Young_NEON_Data_Institute_Week3_Assignment

July 8, 2018

0.1 # Week 3 Assignment

0.2 Check that Python version is 3.5.x

```
In [3]: import sys
        sys.version
```

```
Out[3]: '3.5.5 |Anaconda custom (64-bit)| (default, Apr 26 2018, 08:11:22) \n[GCC 4.2.1 Compatib
```

0.3 Import gdal

```
In [4]: import gdal
```

0.4 Import numpy and matplotlib. Turn warnings 'off'

```
In [5]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        import warnings
        warnings.filterwarnings('ignore')
```

0.5 Define function to read in RGB image

This function was obtained from NEON Data Institute

```
In [6]: def RGBraster2array(RGB_geotif):
            """RGBraster2array reads in a NEON AOP geotif file and returns
            a numpy array, and header containing associated metadata with spatial information.
            --------
            Parameters
                RGB_geotif -- full or relative path and name of reflectance hdf5 file
            --------
            Returns
            --------
            array:
                numpy array of geotif values
            metadata:
                dictionary containing the following metadata (all strings):
                    array_rows
```

```
        array_cols
        bands
        driver
        projection
        geotransform
        pixelWidth
        pixelHeight
        extent
        noDataValue
        scaleFactor
--------
Example Execution:
--------
RGB_geotif = '2017_SERC_2_368000_4306000_image.tif'
RGBcam_array, RGBcam_metadata = RGBraster2array(RGB_geotif) """

metadata = {}
dataset = gdal.Open(RGB_geotif)
metadata['array_rows'] = dataset.RasterYSize
metadata['array_cols'] = dataset.RasterXSize
metadata['bands'] = dataset.RasterCount
metadata['driver'] = dataset.GetDriver().LongName
metadata['projection'] = dataset.GetProjection()
metadata['geotransform'] = dataset.GetGeoTransform()

mapinfo = dataset.GetGeoTransform()
metadata['pixelWidth'] = mapinfo[1]
metadata['pixelHeight'] = mapinfo[5]

metadata['ext_dict'] = {}
metadata['ext_dict']['xMin'] = mapinfo[0]
metadata['ext_dict']['xMax'] = mapinfo[0] + dataset.RasterXSize/mapinfo[1]
metadata['ext_dict']['yMin'] = mapinfo[3] + dataset.RasterYSize/mapinfo[5]
metadata['ext_dict']['yMax'] = mapinfo[3]

metadata['extent'] = (metadata['ext_dict']['xMin'],metadata['ext_dict']['xMax'],
                      metadata['ext_dict']['yMin'],metadata['ext_dict']['yMax'])

raster = dataset.GetRasterBand(1)
array_shape = raster.ReadAsArray(0,0,metadata['array_cols'],metadata['array_rows']).
metadata['noDataValue'] = raster.GetNoDataValue()
metadata['scaleFactor'] = raster.GetScale()

array = np.zeros((array_shape[0],array_shape[1],dataset.RasterCount),'uint8') #pre-a
for i in range(1, dataset.RasterCount+1):
    band = dataset.GetRasterBand(i).ReadAsArray(0,0,metadata['array_cols'],metadata[
    band[band==metadata['noDataValue']]=np.nan
    band = band/metadata['scaleFactor']
```

```
            array[...,i-1] = band

        return array, metadata
```

0.6 Set directory path to downloaded image and run 'RGBraster2array' function.

```
In [8]: RGB_geotif = '/Users/adam/Documents/neon/2017_SERC_2_368000_4306000_image.tif'
        SERC_RGBcam_array, SERC_RGBcam_metadata = RGBraster2array(RGB_geotif)
```

