Plotting a NEON RGB Camera Image (Geotif) in Python

This lesson is a brief introduction to RGB camera images and the geotif raster format in Python. In this lesson, we will read in an RGB camera image for a single tile (1000m x 1000m image) of the NEON Smithsonian Environmental Research Center (SERC) site, which is used as a teaching example during the Data Institute. We will load a Python module containing the functions `RGBraster2array` and `plotRGBimage` and run these functions to read in the image as an array, plot an RGB image of this raster, and plot a histogram of the intensities of each of the three bands.

Objectives

In this tutorial, you will gain familiarity running Python through Jupyter Notebook

1. Importing Python libraries `NumPy` `Matplotlib`
2. Importing the NEON Python module `PreInstitute.py`
3. Plotting a NEON RGB Camera Tile
4. Plotting a histogram of a single band of an RGB Camera Tile

Background

Note: Don't worry about understanding everything in the raster2array function at this point. If you are curious, we encourage you to read the docstrings, but we will go into the inner-workings of this function in more detail during the data institute.

Data Tip: To run a cell you can either select Cell > Run Cells (with your cursor in the cell you want to run), or use the shortcut key Shift + Enter. For more handy shortcuts, refer to the tab Help > Keyboard Shortcuts.

First we will import the gdal package, which contains tools for programming and manipulating the Geospatial Data Abstraction Library (GDAL). For more information on GDAL, please refer to:

http://www.gdal.org/ (http://www.gdal.org/) https://pcjericks.github.io/py-gdalogr-cookbook/ (https://pcjericks.github.io/py-gdalogr-cookbook/)

In [1]:
```
import gdal, osr
```

If you get the following message (or another similar error message), stop here.

Troubleshooting steps:

- from a Jupyter Python cell, run the command: `!conda install gdal`
- from a Command Prompt (Windows) or Terminal (Mac), activate the appropriate environment

Next we will import the `numpy` and `matplotlib` packages. Numpy stands for **Num**erical **Py**thon This is a standard package that comes with the Anaconda installation of Python, so you should not need to do any additional steps to install it.

In [2]:
```
import time
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
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 = # of rows
            array_cols = # of columns
            bands = # of bands
            driver =
            projection =
            geotransform =
            pixelWidth =
            pixelHeight =
            ext_dict = dictionary containing spatial extent with keys 'XMin', 'XMax', 'YMin', 'YMax'
            extent = tuple containing spatial 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']).astype(np.float).shape
    metadata['noDataValue'] = raster.GetNoDataValue()
    metadata['scaleFactor'] = raster.GetScale()

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

    return array, metadata
```

In [4]:
```
RGB_geotif = './2017_SERC_2_368000_4306000_image.tif'
SERC_RGBcam_array, SERC_RGBcam_metadata = RGBraster2array(RGB_geotif)
```

In [5]:
```
print('Shape of SERC RGB Camera Tile Array:',SERC_RGBcam_array.shape)
```

```
Shape of SERC RGB Camera Tile Array: (10000, 10000, 3)
```

We can list all the information stored in the

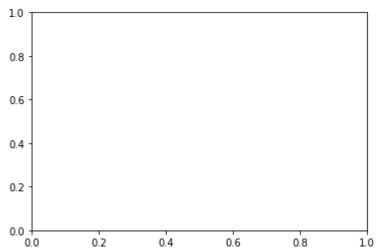
In [6]:
```
#Display information stored in header
for key in sorted(SERC_RGBcam_metadata.keys()):
  print(key)
```

```
array_cols
array_rows
bands
driver
ext_dict
extent
geotransform
noDataValue
pixelHeight
pixelWidth
projection
scaleFactor
```

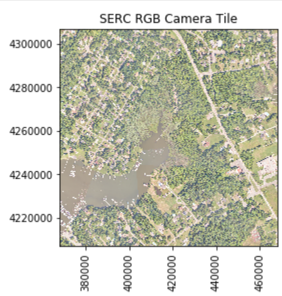
In [7]:
```
def plot_band_array(band_array,refl_extent,colorlimit,ax=plt.gca(),title='',cbar ='on',cmap_title='',colormap='spectral'):

    '''plot_band_array reads in and plots a single band of a reflectance array
    --------
    Parameters
    --------
        band_array: flightline array of reflectance values, created from h5refl2array function
        refl_extent: extent of reflectance data to be plotted (xMin, xMax, yMin, yMax) - use metadata['extent'] from h5refl2array function
        colorlimit: range of values to plot (min,max). Best to look at the histogram of reflectance values before plotting to determine colorlimit.
        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_reference.html for list of colormaps
    --------
    Returns
        plots flightline array of single band of reflectance data
    --------
    See Also
    --------
    plot_subset_band:
        plots a subset of a full flightline reflectance band array
    Example:
    --------
    plot_band_array(SERC_b56_clean,sercRefl_md['extent'],(0,0.3),ax,title='SERC Band 56 Reflectance',cmap_title='Reflectance',colormap='spectral') '''

    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 notation #
    rotatexlabels = plt.setp(ax.get_xticklabels(),rotation=90); #rotate x tick labels 90 degrees
 #  plt.close();
```



In [8]:
```
plot_band_array(SERC_RGBcam_array, SERC_RGBcam_metadata['extent'],(1,255),title='SERC RGB Camera Tile',cbar='off')
```

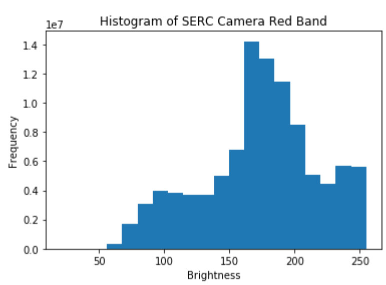


In [9]:
```
hist_start_time = time.time()

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

print('\nData took', round(time.time() - hist_start_time,1), 'seconds to clean.')
```

```
Data took 1.9 seconds to clean.
```



On Your Own

Now that you've followed along to read in and plot an RGB camera image and band, try the following exercises on your own:

1. **Plot histograms of the green and blue bands**
2. **Explore the data** to see what you can learn about the `SERC_RGBcam_array` and associated `SERC_RGBcam_metadata`

```
a. Determine the minimum and maximum reflectance for each band. Print these values with a print statement.
*HINT: Use the numpy functions* `np.amin()` and `np.amax()`
b. What UTM zone is this data in? *HINT: Print out* `SERC_RGBcam_metadata['projection']`
c. ...
```