PS5 Andy Fan Will Sigal

PS4: Due Sat Nov 9 at 5:00PM Central. Worth 100 points.

Style Points (10 pts)

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): Andy Fan, fanx
    - Partner 2 (name and cnet ID): Will Sigal, Wsigal
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:

AF WS

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: (Andy:0); (Will:0) Late coins left after submission: (Andy: 3) ; (Will: 4)
7. Knit your `ps5.qmd` to an PDF file to make `ps5.pdf`,
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
8. (Partner 1): push `ps5.qmd` and `ps5.pdf` to your github repo.
9. (Partner 1): submit `ps5.pdf` via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope

```
### SETUP
import pandas as pd
import altair as alt
import time
import os
import warnings
import geopandas as gpd
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
import requests
from bs4 import BeautifulSoup
import concurrent.futures
```

(30 points) Step 1: Develop initial scraper and crawler

1. (Partner 1) Scraping: Go to the first page of the HHS OIG's "Enforcement Actions"page and scrape and collect the following into a dataset: ▪ Title of the enforcement action ▪ Date ▪ Category (e.g, "Criminal and Civil Actions") ▪ Link associated with the enforcement action Collect your output into a tidy dataframe and print its head.

```
### making soup
url1 = 'https://oig.hhs.gov/fraud/enforcement'
response1 = requests.get(url1)
soup1 = BeautifulSoup(response1.content, 'lxml')
```

```
### find title of enforcements
li_blocks = soup1.find_all('h2') # h2 classes with nested 'a' titles.
 ↪  li_blocks[2:21] are the 20 datapoints
li_titles = []
for h2 in li_blocks:
    for a_tag in h2.find_all('a'):
        li_titles.append(a_tag)
li_titles[0:5]
df_title = pd.DataFrame(li_titles) # dataframe with titles
df_title.columns = ['title']
```

Title: each title is a 'a' class, under h2 class. 'h2 class' under 'header class' under 'div class' under 'li class'

```
### find date
span_blocks = soup1.find_all('span', attrs={'class': 'text-base-dark
 ↪  padding-right-105'})
span_blocks[0:5]
df_date = pd.DataFrame(span_blocks)
df_date.columns = ['date']
#asked chat gpt 'how to i search for 'span' class with attribute xxx'
```

Date: is under 'span' class

```
### find category
li_blocks_dt = soup1.find_all('li', attrs={'class': 'display-inline-block
 ↪  usa-tag text-no-lowercase text-base-darkest bg-base-lightest
 ↪  margin-right-1'})
li_blocks[0:5]
df_category = pd.DataFrame(li_blocks_dt)
df_category.columns = ['category']
```

Category: each category is a li class (search by attribute).'l1 class' under 'ul class' under 'div class' under 'header class'

```
### find link
#use the list of titles from p1 and extract href
link_blocks = [link.get('href') for link in li_titles]
df_link = pd.DataFrame(link_blocks)
df_link.columns = ['link']
df_link['link'] = "https://oig.hhs.gov" + df_link['link'] #add front part of
 ↪  link to make it clickable
```

Link: link=href class, under h2 class. 'h2 class' under 'header class' under 'div class' under 'li class'

```
### combine dataframes
df_Q1 = pd.concat([df_title, df_date, df_category, df_link], axis=1)
print(df_Q1.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. (Partner 1) Crawling: Then for each enforcement action, click the link and collect the name of the agency involved (e.g., for this link, it would be U.S. Attorney's Office, Eastern District of Washington).

```
### webcrawl
urls_1_2 = df_Q1['link']

### find agency by finding 'li' tags nested in 'article' tags
article_tags = []
for url in urls_1_2:
    response = requests.get(url)
    if response.status_code == 200:  # Check if the request was successful
        soup = BeautifulSoup(response.text, 'html.parser')

        # Find all <articel> tags
        #article_tags = soup2.find_all('article')

        # Find all <li> tags within <article> tags
        articles = soup.find_all('article')
        for article in articles:
            li_tags = article.find_all('li')
            for li in li_tags:
                article_tags.append(li.get_text(strip=True))  # Append the
 ↪  text of each <li> tag
    else:
```

```
        print(f"Failed to retrieve {link}")

### make dataframe and remove non-relevant items from list
df_agency = pd.DataFrame(article_tags)
df_agency.columns = ['agency']
df_agency = df_agency[df_agency['agency'].str.contains('Agency', case=False,
 ↪  na=False)]

print(df_agency.head())
#asked chat gpt 'how to go to every link in a list and extracts all 'li' tags
 ↪  that are nested in 'article' tag and create them into a new list'
```

```
                                              agency
1                    Agency:U.S. Department of Justice
5   Agency:November 7, 2024; U.S. Attorney's Offic...
9   Agency:U.S. Attorney's Office, District of Mas...
13  Agency:U.S. Attorney's Office, Eastern Distric...
17  Agency:U.S. Attorney's Office, Middle District...
```

Name of agency is 'li' class, nested in 'ul' nested in 'div' nested in 'article'

*Some links do not have the corresponding agency listed, and a couple (such as "U.S. Attorney's Office, District of Idaho" are not formatted correctly)

(30 points) Step 2: Making the scraper dynamic

1. Turning the scraper into a function: You will write a function that takes as input a month and a year, and then pulls and formats the enforcement actions like in Step 1 starting from that month+year to today.

a (Partner 2) Before writing out your function, write down pseudo-code of the steps that your function will go through. If you use a loop, discuss what kind of loop you will use and how you will define it.

```
#1) Check if the year if the year is 2013 or later if not, terminte function
#2) Create dictionary for params and use a while loop for pagnation setting
 ↪  for it to break when there's no next.
#3) Send Get request to get pages, dates, categories and links
#4) append new info to our dictionary
#5) wait one sec before going to the next page
```

b (Partner 2) Now code up your dynamic scraper and run it to start collecting the enforcement actions since January 2023. How many enforcement actions do you get in your final dataframe? What is the date and details of the earliest enforcement action it scraped?

```
import time
from datetime import datetime

def enforcement_scrapper(start_month, start_year):
    if start_year < 2013:
        print("Please input a year >= 2013, as enforcement actions before
         ↪  2013 are unavailable.")
        return

    base_url = 'https://oig.hhs.gov/fraud/enforcement'
    current_page = 1
    results = []

    while True:
        url = f"{base_url}/?page={current_page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'lxml')
#titles links
        titles = []
        links = []
        for h2 in soup.find_all('h2'):
            a_tag = h2.find('a')
            if a_tag:
                titles.append(a_tag.get_text(strip=True))
                links.append("https://oig.hhs.gov" + a_tag['href'])

        #get dates
        dates = [span.get_text(strip=True) for span in soup.find_all('span',
 ↪  class_='text-base-dark padding-right-105')]

        #get categories
        categories = [
            li.get_text(strip=True) for li in soup.find_all('li',
 ↪  class_='display-inline-block usa-tag text-no-lowercase text-base-darkest
 ↪  bg-base-lightest margin-right-1')
        ]
```

```
        for title, date, category, link in zip(titles, dates, categories,
         ↪ links):
            results.append({'Title': title, 'Date': date, 'Category':
↪  category, 'Link': link})

        #check if page has dates earlier than when we're looking
        date_check = pd.to_datetime(dates, errors='coerce')
        if date_check.min() < pd.Timestamp(datetime(start_year, start_month,
         ↪ 1)):
            break

        current_page += 1



        time.sleep(1)


    scrapper = pd.DataFrame(results)
    scrapper['Date'] = pd.to_datetime(scrapper['Date'])
    start_date = datetime(start_year, start_month, 1)
    scrapper = scrapper[scrapper['Date'] >= start_date]
    return scrapper
```

```
twenty_three = enforcement_scrapper(1, 2023)

print(twenty_three.shape[0])
```

```
1534
```

```
earliest_date = twenty_three['Date'].min()
print(f"Earliest Date: {earliest_date.date()}")

num_enforcements = len(twenty_three)
print(f"Number of Enforcement Actions: {num_enforcements}")
```

```
Earliest Date: 2023-01-03
Number of Enforcement Actions: 1534
```

c (Partner 1) Now, let's go a little further back. Test your partner's code by collecting the actions since January 2021. Note that this can take a while. How many enforcement actions do you get in your final dataframe? What is the date and details of the earliest enforcement action it scraped? Use the dataframe from this process for every question after this

```
twenty_one = enforcement_scrapper(1, 2021)
num_enforcements = len(twenty_one)
```

```
#my code was taking me far far too long and so I used chat gpt to find how I
 ↪  can increase the speed of my crawler
import concurrent.futures
def scrape_link(link):
    try:
        response = requests.get(link)
        detail_soup = BeautifulSoup(response.content, 'lxml')
        agency_label = detail_soup.find('span', class_='padding-right-2
 ↪  text-base', text='Agency:')
        if agency_label:
            # Get the actual text that follows the Agency: label
            agency_text = agency_label.next_sibling
            if agency_text:
                return agency_text.strip()
        return None
    except Exception as e:
        print(f"Error scraping {link}: {e}")
        return None

def main():
    with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
        futures = {executor.submit(scrape_link, link): link for link in
 ↪  twenty_one['Link']}
        agencies = [future.result() for future in futures]

    twenty_one['Agency'] = agencies
    return twenty_one

results = main()
```

```
print(f"Number of Enforcement Actions: {num_enforcements}")
```

```
earliest_action = twenty_one.loc[twenty_one['Date'].idxmin()]
earliest_date = earliest_action['Date']
print(f"Earliest Date: {earliest_date.date()}")
print("Details of the Earliest Enforcement Action:")
print(earliest_action)
```

```
Number of Enforcement Actions: 3022
Earliest Date: 2021-01-04
Details of the Earliest Enforcement Action:
Title       The United States And Tennessee Resolve Claims...
Date                                      2021-01-04 00:00:00
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
```

(15 points) Step 3: Plot data based on scraped data (using altair)

1. (Partner 2) Plot a line chart that shows: the number of enforcement actions over time (aggregated to each month+year) overall since January 2021,

```
import altair as alt

twenty_one['Month_Year'] =
 ↪  twenty_one['Date'].dt.to_period('M').dt.to_timestamp()
monthly_counts =
 ↪  twenty_one.groupby('Month_Year').size().reset_index(name='Count')

line_agg = alt.Chart(monthly_counts).mark_line(point = True).encode(
  x = alt.X('Month_Year:T', title = 'Month/Year'),
  y = alt.Y('Count:Q', title = 'Number of Actions'),
  tooltip=['Month_Year:T', 'Count:Q']
).properties( title = 'Number of Enforcement Actions Over Time (Aggregated by
 ↪  Month-Year)')

line_agg.save("line_agg.png")
```

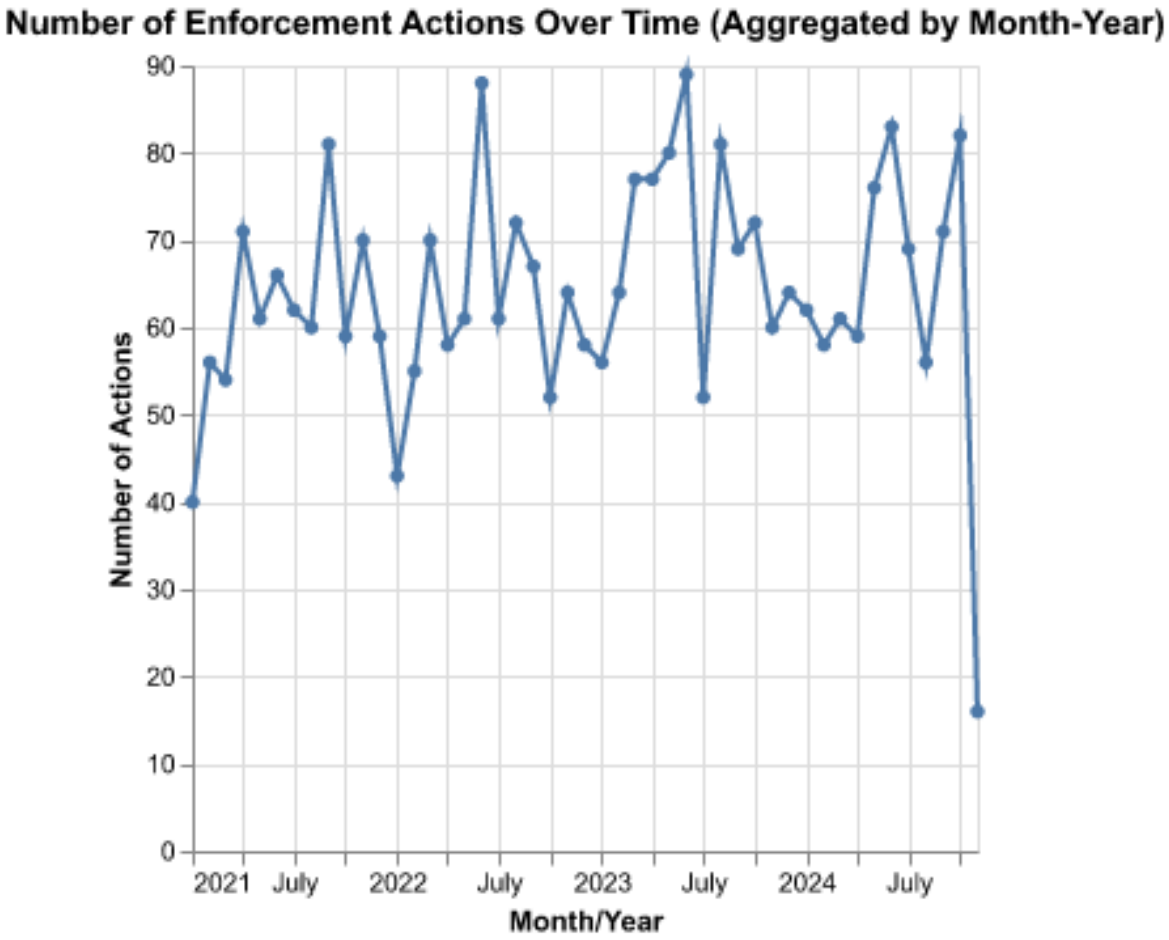


Figure 1: line_agg

2. (Partner 1) Plot a line chart that shows: the number of enforcement actions split out by: ▪ "Criminal and Civil Actions" vs. "State Enforcement Agencies" ▪ Five topics in the "Criminal and Civil Actions" category: "Health Care Fraud", "Financial Fraud", "Drug Enforcement", "Bribery/Corruption", and "Other".

```
### subset for criminal vs state
twenty_one1 = twenty_one[twenty_one['Category'].isin(['State Enforcement
 ↪  Agencies', 'Criminal and Civil Actions'])]
dfQ2_2_1 = twenty_one1.groupby(['Month_Year',
 ↪  'Category']).size().reset_index(name='Count')

chartQ2_2_1 = alt.Chart(dfQ2_2_1).mark_line(point = True).encode(
```

```
  x = alt.X('Month_Year:T', title = 'Month/Year'),
  y = alt.Y('Count:Q', title = 'Number of Actions'),
  color='Category:N',
  tooltip=['Month_Year:T', 'Count:Q']
).properties( title = 'Number of Enforcement Actions By Criminal and Civil,
 ↪  VS State Enforcement')

chartQ2_2_1.save("chartQ2_2_1.png")
```

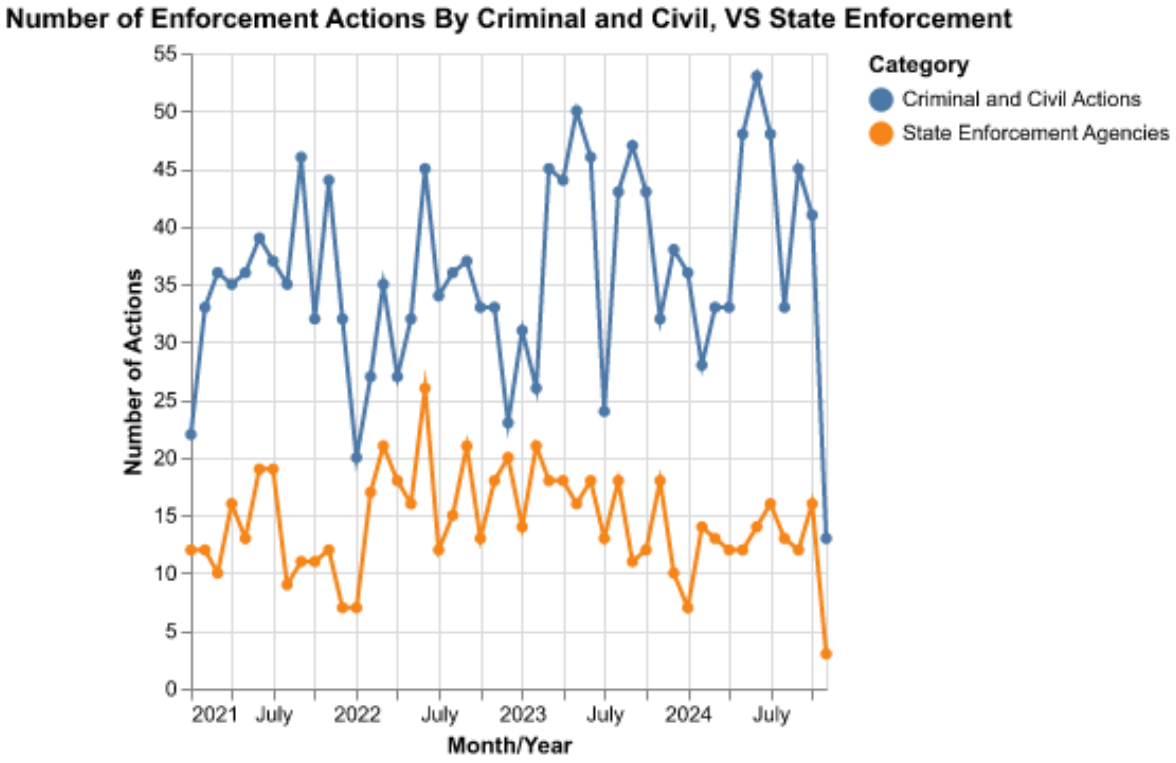


Figure 2: chartQ2_2_1

```
### subset and categrozie for types of fraud (topics)
twenty_one2 = twenty_one.copy()
twenty_one2['Topic']= None

### fucntcion filter
def categorize_topic(text):
    if 'Health' in text or 'Medicare' in text or 'Medicaid' in text:
        return 'Health Care Fraud'
```

```
    elif 'Finance' in text or 'Financial' in text or 'Monetary' in text:
        return 'Financial Fraud'
    elif 'Drug' in text or 'Medicine' in text or 'Prescribe' in text or
     ↪  'Prescribing' in text:
        return 'Drug Enforcement'
    elif 'Bribe' in text or 'Bribery' in text or 'Corruption' in text:
        return 'Bribery/Corruption'
    else:
        return 'Other'

twenty_one2['Topic'] = twenty_one2['Title'].apply(categorize_topic)
```

```
### plot
dfQ2_2_2 = twenty_one2.groupby(['Month_Year',
 ↪  'Topic']).size().reset_index(name='Count')

chartQ2_2_2 = alt.Chart(dfQ2_2_2).mark_line(point = True).encode(
  x = alt.X('Month_Year:T', title = 'Month/Year'),
  y = alt.Y('Count:Q', title = 'Number of Actions'),
  color='Topic:N',
  tooltip=['Month_Year:T', 'Count:Q']
).properties( title = 'Number of Enforcement Actions By Topic of Fraud')

chartQ2_2_2.save("chartQ2_2_2.png")
```

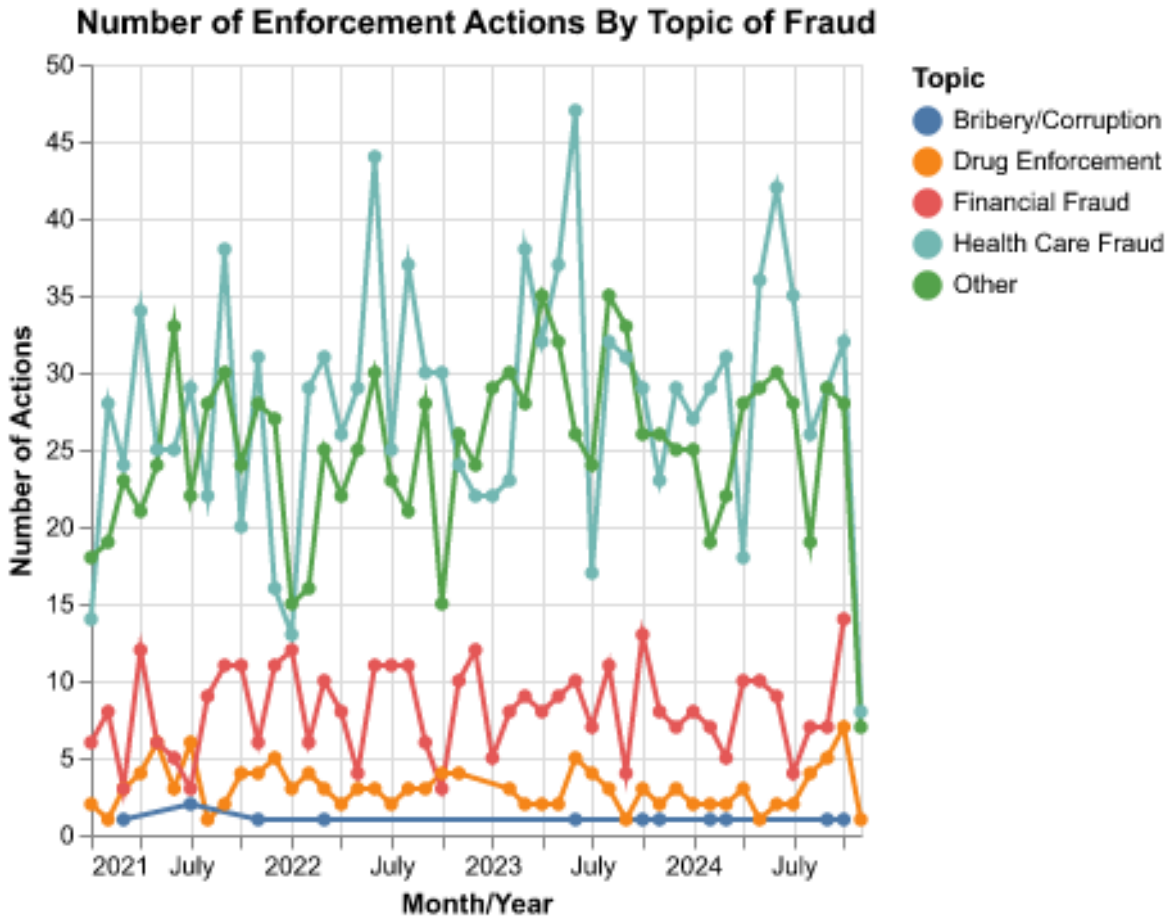


Figure 3: chartQ2_2_2

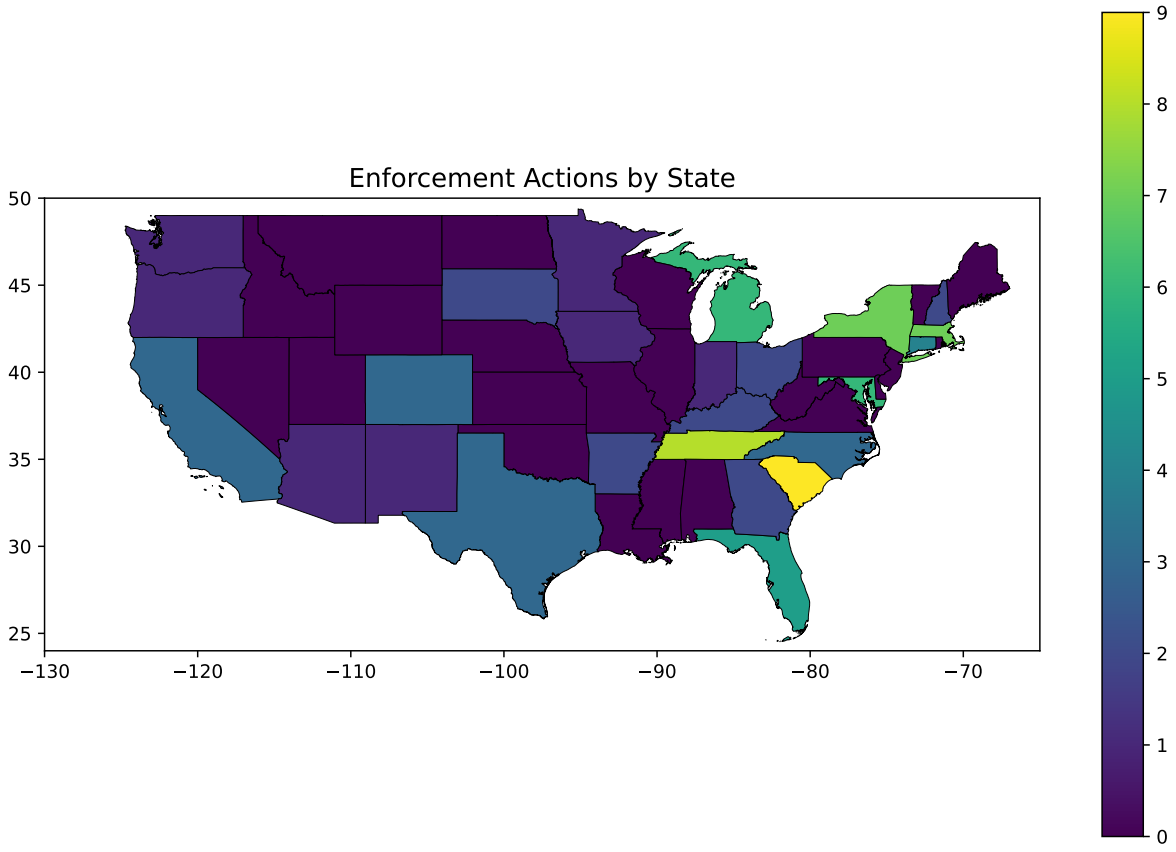
(15 points) Step 4: Create maps of enforcement activity For these questions, use this US Attorney District shapefile (link) and a Census state shapefile (link)

1. (Partner 1) Map by state: Among actions taken by state-level agencies, clean the state names you collected and plot a choropleth of the number of enforcement actions for each state. Hint: look for "State of" in the agency info!

