NDVI Calculation & Visualization Script Supporting Document

Defining the Task

Image processing is a fundamental component of the project. The **images/** directory contains 43 GeoTIFFs (embedded in a TIFF file, it is a metadata standard that allows georeferencing information) from an area in Minnesota. Though optical imagery typically contains red, green, and blue bands (layers), these files contain only red and near-infrared (**NIR**) bands. These bands are critical for determining the health of vegetation, particularly with Normalized Difference Vegetation Index (NDVI).

The task is to load the imagery into a NumPy array and produce several outputs:

- A time series of the average NDVI over time. This is the average NDVI across each image.
- A mosaicked NumPy array with three bands. Mosaicking is a method of flattening information from many images into a single image. The values generated are:
    - Max, Mean, Median & Min NDVI

Key concepts to know before starting

What is a Normalized Difference Vegetation Index (NDVI)?

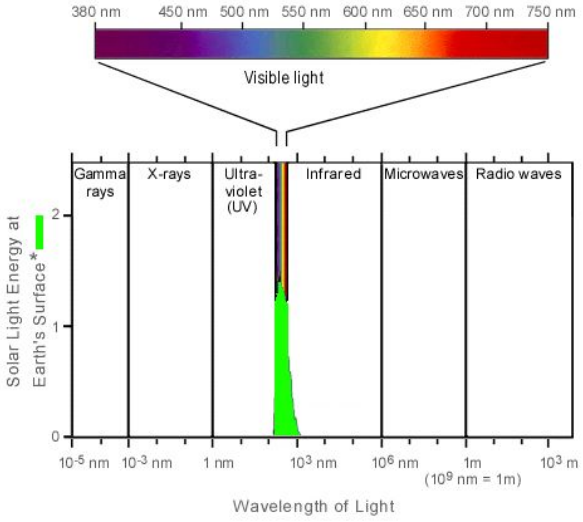
NDVI allows for the quantification of vegetation by measuring the difference between near-infrared(NIR), which vegetation strongly reflects & red light, which vegetation absorbs. NDVI has a range between -1 to +1. The negative values often times are attributed to water while the closer to +1 the values are, it indicated the high possibility of being dense green leaves. On the other hand, if the NDVI is close to zero, this can be an indication of an urbanized area rather than water or dense green leaves. The calculation for NDVI can be seen below.

$$NDVI = \frac{(NIR - Red)}{(NIR + Red)}$$

For our analysis in this python script, we will be using SENTINEL-2 Imagery to calculate the NDVI. SENTINEL-2 has 13 spectral bands: four bands at 10 meters, six bands at 20 meters and three bands at 60 meters spatial resolution with an orbital swath width of 290 km. In order to calculate the NDVI using this type of imagery, we need Band 4 (red) and Band 8 (NIR)

What are Spectral Bands?

Spectral bands are a limited range of values the satellite sensor is able to detect along a spectrum. Visible light is only a very small portion of all light waves as shown in the image below.



Source: Arizona State University

Code Breakdown

Section 1: Importing different modules into our python Script:

I. Rasterio allows us to read and write GeoTIFF and other formats in order to store gridded raster datasets such as the Sentinel-2 imagery we are working with.
   A. Rasterio reads raster data into NumPy arrays allowing the plotting of a single band as two-dimensional data.

II. Matplotlib allows for creating comprehensive visualization using animation and interactive figures.

III. Folium, similar to the previous plotting modules, it allows data that has already been manipulated in python, such as NDVI, to create visualized data on a leaflet map.

IV. NumPy is used throughout the code to allow for scientific calculation within python such as NDVI using NumPy Arrays, which is seen later in the code.

```
import rasterio
from rasterio import plot
from rasterio.plot import show, show_hist
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
import folium
import numpy as np
import os

print ('All neccessary modules imported')
```

Section 2: Testing to see the red and NIR band:

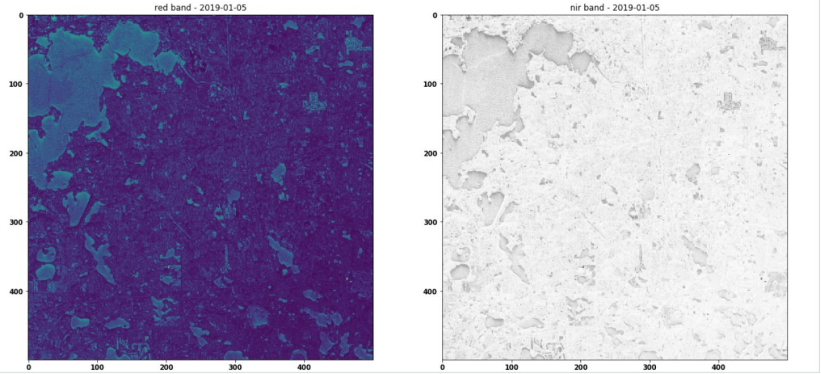
I. In the following lines, we are stacking the red and NIR bands in a NumPy array for the 2019-01-05 imagery. Then we have to see the shape of that NumPy array

```
In [517]: print ('Stacking the red and nir bands in a numpy array for the 2019-01-05 imagery')
          fp = 'images/sentinel-2:L1C:2019-01-05_15TVK_99_S2B_v1-red-nir.tif'
          with rasterio.open(fp, 'r') as f:
              red = f.read(1)

          with rasterio.open(fp, 'r') as f:
              nir = f.read(2)

          array = np.stack([nir, red])
          array.shape

          Stacking the red and nir bands in a numpy array for the 2019-01-05 imagery

Out[517]: (2, 500, 500)
```

Section 3: Testing to see the red and NIR band:

I. In the following lines, we are plotting the red band, displayed in the blue plot on the left and NIR band displayed in the grey plot on the right.

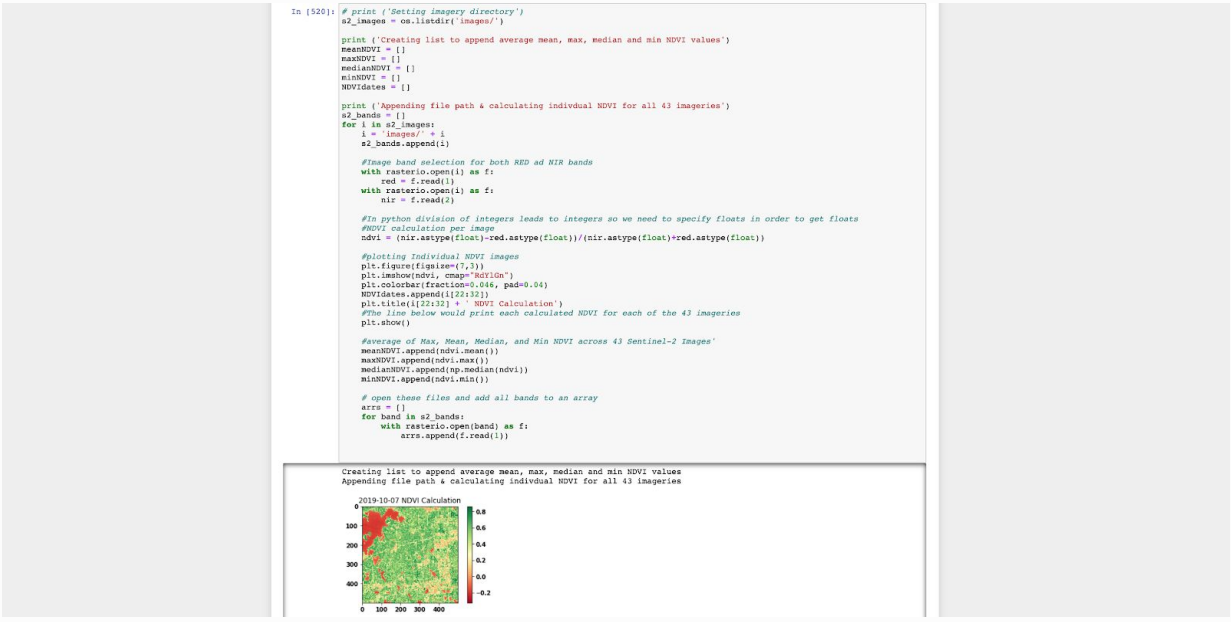
```
In [518]: print ('Plotting the red and nir bans for 2019-01-05 imagery')
          plt.figure(figsize=(20,10))

          plt.subplot(1, 2, 1)
          plt.imshow(red)
          plt.title('red band - 2019-01-05')

          plt.subplot(1, 2, 2)
          plt.imshow(nir, cmap=plt.cm.Greys)
          plt.title('nir band - 2019-01-05');

          Plotting the red and nir bans for 2019-01-05 imagery
```



