GEOG 432/832: Programming, Scripting, and Automation for GIS

Unit 08.01: Rasters

Dr. Bitterman

Today's schedule

- Open discussion
- Reading raster data sets, interrogating the data structure, raster algebra and more
- For next class

Open discussion

How's lab 3 going?

Projects?

Raster data structure review

1. What is a raster dataset?
2. How are they used?
3. What are the key properties of a raster?
4. Any other concerns (e.g., toppolgy) that you're aware of?

Raster processing in ArcPy

- Requires the Spatial Analyst module (`arcpy.sa.*`)
- Which requires the proper license (check your textbook for how to do it programmatically)
- Your book likes to do the following:

```
import arcpy
from arcpy.sa import *

#then you can call functions directly, for example
outraster = Slope(elev)
```

I am not a fan of importing *, I prefer to be more explicit (but you do you)

```
#my way
outraster = arcpy.sa.Slope(elev)
```

Download the *week08inclass.zip* file from Canvas

- Start a new project
- Extract the data
- Create a new arcpy notebook

Let's start with listing our data

- How have we done this in the past?
- Listing rasters is pretty similar

```
wkspc_string = "C:\\Users\\pjbitterman\\Dropbox\\GEOG432\\week8_rasters\\unit08_data"
arcpy.env.workspace = wkspc_string

arcpy.ListRasters()
```

Give it a shot - what were the results?

Let's read one

How is reading rasters different than reading feature classes?

```
# read
myraster = arcpy.Raster("nlcd_lc_14n")
# display
myraster
```

What happened?

How do we learn something about our datasets?

We can "describe" our rasters, just like feature classes

For example...

before you try it, what's the output?

```
mydesc = arcpy.da.Describe(myraster)
mydesc
```

Checking raster properties

```
print(mydesc["dataType"])
print(mydesc["bandCount"])
print(mydesc["compressionType"])
print(mydesc["height"])
print(mydesc["width"])
print(mydesc["meanCellHeight"])
print(mydesc["meanCellWidth"])
print(mydesc["pixelType"])
print(mydesc["isInteger"])
print(mydesc["noDataValue"])
```

Try a few

We can also access cells directly to determine their values:

Let's break it down: what does the code do?

```
for x, y in myraster:
    if(x < 10 and y < 10):
        print(x, y, myraster[x, y])
```

What happened?

Modifying values directly

As always, let's break it down

```
myraster.save("nlcd_test")
testraster = arcpy.Raster("nlcd_test")
testraster.readOnly = False

for x, y in testraster:
    if testraster[x,y] == 41 or testraster[x,y] == 81:
        testraster[x,y] = 71

testraster.save()
```

What happened? How can you check it worked?

Using "canned" geoprocessing functions

- Let's use `ned30lc.tif` to develop a raster that describes the slope of the landscape
- But first:
  - what is `ned30lc.tif`?
  - How would you know?

Calculating slope from a DEM

```
in_dem = arcpy.Raster("ned30lc.tif")

outraster = arcpy.sa.Slope(in_dem)
outraster.save("myslope.tif")
```

What happened?

Reclassification

- What does it mean to "reclassify" a raster?
- When might you reclassify a raster?
- What does your textbook say about how we do it?
    - Values
    - Ranges
- Using our for loop from last class, how might we implement a reclassify?

Using built-in methods

Syntax:

```
Reclassify(in_raster, reclass_field, remap, {missing_values})
```

- in_raster: the raster you're working on
- reclass_field: the field you are reclassifying
- remap: an object that describes the reclassifcation
- {missing_values}: how to handle missing values

Reclassifying with values

How does this one work? Let's break it down

```
myraster = arcpy.Raster("nlcd_lc_14n")
myraster.save("nlcd_test")

testraster = arcpy.Raster("nlcd_test")
testraster.readOnly = False

myfield = "VALUE"
myremap = arcpy.sa.RemapValue([[41, 71],[81, 71]])

outraster_value = arcpy.sa.Reclassify(testraster, myfield, myremap)
```

What happened?

Reclassifying with ranges

break it down

```
myraster = arcpy.Raster("nlcd_lc_14n")
myraster.save("nlcd_test")

testraster = arcpy.Raster("nlcd_test")
testraster.readOnly = False

myfield = "VALUE"
myremap = arcpy.sa.RemapRange([[41, 51, 71], [80, 89, 71]])

outraster_range = arcpy.sa.Reclassify(testraster, myfield, myremap)
```

How could you test/display what cells changed?

A simple change detection

What's this code doing?

```
change_detect_val = myraster == outraster_value
```

Raster algebra

Arithmetic functions:

+, -, /, *

Relational functions:

==, <, >, !=

Also: Boolean and bitwise (see your book)

Assumptions of raster algebra:

- Same resolution
- Same extent
- Orthogonal

(But ArcGIS will "cheat it" for you)

You have some rasters... give it a shot. Try out a few operators

Your final task:

- Imagine a habitat suitability analysis for the will spotted hookbilled two-legged platypus (it's real, trust me 😉)
- Criteria: only lives in wetlands below 400m
- Your task: find the suitable habitat using the tools we used today

(do we want to start on the whiteboard?)

For next class

- Read Chapter 10 for this week