```
#os.chdir('/Users/willsigal/Desktop') #will wd
os.chdir('d:\\UChicago\\Classes\\2024Qfall\\Programming
 ↪  Python\\problem-set-5\\Shapefiles') #andy wd

state = gpd.read_file('cb_2018_us_state_500k/cb_2018_us_state_500k.shp')
state_geometries = state['geometry']
```

```
### df of state gdf
state_df = pd.DataFrame(state)

### subset scraped df to state agency only
twenty_one_Q4_1 = twenty_one[twenty_one['Agency'].str.contains('State',
 ↪  na=False)].copy()

### subset to only agencies with state in their names via state_df
def find_state(title):
    for state in state_df['NAME']:
        if state in title:
            return state
    return None

twenty_one_Q4_1['State'] = twenty_one_Q4_1['Agency'].apply(find_state)

### groupby
twenty_one_Q4_1_grouped =
 ↪  twenty_one_Q4_1.groupby('State').size().reset_index(name='Count')
twenty_one_Q4_1_grouped = twenty_one_Q4_1_grouped.rename(columns={'State':
 ↪  'NAME'})

### merge into state gpd
state_merged = state.merge(twenty_one_Q4_1_grouped, on='NAME', how='left')
state_merged['Count'] = state_merged['Count'].fillna(0)
state_merged_test = pd.DataFrame(state_merged)
```

```
ig, ax = plt.subplots(figsize=(12, 8))
state_merged.plot(
    column='Count',
    cmap='viridis',
    legend=True,
    ax=ax,
    edgecolor='black',  # Add borders for clarity
    linewidth=0.5)
ax.set_title('Enforcement Actions by State', fontsize=14)
#Removed Alska as it made the graph look horrible
ax.set_xlim([-130, -65])
ax.set_ylim([24, 50])
plt.show()
```



2. (Partner 2) Map by district: Among actions taken by US Attorney District-level agencies, clean the district names so that you can merge them with the shapefile, and then plot a choropleth of the number of enforcement actions in each US Attorney District. Hint: look for "District" in the agency info.

```
#os.chdir('/Users/willsigal/Desktop/US Attorney Districts Shapefile
 ↪  simplified_20241108') #will wd
#districts_gdf =
 ↪  gpd.read_file('geo_export_6b570657-b5c6-4bbc-b83a-415e4880d085.shp')
 ↪  #will import

os.chdir('d:\\UChicago\\Classes\\2024Qfall\\Programming
 ↪  Python\\problem-set-5\\Shapefiles\\US Attorney Districts Shapefile
 ↪  simplified_20241109') #andy wd
districts_gdf =
 ↪  gpd.read_file('geo_export_5d56b225-6390-4cf1-bb2b-122a8672faf1.shp')
 ↪  #andy import
```

```
district_df = pd.DataFrame(districts_gdf)
```

```
district_twentyone_df =
 ↪  twenty_one[twenty_one['Agency'].str.contains('District',
 ↪  na=False)].copy()
```

```
district_df.head
```

```
<bound method NDFrame.head of    statefp
judicial_d         aland        awater  \
0       21           Western District of Kentucky  4.970555e+10  1.651516e+09
1       21           Eastern District of Kentucky  5.257394e+10  7.238213e+08
2       18           Southern District of Indiana  5.824517e+10  5.941176e+08
3       01             Middle District of Alabama  3.412673e+10  5.472423e+08
4       01           Southern District of Alabama  6.235882e+10  3.052681e+09
..     ...                                    ...           ...           ...
89      69  District of Northern Marianas Islands  4.722925e+08  4.644252e+09
90      12           Southern District of Florida  2.448809e+10  1.159277e+10
91      40          Northern District of Oklahoma  2.231989e+10  7.189768e+08
92      50                    District of Vermont  2.387418e+10  1.030417e+09
93      10                   District of Delaware  5.045926e+09  1.399986e+09

                       state           chief_judg            nominating  \
0                   Kentucky      Greg N. Stivers      Barack Obama (D)
1                   Kentucky         Danny Reeves    George W. Bush (R)
2                    Indiana  Jane Magnus-Stinson      Barack Obama (D)
3                    Alabama    Emily Coody Marks      Donald Trump (R)
4                    Alabama        Kristi DuBose    George W. Bush (R)
..                       ...                  ...                   ...
89  Northern Marianas Islands   Ramona V. Manglona      Barack Obama (D)
90                   Florida     K. Michael Moore  George H.W. Bush (R)
91                  Oklahoma         John Dowdell      Barack Obama (D)
92                   Vermont    Geoffrey Crawford      Barack Obama (D)
93                  Delaware        Leonard Stark      Barack Obama (D)

    term_as_ch  shape_leng  shape_area abbr district_n    shape__are  \
0       2018.0   16.200585    5.216899  KYW          6  8.123902e+10
1       2019.0   13.514251    5.451047  KYE          6  8.547129e+10
2       2016.0   14.956126    6.137433  INS          7  9.818187e+10
3       2019.0   10.235799    3.858442  ALM         11  5.645450e+10
```

```
4       2017.0   12.976906    3.278871  ALS         11  4.772733e+10
..         ...         ...         ...  ...        ...           ...
89      2011.0    3.252892    0.040335   MP          9  5.200276e+08
90      2014.0   17.941594    2.503158  FLS         11  3.466156e+10
91      2019.0    8.154257    2.316496  OKN         10  3.569729e+10
92      2017.0    9.571257    2.797955   VT          2  4.826994e+10
93      2014.0    4.285079    0.541590   DE          3  8.634830e+09

      shape__len                                           geometry
0   1.964255e+06  MULTIPOLYGON (((-89.48248 36.50214, -89.48543 ...
1   1.654681e+06  POLYGON ((-84.62012 39.07346, -84.60793 39.073...
2   1.887626e+06  POLYGON ((-85.86281 40.46476, -85.86212 40.406...
3   1.236201e+06  POLYGON ((-85.33828 33.49471, -85.33396 33.492...
4   1.567095e+06  MULTIPOLYGON (((-88.08682 30.25987, -88.07676 ...
..           ...                                                ...
89  3.702108e+05  MULTIPOLYGON (((145.28433 14.17537, 145.28404 ...
90  2.117424e+06  MULTIPOLYGON (((-81.9667 24.52376, -81.97837 2...
91  9.857548e+05  POLYGON ((-95.04951 36.99959, -95.03786 36.999...
92  1.248962e+06  POLYGON ((-72.67477 45.01547, -72.58988 45.013...
93  5.622177e+05  MULTIPOLYGON (((-75.56246 39.51266, -75.56745 ...

[94 rows x 15 columns]>
```

```
#removing the prefixes
district_twentyone_df['District'] =
 ↪  district_twentyone_df['Agency'].apply(lambda x: x.split(', ')[-1] if ', '
 ↪  in x else x)
district_twentyone_df['District'] =
 ↪  district_twentyone_df['District'].str.strip()

district_twentyone_df['District'] =
 ↪  district_twentyone_df['District'].str.replace('U.S. Attorney\'s Office,
 ↪  ', '')

district_twentyone_df['District'].head
```

```
<bound method NDFrame.head of 1                   District of Idaho
2          District of Massachusetts
3       Eastern District of Virginia
4         Middle District of Florida
5          Western District of Texas
                   ...
```

```
3016             District of Montana
3017              District of Alaska
3018          District of New Jersey
3020          District of New Jersey
3021    Middle District of Tennessee
Name: District, Length: 1413, dtype: object>
```

```
enforcement_counts =
 ↪  district_twentyone_df['District'].value_counts().reset_index()
enforcement_counts.columns = ['District', 'Enforcement Actions']
```

