Geoinformatics | Course Remote Sensing (1)

Schmitt | Ulloa

Summer Semester 2020

Practice 7: Creation of maps and Pre-processing of raster data I

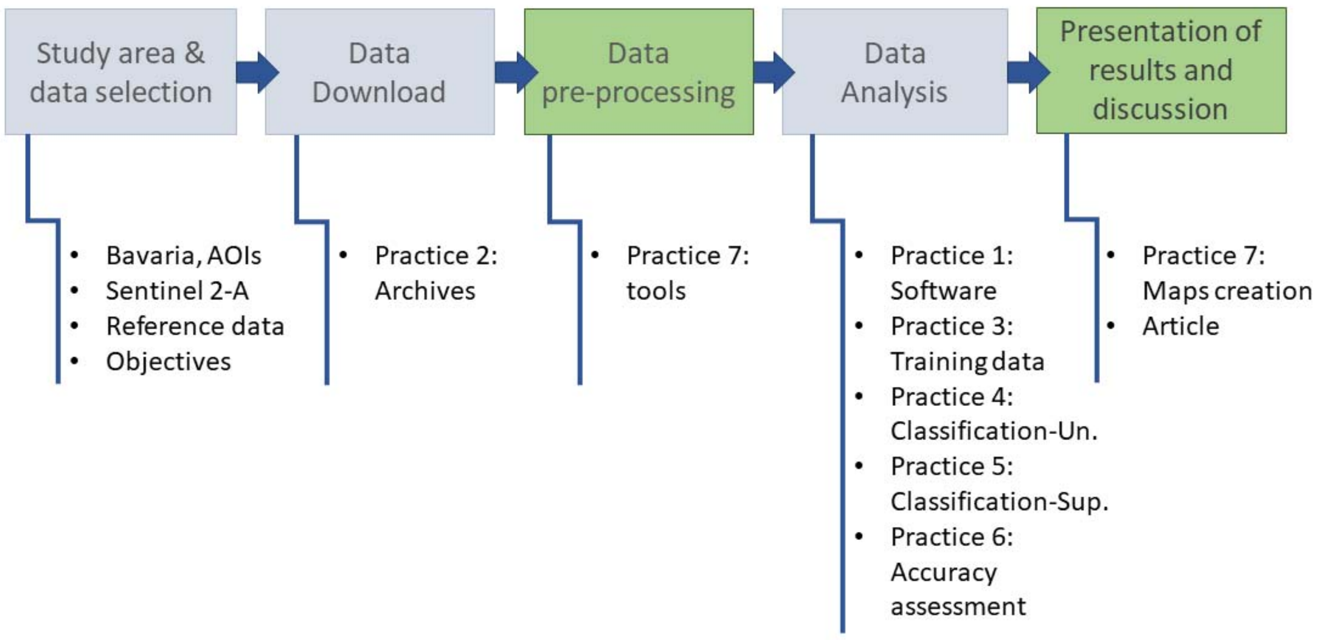
Overview

Objectives: Create a cartographic map using your classification GeoTIFF results. Explore some pre-processing functions for raster data.

Data: For this practice, use the following files:

- Raster files:
  - Sentinel 2-A subset images from previous practices (S2A_L2A_T32UPU_rstck_id1.tif, S2A_L2A_T32UPU_rstck_id2.tif or S2A_L2A_T32UPU_rstck_id3.tif)
  - Sentinel 2-A TCI subset, S2A_L2A_T32UPU_TCI.tif
  - Sentinel 2-A radiometric corrected bands 2, 3, 4, 8.
  - Classification: GeoTIFF raster with the classification result using either RF or MD.
- Vector file: shapefiles of the various AOIs from previous practices (id_1.shp, id_2.shp, id_3.shp)

