Geo Referenced Images in Python

Objectives

learn how to use python library call GDAL which is dedicated to geo referenced images handling and related function

Content

1) Installation and Importing GDAL Library

2) Reading Images using GDAL library

3) Writing Images using GDAL library

4) Subsetting (break images to tiles)

We will learn, how to bring raster image such as GeoTIFF file to our python workspace as NumPy matrix. that we can use for further analysis by a library call GDAL.

GDAL (Geospatial Data Abstraction Library) is a computer software library for reading and writing raster and vector geospatial data formats.

In this exercise, we will use GDAL library to read geo referenced TIFF image (GeoTIFF) to our python workspace as a NumPy matrix and run some operations on it

Eventhough, we use only GeoTIFF images for this exercise, GDAL library supports 100s of raster data formats. some of them are as follows

```
TIFF
JPEG, JPEG2000, PNG, GIF
ArcInfo grids, Imagine
ENVI, GRASS
HDF4, HDF5
and many more
```

1) Installation and Importing GDAL Library

First we have to install GDAL to the Python using pip3 command as below. with NumPy library we directly install from internet using pip3. but in case of GDAL version compatibility is sensitive. So first, we download the GDAL library and then install it using pip3 command

most of python libraries including GDAL can be downloaded from this link.

```
https://www.lfd.uci.edu/~gohlke/pythonlibs/
```

now we go to GDAL section

```
https://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal
```

then download GDAL version which is compatible with python version and OS (32bit or 64bit)

since we are using python 3.6 and

if you have windows 64 bit OS, first download "GDAL-2.3.3-cp36-cp36m-win_amd64.whl" file, navigate to downloaded directory and install using pip3

```
pip3 install GDAL-2.3.3-cp36-cp36m-win_amd64.wh
```

if you have windows 32 bit OS, first download "GDAL-2.3.3-cp36-cp36m-win32.whl" file, navigate to downloaded directory and install using pip3

```
pip3 install GDAL-2.3.3-cp36-cp36m-win32.whl
```

or if you use Anaconda, you can use, Anaconda Navigator to install GDAL library

then we can import and access various functions from GDAL library to our python work space

additionally install numpy and matplotlib libraries too

In [3]:

```python
'''import gdal, numpy and matplotlib library'''

import gdal
import numpy as np
import matplotlib.pyplot as plt
```

1) Reading Images using GDAL library

In this example, we will read sample image as GeoTIFF file which is most common geo referenced image file format

In [4]:

```
'''now we can read a tiff file and get information about the image'''

img = gdal.Open('MODIS_721-2017-02-06.tiff')

print(img.RasterXSize)
print(img.RasterYSize)
print(img.RasterCount)

print('\n---------------------\n')

print(img.GetProjection())

print('\n---------------------\n')

print(img.GetGeoTransform())

print('\n---------------------\n')
```

```
657
633
3

---------------------

GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTH
ORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORIT
Y["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],
AXIS["Latitude",NORTH],AXIS["Longitude",EAST],AUTHORITY["EPSG","4326"]]

---------------------

(103.392333984375, 0.002197265625, 0.0, 13.550537109374998, 0.0, -0.002197265
625)

---------------------
```

Note that, srtm_img.GetGeoTransform() contains 5 values, which are corresponding to geopraphic cordinates of image

```
[0] : top left x
[1] : w-e pixel resolution
[2] : rotation, 0 if image is "north up"
[3] : top left y
[4] : rotation, 0 if image is "north up"
[5] : n-s pixel resolution
```

In [5]:

```
'''read band from imported image and band information'''

band_1 = img.GetRasterBand(1) #index starts from 1

print(band_1.GetNoDataValue())
```

```
None
```

In [6]:

```
'''from band, we can read our data as numpy matrix that can be used for furt
her analysis'''

mat_band_1 = band_1.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize)
```

When we are reading data to a NumPy matrix, we pass 4 parameters to, which are corresponding to size of data chunk that we are reading

```
ReadAsArray(<xoff>, <yoff>, <xsize>, <ysize>)

xoff : starting image x cordinate
yoff : starting image y cordinate
xsize : width
ysize : height
```

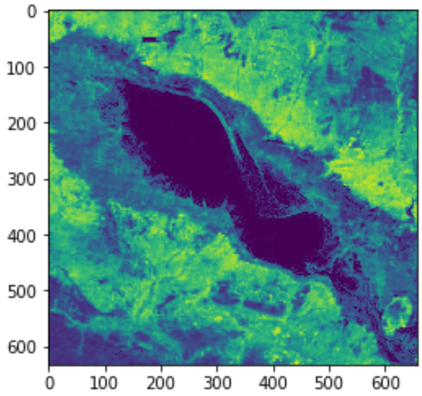
In [7]:

```
'''get information about matrix'''

print(type(mat_band_1))
print(mat_band_1.dtype)
print(mat_band_1.shape)

print('\n--------------------\n')

print(np.max(mat_band_1))
print(np.min(mat_band_1))
```

```
<class 'numpy.ndarray'>
uint8
(633, 657)

---------------------

195
0
```

