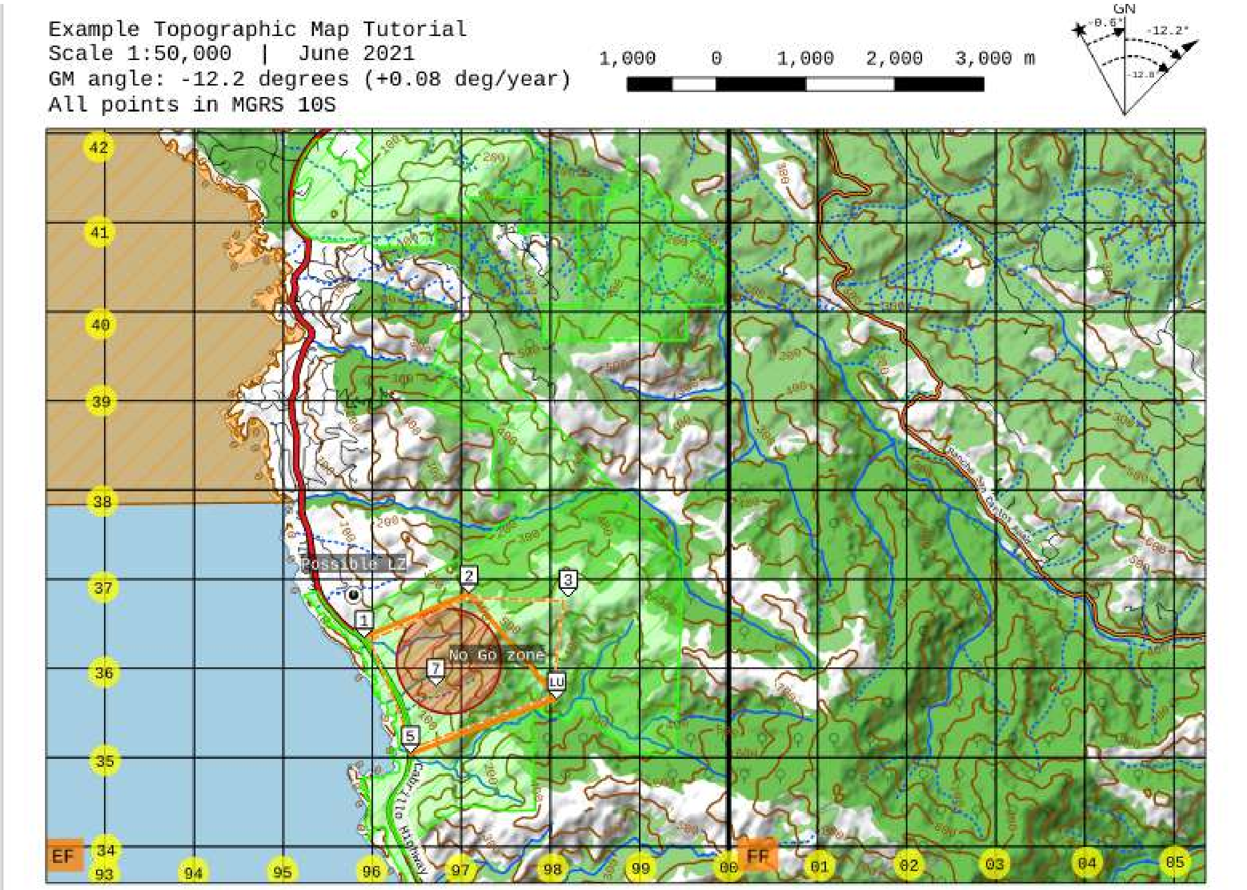
ATAK is great. Google Maps is great(ish, privacy concerns not withstanding). Being able to read the stars and find a general idea of where to go is great. None of those, however, are a complete and reliable replacement for a proper paper map, compass, and protractor when it comes to professional, all-weather use in finding your way from point A to point B, or locating features around you. The explosion of open-source software and data has given us the tools we need to create said professional-quality maps on our own, instead of relying on a government or private entity to create them for us with the points we actually care about marked, and in a fashion usable for navigation.

In this guide, I'm going to lay out, step-by-step, how to use free and open-source software and datasets to create navigation-quality maps usable with the Military Grid Reference System (MGRS), allowing you to seamlessly use GPS devices to confirm position, use standard protractors to plot points, and use a compass in conjunction with the map to navigate. We'll start from scratch and end up with a military-style topographic map similar in appearance to the following:



I'll be making a few assumptions throughout this guide:

- I'm assuming that you, the reader, have a basic level of technological literacy and can pick up on how to access menus in a program; generally speaking, once I introduce how to get somewhere in the software in this guide, I won't do it again and skip straight to what's new in each subsequent step.
- Similarly, I assume you've got a basic ability to poke around with the software and tweak things on a basic level if the settings I'm using don't quite fit your needs.
- I'm assuming you have a basic familiarity with land navigation principles, so I won't go into too much depth on *why* things like contour lines and grids are useful or essential.
- I'm assuming that you're using a relatively recent version of the software I'm using. The same principles will work on other platforms, and with other software, but I don't use those, so apologies if things are different.

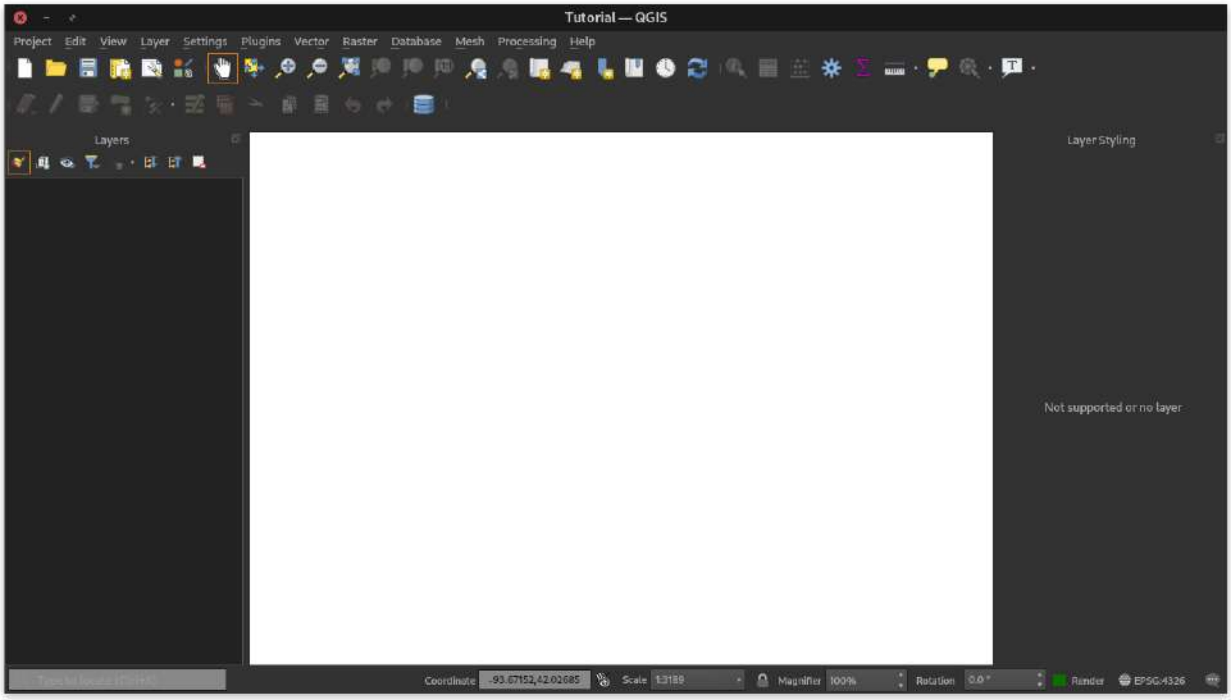
Step 0: The necessary tools

My off-duty GIS of choice is QGIS; like most of the software I use for reasons that compose an extended rant in and of themselves^[the tl;dr is that, like physical equipment, I prefer to use stuff that can be maintained by the user and isn't beholden to some monolithic entity, whether government or business in nature.], it's free and open-source. It can be downloaded from the QGIS site for Linux, Windows, or Mac; depending on what operating system you're using, you may be able to download from a package manager or app store as well. I personally use the latest builds, but you may want to use the LTR (long-term release) version, which is a little older but is guaranteed to work for much longer, and has a few more plugins available if you really want to get into the weeds on mapmaking.

For the purposes of this map, I'll be using scenic Monterey, CA as my example, and using QGIS 3.18.3 "Zurich" as my QGIS version.

Step 1: Creating the base map

Like any other program, go to ‘`Project > New`, or hit `Ctrl+N`. You should be immediately greeted by a blank white screen, like the below:


Tutorial — QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Mesh Processing Help
Layers
Layer Styling
Not supported or no layer

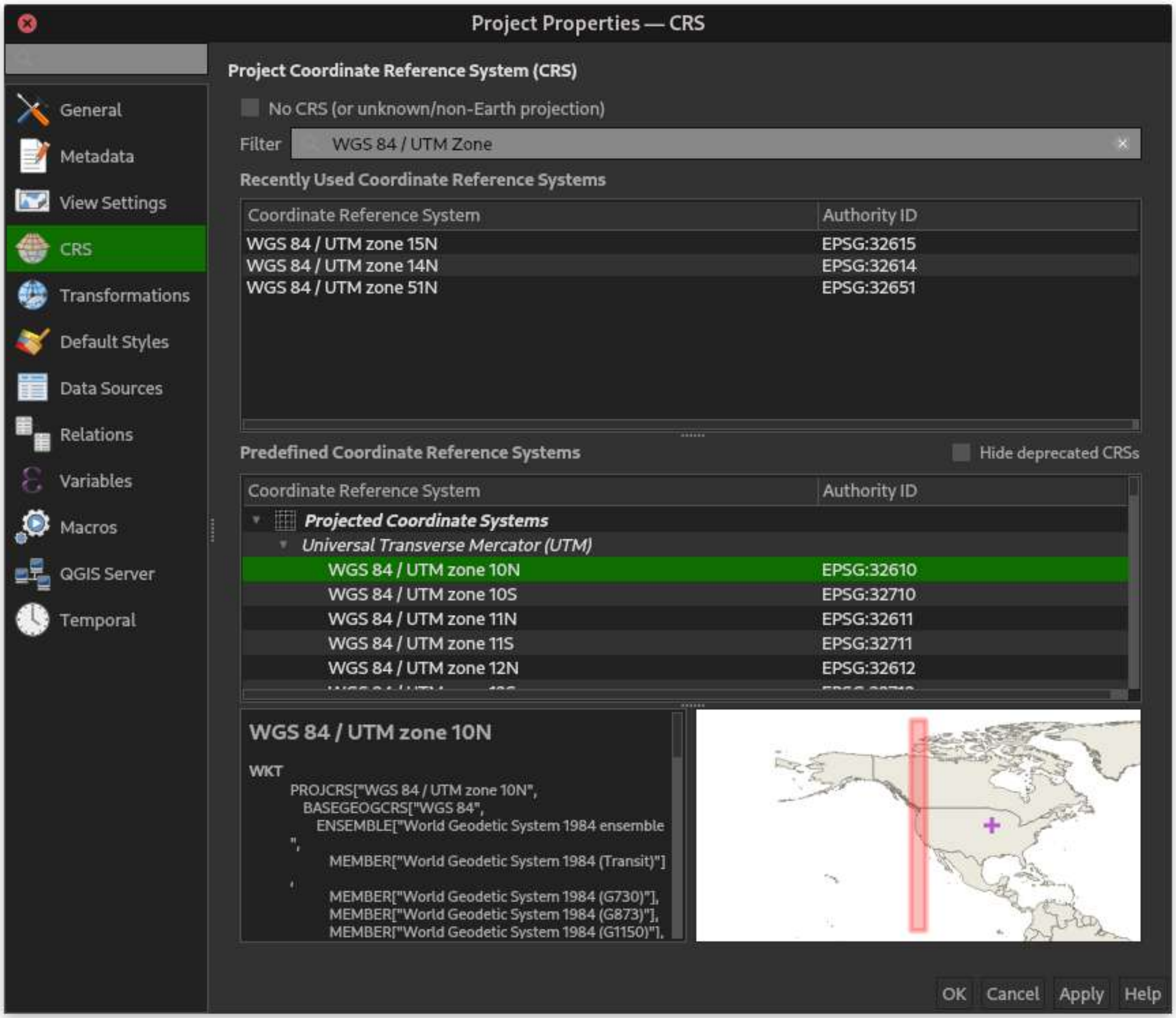
Coordinate -93.67152,42.02685 Scale 1:3189 Magnifier 100% Rotation 0.0 ° Render EPSG:4326

This is your canvas - however, like everything else in the project, it doesn't matter without a way for the program to convert arbitrary values ("Town at -1, -1 km from the center") into a point on the Earth's surface. You need to set a coordinate reference system (CRS) in order to allow this, and messing up the CRS can seriously mess up your whole project. Go to `Project > Properties`; if it doesn't take you there directly, select CRS from the left side.

For the purposes of military maps, we want to be able to use a map and protractor and ensure we have correct military grid reference system (MGRS) coordinates on anything we plot. Long story short, the MGRS system uses the WGS-84/UTM datum (baseline) for everything, so we need to pick an appropriate UTM zone for our CRS.

For maps covering more than one MGRS zone, or if you're at all unsure, I recommend using the `EPSG:4326 - WGS 84` CRS.

Type in "WGS-84 / UTM zone" in the search bar; it'll filter it down to UTM zones, and you can see on the map to the right which area the zone covers. Keep scrolling through until you find the right one; those familiar with MGRS will note that the UTM zone lines up almost, but not exactly, with the first couple characters of a grid coordinate. MGRS zones are further subdivided vertically; for our purposes, just understand that if you've got a GPS, the first digits in your full coordinates should line up with the UTM zone.



Project Coordinate Reference System (CRS)

☐ No CRS (or unknown/non-Earth projection)

Filter: WGS 84 / UTM Zone

Recently Used Coordinate Reference Systems

Coordinate Reference System	Authority ID
WGS 84 / UTM zone 15N	EPSG:32615
WGS 84 / UTM zone 14N	EPSG:32614
WGS 84 / UTM zone 51N	EPSG:32651

Predefined Coordinate Reference Systems

☐ Hide deprecated CRSs

Coordinate Reference System	Authority ID
Projected Coordinate Systems	
Universal Transverse Mercator (UTM)	
WGS 84 / UTM zone 10N	EPSG:32610
WGS 84 / UTM zone 10S	EPSG:32710
WGS 84 / UTM zone 11N	EPSG:32611
WGS 84 / UTM zone 11S	EPSG:32711
WGS 84 / UTM zone 12N	EPSG:32612

WGS 84 / UTM zone 10N

WKT

```
    PROJCRS["WGS 84 / UTM zone 10N",
      BASEGEOGCRS["WGS 84",
        ENSEMBLE["World Geodetic System 1984 ensemble
    ",
          MEMBER["World Geodetic System 1984 (Transit)"]
    ,
          MEMBER["World Geodetic System 1984 (G730)"],
          MEMBER["World Geodetic System 1984 (G873)"],
          MEMBER["World Geodetic System 1984 (G1150)"],
```

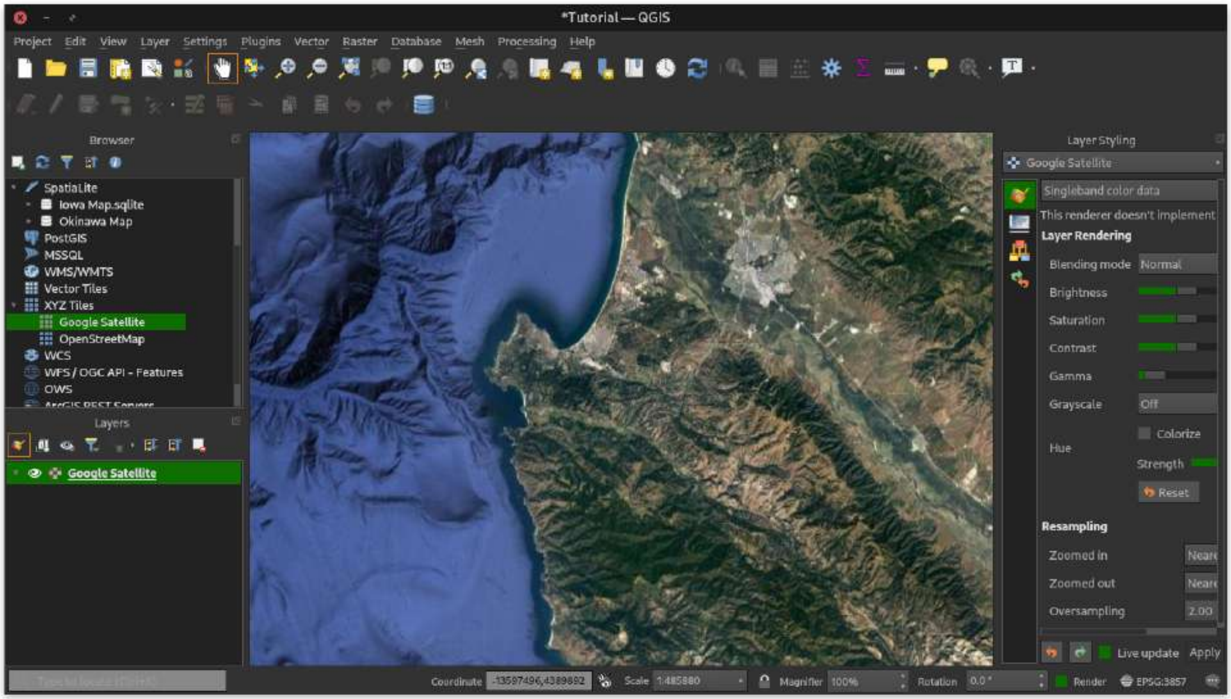
I'll be using the UTM zone 10 North (corresponding); so I'll go ahead and select that and hit OK. Nothing should visibly change; this is exactly what we should expect, since we don't have any data for our map yet.

Time to add some data, now. There are two types of data sets QGIS works with, raster and vector. Raster is simply stuff that has pixels and is defined by pre-existing images; vectors are mathematical equations that are independent of how they're rendered. If you want something high-resolution or precise, use vectors; if you want something incorporating high-density photographic imagery, use rasters.

A good baseline for any map is a satellite view, even if you're not going to use a satellite view in the final topographic map.

Make sure you have your layers open; go to `View > Panels > Layers` if you don't. Go ahead and open up the Browser panel as well, using `View > Panels > Browser`. This'll let you see the various sources of imagery you can pull from by default. (I'd also go ahead and save your project in its own folder now, if you haven't already.)

In the Browser panel, scroll down until you see XYZ tiles, then expand that; you should have the option to use Google satellite view, or OpenStreetMap (think open-source Google Maps.) For now, we'll be using Google Satellite. Right click on it and hit Add Layer to Project. It'll pop up in your layers. By default, this will use its own CRS, not the project CRS we set earlier, but that's fine for now. Go ahead and zoom in on the are you're interested in. The end result should be something like what you see below.



Blending mode Normal

Grayscale Off

☐ Colorize

Zoomed in Neare

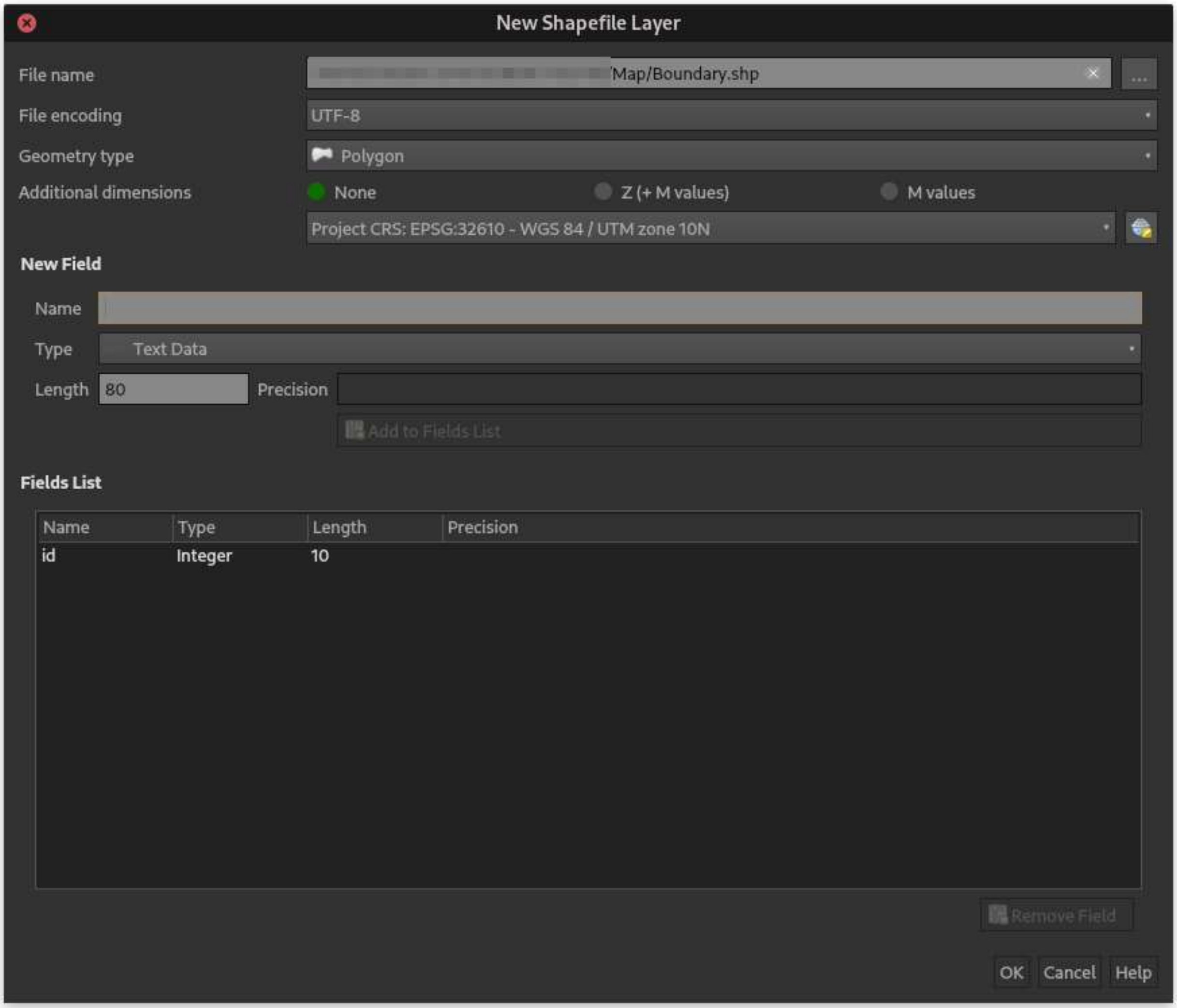
Zoomed out Neare

Oversampling 2.00

Coordinate -13597496,4389892 Scale 1:485880 Magnifier 100% Rotation 0.0 ° EPSG:3857

Let's also go ahead and define the boundaries of where we're interested in. We'll create a new vector layer for it; go to `Layer > Create Layer`. There are several different formats with their own advantage, but by default I recommend using a Shapefile, which means that each individual set of vector imagery will be in its own file. I personally prefer using a Spatialite database for speed's sake, but how to use that is outside the scope of this guide.

Make sure you pick a file location within the folder you saved your project, and name it something sensible. Also make sure you pick "polygon" as the geometry type, since we'll be drawing a big rectangle around the area of interest, and select the same CRS we picked for the project earlier. In the CRS dropdown, you should see "Project CRS: ..." listed with the project CRS, conveniently enough. The overall dialog before you click OK should look similar to the following:



New Shapefile Layer

File name: [redacted]Map/Boundary.shp

File encoding: UTF-8

Geometry type: Polygon

Additional dimensions: ● None ○ Z (+ M values) ○ M values

Project CRS: EPSG:32610 - WGS 84 / UTM zone 10N

New Field

Name:

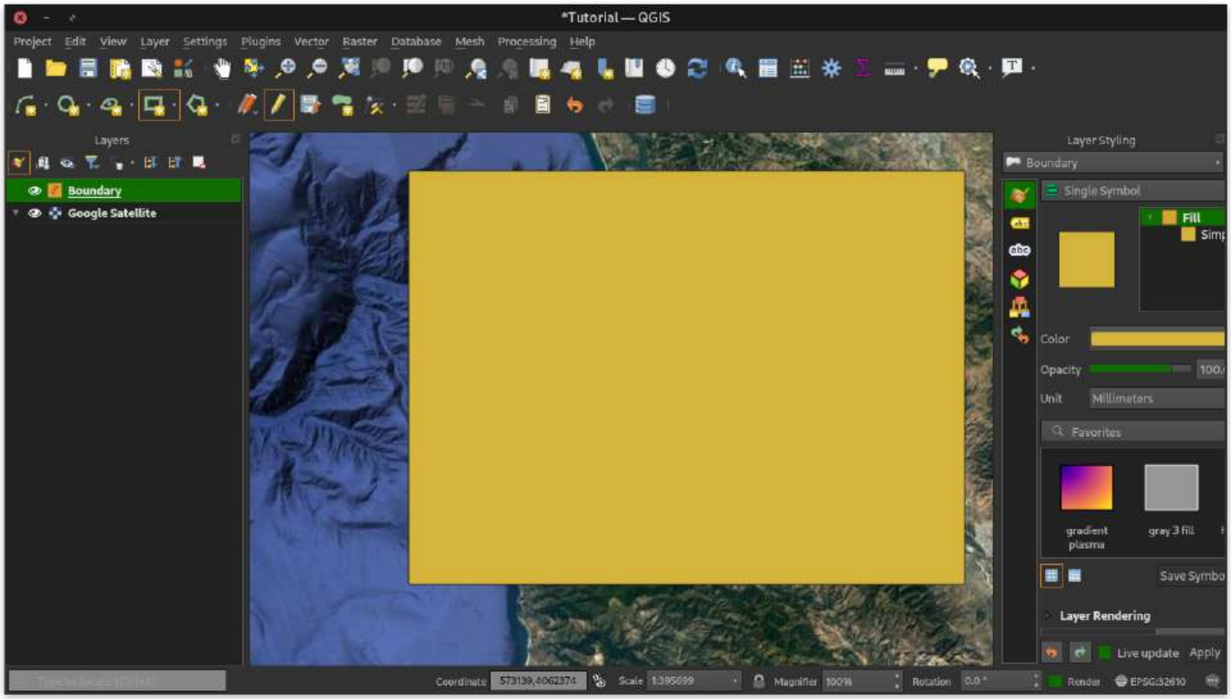
Type: Text Data

Length: 80 Precision:

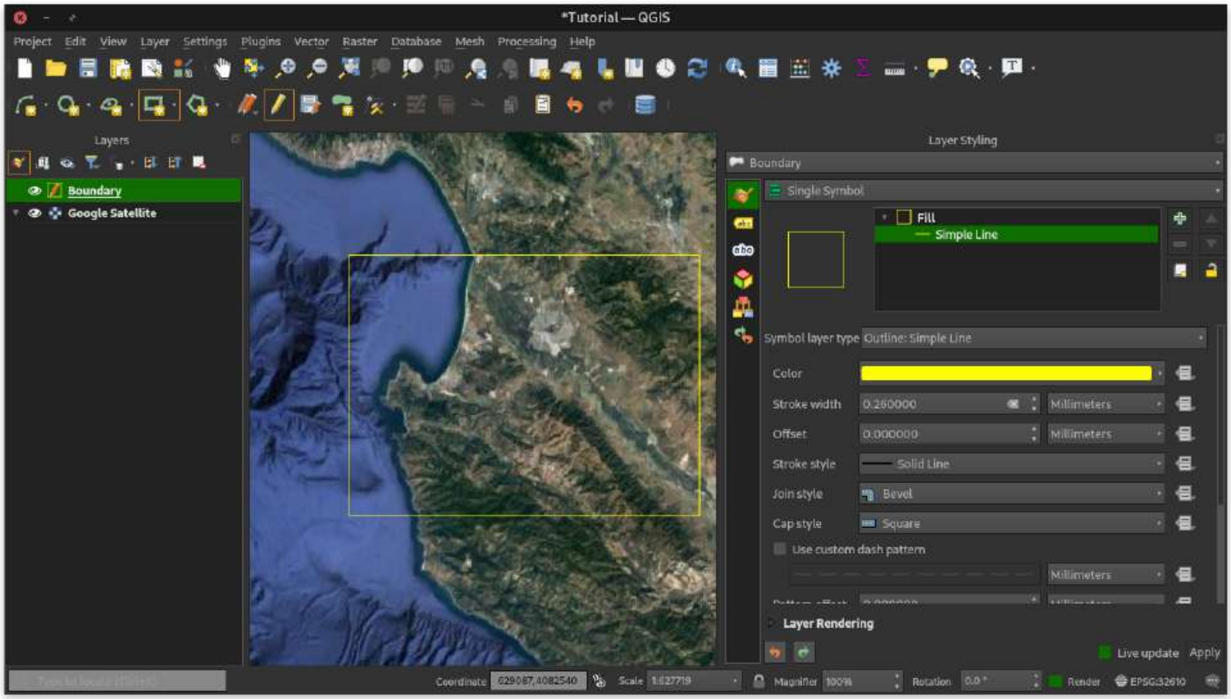
Fields List

Name	Type	Length	Precision
id	Integer	10	

Make sure you have the toolbar for digitizing (vector drawing) enabled; go to `View > Toolbars > Digitizing` and `View > Toolbars > Shape Digitizing`. Make sure you have your boundary layer selected in the layer view, and click the yellow pencil icon (Toggle Editing) on the top. The shape icons should enable; go ahead and pick the rectangle one. Your cursor will turn into crosshairs; left-click once at the top left corner of your desired area, then right-click at the bottom right to finish drawing. You should get a large rectangle drawn on your image as seen:



Obviously, a giant, opaque shape over our area of interest is a problem. Enable the layer styling panel (`View > Panels > Layer Styling`) and drag it over to a side of the screen. You should see a layer-like view in that panel similar to Photoshop; what we want to change is the Simple Fill. Simply change that to "Outline: Simple Line" in the Layer Type dropdown; this'll change it to a simple outline. You can select a color for it in the color picker; the end result should look like the below.



Symbol layer type: Outline: Simple Line

Color

Stroke width: 0.260000 Millimeters

Offset: 0.000000 Millimeters

Stroke style: Solid Line

Join style: Bevel

Cap style: Square

□ Use custom dash pattern

Millimeters

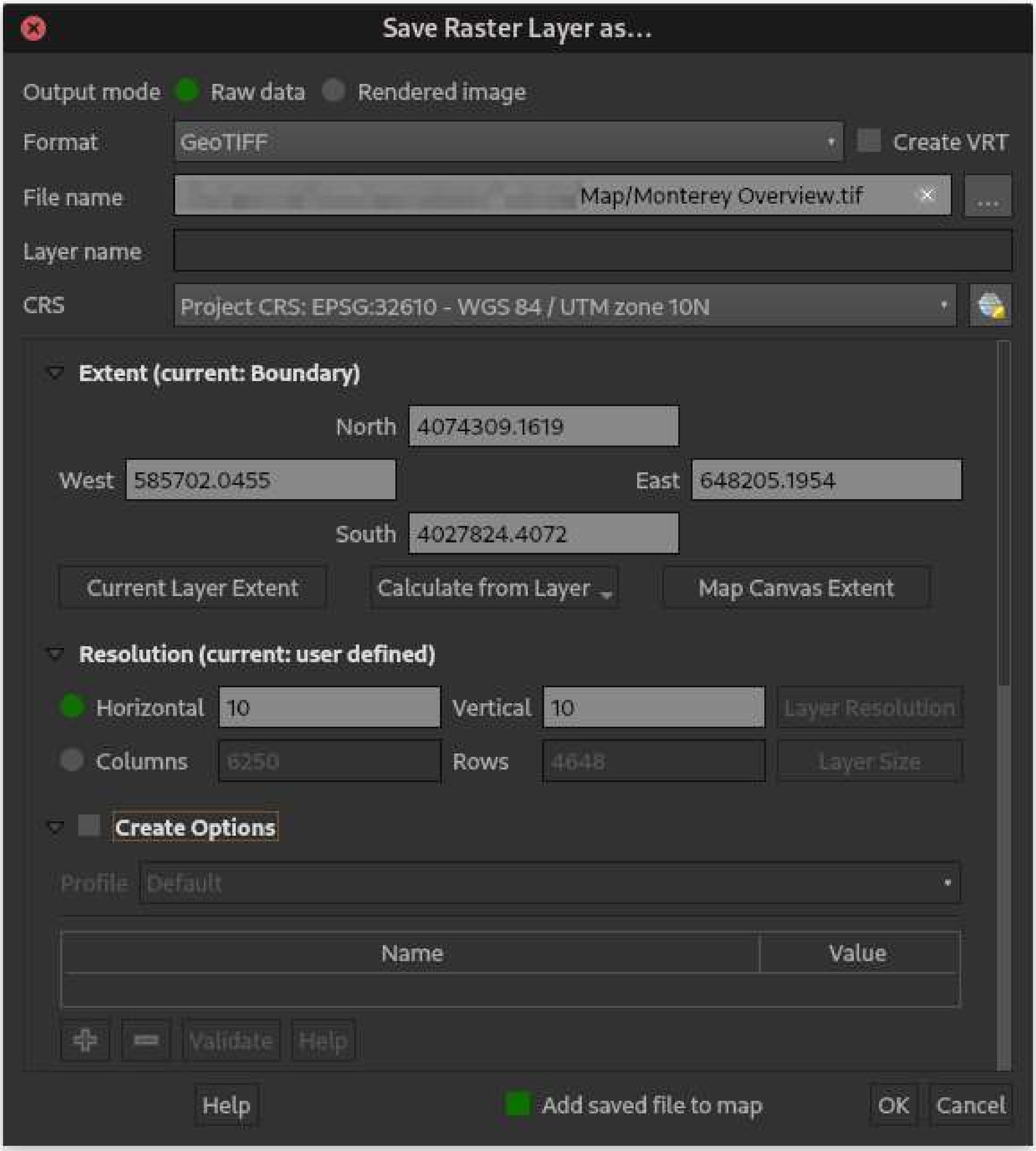
Once you're done drawing your boundary, click the orange pencil to stop editing, and the save icon to the right of it.

Side story: Downloading satellite imagery

First, as a note, you can pull from any service with tile-enabled raster imagery provision; those with accounts at MAXAR (ATAK-MIL's imagery provider) can pull from those, though I'll leave out how here.

For those using ATAK, or those who just want to have an offline copy of all their imagery, you can do that too with QGIS! Simply right click the layer with the satellite imagery of interest and pick `Export > Save As...`. Uncheck "Create VRT"; we want a single image, not a series of multiple files, for this. Pick a good location to save, just like we did when creating our boundary layer. There's a CRS selection field again; just like before, select the project CRS we determined earlier. To determine which area to save, just go to `Calculate From Layer` and select the boundary layer we just created. The coordinates will auto-populate from there.

The resolution is given in pixels per meter; if you want one pixel per meter, input 1 for both horizontal and vertical resolution. If you want to add the file to your map, click `Add saved file to map` at the bottom and hit OK. The settings should look like the below image; a progress bar will appear in the bottom panel of the main QGIS window, and a layer with that saved image will appear in your layers view when complete.



Save Raster Layer as...

