In [1]:

```
import rasterio
```

In [2]:

```
image_file = '../input/aerialtif/aerial.tif'
satdat = rasterio.open(image_file)
```

```
/opt/conda/lib/python3.7/site-packages/rasterio/__init__.py:207: NotGeoreferencedWarning: Dataset
has no geotransform, gcps, or rpcs. The identity matrix be returned.
  s = DatasetReader(path, driver=driver, sharing=sharing, **kwargs)
```

In [3]:

```
print(satdat.count)
print(satdat.name)
```

```
3
../input/aerialtif/aerial.tif
```

In [4]:

```
print(satdat.indexes)
```

```
(1, 2, 3)
```

In [5]:

```
blue,green,red = satdat.read()
```

In [6]:

```
blue_b = satdat.read(1)
green_b = satdat.read(2)
red_b = satdat.read(3)

print(blue)
print('------------------------------------------------------------>')
print(blue_b)
```

```
[[ 43   8  10 ... 101 108 119]
 [ 44   5   7 ...  92 126 123]
 [ 46   6   2 ...  95 146 102]
 ...
 [ 52  56  56 ...  51  49  47]
 [ 52  58  57 ...  37  37  39]
 [ 53  59  56 ...  37  40  39]]
------------------------------------------------------------>
[[ 43   8  10 ... 101 108 119]
 [ 44   5   7 ...  92 126 123]
 [ 46   6   2 ...  95 146 102]
 ...
 [ 52  56  56 ...  51  49  47]
 [ 52  58  57 ...  37  37  39]
 [ 53  59  56 ...  37  40  39]]
```

In [7]:

```
w = blue.shape[0]
h = blue.shape[1]
print('width :{w}  , height : {h}' .format(w=w,h=h))
```

```
width :554  , height : 763
```

Extracting Metadata information from a image

In [8]:

```
satdata = rasterio.open(image_file)
print(satdata.bounds)
```

```
BoundingBox(left=0.0, bottom=554.0, right=763.0, top=0.0)
```

In [9]:

```
width_in_projected_units = satdata.bounds.right - satdata.bounds.left
height_in_projected_units = satdata.bounds.top - satdata.bounds.bottom

print('width:{} , height:{}'.format(width_in_projected_units, height_in_projected_units))
```

```
width:763.0 , height:-554.0
```

In [10]:

```
print('Rows:{} , Columns:{}'.format(satdata.height, satdata.width))
```

```
Rows:554 , Columns:763
```

In [11]:

```
xres = (satdata.bounds.right - satdata.bounds.left) / satdata.width
yres = (satdata.bounds.top - satdata.bounds.bottom) / satdata.height

print(xres,yres)
print('Are these pixels equal:{}'.format(xres==yres))
```

```
1.0 -1.0
Are these pixels equal:False
```

In [12]:

```
satdata.crs
```

In [13]:

```
row_min = 0
col_min = 0

row_max = satdata.height - 1
col_max = satdata.width - 1

topleft = satdata.transform*(row_min,col_min)
bot_right = satdata.transform*(row_max,col_max)

print('Top left coordinates :{}'.format(topleft))
print('Top right coordinates : {}'.format(bot_right))
```

```
Top left coordinates :(0.0, 0.0)
Top right coordinates : (553.0, 762.0)
```

In [14]:

```
satdata.profile
```

Out[14]:

```
{'driver': 'GTiff', 'dtype': 'uint8', 'nodata': None, 'width': 763, 'height': 554, 'count': 3, 'cr
s': None, 'transform': Affine(1.0, 0.0, 0.0,
       0.0, -1.0, 0.0), 'tiled': False, 'interleave': 'pixel'}
```

0.0, 1.0, 0.0]), 'tiled': False, 'interleave': 'pixel'}

File Compression

Raster data uses compression to reduce filesize.Although there are number of methods , but all of them fall under two categories : lossy and lossless

Lossless compression retains the original value in each pixel of the raster, while lossy methods result in some values being removed.

In [15]:

```
import os
from humanize import naturalsize as sz

size = os.path.getsize('../input/aerialtif/aerial.tif')
print(sz(size))
```

```
1.3 MB
```

In [16]:

```
data = satdata.read()
```

In [17]:

```
# new file using profile metdata from original dataset
profile = satdata.profile
profile['compress'] = 'JPEG'

with rasterio.open('compressed.tif','w',**profile) as dst:
    dst.write(data)
```

```
/opt/conda/lib/python3.7/site-packages/rasterio/__init__.py:223: NotGeoreferencedWarning: The give
n matrix is equal to Affine.identity or its flipped counterpart. GDAL may ignore this matrix and s
ave no geotransform without raising an error. This behavior is somewhat driver-specific.
  **kwargs)
```

Lossy Compression results

In [18]:

```
new_size = os.path.getsize('compressed.tif')
print(sz(new_size))
```

```
358.4 kB
```

In [19]:

```
blue,red,green = satdata.read()
```

Pixels grids as numpy arrays

In [20]:

```
print(type(blue))
```

```
<class 'numpy.ndarray'>
```

In [21]:

```
print(blue.dtype)
```

```
print(blue.dtype)
```

```
uint8
```

In [22]:

```
print(blue.ndim)
```

```
2
```

In [23]:

```
for band in  red,green,blue:
    print('min {min} max {max}'.format(min = band.min() , max = band.max()))
```

```
min 0 max 255
min 0 max 255
min 0 max 255
```

Visualizing imagery with matplotlib

In [27]:

```
def scale(band):
    return band / 100

blue = scale(satdata.read(1))
green = scale(satdata.read(2))
red = scale(satdata.read(3))
```

In [28]:

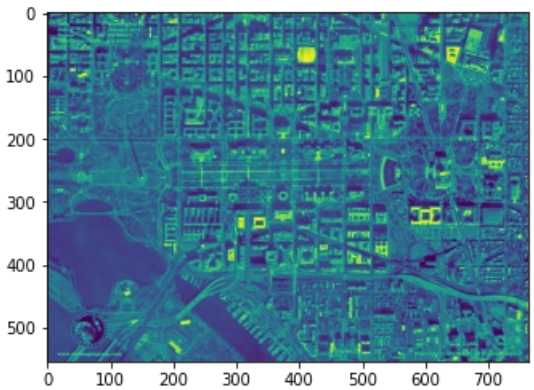
```
import numpy

rgb = numpy.dstack((red,green,blue))
bgr =  numpy.dstack((blue,red,green))
```

In [29]:

```
import matplotlib.pyplot as plt
plt.imshow(blue)
```

Out[29]:

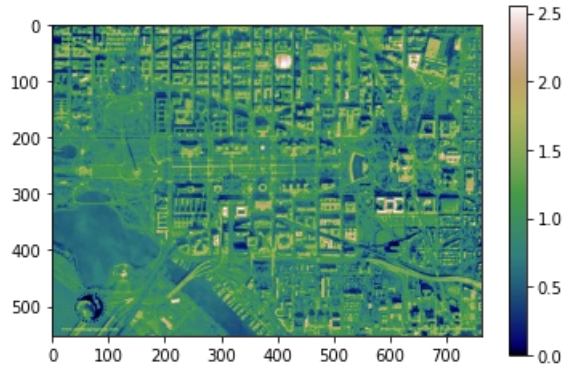
```
<matplotlib.image.AxesImage at 0x7f0693b57510>
```



In [30]:

```
fig = plt.imshow(green)
fig.set_cmap('gist_earth')
plt.colorbar()
```

```
plt.show()
```