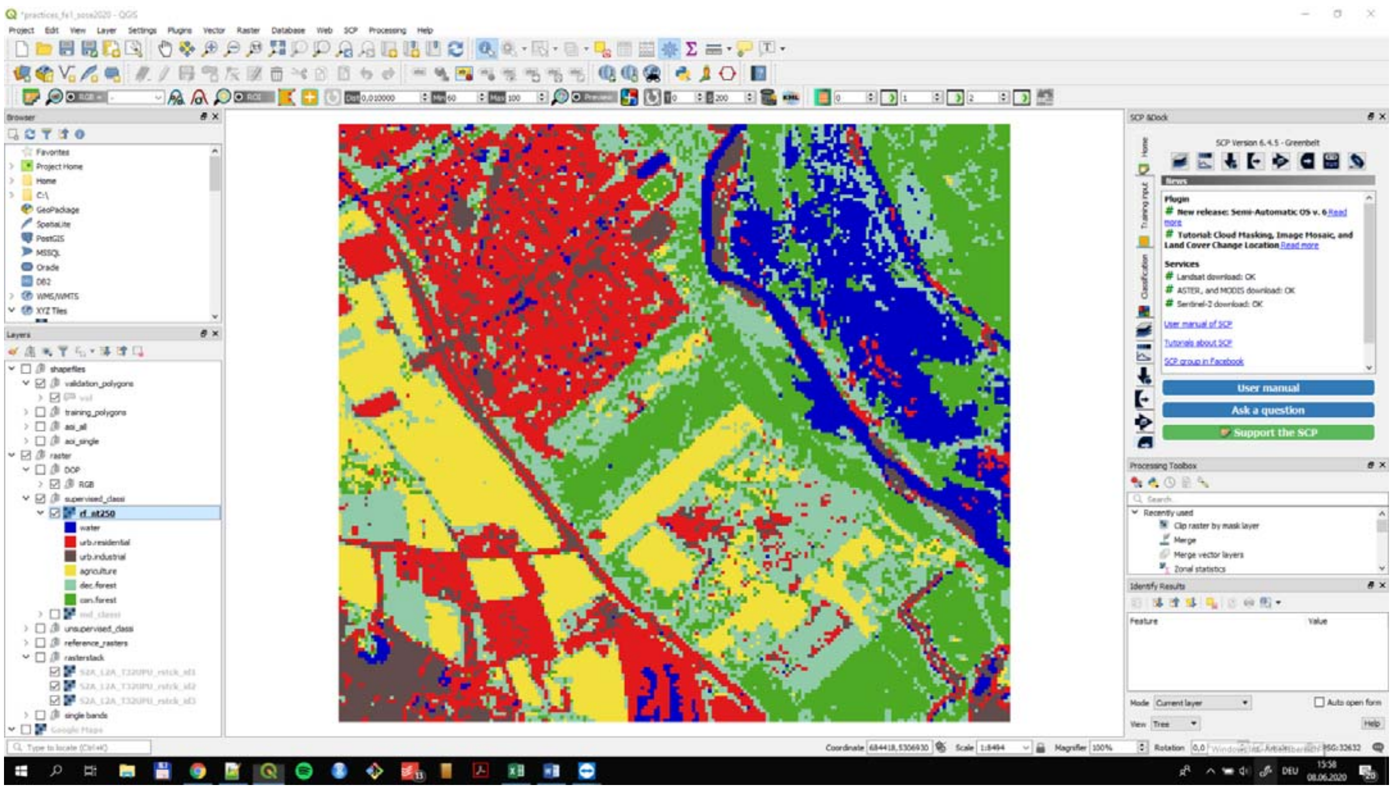
Tasks: load your classification raster in QGIS to create a map for presenting your results. In Python, use the diverse Sentinel-2A images and vector files to learn about some pre-processing functions, like crop, stacking, and data streching. These tools will also help you to present your data with the right properties in an article.


Study area & data selection
Data Download
Data pre-processing
Data Analysis
Presentation of results and discussion
• Bavaria, AOIs
• Sentinel 2-A
• Reference data
• Objectives
• Practice 2: Archives
• Practice 7: tools
• Practice 1: Software
• Practice 3: Training data
• Practice 4: Classification-Un.
• Practice 5: Classification-Sup.
• Practice 6: Accuracy assessment
• Practice 7: Maps creation
• Article

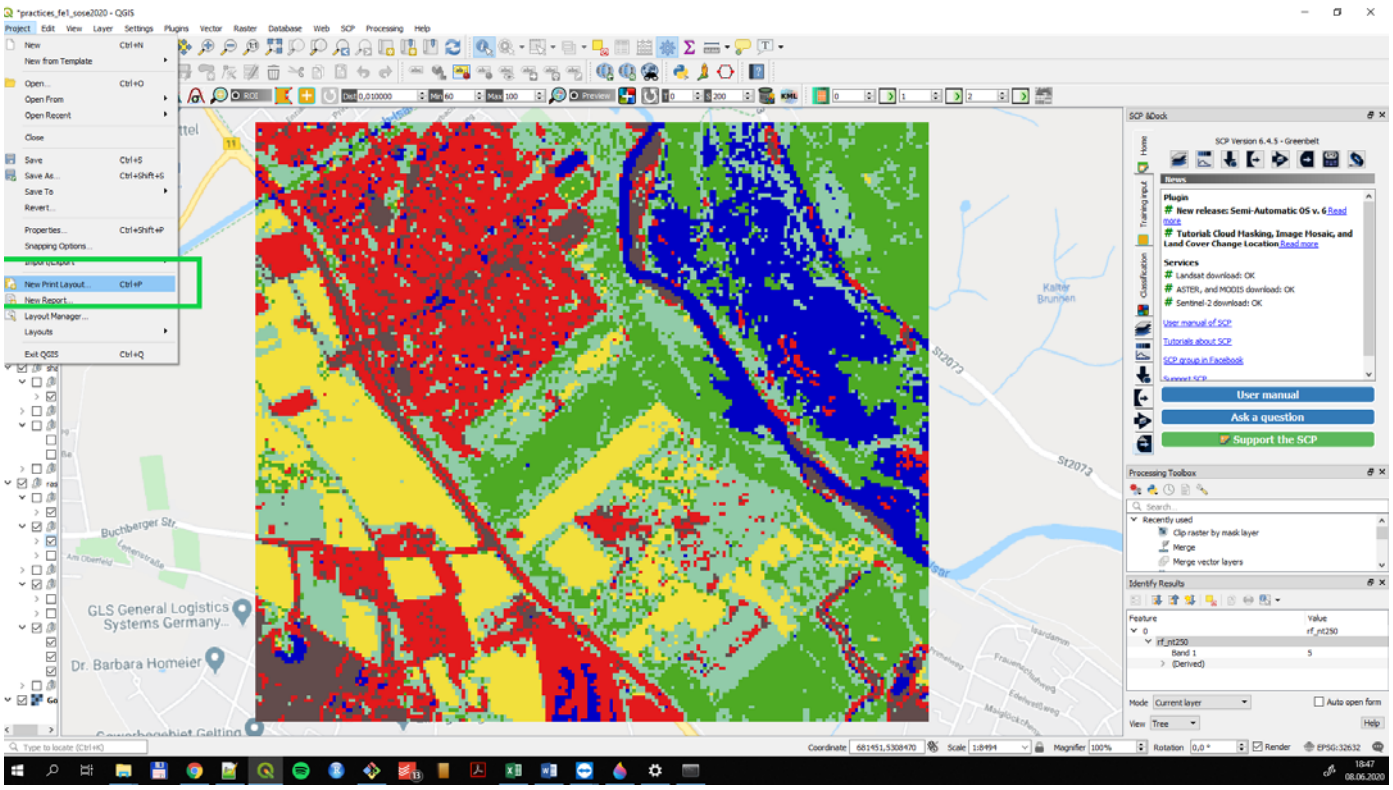
Procedure

In QGIS: create map

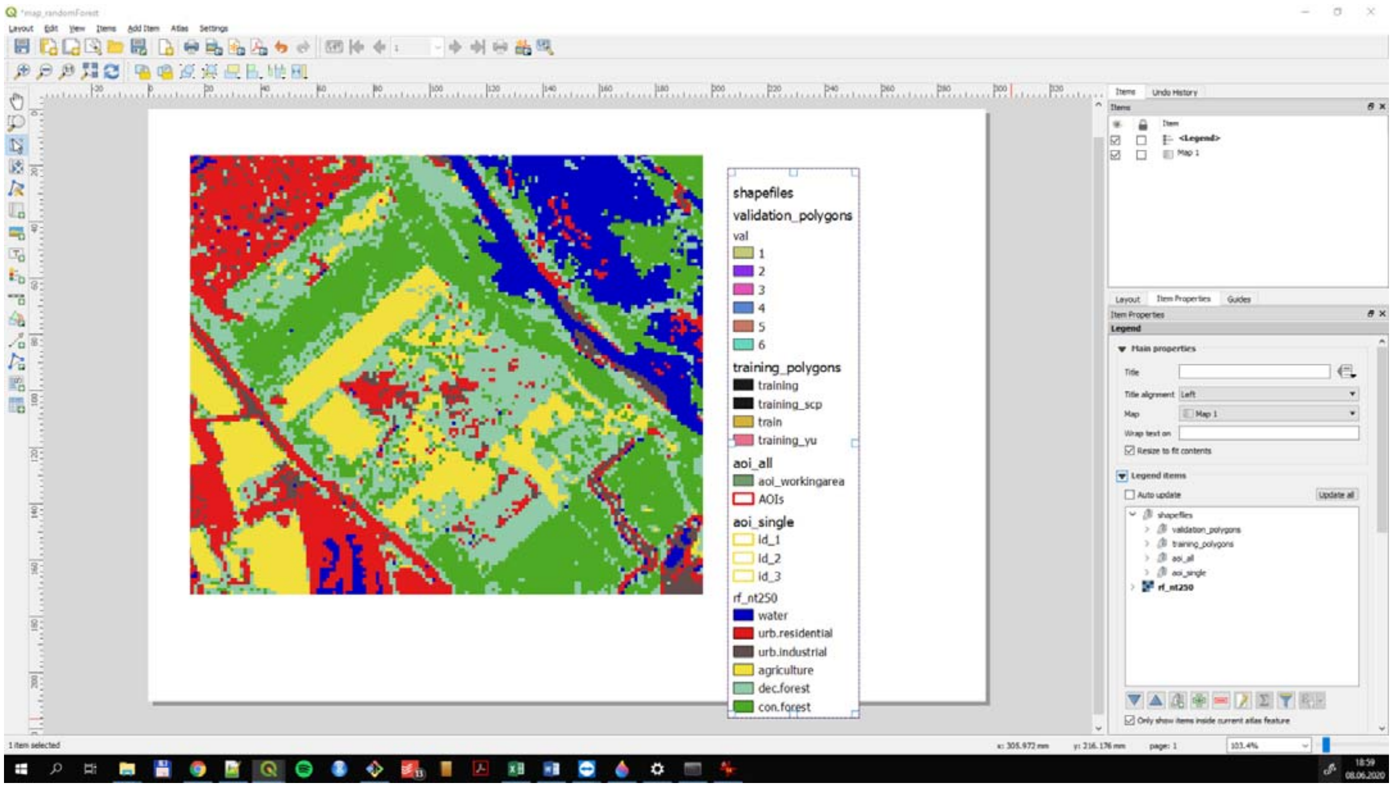
1. Open QGIS and load your classification raster file.



2. Under Project > NewPrintLayout you will open a new interface that will allow you to create a map. Give a name to your map.



3. You will find many tools that will help you with the map elaboration. Here you find a short summary of them:
   - Add Map > draw area of where your map will be included
   - Select/Move item > define size of the map and position
   - Move item content > move map inside of working area
   - Add Legend > Pay attention to Legend Properties. Select relevant layers. Change name to classification layer. Pay attention to "Auto upadate" feature. If you edit the names on the raster settings, it will be updated on the map as well.
   - Adds a new label to the layout > to add titles and subtiles.
   - Select map and Item Properties > Grids > + > Modify grid: coordinates, CRS, appearance, etc.
   - Add a new picture to the layout > for adding a new picture like the HM logo. On Picture properties select file. If you want to add a North Arrow, go to "Search Directories"
   - Add Map, and repeat for adding a second layer to the map
   - Layout > Export


*map_randomForest
Layout Edit View Items Add Item Atlas Settings
Items Undo History
<Legend>
Map 1
Layout Item Properties Guides

