Application of GIS with Python

Submitted By: Prakash Gharti Magar

Class Roll No.: PAS075BGE029

Submitted To Er. Saurav Gautam

Department of Geomatics Engineering

Tribhuvan University

Institute of Engineering

Pashchimanchal Campus

Department of Geomatics Engineering

Lamachaur, Pokhara

1. Read and compute the statistics of raster data.

File Edit Selection View Go Run Terminal Help FinalRasterStatistics.ipynb - gdalEnv - Visual Studio Code

FinalRasterStatistics.ipynb

D: > Course > Python Assignments > FinalRasterStatistics.ipynb > For All in One Statistics of Raster Data > Plotting Analysed Image > empty cell

+ Code + Markdown | Run All Clear Outputs of All Cells Restart Interrupt | Variables Outline ··· gdalEnv (Python 3.8.12)

Importing All Libraries

+ Code + Markdown

```python
from osgeo import gdal
import numpy as np
import matplotlib.pyplot as plt
import os
```

[29] ✓ 0.3s Python

Changing Current Directory

```python
os.chdir(r'D:\Course\Python Assignments\Geospatial Analysis using Python')
```

[30] ✓ 0.4s Python

Getting Statistics of Raster Data 'aresh.tif'

```python
dataset_source = gdal.Open('aresh.tif', 0)
print('Data Source:\t', dataset_source)
count = dataset_source.RasterCount
print("Raster Count:\t", count)
raster_band = dataset_source.GetRasterBand(1)
print("Raster Band:\t",raster_band)
response = {}  # For starage of data
for counter in range(1, count + 1):
    statistics = dataset_source.GetRasterBand(counter).GetStatistics(0, 1)
    response["Band: {}".format(counter)] = '''Minimum = %.3f, Maximum=%.3f, Mean=%.3f,
    Standard Deviation=%.3f''' % (statistics[0], statistics[1], statistics[2], statistics[3])
print("Statistics of Raster Data aresh.tif. \n", response)
```

[31] ✓ 0.6s Python

```
Data Source:   <osgeo.gdal.Dataset; proxy of <Swig Object of type 'GDALDatasetShadow *' at
0x0000019EB0467E70> >
Raster Count:  1
Raster Band:   <osgeo.gdal.Band; proxy of <Swig Object of type 'GDALRasterBandShadow *' at
0x0000019EB0472ED0> >
Statistics of Raster Data aresh.tif.
 {'Band: 1': 'Minimum = 178.000, Maximum=5789.000, Mean=1947.916, \n    Standard Deviation=1009.698'}
```

--- For More Information

Reading Other Data

```python
geo_transform = dataset_source.GetGeoTransform()
print('GeoTransform: ')
geo_transform   # Gives the Origin of Raster Data
```

[32] ✓ 0.3s Python

```python
geo_transform = dataset_source.GetGeoTransform()
print('GeoTransform: ')
geo_transform   # Gives the Origin of Raster Data
```

[32] ✓ 0.3s Python

```
GeoTransform:
```

```
(81.99986111111112,
 0.0002777777777777778,
 0.0,
 29.00013888888889,
 0.0,
 -0.0002777777777777778)
```

```python
print("Metadata: \n", dataset_source.GetMetadata())
```

[33] ✓ 0.8s Python

```
Metadata:
 {'AREA_OR_POINT': 'Point', 'DTED_CompilationDate': '0002', 'DTED_DataEdition': '02',
'DTED_DigitizingSystem': 'SRTM      ', 'DTED_HorizontalAccuracy': '0009', 'DTED_HorizontalDatum': 'WGS84',
'DTED_MaintenanceDate': '0000', 'DTED_MaintenanceDescription': '0000', 'DTED_MatchMergeDate': '0000',
'DTED_MatchMergeVersion': 'A', 'DTED_NimaDesignator': 'DTED2', 'DTED_OriginLatitude': '0280000N',
'DTED_OriginLongitude': '0820000E', 'DTED_Producer': 'USCNIMA ', 'DTED_RelHorizontalAccuracy': 'NA  ',
'DTED_RelVerticalAccuracy': '0018', 'DTED_SecurityCode_DSI': 'U', 'DTED_SecurityCode_UHL': 'U  ',
'DTED_UniqueRef_DSI': 'H09 054        ', 'DTED_UniqueRef_UHL': 'H09 054     ', 'DTED_VerticalAccuracy_ACC':
'0013', 'DTED_VerticalAccuracy_UHL': '0013', 'DTED_VerticalDatum': 'E96'}
```

For All in One Statistics of Raster Data

```python
!gdalinfo -stats aresh.tif
```

[34] ✓ 0.1s Python

```
Driver: GTiff/GeoTIFF
Files: aresh.tif
       aresh.tif.ovr
       aresh.tif.aux.xml
Size is 3601, 3601
Coordinate System is:
GEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4326]]
Data axis to CRS axis mapping: 2,1
Origin = (81.99986111111116,29.00013888888891)
```

```
    ID[EPSG,4326]]
Data axis to CRS axis mapping: 2,1
Origin = (81.999861111111116,29.000138888888891)
Pixel Size = (0.000277777777778,-0.000277777777778)
Metadata:
  AREA_OR_POINT=Point
  DTED_CompilationDate=0002
  DTED_DataEdition=02
  DTED_DigitizingSystem=SRTM
  DTED_HorizontalAccuracy=0009
  DTED_HorizontalDatum=WGS84
  DTED_MaintenanceDate=0000
  DTED_MaintenanceDescription=0000
  DTED_MatchMergeDate=0000
  DTED_MatchMergeVersion=A
  DTED_NimaDesignator=DTED2
  DTED_OriginLatitude=0280000N
  DTED_OriginLongitude=0820000E
  DTED_Producer=USCNIMA
  DTED_RelHorizontalAccuracy=NA
  DTED_RelVerticalAccuracy=0018
  DTED_SecurityCode_DSI=U
  DTED_SecurityCode_UHL=U
```

```
  DTED_Producer=USCNIMA
  DTED_RelHorizontalAccuracy=NA
  DTED_RelVerticalAccuracy=0018
  DTED_SecurityCode_DSI=U
  DTED_SecurityCode_UHL=U
  DTED_UniqueRef_DSI=H09 054
  DTED_UniqueRef_UHL=H09 054
  DTED_VerticalAccuracy_ACC=0013
  DTED_VerticalAccuracy_UHL=0013
  DTED_VerticalDatum=E96
Image Structure Metadata:
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  (  81.9998611,  29.0001389) ( 81d59'59.50"E, 29d 0' 0.50"N)
Lower Left  (  81.9998611,  27.9998611) ( 81d59'59.50"E, 27d59'59.50"N)
Upper Right (  83.0001389,  29.0001389) ( 83d 0' 0.50"E, 29d 0' 0.50"N)
Lower Right (  83.0001389,  27.9998611) ( 83d 0' 0.50"E, 27d59'59.50"N)
Center      (  82.5000000,  28.5000000) ( 82d30' 0.00"E, 28d30' 0.00"N)
Band 1 Block=3601x1 Type=Int16, ColorInterp=Gray
  Min=178.000 Max=5789.000
  Minimum=178.000, Maximum=5789.000, Mean=1947.916, StdDev=1009.698
  NoData Value=-32767
  Overviews: 1801x1801, 901x901, 451x451, 226x226
```

```
Upper Left  (  81.9998611,  29.0001389) ( 81d59'59.50"E, 29d 0' 0.50"N)
Lower Left  (  81.9998611,  27.9998611) ( 81d59'59.50"E, 27d59'59.50"N)
Upper Right (  83.0001389,  29.0001389) ( 83d 0' 0.50"E, 29d 0' 0.50"N)
Lower Right (  83.0001389,  27.9998611) ( 83d 0' 0.50"E, 27d59'59.50"N)
Center      (  82.5000000,  28.5000000) ( 82d30' 0.00"E, 28d30' 0.00"N)
Band 1 Block=3601x1 Type=Int16, ColorInterp=Gray
  Min=178.000 Max=5789.000
  Minimum=178.000, Maximum=5789.000, Mean=1947.916, StdDev=1009.698
  NoData Value=-32767
  Overviews: 1801x1801, 901x901, 451x451, 226x226
  Unit Type: m
  Metadata:
    STATISTICS_MAXIMUM=5789
    STATISTICS_MEAN=1947.9155001146
    STATISTICS_MINIMUM=178
    STATISTICS_STDDEV=1009.6982631214
    STATISTICS_VALID_PERCENT=100
```

```
array = raster_band.ReadAsArray()
print("Raster Band Array: \n", array)
```

[35] ✓ 0.8s Python

```
Raster Band Array:
 [[2952 2973 2994 ... 5032 5046 5061]
 [2948 2962 2977 ... 5029 5042 5065]
 [2943 2951 2959 ... 5036 5048 5068]
 ...
 [ 212  216  219 ... 1179 1171 1160]
 [ 213  219  226 ... 1186 1177 1164]
 [ 218  226  233 ... 1190 1184 1170]]
```

Applying Analysis

```python
binary_mask = np.where((array >= np.mean(array)), 1, 0)
print("Binary Mask: \n", binary_mask)
```

✓ 0.1s Python

```
Binary Mask:
 [[1 1 1 ... 1 1 1]
 [1 1 1 ... 1 1 1]
 [1 1 1 ... 1 1 1]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Plotting Analysed Image

```python
plt.Figure()
plt.imshow(binary_mask)
```

[37] ✓ 0.7s Python

```
<matplotlib.image.AxesImage at 0x19eb800b220>
```


```
<matplotlib.image.AxesImage at 0x19eb800b220>
```

2. Perform layer stacking and mosaicing lab from Geoprocessing with Python book.

File Edit Selection View Go Run Terminal Help — Layer Stacking and Mosaicing.ipynb - gdalEnv - Visual Studio Code

Raster (1).ipynb | Layer Stacking and Mosaicing.ipynb | FinalRasterStatistics.ipynb | Vector.ipynb

D: > Course > Python Assignments > Layer Stacking and Mosaicing.ipynb > Importing All Required Libraries > from osgeo import gdal

+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ··· gdalEnv (Python 3.8.12)

Importing All Required Libraries

+ Code + Markdown

```python
from osgeo import gdal
import os
import matplotlib.pyplot as plt
```

[4] ✓ 0.9s Python

Changing Directory and Opening Data Source

```python
os.chdir(r'D:\Course\Python Assignments\Geospatial Analysis using Python\Raster Data\True Color Band')
dataset_source = gdal.Open('Band 2.tif')
```

[5] ✓ 0.2s Python

```python
band = dataset_source.GetRasterBand(1)
print(band)
count = dataset_source.RasterCount
count
```

⊗ 0 ⚠ 0 — Jupyter Server: Local — Cell 2 of 22 — Go Live

```
<osgeo.gdal.Band; proxy of <Swig Object of type 'GDALRasterBandShadow *' at 0x0000022A0464BA80> >
```

```
1
```

+ Code + Markdown

Reading Information and Plotting Raster Data

```python
projection = dataset_source.GetProjection()
metadata = dataset_source.GetMetadata()
print(metadata)
band = dataset_source.GetRasterBand(1)
array = band.ReadAsArray()
array
graph = plt.imshow(array, cmap='twilight')
```

[7] ✓ 2.5s Python

```
{'AREA_OR_POINT': 'Point'}
```

```python
plt.figure()
plt.imshow(array)
```

[8] ✓ 1.4s Python

```
<matplotlib.image.AxesImage at 0x22a1bb5db80>
```

All in One Information about Raster Data

```python
!gdalinfo "Band 2.TIF"
```

[9] ✓ 0.1s Python

0 Jupyter Server: Local Cell 2 of 22 Go Live

All in One Information about Raster Data

```
!gdalinfo "Band 2.TIF"
```

[9] ✓ 0.1s Python

```
Driver: GTiff/GeoTIFF
Files: Band 2.TIF
Size is 7591, 7741
Coordinate System is:
PROJCRS["WGS 84 / UTM zone 44N",
    BASEGEOGCRS["WGS 84",
        DATUM["World Geodetic System 1984",
            ELLIPSOID["WGS 84",6378137,298.257223563,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4326]],
    CONVERSION["UTM zone 44N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
```

```
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",81,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
```

```
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["Engineering survey, topographic mapping."],
        AREA["Between 78°E and 84°E, northern hemisphere between equator and 84°N, onshore and offshore.
China. India. Kazakhstan. Kyrgyzstan. Nepal. Russian Federation. Sri Lanka."],
        BBOX[0,78,84,84]],
    ID["EPSG",32644]]
Data axis to CRS axis mapping: 1,2
Origin = (504285.000000000000000,3150315.000000000000000)
Pixel Size = (30.000000000000000,-30.000000000000000)
Metadata:
  AREA_OR_POINT=Point
Image Structure Metadata:
  COMPRESSION=DEFLATE
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  (  504285.000, 3150315.000) ( 81d 2'37.59"E, 28d28'45.98"N)
Lower Left  (  504285.000, 2918085.000) ( 81d 2'34.64"E, 26d22'58.30"N)
Upper Right (  732015.000, 3150315.000) ( 83d22'10.11"E, 28d27'31.69"N)
Lower Right (  732015.000, 2918085.000) ( 83d19'30.62"E, 26d21'50.34"N)
Center      (  618150.000, 3034200.000) ( 82d11'43.20"E, 27d25'34.00"N)
```

```
Corner Coordinates:
Upper Left  (  504285.000, 3150315.000) ( 81d 2'37.59"E, 28d28'45.98"N)
Lower Left  (  504285.000, 2918085.000) ( 81d 2'34.64"E, 26d22'58.30"N)
Upper Right (  732015.000, 3150315.000) ( 83d22'10.11"E, 28d27'31.69"N)
Lower Right (  732015.000, 2918085.000) ( 83d19'30.62"E, 26d21'50.34"N)
Center      (  618150.000, 3034200.000) ( 82d11'43.20"E, 27d25'34.00"N)
Band 1 Block=256x256 Type=UInt16, ColorInterp=Gray
  NoData Value=0
  Overviews: 3796x3871, 1898x1936, 949x968, 475x484, 238x242, 119x121
```

File Edit Selection View Go Run Terminal Help — Layer Stacking and Mosaicing.ipynb - gdalEnv - Visual Studio Code

Raster (1).ipynb | Layer Stacking and Mosaicing.ipynb | FinalRasterStatistics.ipynb | Vector.ipynb

D: > Course > Python Assignments > Layer Stacking and Mosaicing.ipynb > Importing All Required Libraries > from osgeo import gdal

+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ··· gdalEnv (Python 3.8.12)

Importing ALl Libraries For Layer Stacking of Raster Data

```python
import os
from osgeo import gdal
```

[10] ✓ 0.3s Python

```python
band2 = 'Band 2.TIF'
band3 = 'Band 3.TIF'
band4 = 'Band 4.TIF'
dataset = gdal.Open(band2)
input_band = dataset.GetRasterBand(1)
```

[11] ✓ 0.7s Python

Creating Driver Which helps to Create New Band

```python
driver = gdal.GetDriverByName('GTiff')
```

```python
output_dataset = driver.Create('true_color.tif', input_band.XSize, input_band.YSize, 3, input_band.DataType)
```

[13] ✓ 0.1s Python

Getting Some Information

```python
projection = dataset.GetProjection()
output_dataset.SetProjection(projection)
geo_transform = dataset.GetGeoTransform()
output_dataset.SetGeoTransform(geo_transform)
```

[14] ✓ 0.5s Python Python

```
0
```

Process of Layer Stacking

```python
input_data = input_band.ReadAsArray()
```

```python
input_data = input_band.ReadAsArray()
output_band = output_dataset.GetRasterBand(3)
output_band.WriteArray(input_data)

dataset = gdal.Open(band3)
output_band = output_dataset.GetRasterBand(2)
output_band.WriteArray(dataset.ReadAsArray())
```

[15] ✓ 0.9s Python

```
0
```

```python
output_dataset.GetRasterBand(1).WriteArray(gdal.Open(band4).ReadAsArray())
output_dataset.FlushCache()
for i in range(1, 4):
    output_dataset.GetRasterBand(i).ComputeStatistics(False)
output_dataset.BuildOverviews('average', [2, 4, 8, 16, 32])
del output_dataset
```

[16] ✓ 1.9s Python

Result: Layer Stacking (True Color)

Mosaicing of Raster Data:

Result: Mosaicing

3. Perform analysis on a Landsat/Sentinel image of your area of interest and submit code with an output screenshot.

NDVI(Normalized Difference Vegetative Index) Calculation

File Edit Selection View Go Run Terminal Help NDVI_Calculation.ipynb - gdalEnv - Visual Studio Code

Layer Stacking and Mosaicing.ipynb | listing10_2.py 1 | mosaicing.py | NDVI_Calculation.ipynb | FinalRasterStatistics

D: > Course > Python Assignments > Geospatial Analysis using Python > Raster Data > NDVI_Calculation.ipynb > NDVI (Normalized Difference Vegetative Index) Calcula

+ Code + Markdown | Run All Clear Outputs of All Cells Restart Interrupt | Variables Outline ··· gdalEnv (Python 3.8.12)

Importing All Required Library for NDVI Calculation

```python
from osgeo import gdal_array as ga
from osgeo import gdal, ogr, os, osr
import numpy as np
```

[10] Python

Changing Directory

```python
os.chdir(r'D:\Course\Python Assignments\Geospatial Analysis using Python\Raster Data\True Color Band')
```

[11] Python

Required Bands

⊗ 0 ⚠ 1 Jupyter Server: Local Cell 10 of 10 Go Live

[11] Python

Required Bands

```python
band3 = 'Band 3.TIF'
band4 = 'Band 4.TIF'
```

[13] Python

Getting Array of Pixel Value of Each Band

```python
array3 = ga.LoadFile(band3)
array4 = ga.LoadFile(band4)
```

[ ] Python

NDVI Calculation

NDVI (Normalized Difference Vegetative Index) Calculation

```python
ga.numpy.seterr(all = "ignore")
ndvi = ((array4 - array3) * 1.0) / ((array4 + array3) * 1.0)
ndvi1 = ga.numpy.nan_to_num(ndvi)
final_ndvi = 'NDVI.tif'
out = ga.SaveArray(ndvi1, final_ndvi, format = "GTiff", prototype = band4)
out = None
```

Python

Result: