30538 Problem Set 5

Joy Wu & Betsy Shi

Nov 9

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 (Betsy Shi, betsyshi):
    - Partner 2 (Joy Wu, lepengw):
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: BS & JW
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: 1 Late coins left after submission: 1
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, timedelta
import requests
from bs4 import BeautifulSoup
import numpy as np
import re
import geopandas as gpd
import matplotlib.pyplot as plt
import os
import json
alt.renderers.enable("png")

import warnings
warnings.filterwarnings('ignore')
```

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, 'lxml')

enforcement_actions = []

cards = soup.find_all('li', class_='usa-card')

for card in cards:
    title = card.find('h2').text.strip() if card.find('h2') else 'No title'
    short_link = card.find('a')['href'] if card.find('a') else None
    link = f"https://oig.hhs.gov{short_link}" if short_link else 'No link'
    date = card.find('span', class_='text-base-dark').text.strip() if
↪   card.find('span', class_='text-base-dark') else 'No date'
    category = card.find('li', class_='display-inline-block').text.strip()
↪   if card.find('li', class_='display-inline-block') else 'No category'

    enforcement_actions.append({
        "Title": title,
        "Date": date,
        "Category": category,
        "Link": link
    })
```

```
df = pd.DataFrame(enforcement_actions, columns=['Title', 'Date', 'Category',
 ↪  'Link'])
print(df)
```

```
                                                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
5   Former Licensed Counselor Sentenced For Defrau...  November 6, 2024
6   Macomb County Doctor And Pharmacist Agree To P...  November 4, 2024
7   Rocky Hill Pharmacy And Its Owners Indicted Fo...  November 4, 2024
8   North Texas Medical Center Pays $14.2 Million ...  November 4, 2024
9   New England Doctor Pleads Guilty To Drug Distr...  November 4, 2024
10  Attorney General Alan Wilson Announces Upstate...  November 4, 2024
11  St. Louis County Woman Accused Of $3 Million H...  November 1, 2024
12  Lab Owner And Marketing Company Owner Both Fou...  November 1, 2024
13  Compound Ingredient Supplier Medisca Inc., To ...  November 1, 2024
14  The New Mexico Department Of Justice Charges F...  November 1, 2024
15  Nashville Woman Indicted, Charged In TBI Medic...  November 1, 2024
16  Michael DePalma, MD and Virginia I-Spine Physi...  October 31, 2024
17  Columbus Doctor, His Clinic Convicted of $1.5 ...  October 31, 2024
18  Mercy Health Youngstown Agreed to Pay $69,000 ...  October 30, 2024
19  Quincy-Based Physician Group To Pay $650,000 T...  October 30, 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
5       Criminal and Civil Actions
6       Criminal and Civil Actions
7       Criminal and Civil Actions
8       Criminal and Civil Actions
9       Criminal and Civil Actions
10      State Enforcement Agencies
11      Criminal and Civil Actions
12      Criminal and Civil Actions
13      Criminal and Civil Actions
14      State Enforcement Agencies
15      State Enforcement Agencies
16  CMP and Affirmative Exclusions
17      State Enforcement Agencies
18          Fraud Self-Disclosures
```

```
19      State Enforcement Agencies

                                                 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-...
5   https://oig.hhs.gov/fraud/enforcement/former-l...
6   https://oig.hhs.gov/fraud/enforcement/macomb-c...
7   https://oig.hhs.gov/fraud/enforcement/rocky-hi...
8   https://oig.hhs.gov/fraud/enforcement/north-te...
9   https://oig.hhs.gov/fraud/enforcement/new-engl...
10  https://oig.hhs.gov/fraud/enforcement/attorney...
11  https://oig.hhs.gov/fraud/enforcement/st-louis...
12  https://oig.hhs.gov/fraud/enforcement/lab-owne...
13  https://oig.hhs.gov/fraud/enforcement/compound...
14  https://oig.hhs.gov/fraud/enforcement/the-new-...
15  https://oig.hhs.gov/fraud/enforcement/nashvill...
16  https://oig.hhs.gov/fraud/enforcement/michael-...
17  https://oig.hhs.gov/fraud/enforcement/columbus...
18  https://oig.hhs.gov/fraud/enforcement/mercy-he...
19  https://oig.hhs.gov/fraud/enforcement/quincy-b...
```

2. Crawling (PARTNER 1)

```python
def agency_names(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'lxml')

    agency_info = soup.find('span', text="Agency:")
    if agency_info:
        agency_name = agency_info.find_next_sibling(text=True)
        if agency_name:
            return agency_name.strip()
    return 'Not found'

agencies = []
for link in df['Link']:
    if link != 'No link':
        agency_name = agency_names(link)
        agencies.append(agency_name)
    else:
        agencies.append('No link provided')
```

```
df['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) Function: scrape_enforcement_actions(month, year)

Step 1: Validate input year Check whether start_year is less than 2013, if so, print a message indicating that only years >= 2013 are allowed and exit the function.

Step 2: Set up Define the base URL, start_date, current_date, and initialize an empty list to store actions. Start with page 1 and set keep_scraping to True.

Step 3: Loop through pages While keep_scraping is True, construct the URL for the current page and send a request. Parse the page content to find all enforcement action cards.

Step 4: Extract data For each card, get the date and check if it's before start_date. If so, stop scraping. Also extract title, link, category, and agency, then add these to the list.

Step 5: Complete Move to the next page and add a 1 second delay to avoid overloading the server When scraping is done, convert the list to a dataframe and save it to a csv file and print the total actions & earliest action date.

- b. Create Dynamic Scraper (PARTNER 2)

```
def scrape_enforcement_actions(start_month, start_year):

    if start_year < 2013:
        print("Please restrict to year >= 2013.")
        return

    base_url = 'https://oig.hhs.gov/fraud/enforcement/'
    start_date = datetime(start_year, start_month, 1)
    current_date = datetime.now()

    enforcement_actions = []
    page = 1
    keep_scraping = True

    while keep_scraping:
        url = f"{base_url}?page={page}"
        print(f"Scraping page {page}...")
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'lxml')

        cards = soup.find_all('li', class_='usa-card')

        for card in cards:
            date_time = card.find('span',
↪   class_='text-base-dark').text.strip()
            date = datetime.strptime(date_time, "%B %d, %Y") if date_time
↪   else None

            if date and date < start_date:
                keep_scraping = False
                break

            title = card.find('h2').text.strip() if card.find('h2') else 'No
↪   title'
            short_link = card.find('a')['href'] if card.find('a') else None
            link = f"https://oig.hhs.gov{short_link}" if short_link else 'No
↪   link'
            category = card.find('li',
↪   class_='display-inline-block').text.strip() if card.find('li',
↪   class_='display-inline-block') else 'No category'
```

```
            agency = agency_names(link) if link != 'No link' else 'No link
 ↪  provided'

            enforcement_actions.append({
                "Title": title,
                "Date": date_time,
                "Category": category,
                "Link": link,
                "Agency": agency
            })

        page += 1

# When I try time.sleep(1), it runs into TimeoutError.
# So I use 2 seconds so that it can have more time to process.
        time.sleep(2)

    df = pd.DataFrame(enforcement_actions, columns=['Title', 'Date',
 ↪  'Category', 'Link', 'Agency'])
    return df

df = scrape_enforcement_actions(1, 2023)
df.to_csv("enforcement_actions_2023_1.csv", index=False)
print(f"The total number of enforcement actions: {len(df)}")

earliest_action = df.iloc[-1]
earliest_date = earliest_action['Date']
print(f"The earliest date: {earliest_date}")
print(f"The earliest enforcement action scraped:\n{earliest_action}")
```

```
# load the enforcement_actions_2023_1.csv
filepath = "output/enforcement_actions_2023_1.csv"
df_2023 = pd.read_csv(filepath)

earliest_action = df_2023.iloc[-1]
earliest_date = earliest_action['Date']
print(f"The earliest date: {earliest_date}")
print(f"The earliest enforcement action scraped:\n{earliest_action}")
```

```
The earliest date: 3-Jan-23
The earliest enforcement action scraped:
Title       Podiatrist Pays $90,000 To Settle False Billin...
Date                                                3-Jan-23
Category                          Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/podiatri...
Agency      U.S. Attorney's Office, Southern District of T...
```

```
Name: 1533, dtype: object
```

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

```
titles = []
dates = []
types = []
links = []
agencies = []

def extract_data_from_page(soup, start_date):
    # Path for agency
    entries = soup.select('#results > div.grid-row.grid-gap >
↪  div.filter-result.grid-col-fill > div.grid-col-fill >
↪  ul.usa-card-group.padding-y-0 > li')

    for entry in entries:
        title = entry.find('h2').get_text(strip=True)
        date_text = entry.find('span').get_text(strip=True)

        # Convert date to datetime
        try:
            date = datetime.strptime(date_text, "%B %d, %Y")
        except ValueError:
            continue

        # Stop loop after meet start_date
        if date < start_date:
            return False

        type_ = [t.get_text(strip=True) for t in entry.find_all('li')]
        link = entry.find('a')['href']

        # Complete link
        full_link = base_url + link
        titles.append(title)
        dates.append(date_text)
        types.append(type_)
        links.append(full_link)

        # Find agency
        response_link = requests.get(full_link)
        soup_link = BeautifulSoup(response_link.text, 'html.parser')
        try:
            agency = soup_link.select_one('#main-content > div >
↪  div:nth-child(2) > article > div > ul >
↪  li:nth-child(2)').get_text(strip=True)
```

```
        except AttributeError:
            agency = None
        agencies.append(agency)

    return True

def scrape_enforcement_actions(year):
    # Check if the date is after 2013
    if year < 2013:
        print("Please input a year >= 2013, as only enforcement actions
        ↪  after 2013 are listed.")
        return

    # Set start_date
    start_date = datetime(year, 1, 1)

    # Loop through pages
    page_num = 1
    while True:
        response = requests.get(f"{enforcement_url}?page={page_num}")
        soup = BeautifulSoup(response.text, 'html.parser')

        # call function to extract data from pages
        continue_scraping = extract_data_from_page(soup, start_date)
        if not continue_scraping:
            break