Output mode ● Raw data ○ Rendered image

Format GeoTIFF □ Create VRT

File name [redacted]Map/Monterey Overview.tif

Layer name

CRS Project CRS: EPSG:32610 - WGS 84 / UTM zone 10N

Extent (current: Boundary)

North 4074309.1619

West 585702.0455 East 648205.1954

South 4027824.4072

Resolution (current: user defined)

● Horizontal 10 Vertical 10

○ Columns 6250 Rows 4648

□ Create Options

Profile Default

Name	Value

☑ Add saved file to map

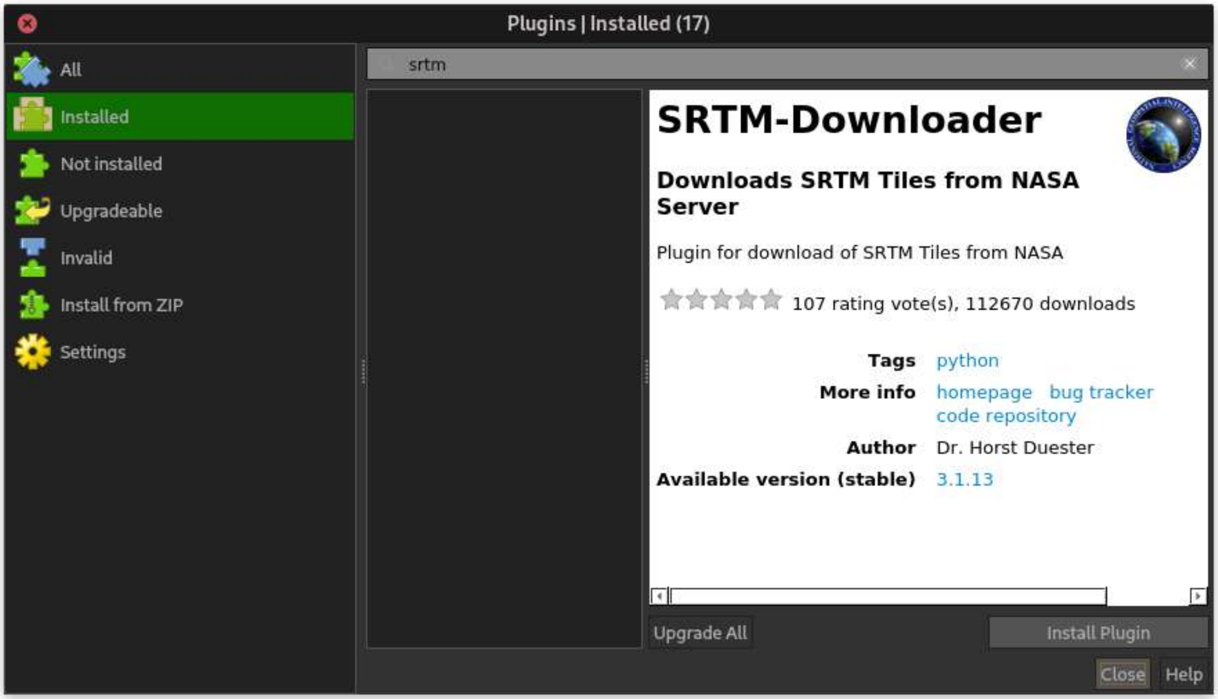
You can actually use this imagery directly in ATAK. Just drag it onto your phone after you've connected the phone to the computer via USB, and drag it into the `atak > imagery` folder. It'll now be available in the ATAK app just like any other imagery/map source.

(There's also a relatively easy way to use elevation data downloaded in the same manner and from the same sources as I'll describe below for height data in ATAK, but that's a topic for another time.)

Adding elevation data

Now comes the trickier part. The best source of up-to-date elevation information, publicly, for us are NASA's SRTM files, essentially using a laser from a satellite to measure height above sea level. QGIS actually makes this pretty easy for us; there's a plugin available to do this directly.

Go to `Plugins > Manage and Install Plugins`; in the search bar, type SRTM. The name of the plugin is "SRTM-Downloader", itself open-source as well. Click `Install plugin`.



srtm

SRTM-Downloader

Downloads SRTM Tiles from NASA Server

Plugin for download of SRTM Tiles from NASA

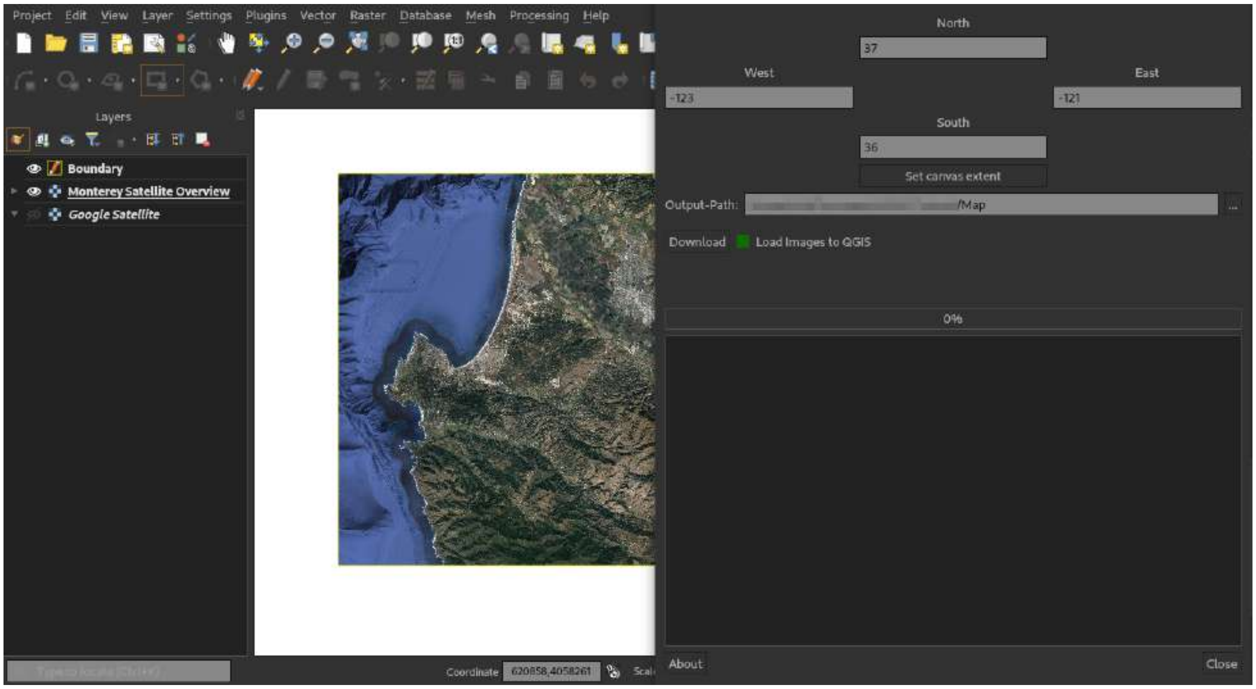
☆☆☆☆☆ 107 rating vote(s), 112670 downloads

Tags	python
More info	homepage bug tracker code repository
Author	Dr. Horst Duester
Available version (stable)	3.1.13

After it's installed, we should see a new entry under Plugins at `Plugins > SRTM-Downloader > SRTM Downloader`.

We can't use it just yet, however; NASA requires us to have an account at their EarthData site. Registration is easy; go to the new user site and register. Once you've finished that process, go back to QGIS.

Zoom out and make sure the visible area of the map is roughly equal to the boundary of our area of interest; once you're there, go to the SRTM Downloader entry I just mentioned. A dialog like the below will pop up; hit "Set canvas extent", which will set the extent of the SRTM images to the visible area in the QGIS view. Select a good folder to save the downloaded data in, and click Download.

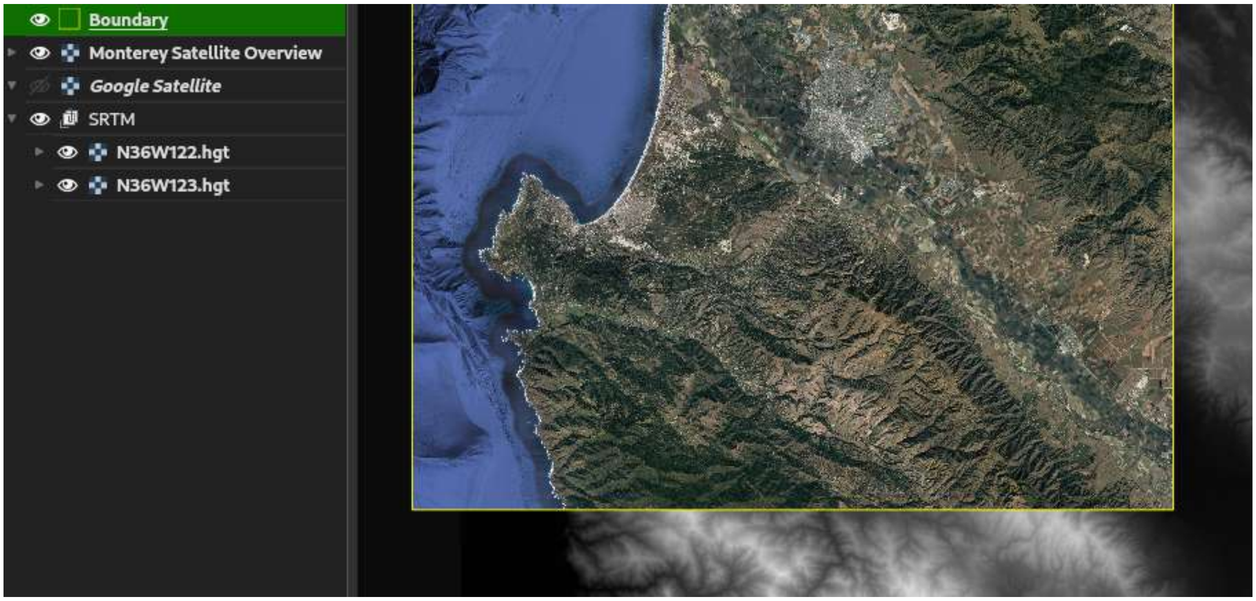

Project Edit View Layer Settings Plugins Vector Raster Database Mesh Processing Help
Layers
Boundary
Monterey Satellite Overview
Google Satellite
Coordinate 620858,4058261
Scale

North: 37
West: -123
East: -121
South: 36
Set canvas extent
Output-Path: [redacted]/Map ...
Download ■ Load Images to QGIS
0%
About
Close

When a prompt comes up asking for a username and password, use your EarthData login from earlier. Progress bars will appear and fill. Once you're done, you should see a couple new layers (as many as needed to fill the requested area; often much larger than needed) appear on your map and in the Layers panel.

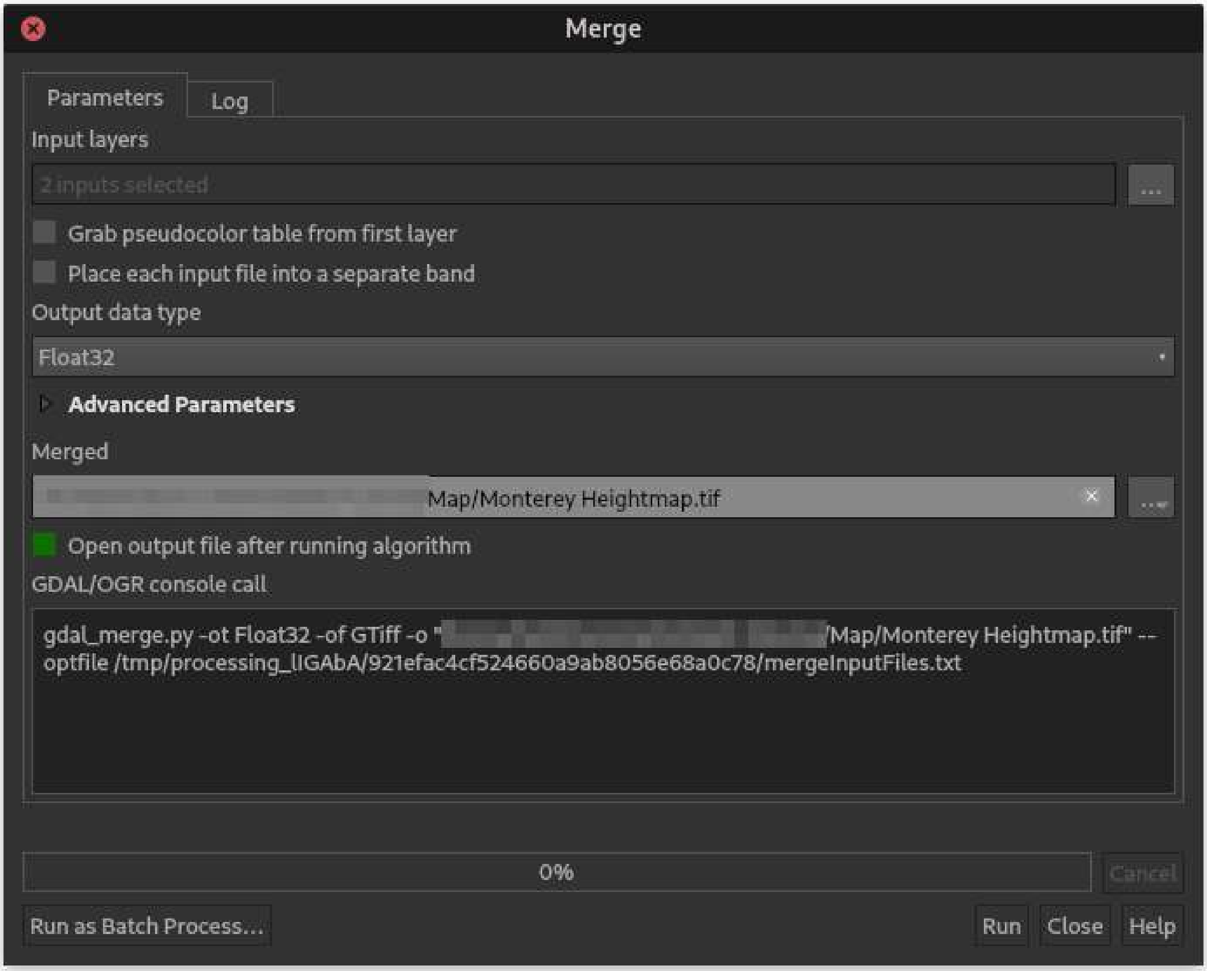
Now's a good time to make a group of layers to contain these; go to ‘`Add Group` at the top of the layer panel. These work just like a normal file viewer's folders, or like layer groups in Photoshop or similar. Drag your SRTM files into there, and give the group a sensible name. You might see a color ramp below each file; this is normal, and clicking the arrows will let you collapse them.

At this point, I have my layers set up like this:



It'll make our life a lot easier if all this height data was in one file... luckily, there's an easy way to do this. Go to `Raster > Miscellaneous > Merge` at the top bar. Select the SRTM layers using the `...` button next to "Input layers." Under "Merged," select a good place to save again, making sure not to hit "Save to

Temporary File," but "Save to File." For filetype, select TIF (it should be the first option selected). We still have a few more steps here, so don't name it "Final Elevation" or anything like that. Hit `Run`.



Input layers: 2 inputs selected

□ Grab pseudocolor table from first layer

□ Place each input file into a separate band

Output data type: Float32

Merged: [redacted]Map/Monterey Heightmap.tif

🟩 Open output file after running algorithm

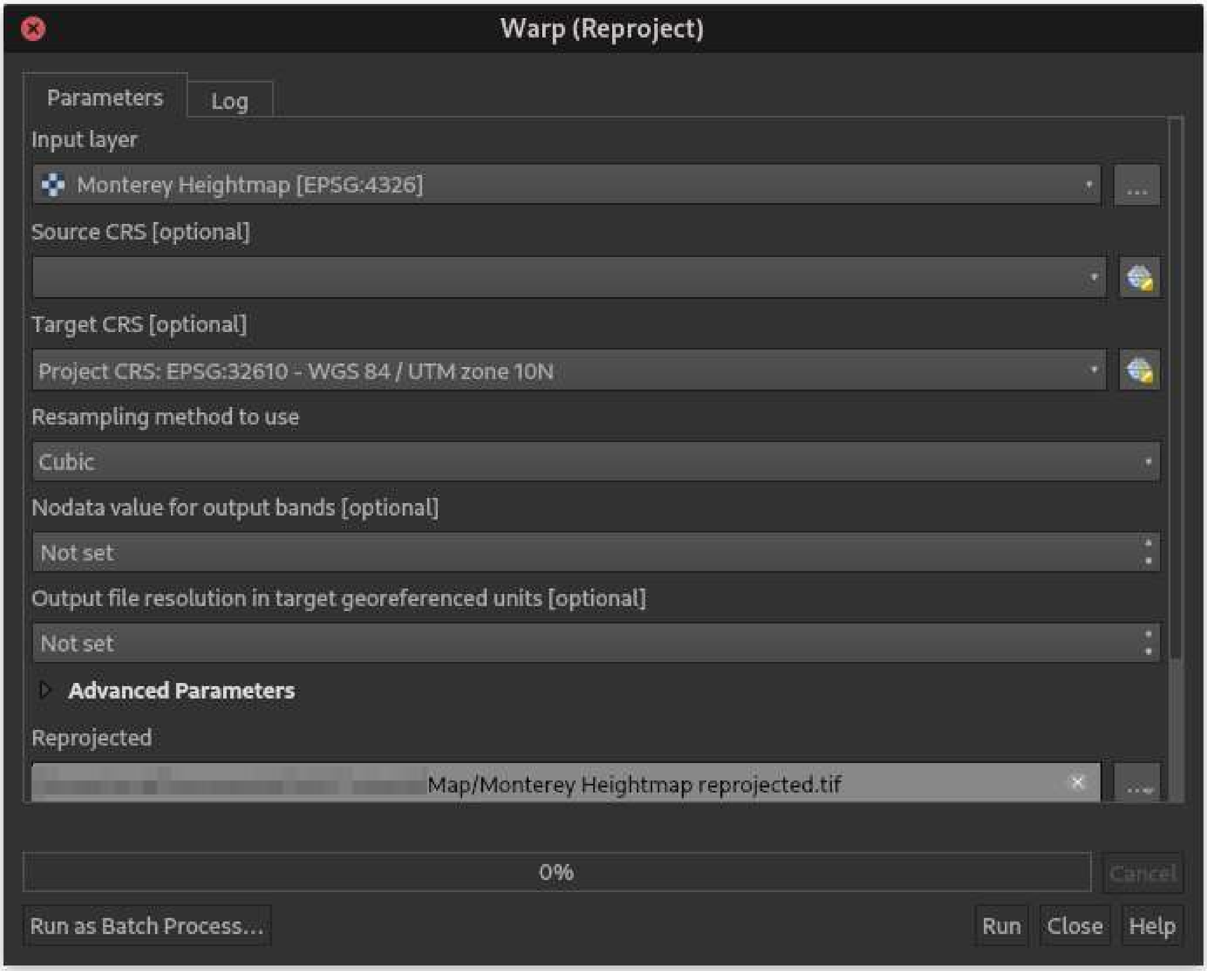
GDAL/OGR console call

```
gdal_merge.py -ot Float32 -of GTiff -o "[redacted]/Map/Monterey Heightmap.tif" --
optfile /tmp/processing_lIGAbA/921efac4cf524660a9ab8056e68a0c78/mergeInputFiles.txt
```

The result should be added to your map. You can now delete the original files from your map by right-clicking on each and selecting `Remove Layer` in the Layers view.

We still need to do a few more steps before this is completely usable, however. Step one is ensuring it's in the same CRS as our project. This is easy - go to `Raster > Projects > Warp (Reproject)`. Leave "Source CRS" blank and make sure "Target CRS" is the project CRS we're using.

Set "Resampling method to use" to "Cubic" (this'll preserve data better instead of clipping it, as the image is warped around to fit the right CRS). Pick a good output file, again as a TIF, leave everything else unset, and click `Run`.



Input layer: Monterey Heightmap [EPSG:4326]

Source CRS [optional]:

Target CRS [optional]: Project CRS: EPSG:32610 - WGS 84 / UTM zone 10N

Resampling method to use: Cubic

Nodata value for output bands [optional]: Not set

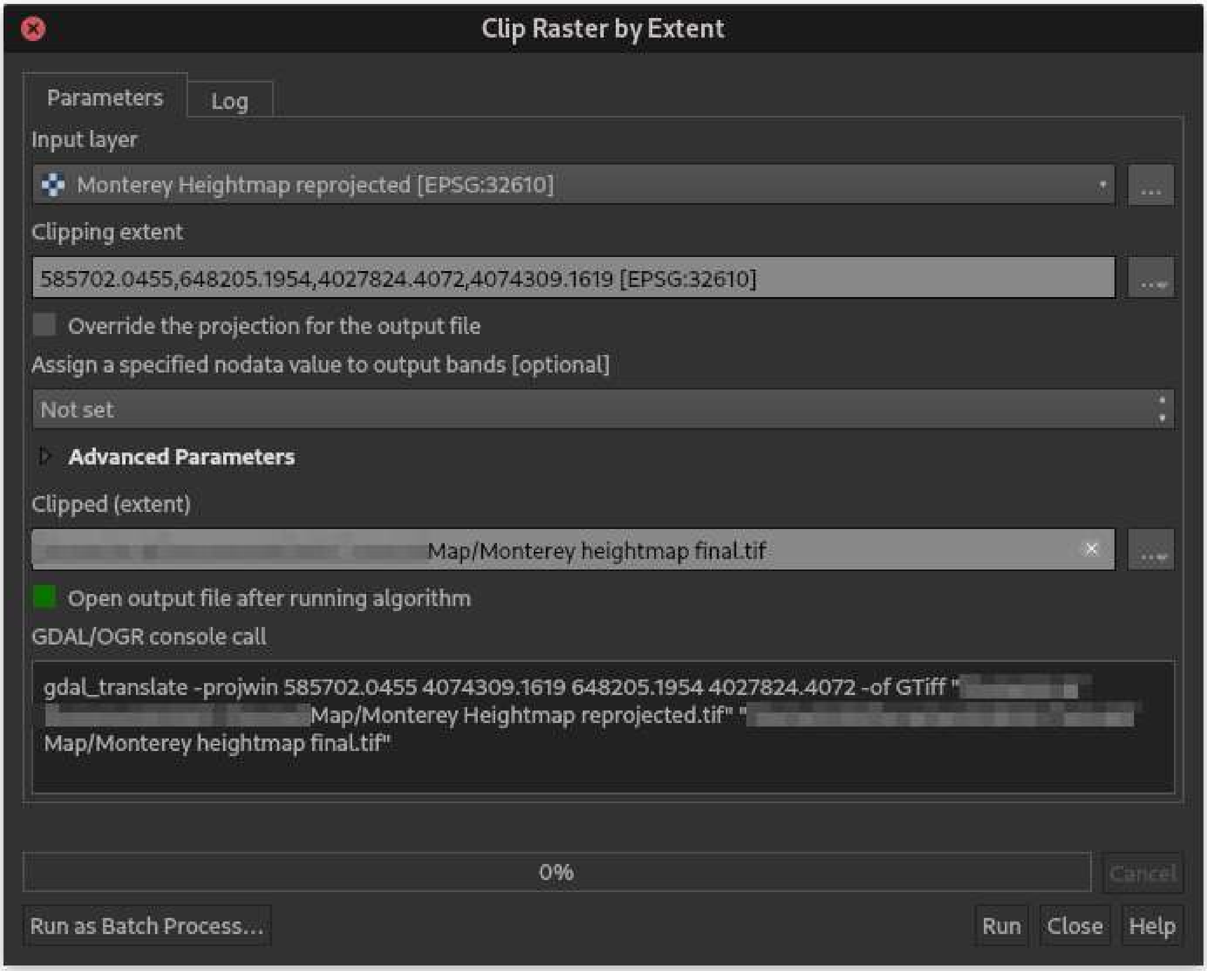
Output file resolution in target georeferenced units [optional]: Not set

Reprojected: [redacted]Map/Monterey Heightmap reprojected.tif

Don't expect anything to change visually, but this will ensure that when the time comes for us to make contour data, it'll be in actual meters.

The next step is to clip the SRTM data to our area of interest, so we don't bog down our map with a ton of contour lines or data outside what we need. There's a pattern starting to form; go to `Raster > Extraction > Clip Raster by Extent`.

Next to "Clipping extent," select ‘`Calculate from Layer` and pick your boundary layer from earlier. Under "Clipped (extent)," pick a good output file. *Now* you can call this one final.



Input layer

Monterey Heightmap reprojected [EPSG:32610]

Clipping extent

585702.0455,648205.1954,4027824.4072,4074309.1619 [EPSG:32610]

□ Override the projection for the output file

Assign a specified nodata value to output bands [optional]

Not set

Advanced Parameters

Clipped (extent)

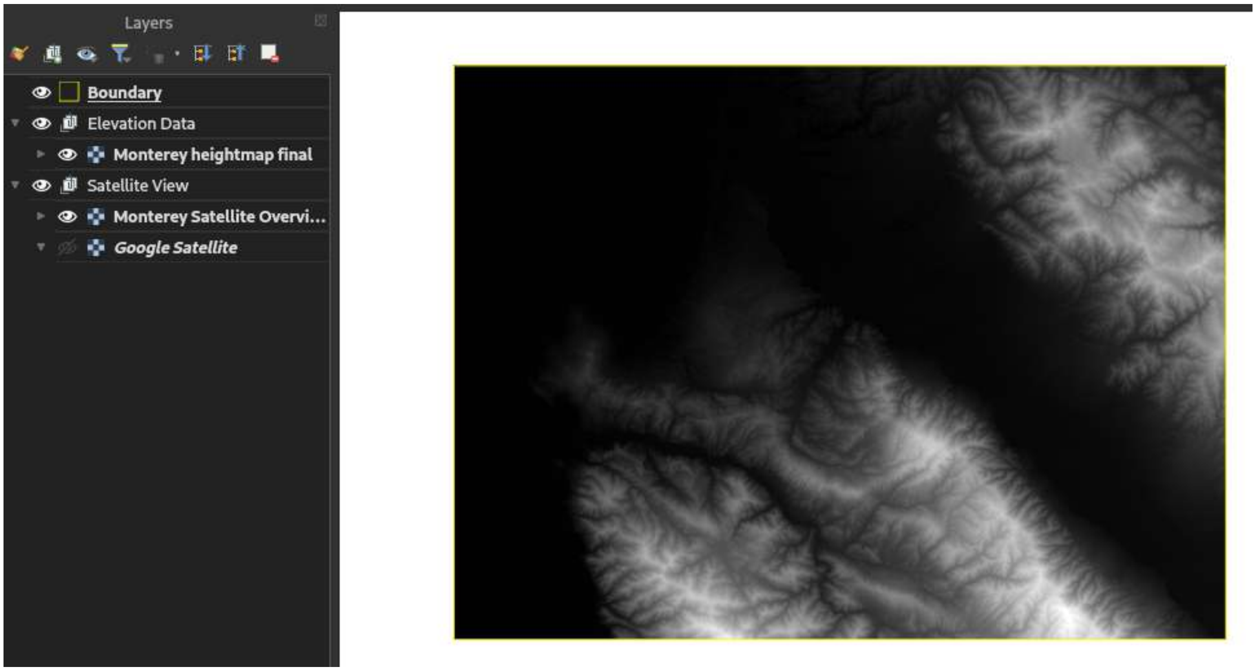
[redacted]Map/Monterey heightmap final.tif

☑ Open output file after running algorithm

GDAL/OGR console call

```
gdal_translate -projwin 585702.0455 4074309.1619 648205.1954 4027824.4072 -of GTiff "[redacted]
[redacted]Map/Monterey Heightmap reprojected.tif" "[redacted]
Map/Monterey heightmap final.tif"
```

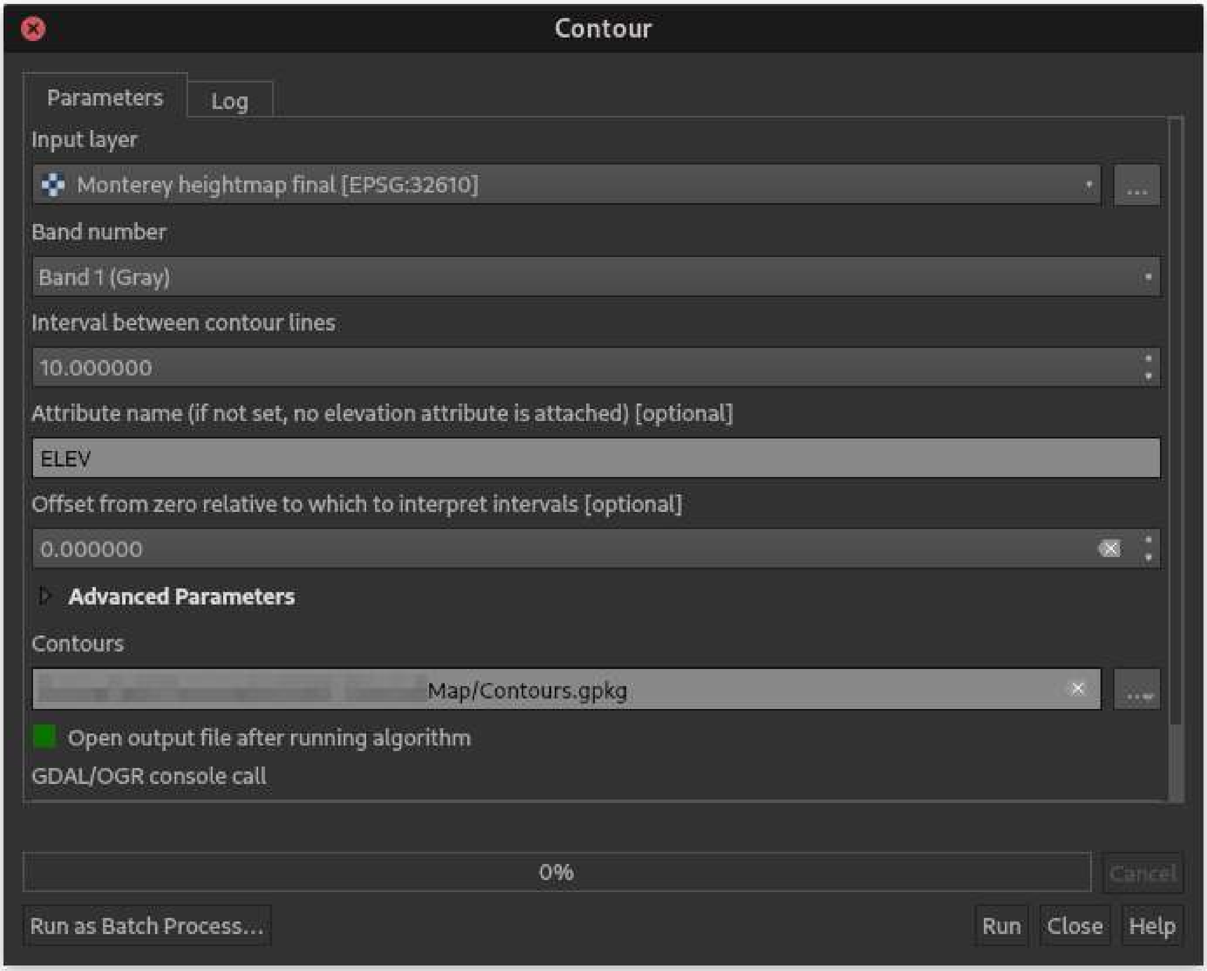
Go ahead and delete all the intermediate SRTM layers on the Layers panel; the result should leave you with one neat heightmap within the bounds of what you selected earlier. If you haven't figured this out yet, you can hide and show layers and groups using the eyeball icons. Right now, my map looks like the following.



Converting elevation data to contours

The latest version of QGIS actually includes a feature to draw heightmaps as contours by default; however, this can behave strangely when zooming in or out and isn't really workable for professional use yet, in my opinion, and doesn't allow us to label contours, which is essential for use on a printed map.

Luckily, there is, yet again, a pretty easy way to extract contours from the elevation data we just finished processing. Select the SRTM layer in your map and go to `Raster > Extraction > Contour` from the top menu. Make sure the SRTM layer is selected in '`Input layer`. Leave everything as it is by default (although you can change the contour elevation default of 10 meters in elevation per contour interval, if you want), select a good place to save the file, and click `Run`. This may take some time, particularly on older/less powerful computers.



Contour

Parameters | Log

Input layer: Monterey heightmap final [EPSG:32610]

Band number: Band 1 (Gray)

Interval between contour lines: 10.000000

Attribute name (if not set, no elevation attribute is attached) [optional]: ELEV

Offset from zero relative to which to interpret intervals [optional]: 0.000000

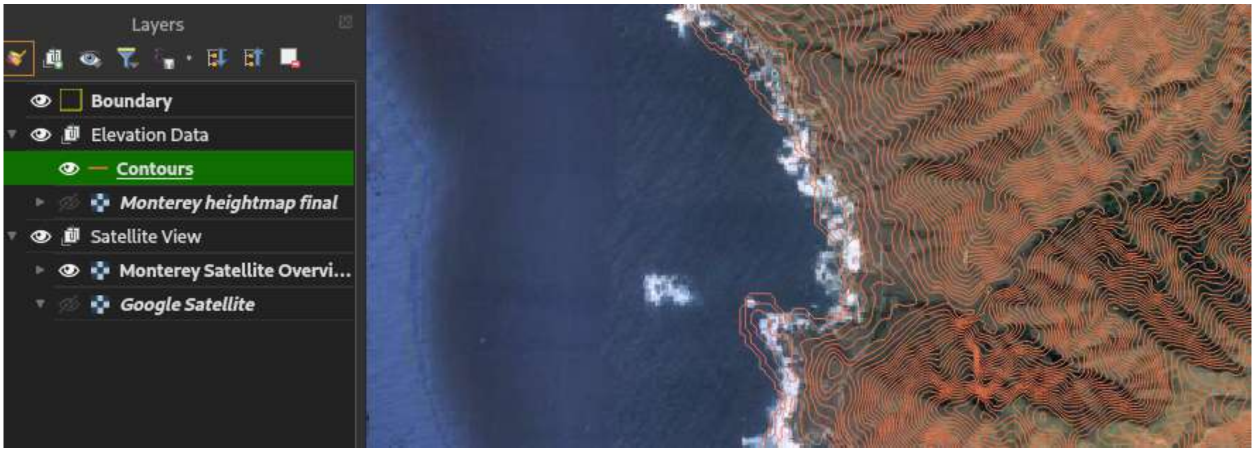
Advanced Parameters

Contours: [redacted]Map/Contours.gpkg

Open output file after running algorithm

GDAL/OGR console call

The end result is, well, a series of contour lines on your map. I applied them to a portion of my map over the satellite view so you can get a sense of what it might look like for you.



Styling contour lines for usability

You might notice that the contour lines are pretty messy, not colored as you might expect if you're familiar with military topographic maps, and have no labeling of any kind or index lines. It's time to dig into the

Layer Styling panel again to turn this raw data into something usable with the naked eye.

The first step is to change the symbols to two types - index contour lines, and regular, unlabeled contour lines. In order to do this you'll need to change the symbol type to "Rule-based" in the top dropdown of the layer styling panel. The view should change to a list of symbols - now, containing only one, blank-labeled line.

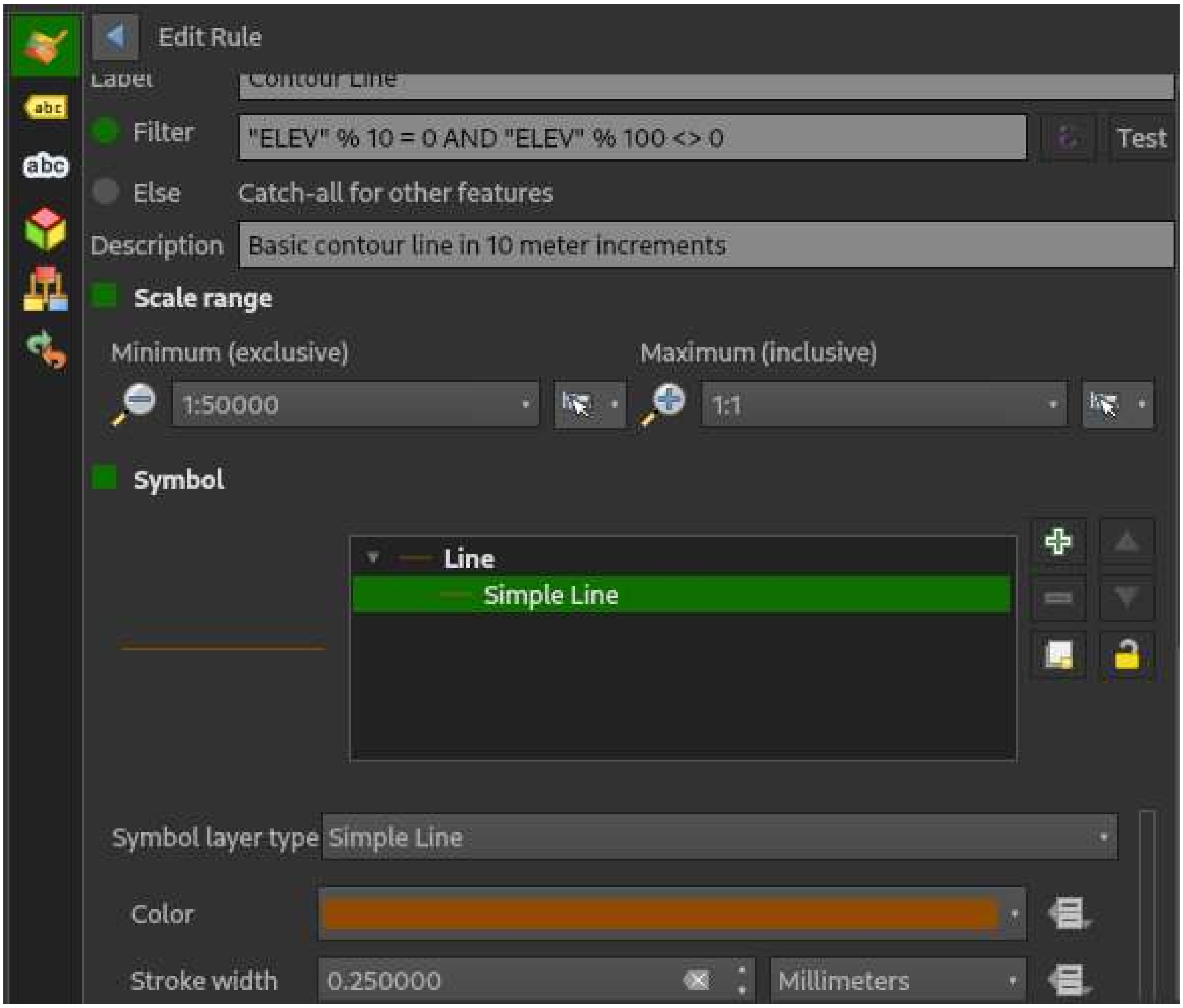
Double-click on that; you should now see a slightly more familiar symbol window appear. Let's label this one as "Contour Line." To filter out just those contour lines that are in 10m increments, but not 100m increments, add this equation into the `Filter` box:

```
"ELEV" % 10 = 0 AND "ELEV" % 100 <> 0
```

As far as QGIS is concerned in its filter boxes, items in double quotes (`"`) are variable names, the `<>` symbol is "not equals" to, and `=` is equals to. (This is *not* the same as `=` meaning assignment like in programming languages, for my fellow nerds out there. The `%` is the modulo operator, i.e. "how much is left from the number on the left when you divide by the number on the right". So 150 modulo 10 is 0, since 150 is divisible evenly by 10, but 155 % 10 is equal to 5, since 5 is left over. This is how we're checking if the ELEV (elevation) attribute for each of those contour-line geometry strings is divisible by 10 (so a contour line), but not divisible by 100 (filtering out 100-meter increments for index contours)

Under `Simple Line` in the "layers" view you might remember from earlier, we can go ahead and change the symbol for these basic contour lines to something more familiar. I personally use 0.25 mm width for contour lines unless I'm using a very large-scale map, and I'll set the color to be an opaque brown like on most military-produced topo maps (`##914a02ff` in hex notation.)

I'll also set a scale range for these, as I don't want 10 meter contour lines visible at a large scale. Click `Scale Range` and set the minimum to 50,000 and the maximum to 1, meaning these lines will show up when the map is between 1:1 and 1:50,000 scale.



Label: Contour Line

● Filter: "ELEV" % 10 = 0 AND "ELEV" % 100 <> 0

○ Else: Catch-all for other features

Description: Basic contour line in 10 meter increments

🟩 Scale range

Minimum (exclusive): 1:50000

Maximum (inclusive): 1:1

🟩 Symbol

- Line
  - Simple Line

Symbol layer type: Simple Line

Color

Stroke width: 0.250000 Millimeters

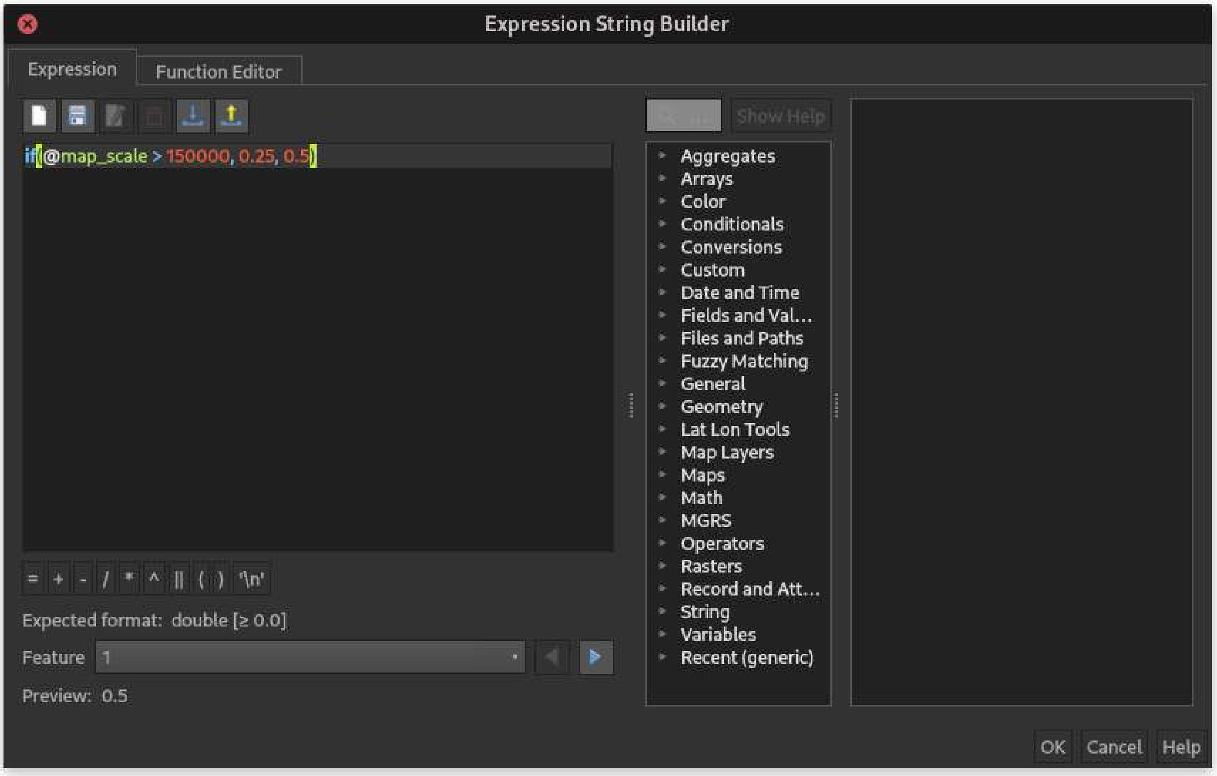
Click the back button, and let's add another rule to take care of our index contours.

We could click the green plus button and do another rule from scratch, but to save time, right-click our existing rule and click `Copy` (not `Copy Symbol`), then right-click just below it in the open space and hit `Paste`. Double-click on the resulting rule that appears to edit this one, too. The filter this time is simply `"ELEV" % 100 = 0`. I'll make this one the same color, but with a stroke width of 0.5 mm, and a scale range of 1:1 through 1:100,000.

We're going to make this width change, though, depending on scale, as we don't need 0.5 mm index contours cluttering our map when we zoom out. Next to width, hit the boxes icon, and click `Edit` in the resulting dialog. In the text box on the right, paste the following:

```
if(@map_scale > 150000, 0.25, 0.5)
```

All this is doing is telling QGIS that if the map's scale (`@map_scale`) is greater than 1:150,000, use the first number (0.25) for the value, and the second if not. Any box with that same icon can be controlled by an expression like this, although I won't cover this again too much as you can really go down the rabbit hole on expression-defined values.



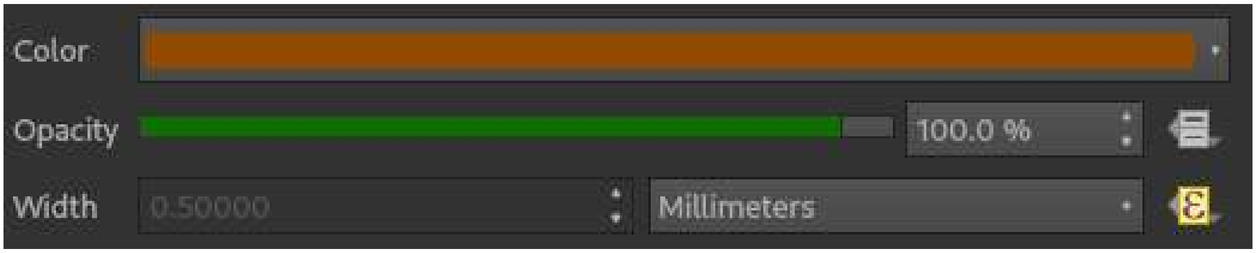
```
if(@map_scale > 150000, 0.25, 0.5)
```

Expected format: double [≥ 0.0]

Feature 1

Preview: 0.5

When you hit OK, you should see the width value grayed out and that icon turn yellow with what looks like an E in it - all this is doing is signifying that this value is controlled by an expression.

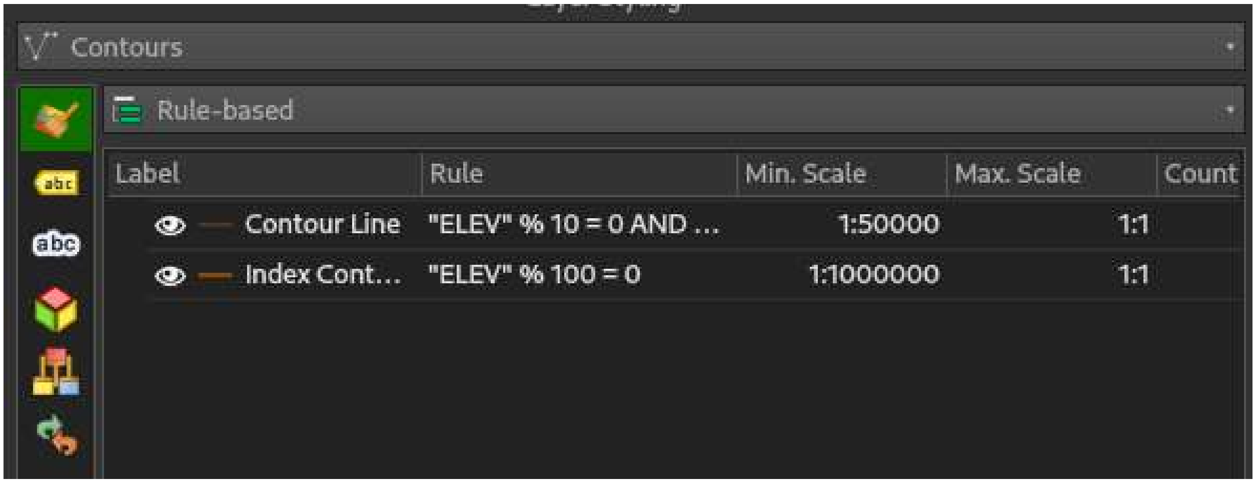


Color

Opacity 100.0 %

Width 0.50000 Millimeters

Backing out of this rule, now, we should see our list of symbols for the contour lines look as follows.

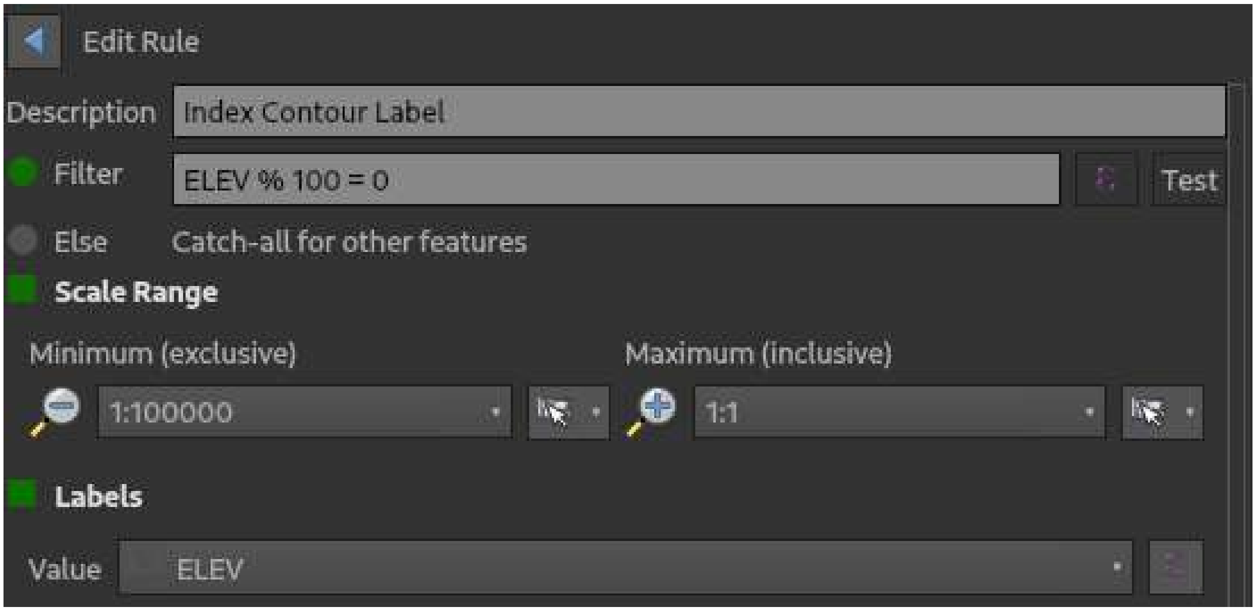


Label	Rule	Min. Scale	Max. Scale	Count
Contour Line	"ELEV" % 10 = 0 AND …	1:50000	1:1	
Index Cont…	"ELEV" % 100 = 0	1:1000000	1:1	

All of this is useless without knowing what our actual elevation is, so let's go ahead and add some labels to those index contours.

Click on the next icon down, the yellow "ABC" tag, and change `No labels` to `Rule-based Labeling`. This looks similar to the symbol rules, and for good reason - let's add a rule to label index contours only by clicking on the green plus sign at the bottom. Label this one appropriately, and just like the symbol for index contour, the filter is `ELEV % 100 = 0`. We'll set the scale range to be 1:1 at maximum, all the wawy to 1:100,000 at minimum.

For the `Value`, we can select ELEV from the dropdown, or type `"ELEV"` or `ELEV` directly in the box. (The `"` on either side only matters when variable names have spaces in them.)



Description: Index Contour Label

● Filter: ELEV % 100 = 0

○ Else: Catch-all for other features

☑ Scale Range

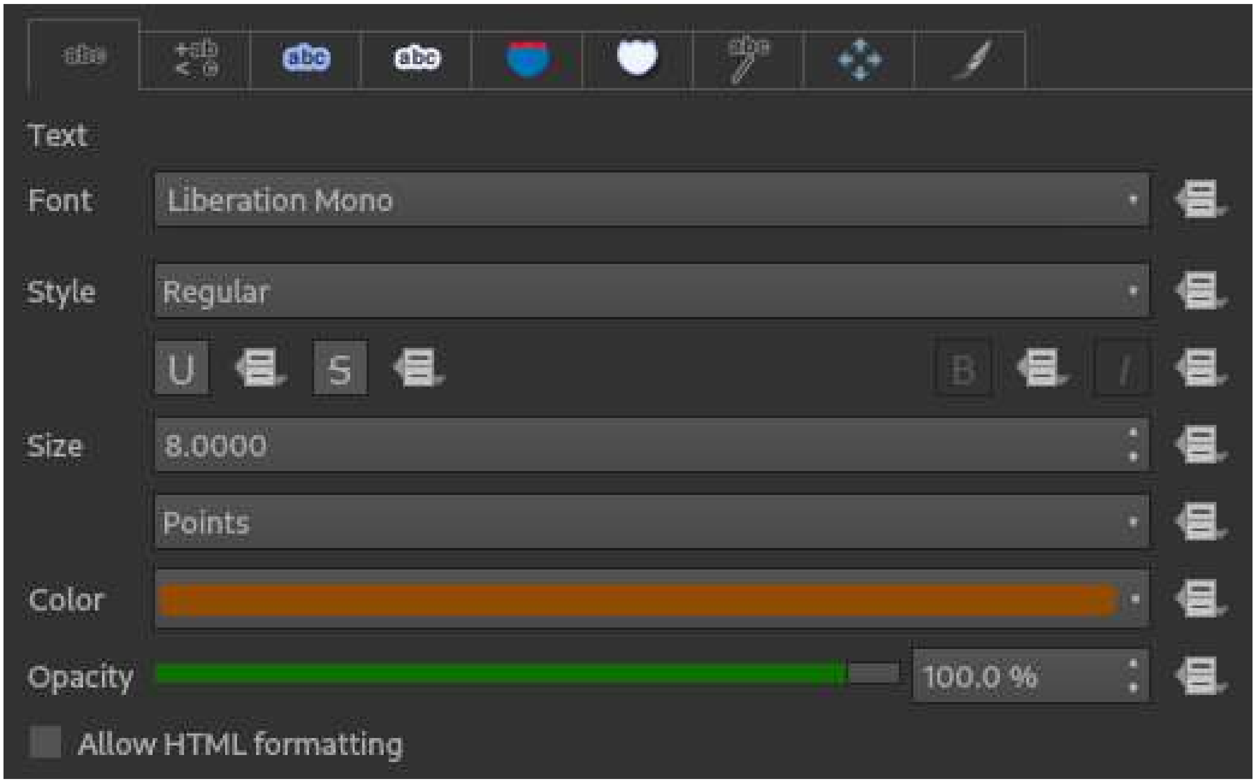
Minimum (exclusive): 1:100000

Maximum (inclusive): 1:1

☑ Labels

Value: ELEV

Scrolling down, under the "abc" tab, we can pick an appropriate font, and set the labels to be the same color as the contour lines themselves. The settings I use look like the folllowing, although if you're savvy, you can make the font size change with the map scale, just like we did with the width of the index contour line symbols themselves.



Text

Font: Liberation Mono

Style: Regular

U S B I

Size: 8.0000

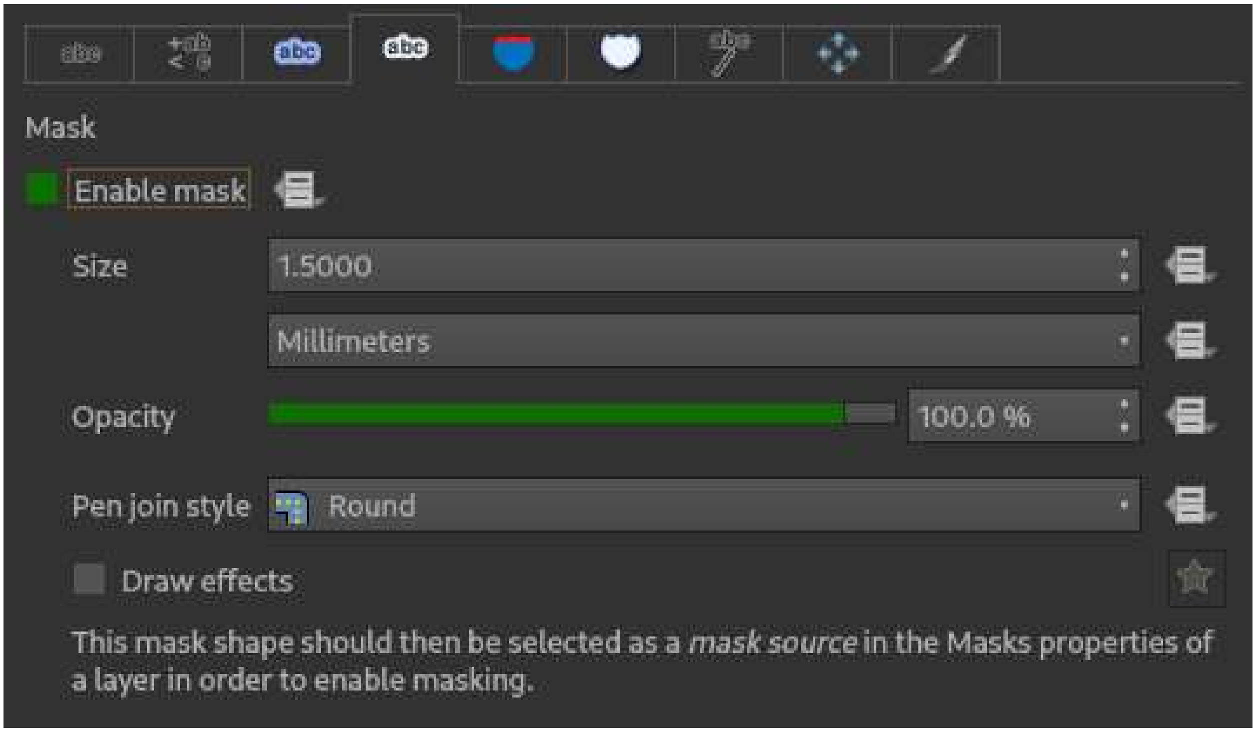
Points

Color

Opacity: 100.0 %

□ Allow HTML formatting

The next tab to pick is the mask tab; we want the contour labels to have a bit of a buffer between themselves and the contour lines so they're readable. I typically keep the size as it is by default, 1.5 mm; all I need to do is hit `Enable Mask` on this tab. Nothing should change just yet; we'll need to do another step a little later on to enable the masking effect on the symbols.



Mask

☑ Enable mask

Size: 1.5000

Millimeters

Opacity: 100.0 %

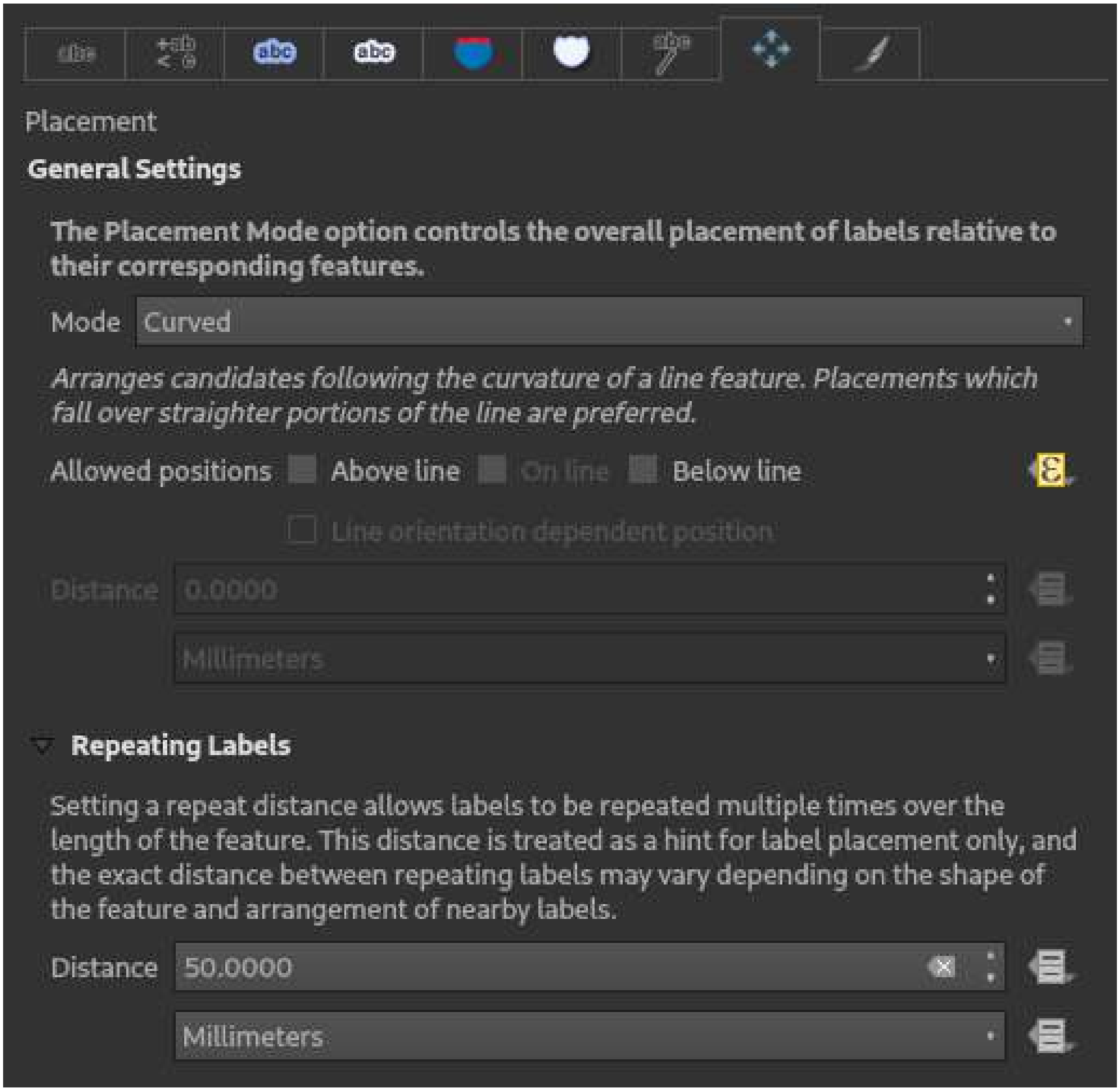
Pen join style: Round

□ Draw effects

This mask shape should then be selected as a *mask source* in the Masks properties of a layer in order to enable masking.

The next tab is placement, delineated by four arrows. Pick `Curved` for the mode to allow the labels to follow their respective contour lines. We'll need to hit the data-defined box by `Allowed positions` just like we did for the index contour widths; in the left text box that appears, simply type `'OL'` . (Values surrounded by single quotes, `'like this'`, are treated as strings by QGIS). This will force the index contour labels to appear on the lines.

Scroll down to "Repeating labels" to set the `Distance` to an appropriate value; I've gone with 50 mm. For every 50 mm, or 5 centimeters, on my final map, then, the index contours will be labeled, allowing me to find my elevation easily. The ending placement settings for the index contour labels, for me, look like the following.



Placement

General Settings

The Placement Mode option controls the overall placement of labels relative to their corresponding features.

Mode Curved

Arranges candidates following the curvature of a line feature. Placements which fall over straighter portions of the line are preferred.

Allowed positions ■ Above line ■ On line ■ Below line

□ Line orientation dependent position

Distance 0.0000

Millimeters

Repeating Labels

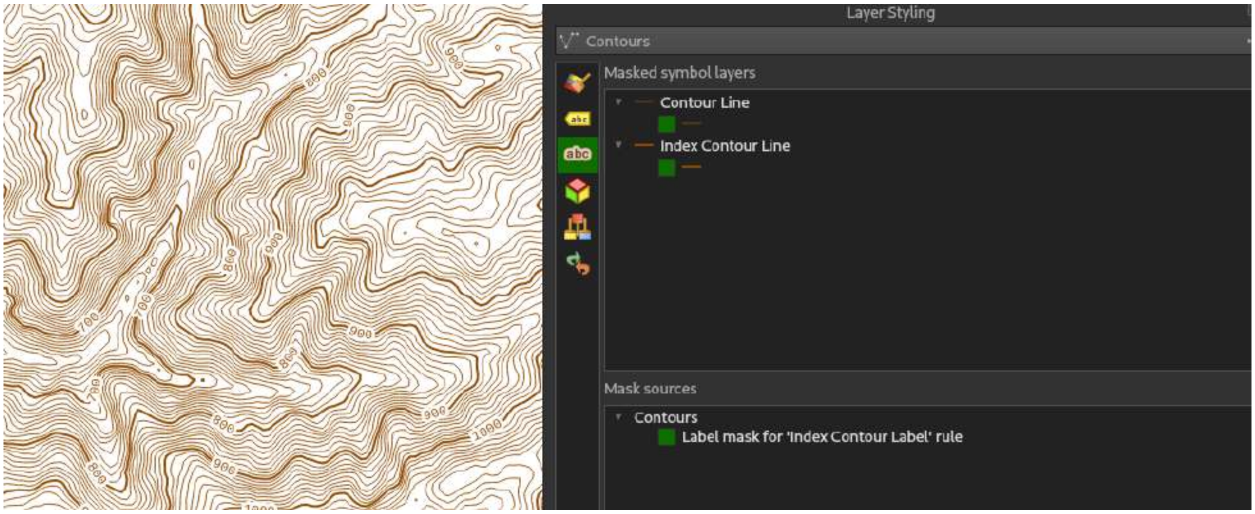
Setting a repeat distance allows labels to be repeated multiple times over the length of the feature. This distance is treated as a hint for label placement only, and the exact distance between repeating labels may vary depending on the shape of the feature and arrangement of nearby labels.

Distance 50.0000

Millimeters

There are other options for labeling, dozens of them. You can set labels to generate symbols of their own, dynamically change font based on value, and so on, but for our purposes, we can leave everything as it is by default and back out of defining this label rule.

Our third and final step is to make the labels mask out the index contour lines themselves, and this is the easiest part. One more down from the labeling tab on the left side of the Layer Styling panel, select the Masking tab. Once you're there, select both symbols we made for the contour lines in the top panel, and the label in the bottom panel. The end result will look like the following.



Layer Styling

Contours

Masked symbol layers

- Contour Line
  - 🟩
- Index Contour Line
  - 🟩

Mask sources

- Contours
  - 🟩 Label mask for 'Index Contour Label' rule

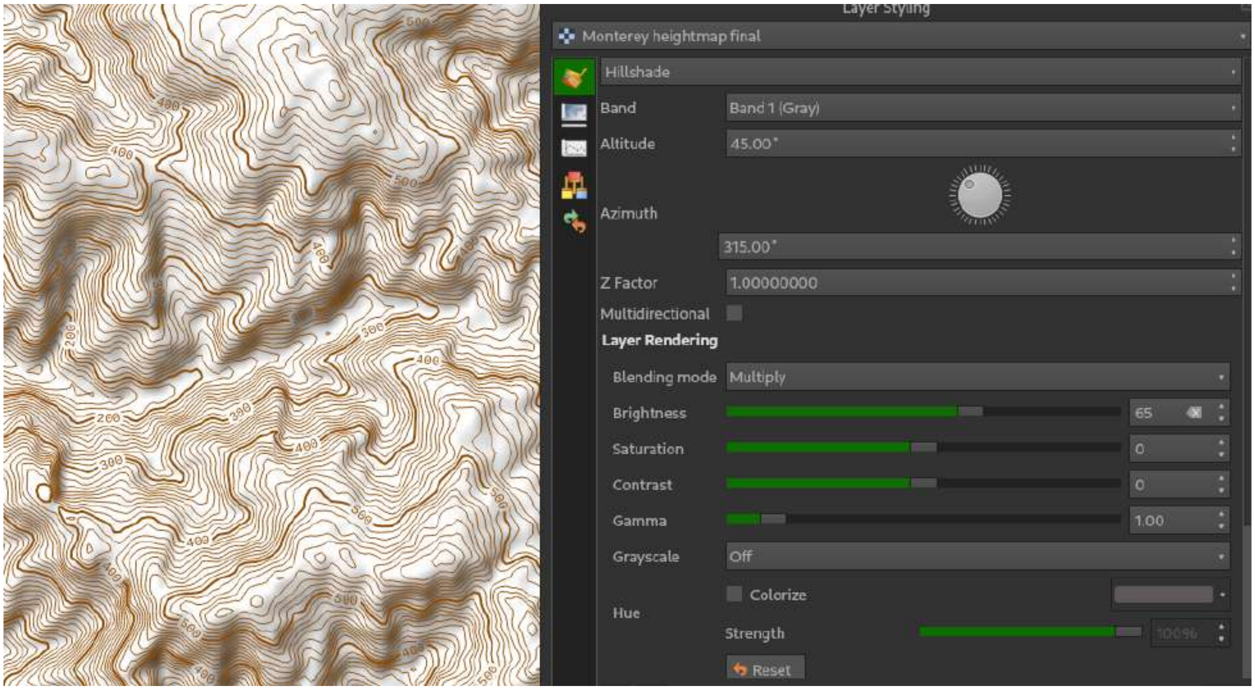
This leaves us with a complete set of contours for our map!

Side story: Adding hillshading for clarity of elevation

For those experienced with topographic maps, or familiar with land navigation, the contour lines are all we need. But for those looking to more accurately judge whether that's a hill or a depression accurately, we can add hill shading to our map.

To do this, re-enable our final processed SRTM file on the map by clicking on the eye icon next to it, then go back into the Layer Styling panel. Change the type at the very top to "Hillshade". Under `Layer Rendering`, let's change the '`Blending mode` to `Multiply`; this will allow the hillshade to show through subsequent layers we add without replacing it. Feel free to adjust the brightness until you get something where the baseline is white.

At the very bottom, make sure you select `Cubic` under `Resampling` for both "Zoomed in" and "Zoomed out" options; this will allow a smooth shading to occur, instead of the strange, blocky appearance that might result with "Nearest Neighbor" selected instead. Most of the settings, and the resulting look on the map, are visible in the following image.



Layer Styling

Monterey heightmap final

Hillshade

Band: Band 1 (Gray)

Altitude: 45.00°

Azimuth: 315.00°

Z Factor: 1.00000000

Multidirectional: □

Layer Rendering

Blending mode: Multiply

Brightness: 65

Saturation: 0

Contrast: 0

Gamma: 1.00

Grayscale: Off

Hue: □ Colorize

Strength: 100%

Reset

Adding linear and natural features from OpenStreetMap

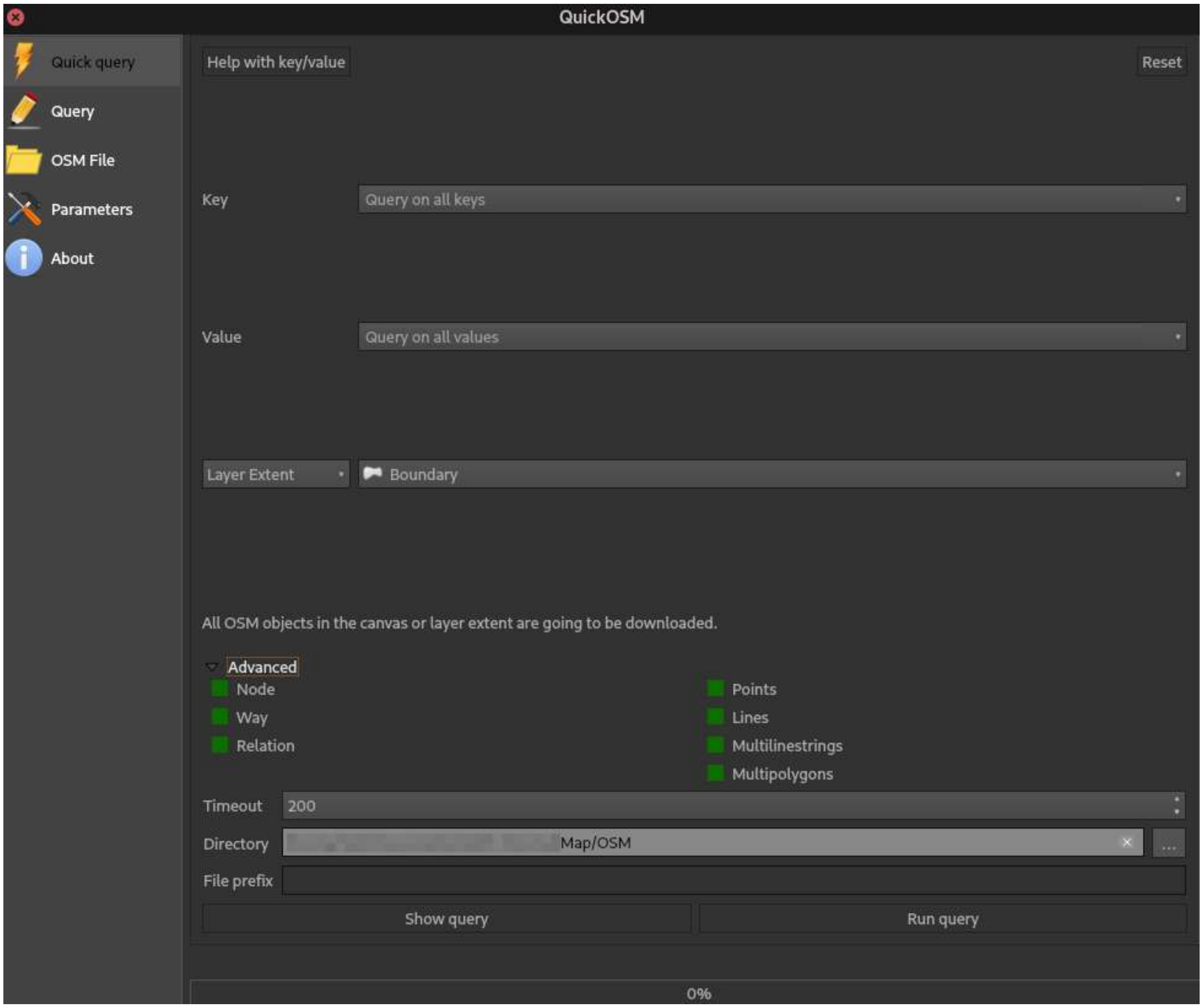
For those of us who like our data like we like our gear - free and unencumbered by someone else's grubby little hands - OpenStreetmap (OSM) provides a free and open-source set of data on virtually any geographic feature you might want, from roads, to rivers, to coastlines and political boundaries like country borders, county lines, and more.

