Problem Set 5

AUTHOR
Kaijie Wu, Griffin Sharps

PUBLISHED
October 24, 2001

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): Kaijie Wu (kaijie7)
   - Partner 2 (name and cnet ID): Griffin Sharps (gsharps)
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: Kaijie Wu, Griffin Sharps
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: **__** Late coins left after submission: **__**
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 datetime import datetime
import requests
from bs4 import BeautifulSoup
import re
import geopandas as gpd
import json

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

```
RendererRegistry.enable('png')
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
# Define the target URL for scraping data
oig_url = 'https://oig.hhs.gov/fraud/enforcement/'

# Initiate a GET request to fetch data from the URL
oig_web_data = requests.get(oig_url)

# Utilize BeautifulSoup to parse the fetched HTML content
oig_parsed_html = BeautifulSoup(oig_web_data.text, 'html.parser')

# Prepare an empty list to hold the data of enforcement actions
collected_enforcement_data = []

# Iterate over each element in the section containing the data
for element in oig_parsed_html.find_all('li', class_='usa-card'):
    # Locate and clean the title and link, ensuring all entries are captured even if some details are missing
    title_element = element.find('a')
    title = title_element.text.strip() if title_element else 'No title found'
    link = title_element['href'] if title_element else 'No link found'

    # Search for and format the date, managing missing entries effectively
    date_element = element.find('span', class_='text-base-dark')
    date = date_element.text.strip() if date_element else 'No date found'

    # Identify and strip the category, accounting for possible absences
    category_element = element.find('li', class_='display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1')
    category = category_element.text.strip() if category_element else 'No category found'

    # Append the collected details to the list
    collected_enforcement_data.append({
        'Title': title,
        'Date': date,
        'Category': category,
        'Link': link
    })

# Transform the collected list of dictionaries into a DataFrame for better data management
enforcement_dataframe = pd.DataFrame(collected_enforcement_data)

# Print the top rows of the DataFrame to verify correct data collection
```

```
print(enforcement_dataframe.head())

# Export the DataFrame to a CSV file in the current directory
enforcement_dataframe.to_csv('enforcement_actions.csv', index=False)
```

```
                                        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  /fraud/enforcement/pharmacist-and-brother-conv...
1  /fraud/enforcement/boise-nurse-practitioner-se...
2  /fraud/enforcement/former-traveling-nurse-plea...
3  /fraud/enforcement/former-arlington-resident-s...
4  /fraud/enforcement/paroled-felon-sentenced-to-...
```

2. Crawling (PARTNER 1)

```
# List to hold data with additional 'Agency' field
collected_data_with_agency = []

# Iterate over each enforcement action entry found in parsed HTML
for element in oig_parsed_html.find_all('li', class_='usa-card'):
    title_element = element.find('a')
    title = title_element.text.strip() if title_element else 'No title found'
    link = 'https://oig.hhs.gov' + title_element['href'] if title_element else 'No link found'

    # Fetch and parse the linked page for agency information
    if title_element:
        agency_response = requests.get(link)
        agency_soup = BeautifulSoup(agency_response.text, 'html.parser')
        agency_info = agency_soup.find(lambda tag: tag.name == "span" and 'Agency:' in tag.text)
        agency_name = agency_info.next_sibling.text.strip() if agency_info and agency_info.next_sibling else 'No agency found'
    else:
        agency_name = 'No agency found'

    # Use previously parsed date and category information
    date_element = element.find('span', class_='text-base-dark')
    date = date_element.text.strip() if date_element else 'No date found'

    category_element = element.find('li', class_='display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1')
    category = category_element.text.strip() if category_element else 'No category found'

    # Append the additional details to the list
    collected_data_with_agency.append({
        'Title': title,
        'Date': date,
        'Category': category,
        'Link': link,
        'Agency': agency_name
    })

# Create a DataFrame and save to CSV
detailed_dataframe = pd.DataFrame(collected_data_with_agency)
print(detailed_dataframe.head())
detailed_dataframe.to_csv('enforcement_actions_detailed.csv', index=False)
```

```
                                        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)

Define the function and its 2 inputs

Check for valid year

Use an if statement: if start_year earlier than 2013, print a message telling user its invalid. No else statement needed.

Setup base_url and an empty list as in previous code.

Create datetime object from inputs and grab today's date as a datetime object.

Use a While loop. The loop should go through the site's pages until it finds a date outside of the range specified. That is, it should start with the current date and move backwards until it finds a date older than the function inputs. But we can't do that here. It makes more sense to do it in the For loop below. The While loop also needs a counter to move forward through the website's pages.

Use requests.get() to fetch the data from the page.

Then, using the code above as a guide...

Use BeautifulSoup to parse the HTML content of the page. This code to be based on the code above.

Begin a For loop. For each action find: Title and Link Date Category Agency Information: This is where we create the conditions to exit the While loop. If the date is older than the inputs of the function, the function should just return what its got and stop looping.

Create a dictionary of the actions. Append this dictionary to empty list.

Use time.sleep(1) to wait for 1 second.

Increment the page counter by 1 to go to next page.

After the loop, convert the created list (of dictionaries) to a DataFrame. Save the DataFrame as a csv.

- b. Create Dynamic Scraper (PARTNER 2)

```
def scrape_function(start_month, start_year):
    # Check that the year is within bounds
    if start_year < 2013:
        print("Please restrict the year to 2013 or later.")
        return

    # Define the base URL
    base_url = 'https://oig.hhs.gov/fraud/enforcement/'
    collected_data = []

    # Convert start date to datetime and determine today's date
    start_date = datetime(start_year, start_month, 1)
    today_date = datetime.today()

    # Paginate through results until we reach the start_date
    page = 1
    while True:
        # Fetch data for the current page
        url = f"{base_url}?page={page}"
        response = requests.get(url)
        if response.status_code != 200:
            print("Failed to retrieve data.")
            break

        # Parse the HTML content
        soup = BeautifulSoup(response.text, 'html.parser')
        actions = soup.find_all('li', class_='usa-card')

        # Process each enforcement action item
        for action in actions:
            title_element = action.find('a')
            title = title_element.text.strip() if title_element else 'No title found'
            link = 'https://oig.hhs.gov' + title_element['href'] if title_element else 'No link found'

            # Extract and parse the date
            date_element = action.find('span', class_='text-base-dark')
            date_str = date_element.text.strip() if date_element else 'No date found'
            date_obj = datetime.strptime(date_str, '%B %d, %Y') if date_element else None

            # Break out of loop if the date is before start_date
            if date_obj and date_obj < start_date:
                # Convert to DataFrame and return if we've reached the start date
                df = pd.DataFrame(collected_data)
                filename = f"enforcement_actions_{start_year}_{start_month:02d}.csv"
```

```
                df.to_csv(filename, index=False)
                print(f"Data saved to {filename}")
                return df

            # Extract category
            category_element = action.find('li', class_='display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1')
            category = category_element.text.strip() if category_element else 'No category found'

            # Visit link to extract agency info
            agency_name = 'No agency found'
            if link != 'No link found':
                agency_response = requests.get(link)
                agency_soup = BeautifulSoup(agency_response.text, 'html.parser')
                agency_info = agency_soup.find(lambda tag: tag.name == "span" and 'Agency:' in tag.text)
                if agency_info and agency_info.next_sibling and hasattr(agency_info.next_sibling, 'text'):
                    agency_name = agency_info.next_sibling.text.strip()

            # Append data
            collected_data.append({
                'Title': title,
                'Date': date_str,
                'Category': category,
                'Link': link,
                'Agency': agency_name
            })

        # Wait before going to the next page to avoid server blocks
        time.sleep(1)

        # Move to the next page
        page += 1

    # Ensure collected_data is returned as a DataFrame at the end of scraping
    df = pd.DataFrame(collected_data)
    filename = f"enforcement_actions_{start_year}_{start_month:02d}.csv"
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")
    return df

# Example usage of the function
test_df = scrape_function(1, 2023)

# Print total enforcement actions in the DataFrame
print(f"\nTotal enforcement actions: {len(test_df)}")

# Find and print the earliest enforcement action
if not test_df.empty:
    earliest_action = test_df.sort_values(by='Date').iloc[0]
    print("Earliest Enforcement Action:")
    print(earliest_action)
```

```
Data saved to enforcement_actions_2023_01.csv

Total enforcement actions: 1534
Earliest Enforcement Action:
Title       Shelby County Man Indicted for Offenses Relati...
Date                                            April 1, 2024
Category                           Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/shelby-c...
Agency      U.S. Attorney's Office, Eastern District of Ke...
Name: 507, dtype: object
```

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

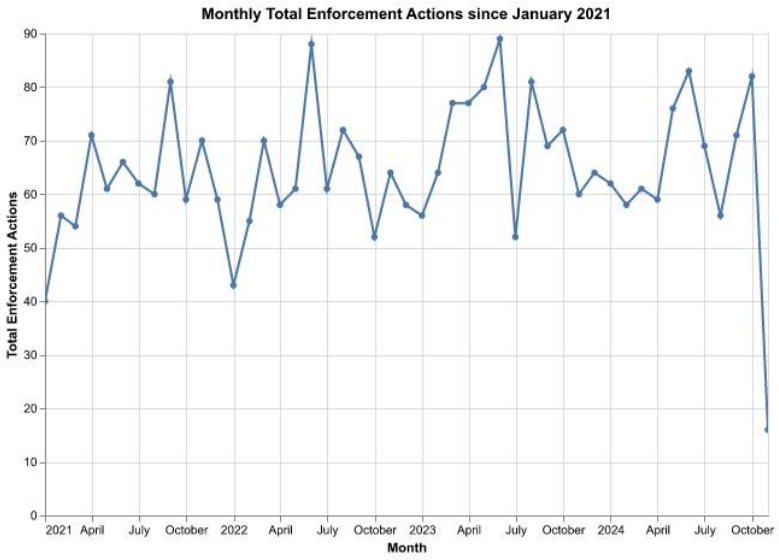
```
# Example usage of the function
partner_test_df = scrape_function(1, 2021)

# Print total enforcement actions in the DataFrame
print(f"\nTotal enforcement actions: {len(partner_test_df)}")

# Find and print the earliest enforcement action
if not partner_test_df.empty:
    earliest_action_partner = partner_test_df.sort_values(by='Date').iloc[0]
    print("Earliest Enforcement Action:")
    print(earliest_action_partner)
```

```
Data saved to enforcement_actions_2021_01.csv

Total enforcement actions: 3022
Earliest Enforcement Action:
Title       Red Rocks Radiation and Oncology, Alliance Hea...
Date                                            April 1, 2021
Category                                 Fraud Self-Disclosures
Link        https://oig.hhs.gov/fraud/enforcement/red-rock...
Agency                                          No agency found
Name: 2870, dtype: object
```

Step 3: Plot data based on scraped data

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

```
# Load the data
file_path = (r'C:\Users\james\Desktop\Kajie-Griffin-ps5\enforcement_actions_2021_01.csv')
enforcement_data = pd.read_csv(file_path)

# Convert 'Date' column to datetime format and drop any rows with invalid dates
enforcement_data['Date'] = pd.to_datetime(enforcement_data['Date'], errors='coerce')
enforcement_data = enforcement_data.dropna(subset=['Date'])

# Extract month and year from 'Date' and group by it
enforcement_data['YearMonth'] = enforcement_data['Date'].dt.to_period('M')
monthly_counts = enforcement_data.groupby('YearMonth').size().reset_index(name='Count')

# Convert YearMonth to string for Altair compatibility
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].astype(str)

# Create the Altair line chart showing total enforcement actions over time
chart = alt.Chart(monthly_counts).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Month'),
    y=alt.Y('Count', title='Total Enforcement Actions'),
    tooltip=['YearMonth', 'Count']
).properties(
    title='Monthly Total Enforcement Actions since January 2021',
    width=600,
    height=400
).interactive()

chart.display()
```

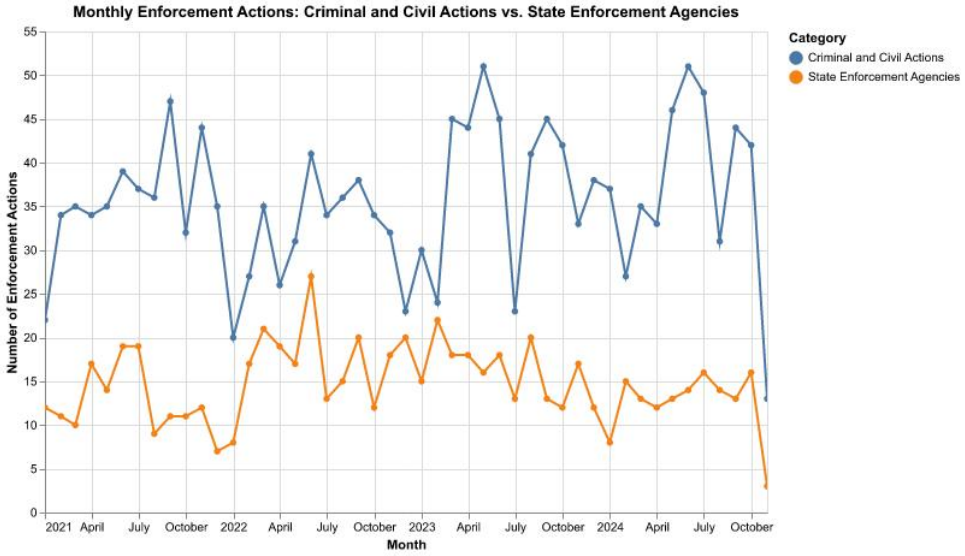


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

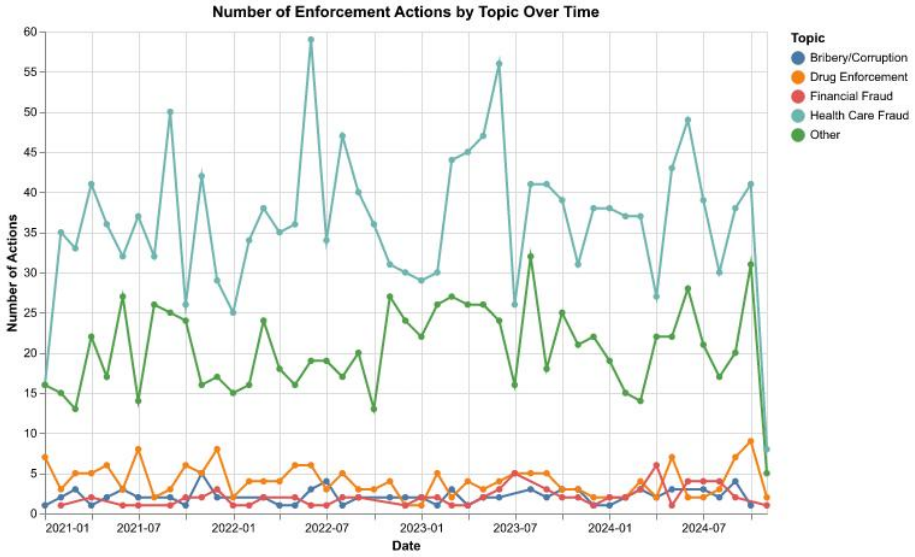
```
# Load the data
file_path = (r'C:\Users\james\Desktop\Kajie-Griffin-ps5\enforcement_actions_2021_01.csv')
enforcement_data = pd.read_csv(file_path)

# Convert 'Date' column to datetime format to handle monthly grouping
enforcement_data['Date'] = pd.to_datetime(enforcement_data['Date'], errors='coerce')

# Drop rows where 'Date' conversion failed
enforcement_data = enforcement_data.dropna(subset=['Date'])

# Extract month and year from 'Date' for grouping
enforcement_data['YearMonth'] = enforcement_data['Date'].dt.to_period('M')

# Filter the data to include only the specified categories
filtered_data = enforcement_data[enforcement_data['Category'].isin(['Criminal and Civil Actions', 'State Enforcement Agencies'])]

# Group by YearMonth and Category, counting occurrences in each group
filtered_monthly_counts = filtered_data.groupby(['YearMonth', 'Category']).size().unstack(fill_value=0)

# Reset the index and convert YearMonth to string for Altair compatibility
filtered_monthly_counts = filtered_monthly_counts.reset_index()
filtered_monthly_counts['YearMonth'] = filtered_monthly_counts['YearMonth'].astype(str)

# Melt the dataframe for Altair compatibility
filtered_long_df = filtered_monthly_counts.melt(id_vars='YearMonth',
                                                var_name='Category',
                                                value_name='Count')

# Create the Altair line chart
```

```
chart = alt.Chart(filtered_long_df).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Month'),
    y=alt.Y('Count', title='Number of Enforcement Actions'),
    color=alt.Color('Category', legend=alt.Legend(title='Category')),
    tooltip=['YearMonth', 'Category', 'Count']
).properties(
    title='Monthly Enforcement Actions: Criminal and Civil Actions vs. State Enforcement Agencies',
    width=600,
    height=400
).interactive()

chart.display()
```



- based on five topics

```
# Load the dataset
data = pd.read_csv(r'C:\Users\james\Desktop\Kajie-Griffin-ps5\enforcement_actions_2021_01.csv')

# Function to categorize titles based on keywords
def categorize_title(title):
    title = title.lower()
    if any(word in title for word in ["health", "medical", "hospital", "clinic", "medicare", "medicaid"]):
        return "Health Care Fraud"
    elif any(word in title for word in ["bank", "financial", "money laundering", "fraudulent", "embezzlement"]):
        return "Financial Fraud"
    elif any(word in title for word in ["drug", "pharmacy", "narcotic", "opioid"]):
        return "Drug Enforcement"
    elif any(word in title for word in ["bribery", "corrupt", "kickback"]):
        return "Bribery/Corruption"
    else:
        return "Other"

# Apply categorization function to the title column
data['Topic'] = data['Title'].apply(categorize_title)

# Convert 'Date' to datetime and format it as 'YYYY-MM'
data['Date'] = pd.to_datetime(data['Date']).dt.strftime('%Y-%m')

# Group the data by Topic and Date and count the occurrences
grouped_data = data.groupby(['Topic', 'Date']).size().reset_index(name='Number of Actions')

# Create a line chart using Altair
line_chart = alt.Chart(grouped_data).mark_line(point=True).encode(
    x=alt.X('Date:T', title='Date', axis=alt.Axis(format='%Y-%m')),
    y=alt.Y('Number of Actions:Q', title='Number of Actions'),
    color='Topic:N',
    tooltip=['Topic', 'Date', 'Number of Actions']
).properties(
    title='Number of Enforcement Actions by Topic Over Time',
    width=600,
    height=400
)

# Display the chart
line_chart.display()
```

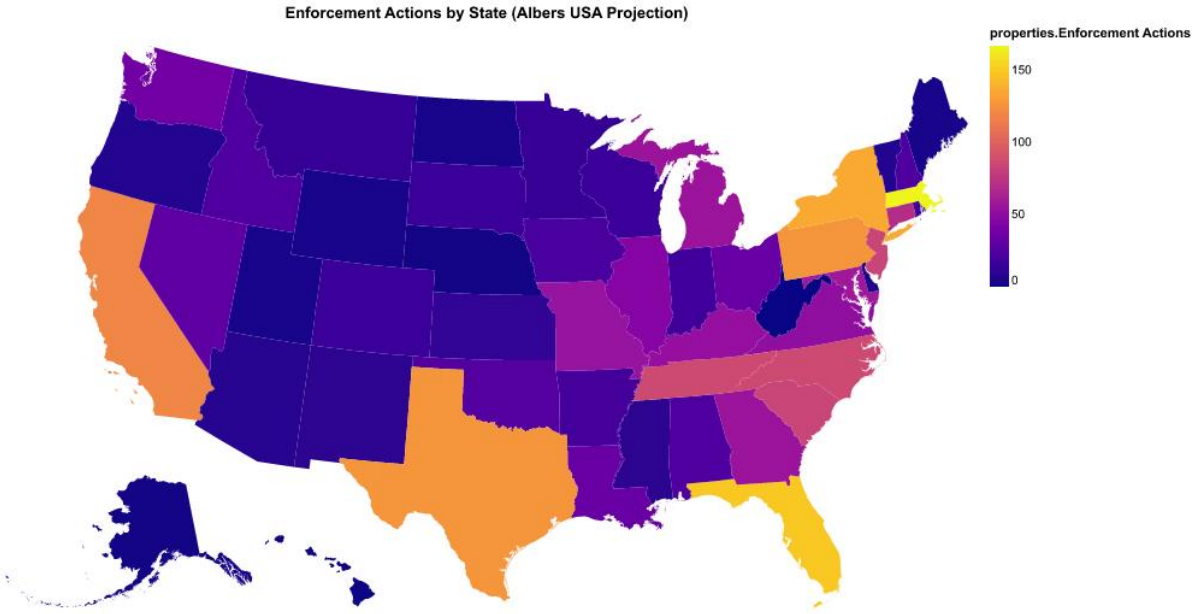


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

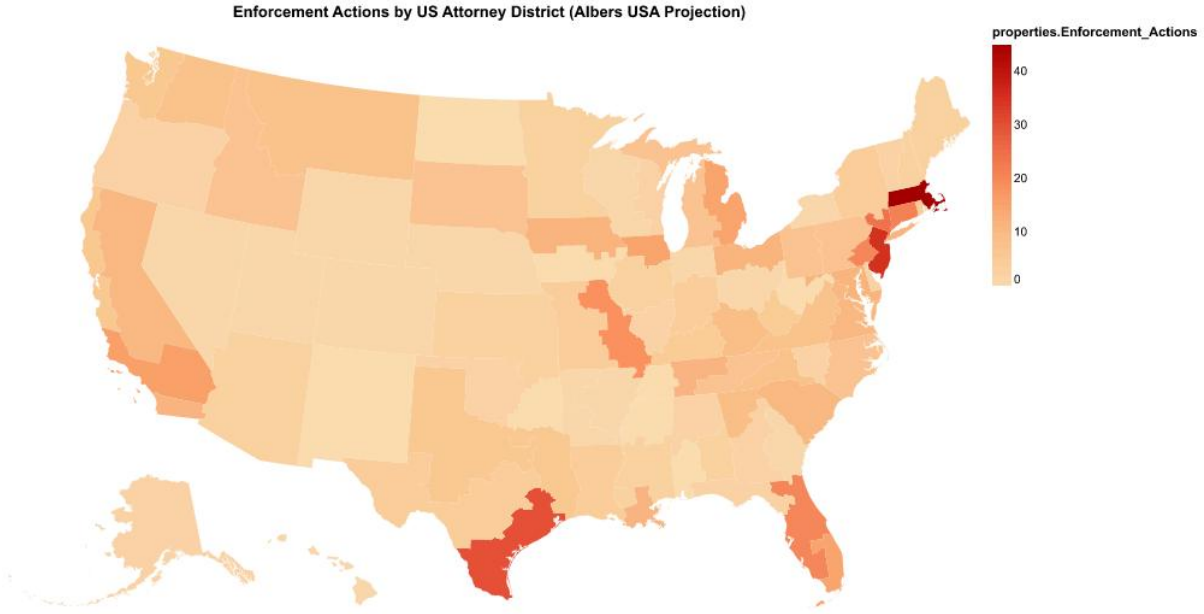
```
# Load the enforcement actions data
file_path = r'C:\Users\james\Desktop\Kajie-Griffin-ps5\enforcement_actions_2021_01.csv'
data = pd.read_csv(file_path)

# Define U.S. states list
us_states = [
    "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado",
    "Connecticut", "Delaware", "Florida", "Georgia", "Hawaii", "Idaho",
    "Illinois", "Indiana", "Iowa", "Kansas", "Kentucky", "Louisiana",
    "Maine", "Maryland", "Massachusetts", "Michigan", "Minnesota",
    "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",
    "New Hampshire", "New Jersey", "New Mexico", "New York",
    "North Carolina", "North Dakota", "Ohio", "Oklahoma", "Oregon",
    "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota",
    "Tennessee", "Texas", "Utah", "Vermont", "Virginia", "Washington",
    "West Virginia", "Wisconsin", "Wyoming"
]

# Define a function to find a state name in a text string
def get_state_from_text(text):
    text = text.lower()  # Normalize to lowercase for matching
    for state in us_states:
        if re.search(r"\b" + state.lower() + r"\b", text):
            return state
    return None

# Apply the function to extract state names
data['State'] = data['Agency'].apply(get_state_from_text)

# Count occurrences of each state and merge with the full state list
state_counts = data['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Enforcement Actions']

# Ensure all states are included, even those with no actions
state_data = pd.DataFrame(us_states, columns=['State']).merge(state_counts, on='State', how='left')
state_data['Enforcement Actions'].fillna(0, inplace=True)

# Load the shapefile and filter only for relevant states
shape_data = gpd.read_file(r'C:\Users\james\Desktop\cb_2018_us_state_500k\cb_2018_us_state_500k.shp')
shape_data = shape_data[shape_data['NAME'].isin(us_states)]  # Keep only the required states

# Merge state enforcement data with the shapefile data
map_with_data = shape_data.merge(state_data, left_on='NAME', right_on='State', how='left')
map_with_data['Enforcement Actions'].fillna(0, inplace=True)

# Convert to GeoJSON for Altair
map_with_data = map_with_data.to_crs("EPSG:4326")  # Albers USA projection in Altair requires WGS84
geojson_data = json.loads(map_with_data.to_json())

# Create the Altair chart with Albers USA projection
chart = alt.Chart(alt.Data(values=geojson_data['features'])).mark_geoshape().encode(
    color=alt.Color('properties.Enforcement Actions:Q', scale=alt.Scale(scheme='plasma')),
    tooltip=[
        alt.Tooltip('properties.State:N', title='State'),
        alt.Tooltip('properties.Enforcement Actions:Q', title='Enforcement Actions')
    ]
).project(
    type='albersUsa'
```

```
).properties(
    width=800,
    height=500,
    title='Enforcement Actions by State (Albers USA Projection)'
)

# Save or display the chart
chart.display()
```



2. Map by District (PARTNER 2)

```
# Load the enforcement actions data
enforcement_df = pd.read_csv(r'C:/Users/james/Desktop/Kajie-Griffin-ps5/enforcement_actions_2021_01.csv')
enforcement_df['District'] = enforcement_df['Agency'].str.extract(r'U\.S\. Attorney\'s Office, (.*)')[0]
enforcement_counts = enforcement_df['District'].value_counts().reset_index()
enforcement_counts.columns = ['District', 'Enforcement_Actions']

# Load the shapefile and rename column for merging
shapefile_gdf = gpd.read_file(r'C:/Users/james/Desktop/US Attorney Districts Shapefile simplified_20241109/geo_export_0fe6c47a-9f8d-4986-a592-42ca38b140ed.shp')
shapefile_gdf = shapefile_gdf.rename(columns={'judicial_d': 'District'})

# Merge shapefile data with enforcement counts
merged_gdf = shapefile_gdf.merge(enforcement_counts, on='District', how='left')
merged_gdf['Enforcement_Actions'] = merged_gdf['Enforcement_Actions'].fillna(0)

# Convert the GeoDataFrame to GeoJSON format
merged_gdf = merged_gdf.to_crs("EPSG:4326")  # Altair requires WGS84 (EPSG:4326)
geojson_data = json.loads(merged_gdf.to_json())

# Create an Altair chart with Albers USA projection
chart = alt.Chart(alt.Data(values=geojson_data['features'])).mark_geoshape().encode(
    color=alt.Color('properties.Enforcement_Actions:Q', scale=alt.Scale(scheme='orangered')),
    tooltip=[
        alt.Tooltip('properties.District:N', title='District'),
        alt.Tooltip('properties.Enforcement_Actions:Q', title='Enforcement Actions')
    ]
).project(
    type='albersUsa'
).properties(
    width=800,
    height=500,
    title='Enforcement Actions by US Attorney District (Albers USA Projection)'
)

# Save the chart as HTML
chart.display()
```



Extra Credit

1. Merge zip code shapefile with population

```
# Open the CSV file containing population information and adjust column names
pop_data = pd.read_csv(r'C:/Users/james/Desktop/DECENNIALDHC2020.P1_2024-11-09T130011/DECENNIALDHC2020.P1-Data.csv')
pop_data.columns = ['GEO_ID', 'NAME', 'Total_Population', 'Unused_Col']
pop_data = pop_data.iloc[1:].drop(columns=['Unused_Col'])  # Remove unneeded row and column
pop_data['ZIP_CODE'] = pop_data['GEO_ID'].str[-5:]  # Extract ZIP code as a new column
pop_data['Total_Population'] = pd.to_numeric(pop_data['Total_Population'])  # Convert to numeric format

# Load ZIP code shapefile and join with the population dataset
zip_shapefile = gpd.read_file(r'C:/Users/james/Desktop/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp')
combined_data = zip_shapefile.merge(pop_data, left_on='ZCTA5', right_on='ZIP_CODE', how='left')

# Display the resulting dataset
print(combined_data)
```

```
            GEO_ID_x  ZCTA5 NAME_x   LSAD  CENSUSAREA  \
0      8600000US01040  01040  01040  ZCTA5      21.281
1      8600000US01050  01050  01050  ZCTA5      38.329
2      8600000US01053  01053  01053  ZCTA5       5.131
3      8600000US01056  01056  01056  ZCTA5      27.205
4      8600000US01057  01057  01057  ZCTA5      44.907
...              ...    ...    ...    ...         ...
33115  8600000US57340  57340  57340  ZCTA5      79.952
33116  8600000US59910  59910  59910  ZCTA5      55.396
33117  8600000US60004  60004  60004  ZCTA5      11.082
33118  8600000US60048  60048  60048  ZCTA5      27.528
33119  8600000US60102  60102  60102  ZCTA5      14.495

                                              geometry        GEO_ID_y  \
0      POLYGON ((-72.62734 42.16203, -72.62764 42.162...  860Z200US01040
1      POLYGON ((-72.95393 42.34379, -72.95385 42.343...  860Z200US01050
2      POLYGON ((-72.68286 42.37002, -72.68287 42.369...  860Z200US01053
3      POLYGON ((-72.39529 42.18476, -72.39653 42.183...  860Z200US01056
4      MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...  860Z200US01057
...                                                  ...             ...
33115  POLYGON ((-97.76724 43.90034, -97.75081 43.900...  860Z200US57340
33116  MULTIPOLYGON (((-114.19828 47.8025, -114.1986 ...  860Z200US59910
33117  POLYGON ((-87.98511 42.14212, -87.98511 42.141...  860Z200US60004
33118  MULTIPOLYGON (((-87.95979 42.34804, -87.95935 ...  860Z200US60048
33119  POLYGON ((-88.3694 42.15406, -88.3694 42.15427...  860Z200US60102

            NAME_y  Total_Population ZIP_CODE
0      ZCTA5 01040           38238.0    01040
1      ZCTA5 01050            2467.0    01050
2      ZCTA5 01053            2031.0    01053
3      ZCTA5 01056           21002.0    01056
4      ZCTA5 01057            8152.0    01057
...            ...               ...      ...
33115  ZCTA5 57340             335.0    57340
33116  ZCTA5 59910             506.0    59910
33117  ZCTA5 60004           52546.0    60004
33118  ZCTA5 60048           29353.0    60048
33119  ZCTA5 60102           31959.0    60102

[33120 rows x 10 columns]
```

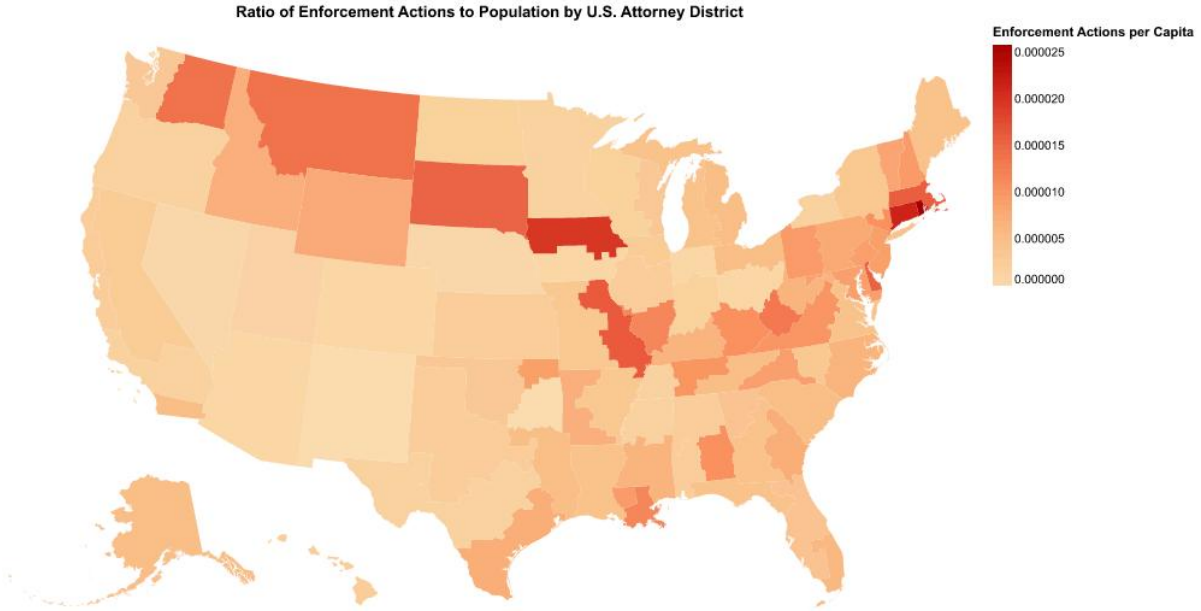
2. Conduct spatial join

```
# Load the district shapefile
gdf_map = gpd.read_file(r'C:\Users\james\Desktop\US Attorney Districts Shapefile simplified_20241109\geo_export_0fe6c47a-9f8d-4986-a592-42ca38b140ed.shp')

# Ensure CRS match
combined_data = combined_data.to_crs(gdf_map.crs)

# Spatial join to associate ZIP codes within districts
zip_district_joined = gpd.sjoin(combined_data, gdf_map, how='inner', predicate='within')

# Aggregate population by district
district_pop_total = zip_district_joined.groupby('judicial_d')['Total_Population'].sum().reset_index()

# Display the aggregated population by district
print(district_pop_total)
```

```
                      judicial_d  Total_Population
0   Central District of California        17541476.0
1     Central District of Illinois         1836364.0
2               District of Alaska          358718.0
3              District of Arizona         6648991.0
4             District of Colorado         5636000.0
..                             ...               ...
86   Western District of Tennessee         1207114.0
87       Western District of Texas         6840310.0
88    Western District of Virginia         1831829.0
89  Western District of Washington         4010453.0
90   Western District of Wisconsin         1944373.0

[91 rows x 2 columns]
```

3. Map the action ratio in each district

```
# Load enforcement actions data
enforcement_data = pd.read_csv(r'C:/Users/james/Desktop/Kajie-Griffin-ps5/enforcement_actions_2021_01.csv')
enforcement_data['Date'] = pd.to_datetime(enforcement_data['Date'], errors='coerce')

# Filter for actions since January 1, 2021, and extract district names
filtered_actions = enforcement_data[enforcement_data['Date'] >= '2021-01-01']
filtered_actions['District'] = filtered_actions['Agency'].str.extract(r'Office, (.*)')[0]

# Aggregate enforcement actions by district
actions_per_district = filtered_actions.groupby('District').size().reset_index(name='Enforcement_Actions')

# Ensure 'district_pop_total' has 'District' and 'Total_Population' columns
# Assume district_pop_total is already loaded with these columns
if 'judicial_d' in district_pop_total.columns:
    district_pop_total = district_pop_total.rename(columns={'judicial_d': 'District'})  # Align with actions_per_district

# Merge enforcement data with district population data
district_data = district_pop_total.merge(actions_per_district, on='District', how='left')
district_data['Enforcement_Actions'] = district_data['Enforcement_Actions'].fillna(0)  # Fill NaNs with 0 for districts with no actions
district_data['Enforcement_Ratio'] = district_data['Enforcement_Actions'] / district_data['Total_Population']

# Load the district shapefile for mapping
district_shapefile = gpd.read_file(r'C:/Users/james/Desktop/US Attorney Districts Shapefile simplified_20241109/geo_export_0fe6c47a-9f8d-4986-a592-42ca38b140ed.sh

# Merge the ratio data with the district shapefile
district_map = district_shapefile.merge(district_data, left_on='judicial_d', right_on='District', how='left')
district_map['Enforcement_Ratio'].fillna(0, inplace=True)  # Replace NaNs with 0 in Enforcement_Ratio for missing data

# Convert to GeoJSON for Altair
district_map = district_map.to_crs("EPSG:4326")  # Albers USA projection in Altair requires WGS84
geojson_data = json.loads(district_map.to_json())

# Create the Altair chart with Albers USA projection
chart = alt.Chart(alt.Data(values=geojson_data['features'])).mark_geoshape().encode(
    color=alt.Color('properties.Enforcement_Ratio:Q', scale=alt.Scale(scheme='orangered'), title="Enforcement Actions per Capita"),
    tooltip=[
        alt.Tooltip('properties.District:N', title='District'),
        alt.Tooltip('properties.Enforcement_Actions:Q', title='Enforcement Actions'),
        alt.Tooltip('properties.Total_Population:Q', title='Population'),
        alt.Tooltip('properties.Enforcement_Ratio:Q', title='Enforcement Ratio')
    ]
).project(
    type='albersUsa'
).properties(
    width=800,
    height=500,
    title='Ratio of Enforcement Actions to Population by U.S. Attorney District'
)

# Save the chart as HTML
chart.display()
```


Ratio of Enforcement Actions to Population by U.S. Attorney District
Enforcement Actions per Capita
0.000025
0.000020
0.000015
0.000010
0.000005
0.000000