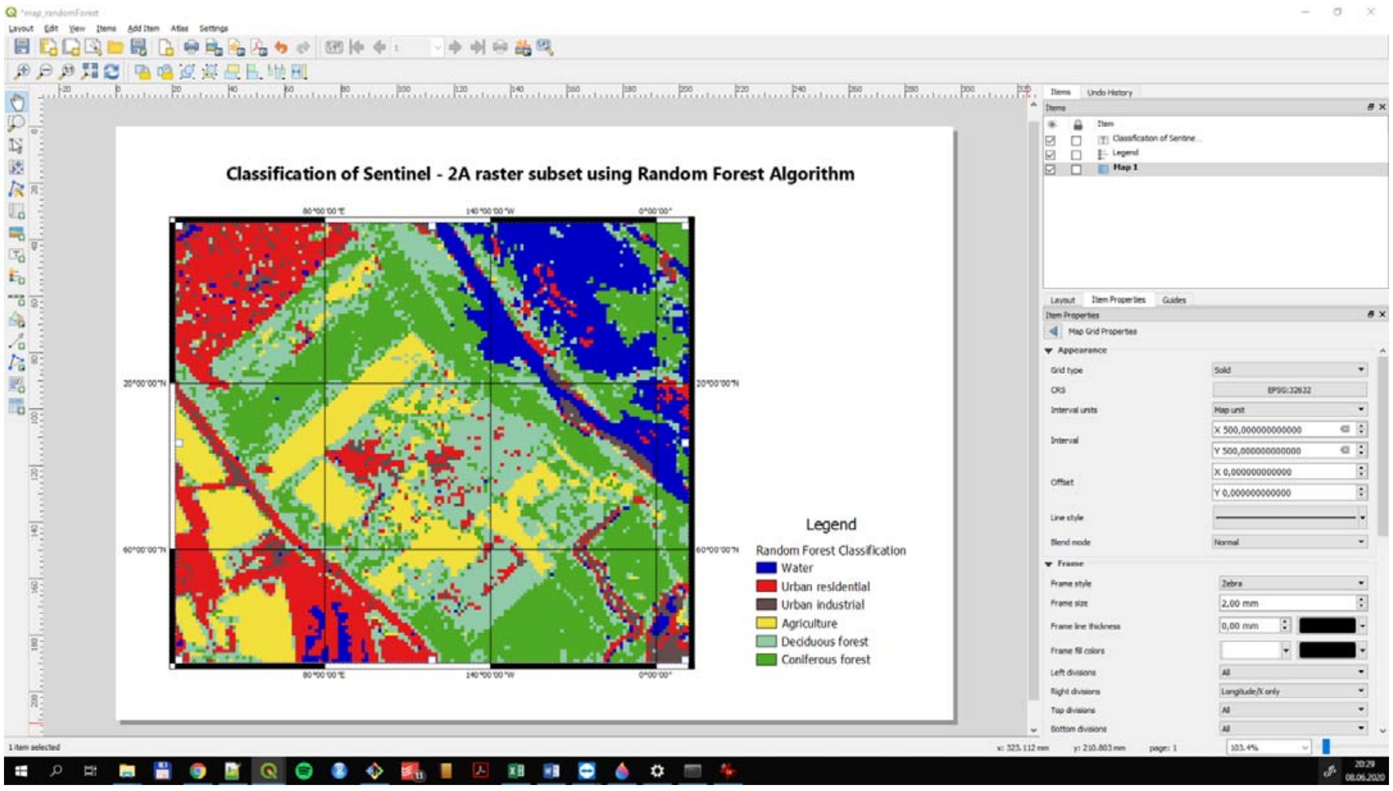
Item Properties – Legend

Main properties
Title
Title alignment: Left
Map: Map 1
Wrap text on
☑ Resize to fit contents

Legend items
☐ Auto update — Update all
shapefiles
validation_polygons
training_polygons
aoi_all
aoi_single
rf_nt250
☑ Only show items inside current atlas feature

shapefiles
validation_polygons
val
1
2
3
4
5
6
training_polygons
training
training_scp
train
training_yu
aoi_all
aoi_workingarea
AOIs
aoi_single
id_1
id_2
id_3
rf_nt250
water
urb.residential
urb.industrial
agriculture
dec.forest
con.forest

More information: https://docs.qgis.org/3.10/en/docs/user_manual/print_composer/overview_composer.html (https://docs.qgis.org/3.10/en/docs/user_manual/print_composer/overview_composer.html)


*map_randomForest
Layout Edit View Items Add Item Atlas Settings
Items Undo History
Classification of Sentine...
Legend
Map 1
Layout Item Properties Guides
80°00'00"E
140°00'00"W
0°00'00"
20°00'00"N
60°00'00"N

Classification of Sentinel - 2A raster subset using Random Forest Algorithm

Legend
Random Forest Classification
Water
Urban residential
Urban industrial
Agriculture
Deciduous forest
Coniferous forest

Item Properties – Map Grid Properties

Appearance
Grid type: Solid
CRS: EPSG:32632
Interval units: Map unit
Interval: X 500,000000000000; Y 500,000000000000
Offset: X 0,000000000000; Y 0,000000000000
Line style
Blend mode: Normal

Frame
Frame style: Zebra
Frame size: 2,00 mm
Frame line thickness: 0,00 mm
Frame fill colors
Left divisions: All
Right divisions: Longitude/X only
Top divisions: All
Bottom divisions: All

4. Once your map contains all relevant elements, you can export it as a PDF or image. I suggest PNG format for the latter.
   This map can be loaded afterwards in Word or LaTeX for the article.

In Python: Pre-processing of raster files

A. Create a RasterStack

5. Load the single bands 2, 3, 4 and 8 of the Sentinel-2A product, and store them in a list

In [1]:

```
# path where my data is located
folder_src = r"C:\Users\ulloa-to\PythonProjects\practices_fe1_ss2020\scripts\data\p7"
```

In [2]:

```
import os

# this command pastes the path to your file (defined by 'folder_src') and the name of your
# since my raster is located in a subfolder 'raster', I include it as well
s2_b2_file = os.path.join(folder_src,'raster/s2_L2A_b2_nan.tif')
s2_b3_file = os.path.join(folder_src,'raster/s2_L2A_b3_nan.tif')
s2_b4_file = os.path.join(folder_src,'raster/s2_L2A_b4_nan.tif')
s2_b8_file = os.path.join(folder_src,'raster/s2_L2A_b8_nan.tif')

file_list = [s2_b2_file, s2_b3_file, s2_b4_file, s2_b8_file]
# print the output
file_list
```

Out[2]:

```
['C:\\Users\\ulloa-to\\PythonProjects\\practices_fe1_ss2020\\scripts\\data
\\p7\\raster/s2_L2A_b2_nan.tif',
 'C:\\Users\\ulloa-to\\PythonProjects\\practices_fe1_ss2020\\scripts\\data
\\p7\\raster/s2_L2A_b3_nan.tif',
 'C:\\Users\\ulloa-to\\PythonProjects\\practices_fe1_ss2020\\scripts\\data
\\p7\\raster/s2_L2A_b4_nan.tif',
 'C:\\Users\\ulloa-to\\PythonProjects\\practices_fe1_ss2020\\scripts\\data
\\p7\\raster/s2_L2A_b8_nan.tif']
```

6. Visualize one of the bands. Use the show() function from rasterio for this.

In [ ]:

```
import rasterio
from rasterio.plot import show
```

7. Check the metadata of the band chosen.

In [ ]:

```

```

8. What can you see in the image? (use QGIS to help yourself if needed)

In [ ]:

```

```

9. Describe in words which area covers this image

In [ ]:

```

```

10. Use all the bands to create a Rasterstack and export it

In [5]:

```python
# Read metadata of first file
with rasterio.open(file_list[0]) as src0:
    meta = src0.meta

# Update meta to reflect the number of layers
meta.update(count = len(file_list))

# Read each layer and write it to stack
with rasterio.open(os.path.join(folder_src,'s2_rasterstack_b2348.tif'), 'w', **meta) as dst
    for id, layer in enumerate(file_list, start=1):
        with rasterio.open(layer) as src1:
            dst.write_band(id, src1.read(1))
```

11. Open your rasterstack in Python, using the os.path.join() and rasterio.open() functions

In [ ]:

```

```

12. Read the metadata using the function meta()

In [ ]:

```python
s2_file.meta
```

13. Using the functions from the rasterio package. Extract information about the extent, pixel values and dimensions of the image

In [ ]:

```

```

14. What do the values of the axes mean?

In [ ]:

```

```

15. Which bands are being plotted here?

In [ ]:

```

```

B. Crop a RasterStack

Load the TCI image. This raster is quite big, which makes it more difficult to perform analysis. We will crop it at the shape of the vector file 'id_3.shp'

16. First, load the raster file, using the os.path.join() and rasterio.open() functions

In [ ]:

```

```

17. Plot the TCI. Use the function show() from rasterio

In [ ]:

```

```

18. Second, load the vector file, using the function 'read_file' from geopandas. Afterwards, check the coordinate reference system.

In [ ]:

```
import geopandas as gp
```

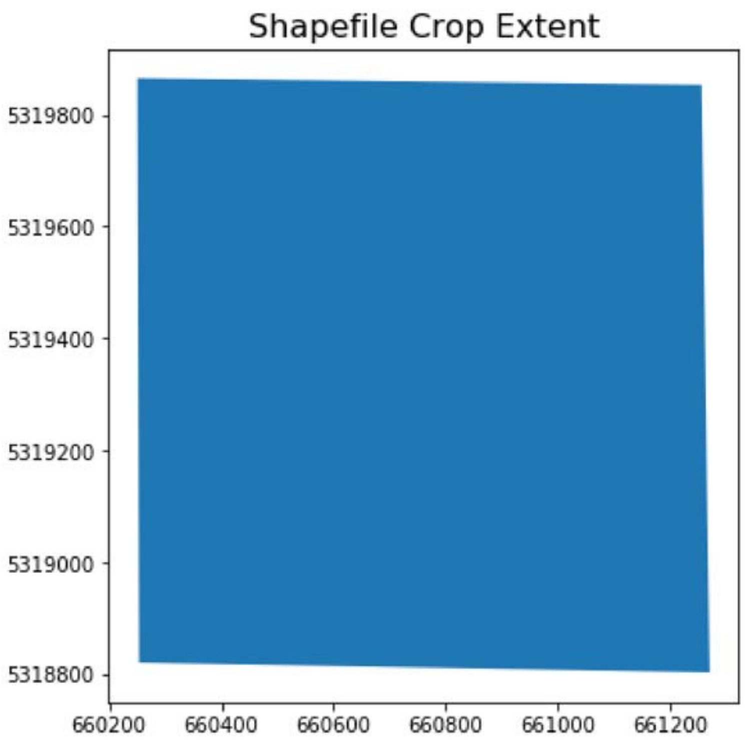
19. Plot the shapefile alone and afterwards, on top of the TCI. Use matplotlib.pyplot

In [10]:

```
import matplotlib.pyplot as plt # nickname
%matplotlib inline

# plot shapefile

fig, ax = plt.subplots(figsize=(6, 6))
shapefile.plot(ax=ax)
ax.set_title("Shapefile Crop Extent", fontsize=16)
```

Out[10]:

```
Text(0.5, 1, 'Shapefile Crop Extent')
```



20. Plot the shapefile on top of the TCI raster.

In [ ]:

```

```

21. Finally, crop the raster to the shape of the shapefile.

In [12]:

```
import fiona
import rasterio
import rasterio.mask

with fiona.open(shape_path, "r") as shapefile:
    shapes = [feature["geometry"] for feature in shapefile]
```

In [13]:

```
with rasterio.open(s2_tci_path) as src:
    out_image, out_transform = rasterio.mask.mask(src, shapes, crop=True)
    out_meta = src.meta
```

In [14]:

```
out_meta.update({"driver": "GTiff",
                 "height": out_image.shape[1],
                 "width": out_image.shape[2],
                 "transform": out_transform})

with rasterio.open(os.path.join(folder_src,"raster/TCI_subset.tif"), "w", **out_meta) as de
    dest.write(out_image)
```

21. Visualize your results. Plot the output of the cropping function. You might need to load the raster into Python using the rasterio module.

In [ ]:

```

```

22. Check the metadata of the cropped TCI

In [ ]:

```

```

23. Compare the metadata from TCI and cropped TCI. Which parameters changed?

In [ ]:

```

```

C. Stretching of the data

Often are the images is not easy to see, and therefore we would need to perfom an streching of the pixels values. This doesn't change the pixel values of the image, it is just a visualization measure.

For the stretching, we will use the package skimage from scikit. The data input should be in numpy arrays. Therefore, we will first adapt the image into numpy arrays before stretching the data.

For more information on image stretching: https://scikit-image.org/docs/dev/api/skimage.exposure.html (https://scikit-image.org/docs/dev/api/skimage.exposure.html)