QGIS comes with a plugin to quickly import OSM, thankfully, and this one doesn't require any sort of account. Go back to the plugins view where you downloaded SRTM Downloader, and this time search for `QuickOSM`. Install this plugin like before; once complete, you shouldn't see any visible change.

Now that QuickOSM is installed, we're able to query the OSM database for all the features in our area of interest.

Go to `Vector > QuickOSM > QuickOSM...`.

You can get extremely specific with what data you want to download, but for our purposes, we'll download everything and let QGIS handle formatting and filtering it. Better to have the data locally and not need it, then to need it and not have it, too. In the resulting dialog that pops up, stay on the "Quick Query" screen that you get by default.



Key: Query on all keys

Value: Query on all values

Layer Extent: Boundary

All OSM objects in the canvas or layer extent are going to be downloaded.

Advanced

☑ Node
☑ Way
☑ Relation
☑ Points
☑ Lines
☑ Multilinestrings
☑ Multipolygons

Timeout: 200

Directory: [redacted]Map/OSM

File prefix:

The only change you need to make is changing the third parameter from `In:` to `Layer Extent`, then selecting our boundary layer.

Hit `Advanced`, and in the file selection dialog, choose a folder where you want the data to be downloaded to. I typically change the timeout to 200, instead of the default 50, to prevent data from not being downloaded due to connection errors.

Hit `Run Query`, and the OSM data will begin downloading. Note that this is, again, likely to take quite some time, especially if you're working with a large area on a slow internet connection.

Once you get a message above the bottom progress bar indicating that loading is complete, you can close out of this window using the X at the top.

The end result is a *ton* of data, even for relatively small areas, and especially for heavily urbanized ones. For now, we'll just ensure all the resulting layers from our OSM import are dragged into a folder in the Layers Panel, and handle them one-by-one.

One thing that might be important to note is that QuickOSM downloads all this data as the text-based, human-readable GeoJSON format by default. While handy if you want to run some sort of processing on it with another program, or script your own processing, this might not be the best for speed and ease of use with other features in QGIS. Just like we saved the map imagery earlier, we can right-click on each of these layers and save them in a different format.

I personally use GeoPackage as an intermediate format before putting them all into a Spatialite database, but all the following steps will work the same no matter if we use a different format or keep the original GeoJSON. If you do decide to transfer the OSM data into a different format, make sure that at each step, you're using the correct project CRS, or the data may not even be visible on the map. Each step of saving will generally take a similar amount of time to the original download, but in the case of large data sets, it's a case of spending time up front to save time afterward.

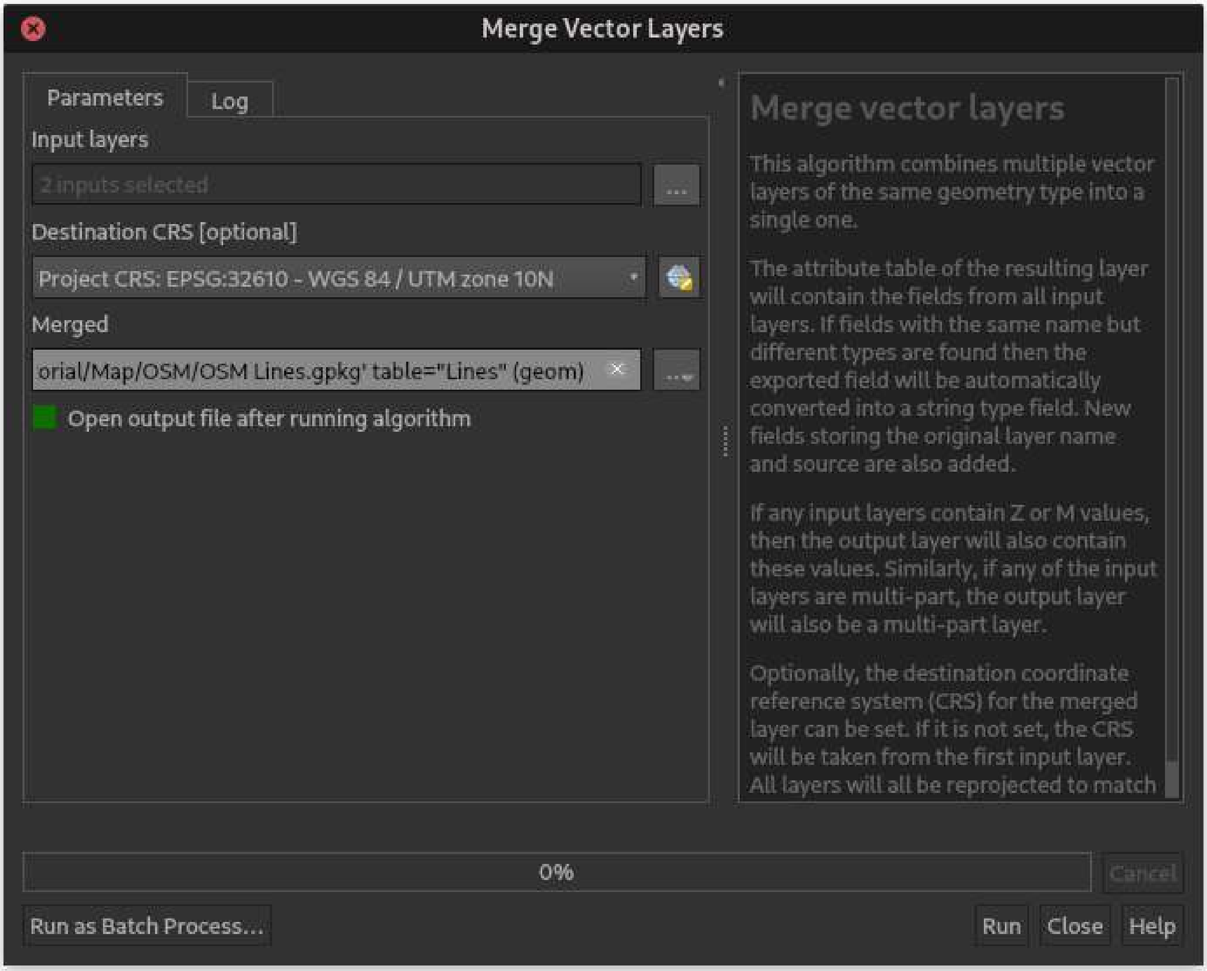
If needed - Merging layers and cleanup

The first step for me, typically, is handling all the lines from an OSM import. You might notice two different line layers; this is deliberate, as the way OSM handles data internally handles them as single lines, and multiline strings; for our purposes, we'll simply be combining the two together into a single layer for ease of handling.

(You may only have one line layer; in that case, skip this next bit.)

To merge two (or more) layers of the same type, go to `Vector > Data Management > Merge Vector Layers`. In the top dropdown, select the two layers you want to merge, then select the output file as usual, like the rest of the steps thus far. I personally use the `Save as GeoPackage` option, and choose something simple when prompted for a table name.

We might also need to reproject the layers into our project CRS. This is simple, and follows a similar process, using `Vector > Data Management > Reproject Layer`.



Input layers

2 inputs selected

Destination CRS [optional]

Project CRS: EPSG:32610 - WGS 84 / UTM zone 10N

Merged

orial/Map/OSM/OSM Lines.gpkg' table="Lines" (geom)

☑ Open output file after running algorithm

Merge vector layers

This algorithm combines multiple vector layers of the same geometry type into a single one.

The attribute table of the resulting layer will contain the fields from all input layers. If fields with the same name but different types are found then the exported field will be automatically converted into a string type field. New fields storing the original layer name and source are also added.

If any input layers contain Z or M values, then the output layer will also contain these values. Similarly, if any of the input layers are multi-part, the output layer will also be a multi-part layer.

Optionally, the destination coordinate reference system (CRS) for the merged layer can be set. If it is not set, the CRS will be taken from the first input layer. All layers will all be reprojected to match

Once you click `Run`, the progress bar will chug away. Again, this can take quite some time. The end result, however (if all goes well), will be one layer with all the data in one.

Note that you can't merge layers of completely different types, i.e. you can merge a point layer with another point layer, but you can't merge a point layer with a line layer.

Styling OSM road line layers

Now that we have one layer with all the line data for our map, it's time to use symbol styling to turn those layers into something readable.

Just like we did with the contour lines, go into the Layer Styling panel and change the symbol type to `Rule-based Symbols`, then double-click on the first (and only) rule yet to edit it.

I'll start with major roadways as my first set of lines, usually. The downside to using OSM data is readily apparent when it comes time to filtering it for symbols - the number of tags each OSM feature has is *vast*, and includes everything from restaurant hours of a particular polygon representing a building, to when a roadway was built, and so on. For our purposes, though, the basic `highway` tag is what we want. There are a lot of different road types OSM recognizes; in the import I'm using, there are:

> abandoned, construction, cycleway, disused, footway, living_street, motorway, motorway_link, path, pedestrian, platform, primary, primary_link, raceway, residential, road, secondary, secondary_link, service, steps, tertiary, tertiary_link, track, trunk, trunk_link, unclassified

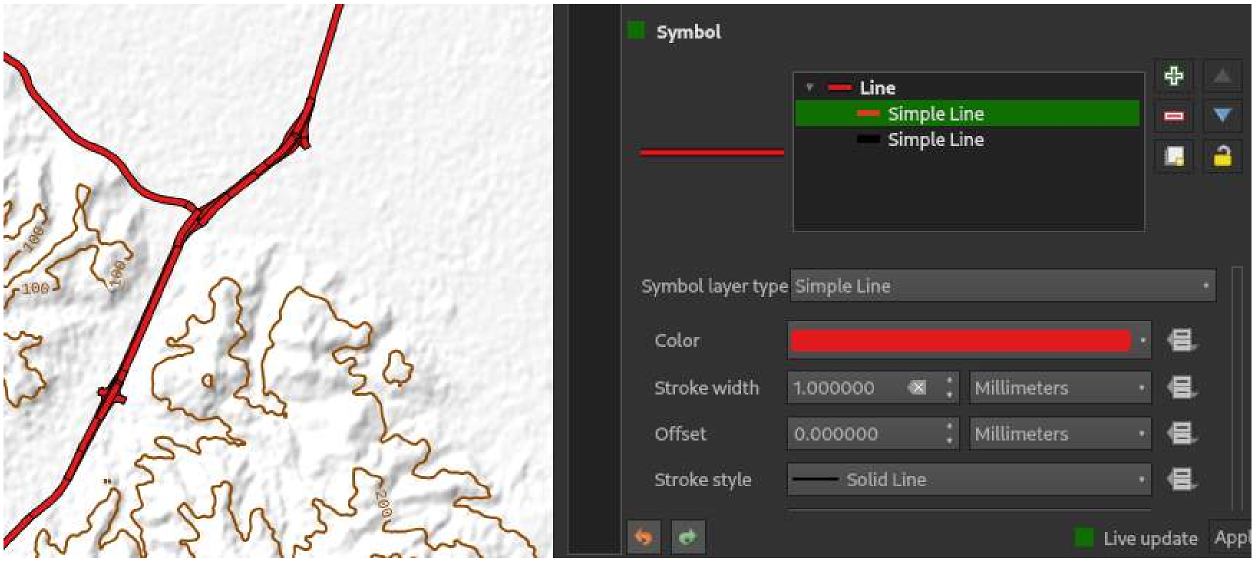
For the purposes of classifying the biggest, multi-lane roads/highways, I recommend using the following statement in the filter box:

```
"highway"='trunk' or "highway"='trunk_link' or "highway"='primary' or "highway"='primary_link'
or "highway" = 'motorway' or "highway" = 'motorway_link'
```

To add styling to it, first add a thick, black background line for the highway border. Change the default `Simple Line` to have a black color and a 1.5 mm thickness.

In order to quickly delineate which type of road we're dealing with when viewed on a map, let's add some color to the larger roads. Symbol layers work just like our actual map layers; to the right of the list of symbols where we see `Line`, and under it that `Simple Line` we just edited, click the green plus icon; another `Simple Line` will be added to the symbol. If needed, click the arrows to the far right until it's on top, indicating that it'll be drawn last (on top) per road. Change it to a red color and give it a 1 mm thickness.

The end result should look like the following.

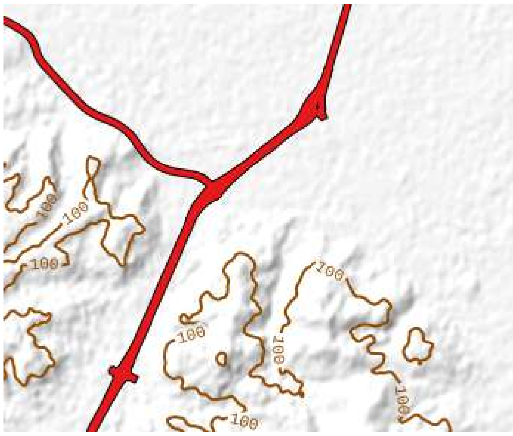


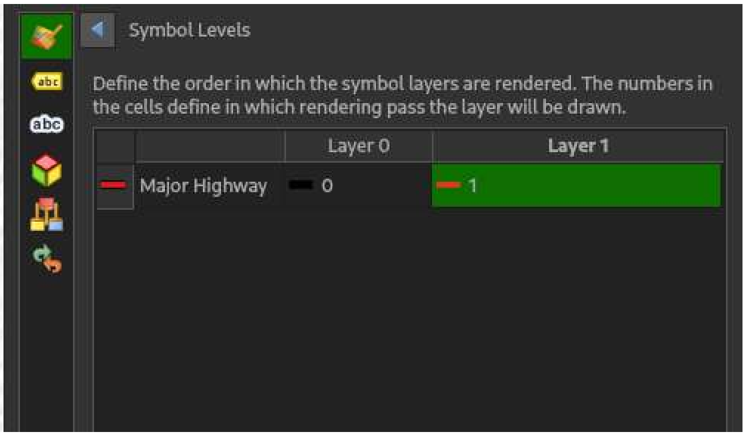
Symbol layer type: Simple Line
Color: 🟥
Stroke width: 1.000000 Millimeters
Offset: 0.000000 Millimeters
Stroke style: Solid Line

As you can see, the road segments look blocked-off by the red lines - the red line for *each* road is being drawn on top of the black line for itself, but not on top of the black line for *all* the road lines. This is intentional - QGIS first determines what should be drawn in order by the ordering of the layers in the Layer Panel, but for each layer itself, it draws each feature one after the other.

There's a way to fix this, though; back out of the symbol we created until we're in the overview of all the rule-based symbols.

To the lower right of the list, there's a box labeled `Symbol Levels`. Click that. The next screen shows us the order each symbol layer is drawn within the layer; right now, both the black background line and the red foreground line are at level 0, i.e. they'll be drawn per feature in the order of the features. Let's change the red foreground line to 1; higher numbers indicate that symbol layer should be drawn more on top. The result should look like the following; as you can see, the highways are now in a form that makes more sense to the viewer, especially once it's printed on a map.

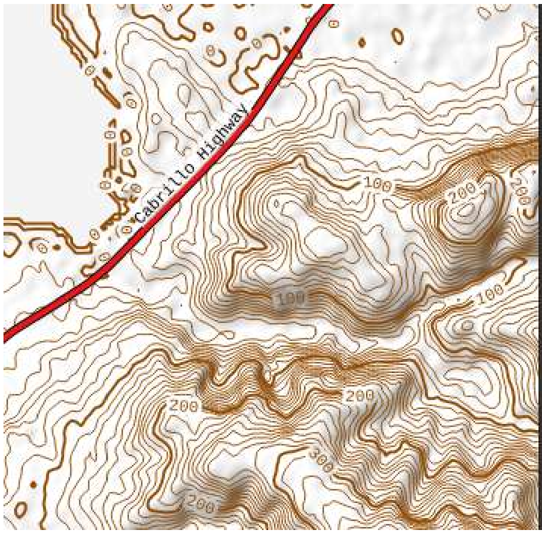


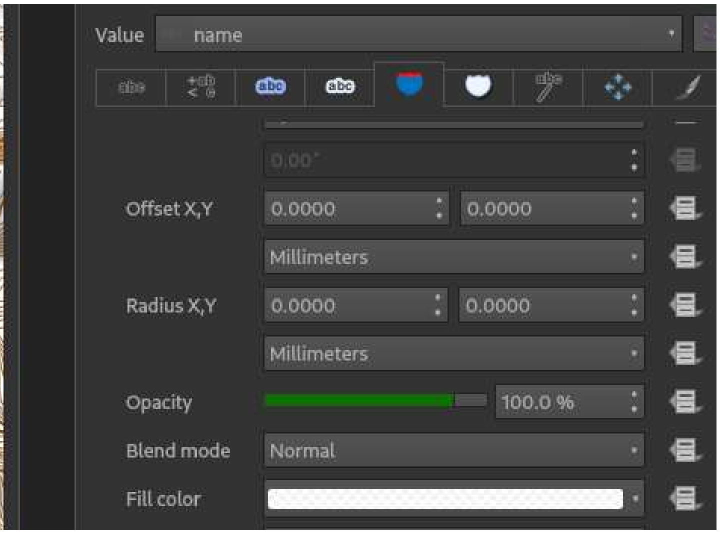

Symbol Levels

Define the order in which the symbol layers are rendered. The numbers in the cells define in which rendering pass the layer will be drawn.

	Layer 0	Layer 1
🟥 Major Highway	0	1

We can also add the names of the highways, if we want, and if it's recorded in the OSM import; to do this, we just add a rule-based label like we did with the index contours and use the `name` attribute.




Value: name

0.00°

Offset X,Y: 0.0000 0.0000

Millimeters

Radius X,Y: 0.0000 0.0000

Millimeters

Opacity: 100.0 %

Blend mode: Normal

Fill color

To add in secondary highways, I've chose to use the following filter, after copying and pasting the primary highway symbol like we did with the contour lines:

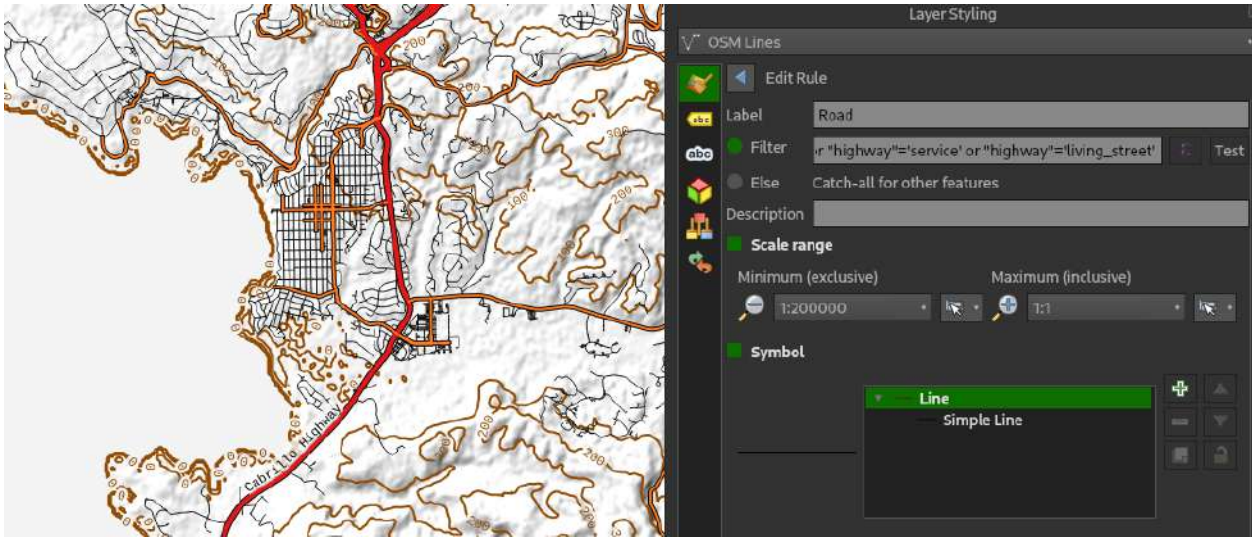
```
"highway"='secondary' or "highway"='secondary_link' or "highway"='tertiary' or "highway"='tertiary_link'
```

This one I've made an orange color, and changed the thicknesses of from 1.5/1 mm to 0.5/1 mm.

For basic roads, create a new rule, and set the filter to:

```
"highway"='road' or "highway"='residential' or "highway"='service' or "highway"='living_street'
```

Make this just a one-symbol-layer symbol by removing the colored foreground, if you copy and pasted; the road can be a 0.25 mm black line for our purposes. You may also want to add a scale-dependent visibility, as every residential street may not be important in a large-scale map; I set this to 1:200,000 to 1:1. We can add labels, with scale-dependent visibility themselves, to these as well as the secondary highways, too, if needed.



Layer Styling

OSM Lines

Edit Rule

Label: Road

● Filter: r "highway"='service' or "highway"='living_street' Test

○ Else: Catch-all for other features

Description

Scale range

Minimum (exclusive): 1:200000

Maximum (inclusive): 1:1

Symbol

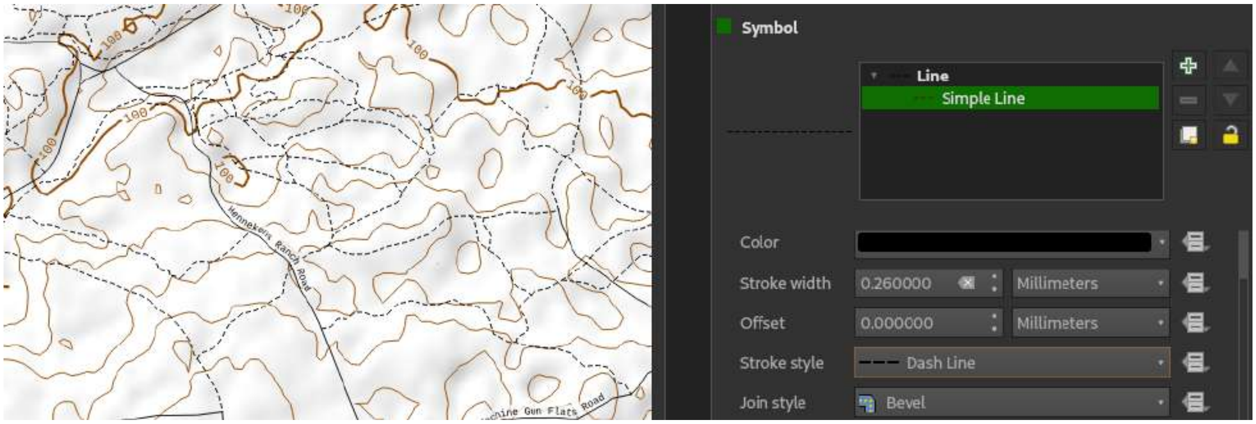
Line

Simple Line

To finish off our roadways, we add in trails and tracks. This can be a copy of the standard roads, with the filter:

```
"highway"='path' or "highway"='footway' or "highway"='pedestrian'  or "highway"='track' or
"highway"='unclassified' or "highway"='raceway' or "highway"='disused' or "highway"='abandoned'
```

Making this into the standard dashed line is fairly easy; under `Stroke style`, we can simply select `Dash line`.



Symbol

Line

Simple Line

Color

Stroke width: 0.260000 Millimeters

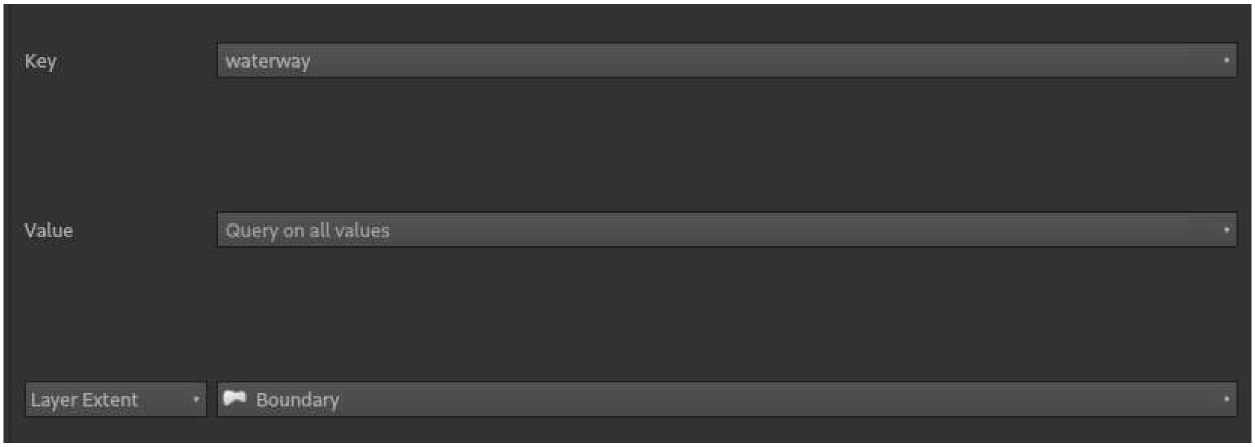
Offset: 0.000000 Millimeters

Stroke style: Dash Line

Join style: Bevel

OSM Waterways and filtering the OSM query itself

Waterways, like roads, are pretty simple to add in, and simply require another filter. However, I'm going to use this as an example on how to fetch only very specific data from the global OSM project. Go back to QuickOSM; instead of choosing the default "Query on all keys" in `Key`, type or select `waterway`. Leave the `Value` as "Query on all values;" we want to get any type of waterway offered by OSM. (You can probably guess by now that we could get a road-only layer earlier on by only selecting "highway" in `Key`, too.)



Key: waterway

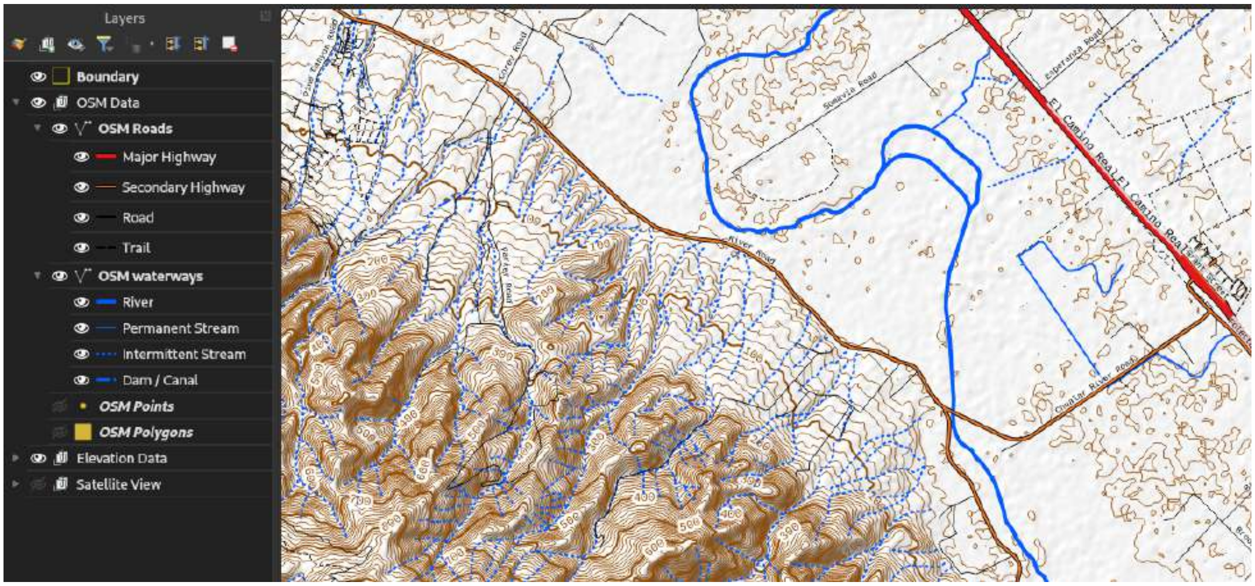
Value: Query on all values

Layer Extent: Boundary

After cleaning it up by reprojecting (Use `Vector > Data Management > Reproject`) and merging layers as needed (or getting rid of any point layers that were downloaded), we can then style the layer how we want. I won't go through this in excessive detail as it's almost identical to doing the roads; instead, below are some of the filters and styles I use:

- **River**:
  - Filter: `"waterway"='river' or "waterway"='tidal channel'`
  - Style: 1 mm solid line in color `#035bff`
- **Permanent stream**:
  - Filter: `"("waterway"='stream' or "waterway"='ditch') and "intermittent"<>'yes'`
  - Style: 0.5 mm solid line in color `#035bff`
- **Intermittent stream**:
  - Filter: `"waterway"='stream' and "intermittent"='yes'`
  - Style: 0.5 mm, `Dot Line`, in color `#035bff`
  - Visibility range: 1:75,000 to 1:1
- **Canals and dams**:
  - Filter: '`"waterway"='canal' or "waterway"='dam'`
  - Style: 0.75 mm, `Dash Hot Line` with end cap type `Round`, in color `#035bff`

The end result of this is depicted below.



Railways and power lines

Adding railways to the map is just like adding waterways, although I'll go through it as it highlights some features that might be helpful when it comes to doing some more advanced styling.

Download another set of OSM data using the `Key` of *railway*. Conduct reprojection and cleanup as needed, just like we did with the waterways.

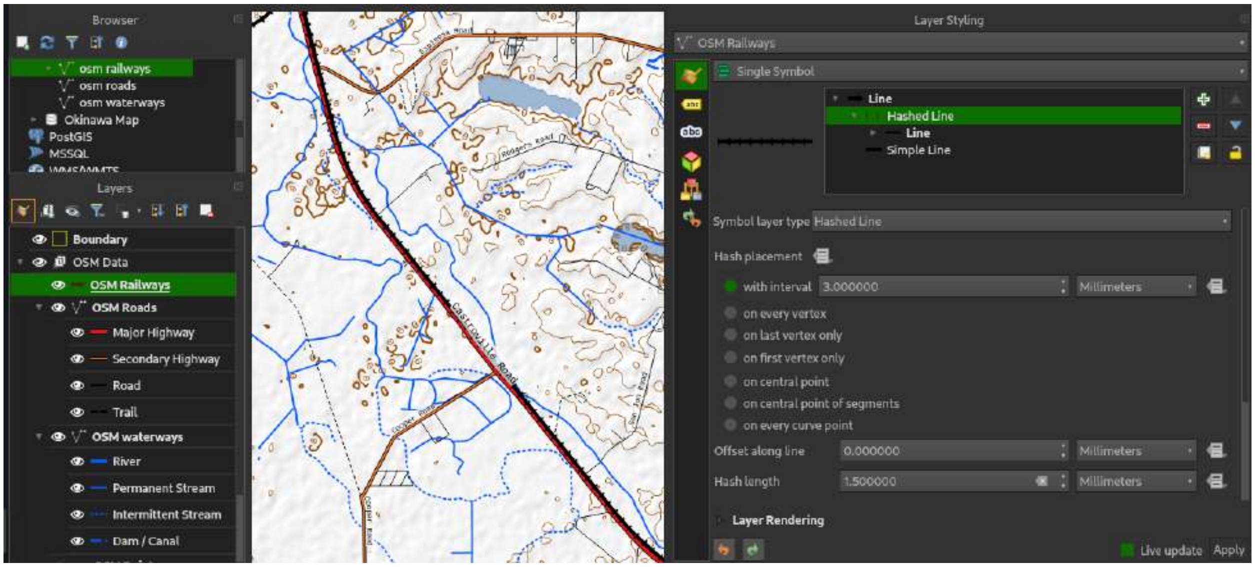
We'll start off by making the symbol a 1 mm black line. After that, however, let's go to the symbol layers and add a new line on top, just like we did with the major roads.

This time, however, pick `Hashed Line` for `Symbol layer type`, instead of `Simple Line`. (We could also accomplish this with `Marker Line`, which allows us to put point markers spaced out along a line, but a hashed line is the easiest way here.)

Another Line will appear in the symbol layers view; we simply need to style this line to form the cross-marks of our railway. I've chosen to make this a simple 0.5 mm black line.

Under the `Hashed Line` item itself, we can choose the interval we want for the hash marks; I leave it at the default 3 mm, and use 1.5 mm as the `Hash Length`, below, which controls how far the lines protrude to the sides.

The end result will give us a view similar to the below.



Symbol layer type: Hashed Line

Hash placement

● with interval 3.000000 Millimeters

○ on every vertex

○ on last vertex only

○ on first vertex only

○ on central point

○ on central point of segments

○ on every curve point

Offset along line 0.000000 Millimeters

Hash length 1.500000 Millimeters

We can add power lines using a QuickOSM query with the key of `power`; the process for adding them is the same as railways. However, this time we'll get both points and lines (polygons as well, but these can be deleted.) Note, though, that this may likely not be a complete listing of all significant power lines, depending on your area of interest.

Due to how the QuickOSM query works, we first do the usual merging nad reprojecting. However, we might also get a lot of points outside our area of interest. There's a fix for that.

Go to `Vector > Geoprocessing Tools > Clip`. The first box, `Input layer`, contains what you want to clip; the `Overlay layer` indicates the boundary you want to clip everything inside. Make the appropriate selection and choose a good output; this is the final data set.

For the lines, here are some of the filters and styles I recommend. Don't worry if they look like roads right now; we'll mask them out using power tower symbols.

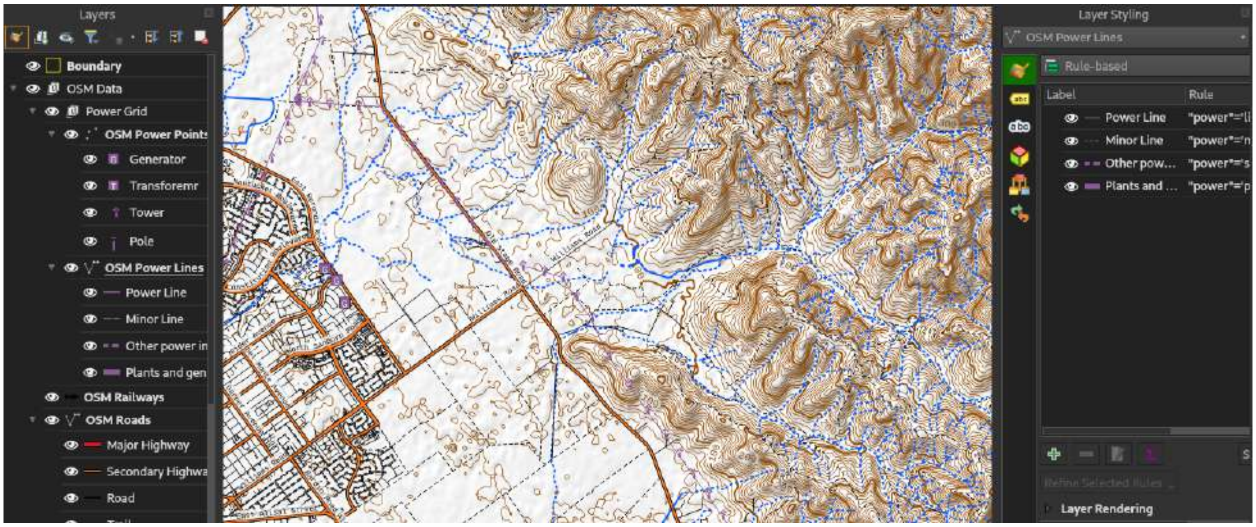
- **Power lines**
  - Filter: `"power"='line'`
  - Style: 0.5 mm solid line in `#8d5a99`

- **Minor power lines**
  - Filter: `"power"='minor_line'`
  - Style: 0.5 mm dashed line in `#8d5a99`
- **Misc. power infrastructure** (outline of facilities)
  - Filter: `"power"='substation' or "power"='transformer' or "power"='switchgear'`
  - Style: 1.0 mm dot line in `#8d5a99`
- **Plants and generator lines** (outline of facilities)
  - Filter: `"power"='plant' or "power"='generator'`
  - Style: 1.5 mm solid line in `#8d5a99`

For the towers and any other point layer, styling is very similar. Simply create a rule-based set of symbols for the point layer, using the following general styles. An exhaustive description of how to create a proper shape for each would be too much to fit in here, so I've provided some simple suggestions, and recommend you set a scale-dependent visibility on them. (I recommend 1:75,000 to 1:1.) Note that there are actual, doctrinal symbols for facilities on military maps; I suggest looking those up if this is for active-duty use.

- **Towers**:
  - Filter: `"power"='tower'`
  - Symbol: Use basic shapes to compose a tower-type shape; I suggest a circle on top of a triangle, both filled in using color `#8d5a99` from the lines. Or you can use `Font Marker` instead of `Simple Marker` for the `Symbol Layer Type`, with font family `D050000L` and character `0xfa`, at size 3 mm, rotated -90 degrees.
- **Poles**:
  - Filter `"power"='pole'`
  - Style: Font marker, with character `0x12b` and size 3 mm.
- **Generator**:
  - Filter: `"power"='generator'`
  - Style: Square simple marker with purple fill and hairline outline, with white text marker with a G in the center, font family set to what you prefer and scaled to fit the border.
- **Transformer**: Same as generator, replaced with "transformer" in the filter, and a T instead of a G in the symbol.

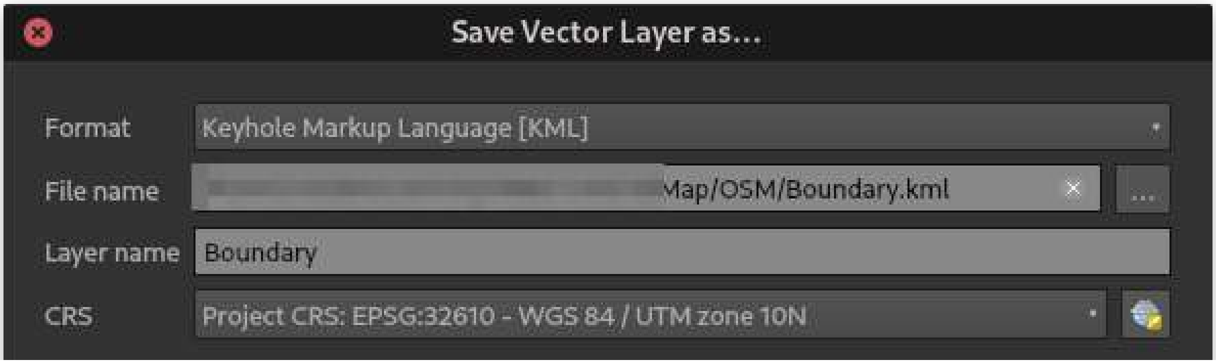
Once all is said and done, you should have power lines and towers plotted like the below image:



Water and vegetation

OpenStreetMap doesn't have *everything* we might be interested in as far as features go; notably, it doesn't contain large-scale polygons for things like bodies of water, oceans, and forest cover. For this, it's time to go back to our friends at NASA for up-to-date data.

First, let's turn our boundary layer into a format that the NASA request database can understand. Right-click on the boundary layer in the layers panel and choose `Export > Save Features As`. In the popup that comes next, we need the `Format` to be `Keyhole Markup Language [KML]`. Pick a good location to save under `File Name` and just call the `Layer name` "Boundary" or similar. Leave everything else as it is by default.



Format: Keyhole Markup Language [KML]

File name: [redacted]Map/OSM/Boundary.kml

Layer name: Boundary

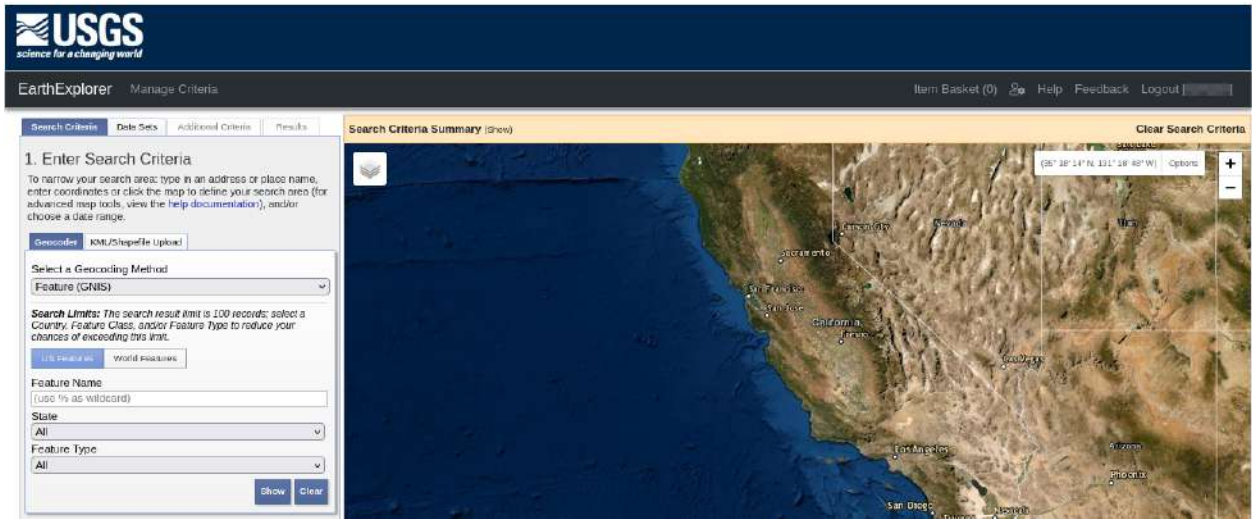
CRS: Project CRS: EPSG:32610 - WGS 84 / UTM zone 10N

Water

Once that's done, let's go to the actual site that SRTM Downloader is pulling its data from, more or less - the USGS's Earth Explorer portal at https://earthexplorer.usgs.gov/.

Create an account if you haven't already. The amount of data this portal exposes for the end user is unbelievably vast, but for our purposes, we'll be using it to download recent (within 30 days) data to populate our map with bodies of water and vegetation.

Once you're on the main portal, which should look something like the following image, it's time to let the system know what our area of interest is.



Logout | [redacted]

1. Enter Search Criteria

To narrow your search area: type in an address or place name, enter coordinates or click the map to define your search area (for advanced map tools, view the help documentation), and/or choose a date range.

Select a Geocoding Method: Feature (GNIS)

Search Limits: The search result limit is 100 records; select a Country, Feature Class, and/or Feature Type to reduce your chances of exceeding this limit.

Feature Name: (use % as wildcard)

State: All

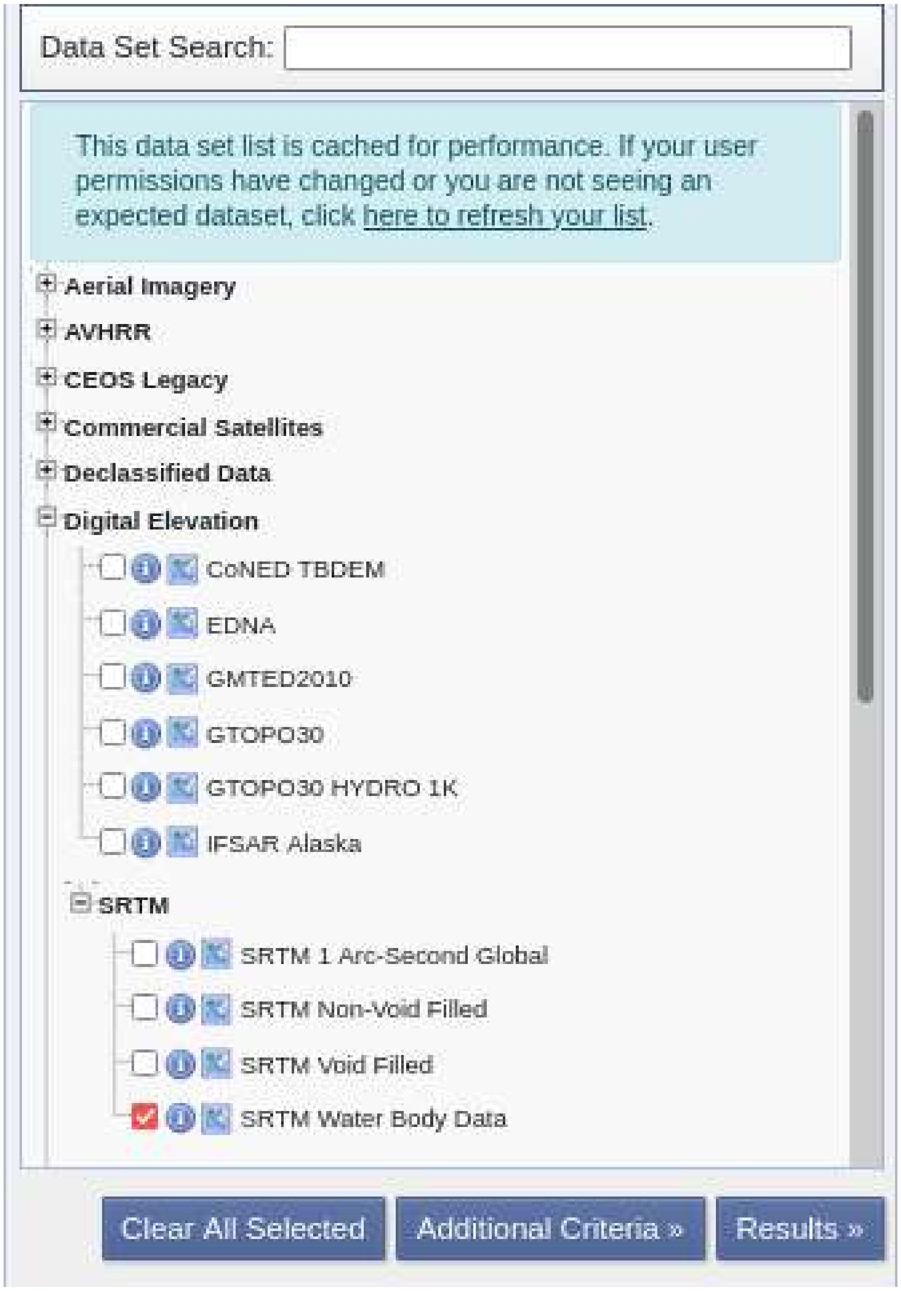
Feature Type: All

As you can probably guess, we want the `KML/Shapefile Upload` tab there in the middle. Click that, and then make sure `KML/KMZ` is selected in the dropdown. Once that's clear, hit `Select File` and pick the boundary file you just saved earlier. A loading bar will appear; unless you get an error message, you're good to click `Close` when it appears. The map should populate with a depiction of our boundary file, and coordinates will appear in the left-hand bar right under where we uploaded the file.

We've successfully told the system where we care about. The next step is to scroll all the way to the bottom and pick `Data Sets >>` in order to request the actual data we're interested in.

Every data set available, nested in folders, will appear on the left. We only need the SRTM Water Body

Data; in the search filter, then, type `SRTM Water Body Data` and select it when it appears in the dropdown. You should see a folder open up below with the appropriate data set selected.

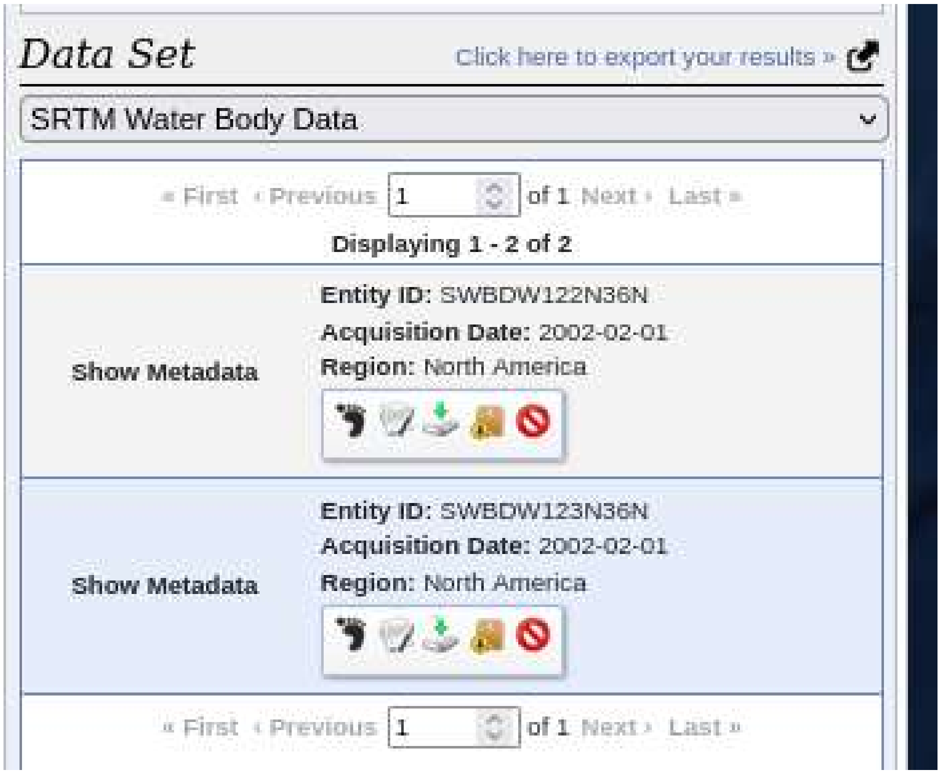


Data Set Search:

This data set list is cached for performance. If your user permissions have changed or you are not seeing an expected dataset, click here to refresh your list.

- ⊞ Aerial Imagery
- ⊞ AVHRR
- ⊞ CEOS Legacy
- ⊞ Commercial Satellites
- ⊞ Declassified Data
- ⊟ Digital Elevation
  - □ CoNED TBDEM
  - □ EDNA
  - □ GMTED2010
  - □ GTOPO30
  - □ GTOPO30 HYDRO 1K
  - □ IFSAR Alaska
  - ⊟ SRTM
    - □ SRTM 1 Arc-Second Global
    - □ SRTM Non-Void Filled
    - □ SRTM Void Filled
    - ☑ SRTM Water Body Data

Once that's done, click `Results >>` at the bottom. Depending on the size of the region you're interested in, multiple files might appear. There's a way to bulk download all at once, but unless you're trying to assemble a map of a continent-sized region, it's easy enough to download all of them individually.


Click here to export your results »
« First ‹ Previous 1 of 1 Next › Last »
Show Metadata
Show Metadata
« First ‹ Previous 1 of 1 Next › Last »

Data Set

SRTM Water Body Data

Displaying 1 - 2 of 2

Entity ID: SWBDW122N36N
Acquisition Date: 2002-02-01
Region: North America

Entity ID: SWBDW123N36N
Acquisition Date: 2002-02-01
Region: North America

You should see results looking like the above; the button in the middle with the save icon is what we want. For each of them, hit the button and save it into a folder with the rest of your map project. If it prompts you for a format, just pick `Standard Format Water Body`

Once that's complete, go into the folder and extract the zip files. You can get rid of the Documents folder in each.

Go back to QGIS. For each file you downloaded, go to `Layer > Add Layer > Add Vector Layer` and leave every option as default. You should see new layers populate in your Layers panel, but nothing on the map. That's intentional; these don't have an embedded CRS with them. To fix that, right-click on the layer and choose `Properties`. Under `Assigned Coordinate Reference System`, pick `EPSG:4326 - WGS-84`. If this isn't available, click on the globe icon to the right of the text box and type `4326` in the search bar; the only one that comes up should be the right one. Click `OK`.

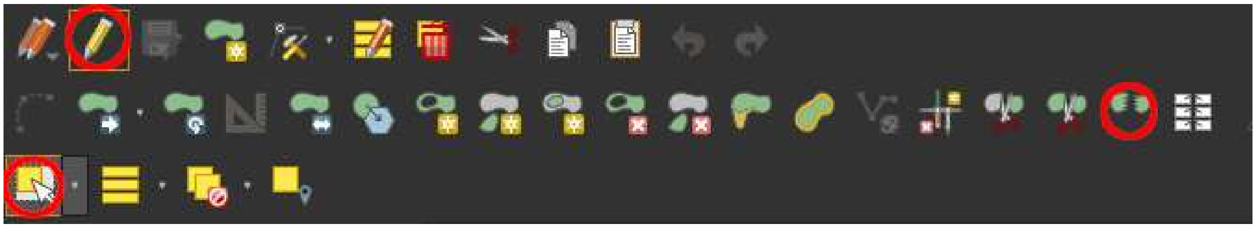
There should now be a large, multicolored polygon patchwork with every body of water greater than about 30 m across represented, and most likely much larger than your area of interest. First, we reproject the layers into our project CRS (remember, it's `Vector > Data Management Tools > Reproject Layer`) into our project CRS, deleting the old files as we go.

Now, it's time to merge the layers together. Go to `Vector > Data Management Tools > Merge Vector Layers`. In the top box, select our imported water layers; in the middle, make sure the project CRS is selected, and in the last one, you can leave it blank as a temporary layer. Click `OK`; the merged layer will now appear on the map.

At this point, we should have a single layer with all our water data, including some areas significantly bigger than the area of interest. Again, we can clip this. Go to `Vector > Geoprocessing Tools > Clip` like before. Select the water layer in the top as the input, the boundary layer as the overlay layer, and pick a final location to save the file. Hit `Run`; the end result will be a new layer with just the water in our area of interest.

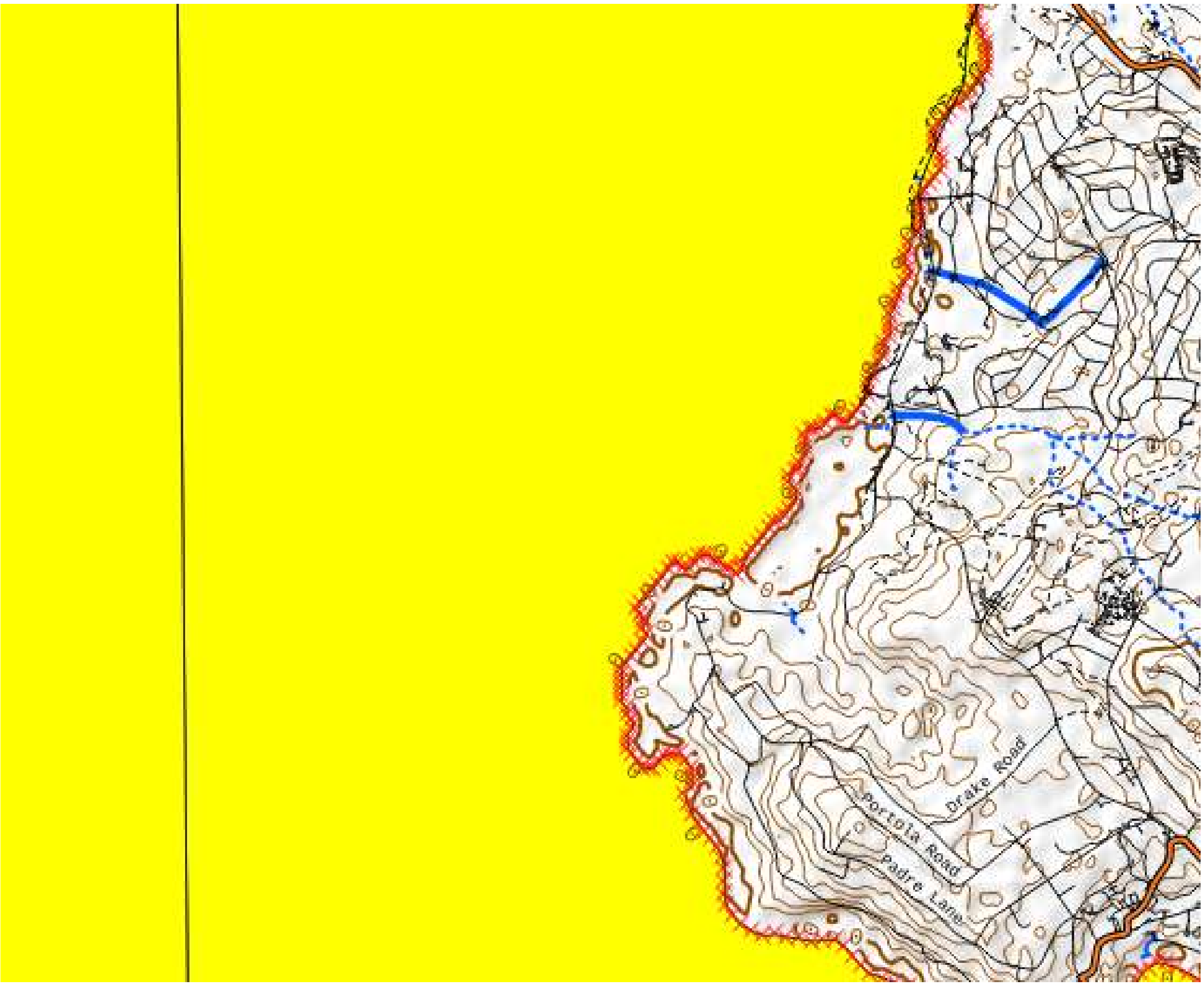
If you have an exceptionally large body of water, like an ocean, that spanned multiple input layers, you might notice there are different, roughly rectangular, lines in it. This is normal; the body of water is actually represented by multiple shapes within the data file. I recommend keeping this so that if you zoom in, hundreds or even thousands of data points aren’t loaded into memory.

For the sake of demonstration, though, we’ll go ahead and merge these together. Under `View > Toolbars`, make sure the ‘`Digitizing`, `Selection`, and `Advanced Digitizing` toolbars are enabled, then make sure our water layer is selected. Hit the yellow pencil icon (`Toggle Editing`) in the digitizing toolbar to start editing the layer, the topmost circled item in the below screenshot.



Click on the `Select Feature by Area or Single Click` tool in the Selection toolbar, the leftmost icon in the toolbar and the bottom-most circled item in the above screenshot.

Now, simply click on one of the pieces of the body of water, then hold down shift and click the rest in turn. They’ll turn yellow as they’re selected, with the outermost outline points turning into red X’s.

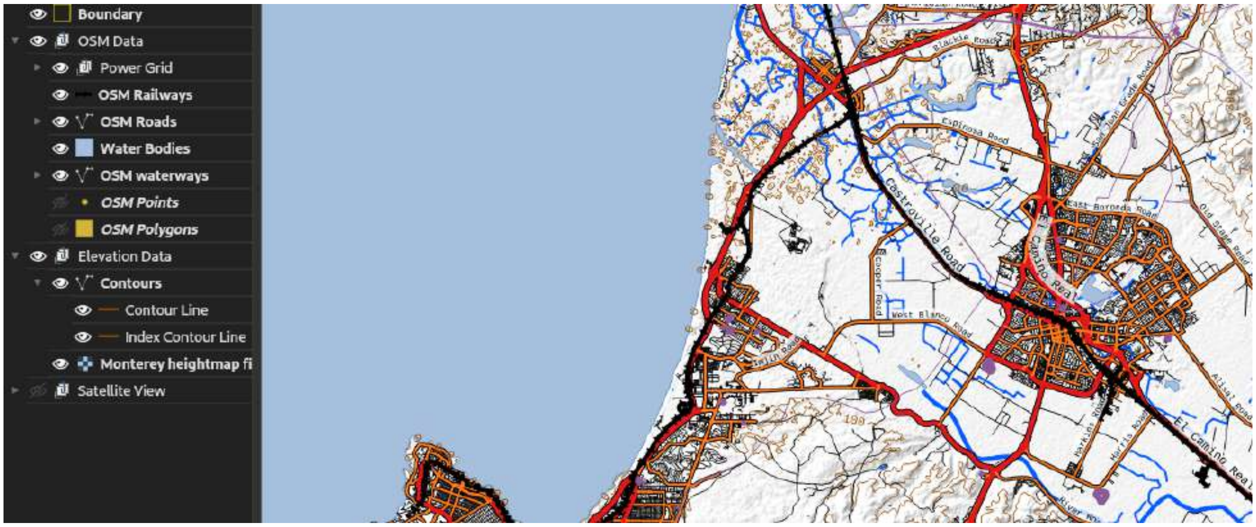


Now hit `Merge Selected Features`, the rightmost circled item from the toolbar image, then hit `OK` on the resultant toolbar. The end result is one shape with the entire body of water you wanted to merge into one. Click on the save icon immediately to the right of the toggle editing button, then hit the toggle editing button again.

Open up the Layer Styling panel and apply a style; no need for rule-based styling for most purposes here. I recommend using the Simple Fill type, with a fill color of `#a6cee3` and an 0.25 mm outline of color `#6498d2`.

I recommend dragging this layer above your waterways layer and below your roads layer, to prevent rivers seemingly cutting through lakes and bridges from being appropriately depicted.

Note that because this is from NASA's satellite data, these bodies of water won't be labeled like the OSM data is. However, with the addition of water, coastal maps are finally looking like a recognizable form that one can already navigate off of roughly.



Vegetation

Now for the most complicated part of this entire process, classifying vegetation and putting it on the map in a human-readable format. Obviously, you can simply trace all the trees, woods, etc. in your area of interest by hand, and put that into QGIS. This is going to be the most accurate, and is viable for smaller areas; however, for large-scale maps, it's almost entirely a non-starter.

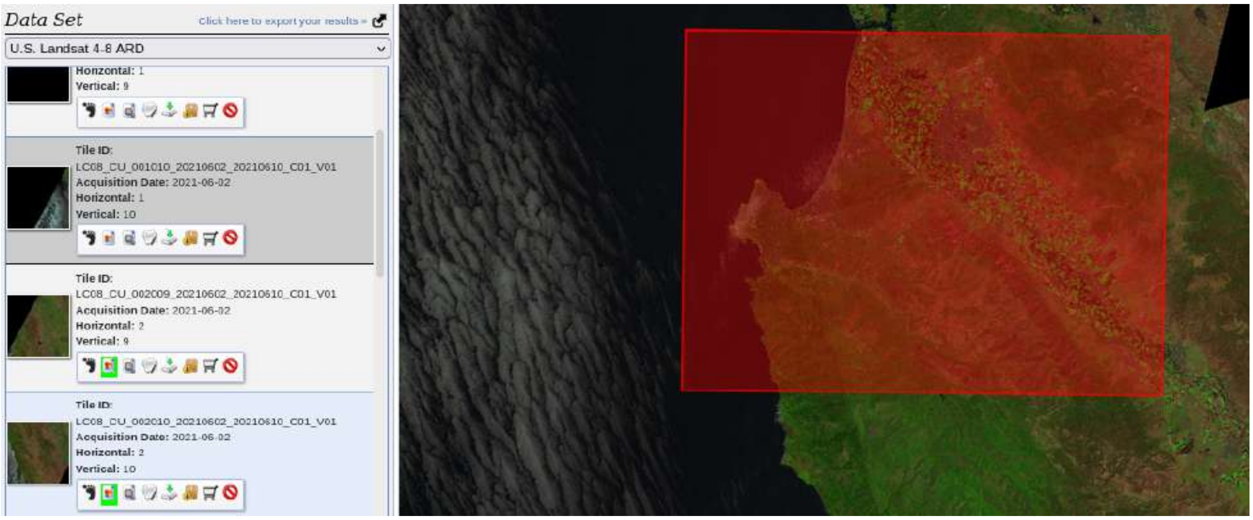
To achieve this, we're going to pull imagery from NASA again, just like we did with the water, and use something called a normalized difference vegetation index (NDVI). Essentially, we're going to use the difference between the red spectrum (fun fact: trees aren't red), and the IR spectrum, captured by a satellite to allow the computer to make an educated guess on the vegetation coverage in an area. It's important to understand that this isn't perfect, and the time the data you're using, farmland in the area, cloud cover, etc. will also affect the result pretty significantly. However, it's a good starting point for doing manual cleanup of the data, and can be pretty helpful for large areas where general trends are important. Remember, vegetation grows and changes; use it as a planning factor for what you're wandering into, *not* for navigating precisely based off of the edges of forests, clumps of trees, and such, especially if you haven't physically verified these features yourself beforehand.

Note that throughout this, until the end, I'll be hiding all the roads, contour lines, water, etc. we've already done in the interest of clarity.

Downloading vegetation index data Let's go back to the Earth Explorer site. Just like before, we set our boundaries of the area of interest. This time, however, we're looking for data from the LANDSAT project, specifically the LANDSAT Analysis-Ready Data (ARD) set. Put `ARD` into the search bar when it comes to data sets; we're looking for the `U.S. Landsat 4-8 ARD` set.

Depending on the size of your area, you might have to download several images. This is normal; however, you might need to check around the first few results pages to find a combination that covers your entire area.

Be warned that cloud cover, incomplete data, etc. may be present; it's a good idea to look at each result item's preview, using the second button from the left, which looks like a document with a generic image on it, on each result, to preview the coverage, as seen below. I highly recommend looking through what's available, as you may not be able to get something



Horizontal: 1
Vertical: 9

Tile ID:
LC08_CU_001010_20210602_20210610_C01_V01
Acquisition Date: 2021-06-02
Horizontal: 1
Vertical: 10

Tile ID:
LC08_CU_002009_20210602_20210610_C01_V01
Acquisition Date: 2021-06-02
Horizontal: 2
Vertical: 9

Tile ID:
LC08_CU_002010_20210602_20210610_C01_V01
Acquisition Date: 2021-06-02
Horizontal: 2
Vertical: 10

While previously, we downloaded each image at once, I'll go through how to bulk download for illustration's sake. Click on the package icon on the right side of each result, to the immediate right of the download button we used previously. Once all the images you want to download are selected, hit `View Item Basket` at the bottom.

You'll be taken to a screen that says `Bulk Download - X Scenes`, where X is the number of images you want to download. Hit `Start Order`. Give it a name when prompted, like `[Area name] LANDSAT`, for your reference, and choose the `Bulk Download` option on the left. Now hit the arrow button next to the `U.S. Landsat 4-8 ARD` text.

We need to pick specifically what we want - for our purposes, we need the `Surface Reflectance` data (which should be the largest file in each set). Pick that for each image you want to download, then head back to the bottom and hit `Submit Product Selections`. It'll indicate the order has been submitted; the email you set up when creating an account will receive something fairly soon. In the mean time, you can download the Java bulk downloader app for your OS on the provided link (https://dds.cr.usgs.gov/bulk).

Once you've gotten the email indicating your order is ready (in most cases, it'll be ready even before the email is sent out), and you've installed the bulk downloader app, open up the bulk downloader using the same username and password as on the site. Once logged in, you should see the order with the name you picked, ready to be downloaded. Select it and click the button to proceed. The end result should look something like the below.

Pick a good folder to download the imagery to in `Settings > Destination Directory Settings`, then hit `Begin Download`. Progress bars will appear.

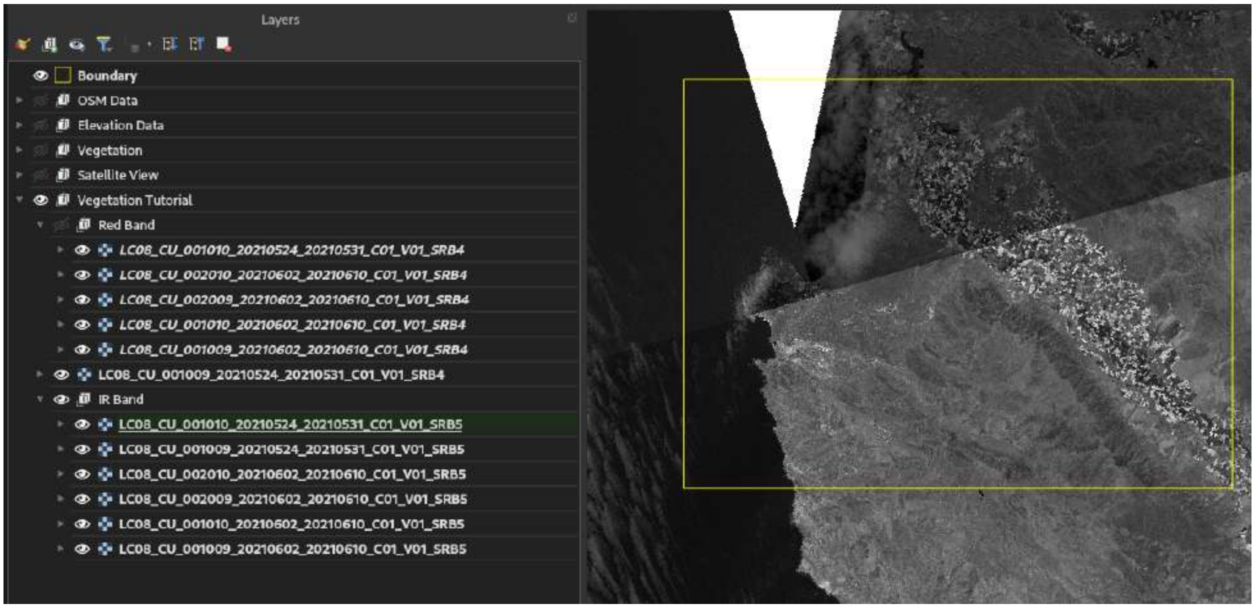
Once all your data is downloaded, it's time to find the specific data we need. LANDSAT downloads multiple bands (wavelengths) of data. For our purposes, we'll need the red channel (band 4), and the infrared channel (band 5). Unzip each folder that was downloaded in the location you picked. For each folder, you'll need the following files:

- The one ending in `_SRB4.tif`. This is the red channel of the LANDSAT data.
- The one ending in `_SRB5.tif`. This is the IR channel of the LANDSAT data.

Don't worry if they look all black or all white in the thumbnail; this is intentional, as each image is composed of data per pixel, essentially, instead of colors like we might be used to with any of other types of images you might happen to download from the internet.

Let's start on the red band images as an example. Drag each of those onto your canvas in QGIS, or use `Layer > Add Layer > Add Raster Layer`.

I've grouped them (and added two more images once I realized part of my area of interest wasn't included in the original download) for clarity; the end result should look similar to the below. Don't worry if the images look slightly darker/brighter than each other; this is normal due to the way QGIS automatically assigns each data raster a grayscale value. If needed, rearrange the way they sit within each band for complete coverage and ensure there aren't any gaps in important locations. The end result of this step will look something like the below image.


Layers
Boundary
OSM Data
Elevation Data
Vegetation
Satellite View
Vegetation Tutorial
Red Band
LC08_CU_001010_20210524_20210531_C01_V01_SRB4
LC08_CU_002010_20210602_20210610_C01_V01_SRB4
LC08_CU_002009_20210602_20210610_C01_V01_SRB4
LC08_CU_001010_20210602_20210610_C01_V01_SRB4
LC08_CU_001009_20210602_20210610_C01_V01_SRB4
LC08_CU_001009_20210524_20210531_C01_V01_SRB4
IR Band
LC08_CU_001010_20210524_20210531_C01_V01_SRB5
LC08_CU_001009_20210524_20210531_C01_V01_SRB5
LC08_CU_002010_20210602_20210610_C01_V01_SRB5
LC08_CU_002009_20210602_20210610_C01_V01_SRB5
LC08_CU_001010_20210602_20210610_C01_V01_SRB5
LC08_CU_001009_20210602_20210610_C01_V01_SRB5

At this point, I'll merge each of the images within each band (all the red images go into one image, all the IR ones into another) for easier processing using `Raster > Miscellaneous > Merge`, leaving the output type as `Float32`. After cleaning up the individual layers, we now have two images; I'll name them `RedBand` and `IRBand`, respectively.

Now it's time to do the NDVI calculation; in math terms, this is:

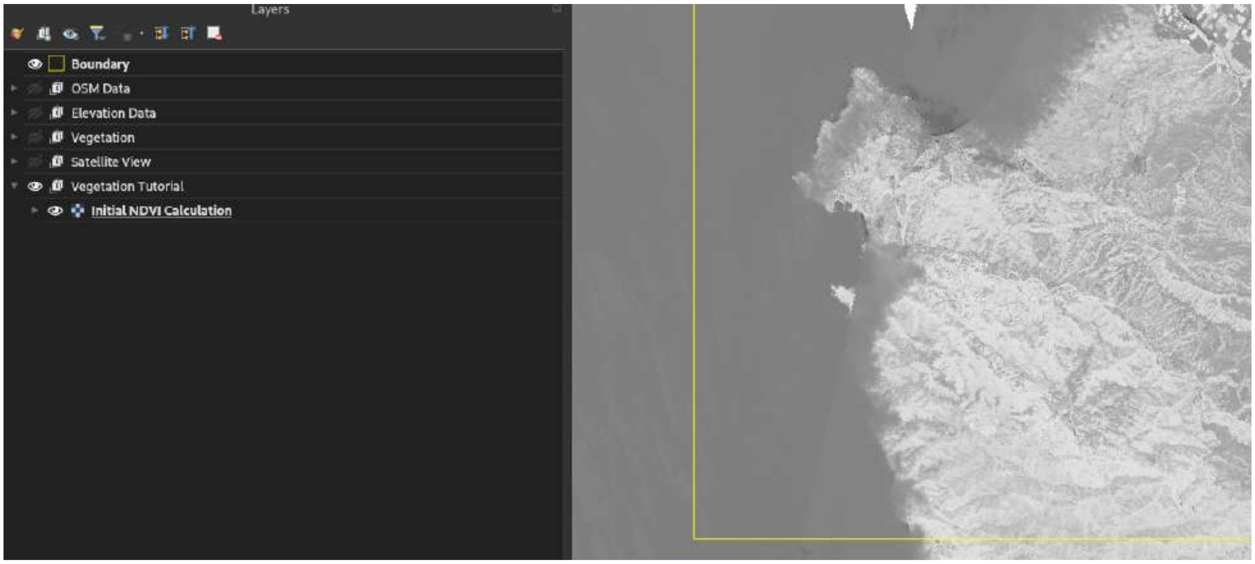
```
(IR - Red) / (IR + Red)
```

There's an easy way to do this, thankfully. Go to `Raster > Raster Calculator`. You should see a list of layers on the left with `@1` (for the first channel) after them, and a `Raster Calculator Expression` box at the bottom. The appropriate formula for that is:

```
("IRBand@1" - "RedBand@1") / ("IRBand@1" + "RedBand@1")
```

Leave everything else as default, and pick a good save location. We haven't reprojected it into our project CRS yet, so leave the CRS as is (likely `Albers`). Hit `OK`, and the calculation will chug away.

