Mapping with GeoPandas and U.S. Census Data

Chetan Tiwari

February 20, 2025

Introduction

This tutorial introduces mapping and geospatial analysis using Python’s `geopandas` library. We will retrieve county-level population data for Georgia from the U.S. Census Bureau and visualize it as a choropleth map.

Part 1: Setting Up the Environment

Prerequisites:

- Python installed on your system.
- Installed libraries: `geopandas`, `pandas`, `requests`, `matplotlib`.

To install the necessary libraries, run:

```
pip install geopandas pandas requests matplotlib
```

Alternatively, you can use the class TLJH server at `https://nas.enterprisegis.net:8888`.

Part 2: Downloading U.S. Census Data

We will retrieve population estimates for Georgia counties using the U.S. Census API. To obtain an API key:

1. Visit the Census Bureau’s API page.
2. Click on ”Request a KEY” and register. Note: this is a banner link on the left side of the page!
3. Copy the API key provided in your email.

Fetching County-Level Population Data

1. Construct the API call using the examples provided here. Click on the section that says *American Community Survey (ACS) Example*.
2. Extract population data using the 5-year ACS estimate for 2023. You can find the variable associated with total population (or other variables) here. You can find additional examples here. Hint: see the examples under *State* → *County*. Note the state code for GA is 13.
3. You can always test your URL request by using a browser. If you generated your request correctly, you should see a listing of the data you requested.

```
import requests
import pandas as pd

# Replace with your Census API key
api_key = ’YOUR_API_KEY’
```

```
# Construct the URL using the instructions and examples provided above
url = ’’ # insert your API request here

# Fetch the data
response = requests.get(url)
data = response.json()
```

1. Print data to see what the structure looks like. You’ll notice that it takes on a form that looks like [[a,b,c,d],[1,2,3,4],[5,6,7,8]]
2. This looks like a list that contains a collection of other lists. The first element of the list provides the column names and the remaining elements provide the data values.
3. You can convert this to a Pandas dataframe using the logic outlined below.

```
# Store the column headers by referencing element 0 - its the first element of the list.
columns = data[0]
# specify the data as all elements 1 (inclusive) onwards
rows = data[1:]
# Convert the rows and columns to a dataframe using the Dataframe function as shown below
df = pd.DataFrame(rows, columns=columns)
```

1. Now that you have a dataframe with population data for GA counties, check the columns and associated datatypes. You can do that by using the info() function.
2. You will need to convert the columns from object to string and numeric as needed. See example below.

```
# Create Geoid by concatenating state and county
df["geoid"] = df["state"].astype(str) + df["county"].astype(str)

# Create a new col called totalpopulation that
# uses the field B01001_001E and converts it to numeric
df[’totalpopulation’] = pd.to_numeric(df[’B01001_001E’])
df.head()
```

What is a Choropleth Map?
A choropleth map is a thematic map where geographic areas are shaded based on a data variable. In this case, counties will be colored based on their population.

Part 3: Working with GeoPandas

Download Georgia’s County Shapefile

The U.S. Census Bureau provides shapefiles for U.S. counties. You can directly download and load the county boundary shapefile into a geopandas dataframe as shon below.

```
# Download and directly store the county shapefile for GA from the census
# We will create a geopandas dataframe called gdf
gdf = gpd.read_file(’https://www2.census.gov/geo/tiger/
␣␣␣␣TIGER2023/COUNTY/tl_2023_us_county.zip’)
gdf.head()
```

Extract GA counties by filtering the geopandas data frame to only include those rows where STATEFP == 13. Please look at the Pandas cheat sheet to figure out how to extract rows where STATEFP = 13. Please note that column names are case sensitive.

Step 2: Load the Shapefile and Merge with Population Data

```
import geopandas as gpd

# Load all US counties shapefile
gdf = gpd.read_file("path_to_extracted_shapefile/tl_2022_13_county.shp")

# Extract data for GA
gdf_ga = gdf[gdf['STATEFP'] == '13']
gdf_ga.head()

# Perform a left join from df to gdf_ga using GEOID and geoid as the common column.
# Note the case.
df_merged = df.merge(gdf_ga, left_on="geoid", right_on="GEOID", how="left")

# Check for missing values
print(df_merged.isnull().sum())
```

Part 4: Creating the Choropleth Map

The code listing below provides some sample code needed to produce a choropleth map using geopandas and matplotlib. We will explore easier ways to do this in a later class. There are several parameters that can be tweaked. Please see comments in the code for details on how some adjustments can be made.

```
import matplotlib.pyplot as plt
import geopandas as gpd
import matplotlib.patches as mpatches
from matplotlib_scalebar.scalebar import ScaleBar
import pandas as pd

# Ensure df_merged is a GeoDataFrame and reproject to a common CRS
df_merged = gpd.GeoDataFrame(df_merged, geometry="geometry")

# Reproject to Conic projection (ESRI:102004) - a meter-based projection
# This is necessary to ensure accurate distance measurements for the scale bar
df_merged = df_merged.to_crs('ESRI:102004')

# ------------------------------
# CLASSIFY DATA INTO QUANTILES (5 BINS)
# ------------------------------
# Quantile-based classification ensures each bin contains approximately
# the same number of counties
df_merged["pop_class"], bin_edges = pd.qcut(df_merged["totalpopulation"],
    q=5, retbins=True, labels=False)

# ------------------------------
# DEFINE COLOR MAP FOR QUANTILE CATEGORIES
# ------------------------------
# Assigning distinct colors for each quantile category (light to dark)
# Colors range from light yellow to dark red (can be modified for different color schemes)
colors = ["#ffffb2", "#fecc5c", "#fd8d3c", "#e31a1c", "#800026"]
cmap_dict = dict(zip(range(5), colors))  # Mapping categories to colors

# ------------------------------
# CREATE PLOT
# ------------------------------
fig, ax = plt.subplots(figsize=(10, 10))  # Define figure size

# Plot the map with discrete quantile colors
df_merged.plot(column="pop_class", cmap="OrRd", linewidth=0.8, edgecolor="0.8",
    ax=ax, legend=False)

# ------------------------------
# ADD A DISCRETE LEGEND WITH POPULATION RANGES
# ------------------------------
# Creating legend labels using the actual numeric population range values
labels = [f"{int(bin_edges[i]):,}␣-␣{int(bin_edges[i+1]):,}"
    for i in range(len(bin_edges) - 1)]

# Creating discrete patches for the legend
patches = [mpatches.Patch(color=cmap_dict[i], label=labels[i]) for i in range(5)]

# Add a legend to the plot (Upper right corner)
ax.legend(handles=patches, title="Population␣Range", loc="upper␣right")

# ------------------------------
# ADD TITLE
# ------------------------------
ax.set_title(’Georgia␣County␣Population’, fontdict={’fontsize’: 15, ’fontweight’: 3})

# Remove axes for a cleaner map display
ax.set_axis_off()

# ------------------------------
# ADD A SCALE BAR
# ------------------------------
# The scale bar is automatically sized based on the CRS projection (meters)
# units=’m’ - Displays distances in meters. Change to ‘units=’km’‘ for kilometers
# length_fraction=0.5 - Scale bar spans 50% of the map width
# pad=0.01 - Adjusts distance between the map and the scale bar
scalebar = ScaleBar(1, units=’m’, pad=0.01, length_fraction=0.5,
    location=’lower␣right’, scale_loc="bottom")
ax.add_artist(scalebar)

# ------------------------------
# ADD A NORTH ARROW (SIMPLE METHOD)
# ------------------------------
# This method uses ‘ax.annotate()‘ to place a simple "N" with an arrow
# Note: There are **better ways** to do this using ‘folium‘, etc.
# If using an interactive map, consider using ‘folium.plugins.MarkerCluster‘
arrow_x, arrow_y = 0.95, 0.1  # Adjust position
ax.annotate(’N’, xy=(arrow_x, arrow_y + 0.05), xytext=(arrow_x, arrow_y),
            xycoords=’axes␣fraction’, textcoords=’axes␣fraction’,
            fontsize=15, ha=’center’, va=’center’,
            arrowprops=dict(facecolor=’black’, edgecolor=’black’, width=3, headwidth=8))

# ------------------------------
# SAVE & DISPLAY FIGURE
# ------------------------------
# Save the figure at high resolution (300 dpi) with tight bounding box
plt.savefig('georgia_county_population.png', dpi=300, bbox_inches='tight')

# Display the figure
plt.show()
```

What to Turn In?

Submit the PNG file produced.