Lab Assignment #6 – Unsupervised Land Cover Classification

1. Logistics

Date assigned: Monday, December 02, 2024

Date due: Monday, January 13, 2025 (via Moodle)

Points: 90 points

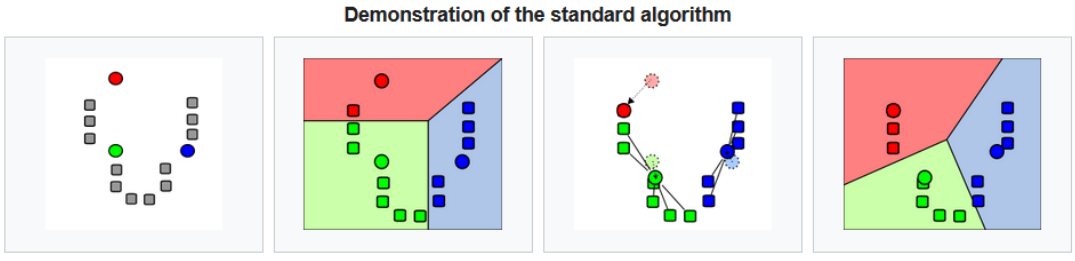
Please submit all homework as a single PDF file via Moodle.

In this lab, we will work with optical remote sensing data to look at unsupervised classification techniques.

2. Introduction to Unsupervised Classification

Unsupervised classification attempts to break an image (or dataset) into a *defined number of classes* using only information contained in the dataset – so, no prior information about the classes. This differs from **Supervised Classification**, which we will discuss in the next lab, where you provide *training data* about which areas/data should be included into which class.

In Lab 3, we attempted one simple way of doing unsupervised classification with SNAP. SNAP implements two unsupervised clustering algorithms: *k-means* and *Expectation Maximization*, which is an extension of the k-means algorithm. **K-means** is one of the most common unsupervised classification algorithms, and is implemented in many different software packages and used in many different applications. It operates by iteratively assigning data to clusters based on *the distance to the cluster centroid*. During each iteration, the cluster centers are updated, and each data point is re-assigned to the new nearest cluster centroid. For example:



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Via Wikipedia: https://en.wikipedia.org/wiki/K-means_clustering

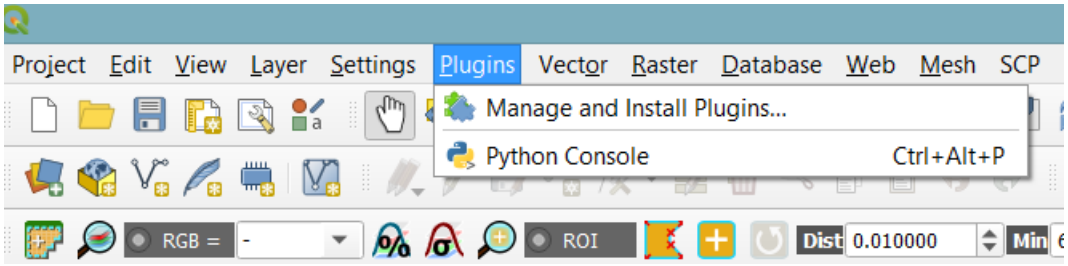
K-means is so widely used because it is FAST – it is able to handle very large amounts of data quite efficiently, which is essential for Remote Sensing applications.

QGIS implements two unsupervised classification algorithms – *k-means and ISODATA* (https://en.wikipedia.org/wiki/Multispectral_pattern_recognition). **ISODATA** differs from *k-means* in that it splits classes that are too variable and combines classes that are too similar or too small. This means that *ISODATA* may not return the same number of classes as was requested, whereas *k-means* always will.