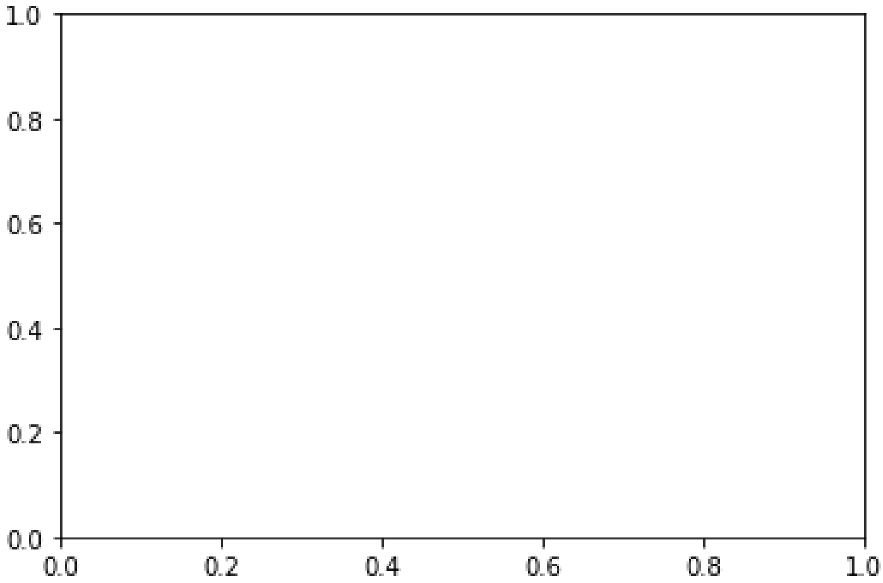
0.7 Check dimensions of image. Should be (1000, 1000, 3).

```
In [9]: SERC_RGBcam_array.shape

Out[9]: (10000, 10000, 3)
```

0.8 Define function to plot array data.

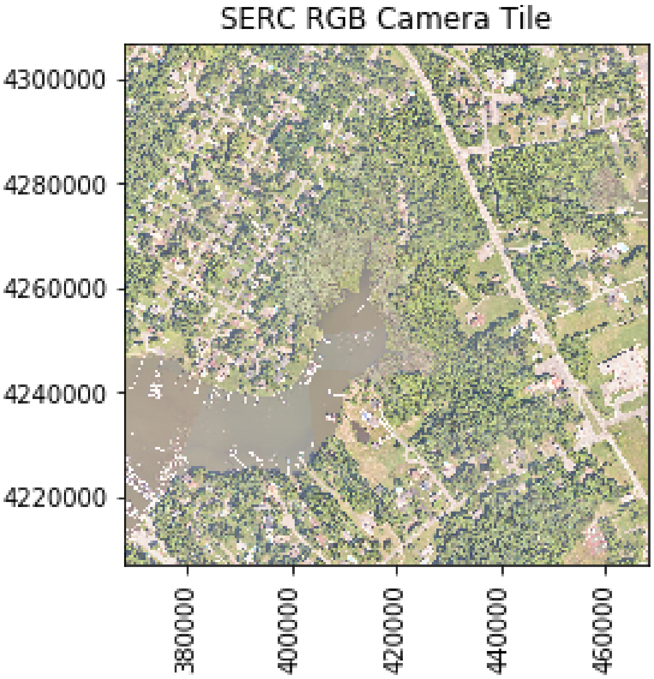
```
In [10]: def plot_band_array(band_array,
                             refl_extent,
                             colorlimit,
                             ax=plt.gca(),
                             title='',
                             cbar ='on',
                             cmap_title='',
                             colormap='spectral'):

             '''plot_band_array reads in and plots a single band or an rgb band combination of a
             --------
             Parameters
             --------
                 band_array: flightline array of reflectance values, created from h5refl2array f
                 refl_extent: extent of reflectance data to be plotted (xMin, xMax, yMin, yMax)
                 colorlimit: range of values to plot (min,max). Best to look at the histogram of
                 ax: optional, default = current axis
                 title: string, optional; plot title
                 cmap_title: string, optional; colorbar title
                 colormap: string, optional; see https://matplotlib.org/examples/color/colormaps
             --------
             Returns
                 plots array of single band or RGB if given a 3-band
             --------
             Example:
             --------
             plot_band_array(SERC_RGBcam_array,
                             SERC_RGBcam_metadata['extent'],
                             (1,255),
                             title='SERC RGB Camera Tile',
                             cbar='off')'''
```

```
    plot = plt.imshow(band_array,extent=refl_extent,clim=colorlimit);
    if cbar == 'on':
        cbar = plt.colorbar(plot,aspect=40); plt.set_cmap(colormap);
        cbar.set_label(cmap_title,rotation=90,labelpad=20)
    plt.title(title); ax = plt.gca();
    ax.ticklabel_format(useOffset=False, style='plain'); #do not use scientific notatio
    rotatexlabels = plt.setp(ax.get_xticklabels(),rotation=90); #rotate x tick labels 9
```



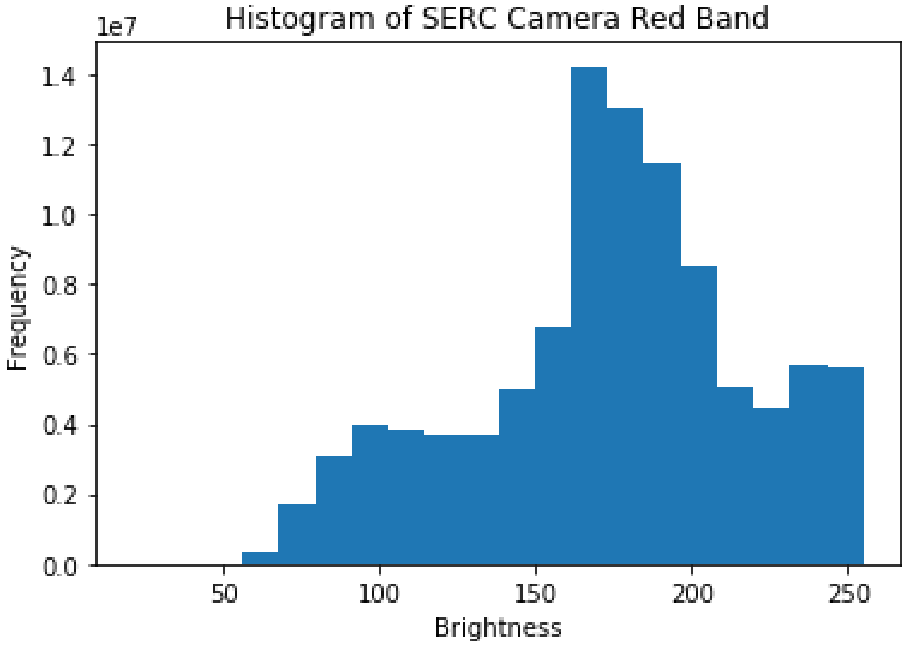
0.9 Now plot the downloaded image from NEON

```
In [11]: plot_band_array(SERC_RGBcam_array,
                         SERC_RGBcam_metadata['extent'],
                         (1,255),
                         title='SERC RGB Camera Tile',
                         cbar='off')
```



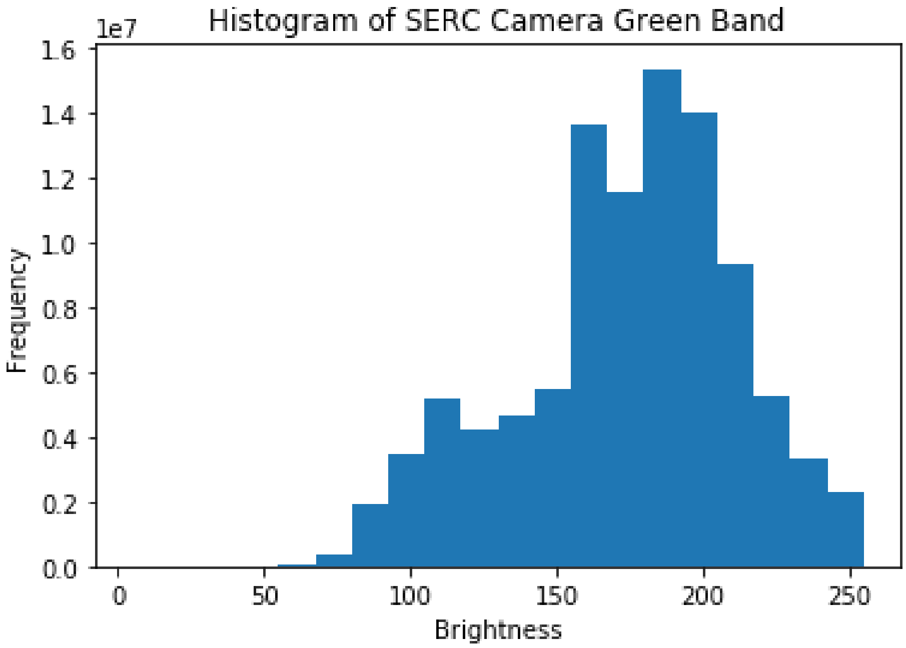
0.10 Now plot histogram of red color channel.

```
In [13]: plt.hist(np.ravel(SERC_RGBcam_array[:,:,0]),20);
         plt.title('Histogram of SERC Camera Red Band')
         plt.xlabel('Brightness'); plt.ylabel('Frequency')

Out[13]: Text(0,0.5,'Frequency')
```



0.11 Green color channel

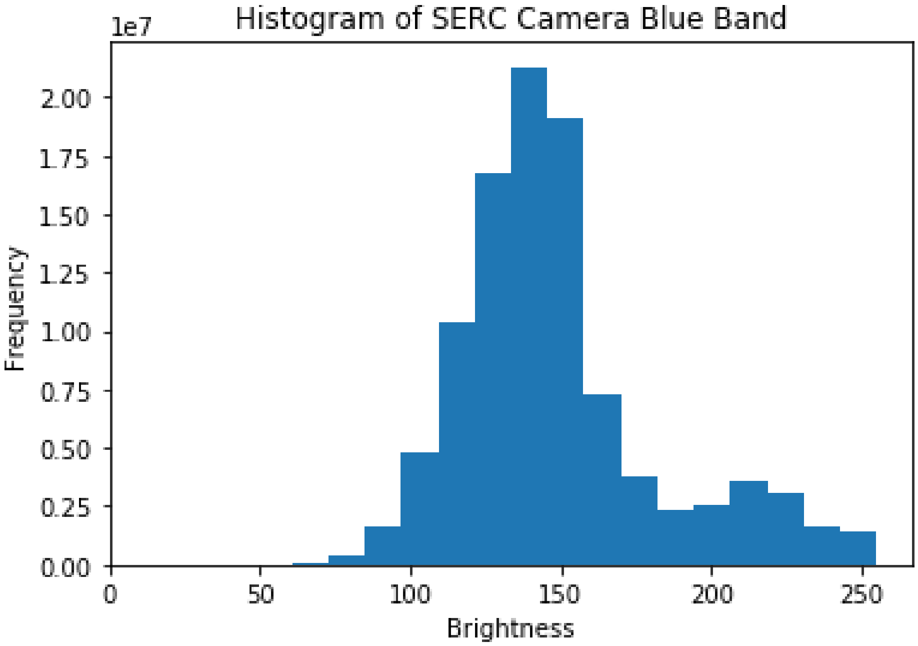
```
In [14]: plt.hist(np.ravel(SERC_RGBcam_array[:,:,1]),20);
         plt.title('Histogram of SERC Camera Green Band')
         plt.xlabel('Brightness'); plt.ylabel('Frequency')

Out[14]: Text(0,0.5,'Frequency')
```



0.12 Blue Color Channel

```
In [15]: plt.hist(np.ravel(SERC_RGBcam_array[:,:,2]),20);
         plt.title('Histogram of SERC Camera Blue Band')
         plt.xlabel('Brightness'); plt.ylabel('Frequency')
```

```
Out[15]: Text(0,0.5,'Frequency')
```



0.13 Print out the [min, max] values for each color (R,G,B)

```
In [23]: # Red values
         rminval = np.amin(SERC_RGBcam_array[:,:,0])
         rmaxval = np.amax(SERC_RGBcam_array[:,:,0])

         # Green Values
         gminval = np.amin(SERC_RGBcam_array[:,:,1])
         gmaxval = np.amax(SERC_RGBcam_array[:,:,1])

         # Blue values
         bminval = np.amin(SERC_RGBcam_array[:,:,2])
         bmaxval = np.amax(SERC_RGBcam_array[:,:,2])

         # Print color values to screen
         print("R values:",[rminval,rmaxval])
         print("G values:",[gminval,gmaxval])
         print("B values:",[bminval,bmaxval])
```

```
R values: [21, 255]
G values: [5, 255]
B values: [12, 255]
```

0.14 What UTM zone is the data in? UTM zone 18N (see metadata printout below).

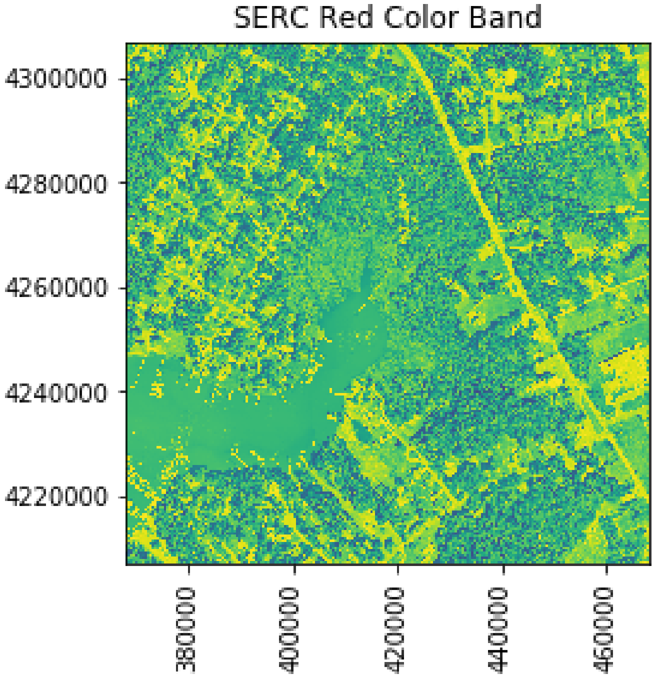
```
In [28]: SERC_RGBcam_metadata['projection']
```

```
Out[28]: 'PROJCS["WGS 84 / UTM zone 18N",GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378
```