Section 4: For loop to calculate individual NDVI and calculate important values:

In the following lines, we are doing a few things:

I. Setting the OS directory where the images are located
II. Creating empty lists for Max, Mean, Median and Min values to be appended as the NDVI and calculation done for each imagery
III. s2_bands = [ ] is a list where the full path of each imagery is appended to as the for loop runs, which it appends 'images/' to the beginning of each file name in order to easy finding the files when doing the NDVI calculations
IV. Using Rasterio, identify the different band (red & NIR) for each image
V. As it iterates through the loop, it will find the proper bands for each imagery located in the directory and calculate the individual NDVI, but it is also is capturing the dates from the file name and appending it to NDVIdates = [ ]. Lastly, it plots the NDVI figures.
VI. It will also calculate the Max, Mean, Median and Min values and append them to their respected lists that we create in Section 4.II.
VII. Lastly, using Rasterio we append all bands in arrs = [ ], which we created in Step IV in order to be used to calculate the array and use for overall NDVI calculation across all 43 imageries.



```
In [520]: # print ('Setting imagery directory')
s2_images = os.listdir('images/')

print ('Creating list to append average mean, max, median and min NDVI values')
meanNDVI = []
maxNDVI = []
medianNDVI = []
minNDVI = []
NDVIdates = []

print ('Appending file path & calculating indivdual NDVI for all 43 imageries')
s2_bands = []
for i in s2_images:
    i = 'images/' + i
    s2_bands.append(i)

    #Image band selection for both RED ad NIR bands
    with rasterio.open(i) as f:
        red = f.read(1)
    with rasterio.open(i) as f:
        nir = f.read(2)

    #In python division of integers leads to integers so we need to specify floats in order to get floats
    #NDVI calculation per image
    ndvi = (nir.astype(float)-red.astype(float))/(nir.astype(float)+red.astype(float))

    #plotting Individual NDVI images
    plt.figure(figsize=(7,7))
    plt.imshow(ndvi, cmap="RdYlGn")
    plt.colorbar(fraction=0.046, pad=0.04)
    NDVIdates.append(i[22:32])
    plt.title(i[22:32] + ' NDVI Calculation')
    #The line below would print each calculated NDVI for each of the 43 imageries
    plt.show()

    #average of Max, Mean, Median, and Min NDVI across 43 Sentinel-2 Images'
    meanNDVI.append(ndvi.mean())
    maxNDVI.append(ndvi.max())
    medianNDVI.append(np.median(ndvi))
    minNDVI.append(ndvi.min())

    # open these files and add all bands to an array
    arrs = []
    for band in s2_bands:
        with rasterio.open(band) as f:
            arrs.append(f.read(1))
```

```
Creating list to append average mean, max, median and min NDVI values
Appending file path & calculating indivdual NDVI for all 43 imageries
```


Section 5: Creating histogram and viewing array:

I. We are creating a histogram using Matplotlib to view all the imagery frequencies
II. Then we are viewing the array composition

```
In [521]: print ('Plotting a histogram of all imagery and their frequency')
          #histogram of the data
          fig, ax = plt.subplots(figsize=(20,12))
          show_hist(sentinel_img, ax=ax)
```

```
In [522]: print('printing numpy array composition')
          sentinel_img[:,0,0]

          printing numpy array composition

Out[522]: array([ 533, 1942,  498, 2889, 2265,  713, 2528, 1491, 2891, 1027, 2224,
                 1312,  519, 1832,  786,  563, 2332,  649,  449,  397,  541,  410,
                  515,  374, 2974,    0, 1286,  406,  438, 1983, 1996, 2449, 2656,
                  460,  381, 2041, 1244, 2248,  803, 2266, 2717,  417, 1235],
                dtype=uint16)
```

