MAPPING THE URBAN FOOTPRINT OF RALEIGH, NC

Authors: Shengjie Liu

(OPTIONAL) CALCULATION WITH PYTHON

For those with a computing background, it may be easier to use Python to handle a large amount of raster data. For this purpose, we provide an alternative way to calculate the urban footprint of Raleigh, NC. We expect readers of this document are with a basic Python background (numpy, matplotlib, etc.). The Anaconda environment is encouraged. In this document, we will install an Anaconda-Python 3.8 environment from scratch.

Although it is possible to read raster data from geodatabase directly, it requires some packages that are difficult to install. In this document, we will first export the raster data as GeoTIFF files, and then read the GeoTIFF files in Python. GeoTIFF is a standard raster data format, just like shapefile for vector data

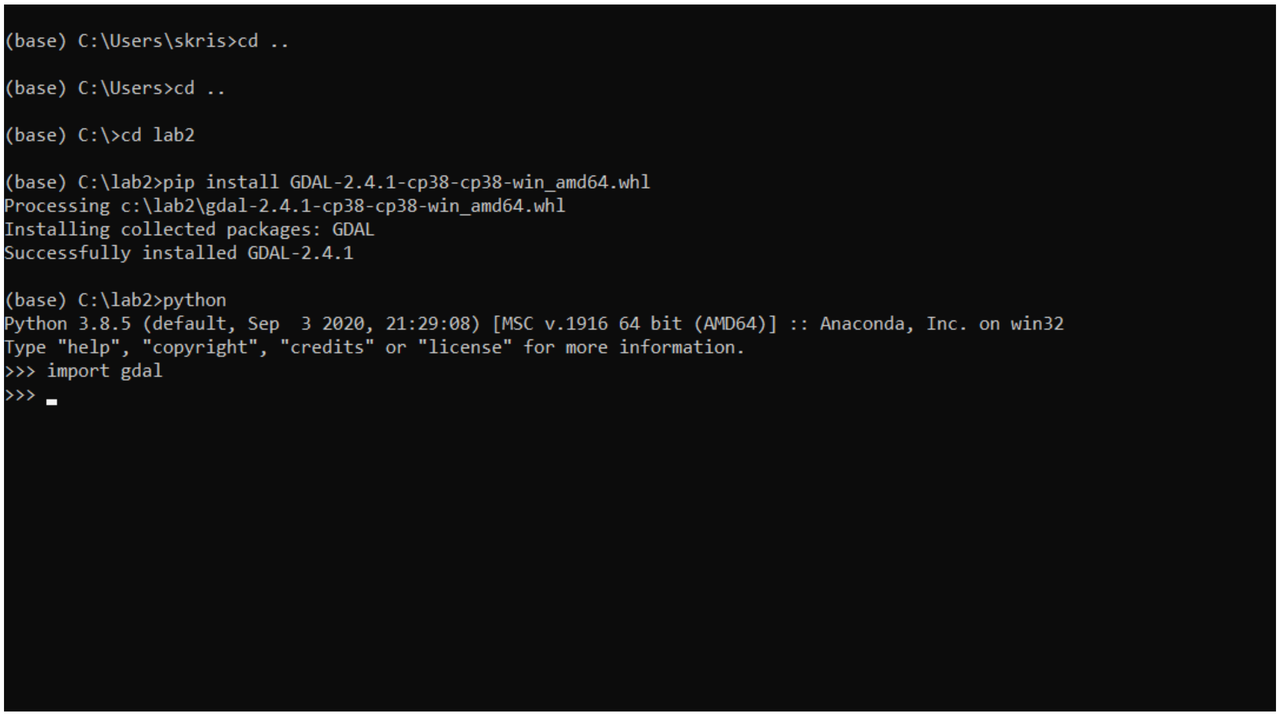
INSTRUCTIONS

1. Export raster data from geodatabase. Open the raster data in ArcGIS Pro, right click the raster data, select *Data* then *Export Data*. In the export setting, change the location (saved folder) and name (file name) accordingly. Leave other settings as default. You will need to do so for the four raster datasets. You can also download the GeoTIFF files from Blackboard.

2. Prepare Python environment. Download and install the Anaconda-Python 3.8 package ("**https://repo.anaconda.com/archive/**"). The version used here is "Anaconda3-2020.11-Windows-x86_64.exe". If you have previously installed Python, keep using your original Python environment should be okay.

3. We will need a Python package named "GDAL" for reading the GeoTIFF data. You can try the normal way (via pip or conda) to install the GDAL package, but it is likely to fail if you are using Windows. In this document, we download the whl package directly from this web page ("**https://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal**"). The version tested was "GDAL-2.4.1-cp38-cp38-win_amd64.whl" as the test computer used Python 3.8 64-bit version on Windows 10. Newer versions of GDAL may not be working. You should download the package based on your system environment. After downloading the whl package, in Anaconda Prompt (Python 3) from the windows Start menu (Anaconda 3 – Anaconda Prompt) or your Python environment, CD to the location where you saved the downloaded whl package, "pip install GDAL-2.4.1-cp38-cp38-win_amd64.whl" (see below image). This will install GDAL from the downloaded whl package. You may need to update pip before doing so. You can test if GDAL was installed successfully by importing GDAL following the figure below. If you cannot install GDAL, another alternative way is to open GeoTIFF via "from PIL import Image" following this stackoverflow ("**https://stackoverflow.com/questions/7569553/working-**

with-tiffs-import-export-in-python-using-numpy”), but you won’t be able to read the GeoTransform and Projection information later (luckily they are not necessary for this assignment).

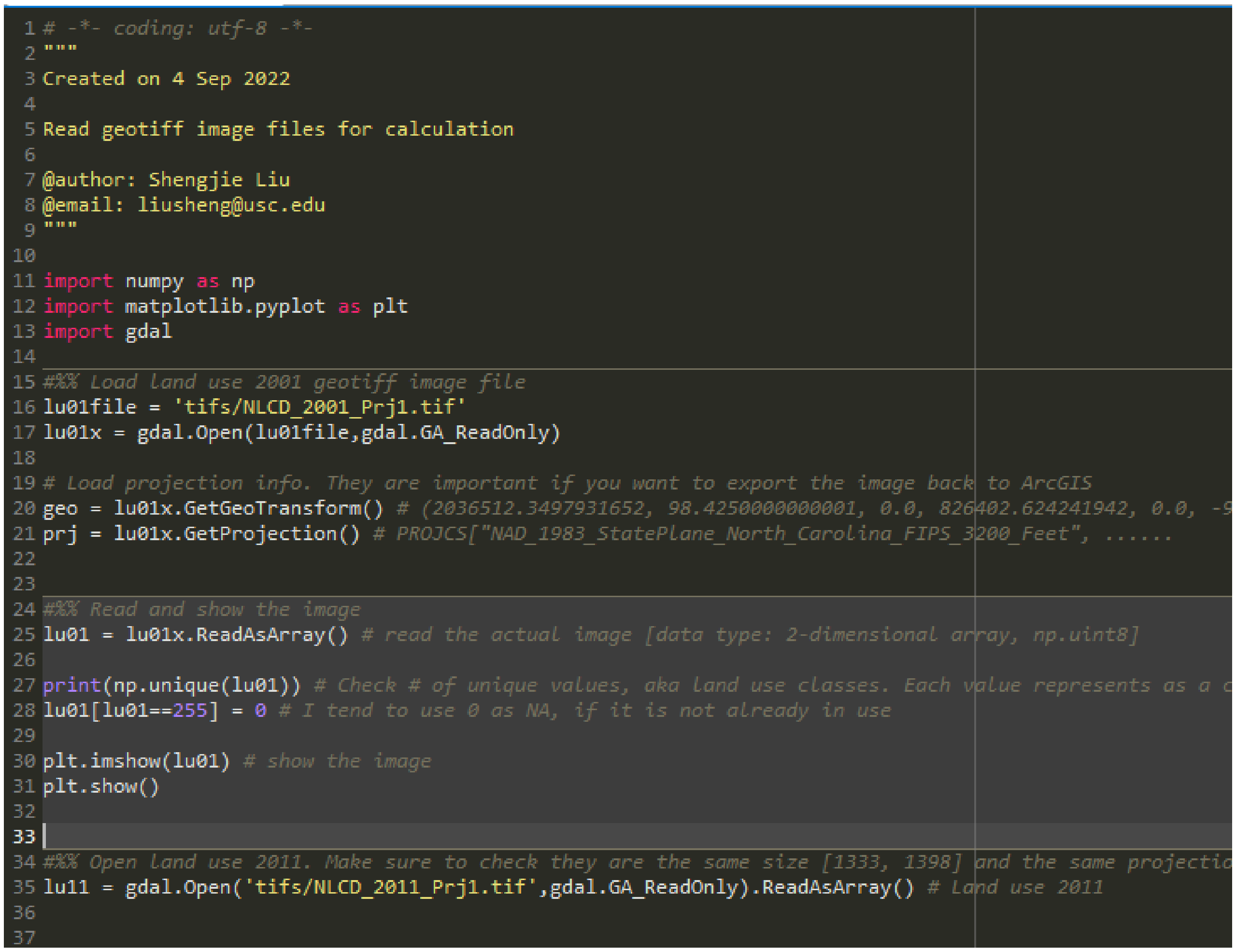


```
(base) C:\Users\skris>cd ..

(base) C:\Users>cd ..

(base) C:\>cd lab2

(base) C:\lab2>pip install GDAL-2.4.1-cp38-cp38-win_amd64.whl
Processing c:\lab2\gdal-2.4.1-cp38-cp38-win_amd64.whl
Installing collected packages: GDAL
Successfully installed GDAL-2.4.1

(base) C:\lab2>python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import gdal
>>> 
```

4. [Lines 11-14] After successfully installing GDAL, open Spyder to write Python code (Windows menu – Anaconda 3 (64-bit) – Spyder-anaconda3). You may use other IDEs or code editors such as PyCharm, Jupyter Notebook, and VS Code. We first import the packages we will use (gdal, numpy, and matplotlib.pyplot).

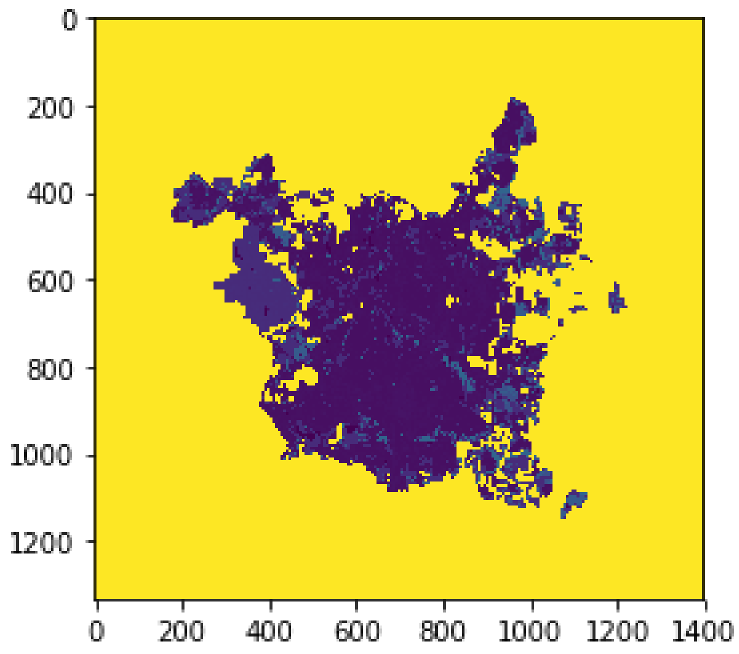
5. [Line 15-26] To read a GeoTIFF file, we need to first load the file header using “gdal.Open(imfile, gdal.GA_ReadOnly)”, which includes GeoTransform and Projection information. As a standard practice, we read the GeoTransform and Projection information. They are strings. GeoTransform (variable “geo”) includes six numbers, recording the location of the image by recording the geocoordinates of the 1st pixel (typically upper left, or northwestern) and the pixel widths in the east and southern direction. Pixel widths in east and south are normally the same, assuming a pixel is a square. In this assignment, it is 98.425 (east) and -98.425 (north). More information can be found at this website (“**https://gdal.org/tutorials/geotransforms_tut.html**”). In Projection (variable “prj”), it includes the name and detailed parameters of this projection. The Projection’s name is “NAD_1983_StatePlane_North_Carolina_FIPS_3200_Feet”. You can google this projection to find out more information. GeoTransform and Projection are necessary when you need to re-export the GeoTIFF file from Python, though we won’t re-export GeoTIFF imagery in this

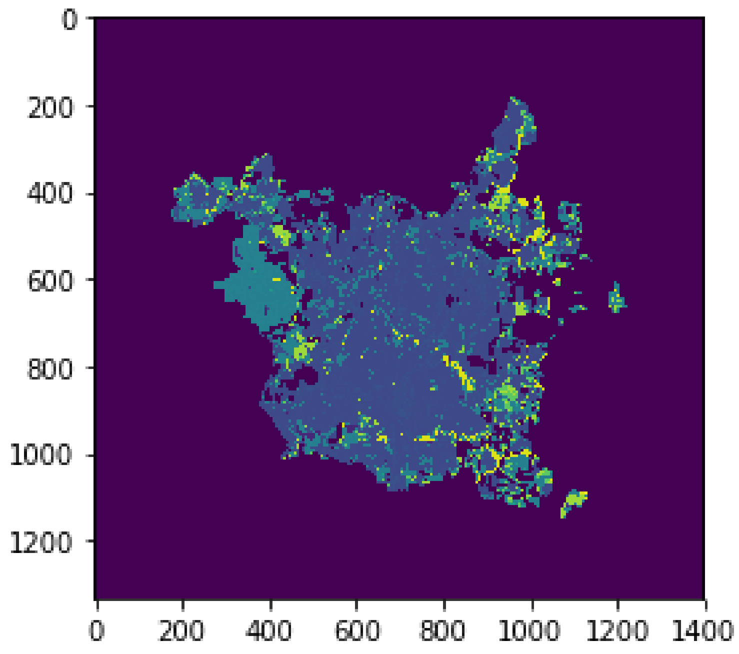
assignment. Line 25, we use "ReadAsArray" to read the image file from the header. The image is a np.unit8 array with a size of [1333, 1398].



```python
1  # -*- coding: utf-8 -*-
2  """
3  Created on 4 Sep 2022
4  
5  Read geotiff image files for calculation
6  
7  @author: Shengjie Liu
8  @email: liusheng@usc.edu
9  """
10 
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import gdal
14 
15 #%% Load land use 2001 geotiff image file
16 lu01file = 'tifs/NLCD_2001_Prj1.tif'
17 lu01x = gdal.Open(lu01file,gdal.GA_ReadOnly)
18 
19 # Load projection info. They are important if you want to export the image back to ArcGIS
20 geo = lu01x.GetGeoTransform() # (2036512.3497931652, 98.4250000000001, 0.0, 826402.624241942, 0.0, -9
21 prj = lu01x.GetProjection() # PROJCS["NAD_1983_StatePlane_North_Carolina_FIPS_3200_Feet", ......
22 
23 
24 #%% Read and show the image
25 lu01 = lu01x.ReadAsArray() # read the actual image [data type: 2-dimensional array, np.uint8]
26 
27 print(np.unique(lu01)) # Check # of unique values, aka land use classes. Each value represents as a c
28 lu01[lu01==255] = 0 # I tend to use 0 as NA, if it is not already in use
29 
30 plt.imshow(lu01) # show the image
31 plt.show()
32 
33 
34 #%% Open land use 2011. Make sure to check they are the same size [1333, 1398] and the same projectio
35 lu11 = gdal.Open('tifs/NLCD_2011_Prj1.tif',gdal.GA_ReadOnly).ReadAsArray() # Land use 2011
36 
37 
```

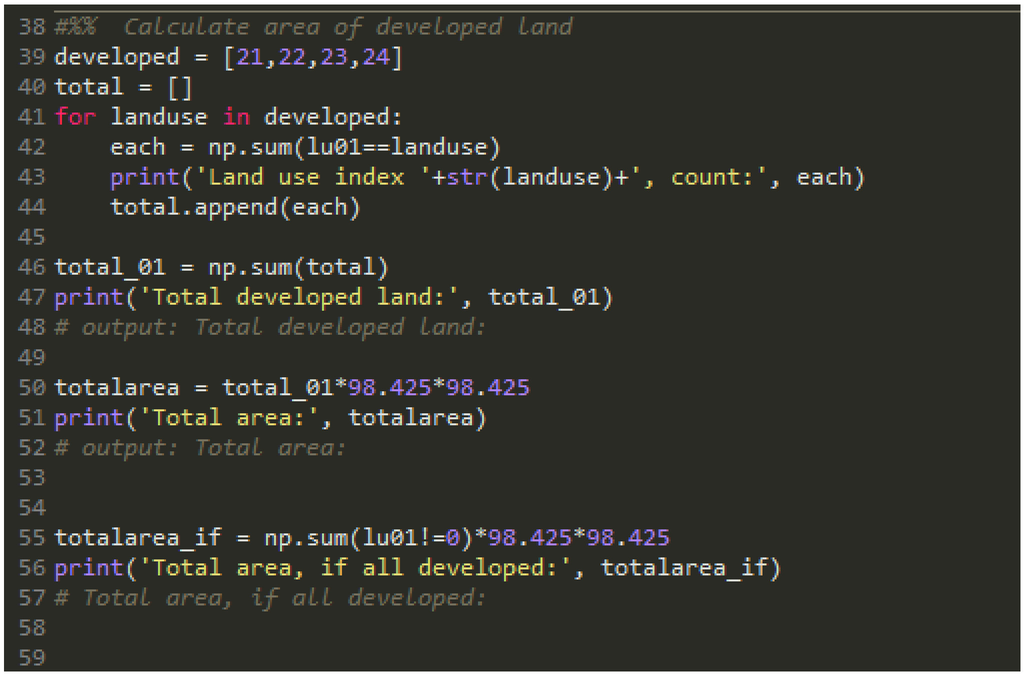
6. [Lines 27-33] By printing the unique values, we know the number of land use classes. When we exported the GeoTIFF file from ArcGIS, we set "NoData" (NA) as 255. Thus, every pixel labeled as 255 should be considered not a valid pixel. In ArcGIS, it will not display these values and show only the valid data, thus giving us a sense of that it is not an image but a raster dataset with the city's shape. Line 27 will give out "[11 21 22 23 24 31 41 42 43 52 71 81 82 90 95 255]". As a common practice, we may change the NoData to 0 when we need to show an image. If you plot the image by keeping NoData as 255, you will get the below 1st image with a yellow background. If you plot the image by changing 255 to 0, you will get the below 2nd image.





7. [Lines 34-37] We load the Land Use 2011 image for later use.
8. [Lines 38-59] We know from the legend that developed land classes are represented by 21, 22, 23, and 24. To count the number of pixels of a specific land use, for example, for land use class 21, we use "np.sum(lu01==21)". We use a loop here to count all developed land classes [21, 22, 23, 24] and use a list to save the number of pixels of each class. By summation of the values

in the list, we get the total number of pixels. To get the area, remember to multiply the actual area of each pixel.



```
#%%  Calculate area of developed land
developed = [21,22,23,24]
total = []
for landuse in developed:
    each = np.sum(lu01==landuse)
    print('Land use index '+str(landuse)+', count:', each)
    total.append(each)

total_01 = np.sum(total)
print('Total developed land:', total_01)
# output: Total developed land:

totalarea = total_01*98.425*98.425
print('Total area:', totalarea)
# output: Total area:


totalarea_if = np.sum(lu01!=0)*98.425*98.425
print('Total area, if all developed:', totalarea_if)
# Total area, if all developed:


```

9. If you can follow through here, you should be able to complete the assignment using Python. If you need to count the number of pixels that are >50 and <=100, you can use "np.sum(np.logical_and(image>50,image<=100))" which will return the value of number of pixels that are >50 and <=100. One final tip here is that please make sure when you do the calculation, you have considered NoData and handled them properly.