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

Unit 10.01: Spatial data and intro to viz

Dr. Bitterman

Today's schedule

- Open discussion
- Project update presentation(s)
- Slides, discussion and exercises
- For next class

Open discussion

Today's prep:

- We'll use *unit10inclass.zip* from Canvas
- Open Anaconda
- Wait

Packages/modules we'll need today

- geopandas
- matplotlib

Reading spatial data

What are some possible forms (and sources) of spatial data?

Package setup

```
%matplotlib inline
import geopandas
import matplotlib
```

Reading spatial data

A zip file from the web (you may want to copy-paste the URL)

```
# reading from the web
# this is a zip file hosted on my Dropbox account. The zip contains a .shp file
muni_url = "https://www.dropbox.com/s/lhojjephcr3ky54/muni_boundaries.zip?dl=1"
muni_boundaries = geopandas.read_file(muni_url)
```

```
muni_boundaries.plot()
muni_boundaries.crs
```

What happened?

Reading from geojson

Wait, what's a geojson?

What's a "json"?

Open street_centerlines_lc.geojson in a text editor and see for yourself

Reading a geojson

```
# Read from geojson
streets_path = "./unit10data/street_centerlines_lc.geojson"

streets = geopandas.read_file(streets_path)
```

display

```
streets.plot() # may take a while
streets.crs
```

Be careful with geojson, files get HUGE

And shapefiles too

```
# Read a bog standard shapefile
schools_path = "./unit10data/Public_Schools.shp"
schools = geopandas.read_file(schools_path)
schools
```

display

```
schools.plot()
```

Anything notable about how we read these 3 file types into memory?

geopandas dataframes are a LOT like non-spatial dataframes

ESDA is ALWAYS a good idea... what's "ESDA"?

- all the operations we've used before still work:
- try some:
  - head()
  - tail()
  - describe()
  - max() (again, might not make sense for some data)

Slicing a spatial dataframe

- just like aspatial dataframes, we can look at a subset
- let's try a few:

```
streets.loc[2500, 'geometry'] # you don't always get a good look the geometry, depending on scale
```

```
muni_boundaries.loc[577, 'geometry']
```

What happened?

Some basic styling

Setting an alpha value

```
schools.plot(alpha = 0.1)
```

What happened?

Super simple mapping

```
# Setup figure and axis
f, ax = matplotlib.pyplot.subplots(1)

# Plot layer of polygons on the axis
muni_boundaries.plot(ax = ax)

# Remove axis frames
ax.set_axis_off()

# Add figure title
f.suptitle("Municipalities in Nebraska")

# Display
matplotlib.pyplot.show()
```

What did we just do?

1. We have first created a figure named f with one axis named ax by using the command `matplotlib.pyplot.subplots` *(Note the method is returning two elements and we can assign each of them to objects with different name (f and ax) by simply listing them at the front of the line, separated by commas)*
2. We plot the geographies, but tell the function that we want it to draw the polygons on the axis we are passing, *ax*. This method returns the axis with the geographies in them, so we make sure to store it on an object with the same name, ax.
3. We remove the box with coordinates
4. We set a title
5. diplayed the figure by calling `matplotlib.pyplot.show()`

A quick multilayer example

- We can do some simple multilayer mapping by adding layers one at a time to a figure
- For example:

```
lc_path = "./unit10data/lancaster_county.shp"
lc = geopandas.read_file(lc_path)

# Setup figure and axis
f, ax = matplotlib.pyplot.subplots(1)

# Add a layer with polygon on to axis `ax`
lc.plot(ax = ax, color = "green")

# Add a layer with lines on top in axis `ax`
streets.plot(ax = ax, color = "yellow")

# give it a title
f.suptitle("What a horrible color scheme")
```

We can also do some basic calculations...

Calculate area

```
muni_areas = muni_boundaries.area
muni_areas.head()
```

what happened? Does it make sense?

Always project your data!!!

What does this code do?

```
munis_14n = muni_boundaries.to_crs(epsg=26914) # EPSG for NAD84 UTM 14N
muni_areas = munis_14n.area
muni_areas.head()
```

What are the units now?

Lengths, too:

```
street_length = streets.to_crs(epsg=26914).length
street_length.head()
```

Garbage in, garbage out (know your datasets)... this works:

```
streets.to_crs(epsg=26914).area.head()
```

What's wrong with it?

And buffers are straightforward:

Break it down:

```
schools14n = schools.to_crs(munis_14n.crs) # set to the CRS of an existing layer
schools14n.crs

school_buff = schools14n.buffer(500) # 500m buffer
school_buff.head()
```

```
school_buff.plot() # at this scale, maybe a bit tough to tell they're buffers
```

For next class

- Lab 5 due April 1st
- Lab 6 starts Friday
- Readings are linked/posted on Canvas
- HOMEWORK: review https://darribas.org/gds_course/content/bC/lab_C.html (the framework for today's slides)