In [12]:

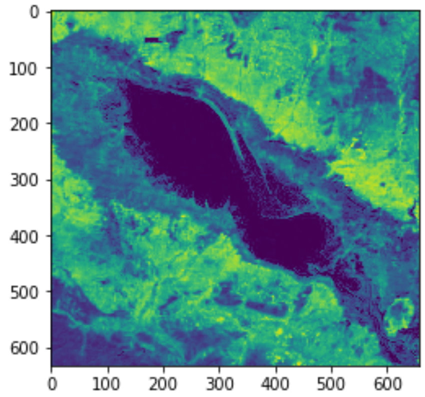
```
'''visualize image by matplotlib library'''

plt.imshow(mat_band_1)
plt.show()
```



Putting all together, how to read image to a NumPy matrix which we can be used for analysis

In [14]:

```
#read image
img = gdal.Open('MODIS_721-2017-02-06.tiff')

# get band
band_1 = img.GetRasterBand(1)

#get numpy matrix
mat_band_1 = band_1.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize)

#visualizing matrix
plt.imshow(mat_band_1)
plt.show()
```



3) Writing Images using GDAL library

In [20]:

```
'''lets read the image again as a numpy file'''

img = gdal.Open('MODIS_721-2017-02-06.tiff')
band_1 = img.GetRasterBand(1)
mat_band_1 = band_1.ReadAsArray(0, 0, img.RasterXSize, img.RasterYSize)

'''writing image to a file'''

# allocating space in hard drive
driver = gdal.GetDriverByName("GTiff")
outdata = driver.Create('out.tif', img.RasterXSize, img.RasterYSize, 1, gda
l.GDT_UInt16)

# set image paramenters (imfrormation related to cordinates)
outdata.SetGeoTransform(img.GetGeoTransform())
outdata.SetProjection(img.GetProjection())

# write numpy matrix as new band
outdata.GetRasterBand(1).WriteArray(mat_band_1)

# flush data from memory to harddrive
outdata.FlushCache()
outdata=None
```

GDAL also have their own data type like NumPy, so we have to provide data type, when image is created. commonly used data type are as follows

```
GDT_Unknown - Unknown or unspecified type
GDT_Byte - Eight bit unsigned integer
GDT_UInt16 - Sixteen bit unsigned integer
GDT_Int16 - Sixteen bit signed integer
GDT_UInt32 - Thirty two bit unsigned integer
GDT_Int32 - Thirty two bit signed integer
GDT_Float32 - Thirty two bit floating point
GDT_Float64 - Sixty four bit floating point
```

in this case, we use GDT_UInt16 as our data type

Exercise 1

Read band 2 again and add value 10 to band and write as float image

Exercise 2

Use threshold on band 2 (near infra red band) to extract watter and write as seperate image

Exercise 3

Use band 2 (near infra red band) and band 3 (red band) to calculate NDVI using (band2-band3)/(band2+band3) formula and write as seperate image

4) Subsetting (break images to tiles)

In [21]:

```
'''lets read the image again as a numpy file'''

img = gdal.Open('MODIS_721-2017-02-06.tiff')
band_1 = img.GetRasterBand(1)

'''write image as tiles'''

# set tile size
x_tile_size = y_tile_size = 50

# get original image cordinates information
img_GeoTran = img.GetGeoTransform()

# write nested for loop to go through x and y directions
for i1 in range(0, img.RasterXSize-x_tile_size, x_tile_size):
    for i2 in range(0, img.RasterYSize-y_tile_size, y_tile_size):

        # read part of the image (a tile)
        tile_band_1 = band_1.ReadAsArray(i1, i2, x_tile_size, x_tile_size)

        # allocating space in hard drive
        driver = gdal.GetDriverByName("GTiff")
        outdata = driver.Create('.//tiles//out_'+str(i1)+'-'+str(i2)+'.tif',
x_tile_size, y_tile_size, 1, gdal.GDT_UInt16)

        # calculate new cordinates corresponding to tiles, only top left x a
nd y will be changed
        subset_GeoTran = list(img_GeoTran)
        subset_GeoTran[0] = img_GeoTran[0] + i1 * img_GeoTran[1]
        subset_GeoTran[3] = img_GeoTran[3] + i2 * img_GeoTran[5]

        # set image paramenters (imfrormation related to cordinates)
        outdata.SetGeoTransform(subset_GeoTran)
        outdata.SetProjection(img.GetProjection())

        # write numpy matrix as new band
        outdata.GetRasterBand(1).WriteArray(tile_band_1)

        # flush data from memory to harddrive
        outdata.FlushCache()
        outdata=None
```

In [ ]: