Anika Berger

IDCE 30274

Final Project Lab Sheet

12/2/19

**Projecting with Arcpy**

The script for this tutorial can be found in my Github repo here: <https://github.com/anberger22/Final-Project-Projections-Tutorial/blob/master/final_script.py>

**Step 1. Create a folder in your personal folder called Projections\_Tutorial**

* Create subfolders called Data, Scripts
* Inside the Data folder, create a folder for the projected layer outputs called proj

**Step 2. Download the data**

<https://docs.digital.mass.gov/dataset/massgis-data-layers>

* Libraries
* Bicycle Trails
* Farmers Markets

<https://www.naturalearthdata.com/downloads/10m-cultural-vectors/>

* Airports
* Railroads

Unzip all the folders into your Data folder (not the proj one)

**Step 3. Time to Code!**

* Open a new file in IDLE
* Name it projection\_tutorial
* Save it in your Scripts folder

import arcpy

arcpy.env.workspace = "Z:/Personal/anberger/Projections\_Tutorial/Data"

* (substitute your drive letter and your personal folder name)
* Make sure the slashes are forward (or you can put an r in front of the path, before the quotes)

**Let’s take a look at the data we’re working with**

feature\_classes = arcpy.ListFeatureClasses()

for fc in feature\_classes:

spatial\_ref = arcpy.Describe(fc).spatialReference

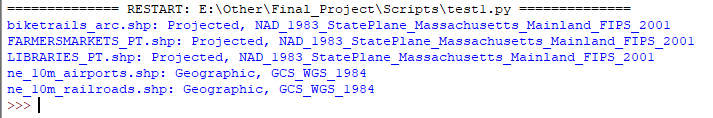
if spatial\_ref.name == "Unknown":

print("{0} has an unknown spatial reference".format(fc))

else:

print("{0}: {1}, {2}".format(fc, spatial\_ref.type, spatial\_ref.name))

The output should look something like this:



Though the above code is not necessary for what we are doing, it makes a good demonstration of the **str.format() function**.

The format() function is very helpful when you need to substitute different parts of a string. In our case, we want to concatenate different spatial reference properties (such as if it uses a Projected or Geographic coordinate system, and which one) along with the name of the feature class.

The syntax is { }.format(value)

Here is another example of how it can be used:

print ("Hello, I am {} years old !".format(18))

Now that we’ve checked the spatial reference properties of our data, we know that they are not all in the same projection. The airports and railroads layers are not even projected.

The next bit of code is in the Github repo: <https://github.com/anberger22/Final-Project-Projections-Tutorial/blob/master/final_script.py>

You are welcome to copy and paste that script. The bullet points here explain step-by-step what is happening, but the same information is commented on each line in the code.

**Step 4. Get the projections of all the data to match**

* Choose a shapefile that is projected in the way you want
* This will be our main file that we will match the other layers to
  + LIBRARIES\_PT.shp is chosen for this tutorial, but this can be substituted for any shapefile
* Create a variable to easily extract spatial reference information from main shapefile

libraries = "LIBRARIES\_PT.shp"

library\_spatial\_ref = arcpy.Describe(libraries).spatialReference

* Check to see what projection the main file is using

print library\_spatial\_ref.name

* Check which factory code corresponds to this projection
  + In times where inconsistent names cause problems, use numbers instead

print library\_spatial\_ref.factoryCode

* Assign a variable to main file's projection to use when projecting the other feature classes

out\_coordinate\_system = arcpy.SpatialReference(library\_spatial\_ref.factoryCode)

* Create a list of the feature classes that are in the Data folder
  + If you kept the first bit of code we did, you don’t need to do this again

feature\_classes = arcpy.ListFeatureClasses()

* Start a for loop that runs through each feature class in the data folder
  + Create a variable to easily extract spatial reference info from any of the feature classes
  + Exclude the main shapefile libraries from the for loop
    - If the projection of the feature class is the same as that of the main file
      * Print this statement
    - If not (else:)
      * Print the spatial reference that it does have
      * Create a variable for feature class output name that can be used in the path by replacing the .shp with nothing
      * Send outputs to the proj folder by adding the file path /proj/ then add \_proj to the end of each newly projected feature class
      * Print to make sure it correctly changed the name
      * Project the feature classes to the coordinate system of the main shapefile and assign them new names in a different folder

for fc in feature\_classes:

spatial\_ref = arcpy.Describe(fc).spatialReference

if fc != libraries:

if spatial\_ref.name == library\_spatial\_ref.name:

print fc, "has the same projection as Libraries"

else:

print spatial\_ref.name

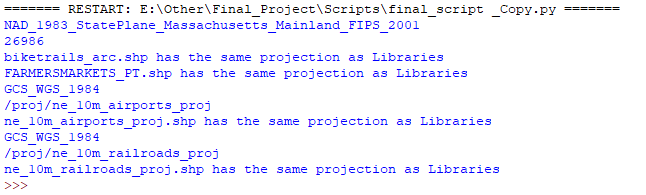
file\_name\_new = fc.replace(".shp", "")

file\_name\_new = '/proj/'+file\_name\_new+'\_proj'

print file\_name\_new

arcpy.Project\_management(fc,file\_name\_new, out\_coordinate\_system)

The output should look like this



**Congratulations**! You’ve successfully created a script to get different data layers’ projections to match!