Problem Set 5

Francesca Leon and Claudia Felipe

2024-11-05

Due 11/9 at 5:00PM Central. Worth 100 points + 10 points extra credit.

Submission Steps (10 pts)

1. This problem set is a paired problem set.
2. Play paper, scissors, rock to determine who goes first. Call that person *Partner 1*.
    - Partner 1 (name and cnet ID): Francesca leon (francescaleon)
    - Partner 2 (name and cnet ID): Claudia Felipe (claudiafelipe)
3. Partner 1 will accept the `ps5` and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted.
4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **FL CF**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point)
6. Late coins used this pset: **0** Late coins left after submission: **4**
7. Knit your `ps5.qmd` to an PDF file to make `ps5.pdf`,
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
8. (Partner 1): push `ps5.qmd` and `ps5.pdf` to your github repo.
9. (Partner 1): submit `ps5.pdf` via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope

```
import pandas as pd
import altair as alt
import time
from bs4 import BeautifulSoup
import requests
from tabulate import tabulate
from datetime import datetime
import geopandas as gpd
import matplotlib.pyplot as plt
import os

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")

base_directory = '/Users/francescaleon/Documents/GitHub/DAPII/ps5_fran_clau/'
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
# Get the HTML
soup = BeautifulSoup(requests.get("https://oig.hhs.gov/fraud/enforcement/").\
  content, "html.parser")

# Find and store titles
titles = soup.find_all("h2", class_="usa-card__heading")
title_list = []
for title in titles:
    title = title.find("a").text.strip()
    title_list.append(title)

# Find and store links
links = soup.find_all("h2", class_="usa-card__heading")
link_list = []
for link in links:
    link = link.find("a").get("href")
    link_list.append("https://oig.hhs.gov" + link)

# Find and store dates
```

```
dates = soup.find_all("span", class_="text-base-dark")
date_list = []
for date in dates:
    date = date.text.strip()
    date_list.append(date)

# Find and store categories
categories = soup.find_all("ul", class_="display-inline")
category_list = []
for category in categories:
    category = category.find("li").text.strip()
    category_list.append(category)

# Merge values
values = pd.DataFrame({
    "Title": title_list,
    "Link": link_list,
    "Date": date_list,
    "Category": category_list
})

# Print head
print(values.head())
```

```
                                               Title  \
0  Macomb County Doctor And Pharmacist Agree To P...
1  Rocky Hill Pharmacy And Its Owners Indicted Fo...
2  North Texas Medical Center Pays $14.2 Million ...
3  New England Doctor Pleads Guilty To Drug Distr...
4  St. Louis County Woman Accused Of $3 Million H...

                                               Link              Date  \
0  https://oig.hhs.gov/fraud/enforcement/macomb-c...  November 4, 2024
1  https://oig.hhs.gov/fraud/enforcement/rocky-hi...  November 4, 2024
2  https://oig.hhs.gov/fraud/enforcement/north-te...  November 4, 2024
3  https://oig.hhs.gov/fraud/enforcement/new-engl...  November 4, 2024
4  https://oig.hhs.gov/fraud/enforcement/st-louis...  November 1, 2024

                    Category
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
```

```
3  Criminal and Civil Actions
4  Criminal and Civil Actions
```

