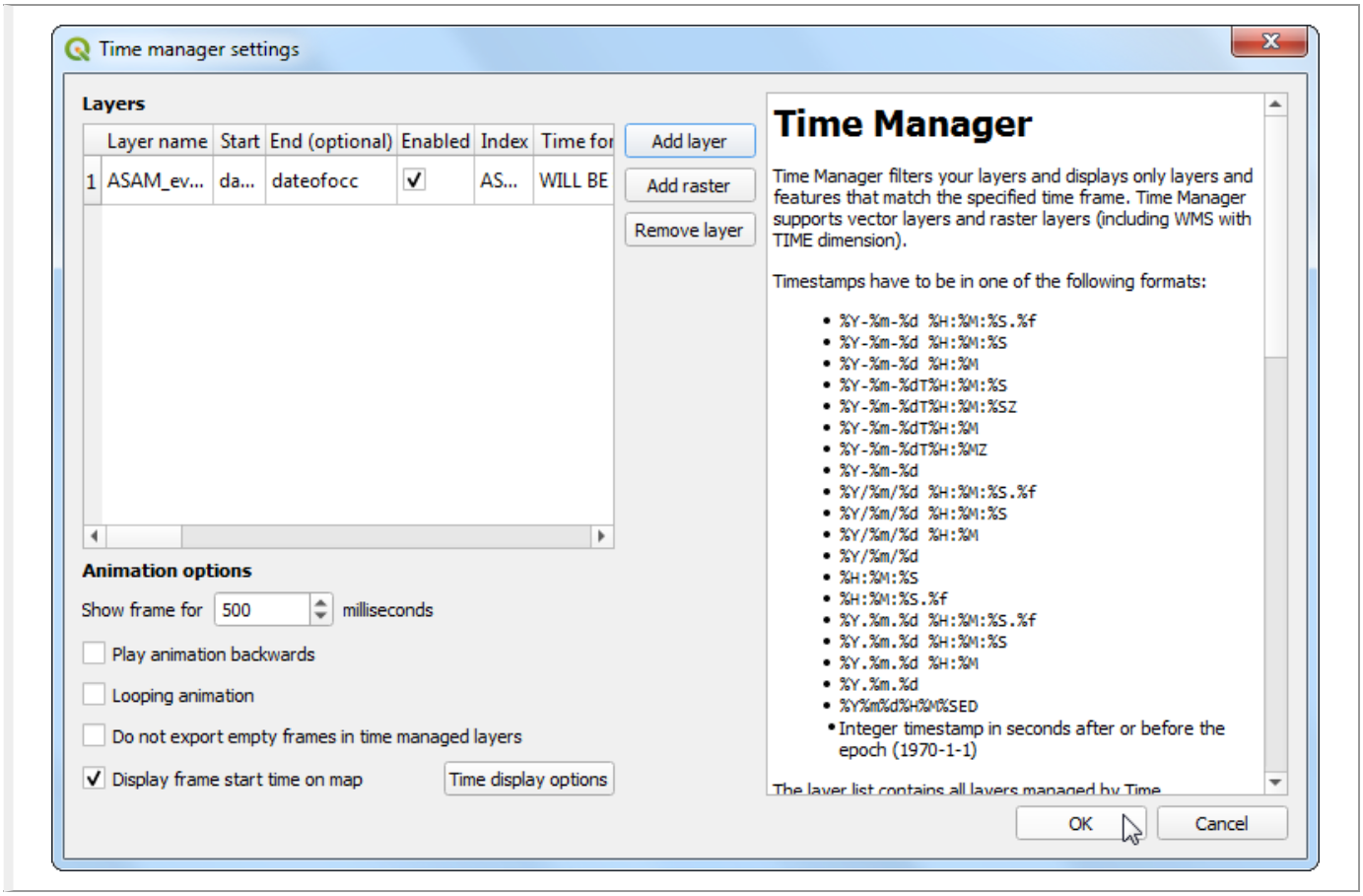
7. In the Time manager settings window, click Add layer button.



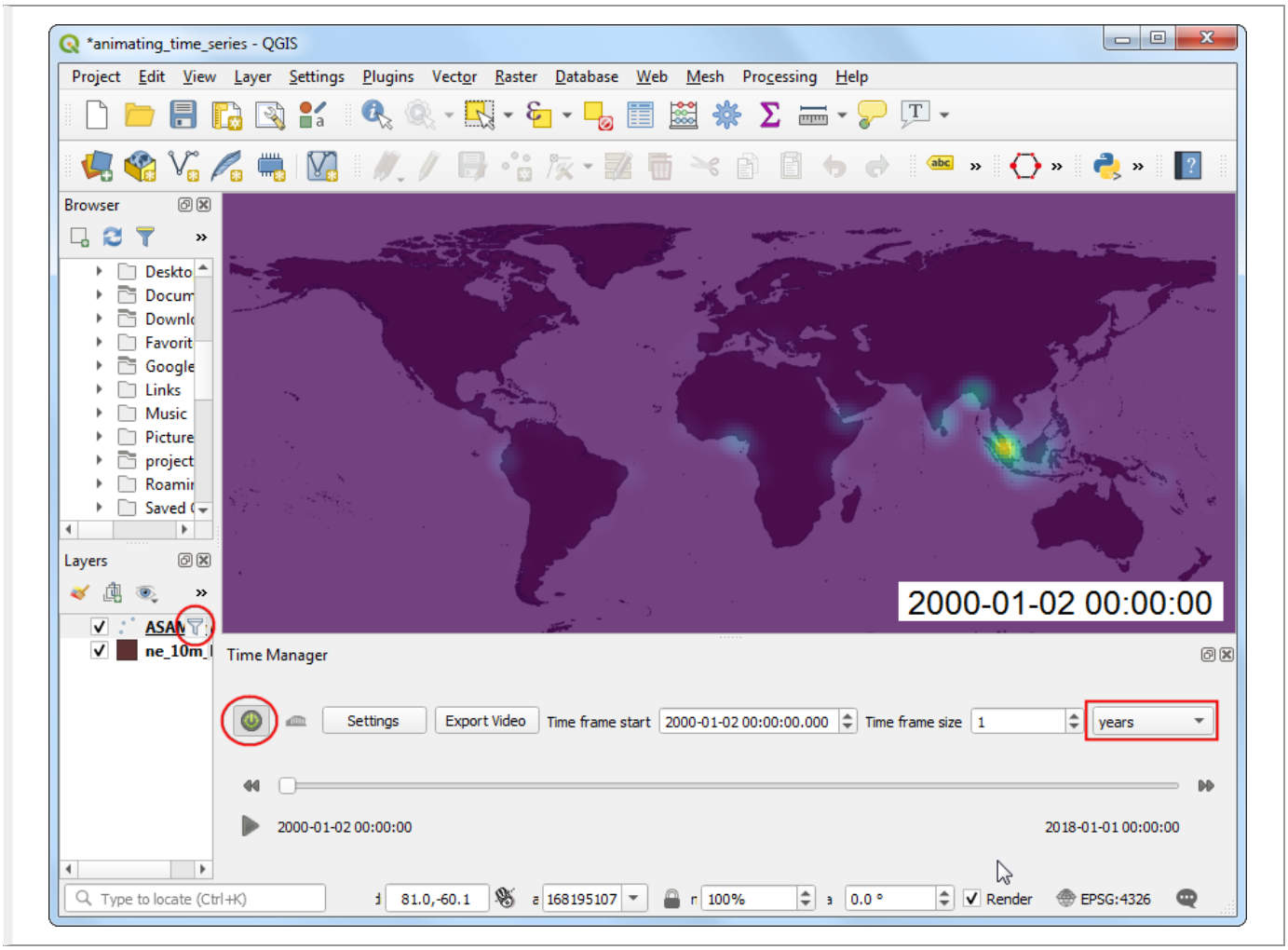
8. The source data contains an attribute `dateofocc` - representing the date on which the incident took place. This is the field that will be used by the plugin to determine the points that are rendered for each time period. Select `ASAM_events` as the Layer and `dateofocc` as the Start time. The End time should be set to `Same as start`. Click OK.



9. Back in the Time manager settings window, click OK.



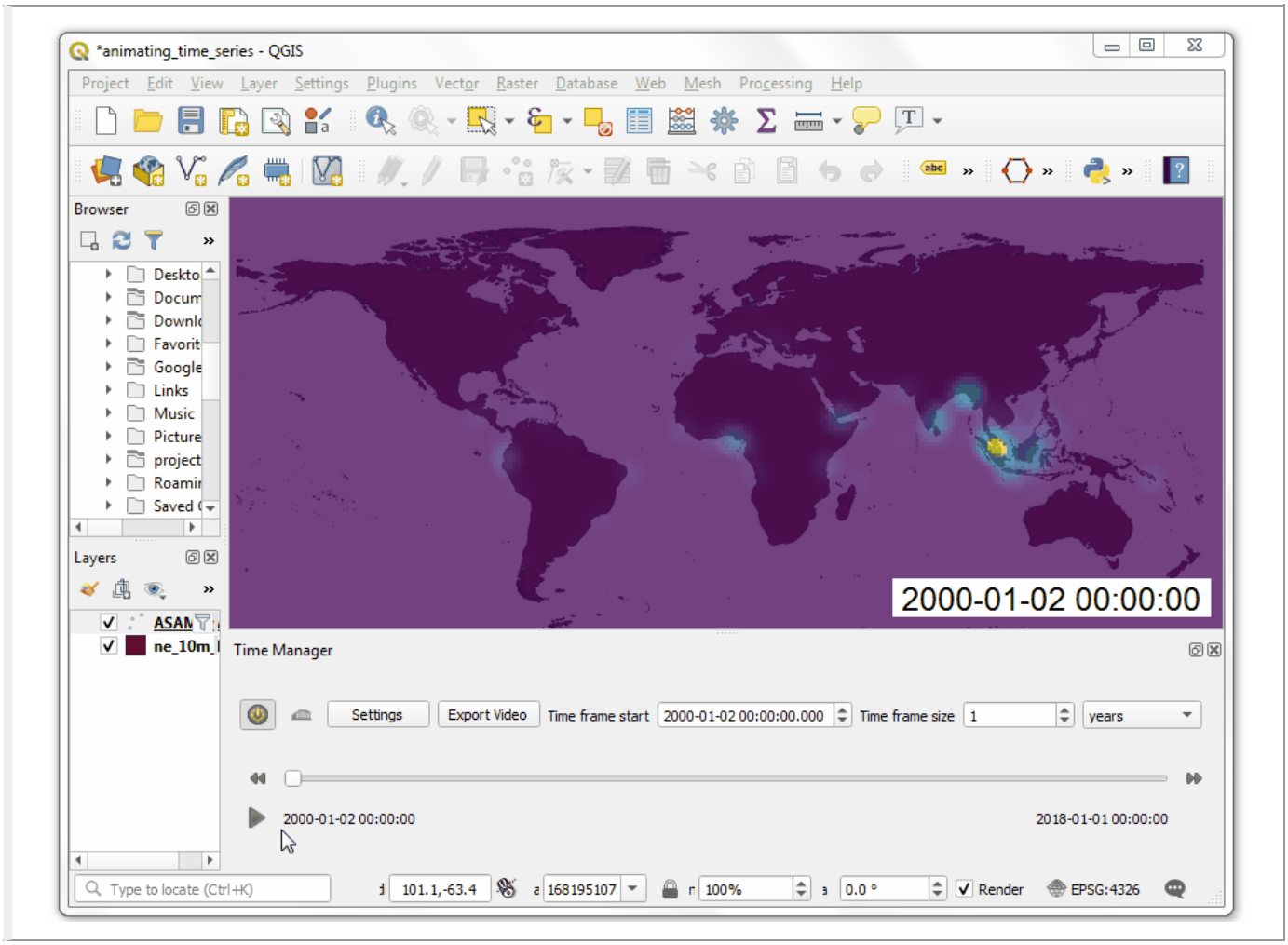
10. Click the Power icon in the TimeManager panel to enable the plugin. Set the Time frame size to be `1 years`. Once enabled, you will see a filter icon next to the `ASAM_events` layer. TimeManager works by applying a filter to the layer based on the selected field and specified time period.



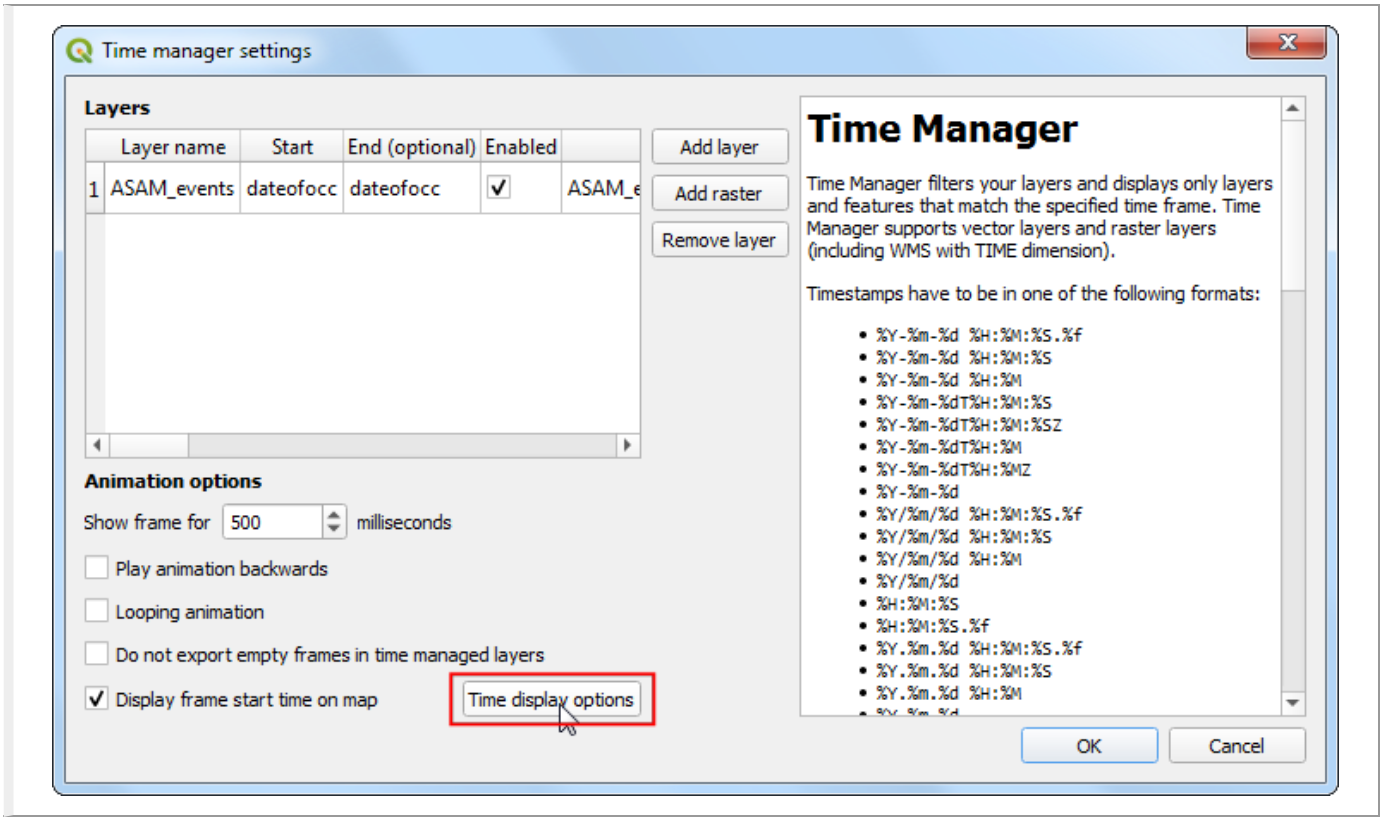
Note

As TimeManager works by applying a filter on the layer, it only works with layer types that support this feature. Most data source types do support it - with a notable exception being temporary memory layers. If you had done some processing earlier and have a temporary layer, right-click and select Make Permanent before using TimeManager on that layer.

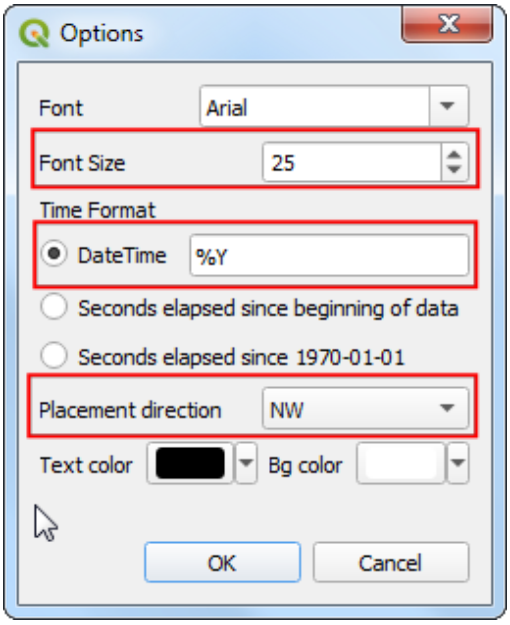
11. Now you are ready to see the animation. Click the Play button to see the yearly piracy hotspot animation.



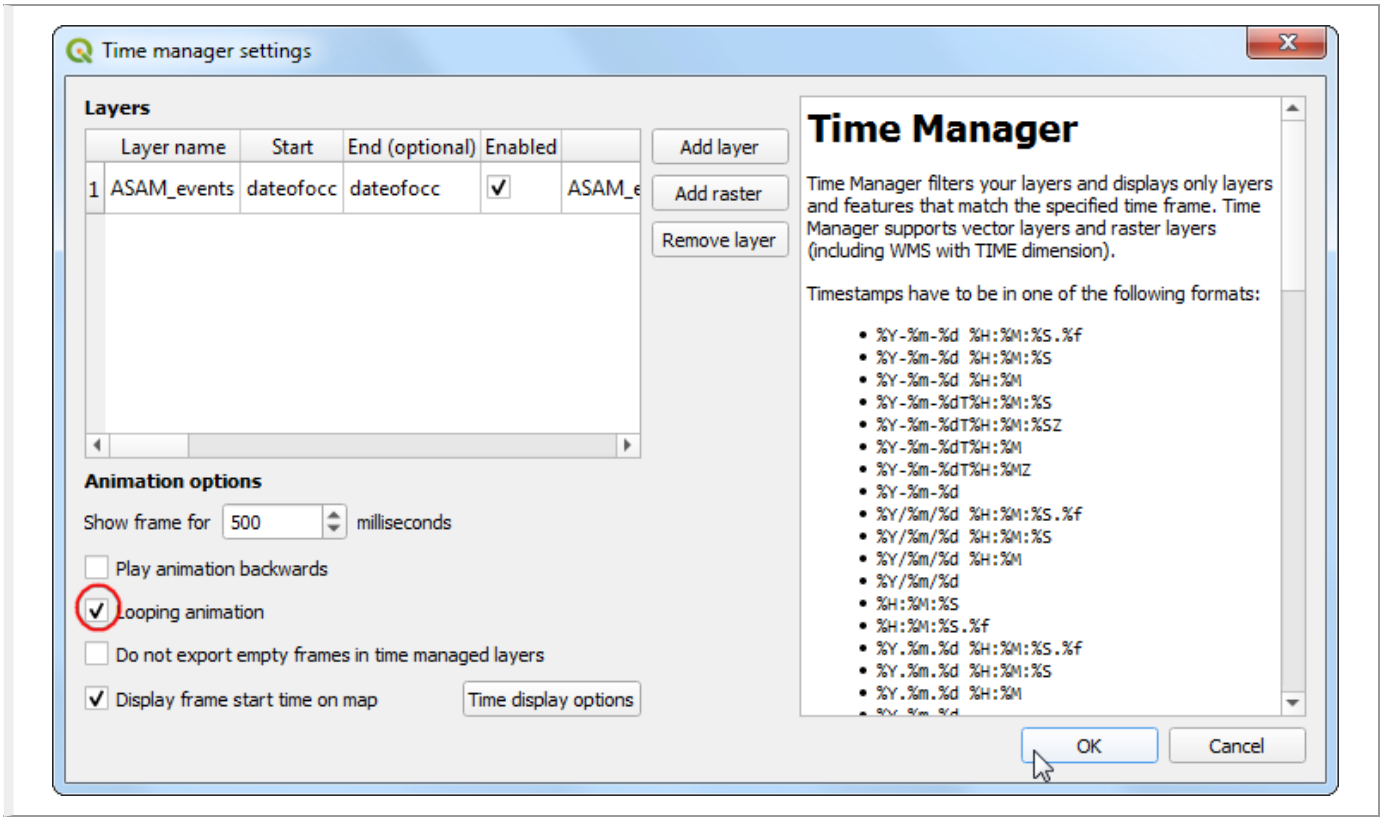
12. You will notice that for each frame of the animation, a date is displayed at the bottom-right. Instead of the full date and time, let's change it to display the year that the map represents. Click Settings in the Time Manager panel. Click Time display options in the Time manager settings dialog.



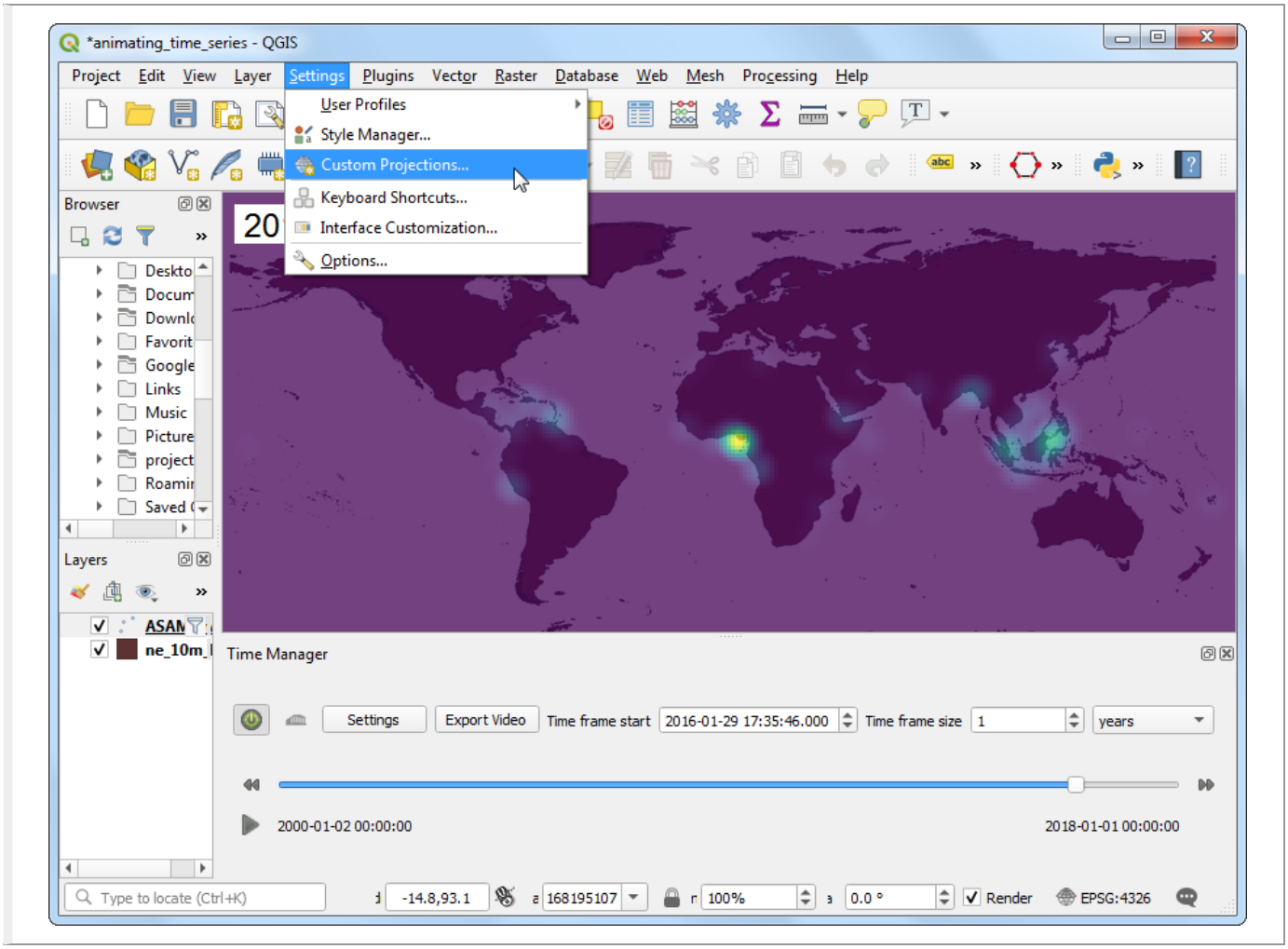
13. Adjust the Font Size to `25`. Change the DateTime format to `%Y`. The time format should be specified in the Python strftime (https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior) format. `%Y` is the short-code for a 4 digit year. Also you can change the Placement direction to `NW`. Click OK.



14. Back in Time manager settings dialog, check the Looping animation checkbox. This helps when you are making changes to styling and adjusting styling to make the animation continue playing back from start. Click OK.

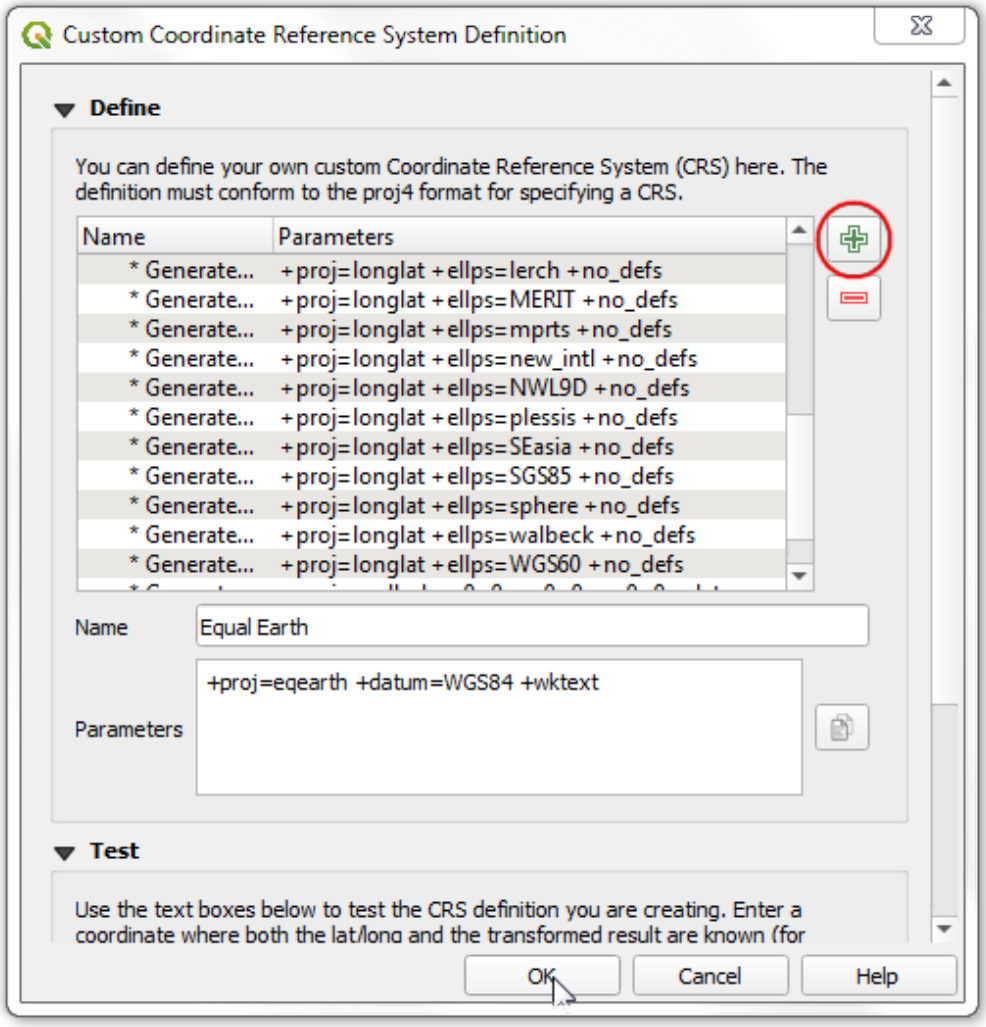


15. Now if you replay the animation, you will see the label will show the year of the animation in the top-left corner. At this point, we can export the animation, but there is one more change that we can apply to make our map better. The default map projection is `EPSG:4326` which is ok for storing the source data, but not ideal for global visualization like this. I really like the Equal Earth Projection (http://equal-earth.com/equal-earth-projection.html) for a visually pleasing and more accurate representation of the world. It is a fairly new projection and not yet available as a predefined option in QGIS. But there is an easy way to use it in QGIS by defining a custom projection. Go to Settings ‣ Custom Projections....

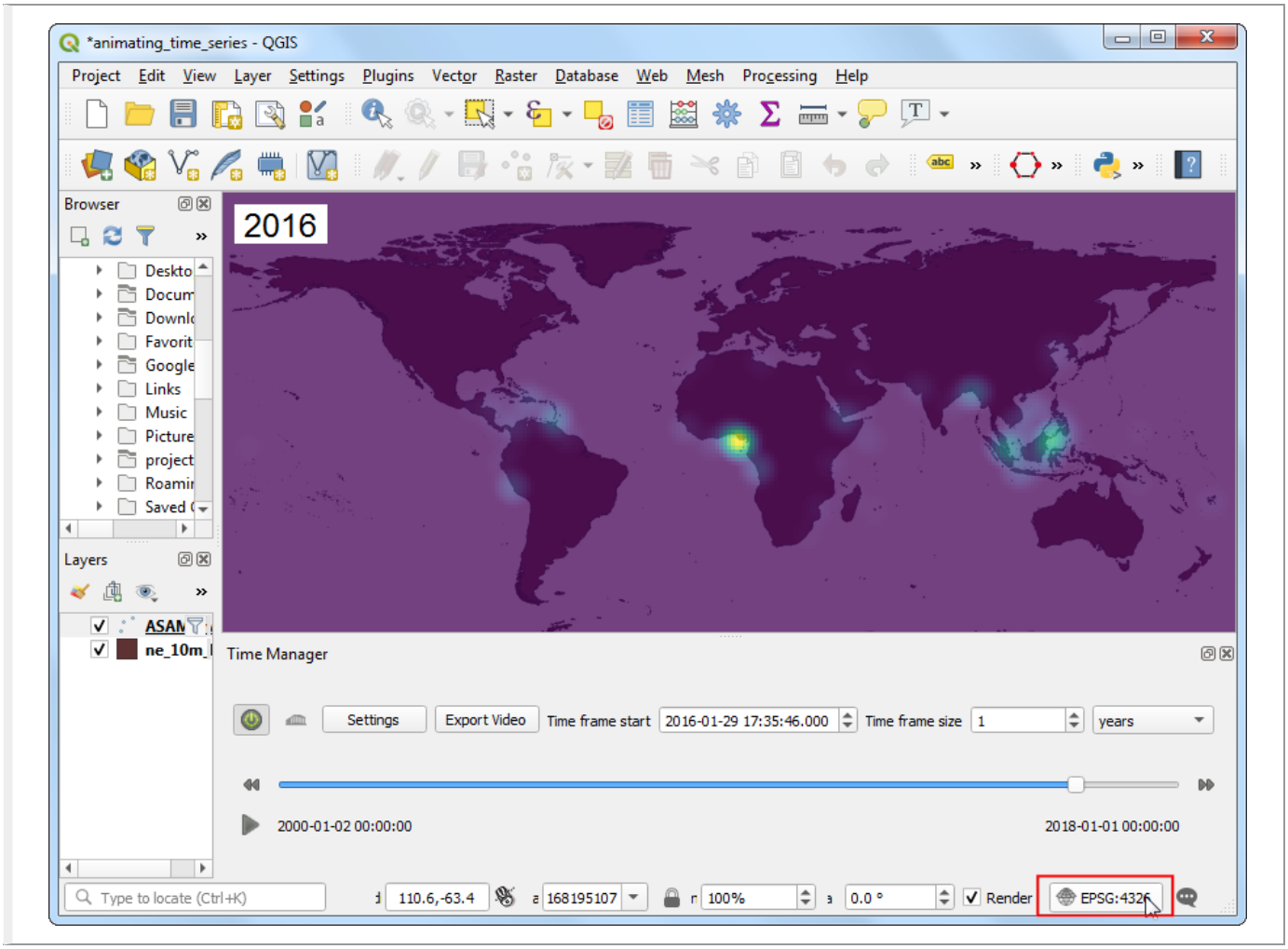


16. In the Custom Coordinate Reference System Definition dialog, click the + button. Enter `Equal Earth` as the name. Enter the following definition in the Parameters box. The parameters need to be specified in the PROJ format (https://proj.org/operations/projections/eqearth.html). After entering the parameters, click OK.

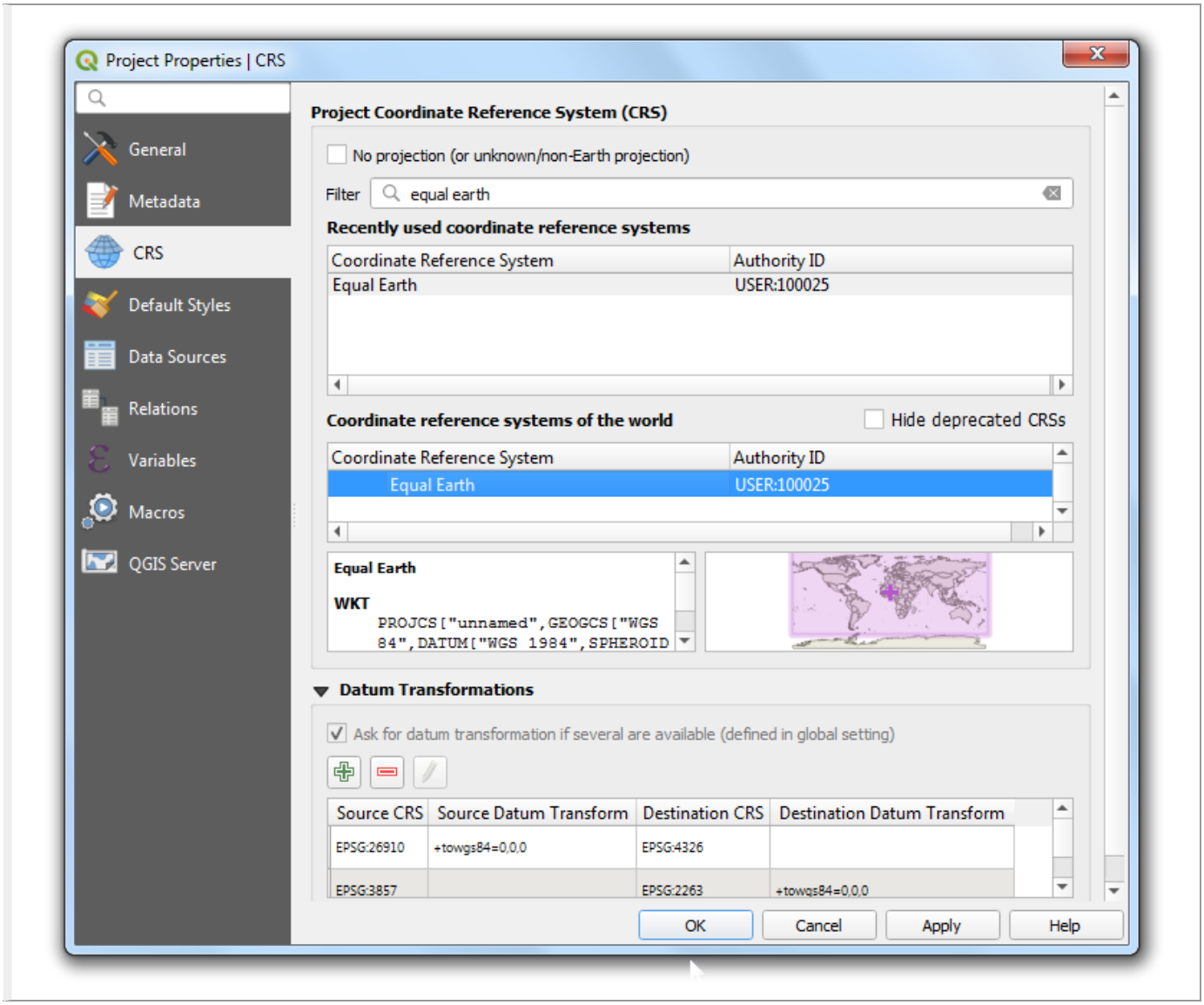
```
+proj=eqearth +datum=WGS84 +wktext
```



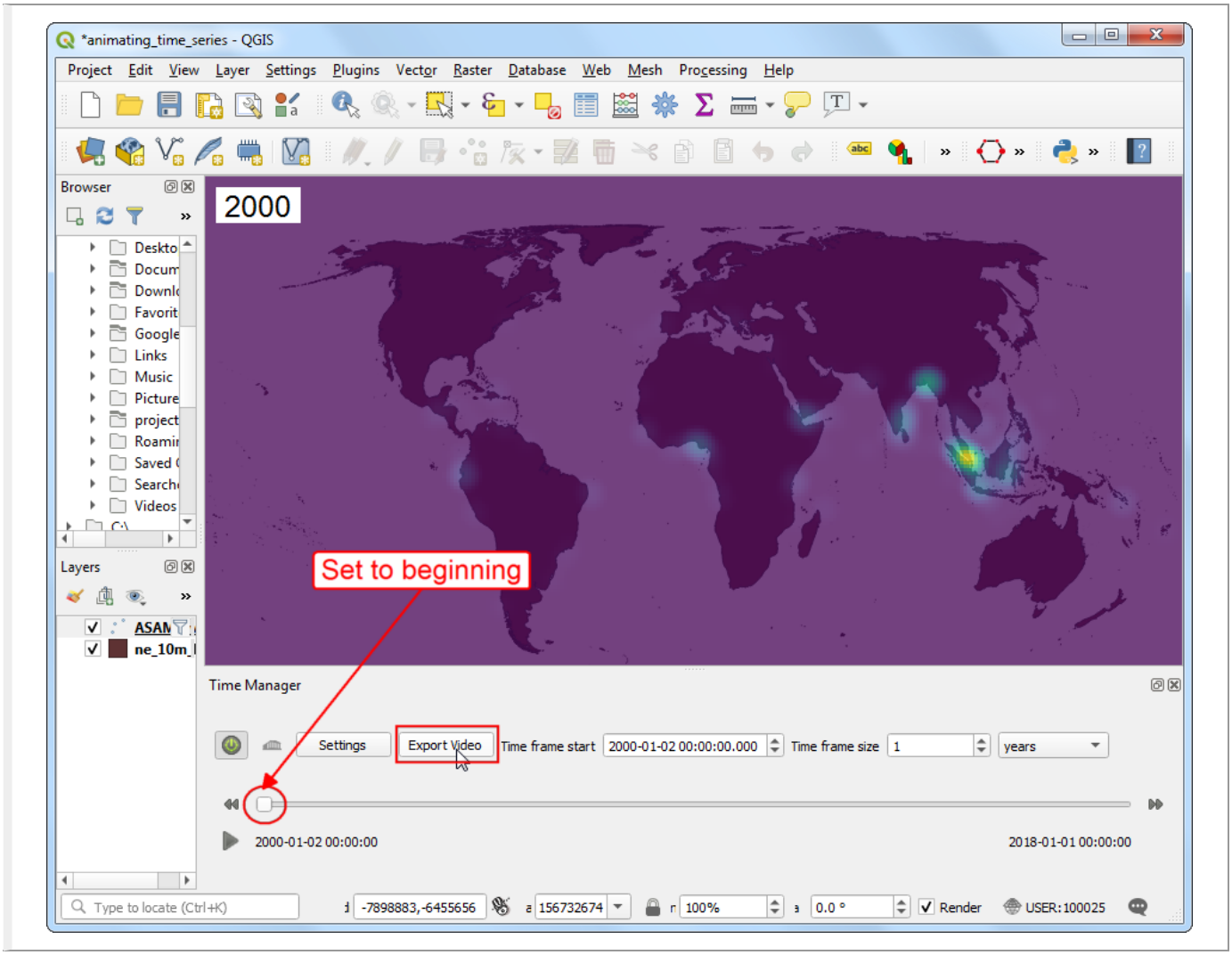
17. In the main QGIS window, click the Current CRS display on the bottom-right corner.



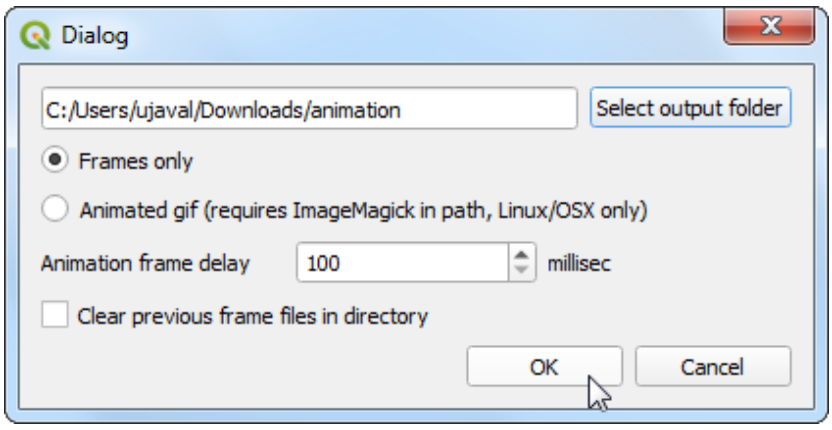
18. Search for `Equal Earth` to find and select the newly defined projection. Click OK.



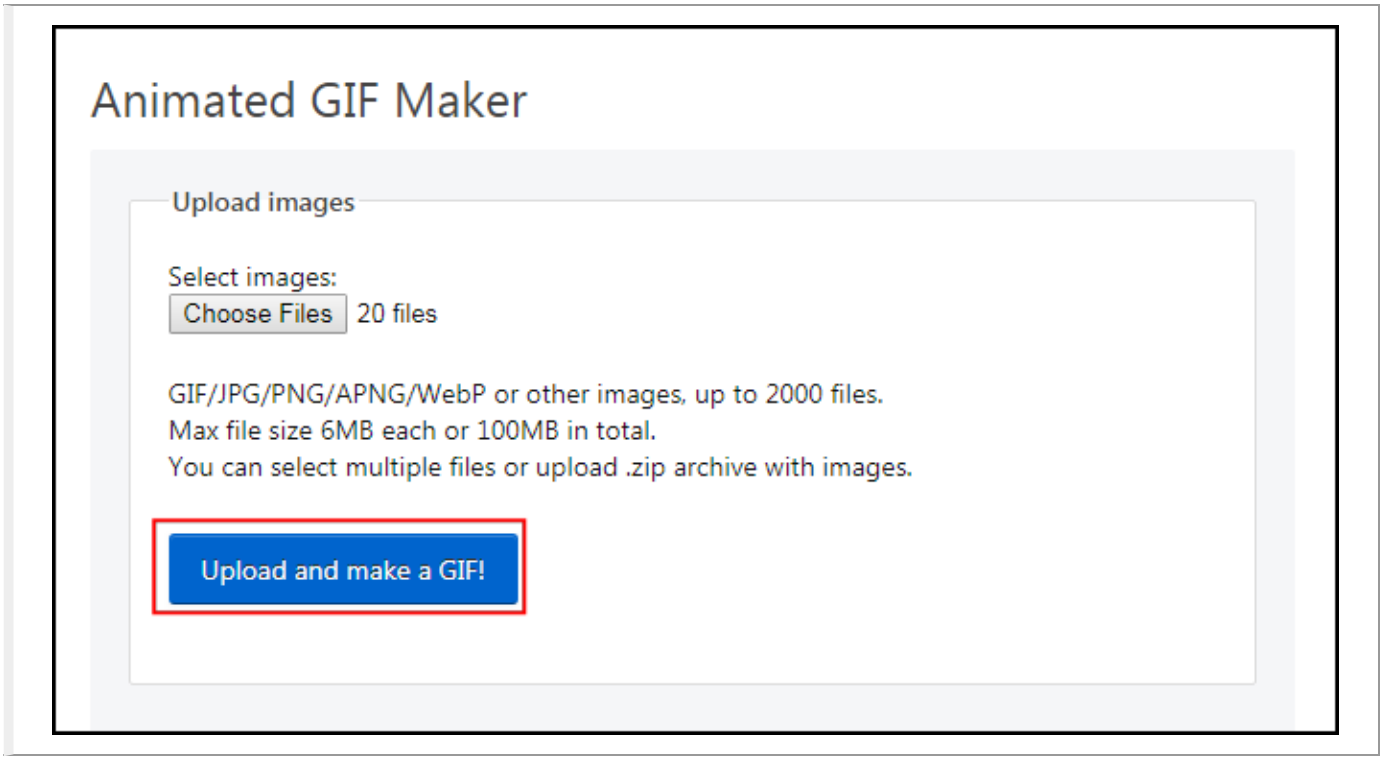
19. You will see the map transform to the Equal Earth projection. Now we are ready to export the animation. Before exporting, make sure to set the time-slider in the Time Manager panel to the start position. Export of the animation will start from the current position of the time slider. Click Export Video button in the Time Manager panel.



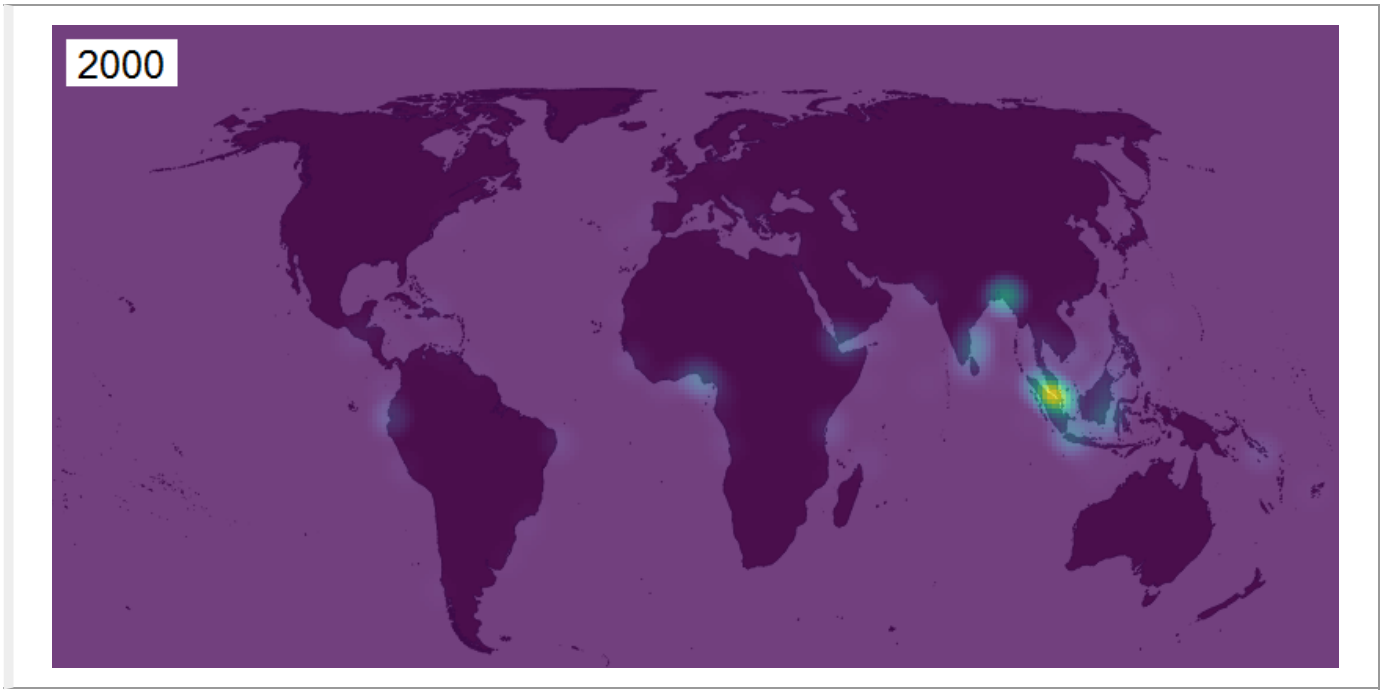
20. In the Export Video dialog, click Select output folder and select a directory on your computer. Select the Frames only option and click OK to start the export process.



21. Once the export finishes, you will see PNG images for each year in the output directory. Now let's create an animated GIF from these images. There are many options for creating animations from individual image frames. I like ezgif.com (https://ezgif.com/maker) for an easy and online tool. Visit the site and click Choose Files and select all the `.png` files. Note that the export folder will also have a `.pgw` file for each frame which contains the georeference information. You may want to sort the images by `Type` to allow easy bulk selection of only `.png` files. Once selected, click the Upload and make a GIF! button.



22. Once the process finishes, click the Save button to download the GIF to your computer.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Performing Spatial Queries (QGIS3)

Spatial queries are core to many types of GIS analysis. Spatial queries allows you to select features in a layer by their spatial relationships (intersect, contain, touch etc.) with features from another layer. In QGIS, this functionality is available via the **Select by Location** and **Extract by Location** Processing tools.

Overview of the task

We will be working with 2 data layers for the city of Melbourne, Australia. Given the data layers for the pubs and bars in the city and locations of all metro stations, we want to find out all bars and pubs within 500 meters of a metro station.

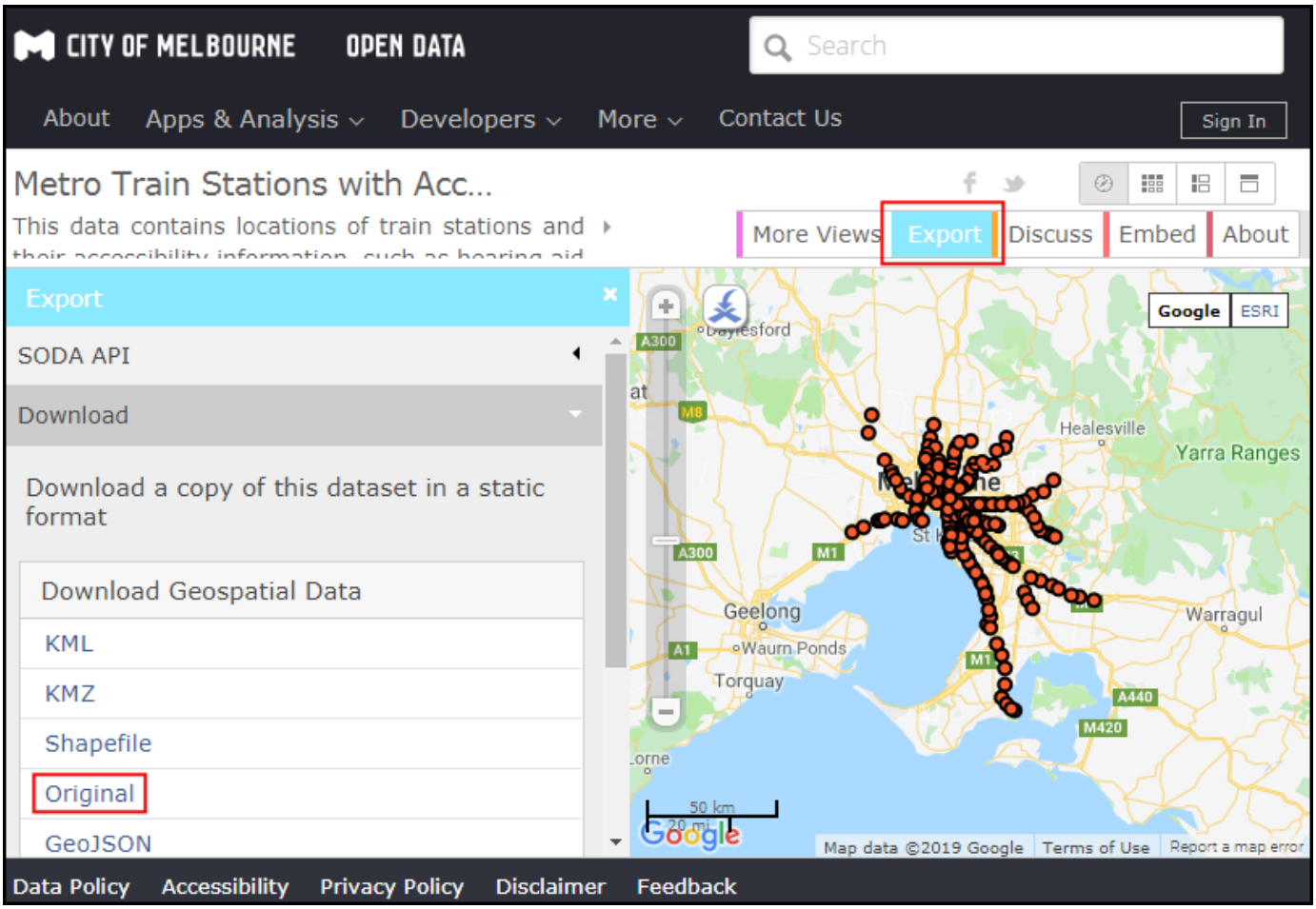
Other skills you will learn

- Choosing an appropriate projection and re-projecting vector data.
- Creating buffers.
- Working with the geopackage (.gpkg) data format.

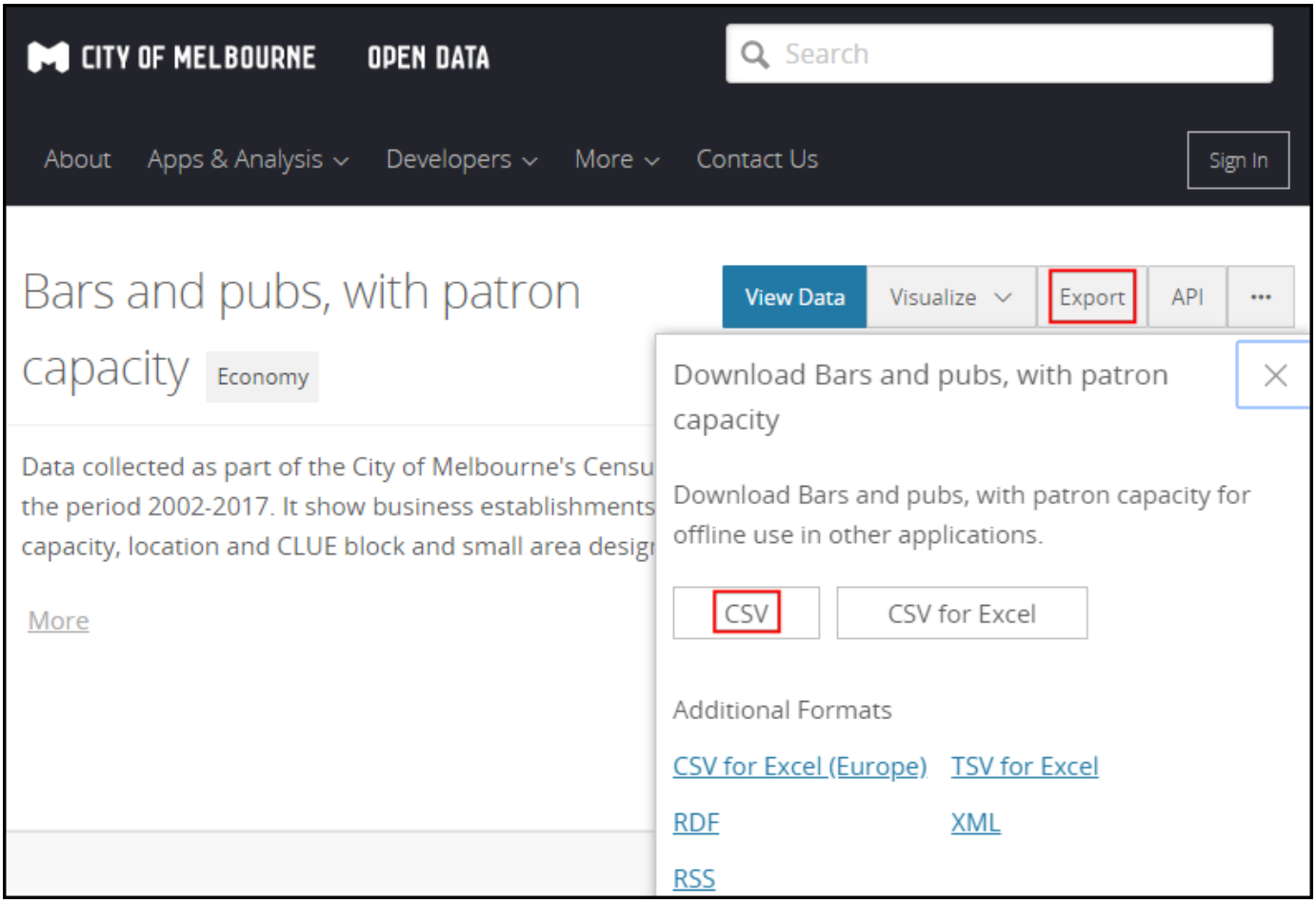
Get the data

City of Melbourne's Open Data Platform (https://data.melbourne.vic.gov.au/) provides many GIS-ready datasets for the city.

Download the Metro Train Stations with Accessibility Information (https://data.melbourne.vic.gov.au/Transport-Movement/Metro-Train-Stations-with-Accessibility-Informatio/mgkp-67ad) dataset by Metro Trains Melbourne. Export the data in the *Original* format.



Download the Bars and pubs, with patron capacity (https://data.melbourne.vic.gov.au/Economy/Bars-and-pubs-with-patron-capacity/mffi-m9yn) dataset by City of Melbourne's Census of Land Use and Employment (CLUE). Export the data as a *CSV*.



For convenience, you may directly download a copy of datasets from the link below:

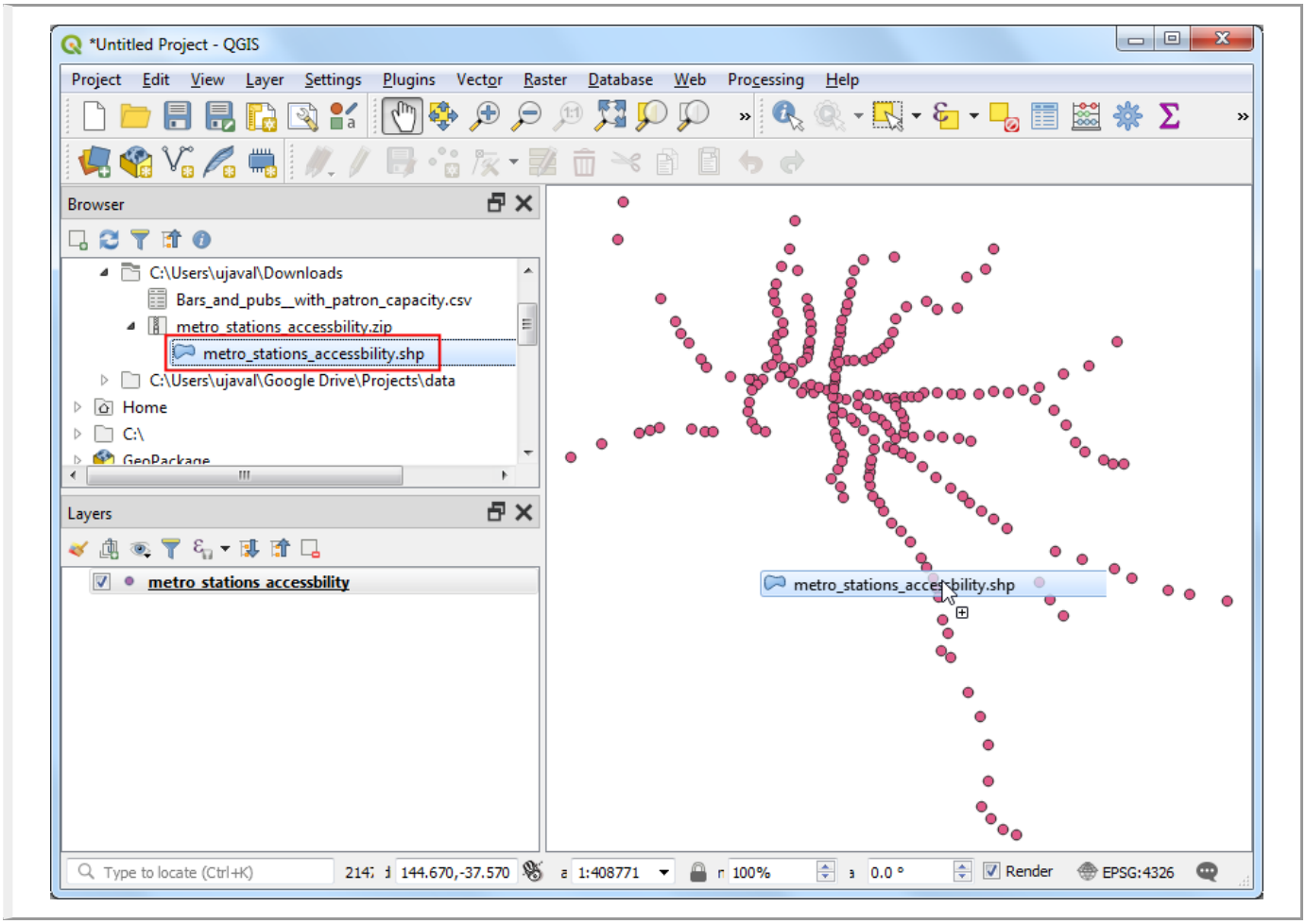
metro_stations_accessbility.zip (http://www.qgistutorials.com/downloads/metro_stations_accessbility.zip)

Bars_and_pubs__with_patron_capacity (http://www.qgistutorials.com/downloads/Bars_and_pubs__with_patron_capacity.csv)

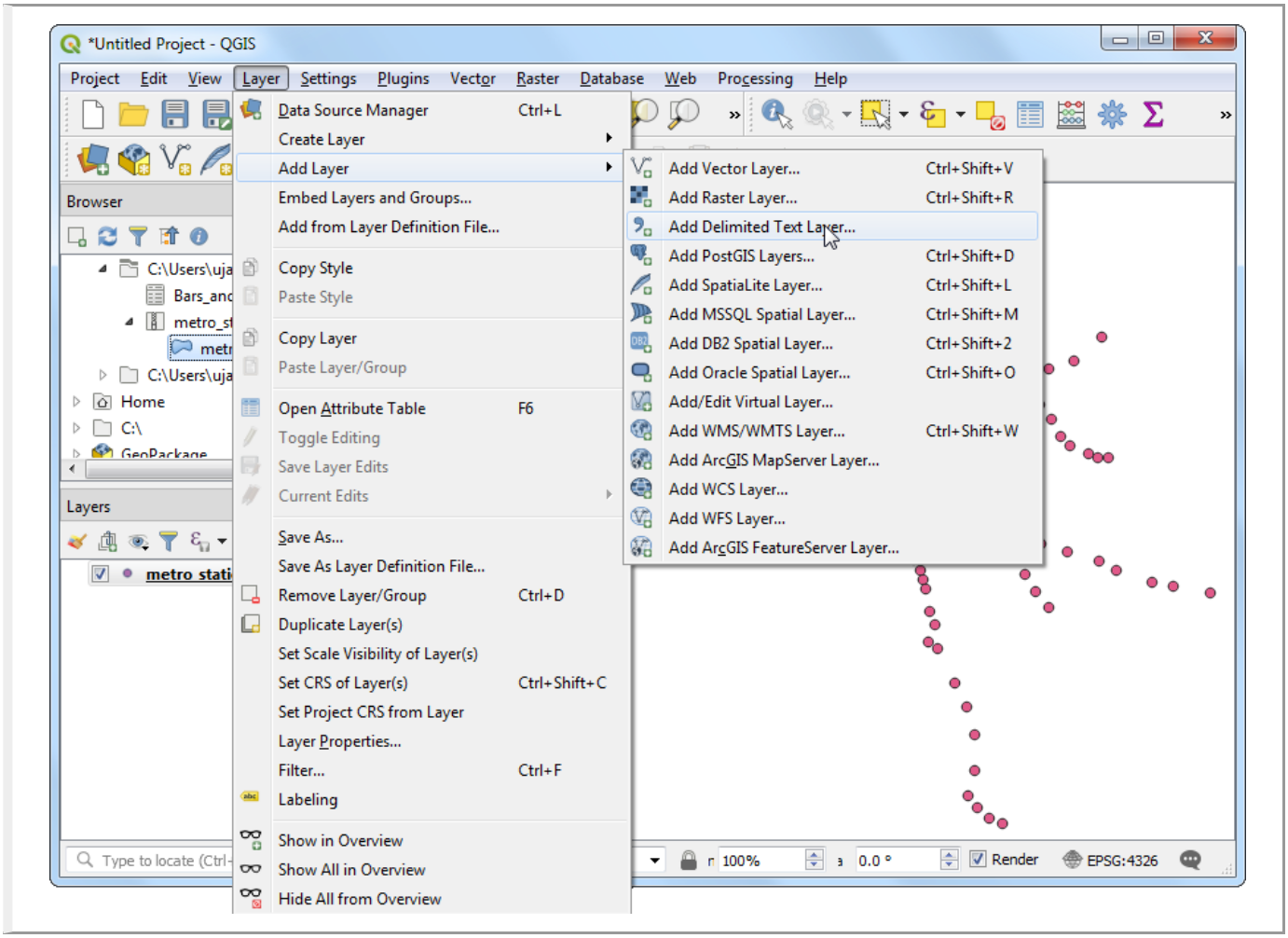
Data Source: [CITYOFMELBOURNE] (../credits.html#cityofmelbourne)

Procedure

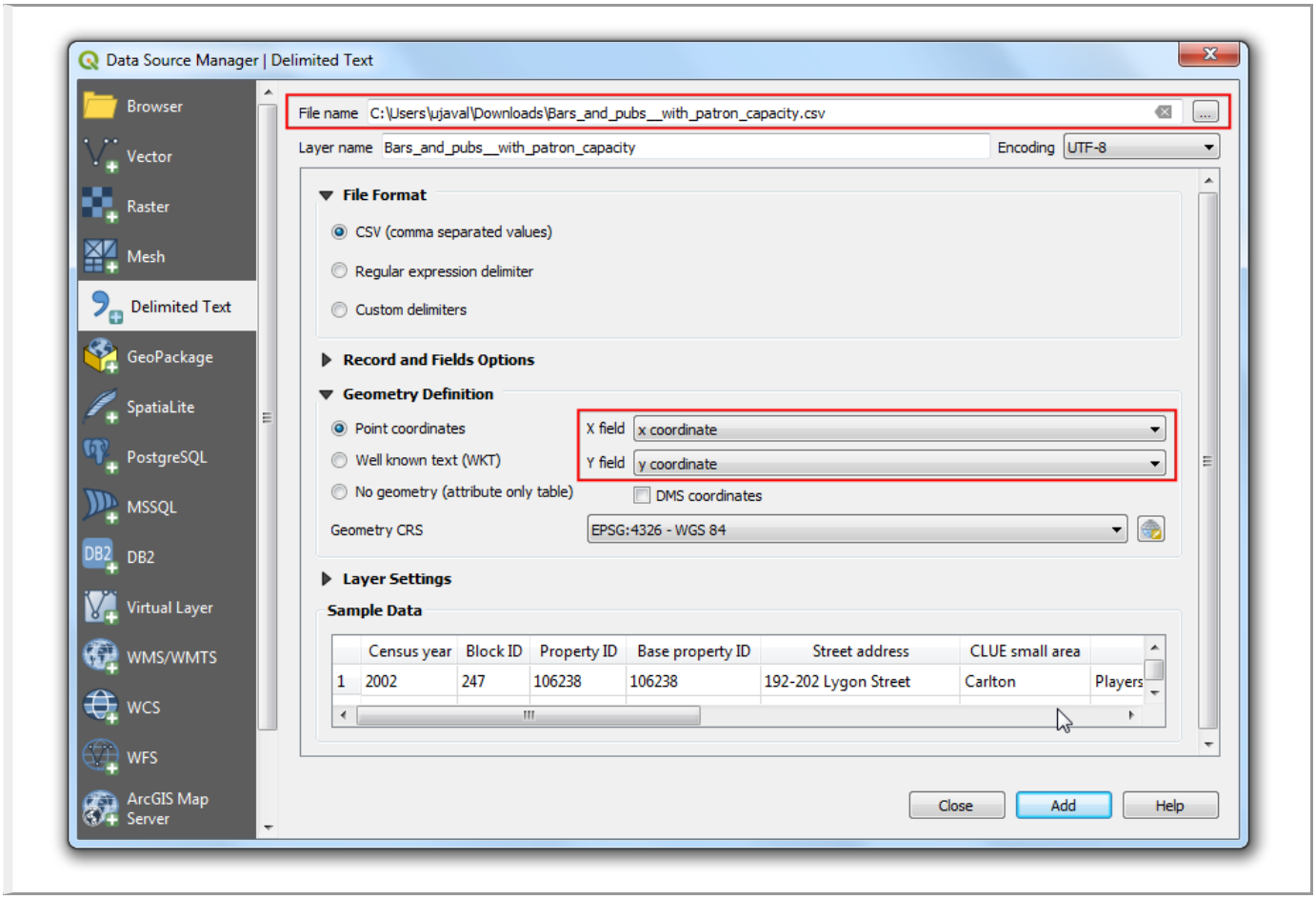
1. Locate the `metro_stations_accessbility.zip` file in the QGIS Browser and expand it. Select the `metro_stations_accessbility.shp` file and drag it to the canvas. A new layer `metro_stations_accessbility` will be loaded in the Layers panel.



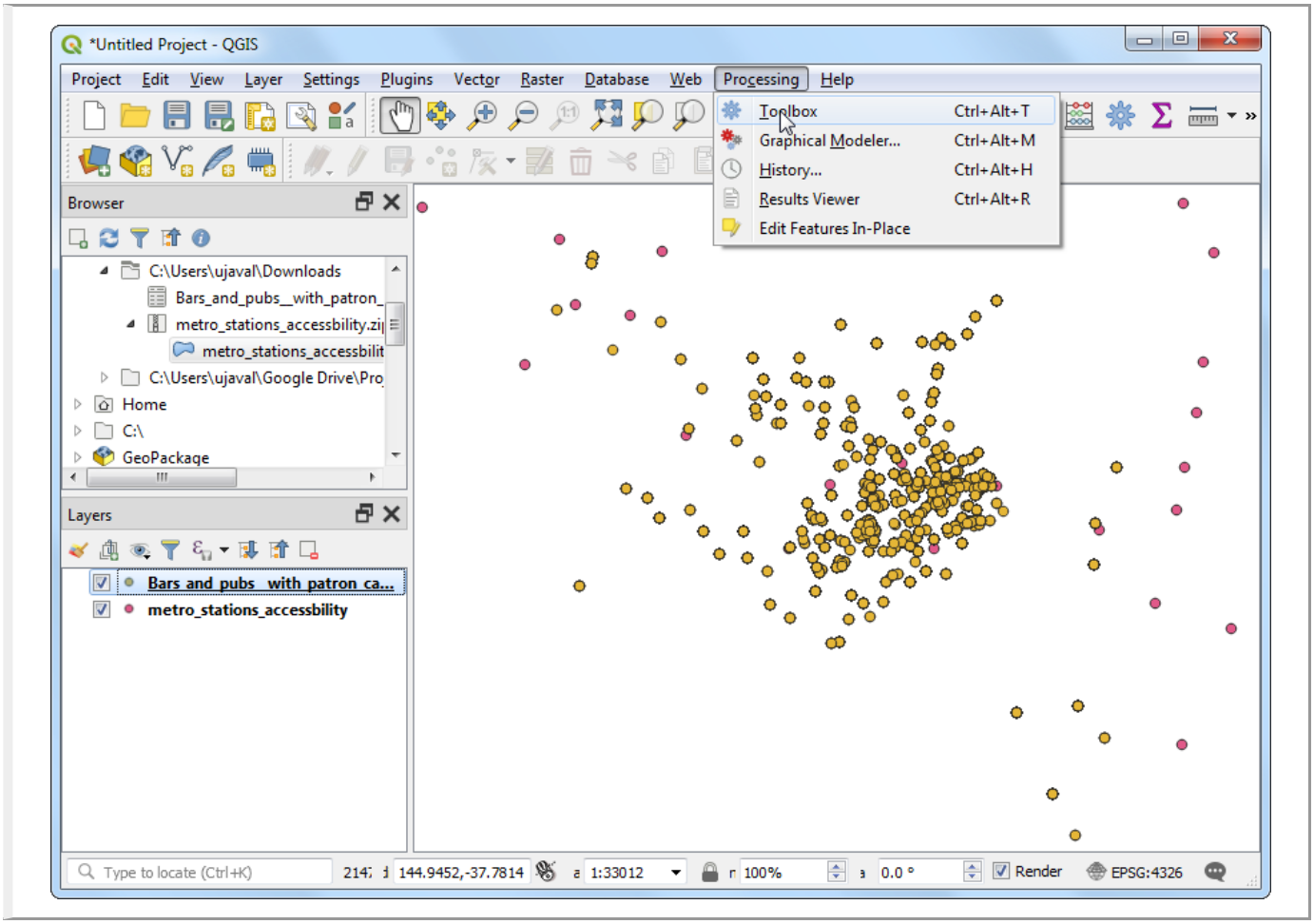
2. The data layer for bars and pubs is in the CSV format. To load it in QGIS, go to Layer ▸ Add Layer ▸ Add Delimited Text Layer.... (See *Importing Spreadsheets or CSV files (QGIS3)* (importing_spreadsheets_csv.html) for more details on importing CSV files)



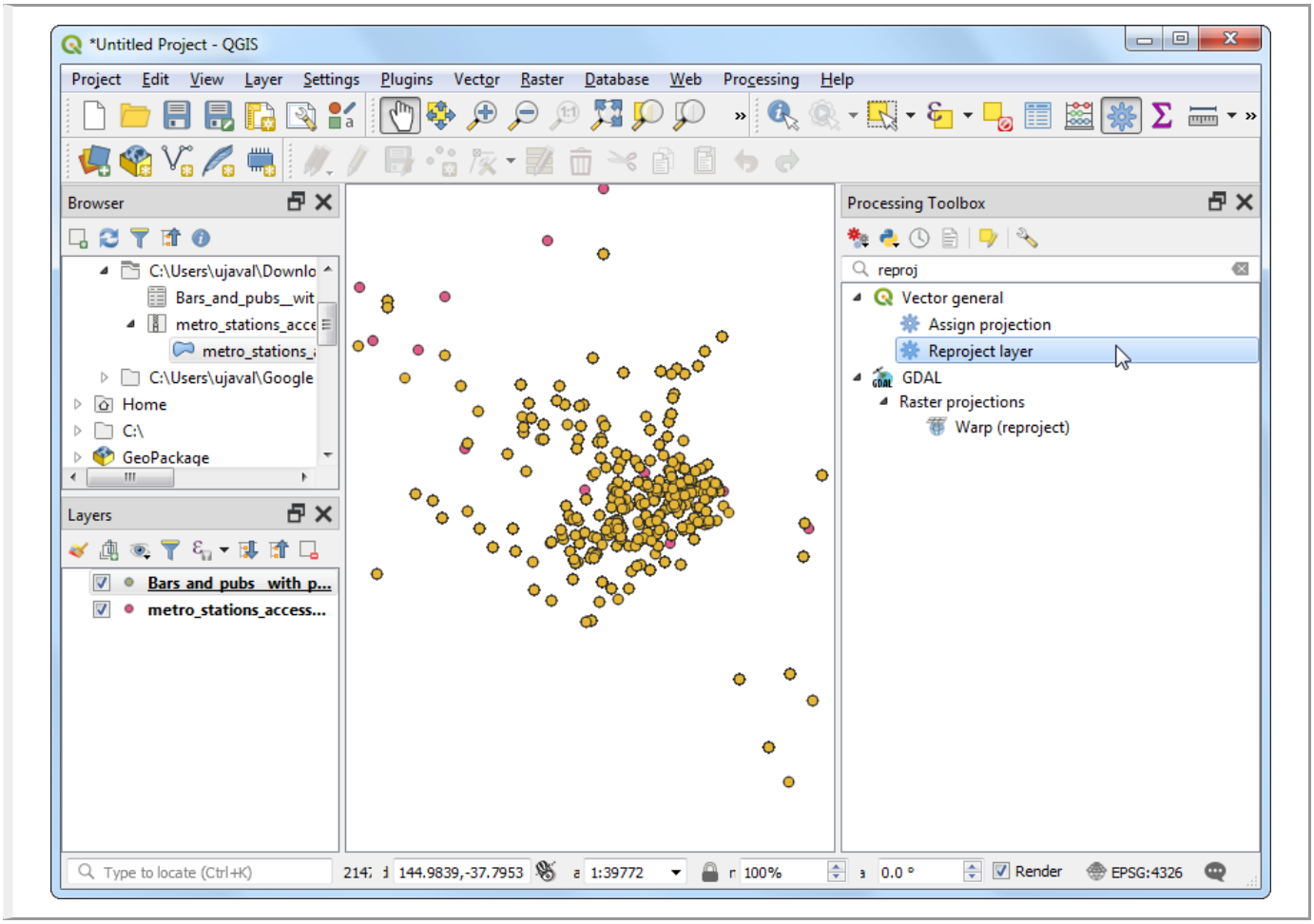
3. In the Data Source Manager | Delimited Text dialog, browse and select the downloaded `Bars_and_pubs__with_patron_capacity.csv` file as File name. The X field and Y field columns should be auto selected to `x coordinate` and `y coordinate` respectively. Click Add.



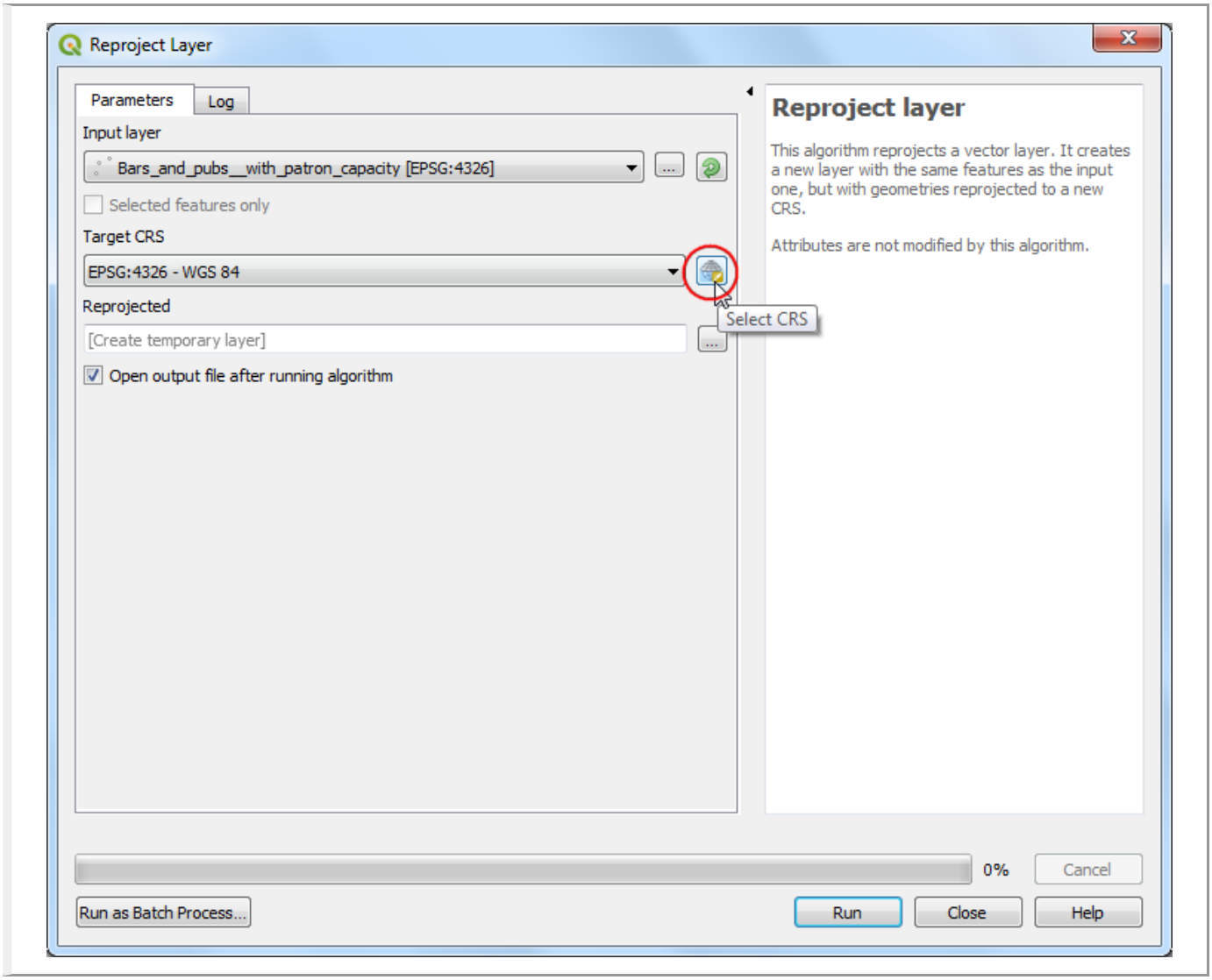
4. You will see a new `Bars_and_pubs__with_patron_capacity` layer added to the Layers panel. Both of the input layers are in the Geograhpic Coordinate Reference System (CRS) `EPSG:436 WGS84`. For performing spatial analysis, it is recommended to use a Projected Coordinate Reference System (CRS). So we will now re-project both the layers to an appropriate regional CRS that minimizes distortions and allows us to work in units of distance such as meters instead of degrees. Go to Processing ▸ Toolbox.



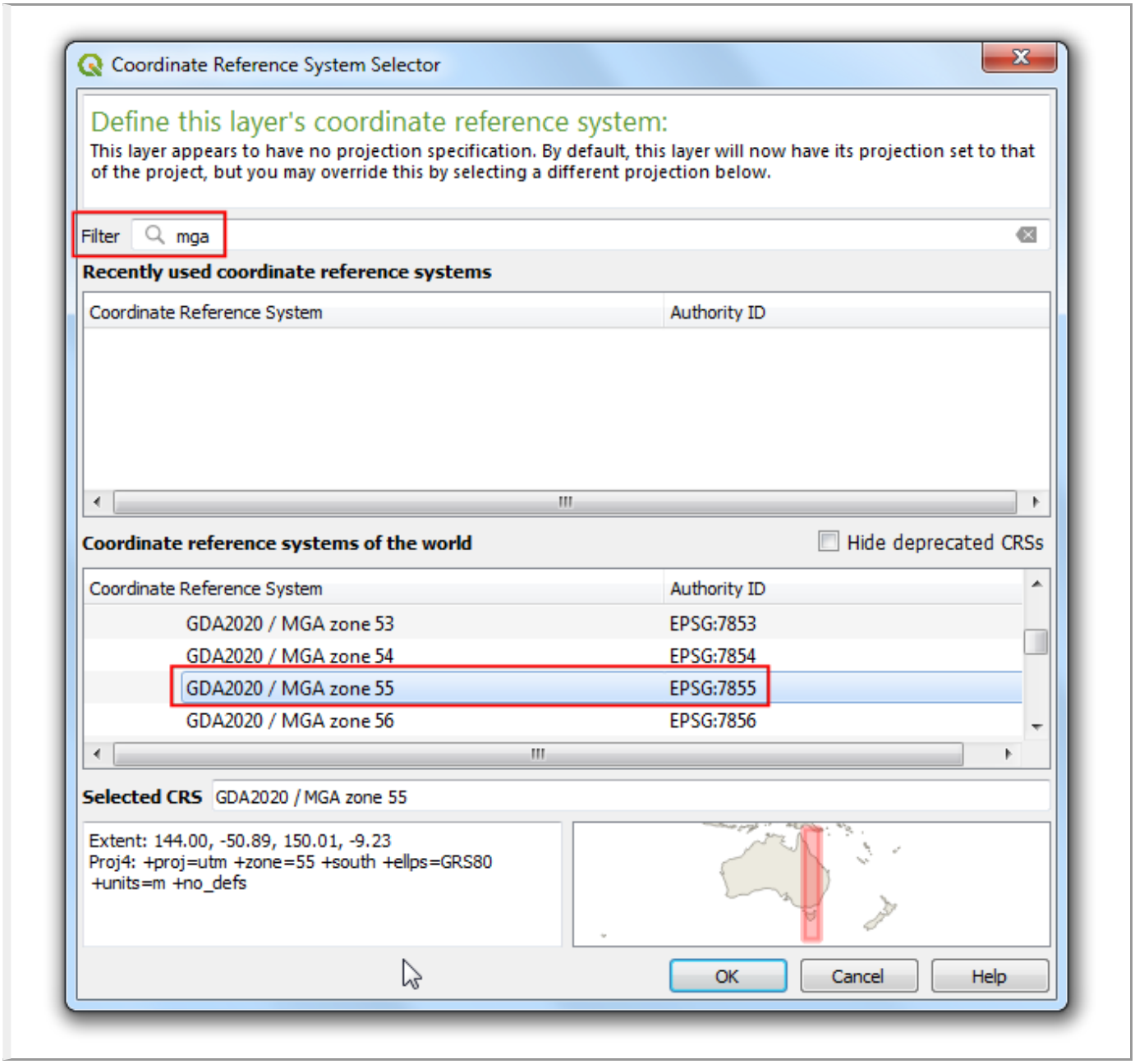
5. Search and locate the Vector general ‣ Reproject layer tool. Double-click to launch it.



6. Select `Bars_and_pubs__with_patron_capacity` as the Input layer. Click the Select CRS button next to Target CRS.



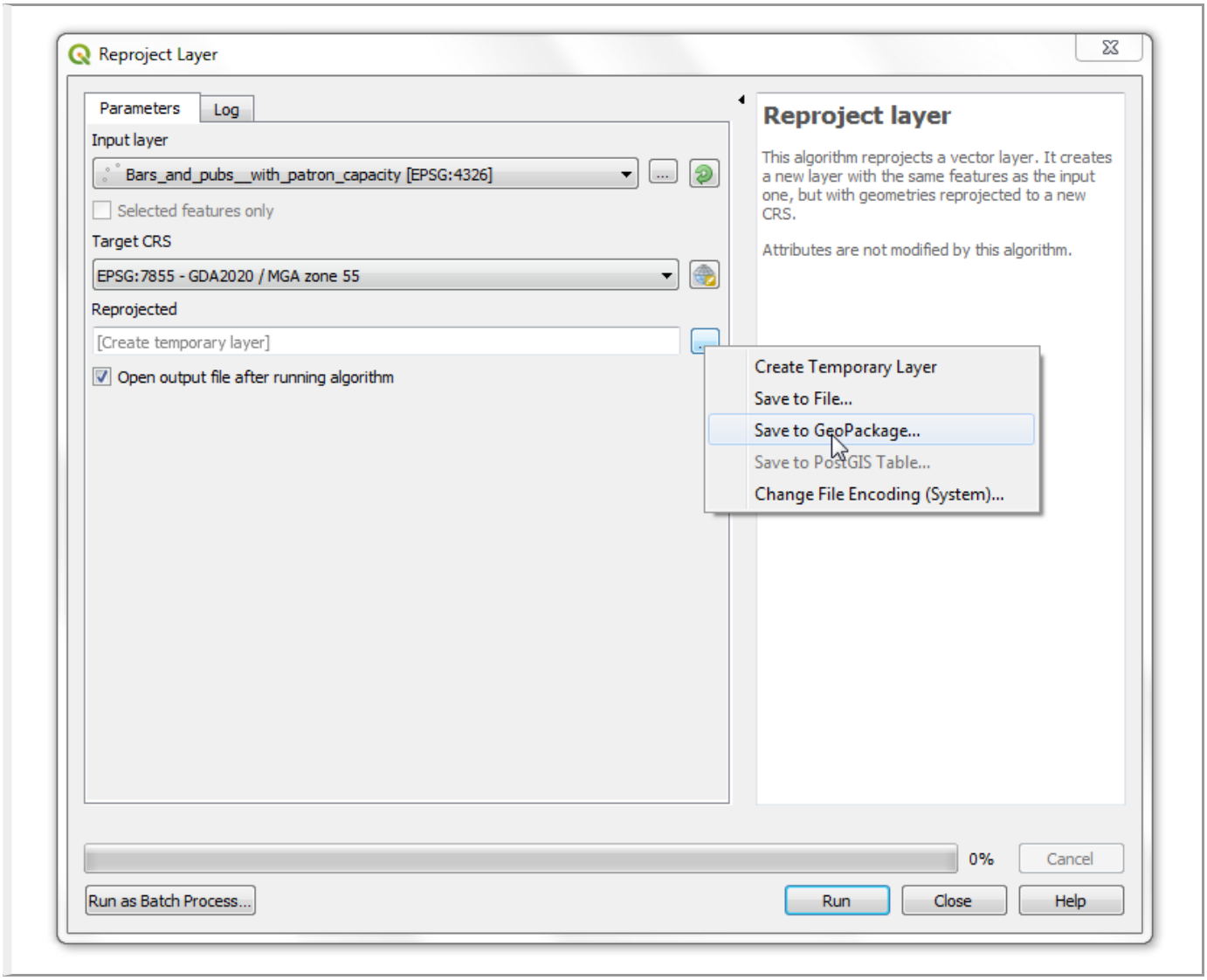
7. When selecting a projected coordinate system for your analysis, the first place to look is for a regional CRS for the area of interest. For Australia, the Map Grid of Australia (MGA) 2020 (https://www.ga.gov.au/scientific-topics/positioning-navigation/geodesy/datums-projections/grid2020) is a UTM-based grid system that is used for local and regional mapping. Melbourne falls in the UTM Zone 55, so we can select the GDA 2020 / MGA zone 55 EPSG:7855` CRS.



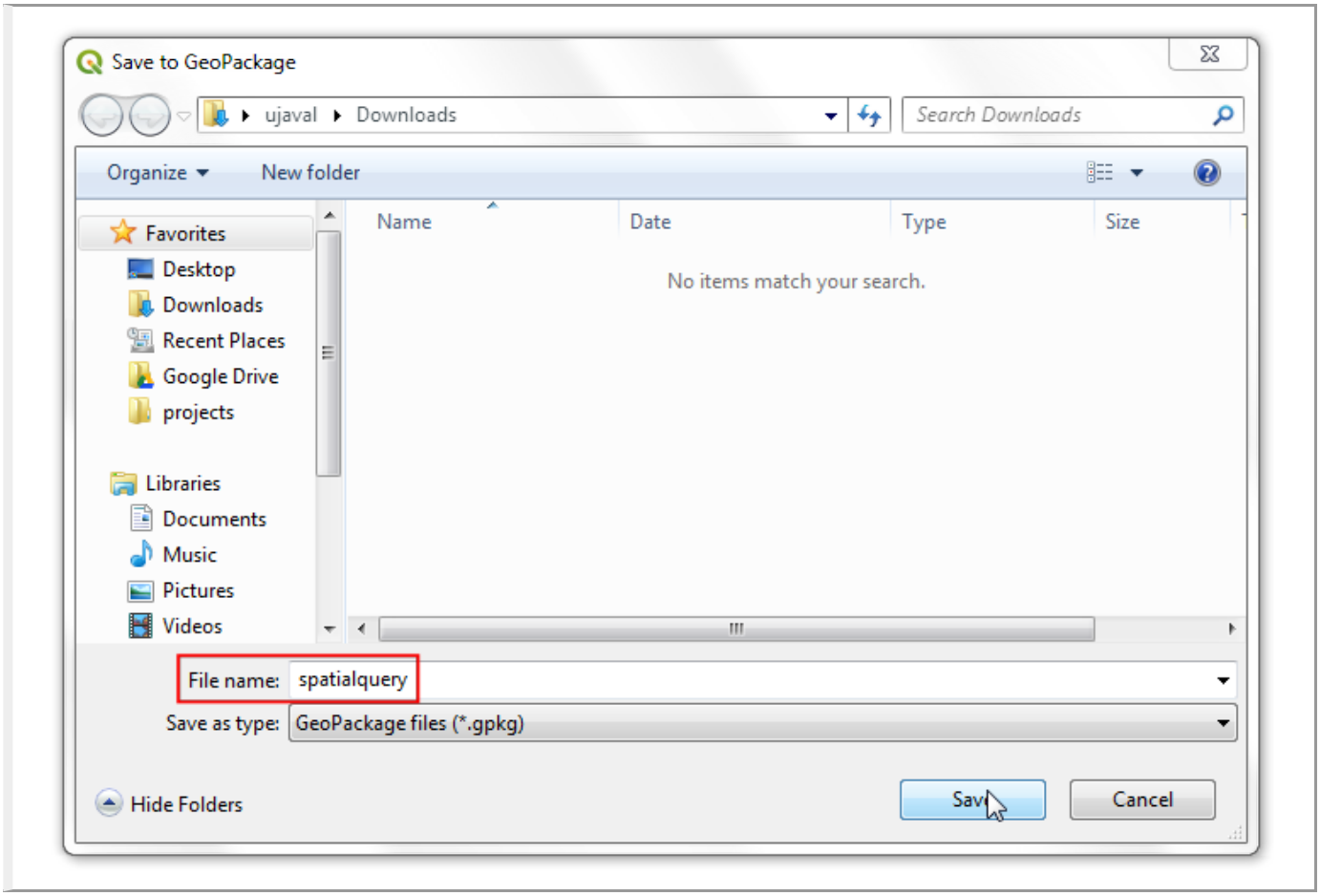
Note

If you are not sure of the a local CRS for the region that you are working in, selecting a CRS for the UTM zone based on the WGS84 datum is a safe choice. You can find out the UTM zone number of your region using UTM Grid Zones of the World (http://www.dmap.co.uk/utmworld.htm).

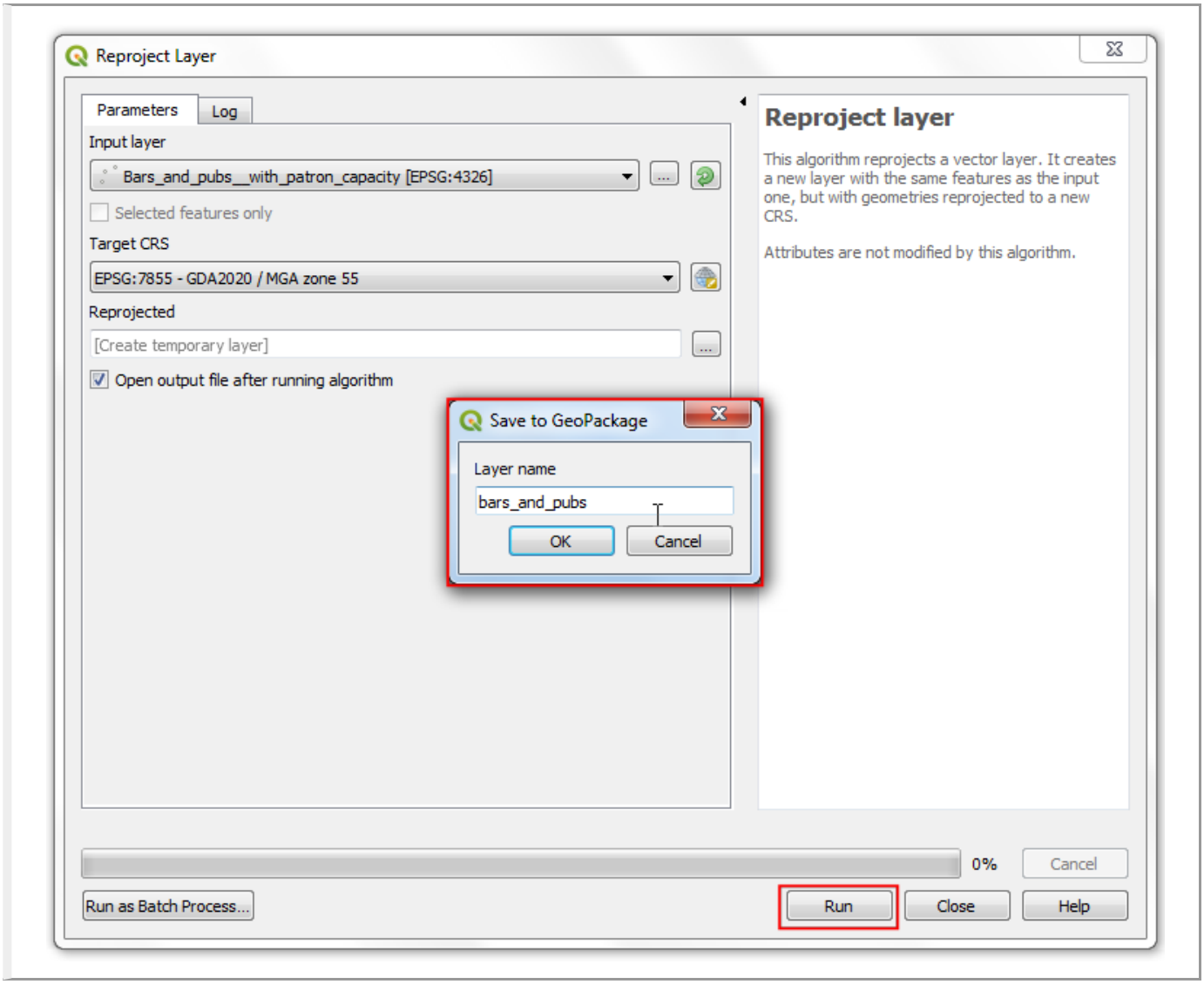
8. Next, click the ... button next to Reprojected and select `Save to GeoPackage`. Geopackage (https://www.geopackage.org/) is the recommended open data format spatial data and is the default data exchange format for QGIS3. A single GeoPackage `.gpkg` file can contain multiple vector and raster layers.



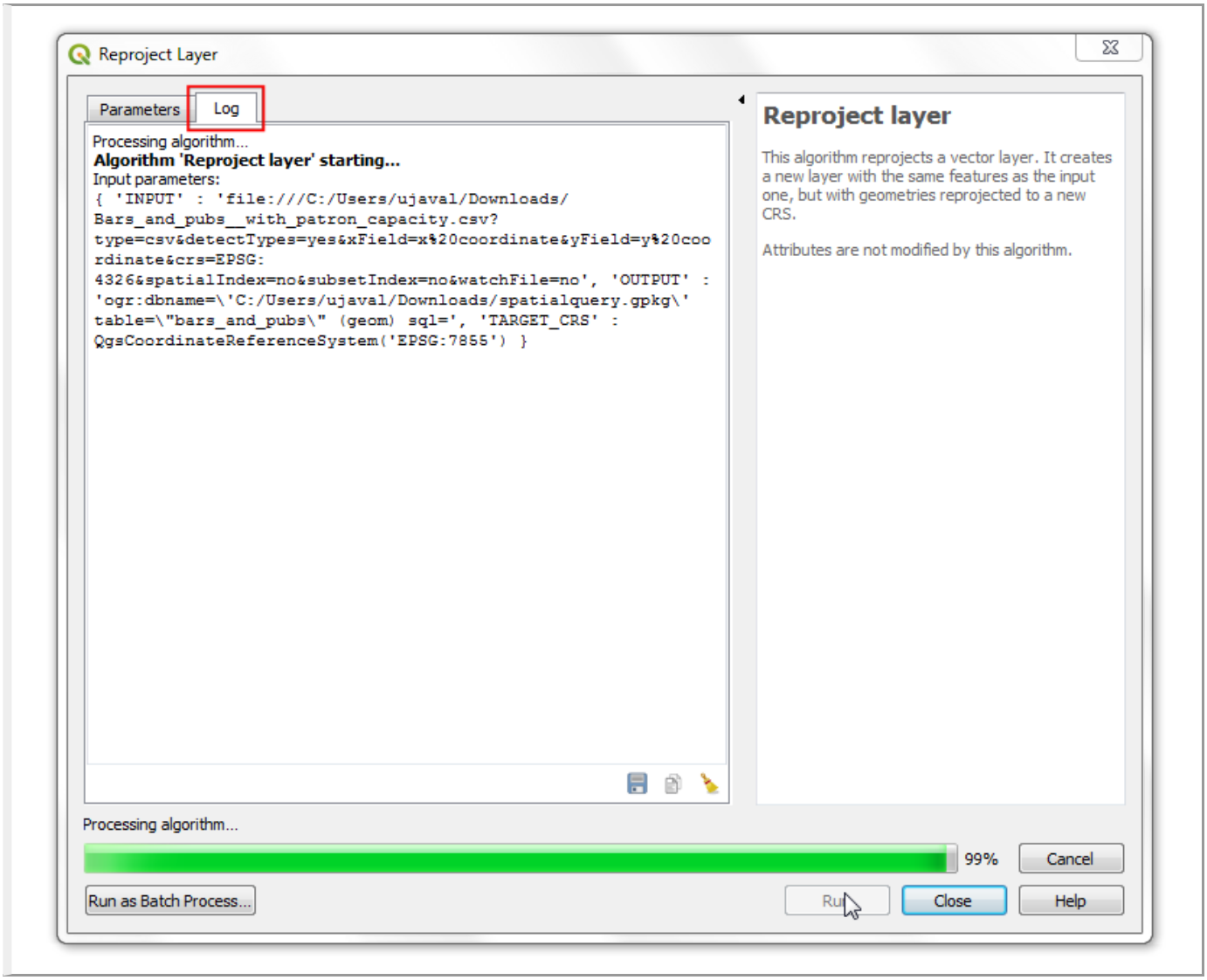
9. Name the geopackage as `spatialquery` and click Save.



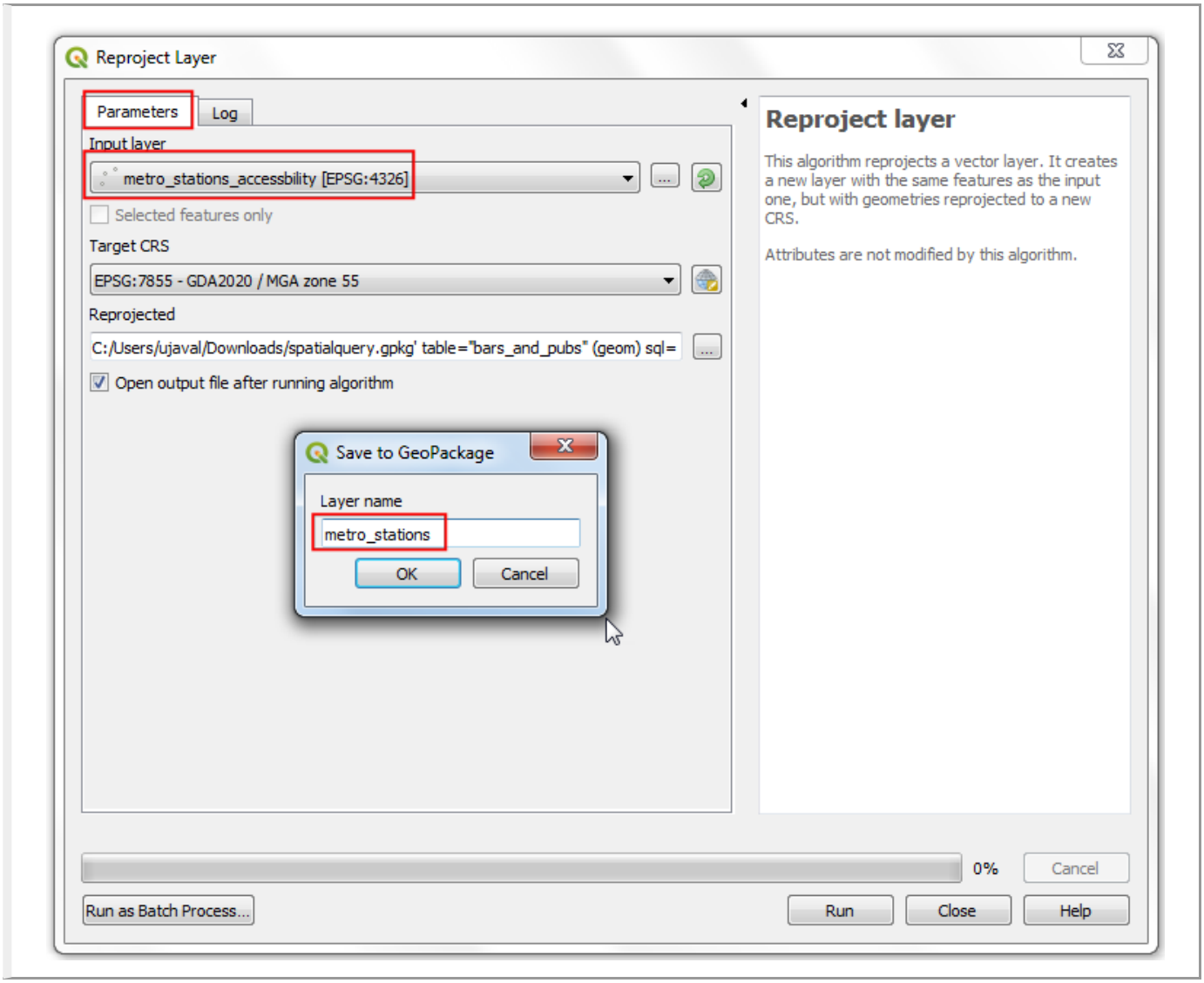
10. When prompted for a layer name, enter `bars_and_pubs` and click OK. Click Run to reproject the layer.



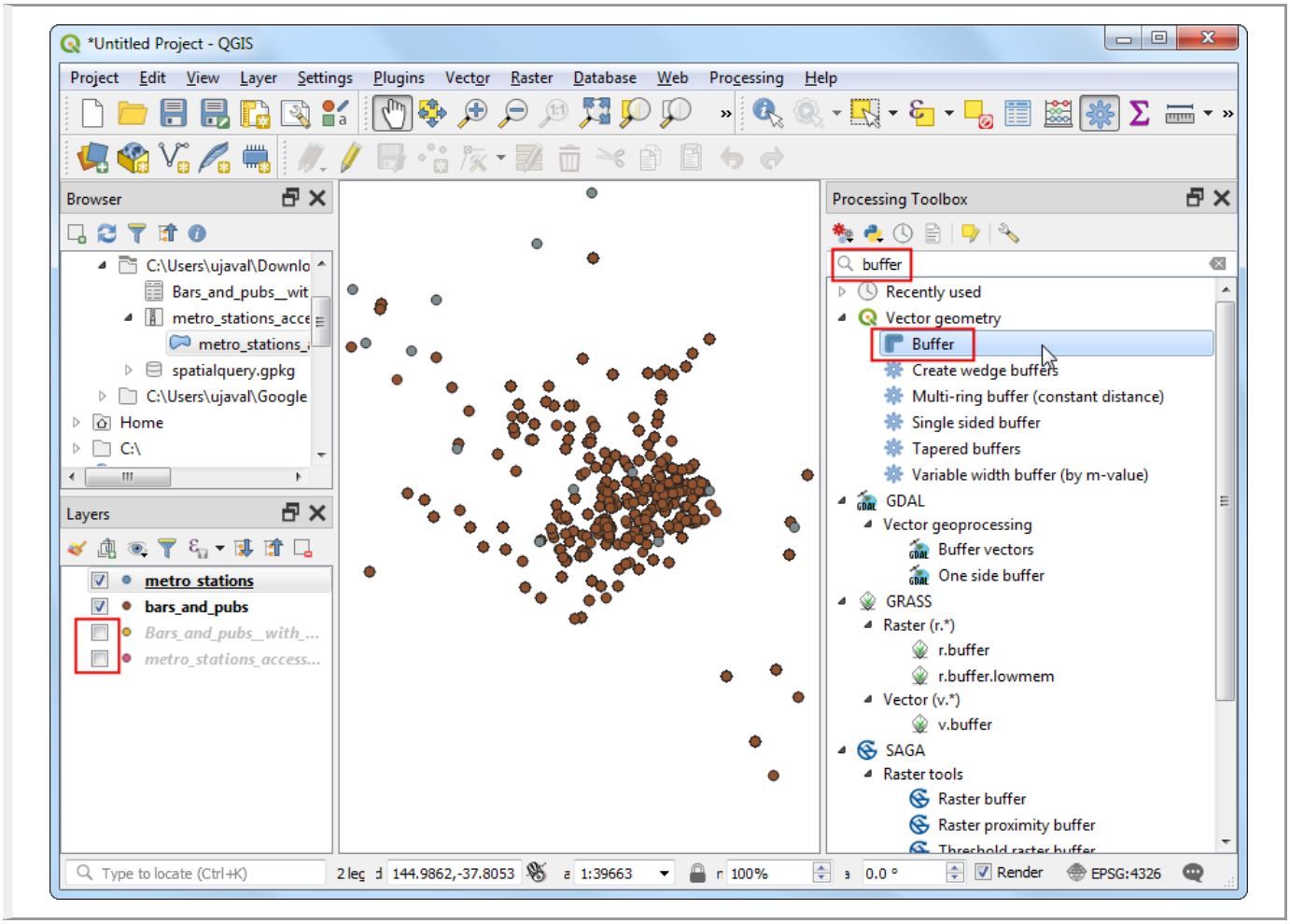
11. The window will switch to the Log tab and you will see the algorithm run and create the new output layer `bars_and_pubs`.



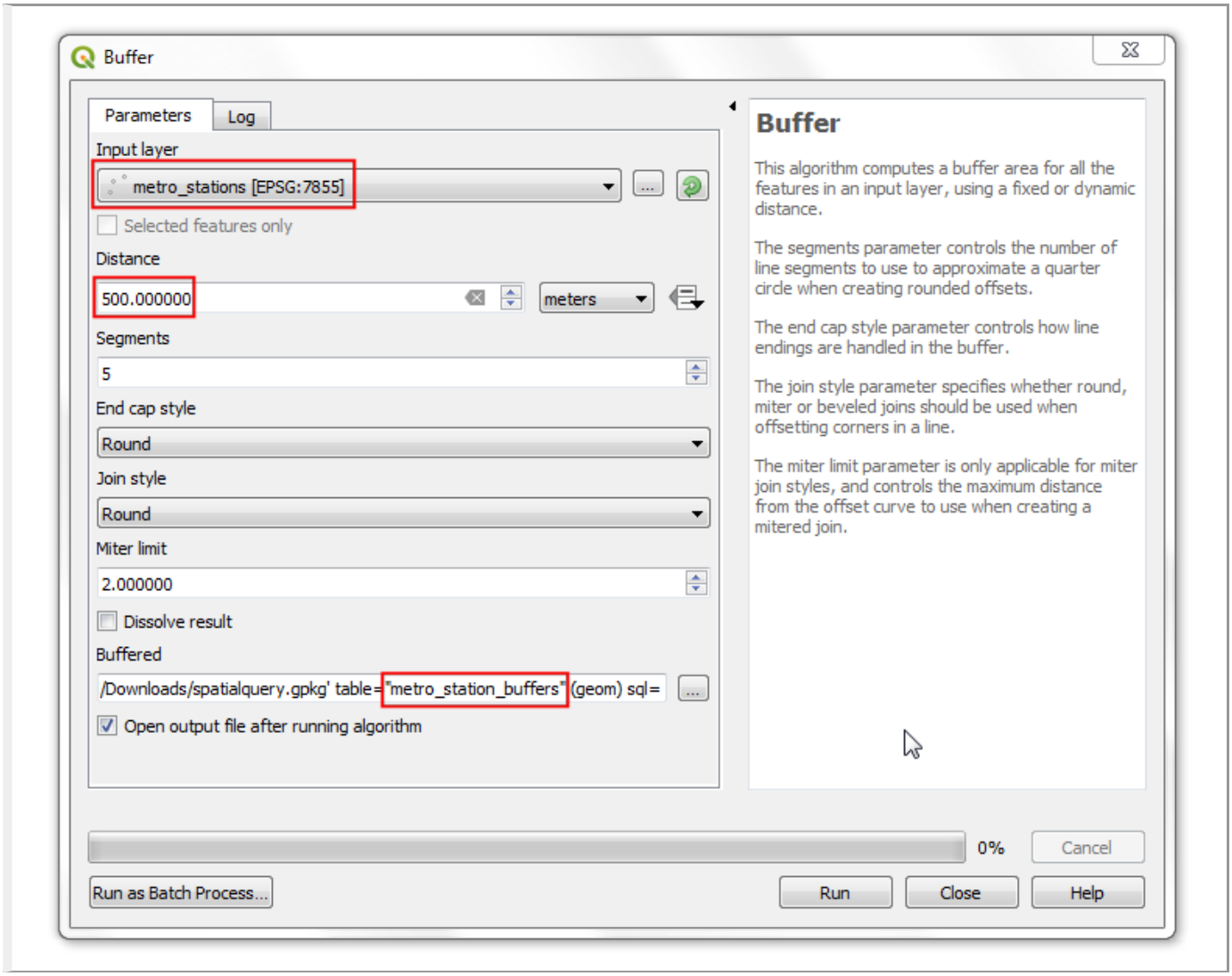
12. Now we will reproject the `metro_stations_accessbility` layer. Switch back to the Paramters tab in the Reproject layer window. Select `metro_stations_accessbility` as the Input layer. Keep the same Target CRS. Next, click the ... button next to Reprojected and select `Save to GeoPackage` . Select the same output file `spatialquery` (Remember that a single geopackage file can contain multiple layers, so we will save the new layer to the same geopackage file). Enter `metro_stations` as the Layer name. Click Run.



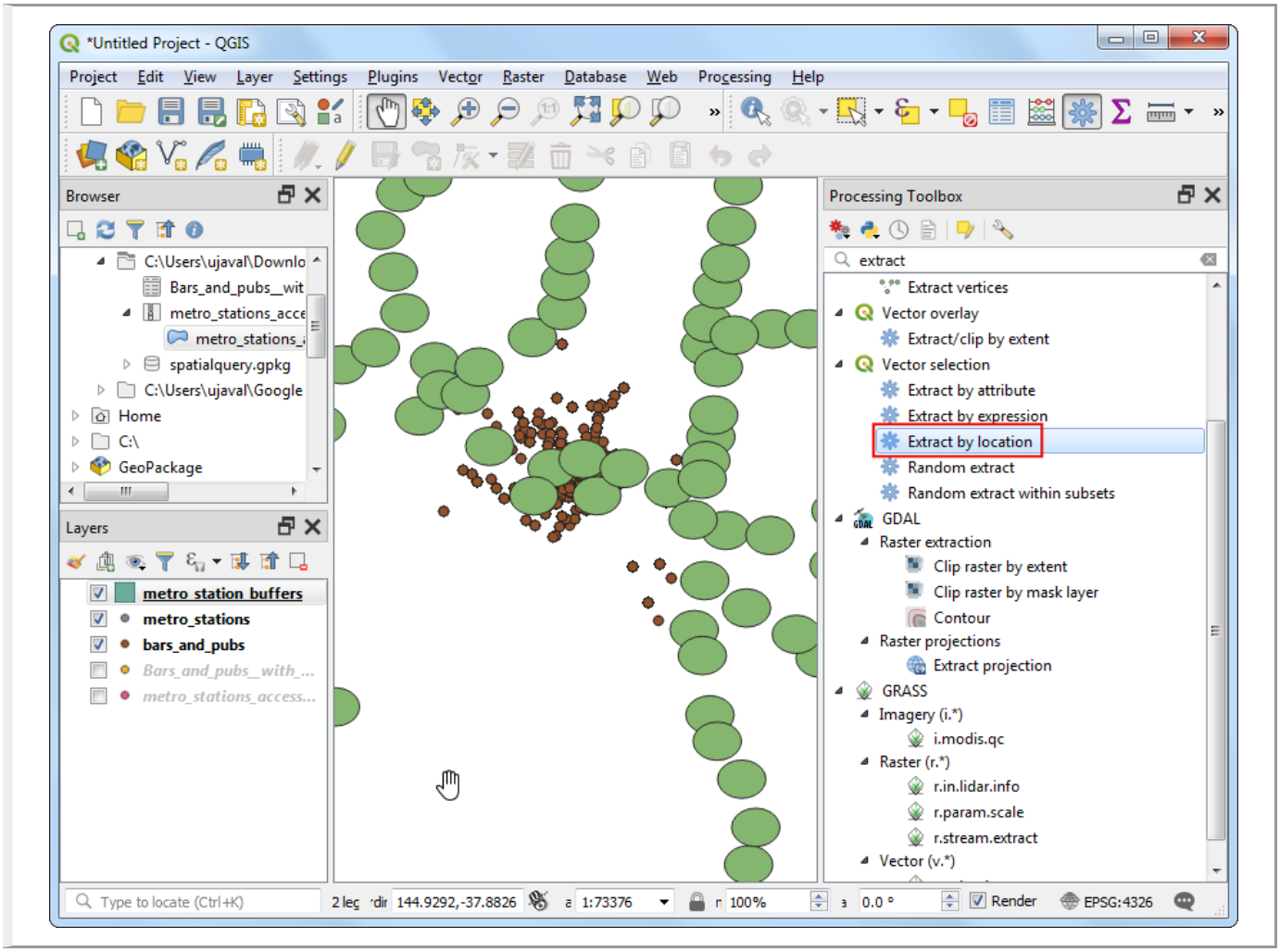
13. Back in the main QGIS window, you will see 2 new layers loaded in the Layers panel: `bars_and_pubs` and `metro_stations`. You may turn off the visibility for original layers. Now, we are ready to do the spatial query. As we are interested in selecting bars and pubs within 500m of the metro stations, the first step is to create a buffer around the metro stations that represents our search area. Search and locate the Vector geometry ▸ Buffer tool in the Processing Toolbox and double-click to launch it.



14. In the Buufer dialog, select `metro_stations` as the Input layer. Set `500` meters as the Distance. Save the output to the same `spatialquery` geopackage and enter `metro_stations_buffers` as the Layer name. Click Run.



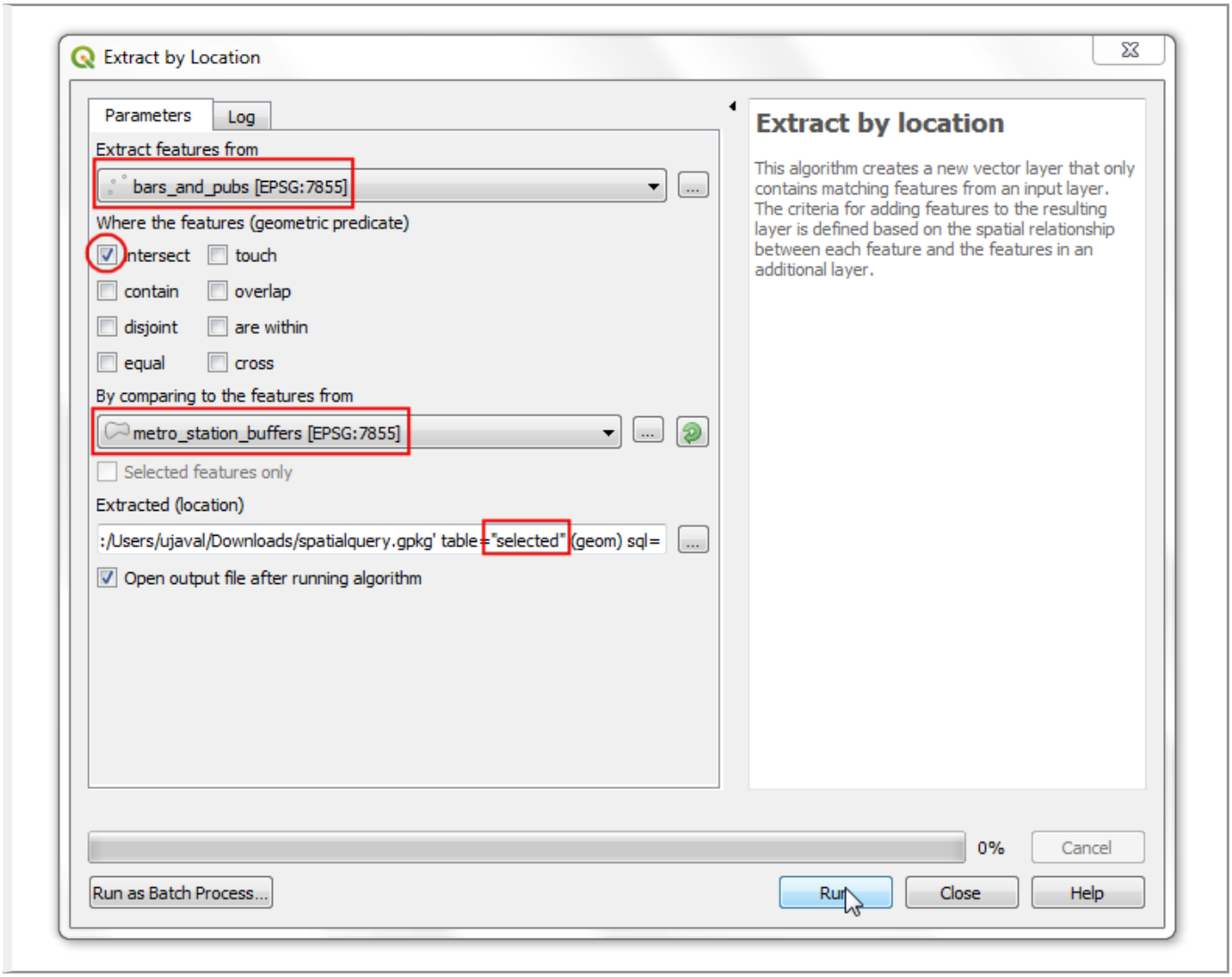
15. You will see a new `metro_stations_buffers` layers loaded in the Layers panel. Now we can find out which points from the `bars_and_pubs` layer falls within the polygons from the `metro_stations_buffers` layer. Locate the Vector selection ▸ Extract by Location tool from the Processing Toolbox and double-click to launch it.



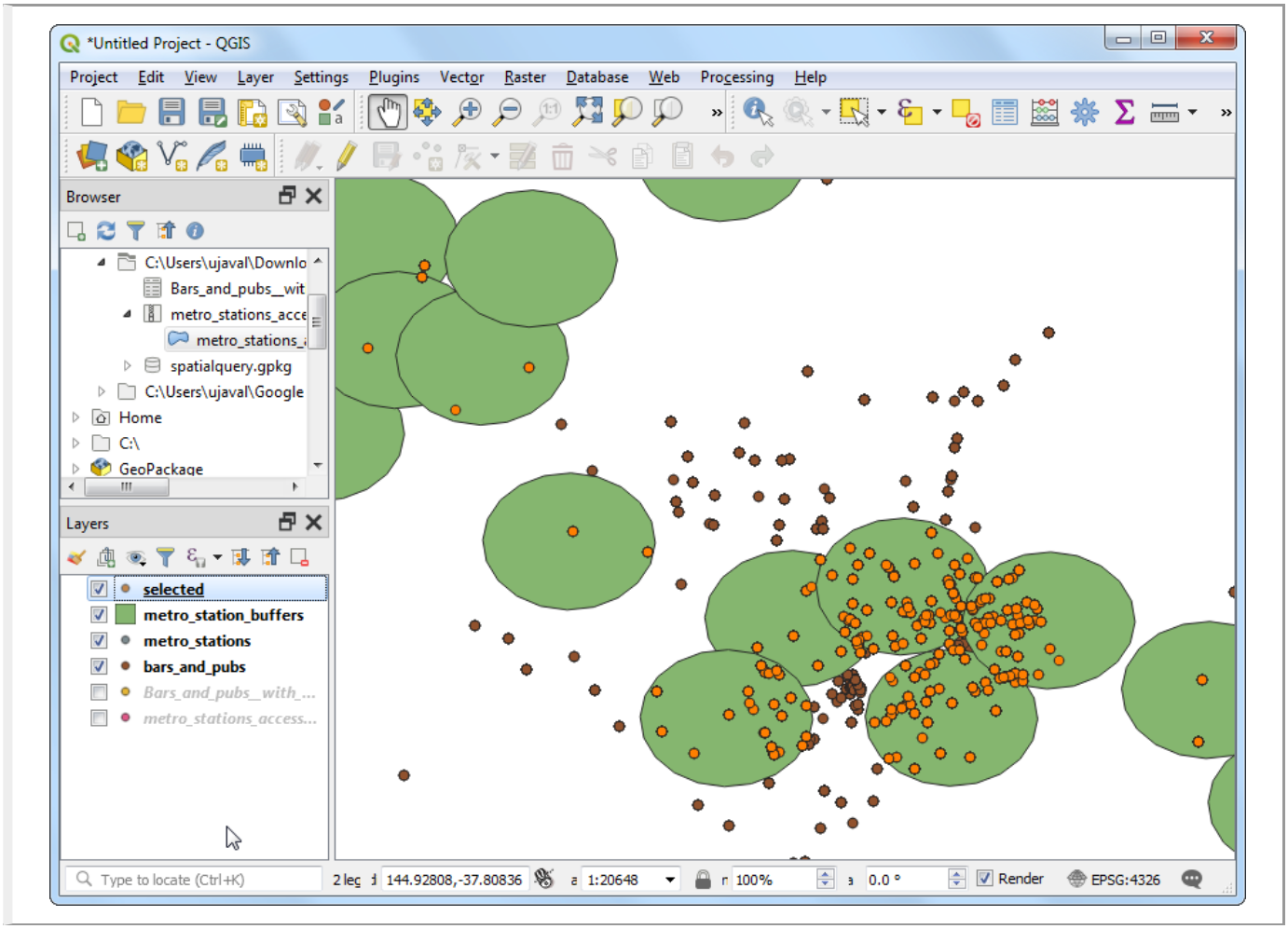
Note

Extract by location will create a new layer with the matching features from the spatial query. If you just want to select features, use the *Select by location* tool.

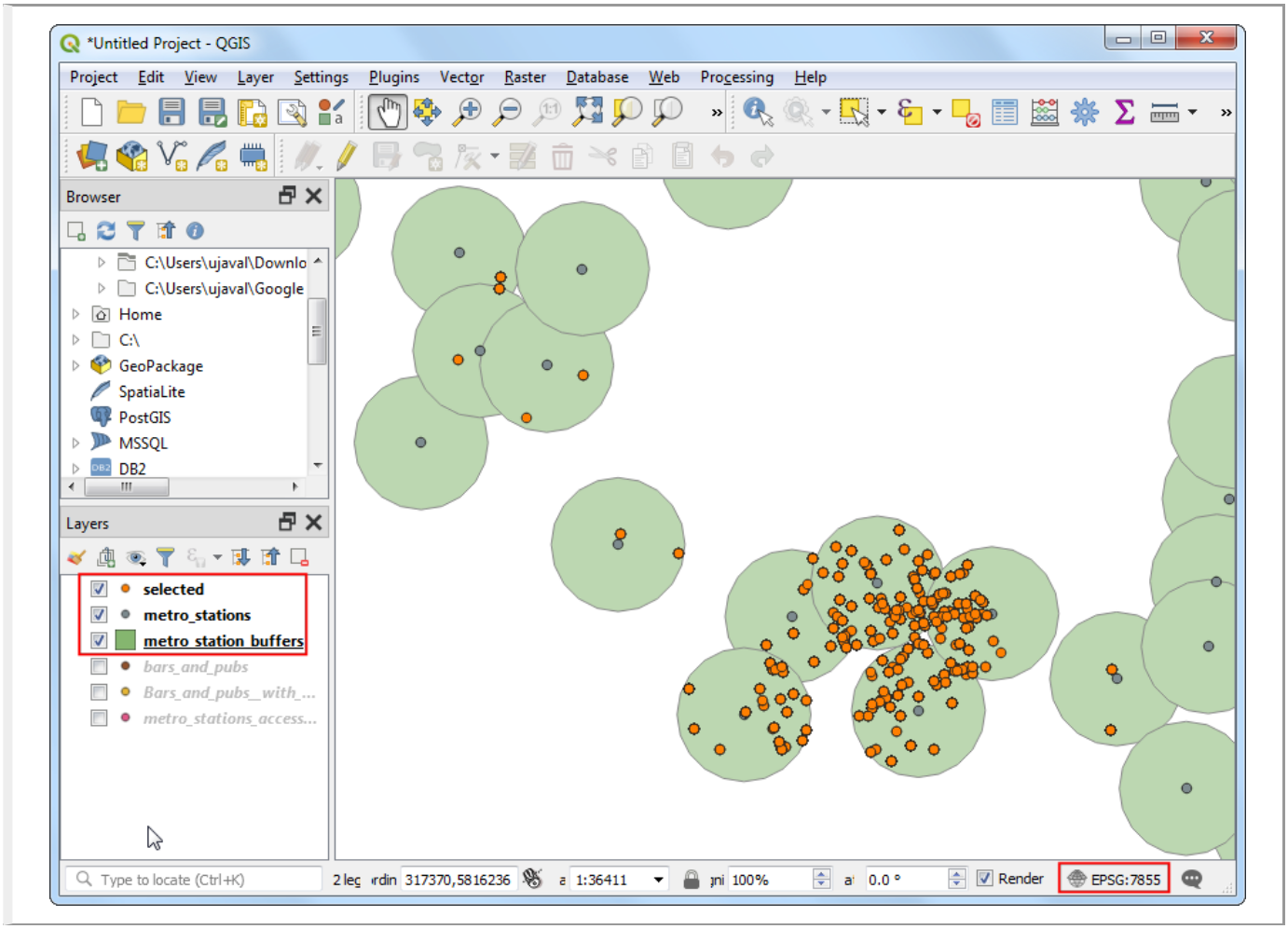
16. In the Extract by location dialog, select `bars_and_pubs` as the Extract features from. Check `Intersect` as the geometry predicate. Set `metro_stations_buffers` as By comparing to the features from. Save the output to the `spatialquery` geopackage as the layer `selected` . Click Run.



17. Once the processing finishes, you will see the `selected` layers added to the Layers panel. Note that this layer only contains points from the `bars_and_pubs` that fall within the buffer polygons.



18. Our analysis is complete. You may notice that the buffer polygons look oval-shaped. This is because our Project CRS is still set to **EPSG:4326 WGS84**. To better visualize the results, you can go to Project ▸ Properties ▸ CRS and select `GDA 2020 / MGA zone 55 EPSG:7855` which we used for the analysis. Once set to this CRS, the buffer will appear in the correct shape.



Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Nearest Neighbor Analysis (QGIS3)

GIS is very useful in analyzing spatial relationship between features. One such analysis is finding out which features are closest to a given feature. There are multiple ways to do this analysis in QGIS. In this tutorial,wWe will explore the **Distance to nearest hub** and **Distance matrix** tools to carry out the nearest neighbor analysis.

Overview of the task

Given the locations of all known significant earthquakes, find out the nearest populated place for each location where the earthquake happened.

Other skills you will learn

- Use the *Geometry Generator* renderer to dynamically create lines from a multipoint layer.

Get the data

We will use NOAA's National Geophysical Data Center's Significant Earthquake Database (http://www.ngdc.noaa.gov/nndc/struts/form?t=101650&s=1&d=1) as our layer representing all major earthquakes. Download the tab-delimited earthquake data (https://www.ngdc.noaa.gov/nndc/struts/results?type_0=Exact&query_0=$ID&t=101650&s=13&d=189&dfn=signif.txt).

Natural Earth has a nice Populated Places (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-populated-places/) dataset. Download the simple (less columns) dataset (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_populated_places_simple.zip)

For convenience, you may directly download a copy of both the datasets from the links below:

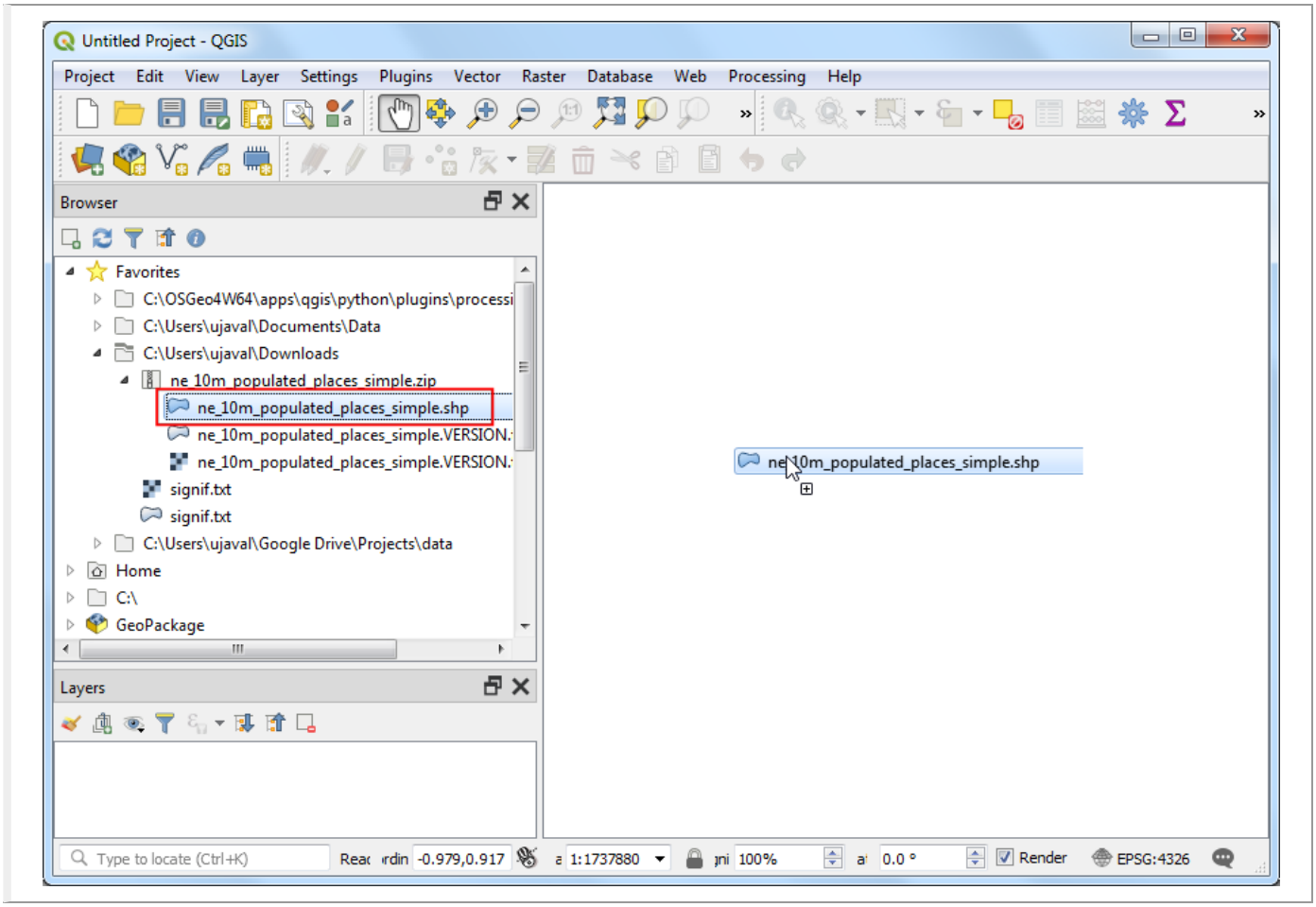
signif.txt (http://www.qgistutorials.com/downloads/signif.txt)

ne_10m_populated_places_simple.zip (http://www.qgistutorials.com/downloads/ne_10m_populated_places_simple.zip)

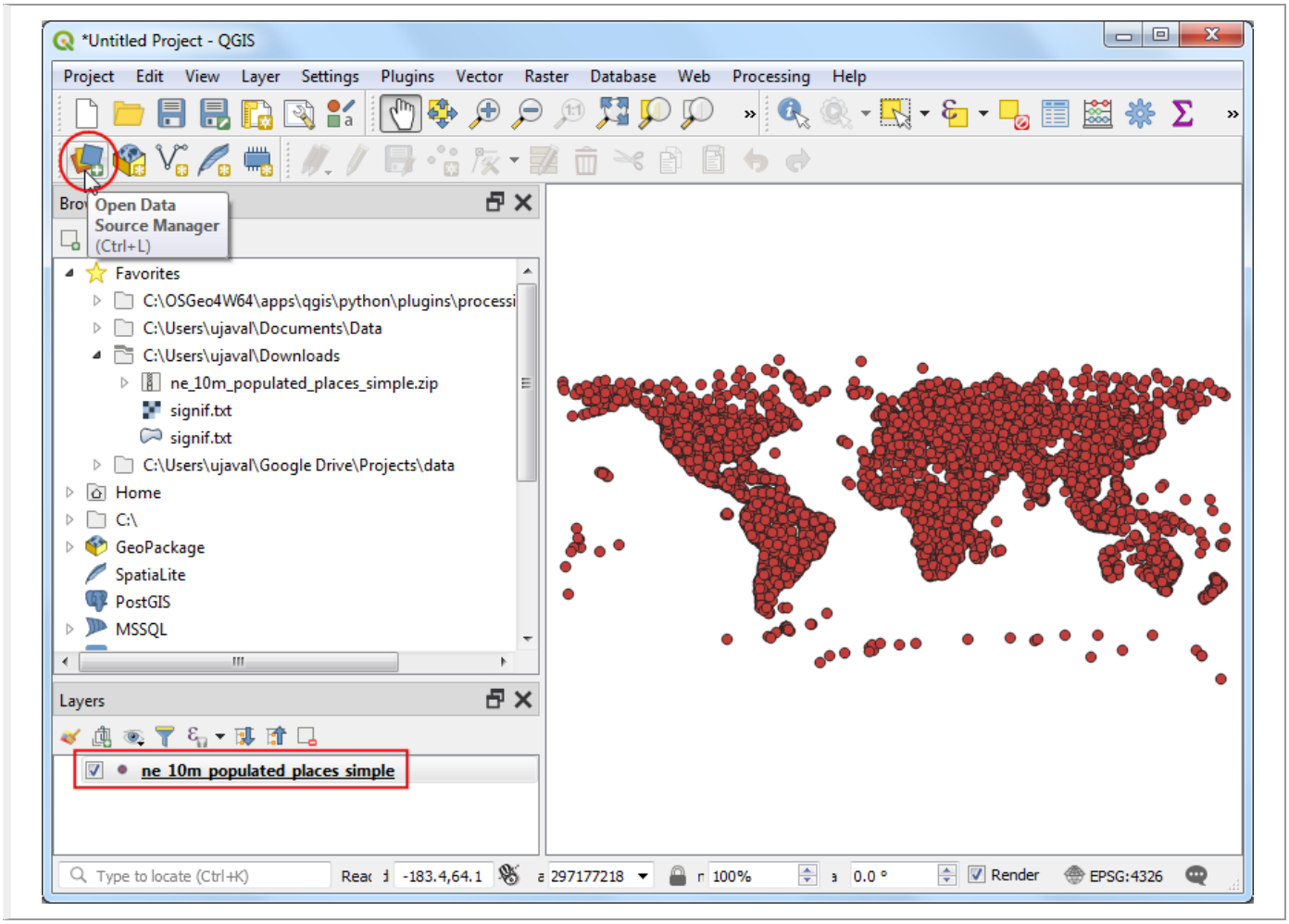
Data Sources: [NGDC] (../credits.html#ngdc) [NATURALEARTH] (../credits.html#naturalearth)

Procedure

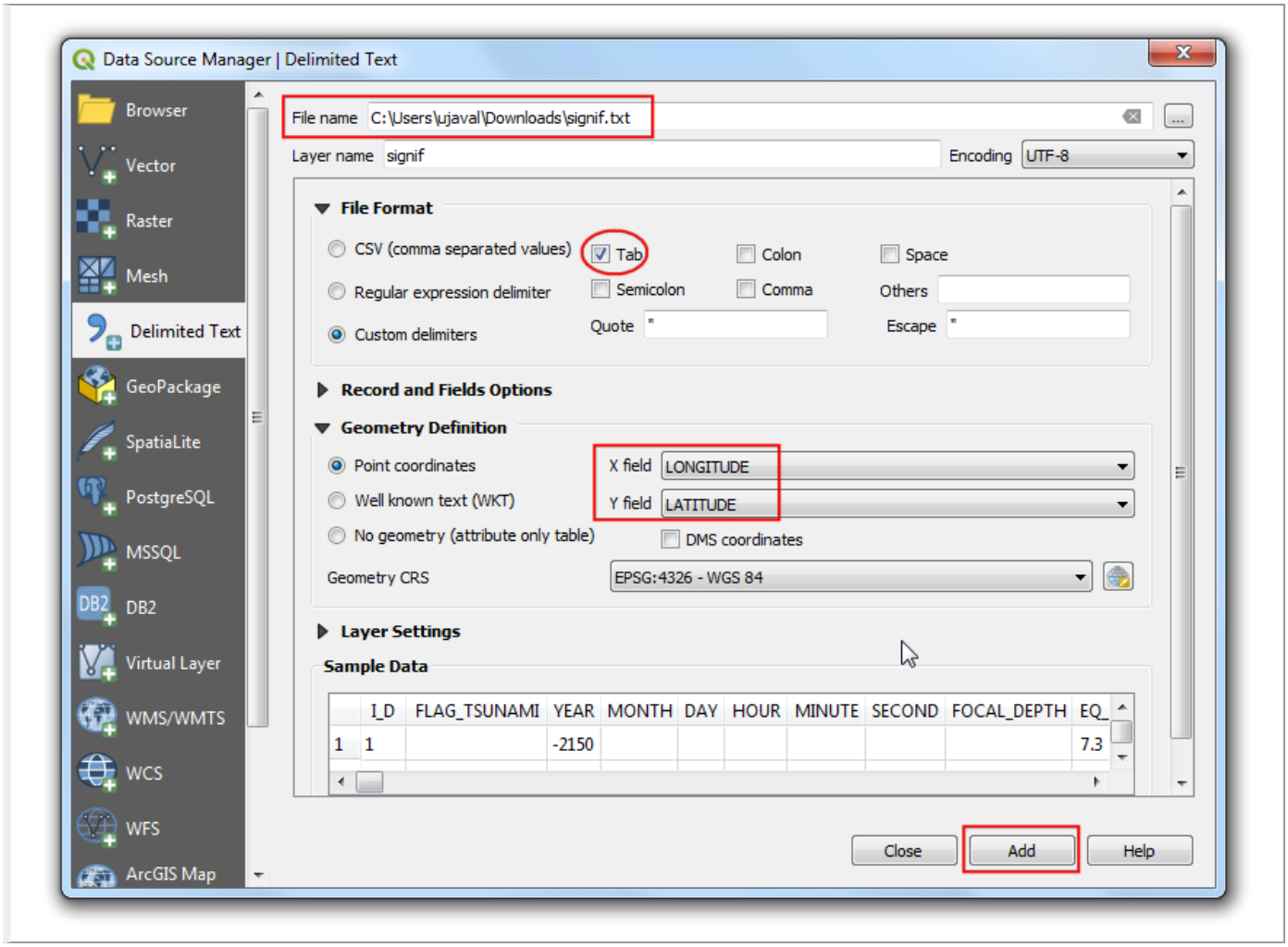
1. Locate the downloaded `ne_10m_populated_places_simple.zip` file in the Browser panel and expand it. Drag the `ne_10m_populated_places_simple.shp` file to the canvas.



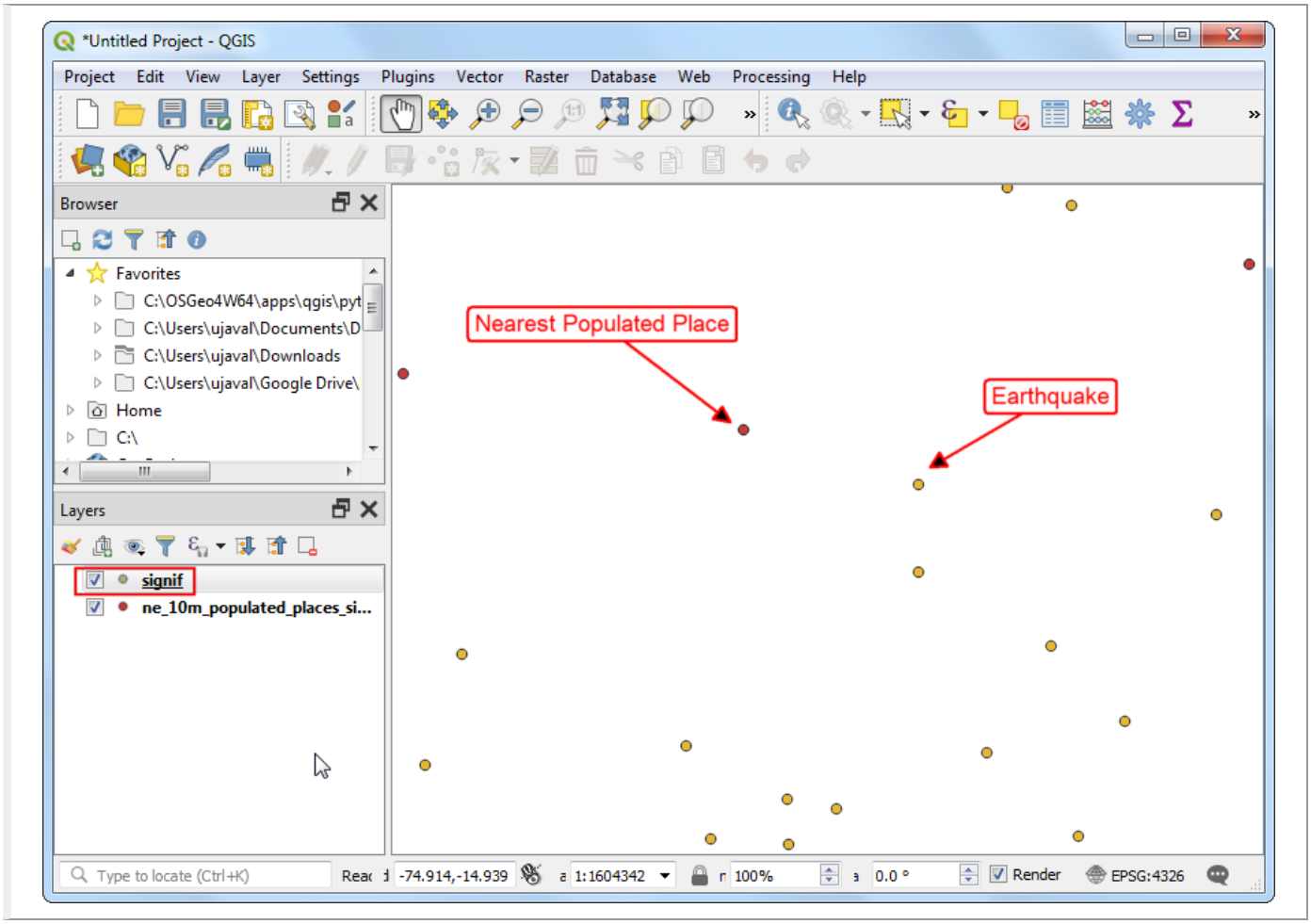
2. You will see a new layer `ne_10m_populated_places_simple` loaded in the Layers panel. This layer contains the points representing populated places. Now we will load the earthquakes layer. This layer comes as a *Tab Serepated Values (TSV)* text file. To load this file, click the Open Data Source Manager button on the Data Source Toolbar. You can also use `Ctrl + L` keyboard shortcut.



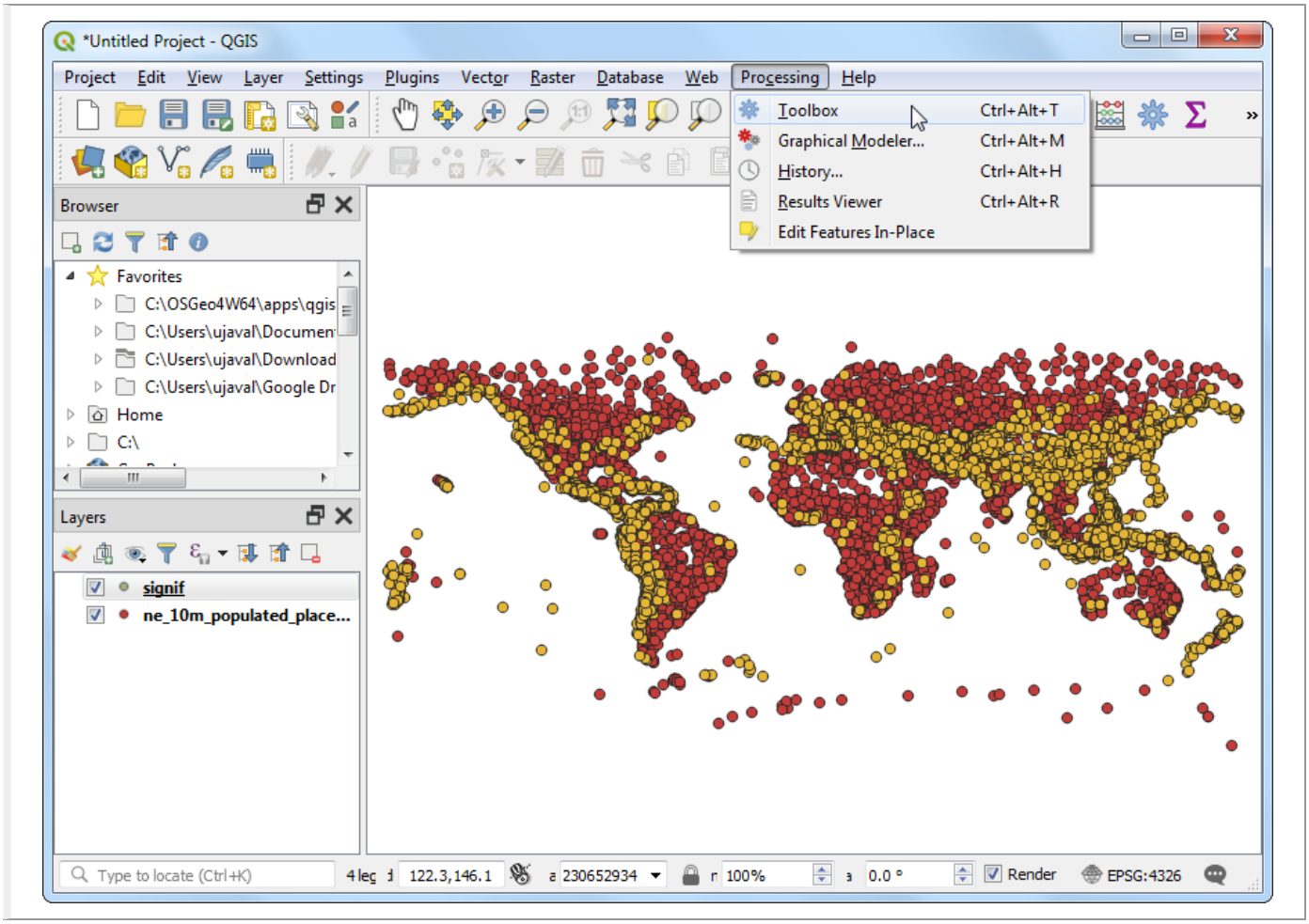
3. Click the ... button next to File name and browse to the downloaded `signif.txt` file. Once loaded, the File Format and Geometry Definition fields should be auto-populated with correct values. Click Add followed by Close.



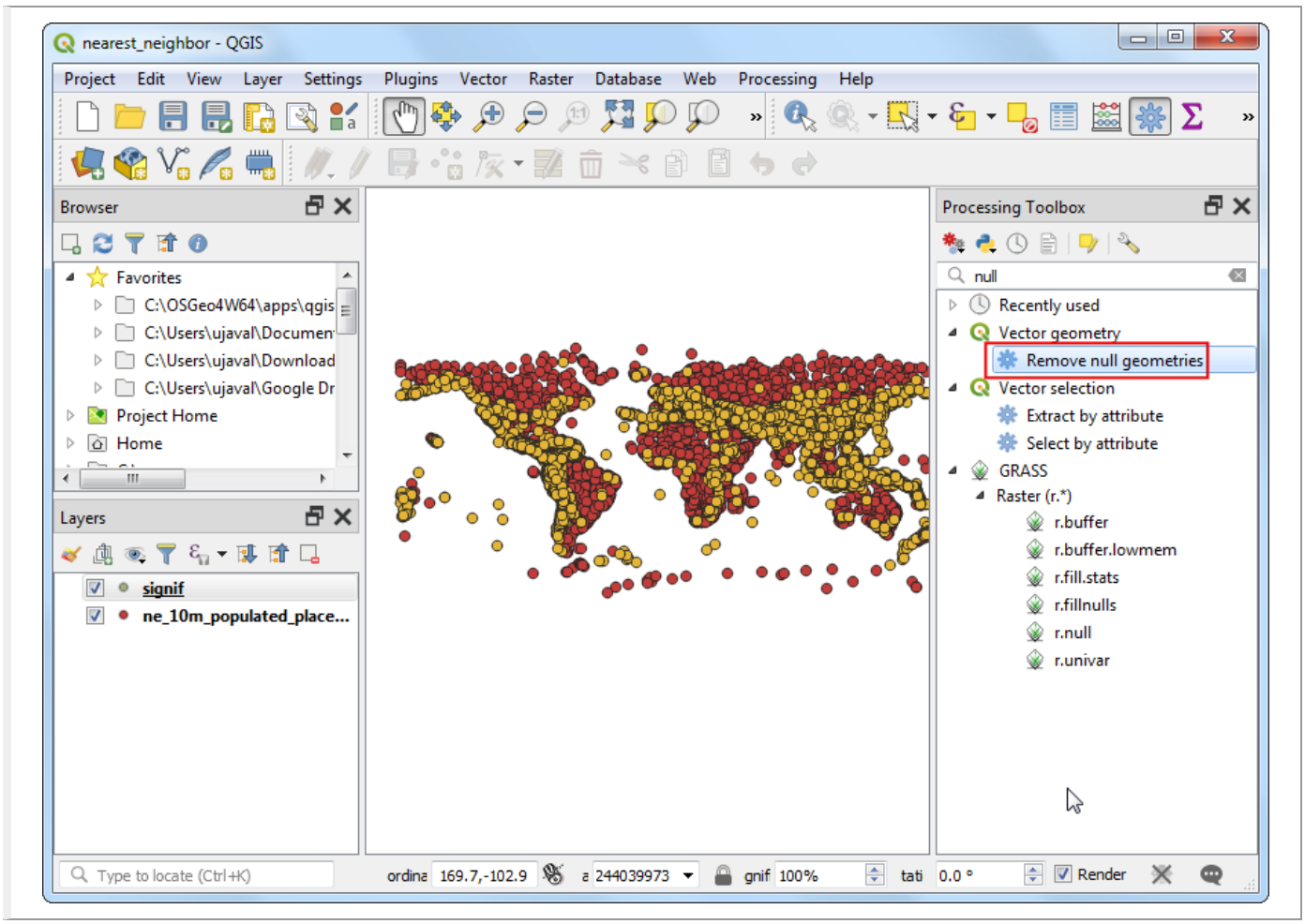
4. Zoom around and explore both the datasets. Each yellow point represents the location of a significant earthquake and each red point represents the location of a populated place. Our goal is to find out the nearest point from the populated places layer for each of the points in the earthquake layer.



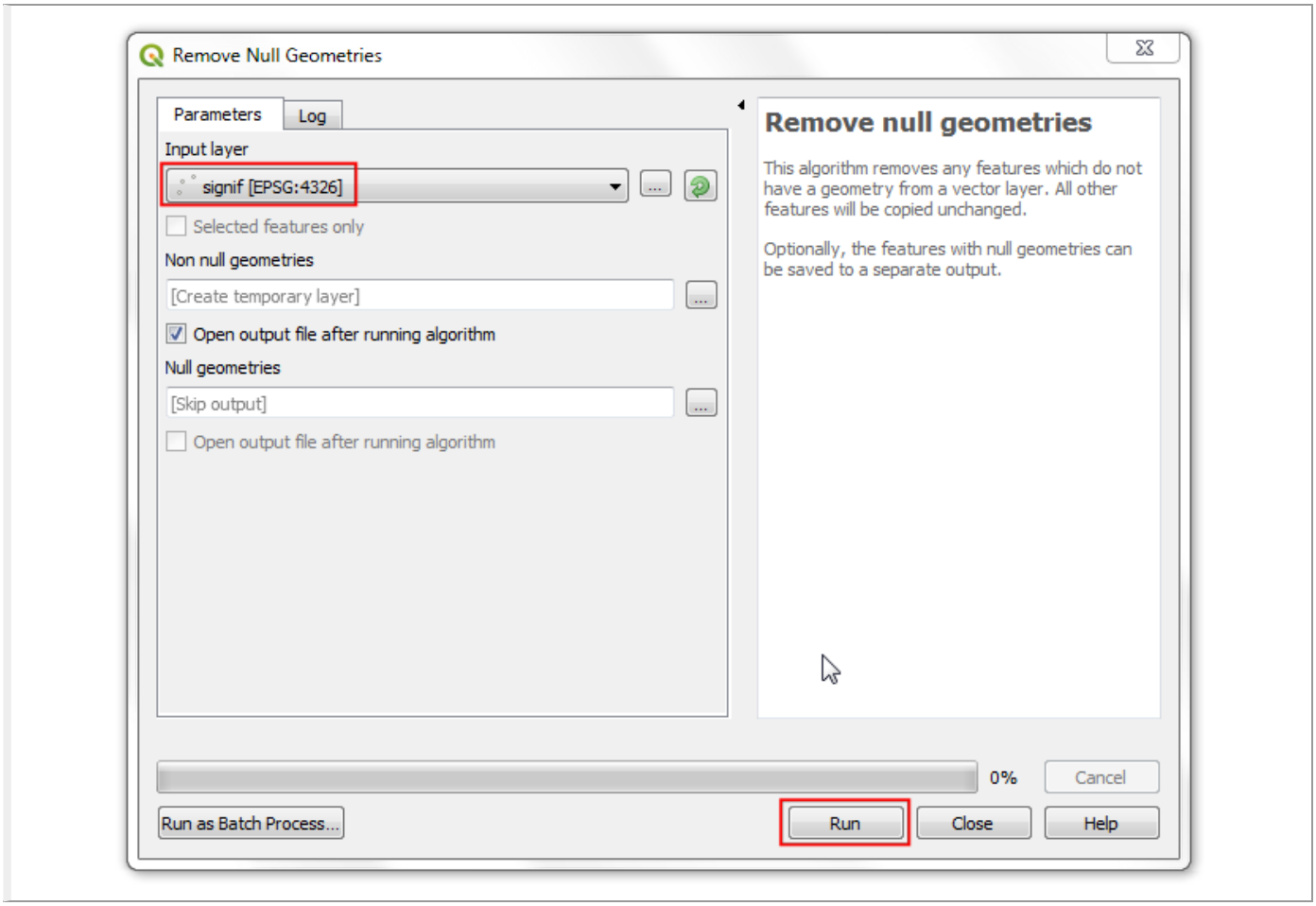
5. Before we do the analysis, we need to clean up our inputs. The `signif` layer contains many records without a valid geometry. These records were imported with a **NULL** geometry. So let's remove these records first. Go to Processing ‣ Toolbox.



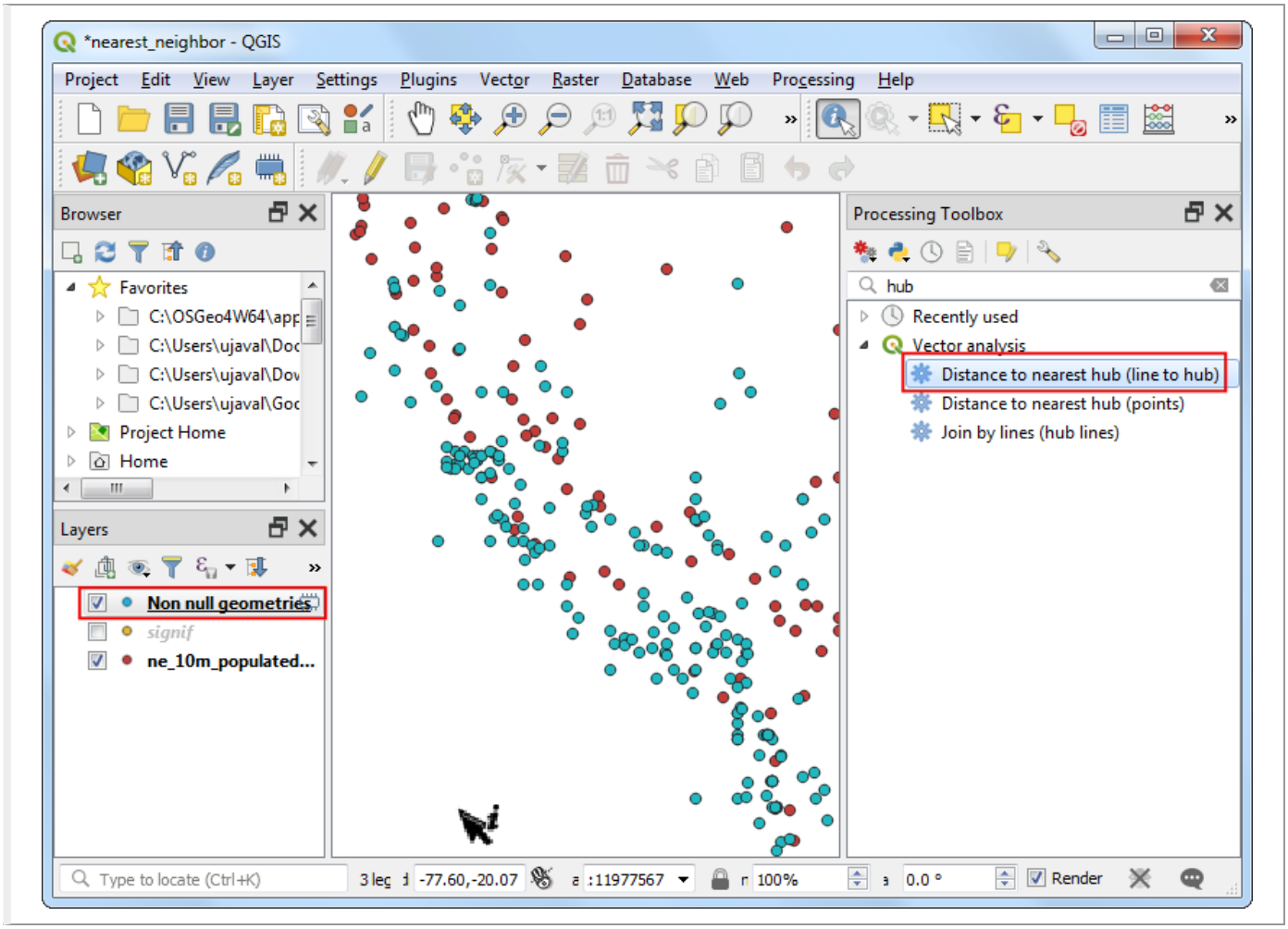
6. Search for and locate the Vector geometry ‣ Remove null geometries tool. Double-click to launch it.



7. Select `signif` as the Input layer and click Run. Once the processing finishes, click Close.



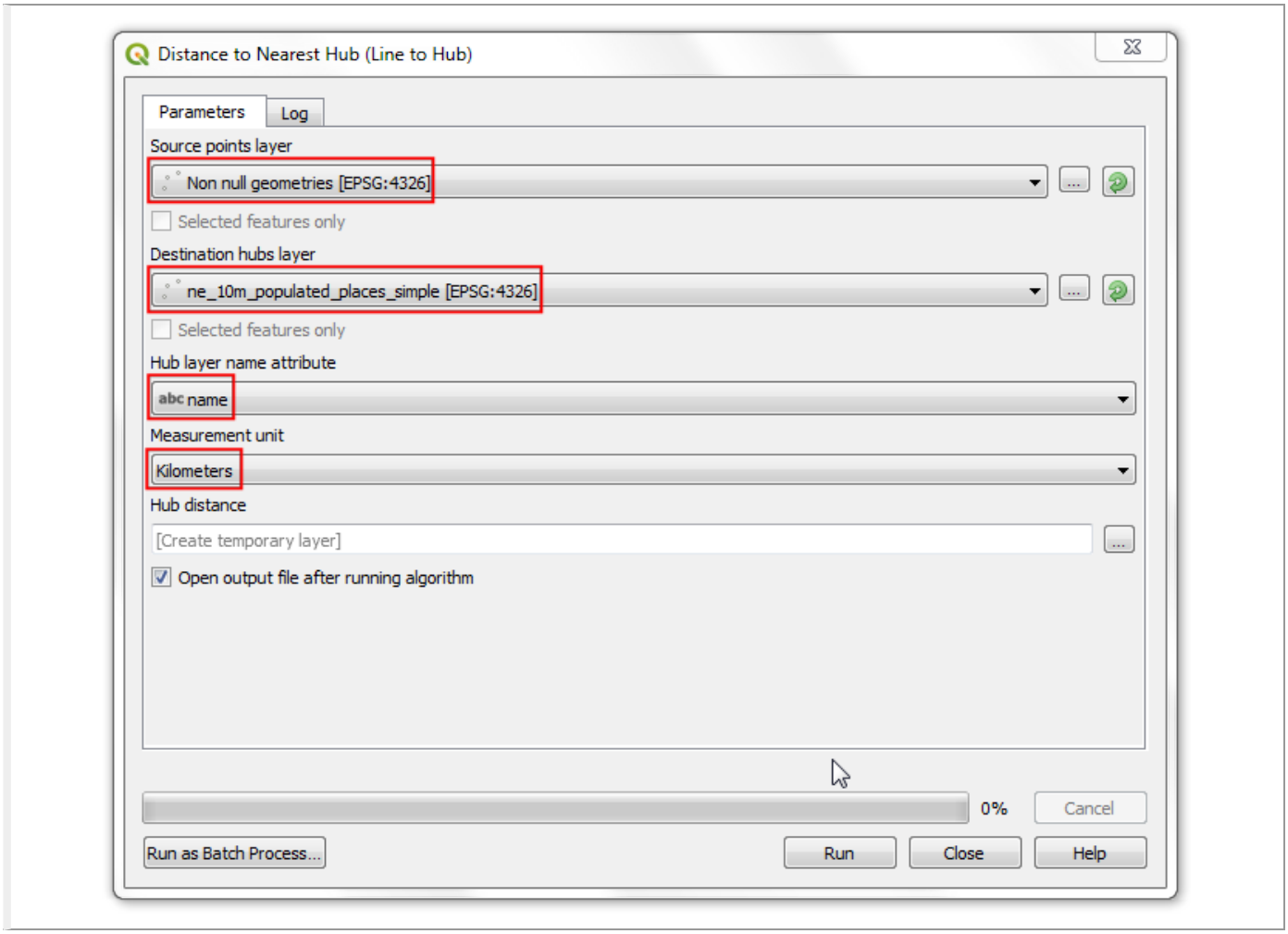
8. You will see a new layer caled `Non null geometries` loaded into the Layers panel. We will use this layer instead of the original `signif` layer in further analysis. Un-check the `signif` layer in the Layers panel to hide it. Now it is time to perform the nearest neighbor analysis. Search and locate the Vector analysis ‣ Distance to nearest hub (line to hub) tool. Double-click to launch it.



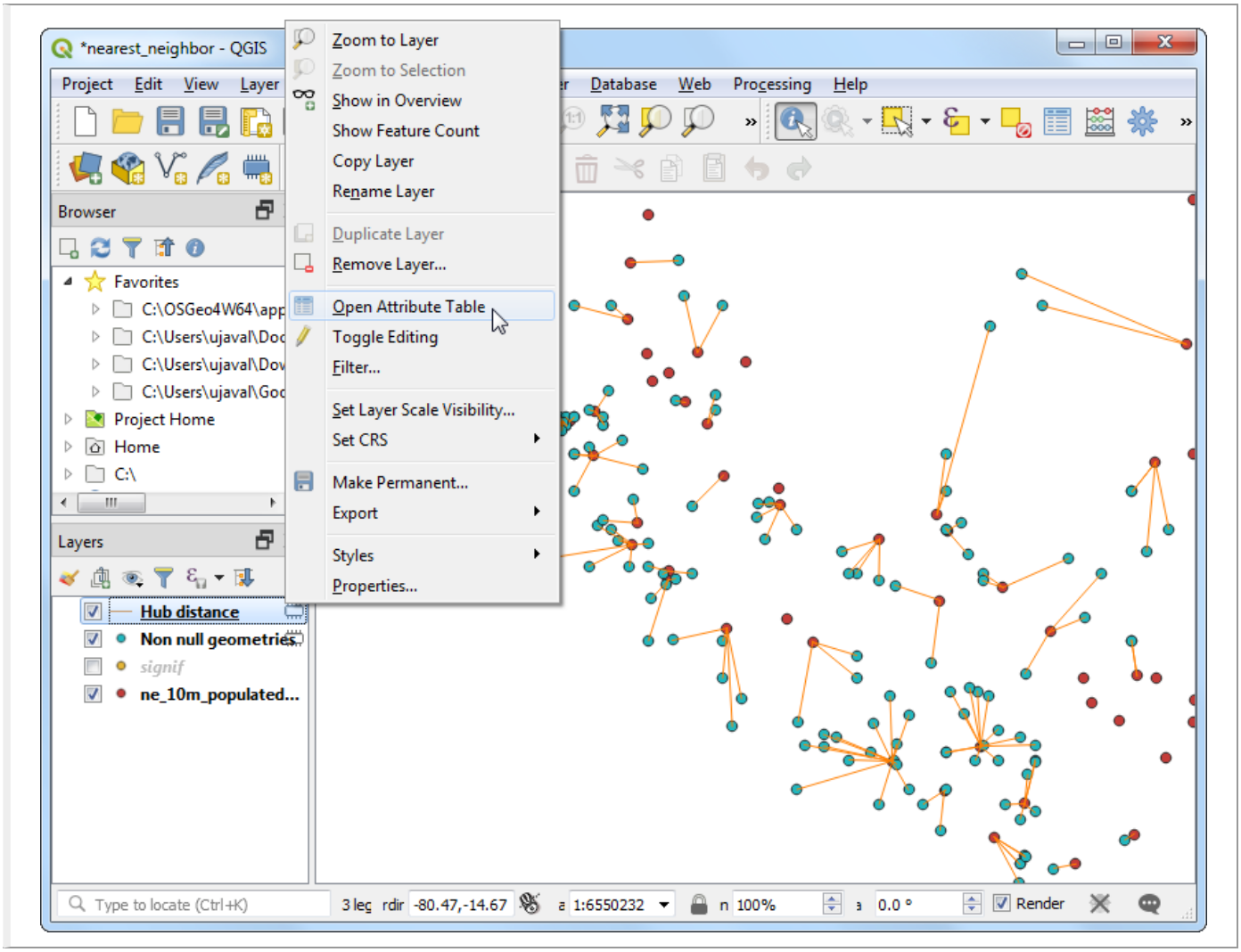
Note

If you need point layer as output, use the *Distance to nearest hub (points)* tool instead.

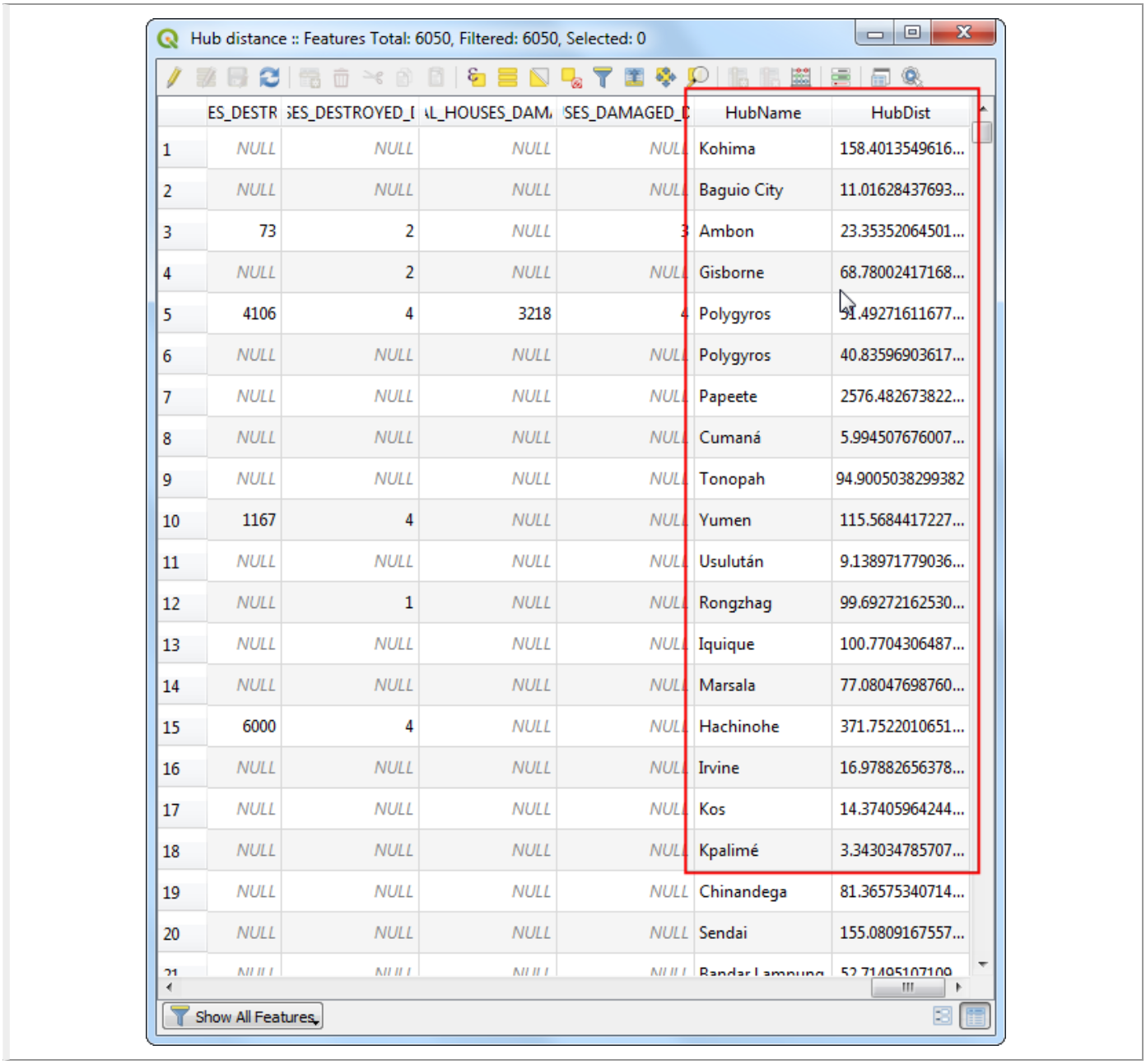
9. In the Distance to Nearest Hub (Line to Hub) dialog, select `Non null geometries` as the Source points layer. Select `ne_10m_populated_places_simple` as the Distination hubs layer. Select `name` as the Hub layer name attribute. The tool will also compute straight-line distance between the populated place and the nearest earthquake. Set `Kilometers` as the Measurement unit. Click Run. Once the processing finishes, click Close.



10. Back in the main QGIS window, you will see a new line layer called `Hub distance` loaded in the Layers panel. This layer has line features connecting each earthquake point to the nearest populated place. Right-click the `Hub distance` layer and select Open Attribute Table.

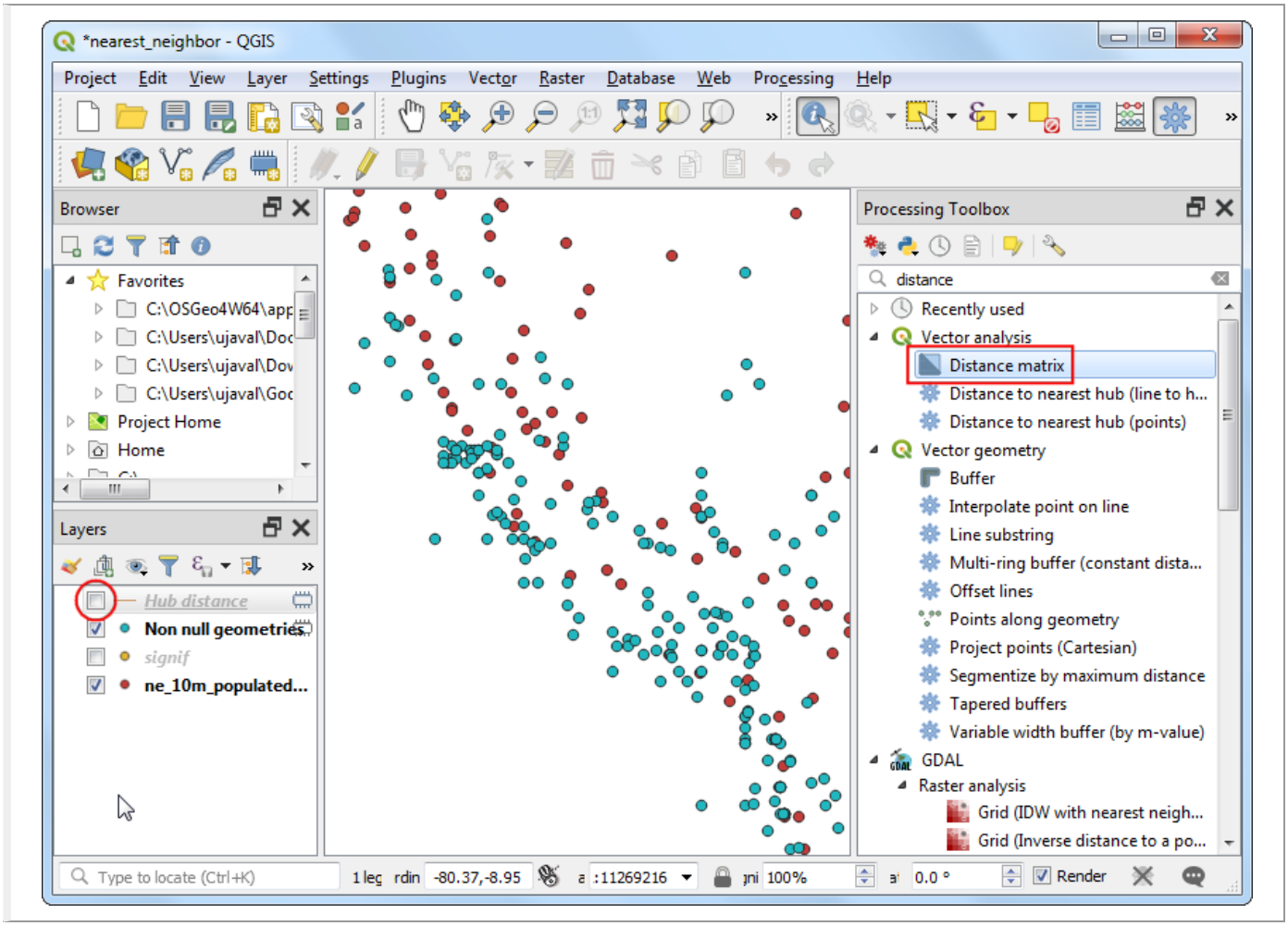


11. Scroll right to the last columns and you will see 2 new attributes called **HubName** and **HubDist** added to the original earthquake features. This is the name the distance to the nearest neighbor from the populated places layer.

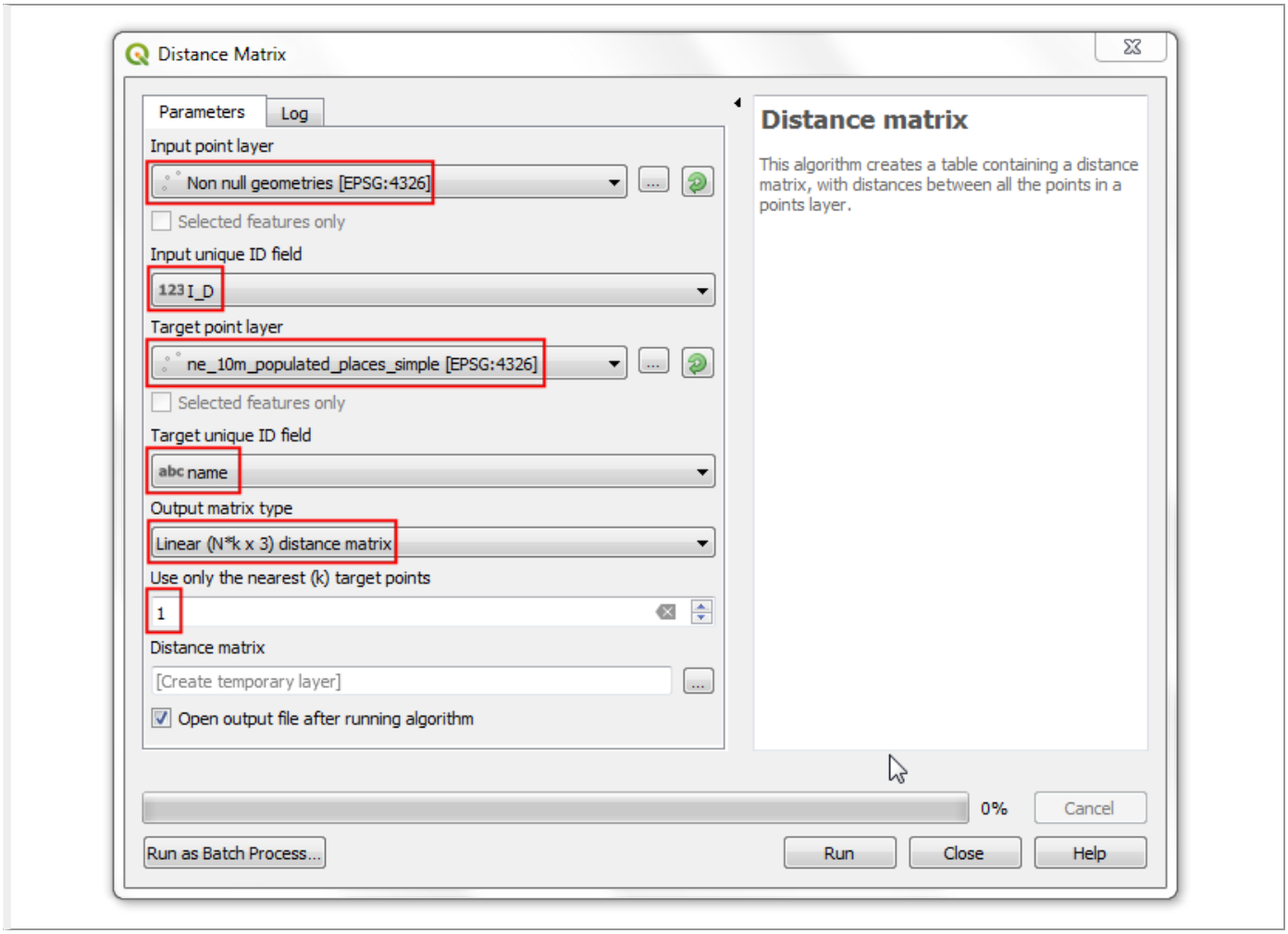


	ES_DESTR	ES_DESTROYED_I	L_HOUSES_DAM	SES_DAMAGED_I	HubName	HubDist
1	NULL	NULL	NULL	NULL	Kohima	158.4013549616...
2	NULL	NULL	NULL	NULL	Baguio City	11.01628437693...
3	73	2	NULL	3	Ambon	23.35352064501...
4	NULL	2	NULL	NULL	Gisborne	68.78002417168...
5	4106	4	3218	4	Polygyros	91.49271611677...
6	NULL	NULL	NULL	NULL	Polygyros	40.83596903617...
7	NULL	NULL	NULL	NULL	Papeete	2576.482673822...
8	NULL	NULL	NULL	NULL	Cumaná	5.994507676007...
9	NULL	NULL	NULL	NULL	Tonopah	94.9005038299382
10	1167	4	NULL	NULL	Yumen	115.5684417227...
11	NULL	NULL	NULL	NULL	Usulután	9.138971779036...
12	NULL	1	NULL	NULL	Rongzhag	99.69272162530...
13	NULL	NULL	NULL	NULL	Iquique	100.7704306487...
14	NULL	NULL	NULL	NULL	Marsala	77.08047698760...
15	6000	4	NULL	NULL	Hachinohe	371.7522010651...
16	NULL	NULL	NULL	NULL	Irvine	16.97882656378...
17	NULL	NULL	NULL	NULL	Kos	14.37405964244...
18	NULL	NULL	NULL	NULL	Kpalimé	3.343034785707...
19	NULL	NULL	NULL	NULL	Chinandega	81.36575340714...
20	NULL	NULL	NULL	NULL	Sendai	155.0809167557...
21	NULL	NULL	NULL	NULL	Bandar Lampung	52.71495107109...

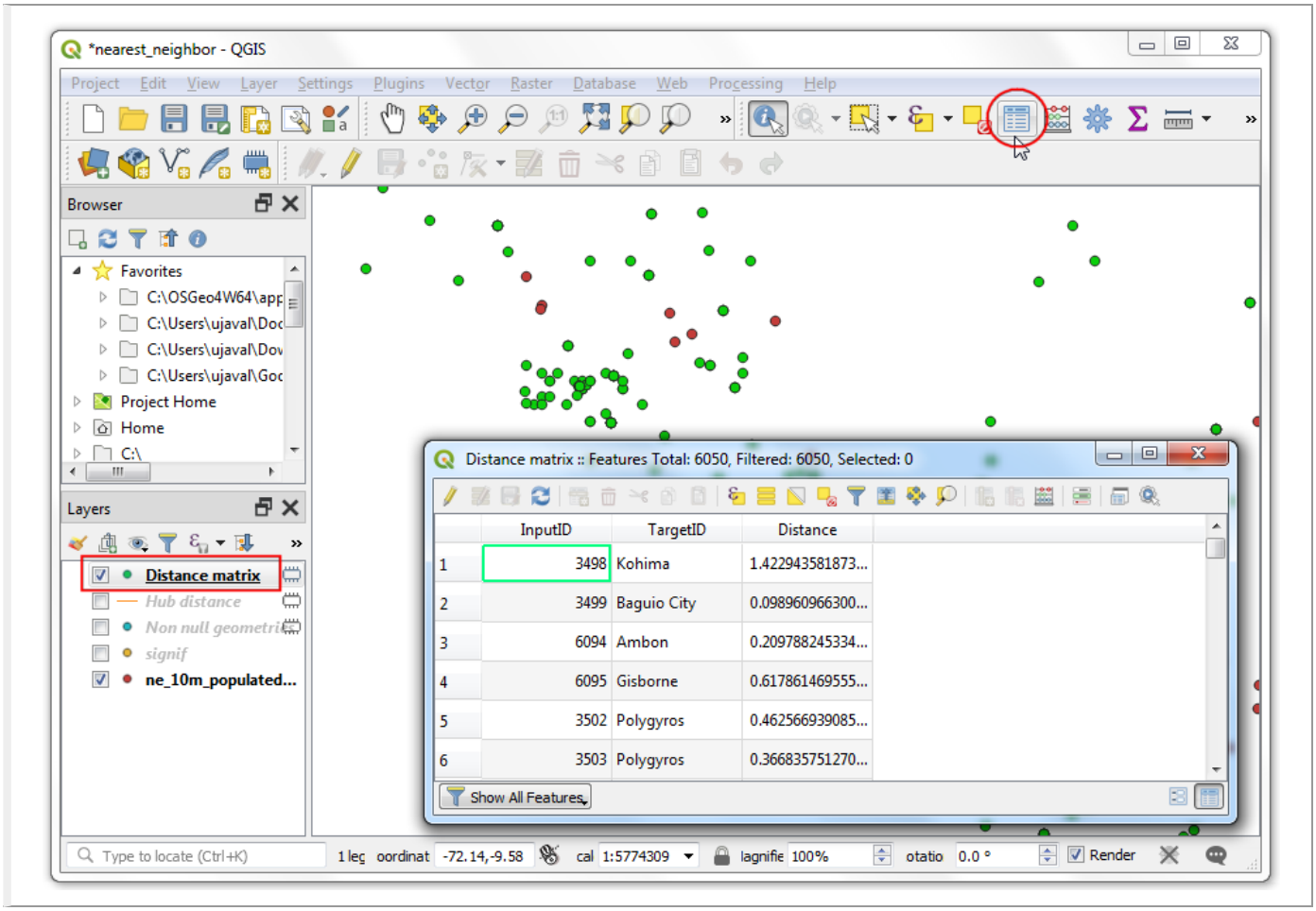
12. Our analysis is complete. We can now explore another tool that can also do a similar analysis. **Distance Matrix** is a powerful tool that allows you to not only compute distance to the nearest point, but to all the points from another layer. We can use this method as an alternative to the *Distance to nearest hub* tool. Un-check the `Hub distance` layer to hide it. Search and locate the Vector analysis ‣ Distance matrix tool.



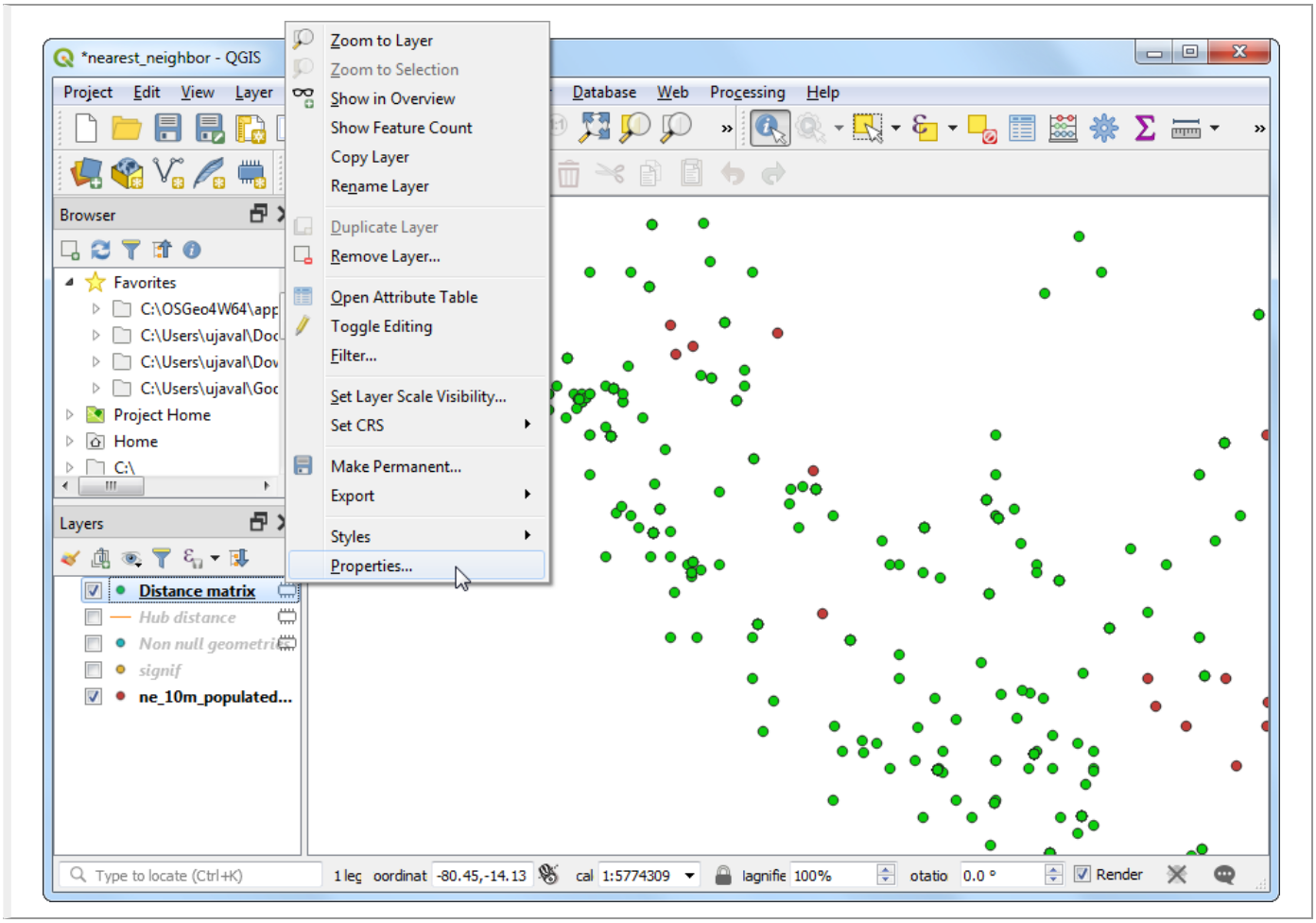
13. In the Distance matrix dialog, set `Non null geometries` as the Input point laeyer and `I_D` as the Input unique ID field. Set `ne_10m_populated_places_simple` as the Target point layer and `name` as the Target unique ID field. Select `Linear (N*k x 3) distance matrix` as the Output matrix type. The key here is to set the Use only the nearest (k) target points parameter to `1` - which will give you only the nearest neighbor in the output. Click Run to start the matrix calculation. Once the processing finishes, click Close.



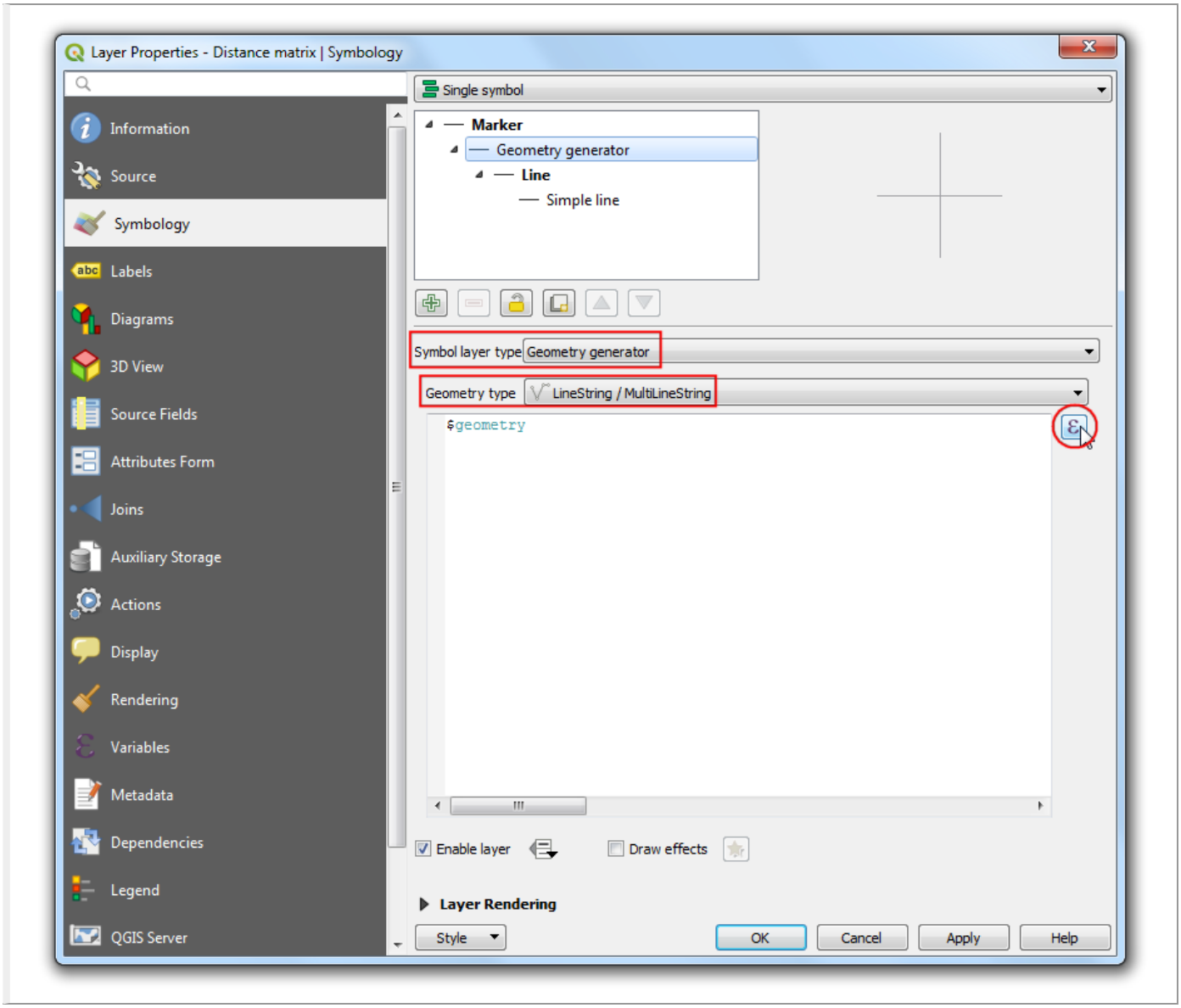
14. Once the processing finishes, a new layer called `Distance matrix` will be loaded. Note that the output of this tool is a layer containin *MultiPoint* geometries. Each feature contains 2 points - source and target. Open the Attribute Table for the layer. You will see that each feature has attributes mapping the earthquake to its nearest populated place. Note that the distance here is in the layer's CRS units (degrees).



15. At this point, you can save your results in the format of your choice by right-clicking the layer and selecting Export ‣ Save Features As. If you want to visualize the results better, we can easily create a hub-spoke rendering from the feature's geometry. Right-click the `Distance matrix` layer and select Properties.

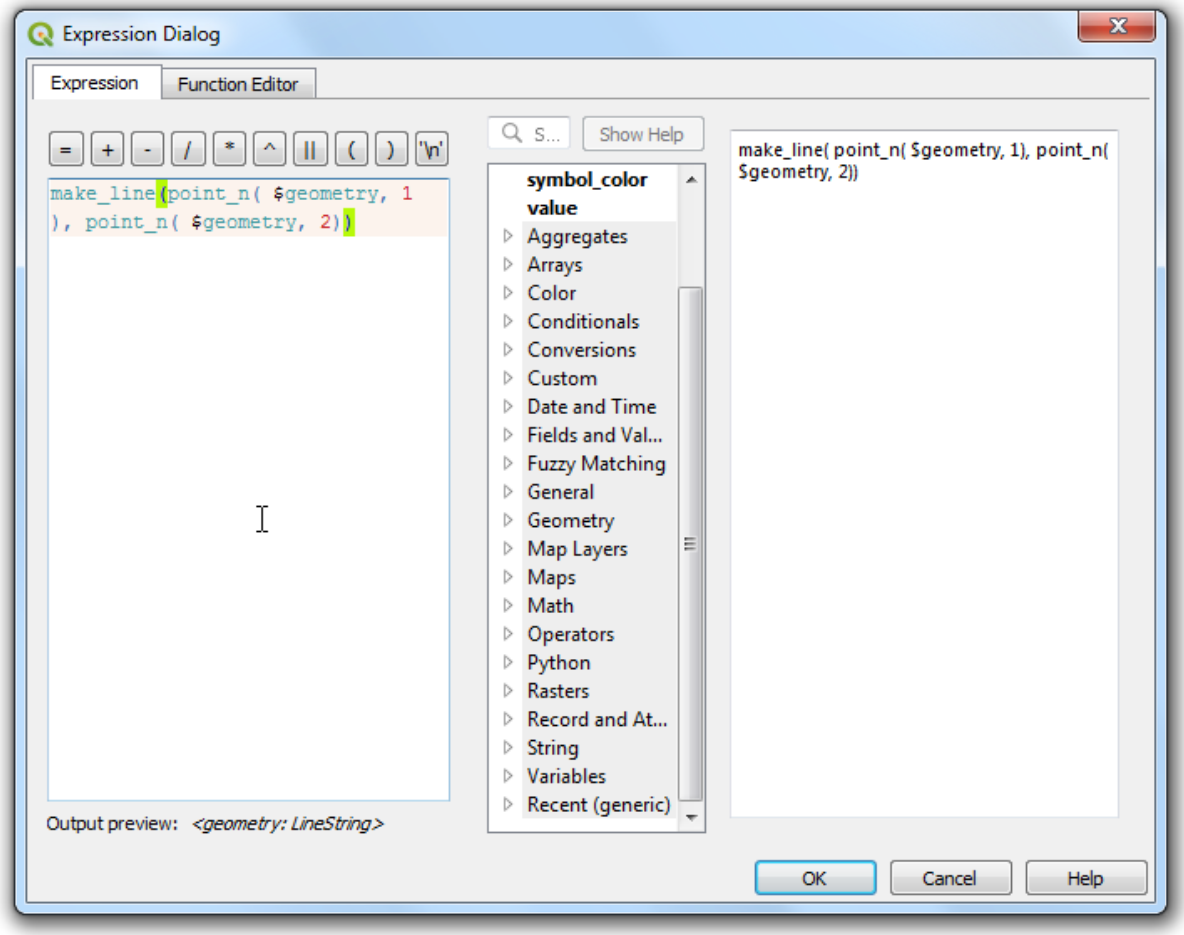


16. In the Properties dialog, switch to the Symbology tab. Click on the `Simple marker` sub-renderer and select `Geometry generator` as the Symbol layer type. Set `LineString / MultiLineString` as the Geometry type. Click the Expression button.

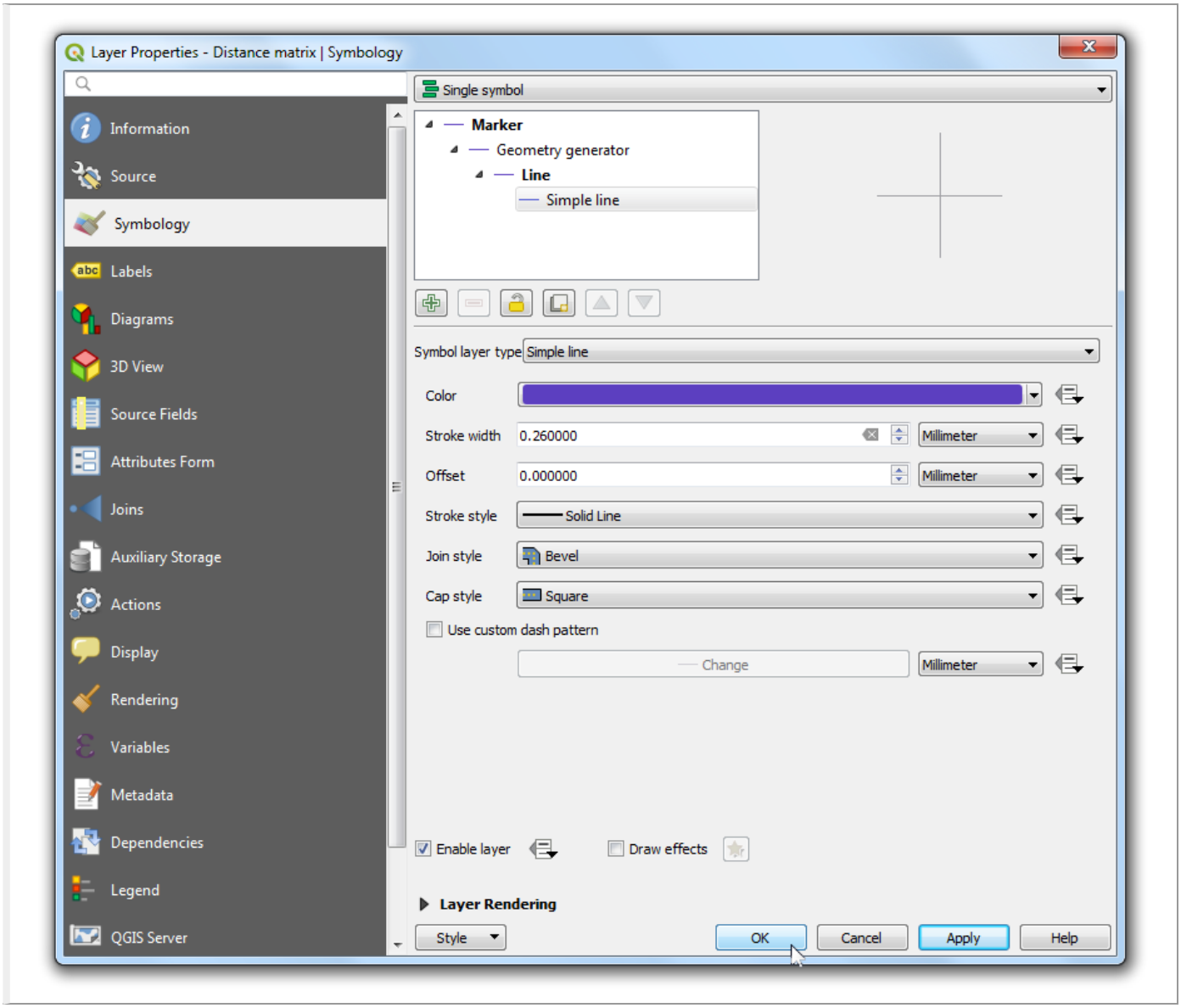


17. Here we can enter an expression to create a line geometry from the 2 points within each multi-point source geometry. Enter the following expression.

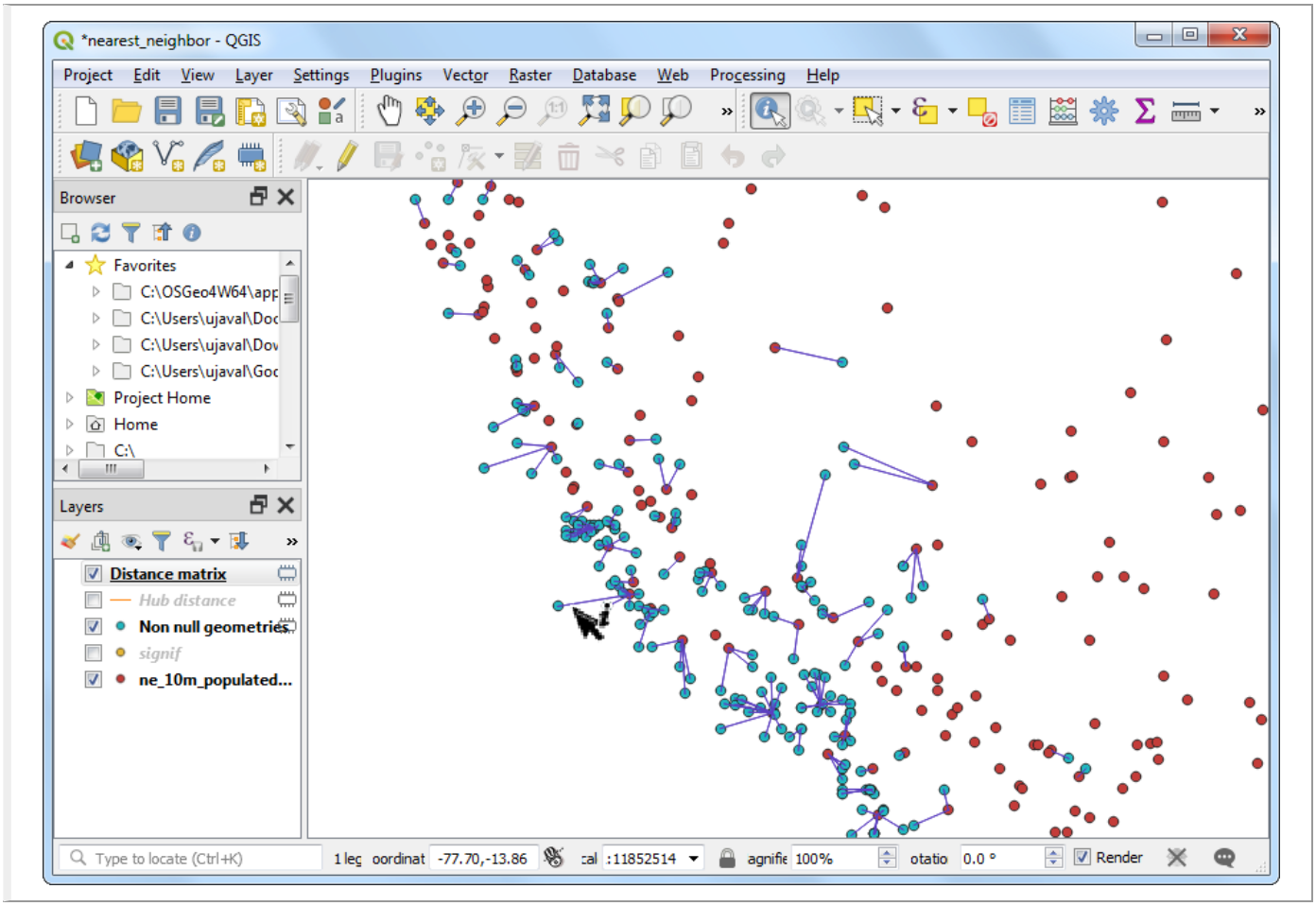
```
make_line(point_n( $geometry, 1), point_n( $geometry, 2))
```



18. Back in the Symbology tab, set the style of the line as per your liking and click OK.



19. You will see the `Distance matrix` layer now rendered with lines instead of points. Note that we did not have to create a new layer for this visualization. The layer still contains MultiPoint geometries, but it is dynamically rendered as lines based on the expression.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Sampling Raster Data using Points or Polygons (QGIS3)

Many scientific and environmental datasets come as gridded rasters. Elevation data (DEM) is also distributed as raster files. In these raster files, the parameter that is being represented is encoded as the pixel values of the raster. Often, one needs to extract the pixel values at certain locations or aggregate them over some area. This functionality is available in QGIS via processing algorithms. `Sample raster values` for point layers and `Zonal Statistics` for polygon layers.

Overview of the task

Given a raster grid of daily maximum temperature in the continental US, we need to extract the temperature at a point layer of all urban areas and calculate the average temperature for a polygon layer of each county in the US.

Other skills you will learn

- Select and remove multiple layers from QGIS Table of Contents.

Get the data

NOAA's Climate Prediction Center (http://www.cpc.ncep.noaa.gov/) provides GIS data (http://www.cpc.ncep.noaa.gov/products/GIS/GIS_DATA/) related to temperature and precipitation in the US. Download the latest grid file for maximum temperatures (ftp://ftp.cpc.ncep.noaa.gov/GIS/GRADS_GIS/GeoTIFF/TEMP/us_tmax/). The file will be named `us.tmax_nohads_ll_{YYYYMMDD}_float.tif`

We will use a CSV file from 2018 US Gazetteer (https://www.census.gov/geographies/reference-files/time-series/geo/gazetteer-files.2018.html) representing urban areas in the US. Download the Urban Areas Gazetteer File (https://www2.census.gov/geo/docs/maps-data/data/gazetteer/2018_Gazetteer/2018_Gaz_ua_national.zip).

US Census Bureau (https://www.census.gov/en.html) provides TIGER/Line Shapefiles (https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html). You can visit the FTP site (https://www2.census.gov/geo/tiger/TIGER2018/) and download census tracts shapefile for California. Download Census Tracts for California (https://www2.census.gov/geo/tiger/TIGER2018/COUNTY/tl_2018_us_county.zip) file.

For convenience, you may directly download a copy of the datasets from the links below:

us.tmax_nohads_ll_20190501_float.tif (http://www.qgistutorials.com/downloads/us.tmax_nohads_ll_20190501_float.tif)

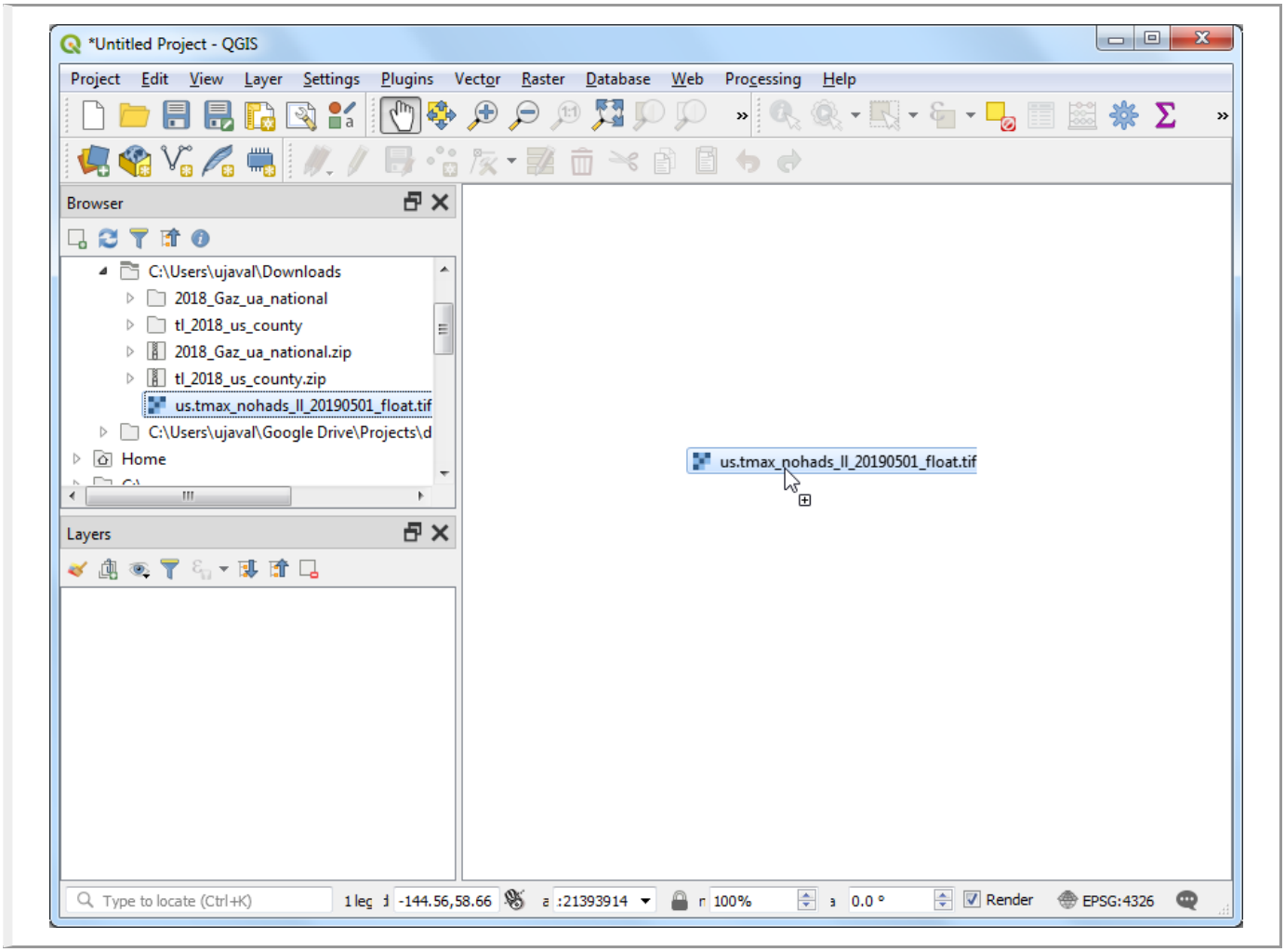
2018_Gaz_ua_national.zip (http://www.qgistutorials.com/downloads/2018_Gaz_ua_national.zip)

tl_2018_us_county.zip (http://www.qgistutorials.com/downloads/tl_2018_us_county.zip)

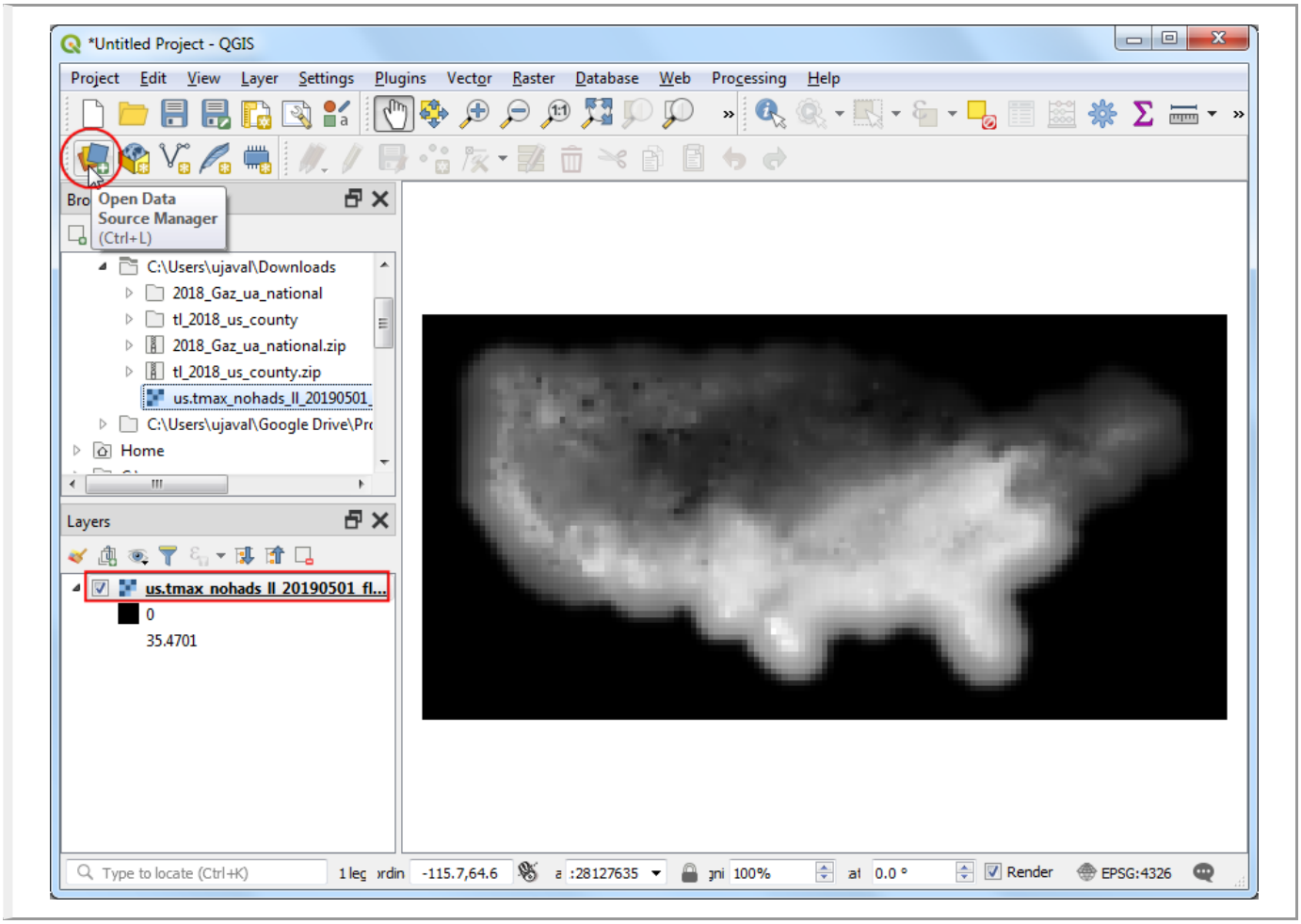
Data Sources: [NOAACPC] (../credits.html#noaacpc), [USGAZETTEER] (../credits.html#usgazetteer) [TIGER] (../credits.html#tiger)

Procedure

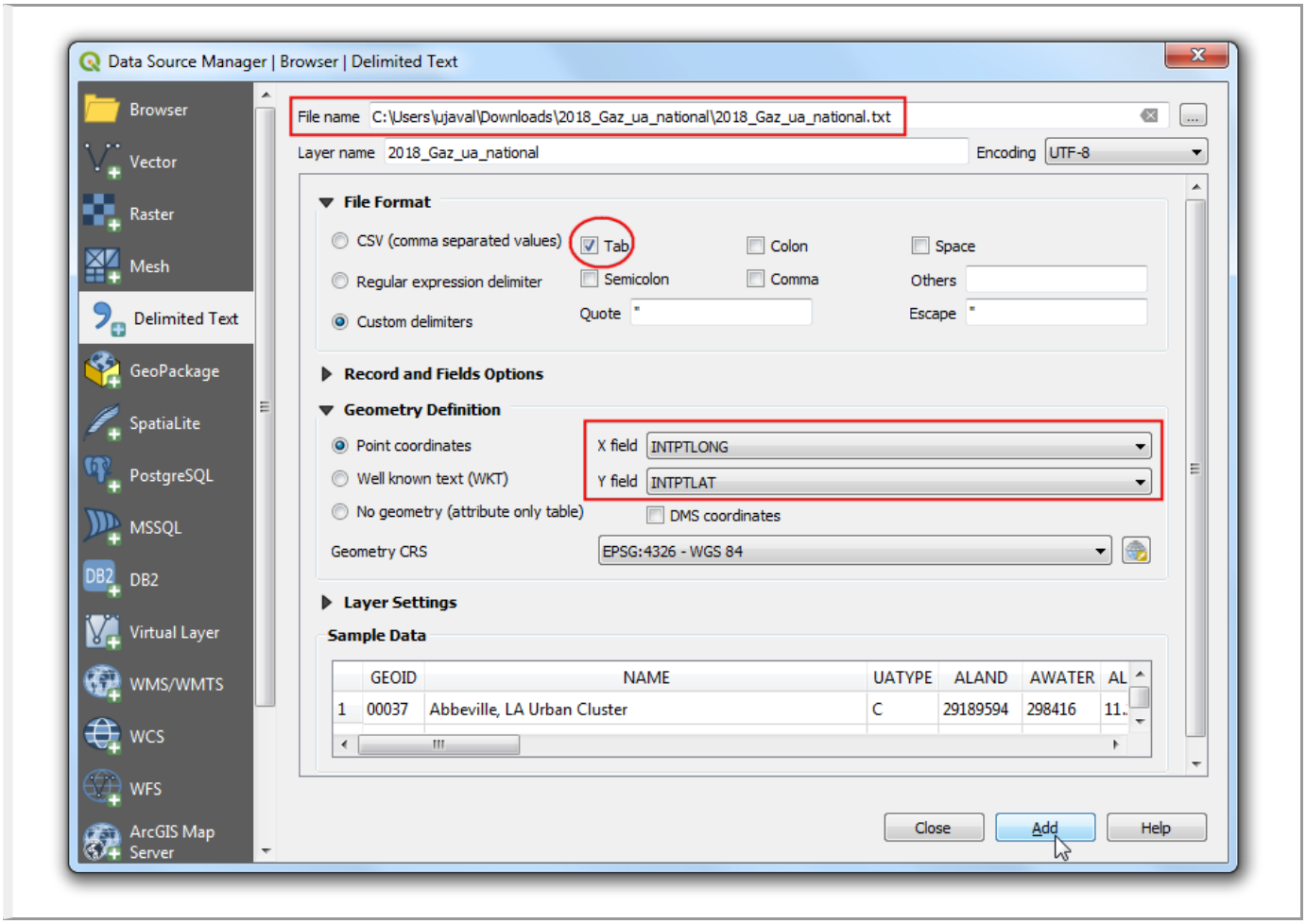
1. Unzip and extract both `2018_Gaz_ua_national.zip` and `tl_2018_us_county.zip` to a folder on your computer. Open QGIS and locate the `us.tmax_nohads_ll_20190501_float.tif` file in the QGIS Browser drag it to the canvas.



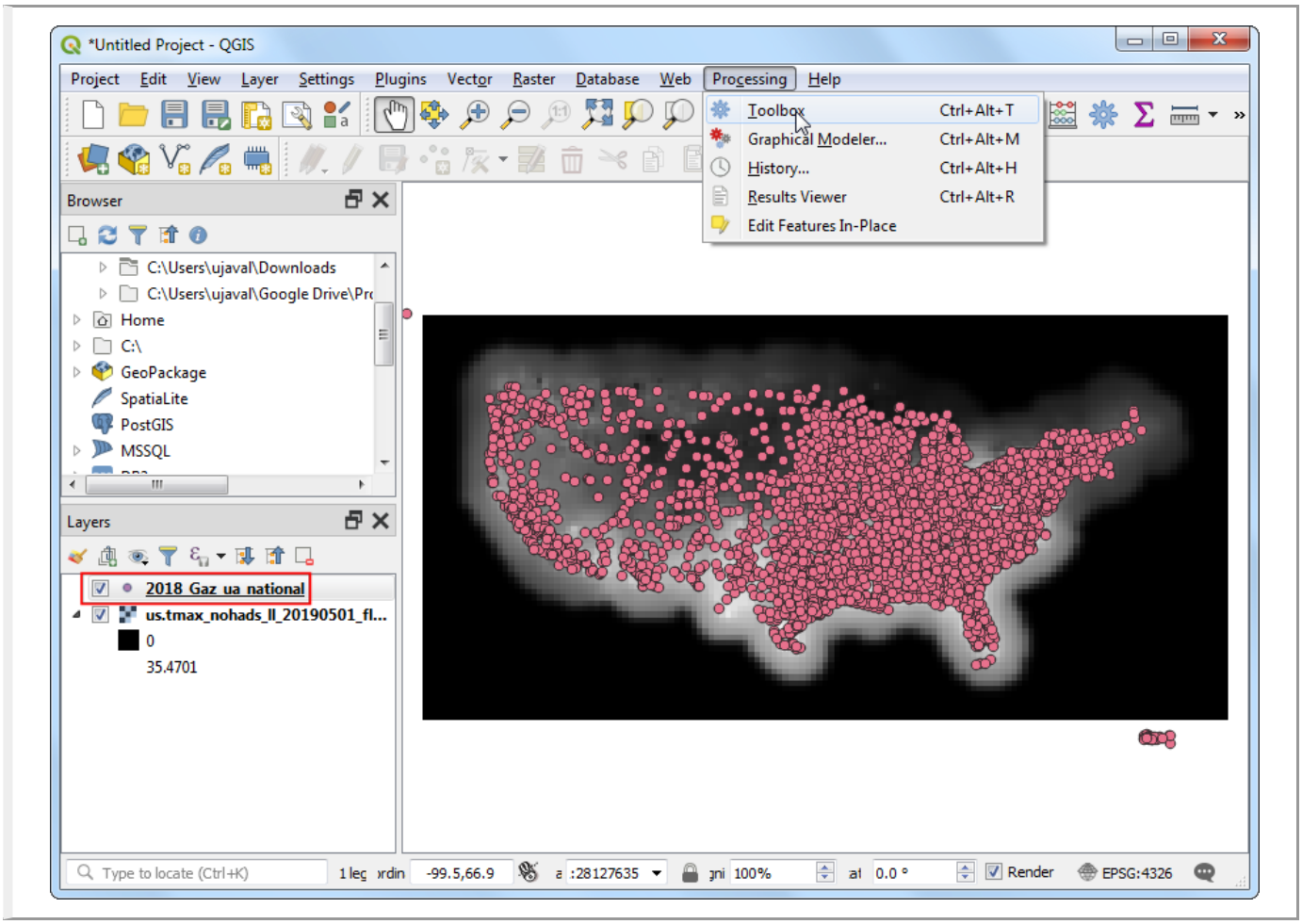
2. You will see a new raster layer `us.tmax_nohads_ll_20190501_float` loaded in the Layers panel. This raster layer contains the maximum temperature recorded at each pixel in degrees Celsius. Next we will load the urban areas point file. This file comes as a text file in the Tab Separated Values (TSV) format. Click the Open Data Source Manager button on the Data Source Toolbar.



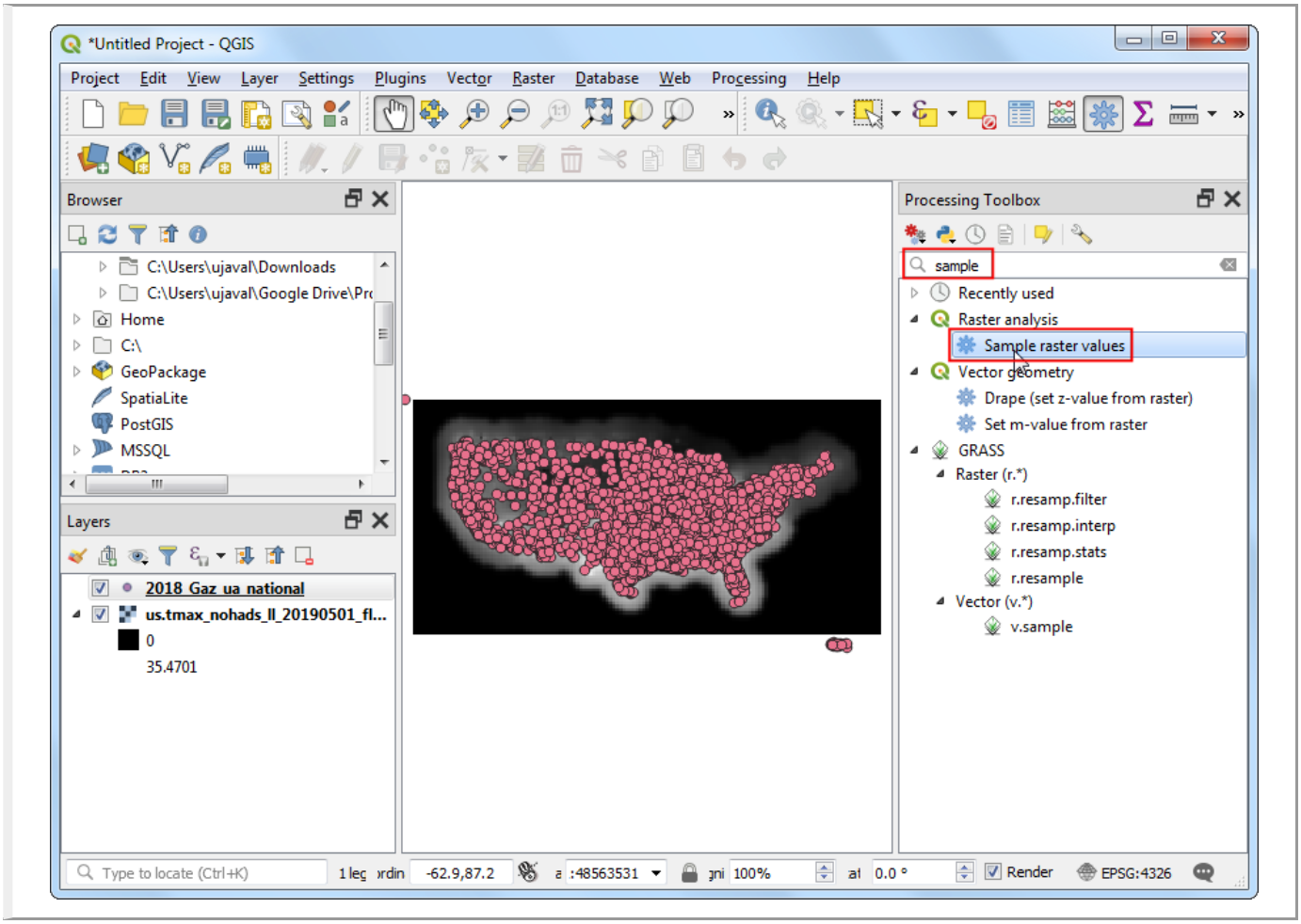
3. Switch to the Delimited Text tab. Click the ... button next to File name and specify the path to the text file you downloaded. In the File format section, select Custom delimiters and check Tab. Select `INTPTLONG` as the X field and `INTPTLAT` as the Y field. Click Add and then Close.



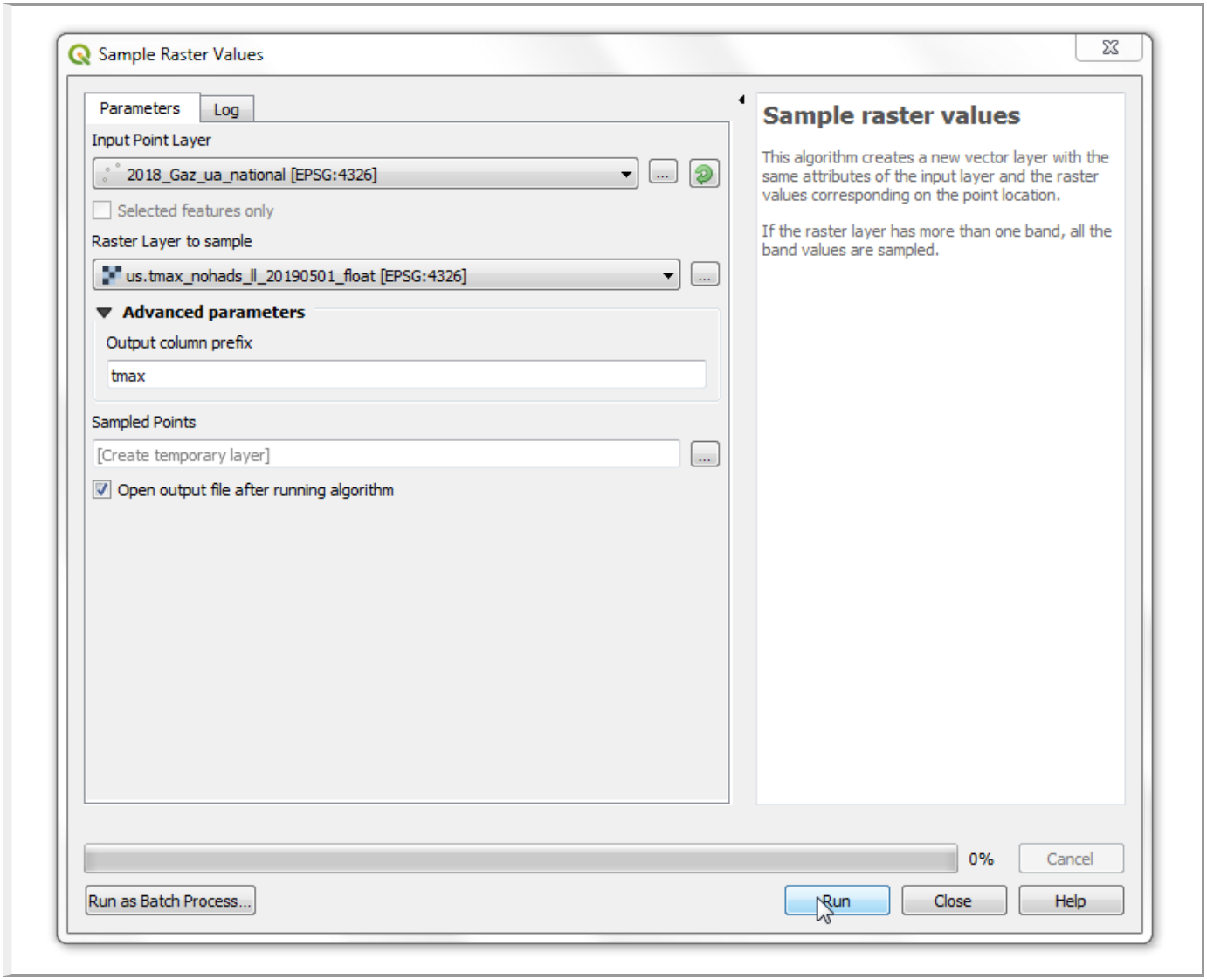
4. A new point layer `2018_Gaz_ua_national` will be loaded in the Layers panel. Now we are ready to extract the values from the raster layer at these points. Go to Processing ‣ Toolbox.



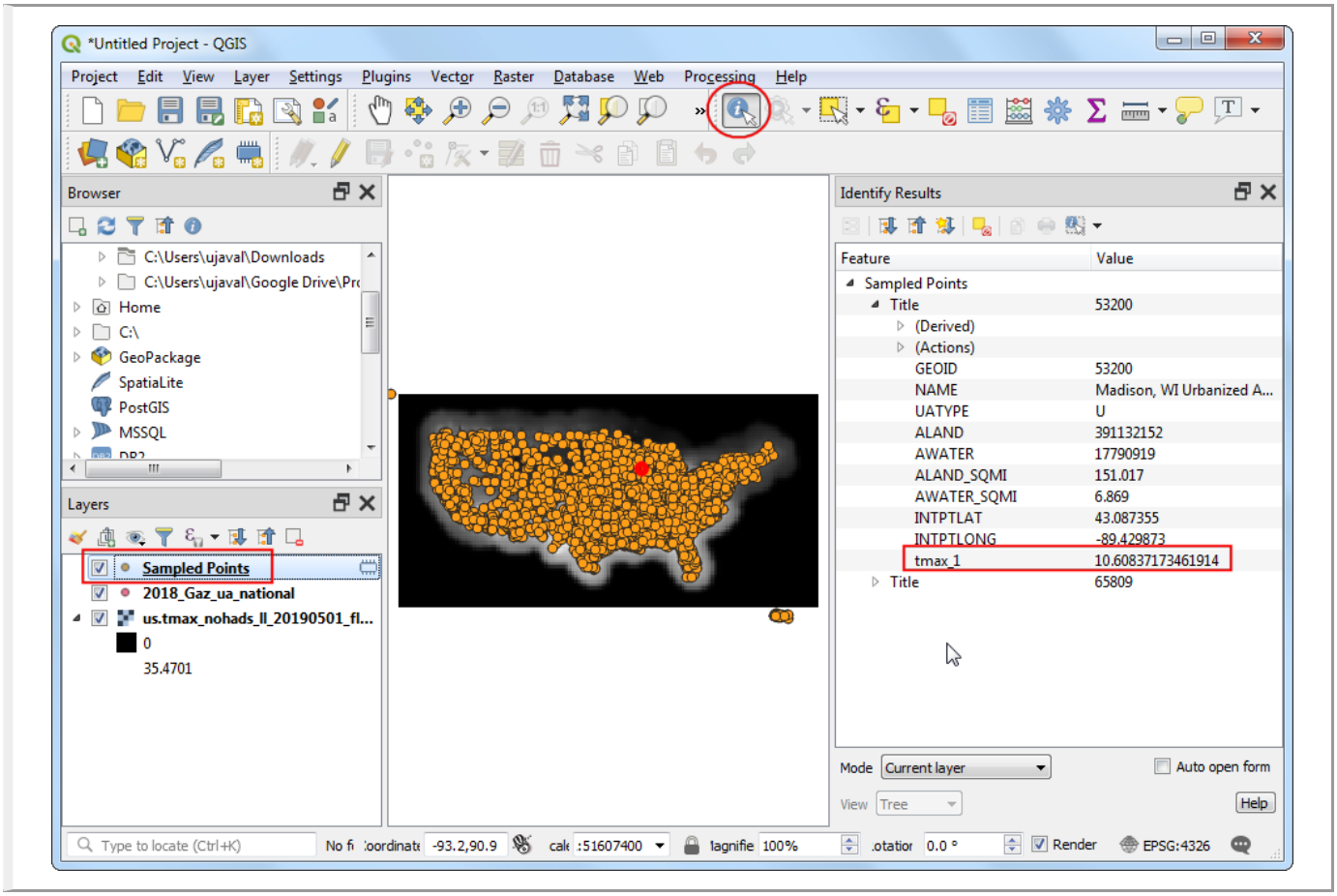
5. Search and locate the Raster analysis ▸ Sample raster values algorithm. Double-click to launch it.



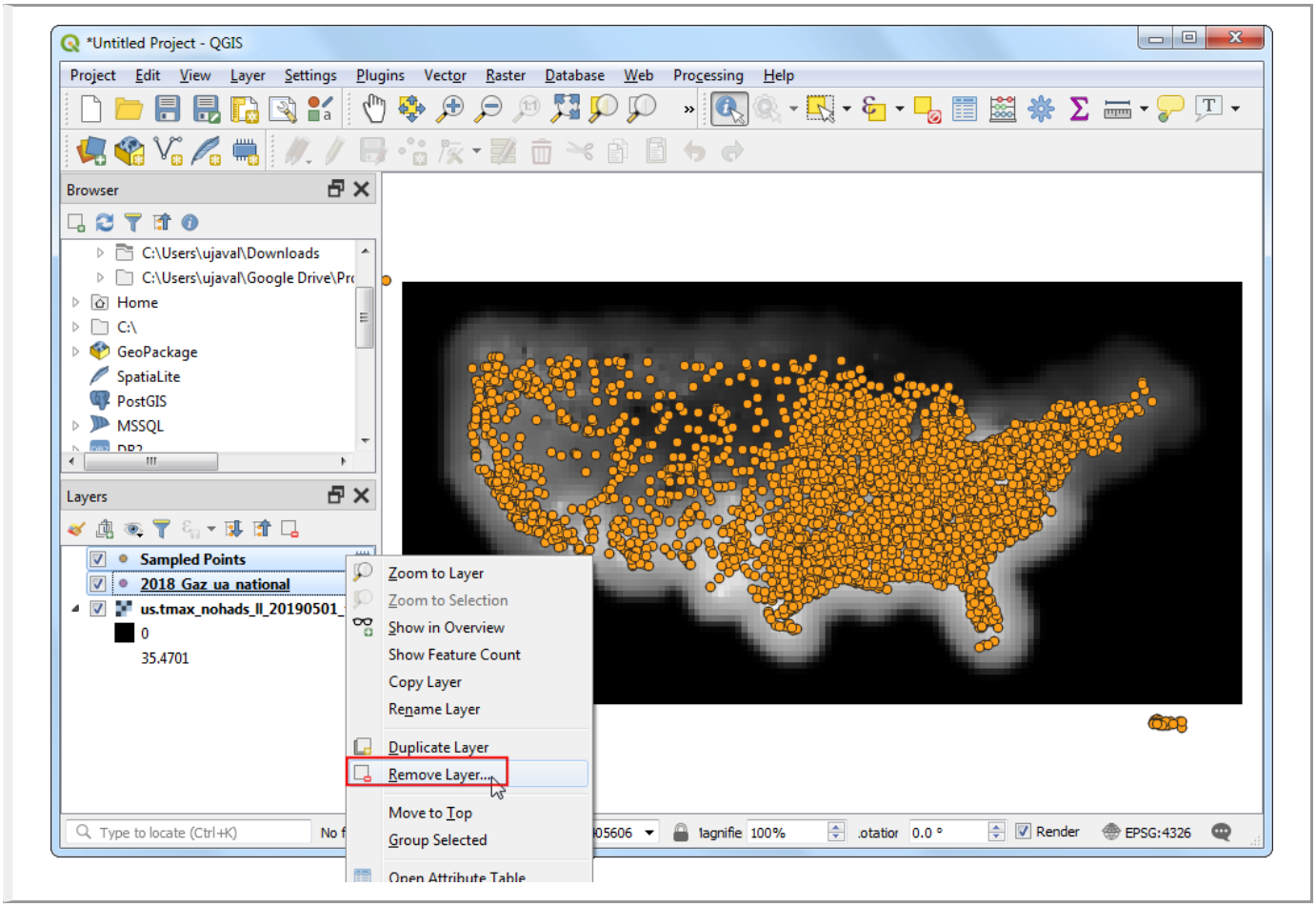
6. Select `2018_Gaz_ua_national` as the Input Point Layer. Select `us.tmax_nohads_ll_20190501_float` as the Raster Layer to sample. Expand the Advanced parameters and enter `tmax` as the Output column prefix. Click Run. Once the processing finishes, click Close.



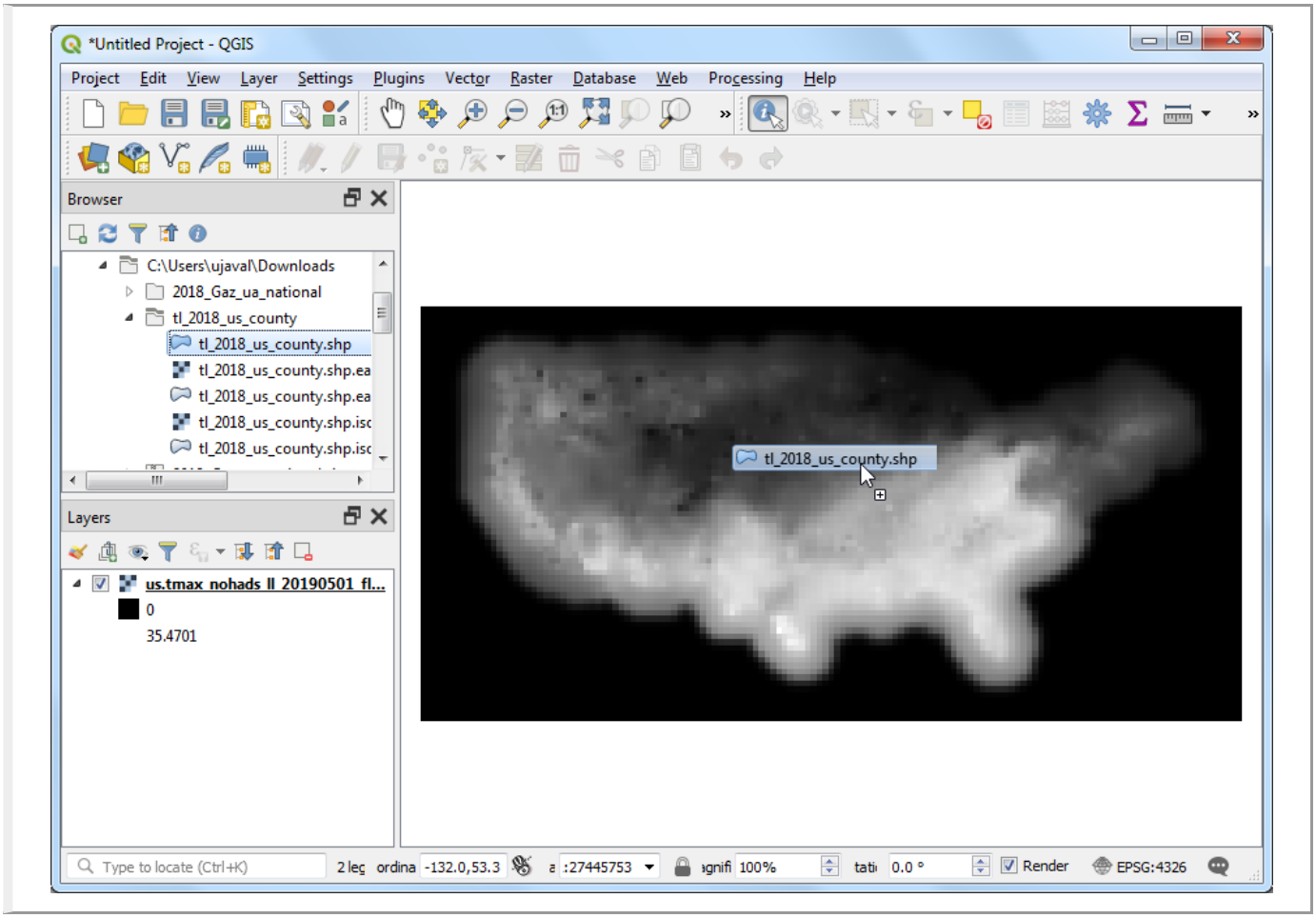
7. A new layer `Sampled Points` will be loaded in the Layers panel. Select the Identify tool in the Attributes Toolbar and click on any point. You will see the attributes displayed in the Identify Results panel. You will see a new attribute called **tmax_1** added to each feature. This is the pixel value of the raster layer extracted at the point's location. The *1* represents the band number of the raster. If the raster layer had multiple bands, you would see multiple new columns in the output layer.



8. First part of our analysis is over. Let's remove the unnecessary layers. Hold the `Shift` key and select `Sampled Points` and `2018_Gaz_ua_national` layers. Right-click and select Remove to remove them from QGIS. When prompted for Remove 2 legend entries?, select OK.



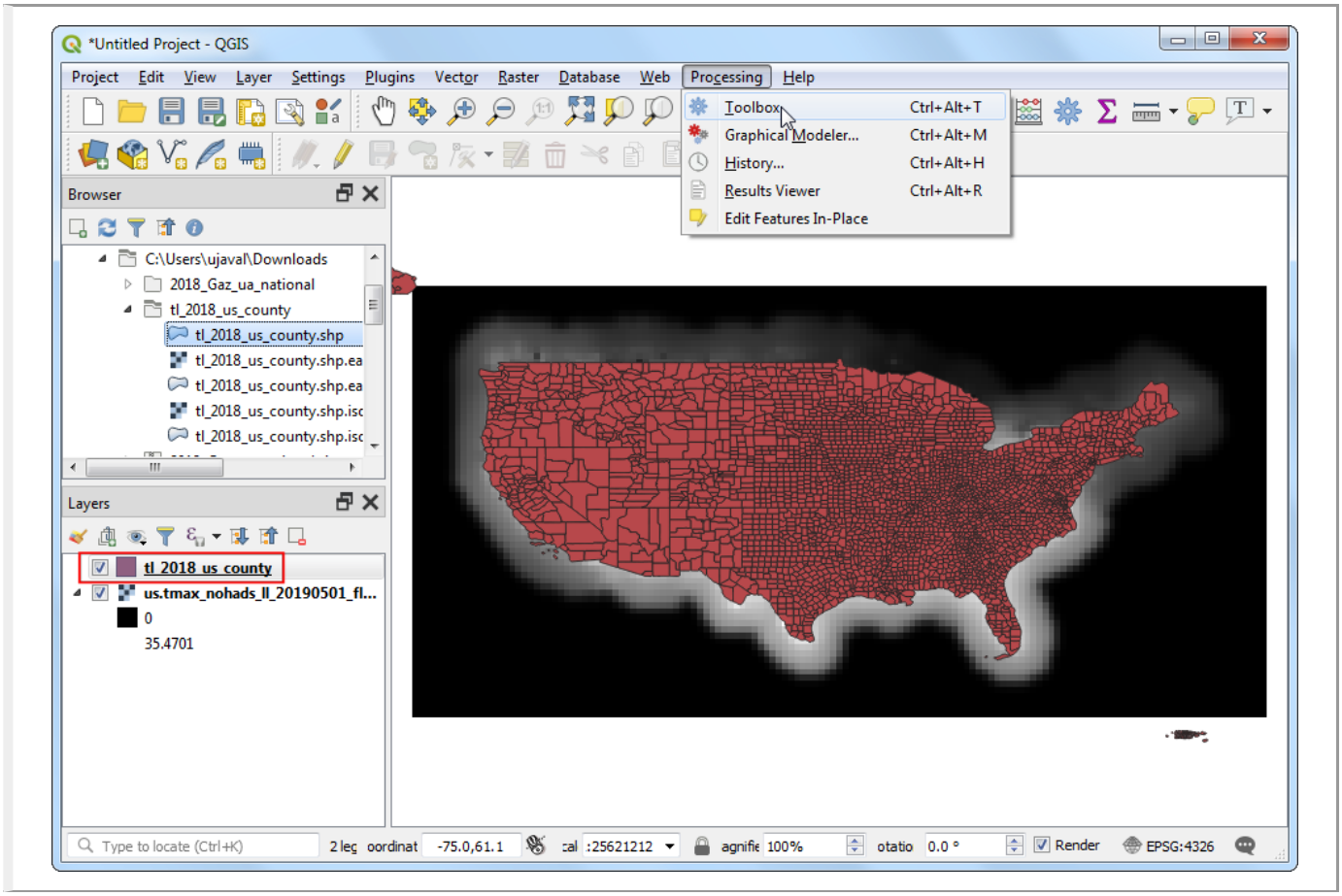
9. Now we will use the counties layer to sample the raster and calculate average temperature for each county. Locate the `tl_2018_us_county.shp` file in the QGIS Browser drag it to the canvas.



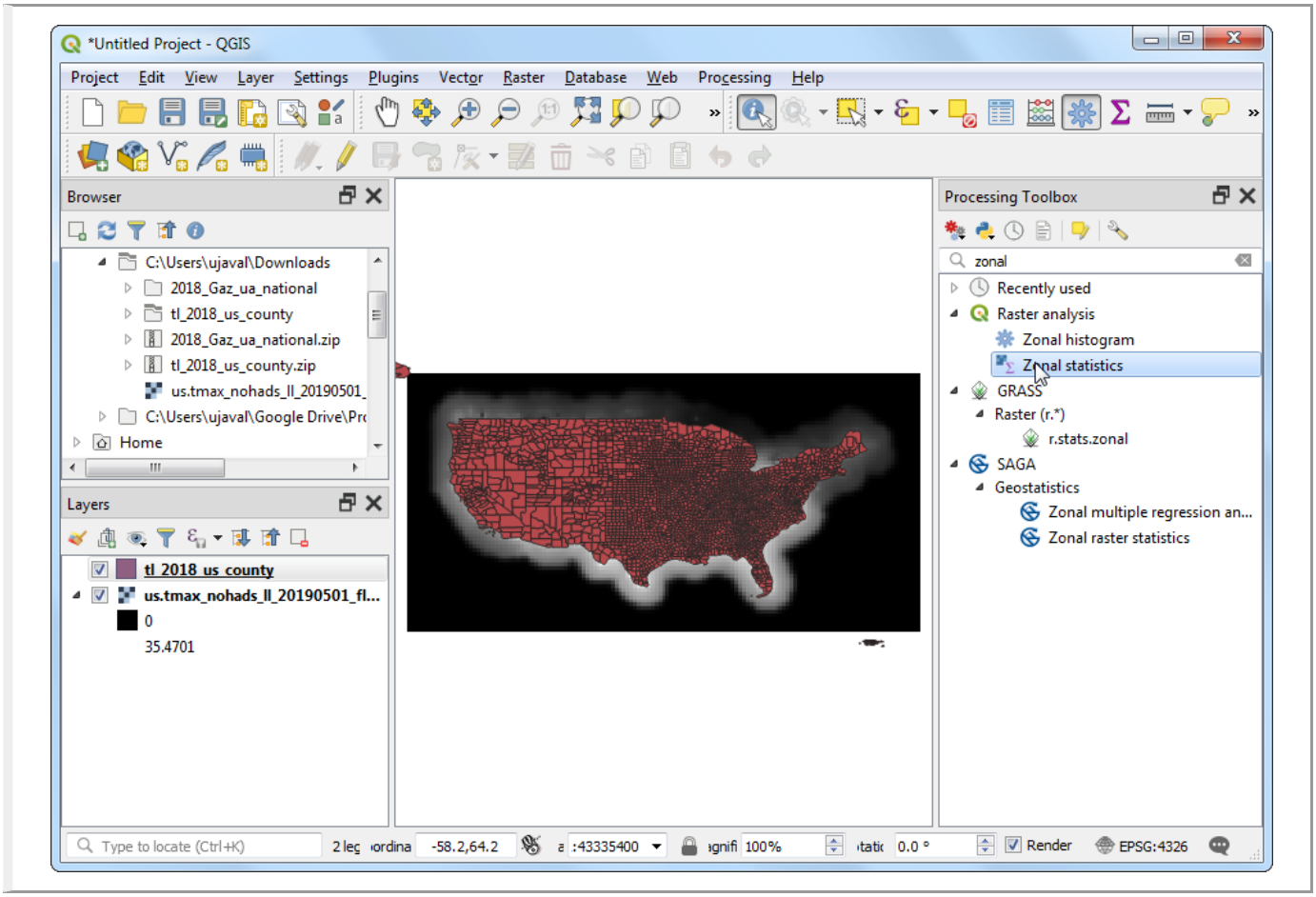
Note

Most processing algorithms will read the input layer and create a new layer. But the **Zonal Statistics** algorithm is different. It modifies the input layer and adds new attributes to it. That's why it is important to unzip the input files first. QGIS can load a layer from a zip archive directly, but it cannot modify a zipped layer. The processing algorithm will fail if it cannot update the input layer.

10. A new layer `tl_2018_us_county` will be loaded to the Layers panel. Go to Processing ▸ Toolbox.

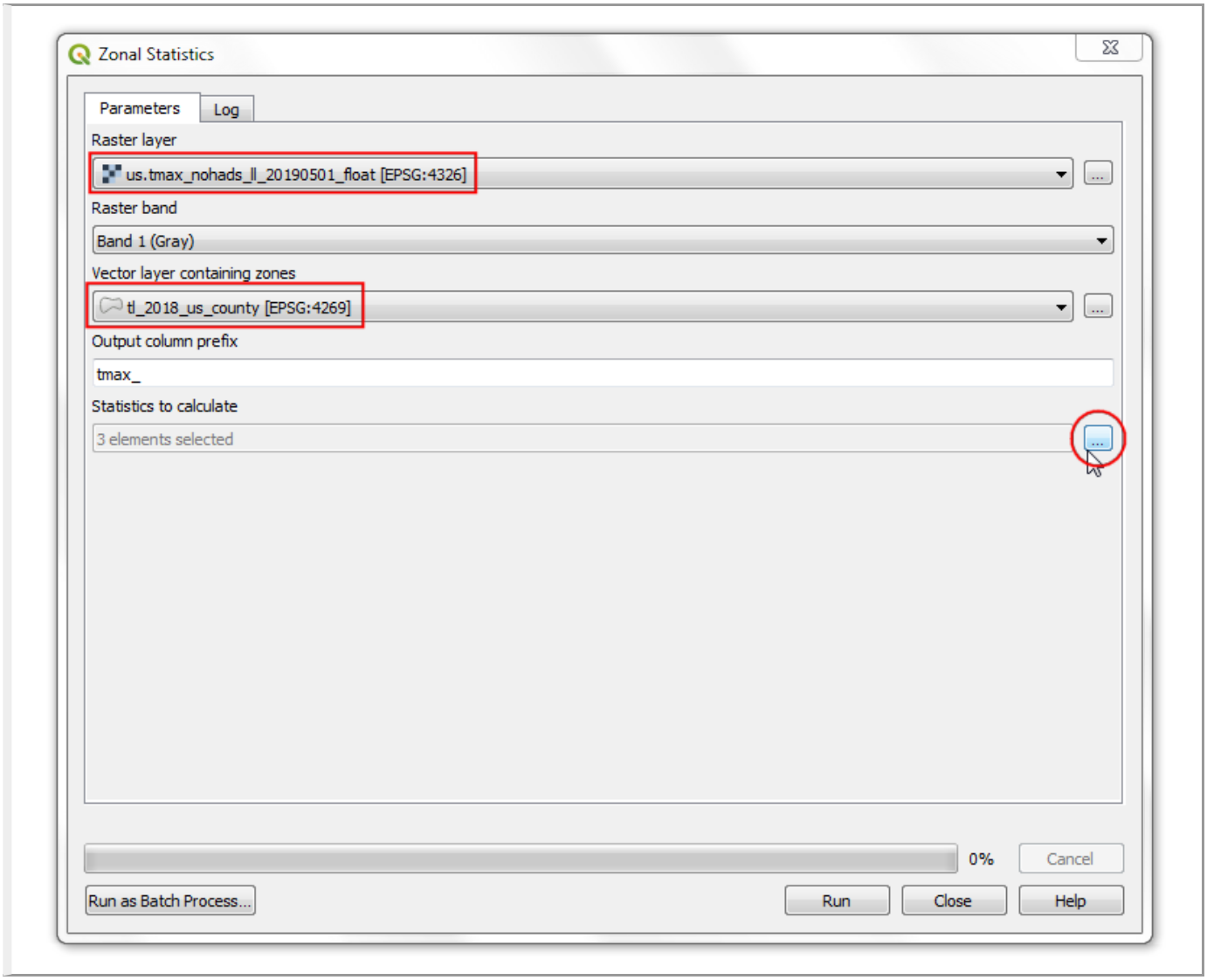


11. Search and locate the Raster analysis ‣ Zonal statistics algorithm and double-click to launch it.

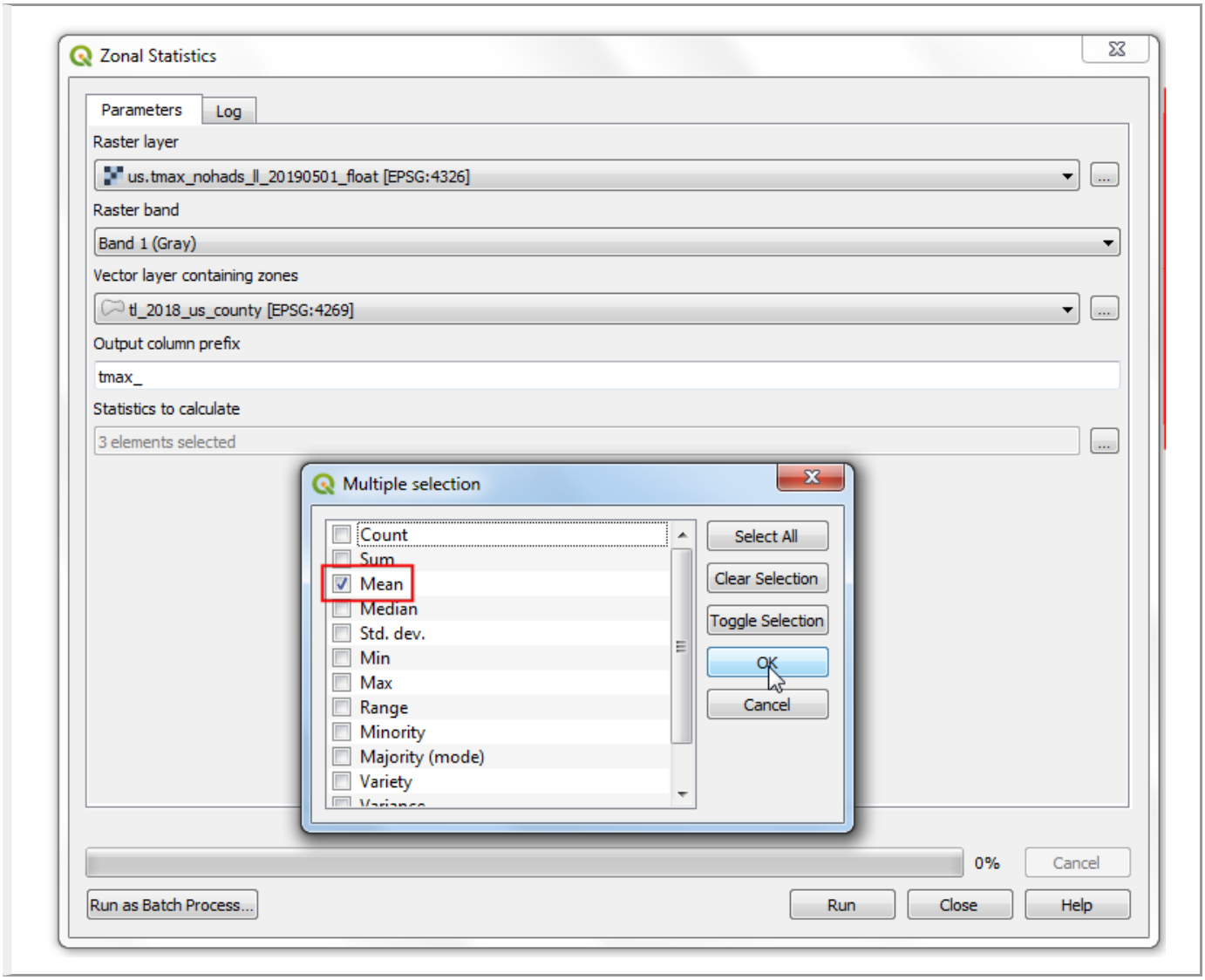


12. Select `us.tmax_nohads_ll_20190501_float` as the Raster layer and `tl_2018_us_county` as the Vector layer containing zones. Enter `tmax_` as the Output column prefix. Click the ... next to Statistics to

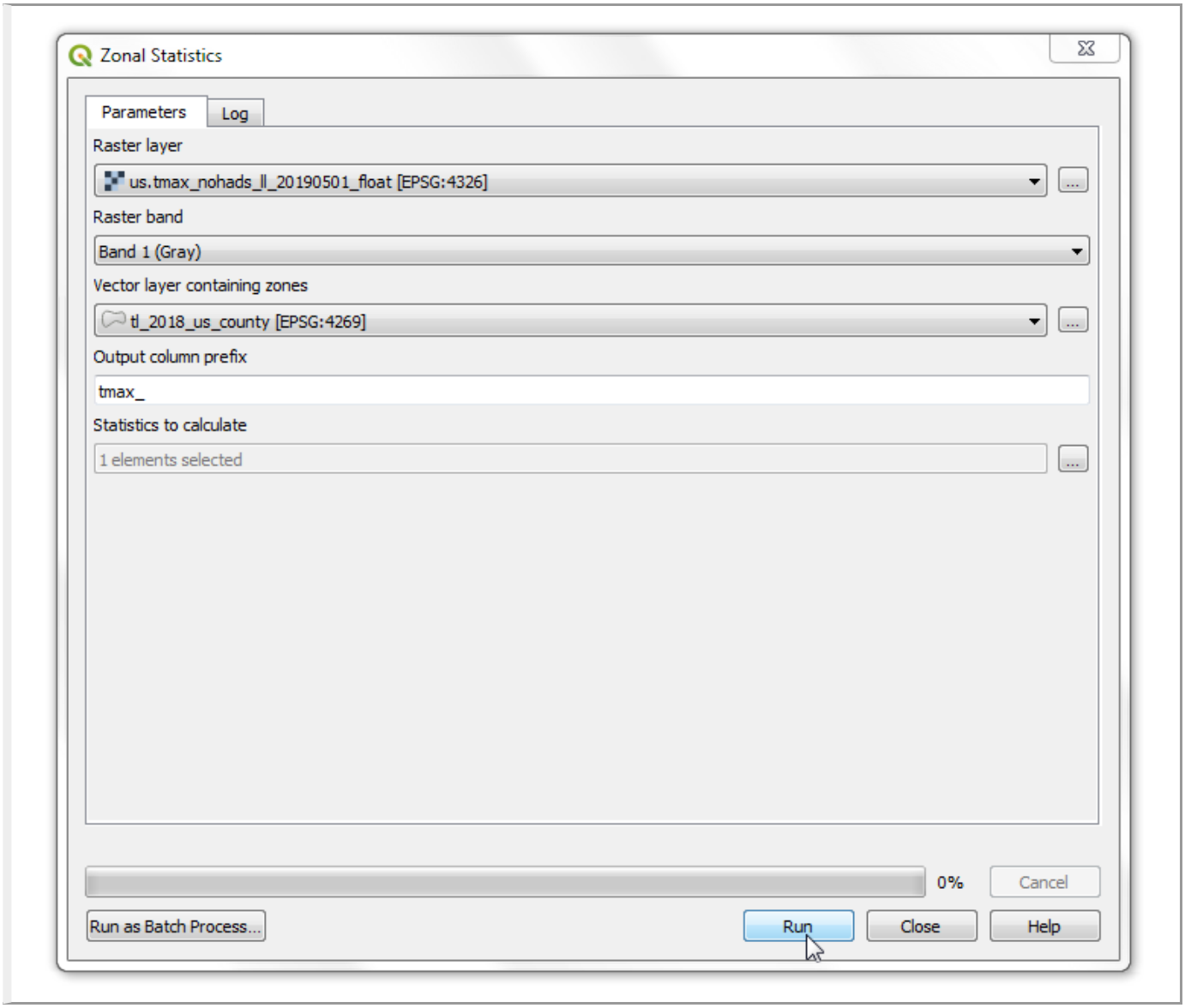
calculate.



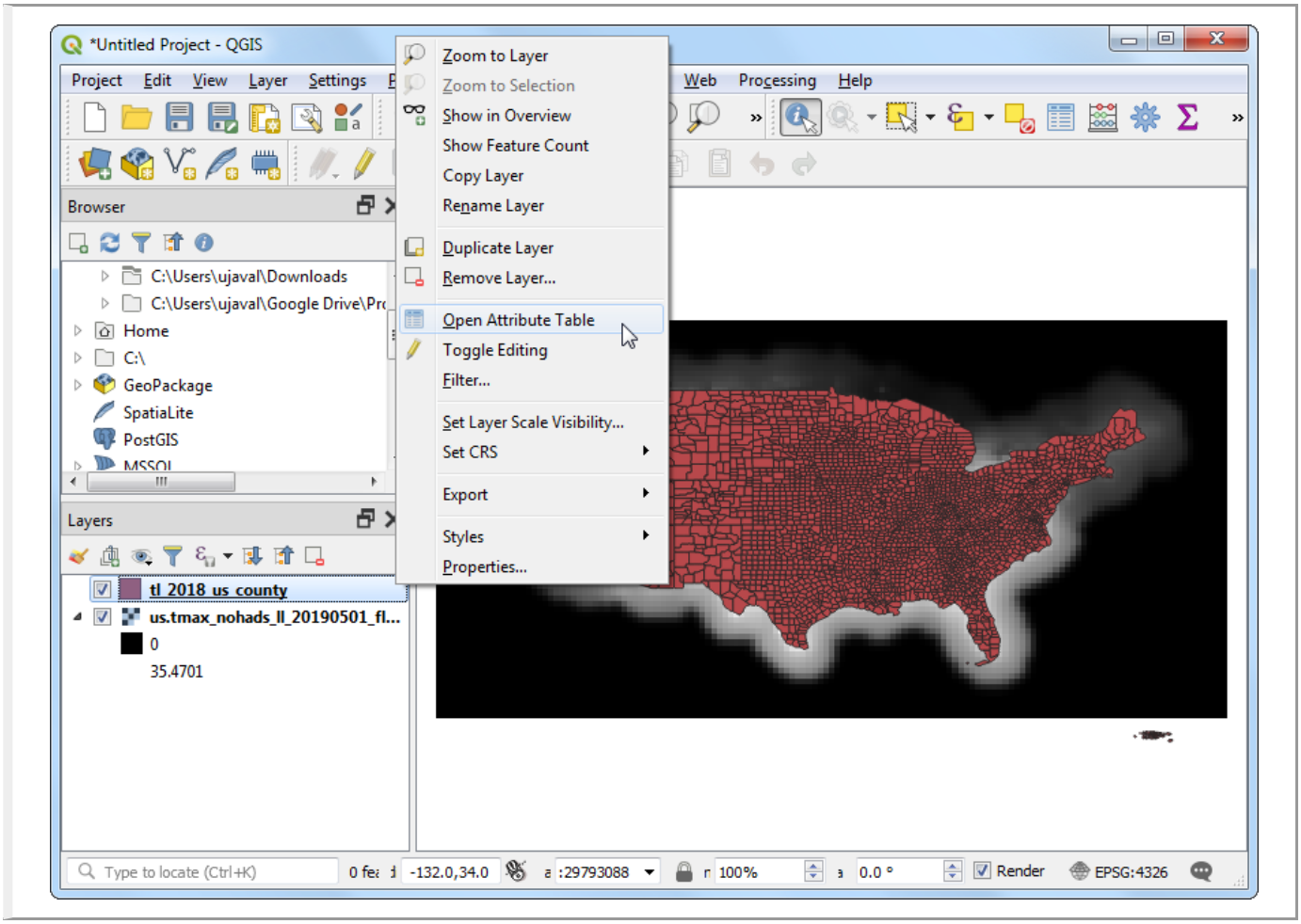
13. Select only the `Mean` value and click OK.



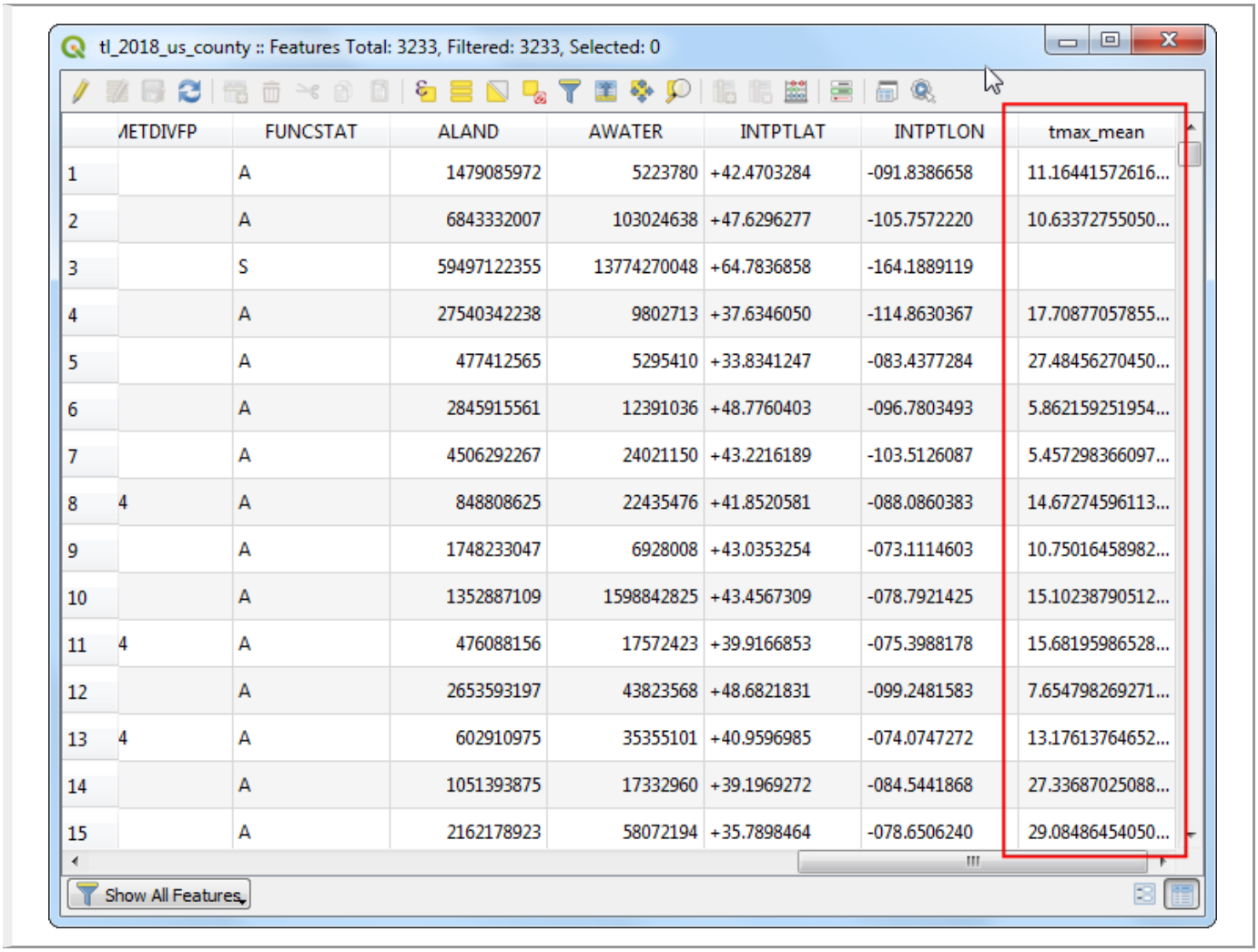
14. Click Run to start the processing. The algorithm may take a few minutes to complete. Click Close.



15. As noted earlier, the **Zonal Statistics** algorithm doesn't create a new layer, but modifies the zone layer. Right-click the `tl_2018_us_county` layer, and select Open Attribute Table.



16. You will see a new column called `tmax_mean` added to the attribute table. This contains the average temperature value extracted over the polygon for each feature. There are some null values because those counties (belonging to Alaska, Hawaii and Puerto Rico) are outside of the raster layer's extent.



	METDIVFP	FUNCSTAT	ALAND	AWATER	INTPTLAT	INTPTLON	tmax_mean
1		A	1479085972	5223780	+42.4703284	-091.8386658	11.16441572616...
2		A	6843332007	103024638	+47.6296277	-105.7572220	10.63372755050...
3		S	59497122355	13774270048	+64.7836858	-164.1889119	
4		A	27540342238	9802713	+37.6346050	-114.8630367	17.70877057855...
5		A	477412565	5295410	+33.8341247	-083.4377284	27.48456270450...
6		A	2845915561	12391036	+48.7760403	-096.7803493	5.862159251954...
7		A	4506292267	24021150	+43.2216189	-103.5126087	5.457298366097...
8	4	A	848808625	22435476	+41.8520581	-088.0860383	14.67274596113...
9		A	1748233047	6928008	+43.0353254	-073.1114603	10.75016458982...
10		A	1352887109	1598842825	+43.4567309	-078.7921425	15.10238790512...
11	4	A	476088156	17572423	+39.9166853	-075.3988178	15.68195986528...
12		A	2653593197	43823568	+48.6821831	-099.2481583	7.654798269271...
13	4	A	602910975	35355101	+40.9596985	-074.0747272	13.17613764652...
14		A	1051393875	17332960	+39.1969272	-084.5441868	27.33687025088...
15		A	2162178923	58072194	+35.7898464	-078.6506240	29.08486454050...

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Interpolating Point Data

Interpolation is a commonly used GIS technique to create continuous surface from discrete points. A lot of real world phenomena are continuous - elevations, soils, temperatures etc. If we wanted to model these surfaces for analysis, it is impossible to take measurements throughout the surface. Hence, the field measurements are taken at various points along the surface and the intermediate values are inferred by a process called 'interpolation'. In QGIS, interpolation is achieved using the built-in `Interpolation plugin` .

Overview of the task

We will take field depth measurements for a Lake Arlington in Texas and create an elevation relief map and contours from these measurements.

Other skills you will learn

- Creating contours from point data.
- Masking no-data values from a raster layer.
- Adding labels to a vector layer.

Get the data

Texas Water Development Board provides the shapefiles for completed lake surveys (http://www.twdb.texas.gov/surfacewater/surveys/completed/list/index.asp).

Download the 2007-12 survey shapefiles for Lake Arlington (http://www.twdb.texas.gov/hydro_survey/Arlington/2007-12/Shapefiles.zip).

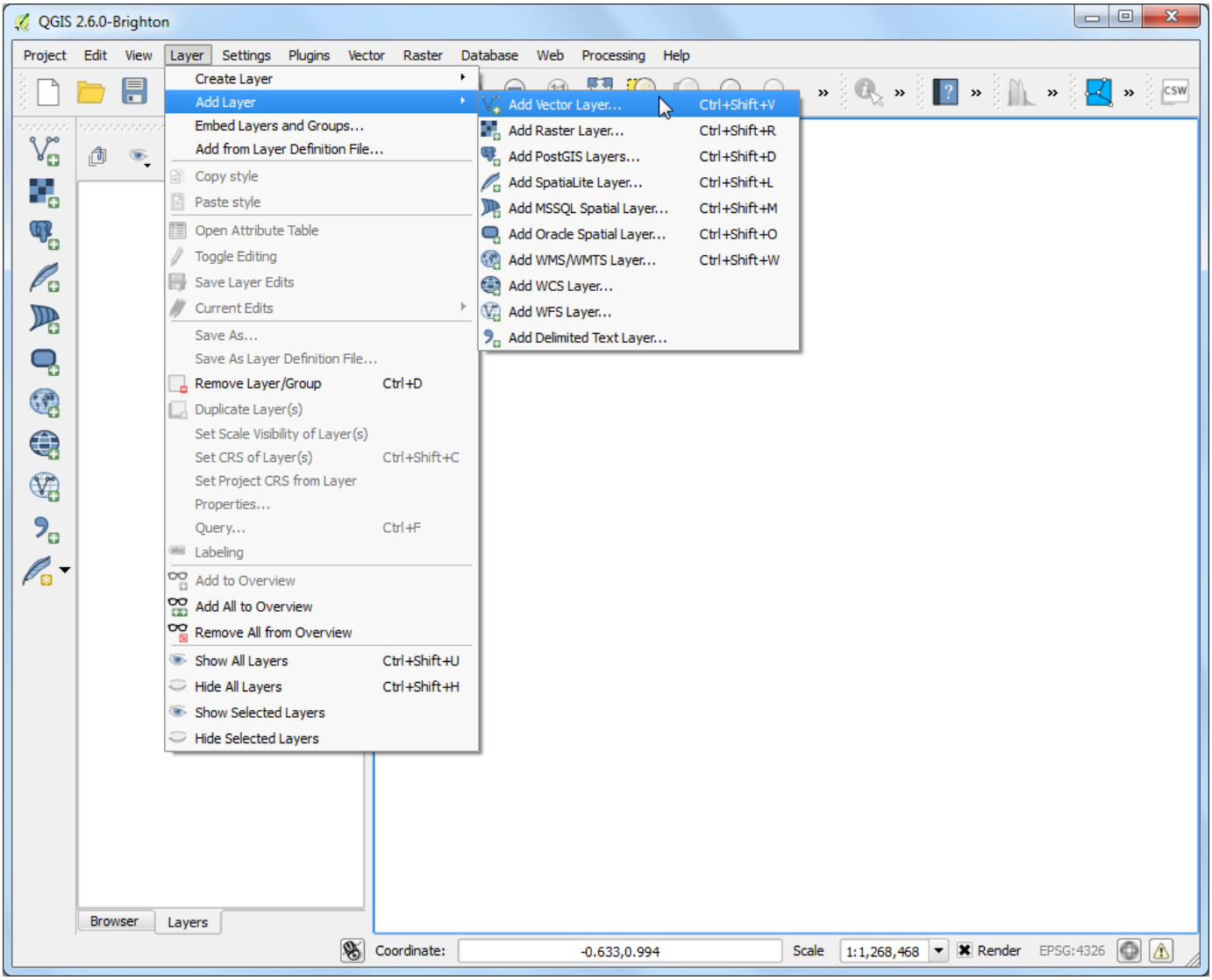
For convenience, you can directly download the sample data used in this tutorial from link below.

Shapefiles.zip (http://www.qgistutorials.com/downloads/Shapefiles.zip)

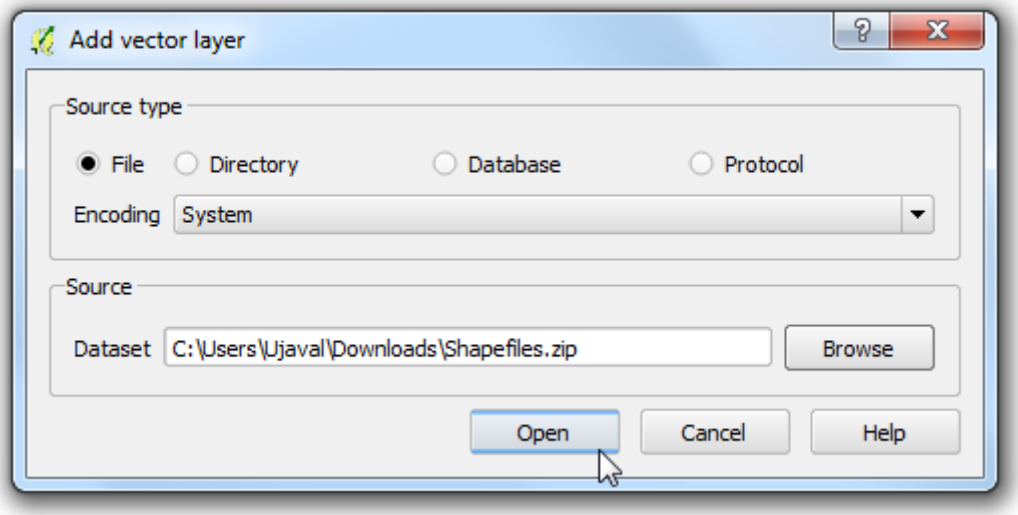
Data Sources: [TWDB] (credits.html#twdb)

Procedure

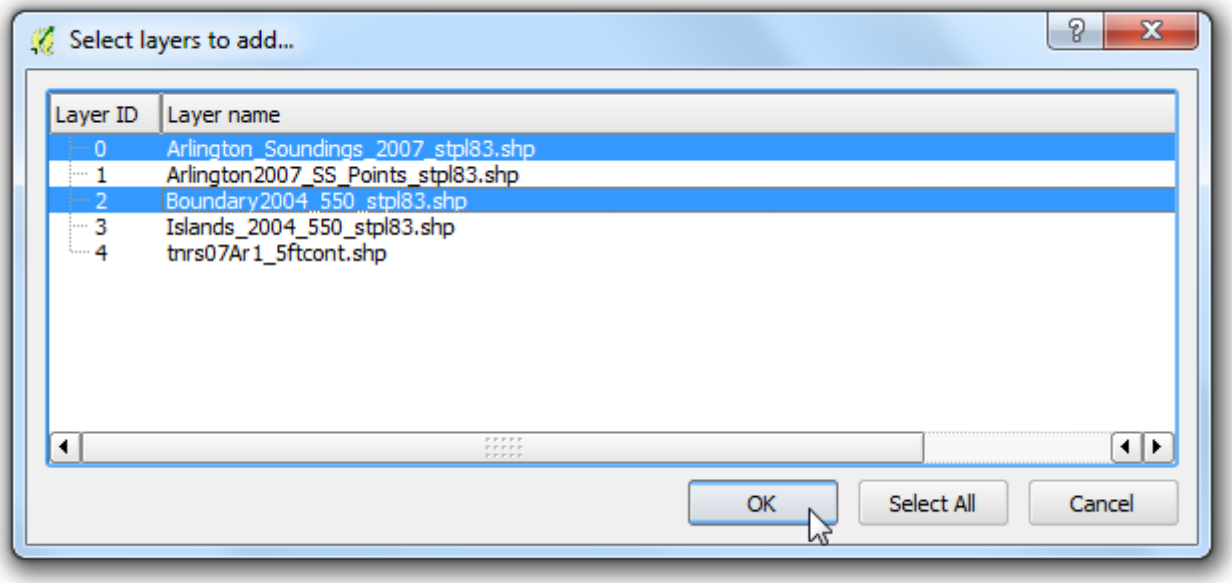
1. Open QGIS. Go to Layer ▸ Add Layer ▸ Add Vector Layer..



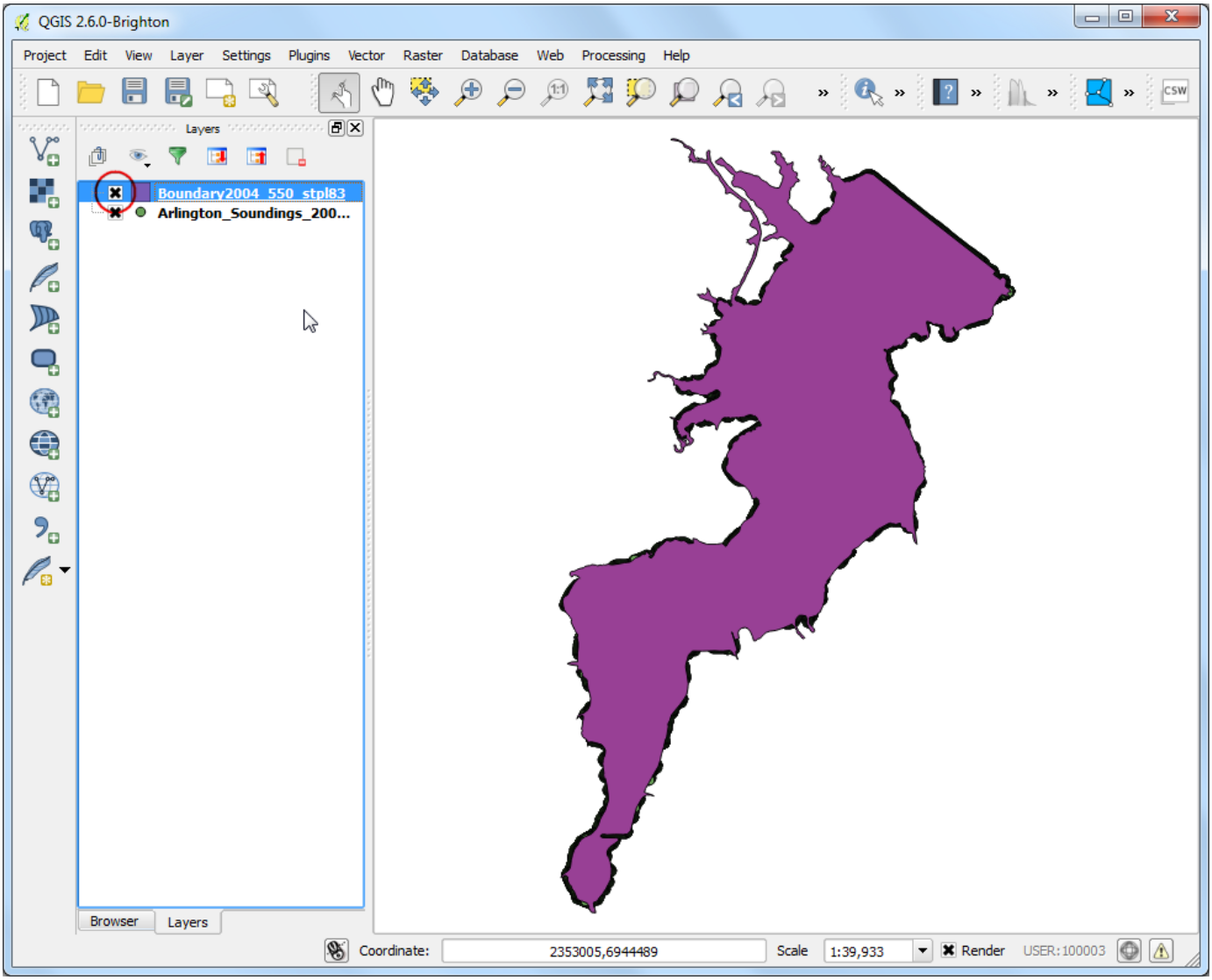
2. Browse to the downloaded `Shapefiles.zip` file and select it. Click Open.



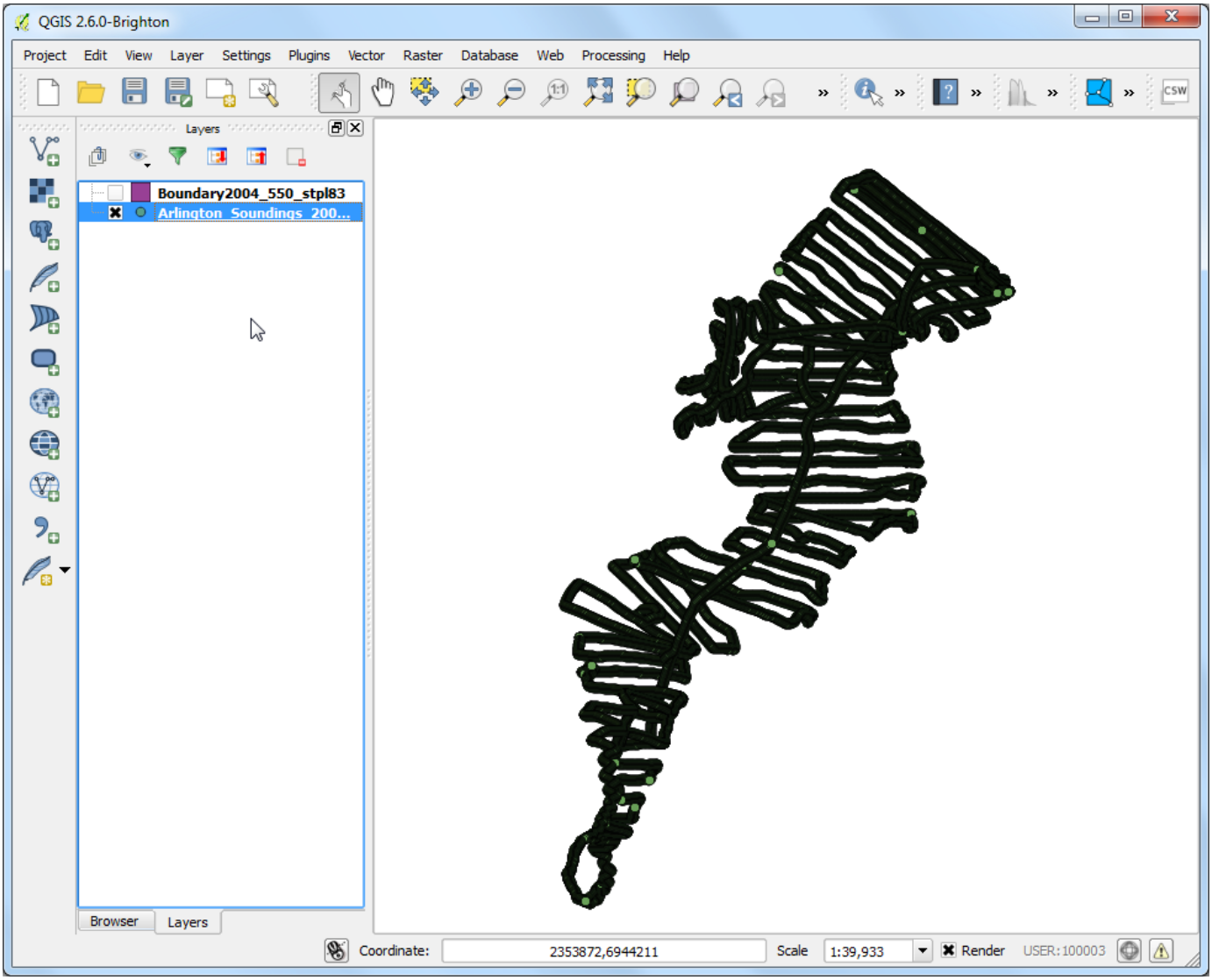
3. In the Select layers to add... dialog, hold the `Shift` key and select `Arlington_Soundings_2007_stpl83.shp` and `Boundary2004_550_stpl83.shp` layers. Click OK.



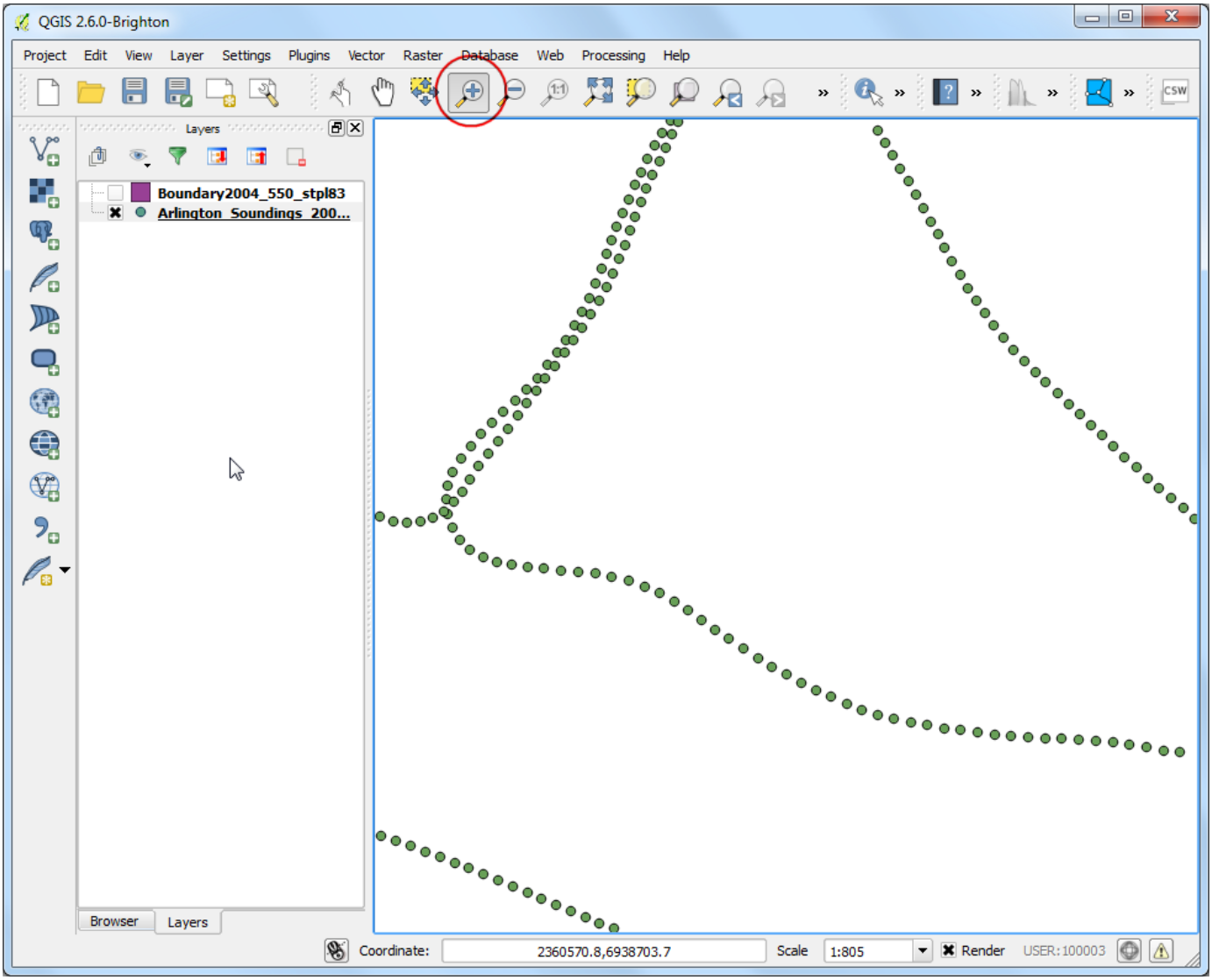
4. You will see the 2 layers loaded in QGIS. The `Boundary2004_550_stpl83` layer represents the boundary of the lake. Un-check the box next to it in the Table of Contents.



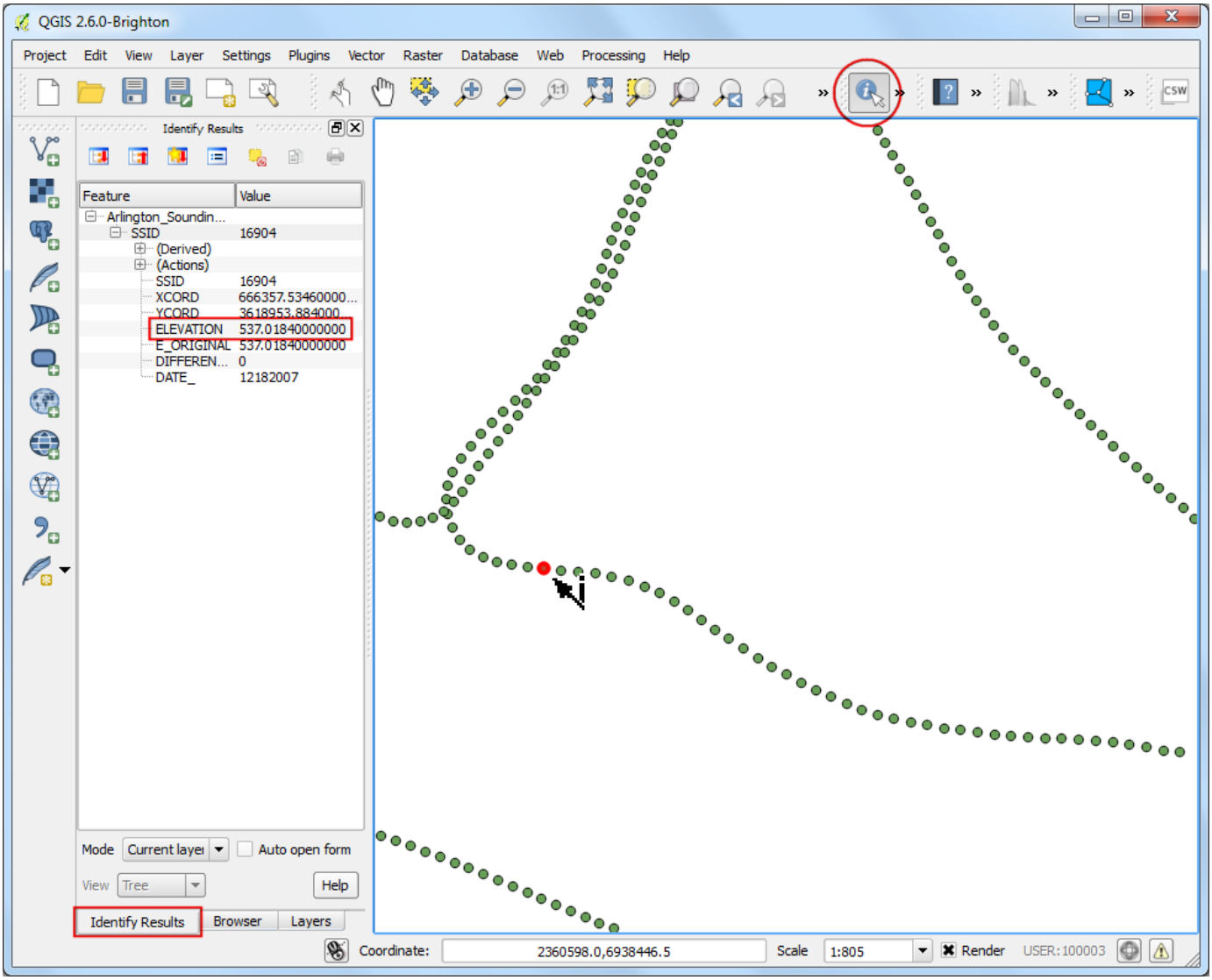
5. This will reveal the data from the second layer `Arlington_Soundings_2007_stpl83` . Though the data looks like lines, it is a series of points that are very close.



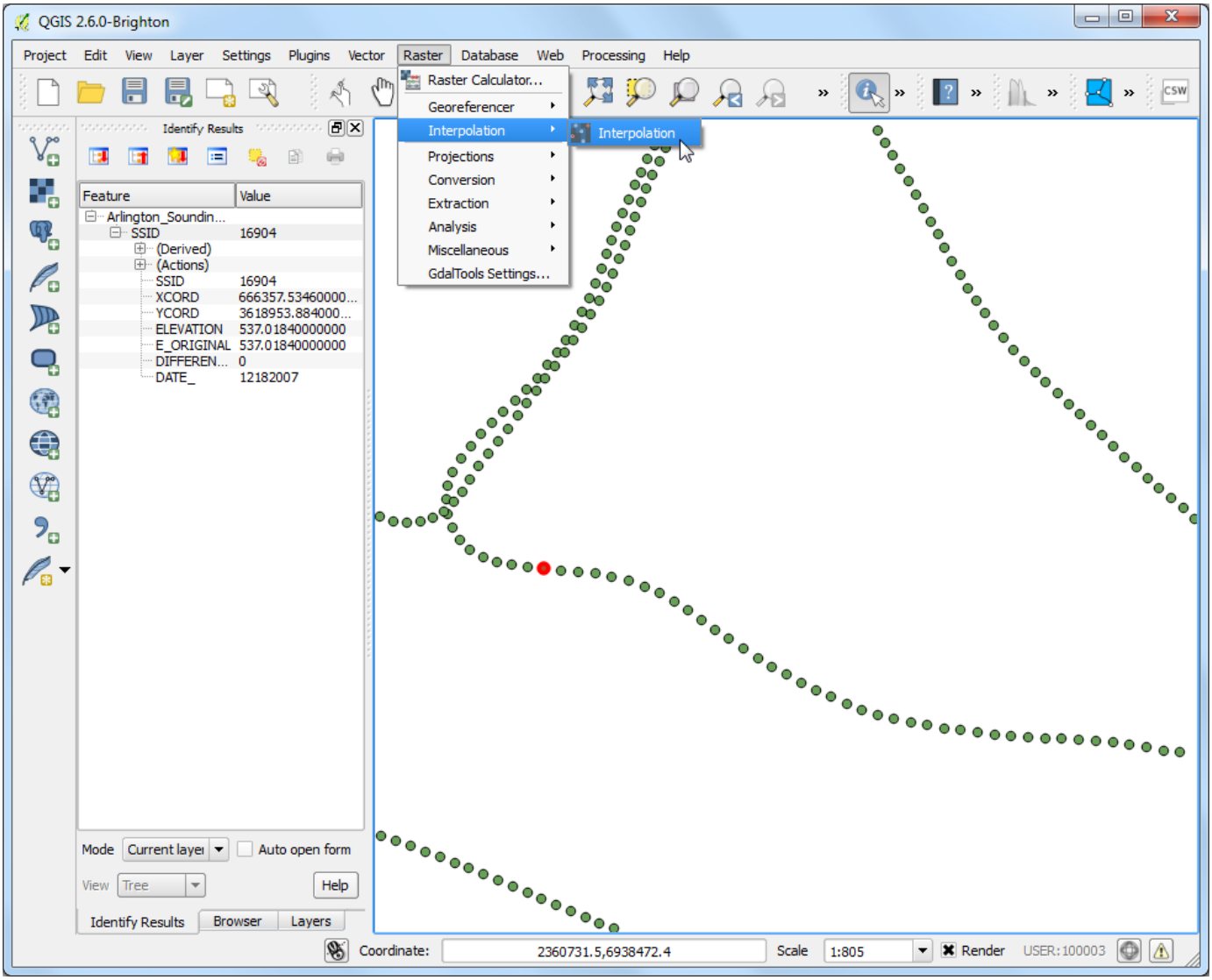
6. Click the Zoom icon and select a small area on the screen. As you zoom closer, you will see the points. Each point represents a reading taken by a *Depth Sounder* at the location recorded by a *DGPS* equipment.



7. Select the Identify tool and click on a point. You will see the Identify Results panel show up on the left with the attribute value of the point. In this case, the `ELEVATION` attribute contains the depth of the lake at the location. As our task is to create a depth profile and elevation contours, we will use this values as input for the interpolation.



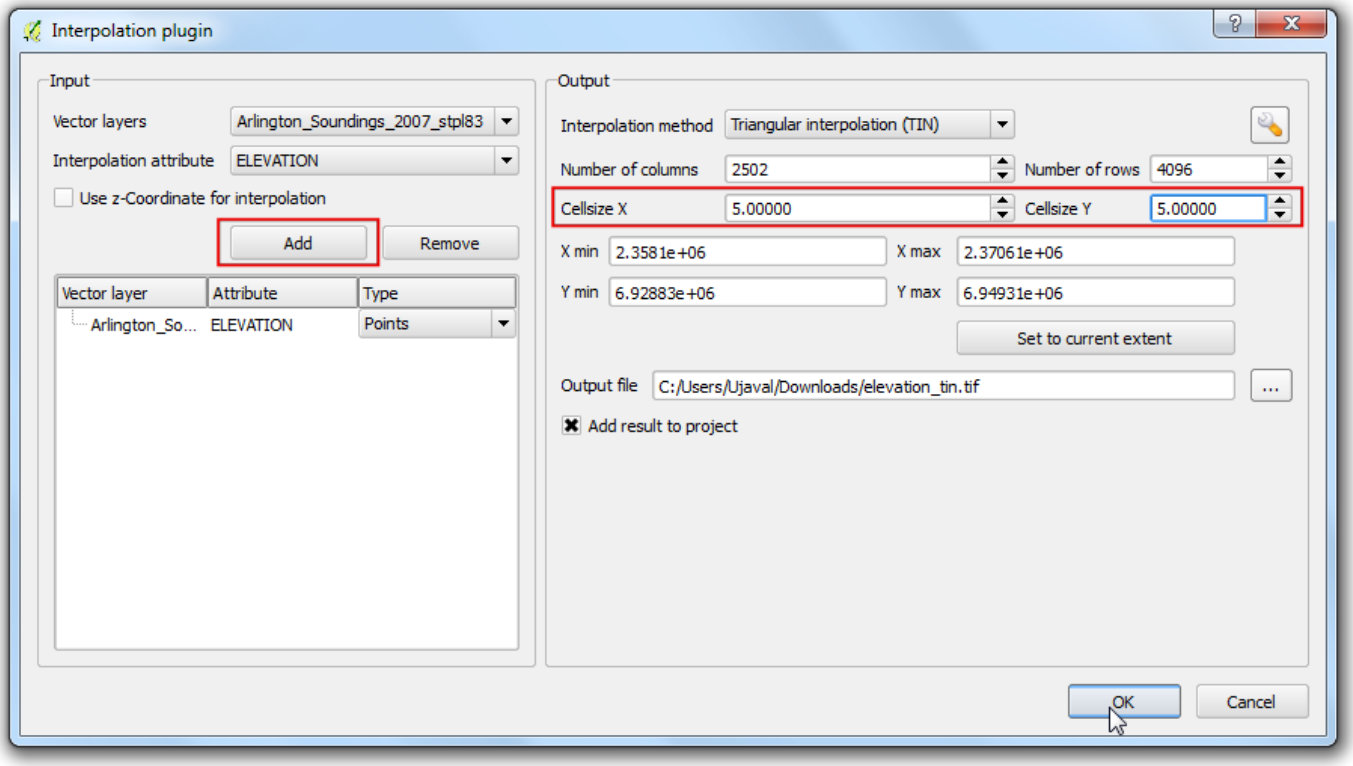
8. Make sure you have the `Interpolation plugin` enabled. See *Using Plugins* (using_plugins.html) for how to enable plugins. Once enabled, go to Raster ▸ Interpolation ▸ Interpolation.



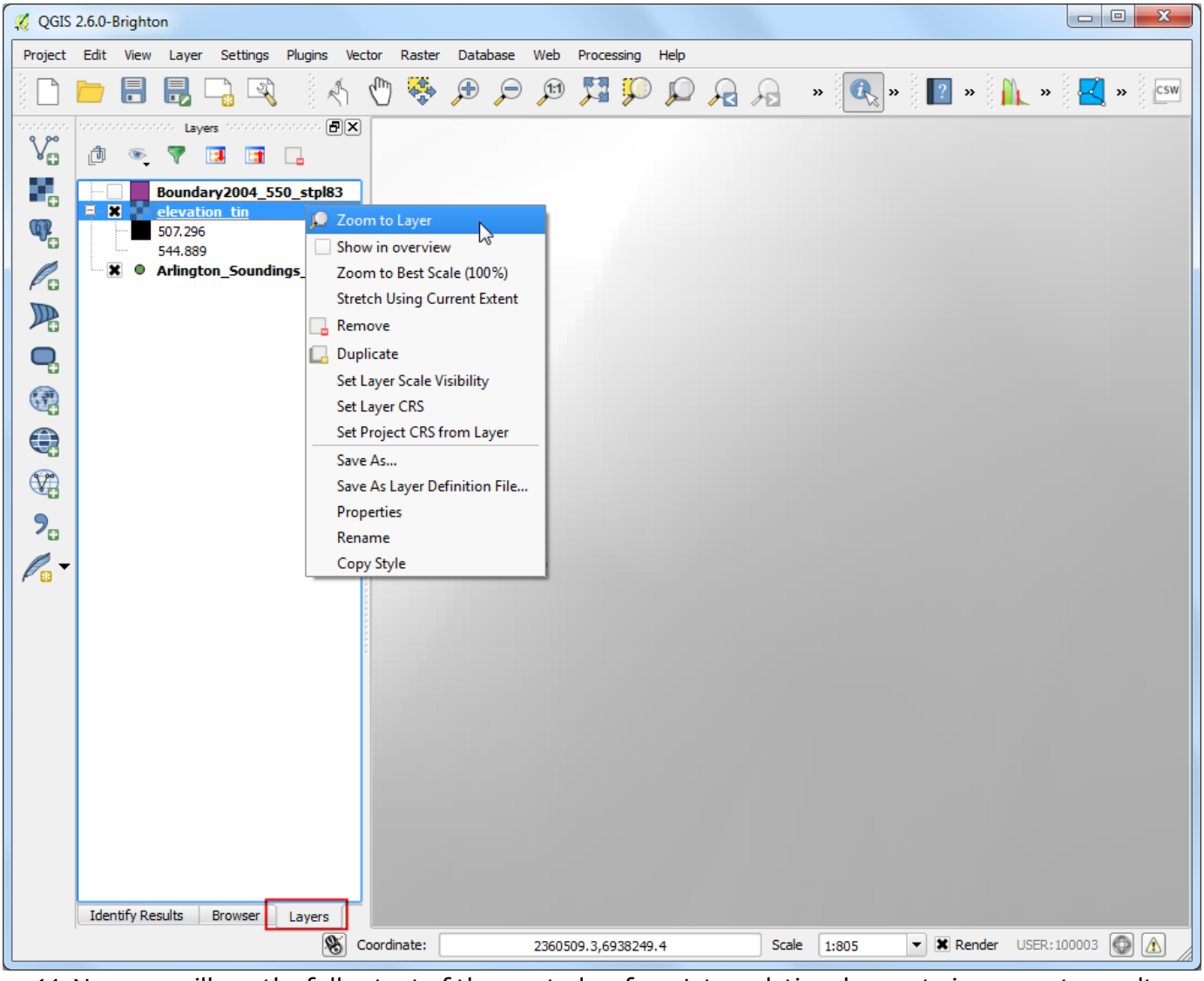
9. In the Interpolation dialog, select `Arlington_Soundings_2007_stpl83` as the Vector layers in the Input panel. Select `ELEVATION` as the Interpolation attribute. Click Add. Change the Cellsize X and Cellsize Y values to `5`. This value is the size of each pixel in the output grid. Since our source data is in a projected CRS with **Feet-US** as units, based on our selection, the grid size will be **5 feet**. Click on the ... button next to Output file and name the output file as `elevation_tin.tif`. CLick OK.

Note

Interpolation results can vary significantly based on the method and parameters you choose. QGIS interpolation supports *Triagulated Irregular Network (TIN)* and *Inverse Distance Weighting (IDW)* methods for interpolation. TIN method is commonly used for elevation data whereas IDW method is used for interpolating other types of data such as mineral concentrations, populations etc. See the Spatial Analysis (http://docs.qgis.org/2.2/en/docs/gentle_gis_introduction/spatial_analysis_interpolation.html) module of the QGIS documentation for more details.

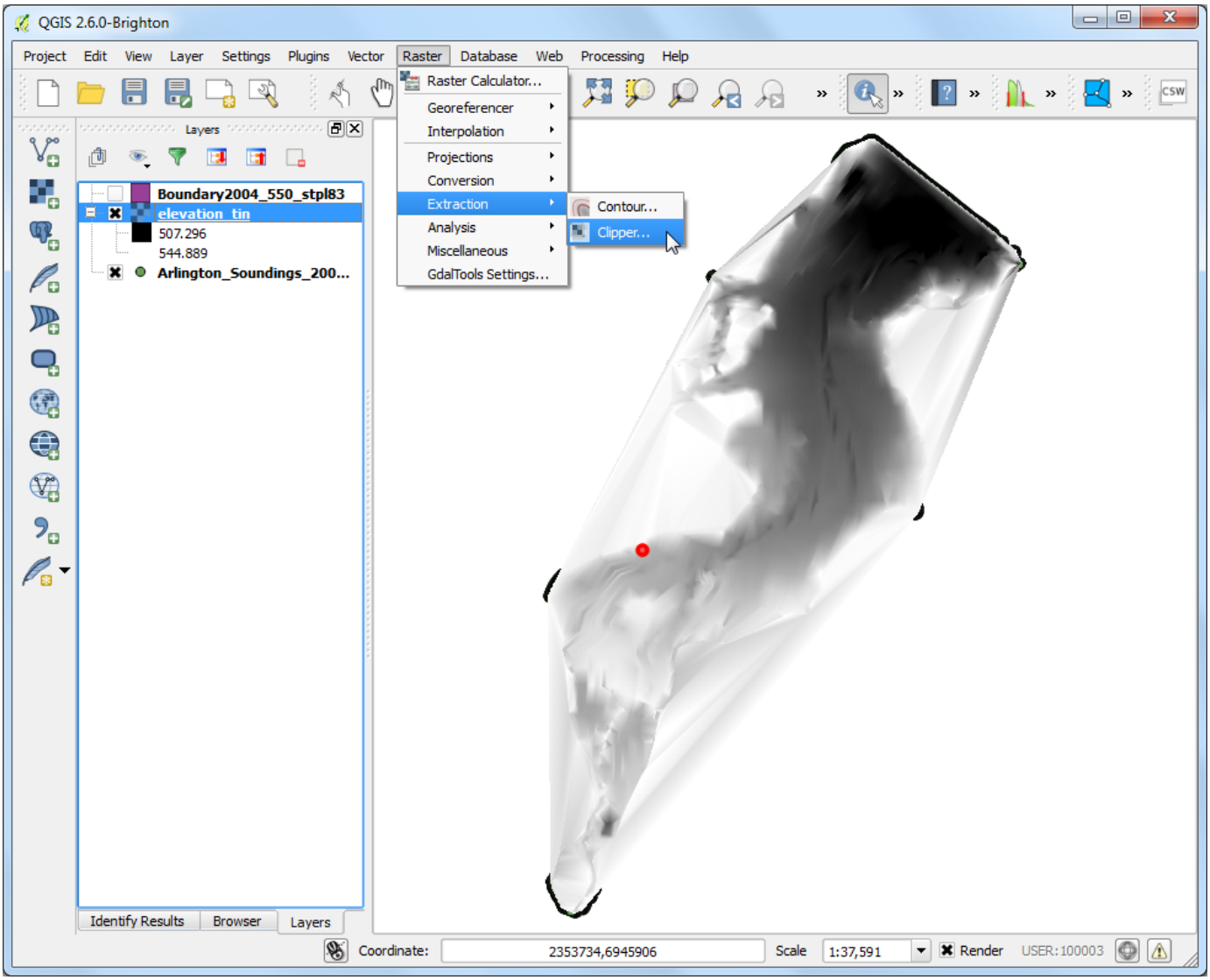


10. You will see the new later `elevation_tin` loaded in QGIS. Right-click the layer and select Zoom to layer.

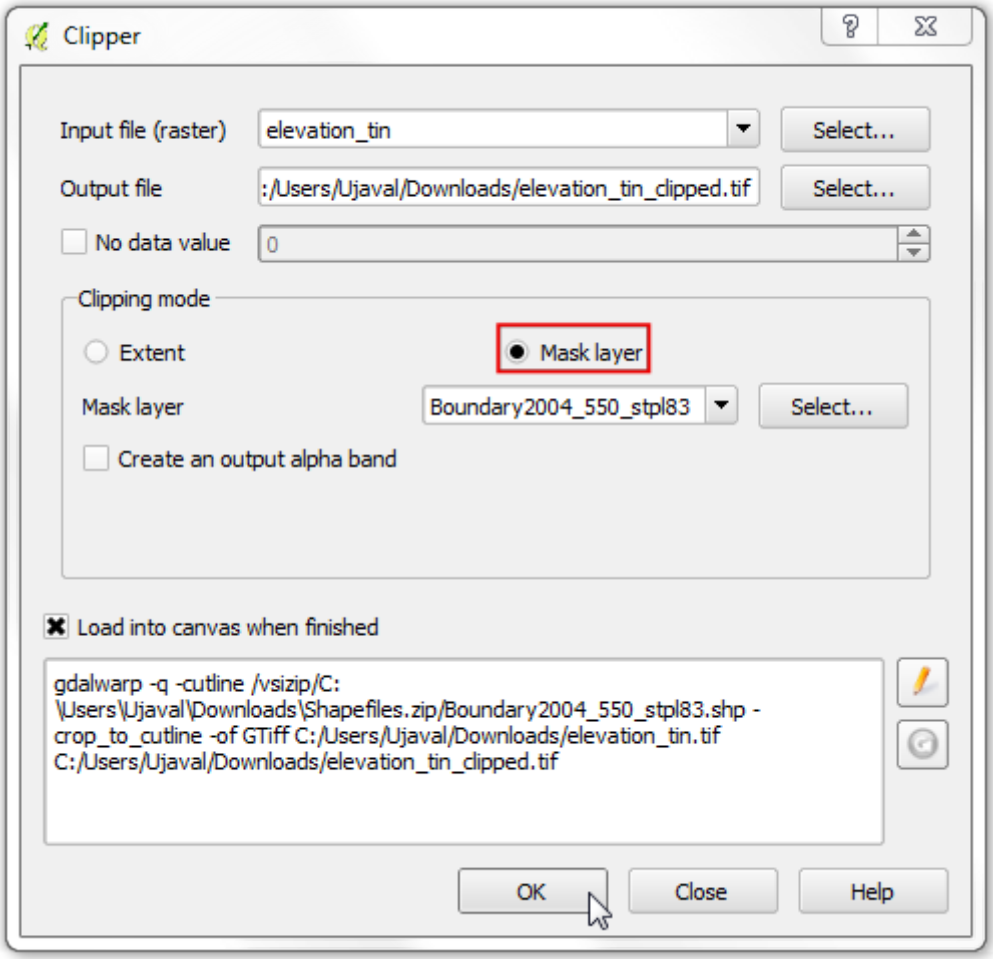


11. Now you will see the full extent of the created surface. Interpolation does not give accurate results outside the collection area. Let's clip the resulting surface with the lake boundary. Go to Raster ▸

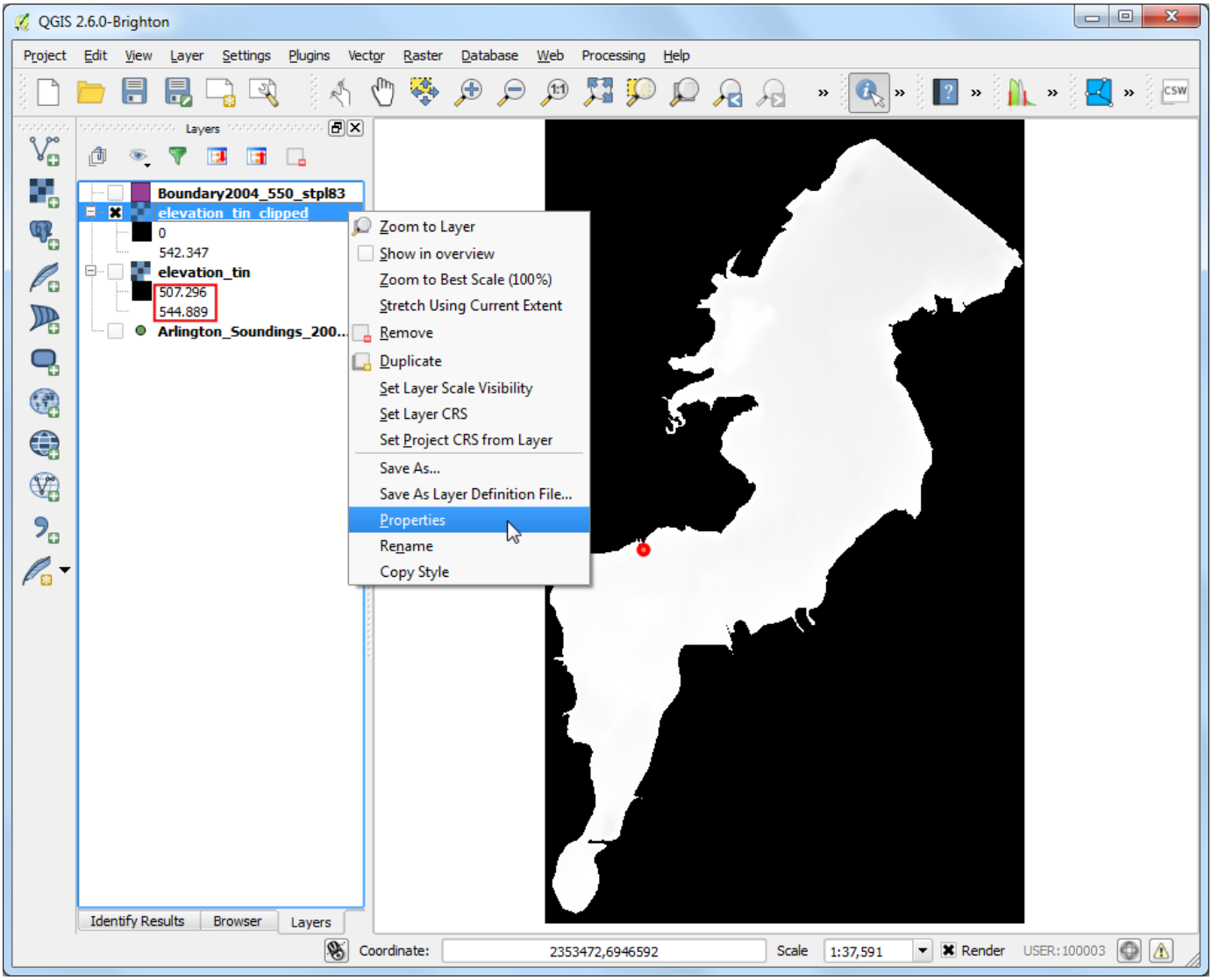
Extraction ▸ Clipper.



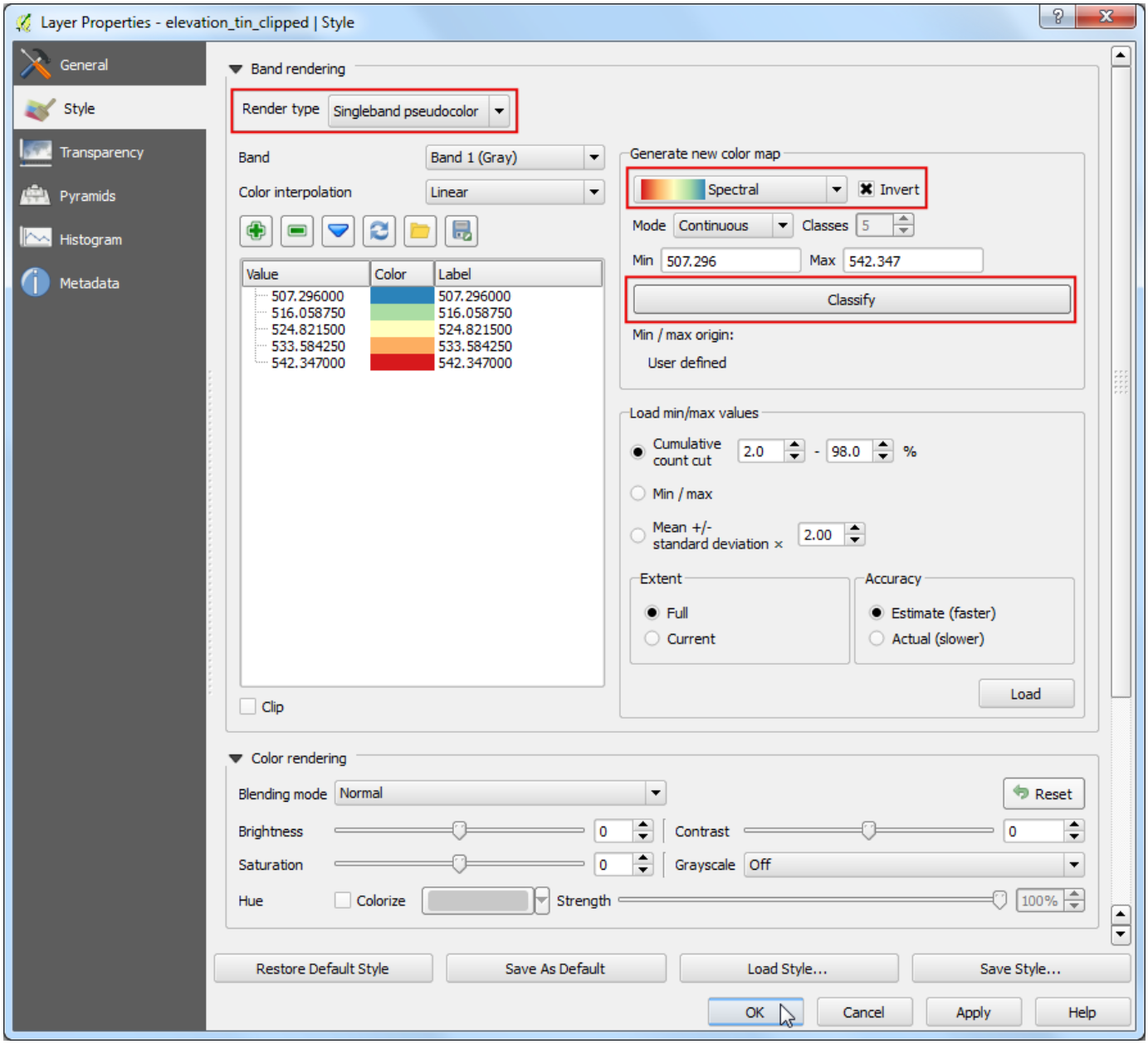
12. Name the Output file as `elevation_tin_clipped.tif` . Select the Cliiped mode as Mask layer. Select `Boundary2004_550_stpl83` as the Mask layer`. Click OK.



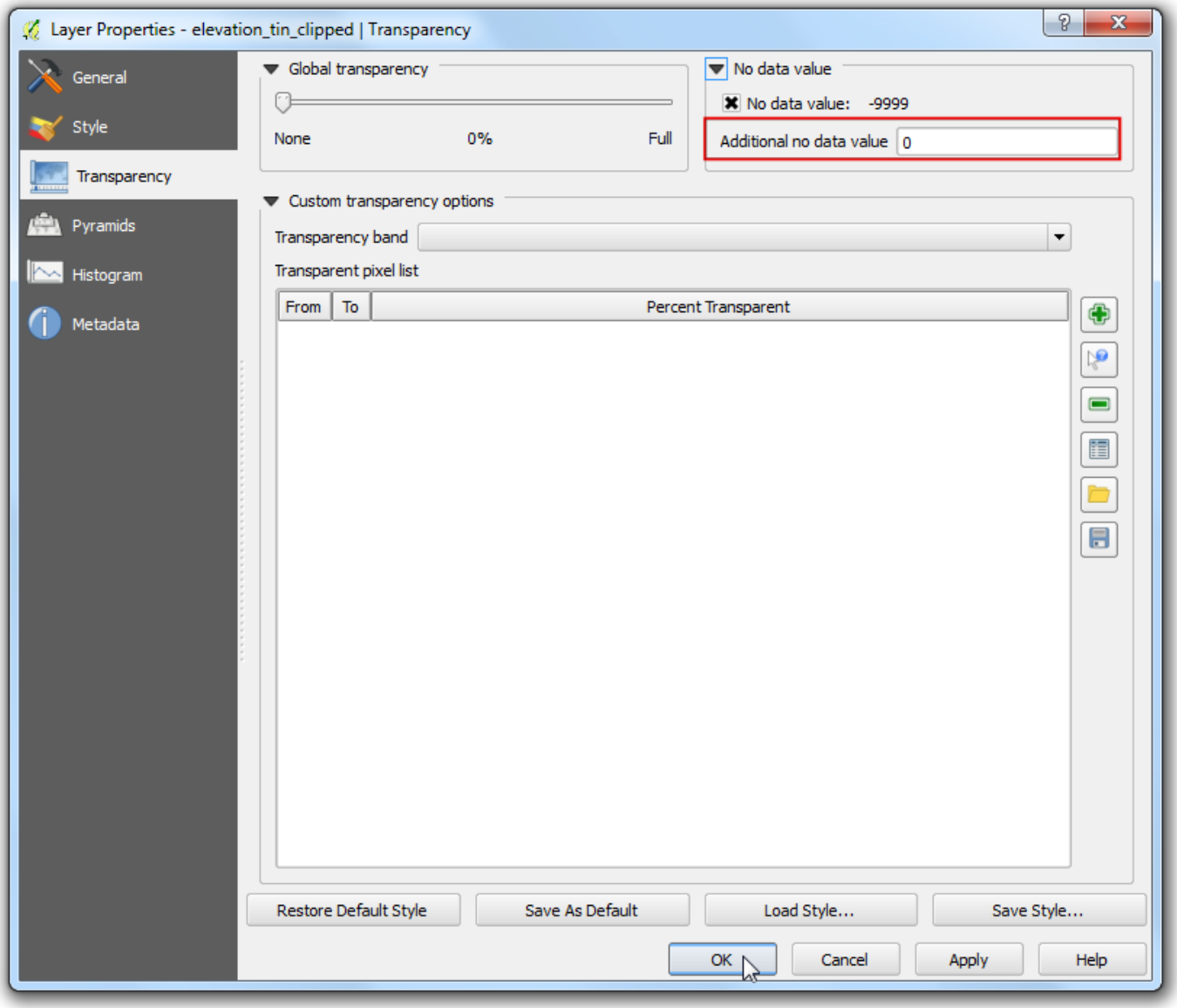
13. A new raster `elevation_tin_clipped` will be loaded in QGIS. We will now style this layer to show the difference in elevations. Note the min and max elevation values from the `elevation_tin` layer. Right-click the `elevation_tin_clipped` layer and select Properties.



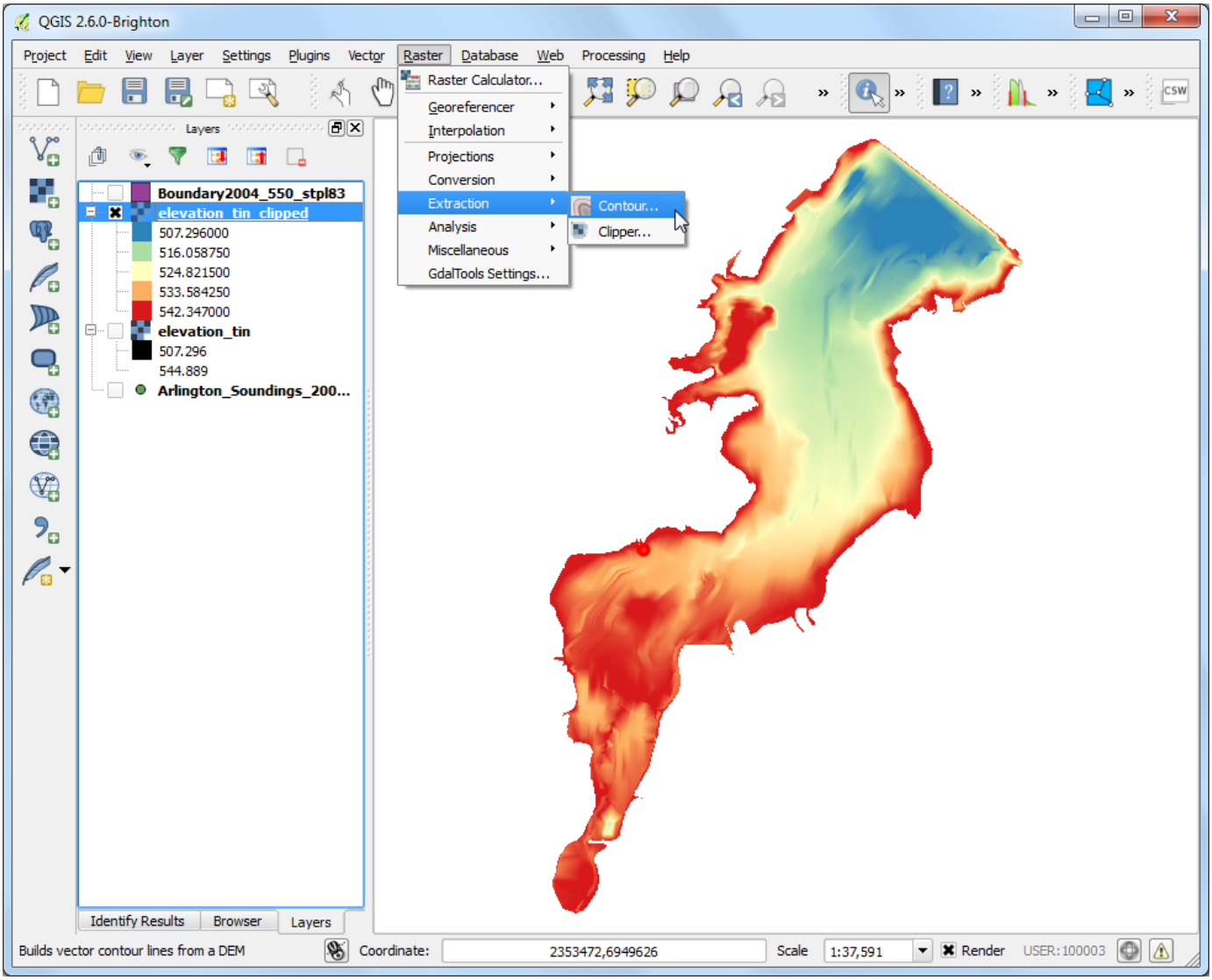
14. Go to the Style tab. Select Render type as `Singleband pseudocolor` . In the Generate new color map panel, select `Spectral` color ramp. As we want to create a depth-map as opposed to a height-map, check the Invert box. This will assign blues to deep areas and reds to shallow areas. Click Classify.



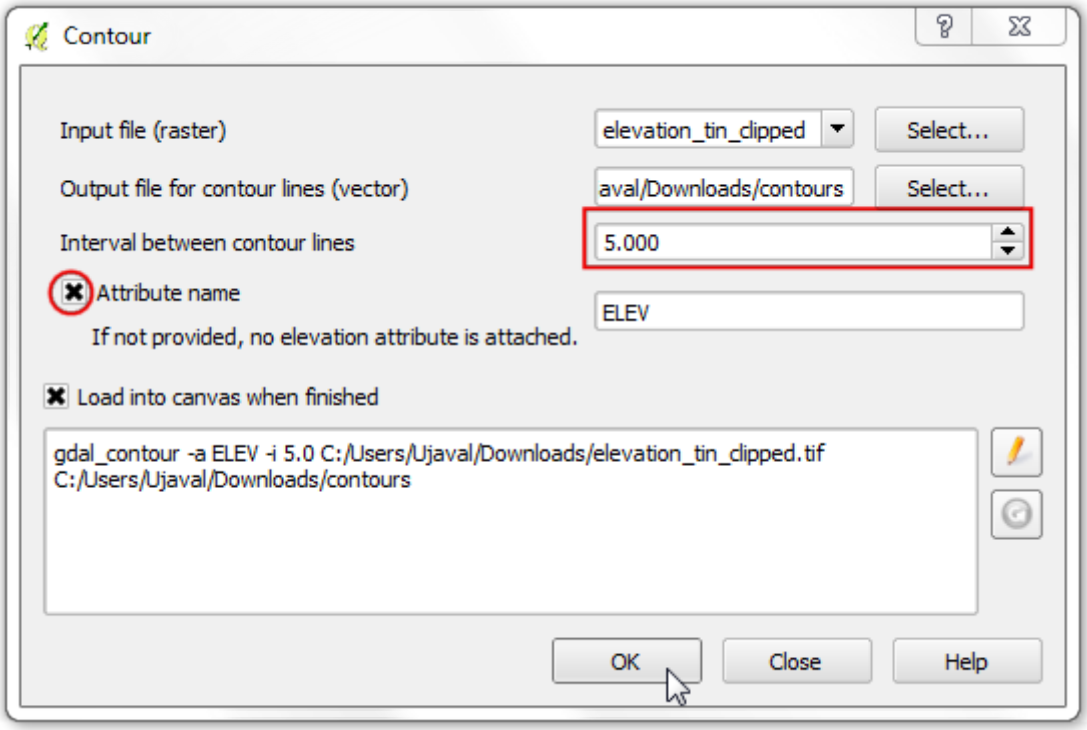
15. Switch to the Tranparency tab. We want to remove the black-pixels from our output. Enter `0` as the Additional no data value. Click OK.



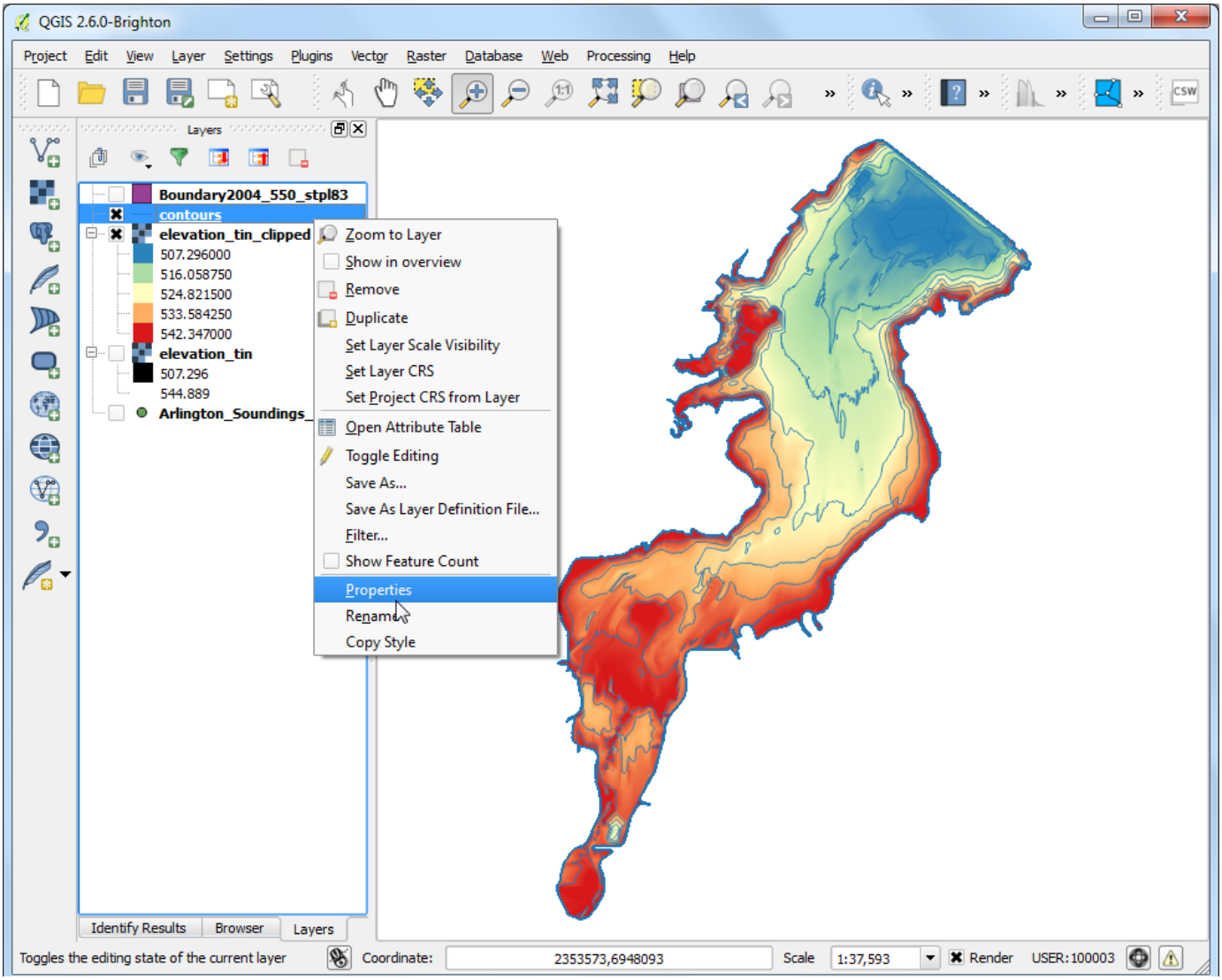
16. Now you have a elevation relief map for the lake generated from the individual depth readings. Let's generate contours now. Go to Raster ▸ Extraction ▸ Contours.



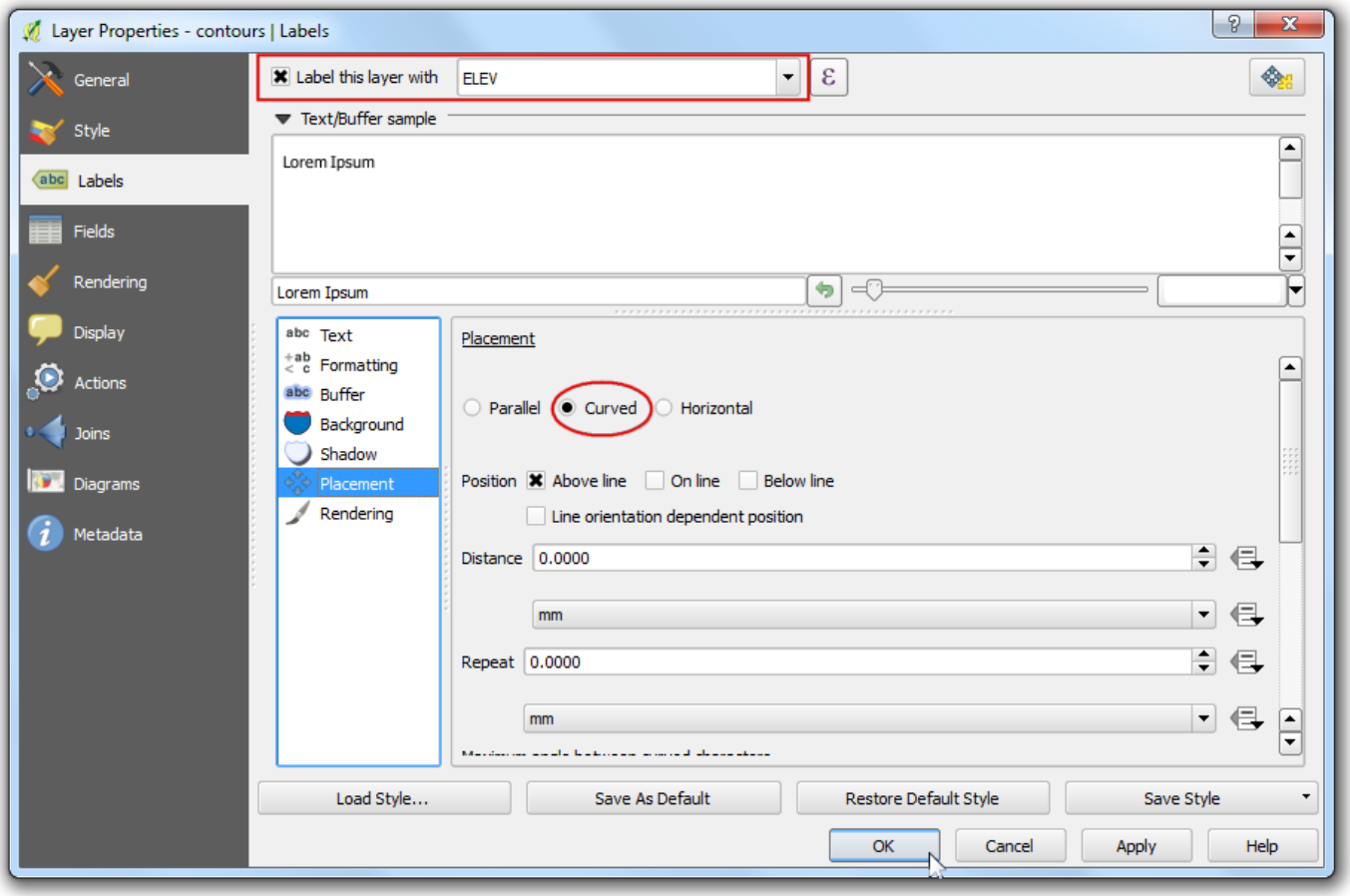
17. In the Contour dialog, enter `contours` as the Output file for contour lines. We will generate contour lines at 5ft intervals, so enter `5.00` as the Interval between contour lines. Check the Attribute name box. Click OK.



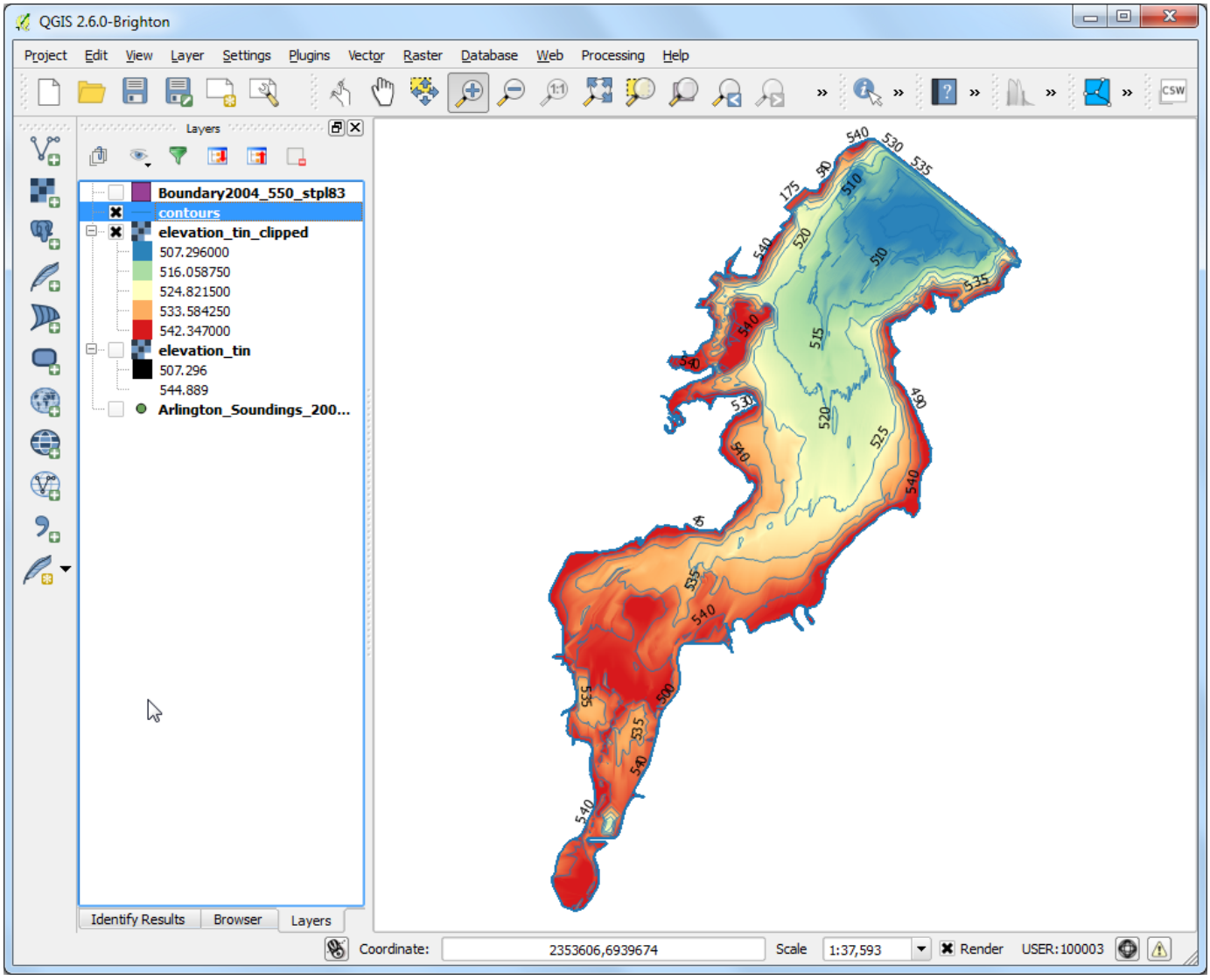
18. The contour lines will be loaded as `contours` layer once the processing is finished. Right-click the layer and select Properties.



19. Go to the Labels tab. Check the Label this layer with box and select `ELEV` as the field. Select `Curved` as the Placement type and click OK.



20. You will see that each contour line will be appropriately labeled with the elevation along the line.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Batch Processing using Processing Framework (QGIS3)

The *Processing Framework* in QGIS provides an environment within QGIS to run native and third-party algorithms for processing data. It contains a nice batch processing interface that allows one to execute an algorithm on several layers easily. Batch processing is a useful tool that can save manual effort and help you automate repetitive tasks.

Overview of the task

We will take several global vector layers and clip them to the extent of Africa in a single batch command.

Other skills you will learn

- Create a **Filter** to remove unwanted features from a layer without creating a new layer.
- Merge multiple layers into a single Geopackage file.

Get the data

Natural Earth (http://naturalearthdata.com) has several global vector layers. Download the following layers

- Admin 0 - Countries (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_admin_0_countries.zip)
- Railroads (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_railroads.zip)
- Ports (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_ports.zip)
- Urban Areas (https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_urban_areas.zip)

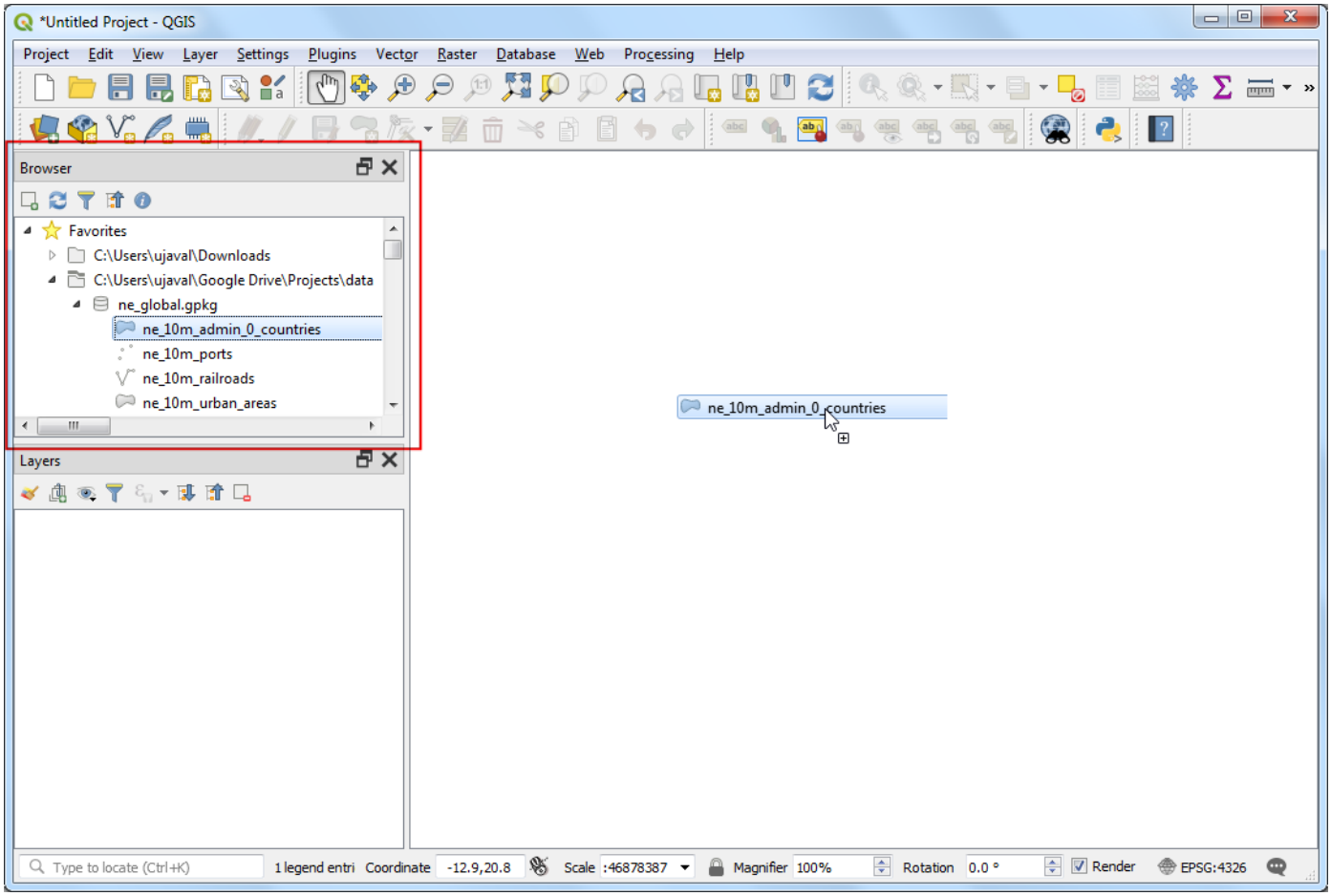
For convenience, you may directly download a geopackage containing the above layers from below:

ne_global.gpkg (http://www.qgistutorials.com/downloads/ne_global.gpkg)

Data Source: [NATURALEARTH] (../credits.html#naturalearth)

Procedure

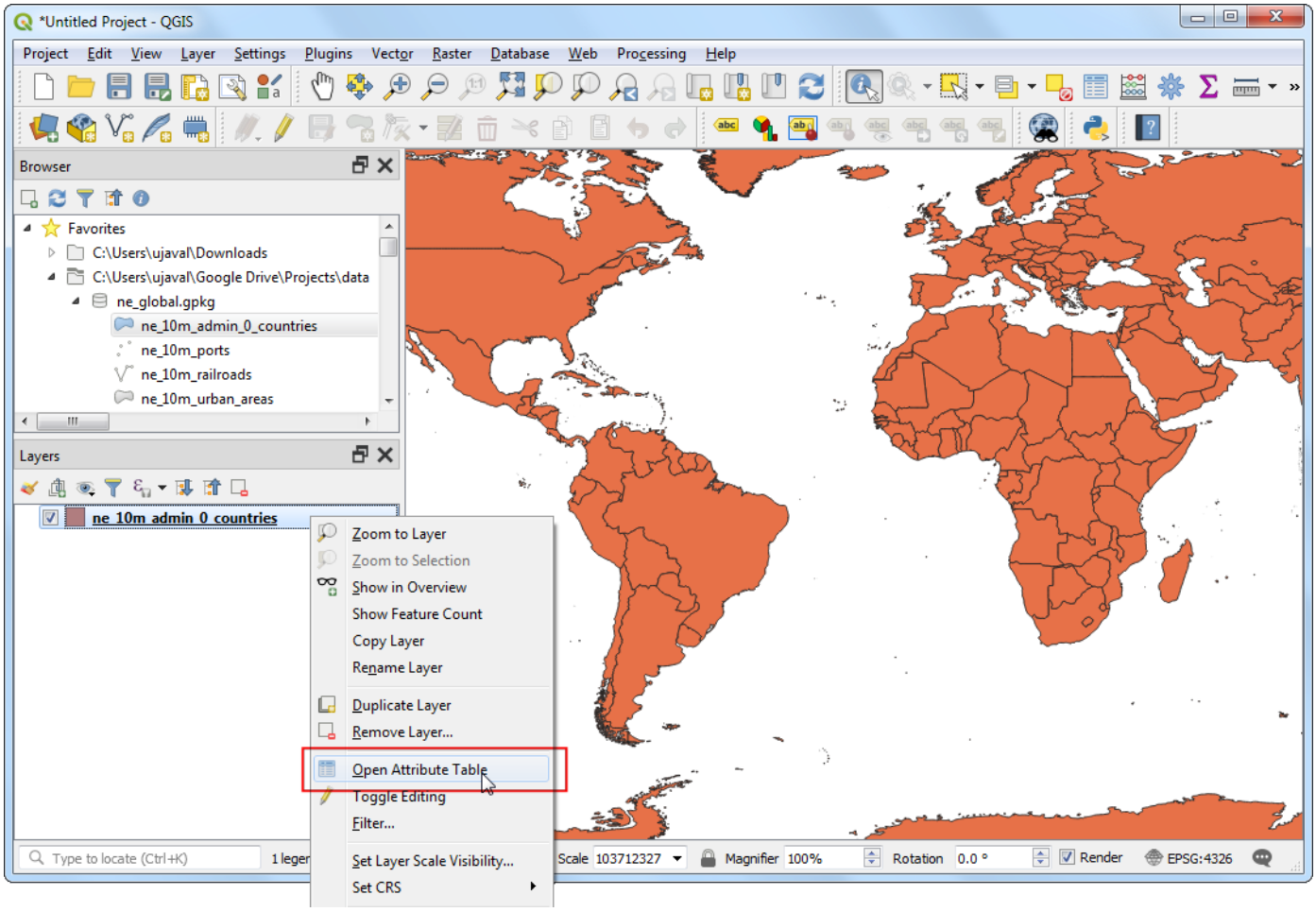
1. In the QGIS Browser Panel, locate the directory where you saved your downloaded data. Expand the `zip` or the `gpkg` entry and select the `ne_10m_admin_0_countries` layer. Drag the layer to the canvas.



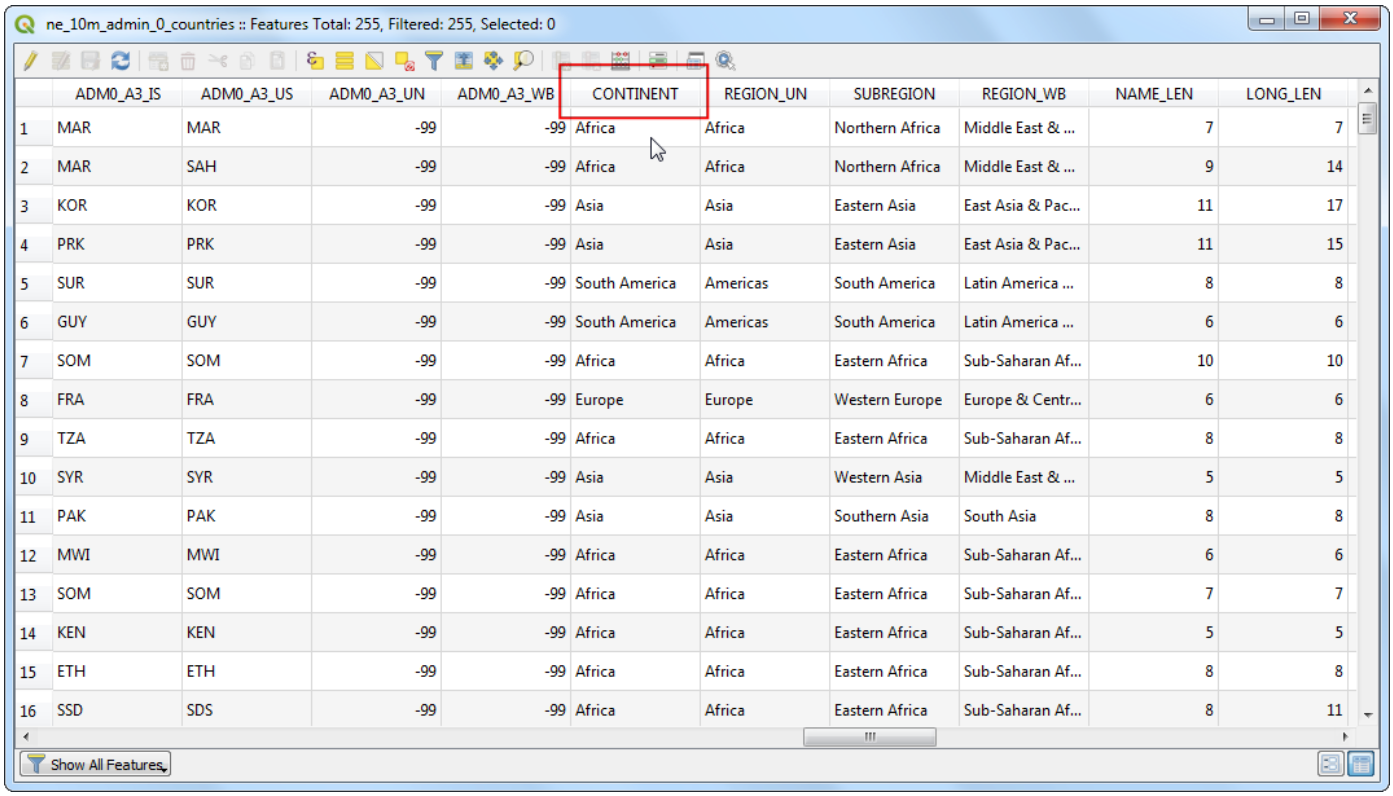
2. You will see the layer loaded in the Layers panel. As our task is to clip the global layers to the boundary of Africa, we need to first prepare a layer containg features only from that continent. Let's look at the attribute table to see what column can be used to query features belonging to a particular continent. Right-click the `ne_10m_admin_0_countries` layer and select Open Attribute Table.

Note

Tip: You can also use the keyboard shortcut `F6` to open the attribute table of the selected layer.

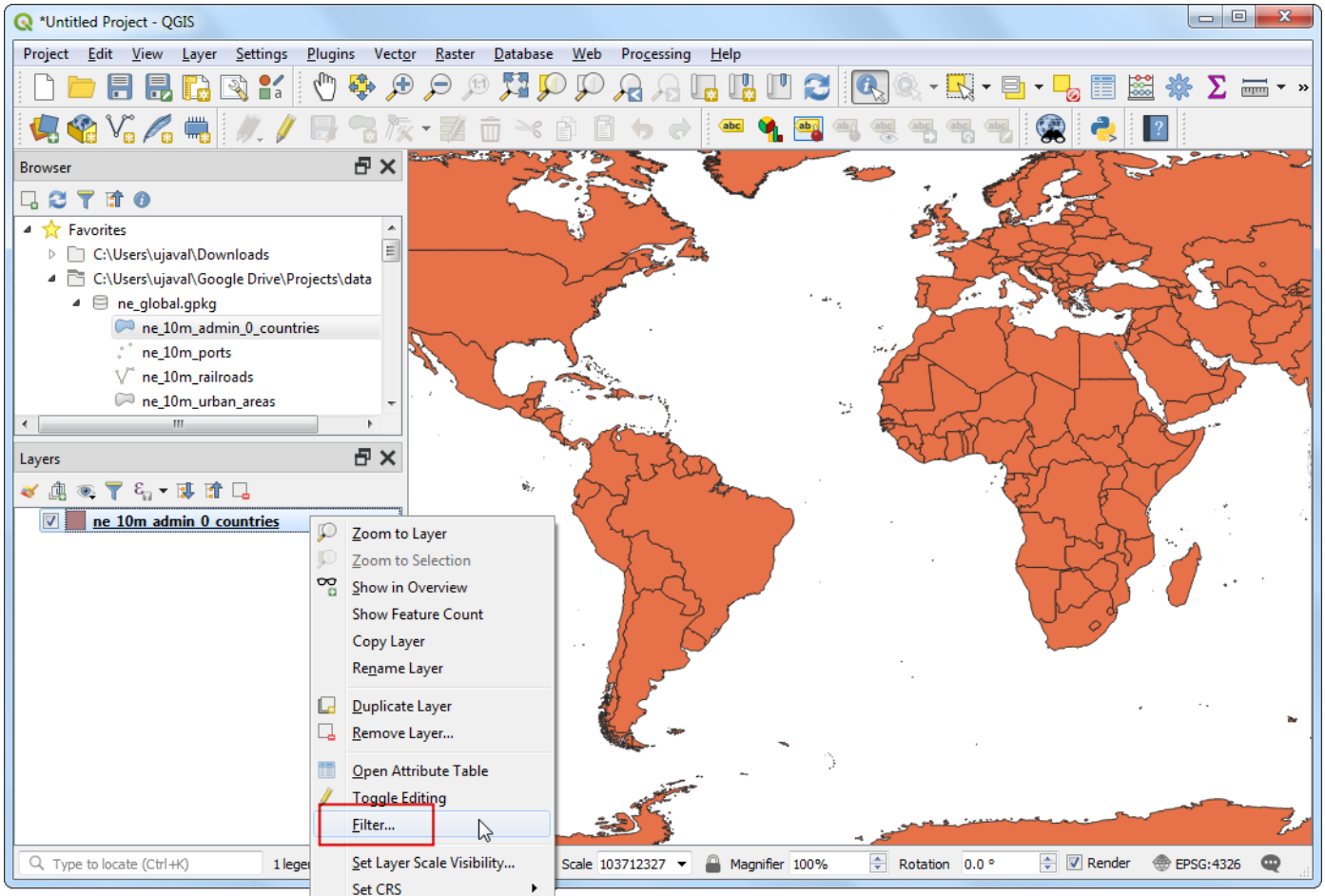


3. In the Attribute Table window, as you scroll horizontally, you will see that the data contains an attribute called **CONTINENT**. We can use this attribute to filter features.

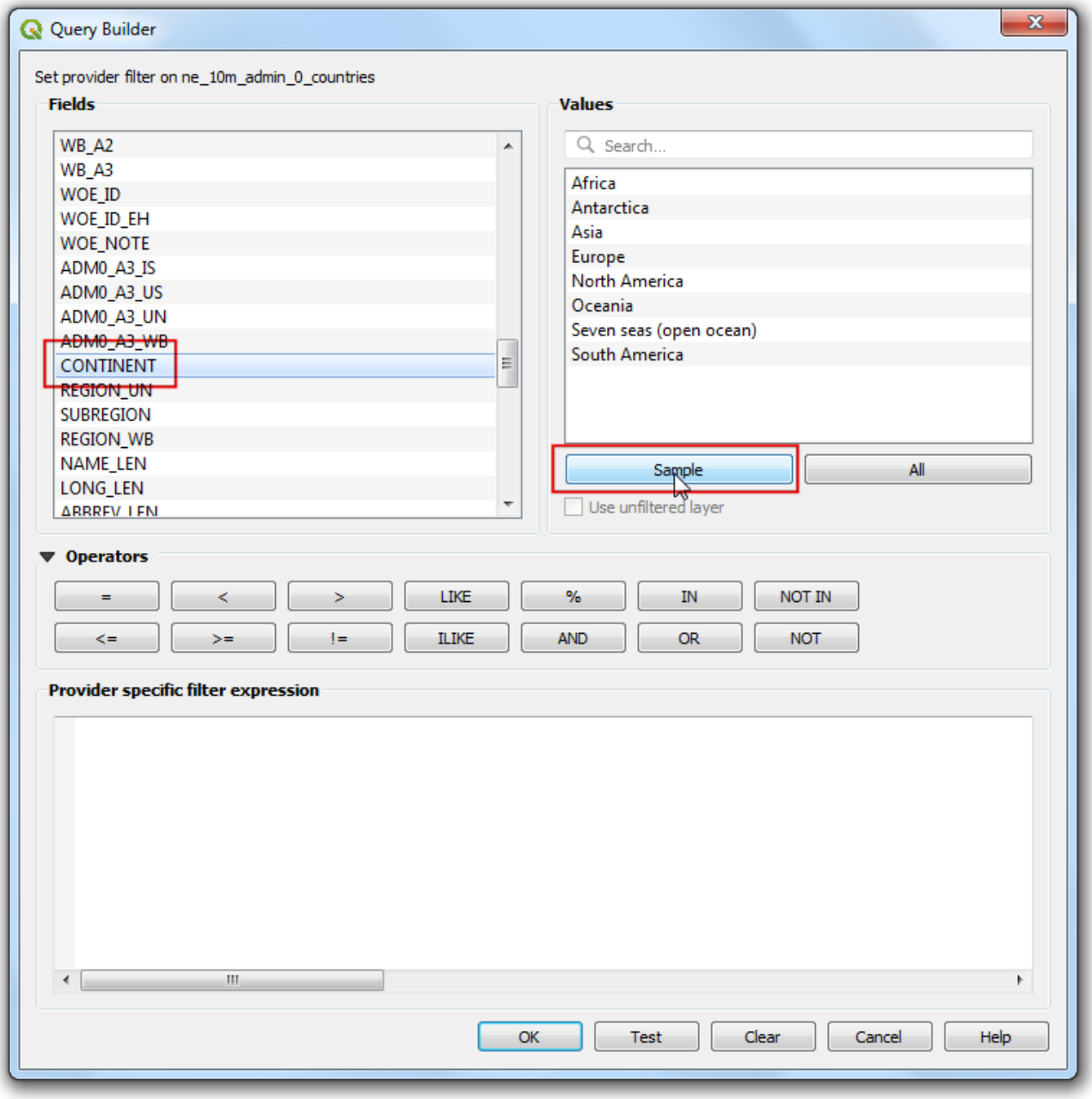


	ADM0_A3_IS	ADM0_A3_US	ADM0_A3_UN	ADM0_A3_WB	CONTINENT	REGION_UN	SUBREGION	REGION_WB	NAME_LEN	LONG_LEN
1	MAR	MAR	-99	-99	Africa	Africa	Northern Africa	Middle East & ...	7	7
2	MAR	SAH	-99	-99	Africa	Africa	Northern Africa	Middle East & ...	9	14
3	KOR	KOR	-99	-99	Asia	Asia	Eastern Asia	East Asia & Pac...	11	17
4	PRK	PRK	-99	-99	Asia	Asia	Eastern Asia	East Asia & Pac...	11	15
5	SUR	SUR	-99	-99	South America	Americas	South America	Latin America ...	8	8
6	GUY	GUY	-99	-99	South America	Americas	South America	Latin America ...	6	6
7	SOM	SOM	-99	-99	Africa	Africa	Eastern Africa	Sub-Saharan Af...	10	10
8	FRA	FRA	-99	-99	Europe	Europe	Western Europe	Europe & Centr...	6	6
9	TZA	TZA	-99	-99	Africa	Africa	Eastern Africa	Sub-Saharan Af...	8	8
10	SYR	SYR	-99	-99	Asia	Asia	Western Asia	Middle East & ...	5	5
11	PAK	PAK	-99	-99	Asia	Asia	Southern Asia	South Asia	8	8
12	MWI	MWI	-99	-99	Africa	Africa	Eastern Africa	Sub-Saharan Af...	6	6
13	SOM	SOM	-99	-99	Africa	Africa	Eastern Africa	Sub-Saharan Af...	7	7
14	KEN	KEN	-99	-99	Africa	Africa	Eastern Africa	Sub-Saharan Af...	5	5
15	ETH	ETH	-99	-99	Africa	Africa	Eastern Africa	Sub-Saharan Af...	8	8
16	SSD	SDS	-99	-99	Africa	Africa	Eastern Africa	Sub-Saharan Af...	8	11

4. Close the attribute table and return to the main QGIS window. Right-click the `ne_10m_admin_0_countries` layer and select Filter.

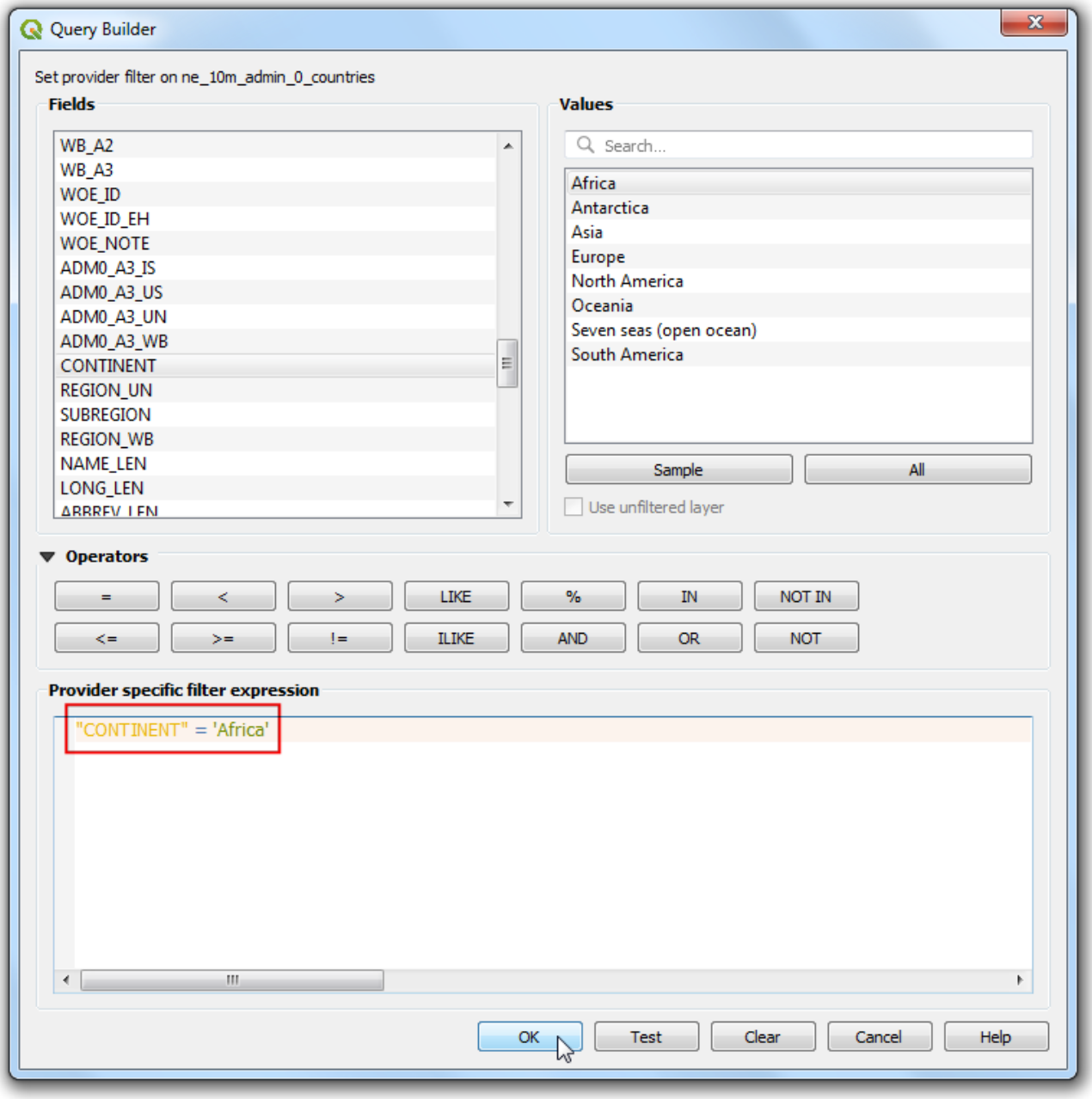


5. In the Query Builder window, select the **CONTINENT** field and click Sample. This will populate the Values panel with the a subset of values of that attribute from features. This step is useful to get an understanding of what type of values are present in your dataset. We can see that our dataset contains contains a value called **Africa** among others.

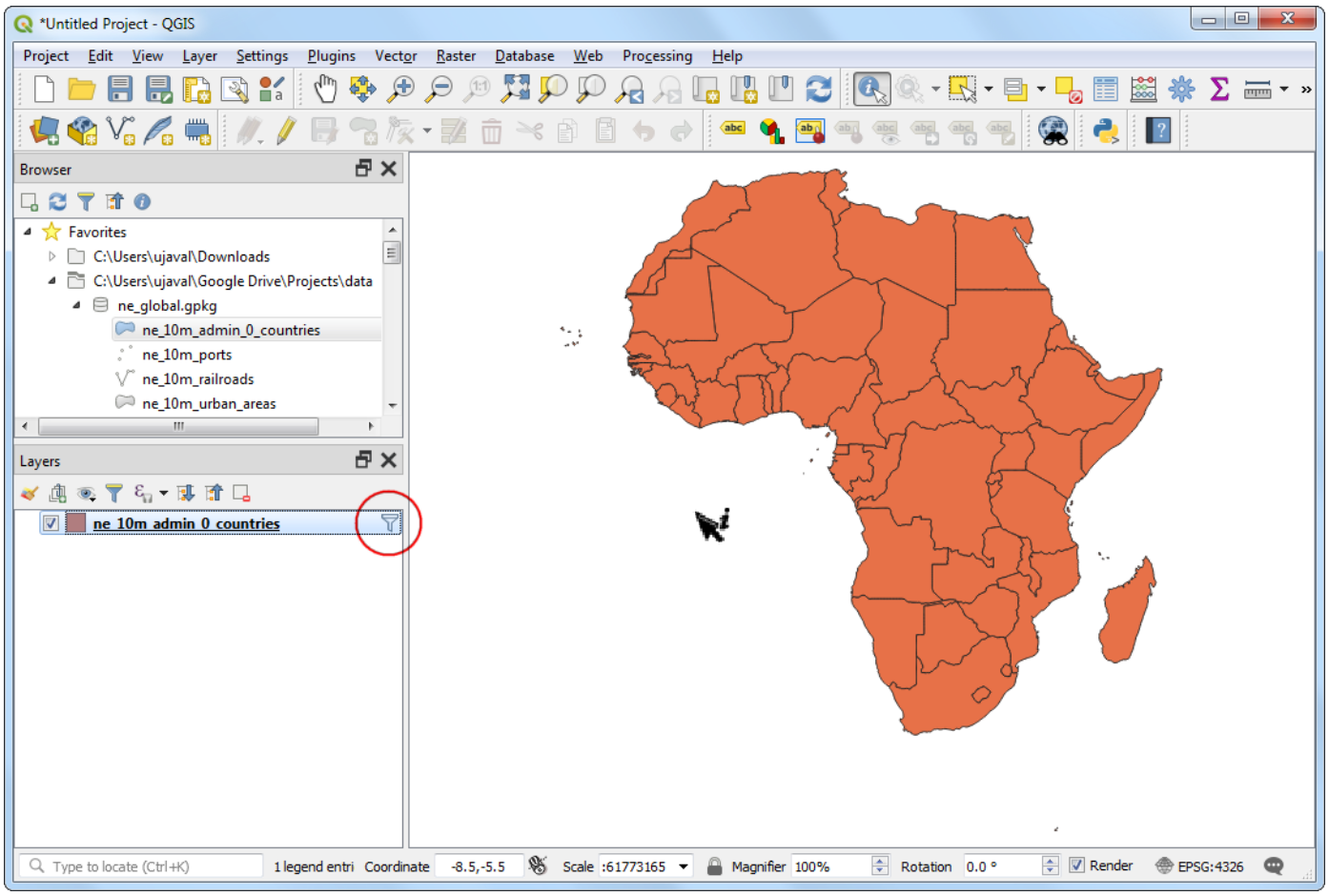


6. Now enter the expression in the Provider specific filter expression textbox. You can click the **CONTINENT** label, followed by = button and **Africa** label. Or you can type the following expression in the textbox. Click OK after entering the expression.

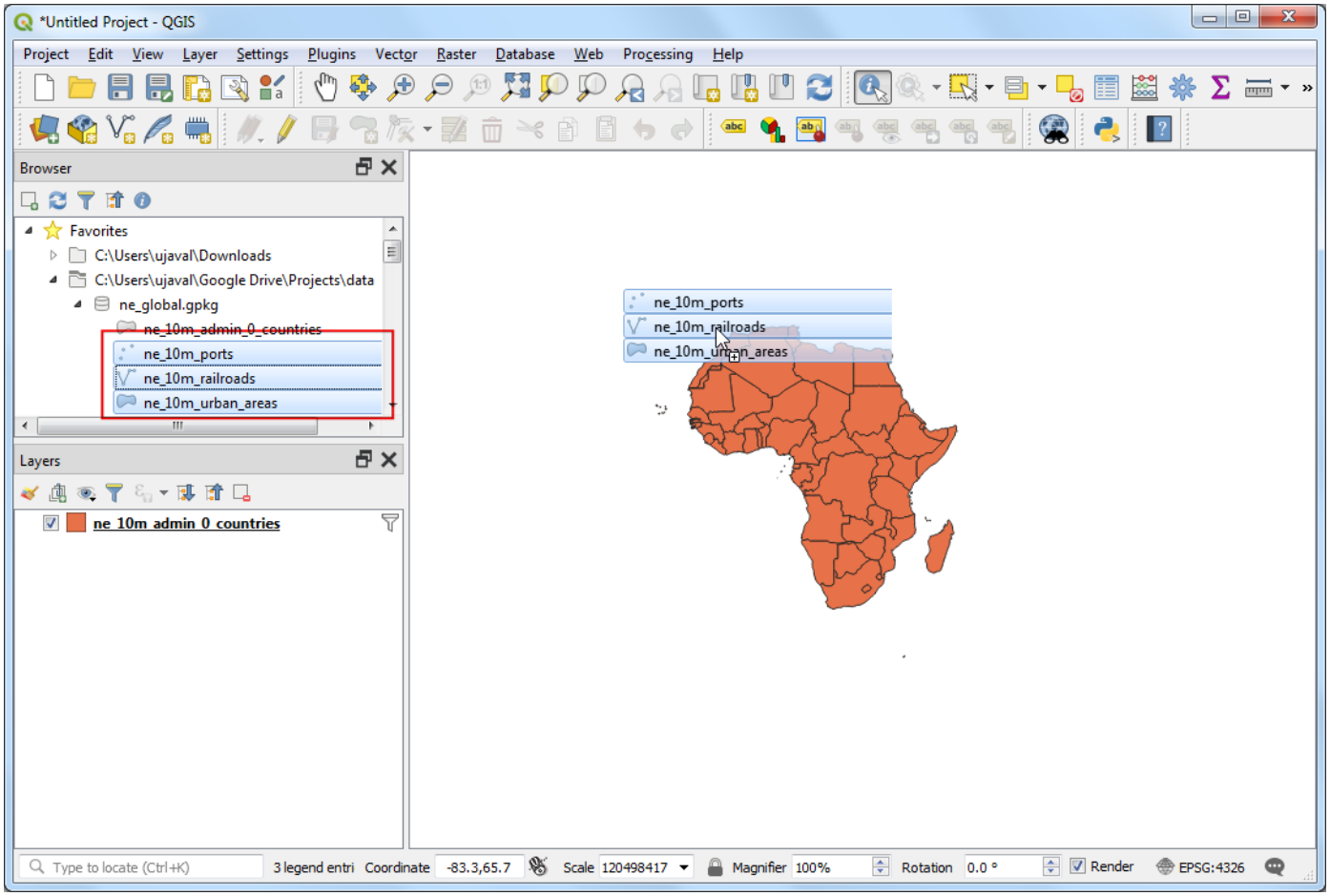
```
"CONTINENT" = 'Africa'
```



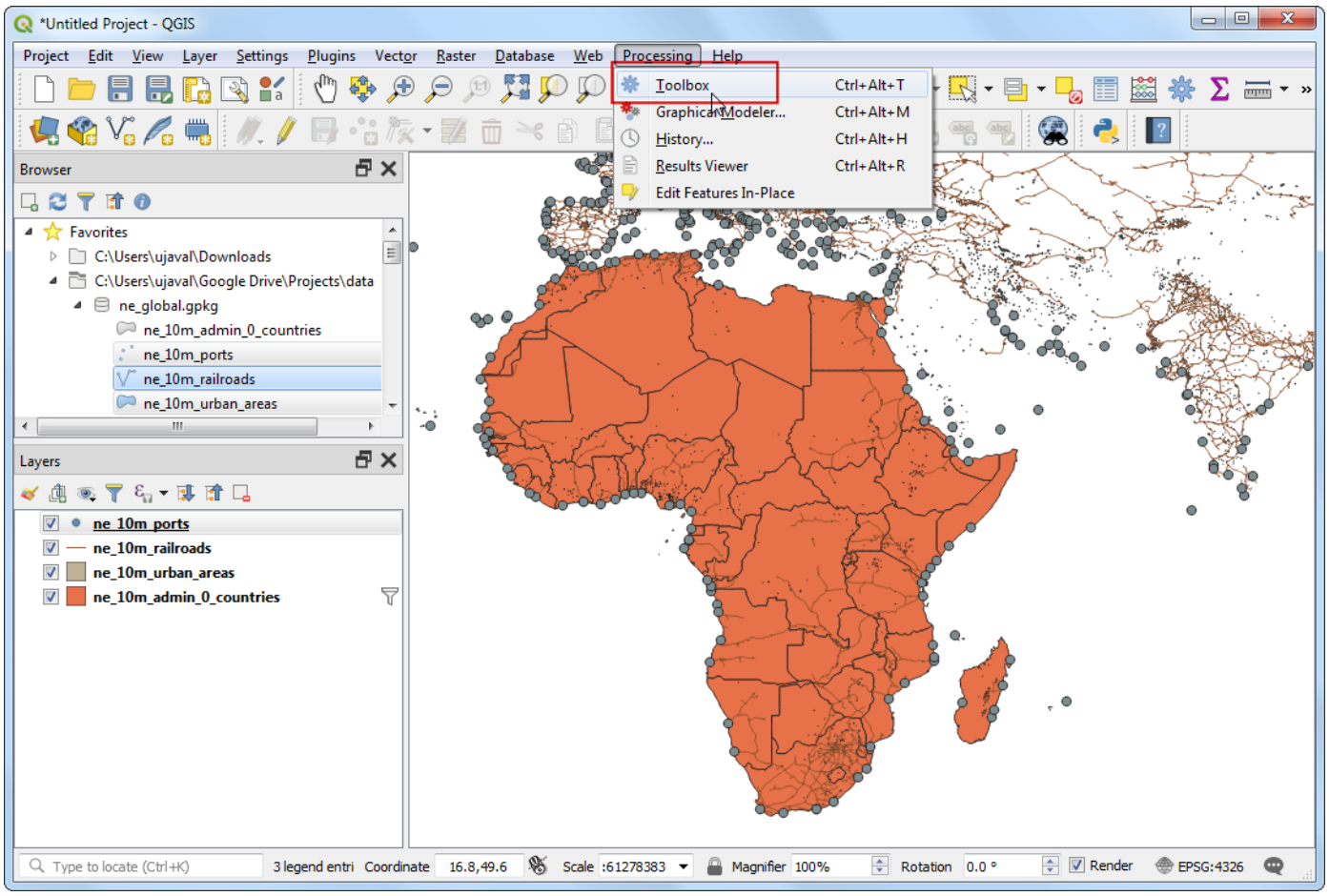
7. You will see that the map canvas now contains only the countries from Africa. Note the filter icon next to the layer name indicating that that layer a filter applied to it. If you wanted to see and use all the features from the layer, you can click the filter icon and clear the expression. For now, we will keep the filter so we can clip other layers to Africa.



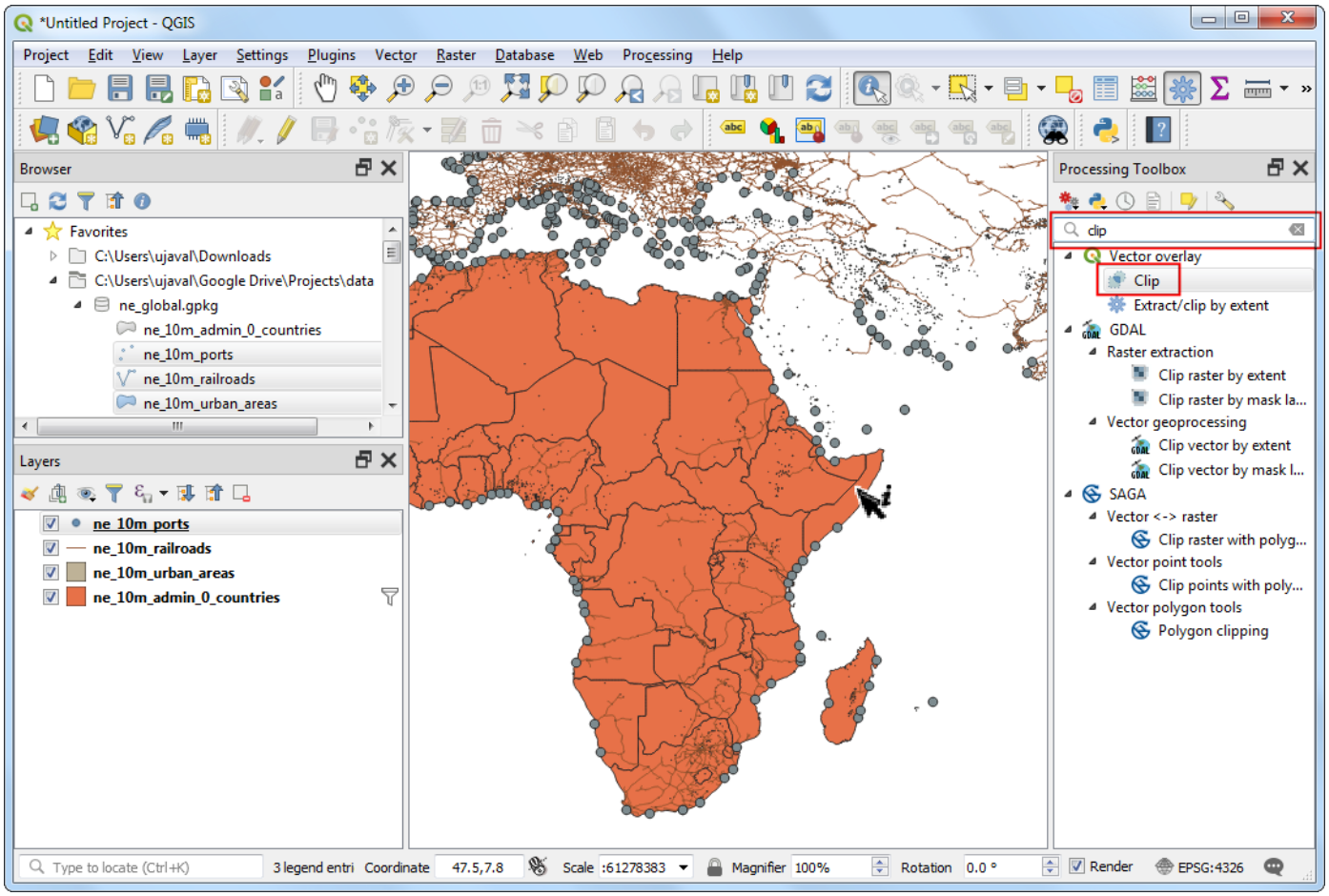
8. We are now ready to run the batch process to clip the layers. Locate the natural earth global layers `ne_10m_railroads` , `ne_10m_ports` and `ne_10m_urban_areas` in the QGIS Browser panel. Hold the `Ctrl` key and click each layer to select them. Once selected, drag them to the canvas.



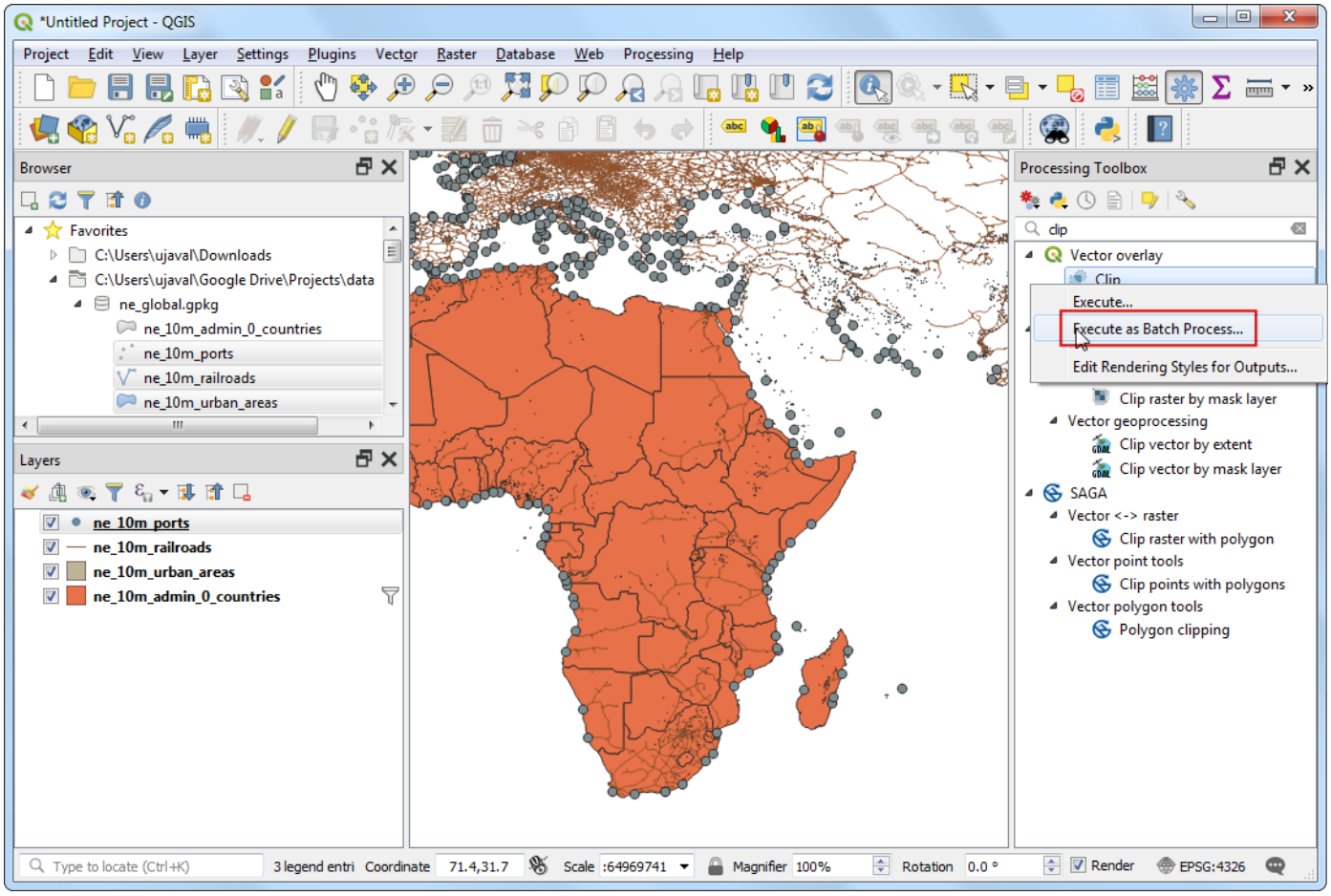
9. Once the layers are loaded, you will notice that they are global layers and have features spanning all the countries. Now, it's time to start our batch clip process to clip these layers to Africa. Open Processing ‣ Toolbox.



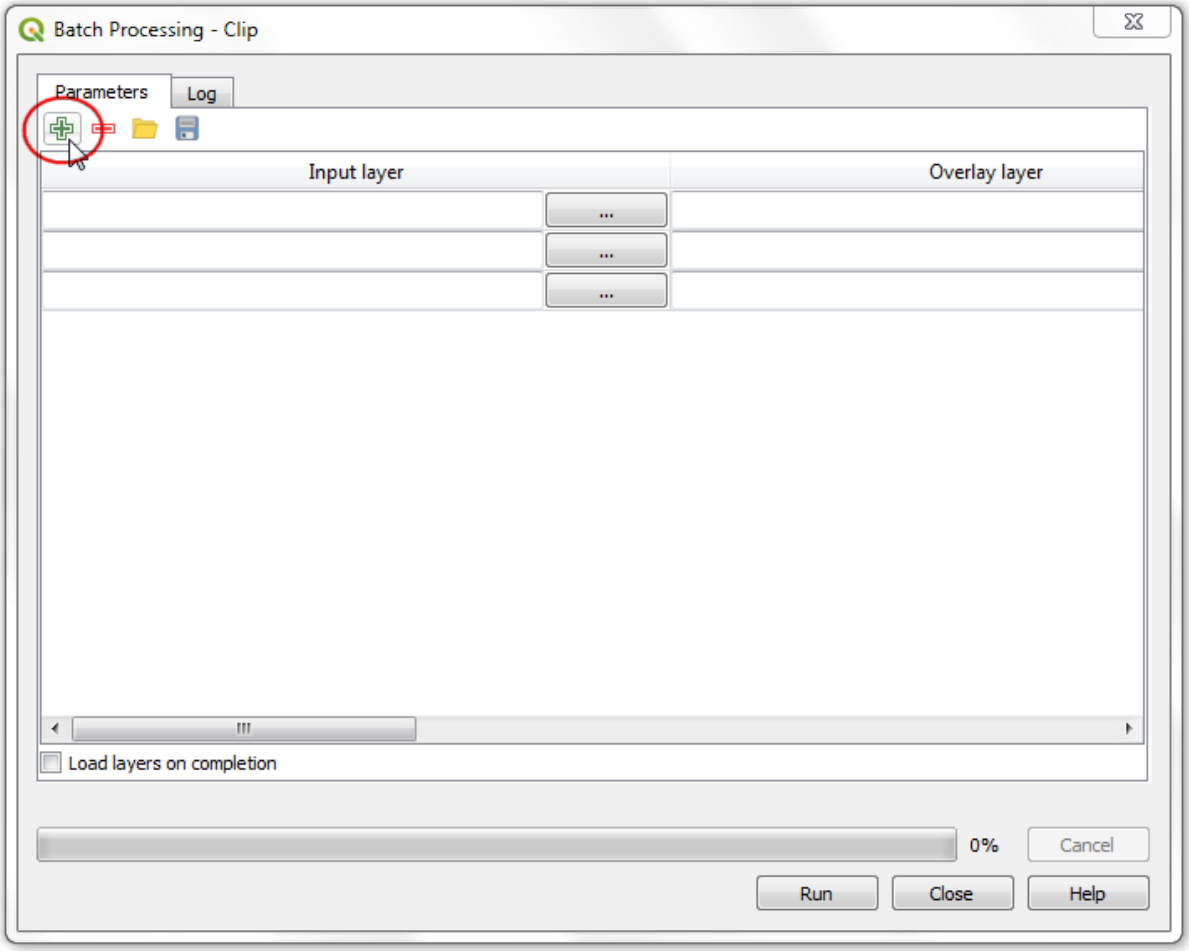
10. Browse all available algorithms and find the Clip tool from Vector overlay ‣ Clip. You may also use the Search box to easily find the algorithm as well.



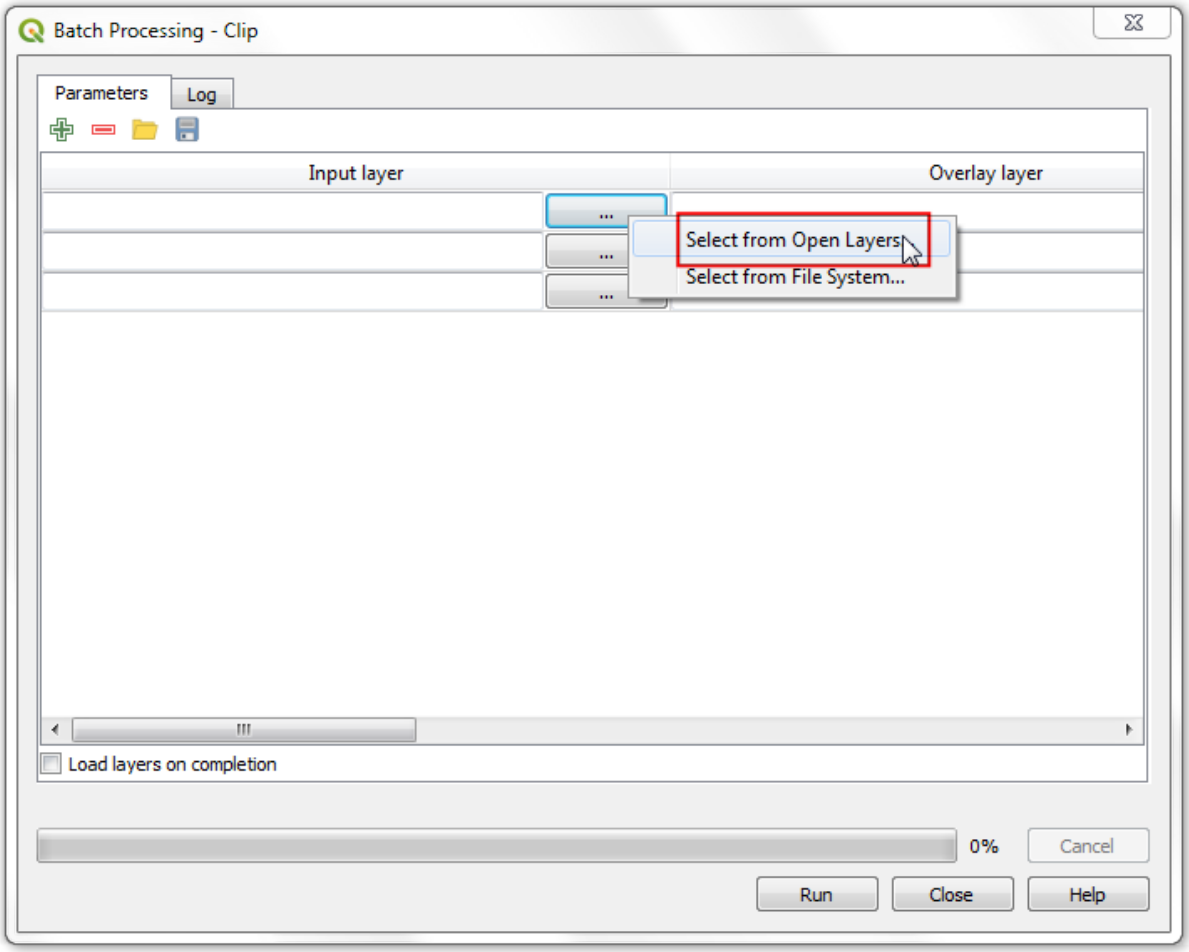
11. Right-click the Clip algorithm and select Execute as Batch Process.



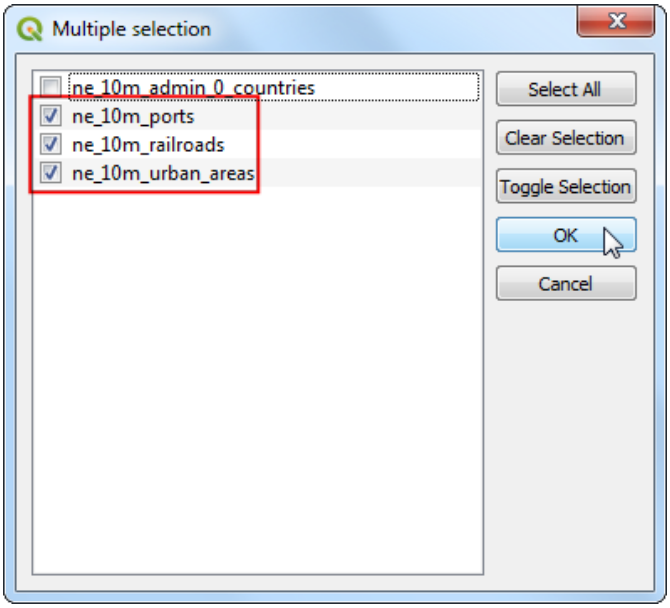
12. In the Batch Processing dialog, the first tab is Parameters where we define our inputs. Each row in the table represents 1 processing task. Click Add row button to add a new row. As our task involves 3 layers, add 3 rows.



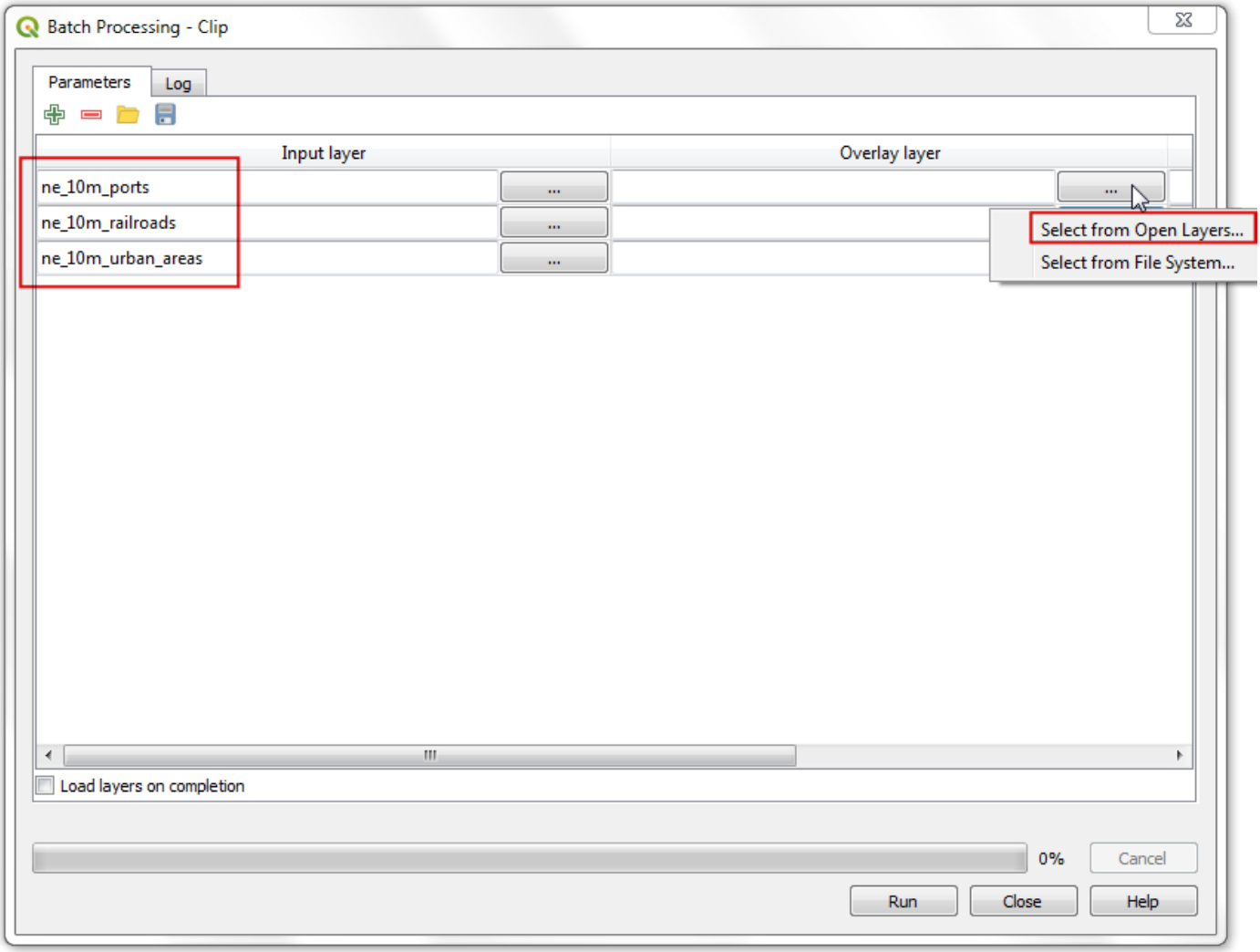
13. Click the ... next to the first row in the Input layers column. Select Select from Open Layers.



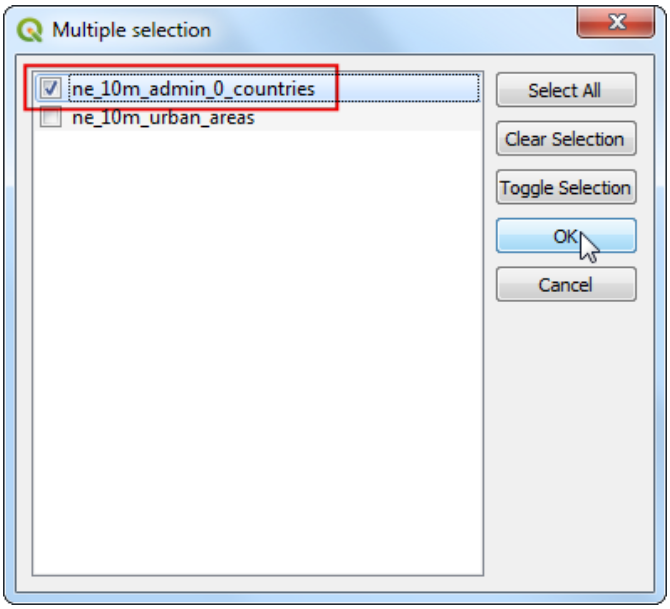
14. In the Multiple selection dialog, check the 3 layers that we want to clip and click OK.



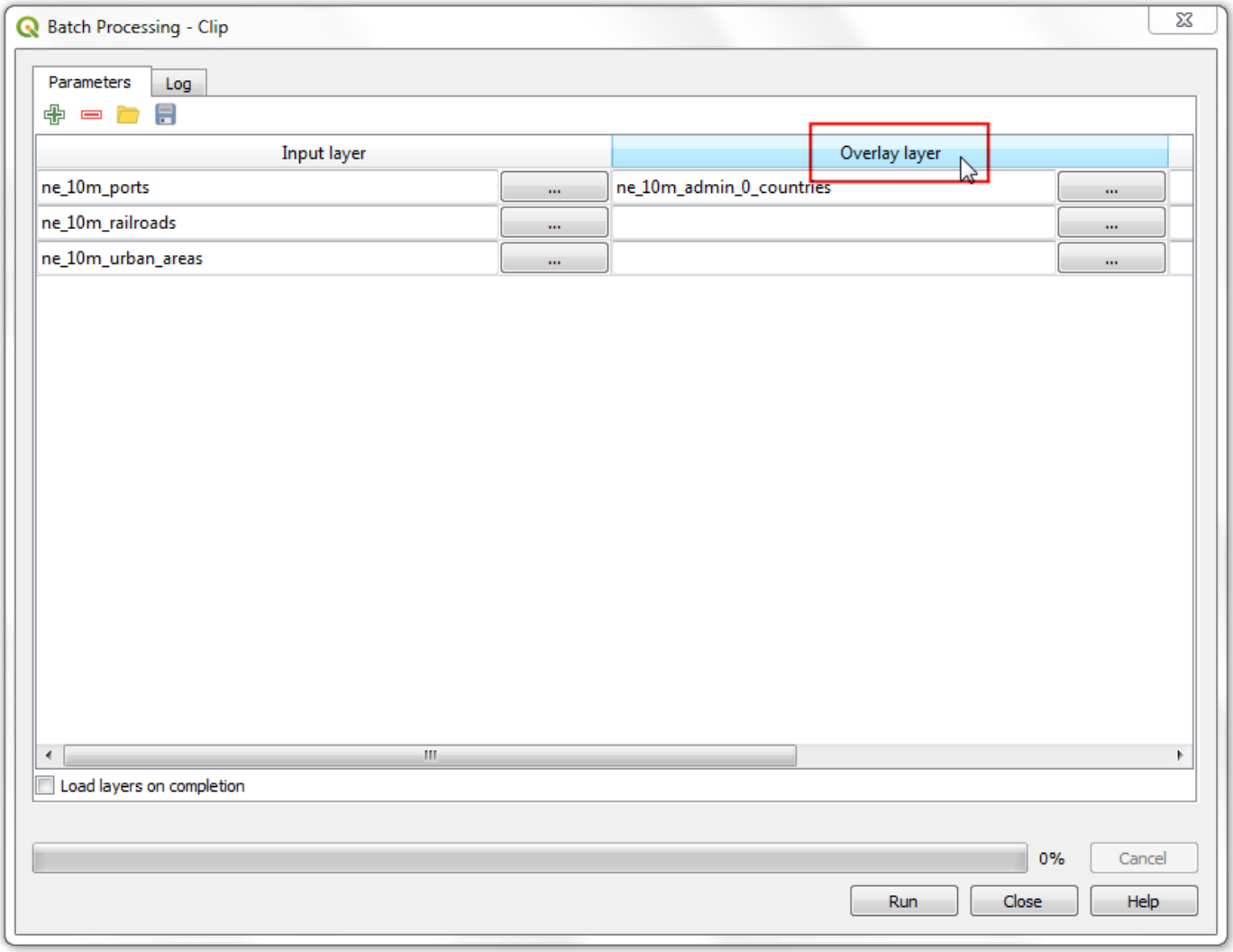
15. You will notice that the Input layer columns will be auto-populated with all layers you had selected. Next, we need to select the layer containing the boundary to clip our input layers. Click the ... button for the first row under Overlay layer column and select Select form Open Layers.



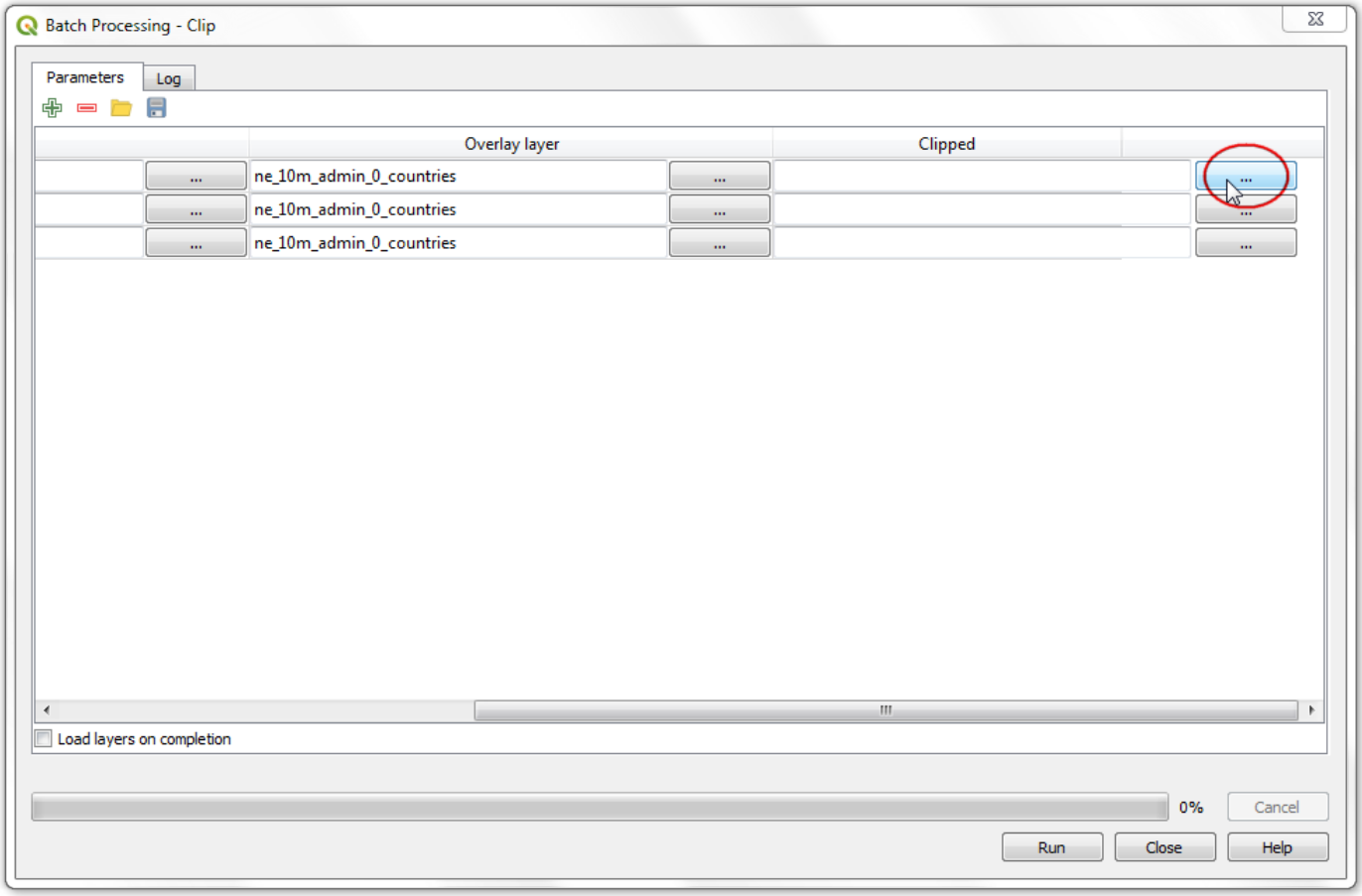
16. In the Multiple selection dialog, check `ne_10m_admin_0_countries` and click OK.



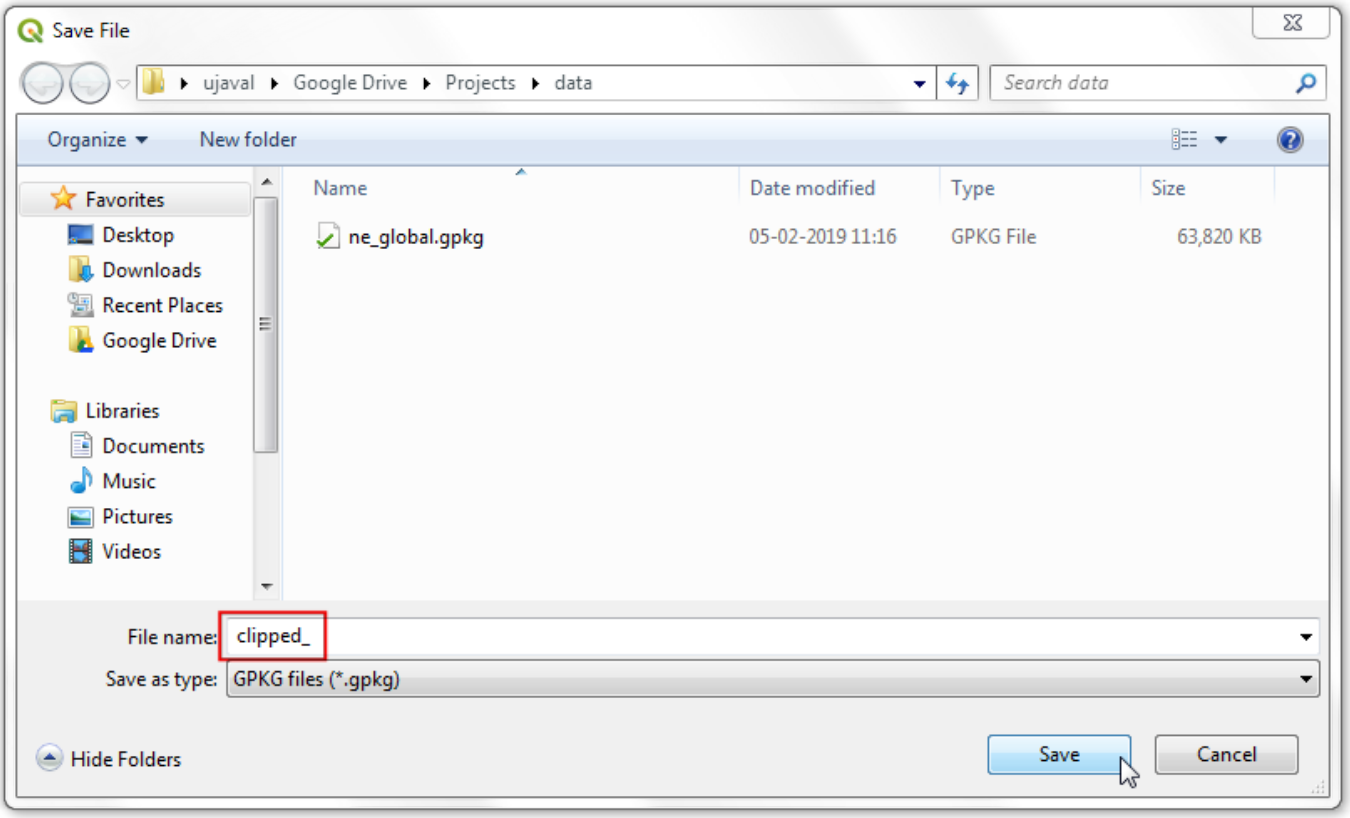
17. Since the clip layer is the same for all our inputs, a handy shortcut is to double-click the column header Overlay layer and the same layer will be auto-filled for all the rows.



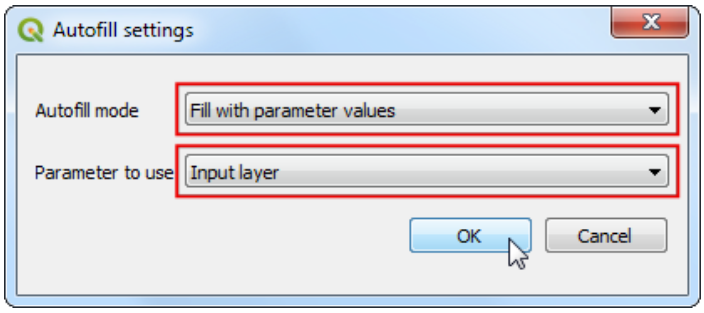
18. Next, we need to define our outputs. Click the ... buton next to the first row in the Clipped column.



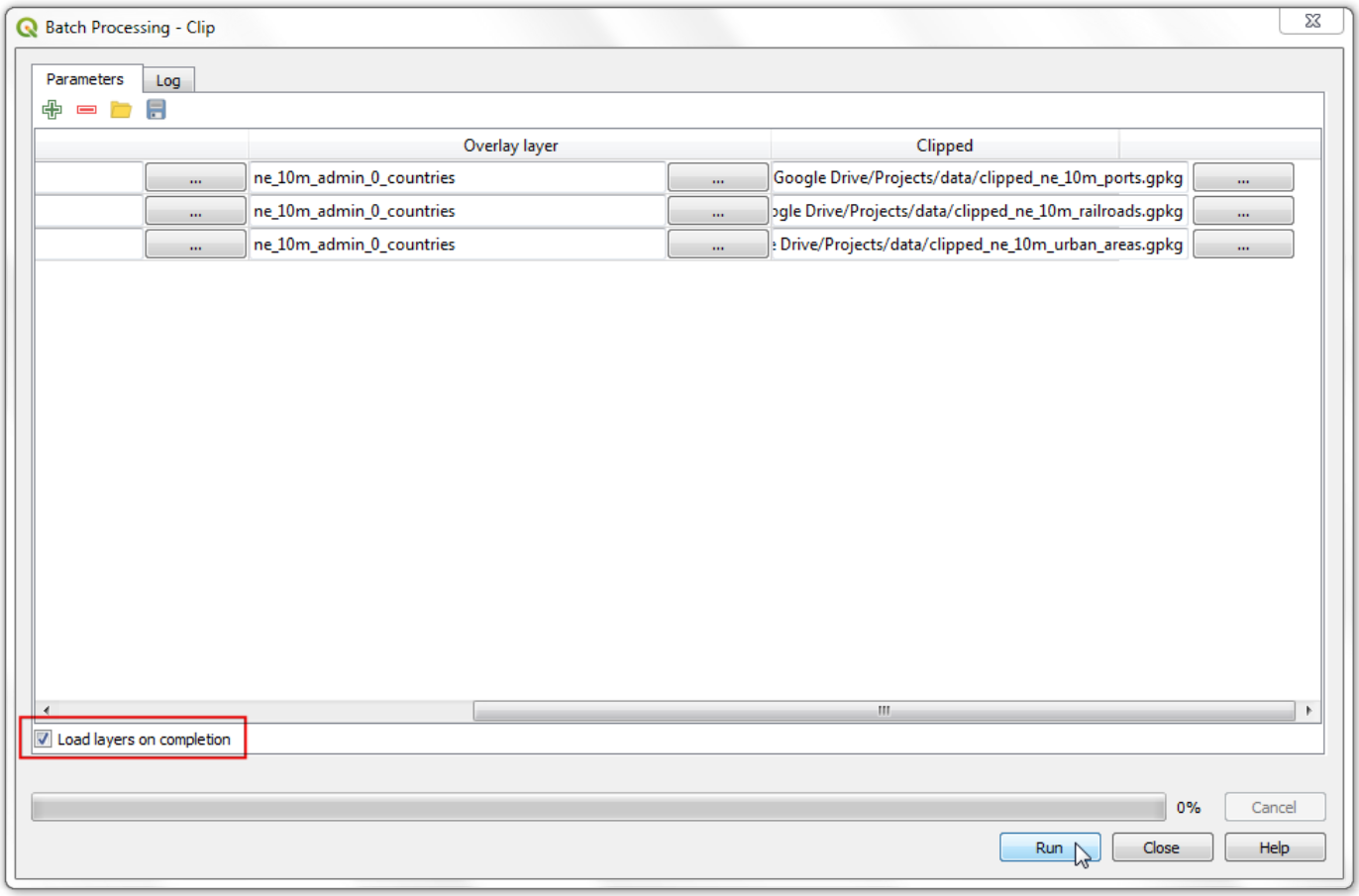
19. Browse the the directory where you want your output layers. Type the filename as `clipped_` and click Save.



20. You will see a new Autofill settings dialog pop up. Select `Fill with parameter values` as the Autofill mode. Select `Input layer` as the Parameter to use. This setting will add the input file name to the output along with the specified `clipped_` filename. This is important to ensure all the output files have unique names and they do not overwrite each other.



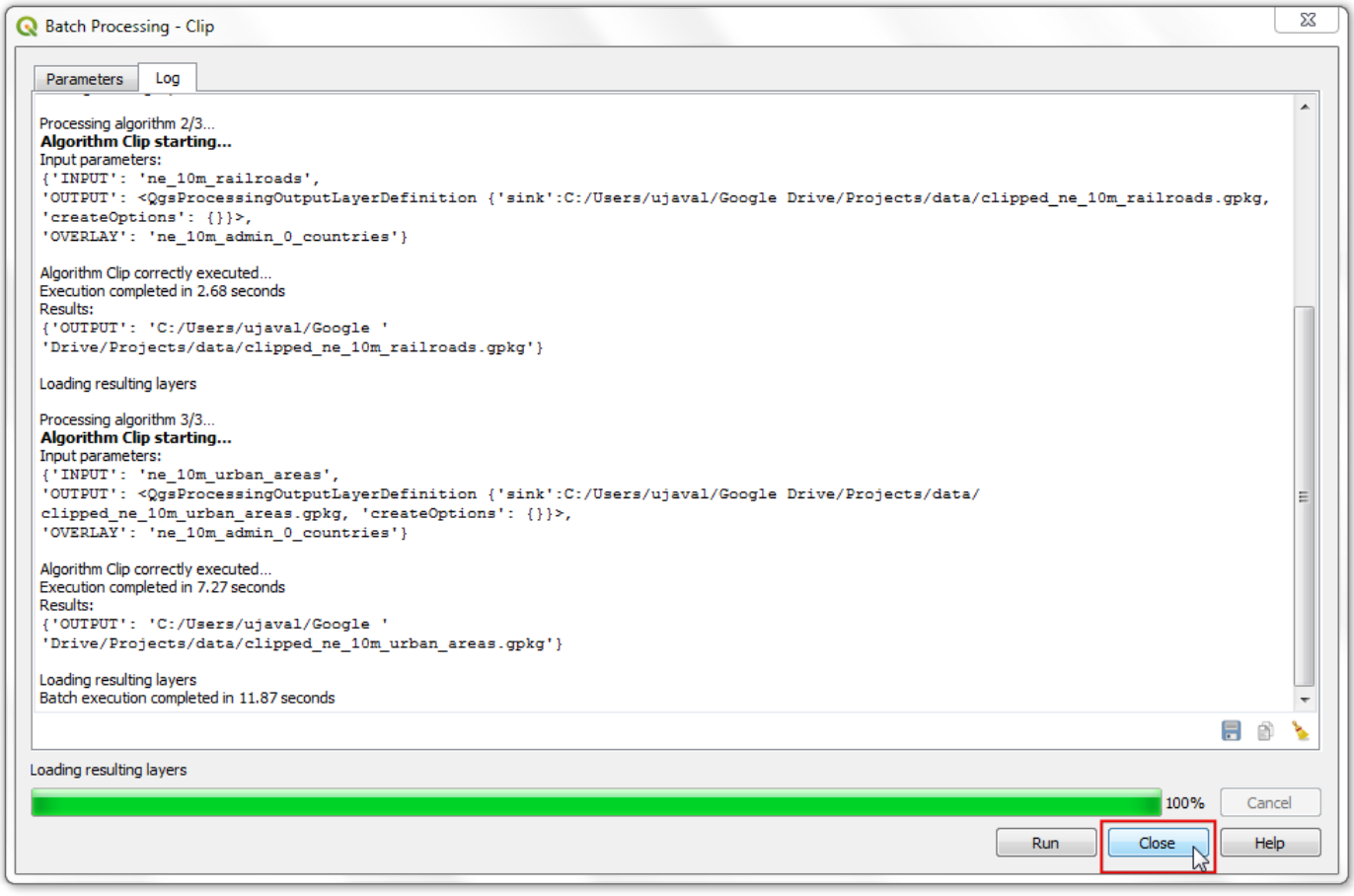
21. Now we are ready to start the batch procesing. Make sure to check Load layers on completion and click Run.



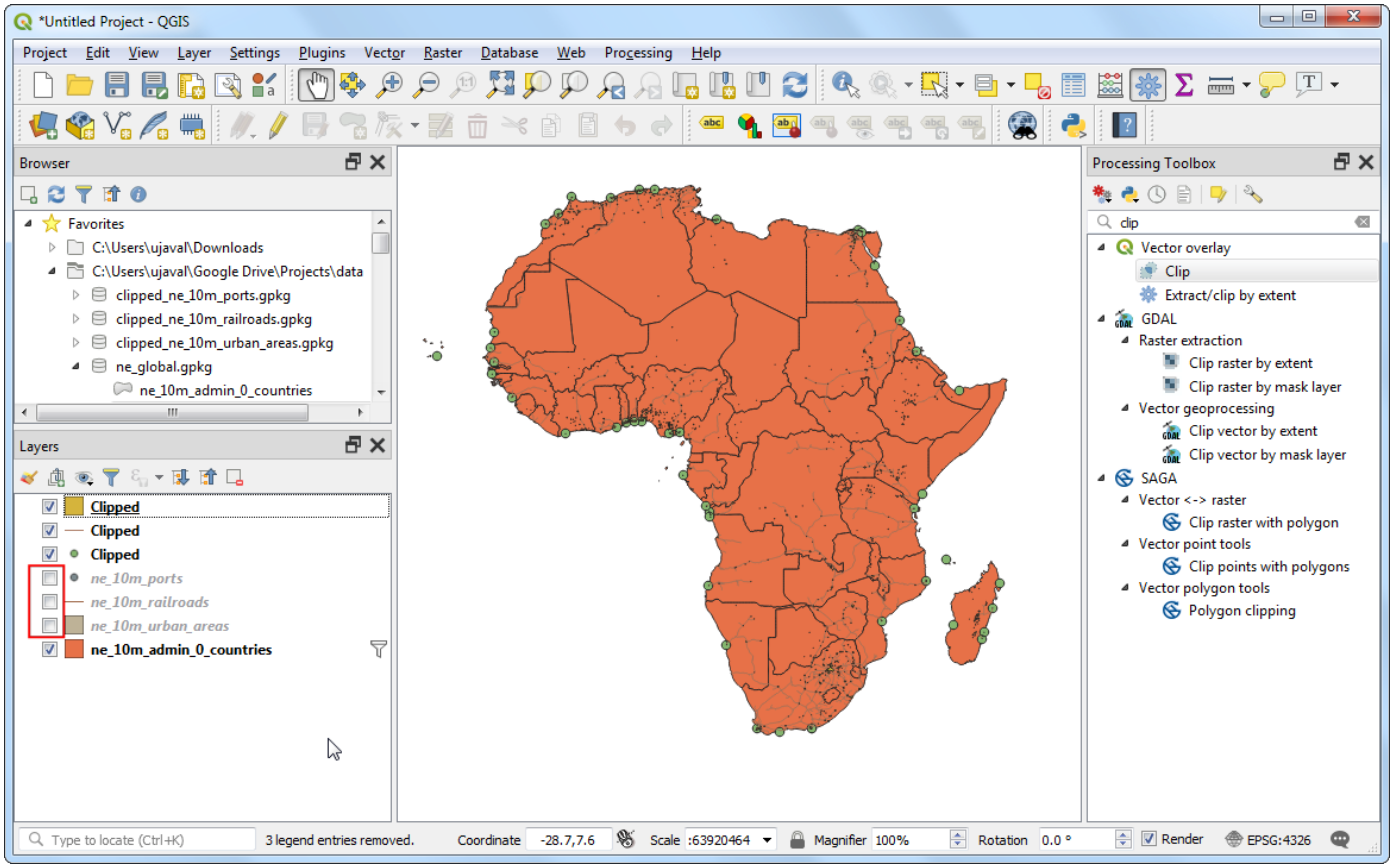
22. The clip algorithm will run for each of the inputs and create output files are we have specified. Once the batch process finishes, click Close to return to QGIS.

Note

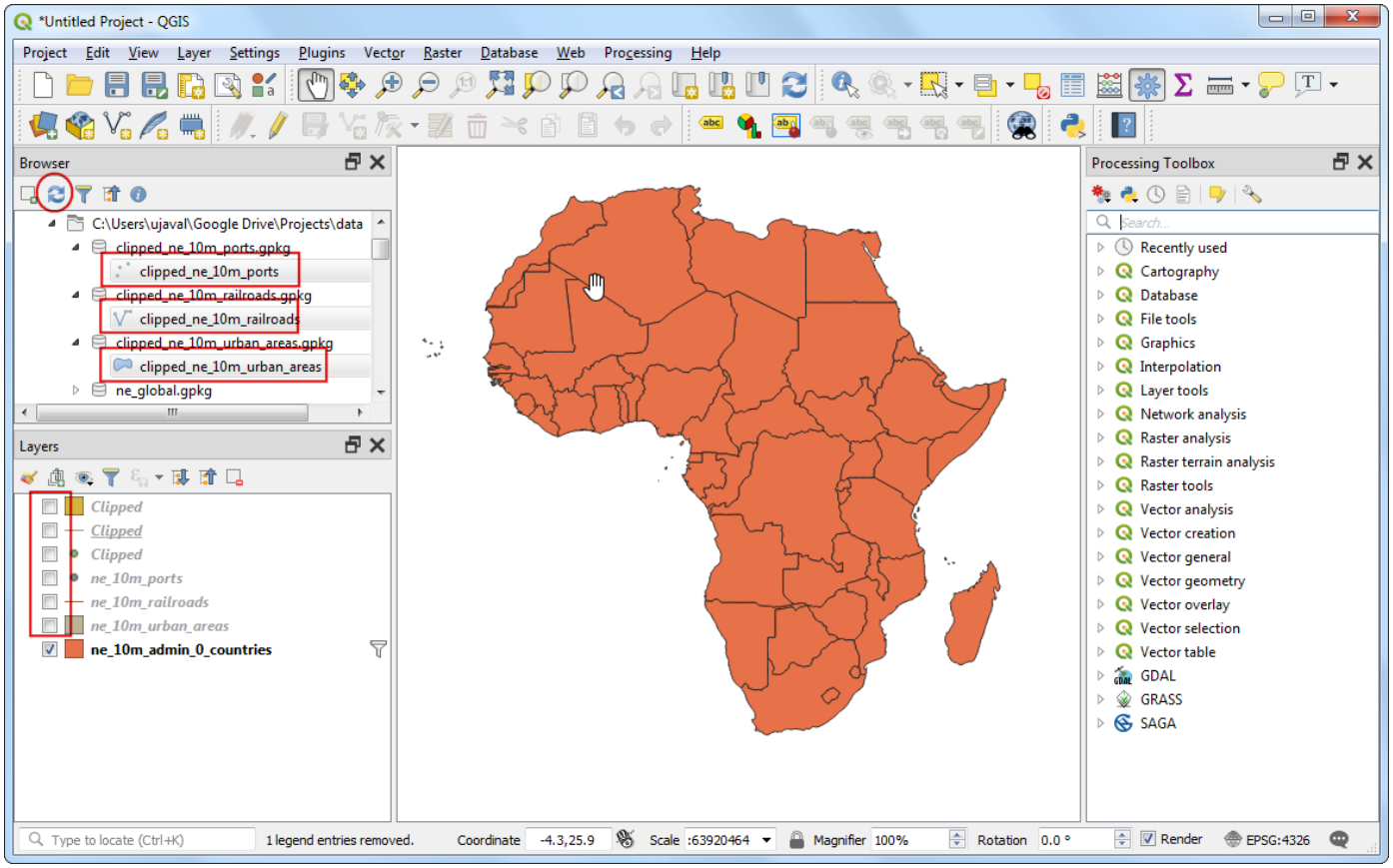
Tip: QGIS can now run Processing tasks in the background without blocking the user interface. So if your batch process is taking long, you can close the dialog and continue to work on other tasks in QGIS while the process keeps running in the background.



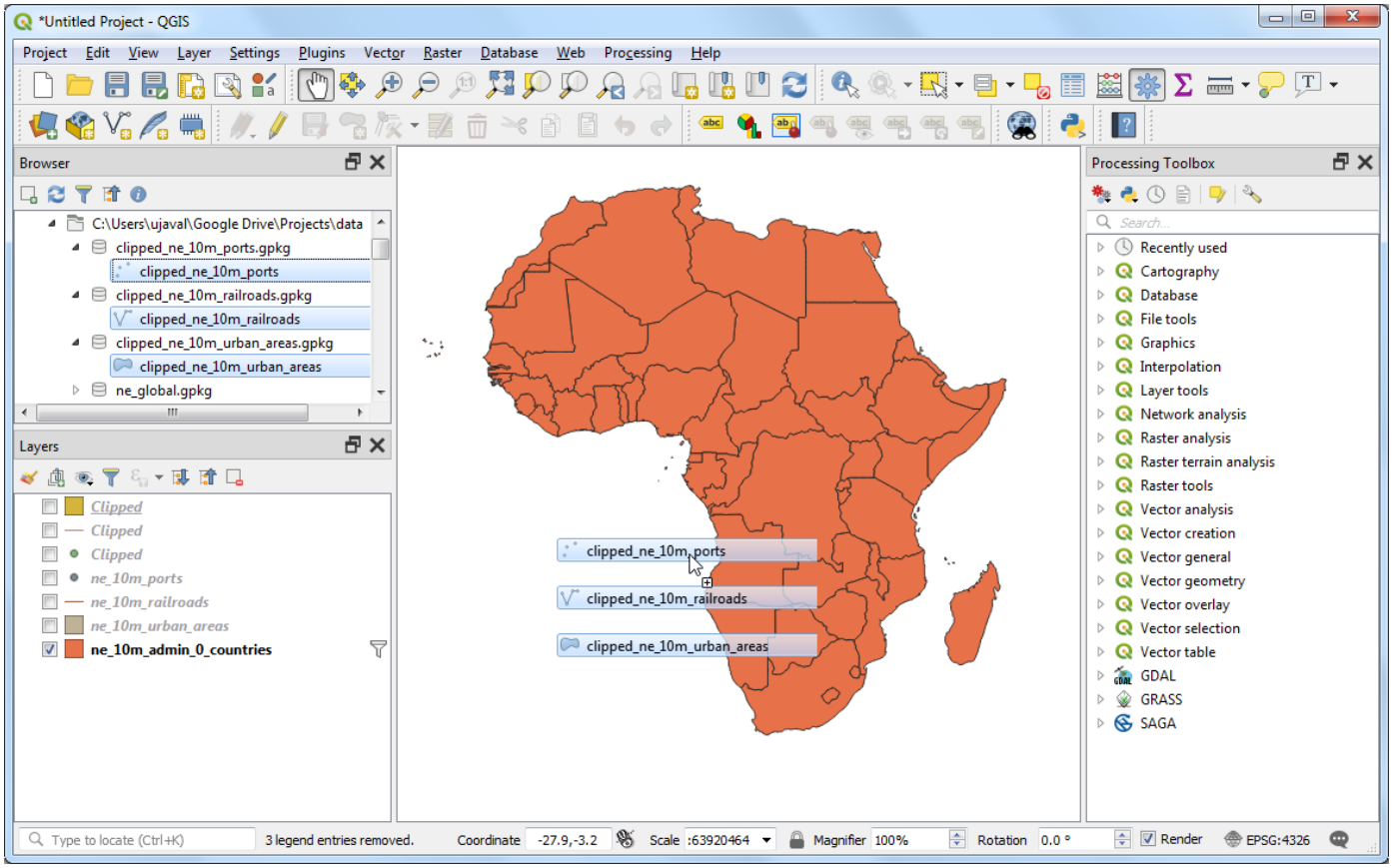
23. Back in the main QGIS window, you will see the layers added to QGIS canvas. As you will notice, all the global layers are properly clipped to the continent boundary that we had specified.



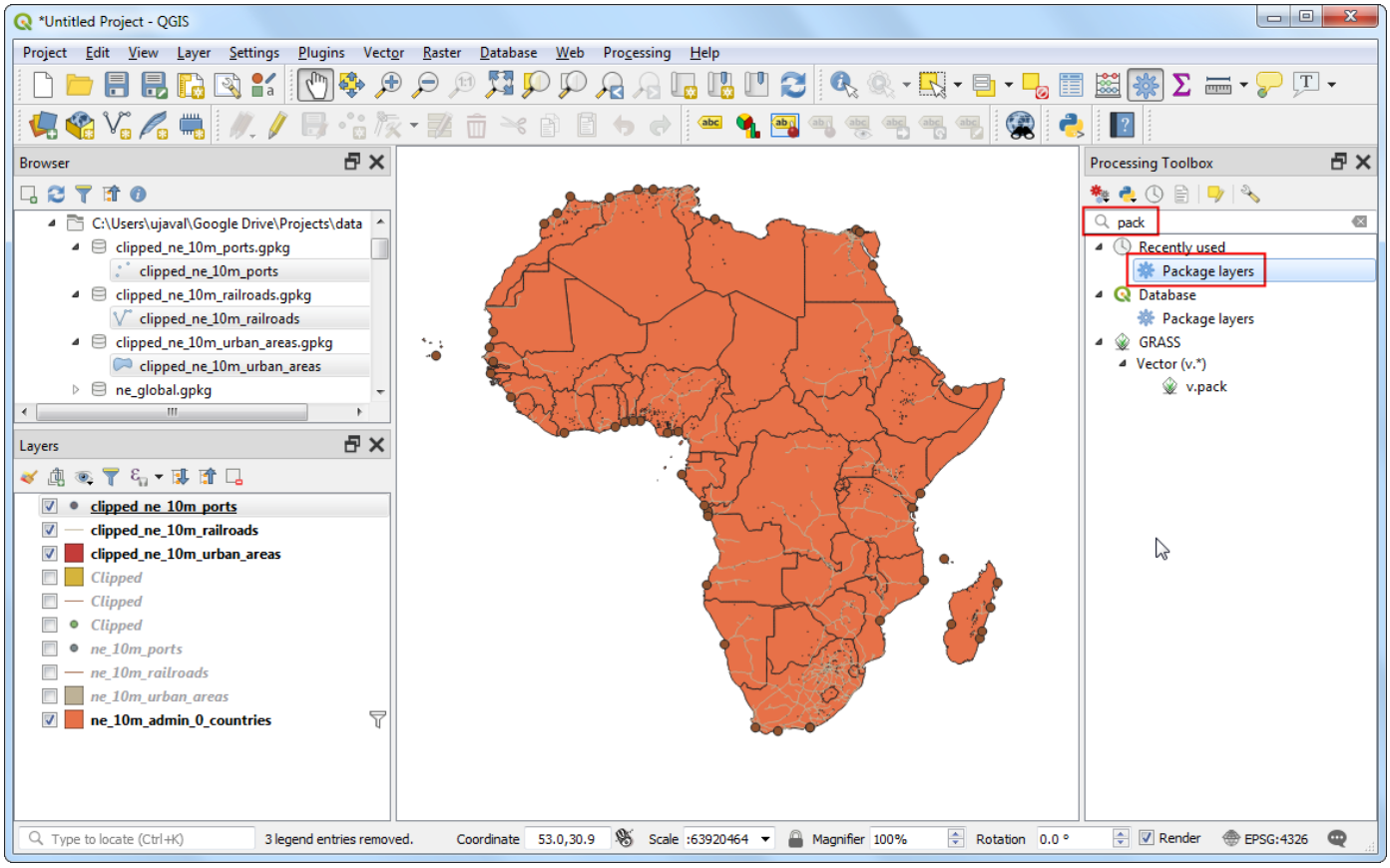
24. We have accomplished the task of clipping multiple layers in a batch. But QGIS3 has another handy feature that will help you save and deliver the result in a more efficient way. If you wanted to deliver the clipped layers to someone, you would zip the individual files outside of QGIS. A better option to to package the output layers in a single Geopackage. In the QGIS Browser, locate the clipped output layers. You may have to click the Refresh button to see the newly added files.



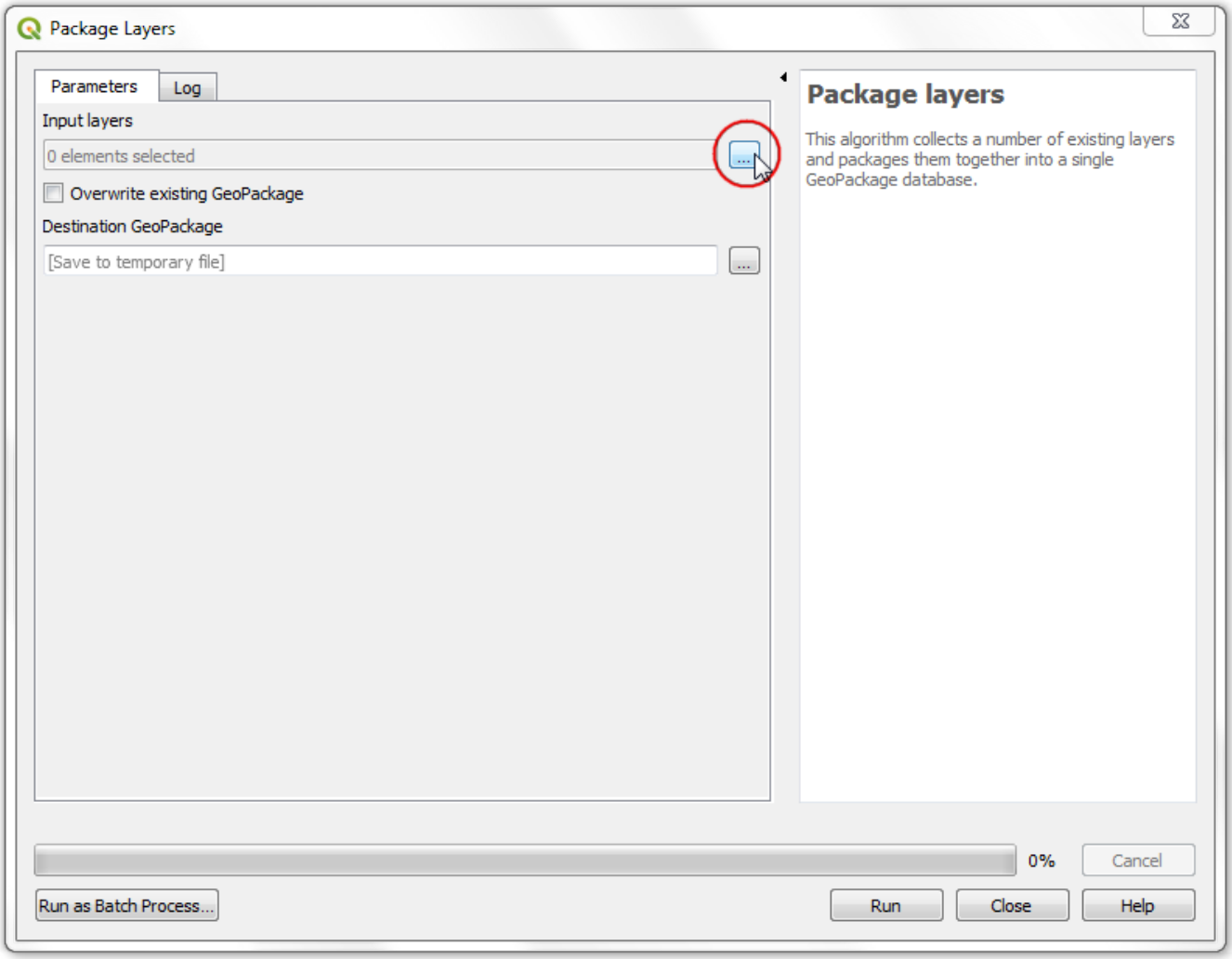
25. Hold `Ctrl` key and select the layers. Drag them to the canvas to load them in QGIS.



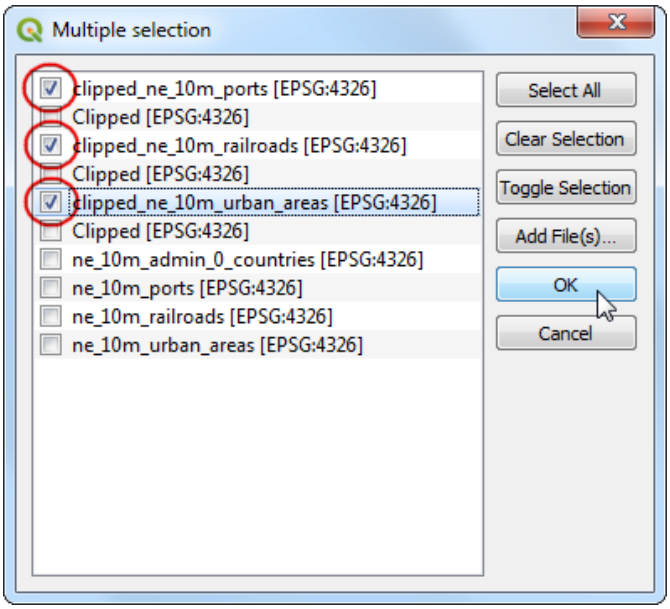
25. In Processing Toolbox, locate the Database ‣ Package layers tool.



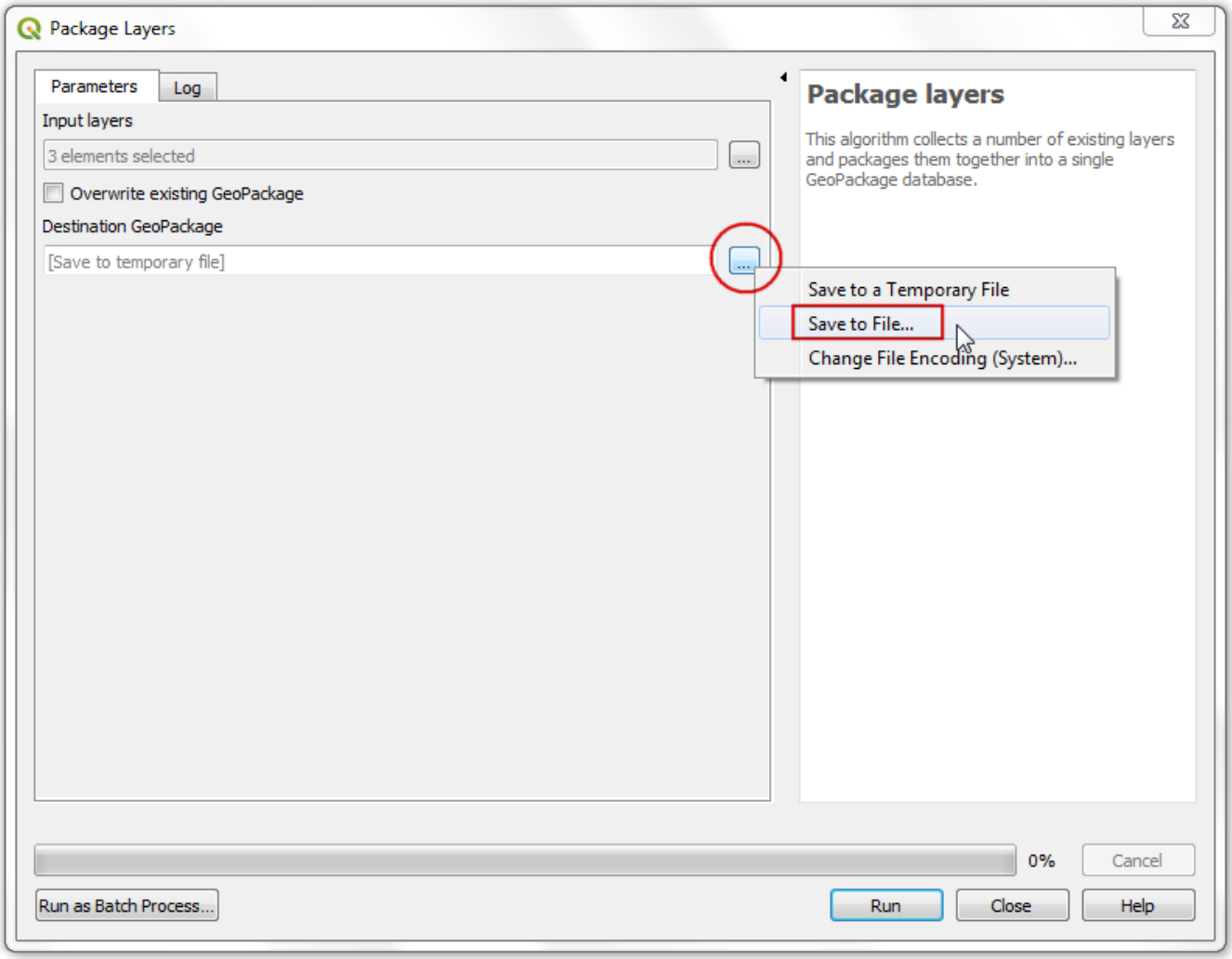
27. In the Package layers dialog, click the ... button next to Input layers.



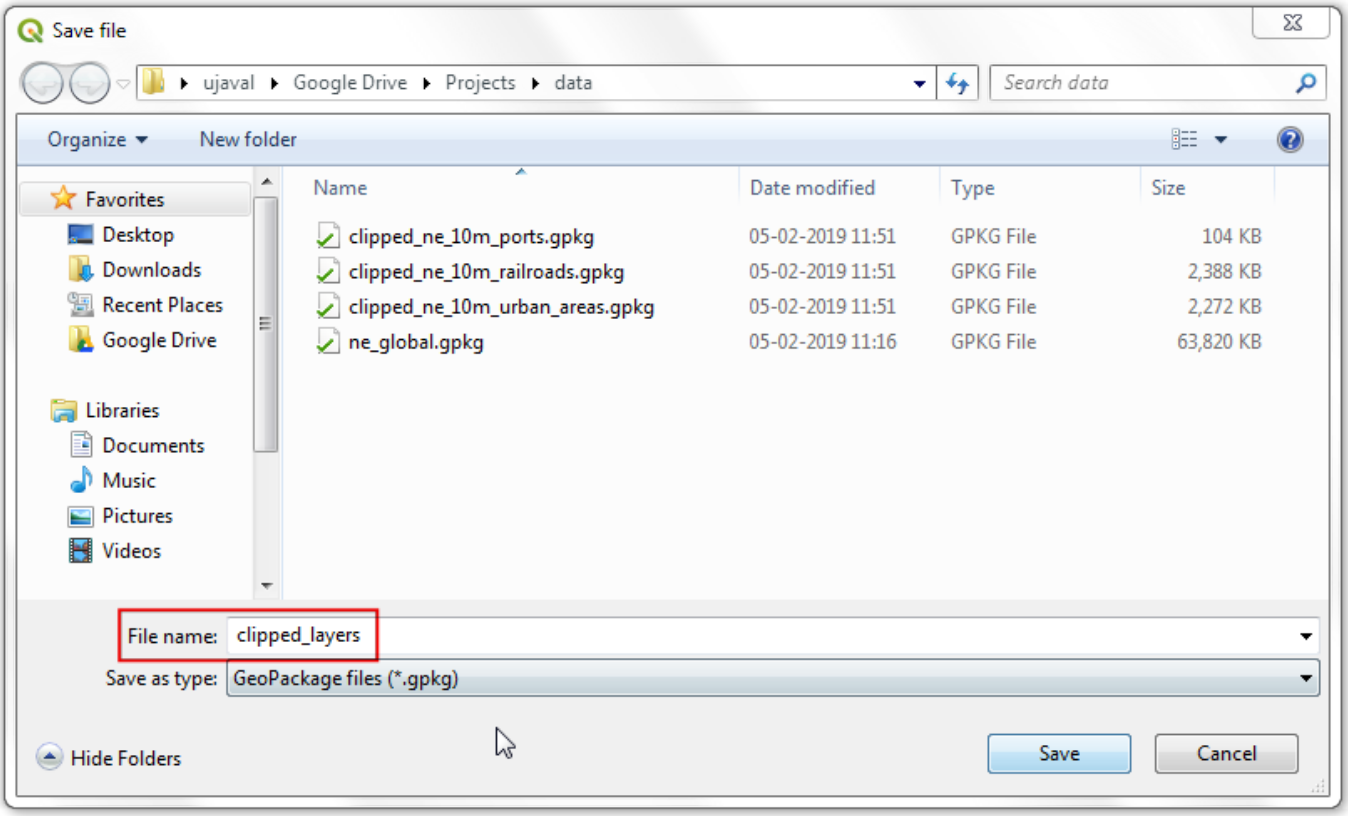
28. In the Multiple selection dialog, check the `clipped_ne_10m_ports`, `clipped_ne_10m_railroads` and `clipped_ne_10m_urban_areas` layers. Click OK.



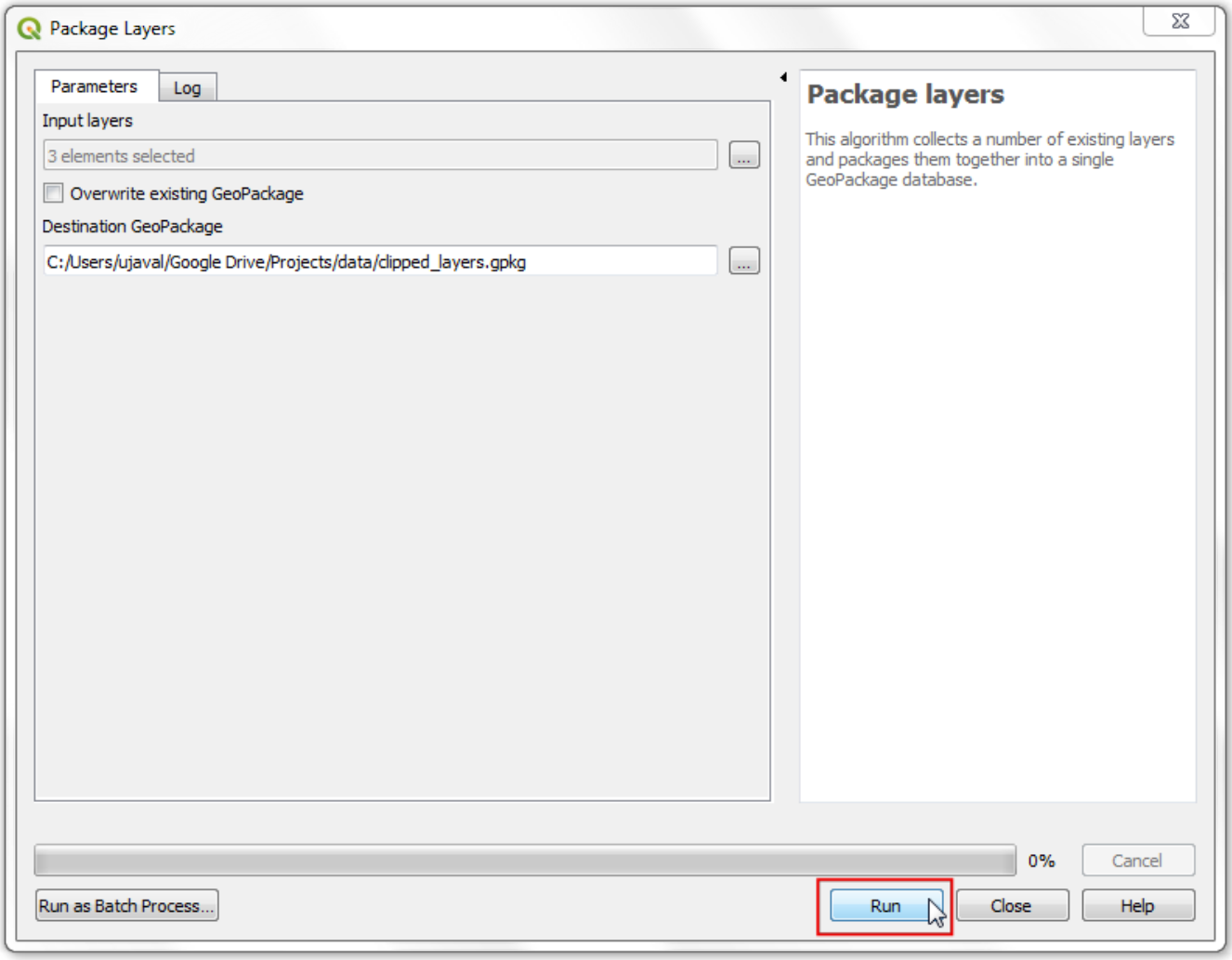
29. Once the input layers are selected, click the ... next to Destination Geopackage and choose Save To File.



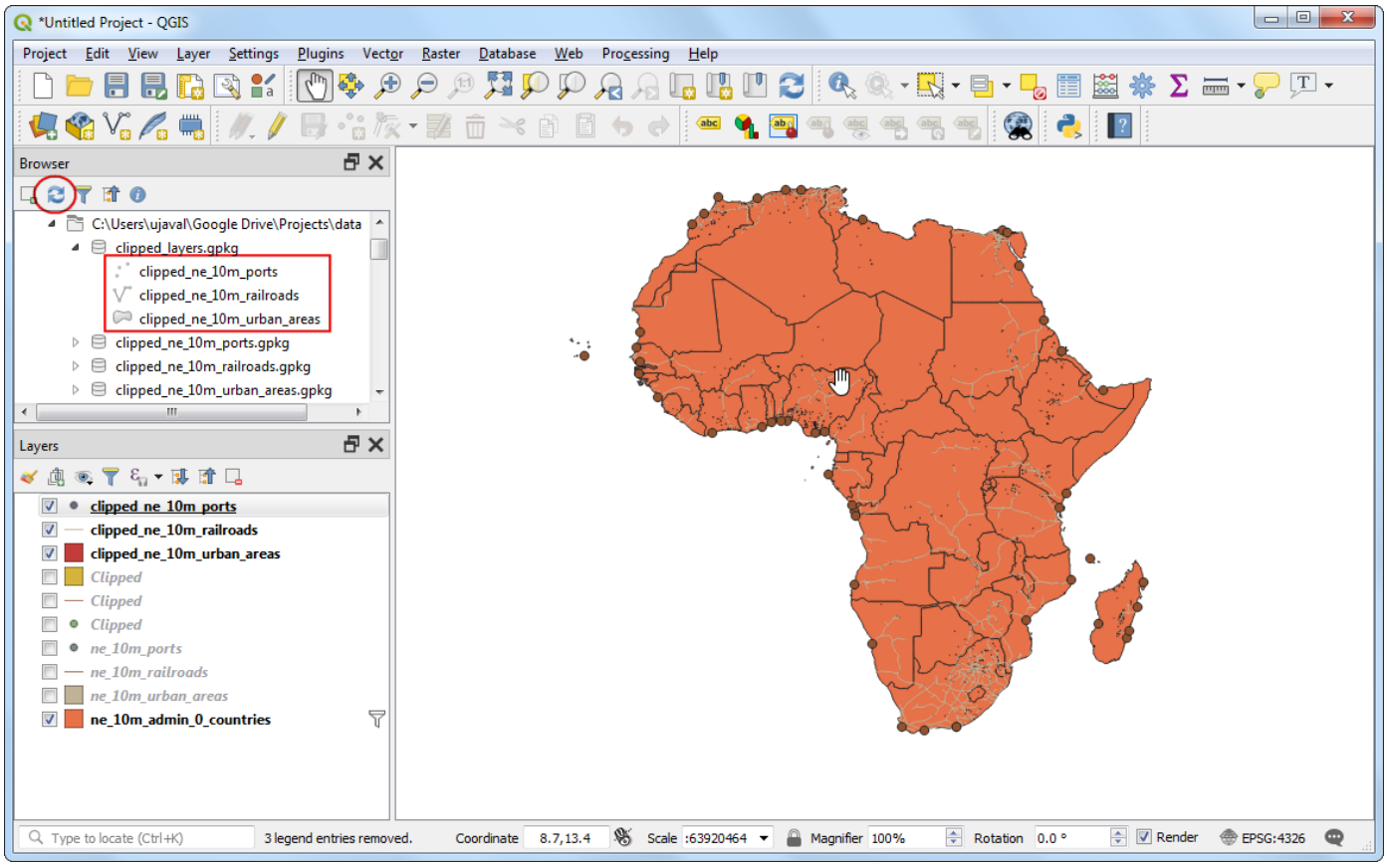
30. Enter the output file name as `clipped_layers` .



31. Click Run to start the packaging process.



32. Once the process finishes, you will see a new geopackage file in your QGIS Browser containing all the clipped output layers. This is a single file on your computer that contains all the output layers.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Automating Complex Workflows using Processing Modeler (QGIS3)

GIS Workflows typically involve many steps - with each step generating intermediate output that is used by the next step. If you change the input data or want to tweak a parameter, you will need to run through the entire process again manually. Fortunately, QGIS has a graphical modeler built-in that can help you define your workflow and run it with a single invocation. You can also run these workflows as a batch over a large number of inputs.

Overview of the task

We will take a point layer of maritime piracy incidents and create a processing model to produce a density map by aggregating them over a global hexagonal grid.

Other skills you will learn

- Using a global equal area projection and setting the Project CRS.
- Applying a Graduated symbology to a polygon layer.

Get the data

National Geospatial-Intelligence Agency's Maritime Safety Information portal (https://msi.nga.mil/NGAPortal/MSI.portal) provides a shapefile of all incidencts of maritine piracy in the form on Anti-shipping Activity Messages (https://msi.nga.mil/Piracy). Download the Arc Shape file (https://msi.nga.mil/api/publications/download?key=16920958/SFH00000/ASAM_shp.zip&type=download) version of the database.

Natural Earth (http://naturalearthdata.com) has several global vector layers. Download the 10m Physical Vectors - Land (https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/physical/ne_10m_land.zip) containing Land polygons.

For convenience, you may directly download a copy of the above layers from below:

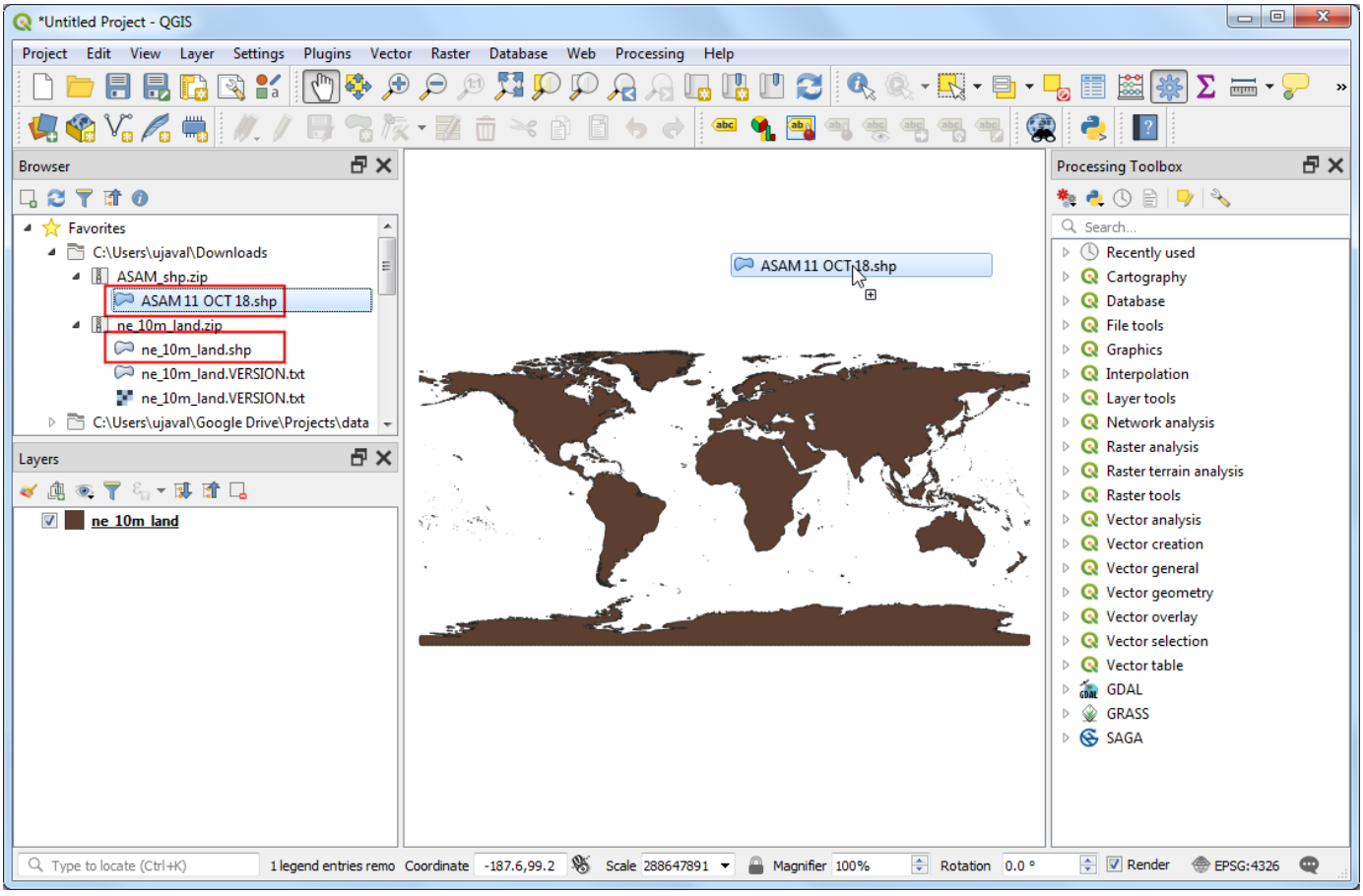
ASAM_shp.zip (http://www.qgistutorials.com/downloads/ASAM_shp.zip)

ne_10m_land.zip (http://www.qgistutorials.com/downloads/ne_10m_land.zip)

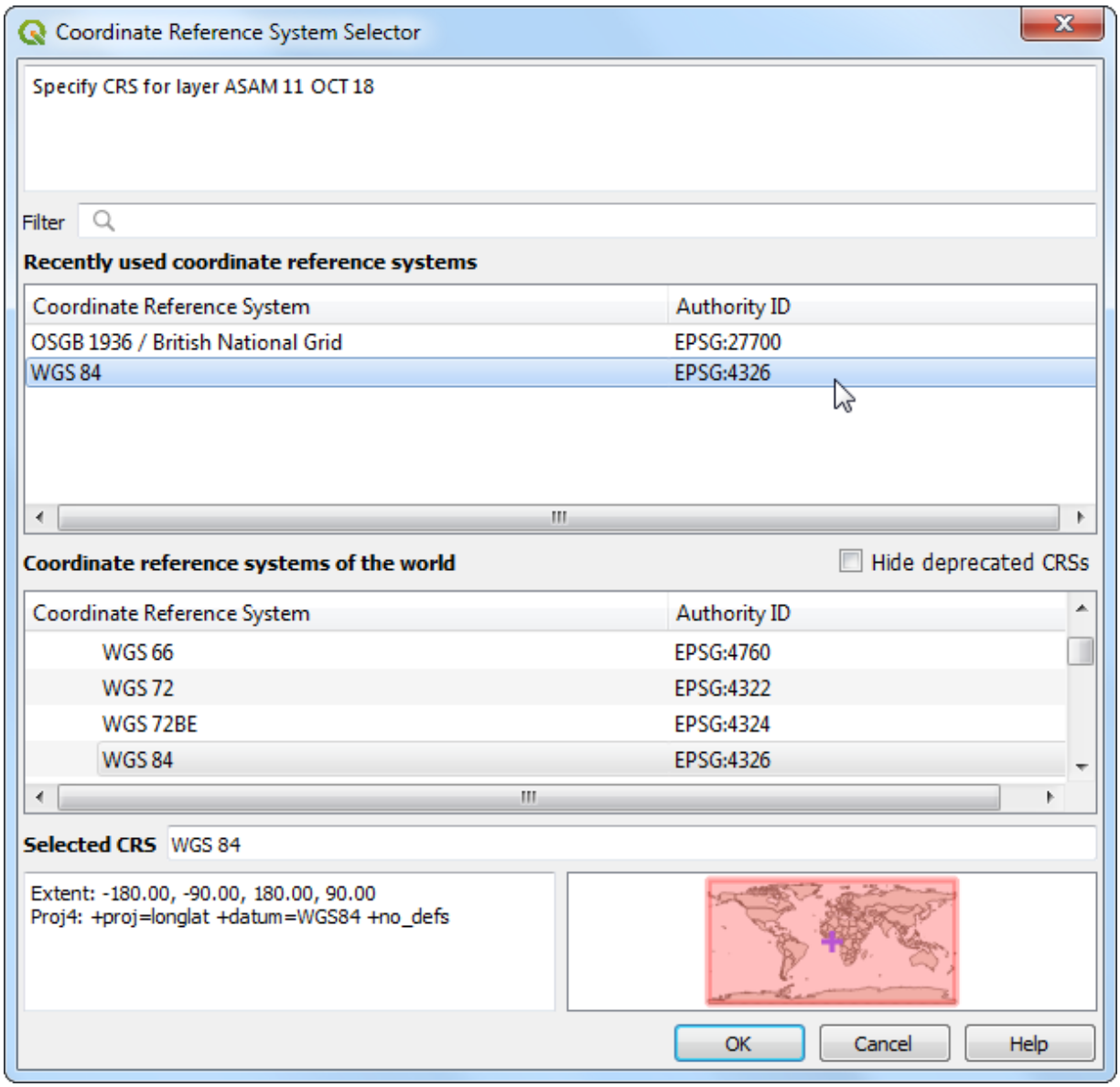
Data Source: [NGA_MSI] (../credits.html#nga-msi) [NATURALEARTH] (../credits.html#naturalearth)

Procedure

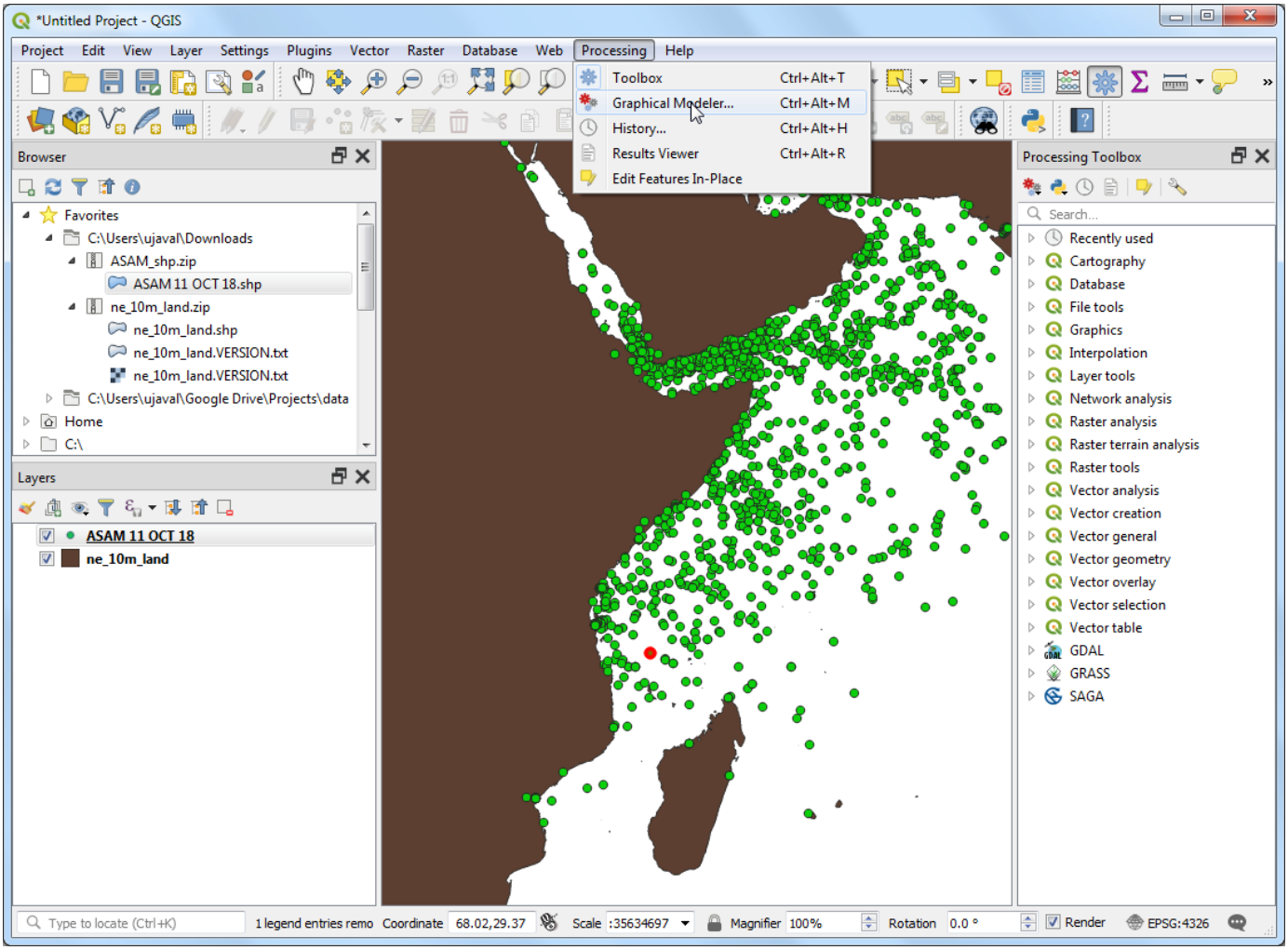
1. In the QGIS Browser Panel, locate the directory where you saved your downloaded data. Expand the `ne_10m_land.zip` and select the `ne_10m_land.shp` layer. Drag the layer to the canvas. Next, locate the `ASAM_shp.zip` file. Expand it and select the `asam_data_download/ASAM_events.shp` layer and drag it on to the canvas.



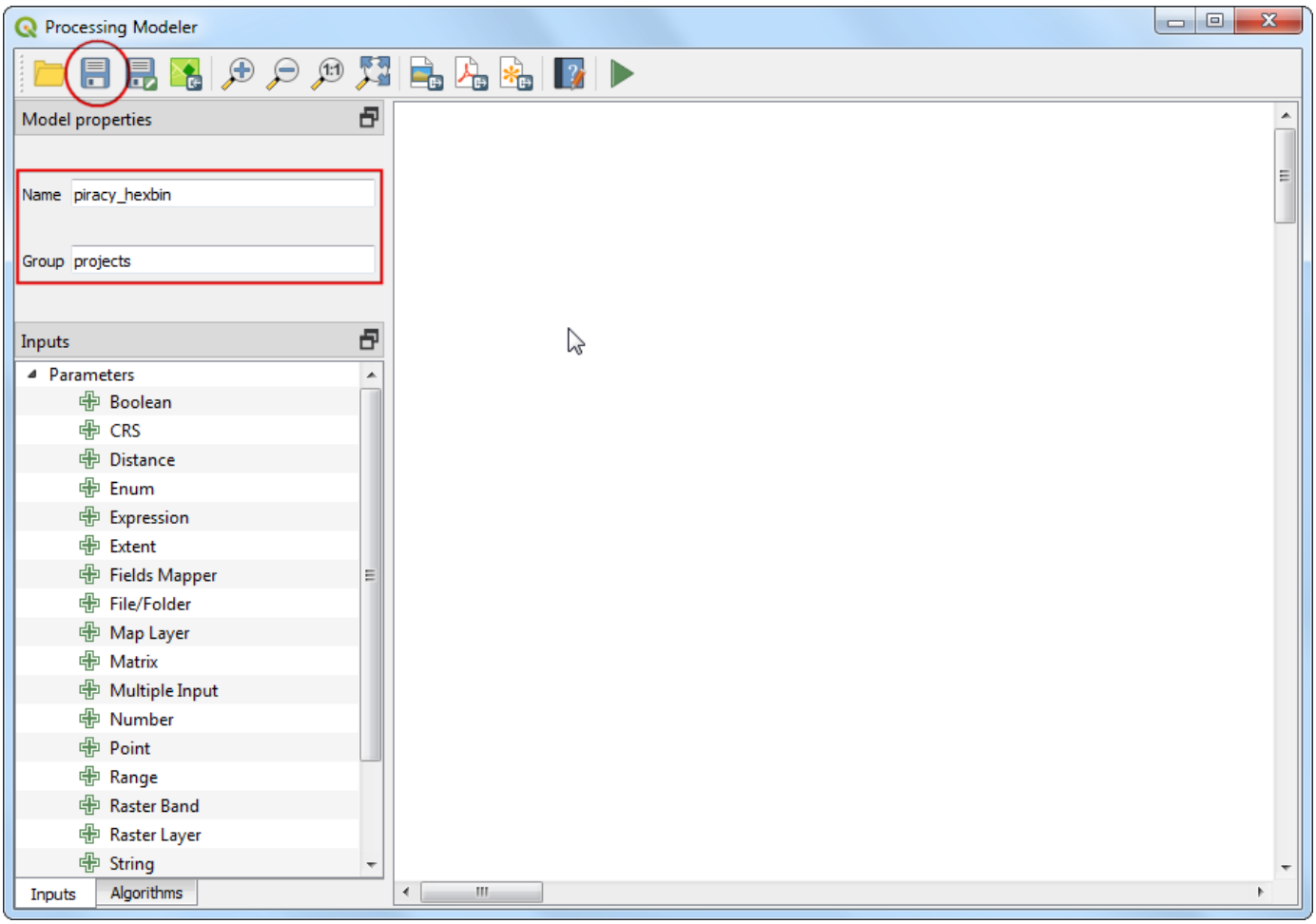
2. The `ASAM_events.shp` layer does not have projection information associated with it, so you will be prompted to select a CRS in the Coordinate Reference System Selector. Here, the points are in the Latitude and Longitude coordinates, so select the `WGS 84` CRS and click OK.



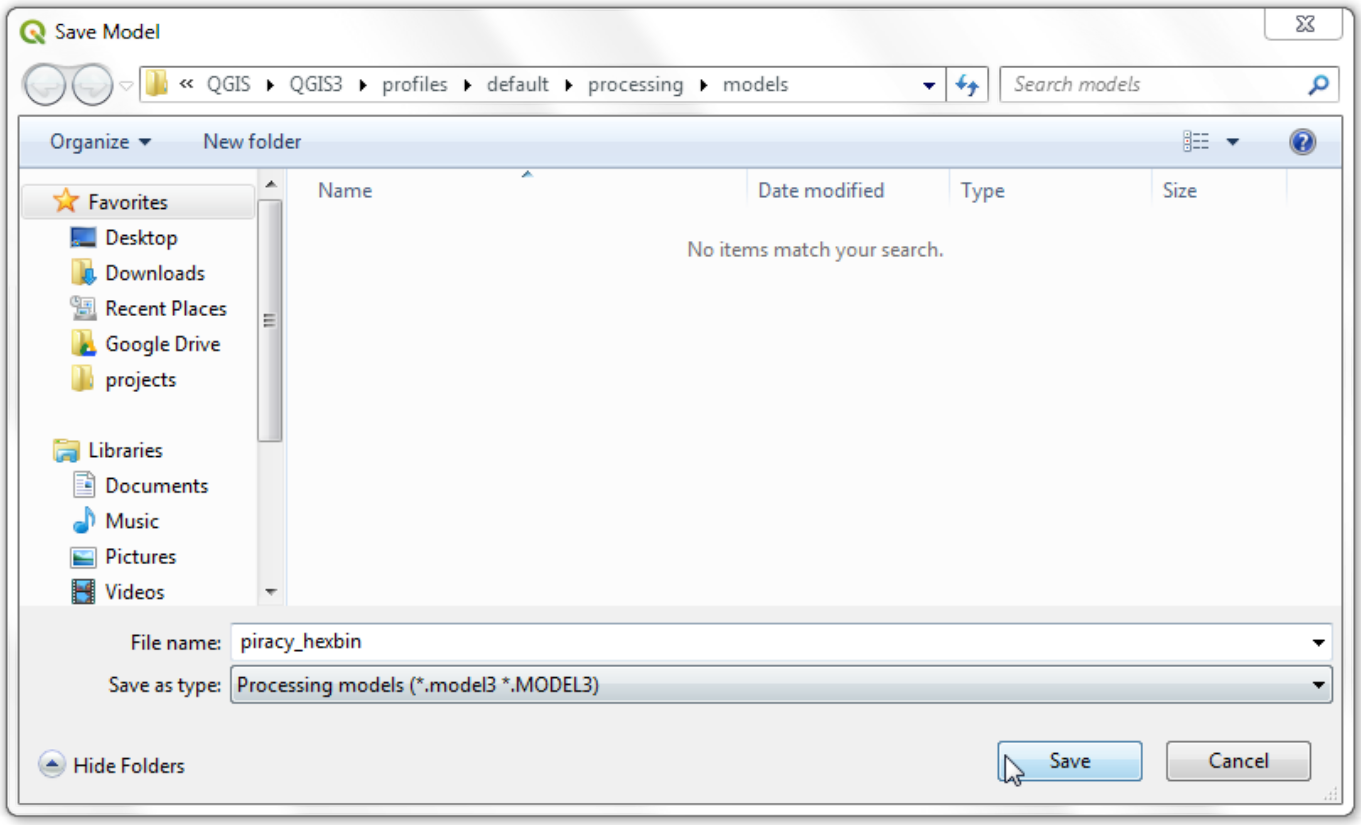
3. Once the layer is loaded, you can see the individual points representing incidents of piracy locations. Let's start building our Processing model to process these layers. Go to Processing ‣ Graphical Modeler....



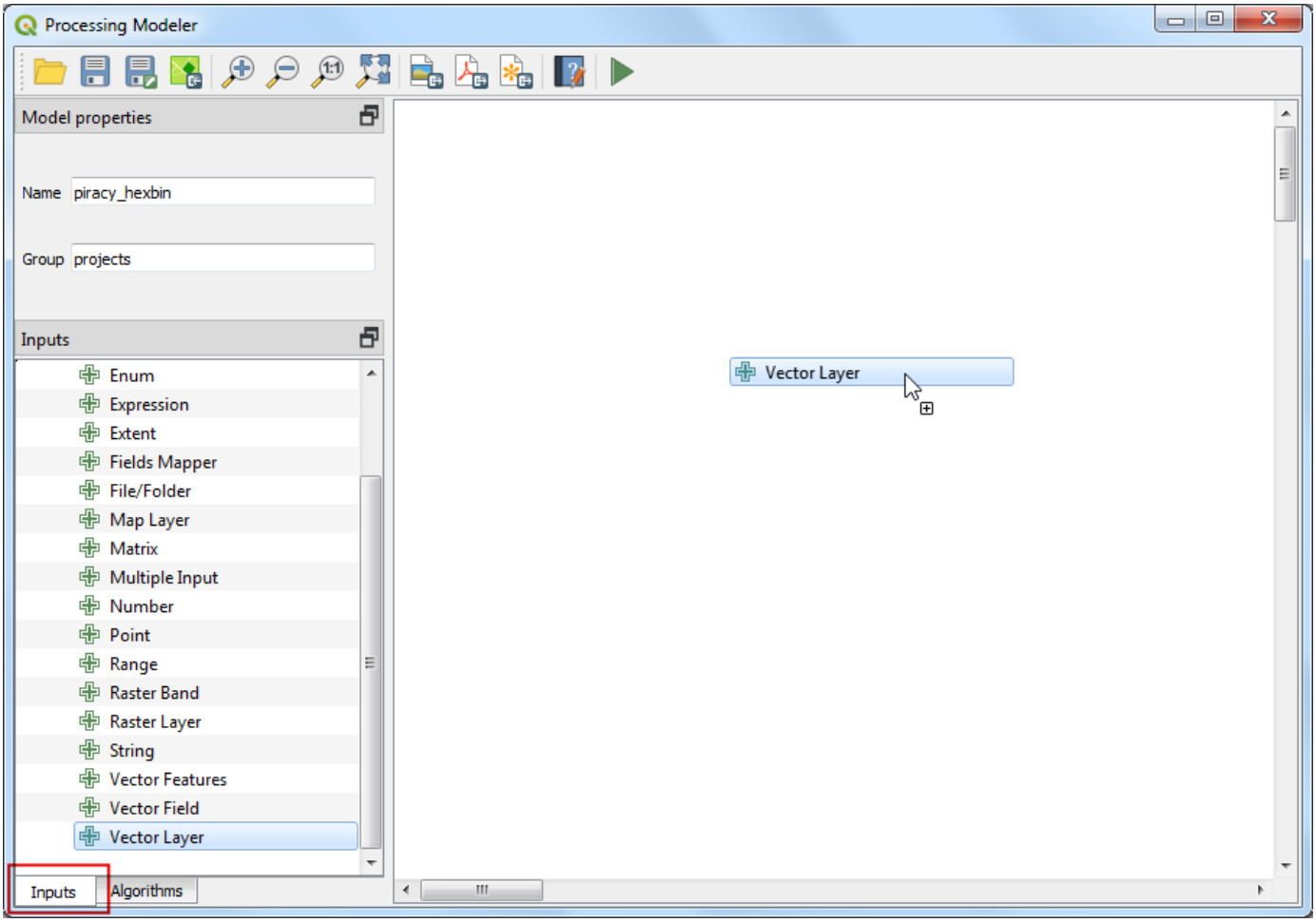
4. In the Processing Modeler dialog, locate the Model Properties panel. Enter `piracy hexbin` as the Name of the model and `projects` as the Groups. Click the Save button.



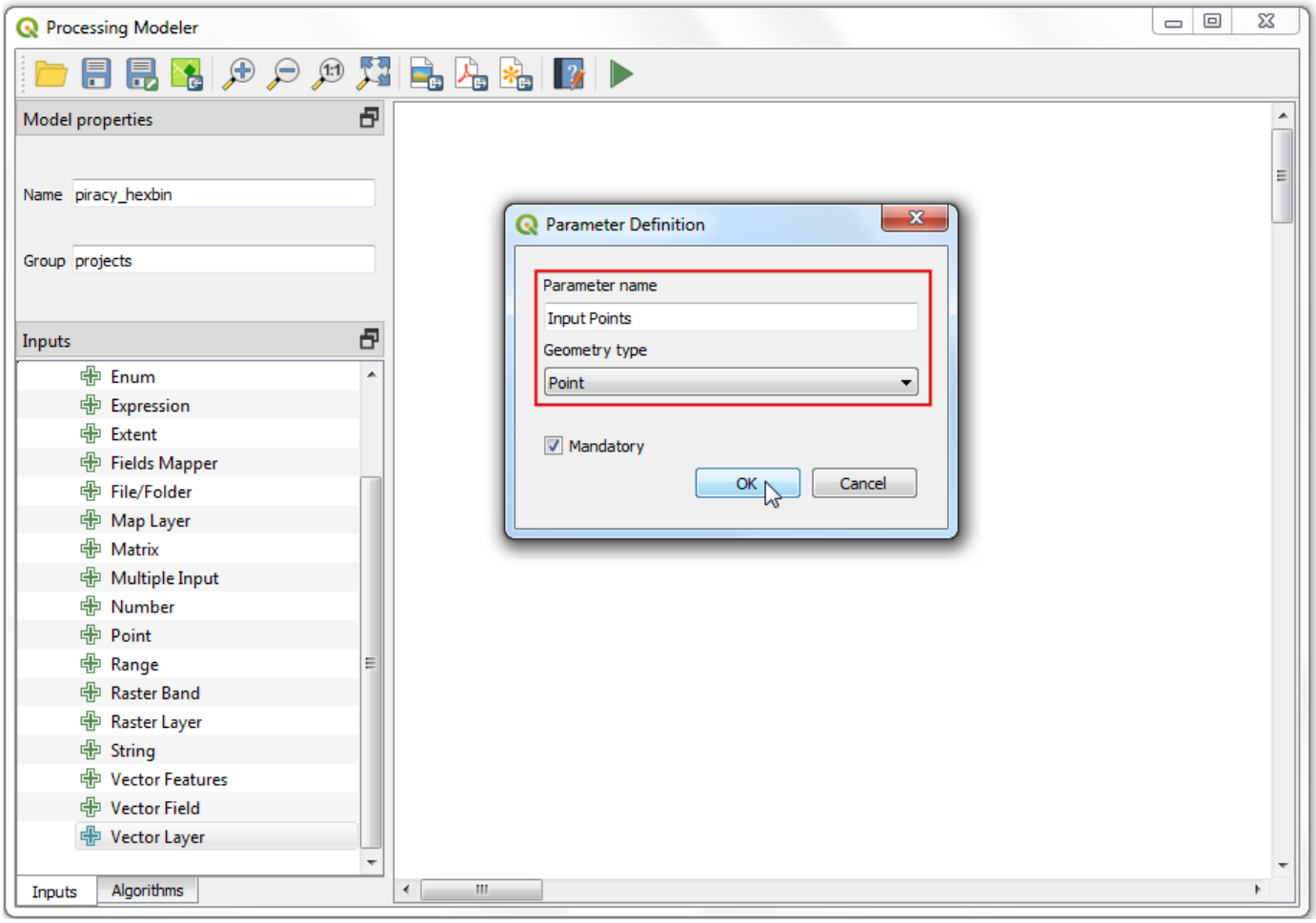
5. Save the model as `piracy_hexbin` .



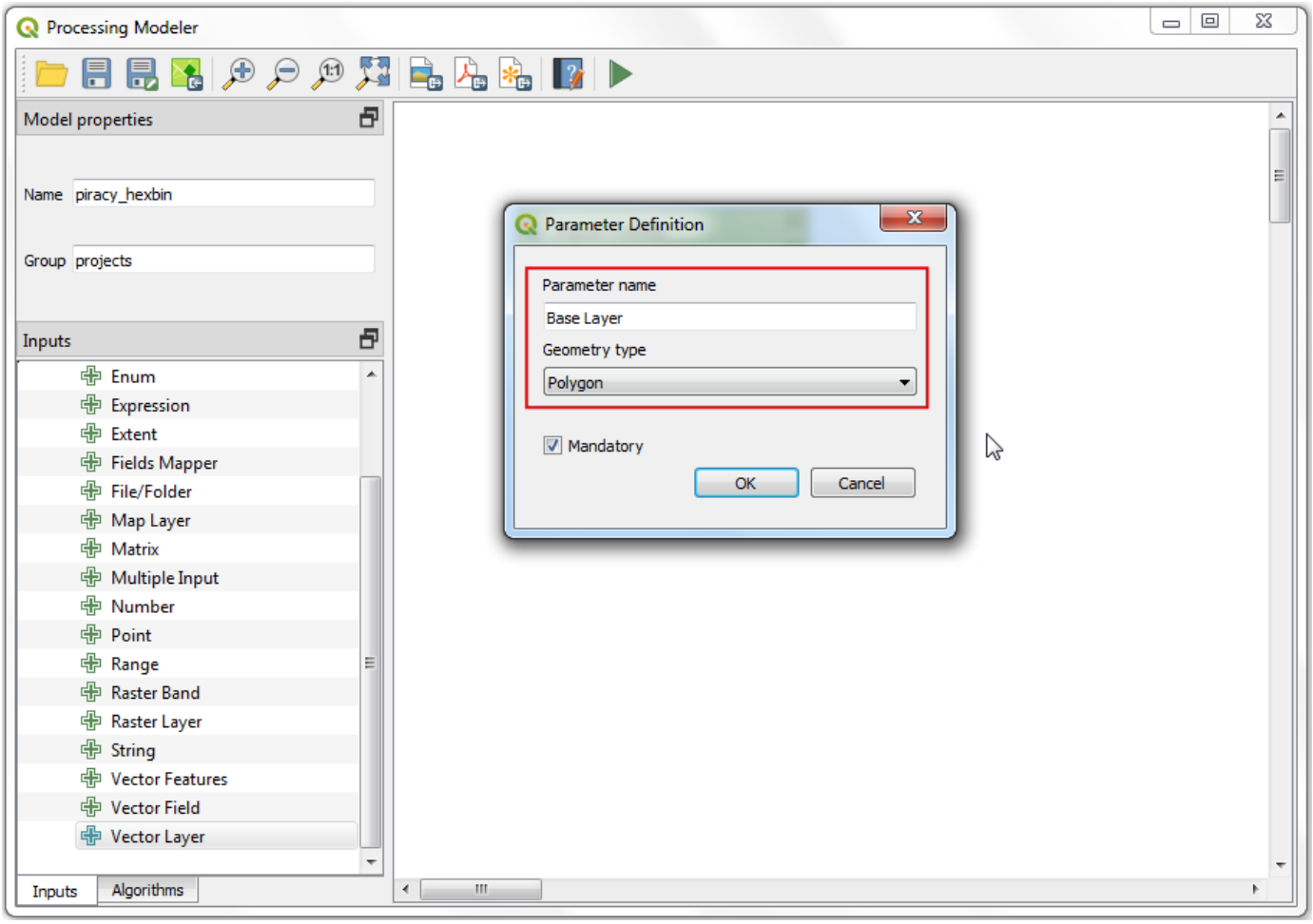
6. Now we can start building a graphical model of our processing pipeline. The Processing modeler dialog contains a left-hand panel and a main canvas. On he left-hand panel, locate the Inputs panel listing various types of input data types. Scroll down and select the + Vector Layer input. Drag it to the canvas.



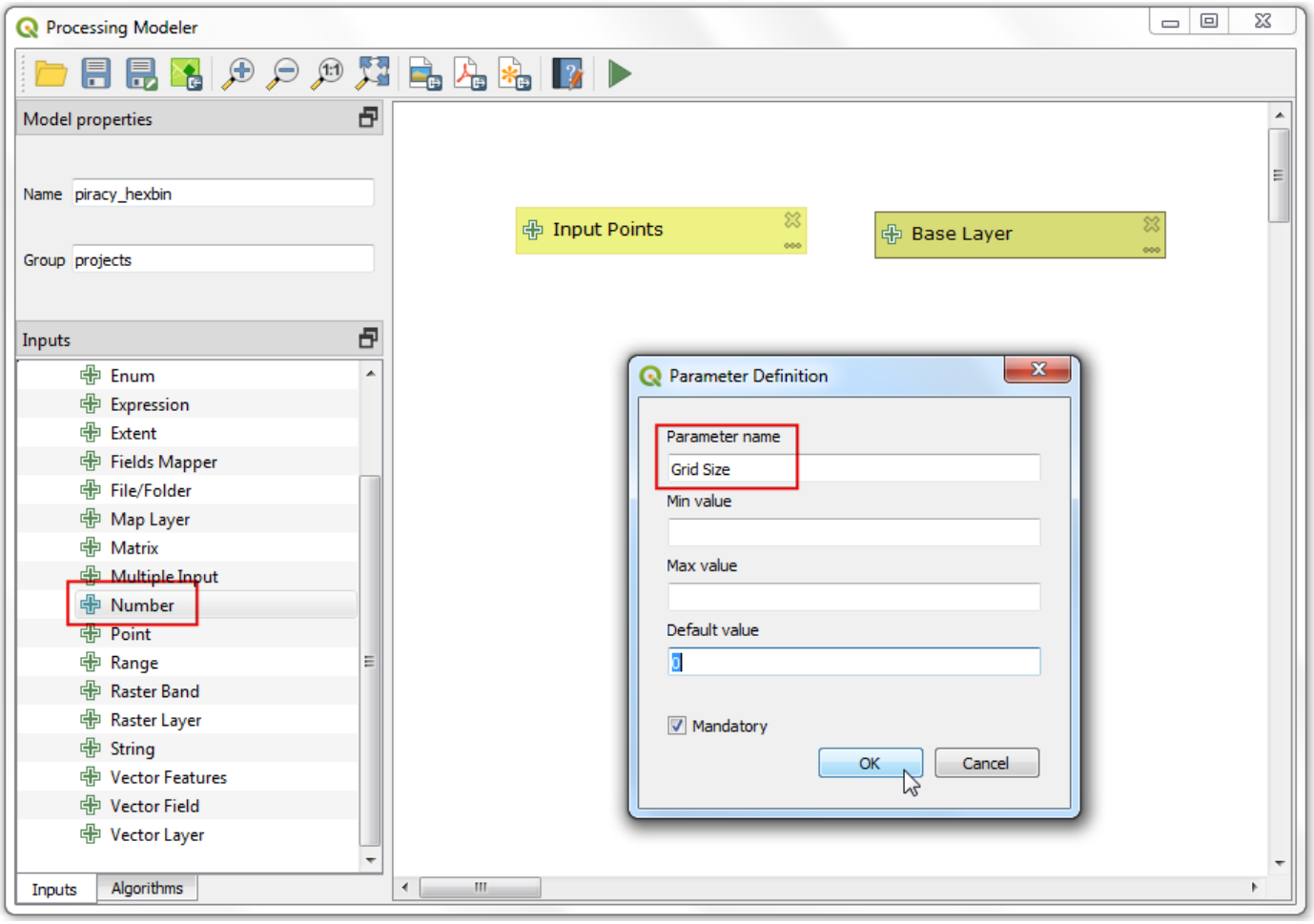
7. Enter `Input Points` as the Parameter name and `Point` as the Geometry type. This input represents the piracy incidents point layer.



8. Next, drag another + Vector Layer input to the canvas. Enter `Base Layer` as the Parameter name and `Polygon` as the Geometry type. This input represents the natural earth global land layer.



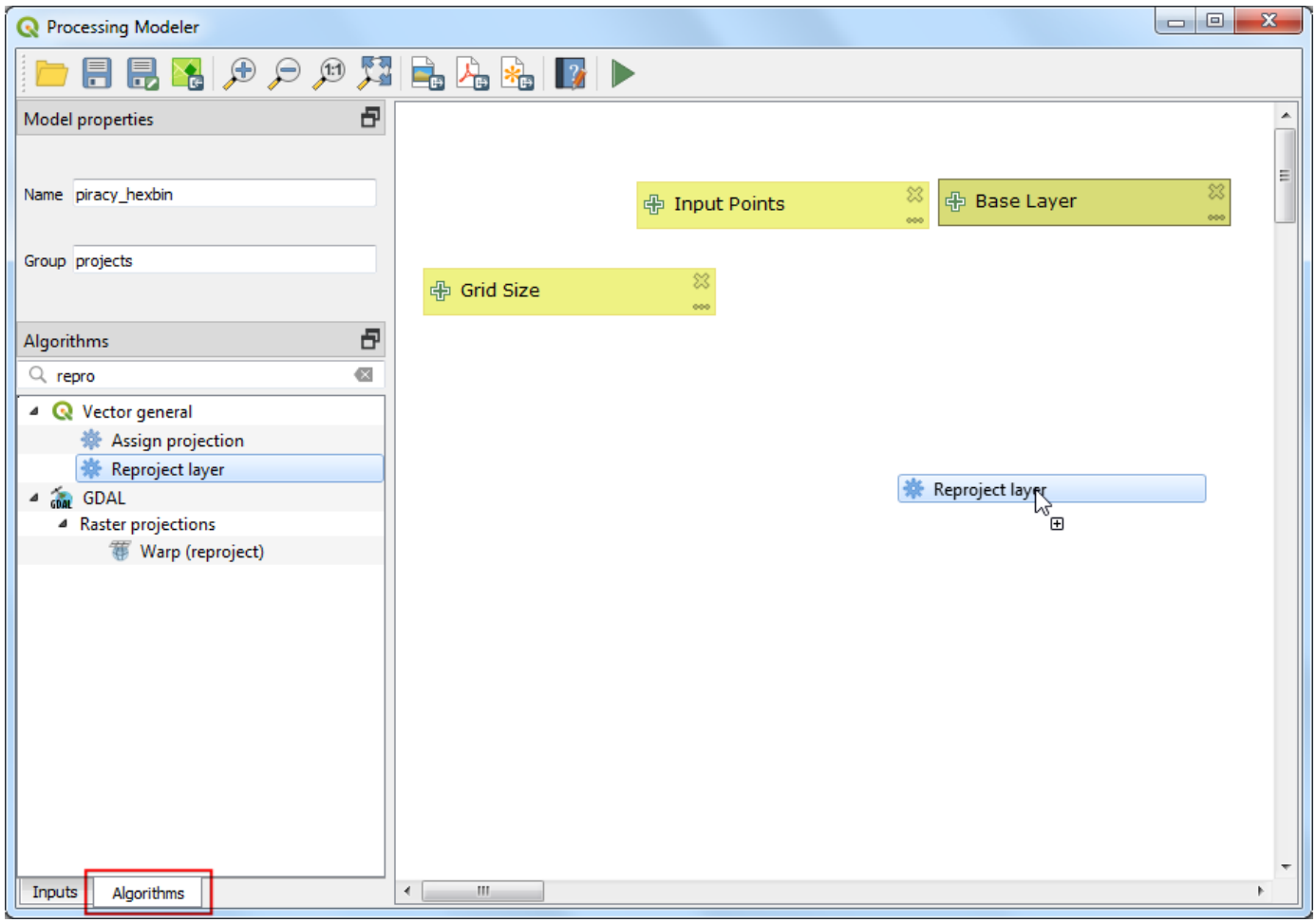
9. As we are generating a global hexagonal grid, we can ask the user to supply us the grid size as an input instead of hard-coding it as part of our model. This way, the user can quickly experiment with different grid sizes without changing the model at all. select a + Number input and drag it to the canvas. Enter `Grid Size` as the Parameter name and click OK.



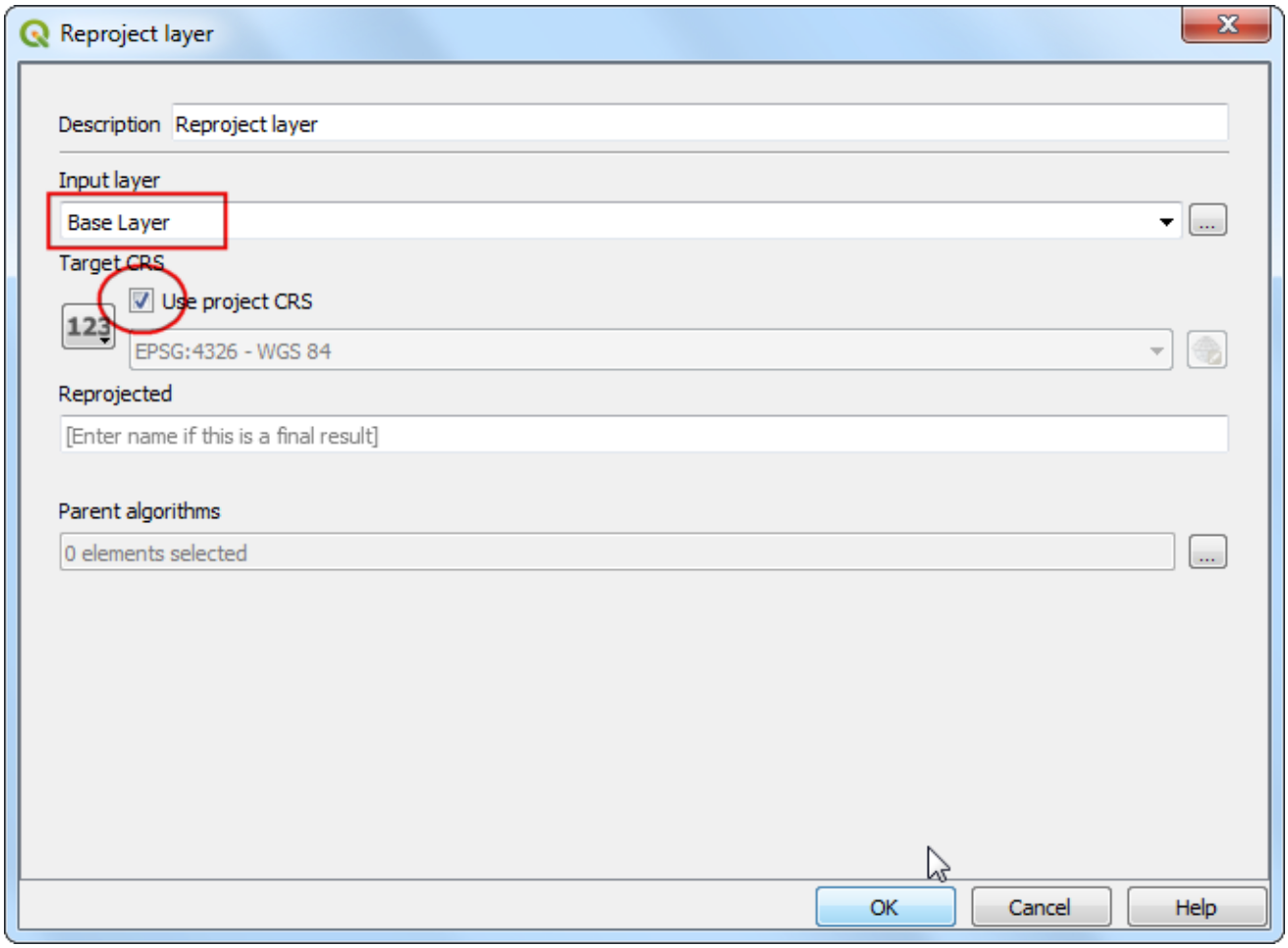
10. Now that we have our user inputs defined, we are ready to add processing steps. All of the processing algorithms are available to you under the Algorithms tab. The first step in our pipeline will be to reproject the base layer to the Project CRS. Search for `Reproject layer` algorithm and drag it to the canvas.

Note

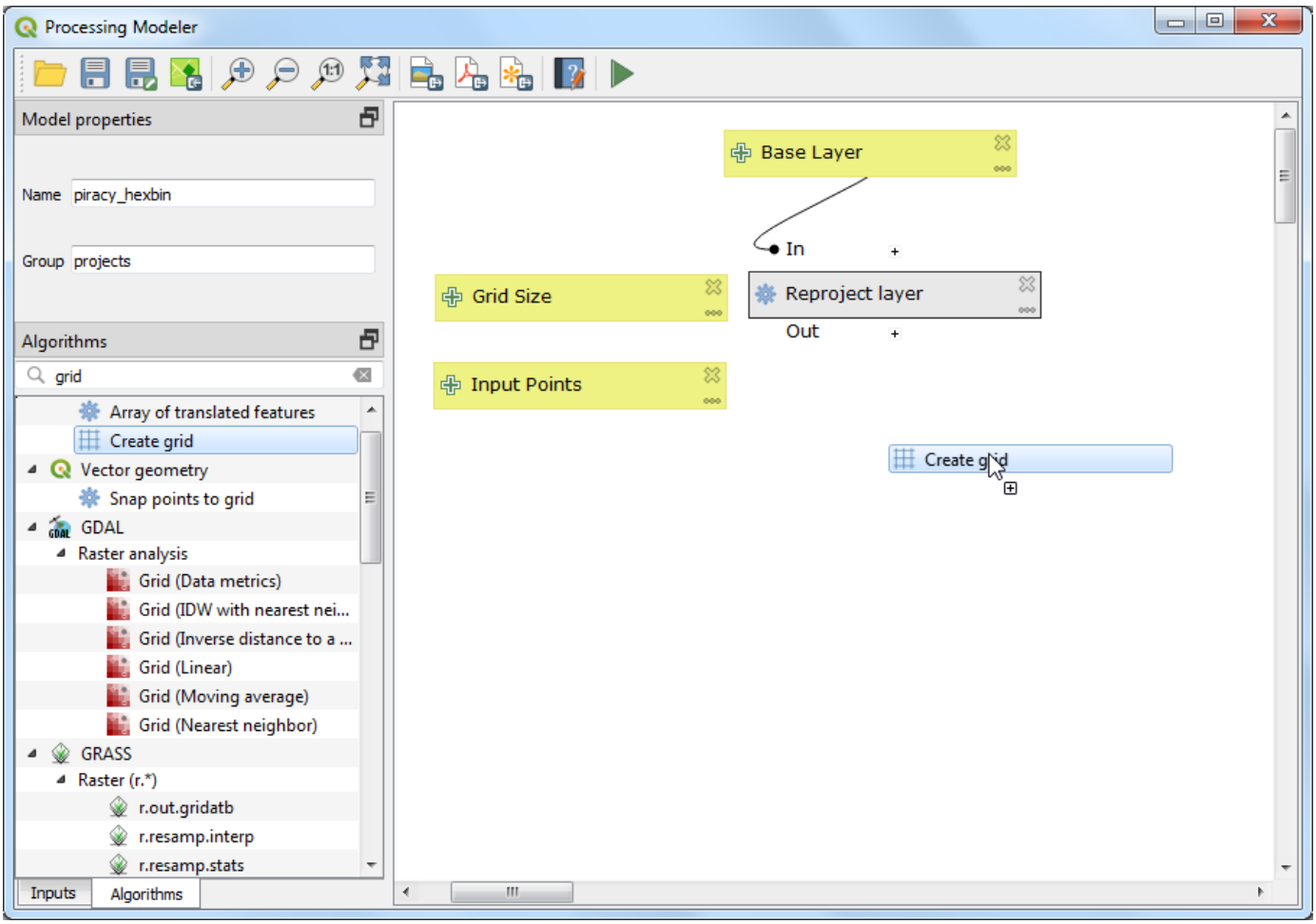
The necessity of this reprojection step will become clear shortly. The grid generation algorithm requires us to specify the extent of the grid in the unit of the Project CRS. We can supply this reprojected layer to compute this extent.



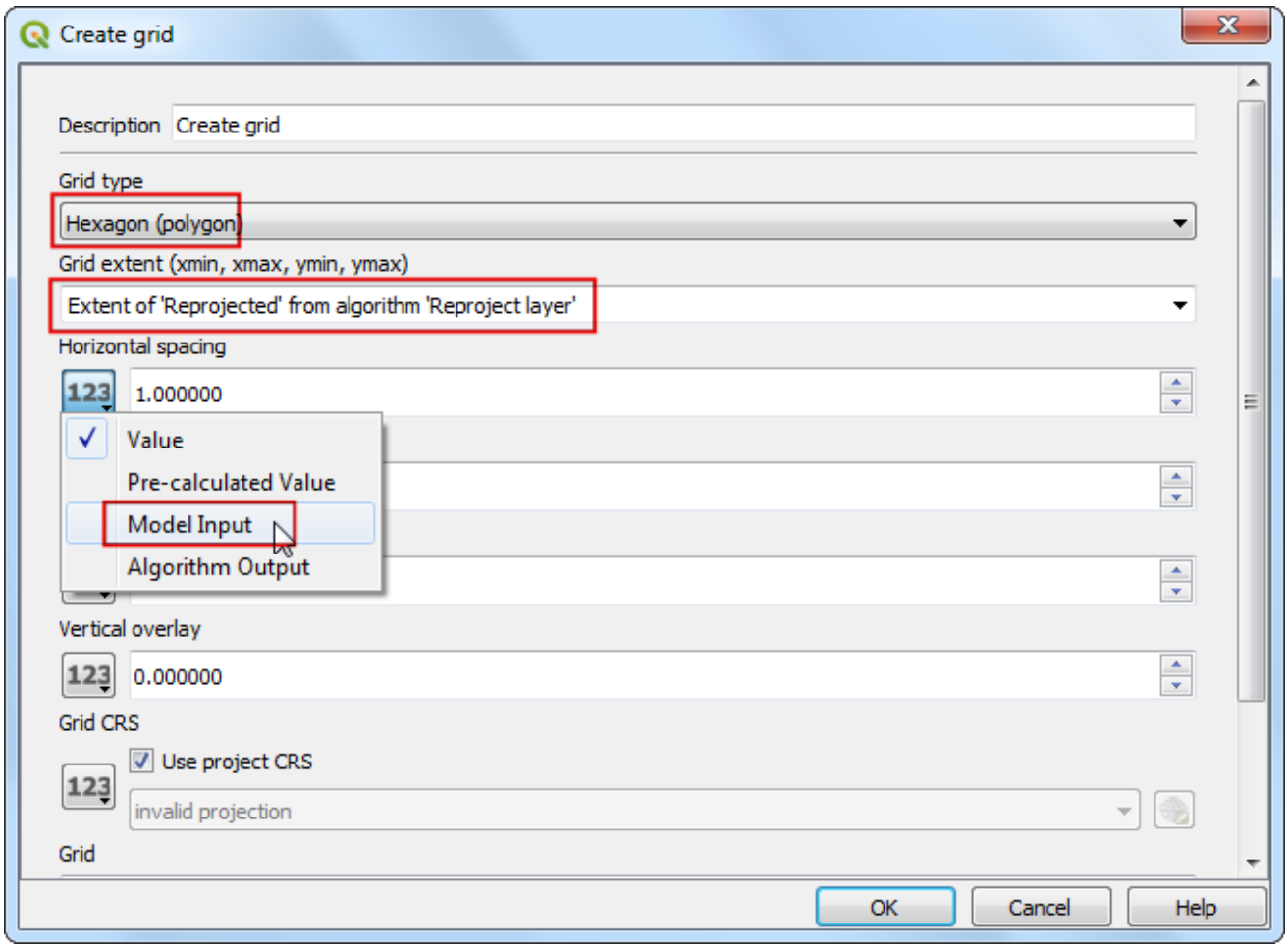
11. In the Reproject layer dialog, select `Base Layer` as the Input layer. Check the Use project CRS as the Target CRS. Click OK.



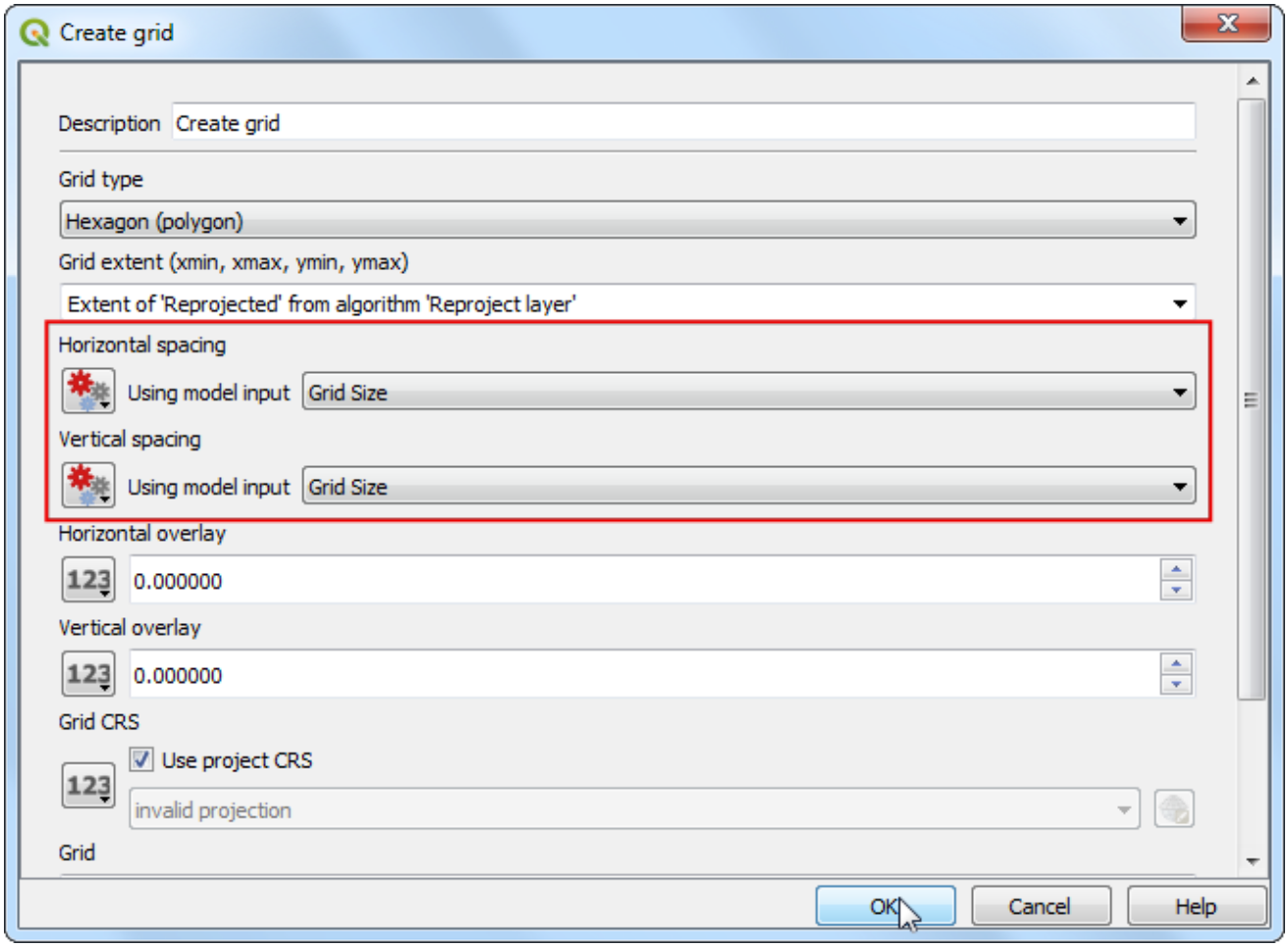
12. In the Processing Modeler canvas, you will notice a connection appear between the + Base Layer input and the Reproject layer algorithm. This connection indicates the flow of our processing pipeline. Next step is to create a hexagonal grid. Search for the `Create grid` algorithm and drag it to the canvas.



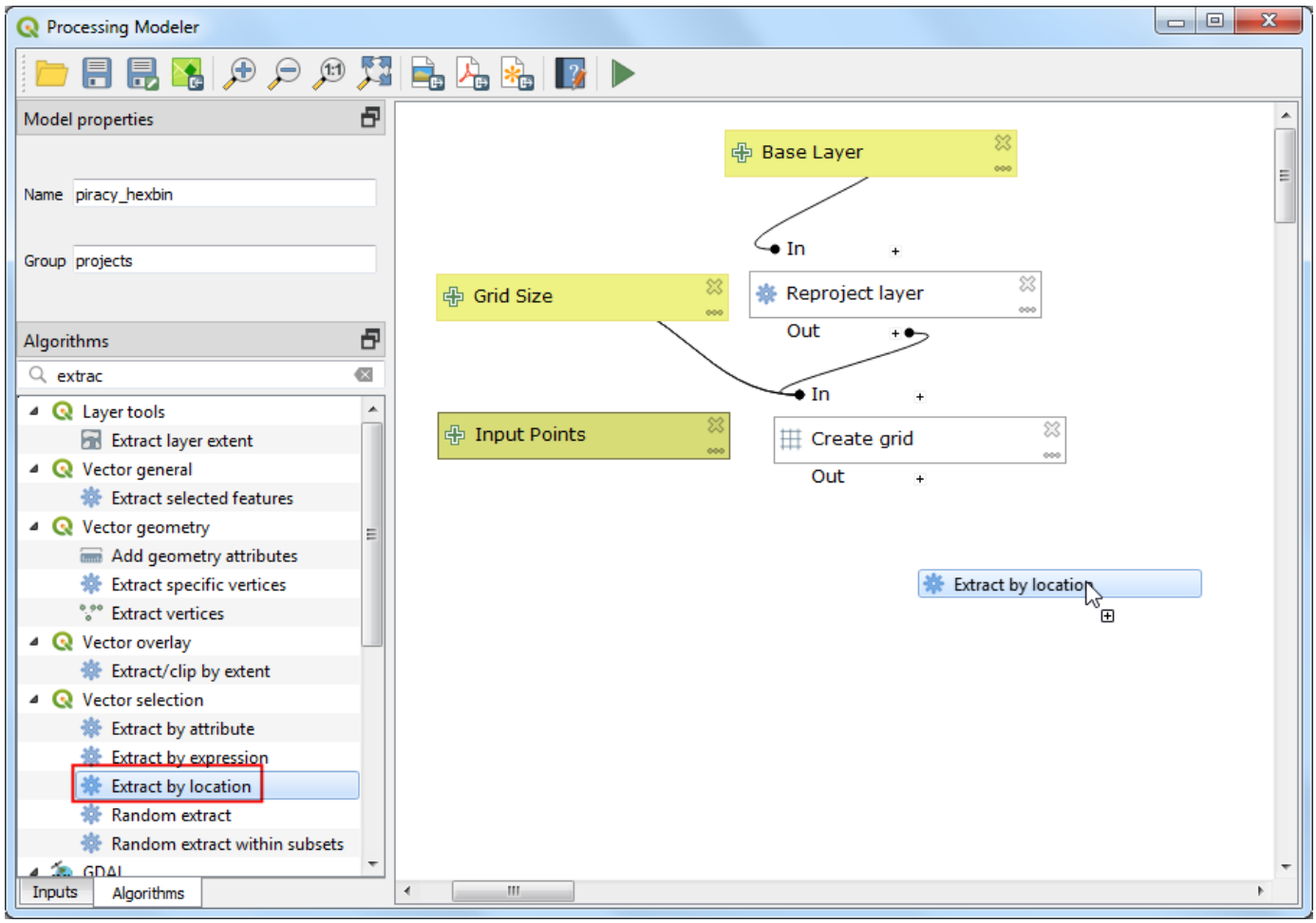
13. In the Generate grid dialog, choose `Hexagon (polygon)` as the Grid type. Select `Extent of 'Reprojected' from algorithm 'Reproject Layer'` as the Grid extent. Click the 123 button under the Horizonal spacing label and choose Model input.



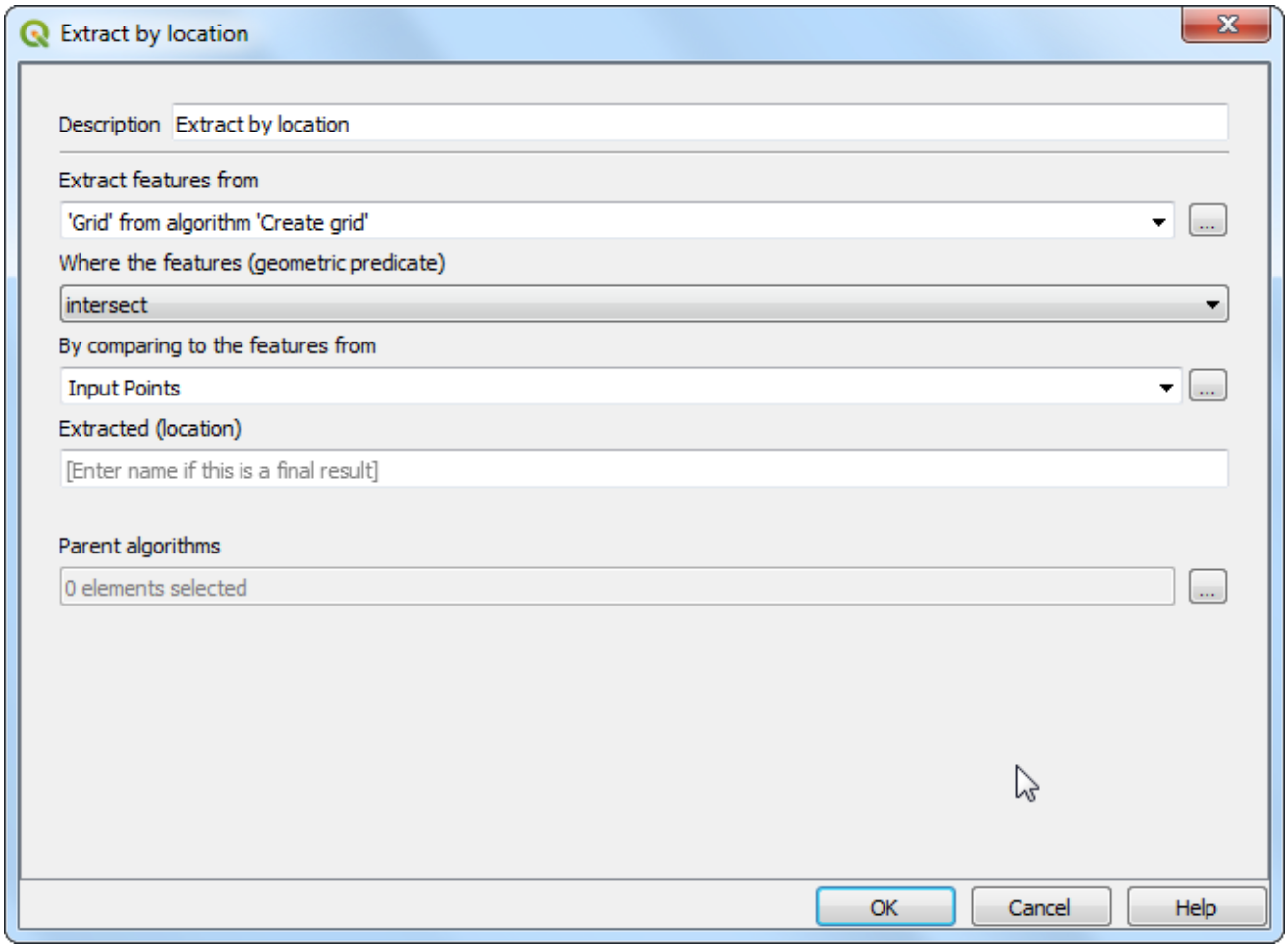
14. Select `Grid Size` input for Using model input. Repeat the same process for Vertical Spacing. Click OK.



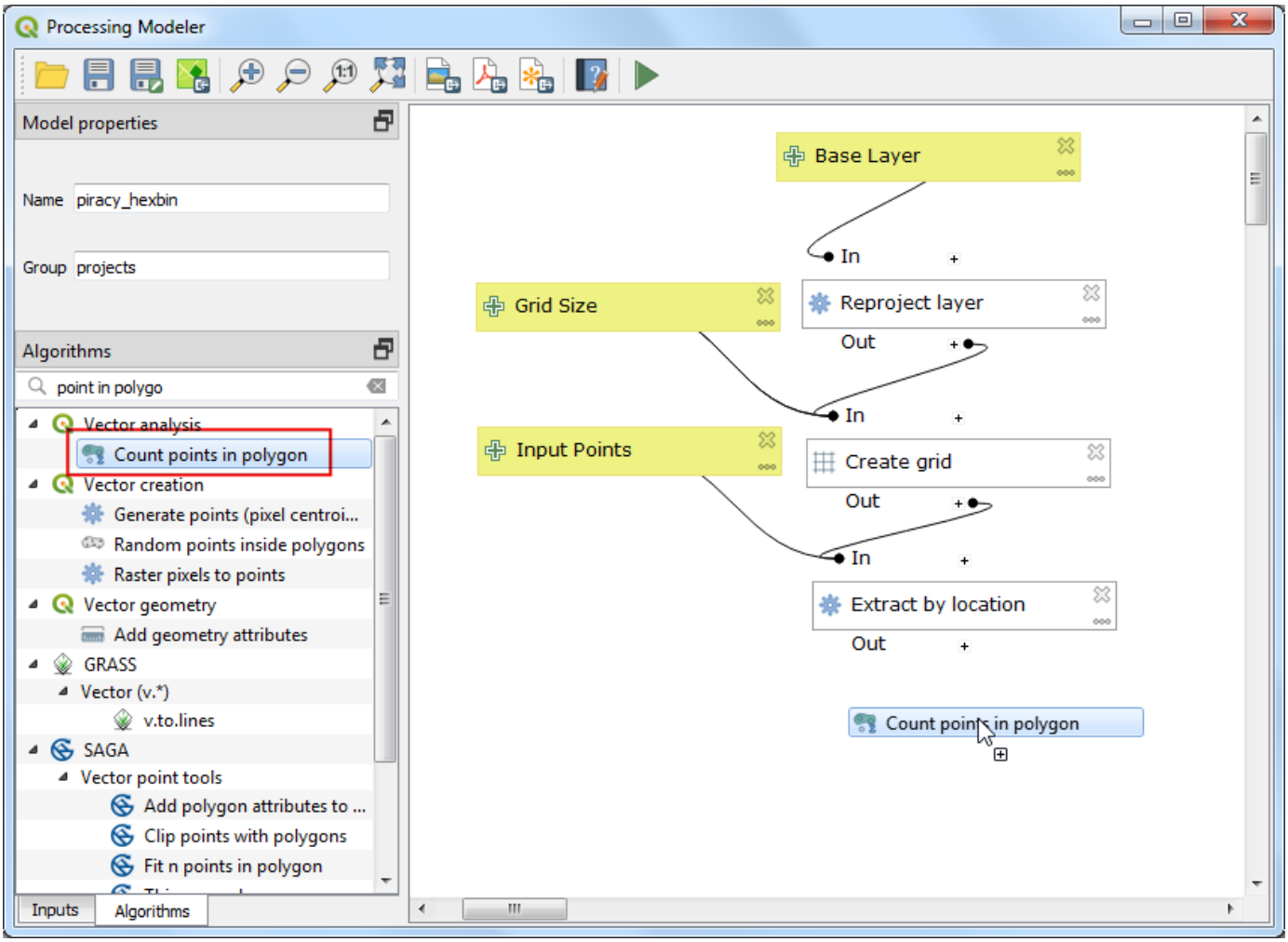
15. At this point, we have a global hexagonal grid. The grid spans the full extent of the base layer, including land areas and places where there are no points. Let's filter out those grid polygons where there are no input points. Search for `Extract by location` algorithm and drag it to the canvas.



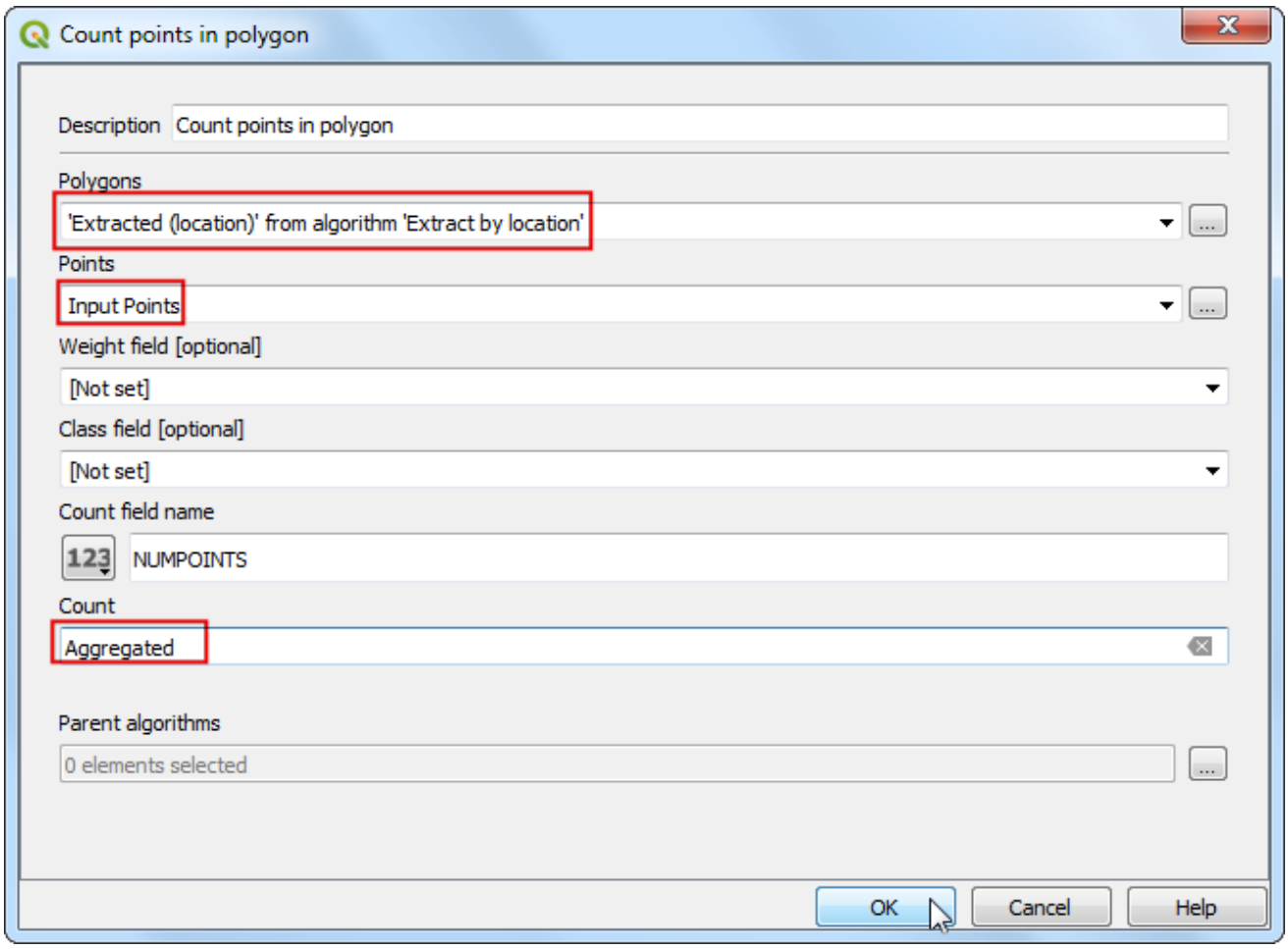
16. For Extract features from, select `'Grid' from algorithm 'Generate Grid'`, Where the features (geometric predicate) as `Intersect` and By compating to the features from as `Input points`. Click OK.



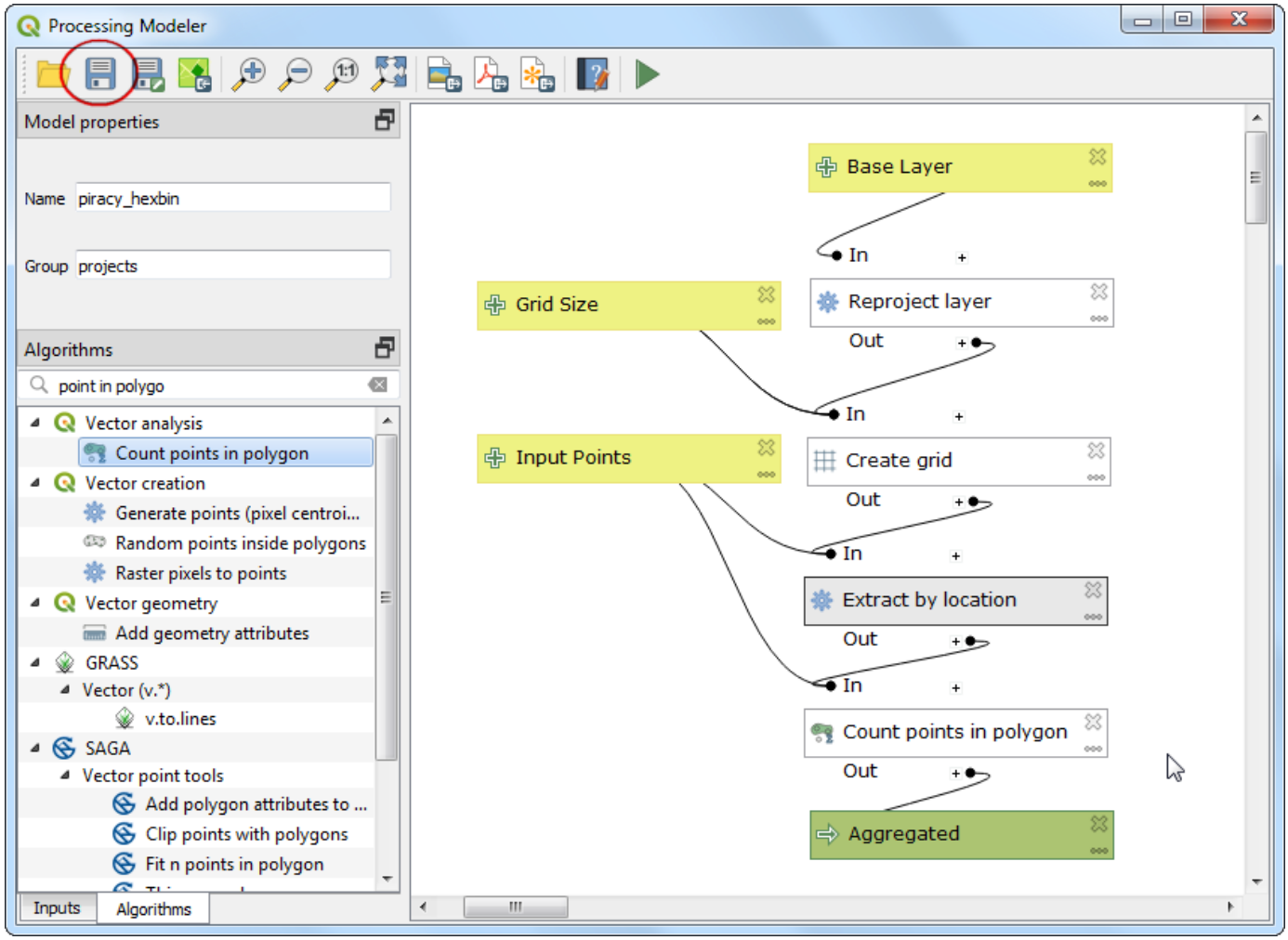
17. Now we have only those grid polygons that contain some input points. To aggregate these points, we will use `Count points in polygon` algorithm. Search and drag it to the canvas.



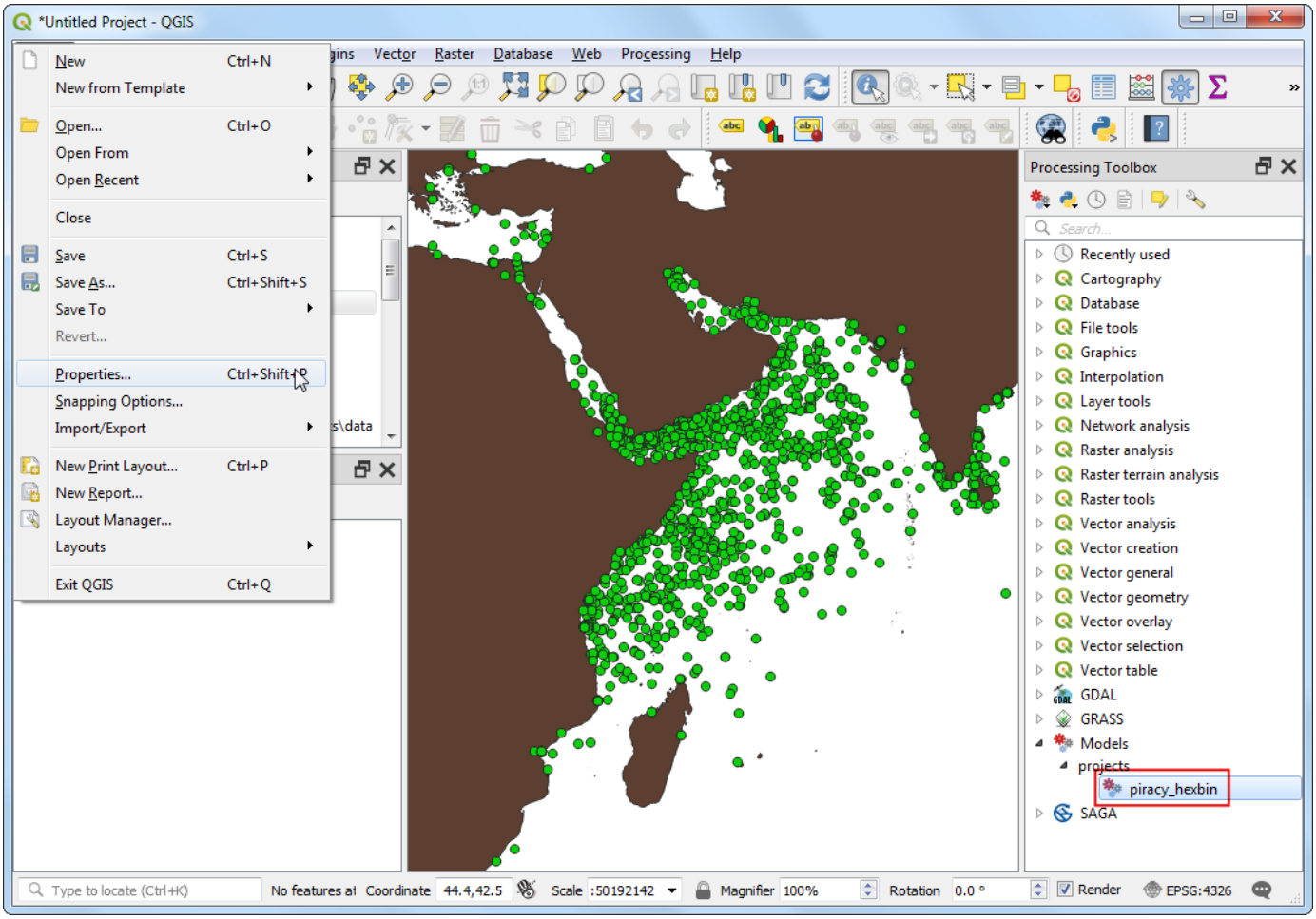
18. Select `'Extracted (location)' from algorithm 'Extract by location'` as the value for Polygons. The Points layer would be `Input Points` . At the bottom, name the Count output layer as `Aggregated` . Click OK.



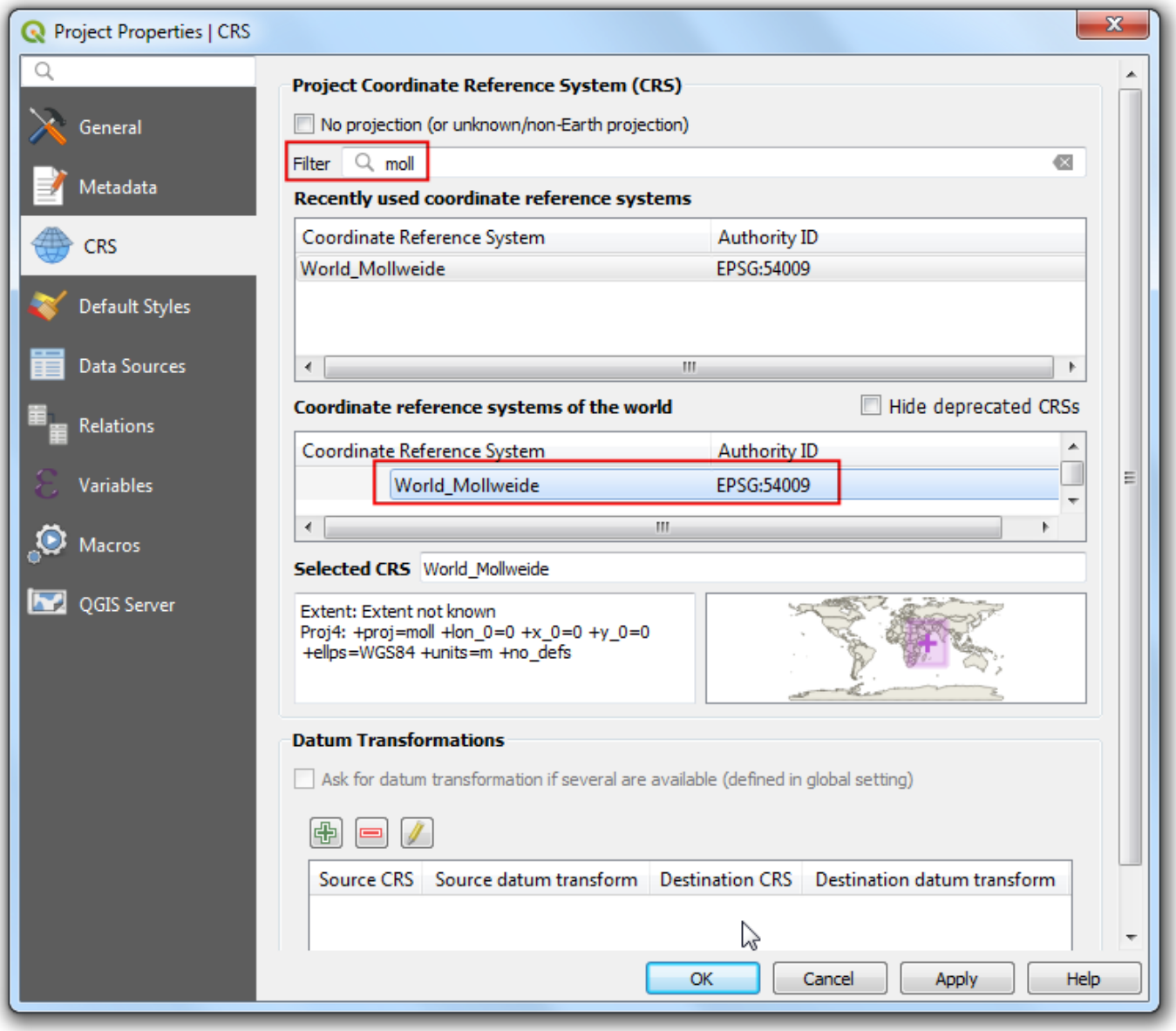
19. The model is now complete. Click the Save button.



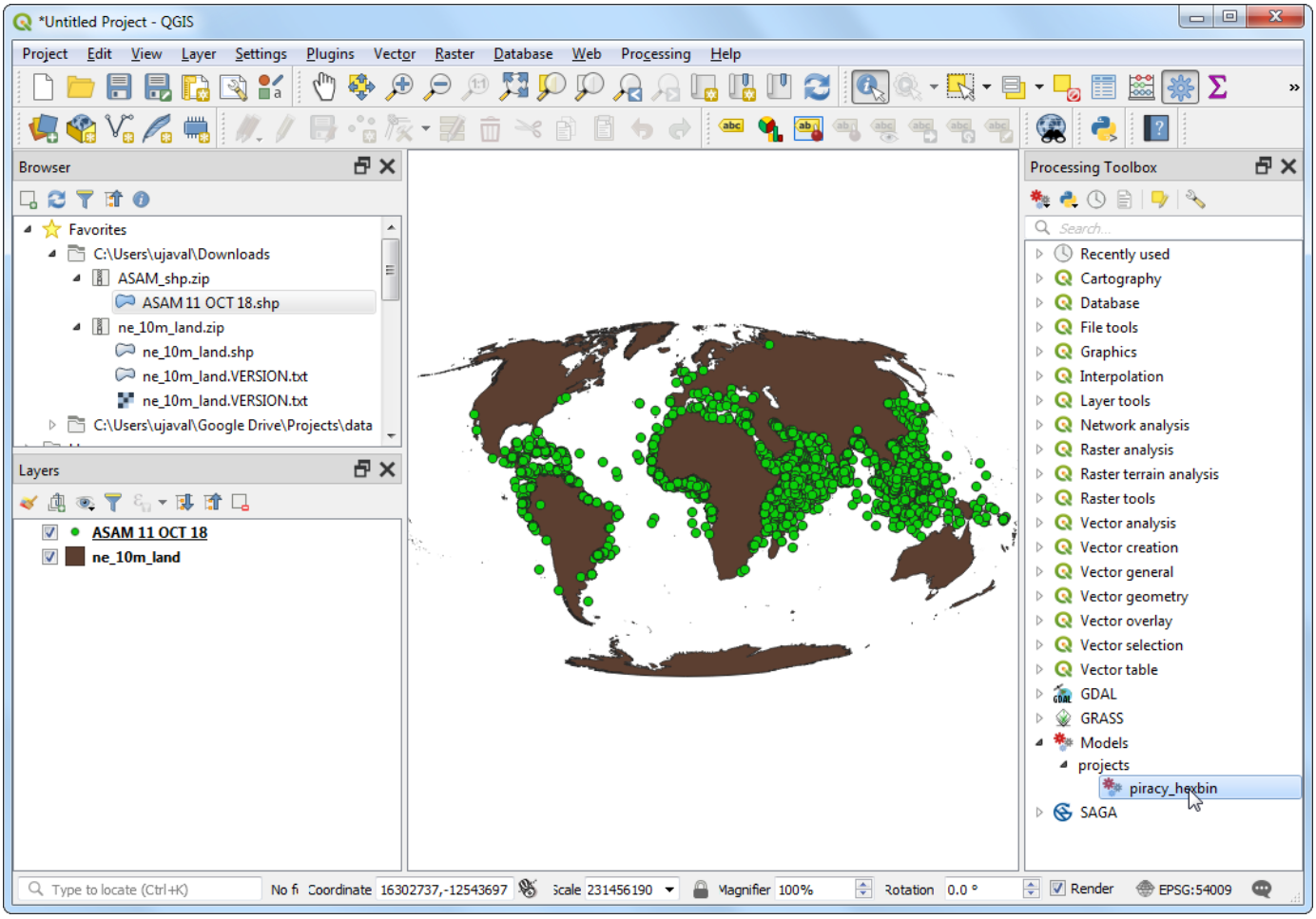
20. Switch to the main QGIS window. You can find your newly created model in the Processing Toolbox under Models ▸ projects ▸ piracy_hexbin. Now it is time to run and test the model. As our goal is to aggregate the input points over hexagonal grids, it is important that the grids are generated using a equal-area projection. This will ensure that regardless of the location of the grid, it will cover exactly the same area. Our model doesn't explicitely ask for a CRS, but uses whatever CRS is set as the **Project CRS**. Let's choose a global equal area projection as the Project CRS. Go to Project ▸ Properties.



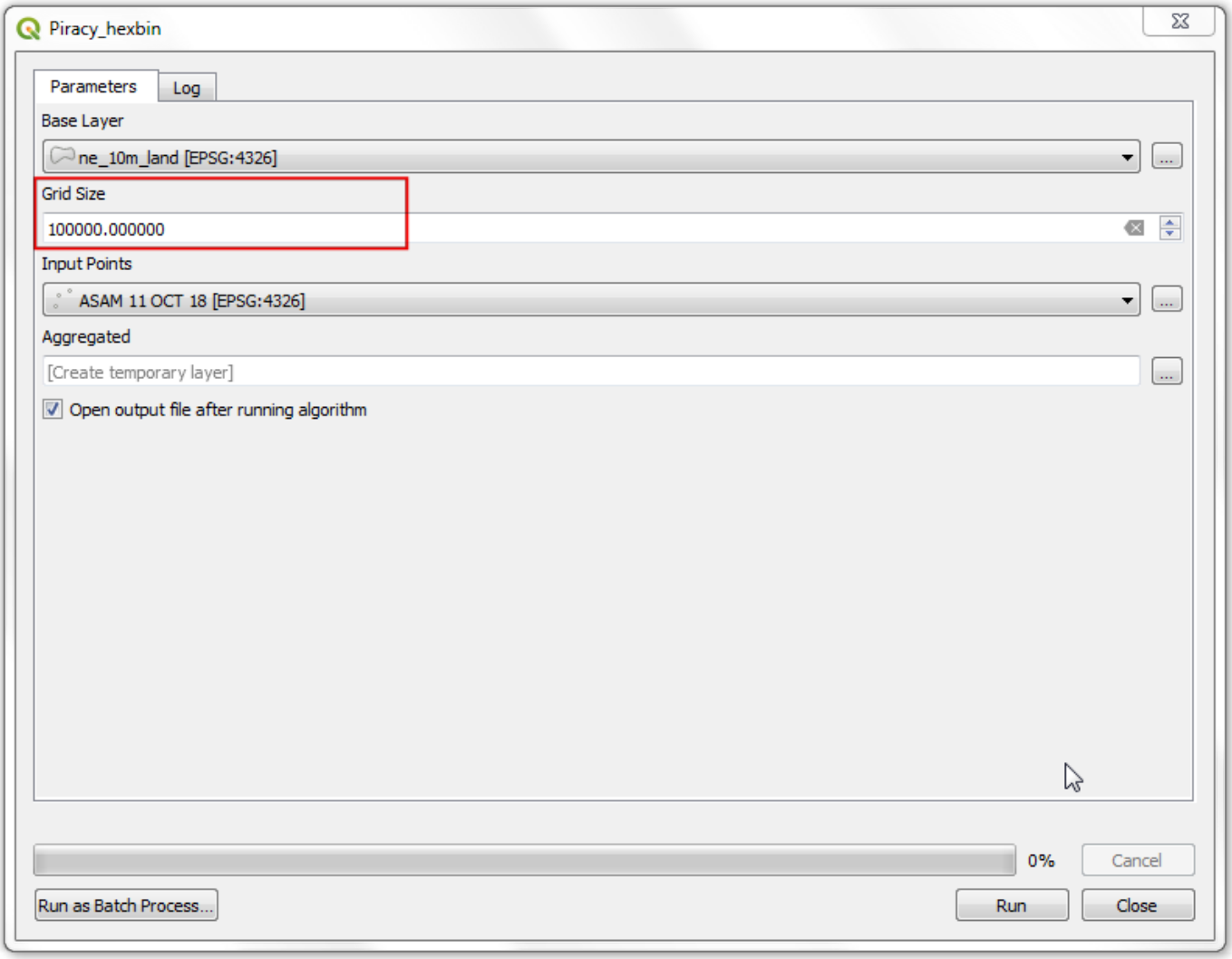
21. In the Project Properties dialog, switch to the CRS tab. We will use a global **Mollweide** projection for this exercise which is a equal area projection. Search for `Mollweide` in the Filter box and select `World_Mollweide EPSG:54009` as the CRS. Click OK.



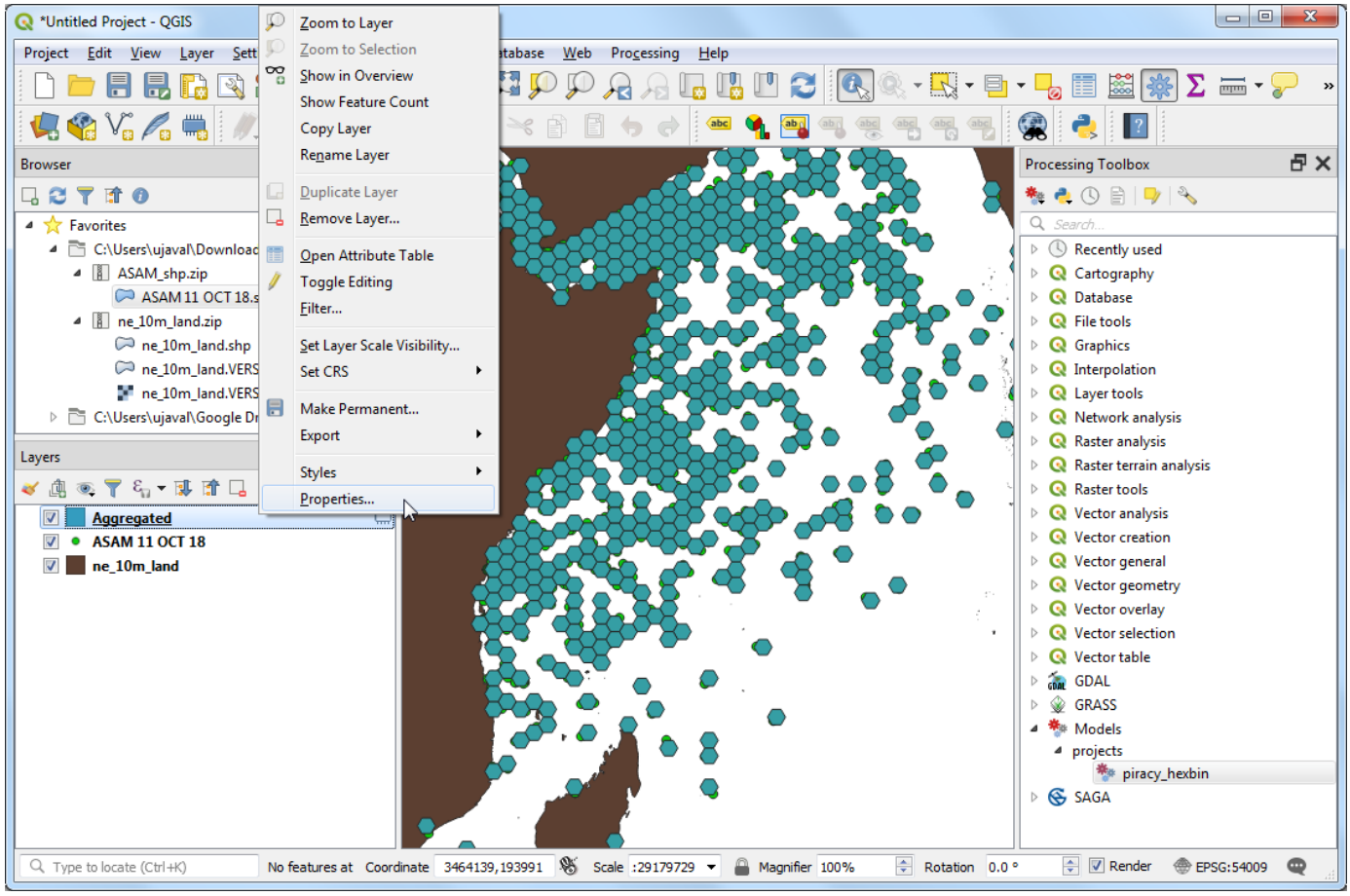
22. You will see the layers getting reprojected on-the-fly to the selected CRS. Locate the `piracy_hexbin` model in the Processing Toolbox and double-click it.



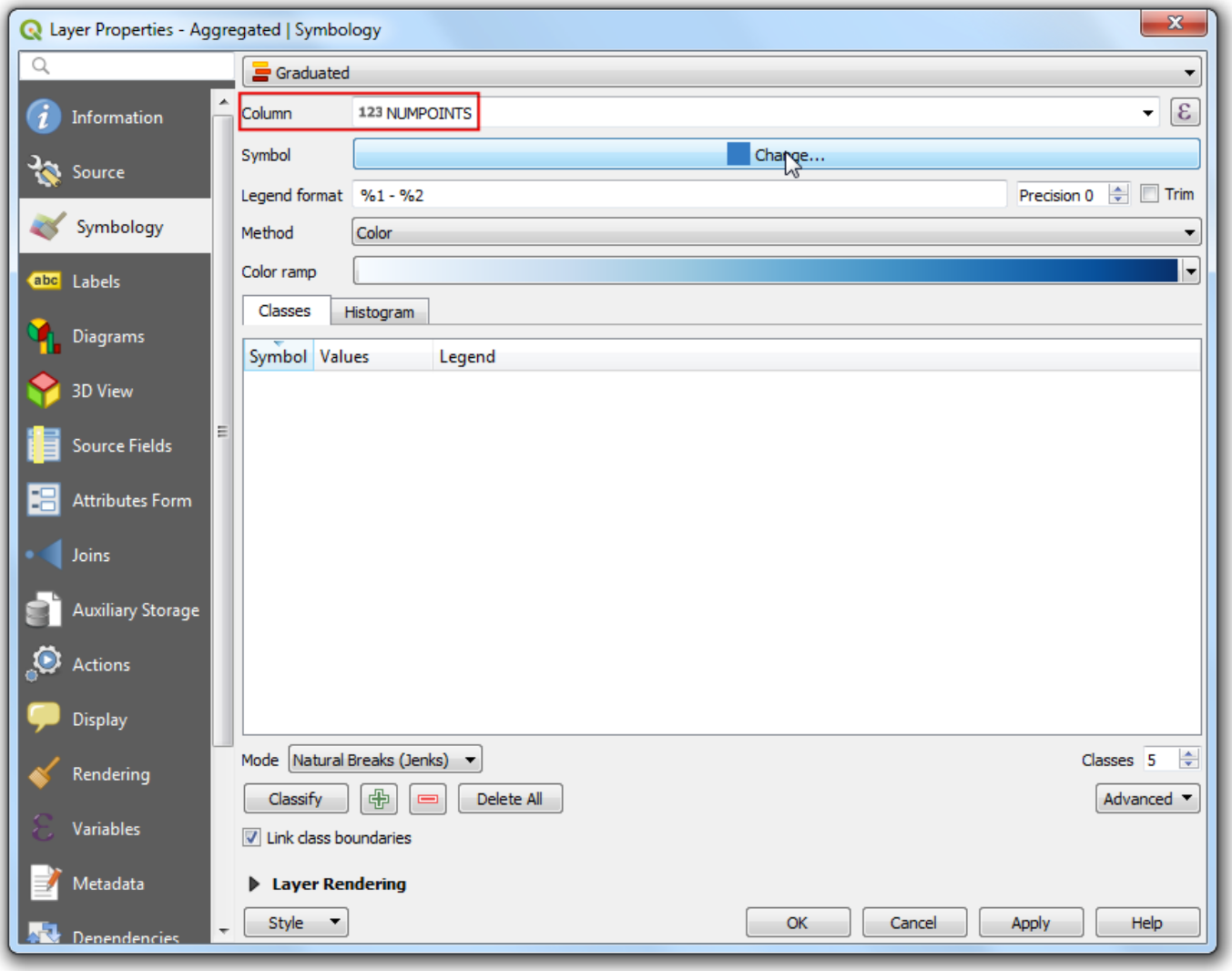
23. Our Base Layer is the `ne_10m_land` and the Input Points layer is `ASAM_events`. The Grid Size needs to be specified in the units of the selected CRS. The World_Mollweide CRS unit is meters, so we specify `100000` m (100 Kms) as the Grid Size. Click Run to start the processing pipeline. Once the process finishes, click Close.



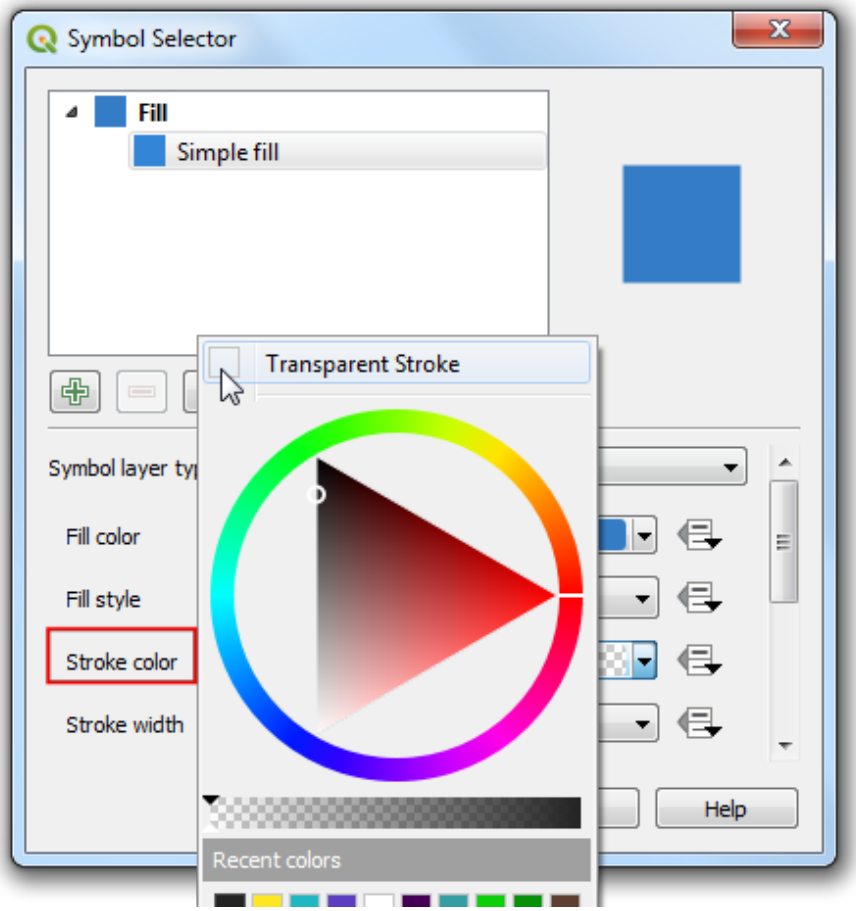
24. You will see a new layer `Aggregated` loaded as the result of the model. As you explore, you will notice the layer contains an attribute called *NUMPOINTS* containing the number of piracy incidents points contained within that grid feature. Let's style this layer to display this information better. Right-click the `Aggregated` layer and select Properties.



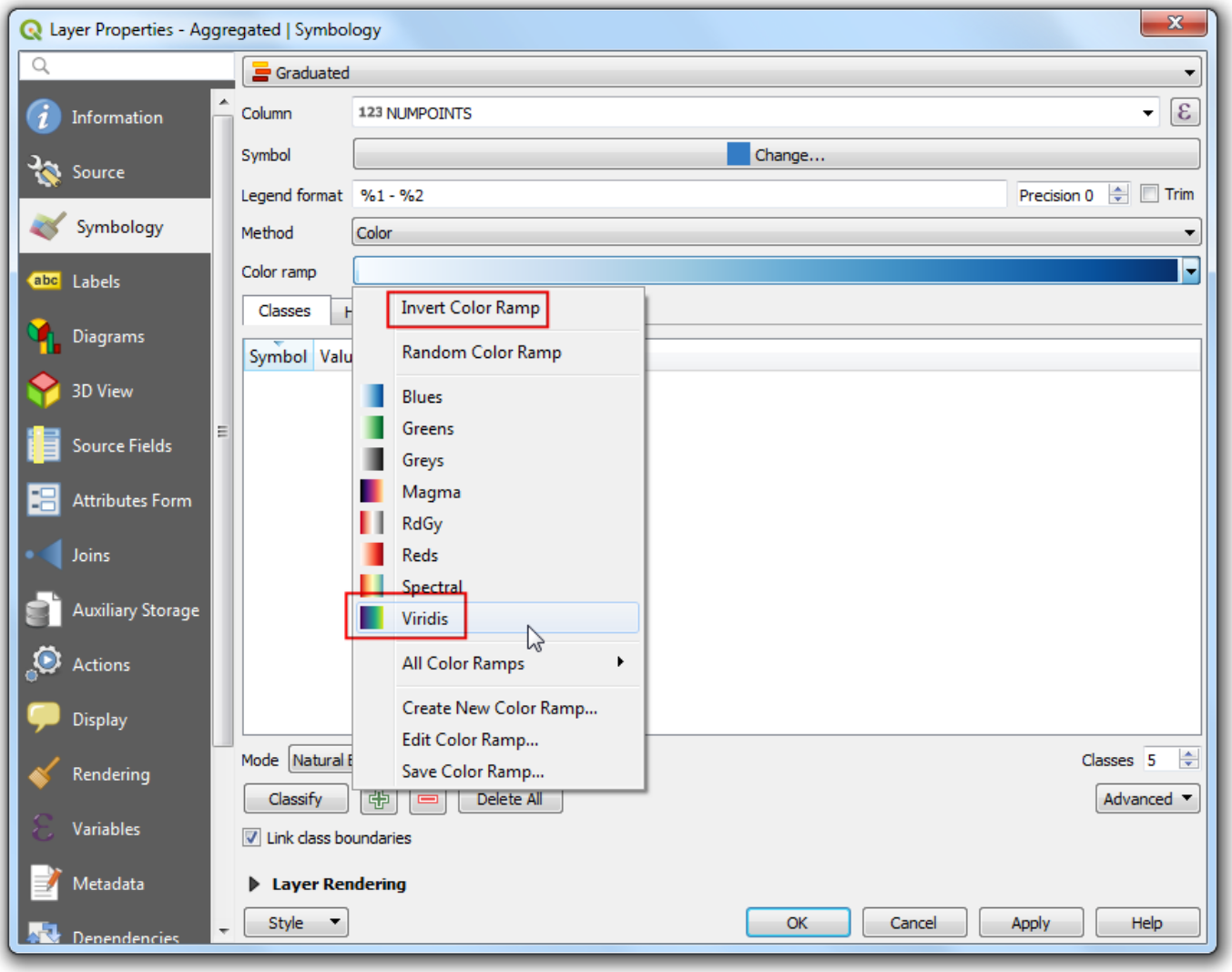
25. Switch to the Symbology tab. Select `Graduated` symbology and `NUMPOINTS` as the Column. Click `Change..` next to Symbol label.



26. Select Simple fill symbol and check the Transparent Stroke button under Stroke color. This is to make the hexagon edges transparent.



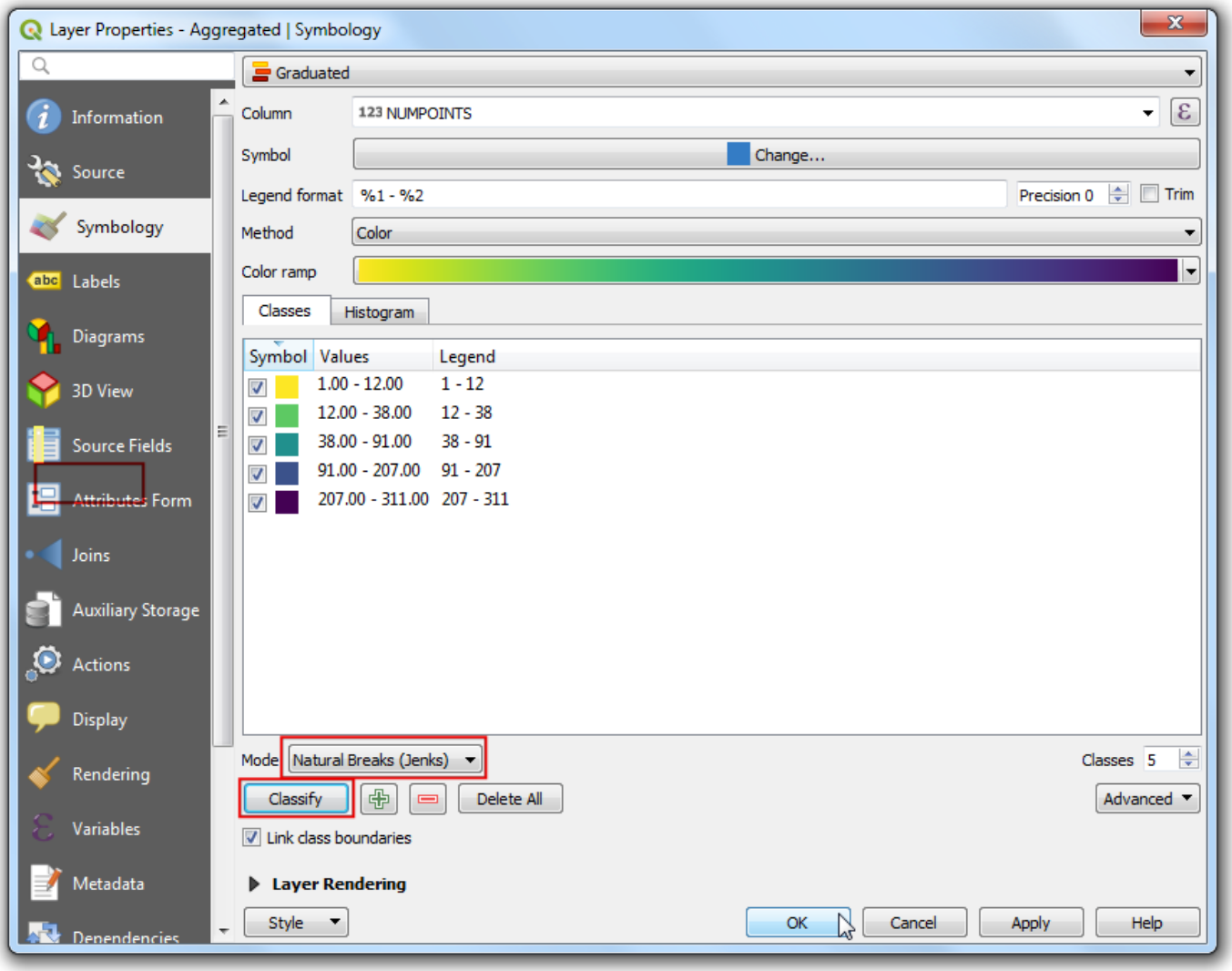
27. Click the dropdown next to Color ramp and select the `Viridis` ramp. Click the dropdown again and select Invert Color Ramp to reverse the order of color.



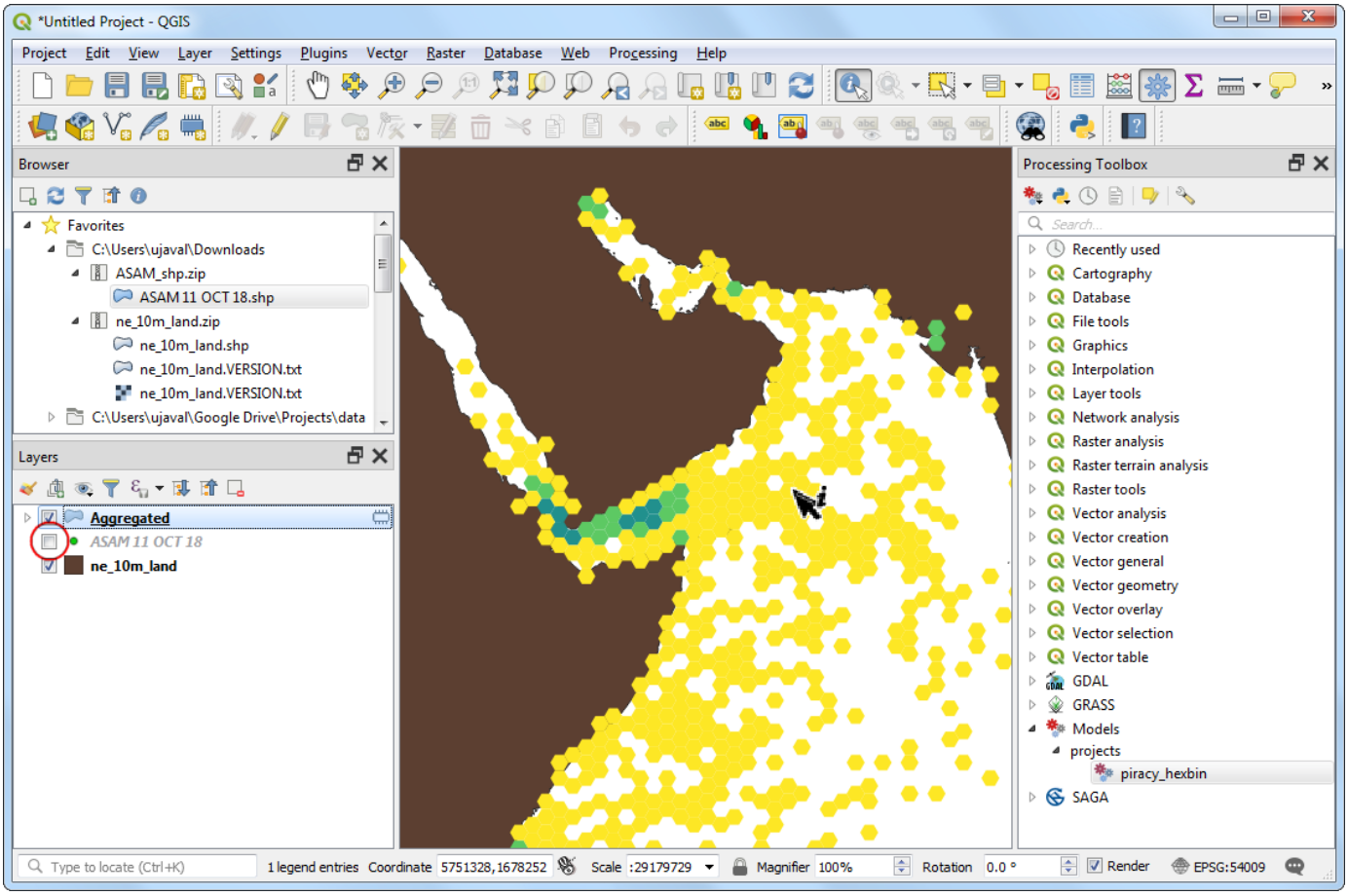
28. The Graduated symbology will divide the values in the selected column into distinct classes and assign a different color to each of the classes. Select `Natural Breaks (Jenks)` as the Mode and click Classify and click OK.

Note

see *Basic Vector Styling* (../basic_vector_styling.html) for a detailed explanation of different modes.



29. Back in the main QGIS window, turn off the `ASAM_events` layer. You will see a nice visualization of piracy hotspots across the globe.



Now that you have encoded the full data pipeline in the model, it is easy to reproduce your results. A model also allows you to experiment quickly without manually repeating each intermediate step every time. If your inputs change over time, say an updated database of piracy is released after a few months, you can run your model on that input to generate a similar visualization without having to remember each step.

Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Automating Map Creation with Print Layout Atlas (QGIS3)

If your organization publishes printed or online maps, you often would need to create many maps with the same template - usually one for each administrative unit or a region of interest. Creating these maps manually can take a long time and if you want to update these on a regular basis, it can turn into a chore. QGIS has a tool called `Atlas` that can help you create a map template and easily publish a large number of maps for different geographic regions. If you are not familiar with the basics of Print Layout, please go through the *Making a Map* (../making_a_map.html) tutorial.

Overview of the task

This tutorial shows how to create wetlands map for each county in the state of Hawaii.

Other skills you will learn

- Using `Inverted Polygons` style renderer to fill areas outside of polygons.
- Write an expression in the `Rule Based` style renderer to show only the current feature in Atlas.
- Write an expression to create dynamic labels in Print Layout.
- Using `Shapeburst fill` style renderer to create a dual-tone polygon fill.

Get the data

We will use the GIS Data Layers (http://planning.hawaii.gov/gis/download-gis-data/) from State of Hawaii - Office of Planning (http://planning.hawaii.gov/)

Download the Wetlands (http://files.hawaii.gov/dbedt/op/gis/data/HI_Wetlands.shp.zip) layer from Biologic and Ecologic category.

Download the Census County Boundaries 2010 (http://files.hawaii.gov/dbedt/op/gis/data/county10.shp.zip) layer from the Cultural and Demographic category.

For convenience, you may directly download a copy of both the datasets from the links below:

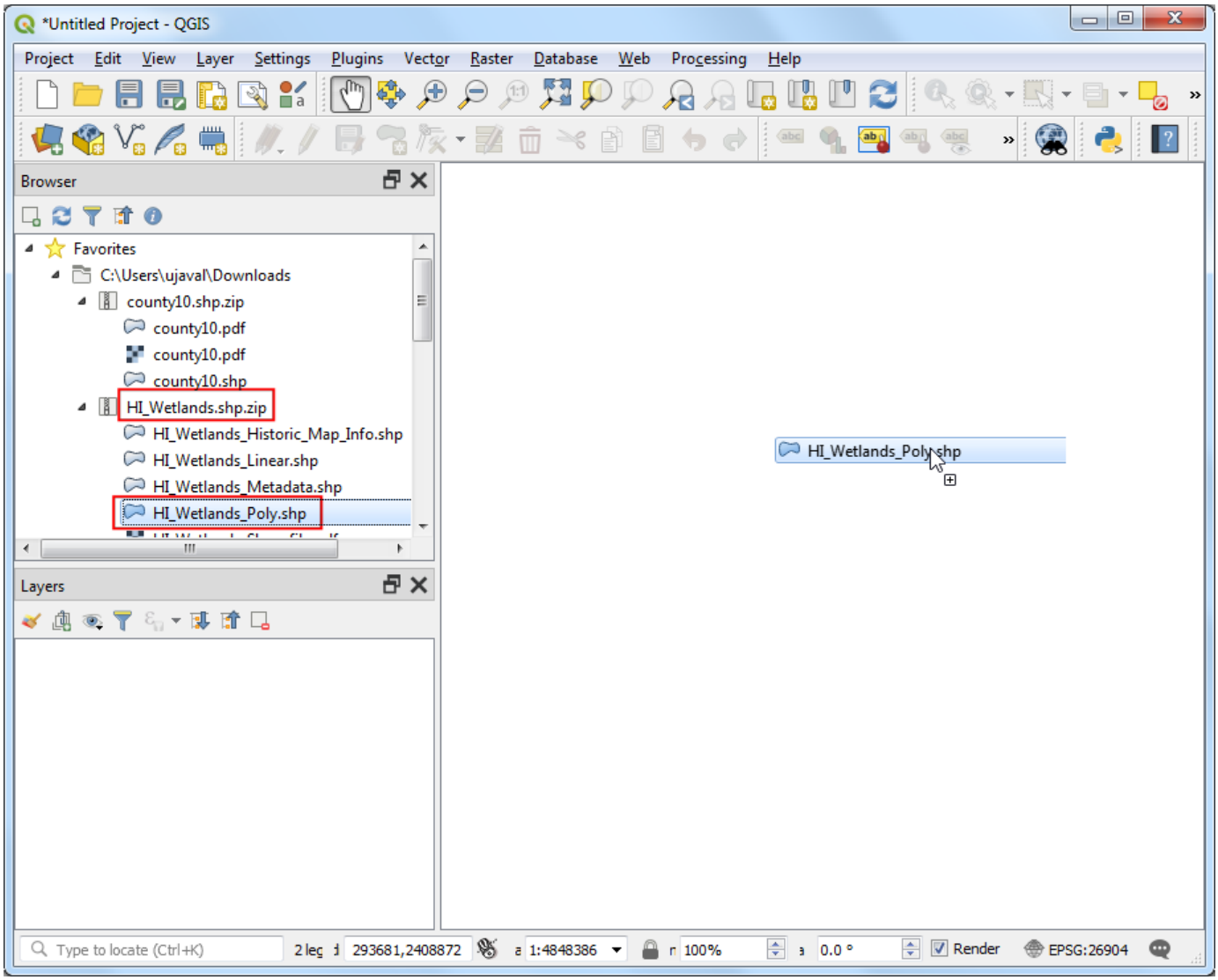
HI_Wetlands.shp.zip (http://www.qgistutorials.com/downloads/HI_Wetlands.shp.zip)

county10.shp.zip (http://www.qgistutorials.com/downloads/county10.shp.zip)

Data Source [HAWAII] (../credits.html#hawaii)

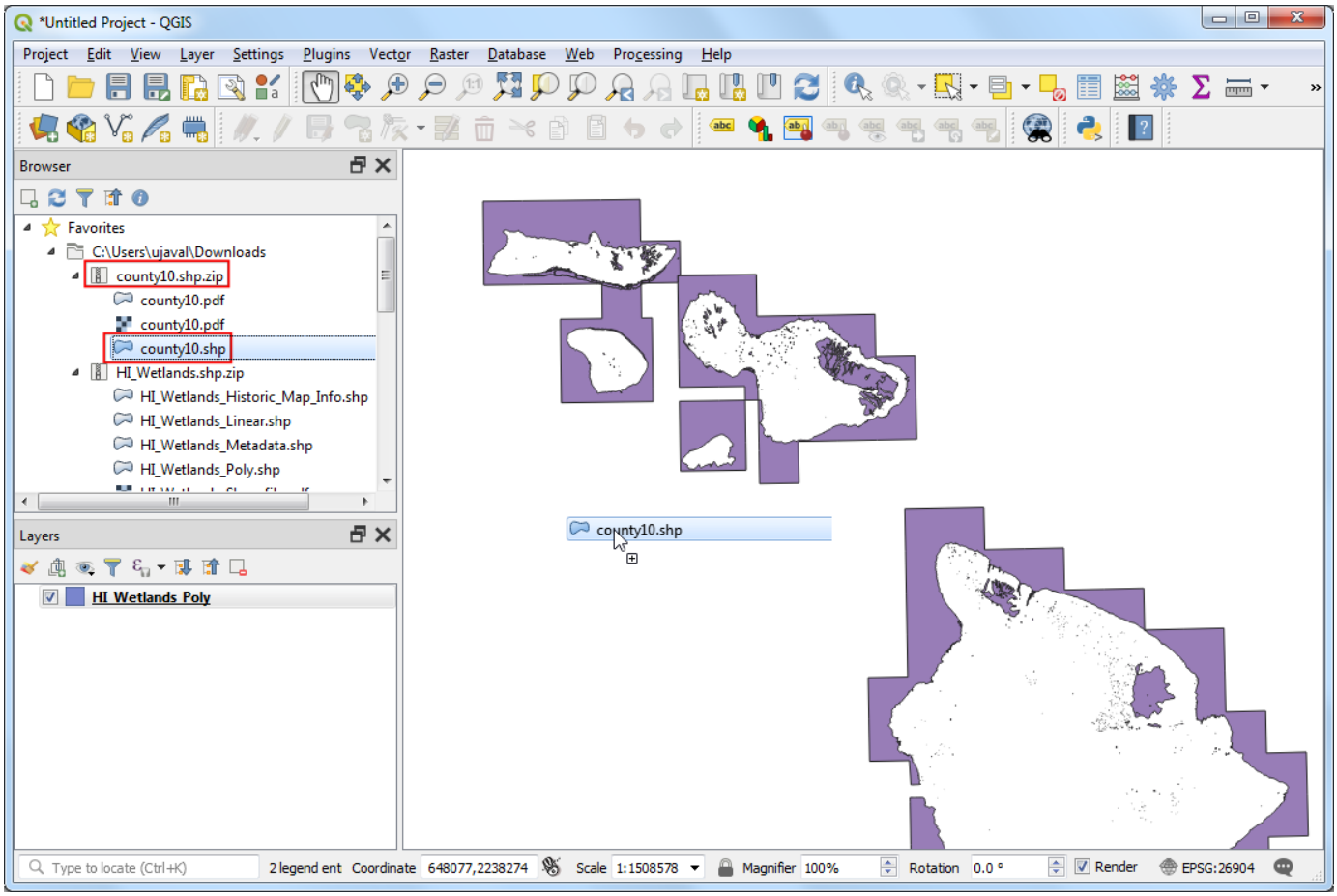
Procedure

1. Locate the `HI_Wetlands.shp.zip` file in the QGIS Browser and expand it. Select the `HI_Wetlands_Poly.shp` file and drag it to the canvas. This layer contains polygons representing wetlands in the entire state of Hawaii.

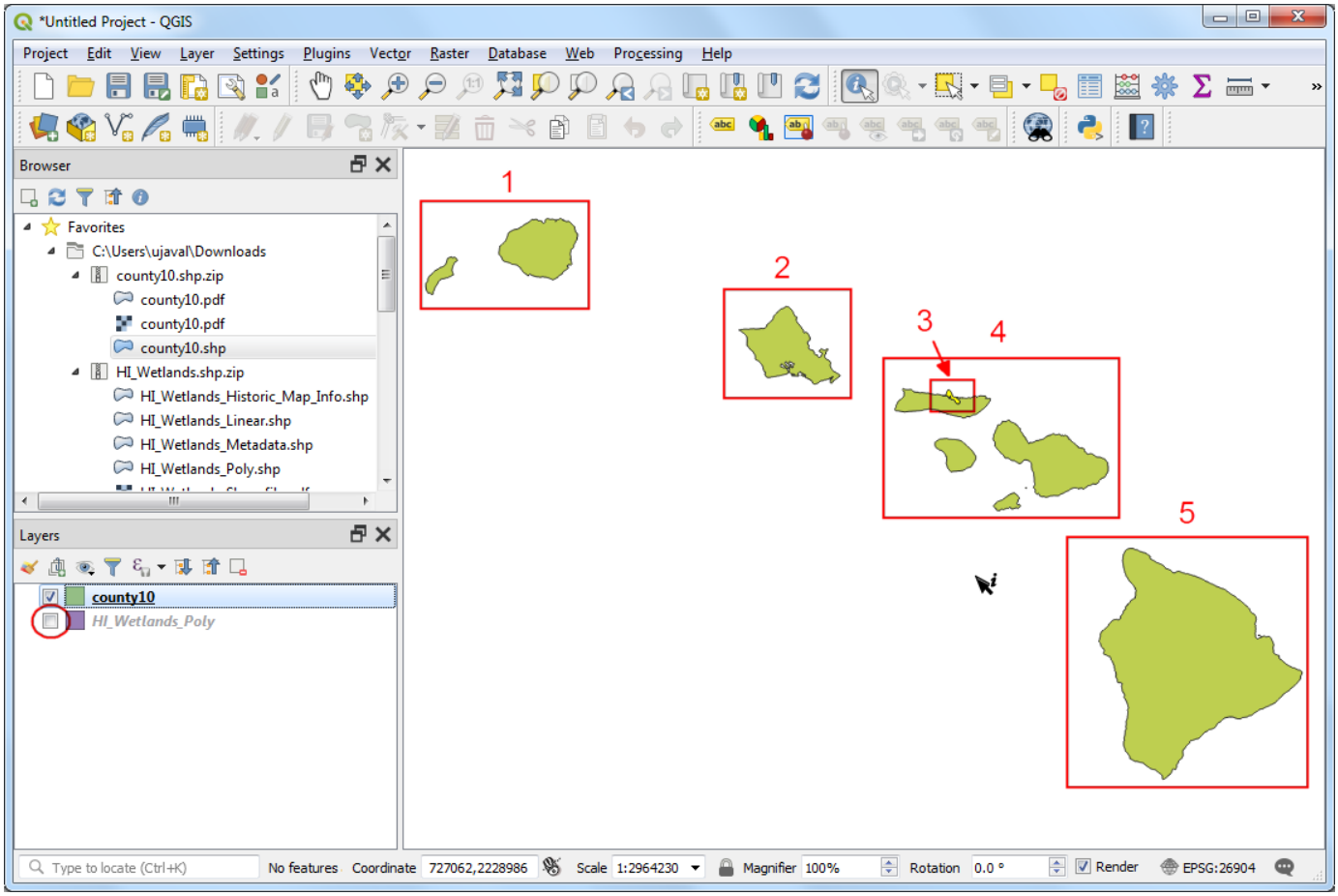


2. Since we want to make separate wetlands map for each county in the state, we will need the county boundaries layer. Browse to the `county10.shp.zip` file and expand it. Select the `county10.shp` file

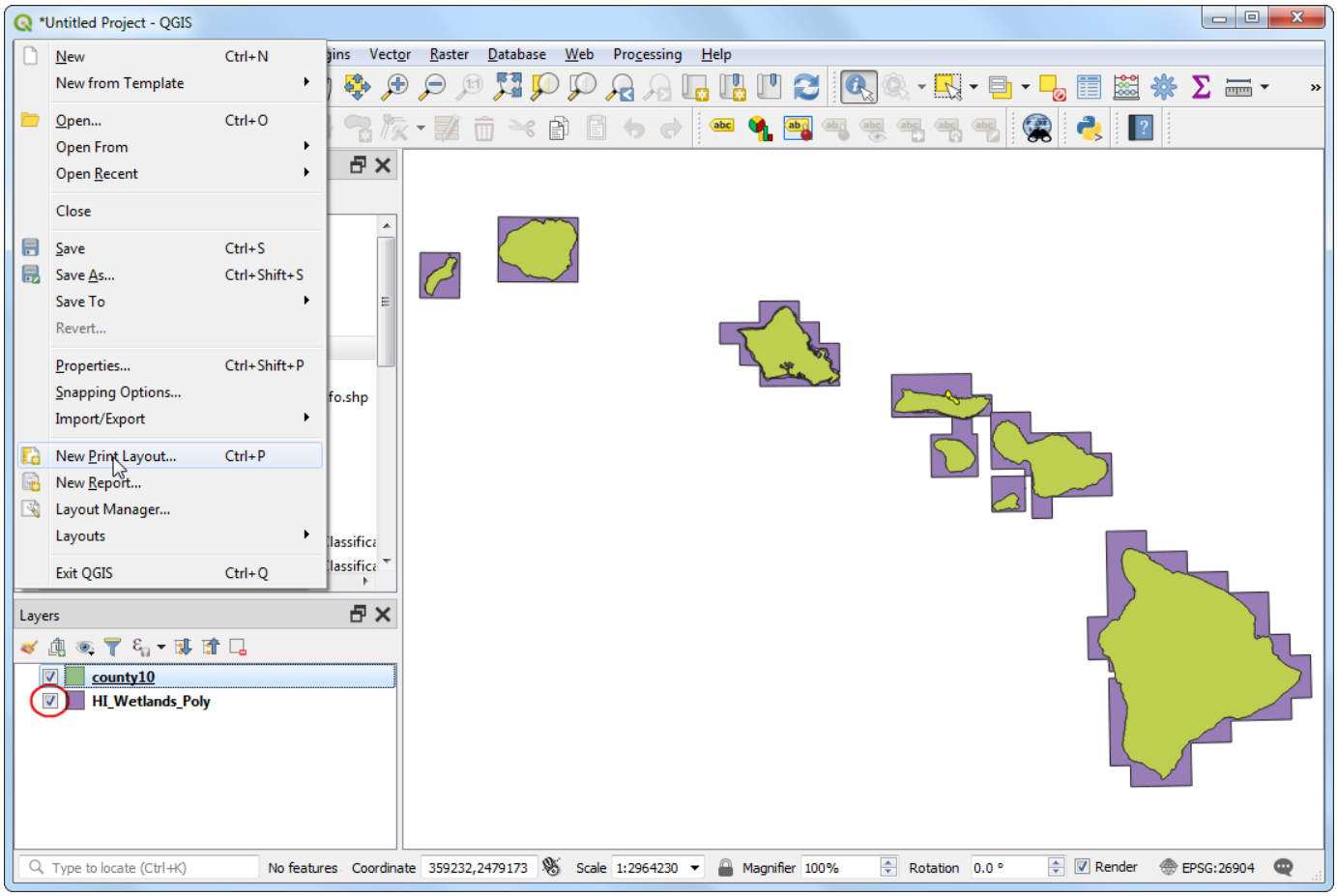
and drag it to the canvas.



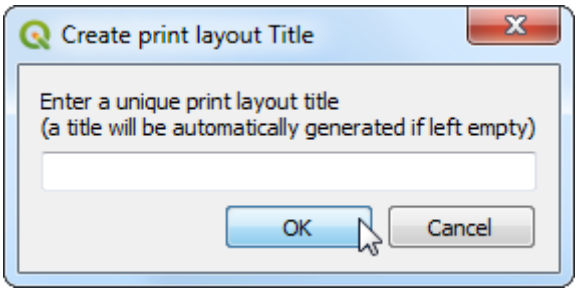
3. Turn off the visibility of the `HI_Wetlands_Poly` layer temporarily. You will see the polygons from the `county10` layer clearly now. There are 5 features contained in this layer, with each feature having 1 or more polygons associated with it. The features represent 5 counties. We will use this layer as the coverage layer and configure QGIS to create 5 separate maps - one for each feature - automatically.



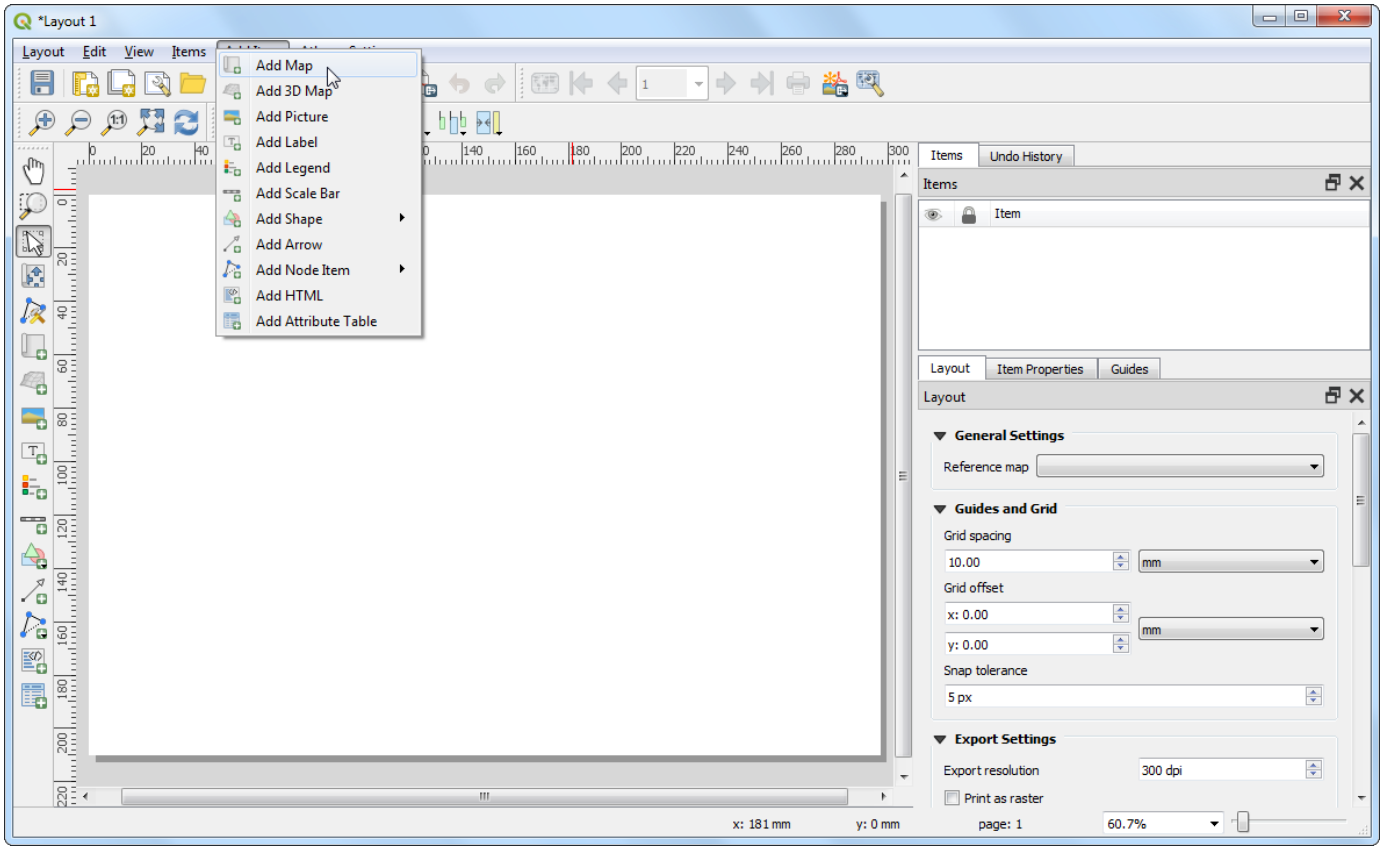
4. Turn on the visibility of the `HI_Wetlands_Poly` layer. Go to Project ▸ New Print Layout...



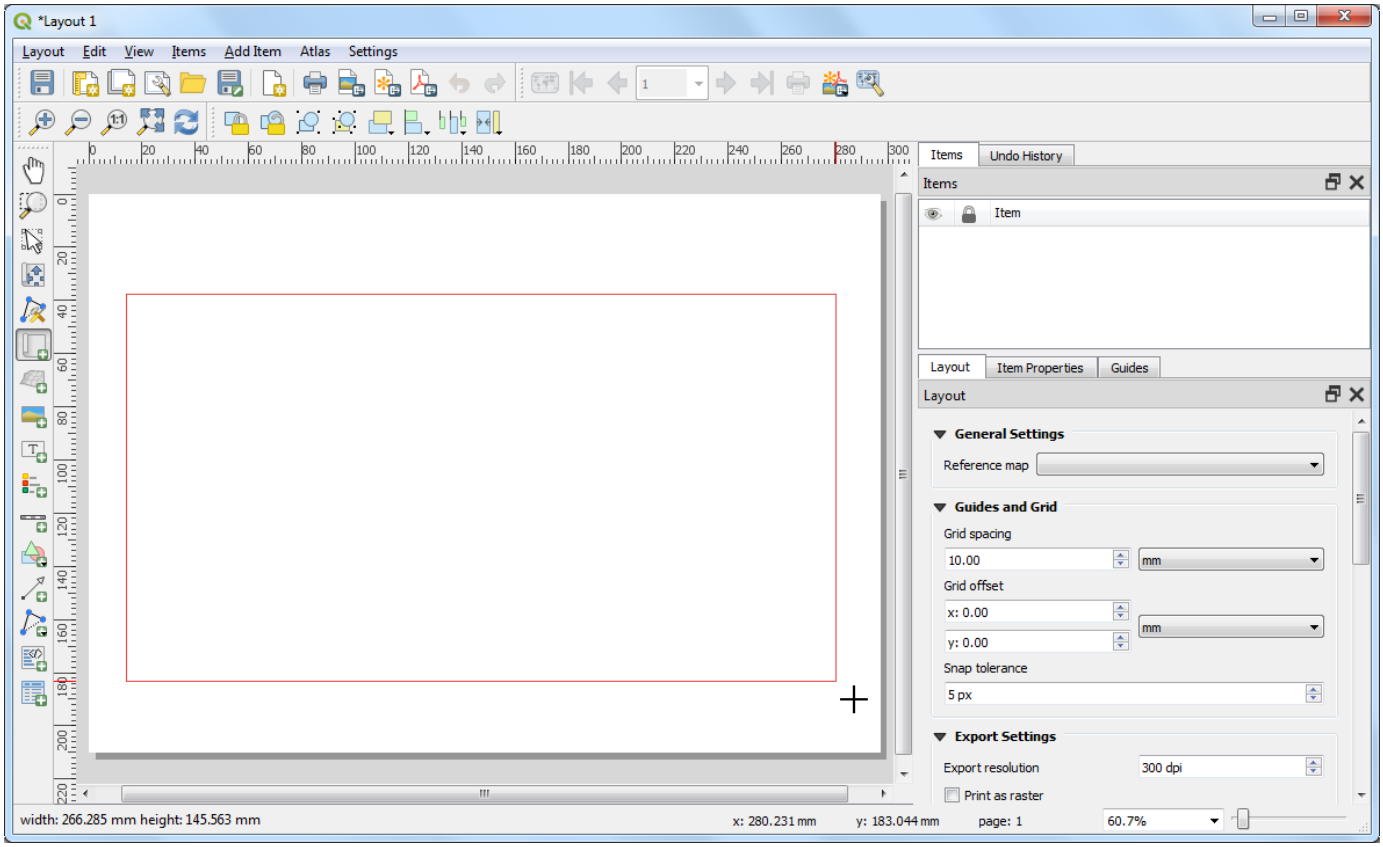
5. Leave the print layout title empty and click OK.



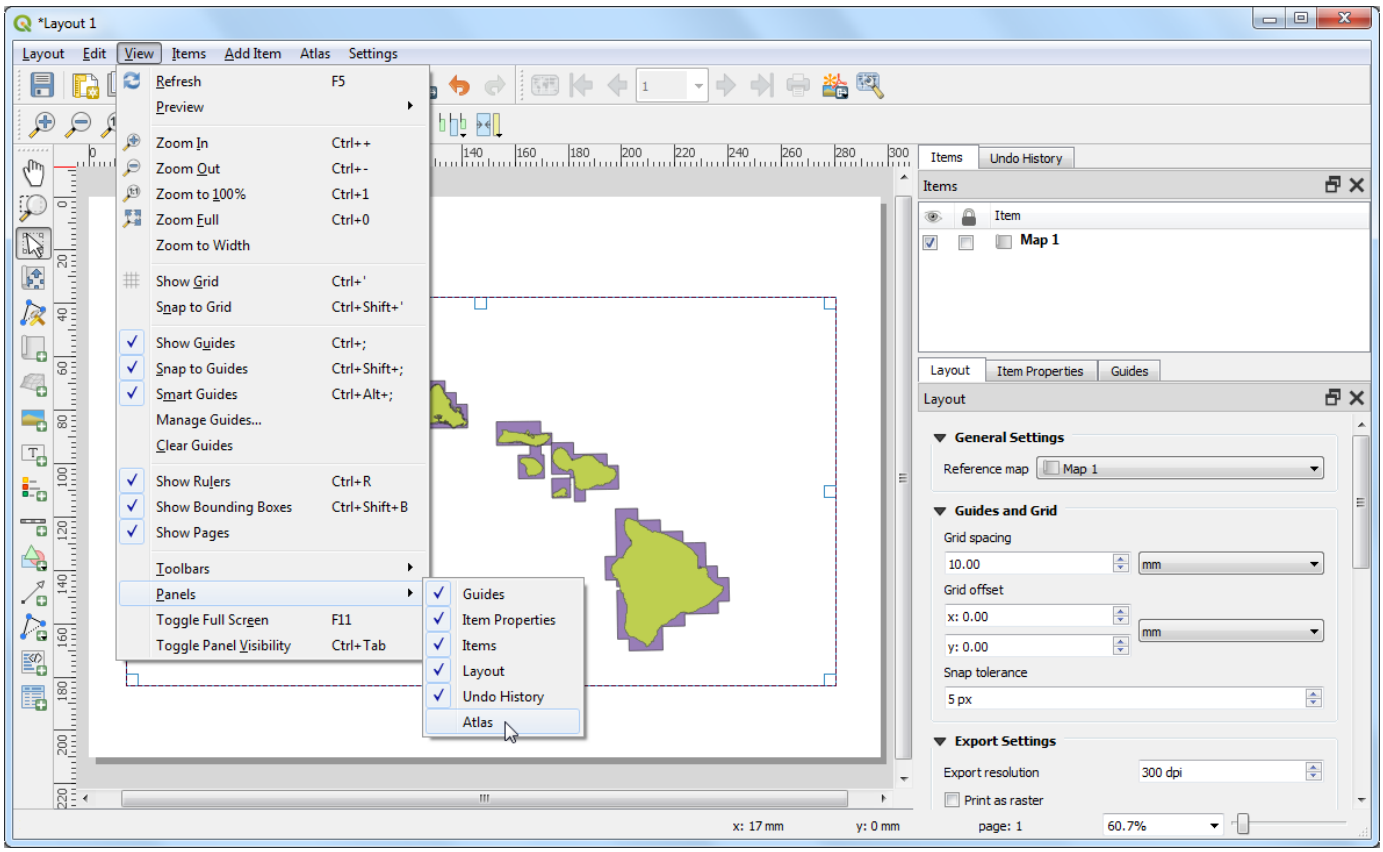
6. In the Print Layout window, go to Layout ▸ Add Map.



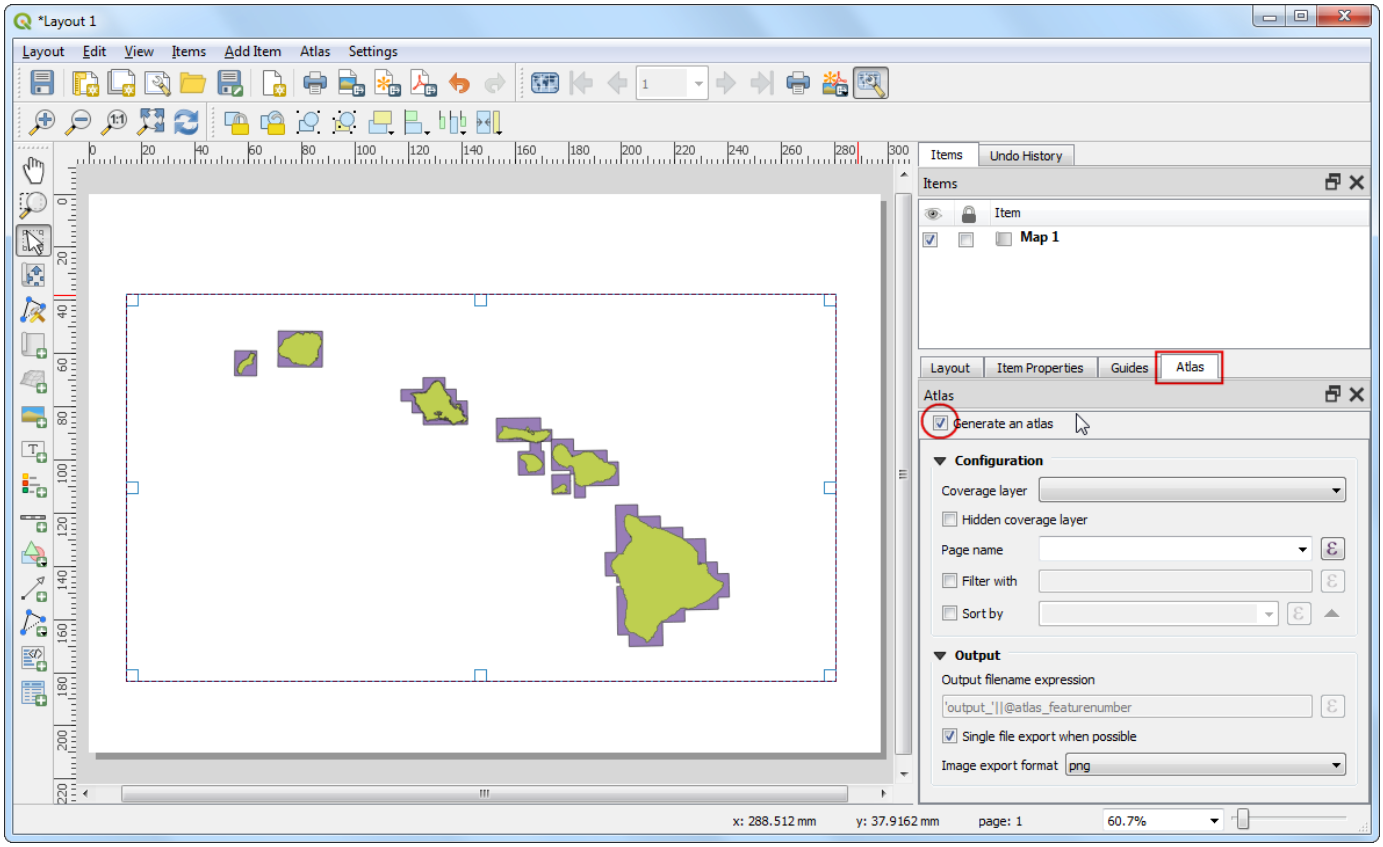
7. Drag a rectangle while holding the left mouse button where you would like to insert the map.



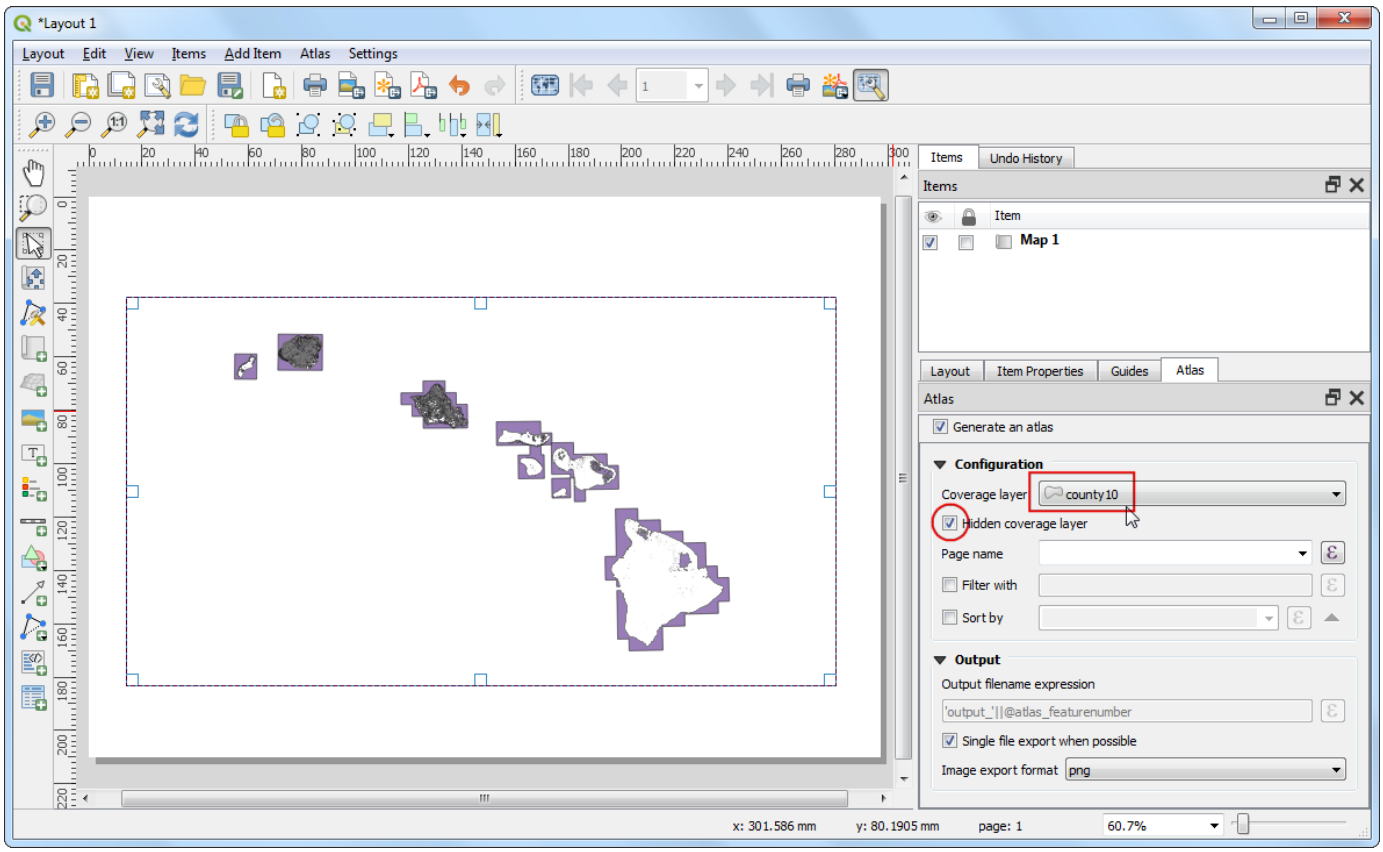
8. In QGIS3, the Atlas tab is not visible by default. Select View ▸ Panels ▸ Atlas.



9. Switch to the Atlas tab. Check the Generate an atlas box.



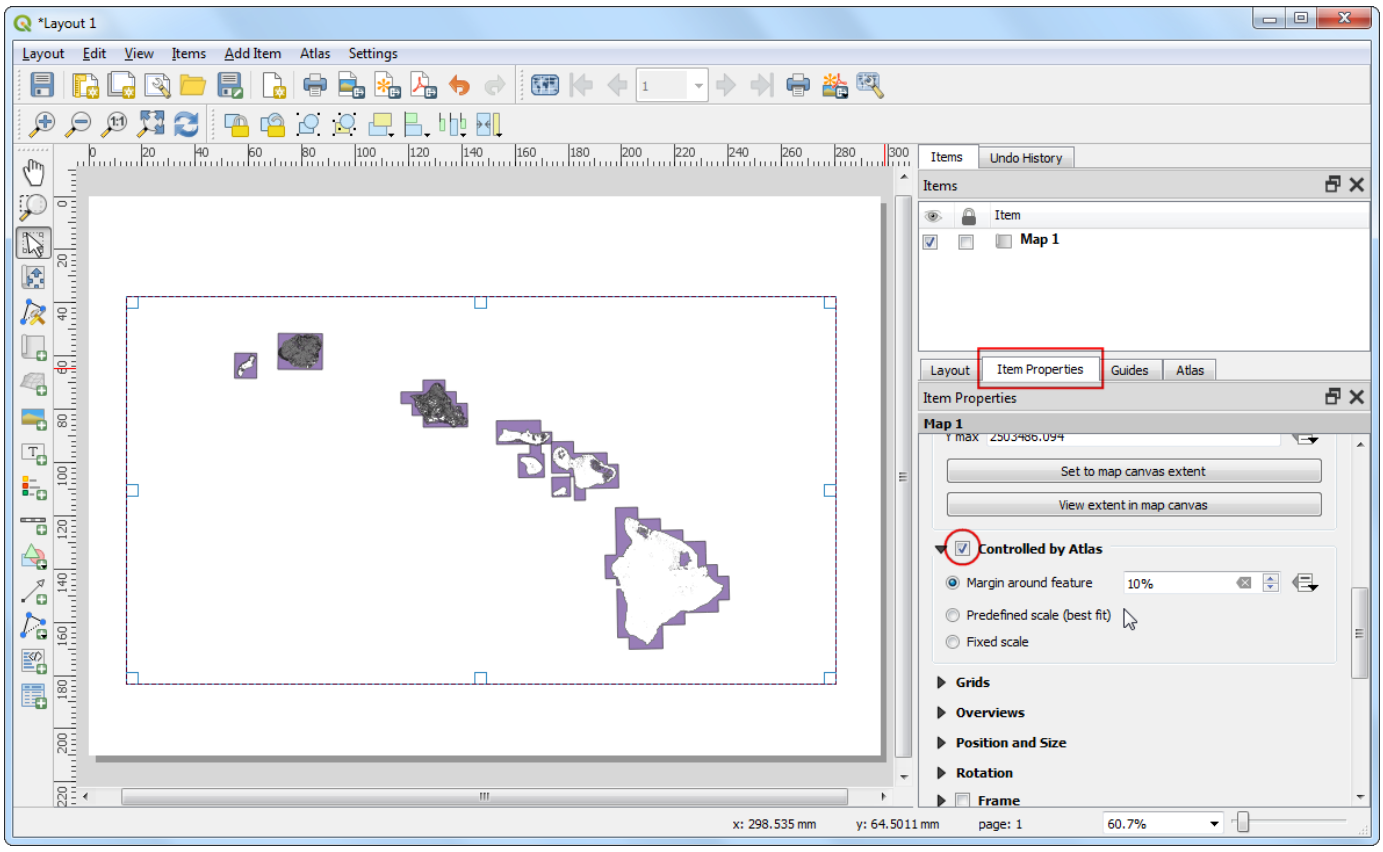
10. Select the `county10` as the Coverage layer. This will indicate that we want to create 1 map each for every polygon feature in the `county10` layer. You can also check the Hidden coverage layer so that the features themselves will not appear on the map.



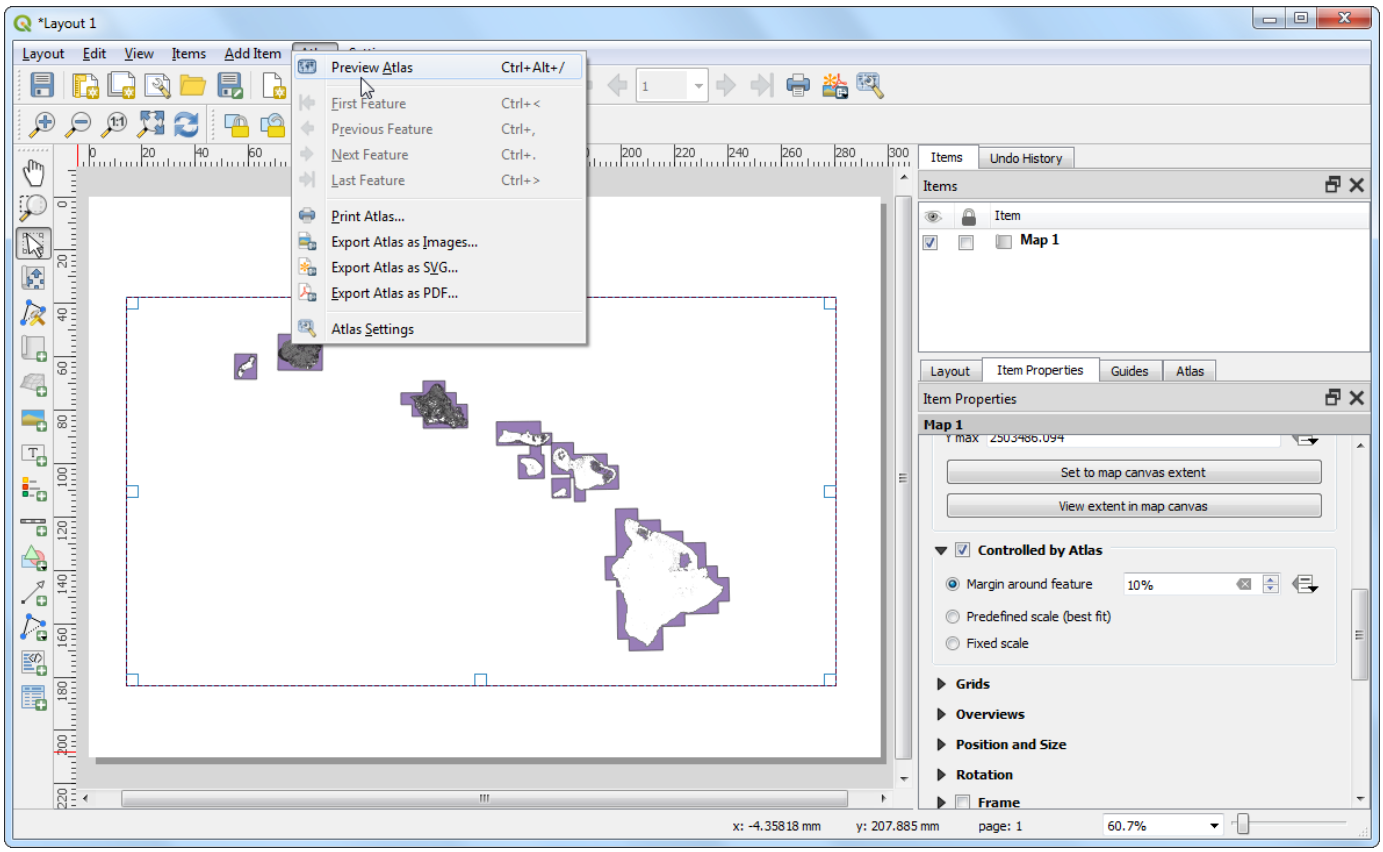
11. Switch to the Item Properties tab. Scroll down and check the Controlled by atlas box. This will indicate the layout that the content of the map displayed in this item will be determined by the `Atlas` tool.

Note

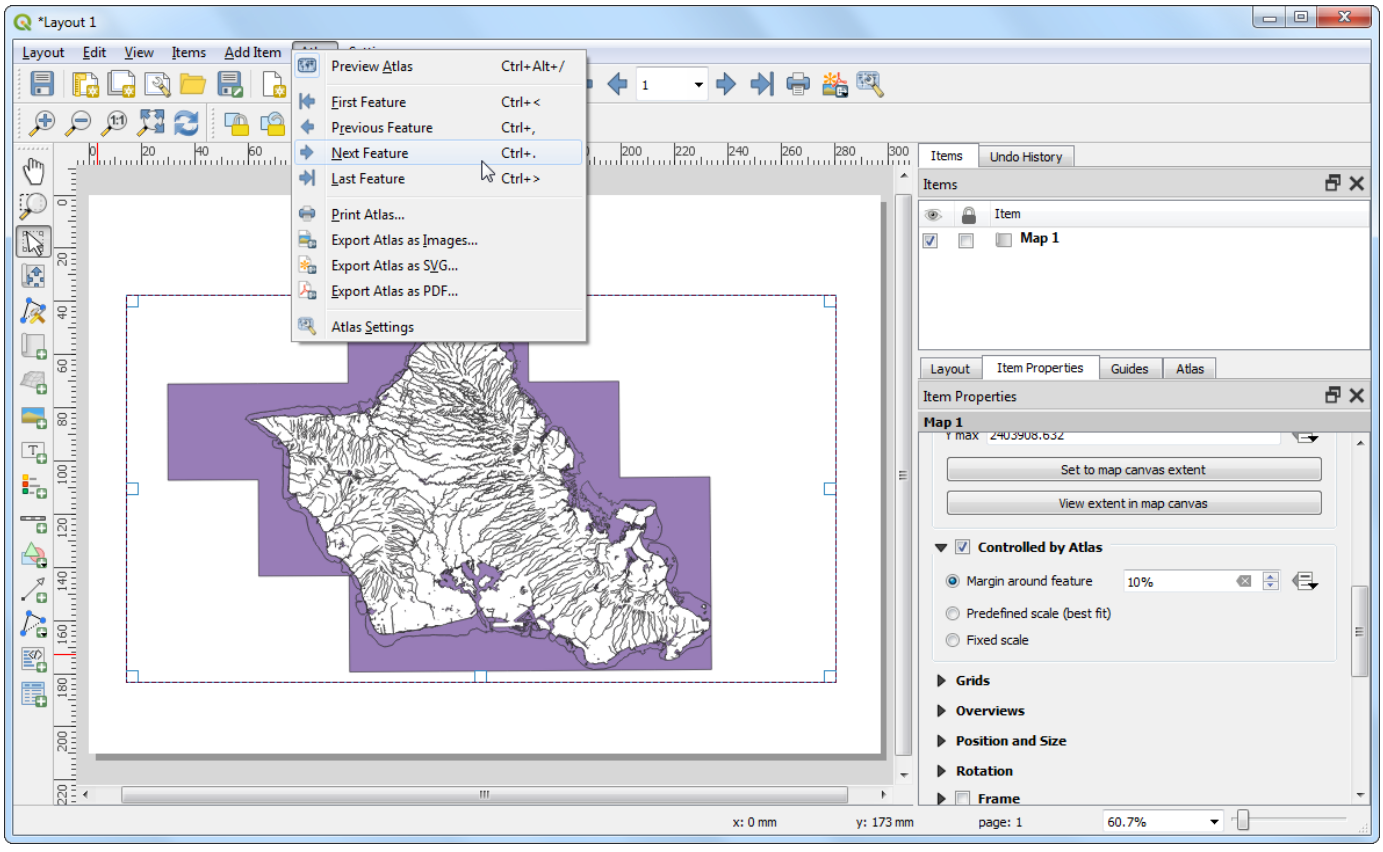
You must enable the Generare an atlas box in the Atlas tab, otherwise the Controlled by atlas checkbox will be diasbled.



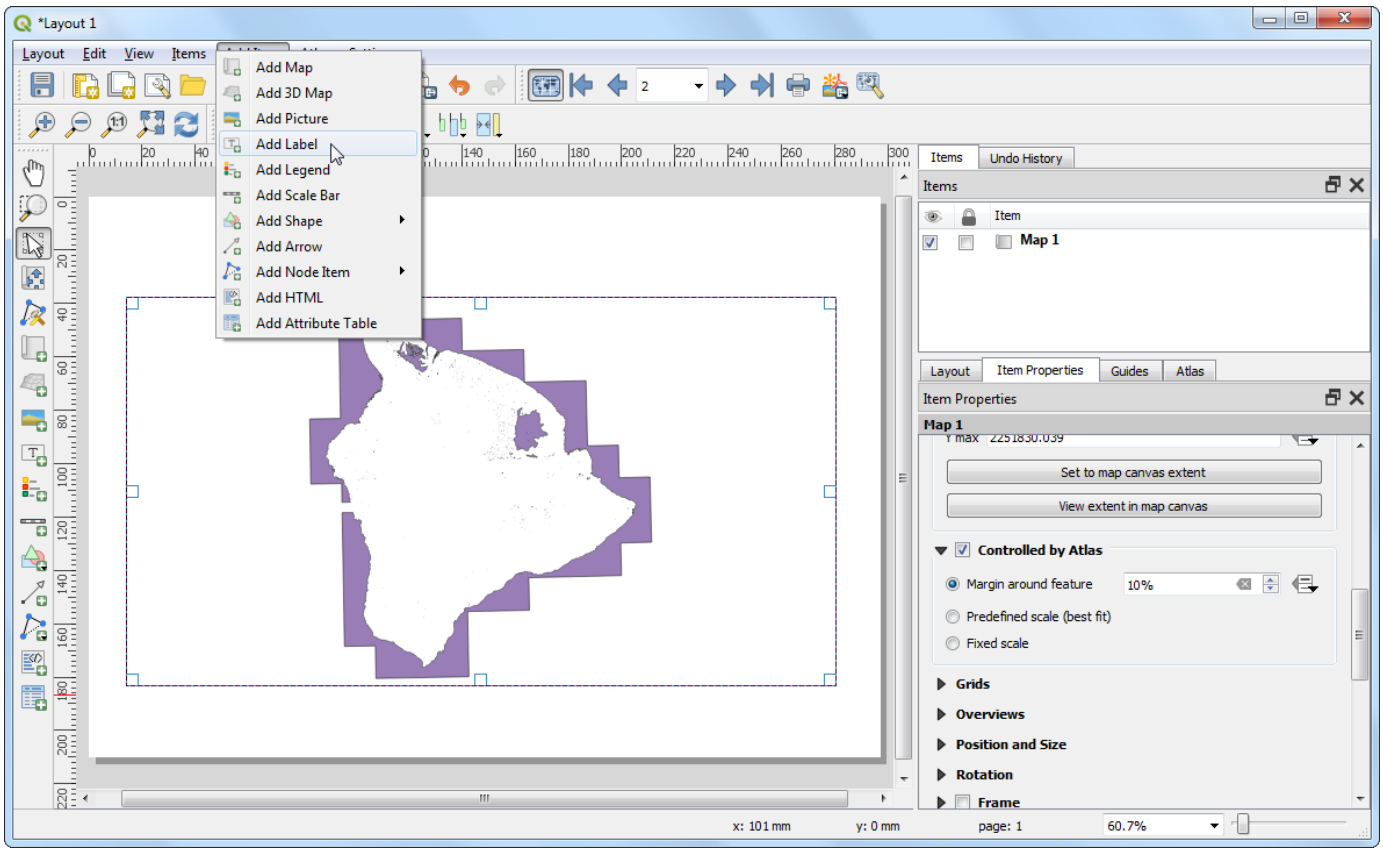
12. Now that you have configuring the Atlas settings, go to Atlas ‣ Preview Atlas.



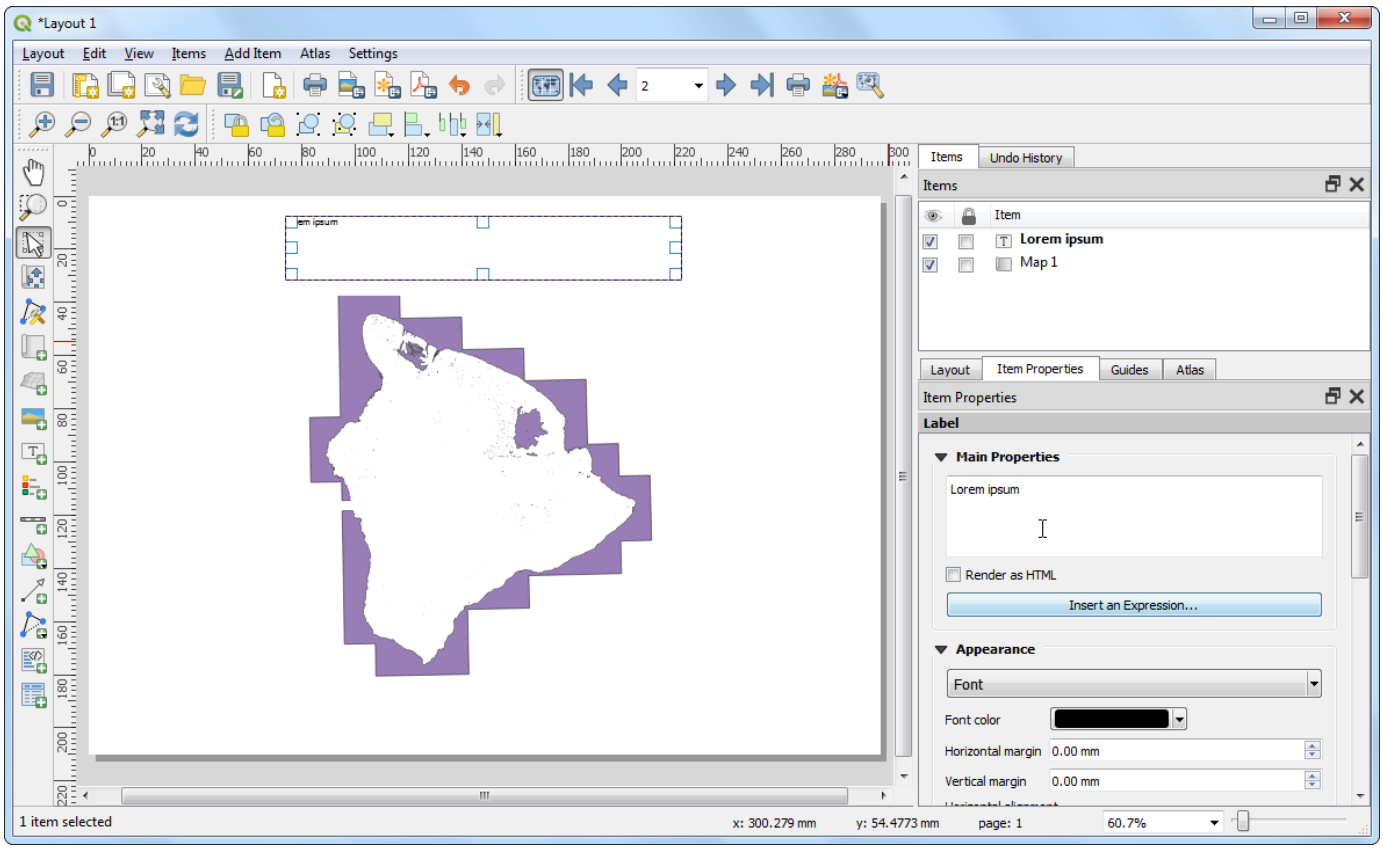
13. You will see the map refresh and show how individual map will look like. You can preview how the map will look for each of the county polygons. Go to Atlas ‣ Next Feature. Atlas will render the map to the extent of the next feature in the coverage layer.



14. Let’s add a label to the map. Go to Layout ‣ Add Label.

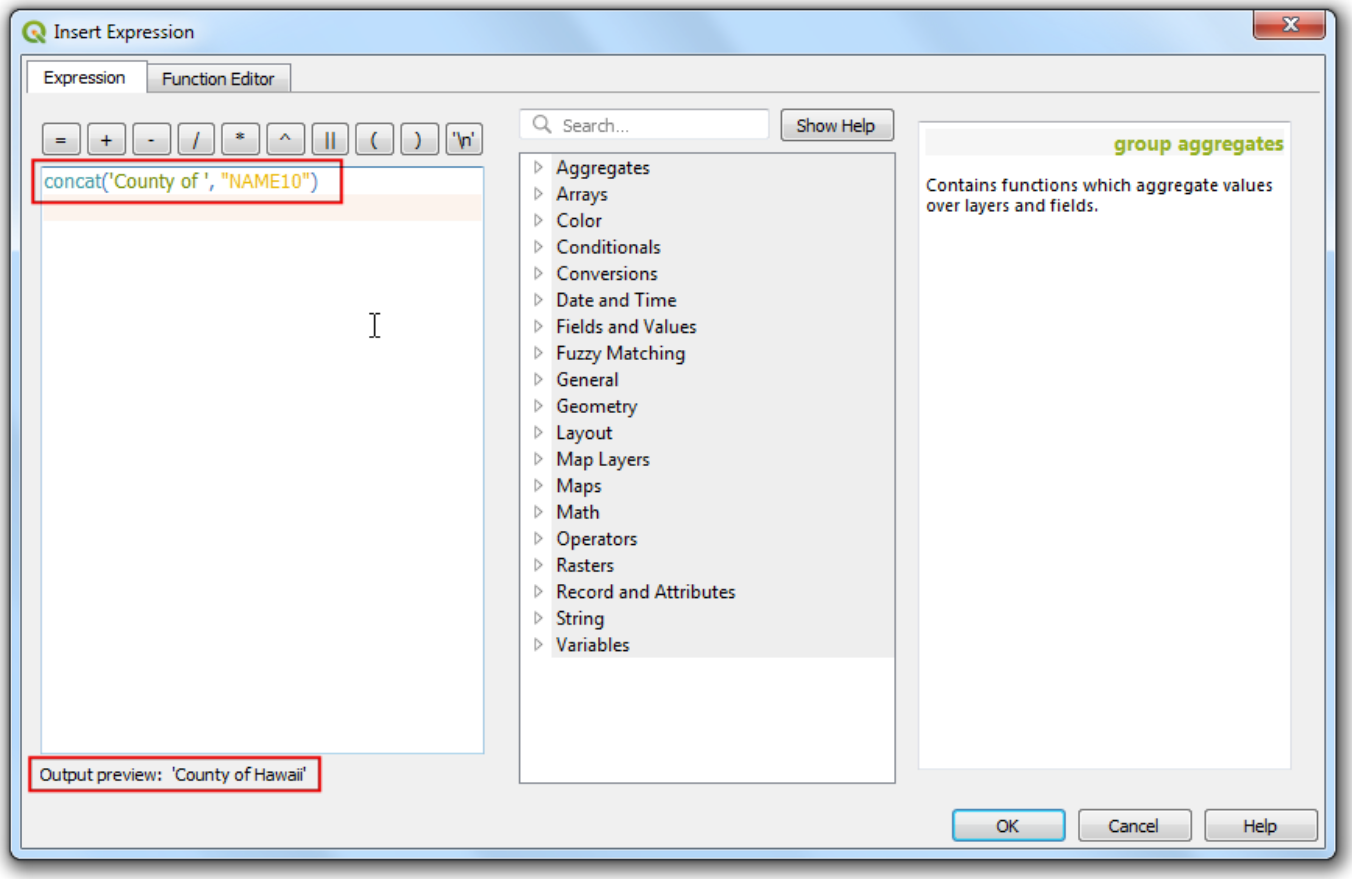


15. Under the Item properties tab, locate the Main properties section and click Insert an Expression... button.

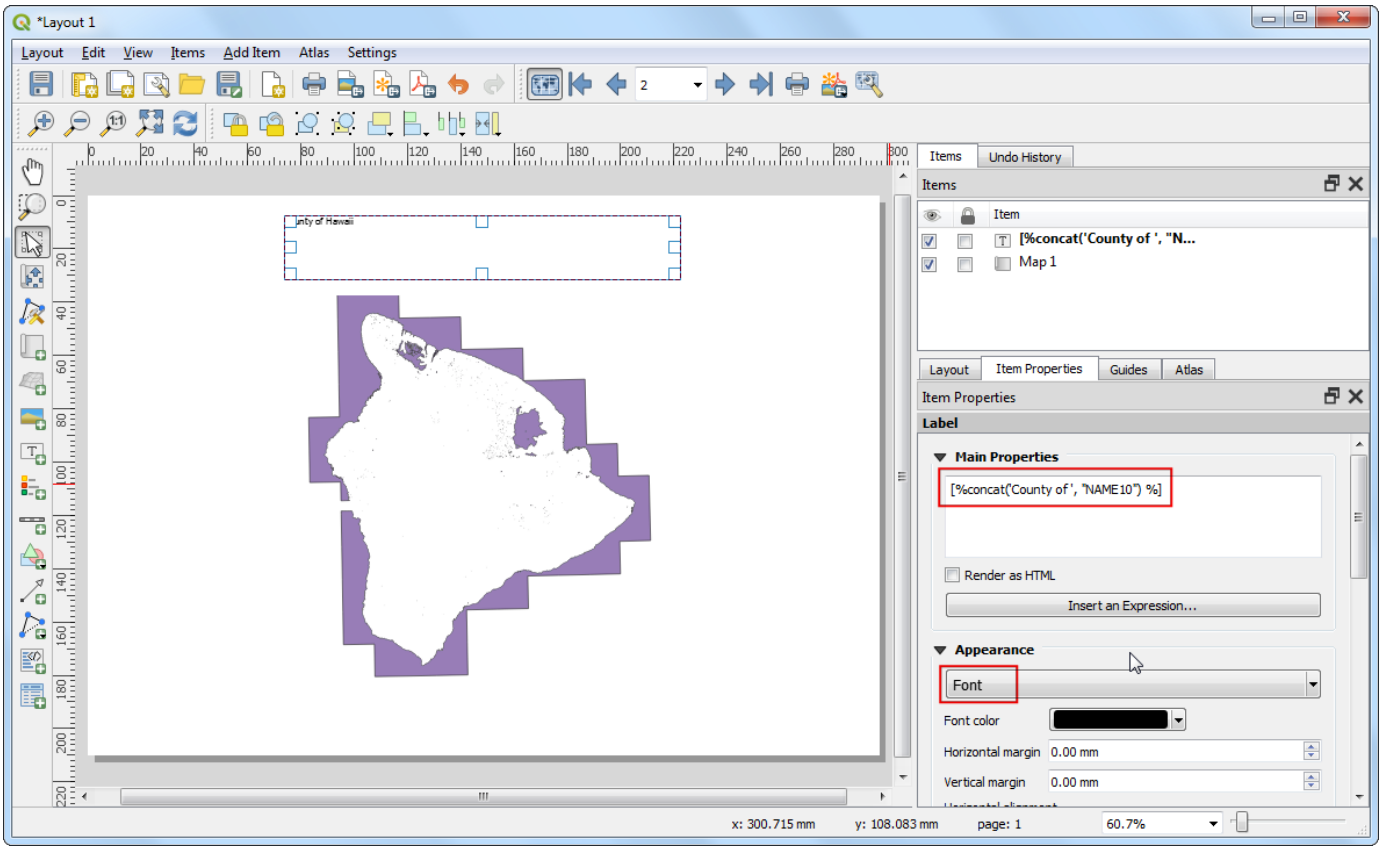


16. The label of the map can use the attributes from the coverage layer. The `concat` function is used to join multiple text items into a single text item. In this case we will join the value of the `NAME10` attribute of the `county10` layer with the text `County of`. Add an expression like below and click OK.

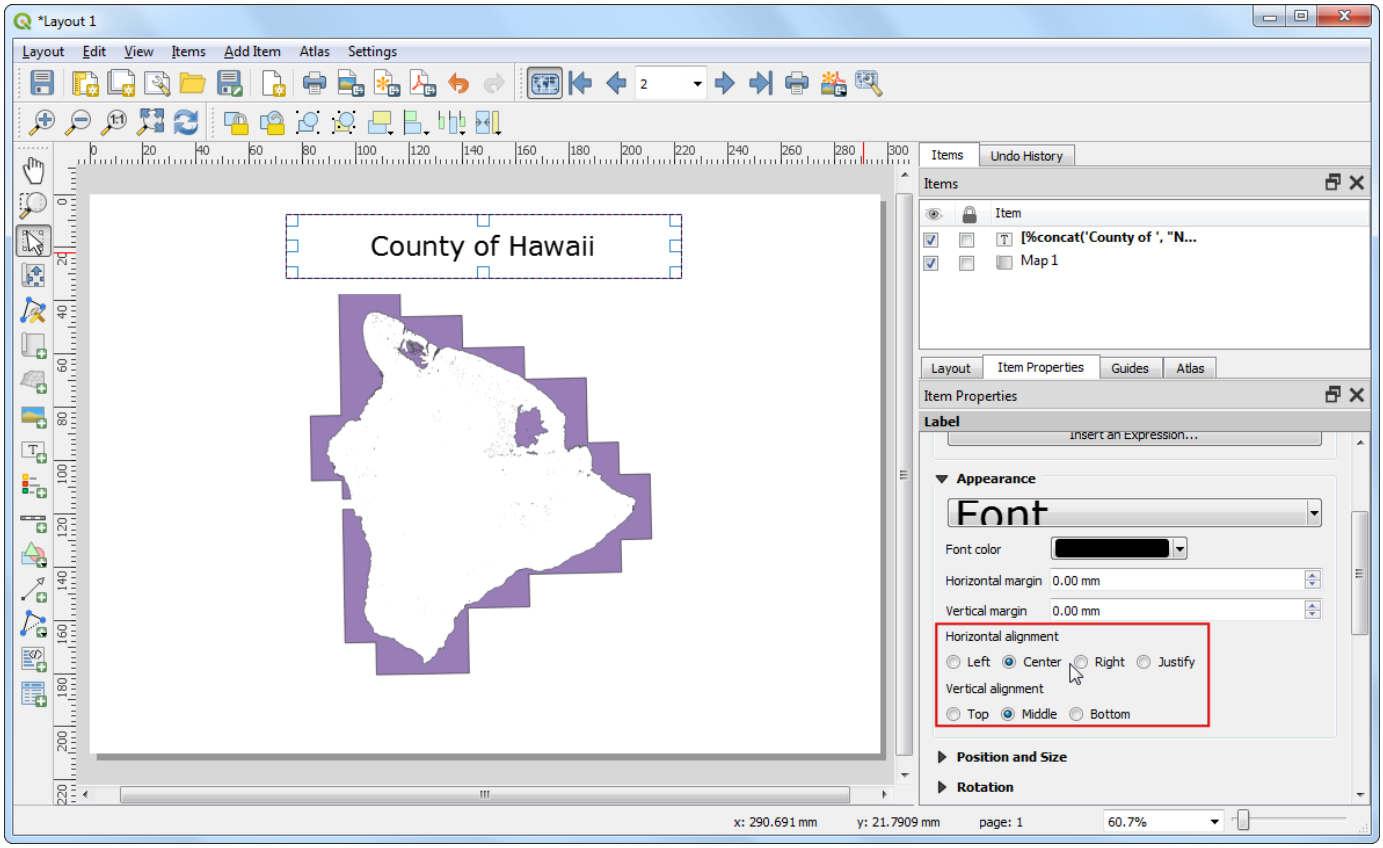
```
concat('County of ', "NAME10")
```



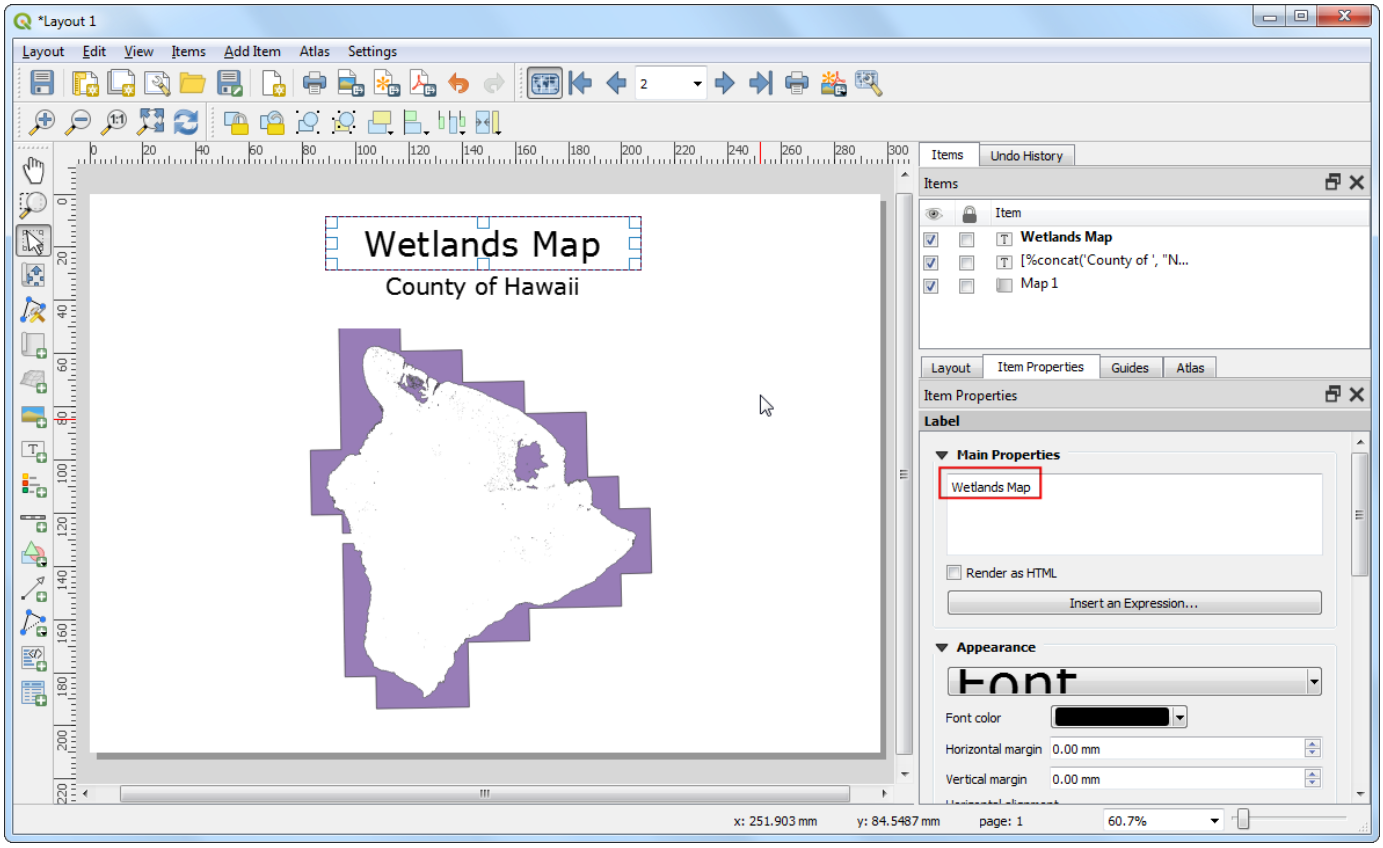
17. Delete the leading *Lorem ipsum* placeholder text so that the textbox contains only the expression. Scroll down to the Appearance section and click on the Font dropdown. Choose the font and adjust the size to your liking.



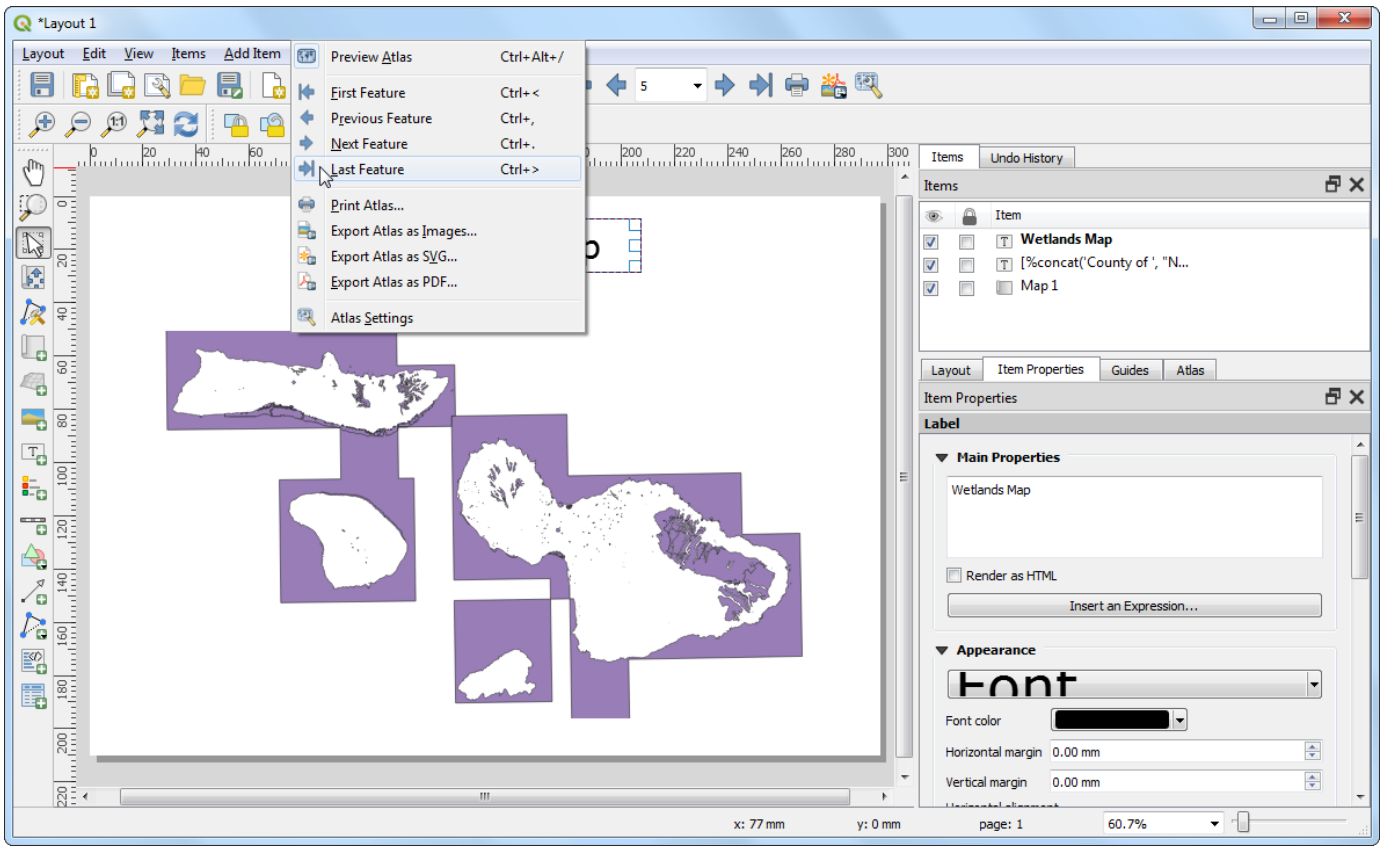
18. Choose `Center` as the Horizontal alignment and `Middle` as the Vertical alignment option.



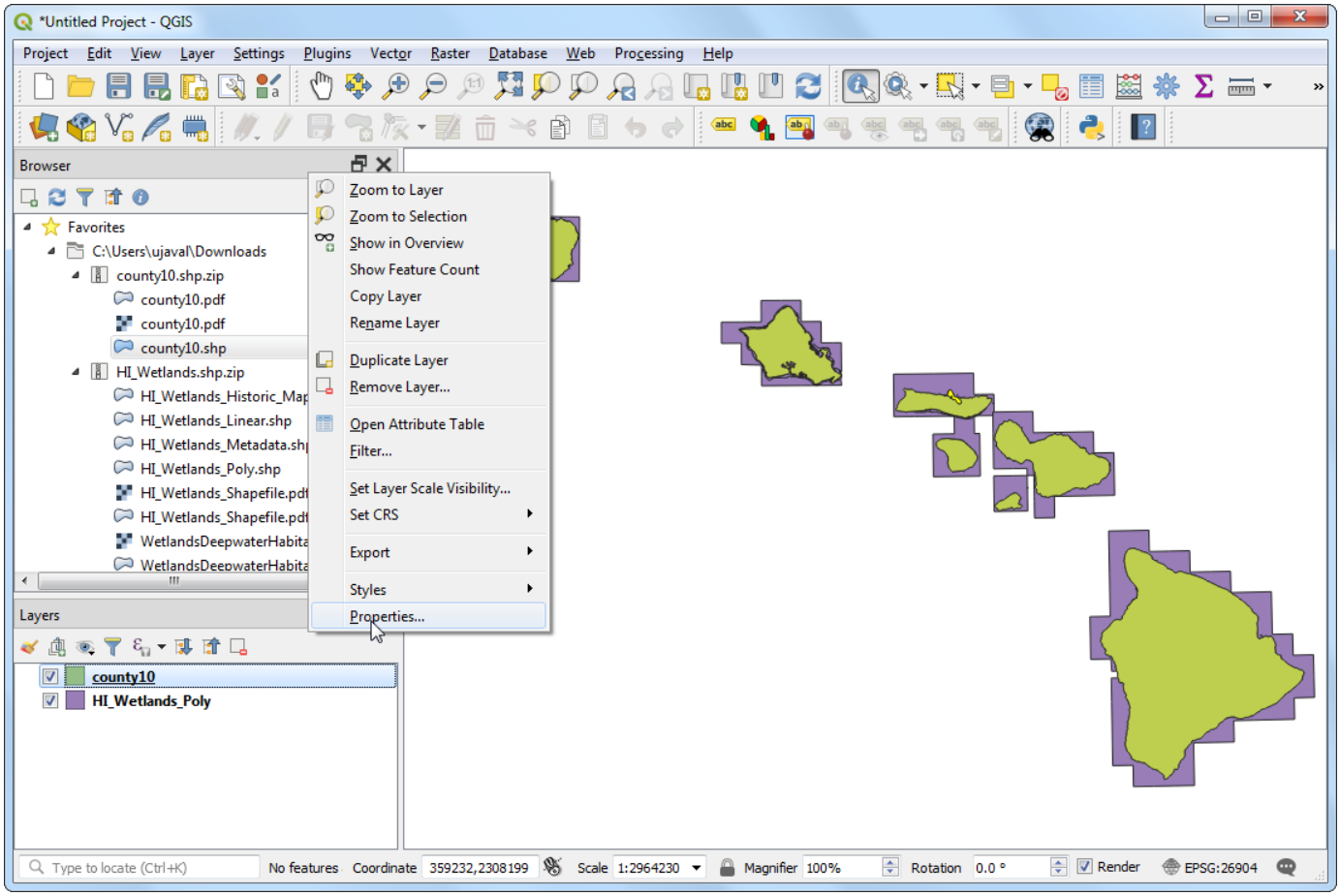
19. Add another label and enter `Wetlands Map` under the Main properties. Since there is no expression here, this text will remain the same on all maps.



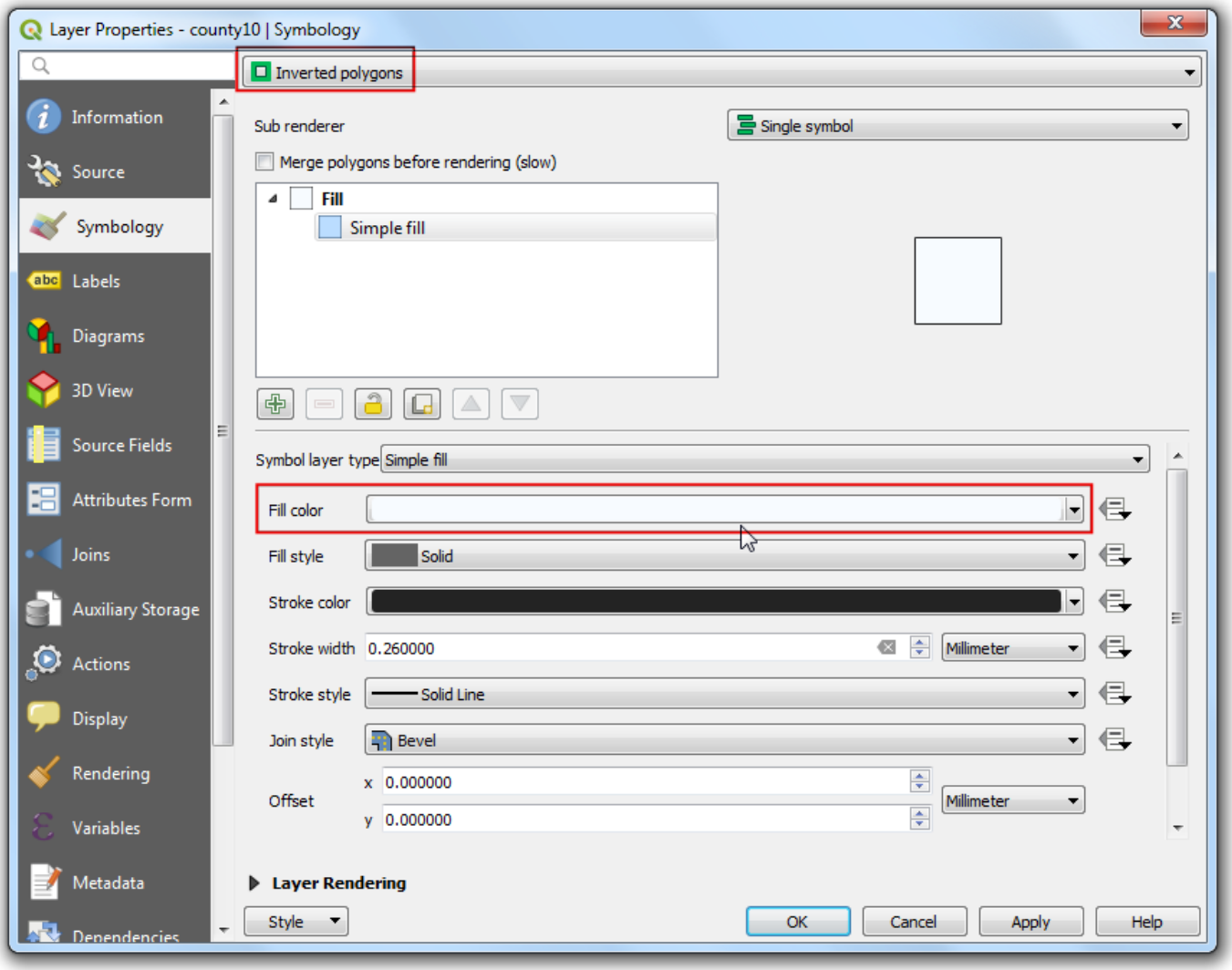
20. Go to Atlas ▸ Last Feature and verify that the map labels do work as intended. You will notice that the wetland map has polygons extending out in the ocean that looks ugly. We can change the style to that areas outside the county boundaries are hidden.



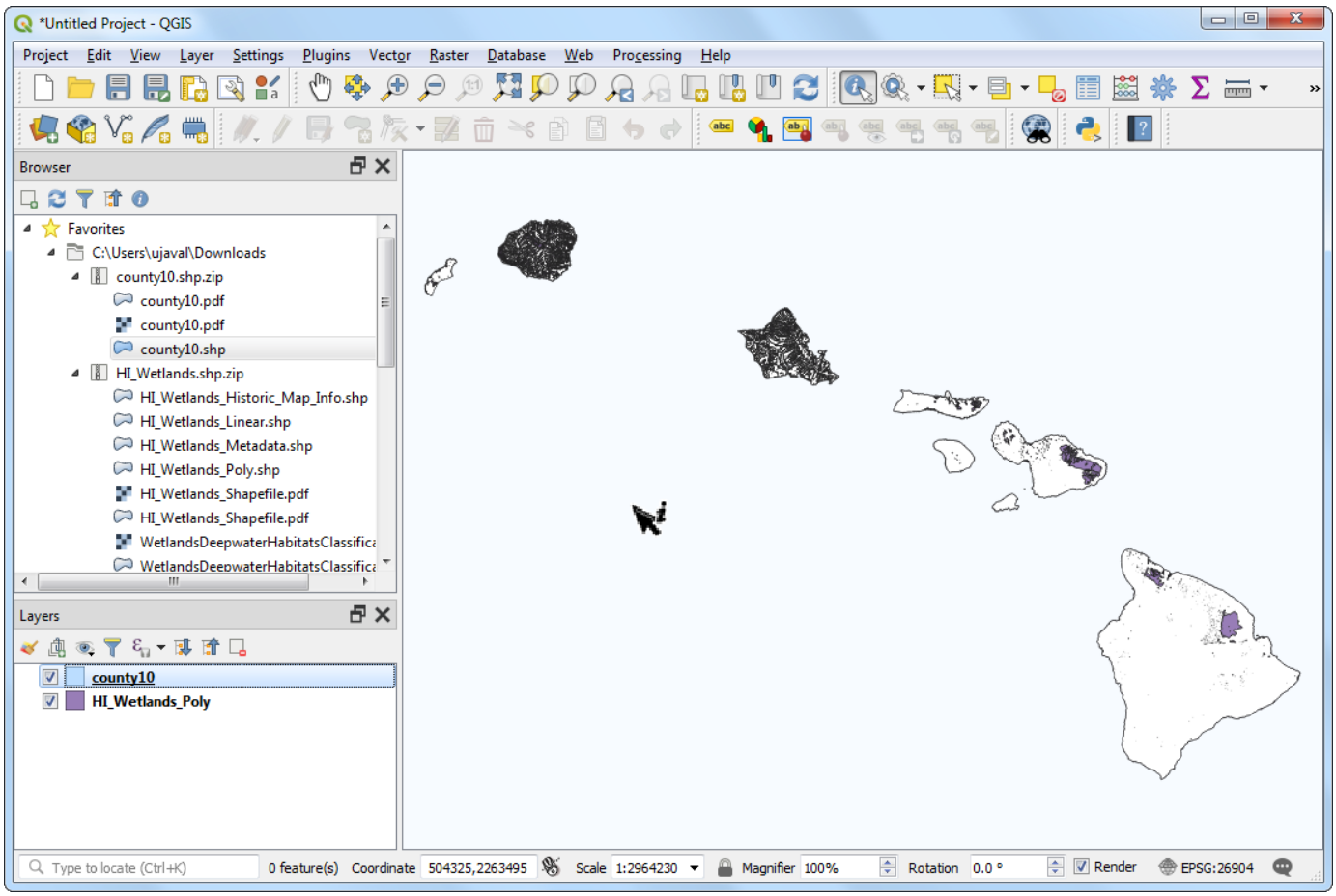
21. Switch to the main QGIS window. Right-click the `county10` layer and select Properties.



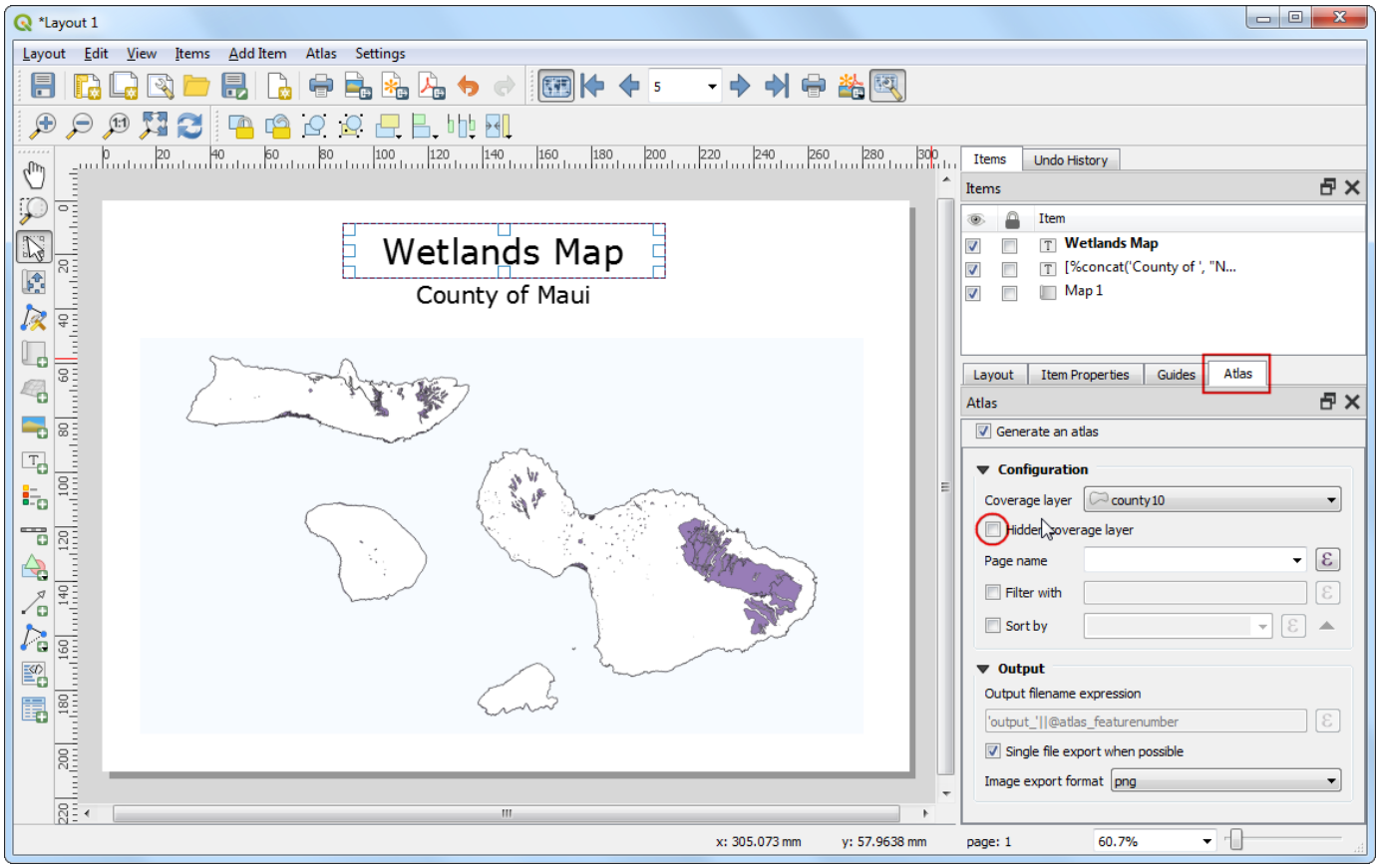
22. In the Symbology tab, select the Inverted polygons renderer. This renderer styles the *outside* of the polygon - not inside. Select white as the fill color and click OK.



23. You will notice that the polygons extending outside of the county boundaries are now disappeared. In reality, they are hidden by the white color fill extending out from the county polygons because of the *Inverted polygons* style.

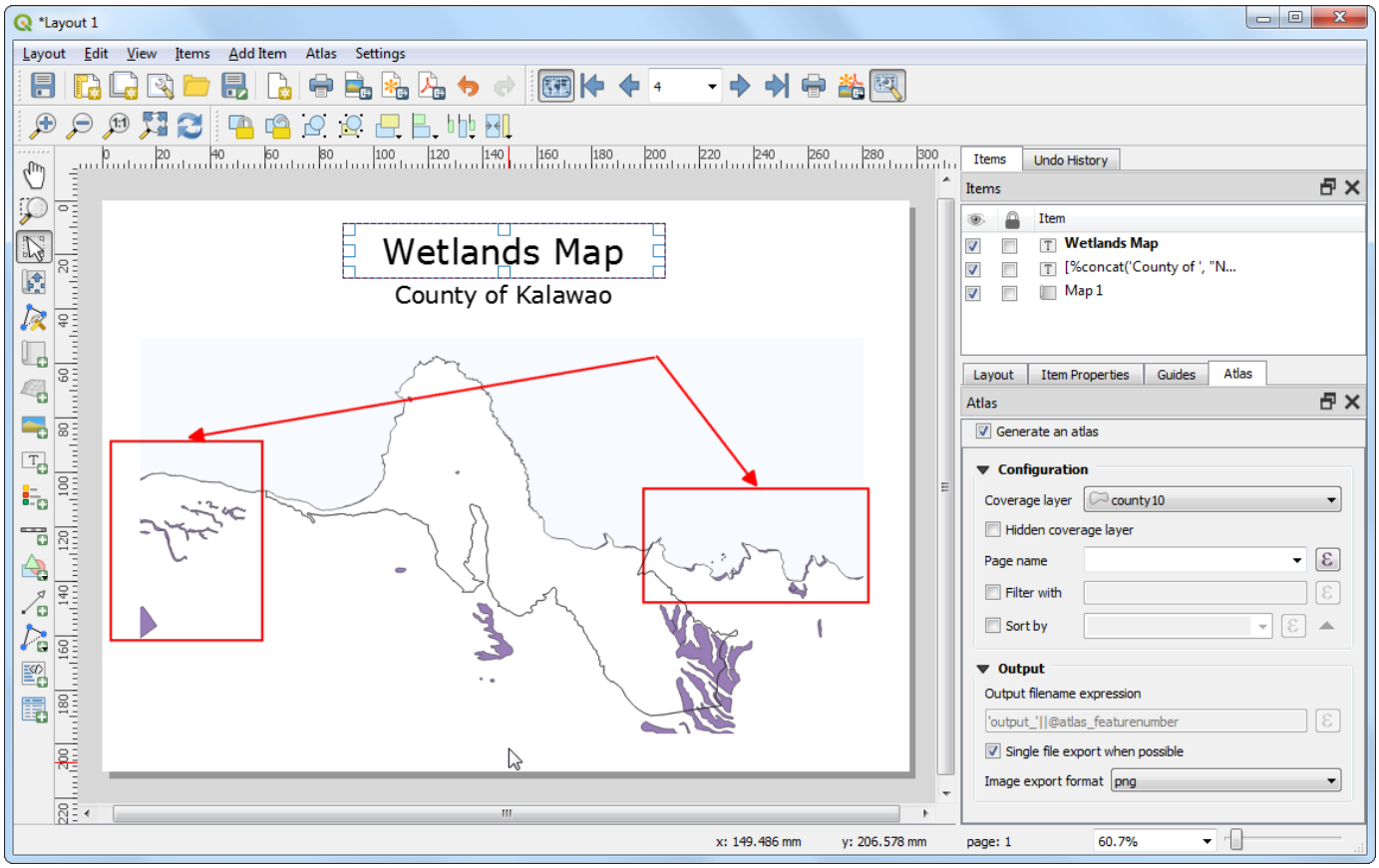


24. Switch to the Layout window. If we want the effect of the inverted polygons to show, we need to uncheck the Hidden coverage layer box under Atlas tab. Once unchecked, the rendered image will appear clean and areas outside the coverage polygon is not visible.

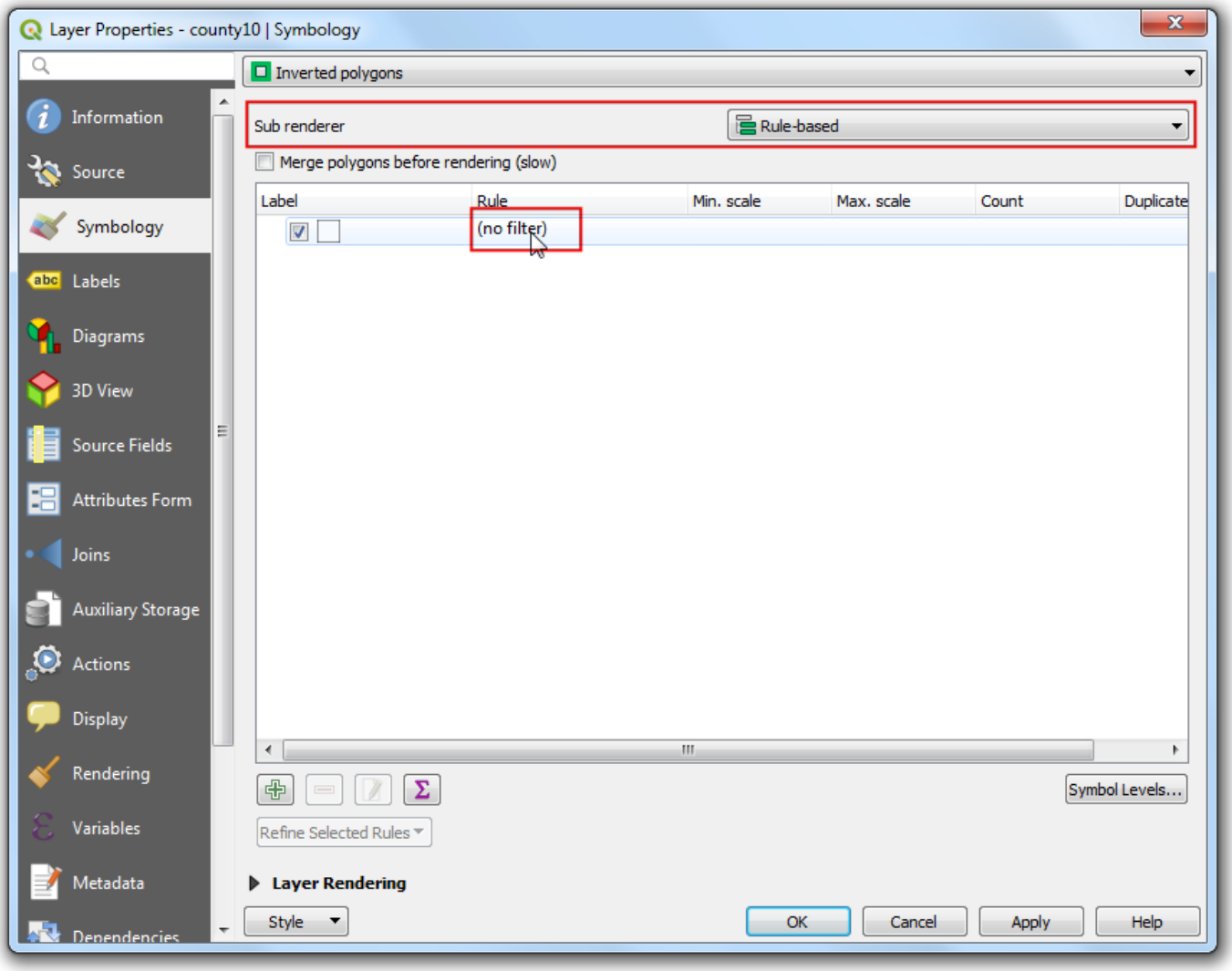


25. There is one more problem though. You will notice that in some cases, parts of the map that are outside the coverage layer boundary are still visible. This is because Atlas doesn't automatically hide other features. This can be useful in some cases, but for our purpose, we only want to show wetlands

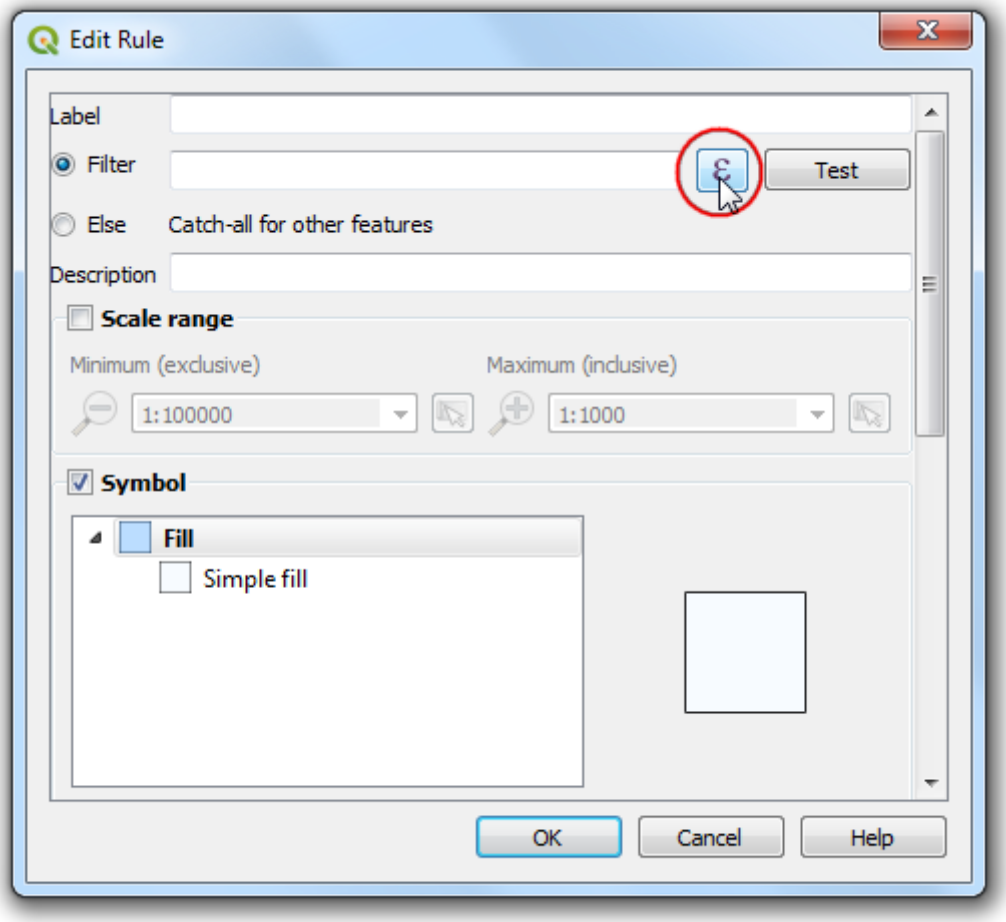
of the county whose map is being generated. To fix this, switch back to the main QGIS window and right-click the `county10` layer and select Properties.



26. In the Symbology tab, select `Rule-based` as the Sub renderer. Double-click the area under Rule.

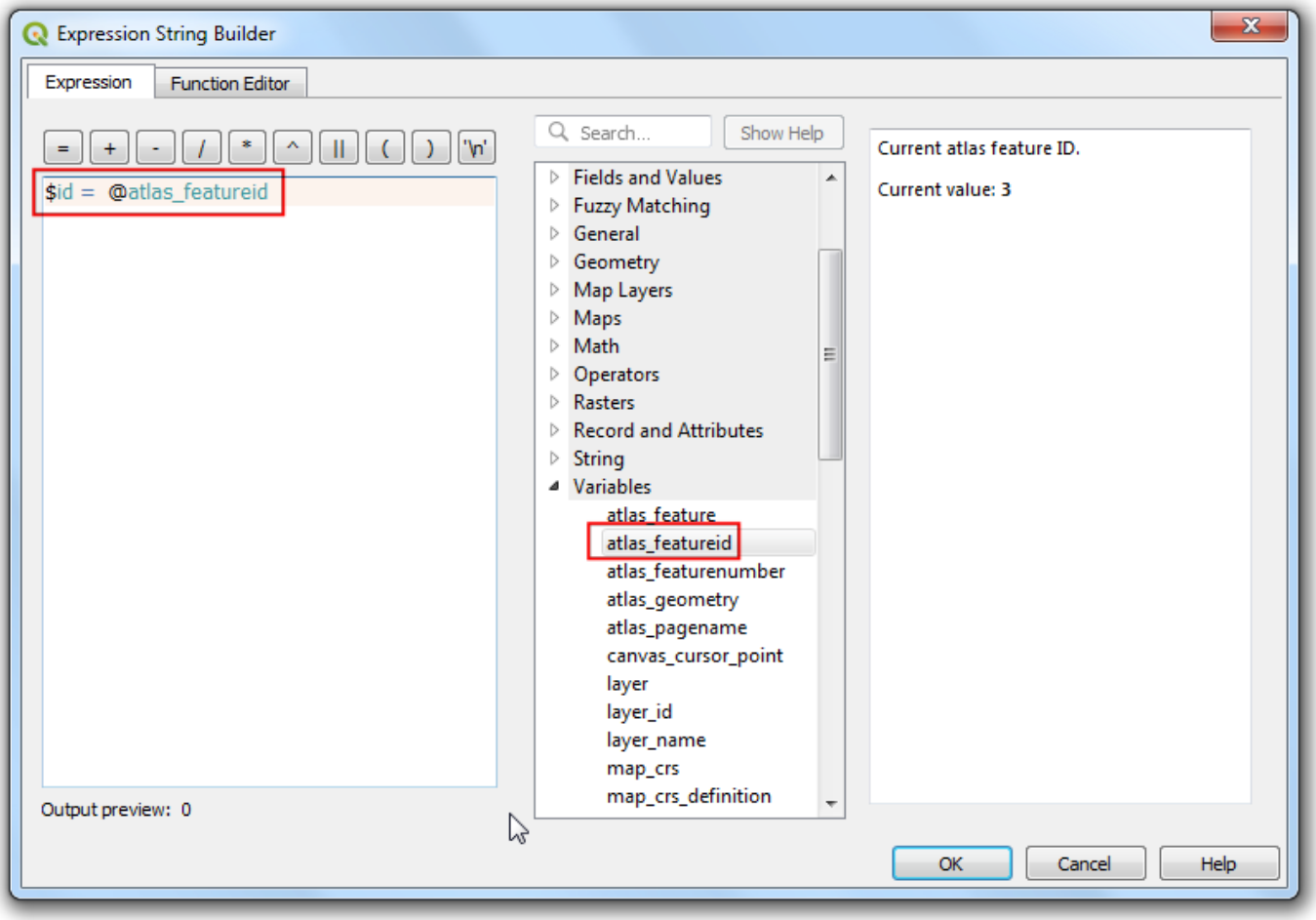


27. In the Edit rule dialog, click the Expression button next to Filter.



28. In the Expression string builder, expand the Variables group of functions. The `@atlas_featureid` variable stores the id of the the currently selected feature. We will construct an expression that will select only the currently selected Atlas feature. Enter the expression as below and click OK.

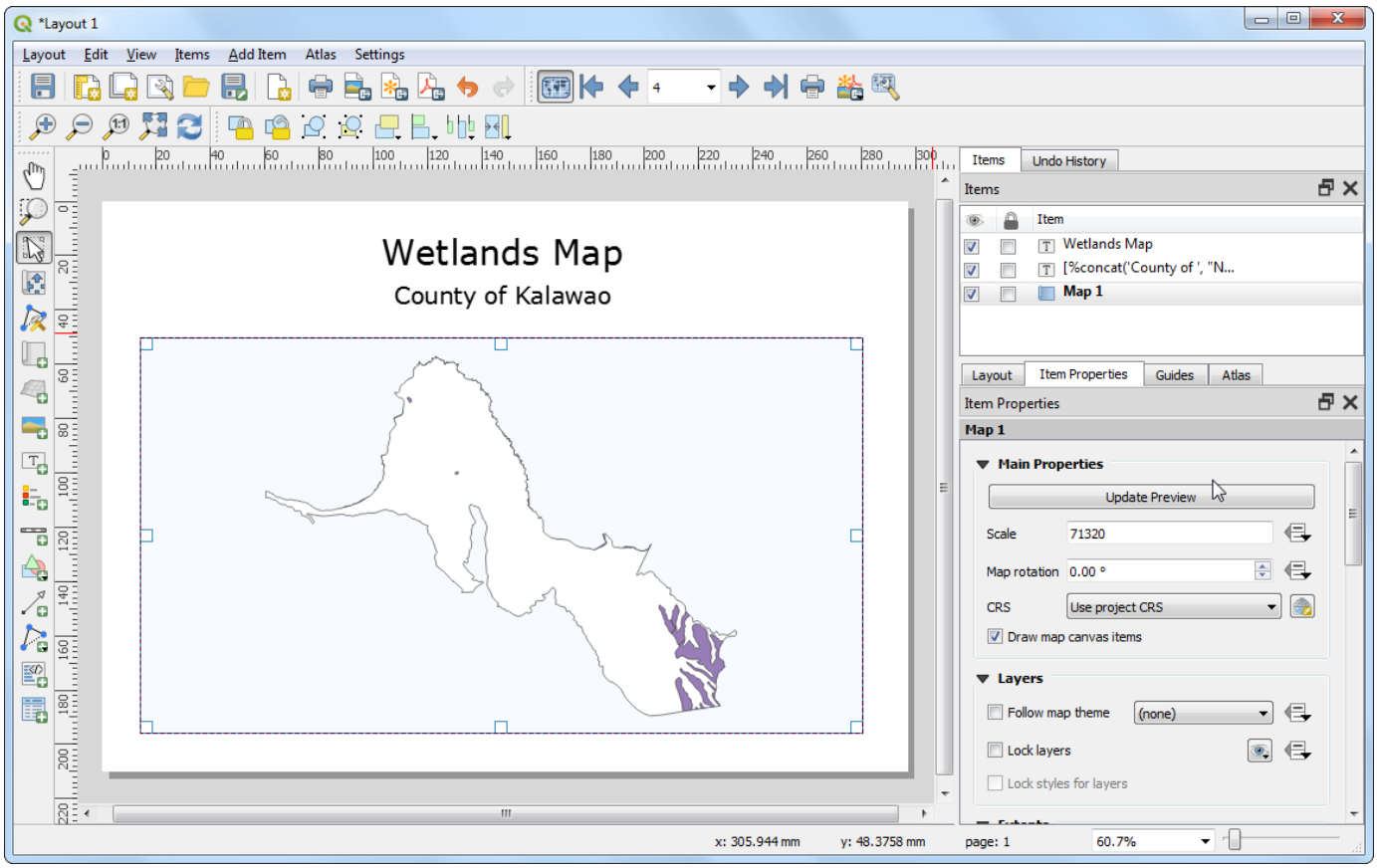
```
$id = @atlas_featureid
```



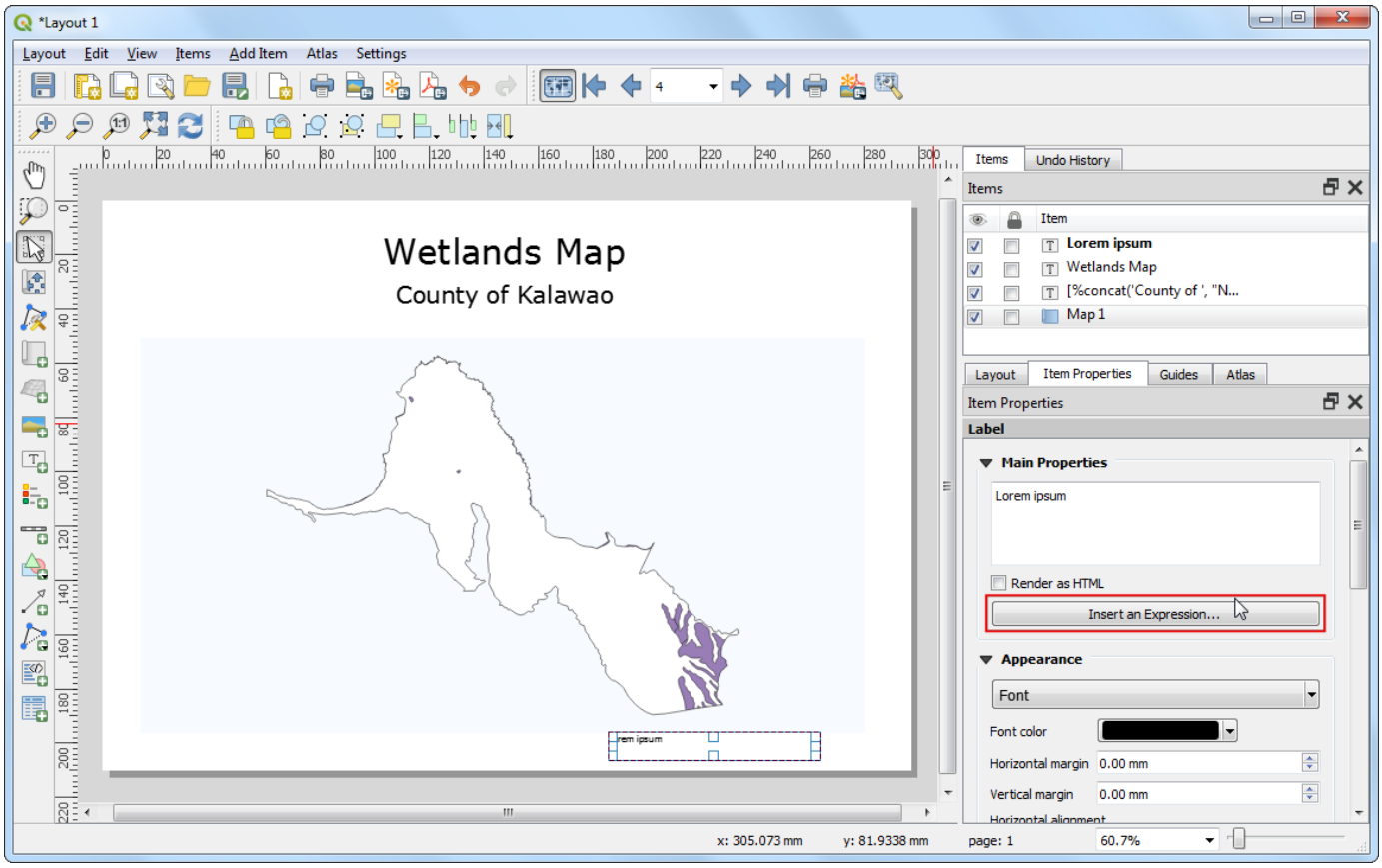
29. Close all intermediate dialogs and switch back to the Layout window. Select Map 1 item and click the Update preview button under Item properties tab to see the changes. Notice that now only the area covering the county boundary is shown.

Note

If you do not see the Update preview button, it may help to select another Item element first and then select Map 1 again.

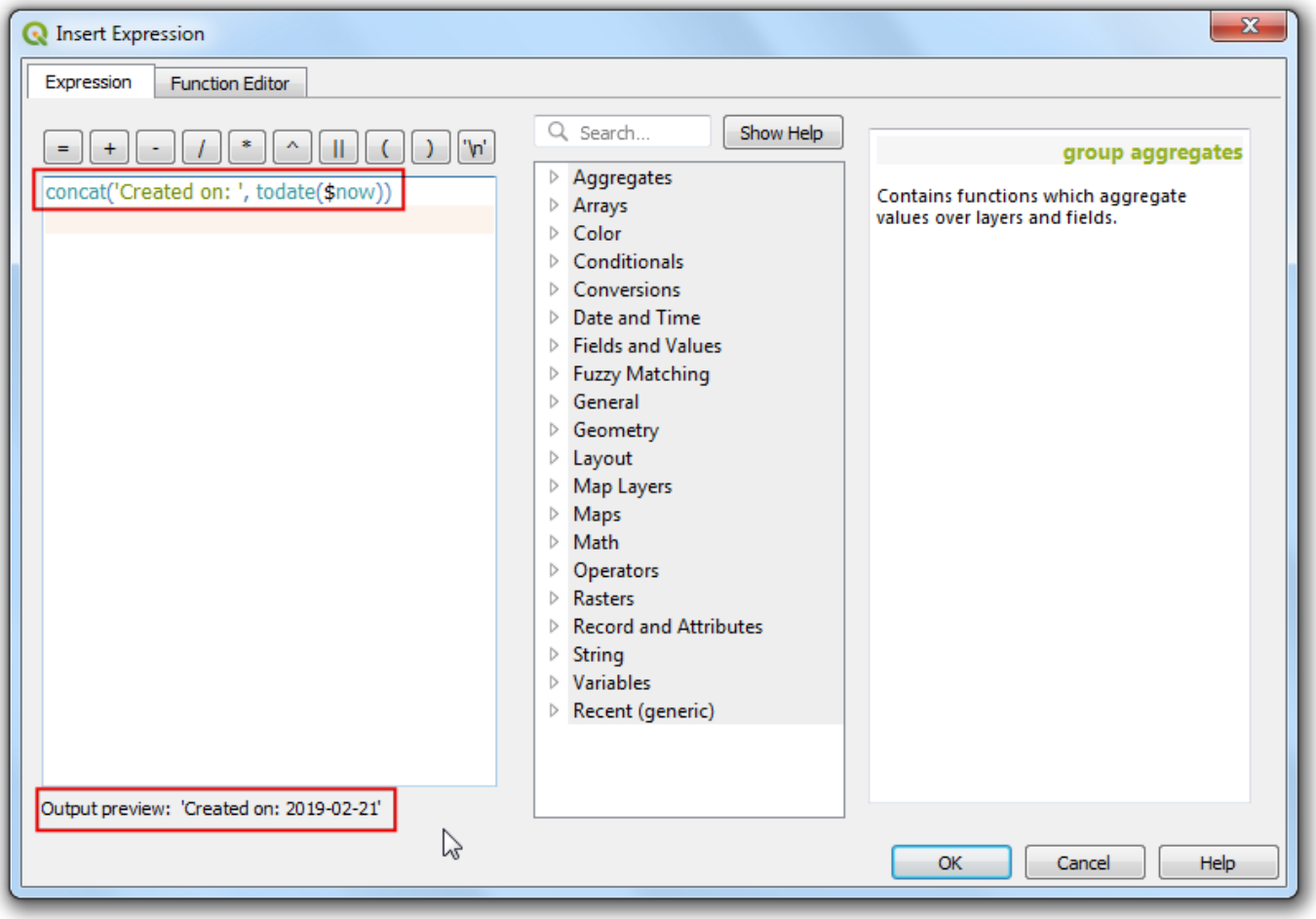


30. We will now add another dynamic label to show the current date. Go to Layout ▸ Add Label and select the area on the map. Click Insert an expression button.

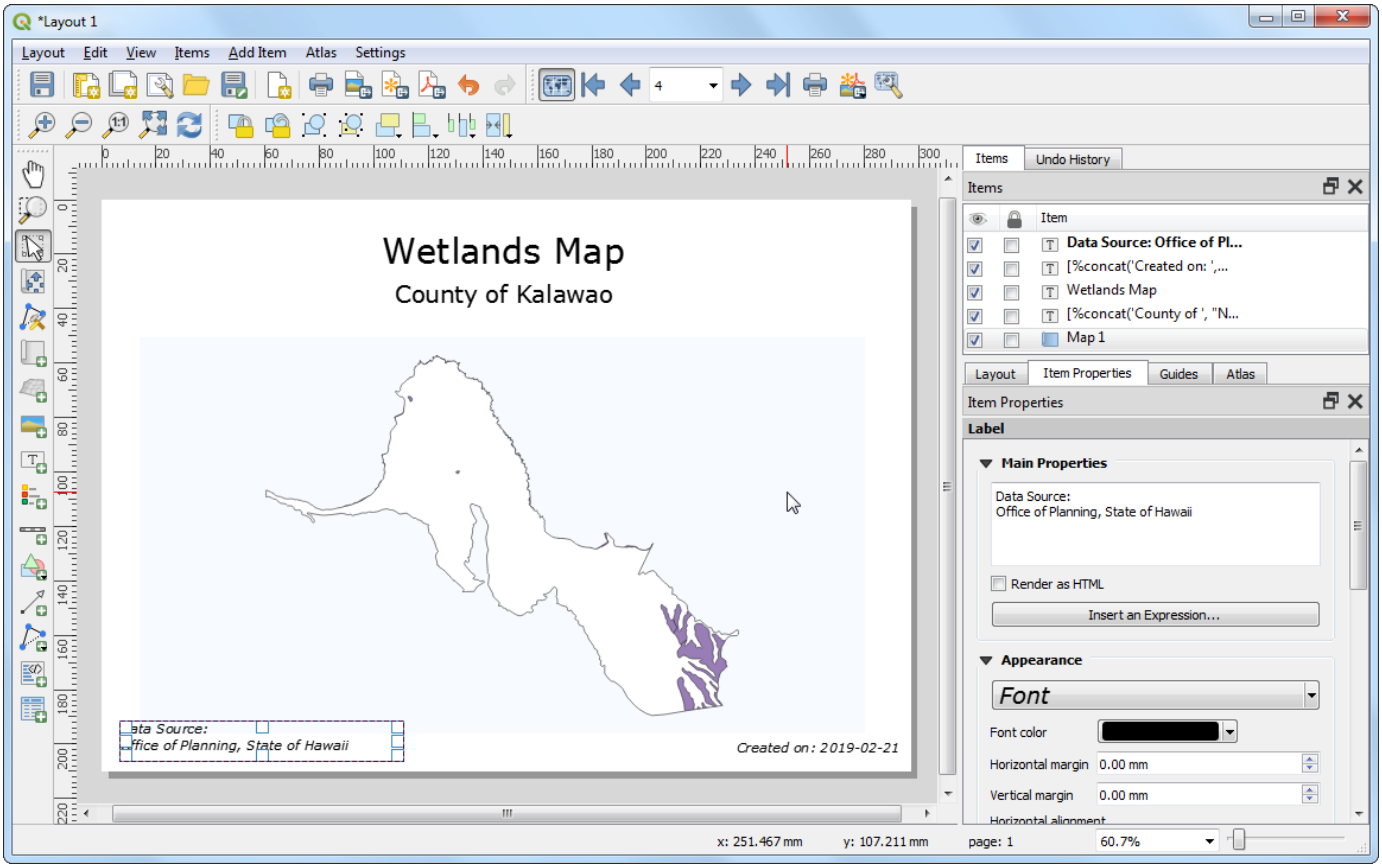


31. Expand the Date and Time functions group and you will find the `$now` function. This holds the current system time. The function `todate()` will convert this to a date string. Enter the expression as below and click OK.

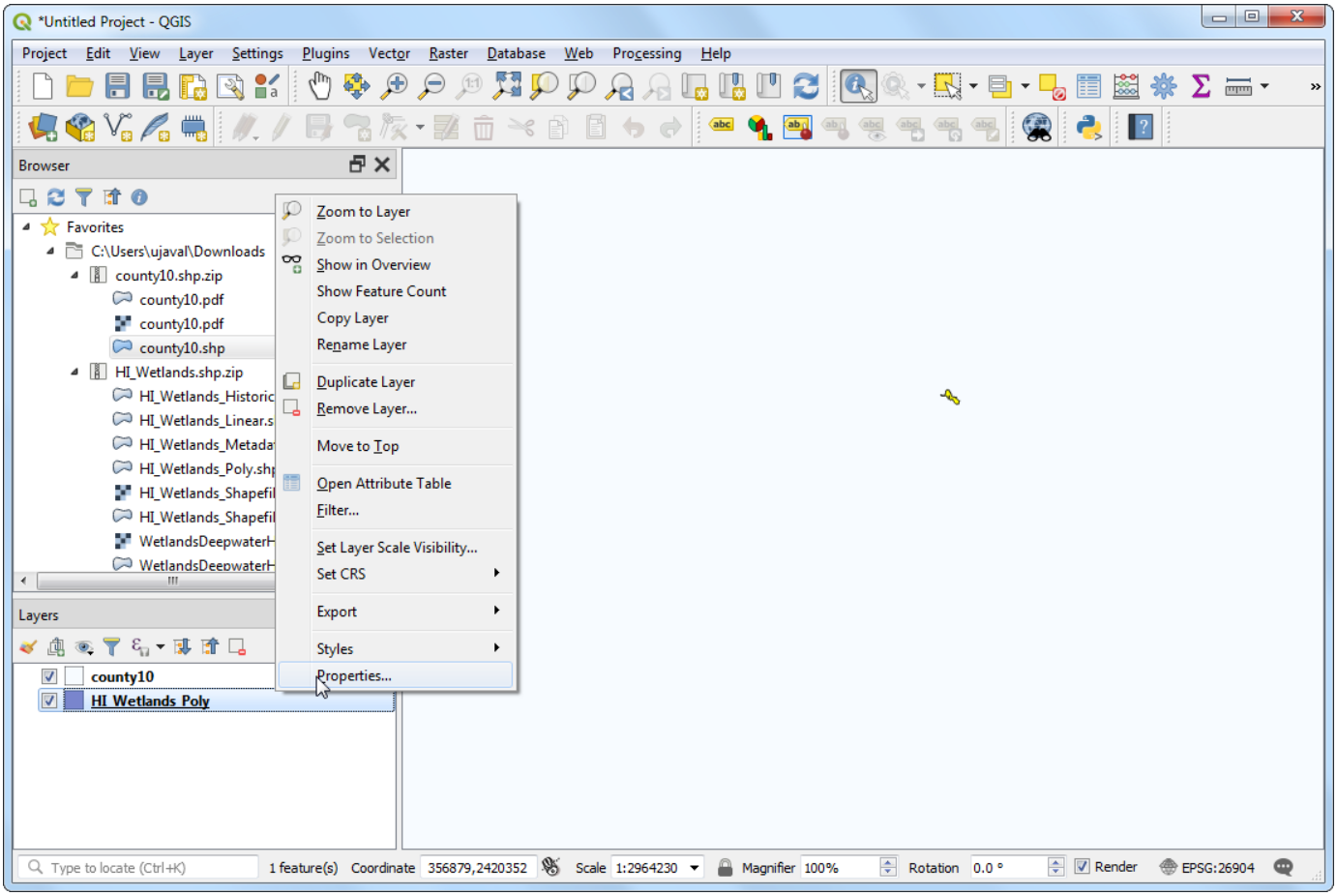
```
concat('Created on: ', todate($now))
```



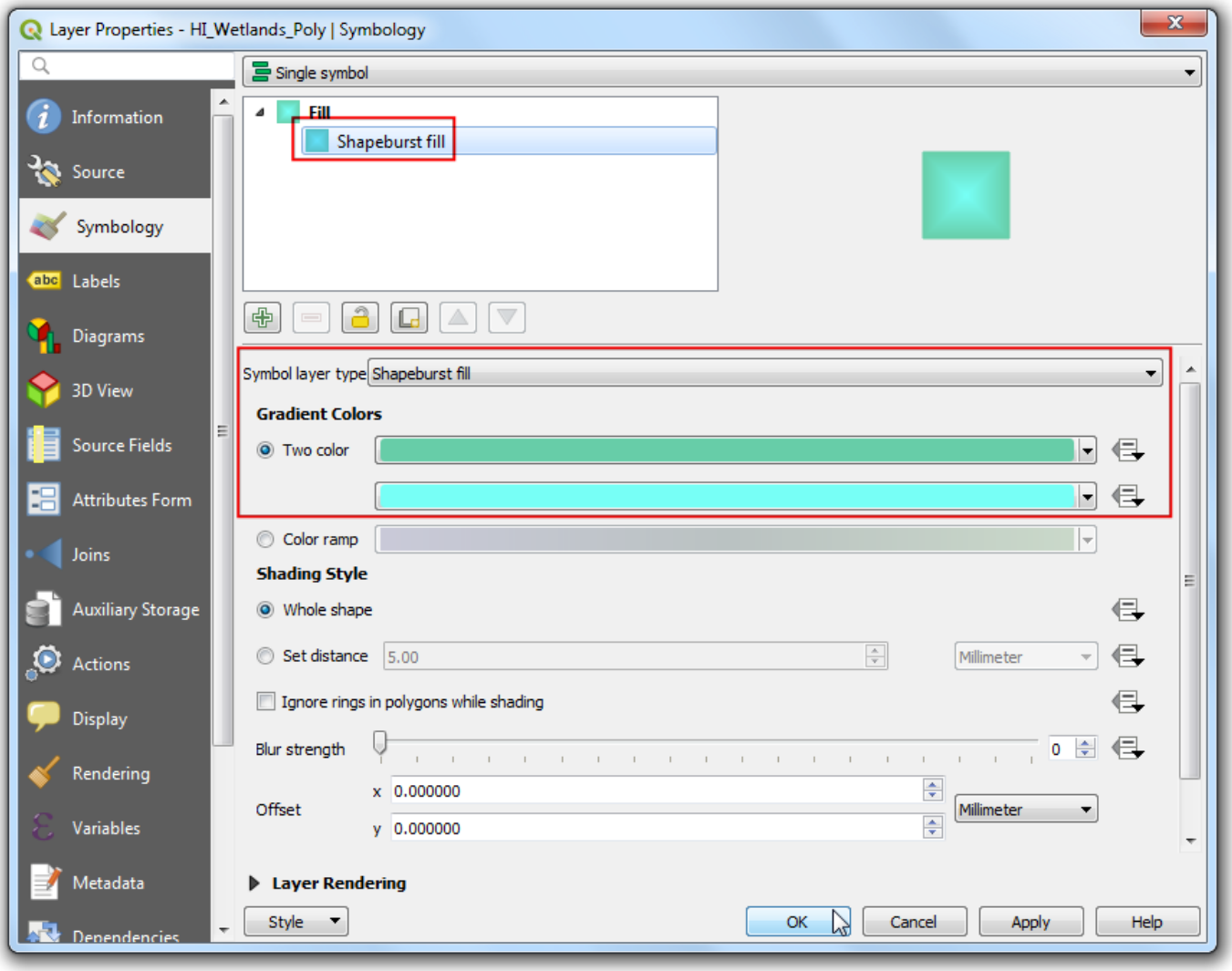
32. Add another label citing the data source. You may also add other map elements such as a north arrow, scalebar etc. as described in *Making a Map* (../making_a_map.html) tutorial.



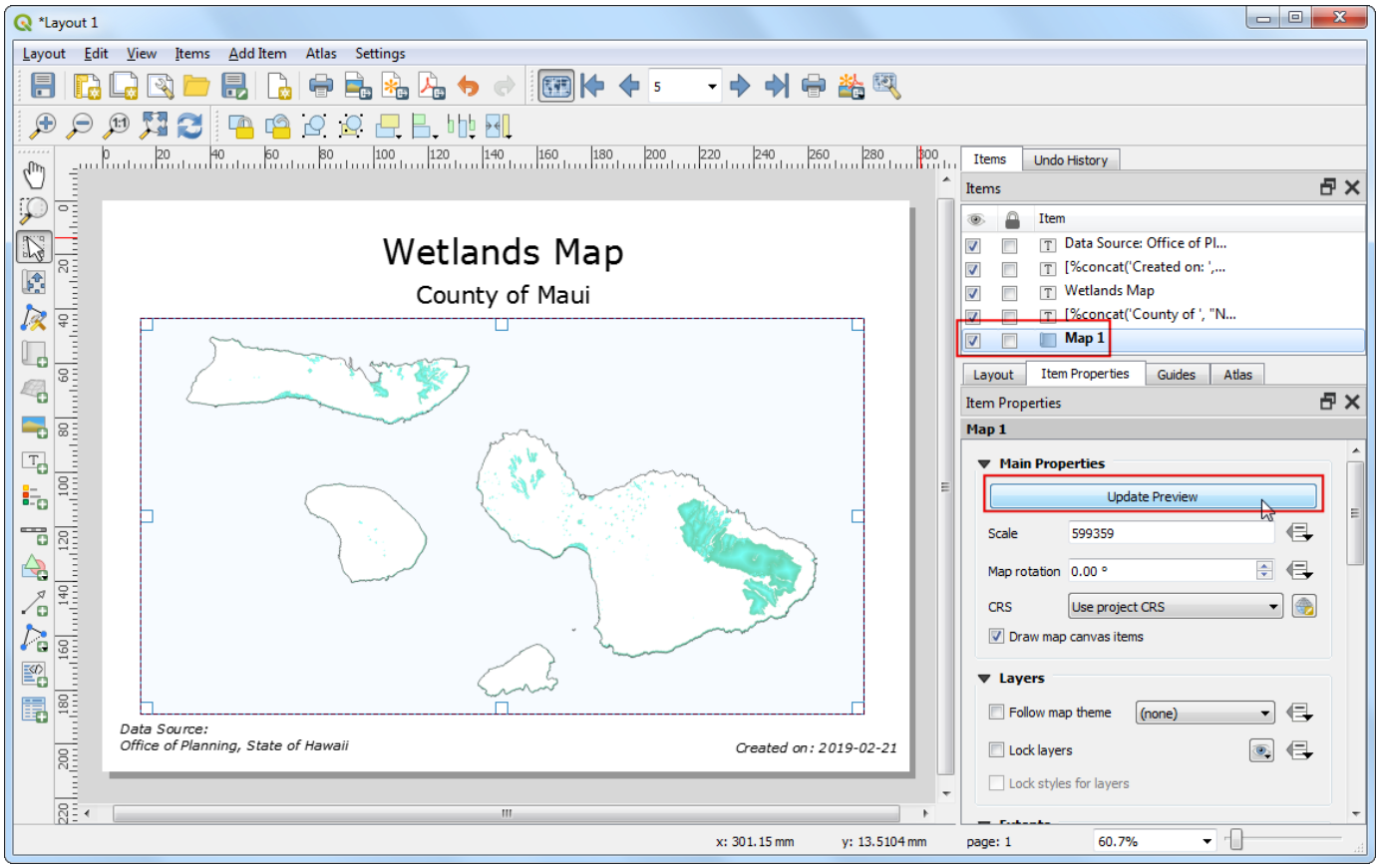
33. We will make one last styling improvement. Switch back to the main QGIS window and right-click the `HI_Wetlands_Poly` layer and select Properties.



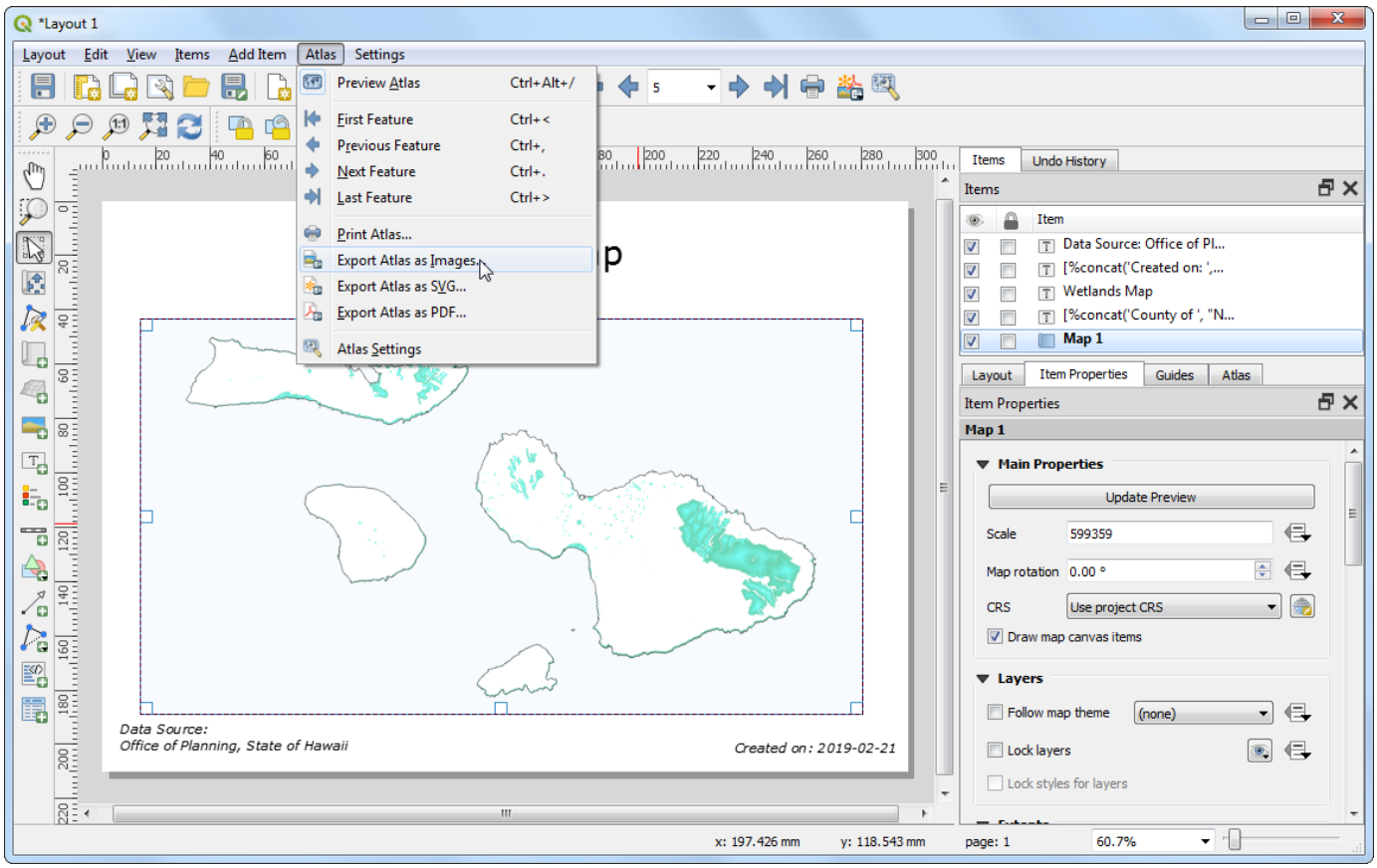
34. In the Symbology tab, click on Simple fill and select `Shapeburst fill` as the Symbol layer type. Choose the Two color option and select shades of green and blue that you like. Click OK.



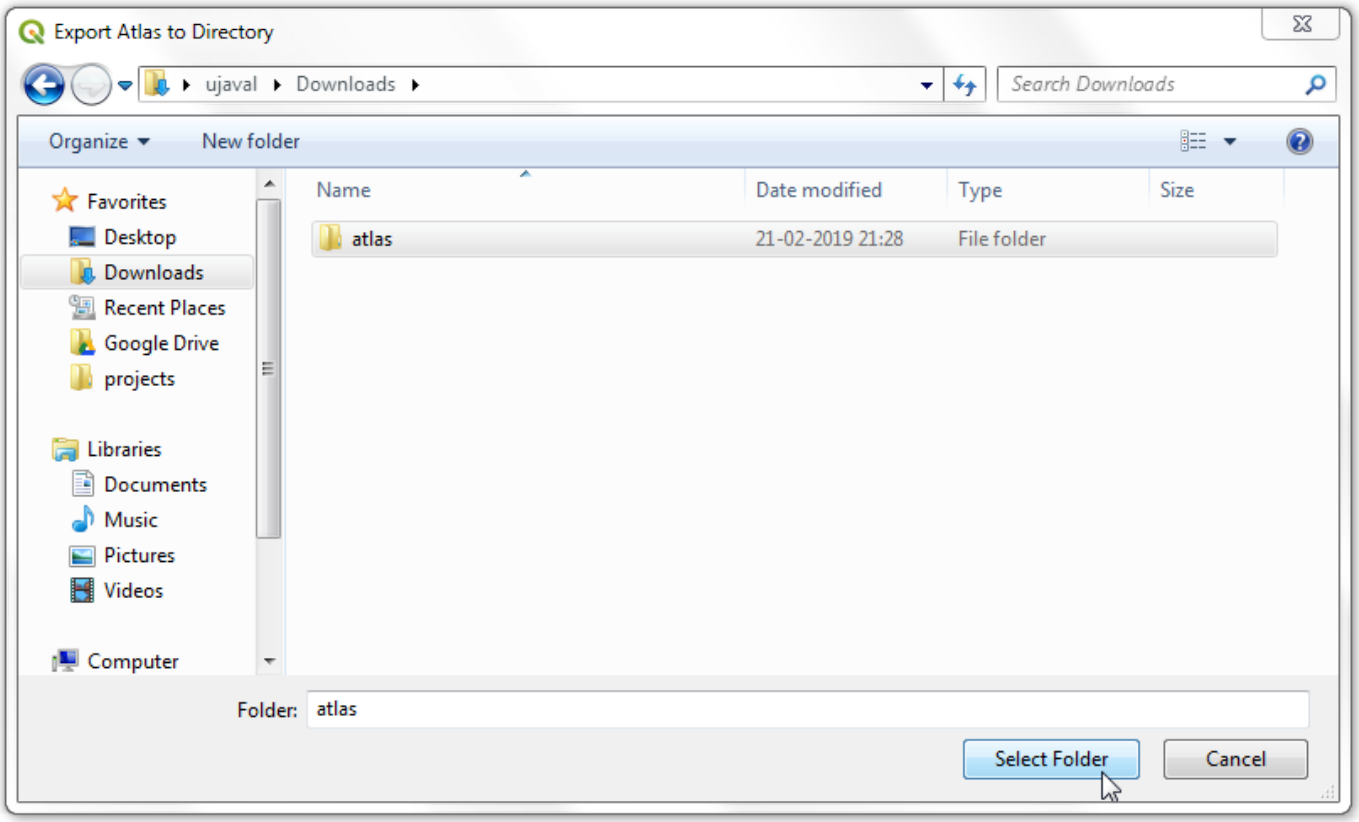
35. Select Map 1 item and click the Update preview button under Item properties tab to see the changes.



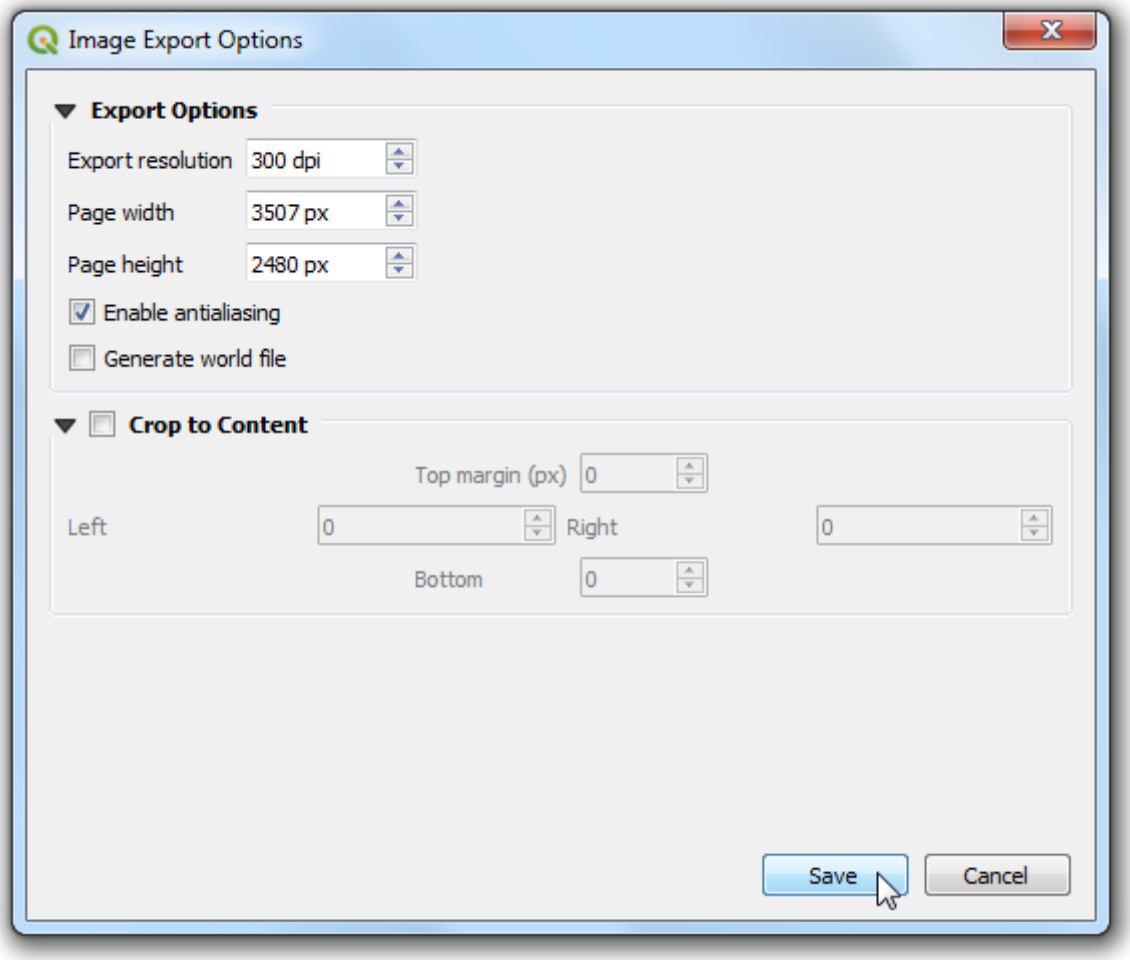
36. Once you are satisfied with the map layout and styling, go to Atlas ▸ Export Atlas as Images.



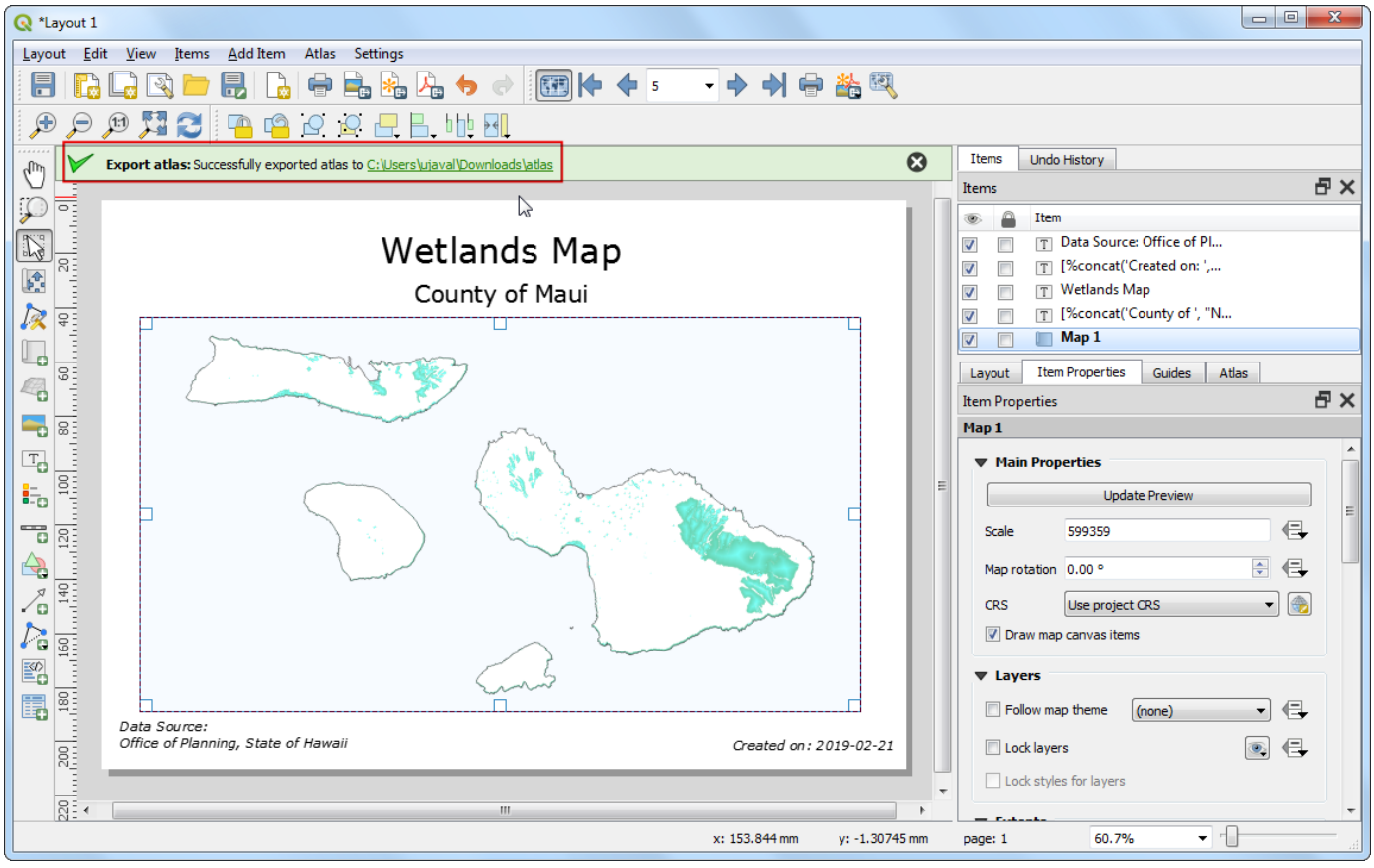
37. Select a directory on your computer and click Choose.



38. Leave the default options in the Image Export Options and click Save.



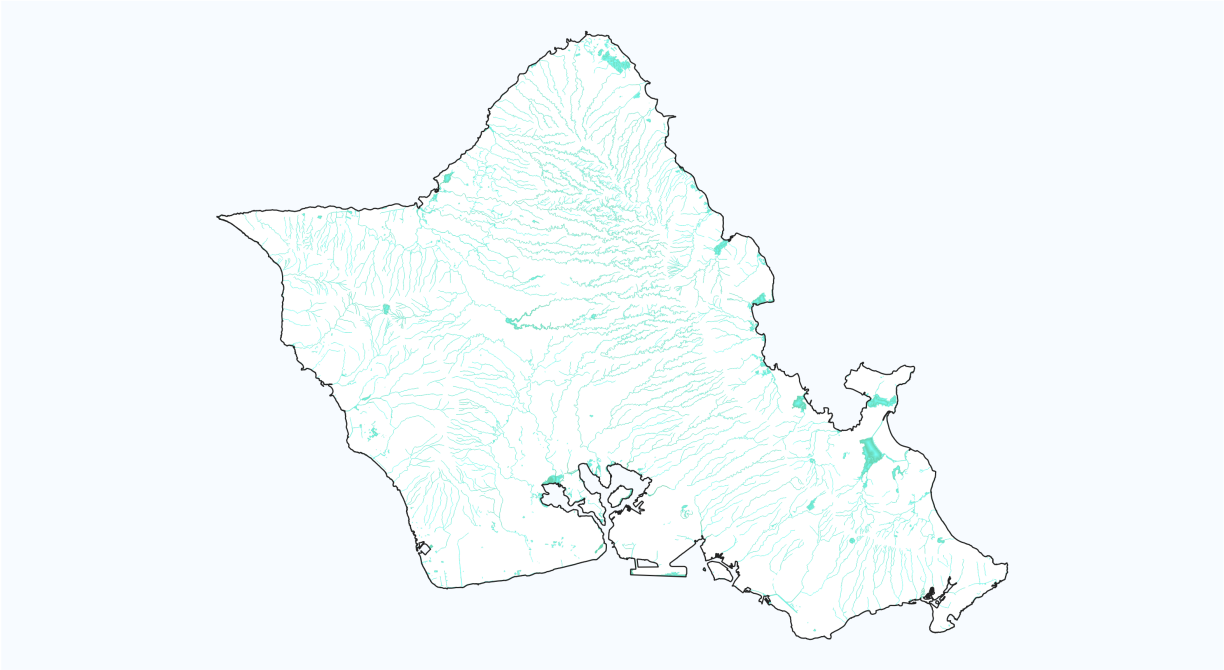
39. The Atlas tool will now iterate through each feature in the coverage layer and create a separate map image based on the template we created. You can see the images in the directory once the process completes.



40. Here are the map images for refeence.

Wetlands Map

County of Honolulu

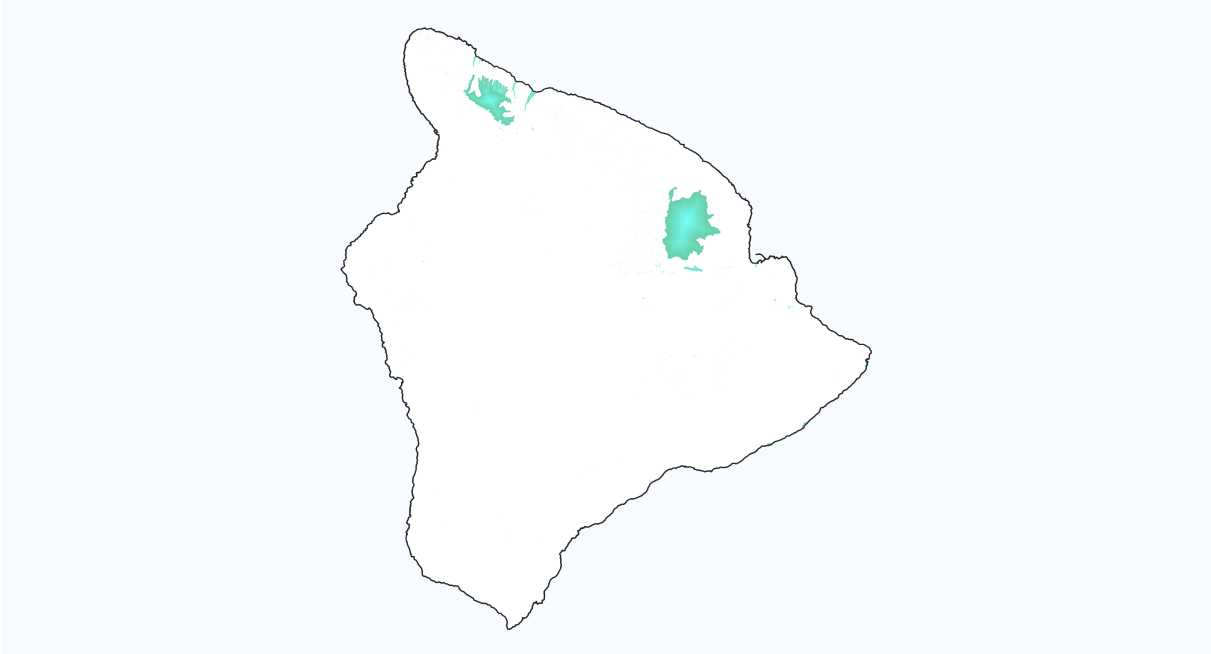


Data Source:
Office of Planning, State of Hawaii

Created on: 2019-02-21

Wetlands Map

County of Hawaii



Data Source:
Office of Planning, State of Hawaii

Created on: 2019-02-21

Wetlands Map

County of Kauai

Data Source:
Office of Planning, State of Hawaii

Created on: 2019-02-21

Wetlands Map

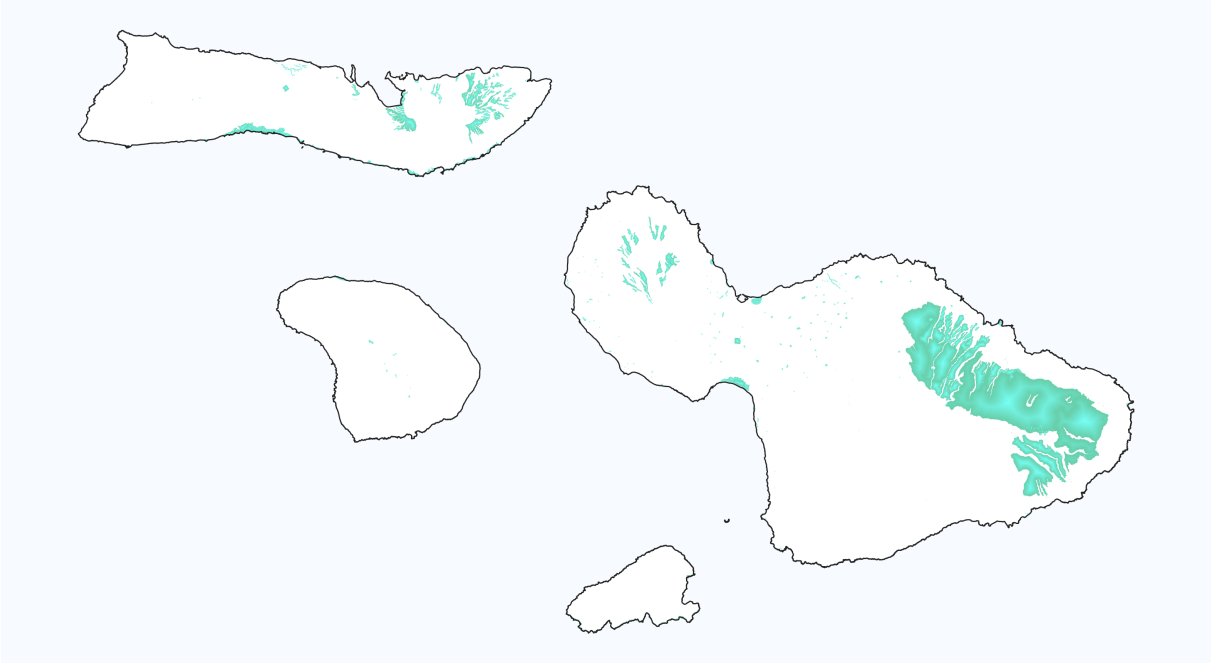
County of Kalawao

Data Source:
Office of Planning, State of Hawaii

Created on: 2019-02-21

Wetlands Map

County of Maui



Data Source:
Office of Planning, State of Hawaii

Created on: 2019-02-21

Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Basic Network Visualization and Routing (QGIS3)

Creating, visualizing, and managing networks is an important part of GIS. Many types of physical infrastructure such as roads, railways, utilities can be modeled as networks with lines and nodes - with properties attached to them. In this tutorial, we will learn how road networks are commonly modeled and apply some styling techniques to visualize the routing properties. We will also use QGIS3's built-in tools for network analysis that to find shortest path between 2 points along the network.

Overview of the task

We will take a layer of road centerlines for Washington DC, visualize the connectivity and build a network to find shortest path between any 2 points in the city.

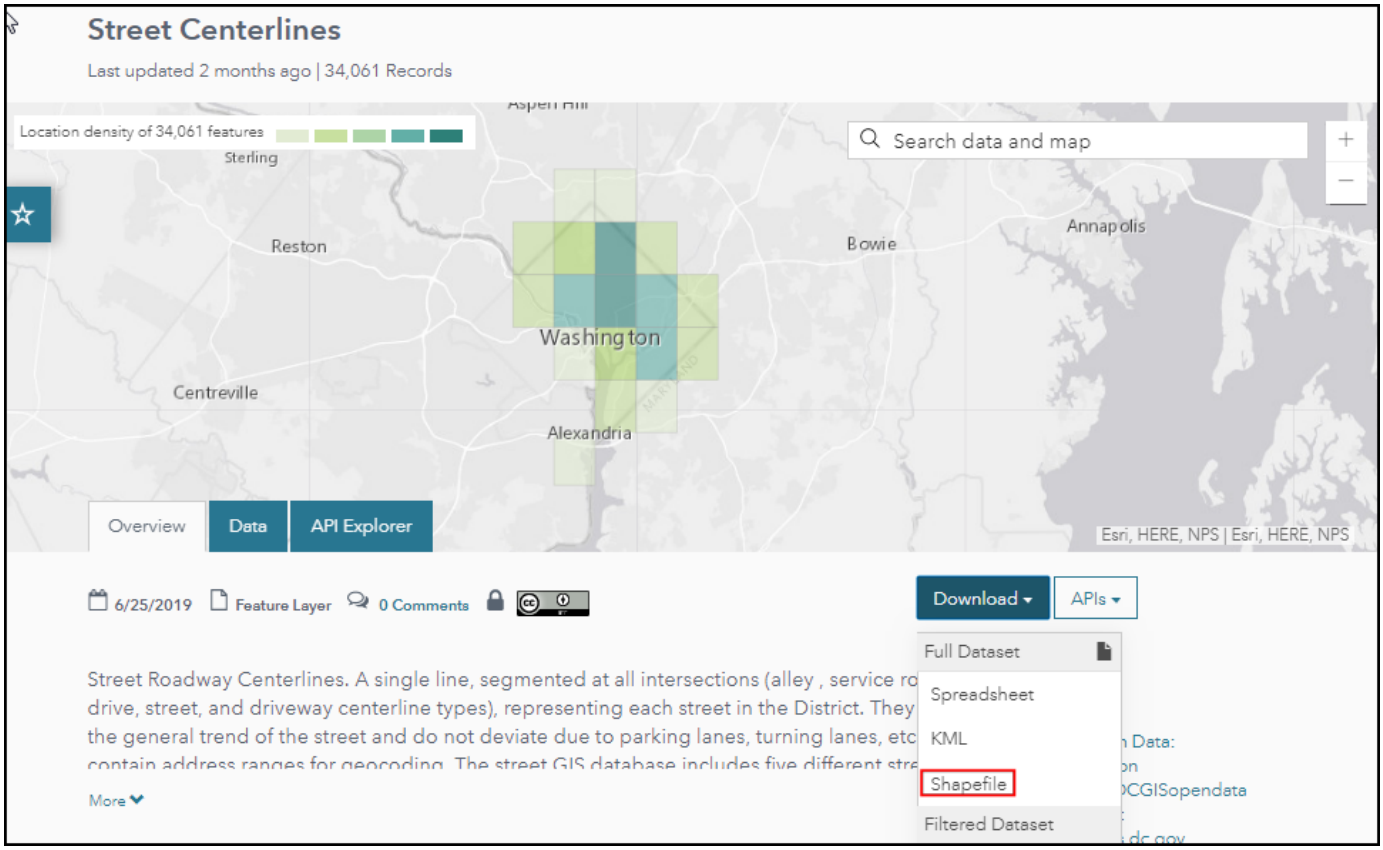
Other skills you will learn

- How to use data defined overrides to align an arrow symbol based on line direction.

Get the data

District of Columbia government freely shares hundreds of datasets on the Open Data Catalog (https://opendata.dc.gov/).

Download the Street Centerlines (https://opendata.dc.gov/datasets/street-centerlines) shared by DCGISopendata data as a shapefile.



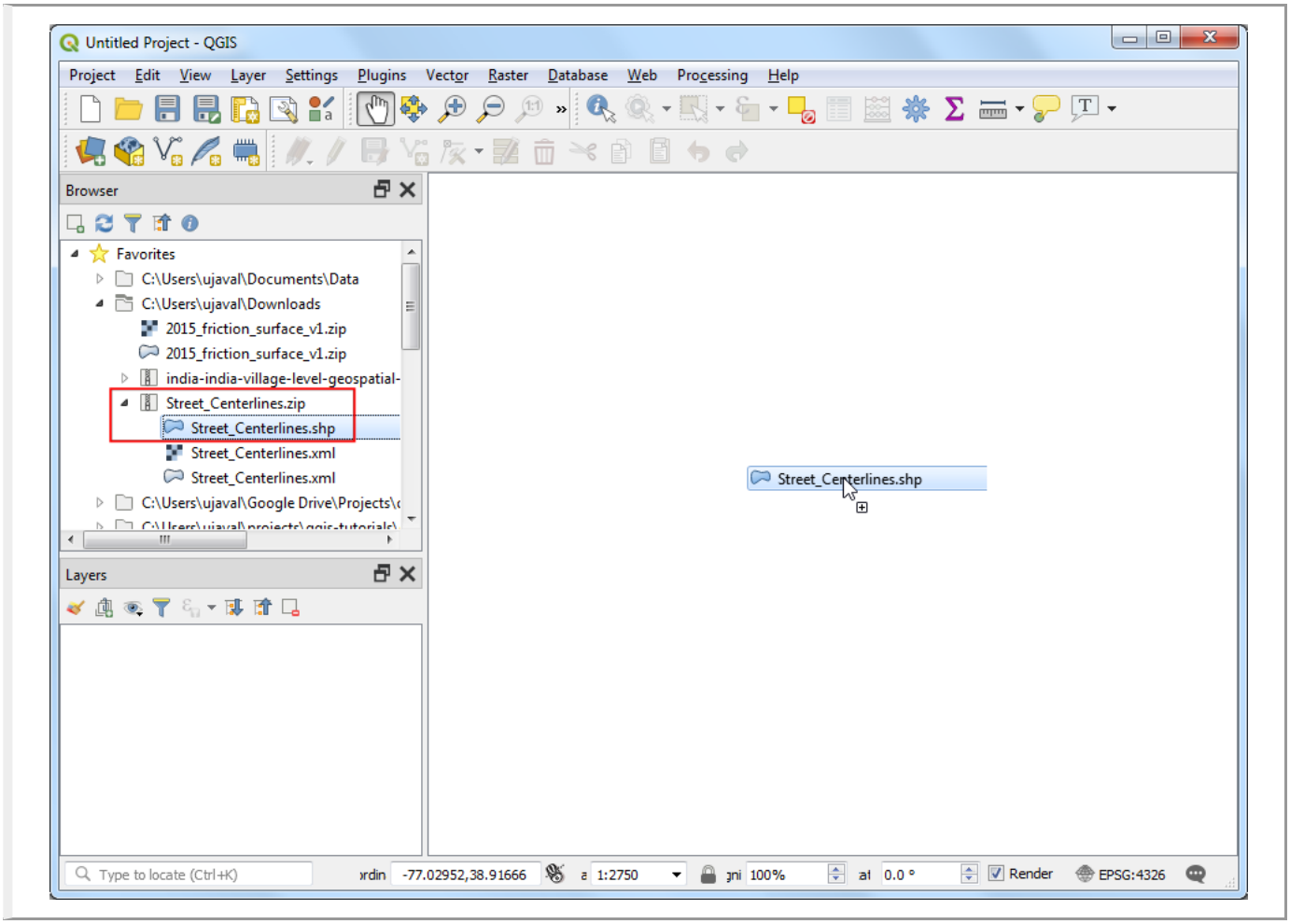
For convenience, you may directly download a copy of the datasets from the links below:

Street_Centerlines.zip (http://www.qgistutorials.com/downloads/Street_Centerlines.zip)

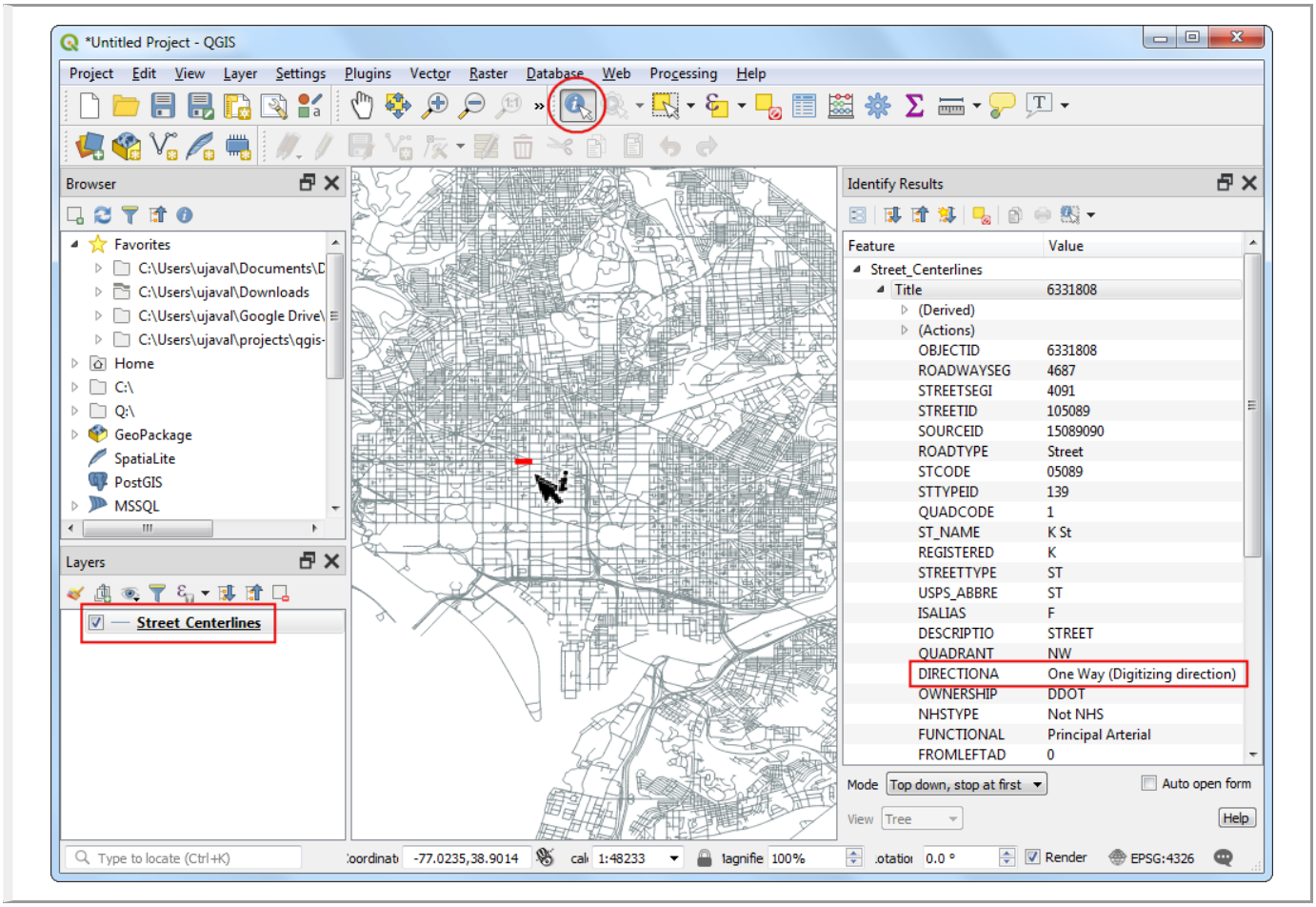
Data Source: [DCOPENDATA] (../credits.html#dcopendata)

Procedure

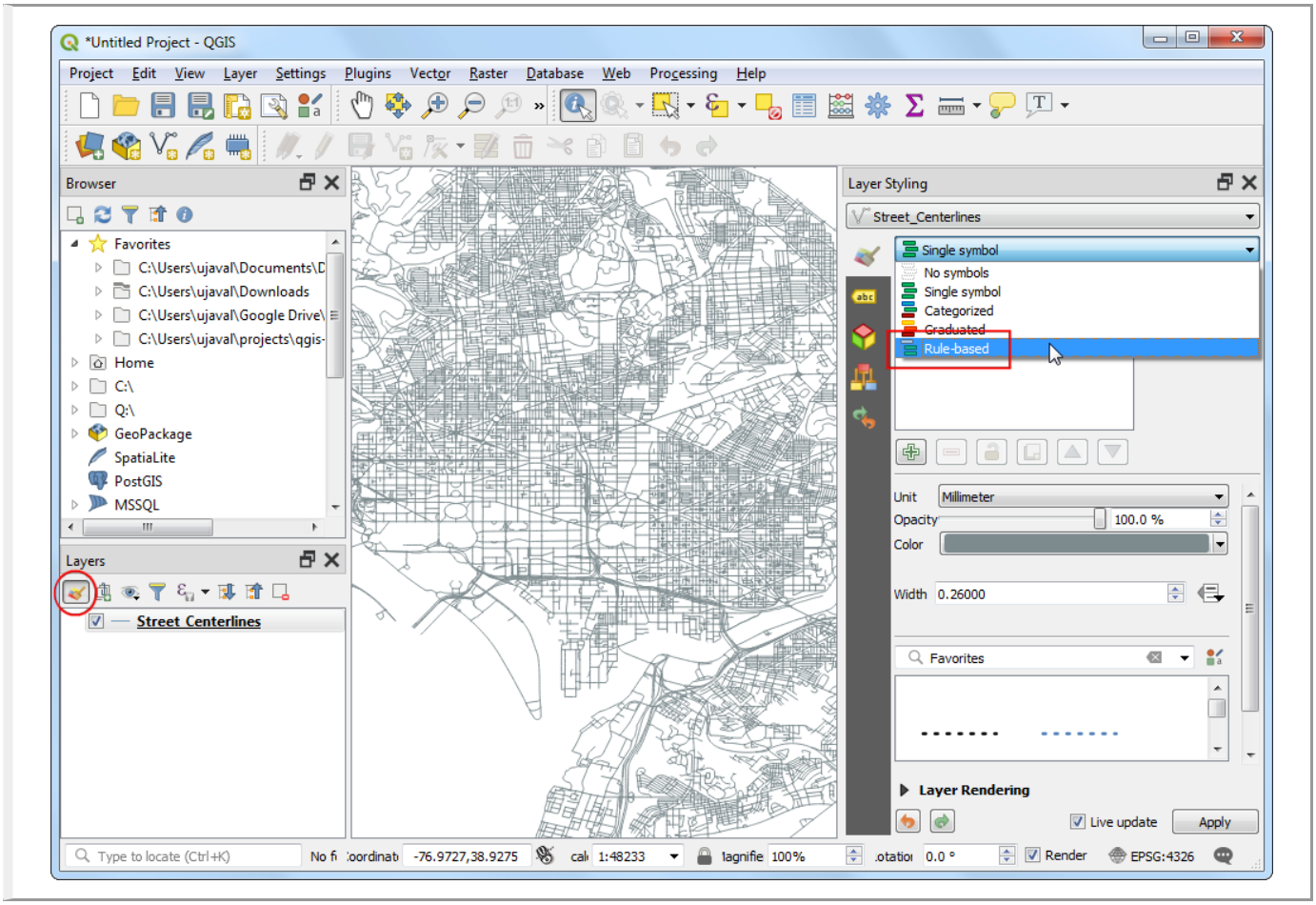
1. Locate the downloaded `Street_Centerlines.zip` file in the Browser panel. Expand it and drag the `Street_Centerlines.shp` file to the canvas.



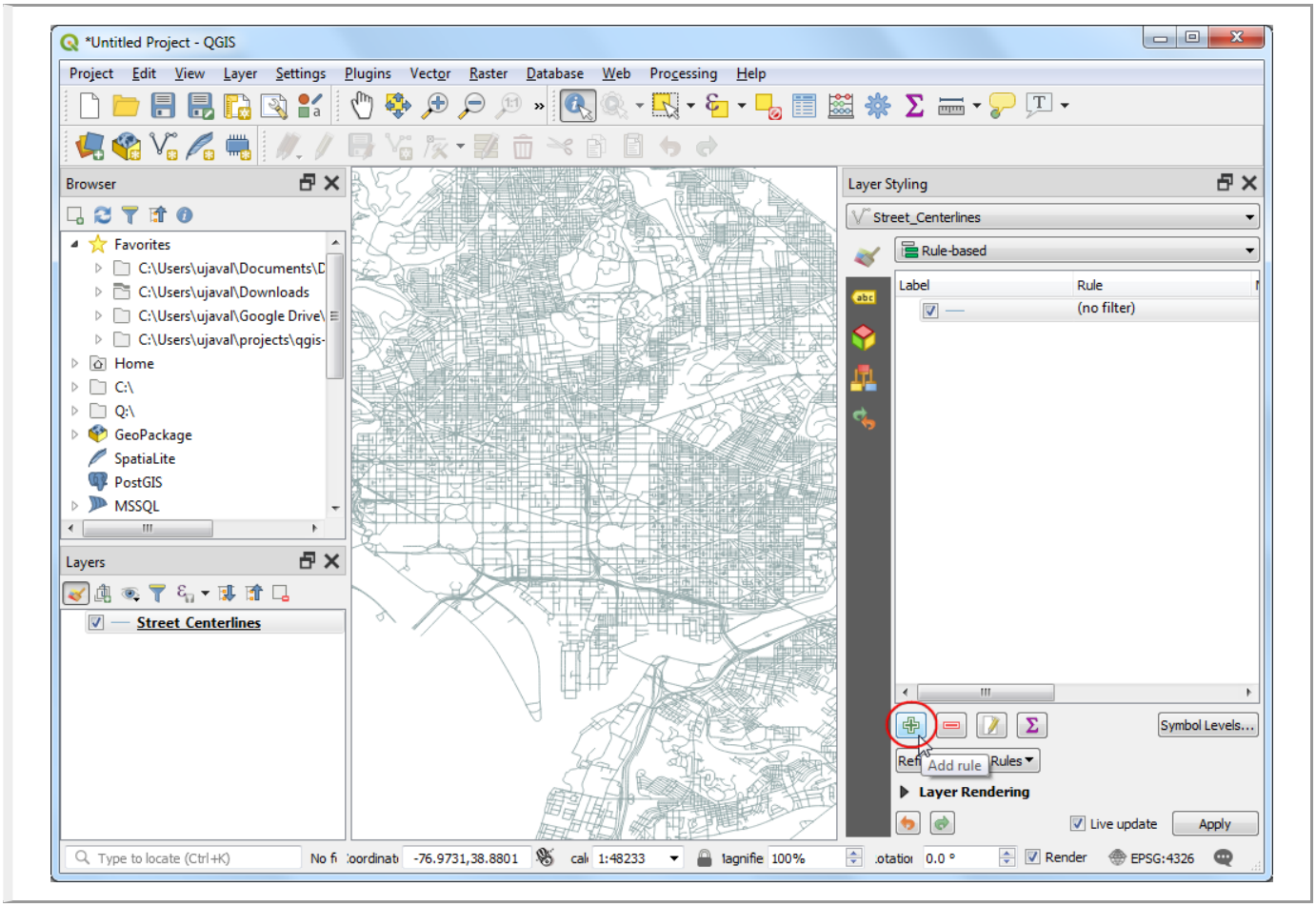
2. You will see a new line layer called `Street_Centerlines` added to the Layers panel. This layer represents each road in Washington DC. Select the Identify tool in the Attributes Toolbar. Click on any road segment to see what attributes are attached to it. There are standard attributes like road name, type etc. there is an attribute called `DIRECTIONA` . This is an import attribute for routing as it specifies whether the segment is two-way or one-way. It contains 4 different values. `Two Way` for two-way streets. `One Way (Digitizing direction)` for one-way streets where the traffic is allowed in the direction of the line (start-point to end-point) and `One way (Against digitizing direction)` for one-way streets where the traffic flows in the opposite direction of the line. There is also `Unknown` value where we will assume two-way traffic. We will now use the information in that attribute to display an arrow on streets that are one-way.



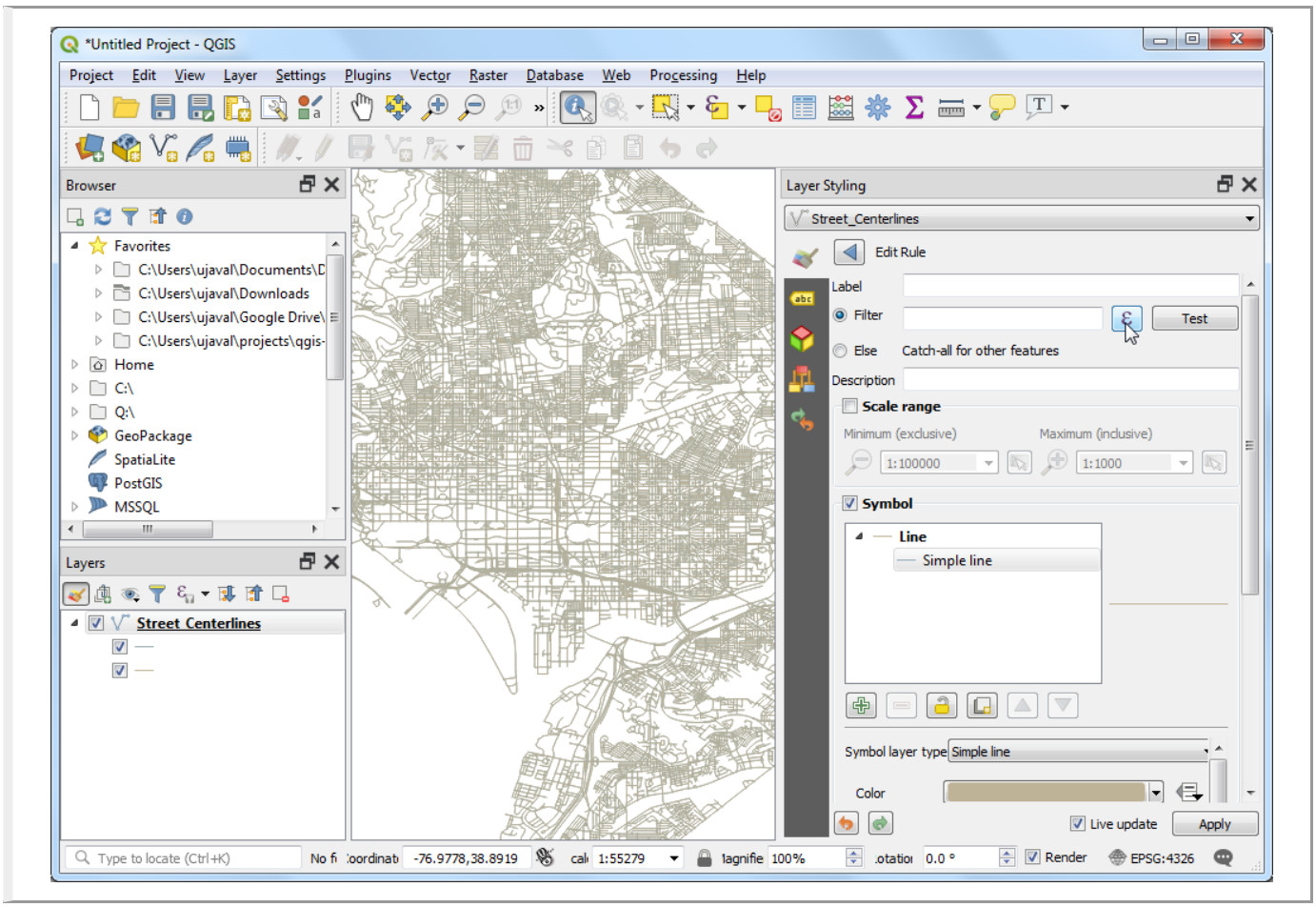
3. Click the Open the layer Styling Panel button in the Layers panel. Select the `Rule-based` renderer from the drop-down menu.



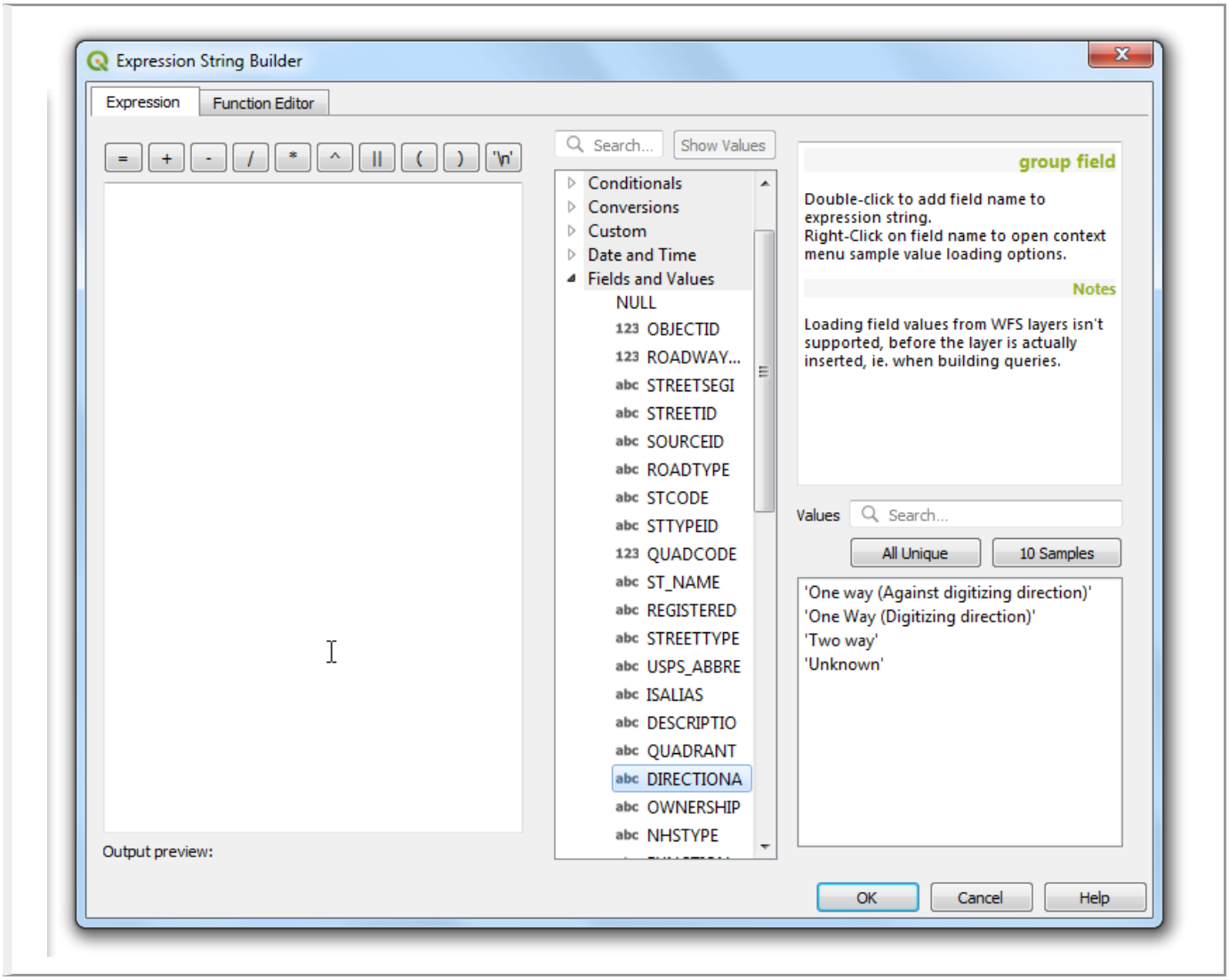
4. We will create a new style with a filter for only the one-way roads. Click the Add rule + button.



5. In the Edit rule dialog, click the Expression button.

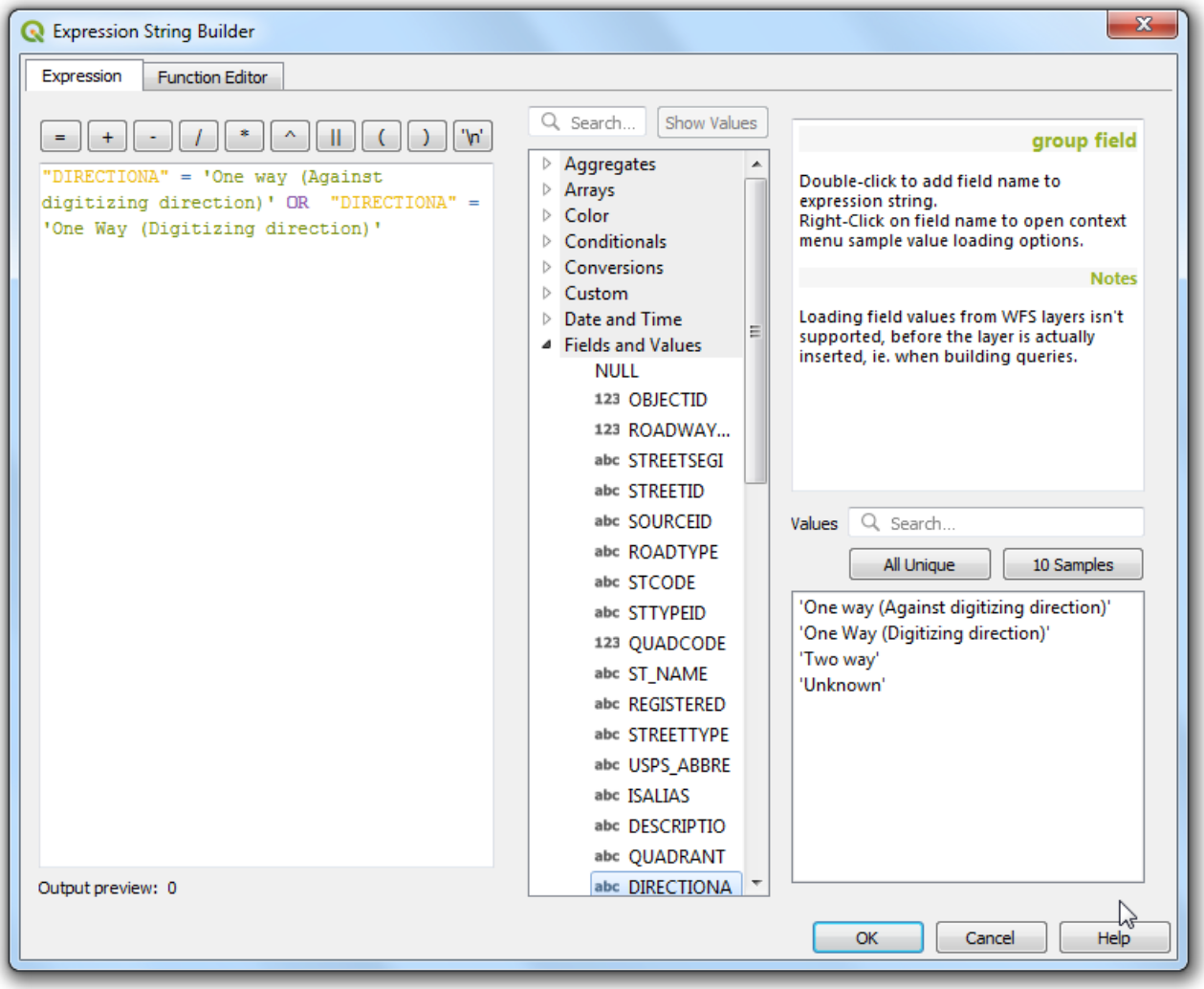


6. In the Expression string builder dialog, expand the Fields and Values section in the middle-panel. Select `DIRECTIONA` attribute and click All Unique in the right-hand panel. The 4 values that we discussed earlier will appear. Having these values here as a reference helps when building the expression. Also, you can double-click on any value to add them to the expression.

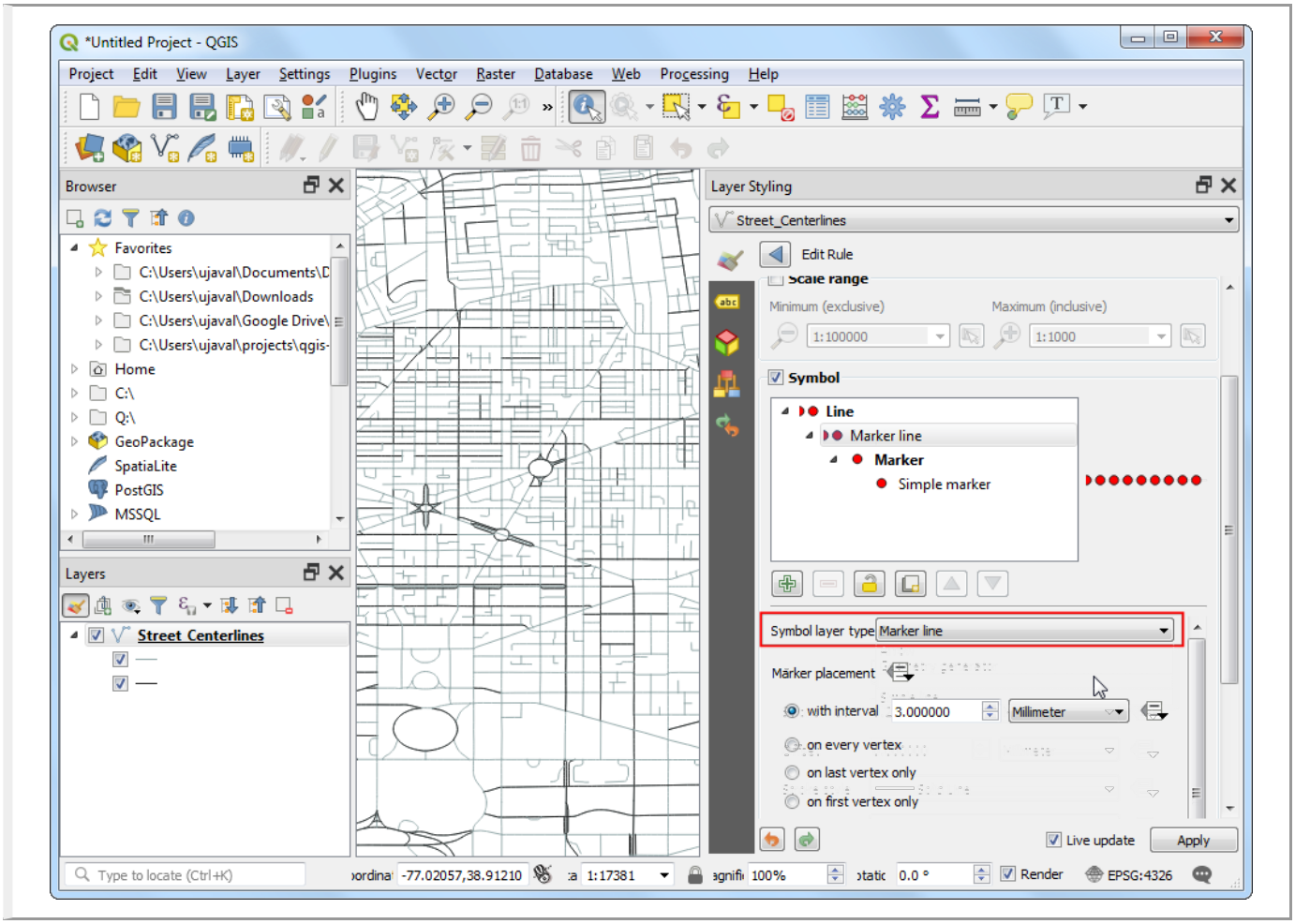


7. The goal is to create an expression that selects all one-way streets. Enter the following expression and click OK.

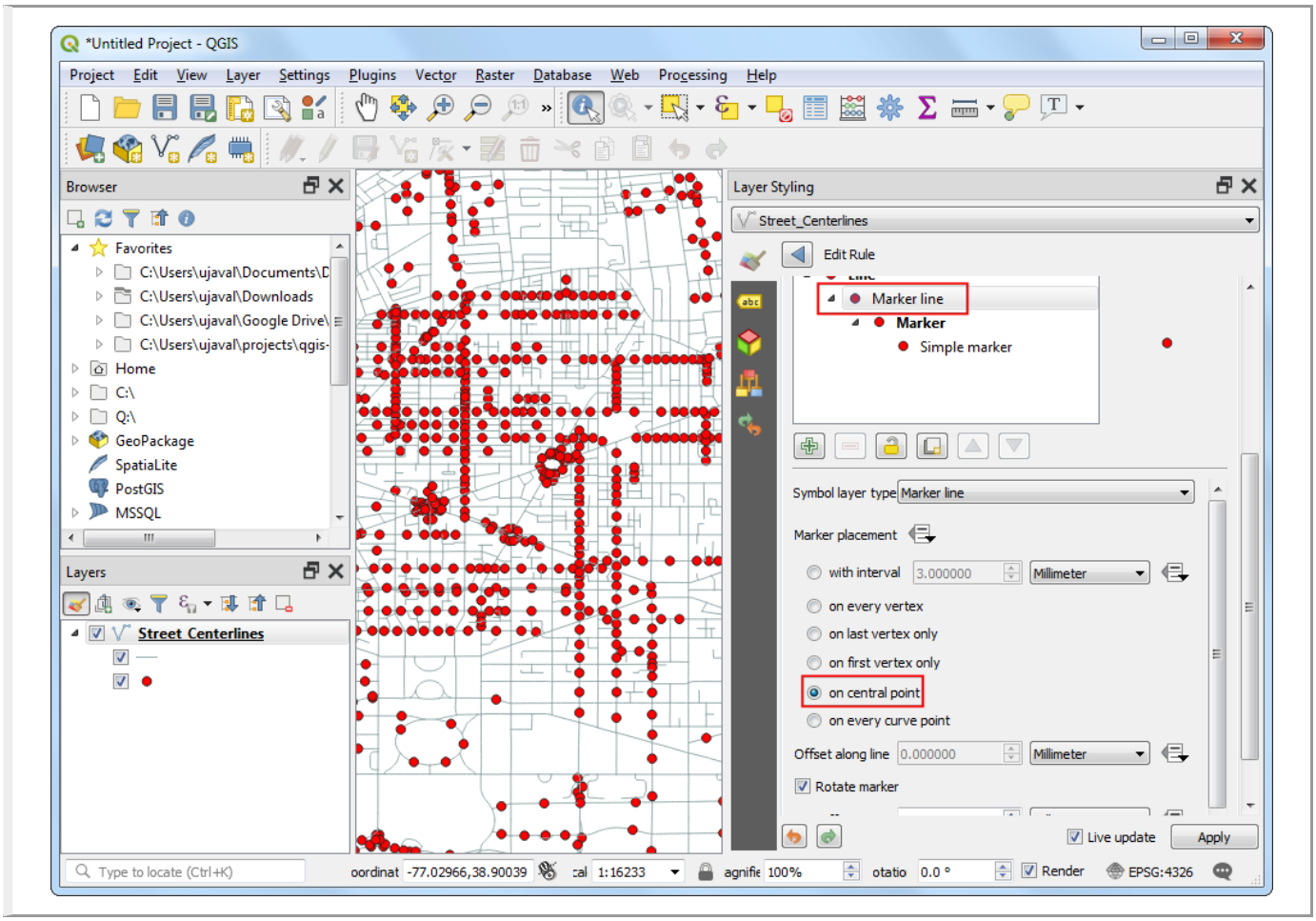
```
"DIRECTIONA" = 'One way (Against digitizing direction)' OR  "DIRECTIONA" ='One Way (Digi
```



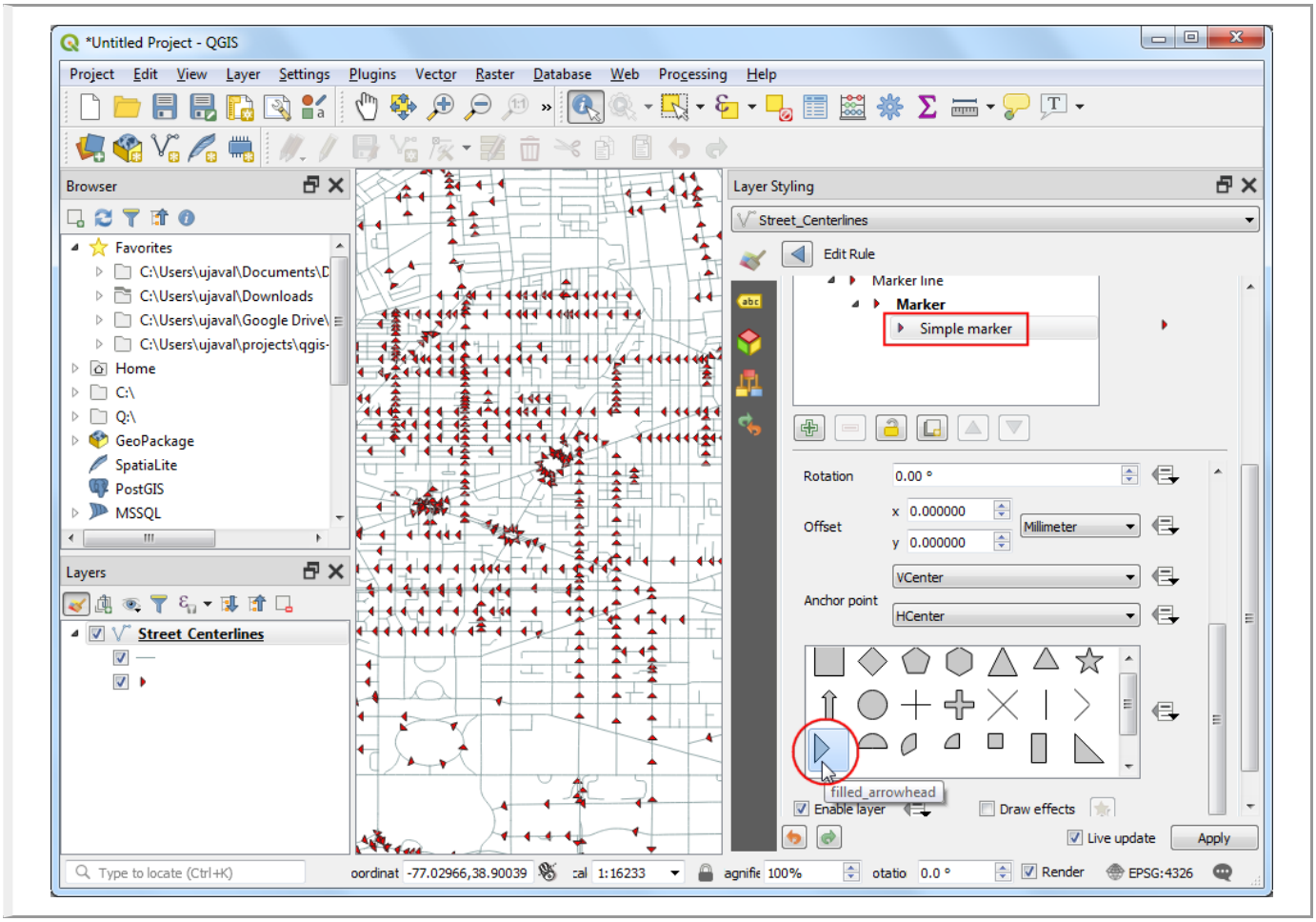
8. Next, change the Symbol layer type to `Marker line`.



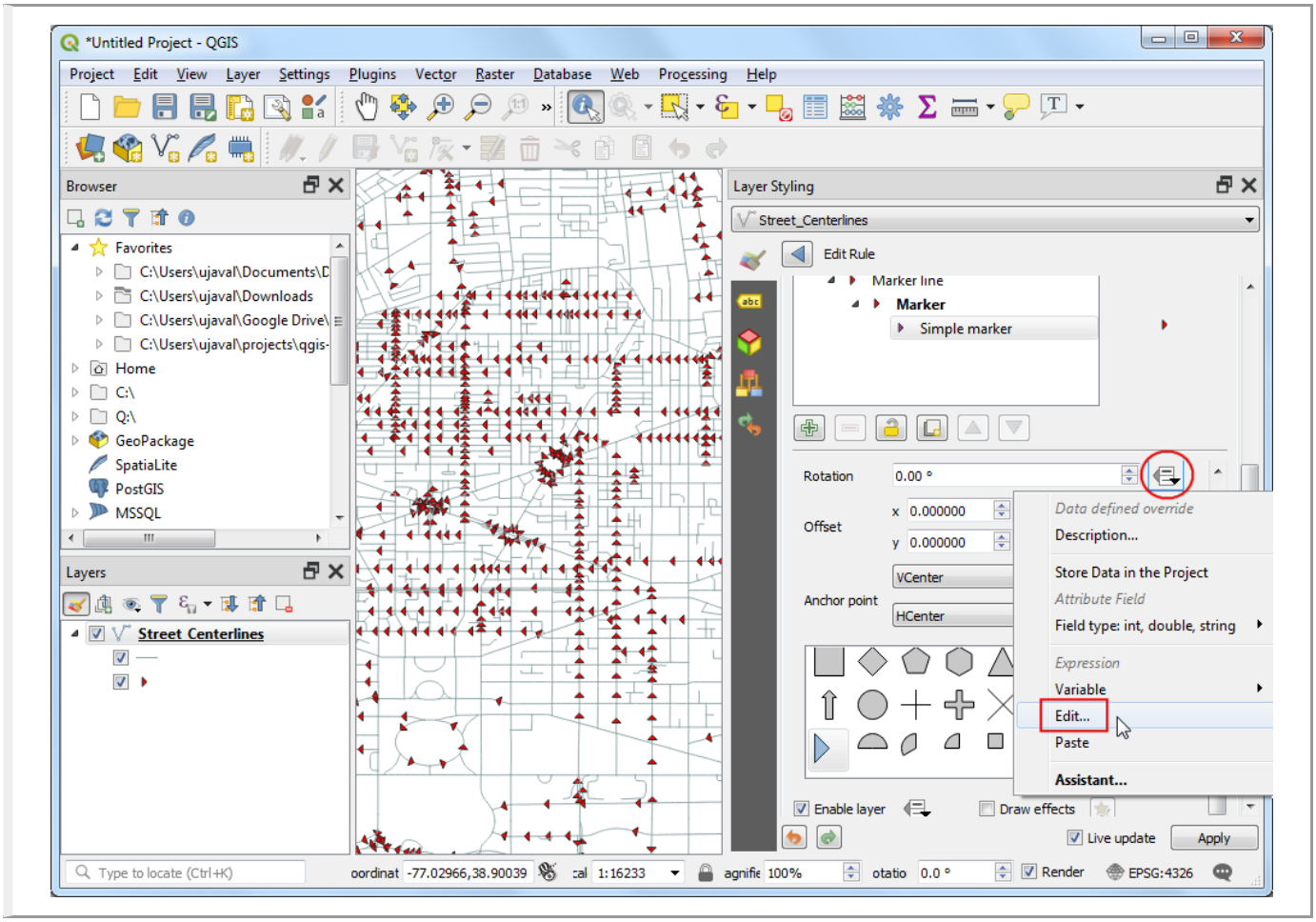
9. Select `on center point` under Marker placement.



10. Click on the `Simple marker` symbol. Scroll down and pick the `filled_arrowhead` marker. You will see that the arrow-like symbol now appears on the one-way streets. But all of them are pointing in a single direction, whereas we know that our filter contains roads in multiple directions. We can further refine the symbols with a data-defined override for the Rotation value.

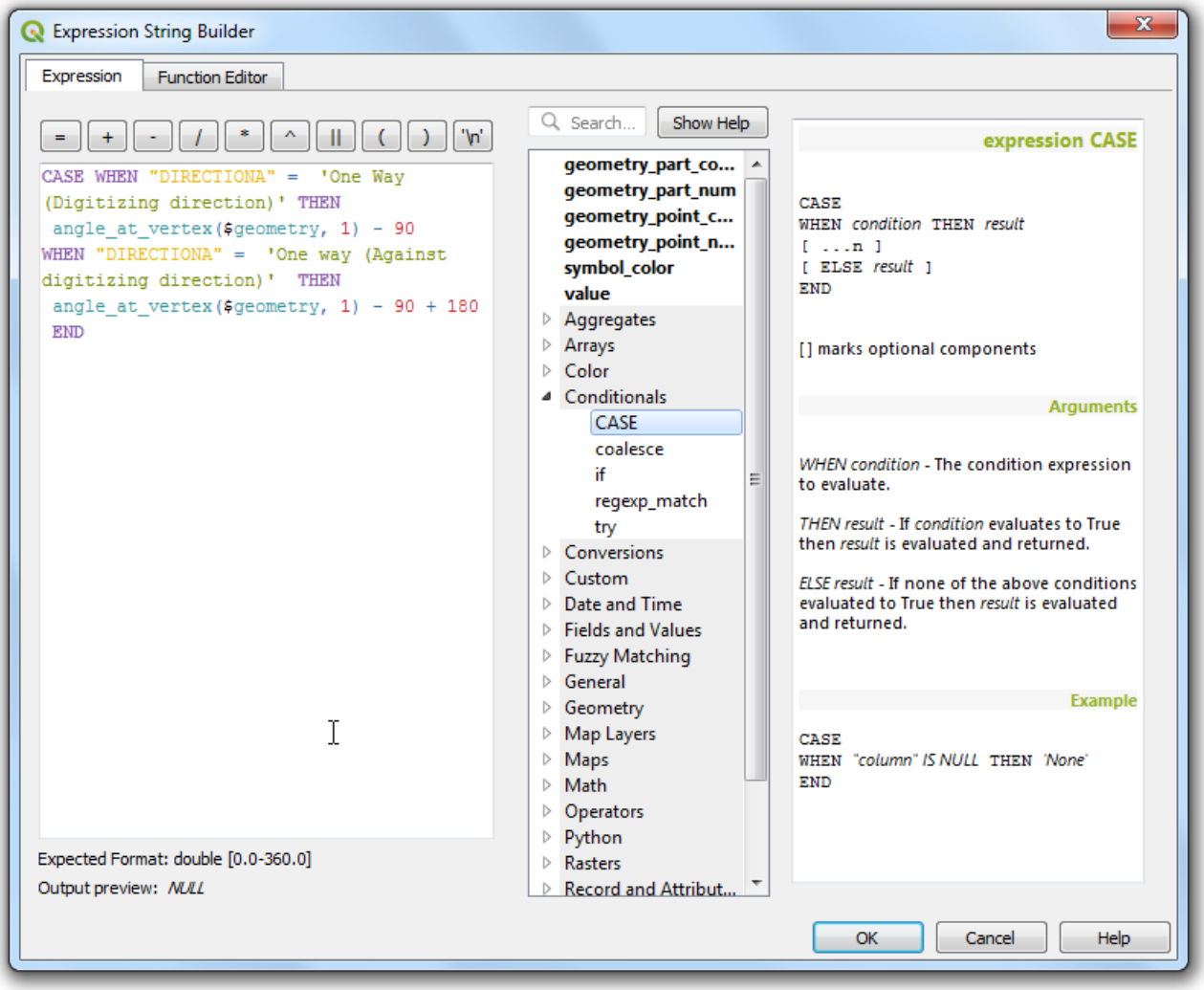


11. Click the Data defined override button next to Rotation.

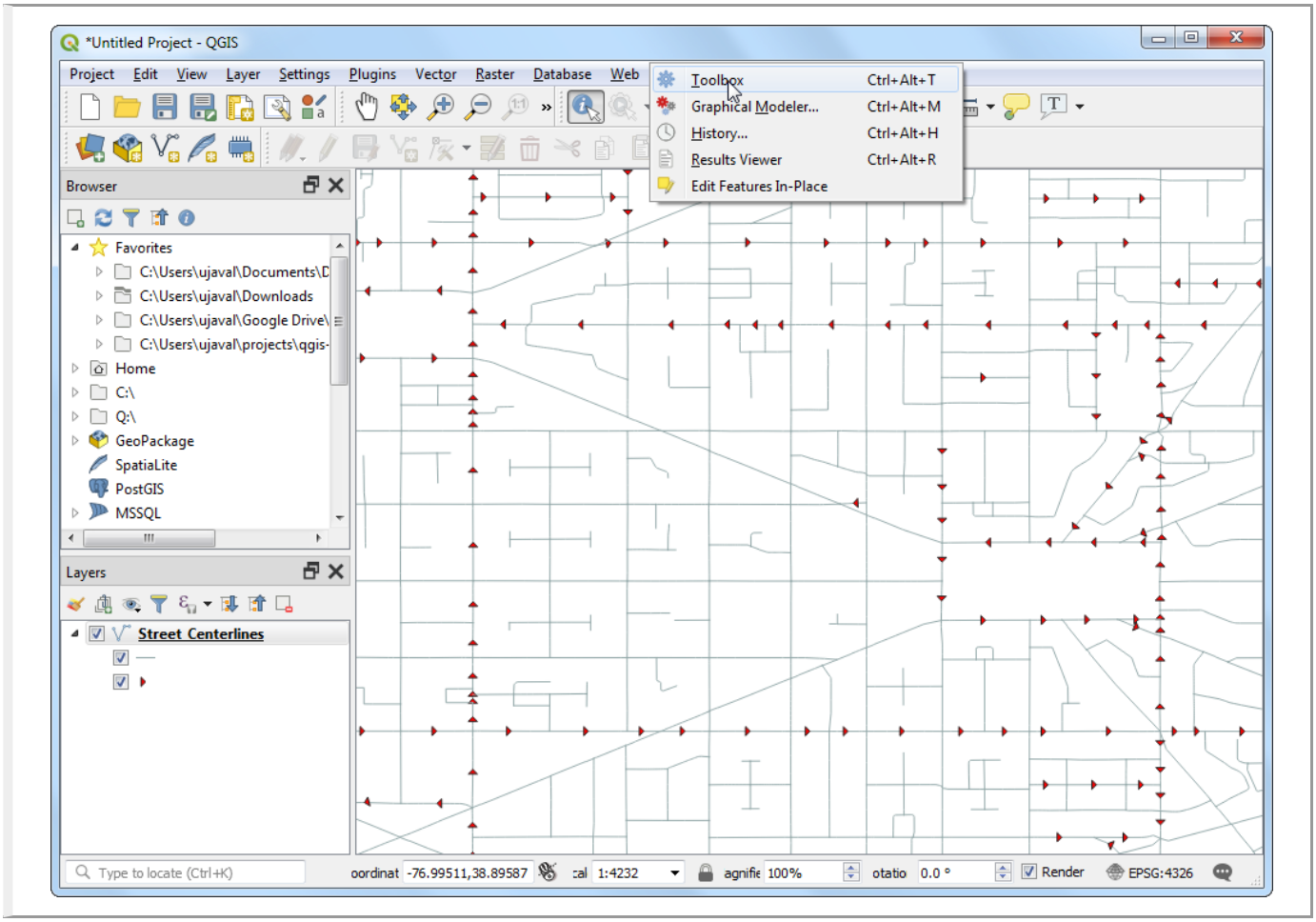


12. We can put a conditional expression that returns different rotation value depending on the one-way direction. A simple 0 or 180 degree rotation works to account for the direction, but it works only for horizontal lines. To align the arrow-head perpendicular to all the lines - we need to account for the angle of the line in the expression as well. The `angle_at_vertex` function helps us find the angle and use it in the expression. Enter the following expression and click OK.

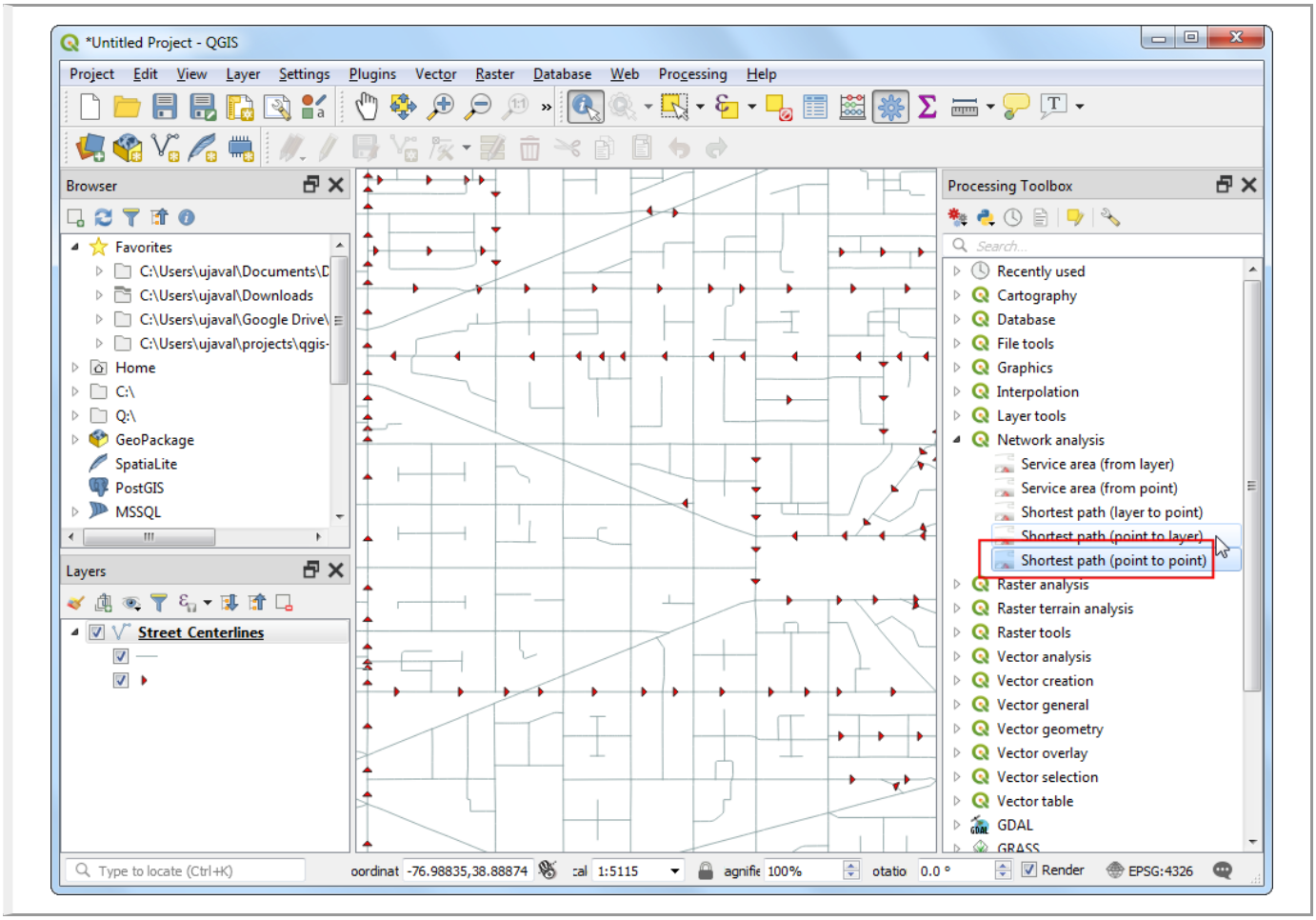
```
CASE
  WHEN "DIRECTIONA" =  'One Way (Digitizing direction)'
    THEN angle_at_vertex($geometry, 1) - 90
  WHEN "DIRECTIONA" =  'One way (Against digitizing direction)'
    THEN angle_at_vertex($geometry, 1) - 90 + 180
END
```



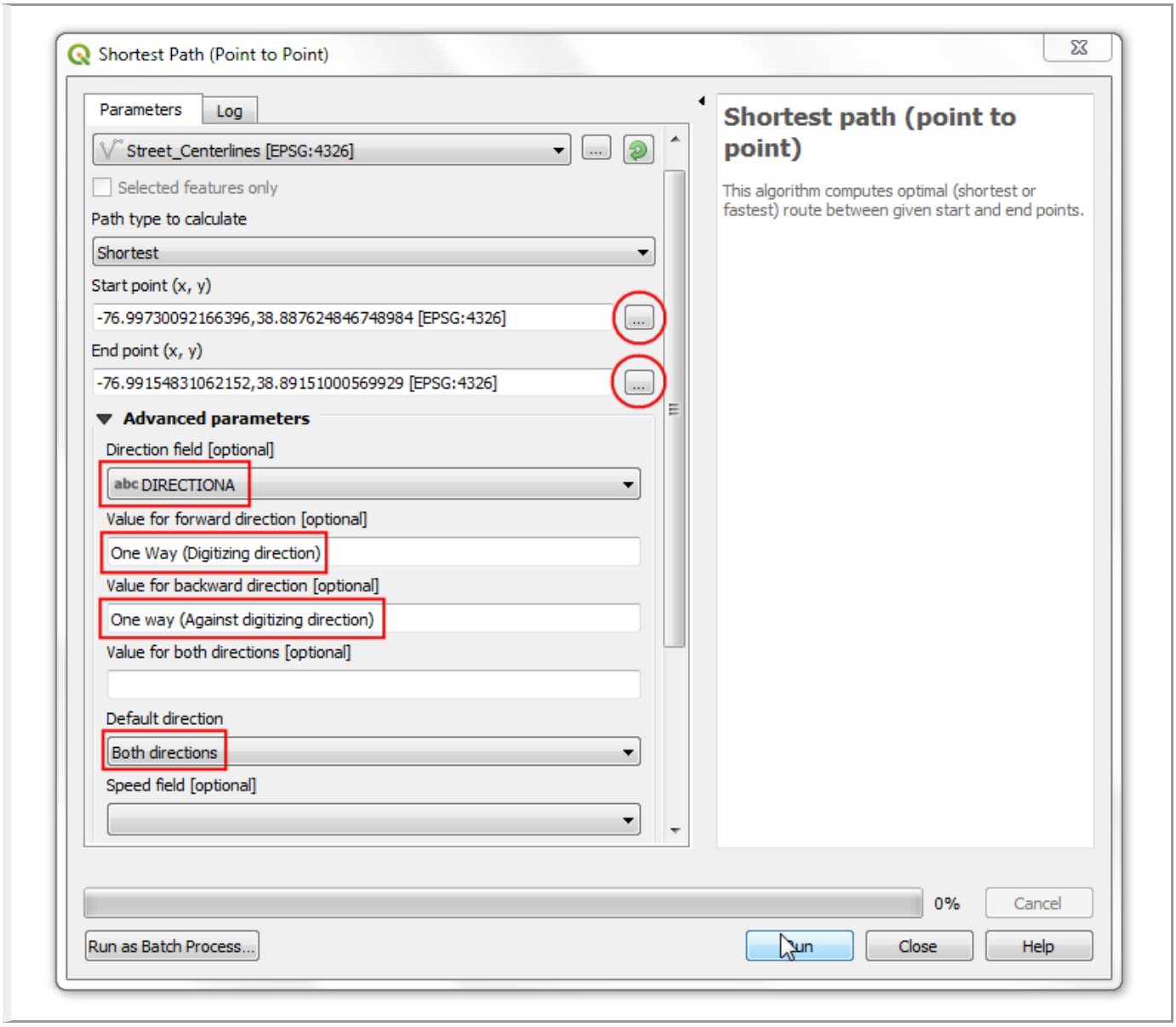
13. Now you will see the arrow-heads aligned to the correct road direction and angle. To keep the style uncluttered, we are choosing to display arrows only on one-way streets. Unlabeled streets are assumed to be two-way. Now that we have the network styled correctly, we can do some analysis. Go to Processing ‣ Toolbox.



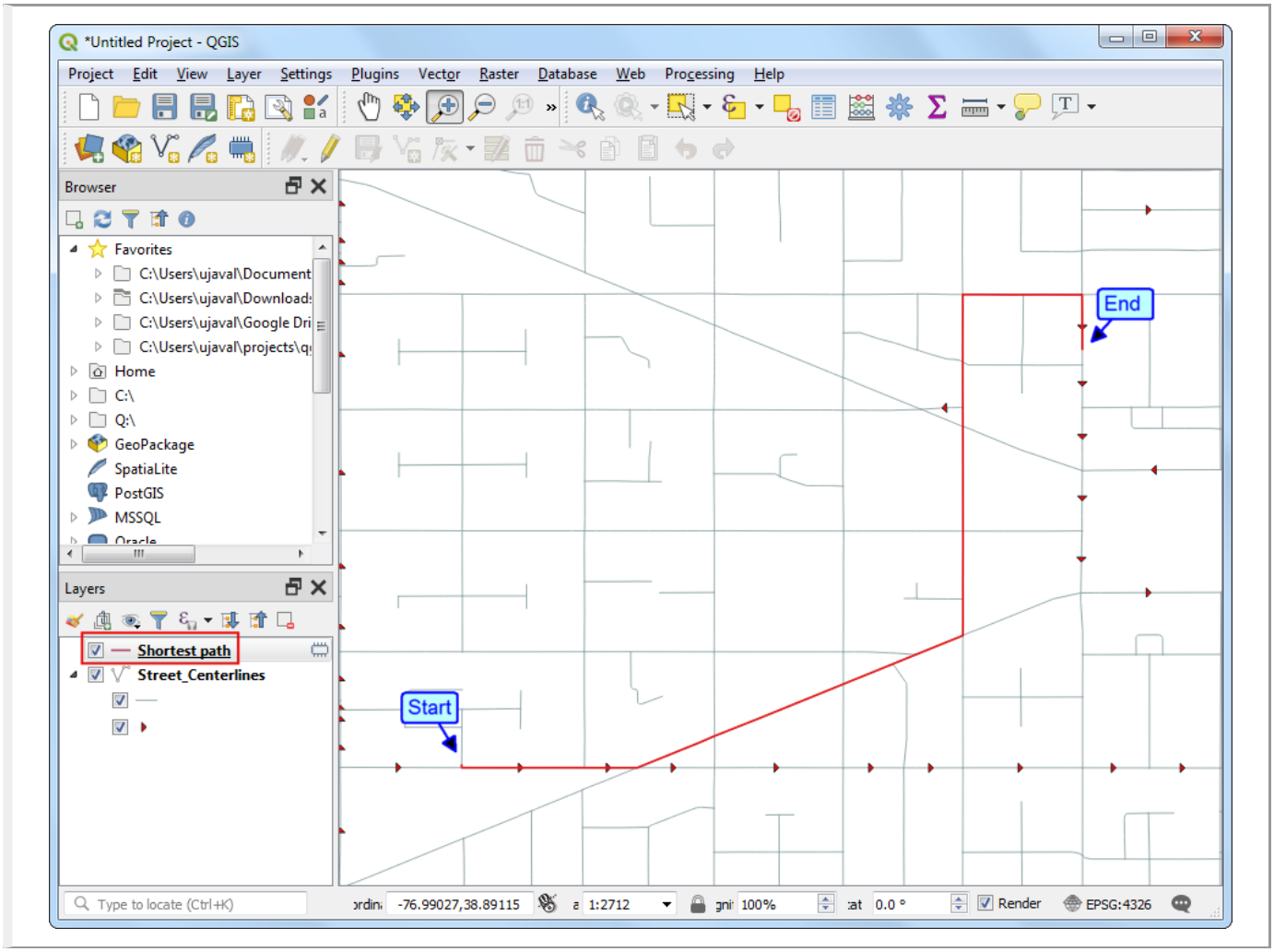
14. Search for and locate the Network analysis ▸ Shortest path (point to point) algorithm. Double-click to launch it.



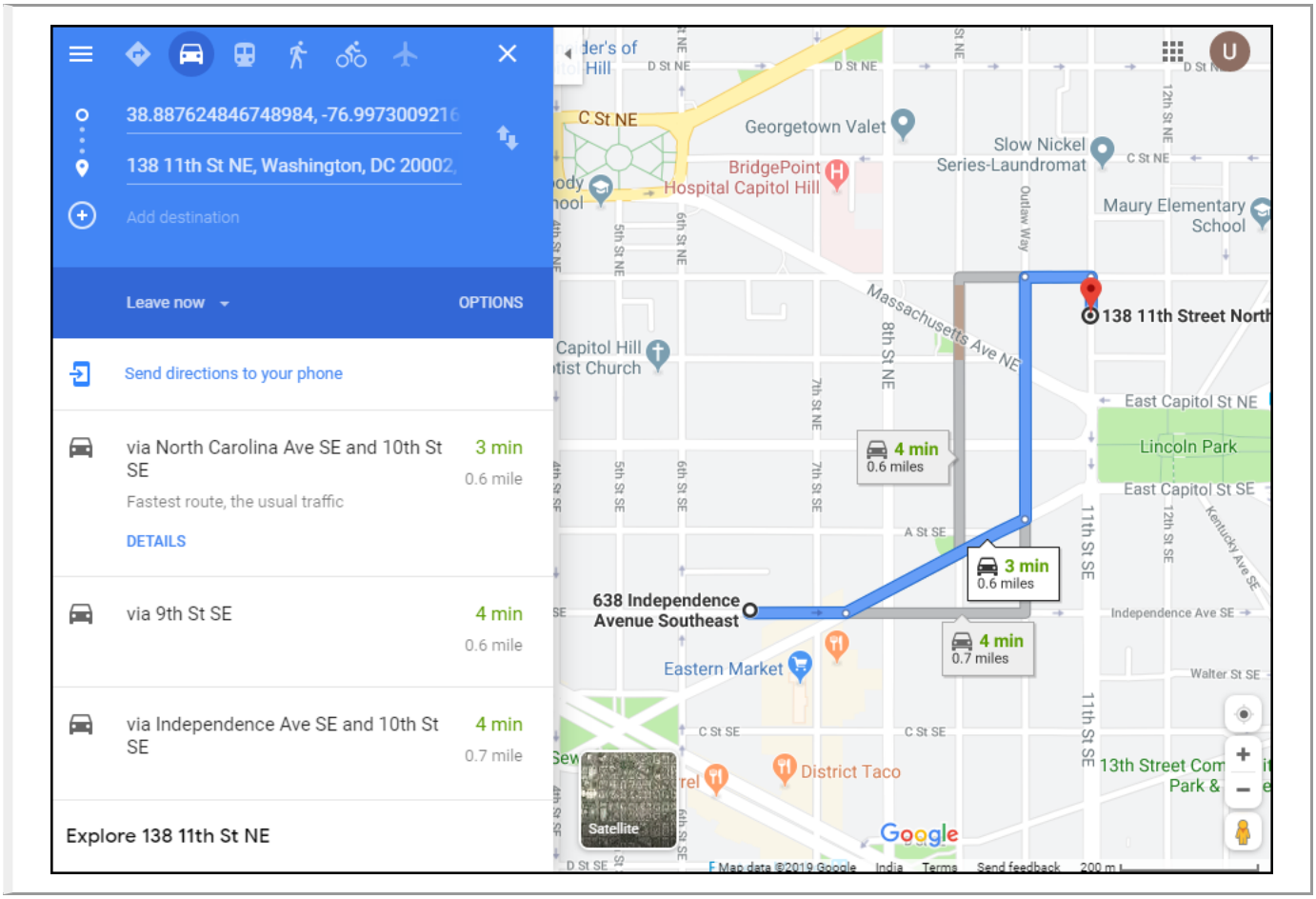
15. In the Shortest Path (Point to Point) dialog, select `Street_Centerlines` as the Vector layer representing network. Keep the Path type to calculate as `Shortest` . Next we need to pick a start and end point. You can click the ... button and click on any point on the network in the canvas. If you want to replicate the results in this tutorial, you can enter `-76.99730092166396,38.887624846748984` as the Start point and `-76.99154831062152,38.89151000569929` as the End point. Expand the Advanced parameter section. Choose `DIRECTIONA` as the Direction field. You must be familiar with the one-way direction values for forward and backward traffic flow. Enter `One Way (Digitizing direction)` as the Value for forward direction and `One way (Against digitizing direction)` as the Value for backward direction. Keep other options to their default values and click Run.



16. The algorithm will use the geometry of the layer and provided parameters to build a network graph. This graph is then used to find the shortest path between the start and end points. Once the algorithm finishes, you will see a new layer `Shortest path` added to the Layers panel that shows the shortest path between start and end points.



17. You will see that there are many possible paths between start and end points. But given the constraints of the network - such as one-ways, the result is the shortest possible path. It is always a good idea to validate your analysis and assumptions. One easy way to validate it is to use a third-party mapping service to see if their results match with the ones we derived. Here is the shortest path suggested by Google Maps (https://goo.gl/maps/XwTXTkvuaCuteocr8) between the same start and end points. As you can see the recommended shortest route matches exactly with our results - validating our analysis.



Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Locating Nearest Facility with Origin-Destination Matrix (QGIS3)

In the previous tutorial, *Basic Network Visualization and Routing (QGIS3)* (basic_network_analysis.html), we learnt how to build a network and calculate the shortest path between 2 points. We can apply that technique for many different types of network-based analysis. One such application is to compute **Origin-Destination Matrix** or **OD Matrix**. Given a set of origin points and another set of destination points, we can calculate shortest path between each origin-destination pairs and find out the travel distance/time between them. Such analysis is useful to locate the closest facility to any given point. For example, a logistics company may use this analysis to find the closest warehouse to their customers to optimize delivery routes. Here we use Distance Matrix algorithm from **QGIS Network Analysis Toolbox (QNEAT3)** plugin to find the nearest health facility to each address in the city.

Note

This tutorial shows how to use your own network data to compute an origin-destination matrix. If you do not have your own network data, you can use **ORS Tools Plugin** and algorithm ORS Tools ‣ Matrix ‣ Matrix from Layers to do the similar analysis using OpenStreetMap data. See *Service Area Analysis using Openrouteservice (QGIS3)* (service_area_analysis.html) to learn how to use ORS Tools plugin.

Overview of the task

We will take 2 layers for Washington DC - one with points representing addresses and another with points representing mental health facilities - and find out the facility with the least travel distance from each address.

Other skills you will learn

- Extract a stratified random sample from a point layer.
- Use Virtual Layers to run SQL query on a QGIS layer.
- Use Python Console Editor to run a pyqgis script.

Get the data

District of Columbia government freely shares hundreds of datasets on the Open Data Catalog (https://opendata.dc.gov/).

Download the following data layers as shapefiles.

- Street Centerlines (https://opendata.dc.gov/datasets/street-centerlines)
- Address Points (https://opendata.dc.gov/datasets/address-points)
- Adult Mental Health Providers (https://opendata.dc.gov/datasets/adult-mental-health-providers)

For convenience, you may directly download a copy of the datasets from the links below:

Street_Centerlines.zip (http://www.qgistutorials.com/downloads/Street_Centerlines.zip)

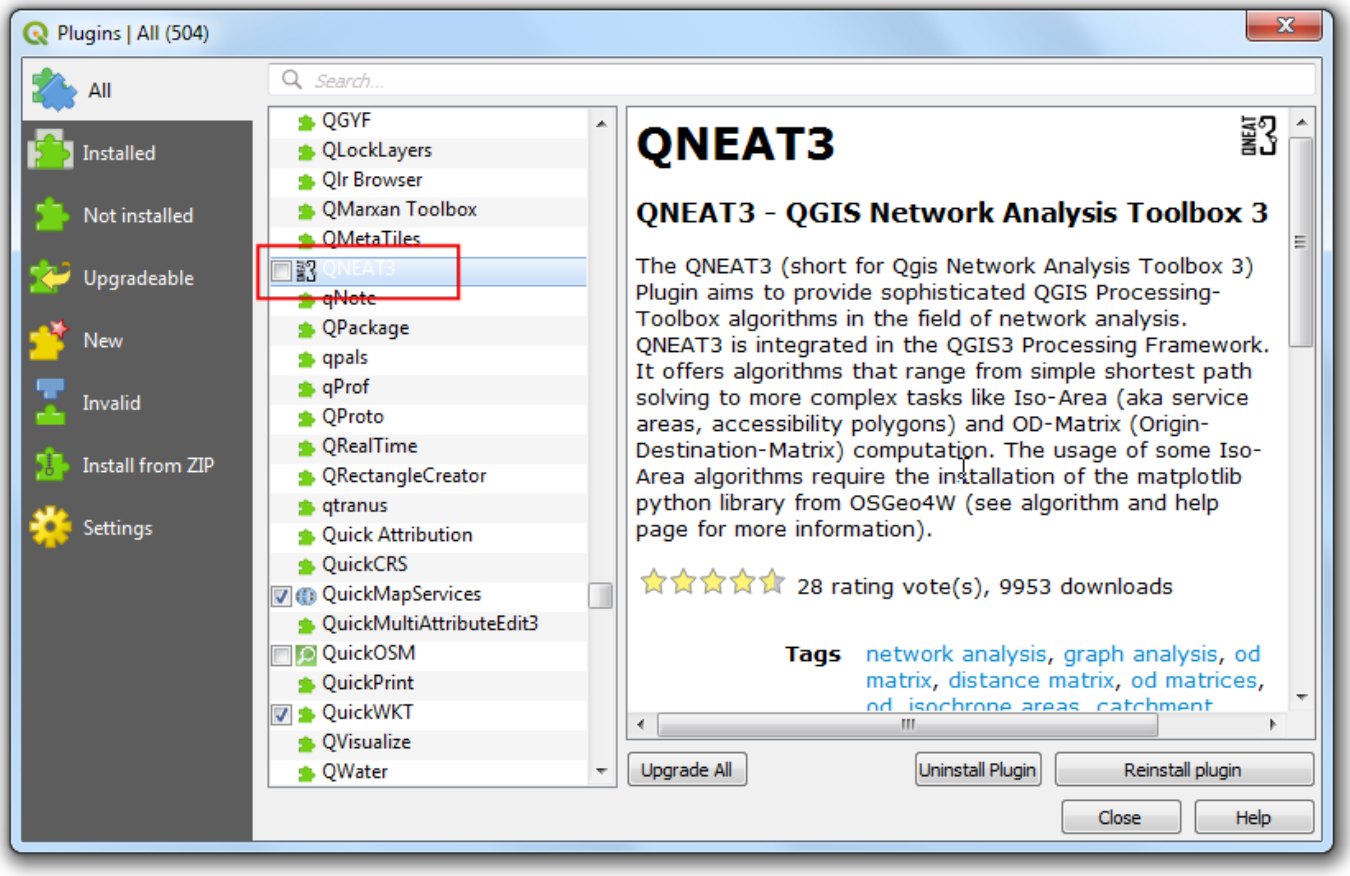
Address_Points.zip (http://www.qgistutorials.com/downloads/Address_Points.zip)

Adult_Mental_Health_Providers.zip
(http://www.qgistutorials.com/downloads/Adult_Mental_Health_Providers.zip)

Data Source: [DCOPENDATA] (../credits.html#dcopendata)

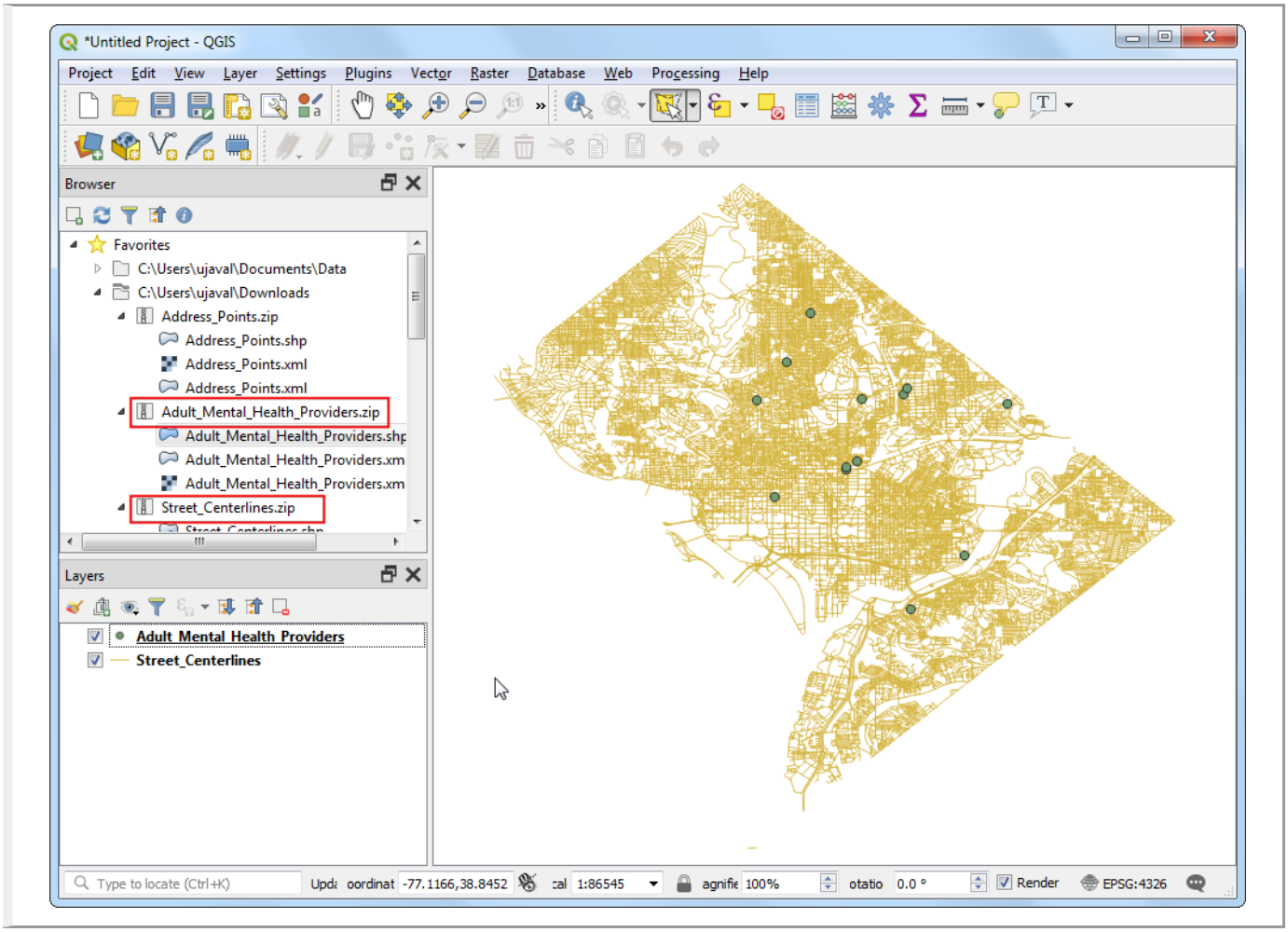
Setup

Visit Plugins ‣ Manage and Install plugins. Search for **QNEAT3** plugin and install it. Click Close.

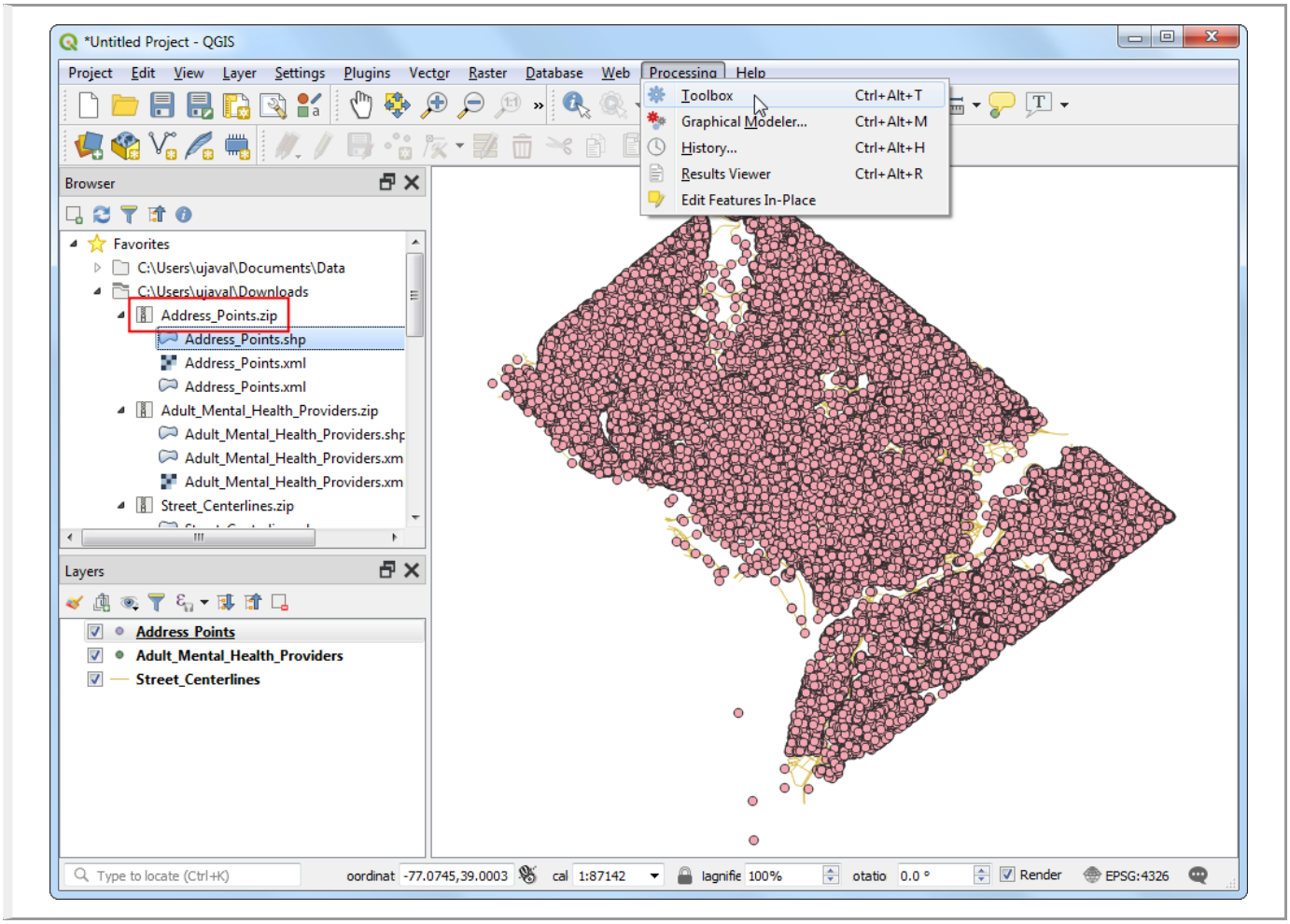


Procedure

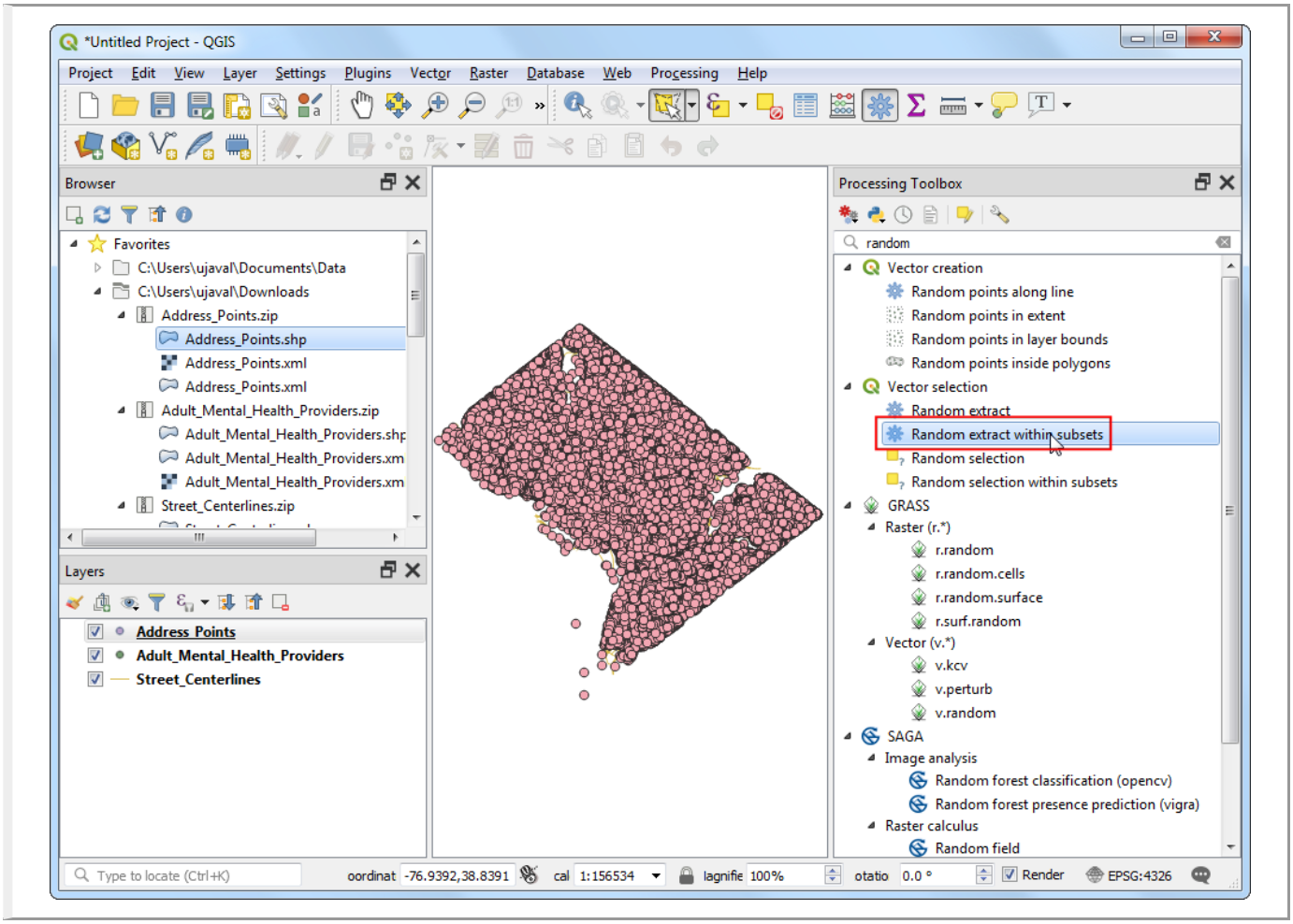
1. Locate the downloaded `Street_Centerlines.zip` file in the Browser panel. Expand it and drag the `Street_Centerlines.shp` file to the canvas. Similarly, locate the `Adult_Mental_Health_Providers.zip` file, expand it and add `Adult_Mental_Health_Providers.shp` to the canvas.



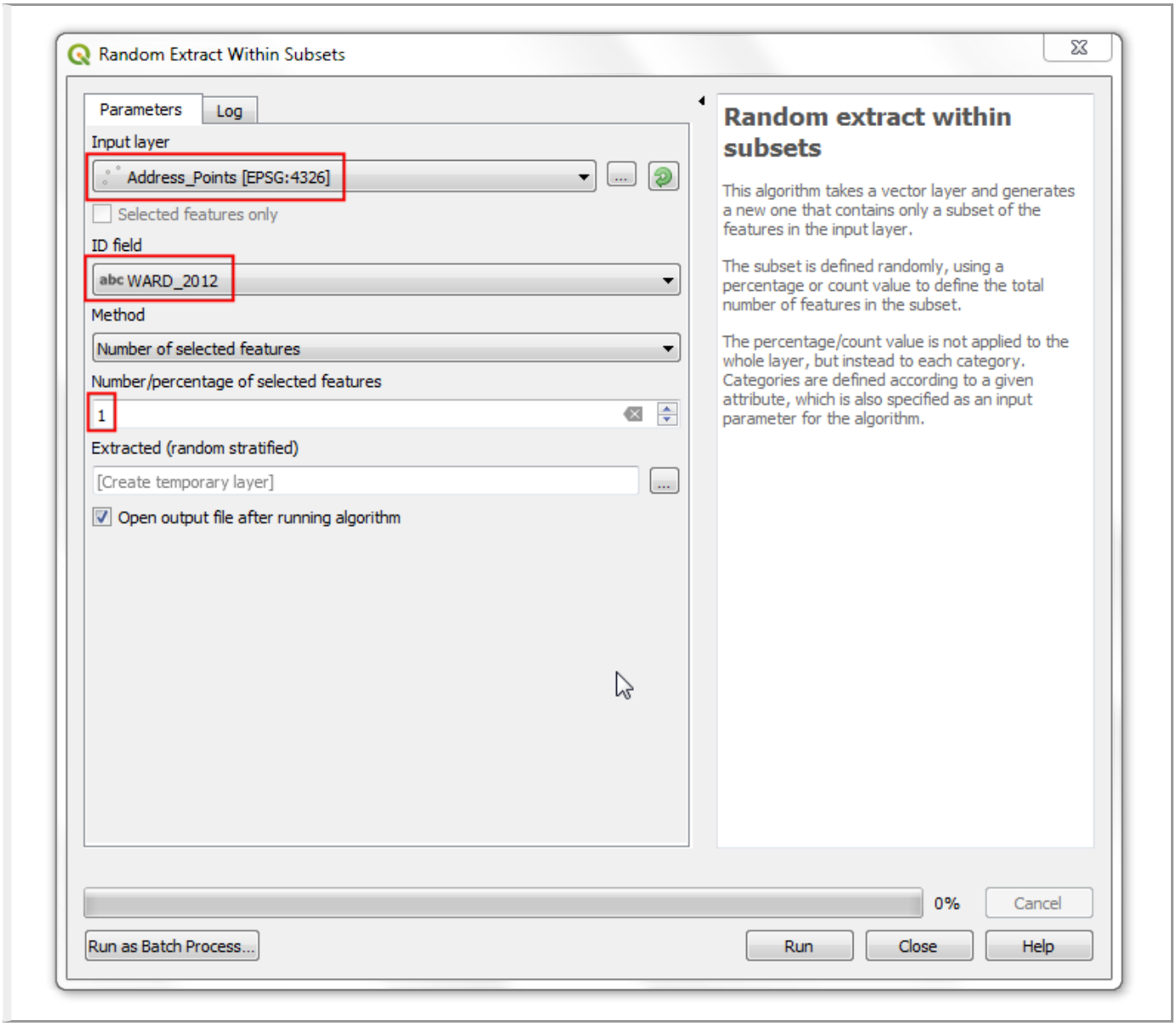
2. Next, locate the `Address_Points.zip` file, expand it and add the `Address_Points.shp`. You will see a lot of points around the city. Each point represents a valid address. We will not randomly select 1 point in each ward to use as the origin points. This technique is called stratified sampling. Go to Processing ▸ Toolbox.



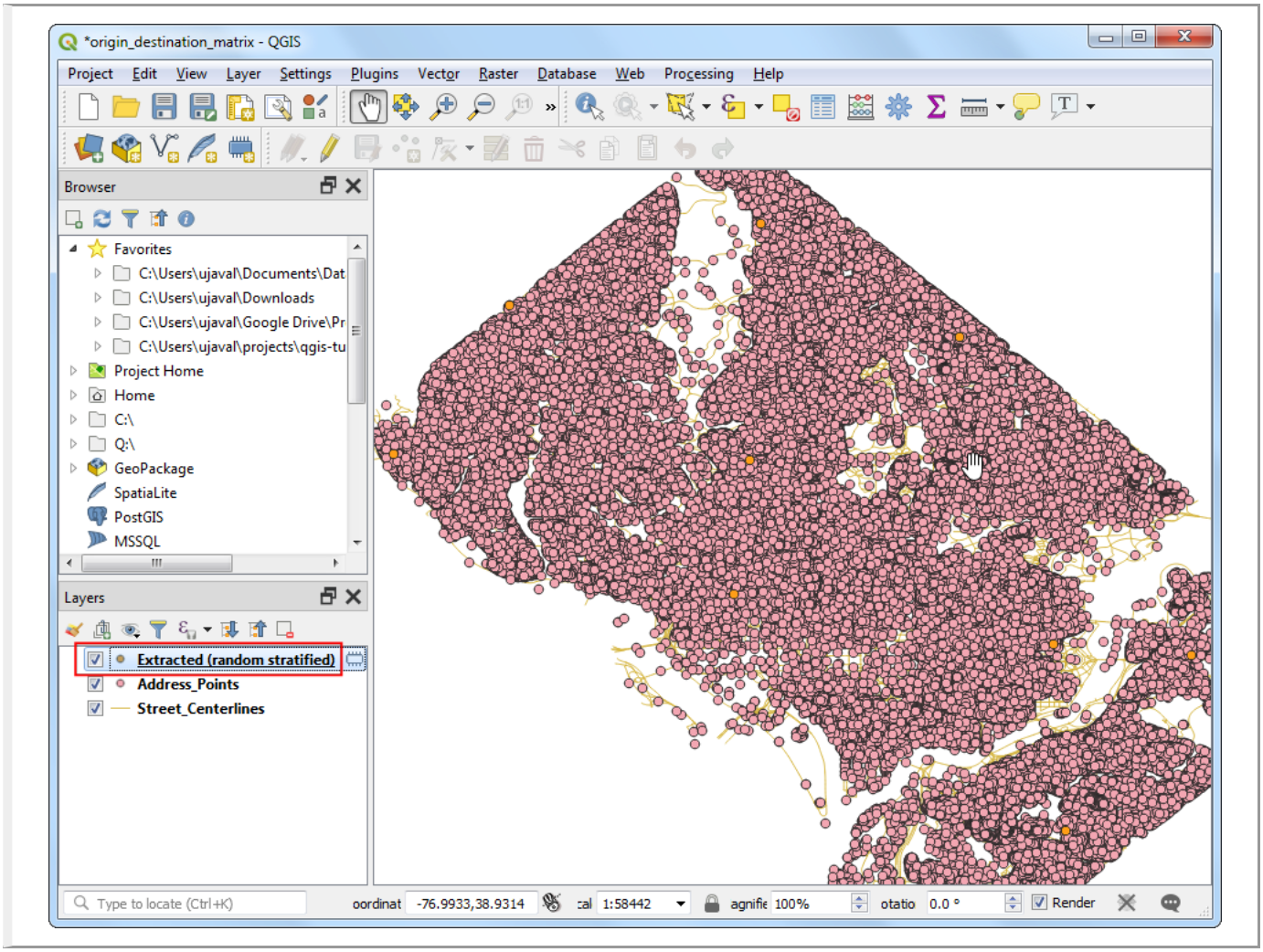
3. Search for and locate the Vector Selection ▸ Random extract within subsets algorithm.



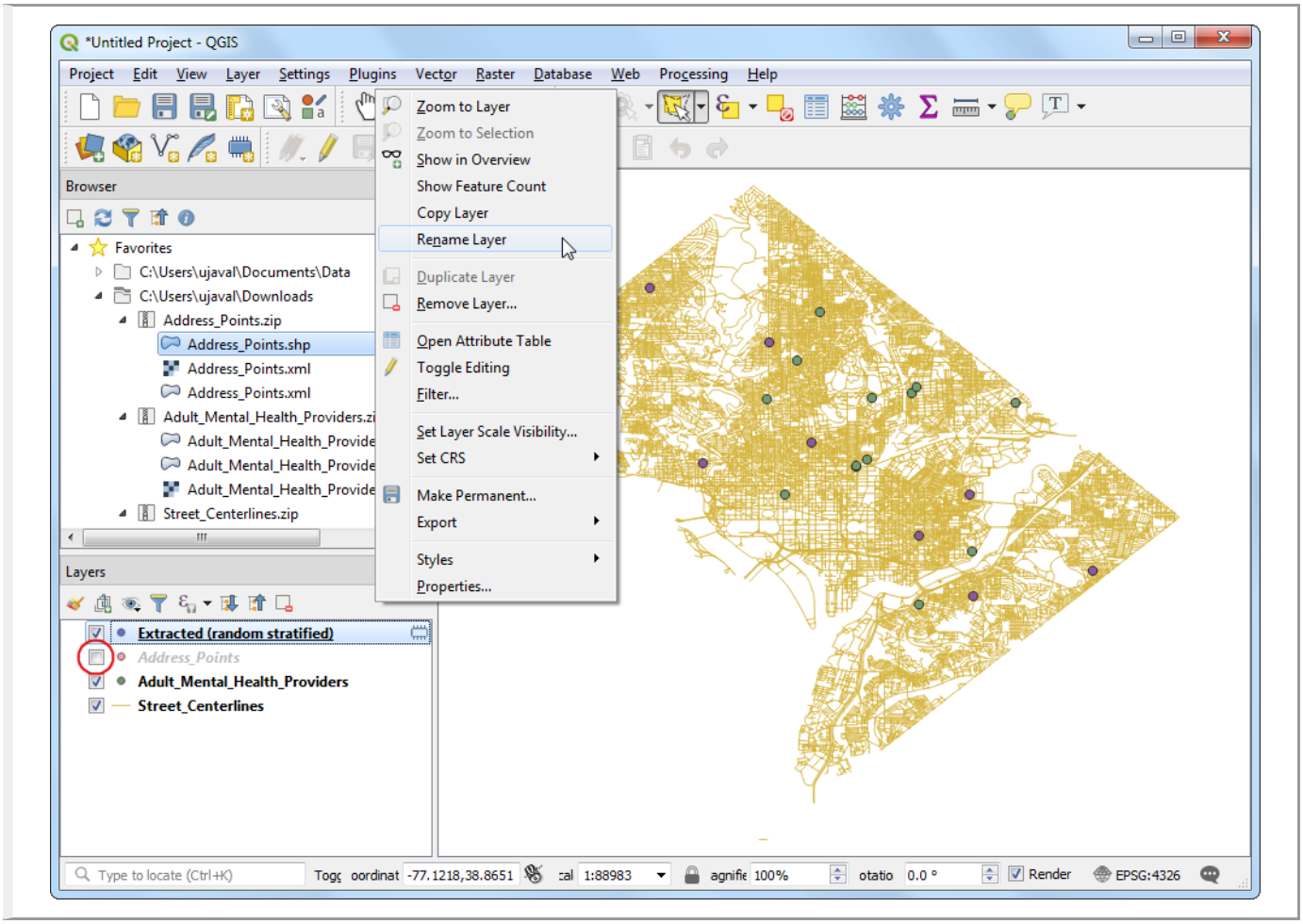
4. Select `Address_Points` as the Input layer. Each address point contains an attribute called `WARD_2012` which has the ward number associated with the address. As we want only 1 point per ward, we use that attribute as the ID field. Set Number/percentage of selected features as `1` .



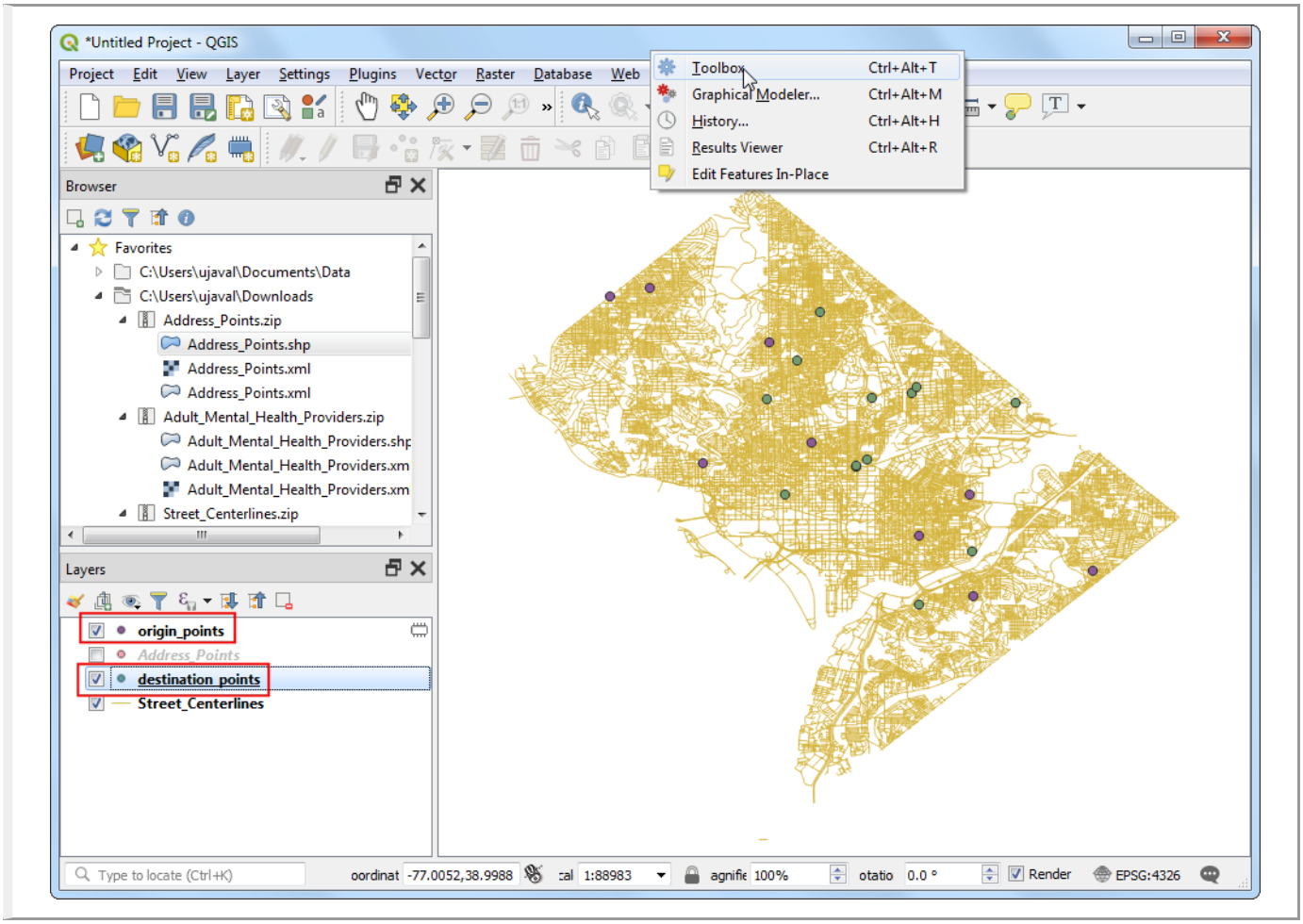
5. A new layer `Extracted (random stratified)` will be added to the Layers panel.



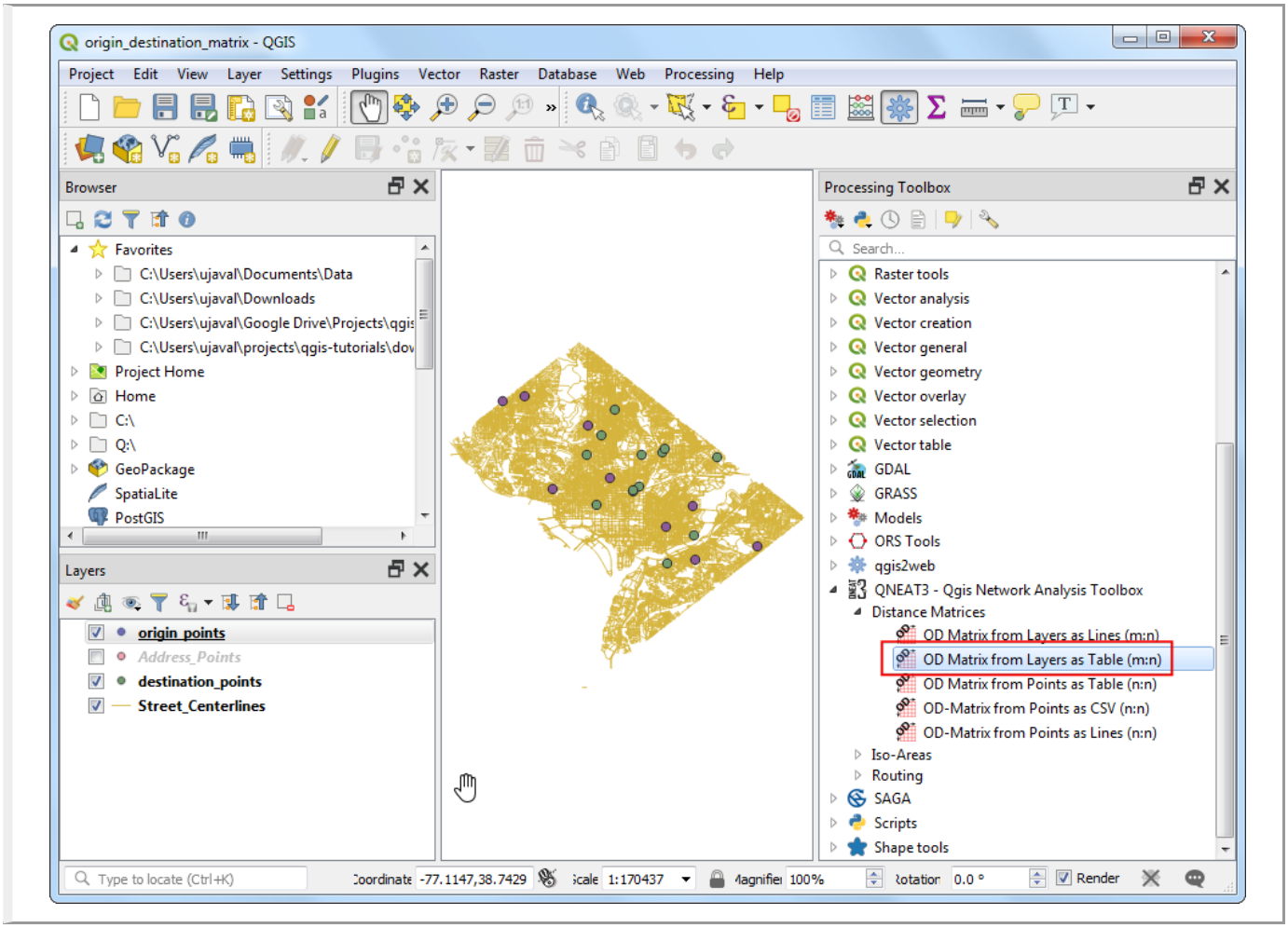
6. Turn-off the visibility for the `Address_Points` layer. Right-click on the `Extracted (random stratified)` layer and select Rename layer.



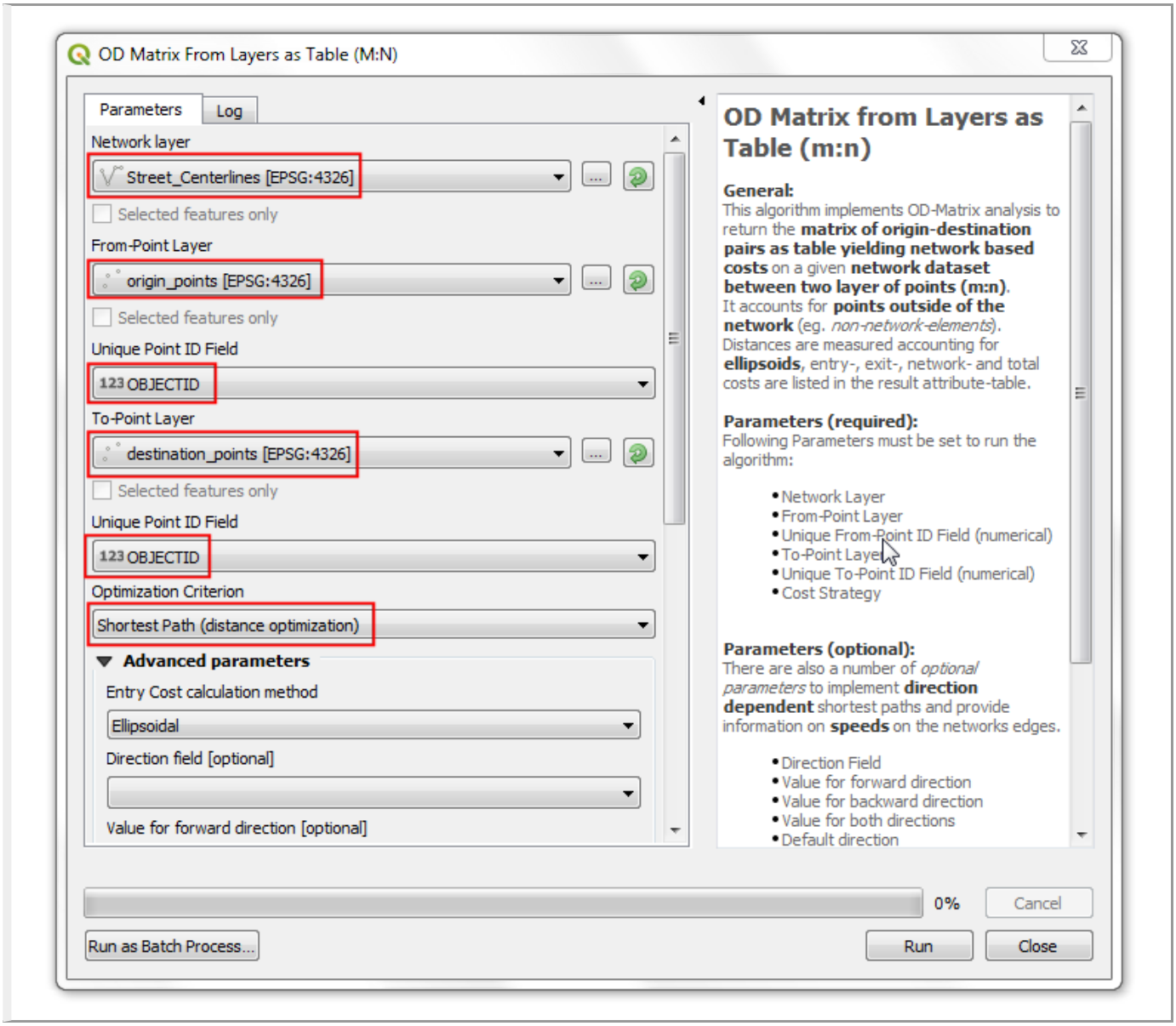
7. Let's rename this layer as `origin_points`. Similarly rename the `Adult_Mental_Health_Providers` layers representing the health facilities as `destination_points`. Naming the layers this way makes it easy to identify them in subsequent processing. Go to Processing ▸ Toolbox.



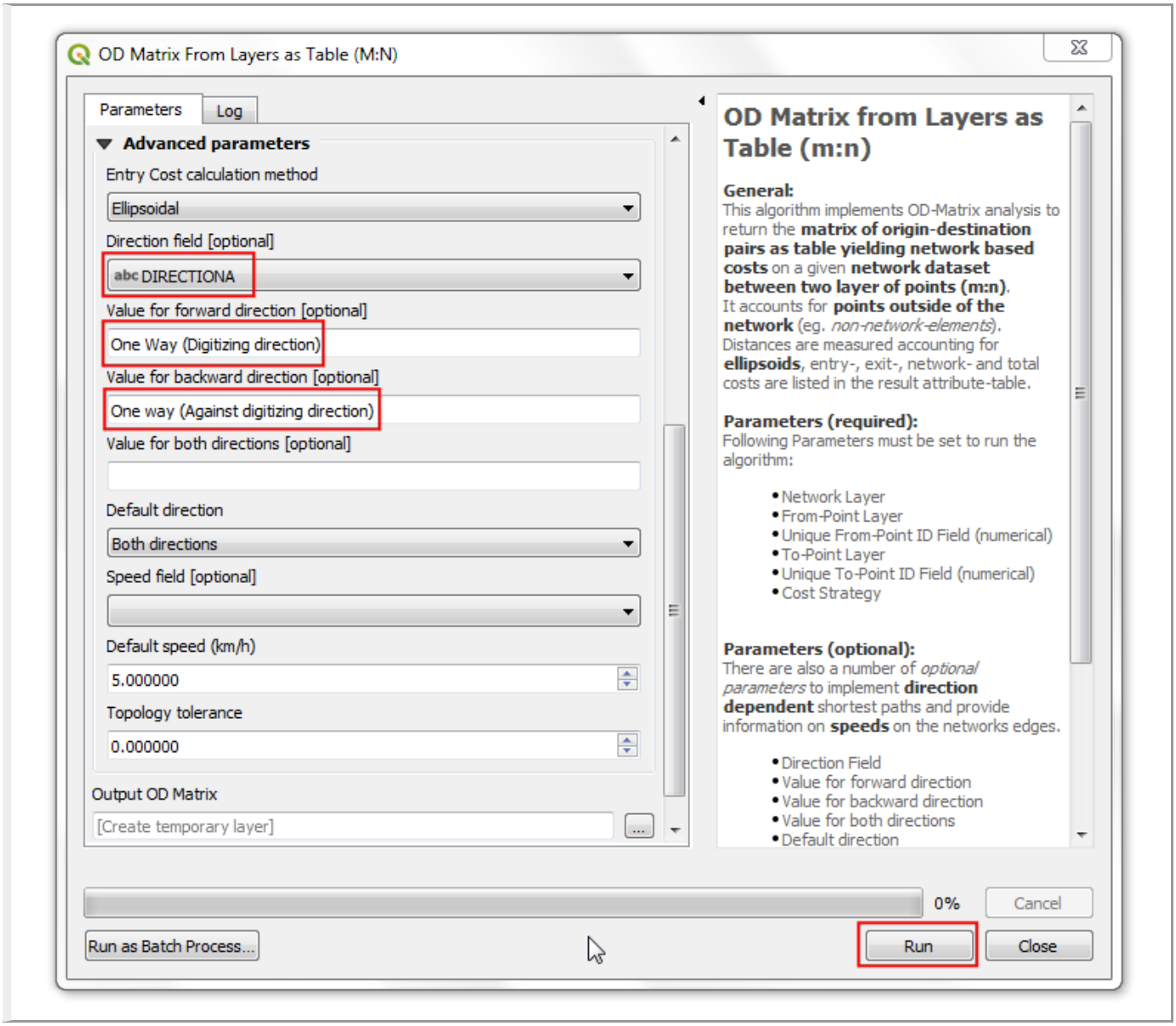
8. Locate the QNEAT3 ▸ Distance matrices ▸ OD Matrix from Layers as Table (m:n) algorithm. If you do not see this algorithm in the toolbox, make sure you have installed the **QNEAT3** plugin.



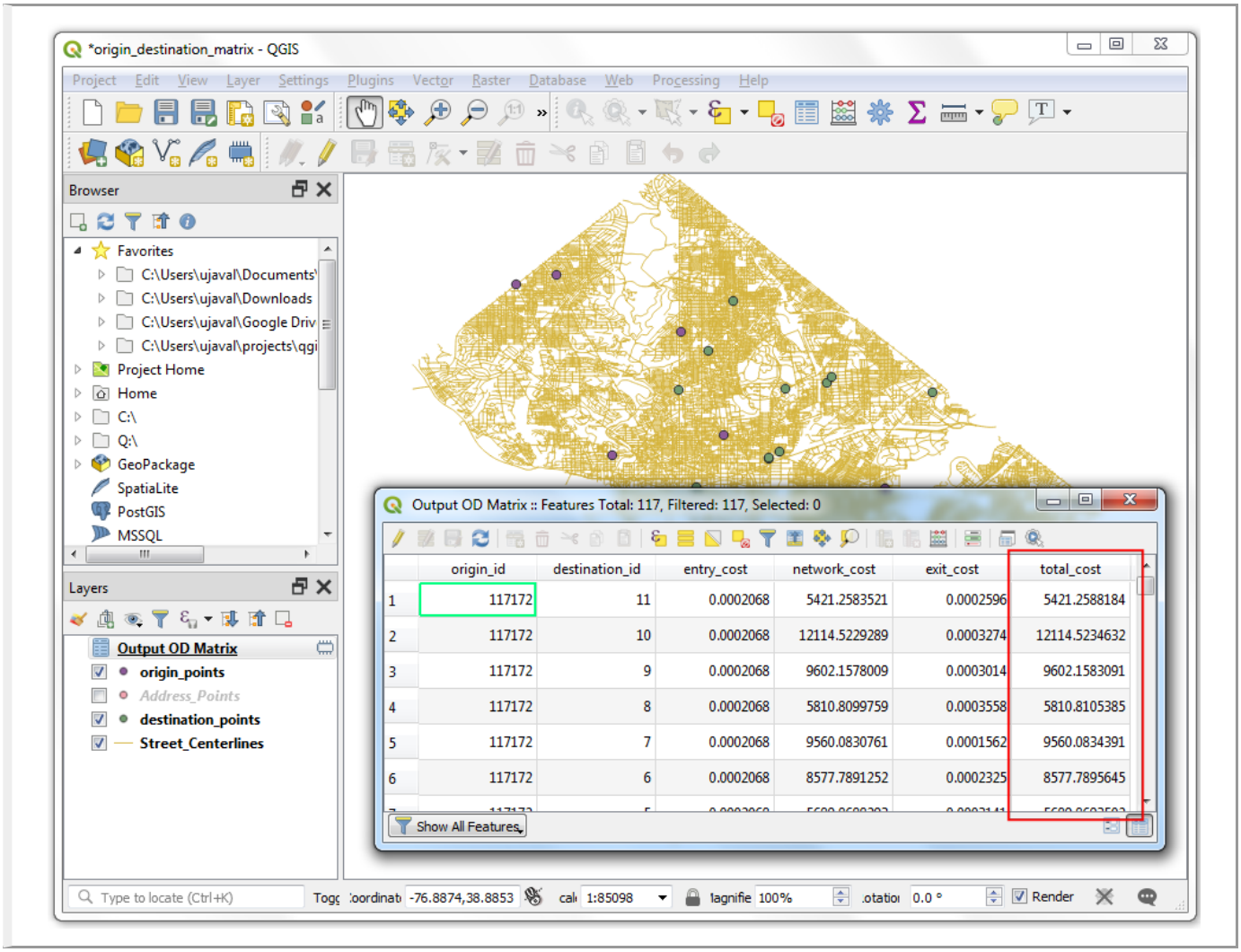
9. This algorithm helps find the distances along the network between selected origin and destination layers. Select `Street_Centerlines` as the Network layer. Select `origin_points` as the From-Points layer and `OBJECTID` as the Unique Point ID field. Similarly, set `destination_points` as the To-Points Layer and `OBJECTID` as the Unique Point ID field. Set the Optimization Criterion as `Shortest Path`.



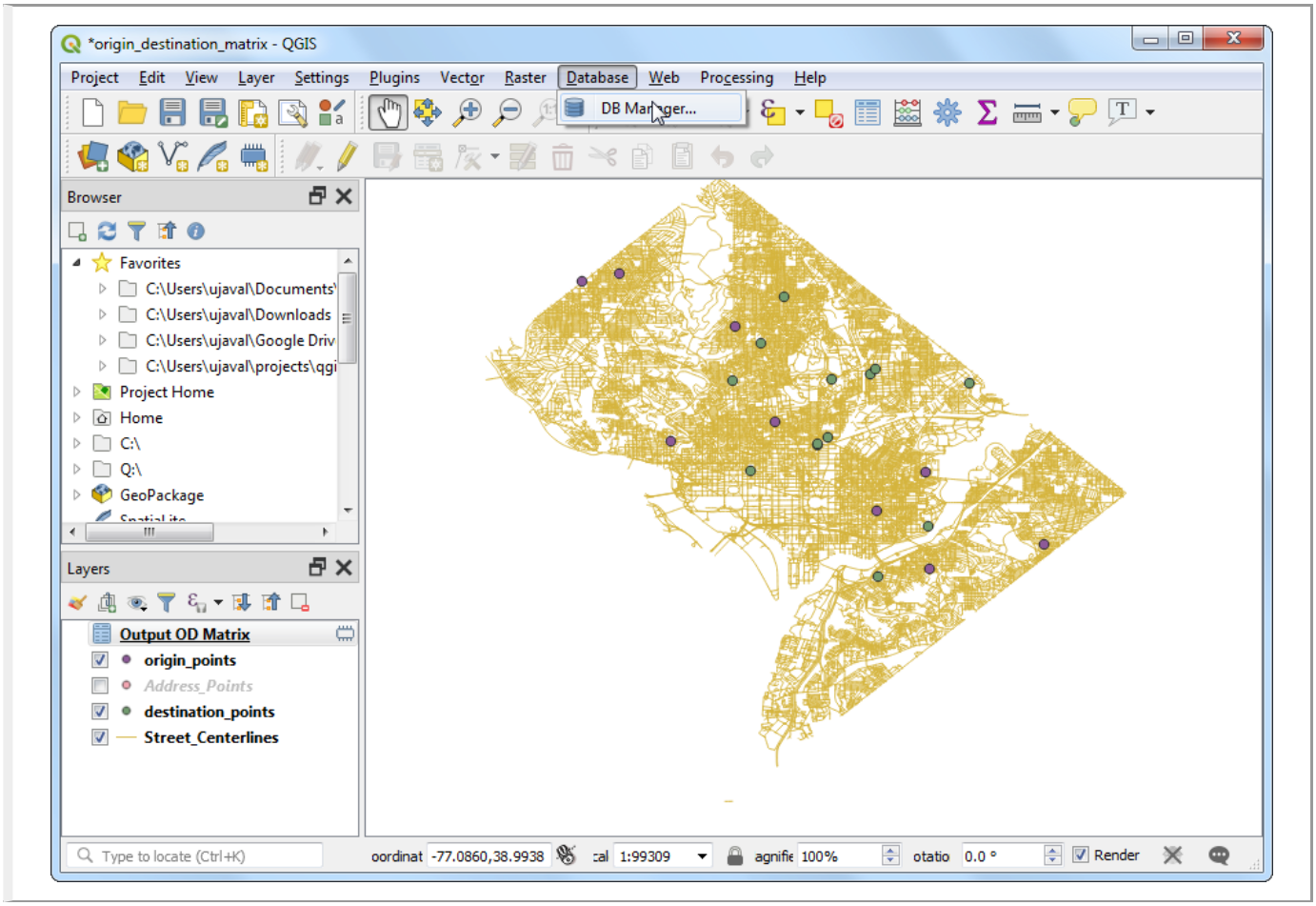
10. As many streets in the network are one-way, we need to set the Advanced parameters to specify the direction. See *Basic Network Visualization and Routing (QGIS3)* (basic_network_analysis.html) for more details on how these attributes are structured. Choose `DIRECTIONA` as the Direction field. Enter `One Way (Digitizing direction)` as the Value for forward direction and `One way (Against digitizing direction)` as the Value for backward direction. Keep other options to their default values and click Run.



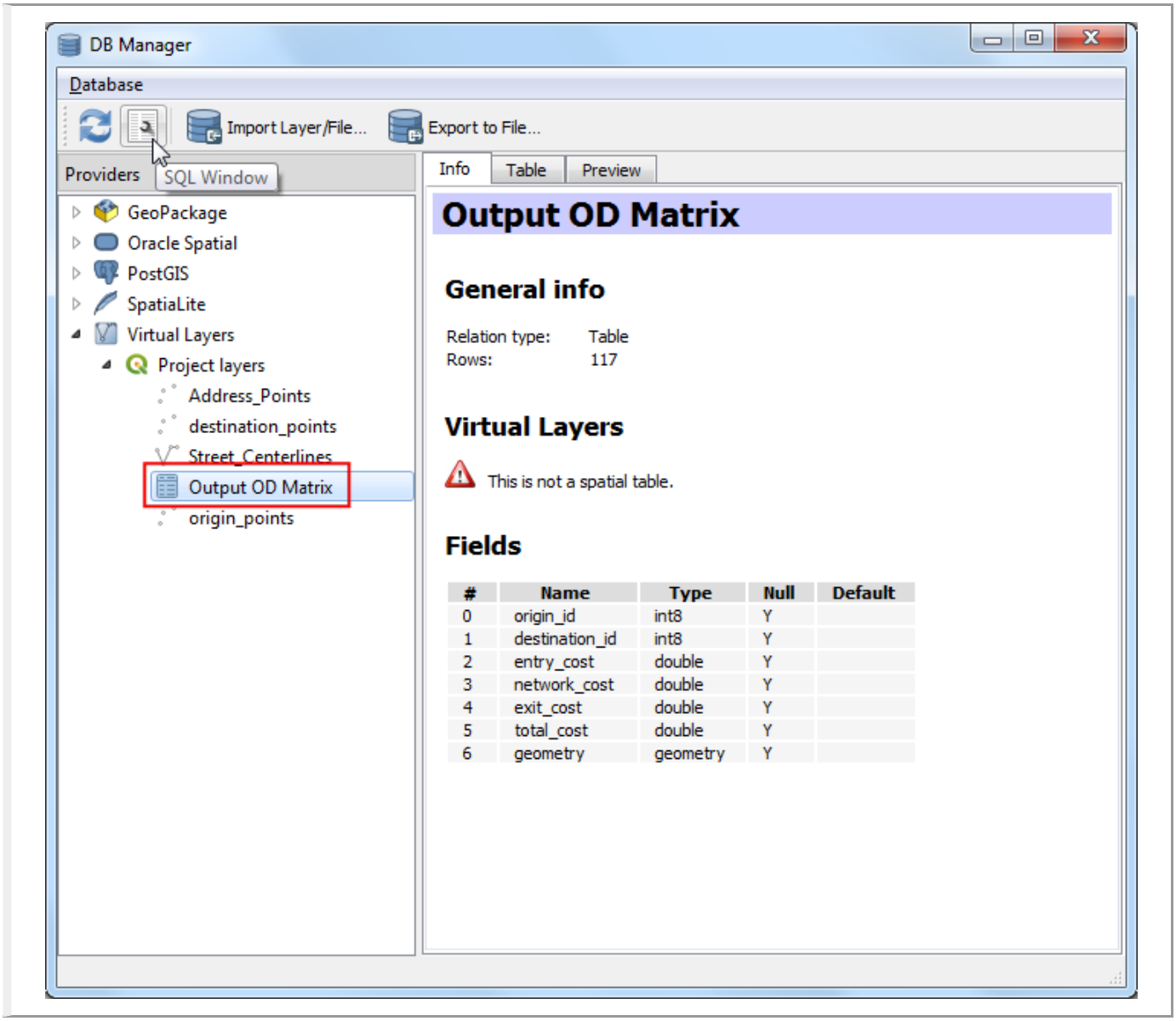
11. A new table layer called `Output OD Matrix` will be added to the Layers panel. Right-click and select Open Attributes Table. You will see that the table contains *117* rows. We had 9 origin points and 13 destination points - so the output contains *9x13 = 117* pairs of origins and destination. The `total_cost` column contains distance in meters between each origin point to every destination point.



12. For this tutorial, we are interested in only the destination point with the shortest distance. We can create a SQL query to pick the destination with the least `total_cost` among all destinations. Go to Database ▸ DB Manager..

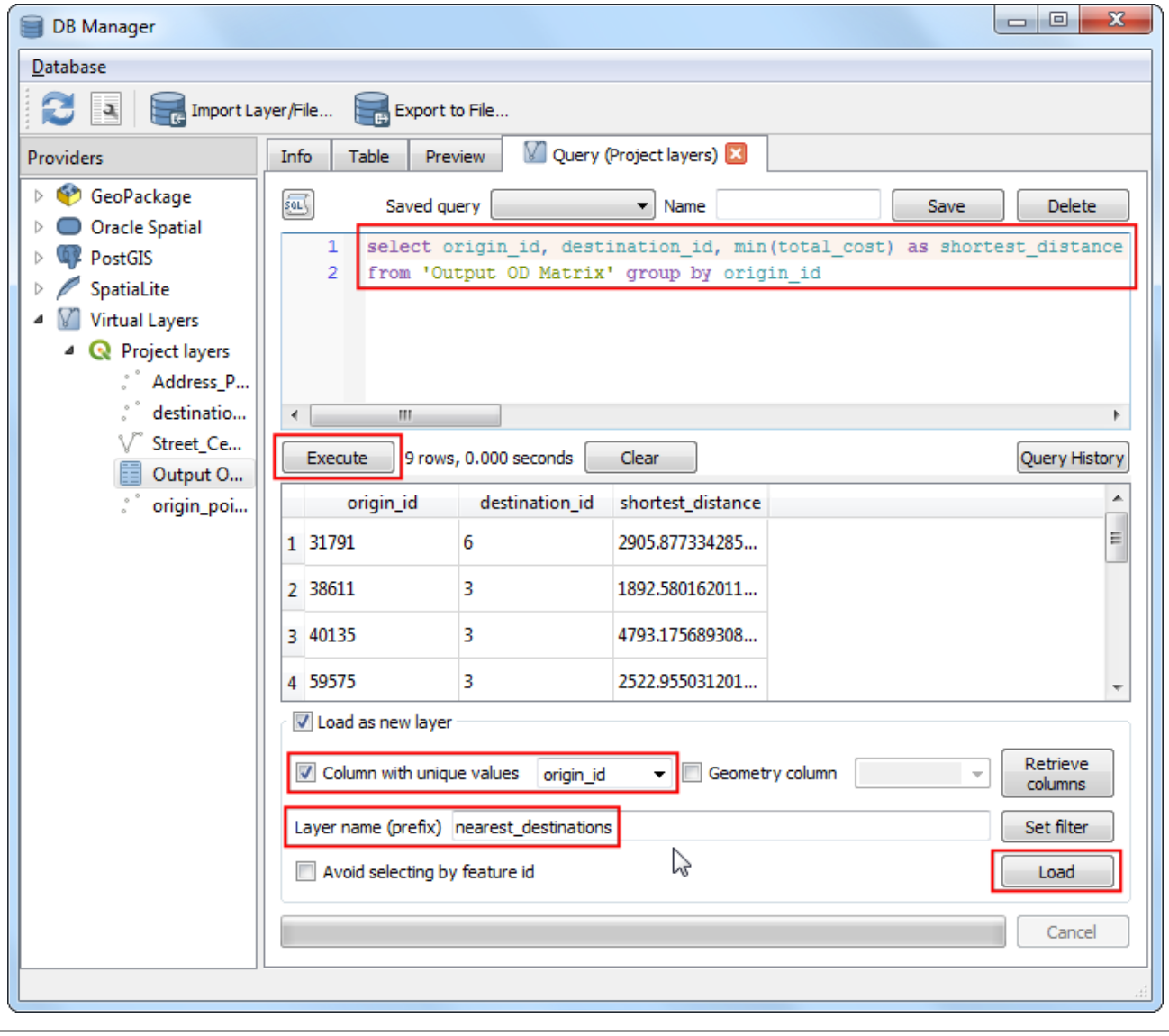


13. In the DB Manager dialog, select the Virtual Layers ▸ Project layers ▸ Output OD Matrix from the left-hand panel. See Virtual layers (https://docs.qgis.org/testing/en/docs/user_manual/working_with_vector/virtual_layers.html) documentation to learn more. Click the SQL Window button.

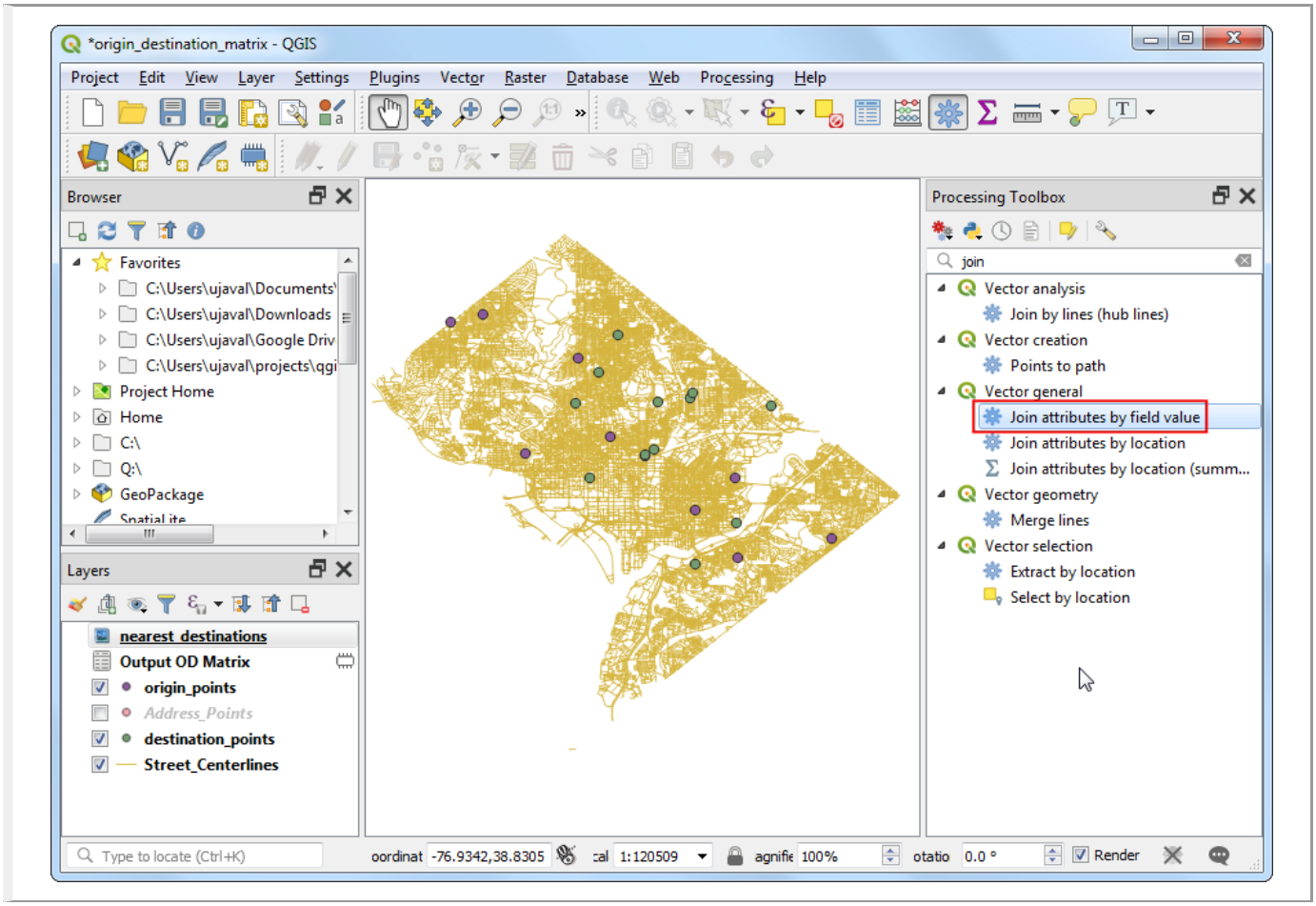


14. Enter the following query and click Execute. The results will be displayed in the panel below. As expected, we have 9 rows in the result - the shortest path destination for each origin point. Check and select Column with unique values as `origin_id` . Enter `nearest_destinations` as the Layer name (prefix). Click Load.

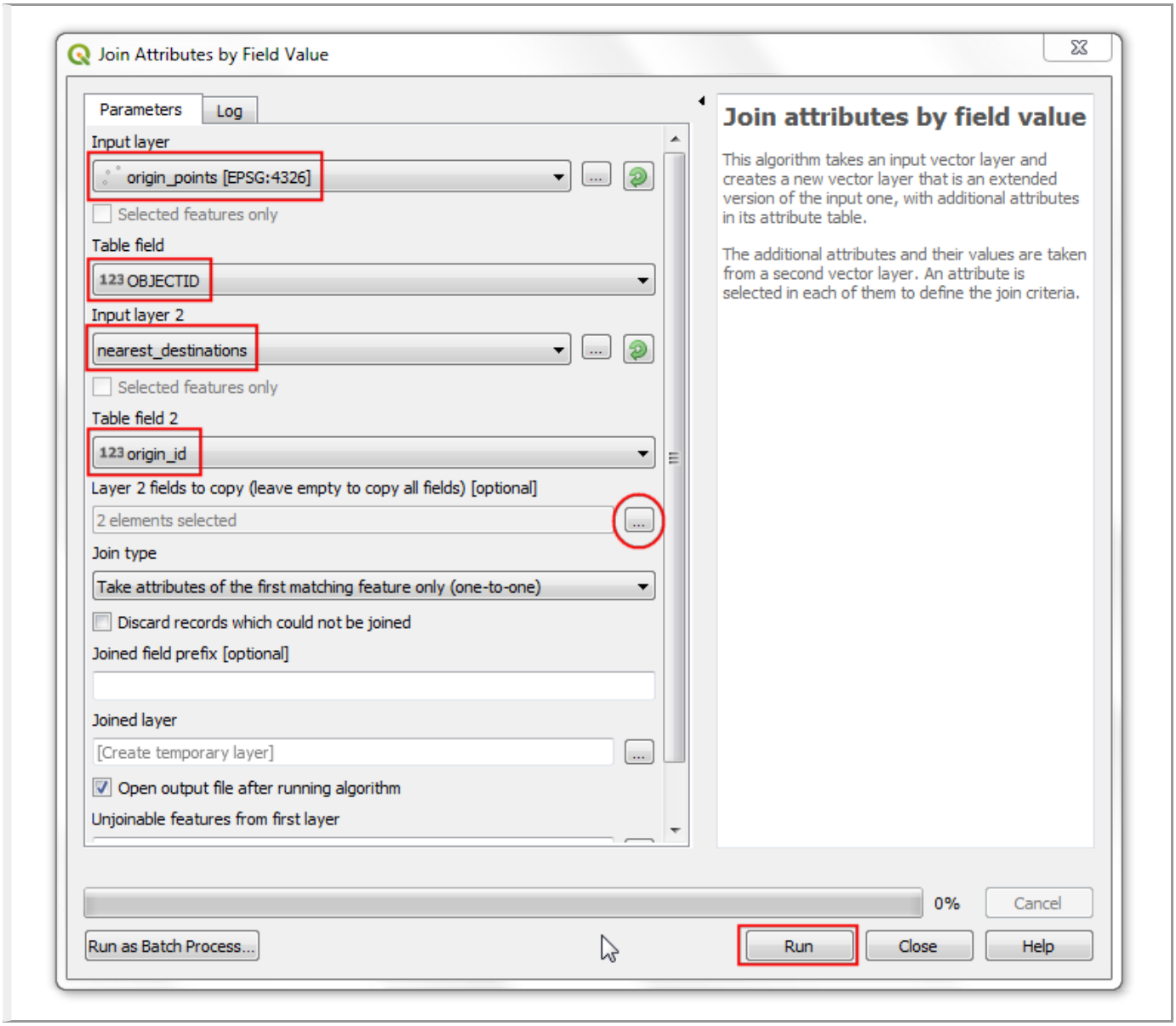
```
select origin_id, destination_id, min(total_cost) as shortest_distance
from 'Output OD Matrix' group by origin_id
```



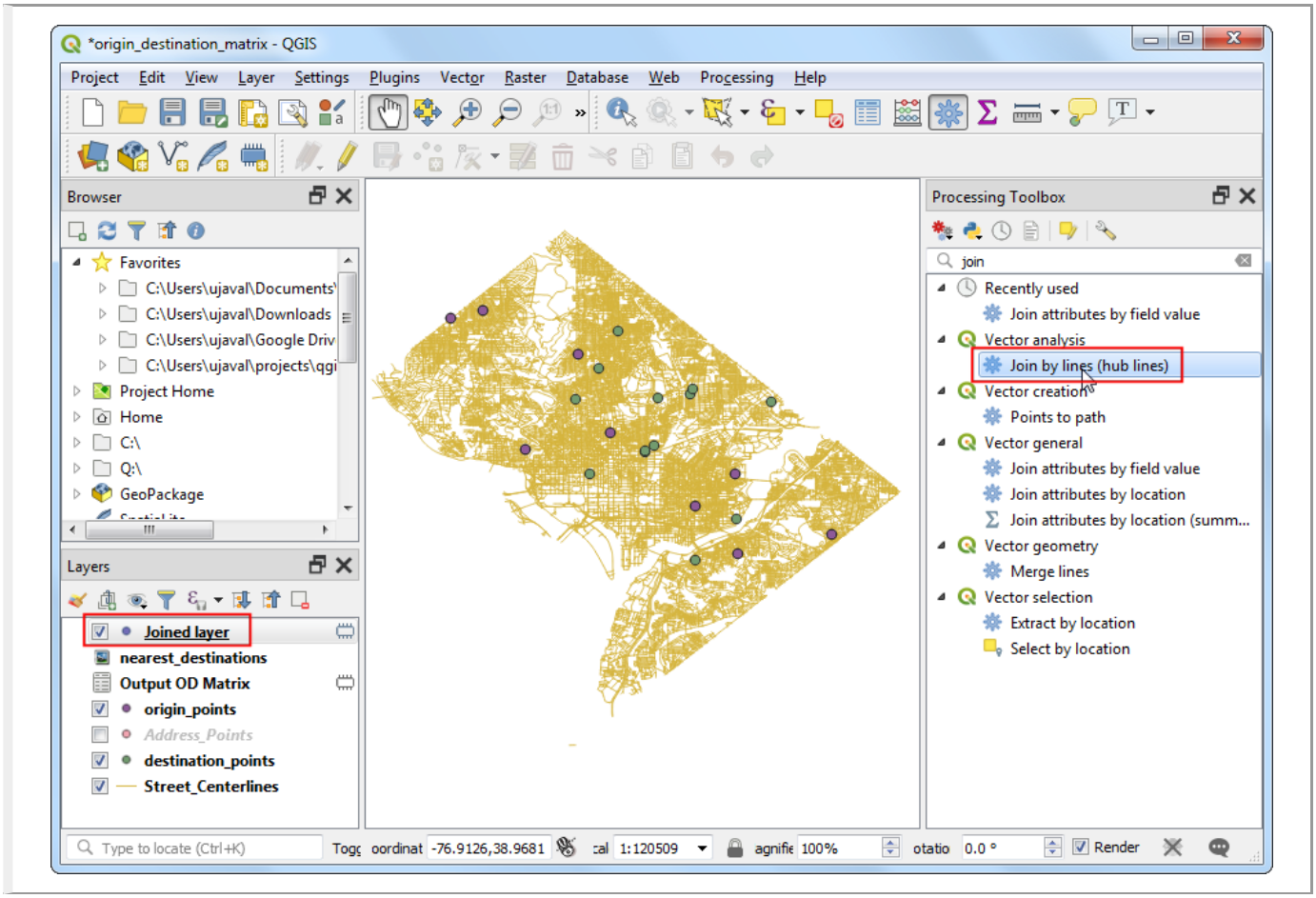
15. A new virtual layer `nearest_destinations` will be added to the Layers panel. This table has the result of our analysis. Nearest mental health center for each of the 9 origin points. Let's try a few different ways to visualize and validate these results. Go to Processing ▸ Toolbox. Search for and locate the Join attributes by field value algorithm. Double-click to launch it.



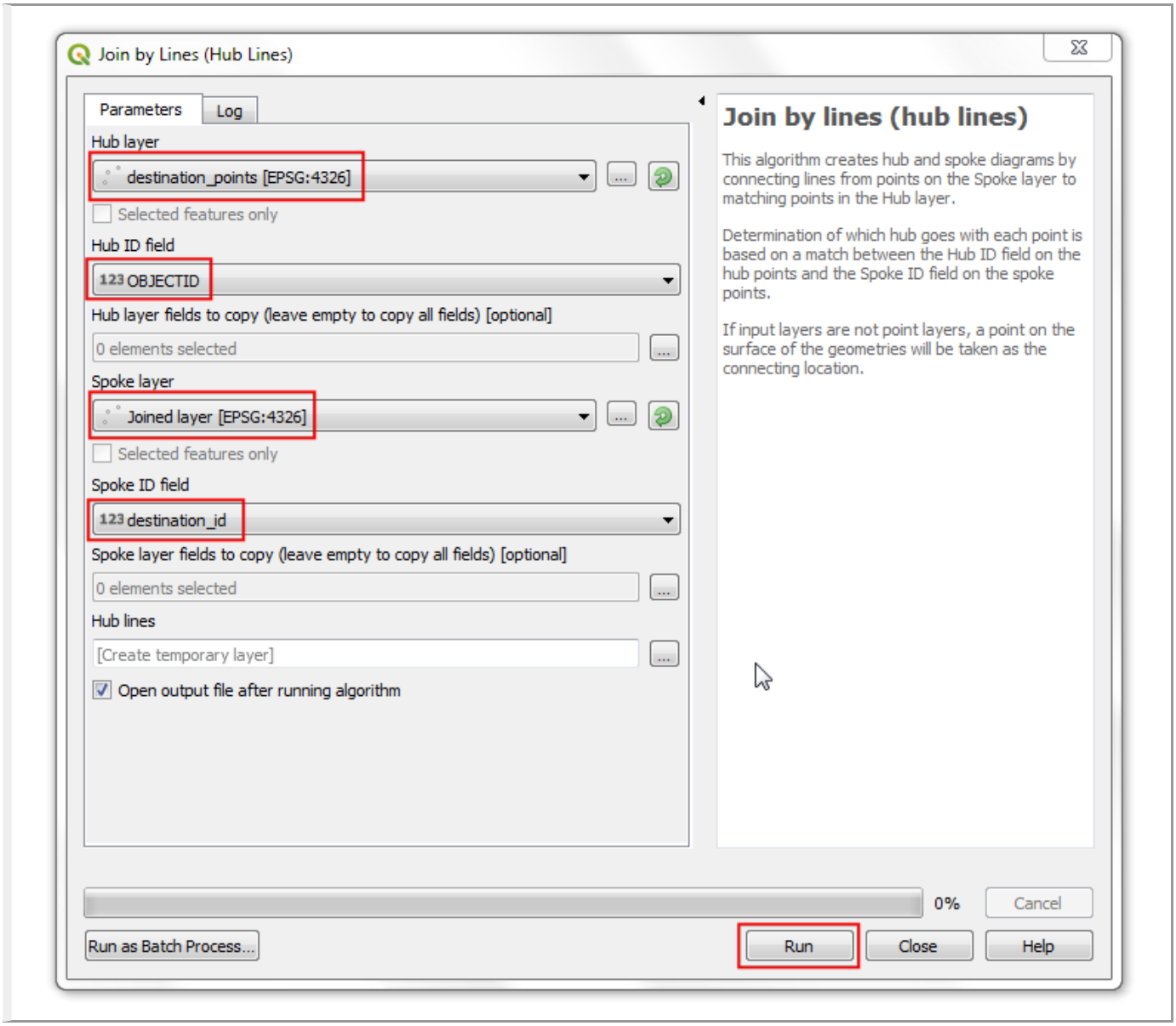
16. Select `origin_points` as the Input layer and `OBJECTID` as the Table field. Set `nearest_destinations` as the Input layer 2 and `origin_id` as the Table field 2. Click the ... button next to Layer 2 fields to copy and select `destination_id` and `shortest_distance` . Click Run.



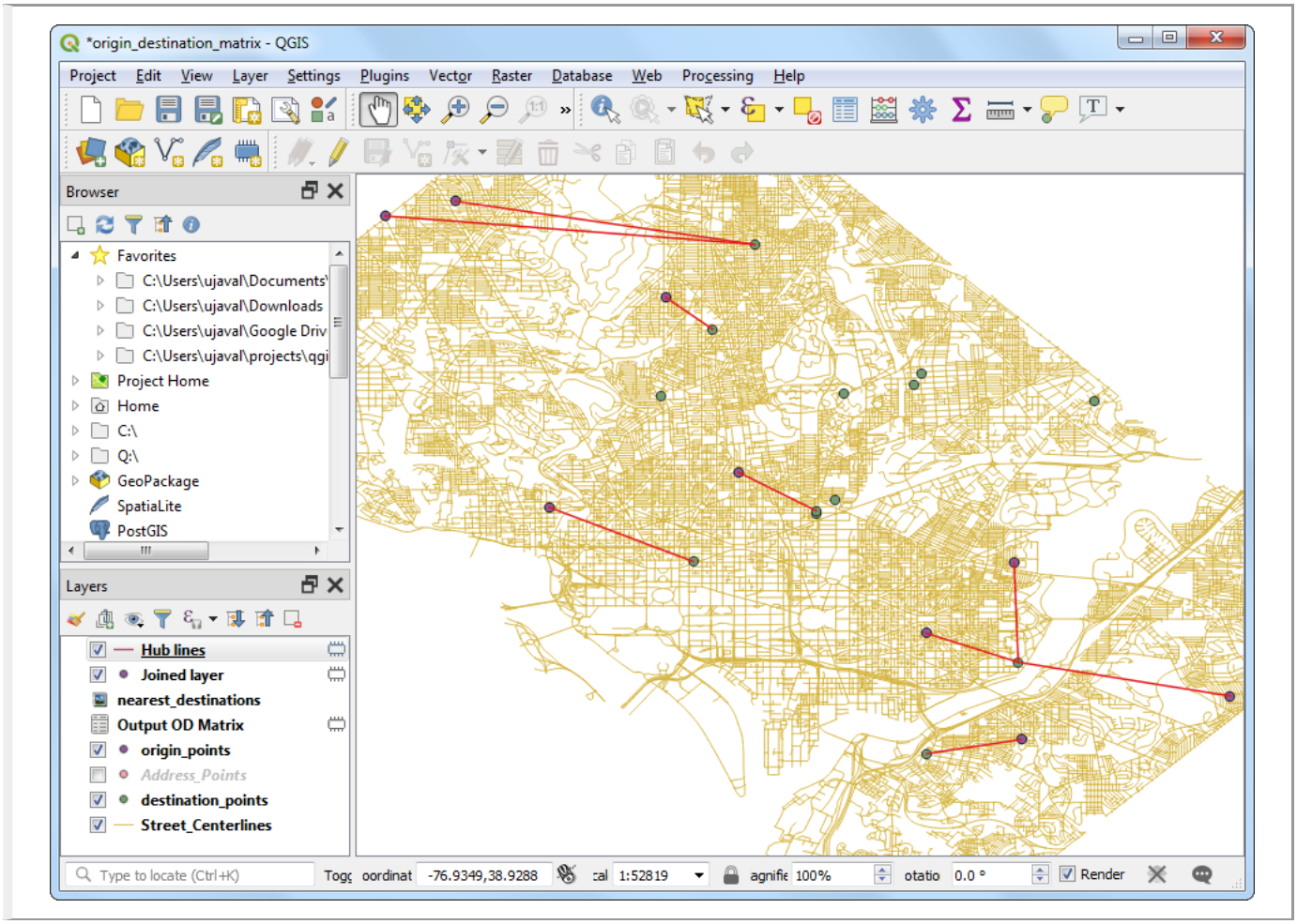
17. A new `Joined layer` will be added to the Layers panel. This layer has the nearest destination id attribute for each origin point. We can now create a hub-spoke visualization using this layer. Search for Vector analysis ▸ Join by lines (hub lines) algorithm. Right-click to launch it.



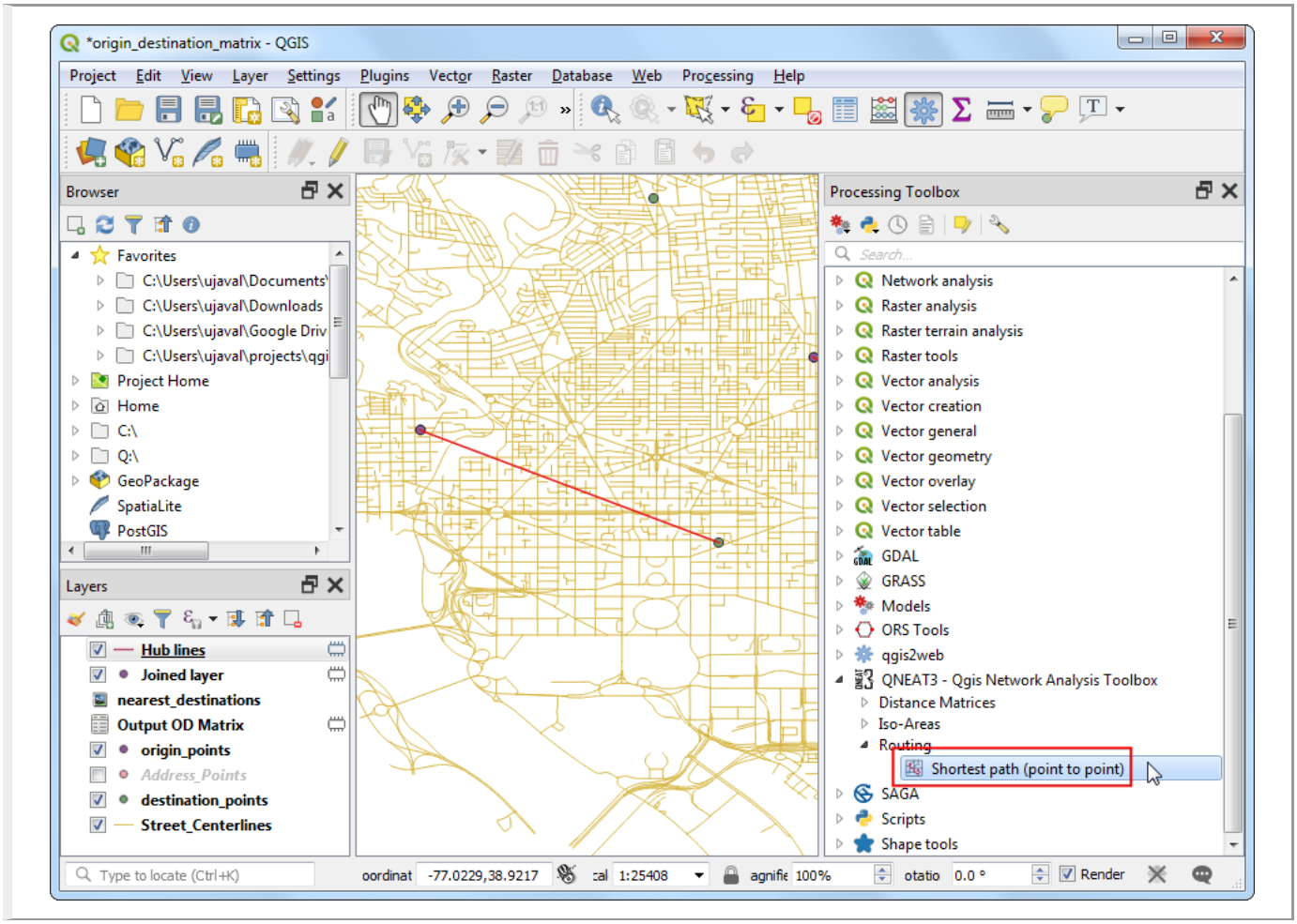
18. Select `destination_points` as the Hub layer and `OBJECTID` as the Hub ID field. Select `Joined layer` as the :guilabel:`Spoke layer` and ``destination_id` as the Spoke ID field. Click Run.



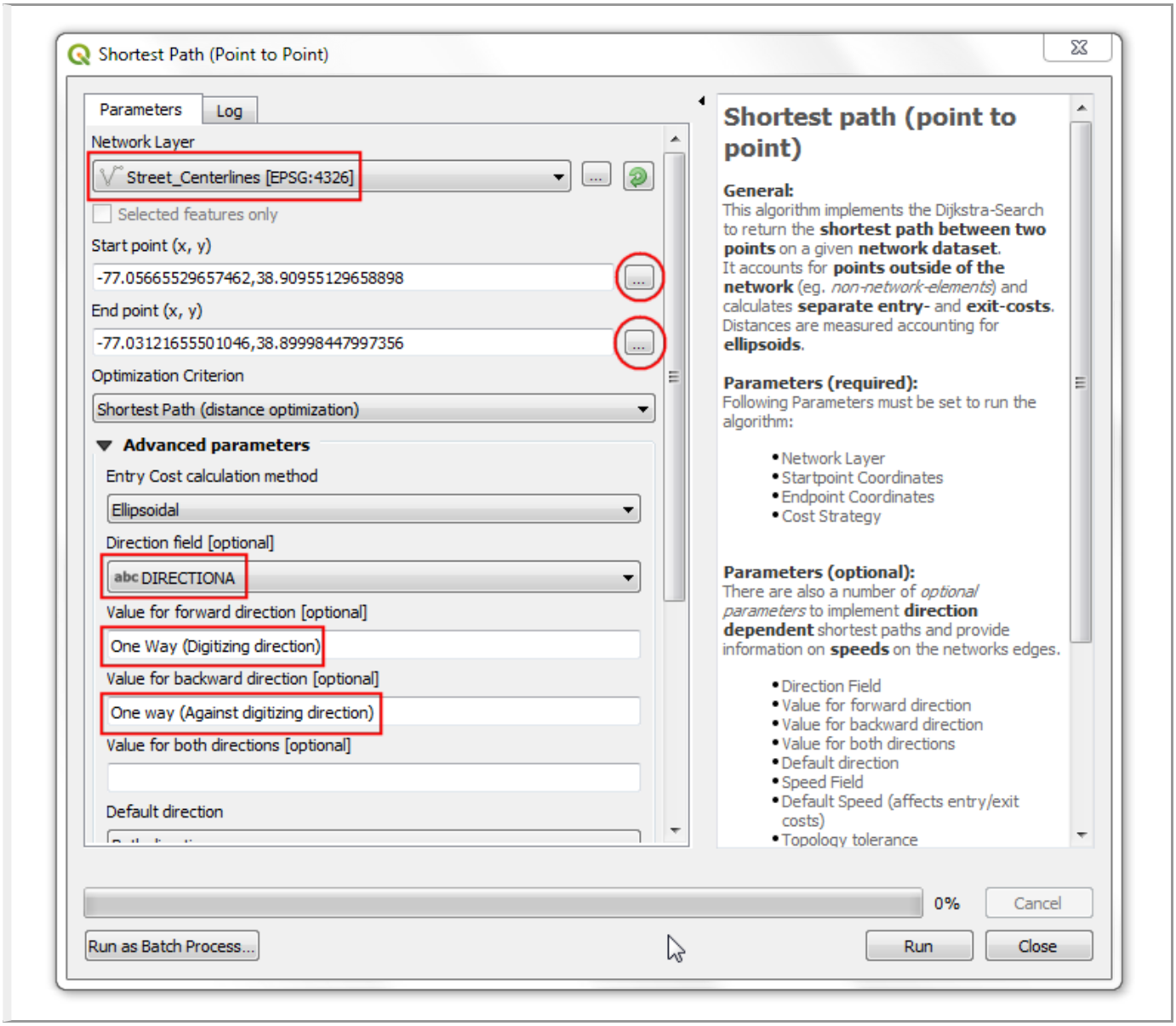
19. Once the processing finishes, a new layer `Hub lines` will be added to the Layers panel. This layer shows the lines connecting each origin with the nearest destination.



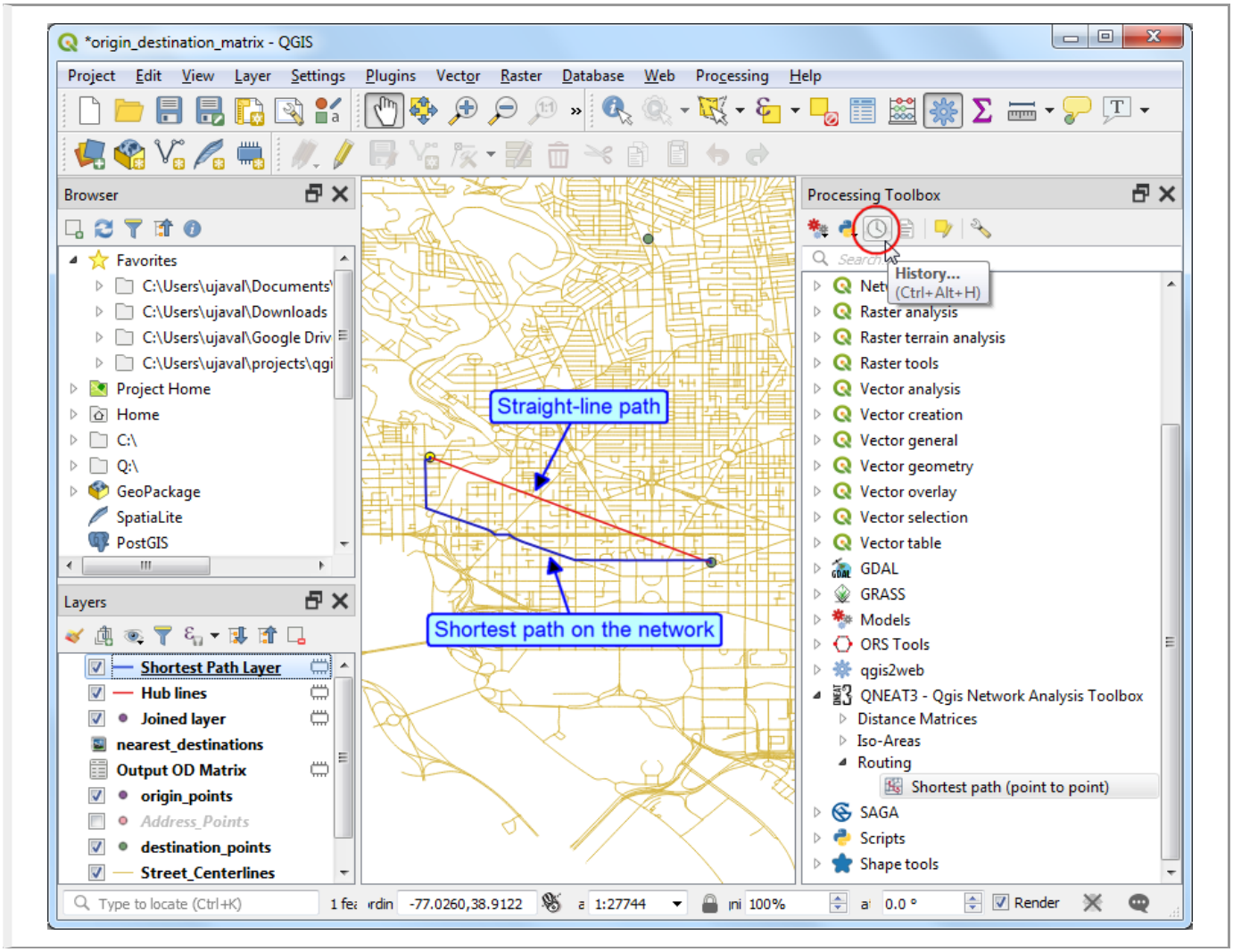
20. Note that even though the lines connecting the origin and destination is a straight-line, the destination was found using the distance along the network. It will be much useful visualization to show the actual shortest-path between each origin-destination. As of now, there is no easy way to generate the shortest-path between multiple origin-destination pairs the way we generated the distance matrix. But I will demonstrate a way to use some python scripting to generate this visualization. Firs, let's run the shortest path algorithm on 1 pair. Locate the QNEAT3 ‣ Routing ‣ Shortest path (point to point) algorithm and launch it.



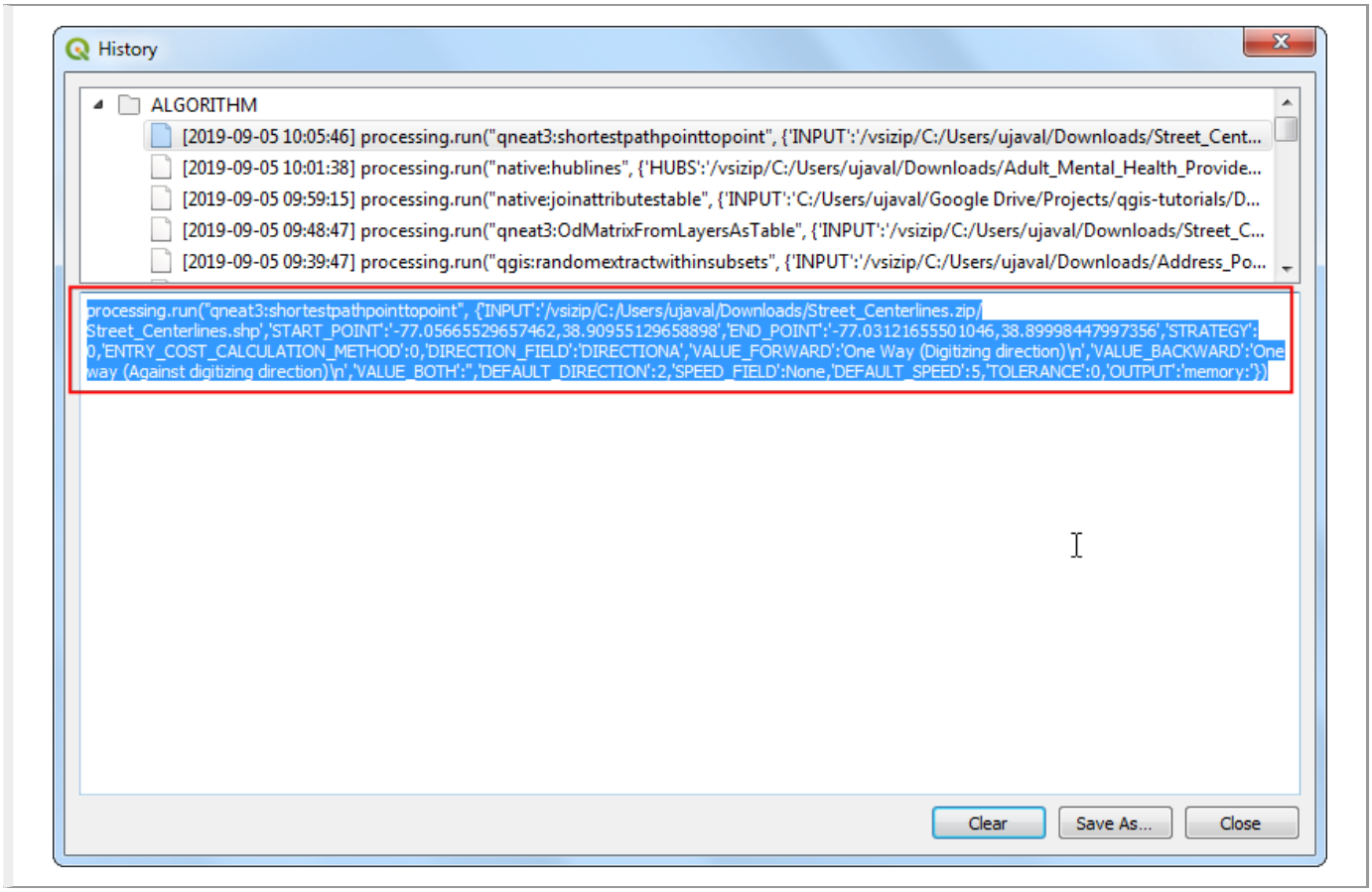
21. In the Shortest Path (Point to Point) dialog, select `Street_Centerlines` as the Vector layer representing network. Keep the Path type to calculate as `Shortest` . Next we need to pick a start and end point. You can click the ... button next to Start point and click on the origin point in the canvas. Similarly select the destination point as the End point. Expand the Advanced parameter section. Choose `DIRECTIONA` as the Direction field. Enter `One Way (Digitizing direction)` as the Value for forward direction and `One way (Against digitizing direction)` as the Value for backward direction. Keep other options to their default values and click Run.



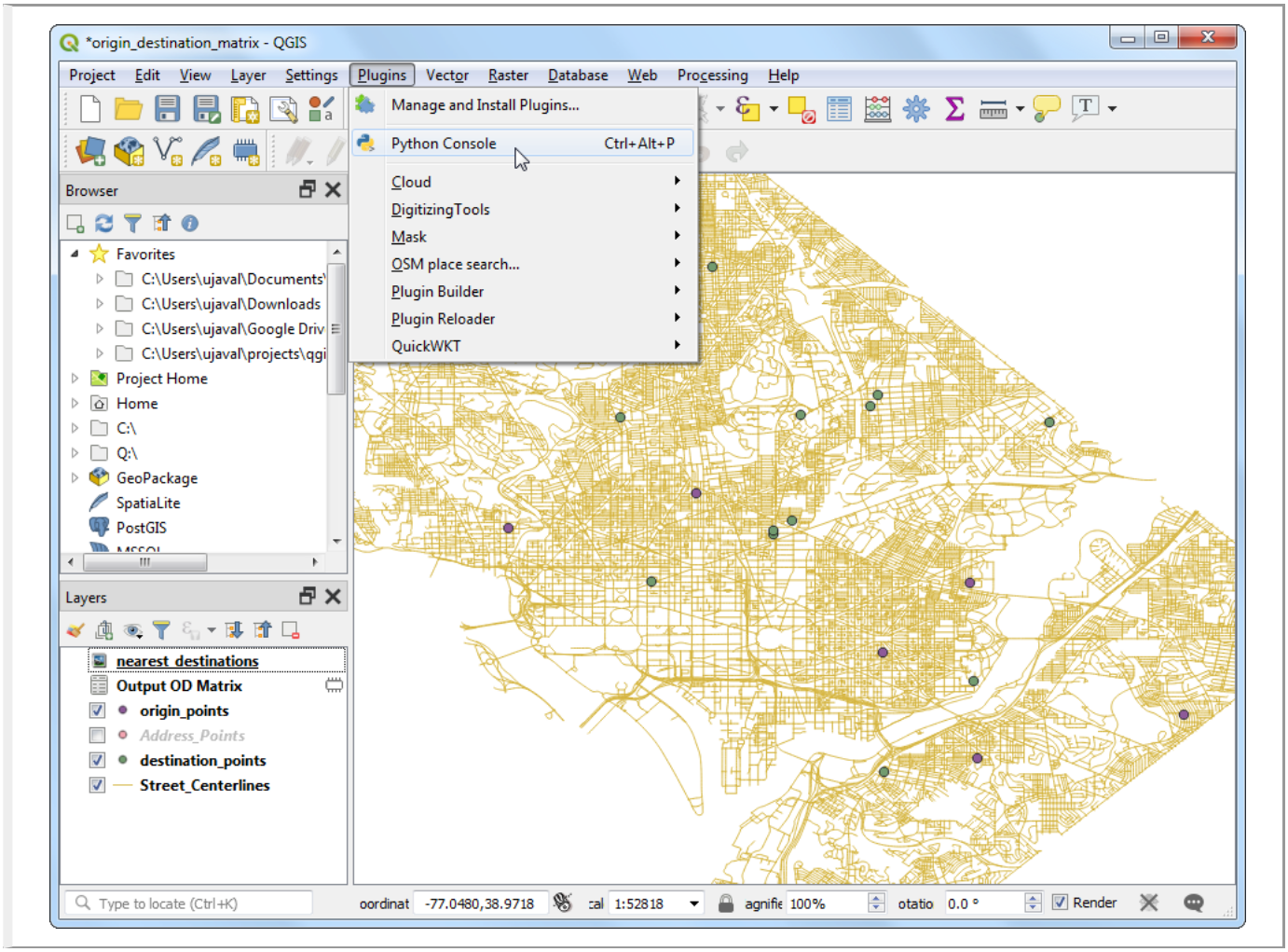
22. A new layer `Shortest Path Layer` wll be added to the Layers panel. You will see that this path follows the network rather than connecting the origin and destination with a straight line. The reason we ran the algorithm on 1 pair is to easily identify the parameter values that we can use in our script. Select both `Hub lines` and `Shortest Path layer` , right-click and select Remove Layer. Click the History button in the Processing Toolbox.



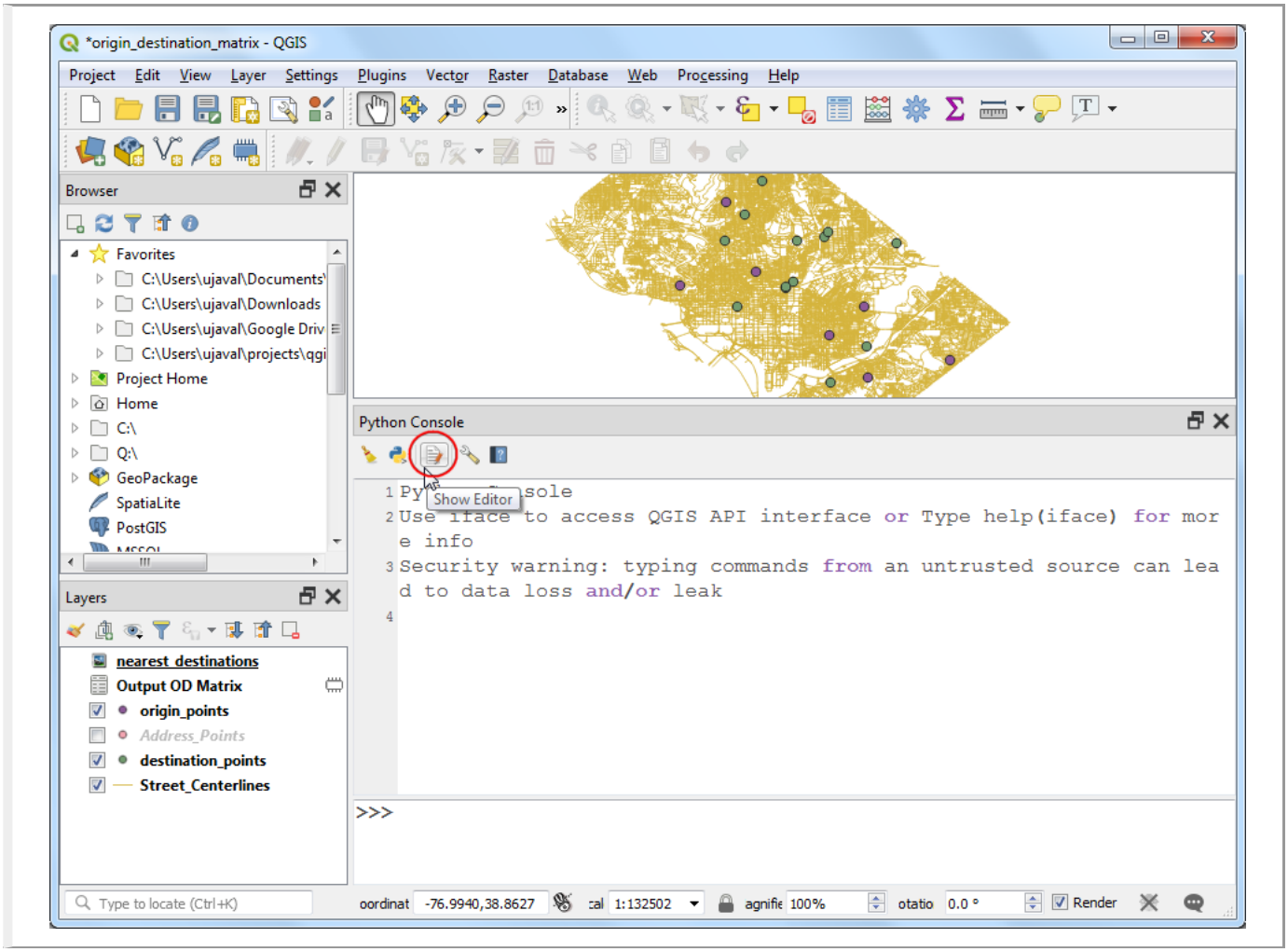
23. Pick the top-most algorithm and you will see the full command displayed in the panel below. Copy the command and click Close.



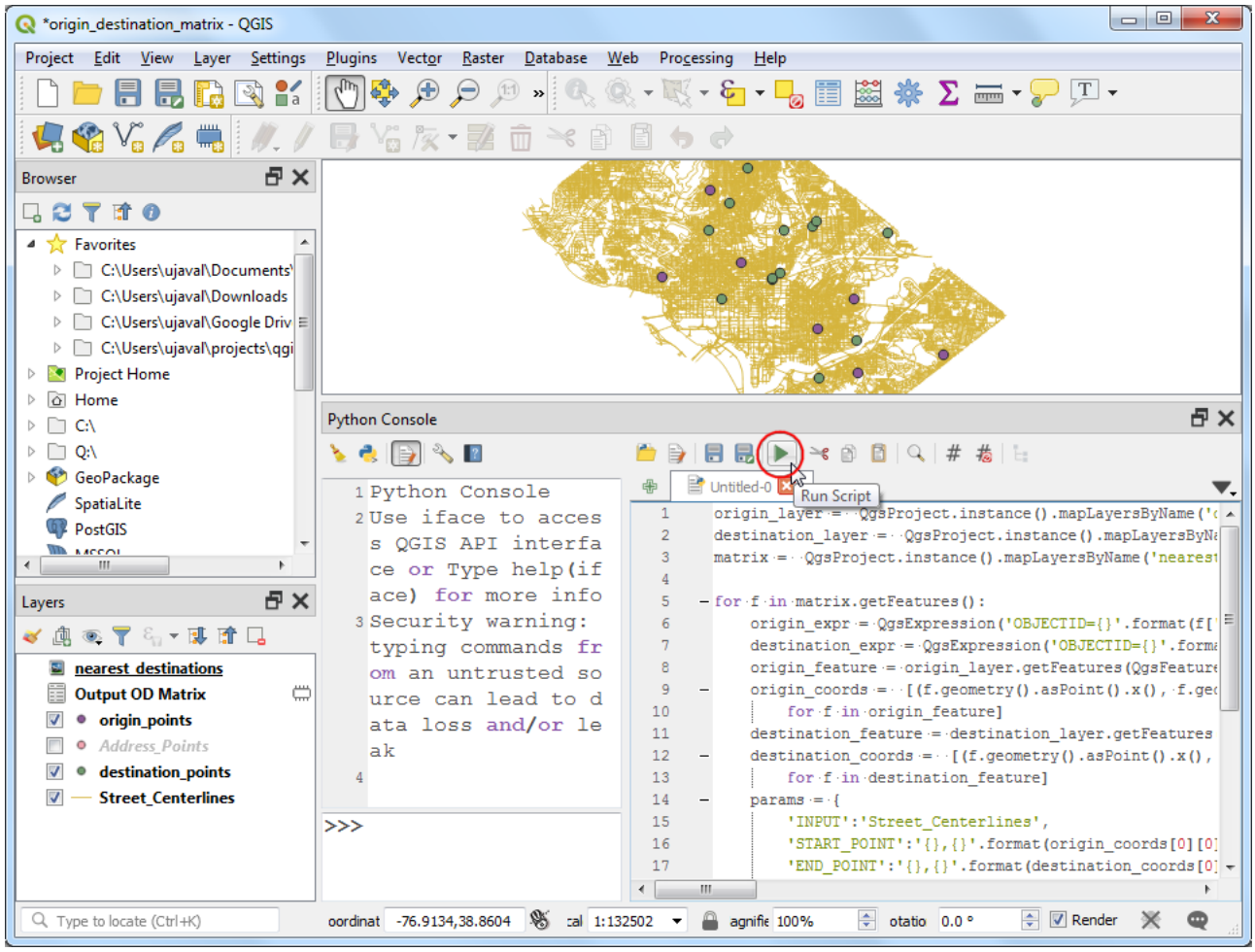
24. Go to Plugins ▸ Python Console.



25. Click the Show Editor button in the Python Console.

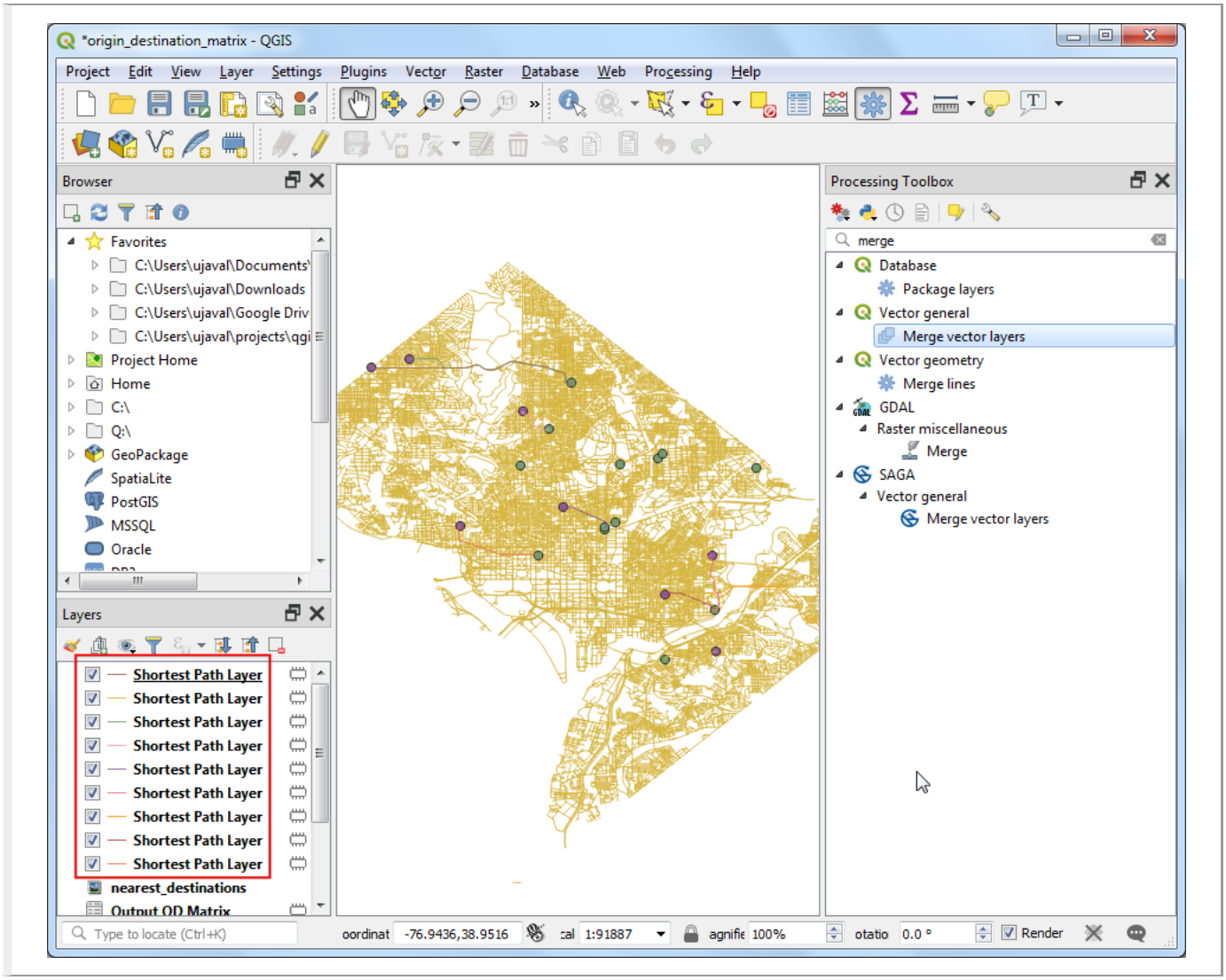


26. In the editor window, copy/paste the following script. This script uses the parameter values from the processing history that we saw earlier. Click Run Script button to start execution.

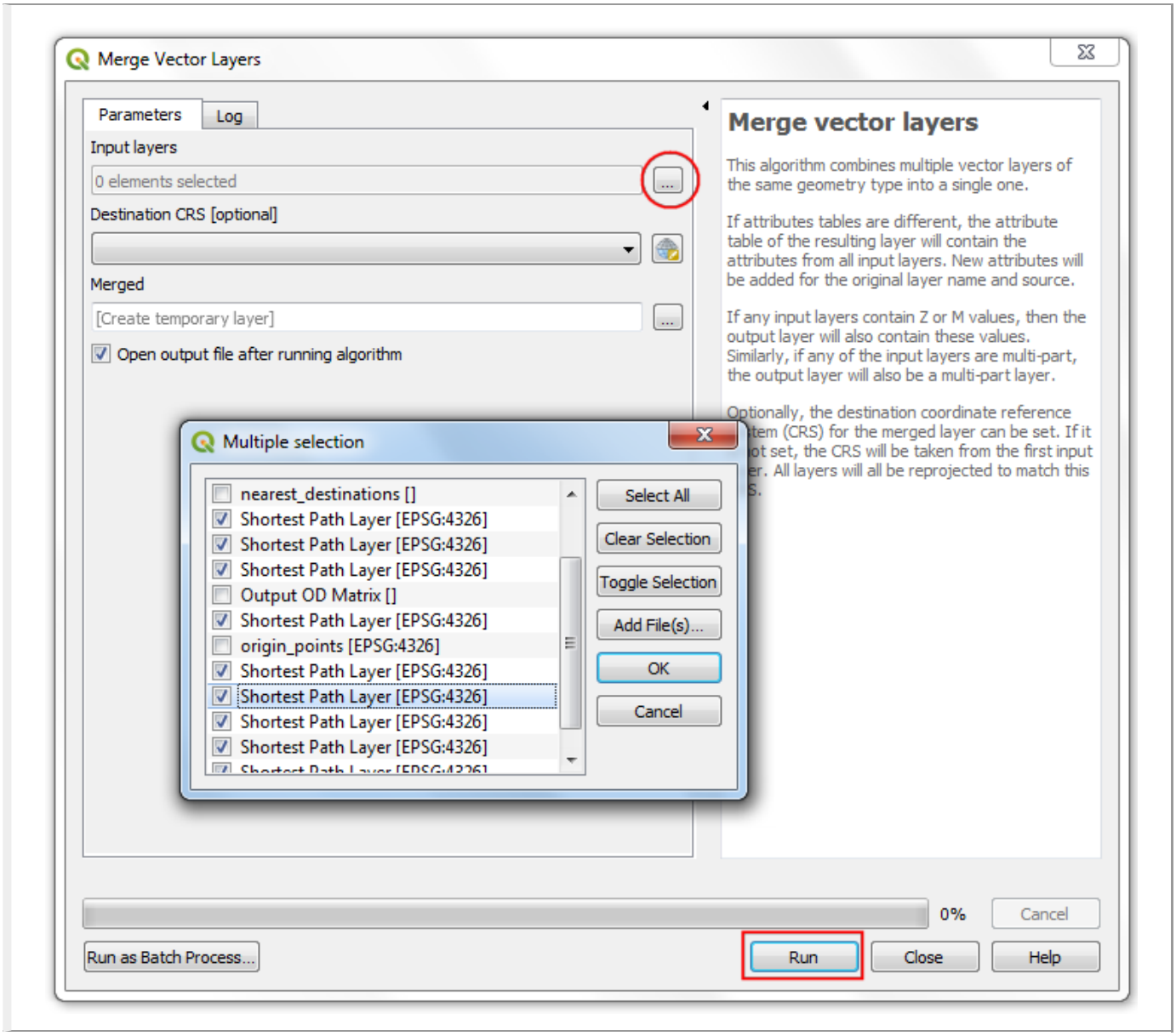

*origin_destination_matrix - QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser
Favorites
C:\Users\ujaval\Documents
C:\Users\ujaval\Downloads
C:\Users\ujaval\Google Driv
C:\Users\ujaval\projects\qgi
Project Home
Home
C:\
Q:\
GeoPackage
SpatiaLite
PostGIS
Layers
nearest_destinations
Output OD Matrix
origin_points
Address_Points
destination_points
Street_Centerlines
Python Console
1 Python Console
2 Use iface to acces
s QGIS API interfa
ce or Type help(if
ace) for more info
3 Security warning:
typing commands fr
om an untrusted so
urce can lead to d
ata loss and/or le
ak
4
>>>
Untitled-0
Run Script
1 origin_layer = QgsProject.instance().mapLayersByName('
2 destination_layer = QgsProject.instance().mapLayersByN
3 matrix = QgsProject.instance().mapLayersByName('nearest
4
5 for f in matrix.getFeatures():
6 origin_expr = QgsExpression('OBJECTID={}'.format(f[
7 destination_expr = QgsExpression('OBJECTID={}'.forma
8 origin_feature = origin_layer.getFeatures(QgsFeature
9 origin_coords = [(f.geometry().asPoint().x(), f.ge
10 for f in origin_feature]
11 destination_feature = destination_layer.getFeatures
12 destination_coords = [(f.geometry().asPoint().x(),
13 for f in destination_feature]
14 params = {
15 'INPUT':'Street_Centerlines',
16 'START_POINT':'{},{}'.format(origin_coords[0][0]
17 'END_POINT':'{},{}'.format(destination_coords[0]
Type to locate (Ctrl+K)
oordinat -76.9134,38.8604
cal 1:132502
agnifie 100%
otatio 0.0 °
Render

```
origin_layer =  QgsProject.instance().mapLayersByName('origin_points')[0]
destination_layer =  QgsProject.instance().mapLayersByName('destination_points')[0]
matrix =  QgsProject.instance().mapLayersByName('nearest_destinations')[0]

for f in matrix.getFeatures():
    origin_expr = QgsExpression('OBJECTID={}'.format(f['origin_id']))
    destination_expr = QgsExpression('OBJECTID={}'.format(f['destination_id']))
    origin_feature = origin_layer.getFeatures(QgsFeatureRequest(origin_expr))
    origin_coords =  [(f.geometry().asPoint().x(), f.geometry().asPoint().y())
        for f in origin_feature]
    destination_feature = destination_layer.getFeatures(QgsFeatureRequest(destination_ex
    destination_coords =  [(f.geometry().asPoint().x(), f.geometry().asPoint().y())
        for f in destination_feature]
    params = {
        'INPUT':'Street_Centerlines',
        'START_POINT':'{},{}'.format(origin_coords[0][0], origin_coords[0][1]),
        'END_POINT':'{},{}'.format(destination_coords[0][0], destination_coords[0][1]),
        'STRATEGY':0,
        'ENTRY_COST_CALCULATION_METHOD':0,
        'DIRECTION_FIELD':'DIRECTIONA',
        'VALUE_FORWARD':'One Way (Digitizing direction)\n',
        'VALUE_BACKWARD':'One way (Against digitizing direction)\n',
        'VALUE_BOTH':'',
        'DEFAULT_DIRECTION':2,
        'SPEED_FIELD':None,
        'DEFAULT_SPEED':5,
        'TOLERANCE':0,
        'OUTPUT':'memory:'}
    print('Executing analysis')
    processing.runAndLoadResults("qneat3:shortestpathpointtopoint", params)
```

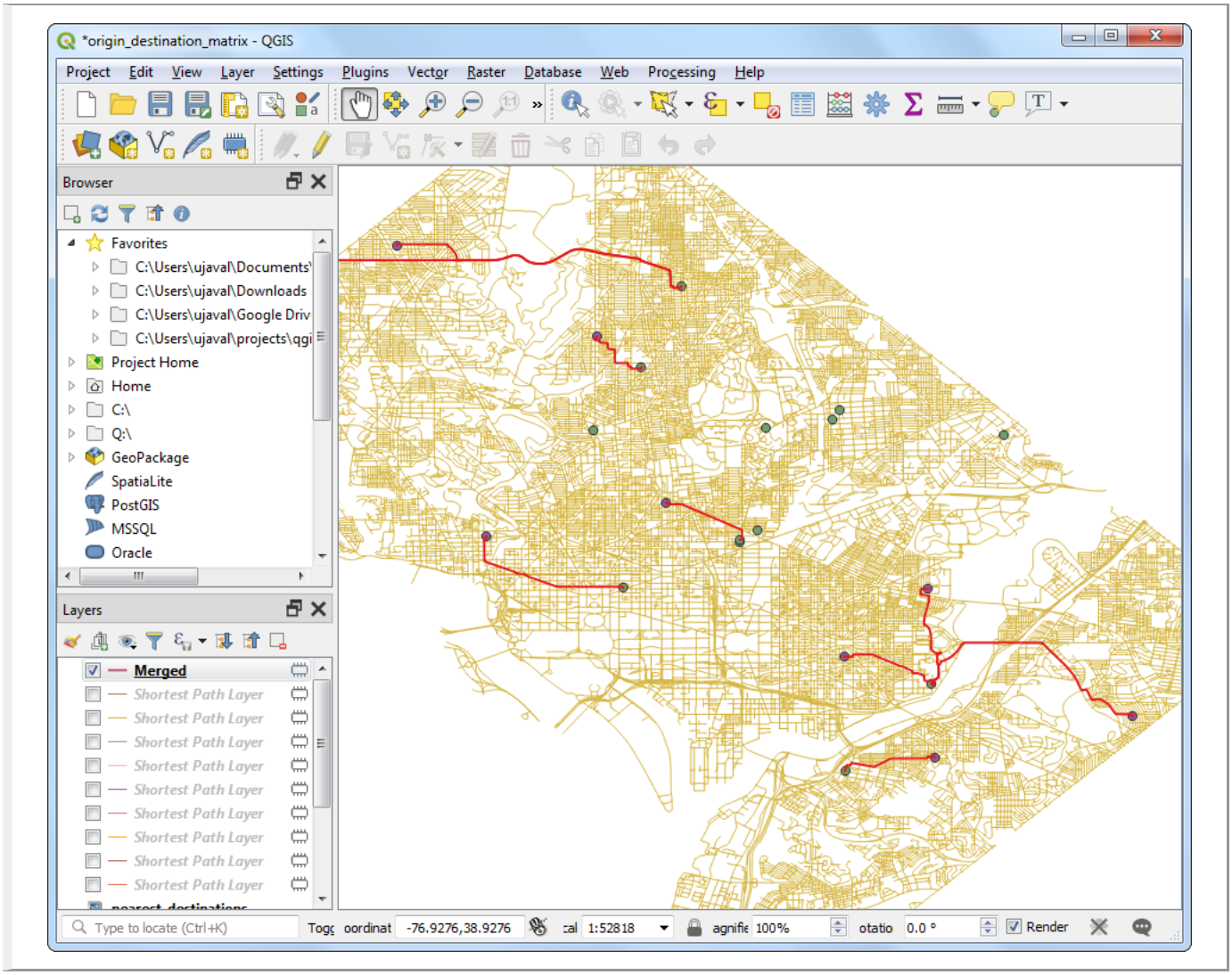
27. The script will take a few minutes to run. Once finished, you will see 9 new layers named `Shortest Path layer`. Let's merge these paths to a single layer. Find the Vector general ▸ Merge vector layers algorithm and launch it.



28. Select all 9 `Shortest Path layer` as the Input layers. Click Run.



29. A new `Merged` layer will be created which will contain shortest path between our origins and destinations.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Service Area Analysis using Openrouteservice (QGIS3)

Service area analysis is useful in evaluating accessibility of locations. Given locations of fire stations, hospitals, public transit stations etc. you can use such analysis to identify what areas can be served from these locations by either amount of distance traveled or by time taken. Till recently, such analysis was difficult using open-source tools and data. But now we have access to a global street network using OpenStreetMap (OSM) and free web-services such as Openrouteservice (ORS) that can perform complex routing tasks using OpenStreetMap (OSM) data. In this tutorial, we will use the **ORS Tools Plugin** to perform service area analysis in QGIS.

Overview of the task

We will use metro rail station data for Kochi, India to determine areas that are within 15-minutes of walking distance.

Other skills you will learn

- How to load General Transit Feed Specification (GTFS) transit feed data in QGIS.
- How to convert sequential point data to line tracks using the *Points to Path* tool.

Get the data

Kochi Metro Rail Limited (KMRL) (https://kochimetro.org) provides open data for the Kochi Metro Rail Project in Global Transit Feed Specification (GTFS-static) (https://developers.google.com/transit/gtfs/reference/) format. Request for data download by visiting the Open Data (https://kochimetro.org/open-data/) page.

For convenience, you may directly download a copy of the datasets from the links below:

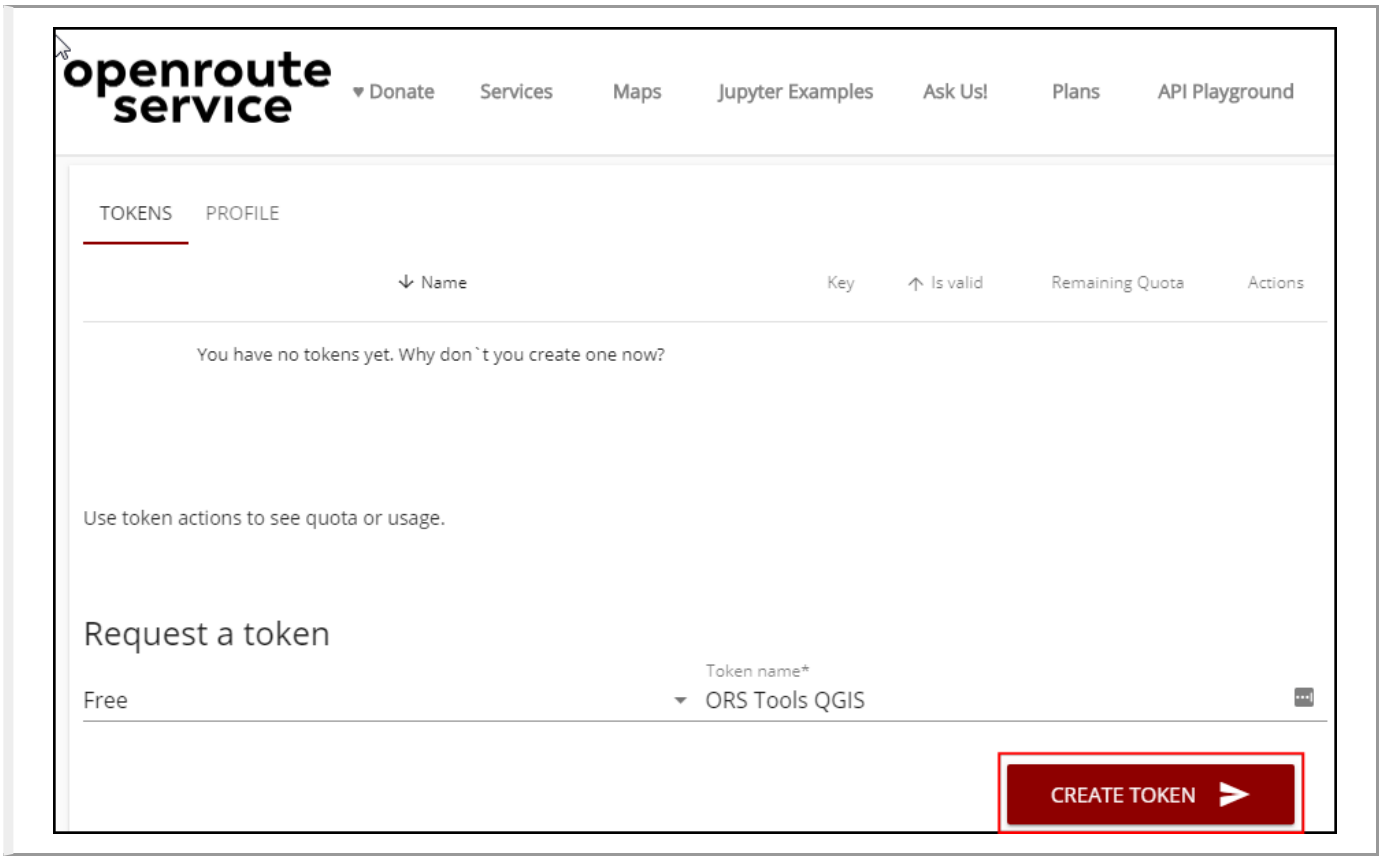
KMRL-Open-Data.zip (http://www.qgistutorials.com/downloads/KMRL-Open-Data.zip)

Data Source [KMRL] (../credits.html#kmrl)

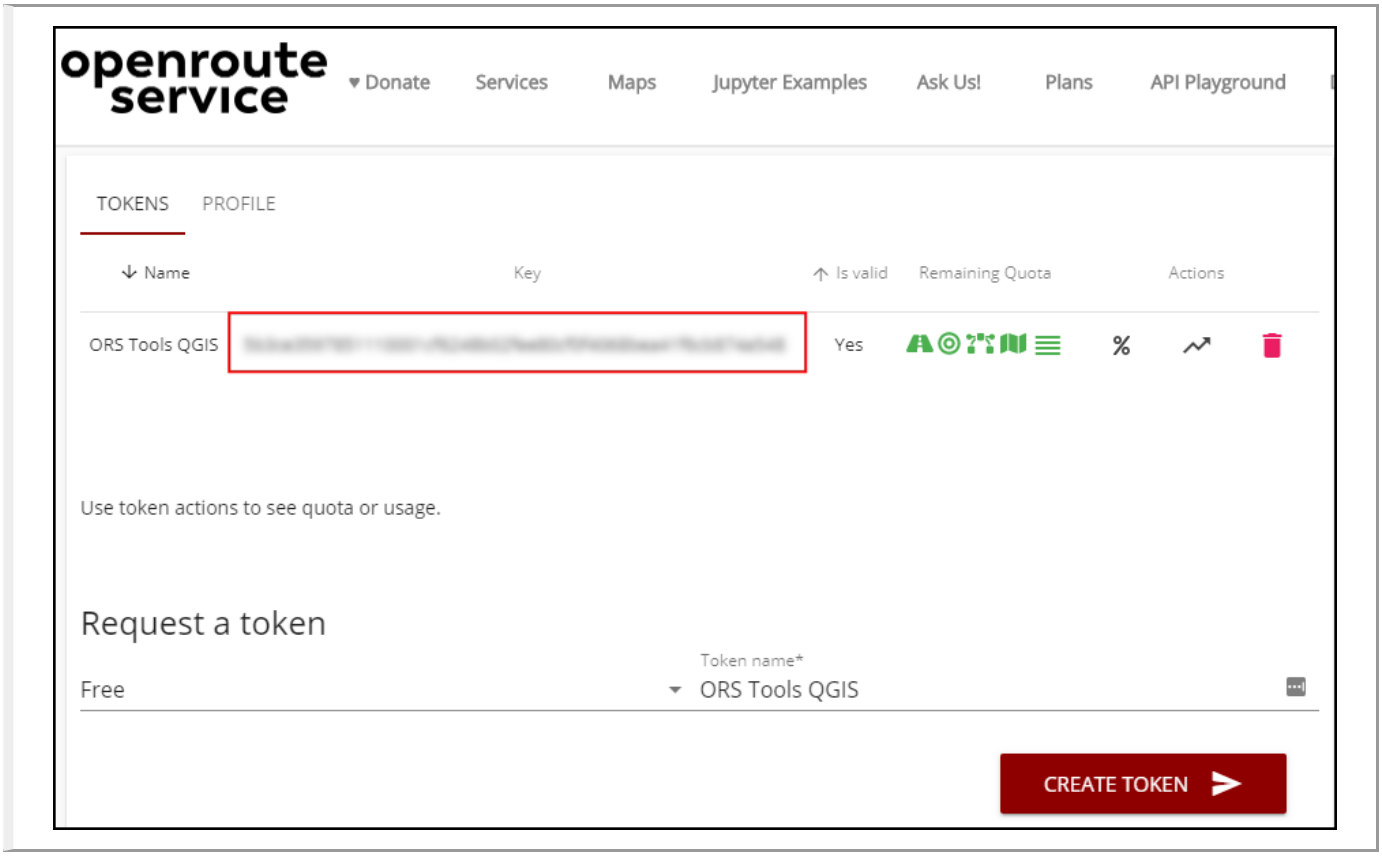
Setup

Openrouteservice API (https://openrouteservice.org/) provides routing algorithms that work on free geographic data from OpenStreetMap. It is a free web-based service that can be accessed via a QGIS plugin. While the service is free, it requires you to sign-up and get an API key. The API key is used to prevent abuse and enforce limits on usage.

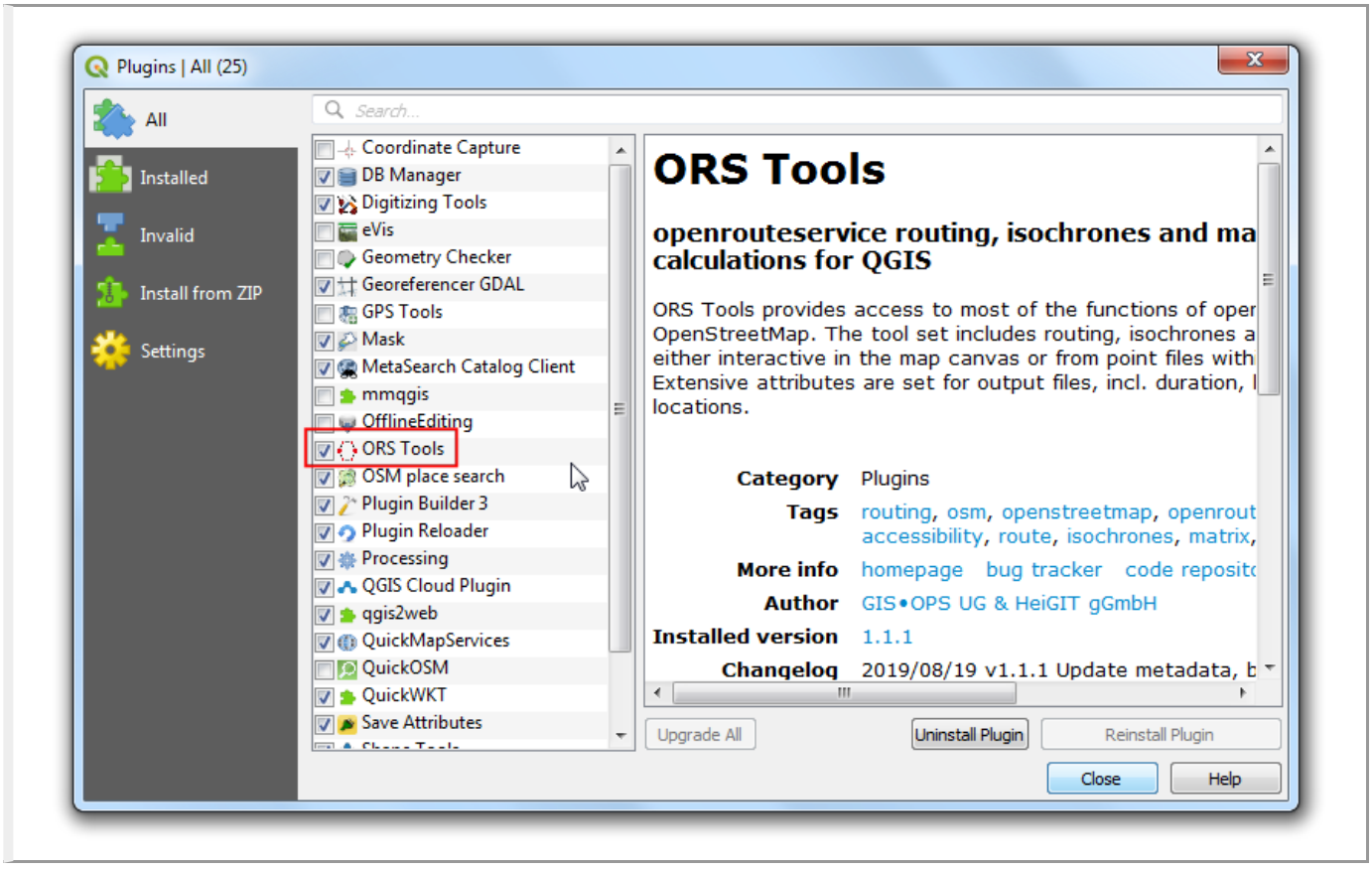
1. Visit Openrouteservice Sign Up (https://openrouteservice.org/dev/#/signup) page and create an account. Once your account is activated, visit your Dashboard (https://openrouteservice.org/dev/#/home) and request a token. Select `Free` as the Token type and enter `ORS Tools QGIS` as the Token name. Click CREATE TOKEN.



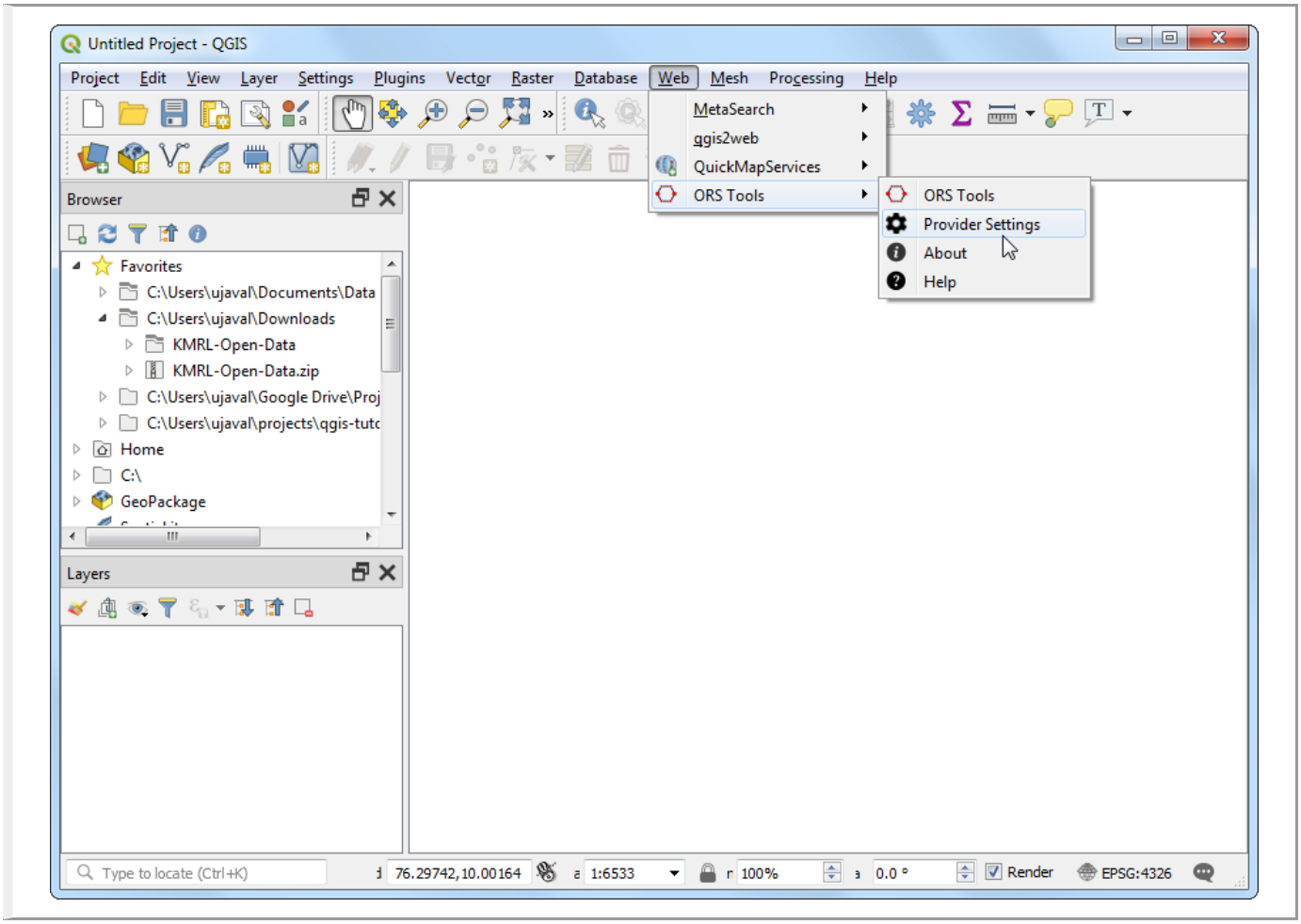
2. Once created, copy the long string displayed under `Key`. This is a unique identifier linked with your account that will be used to authorize use of this service.



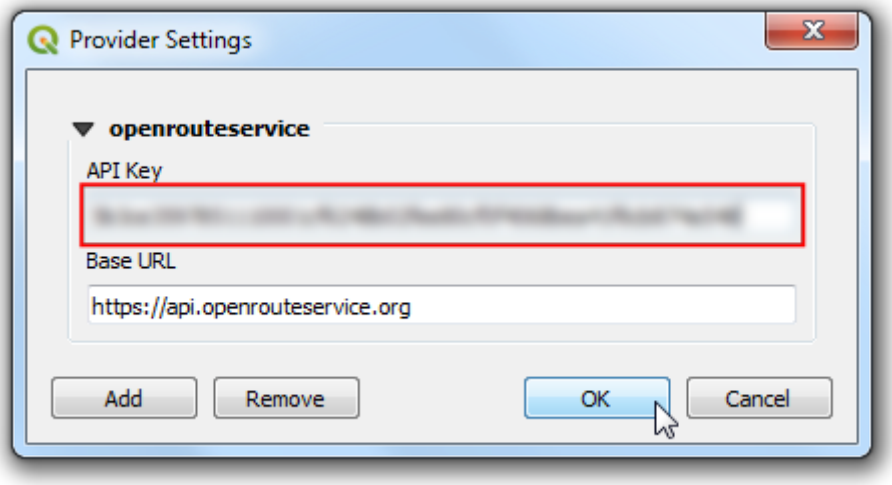
3. Open QGIS. Visit Plugins ‣ Manage and Install plugins. Search for **ORS Tools** plugin and install it. Click Close.



4. In the main QGIS Window, go to Web ‣ ORS Tools ‣ Provider Settings.

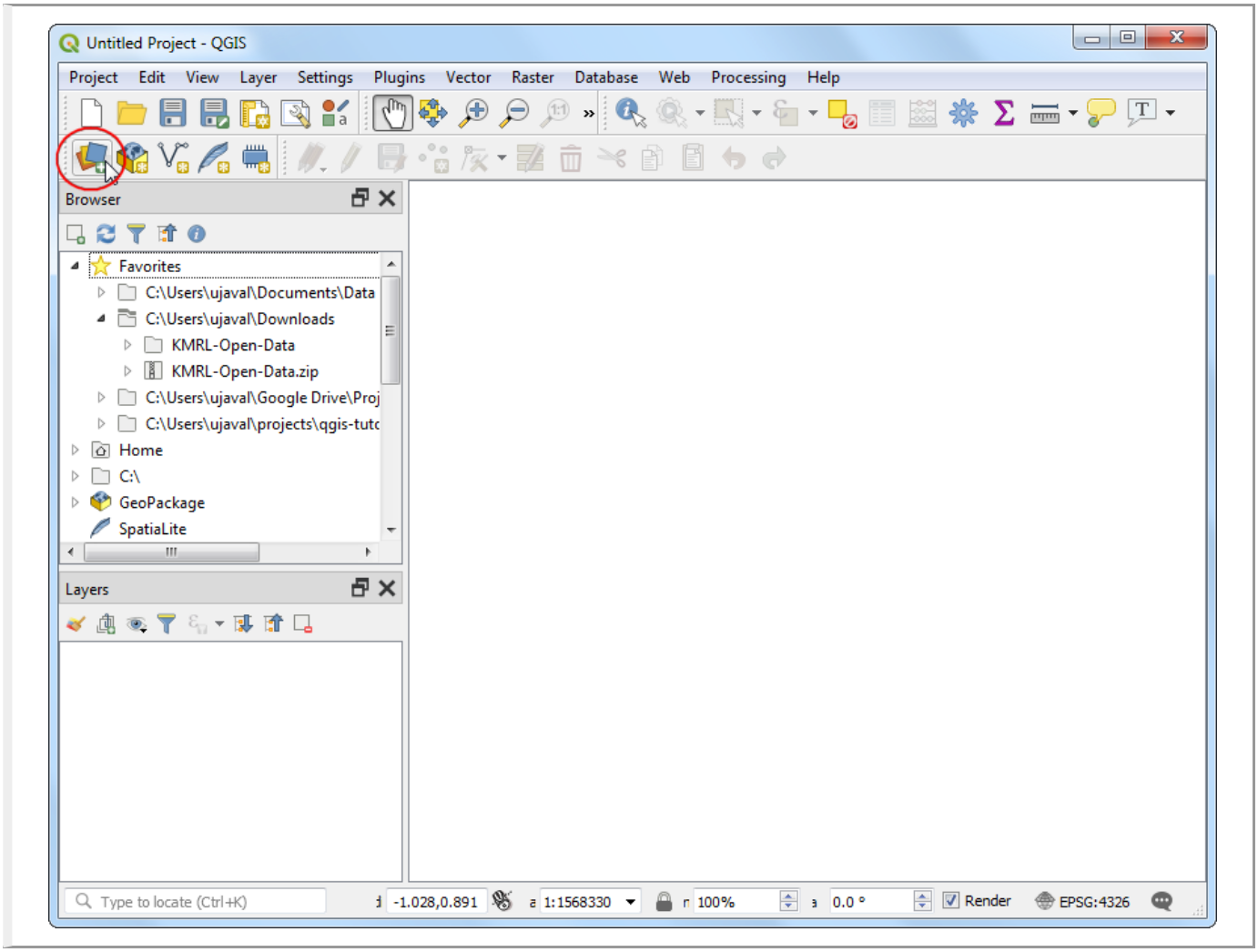


5. Expand the openrouteservice section and paste the `key` (copied in step 2) in the API Key text-box. Click OK.

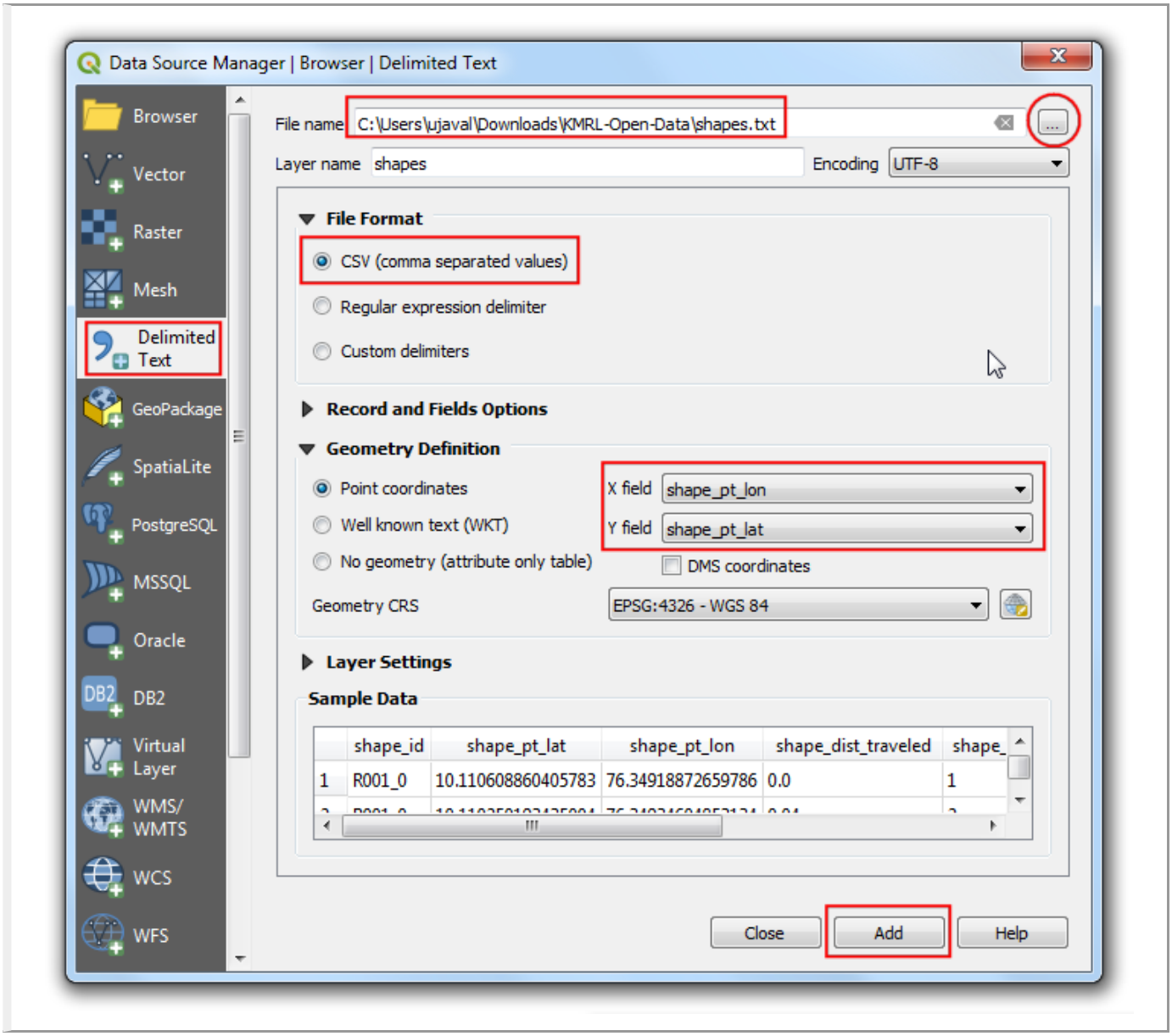


Procedure

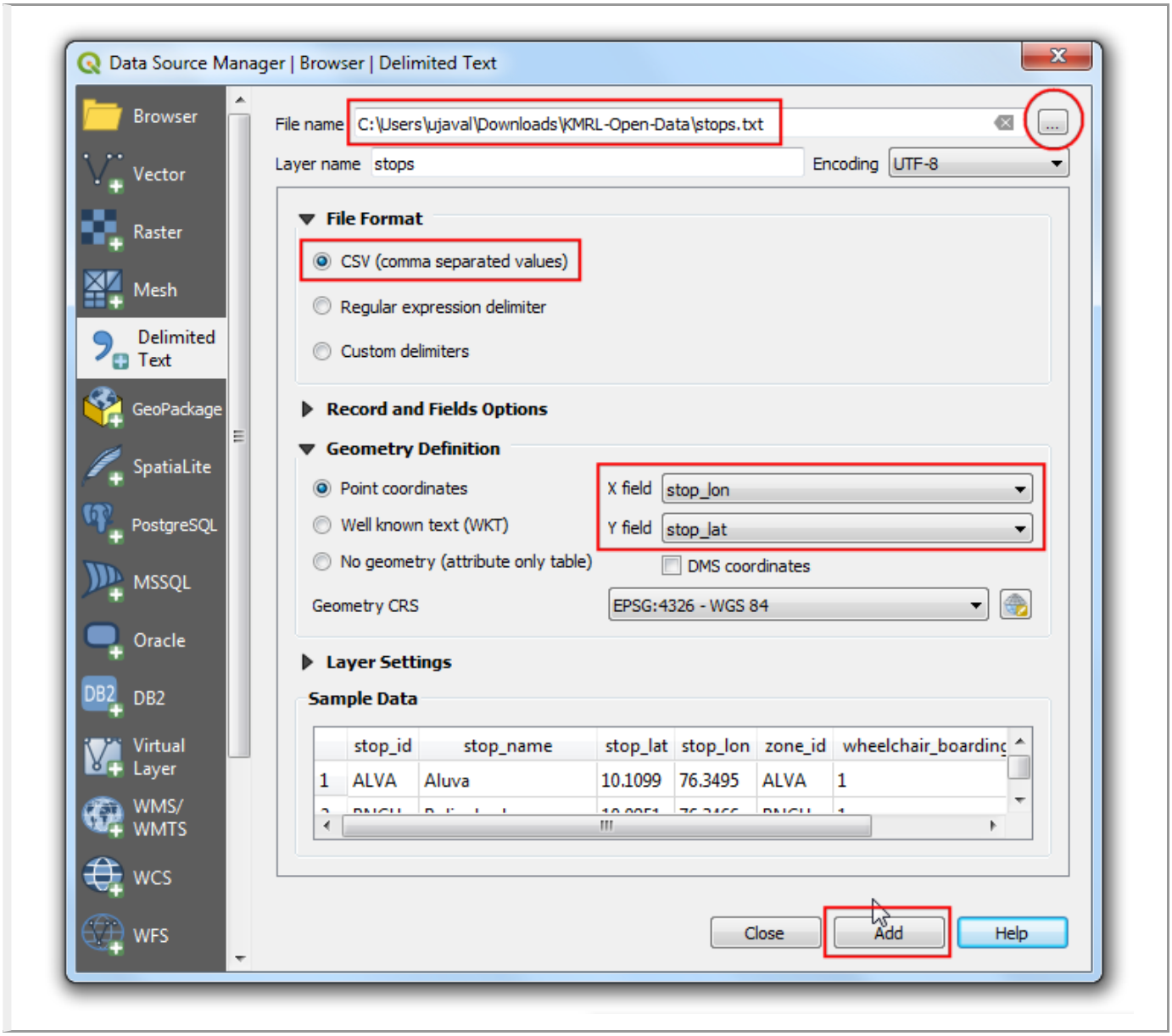
1. Unzip the downloaded `KMRL-Open-Data.zip` file to a folder on your computer. You will notice that the unzipped directory contains many text files. Each file contains data for a different aspect of the transit system. The format of the files and their uses are described in GTFS Reference (https://developers.google.com/transit/gtfs/reference/). Out of all the files, 2 files contains geospatial data and is of interest to us. The file `shapes.txt` contains points that describe a physical path that the vehicle takes, and the file `stops.txt` contains the location of each transit stop. Both of these are CSV files that can be imported into QGIS. Click the Open Data Source Manager button.



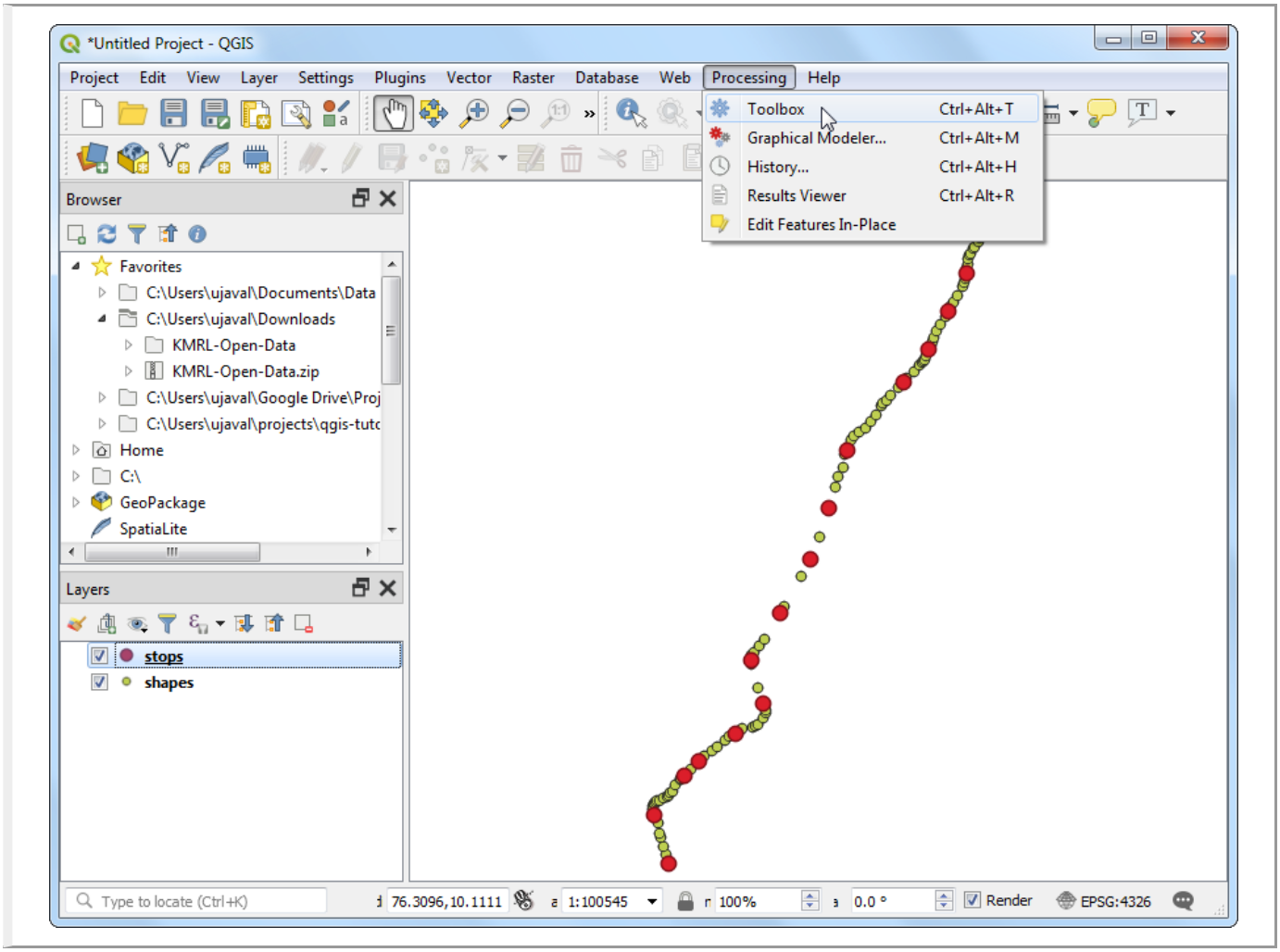
2. In the Data Source Manager dialog, switch to the Delimited Text tab. Click the ... button next to File name and browse to the `shapes.txt` file. Select `CSV (comma separated values)` as the File Format. The X field and Y field should be auto populated. Click Add.



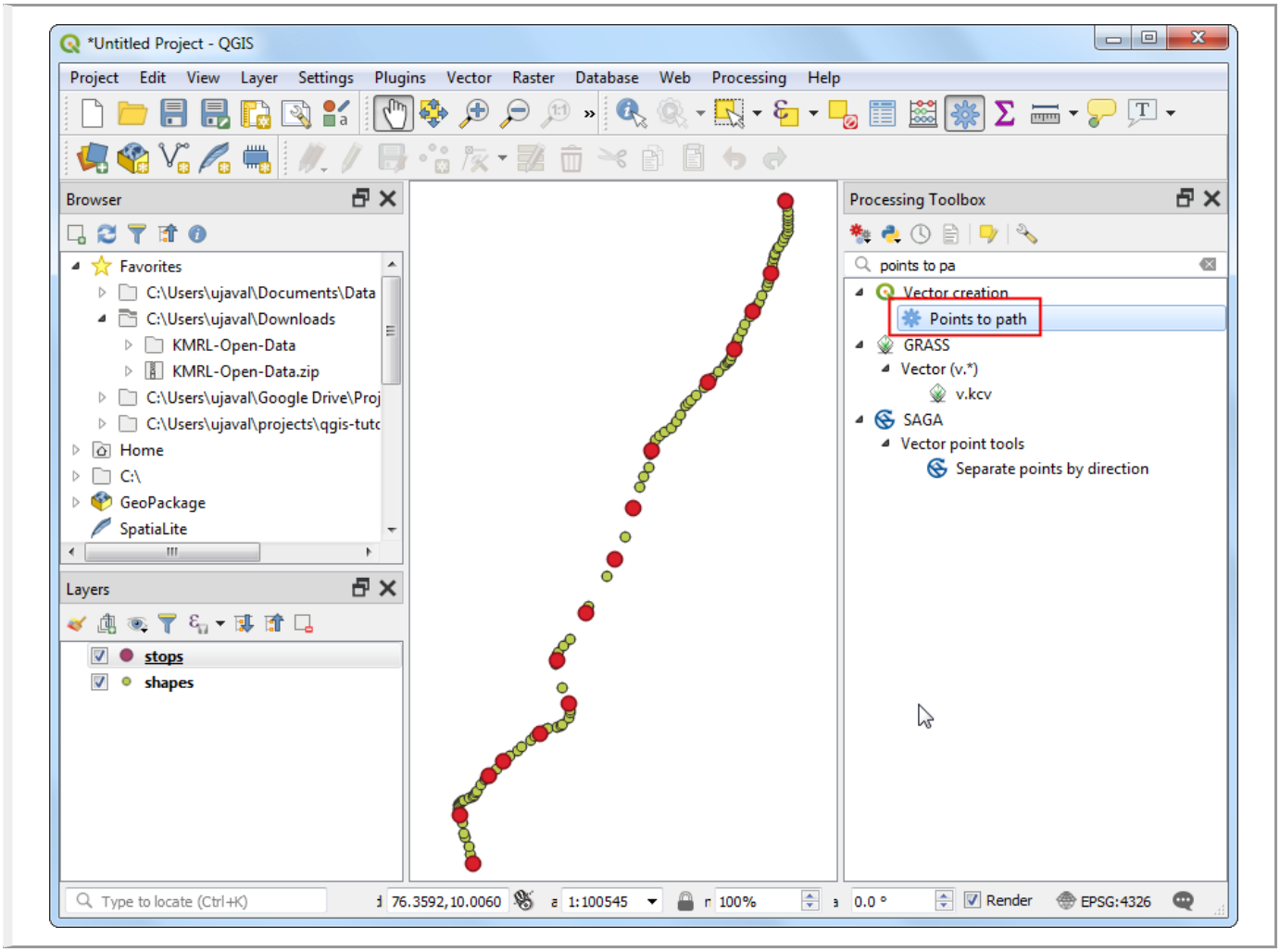
3. Similarly, click the ... button again and select `stops.txt` file. Click Add. Click Close.



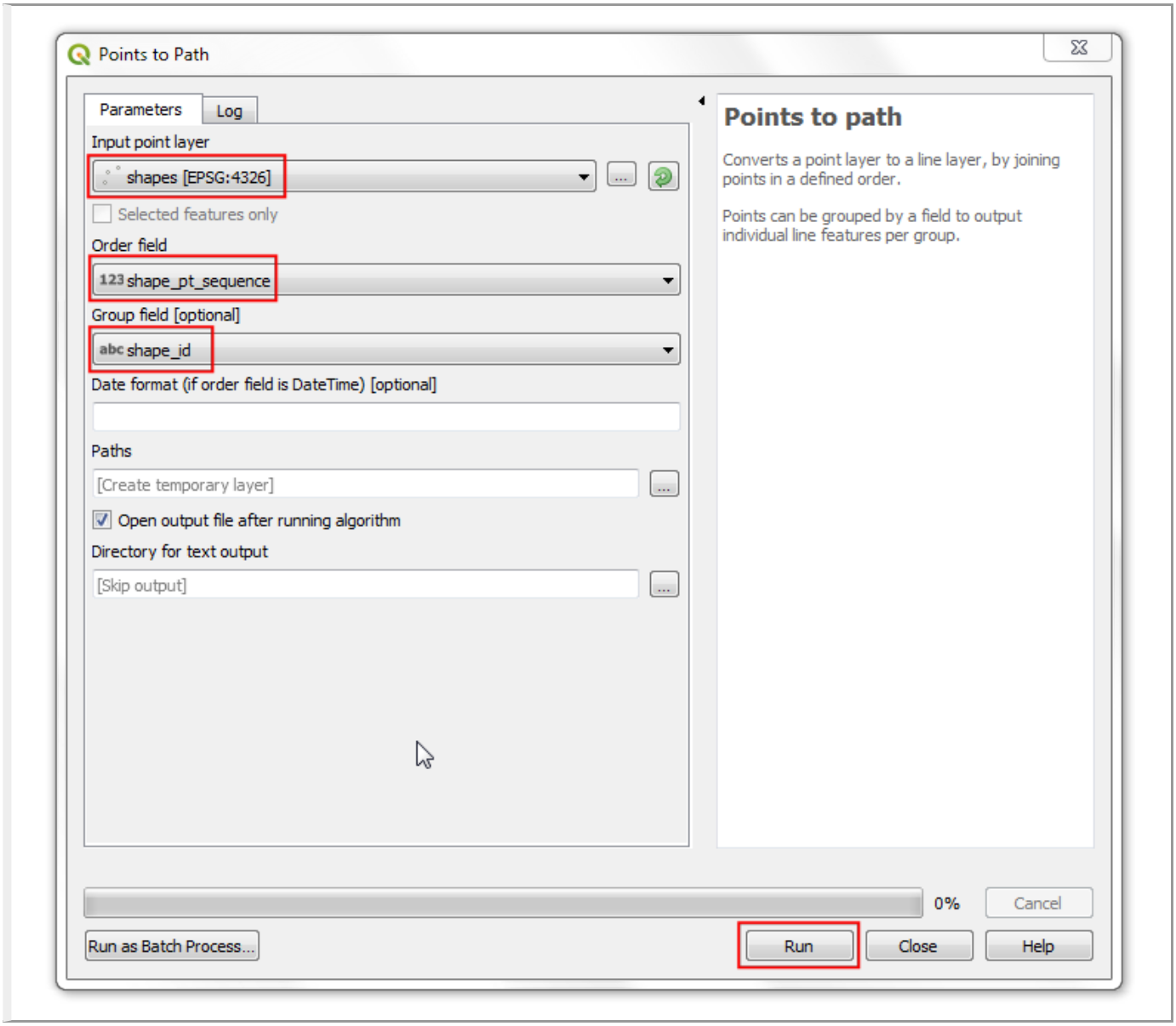
4. You will see 2 new layers `stops` and `shapes` added to the Layers panel. Let's convert the `shapes` point layer into a line layer representing the path of the metro line. Go to Processing ▸ Toolbox.



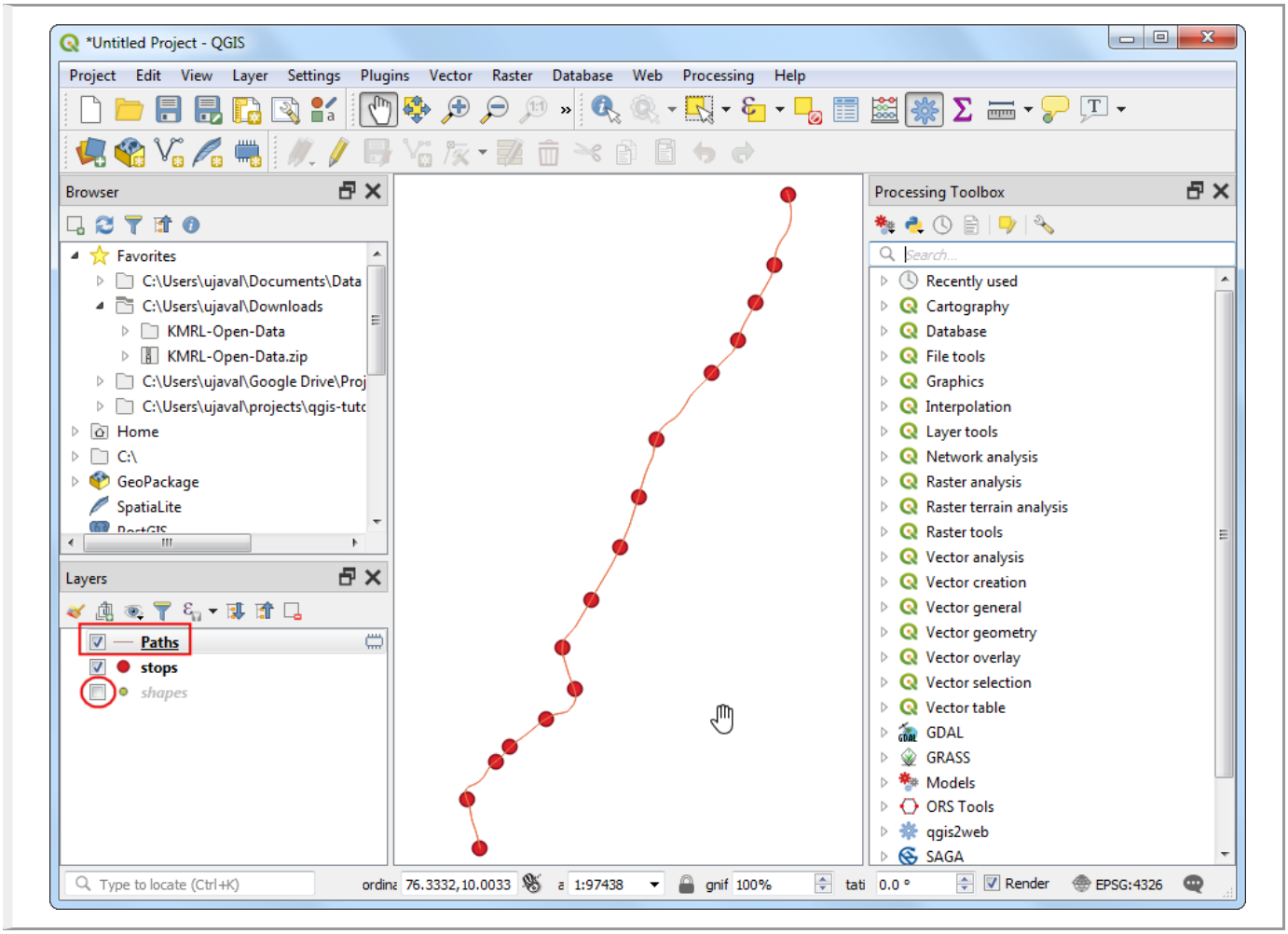
5. Search and locate the Vector creation ▸ Points to path tool. Double-click to launch it.



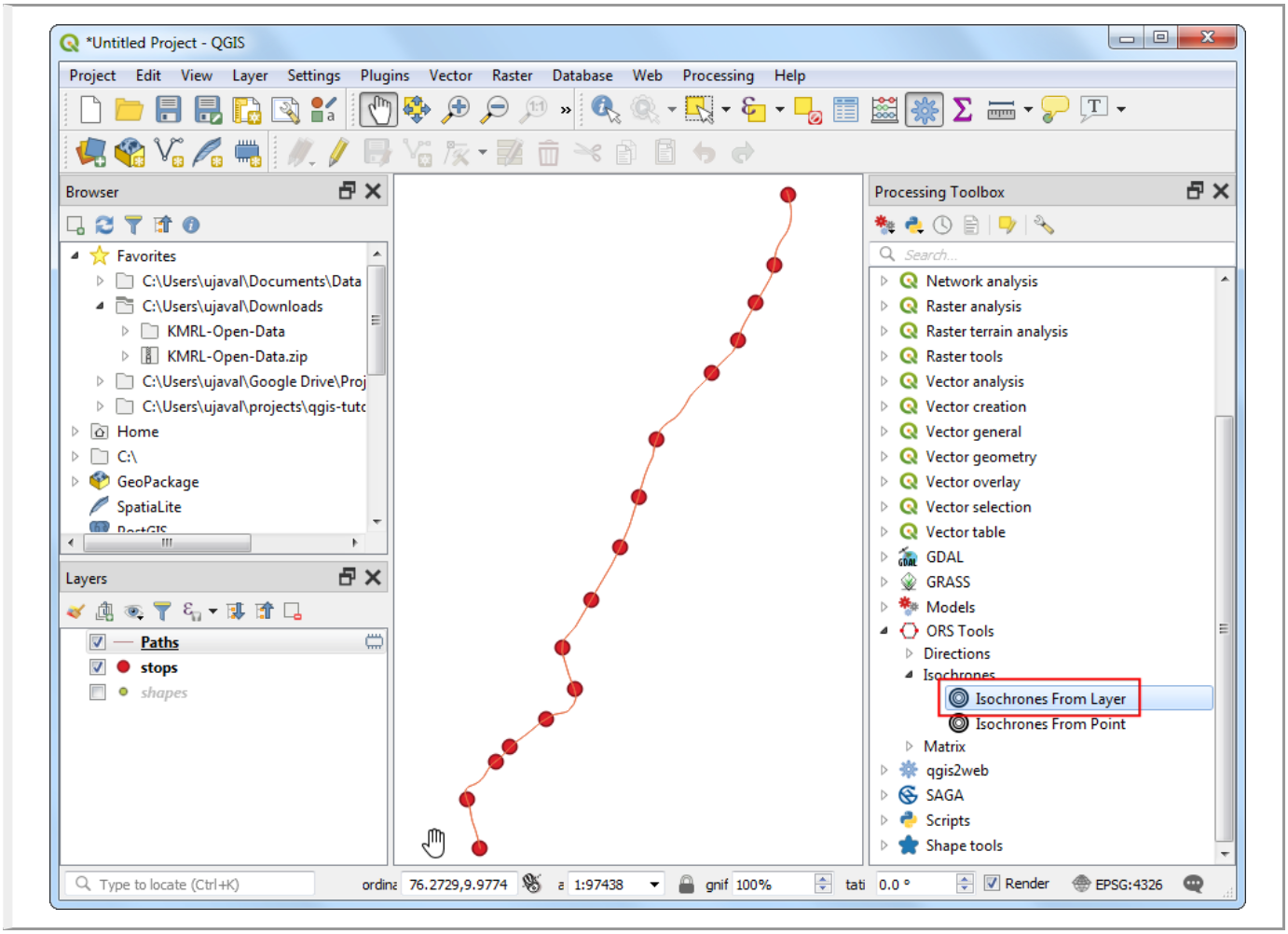
6. Select `shapes` as the Input point layer. As per GTFS specifications, each individual route has a unique `shape_id` so select that as the Group field. We can also specify the order of points that will form the line by selecting `shape_pt_sequence` as the Order field. Click Run.



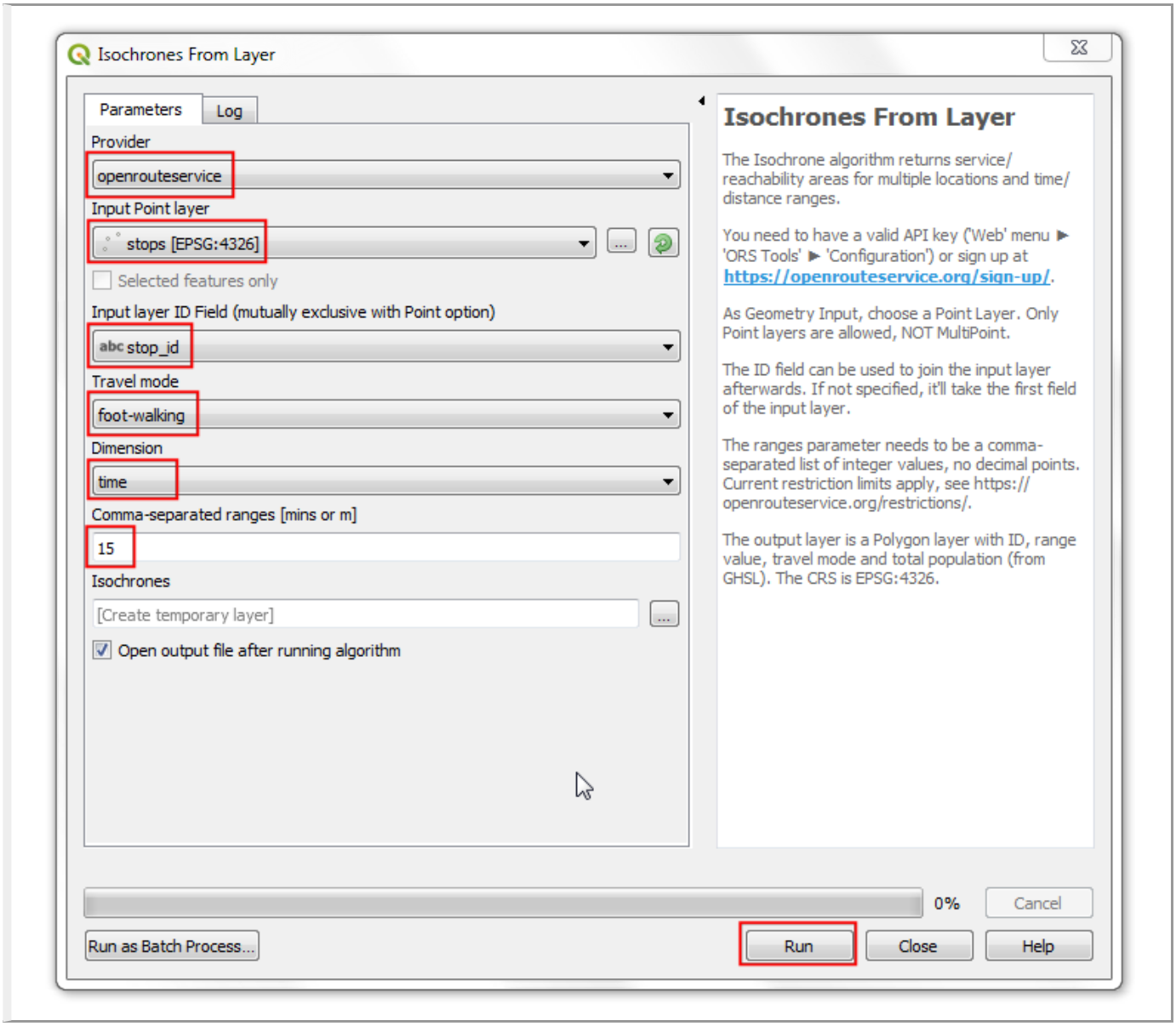
7. A new layer `Paths` will be added to the Layers panel. You can turn off the visibility of the `shapes` layer to see the newly added line layer.



8. Now that we have the metro stations and line data added, we are ready to start the network analysis. In the Processing Toolbox, search for and locate the ORS Tools ▸ Isochrones ▸ Isochrones From Layer tool. Double-click to launch it.



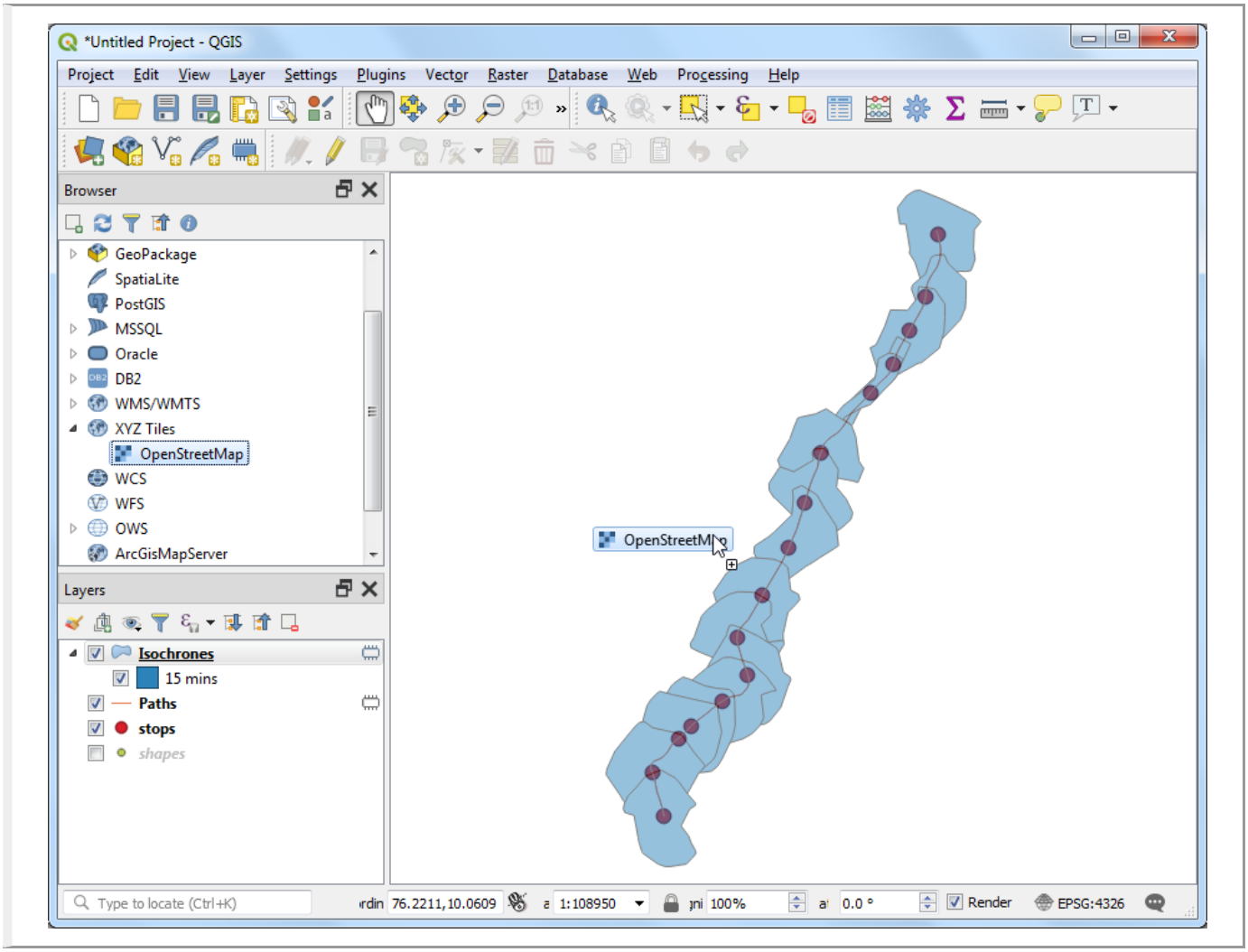
9. Select `openrouteservice` as the Provider. We will be computing a 15-min walking distance polygon from each metro station. Select `stops` as Input Point Layer. Select `stop_id` as the Input Layer ID Field. From the Travel mode drop-down, select `foot-walking` . As we are interested in time-based area, select `time` as the Dimension. Finally enter `15` minutes as the ranges. Click Run.



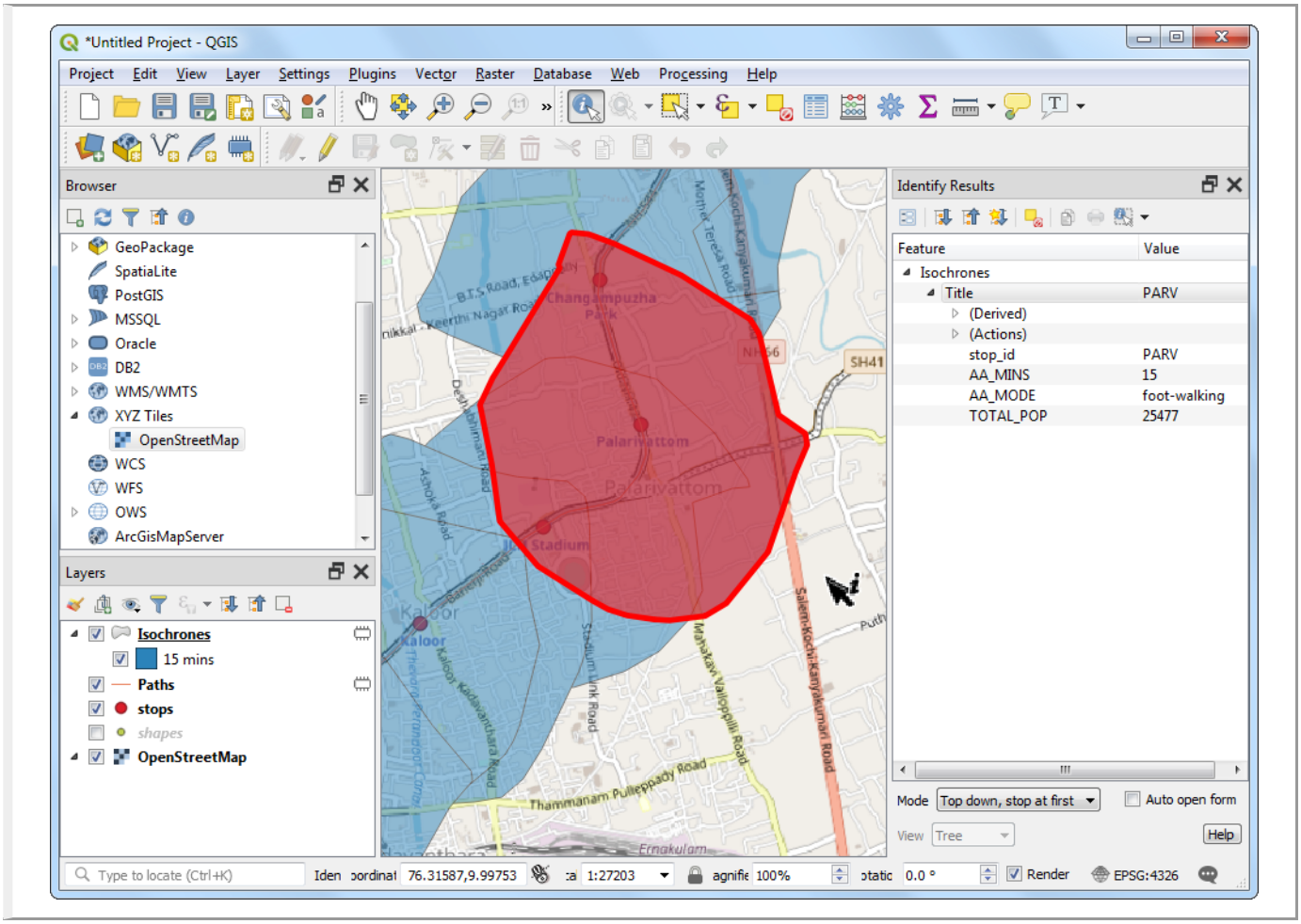
Note

Note that the Openrouteservice API has a limit of 20 requests per minute for Isochrones. So if your layer has more than 20 points, you may see errors indicating that the rate limit exceeded. You can keep the tool running and it will continue processing 20 points / min.

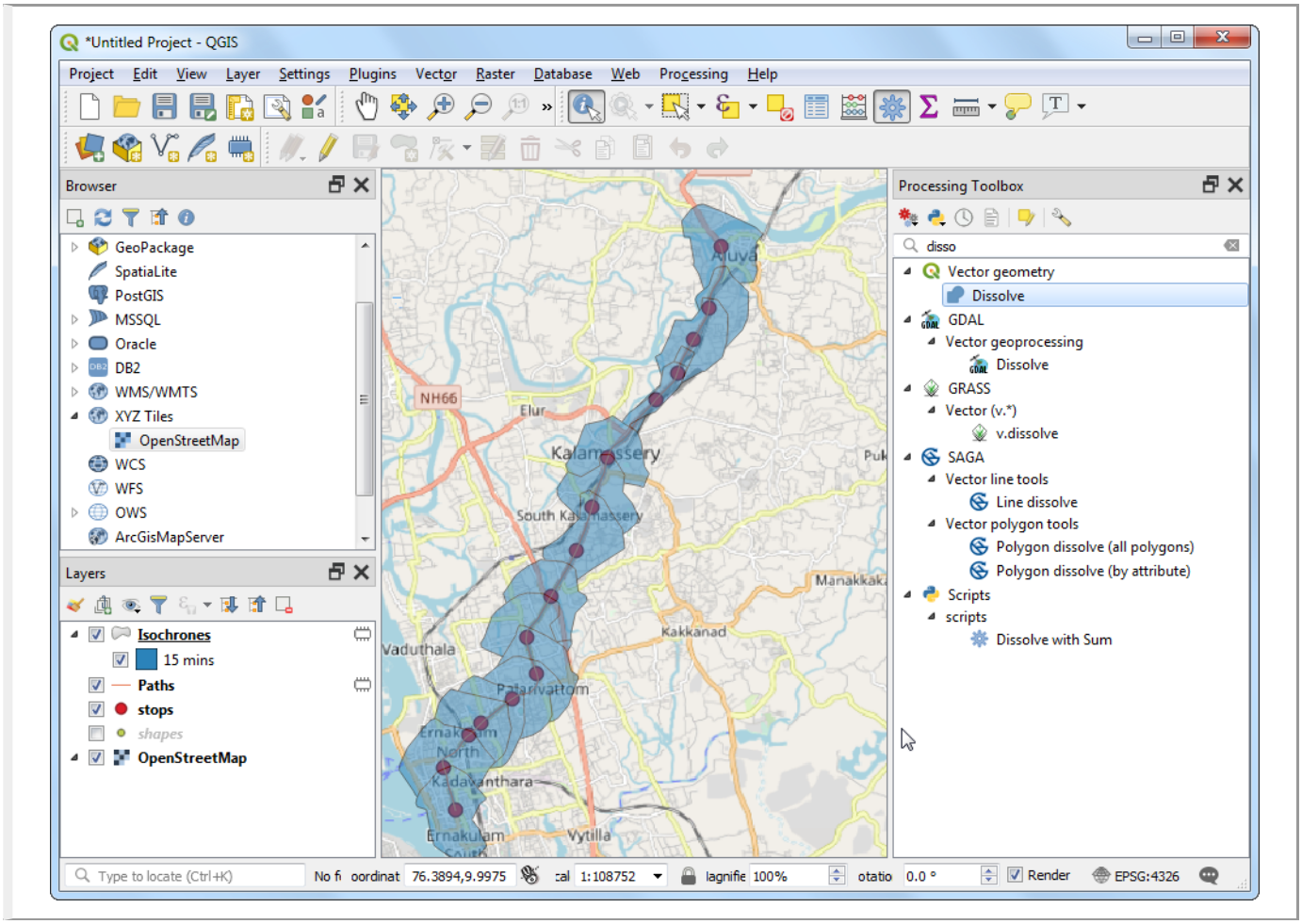
10. Once the tool finishes, you will see a new layer `Isochrones` loaded in the Layers panel. Each point has an associated polygon representing the area that is accessible within 15 minutes by walk. To see this in the context the data that was used to generate them, we can add the OpenStreetMap basemap. Scroll down the Browser panel and locate XYZ Tiles ‣ OpenStreetMap. Drag it to the canvas.



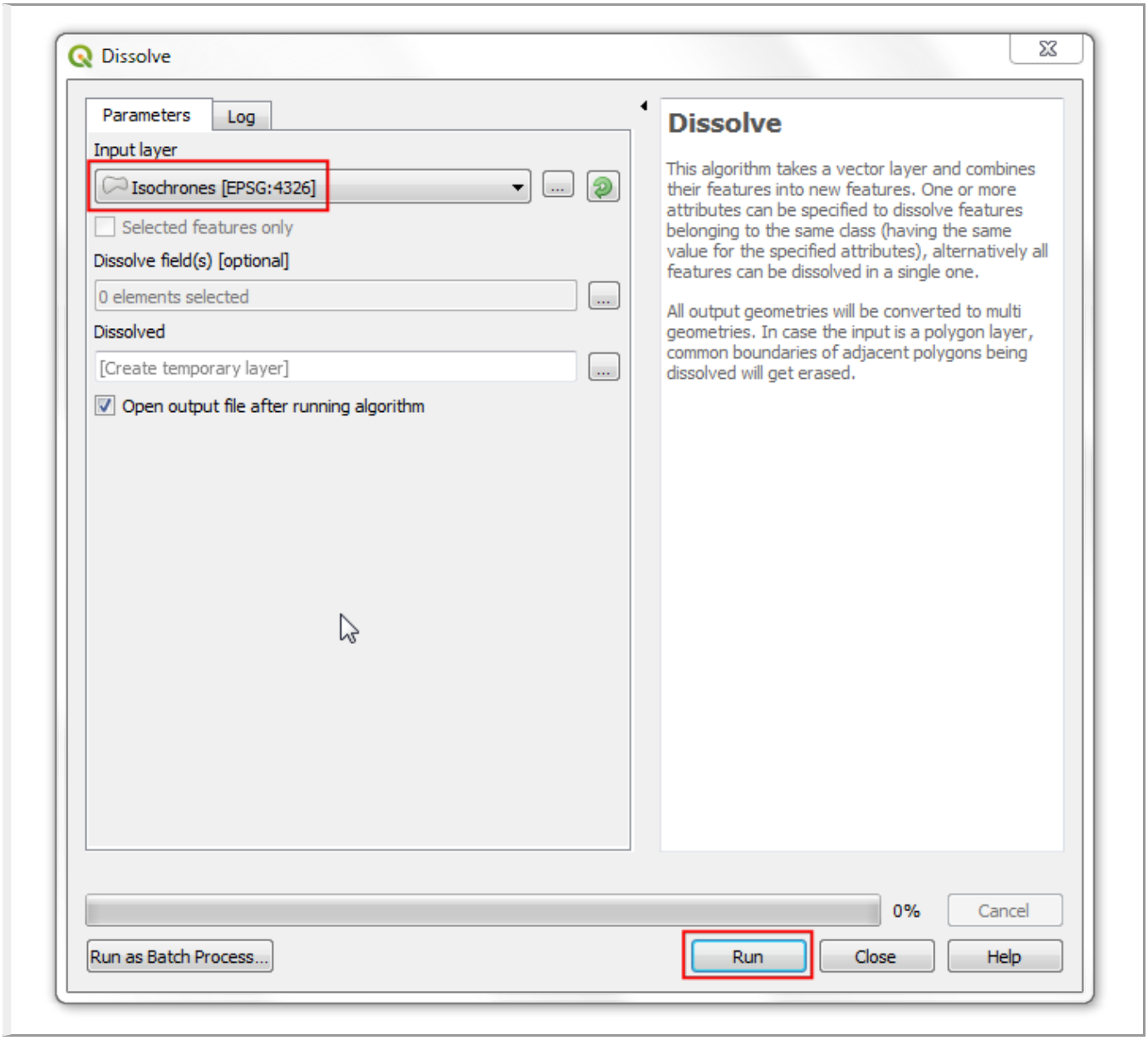
11. A new layer `OpenStreetMap` will be added to the Layers panel. Drag it down to change the layer order and keep it at the bottom of the layer stack. Zoom and pan to see if the results match the road network. You will see that the polygons are not circular, because the travel time is computed along roads, so the regions that have no roads will have lesser area covered.



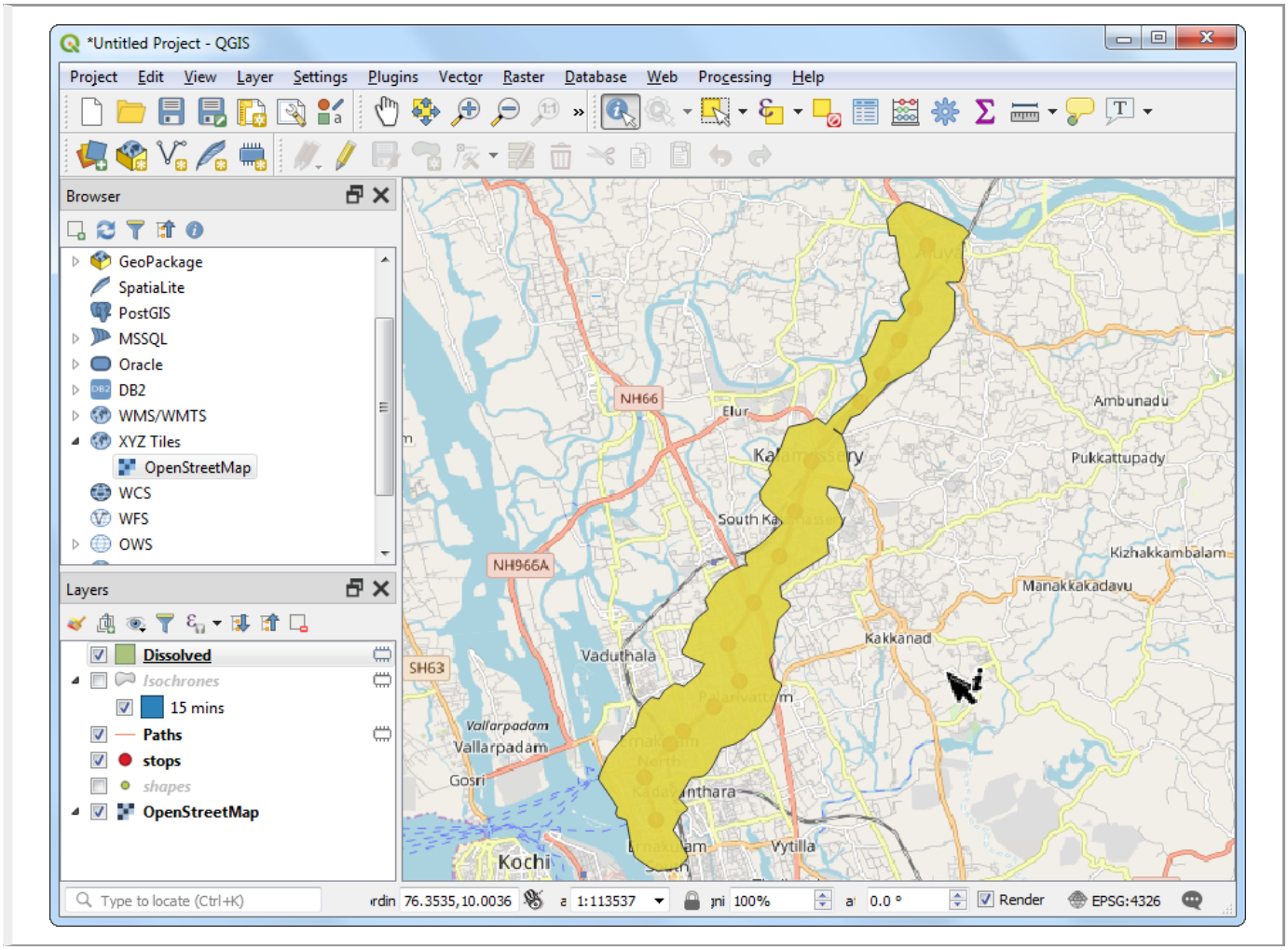
12. To compute the service area, we need to complete one last task. We can merge individual isochrone polygons to form a single polygon representing the areas that are accessible. Search for and locate Vector geometry ▸ Dissolve.



13. Select `Isochrones` as the Input layer and click Run.



14. Once the processing finishes, a new layer `Dissolved` will be added to the Layers panel. This polygons represents the full region that is accessible from the metro system within 15-minutes of walk.



Note

This is a simple example of how a service area analysis for a public transportation project can be done in QGIS. A more comprehensive service-area analysis for the metro system would include other modes of transport. We could include feeder buses, nearby bus stops and routes serving those bus stops to expand the analysis. We may also include travel by other modes such as car and taxi.

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Using the QGIS Browser

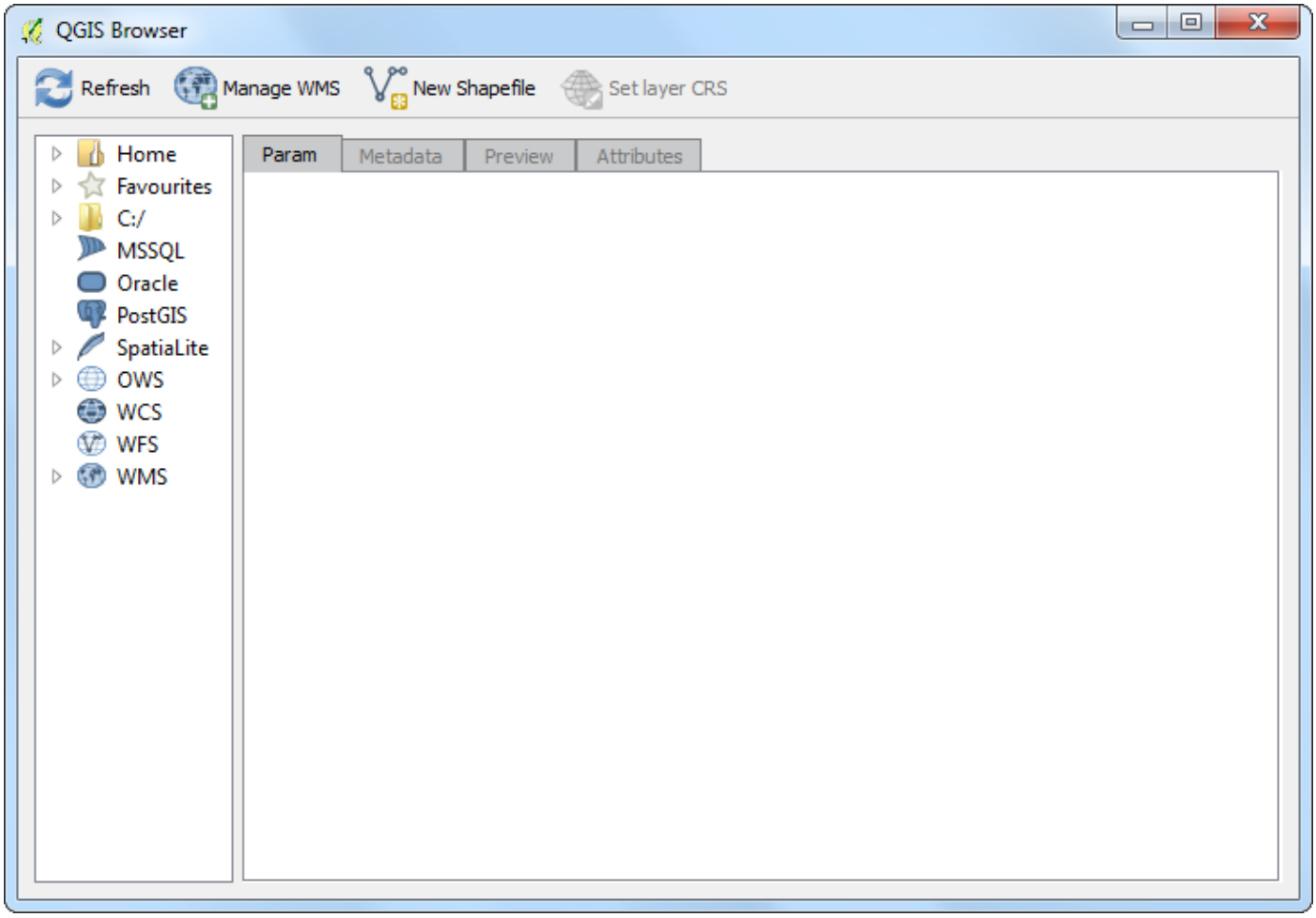
QGIS comes with a standalone application called **QGIS Browser**. This is a useful companion tool to QGIS and helpful in managing GIS datasets. ArcGIS users may think of it as an application similar to ArcCatalog.

Locating the QGIS Browser

QGIS Browser Standalone Application

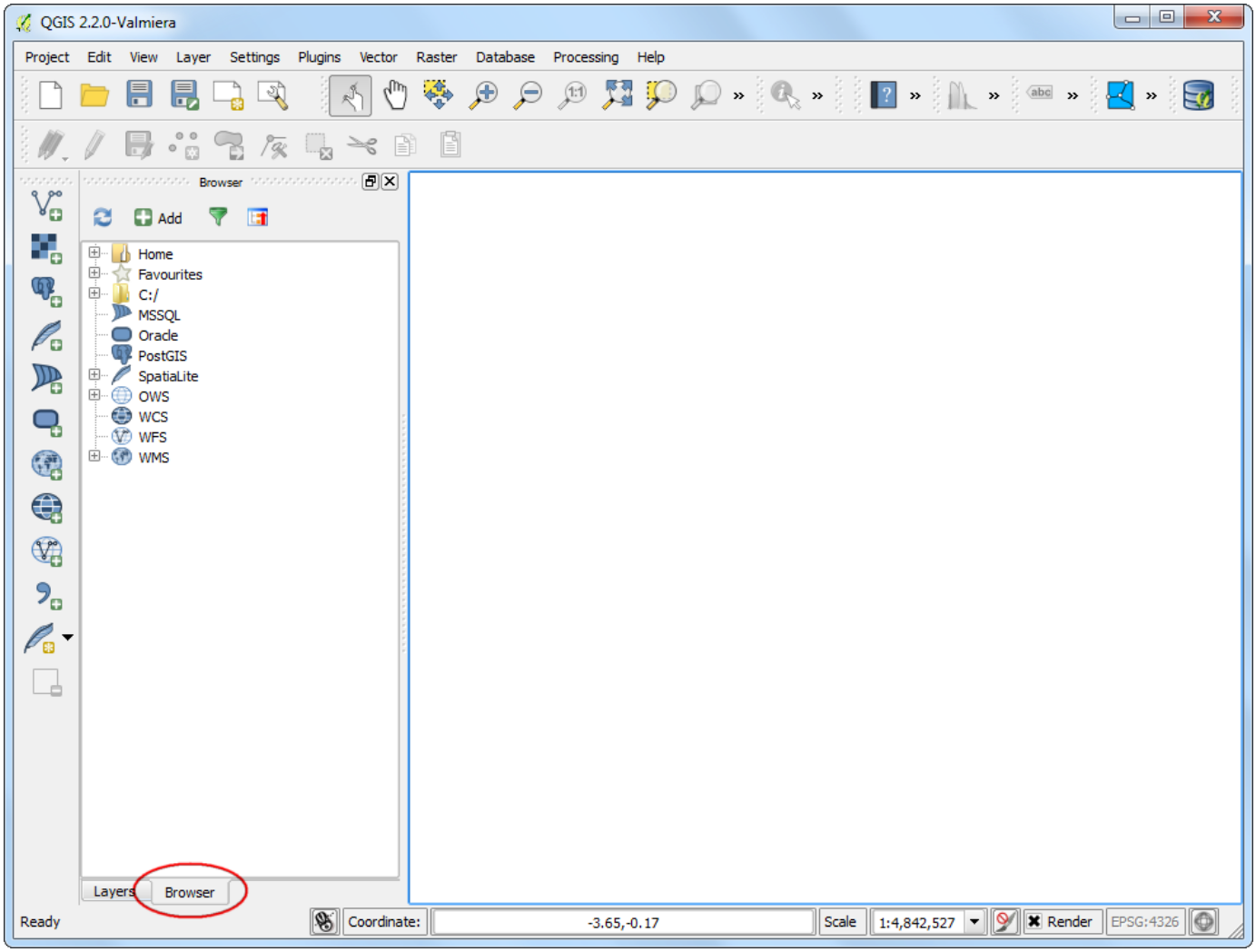
QGIS Browser is part of the standard install of QGIS.

- Windows: If you installed QGIS via OSGEO4W installer, you will see `QGIS Browser` in your start menu.
- Mac: The application is located at `QGIS.app/Contents/MacOS/bin/QGIS Browser.app` . You can create a symlink to this app. Navigate to the Application folder, right-click the QGIS icon and select Show Package Contents. Browse to MacOS ▸ bin ▸ QGIS Browser. Right-click the `QGIS Browser` icon and select Make Alias. Drag the `QGIS Browser alias` to the Applications folder. Now you can access the `QGIS Browser` like any other application.
- Linux: You can launch the QGIS browser by the command `qbrowser` . It is located in the same directory as the qgis application.



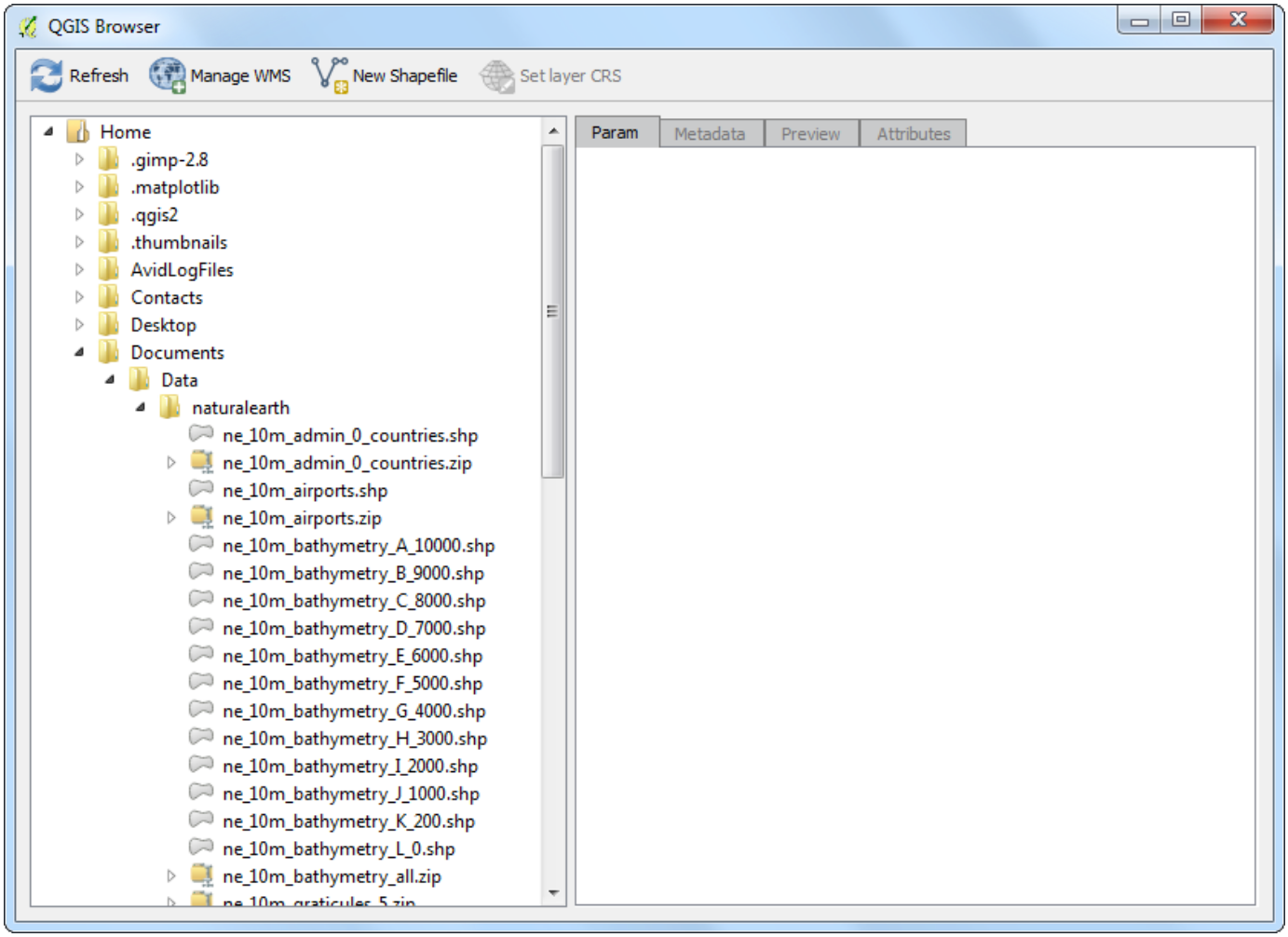
Browser Panel in QGIS

A convenient way to access the QGIS Browser is from within the main QGIS Desktop application itself. The browser panel is located at the bottom of the left-hand panel in QGIS. Click on the Browser tab to open the QGIS Browser. If you do not see the Browser tab, enable it by doing to View ▸ Panels ▸ Browser (Windows and Mac) or Settings ▸ Panels ▸ Browser (Linux).

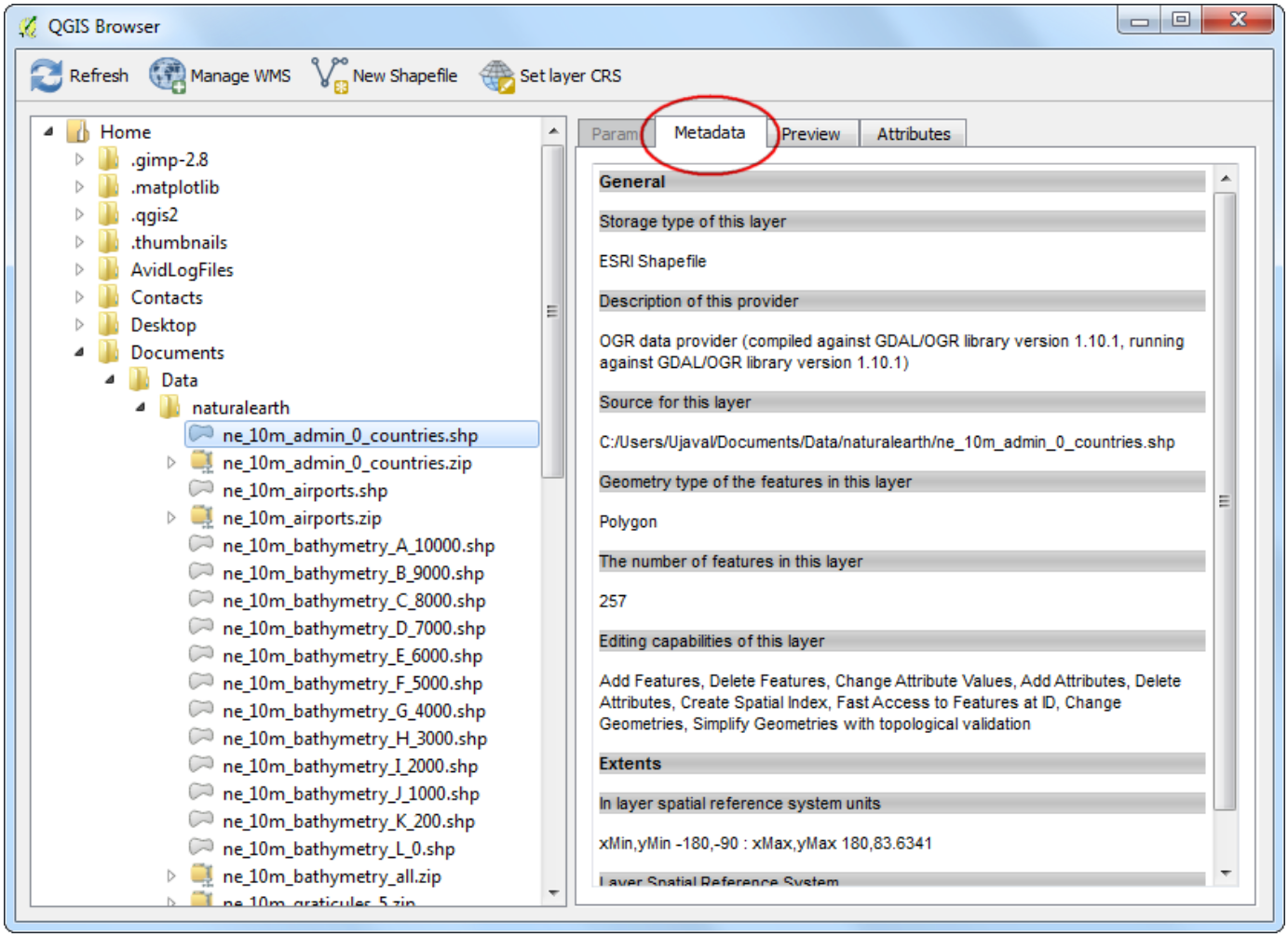


Procedure

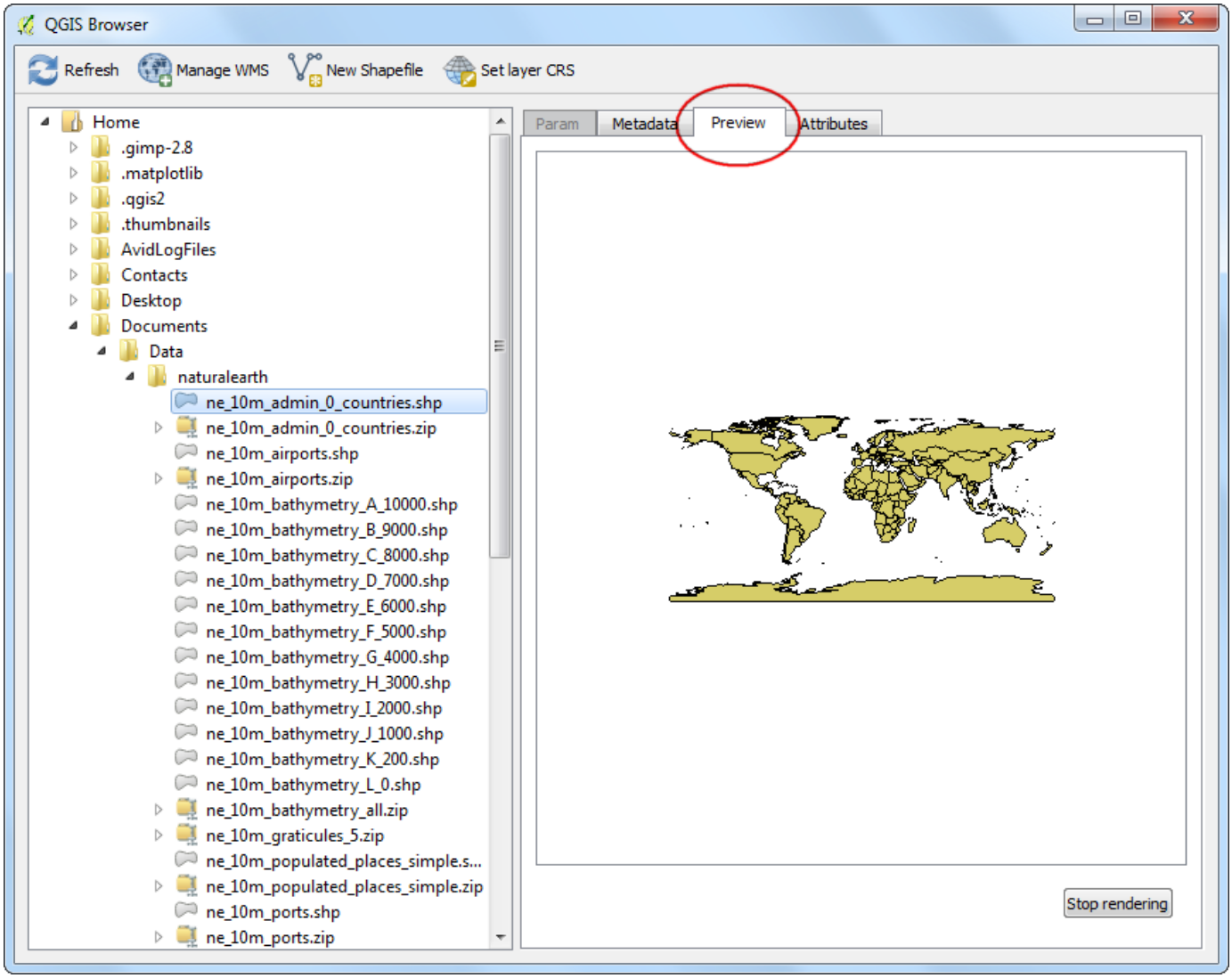
1. Now let us explore some features of the QGIS Browser. Switch to the standalone QGIS Browser application. Browse to a directory on your system where you have some GIS data. You will immediately notice the advantage of using the Browser. Instead of seeing all support files and non-spatial data, you see only the spatial layers that are supported by QGIS. Click on a layer to select it.



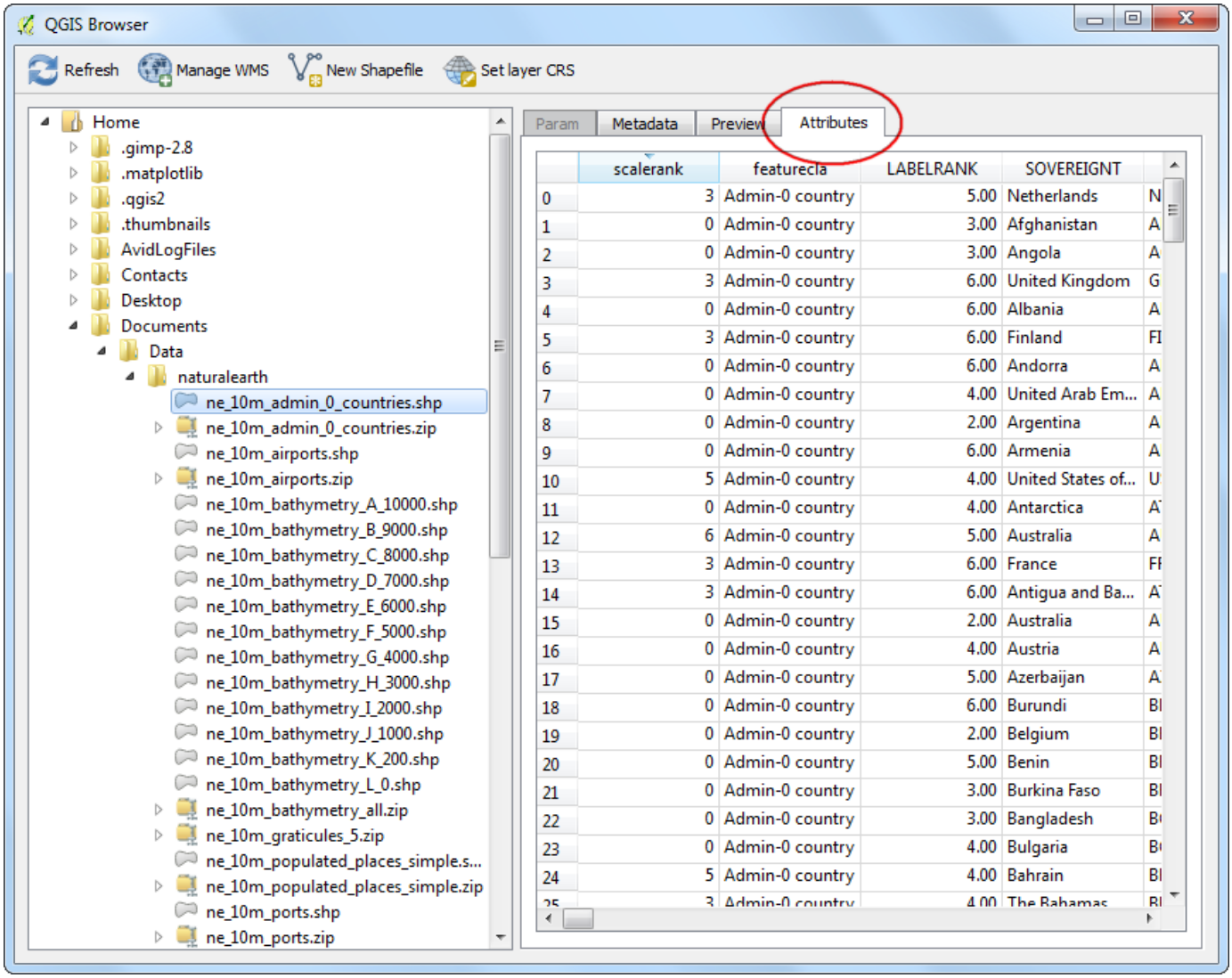
2. As you select a layer, you will see the Metadata in the first tab on the right-hand panel. You can quickly gather basic information about the dataset from this panel, such as number of features, projection etc.



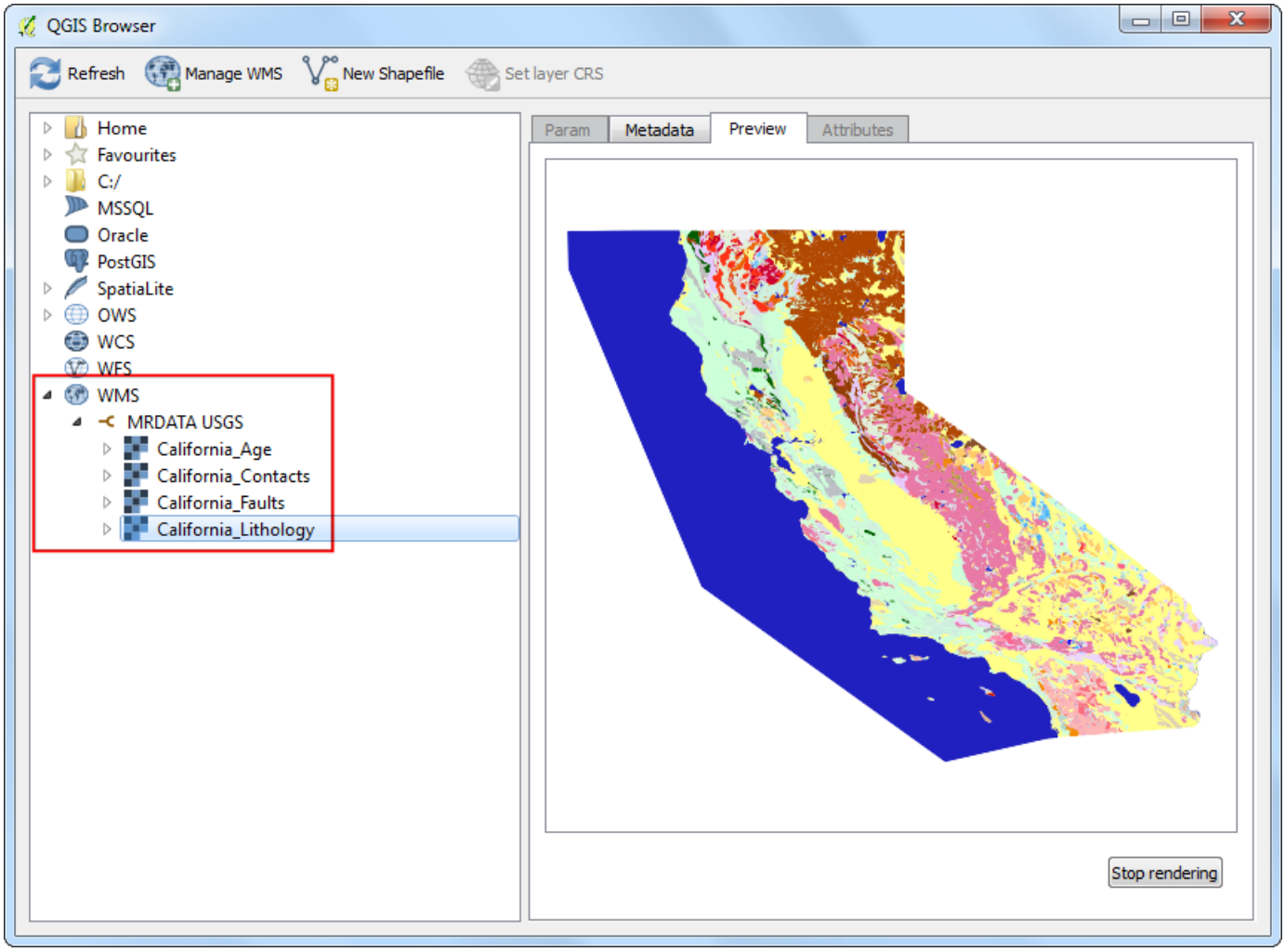
3. If you switch to the Preview tab, you will a preview of the dataset. This is a quick way to determine how the dataset looks before opening it in QGIS.



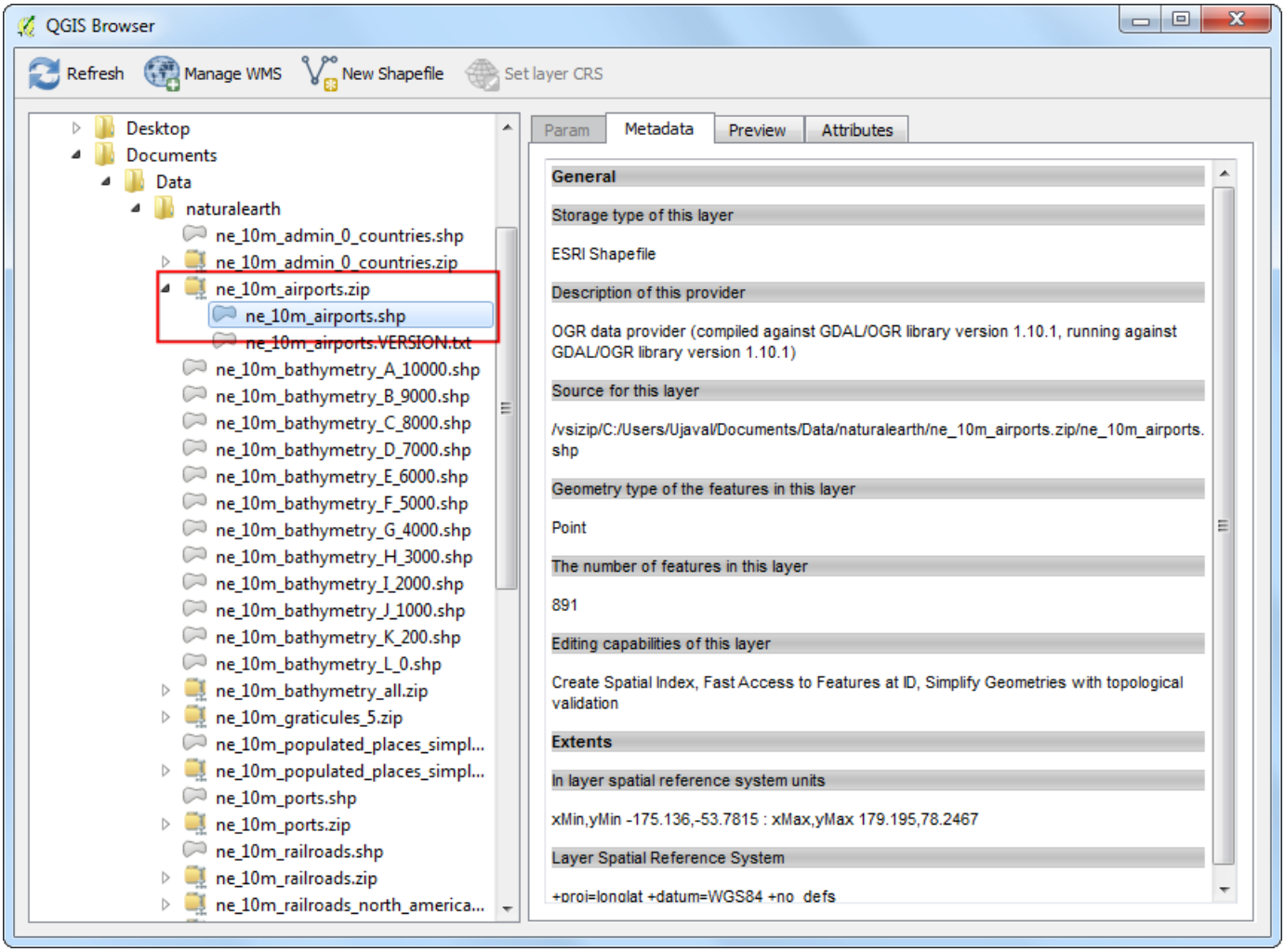
4. The last tab is the Attributes tab. Here you can see the attribute table of the dataset to get an idea of the fields available and their values.



5. The QGIS Browser not only gives you access to vector and imagery layers on your system, but also databases and network resources. If you use any online data via WMS, you can quickly preview it within the browser. Just expand the WMS location and you will see the resources you have setup. Similarly, if you have PostGIS, SpatialLite or MSSQL databases available, you can access those as well.



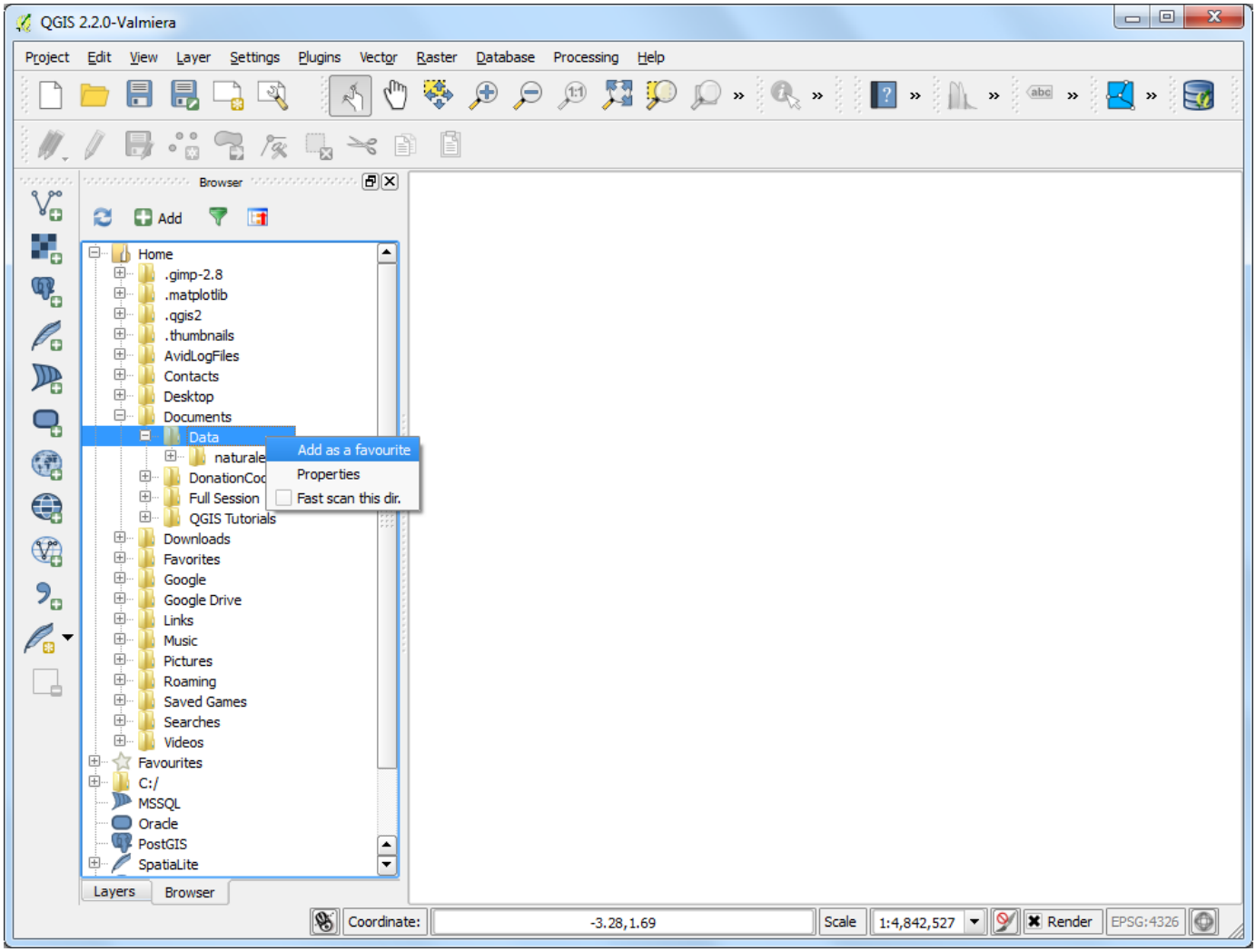
6. QGIS Browser has the ability to browse and open zip files directly. Navigate to any folder containing zip files. You will see that the zip files also appear as a supported dataset and you can preview it just like any other dataset.



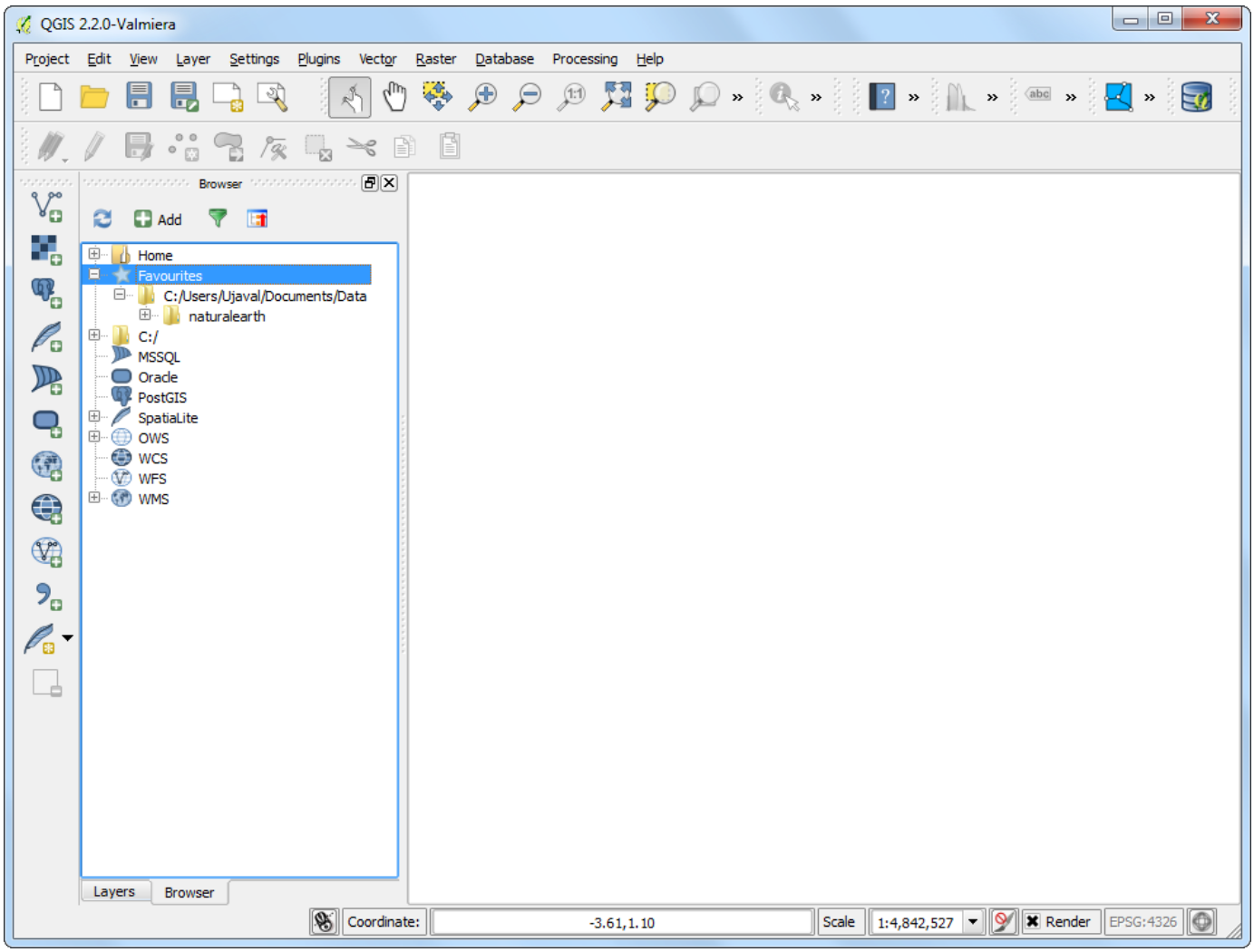
7. Another useful feature is to add certain folders in your system as Favorites. Right-click any folder and select Add as a favorite.

Note

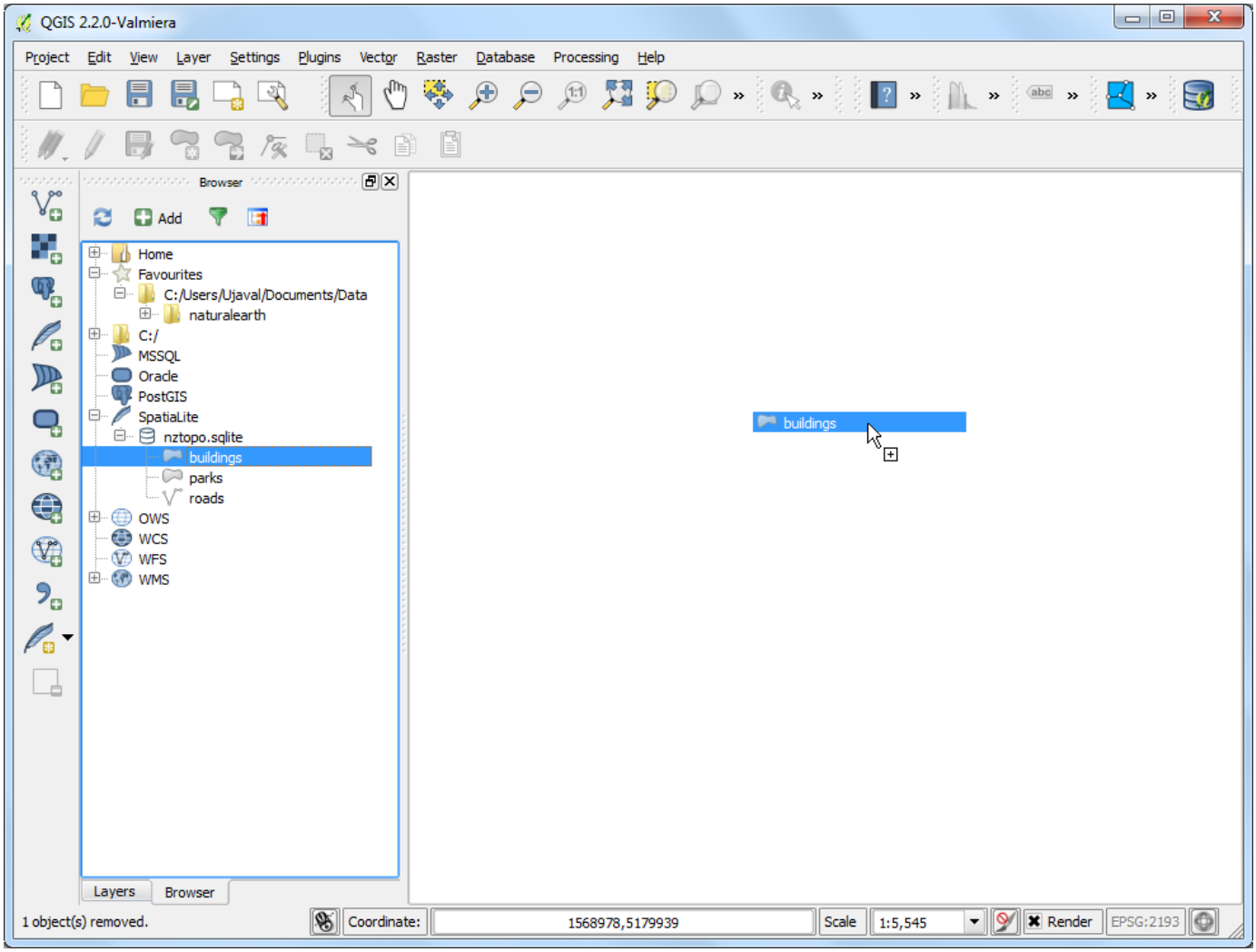
Adding a folder to your favorites list currently works only from the Browser panel in QGIS. This feature is not available in the standalone application.



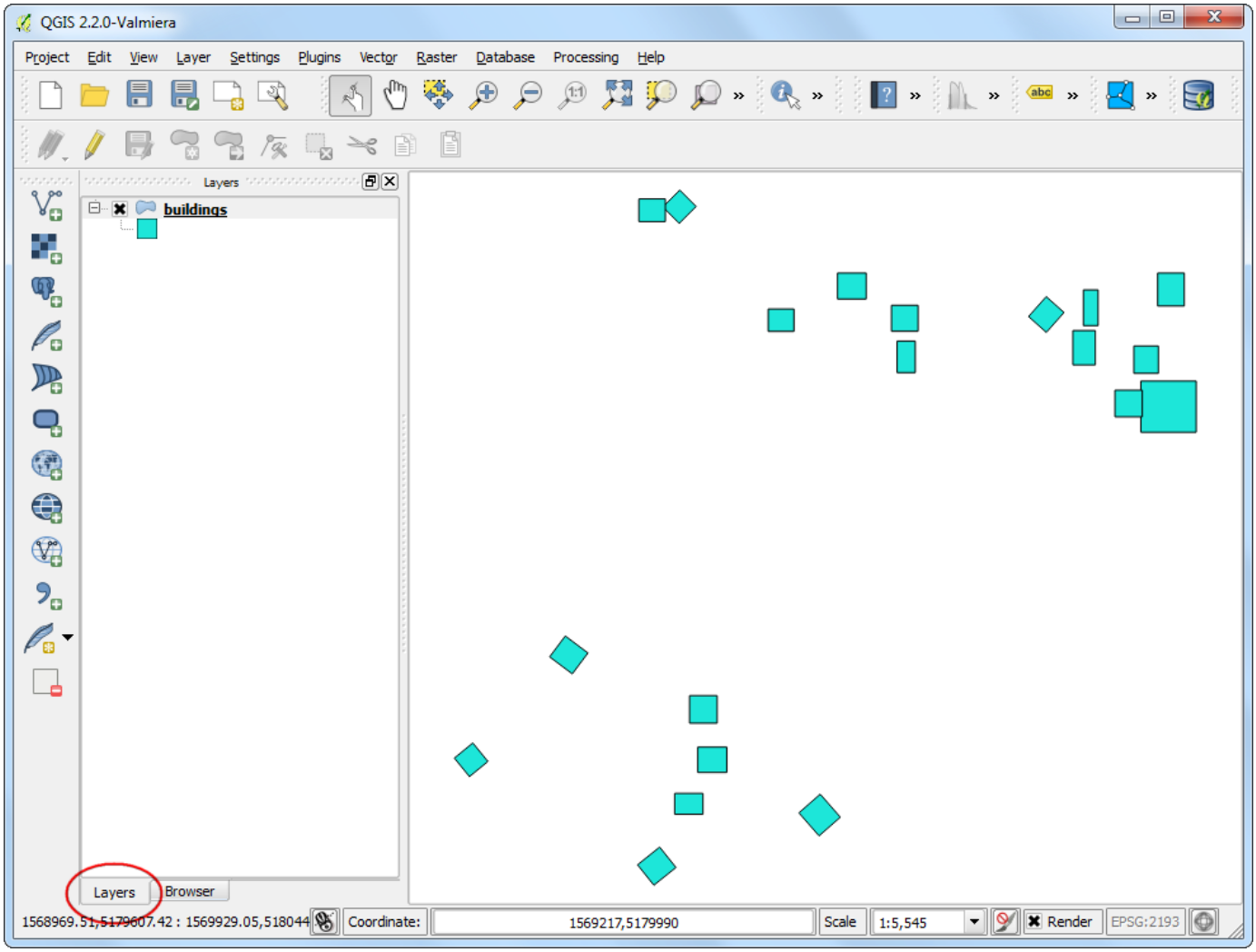
8. After adding the location as a favorite, it can be quickly accessed from the Favorites folder in the browser.



9. Once you have selected the layer, you can double-click it to add it to the QGIS canvas. You can also drag-and-drop the layer to the QGIS Canvas.



10. You can switch back to the Layers panel from the bottom of the left-hand panel in QGIS to view the added layer.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Open BIL, BIP or BSQ files in QGIS

When dealing with remote sensing and scientific datasets, one often comes across data in formats like **BIL**, **BIP** or **BSQ**. The GDAL library (http://www.gdal.org) - which is used by QGIS to read raster files - has support for these formats, but it cannot open these files by itself. We will go through the process of creating support files so these formats can be read by QGIS.

Band interleaved by line (BIL), band interleaved by pixel (BIP), and band sequential (BSQ) are common methods of organizing image data for multiband images. (Read more about these formats (http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=BIL,_BIP,_and_BSQ_raster_files))

Typically, these files are accompanies by a `.hdr` file. If your dataset came with a `.hdr` file, make sure the root name of the .bil, .bsq or .bip file and the .hdf files match and they are in the same directory. For example, if the file is called `image.bil` , the associated file should be named `image.hdr` and present in the same directory as the `image.bil` file. Then when you go to Layer ‣ Add Raster Layer, select the `image.bil` file and it will open without problems.

Many a times, the files do not come with an associated `.hdr` file. In such cases, you must create this file by hand as shown in this tutorial.

Get the data

We will use the AVHRR Global Land Cover Classification data (http://glcf.umd.edu/data/landcover/data.shtml) from Global Land Cover Facility (http://glcf.umd.edu/) as an example.

The Global Coverage datasets are distributed as **BSQ** files. Download the 1 Degree pixel resolution (ftp://ftp.glcf.umd.edu/glcf/Global_Land_Cover/Global/1deg/gl-latlong-1deg-landcover.bsq.gz) dataset.

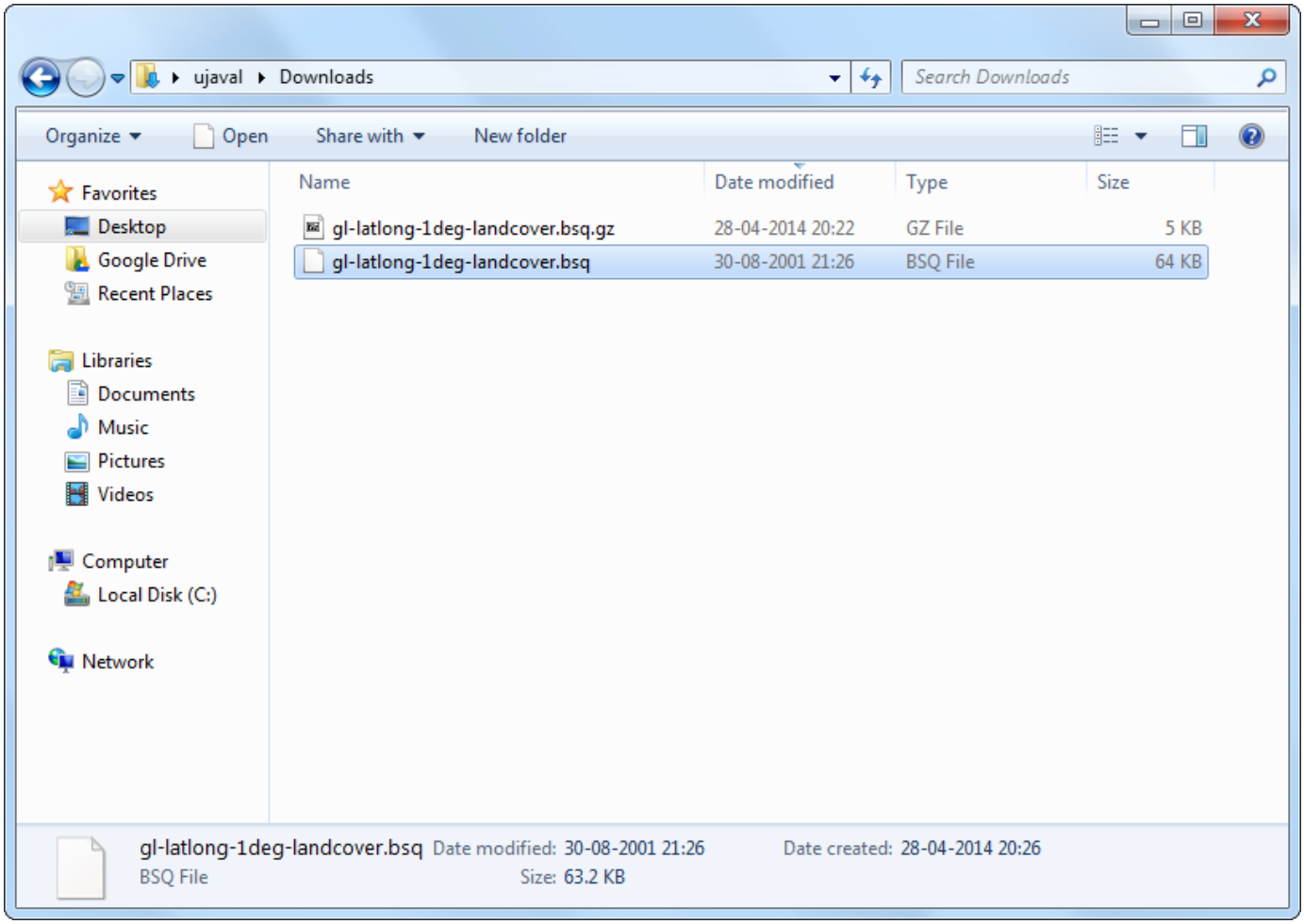
For convenience, you may directly download a copy of the dataset from the link below:

gl-latlong-1deg-landcover.bsq.gz (http://www.qgistutorials.com/downloads/gl-latlong-1deg-landcover.bsq.gz)

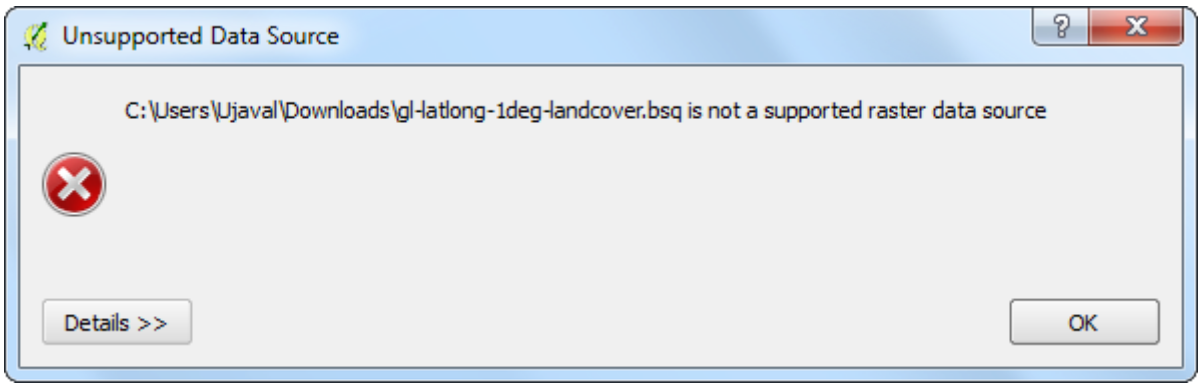
Data Source [GLCF] (credits.html#glcf)

Procedure

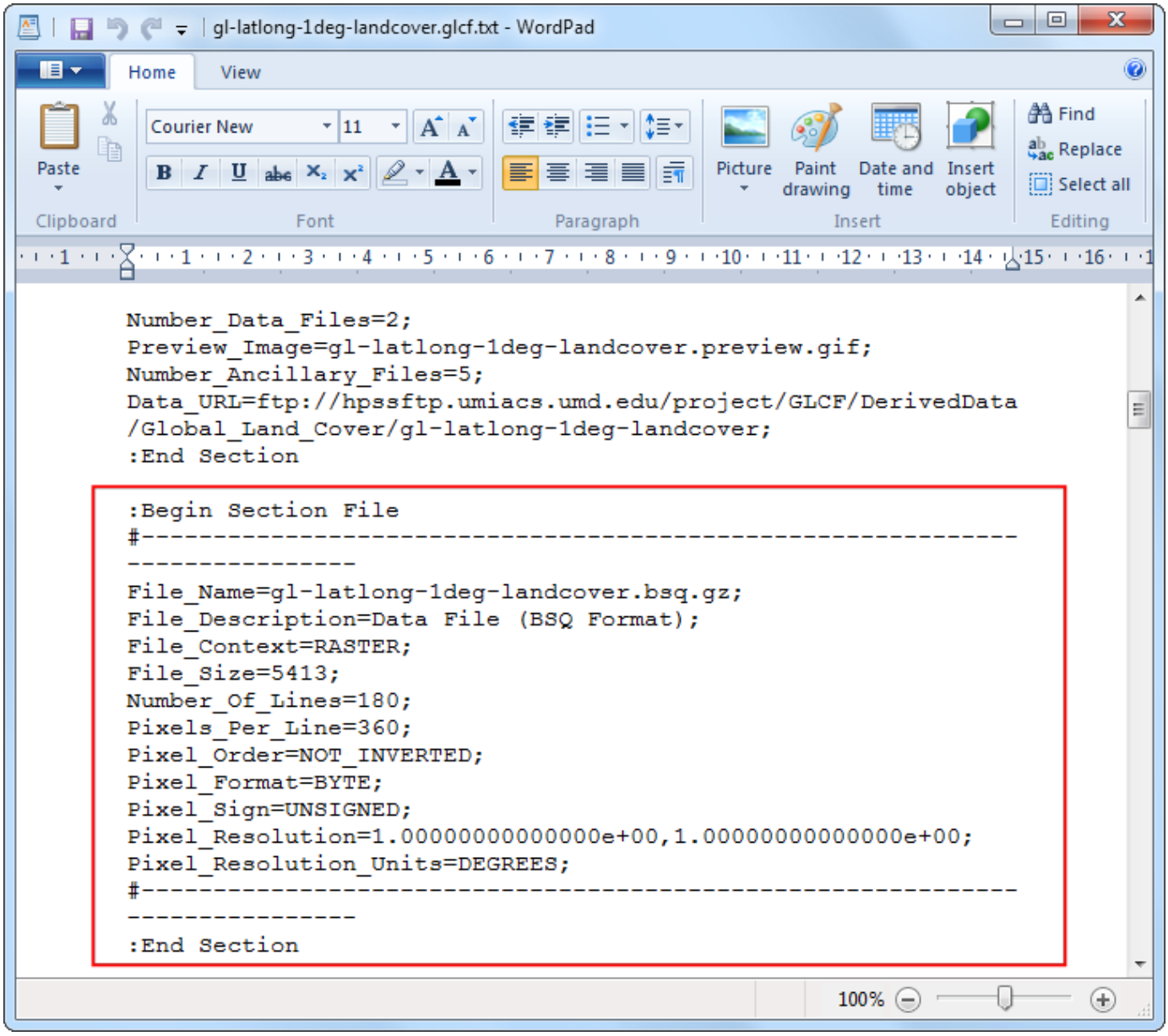
1. Unzip and extract the .bsq file. On Windows, you may use the excellent 7-Zip utility (http://www.7-zip.org/) to read and extract .gz file. You will see that you only have a .bsq file named `gl-latlong-1deg-landcover.bsq` . There is no hdr file.



2. Note that if you try to open the `gl-latlong-1deg-landcover.bsq` file in QGIS as it is, you will get an error message.



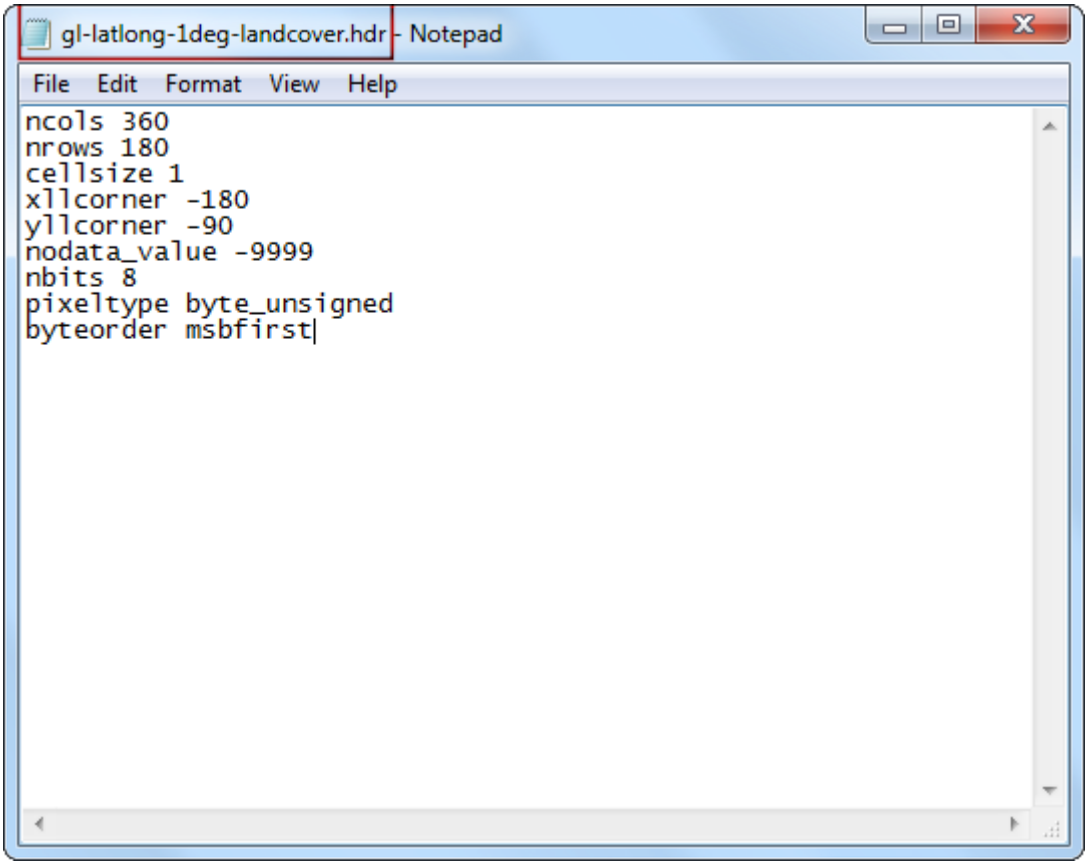
3. To overcome this error, we will create a header file with `.hdr` extension. The header file contains information about the dataset and how it is organized. Usually, this information is supplied as part of Metadata for the dataset. If you do not have the metadata, look at the website or documentation for clues. Some of the information can be guessed if you do not know it. In case of this dataset, the data download page links to the metadata (ftp://ftp.glcf.umd.edu/glcf/Global_Land_Cover/Global/1deg/gl-latlong-1deg-landcover.glcf). Download the metadata and open it.



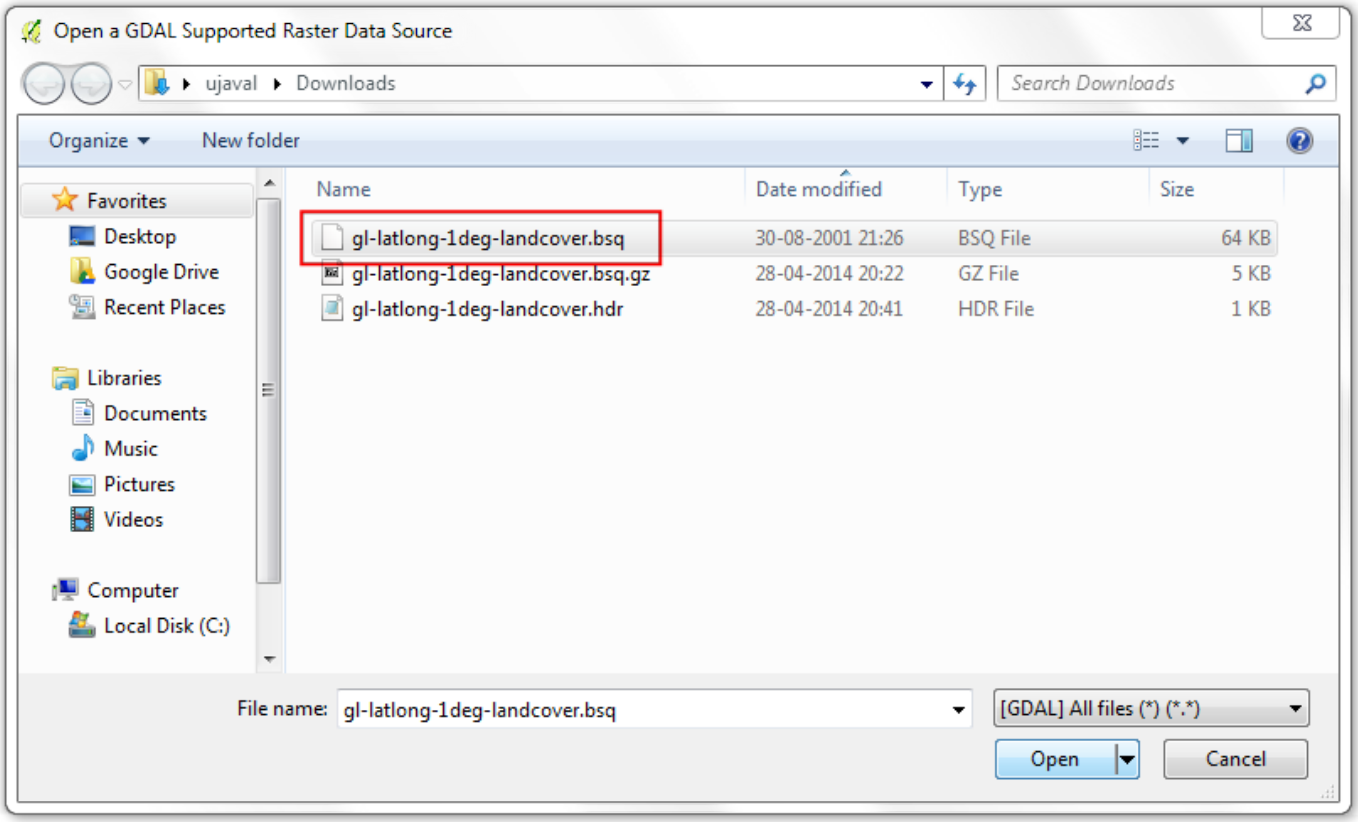
4. The .hdr file needs to be a plain text file in the following format. Some of these parameters are given to us and some needs to be worked out. Learn more about the format (http://www.gdal.org/frmt_various.html).

```
ncols <number of columns or width of the raster>
nrows <number of rows or height of the raster>
cellsize <pixel size or resolution>
xllcorner <X coordinate of lower-left corner of the raster>
yllcorner <Y coordinate of the lower-left corner of the raster>
nodata_value <pixel value to be ignored>
nbits <number of bits per pixel>
pixeltype <type of values stored in a pixel, typically float or integer>
byteorder <byte order in which image pixel values are stored, msb or lsb>
```

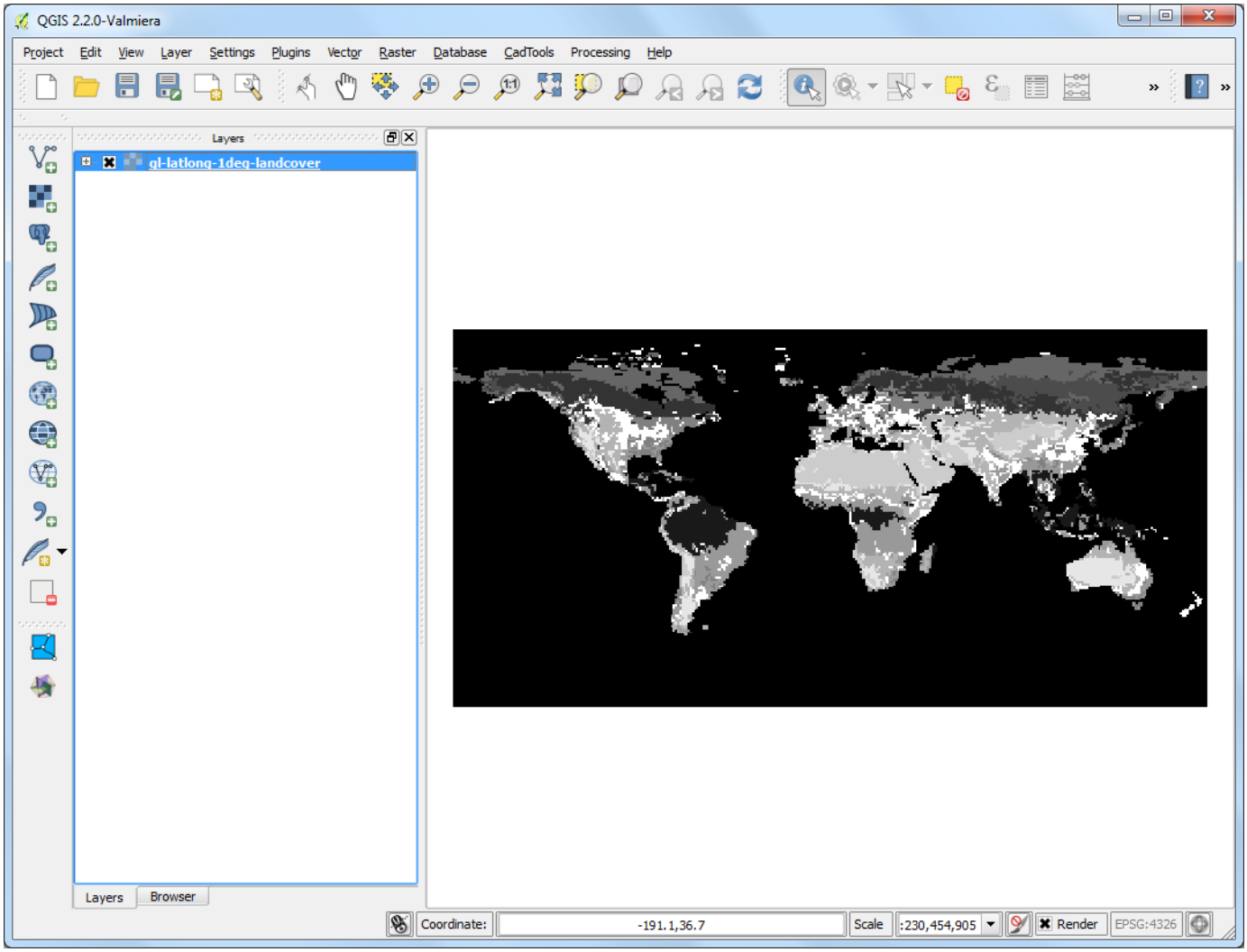
5. Open a text editor and create a file in the format specified in the previous step. Save the file as `gl-latlong-1deg-landcover.hdr`. Make sure the file doesn't have `.txt` at the end. Some of the values in the text files are easy to understand. The **ncols** and **nrows** come from the metadata as the Number of Lines and Number of Pixels per Line. The **cellsize** is 1 as the Pixel resolution from the metadata. The X,Y coordinate of lower-left corner needs to be worked out by us. Since the file covers the entire world and units are lat/long, **xllcorner** and **yllcorner** are -180 and -90 respectively. We do not have any information about the nodata_value, so -9999 is a safe bet. From metadata again, Pixel Format is Byte, so **nbits** will equal to 8 and pixeltype will be **byte_unsigned**. We do not have information about the byteorder, so leave it as msbfirst. You may download the correctly formatted HDR file from here (../../downloads/gl-latlong-1deg-landcover.hdr).



6. Now that you have the header file, put it in the same directory as `gl-latlong-1deg-landcover.bsq`. Then in QGIS, go to Layer ‣ Add Raster Layer. Select `gl-latlong-1deg-landcover.bsq` as your input and click Open.



7. In the next screen, you may be prompted to choose a CRS. Since the data is in Lat/Long, choose **WGS84 EPSG:4326** as your CRS. Now you will see the dataset loaded in QGIS.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Getting Started With Python Programming (QGIS3)

QGIS has a powerful programming interface that allows you to extend the core functionality of the software as well as write scripts to automate your tasks. QGIS supports the popular Python scripting language. Even if you are a beginner, learning a little bit of Python and QGIS programming interface will allow you to be much more productive in your work. This tutorial assumes no prior programming knowledge and is intended to give an introduction to python scripting in QGIS (PyQGIS).

Overview of the task

We will load a vector point layer representing all major airports and use python scripting to create a text file with the airport name, airport code, latitude and longitude for each of the airport in the layer.

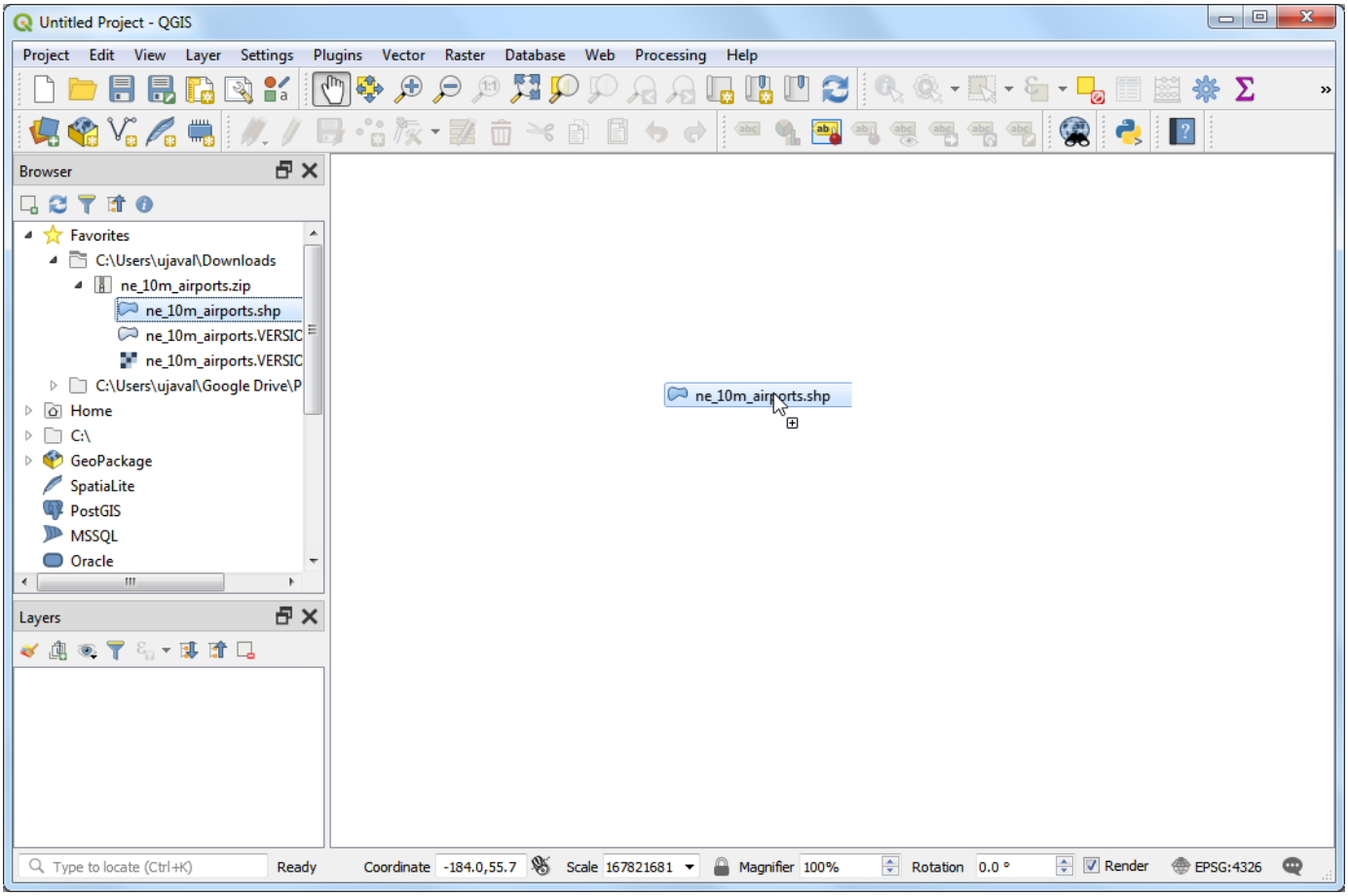
Get the data

We will use the Airports (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/airports/) dataset from Natural Earth.

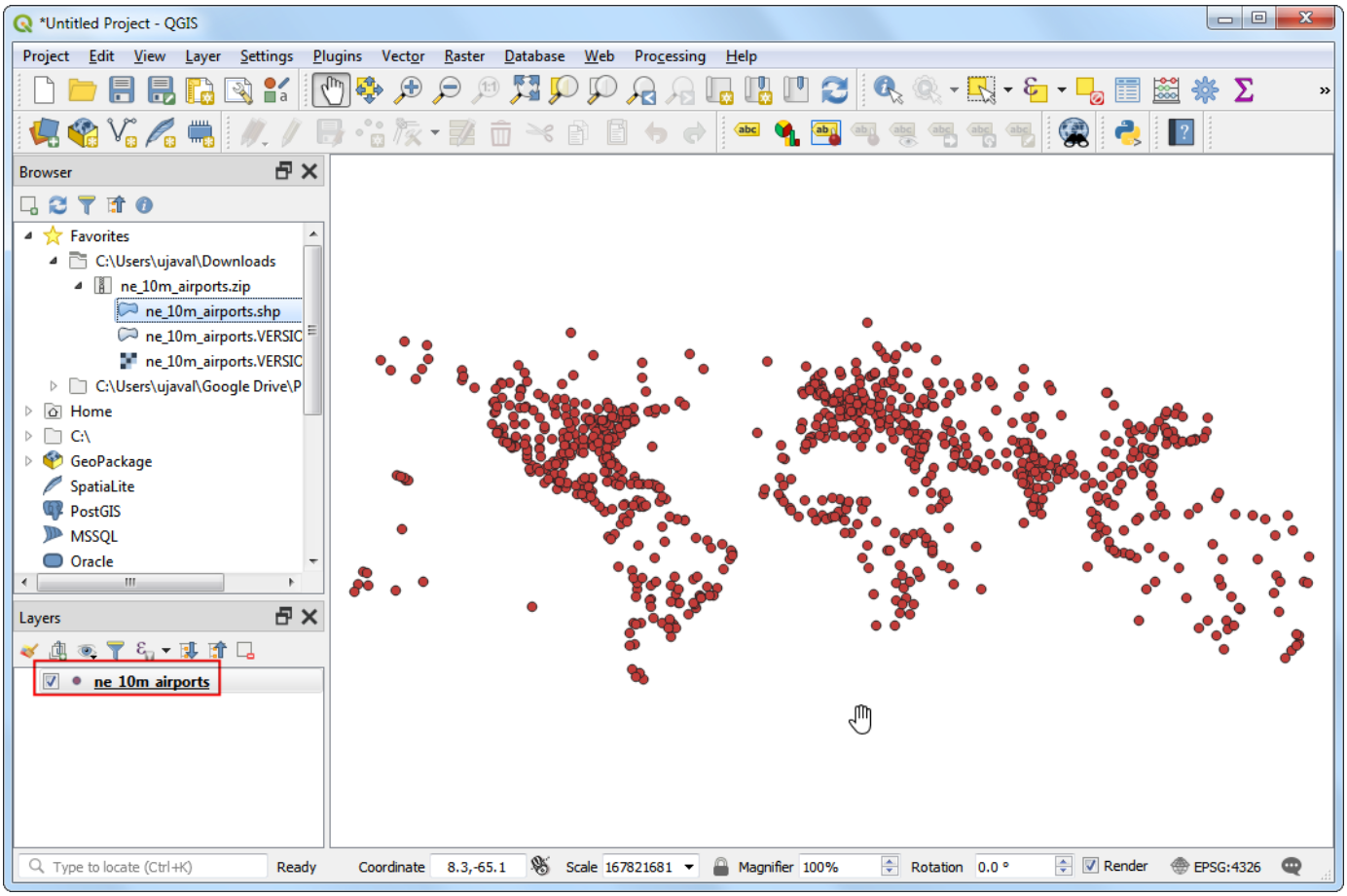
Download the Airports shapefile (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_airports.zip).

Procedure

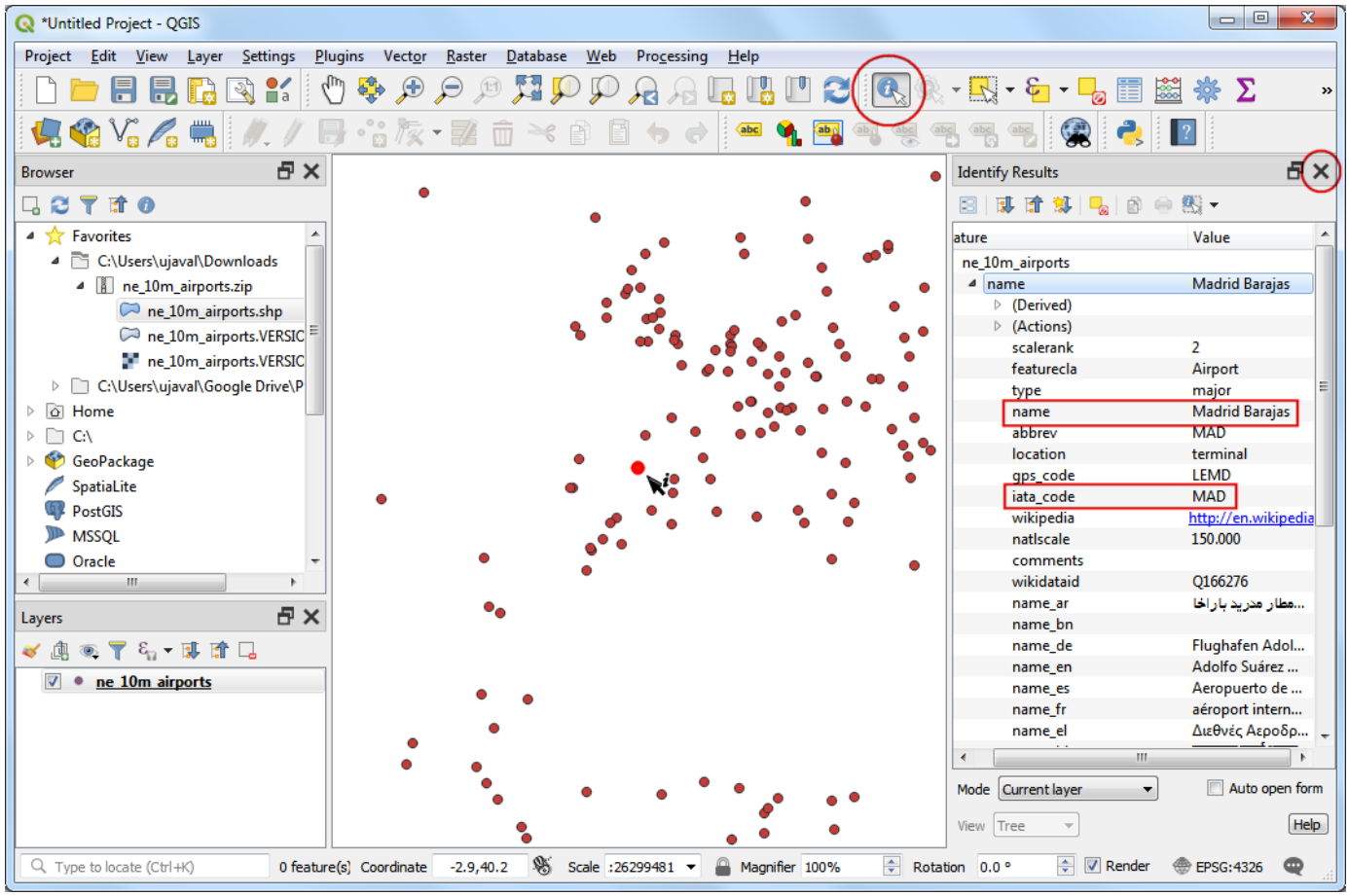
1. Locate the `ne_10m_airports.zip` file in the QGIS Browser and expand it. Select the `ne_10m_airports.shp` file and drag it to the canvas.



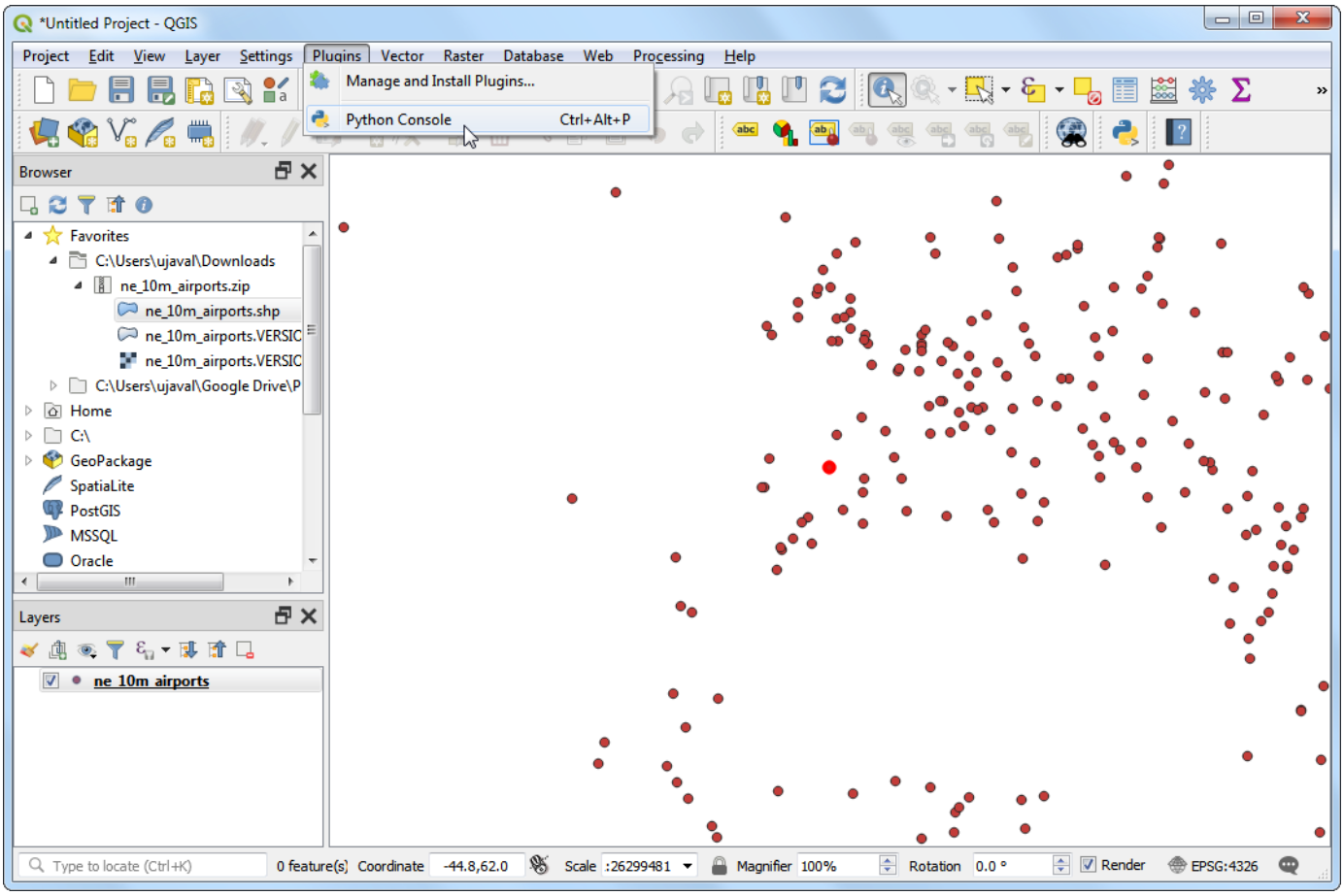
2. You will see the `ne_10m_airports` layer loaded in QGIS.



3. Select the Identify tool and click on any of the points to examine the available attributes. You will see that the name of the airport and it's 3 digit code are contained in the attributes `name` and `iata_code` respectively. You can close the Identify window.

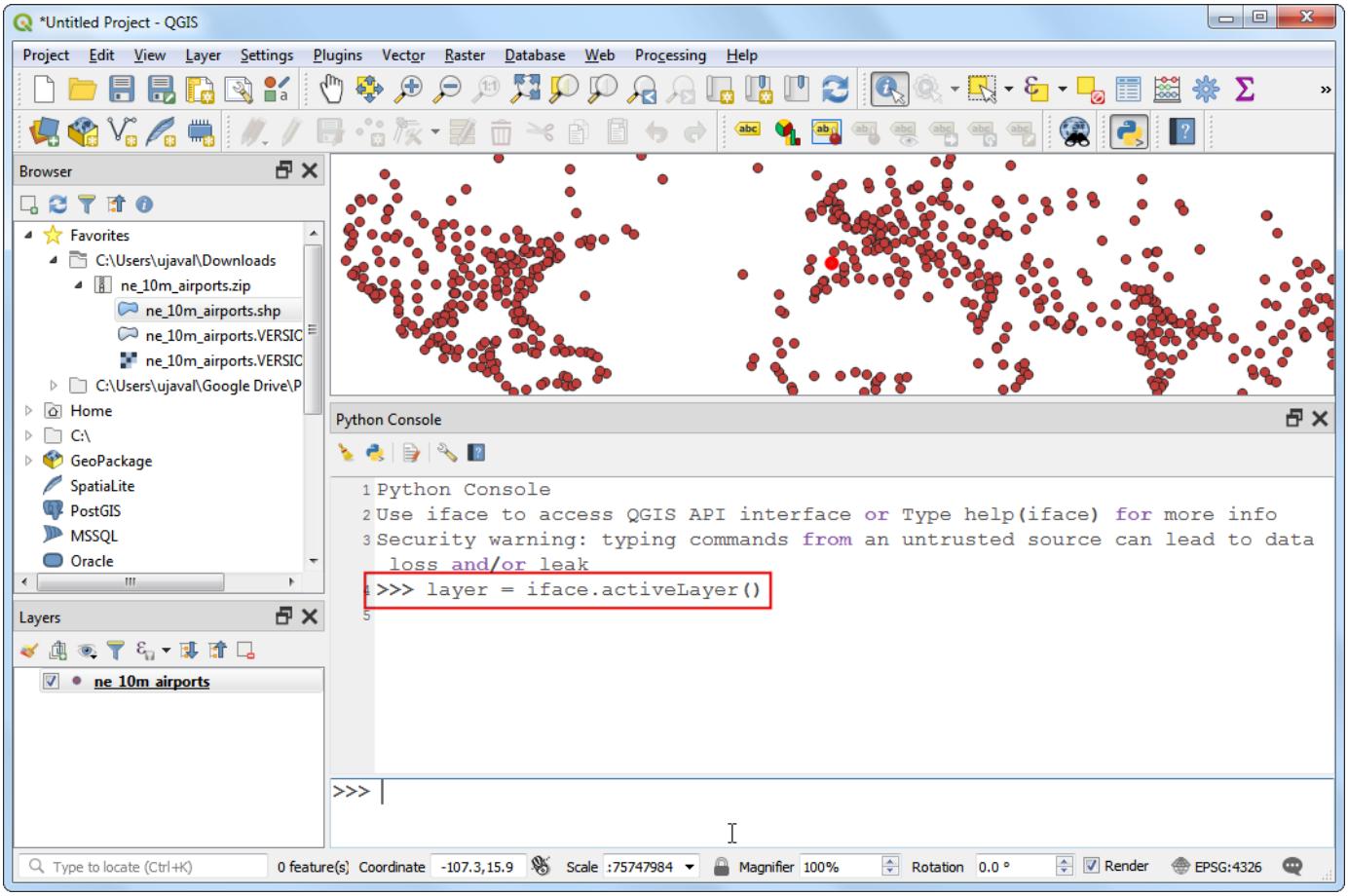


4. QGIS provides a built-in console where you can type python commands and get the result. This console is a great way to learn scripting and also to do quick data processing. Open the Python Console by going to Plugins ‣ Python Console.



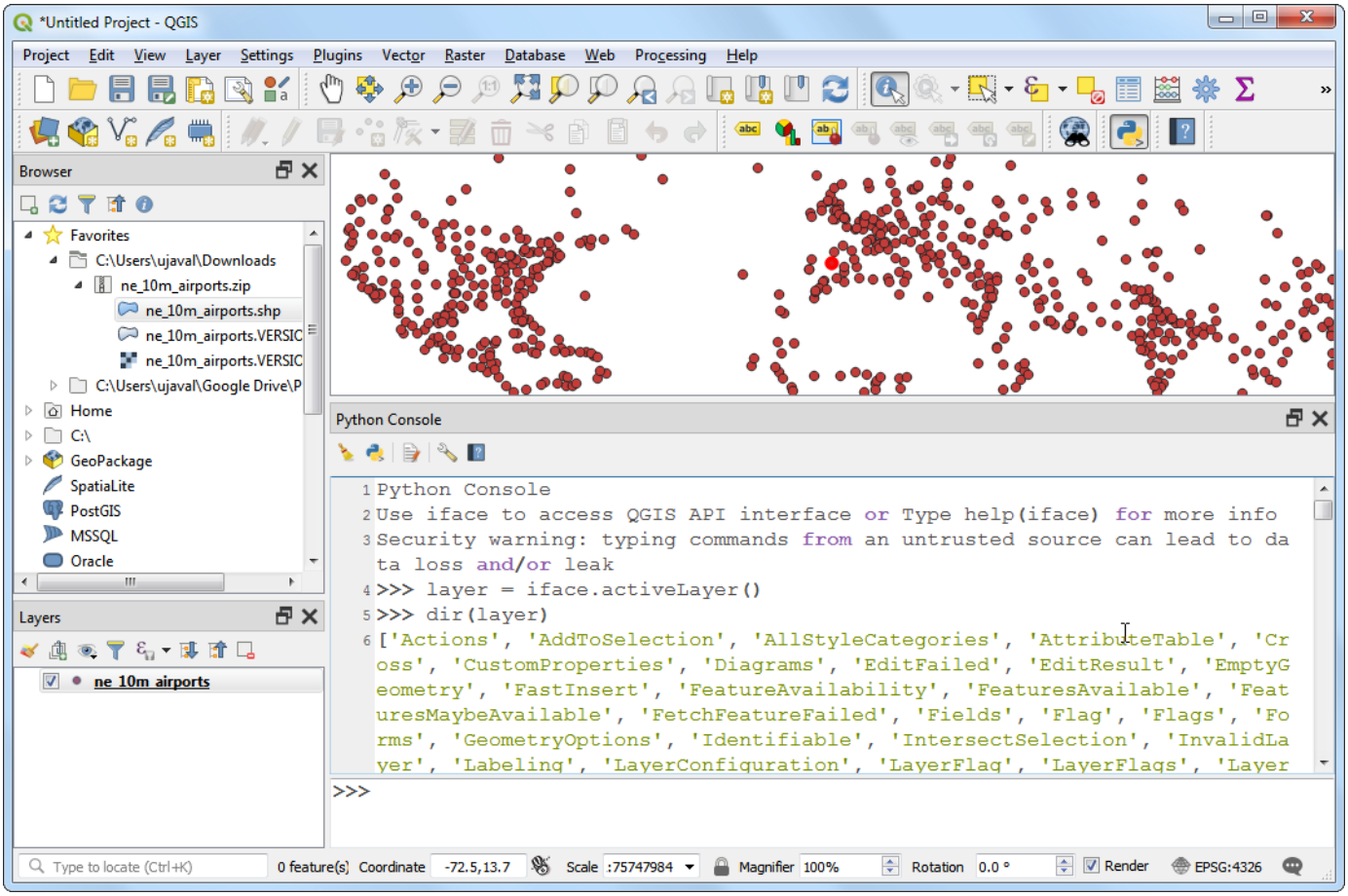
5. You will see a new panel open at the bottom of QGIS canvas. You will see a prompt like `>>>` at the bottom where you can type commands. For interacting with the QGIS environment, we must use the `iface` variable. To access the currently active layer in QGIS, you can type the following and press `Enter`. This command fetches the reference to the currently loaded layer and stores it in the `layer` variable.

```
layer = iface.activeLayer()
```



6. There is a handy function called `dir()` in python that shows you all available methods for any object. This is useful when you are not sure what functions are available for the object. Run the following command to see what operations we can do on the `layer` variable.

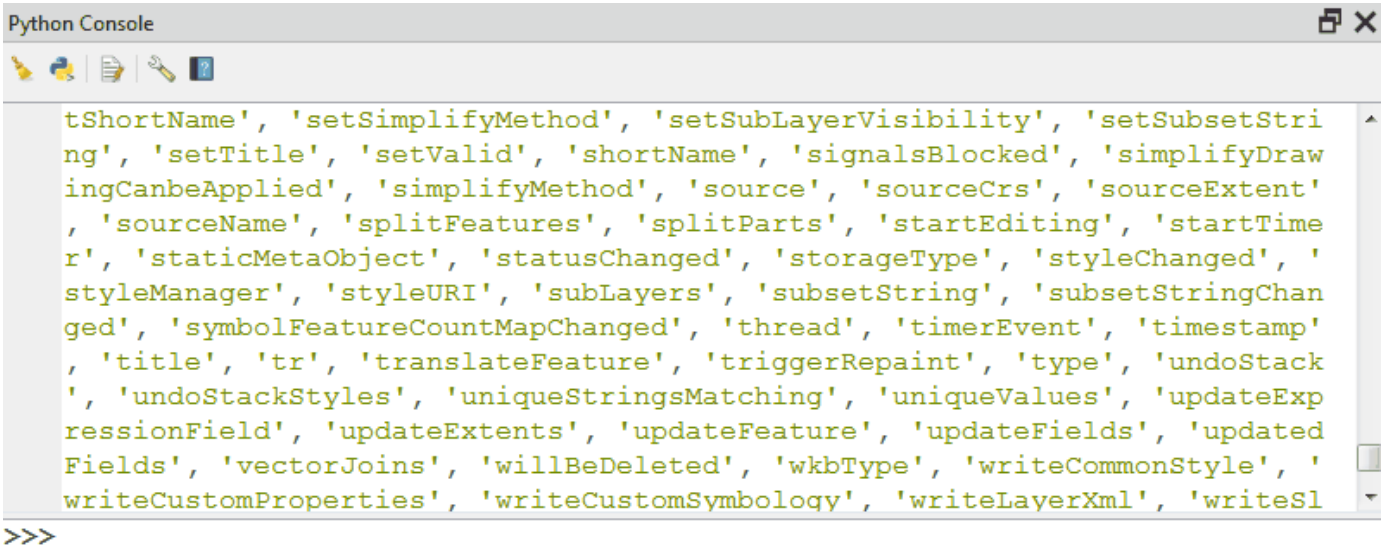
```
dir(layer)
```



7. You will see a long list of available functions. For now, we will use a function called `getFeatures()` which will gets you the reference to all features of a layer. In our case, each feature will be a point representing an airport. You can type the following command to iterate through each of the features in the current layer.

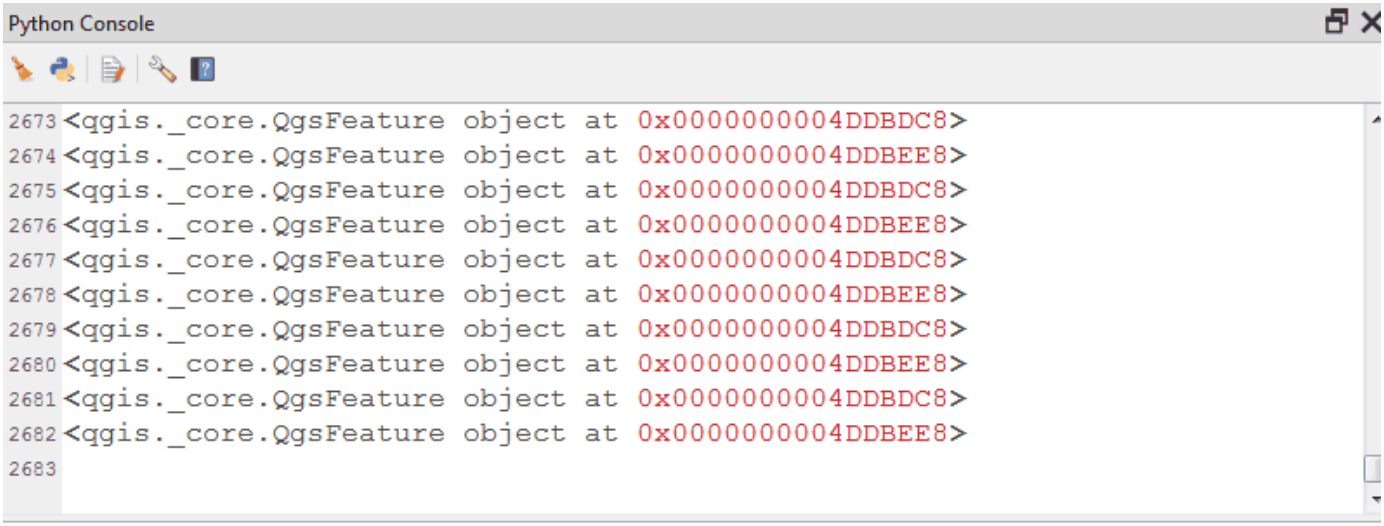
> **Note**
>
> Indentation (or number of spaces before each statement) is very important in Python. If you get error in this step, make sure you have added 2 spaces before typing the second line.
>
> As the print(f) statement is inside a for-loop, you will need to press `Enter` twice after that statement - once to exit the loop - and another to execute the command.

```
for f in layer.getFeatures():
  print(f)
```



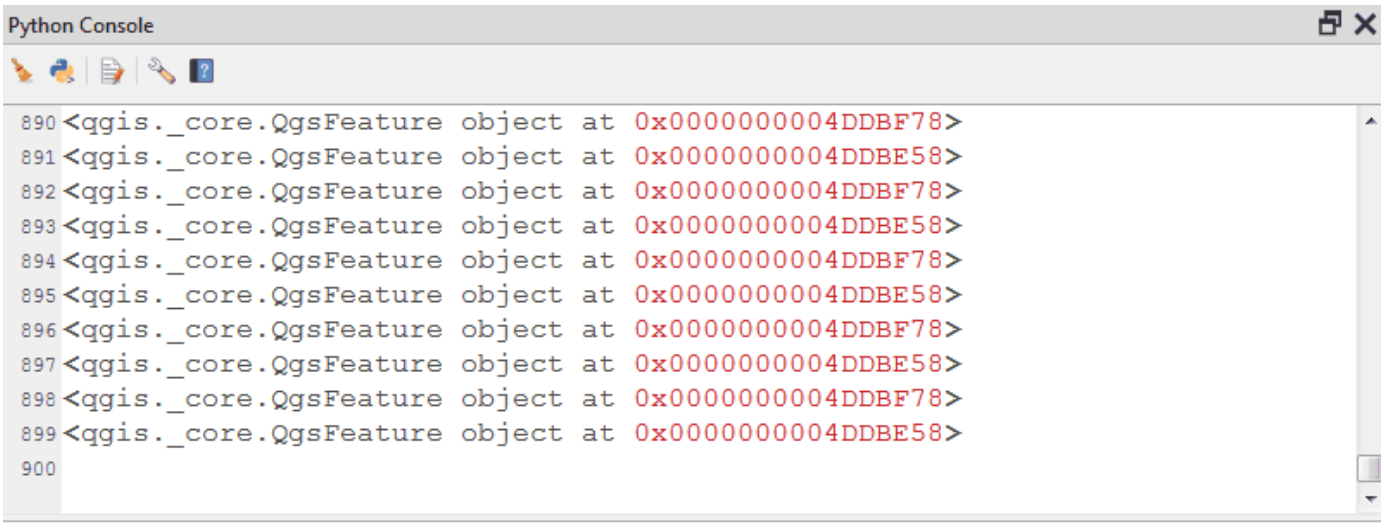
8. As you will see in the output, each line contains a reference to a feature within the layer. The reference to the feature is stored in the `f` variable. We can use the `f` variable to access the attributes of each feature. Type the following to print the `name` and `iata_code` for each airport feature.

```
for f in layer.getFeatures():
  print(f['name'], f['iata_code'])
```



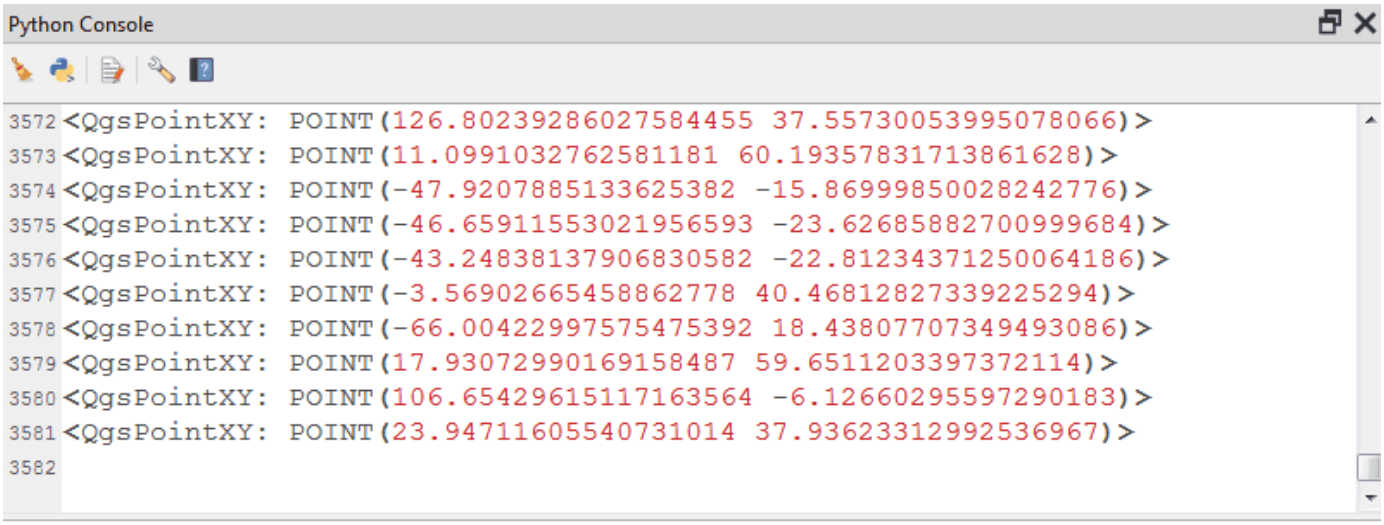
9. So now you know how to programatically access the attribute of each feature in a layer. Let's see how we can access the coordinates of the feature. The coordinates of a vector feature can be accessed by calling the `geometry()` function. This function returns a geometry object that we can store in the variable `geom`. You can run `asPoint()` function on the geometry object to get the x and y coordinates of the point. If your feature is a line or a polygon, you can use `asPolyline()` or `asPolygon()` functions. Type the following code at the prompt and press `Enter` to see the x and y coordinates of each feature.

```
for f in layer.getFeatures():
  geom = f.geometry()
  print(geom.asPoint())
```



10. What if we wanted to get only the `x` cordinate of the feature? You can call the `x()` function on the point object and get its x coordinate.

```
for f in layer.getFeatures():
  geom = f.geometry()
  print(geom.asPoint().x())
```



11. Now we have all the pieces that we can stitch together to generate our desired output. Type the following code to print the name, iata_code, latitude and longitude of each of the airport features. Here we are using the `.format()` method which gives more control on printing multiple variables. The `.2f` notation is to limit the coordinates to 2 decimals.

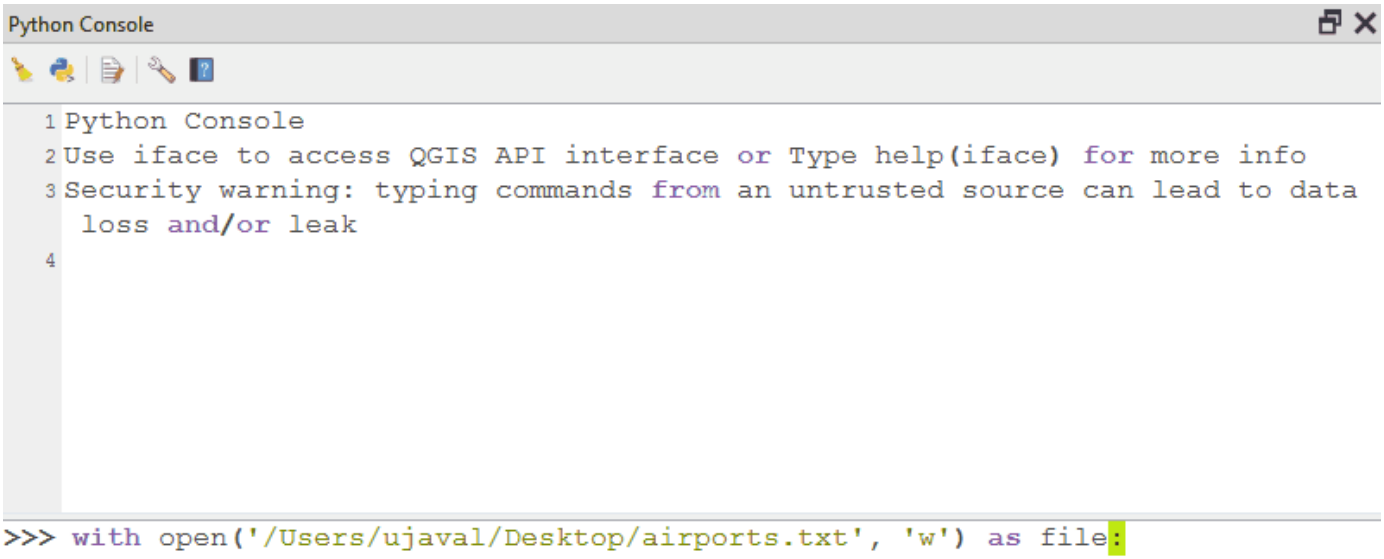
```
for f in layer.getFeatures():
  geom = f.geometry()
  print('{},{},{:.2f},{:.2f}'.format(f['name'], f['iata_code'], geom.asPoint().y(), geom.asPoint().
```

12. You can see the output printed on the console. A more useful way to store the output would be in a file. You can type the following code to create a file and write the output there. Replace the file path with a path on your own system. Note that we add `\n` at the end of our line formatting. This is to add a newline after we add the data for each feature.

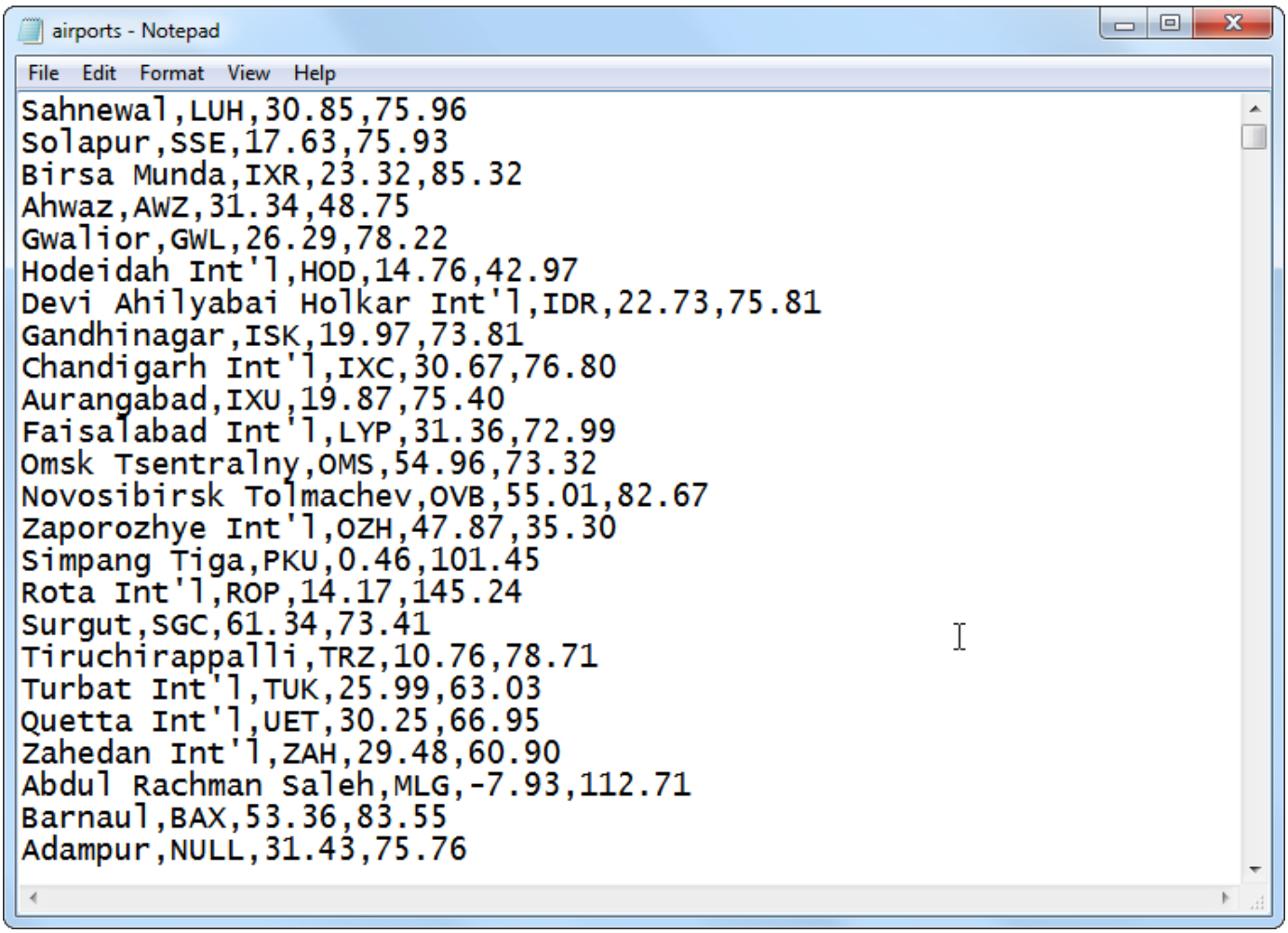
Note

There are 2 levels of code blocks below. Do make sure to add 4 spaces to the code starting line 3.

```
with open('/Users/ujaval/Desktop/airports.txt', 'w') as file:
  for f in layer.getFeatures():
    geom = f.geometry()
    line = '{},{},{:.2f},{:.2f}\n'.format(f['name'], f['iata_code'], geom.asPoint().y(), geom.asPoi
    file.write(line)
```



13. You can go to the output file location you specified and open the text file. You will see the data from the airports shapefile that we extracted using python scripting.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Running Processing Algorithms via Python (QGIS3)

The Processing Toolbox in QGIS contain an ever-growing collection of geoprocessing tools. The toolbox provides an easy batch processing interface to run any algorithm on a large number of inputs. See *Batch Processing using Processing Framework (QGIS3)* (batch_processing.html). But there are cases where you need to incorporate a little bit of custom logic in your batch processing. As all the processing algorithms can be run programmatically via the Python API, you can run them via the Python Console. This tutorial shows how to run a processing algorithm via the Python Console to perform a custom geoprocessing task in just a few lines of code. Please review the *Getting Started With Python Programming (QGIS3)* (getting_started_with_pyqgis.html) tutorial to get familiar with the basics of the Python Scripting environment in QGIS.

Overview of the Task

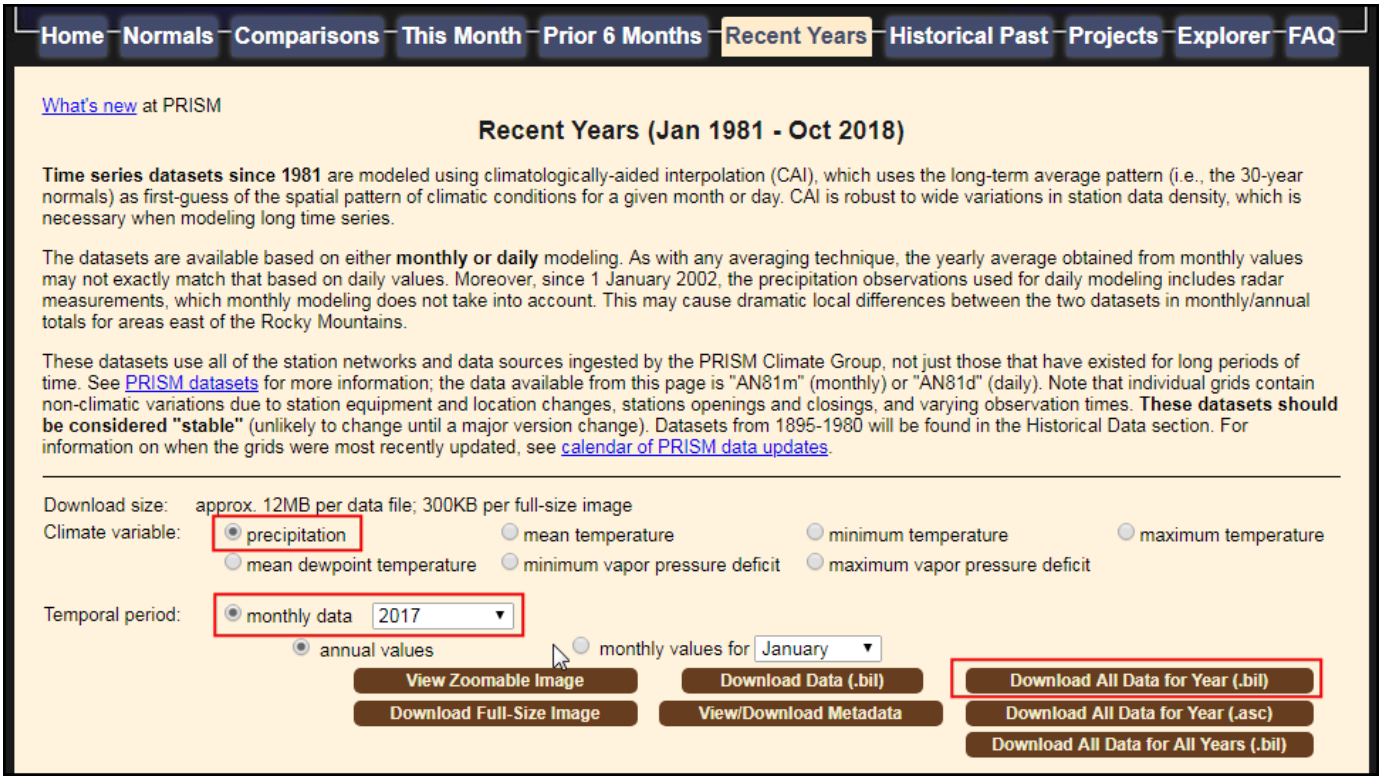
We will use 12 gridded raster layers representing precipitation for each month of year and calculate average monthly rainfall for all zip codes in the Seattle area.

Other skills you will learn

- How to delete a column (i.e. field) from a vector layer.

Get the data

The PRISM Climate Group (http://www.prism.oregonstate.edu/) gathers climate observation and provides historic and current climate data for the conterminous US. Head over to the Recent Years (http://www.prism.oregonstate.edu/recent/) data page and download the monthly precipitation data for the year 2017 in BIL format.



City of Seattle Open Data portal (https://data.seattle.gov/) provides free and open data for the city. Search for and download the Zip Codes (https://data.seattle.gov/Land-Base/Zip-Codes/n58k-cykw) data in the shapefile format.

For convenience, you may directly download a copy of both the datasets from the links below:

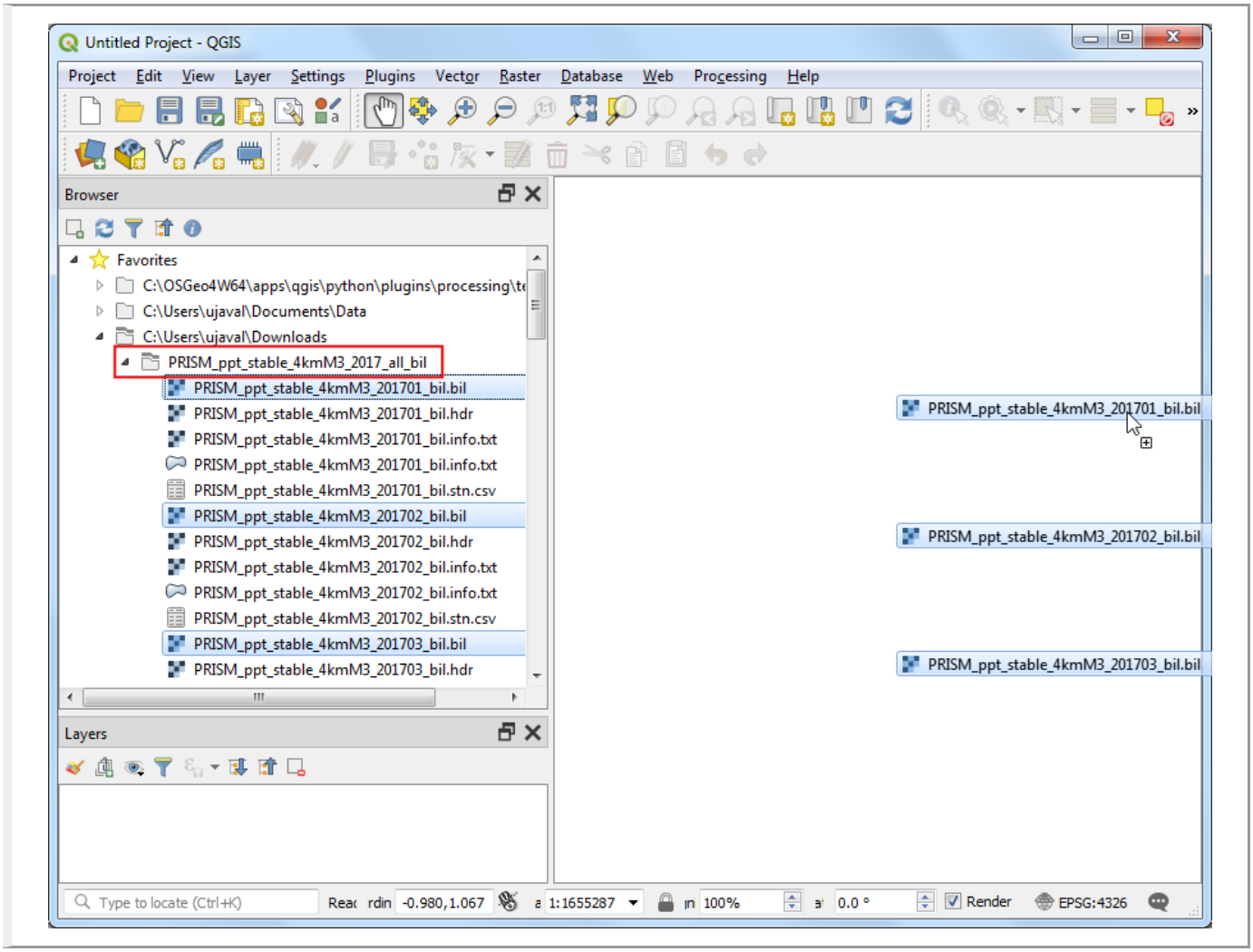
PRISM_ppt_stable_4kmM3_2017_all_bil.zip (http://www.qgistutorials.com/downloads/PRISM_ppt_stable_4kmM3_2017_all_bil.zip)

Zip_Codes.zip (http://www.qgistutorials.com/downloads/Zip_Codes.zip)

Data Source [PRISM] (../credits.html#prism) [CITYOFSEATTLE] (../credits.html#cityofseattle)

Procedure

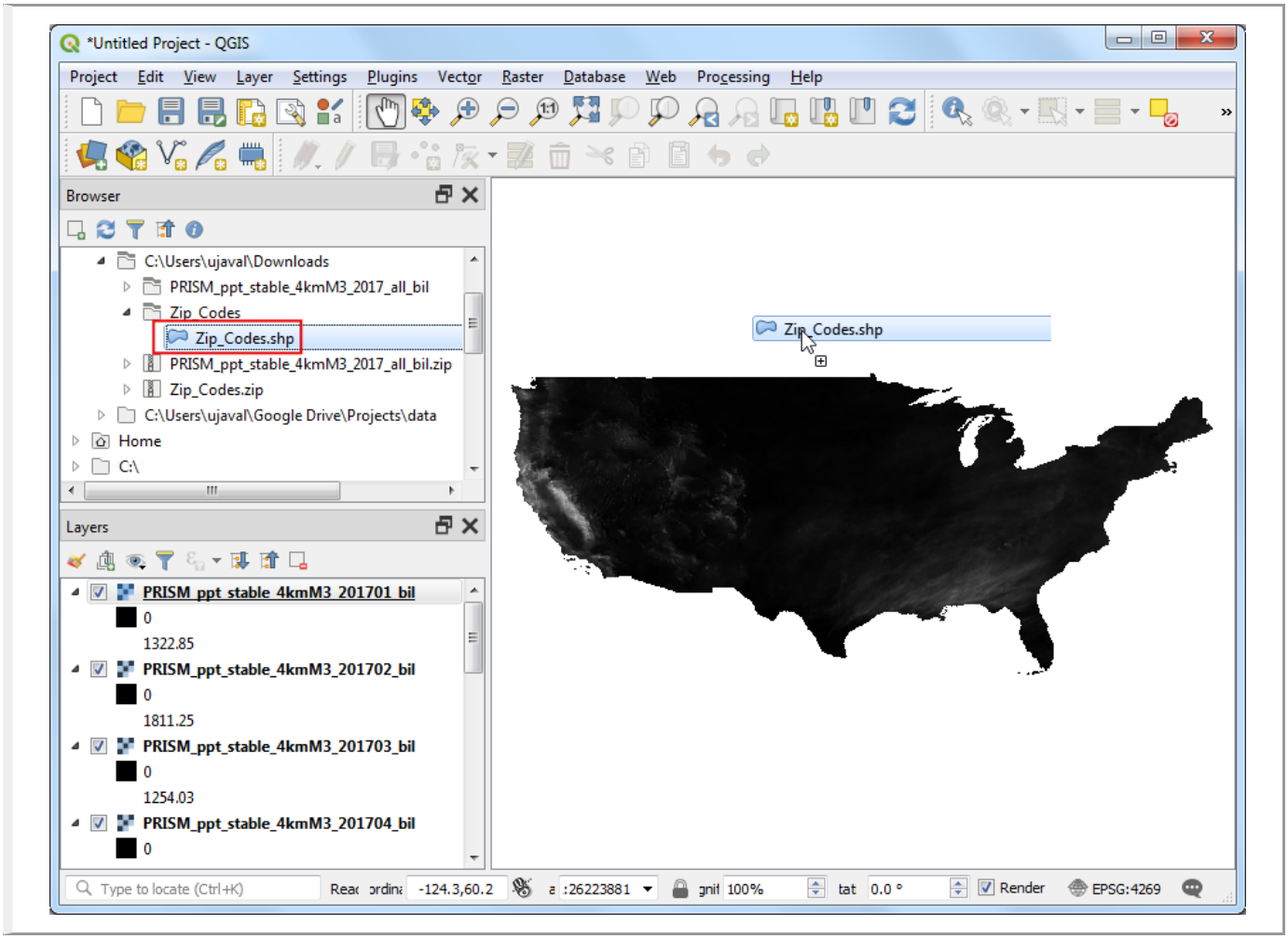
1. Unzip the `PRISM_ppt_stable_4kmM3_2017_all_bil.zip` file. Locate the `PRISM_ppt_stable_4kmM3_2017_all_bil` folder in the QGIS Browser and expand it. The folder contains 12 individual layers for each month. Hold the `Ctrl` key and select the `.bil` files for all 12 months. Once selected, drag them to the canvas.



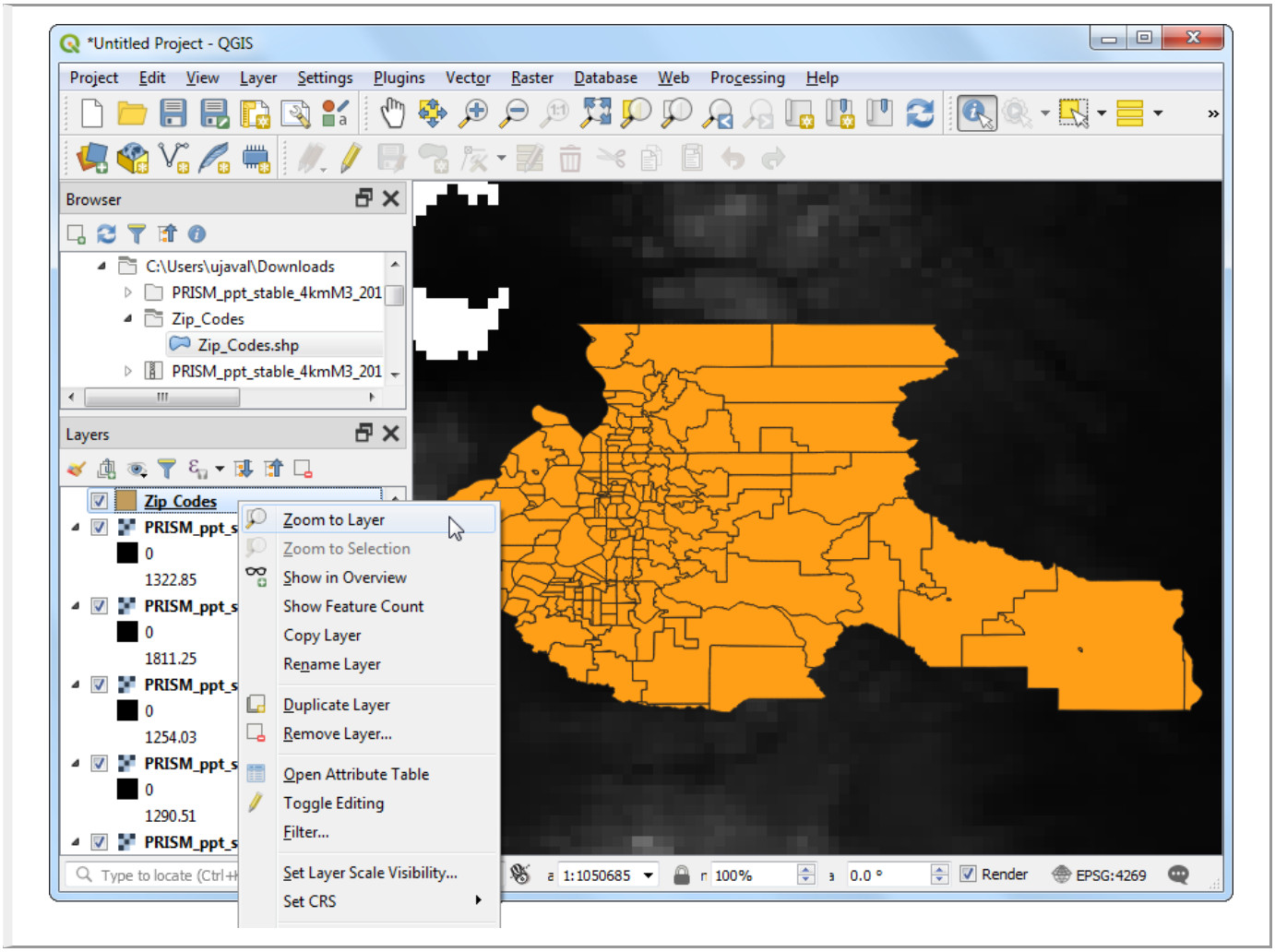
Note

The data is provided in the BIL format (http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/bil-bip-and-bsq-raster-files.htm). Each layer is presented with a set of files `.bil` file containing the actual data, a `.hdr` file describing the data structure and a `.prj` file containing the projection information. QGIS can load the `.bil` file and provided the other files exist in the same directory.

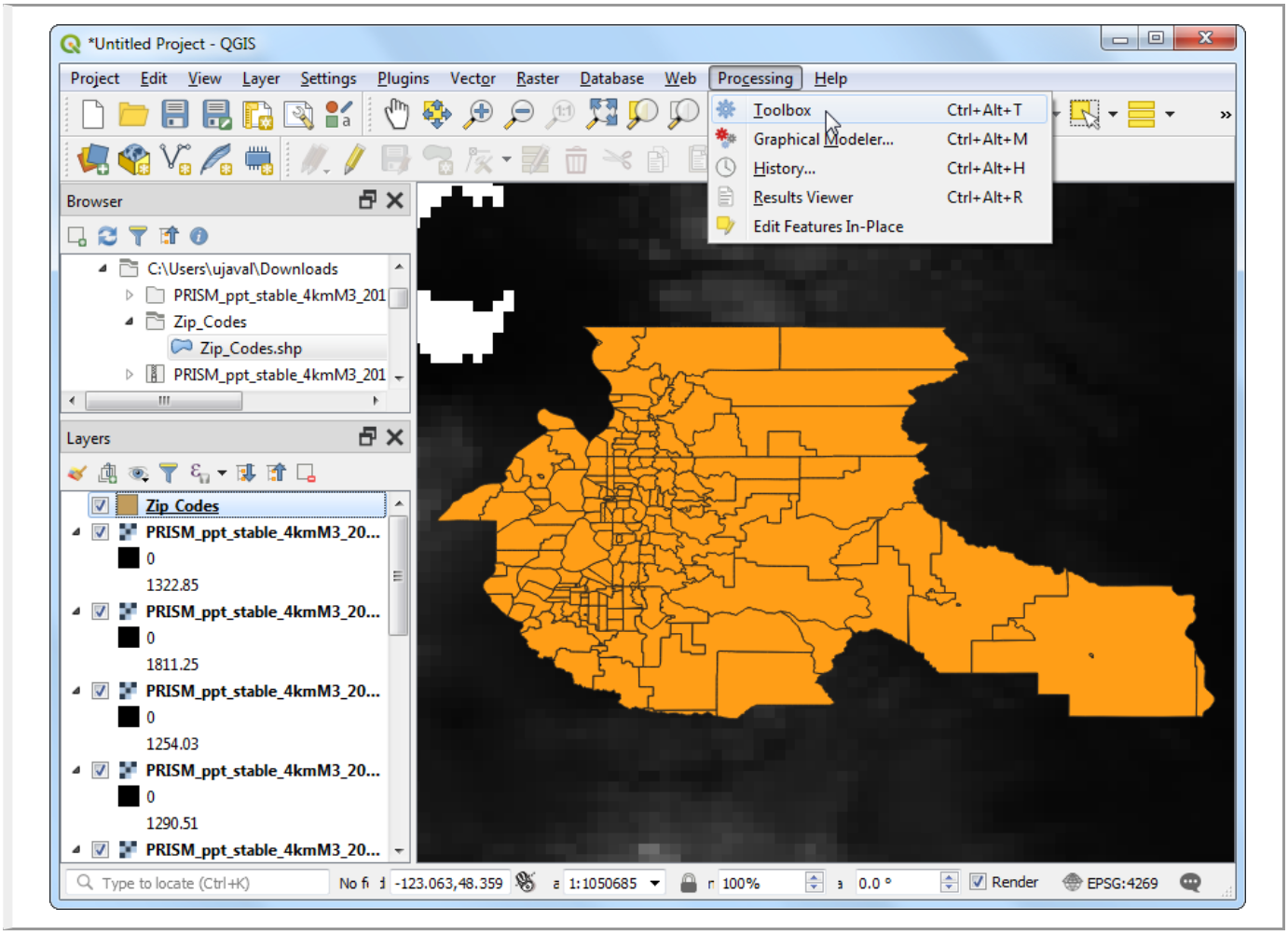
2. Next, unzip the `Zip_Codes.zip` file and extract the shapefile to a folder. Locate the `Zip_Codes` folder and expand it. Drag the `Zip_Codes.shp` file to the canvas.



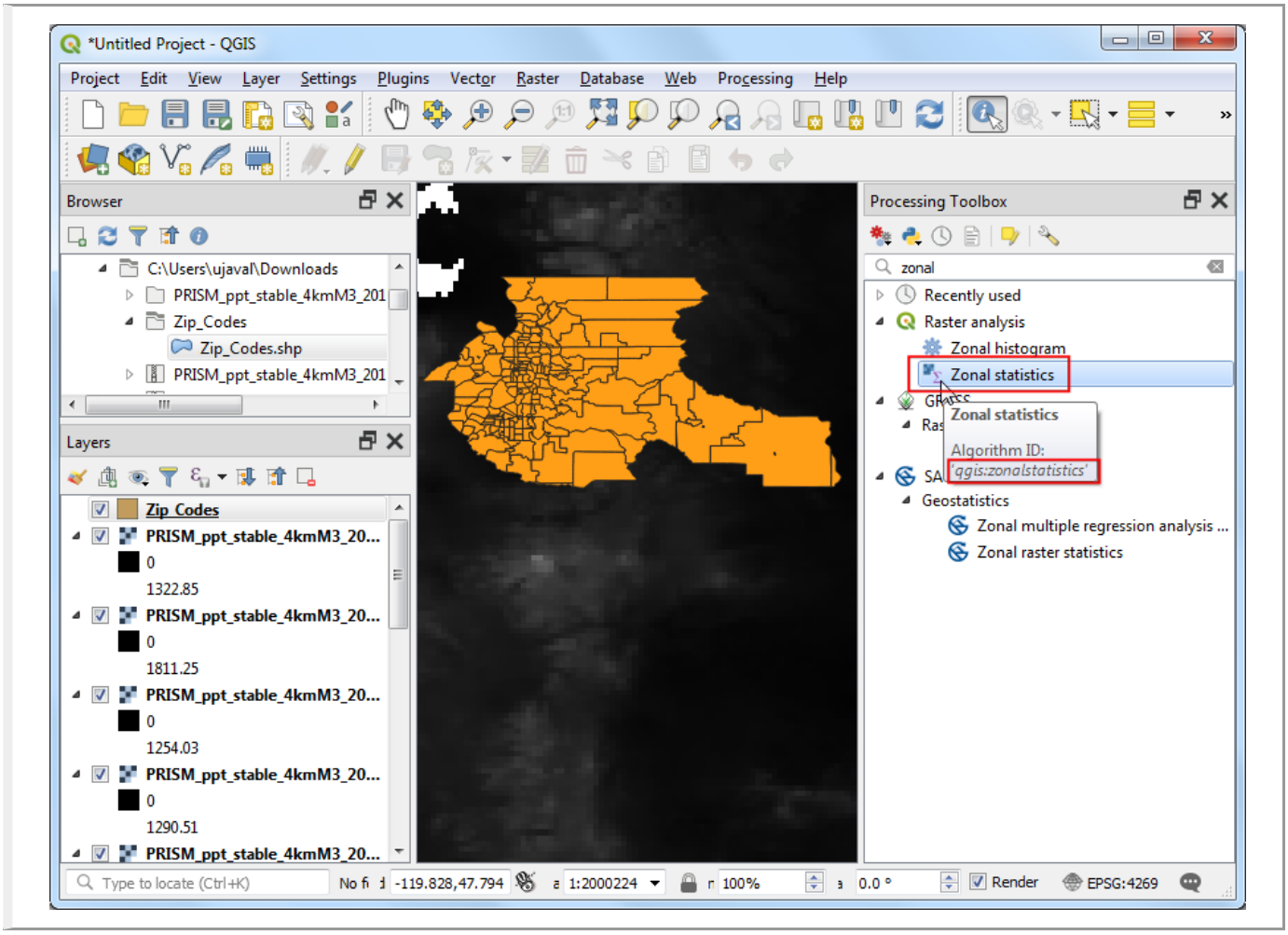
3. Right-click the `Zip_Codes` layer and select Zoom to Layer. You will see the zip code polygons for the city of seattle and neighboring areas.



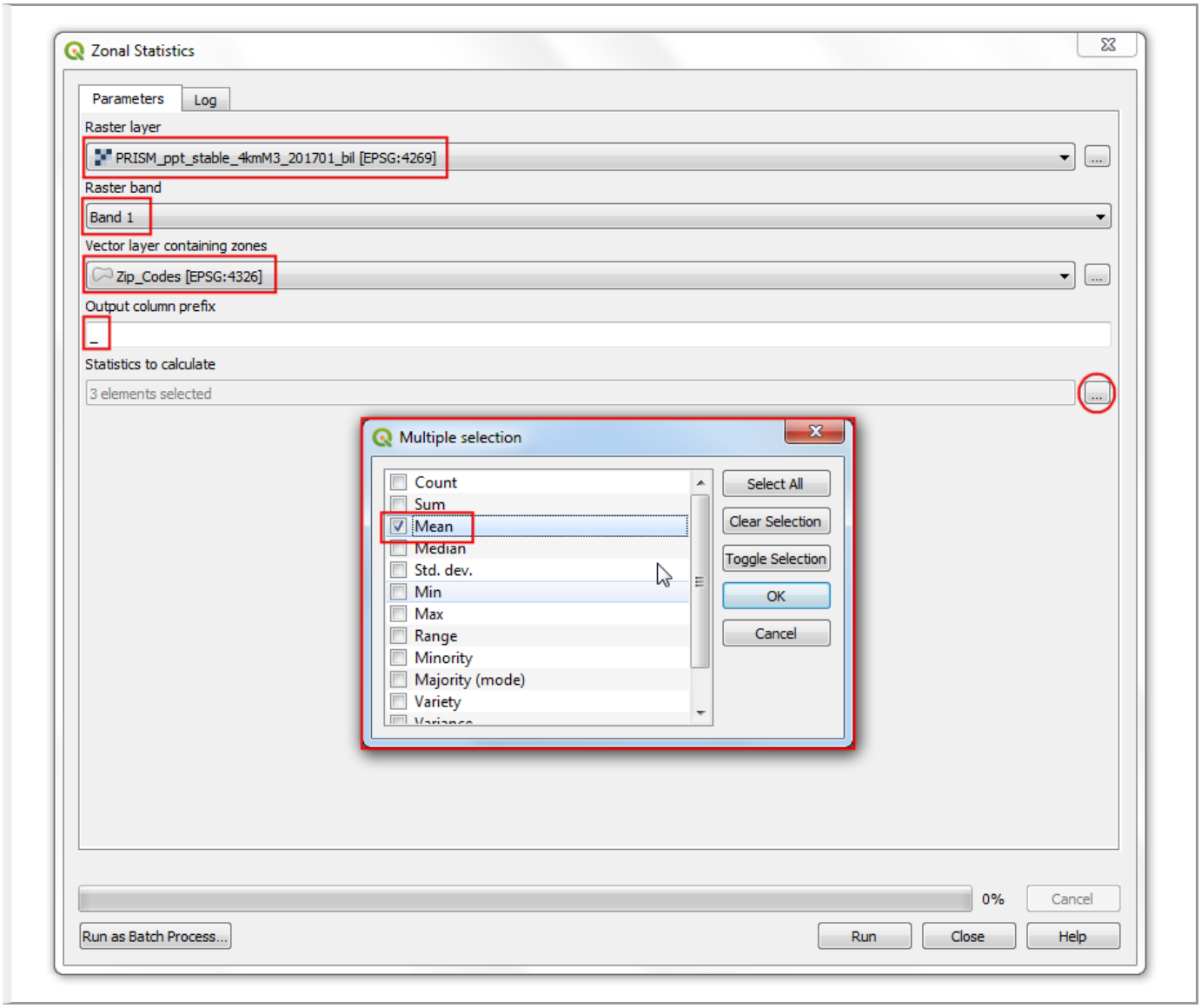
4. Go to Processing ▸ Toolbox.



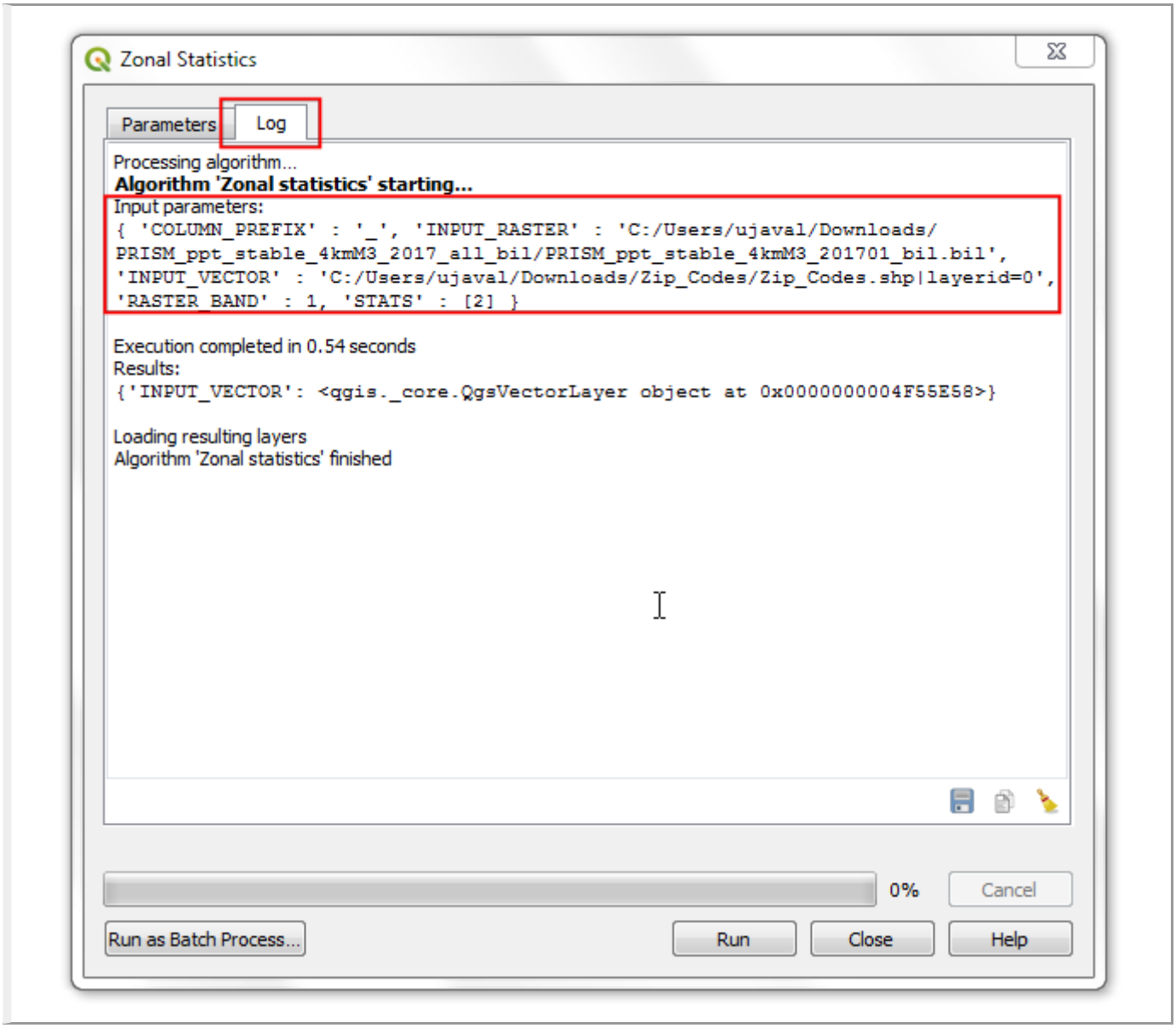
5. The algorithm to sample a raster layer using vector polygons is known as `Zonal statistics`. Search for the algorithm in the Processing Toolbox. Select the algorithm and hover your mouse over it. You will see a tooltip with the text *Algorithm ID: 'qgis:zonalstatistics'*. Note this id which will be needed to call this algorithm via the Python API. Double-click the `Zonal Statistics` algorithm to launch it.



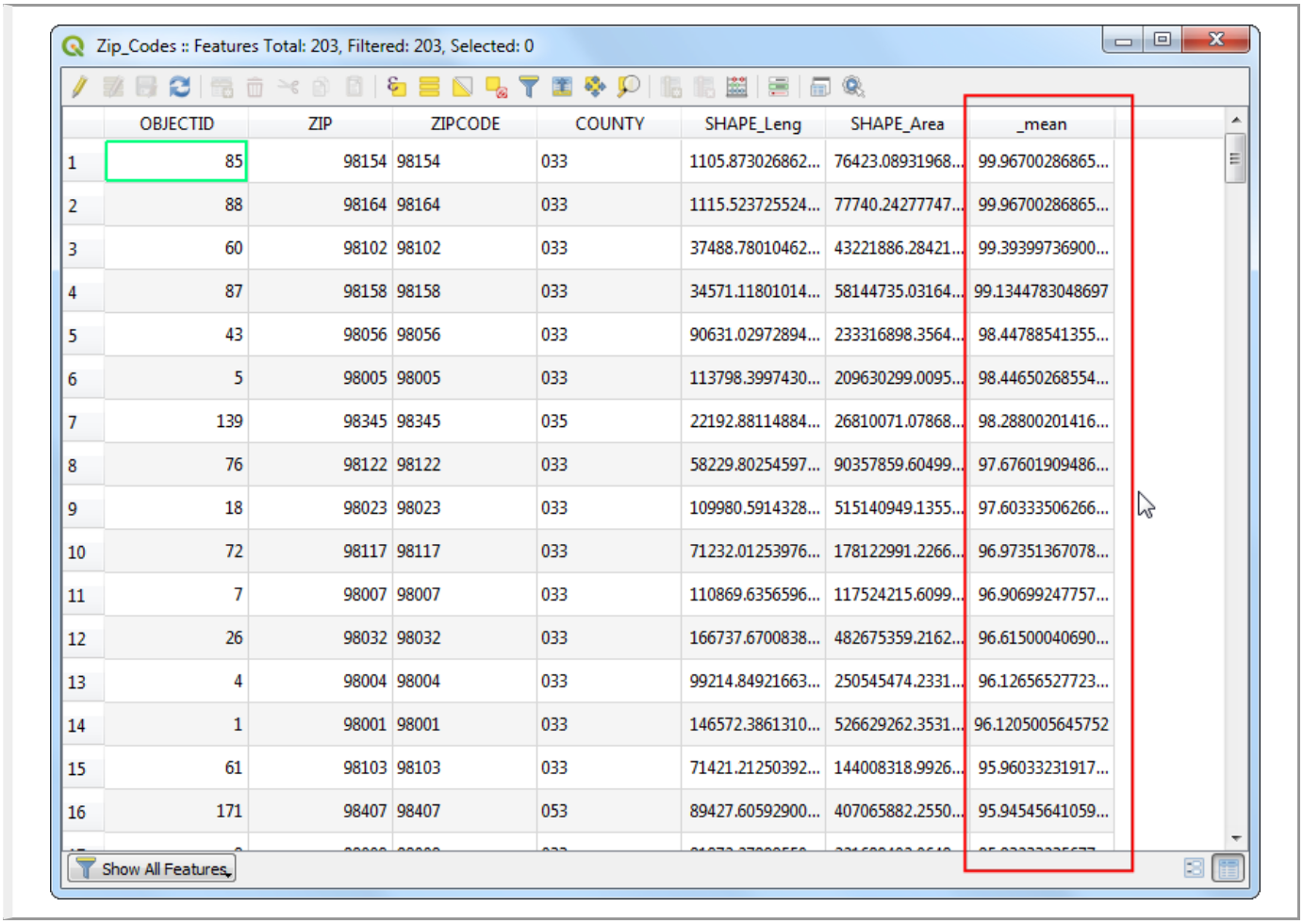
6. We will do a manual test run of the algorithm for a single layer. This is a useful way to check if the algorithm behaves as expected and also an easy way to find out how to pass on relevant parameters to the algorithm when using it via Python. In the Zonal Statistics dialog, select `PRISM_ppt_stable_4kmM3_201701_bil` as the Raster Layer and `Zip_Codes` as the Vector layer containing zones. Leave other paramters to default. Click the ... button next to Statistics to calculate and select only `Mean` . Click Run.



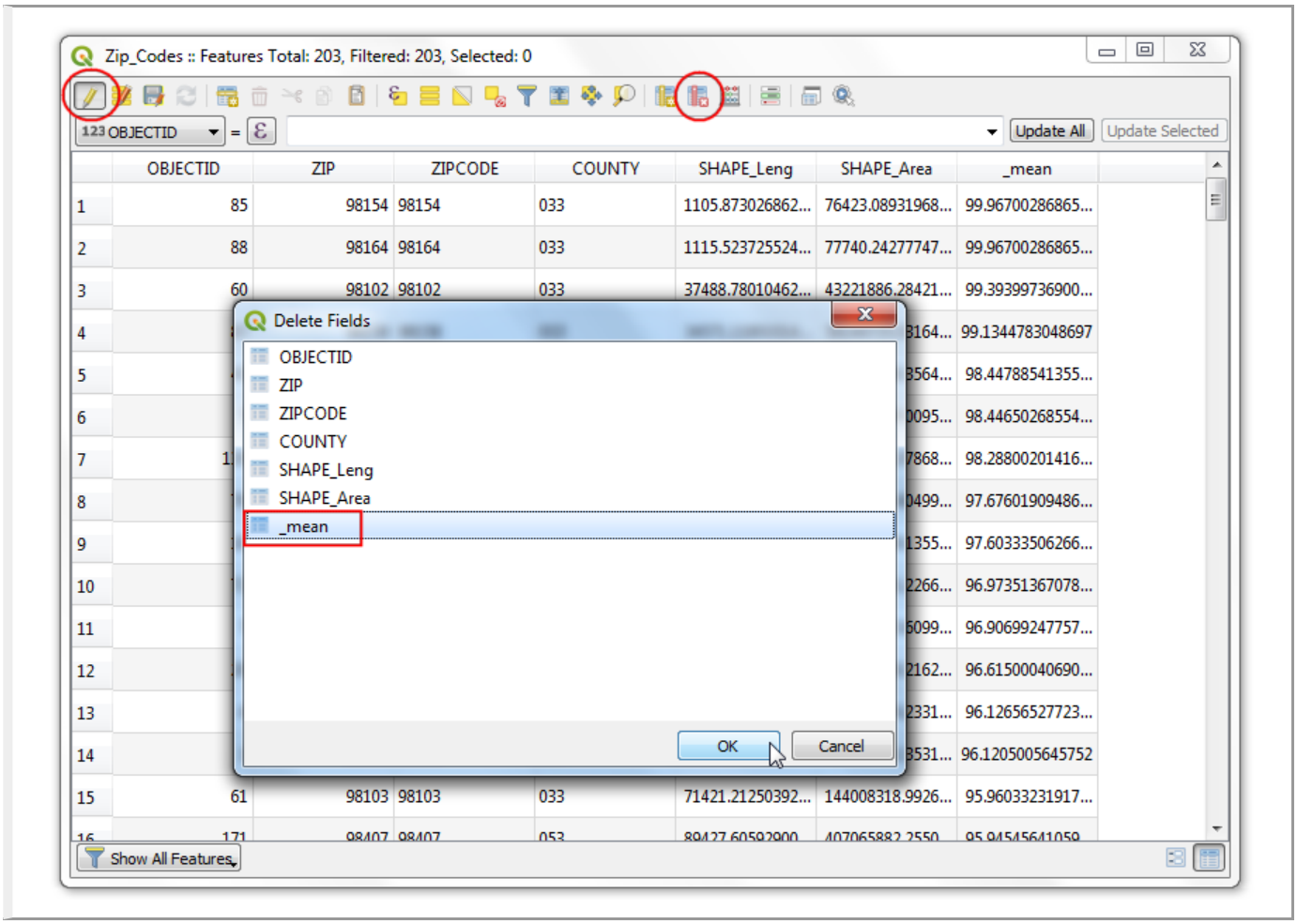
7. Once the algorithm finishes, switch to the Log tab. Make a note of the Input Parameters that were passed to the algorithm. Click Close.



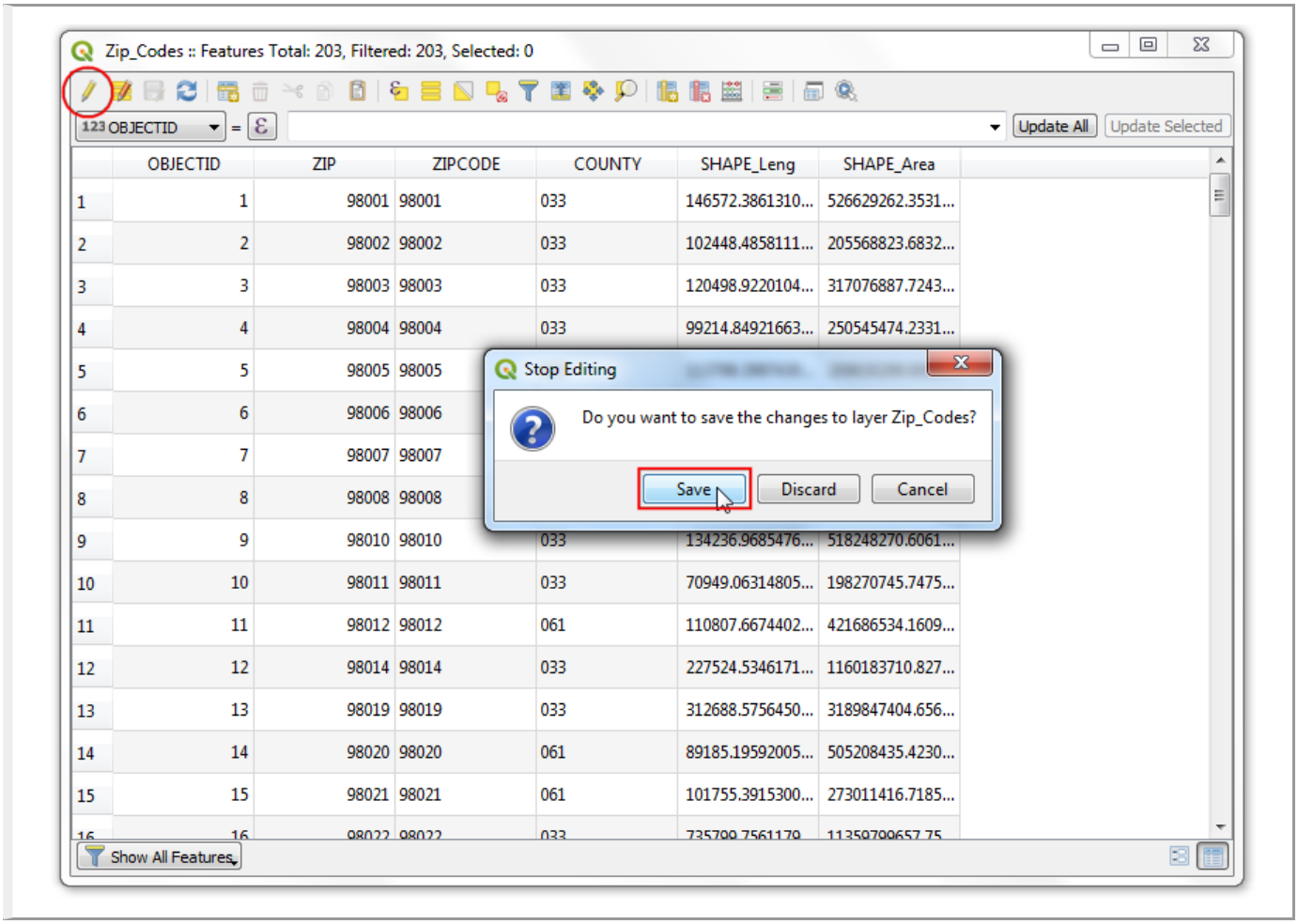
8. Let's check the results of the test run. In the main QGIS window, right-click the `Zip_Codes` layer and select Open Attribute Table. This particular algorithm modifies the input zone layer in-place and adds a new column for every statistic that was selected. As we had selected only `Mean` value, a new column named `_mean` is added to the table. The `_` was the default prefix. When we run the algorithm for layers of each month, it will be useful to specify a custom prefix with the month number so we can easily identify the mean values for each month (i.e. 01_mean, 02_mean etc.). Specifying this custom prefix is not possible in the Batch Processing interface of QGIS and if we ran this command using that interface, we would have to manually enter the custom prefix for each layer. If you are working with a large number of layers, this can be very cumbersome. Hence, we can add this custom logic using the Python API and run the algorithm in a for-loop for each layer.



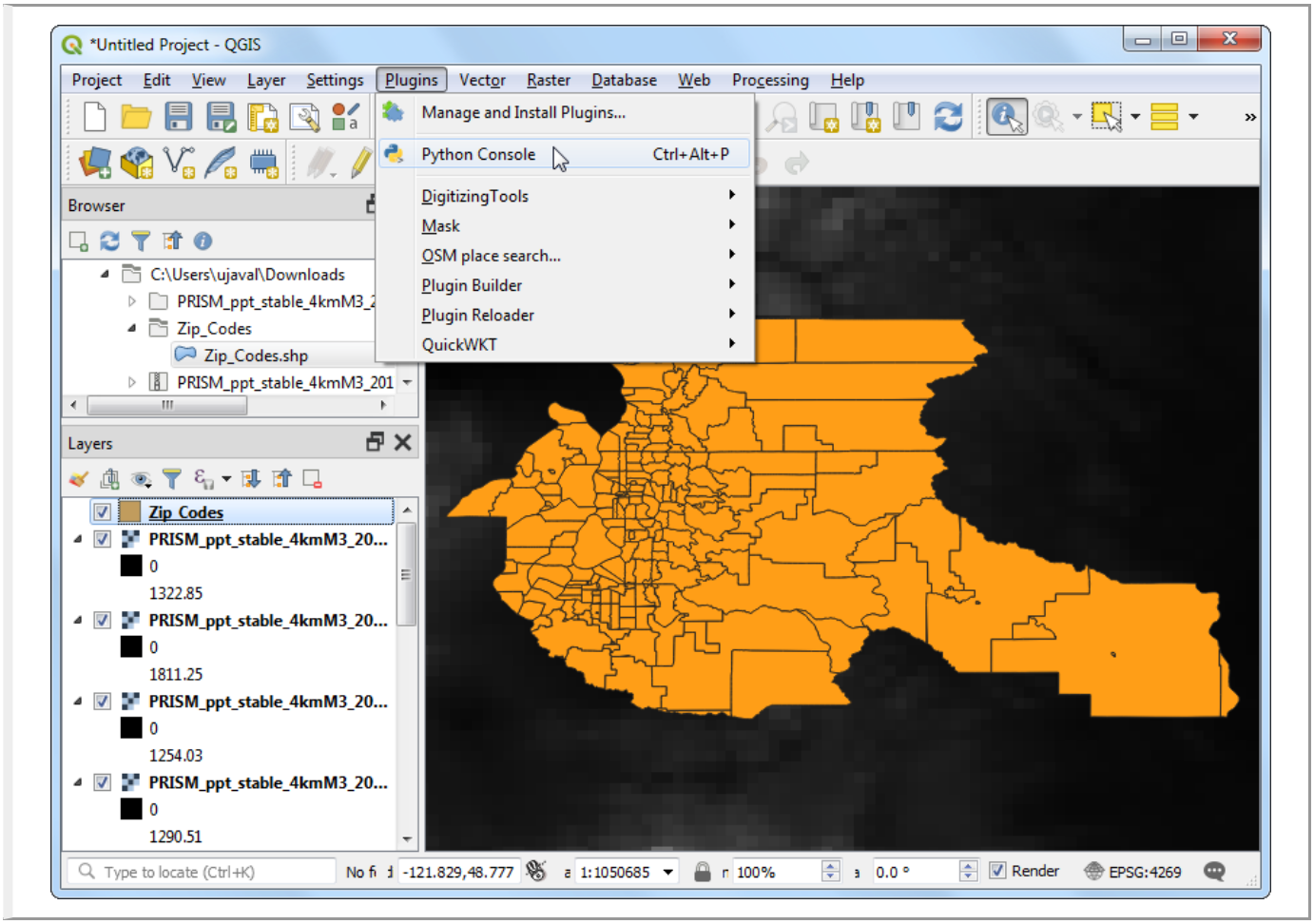
9. Before we proceed, let's delete the `_mean` column that was created during our test run. Click the Toggle Editing mode button, followed by Delete field button. Select the `_mean` field and click OK.



10. Click the Toggle Editing mode button again and Save the changes.

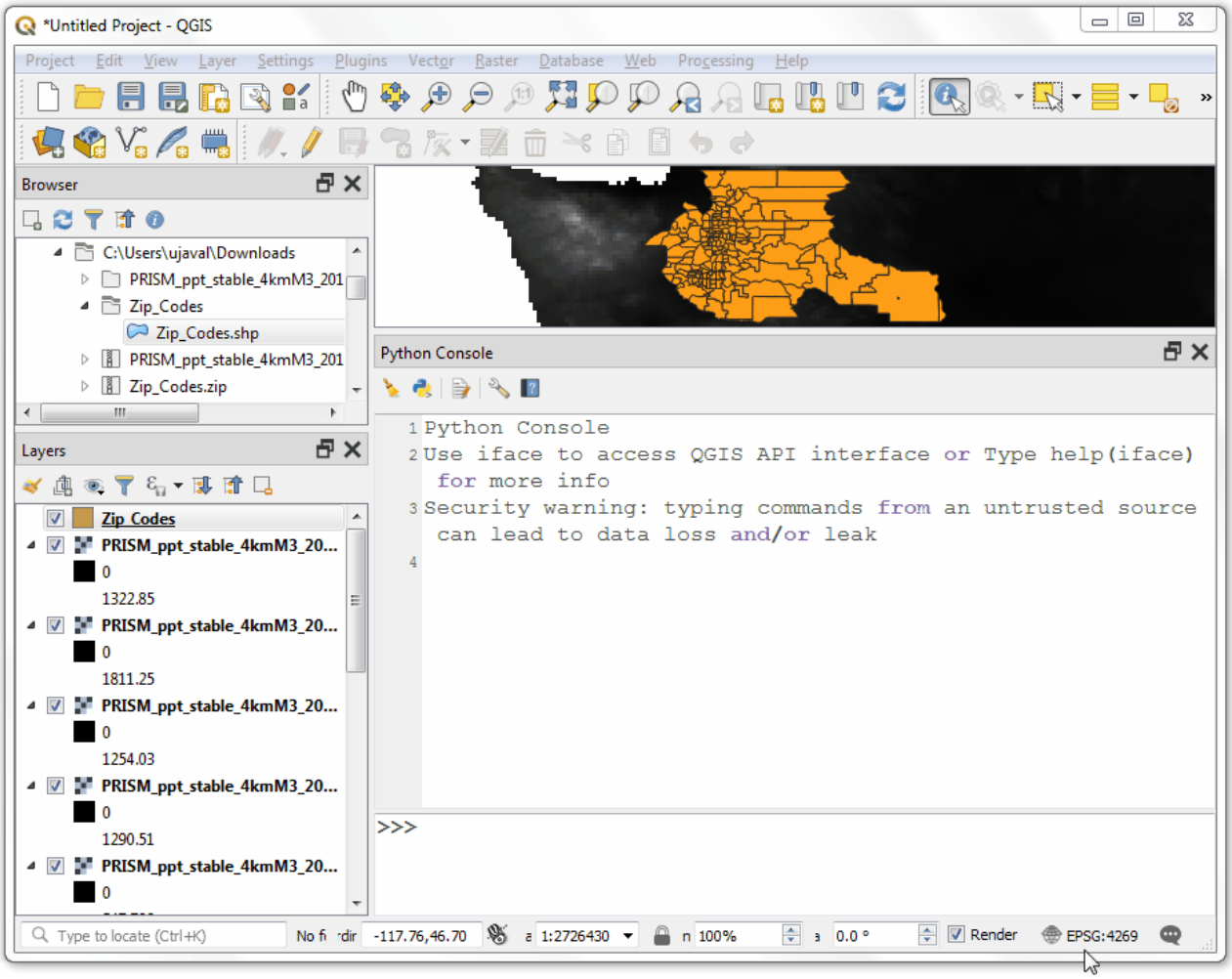


11. Back in the main QGIS window, go to Plugins ‣ Python Console.



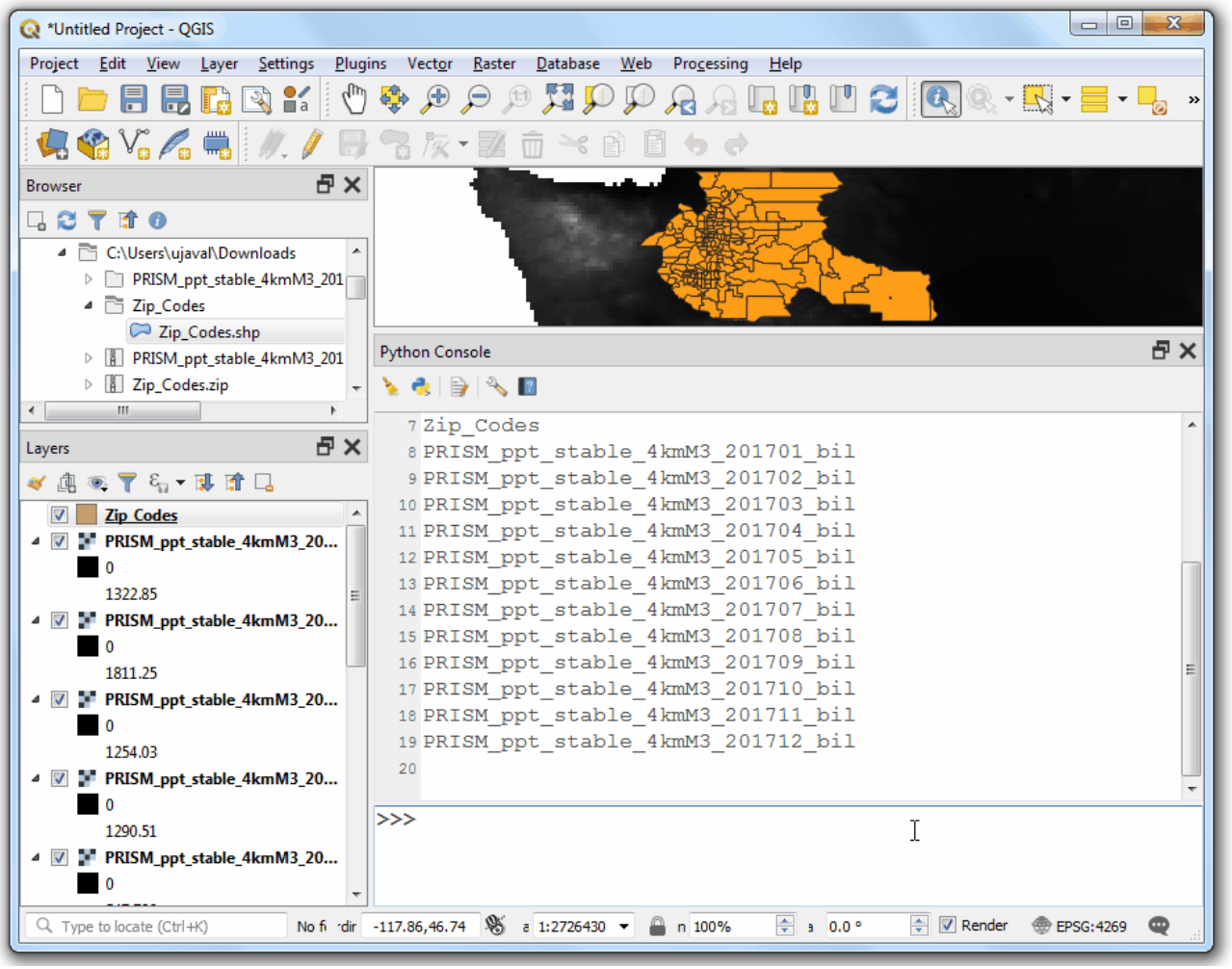
12. To run the processing algorithm via Python, we need to access names of all the layers. Enter the following code in the Python Console and hit `Enter`. You will see the names of all layers printed in the console.

```
root = QgsProject.instance().layerTreeRoot()
for layer in root.children():
  print(layer.name())
```

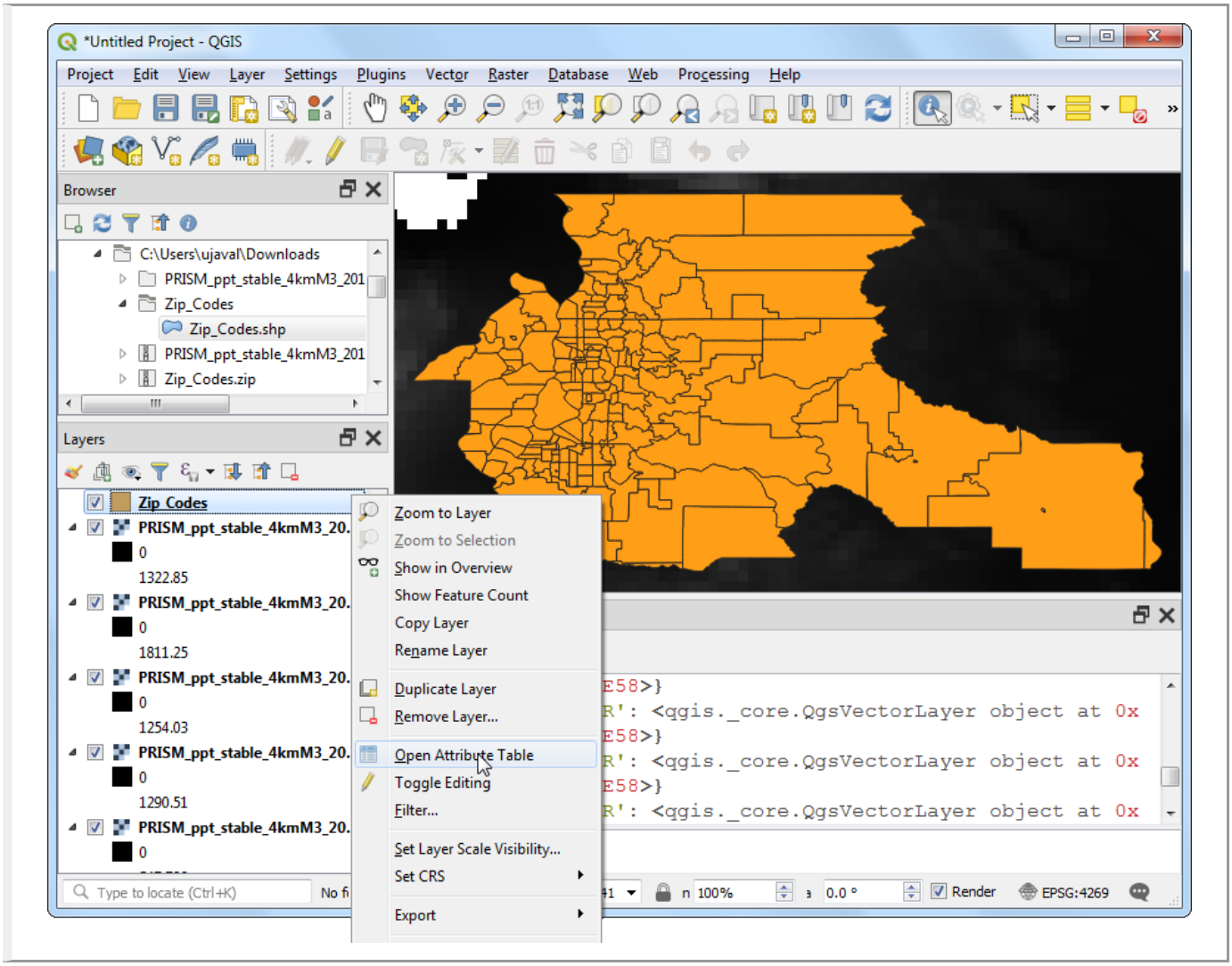


13. For adding a custom prefix, we need to look at the layer name and extract a substring representing the month number. Enter the following code to iterate over all raster layers, extract the custom prefix and run the `qgis:zonalstatistics` algorithm using it.

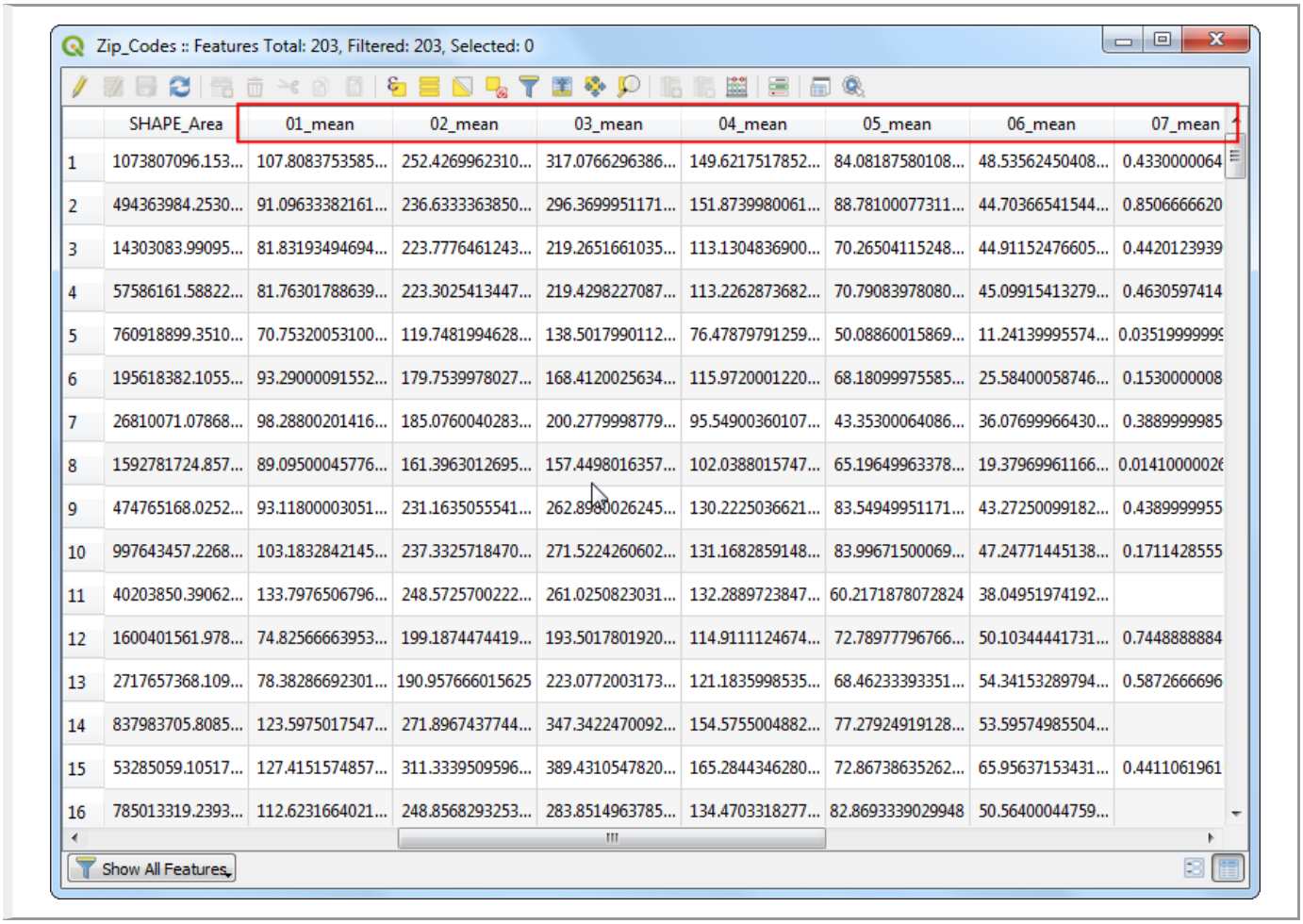
```
root = QgsProject.instance().layerTreeRoot()
for layer in root.children():
  if layer.name().startswith('PRISM'):
    prefix = layer.name()[-6:-4]
    params = {'INPUT_RASTER': layer.name(), 'RASTER_BAND': 1, 'INPUT_VECTOR': 'Zip_Codes
    processing.run("qgis:zonalstatistics", params)
```



14. Once the processing finishes, right-click on the `Zip_Codes` layer and select Open Attribute Table.



15. You will see 12 new columns added to the table with custom prefixes and mean precipitation values extracted from the raster layers.



	SHAPE_Area	01_mean	02_mean	03_mean	04_mean	05_mean	06_mean	07_mean
1	1073807096.153...	107.8083753585...	252.4269962310...	317.0766296386...	149.6217517852...	84.08187580108...	48.53562450408...	0.4330000064
2	494363984.2530...	91.09633382161...	236.6333363850...	296.3699951171...	151.8739980061...	88.78100077311...	44.70366541544...	0.8506666620
3	14303083.99095...	81.83193494694...	223.7776461243...	219.2651661035...	113.1304836900...	70.26504115248...	44.91152476605...	0.4420123939
4	57586161.58822...	81.76301788639...	223.3025413447...	219.4298227087...	113.2262873682...	70.79083978080...	45.09915413279...	0.4630597414
5	760918899.3510...	70.75320053100...	119.7481994628...	138.5017990112...	76.47879791259...	50.08860015869...	11.24139995574...	0.0351999999
6	195618382.1055...	93.29000091552...	179.7539978027...	168.4120025634...	115.9720001220...	68.18099975585...	25.58400058746...	0.1530000008
7	26810071.07868...	98.28800201416...	185.0760040283...	200.2779998779...	95.54900360107...	43.35300064086...	36.07699966430...	0.3889999985
8	1592781724.857...	89.09500045776...	161.3963012695...	157.4498016357...	102.0388015747...	65.19649963378...	19.37969961166...	0.0141000002
9	474765168.0252...	93.11800003051...	231.1635055541...	262.8980026245...	130.2225036621...	83.54949951171...	43.27250099182...	0.4389999955
10	997643457.2268...	103.1832842145...	237.3325718470...	271.5224260602...	131.1682859148...	83.99671500069...	47.24771445138...	0.1711428555
11	40203850.39062...	133.7976506796...	248.5725700222...	261.0250823031...	132.2889723847...	60.2171878072824	38.04951974192...	
12	1600401561.978...	74.82566663953...	199.1874474419...	193.5017801920...	114.9111124674...	72.78977796766...	50.10344441731...	0.7448888884
13	2717657368.109...	78.38286692301...	190.957666015625	223.0772003173...	121.1835998535...	68.46233393351...	54.34153289794...	0.5872666696
14	837983705.8085...	123.5975017547...	271.8967437744...	347.3422470092...	154.5755004882...	77.27924919128...	53.59574985504...	
15	53285059.10517...	127.4151574857...	311.3339509596...	389.4310547820...	165.2844346280...	72.86738635262...	65.95637153431...	0.4411061961
16	785013319.2393...	112.6231664021...	248.8568293253...	283.8514963785...	134.4703318277...	82.8693339029948	50.56400044759...	

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Building a Python Plugin (QGIS3)

Plugins are a great way to extend the functionality of QGIS. You can write plugins using Python that can range from adding a simple button to sohpisticated toolkits. This tutorial will outline the process involved in setting up your development environment, designing the user interface for a plugin and writing code to interact with QGIS. Please review the *Getting Started With Python Programming (QGIS3)* (getting_started_with_pyqgis.html) tutorial to get familiar with the basics.

Overview of the Task

We will develop a simple plugin called `Save Attributes` that will allow users to pick a vector layer and write its attributes to a CSV file.

Get the Tools

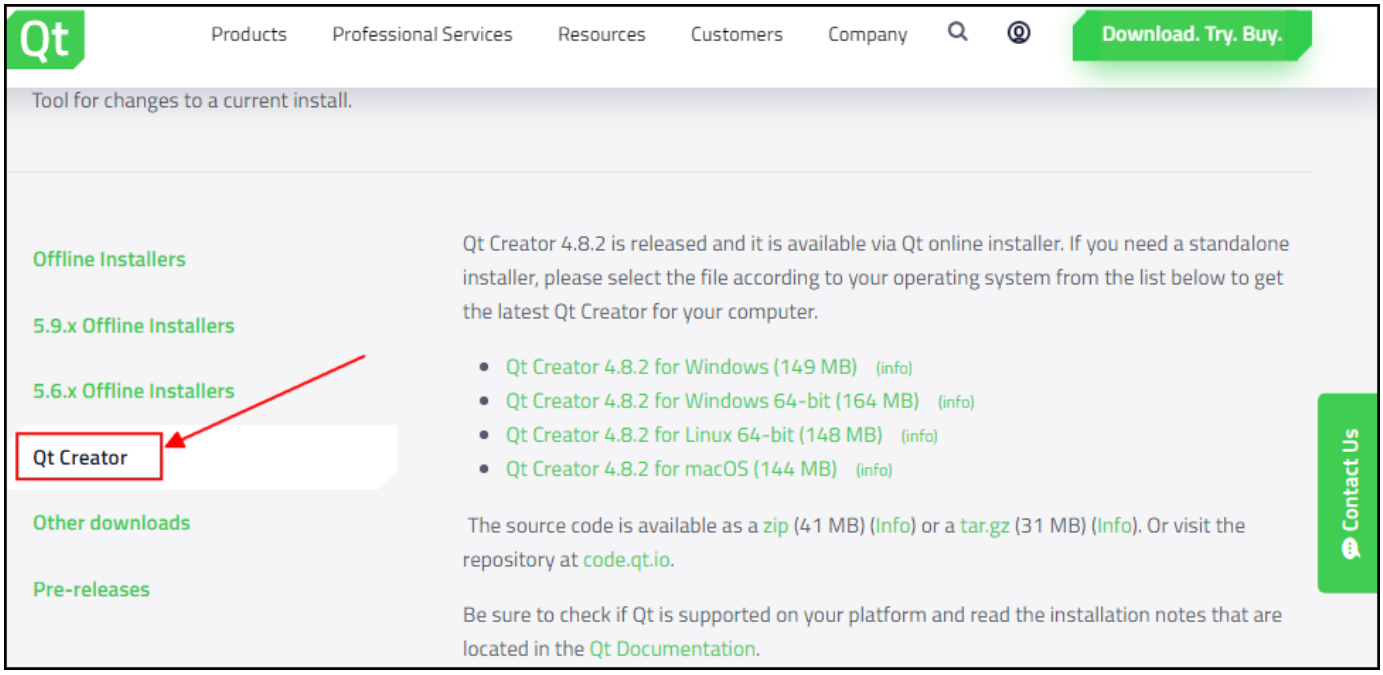
Qt Creator

Qt (http://www.qt.io/) is a software development framework that is used to develop applications that run on Windows, Mac, Linux as well as various mobile operating systems. QGIS itself is written using the Qt framework. For plugin development, we will use an application called Qt Creator (https://doc.qt.io/qt-5/qtdesigner-manual.html) to design the interface for our plugin.

Download and install the Qt Creator installer from Qt Offline Installers (https://www.qt.io/offline-installers). Make sure you select **Qt Creator** on the download page. Note that you will have to create a free Qt account to install the package.

Note

OSGeo4w installer for QGIS on Windows include a copy of **Qt Designer** program which is a lightweight version of **Qt Creator** and perfectly suitable for building plugins. You may skip downloading Qt Creator and use it instead from C:\OSGeo4W64\bin\qgis-designer.



Python Bindings for Qt

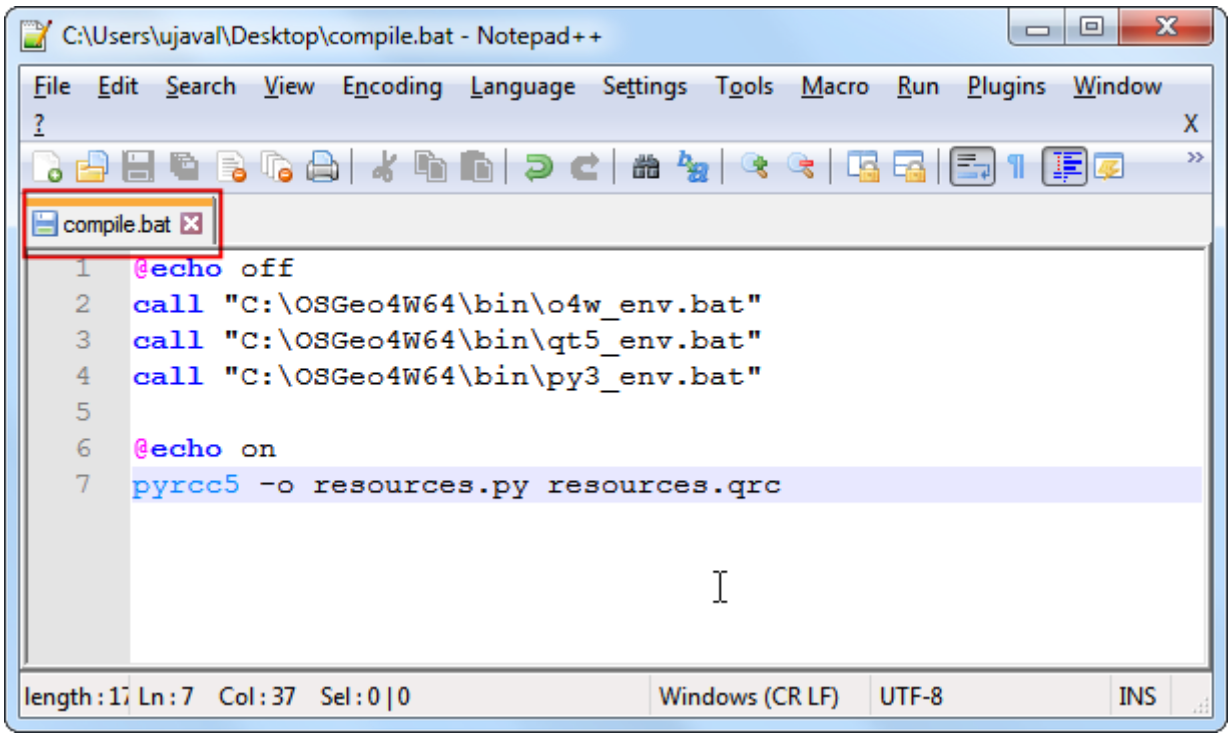
Since we are developing the plugin in Python, we need to install the python bindings for Qt. The method for installing these will depend on the platform you are using. For building plugins we need the `pyrcc5` command-line tool.

Windows

Relevant pyhon bindings are included in the QGIS install on Windows. But to use them from the plugin folder, we need to indicate the path to the QGIS install.

Create a Windows Batch file (.bat extension) with the following content and save it on your computer as `compile.bat` . We will later copy this file to the plugin folder. If you installed QGIS at a different path, replace the `C:\OSGeo4W64\bin\` with your path.

```
@echo off
call "C:\OSGeo4W64\bin\o4w_env.bat"
call "C:\OSGeo4W64\bin\qt5_env.bat"
call "C:\OSGeo4W64\bin\py3_env.bat"

@echo on
pyrcc5 -o resources.py resources.qrc
```



Mac

Install the Homebrew (http://brew.sh) package manager. Install `PyQt` package by running the following command:

```
brew install pyqt
```

Linux

Depending on your distribution, find and install the `python-qt5` package. On Ubuntu and Debian-based distributions, you can run the following command:

```
sudo apt-get install python-qt5
```

Note

You may find that QGIS has already installed this package.

A Text Editor or a Python IDE

Any kind of software development requires a good text editor. If you already have a favorite text editor or an IDE (Integrated Development Environment), you may use it for this tutorial. Otherwise, each platform offers a wide variety of free or paid options for text editors. Choose the one that fits your needs.

This tutorial uses Notepad++ editor on Windows.

Windows

Notepad++ (http://notepad-plus-plus.org/) is a good free editor for windows. Download and install the Notepad++ editor (https://notepad-plus-plus.org/download/).

Note

If you are using Notepad++, makes sure to go to Settings ‣ Preferences ‣ Tab Settings and enable Replace by space. Python is very sensitive about whitespace and this setting will ensure tabs and spaces are treated properly.

Plugin Builder plugin

There is a helpful QGIS plugin named `Plugin Builder` which creates all the necessary files and the boilerplate code for a plugin. Find and install the `Plugin Builder` plugin. See *Using Plugins* (../using_plugins.html) for more details on how to install plugins.

Plugins Reloader plugin

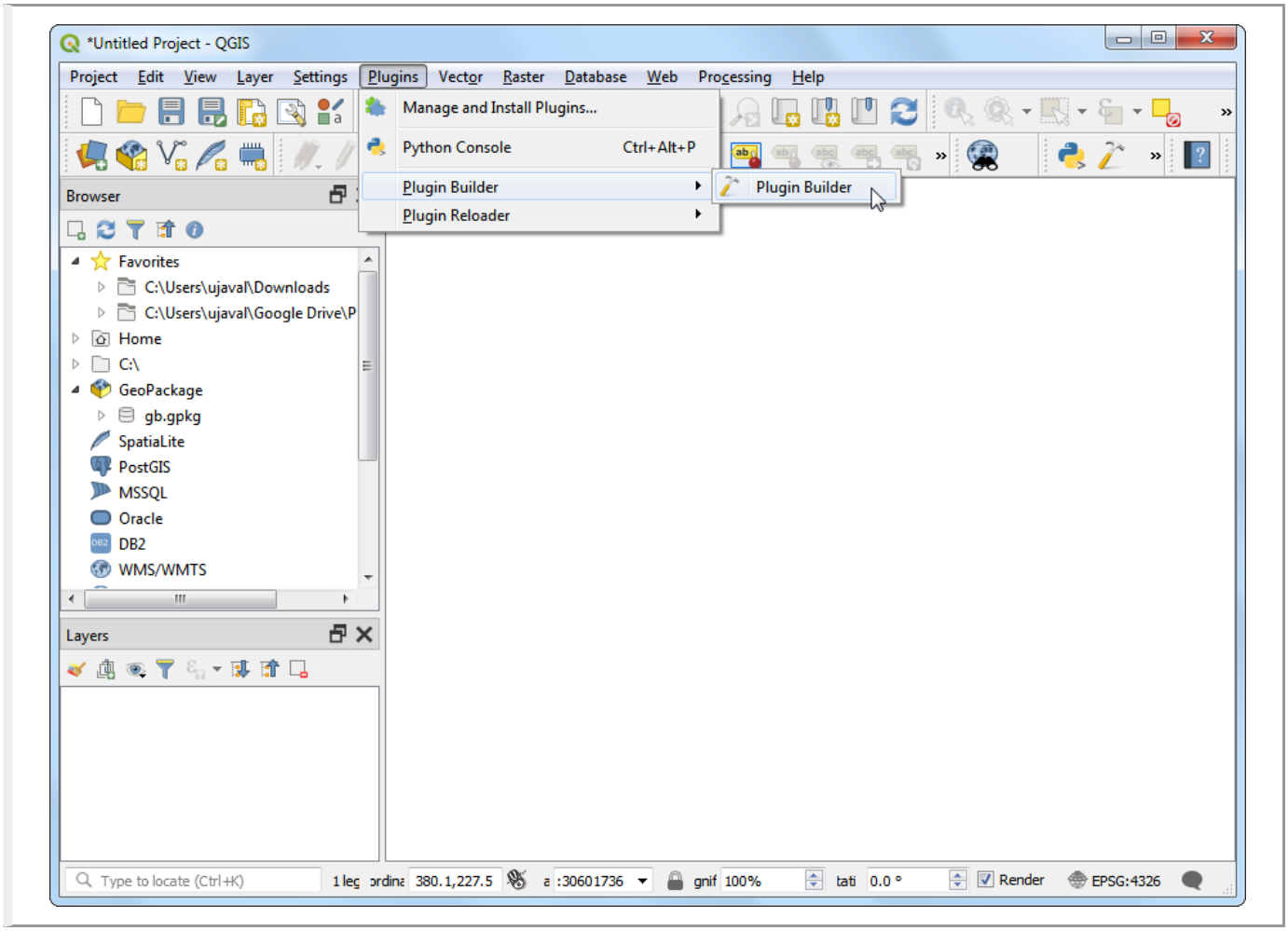
This is another helper plugin which allows iterative development of plugins. Using this plugin, you can change your plugin code and have it reflected in QGIS without having to restart QGIS every time. Find and install the `Plugin Reloader` plugin. See *Using Plugins* (../using_plugins.html) for more details on how to install plugins.

Note

Plugin Reloader is an experimental plugin. Make sure you have checked Show also experimental plugins in Plugin Manager settings if you cannot find it.

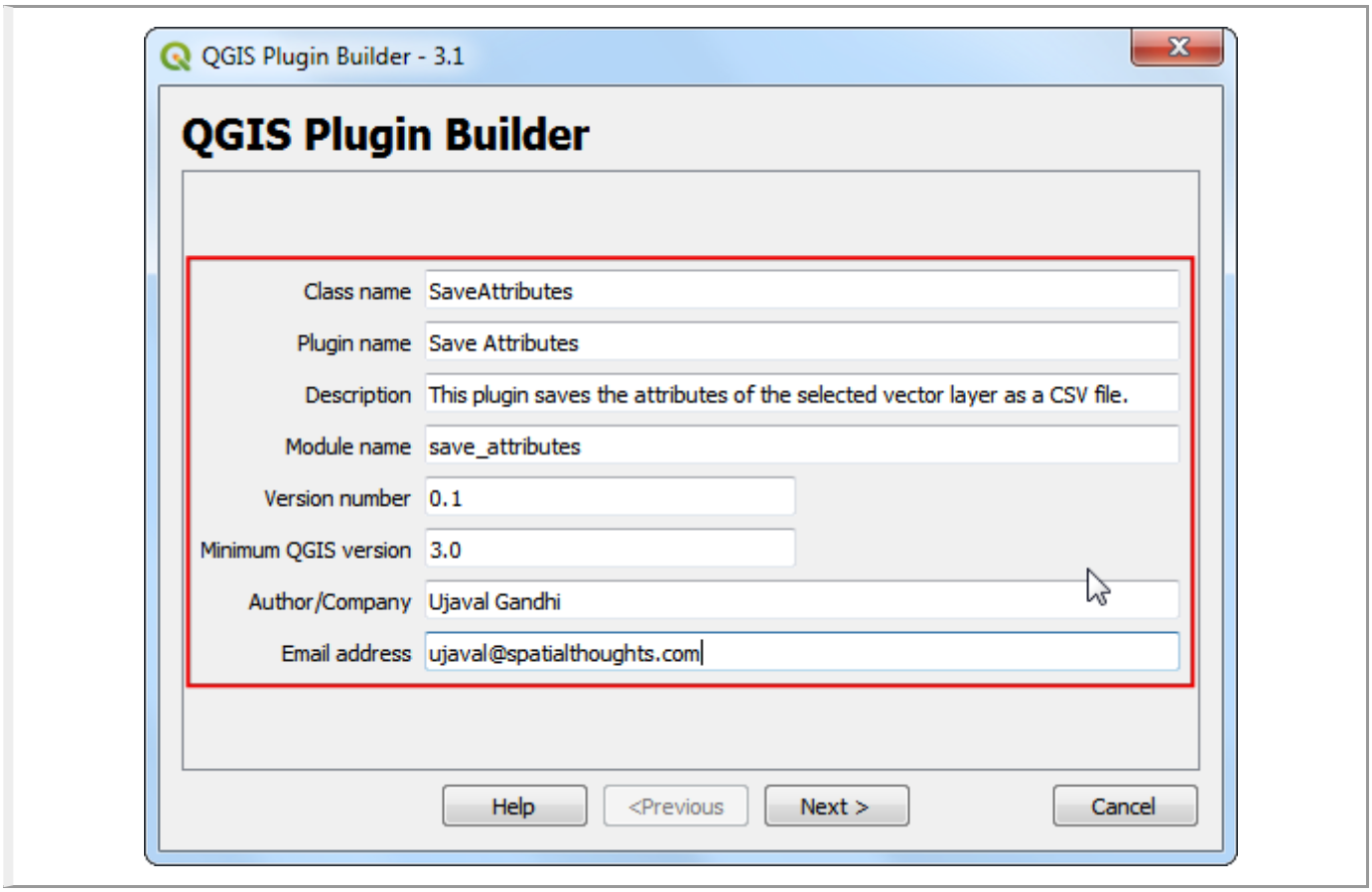
Procedure

1. Open QGIS. Go to Plugins ‣ Plugin Builder ‣ Plugin Builder.

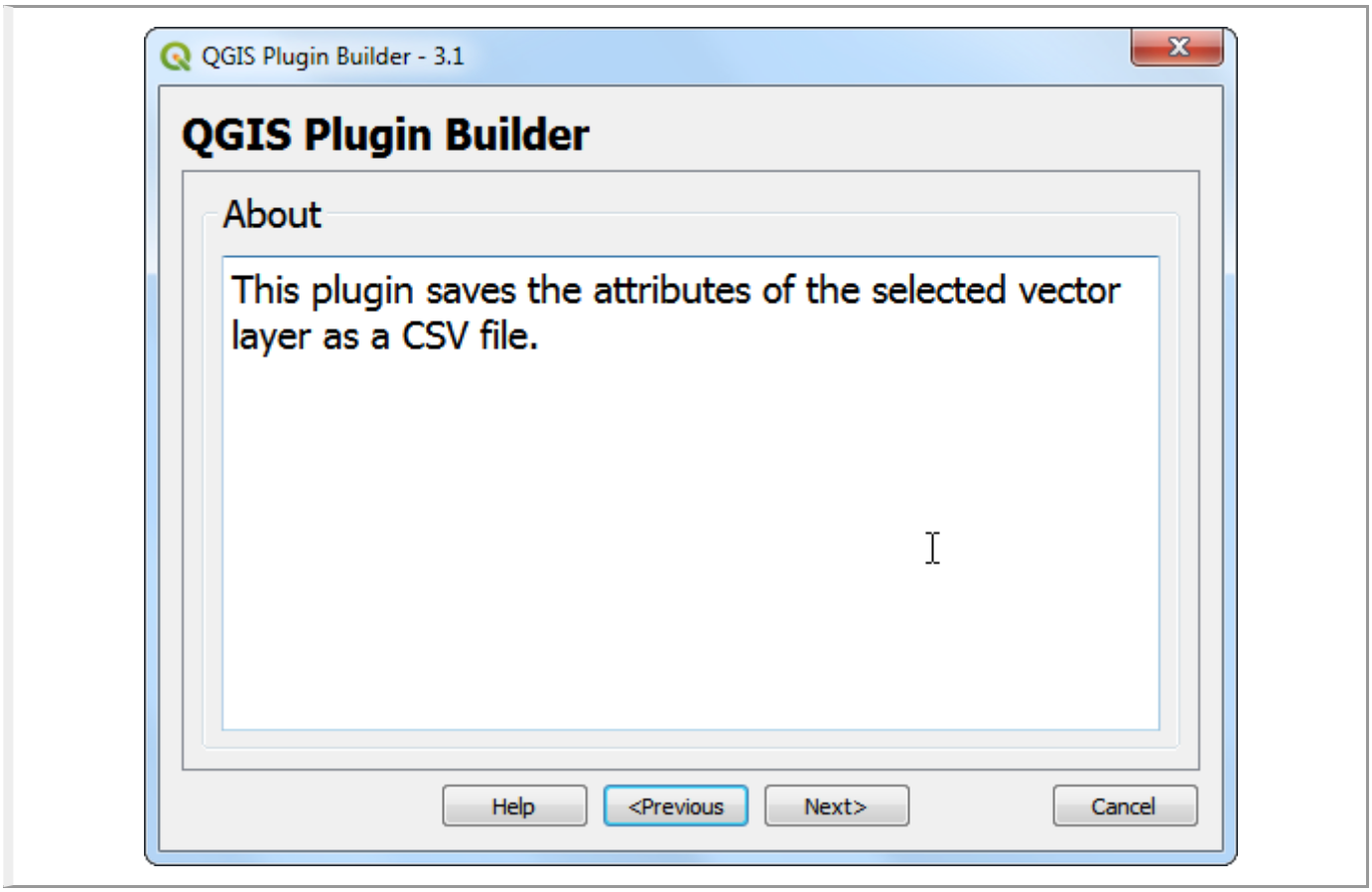


2. You will see the QGIS Plugin Builder dialog with a form. You can fill the form with details relating to our plugin. The Class name will be the name of the Python Class containing the logic of the plugin. This will also be the name of the folder containing all the plugin files. Enter `SaveAttributes` as the class name. The Plugin name is the name under which your plugin will appear in the Plugin Manager. Enter the name as `Save Attributes` . Add a description in the Description field. The Module name will

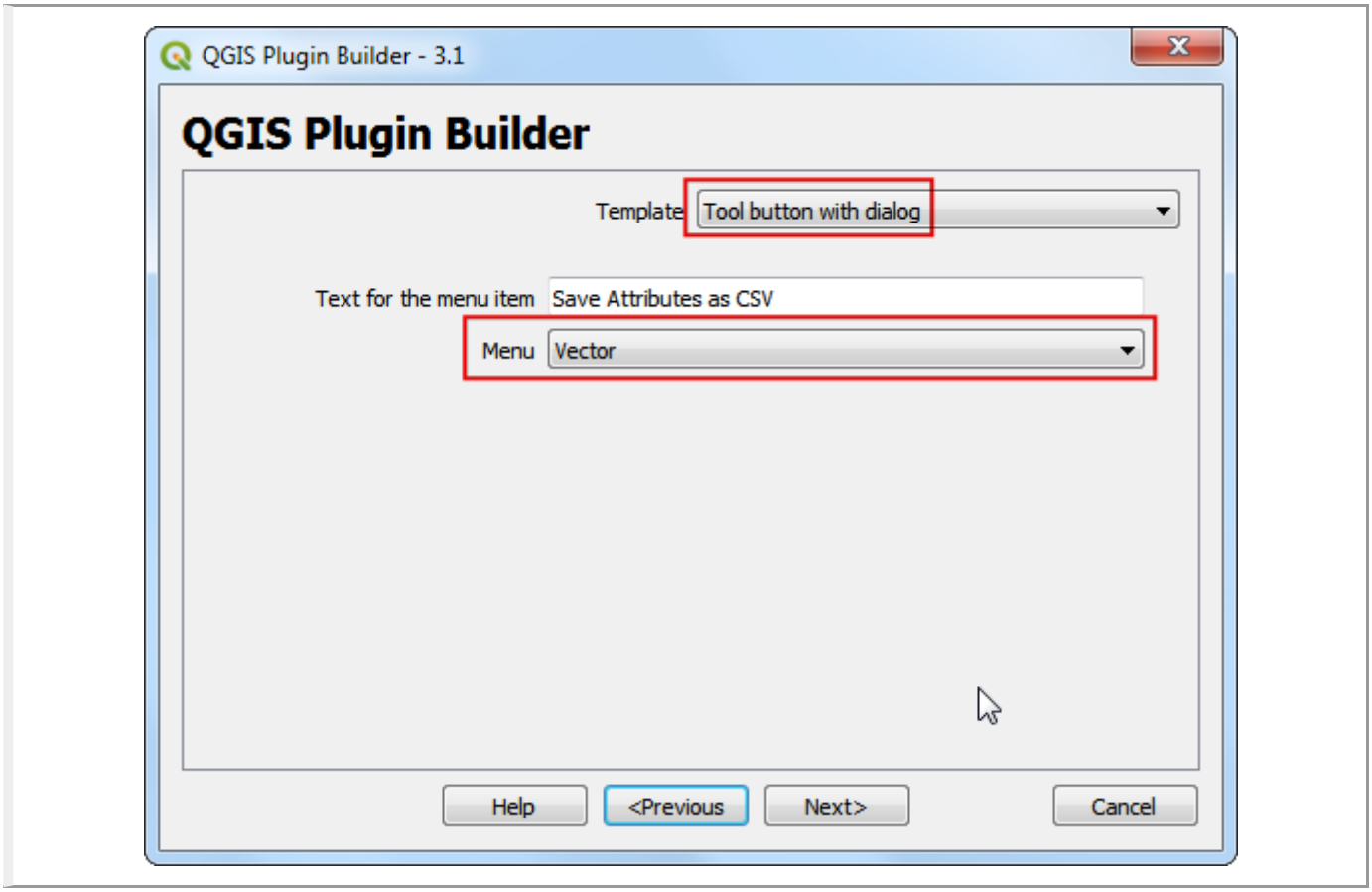
be the name of the main python file for the plugin. Enter it as `save_attributes` . Leave the version numbers as they are and enter your name and email address in the appropriate fields. Click Next.



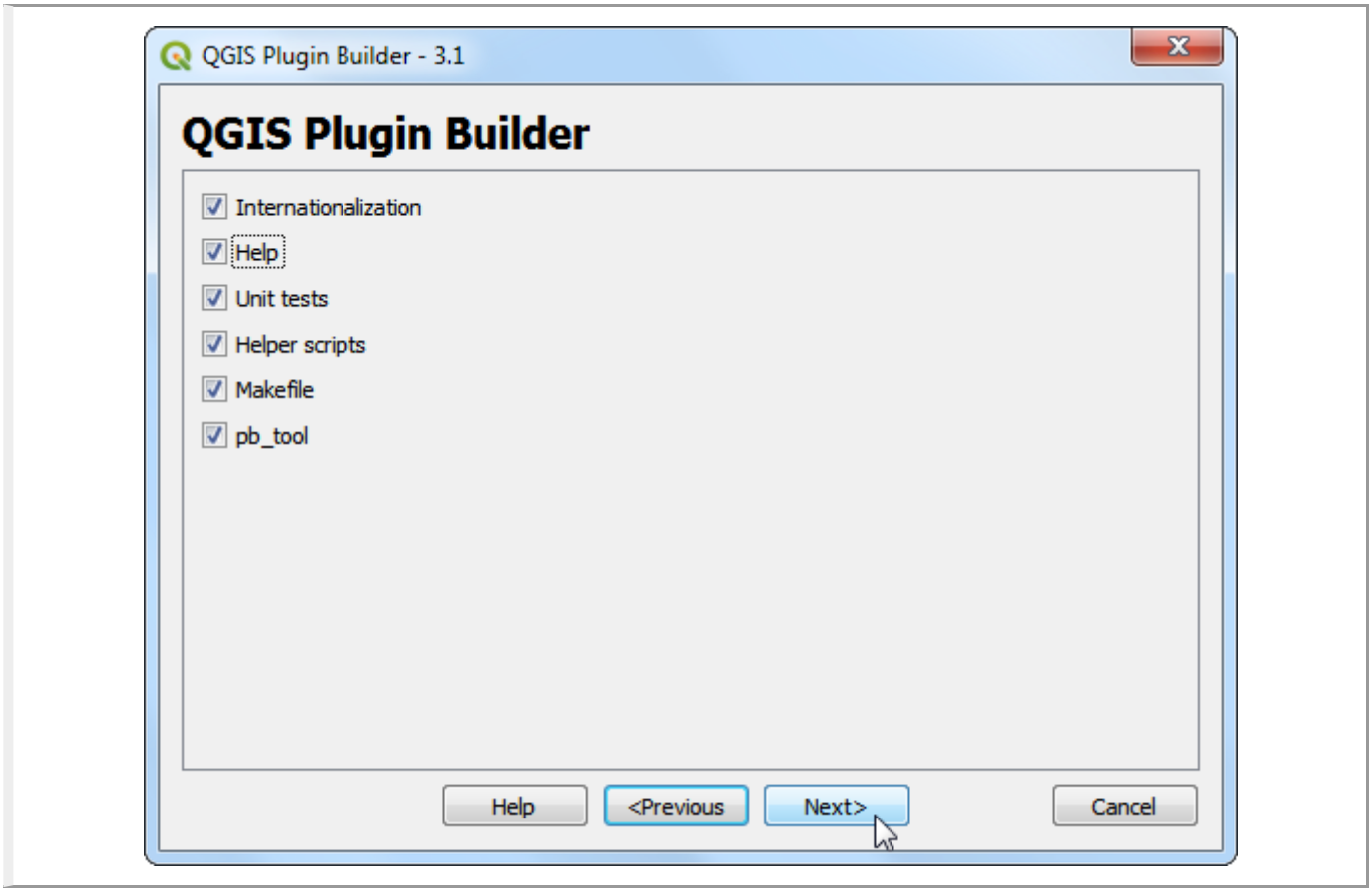
3. Enter a brief description of the plugin for the About dialog and click Next.



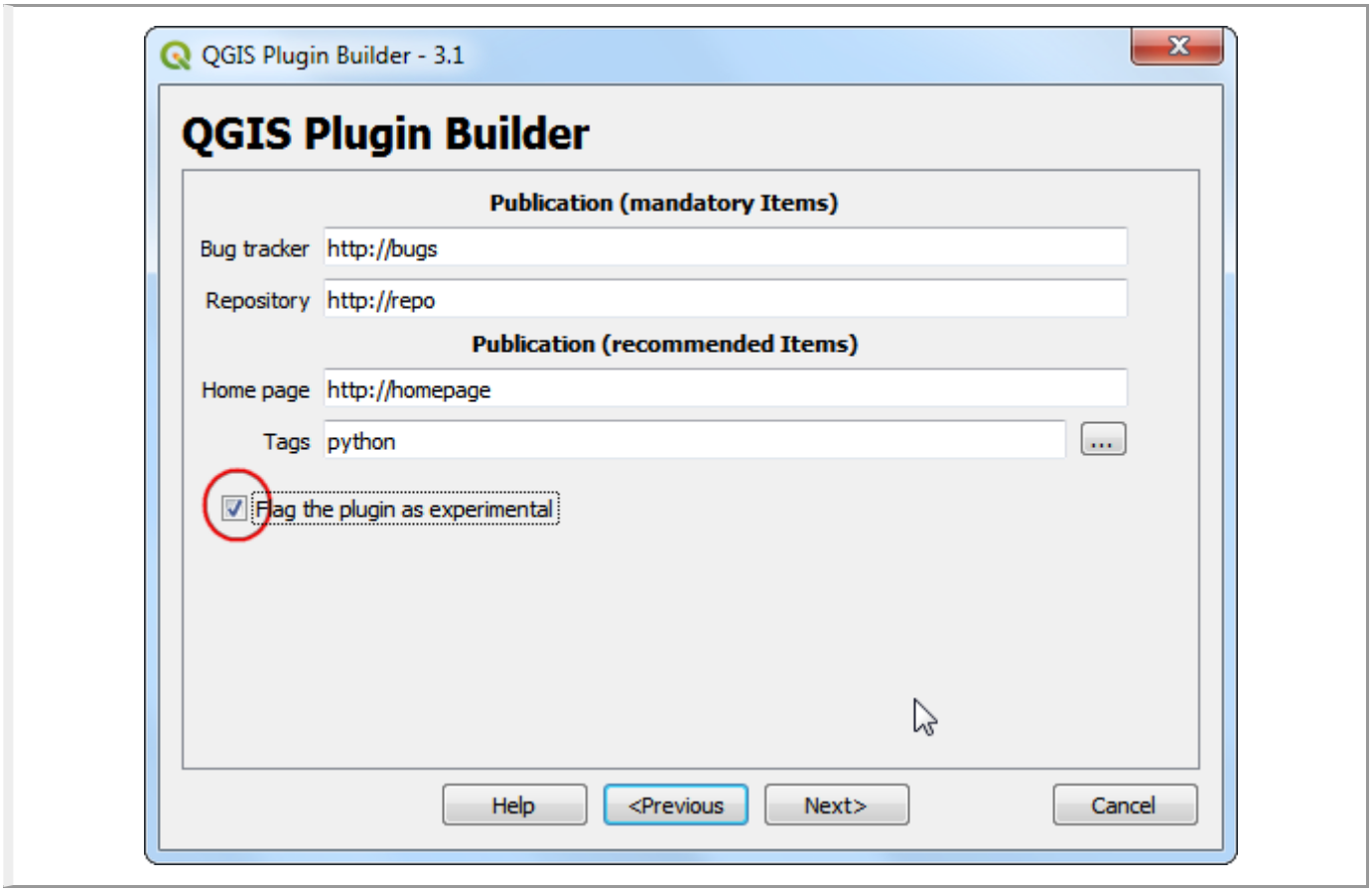
4. Select the `Tool button with dialog` from the Template selector`. The Text for menu item value will be how the users will find your plugin in QGIS menu. Enter it as `Save Attributes as CSV`. The Menu field will decide where your plugin item is added in QGIS. Since our plugin is for vector data, select `Vector`. Click Next.



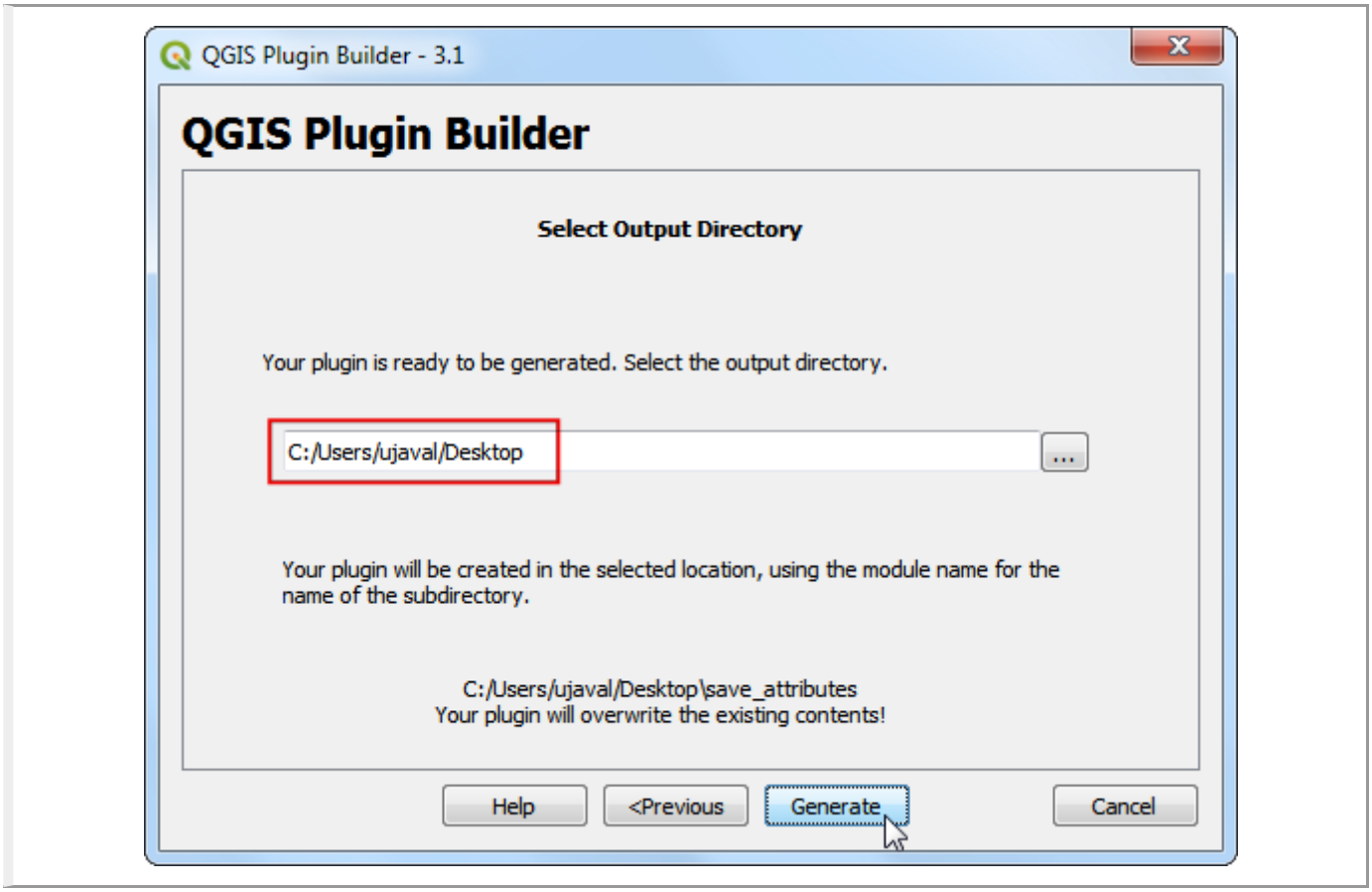
5. Plugin builder will prompt you for the type of files to generate. Keep the default selection and click Next.



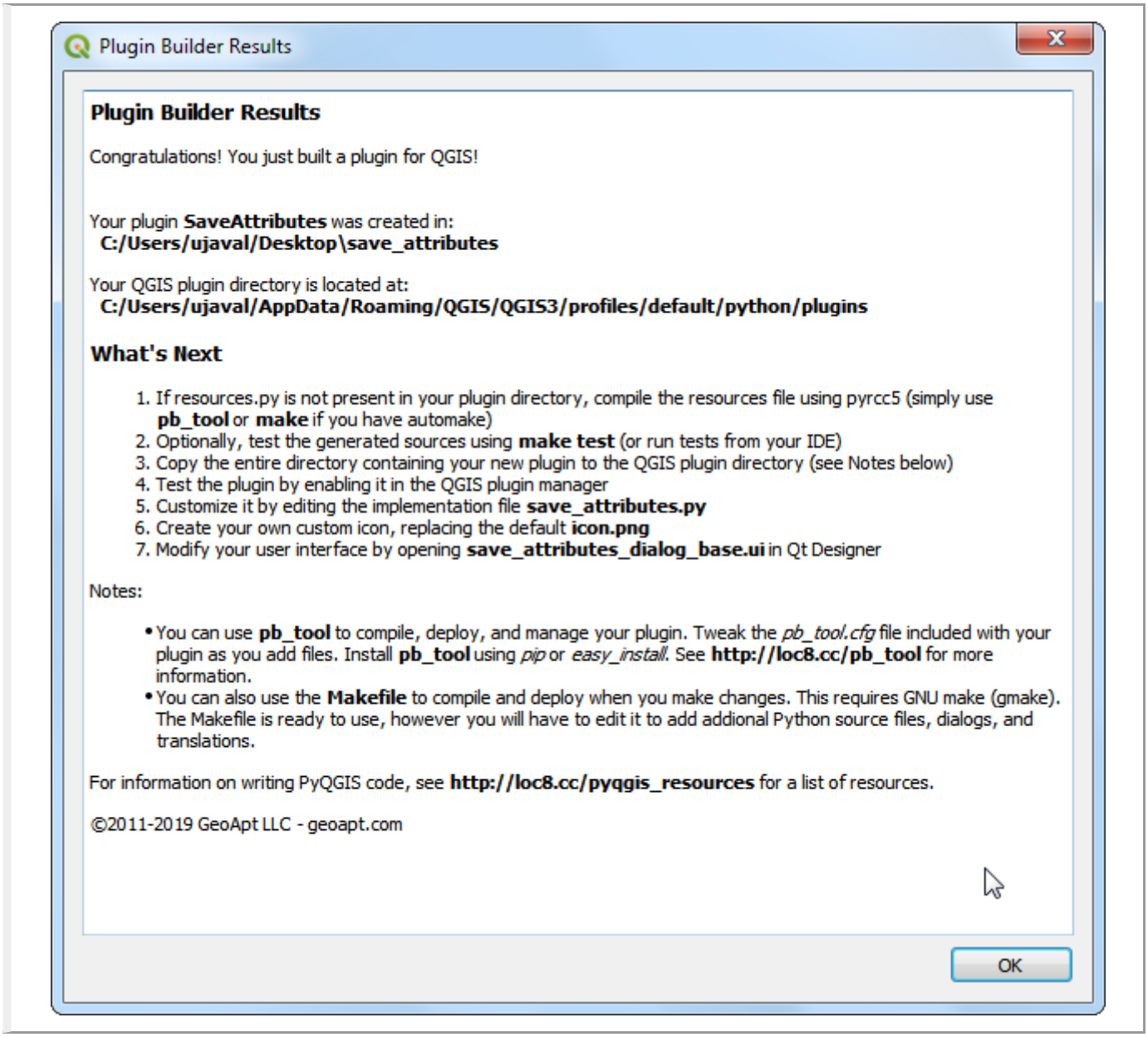
6. As we do not intend to publish the plugin, you may leave the Bug tracker, Repository and Home page values to default. Check the Flag the plugin as experimental box at the bottom and click Next.



7. You will be prompted to choose a directory for your plugin. For now, save it to a directory you can locate easily on your computer and click Generate.



8. Next, press the generate button. You will see a confirmation dialog once your plugin template is created.



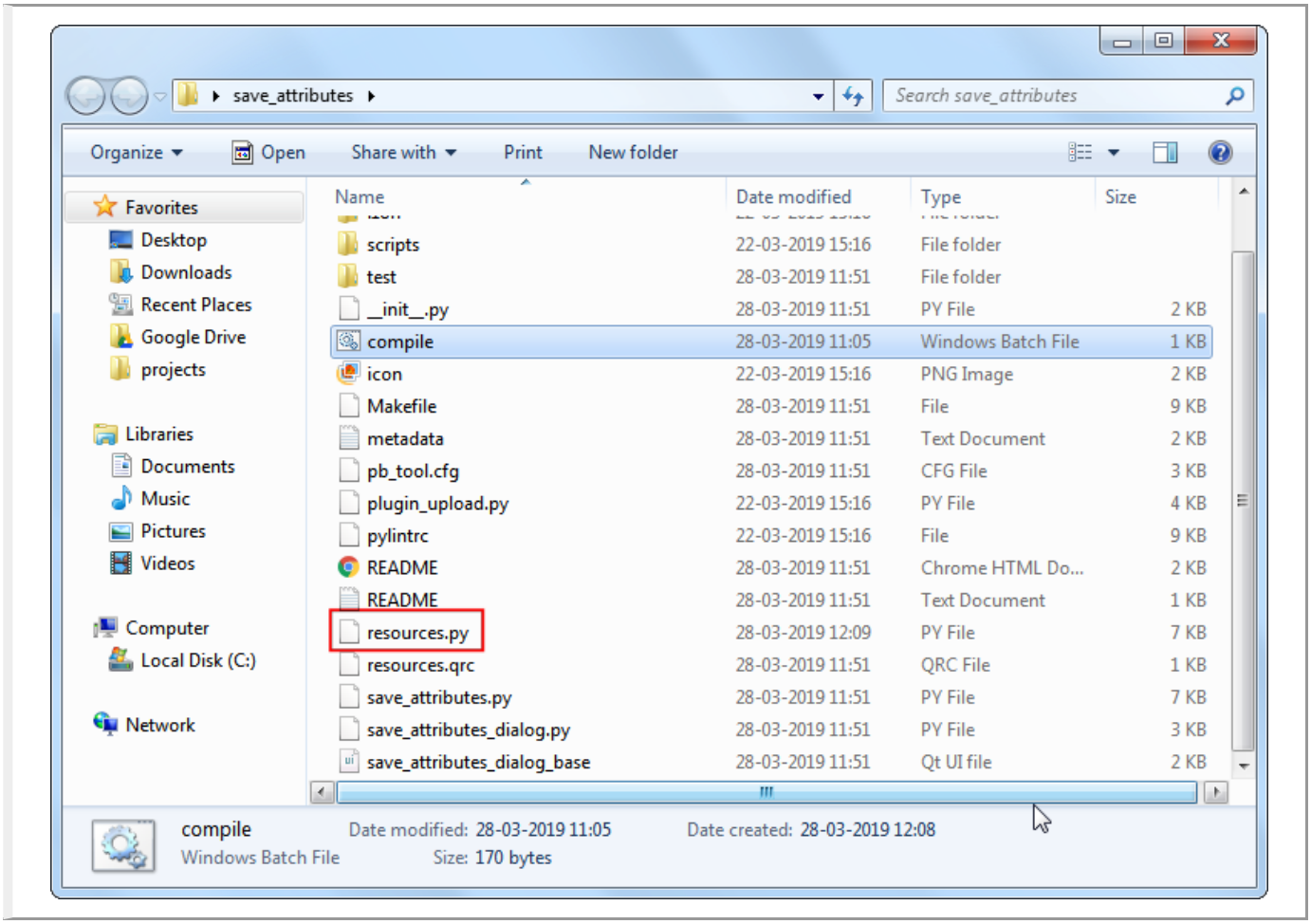
Note

You may get a prompt saying that pyrcc5 is not found in the path. You can ignore this message.

9. Before we can use the newly created plugin, we need to compile the `resources.qrc` file that was created by Plugin Builder. This file is part of the Qt Resource System (https://doc.qt.io/qt-5/resources.html) which references all binary files used in the plugin. For this plugin, it will only have the plugin icon. Compiling this file generates application code that can be used in the plugin independent which platform the plugin is being run. Follow the platform specific instruction for this step.

Windows

You can now copy the `compile.bat` file (created during the *Python Bindings for Qt* section at the start) to the plugin folder. Once copied, double-click the file to run it. If the run was successful, you will see a new file called `resources.py` in the folder.



Note

If this step fails, you can launch `cmd.exe` and browse to the plugin folder using `cd` command. Run the Batch file by running `compile.bat` to see the error.

Mac and Linux

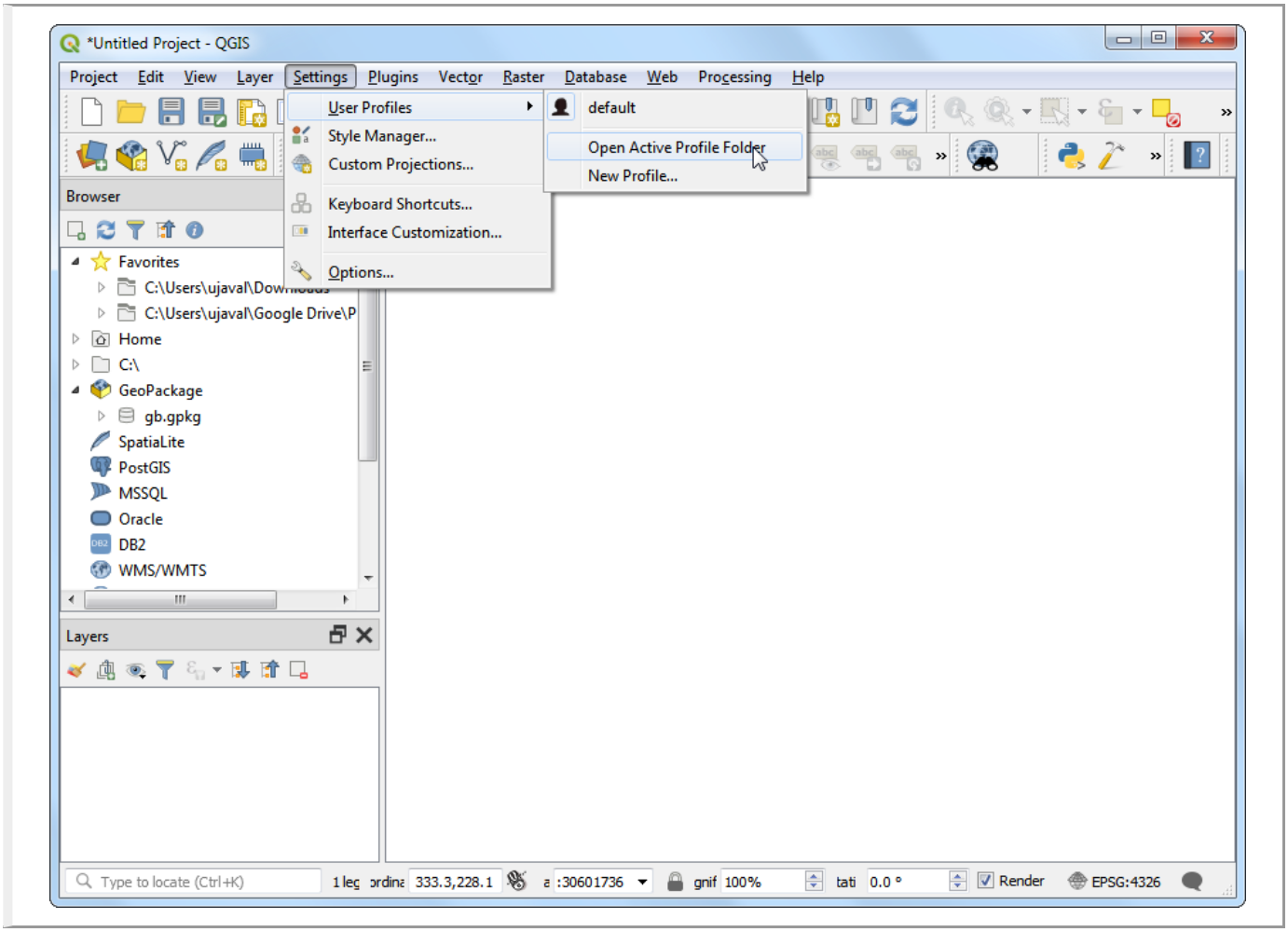
You will need to install `pb_tool` first. Open a Terminal and install it via pip.

```
sudo pip3 install pb_tool
```

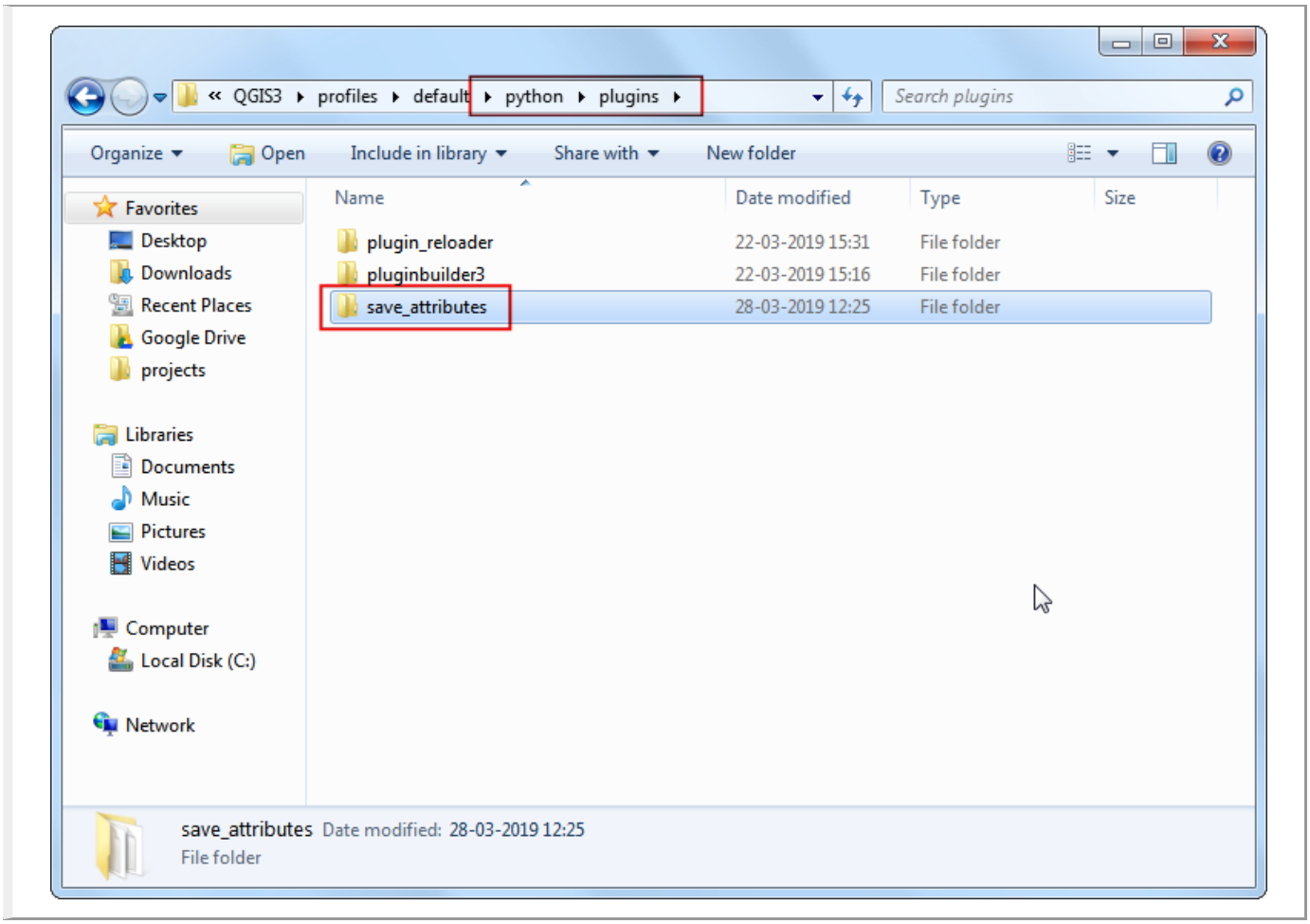
Open a Terminal and go to the plugin directory and type `pb_tool compile` . This will run the `pyrcc5` command that we had installed as part *Python Bindings for Qt* section.

```
pb_tool compile
```

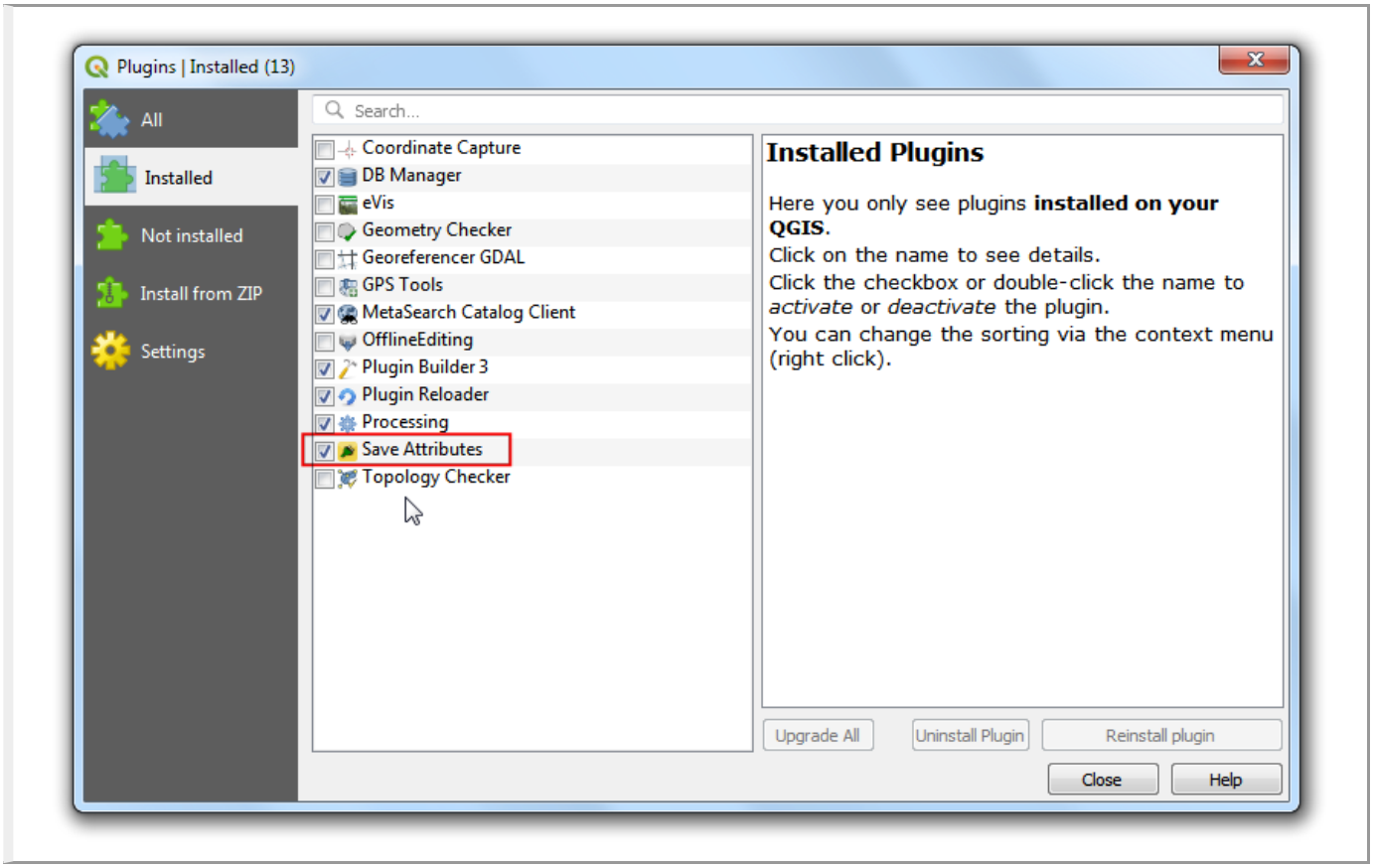
10. Plugins in QGIS are stored in a special folder. We must copy our plugin directory to that folder before it can be used. In QGIS, locate your current profile folder by going to Settings ‣ User Profiles ‣ Open Active Profile Folder.



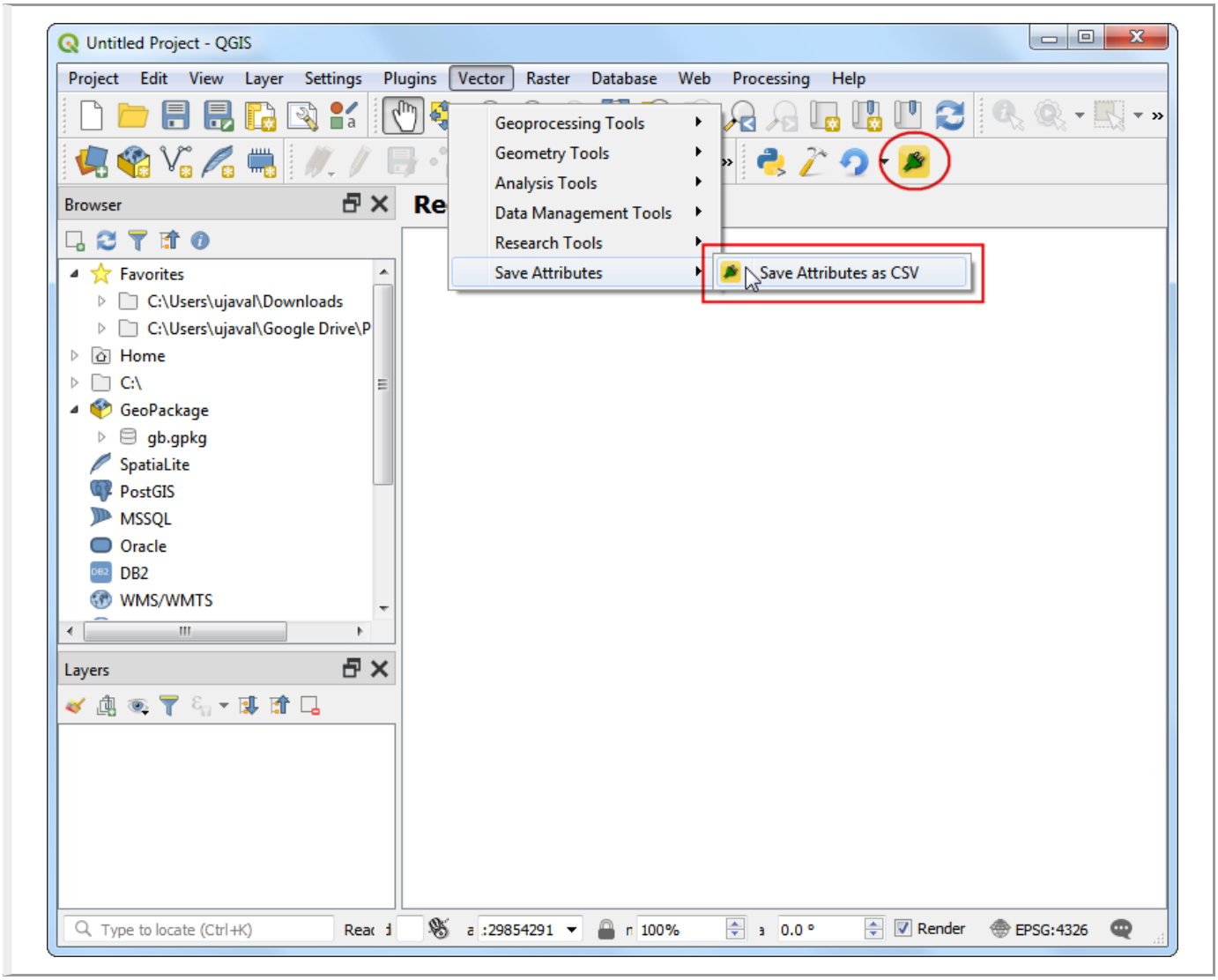
11. In the profile folder, copy the plugin folder to python ‣ plugins subfolder.



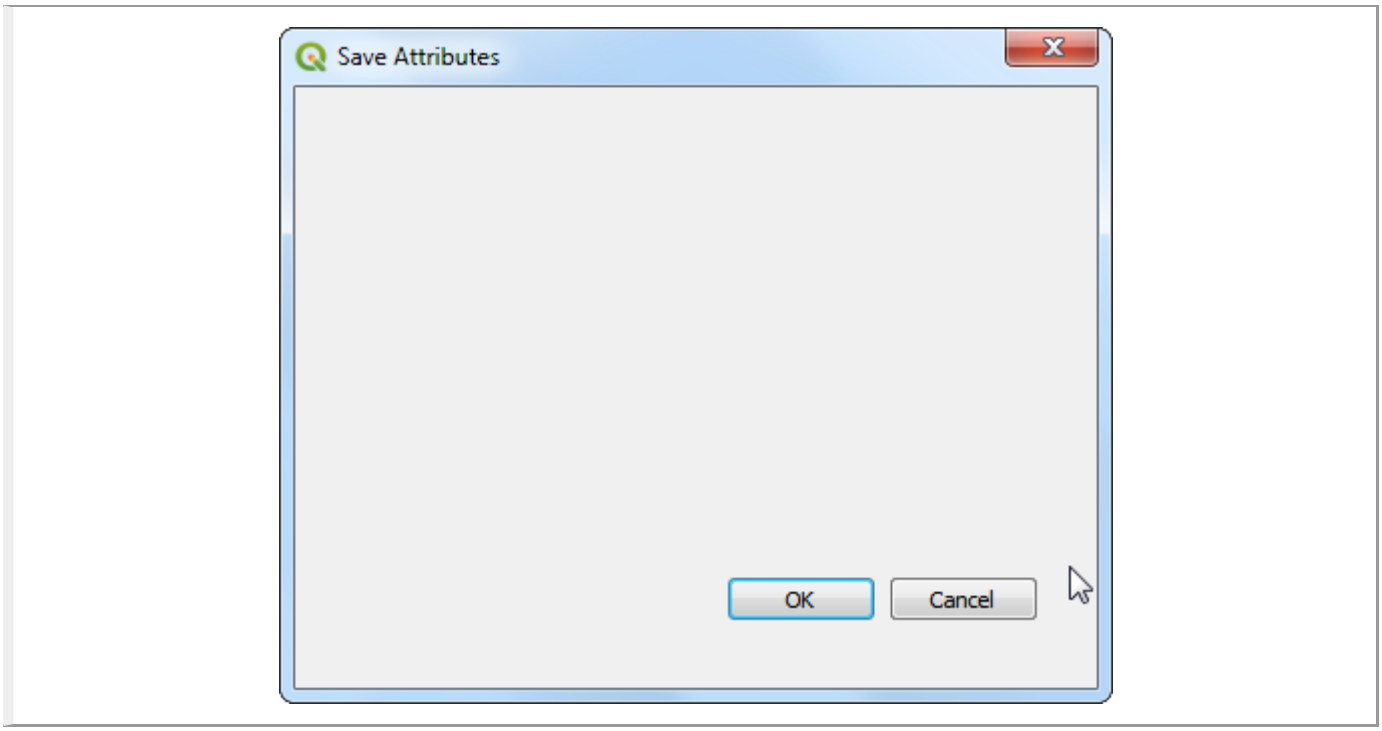
12. Now we are ready to have a first look at the brand new plugin we created. Close QGIS and launch it again. Go to Plugins ‣ Manage and Install plugins and enable the `Save Attributes` plugin in the Installed tab.



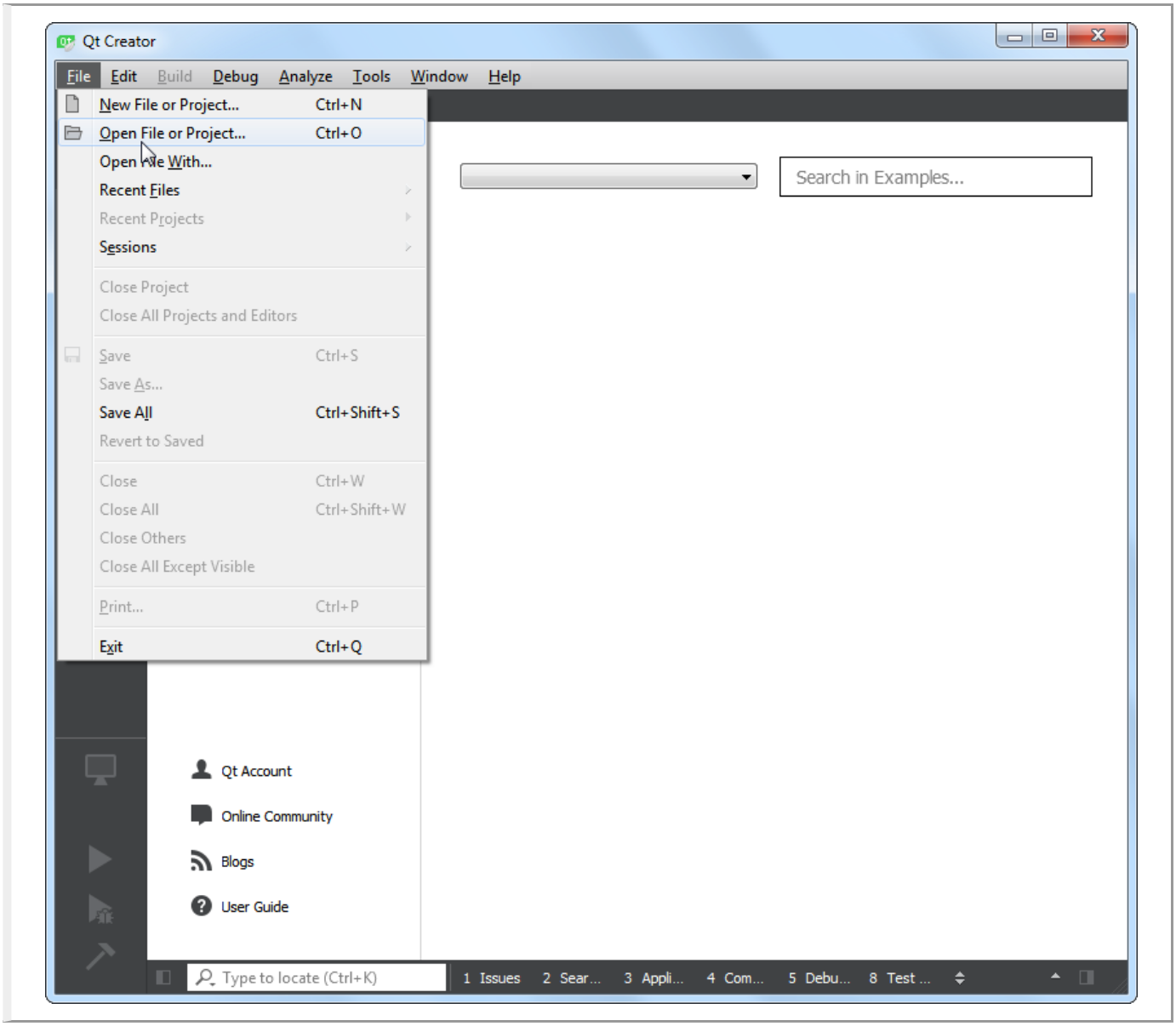
13. You will notice that there is a new icon in the plugin toolbar and a new menu entry under Vector ‣ Save Attributes ‣ Save Attributes as CSV`. Select it to launch the plugin dialog.



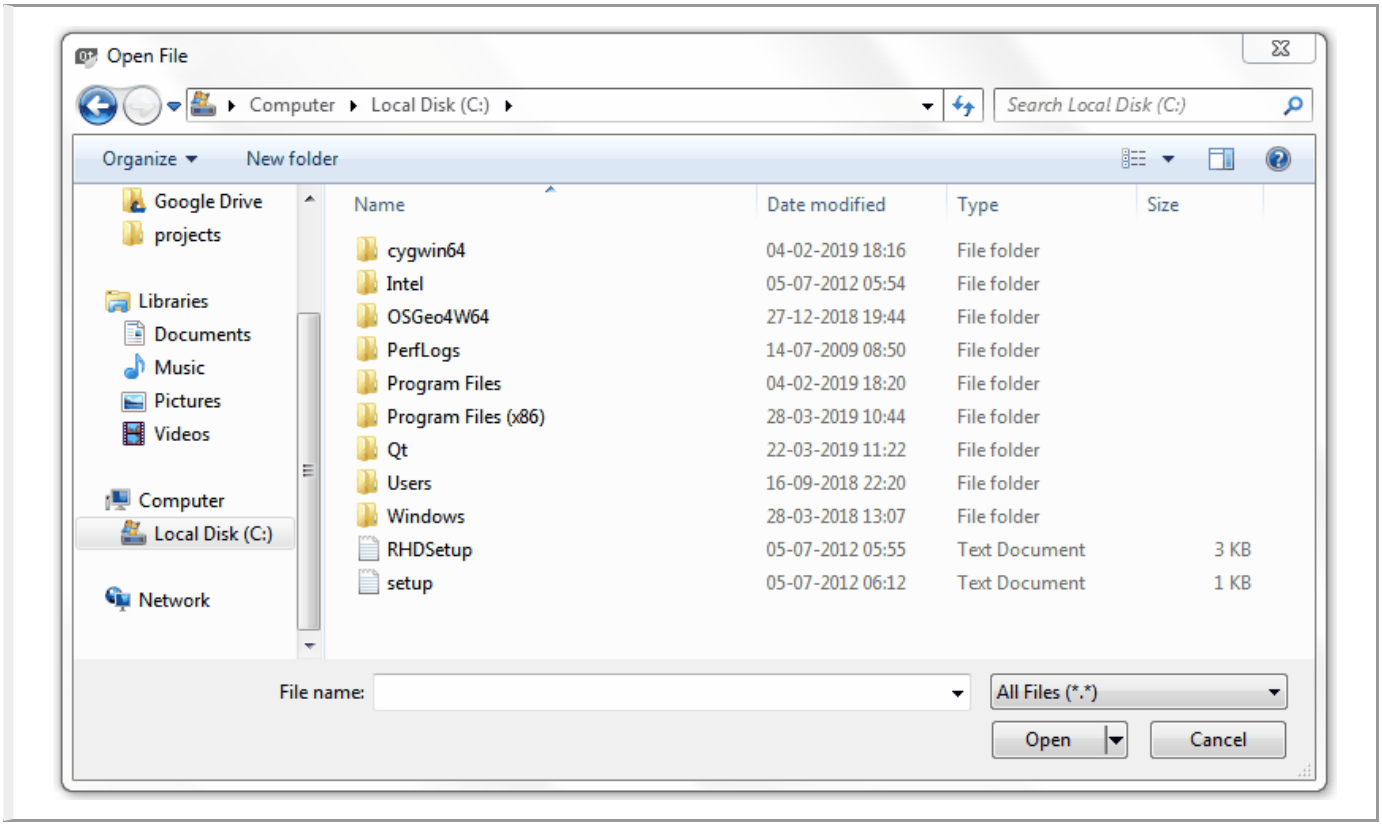
14. You will notice a new blank dialog named Save Attributes. Close this dialog.



15. We will now design our dialog box and add some user interface elements to it. Open the `Qt Creator` program and go to File ‣ Open File or Project.



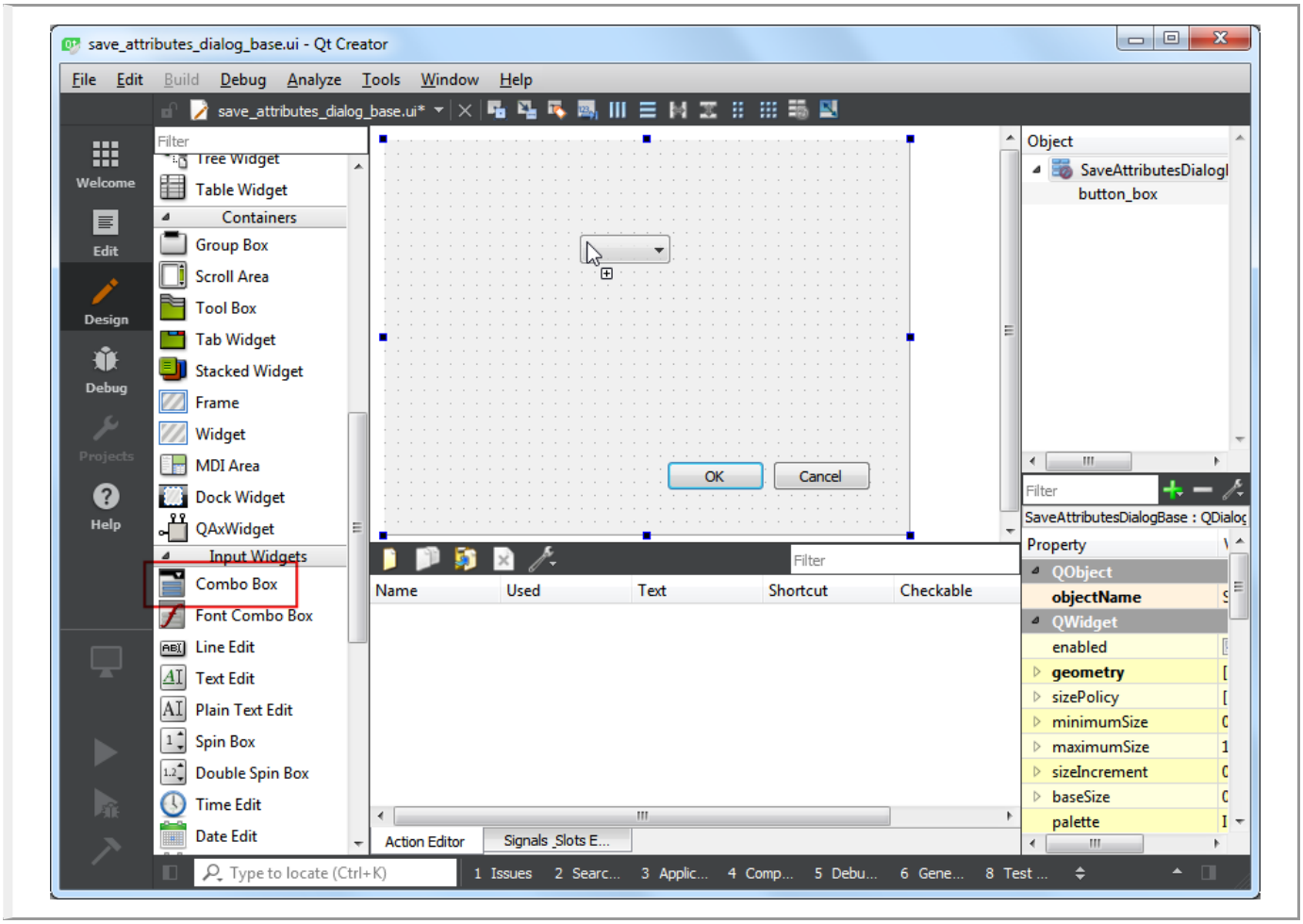
16. Browse to the plugin directory and select the `save_attributes_dialog_base.ui` file. Click Open.



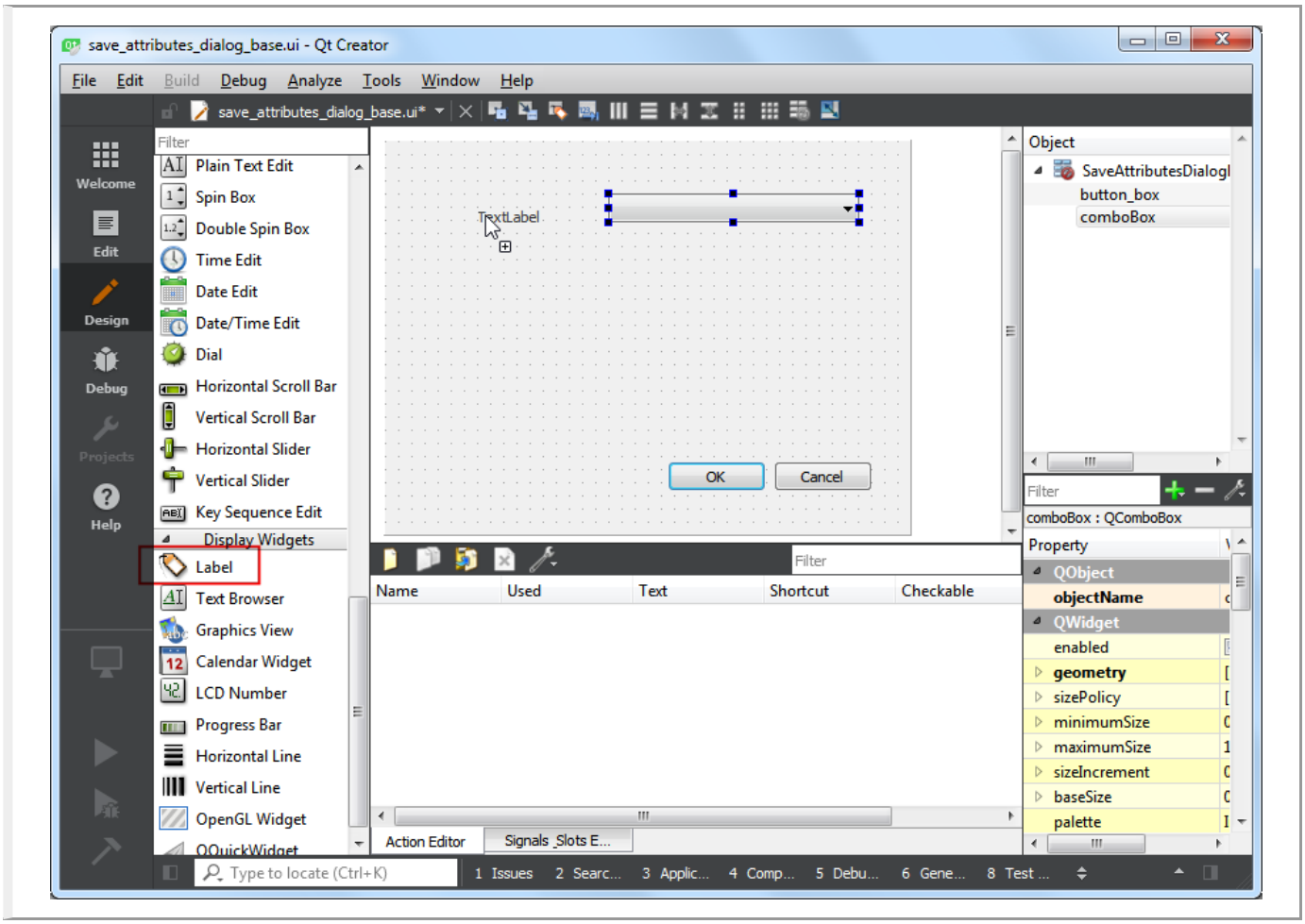
Note

Windows hides the `AppData` folder so you may not see it in the file selector dialog. You can enter `AppData` in the File name prompt from its parent directory to open it.

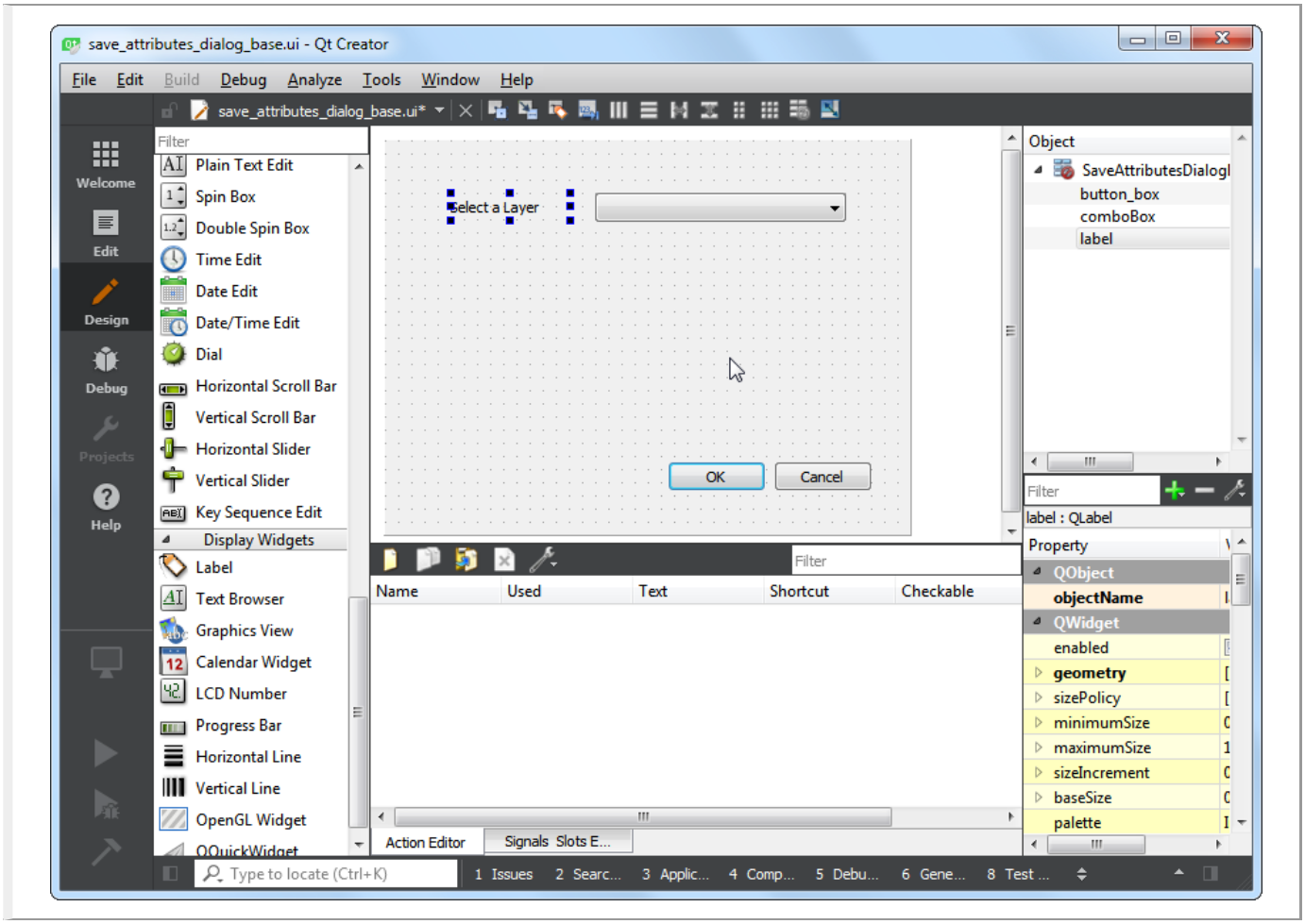
17. You will see the blank dialog from the plugin. You can drag-and-drop elements from the left-hand panel on the dialog. We will add a Combo Box type of Input Widgets. Drag it to the plugin dialog.



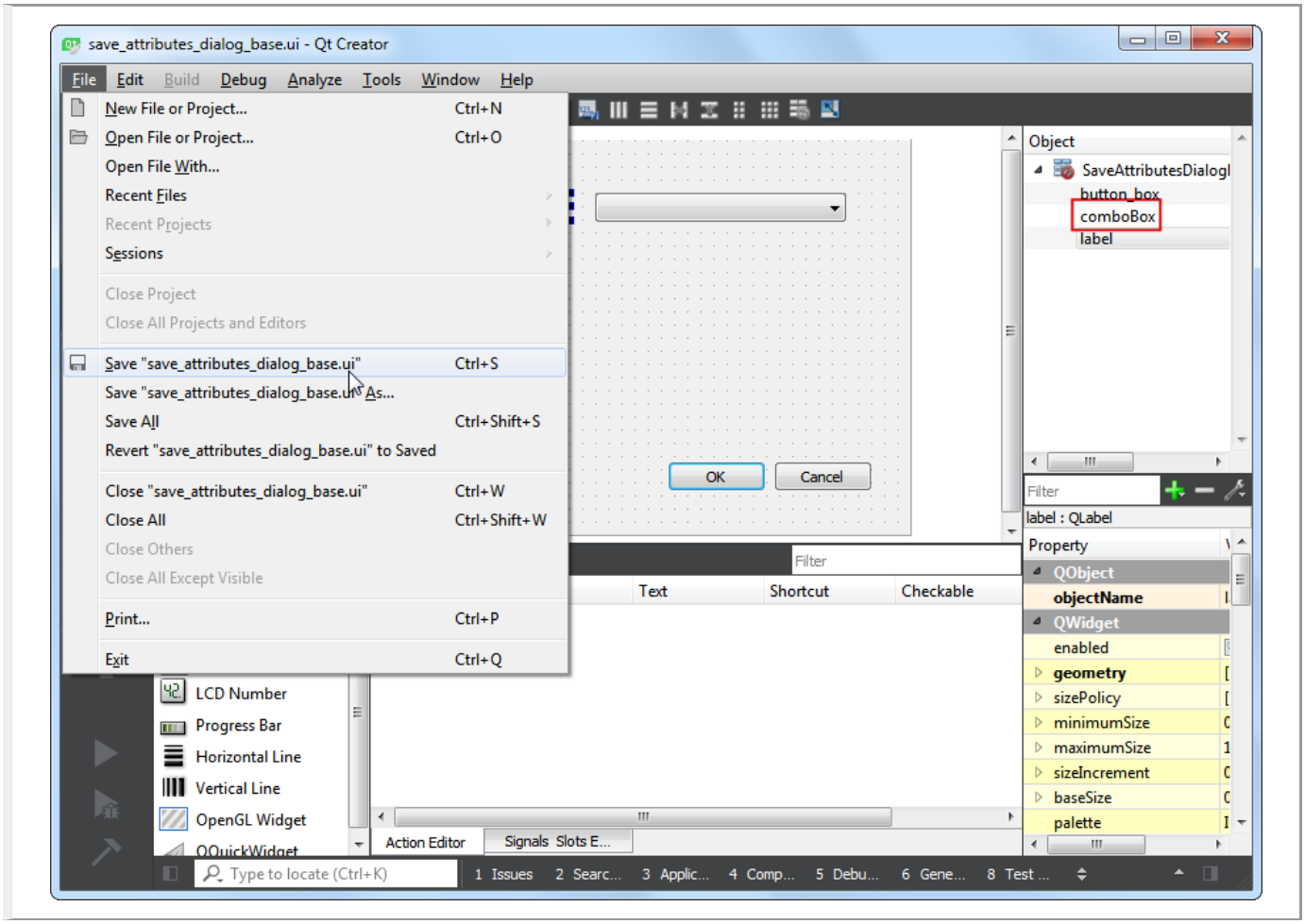
18. Resize the combo box and adjust its size. Now drag a Label type Display Widget on the dialog.



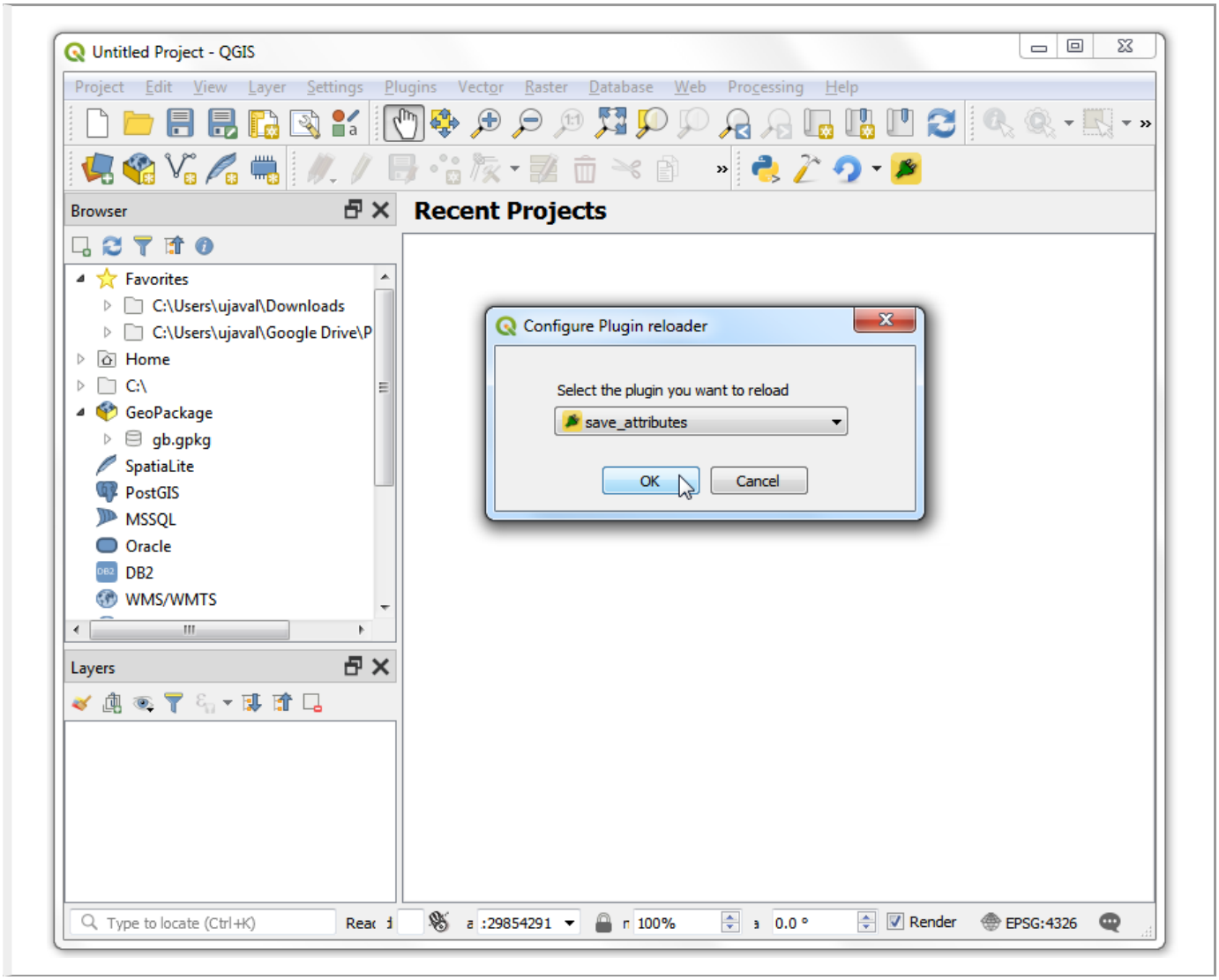
19. Click on the label text and enter `Select a layer`.



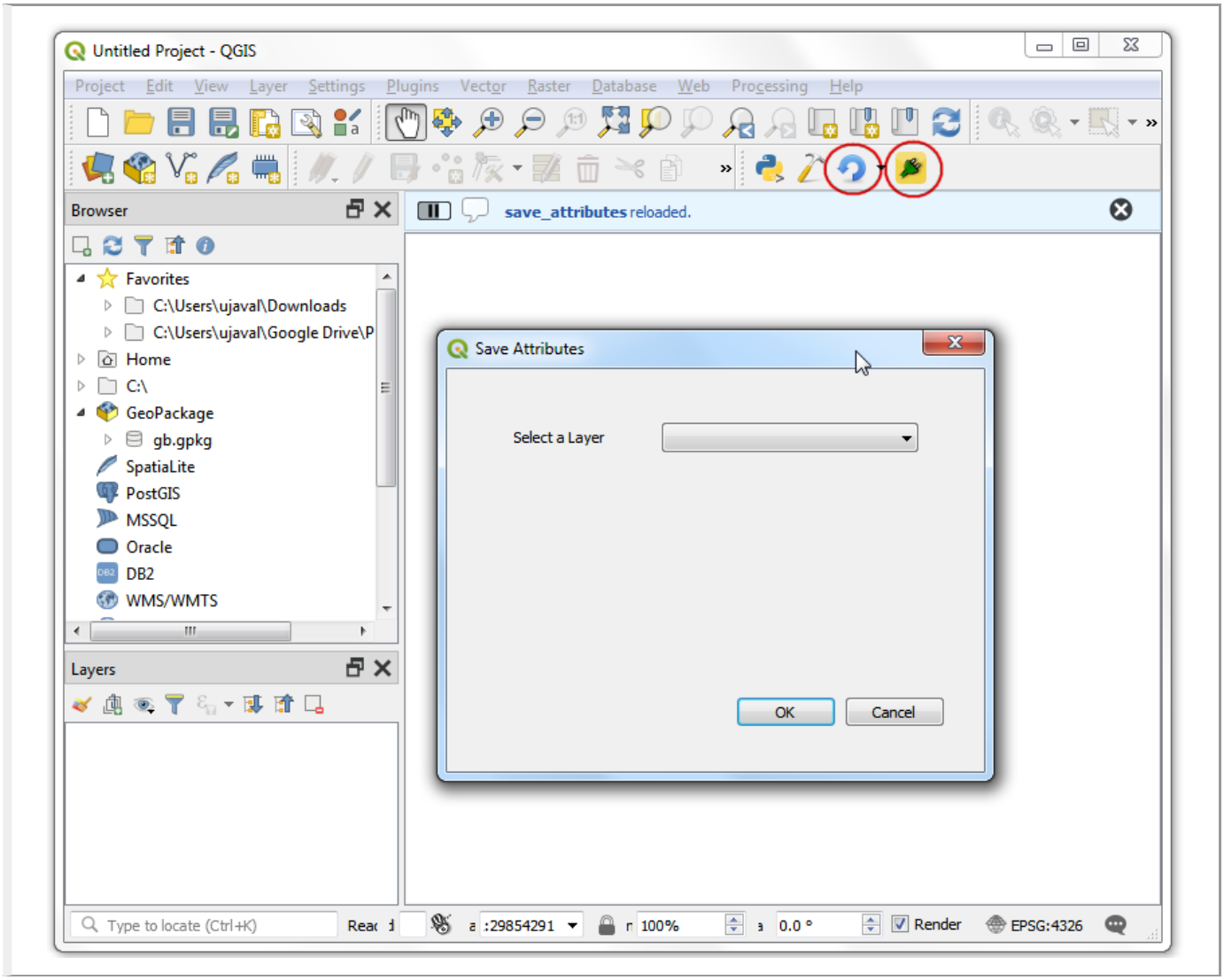
20. Save this file by going to File ‣ Save save_attributes_dialog_base.ui. Note the name of the combo box object is `comboBox`. To interact with this object using python code, we will have to refer to it by this name.



21. Let's reload our plugin so we can see the changes in the dialog window. Go to Plugin ‣ Plugin Reloader ‣ Choose a plugin to be reloaded. Select `SaveAttributes` in the Configure Plugin reloader dialog.



22. Click the Reload plugin button to load the latest version of the plugin. Click the Save Attributes as CSV button to open the newly designed dialog box.



23. Let's add some logic to the plugin that will populate the combo box with the layers loaded in QGIS. Go to the plugin directory and load the file `save_attributes.py` in a text editor. First, insert at the top of the file with the other imports:

```
from qgis.core import QgsProject
```

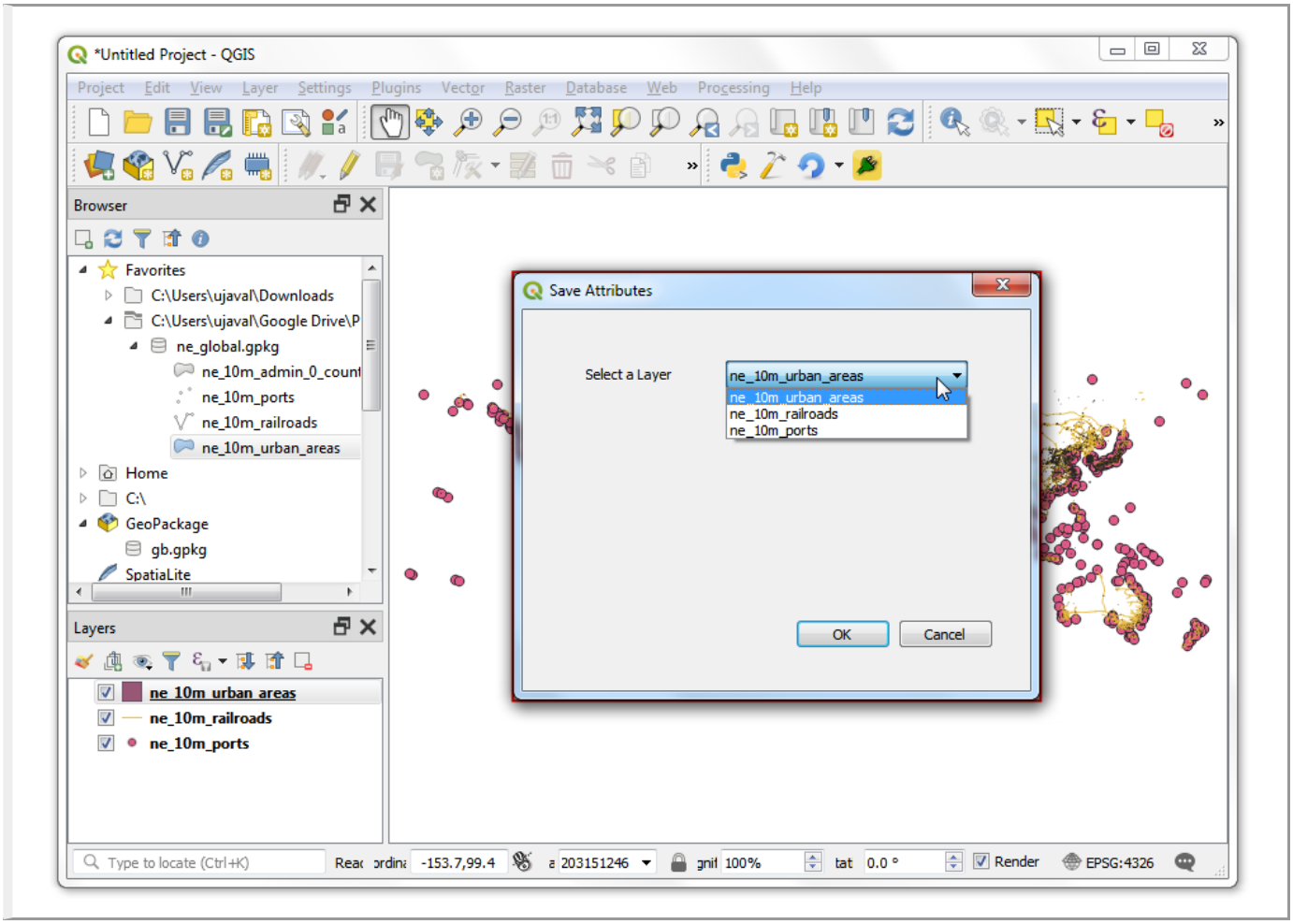
Then scroll down to the end and find the `run(self)` method. This method will be called when you click the toolbar button or select the plugin menu item. Add the following code at the beginning of the method. This code gets the layers loaded in QGIS and adds it to the `comboBox` object from the plugin dialog.

```
# Fetch the currently loaded layers
layers = QgsProject.instance().layerTreeRoot().children()
# Clear the contents of the comboBox from previous runs
self.dlg.comboBox.clear()
# Populate the comboBox with names of all the loaded layers
self.dlg.comboBox.addItems([layer.name() for layer in layers])
```

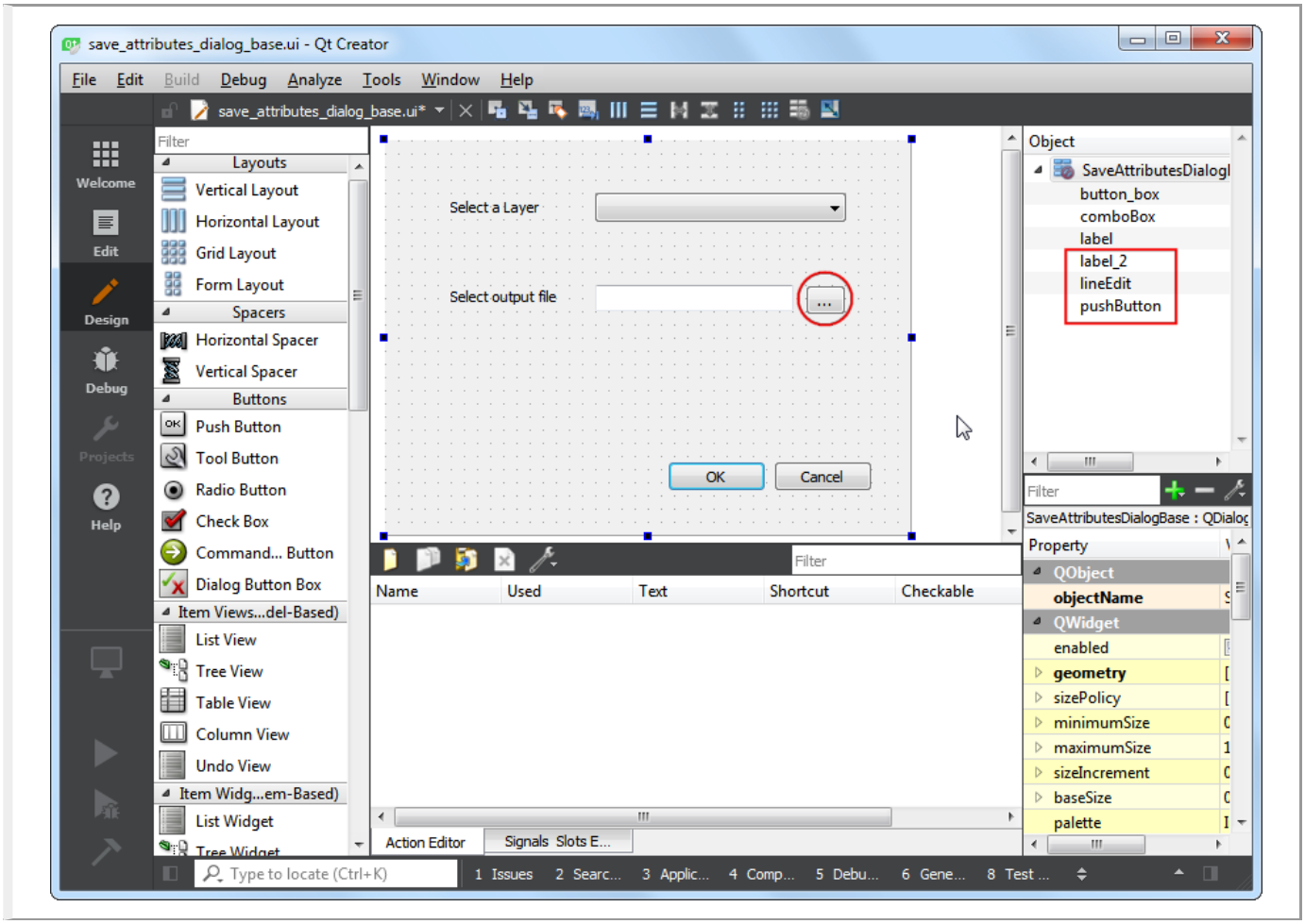
```
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""
from PyQt5.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QAction
from qgis.core import QgsProject

# Initialize Qt resources from file resources.py
from .resources import *
```

```
        self.iface.removeToolBarIcon(action)


    def run(self):
        """Run method that performs all the real work"""

        # Create the dialog with elements (after translation) and keep
        reference
        # Only create GUI ONCE in callback, so that it will only load when
        the plugin is started
        if self.first_start == True:
            self.first_start = False
            self.dlg = SaveAttributesDialog()

        # Fetch the currently loaded layers
        layers = QgsProject.instance().layerTreeRoot().children()
        # Clear the contents of the comboBox from previous runs
        self.dlg.comboBox.clear()
        # Populate the comboBox with names of all the loaded layers
        self.dlg.comboBox.addItems([layer.name() for layer in layers])

        # show the dialog
        self.dlg.show()
```

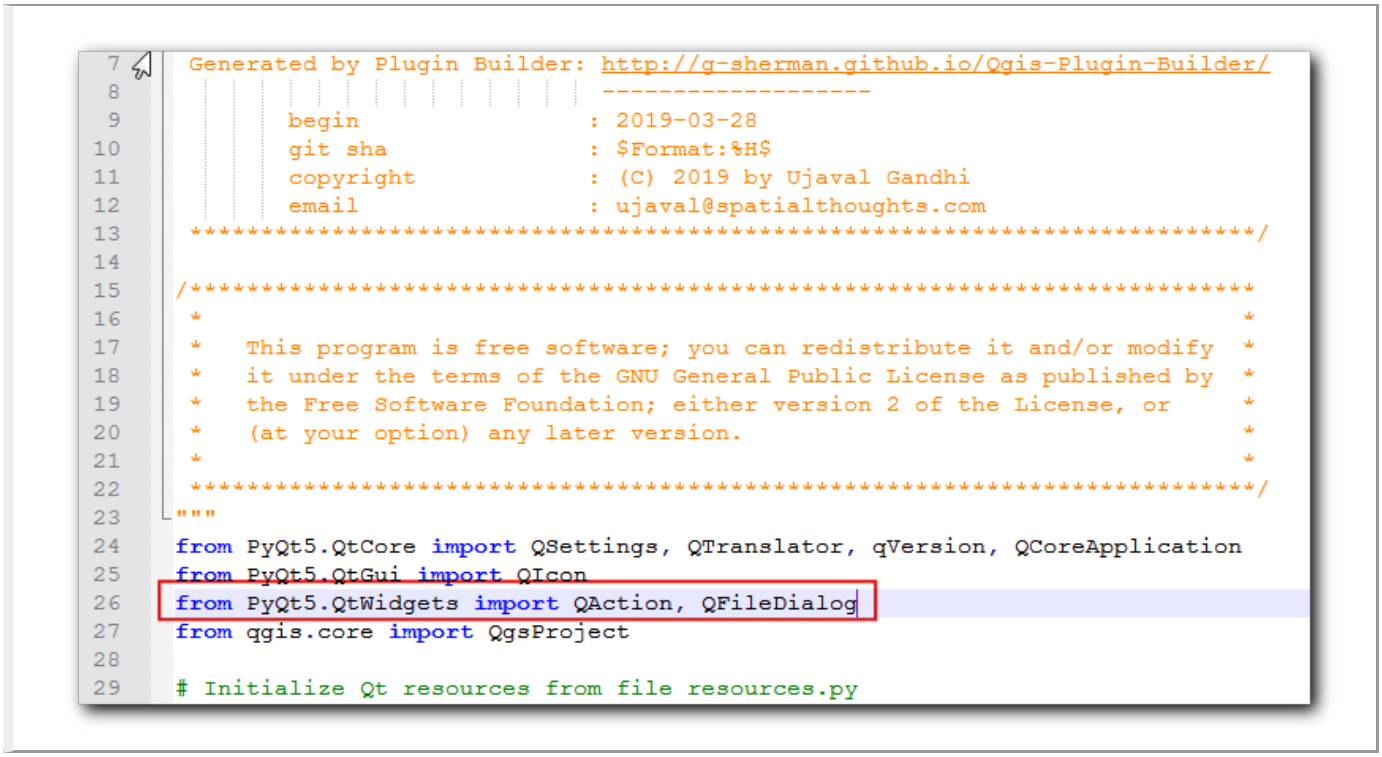
24. Back in the main QGIS window, reload the plugin by clicking on the Reload plugin button. To test this new functionality, we must load some layers in QGIS. After you have loaded some layers, launch the plugin by going to Vector ▸ Save Attributes ▸ Save Attributes as CSV. You will see that our combo box is now populated with the layer names that are loaded in QGIS.



25. Let's add the remaining user interface elements. Switch back to Qt Creator and load the `save_attributes_dialog_base.ui` file. Add a `Label` Display Widget and change the text to `Select output file`. Add a `LineEdit` type Input Widget that will show the output file path that the user has chosen. Next, add a `Push Button` type Button and change the button label to `...`. Note the object names of the widgets that we will have to use to interact with them. Save the file.



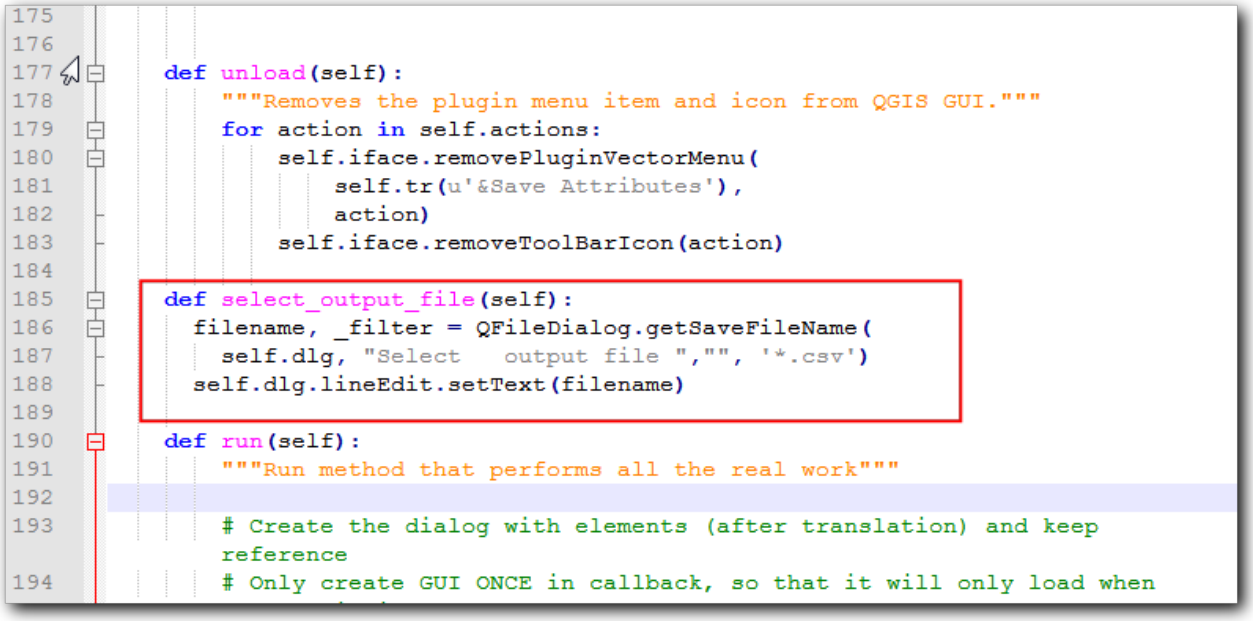
26. We will now add python code to open a file browser when the user clicks the `...` push button and show the select path in the line edit widget. Open the `save_attributes.py` file in a text editor. Add `QFileDialog` to `QtWidgets` list of imports at the top of the file.



```
 7  Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
 8                               -------------------
 9          begin                : 2019-03-28
10          git sha              : $Format:%H$
11          copyright            : (C) 2019 by Ujaval Gandhi
12          email                : ujaval@spatialthoughts.com
13  ***************************************************************************/
14
15 /***************************************************************************
16  *                                                                         *
17  *   This program is free software; you can redistribute it and/or modify  *
18  *   it under the terms of the GNU General Public License as published by  *
19  *   the Free Software Foundation; either version 2 of the License, or     *
20  *   (at your option) any later version.                                   *
21  *                                                                         *
22  ***************************************************************************/
23 """
24 from PyQt5.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
25 from PyQt5.QtGui import QIcon
26 from PyQt5.QtWidgets import QAction, QFileDialog
27 from qgis.core import QgsProject
28
29 # Initialize Qt resources from file resources.py
```

27. Add a new method called `select_output_file` with the following code. This code will open a file browser and populate the line edit widget with the path of the file that the user chose. Note, how `getSaveFileName` returns a tuple with the filename and the filter used.

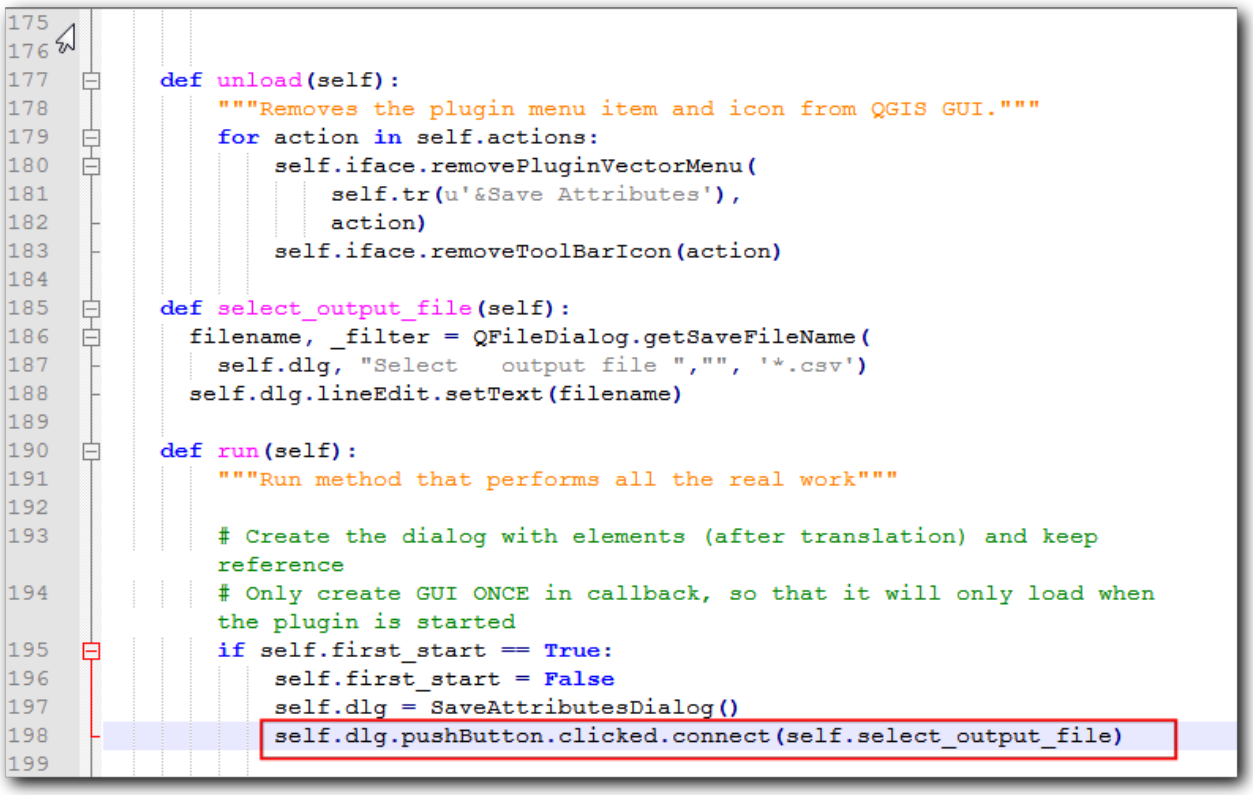
```
def select_output_file(self):
  filename, _filter = QFileDialog.getSaveFileName(
    self.dlg, "Select   output file ","", '*.csv')
  self.dlg.lineEdit.setText(filename)
```



```
    def unload(self):
        """Removes the plugin menu item and icon from QGIS GUI."""
        for action in self.actions:
            self.iface.removePluginVectorMenu(
                self.tr(u'&Save Attributes'),
                action)
            self.iface.removeToolBarIcon(action)

    def select_output_file(self):
      filename, _filter = QFileDialog.getSaveFileName(
        self.dlg, "Select   output file ","", '*.csv')
      self.dlg.lineEdit.setText(filename)

    def run(self):
        """Run method that performs all the real work"""

        # Create the dialog with elements (after translation) and keep
        reference
        # Only create GUI ONCE in callback, so that it will only load when
```

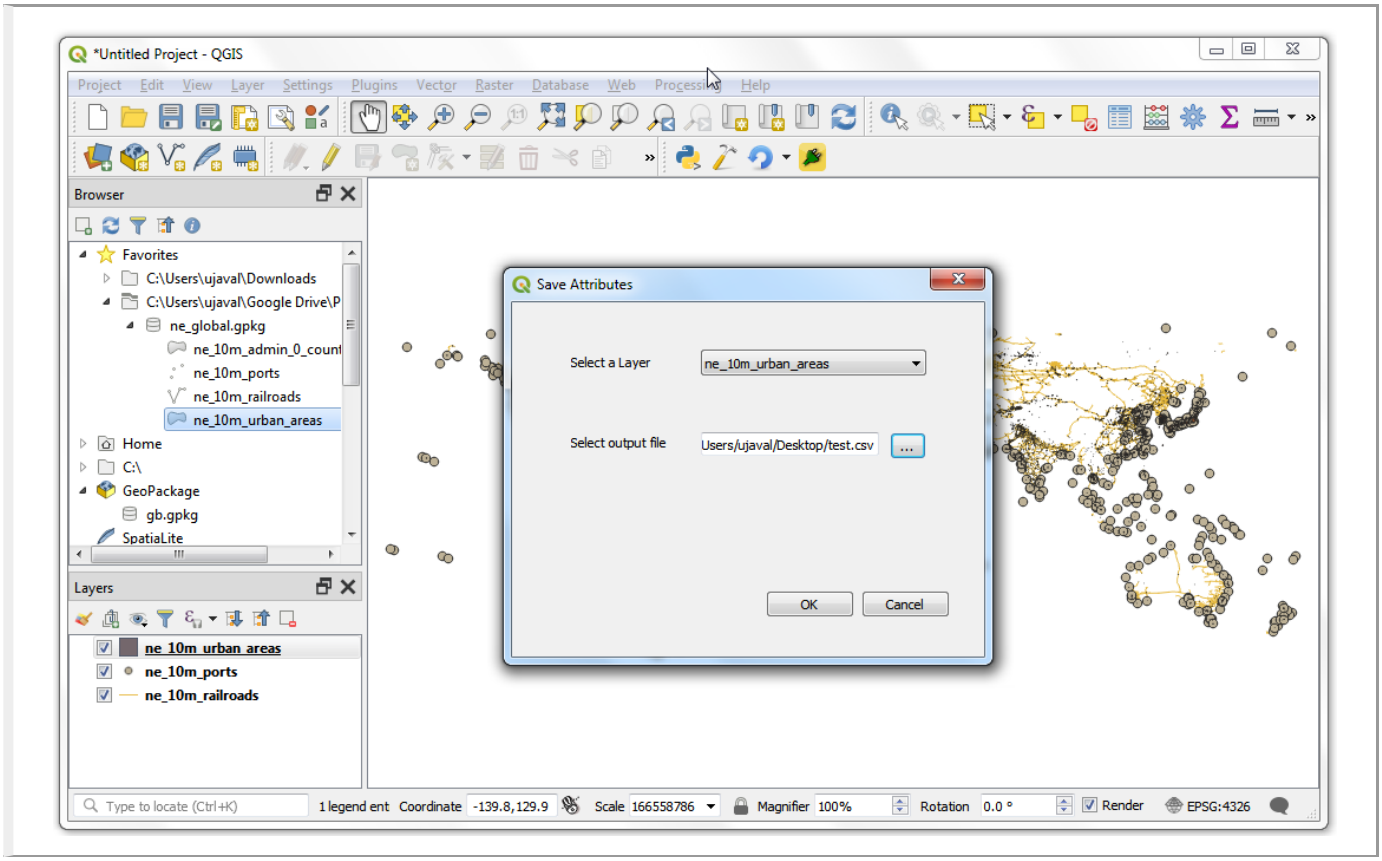
28. Now we need to add code so that when the ... button is clicked, `select_output_file` method is called. Scroll down to the `run` method and add the following line in the block where the dialog is initialized. This code will connect the `select_output_file` method to the `clicked` signal of the push button widget.

```
self.dlg.pushButton.clicked.connect(self.select_output_file)
```



```
    def unload(self):
        """Removes the plugin menu item and icon from QGIS GUI."""
        for action in self.actions:
            self.iface.removePluginVectorMenu(
                self.tr(u'&Save Attributes'),
                action)
            self.iface.removeToolBarIcon(action)

    def select_output_file(self):
      filename, _filter = QFileDialog.getSaveFileName(
        self.dlg, "Select   output file ","", '*.csv')
      self.dlg.lineEdit.setText(filename)

    def run(self):
        """Run method that performs all the real work"""

        # Create the dialog with elements (after translation) and keep
        reference
        # Only create GUI ONCE in callback, so that it will only load when
        the plugin is started
        if self.first_start == True:
            self.first_start = False
            self.dlg = SaveAttributesDialog()
            self.dlg.pushButton.clicked.connect(self.select_output_file)
```

29. Back in QGIS, reload the plugin and run it. If all went fine, you will be able to click the `...` button and select an output text file from your disk.



30. When you click OK on the plugin dialog, nothing happens. That is because we have not added the logic to pull attribute information from the layer and write it to the text file. We now have all the pieces in place to do just that. Find the place in the `run` method where it says `pass`. Replace it with the code below. The explanation for this code can be found in *Getting Started With Python Programming (QGIS3)* (getting_started_with_pyqgis.html).

```
filename = self.dlg.lineEdit.text()
with open(filename, 'w') as output_file:
  selectedLayerIndex = self.dlg.comboBox.currentIndex()
  selectedLayer = layers[selectedLayerIndex].layer()
  fieldnames = [field.name() for field in selectedLayer.fields()]
  # write header
  line = ','.join(name for name in fieldnames) + '\n'
  output_file.write(line)
  # wirte feature attributes
  for f in selectedLayer.getFeatures():
    line = ','.join(str(f[name]) for name in fieldnames) + '\n'
    output_file.write(line)
```

```
            self.dlg.pushButton.clicked.connect(self.select_output_file)


        # Fetch the currently loaded layers
        layers = QgsProject.instance().layerTreeRoot().children()
        # Clear the contents of the comboBox from previous runs
        self.dlg.comboBox.clear()
        # Populate the comboBox with names of all the loaded layers
        self.dlg.comboBox.addItems([layer.name() for layer in layers])

        # show the dialog
        self.dlg.show()
        # Run the dialog event loop
        result = self.dlg.exec_()
        # See if OK was pressed
        if result:
          filename = self.dlg.lineEdit.text()
          with open(filename, 'w') as output_file:
            selectedLayerIndex = self.dlg.comboBox.currentIndex()
            selectedLayer = layers[selectedLayerIndex].layer()
            fieldnames = [field.name() for field in selectedLayer.fields()]
            # write header
            line = ','.join(name for name in fieldnames) + '\n'
            output_file.write(line)
            # wirte feature attributes
            for f in selectedLayer.getFeatures():
              line = ','.join(str(f[name]) for name in fieldnames) + '\n'
              output_file.write(line)
```

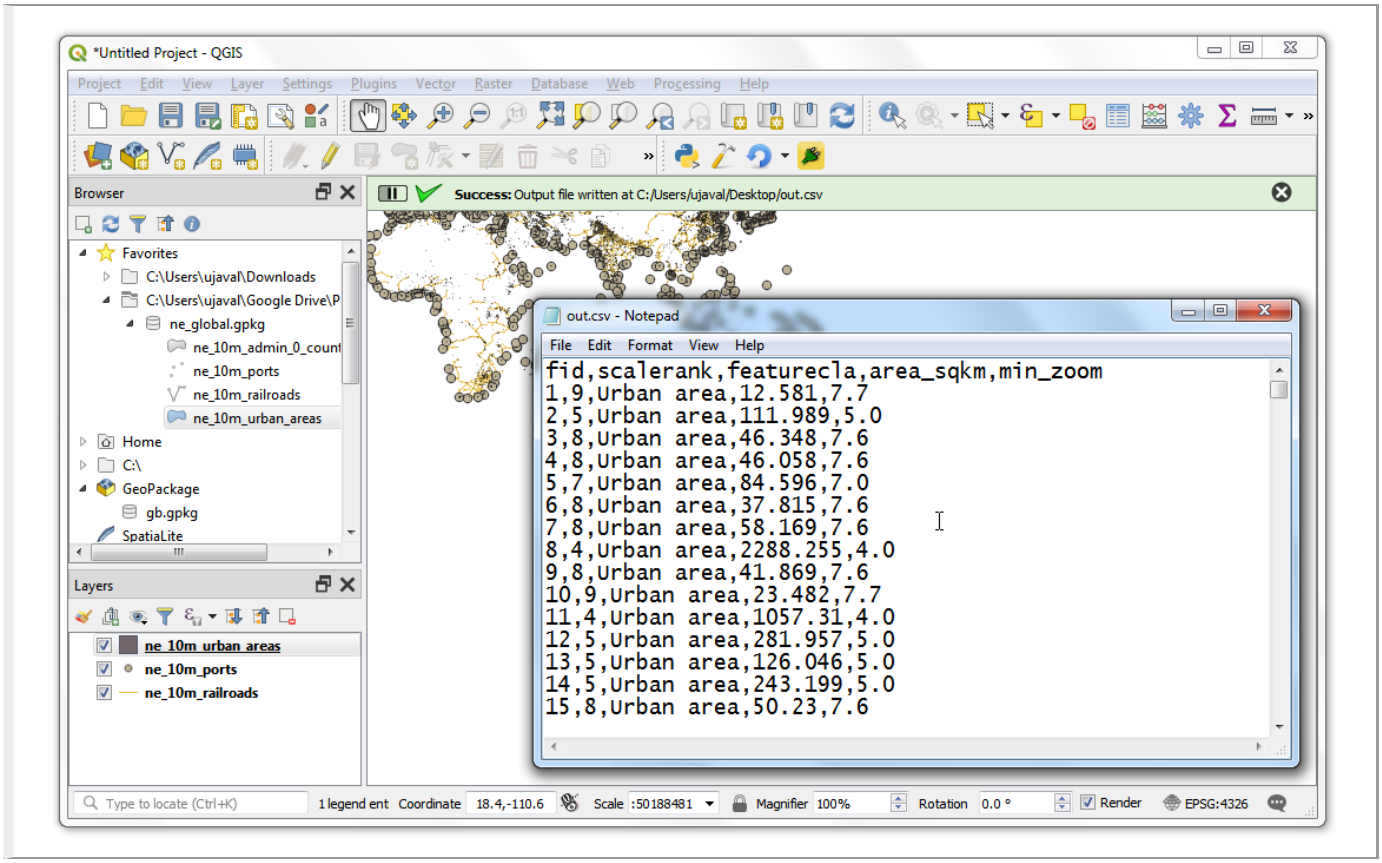
31. We have one last thing to add. When the operation finishes successfully, we should indicate the same to the user. The preferred way to give notifications to the user in QGIS is via the `self.iface.messageBar().pushMessage()` method. Add `Qgis` to `qgis.core` list of imports at the top of the file and add the code below at the end of the `run` method.

```
self.iface.messageBar().pushMessage(
  "Success", "Output file written at " + filename,
  level=Qgis.Success, duration=3)
```

```
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""
from PyQt5.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QAction, QFileDialog
from qgis.core import QgsProject, Qgis

# Initialize Qt resources from file resources.py
from .resources import *
```

```
        # See if OK was pressed
        if result:
          filename = self.dlg.lineEdit.text()
          with open(filename, 'w') as output_file:
            selectedLayerIndex = self.dlg.comboBox.currentIndex()
            selectedLayer = layers[selectedLayerIndex].layer()
            fieldnames = [field.name() for field in selectedLayer.fields()]
            # write header
            line = ','.join(name for name in fieldnames) + '\n'
            output_file.write(line)
            # wirte feature attributes
            for f in selectedLayer.getFeatures():
              line = ','.join(str(f[name]) for name in fieldnames) + '\n'
              output_file.write(line)
          self.iface.messageBar().pushMessage(
            "Success", "Output file written at " + filename,
            level=Qgis.Success, duration=3)

```

32. Now our plugin is ready. Reload the plugin and try it out. You will find that the output text file you chose will have the attributes from the vector layer.



33. You can zip the plugin directory and share it with your users. They can unzip the contents to their plugin directory and try out your plugin. If this was a real plugin, you would upload it to the QGIS Plugin Repository (https://plugins.qgis.org/) so that all QGIS users will be able to find and download your plugin.

> **Note**
>
> This plugin is for demonstration purpose only. Do not publish this plugin or upload it to the QGIS plugin repository.

Below is the full `save_attributes.py` file as a reference.

```
# -*- coding: utf-8 -*-
"""
/***************************************************************************
 SaveAttributes
                                 A QGIS plugin
 This plugin saves the attributes of the selected vector layer as a CSV file.
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                              -------------------
        begin                : 2019-03-28
        git sha              : $Format:%H$
        copyright            : (C) 2019 by Ujaval Gandhi
        email                : ujaval@spatialthoughts.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""
from PyQt5.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QAction, QFileDialog
from qgis.core import QgsProject, Qgis

# Initialize Qt resources from file resources.py
from .resources import *
# Import the code for the dialog
from .save_attributes_dialog import SaveAttributesDialog
import os.path


class SaveAttributes:
    """QGIS Plugin Implementation."""

    def __init__(self, iface):
        """Constructor.

        :param iface: An interface instance that will be passed to this class
            which provides the hook by which you can manipulate the QGIS
            application at run time.
        :type iface: QgsInterface
        """
        # Save reference to the QGIS interface
        self.iface = iface
        # initialize plugin directory
        self.plugin_dir = os.path.dirname(__file__)
        # initialize locale
        locale = QSettings().value('locale/userLocale')[0:2]
        locale_path = os.path.join(
            self.plugin_dir,
            'i18n',
            'SaveAttributes_{}.qm'.format(locale))
```

```
        if os.path.exists(locale_path):
            self.translator = QTranslator()
            self.translator.load(locale_path)

            if qVersion() > '4.3.3':
                QCoreApplication.installTranslator(self.translator)

        # Declare instance attributes
        self.actions = []
        self.menu = self.tr(u'&Save Attributes')

        # Check if plugin was started the first time in current QGIS session
        # Must be set in initGui() to survive plugin reloads
        self.first_start = None

    # noinspection PyMethodMayBeStatic
    def tr(self, message):
        """Get the translation for a string using Qt translation API.

        We implement this ourselves since we do not inherit QObject.

        :param message: String for translation.
        :type message: str, QString

        :returns: Translated version of message.
        :rtype: QString
        """
        # noinspection PyTypeChecker,PyArgumentList,PyCallByClass
        return QCoreApplication.translate('SaveAttributes', message)


    def add_action(
        self,
        icon_path,
        text,
        callback,
        enabled_flag=True,
        add_to_menu=True,
        add_to_toolbar=True,
        status_tip=None,
        whats_this=None,
        parent=None):
        """Add a toolbar icon to the toolbar.

        :param icon_path: Path to the icon for this action. Can be a resource
            path (e.g. ':/plugins/foo/bar.png') or a normal file system path.
        :type icon_path: str

        :param text: Text that should be shown in menu items for this action.
        :type text: str

        :param callback: Function to be called when the action is triggered.
        :type callback: function

        :param enabled_flag: A flag indicating if the action should be enabled
            by default. Defaults to True.
        :type enabled_flag: bool
```

```
        :param add_to_menu: Flag indicating whether the action should also
            be added to the menu. Defaults to True.
        :type add_to_menu: bool

        :param add_to_toolbar: Flag indicating whether the action should also
            be added to the toolbar. Defaults to True.
        :type add_to_toolbar: bool

        :param status_tip: Optional text to show in a popup when mouse pointer
            hovers over the action.
        :type status_tip: str

        :param parent: Parent widget for the new action. Defaults None.
        :type parent: QWidget

        :param whats_this: Optional text to show in the status bar when the
            mouse pointer hovers over the action.

        :returns: The action that was created. Note that the action is also
            added to self.actions list.
        :rtype: QAction
        """

        icon = QIcon(icon_path)
        action = QAction(icon, text, parent)
        action.triggered.connect(callback)
        action.setEnabled(enabled_flag)

        if status_tip is not None:
            action.setStatusTip(status_tip)

        if whats_this is not None:
            action.setWhatsThis(whats_this)

        if add_to_toolbar:
            # Adds plugin icon to Plugins toolbar
            self.iface.addToolBarIcon(action)

        if add_to_menu:
            self.iface.addPluginToVectorMenu(
                self.menu,
                action)

        self.actions.append(action)

        return action

    def initGui(self):
        """Create the menu entries and toolbar icons inside the QGIS GUI."""

        icon_path = ':/plugins/save_attributes/icon.png'
        self.add_action(
            icon_path,
            text=self.tr(u'Save Attributes as CSV'),
            callback=self.run,
            parent=self.iface.mainWindow())

        # will be set False in run()
```

```
        self.first_start = True


    def unload(self):
        """Removes the plugin menu item and icon from QGIS GUI."""
        for action in self.actions:
            self.iface.removePluginVectorMenu(
                self.tr(u'&Save Attributes'),
                action)
            self.iface.removeToolBarIcon(action)

    def select_output_file(self):
      filename, _filter = QFileDialog.getSaveFileName(
        self.dlg, "Select   output file ","", '*.csv')
      self.dlg.lineEdit.setText(filename)

    def run(self):
        """Run method that performs all the real work"""

        # Create the dialog with elements (after translation) and keep reference
        # Only create GUI ONCE in callback, so that it will only load when the plugin is starte
        if self.first_start == True:
            self.first_start = False
            self.dlg = SaveAttributesDialog()
            self.dlg.pushButton.clicked.connect(self.select_output_file)


        # Fetch the currently loaded layers
        layers = QgsProject.instance().layerTreeRoot().children()
        # Clear the contents of the comboBox and lineEdit from previous runs
        self.dlg.comboBox.clear()
        self.dlg.lineEdit.clear()

        # Populate the comboBox with names of all the loaded layers
        self.dlg.comboBox.addItems([layer.name() for layer in layers])

        # show the dialog
        self.dlg.show()
        # Run the dialog event loop
        result = self.dlg.exec_()
        # See if OK was pressed
        if result:
          filename = self.dlg.lineEdit.text()
          with open(filename, 'w') as output_file:
            selectedLayerIndex = self.dlg.comboBox.currentIndex()
            selectedLayer = layers[selectedLayerIndex].layer()
            fieldnames = [field.name() for field in selectedLayer.fields()]
            # write header
            line = ','.join(name for name in fieldnames) + '\n'
            output_file.write(line)
            # wirte feature attributes
            for f in selectedLayer.getFeatures():
              line = ','.join(str(f[name]) for name in fieldnames) + '\n'
              output_file.write(line)
          self.iface.messageBar().pushMessage(
            "Success", "Output file written at " + filename,
            level=Qgis.Success, duration=3)
```

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Building a Processing Plugin (QGIS3)

In the previous tutorial *Building a Python Plugin (QGIS3)* (building_a_python_plugin.html), you learnt how to create a python plugin - including the user interface and custom logic for processing the data. While these type of plugins are useful, it puts the burden of designing the user interface on the plugin author. This results in each plugin having different ways to interact with it - which is confusing to the users. Also, regular plugins do not interact with another parts of QGIS. For example, you cannot use the plugin functionality from another algorithm. If the plugin that you want to write is primarily for analysis, and the user interaction that you want is limited to letting the user pick inputs and outputs, there is a much easier and preferred way to write plugins using the Processing Framework. It removes the need for you to design the user interface - simplifying the process. The built-in processing library creates a standard processing interface depending on your inputs that looks and behaves just like any other processing algorithm in QGIS. It also seamlessly integrates with rest of the Processing framework - so your plugin algorithms can be used in batch processing, graphical modeler, called from python console etc.

Overview of the Task

We will re-implement a simple plugin from the tutorial *Building a Python Plugin (QGIS3)* (building_a_python_plugin.html) as a processing plugin. It will result in a new processing provider called `Save Attributes` and an algorithm `Save Attributes as CSV` that will allow users to pick a vector layer and write its attributes to a CSV file.

Get the Tools

A Text Editor or a Python IDE

Any kind of software development requires a good text editor. If you already have a favorite text editor or an IDE (Integrated Development Environment), you may use it for this tutorial. Otherwise, each platform offers a wide variety of free or paid options for text editors. Choose the one that fits your needs.

This tutorial uses Notepad++ editor on Windows.

Windows

Notepad++ (http://notepad-plus-plus.org/) is a good free editor for windows. Download and install the Notepad++ editor (https://notepad-plus-plus.org/download/).

> **Note**
>
> If you are using Notepad++, makes sure to go to Settings ‣ Preferences ‣ Tab Settings and enable Replace by space. Python is very sensitive about whitespace and this setting will ensure tabs and spaces are treated properly.

Plugin Builder plugin

There is a helpful QGIS plugin named `Plugin Builder` which creates all the necessary files and the boilerplate code for a plugin. Find and install the `Plugin Builder` plugin. See *Using Plugins* (../using_plugins.html) for more details on how to install plugins.

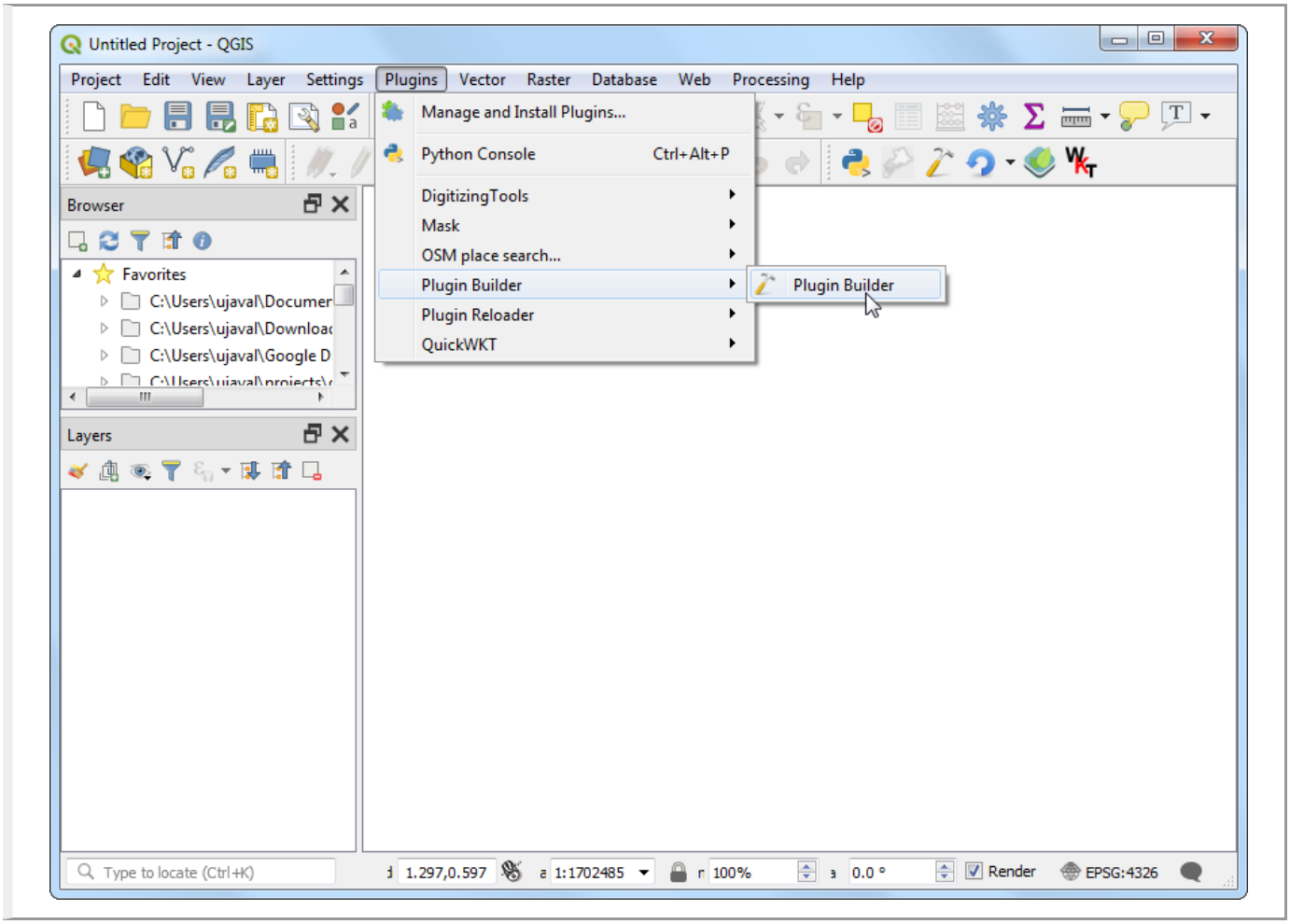
Plugins Reloader plugin

This is another helper plugin which allows iterative development of plugins. Using this plugin, you can change your plugin code and have it reflected in QGIS without having to restart QGIS every time. Find and install the `Plugin Reloader` plugin. See *Using Plugins* (../using_plugins.html) for more details on how to install plugins.

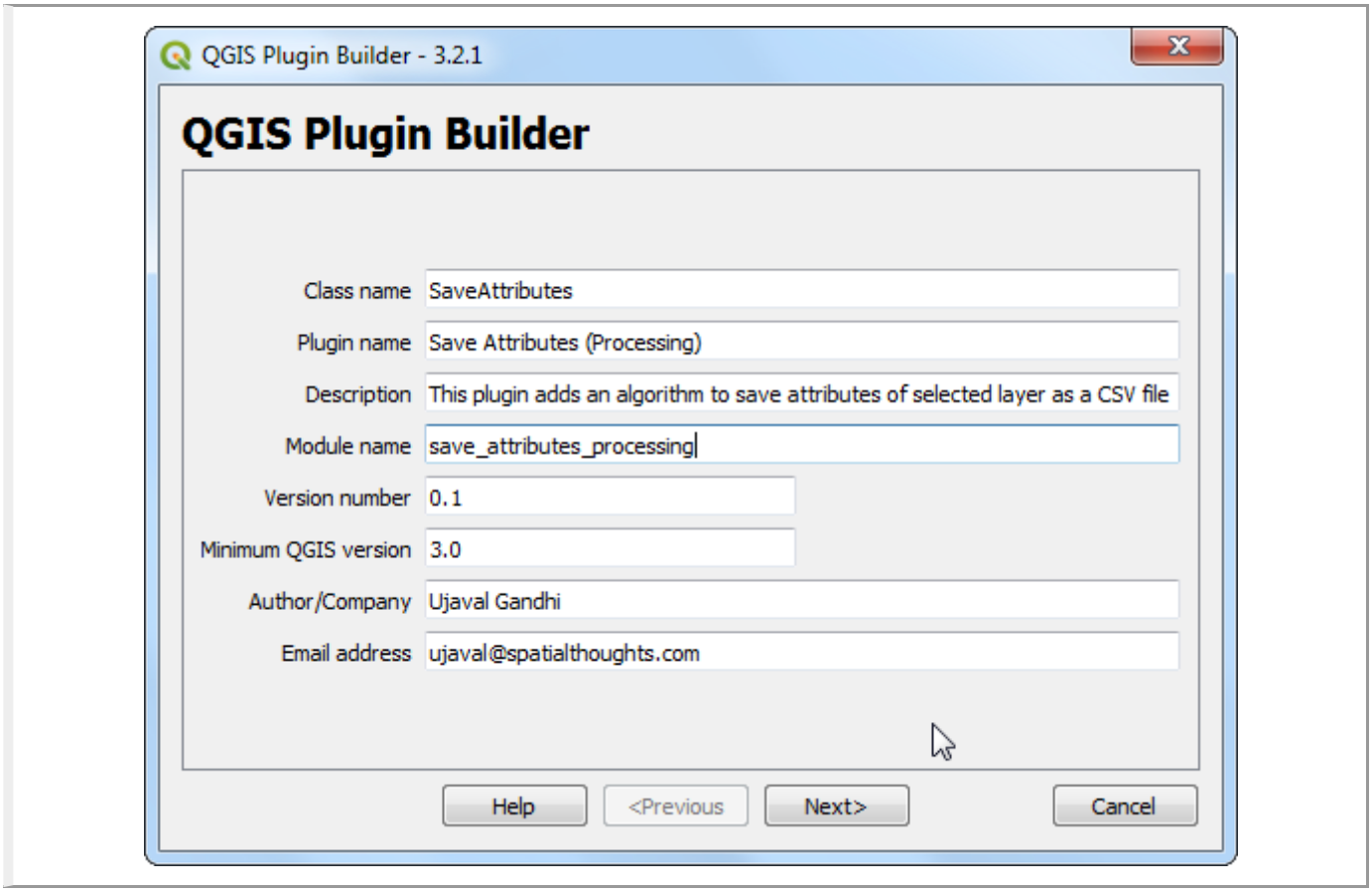
> **Note**
>
> Plugin Reloader is an experimental plugin. Make sure you have checked Show also experimental plugins in Plugin Manager settings if you cannot find it.

Procedure

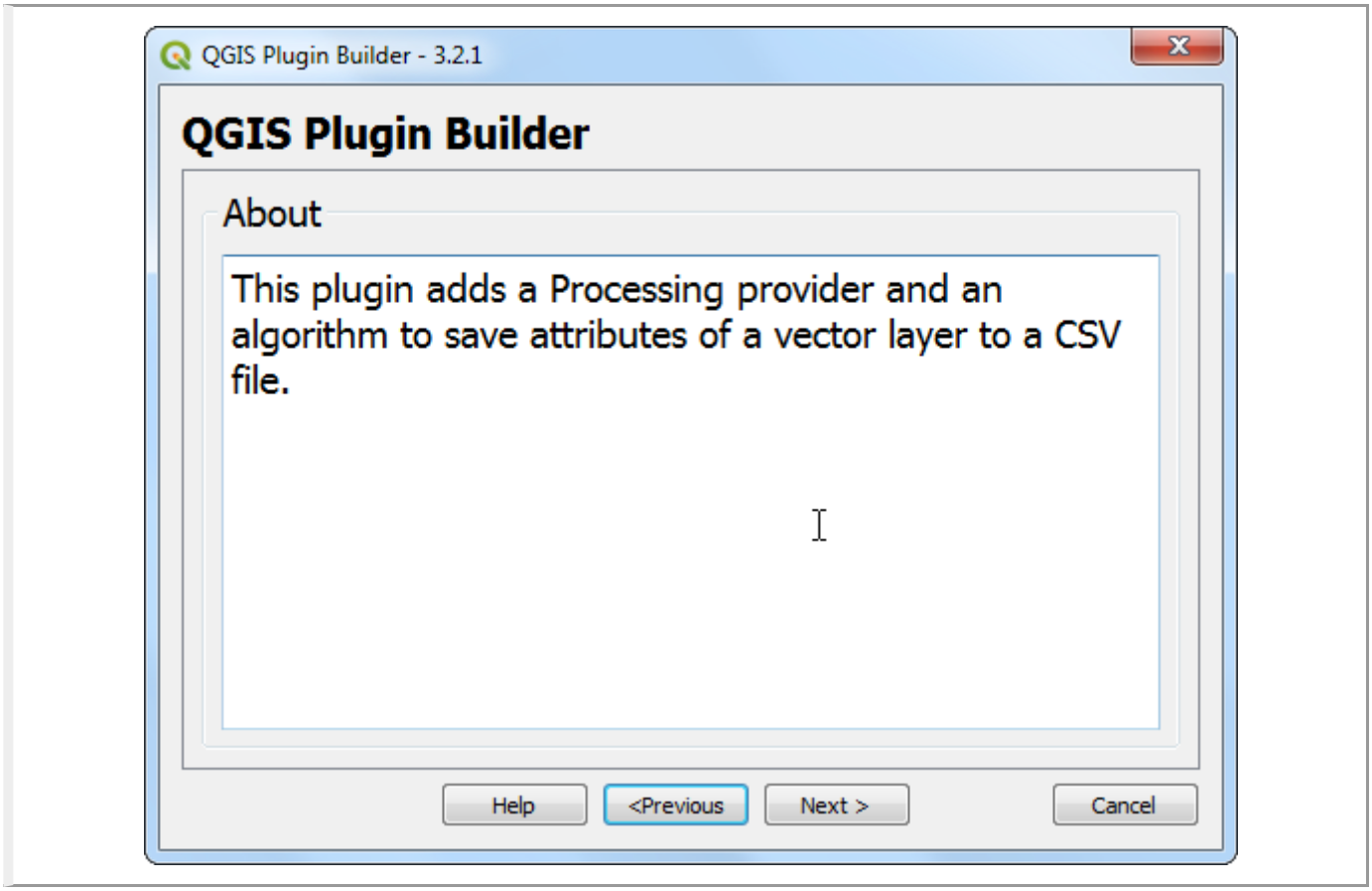
1. Open QGIS. Go to Plugins ‣ Plugin Builder ‣ Plugin Builder.



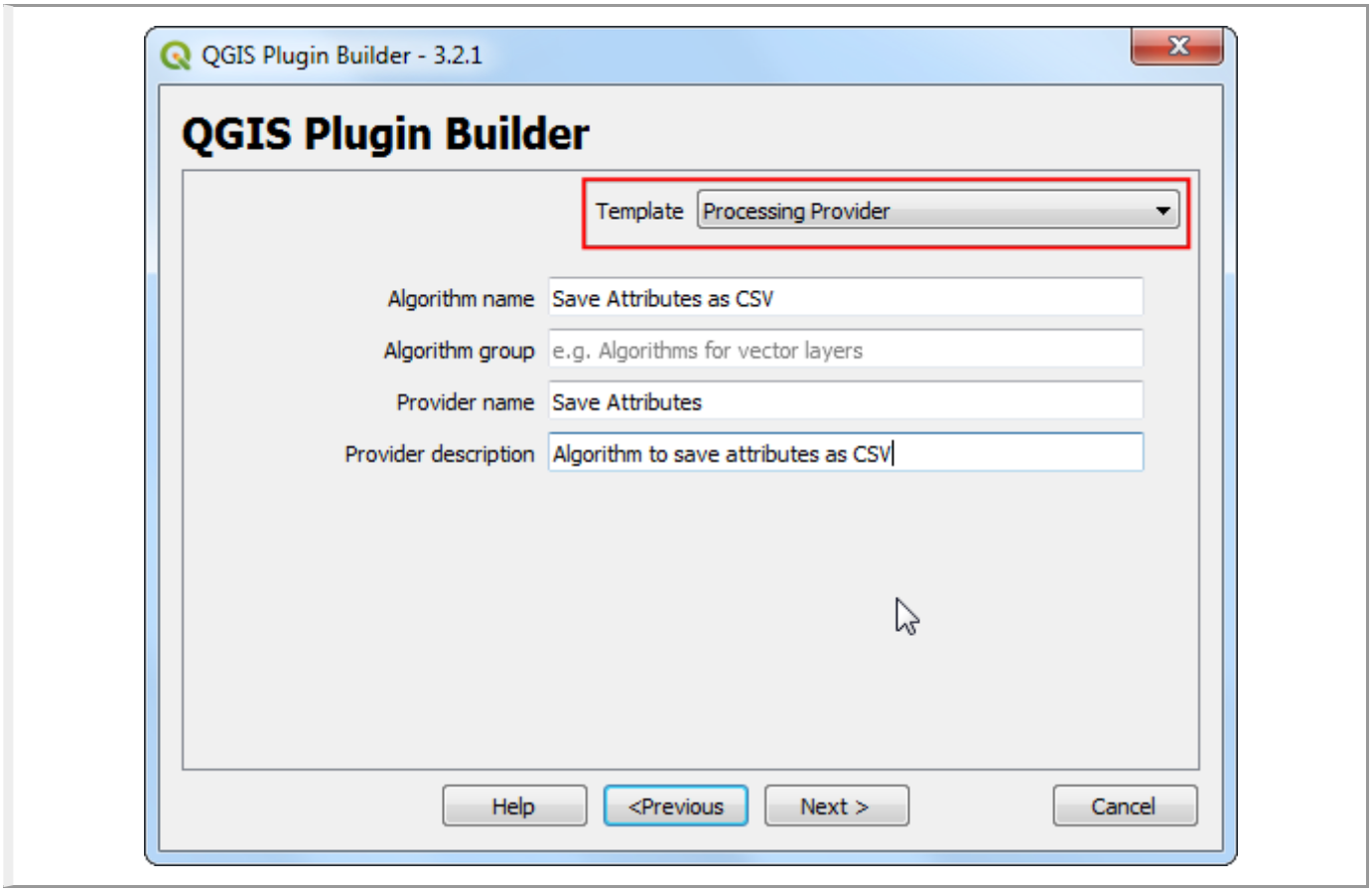
2. You will see the QGIS Plugin Builder dialog with a form. You can fill the form with details relating to our plugin. The Class name will be the name of the Python Class containing the logic of the plugin. This will also be the name of the folder containing all the plugin files. Enter `SaveAttributes` as the class name. The Plugin name is the name under which your plugin will appear in the Plugin Manager. Enter the name as `Save Attributes (Processing)`. Add a description in the Description field. The Module name will be the name of the main python file for the plugin. Enter it as `save_attributes_processing`. Leave the version numbers as they are and enter your name and email address in the appropriate fields. Click Next.



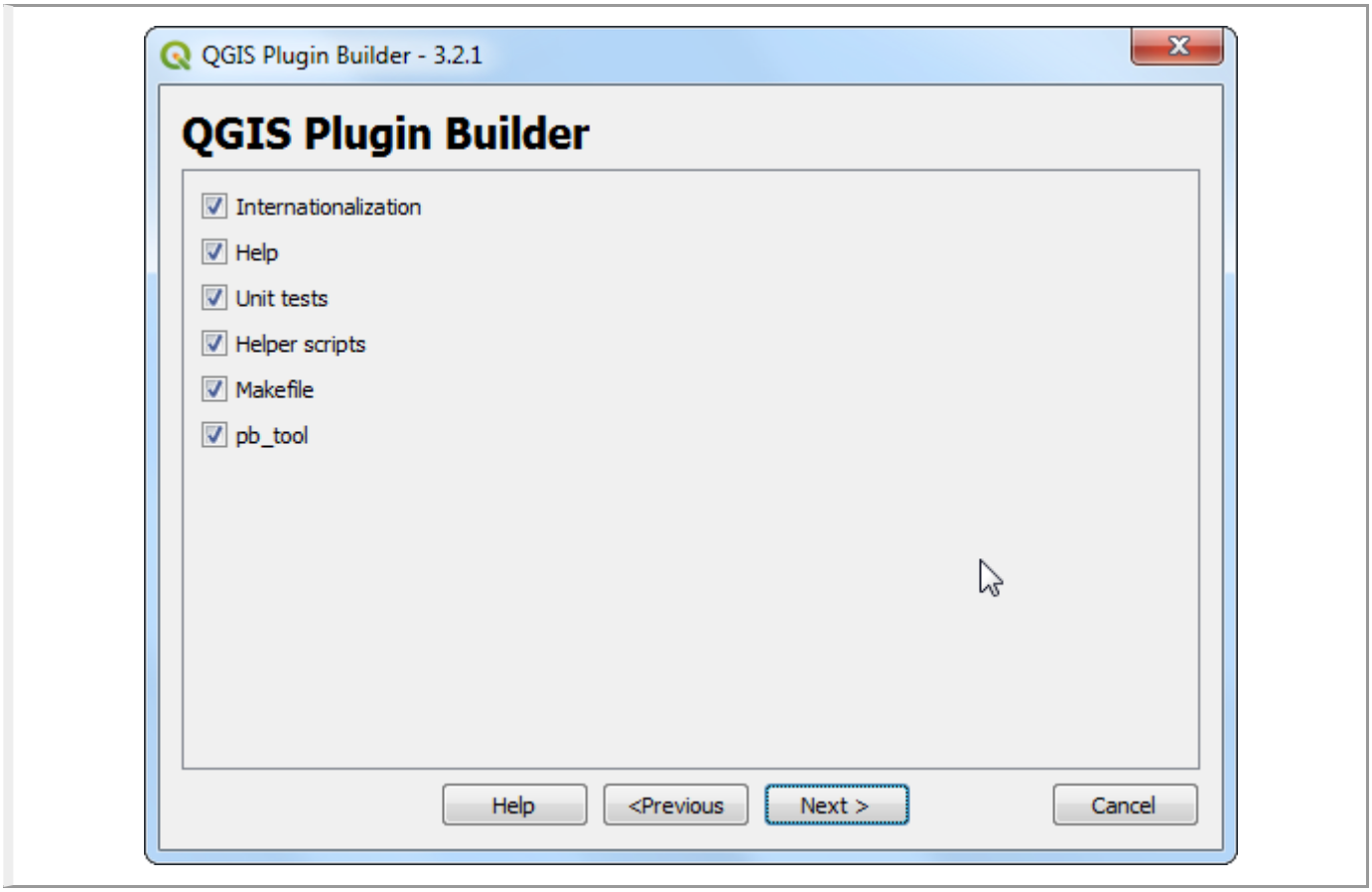
3. Enter a brief description of the plugin for the About dialog and click Next.



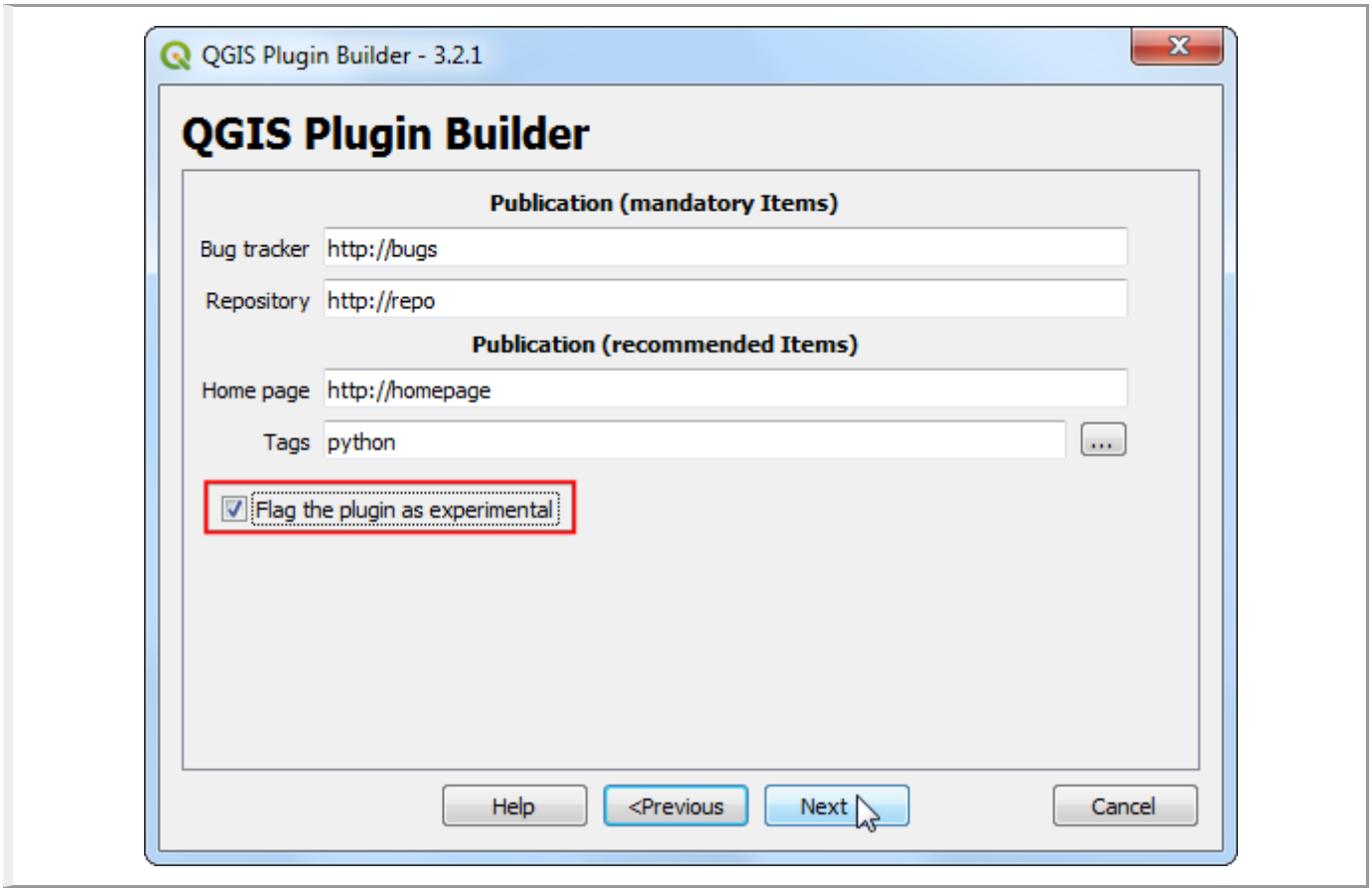
4. Select the `Processing Provider` from the Template selector. The Algorithm name value will be how the users will find the processing algorithm in the Processing Toolox. Enter it as `Save Attributes as CSV` . Leave the Algorithm group blank. Enter the Provider name as `Save Attributes` . Enter a description in the Provider description field. Click Next.



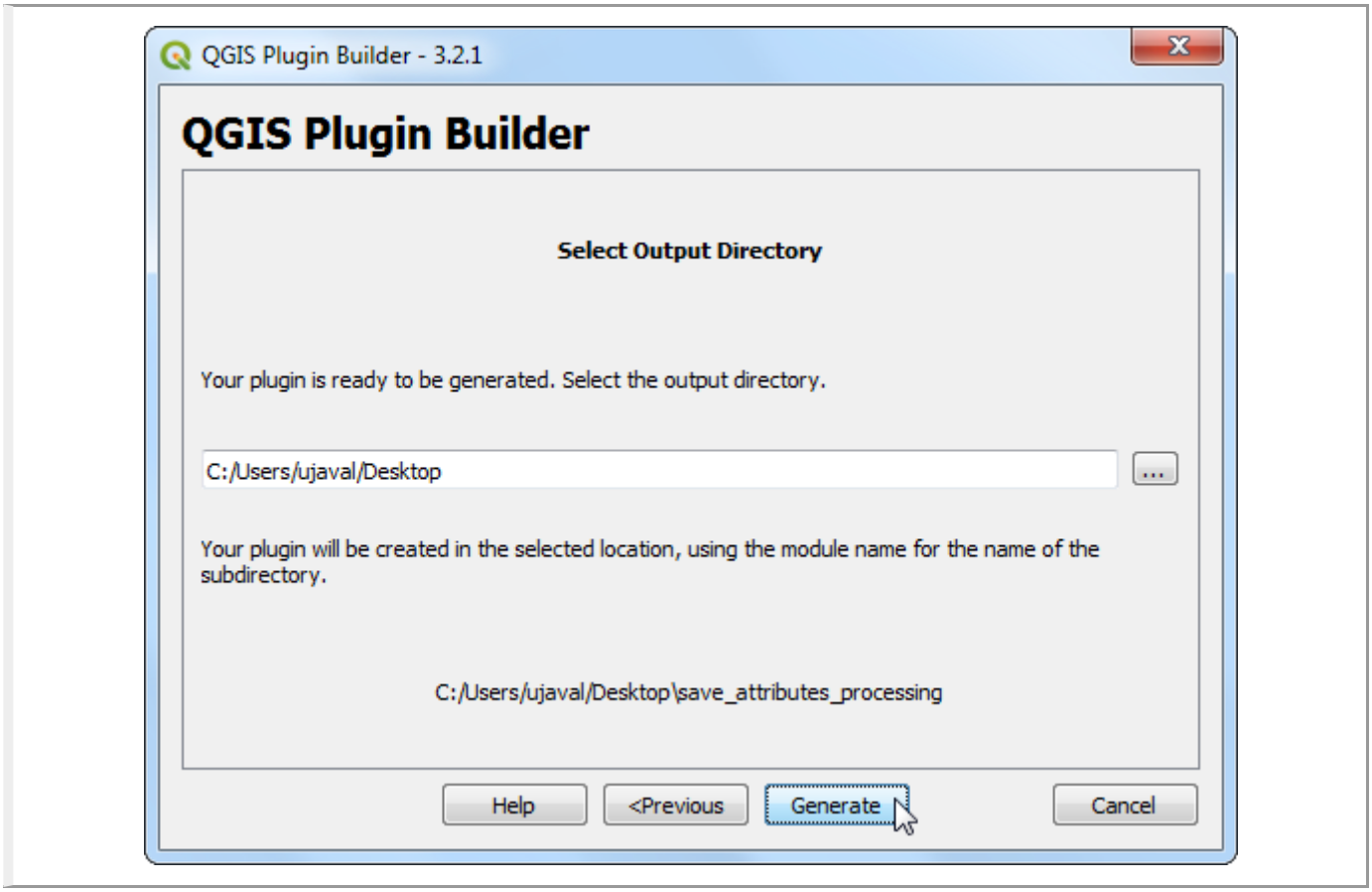
5. Plugin builder will prompt you for the type of files to generate. Keep the default selection and click Next.



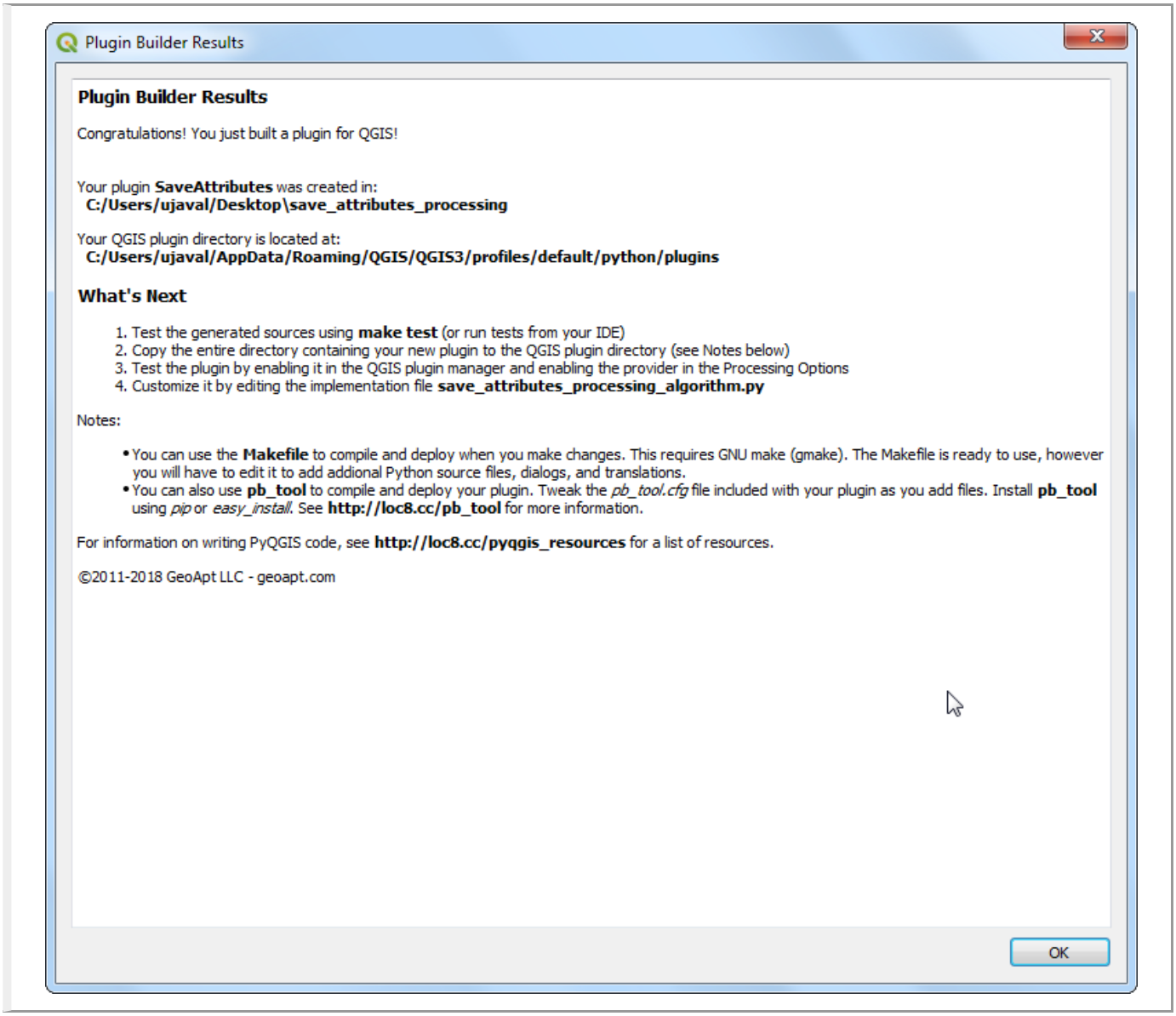
6. As we do not intend to publish the plugin, you may leave the Bug tracker, Repository and Home page values to default. Check the Flag the plugin as experimental box at the bottom and click Next.



7. You will be prompted to choose a directory for your plugin. For now, save it to a directory you can locate easily on your computer and click Generate.



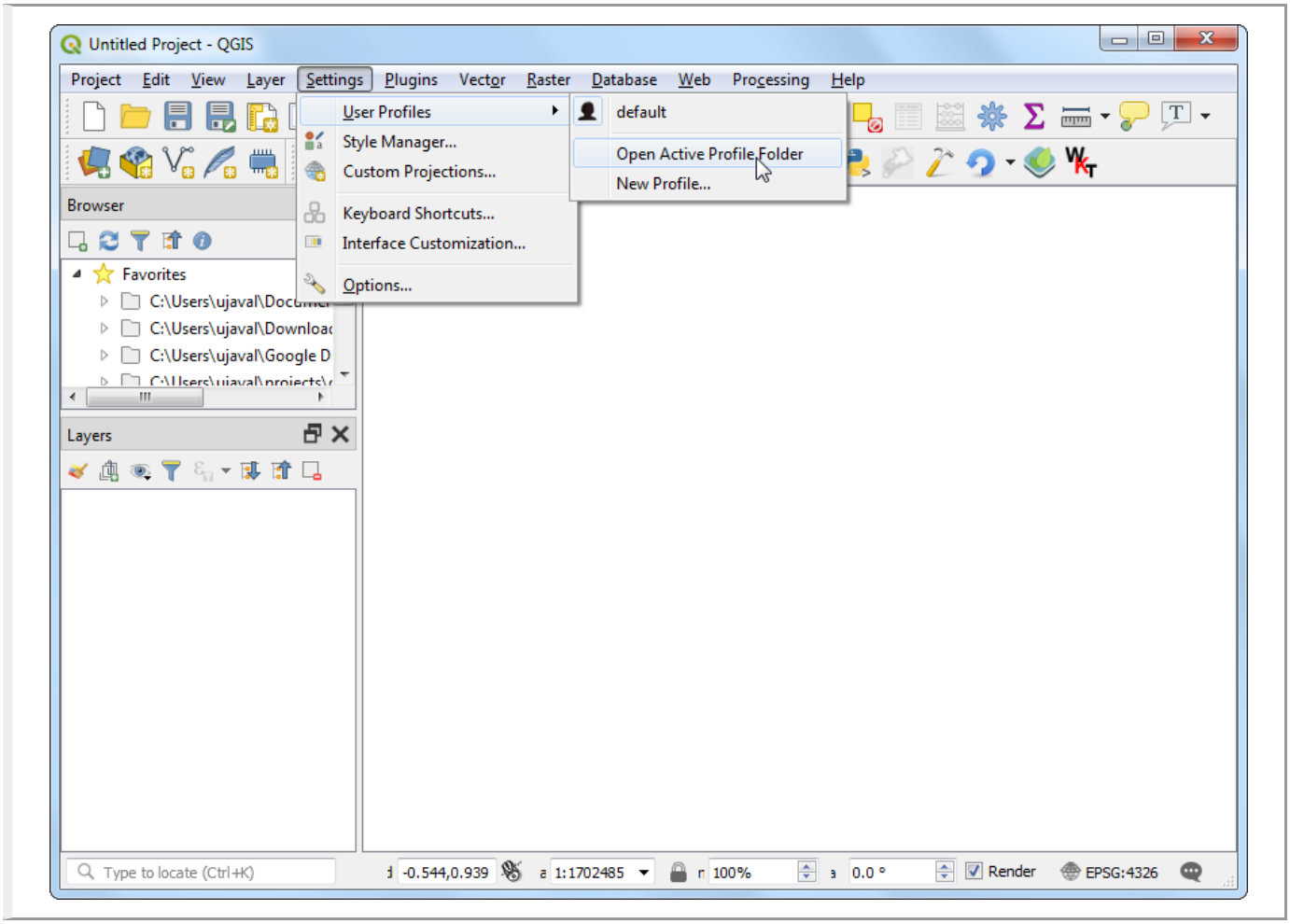
8. Next, press the generate button. You will see a confirmation dialog once your plugin template is created.



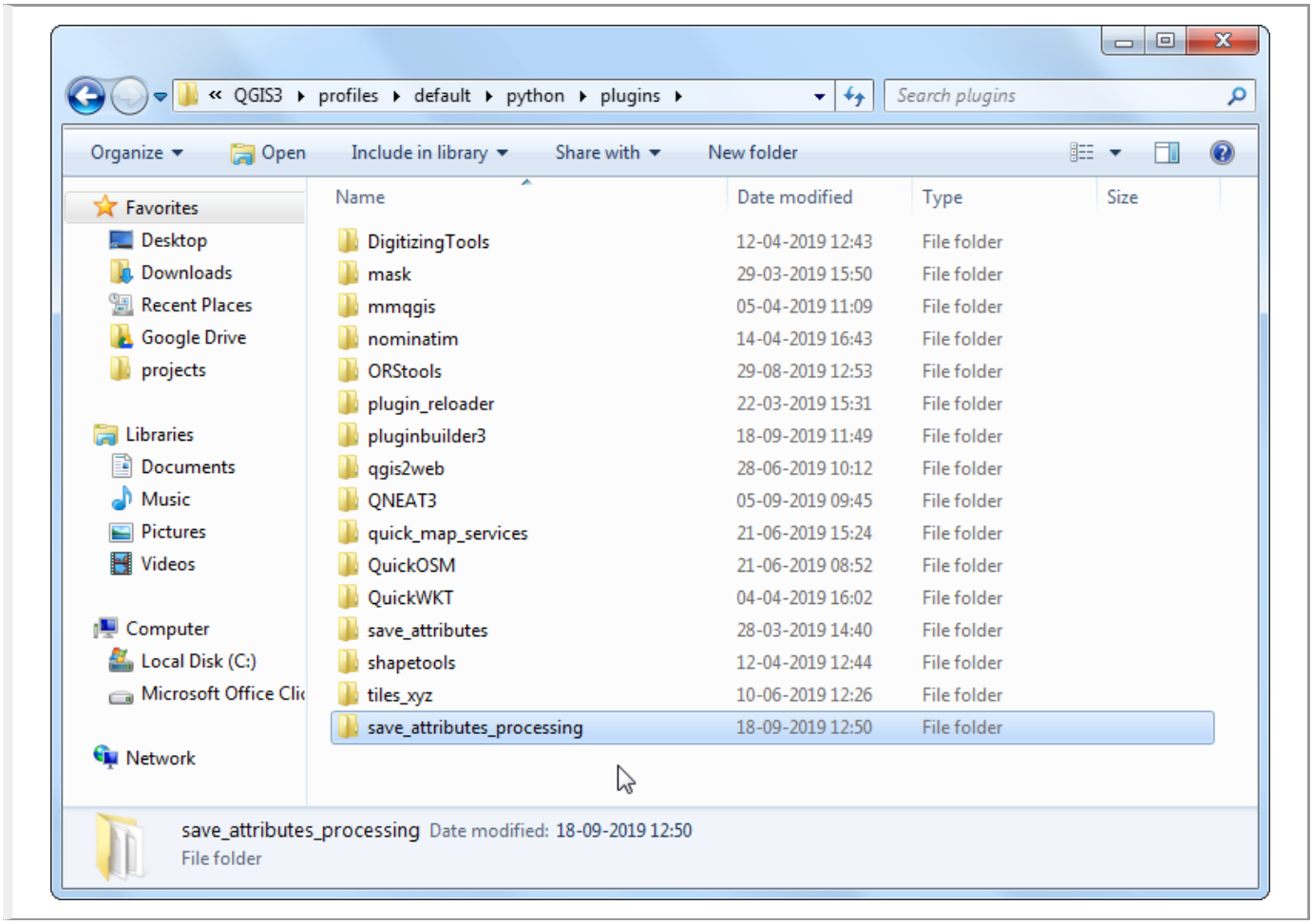
Note

You may get a prompt saying that pyrcc5 is not found in the path. You can ignore this message.

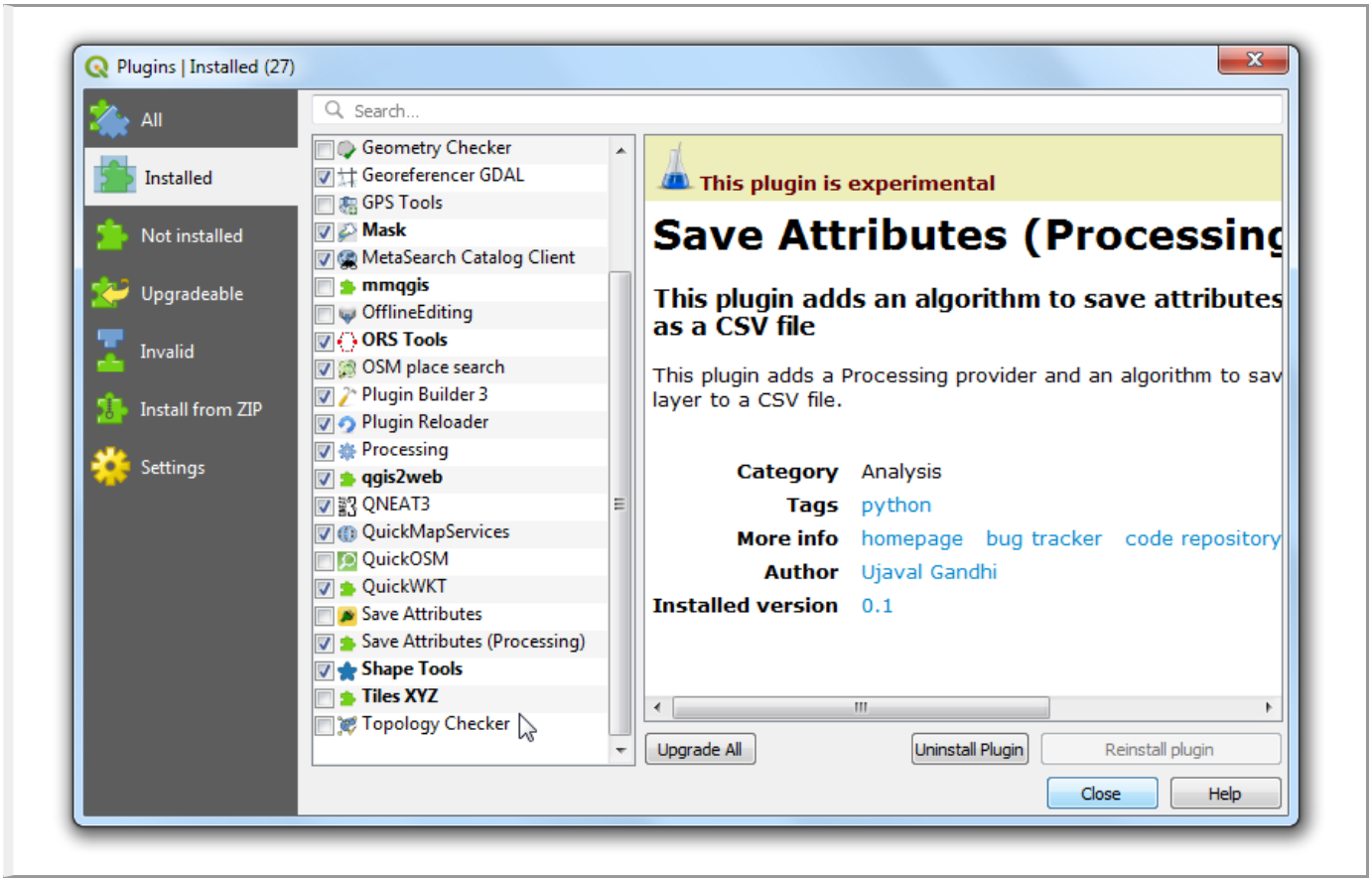
9. Plugins in QGIS are stored in a special folder. We must copy our plugin directory to that folder before it can be used. In QGIS, locate your current profile folder by going to Settings ‣ User Profiles ‣ Open Active Profile Folder.



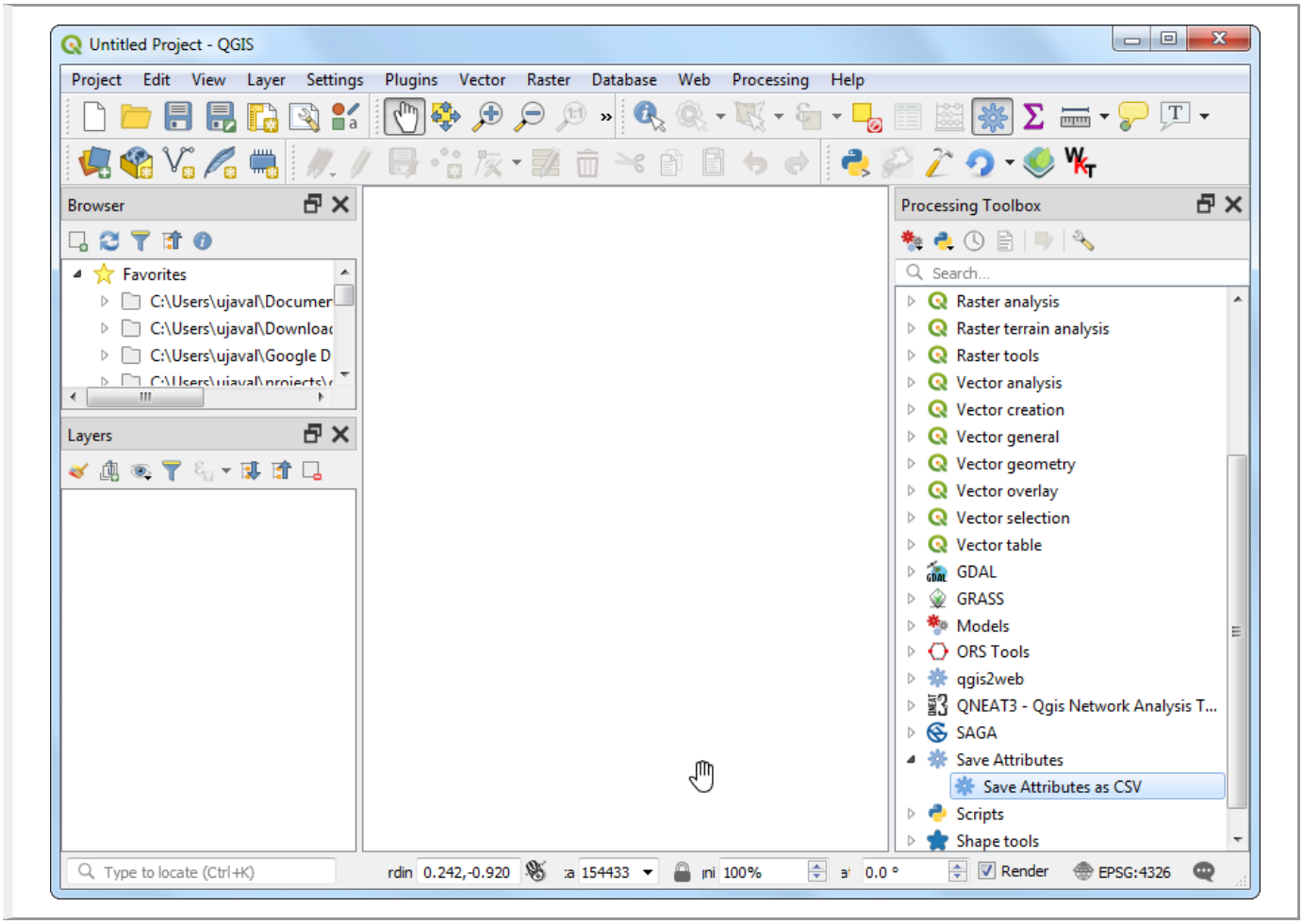
10. In the profile folder, copy the plugin folder to python ▸ plugins subfolder.



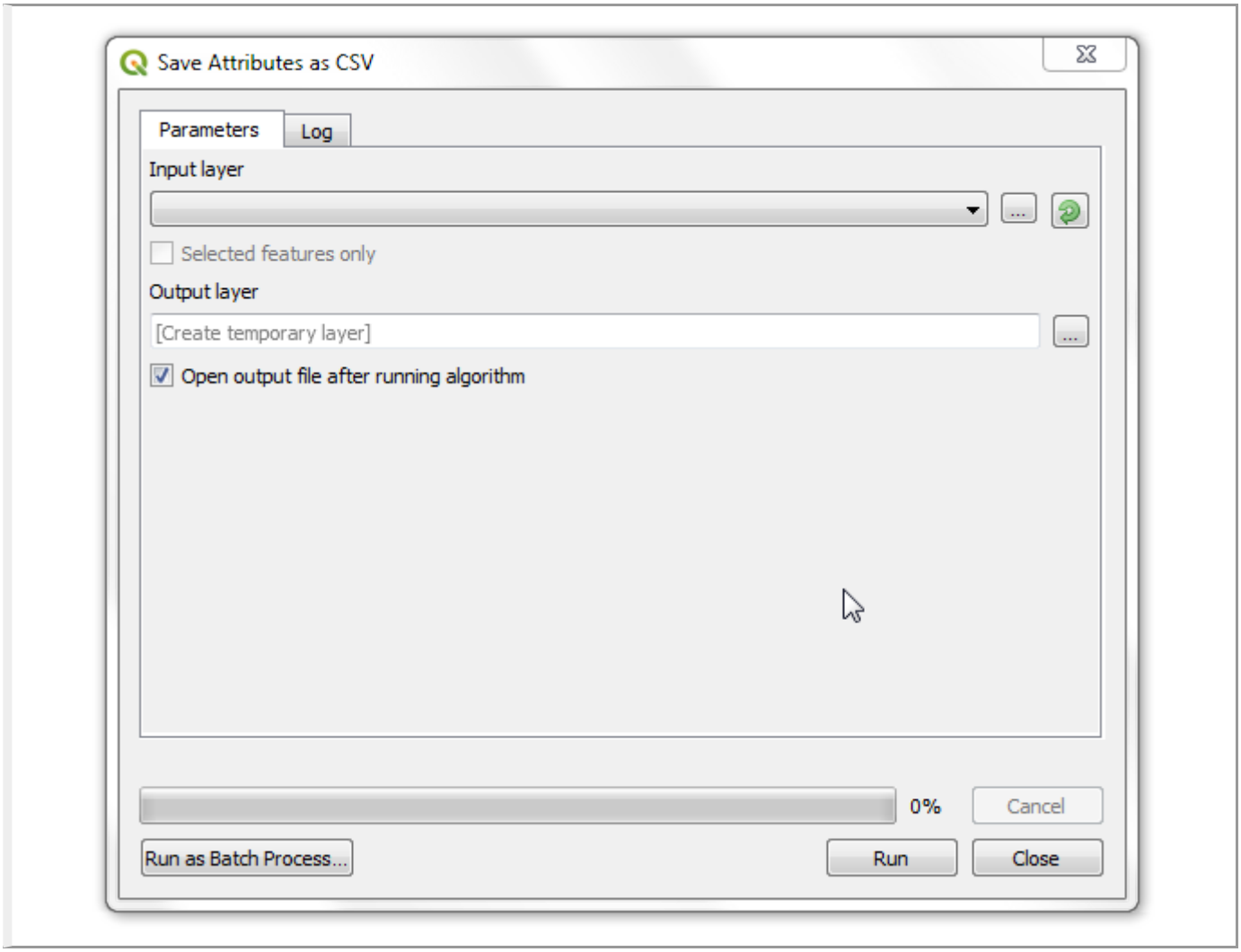
11. Now we are ready to have a first look at the brand new plugin we created. Close QGIS and launch it again. Go to Plugins ‣ Manage and Install plugins and enable the `Save Attributes (Processing)` plugin in the Installed tab.



12. Go to Processing ‣ Toolbox. You will notice that there is a new provider at the bottom called `Save Attributes` . Expand it to find an algorithm named `Save Attributes as CSV` . Double-click to launch it.



13. You will notice a familiar processing algorithm dialog with a dropdown for an input layer and a selector for an output layer. We will now customize this dialog to suit our needs. Close this dialog.

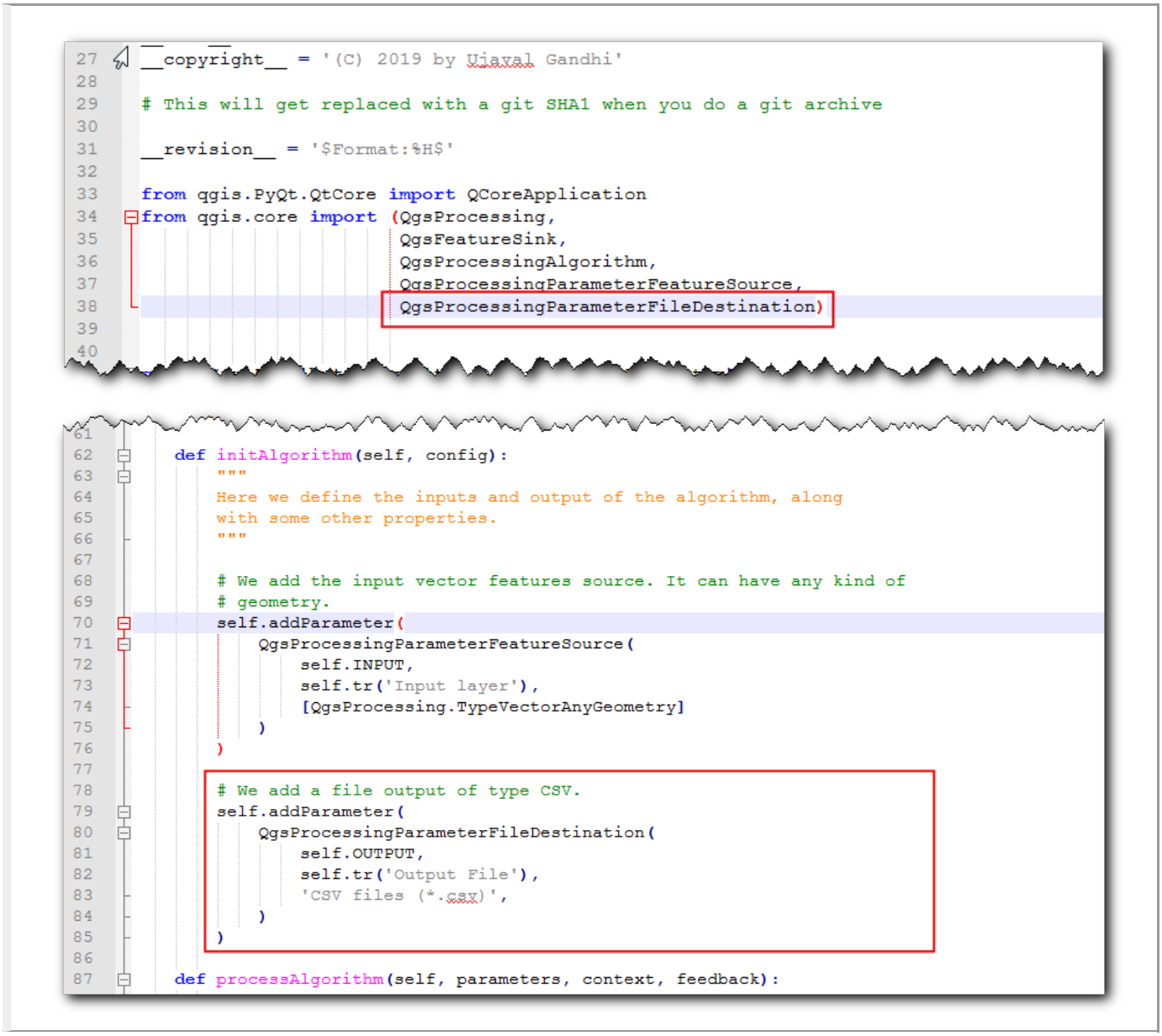


14. Go to the plugin directory and load the file `save_attributes_processing_algorithm.py` in a text editor. For our plugin, we take a vector layer as an input and write out a CSV file as output. So instead of importing `QgsProcessingParameterFeatureSink` as output - which is for vector layer - add `QgsProcessingParameterFileDestination` which is for a file.

```
from qgis.core import (QgsProcessing,
                       QgsFeatureSink,
                       QgsProcessingAlgorithm,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFileDestination)
```

Next, scroll down and define the output parameter under `initAlgorithm()` method with the following code.

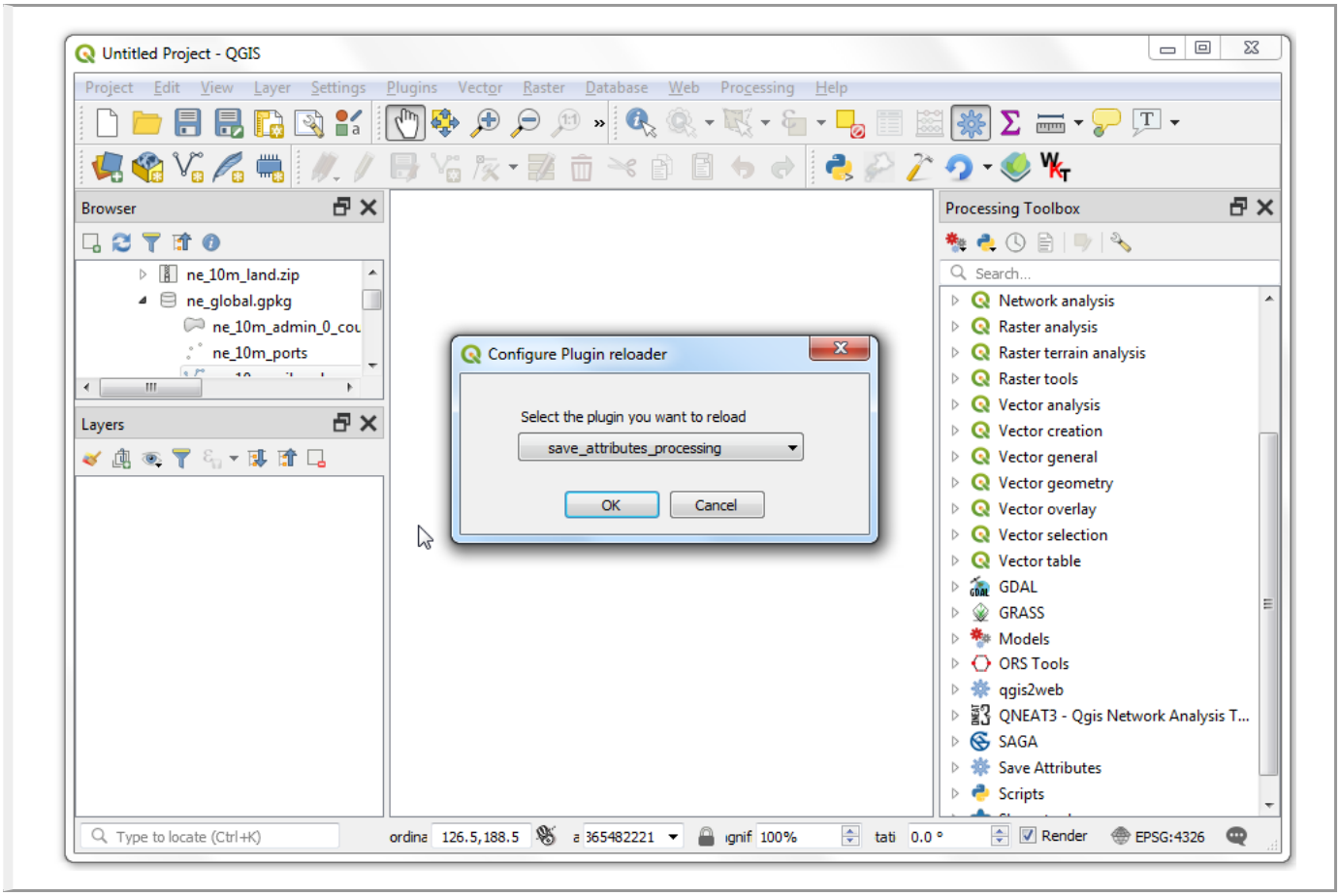
```
self.addParameter(
      QgsProcessingParameterFileDestination(
          self.OUTPUT,
          self.tr('Output File'),
          'CSV files (*.csv)',
      )
)
```



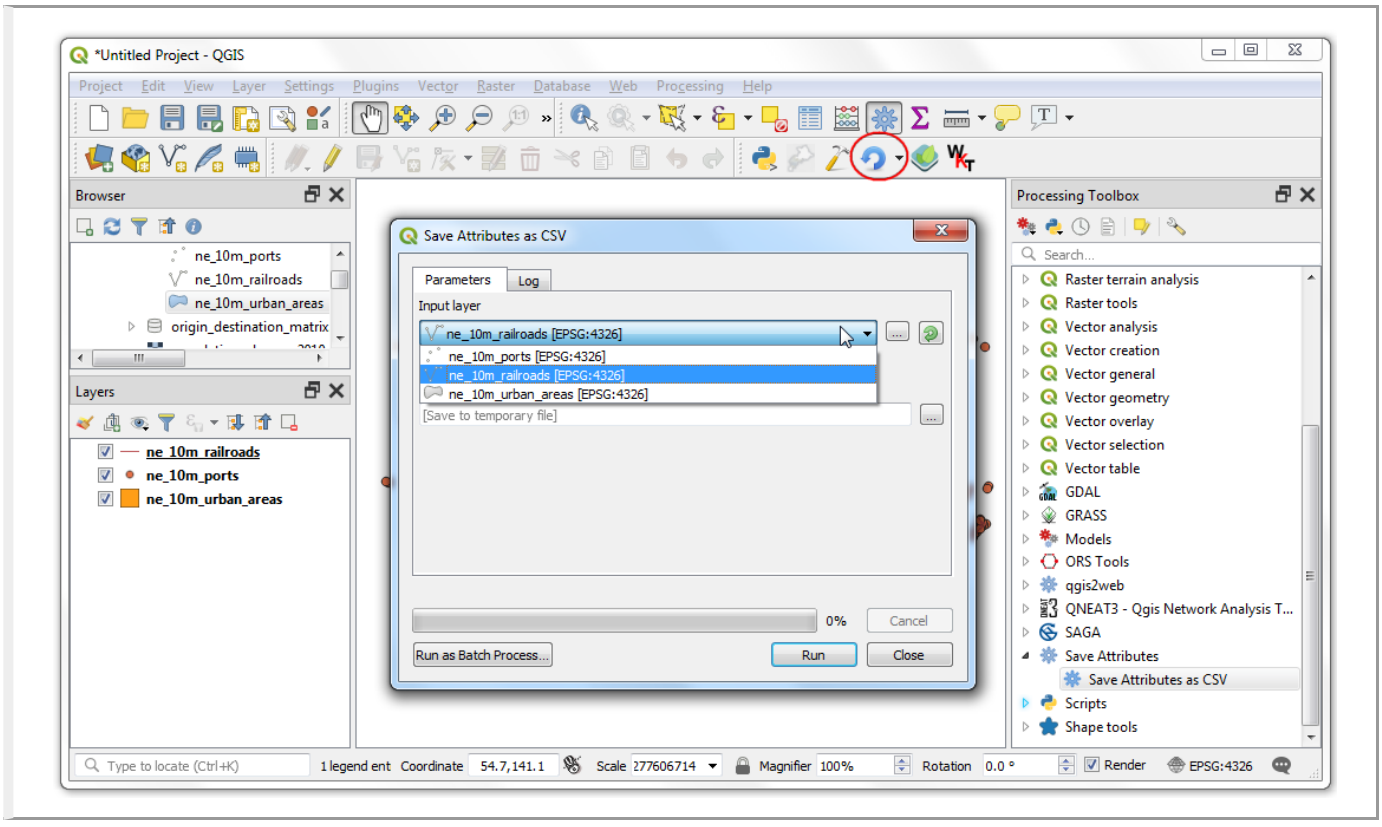
```
__copyright__ = '(C) 2019 by Ujaval Gandhi'

# This will get replaced with a git SHA1 when you do a git archive

__revision__ = '$Format:%H$'

from qgis.PyQt.QtCore import QCoreApplication
from qgis.core import (QgsProcessing,
                       QgsFeatureSink,
                       QgsProcessingAlgorithm,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFileDestination)
```

```
    def initAlgorithm(self, config):
        """
        Here we define the inputs and output of the algorithm, along
        with some other properties.
        """

        # We add the input vector features source. It can have any kind of
        # geometry.
        self.addParameter(
            QgsProcessingParameterFeatureSource(
                self.INPUT,
                self.tr('Input layer'),
                [QgsProcessing.TypeVectorAnyGeometry]
            )
        )

        # We add a file output of type CSV.
        self.addParameter(
            QgsProcessingParameterFileDestination(
                self.OUTPUT,
                self.tr('Output File'),
                'CSV files (*.csv)',
            )
        )

    def processAlgorithm(self, parameters, context, feedback):
```

15. Let's reload our plugin so we can see the changes in the dialog window. Go to Plugin ‣ Plugin Reloader ‣ Choose a plugin to be reloaded. Select `save_attributes_processing` in the Configure Plugin reloader dialog.



16. Click the Reload plugin button to load the latest version of the plugin. To test this new functionality, we must load some layers in QGIS. After you have loaded some layers, launch the Save Attributes ▸ Save Attributes as CSV algorithm. You will see the output is changed to a file instead of a layer.



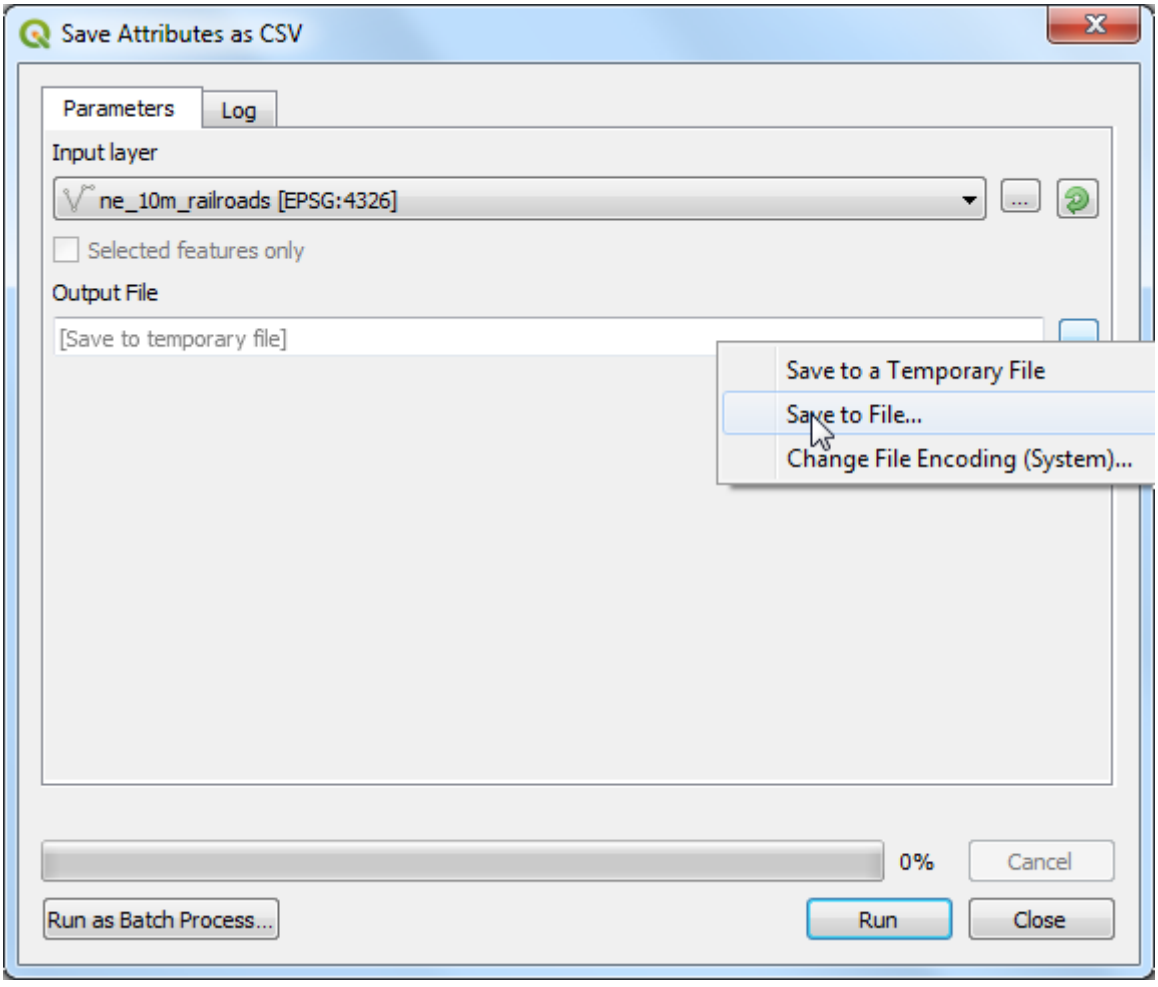
17. Let's add some logic to the algorithm that takes the selected vector layer and writes the attributes to a CSV file. The explanation for this code can be found in *Getting Started With Python Programming (QGIS3)* (getting_started_with_pyqgis.html). Notable difference here is the counter that helps show

the progress of the processing. Add the following code to the `processAlgorithm` method and save the file.

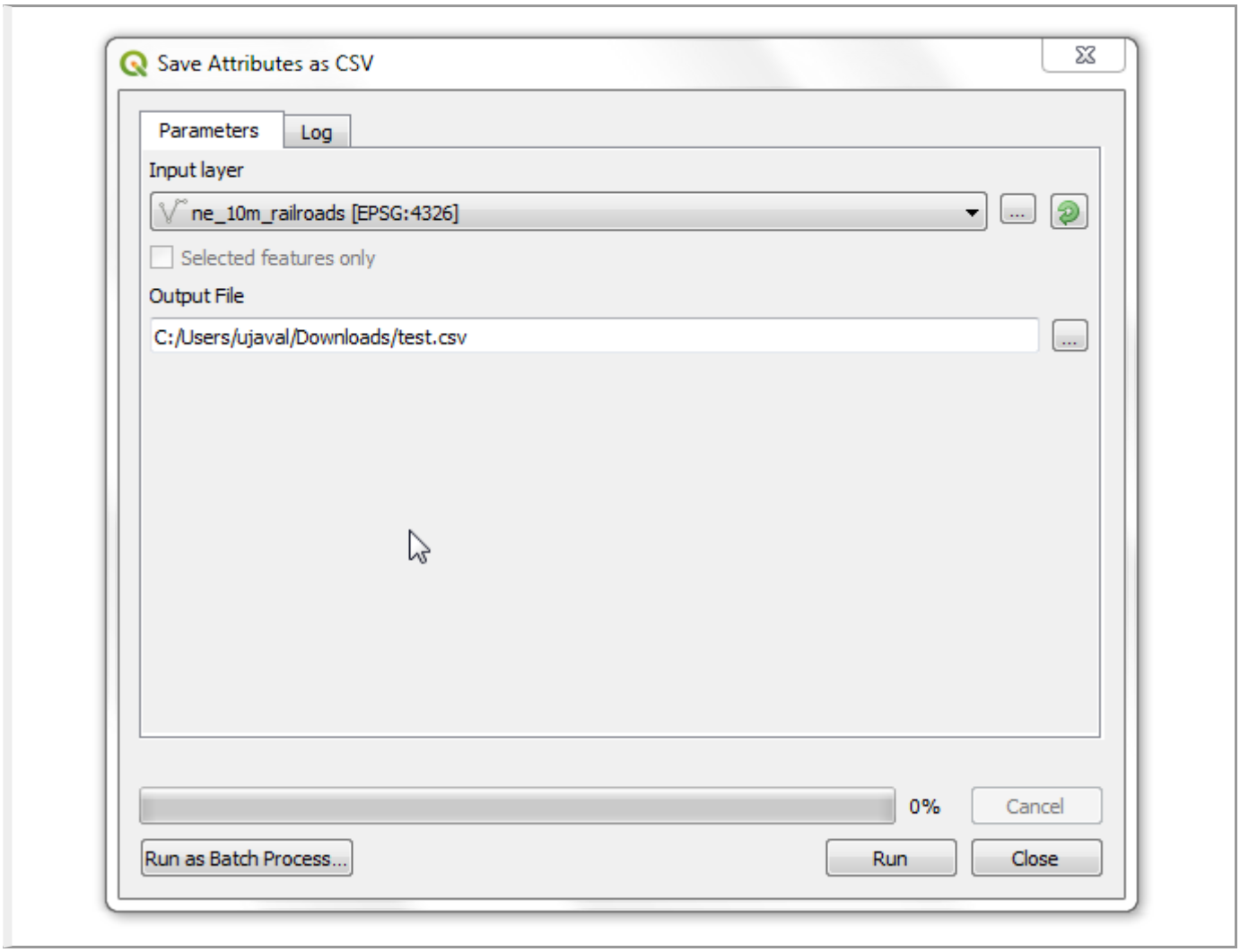
```
def processAlgorithm(self, parameters, context, feedback):
  """
  Here is where the processing itself takes place.
  """
  source = self.parameterAsSource(parameters, self.INPUT, context)
  csv = self.parameterAsFileOutput(parameters, self.OUTPUT, context)

  fieldnames = [field.name() for field in source.fields()]

  # Compute the number of steps to display within the progress bar and
  # get features from source
  total = 100.0 / source.featureCount() if source.featureCount() else 0
  features = source.getFeatures()

  with open(csv, 'w') as output_file:
    # write header
    line = ','.join(name for name in fieldnames) + '\n'
    output_file.write(line)
    for current, f in enumerate(features):
        # Stop the algorithm if cancel button has been clicked
        if feedback.isCanceled():
            break

        # Add a feature in the sink
        line = ','.join(str(f[name]) for name in fieldnames) + '\n'
        output_file.write(line)

        # Update the progress bar
        feedback.setProgress(int(current * total))

  return {self.OUTPUT: csv}
```

```
    def processAlgorithm(self, parameters, context, feedback):
        """
        Here is where the processing itself takes place.
        """.
        source = self.parameterAsSource(parameters, self.INPUT, context)
        csv = self.parameterAsFileOutput(parameters, self.OUTPUT, context)

        fieldnames = [field.name() for field in source.fields()]

        # Compute the number of steps to display within the progress bar and
        # get features from source
        total = 100.0 / source.featureCount() if source.featureCount() else 0
        features = source.getFeatures()

        with open(csv, 'w') as output_file:
            # write header
            line = ','.join(name for name in fieldnames) + '\n'
            output_file.write(line)
            for current, f in enumerate(features):
                # Stop the algorithm if cancel button has been clicked
                if feedback.isCanceled():
                    break

                # Add a feature in the sink
                line = ','.join(str(f[name]) for name in fieldnames) + '\n'
                output_file.write(line)

                # Update the progress bar
                feedback.setProgress(int(current * total))

        return {self.OUTPUT: csv}
```

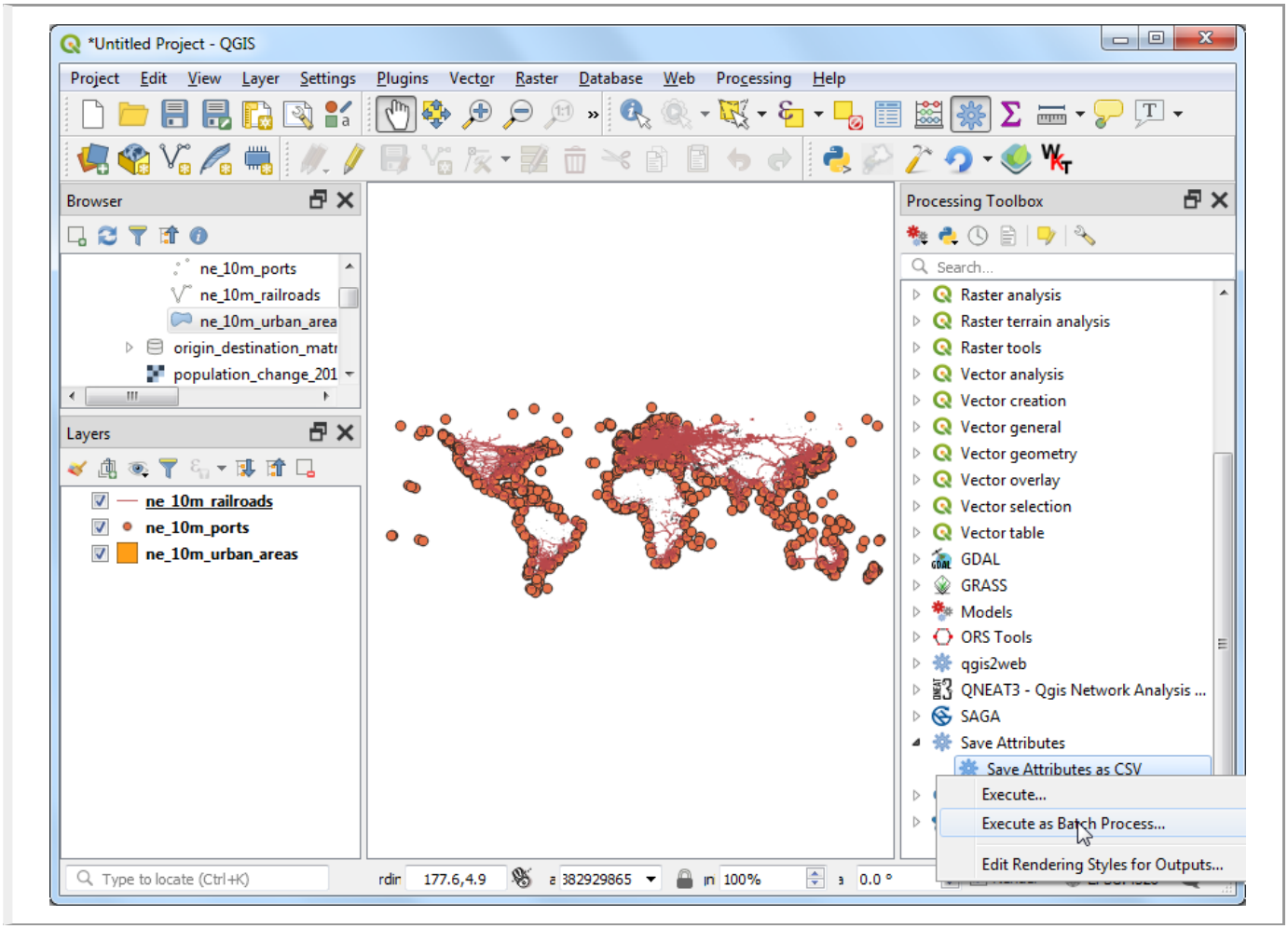
18. Back in the main QGIS window, reload the plugin by clicking on the Reload plugin button. Launch the Save Attributes ▸ Save Attributes as CSV algorithm. Select a layer for the Input layer. Next, click the … button next to Output file.



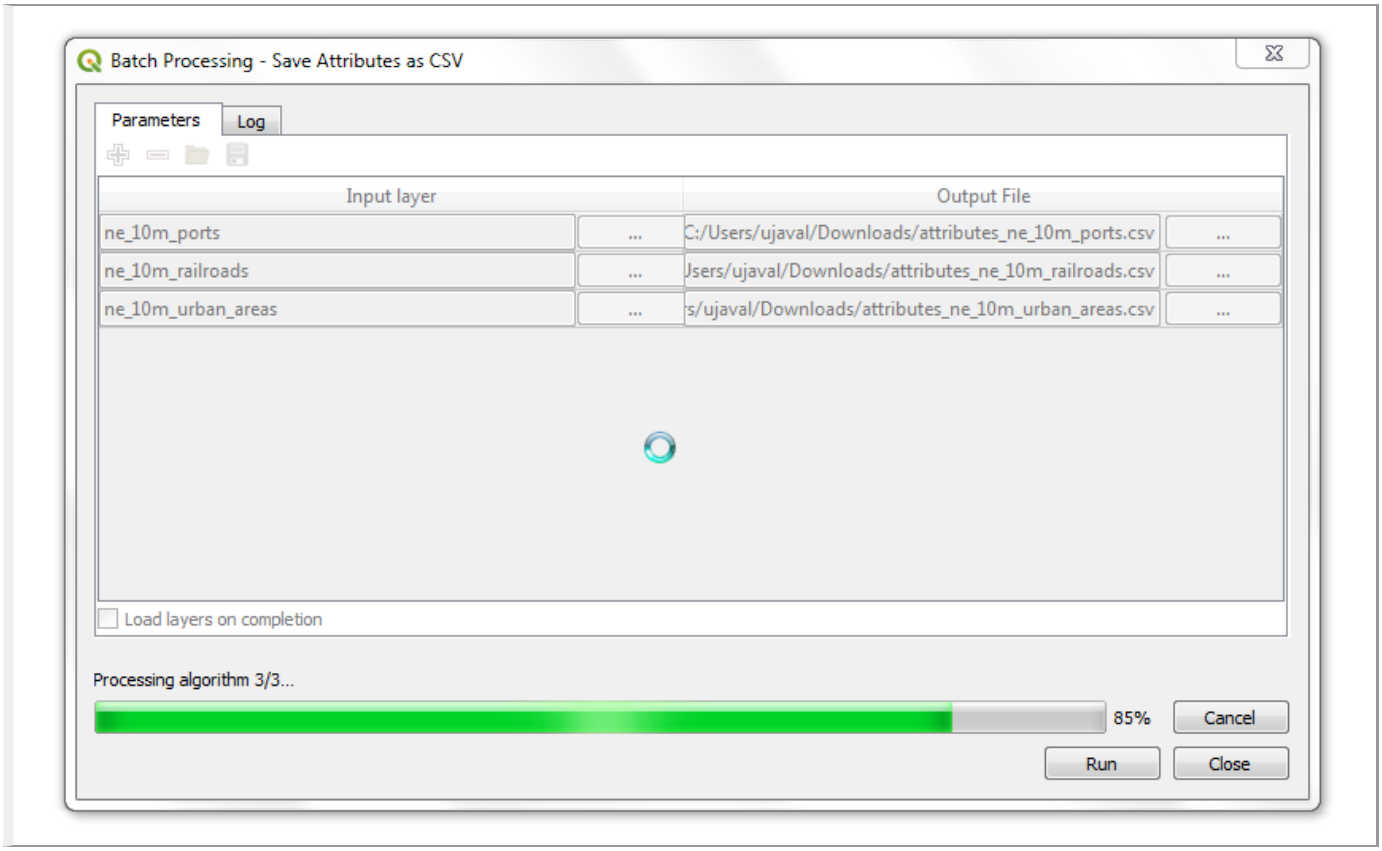
19. Name the output file `test.csv` and click Run. The algorithm will run and produce a CSV file at the chosen location.



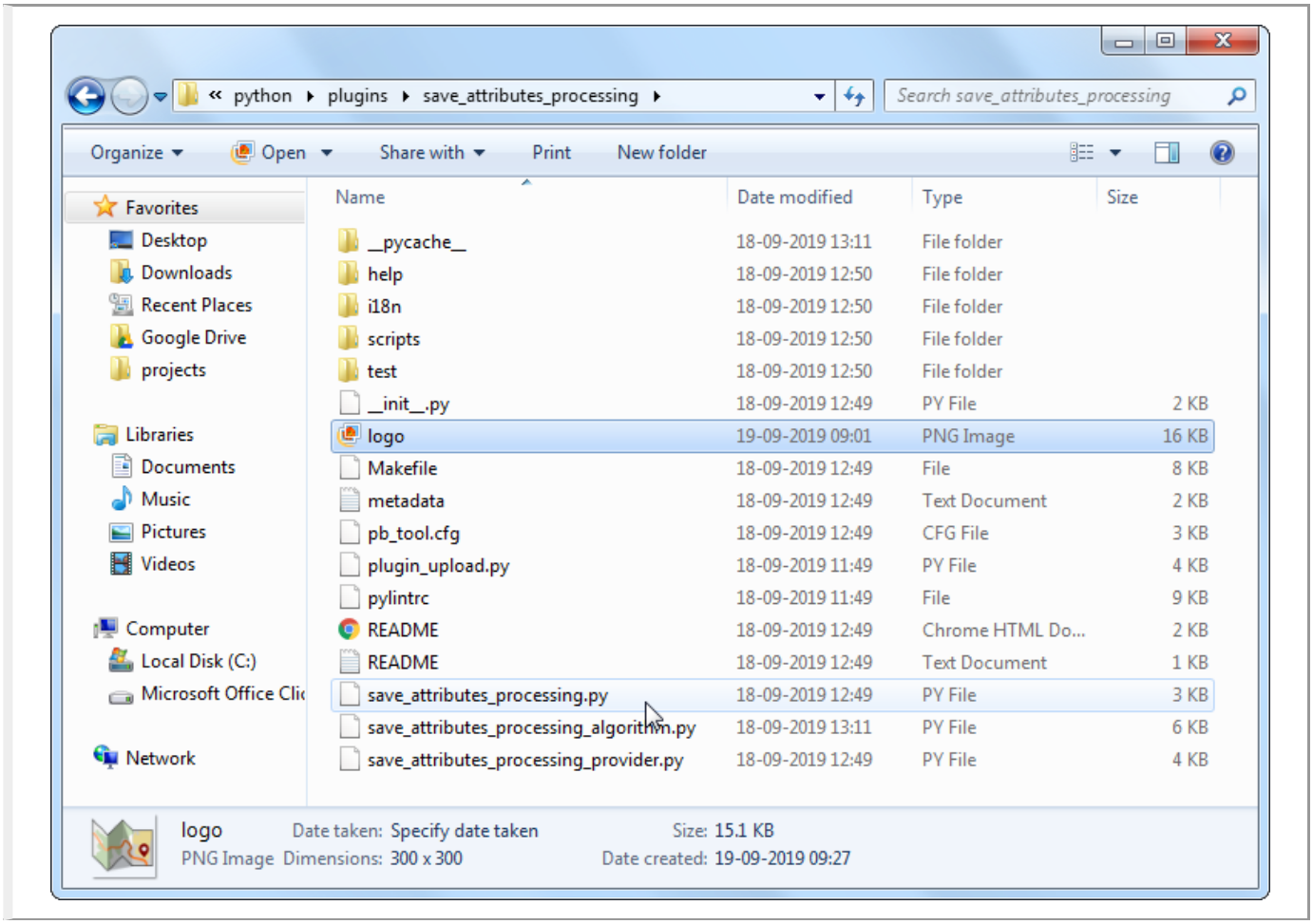
20. As mentioned earlier, even though this algorithm comes from a plugin, it integrates very well with the built-in processing tools. To demonstrate this, let's run this algorithm using the built-in batch processing interface. Right-click the algorithm and select Execute as Batch Process...



21. You can select multiple inputs and run this algorithm in a batch to produce multiple CSV files in a single run. If you are not familiar with the batch processing interface, see *Batch Processing using Processing Framework (QGIS3)* (batch_processing.html) for step-by-step instructions.



22. The plugin is ready and you can ship it in the current form. But we can improve the user experience by making the processing plugin behave like a regular plugin. Using the hybrid approach outlined below, you can add a menu item and a toolbar button. This way, you give the users an easier way to discover and launch the tools that are installed as part of the plugin. We will need an icon for the plugin. Download logo.png (http://www.qgistutorials.com/downloads/logo.png) and copy it to the plugin directory.



23. Open the file `save_attributes_processing.py` . Add the following imports at top of the file.

```
from qgis.PyQt.QtWidgets import QAction
from qgis.PyQt.QtGui import QIcon

from qgis.core import QgsProcessingAlgorithm, QgsApplication
import processing
```

Scroll down and locate the `initGui` method. It only contains the code to initialize the processing provider. We will add the code to add a toolbar button and a menu item. We will also need to add code to the `unload` method, to remove these elements when plugin is removed.

```
def initGui(self):
  self.initProcessing()

  icon = os.path.join(os.path.join(cmd_folder, 'logo.png'))
  self.action = QAction(
      QIcon(icon),
      u"Save Attributes as CSV", self.iface.mainWindow())
  self.action.triggered.connect(self.run)
  self.iface.addPluginToMenu(u"&SaveAttributes", self.action)
  self.iface.addToolBarIcon(self.action)

def unload(self):
  QgsApplication.processingRegistry().removeProvider(self.provider)
  self.iface.removePluginMenu(u"&SaveAttributes", self.action)
  self.iface.removeToolBarIcon(self.action)
```

We have connected the button and the menu item to trigger the `run` method when clicked. Add a new method at the bottom that uses the helper method `execAlgorithmDialog` to launch the processing algorithm.

```
def run(self):
  processing.execAlgorithmDialog("Save Attributes:Save Attributes as CSV")
```

```
import os
import sys
import inspect

from qgis.PyQt.QtWidgets import QAction
from qgis.PyQt.QtGui import QIcon

from qgis.core import QgsProcessingAlgorithm, QgsApplication
import processing
from .save_attributes_processing_provider import SaveAttributesProvider
```

```
        QgsApplication.processingRegistry().addProvider(self.provider)

    def initGui(self):
        self.initProcessing()

        icon = os.path.join(os.path.join(cmd_folder, 'logo.png'))
        self.action = QAction(
            QIcon(icon),
            u"Save Attributes as CSV", self.iface.mainWindow())
        self.action.triggered.connect(self.run)
        self.iface.addPluginToMenu(u"&SaveAttributes", self.action)
        self.iface.addToolBarIcon(self.action)

    def unload(self):
        QgsApplication.processingRegistry().removeProvider(self.provider)
        self.iface.removePluginMenu(u"&SaveAttributes", self.action)
        self.iface.removeToolBarIcon(self.action)
```

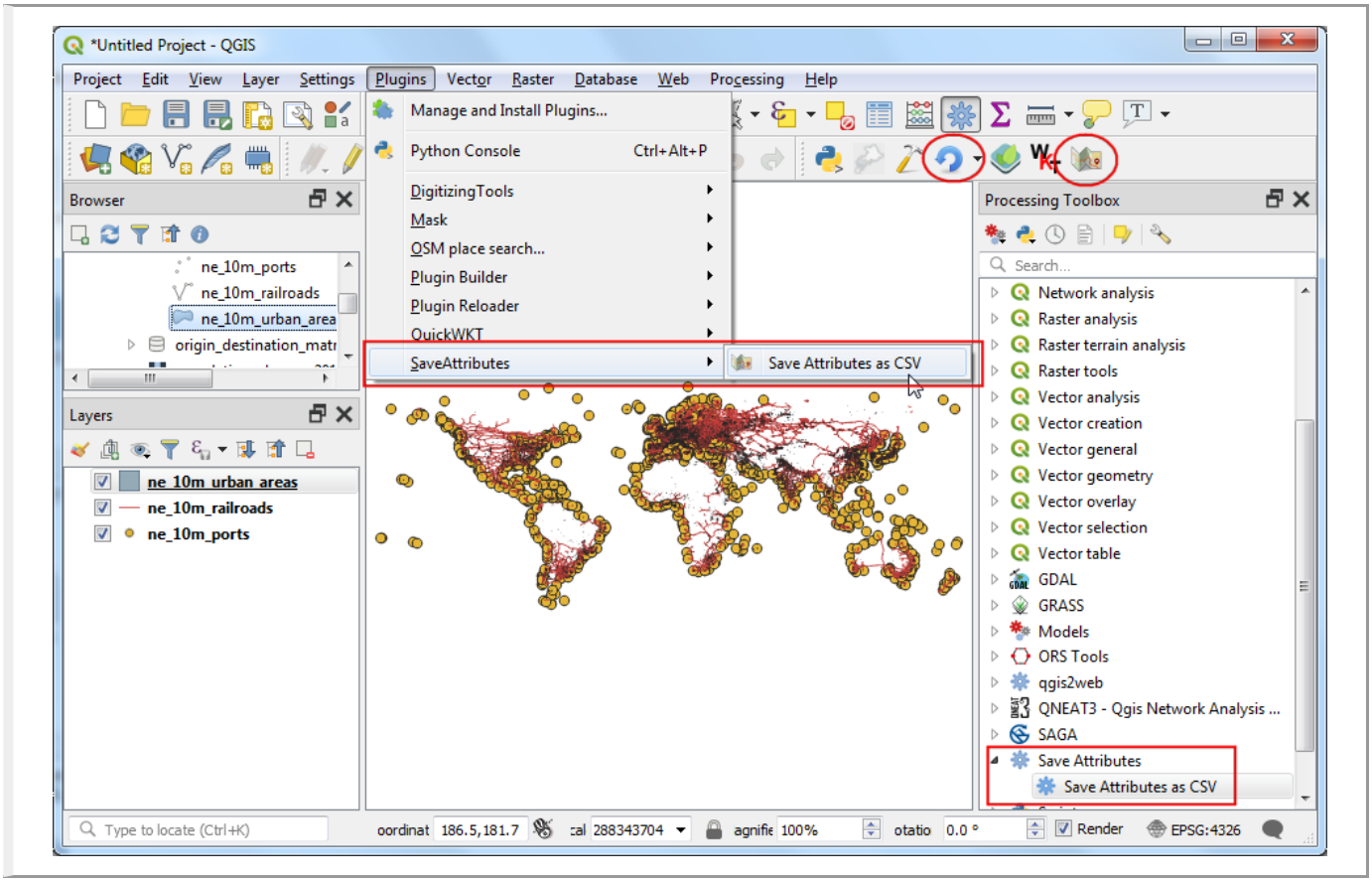
```
        self.iface.removePluginMenu(u"&SaveAttributes", self.action)
        self.iface.removeToolBarIcon(self.action)

    def run(self):
        processing.execAlgorithmDialog("Save Attributes:Save Attributes as CSV")
```

24. Next, we need a minor fix to the `__init__.py` file in the plugin directory. Open the file and add `iface` to the return statement, so the reference to the QGIS interface is passed on to the plugin.

```
# noinspection PyPep8Naming
def classFactory(iface):  # pylint: disable=invalid-name
    """Load SaveAttributes class from file SaveAttributes.

    :param iface: A QGIS interface instance.
    :type iface: QgsInterface
    """
    #
    from .save_attributes_processing import SaveAttributesPlugin
    return SaveAttributesPlugin(iface)
```

25. Back in the main QGIS window, reload the plugin by clicking on the Reload plugin button. You will see a new toolbar icon and a menu item under Plugins ▸ SaveAttributes ▸ Save Attributes as CSV. You can

click these to launch the `Save Attributes as CSV` algorithm. You will notice that the processing provider and the algorithm in the toolbar still have the default icons. Let's fix that.

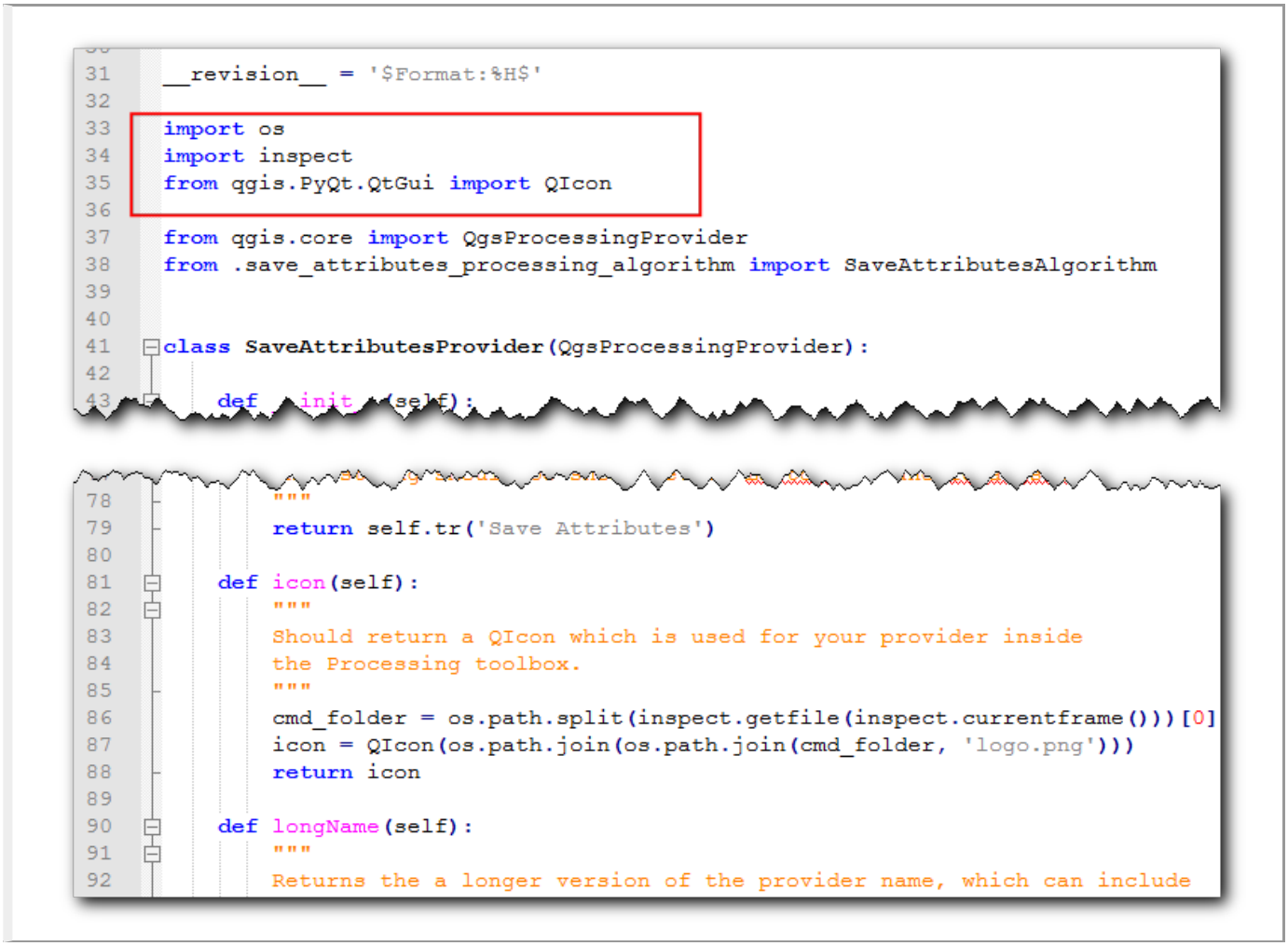


26. Open the `save_attributes_processing_provider.py` file from the plugin directory. Add the imports at the top as follows.

```
import os
import inspect
from qgis.PyQt.QtGui import QIcon
```

Modify the `icon` method as follows to add the custom icon.

```
def icon(self):
  cmd_folder = os.path.split(inspect.getfile(inspect.currentframe()))[0]
  icon = QIcon(os.path.join(os.path.join(cmd_folder, 'logo.png')))
  return icon
```



27. Next, open the `save_attributes_processing_algorithm.py` file. Add the imports at the top as follows.

```
import os
import inspect
from qgis.PyQt.QtGui import QIcon
```

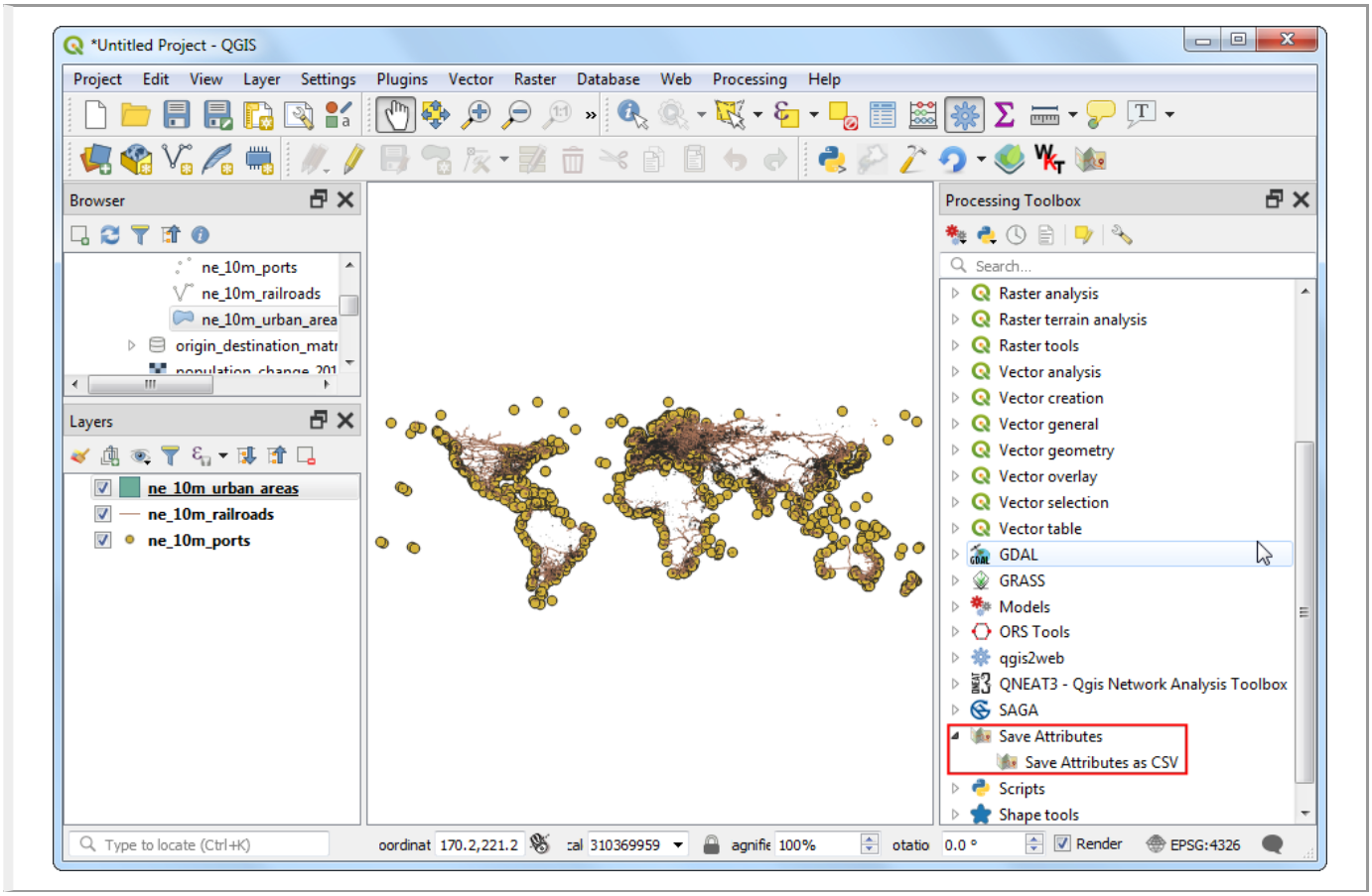
Add a new `icon` method with the following code.

```
def icon(self):
  cmd_folder = os.path.split(inspect.getfile(inspect.currentframe()))[0]
  icon = QIcon(os.path.join(os.path.join(cmd_folder, 'logo.png')))
  return icon
```

```
32
33  import os
34  import inspect
35  from qgis.PyQt.QtGui import QIcon
36
37  from qgis.PyQt.QtCore import QCoreApplication
38  from qgis.core import (QgsProcessing,
39                         QgsFeatureSink,
40                         QgsProcessingAlgorithm,
41                         QgsProcessingParameterFeatureSource,
42                         QgsProcessingParameterFileDestination)
43
44
45  class SaveAttributesAlgorithm(QgsProcessingAlgorithm):
46      """
47      This is an example algorithm that takes a vector layer and
```

```
155         return ''
156
157     def tr(self, string):
158         return QCoreApplication.translate('Processing', string)
159
160     def icon(self):
161         """
162         Should return a QIcon which is used for your provider inside
163         the Processing toolbox.
164         """
165         cmd_folder = os.path.split(inspect.getfile(inspect.currentframe()))[0]
166         icon = QIcon(os.path.join(os.path.join(cmd_folder, 'logo.png')))
167         return icon
168
169     def createInstance(self):
170         return SaveAttributesAlgorithm()
171
```

28. Reload the plugin and you will see the provider and algorithm both have our custom icon.



29. You can zip the plugin directory and share it with your users. They can unzip the contents to their plugin directory and try out your plugin. If this was a real plugin, you would upload it to the QGIS Plugin Repository (https://plugins.qgis.org/) so that all QGIS users will be able to find and download your plugin.

Note

This plugin is for demonstration purpose only. Do not publish this plugin or upload it to the QGIS plugin repository.

Below are the full source file as a reference.

`__init__.py`

```
# -*- coding: utf-8 -*-
"""
/***************************************************************************
 SaveAttributes
                                 A QGIS plugin
 This plugin adds an algorithm to save attributes of selected layer as a CSV file
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                             -------------------
        begin                : 2019-09-18
        copyright            : (C) 2019 by Ujaval Gandhi
        email                : ujaval@spatialthoughts.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
 This script initializes the plugin, making it known to QGIS.
"""

__author__ = 'Ujaval Gandhi'
__date__ = '2019-09-18'
__copyright__ = '(C) 2019 by Ujaval Gandhi'


# noinspection PyPep8Naming
def classFactory(iface):  # pylint: disable=invalid-name
    """Load SaveAttributes class from file SaveAttributes.

    :param iface: A QGIS interface instance.
    :type iface: QgsInterface
    """
    #
    from .save_attributes_processing import SaveAttributesPlugin
    return SaveAttributesPlugin(iface)
```

save_attributes_processing.py

```
# -*- coding: utf-8 -*-

"""
/***************************************************************************
 SaveAttributes
                                 A QGIS plugin
 This plugin adds an algorithm to save attributes of selected layer as a CSV file
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                              -------------------
        begin                : 2019-09-18
        copyright            : (C) 2019 by Ujaval Gandhi
        email                : ujaval@spatialthoughts.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""

__author__ = 'Ujaval Gandhi'
__date__ = '2019-09-18'
__copyright__ = '(C) 2019 by Ujaval Gandhi'

# This will get replaced with a git SHA1 when you do a git archive

__revision__ = '$Format:%H$'

import os
import sys
import inspect

from qgis.PyQt.QtWidgets import QAction
from qgis.PyQt.QtGui import QIcon

from qgis.core import QgsProcessingAlgorithm, QgsApplication
import processing
from .save_attributes_processing_provider import SaveAttributesProvider

cmd_folder = os.path.split(inspect.getfile(inspect.currentframe()))[0]

if cmd_folder not in sys.path:
    sys.path.insert(0, cmd_folder)


class SaveAttributesPlugin(object):

    def __init__(self, iface):
        self.provider = None
        self.iface = iface

    def initProcessing(self):
        """Init Processing provider for QGIS >= 3.8."""
```

```
        self.provider = SaveAttributesProvider()
        QgsApplication.processingRegistry().addProvider(self.provider)

    def initGui(self):
        self.initProcessing()

        icon = os.path.join(os.path.join(cmd_folder, 'logo.png'))
        self.action = QAction(
            QIcon(icon),
            u"Save Attributes as CSV", self.iface.mainWindow())
        self.action.triggered.connect(self.run)
        self.iface.addPluginToMenu(u"&SaveAttributes", self.action)
        self.iface.addToolBarIcon(self.action)

    def unload(self):
        QgsApplication.processingRegistry().removeProvider(self.provider)
        self.iface.removePluginMenu(u"&SaveAttributes", self.action)
        self.iface.removeToolBarIcon(self.action)

    def run(self):
      processing.execAlgorithmDialog("Save Attributes:Save Attributes as CSV")
```

save_attributes_processing_algorithm.py

```
# -*- coding: utf-8 -*-

"""
/***************************************************************************
 SaveAttributes
                                 A QGIS plugin
 This plugin adds an algorithm to save attributes of selected layer as a CSV file
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                              -------------------
        begin                : 2019-09-18
        copyright            : (C) 2019 by Ujaval Gandhi
        email                : ujaval@spatialthoughts.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""

__author__ = 'Ujaval Gandhi'
__date__ = '2019-09-18'
__copyright__ = '(C) 2019 by Ujaval Gandhi'

# This will get replaced with a git SHA1 when you do a git archive

__revision__ = '$Format:%H$'

import os
import inspect
from qgis.PyQt.QtGui import QIcon

from qgis.PyQt.QtCore import QCoreApplication
from qgis.core import (QgsProcessing,
                       QgsFeatureSink,
                       QgsProcessingAlgorithm,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFileDestination)


class SaveAttributesAlgorithm(QgsProcessingAlgorithm):
    """
    This is an example algorithm that takes a vector layer and
    creates a new identical one.

    It is meant to be used as an example of how to create your own
    algorithms and explain methods and variables used to do it. An
    algorithm like this will be available in all elements, and there
    is not need for additional work.

    All Processing algorithms should extend the QgsProcessingAlgorithm
    class.
    """
```

```
    # Constants used to refer to parameters and outputs. They will be
    # used when calling the algorithm from another algorithm, or when
    # calling from the QGIS console.

    OUTPUT = 'OUTPUT'
    INPUT = 'INPUT'

    def initAlgorithm(self, config):
        """
        Here we define the inputs and output of the algorithm, along
        with some other properties.
        """

        # We add the input vector features source. It can have any kind of
        # geometry.
        self.addParameter(
            QgsProcessingParameterFeatureSource(
                self.INPUT,
                self.tr('Input layer'),
                [QgsProcessing.TypeVectorAnyGeometry]
            )
        )

        # We add a file output of type CSV.
        self.addParameter(
            QgsProcessingParameterFileDestination(
                self.OUTPUT,
                self.tr('Output File'),
                'CSV files (*.csv)',
            )
        )

    def processAlgorithm(self, parameters, context, feedback):
        """
        Here is where the processing itself takes place.
        """
        source = self.parameterAsSource(parameters, self.INPUT, context)
        csv = self.parameterAsFileOutput(parameters, self.OUTPUT, context)

        fieldnames = [field.name() for field in source.fields()]

        # Compute the number of steps to display within the progress bar and
        # get features from source
        total = 100.0 / source.featureCount() if source.featureCount() else 0
        features = source.getFeatures()

        with open(csv, 'w') as output_file:
          # write header
          line = ','.join(name for name in fieldnames) + '\n'
          output_file.write(line)
          for current, f in enumerate(features):
              # Stop the algorithm if cancel button has been clicked
              if feedback.isCanceled():
                  break

              # Add a feature in the sink
              line = ','.join(str(f[name]) for name in fieldnames) + '\n'
```

```
                output_file.write(line)

                # Update the progress bar
                feedback.setProgress(int(current * total))

        return {self.OUTPUT: csv}

    def name(self):
        """
        Returns the algorithm name, used for identifying the algorithm. This
        string should be fixed for the algorithm, and must not be localised.
        The name should be unique within each provider. Names should contain
        lowercase alphanumeric characters only and no spaces or other
        formatting characters.
        """
        return 'Save Attributes as CSV'

    def displayName(self):
        """
        Returns the translated algorithm name, which should be used for any
        user-visible display of the algorithm name.
        """
        return self.tr(self.name())

    def group(self):
        """
        Returns the name of the group this algorithm belongs to. This string
        should be localised.
        """
        return self.tr(self.groupId())

    def groupId(self):
        """
        Returns the unique ID of the group this algorithm belongs to. This
        string should be fixed for the algorithm, and must not be localised.
        The group id should be unique within each provider. Group id should
        contain lowercase alphanumeric characters only and no spaces or other
        formatting characters.
        """
        return ''

    def tr(self, string):
        return QCoreApplication.translate('Processing', string)

    def icon(self):
        """
        Should return a QIcon which is used for your provider inside
        the Processing toolbox.
        """
        cmd_folder = os.path.split(inspect.getfile(inspect.currentframe()))[0]
        icon = QIcon(os.path.join(os.path.join(cmd_folder, 'logo.png')))
        return icon

    def createInstance(self):
        return SaveAttributesAlgorithm()
```

save_attributes_processing_provider.py

```
# -*- coding: utf-8 -*-

"""
/***************************************************************************
 SaveAttributes
                                 A QGIS plugin
 This plugin adds an algorithm to save attributes of selected layer as a CSV file
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                              -------------------
        begin                : 2019-09-18
        copyright            : (C) 2019 by Ujaval Gandhi
        email                : ujaval@spatialthoughts.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""

__author__ = 'Ujaval Gandhi'
__date__ = '2019-09-18'
__copyright__ = '(C) 2019 by Ujaval Gandhi'

# This will get replaced with a git SHA1 when you do a git archive

__revision__ = '$Format:%H$'

import os
import inspect
from qgis.PyQt.QtGui import QIcon

from qgis.core import QgsProcessingProvider
from .save_attributes_processing_algorithm import SaveAttributesAlgorithm


class SaveAttributesProvider(QgsProcessingProvider):

    def __init__(self):
        """
        Default constructor.
        """
        QgsProcessingProvider.__init__(self)

    def unload(self):
        """
        Unloads the provider. Any tear-down steps required by the provider
        should be implemented here.
        """
        pass

    def loadAlgorithms(self):
        """
```

```
        Loads all algorithms belonging to this provider.
        """
        self.addAlgorithm(SaveAttributesAlgorithm())
        # add additional algorithms here
        # self.addAlgorithm(MyOtherAlgorithm())

    def id(self):
        """
        Returns the unique provider id, used for identifying the provider. This
        string should be a unique, short, character only string, eg "qgis" or
        "gdal". This string should not be localised.
        """
        return 'Save Attributes'

    def name(self):
        """
        Returns the provider name, which is used to describe the provider
        within the GUI.

        This string should be short (e.g. "Lastools") and localised.
        """
        return self.tr('Save Attributes')

    def icon(self):
        """
        Should return a QIcon which is used for your provider inside
        the Processing toolbox.
        """
        cmd_folder = os.path.split(inspect.getfile(inspect.currentframe()))[0]
        icon = QIcon(os.path.join(os.path.join(cmd_folder, 'logo.png')))
        return icon

    def longName(self):
        """
        Returns the a longer version of the provider name, which can include
        extra details such as version numbers. E.g. "Lastools LIDAR tools
        (version 2.2.1)". This string should be localised. The default
        implementation returns the same string as name().
        """
        return self.name()
```

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Using Custom Python Expression Functions (QGIS3)

Expressions in QGIS have a lot of power and are used in many core features: selection, calculating field values, styling, labelling etc. QGIS also has support for user-defined expressions. With a little bit of python programming, you can define your own functions that can be used within the expression engine.

Overview of the task

We will define a custom function that finds the UTM zone number (http://www.dmap.co.uk/utmworld.htm) of a map feature and use this function to write an expression that displays the UTM zone as a map tip when hovered over the point.

Other skills you will learn

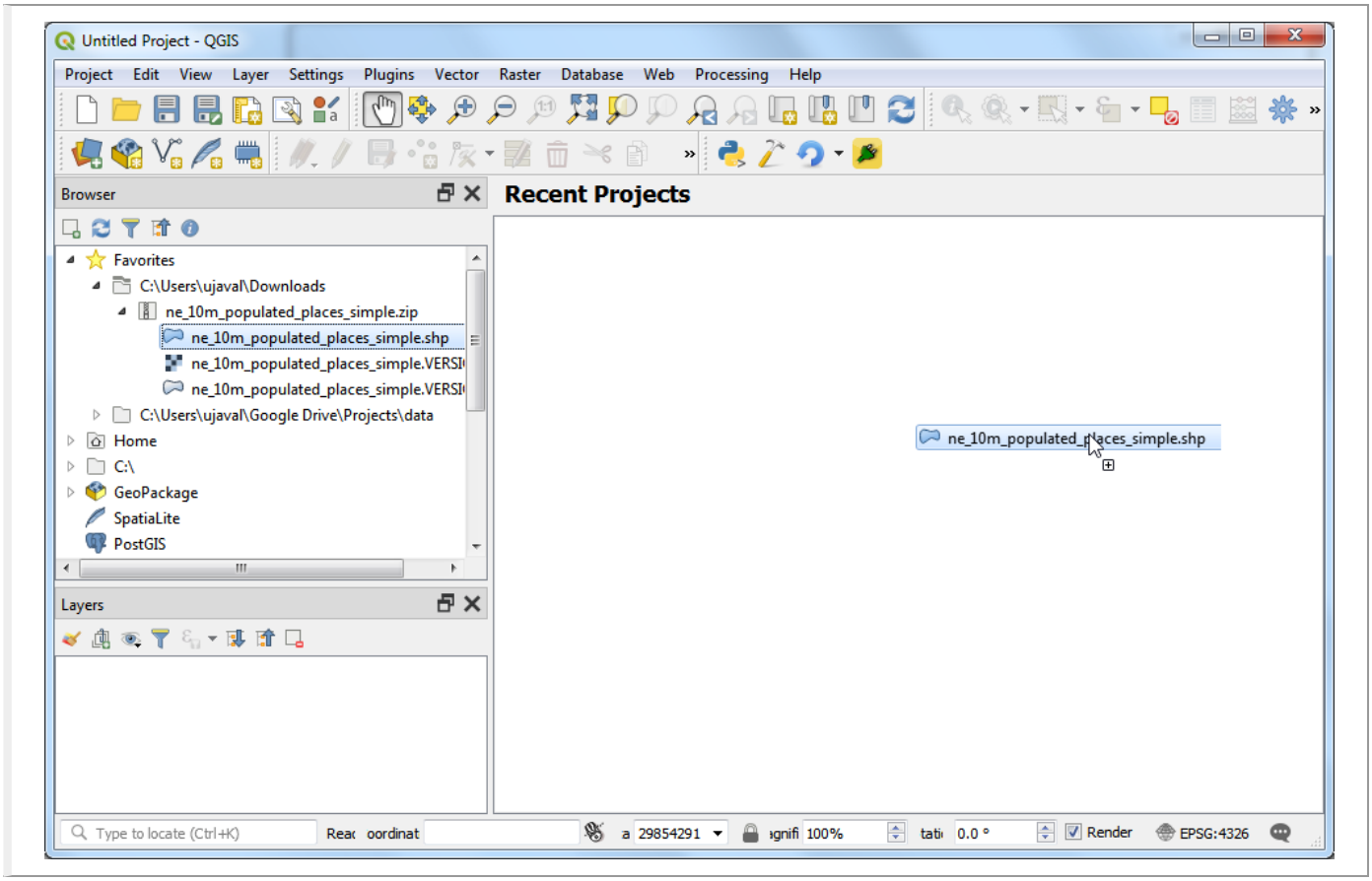
- How to use the `Map Tips` tool to display custom text when hovering over a feature.

Get the data

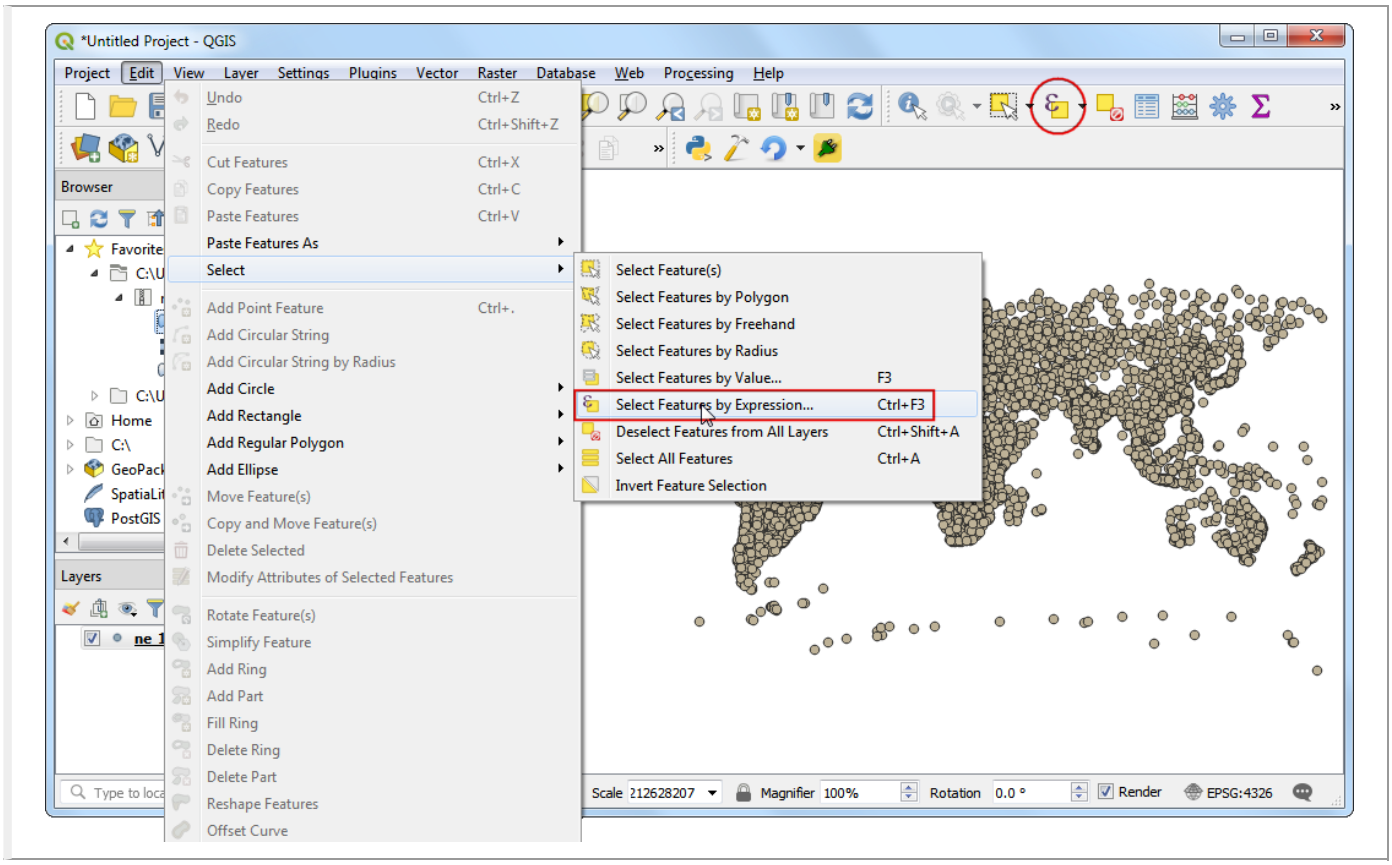
We will use Natural Earth's Populated Places (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-populated-places/) dataset. Download the simple (less columns) dataset (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_populated_places_simple.zip)

Procedure

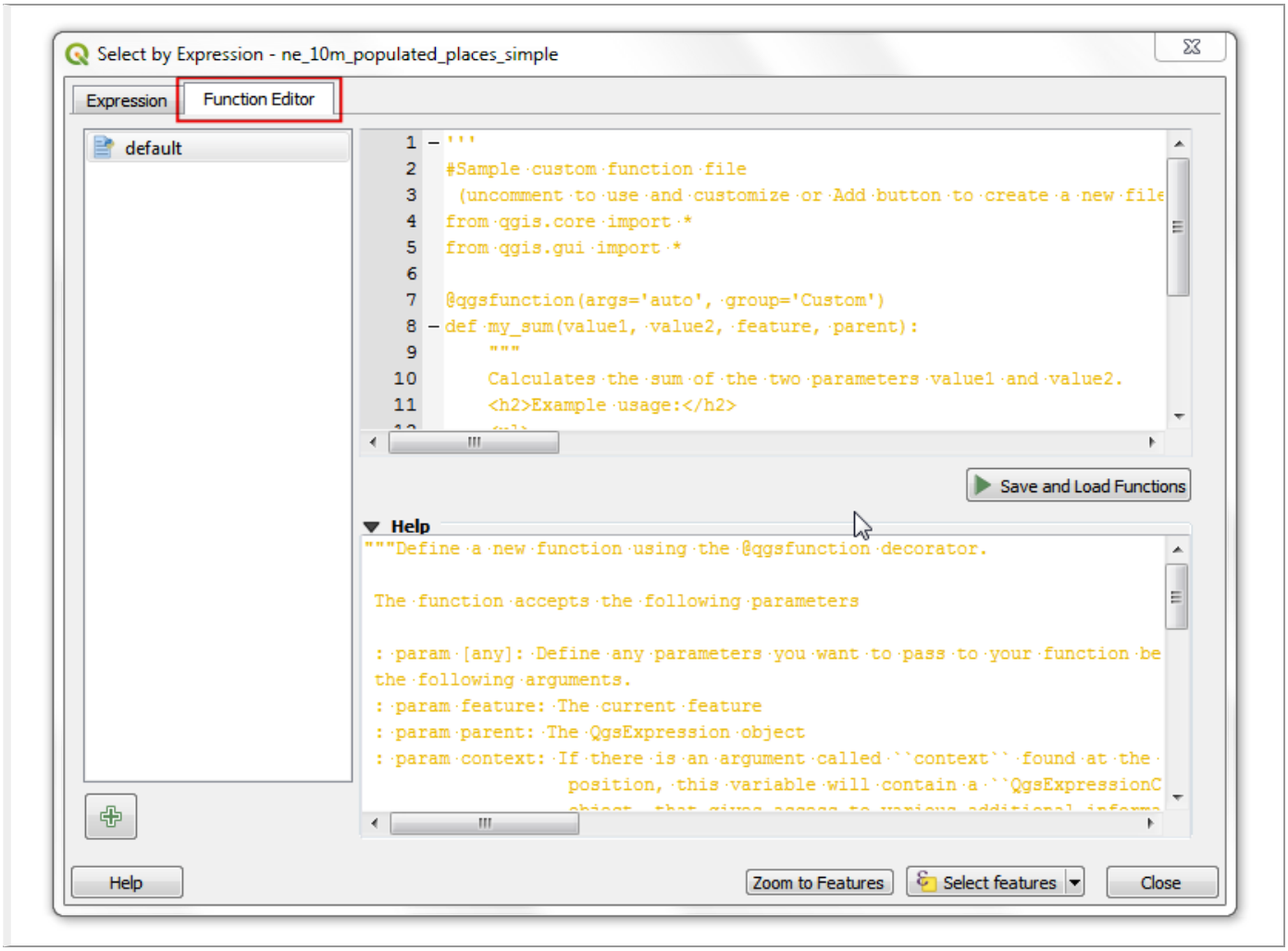
1. Locate the `ne_10m_populated_places_simple.zip` file in the QGIS Browser and expand it. Select the `ne_10m_populated_places_simple.shp` file and drag it to the canvas.



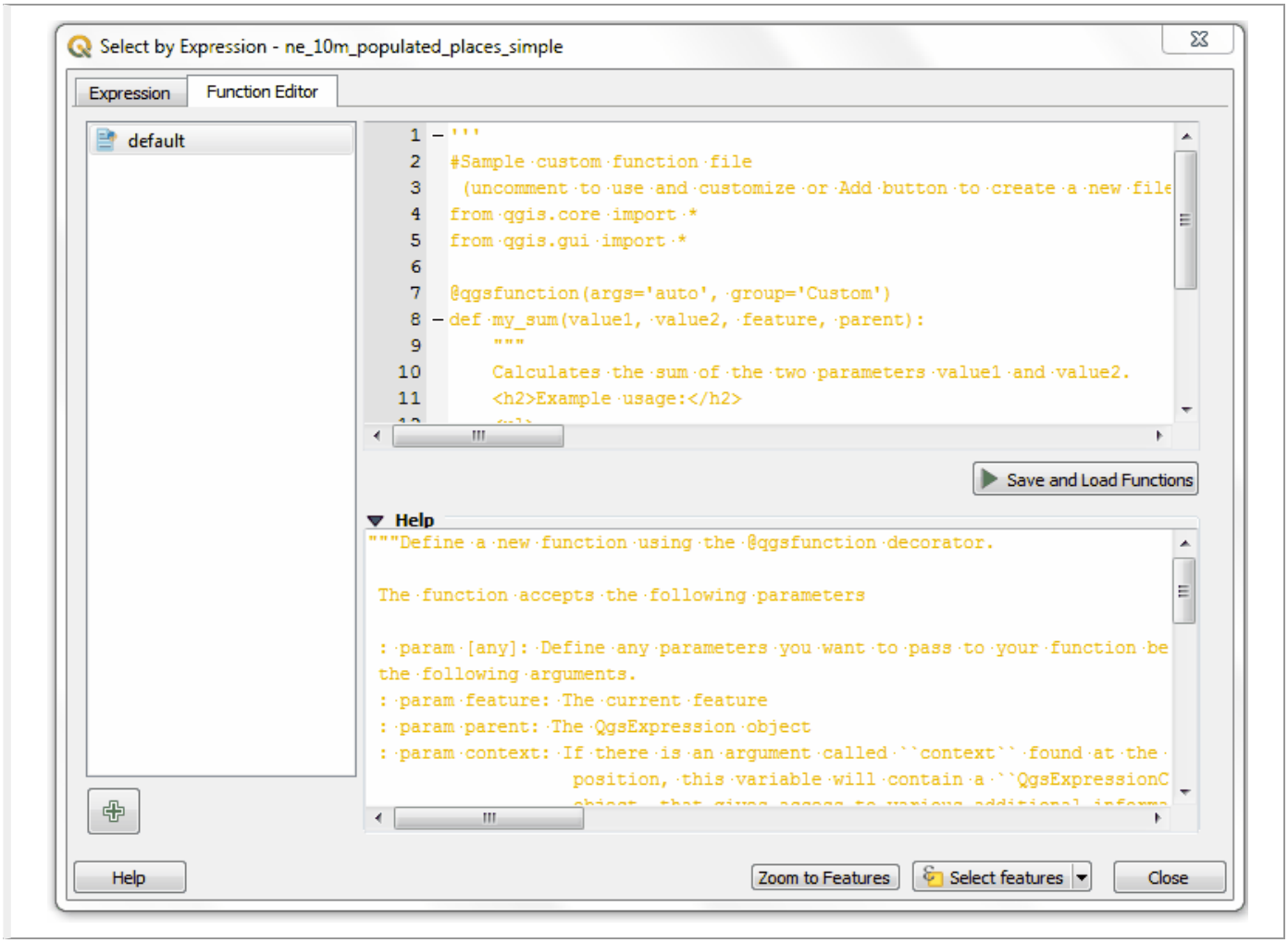
2. Go to Edit ‣ Select ‣ Select Features By Expression... or click the Select features using an expression button on the Attributes Toolbar.



3. In the Select by Expression dialog, switch to the Function Editor tab. Here you can write any PyQGIS code that will be executed by the expression engine.

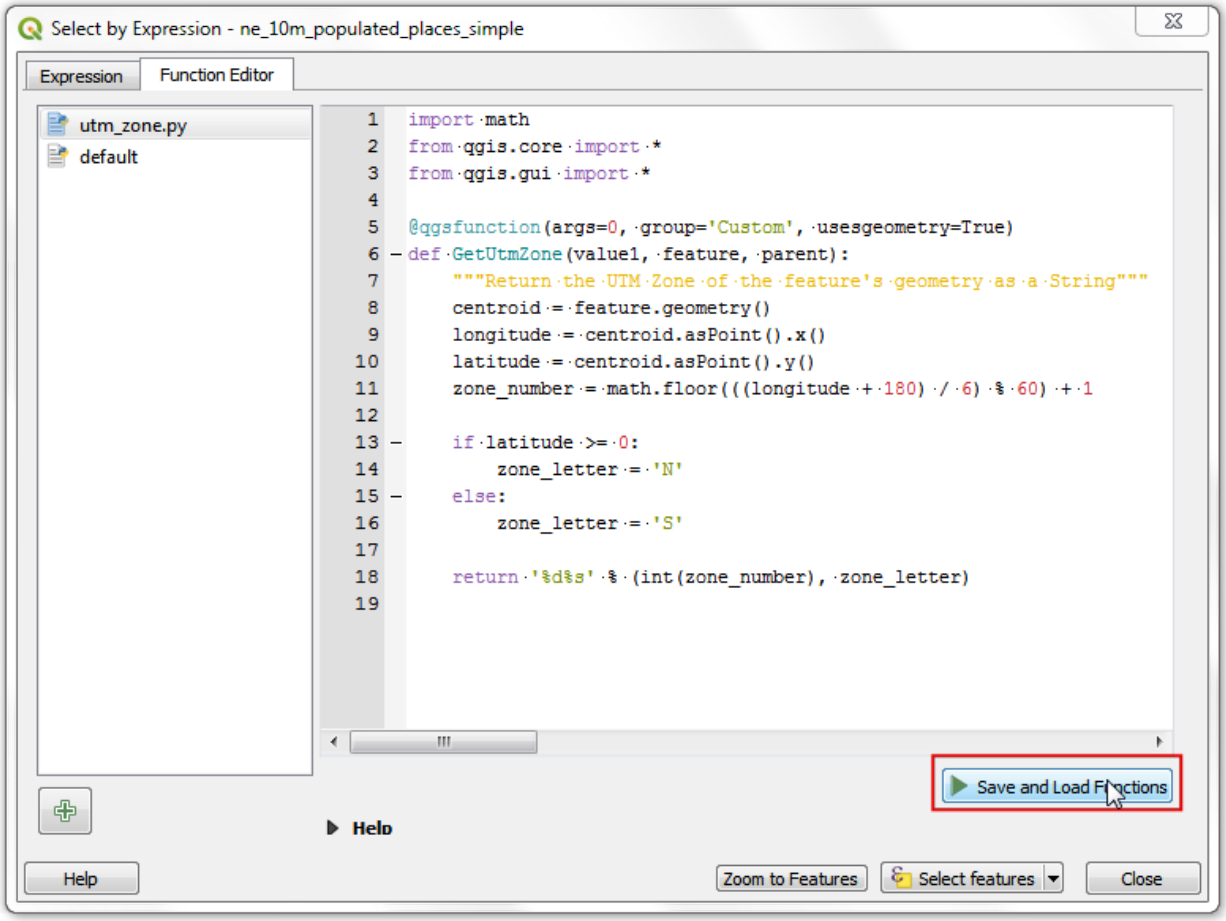


4. We will define a custom function named `GetUtmZone` that will calculate the UTM zone number for each feature. Since custom functions in QGIS work at the feature level. We will use the centroid of the feature's geometry and compute the UTM Zone from the latitude and longitude of the centroid geometry. We will also add a 'N' or 'S' designation to the zone to indicate whether the zone is in the northern or southern hemisphere. Press the + button in the lower left of the screen and type `utm_zones.py` as the file name. You can click the Help label in the bottom panel to close it and expand the code panel.



5. UTM Zones are longitudinal projection zones numbered from 1 to 60. Each UTM zone is 6 degree wide. Here we use a simple mathematical formula to find the appropriate zone for a given longitude value. This formula works for all except a few special UTM zones (https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system#Exceptions). Type the following code into the editor window. When you are finished click Save and Load Functions.

```
import math
from qgis.core import *
from qgis.gui import *

@qgsfunction(args=0, group='Custom', usesgeometry=True)
def GetUtmZone(value1, feature, parent):
    """Return the UTM Zone of the feature's geometry as a String"""
    centroid = feature.geometry()
    longitude = centroid.asPoint().x()
    latitude = centroid.asPoint().y()
    zone_number = math.floor(((longitude + 180) / 6) % 60) + 1

    if latitude >= 0:
        zone_letter = 'N'
    else:
        zone_letter = 'S'

    return '%d%s' % (int(zone_number), zone_letter)
```

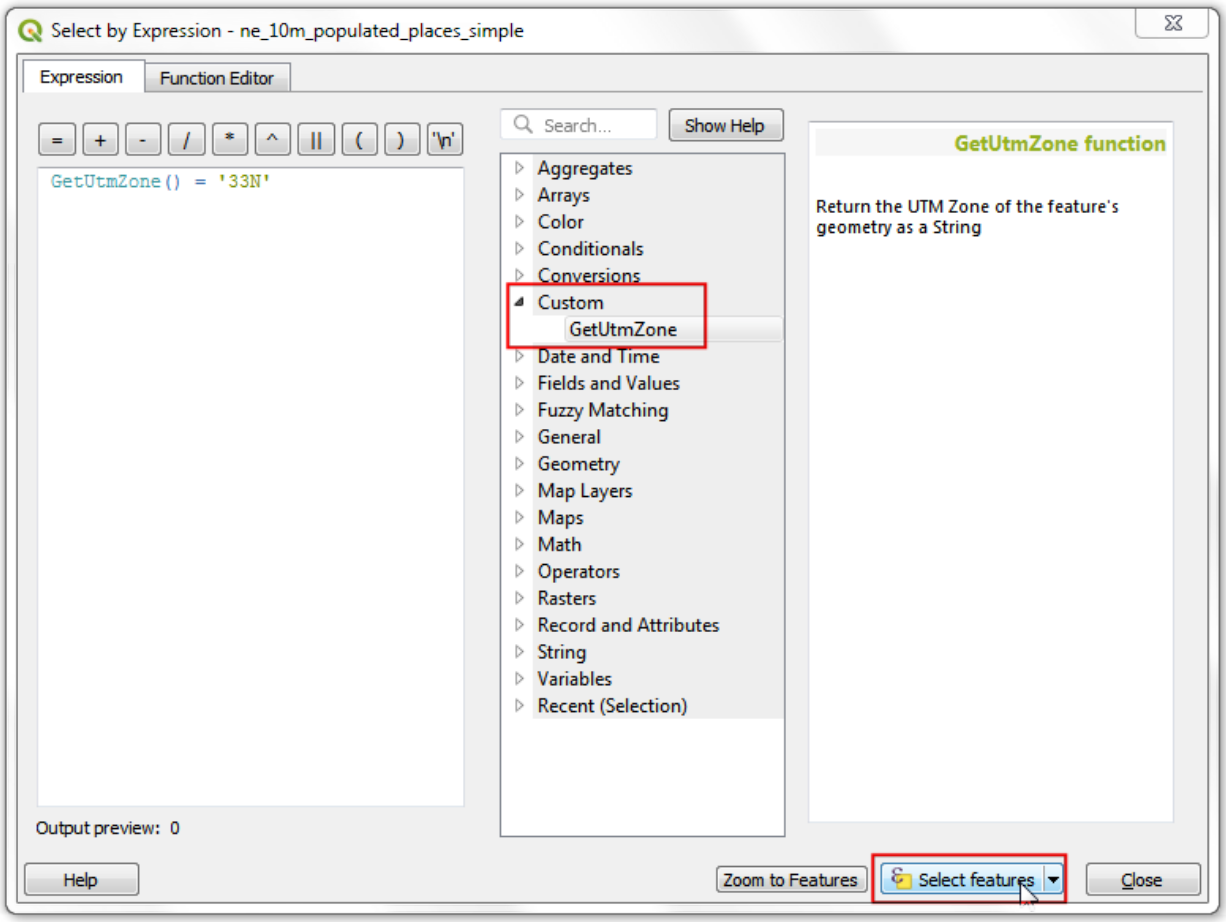


Note

Currently there is no way to delete an expression file from the GUI. If you wish to delete the `utm_zone.py` file, you can go to Settings ▸ User Profiles ▸ Open Active Profile Folder and delete the file from python ▸ expressions.

6. Switch to the Expression tab in the Select by expression dialog. Find and expand the Custom group in the Functions section. You will notice a new custom function `GetUtmZone` in the list. We can now use this function in the expressions just like any other function. Type the following expression in the editor. This expression will select all points that fall in the UTM Zone `33N`. Click Zoom to Features and the map will change, if you click Select Features you should see the points in UTM zone 33N change colour to yellow.

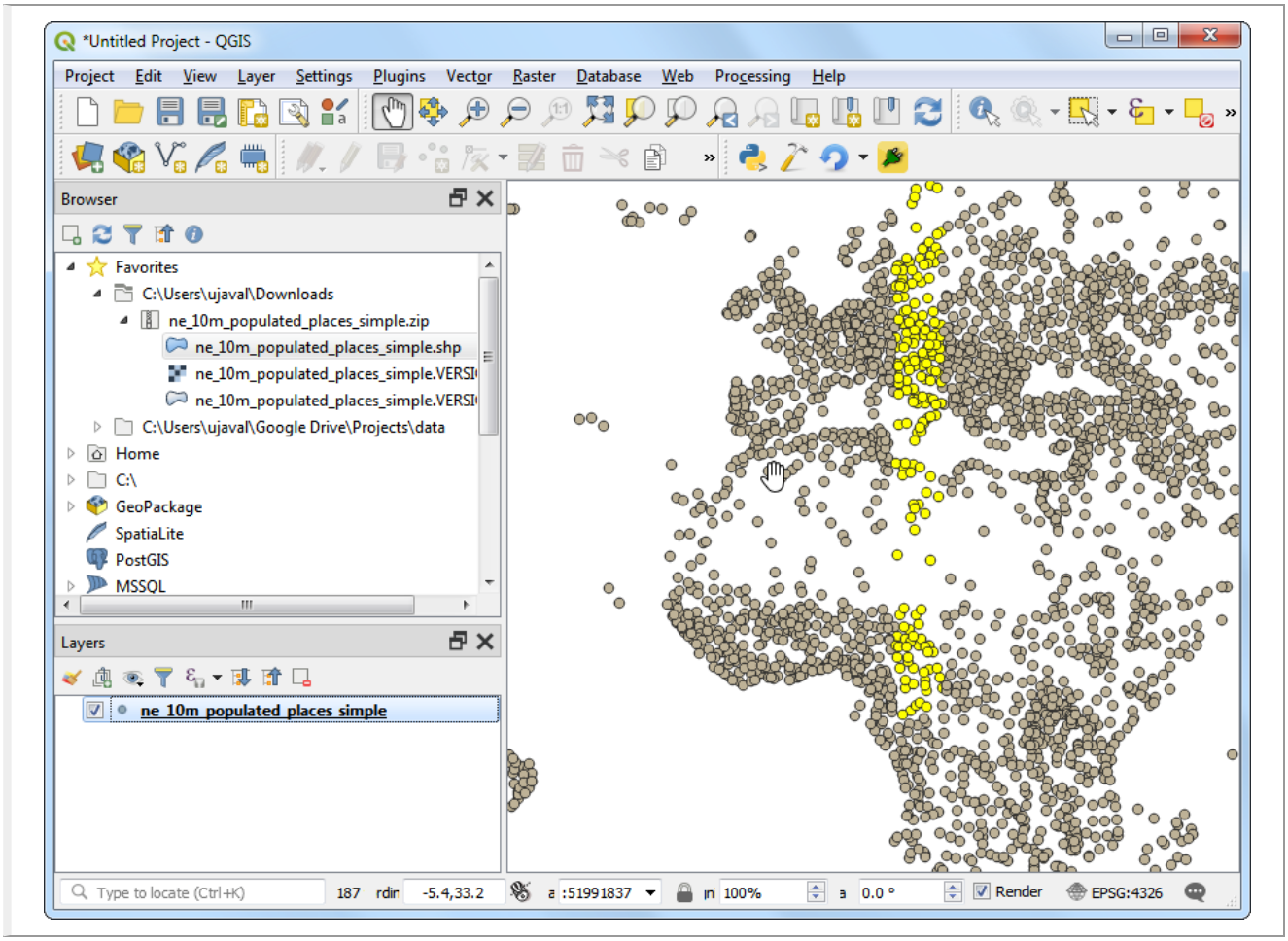
```
GetUtmZone() = '33N'
```



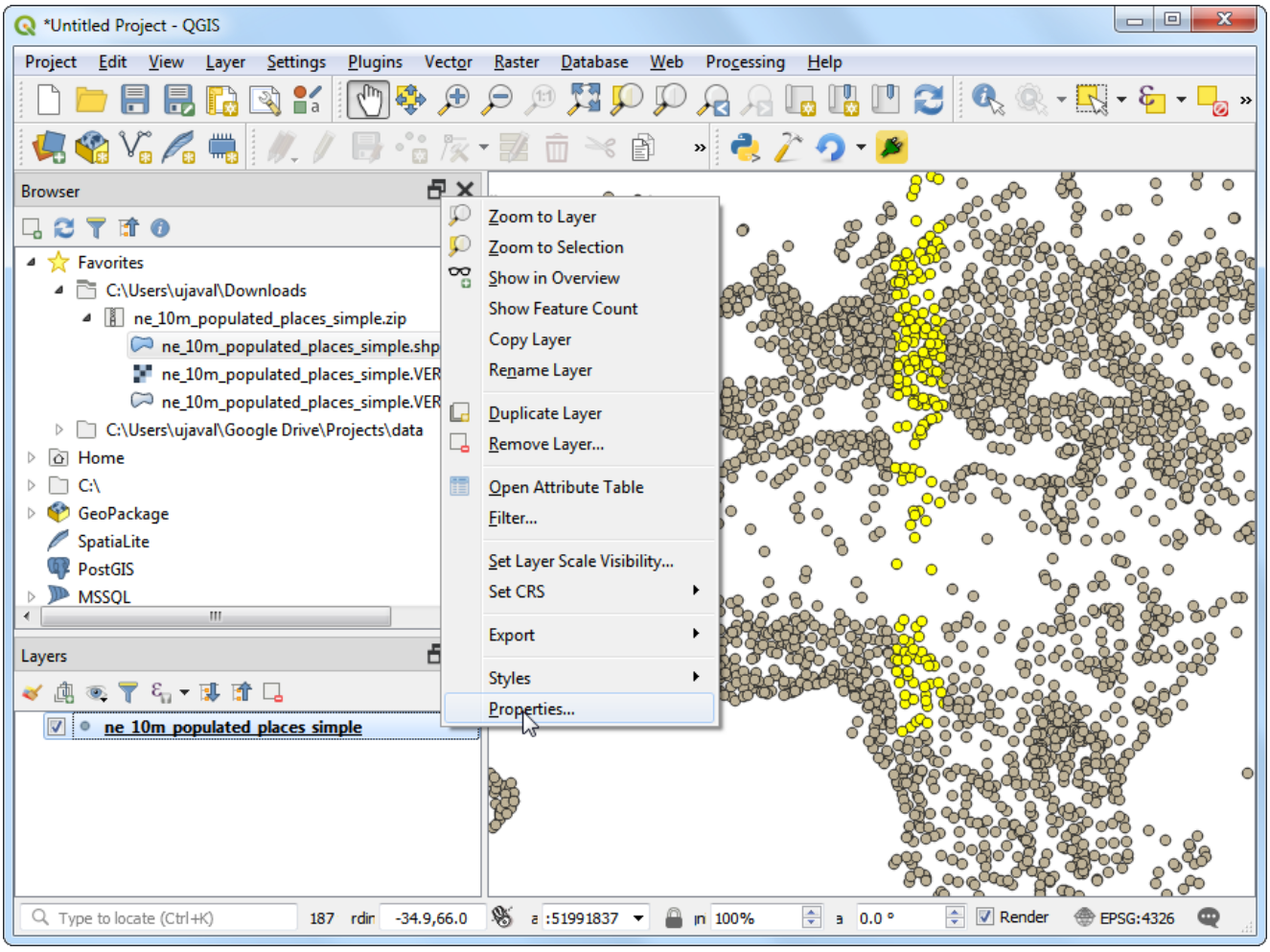
Note

Due to a bug, this feature did not work in earlier versions of QGIS 3. It has been fixed from version 3.4.5 onwards.

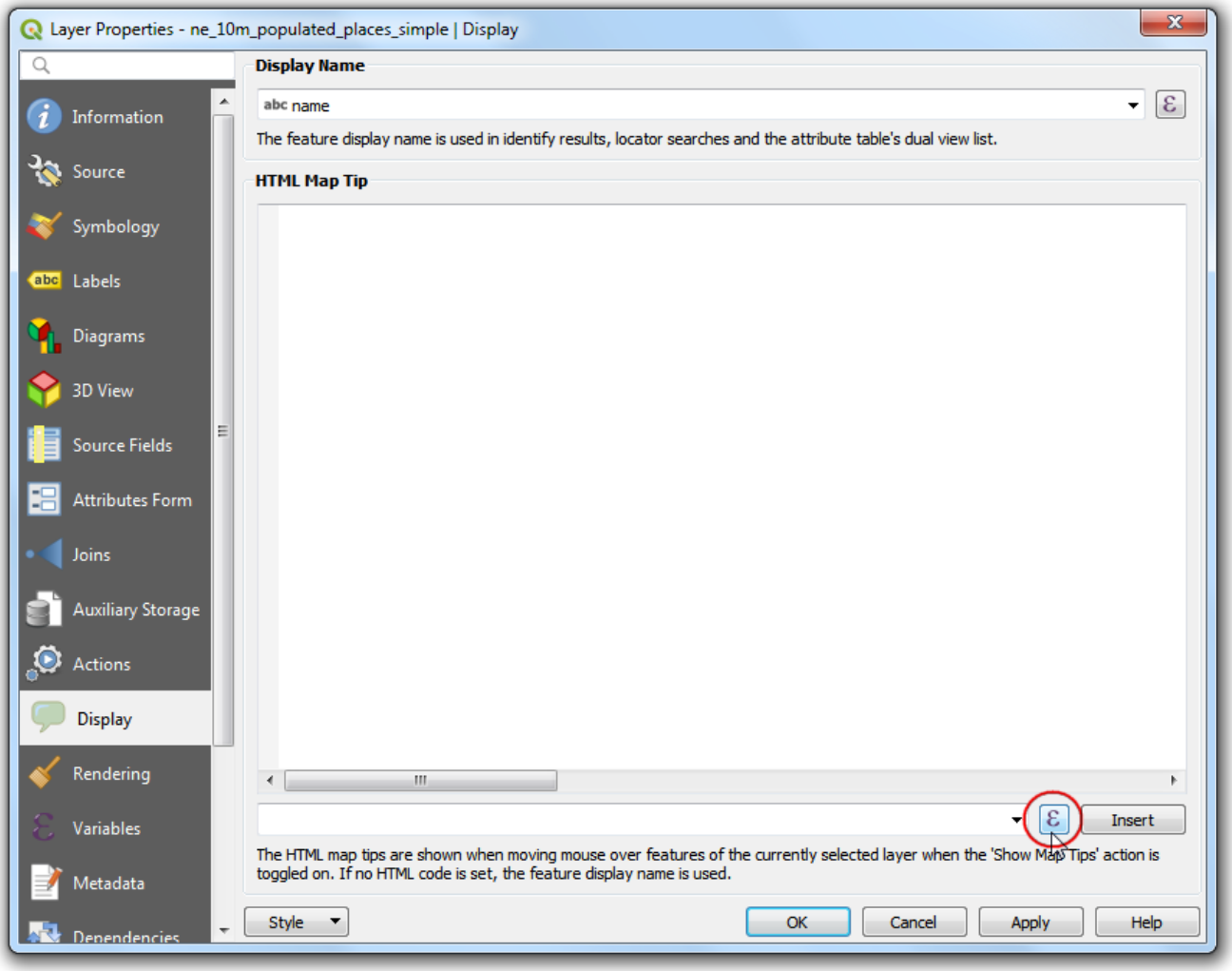
7. Back in the main QGIS window, you should see some points highlighted in yellow. These are the points falling in the UTM Zone we specified in the expression.



8. You saw how we defined and used a custom function to select features by expression. We will now use the same function in another context. One of the hidden gems in QGIS is the **Map Tip** tool. This tool shows user-defined text when you hover over a feature. Right-click the `ne_10m_populated_places_simple` layer and select Properties.

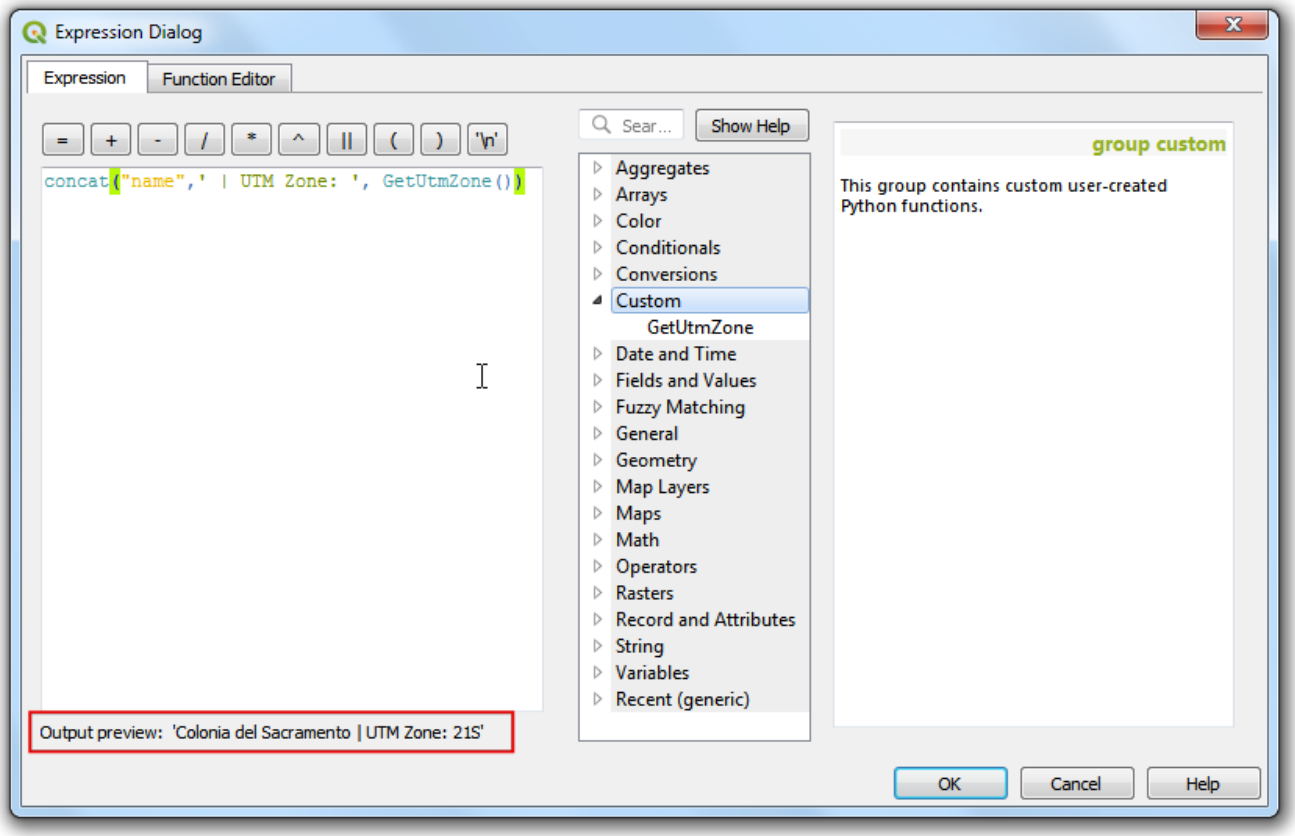


9. Switch to the Display tab. Here you can enter any text that will be displayed when you hover over the features of the layer. Even better, you can use layer field values and expressions to define a much more useful message. Click on the Ɛ button.

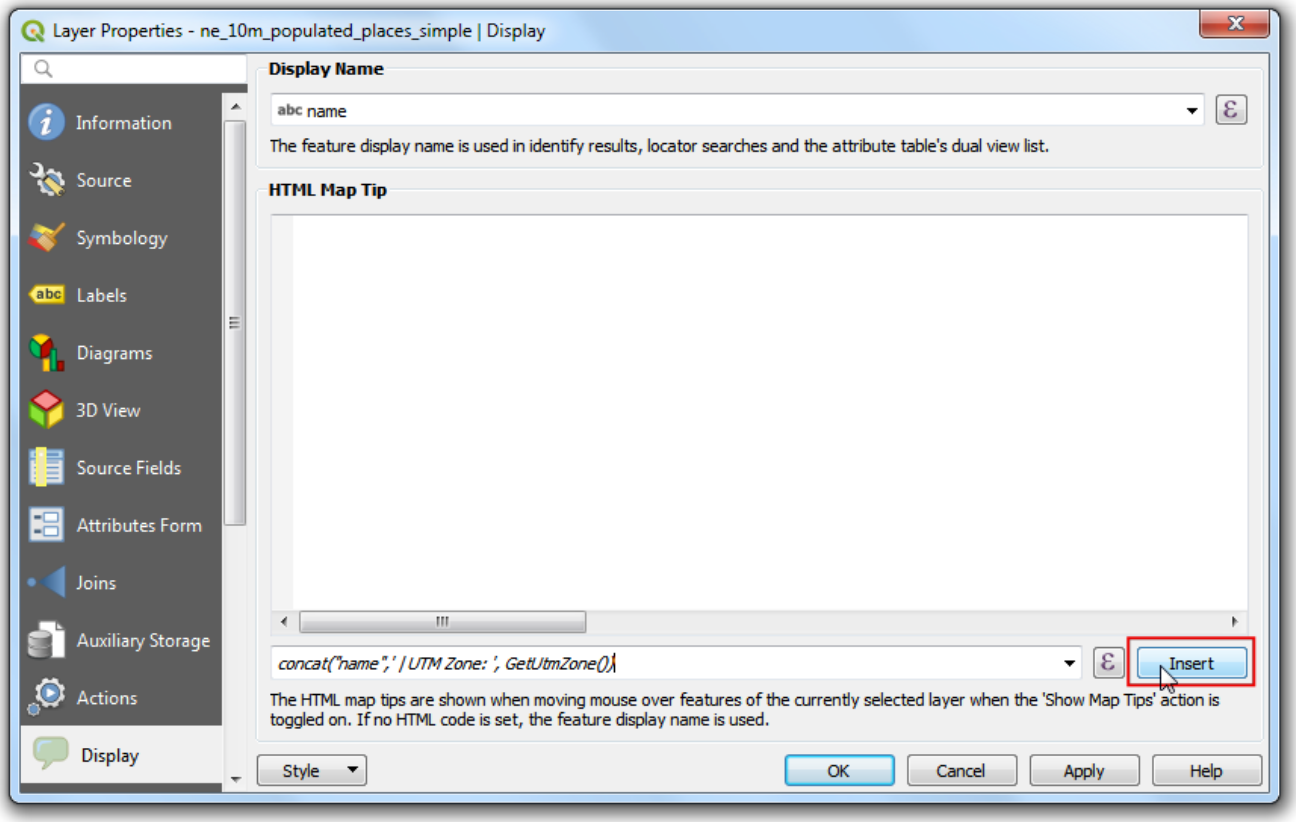


10. You will see the familiar expression editor again. We will use the `concat` function to join the value of the field `name` and the result of our custom function `GetUtmZone`. Enter the following expression and click OK.

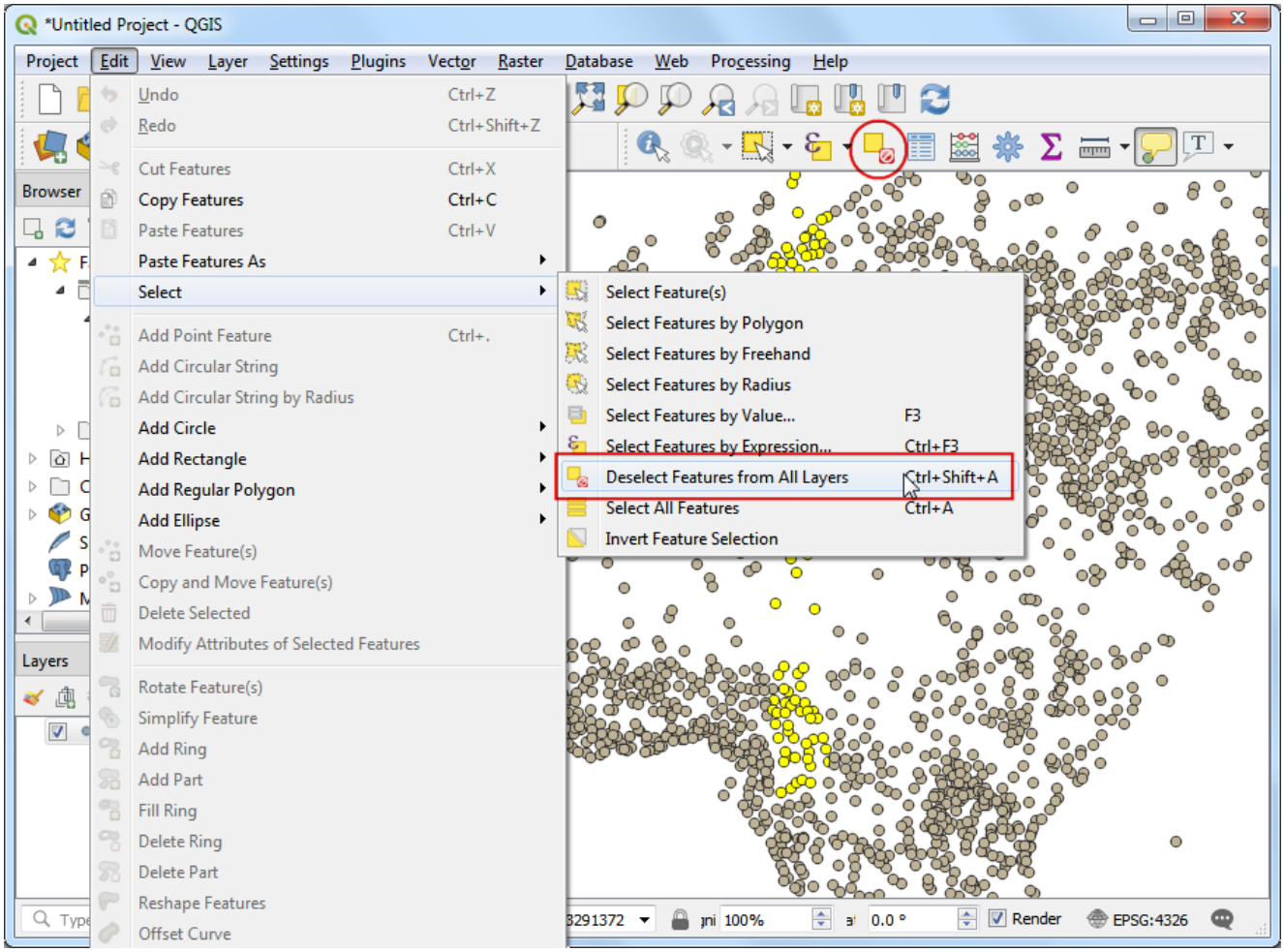
```
concat("name",' | UTM Zone: ', GetUtmZone())
```



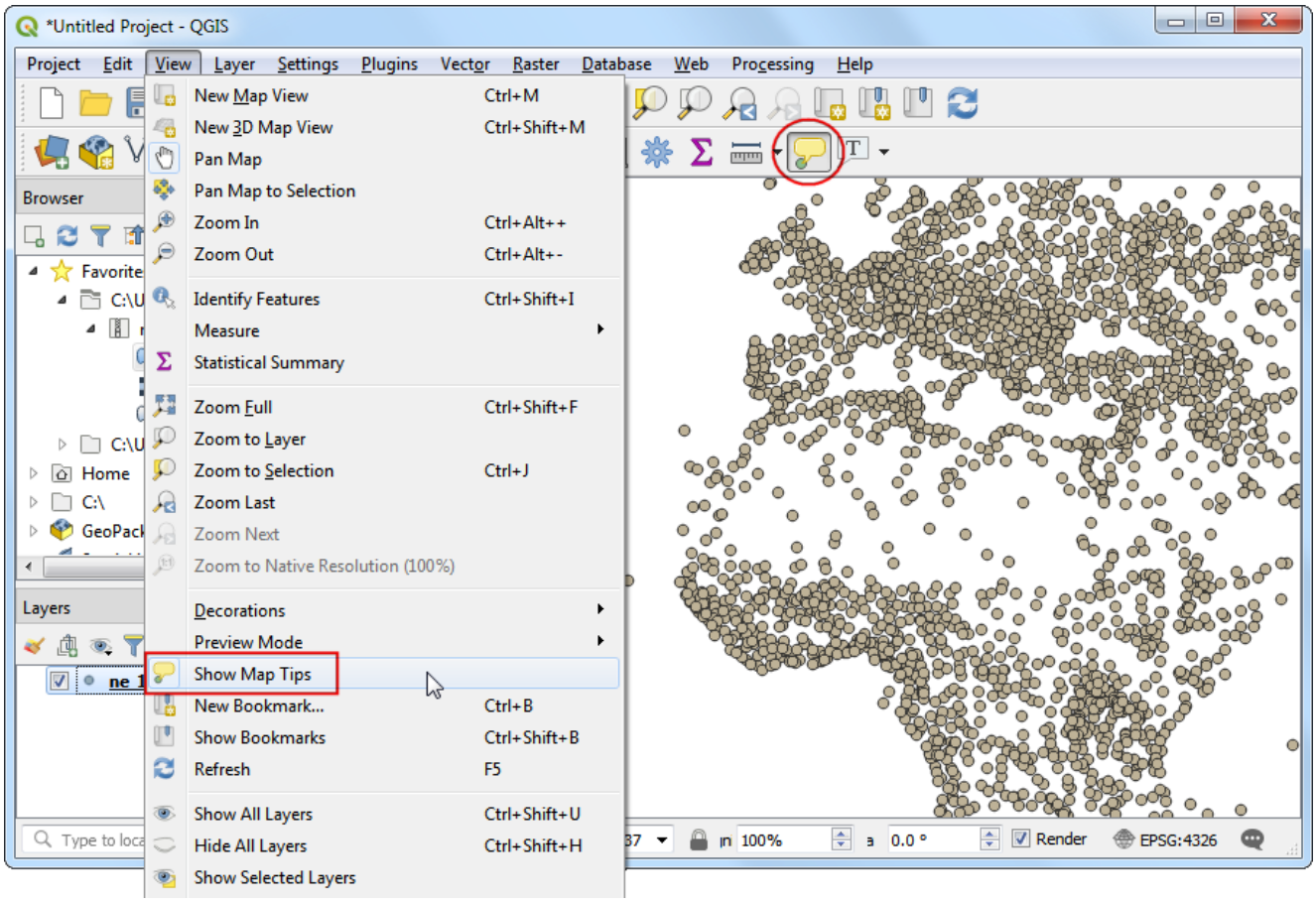
11. You will see the expression entered as the value of the Display text. Click Insert to add it to the HTML box and then press OK.



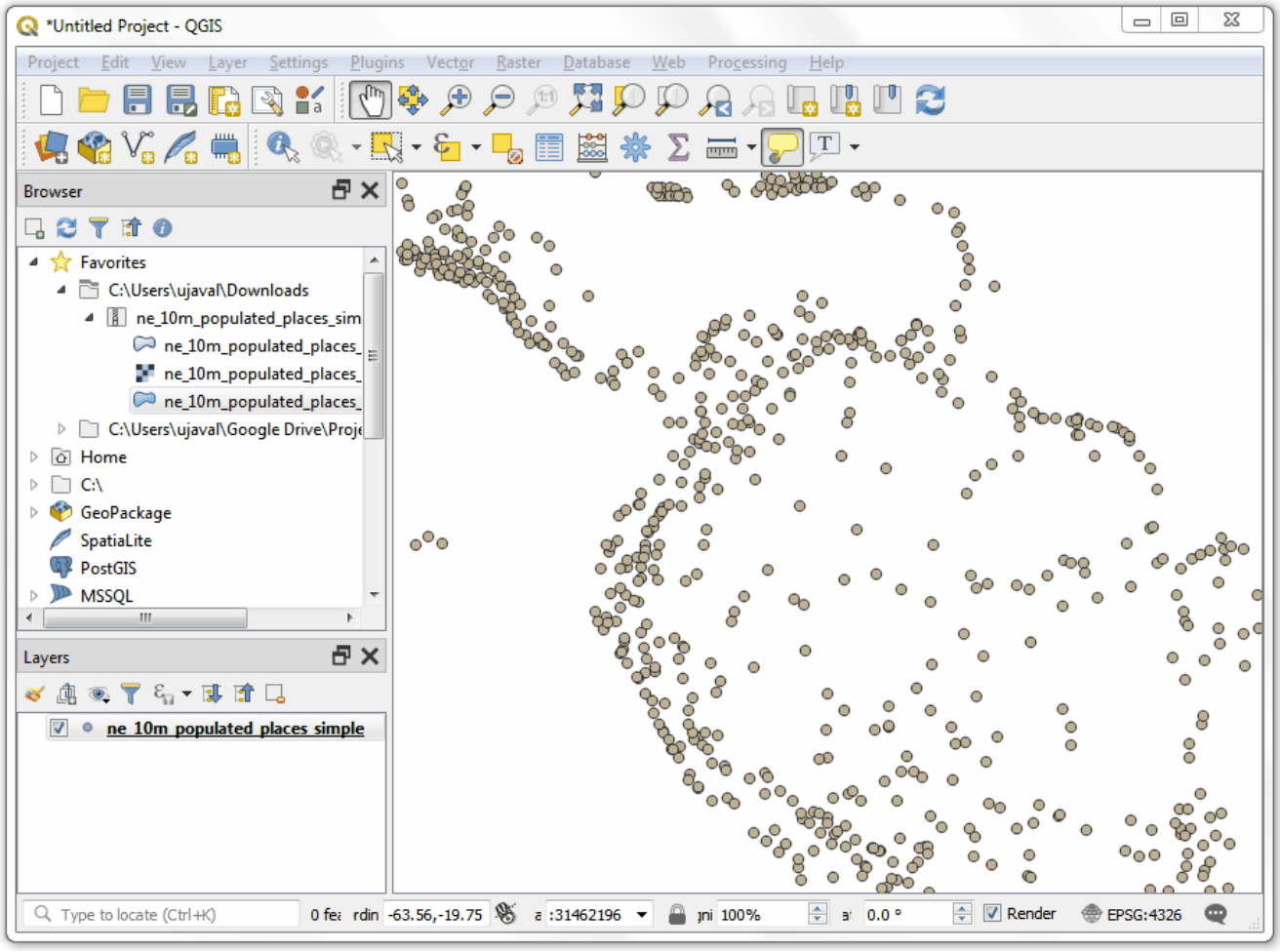
12. Before we proceed, let us de-select the features that were selected in the previous step. Go to Edit ‣ Select ‣ Deselect Features From All Layers or click the Deselect Features From All Layers button on the Attribute Toolbar.



13. Activate the `Map Tips` tool by going to View ▸ Map Tips or clicking the Show Map Tips button on the Attributes Toolbar.



14. Zoom into any area of the map and put your mouse cursor over any feature. You will see the name of the city and corresponding UTM zone displayed as the map tip.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Writing Python Scripts for Processing Framework (QGIS3)

One can write standalone pyqgis scripts that can be run via the Python Console in QGIS. With a few tweaks, you can make your standalone scripts run via the Processing Framework. This has several advantages. First, taking user input and writing output files is far easier because Processing Framework offers standardized user interface for these. Second, having your script in the Processing Toolbox also allows it to be part of any Processing Model or be run as a Batch job with multiple inputs. This tutorial will show how to write a custom python script that can be part of the Processing Framework in QGIS.

Note

The Processing API was completely overhauled in QGIS3. Please refer to this guide (https://raw.githubusercontent.com/qgis/QGIS/master/doc/porting_processing.dox) for best practices and tips.

Overview of the task

Our script will perform a dissolve operation based on a field chosen by the user. It will also sum up values of another field for the dissolved features. In the example, we will dissolve a world shapefile based on a `CONTINENT` attribute and sum up `POP_EST` field to calculate total population in the dissolved region.

Get the data

We will use the Admin 0 - Countries (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-admin-0-countries/) dataset from Natural Earth.

Download the Admin 0 - countries shapefile. (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_admin_0_countries.zip).

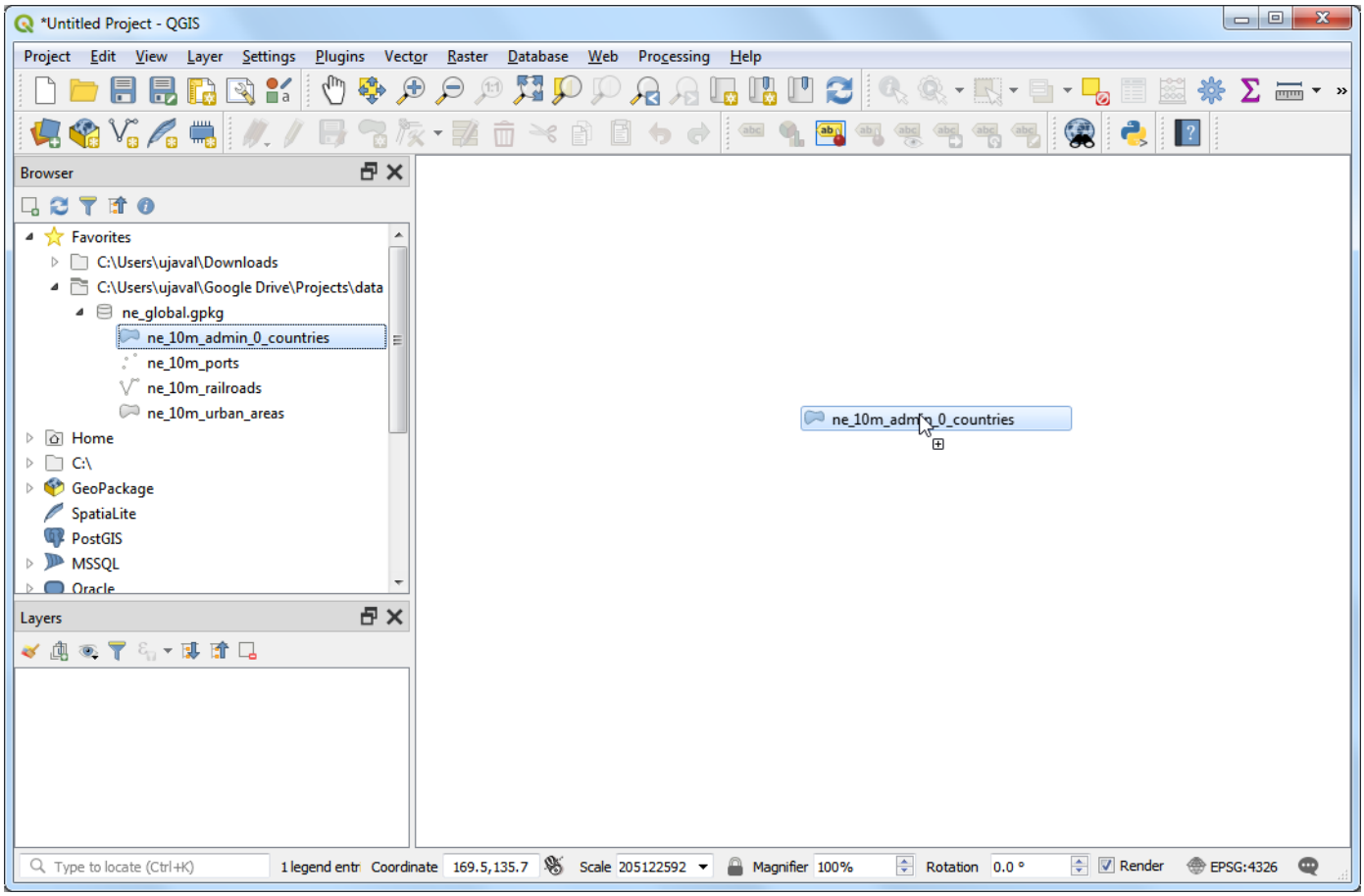
Data Source [NATURALEARTH] (../credits.html#naturalearth)

For convenience, you may directly download a geopackage containing the above layer from below:

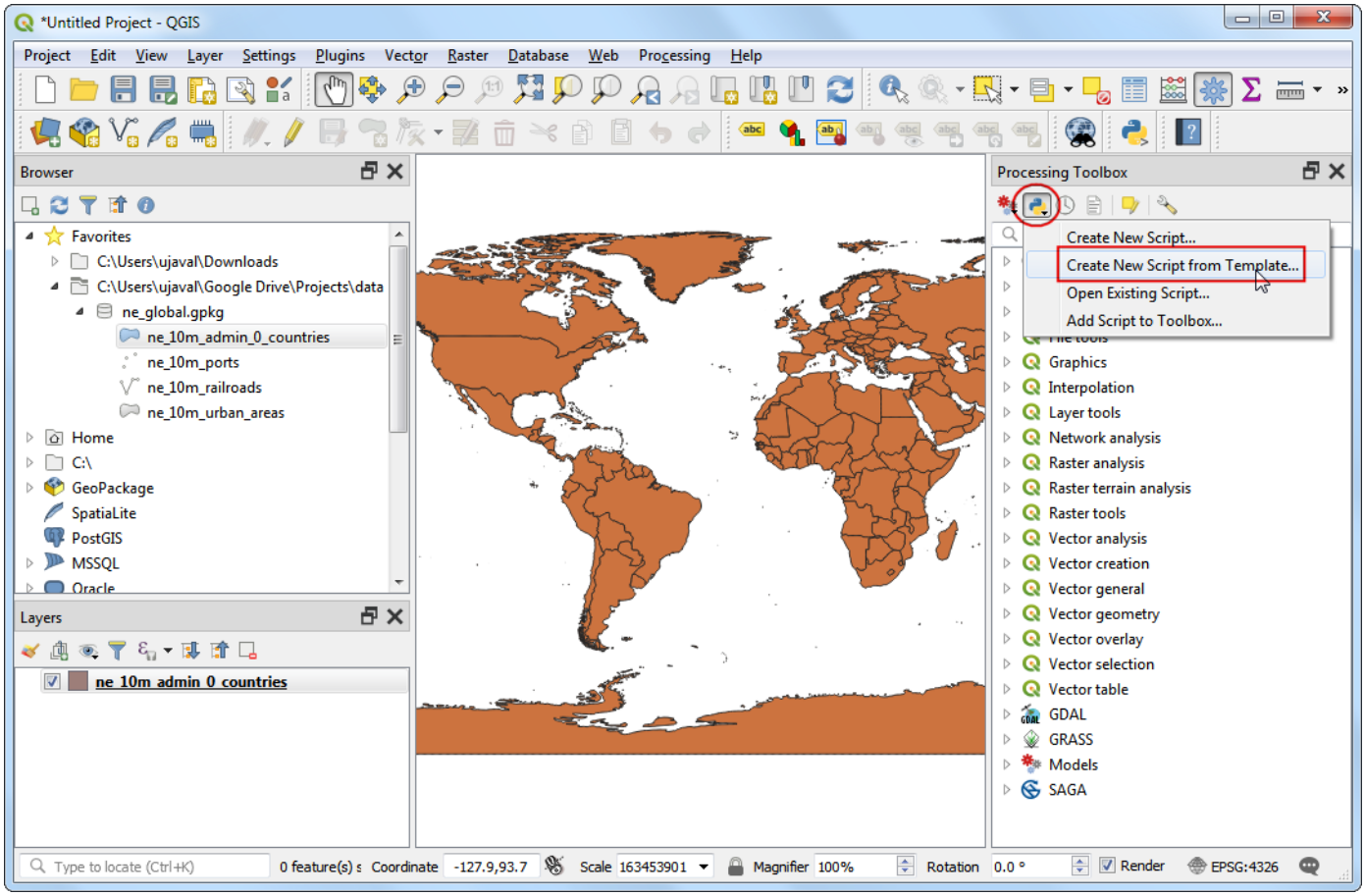
ne_global.gpkg (http://www.qgistutorials.com/downloads/ne_global.gpkg)

Procedure

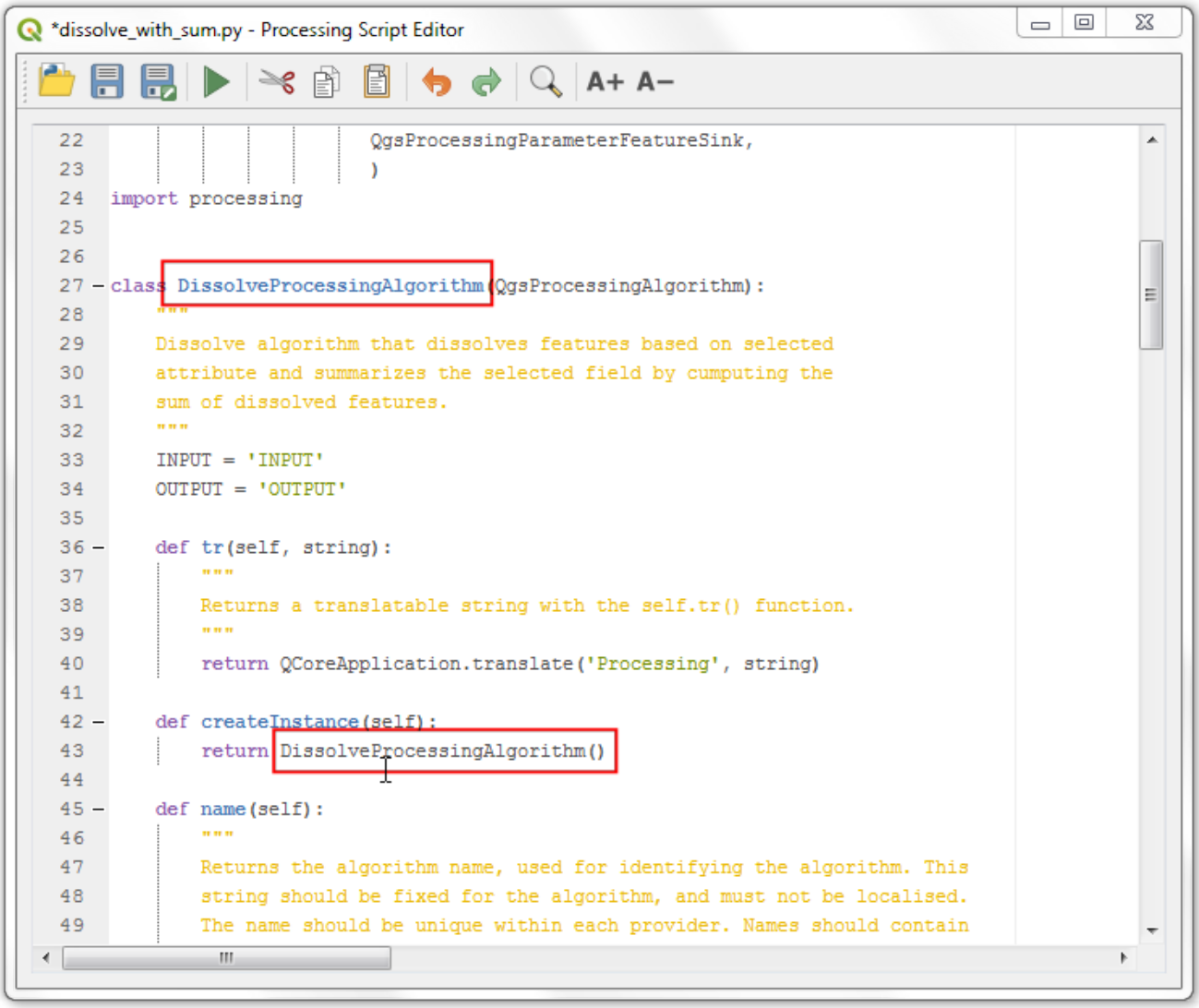
1. In the QGIS Browser Panel, locate the directory where you saved your downloaded data. Expand the `zip` or the `gpkg` entry and select the `ne_10m_admin_0_countries` layer. Drag the layer to the canvas.



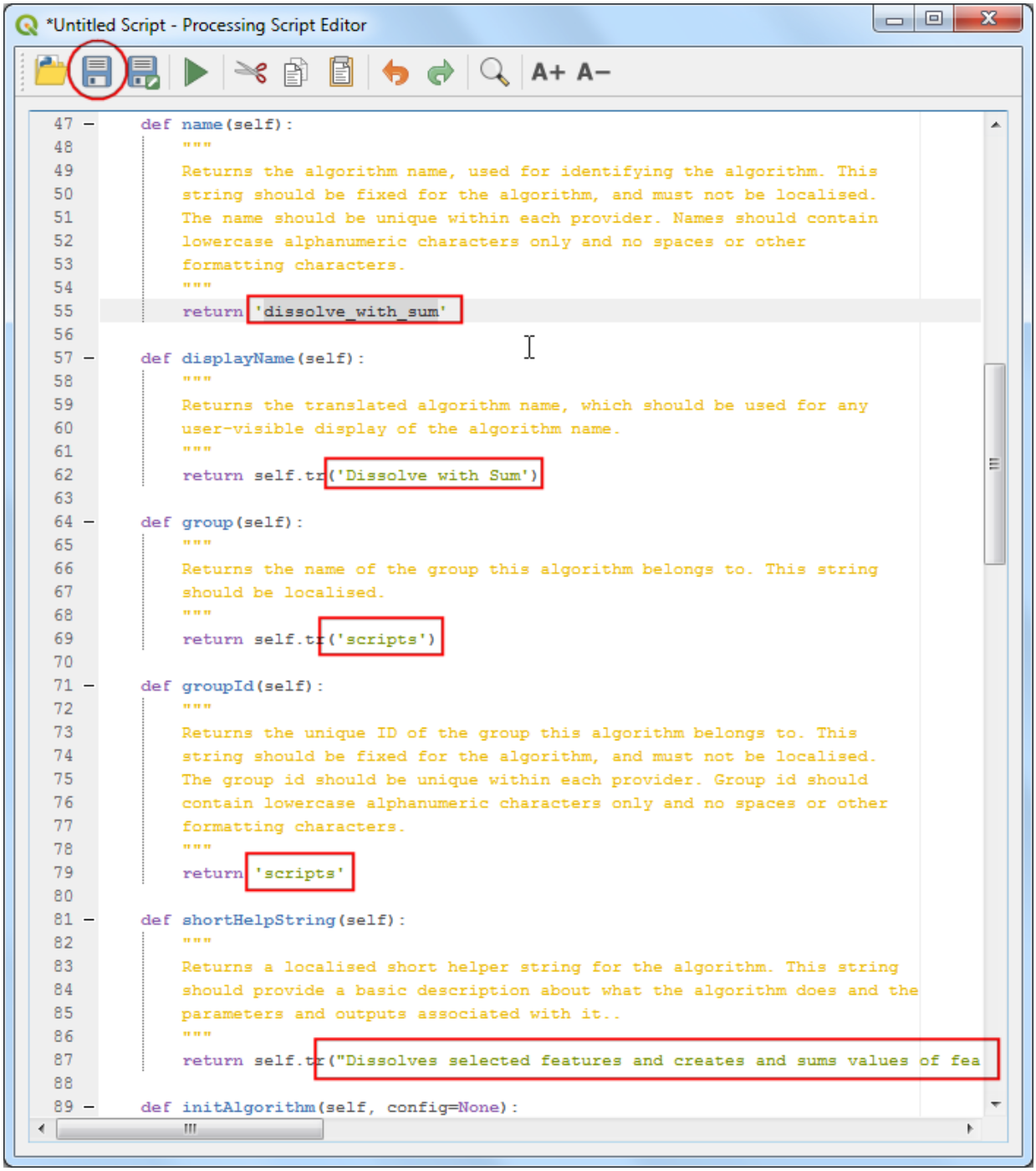
2. Go to Processing ‣ Toolbox. Click the Scripts button in the toolbar and select Create New Script from Template.



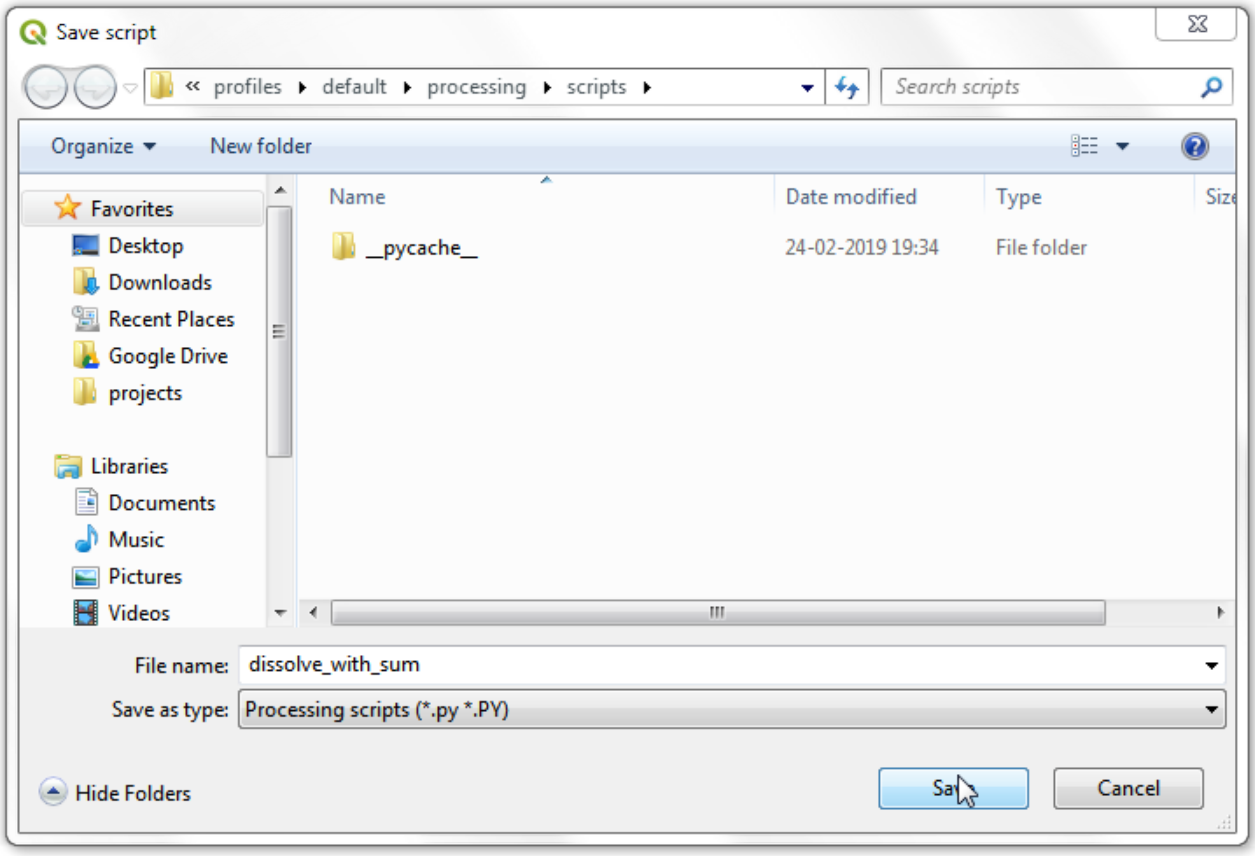
3. The template contains all the boilerplate code that is required for the Processing Framework to recognize it as a Processing script, and manage inputs/outputs. Let's start customizing the example template to our needs. First change the class name from `ExampleProcessingAlgorithm` to `DissolveProcessingAlgorithm`. This name also needs to be updated in the `createInstance` method. Add a docstring to the class that explains what the algorithm does.



4. As you scroll down, you will see methods that assign name, group, description etc. to the script. Change the return values for *name* method to be `dissolve_with_sum`, *displayName* method to `Dissolve with Sum`, *group* method and *groupId* method to `scripts`. Change the return value of *shortHelpString* method to a description that will appear to the user. Click the Save button.

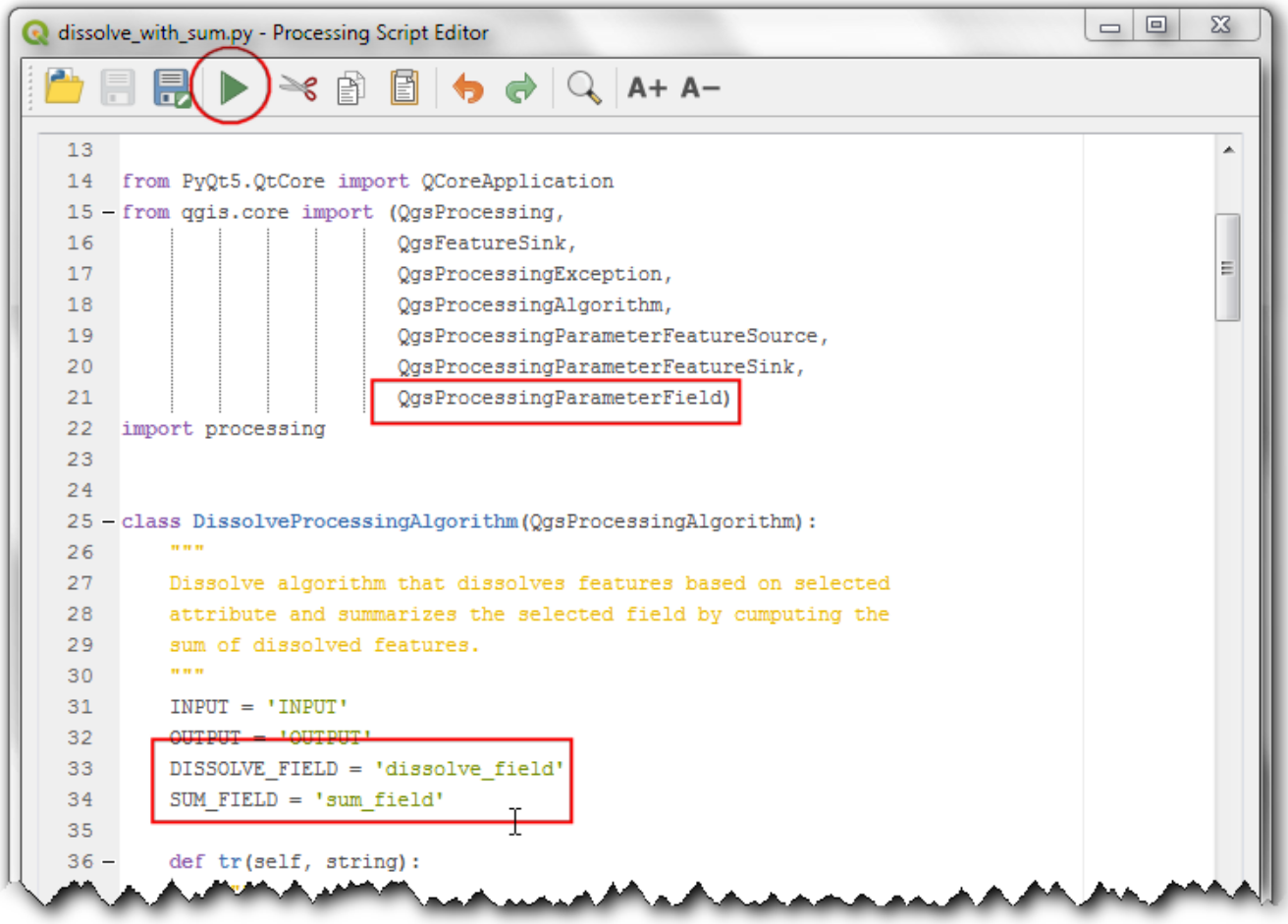


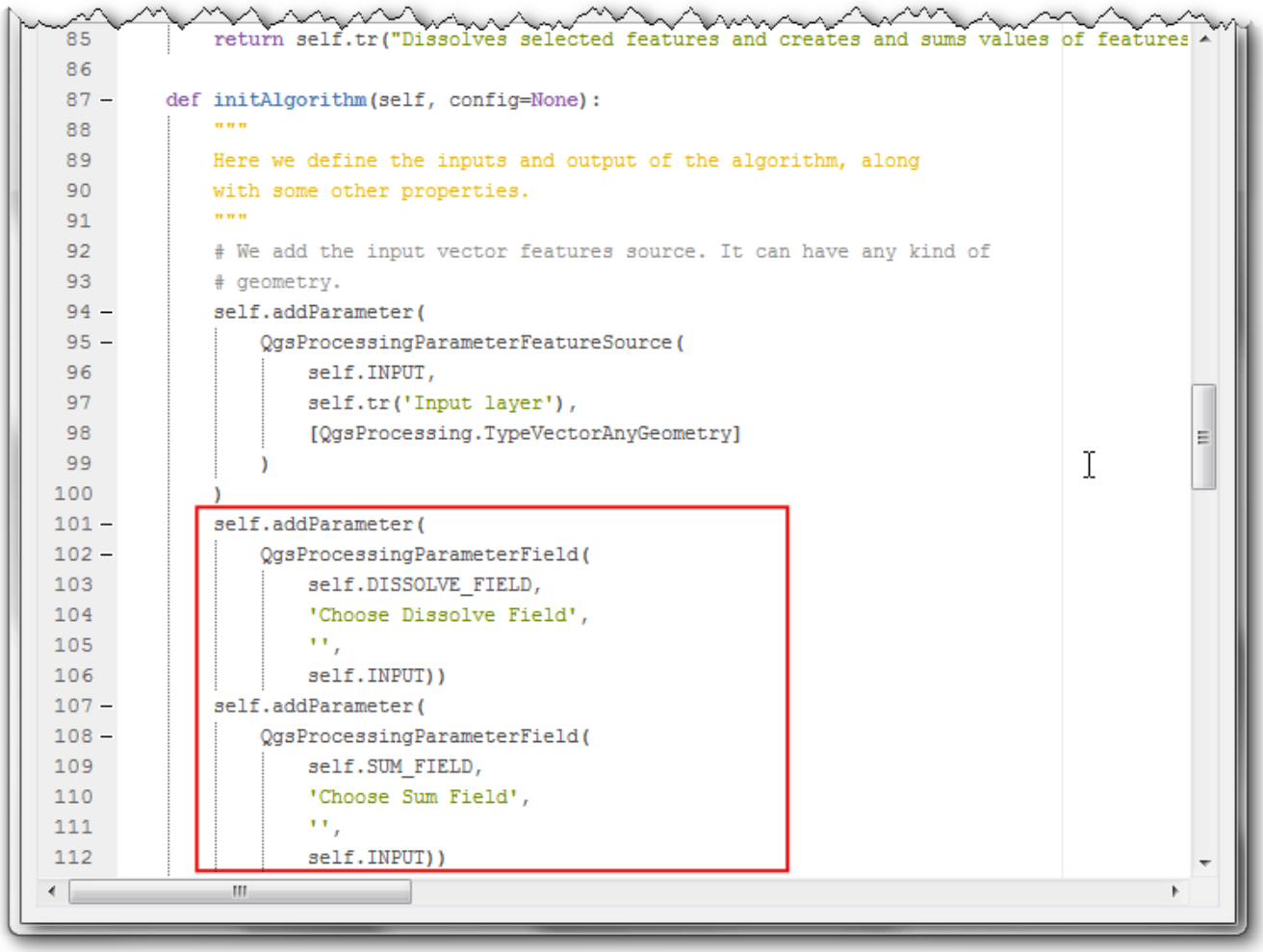
5. Name the script `dissolve_with_sum` and save it at the default location under profiles ‣ default ‣ processing ‣ scripts folder.



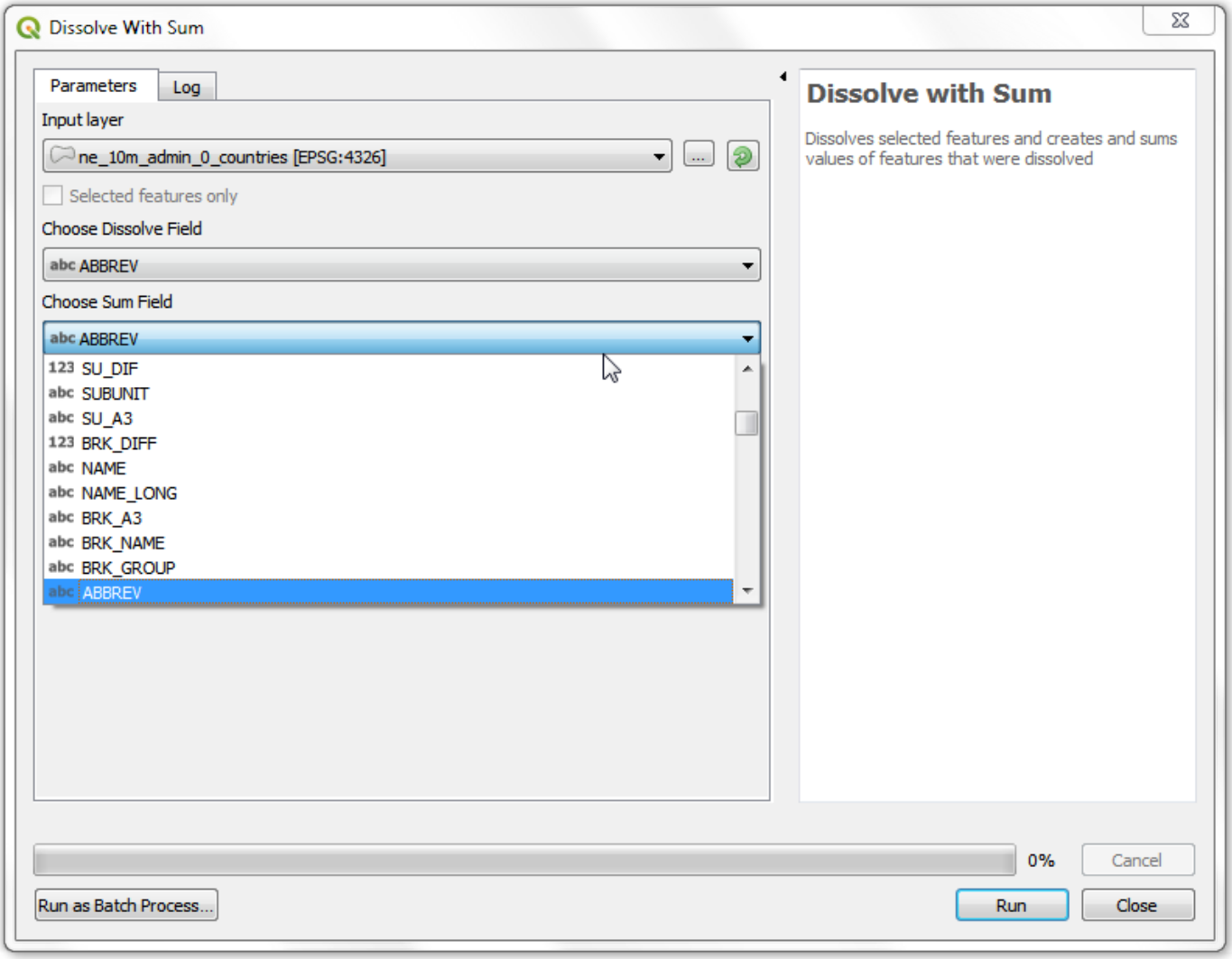
6. Now we will define the inputs for the script. The template already contains a definition of an `INPUT` Vector Layer and a `OUTPUT` layer. We will add 2 new inputs which allows the user to select a `DISSOLVE_FIELD` and a `SUM_FIELD`. Add a new import at the top and the following code in the `initAlgorithm` method. Click the Run button to preview the changes.

```
from qgis.core import QgsProcessingParameterField

self.addParameter(
        QgsProcessingParameterField(
                self.DISSOLVE_FIELD,
                'Choose Dissolve Field',
                '',
                self.INPUT))

self.addParameter(
        QgsProcessingParameterField(
                self.SUM_FIELD,
                'Choose Sum Field',
                '',
                self.INPUT))
```


dissolve_with_sum.py - Processing Script Editor
A+ A−
from PyQt5.QtCore import QCoreApplication
from qgis.core import (QgsProcessing,
QgsFeatureSink,
QgsProcessingException,
QgsProcessingAlgorithm,
QgsProcessingParameterFeatureSource,
QgsProcessingParameterFeatureSink,
QgsProcessingParameterField)
import processing
class DissolveProcessingAlgorithm(QgsProcessingAlgorithm):
"""
Dissolve algorithm that dissolves features based on selected
attribute and summarizes the selected field by cumputing the
sum of dissolved features.
"""
INPUT = 'INPUT'
OUTPUT = 'OUTPUT'
DISSOLVE_FIELD = 'dissolve_field'
SUM_FIELD = 'sum_field'
def tr(self, string):

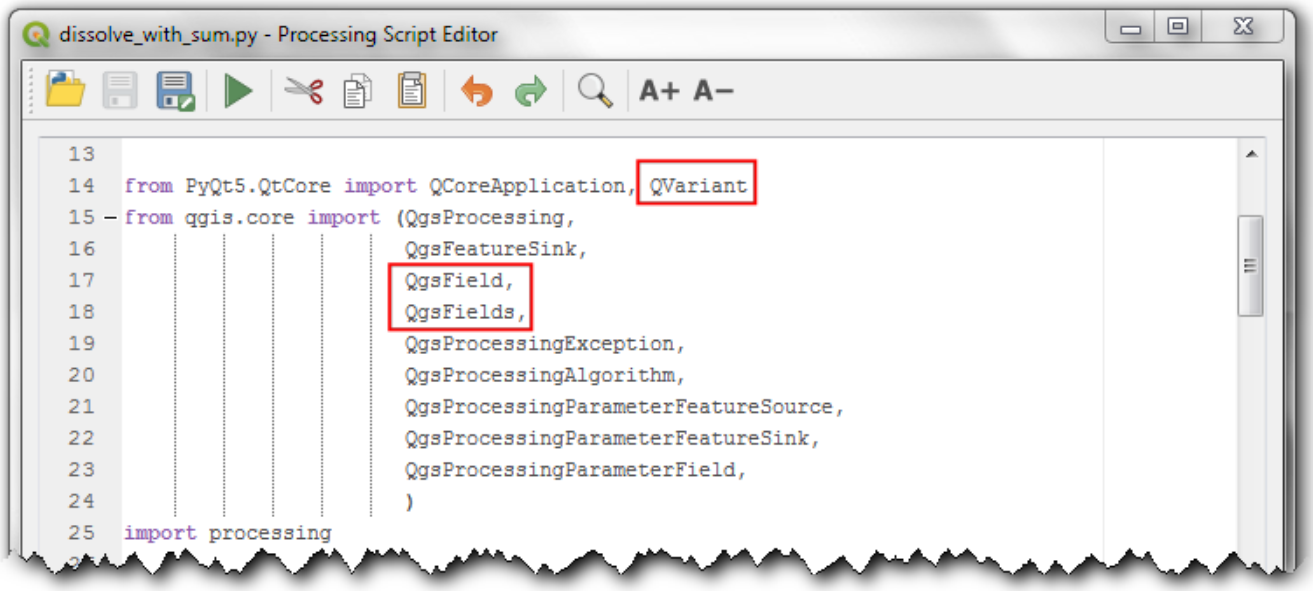

return self.tr("Dissolves selected features and creates and sums values of features
def initAlgorithm(self, config=None):
"""
Here we define the inputs and output of the algorithm, along
with some other properties.
"""
We add the input vector features source. It can have any kind of
geometry.
self.addParameter(
QgsProcessingParameterFeatureSource(
self.INPUT,
self.tr('Input layer'),
[QgsProcessing.TypeVectorAnyGeometry]
)
)
self.addParameter(
QgsProcessingParameterField(
self.DISSOLVE_FIELD,
'Choose Dissolve Field',
'',
self.INPUT))
self.addParameter(
QgsProcessingParameterField(
self.SUM_FIELD,
'Choose Sum Field',
'',
self.INPUT))

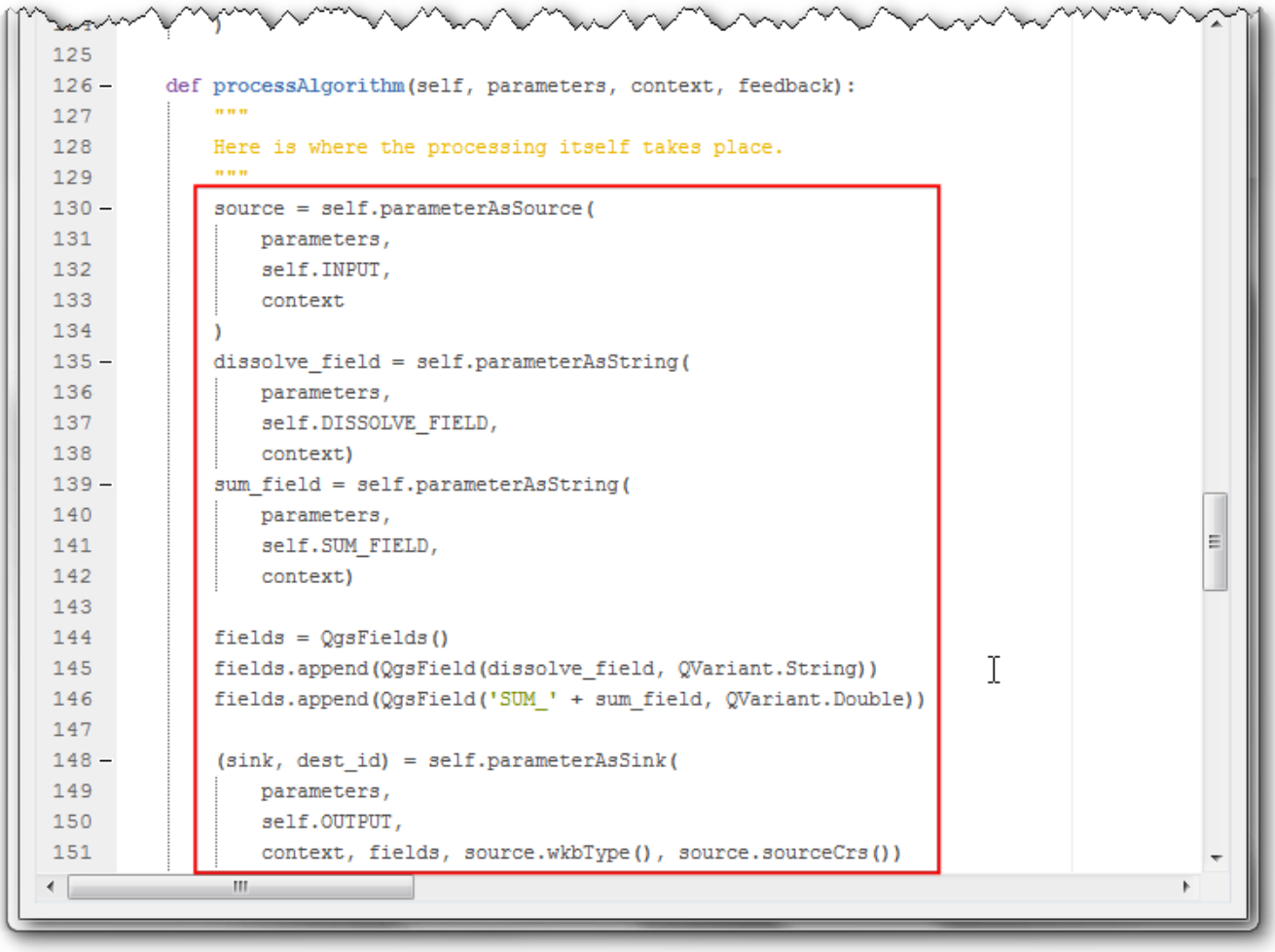
7. You will see a Dissolve with Sum dialog with our newly defined inputs. Select the `ne_10m_admin_0_countries` layer as Input layer`. As both Dissolve Field and Sum Fields are filtered based on the input layer, they will be pre-populated with existing fields from the input layer. Click the Close button.



8. Now we define our custom logic for processing data in the `processAlgorithm` method. This method gets passed a dictionary called `parameters` . It contains the inputs that the user has selected. There are helper methods that allow you to take these inputs and create appropriate objects. We first get our inputs using `parameterAsSource` and `parameterAsString` methods. Next we want to create a feature sink where we will write the output. QGIS3 has a new class called `QgsFeatureSink` which is the preferred way to create objects that can accept new features. The output needs only 2 fields - one for the value of dissolved field and another for the sum of the selected field.

```
from PyQt5.QtCore import QVariant
from qgis.core import QgsField, QgsFields

source = self.parameterAsSource(
        parameters,
        self.INPUT,
        context)
dissolve_field = self.parameterAsString(
        parameters,
        self.DISSOLVE_FIELD,
        context)
sum_field = self.parameterAsString(
        parameters,
        self.SUM_FIELD,
        context)

fields = QgsFields()
fields.append(QgsField(dissolve_field, QVariant.String))
fields.append(QgsField('SUM_' + sum_field, QVariant.Double))

(sink, dest_id) = self.parameterAsSink(
        parameters,
        self.OUTPUT,
        context, fields, source.wkbType(), source.sourceCrs())
```

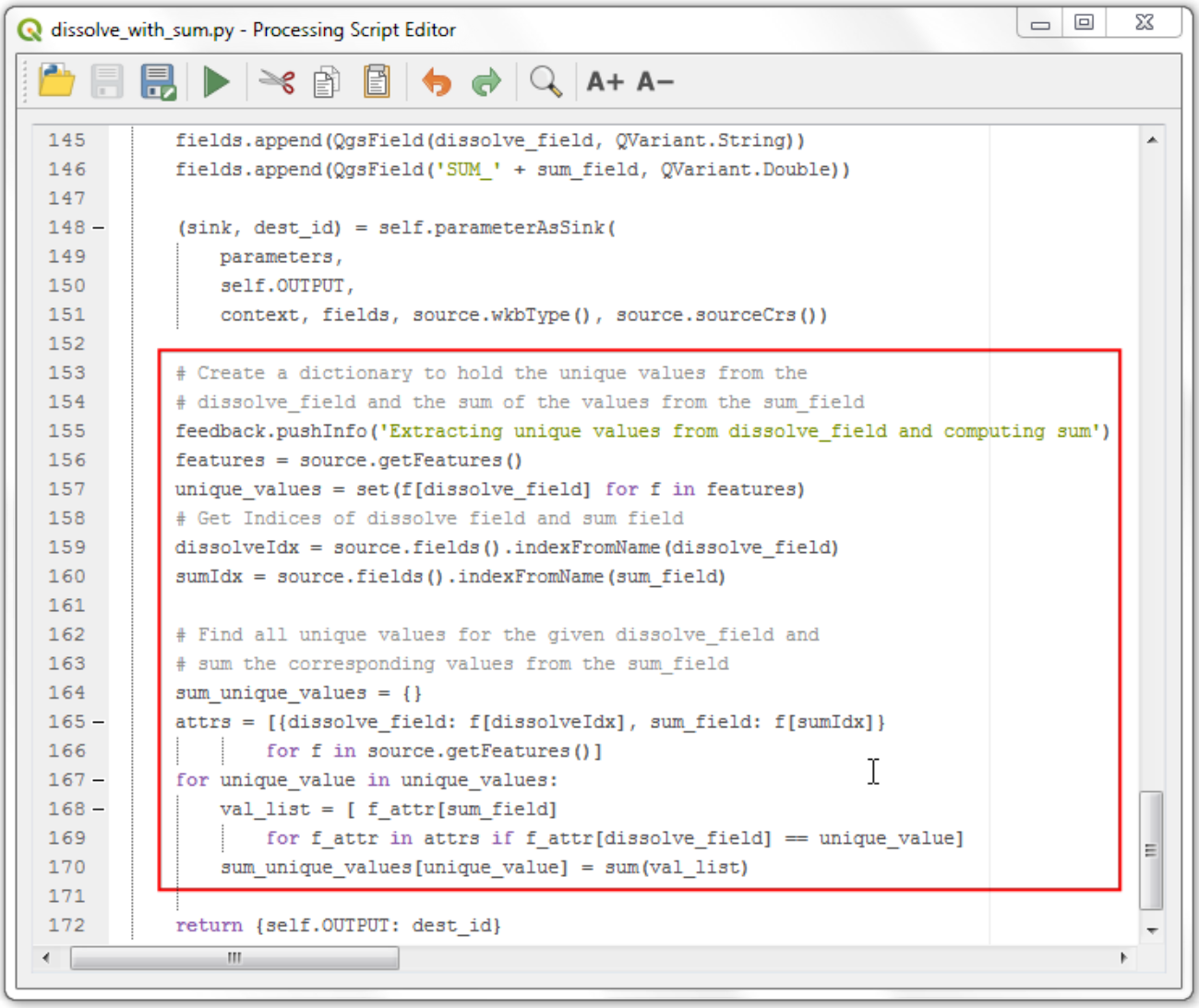




```
125
126 -    def processAlgorithm(self, parameters, context, feedback):
127          """
128          Here is where the processing itself takes place.
129          """
130 -        source = self.parameterAsSource(
131              parameters,
132              self.INPUT,
133              context
134          )
135 -        dissolve_field = self.parameterAsString(
136              parameters,
137              self.DISSOLVE_FIELD,
138              context)
139 -        sum_field = self.parameterAsString(
140              parameters,
141              self.SUM_FIELD,
142              context)
143
144          fields = QgsFields()
145          fields.append(QgsField(dissolve_field, QVariant.String))
146          fields.append(QgsField('SUM_' + sum_field, QVariant.Double))
147
148 -        (sink, dest_id) = self.parameterAsSink(
149              parameters,
150              self.OUTPUT,
151              context, fields, source.wkbType(), source.sourceCrs())
```

9. Now we will ready the input features and create a dictionary to hold the unique values from the dissolve_field and the sum of the values from the sum_field. Note the use of `feedback.pushInfo()` method to communicate the status with the user.

```
feedback.pushInfo('Extracting unique values from dissolve_field and computing sum')

features = source.getFeatures()
unique_values = set(f[dissolve_field] for f in features)

# Get Indices of dissolve field and sum field
dissolveIdx = source.fields().indexFromName(dissolve_field)
sumIdx = source.fields().indexFromName(sum_field)

# Find all unique values for the given dissolve_field and
# sum the corresponding values from the sum_field
sum_unique_values = {}
attrs = [{dissolve_field: f[dissolveIdx], sum_field: f[sumIdx]} for f in source.getFeatures()]
for unique_value in unique_values:
        val_list = [ f_attr[sum_field] for f_attr in attrs if f_attr[dissolve_field] == unique_value]
        sum_unique_values[unique_value] = sum(val_list)
```

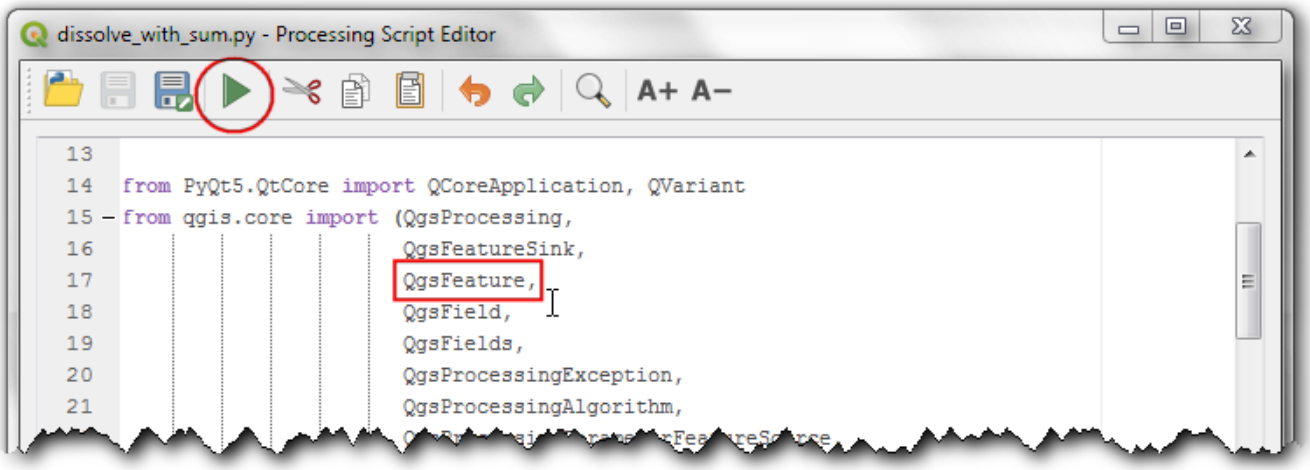


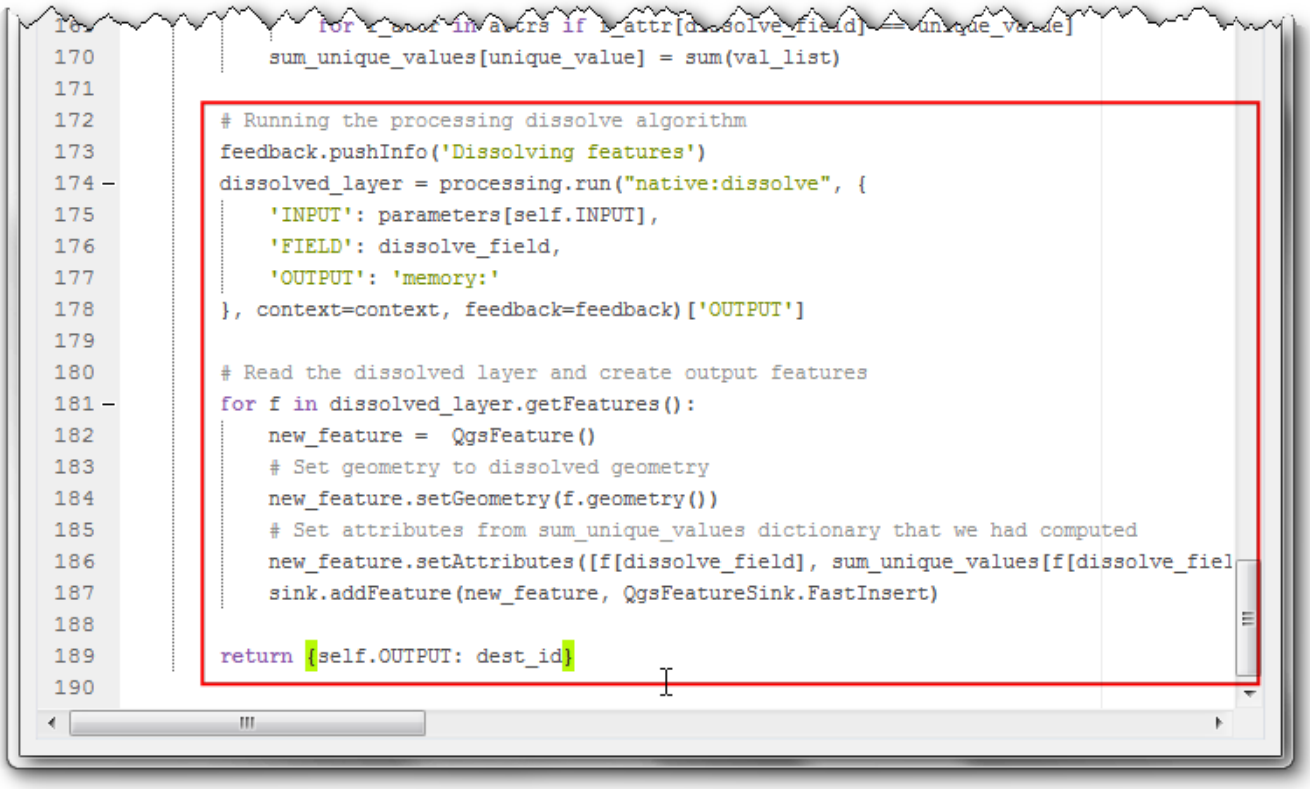
10. Next, we will call the built-in processing algorithm `native:dissolve` on the input layer to generate the dissolved geometries. Once we have the dissolved geometries, we iterate through the output of the dissolve algorithm and create new features to be added to the output. At the end we return the `dest_id` FeatureSink as the output. Now the script is ready. Click the Run button.

Note

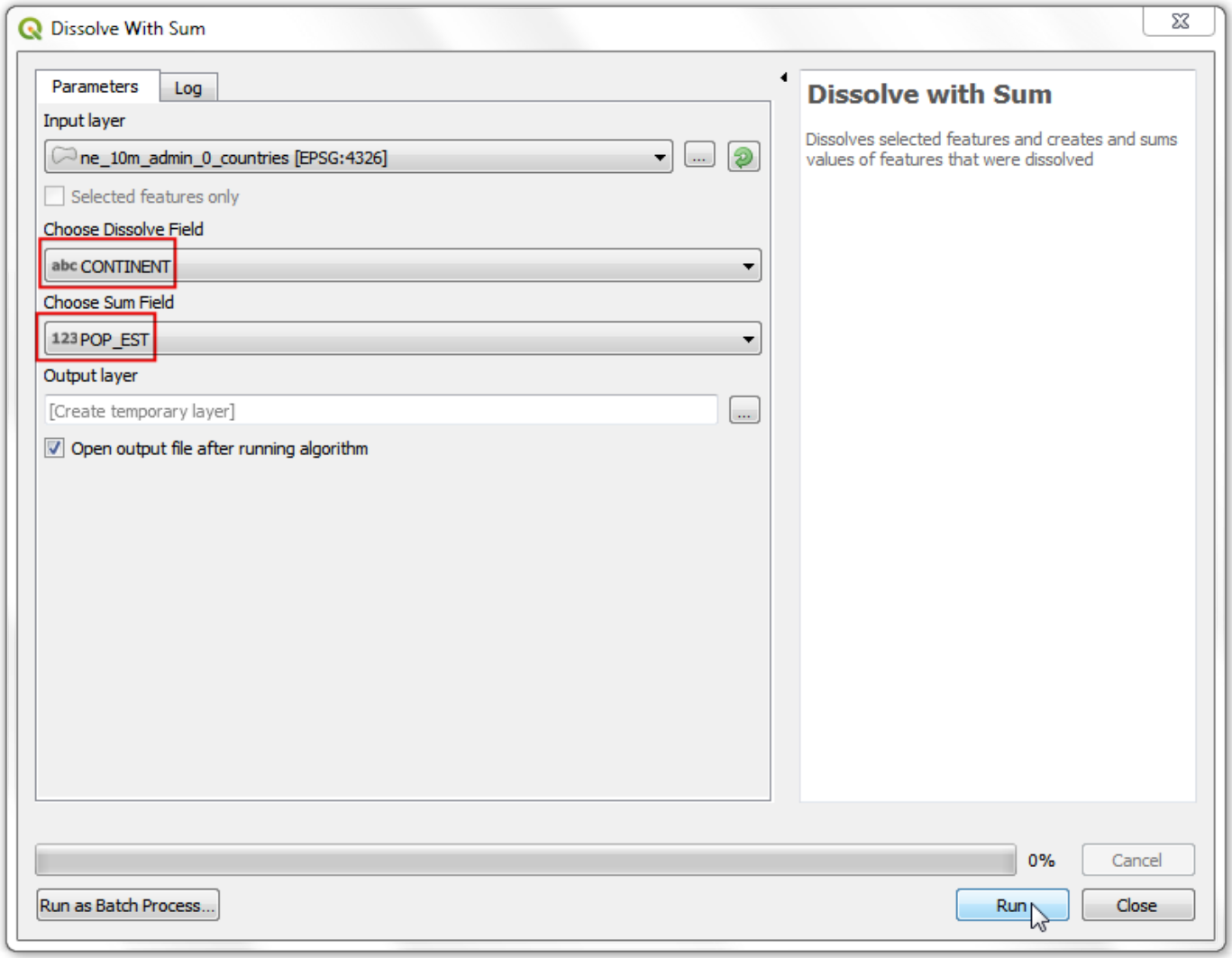
Notice the use of `parameters[self.INPUT]` to fetch the input layer from the parameters dictionary directly without defining it as a source. As we are passing the input object to the algorithm without doing anything with it, it's not necessary to define it as a source.

```
from qgis.core import QgsFeature

# Running the processing dissolve algorithm
feedback.pushInfo('Dissolving features')
dissolved_layer = processing.run("native:dissolve", {
        'INPUT': parameters[self.INPUT],
        'FIELD': dissolve_field,
        'OUTPUT': 'memory:'
        }, context=context, feedback=feedback)['OUTPUT']

# Read the dissolved layer and create output features
for f in dissolved_layer.getFeatures():
        new_feature =  QgsFeature()
        # Set geometry to dissolved geometry
        new_feature.setGeometry(f.geometry())
        # Set attributes from sum_unique_values dictionary that we had computed
        new_feature.setAttributes([f[dissolve_field], sum_unique_values[f[dissolve_field]]])
        sink.addFeature(new_feature, QgsFeatureSink.FastInsert)

return {self.OUTPUT: dest_id}
```

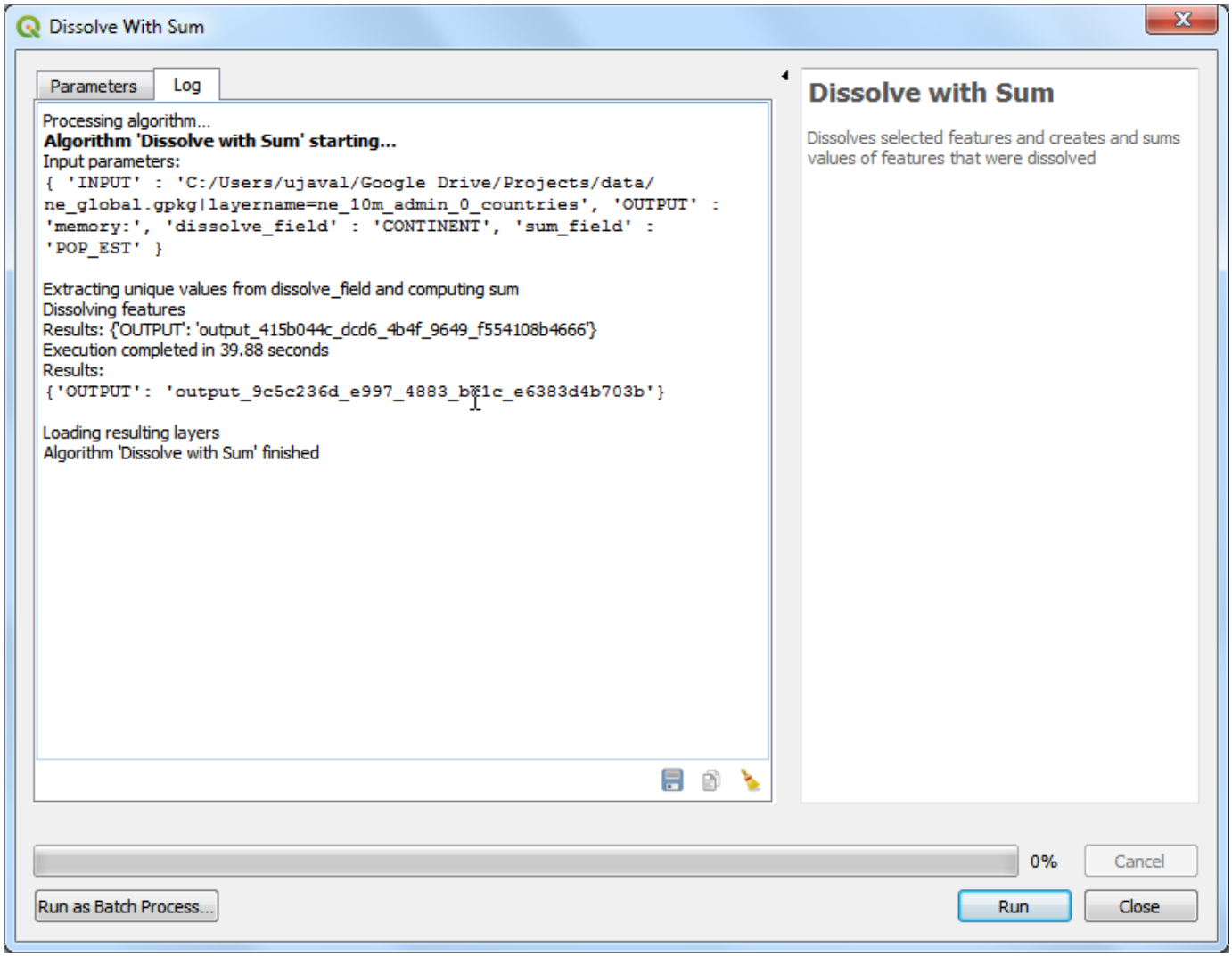




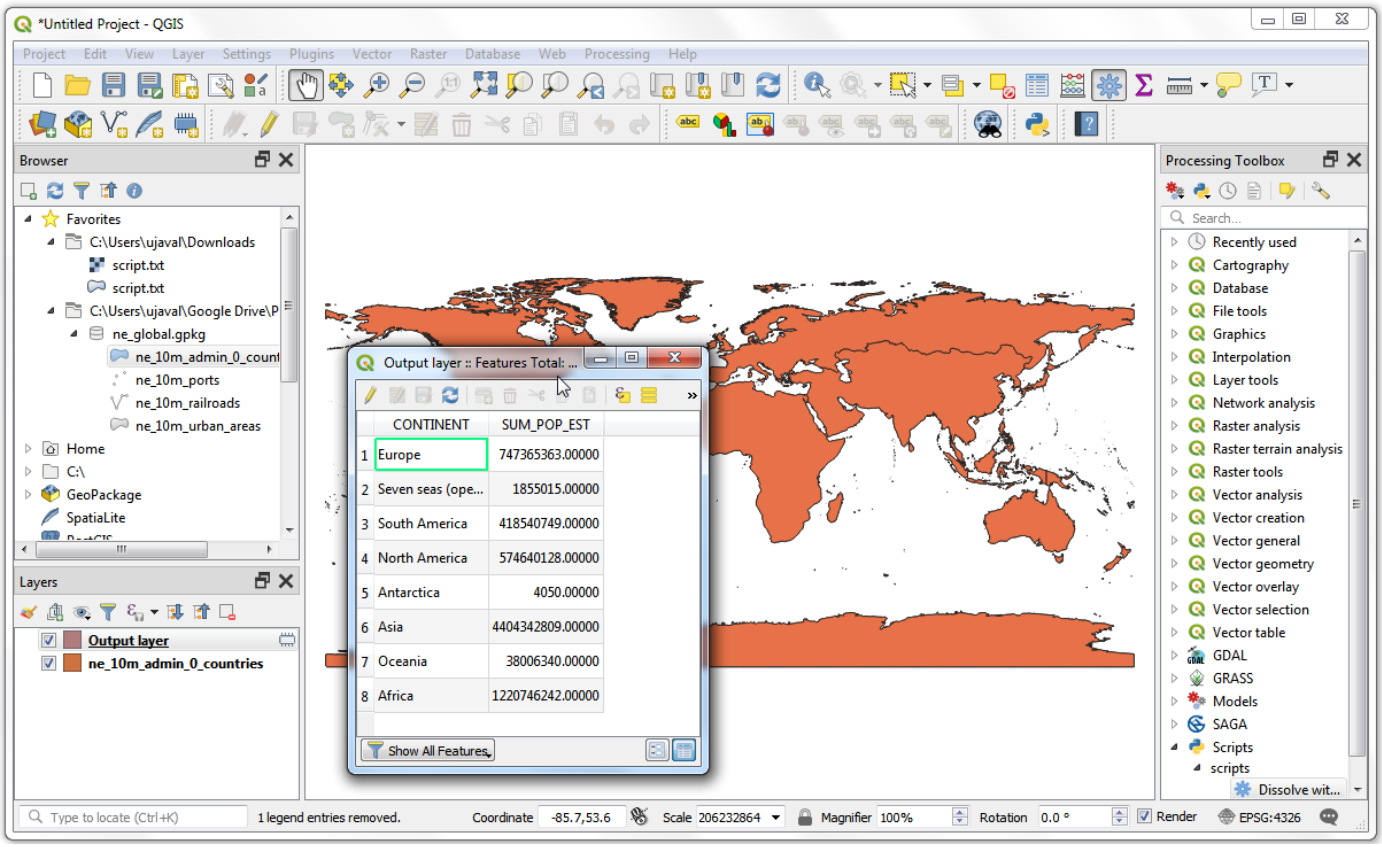
11. In the Dissolve with Sum, dialog, select `ne_10m_admin_0_countries` as the Input layer, `CONTINENT` as the Dissolve field and `POP_EST` as the Sum field. Click Run.



12. Once the processing is finished, click the Close button and switch to the main QGIS window.

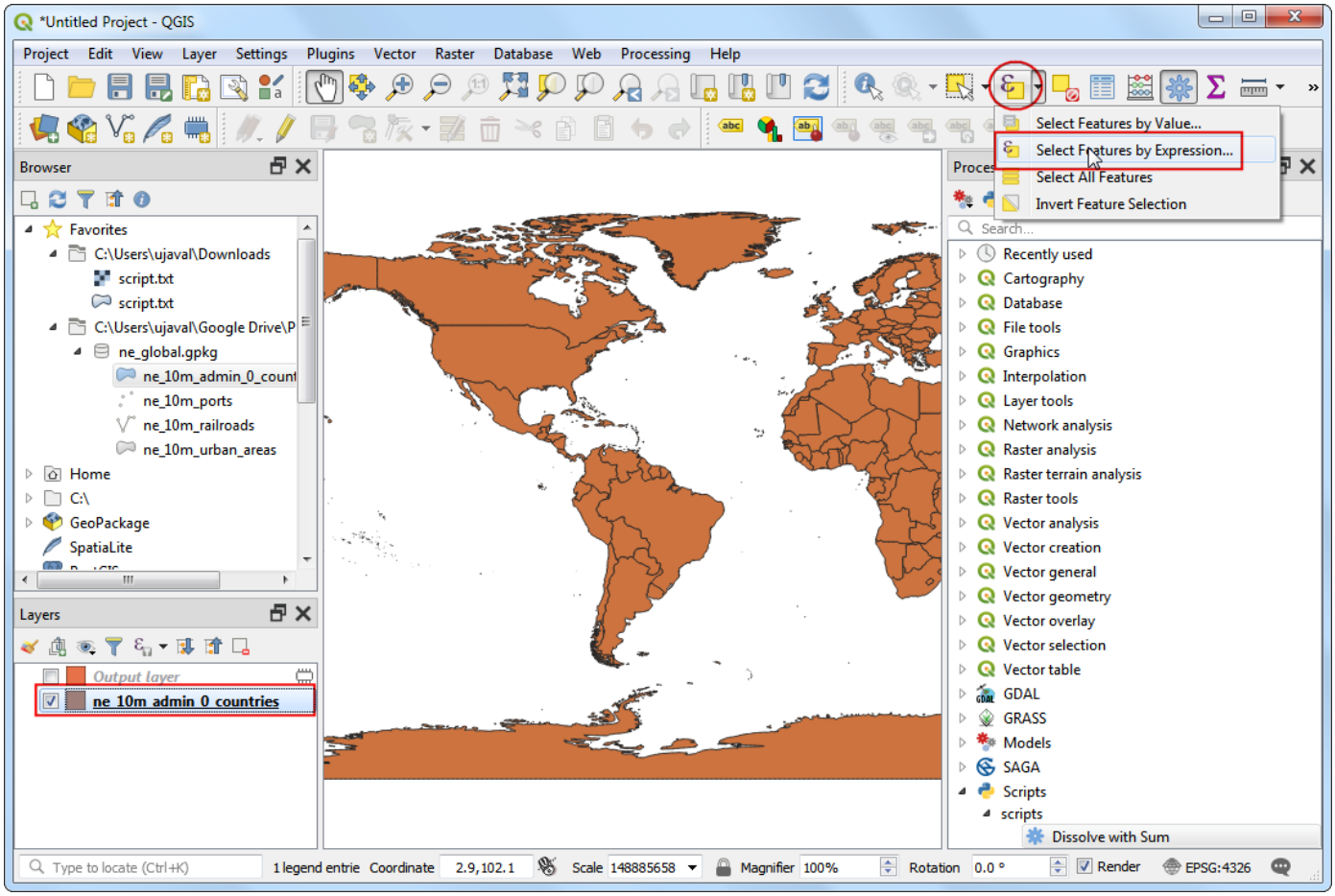


13. You will see the dissolved output layer with one feature for every continent and the total population summed from the individual countries belonging to that continent.



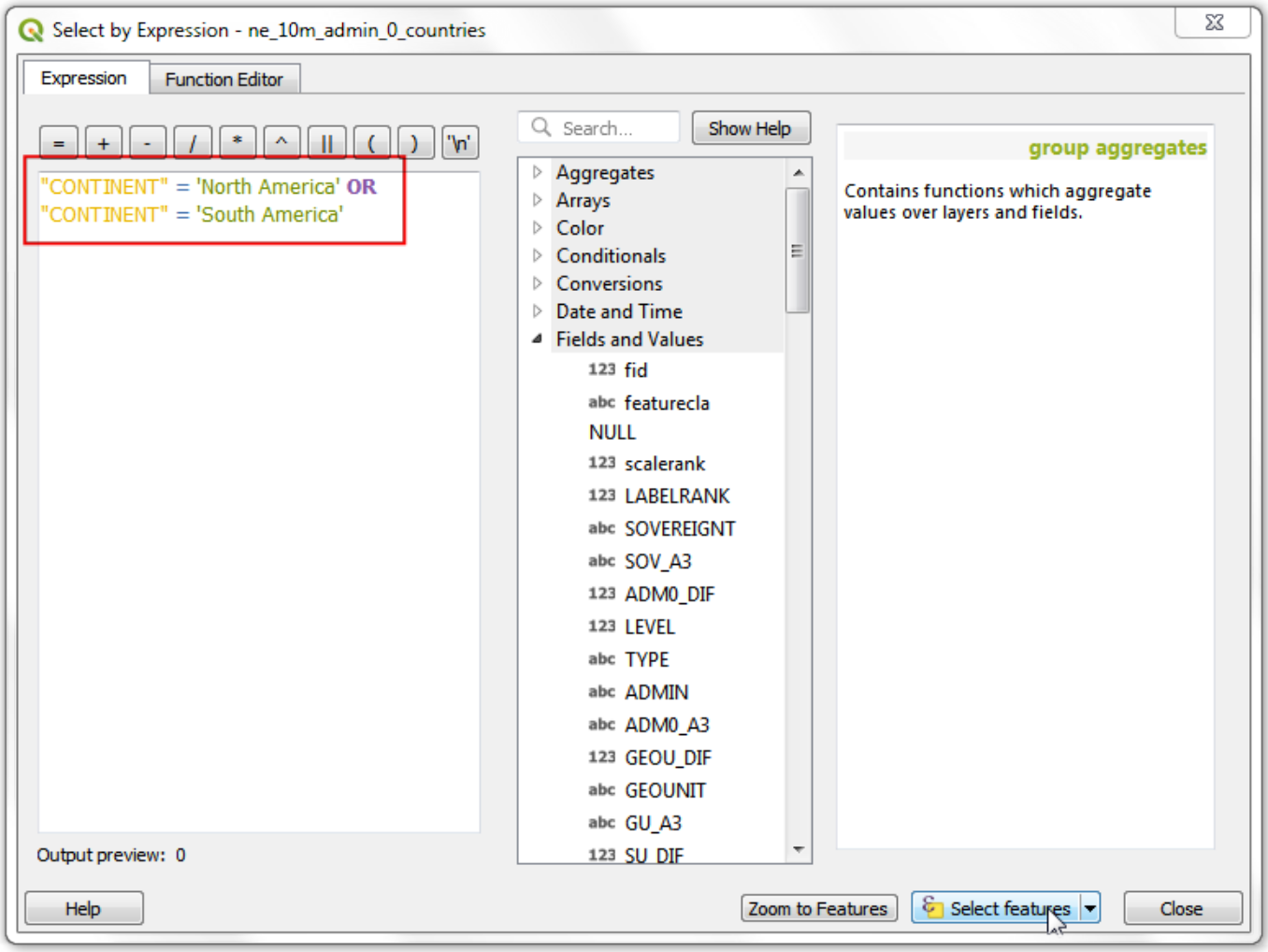
14. One another advantage of writing processing script is that the methods within the Processing Framework are aware of layer selection and automatically filter your inputs to use only the selected features. This happens because we are

defining our input as a `QgsProcessingParameterFeatureSource`. A feature source allows use of ANY object which contains vector features, not just a vector layer, so when there are selected features in your layer and ask Processing to use selected features, the input is passed on to your script as a `QgsProcessingFeatureSource` object containing selected features and not the full vector layer. Here's a quick demonstration of this functionality. Let's say we want to dissolve only certain continents. Let's create a selection using Select feature by Expression tool.

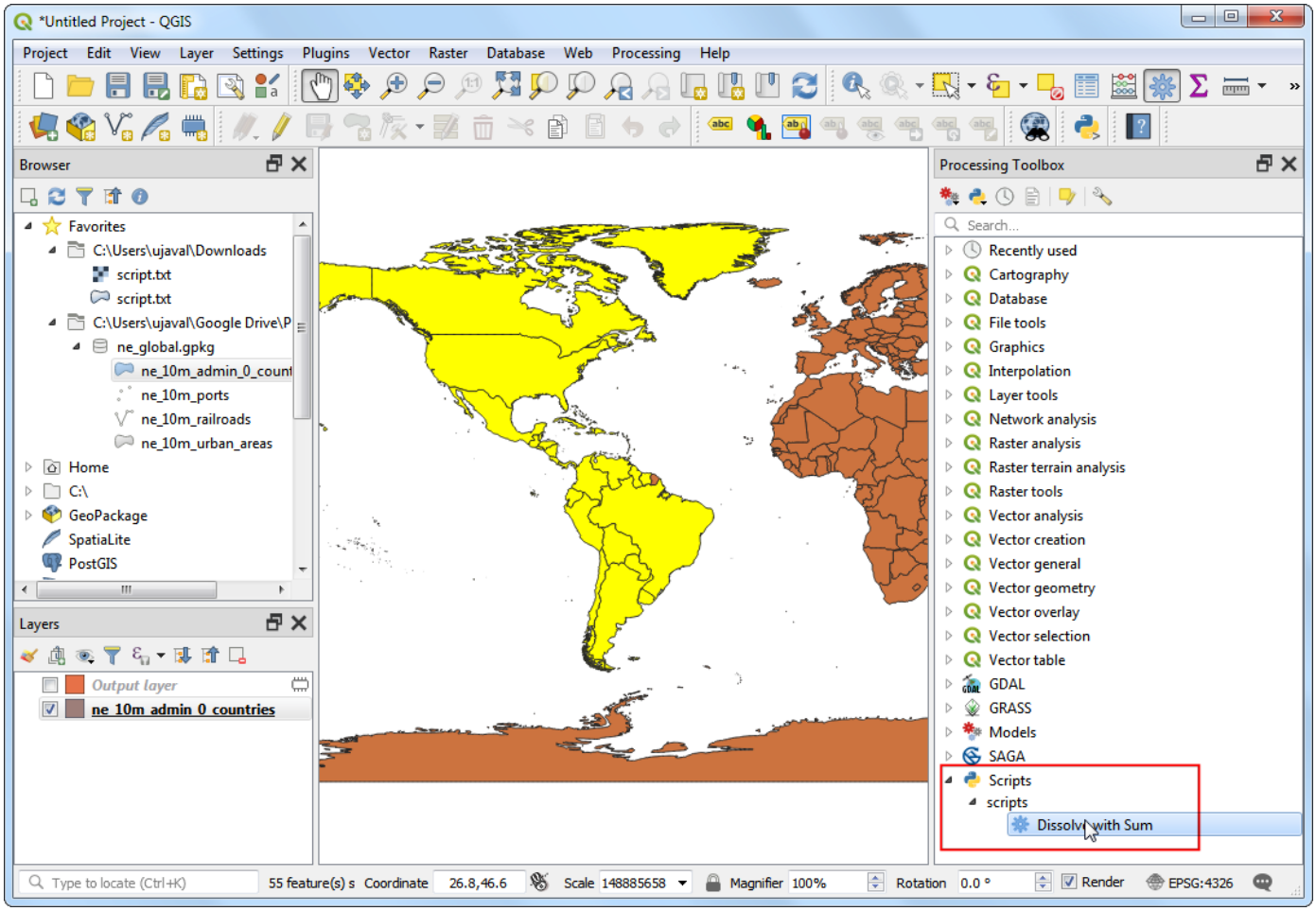


15. Enter the following expression to select features from North and South America and click Select.

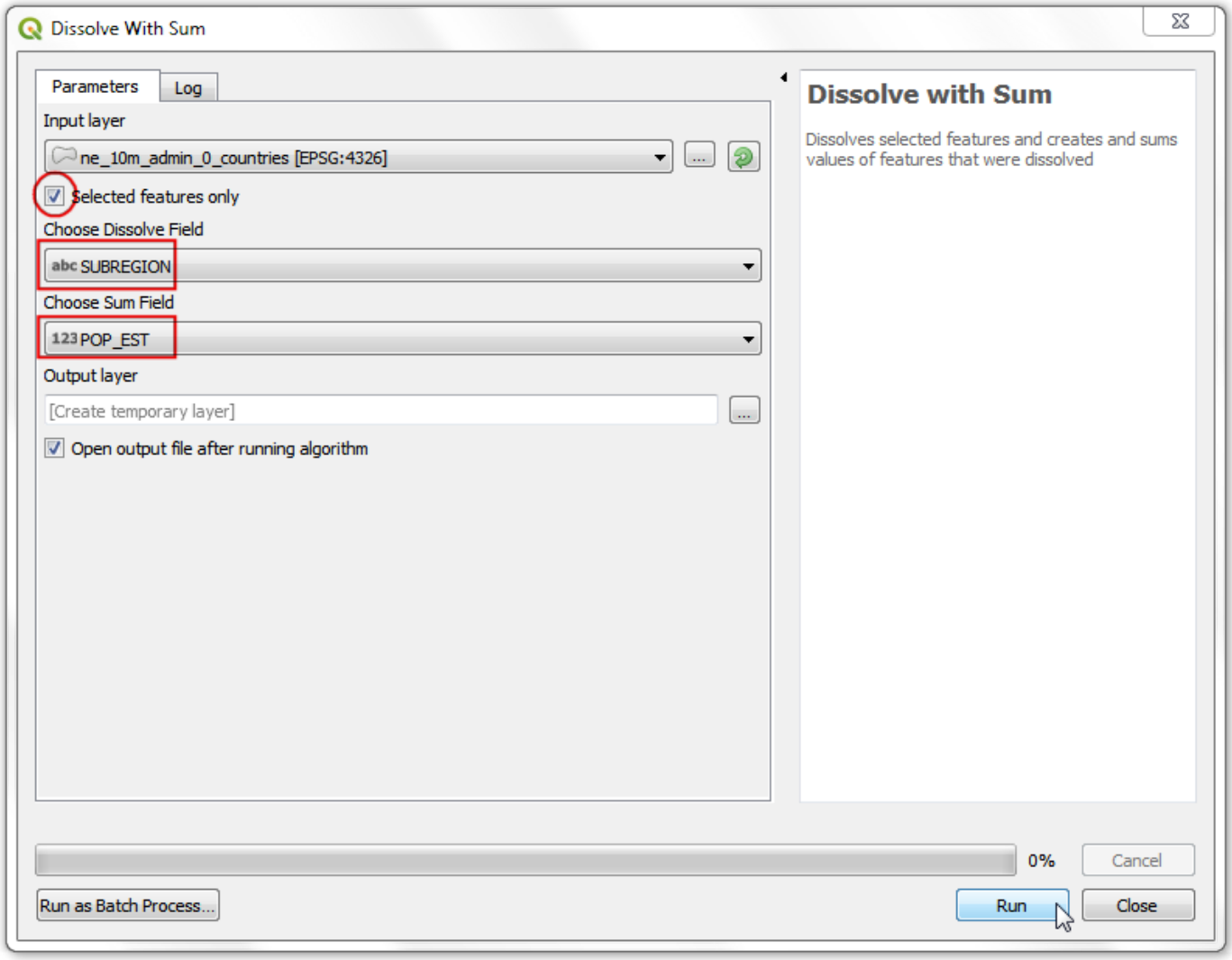
```
"CONTINENT" = 'North America' OR "CONTINENT" = 'South America'
```



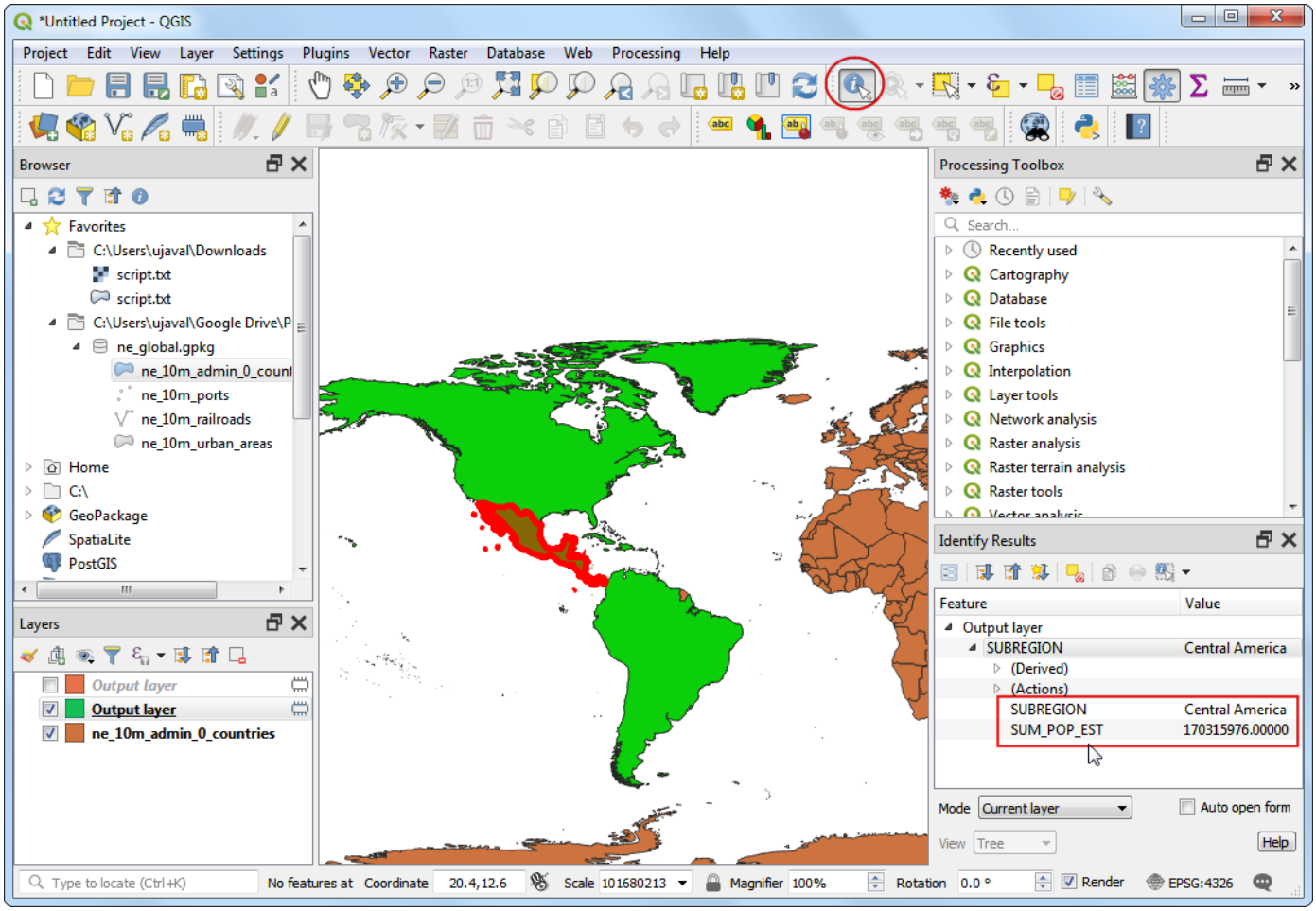
16. You will see the selected features highlighted in yellow. Locate the `dissolve_with_sum` script and double-click it to run it.



17. In the Dissolve with Sum dialog, select the `ne_10m_admin_0_countries` as the Input layer. This time, make sure you check the Selected features only box. Choose `SUBREGION` as the Dissolve field and `POP_EST` as the Sum field.



18. Once the processing finishes, click Close and switch back to the main QGIS window. You will notice a new layer with only the selected features dissolved. Click Identify button and click on a feature to inspect and verify that the script worked correctly.



Below is the complete script for reference. You may modify it to suit your needs.

```
# -*- coding: utf-8 -*-

"""
***************************************************************************
*                                                                         *
*   This program is free software; you can redistribute it and/or modify  *
*   it under the terms of the GNU General Public License as published by  *
*   the Free Software Foundation; either version 2 of the License, or     *
*   (at your option) any later version.                                   *
*                                                                         *
***************************************************************************
"""

from PyQt5.QtCore import QCoreApplication, QVariant
from qgis.core import (QgsProcessing,
                       QgsFeatureSink,
                       QgsFeature,
                       QgsField,
                       QgsFields,
                       QgsProcessingException,
                       QgsProcessingAlgorithm,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFeatureSink,
                       QgsProcessingParameterField,
                       )
import processing


class DissolveProcessingAlgorithm(QgsProcessingAlgorithm):
    """
    Dissolve algorithm that dissolves features based on selected
    attribute and summarizes the selected field by cumputing the
    sum of dissolved features.
    """
    INPUT = 'INPUT'
    OUTPUT = 'OUTPUT'
    DISSOLVE_FIELD = 'dissolve_field'
    SUM_FIELD = 'sum_field'

    def tr(self, string):
        """
        Returns a translatable string with the self.tr() function.
        """
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        return DissolveProcessingAlgorithm()

    def name(self):
        """
        Returns the algorithm name, used for identifying the algorithm. This
        string should be fixed for the algorithm, and must not be localised.
        The name should be unique within each provider. Names should contain
        lowercase alphanumeric characters only and no spaces or other
        formatting characters.
        """
        return 'dissolve_with_sum'

    def displayName(self):
        """
        Returns the translated algorithm name, which should be used for any
        user-visible display of the algorithm name.
        """
        return self.tr('Dissolve with Sum')
```

```
    def group(self):
        """
        Returns the name of the group this algorithm belongs to. This string
        should be localised.
        """
        return self.tr('scripts')

    def groupId(self):
        """
        Returns the unique ID of the group this algorithm belongs to. This
        string should be fixed for the algorithm, and must not be localised.
        The group id should be unique within each provider. Group id should
        contain lowercase alphanumeric characters only and no spaces or other
        formatting characters.
        """
        return 'scripts'

    def shortHelpString(self):
        """
        Returns a localised short helper string for the algorithm. This string
        should provide a basic description about what the algorithm does and the
        parameters and outputs associated with it..
        """
        return self.tr("Dissolves selected features and creates and sums values of features that were dissolv

    def initAlgorithm(self, config=None):
        """
        Here we define the inputs and output of the algorithm, along
        with some other properties.
        """
        # We add the input vector features source. It can have any kind of
        # geometry.
        self.addParameter(
            QgsProcessingParameterFeatureSource(
                self.INPUT,
                self.tr('Input layer'),
                [QgsProcessing.TypeVectorAnyGeometry]
            )
        )
        self.addParameter(
            QgsProcessingParameterField(
                self.DISSOLVE_FIELD,
                'Choose Dissolve Field',
                '',
                self.INPUT))
        self.addParameter(
            QgsProcessingParameterField(
                self.SUM_FIELD,
                'Choose Sum Field',
                '',
                self.INPUT))
        # We add a feature sink in which to store our processed features (this
        # usually takes the form of a newly created vector layer when the
        # algorithm is run in QGIS).
        self.addParameter(
            QgsProcessingParameterFeatureSink(
                self.OUTPUT,
                self.tr('Output layer')
            )
        )

    def processAlgorithm(self, parameters, context, feedback):
        """
        Here is where the processing itself takes place.
        """
        source = self.parameterAsSource(
```

```
            parameters,
            self.INPUT,
            context
        )
        dissolve_field = self.parameterAsString(
            parameters,
            self.DISSOLVE_FIELD,
            context)
        sum_field = self.parameterAsString(
            parameters,
            self.SUM_FIELD,
            context)

        fields = QgsFields()
        fields.append(QgsField(dissolve_field, QVariant.String))
        fields.append(QgsField('SUM_' + sum_field, QVariant.Double))

        (sink, dest_id) = self.parameterAsSink(
            parameters,
            self.OUTPUT,
            context, fields, source.wkbType(), source.sourceCrs())

        # Create a dictionary to hold the unique values from the
        # dissolve_field and the sum of the values from the sum_field
        feedback.pushInfo('Extracting unique values from dissolve_field and computing sum')
        features = source.getFeatures()
        unique_values = set(f[dissolve_field] for f in features)
        # Get Indices of dissolve field and sum field
        dissolveIdx = source.fields().indexFromName(dissolve_field)
        sumIdx = source.fields().indexFromName(sum_field)

        # Find all unique values for the given dissolve_field and
        # sum the corresponding values from the sum_field
        sum_unique_values = {}
        attrs = [{dissolve_field: f[dissolveIdx], sum_field: f[sumIdx]}
                for f in source.getFeatures()]
        for unique_value in unique_values:
            val_list = [ f_attr[sum_field]
                for f_attr in attrs if f_attr[dissolve_field] == unique_value]
            sum_unique_values[unique_value] = sum(val_list)

        # Running the processing dissolve algorithm
        feedback.pushInfo('Dissolving features')
        dissolved_layer = processing.run("native:dissolve", {
            'INPUT': parameters[self.INPUT],
            'FIELD': dissolve_field,
            'OUTPUT': 'memory:'
        }, context=context, feedback=feedback)['OUTPUT']

        # Read the dissolved layer and create output features
        for f in dissolved_layer.getFeatures():
            new_feature =  QgsFeature()
            # Set geometry to dissolved geometry
            new_feature.setGeometry(f.geometry())
            # Set attributes from sum_unique_values dictionary that we had computed
            new_feature.setAttributes([f[dissolve_field], sum_unique_values[f[dissolve_field]]])
            sink.addFeature(new_feature, QgsFeatureSink.FastInsert)

        return {self.OUTPUT: dest_id}
```

Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Running and Scheduling QGIS Processing Jobs

You can automate a lot of tasks in QGIS using Python scripting (PyQGIS) and the Processing Framework. Most of the time, you would run these scripts manually while QGIS is open. While that is helpful, many times you need a way to run this jobs via the command-line and without needing to open QGIS. Fortunately, you can write standalone python scripts that use QGIS libraries and can be run via the command-line. In this tutorial, we will learn how to write and schedule a job that uses the QGIS Processing framework.

Overview of the task

Let's say we are working on some analysis using shapefiles of a region. The shapefiles are updated on a daily basis and we always need the latest file. But before we can use these files, we need to cleanup the data. We can setup a QGIS job that automates this process and runs it daily so you have the latest cleaned up shapefiles for your work. We will write a standalone Python script that downloads a shapefile and run topological cleaning operations on a daily basis.

Other skills you will learn

- Downloading and unzipping files using Python.

- Running any Processing algorithm via PyQGIS.
- Fixing topological errors in a vector layer.

Get the data

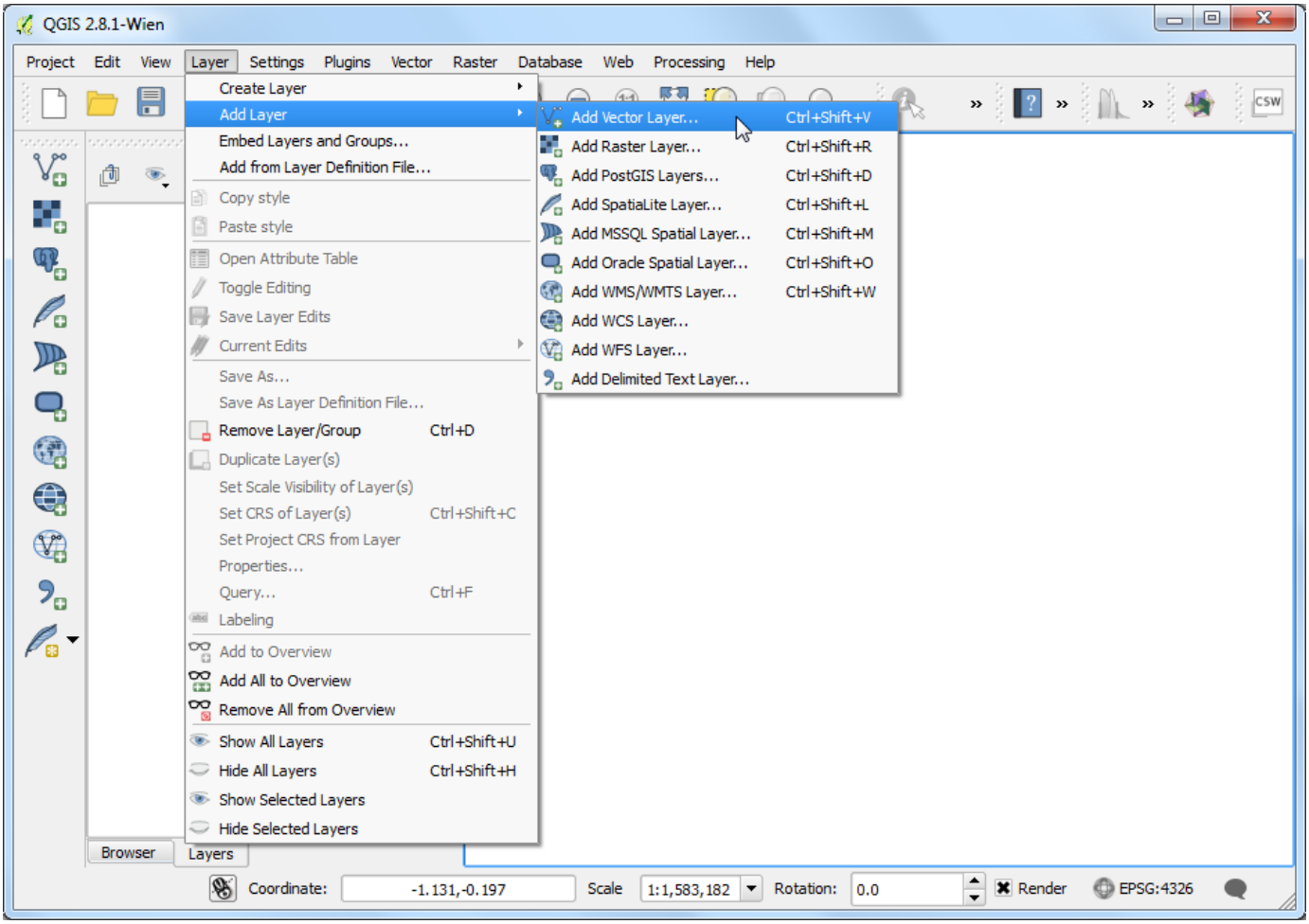
Geofabrik (http://www.geofabrik.de/) provides daily updated shapefiles of OpenStreetMap (http://www.openstreetmap.org/) datasets.

We will use shapefiles for Fiji (http://download.geofabrik.de/australia-oceania.html) for this exercise. Download the fiji-latest.shp.zip (http://download.geofabrik.de/australia-oceania/fiji-latest.shp.zip) and unzip it to a folder on your disk.

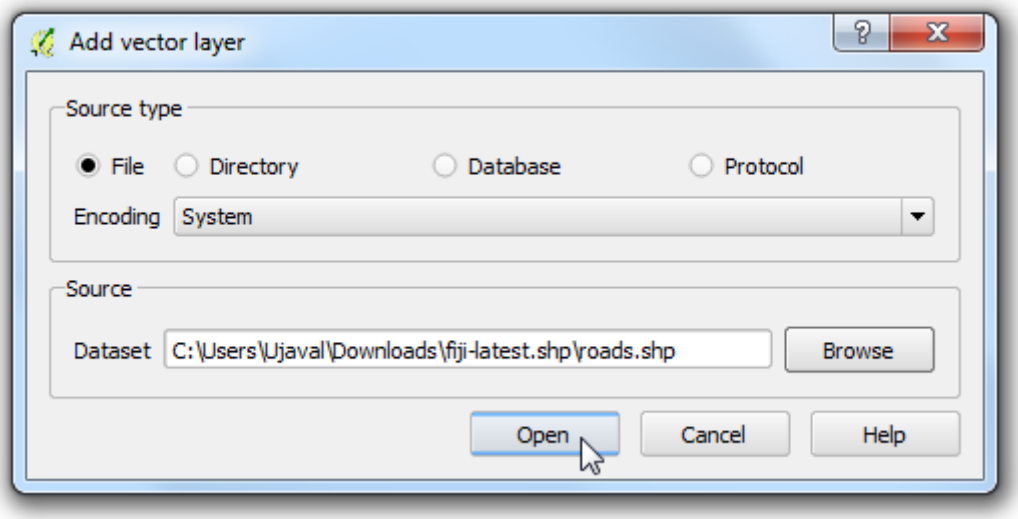
Data Source [GEOFABRIK] (credits.html#geofabrik)

Procedure

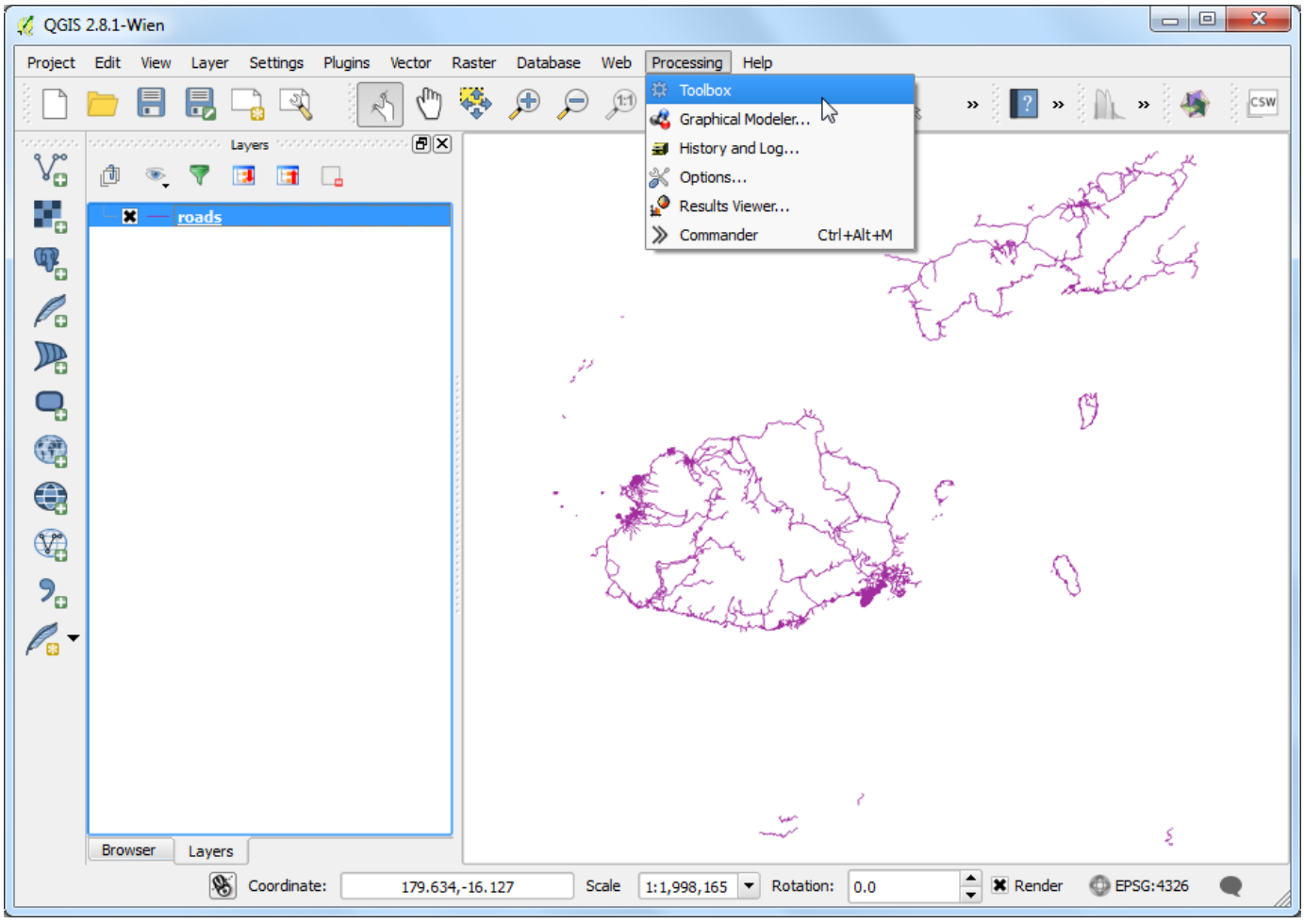
1. We will first run through the process of cleaning the shapefile manually to note the commands that we will use in the python script. Launch QGIS and go to Layer ▸ Add Layer ▸ Add Vector Layer.



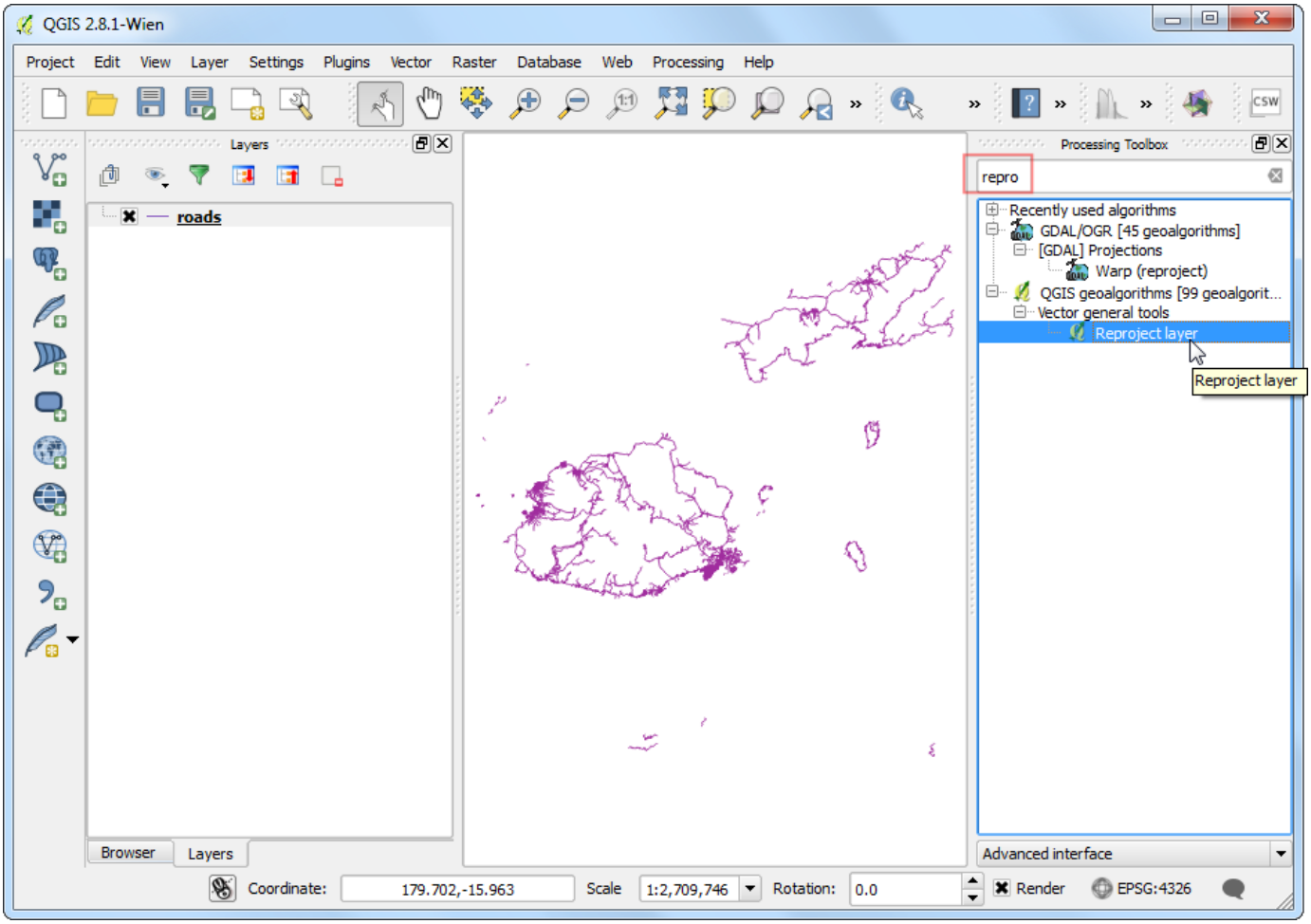
2. Browse to the folder containing the unzipped shapefiles and select the `roads.shp` file and click Open.



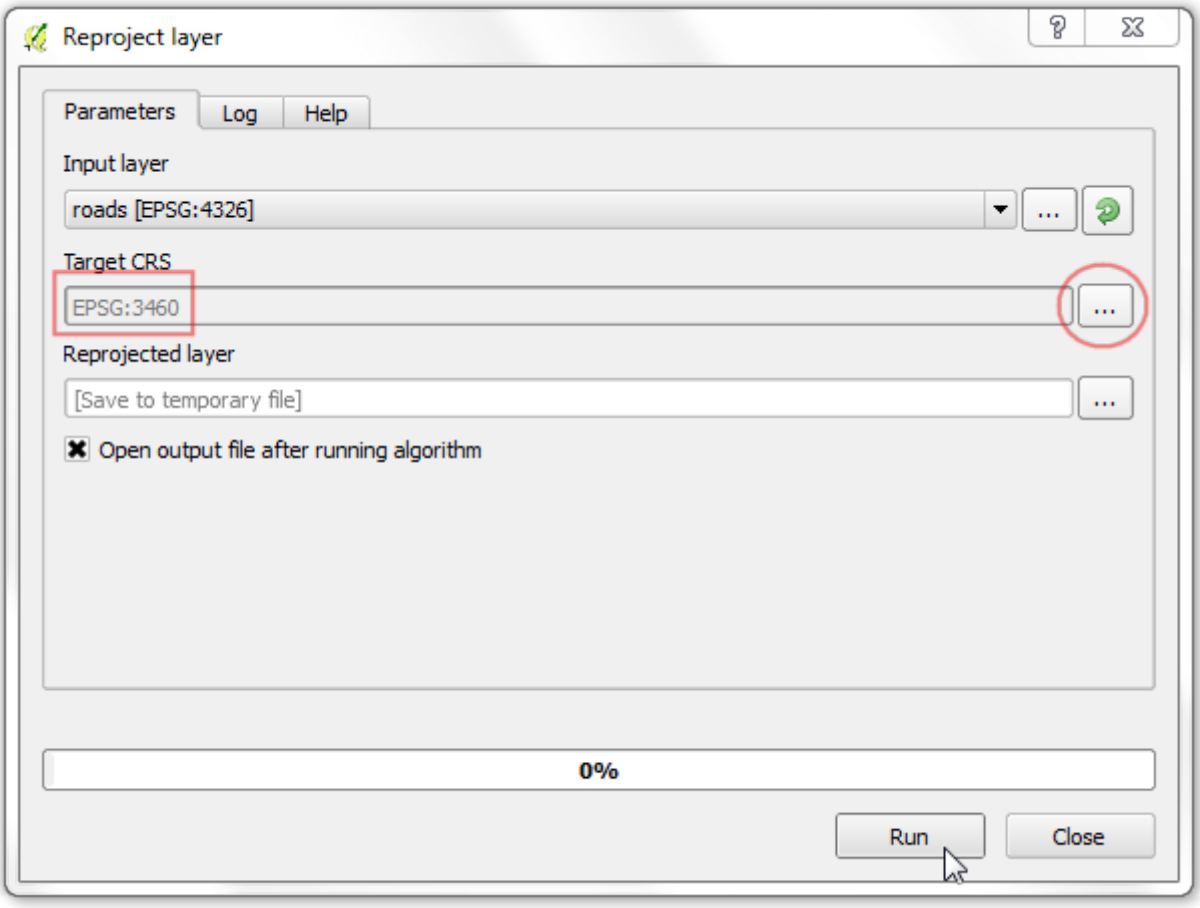
3. First we must re-project the roads layer to a Projected CRS. This will allow us to use *meters* as units when performing analysis instead of degrees. Open Processing ‣ Toolbox.



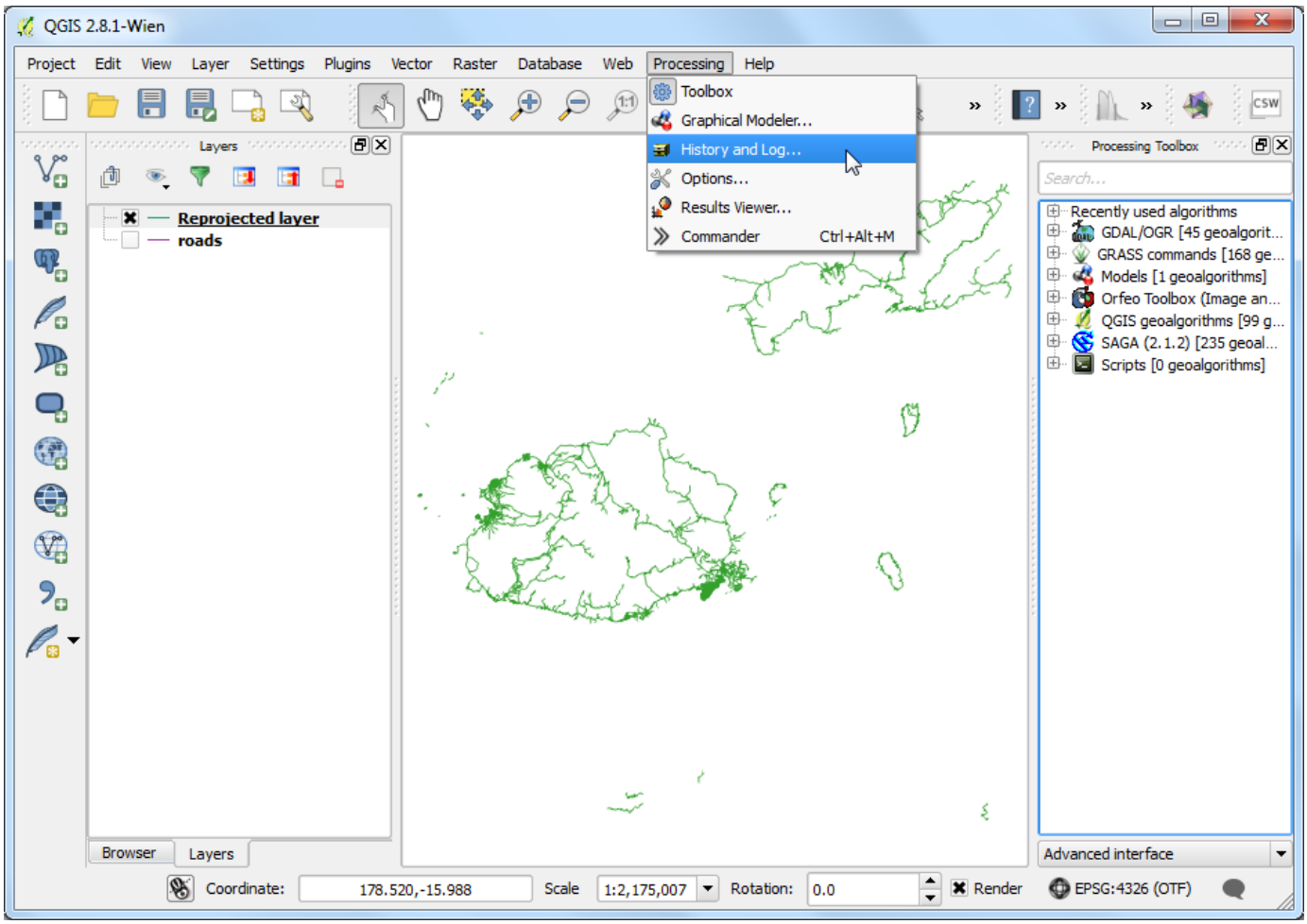
4. Search for the Reproject layer tool. Double-click it to launch the dialog.



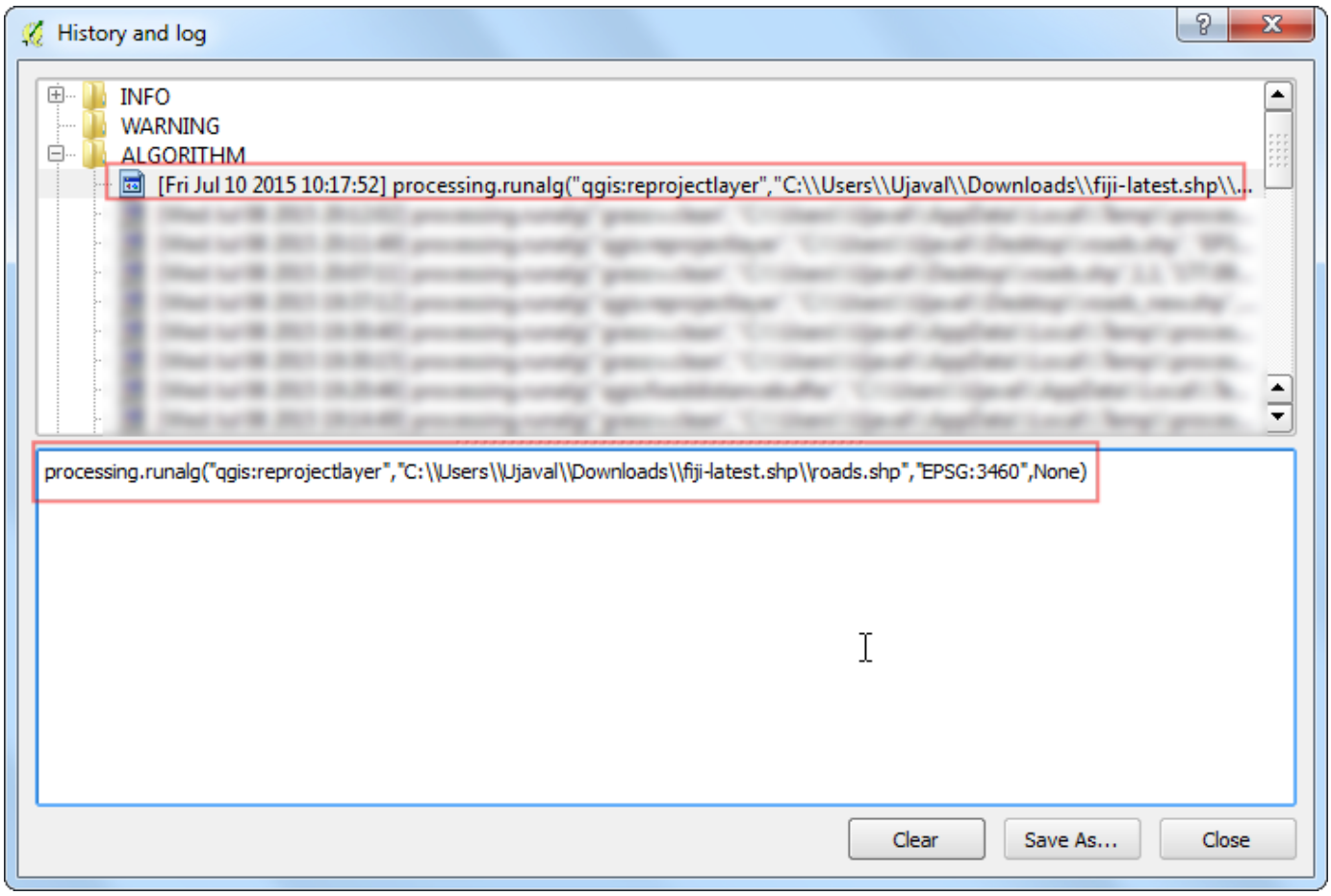
5. In the Reproject layer dialog, select the `roads` layer as Input layer. We will use `EPSG:3460 Fiji 1986 / Fiji Map Grid` CRS as the Target CRS. Click Run.



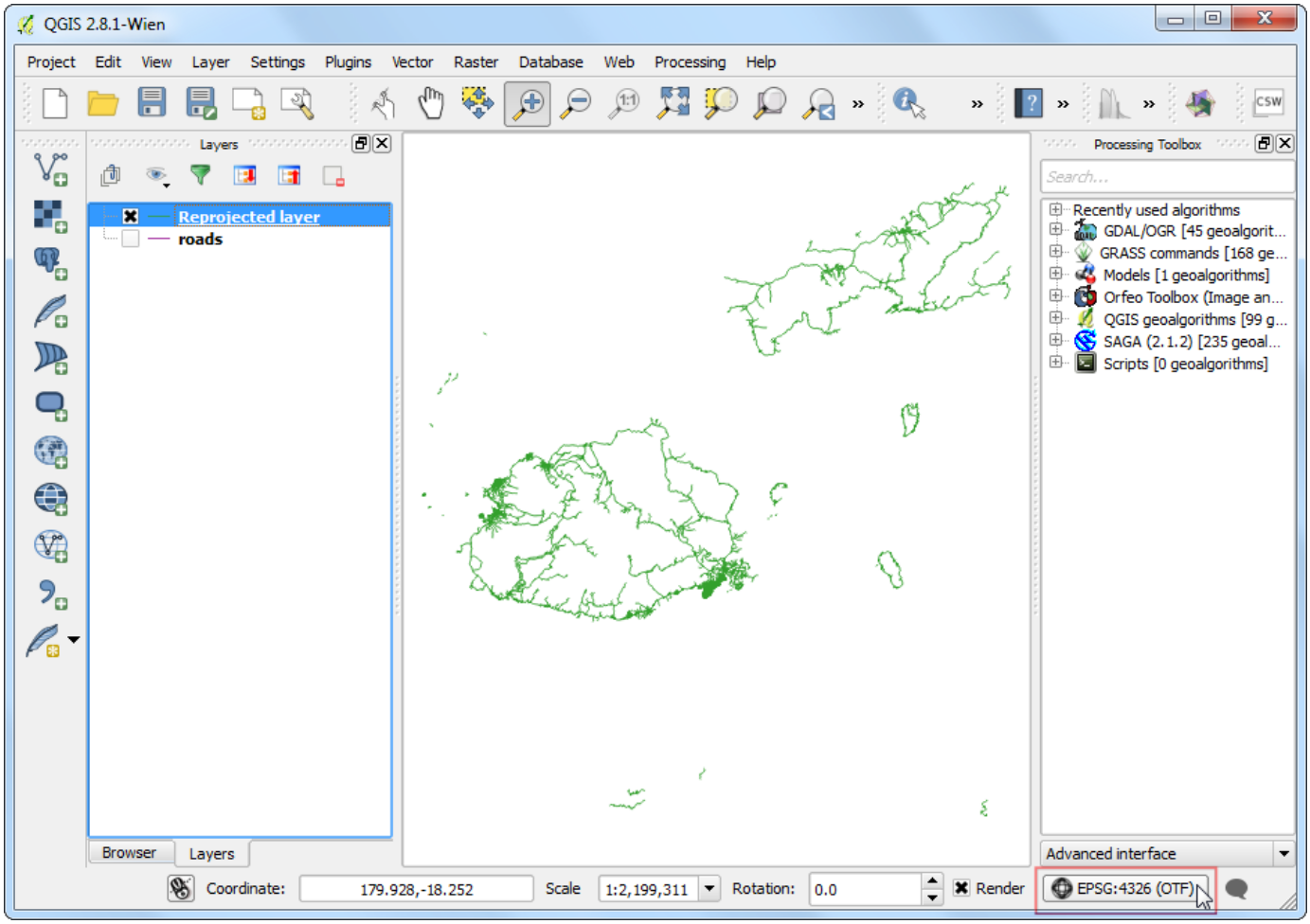
6. Once the process finishes, you will see the reprojected layer loaded in QGIS. Go to Processing ▸ History and Log...



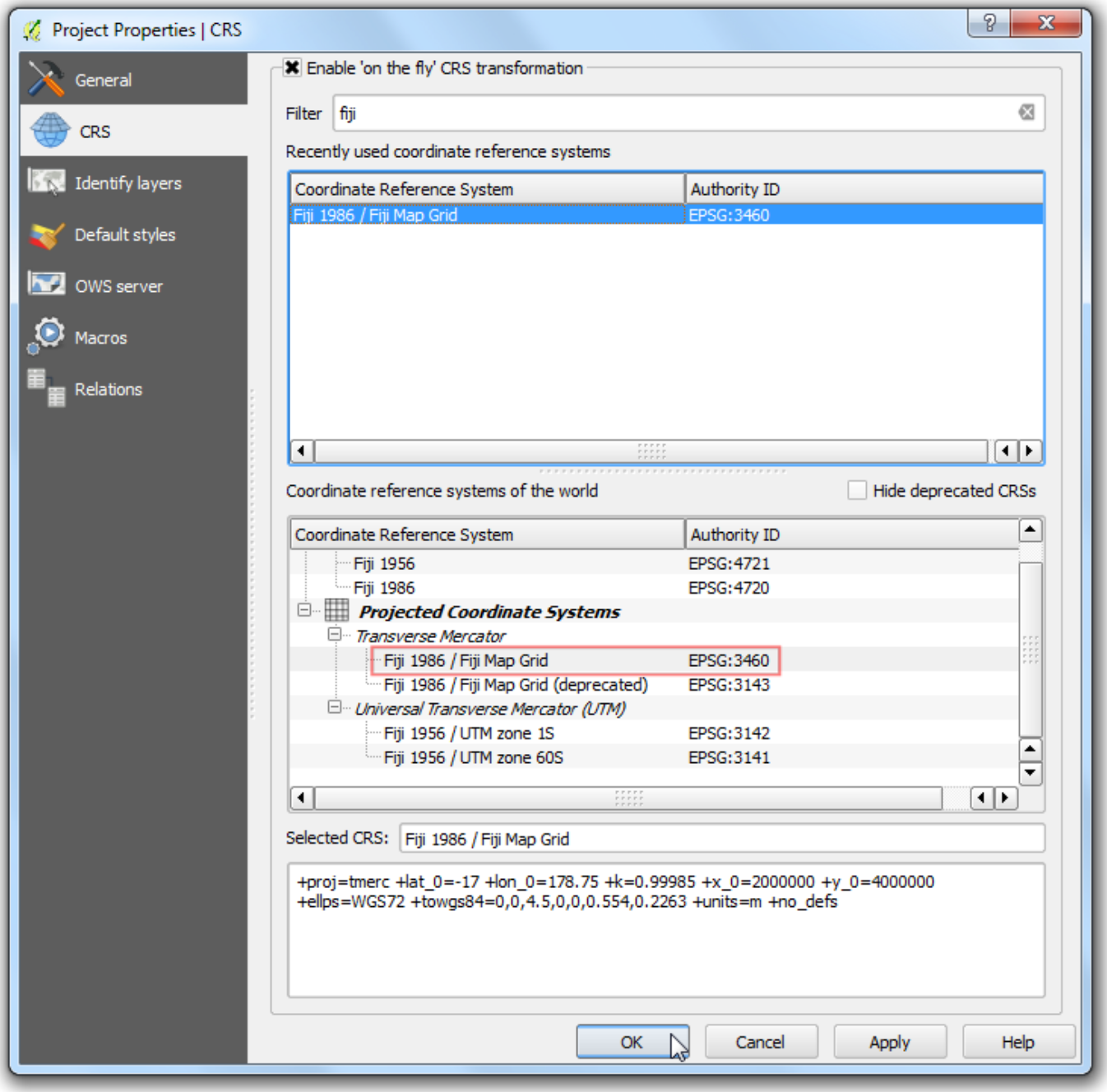
7. In the History and Log dialog, expand the Algorithm folder and select the latest entry. You will see the full processing command shown in the bottom panel. Note this command for use in our script.



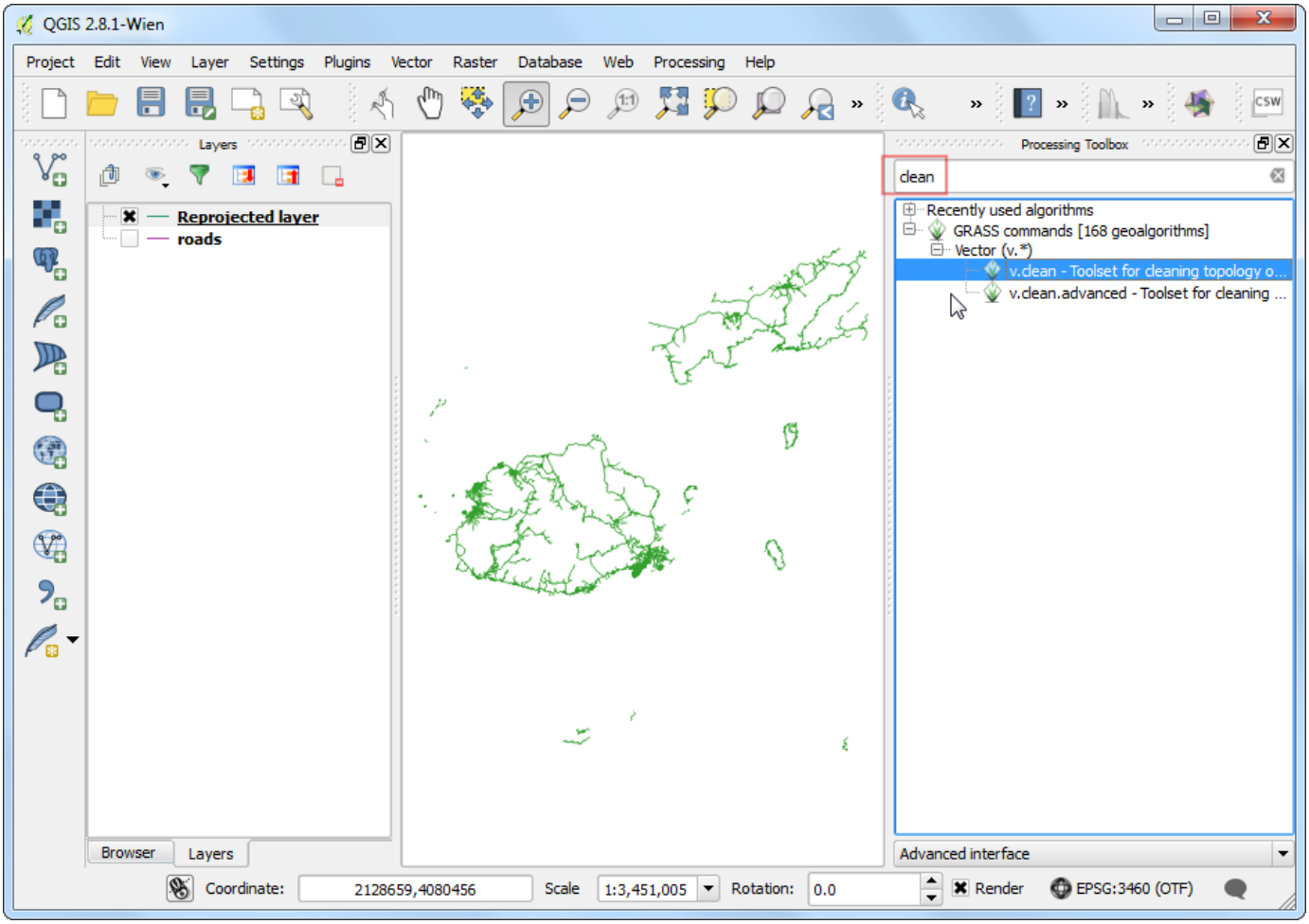
8. Back in the main QGIS Window, click at the CRS button in the bottom-right corner.



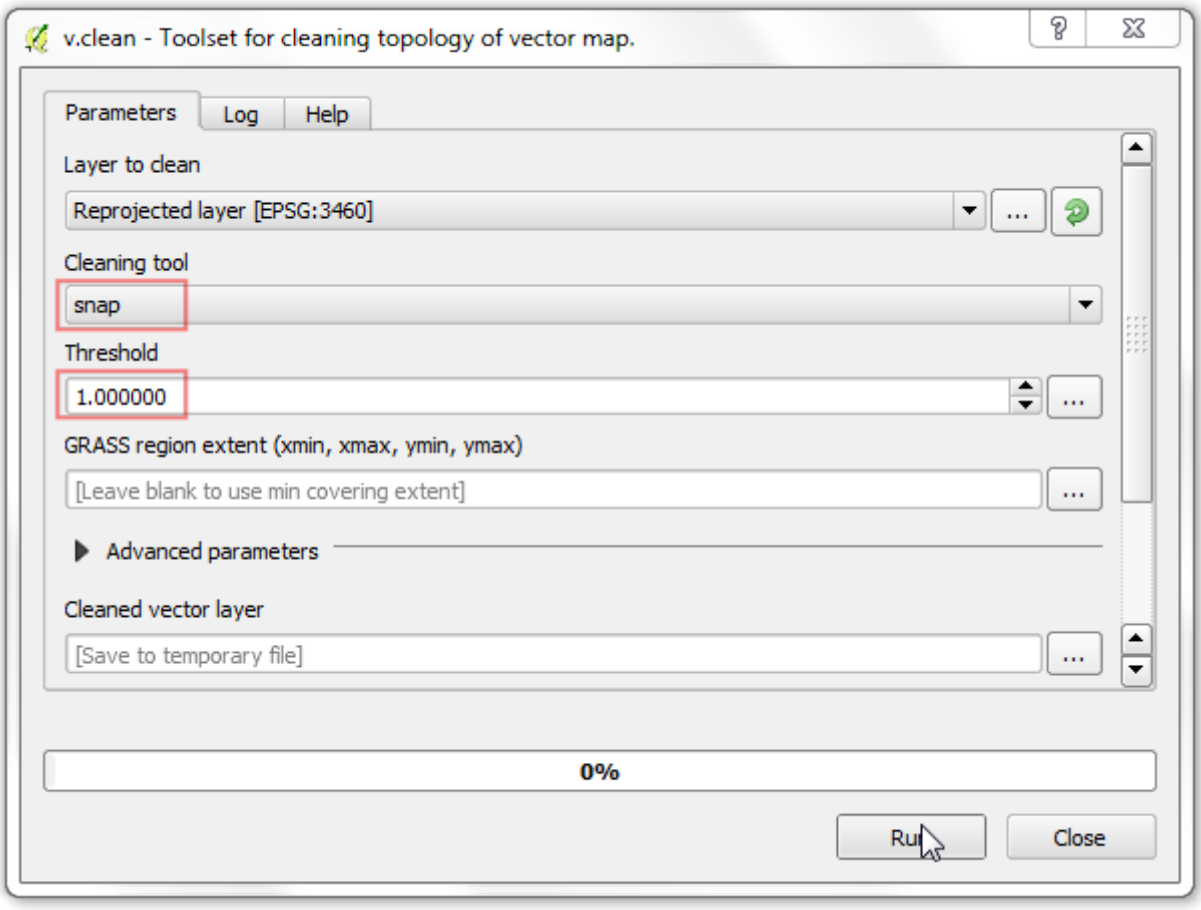
9. In the Project Properties | CRS dialog, check the Enable on-the-fly CRS transformation and select `EPSG:3460 Fiji 1986 / Fiji Map Grid` as the CRS. This will ensure that our original and reprojected layers will line up correctly.



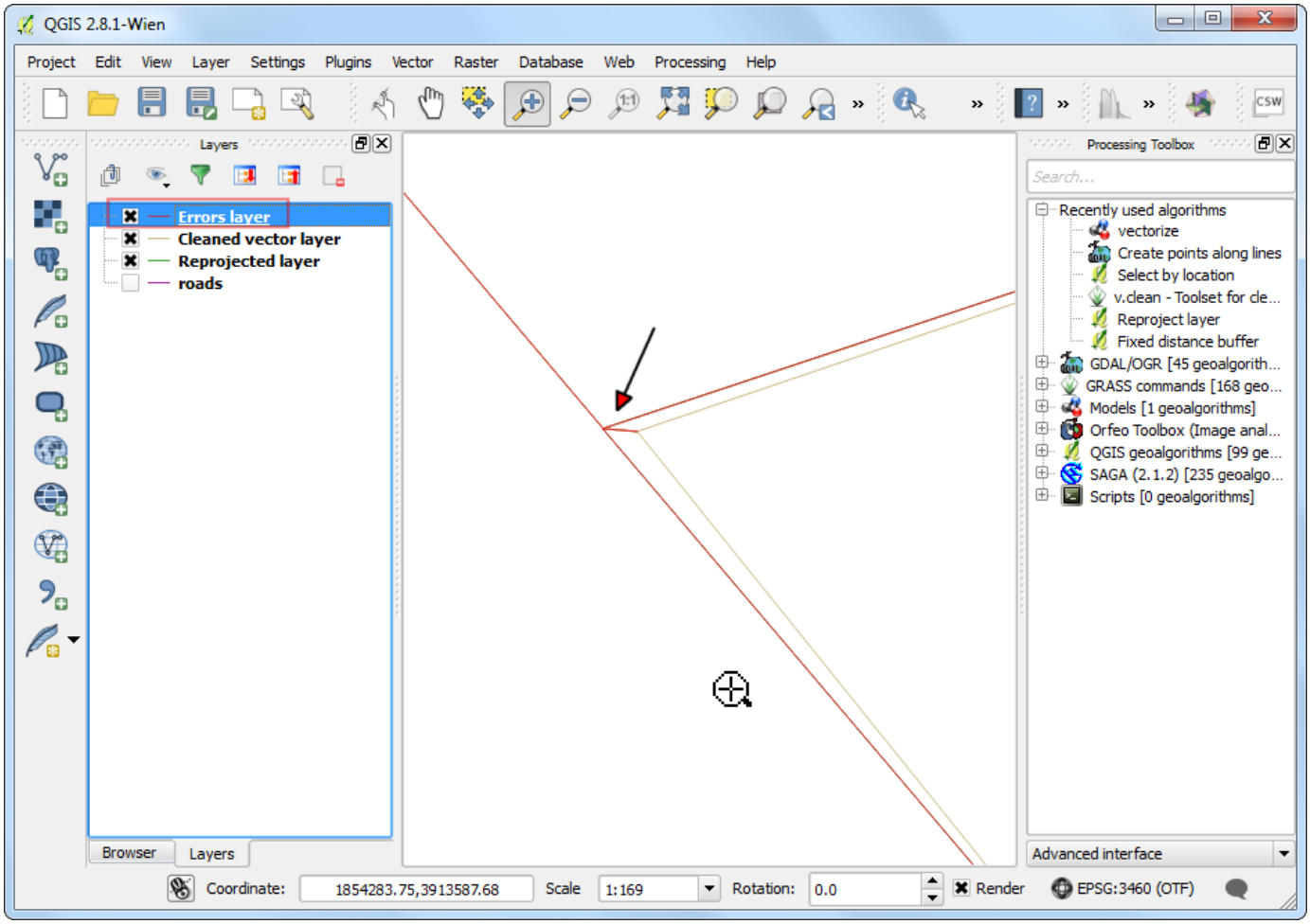
10. Now we will run the cleaning operation. GRASS has a very powerful set of topological cleaning tools. These are available in QGIS via the `v.clean` algorithm. Search for this algorithm in the Processing Toolbox and double-click it to launch the dialog.



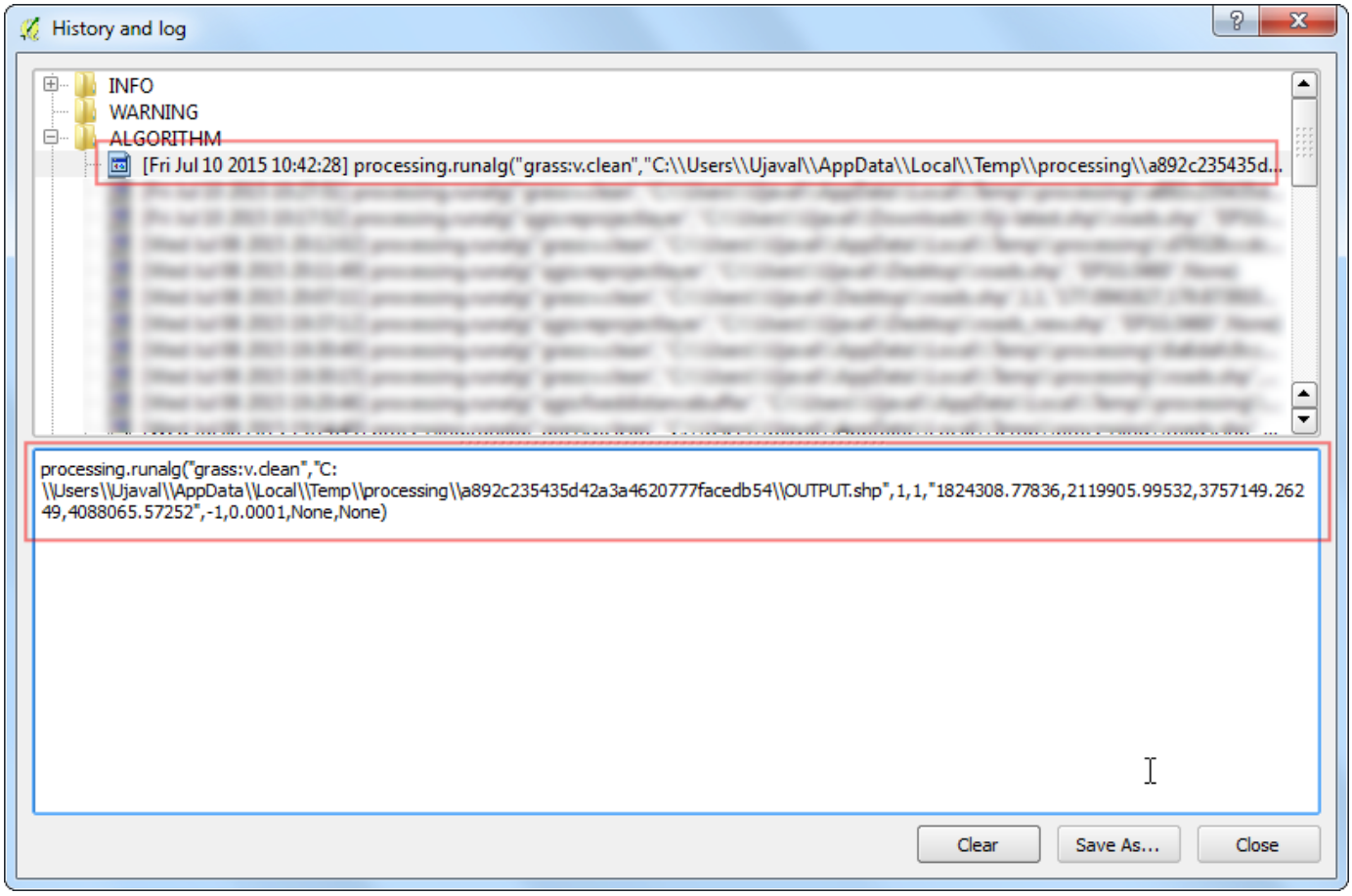
11. You can read more about various tools and options in the Help tab. For this tutorial, we will be using the `snap` tool to remove duplicate vertices that are within 1 meter of each other. Select `Reprojected layer` as the Layer to clean. Choose `snap` as the Cleaning tool. Enter `1.00` as the Threshold. Leave the other fields blank and click Run.



12. Once the processing finishes, you will see 2 new layers added to QGIS. The `Cleaned vector layer` is the layer with topological errors corrected. You will also have a `Errors layer` which will highlight the features which were repaired. You can use the errors layer as a guide and zoom in to see vertices that were removed.



13. Go to Processing ‣ History and Log dialog and note the full processing command for later use.

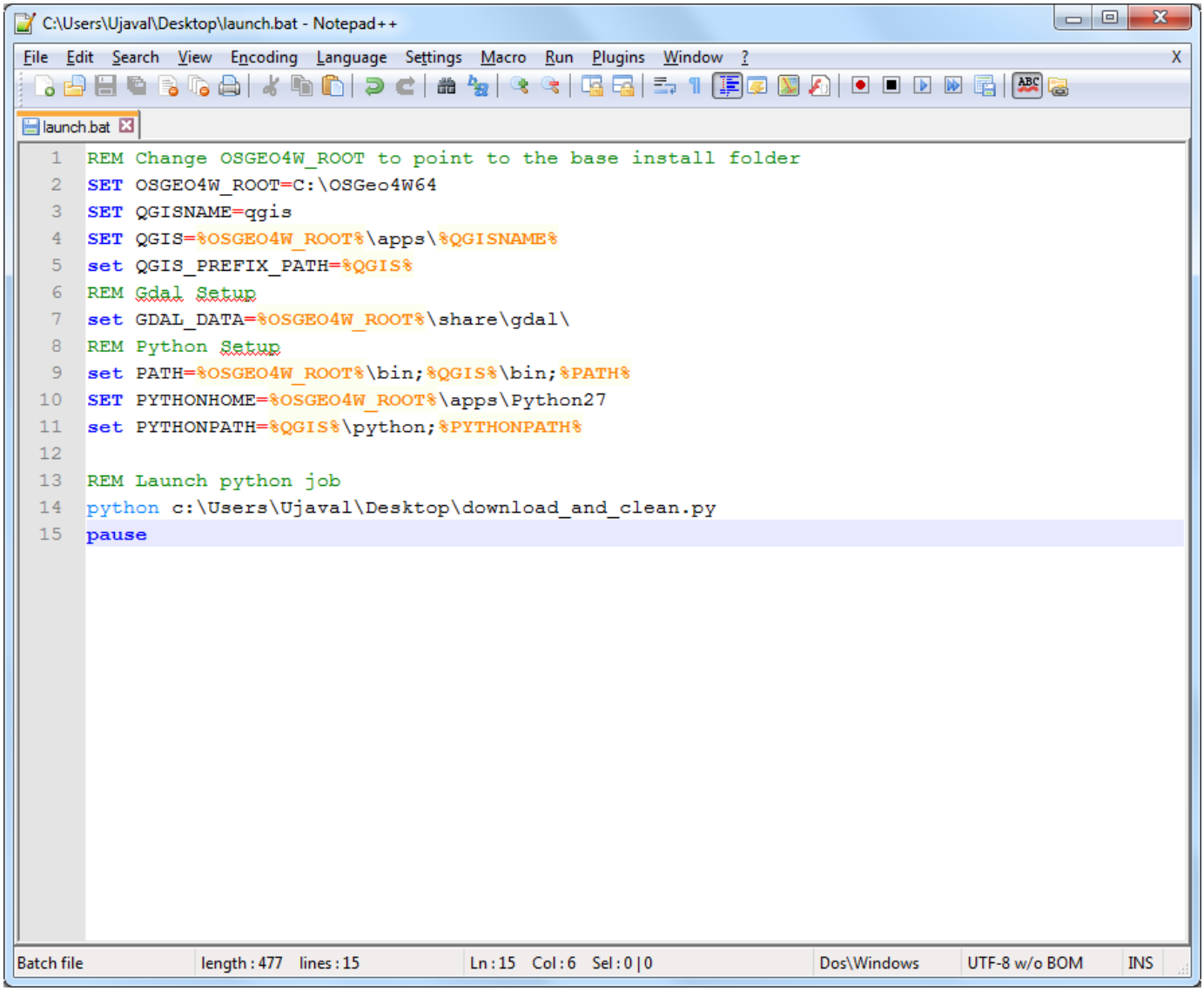


14. We are ready to start coding now. See the **A Text Editor or a Python IDE** section in the *Building a Python Plugin* (building_a_python_plugin.html) tutorial for instructions to setup your text editor or IDE. For running standalone python scripts that use QGIS, we must set various configuration options. A good way to run standalone scripts is to launch them via a `.bat` file. This file will first set the correct configuration options and then call the python script. Create a new file named `launch.bat` and enter the following text. Change the values according to your QGIS configuration. Don't forget to replace the username with your own username in the path to the python script. The paths in this file will be the same on your system if you installed QGIS via the `OSGeo4W Installer`. Save the file on your Desktop.

Note

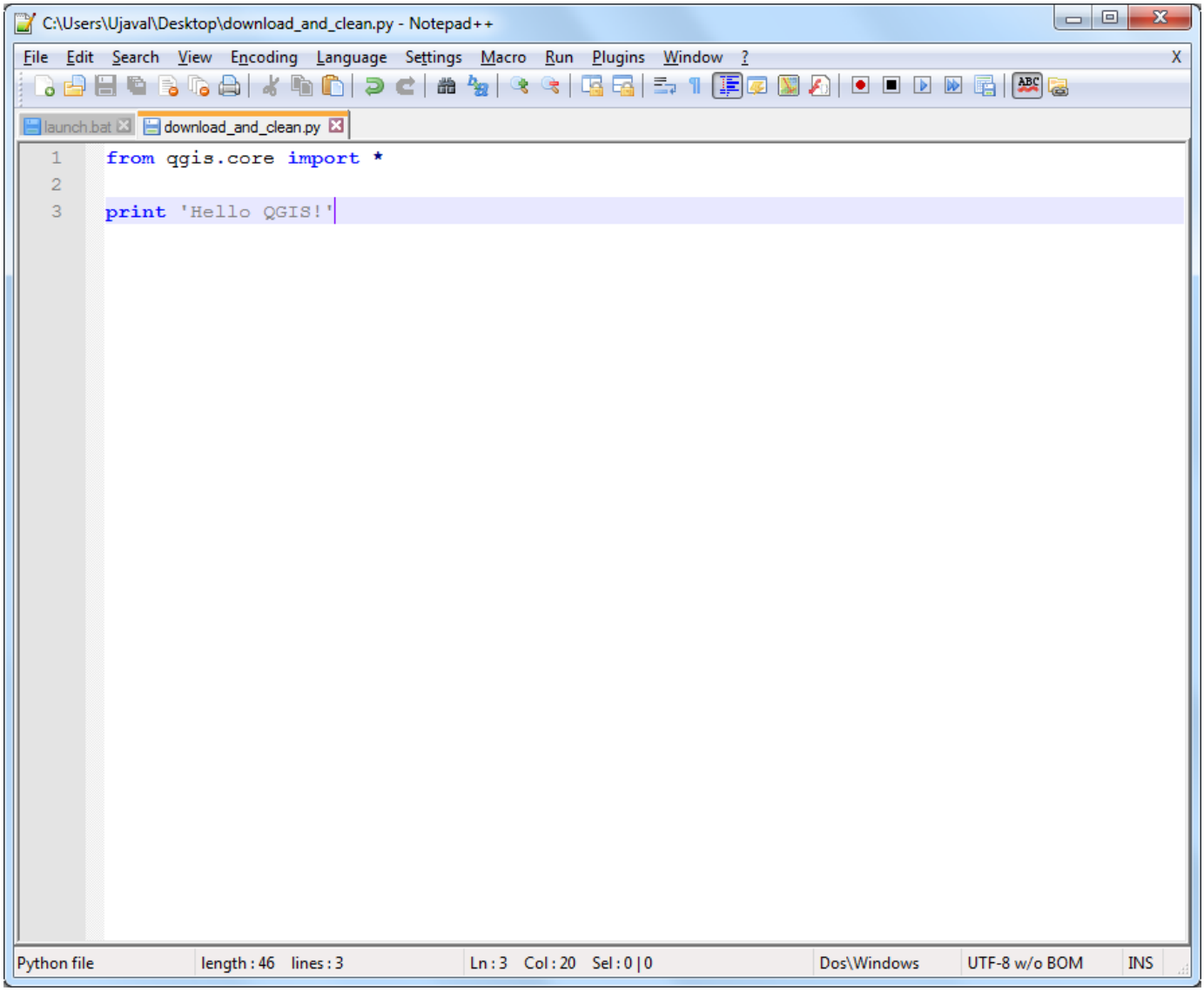
Linux and Mac users will need to create a shell script to set the paths and environment variables.

```
REM Change OSGEO4W_ROOT to point to the base install folder
SET OSGEO4W_ROOT=C:\OSGeo4W64
SET QGISNAME=qgis
SET QGIS=%OSGEO4W_ROOT%\apps\%QGISNAME%
set QGIS_PREFIX_PATH=%QGIS%
REM Gdal Setup
set GDAL_DATA=%OSGEO4W_ROOT%\share\gdal\
REM Python Setup
set PATH=%OSGEO4W_ROOT%\bin;%QGIS%\bin;%PATH%
SET PYTHONHOME=%OSGEO4W_ROOT%\apps\Python27
set PYTHONPATH=%QGIS%\python;%PYTHONPATH%

REM Launch python job
python c:\Users\Ujaval\Desktop\download_and_clean.py
pause
```

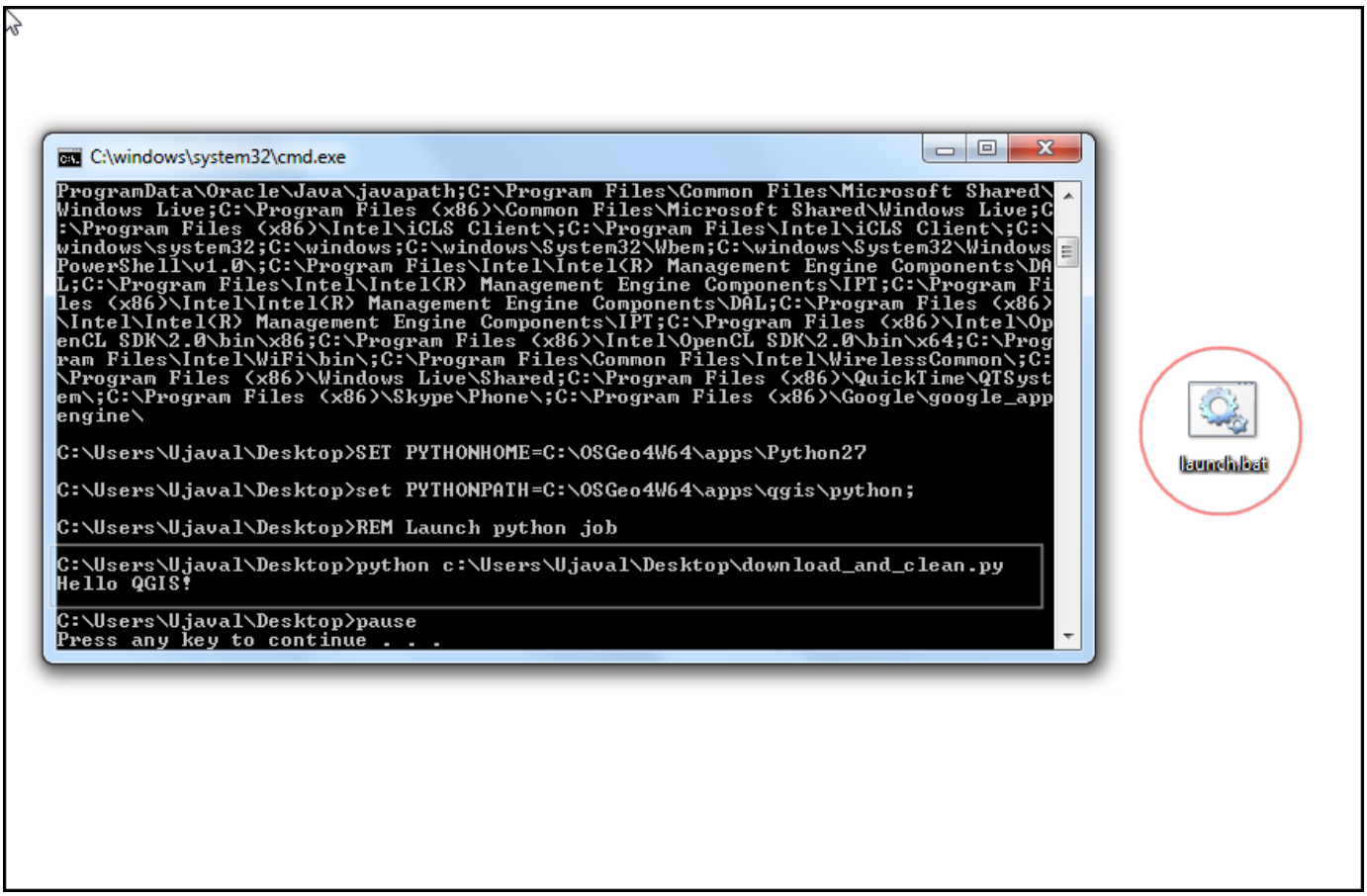


15. Create a new python file and enter the following code. Name the file as `download_and_clean.py` and save it on your Desktop.

```
from qgis.core import *
print 'Hello QGIS!'
```

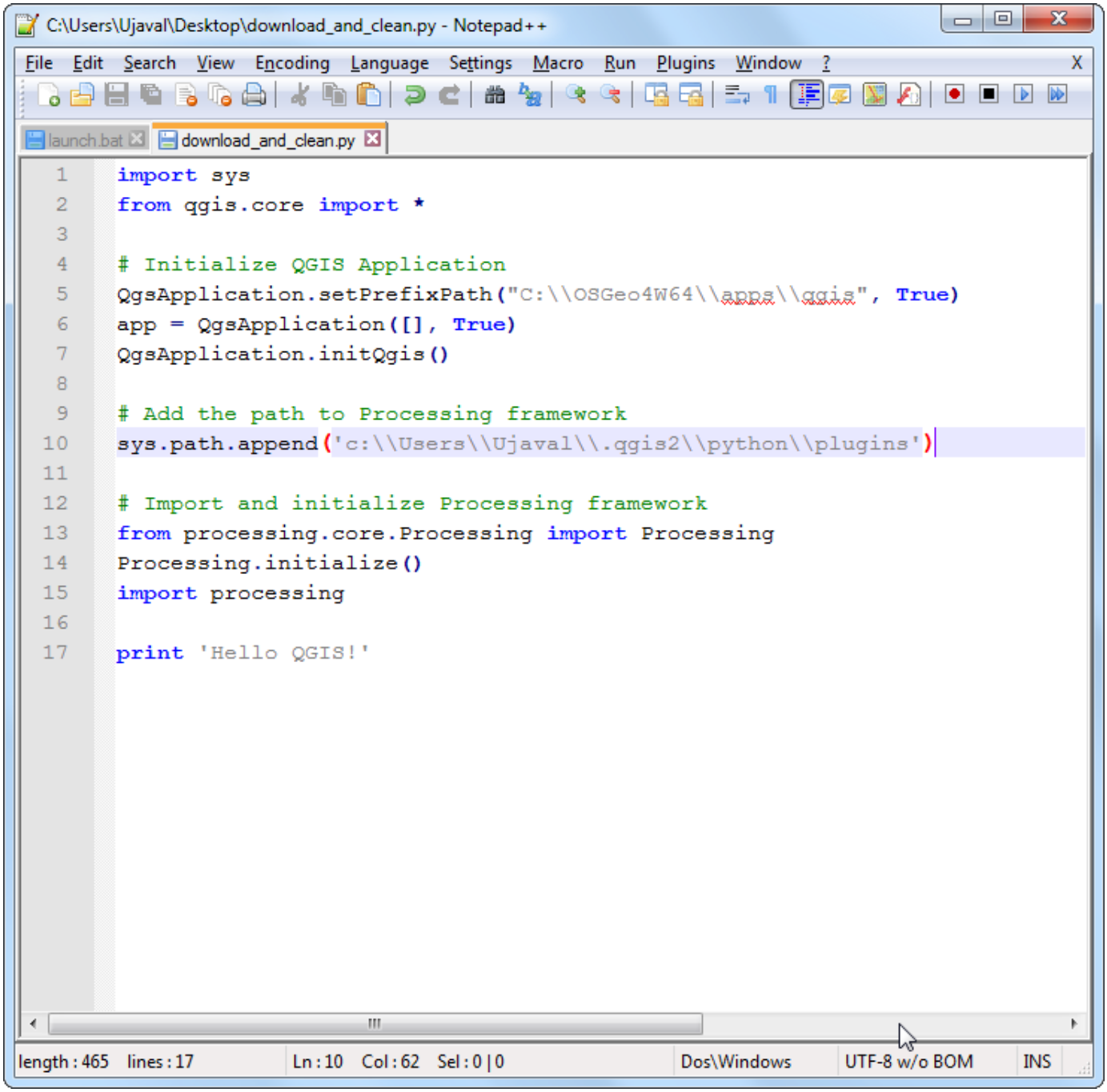


16. Switch to your Desktop and locate the `launch.bat` icon. Double-click it to launch a new command window and run the script. If you see `Hello QGIS!` printed in the command window, your configuration and setup worked fine. If you see errors or do not see the text, check your `launch.bat` file and make sure all the paths match the locations on your system.



17. Back in your text editor, modify the `download_and_clean.py` script to add the following code. This is the bootstrap code to initialize QGIS. These are unnecessary if you are running the script within QGIS. But since we are running it outside QGIS, we need to add these at the beginning. Make sure you replace the username with your username. After making these changes, save the file and run `launch.bat` again. If you see `Hello QGIS!` printed, you are all set to do add the processing logic to the script.

```
import sys
from qgis.core import *

# Initialize QGIS Application
QgsApplication.setPrefixPath("C:\\OSGeo4W64\\apps\\qgis", True)
app = QgsApplication([], True)
QgsApplication.initQgis()

# Add the path to Processing framework
sys.path.append('c:\\Users\\Ujaval\\.qgis2\\python\\plugins')

# Import and initialize Processing framework
from processing.core.Processing import Processing
Processing.initialize()
import processing

print 'Hello QGIS!'
```



18. Recall the first processing command that we had saved from the log. This was the command to re-project a layer. Paste the command to your script and add the surrounding code as follows. Note that processing commands return the path to the output layers as a dictionary. We are storing this as the `ret` value and printing the path to the reprojected layer.

```
roads_shp_path = "C:\\Users\\Ujaval\\Downloads\\fiji-latest.shp\\roads.shp"
ret = processing.runalg('qgis:reprojectlayer', roads_shp_path, 'EPSG:3460',
None)
output = ret['OUTPUT']
print output
```

C:\Users\Ujaval\Desktop\download_and_clean.py - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

launch.bat download_and_clean.py

```
1  import sys
2  from qgis.core import *
3
4  # Initialize QGIS Application
5  QgsApplication.setPrefixPath("C:\\OSGeo4W64\\apps\\qgis", True)
6  app = QgsApplication([], True)
7  QgsApplication.initQgis()
8
9  # Add the path to Processing framework
10 sys.path.append('c:\\Users\\Ujaval\\.qgis2\\python\\plugins')
11
12 # Import and initialize Processing framework
13 from processing.core.Processing import Processing
14 Processing.initialize()
15 import processing
16
17 roads_shp_path = "C:\\Users\\Ujaval\\Downloads\\fiji-latest.shp\\roads.shp"
18 ret = processing.runalg('qgis:reprojectlayer', roads_shp_path, 'EPSG:3460',
19 output = ret['OUTPUT']
20 print output
21
```

Py length : 644 lines : 21 Ln : 18 Col : 62 Sel : 0 | 0 Dos\Windows UTF-8 w/o BOM INS

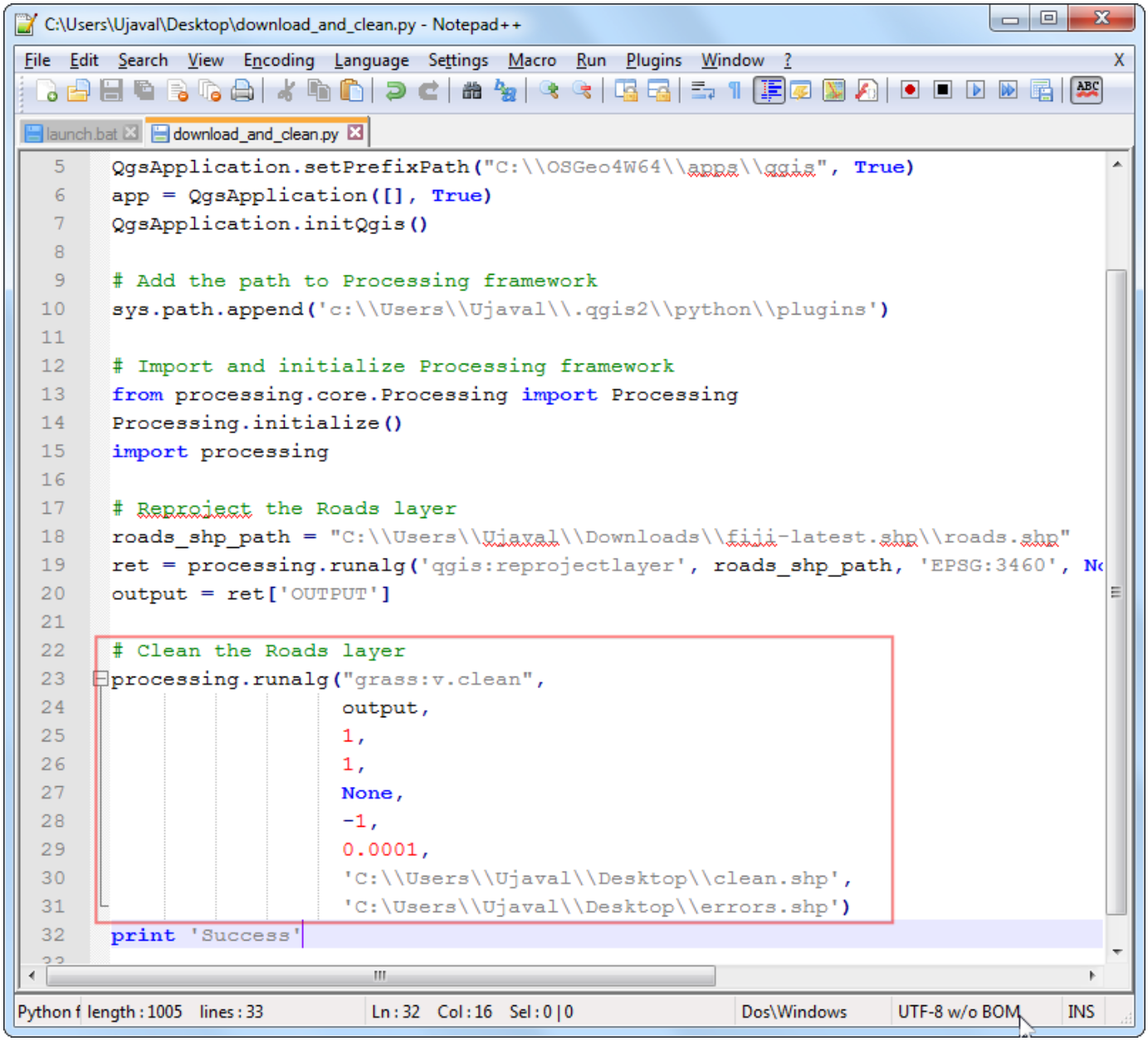
19. Run the script via `launch.bat` and you will see the path to the newly created reprojected layer.

C:\windows\system32\cmd.exe

```
Windows Live;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C
:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS Client\;C:\
windows\system32;C:\windows;C:\windows\System32\Wbem;C:\windows\System32\Windows
PowerShell\v1.0\;C:\Program Files\Intel\Intel(R) Management Engine Components\DA
L;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Fi
les (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)
\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\Intel\Op
enCL SDK\2.0\bin\x86;C:\Program Files (x86)\Intel\OpenCL SDK\2.0\bin\x64;C:\Prog
ram Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;C:
\Program Files (x86)\Windows Live\Shared;C:\Program Files (x86)\QuickTime\QTSyst
em\;C:\Program Files (x86)\Skype\Phone\;C:\Program Files (x86)\Google\google_app
engine\

C:\Users\Ujaval\Desktop>SET PYTHONHOME=C:\OSGeo4W64\apps\Python27

C:\Users\Ujaval\Desktop>set PYTHONPATH=C:\OSGeo4W64\apps\qgis\python;

C:\Users\Ujaval\Desktop>REM Launch python job

C:\Users\Ujaval\Desktop>python c:\Users\Ujaval\Desktop\download_and_clean.py
C:\Users\Ujaval\AppData\Local\Temp\processing\48a32ea5978a49a4b7ee3a775c88a7bc\O
UTPUT.shp

C:\Users\Ujaval\Desktop>pause
Press any key to continue . . .
```

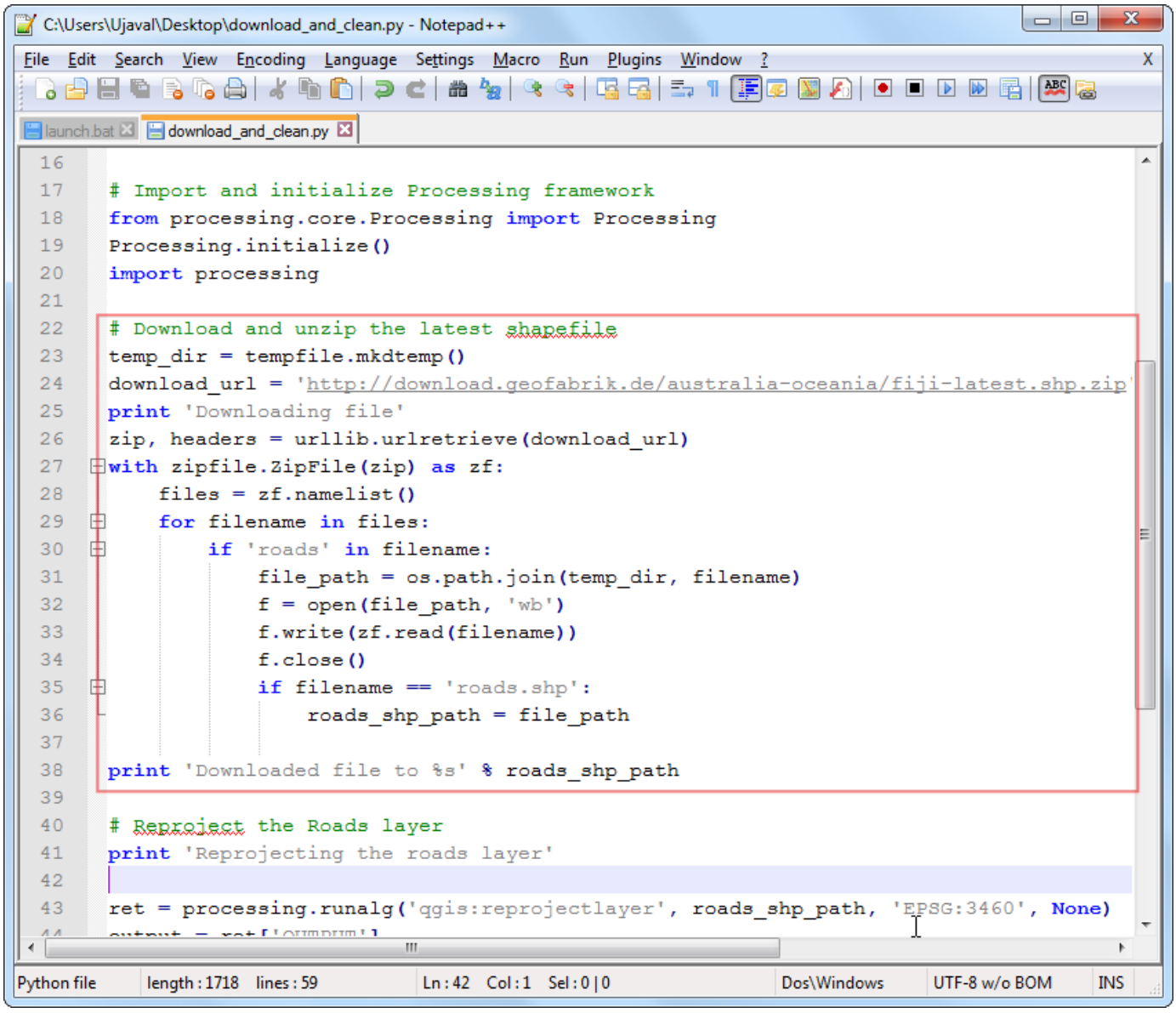
20. Now add the code for cleaning the topology. Since this is our final output, we will add the output file paths as the last 2 arguments for the `grass.v.clean` algorithm. If you left these blank, the output will be created in a temporary directory.

```
processing.runalg("grass:v.clean",
                  output,
                  1,
                  1,
                  None,
                  -1,
                  0.0001,
                  'C:\\Users\\Ujaval\\Desktop\\clean.shp',
                  'C:\Users\\Ujaval\\Desktop\\errors.shp')
```

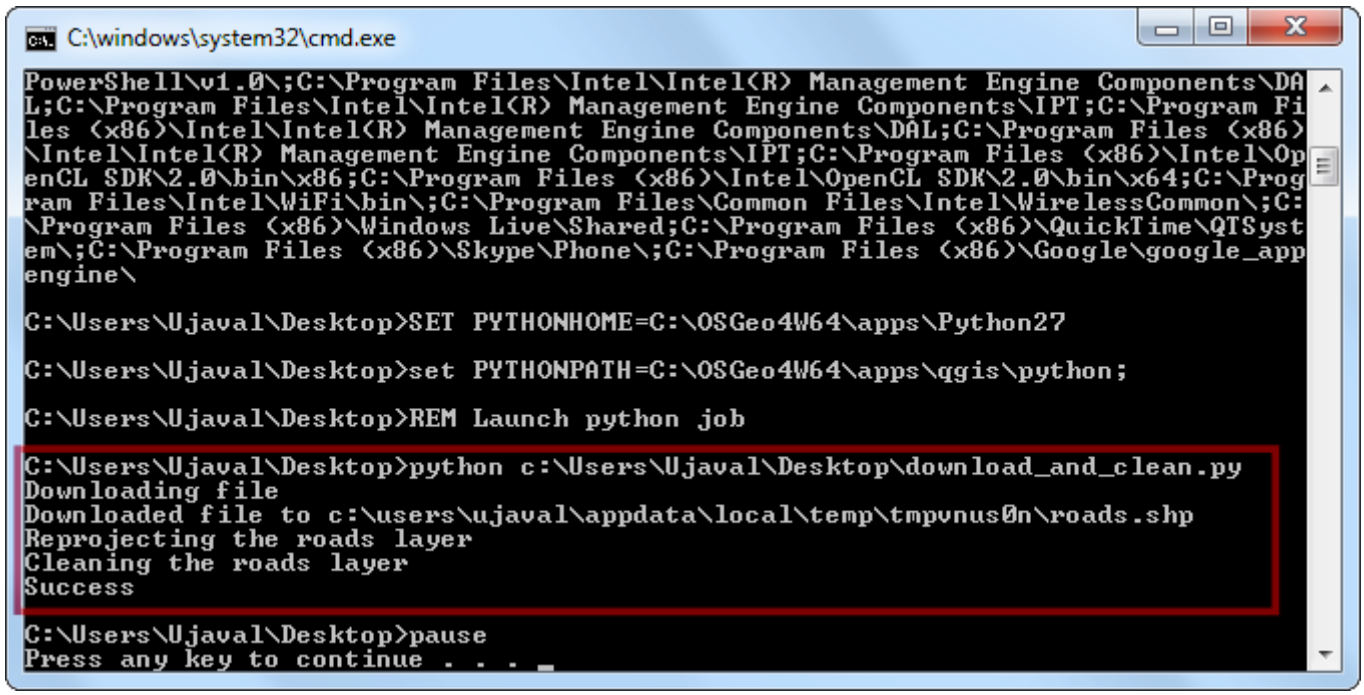


21. Run the script and you will see 2 new shapefiles created on your Desktop. This completes the processing part of the script. Let's add the code to download the data from the original website and unzip it automatically. We will also store the path to the unzipped file in a variable that we can pass to the processing algorithm later. We will need to import some additional modules for doing this. (See the end of the tutorial for the full script with all the changes)

```
import os
import urllib
import zipfile
import tempfile

temp_dir = tempfile.mkdtemp()
download_url = 'http://download.geofabrik.de/australia-oceania/fiji-latest.shp.zip'
print 'Downloading file'
zip, headers = urllib.urlretrieve(download_url)
with zipfile.ZipFile(zip) as zf:
    files = zf.namelist()
    for filename in files:
        if 'roads' in filename:
            file_path = os.path.join(temp_dir, filename)
            f = open(file_path, 'wb')
            f.write(zf.read(filename))
            f.close()
            if filename == 'roads.shp':
                roads_shp_path = file_path
```



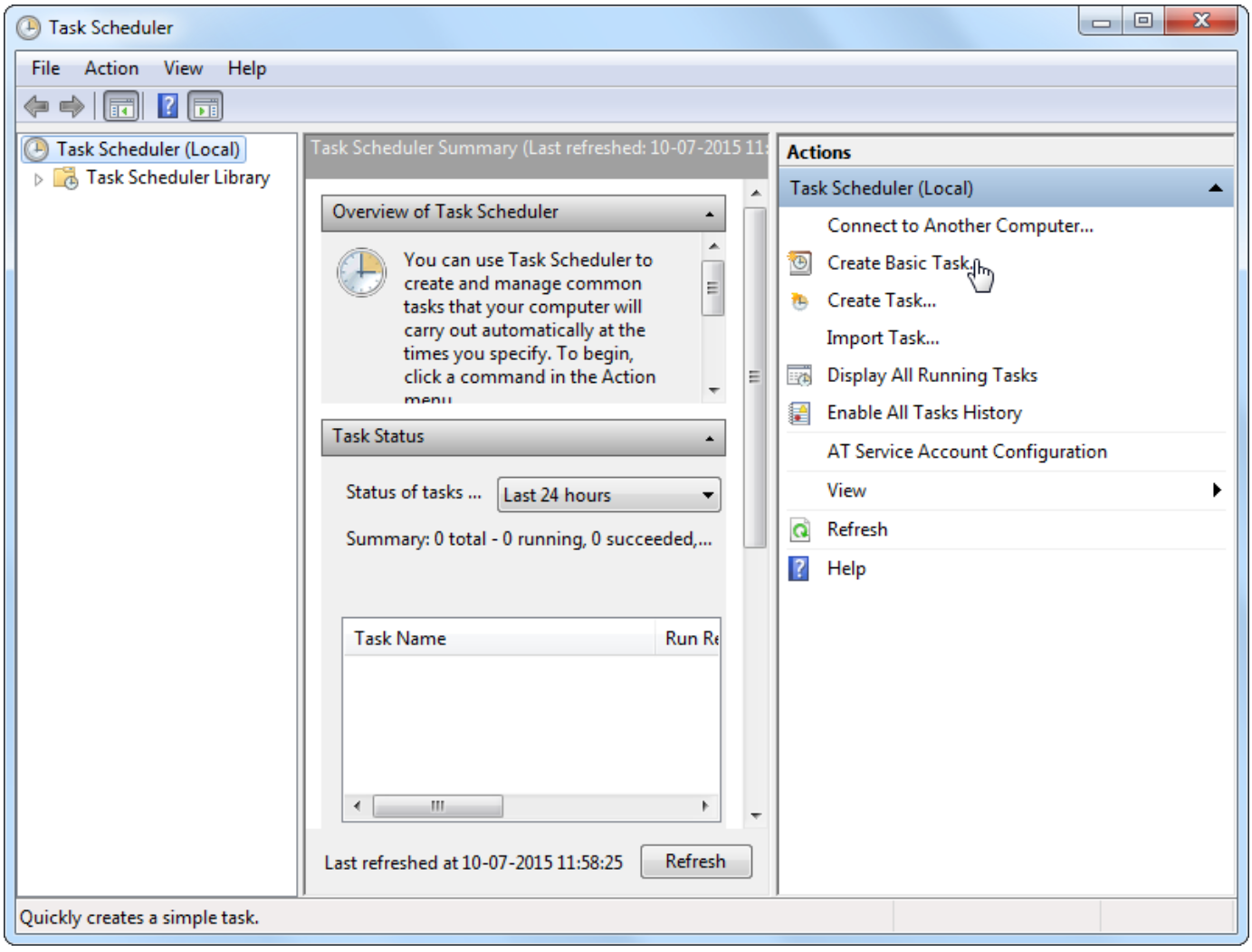
22. Run the completed script. Everytime you run the script, a fresh copy of the data will be downloaded and processed.



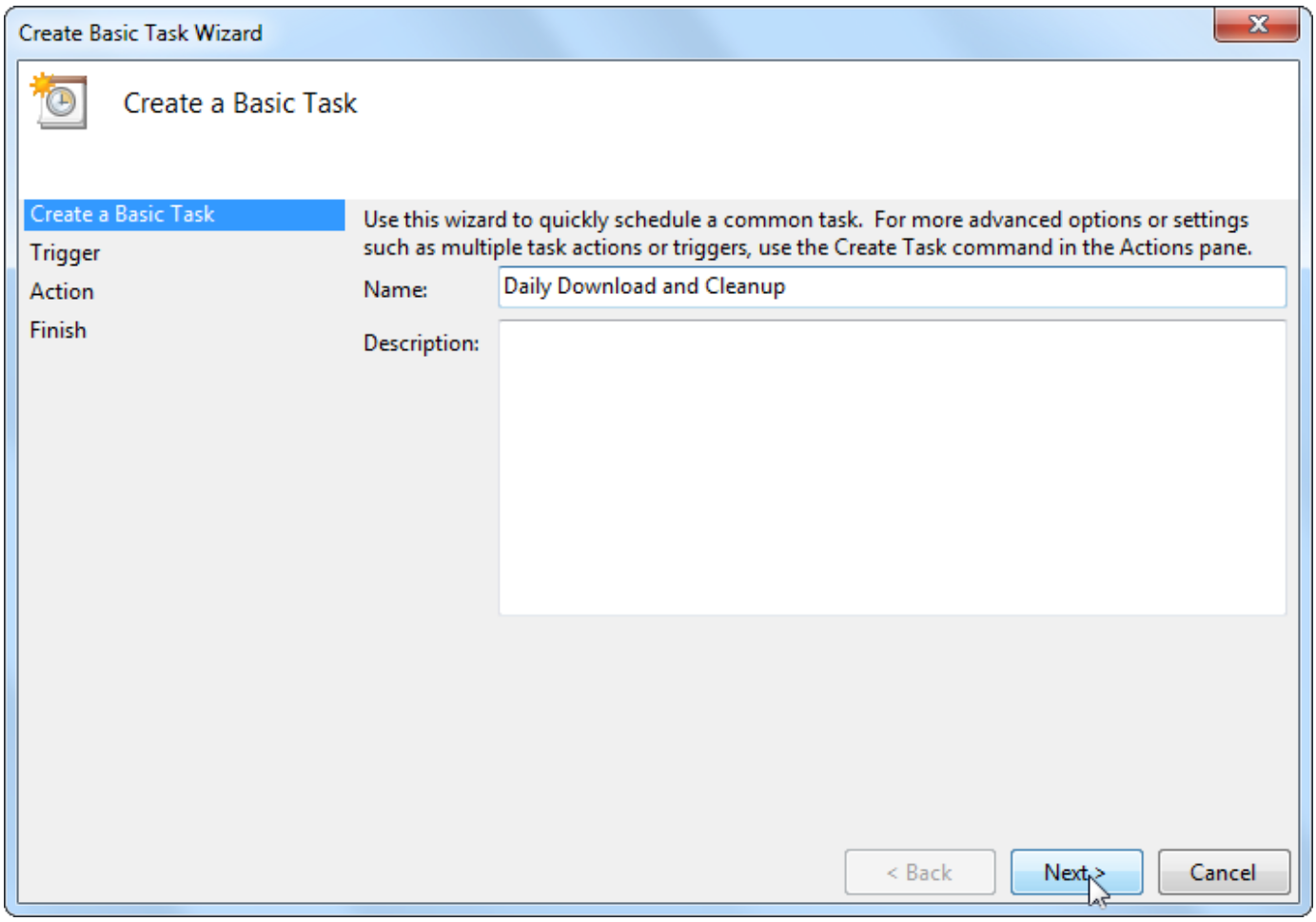
23. To automate running on this script on a daily basis, we can use the `Task Scheduler` in Windows. Launch the Task Scheduler and click Create Basic Task.

Note

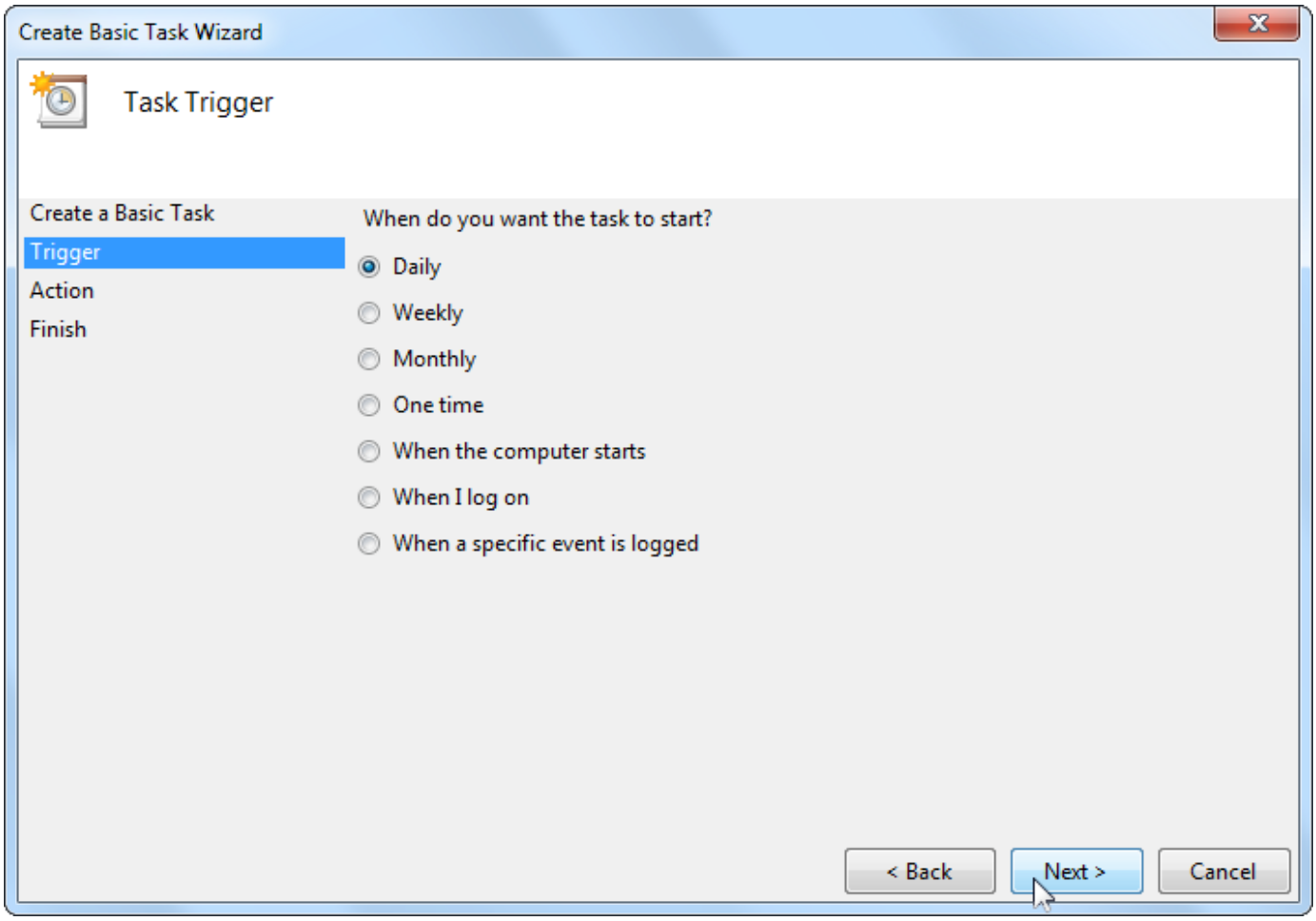
Linux and Mac users can use cron jobs to schedule tasks.



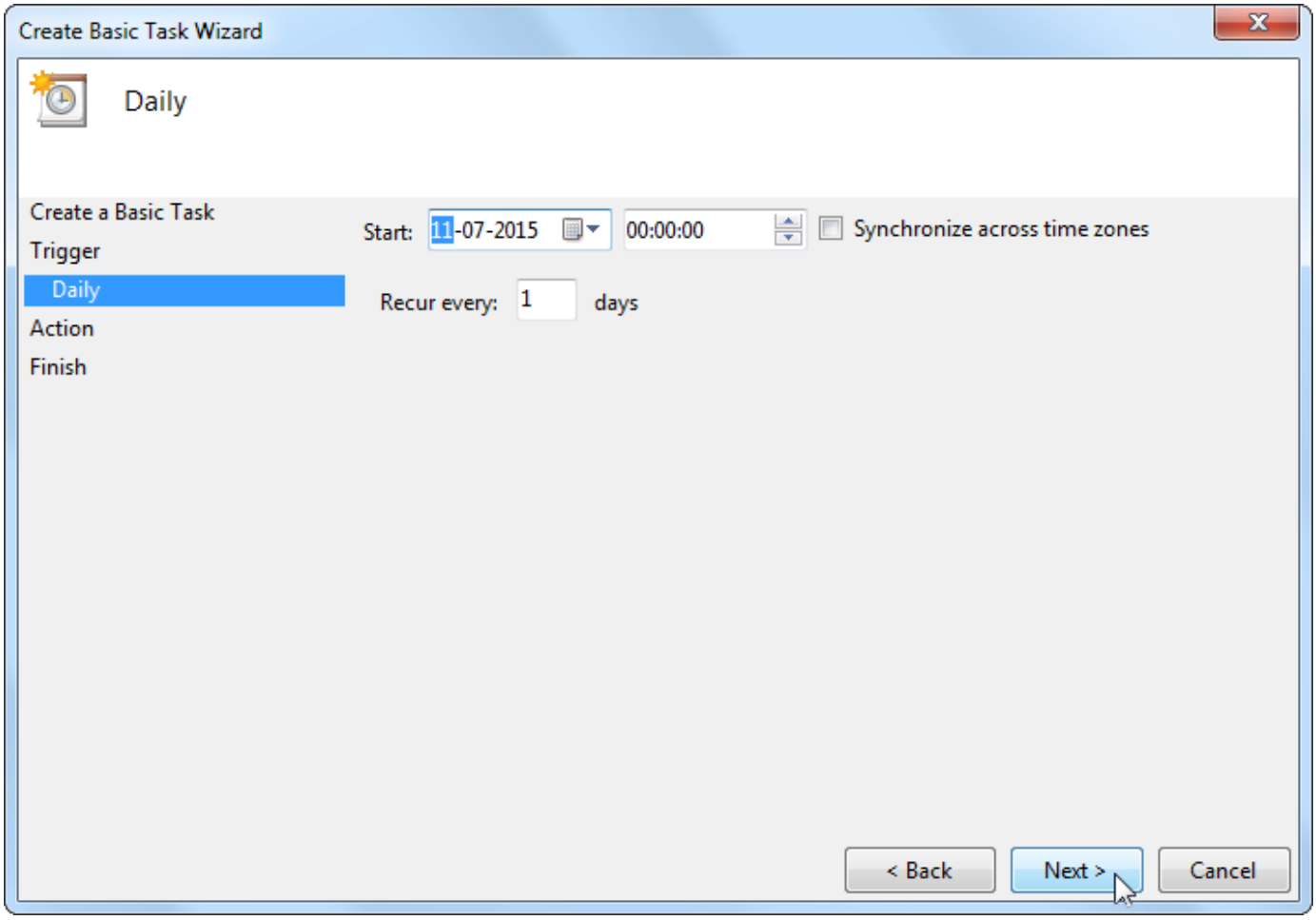
24. Name the task as `Daily Download and Cleanup` and click Next.



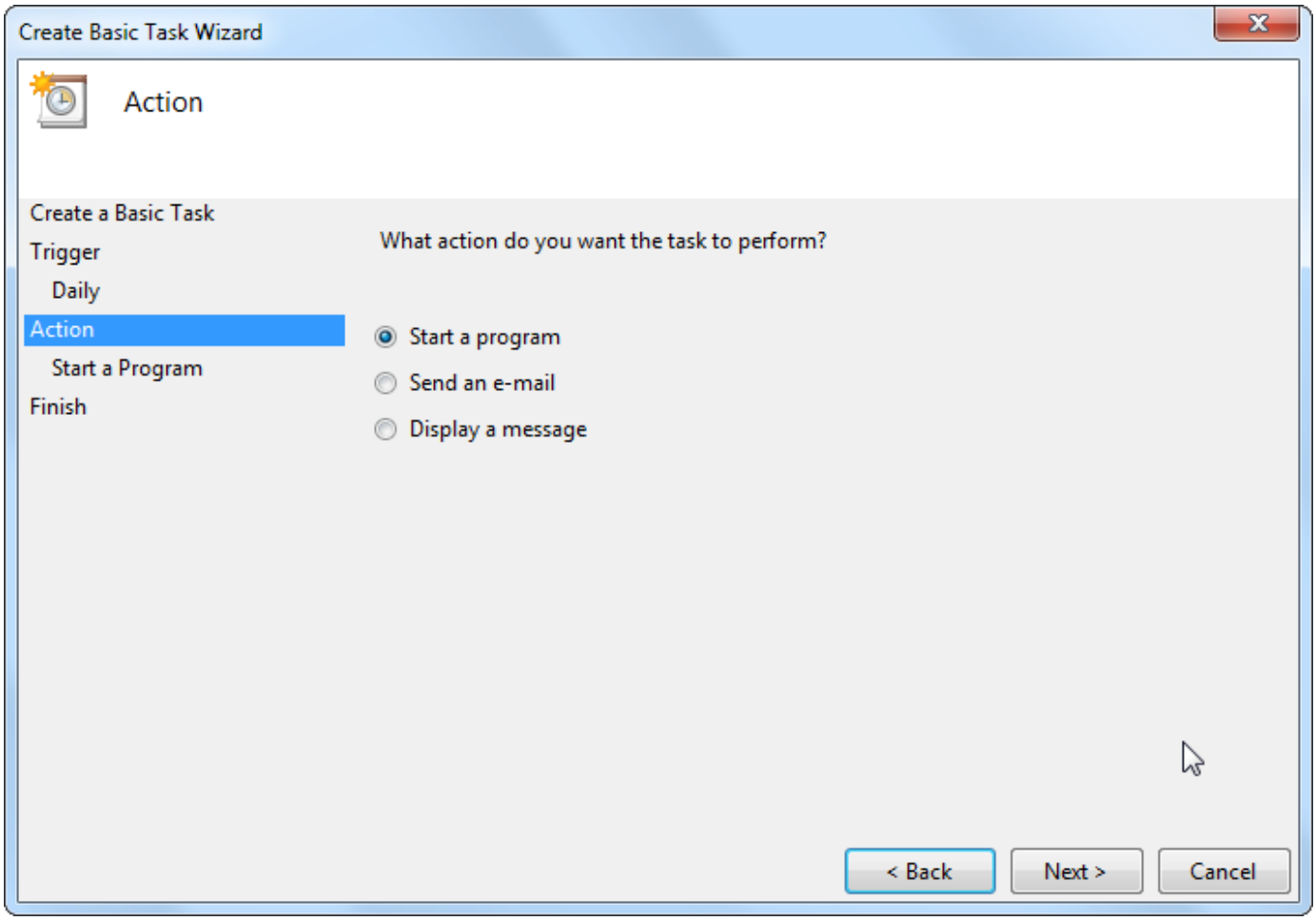
25. Select `Daily` as the Trigger and click Next



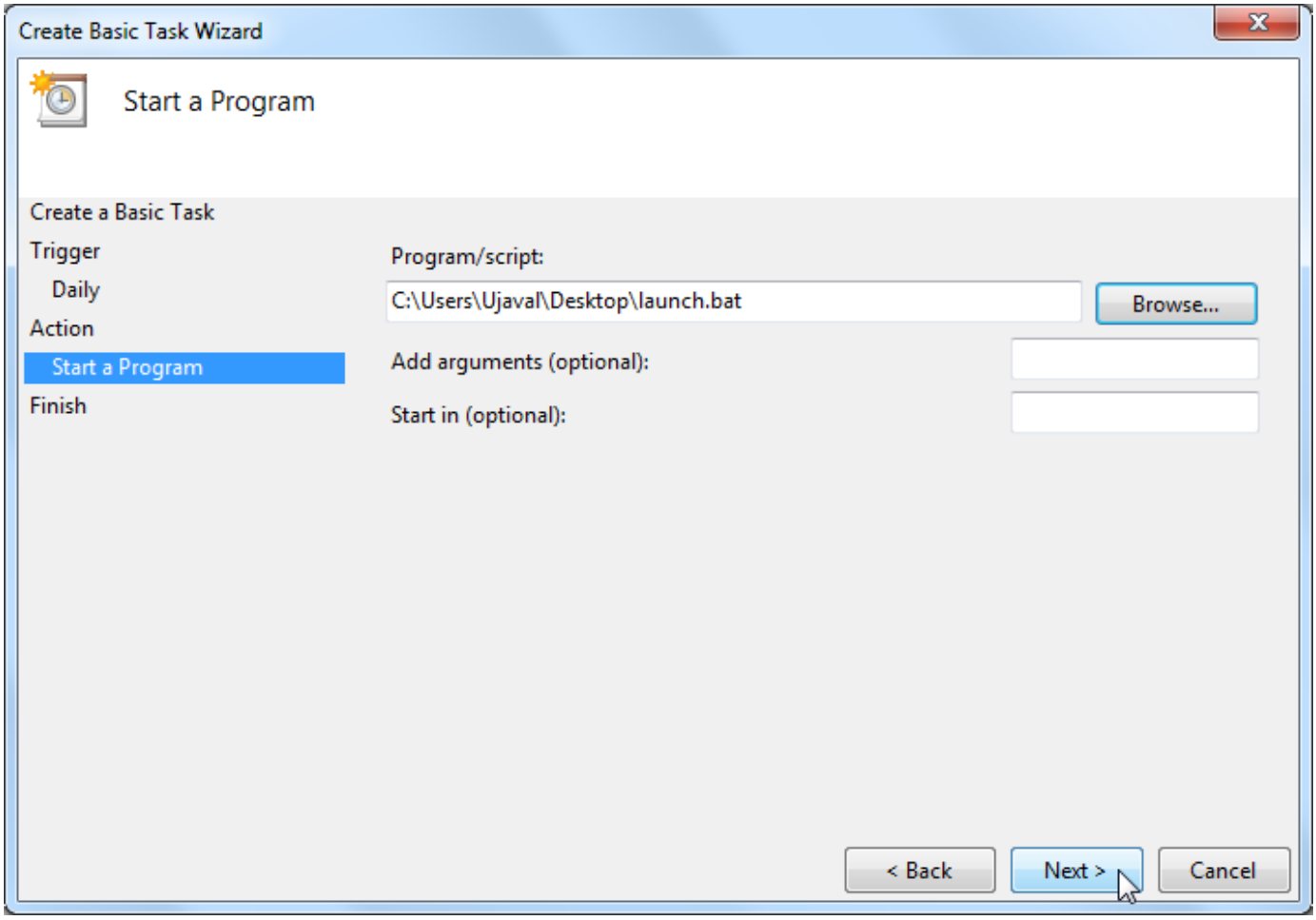
26. Select a time as per your liking and click Next.



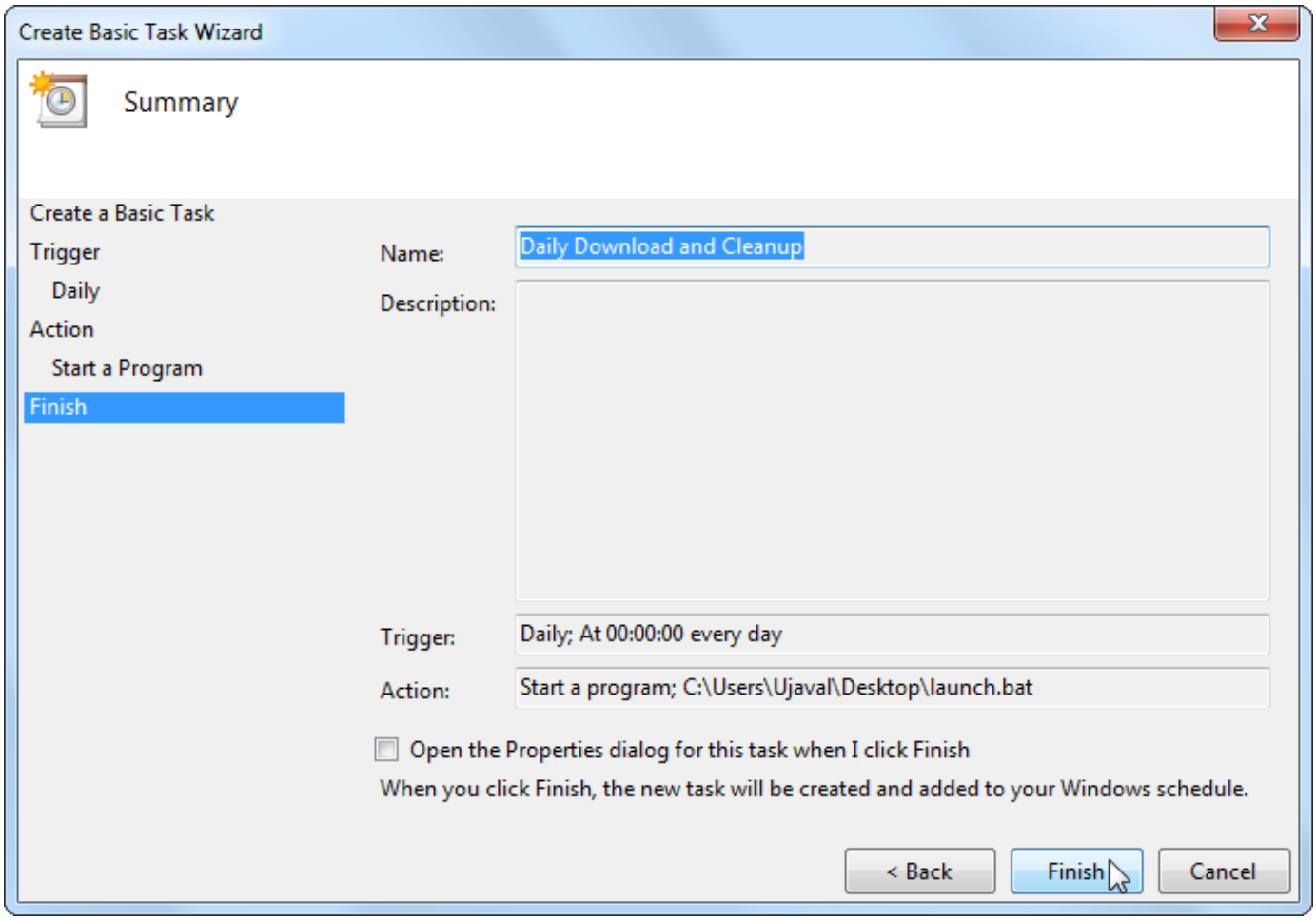
27. Choose `Start a program` as the Action and click Next.



28. Click Browse and locate the `launch.bat` script. Click Next.



29. Click Finish at the last screen to schedule the task. Now the script will automatically launch at the specified time to give you a fresh copy of cleaned data everyday.



Below is the full `download_and_clean.py` script for your reference.

```
import sys
from qgis.core import *

import os
import urllib
import zipfile
import tempfile

# Initialize QGIS Application
QgsApplication.setPrefixPath("C:\\OSGeo4W64\\apps\\qgis", True)
app = QgsApplication([], True)
QgsApplication.initQgis()

# Add the path to Processing framework
sys.path.append('c:\\Users\\Ujaval\\.qgis2\\python\\plugins')

# Import and initialize Processing framework
from processing.core.Processing import Processing
Processing.initialize()
import processing

# Download and unzip the latest shapefile
temp_dir = tempfile.mkdtemp()
download_url = 'http://download.geofabrik.de/australia-oceania/fiji-latest.shp.zip'
print 'Downloading file'
zip, headers = urllib.urlretrieve(download_url)
with zipfile.ZipFile(zip) as zf:
    files = zf.namelist()
    for filename in files:
        if 'roads' in filename:
            file_path = os.path.join(temp_dir, filename)
            f = open(file_path, 'wb')
            f.write(zf.read(filename))
            f.close()
            if filename == 'roads.shp':
                roads_shp_path = file_path

print 'Downloaded file to %s' % roads_shp_path

# Reproject the Roads layer
print 'Reprojecting the roads layer'

ret = processing.runalg('qgis:reprojectlayer', roads_shp_path, 'EPSG:3460', None)
output = ret['OUTPUT']

# Clean the Roads layer
print 'Cleaning the roads layer'

processing.runalg("grass:v.clean",
                  output,
                  1,
                  1,
                  None,
                  -1,
                  0.0001,
                  'C:\\Users\\Ujaval\\Desktop\\clean.shp',
```

```
                'C:\Users\\Ujaval\\Desktop\\errors.shp')
print 'Success'
```


Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Performing Table Joins (PyQGIS)

This tutorial shows how to use Python scripting in QGIS (PyQGIS) to perform a table join and apply a graduated style to the resulting layer. This tutorial replicates the steps of the *Performing Table Joins* (performing_table_joins.html) tutorial using only python scripting.

Overview of the task

Please refer to *Performing Table Joins* (performing_table_joins.html) tutorial for the overview.

Other skills you will learn

- Loading zipped layers in QGIS via Python.
- Using QgsGraduatedSymbolRendererV2 to apply a graduated style to a vector layer.

Get the data

Download the following files to your computer.

tl_2013_06_tract.zip (http://www.qgistutorials.com/downloads/tl_2013_06_tract.zip)

ca_tracts_pop.csv (http://www.qgistutorials.com/downloads/ca_tracts_pop.csv)

ca_tracts_pop.csvt (http://www.qgistutorials.com/downloads/ca_tracts_pop.csvt)

Data Source [TIGER] (credits.html#tiger) [USCENSUS] (credits.html#uscensus)

Procedure

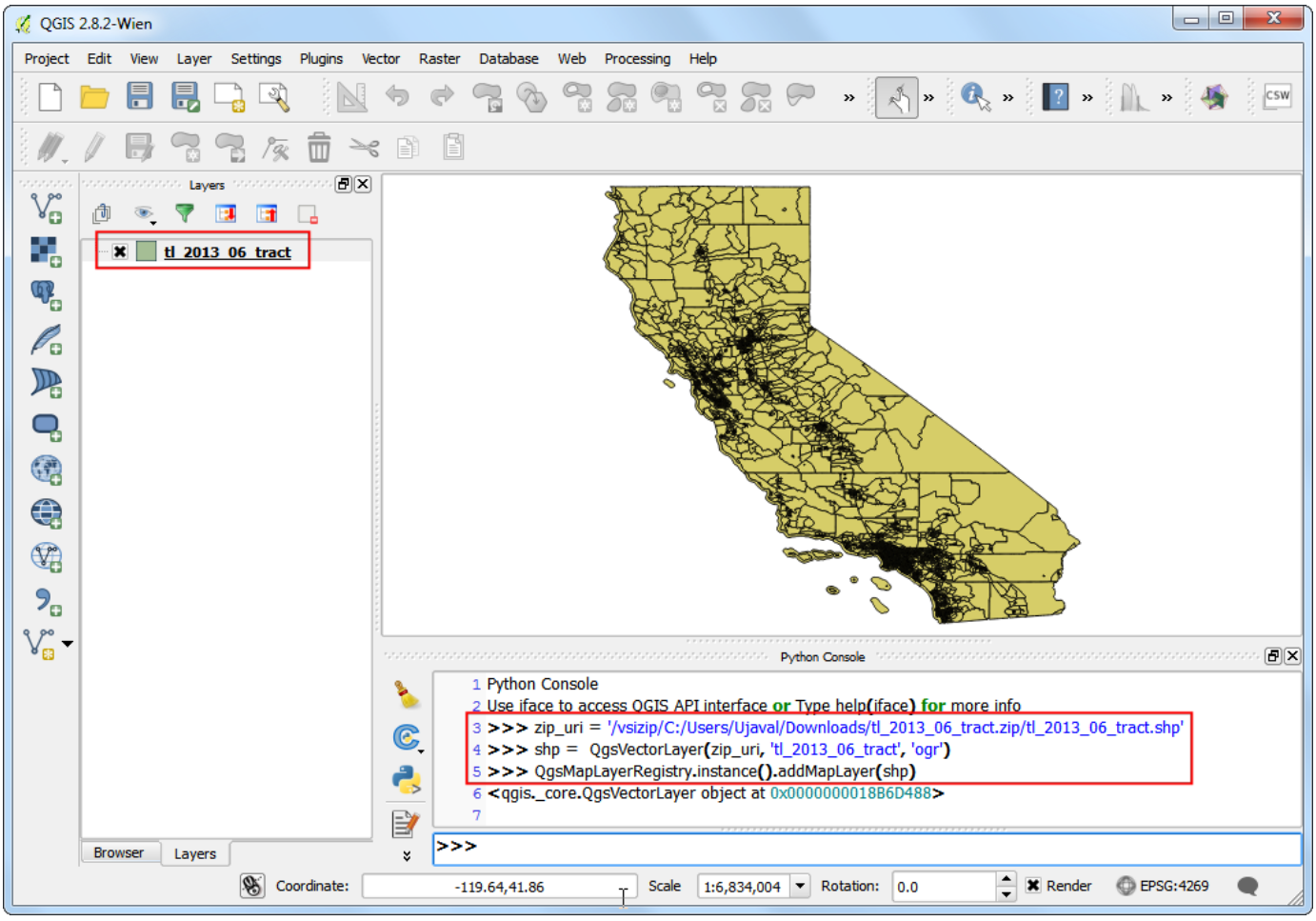
You can type the following commands in the Python Console or the built-in Editor in QGIS.

1. Load the shapefile. The Census Tracts file is a zip file containing the shapefile. While we can unzip it and load the shapefile, The OGR provider has the ability to load the zip file directly via a Virtual Filesystem. Adding `/vsizip/` in the path, we can access the shapefile contained in the zip archive.

Note

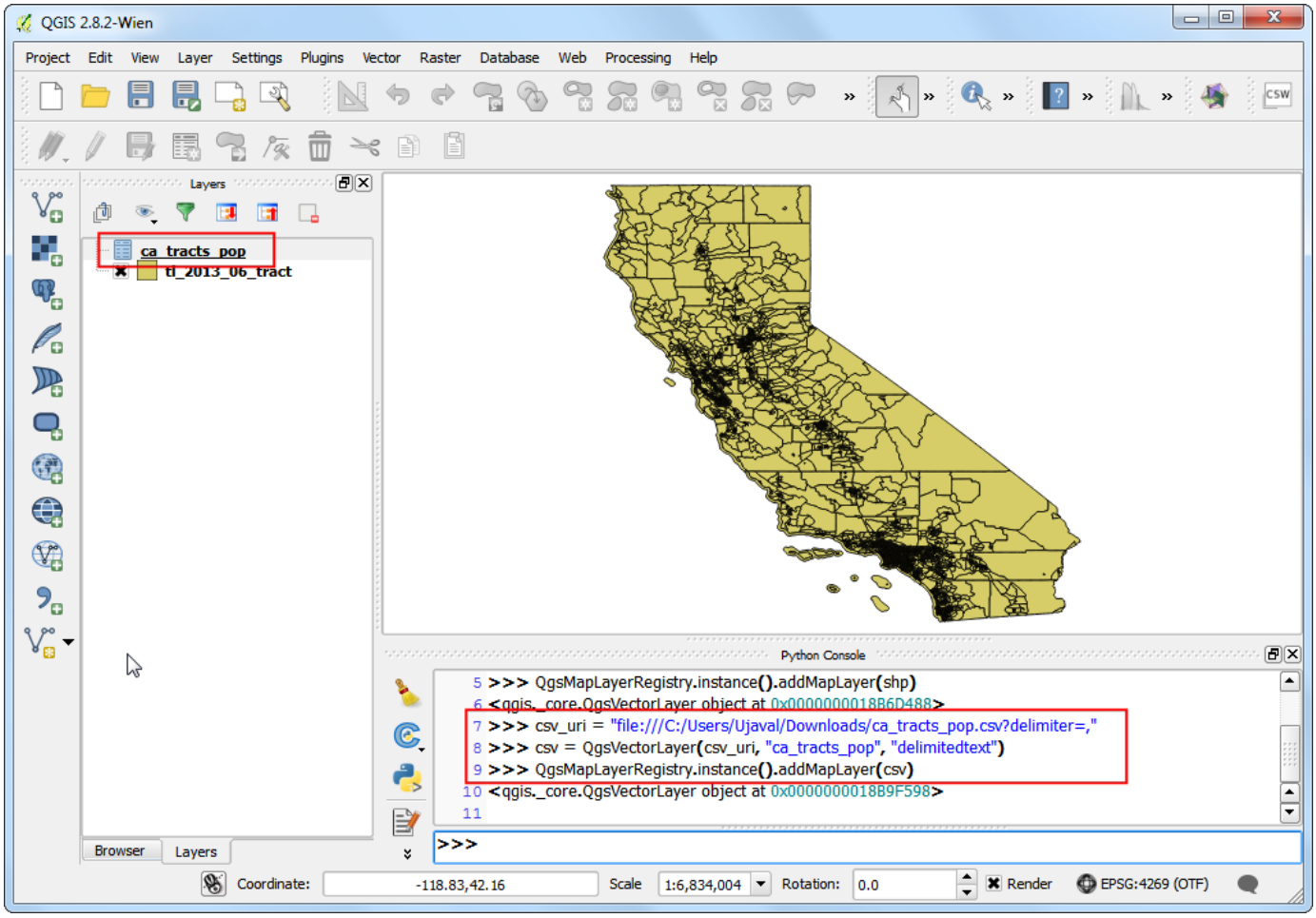
The zip_uri would begin with `/vsizip//` on Linux and Mac systems. (Note the extra /)

```
zip_uri = '/vsizip/C:/Users/Ujaval/Downloads/tl_2013_06_tract.zip'
shp =  QgsVectorLayer(zip_uri, 'tl_2013_06_tract', 'ogr')
QgsMapLayerRegistry.instance().addMapLayer(shp)
```



2. Load the CSV file. As the CSV file doesn't contain any spatial data, we load it as a table using the `delimitedtext` provider.

```
csv_uri = 'file:///C:/Users/Ujaval/Downloads/ca_tracts_pop.csv?delimiter=,'
csv = QgsVectorLayer(csv_uri, 'ca_tracts_pop', 'delimitedtext')
QgsMapLayerRegistry.instance().addMapLayer(csv)
```

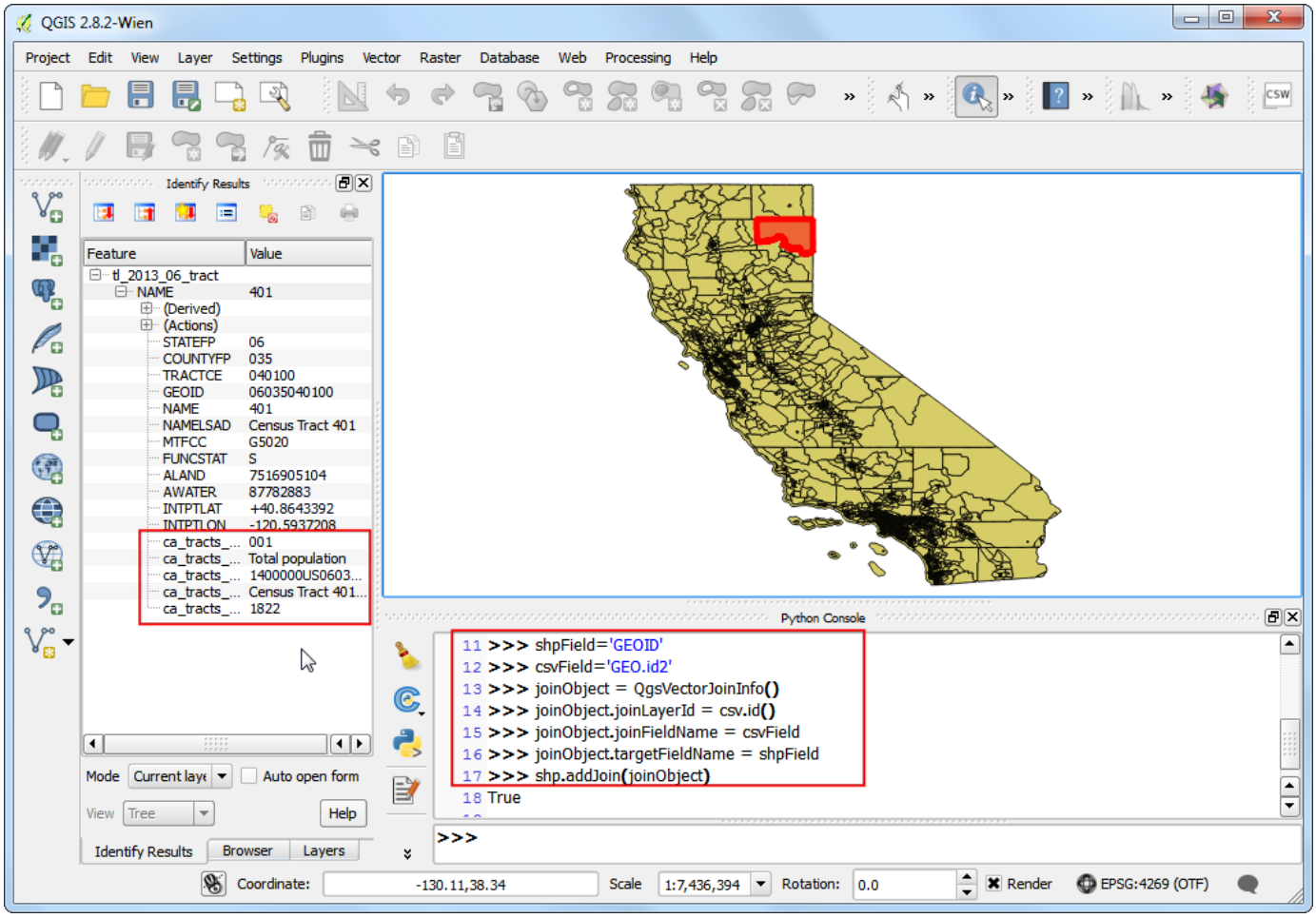


3. Create the table join. Table joins in QGIS are performed using `QgsVectorJoinInfo` object. We need to specify the `GEO.id2` field from the CSV layer as the Join Field and the `GEOID` field from the shapefile layer as the Target Field. Once you run the following code, the shapefile layer will have additional attributes joined from the csv layer.

Note

A common pitfall when using QgsVectorJoinInfo is that both the layers must be loaded in the QgsMapLayerRegistry - otherwise the join would not work.

```
shpField='GEOID'
csvField='GEO.id2'
joinObject = QgsVectorJoinInfo()
joinObject.joinLayerId = csv.id()
joinObject.joinFieldName = csvField
joinObject.targetFieldName = shpField
joinObject.memoryCache = True
shp.addJoin(joinObject)
```

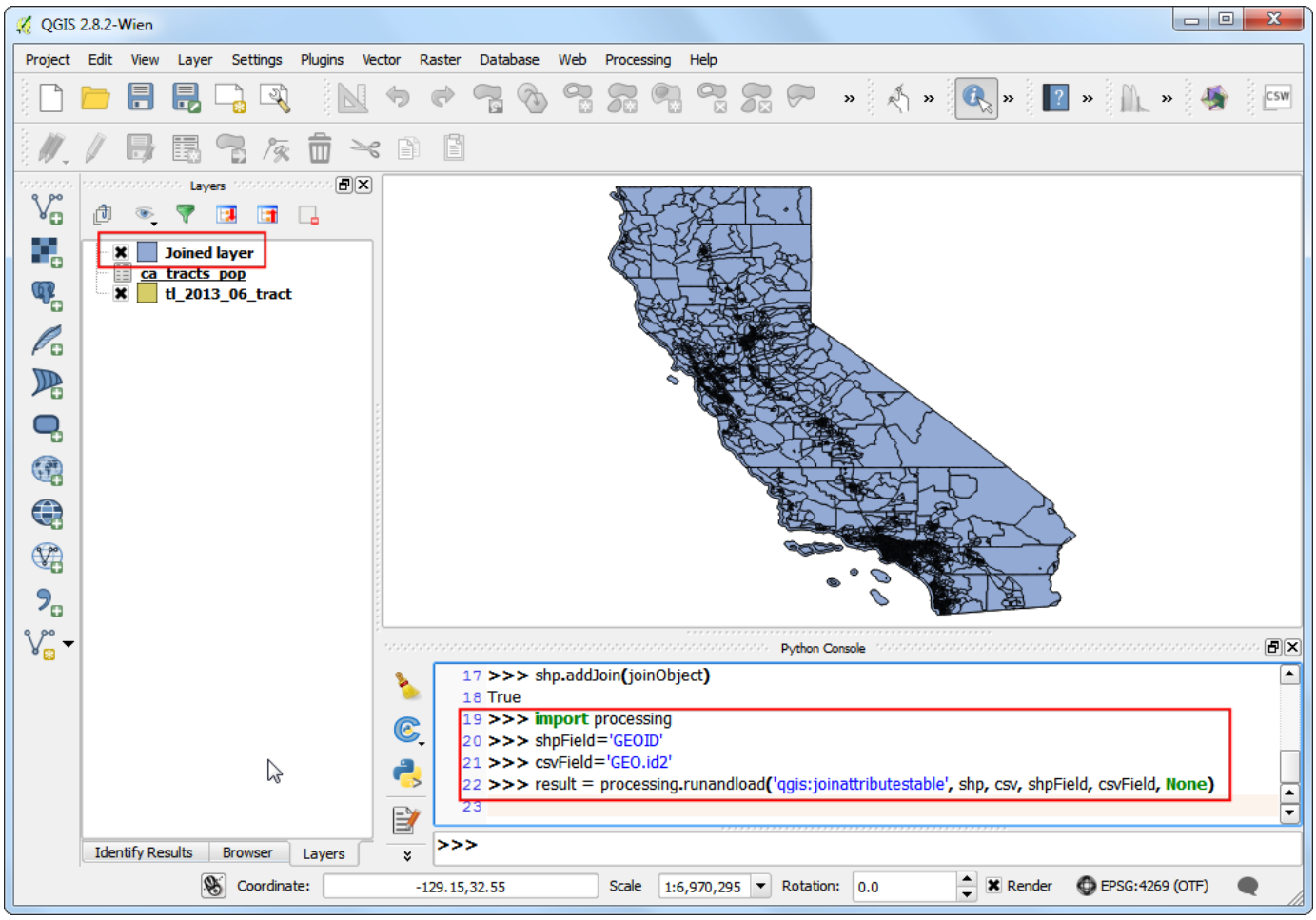


4. An easier - and preferred way of accomplishing the same thing is via the Processing Framework. You can call the algorithm `qgis:joinattributestable` and create a joined layer.

Note

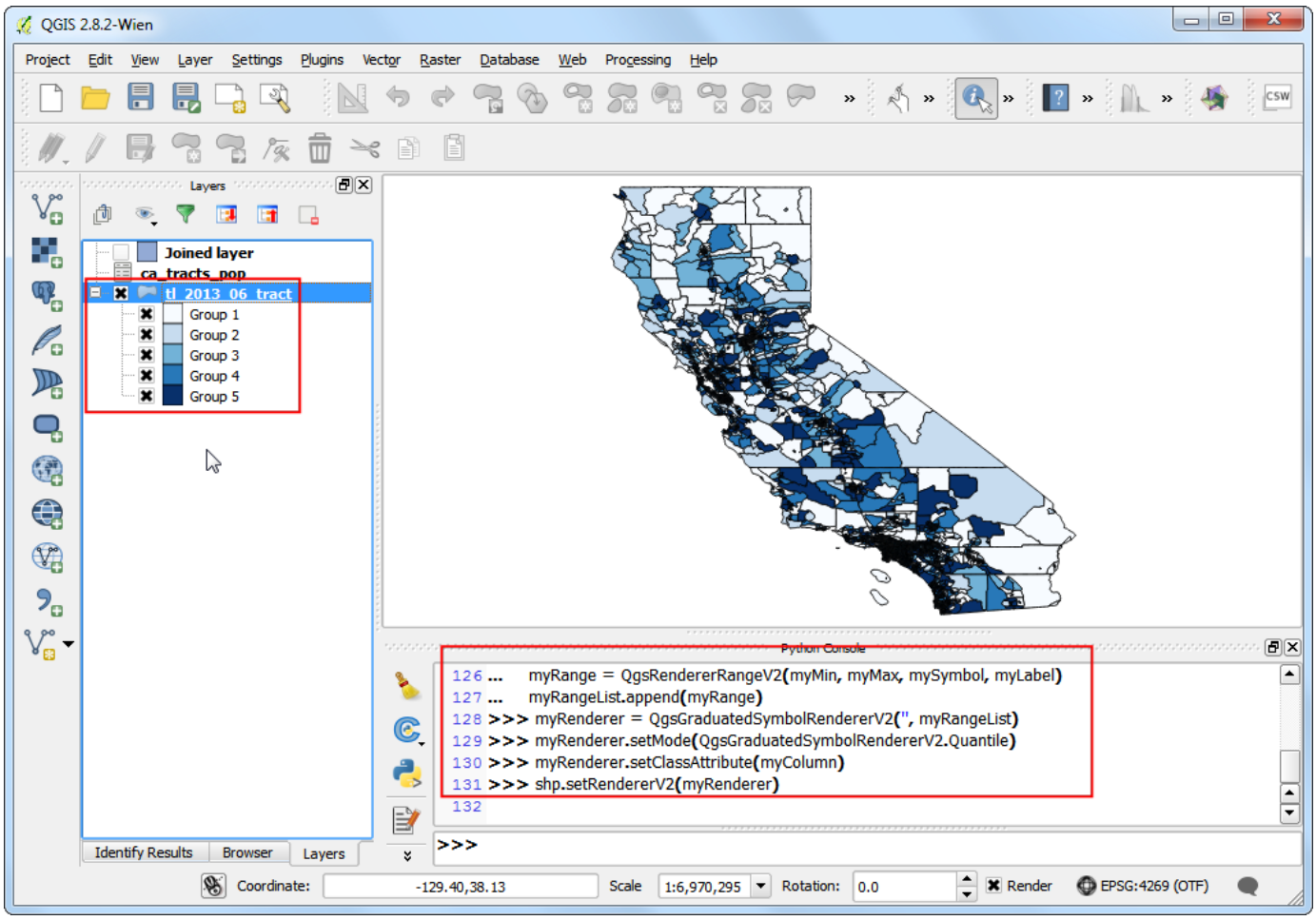
We are using the `processing.runandload()` method to execute the algorithm instead of the more common `processing.runalg()`. Since we want to load the resulting joined layer in QGIS, `processing.runandload()` is a better choice.

```
import processing
shpField='GEOID'
csvField='GEO.id2'
result = processing.runandload('qgis:joinattributestable', shp, csv, shpField, csvField, None)
```

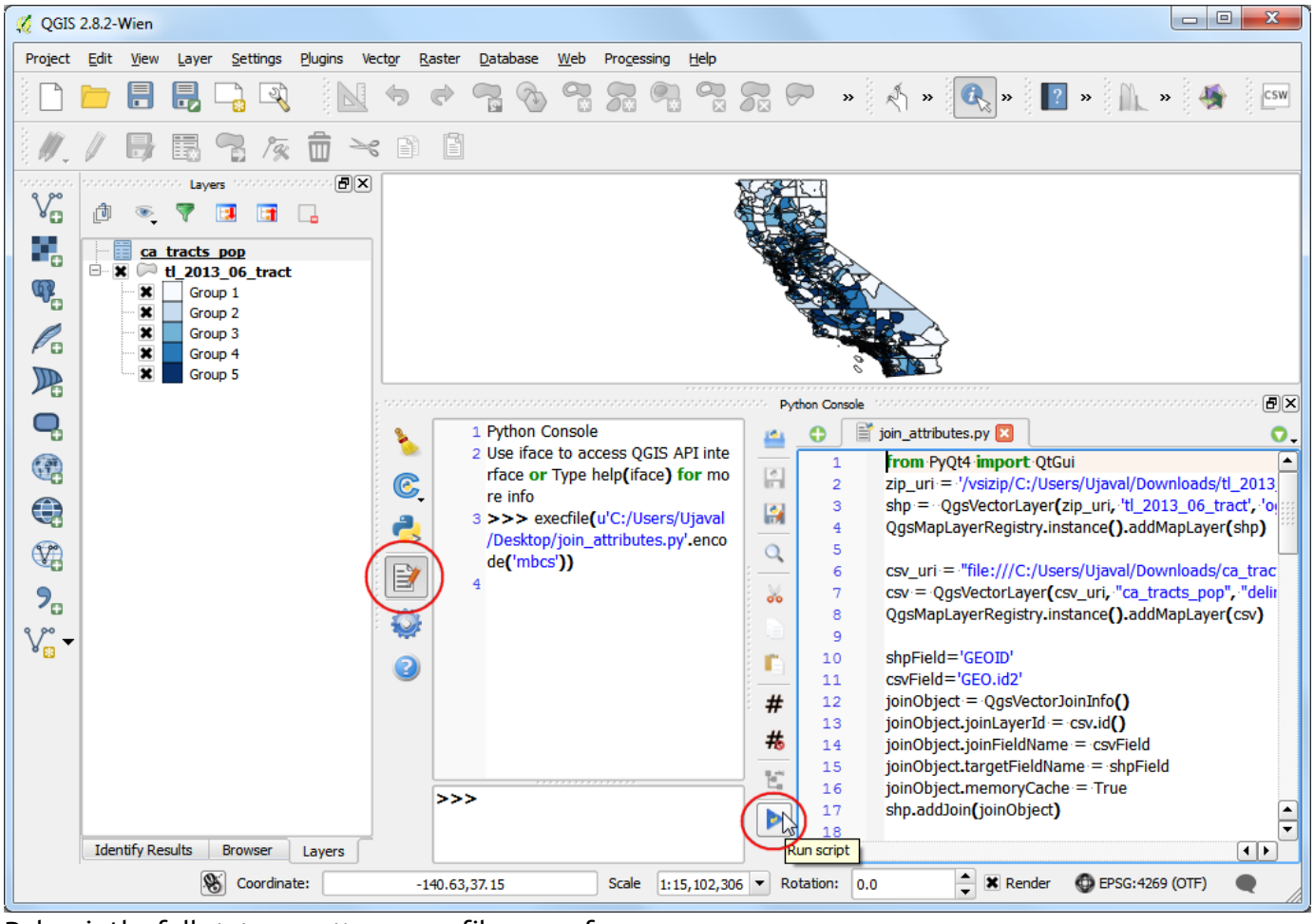


5. We will stick with the original join using `QgsVectorJoinInfo` for the remainder of the tutorial. Now it is time to apply a graduated style to the joined layer. The population field name in the joined layer is `ca_tracts_pop_D001`. We will apply a graduated renderer using the `QgsGraduatedSymbolRendererV2` class in the `Quantile` mode. Refer to *Performing Table Joins* (performing_table_joins.html) for the colors and ranges that we need to use.

```
from PyQt4 import QtGui

myColumn = 'ca_tracts_pop_D001 '
myRangeList = []
myOpacity = 1

ranges = []

myMin1 = 0.0
myMax1 = 3157.2
myLabel1 = 'Group 1'
myColor1 = QtGui.QColor('#f7fbff')
ranges.append((myMin1, myMax1, myLabel1, myColor1))

myMin2 = 3157.2
myMax2 = 4019.0
myLabel2 = 'Group 2'
myColor2 = QtGui.QColor('#c7dcef')
ranges.append((myMin2, myMax2, myLabel2, myColor2))

myMin3 = 4019.0
myMax3 = 4865.8
myLabel3 = 'Group 3'
myColor3 = QtGui.QColor('#72b2d7')
ranges.append((myMin3, myMax3, myLabel3, myColor3))

myMin4 = 4865.8
myMax4 = 5996.4
myLabel4 = 'Group 4'
myColor4 = QtGui.QColor('#2878b8')
ranges.append((myMin4, myMax4, myLabel4, myColor4))

myMin5 = 5996.4
myMax5 = 37452.0
myLabel5 = 'Group 5'
myColor5 = QtGui.QColor('#08306b')
ranges.append((myMin5, myMax5, myLabel5, myColor5))

for myMin, myMax, myLabel, myColor in ranges:
  mySymbol = QgsSymbolV2.defaultSymbol(shp.geometryType())
  mySymbol.setColor(myColor)
  mySymbol.setAlpha(myOpacity)
  myRange = QgsRendererRangeV2(myMin, myMax, mySymbol, myLabel)
  myRangeList.append(myRange)

myRenderer = QgsGraduatedSymbolRendererV2('', myRangeList)
myRenderer.setMode(QgsGraduatedSymbolRendererV2.Quantile)
myRenderer.setClassAttribute(myColumn)

shp.setRendererV2(myRenderer)
```



6. Typing the code in the Python Console is useful for small tasks, but it is far easier to use the built-in Editor. You can copy the entire script in the Editor and click Run. As the script finishes, you would have created a table join and styled the resulting layer without any manual steps.



Below is the full `join_attributes.py` file as a reference.

```
from PyQt4 import QtGui
zip_uri = '/vsizip/C:/Users/Ujaval/Downloads/tl_2013_06_tract.zip/tl_2013_06_tract.shp'
shp =  QgsVectorLayer(zip_uri, 'tl_2013_06_tract', 'ogr')
QgsMapLayerRegistry.instance().addMapLayer(shp)

csv_uri = "file:///C:/Users/Ujaval/Downloads/ca_tracts_pop.csv?delimiter=,"
csv = QgsVectorLayer(csv_uri, "ca_tracts_pop", "delimitedtext")
QgsMapLayerRegistry.instance().addMapLayer(csv)

shpField='GEOID'
csvField='GEO.id2'
joinObject = QgsVectorJoinInfo()
joinObject.joinLayerId = csv.id()
joinObject.joinFieldName = csvField
joinObject.targetFieldName = shpField
joinObject.memoryCache = True
shp.addJoin(joinObject)

myColumn = 'ca_tracts_pop_D001 '
myRangeList = []
myOpacity = 1

ranges = []

myMin1 = 0.0
myMax1 = 3157.2
myLabel1 = 'Group 1'
myColor1 = QtGui.QColor('#f7fbff')
ranges.append((myMin1, myMax1, myLabel1, myColor1))

myMin2 = 3157.2
myMax2 = 4019.0
myLabel2 = 'Group 2'
myColor2 = QtGui.QColor('#c7dcef')
ranges.append((myMin2, myMax2, myLabel2, myColor2))

myMin3 = 4019.0
myMax3 = 4865.8
myLabel3 = 'Group 3'
myColor3 = QtGui.QColor('#72b2d7')
ranges.append((myMin3, myMax3, myLabel3, myColor3))

myMin4 = 4865.8
myMax4 = 5996.4
myLabel4 = 'Group 4'
myColor4 = QtGui.QColor('#2878b8')
ranges.append((myMin4, myMax4, myLabel4, myColor4))

myMin5 = 5996.4
myMax5 = 37452.0
myLabel5 = 'Group 5'
myColor5 = QtGui.QColor('#08306b')
ranges.append((myMin5, myMax5, myLabel5, myColor5))

for myMin, myMax, myLabel, myColor in ranges:
    mySymbol = QgsSymbolV2.defaultSymbol(shp.geometryType())
    mySymbol.setColor(myColor)
```

```
    mySymbol.setAlpha(myOpacity)
    myRange = QgsRendererRangeV2(myMin, myMax, mySymbol, myLabel)
    myRangeList.append(myRange)

myRenderer = QgsGraduatedSymbolRendererV2('', myRangeList)
myRenderer.setMode(QgsGraduatedSymbolRendererV2.Quantile)
myRenderer.setClassAttribute(myColumn)

shp.setRendererV2(myRenderer)
```


Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Looking for QGIS Training? Visit spatialthoughts.com (https://spatialthoughts.com) for more resources and contact information.

Web Mapping with QGIS2Web

Web mapping is a great medium to publish your GIS data to ther web and make it accessible by other users. Creating a web map is a very different process than creating one in a GIS. GIS users are typically aren't web programmers and it presents a challenge when one needs to create a web map that is of the same quality as a map creating in a GIS. Fortunately, there are tools available to easily translate your work in QGIS to web maps. In this tutorial, you will learn how to use the **QGIS2Web** plugin to create a web map using OpenLayers or Leaflet lubraries from your QGIS project.

Overview of the task

We will create a openlayers web map of world's airports.

Other skills you will learn

- How to use Edit Widgets in QGIS to hide certain fields and set custom types.
- How to create a virtual field using Field Calculator.
- Creating labels for features that appear only at certain scale.

Get the data

We will use the Airports (http://www.naturalearthdata.com/downloads/10m-cultural-vectors/airports/) dataset from Natural Earth.

Download the Airports shapefile (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural/ne_10m_airports.zip).

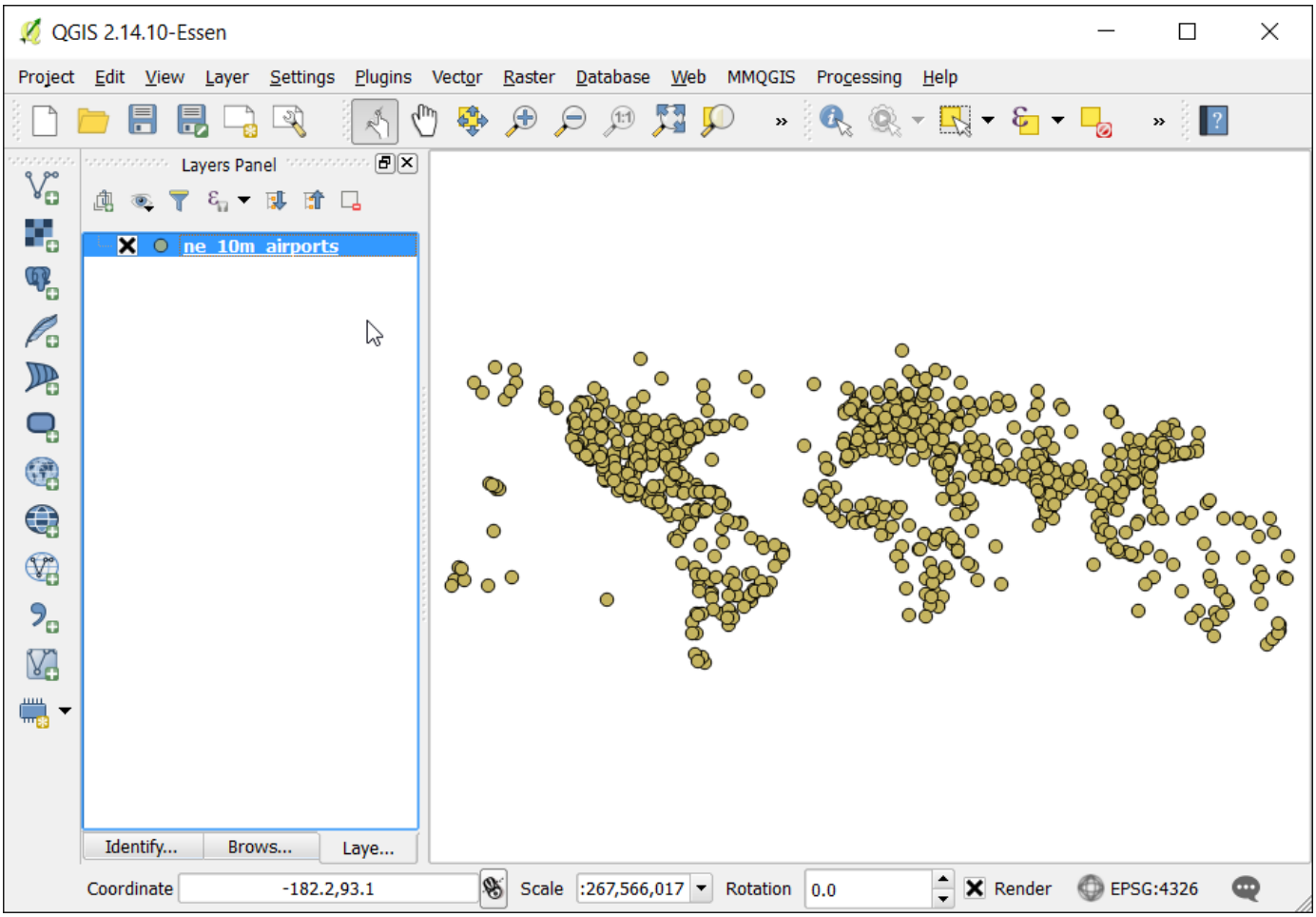
For convenience, you may directly download a copy of the datasets from the links below:

ne_10m_airports.zip (http://www.qgistutorials.com/downloads/ne_10m_airports.zip)

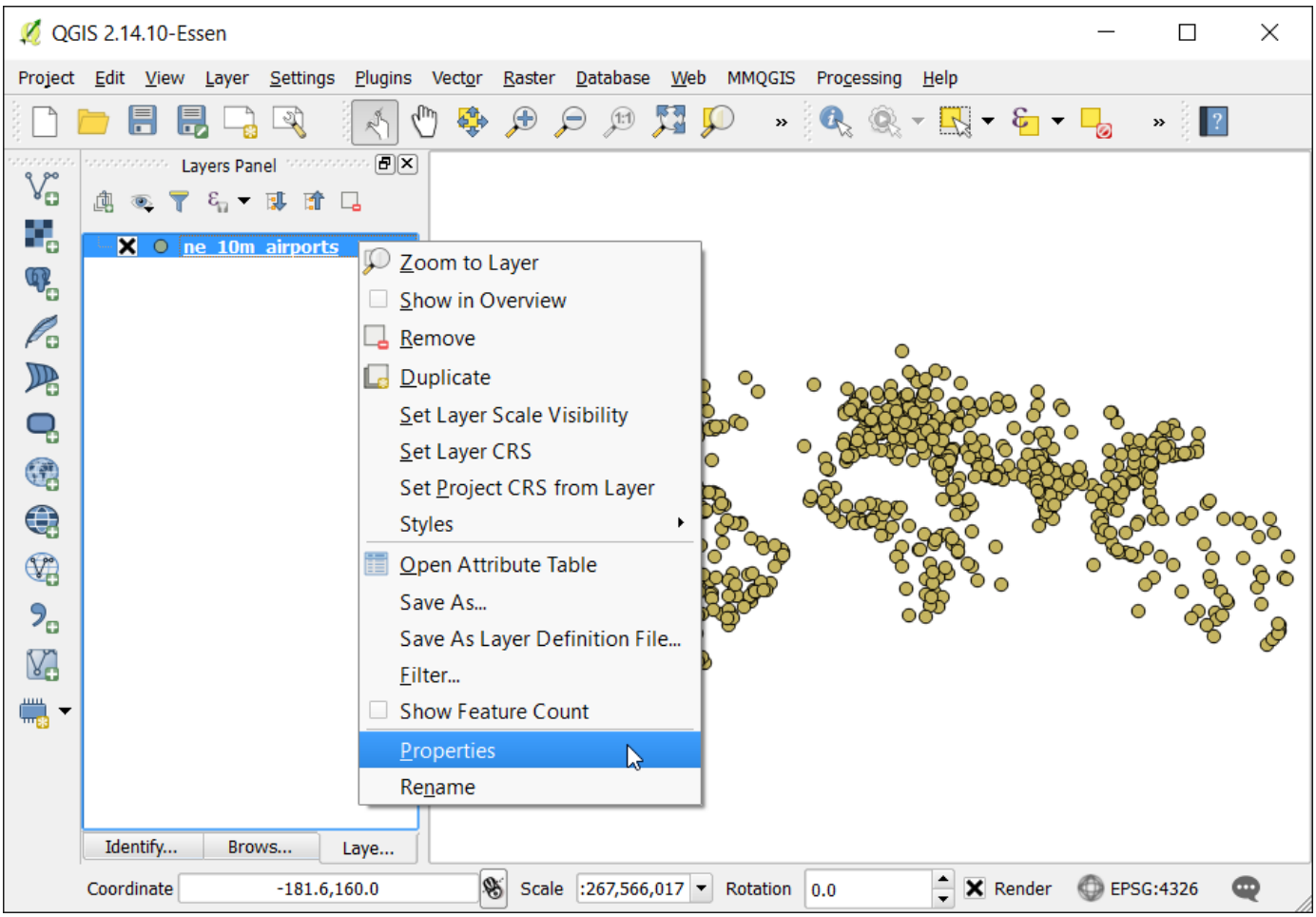
Data Source [NATURALEARTH] (credits.html#naturalearth)

Procedure

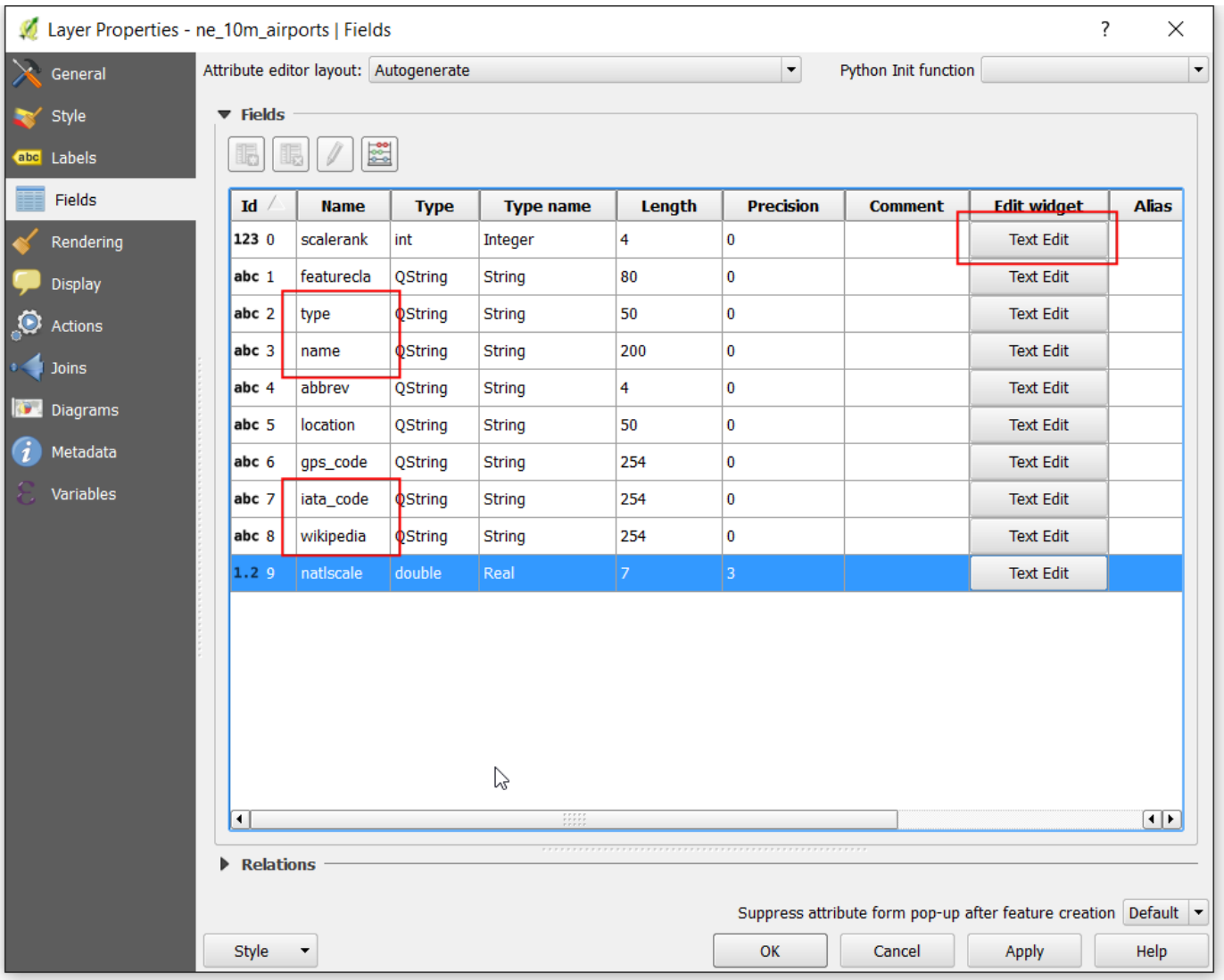
1. Open QGIS and go to Layer ▸ Add Vector Layer. Browse to the location of the downloaded file and select `ne_10m_airports.zip` . Click OK.



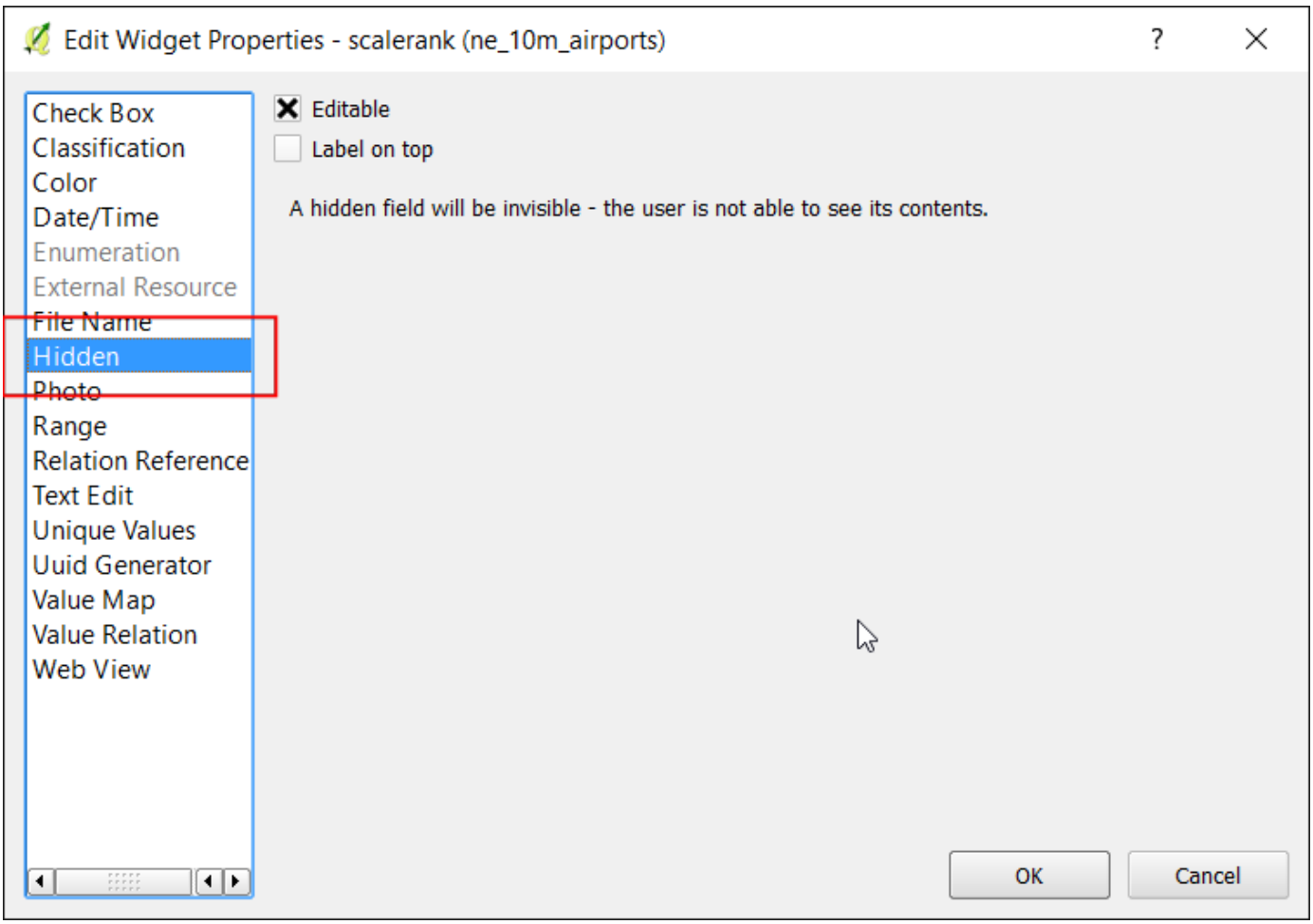
2. We will now create a map in QGIS that looks and behaves just like we would like in the web map. The plugin `qgis2web` will use replicate the QGIS settings and automatically create the web map without us knowing about web mapping libraries. When a user clicks on a airport marker, we want an info-window to disply useful information about the airport. This information is already present in the attribute table of the `ne_10m_airports` layers. Right-click on the `ne_10m_airports` layer and select Properties.



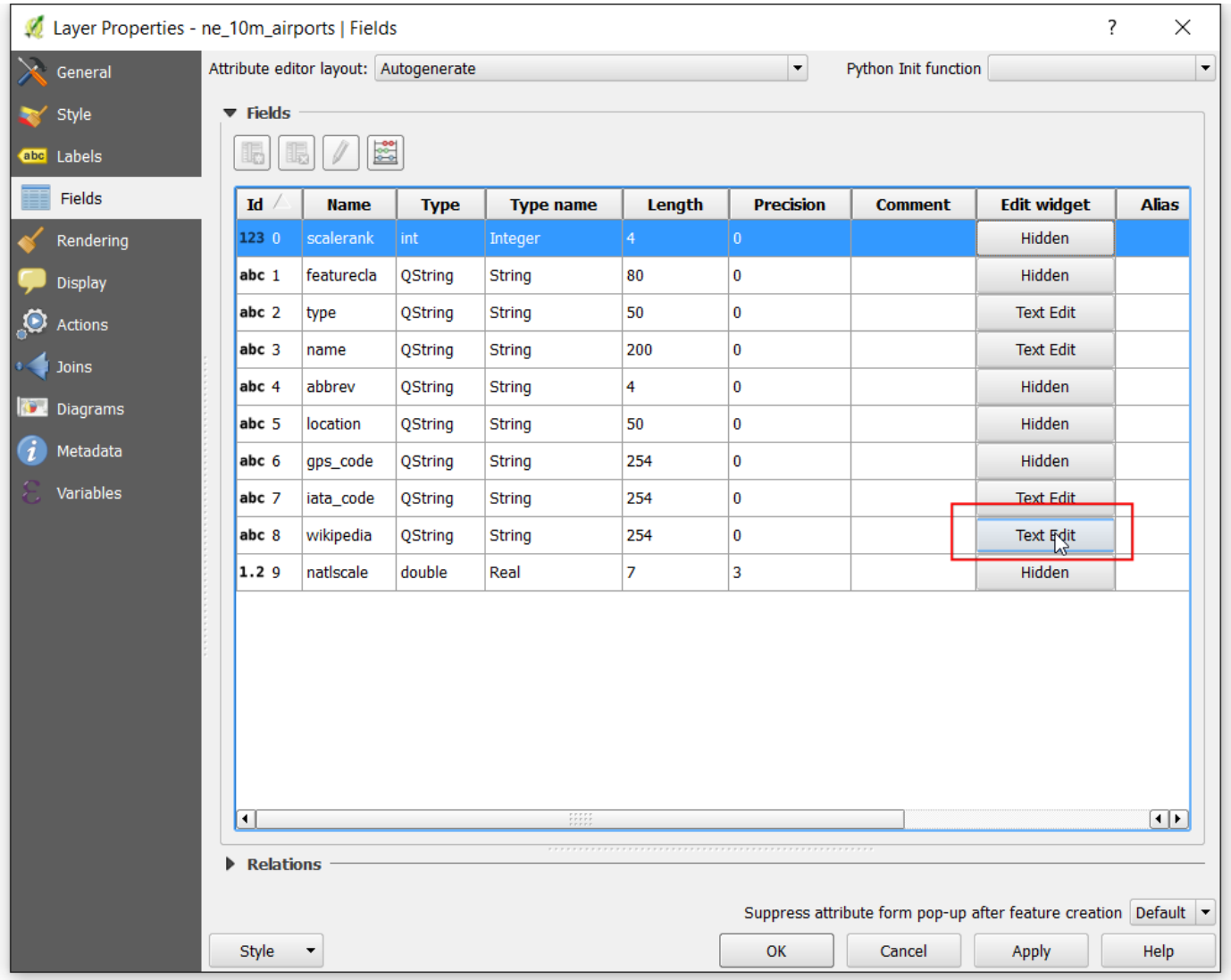
3. Switch to the Fields tab. You will notice the different attributes present in the layer. Some of these aren't relevant to the users of our web map, so we can choose to hide these. We will keep `type`, `name`, `iata_code` and `wikipedia` fields and hide the others. Click on Text Edit button under the Edit widget column for `` `scalerank `` field.



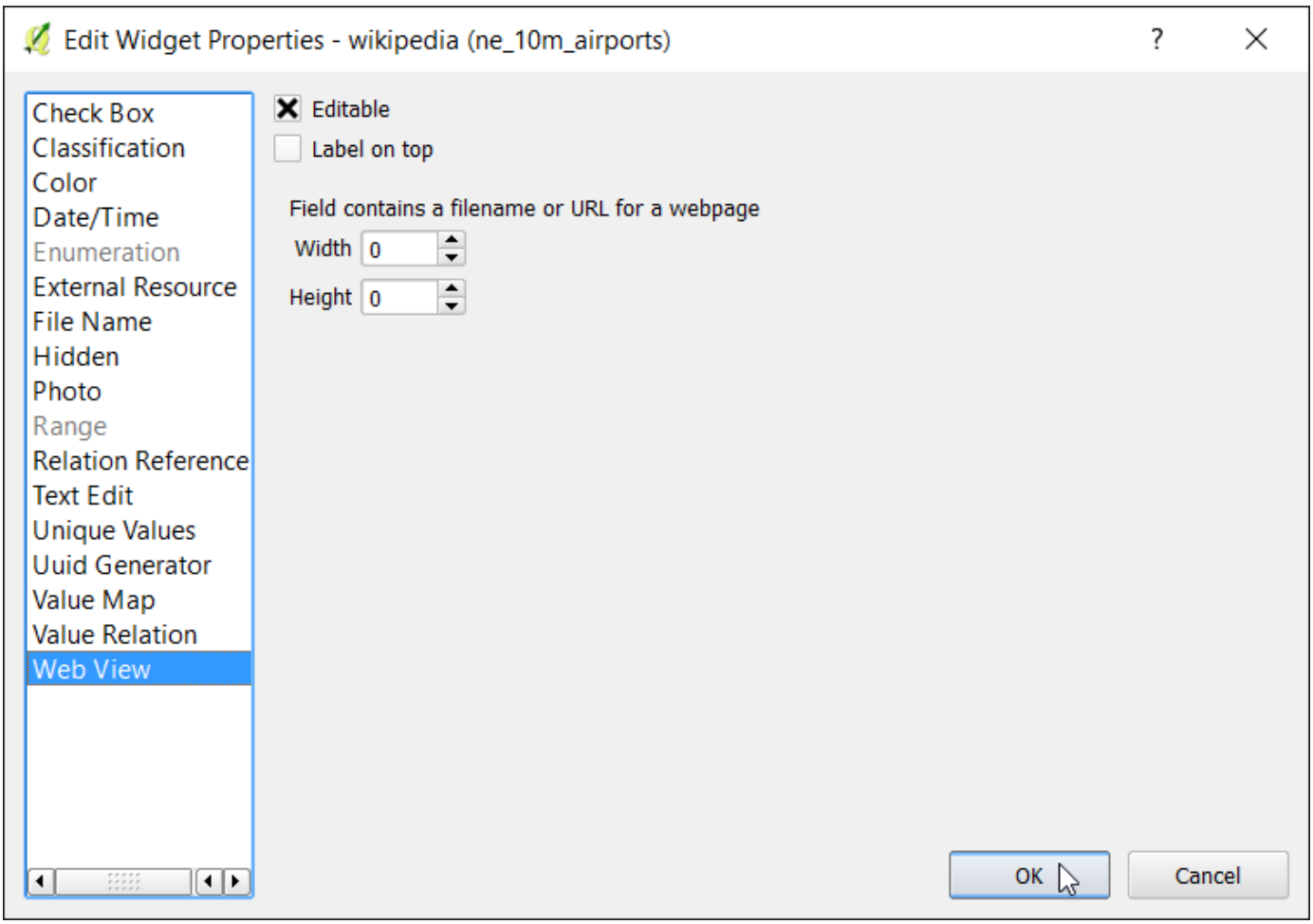
4. In the Edit Widget Properties dialog, choose `Hidden` as the type. Click OK.



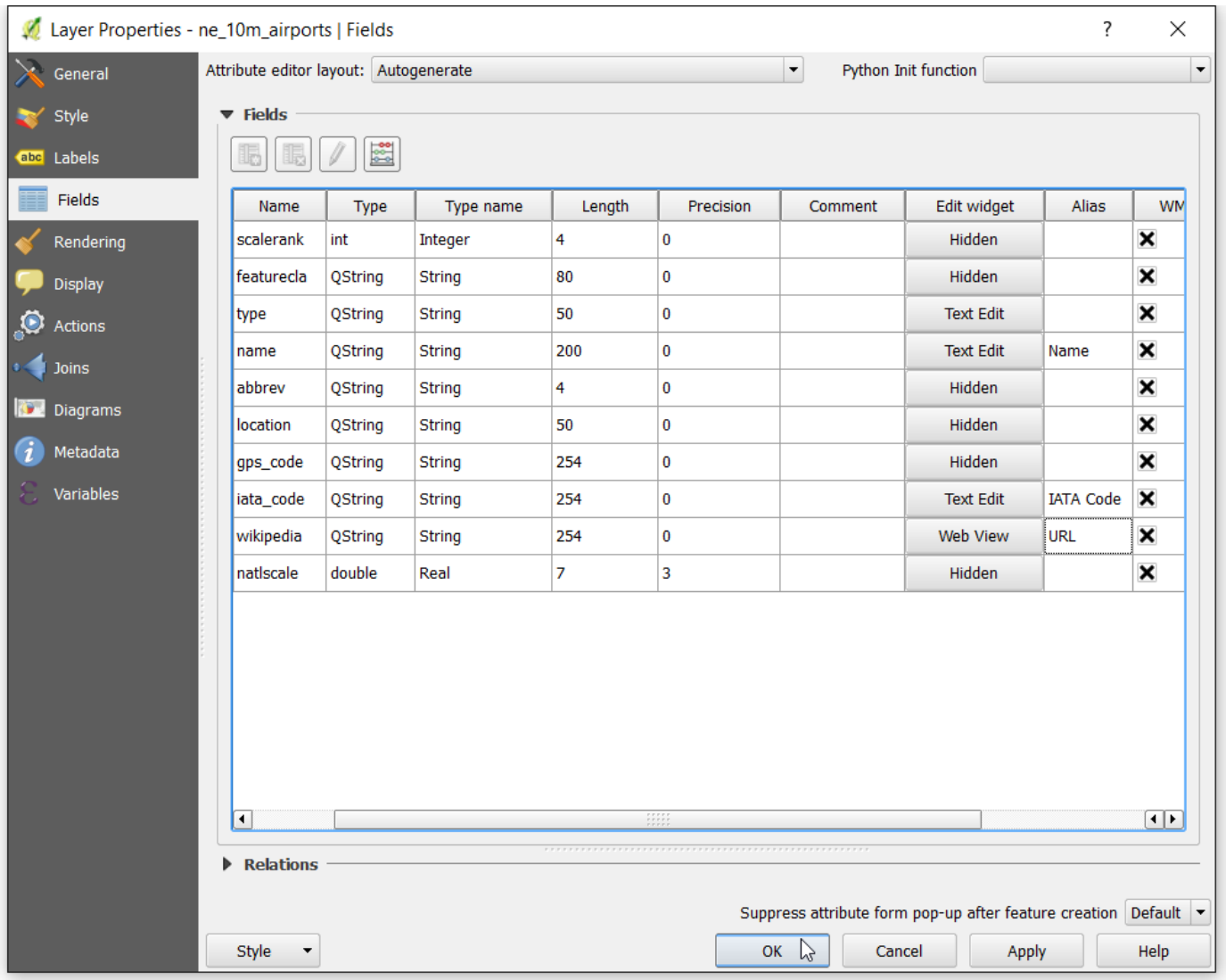
5. Similarly set other fields to Hidden type. As you may have notices, there are other field types available that allow us to set how the fields appear to the users of our map. Click Edit Widget for `wikipedia` field.



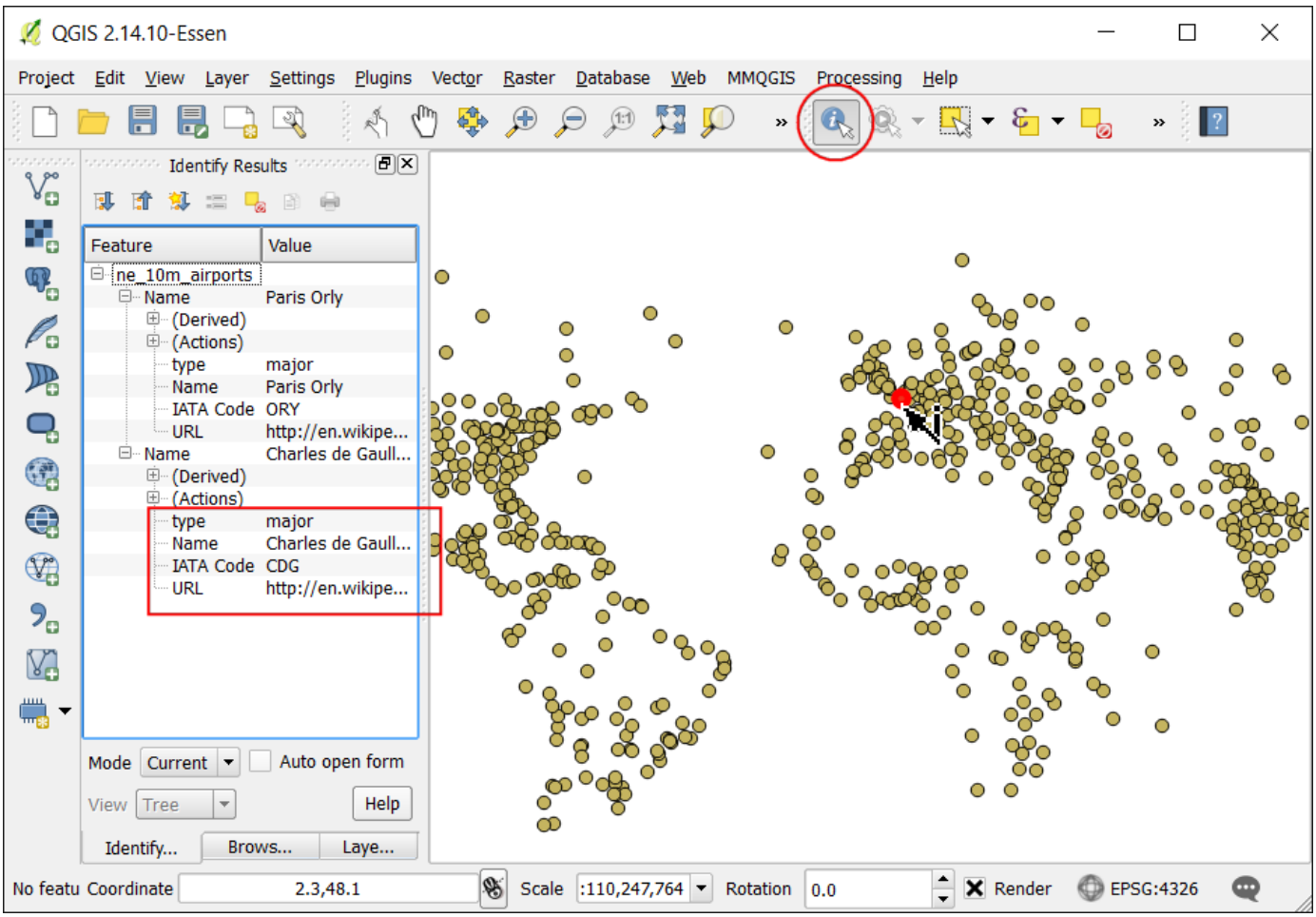
6. Select `Web View` as the field type. This type indicates that the value contained in this field is a URL.



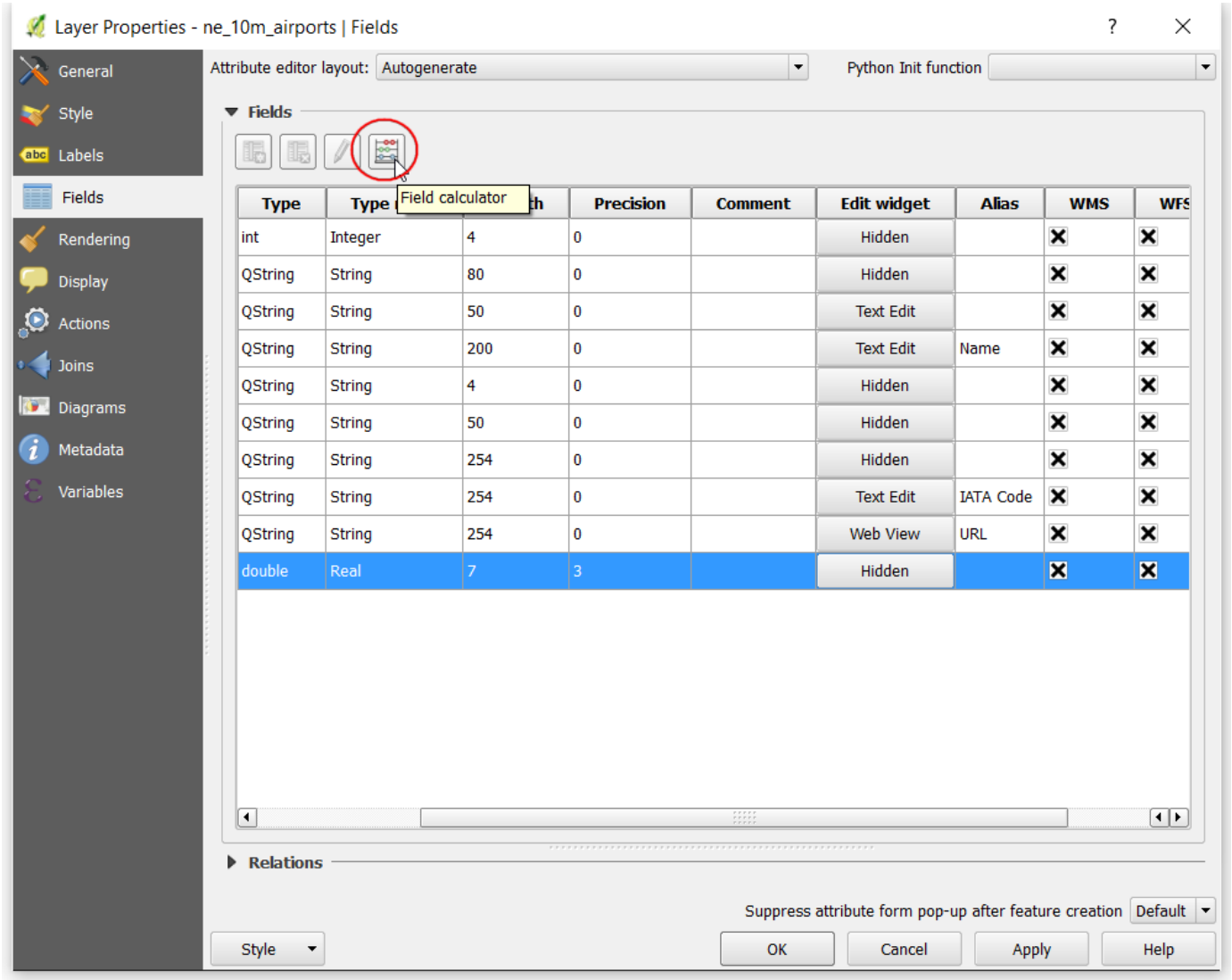
7. We can also use the Alias column to indicate an alternate name for the fields without actually changing the underlying data table. This is useful to give more user-friendly field names to the users of our map. Add aliases as per your choices and click Ok.



8. Back in the main QGIS window, choose the Identify tool and click on any point. The Identify Results panel will display the nicely formallted attributes with the newly added aliases. You will notice that the hidden fields do not appear in the results.

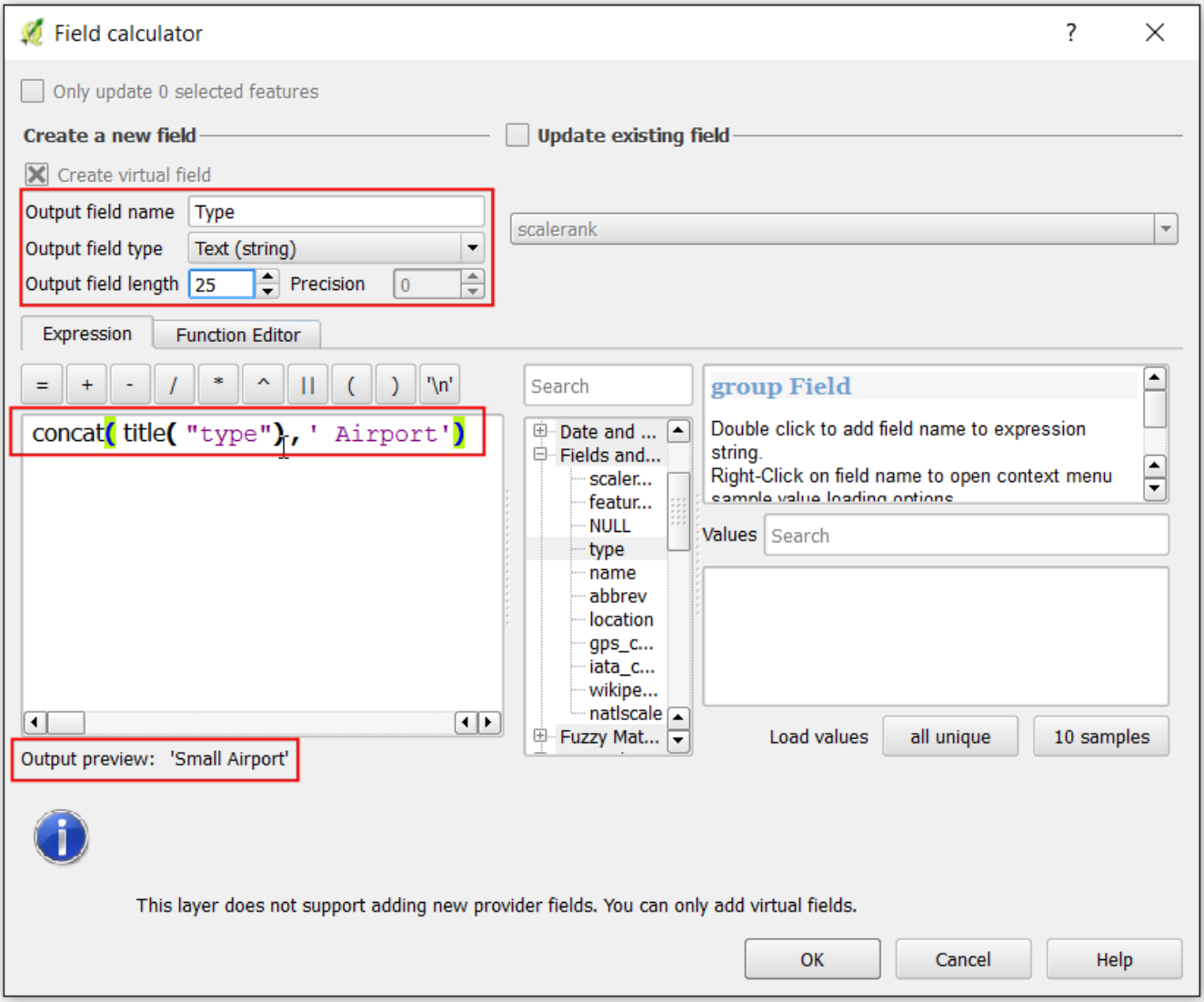


9. While this method is useful, there is one limitation. We are not able to change the order of the fields. One way to overcome this limitation is to create a `Virtual Field`. In our case, if we wanted the `type` field to appear at the end of the info window, we can simply add a new virtual field the end and hide the original `type` field. While we are at it - we can also use an expression to better format the type values. Right-click the `ne_10m_airports` layer and choose Properties. Go to the Fields tab and click Field Calculator.

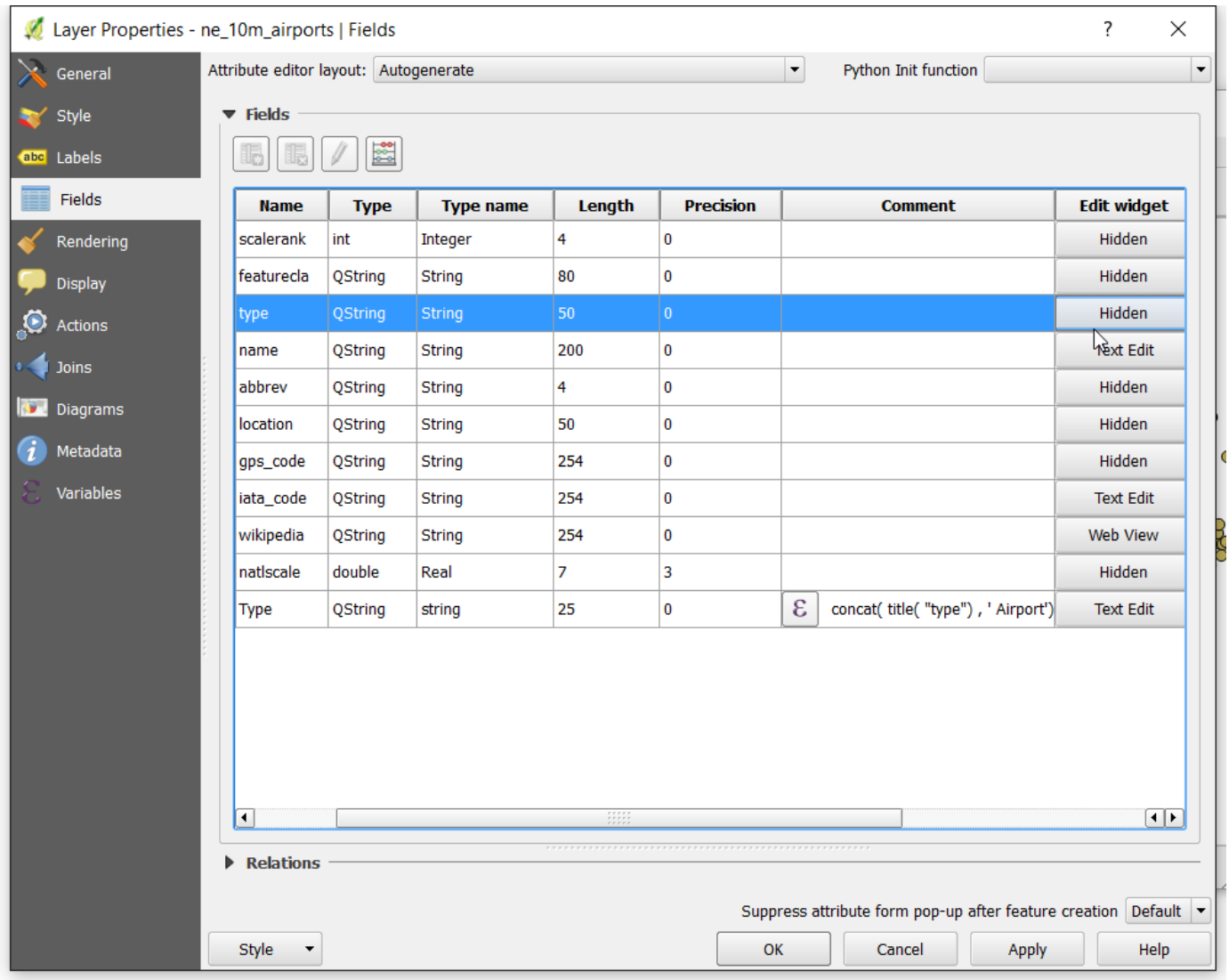


10. As the field names need to be unique, use `Type` as the new field name. Set the field type to `Text (String)` with a length of `25` characters. The field `type` contains values such as `small`, `mid`, `large` etc. We can add an expression to change the case of the words to sentence case and append the word *airport* for better readability. Enter the following expression in the Expression box and click OK.

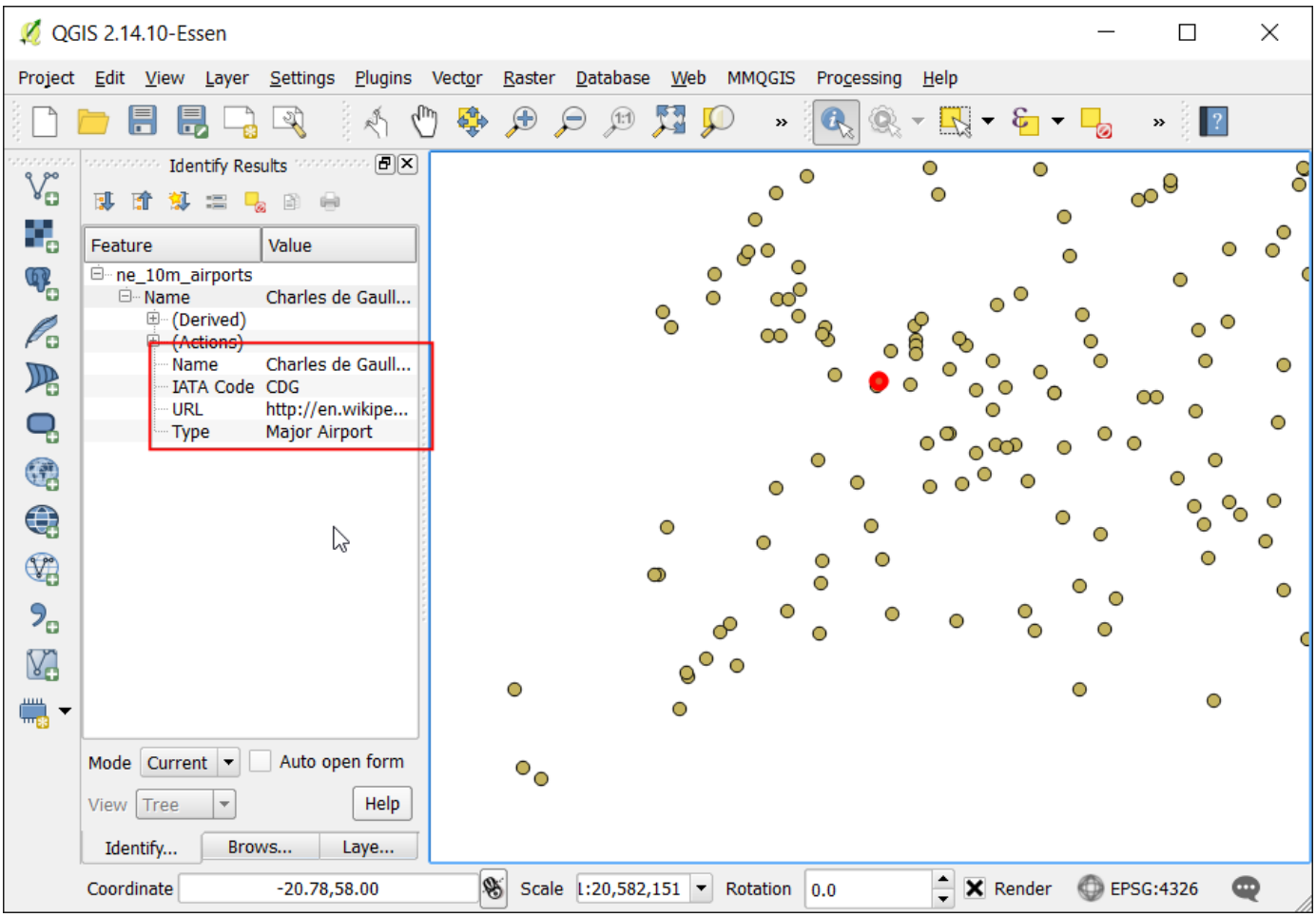
```
concat( title("type"), ' Airport')
```



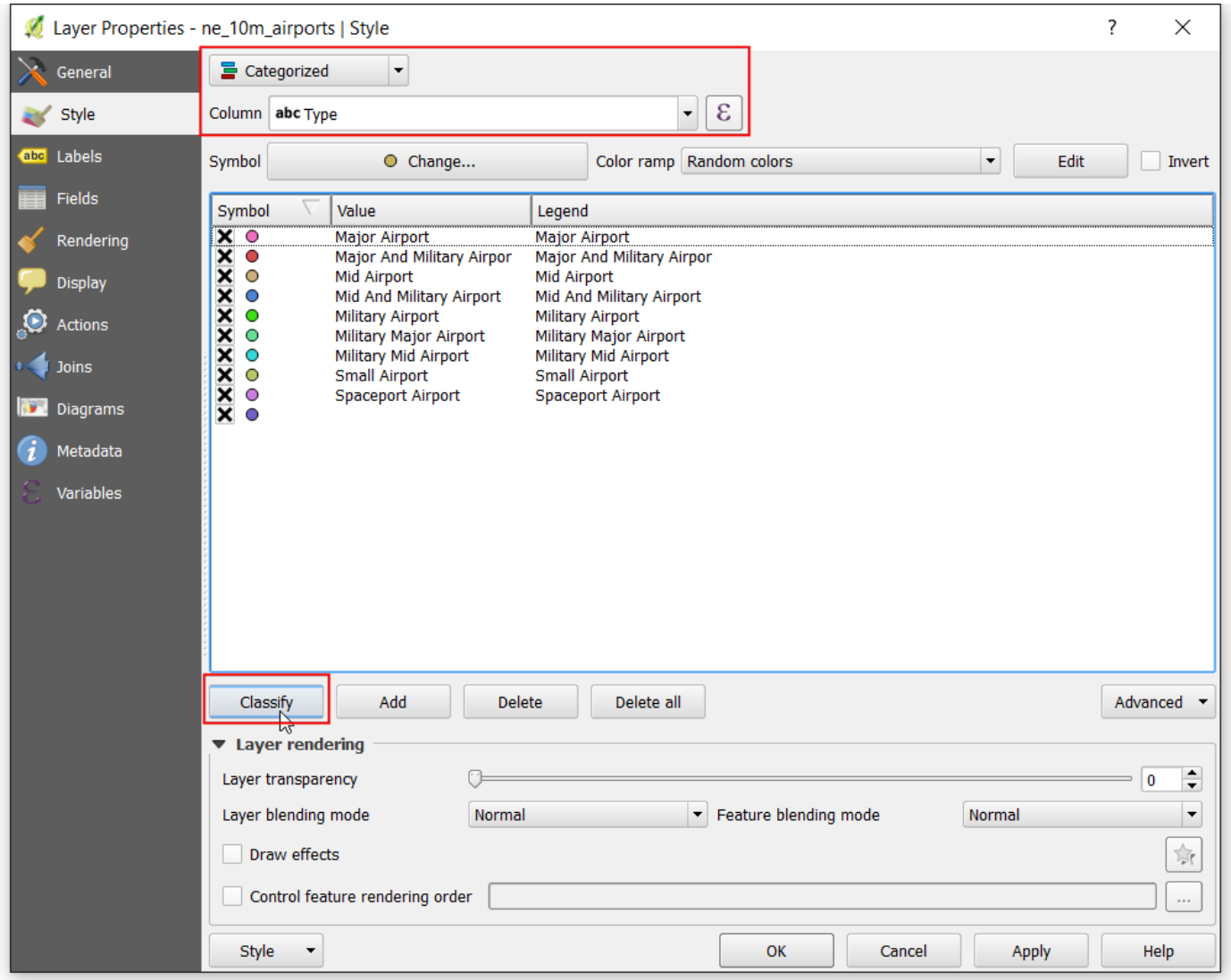
11. Now that we have a much better looking `Type` field, you can go ahead and set the Edit Widget for `type` field to `Hidden`.



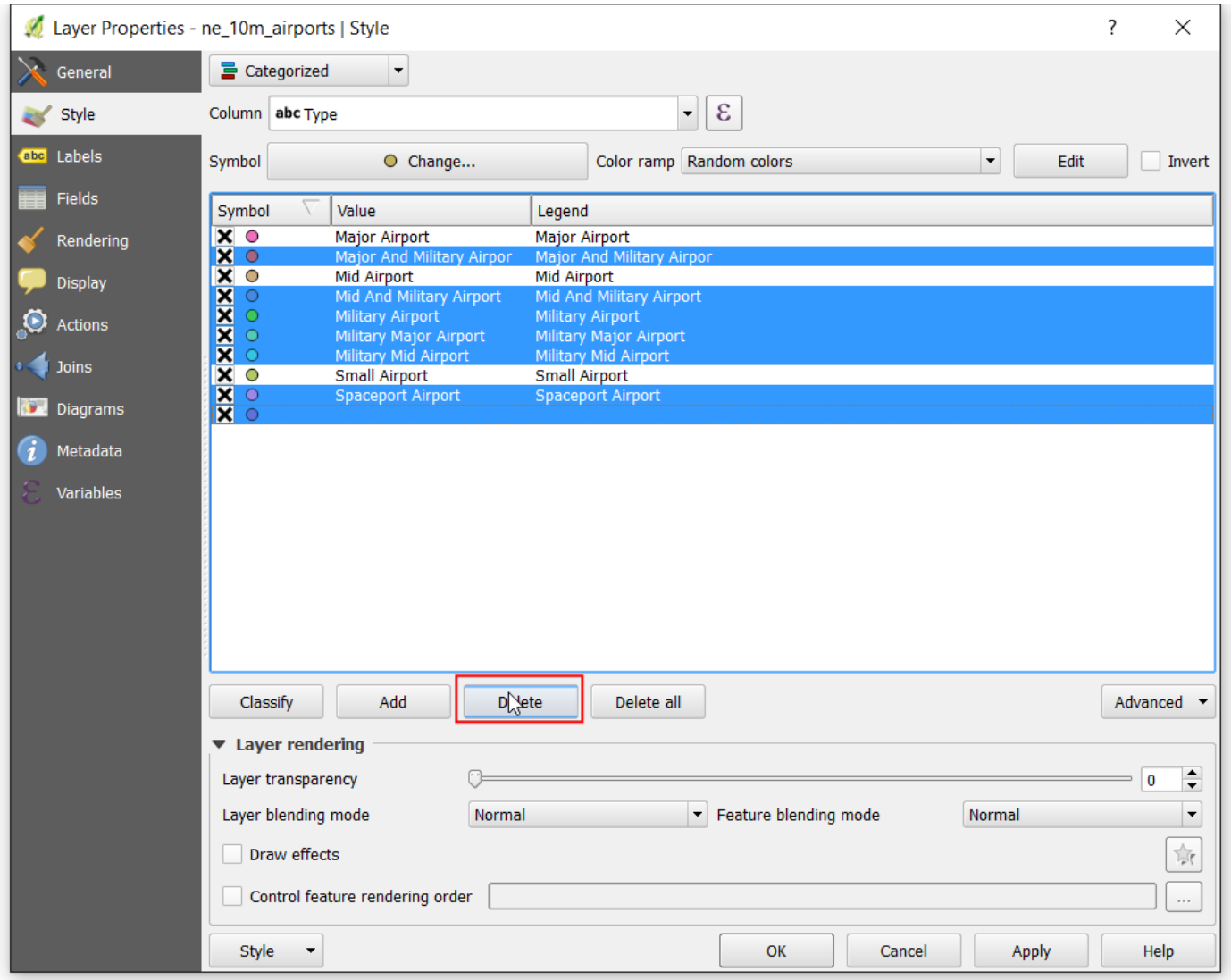
12. Use the Identify tool to verify that the attributes appear as expected.



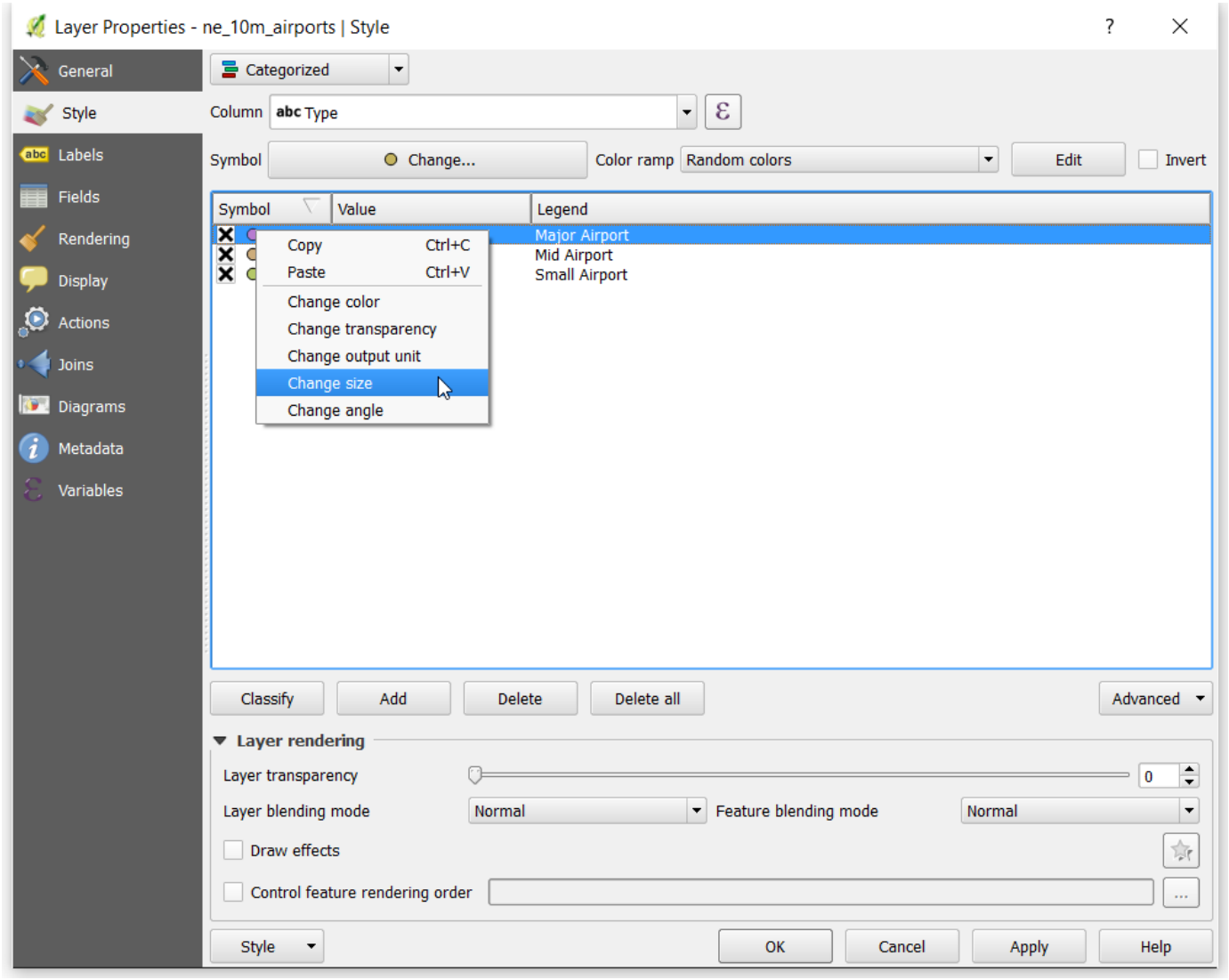
13. Now let's style our layer to be more visually appealing and informative. Right-click the `ne_10m_airports` layer and select Properties. Switch to the Style tab. Choose `Categorized` style and our virtual field `Type` as the Column. Click Classify.



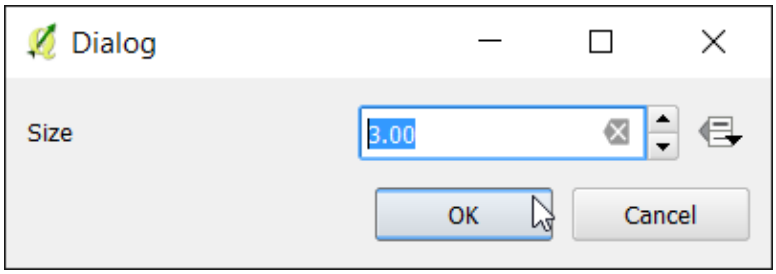
14. You will see a different colored circle gets assigned to a differnt type of airport. For the purpose of this tutorial, we will restrict the map to civilian airports. Hold the `Ctrl` key and select all categories for military airports. Once selected, click Delete.



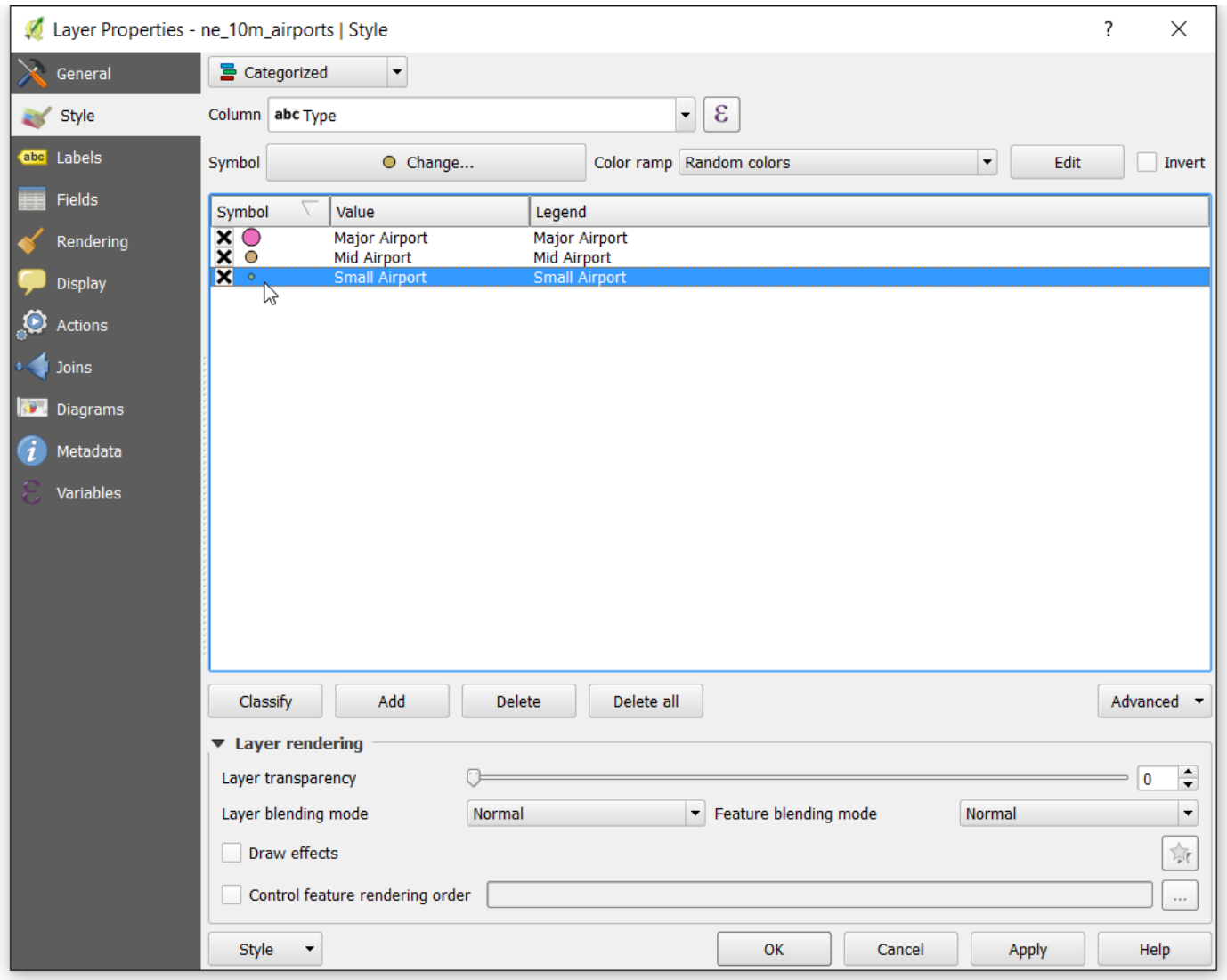
15. Apart from assigning a different color to the category, we can change the size of the symbol to visually help our users distinguish different type of airports. Right-click on a category and select Change size.



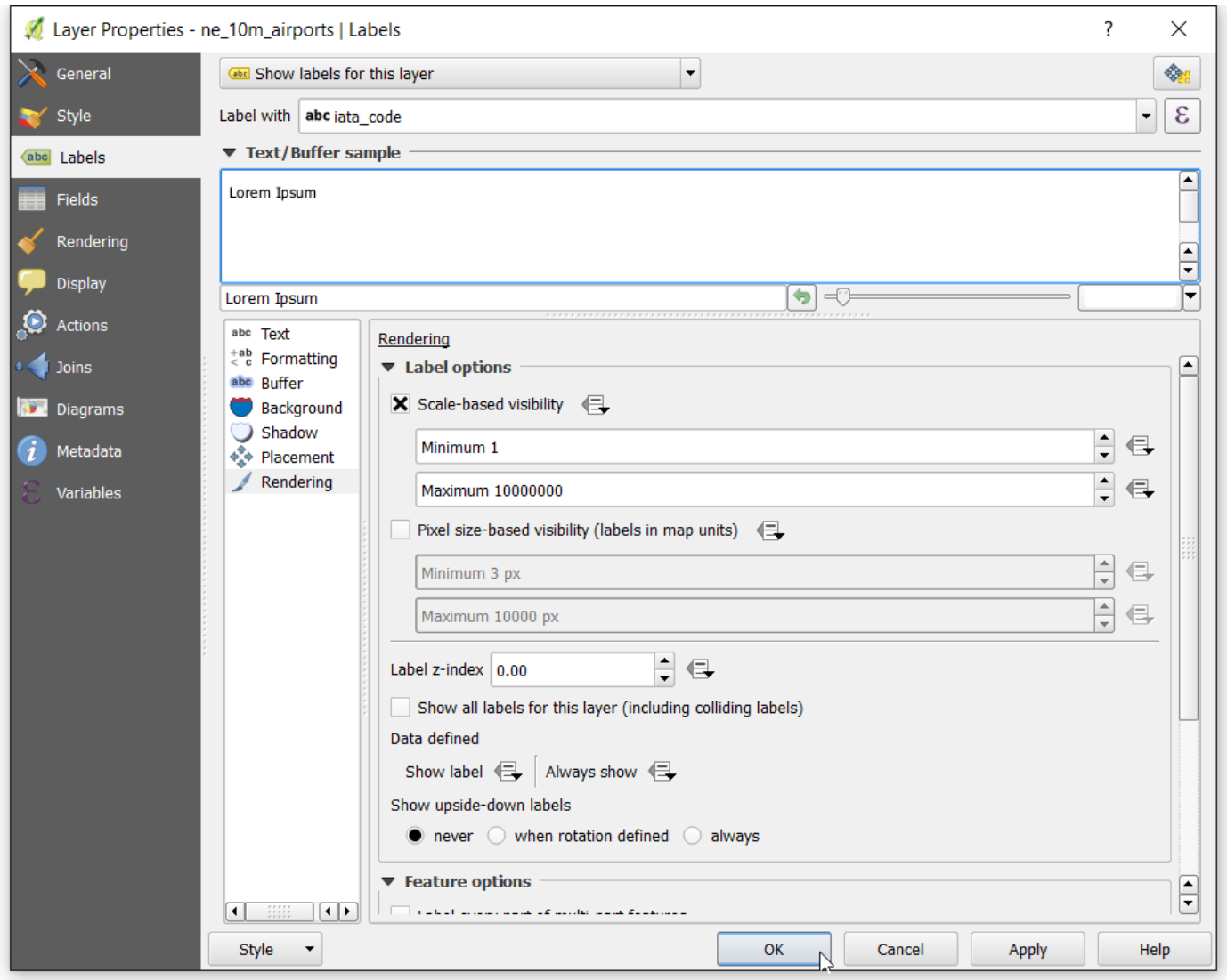
16. Set the Size value to `3` for the `Large Airport` category.



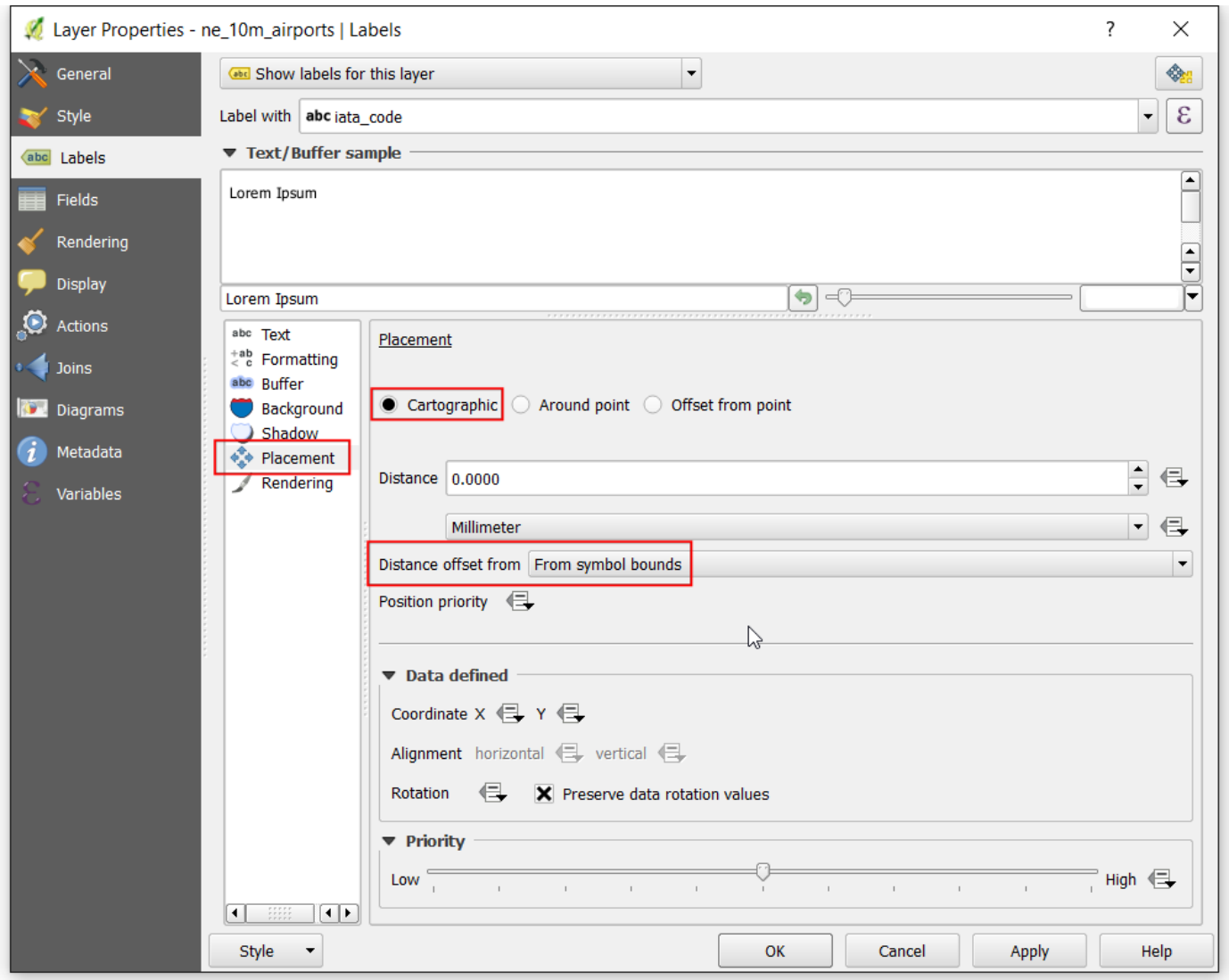
17. Similarly, set the Size to `2` for `Mid Airport` and `1` for `Small Airport`.



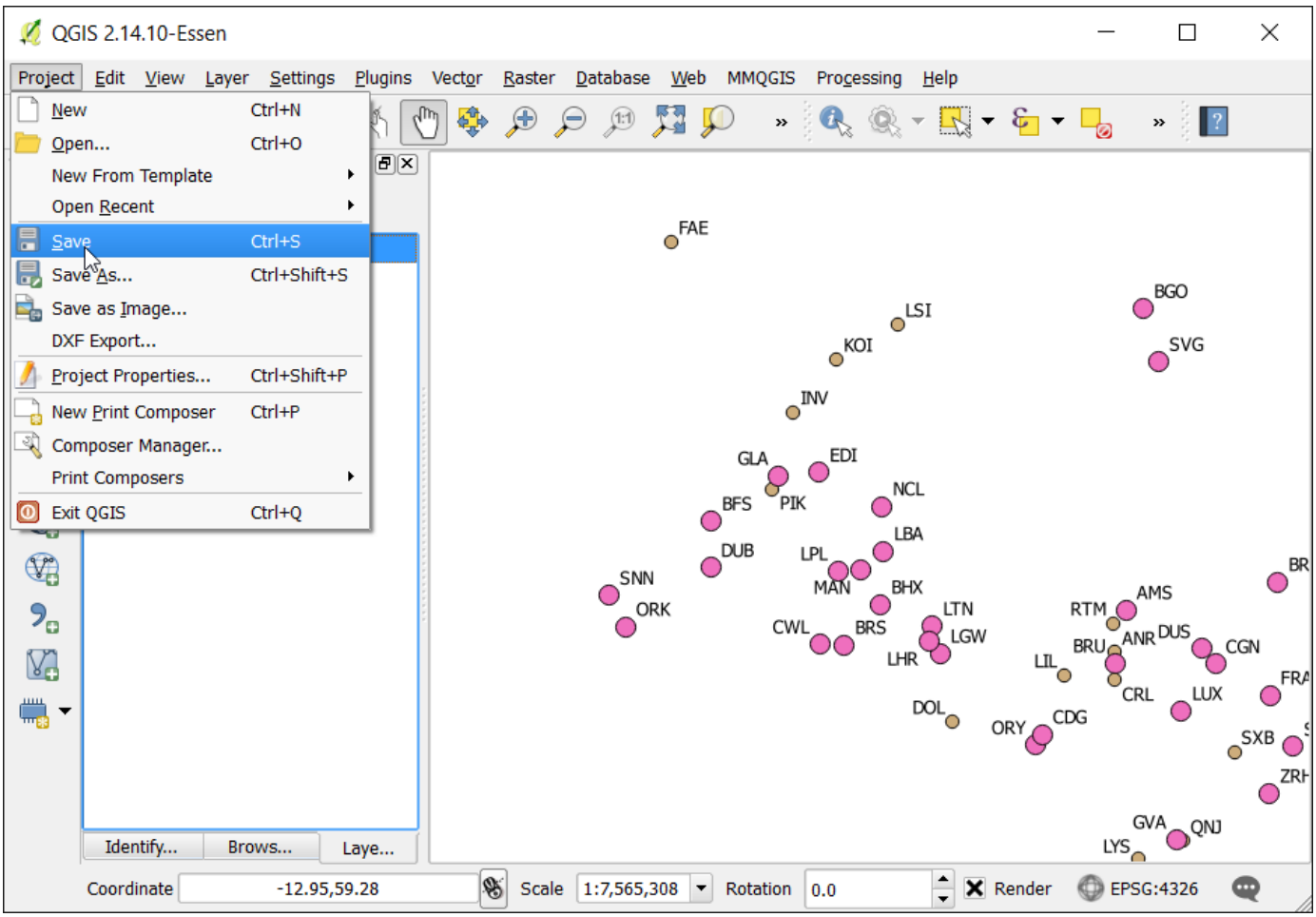
18. For a complete map, we also need to label each airport. Switch to the Labels tab in the Properties dialog. Select `Show labels for this layer` and choose `iata_code` as the value for Label with. We will also set Rendering option so that the labels only appear when the user is sufficiently zoomed in. Check Scale-based visibility under Label options. Enter `1` as the Minimum scale and `10000000` as the maximum scale. This setting will render the labels only after the user has zoomed in more than `1:10000000` scale and will be visible till `1:1` scale.



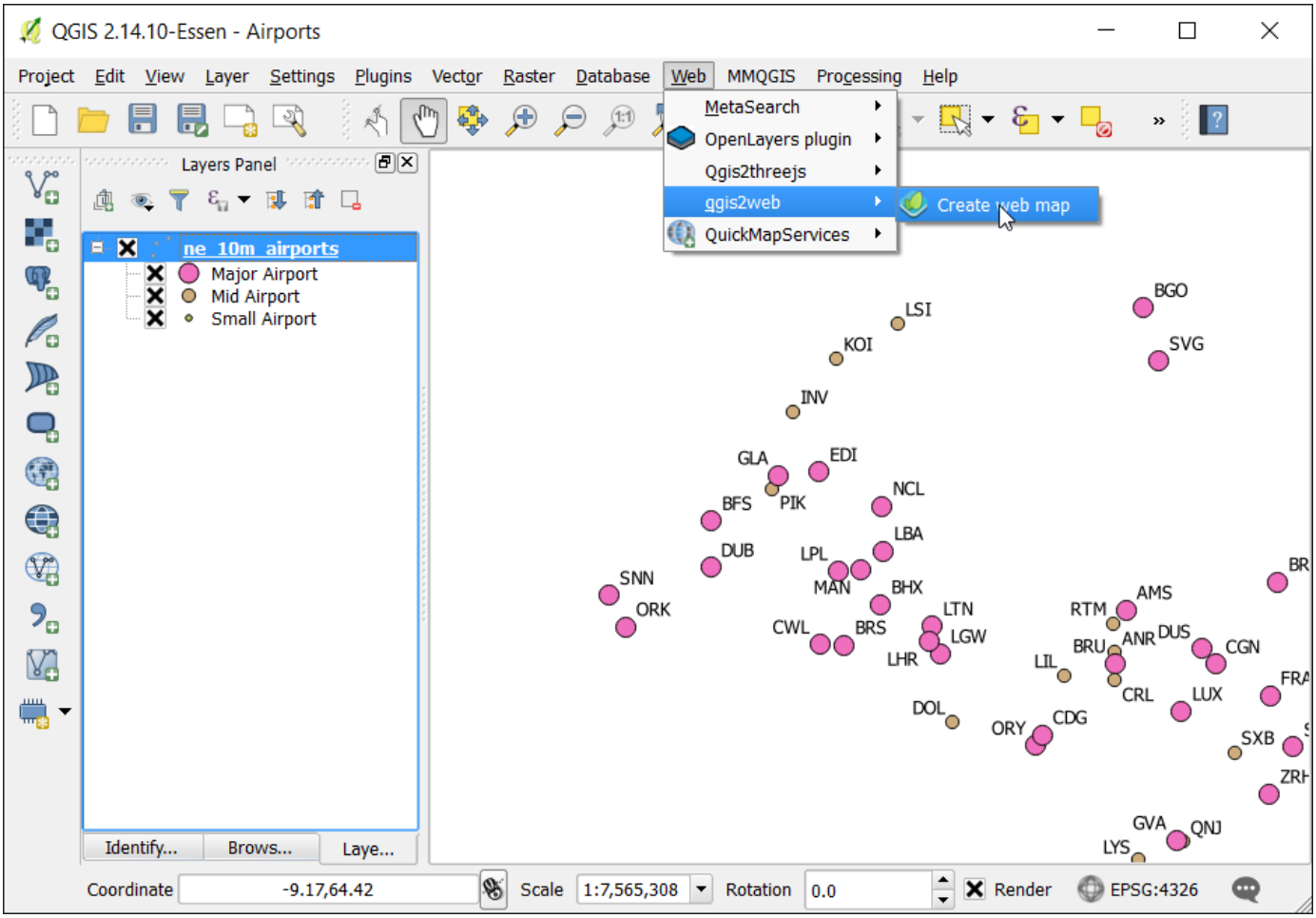
19. As we are using circles to depict the airports, we need to ensure that the labels don't overlap with the circles. Go to the Placement tab in the Labels dialog and set the Placement to `Cartographic`. Select `From symbol bounds` as Distance offset from. Click OK.



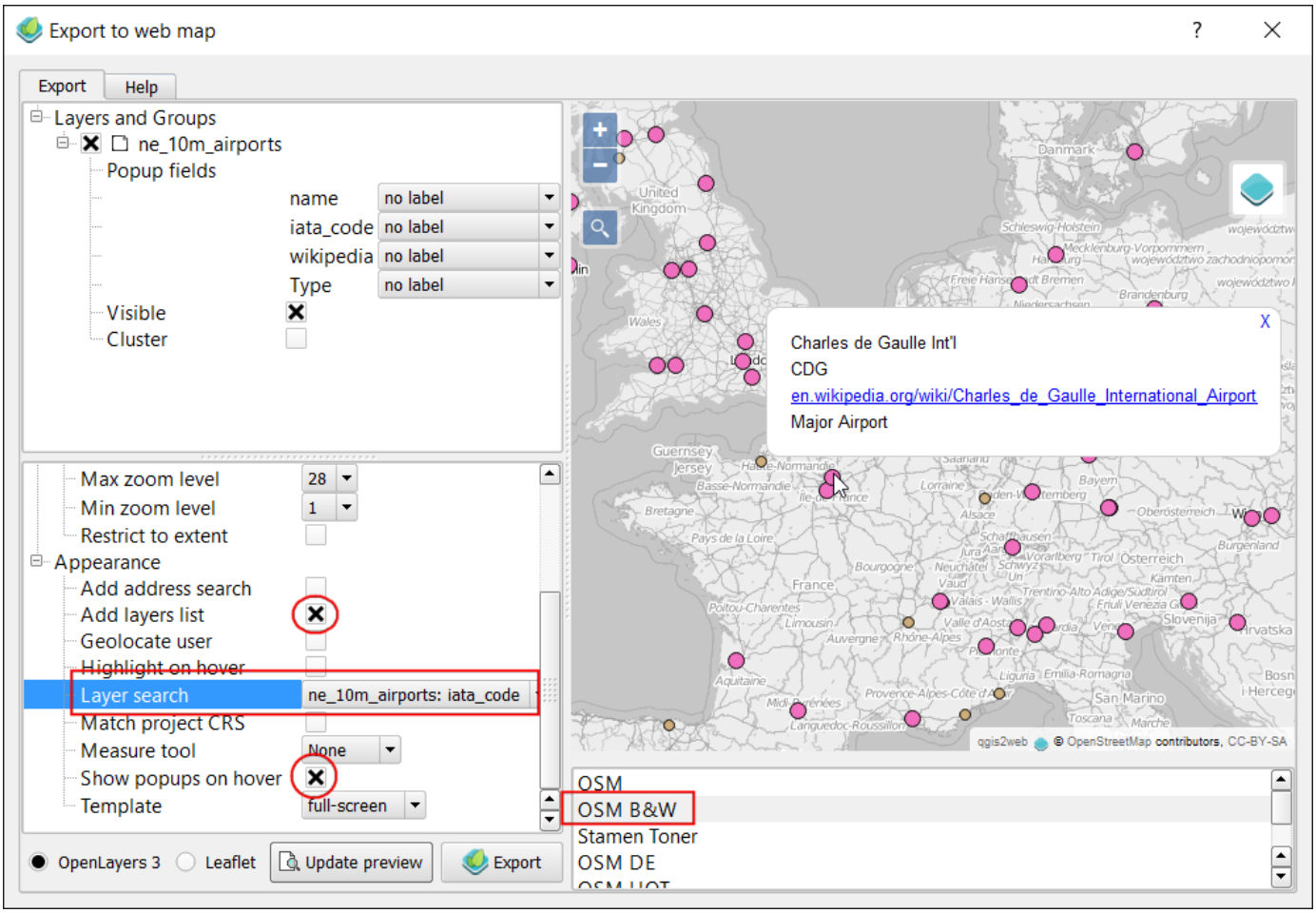
20. Our map is now ready. This is a good time to save our work. Go to Project ▸ Save. Enter `Airports` as the name of the project.



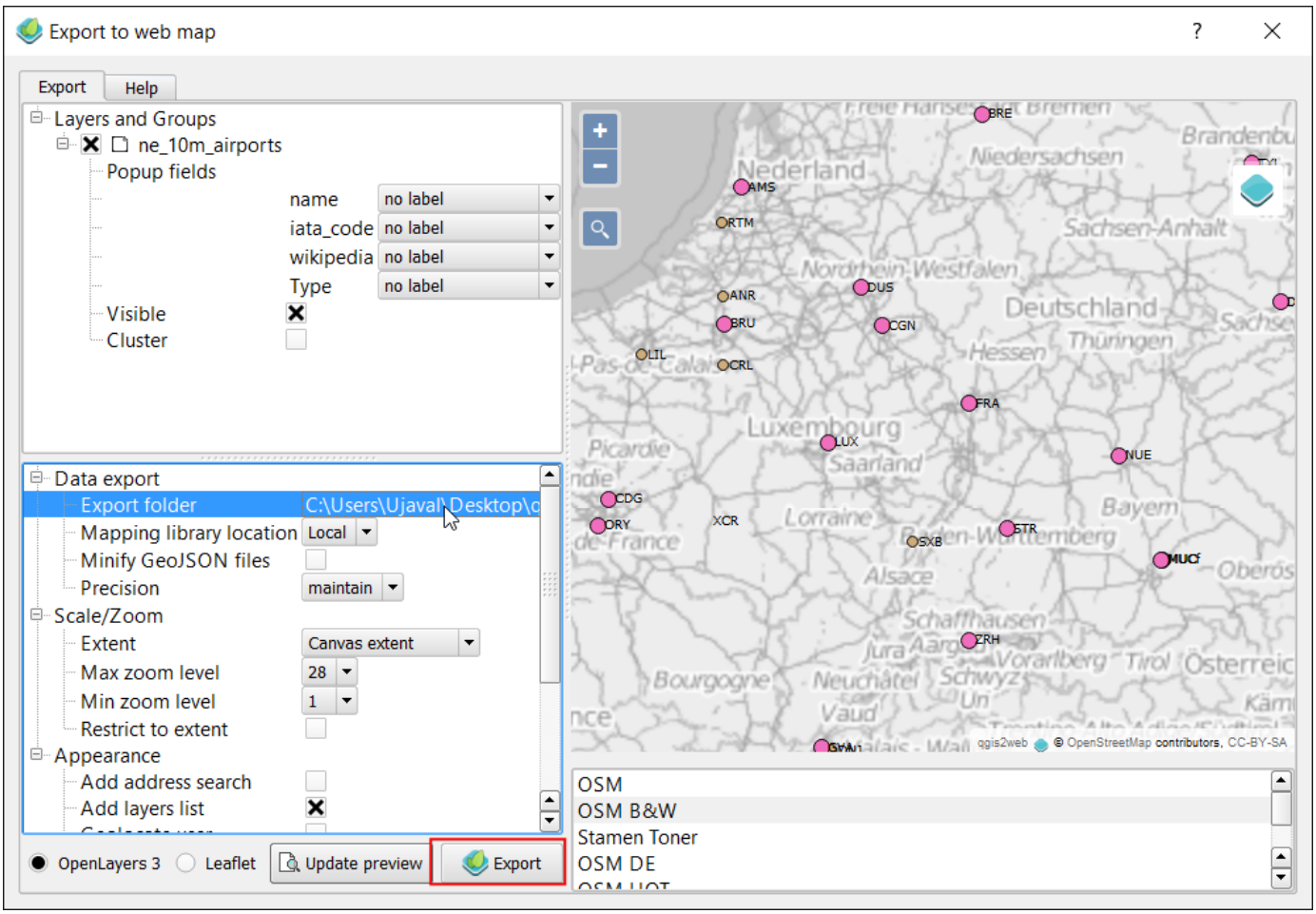
21. Now we are ready to export our project to a web map. Install the `qgis2web` plugin by going to Plugins ▸ Manage and Install Plugin (See *Using Plugins* (using_plugins.html) for more details on installing plugins in QGIS). Once the plugin is installed, go to Web ▸ qgis2web ▸ Create a web map.



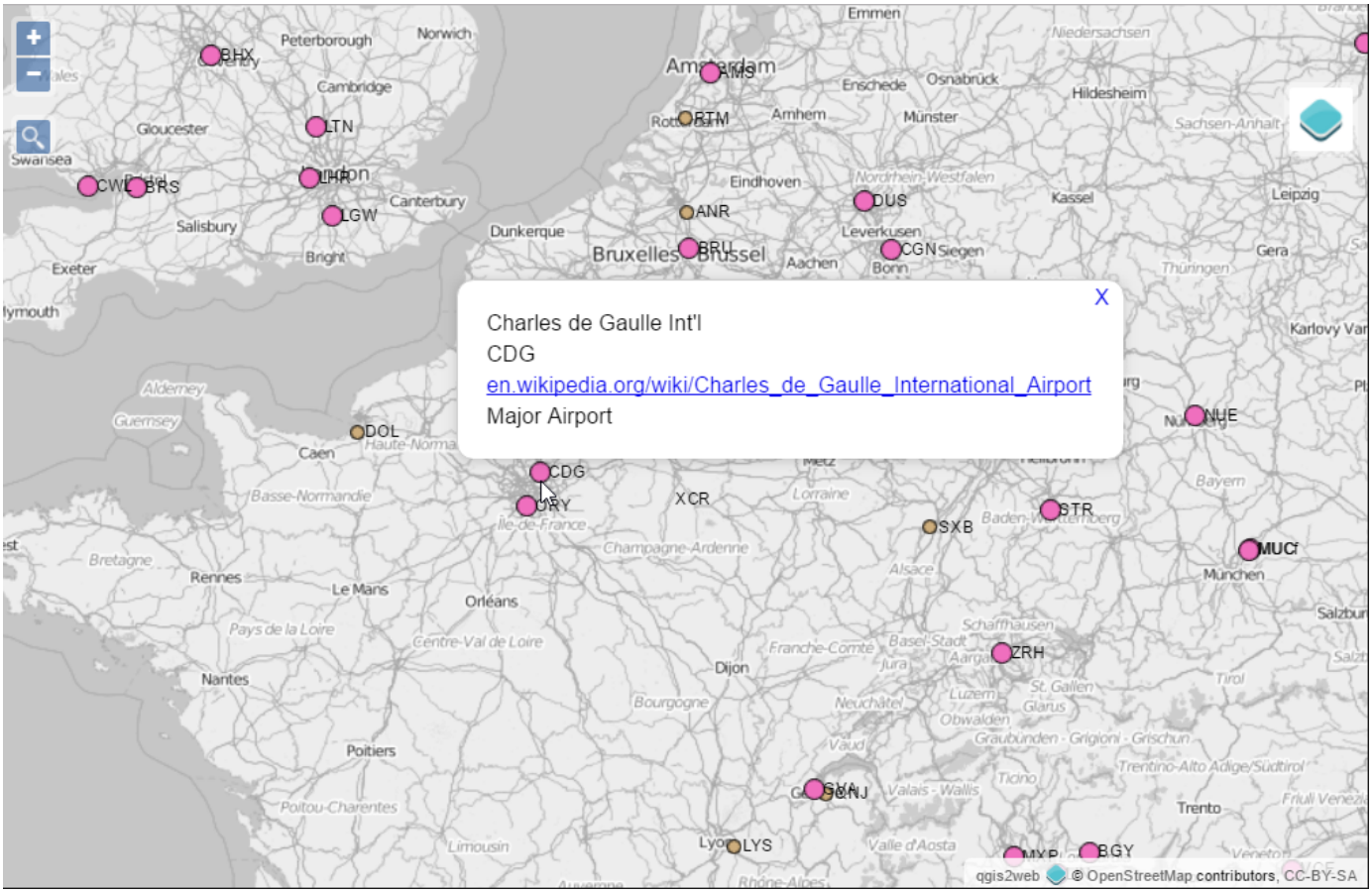
22. In the Export to web map dialog, check Add layers list in the bottom panel under the Appearance section. Also select `ne_10m_airports: iata_code` as the field for Label search. Check the Show popups on hover to allow display of info-windows on hover. We can also set a basemap so the users have more context when looking at the airports layer. Select `OSM B&W` to use a black-and-white themed basemap create using OpenStreetMap data. You also have an option to choose from either `OpenLayers` or `Leaflet` as the web mapping library. We will restrict this tutorial to use the `OpenLayers` library. Click Update Preview` to see how your exported map will look like. Before we do the actual export, we need to set the Export folder. You can select a folder of your choice and click Export.



23. Once the export is complete, the default browser for your computer will open and show the interactive web map.

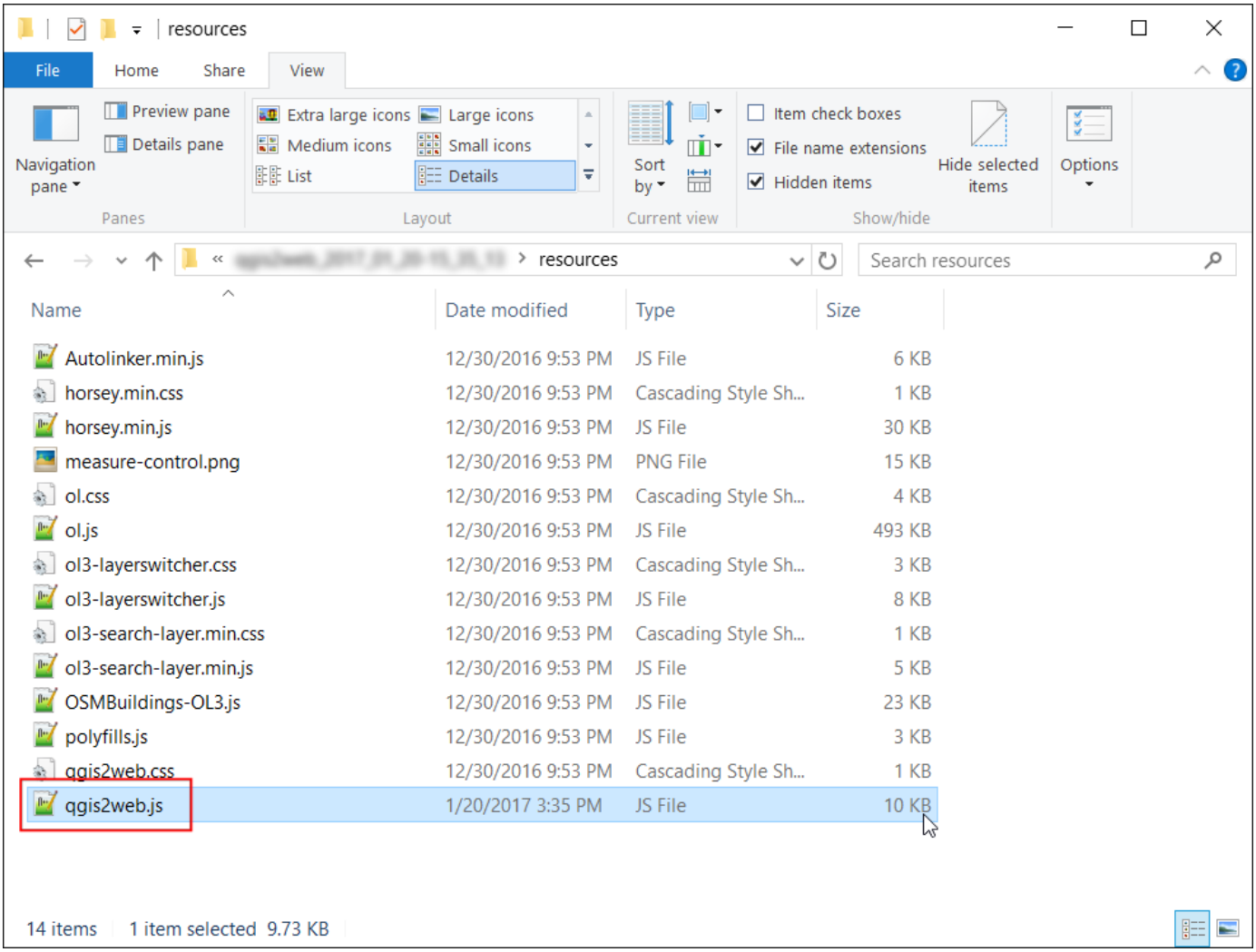


24. Your web map is now ready for publishing.



25. The `qgis2web` plugin has many limitations and it cannot do everything that the powerful web mapping libraries `OpenLayers` and `Leaflet` can do. This process can act as the starting point in your web mapping process and save you valuable time by creating a basic template from which you can further customize the web map. To highlight the fact that the output created from this process can be readily changed to suit your requirement - we will make a simple change to the web map to zoom to a particular airport when the

user initially loads the map. On your computer, go to the folder where the web map was exported. Locate the `resources` folder and open `qgis2web.js` file in a text editor.



26. Locate the line where the `map.getView().fit()` function is called and add the following code after that. This new line of code instructs the web browser to center the map on the coordinates of Paris. Save the changes to the `qgis2web.js` file.

```
map.getView().setCenter(ol.proj.fromLonLat([2.35, 48.85]))
```

resources\qgis2web.js - Notepad++

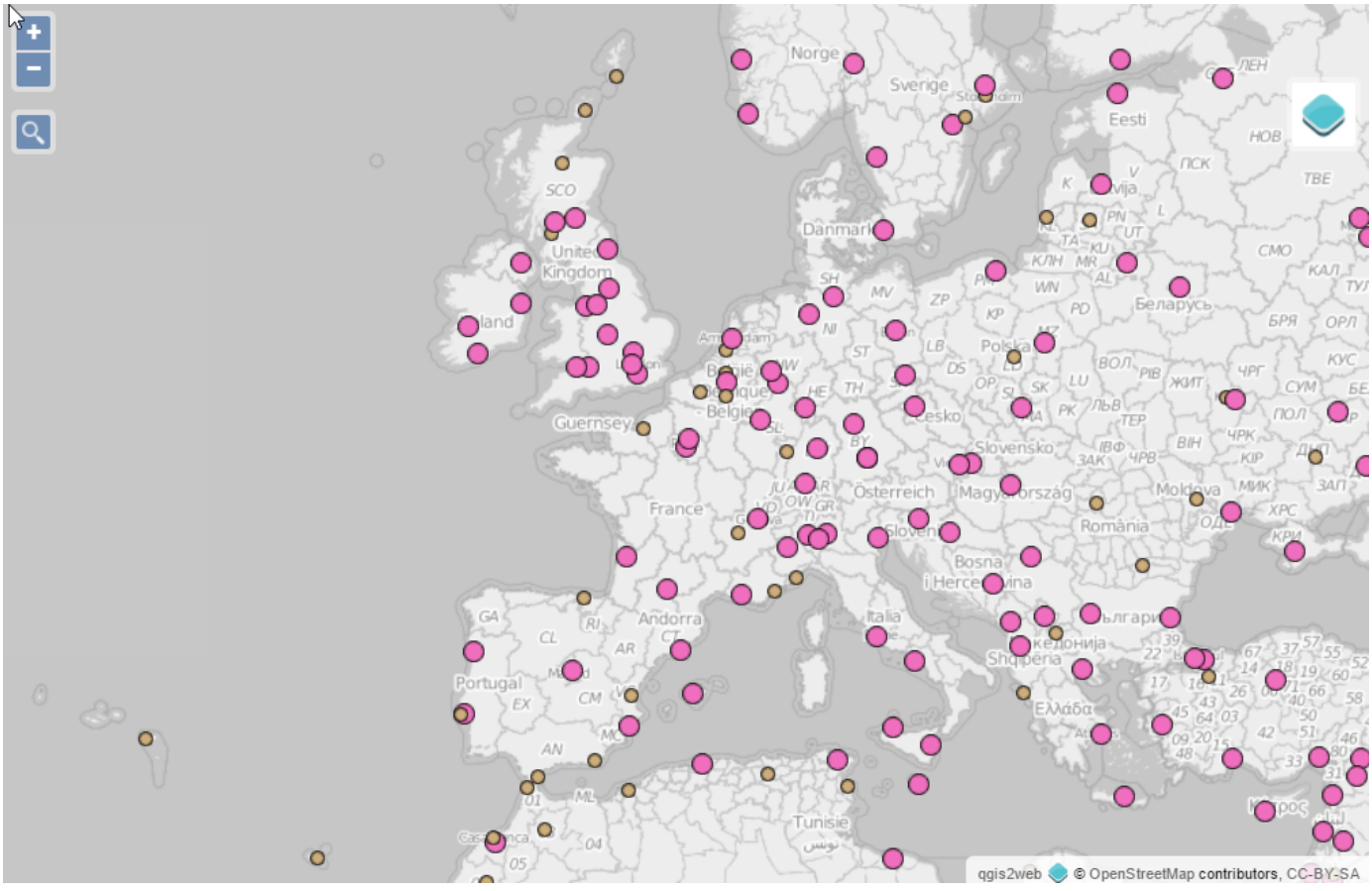
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

qgis2web.js

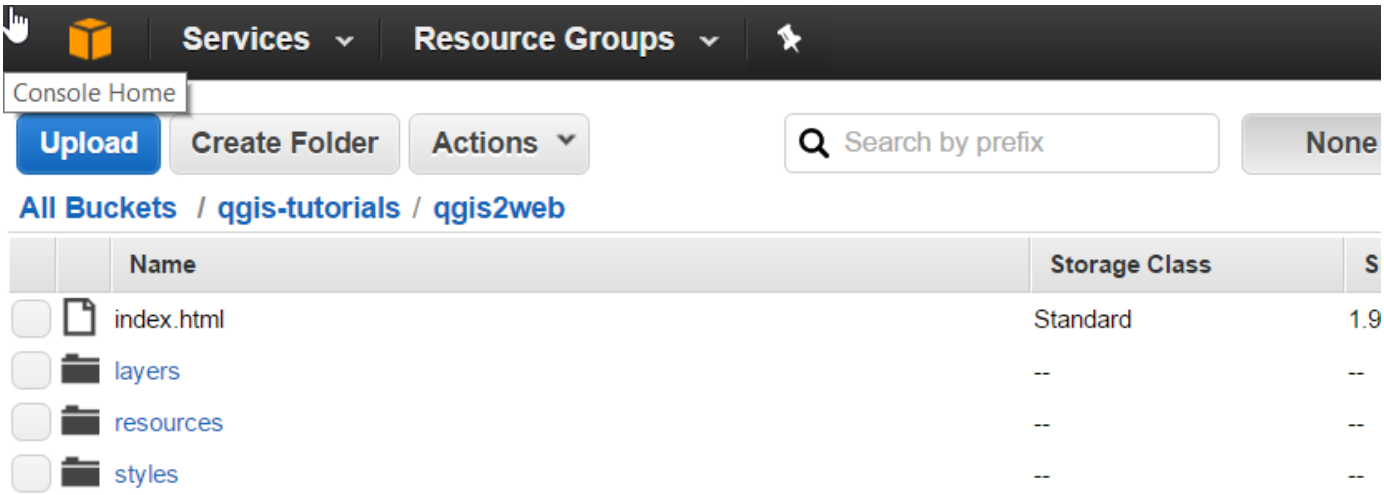
```
    })
});

    var searchLayer = new ol.SearchLayer({
      layer: lyr_ne_10m_airports,
      colName: 'iata_code',
      zoom: 10,
      collapsed: true,
      map: map
    });

    map.addControl(searchLayer);
map.getView().fit([-1519627.389624, 5724361.148262, 1068882.161610, 9457802.6828
map.getView().setCenter(ol.proj.fromLonLat([2.35, 48.85]))

var NO_POPUP = 0
var ALL_FIELDS = 1

/**
 * Returns either NO_POPUP, ALL_FIELDS or the name of a single field to use for
 * a given layer
 * @param layerList {Array} List of ol.Layer instances
```

JavaScrip length : 10,031 lines : 268 Ln : 42 Col : 59 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

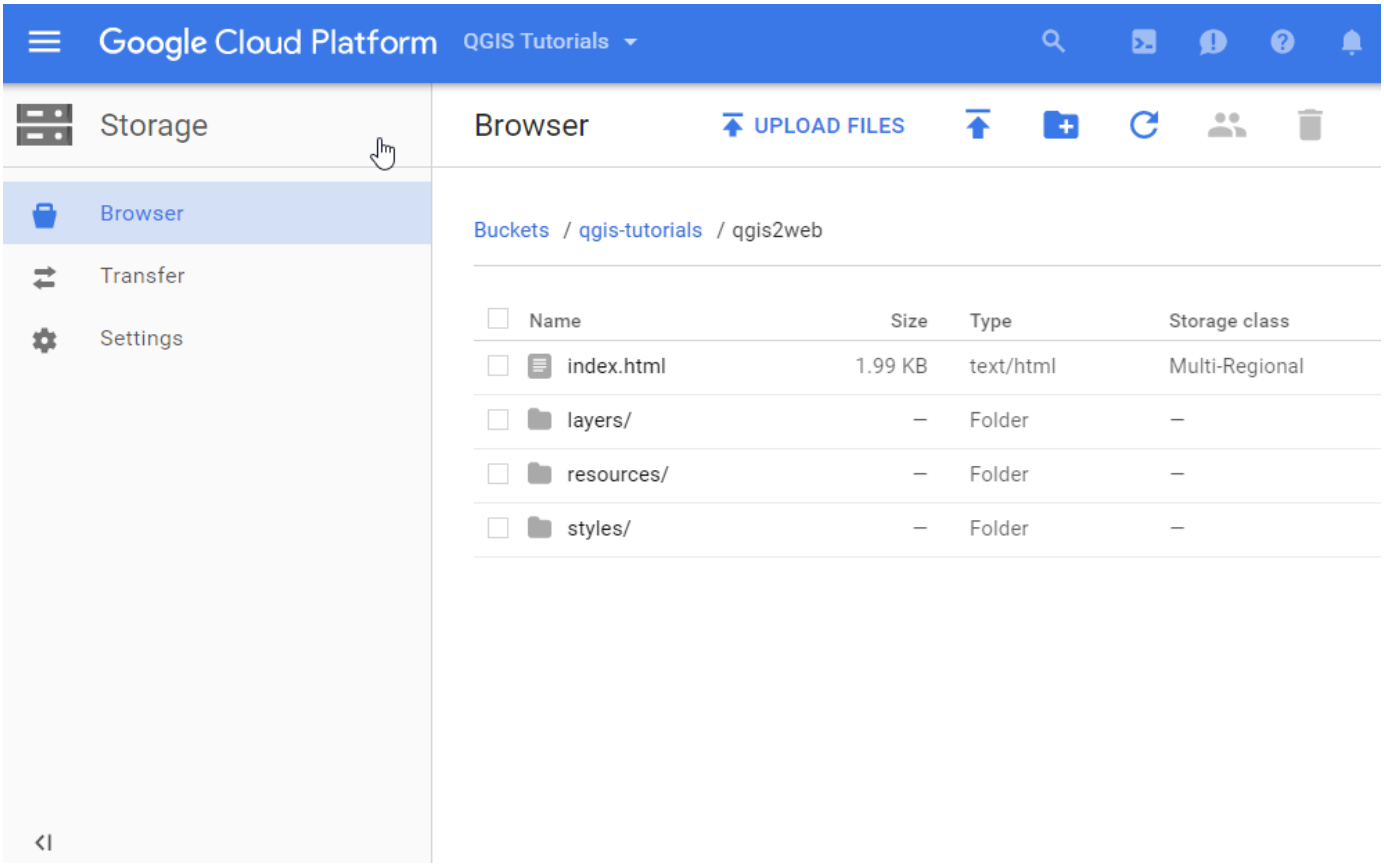
27. Refresh your browser and see that the web map will load with Paris at the center. This is a trivial example, but you can see how you can use any function available in the `OpenLayers` or `Leaflet` libraries to customize the web map.



28. The exported map resides on your computer. While you can see it in action, it is not very useful since you cannot share it anyone. For others to be able to see the map, you need to upload it to a web server. While the upload process will vary on the type of server you have access to - a cheap and easy way to publish your map on the web would be to use any of the public cloud storage services. Amazon S3 (https://aws.amazon.com/s3/) is a popular storage service. You will need to sign up for an account and follow the instructions to create a bucket. Once a bucket is created, you can upload the contents of your exported folder to the bucket and set it to public. Here I created a bucket named `qgis-tutorials` and uploaded the contents of my exported folder to a sub-folder named `qgis2web` . You can access the resulting map at http://s3.amazonaws.com/qgis-tutorials/qgis2web/index.html (http://s3.amazonaws.com/qgis-tutorials/qgis2web/index.html)



29. Similarly, Google also offers a cloud storage service called Google Cloud Storage (https://cloud.google.com/storage/). Once you have created an account and enable billing, you can create a bucket and upload objects to the bucket. I create a bucket and sub-folder similar to Amazon and set the folder to public. The resulting map can be viewed at https://storage.googleapis.com/qgis-tutorials/qgis2web/index.html (https://storage.googleapis.com/qgis-tutorials/qgis2web/index.html)



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Creating Basemaps with QTiles

Tiles have revolutionized the idea of web mapping and has given us fast and easy access to large datasets. Tiling schemes divide the world into small tiles (typically 256 x 256 pixels) for each zoom level and pre-render datasets to these tiles. This way only a small fraction of a large dataset is served to the user at any given time - resulting in a map that can be zoomed or panned with ease over the internet. There are many methods to create tiles from GIS datasets. One easy way to create tiles from your QGIS project is a plugin called **QTiles**. In this tutorial, you will learn how to create PNG tiles from any set of layers loaded in QGIS and create a basemap to be used in a web mapping project.

Overview of the task

We will create tiles from the Natural Earth raster covering the entire planet.

Get the data

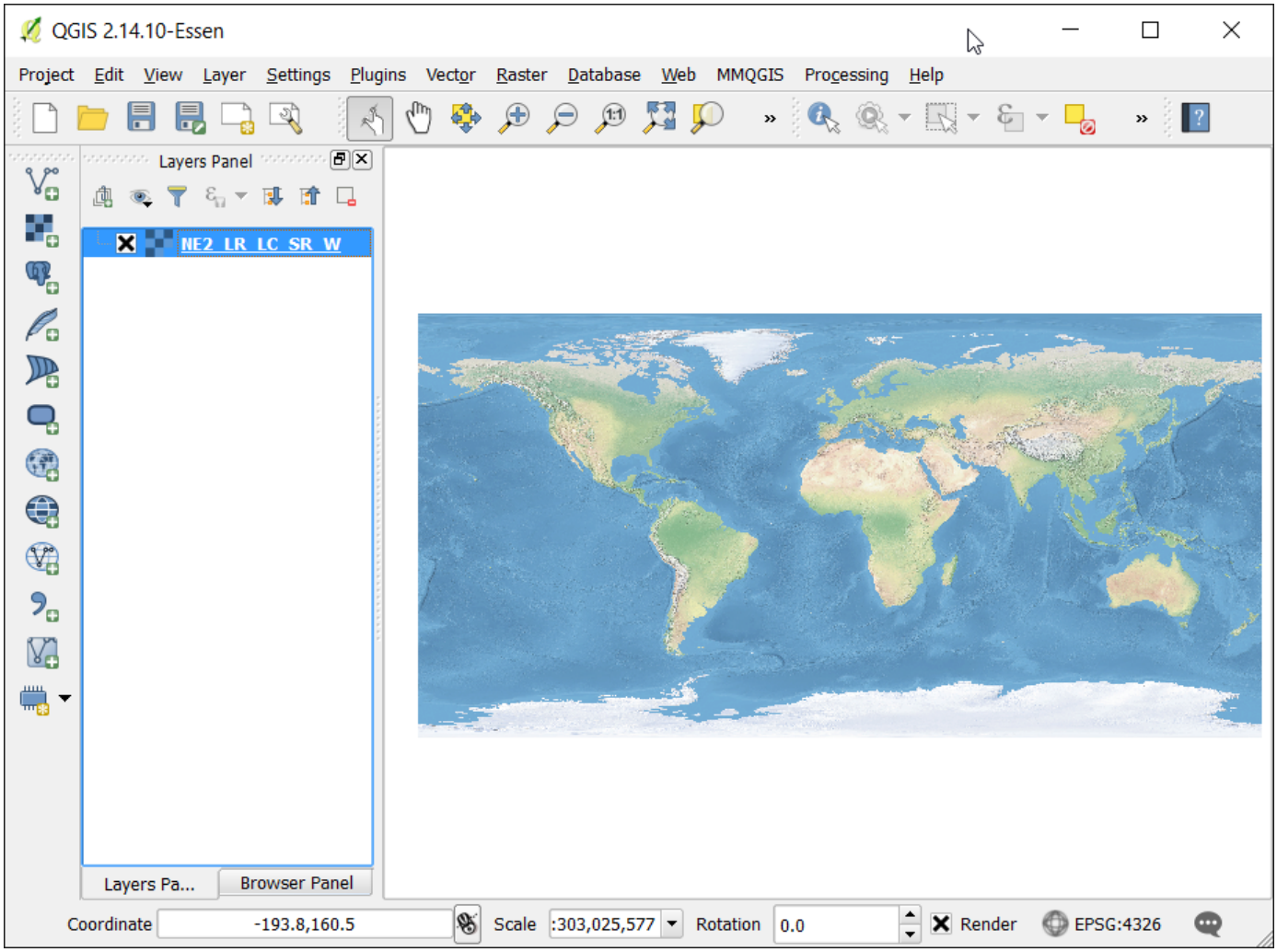
We will use the Natural Earth 2 (http://www.naturalearthdata.com/downloads/10m-raster-data/10m-natural-earth-2/) dataset from Natural Earth.

Download the medium-size Natural Earth II with Shaded Relief, Water, and Drainages (http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/raster/NE2_LR_LC_SR_W_DR.zip) zip file.

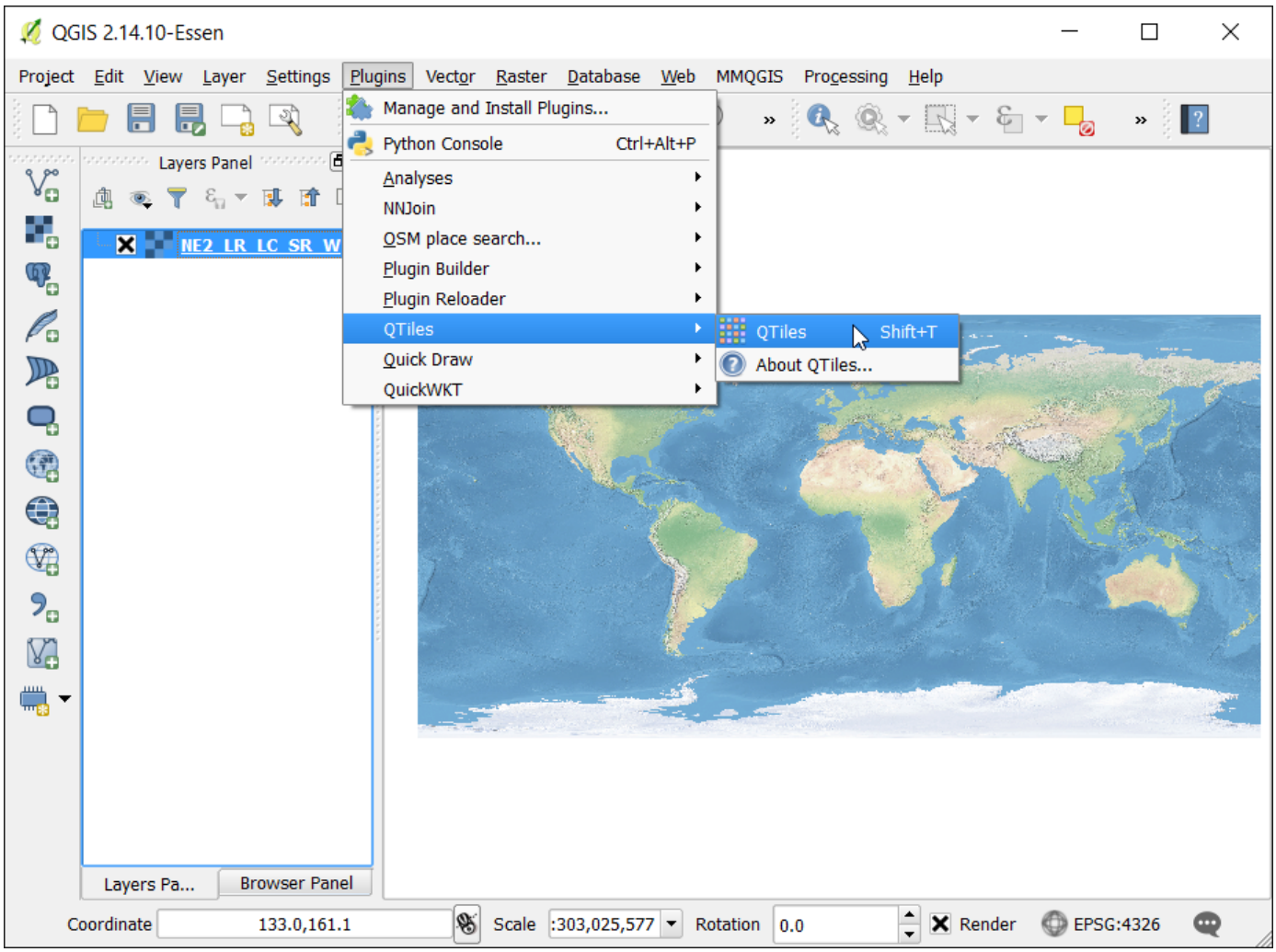
Data Source [NATURALEARTH] (credits.html#naturalearth)

Procedure

1. Unzip the downloaded `NE2_LR_LC_SR_W.zip` file to a folder on your computer. Open QGIS and go to Layer ‣ Add Raster Layer. Browse to the location of the extracted files and select `NE2_LR_LC_SR_W.tif`. Click OK.



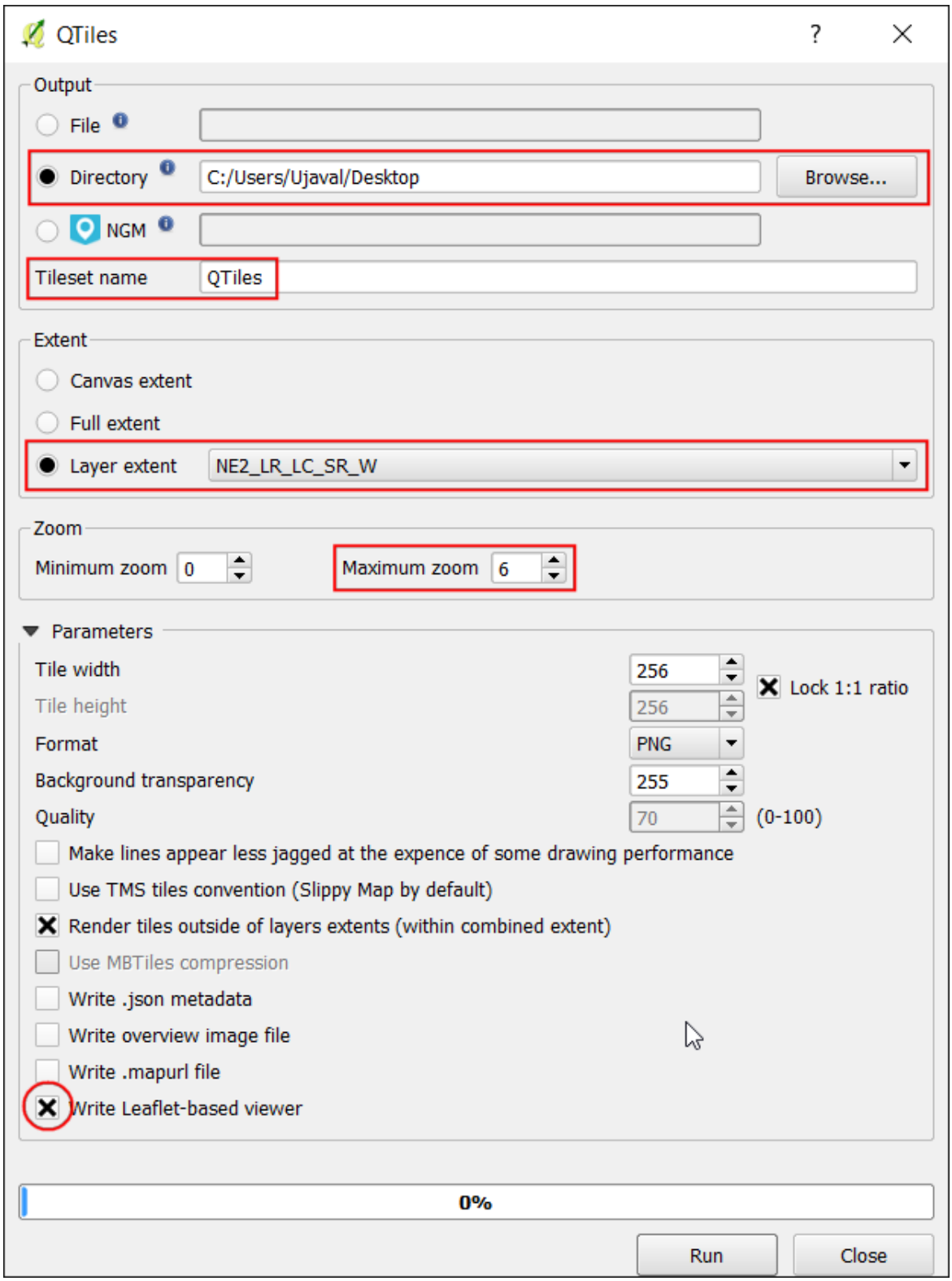
2. Install the `QTiles` plugin by going to Plugins ‣ Manage and Install Plugin. Note that the plugin is currently marked **experimental**, so you will need to check Show also experimental plugins in Plugin Settings. (See *Using Plugins* (using_plugins.html) for more details on installing plugins in QGIS). Once the plugin is installed, go to Plugins ‣ QTiles ‣ QTiles.



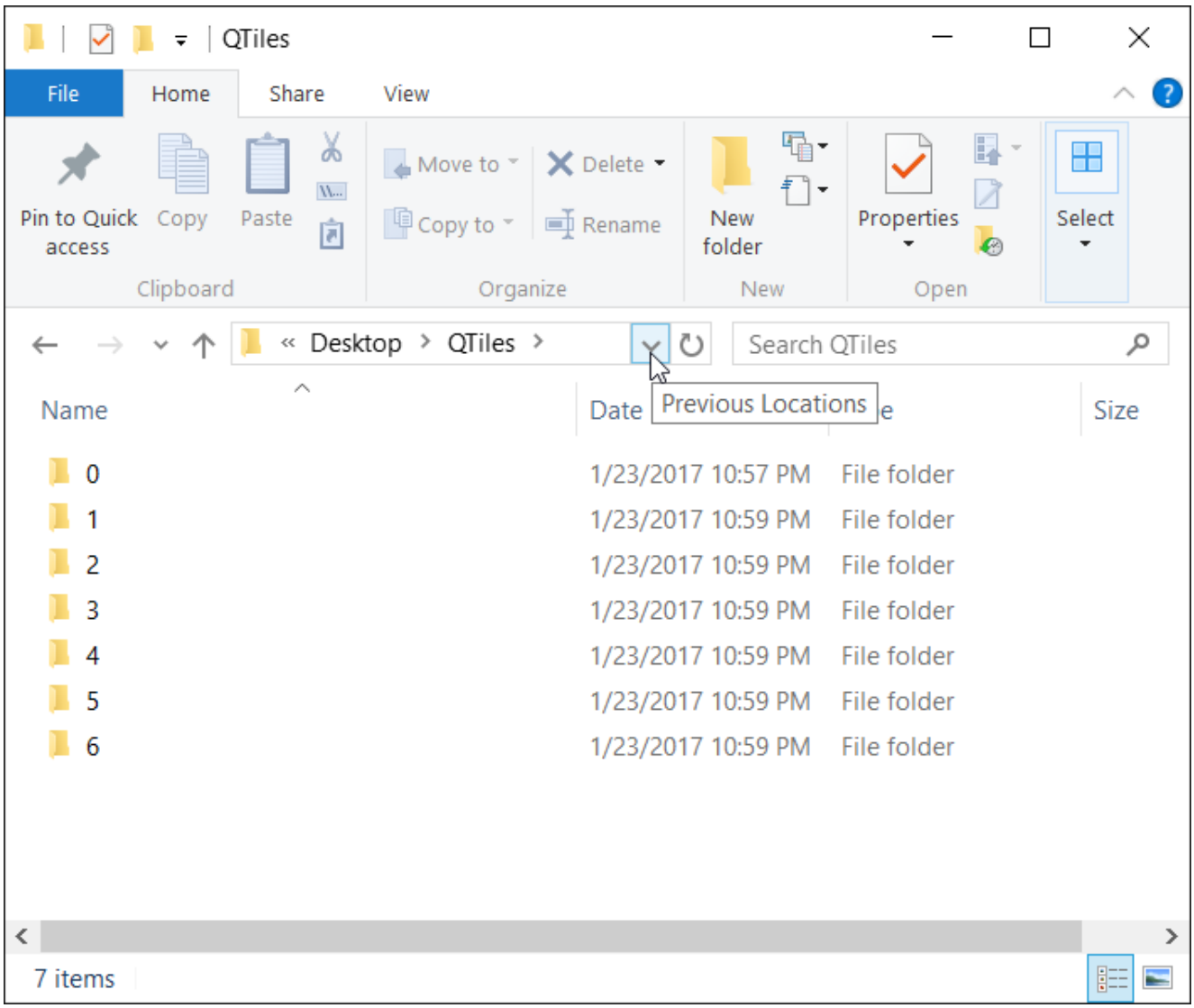
3. In the QTiles dialog, select Directory as the Output and browse to a folder of your choice where the output tiles will be created. Choose Layer extent of the `NE2_LR_LC_SR_W` layer as the extent of the tiles. Set the Maximum Zoom to `6`. Expand the Parameters section and check the Write Leaflet-based viewer. Click Run to start the process of rendering the tiles.

Note

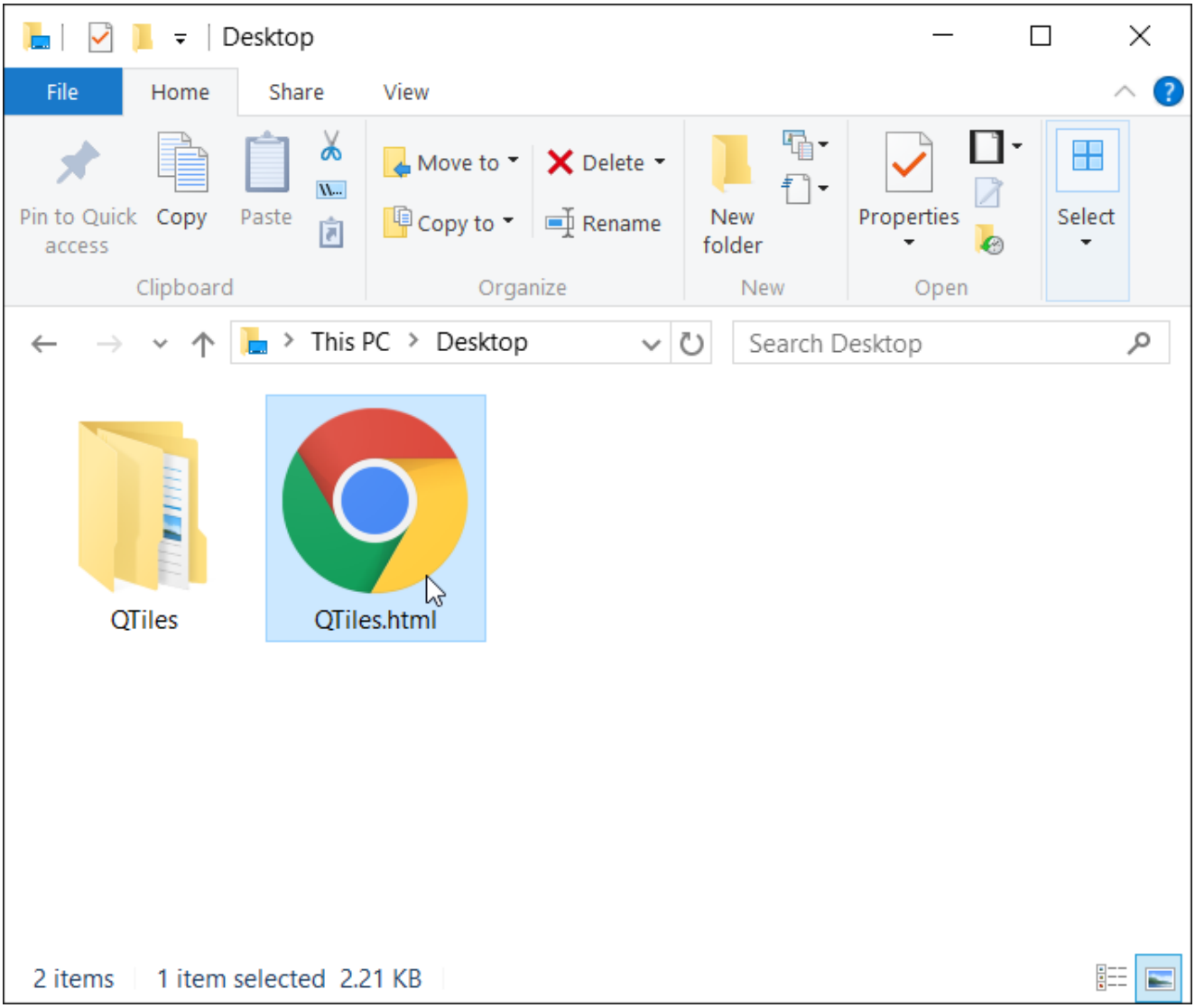
The number of tiles increase 4 times for every additional zoom level and since our layer has an extend of the entire world - there will be millions of tiles at higher zoom levels.



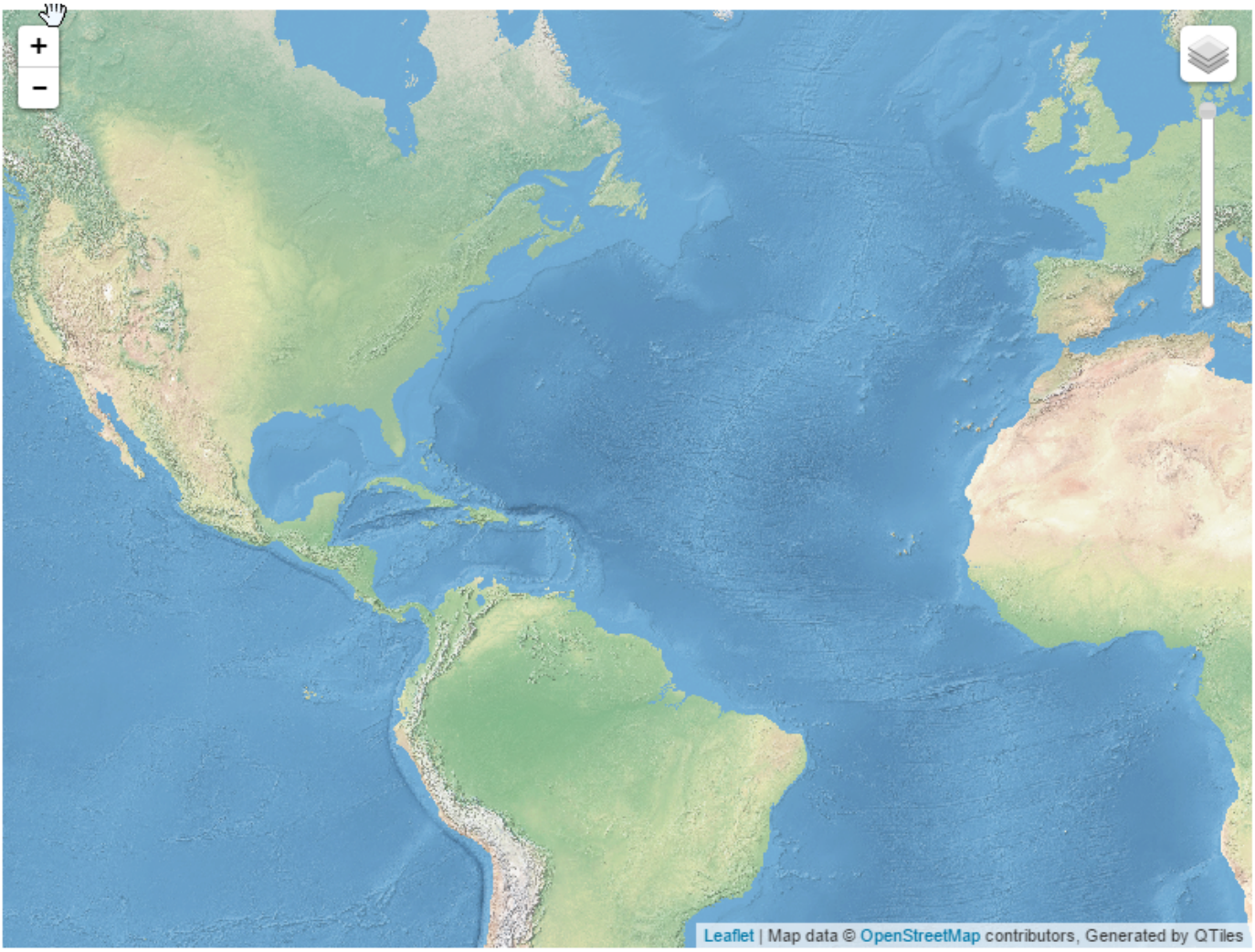
4. Once the process finishes, close the QTiles dialog and browse to the output folder you had selected. You will notice folders for each zoom level upto the maximum zoom level. Each folder further contains subfolder for X coordinates and then the actual tiles named for Y coordinates.



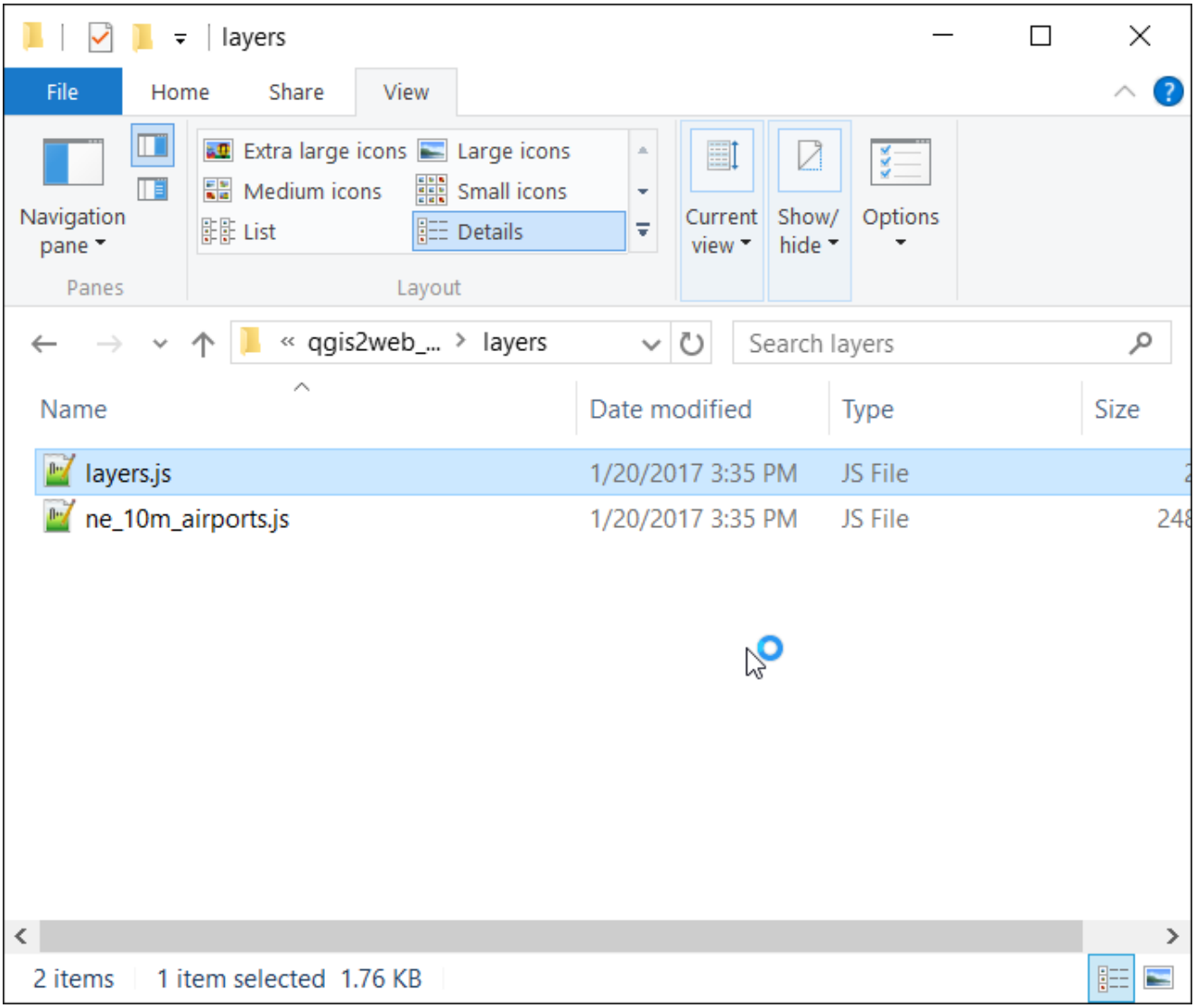
5. In the parent directory of top-level tiles directory, you will find a `QTiles.html` file. This is a simple viewer to explore the tiles using the Leaflet web mapping library.



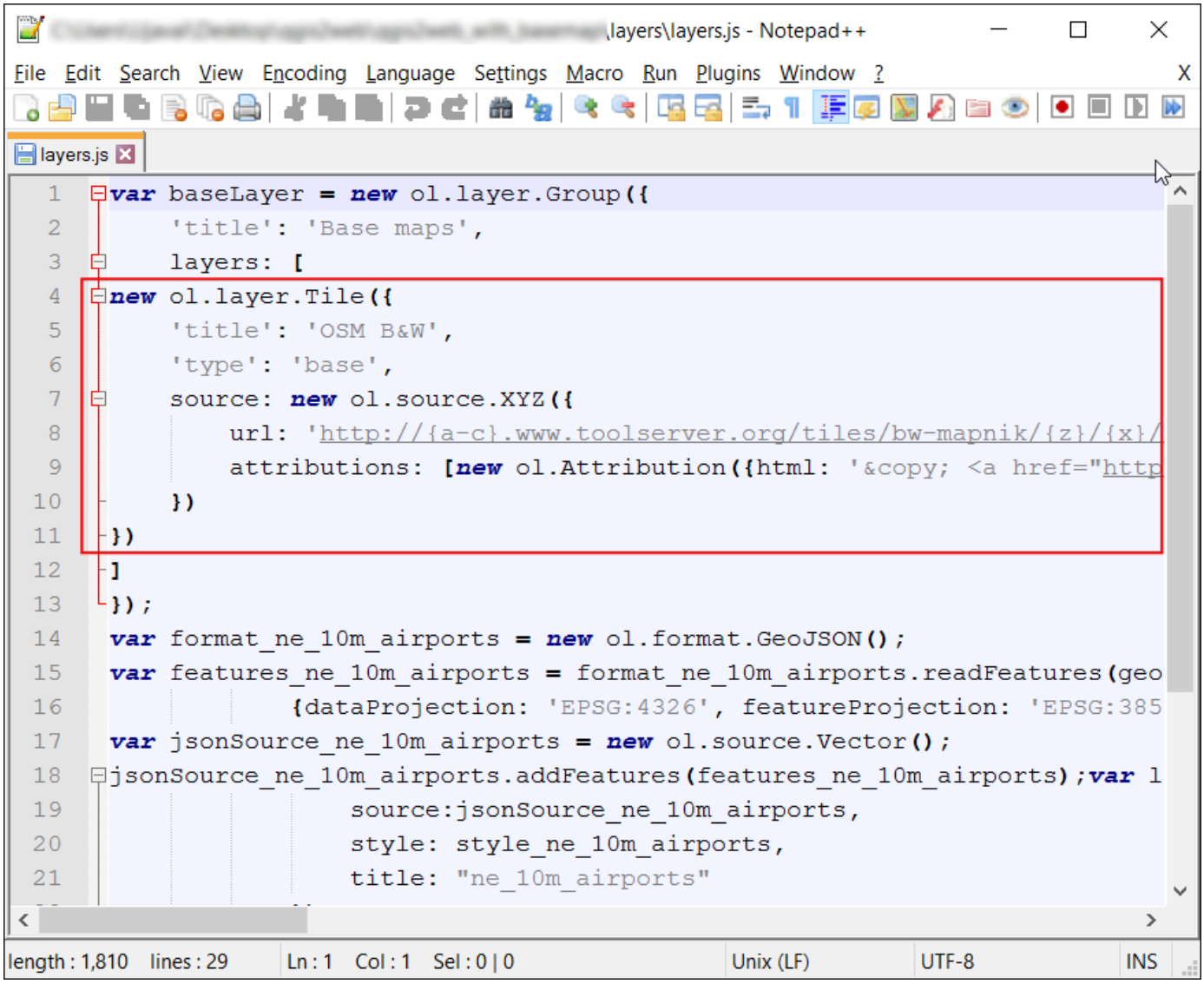
6. Double-click the `QTiles.html` to open it in a web browser. You can zoom and pan around to see the tiles seamlessly form the original raster layer.



7. You can use these tiles with any web-mapping library that supports XYZ tiling schemes and overlay other layers on top. To demonstrate the usefulness and portability of such tiles, we will now add the tiles created in this tutorial as the basemap for the airports map created in *Web Mapping with QGIS2Web* (web_mapping_with_qgis2web.html) tutorial. In that tutorial, we chose to use a ready-made basemap from OpenStreetMap. We can easily swap that with our own custom basemap created in this tutorial. Go to the output directory where the qgis2web map was exported. Open the Layers ▸ layers.js file created during the export.

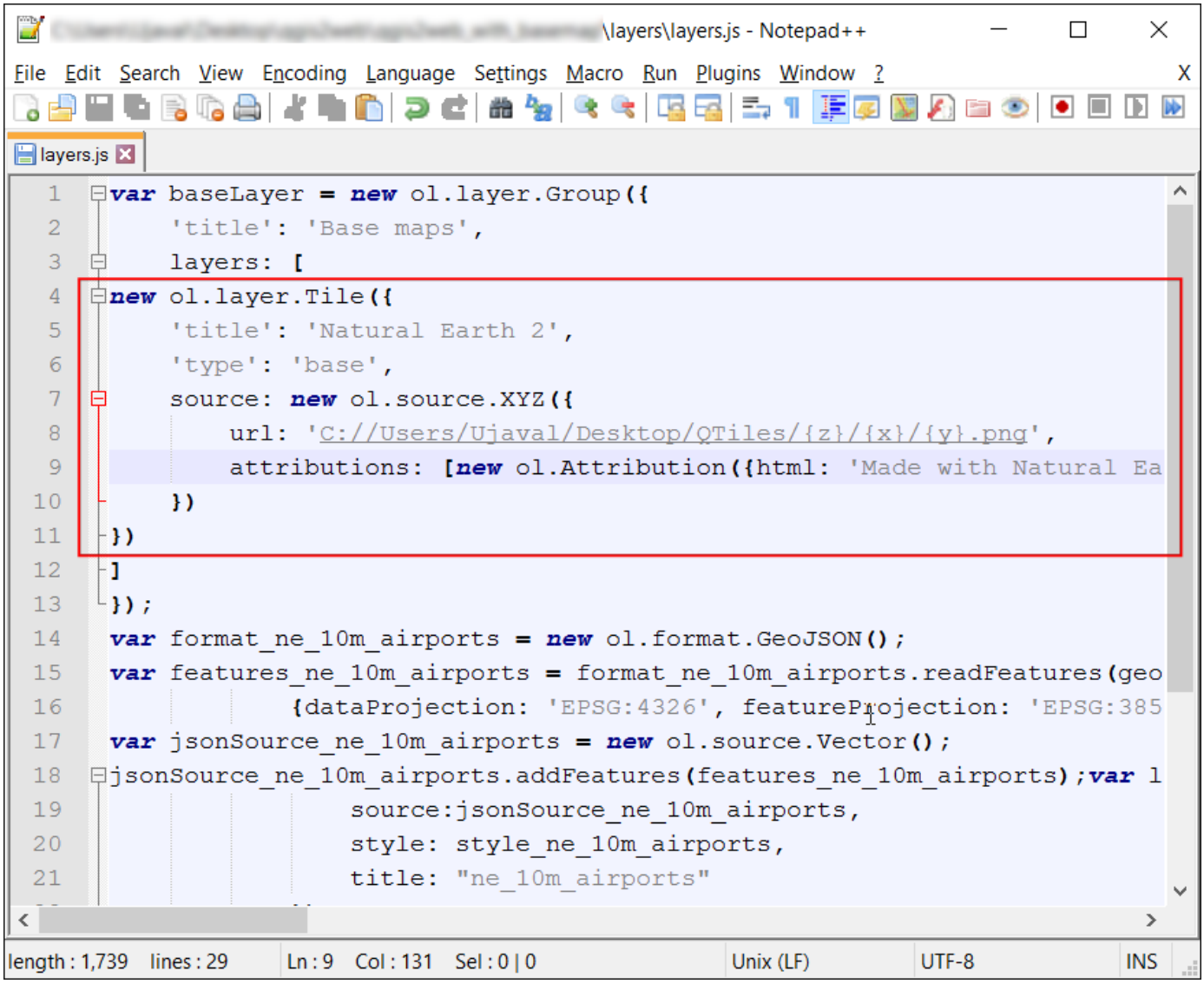


8. Locate the code block where the `OSM B & W` base layer is defined.

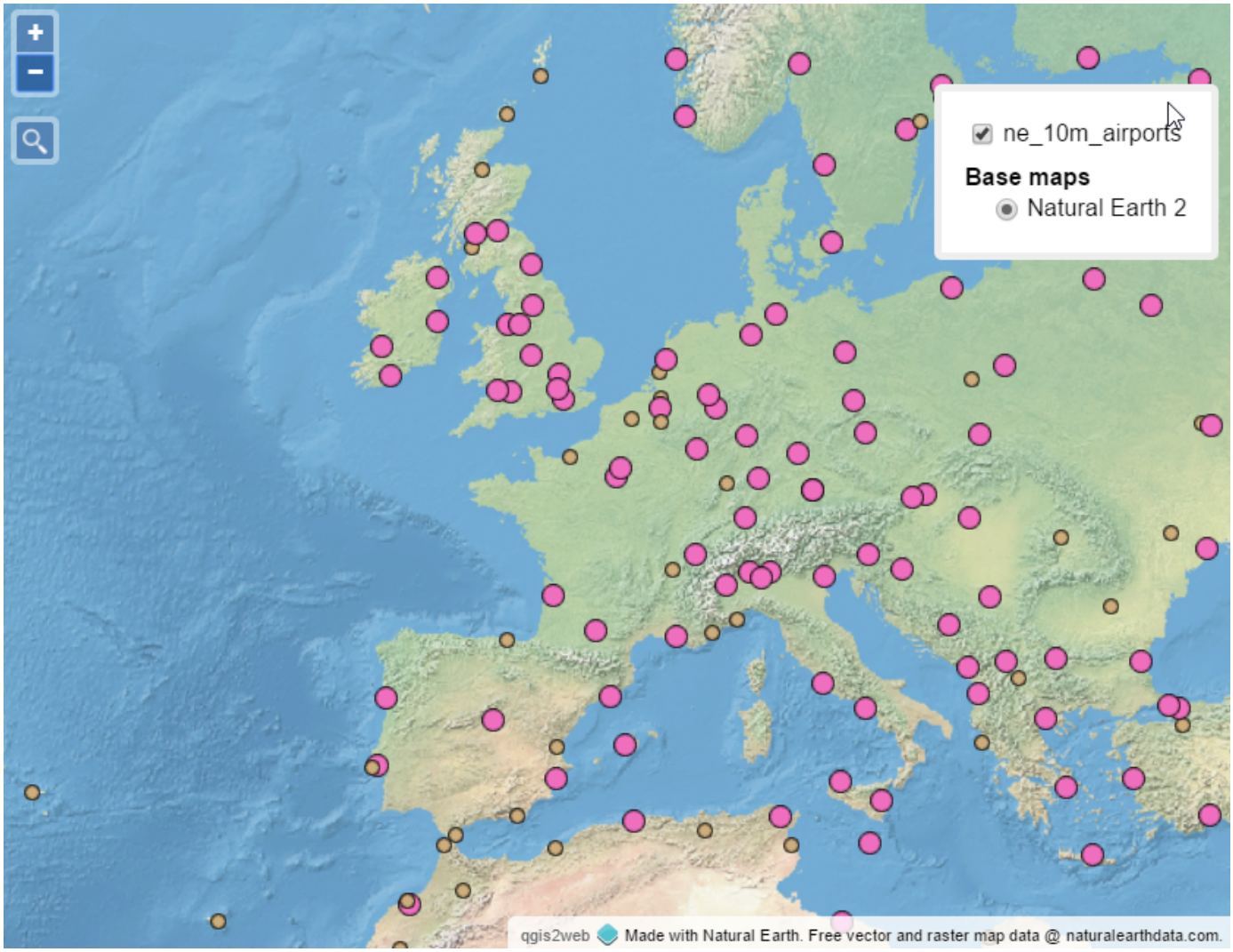


9. Replace the definition of the base layer with our own tiles. At this point, the tiles exist only on your computer, so the URL will be the local directory. But you can also upload the tiles to a server and give the URL of the server. Change the title and source with appropriate values for Natural Earth. Save the file.

```
new ol.layer.Tile({
 'title': 'Natural Earth 2',
 'type': 'base',
 source: new ol.source.XYZ({
     url: 'C://Users/Ujaval/Desktop/QTiles/{z}/{x}/{y}.png',
     attributions: [new ol.Attribution({html: 'Made with Natural Earth. Free vector and raster map dat
 })
})
```



10. Open the web map in a browser and you will see that the B & W OSM layer is replaced by our freshly created tiles.



Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Back to top

Author

Ujaval Gandhi

Follow @spatialthoughts

Using Plugins

Plugins in QGIS add useful features to the software. Plugins are written by QGIS developers and other independent users who want to extend the core functionality of the software. These plugins are made available in QGIS for all the users.

Overview of the task

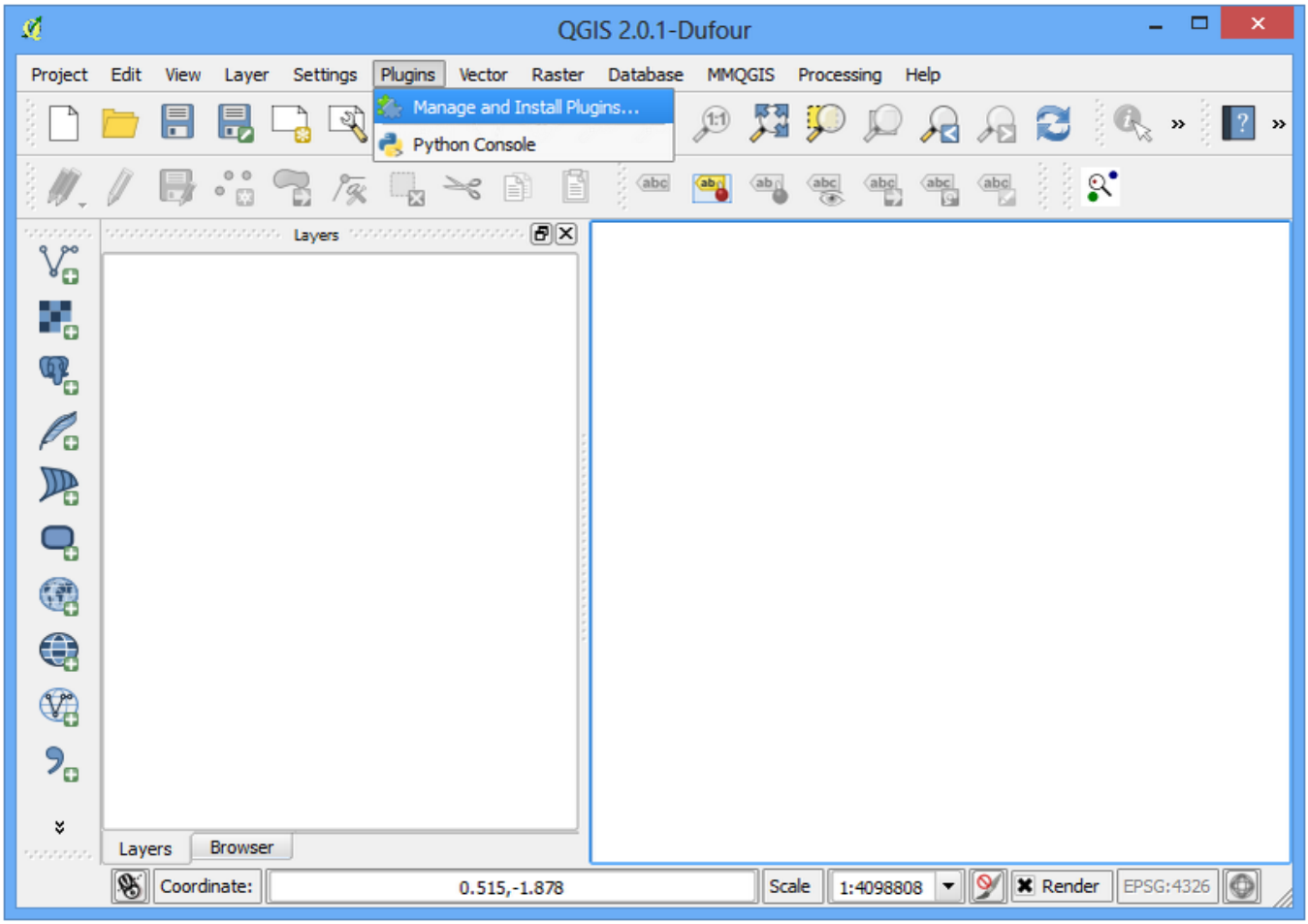
In this tutorial, you will learn how to enable *Core Plugins* as well as download and install *External Plugins*. You will also learn how to locate the plugin from the QGIS menu once they are installed.

Procedure

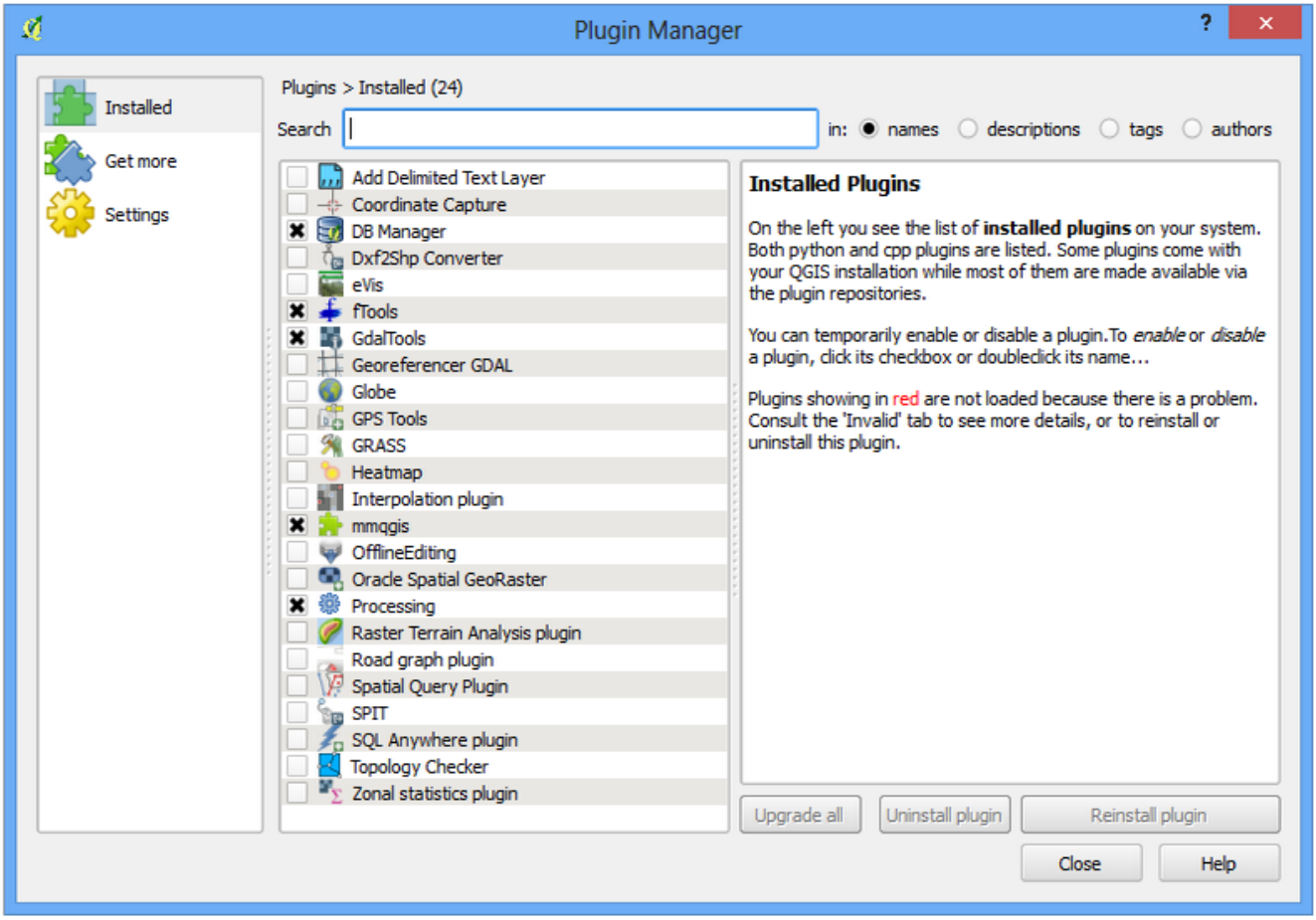
Core Plugins

Core plugins are already part of the standard QGIS installation. To use these, you just need to enable them.

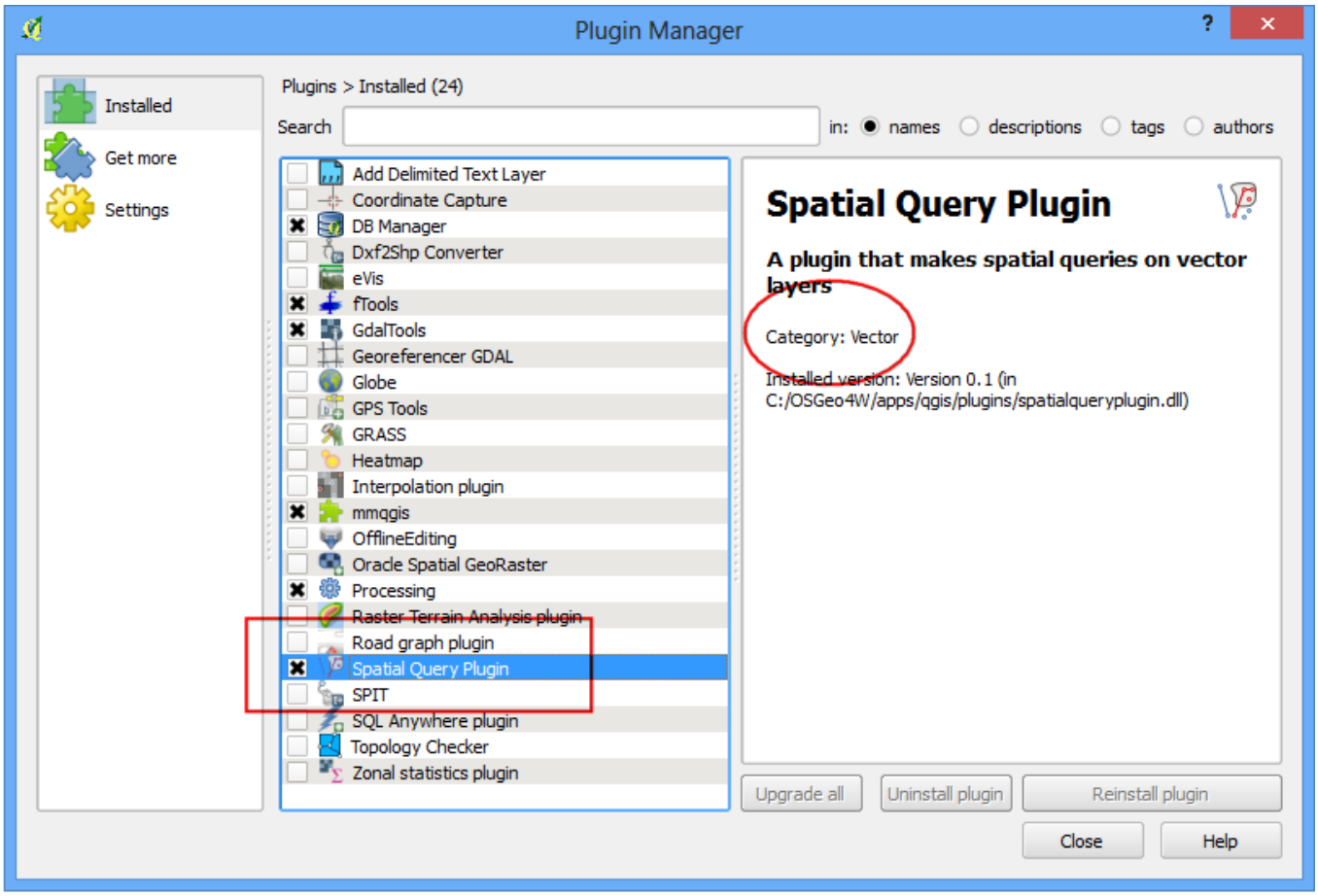
1. Open QGIS. Click on Plugins ‣ Manage and Install Plugins.... to open the Plugin Manager dialog.



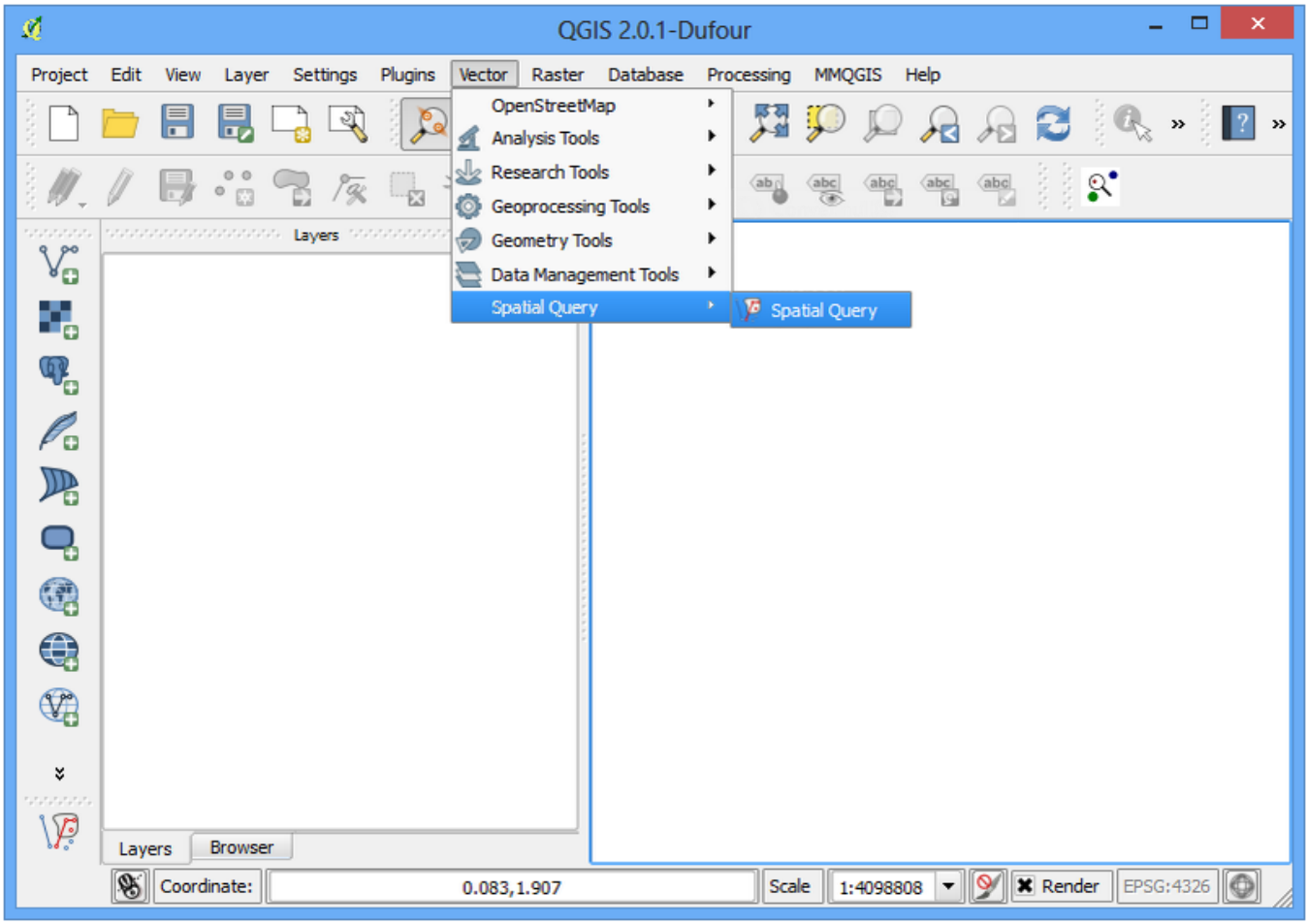
2. Even if this is your first time using QGIS, you will see a lot of plugins listed under the Installed tab. This is because they are *Core Plugins* and were installed during QGIS installation.



3. Let's enable one of the plugins. Check on the checkbox next to Spatial Query Plugin. This will enable the plugin and you will be able to use it. One thing to note is that plugins have the ability to insert menu items at various locations and create new panels and toolbars. Sometimes it is difficult to know how to find the newly enabled tools. Once clue is to look in the plugin discription. Here the description says *Category: Vector*. That indicates that the plugin would be found under the Vector menu once enabled. Click Close.



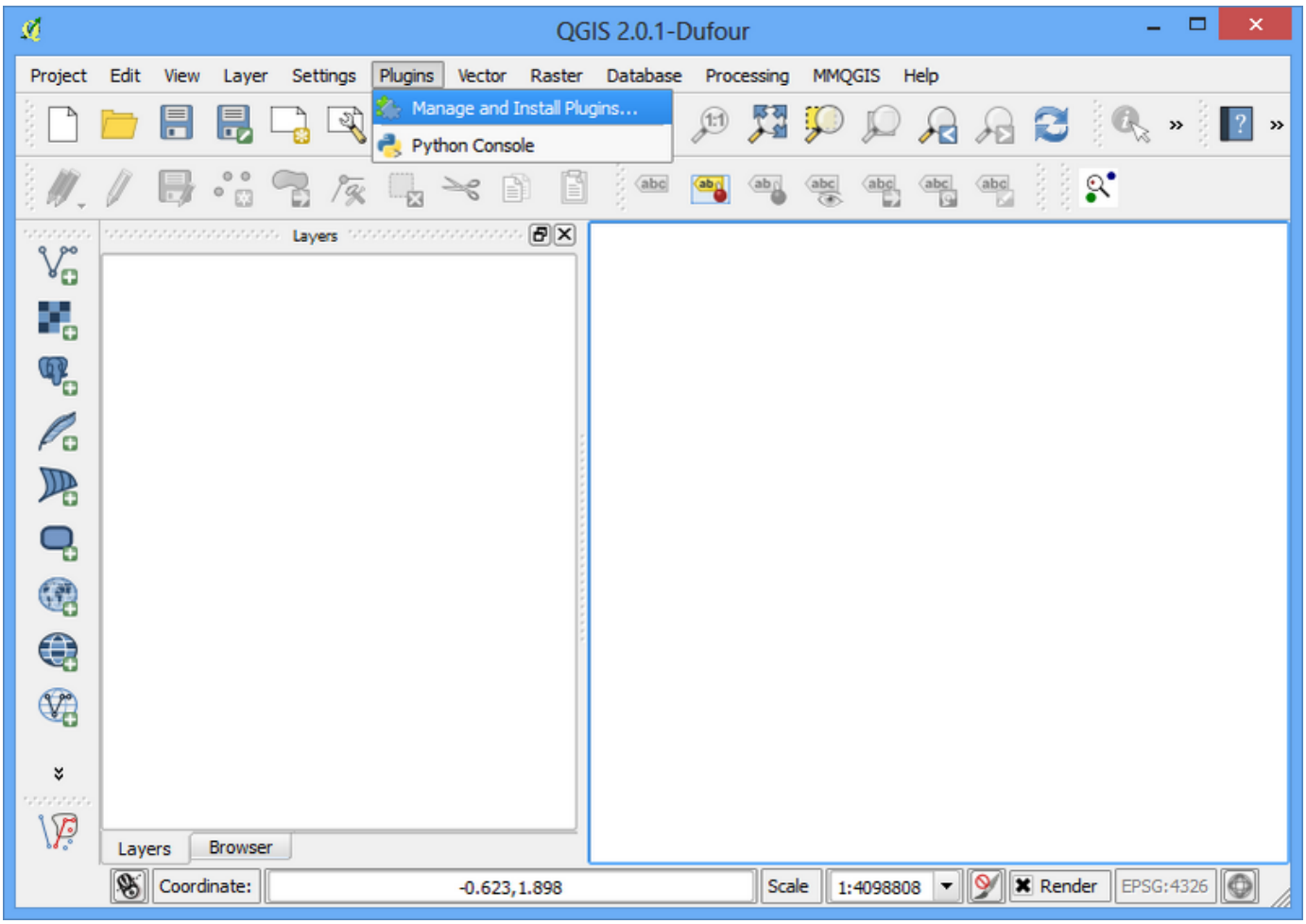
4. Now that the Spatial Query Plugin is enabled, you can go to the Vector ‣ Spatial Query to use the functionality added by the plugin.



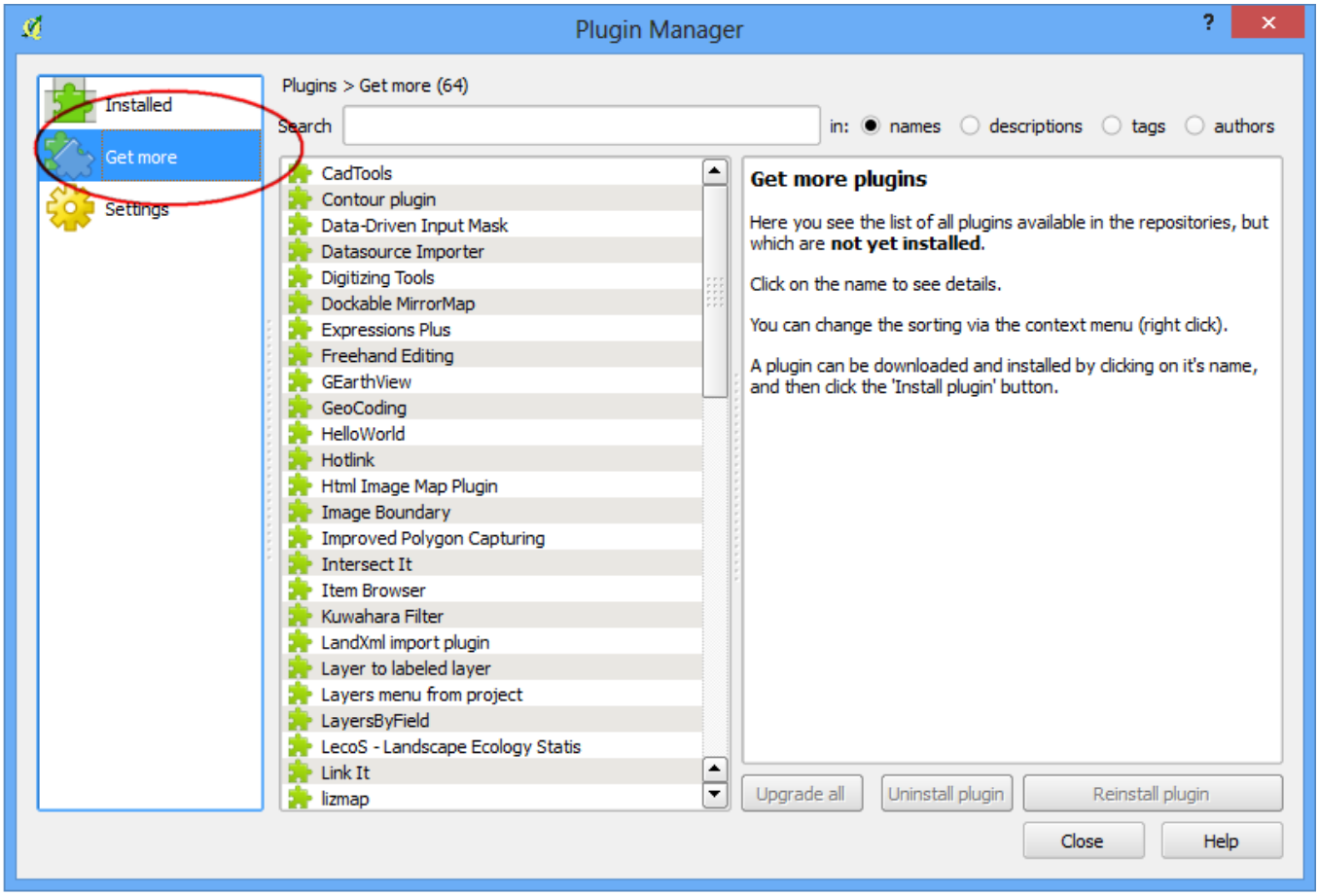
External Plugins

External plugins are available in the QGIS Plugins Repository (http://plugins.qgis.org/) and need to be installed by the users before using them. An easy way to browse and install these plugins is by using the Plugin Manager tool.

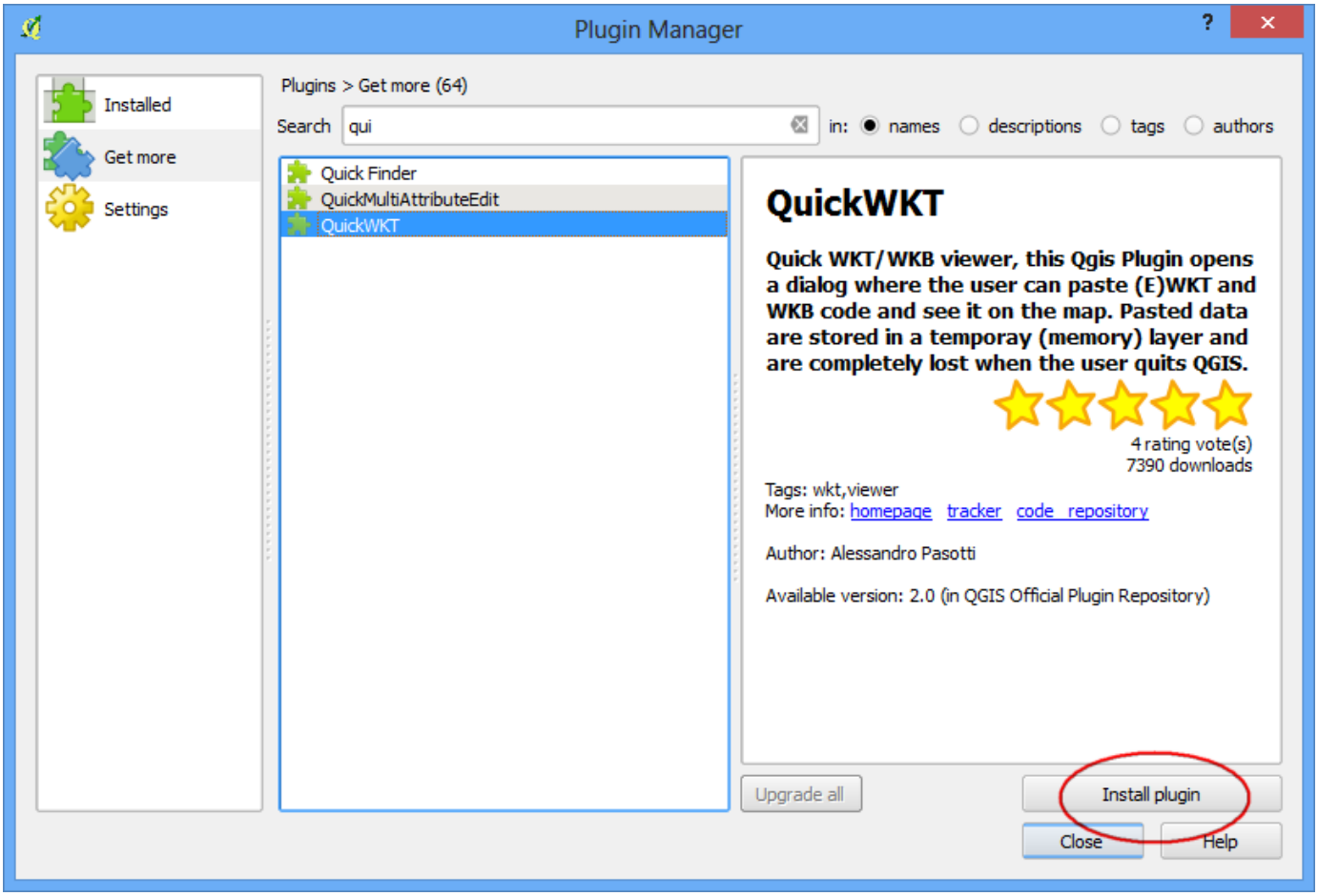
1. Open QGIS. Click on Plugins ‣ Manage and Install Plugins.... to open the Plugin Manager dialog.



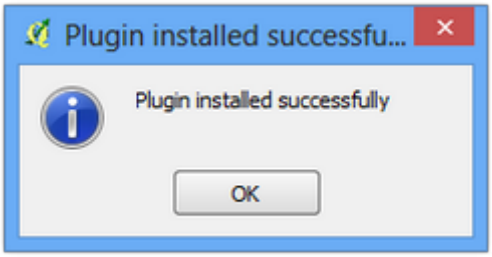
2. Click on Get more tab. Here you will see a list of plugins listed.



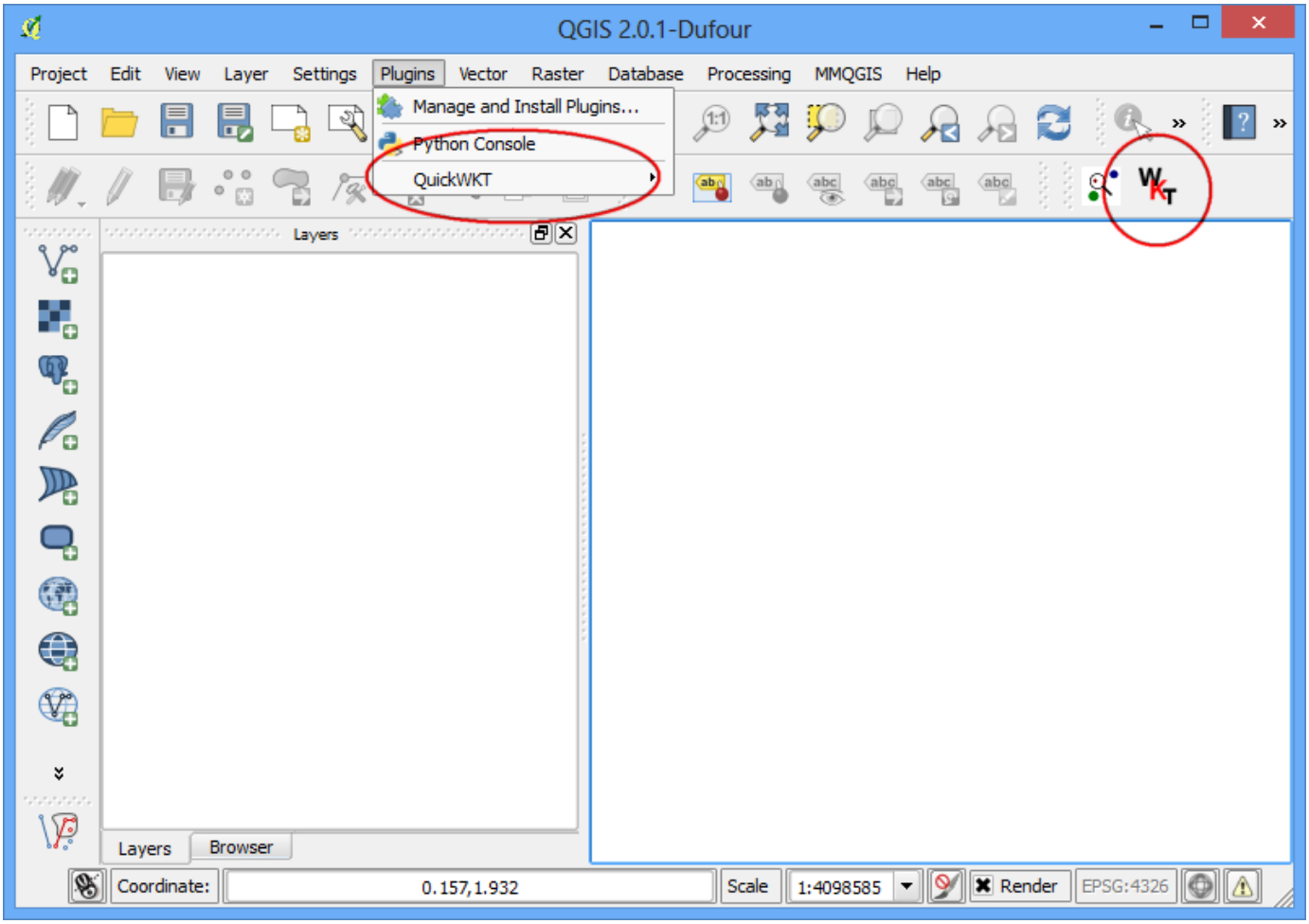
3. For this tutorial, let's find and install a plugin called 'QuickQKT'. As you start typing *qui* in the search box, you will see the search results below. Click on the QuickWKT. Next, click on Install plugin button to install it.



4. Once the plugin is downloaded and installed, you will see a confirmation dialog.



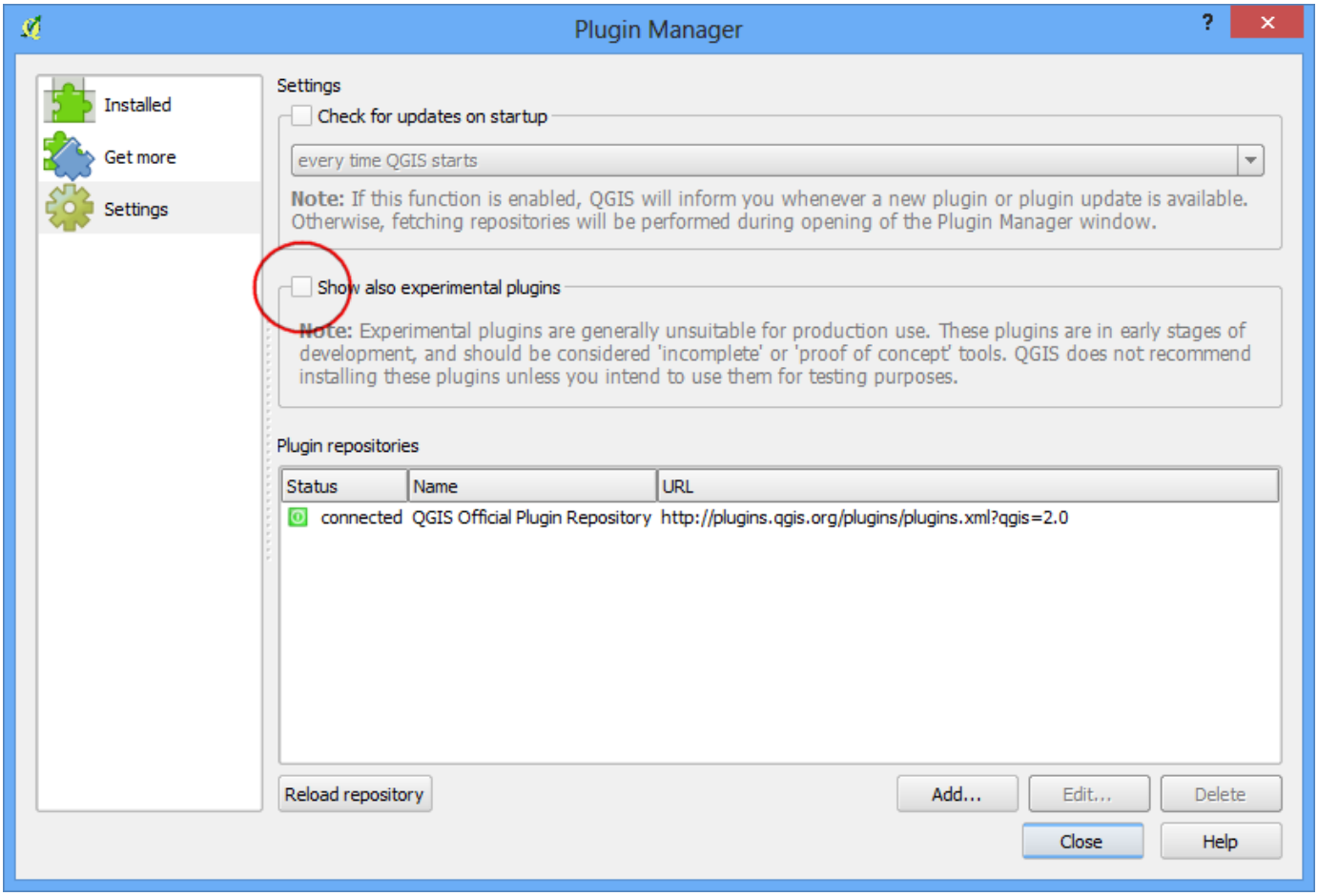
5. If you noticed, there was no mention of the plugin category in the description. That makes it hard to determine how to access the newly installed plugin. Most external plugins are installed under the Plugins menu itself in QGIS. Click on Plugins ‣ QuickWKT and you will see the newly installed plugin. Usually, external plugins also install a button in the Plugins toolbar also. You may also use that button to access the plugin.



Experimental Plugins

Now you know how to install and find an *External Plugin* in QGIS. Let's explore some advanced options. Sometimes you are looking for a specific plugin, but cannot find it in the Get more tab. It maybe because the plugin is marked *Experimental*. Here is how to install *experimental* plugins.

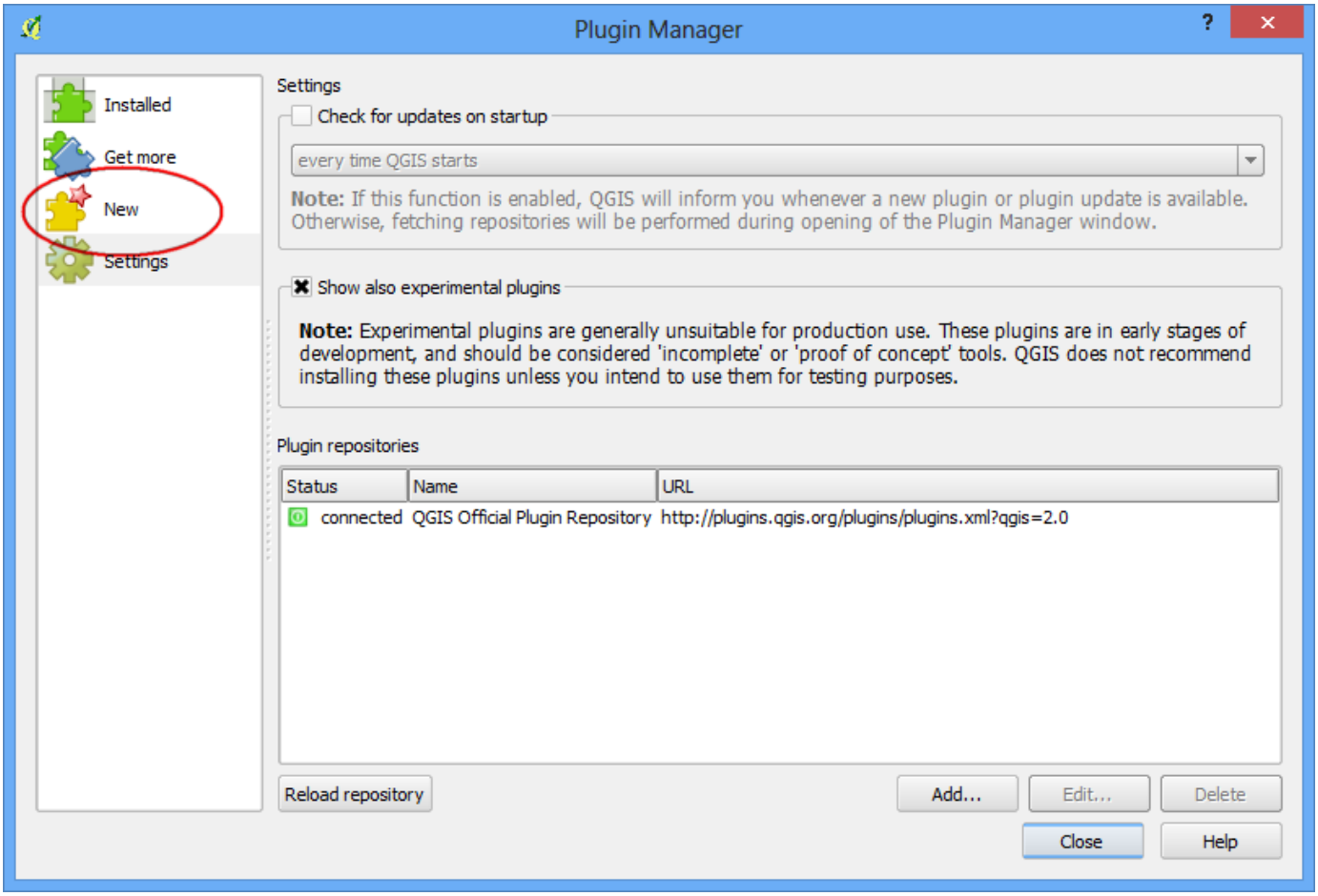
1. Open Plugin Manager by Plugins ▸ Manage and Install Plugins.... Click on the Settings tab. You will see an option called Show also experimental plugins. Click the checkbox next to it, to enable it.



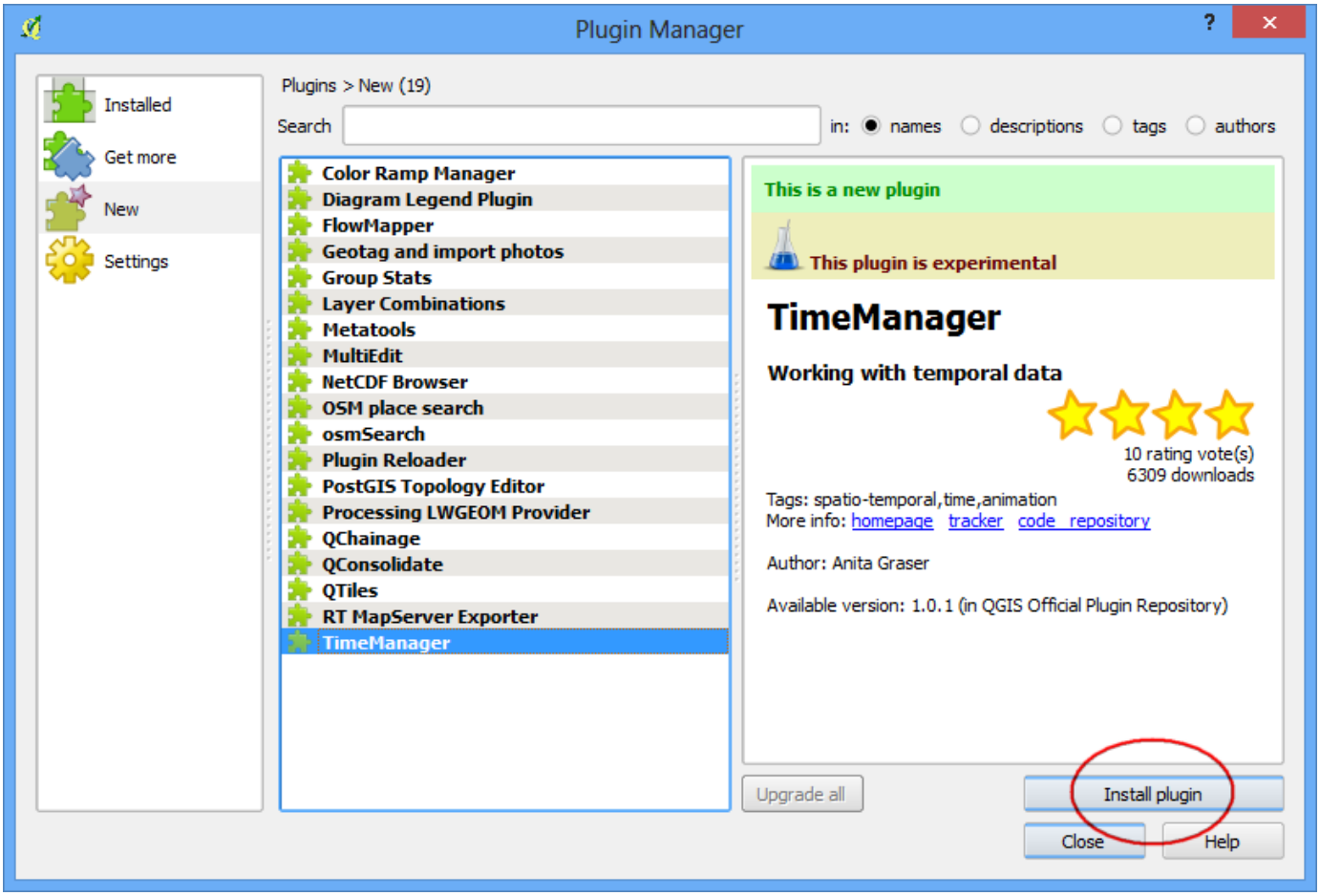
2. You will see a new tab called New. The newly enabled experimental plugins will show up here.

Note

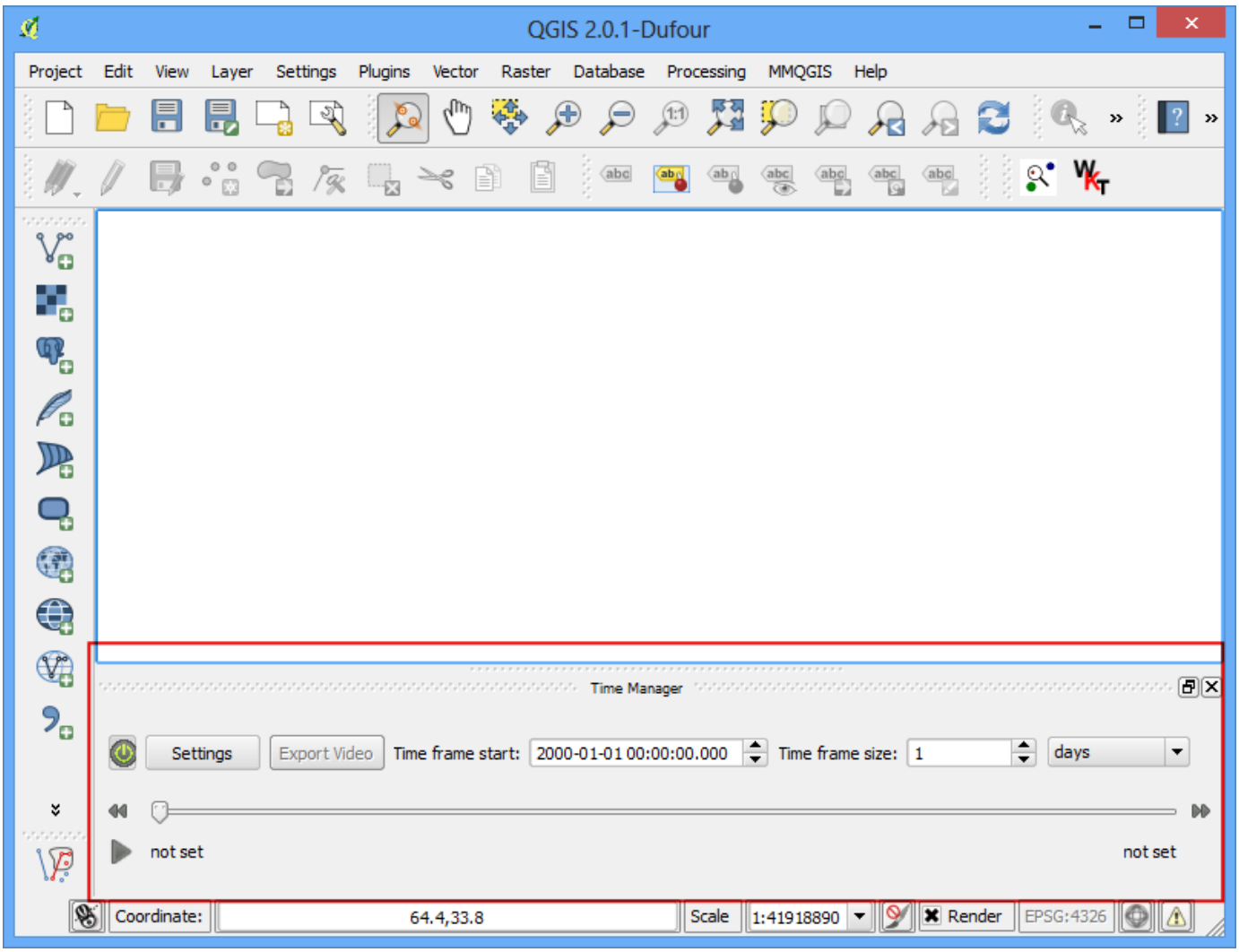
The New tab will appear only temporarily once you enable the experimental plugins. The next time you open Plugin Manager, the experimental plugins will show alongside regular plugins in the Get more tab.



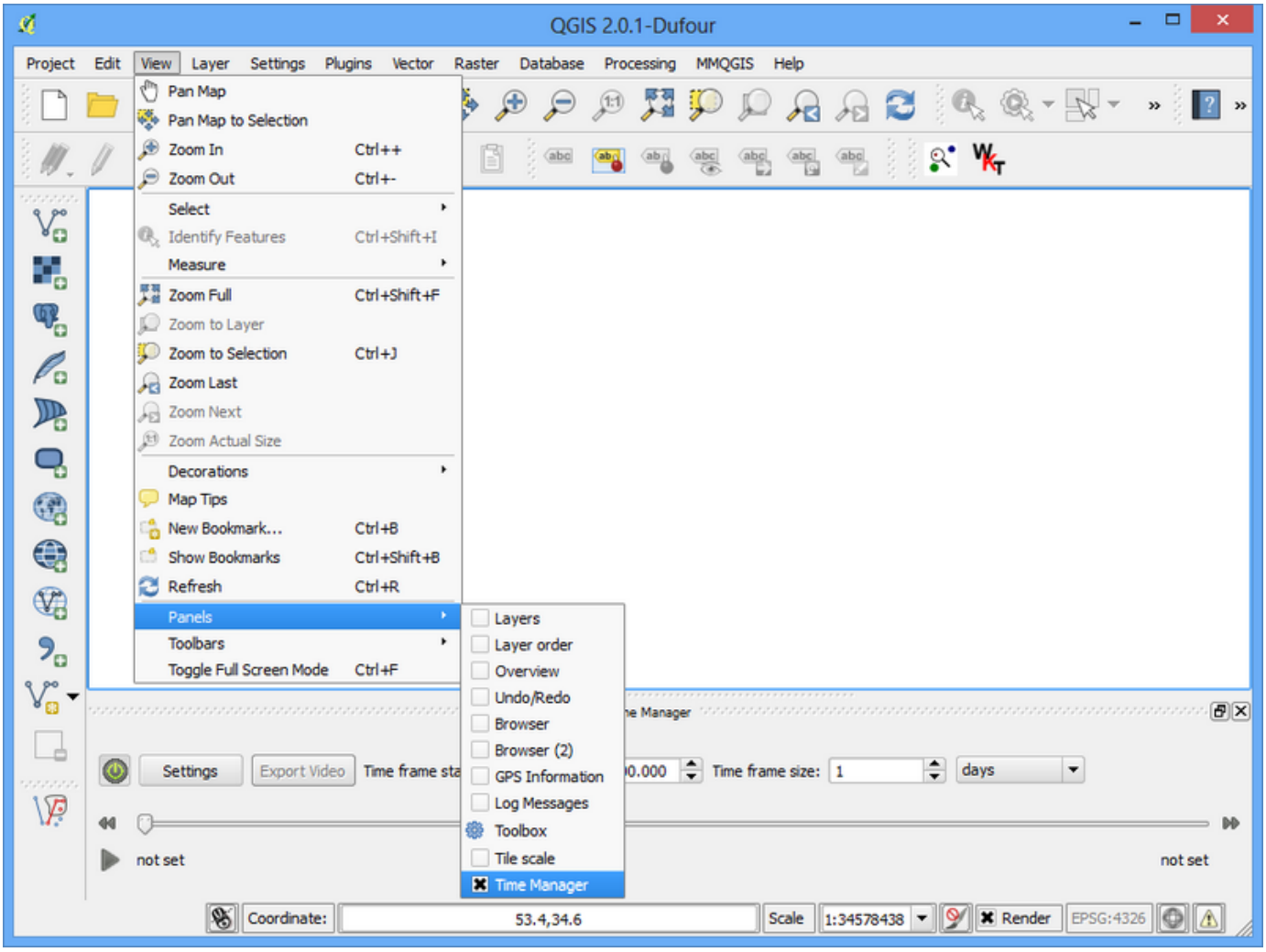
3. Let's install a plugin called TimeManager. Click on the plugin name and then Click Install.



4. Now when you come back to the main QGIS window, you will see a new *Panel* at the bottom of the canvas. This panel is created by the TimeManager plugin. This is yet another way of plugins to add useful functionality to the user interface .



5. You can enable/disable this panel from View ▸ Panels ▸ Time Manager.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.

Author

Ujaval Gandhi

Follow @spatialthoughts

Searching and Downloading OpenStreetMap Data (QGIS3)¶

Getting high quality data is essential for any GIS task. One great resource for free and openly licensed data is OpenStreetMap(OSM) (http://www.openstreetmap.org/) . The OSM database consists of all types of mapping data - streets, local data, building polygons, administrative boundaries etc. Getting access to OSM data in a GIS format in QGIS is possible via the *QuickOSM** plugin. This tutorial explains the process for searching, downloading and using this plugin.

Overview of the task

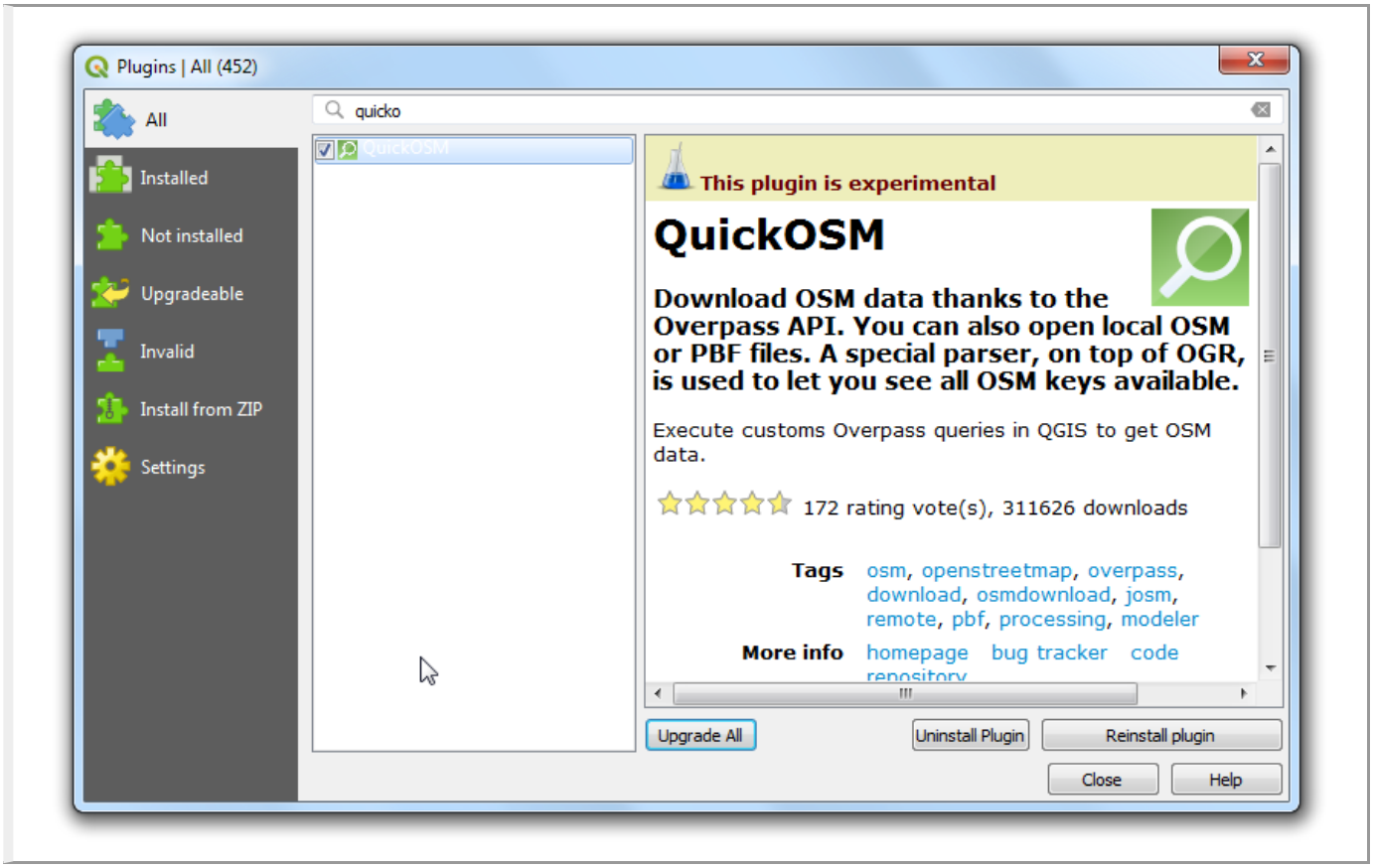
We will extract locations of all bars and pubs in London from the OpenStreetMap database and save it as a vector layer.

Other skills you will learn

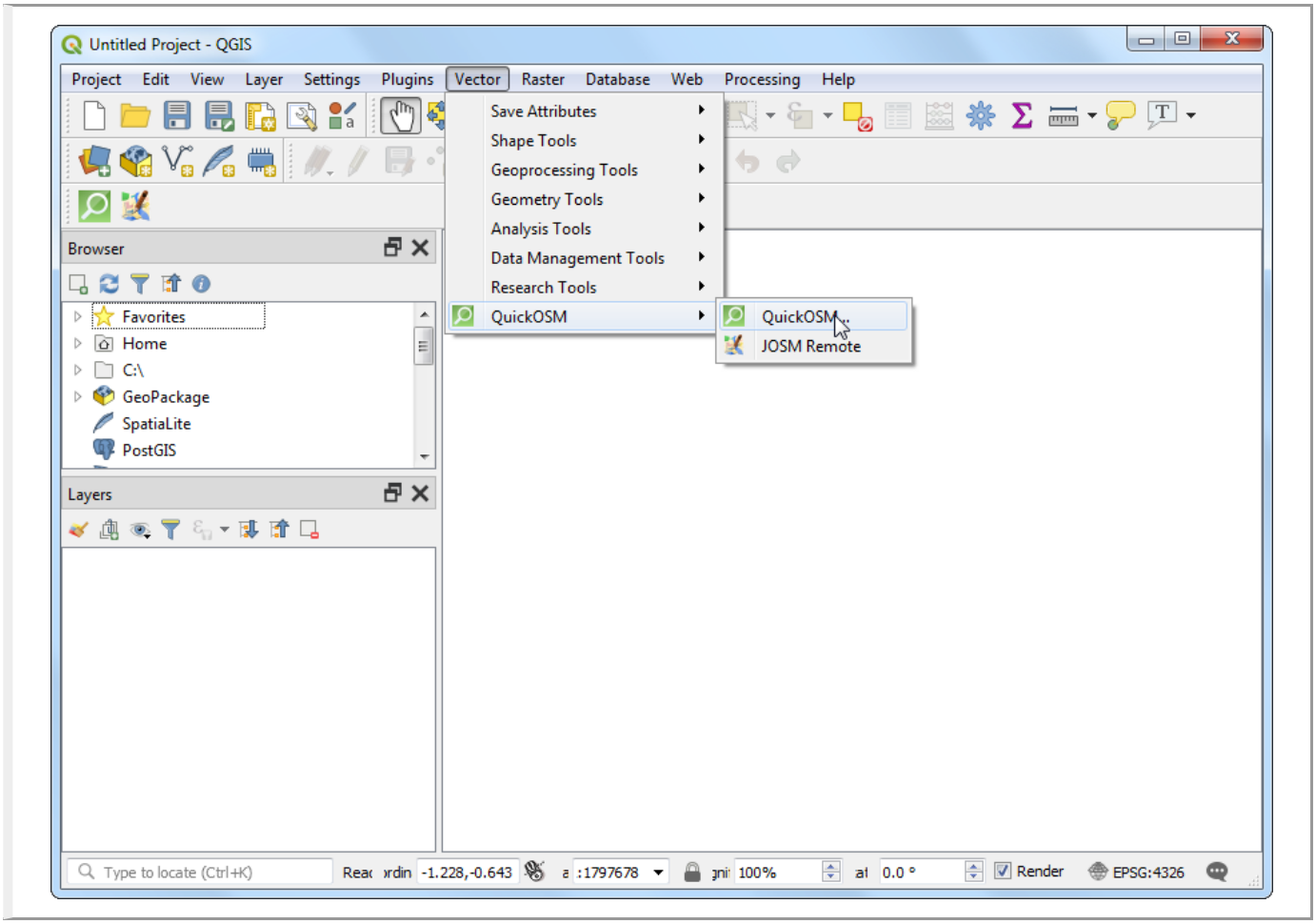
- How to Merge multiple vector layers.

Procedure

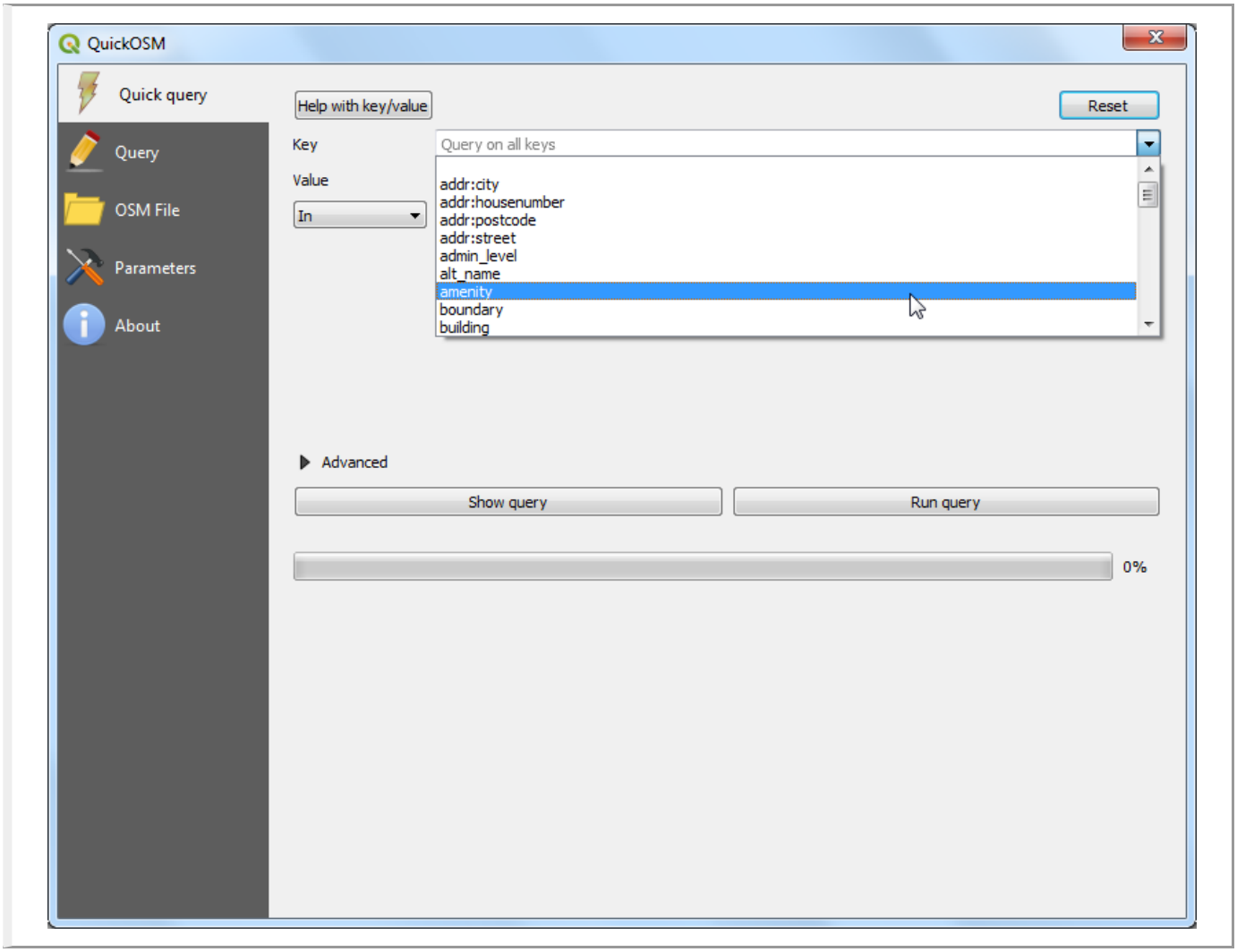
1. Search and install the **QuickOSM** plugin from the QGIS Official Plugin Repository. See *Using Plugins* (../using_plugins.html) for instructions on downloading plugins. Note that at the time of writing this tutorial, this plugin is marked as *Experimental*, so make sure you check `Show also experimental plugins` in the Settings tab in the Plugins dialog to be able to install it.



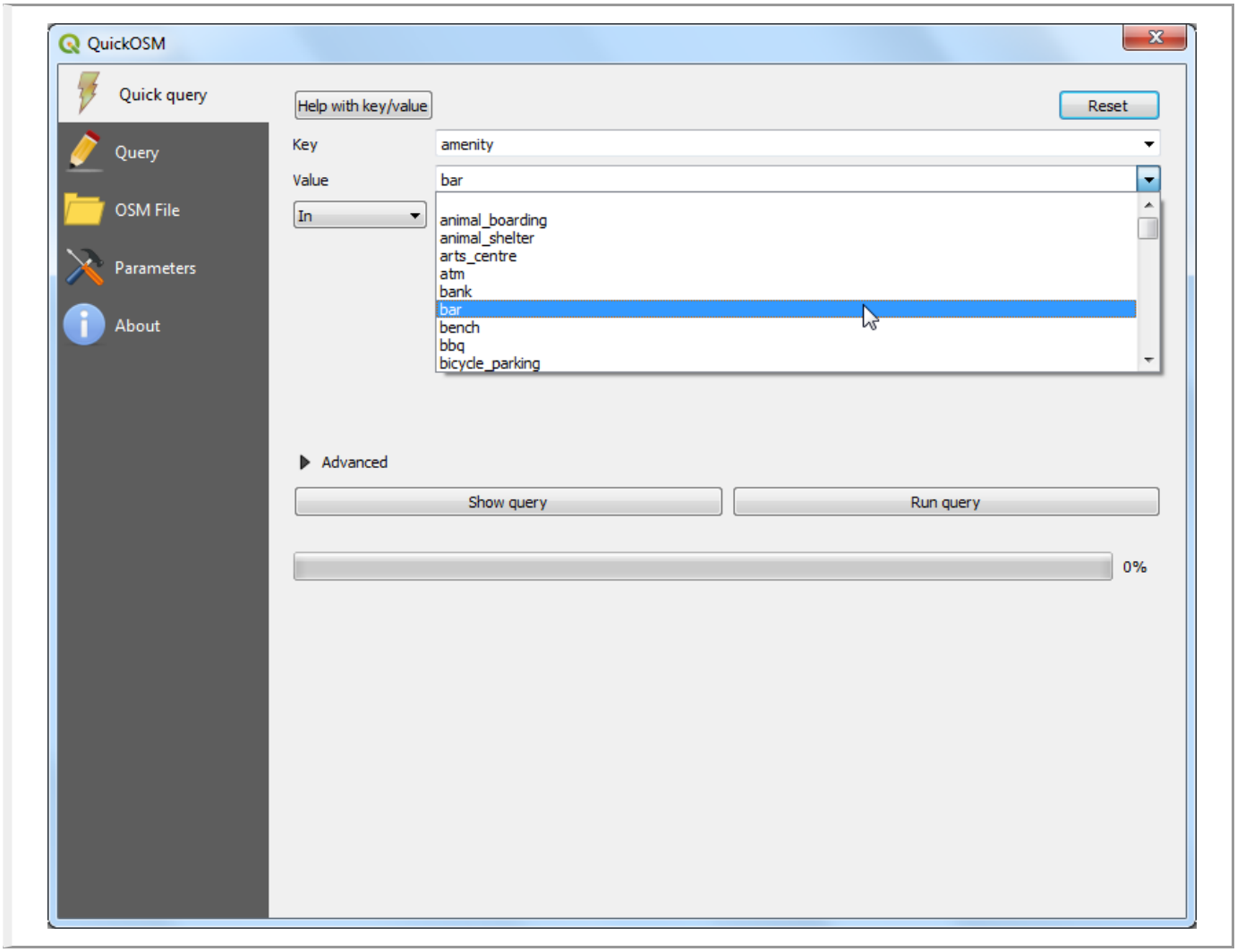
2. Once installed, launch the plugin from Vector ‣ QuickOSM -> QuickOSM....



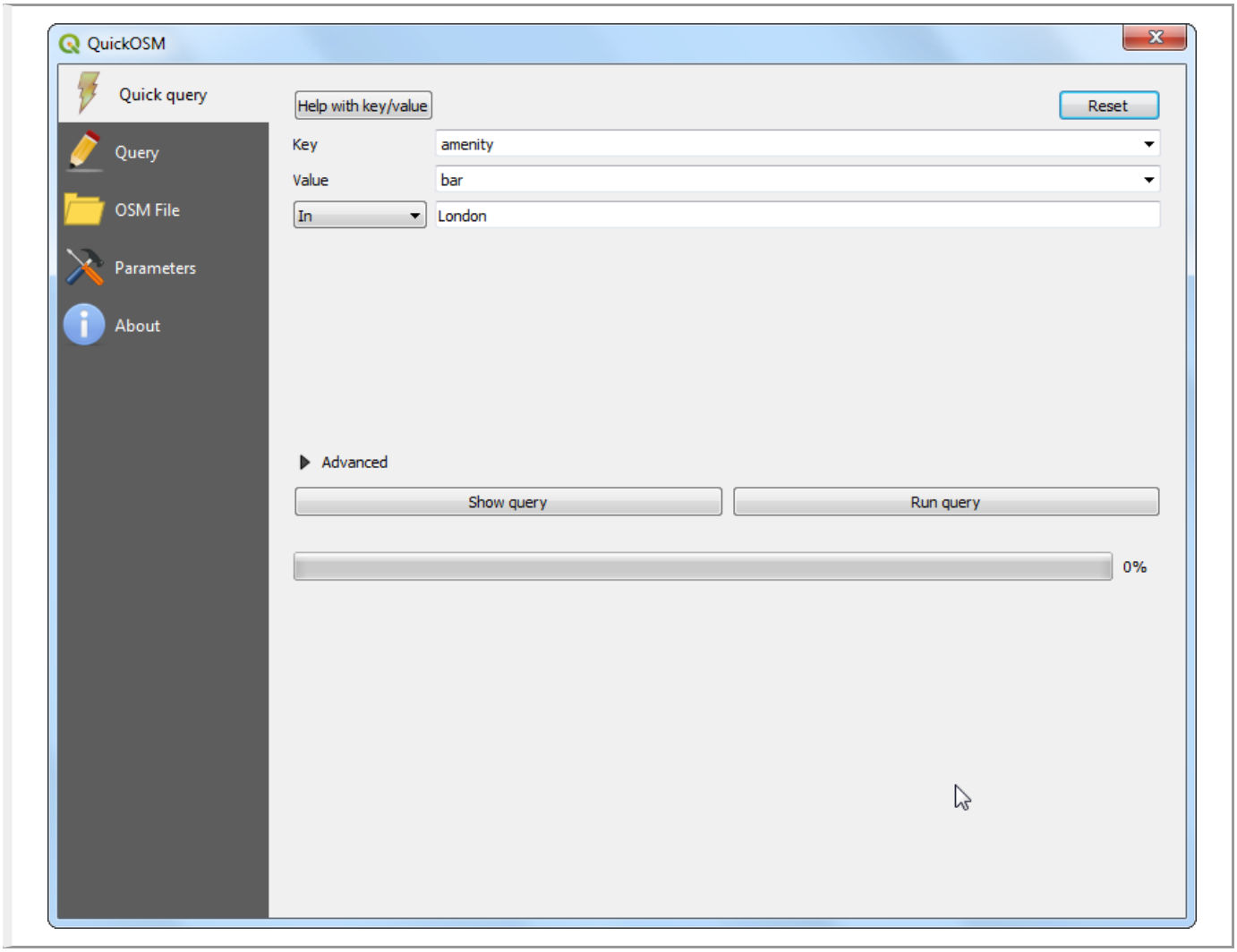
3. In the Quick query tab, you can set a filter to select a subset. The attributes of the map features in the OSM database are stored as Tags (https://wiki.openstreetmap.org/wiki/Tags). Tags are represented with a key and a value. The key is a topic and a value is a specific form. See this page (https://wiki.openstreetmap.org/wiki/Map_Features) for a comprehensive list of tags for various types of features. Bars are represented using the tag `amenity:bar` and pubs with the tag `amenity:pub`. We will first extract the bars. Select `amenity` as the Key from the drop-down menu.



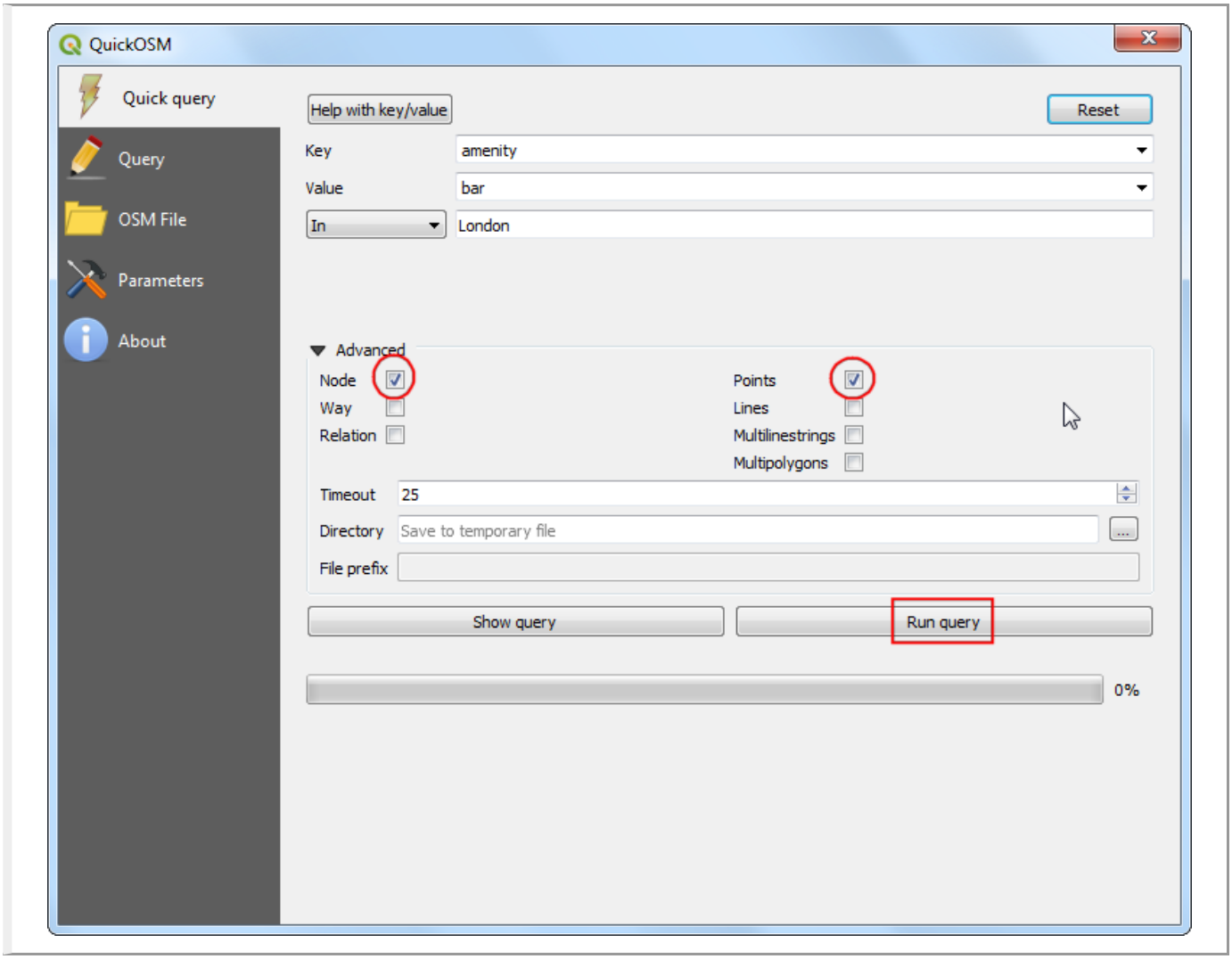
4. Select `bar` from the Value drop-down menu.



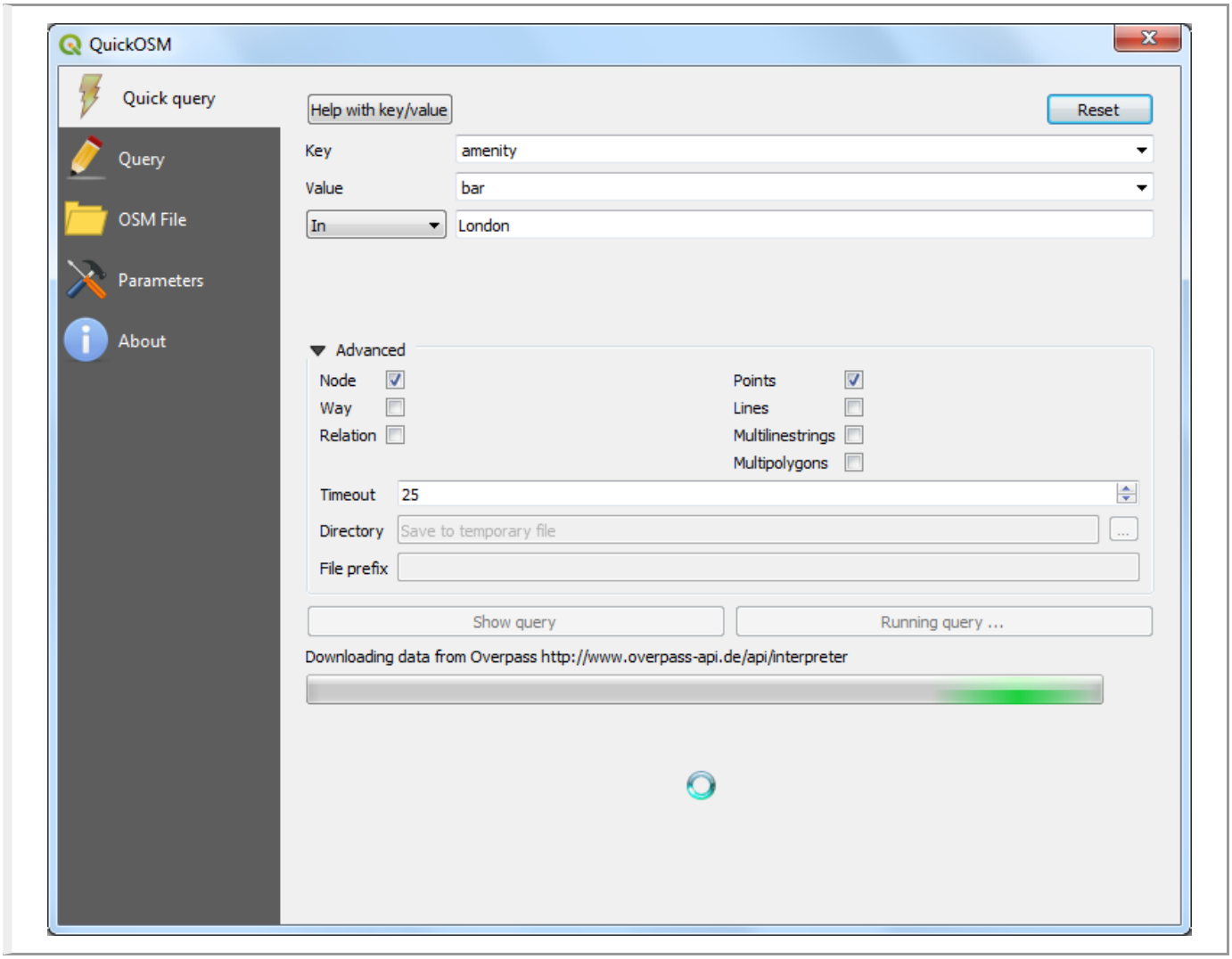
5. Enter London as the In to restrict the search within the city boundary.



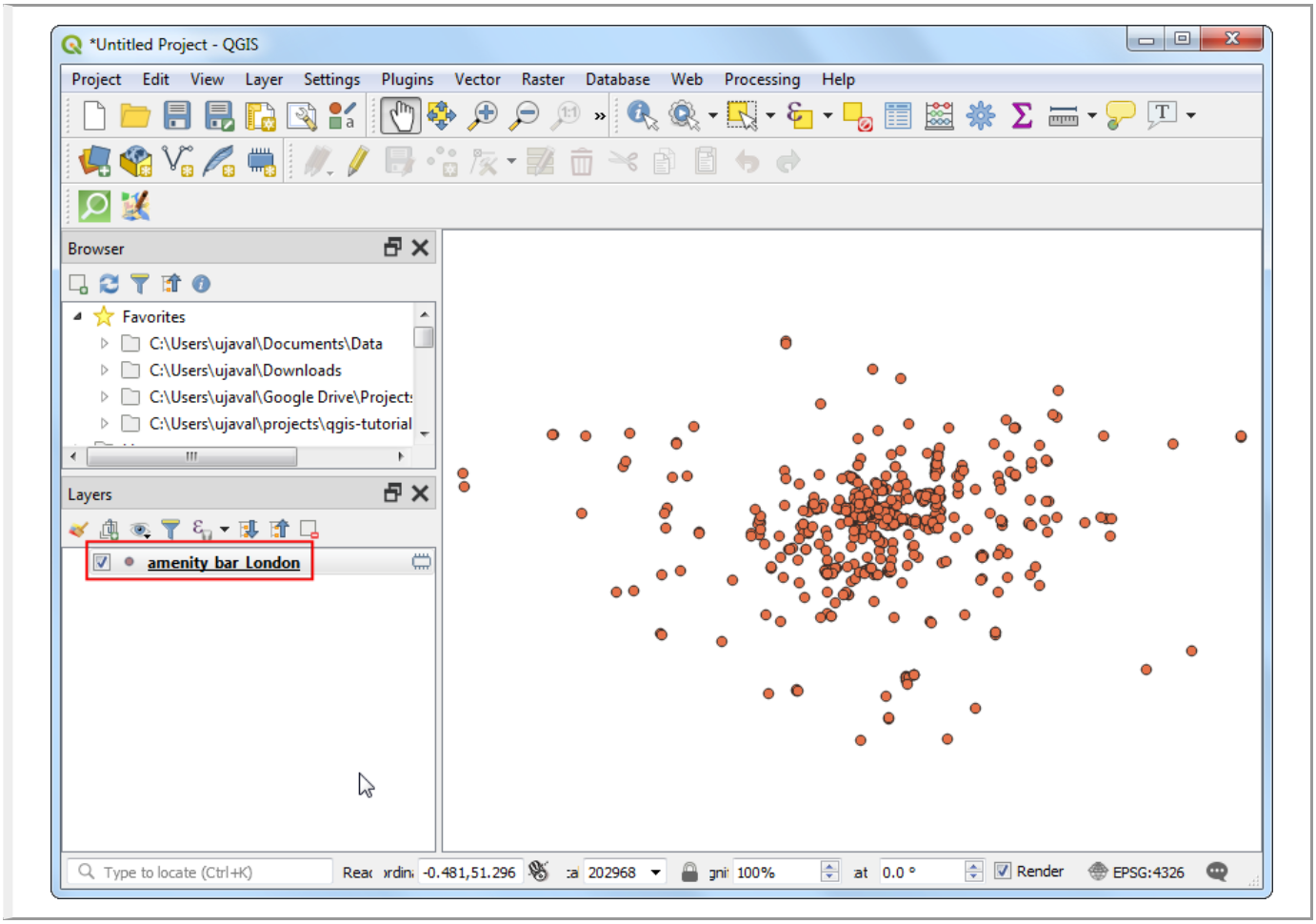
6. Expand the Advanced section. In the OSM data model, features are represented using nodes, ways and relations (https://wiki.openstreetmap.org/wiki/Elements). As we are interested in the point features, you can select only `Node` and `Points` . Click Run query.



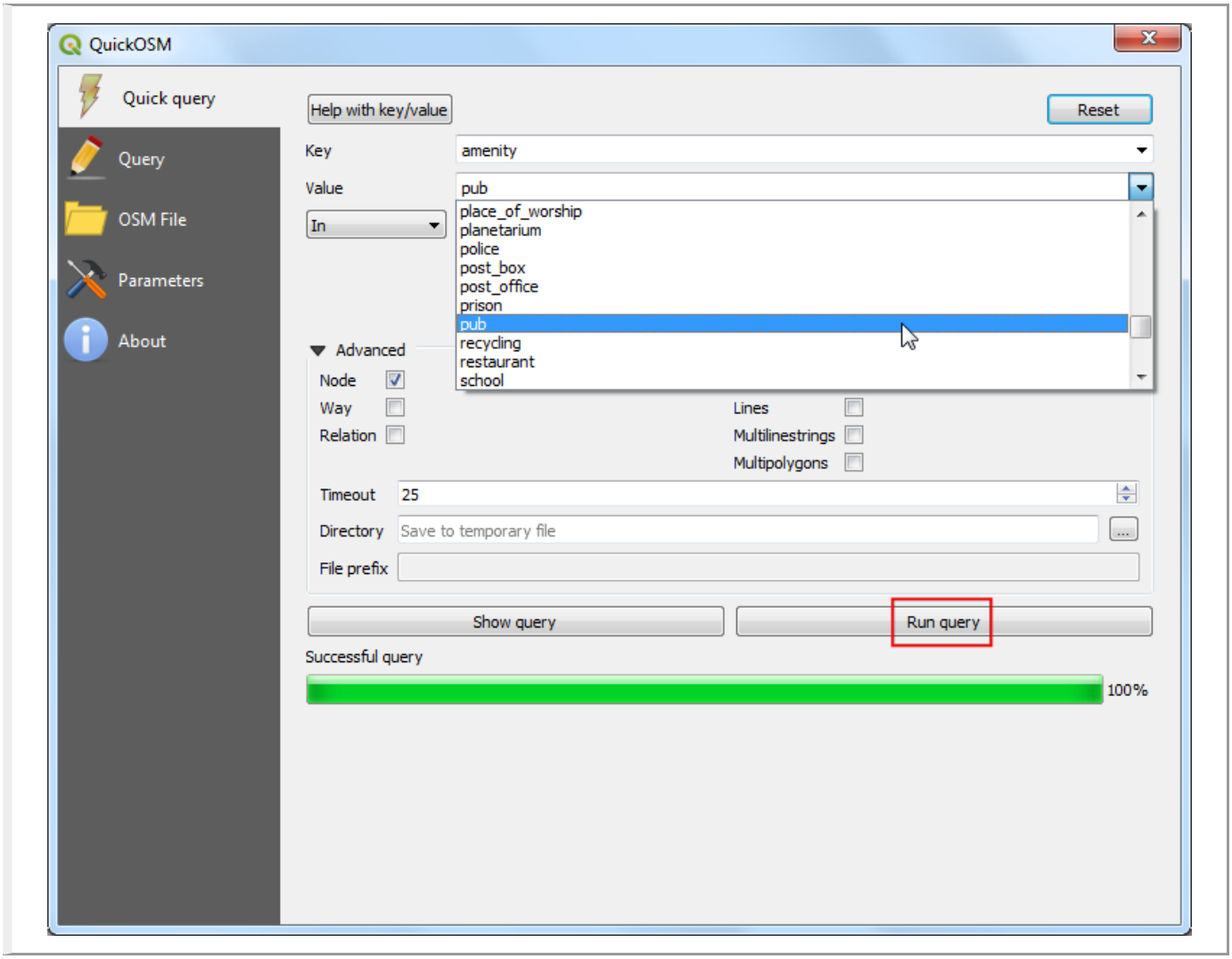
7. The plugin with query the OpenStreetMap database using the Overpass API (https://wiki.openstreetmap.org/wiki/Overpass_API) and convert the data into a QGIS vector layer.



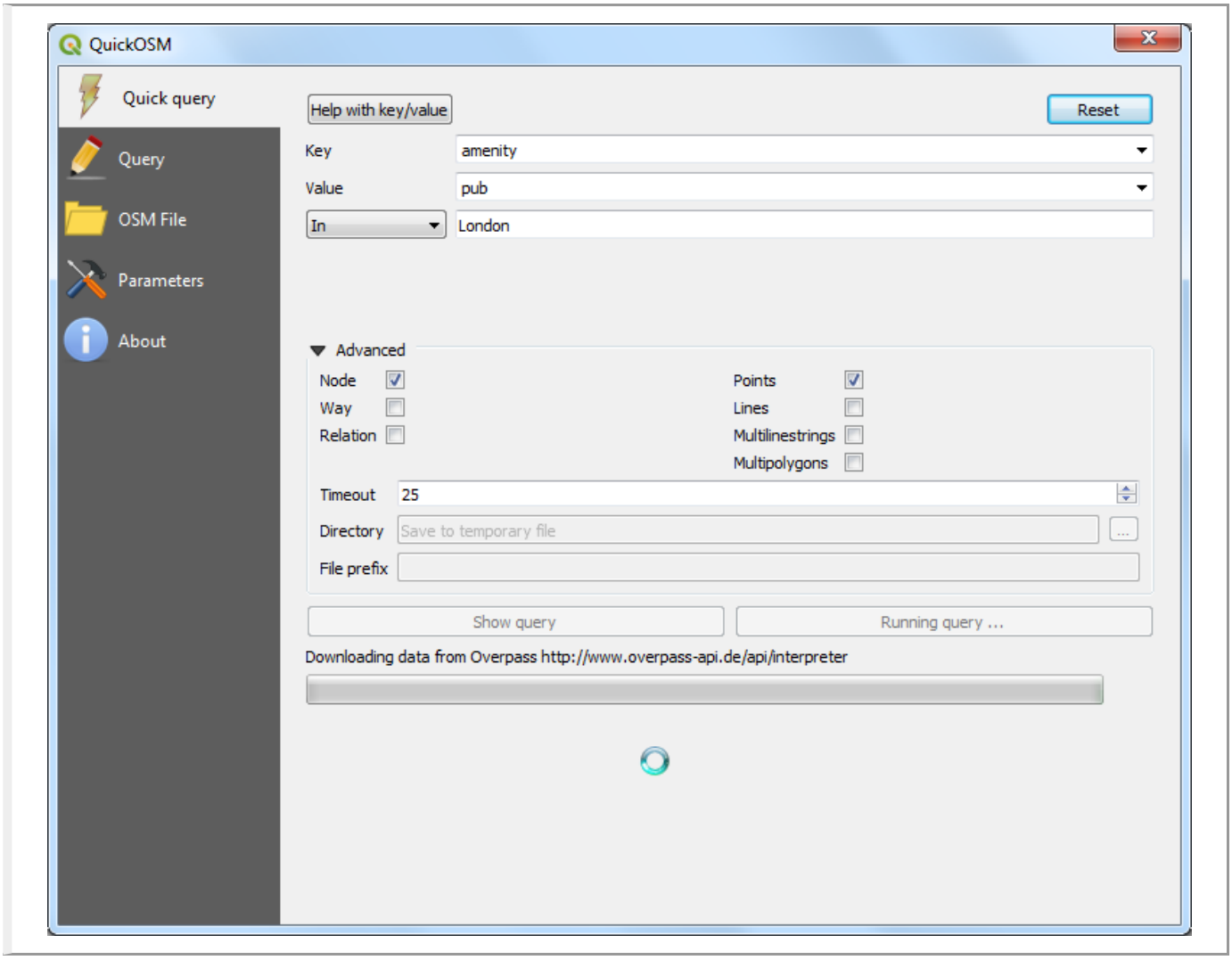
8. Once the query finishes, switch to the main QGIS window. You will see a new layer called `amenity_bar_london` added to the Layers panel. The canvas will show the locations of the bars that were extracted.



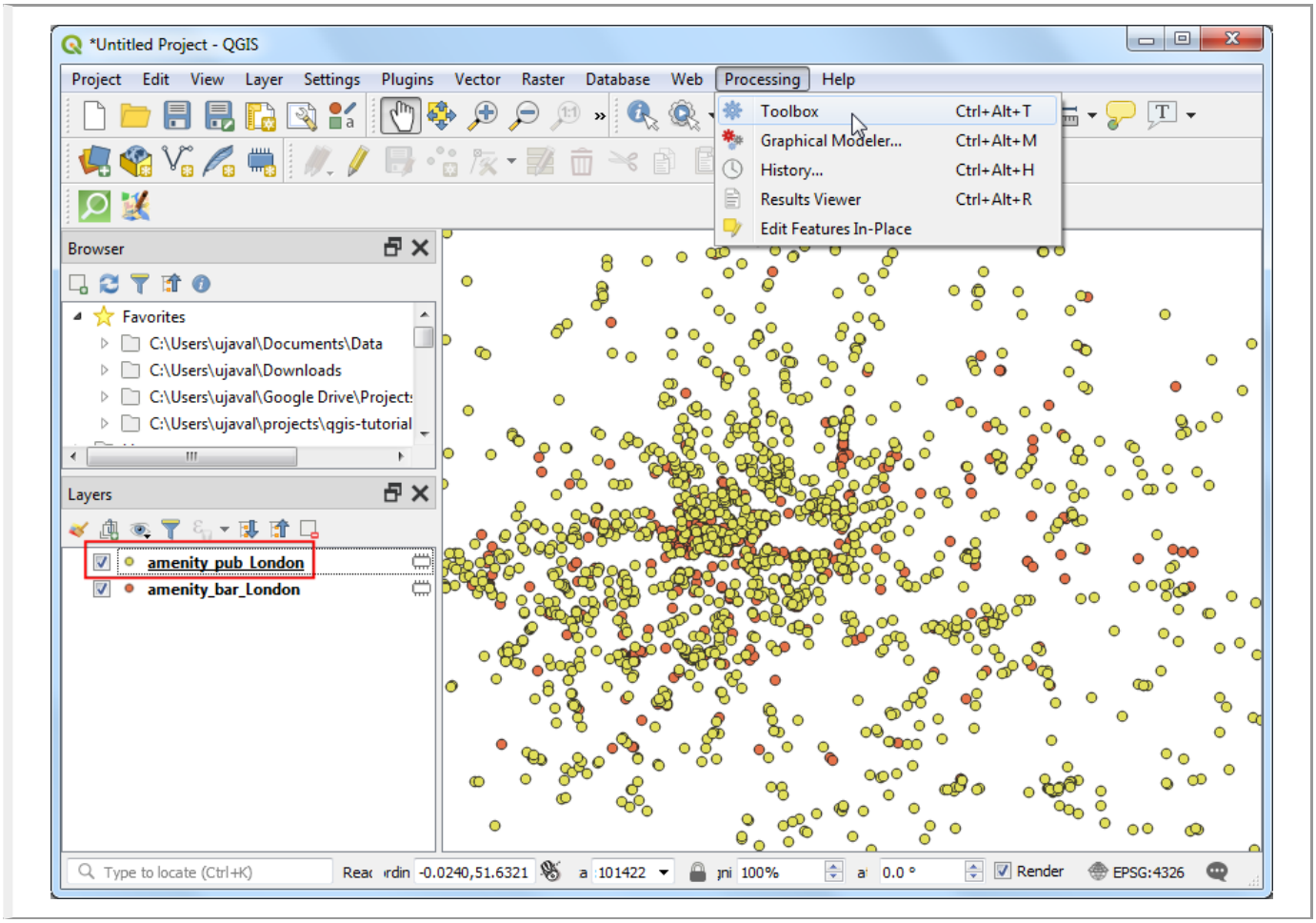
9. Switch back to the QuickOSM window, and edit the query to select `pub` as the Value. Click Run query.



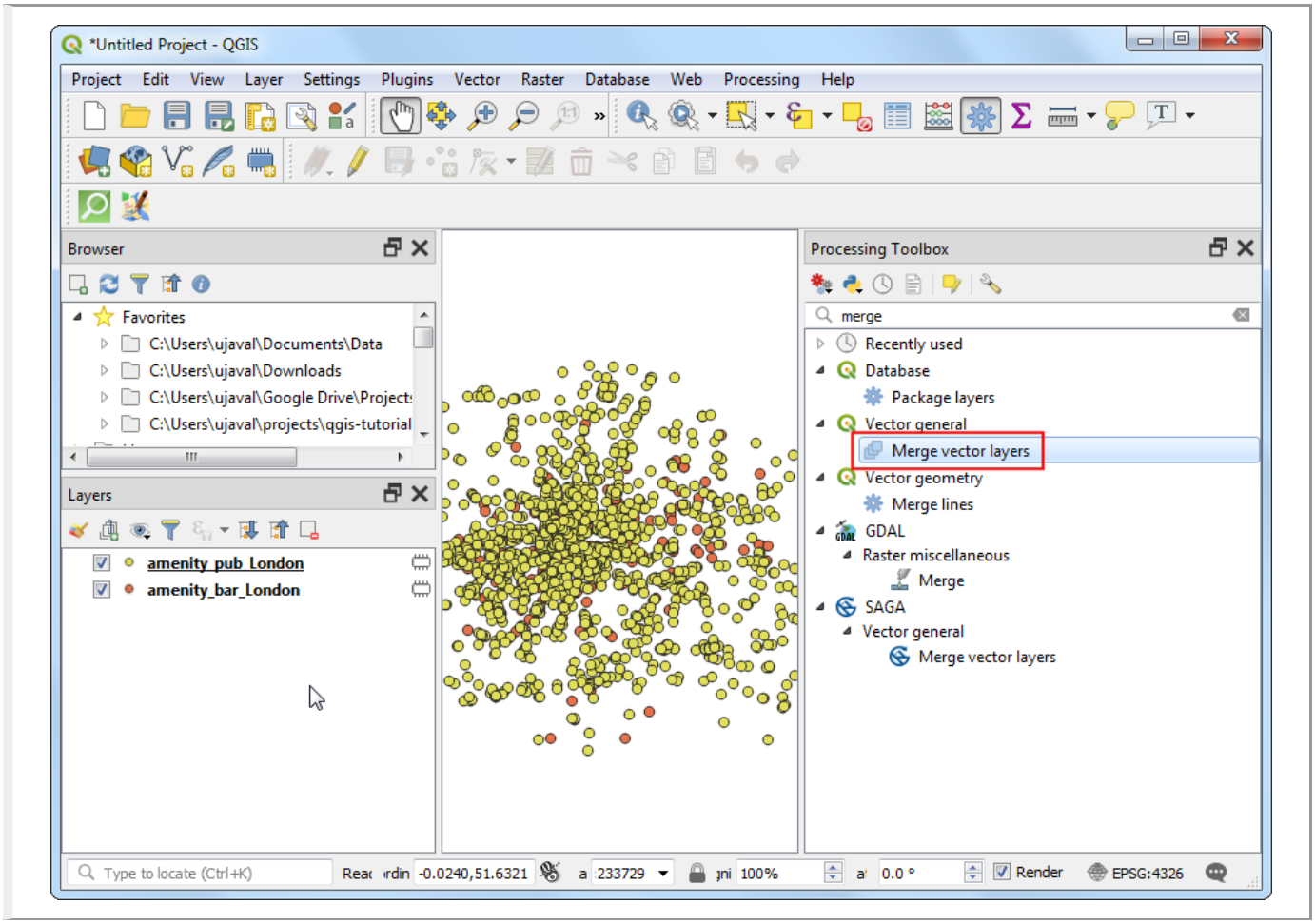
10. This time the plugin will fetch all the points tagged with `amenity:pub` from the OSM database.



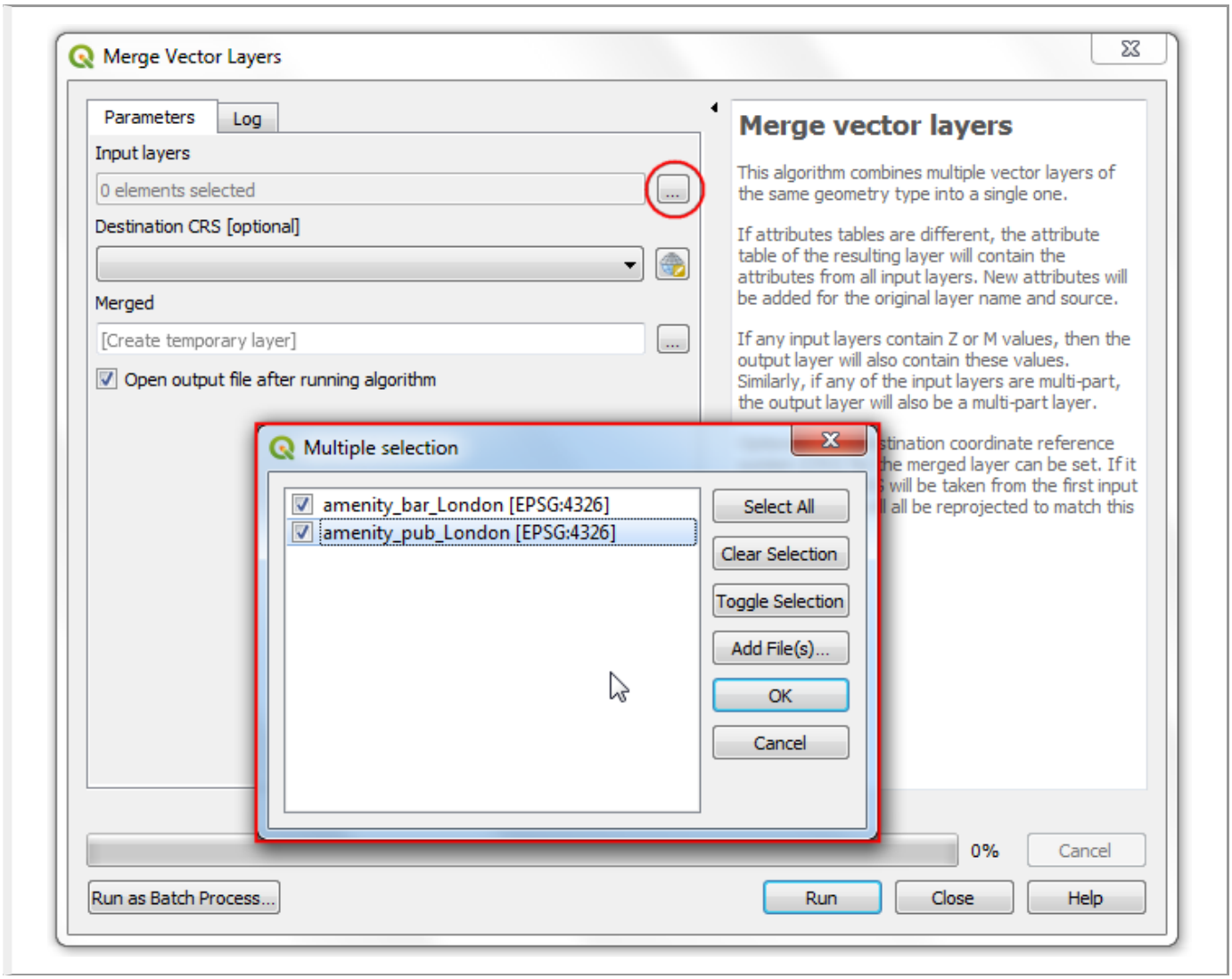
11. Once the query is complete, a new layer `amenity_pub_london` will be added to the Layers panel. We now have 2 vector layers. These are temporary memory layers that will get lost after we exit QGIS. Let's merge these to a single vector layer and save it to the disk. Go to Processing ‣ Toolbox.



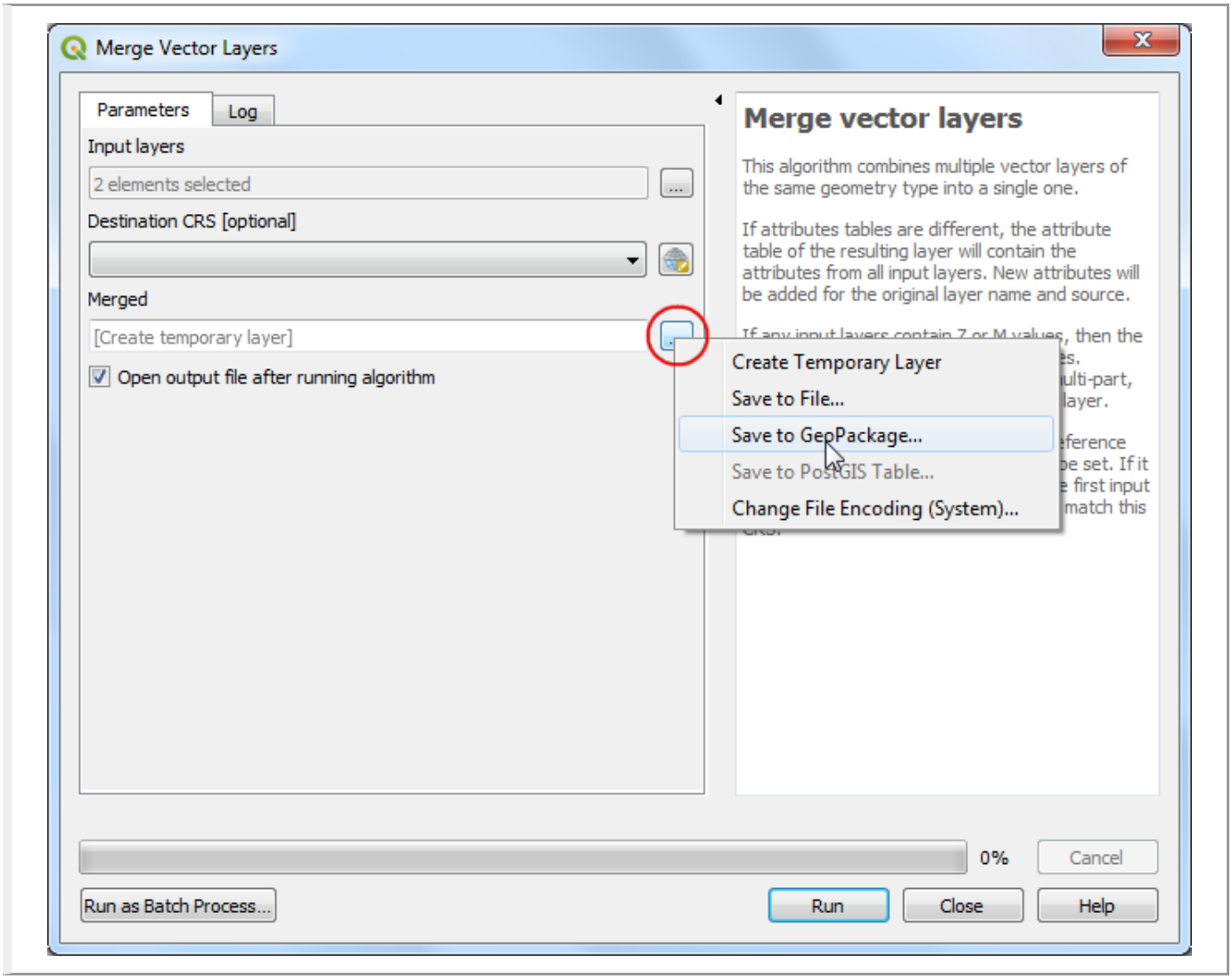
12. Search and locate the Vector general ‣ Merge vector layers tool. Double-click to launch it.



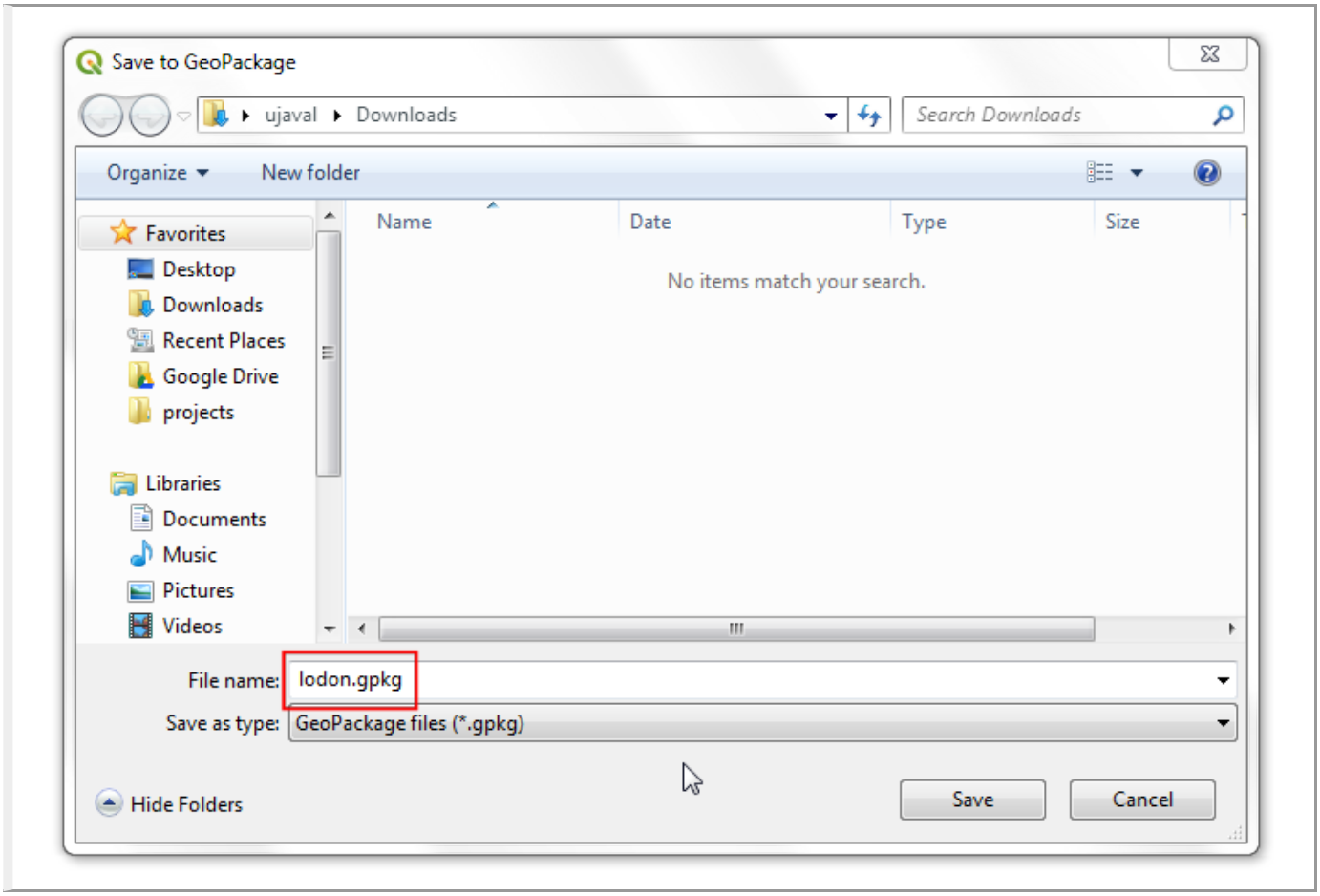
13. In the Merge Vector Layers dialog, click the ... button next to Input layers. Select both the `amenity_bar_london` and `amenity_pub_london` layers. Click OK.



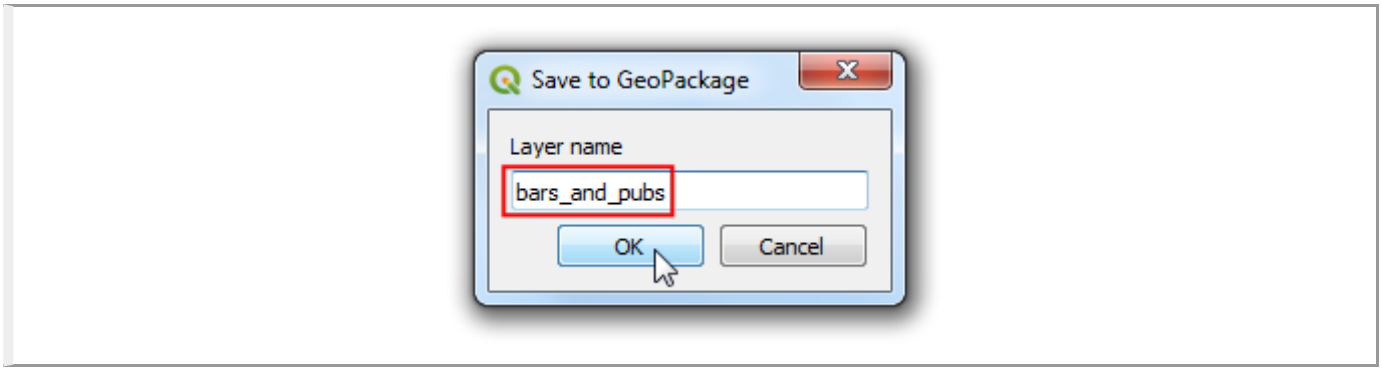
14. Click ... button next to Merged and select Save to GeoPackage.



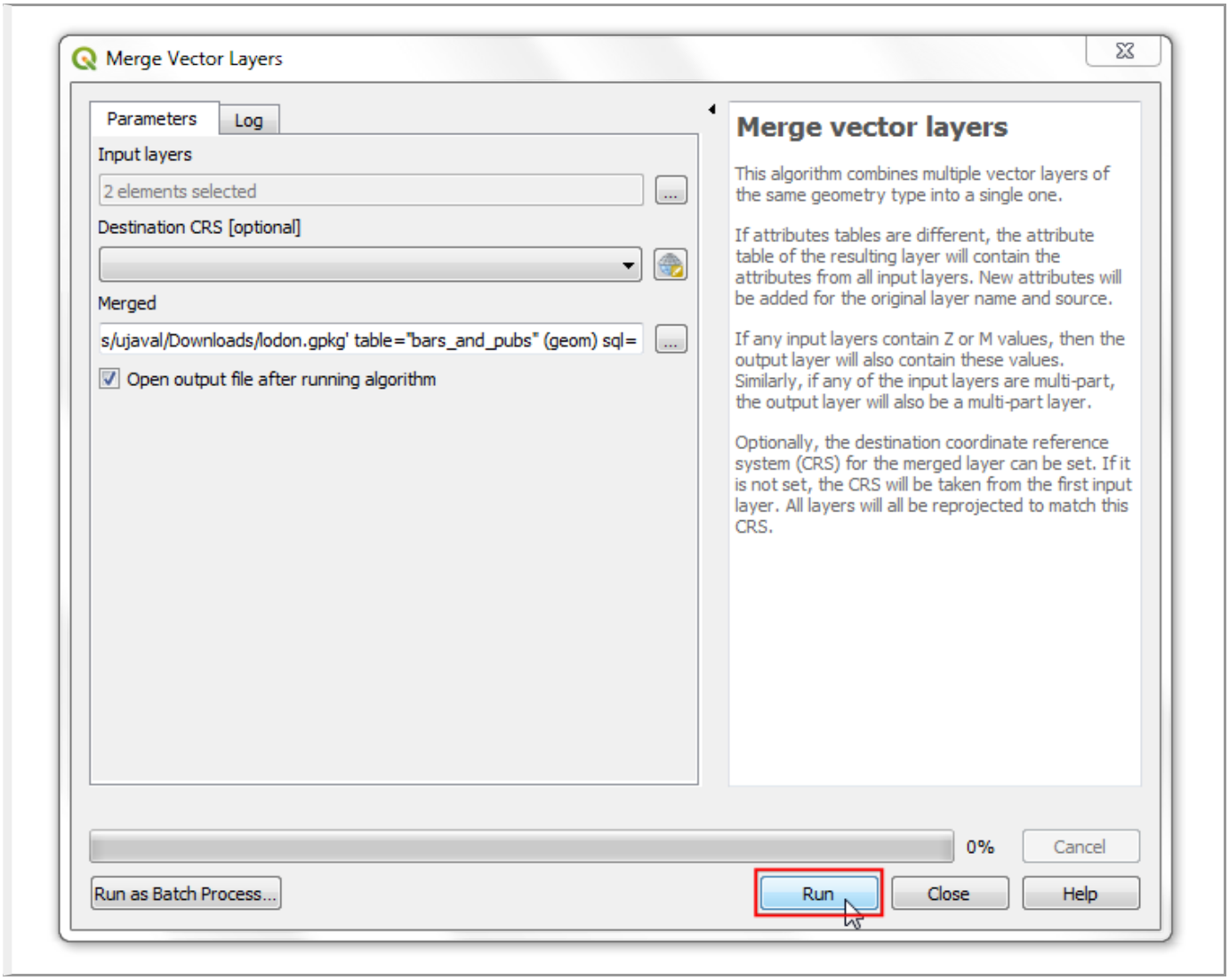
15. Browse to the directory where you want to save the data and name the output `longon.gpkg` .



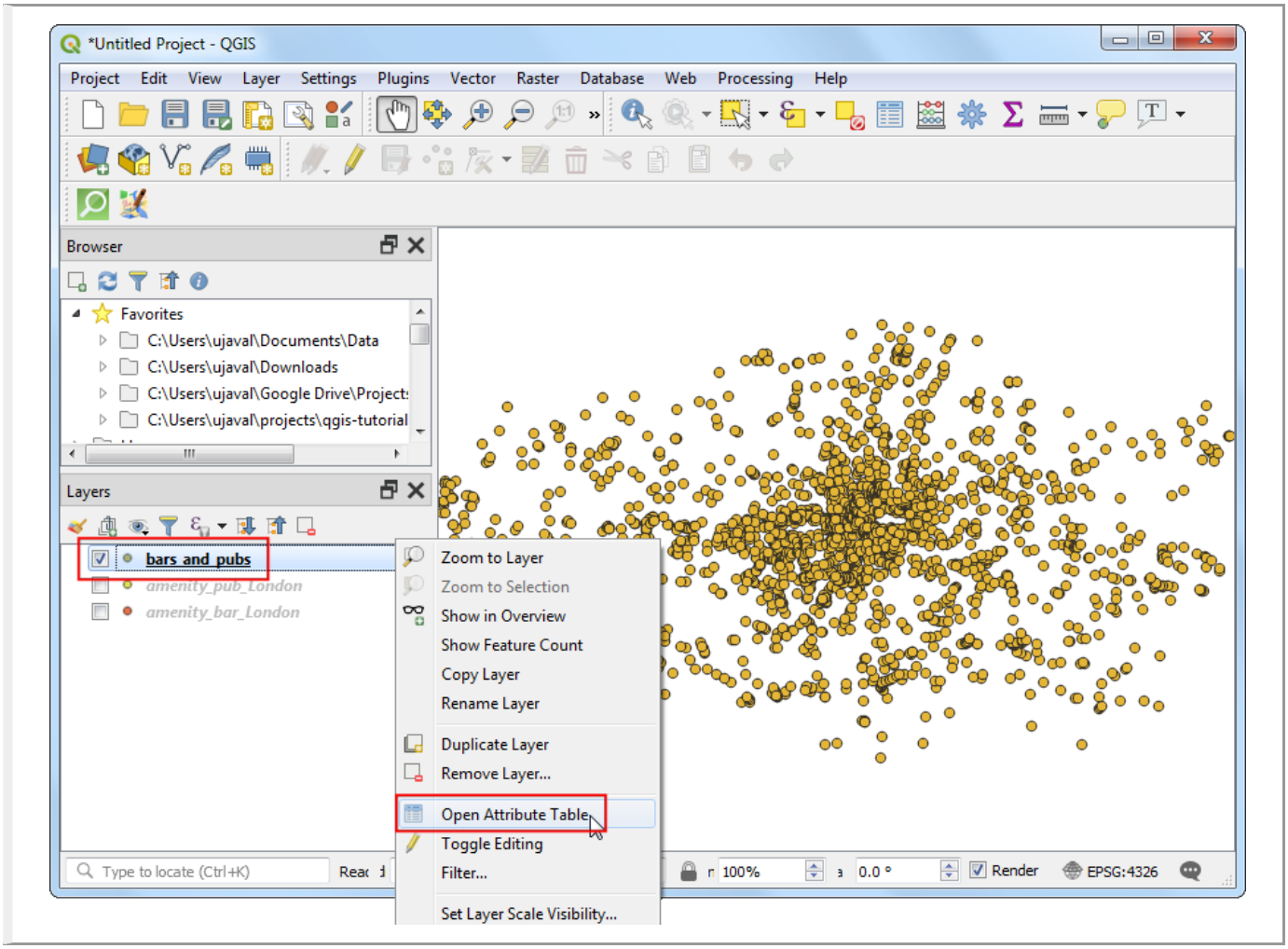
16. Enter `bars_and_pubs` as the Layer name.



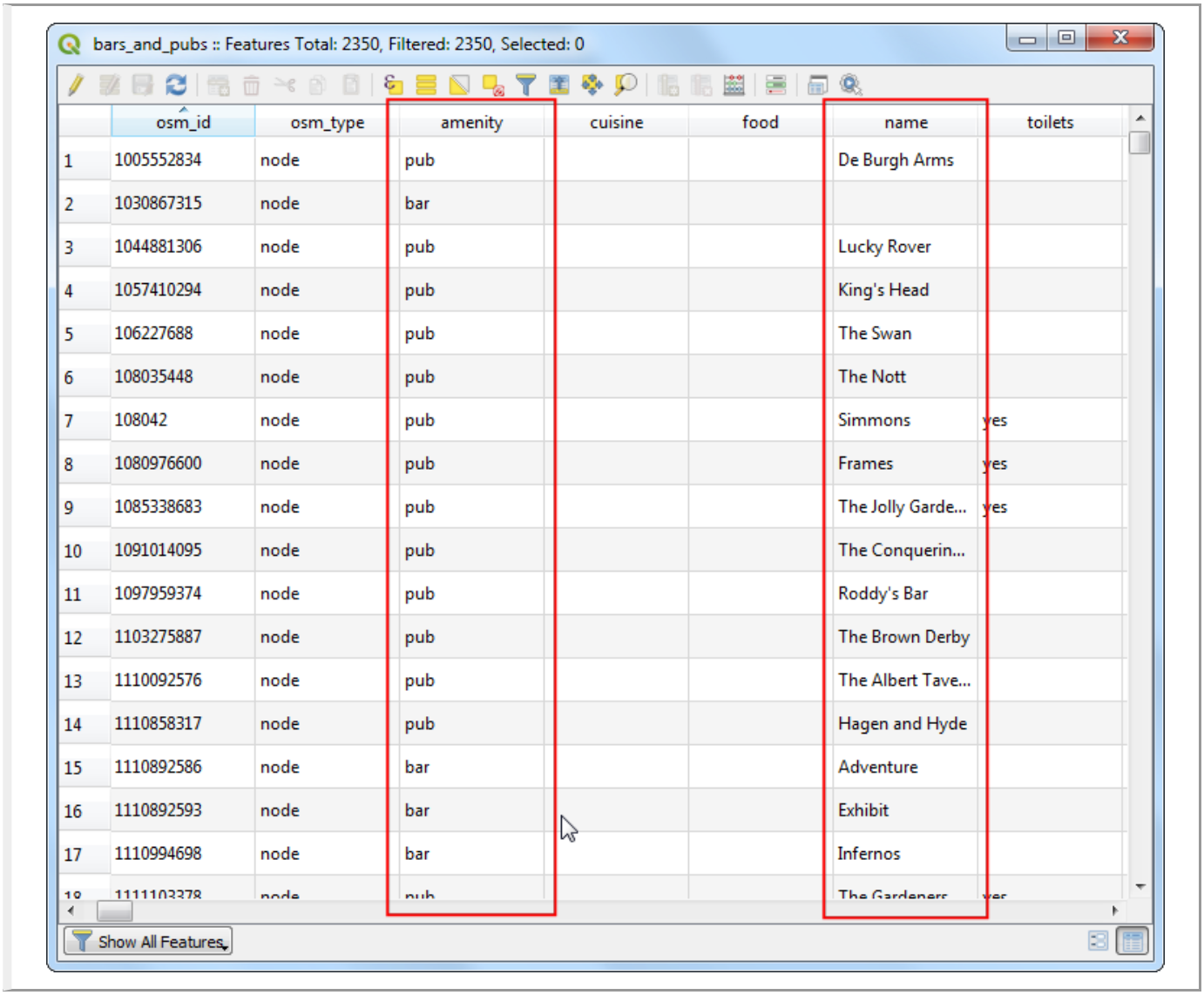
17. Click Run to execute the merge process.



18. Once the processing finishes, you will see a new layer `bars_and_pubs` added in the Layers panel. You will see that this layer is the union of all features from both the previous layers. Right-click the `bars_and_pubs` layer and select Open Attribute Table.

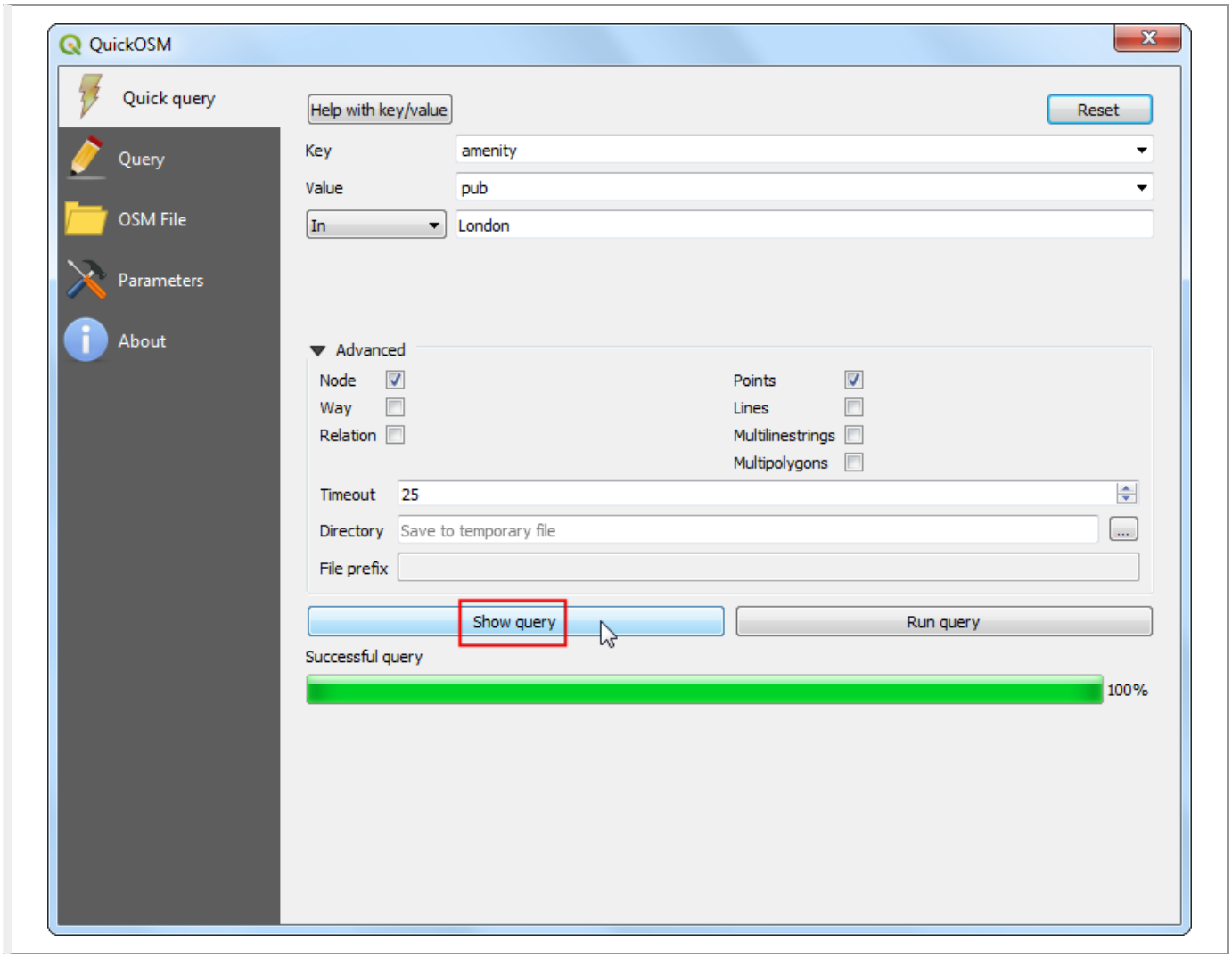


19. In the Attribute Table, you will see that the layer contains both pub and bar amenity types along with the names of these establishments and other attributes.

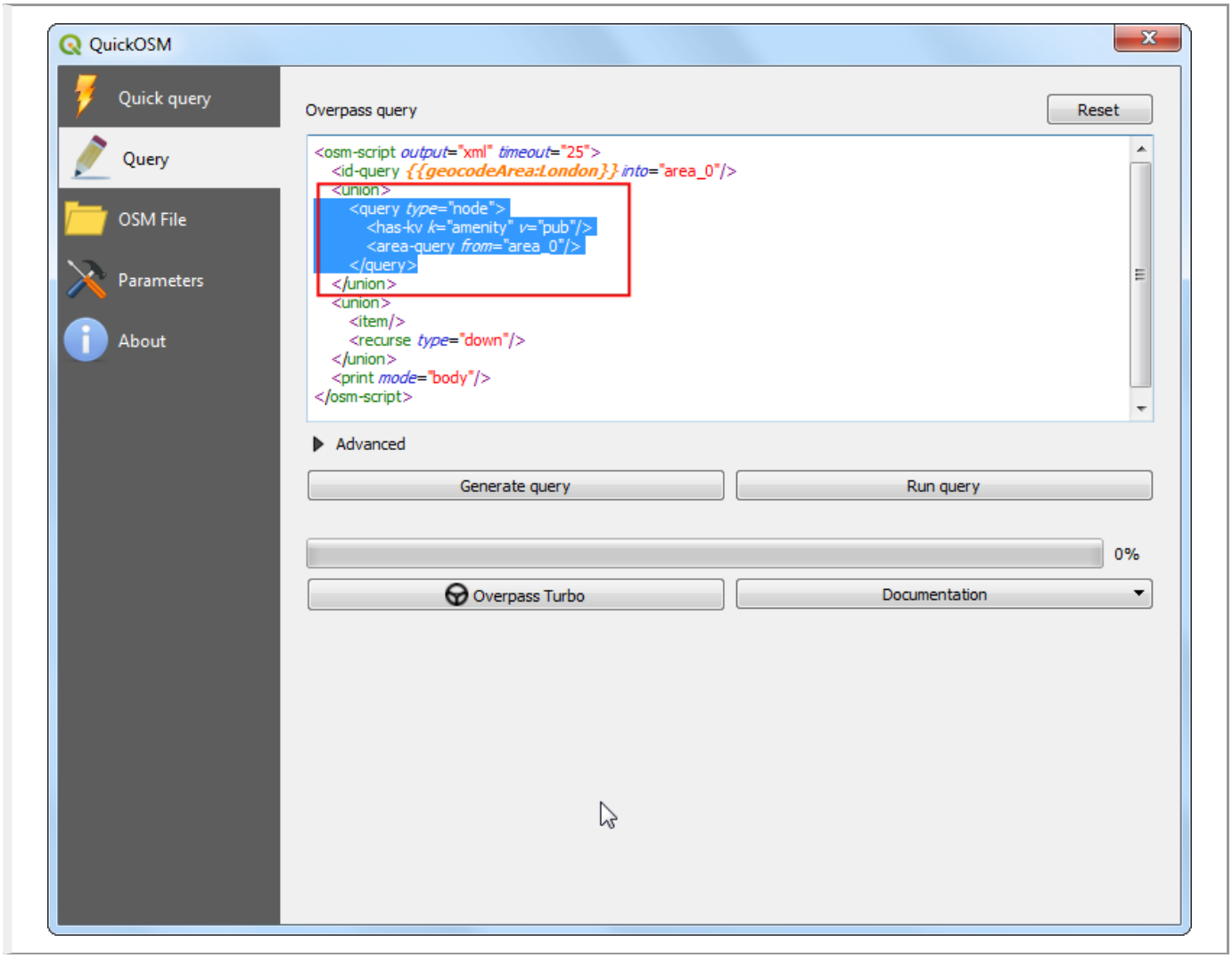


	osm_id	osm_type	amenity	cuisine	food	name	toilets
1	1005552834	node	pub			De Burgh Arms	
2	1030867315	node	bar				
3	1044881306	node	pub			Lucky Rover	
4	1057410294	node	pub			King's Head	
5	106227688	node	pub			The Swan	
6	108035448	node	pub			The Nott	
7	108042	node	pub			Simmons	yes
8	1080976600	node	pub			Frames	yes
9	1085338683	node	pub			The Jolly Garde...	yes
10	1091014095	node	pub			The Conquerin...	
11	1097959374	node	pub			Roddy's Bar	
12	1103275887	node	pub			The Brown Derby	
13	1110092576	node	pub			The Albert Tave...	
14	1110858317	node	pub			Hagen and Hyde	
15	1110892586	node	bar			Adventure	
16	1110892593	node	bar			Exhibit	
17	1110994698	node	bar			Infernos	

20. We have achieved the objective of extracting the bars and pub locations in London. We had to perform 2 separate queries to get the relevant data and merge it. This is fine for our task, but you maybe in a situation where you need to perform a complex query to get the right set of data for your project. Fortunately, the QuickOSM plugin provides a way to write and execute custom queries. Let's see how we can write a singel query for the task at hand. Switch to the QuickOSM window and click Show query.

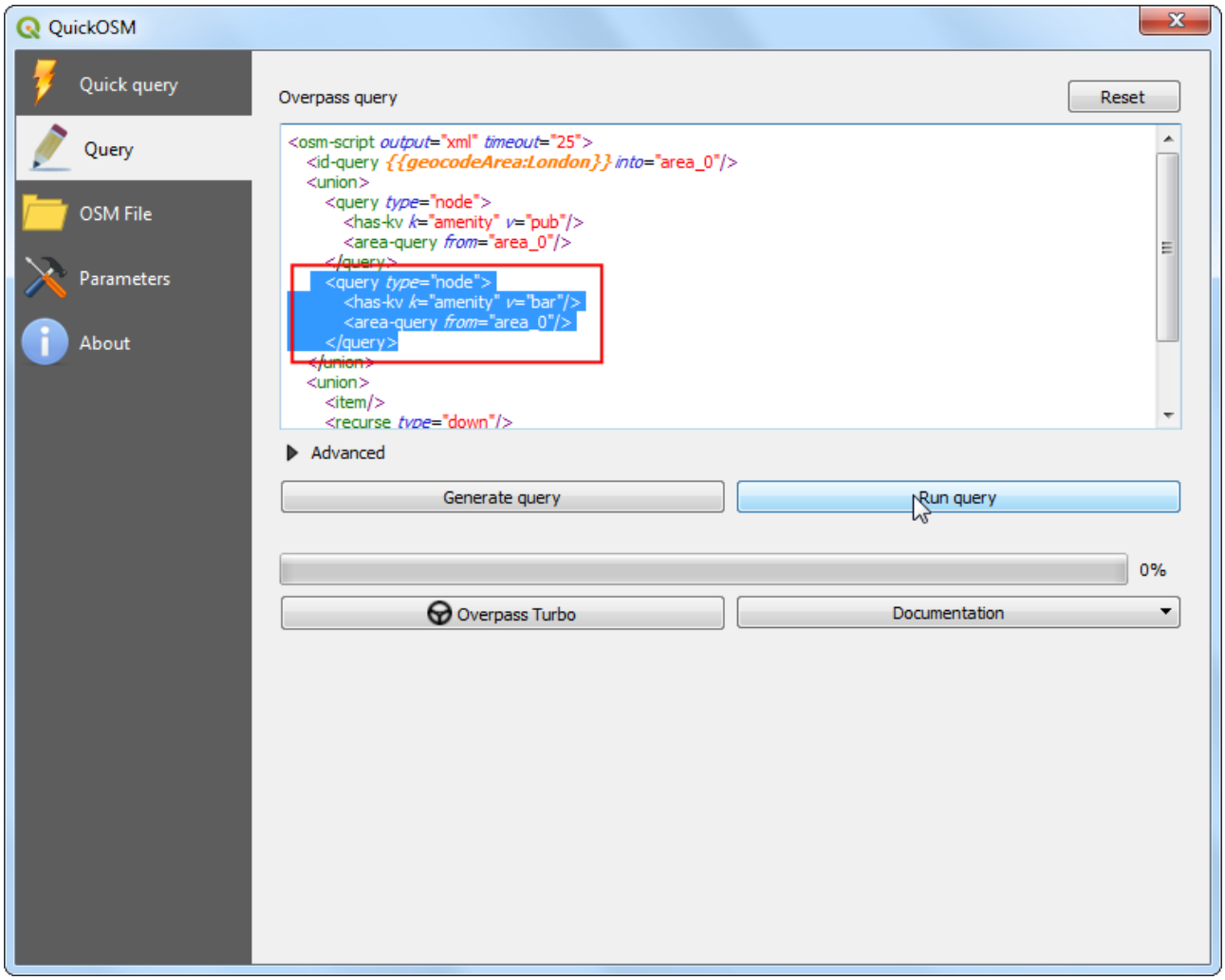


21. The plugin will switch to the Query tab. The Overpass query section will show the query that was constructed based on the user input. This field is editable and one can enter any query. The format of the query is in the Overpass Query Language (QL) (https://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide). For our purpose, select the section between the <query> ... </query> XML tags and copy it.

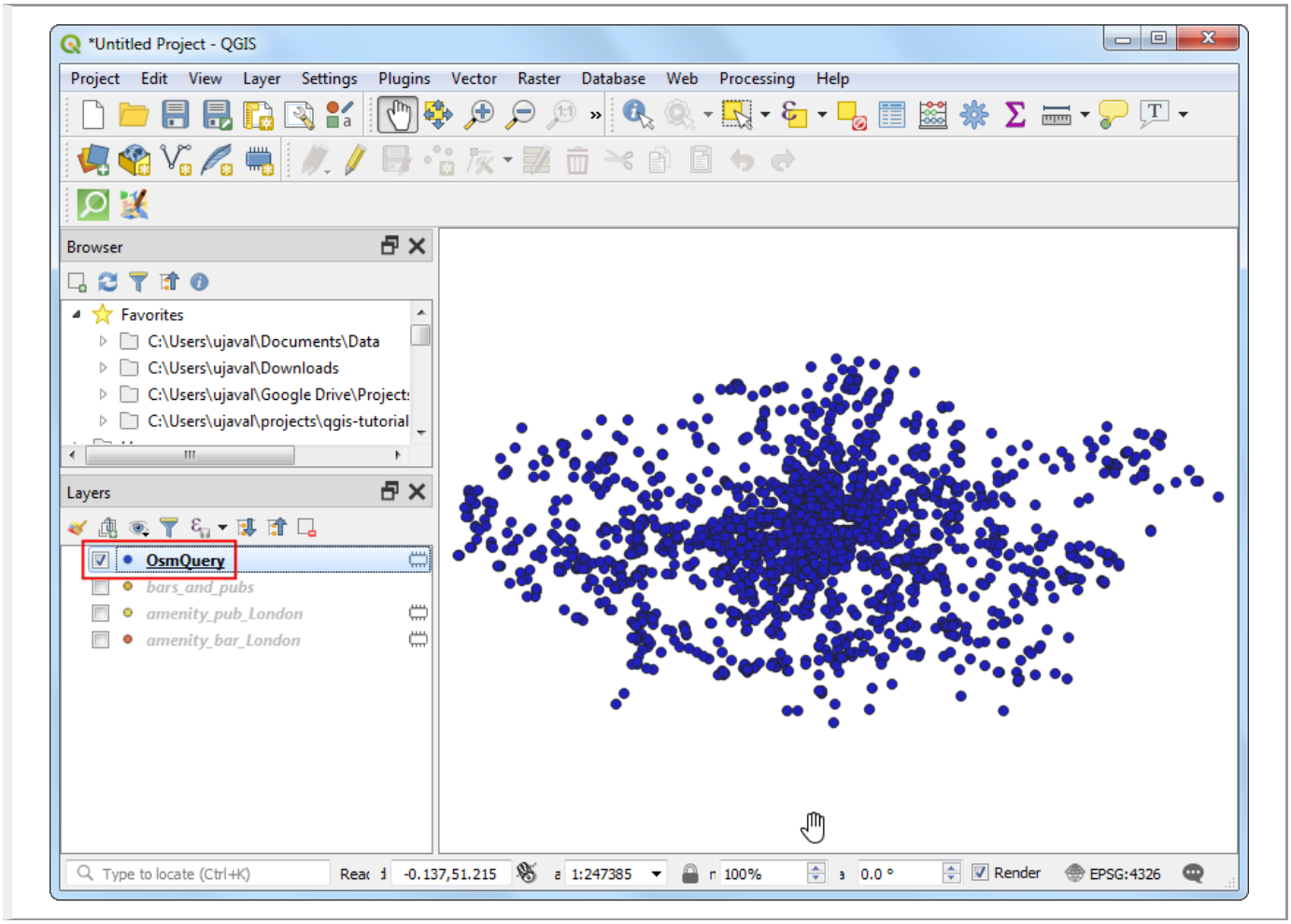


22. Paste it after the existing query section and change the value from `pub` to `bar`. Below is the full query that will fetch values from both the tags in a single query. Click Run query.

```
<osm-script output="xml" timeout="25">
<id-query {{geocodeArea:London}} into="area_0"/>
<union>
    <query type="node">
        <has-kv k="amenity" v="pub"/>
        <area-query from="area_0"/>
    </query>
    <query type="node">
        <has-kv k="amenity" v="bar"/>
        <area-query from="area_0"/>
    </query>
</union>
<union>
    <item/>
    <recurse type="down"/>
</union>
<print mode="body"/>
</osm-script>
```



23. Once the query finishes, you will see a new layer `OsmQuery` added to the Layers panel. This layer contains points representing both bars and pubs in London.



Back to top
Last updated on Nov 09, 2019.
Created using Sphinx (http://sphinx.pocoo.org/) 1.3.6.