# assignment_eight

March 25, 2023


### 0.1 Timothy Miller

### 0.2 GTECH 73100, Dr. Sun


## 1 Assignment Eight

States and counties in geopandas

#### 1.0.1 Import modules

[ ]:
```
import pandas as pd
import geopandas as gpd
import json
import io
```

#### 1.0.2 Translate encoding of json file

The original "data/gz_2010_us_050_00_20m.json" file appears to be encoded in 'latin-1', causing errors when attempting read it directly into geopandas This is resolved by first reading in the file as json and then reexporting as json, changing the name to geojson

[ ]:
```
with io.open("data/gz_2010_us_050_00_20m.json", encoding="latin-1") as f:
    data = json.load(f)

with open("data/gz_2010_us_050_00_20m.geojson", "w") as fp:
    json.dump(data, fp)
```

#### 1.0.3 Read files

[ ]:
```
county_data = gpd.read_file("data/gz_2010_us_050_00_20m.geojson")

with io.open("data/fipsToState.json") as f:
    fips_to_state = json.load(f)

with io.open("data/stateCodeToFips.json") as f:
    state_code_to_fips = json.load(f)
```

[ ]:
```
fips_to_state_df = pd.DataFrame(
    {
```

```
        "STATE": fips_to_state.keys(),
        "STATE_NAME": fips_to_state.values(),
    }
)
```

### 1.0.4 Task 1

Find the top n most common county names

[ ]:
```
def get_most_common_county_names(n):
    return county_data.groupby("NAME").size().nlargest(n)
```

[ ]:
```
print(get_most_common_county_names(5))
```

```
NAME
Washington    31
Franklin      26
Jefferson     26
Jackson       24
Lincoln       24
dtype: int64
```

### 1.0.5 Task 3

Join fips code to get fill names of states using merge
(Doing before task two, in order to format task two with state names)

[ ]:
```
state_county_data = county_data.merge(fips_to_state_df)
state_county_data.head()
```

[ ]:

| | GEO_ID | STATE | COUNTY | NAME | LSAD | CENSUSAREA | \ |
|---|---|---|---|---|---|---|---|
| 0 | 0500000US01001 | 01 | 001 | Autauga | County | 594.436 | |
| 1 | 0500000US01009 | 01 | 009 | Blount | County | 644.776 | |
| 2 | 0500000US01017 | 01 | 017 | Chambers | County | 596.531 | |
| 3 | 0500000US01021 | 01 | 021 | Chilton | County | 692.854 | |
| 4 | 0500000US01033 | 01 | 033 | Colbert | County | 592.619 | |

| | geometry | STATE_NAME |
|---|---|---|
| 0 | POLYGON ((-86.49677 32.34444, -86.71790 32.402… | Alabama |
| 1 | POLYGON ((-86.57780 33.76532, -86.75914 33.840… | Alabama |
| 2 | POLYGON ((-85.18413 32.87053, -85.12342 32.772… | Alabama |
| 3 | POLYGON ((-86.51734 33.02057, -86.51596 32.929… | Alabama |
| 4 | POLYGON ((-88.13999 34.58170, -88.13925 34.587… | Alabama |

### 1.0.6 Task 2

Summary statistics for states - number of counties - min and max area of counties within state

[ ]:
```
state_groups = state_county_data.groupby("STATE_NAME")
```

**Task 2, Part A** Number of counties (output limited to five largest counts)

```
[ ]: state_groups["NAME"].size().nlargest(5)
```

```
[ ]: STATE_NAME
     Texas       254
     Geogia      159
     Virginia    134
     Kentucky    120
     Missouri    115
     Name: NAME, dtype: int64
```

**Task 2, Part B** Area of largest county (output limited to states with the top 5 largest counties)

Method 1:
Sort the data on census area as is, and then drop all states less than the max state (Advantage is that it provides all other fields)

```
[ ]: state_county_data.sort_values("CENSUSAREA", ascending=False).drop_duplicates(
         ["STATE_NAME"]
     )[:5]
```

```
[ ]:              GEO_ID STATE COUNTY            NAME    LSAD  CENSUSAREA  \
     94    0500000US02290    02    290   Yukon-Koyukuk      CA  145504.789
     220   0500000US06071    06    071  San Bernardino  County   20056.938
     529   0500000US04005    04    005        Coconino  County   18618.885
     1750  0500000US32023    32    023             Nye  County   18181.924
     3047  0500000US56037    56    037      Sweetwater  County   10426.649

                                                  geometry  STATE_NAME
     94    POLYGON ((-153.00134 62.72744, -153.00126 62.2…      Alaska
     220   POLYGON ((-115.64803 35.80963, -115.64768 35.8…  California
     529   POLYGON ((-112.53859 37.00067, -112.53454 37.0…     Arizona
     1750  POLYGON ((-115.84580 36.12024, -115.84608 35.9…      Nevada
     3047  POLYGON ((-110.04800 41.57802, -110.05371 42.2…     Wyoming
```

Method 2:
Group the states together and then find the max CENSUSAREA (Advantage is that it seems more 'pythonic')

```
[ ]: state_groups["CENSUSAREA"].max().nlargest(5).reset_index()
```

```
[ ]:    STATE_NAME  CENSUSAREA
     0      Alaska  145504.789
     1  California   20056.938
     2     Arizona   18618.885
     3      Nevada   18181.924
     4     Wyoming   10426.649
```

(Getting the index instead of the value could allow for looking of the full entry later)

[ ]:
```
state_groups["CENSUSAREA"].idxmax().head()
```

[ ]:
```
STATE_NAME
Alabama        39
Alaska         94
Arizona       529
Arkansas      154
California    220
Name: CENSUSAREA, dtype: int64
```

**Task 2, Part C** Area of smallest county (output limited to 5 smallest)
Using Method 1, as it preserves the most data

[ ]:
```
state_county_data.sort_values("CENSUSAREA").drop_duplicates(["STATE_NAME"])[:5]
```

[ ]:

| | GEO_ID | STATE | COUNTY | NAME | LSAD | CENSUSAREA | \ |
|---|---|---|---|---|---|---|---|
| 2879 | 0500000US51610 | 51 | 610 | Falls Church | city | 1.999 | |
| 545 | 0500000US15005 | 15 | 005 | Kalawao | County | 11.991 | |
| 2184 | 0500000US36061 | 36 | 061 | New York | County | 22.829 | |
| 3138 | 0500000US44001 | 44 | 001 | Bristol | County | 24.164 | |
| 231 | 0500000US08014 | 08 | 014 | Broomfield | County | 33.034 | |

| | geometry | STATE_NAME |
|---|---|---|
| 2879 | POLYGON ((-77.15029 38.87619, -77.15497 38.872… | Virginia |
| 545 | POLYGON ((-157.01455 21.18550, -156.99911 21.1… | Hawaii |
| 2184 | MULTIPOLYGON (((-74.04086 40.70012, -74.04002 … | New York |
| 3138 | POLYGON ((-71.22480 41.71050, -71.22787 41.705… | Rhode Island |
| 231 | POLYGON ((-105.14734 39.91389, -105.14734 39.9… | Colorado |

### 1.0.7 Task 4

Map the top five counties with the most common names

[ ]:
```
top_counties = (
    state_county_data.groupby("NAME").size().nlargest(5).reset_index()["NAME"]
)
```

[ ]:
```
base_map = None
colors = ["#8dd3c7", "#ffffb3", "#bebada", "#fb8072", "#80b1d3"]
for idx, county in enumerate(top_counties):
    matches = state_county_data.loc[state_county_data["NAME"] == county]
    series = gpd.GeoSeries(matches["geometry"])
    if base_map == None:
        base_map = series.explore(style_kwds={"color": colors[idx]})
    else:
        base_map = series.explore(m=base_map, style_kwds={"color": colors[idx]})
```

```
base_map
```

[ ]: <folium.folium.Map at 0x7fb92acb2250>