```
merged_districts_df = pd.merge(district_df, enforcement_counts,
 ↪  left_on='judicial_d', right_on='District', how='left')

merged_districts_df['Enforcement Actions'] = merged_districts_df['Enforcement
 ↪  Actions'].fillna(0)
merged_districts_df.columns
```

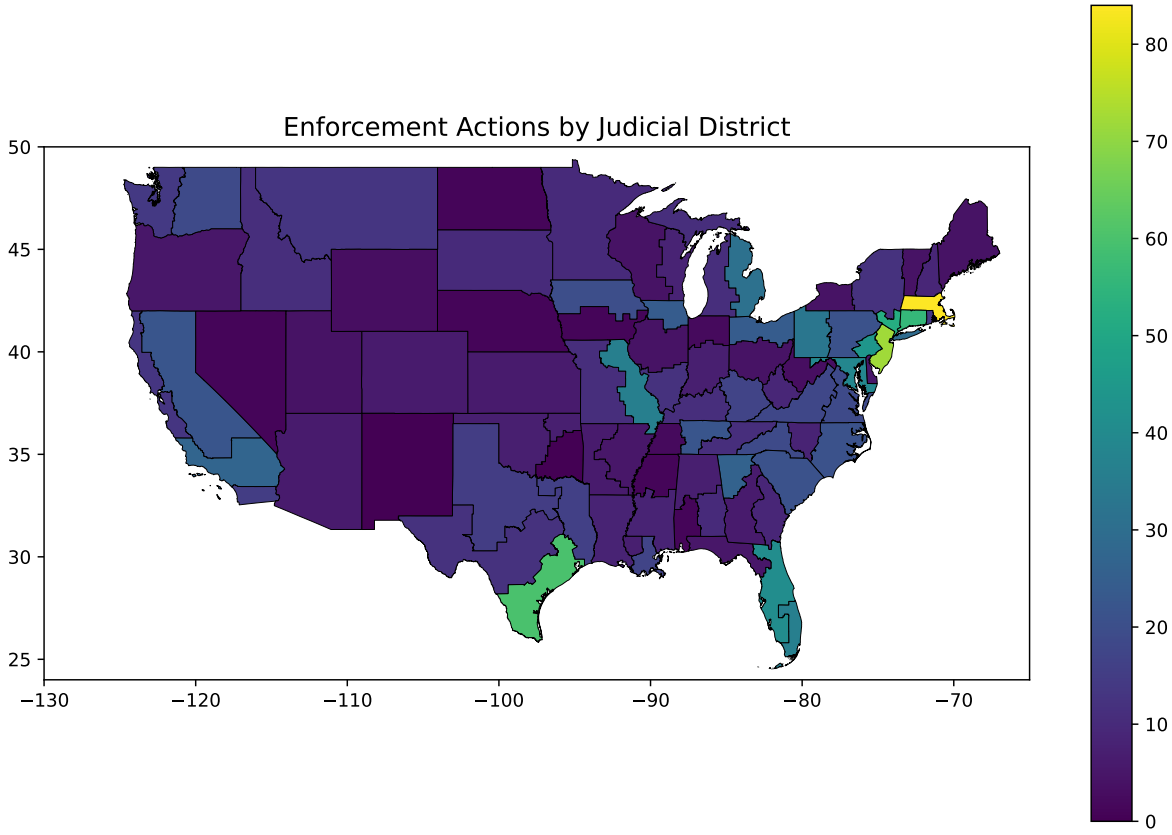
```
Index(['statefp', 'judicial_d', 'aland', 'awater', 'state', 'chief_judg',
       'nominating', 'term_as_ch', 'shape_leng', 'shape_area', 'abbr',
       'district_n', 'shape__are', 'shape__len', 'geometry', 'District',
       'Enforcement Actions'],
      dtype='object')
```

```
gdf = gpd.GeoDataFrame(merged_districts_df, geometry='geometry')
print(gdf.crs)
gdf['Enforcement Actions'] = gdf['Enforcement Actions'].fillna(0)
```

```
EPSG:4326
```

```
ig, ax = plt.subplots(figsize=(12, 8))  # Adjust figure size
gdf.plot(
    column='Enforcement Actions',
    cmap='viridis',
    legend=True,
    ax=ax,
    edgecolor='black',  # Add borders for clarity
    linewidth=0.5)
ax.set_title('Enforcement Actions by Judicial District', fontsize=14)
#Removed Alska as it made the graph look horrible
ax.set_xlim([-130, -65])
```

```
ax.set_ylim([24, 50])
plt.show()
```



(10 points) Extra credit: Calculate the enforcement actions on a per-capita basis (Both partners can work together)

1. Use the zip code shapefile from the previous problem set and merge it with zip code level population data. (Go to Census Data Portal, select "ZIP Code Tabulation Area", check "All 5-digit ZIP Code Tabulation Areas within United States", and under "P1 TOTAL POPULATION" select "2020: DEC Demographic and Housing Characteristics". Download the csv.).

2. Conduct a spatial join between zip code shapefile and the district shapefile, then aggregate to get population in each district.

3. Mapthe ratio of enforcement actions in each US Attorney District. You can calculate the ratio by aggregating the number of enforcement actions since January 2021 per district, and dividing it with the population data.