24. Load raster file S2A_L2A_T32UPU_rstck_id1.tif as an array. Check the dimensions of the file using shape.

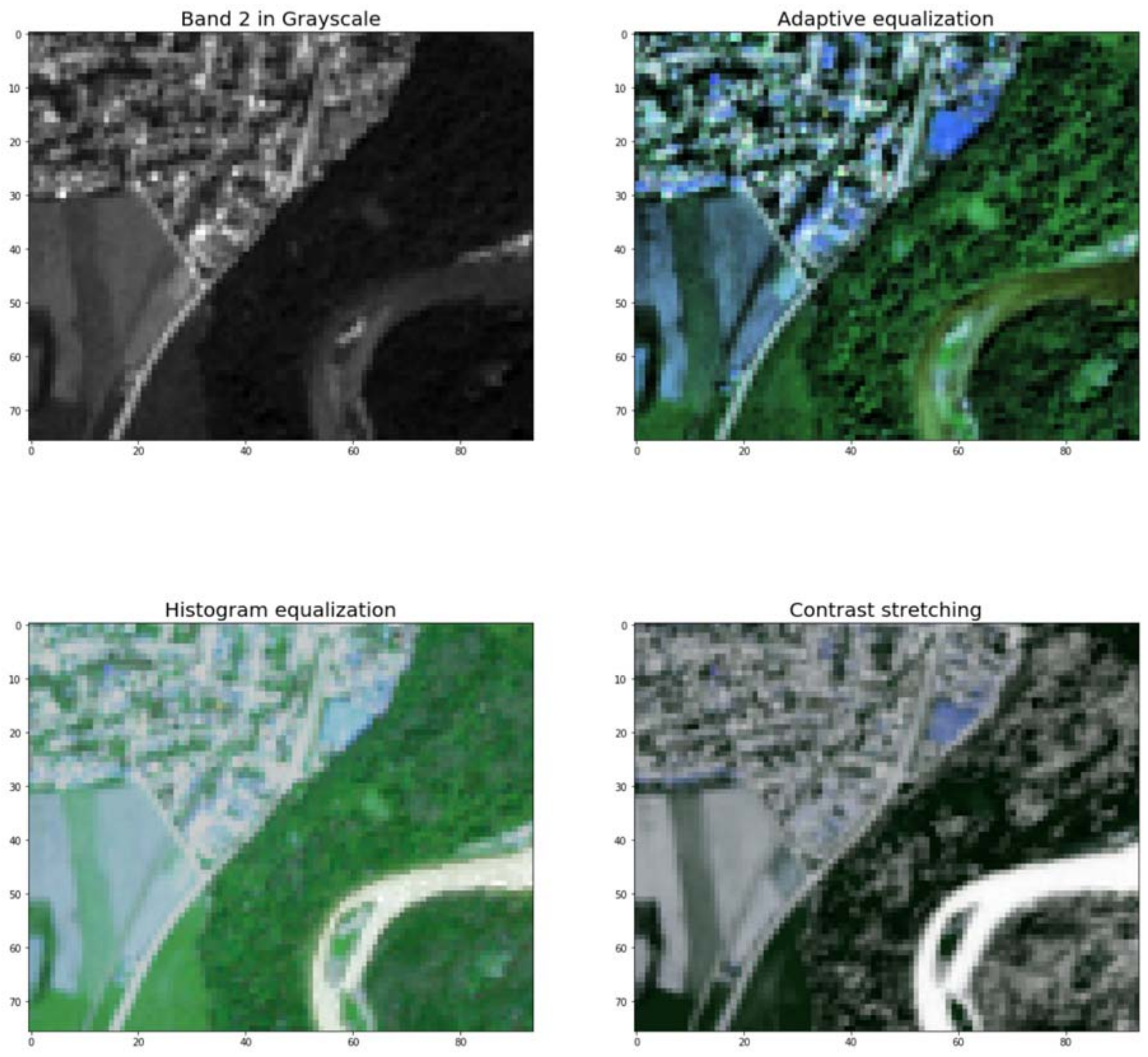
In [ ]:

```
from osgeo import gdal, ogr, gdal_array # I/O image data
import numpy as np # math and array handling
```

25. Perform diverse stretching on the images and plot them

In [19]:

```
from skimage import data, exposure, img_as_float

# Define the size of the graphs
fig = plt.figure(figsize=(20,20))
# Define the spacing among graphs
fig.tight_layout(pad=4.0)

plt.subplot(221)
plt.imshow(img[:, :, 0], cmap=plt.cm.Greys_r)
plt.title("Band 2 in Grayscale", fontsize=20)

# Adaptive Equalization
plt.subplot(222)
plt.imshow(exposure.equalize_adapthist(img, clip_limit=0.03))
plt.title('Adaptive equalization', fontsize=20)

# Equalization
plt.subplot(223)
plt.imshow(exposure.equalize_hist(img))
plt.title('Histogram equalization', fontsize=20)

# Contrast stretching
plt.subplot(224)
p2, p98 = np.percentile(img, (2, 98))
plt.imshow(exposure.rescale_intensity(img, in_range=(p2, p98)))
plt.title('Contrast stretching', fontsize=20)

# show your plots. This command has to be run only once.
plt.show()
```



26. Which one would be your preferred method? Discuss using the documentation provided in scikit: [https://scikit-image.org/docs/dev/api/skimage.exposure.html (https://scikit-image.org/docs/dev/api/skimage.exposure.html)](https://scikit-image.org/docs/dev/api/skimage.exposure.html)

In [ ]:

```

```

This tutorial was prepared with the support from Gabriel Cevallos. June 2020