A Stand-Alone NDVI Tool using Python 3.7, GDAL, NumPy and Matplotlib

GISC 3200K
PROGRAMMING FOR GEOSPATIAL SCIENCE AND TECHNOLOGY
DR. HUIDAE CHO
PRESENTED BY MEGHAN TOUAT

A Stand-Alone NDVI Tool using Python 3.7, GDAL, NumPy and Matplotlib

Meghan Touat
Lewis F. Rogers Institute for Environmental and Spatial Analysis

GISC 3200K
Programming for Geospatial Science and Technology
Dr. Huidae Cho

Abstract

The objective of this project was to determine if NDVI could accurately be performed on a Landsat 8 GeoTiff image without using ESRI software or API's. A NDVI Python script was created that reads a Landsat 8 GeoTiff file with GDAL, assigns the red and near infrared bands to NumPy arrays, calculates NDVI, creates a new GeoTiff and plots it using Matplotlib. All values within the plot are between -1 and 1, since NDVI is a ratio of red to near infrared.

$$NDVI = \frac{(NIR - Red)}{(NIR + Red)}$$

Background

Python is an object-oriented programming language invented in 1991. It's known mainly for its ease of use, readability and adaptability. For this project, version 3.7 was used.

Geospatial Data Abstraction Library (**GDAL**) is a library for reading and writing raster and vector geospatial data created by the Open Source Geospatial Foundation (OSGeo).

NumPy is a common Python library used for multi-dimensional arrays and matrices, and for its high level mathematical functions.

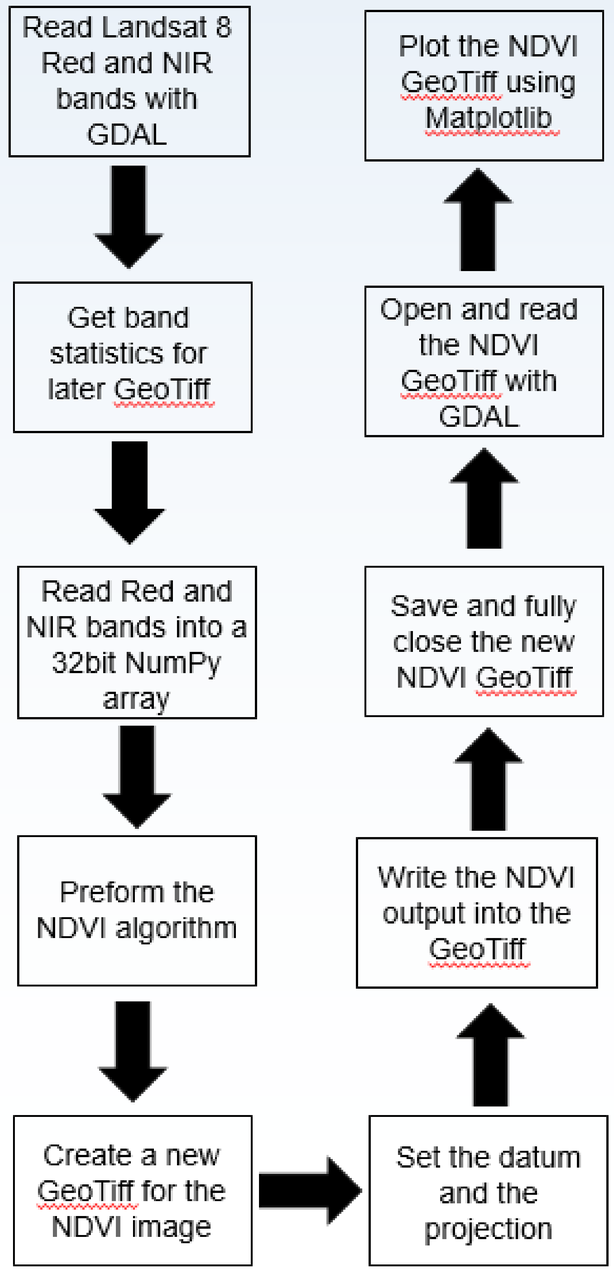
Matplotlib is a 2D plotting library Python and provides an object-oriented API for integrating plots with GUI tools like Tkinter.

Normalized Difference Vegetation Index (**NDVI**) quantifies vegetation by measuring the difference between near-infrared (which vegetation strongly reflects) and red light (which vegetation absorbs)

Objectives

The objective of this project was to determine if NDVI could accurately be performed on a Landsat 8 GeoTiff image without using ESRI software or API's. Another objective was determining if a complete and functioning Python code could be compiled that was also accurate and useful in the day to day life of a Geospatial Analyst.