In [31]:

```
fig2 = plt.figure(figsize=(20,10))
ax = fig2.add_subplot(111)

plt.title('Histogram Example', fontsize=18, fontweight='bold')
plt.xlabel('pixel values', fontsize=18)
plt.ylabel('Number of pixels', fontsize=18)

x = blue[numpy.not_equal(blue,satdata.nodata)]
bins=18
color = 'lightgreen'

ax.hist(x,bins,color=color)
plt.show()
```

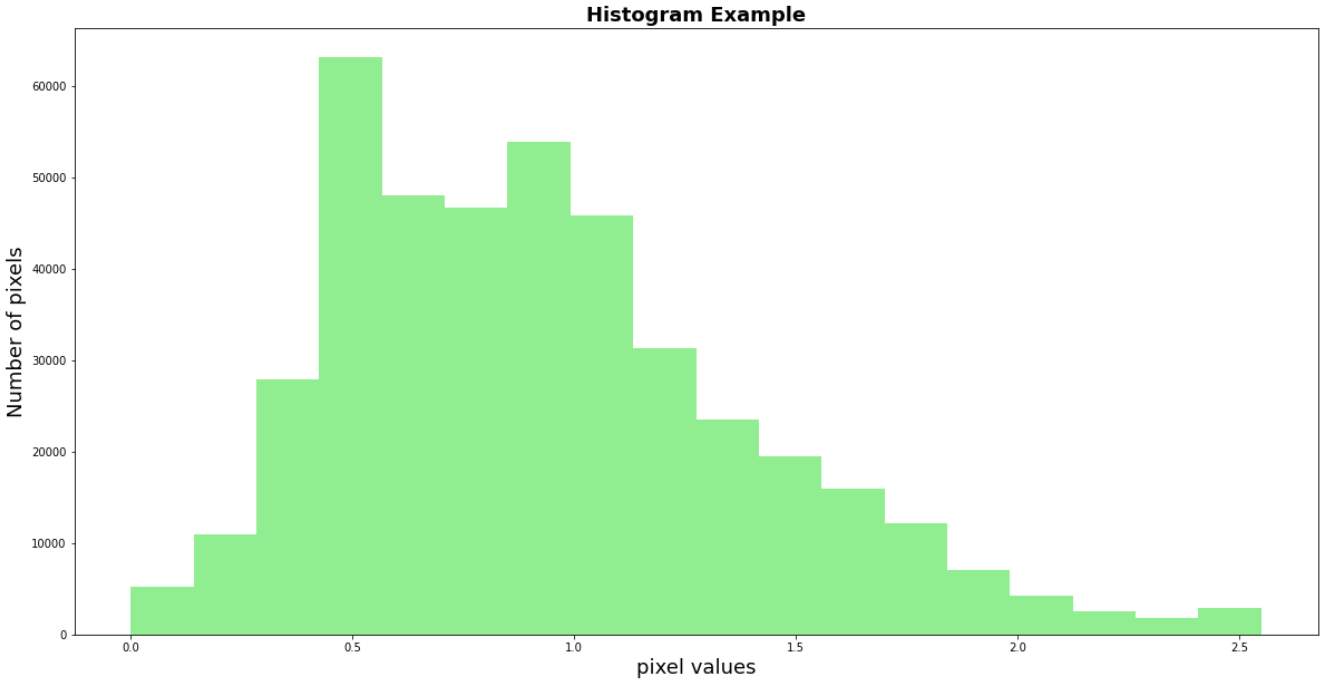


Image Segmentation using KMeans

In [32]:

```
from sklearn.cluster import KMeans
satdat.shape
```

Out[32]:

```
(554, 763)
```

In [33]:

```
address = '../input/aerialtif/aerial.tif'
image = rasterio.open(address).read()
image.shape
```

```
/opt/conda/lib/python3.7/site-packages/rasterio/__init__.py:207: NotGeoreferencedWarning: Dataset
has no geotransform, gcps, or rpcs. The identity matrix be returned.
  s = DatasetReader(path, driver=driver, sharing=sharing, **kwargs)
```