Section 6: Creating a NumPy Array and Calculating Mosaicked NDVI:

In the following several lines some data manipulation and calculation:

I. In the first line, we will be converting the arrs = [ ] array we created in Section 4.VII into a NumPy Array using np.array and store it in sentinel_img, which will allow us to do the overall NDVI calculation

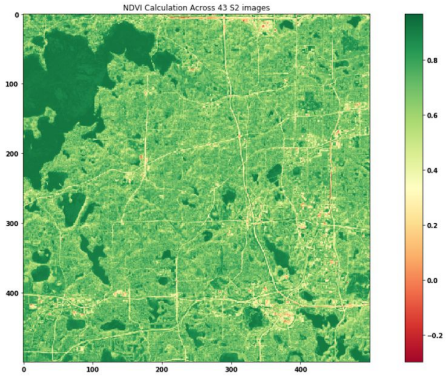
II. In the next line, in order not to get errors in our calculation we have to ignore the nan values in some of the imagery.

III. Using the now Numpy Array sentinel_img, we identify the red and NIR bands

IV. We then calculate the NDVI across all values in the array using the formula for NDVI.

V. We then plot the NDVI image using Matplotlib

```
In [523]: print ('appending each image path into an number array')
          #convert the list to a numpy array
          sentinel_img = np.array(arrs, dtype=arrs[0].dtype)
          #check the shape of this array
          sentinel_img.shape

          print ('ignoring nan values for NDVI calculation across all  ')
          np.seterr(divide='ignore', invalid='ignore')

          print ('assigning red and nir bands to variables')
          bandRed = sentinel_img[0] # eigth band
          bandNIR = sentinel_img[1] # fourth band

          print ('Calculating NDVI on the combine numpy array contaning all 43 imageries')
          #In python division of integers leads to integers so we need to specify floats in order to get floats
          ndvi = (bandNIR.astype(float)-bandRed.astype(float))/(bandNIR.astype(float)+bandRed.astype(float))

          print ('plotting NDVI imagery across the combined 43 images')
          #normal matrix for all 43 imags calculated from the numpy array sentinel_img
          plt.figure(figsize=(20,10))
          plt.title('NDVI Calculation Across 43 S2 images');
          plt.imshow(ndvi, cmap="RdYlGn")
          plt.colorbar()
          plt.show()

          appending each image path into an number array
          ignoring nan values for NDVI calculation across all
          assigning red and nir bands to variables
          Calculating NDVI on the combine numpy array contaning all 43 imageries
          plotting NDVI imagery across the combined 43 images
```



Section 7: Calculating Max, Mean, Median & MIN of the NDVI across all 43 images:

```
In [524]: print ('Calculating max, mean, median and min NDVI values for all 43 images')
          print('\nMax NDVI: {m}'.format(m=ndvi.max()))
          print('Mean NDVI: {m}'.format(m=ndvi.mean()))
          print('Median NDVI: {m}'.format(m=np.median(ndvi)))
          print('Min NDVI: {m}'.format(m=ndvi.min()))

          Calculating max, mean, median and min NDVI values for all 43 images

          Max NDVI: 0.9678585777420651
          Mean NDVI: 0.7034083353725836
          Median NDVI: 0.7073682637321146
          Min NDVI: -0.2974963181148748
```

Section 8: Calculating the average Max, Mean, Median and MIN of the NDVI.

I. Slightly different from the above calculation because we are taking the individual values for each category and for each imagery that we appended in Section 4.VI (meanNDVI = [ ], maxNDVI = [ ], medianNDVI = [ ], & minNDVI = [ ]) and taking their average values.

