Problem Set 5

Xinyi Zhou & Wuzhen Han

2024-11-09

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): Xinyi Zhou (xzhou66)
    - Partner 2 (name and cnet ID): Wuzhen Han (hanwuzhen)
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: xyz& wzh **___** **___**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" xyz&wzh (1 point)
6. Late coins used this pset: xyz&wzh:used 1**___** Late coins left after submission: both left with 2**___**
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

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

from datetime import datetime

import requests
from bs4 import BeautifulSoup
import pandas as pd

import concurrent.futures
from datetime import datetime

import geopandas as gpd
import matplotlib.pyplot as plt
import re
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

titles = []
dates = []
links = []
categories = []

enforcement_items = soup.find_all('li', class_='usa-card card--list
 ↪  pep-card--minimal mobile:grid-col-12')

for item in enforcement_items:
    title_tag = item.find('h2', class_='usa-card__heading')
```

```
    title = title_tag.get_text(strip=True)
    titles.append(title)

    date_tag = item.find('span', class_='text-base-dark padding-right-105')
    date = date_tag.get_text(strip=True)
    dates.append(date)

    category_tag = item.find('li', class_='display-inline-block usa-tag
↪   text-no-lowercase text-base-darkest bg-base-lightest margin-right-1')
    category = category_tag.get_text(strip=True)
    categories.append(category)

    link_tag = title_tag.find('a', href=True)
    link = link_tag['href'] if link_tag else 'N/A'
    if not link.startswith('http'):
        full_link = f'https://oig.hhs.gov{link}'
    else:
        full_link = link
    links.append(full_link)

df = pd.DataFrame({
    'Title': titles,
    'Date': dates,
    'Category': categories,
    'Link': links
})

print(df.head())
```

```
                                               Title              Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  November 8, 2024
1  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
2  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
3  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
4  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 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
```

```
                                                Link
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...
```

2. Crawling (PARTNER 1)

```
agencies = []

for full_link in links:
    action_response = requests.get(full_link)
    action_response.raise_for_status()
    action_soup = BeautifulSoup(action_response.text, 'html.parser')

    agency_tag = action_soup.find('span', string='Agency:')
    if agency_tag:
        agency = agency_tag.find_parent('li').get_text(
            strip=True).replace('Agency:', '').strip()
    else:
        agency = 'N/A'

    agencies.append(agency)

    time.sleep(1)

df = pd.DataFrame({
    'Title': titles,
    'Date': dates,
    'Category': categories,
    'Link': links,
    'Agency': agencies
})

print(df.head())
```

```
                                               Title              Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  November 8, 2024
1  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
```

```
2  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
3  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
4  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 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

                                                 Link  \
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...

                                               Agency
0                          U.S. Department of Justice
1  November 7, 2024; U.S. Attorney's Office, Dist...
2  U.S. Attorney's Office, District of Massachusetts
3  U.S. Attorney's Office, Eastern District of Vi...
4  U.S. Attorney's Office, Middle District of Flo...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

Ans:

1. Input Validation: Check if the provided `start_year` is greater than or equal to 2013. If not, display an error message and terminate the function.
2. Initialization: Initialize empty lists to store scraped data (e.g., titles, dates, links, and categories). Define the base URL of the website and set `page_number` to start from the first page.
3. Date Handling: Retrieve the current date and convert the input start month and year into a `datetime` object for filtering purposes.

4. Loop for Scraping: Use a `while` loop to process pages:
    - Construct the URL for the current page.
    - Send an HTTP GET request and check the response status.
    - Parse the HTML content using BeautifulSoup.
    - Locate all enforcement items on the page.
5. Data Extraction: For each enforcement item:
    - Extract the title, date, category, agency, and link.
    - Skip entries where the date is earlier than the specified start date.
    - Append valid data to the corresponding lists.
6. Breaking Conditions: If no enforcement items are found on the page, or if all data on the page is invalid, terminate the scraping process.
7. Add Delay: Add a delay between requests to prevent server overload or blocking.
8. Data Saving: After the loop, create a DataFrame with the collected data and save it as a CSV file. The filename should include the `start_year` and `start_month`.
9. Completion Message:
   Print a message indicating that the scraping is complete, showing the total number of records saved.