2. Crawling (PARTNER 1)

```python
# List to store agencies
agencies = []

# Loop through each link
for link in values["Link"]:
        response = requests.get(link)
        soup_agency = BeautifulSoup(response.content, "html.parser")
        agency = soup_agency.find("span", text="Agency:")
        agencies.append(agency.next_sibling.strip())

# Add column to the dataframe
values["Agency"] = agencies

# Print head
print(values.head())
```

```
                                               Title  \
0  Macomb County Doctor And Pharmacist Agree To P...
1  Rocky Hill Pharmacy And Its Owners Indicted Fo...
2  North Texas Medical Center Pays $14.2 Million ...
3  New England Doctor Pleads Guilty To Drug Distr...
4  St. Louis County Woman Accused Of $3 Million H...

                                               Link              Date  \
0  https://oig.hhs.gov/fraud/enforcement/macomb-c...  November 4, 2024
1  https://oig.hhs.gov/fraud/enforcement/rocky-hi...  November 4, 2024
2  https://oig.hhs.gov/fraud/enforcement/north-te...  November 4, 2024
3  https://oig.hhs.gov/fraud/enforcement/new-engl...  November 4, 2024
4  https://oig.hhs.gov/fraud/enforcement/st-louis...  November 1, 2024

                    Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions
```

```
                                                 Agency
0  U.S. Attorney's Office, Eastern District of Mi...
1  U.S. Attorney's Office, Eastern District of Te...
2  U.S. Attorney's Office, Northern District of T...
3                          U.S. Department of Justice
4  U.S. Attorney's Office, Eastern District of Mi...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

Step 1: Inspect the HTML of the first enforcement actions page we want to scrape

Step 2: Check if the year is 2013 or later. If not, ask the user to enter a valid year.

Step 3: Initialize empty lists for storing titles, links, dates, and categories.

Step 4: Parse HTML from first page and start web scraping as in Step 1, using a "while" loop to go through each page to find all tags.

Step 5: Use a for loop to extract title, link, date, and category -Identify the date of each enforcement and make a conditional (if the date is earlier than the user-specified start date, break the loop.) -Crawl into each enforcement action link to get agency information -Then, append this information to the respective lists.

Step 6: Create a DataFrame from the lists.

Step 7: Save the final DataFrame to a CSV file.

- Function used We use a while loop because this loop is used to execute a block of code repeatedly as long as a given condition is true. When the condition becomes false, the loop terminates and the next statement after the loop is executed. In our case, we want that the loop iterates (extract title, link, date, category and agency) until the date is earlier that the user-specified start date.

- b. Create Dynamic Scraper (PARTNER 2)

```
def dynamic_scraper(month, year):
    # Check if the year is 2013 or later
    if year < 2013:
        print("Please enter a year of 2013 or later.")
        return None
```

```
    start_date = datetime(year, month, 1)

    # Empty lists to store data
    title_list, link_list, date_list, category_list, agency_list = [], [],
↪   [], [], []

    # Begin looping through pages for web scraping
    page = 1
    stop_scraping = False

    while not stop_scraping:
        url = f"https://oig.hhs.gov/fraud/enforcement/?page={page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.content, "html.parser")

        titles = soup.find_all("h2", class_="usa-card__heading")
        dates = soup.find_all("span", class_="text-base-dark")
        categories = soup.find_all("ul", class_="display-inline")

        for i in range(len(titles)):
            title_text = titles[i].find("a").text.strip()
            link = titles[i].find("a").get("href")
            full_link = "https://oig.hhs.gov" + link

            date_text = dates[i].text.strip()
            date_object = datetime.strptime(date_text, "%B %d, %Y")

            # If the date is older than the specified start date, break
            if date_object < start_date:
                stop_scraping = True
                break

            category_text = categories[i].find("li").text.strip()

            # Crawl into each enforcement action link to get agency
             ↪  information
            action_response = requests.get(full_link)
            action_soup = BeautifulSoup(action_response.content,
↪   "html.parser")
            agency = action_soup.find("span", text="Agency:")
            agency_text = agency.next_sibling.strip() if agency else "N/A"
```

```
            # Append data to lists
            title_list.append(title_text)
            link_list.append(full_link)
            date_list.append(date_object)
            category_list.append(category_text)
            agency_list.append(agency_text)

        time.sleep(1)
        page += 1

    # Create a DataFrame from the collected lists
    data = pd.DataFrame({
        "Title": title_list,
        "Link": link_list,
        "Date": date_list,
        "Category": category_list,
        "Agency": agency_list
    })

    # Save the DataFrame
    filename = f"enforcement_actions_{year}_{month}.csv"
    data.to_csv(filename, index=False)
    print(f"Data saved to {filename}")

    return data

# For January 2023
enforcement_actions_2023 = dynamic_scraper(1, 2023)

# Count the number of enforcement actions in the DataFrame
num_actions_2023 = enforcement_actions_2023.shape[0]
print(f"Number of enforcement actions scraped: {num_actions_2023}")

# Earliest enforcement action
enforcement_actions_2023_sorted =
 ↪  enforcement_actions_2023.sort_values(by="Date", ascending=True)
earliest_action_2023 = enforcement_actions_2023_sorted.iloc[0]
print("\nEarliest enforcement action scraped:")
print(earliest_action_2023)
```

```
Data saved to enforcement_actions_2023_1.csv
Number of enforcement actions scraped: 1507
```

```
Earliest enforcement action scraped:
Title       Podiatrist Pays $90,000 To Settle False Billin...
Link        https://oig.hhs.gov/fraud/enforcement/podiatri...
Date                                    2023-01-03 00:00:00
Category                            Criminal and Civil Actions
Agency      U.S. Attorney's Office, Southern District of T...
Name: 1506, dtype: object
```

- c. Test Partner's Code (PARTNER 1)

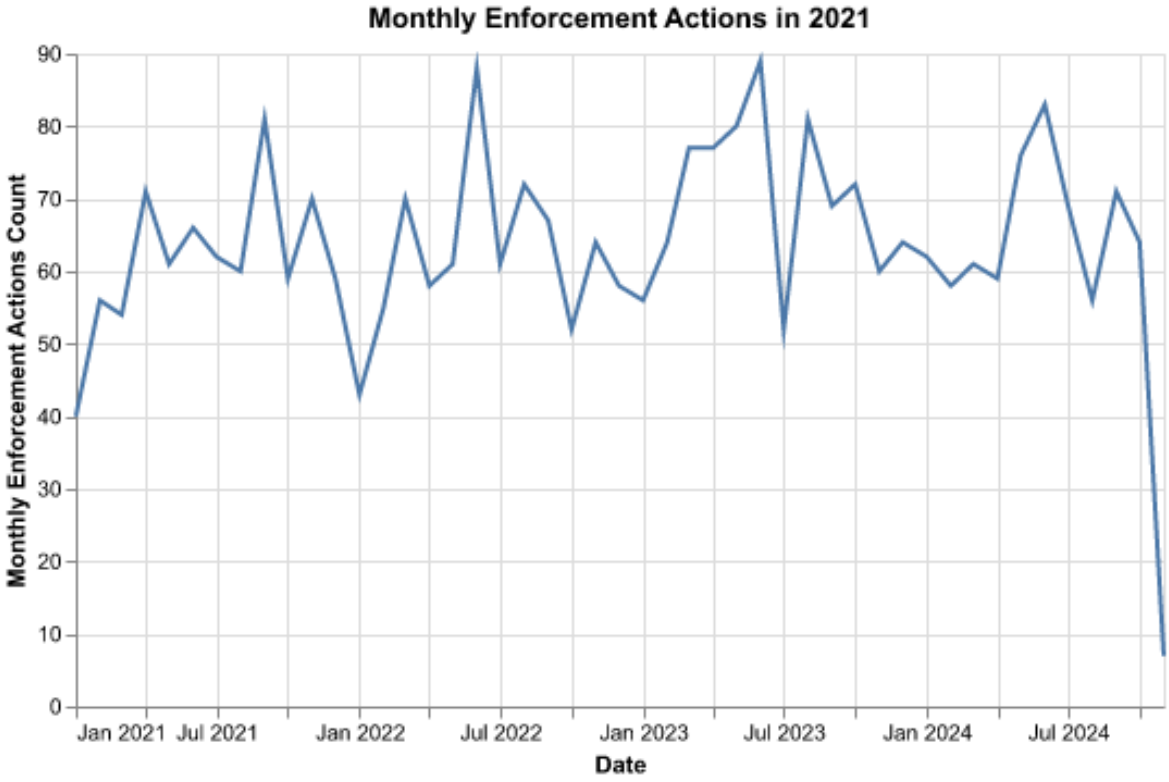
```python
# From January 2021
enforcement_actions_2021 = dynamic_scraper(1, 2021)

# Count the number of enforcement actions in the DataFrame
num_actions_2021 = enforcement_actions_2021.shape[0]
print(f"Number of enforcement actions scraped: {num_actions_2021}")

# Earliest enforcement action
enforcement_actions_2021_sorted =
 ↪  enforcement_actions_2021.sort_values(by="Date", ascending=True)
earliest_action_2021 = enforcement_actions_2021_sorted.iloc[0]
print("\nEarliest enforcement action scraped since 2021:")
print(earliest_action_2021)
```

```
Data saved to enforcement_actions_2021_1.csv
Number of enforcement actions scraped: 2995

Earliest enforcement action scraped since 2021:
Title       The United States And Tennessee Resolve Claims...
Link        https://oig.hhs.gov/fraud/enforcement/the-unit...
Date                                    2021-01-04 00:00:00
Category                            Criminal and Civil Actions
Agency      U.S. Attorney's Office, Middle District of Ten...
Name: 2994, dtype: object
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time (PARTNER 2)

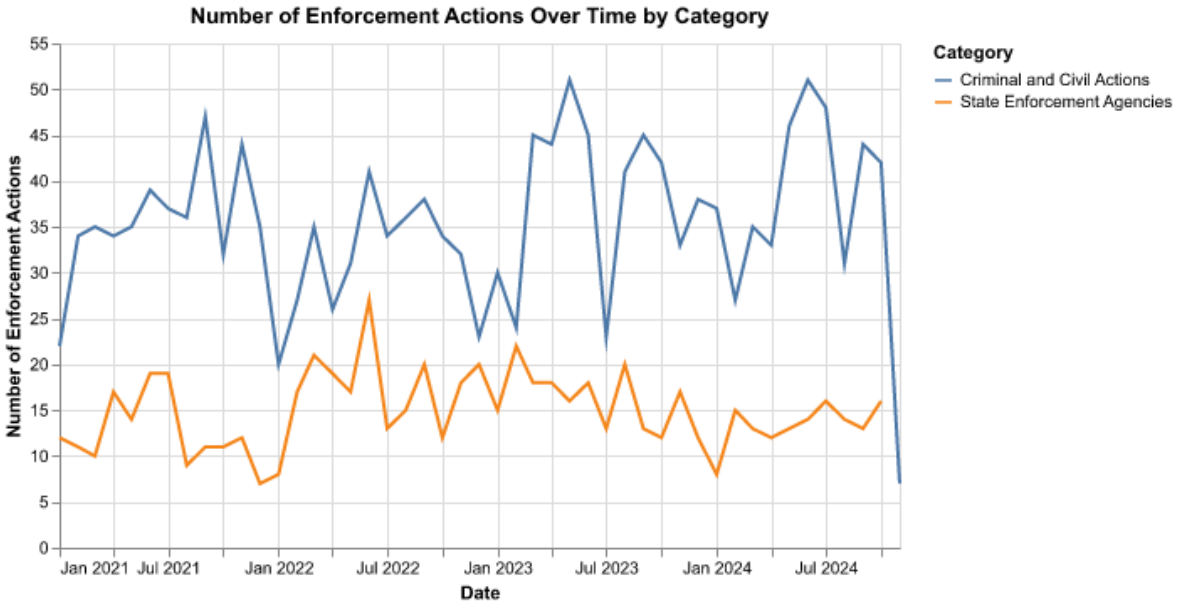
```
# Create the line chart
line = alt.Chart(enforcement_actions_2021).mark_line().encode(
    x=alt.X('yearmonth(Date):T', title='Date'),
    y=alt.Y('count():Q', title='Monthly Enforcement Actions Count')
).properties(
    width=500,
    height=300,
    title="Monthly Enforcement Actions in 2021"
)

line
```



2. Plot the number of enforcement actions categorized: (PARTNER 1)

- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"

