PS5 Answer

Sienna Wang & Hengyi Xing

2024-11-07

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): Hengyi Xing, hengyix
   - Partner 2 (name and cnet ID): Sienna Wang, shiying
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: ***HX* *SW***
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

```python
import pandas as pd
import altair as alt
import geopandas as gpd
import requests
import time
from bs4 import BeautifulSoup
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from shapely.wkt import loads

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

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```python
# Fetch contents from the website
url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
li_action = soup.find_all(
    'li', class_='usa-card card--list pep-card--minimal mobile:grid-col-12')

# Parse into a dataset
title_list = []
date_list = []
category_list = []
link_list = []
for item in li_action:
    title = item.find('a').text
    title_list.append(title)
    date = item.find('span').text
    date_list.append(date)
    category = item.find('li').text
    category_list.append(category)
    link = item.find('a').attrs['href'] # Relative Path
    link = 'https://oig.hhs.gov' + link
    link_list.append(link)

df_dict = {'Title': title_list, 'Date': date_list,
           'Category': category_list, 'Link': link_list}
df_tidy = pd.DataFrame(df_dict)
print(df_tidy.head(5))
```

```
                                               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-...
```

```
# Since the list cells are too long to display
# We demonstrate a complete example here
print(link_list[0][0:80])
print(link_list[0][80:])
```

```
https://oig.hhs.gov/fraud/enforcement/pharmacist-and-brother-convicted-of-15m-me
dicare-medicaid-and-private-insurer-fraud-scheme/
```

2. Crawling (PARTNER 1)

```
agency_list = []

for link in link_list:
    response = requests.get(link)
    soup = BeautifulSoup(response.content, "lxml")

    agency_li = soup.find("span", text=lambda t: t and "Agency" in t)

    if agency_li:
        agency_name = agency_li.find_parent("li").text.replace("Agency:",
↪   "").strip()
    else:
        agency_name = "missing value"

    agency_list.append(agency_name)

df_tidy["Agency"] = agency_list
print(df_tidy.head(5))
```

```
                                               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)

Step 1:
The function will first check if year is earlier than 2013.

Step 2:
If the year is not earlier than 2013, we will loop through the input date to today.
Here I will use a **while** loop, which will go through each page until there is no more dates that fit in our specified date range. For each page, we will read through all enforcement action entries.

Step 3:
Here I will use a **for** loop. For each entry on that page, I will extract date to check if that entry is in the specified range, and will only process entries that are from the specified start date to today.

Step 4:
Continue to extract the title, category, and link, and append them to to the list respectively.

Step 5:
Crawl into each link to extract the agency information. And move to the next page (another round in the While loop).

Step 6:
Create dataframe and save to CSV.

b. Create Dynamic Scraper (PARTNER 2)

```
def enforce_actions_crawl(month, year):
  '''Output a CSV file including information of enforcement actions from the
   ↪  date input to today'''

  if year < 2013:
    return "Reminder: Please enter a year after 2013, as only enforcement
     ↪  actions after 2013 are listed."

  # For year >= 2013:
  title_list = []
  date_list = []
  category_list = []
  link_list = []
  agency_list = []

  url_base = "https://oig.hhs.gov/fraud/enforcement/"
  page = 1
  today = datetime.now()
  start_date = datetime(year, month, 1)
  should_continue = True

  while should_continue:
    print(f"Processing page {page}") # To help us learn the progress
    url = url_base + f"?page={page}"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "lxml")
    li_action = soup.find_all(
      "li", class_="usa-card card--list pep-card--minimal
       ↪  mobile:grid-col-12")

    if not li_action:
      break

    for item in li_action:
      # Check if the date is in the specified date range
      date_text = item.find("span").text.strip()
      action_date = datetime.strptime(date_text, "%B %d, %Y")
      if action_date < start_date:
        should_continue = False # Finish the while loop
        break

      # If within the specified date range, continue
      title = item.find("a").text.strip()
      category = item.find("li").text.strip()
      link = item.find("a")["href"]
      link = "https://oig.hhs.gov" + link

      title_list.append(title)
      date_list.append(date_text)
      category_list.append(category)
      link_list.append(link)

      # Crawl each link and find agency information
      action_response = requests.get(link)
      action_soup = BeautifulSoup(action_response.text, "lxml")

      agency_li = action_soup.find("span", text=lambda t: t and "Agency" in t)
      if agency_li:
        agency = agency_li.find_parent("li").text.replace("Agency:", "").strip()
      else:
        agency = "missing value"
      agency_list.append(agency)

    if not should_continue:
      break

    page += 1
    time.sleep(1) # Avoid server block

  # Create dataframe and save to CSV file
  df_dict = {"Title": title_list, "Date": date_list, "Category": category_list, "Link": link_list, "Agency": agency_list}
  df_tidy = pd.DataFrame(df_dict)

  csv_filename = f"enforcement_actions_{year}_{month}.csv"
  df_tidy.to_csv(csv_filename, index=False)
  print(f"Data saved to {csv_filename}")
```

And then, use the function to collect enforcement actions since 2023.1.

```
enforce_actions_crawl(1, 2023)
```

```
path = '/Users/hengyix/Documents/GitHub/DAP2-PS5-SWang/data/'
file = 'enforcement_actions_2023_1.csv'
df_2023 = pd.read_csv(path + file)
print(f"There are {len(df_2023)} enforcement actions in the final
↪  dataframe.")
print(df_2023.iloc[-1])
```

```
There are 1534 enforcement actions in the final dataframe.
Title       Podiatrist Pays $90,000 To Settle False Billin...
Date                                          January 3, 2023
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
```

The date of the earliest record is January 3, 2023. The title is "Podiatrist Pays $90,000 To Settle False Billing Allegations", in the cateogry of Criminal and Civil Actions, conducted by U.S. Attorney's Office, Southern District of Texas. And the relevant link is https://oig.hhs.gov/fraud/enforcement/podiatrist-pays-90000-to-settle-false-billing-allegations/.

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

```
enforce_actions_crawl(1, 2021)
```

```
# Load the CSV file
file = 'enforcement_actions_2021_1.csv'
df = pd.read_csv(path + file)
print(f'We get {len(df)} enforcement actions in our dataframe.')
print(df.iloc[3021])
```

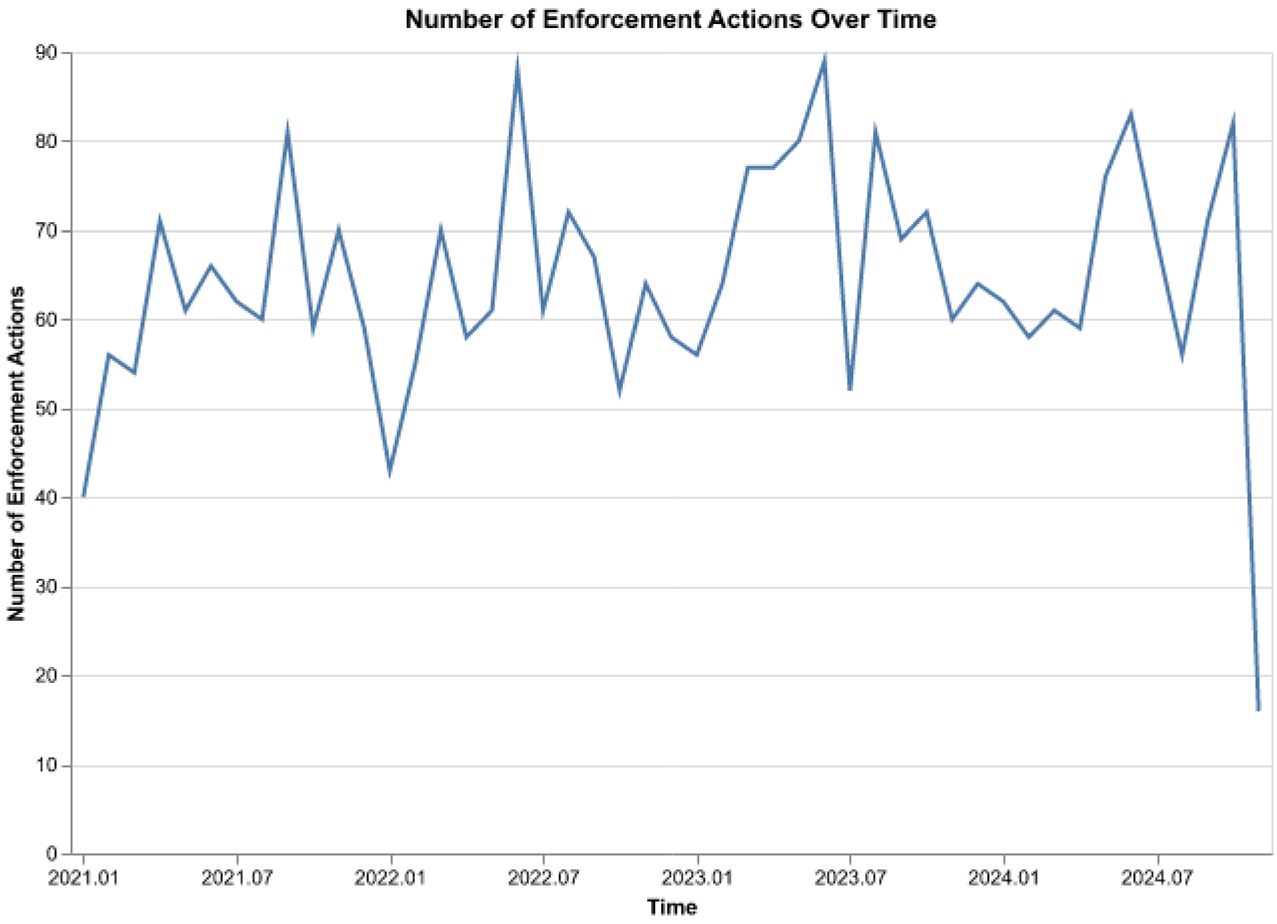
```
We get 3022 enforcement actions in our dataframe.
Title       The United States And Tennessee Resolve Claims...
Date                                          January 4, 2021
Category                              Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/the-unit...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

SO we know that the earliest action scraped is the record of The United States And Tennessee Resolve Claims With Three Providers For False Claims Act Liability Relating To 'P-Stim' Devices For A Total Of $1.72 Million, with date being 2021-01-04. The category and agency are Criminal and Civil Actions and U.S. Attorney's Office, Middle District of Tennessee, link: https://oig.hhs.gov/fraud/enforcement/the-united-states-and-tennessee-resolve-claims-with-three-providers-for-false-claims-act-liability-relating-to-p-stim-devices-for-a-total-of-172-million/

Step 3: Plot data based on scraped data

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

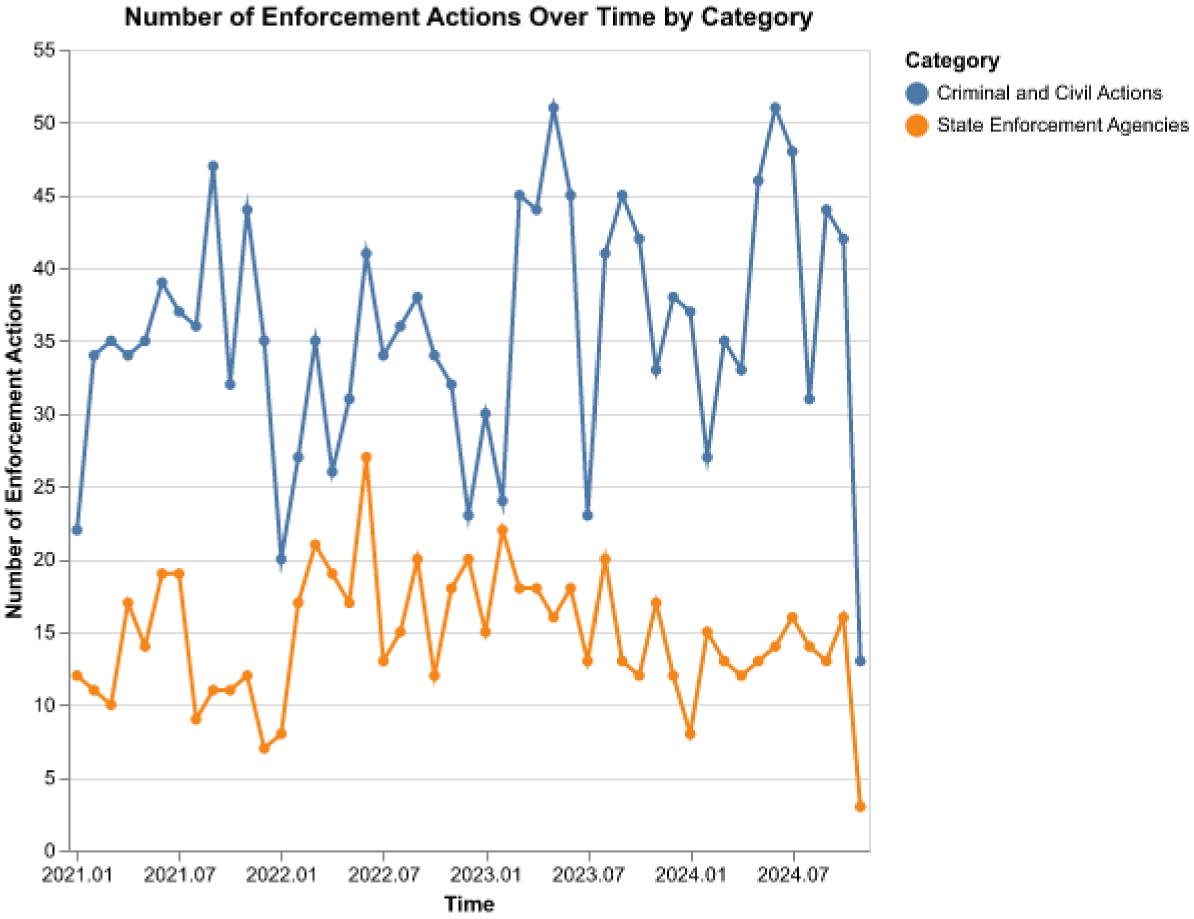
```
# Create a new "Year_Month" column
df["Date"] = pd.to_datetime(df["Date"], format="%B %d, %Y")
df["Year_Month"] = df["Date"].dt.strftime("%Y.%m")

# Count the number by "Year_Month"
monthly_count = df.groupby("Year_Month").size().reset_index(name="Count")

# Plot a line chart using Altair
alt.Chart(monthly_count).mark_line().encode(
    alt.X("Year_Month:O",
          title="Time",
          axis=alt.Axis(
              values=["2021.01", "2021.07", "2022.01", "2022.07",
                      "2023.01", "2023.07", "2024.01", "2024.07"],
              labelAngle=0)),
    alt.Y("Count:Q", title="Number of Enforcement Actions")
).properties(
    title="Number of Enforcement Actions Over Time",
    width=600,
    height=400
)
```



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

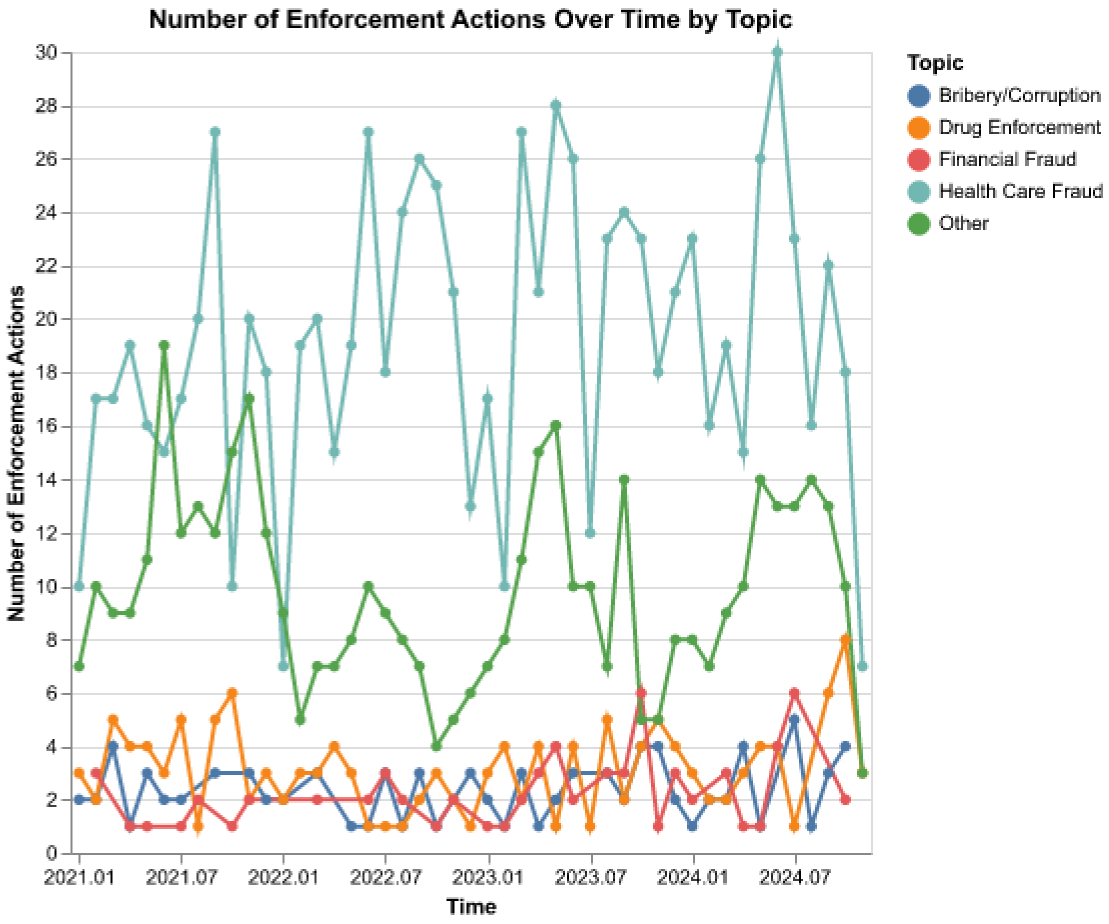
based on “Criminal and Civil Actions” vs. “State Enforcement Agencies”

```
# Filter out records and do monthly count
df_actions_agencies = df[(df['Category'] == 'Criminal and Civil Actions') | (
    df['Category'] == 'State Enforcement Agencies')]
category_count = df_actions_agencies.groupby(
    ['Category', 'Year_Month']).size().reset_index(name="Count")

# Draw the plot
alt.Chart(category_count ).mark_line(point=True).encode(
     alt.X("Year_Month:O",
          title="Time",
            axis=alt.Axis(
                values=["2021.01", "2021.07", "2022.01", "2022.07",
                        "2023.01", "2023.07", "2024.01", "2024.07"],
                labelAngle=0)),
    alt.Y("Count:Q", title="Number of Enforcement Actions"),
    alt.Color('Category:N')
).properties(
    title="Number of Enforcement Actions Over Time by Category",
    width=400,
    height=400
)
```



based on five topics

```
# Define topics roughly
health_care = ['Medicare', 'Medicaid', 'Healthcare',
               'Medical', 'Surgeon', 'Pharmacy', 'Medi-cal']  # 'Health Care'
                ↪  should also be added
drug_enforcement = ['Drug', 'Drugs', 'Pill', 'Pills', 'Substance',
                    'Substances', 'Medication', 'Narcotic', 'Morphine']
bribery_corruption = ['Corruption', 'Corrupt', 'Bribery',
                      'Bribes', 'Gratuities', 'Kickback', 'Kickbacks']
financial = ['Financial', 'Monetary', 'Embezzlement', 'Marketing',
             'Laundering', 'Business', 'Bank', 'Money', 'Scheme']


def find_topic(action_name):
    '''
    To match action names to five topics.
    We consider the special case of 'Health Care' phrase, because 'Health'
 ↪  and 'Care'
    are inaccurate when matched separately.
    '''
    health_match = False
    drug_match = False
    bribery_match = False
    financial_match = False

    # Special case for "Health Care" phrase
    if 'Health Care' in action_name:
        health_match = True

    action_text = action_name.split(' ')
    for text in action_text:
        text = text.strip(',')
        if text in health_care:
            health_match = True
        if text in drug_enforcement:
            drug_match = True
        if text in bribery_corruption:
            bribery_match = True
        if text in financial:
            financial_match = True
    if health_match:
        return "Health Care Fraud"
    elif drug_match:
        return "Drug Enforcement"
    elif bribery_match:
        return "Bribery/Corruption"
    elif financial_match:
        return "Financial Fraud"
    else:
        return "Other"


df_criminal = df[df['Category'] == 'Criminal and Civil Actions']
df_criminal['Topic'] = df_criminal['Title'].apply(find_topic)

# Filter out records and do monthly count
topic_count = df_criminal.groupby(
    ['Topic', 'Year_Month']).size().reset_index(name="Count")


alt.Chart(topic_count).mark_line(point=True).encode(
    alt.X("Year_Month:O",
          title="Time",
          axis=alt.Axis(
              values=["2021.01", "2021.07", "2022.01", "2022.07",
                      "2023.01", "2023.07", "2024.01", "2024.07"],
              labelAngle=0)),
    alt.Y("Count:Q", title="Number of Enforcement Actions"),
    alt.Color('Topic:N')
).properties(
    title="Number of Enforcement Actions Over Time by Topic",
    width=400,
    height=400
)
```

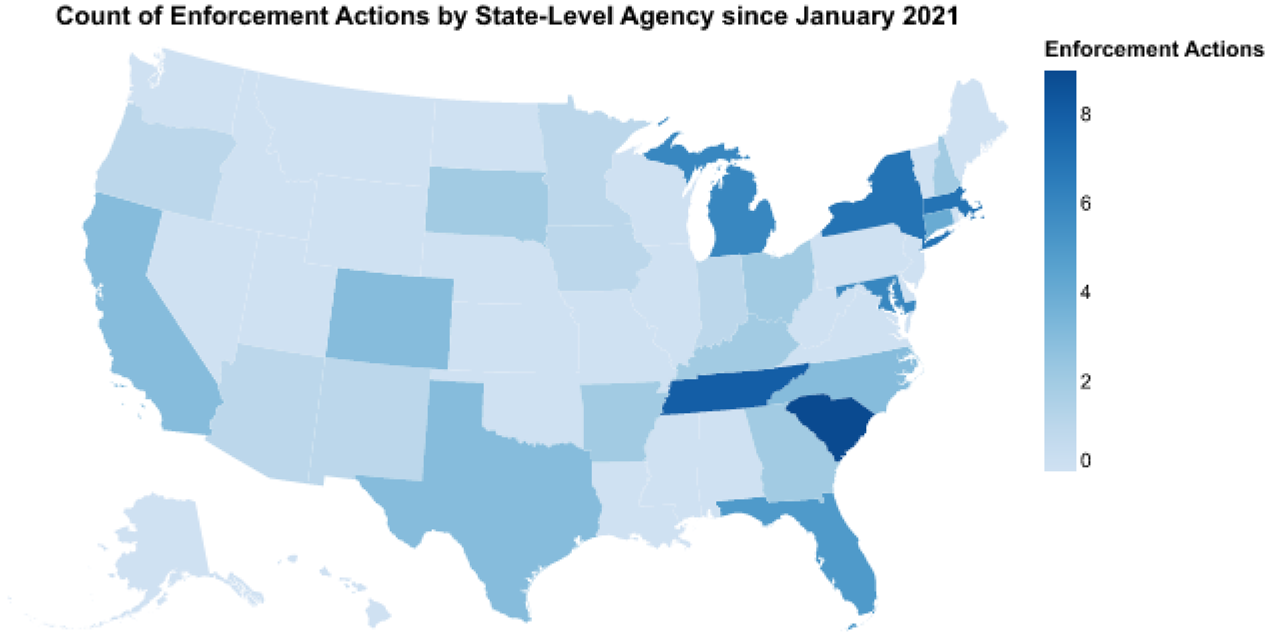


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
# Filter out state level records and clean state names
df_state = df[df['Agency'].str.contains('State of', na=False)]
df_state['Agency'] = df_state['Agency'].str.replace(
    'State of ', '', regex=False)
# Count by state
state_count = df_state.groupby('Agency').size().reset_index(name='Count')
state_count = state_count.rename(columns={'Agency': 'State'})

path = '/Users/hengyix/Documents/GitHub/DAP2-PS5-SWang/data/'
```

```
file = "cb_2018_us_state_500k/cb_2018_us_state_500k.shp"
state_shapefile = gpd.read_file(path + file)
gdf_state = pd.merge(state_shapefile, state_count,
                     how='left', left_on='NAME', right_on='State')
# Fill in NAs with zero
gdf_state['Count'] = gdf_state['Count'].fillna(0)

# Choropleth Map
alt.Chart(gdf_state).mark_geoshape().encode(
    color=alt.Color('Count:Q', scale=alt.Scale(scheme='blues'),
                    legend=alt.Legend(title="Enforcement Actions"))
).project(
    type='albersUsa'
).properties(
    width=500,
    height=300,
    title='Count of Enforcement Actions by State-Level Agency since January
 ↪  2021'
)
```


Count of Enforcement Actions by State-Level Agency since January 2021
Enforcement Actions
8
6
4
2
0

2. Map by District (PARTNER 2)

```
path = '/Users/hengyix/Documents/GitHub/DAP2-PS5-SWang/data/'
file = 'US Attorney Districts Shapefile
 ↪  simplified_20241110/geo_export_9484d0ea-f168-4c8e-ba9c-c45ba8de7107.shp'
gdf_district = gpd.read_file(path + file)
gdf_district = gdf_district.rename(columns={"judicial_d":
 ↪  "judicial_district"})
```

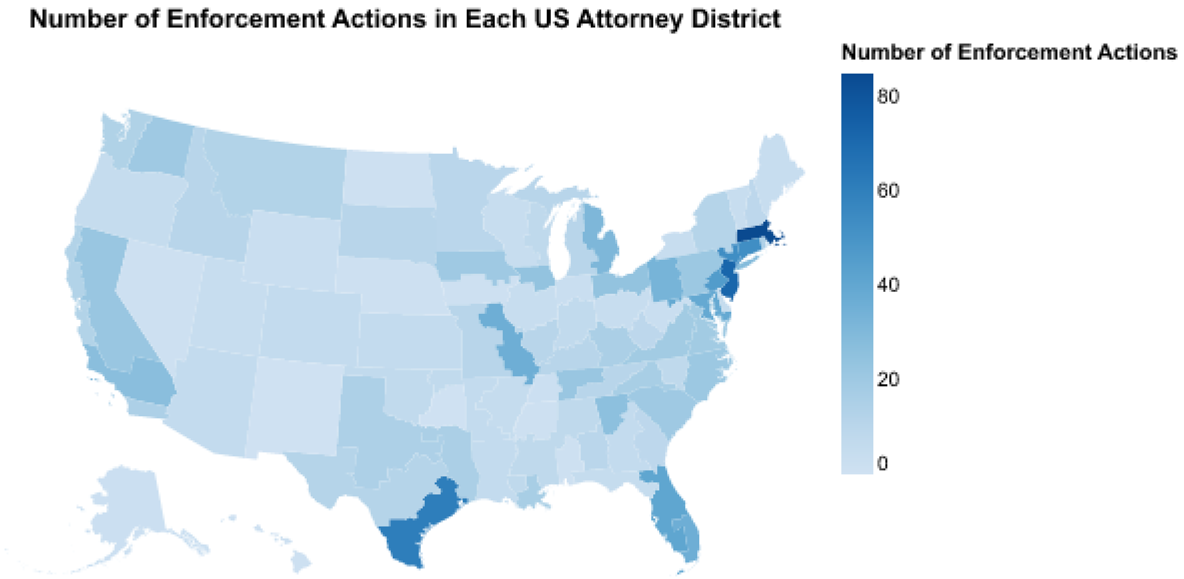
```
# Filter the data with agencies from districts
df_district = df[df["Agency"].str.contains("District", na=False)]
df_district["judicial_district"] = df_district["Agency"].str.extract(
    r"((?:Western|Eastern|Northern|Southern|Middle|Central)?\s*District of
     ↪  [A-Za-z\s]+)")
df_district["judicial_district"] =
 ↪  df_district["judicial_district"].str.strip()
```

We find that there's a mismatch for the District of Columbia. In our collected dataset, it is named "District of Columbia", while for the shapefile, it is named "District of District of Columbia". Therefore, we change its name in our collected datset to "District of District of Columbia" to merge conveniently. Also there are several actions taken by 2 different districts. In this case, we only keep the first one in our record.

```
# Clean the "District" column
df_district["judicial_district"] = df_district["judicial_district"].replace(
    ["District of Columbia Inspector General", "District of Columbia"],
 ↪  "District of District of Columbia")
df_district["judicial_district"] = df_district["judicial_district"].replace(
    "Western District of Kentucky and U", "Western District of Kentucky")
df_district["judicial_district"] = df_district["judicial_district"].replace(
    "Eastern District of Pennsylvani", "Eastern District of Pennsylvania")
df_district["judicial_district"] = df_district["judicial_district"].replace(
    "Southern District of Florida and U", "Southern District of Florida")
df_district["judicial_district"] = df_district["judicial_district"].replace(
    "Southern District of Texas and U", "Southern District of Texas")

# Count how many actions are there by each district
district_counts =
 ↪  df_district["judicial_district"].value_counts().reset_index()
```

```
# Merge into the GeoDataFrame
gdf_district = gdf_district.merge(
    district_counts, on="judicial_district", how="left")
```

```
# Clean the GeoDataFrame for Altair
gdf_district = gdf_district[["geometry", "judicial_district", "count"]]
gdf_district["count"] = gdf_district["count"].fillna(0)

gdf_district_count = gdf_district.copy()
# Plot the Choropleth
alt.Chart(gdf_district_count).mark_geoshape().encode(
    alt.Color("count:Q", title="Number of Enforcement Actions",
              scale=alt.Scale(scheme="blues"))
).project(
    type="albersUsa"
).properties(
    title="Number of Enforcement Actions in Each US Attorney District",
    width=400
)
```


Number of Enforcement Actions in Each US Attorney District
Number of Enforcement Actions
80
60
40
20
0

Extra Credit

1. Merge zip code shapefile with population

```
# Import the ZIP code shapefile
path = "/Users/hengyix/Documents/GitHub/problem-set-4-hengyi-and-sienna/data"
gdf_zip = gpd.read_file(path +
 ↪  "/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp")
```

```
# Import the ZIP code level population data
path = '/Users/hengyix/Documents/GitHub/DAP2-PS5-SWang/data'
df_zip_pop = pd.read_csv(path +
 ↪  "/DECENNIALDHC2020.P1_2024-11-10T034849/DECENNIALDHC2020.P1-Data.csv")

# Drop the empty column and label row
df_zip_pop = df_zip_pop.drop(columns=["Unnamed: 3"])
df_zip_pop = df_zip_pop.iloc[1:]

# Clean the ZCTA5 and Population columns
df_zip_pop["ZCTA5"] = df_zip_pop["NAME"].str.replace("ZCTA5 ", "")
df_zip_pop = df_zip_pop.rename(columns={"P1_001N": "Population"})
```

```
# Merge the population data into the shapefile
gdf_zip = gdf_zip.merge(df_zip_pop[["ZCTA5", "Population"]], on="ZCTA5",
 ↪  how="left")
```

```
# Show the first 5 rows
print(gdf_zip.head(5))
```

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

2. Conduct spatial join

```
path = '/Users/hengyix/Documents/GitHub/DAP2-PS5-SWang/data/'
file = 'US Attorney Districts Shapefile
↪  simplified_20241110/geo_export_9484d0ea-f168-4c8e-ba9c-c45ba8de7107.shp'
district_shapefile = gpd.read_file(path + file)
gdf_district = gdf_district.rename(columns={"judicial_d":
↪  "judicial_district"})

# Keep only useful columns
gdf_district = gdf_district[["judicial_district", "geometry"]]
```

```
# Spatial join the 2 GeoDataFrames
gdf_district_pop = gpd.sjoin(gdf_district, gdf_zip, how="inner",
↪  predicate="intersects")
```

Here, as one zip code area could intersect with several different districts, and might lead to inaccuracy in population aggregation. Therefore, in this case, we assign these ZIP code to the district that contains its most area.

```
# Find the area
gdf_intersection = gpd.overlay(gdf_district, gdf_zip, how="intersection")
gdf_intersection["area"] = gdf_intersection.area
gdf_district_pop = gdf_district_pop.merge(gdf_intersection[["ZCTA5",
↪  "area"]], how="left", on="ZCTA5")

# Assign the ZIP codes to the most reasonable districts
gdf_district_pop = gdf_district_pop.sort_values(by="area",
↪  ascending=False).drop_duplicates(subset="ZCTA5", keep="first")
```

Finally, we can aggregate the population to the district level.

```
gdf_district_pop["Population"] =
↪  pd.to_numeric(gdf_district_pop["Population"], errors="coerce")
gdf_district_pop = gdf_district_pop.dissolve(by="judicial_district",
↪  aggfunc={"Population": "sum"})
```

```
gdf_district_pop = gdf_district_pop.reset_index()
print(gdf_district_pop.head(10))
```

```
                  judicial_district  \
0    Central District of California
1      Central District of Illinois
2                District of Alaska
3               District of Arizona
4              District of Colorado
5           District of Connecticut
6              District of Delaware
7  District of District of Columbia
8                District of Hawaii
9                 District of Idaho

                                            geometry  Population
0  MULTIPOLYGON (((-120.34772 34.02012, -120.3538...  19132406.0
1  POLYGON ((-90.24672 41.74559, -90.24413 41.745...   2234070.0
2  MULTIPOLYGON (((-179.13027 51.21418, -179.1093...    707199.0
3  POLYGON ((-110.17572 36.9984, -110.14987 36.99...   6995090.0
4  POLYGON ((-106.32118 40.99913, -106.21758 40.9...   5708882.0
5  MULTIPOLYGON (((-73.64577 40.99601, -73.64449 ...   3553969.0
6  MULTIPOLYGON (((-75.58036 39.59946, -75.57772 ...    900370.0
7  POLYGON ((-77.1007 38.94892, -77.10053 38.9490...    631870.0
8  MULTIPOLYGON (((-175.93621 27.75999, -175.9303...   1454203.0
9  POLYGON ((-116.0491 48.44032, -116.04902 48.39...   1887633.0
```

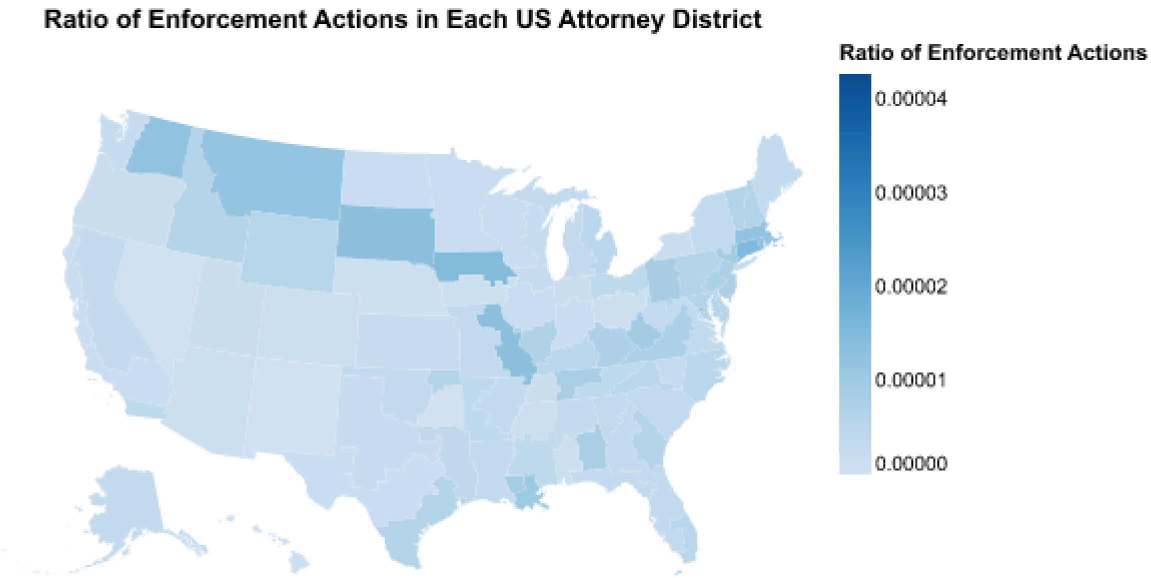
3. Map the action ratio in each district

```
# Calculate the ratio of actions per district
# The number of actions per district has been calculated, saved in
↪   gdf_district_count

# Merge the number of actions info into population geodategrame
gdf_district_ratio = gdf_district_pop.merge(
    gdf_district_count[["judicial_district", "count"]],
    on="judicial_district",
    how="left"
)
```

```
# Calculate the ratio
gdf_district_ratio["ratio"] = (
    gdf_district_ratio["count"].astype('float') /
 ↪  gdf_district_ratio["Population"].astype('float')
)
```

As mentioned in Question 4, due to the name mismatch for the District of Columbia across datasets (websites list it as “District of Columbia” or “District of Columbia General Inspector,” while the shapefile labels it as “District of District of Columbia”), we resolved the inconsistency by standardizing the names. This adjustment led to a higher count of enforcement actions in DC, resulting in an exceptionally high ratio of actions per capita here (4.27e-5), compared to the next highest (1.49e-5). Consequently, our map appears lighter overall, as DC’s value skews the color scale.

```
# Map the choropleth
alt.Chart(gdf_district_ratio).mark_geoshape().encode(
    alt.Color("ratio:Q", title="Ratio of Enforcement Actions",
              scale=alt.Scale(scheme="blues"))
).project(
    type="albersUsa"
).properties(
    title="Ratio of Enforcement Actions in Each US Attorney District",
    width=400
)
```



Ratio of Enforcement Actions in Each US Attorney District
Ratio of Enforcement Actions
0.00004
0.00003
0.00002
0.00001
0.00000