0.15 Finally , plot the each color band separately (i.e. Red, Green, Blue).

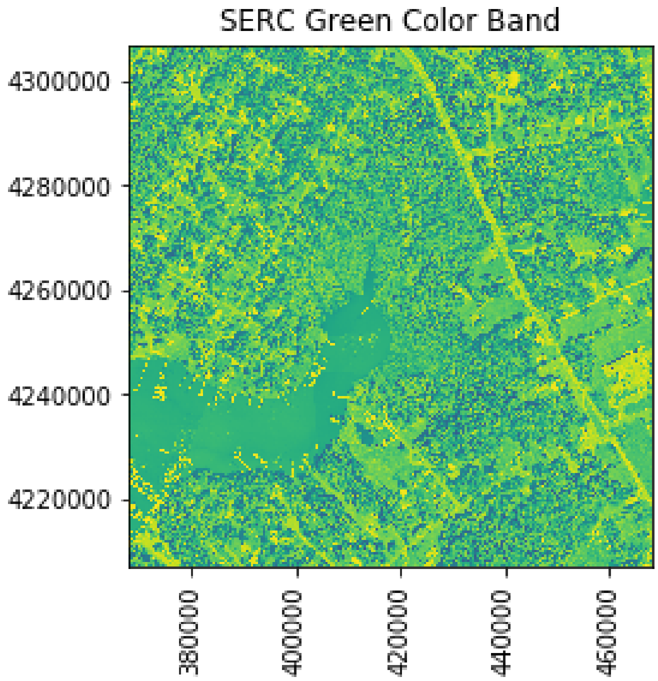
0.16 Red

```
In [29]: plot_band_array(SERC_RGBcam_array[:,:,0],
                         SERC_RGBcam_metadata['extent'],
                         (1,255),
                         title='SERC Red Color Band',
                         cbar='off')
```



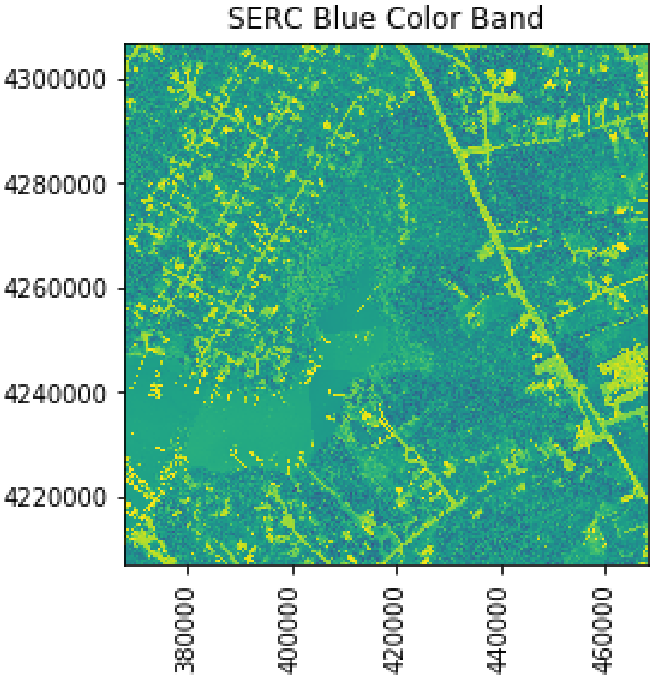
0.17 Green

```
In [30]: plot_band_array(SERC_RGBcam_array[:,:,1],
                         SERC_RGBcam_metadata['extent'],
                         (1,255),
                         title='SERC Green Color Band',
                         cbar='off')
```



0.18 Blue

```
In [31]: plot_band_array(SERC_RGBcam_array[:,:,2],
                         SERC_RGBcam_metadata['extent'],
                         (1,255),
                         title='SERC Blue Color Band',
                         cbar='off')
```


SERC Blue Color Band
4300000
4280000
4260000
4240000
4220000
380000
400000
420000
440000
460000