        # Next page and 1 second rest
        page_num += 1
        time.sleep(1)

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

    # Define file path for .csv
    current_month = datetime.now().strftime("%Y_%m")
    output_dir = "output"
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    csv_filename = f"{output_dir}/enforcement_actions_{current_month}.csv"
```

```
    df.to_csv(csv_filename, index=False)

    return df

year = 2021
df = scrape_enforcement_actions(year)

non_agency = df[~df['Agency'].str.startswith("Agency:", na=False)]
non_agency_head = non_agency['Agency'].str[:11]
non_agency_head.unique()

# Remove the agency prefix
# fill in the remaining blank values
df['Agency'] = df['Agency'].str[7:]
df.loc[non_agency_head.index, 'Agency'] = np.nan

# Convert category list to string
df['Category'] = df['Category'].apply(lambda x: x[0] if isinstance(x, list)
 ↪  and len(x) > 0 else "")

# Counts rows and earliest date
num_rows = len(df)
print(f"Number of rows: {num_rows}")

df['Date'] = pd.to_datetime(df['Date'], format='%B %d, %Y',
 ↪  errors='coerce').dt.date
earliest_date = df['Date'].min()
print(f"Earliest date: {earliest_date}")

earliest_date_rows = df[df['Date'] == earliest_date]
earliest_date_array = earliest_date_rows.values
print("Rows with the earliest date:")
print(earliest_date_array)
```

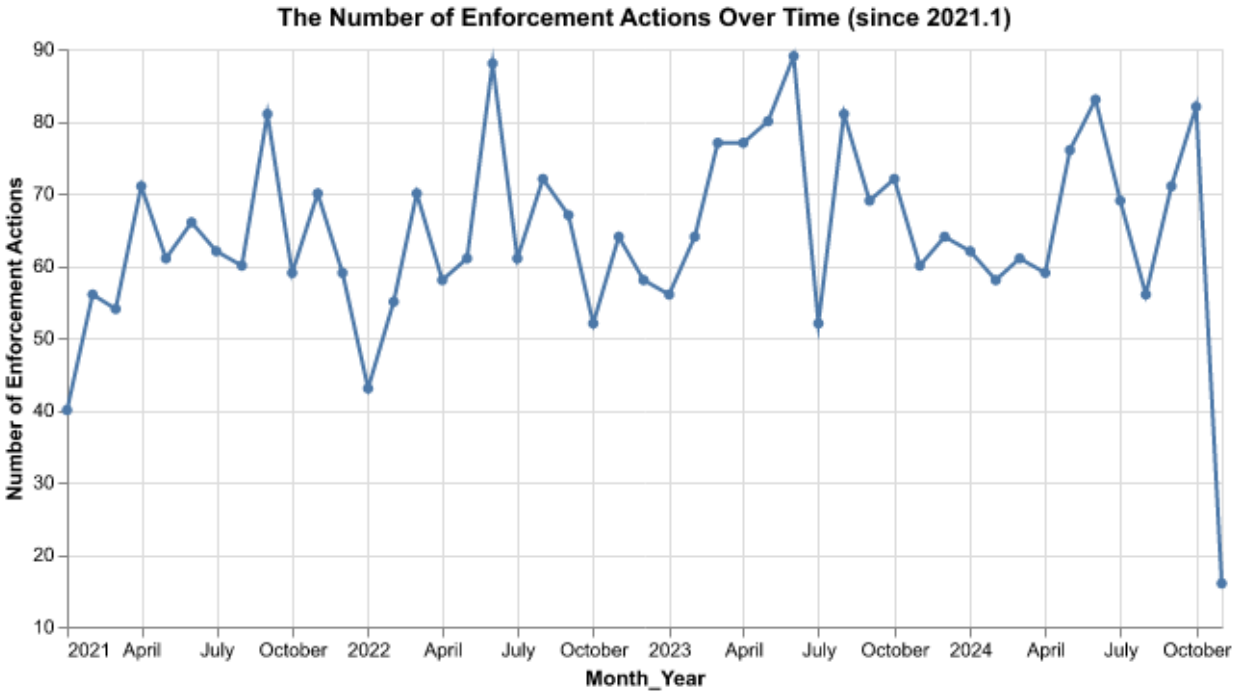
```
# load the enforcement_actions_2021_1.csv
filepath = "output/enforcement_actions_2021_1.csv"
df = pd.read_csv(filepath)

enforcement_action_count = len(df)

earliest_action = df.iloc[-1]
earliest_date = earliest_action['Date']
print(f"The total number of enforcement actions: {len(df)}")
print(f"The earliest date: {earliest_date}")
print(f"The earliest enforcement action scraped:\n{earliest_action}")
```

```
The total number of enforcement actions: 3022
The earliest date: January 4, 2021
The earliest enforcement action scraped:
Title       The United States And Tennessee Resolve Claims...
Date                                            January 4, 2021
Category                              Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/the-unit...
Agency      U.S. Attorney's Office, Middle District of Ten...
Name: 3021, dtype: object
```

Step 3: Plot data based on scraped data

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

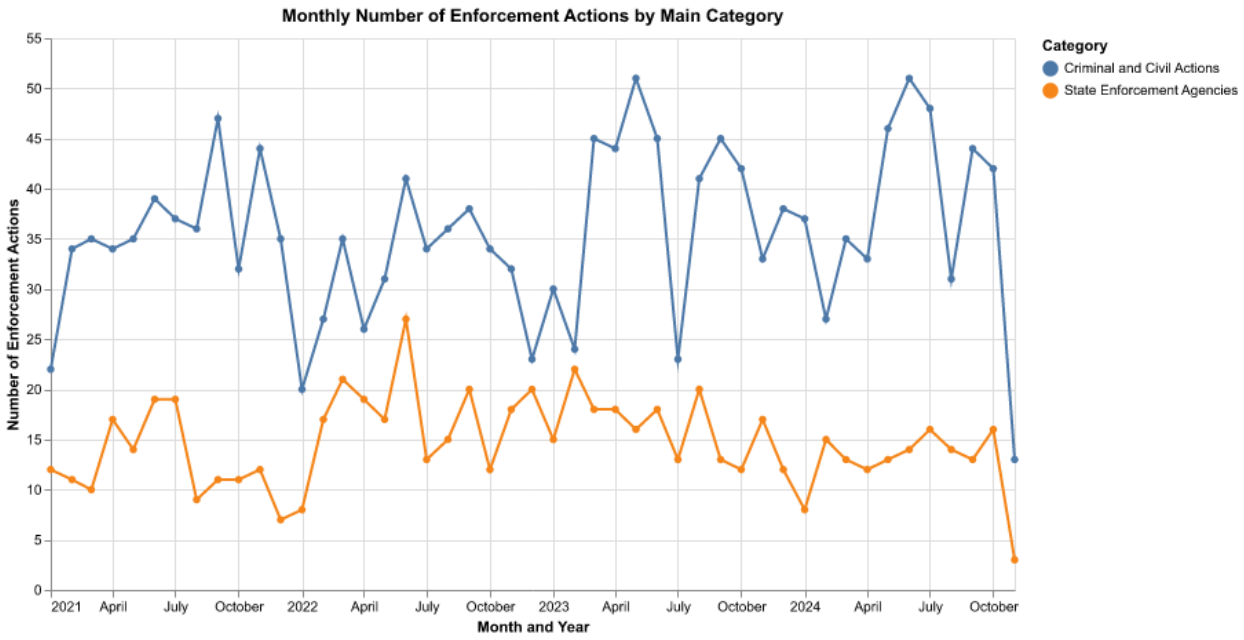
```
df['Date'] = pd.to_datetime(df['Date'], format='%B %d, %Y')

df['Month_Year'] = df['Date'].dt.to_period('M').dt.to_timestamp()

monthly_numbers = df.groupby('Month_Year').size().reset_index(name='Number')

alt.Chart(monthly_numbers).mark_line(point=True).encode(
    x=alt.X('Month_Year:T', title='Month_Year'),
    y=alt.Y('Number', title='Number of Enforcement Actions',
    ↪ scale=alt.Scale(zero=False)),
    tooltip=['Month_Year', 'Number']
).properties(
    title='The Number of Enforcement Actions Over Time (since 2021.1)',
    width=600,
    height=300
)
```



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

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

```
df_1 = df[df['Category'].isin(['Criminal and Civil Actions', 'State
 ↪  Enforcement Agencies'])]

numbers_category = df_1.groupby(['Month_Year',
 ↪  'Category']).size().reset_index(name='Number')

alt.Chart(numbers_category).mark_line(point=True).encode(
    x=alt.X('Month_Year:T', title='Month and Year'),
    y=alt.Y('Number:Q', title='Number of Enforcement Actions',
 ↪  scale=alt.Scale(zero=False)),
    color='Category:N',
    tooltip=['Month_Year', 'Category', 'Number']
).properties(
    title='Monthly Number of Enforcement Actions by Main Category',
    width=700,
    height=400
)
```

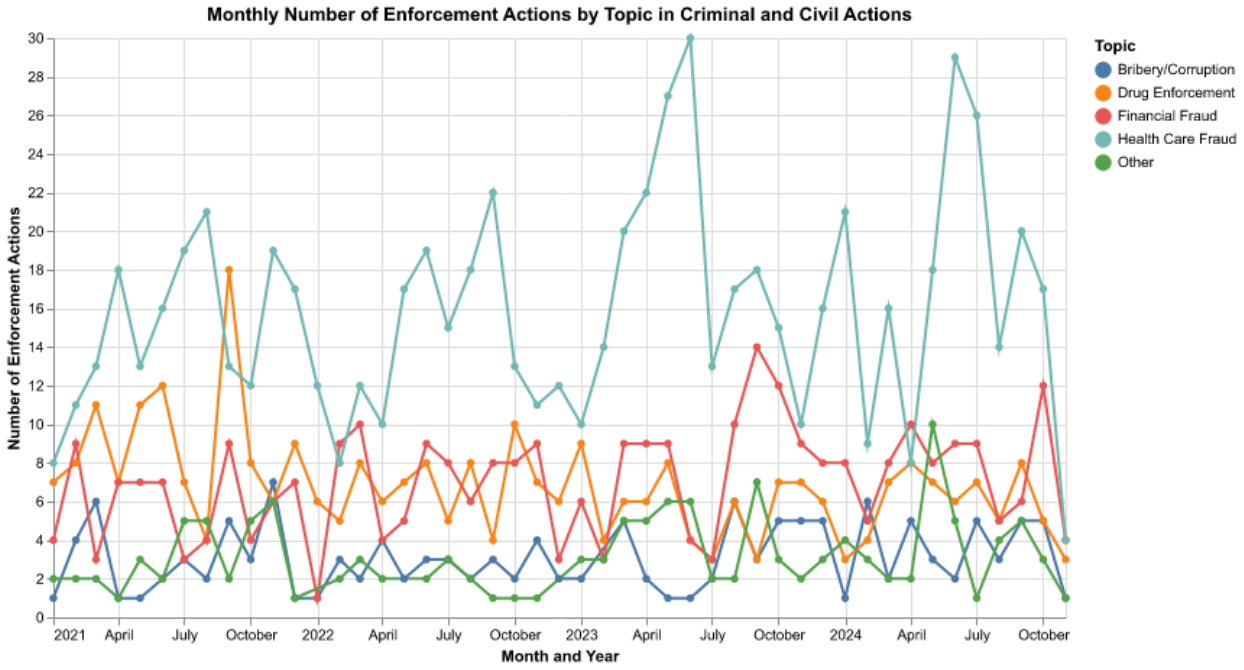


- based on five topics

```
df_2 = df[df['Category'] == 'Criminal and Civil Actions']

def categorize_topic(title):

    # Define search word roots
    health_keywords = ['health', 'medicare', 'medicaid', 'pharmacy', 'care',
    ↪ 'medical', 'doctor', 'billing', 'insurance', 'prescription', 'medical',
    ↪ 'therapist', 'psychotherapy', 'physician', 'false claims act',
    ↪ 'healthcare records', 'healthcare fraud', 'kickbacks', 'false claims',
    ↪ 'illegal kickbacks', 'health fraud']
    financial_keywords = ['embezzlement', 'kickback', 'scheme', 'theft',
    ↪ 'money', 'bank', 'invest', 'social security', 'finance', 'financial',
    ↪ 'tax evasion', 'false statements', 'financial fraud']
    drug_keywords = ['drug', 'opioid', 'oxy', 'pill mill', 'substance',
    ↪ 'distribution', 'controlled', 'morphine', 'meth', 'pill', 'mill',
    ↪ 'conspiracy', 'prescription fraud', 'opioid crisis', 'misuse']
    bribery_keywords = ['bribery', 'corruption', 'misconduct', 'kickback',
    ↪ 'payoff', 'bribe', 'illegal kickbacks', 'cover up', 'abuse of power']

    # Priority based classification, regardless of capitalization
    if any(re.search(rf'\b{word}\b', title, re.IGNORECASE) for word in
     ↪ drug_keywords):
        return "Drug Enforcement"
    elif any(re.search(rf'\b{word}\b', title, re.IGNORECASE) for word in
     ↪ bribery_keywords):
        return "Bribery/Corruption"
```

```
    elif any(re.search(rf'\b{word}\b', title, re.IGNORECASE) for word in
     ↪ financial_keywords):
        return "Financial Fraud"
    elif any(re.search(rf'\b{word}\b', title, re.IGNORECASE) for word in
     ↪ health_keywords):
        return "Health Care Fraud"
    else:
        return "Other"

df_2['Topic'] = df_2['Title'].apply(categorize_topic)

topic_counts = df_2['Topic'].value_counts()
print(topic_counts)

numbers_topic = df_2.groupby(['Month_Year',
 ↪ 'Topic']).size().reset_index(name='Number')

alt.Chart(numbers_topic).mark_line(point=True).encode(
    x=alt.X('Month_Year:T', title='Month and Year'),
    y=alt.Y('Number:Q', title='Number of Enforcement Actions',
 ↪ scale=alt.Scale(zero=False)),
    color='Topic:N',
    tooltip=['Month_Year', 'Topic', 'Number']
).properties(
    title='Monthly Number of Enforcement Actions by Topic in Criminal and
 ↪ Civil Actions',
    width=700,
    height=400
)
```

