Philip Hess

SSCI 586

Project 5

4/29/21

**BUILDING COMPREHENSIVE TOOLS FOR SPATIAL DATA COLLECTION,**

**STORAGE, ANALYSIS, AND VISUALIZATION**

*Introduction*

The goal of this project is to increase familiarity with Python and how to using it to write shareable script tools. I am interested in agriculture and the interconnection of livestock grazing and soil health. With the idea of using coding to help automate tasks and improve efficiency, I made a custom python tool to help organize and manage rotational grazing operations.

I chose to study a rural area near Burlington Vermont, and analyze how suitable for rotational grazing the land is. Rotational grazing is a method of building soil organic matter with the help of livestock animals (Oates, 2011). A herd of large ungulates (for example cows) tramples and eats the top layers of grass, but moves on to new areas before eating the grasses all the way down. By not grazing any one area for too long, some of the grass roots slough off and add organic matter to the soil. Meanwhile, the grasses continue to grow and are ready to be grazed again sooner. Birds (for example chickens) follow a few days behind the cows and pick through the trampled grasses to find insects and larvae to eat - in the process they spread the manure left by the cows (where many larvae are maturing). The key to this whole process is rotating animals through grazing areas at the right pace so that manure can break down and grasses can re-grow before being grazed again. The tool that I made can help assess how much of that regrowth has happened, which would empower farmers and agricultural planners to design more efficient rotational grazing systems.

My tool is called pthess5Graze: It takes a polygon layer (of property boundaries) and raster imagery (ideally NAIP imagery) of the same area and divides the properties into grazing paddocks appropriately sized for rotational grazing (based on user input).It calculates the NDVI of the raster image, then summarizes each paddock area to determine how suitable it is for grazing.

*Data and Methods*

This study used NAIP imagery data collected from the USGS earth explorer website (USGS, 2021), and a census block polygon layer (the closest thing to individual property/ parcel data available in this area) from the Vermont Open Data Portal (VT, 2020). To save storage space and speed up processing time, I created a polygon layer in ArcGIS Pro to serve as a single property bounding box. All layers will be projected to match the projection of the imagery layer (NAD 83 UTM 18N in this case) (Figure 1).



Figure 1: Initial data in study area of rural Vermont.

NAIP stands for National Agricultural Imagery Program and provides multispectral aerial imagery of the continental US acquired during the growing season (FSA, 2011). NAIP has seen many improvements since its founding in 2003. NAIP imagery now records 4 bands of wavelengths (near infrared, red, green, and blue), which is particularly useful for identifying living plants. Chlorophyll in the leaves of plants reflect a lot of energy in the near infrared spectrum. False color composites using infrared imagery make it much easier to distinguish living plants from other features, such as settlements or water (Figure 2).Normalized Difference Vegetation Index (NDVI) expands on this concept but returns a normalized value between -1 and 1 (FSA, 2011). NDVI is calculated as a raster band arithmetic equation of (NIR + Red) / (NIR - Red) . These new pixel values from NDVI are more useful for comparisons and calculations. Note in figure 2 in the center bottom of each image: The body of water is much easier to distinguish from the surrounding trees in the CIR and NDVI images (center and right) compared to the natural color display (left). The distinction between different levels of vegetation is also more prominent in the NDVI image (right) than in the others.

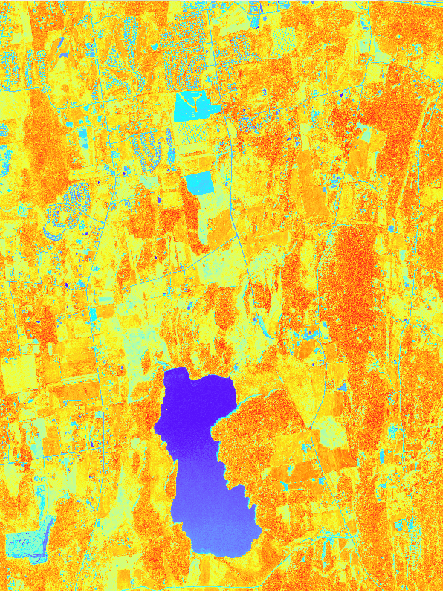
  

Figure 2: NAIP Imagery of rural Vermont with Natural Color Display (left image- Bands 1,2,3 as Red, Green, and Blue). Imagery of the same land area with Color Infrared (CIR) display (center image- Bands 4,1,2 as Red, Green, and Blue). NDVI symbolized with a red to blue palette (right image).

After reading the Zandbergen text (Zandbergen, 2020) and doing the corresponding exercises, I started writing my code from scratch using IDLE. There are many steps in my tool: First it gets the spatial reference system from the raster file, then projects the boundary layer to be in the same coordinate system. Next it clips the imagery layer so they cover the same area as the property polygon. It then uses the subdivide polygons tool to create equal area polygons (size from user input) within each property to be viewed as potential grazing paddocks. In the next step, the tool calculates NDVI using the BandArithmetic function. NDVI values are then averaged within each potential grazing paddock (Table 1). This is the final imagery layer produced but the tool goes further by creating tables identifying which paddocks have a high enough average NDVI to be grazed. I was going to ). I exported the code from model builder, added custom variables, combined it with my initial draft script and was able to run the python code successfully.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Syntax | Input | Description |
| Project | arcpy.management.Project(in\_dataset, out\_dataset, out\_coor\_system, {transform\_method}, {in\_coor\_system}, {preserve\_shape}, {max\_deviation}, {vertical}) | Polygon shapefile | Projects data to be in the same coordinate system |
| Clip Raster | arcpy.management.Clip(in\_raster, rectangle, out\_raster, {in\_template\_dataset}, {nodata\_value}, {clipping\_geometry}, {maintain\_clipping\_extent}) | Raster layer | Clips raster image to size of study area |
| Subdivide Polygon | arcpy.management.SubdividePolygon(in\_polygons, out\_feature\_class, method, {num\_areas}, {target\_area}, {target\_width}, {split\_angle}, {subdivision\_type}) | Polygon shapefile | Subdivides property bounds into equal size grazing paddocks |
| Band Arithmetic | arcpy.sa.BandArithmetic (raster, band\_ids, {method}) | Raster layer | Calculates NDVI from Red and Near Infrared bands |
| Zonal Statistics | arcpy.sa.ZonalStatistics(in\_zone\_data, zone\_field, in\_value\_raster, {statistics\_type}, {ignore\_nodata}, {process\_as\_multidimensional}, {percentile\_value}, {percentile\_interpolation\_type}) | Polygon shapefile and raster image | Calculates average NDVI value of each polygon area |

Table 1: The main Arcpy functions used to create custom grazing tool (ArcGIS, 2021).

*Results and Discussion*

My python code eventually ran well and worked as a tool, producing the expected output files. The final results for this study are displayed and symbolized in Figure 3. The resulting image shows the property boundary divided into grazing paddocks with dotted lines, and those grazing paddocks symbolized by their grazing suitability. I defined paddocks as segments of land within a property that have an area of 10,000 m2. Grazing suitability was determined by having an average NDVI value greater than 0.1. The grazing area is an input of the tool and should be determined based on how many animals you want to have on the property. For example, if 1 cow needs ~1,000 m2 per day and the user wants to graze 10 cows, they will need paddocks that are 10,000m2to rotate their herd daily. The final output of this tool adds a message to the tool history that tells how many days they can graze the land based on how many paddocks of that size have sufficient vegetation to be grazed (Figure 4, right side).

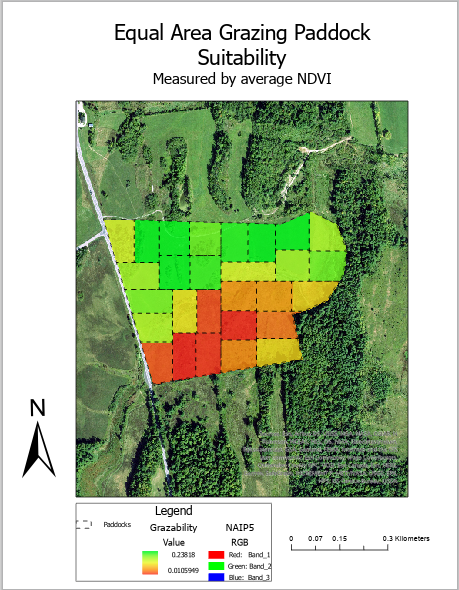


Figure 3: Results of grazing suitability too. Green and yellow paddocks have an average NDVI over 0.1 and are therefore suitable for grazing. Orange and red paddocks have average NDVI under 0.1 and are not suitable for grazing.

I had many issues along the way, but was successful in creating a comprehensive, usable tool. My biggest problem starting out was the size of my data and the limited storage space that I had on the virtual machine. NAIP files are large and I was greeting intermediary raster files of similar size, so I ran out of space quickly. To get past this, I clipped my initial data files to be about the size of just one property, and made sure to overwrite all of my outputs. This made the processing time much faster and I had enough storage space to experiment with the data.

Once I had my small dataset and had figured out the general flow of tools that I wanted to use, I tried starting in model builder. I had used this method in the past, but this time, one of my Raster band arithmetic tools would not show up or was not supported by model builder. Instead of coding that one part separately, I decided to just write the entire code from scratch. This initial problem actually ended up helping me in the long run. I got much more familiar with my code from building, testing, and running it all without the model builder interface.This way I fully understand each aspect of my code and can make more adjustments accordingly.

After numerous spelling and syntax errors, I was able to get the script working with hard coded paths. I then rewrote it as a tool and added parameters and metadata, and it ran properly. The next day it gave me error messages every time it got to arcpy.AddMessages()... despite me not changing the python at all (Figure 5). I tried troubleshooting, and the code and tool ran fine when I commented out the AddMessages(). This line of code is meant as a progress update and to provide summary statistics in an easy interface, but it should not affect how the tool actually runs. I tried again the next day and the tool worked with or without the AddMessages() so I am unsure what changed. I posted about this issue in Geonet and am awaiting a response.

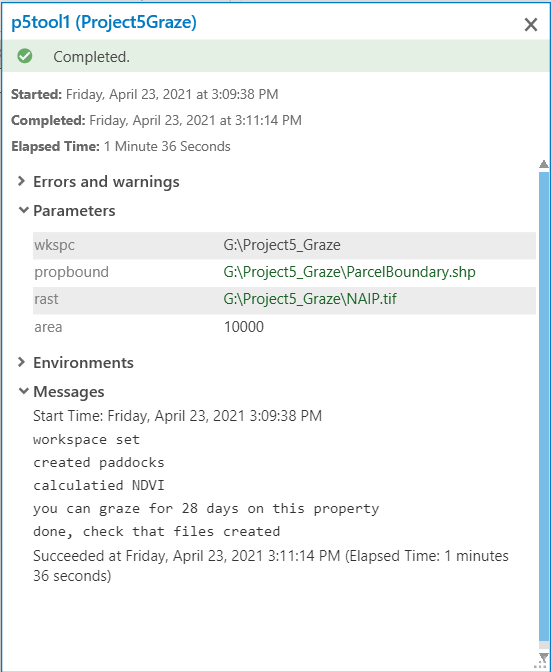
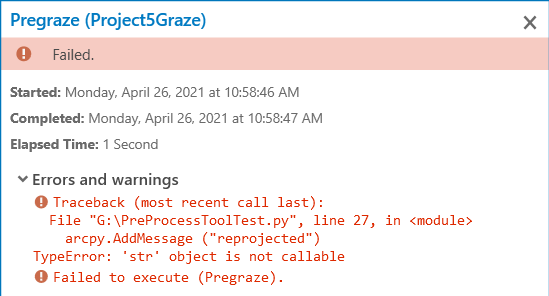


Figure 4: Spontaneous error when using arcpy.AddMessage (left). Successful run of tool, with summary message in the output (right).

This final average for each paddock is meant to represent grazability, and it does to an extent, but there is always more ancillary information that could be useful. The NAIP imagery that I used did not have an attribute table, but it did have multiple spectral bands represented. Changing this display to CIR imagery did not change the output statistics, that is why I needed to calculate NDVI first and output the statistics as a table. That being said, even NDVI is not a surefire measure of grazing suitability. Ideally other factors would be considered in this assessment, such as topography, climate, and ground cover species. An update to this tool could also include the option to draw the grazing paddocks based on the desired NDVI values (rather than summarizing the NDVI of pre-defined equal area paddocks). This tool could benefit from being used with a consistent temporal scale to analyze change in grazability throughout the season and between years.

Overall I learned a lot during this project. I got more comfortable troubleshooting python code and testing it as a tool. Typing out functions as actual code introduces a lot more potential for error compared to just running tools and models in ArcGIS Pro, but can also be much more powerful. Writing tools from scratch (without copy/pasting the syntax) and testing it every step of the way, is a much more efficient way to get comfortable with python than troubleshooting errors from macro builder. Building custom tools helps automate and speed up repetitive workflows, and it can be done with beginner level python coding experience.

*References*

ArcGIS. “ArcGIS Pro Python Reference.” ArcGIS Pro Python reference-ArcGIS Pro |

Documentation. ESRI. Accessed March 20, 2021.

https://pro.arcgis.com/en/pro-app/latest/arcpy/main/arcgis-pro-arcpy-reference.htm.

FSA, USDA. “NAIP Info Sheet,” April 2011.

<https://www.fsa.usda.gov/Internet/FSA_File/fourband_info_sheet_2011.pdf>.

Oates, Lawrence G, Daniel J Undersander, Claudio Gratton, Michael M Bell, and Randall D

Jackson. “Management‐Intensive Rotational Grazing Enhances Forage Production and

Quality of Subhumid Cool‐Season Pastures.” Crop science 51, no. 2 (2011): 892–901.

USGS. EarthExplorer. US Department of the Interior. Accessed March 20, 2021.

https://earthexplorer.usgs.gov/.

VT, GOV. “VT Open Geodata Portal.” Vermont Open Geodata Portal Your source for geospatial

data. State of Vermont, 2020. https://geodata.vermont.gov/.

Zandbergen, Paul A. “Python Scripting for ArcGIS Pro” First edition. Redlands, California:

Esri Press, 2020.

Zandbergen, Paul A. “Advanced Python Scripting for ArcGIS Pro” Redlands, California: Esri

Press, 2020.