

# PS 5

Yuxuan Geng & George Wang

2024-11-10

**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): George Wang, gwang613
  - Partner 2 (name and cnet ID): Yuxuan Geng, yuxuan1123
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: **GW YG**
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: **2** 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

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

RendererRegistry.enable('png')

```

## Step 1: Develop initial scraper and crawler

### 1. Scraping (PARTNER 1)

```

import pandas as pd
import requests
from bs4 import BeautifulSoup

# Get web page through soup
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')

# Lists to store scraped data
titles = []
dates = []
categories = []
links = []

for action in soup.find_all('li', class_='usa-card card--list
    ↵ pep-card--minimal mobile:grid-col-12'):
    # Extract title
    title = action.find('h2',
    ↵ class_='usa-card__heading').get_text(strip=True)
    titles.append(title)

    # Extract link
    link = action.find('a')['href']
    links.append(f"https://oig.hhs.gov{link}")

```

```

# Extract date
date = action.find('span', class_='text-base-dark
↪ padding-right-105').get_text(strip=True)
dates.append(date)

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

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

data

```

	Title	Date	Category
0	Pharmacist and Brother Convicted of \$15M Medic...	November 8, 2024	Criminal and Civil Action
1	Boise Nurse Practitioner Sentenced To 48 Month...	November 7, 2024	Criminal and Civil Action
2	Former Traveling Nurse Pleads Guilty To Tamper...	November 7, 2024	Criminal and Civil Action
3	Former Arlington Resident Sentenced To Prison ...	November 7, 2024	Criminal and Civil Action
4	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024	Criminal and Civil Action
5	Former Licensed Counselor Sentenced For Defrau...	November 6, 2024	Criminal and Civil Action
6	Macomb County Doctor And Pharmacist Agree To P...	November 4, 2024	Criminal and Civil Action
7	Rocky Hill Pharmacy And Its Owners Indicted Fo...	November 4, 2024	Criminal and Civil Action
8	North Texas Medical Center Pays \$14.2 Million ...	November 4, 2024	Criminal and Civil Action
9	New England Doctor Pleads Guilty To Drug Distr...	November 4, 2024	Criminal and Civil Action
10	Attorney General Alan Wilson Announces Upstate...	November 4, 2024	State Enforcement Agency
11	St. Louis County Woman Accused Of \$3 Million H...	November 1, 2024	Criminal and Civil Action
12	Lab Owner And Marketing Company Owner Both Fou...	November 1, 2024	Criminal and Civil Action
13	Compound Ingredient Supplier Medisca Inc., To ...	November 1, 2024	Criminal and Civil Action
14	The New Mexico Department Of Justice Charges F...	November 1, 2024	State Enforcement Agency
15	Nashville Woman Indicted, Charged In TBI Medic...	November 1, 2024	State Enforcement Agency
16	Michael DePalma, MD and Virginia I-Spine Physi...	October 31, 2024	CMP and Affirmative Exec...
17	Columbus Doctor, His Clinic Convicted of \$1.5 ...	October 31, 2024	State Enforcement Agency
18	Mercy Health Youngstown Agreed to Pay \$69,000 ...	October 30, 2024	Fraud Self-Disclosures

Title	Date	Category
19 Quincy-Based Physician Group To Pay \$650,000 T...	October 30, 2024	State Enforcement Agency

## 2. Crawling (PARTNER 1)

```
# Initialize
agencies = []
row_count = 0

# Loop through the dataframe to get agencies
for link in data['Link']:

    # Row counting
    row_count += 1
    print(f"Processing row {row_count}")

    # Make a request
    detail_response = requests.get(link)
    detail_soup = BeautifulSoup(detail_response.content, 'lxml')

    # Extract the agency NameError()
    agency_elements = detail_soup.find_all('span', class_='padding-right-2
    ↵ text-base')

    # Check if the second element exists
    if len(agency_elements) > 1:
        agency_li = agency_elements[1].next_sibling
        # Check if next_sibling is a string and strip it
        agency_name = agency_li.strip() if isinstance(agency_li, str) else
    ↵ str(agency_li).strip()
    else:
        agency_name = None

    # Append the extracted agency
    agencies.append(agency_li)

# Add the 'Agency' column
data['Agency'] = agencies

data
```

Processing row 1  
Processing row 2  
Processing row 3  
Processing row 4  
Processing row 5  
Processing row 6  
Processing row 7  
Processing row 8  
Processing row 9  
Processing row 10  
Processing row 11  
Processing row 12  
Processing row 13  
Processing row 14  
Processing row 15  
Processing row 16  
Processing row 17  
Processing row 18  
Processing row 19  
Processing row 20

	Title	Date	Category
0	Pharmacist and Brother Convicted of \$15M Medic...	November 8, 2024	Criminal and Civil Action
1	Boise Nurse Practitioner Sentenced To 48 Month...	November 7, 2024	Criminal and Civil Action
2	Former Traveling Nurse Pleads Guilty To Tamper...	November 7, 2024	Criminal and Civil Action
3	Former Arlington Resident Sentenced To Prison ...	November 7, 2024	Criminal and Civil Action
4	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024	Criminal and Civil Action
5	Former Licensed Counselor Sentenced For Defrau...	November 6, 2024	Criminal and Civil Action
6	Macomb County Doctor And Pharmacist Agree To P...	November 4, 2024	Criminal and Civil Action
7	Rocky Hill Pharmacy And Its Owners Indicted Fo...	November 4, 2024	Criminal and Civil Action
8	North Texas Medical Center Pays \$14.2 Million ...	November 4, 2024	Criminal and Civil Action
9	New England Doctor Pleads Guilty To Drug Distr...	November 4, 2024	Criminal and Civil Action
10	Attorney General Alan Wilson Announces Upstate...	November 4, 2024	State Enforcement Agency
11	St. Louis County Woman Accused Of \$3 Million H...	November 1, 2024	Criminal and Civil Action
12	Lab Owner And Marketing Company Owner Both Fou...	November 1, 2024	Criminal and Civil Action
13	Compound Ingredient Supplier Medisca Inc., To ...	November 1, 2024	Criminal and Civil Action
14	The New Mexico Department Of Justice Charges F...	November 1, 2024	State Enforcement Agency
15	Nashville Woman Indicted, Charged In TBI Medic...	November 1, 2024	State Enforcement Agency
16	Michael DePalma, MD and Virginia I-Spine Physi...	October 31, 2024	CMP and Affirmative Exec
17	Columbus Doctor, His Clinic Convicted of \$1.5 ...	October 31, 2024	State Enforcement Agency

Title	Date	Category
18 Mercy Health Youngstown Agreed to Pay \$69,000 ...	October 30, 2024	Fraud Self-Disclosures
19 Quincy-Based Physician Group To Pay \$650,000 T...	October 30, 2024	State Enforcement Agency

## Step 2: Making the scraper dynamic

### 1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2) To start, we define a function that takes month and year as input. If the year is before 2013, print a message that only years since are allowed, and then exit the function. Inside the function to initialize, set up an empty list to store the scraped data and a page variable starting at 1.

Use a while loop to keep scraping each page until there are no more enforcement actions. For each page, create a URL based on the page number and make a request to it. Parse the HTML content with BeautifulSoup and find all enforcement action items. If no items are found, stop the loop.

For each action, extract details like title, date, category, and link. Check if the date is within the specified range, and if not, skip it. Then, request the detail page for each action to get the agency name and add this data to the list.

After processing each page, increase the page number by 1 and pause for a second before moving. When finished, turn the list of data into a DataFrame and save it as enforcement\_actions\_year\_month.csv.

- b. Create Dynamic Scraper (PARTNER 2)

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
from urllib.parse import urljoin
from datetime import datetime
import time

# Define the scraper function to dynamically fetch
def scrape_enforcement_actions(start_year, start_month):
    # Base URL and start date for filtering actions
    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    start_date = datetime(start_year, start_month, 1)

    # List to hold scraped data
```

```

data = []
page = 1
keep_scraping = True
detail_page_counter = 0

while keep_scraping:
    # Construct page URL
    url = f"{base_url}?page={page}"

    # Request the page and parse the HTML
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Find enforcement actions on the page
    actions = soup.find_all("div", class_="usa-card__container")

    # Break if no actions are found
    if not actions:
        break

    for action in actions:
        # Extract title, link, date, and category
        title_tag = action.find("h2",
        ↵ class_="usa-card__heading").find("a")
        title = title_tag.text.strip()
        link = urljoin(base_url, title_tag["href"])

        # Parse date and check if it's within the specified range
        date_text = action.find("span", class_="text-base-dark
        ↵ padding-right-105").text.strip()
        date = datetime.strptime(date_text, "%B %d, %Y")
        if date < start_date:
            keep_scraping = False # Stop if the date is before
        ↵ start_date
            break

        category_list = action.find("ul", class_="display-inline
        ↵ add-list-reset")
        category = category_list.find("li").text.strip() if category_list
        ↵ else "Unknown"

        # Visit the detail page to extract agency information

```

```

agency_response = requests.get(link)
agency_soup = BeautifulSoup(agency_response.text, 'html.parser')
agency_tag = agency_soup.find("span", text="Agency:")
agency = agency_tag.find_next_sibling(text=True).strip() if
↪ agency_tag else "Unknown"

# Append data including agency information
data.append({
    "Title": title,
    "Date": date_text,
    "Category": category,
    "Link": link,
    "Agency": agency
})

# Batch delay: Apply delay after every 10 detail page requests
detail_page_counter += 1
if detail_page_counter % 10 == 0:
    time.sleep(1) # Add a delay every 10 detail requests

# Progress update and delay for next page
print(f"Completed scraping page {page}")
page += 1
time.sleep(1) # Delay between main pages

# Convert to DataFrame
df = pd.DataFrame(data)
return df

# Run the scraper for January 2023
df_2023 = scrape_enforcement_actions(2023, 1)

```

Find the earliest date:

```

df_2023 = pd.read_csv("enforcement_actions_2023_01.csv")
earliest_action_2023 = df_2023.iloc[-1] if not df_2023.empty else None

# Display the results for both cases
len_2023 = len(df_2023)
earliest_date_2023 = earliest_action_2023['Date'] if earliest_action_2023 is
↪ not None else None
earliest_details_2023 = earliest_action_2023.to_dict() if
↪ earliest_action_2023 is not None else None

```

```

print(f"Total Records for 2023: {len_2023}")
print(f"Earliest Date in 2023: {earliest_date_2023}")
print("Details of the Earliest Enforcement Action in 2023:")
for key, value in earliest_details_2023.items():
    print(f"  {key}: {value}")

```

Total Records for 2023: 1534  
Earliest Date in 2023: January 3, 2023  
Details of the Earliest Enforcement Action in 2023:  
Title: Podiatrist Pays \$90,000 To Settle False Billing Allegations  
Date: January 3, 2023  
Category: Criminal and Civil Actions  
Link:  
<https://oig.hhs.gov/fraud/enforcement/podiatrist-pays-90000-to-settle-false-billing-allegations>  
Agency: U.S. Attorney's Office, Southern District of Texas

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

```

# Get data since 2021
df_2021 = scrape_enforcement_actions(2021, 1)

```

Find the earliest date:

```

df_2021 = pd.read_csv("enforcement_actions_2021_01.csv")

# To get the earliest date
earliest_action_2021 = df_2021.iloc[-1] if not df_2021.empty else None

# Display the results for both cases
len_2021 = len(df_2021)
earliest_date_2021 = earliest_action_2021['Date'] if earliest_action_2021 is
    not None else None
earliest_details_2021 = earliest_action_2021.to_dict() if
    earliest_action_2021 is not None else None

# Display
print(f"Total Records for 2023: {len_2021}")
print(f"Earliest Date in 2023: {earliest_date_2021}")
print("Details of the Earliest Enforcement Action in 2023:")
for key, value in earliest_details_2021.items():
    print(f"  {key}: {value}")

```

Total Records for 2023: 3022

Earliest Date in 2023: January 4, 2021

Details of the Earliest Enforcement Action in 2023:

Title: 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

Date: January 4, 2021

Category: Criminal and Civil Actions

Link:

<https://oig.hhs.gov/fraud/enforcement/the-united-states-and-tennessee-resolve-claims-with->

Agency: U.S. Attorney's Office, Middle District of Tennessee

### Step 3: Plot data based on scraped data

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

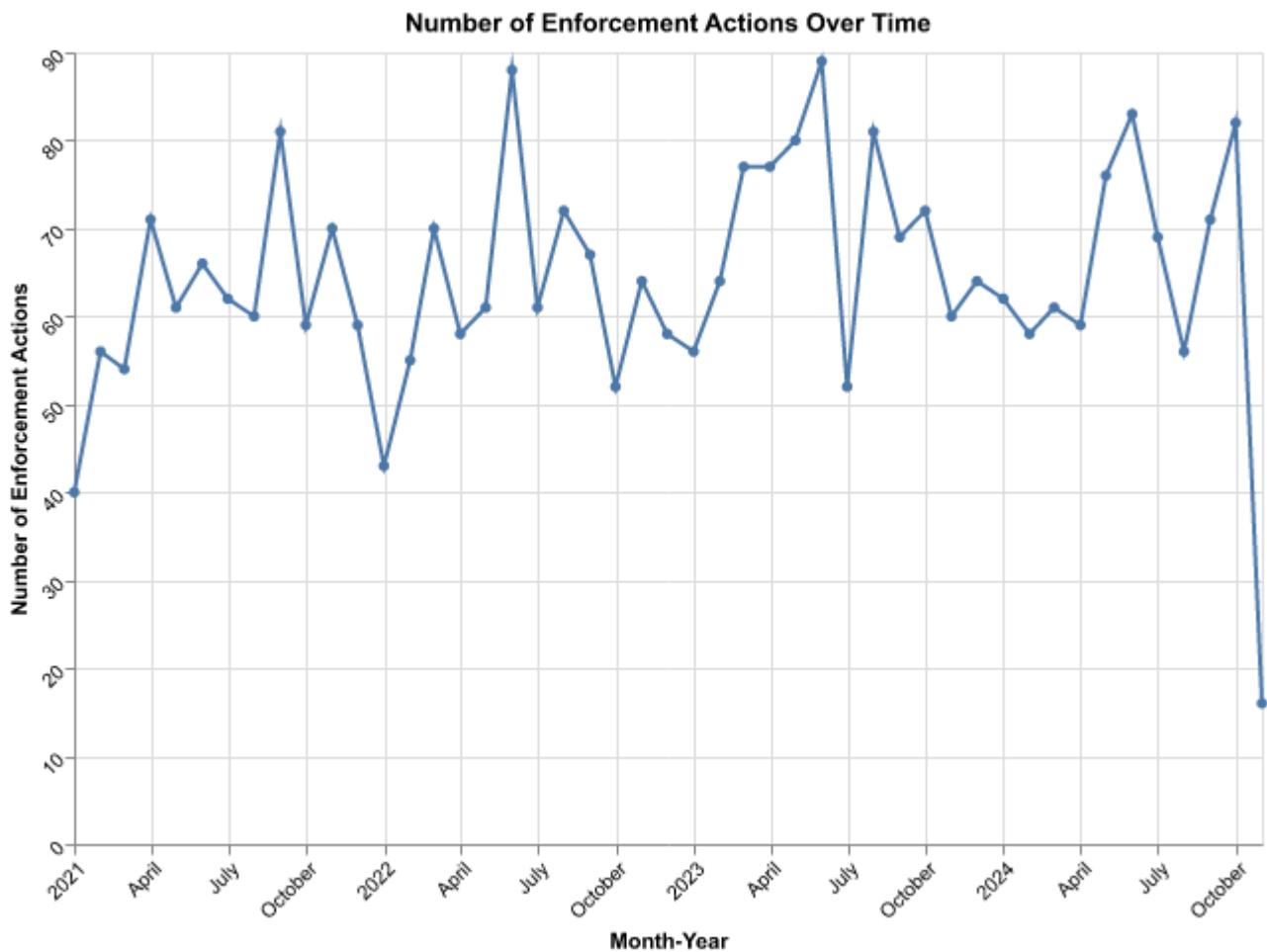
```
import matplotlib.pyplot as plt
import altair as alt
import pandas as pd

# Load and process the data
df_csv_2021 = pd.read_csv("enforcement_actions_2021_01.csv")
df_csv_2021['Date'] = pd.to_datetime(df_csv_2021['Date'], errors='coerce')

# Create 'YearMonth' column and count actions per month
df_csv_2021['YearMonth'] = df_csv_2021['Date'].dt.to_period('M').astype(str)
monthly_counts =
    df_csv_2021.groupby('YearMonth').size().reset_index(name='Enforcement Actions')

# Plot with Altair
chart = alt.Chart(monthly_counts).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Month-Year'),
    y=alt.Y('Enforcement Actions:Q', title='Number of Enforcement Actions'),
    tooltip=['YearMonth:T', 'Enforcement Actions']
).properties(
    title="Number of Enforcement Actions Over Time",
    width=600,
    height=400
).configure_axis(
    labelAngle=-45
)
```

```
chart.show()
```



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

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

```
# Load data
df_csv_2021 = pd.read_csv("enforcement_actions_2021_01.csv")
df_csv_2021['Date'] = pd.to_datetime(df_csv_2021['Date'], errors='coerce')
df_csv_2021.set_index('Date', inplace=True)

def categorize_topic(title):
```

```

title = title.lower()
if 'health care' in title:
    return 'Health Care Fraud'
elif 'financial' in title or 'bank' in title:
    return 'Financial Fraud'
elif 'drug' in title:
    return 'Drug Enforcement'
elif 'bribery' in title or 'corruption' in title:
    return 'Bribery/Corruption'
else:
    return 'Other'

df_csv_2021['Topic'] = df_csv_2021.apply(lambda x:
    categorize_topic(x['Title']) if x['Category'] == 'Criminal and Civil
    Actions' else None, axis=1)

# Split and resample
criminal_civil_df = df_csv_2021[df_csv_2021['Category'] == 'Criminal and
    Civil Actions'].resample('M').size().reset_index(name='Count')
criminal_civil_df['Category'] = 'Criminal and Civil Actions'

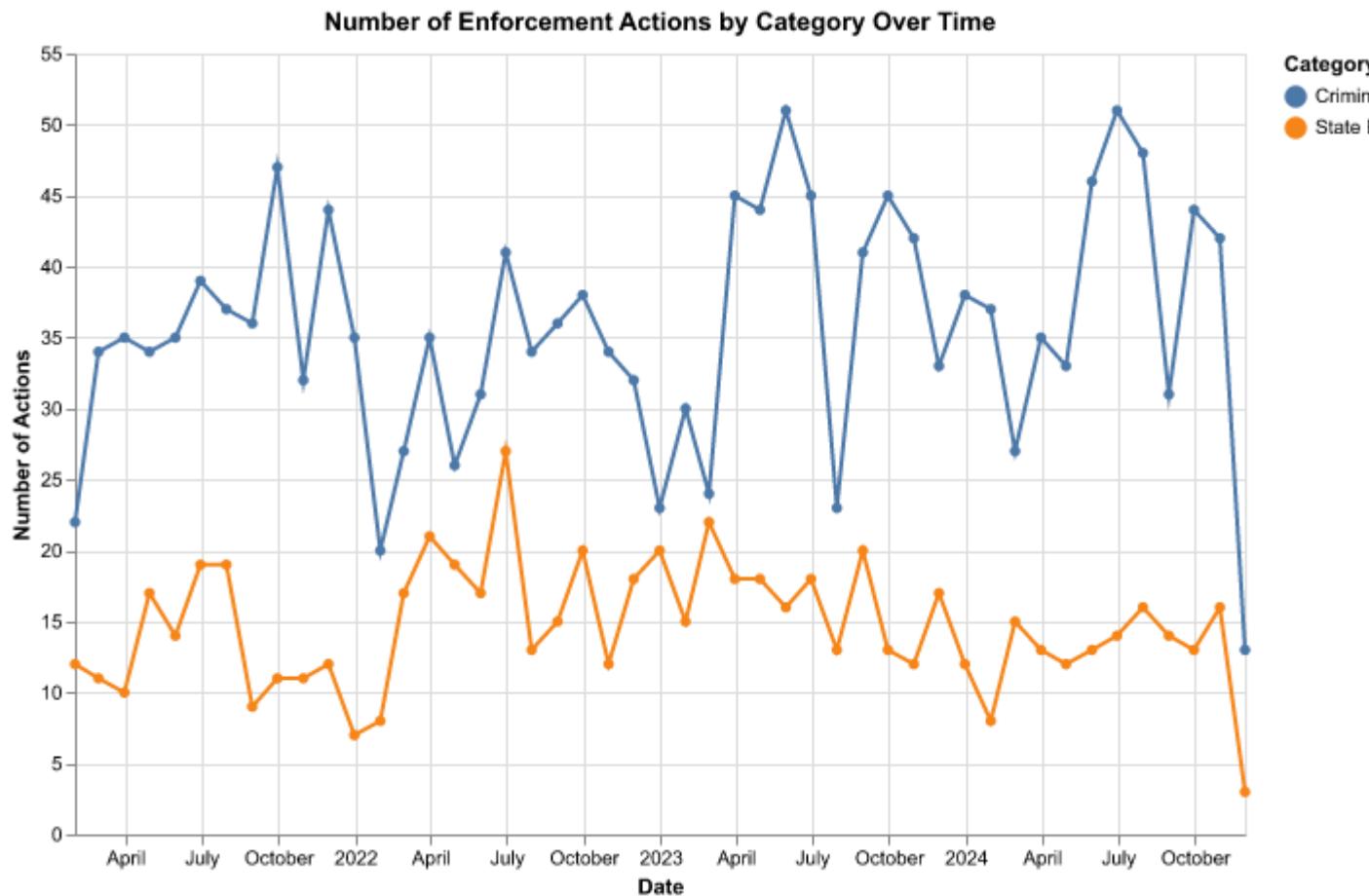
state_enforcement_df = df_csv_2021[df_csv_2021['Category'] == 'State
    Enforcement Agencies'].resample('M').size().reset_index(name='Count')
state_enforcement_df['Category'] = 'State Enforcement Agencies'

# Combine data
combined_df = pd.concat([criminal_civil_df, state_enforcement_df])

# Plot with Altair
chart = alt.Chart(combined_df).mark_line(point=True).encode(
    x=alt.X('Date:T', title='Date'),
    y=alt.Y('Count:Q', title='Number of Actions'),
    color='Category',
    tooltip=['Date:T', 'Count', 'Category']
).properties(
    title="Number of Enforcement Actions by Category Over Time",
    width=600,
    height=400
)

chart.show()

```



- based on five topics

```
import altair as alt
import pandas as pd

# Load and preprocess data
df_csv_2021 = pd.read_csv("enforcement_actions_2021_01.csv")
df_csv_2021['Date'] = pd.to_datetime(df_csv_2021['Date'], errors='coerce')
df_csv_2021.set_index('Date', inplace=True)

def categorize_topic(title):
    title = title.lower()
    if 'health care' in title:
        return 'Health Care Fraud'
    elif 'financial' in title or 'bank' in title:
        return 'Financial Fraud'
```

```

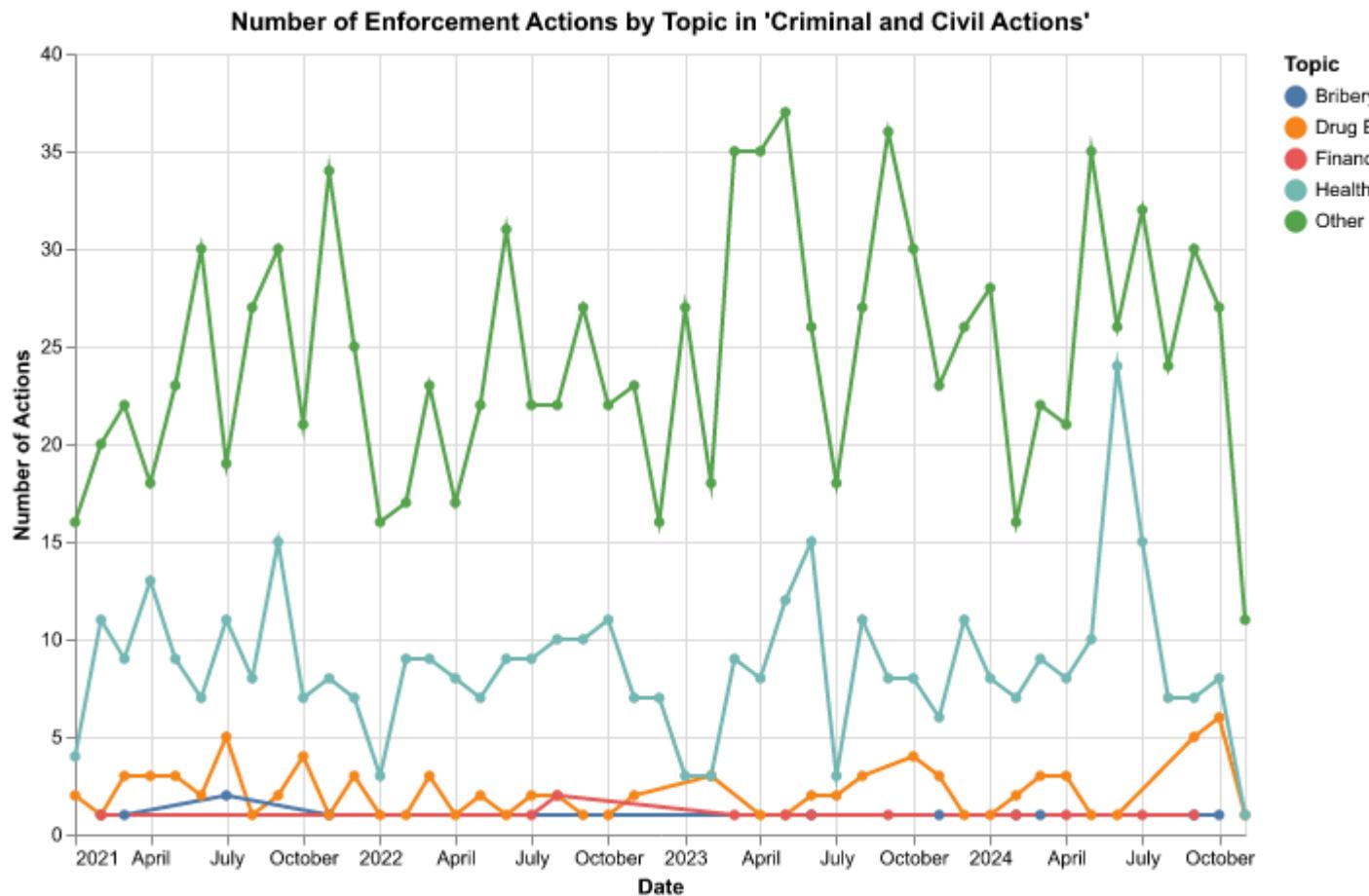
    elif 'drug' in title:
        return 'Drug Enforcement'
    elif 'bribery' in title or 'corruption' in title:
        return 'Bribery/Corruption'
    else:
        return 'Other'

df_csv_2021['Topic'] = df_csv_2021.apply(lambda x:
    categorize_topic(x['Title']) if x['Category'] == 'Criminal and Civil
    Actions' else None, axis=1)

# Filter and group data by month and topic
criminal_civil_df = df_csv_2021[df_csv_2021['Category'] == 'Criminal and
    Civil Actions']
topic_counts =
    criminal_civil_df.groupby([criminal_civil_df.index.to_period('M'),
    'Topic']).size().reset_index(name='Count')
topic_counts['Date'] = topic_counts['Date'].dt.to_timestamp()

# Plot with Altair
chart = alt.Chart(topic_counts).mark_line(point=True).encode(
    x=alt.X('Date:T', title='Date'),
    y=alt.Y('Count:Q', title='Number of Actions'),
    color='Topic',
    tooltip=['Date:T', 'Count', 'Topic']
).properties(
    title="Number of Enforcement Actions by Topic in 'Criminal and Civil
    Actions'",
    width=600,
    height=400
)
chart.show()

```



#### Step 4: Create maps of enforcement activity

##### 1. Map by State (PARTNER 1)

```

import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import re

# Extract state name from the 'Agency' column
def extract_state(agency):
    if pd.notnull(agency):
        match = re.search(r"of\s+(\w+)\$", agency)
        if match:
            return match.group(1)
        else:
            return None
    else:
        return None

```

```

        return match.group(1)
    return None

df_csv_2021['State'] = df_csv_2021['Agency'].apply(extract_state)

# Filter rows with valid state names
state_actions = df_csv_2021.dropna(subset=['State'])

# Count actions by state
state_counts = state_actions['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Count']

# Standardize
state_counts['State'] = state_counts['State'].str.title()

# Load Census state
census_states = gpd.read_file("/Users/georgew/Desktop/Fall 2024/PPHA
    ↴ 30538/ps5/cb_2018_us_state_500k")

# Standardize shapefile state names to title case
census_states['NAME'] = census_states['NAME'].str.title()

# Merge state counts with shapefile data
choropleth_data = census_states.merge(state_counts, left_on='NAME',
    ↴ right_on='State', how='left')

# Fill missing values
choropleth_data['Count'] = choropleth_data['Count'].fillna(0)

# Print states with no data
missing_states = choropleth_data[choropleth_data['Count'] == 0]['NAME']
print("States with no data (potential mismatch):", missing_states.tolist())

# Plot the choropleth map
choropleth_data = choropleth_data.to_crs("EPSG:5070")
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
choropleth_data.plot(
    column='Count',
    cmap='OrRd',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.8',

```

```

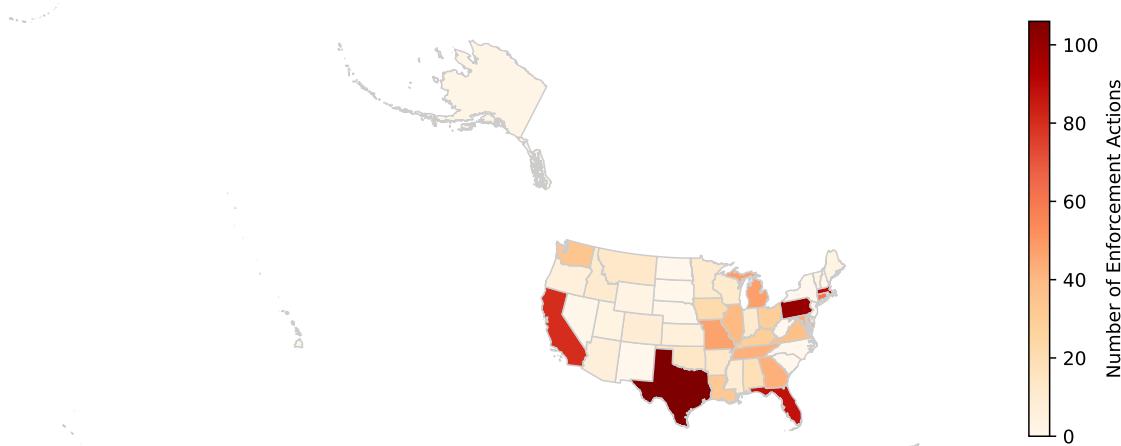
        legend=True,
        legend_kwds={
            'label': "Number of Enforcement Actions",
            'shrink': 0.5
        }
    )

# Customize the plot
ax.set_title("Number of State-Level Enforcement Actions by State")
ax.axis('off')
plt.show()

```

States with no data (potential mismatch): ['North Carolina', 'West Virginia', 'New Mexico', 'Puerto Rico', 'South Dakota', 'New York', 'South Carolina', 'New Hampshire', 'District Of Columbia', 'American Samoa', 'United States Virgin Islands', 'New Jersey', 'Guam', 'Commonwealth Of The Northern Mariana Islands', 'Rhode Island', 'North Dakota']

Number of State-Level Enforcement Actions by State



## 2. Map by District (PARTNER 2)

```

import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import re

```

```

# Define paths for the US Attorney Districts
district_shapefile_path = "/Users/georgew/Desktop/Fall 2024/PPHA 30538/ps5/US
→ Attorney Districts Shapefile simplified_20241109"

# Define the function to extract district
def extract_district(agency):
    if pd.notnull(agency) and "U.S. Attorney's Office" in agency:
        # Capture district name after "U.S. Attorney's Office"
        match = re.search(r"U\.S\. Attorney's Office, ((?:[A-Za-z]+
→ )?District of [A-Za-z\s]+)", agency)
        if match:
            return match.group(1).strip()
    return None

# Apply the function to extract
df_csv_2021['District'] = df_csv_2021['Agency'].apply(extract_district)

# Drop rows without a valid district name
district_actions = df_csv_2021.dropna(subset=['District'])

# Count the number of enforcement actions per district
district_counts = district_actions['District'].value_counts().reset_index()
district_counts.columns = ['District', 'Count']

# Standardize
district_counts['District'] = district_counts['District'].str.title()

# Load the US Attorney District shapefile
district_shapefile = gpd.read_file(district_shapefile_path)

# Standardize district names in the shapefile
district_shapefile['District'] = district_shapefile['judicial_d'].str.title()

# Merge the district counts with the shapefile
choropleth_data = district_shapefile.merge(district_counts, on='District',
→ how='left')

# Fill missing count values
choropleth_data['Count'] = choropleth_data['Count'].fillna(0)

# Plot

```

```

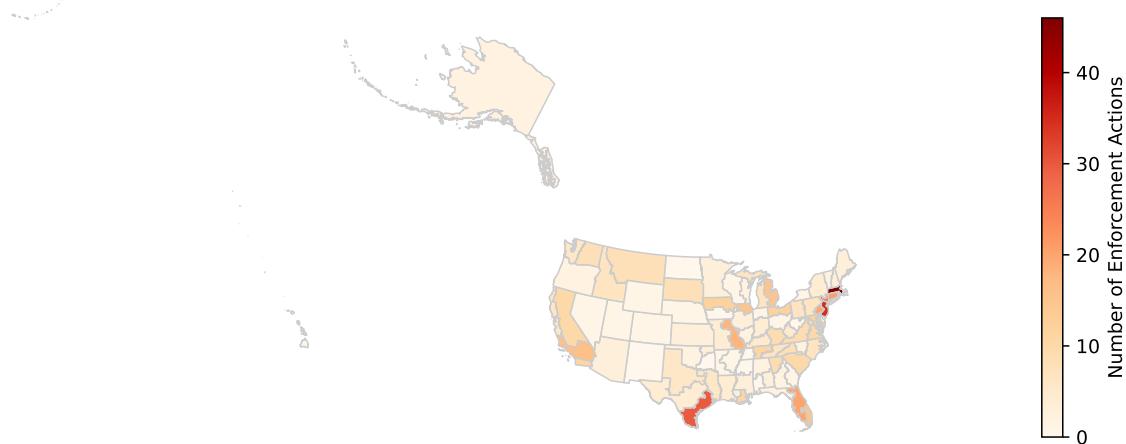
choropleth_data = choropleth_data.to_crs("EPSG:5070")

# Plot with Albers USA projection
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
choropleth_data.plot(
    column='Count',
    cmap='OrRd',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.8',
    legend=True,
    legend_kwds={
        'label': "Number of Enforcement Actions",
        'shrink': 0.5
    }
)

# Customize the plot
ax.set_title("Number of Enforcement Actions by U.S. Attorney District")
ax.axis('off')
plt.show()

```

Number of Enforcement Actions by U.S. Attorney District



## Extra Credit

### 1. Merge zip code shapefile with population

```
import geopandas as gpd
import pandas as pd

# Load ZIP code shapefile
zip_shapefile_path = "/Users/georgew/Desktop/Fall 2024/PPHA
↪ 30538/ps5/cb_2018_us_zcta510_500k/cb_2018_us_zcta510_500k.shp"
zip_shapefile = gpd.read_file(zip_shapefile_path)

# Load population data CSV
population_data_path = "/Users/georgew/Desktop/Fall 2024/PPHA
↪ 30538/ps5/DECENNIALDHC2020.P1_2024-11-09T180943/DECENNIALDHC2020.P1-Data.csv"
population_data = pd.read_csv(population_data_path)

# Extract 5-digit ZIP codes
population_data['ZIP'] = population_data['NAME'].str.replace('ZCTA5 ',
↪ '').str.zfill(5)
zip_shapefile['ZIP'] = zip_shapefile['ZCTA5CE10'].astype(str).str.zfill(5)

# Merge shapefile with population data on ZIP code
zip_population = zip_shapefile.merge(population_data[['ZIP', 'P1_001N']],
↪ on='ZIP', how='left')

# Fill missing population values with 0
zip_population['P1_001N'] = zip_population['P1_001N'].fillna(0)

# Preview merged data
print(zip_population.head())
```

```
ZCTA5CE10      AFFGEOID10  GEOID10      ALAND10  AWATER10 \
0    36083  8600000US36083    36083  659750662  5522919
1    35441  8600000US35441    35441  172850429  8749105
2    35051  8600000US35051    35051  280236456  5427285
3    35121  8600000US35121    35121  372736030  5349303
4    35058  8600000US35058    35058  178039922  3109259
```

```
geometry      ZIP P1_001N
0  MULTIPOLYGON (((-85.63225 32.28098, -85.62439 ... 36083  9367
```

```

1 MULTIPOLYGON (((-87.83287 32.84437, -87.83184 ... 35441 1077
2 POLYGON ((-86.74384 33.25002, -86.73802 33.251... 35051 9173
3 POLYGON ((-86.58527 33.94743, -86.58033 33.948... 35121 16016
4 MULTIPOLYGON (((-86.87884 34.21196, -86.87649 ... 35058 11098

```

## 2. Conduct spatial join

```

import geopandas as gpd

# Load the district shapefile
district_shapefile_path = "/Users/georgew/Desktop/Fall 2024/PPHA 30538/ps5/US
    ↪ Attorney Districts Shapefile simplified_20241109"
district_shapefile = gpd.read_file(district_shapefile_path)

# Ensure both GeoDataFrames use the same CRS
zip_population = zip_population.to_crs(district_shapefile.crs)

# Conduct the spatial join
zip_district_join = gpd.sjoin(zip_population, district_shapefile,
    ↪ how="inner", predicate="intersects")

# Preview the result of the spatial join
print(zip_district_join.head())

```

	ZCTA5CE10	AFFGEOID10	GEOID10	ALAND10	AWATER10	geometry	ZIP	P1_001N
0	36083	8600000US36083	36083	659750662	5522919	MULTIPOLYGON (((-85.63225 32.28098, -85.62439 ...	36083	9367
1	35441	8600000US35441	35441	172850429	8749105	MULTIPOLYGON (((-87.83287 32.84437, -87.83184 ...	35441	1077
1	35441	8600000US35441	35441	172850429	8749105	MULTIPOLYGON (((-87.83287 32.84437, -87.83184 ...	35441	1077
2	35051	8600000US35051	35051	280236456	5427285	POLYGON ((-86.74384 33.25002, -86.73802 33.251...)	35051	9173
3	35121	8600000US35121	35121	372736030	5349303	POLYGON ((-86.58527 33.94743, -86.58033 33.948...)	35121	16016
						index_right statefp ... state chief_judg nominating		
0		3	01 ... Alabama	Emily Coody Marks	Donald Trump (R)			

```

1          4    01 ... Alabama      Kristi DuBose  George W. Bush (R)
1          80   01 ... Alabama     Scott Coogler  George W. Bush (R)
2          80   01 ... Alabama     Scott Coogler  George W. Bush (R)
3          80   01 ... Alabama     Scott Coogler  George W. Bush (R)

  term_as_ch shape_leng shape_area  abbr  district_n    shape_are  \
0    2019.0   10.235799  3.858442  ALM       11  5.645450e+10
1    2017.0   12.976906  3.278871  ALS       11  4.772733e+10
1    2020.0   12.563893  5.754560  ALN       11  8.593125e+10
2    2020.0   12.563893  5.754560  ALN       11  8.593125e+10
3    2020.0   12.563893  5.754560  ALN       11  8.593125e+10

  shape_len
0  1.236201e+06
1  1.567095e+06
1  1.529854e+06
2  1.529854e+06
3  1.529854e+06

[5 rows x 23 columns]

```

### 3. Map the action ratio in each district

```

import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import re

# Convert 'P1_001N' to numeric, coercing errors to NaN and then filling with
# 0
zip_district_join['P1_001N'] = pd.to_numeric(zip_district_join['P1_001N'],
                                             errors='coerce').fillna(0)

# Aggregate population by district
district_population =
    zip_district_join.groupby('judicial_d')['P1_001N'].sum().reset_index()
district_population.columns = ['judicial_d', 'Total_Population']

# Load and filter
enforcement_data_path = "enforcement_actions_2021_01.csv"

```

```

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

# Extract district names
enforcement_data['District'] =
    enforcement_data['Agency'].apply(extract_district)
district_actions =
    enforcement_data.dropna(subset=['District']).groupby('District').size().reset_index(name='Actions')

# Merge population and action counts
district_data = district_population.merge(district_actions,
                                           left_on='judicial_d', right_on='District', how='left')
district_data['Action_Count'] = district_data['Action_Count'].fillna(0)
district_data['Enforcement_Ratio'] = district_data['Action_Count'] /
    district_data['Total_Population']

# Merge with district shapefile for plotting
choropleth_data = district_shapefile.merge(district_data,
                                             left_on='judicial_d', right_on='judicial_d', how='left')

# Change projection
choropleth_data = choropleth_data.to_crs("EPSG:5070")

# Change maximum
norm = mcolors.Normalize(vmin=0, vmax=0.00001)

# Plot the choropleth map
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
choropleth_data.plot(
    column='Enforcement_Ratio',
    cmap='Reds',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.8',
    legend=True,
    norm=norm,
    legend_kwds={'label': "Enforcement Actions per Capita"})
)

ax.set_title("Ratio of Enforcement Actions to Population by U.S. Attorney
             District")

```

```
ax.axis('off')  
  
plt.show()
```

