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

## Week 11.01: Spatial data and intro to viz

Dr. Bitterman

# Today's schedule

- Open discussion
- Slides, discussion and exercises
- For next class

# Open discussion

# Today's prep:

- We'll use `week11inclass.zip` from Canvas
- Launch Anaconda
- Wait

# Packages/modules we'll need today

- `geopandas`
- `matplotlib`
- `mapclassify`
- `pysal`
- `libpysal`

# Reading spatial data

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

# Package setup

```
%matplotlib inline
import geopandas
import matplotlib
import mapclassify
import pysal
import libpysal
```

# 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 the GitHub page. The zip contains a .shp file
county_url = "https://github.com/pjbitterman/UNL_geog432/blob/main/data/county_boundaries.zip?raw=true"
county_boundaries = geopandas.read_file(county_url)
```

```
county_boundaries.plot()
county_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 = "./week11data/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 = "./week11data/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
```

```
county_boundaries.loc[91, '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
county_boundaries.plot(ax = ax)

# Remove axis frames
ax.set_axis_off()

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

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

# What did we just do?

1. 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 assigned each of them to objects with different name (f and ax) by listing them at the front of the statement)*
2. Plotted the geographies, telling the function that to draw the polygons on the axis we passed, *ax*. This method returns the axis with the geographies in them, so we stored it on an object with the same name, ax.
3. Removed the box with coordinates
4. Set a title
5. Displayed 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 = "./week11data/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")

# save it to disk
#matplotlib.pyplot.savefig('lc_streets.png')
```

# We can also do some basic spatial calculations...

### Calculate area

```
county_areas = county_boundaries.area
county_areas.head()
```

### what happened? Does it make sense?

# Always project your data!!!

## What does this code do?

```
counties_14n = county_boundaries.to_crs(epsg=26914) # EPSG for NAD84 UTM 14N
county_areas = counties_14n.area
county_areas.head()

# if you get a ProjError, update proj and pyproj packages in your env.
# then restart the jupyter server
```

## 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 and data types)... 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(counties_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
```

# Paired programming exercise

# Something a bit different

- We're going to use datasets built into the libpysal package
- I'm going to share with you an .ipynb file
- Why? Because I want to replicate the experience of how we "feel around" to accomplish a new task

# Your goal: make some choropleth maps

- What's a choropleth map?
- What are its characteristics?
- What are some limitations of choropleth mapping?

# What does choropleth mapping require we do to our data?

# Classification methods

## What are they?

TO THE WHITEBOARD!!!!

# Work!

## Then, change it up

- Choose a different variable
- Choose a different color scheme (where to look?)
- Play around... ESDA

# For next class

- Update presentations THURSDAY!
- Lab 5 starts Thursday
- Readings are linked/posted on Canvas
- HOMEWORK: review https://darribas.org/gds_course/content/bC/lab_C.html (the framework for today's slides)