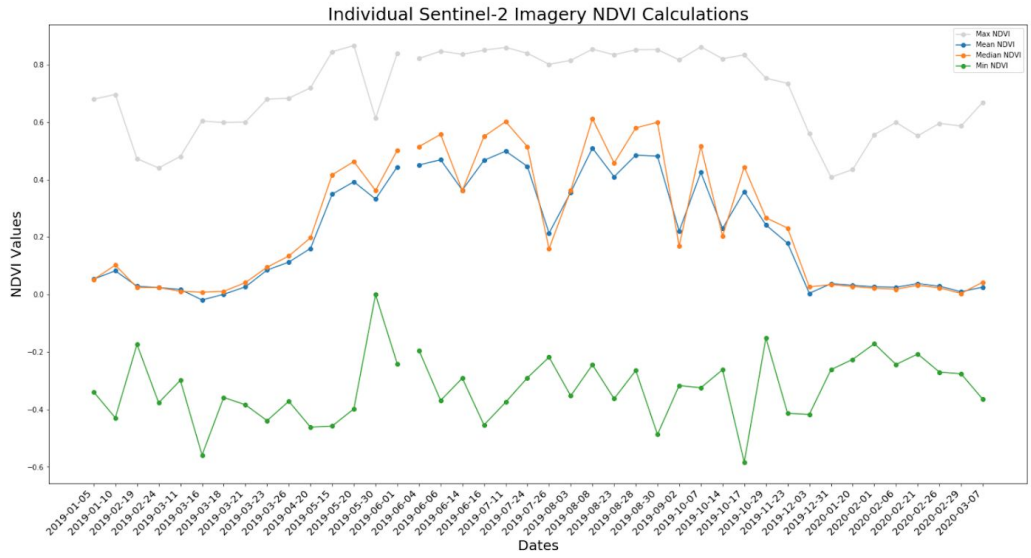
```
In [525]: print ('Printing  the average of Max, Mean, Median, and Min NDVI across 43 Sentinel-2 Images')
          print('\nAverage Max NDVI: ', np.nanmean(maxNDVI))
          print('Average Mean NDVI: ', np.nanmean(meanNDVI))
          print('Average Median NDVI: ', np.nanmean(medianNDVI))
          print('Average Min NDVI: ', np.nanmean(minNDVI))

          Printing  the average of Max, Mean, Median, and Min NDVI across 43 Sentinel-2 Images

          Average Max NDVI:  0.7067394839962414
          Average Mean NDVI:  0.2174871451123512
          Average Median NDVI:  0.24731499726291745
          Average Min NDVI:  -0.32563038711980197
```

Section 9: Plotting the average Max, Mean, Median and MIN NDVI values based on each date.

I. The dates for the x-axis are generated by the list in NDVIdates = [ ] that is appended in Section 4.V in the same order the individual imagery NDVI calculation was taking place.

In [105]:
```
print ('Plotting max, mean, median and min NDVI values for each imagery')

fig, ax = plt.subplots(figsize=(25,12))

plt.plot(NDVIdates, maxNDVI, 'o-', color='lightgrey', label='Max NDVI')
plt.plot(NDVIdates, meanNDVI, 'o-', label='Mean NDVI')
plt.plot(NDVIdates, medianNDVI, 'o-', label='Median NDVI')
plt.plot(NDVIdates, minNDVI, 'o-', label='Min NDVI')

plt.xticks(
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)

plt.xlabel("Dates", fontsize=20)
plt.ylabel("NDVI Values", fontsize=20)
#plt.legend(loc='center')

plt.legend()
plt.title('Individual Sentinel-2 Imagery NDVI Calculations', fontsize=25)
plt.show()
```

```
Plotting max, mean, median and min NDVI values for each imagery
```



Summary

The project consisted of 43 SENTINEL-2 imagery over Minnesota. Throughout the script, we utilize python modules such as Rasterio, Matplotlib, Folium, and NumPy to perform various calculations and visualization tasks. The main purpose of the project was to determine a time series of the average NDVI over time and calculate the Max, Mean, Median & Min NDVI values, both across 43 images and the average of the independent images. This was accomplished using the NDVI formula as well as many visualization techniques.

END OF CODE DOCUMENTATION!