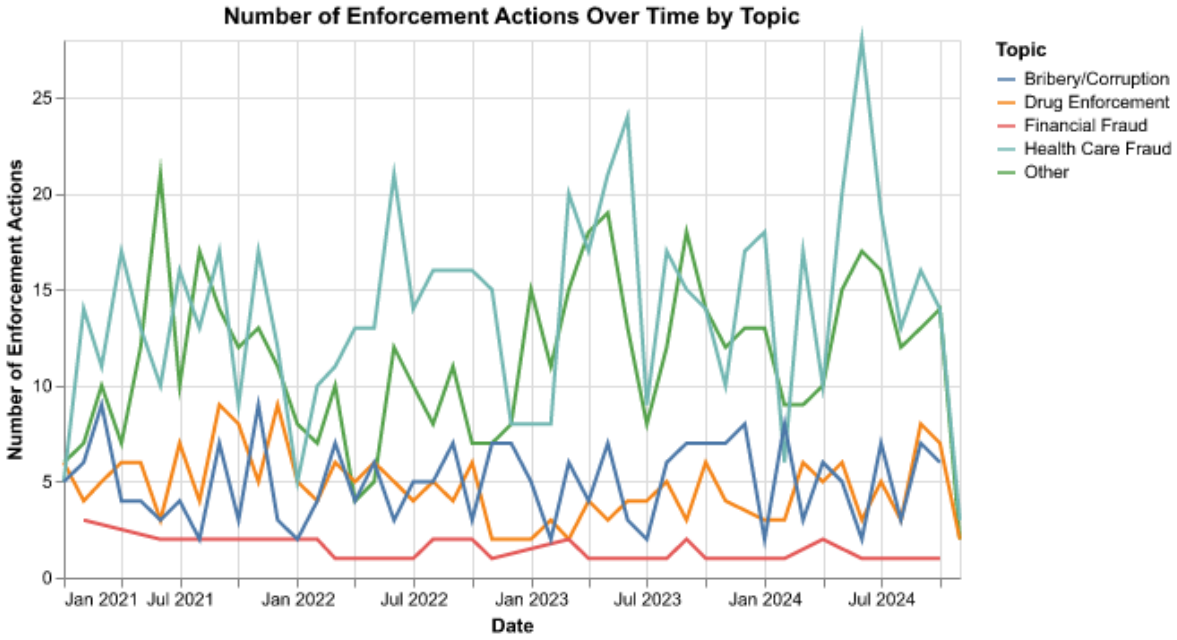
```
# Filter categories of interest
graph_categories =
 ↪  enforcement_actions_2021[enforcement_actions_2021['Category'].isin(['Criminal
 ↪  and Civil Actions', 'State Enforcement Agencies'])]

# Graph using category
alt.Chart(graph_categories, title = 'Number of Enforcement Actions Over Time
 ↪  by Category').mark_line().encode(
  alt.X('yearmonth(Date):T').title('Date'),
  alt.Y('count():Q').title('Number of Enforcement Actions'),
  color=alt.Color('Category:N', title='Category')
).properties(width = 500)
```



- based on five topics

```
# Filter category of interest
criminal_and_civil =
 ↪  enforcement_actions_2021[enforcement_actions_2021['Category'] ==
 ↪  'Criminal and Civil Actions']

# Assign topics based on words in the title
criminal_and_civil["Topic"] = "Other"
criminal_and_civil.loc[criminal_and_civil["Title"].str.lower().str.contains("health|medicare
 ↪  "Topic"] = "Health Care Fraud"
```

```
criminal_and_civil.loc[criminal_and_civil["Title"].str.lower().str.contains("bank|financial|
 ↪  laundering|investment"), "Topic"] = "Financial Fraud"
criminal_and_civil.loc[criminal_and_civil["Title"].str.lower().str.contains("drug|narcotics|t
 ↪  "Topic"] = "Drug Enforcement"
criminal_and_civil.loc[criminal_and_civil["Title"].str.lower().str.contains("bribery|corrupti
 ↪  "Topic"] = "Bribery/Corruption"

# Graph using topics
alt.Chart(criminal_and_civil, title = 'Number of Enforcement Actions Over
 ↪  Time by Topic').mark_line().encode(
  alt.X('yearmonth(Date):T').title('Date'),
  alt.Y('count():Q').title('Number of Enforcement Actions'),
  color=alt.Color('Topic:N', title='Topic')
).properties(width = 500)
```

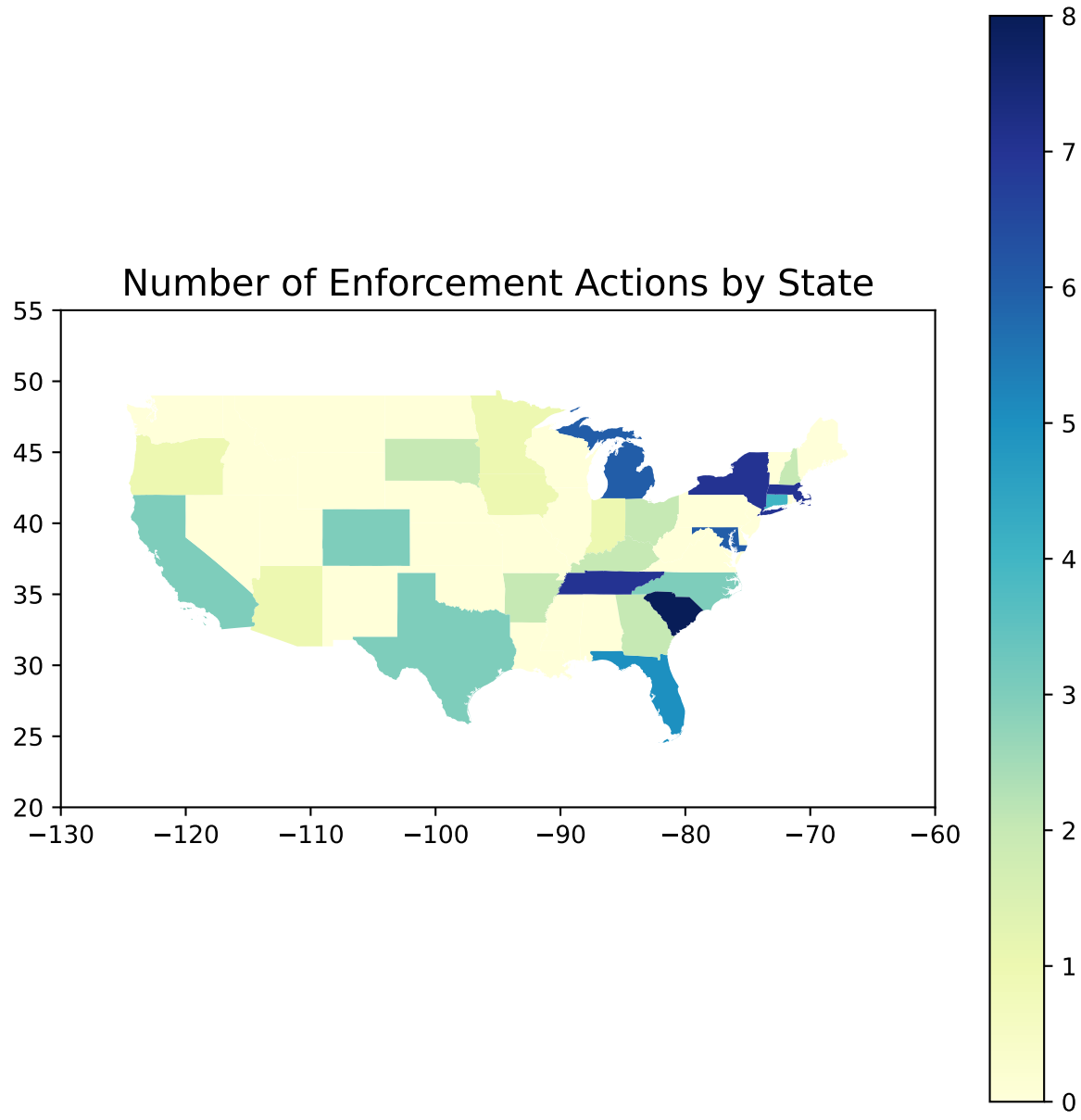


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
# Import shapefile
states_shp = gpd.read_file(os.path.join(base_directory,
 ↪  'cb_2018_us_state_500k/cb_2018_us_state_500k.shp'))
```

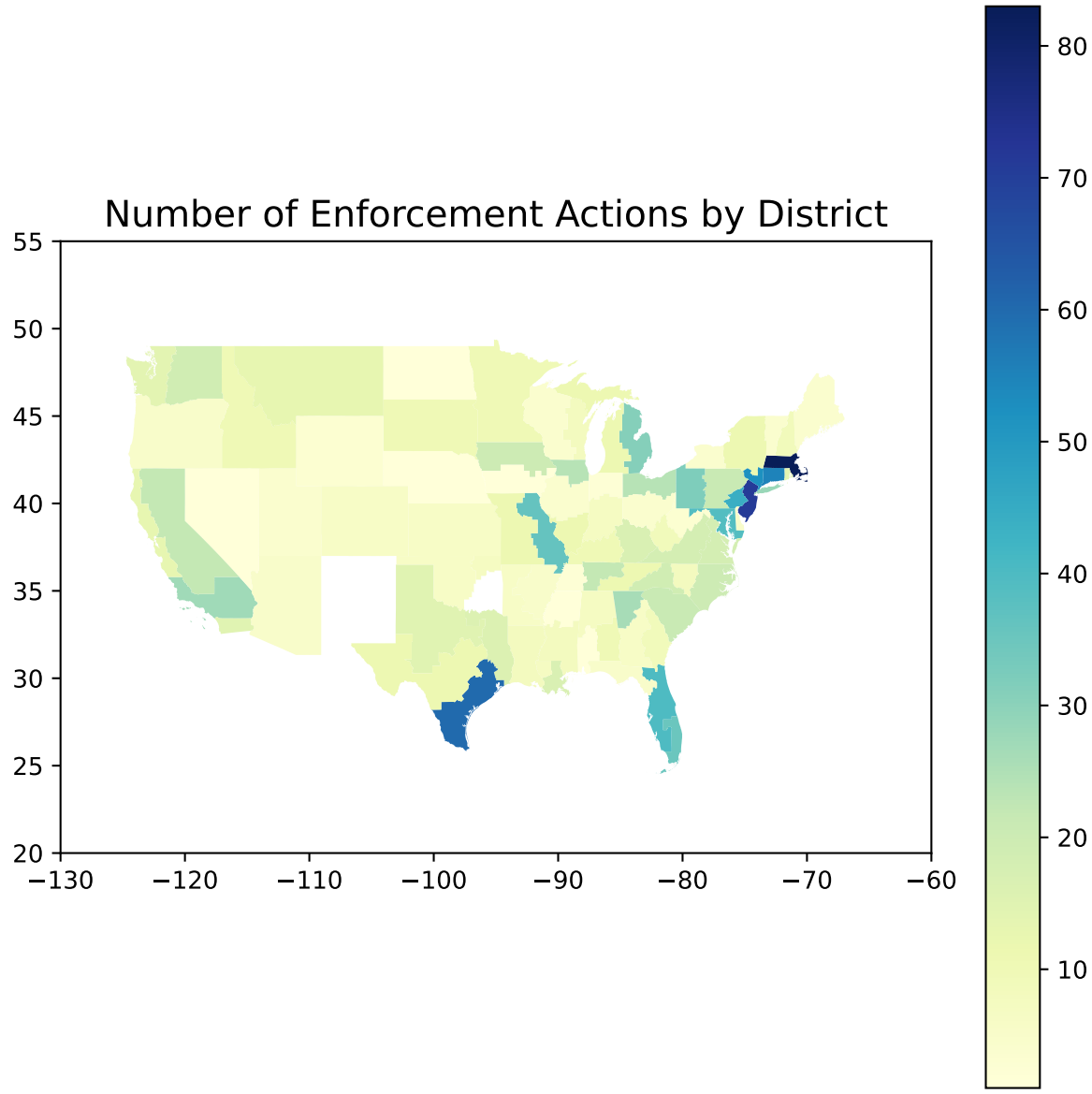
```
# Filter enforcement actions taken by state-level agencies
states_enf_actions =
 ↪  enforcement_actions_2021[enforcement_actions_2021['Agency'].str.contains("State
 ↪  of")]

# Keep only state names
states_enf_actions["State"] =
 ↪  states_enf_actions["Agency"].str.extract(r"State of (.+)", expand=False)

# Merge shapefile with enforcement actions
states_merge = states_shp.merge(states_enf_actions, left_on="NAME",
 ↪  right_on="State", how="left")

# Count values by state
states_merge = gpd.GeoDataFrame(states_merge[['NAME', 'geometry',
 ↪  'Agency']].groupby(['NAME',
 ↪  'geometry']).count().reset_index().rename(columns={'Agency':
 ↪  'enforcement_actions'}), geometry='geometry')


#Plot choropleth
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
states_merge.plot(column='enforcement_actions', cmap='YlGnBu', legend=True,
 ↪  ax=ax)
ax.set_xlim([-130, -60])
ax.set_ylim([20, 55])
ax.set_title("Number of Enforcement Actions by State", fontsize=15)
plt.show()
```



2. Map by District (PARTNER 2)

```
# Load the shapefile
attorney_districts = gpd.read_file(os.path.join(base_directory, 'US Attorney
 ↪  Districts Shapefile
 ↪  simplified_20241105/geo_export_ae39349c-4874-41e8-809c-af605a0ac6fa.shp'))
```

```
# Filter enforcement actions taken by state-level agencies
states_enf_actions =
 ↪  enforcement_actions_2021[enforcement_actions_2021['Agency'].str.contains("District")]

# Function to extract text after the comma
def extract_district_after_comma(text):
    parts = text.split(",")
    if len(parts) > 1:
        return parts[1].strip()
    return "N/A"

# Apply function to extract districts
enforcement_actions_2021['District'] =
 ↪  enforcement_actions_2021['Agency'].apply(extract_district_after_comma)

# Merge shapefile with enforcement actions
district_merge = pd.merge(enforcement_actions_2021, attorney_districts,
 ↪  left_on=['District'], right_on=['judicial_d'], how='left')

# Count values by District
district_merge = gpd.GeoDataFrame(district_merge[['District', 'geometry',
 ↪  'Agency']].groupby(['District',
 ↪  'geometry']).count().reset_index().rename(columns={'Agency':
 ↪  'enforcement_actions'}), geometry='geometry')

# Plot the choropleth map
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
district_merge.plot(column='enforcement_actions', cmap='YlGnBu', legend=True,
 ↪  ax=ax)
ax.set_xlim([-130, -60])
ax.set_ylim([20, 55])
ax.set_title("Number of Enforcement Actions by District", fontsize=15)
plt.show()
```



Extra Credit

1. Merge zip code shapefile with population

```
# Import shapefile previous problem set
zips_shp =
 ↪  gpd.read_file('/Users/francescaleon/Documents/GitHub/DAPII/problem-set-4-fran-clau/gz_20
```

```
# Import and clean census data
census = pd.read_csv(os.path.join(base_directory,
 ↪  'DECENNIALDHC2020.P1_2024-11-05T214024/DECENNIALDHC2020.P1-Data.csv'))
census["NAME"] = census["NAME"].str.replace("ZCTA5 ", "", regex=False)
census = census[['GEO_ID', 'NAME',
 ↪  'P1_001N']].drop(index=0).reset_index(drop=True).rename(columns={'P1_001N':
 ↪  'POPULATION', 'NAME': 'ZCTA5'})

# Merge shapefile with census data
zips_census = zips_shp.merge(census, on="ZCTA5", how="left")
```

2. Conduct spatial join

```
# Spatial join between zips population and districts
zips_district = gpd.sjoin(district_merge, zips_census, how="inner",
 ↪  predicate="intersects")

# Ensure all values are numeric
zips_district['enforcement_actions'] =
 ↪  pd.to_numeric(zips_district['enforcement_actions'], errors='coerce')
zips_district['POPULATION'] = pd.to_numeric(zips_district['POPULATION'],
 ↪  errors='coerce')

# Aggregate variables
zips_district_agg = gpd.GeoDataFrame(zips_district[['District', 'geometry',
 ↪  'enforcement_actions', 'POPULATION']].groupby(['District',
 ↪  'geometry']).agg({'enforcement_actions': 'mean', 'POPULATION':
 ↪  'sum'}).reset_index())
```

3. Map the action ratio in each district

```
# Calculate ratio of enforcement actions
zips_district_agg['ratio_enforcement'] =
 ↪  (zips_district_agg['enforcement_actions']/zips_district_agg['POPULATION'])*100

# Plot the choropleth map
```

```
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
zips_district_agg.plot(column='ratio_enforcement', cmap='YlGnBu',
 ↪  legend=True, ax=ax)
ax.set_xlim([-130, -60])
ax.set_ylim([20, 55])
ax.set_title("Ratio of Enforcement Actions by District", fontsize=15)
plt.show()
```