Out[33]:

```
(3, 554, 763)
```

In [35]:

```
import numpy as np
image = np.rollaxis(image,0,3)
image.shape
```

Out[35]:

```
(554, 763, 3)
```

In [37]:

```
RGB = image[...,3:0:-1]
print(RGB.min())
RGB.max()
```

```
0
```

Out[37]:

```
255
```

In [38]:

```
RGB = RGB / np.percentile(RGB,99)
RGB
```

Out[38]:

```
array([[[0.20425532, 0.18723404],
        [0.03829787, 0.04255319],
        [0.01276596, 0.02553191],
        ...,
        [0.39574468, 0.41702128],
        [0.42553191, 0.44680851],
        [0.51914894, 0.49787234]],

       [[0.18723404, 0.18723404],
        [0.05106383, 0.03829787],
        [0.01702128, 0.02553191],
        ...,
        [0.39574468, 0.41276596],
        [0.52340426, 0.54468085],
        [0.52340426, 0.52340426]],

       [[0.17446809, 0.19148936],
        [0.0212766 , 0.03404255],
        [0.00425532, 0.01702128],
        ...,
        [0.39148936, 0.41276596],
        [0.60425532, 0.62553191],
        [0.41276596, 0.43829787]],

       ...,

       [[0.20851064, 0.22978723],
        [0.26382979, 0.32765957],
```

```
        [0.27659574, 0.33191489],
        ...,
        [0.21276596, 0.22553191],
        [0.21276596, 0.22978723],
        [0.21276596, 0.21702128]],

       [[0.21702128, 0.22978723],
        [0.25531915, 0.32340426],
        [0.28510638, 0.33617021],
        ...,
        [0.2       , 0.19148936],
        [0.19574468, 0.1787234 ],
        [0.2       , 0.1787234 ]],

       [[0.21276596, 0.23404255],
        [0.26808511, 0.32765957],
        [0.26382979, 0.32765957],
        ...,
        [0.2       , 0.17021277],
        [0.18297872, 0.17446809],
        [0.18723404, 0.18297872]]])
```

In [39]:

```
height , width = image.shape[:2]
X = image.reshape((height*width,3))
X.shape
```

Out[39]:

```
(422702, 3)
```

In [40]:

```
kmeans = KMeans(n_clusters=4).fit(X)
kmeans.labels_.shape
```

Out[40]:

```
(422702,)
```

In [41]:

```
labels = kmeans.labels_.reshape((height,width))
labels.shape
```

Out[41]:

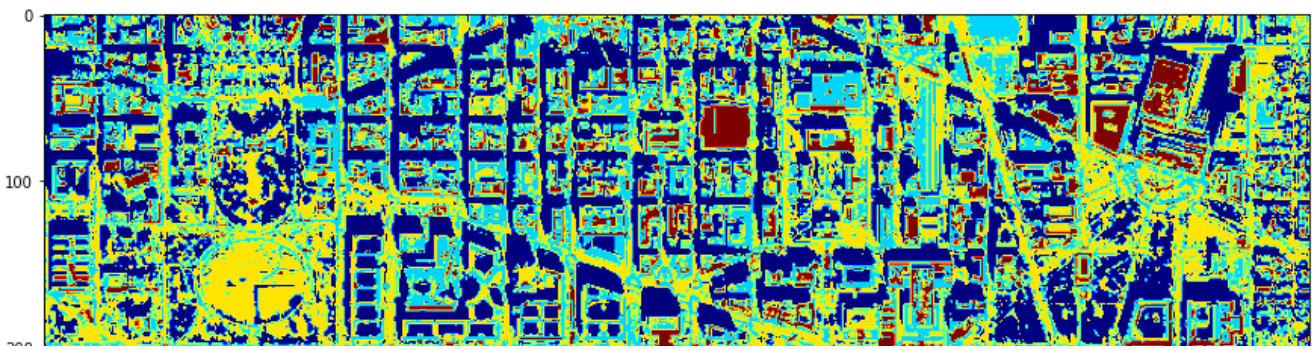
```
(554, 763)
```

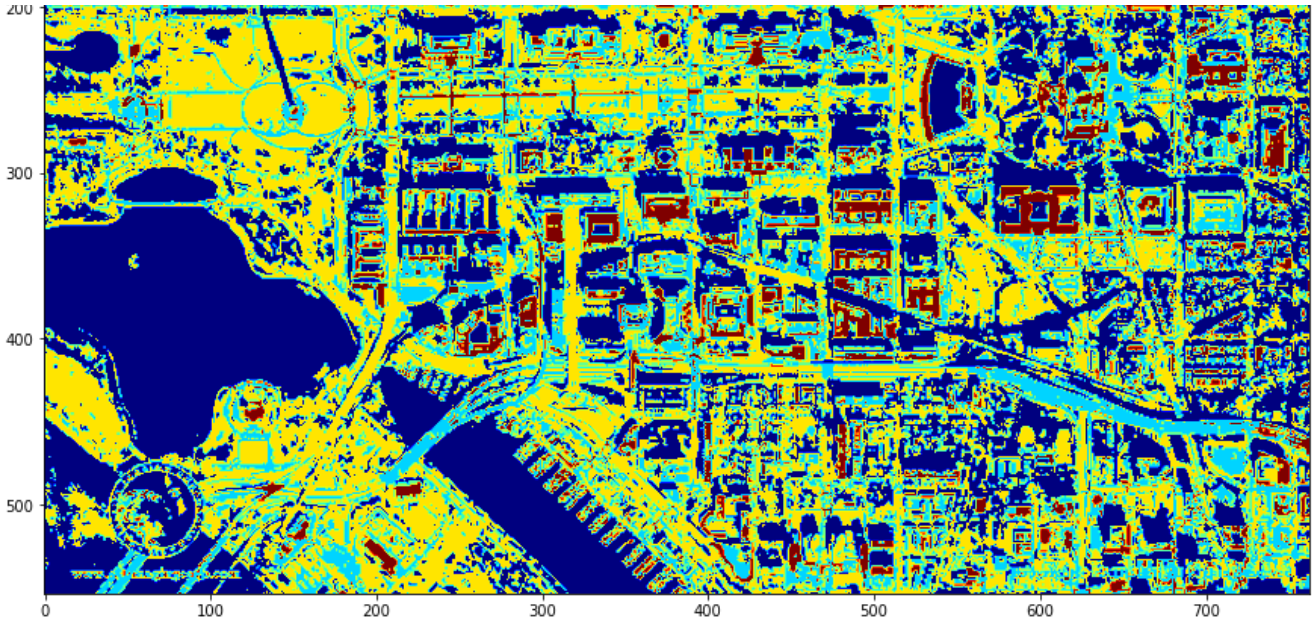
In [42]:

```
fig,ax = plt.subplots(figsize=(15,15))
ax.imshow(labels,cmap='jet')
```

Out[42]:

```
<matplotlib.image.AxesImage at 0x7f0669b4ec10>
```





In [43]:

```
import rasterio
from rasterio.plot import show
from rasterio.merge import merge
from rasterio  import plot
import glob
%matplotlib inline
import  os
```

In [45]:

```
dir_path = '../input/tiff-image/'
out_fp = r'../Mosaic.tif'

search_criteria  = '*.tif'
q = os.path.join(dir_path,search_criteria)
print(q)
```

```
../input/tiff-image/*.tif
```

In [46]:

```
Im_fps = glob.glob(q)
Im_fps
```

Out[46]:

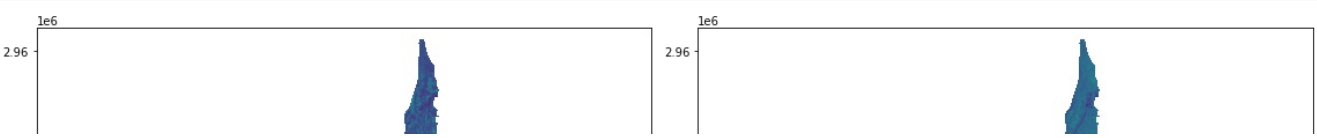
```
['../input/tiff-image/Band_5(ndvi).tif', '../input/tiff-image/Band_4(Red).tif']
```

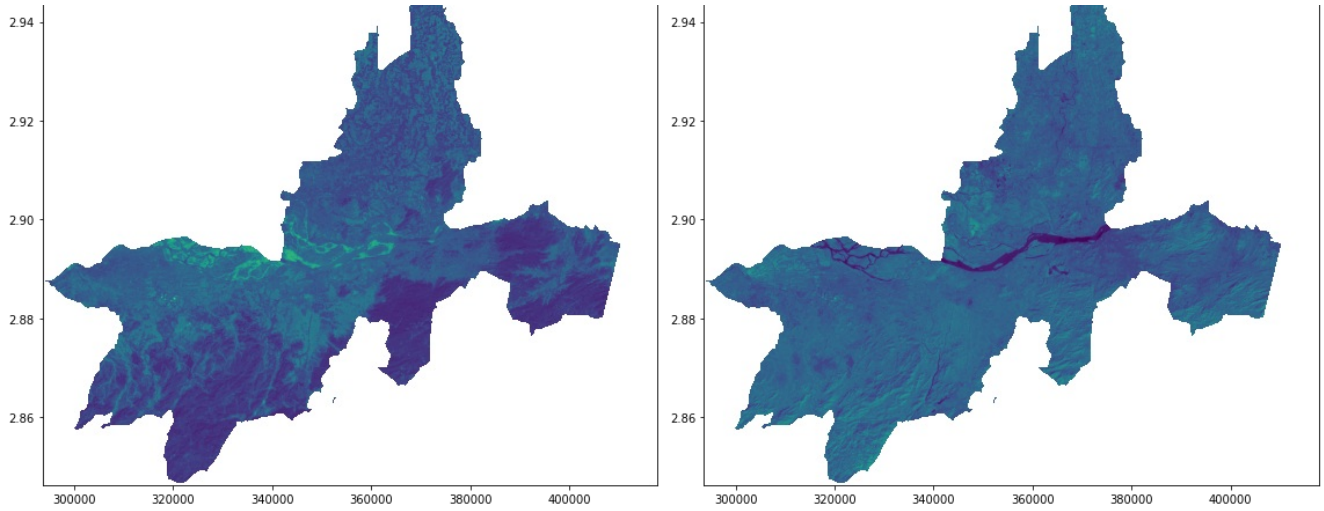
In [47]:

```
band5 = rasterio.open('../input/tiff-image/Band_5(ndvi).tif')
band4 = rasterio.open('../input/tiff-image/Band_4(Red).tif')
```

In [50]:

```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(16,9))
plot.show(band4,ax=ax1)
plot.show(band5,ax=ax2)
fig.tight_layout()
```





In [51]:

```
src_files_to_mosaic = []
```

In [52]:

```
for fp in Im_fps:
    src = rasterio.open(fp)
    src_files_to_mosaic.append(src)
    src_files_to_mosaic
```

merge those together and create a mosaic with rasterio's merge function

In [66]:

```
mosaic,out_trans = merge(src_files_to_mosaic)
```

In [69]:

```
out_meta = src.meta.copy()
out_meta.update({'driver':'GTiff',
                 'height':mosaic.shape[1],
                 'width': mosaic.shape[2],
                 'transform':out_trans,
                 'crs':'+proj=utm +zone=35 +ellps=GRS80 +units=m + no_defs'})
```

In [70]:

```
with rasterio.open(out_fp, "w", **out_meta) as dest:
    dest.write(mosaic)
```

In [ ]:

```
```