```
Topic
Health Care Fraud     743
Financial Fraud       330
Drug Enforcement      320
Bribery/Corruption    142
Other                 140
Name: count, dtype: int64
```

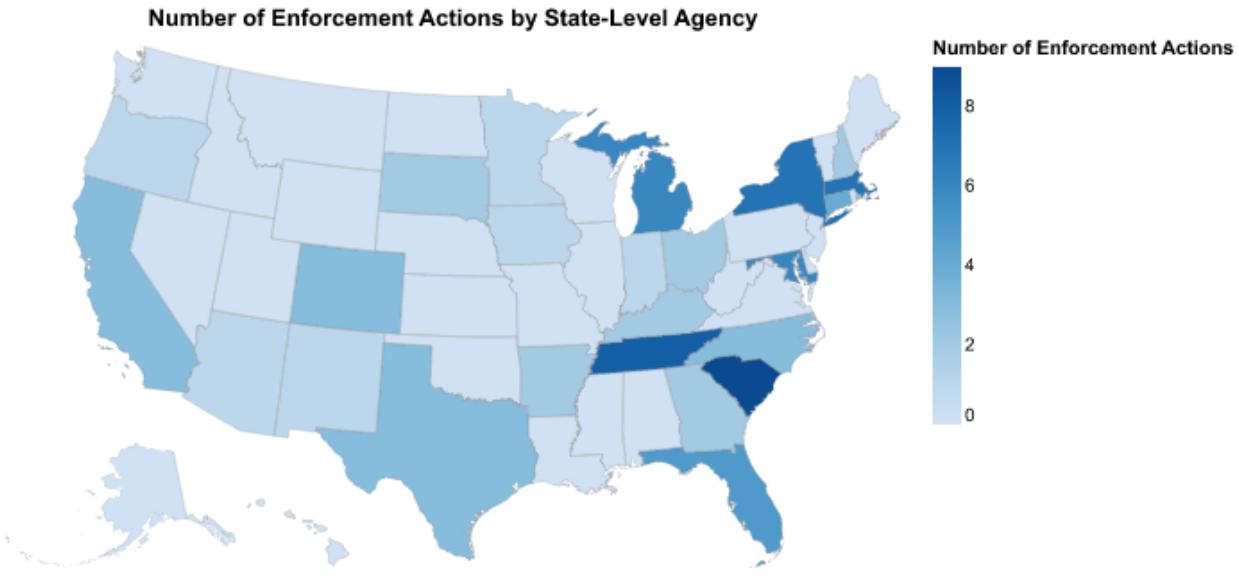


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
df_state =
 ↪  gpd.read_file('data/cb_2018_us_state_500k/cb_2018_us_state_500k.shp')

df_cleaned = df.dropna(subset=['Agency'])
filtered_df = df_cleaned[df_cleaned['Agency'].str.contains('State of ',
 ↪  case=False)]
filtered_df['State'] = filtered_df['Agency'].str.replace('State of ', '',
 ↪  regex=False)

# Count rows for each state
state_enforcement_counts =
 ↪  filtered_df.groupby('State').size().reset_index(name='enforcement_count')

# Merge state info with enforcement counts
state_shape = df_state.merge(state_enforcement_counts, left_on='NAME',
 ↪  right_on='State', how='left')
state_shape['enforcement_count'] =
 ↪  state_shape['enforcement_count'].fillna(0)

merged_state_json = json.loads(state_shape.to_json())
geojson_data = alt.Data(values=merged_state_json['features'])
```

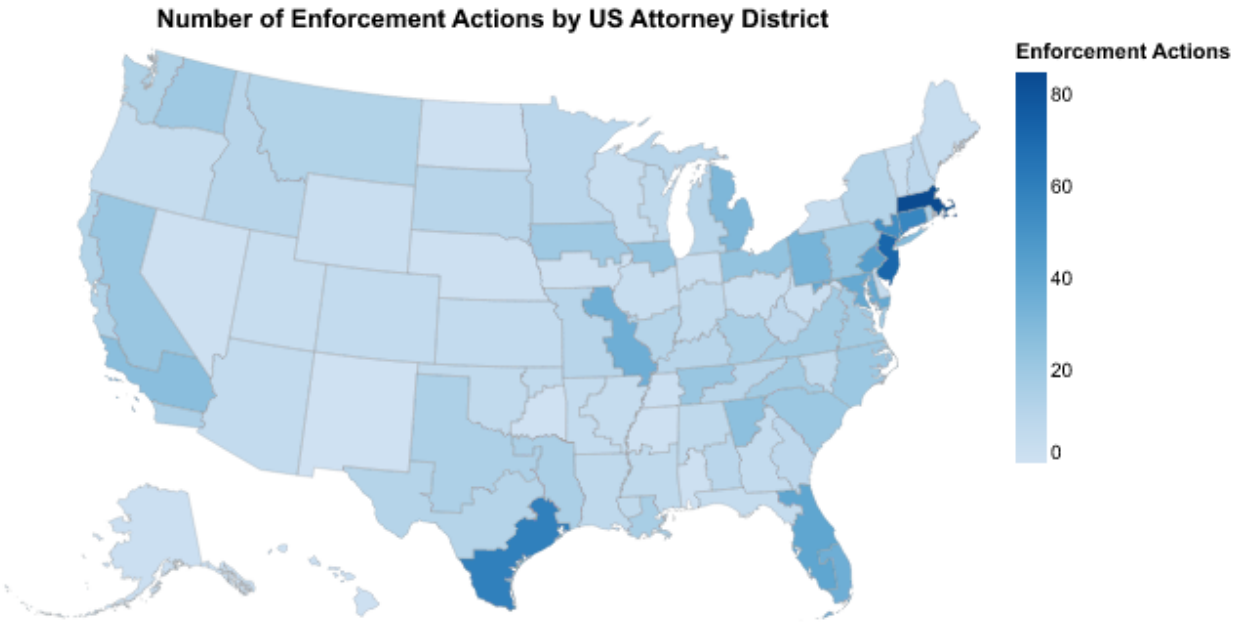
```
choropleth_inner = alt.Chart(geojson_data).mark_geoshape(
    stroke=None
).encode(
    color=alt.Color('properties.enforcement_count:Q',
                    scale=alt.Scale(scheme='blues'),
                    title="Number of Enforcement Actions")
).project(
    type='albersUsa'
).properties(
    width=500,
    height=300,
    title='Number of Enforcement Actions by State-Level Agency'
)

choropleth_outline = alt.Chart(geojson_data).mark_geoshape(
    fillOpacity=0,
    stroke='gray',
    strokeWidth=0.2
).project(
    type='albersUsa'
)

choropleth = choropleth_inner + choropleth_outline
choropleth.display()
```



2. Map by District (PARTNER 2)

```
df_district = gpd.read_file('data/US Attorney Districts Shapefile
 ↪  simplified_20241109/geo_export_2f7c0256-d6f4-4537-956a-931cb7e3f87e.shp')

filtered_df_district =
 ↪  df_cleaned[df_cleaned['Agency'].str.contains('District of ',
 ↪  case=False)]
filtered_df_district['District'] =
 ↪  filtered_df_district['Agency'].str.rsplit(',', n=1).str[1].str.strip()

def missing_district(agency):
    if "U.S. Attorney" in agency:
        parts = agency.split()
        if len(parts) > 3:
            district = ' '.join(parts[3:]).strip()
        else:
            district = agency.strip()
        return district
    else:
        return agency.strip()

filtered_df_district['District'] = filtered_df_district.apply(
    lambda row: missing_district(row['Agency']) if pd.isna(row['District'])
     ↪  else row['District'], axis=1)

filtered_df_district['District'] =
 ↪  filtered_df_district['District'].str.replace('†', '',
 ↪  regex=False).str.strip()
filtered_df_district['District'] =
 ↪  filtered_df_district['District'].str.replace(r'††', '', regex=True)
district_counts =
 ↪  filtered_df_district.groupby('District').size().reset_index(name='Enforcement
 ↪  Actions')

merged_district = df_district.merge(district_counts, left_on='judicial_d',
 ↪  right_on='District', how='left')
merged_district['Enforcement Actions'] = merged_district['Enforcement
 ↪  Actions'].fillna(0)

merged_district_json = json.loads(merged_district.to_json())
geojson_data = alt.Data(values=merged_district_json['features'])

choropleth_inner = alt.Chart(geojson_data).mark_geoshape(
    stroke=None
).encode(
```

```
    color=alt.Color('properties.Enforcement Actions:Q',
                    scale=alt.Scale(scheme='blues'),
                    title="Enforcement Actions")
).project(
    type='albersUsa'
).properties(
    width=500,
    height=300,
    title='Number of Enforcement Actions by US Attorney District'
)

choropleth_outline = alt.Chart(geojson_data).mark_geoshape(
    fillOpacity=0,
    stroke='gray',
    strokeWidth=0.2
).project(
    type='albersUsa'
)

choropleth = choropleth_inner + choropleth_outline
choropleth.display()
```



Extra Credit

1. Merge zip code shapefile with population

```
df_pop = pd.read_csv('data/DECENNIALDHC2020/DECENNIALDHC2020.P1-Data.csv')
df_zip =
 ↪  gpd.read_file('/Users/Betsy/Documents/GitHub/problem-set-4-joy-betsy/data/gz_2010_us_860

df_zip['ZCTA5'] = df_zip['ZCTA5'].astype(str)
df_pop['NAME'] = df_pop['NAME'].astype(str)

df_pop['ZIP_CODE'] = df_pop['NAME'].str.replace('ZCTA5 ', '')

merged_df = df_zip.merge(df_pop[['ZIP_CODE', 'P1_001N']], left_on='ZCTA5',
 ↪  right_on='ZIP_CODE', how='left')

print(merged_df.head())
```

```
          GEO_ID  ZCTA5   NAME   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

                                            geometry ZIP_CODE P1_001N
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...    01040   38238
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...    01050    2467
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...    01053    2031
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...    01056   21002
4  MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...    01057    8152
```

2. Conduct spatial join

```
print("merged_df CRS:", merged_df.crs)
print("df_district CRS:", df_district.crs)

merged_df = merged_df.to_crs(epsg=4326)

df_district = df_district.rename(columns={'the_geom': 'geometry'})
df_district = df_district.set_geometry('geometry')
print(df_district.head())
```

```
zip_district = gpd.sjoin(merged_df, df_district, how="inner",
 ↪ predicate='intersects')
zip_district['P1_001N'] = pd.to_numeric(zip_district['P1_001N'],
 ↪ errors='coerce')
print(zip_district.head())

district_pop =
 ↪ zip_district.groupby('judicial_d')['P1_001N'].sum().reset_index()
print(district_pop)
```

```
merged_df CRS: EPSG:4269
df_district CRS: EPSG:4326
```

```
  statefp                    judicial_d         aland        awater     state  \
0      21  Western District of Kentucky  4.970555e+10  1.651516e+09  Kentucky
1      21  Eastern District of Kentucky  5.257394e+10  7.238213e+08  Kentucky
2      18  Southern District of Indiana  5.824517e+10  5.941176e+08   Indiana
3      01    Middle District of Alabama  3.412673e+10  5.472423e+08   Alabama
4      01  Southern District of Alabama  6.235882e+10  3.052681e+09   Alabama

            chief_judg          nominating  term_as_ch  shape_leng  \
0      Greg N. Stivers    Barack Obama (D)      2018.0   16.200585
1         Danny Reeves  George W. Bush (R)      2019.0   13.514251
2  Jane Magnus-Stinson    Barack Obama (D)      2016.0   14.956126
3    Emily Coody Marks    Donald Trump (R)      2019.0   10.235799
4        Kristi DuBose  George W. Bush (R)      2017.0   12.976906

   shape_area abbr district_n    shape__are    shape__len  \
0    5.216899  KYW          6  8.123902e+10  1.964255e+06
1    5.451047  KYE          6  8.547129e+10  1.654681e+06
2    6.137433  INS          7  9.818187e+10  1.887626e+06
3    3.858442  ALM         11  5.645450e+10  1.236201e+06
4    3.278871  ALS         11  4.772733e+10  1.567095e+06

                                          geometry
0  MULTIPOLYGON (((-89.48248 36.50214, -89.48543 ...
1  POLYGON ((-84.62012 39.07346, -84.60793 39.073...
2  POLYGON ((-85.86281 40.46476, -85.86212 40.406...
3  POLYGON ((-85.33828 33.49471, -85.33396 33.492...
4  MULTIPOLYGON (((-88.08682 30.25987, -88.07676 ...

          GEO_ID  ZCTA5   NAME   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
```

```
                                              geometry ZIP_CODE  P1_001N  \
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...    01040  38238.0
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...    01050   2467.0
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...    01053   2031.0
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...    01056  21002.0
4  MULTIPOLYGON (((-72.39191 42.08065, -72.39077 ...    01057   8152.0

   index_right statefp  ...          state        chief_judg  \
0           50      25  ...  Massachusetts     Dennis Saylor
1           50      25  ...  Massachusetts     Dennis Saylor
2           50      25  ...  Massachusetts     Dennis Saylor
3           50      25  ...  Massachusetts     Dennis Saylor
4           79      09  ...     Connecticut  Stefan Underhill

           nominating term_as_ch shape_leng shape_area  abbr  district_n  \
0  George W. Bush (R)     2019.0  17.629736   2.318549    MA           1
1  George W. Bush (R)     2019.0  17.629736   2.318549    MA           1
2  George W. Bush (R)     2019.0  17.629736   2.318549    MA           1
3  George W. Bush (R)     2019.0  17.629736   2.318549    MA           1
4    Bill Clinton (D)     2018.0   6.706574   1.396680    CT           2

     shape__are    shape__len
0  3.881867e+10  2.199726e+06
1  3.881867e+10  2.199726e+06
2  3.881867e+10  2.199726e+06
3  3.881867e+10  2.199726e+06
4  2.315274e+10  8.502177e+05

[5 rows x 23 columns]
                        judicial_d     P1_001N
0   Central District of California  19621862.0
1     Central District of Illinois   2684528.0
2               District of Alaska    707199.0
3              District of Arizona   7334666.0
4             District of Colorado   5935657.0
..                             ...         ...
86   Western District of Tennessee   1895710.0
87       Western District of Texas   8203041.0
88    Western District of Virginia   3067463.0
89  Western District of Washington   6289276.0
90   Western District of Wisconsin   2989929.0

[91 rows x 2 columns]
```

3. Map the action ratio in each district

```
district_per_pop = merged_district.merge(district_pop[['judicial_d',
 ↪  'P1_001N']], on='judicial_d', how='left')

district_per_pop['Enforcement Actions'] = district_per_pop['Enforcement
 ↪  Actions'].fillna(0)
district_per_pop['P1_001N'] = district_per_pop['P1_001N'].fillna(0)

district_per_pop['Enforcement_Ratio'] = district_per_pop['Enforcement
 ↪  Actions'] / district_per_pop['P1_001N']
print(district_per_pop[['Enforcement_Ratio']])

# Scale the Enforcement_Ratio by multiplying it by 100,000
# to increase the values and make them easier to interpret
district_per_pop['Enforcement_Ratio_Scaled'] =
 ↪  district_per_pop['Enforcement_Ratio'] * 100000

district_per_pop_json = json.loads(district_per_pop.to_json())
geojson_data = alt.Data(values=district_per_pop_json['features'])

choropleth_inner = alt.Chart(geojson_data).mark_geoshape(
    stroke=None
).encode(
    color=alt.Color('properties.Enforcement_Ratio_Scaled:Q',
                    scale=alt.Scale(scheme='blues'),
                    title="Enforcement Actions per 100,000 People")
).project(
    type='albersUsa'
).properties(
    width=500,
    height=300,
    title='Scaled Ratio of Enforcement Actions per Population in Each US
 ↪  Attorney District'
)

choropleth_outline = alt.Chart(geojson_data).mark_geoshape(
    fillOpacity=0,
    stroke='gray',
    strokeWidth=0.2
).project(
    type='albersUsa'
)

choropleth = choropleth_inner + choropleth_outline
choropleth.display()
```

```
    Enforcement_Ratio
0        3.549760e-06
1        6.192296e-06
2        1.442539e-06
3        6.497017e-06
4        8.160554e-07
..                ...
89                NaN
90       5.047803e-06
91       5.016680e-06
92       5.118120e-06
93       4.397015e-06

[94 rows x 1 columns]
```