- b. Create Dynamic Scraper (PARTNER 2)

```
def scrape_enforcement_actions(start_month, start_year, end_page=480):
    if start_year < 2013:
        print("Year must be >= 2013. Please provide a valid year.")
        return

    titles, dates, links, categories, agencies = [], [], [], [], []

    base_url = 'https://oig.hhs.gov/fraud/enforcement/'
    page_number = 1

    while page_number <= end_page:
        url = f"{base_url}?page={page_number}"
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')

        enforcement_items = soup.find_all('li', class_='usa-card card--list
↪  pep-card--minimal mobile:grid-col-12')
```

```
        if not enforcement_items:
            break

        valid_data_found = False

        for item in enforcement_items:
            try:
                title = item.find('h2',
↪   class_='usa-card__heading').get_text(strip=True)
                date = item.find('span', class_='text-base-dark
↪   padding-right-105').get_text(strip=True)
                date_obj = datetime.strptime(date, "%B %d, %Y")

                # Stop if the date is earlier than start date
                if date_obj < datetime(start_year, start_month, 1):
                    continue

                valid_data_found = True
                category = item.find('ul', class_='display-inline
↪   add-list-reset').get_text(strip=True) if item.find('ul',
↪   class_='display-inline add-list-reset') else 'N/A'
                link = item.find('a', href=True)['href']
                full_link = f'https://oig.hhs.gov{link}'
            except AttributeError:
                continue

            titles.append(title)
            dates.append(date)
            categories.append(category)
            links.append(full_link)

        time.sleep(1)

        if not valid_data_found:
            break

        page_number += 1

    def get_agency_info(link):
        try:
            action_response = requests.get(link)
```

```
            action_response.raise_for_status()
            action_soup = BeautifulSoup(action_response.text, 'html.parser')

            agency_tag = action_soup.find('span', string='Agency:')
            agency =
↪  agency_tag.find_parent('li').get_text(strip=True).replace('Agency:',
↪  '').strip() if agency_tag else 'N/A'
        except (requests.exceptions.RequestException, AttributeError):
            agency = 'N/A'
        return agency

    if links:
        with concurrent.futures.ThreadPoolExecutor(max_workers=10) as
         ↪  executor:
            agencies = list(executor.map(get_agency_info, links))

        df = pd.DataFrame({
            'Title': titles,
            'Date': dates,
            'Category': categories,
            'Link': links,
            'Agency': agencies
        })

        filename = f"enforcement_actions_{start_year}_{start_month}.csv"
        df.to_csv(filename, index=False)

        print(f"Scraping complete. Data saved to {filename}.")

scrape_enforcement_actions(1, 2023)
```

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

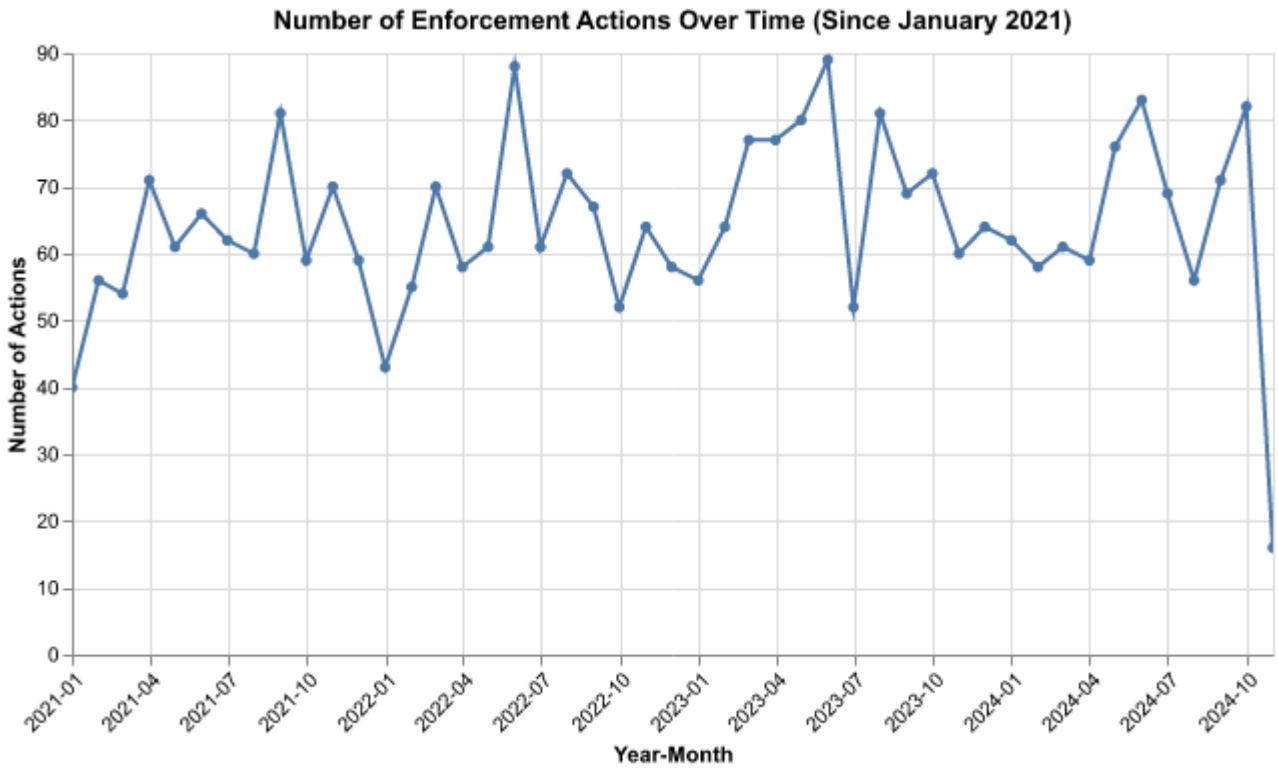
```
scrape_enforcement_actions(1, 2021)

filename = "enforcement_actions_2021_1.csv"
df = pd.read_csv(filename)

total_actions = len(df)
print(f"Total number of enforcement actions scraped: {total_actions}")
```

```
earliest_action = df.iloc[-1]
print(f"Earliest enforcement action: Date - {earliest_action['Date']}, Title
 ↪  - {earliest_action['Title']}, Link - {earliest_action['Link']}")
```

Step 3: Plot data based on scraped data

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

```
file_path =
 ↪  "/Users/cynthia/Desktop/problem-set-5-xy-wz/enforcement_actions_2021_1.csv"
data = pd.read_csv(file_path)

data['Date'] = pd.to_datetime(data['Date'], errors='coerce')
data = data.dropna(subset=['Date'])

data['YearMonth'] = data['Date'].dt.to_period('M')

monthly_counts = data.groupby('YearMonth').size().reset_index(name='Count')

monthly_counts['YearMonth'] = monthly_counts['YearMonth'].astype(str)

chart = alt.Chart(monthly_counts).mark_line(point=True).encode(
    alt.X('YearMonth:T', title='Year-Month', axis=alt.Axis(format='%Y-%m',
 ↪  labelAngle=-45)),
    alt.Y('Count:Q', title='Number of Actions'),
    tooltip=['YearMonth', 'Count']
).properties(
    title='Number of Enforcement Actions Over Time (Since January 2021)',
    width=600,
    height=300
).interactive()


chart.show()
```

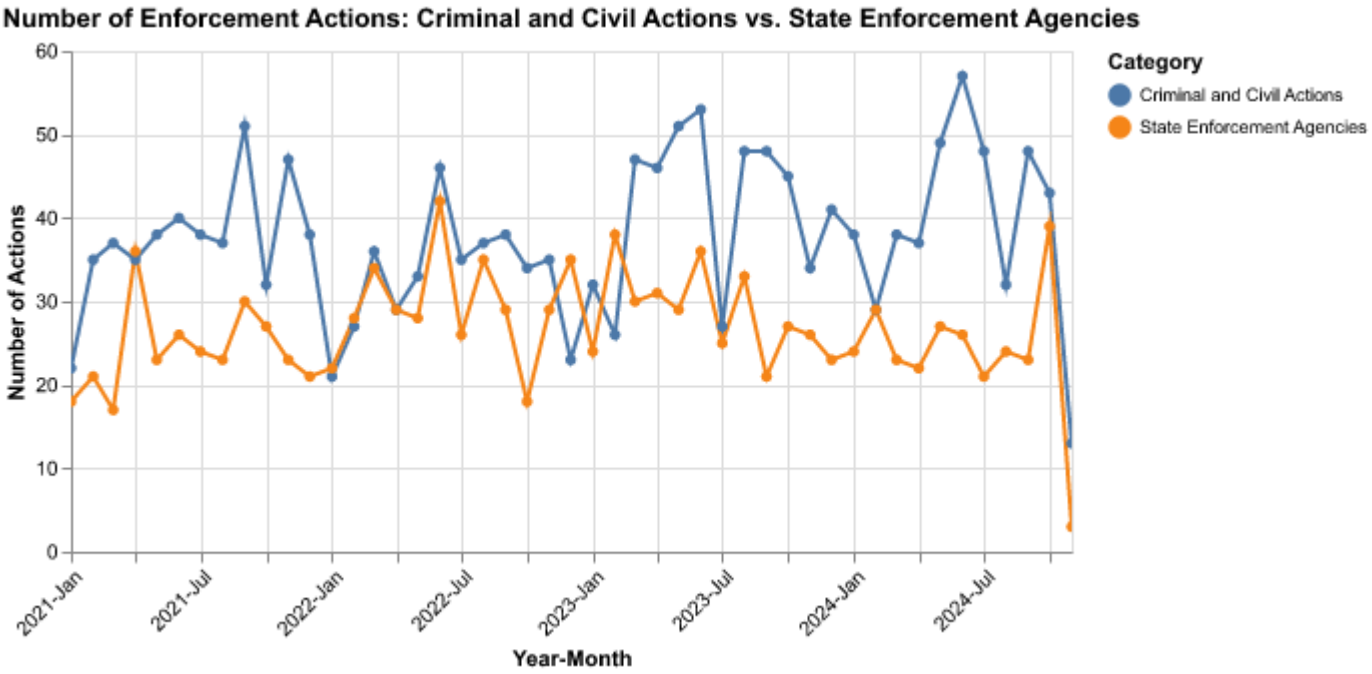


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

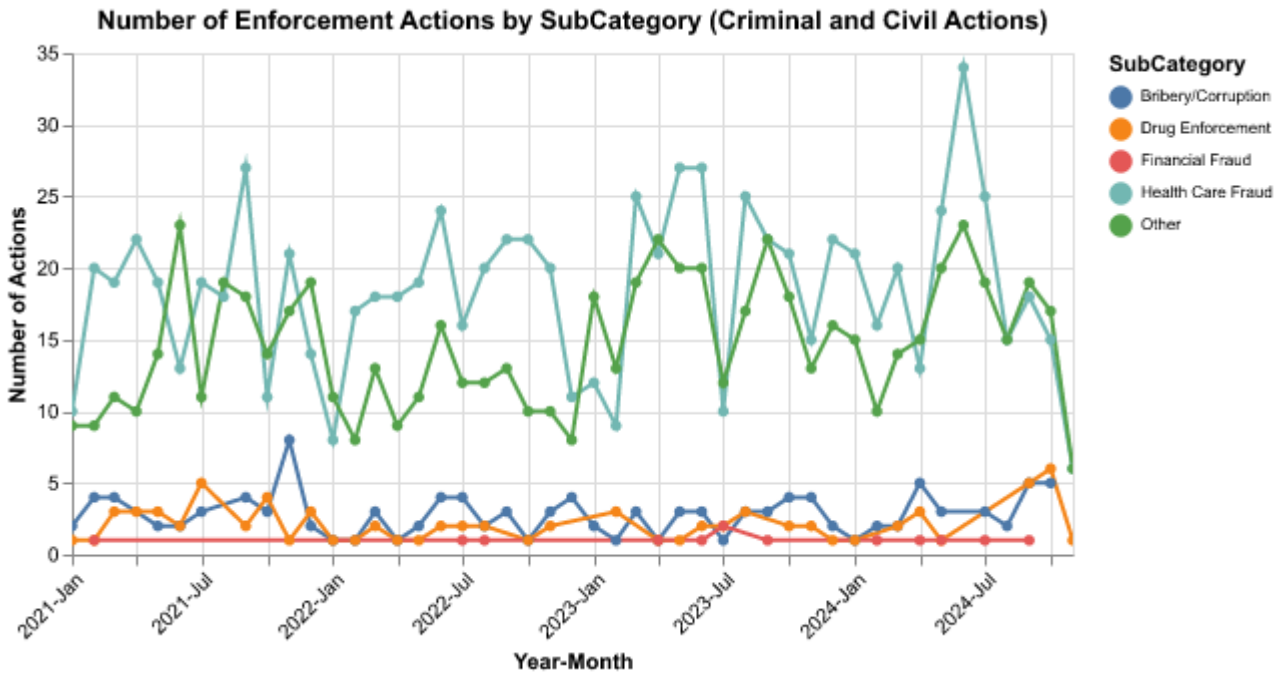
```
data['Date'] = pd.to_datetime(data['Date'], errors='coerce')
data = data.dropna(subset=['Date'])

data['YearMonth'] = data['Date'].dt.to_period('M').astype(str)

data['Category'] = data['Category'].apply(lambda x: 'Criminal and Civil
 ↪  Actions' if 'Criminal and Civil Actions' in x else 'State Enforcement
 ↪  Agencies')

def assign_subcategory(title):
    title_lower = title.lower()
    if 'health' in title_lower or 'medicare' in title_lower or 'medicaid' in
     ↪  title_lower:
        return 'Health Care Fraud'
    elif 'financial' in title_lower or 'bank' in title_lower or 'money
     ↪  laundering' in title_lower:
```

```
        return 'Financial Fraud'
    elif 'drug' in title_lower or 'narcotic' in title_lower:
        return 'Drug Enforcement'
    elif 'bribe' in title_lower or 'corrupt' in title_lower or 'kickback' in
     ↪  title_lower:
        return 'Bribery/Corruption'
    else:
        return 'Other'

data['SubCategory'] = data.apply(lambda row: assign_subcategory(row['Title'])
 ↪  if row['Category'] == 'Criminal and Civil Actions' else None, axis=1)

main_category_counts = data.groupby(['YearMonth',
 ↪  'Category']).size().reset_index(name='Count')

main_chart = alt.Chart(main_category_counts).mark_line(point=True).encode(
    alt.X('YearMonth:T', title='Year-Month', axis=alt.Axis(labelAngle=-45,
 ↪  format='%Y-%b')),
    alt.Y('Count:Q', title='Number of Actions'),
    color=alt.Color('Category:N', legend=alt.Legend(labelFontSize=9)),
    tooltip=['YearMonth', 'Category', 'Count']
).properties(
    title='Number of Enforcement Actions: Criminal and Civil Actions vs.
 ↪  State Enforcement Agencies',
    width=500,
    height=250
).interactive()

main_chart.show()
```



- based on five topics

```
subcategory_data = data[data['Category'] == 'Criminal and Civil Actions']
subcategory_counts = subcategory_data.groupby(['YearMonth',
 ↪  'SubCategory']).size().reset_index(name='Count')
subcategory_counts['YearMonth'] = subcategory_counts['YearMonth'].astype(str)

subcategory_chart =
 ↪  alt.Chart(subcategory_counts).mark_line(point=True).encode(
    alt.X('YearMonth:T', title='Year-Month', axis=alt.Axis(labelAngle=-45,
 ↪  format='%Y-%b')),
    alt.Y('Count:Q', title='Number of Actions'),
    color=alt.Color('SubCategory:N', legend=alt.Legend(labelFontSize=8)),
    tooltip=['YearMonth', 'SubCategory', 'Count']
).properties(
    title='Number of Enforcement Actions by SubCategory (Criminal and Civil
 ↪  Actions)',
    width=500,
    height=250
).interactive()

subcategory_chart.show()
```

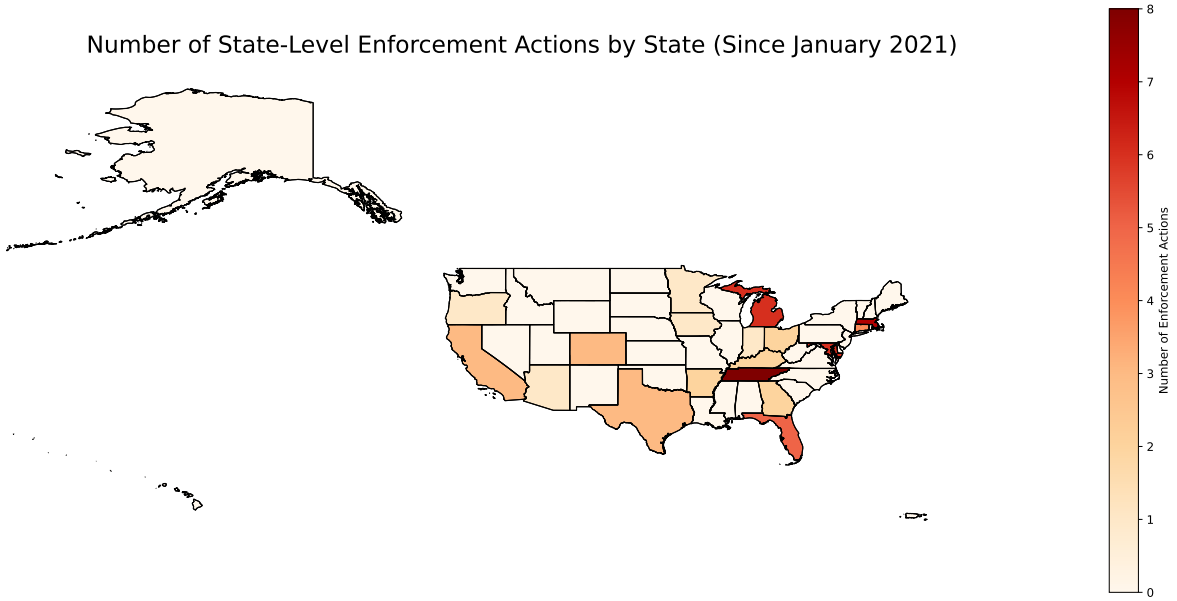


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
state_data = data[data['Agency'].str.contains(
    'State of', na=False, case=False)]


def extract_state_name(agency):
    parts = agency.split()
    if "State" in parts and "of" in parts:
        index = parts.index("of") + 1
        if index < len(parts):
            return parts[index]
    return None


state_data['State'] = state_data['Agency'].apply(extract_state_name)

state_data = state_data.dropna(subset=['State'])
```

```
state_data['State'] = state_data['State'].str.strip().str.title()

state_counts = state_data['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Count']

gdf_states = gpd.read_file(

    ↪ "/Users/cynthia/Desktop/problem-set-5-xy-wz/cb_2018_us_state_500k/cb_2018_us_state_5

gdf_states['NAME'] = gdf_states['NAME'].str.strip().str.title()

gdf_merged = gdf_states.merge(
    state_counts, how='left', left_on='NAME', right_on='State')
gdf_merged['Count'] = gdf_merged['Count'].fillna(0)

fig, ax = plt.subplots(1, 1, figsize=(20, 15))

gdf_merged.boundary.plot(ax=ax, linewidth=1, edgecolor='black')
gdf_merged.plot(column='Count', cmap='OrRd', linewidth=0.8, ax=ax,
 ↪ edgecolor='black', legend=True,
               legend_kwds={'label': "Number of Enforcement Actions",
                            'shrink': 0.6,
                            })



plt.title('Number of State-Level Enforcement Actions by State (Since January
 ↪ 2021)', fontsize=20)
plt.axis('off')
ax.set_aspect('equal')
ax.set_xlim(-180, -50)
ax.set_ylim(15, 75)
plt.show()
```

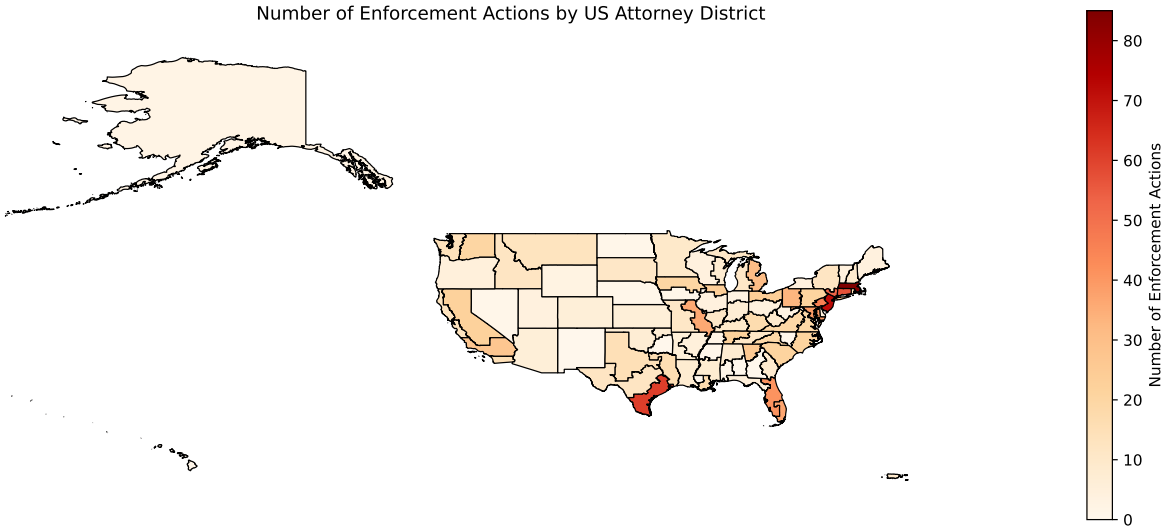


2. Map by District (PARTNER 2)

```
enforcement_data_path =
 ↪  "/Users/cynthia/Desktop/problem-set-5-xy-wz/enforcement_actions_2021_1.csv"
shapefile_path = "/Users/cynthia/Desktop/problem-set-5-xy-wz/US Attorney
 ↪  Districts Shapefile
 ↪  simplified_20241109/geo_export_8dfa149c-a5a0-43ab-aa3e-6a96f6f0b711.shp"

enforcement_data = pd.read_csv(enforcement_data_path)
enforcement_data['Date'] = pd.to_datetime(enforcement_data['Date'],
 ↪  errors='coerce')
enforcement_data = enforcement_data[enforcement_data['Date'] >= '2021-01-01']

def extract_district_name(agency):
    if pd.isna(agency):
        return None

    agency = agency.lower().strip()

    # Specific matches for multi-district regions and known variations
    if "northern district of florida" in agency:
        return "northern district of florida"
    elif "middle district of florida" in agency:
```

```
        return "middle district of florida"
    elif "southern district of florida" in agency:
        return "southern district of florida"
    elif "eastern district of pennsylvania" in agency:
        return "eastern district of pennsylvania"
    elif "western district of pennsylvania" in agency:
        return "western district of pennsylvania"
    elif "middle district of pennsylvania" in agency:
        return "middle district of pennsylvania"
    elif "district of columbia" in agency:
        return "district of columbia"
    elif "northern district of tennessee" in agency:
        return "northern district of tennessee"
    elif "middle district of tennessee" in agency:
        return "middle district of tennessee"
    elif "western district of tennessee" in agency:
        return "western district of tennessee"
    elif "district of idaho boise" in agency:
        return "district of idaho"
    elif "southern district of texas and u" in agency:
        return "southern district of texas"
    elif "southern district of pennsylvania" in agency:
        return "southern district of pennsylvania"
    elif "eastern district of pennsylvani" in agency:
        return "eastern district of pennsylvania"

    match =
 ↪  re.search(r'((?:northern|southern|eastern|western|central)?\s*district of
 ↪  [a-z\s]+)', agency)
    if match:
        return match.group(0).strip().lower()

    return None

enforcement_data['cleaned_district'] =
 ↪  enforcement_data['Agency'].apply(extract_district_name)
enforcement_data = enforcement_data.dropna(subset=['cleaned_district'])

district_counts =
 ↪  enforcement_data['cleaned_district'].value_counts().reset_index()
district_counts.columns = ['district', 'count']
```

```
us_districts_gdf = gpd.read_file(shapefile_path)
us_districts_gdf['district'] =
 ↪  us_districts_gdf['judicial_d'].str.lower().str.strip()

merged_gdf = us_districts_gdf.merge(district_counts, how='left',
 ↪  on='district')

merged_gdf['count'] = merged_gdf['count'].fillna(0)

fig, ax = plt.subplots(1, 1, figsize=(15, 10))
merged_gdf.plot(column='count', cmap='OrRd', linewidth=0.8, ax=ax,
 ↪  edgecolor='black', legend=True,
                legend_kwds={'label': "Number of Enforcement Actions",
 ↪  'shrink': 0.6})
plt.title('Number of Enforcement Actions by US Attorney District')
plt.axis('off')
ax.set_aspect('equal')
ax.set_xlim(-180, -50)
ax.set_ylim(15, 75)
plt.axis('off')
plt.show()
```



Extra Credit

1. Merge zip code shapefile with population

```
population_data_path =
 ↪  "/Users/cynthia/Desktop/problem-set-5-xy-wz/population/2020_data.csv"

population_data = pd.read_csv(population_data_path)
population_data = population_data[['GEO_ID', 'NAME',
 ↪  'P1_001N']].rename(columns={'NAME': 'ZIPCode', 'P1_001N': 'Population'})
population_data['ZIPCode'] =
 ↪  population_data['ZIPCode'].str.extract(r'(\d{5})')[0]
population_data['Population'] = pd.to_numeric(population_data['Population'],
 ↪  errors='coerce').fillna(0)

zip_gdf =
 ↪  gpd.read_file("/Users/cynthia/Desktop/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp")
zip_gdf = zip_gdf.rename(columns={'ZCTA5': 'ZIPCode'})
zip_gdf['ZIPCode'] = zip_gdf['ZIPCode'].astype(str)

merged_gdf = zip_gdf.merge(population_data, on='ZIPCode', how='left')
```

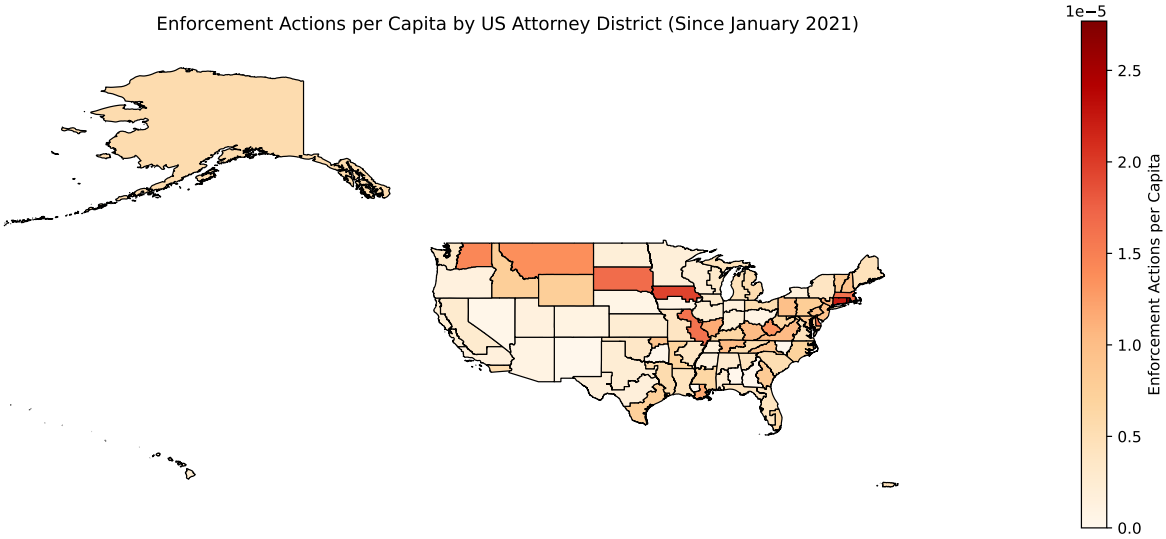
2. Conduct spatial join

```
district_shapefile_path = "/Users/cynthia/Desktop/problem-set-5-xy-wz/US
 ↪  Attorney Districts Shapefile
 ↪  simplified_20241109/geo_export_8dfa149c-a5a0-43ab-aa3e-6a96f6f0b711.shp"

district_gdf = gpd.read_file(district_shapefile_path)
district_gdf = district_gdf.rename(columns={'judicial_d': 'district'})
district_gdf['district'] = district_gdf['district'].str.lower().str.strip()

merged_gdf = merged_gdf.to_crs(district_gdf.crs)

zip_district_join = gpd.sjoin(merged_gdf, district_gdf, how="inner",
 ↪  predicate="within")
district_population =
 ↪  zip_district_join.groupby('district')['Population'].sum().reset_index()
district_population.columns = ['district', 'totalpopulation']
```

3. Map the action ratio in each district

```
import re

def extract_district_name(agency):
    if pd.isna(agency):
        return None
    agency = agency.lower().strip()

    if "northern district of florida" in agency:
        return "northern district of florida"
    elif "middle district of florida" in agency:
        return "middle district of florida"
    elif "southern district of florida" in agency:
        return "southern district of florida"
    elif "eastern district of pennsylvania" in agency:
        return "eastern district of pennsylvania"
    elif "western district of pennsylvania" in agency:
        return "western district of pennsylvania"
    elif "middle district of pennsylvania" in agency:
        return "middle district of pennsylvania"
    elif "district of columbia" in agency:
        return "district of columbia"
    elif "northern district of tennessee" in agency:
        return "northern district of tennessee"
    elif "middle district of tennessee" in agency:
        return "middle district of tennessee"
    elif "western district of tennessee" in agency:
        return "western district of tennessee"
    elif "district of idaho boise" in agency:
        return "district of idaho"
    elif "southern district of texas and u" in agency:
        return "southern district of texas"
    elif "southern district of pennsylvania" in agency:
        return "southern district of pennsylvania"
    elif "eastern district of pennsylvani" in agency:
        return "eastern district of pennsylvania"

    match =
↪   re.search(r'((?:northern|southern|eastern|western|central)?\s*district of
↪   [a-z\s]+)', agency)
```

```
    if match:
        return match.group(0).strip().lower()

    return None

enforcement_data['district'] =
 ↪  enforcement_data['Agency'].apply(extract_district_name)
enforcement_data = enforcement_data.dropna(subset=['district'])

district_enforcement_counts =
 ↪  enforcement_data['district'].value_counts().reset_index()
district_enforcement_counts.columns = ['district', 'EnforcementActions']
district_data = district_gdf.merge(district_population, on='district',
 ↪  how='left')
district_data = district_data.merge(district_enforcement_counts,
 ↪  on='district', how='left')

district_data['totalpopulation'] = district_data['totalpopulation'].fillna(0)
district_data['EnforcementActions'] =
 ↪  district_data['EnforcementActions'].fillna(0)
district_data['EnforcementPerCapita'] = district_data['EnforcementActions'] /
 ↪  district_data['totalpopulation']
district_data['EnforcementPerCapita'] =
 ↪  district_data['EnforcementPerCapita'].replace([float('inf'),
 ↪  -float('inf')], 0)
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
district_data.plot(column='EnforcementPerCapita', cmap='OrRd', linewidth=0.8,
 ↪  ax=ax, edgecolor='black', legend=True,
                   legend_kwds={'label': "Enforcement Actions per Capita",
 ↪  'shrink': 0.6})
plt.title('Enforcement Actions per Capita by US Attorney District (Since
 ↪  January 2021)')
plt.axis('off')
ax.set_aspect('equal')
ax.set_xlim(-180, -50)
ax.set_ylim(15, 75)
plt.show()
```


Enforcement Actions per Capita by US Attorney District (Since January 2021)
1e−5
2.5
2.0
1.5
1.0
0.5
0.0
Enforcement Actions per Capita