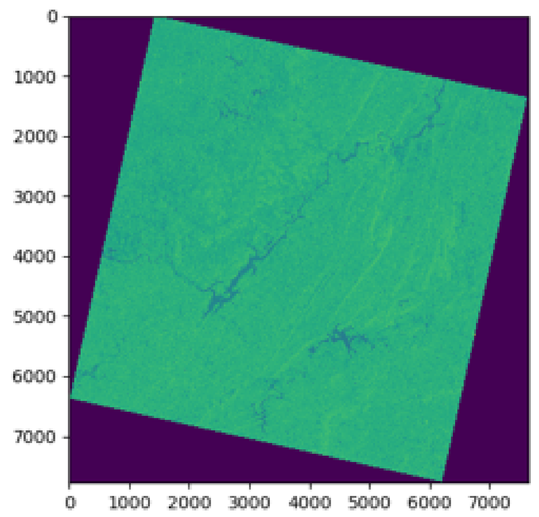
Methods



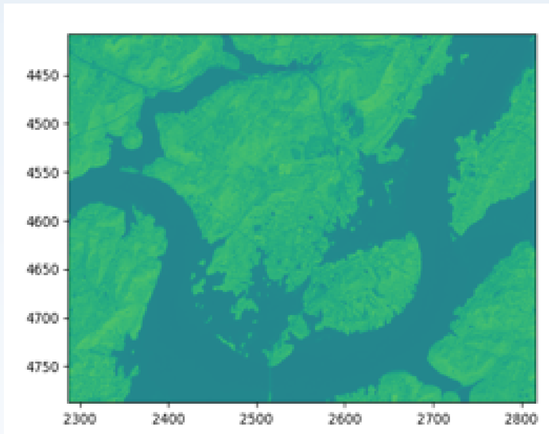
Results

The results of this script are as expected. There was a "true divide by zero" warning which occurs when the NDVI algorithm is running, but the tool still plots the raster effectively. When added as a dataset in ESRI software, there are no issues with datum or projection, but the GeoTiff does import as black and white instead of the teal/purple color scheme added by Matplotlib. A "gotcha" was found when creating the new NDVI GeoTiff, in that the file must be created, saved and closed with the NDVI NumPy array data before plotting successfully. This is due to a catch in the Matplotlib library; there must be a "completed" GeoTiff image before it can be plotted, thus GDAL was again used to read in the file.

There was also some difficulty with managing data types within the NumPy arrays. It generally appears that GDAL, NumPy and Matplotlib work best in unison when the array data type is a single precision float (sign bit, 8 bits exponent, 23 bits mantissa), as opposed to an integer or even double precision float. A "np.where" clause was also implemented to process the black mask (no data) values Landsat 8 tiles typically have. The NumPy clause parses the cells, and if the cell value is not 0, the NDVI algorithm is performed. However, if the cell value is 0, it is changed to a value of -1, or no data. Because of this, the Landsat 8 mask appears purple, with a value of -1. All other cells within the image thus contain values ranging from -0.99 to 1, as per typical NDVI standards. Statistics were calculated for the NDVI GeoTiff to measure the accuracy and precision of the NDVI algorithm when used in combination with the "np.where" argument.



Results Cont...



Conclusion

By using all open source software, a better understanding of Python and the geospatial functioning of the NDVI tool was gained. Since all open source software is free, innovation and customization for this project allowed the exploration of different libraries in order to determine the best method of calculating NDVI. This NDVI script is not only simple, but extremely effective in producing an accurate 2D NDVI GeoTiff analysis without the use or ESRI software or other API's.

Further research for this project includes: removing the Landsat 8 mask, adding a coordinate system to the axes, changing the color scheme of the plot, and potentially pansharpening the Landsat 8 tile to achieve a 15m NDVI rather than a 30m one. In addition to this output, a historical NDVI analysis could be completed for this tile, due to the high temporal resolution of the Landsat 8 satellite. A simple NDVI Python toolbox could be created as well, in order to streamline data processing.