The result at this stage should look similar to the below.



At this point, it's already clear that, except for clouds over water, the whiter areas have more vegetation, when we compare this to our satellite map. We're on the right track. Just like before, let's reproject the image to our project CRS, now, with `Raster > Projections > Warp (Reproject)` and clip the image to our boundary with `Raster > Extractions > Clip by Extent`. (I personally use 0.0 for the `nodata` value when prompted for the reprojection, to handle areas of the image that didn't have any data initially, too).

Processing the vegetation data Now it's time to actually turn this value into usable vegetation indices. This part is more art than science, but we'll need to use the advanced processing tools to do it. Open up the processing toolbox panel with `Processing > Toolbox`, then go into the items on the right and enter `classify` in the search bar at the top. The item we're looking for is `Reclassify by table`, which will allow us to sort the image into, roughly, light/medium/heavy vegetation to add to our topographic map.

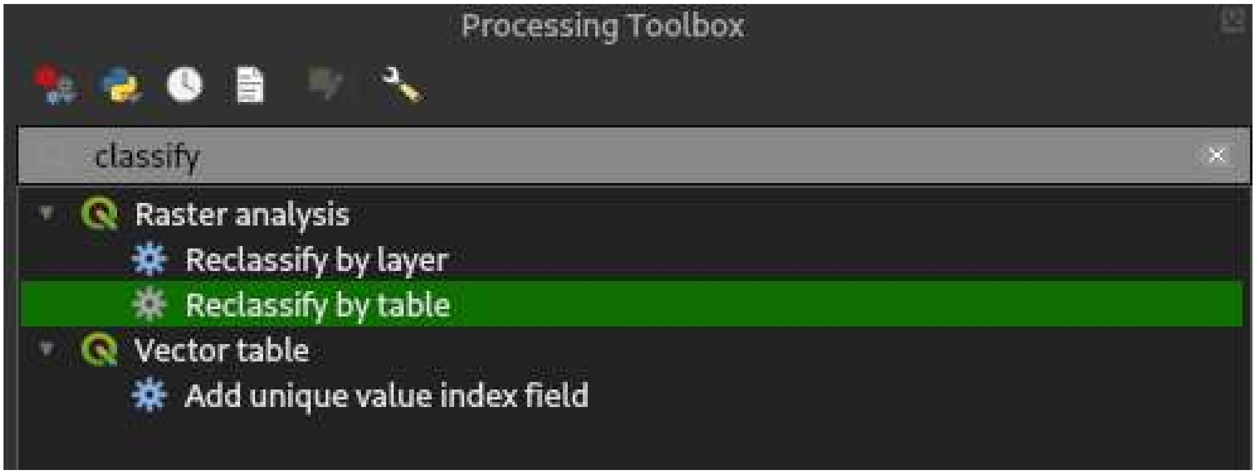
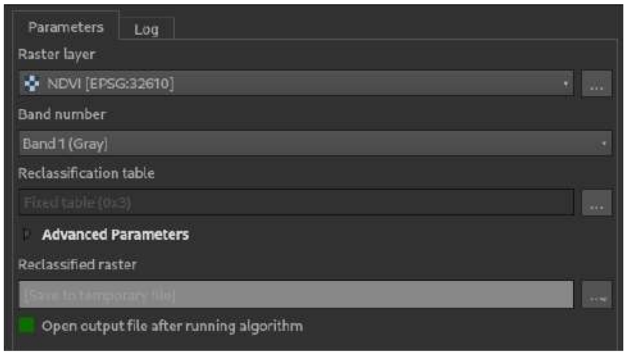


Figure 1: Classifying by table

Double-click on `Reclassify by table`. Our input layer should be the final clipped NDVI calculation we did, and the band number is band 1, the only band that should be in this grayscale image.

We need to create a table of values, now; click on the three dots to the right of `Fixed table (0x3)`. We have a minimum value for each classification, a maximum, and what to set each value in that band to. I typically use the following values, clicking `Add Row` to add each row in turn, although you'll need to play around to get the light-or-no/medium/heavy vegetation values to line up with common sense about your area of interest:

Minimum	Maximum	Value	(Notes)
-1.0	0.25	0.0	None/Light
0.25	0.6	0.5	Medium
0.6	1.0	1.0	Heavy



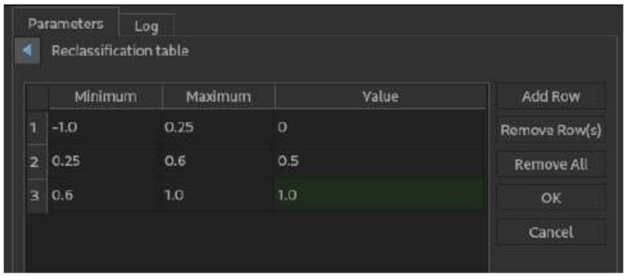
Raster layer: NDVI [EPSG:32610]

Band number: Band 1 (Gray)

Reclassification table: Fixed table (0x3)

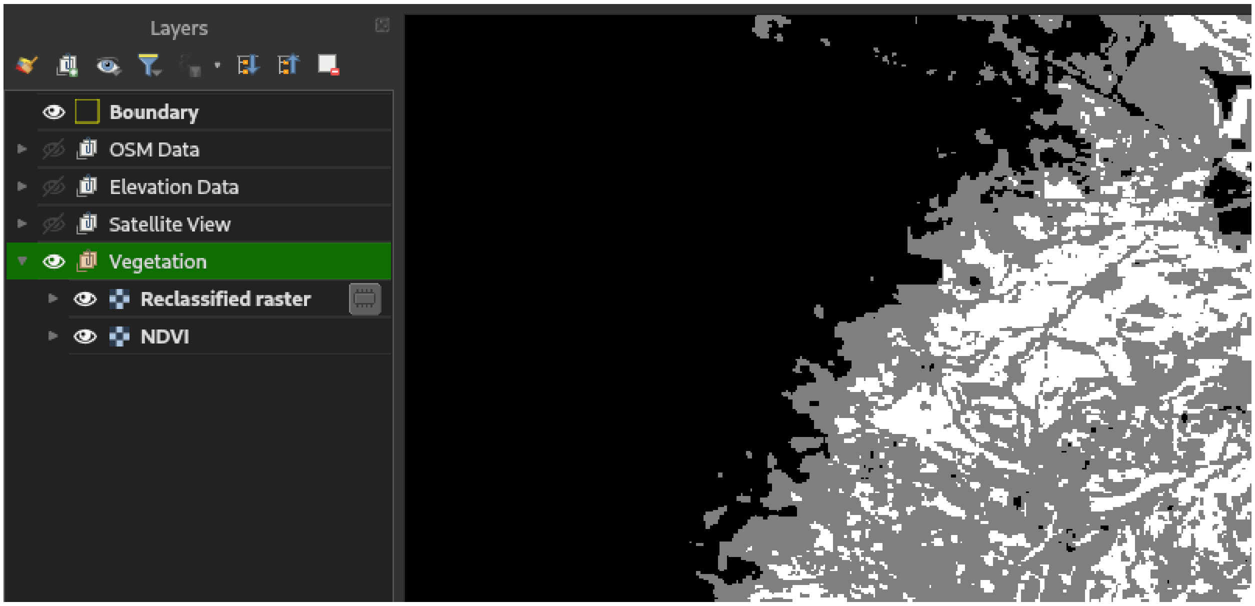
Reclassified raster: [Save to temporary file]

☑ Open output file after running algorithm



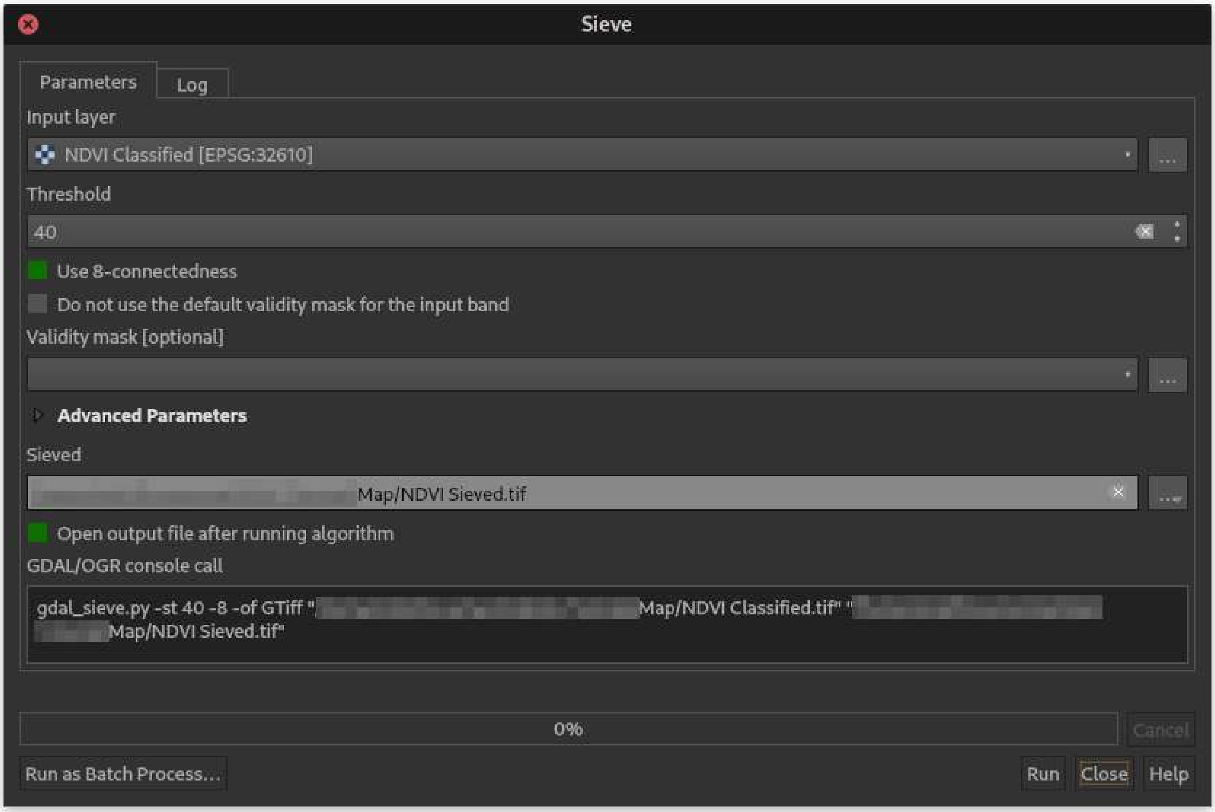
	Minimum	Maximum	Value
1	-1.0	0.25	0
2	0.25	0.6	0.5
3	0.6	1.0	1.0

The end result should look much more useful as far as classifying vegetation for a topographic map, looking something like the below.



However, we have all sorts of speckles and outliers here; the map will be cluttered with details that will change from season to season if we leave these in, and it might be best to simply remove them. To do this, we first need to save our one-band classified NDVI data as a single image, in anticipation of letting QGIS's Sieve filter do its thing. Right click on the layer and pick `Export > Save As`, and pick a good file location for the classified data. Crucially, at the very top, we need to pick `Rendered image`, not `Raw Data`, now. Make sure we still have the appropriate CRS, and click `OK`. You might see the new layer pop up with red, green, and blue bands; this is normal and expected.

Time to sieve the data, i.e. remove the outliers and make it a more homogeneous set of data to prepare for turning it into vector data. Go to `Raster > Analysis > Sieve`. This filter works by removing any splotches of color less than the `Threshold` value in size, in pixels. You may need to play around with this value, depending on the detail you want and the size of your area of interest. I've displayed the settings I'm using below.



Input layer

NDVI Classified [EPSG:32610]

Threshold

40

☑ Use 8-connectedness

□ Do not use the default validity mask for the input band

Validity mask [optional]

Sieved

[redacted]Map/NDVI Sieved.tif

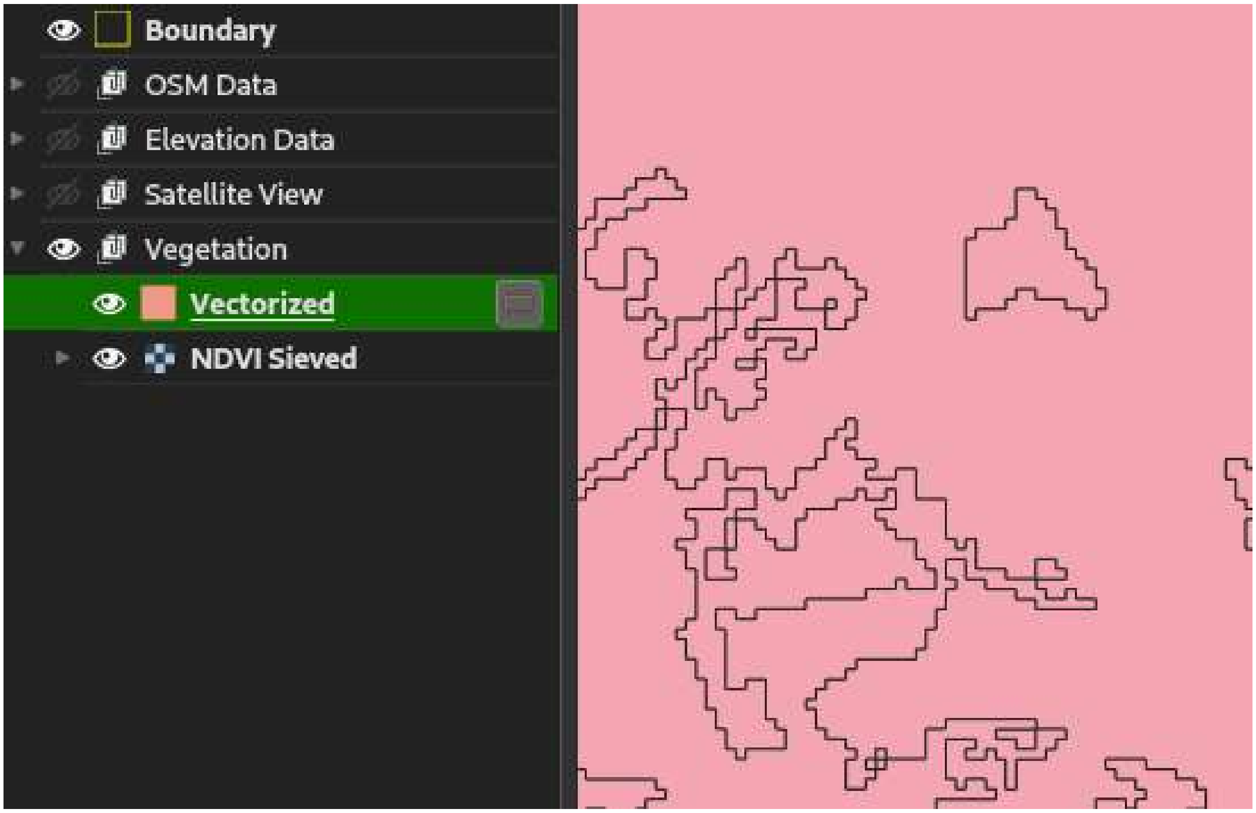
☑ Open output file after running algorithm

GDAL/OGR console call

```
gdal_sieve.py -st 40 -8 -of GTiff "[redacted]Map/NDVI Classified.tif" "[redacted]
Map/NDVI Sieved.tif"
```

The result should be a lot cleaner looking; it's time to turn this raster data into usable polygons on our map. The appropriate function is, logically enough, found at `Raster > Conversion > Polygonize (Raster to Vector)`. Keep all the values at default, but under "Name of the field to create," just use "NDVI" for clarity. I generally pick `Use 8-connectedness`, which is slower, but provides a better result, in my opinion. You can leave it as a temporary layer, as we'll be doing some additional processing to smooth it out.

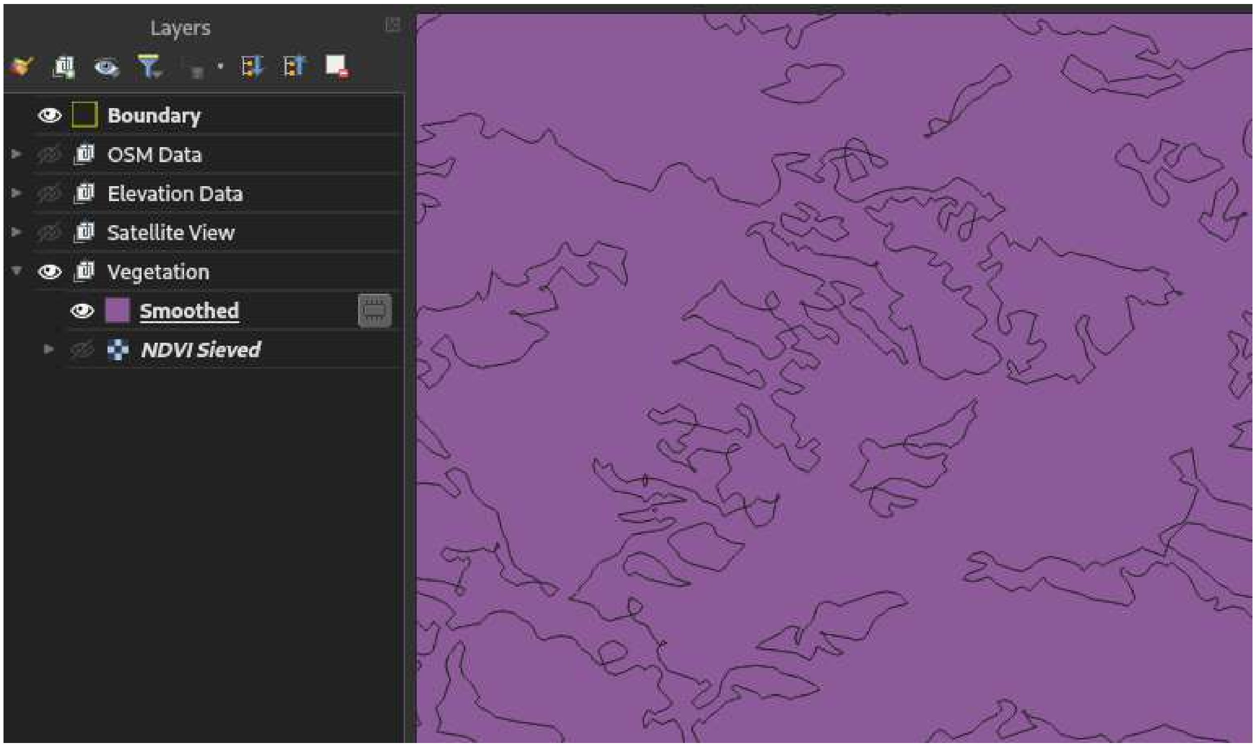
Your screen will fill with a ton of very squarish, pixelated shapes and a new vector layer. It looks like a huge step back at first, but it's still headed in the right direction.



Before we turn this into a usable symbol, let's smooth these shapes out to avoid it looking so artificial, and to give us a better sense of the blurrier boundaries of vegetation on the ground. Go back to the processing toolbox (`Processing > Toolbox`), and this time, pick `Smooth` from the search bar. `Vector Geometry > Smooth` should be the only option; double click this to bring up the settings for the filter.

I recommend using 0.5 for the `Offset`, or how close the filter will stick to the original; we want roughly the same shape, but not so rigidly adhering to pixels. I usually go with somewhere between 4 to 6 for the `Iterations` - larger values can crash QGIS or your computer due to the memory required; all this is doing is telling the filter to repeat the smoothing process a few times to ensure things are appropriately de-pixelized. At this point, we can save the file somewhere, so feel free to pick a good file location in the output.

The result should look similar to the below; the shapes are much more organic, and are ready to have symbology applied.



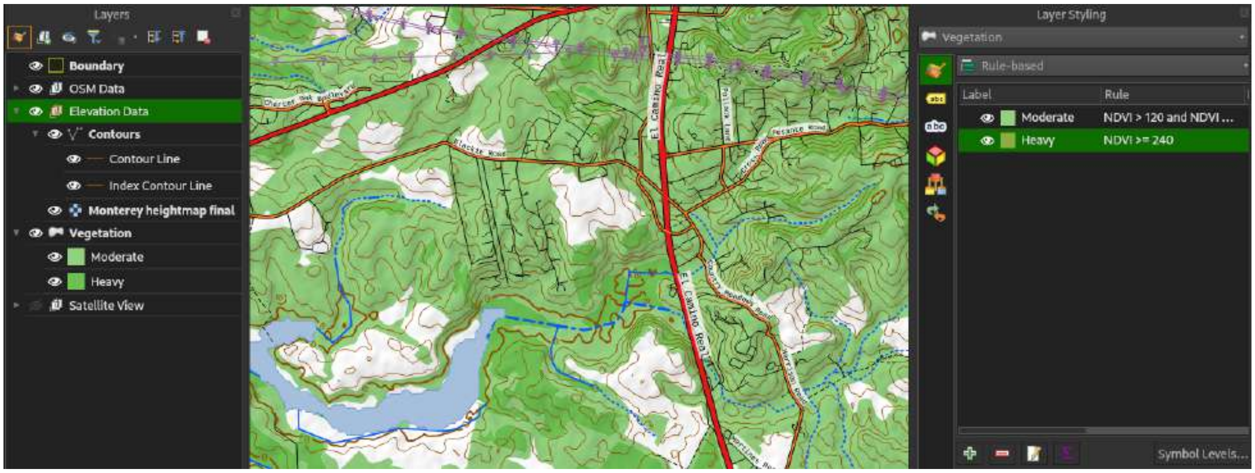
Styling vegetation We have multiple classes of vegetation in one layer, and need to apply a different style for each. If that sounds like another rule-based symbol in the Layer Styling panel, you're right. Go to the Layer Styling panel, and create a new rule.

The process of polygonizing our data has turned the values from 0-1.0 to 0-255, and smoothing it out has introduced some flex in what values these polygons ended up being assigned. As a result, I usually use values similar to the following for the rules for vegetation, although you might want to play around with the values to see what gets results most accurate to your satellite imagery.

- Moderate vegetation: `NDVI > 120 and NDVI < 240`
- Heavy vegetation: `NDVI >= 240`

For moderate and heavy vegetation, I recommend using no stroke, and fill `#94d180` and `#6fc154`, respectively. I typically will remove moderate vegetation for printing, if I look at the area and assess that most of the "moderate" vegetation isn't going to affect mobility.

With all the other map features re-enabled, we now have a pretty solid map going like the below; we could add a grid here and it'd be extremely usable for backcountry navigation.



Label	Rule
Moderate	NDVI > 120 and NDVI ...
Heavy	NDVI >= 240

There are a few more steps we're going to take, though, adding populated areas, working with some more of the OSM data we downloaded earlier, and finally adding a grind, before we're ready to turn all this map data into a completed, printable map.

Adding manmade features

Buildings

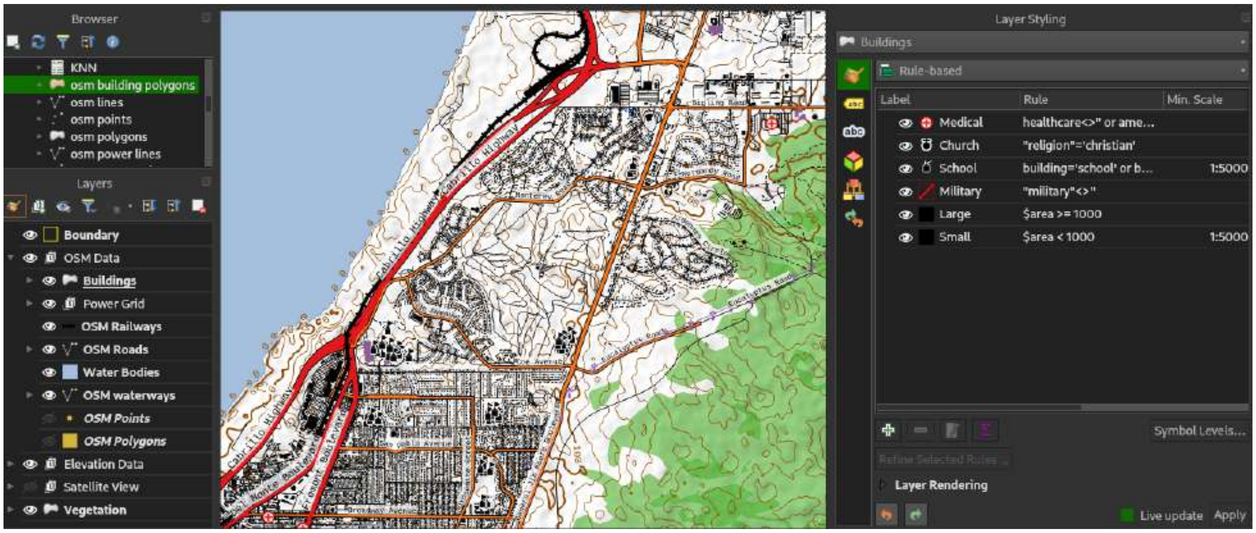
We can do a rule-based symbol set for the overall QuickOSM import of polygons we did earlier for buildings, or we can make a QuickOSM query, querying on key `building` for all values. I'll do the latter for convenience's sake, and to allow for easier separation of objects later on.

Depending on the area you live, this may be a *ton* of data, with each individual building down to single-family homes being its own polygon. You can get rid of the lines layer, however, if one is imported, and likely the points layer; many of the latter are simply names of buildings.

A no-outline, black-filled symbol for the buildings is most appropriate, however, there are a few tricks we can use if we want to reduce clutter:

- We can only show buildings larger than a certain square meter footprint, by having a rule-based filter of `$area > X`, where X is the footprint. I use 1000 m as a rule of thumb here. We can even combine this to show smaller buildings closer in by using a scale-dependent visibility (I use 1:50,000 to 1:1).
- We can highlight sites of particular interest, like highlighting military buildings with a filter like `"military"<>''` .
- For highlighting a building with a single marker, instead, we can pick the `Centroid fill` type, which will let us add a single marker to mark out, say, churches, with `religion='Christian'`.

For example, a map filled with rule-based symbols for churches, medical facilities, and schools, in addition to buildings, might look similar to the below.



Label	Rule	Min. Scale
Medical	healthcare<>'' or ame...	
Church	"religion"='christian'	
School	building='school' or b...	1:5000
Military	"military"<>''	
Large	$area >= 1000	
Small	$area < 1000	1:5000

I recommend using the OSM points layer, however, as it typically has more of the building-type data, to add things like churches, medical facilities, and the like. Some useful filters:

- Medical: `healthcare = 'hospital' or healthcare = 'clinic'`
- Churches/mosques/temples: `religion='christian'` / `religion='muslim'` / `religion='buddhist'` / etc.
- Misc. religion: `religion not in ('christian', 'muslim') and religion <> ''`. You might notice this as a shortcut for a lot of `or` statements, too...

Other OSM areas of note

From the full OSM polygon import, we can also add miscellaneous areas that may be important to note:

- Wetlands, with filter `wetland<>'' or nature='wetland'`
- Beaches, with filter `natural in ('beach', 'sand')`
- Forests, with filter `natural='wood'`

Political boundaries

Querying OSM on key `boundary`, just like all the other OSM queries we've done, will get us boundaries (country, state, city, etc.)

However, we can pull this directly from our data by creating a new virtual layer, essentially a layer pulled from another layer's data.

If our OSM polygons layer is called "OSM Polygons," for example, we can pull boundary data by going to `Layer > Create Layer > New Virtual Layer`. In the bottom query box, we can put:

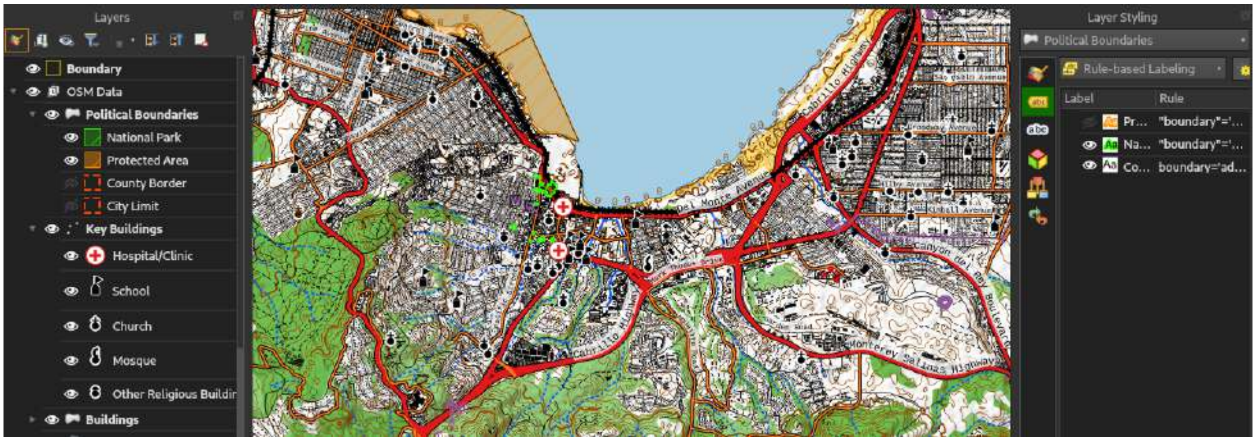
```
Select geometry, boundary, admin_level, name from "OSM Polygons" where boundary<>''
```

Make sure `Geometry` is set to `Autodetect` and hit `Add`, then, after it's ready, `Close`. We should have a new layer that's identical to doing a new OSM query. We can clip, reproject, etc. this layer in order to create a new, lightweight layer that's just boundaries.

Some filters for rule-based symbols here that might be helpful:

- National park boundaries: `boundary='national_park'`
- Protected area (wildlife preserve, etc.): `boundary='protected_area'`
- Political boundary: `boundary='administrative' and admin_level=X`
  - X is the admin level, where 8 = city, 6 = county, 5 = state, etc.

After adding some of these areas and key buildings as mentioned above, the map is starting to look like the below:



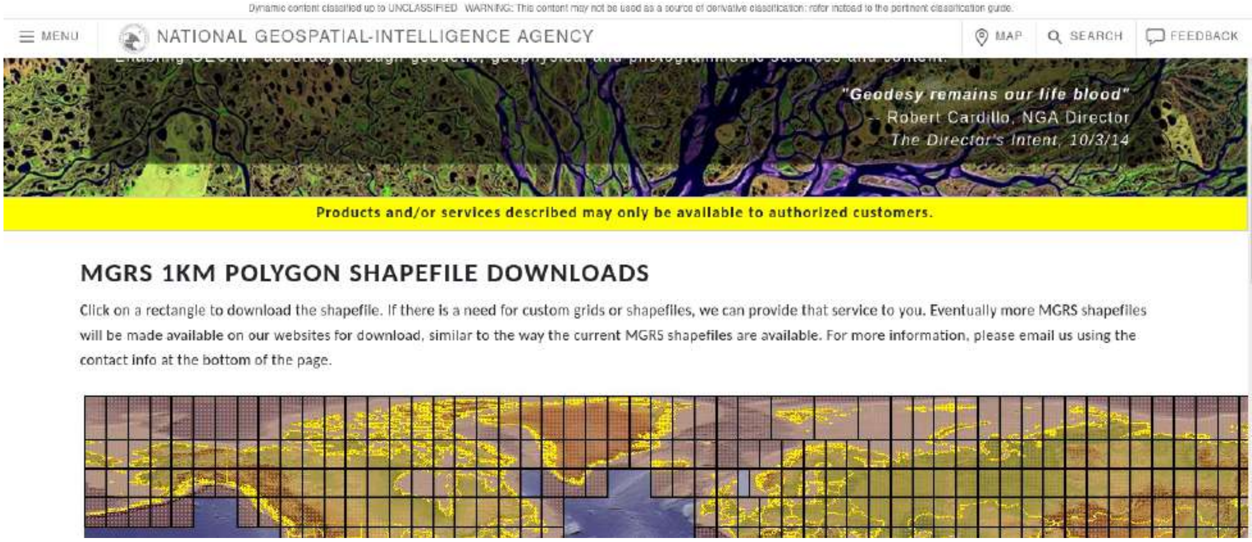
Adding the MGRS grid

In order to position ourselves (relatively) precisely on the map and communicate our position to others, we'll make use of the Military Grid Reference System (MGRS). The MGRS is based on the UTM grid - the actual grid lines are the same, but parts of the actual coordinates used are different. For our purposes .

There are multiple ways to do this, some of which are quite complex, especially when multiple grid zones (whose lines don't align between each) and involve some basic programming to extract coordinates. However, there's a simple shortcut, and involves downloading the freely available MGRS shapefiles made available by the National Geospatial agency.

Go to their site to download the shapefiles:

https://earth-info.nga.mil/index.php?dir=coordsys&action=mgrs-1km-polyline-dloads



Dynamic content classified up to UNCLASSIFIED WARNING: This content may not be used as a source of derivative classification; refer instead to the pertinent classification guide.

"Geodesy remains our life blood"
-- Robert Cardillo, NGA Director
The Director's Intent, 10/3/14

Products and/or services described may only be available to authorized customers.

MGRS 1KM POLYGON SHAPEFILE DOWNLOADS

Click on a rectangle to download the shapefile. If there is a need for custom grids or shapefiles, we can provide that service to you. Eventually more MGRS shapefiles will be made available on our websites for download, similar to the way the current MGRS shapefiles are available. For more information, please email us using the contact info at the bottom of the page.

Select the region you're interested in (or multiple, if your area of interest spans multiple grid zones), and save the resulting file in a good location.

Note that if you have a large area of interest, it may be of use to download the 100km (letter designator) shapefiles as well at their corresponding page for 100 km square. I'll only be covering the process for 1 km grid lines, but the process is the exact same for each. You can similarly download actual grid zone shapefiles from the NGA, if you have an exceptionally large area of interest.

Go back to QGIS and add a new vector layer with `Layer > Add Layer > Add Vector Layer`, then select the `.shp` file from the folder you extracted the MGRS grid shape files to. Leave everything as it is by default, and hit `Add`. Like other vector layers, it's a good idea to clip it to the boundary of our area of interest, and it's **essential** for us to reproject it into our project CRS.

Now that we have our grid overlaid on the map, we'll have to style it; having an opaque set of squares on our map is probably not the best move for usability's sake.

Open up the Layer Styling panel again and change the fill type to `Outline: Simple Line`. A simple 0.25 mm black line is all we need.

Side story: Algorithmically generating grid subdivisions

What about adding in further subdivisions of 1 km grids, for example when we have a 1:12,500 scale map and want to use a protractor that tops out at 1:25,000 scale to navigate with a little mental math?

We could, for example, change our MGRS grid layer's styling into a rule-based symbol, with the existing black line on without a filter.

Add another rule, also without a filter, but this time set a scale-dependent visibility, i.e. from 1:40,000 to 1:1.

Change the fill type from `Simple Fill` to `Geometry Generator`, though. The geometry generator function is incredibly flexible, although requires some more thought; it lets us programatically create geometry (points, lines, and polygons) from other geometry. Set the `Geometry type` to `LineString / MultiLineString`, since we want to create lines within each grid square. In the larger box below, we'll use the following expression:

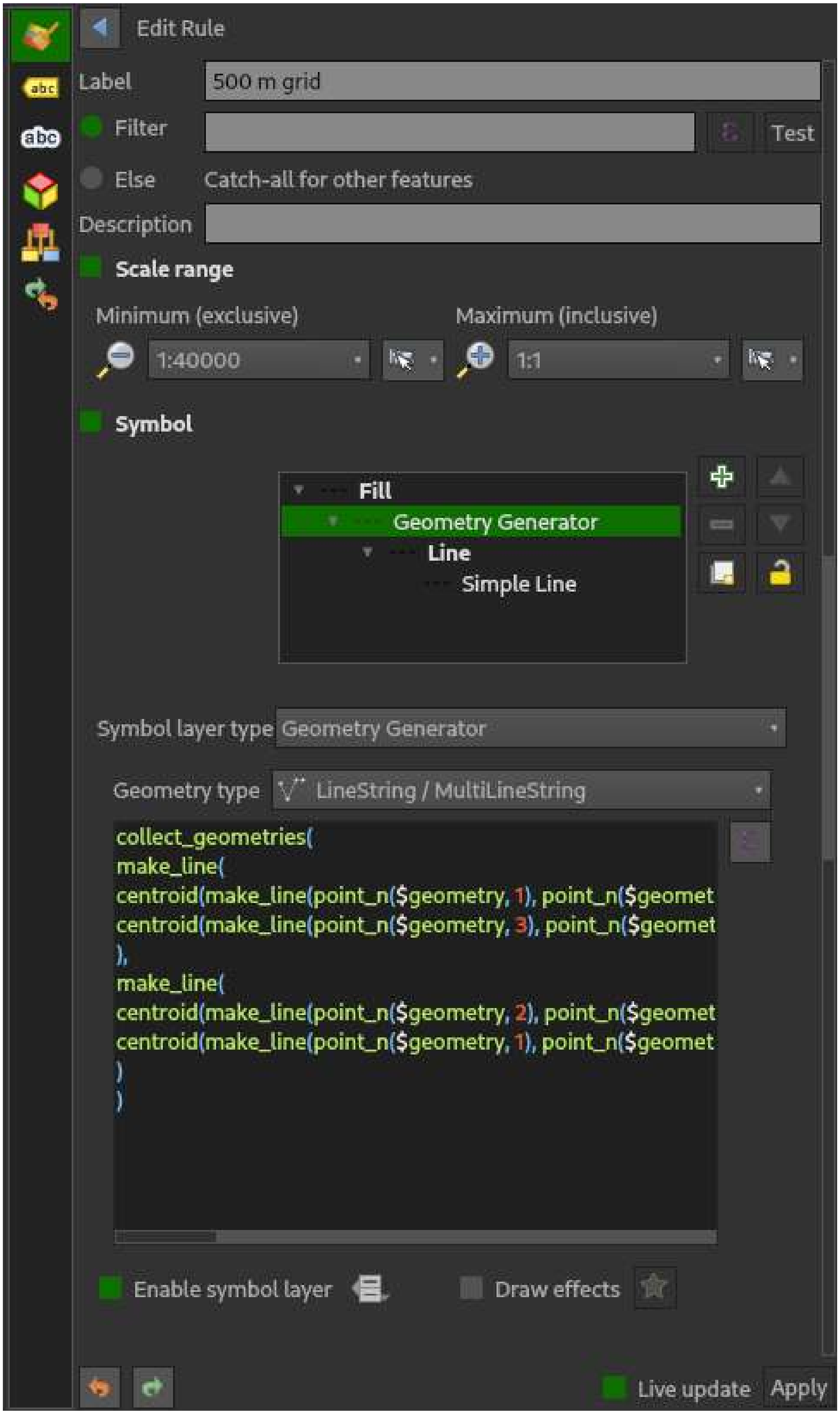
```
collect_geometries(
make_line(
centroid(make_line(point_n($geometry, 1), point_n($geometry, 2))),
centroid(make_line(point_n($geometry, 3), point_n($geometry, 4)))
),
make_line(
centroid(make_line(point_n($geometry, 2), point_n($geometry, 3))),
centroid(make_line(point_n($geometry, 1), point_n($geometry, 4)))
)
)
```

A brief explanation on what all these statements mean:

- The `collect_geometries` statement is essentially telling QGIS to treat each of the following items, separated by commas, as new geometry items.
- The `make_line` functions, similarly, tell it to make a line out of the two comma-separated points within its parenthesis.
- `centroid` tells it to return the centroid (middle point), of the geometry within its parenthesis.
- `point_n` returns the Nth point of the geometry, in the form `(geometry, n)`. Note that unlike common programming languages, which start counting indices at 0, QGIS starts at 1 for some reasonˆ[Yes, it bothers me too.].

After entering that, we'll see our grid be subdivided into two, creating a 500 m grid. We can edit the `Line` symbol that appears below `Geometry Generator` to control the appearance of the generated geometry; an 0.25 mm black line is what I recommend going with.

The overall settings for our 500 m grid rule should look similar to the below.



Label: 500 m grid

Filter

○ Else: Catch-all for other features

Description

🟩 Scale range

Minimum (exclusive): 1:40000

Maximum (inclusive): 1:1

🟩 Symbol

Symbol layer type: Geometry Generator

Geometry type: LineString / MultiLineString

```
collect_geometries(
make_line(
centroid(make_line(point_n($geometry, 1), point_n($geomet
centroid(make_line(point_n($geometry, 3), point_n($geomet
),
make_line(
centroid(make_line(point_n($geometry, 2), point_n($geomet
centroid(make_line(point_n($geometry, 1), point_n($geomet
)
)
```

🟩 Enable symbol layer

□ Draw effects

🟩 Live update

Grid labels

The handy thing about the NGA imported MGRS shapes is that they come with attributes already defined, including the grid zone designator, their 4-digit grid coordinate (i.e. "This is the square 10S EF 1234"), and other useful information. Turning that into useful labels in a form that's actually readable without cluttering the map, however, will take a bit of work.

(For those curious about seeing or editing the attributes of a layer, you can right-click on the layer in the Layers panel and hit `Open Attribute Table`, as well.)

Let's create a new rule-based label set for our MGRS shapes.

For our eastings (west-to-east) numbering, we won't set a filter; free free to hit the `Else` option that will show every feature. To actually filter out the labels and prevent every square from showing its label, we'll rely on the label placement itself.

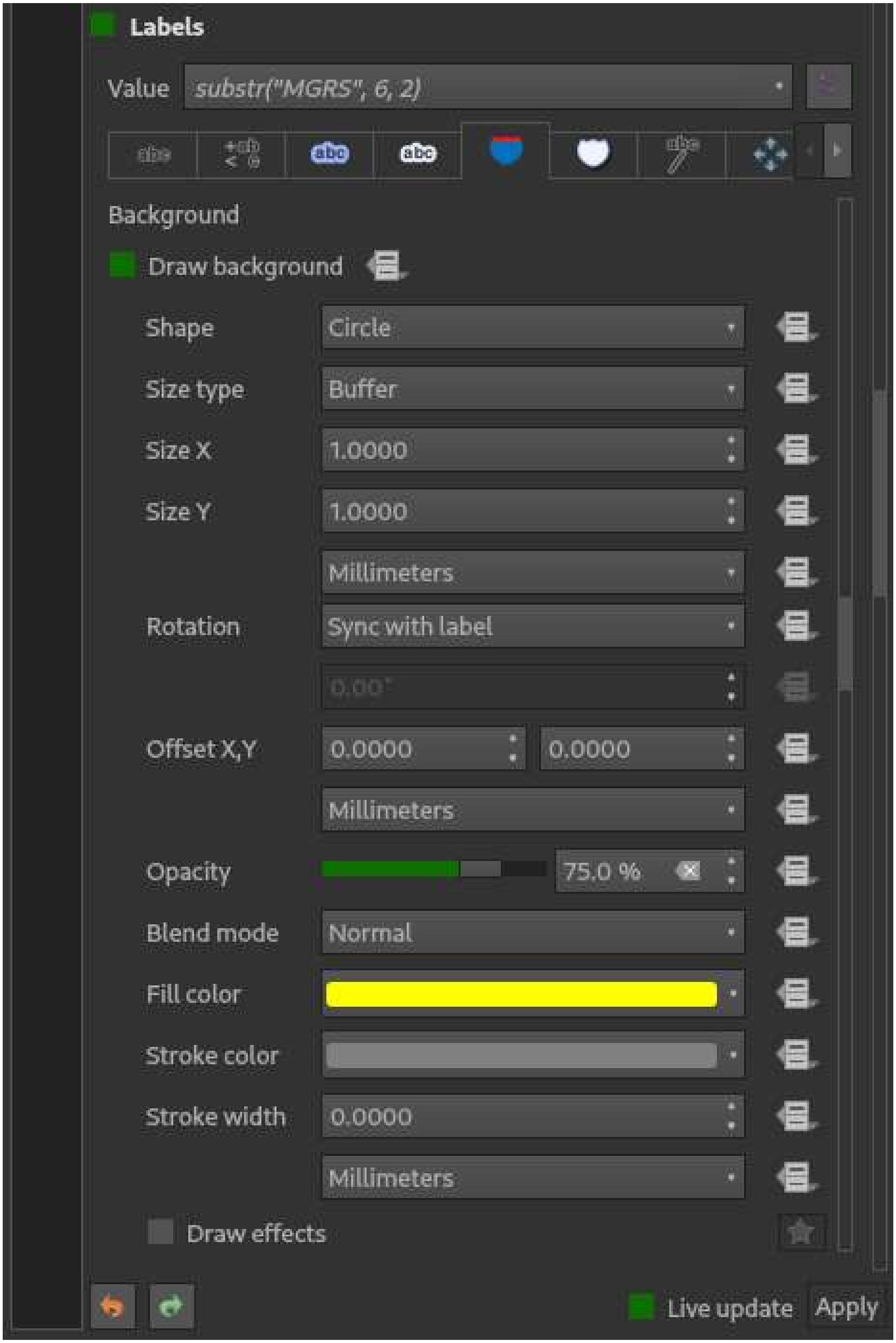
The attribute for the MGRS square within the shapefiles we downloaded is, appropriately enough, `MGRS`. The actual value will include the grid zone designator and the 100-km square value, so a grid zone might look like `10SEF1234`. We need to turn that label into just, for the east-to-west, the `12` portion. In the `Value` box under `Labels`, we'll use the substring function for QGIS's expressions to get only those numbers, remembering QGIS counts from 1:

```
substr("MGRS", 6, 2)
```

Every square will, at this point, have its center (roughly speaking) labeled with its east/west two-digit grid only, at this point. We want to make only one label per map for this, however, and appropriately position it for ease of legibility and usability.

First, though, we'll add a style to it to make it a little more legible.

We'll first set the font to `Liberation Mono` again, or another font of your choice, ideally monospace, at 10-point size. Then we'll go over to the `Background` tab, with the icon looking like a road sign, and hit `Draw background`. Choosing a circle shape and coloring it a semi-transparent yellow (I find 75% opacity works well), we now have a pretty readable but not distracting style. The settings I use for the background are as below.



Labels

Value *substr("MGRS", 6, 2)*

Background

Draw background

Shape	Circle
Size type	Buffer
Size X	1.0000
Size Y	1.0000
	Millimeters
Rotation	Sync with label
	0.00°
Offset X,Y	0.0000 0.0000
	Millimeters
Opacity	75.0 %
Blend mode	Normal
Fill color	🟨
Stroke color	▨
Stroke width	0.0000
	Millimeters

□ Draw effects

Label placement Go down to `Placement`, the four-arrow icon you might remember from earlier.

We want the mode to be `Offset from point`, and ensure the center item is picked, since we don't want the label to be offset to one side of the actual point we designate for the grid label.

To actually define the location of the grid we want, it's time to use the Geometry Generator again. Enable that checkbox. We'll use the following expression to place *every* east/west MGRS grid on the bottom edge of the map:

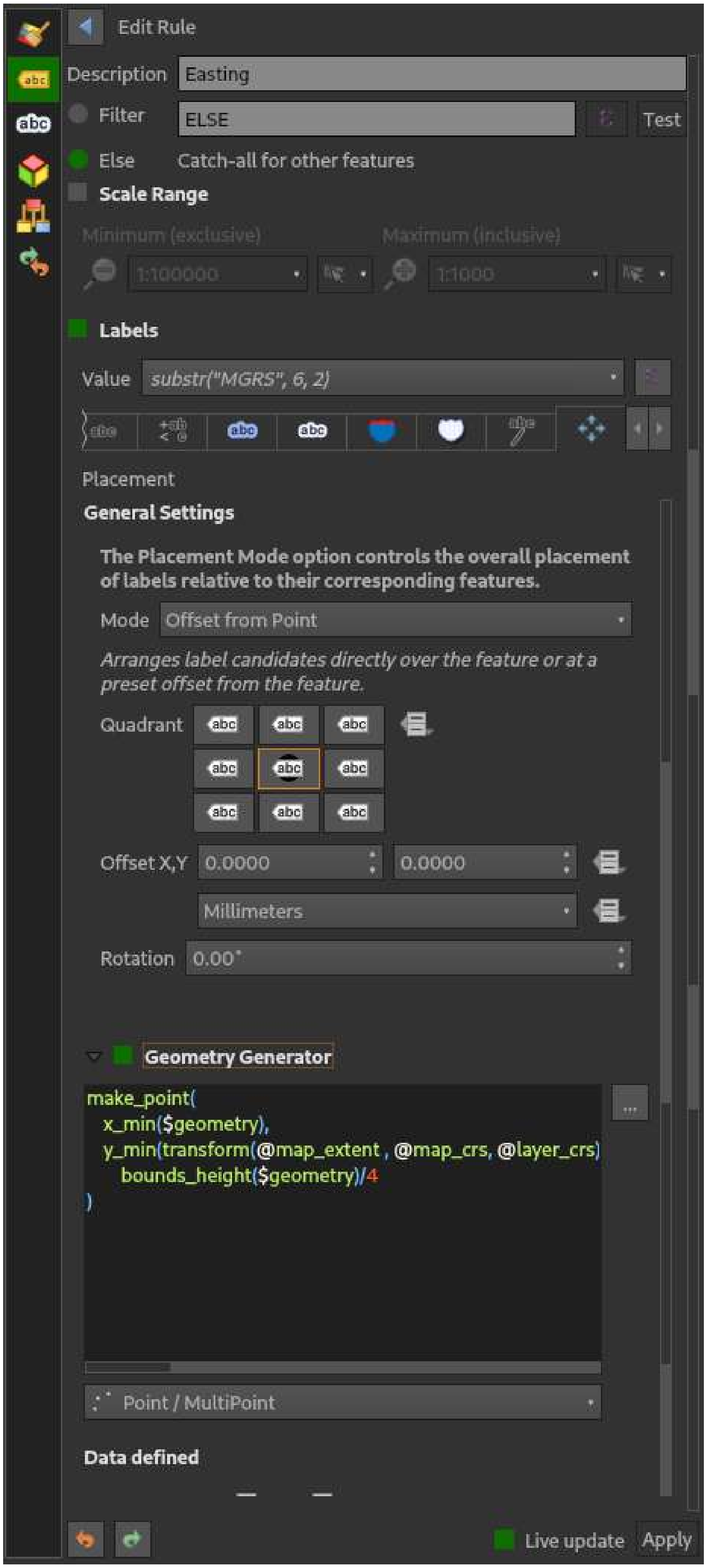
```
make_point(
    x_min($geometry),
    y_min(transform(@map_extent , @map_crs, @layer_crs)) +
        bounds_height($geometry)/4
)
```

To explain this:

- `make_point(X, Y)` designates a new point at the given (X, Y) coordinate.
- `x_min` gets the minimum X (east to west) value of a particular geometry; '`x_min($geometry)`, in this context, tells QGIS that we want the x coordinate of our grid label to be exactly on the left edge of each grid square. Since, if we've chosen the right coordinate system, this will be straight up and down, this will fall exactly on the grid line.^[Note that this won't be the case if we're not using a grid zone CRS, such as when we have multiple top-level grid zones in a map. I'll leave figuring this out as an exercise to the reader; as a hint, labeling on the centroid of the line will give you just one label as well...]
- `@map_extent` returns a polygon, in QGIS's internal CRS for drawing the map, representing the edges of the active map view.
- `transform(G, F, T)` will transform the given geometry (G) from its original CRS (F) into the specified CRS (T). We need to use this, otherwise adding values with different CRS's will produce strange or wildly off the map results.
- `@layer_crs` returns the CRS of the active layer getting styled - in this case, the CRS of our grid shapefiles.
- `@bounds_height(G)` returns the height of the distance from the topmost point in a geometry (G) to the bottom.

All in all, we're telling the geometry generator that we want the label to be centered on a point whose X component is the leftmost edge of the grid square itself, and with a Y component equal to the bottom edge of the map plus 1/4 of the grid square height (so it's not hanging unreadably off the map).

The settings should look like the following:



Edit Rule

Description: Easting

○ Filter: ELSE

● Else: Catch-all for other features

□ Scale Range

Minimum (exclusive): 1:100000

Maximum (inclusive): 1:1000

🟩 Labels

Value: substr("MGRS", 6, 2)

Placement

General Settings

The Placement Mode option controls the overall placement of labels relative to their corresponding features.

Mode: Offset from Point

Arranges label candidates directly over the feature or at a preset offset from the feature.

Quadrant

Offset X,Y: 0.0000 0.0000

Millimeters

Rotation: 0.00°

🟩 Geometry Generator

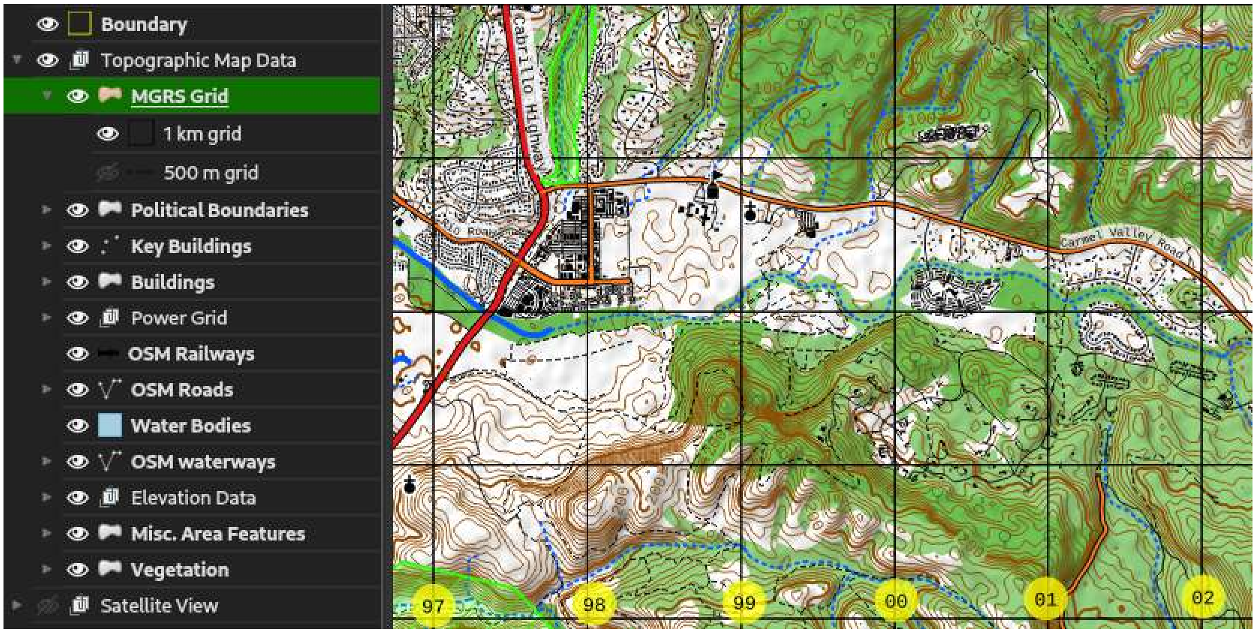
```
make_point(
  x_min($geometry),
  y_min(transform(@map_extent , @map_crs, @layer_crs)
    bounds_height($geometry)/4
)
```

Point / MultiPoint

Data defined

🟩 Live update

Hitting `Apply` will bring us to a view similar to the following, with eastings neatly arrayed at the bottom of the map:



You'll notice that the numbering restarts from `00` when it crosses a 100 km square boundary; I'll address this in a similar fashion after I add the northings.

Adding the northings (south-to-north numbering) is virtually identical; the only differences are that we use `substr("MGRS", 8, 2)` for the actual value, and the following expression in the geometry generator:

```
make_point(
    x_min(transform(@map_extent , @map_crs, @layer_crs)) +
        bounds_width($geometry)/1.5,
    y_min($geometry)
)
```

Note that I've made the distance the labels are pushed out along the X axis a little bit more; this is to prevent them from overlapping with the easting labels at the bottom left corner of the map.

100 km grid Adding the 100 km shape is exactly the same, downloaded from the NGA page mentioned above. The download comes bundled in quite a few folders, but dig in all the way and import the ‘`.shp` into QGIS the same way.

The only changes I recommend for the symbol itself making are changing the grid style from 0.25 mm to 1 mm in width.

For the label, I make the background an orange rectangle, with 12-point font instead of 10 point, and use `100kmSQ_ID` for the label value.

Under the `Rendering` tab (just to the right of the label placement tab), I also hit `Show all labels for this layer`, to force the rendering of the 100 km grid labels no matter what, and input 1 for the `Label z-index` to ensure they remain on top.

For the geometry generator, I use the following statement:

```
make_point(
    min(
        max(
            x_min(transform(@map_extent , @map_crs, @layer_crs)),
            x_min($geometry)
        ) + bounds_width($geometry)/240,
        x_max($geometry)
        ),
    min(
        max(
            y_min(transform(@map_extent , @map_crs, @layer_crs)),
            y_min($geometry)
        )+ bounds_height($geometry)/240,
        y_max($geometry)
    )
)
```

This is similar to the statement for the 1 km labels, however, it has a few more notable features that should be easily identifiable to those conversant with any kind of programming. Essentially, it uses, for an X-coordinate, the left most of the following two values:

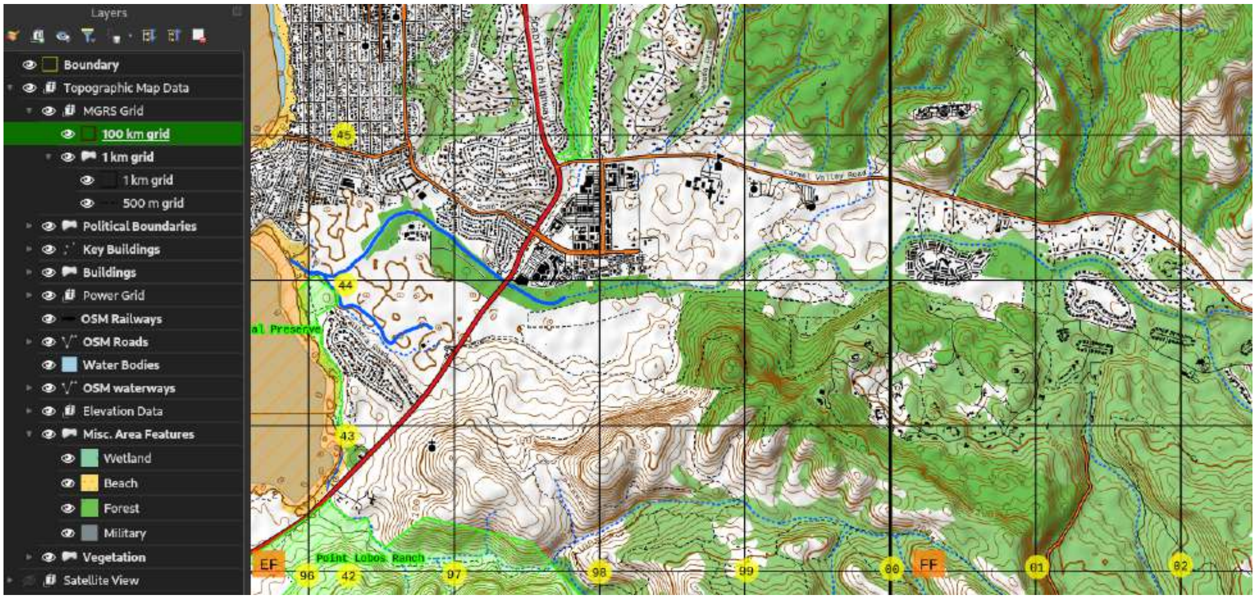
- The rightmost edge of the 100 km square
- The leftmost of either the left edge of the 100 km square, or the left edge of the screen.

It uses a near-identical calculation for the Y-coordinate. In sum, this will draw the 100 km square label at the lower left edge of its square, or the lower left edge of the screen as long as some of the square is visible, but will not cross into the next 100 km square.

This will ensure that reasonable-looking labels will occur no matter what, keeping in mind that either or both of the labels can be turned off for printing if they happen to get in the way.

Adding in top-level grid zone separators is similar, although requires more math due to the fact that lines from two adjacent grid zones are not parallel.

With a few more tweaks to the buffer size for the label backgrounds, we can finally arrive at a map that's usable for navigation and positioning, as seen below.



Step 2: Adding our own information

Depending on the purpose of your map, you might need or want to add your own specific information on the map before printing, including key locations of personal interest to you, such as your home, waypoints for a planned trip, no-go areas in a national park or forest, range lateral limits, and other important information that's less likely to change quickly.

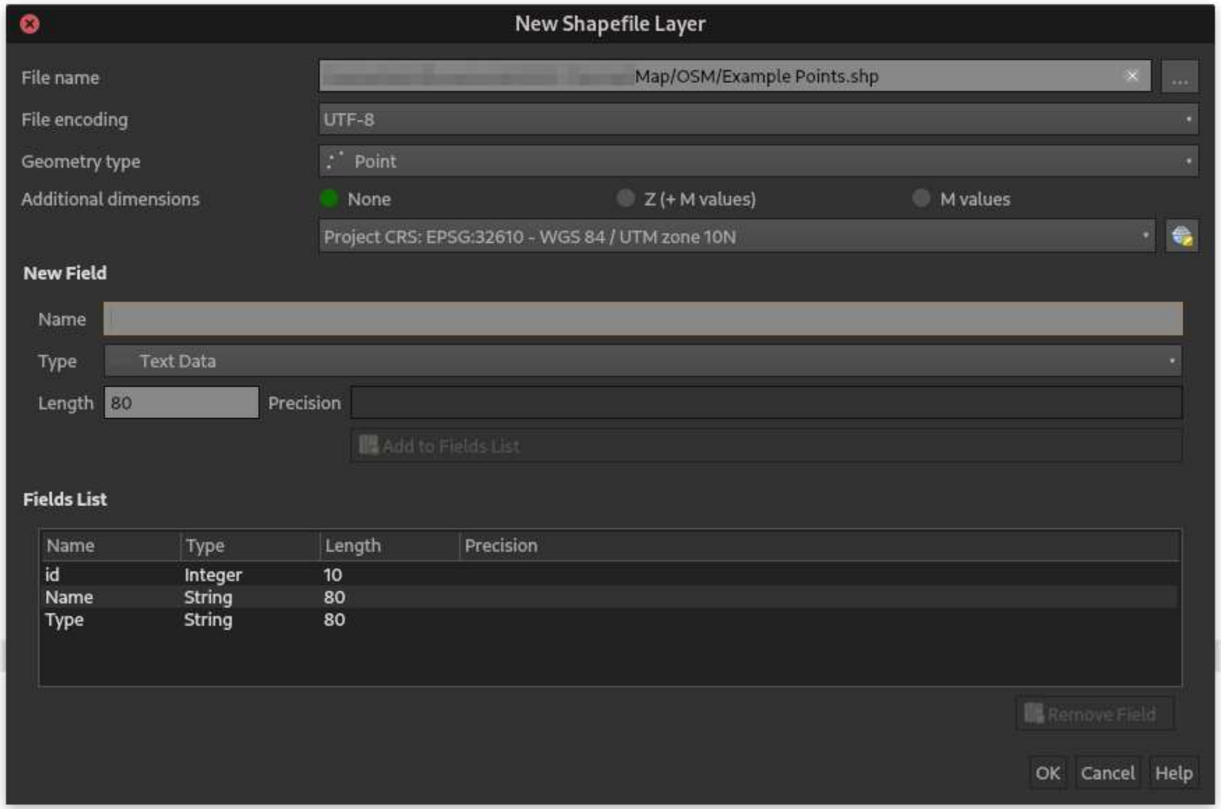
Luckily, we've already finished the hard part of plotting data on our map, and at this point, you've likely developed a pretty good basic understanding of how to add and style layers in QGIS. In the next segment, I'll walk you through how to add your own basic point, line, and area layers and fill them with data, as well as how to add precise points from existing MGRS coordinates.

Creating data layers

The process is simple; let's add an example point layer. I'll be adding trip waypoints through a park purely for the purpose of example.

Go to `Layer > Create Layer > New Shapefile Layer`. First and foremost, pick a location to save your layer, just like we've done with all our layer processing before. Leave the encoding as is, then make sure we have the correct CRS selected. (Ignore the Z-coordinates for now, although note that these can be used to create 3D maps if you're into that sort of thing, later.)

Let's add a couple fields to our data. Under `New Field`, put `Name` in the `Name` box, and click `Add to Fields List`. It'll appear in the box below. Do the same and add a `Type` field. The actual name of the field is unimportant, as a note, and you can create fields with other types of data - numbers and dates, for shapefile fields, or pick a different max length from the default 80-character limit for the field. The completed layer setup screen should look like the below image.


New Shapefile Layer

File name: [redacted]Map/OSM/Example Points.shp

File encoding: UTF-8

Geometry type: Point

Additional dimensions: ● None ○ Z (+ M values) ○ M values

Project CRS: EPSG:32610 - WGS 84 / UTM zone 10N

New Field

Name:

Type: Text Data

Length: 80

Precision:

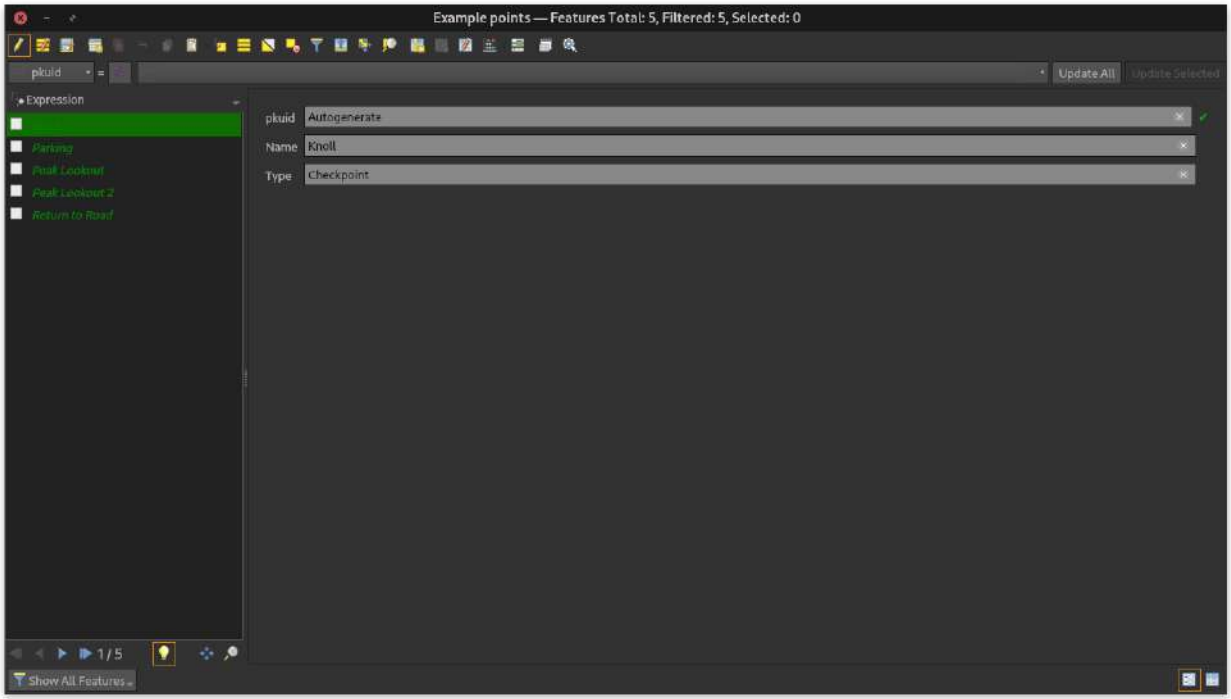
Fields List

Name	Type	Length	Precision
id	Integer	10	
Name	String	80	
Type	String	80	

Once we have all the fields input, click `OK`. The new layer will appear in our Layers panel, and we can start adding data.

Ensure the `Digitizing`, `Advanced Digitizing`, and `Shape Digitizing` toolbars are enabled under `View > Toolbars`. Then, like before, we'll enable editing on the layer by clicking the yellow pencil icon (`Toggle Editing`). After that, go to `Add Point Feature`, two icons to the right, or hit `Ctrl+.` Click on the map where you want the point, and fill in the `Name` and `Type` fields on the subsequent dialog before hitting `OK`. Points will appear on the map as you go.

If you need to alter the data, feel free to right-click on the layer in the Layers panel and use `Open Attribute Table` to edit the data in the points' fields, as seen in the below image.



Example points — Features Total: 5, Filtered: 5, Selected: 0

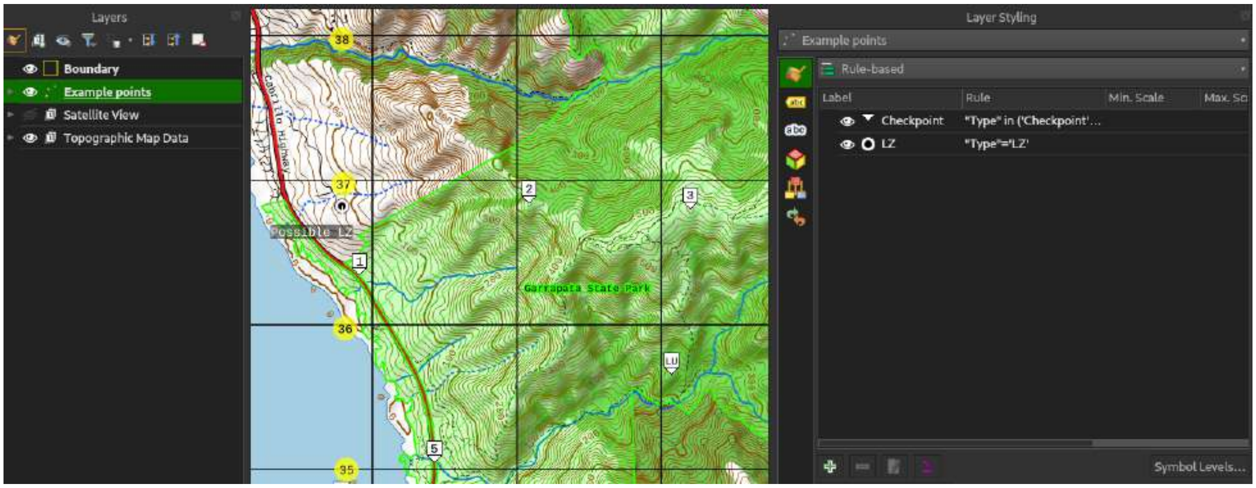
pkuid: Autogenerate

Name: Knoll

Type: Checkpoint

When you're done editing, click `Save Edits` in the attributes table, or `Save Layer Edits` on the digitizing toolbar, and disable editing with the yellow pencil again.

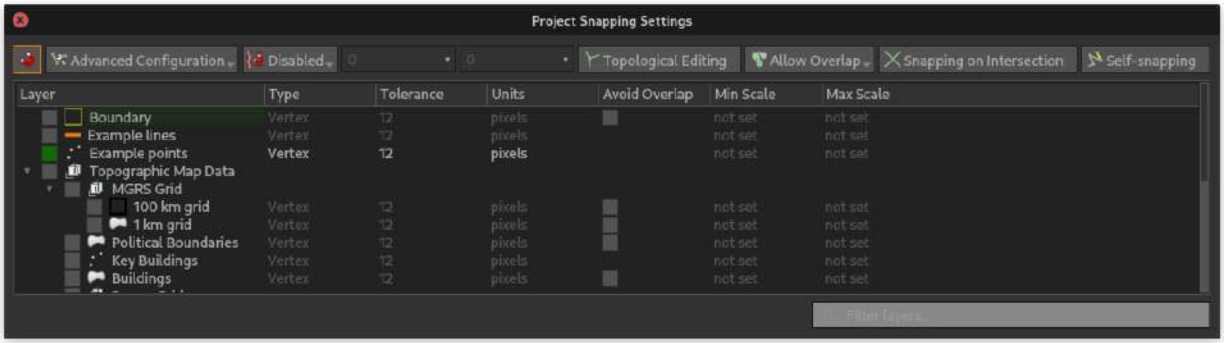
We can style or add labels to these points just like we have for all the data we pulled from elsewhere, using the field names we created as the label value, including creating rule-based symbols and labels for them. For example, here are the points I've added using a rule-based setup to distinguish between checkpoints and a point I plan to link up with a friend at:



Label	Rule	Min. Scale	Max. Sc
Checkpoint	"Type" in ('Checkpoint'...		
LZ	"Type"='LZ'		

Adding a line layer is just as straightforward. Go through the same process, but simply choose `LineString` as the `Geometry type`. Adding the line is just as easy - left-click on each point (including the ending point), then right-click to finish the line. If you want to snap to features on the map, go to `Project > Snapping`

`Options` and click on the magnet icon on the toolbar that appears, as well as configure the options as you like, including using the `Advanced Configuration` button to only snap to certain layers.



Layer	Type	Tolerance	Units	Avoid Overlap	Min Scale	Max Scale
☐ Boundary	Vertex	12	pixels	☐	not set	not set
☐ Example lines	Vertex	12	pixels		not set	not set
☑ Example points	Vertex	12	pixels		not set	not set
☐ Topographic Map Data						
☐ MGRS Grid						
☐ 100 km grid	Vertex	12	pixels	☐	not set	not set
☐ 1 km grid	Vertex	12	pixels	☐	not set	not set
☐ Political Boundaries	Vertex	12	pixels	☐	not set	not set
☐ Key Buildings	Vertex	12	pixels		not set	not set
☐ Buildings	Vertex	12	pixels	☐	not set	not set

Adding areas is similar, yet again. This time, however, we can use the ‘`Shapes Digitizing` toolbar to add shapes easily, like circles and rectangles. The same logic of "left-click to add points, right-click to end the shape" applies.

Adding points from MGRS coordinates

There’s actually a plugin for working with a lot of different types of coordinates for QGIS, and it’s extremely helpful for inputting coordinates and copying them for use in other activities.

Like we did with SRTM Downloader, go to `Plugins > Manage and Install Plugins`; the name of the plugin we’re looking for is `Lat Lon Tools`. Go to `Toolbar > Lat Lon Tools` to enable it, or use the same functions located at `Plugins > Lat Lon Tools`.

For just MGRS, or faster access to MGRS specific tools, there’s a plugin by the same author with just MGRS functionality, appropriately called `MGRS Tools`. It can be accessed through `Toolbars > MGRS Toolbar` or `Plugins > MGRS Tools`

A couple of the features you can use between these two:

- **Inputting points from coordinates**: When a layer is open for editing, go to `Lat Lon Digitize` in Lat Lon Tools. Use the globe icon to select your preferred format (e.g. MGRS), then just paste the coordinates into the bar. It’ll create a point in the layer you’re editing at those coordinates.
- **Copying coordinates to the clipboard**: Use the `Copy/Display MGRS Coordinate` tool in MGRS Tools to click a spot on the map and have its MGRS coordinates added to the clipboard for pasting elsewhere.
- **Converting coordinates between systems**: The `Coordinate Conversion` tool in Lat Lon Tools will help you there - just enter coordinates in the appropriate box in the dialog, and the other boxes will fill with the equivalent in other coordinate systems.

There’s one more tool that may be of interest, and it’s Lat Lon Tools’ `Plugins > Lat Lon Tools > Conversions > Point Layer to MGRS`. Like most of the other layer tools in QGIS, this will let you pick an input point layer, name a field for the output layer, by default "mgrs", and an output layer location. The result will create a new layer that’s the same as the existing one, but with the MGRS coordinates of the points embedded in the field you named in the dialog. This can be handy for exporting to other programs, for instance.

In an expression for the label value (or anywhere else that takes an expression), you can also get a feature’s MGRS coordinate using the following expression:

```
mgrs(y($geometry), x($geometry), @layer_crs)
```

This can be handy to label items on the fly with their coordinates, without having to create a new layer.

Side story: Exporting point layers to spreadsheets or CSV files

If you've got a bunch of points and want to export them to a spreadsheet for further processing, or to adjust them into a format for another program, share with a friend, etc., QGIS makes it easy to do so as well.

Just right-click on the layer in the layer panel, and go to `Export > Save Features As`. From there, you can choose a `.ods` Open Document spreadsheet, a `.xlsx` Microsoft Excel file, or a `.csv` comma-separated value text document.

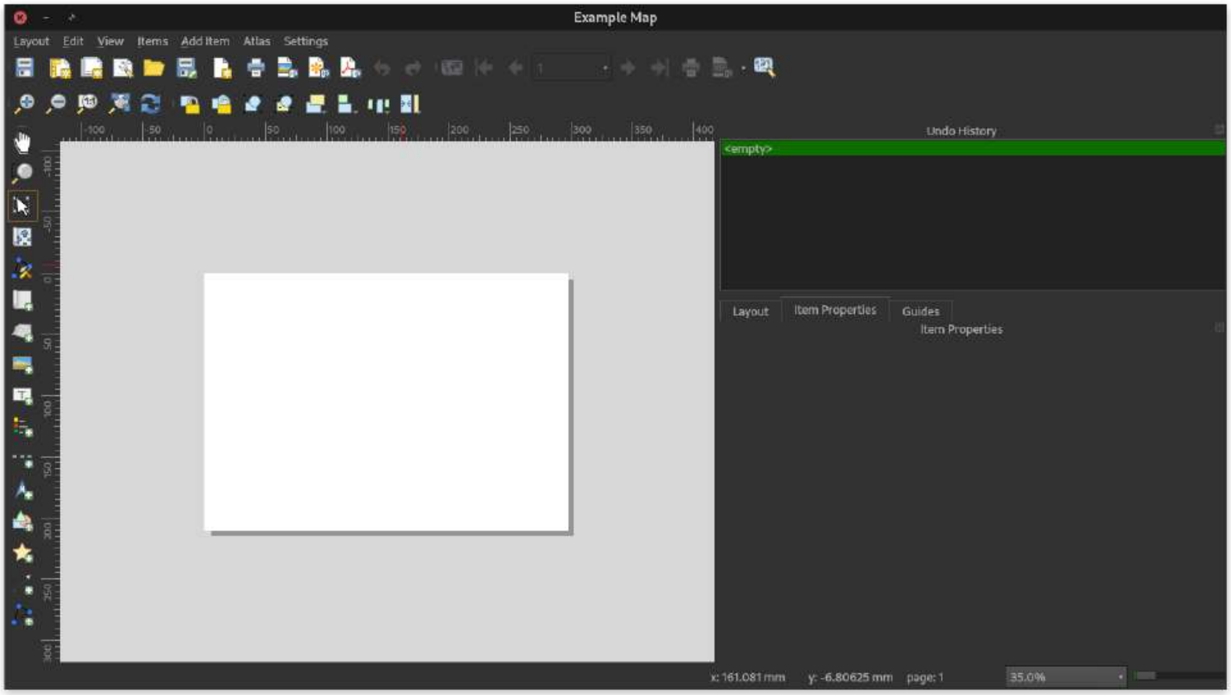
This is only under its own heading to let you jump to it from the table of contents in your PDF reader; it's really that easy and that's all there is to it.

Step 2: Layout and printing

Now that we have all the data we need on the computer, it's time to make the jump to putting it on a piece of paper we can use for actually navigating around. This is, luckily, a lot simpler than the process of getting and formatting all the data we already took care of.

QGIS manages its printing views with layouts, and you can have multiple layouts per project - for example, to have multiple maps at different scales for different parts of a trip. To add a new layout, go to `Project > New Print Layout`, and put in a title for your map when prompted. Hit `OK`.

The layout interface is pretty similar to other graphics programs you may have used before. The toolbars along the top are your save/import/export buttons, followed by alignment and zoom buttons. The toolbar along the left side, as in the image below, is where you select tools and add items, and the panel at the right is your one-stop shop for styling the items.



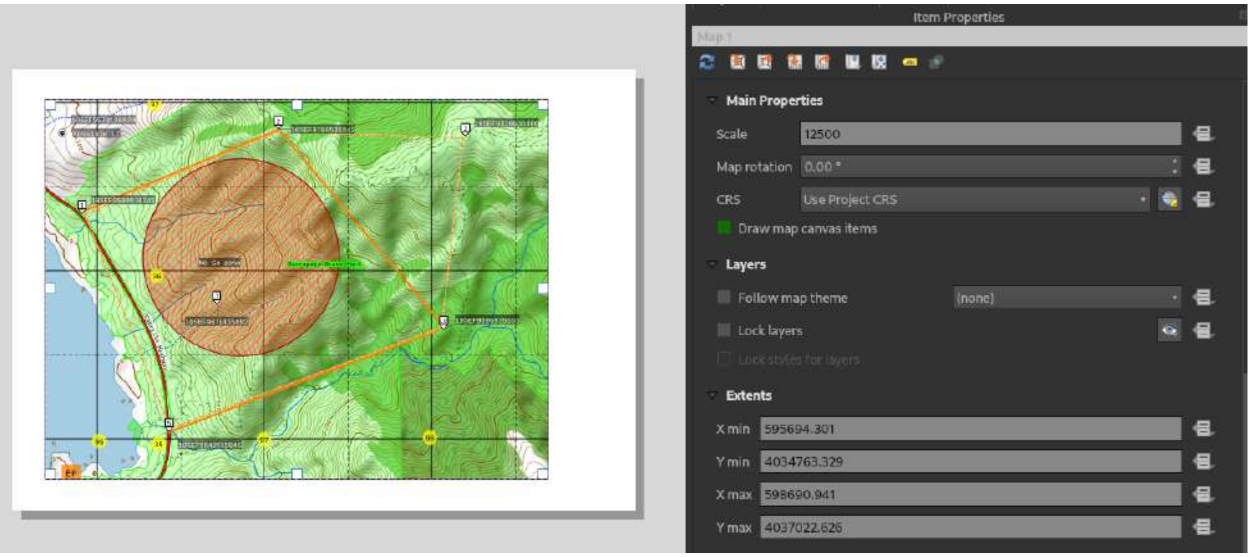
Right click on the blank canvas and choose `Page Properties` - from here, you'll notice that `Page Size` appears under the `Item Properties` tab on the right panel. (If you don't have this panel, go to `View > Panels > Item Properties Panel`.) Pick a paper size you prefer. If you're in the United States, letter paper is your best bet unless you have access to a larger printer; elsewhere in the world, A4 paper is the most common. It's important to size correctly, or you may not be able to print all the map you need, or your printer may refuse to print at the correct scale.

The first step, of course, is to add our map view.

Hit `Add Map` (the icon with the left side partly rolled up, sixth fro the top) on the left toolbar, then drag on the canvas to draw the map. (Unlike drawing shapes in editing a layer, simply click, drag, and release, no right-click needed.) After a brief interlude where the program indicates it's rendering, you should see your map appear in the shape you just drew.

Under the `Item Properties` tab on the right, you can set the scale. I recommend doing this first, as this will determine. If you're planning to use a standard military protractor to navigate with this, I highly recommend setting the scale to 1:25,000, 1:50,000, or 1:12,500. (Remember, for QGIS's purposes, just put in the number after the 1: in anywhere it asks for a scale.)

The third button from the right at the top of the Item Properties tab, `Interactively Edit Map Extent`, will let you pan around the map to get the area you want in sight. Click that, then click on your map in the layout and drag to get the location you want. (You can also click the same tool, fourth from the top, on the toolbar at the far left.) Make sure that if you zoom in or out, you change the scale to an appropriate value afterward.



Item Properties

Map 1

Main Properties

Scale: 12500

Map rotation: 0.00 °

CRS: Use Project CRS

🟩 Draw map canvas items

Layers

□ Follow map theme: (none)

□ Lock layers

□ Lock styles for layers

Extents

X min: 595694.301

Y min: 4034763.329

X max: 598690.941

Y max: 4037022.626

The `Select/Move Item` tool, third from the top in the side toolbar, will let us adjust the size of the map to make room for titles and marginal information.

`Add Label`, the both with a T inside on the left toolbar, will let us add text labels of any kind to the map. Like the map, just click and drag in the space you want the text to take up.

In the item properties panel, the top box will let us change the text itself; under `Appearance` there, we can click on the `Font` button to change the font (including the font size), the alignment of the text, and so on. One feature I recommend making use of is the dynamic text functionality.

Under the title of the map, I've chosen to add a note about the scale of the map. This is highly recommended, so you or others can instantly identify the appropriate protractor to use. Click the `Dynamic text` button under the text box, and pick `Map Properties > Map 1 > Scale`. The appropriate expression will be inserted right next to the cursor in the text box, and you can edit the expression or add text to either side, and even edit the expression to round a spurious 12499.9999, for example, to 12500; for me, this looks like:

```
Example Topographic Map Tutorial
Scale 1:[%format_number(item_variables('Map 1')['map_scale'], 0)%]
```

This will automatically change if the map scale changes.

Let's add a scale and north-seeking arrow to the map, as well, to aid navigation. Luckily, both are built-in features, fourth and fifth from the bottom of the left toolbar. Click and drag over the map in the corner to add them; they'll automatically link to our map.

Do the scale first. You'll notice that by default, it only delineates 1 km; this can be adjusted in the Item Properties tab in the panel to the right. Under `Segments`, you can pick the number of segments to the left and right of 0, and next to `Fixed width`, you can pick the distance increment of your choice. (I typically go with 100, 200, or 500 meters, depending on the scale.)

Declination

The north-seeking arrow requires another detour, though, on the topic of magnetic declination. If you're familiar with land navigation, you know what this is; if not, the basic version is that the lines on your map represent **grid north**, often represented with a "GN" or a "y" indicator.

This is in contrast to **magnetic north** - the direction a compass points at a particular location on the Earth's surface, often represented with a half arrow - and **true north**, the actual direction to the North Pole, often represented with a line capped by a star. These are not the same, and magnetic and grid north change relative to each other over time and as you move across the Earth's surface. Unless you're navigating over a huge area, though, we can reasonably assume that distances traveled on foot not as part of some huge nation-spanning vision quest will have more or less the same magnetic north for each grid zone covered.

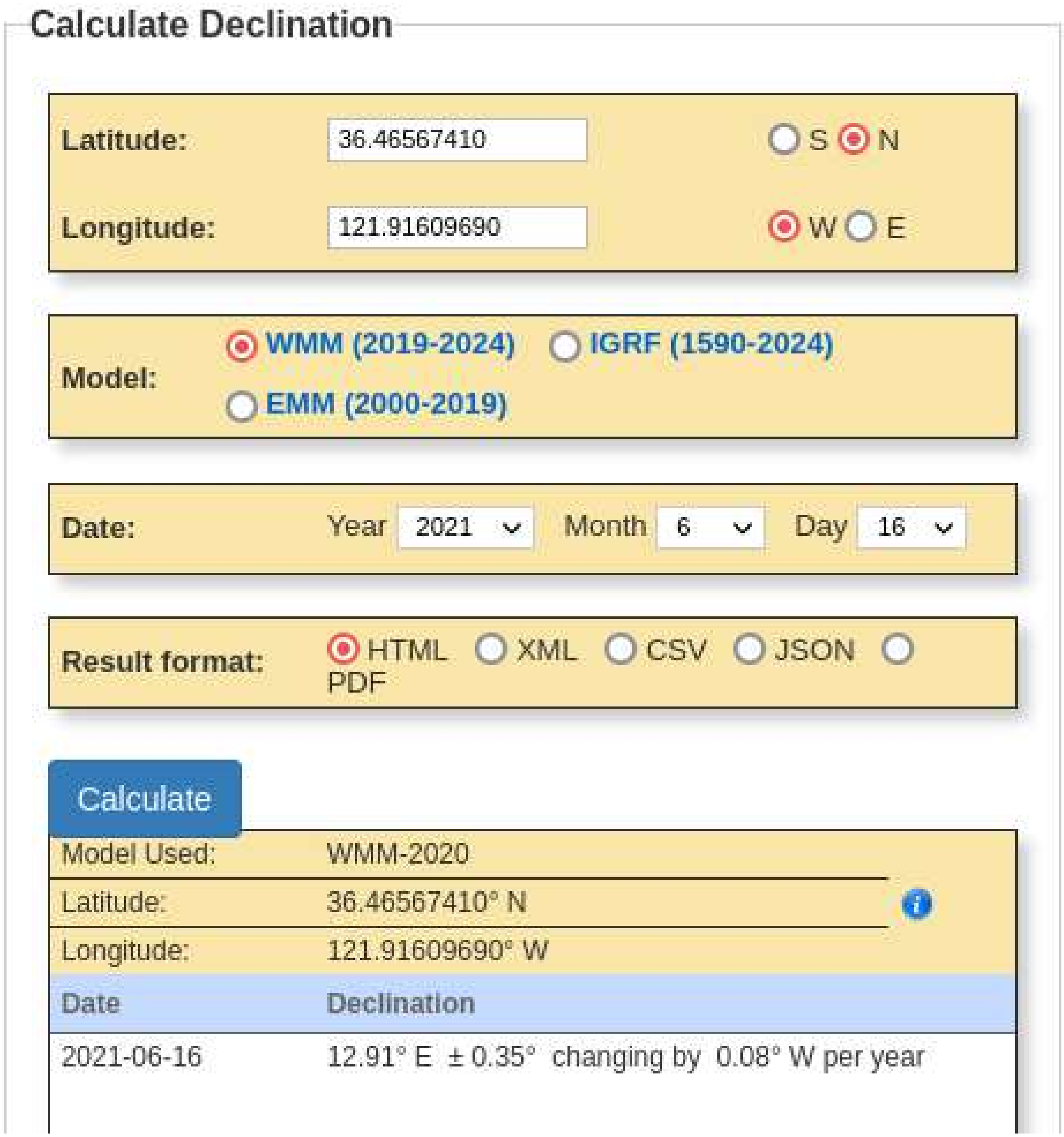
We still need to annotate grid and magnetic north, at least, on our map - points plotted on the map, and azimuths taken, will be in degrees (or mils^[If you don't know what these are, you don't need to know what these are, thankfully.]) grid, not degrees magnetic. We need to have a way to read the map and identify how to convert between grid and magnetic angles, in order to appropriately navigate.

There's a way to calculate magnetic declination on the fly within QGIS; however, it involves a bit of Python programming and the installation of additional packages, which isn't something that's really within the scope of a relatively simple guide like this. As a result, I recommend using the declination calculator on NOAA's site:

https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml

Remember from earlier that we can use Lat Lon Tools to capture a coordinate; under the Lat Lon Tools toolbar or menu (`Lat Lon Tools > Coordinate Conversion`), click the upper-right button on the dialog that pops up (`Capture coordinates by clicking on map`). Click in the center of your map.

Once you have the latitude and longitude in decimal degrees (at the very top of the dialog), input it into the converter page. If either the latitude or longitude are negative, you'll need to make sure the value is set to South or West, respectively.



Latitude: 36.46567410 ○ S ● N

Longitude: 121.91609690 ● W ○ E

Model: ● WMM (2019-2024) ○ IGRF (1590-2024) ○ EMM (2000-2019)

Date: Year 2021 Month 6 Day 16

Result format: ● HTML ○ XML ○ CSV ○ JSON ○ PDF

Model Used:	WMM-2020
Latitude:	36.46567410° N
Longitude:	121.91609690° W
Date	**Declination**
2021-06-16	12.91° E ± 0.35° changing by 0.08° W per year

Make sure to take note of the change, too - this is valuable information to put on the map in case we need to use it in the future.

Leave everything else as default and click `Calculate` to get your result. Note that the resultant value, made negative if it's west (i.e. 5 degrees east = 5 degrees; 5 degrees west = -5 degrees), is what we **add** to go from a **magnetic azimuth** to a **true azimuth**. If our declination is 5 degrees and our compass says 10 degrees, for example, we'd add 5 degrees to get the true bearing, 15 degrees. If the declination is -5 degrees, we add 10+-5 = 5 degrees for the true bearing.

We do the opposite to go from azimuths plotted on a true-north map - so if our map and protractor in this example say 10 degrees and our magnetic declination is 5 degrees, we subtract 5 to get a magnetic bearing of 5 degrees.

Now, this is the angle between true north and magnetic north - **not** between grid north and magnetic north.

There's one more step we have to do to get the grid convergence; think of this as the same thing for magnetic declination, but for the grid lines and true north.

The grid convergence can be found for your map by pasting this into a label field on the layout:

```
[%with_variable('GtT',
-with_variable('raw_GtT',
with_variable('PT',
    transform(centroid(item_variables('Map 1')['map_extent']),
        @project_crs,'EPSG:4326'),
    with_variable('displacedPT',translate(@PT,0,0.001),
    azimuth(centroid(item_variables('Map 1')['map_extent']),
    transform(@displacedPT,'EPSG:4326',@project_crs))
))/pi()*180.0,
if(@raw_GtT > 180, '-' || (360-@raw_GtT), @raw_GtT)),
if(@GtT > 0, '+', '-') || format_number(@GtT, 1) || ' degrees'
)%]
```

In sum, this takes the center of the map and converts that point into the universal WGS-84 CRS, and then takes a point a short distance north of it. It then converts that projected point back into the original CRS, and calculates the angle between the line formed from the original point to the projected point, and the grid north line. That angle is returned in radians, and to finish it off the expression converts that angle into degrees and rounds it to one decimal place.

For my example map, my magnetic-to-true angle is about 12.82 degrees. This expression gives me a grid-to-true angle of about 0.6 degrees.

Once we have those values, we use the following formula to get our GM angle:

`(GT - MT) = GM`

In this case, MT is our magnetic-to-true angle (the magnetic declination), and GT is our grid convergence angle.

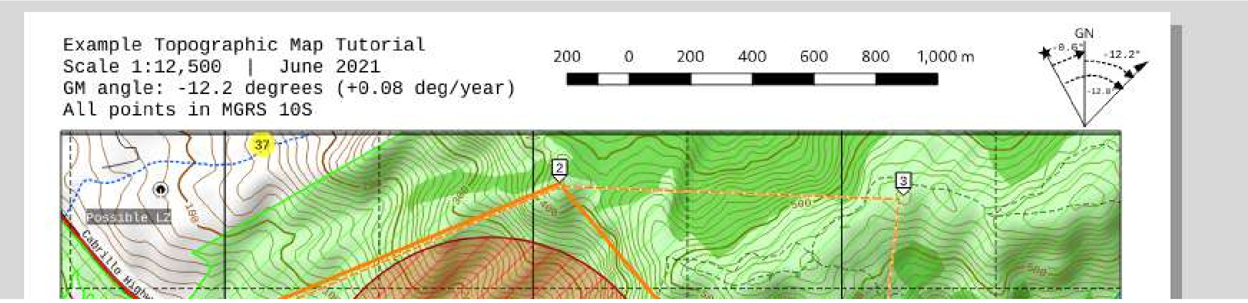
In this example, my GM angle is `(0.6-12.82) = -12.22` degrees. To convert a grid azimuth to a magnetic azimuth, I add this value to the grid azimuth, and subtract it from a magnetic azimuth to get a grid azimuth.

Now, given that my intended use case is navigating with a standard 3H compass marked in degrees, not firing artillery and using an M2 compass in mils, it's highly unlikely that I'll need more than a degree's worth of precision. We can safely round to the nearest degree, making the GM angle -12 degrees for this map.

Again, there are ways to put this on the map and have it update automatically, but for our purposes, it's much easier and faster to do the appropriate calculations and put in manually in a label, just like we did with the title.

Now we can add a north-seeking arrow to our map. This is easy - go to the left bar and click `Add North Arrow` (the icon with the arrow on it, usually fifth from the bottom). Click and drag it into a space on the map; in the panel on the right, you can select the arrow icon of your choosing. The end result is nearly finished; you can see it on the map below.

(Those familiar with military maps might notice you can easily construct a declination diagram itself using the `Add Arrow` and `Add Shape` tools instead. I've opted to do this on the map below.)



Example Topographic Map Tutorial
Scale 1:12,500 | June 2021
GM angle: -12.2 degrees (+0.08 deg/year)
All points in MGRS 10S

Multiple pages and legends

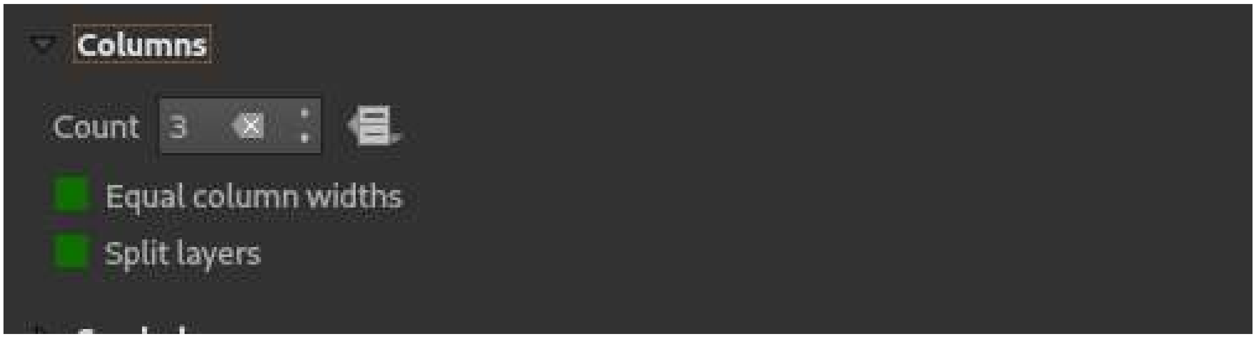
There are a lot of additional types of marginal information you can add, all using the basic building blocks the layout editor makes available to us, from noting multiple maps in a series, indicating connecting maps with labels to each side, putting serial numbers on the maps, and so on. Right now, though, we have all the information we need to successfully navigate using the map, and all that's left is a little bit of additional useful information.

By this point, you probably have a pretty good idea of the conventions of military or topographic maps, but may need a bit of a refresher on what you had in mind when you made the map at some point, or may need to work with someone less proficient. For this reason, it's never a bad idea to add a key.

Let's add it on the back of the map. Go to `Layout > Add Pages` and pick the correct size paper in the dialog that appears. A new page will appear in your layout. Click on it to ensure that new items are added on this, instead of the previous page.

Add a legend with `Add Item > Add Legend`, then drag on the second page for the area you want the legend to cover. You'll immediately notice that there may be items on this legend you don't want, or that it may be longer than you can fit on the page.

To make it fit, simply go to `Columns` under `Item Properties` and increase the number to fit on the page; I recommend also checking `Equal column widths`, as below, to ensure the symbols don't overlap, and `Split layers`, to make sure symbols are kept with their appropriate grouping on the layer.



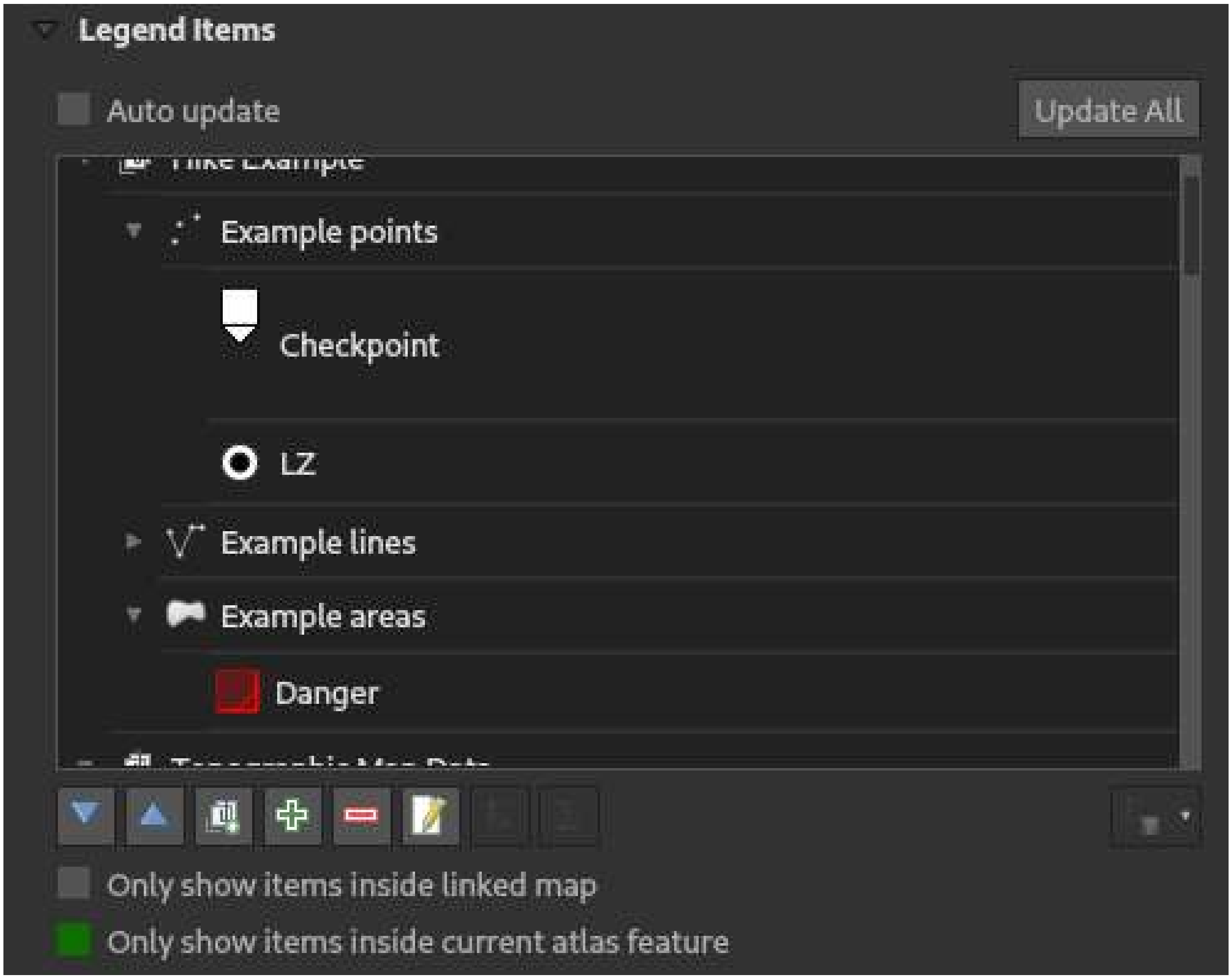
Columns

Count 3

☑ Equal column widths

☑ Split layers

Under `Legend Items`, we can click `Only show items on linked map`, which will only display items that are actually present within the map. This can be helpful with a large number of symbols. If we want to delete specific symbols, however, we can uncheck `Auto update`, and manually remove/add items from the legend. This won't erase the underlying data, and won't hide the items from the map view in the main QGIS window; it only makes it disappear from the legend.



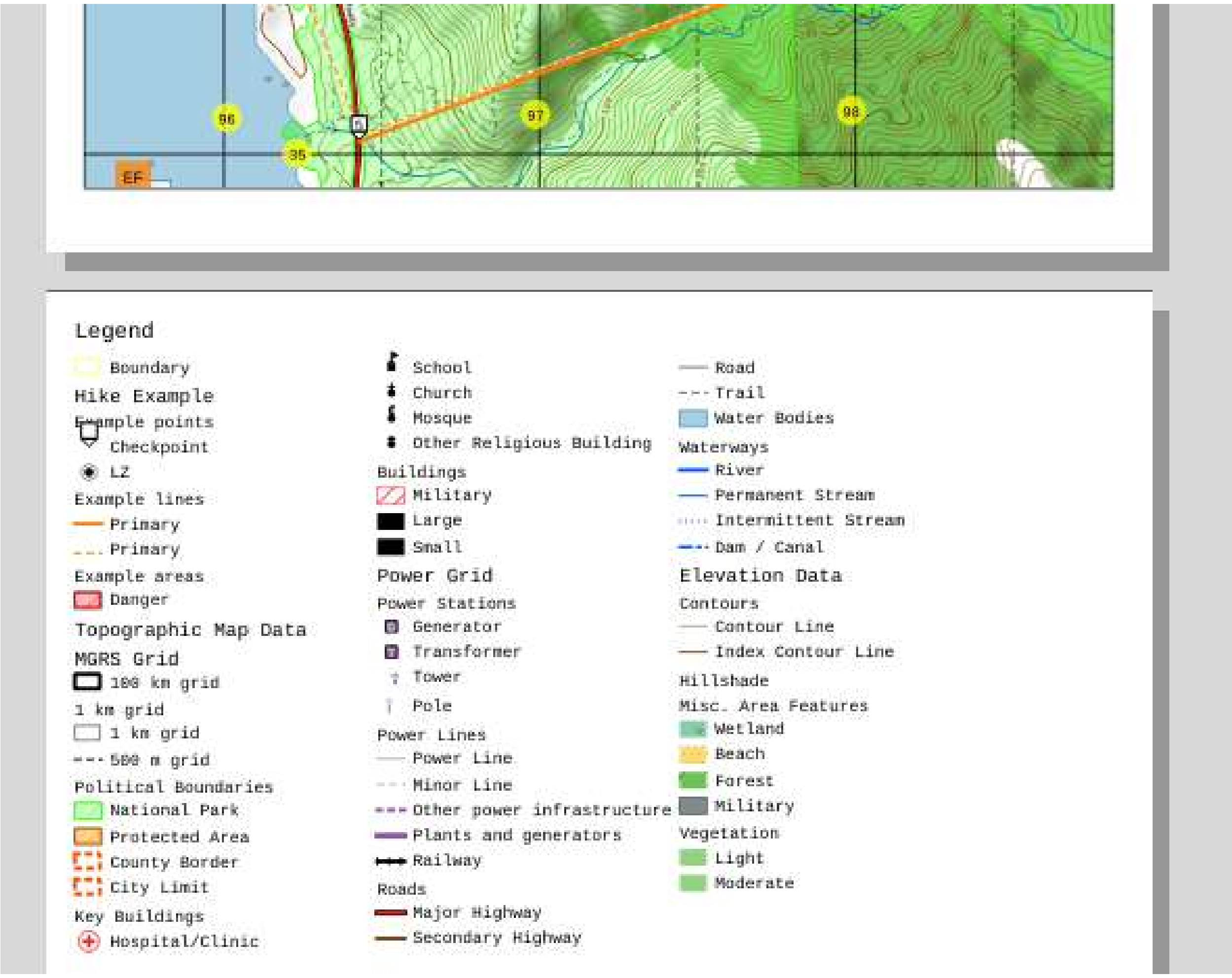
☐ Auto update

- Hike Example
  - Example points
    - Checkpoint
    - LZ
  - Example lines
  - Example areas
    - 🟥 Danger
- Topographic Map Data

☐ Only show items inside linked map

☑ Only show items inside current atlas feature

Other items in the `Item Properties` tab for legends let you change the fonts, adjust the display, remove the default white background, and other features that are largely self-explanatory. With a few more tweaks, we have a complete legend on the back:



Export and printing

We're finally done with mapmaking; the next step is to export it, or print it directly from QGIS.

Since the number of possible printer setups one can have is near-infinite, I won't go into exhaustive detail on the process, but instead leave a few notes of relevance:

- When exporting to share the map, I highly recommend using the PDF export option (`Layout > Export as PDF`). Exporting as an image will strip your map of the paper sizing and may make it incredibly difficult to print at the appropriate scale for using with a protractor. I recommend, if file size isn't an issue, to check `Always export as vectors` and set `Text export` to always export text as paths, to prevent issues with different computers having different font rendering mechanisms, or for a part of the map with key detail be blurred out by turning the map into a raster image. The option to simplify geometry can usually be left on, though.
- When printing, it's **critical** that you ensure the page is printed without any scaling. A lot of printers or programs will automatically scale the print job to fit on the paper provided. If you want to use this with a protractor, you need to ensure that whatever "printer properties," "print settings," etc. dialogs you have are combed through too turn all these settings off. If you're getting a larger format map printed at a print shop, make it extremely clear to them that there can be no scaling (and even then, I don't recommend trusting it to them), or use a self-service kiosk.
- 300 dpi resolution is typically sufficient for printing. If you printer can go higher resolution, though, it may be worth it to ensure smaller text is still legible.

- You can turn on and off layers to alter what the map view sees; it'll automatically update based on what's available in the main map view. Keeping custom items on the map and grid lines, overlaid over a satellite map, makes a great B-side to a topographic map, giving you the best of both worlds on a single footprint.