While both classifiers are useful, they have certain drawbacks which will become apparent as we work through the lab. The three largest issues from a remote-sensing perspective are: **(1) cluster initial centers have a large impact on final results, (2) clusters of different shapes/diversities (e.g., desert vs forest) are not equally well-defined, and (3) we do not know what the clusters represent without further interpretation.**

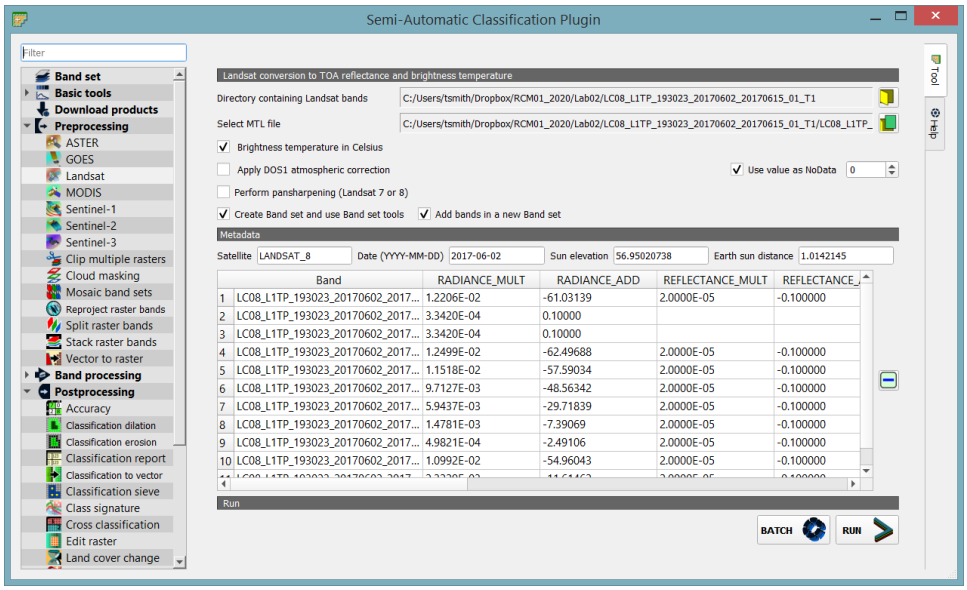
3. Loading the Data into QGIS

To load the data into QGIS, we will use the **Semi-Automatic Classification Plugin**. If this is not yet installed on your system, you can install it via the 'Plugins' menu:



In this lab we will use **Landsat Top-of-Atmosphere Corrected Data**, which can either be the data you created in **Lab 2**, or fresh data you choose to use for this lab. In this manual, will use the data from **Berlin** that was used in Lab 2, but feel free to use your own data if you prefer.

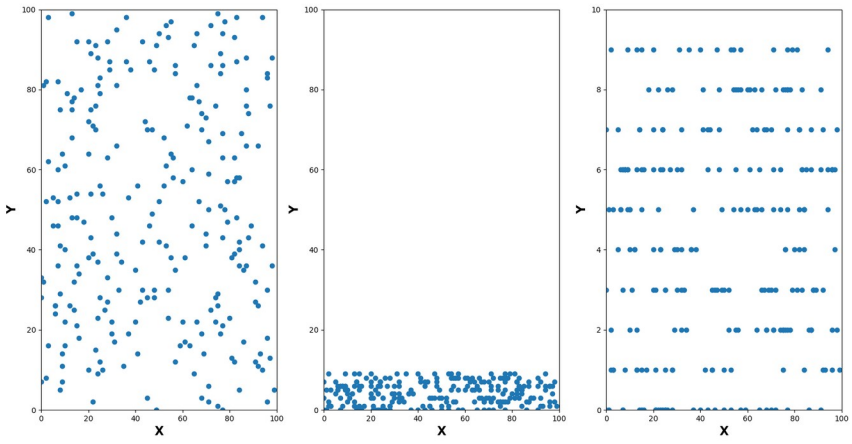
First, extract the *LC08_L1TP_193023_20170602_20170615_01_T1.tar.gz* zipfile if you have not done so already. If you do not have the corrected data already from Lab 2, you can quickly re-create it using the **SCP Plugin**. Under '**Preprocessing**', there is a tool for Landsat correction. Simply point the tool to the directory where the Landsat Data is stored and the Metadata file. This tool will then correct the most relevant bands for Landcover classification:



This will take a few minutes to run, but will output all of the corrected bands into one place. **It is important to note that these bands are all scaled on the same range (0-1).** Why is this important?

Side note: Why do we need to standardize/normalize data?

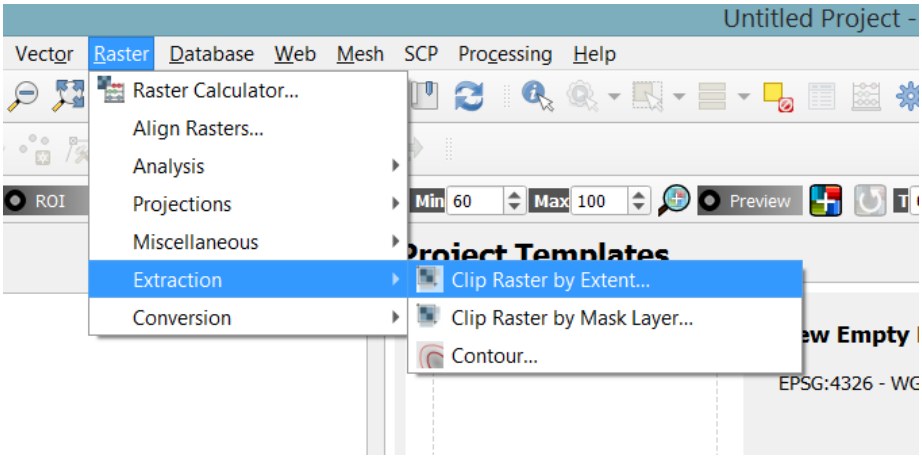
In most classification problems, you take several datasets together with the goal of yielding one single dataset which has **class labels**. In our simple case, our input datasets are different Landsat Bands. However, you might want to include other things than the raw Landsat bands – for example, an NDVI ratio, an elevation dataset, or radar amplitudes. If those datasets do not have the same **scale**, the classification will not be **evenly weighted**. Consider, for example, a scatter of x/y points comprising two datasets:



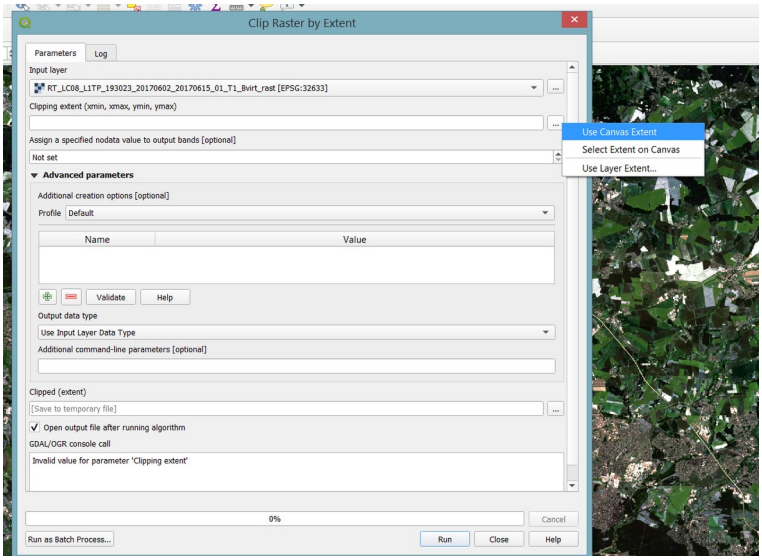
The dataset on the left has x and y values running from 0-100. On the right two panels, x still runs from 0-100, but y only runs from 0-10. Essentially, **it will be much harder to find breaks between y-values than between x-values, since the data range is much smaller.** If this principle is extended to three or four or ten dimensions, like we do with remote sensing images, you can very easily end up with poorly formed classifications.

On the flip side, you can also use this idea to **give different weights to different bands** in your classification, if you want to emphasize or de-emphasize a particular feature/band. However, in most cases you want to have **equally weighted data across all inputs**.

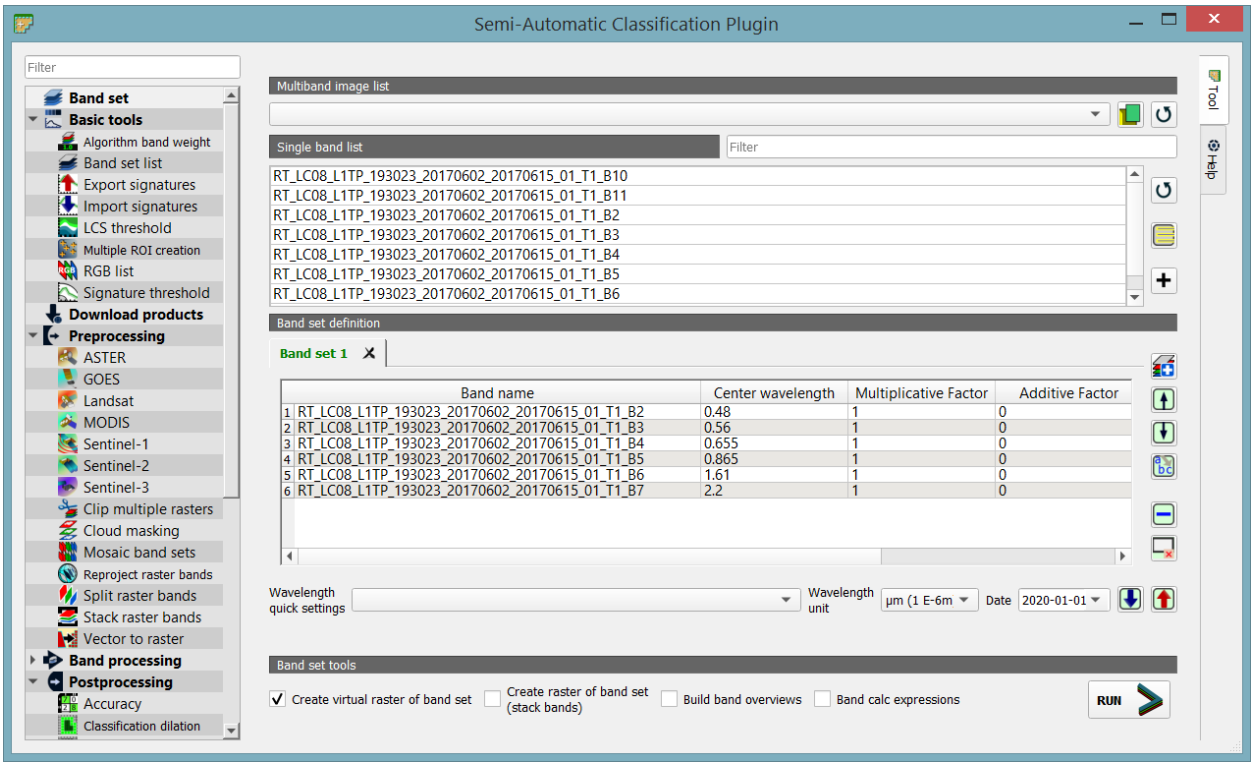
Hopefully our corrected Landsat data is now ready. Since the unsupervised clustering algorithms can take a while to run over large datasets, we are going to clip down our dataset to a smaller region of interest. This can be done via 'Raster -> Extraction -> Clip Raster by Extent' or Clip Raster by Mask Layer.



If you zoom into a region of interest, you can clip by the 'Canvas Extent':



We can load the data into the **SCP Plugin** using the 'Band Set' tool:

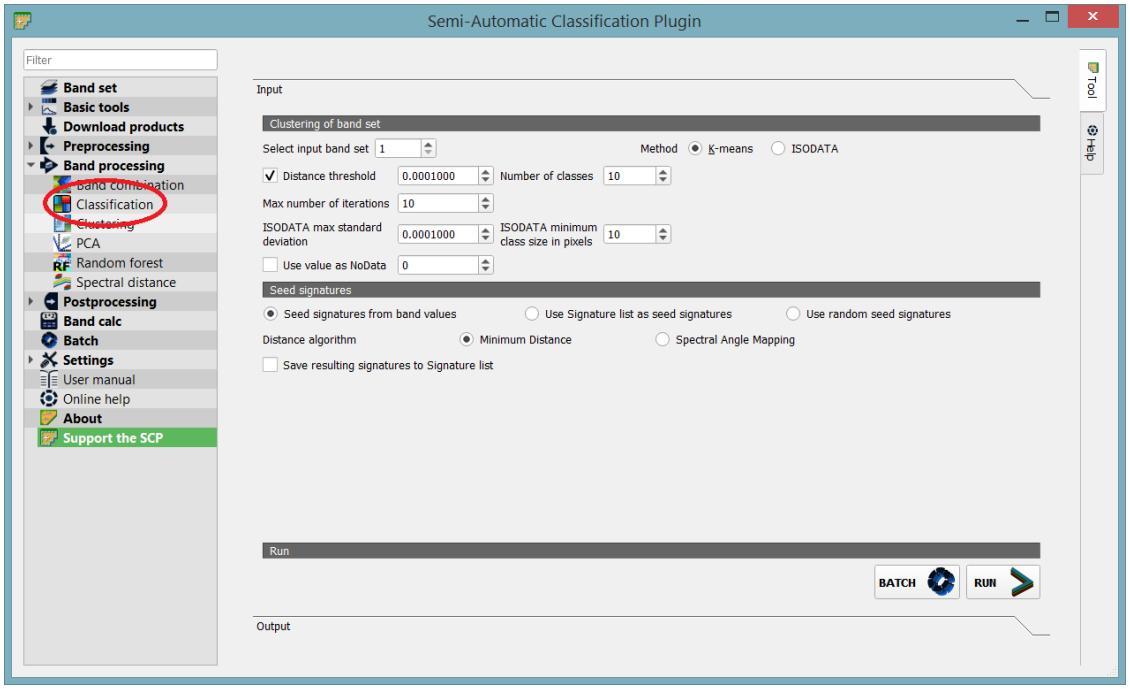


If you've just run your image correction routine, this should already be set. Click the **Create Virtual Raster of Band Set** box to export a virtual raster – this is a small metadata file that lets you operate on and visualize all of the individual band files as if they were a single image file.

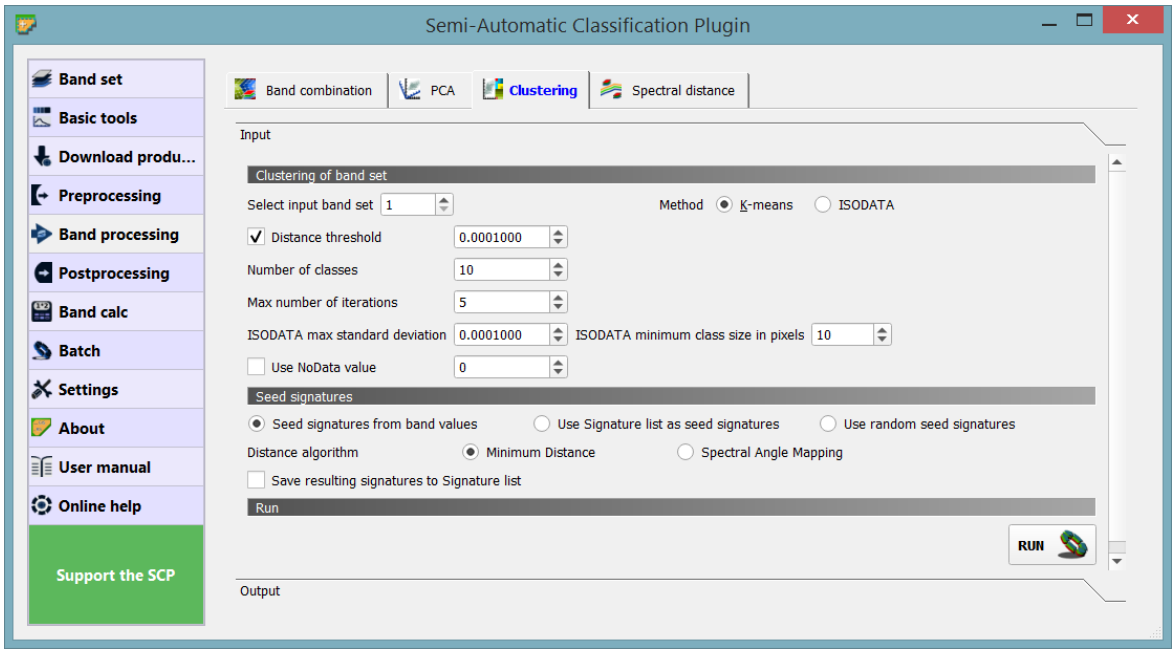
Now that we have our bands set, we can start doing classification.

4. Using Unsupervised Learning

Let's take a look at the classification options we have under the **Classification** tool:



If you are using an older version of the plugin, it will look like this:



As mentioned before, we have two algorithms to choose from: **k-means** and **ISODATA**. These two algorithms have several options to choose from, but the most important is the **number of classes**. Essentially, both algorithms will try to divide the spectral distribution of our data into exactly that number of classes in the best way possible.

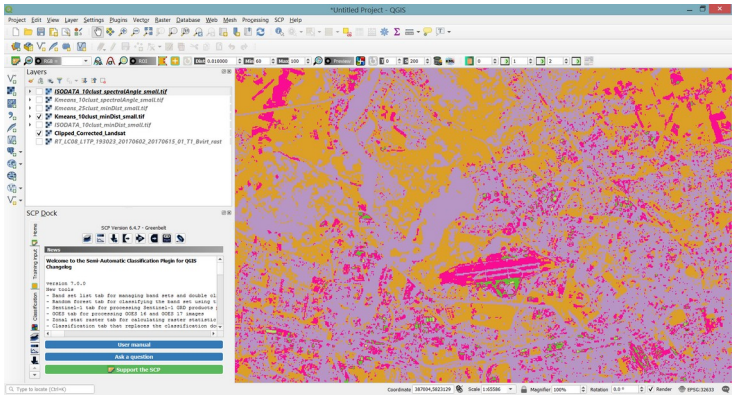
Let's try running it with the default settings (**k-means, 10 classes, minimum distance algorithm**) **EXCEPT FOR THE NUMBER OF ITERATIONS**. Please set this to **5** for your initial test to speed things up. Also remember to give your output file a useful name so you know what parameters you used!

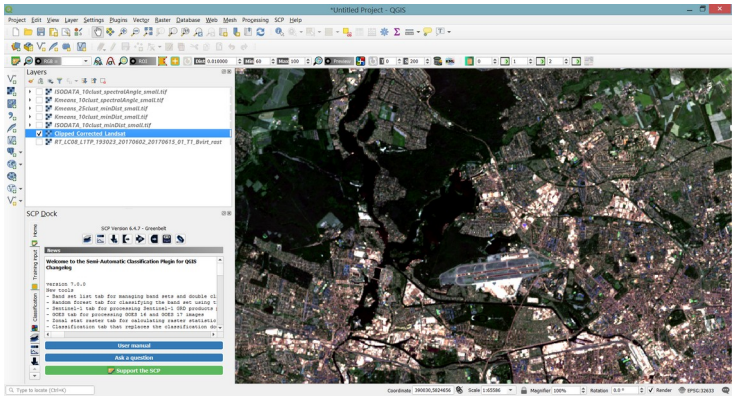
I called my output **Kmeans_10Clust_MinDist.tif**

This should only take a few minutes to run. When it finishes, let's take a look at our output. It is important to note that **more iterations will take more time**. The time can also be shortened by raising the **Distance threshold** at which *k-means* decides that the clusters are 'different enough'.

QGIS will tell you the estimated amount of time the tool will take to run (on the top of the screen). If it looks like this will take a long time, you can also stop the classification, clip out a smaller subset of your data, and run the tool over a smaller area.

Let's take a look at some initial results:





It looks like the algorithm is doing a very bad job of separating classes – I see water and urban areas being very strongly mixed up – the big lake just north of Tegel Airport (middle of image) is the same color as many forest and urban areas nearby.

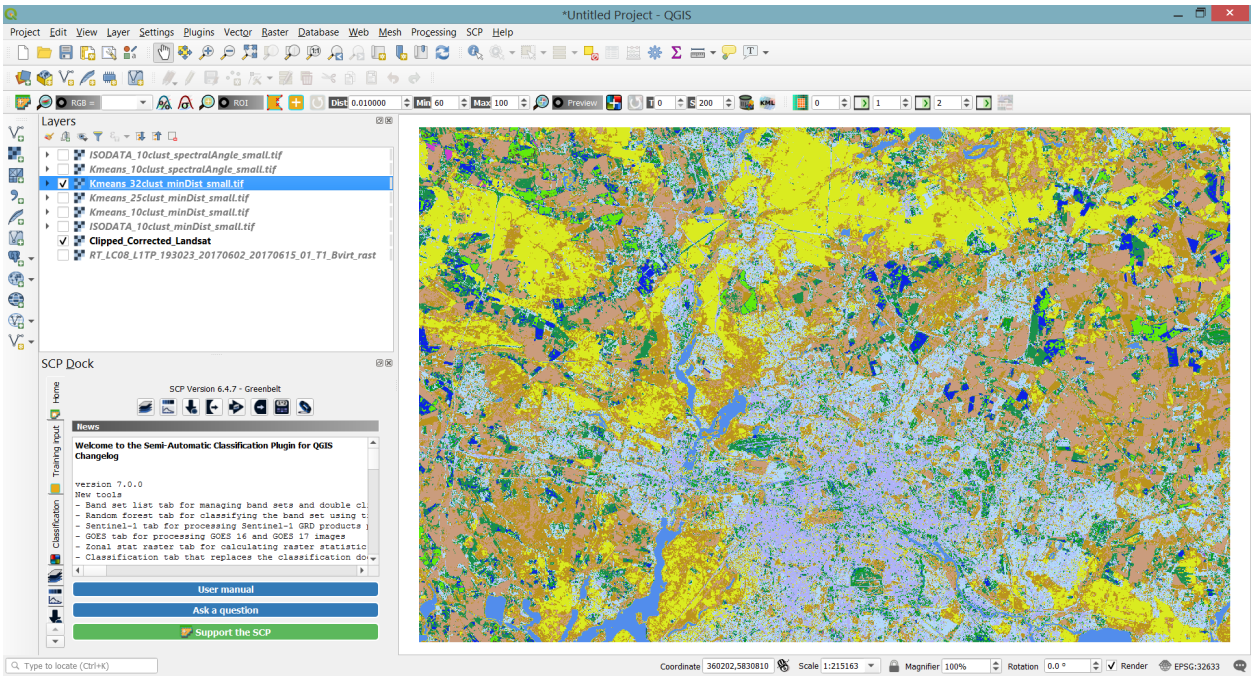
Repeat the classification with different numbers of classes, algorithms, and distance metrics. What do you notice about the differences?

> **Question 1:** Create four maps showing different algorithms/numbers of classes/distance metrics. How do they differ? Where are they similar? Which features are well defined and which ones are messy? (20 points)

Is there an optimal number of classes to use? Generally, the rule of thumb is to use 5-6x as many classes in your unsupervised classification as you want to end up with. So, if I wanted water/urban/agriculture/forest as my final output, I would need at least 20 classes to start being able to separate things. It is always easier to combine classes afterwards than to split them once they are created.

5. Combining and Merging Classes

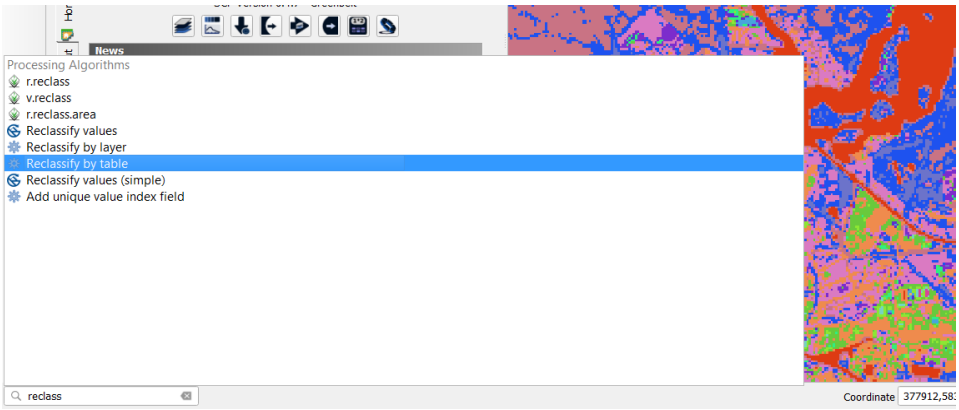
How can we best exploit the results of the unsupervised algorithms? Let's consider an output with **32 classes** created with **K-Means** and a **Minimum Distance** algorithm. I also let my processing run for **10 iterations**.



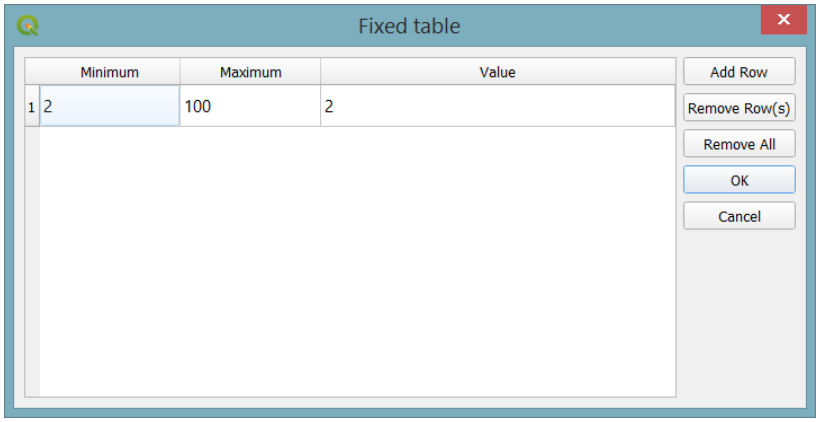
This is starting to look interesting – water seems to be well-defined into a single class, urban areas are generally also in one or two classes. Forests and other vegetation are rather harder, however. From this point on, we have to use our own

expert knowledge of the location (and our interpretations of the images) to turn this rough classification into something useful.

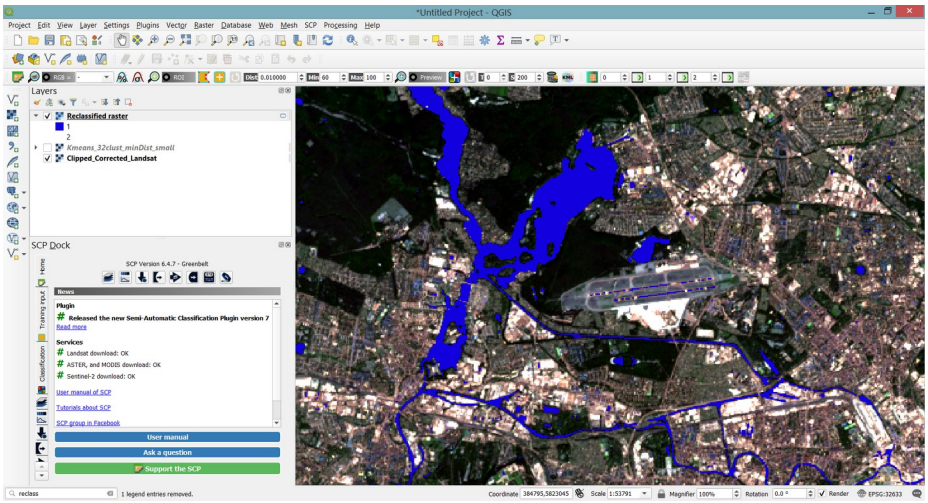
The main tool for doing so is **Reclassify by Table**.



Let’s try something simple – making a mask of only ‘Water’ pixels. For me, it looks like the value ‘1’ is most associated with water on my map. To isolate that, I will classify every value larger than 1 to an arbitrary new value:



Try running this. How well did it work? Overlay your result on your original image, with the water class in blue and the non-water class as transparent. Mine looks like this:



While some areas are wrong (water in the middle of Tegel's runway), many areas are quite well classified.

> **Question 2:** Try to come up with an 'Urban' class from your unsupervised classification. How many initial classes did you need? How many did you combine into your urban class? Please submit a map that is transparent non-urban areas (e.g., you can see the normal satellite colors), and a colored overlay for 'urban'. For reference, please see the figure above showing a water mask on top of the color image. Include a scale, north arrow, etc. (20 points)

6. Homework: Using your own Data

Choose your own area of interest. You may work with the data you downloaded for previous labs (e.g., NOT BERLIN) or use a new area of interest. You are also welcome to use a different data source than Landsat (e.g., Sentinel-1, Sentinel-2, MODIS) if you would like to test out classification results on other data sources. To receive full credit, please submit:

1. Four maps showing different unsupervised algorithms, class numbers, or distance metrics.
2. One map showing a simple mask overlay on your original data (e.g., only show water or snow or urban or one other land cover type of your choice). This should be visualized on top of your original True-Color data.
3. A final reclassified map which separates your image into 4-5 primary classes (for example, snow, water, urban, vegetation, agriculture, bare soil, river, sand, etc). This can be created by combining classes from your Unsupervised Classification results.
4. An explanation of which classes worked well, what challenges you faced in creating the final image, and whether or not you think the classification does a good job of describing your area.

Total points: 50