References

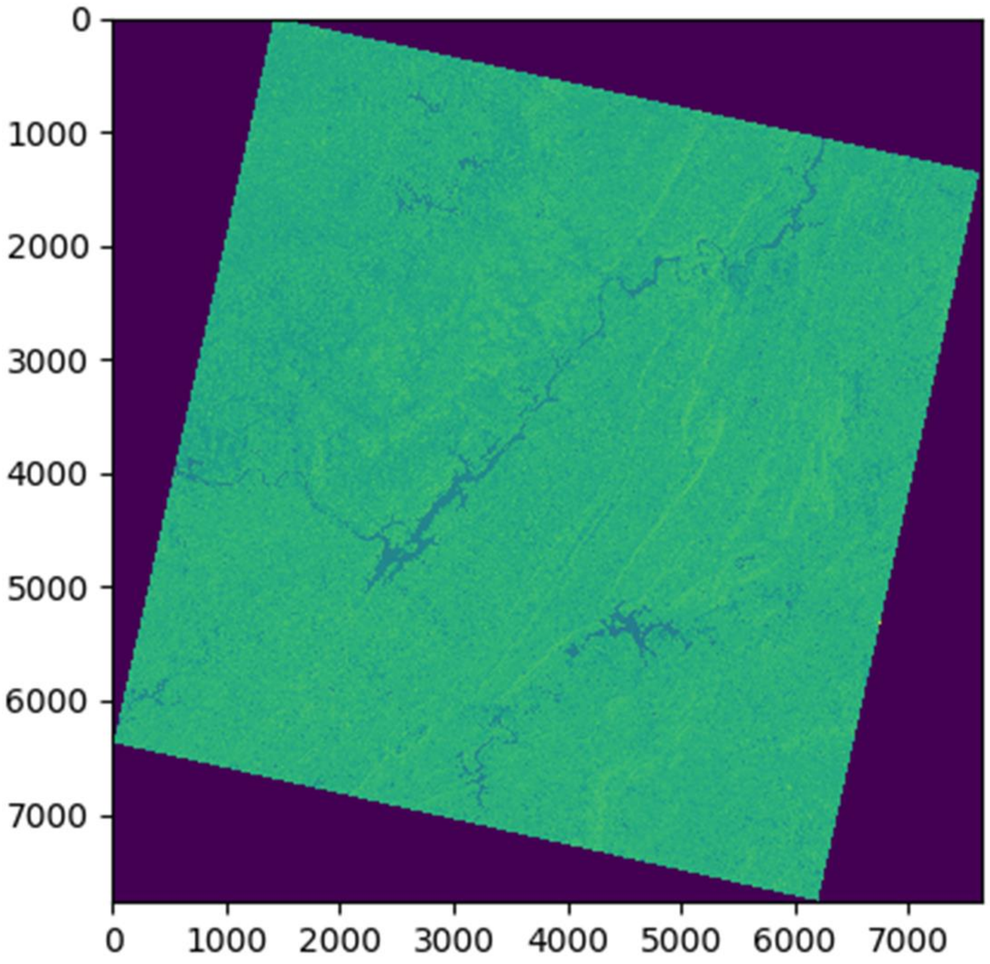
Dr. Huidae Cho

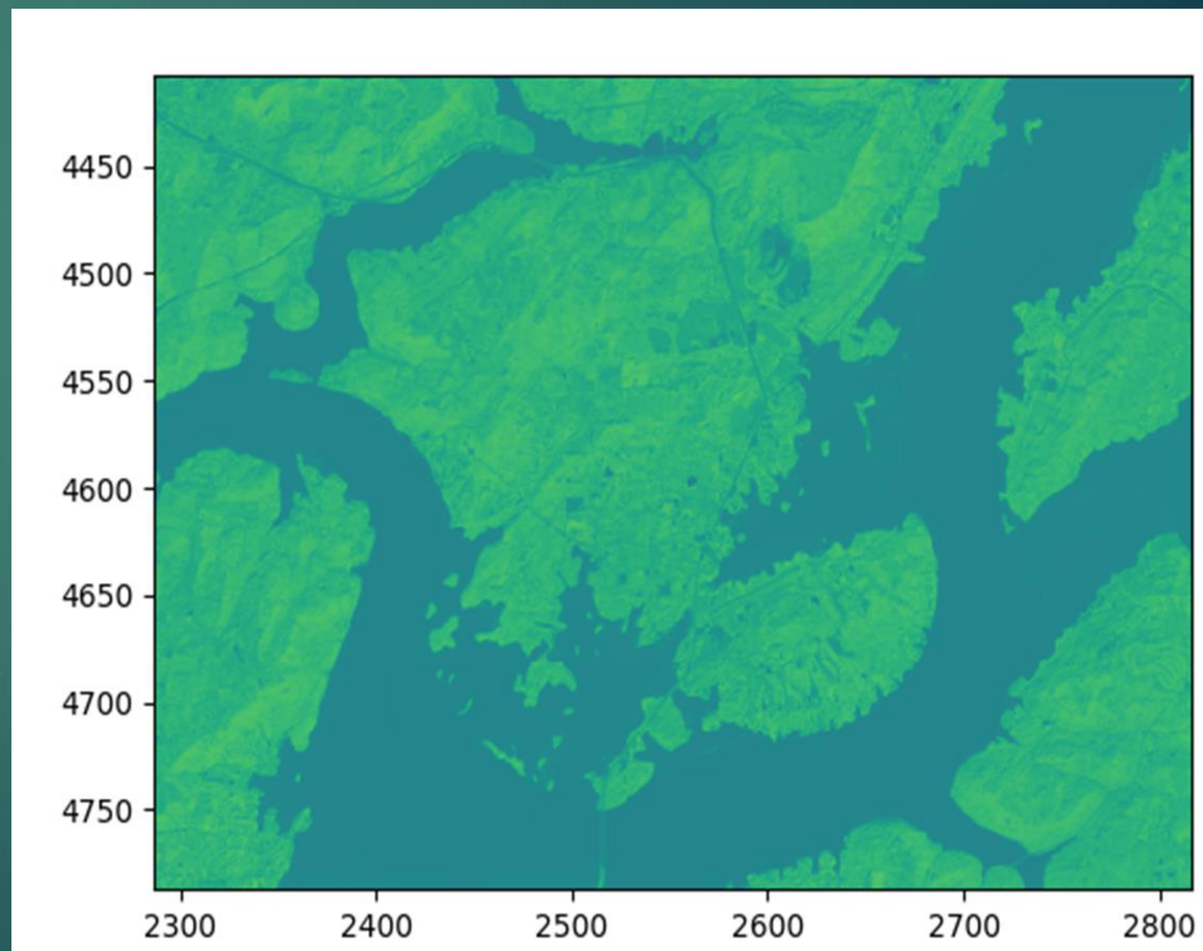
Zac Miller

The UNG IESA Department

www.gdal.org (GDAL documentation page)

```
import gdal
import numpy as np
import matplotlib.pyplot as plt
#https://gdal.org/tutorials/raster_api_tut.html

#Open the Image
red = gdal.Open("LC08_L1TP_020036_20191103_20191115_01_T1_B4.tif")
NIR = gdal.Open("LC08_L1TP_020036_20191103_20191115_01_T1_B5.tif")

#Get info from red band
cols = red.RasterXSize #number columns
rows = red.RasterYSize #number rows
proj = red.GetProjection() #get projection
trans = red.GetGeoTransform() #get transformation
bands = red.RasterCount #gives number of bands
meta = red.GetMetadata() #gives metadata of the file (optional)
#we don't have to repeat this code^^ for NIR because all band info should be identical

#Read the Red band and the Near Infrared band as 32bt floating numpy arrays
band4 = red.GetRasterBand(1).ReadAsArray().astype(np.float32)
band5 = NIR.GetRasterBand(1).ReadAsArray().astype(np.float32)

#do the NDVI math
n1 = (band5 - band4)
n2 = (band5 + band4)

np.seterr(divide='ignore', invalid='ignore')#allow it to divide by zero, since we'll have negative values

#if results aren't from -1 to 1 (values of the black background), use this:
ndvi = np.where (n2 != 0,(n1)/(n2), -1)

#Create the output NDVI image
driver = gdal.GetDriverByName('GTiff') #GTiff is the read only driver for this specific instance
ndvi_image = driver.Create("NDVI_image.tif", cols, rows, 1, gdal.GDT_Float32) #now create new file. file name, cols(x size), rows(y size), bands, array type

#Set the datum and the projection
ndvi_image.SetGeoTransform(trans) #give it the same datum as before
ndvi_image.SetProjection(proj) #give it the same projection as before

#Write the data into band 1 of the new image
ndvi_image.GetRasterBand(1).WriteArray(ndvi)
stat = ndvi_image.GetRasterBand(1).GetStatistics(1,1) #GetStatistics(0,1) will return Min, Max, Mean, StdDev
ndvi_image.GetRasterBand(1).SetStatistics(stat[0], stat[1], stat[2], stat[3]) #set stats from above as indexed values

#save and close ndvi_image
ndvi_image = None #this is apparently a "gotcha" and must be done
#now open the new image
ndvi_image = gdal.Open("NDVI_image.tif")
#read it in
ndvi_image = ndvi_image.GetRasterBand(1).ReadAsArray().astype(np.float32)
#plot the NDVI image
plt.imshow(ndvi_image) #prepares ndvi image data to be shown
plt.show()
```


0
1000
2000
3000
4000
5000
6000
7000
0
1000
2000
3000
4000
5000
6000
7000


4450
4500
4550
4600
4650
4700
4750
2300
2400
2500
2600
2700
2800