\RecustomVerbatimEnvironment{verbatim}{Verbatim}{ showspaces = false, showtabs = false, breaksymbolleft={}, breaklines }

PS5

AUTHOR
Kohan Chen, Katherine Tu

PUBLISHED
November 7, 2024

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): Kohan Chen, kohanchen
    - Partner 2 (name and cnet ID): Katherine Tu, katherinetu
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: **KC KT**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **KC KT**" (1 point)
6. Late coins used this pset: **0** 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
```

```
url = "https://oig.hhs.gov/fraud/enforcement/"

response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')

titles = []
dates = []
categories = []
links = []

entries = soup.find_all('h2', class_='usa-card__heading')

for entry in entries:
    link = entry.find('a')
    if link:
        titles.append(link.text.strip())
        links.append('https://oig.hhs.gov' + link['href'])

    div = entry.find_parent('div')
    if div:
        date = div.find('span', class_='text-base-dark padding-right-105')
        dates.append(date.text.strip() if date else 'No date found')

        category = div.find('li', class_='display-inline-block usa-tag text-no-lowerca
        categories.append(category.text.strip() if category else 'No category found')

print(f"Lengths - Titles: {len(titles)}, Dates: {len(dates)}, Categories: {len(categor


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

df.head()
```

```
Lengths - Titles: 20, Dates: 20, Categories: 20, Links: 20
```

	Title	Date	Category	Link
0	Pharmacist and Brother Convicted of $15M Medic...	November 8, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/pharmaci...
1	Boise Nurse Practitioner Sentenced To 48 Month...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/boise-nu...
2	Former Traveling Nurse Pleads Guilty To Tamper...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/former-t...
3	Former Arlington Resident Sentenced To Prison ...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/former-a...

	Title	Date	Category	Link
4	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/paroled-...

2. Crawling (PARTNER 1)

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

url = "https://oig.hhs.gov/fraud/enforcement/"

response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')

titles = []
dates = []
categories = []
links = []
agencies = []

for entry in entries:
    link = entry.find('a')
    if link:
        titles.append(link.text.strip())
        full_link = 'https://oig.hhs.gov' + link['href']
        links.append(full_link)

        try:
            response = requests.get(full_link)
            detailed_soup = BeautifulSoup(response.content, "html.parser")

            detail_div = detailed_soup.find('div', class_='margin-top-5 padding-y-3 bo
            if detail_div:
                all_li = detail_div.find_all('li')
                agency_found = False
                for li in all_li:
                    span = li.find('span')
                    if span and 'Agency:' in span.text:
                        agencies.append(li.text.replace('Agency:', '').strip())
                        agency_found = True
                        break
                if not agency_found:
                    agencies.append('No agency found')
        except Exception as e:
            print(f"Error processing {full_link}: {str(e)}")
            agencies.append('Error retrieving agency')

    div = entry.find_parent('div')
    if div:
        date = div.find('span', class_='text-base-dark padding-right-105')
        dates.append(date.text.strip() if date else 'No date found')
```

```
        category = div.find('li', class_='display-inline-block usa-tag text-no-lowerca
        categories.append(category.text.strip() if category else 'No category found')

print(f"Lengths - Titles: {len(titles)}, Dates: {len(dates)}, Categories: {len(categor


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

df.head()
```

```
Lengths - Titles: 20, Dates: 20, Categories: 20, Links: 20
```

	Title	Date	Category	Link	Agency
0	Pharmacist and Brother Convicted of $15M Medic...	November 8, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/pharmaci...	U.S. Department of Justice
1	Boise Nurse Practitioner Sentenced To 48 Month...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/boise-nu...	November 7, 2024; U.S. Attorney's Office, Dist...
2	Former Traveling Nurse Pleads Guilty To Tamper...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/former-t...	U.S. Attorney's Office, District of Massachusetts
3	Former Arlington Resident Sentenced To Prison ...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/former-a...	U.S. Attorney's Office, Eastern District of Vi...
4	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/paroled-...	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)

```
def scrape_enforcement_actions(year, month):
    """
    Pseudo-code for scraping enforcement actions from a given month/year to present

    Parameters:
    year (int): Starting year to scrape from
    month (int): Starting month to scrape from
    """
```

FUNCTION scrape_enforcement_actions(year, month):

1. INPUT VALIDATION IF year is less than 2013 Print “Please enter a year >= 2013” Exit function
2. SETUP Create empty lists for storing:
    - titles
    - dates
    - categories
    - links
    - agencies Set base URL to HHS OIG enforcement page
3. PAGINATION LOOP (WHILE loop) WHILE there are more pages to process: Get current page content FOR each enforcement action on page: Extract title and link Get agency from detailed page Get date and category Store all information in lists Wait 1 second before next action

    Look for “next page” link IF found: Update URL to next page Wait 1 second before next page ELSE: Exit WHILE loop
4. DATA PROCESSING Create DataFrame from collected lists Convert dates to proper format Filter to keep only entries >= input month/year
5. SAVE RESULTS Create filename: “enforcement_actions_[year]_[month].csv” Save DataFrame to CSV file
6. OUTPUT Return the filtered DataFrame

- b. Create Dynamic Scraper (PARTNER 2)

I will use ThreadPoolExecutor to get data much more faster than the regular process. And I used several strategies to avoid frequent timeout and connection errors and consistent failures on specific pages (like pages 192 and 193).

```
import pandas as pd
import requests
from bs4 import BeautifulSoup
from concurrent.futures import ThreadPoolExecutor, as_completed
import random
import time
import sys

# add restriction
sys.setrecursionlimit(10000)
```

```
def fetch_detailed_info(link, session=None, max_retries=3):
    if session is None:
        session = requests.Session()
        session.headers.update({
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3
        })

    for attempt in range(max_retries):
        try:
            response = session.get(link, timeout=10)
            if response.status_code == 200:
                detailed_soup = BeautifulSoup(response.content, "html.parser")
                agency = 'No agency found'
                detail_div = detailed_soup.find('div', class_='margin-top-5 padding-y-
                if detail_div:
                    all_li = detail_div.find_all('li')
                    for li in all_li:
                        span = li.find('span')
                        if span and 'Agency:' in span.text:
                            agency = li.text.replace('Agency:', '').strip()
                            break
                return agency

            if attempt < max_retries - 1:
                time.sleep(1)

        except Exception as e:
            if attempt < max_retries - 1:
                time.sleep(1)
            else:
                print(f"Error processing {link}: {str(e)}")
    return 'Error retrieving agency'

def fetch_page_data(page_num, session=None, max_retries=3):
    if session is None:
        session = requests.Session()
        session.headers.update({
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3
        })

    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    for attempt in range(max_retries):
        try:
            current_url = f"{base_url}?page={page_num}" if page_num > 1 else base_url
            response = session.get(current_url, timeout=10)

            if response.status_code == 200:
                soup = BeautifulSoup(response.content, 'html.parser')
                entries = soup.find_all('h2', class_='usa-card__heading')

                page_data = []
                for entry in entries:
                    link = entry.find('a')
```

```
                    if link:
                        title = link.text.strip()
                        full_link = 'https://oig.hhs.gov' + link['href']
                        div = entry.find_parent('div')
                        if div:
                            date = div.find('span', class_='text-base-dark padding-rig
                            date_text = date.text.strip() if date else 'No date found'
                            category_tags = div.find_all('li', class_='display-inline-
                            entry_categories = [cat.text.strip() for cat in category_t
                            page_data.append({
                                'Title': title,
                                'Date': date_text,
                                'Categories': entry_categories,
                                'Link': full_link
                            })
                return page_data if page_data else None

            if attempt < max_retries - 1:
                time.sleep(1)

        except Exception as e:
            if attempt < max_retries - 1:
                time.sleep(1)
            else:
                print(f"Error on page {page_num}: {str(e)}")
    return None

def scrape_enforcement_actions(year, month, max_pages=482):
    if year < 2013:
        print("Please enter a year >= 2013")
        return None

    data_list = []
    session = requests.Session()
    session.headers.update({
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
    })

    # Stage One: quickly fetch and read the page
    print("Fetching pages...")
    failed_pages = set()

    with ThreadPoolExecutor(max_workers=8) as executor:
        future_to_page = {
            executor.submit(fetch_page_data, page, session): page
            for page in range(1, max_pages + 1)
        }

        for future in as_completed(future_to_page):
            page = future_to_page[future]
            try:
                page_data = future.result()
                if page_data:
                    data_list.extend(page_data)
```

```
                print(f"Added data from page {page}")
            else:
                failed_pages.add(page)
        except Exception as e:
            failed_pages.add(page)
            print(f"Error on page {page}: {str(e)}")

# Stage Two: Dealing with failed pages only
if failed_pages:
    print(f"\nRetrying {len(failed_pages)} failed pages...")
    for page in sorted(failed_pages):
        try:
            page_data = fetch_page_data(page, session, max_retries=5)
            if page_data:
                data_list.extend(page_data)
                failed_pages.remove(page)
                print(f"Successfully retrieved page {page} in retry")
        except Exception as e:
            print(f"Failed to retrieve page {page} in retry: {str(e)}")

print(f"\nCollected {len(data_list)} total entries")

# Stage three: quickly fetching agency information
print("Fetching agency information...")
with ThreadPoolExecutor(max_workers=10) as executor:
    future_to_entry = {
        executor.submit(fetch_detailed_info, entry['Link'], session): i
        for i, entry in enumerate(data_list)
    }

    completed = 0
    total = len(data_list)

    for future in as_completed(future_to_entry):
        try:
            idx = future_to_entry[future]
            agency = future.result()
            data_list[idx]['Agency'] = agency
            completed += 1
            if completed % 20 == 0:
                print(f"Processed {completed}/{total} entries")
        except Exception as e:
            print(f"Error processing agency info: {str(e)}")

# Create separate rows version
rows_separate = []
for entry in data_list:
    for category in entry['Categories']:
        rows_separate.append({
            'Title': entry['Title'],
            'Date': entry['Date'],
            'Category': category,
            'Link': entry['Link'],
            'Agency': entry.get('Agency', 'Error retrieving agency')
```

```
            })

    # Create combined categories version
    rows_combined = []
    for entry in data_list:
        rows_combined.append({
            'Title': entry['Title'],
            'Date': entry['Date'],
            'Category': ' and '.join(entry['Categories']),
            'Link': entry['Link'],
            'Agency': entry.get('Agency', 'Error retrieving agency')
        })

    # Create DataFrames
    df_separate = pd.DataFrame(rows_separate)
    df_combined = pd.DataFrame(rows_combined)

    # Convert date and filter
    df_separate['Date'] = pd.to_datetime(df_separate['Date'], errors='coerce')
    df_separate = df_separate[df_separate['Date'] >= f"{year}-{month:02d}-01"]

    df_combined['Date'] = pd.to_datetime(df_combined['Date'], errors='coerce')
    df_combined = df_combined[df_combined['Date'] >= f"{year}-{month:02d}-01"]

    # Save to CSV
    filename_separate = f"enforcement_actions_{year}_{month}_separate.csv"
    filename_combined = f"enforcement_actions_{year}_{month}_combined.csv"

    df_separate.to_csv(filename_separate, index=False)
    df_combined.to_csv(filename_combined, index=False)

    print(f"\nSaved separate categories version to: {filename_separate}")
    print(f"Saved combined categories version to: {filename_combined}")

    return df_combined
```

```
#scrape data from year 2023 January
if __name__ == "__main__":
    try:
        df = scrape_enforcement_actions(2023, 1)
        if df is not None:
            print(f"\nTotal number of enforcement actions: {len(df)}")
            print("\nEarliest enforcement action:")
            earliest = df.sort_values('Date').iloc[0]
            print(f"Date: {earliest['Date']}")
            print(f"Title: {earliest['Title']}")
            print(f"Agency: {earliest['Agency']}")
            print(f"Categories: {earliest['Category']}")
    except Exception as e:
        print(f"Error running scraper: {str(e)}")
```

Total number of enforcement actions: 1534 Earliest enforcement action: Date: 2023-01-03 00:00:00 Title: Podiatrist Pays $90,000 To Settle False Billing Allegation Agency: U.S. Attorney's Office,

Southern District of Texas Categories: Criminal and Civil Actions

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

```
# Run scraper from January 2021
if __name__ == "__main__":
    try:
        df = scrape_enforcement_actions(2021, 1)
        if df is not None:
            print(f"\nTotal number of enforcement actions: {len(df)}")
            print("\nEarliest enforcement action:")
            earliest = df.sort_values('Date').iloc[0]
            print(f"Date: {earliest['Date']}")
            print(f"Title: {earliest['Title']}")
            print(f"Agency: {earliest['Agency']}")
            print(f"Categories: {earliest['Category']}")
    except Exception as e:
        print(f"Error running scraper: {str(e)}")
```

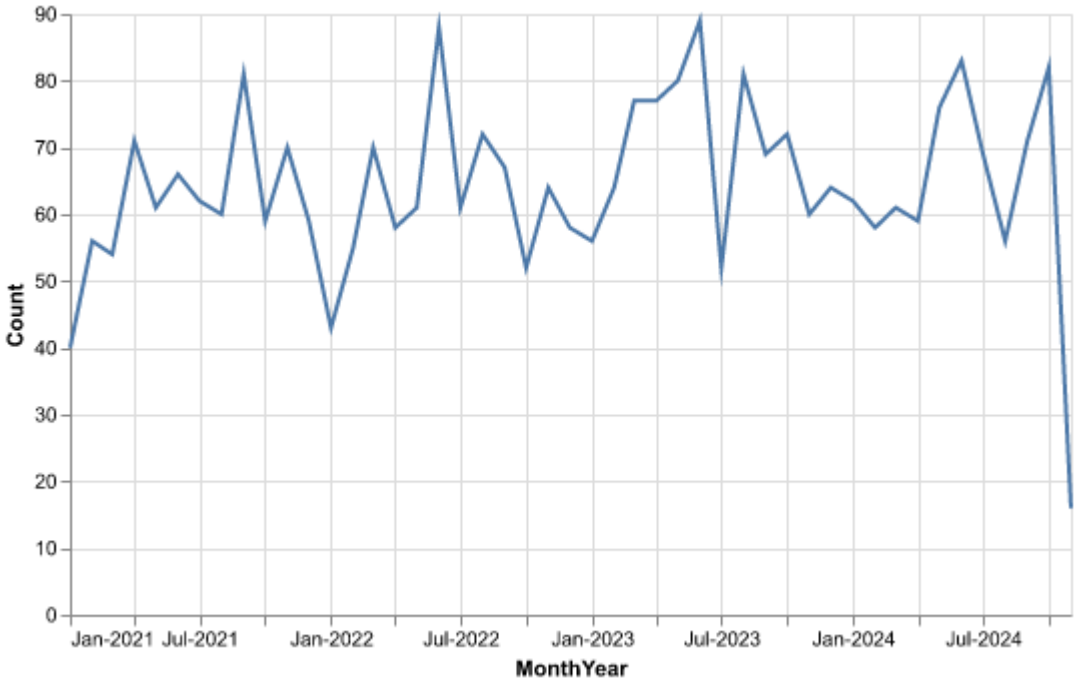
Total number of enforcement actions:1534 Earliest enforcement action: Date: 2021-01-04 00:00:00 Title: The United States And Tennessee Resolve Claims With Three Providers For False Claims Act Liability Relating To 'P-Stim' Devices For A Total Of $1.72 Million Agency: U.S. Attorney's Office, Middle District of Tennessee Categories: Criminal and Civil Actions

Step 3: Plot data based on scraped data

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

```
#Load data
df_2021 = pd.read_csv('/Users/katherinetu/Desktop/PS5/enforcement_actions_2021_1_combi

#Ensure it is datetime formate
df_2021["Date"] = pd.to_datetime(df_2021["Date"])

#Aggregate to month+year and count occurrences
df_2021["MonthYear"] = df_2021["Date"].dt.to_period("M")
df_2021["MonthYear"] = df_2021["MonthYear"].dt.strftime("%b-%Y")
monthly_counts = df_2021.groupby("MonthYear").size().reset_index(name = "Count")
```

```
#Create Altair Graph
count_2021 = alt.Chart(monthly_counts).mark_line().encode(
    alt.X("MonthYear:T", axis=alt.Axis(format='%b-%Y')),
    alt.Y("Count:Q")
).properties(
    width = 500
)

count_2021
```

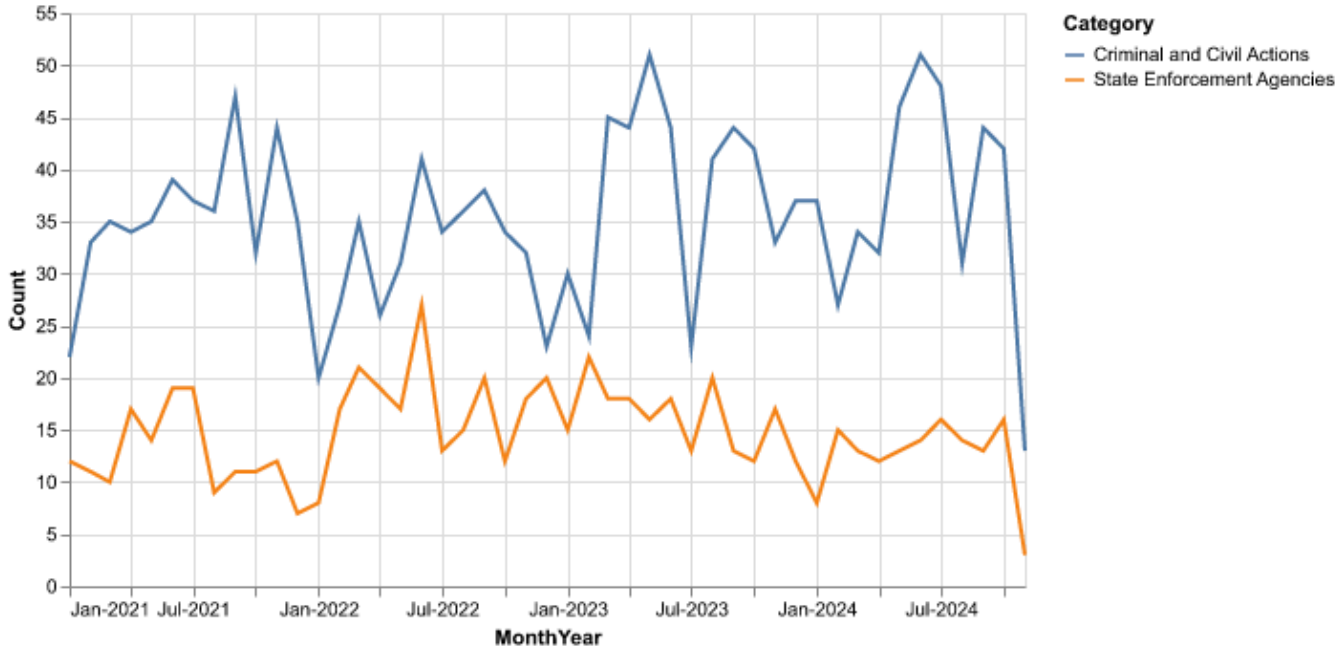


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

```
#The number of enforcements split by "Criminal and Civil Actions" & "State Enforcement
df_category = df_2021.groupby(["MonthYear","Category"]).size().reset_index(name = "Cou
df_cc_sea = df_category[df_category["Category"].isin(["Criminal and Civil Actions","St
```

```
#plot a line graph of the two lines
chart_cc_sea = alt.Chart(df_cc_sea).mark_line().encode(
    alt.X("MonthYear:T", axis=alt.Axis(format='%b-%Y')),
    alt.Y("Count:Q"),
    alt.Color("Category:N")
).properties(
    width = 500
)

chart_cc_sea
```



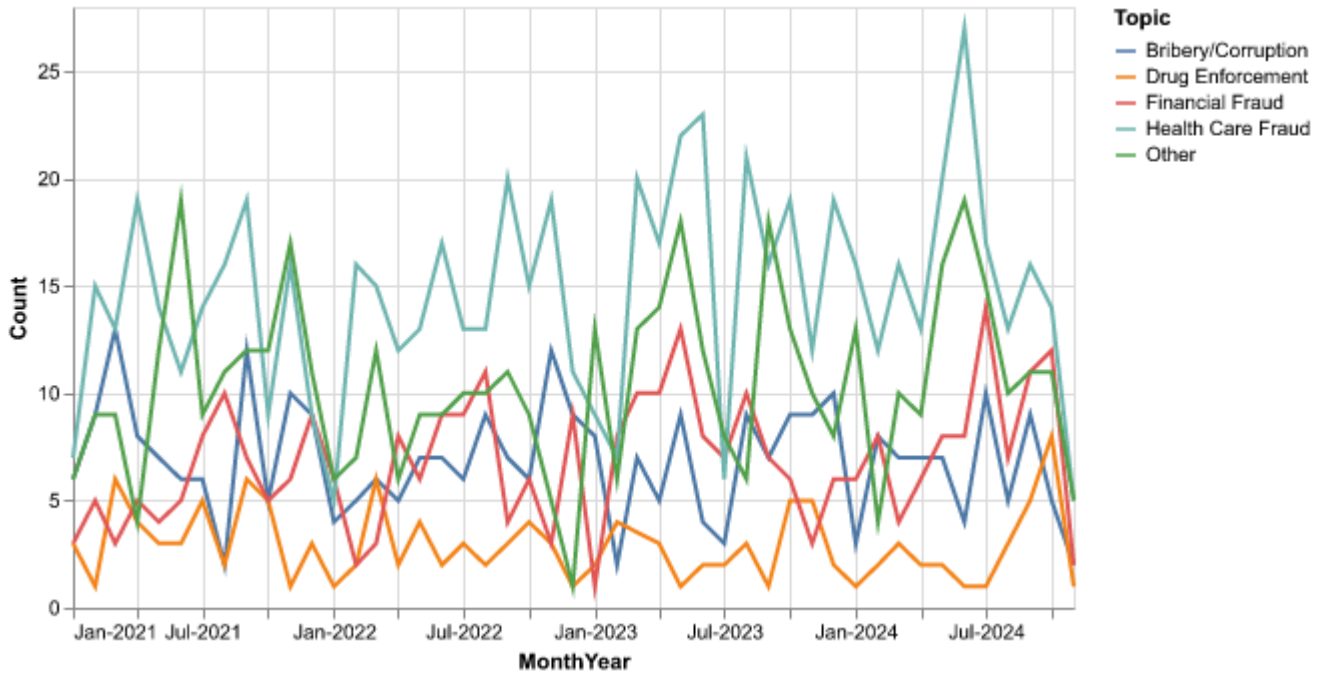
- based on five topics

```
#filter dataframe to only contain "criminal and civil actions" data
df_cc = df_2021[df_2021["Category"].isin(["Criminal and Civil Actions"])]
```

```
#write a function to assign topics according to key words in the titles
def assign_topic(df):
    #Iterate over each row and create a topic list to account for multiple topics
    rows = []
    for _, row in df.iterrows():
        title = row['Title']
        title_lower = title.lower()
        topics = []

    # Define the conditions and corresponding categories
        if any(keyword in title_lower for keyword in ["health care","medicare", "medic
            topics.append("Health Care Fraud")
        if any(keyword in title_lower for keyword in ["false claim","tax"]):
            topics.append("Financial Fraud")
        if any(keyword in title_lower for keyword in ["drug","drugs","pill", "pills"])
            topics.append("Drug Enforcement")
        if any(keyword in title_lower for keyword in ["kickback","bribery","conspiracy
            topics.append("Bribery/Corruption")
        if not topics:
            topics.append("Other")

    #Add each topic as a separate row
        for topic in topics:
            row_data = row.to_dict()
            row_data["Topic"] = topic
            rows.append(row_data)

    return pd.DataFrame(rows)
```

```
#apply it to the Criminal and Civil Actions Category
df_5topics = assign_topic(df_cc)

#aggregate the number of actions by monthyear and by the 5 topics
df_5topics_count = df_5topics.groupby(["MonthYear","Topic"]).size().reset_index(name =
```

```
#Make a line graph
chart_5topics = alt.Chart(df_5topics_count).mark_line().encode(
    alt.X("MonthYear:T", axis=alt.Axis(format='%b-%Y')),
    alt.Y("Count:Q"),
    alt.Color("Topic:N"),
    alt.Tooltip(["MonthYear:T","Count:Q"])
).properties(
    width = 500
)

chart_5topics
```



Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

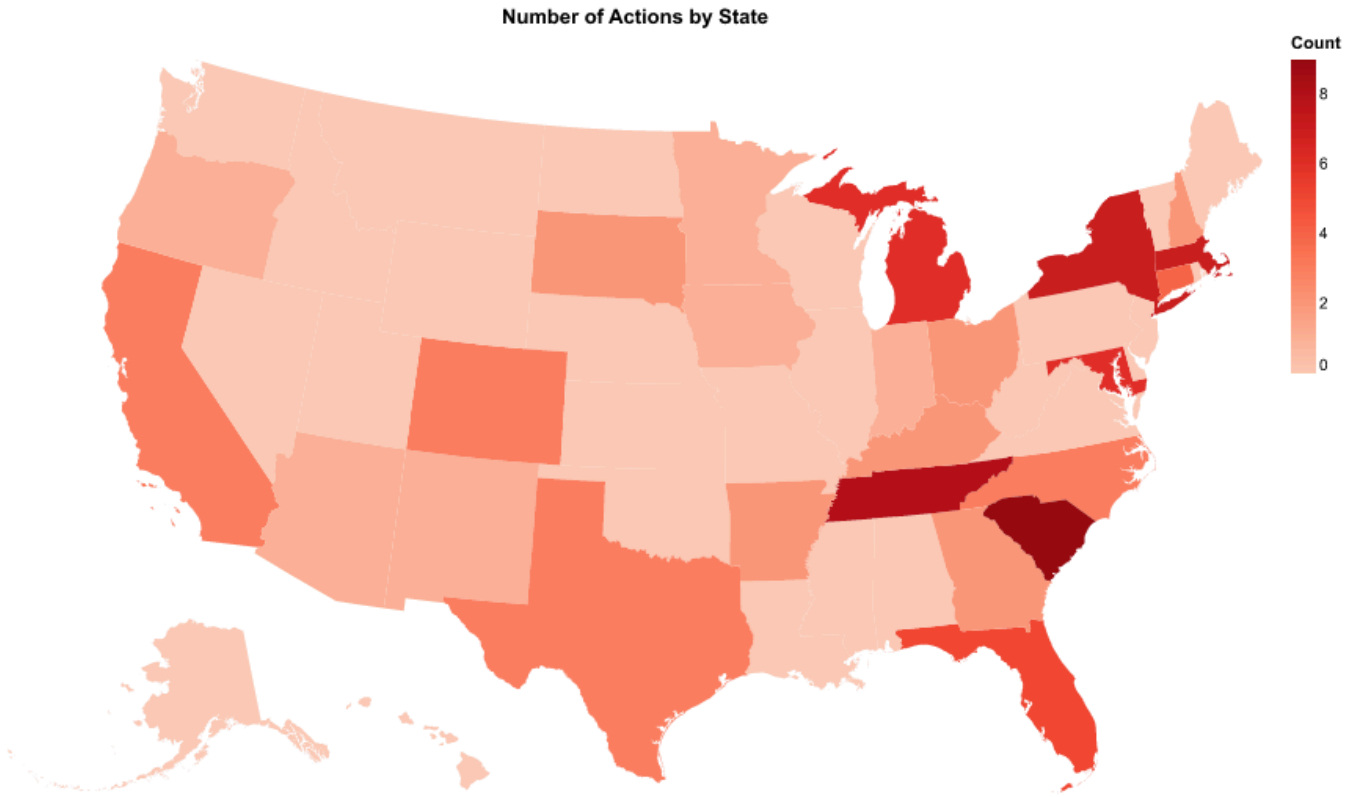
```
import geopandas as gpd
import matplotlib.pyplot as plt
```

```
#Filter the dataset to include only state agency actions
df_state = df_2021[df_2021["Agency"].str.contains("State of", na = False)]

#Clean the dataset to include only state names
df_state["Agency"] = df_state["Agency"].str.replace(r"(?i)state of ", "", regex=True).
```

```
#Find the actions taken by state
df_state_count = df_state.groupby("Agency").size().reset_index(name = "Count")
```

```
#Read the shapefile
state_shp = gpd.read_file("/Users/katherinetu/Desktop/PS5/cb_2018_us_state_500k/cb_201

#Merge the count data
state_count_shp = state_shp.merge(df_state_count,
                                  left_on="NAME",
                                  right_on="Agency",
                                  how="left")

state_count_shp["Count"] = state_count_shp["Count"].fillna(0)
```

```
#make a map
chart_state_count = alt.Chart(state_count_shp).mark_geoshape().encode(
    alt.Color('Count:Q', scale=alt.Scale(scheme='Reds')),
    tooltip=['State:N', 'Count:Q']
).project(
    type='albersUsa'  # Apply Albers USA projection
).properties(
    width=800,  # Adjust the width as needed
    height=500,  # Adjust the height as needed
    title="Number of Actions by State"
)

chart_state_count
```



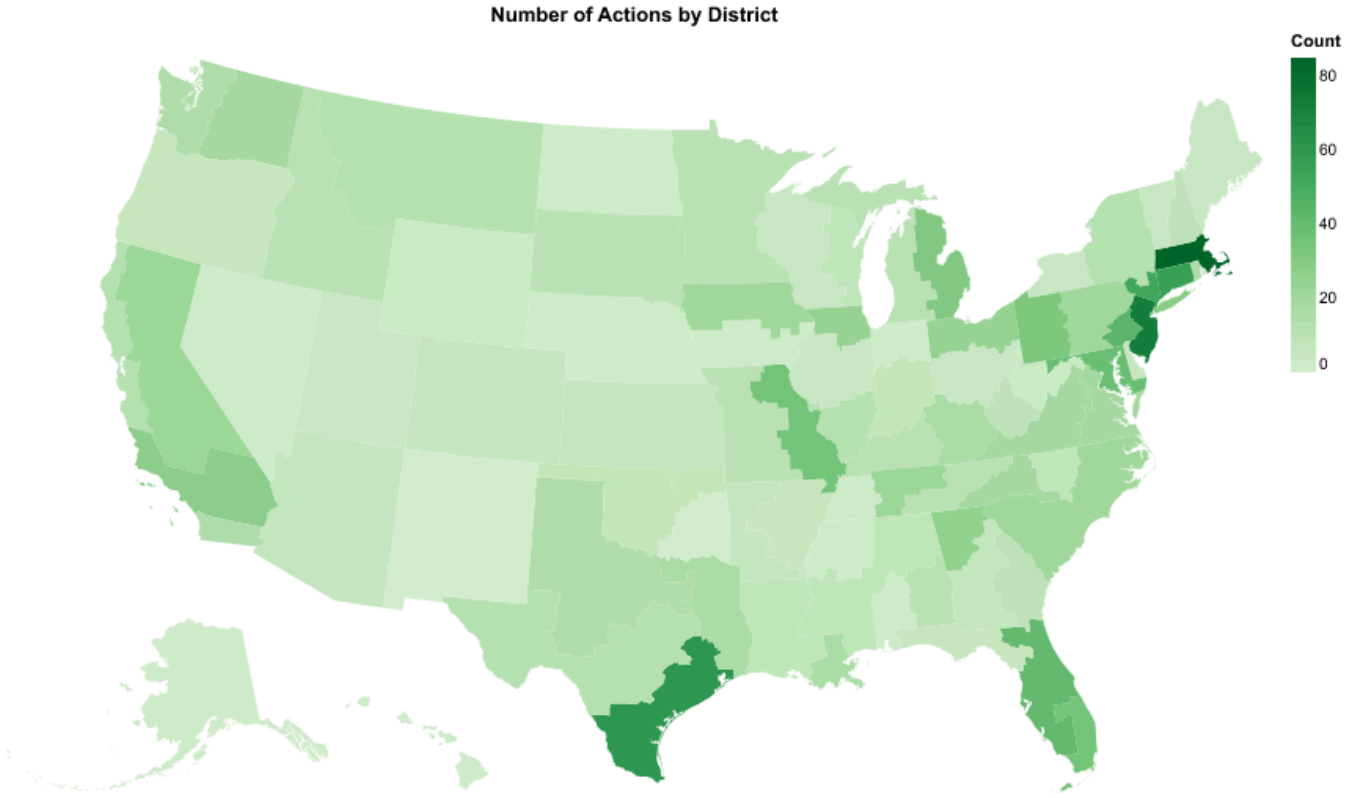
2. Map by District (PARTNER 2)

```
# Filter the dataset to include only district actions
df_district = df_2021[df_2021["Agency"].str.contains("District", na=False)]

# Clean the dataset to include only district names
df_district["Agency"] = df_district["Agency"].str.strip()
df_district["Agency"] = df_district["Agency"].replace({r'[^\x00-\x7F]+': ' '}, regex=T

df_district["Agency"] = df_district["Agency"].str.replace(
    r"^.*,(.*)", r"\1", regex=True).str.replace(
        r"^.*?;", "", regex=True).str.replace(
            r"(?i)\s*U\.S\.\s*Attorney's Office\s*", "", regex=True).str.strip()

# Find the actions taken by district
df_district_count = df_district.groupby(
    "Agency").size().reset_index(name="Count")
print(df_district_count.head())
```

```
                         Agency  Count
0  Central District of California     27
1    Central District of Illinois      4
2              District of Alaska      2
3             District of Arizona      6
4            District of Colorado      6
```

```
#Read the shapefile
district_shp = gpd.read_file("/Users/katherinetu/Desktop/PS5/US Attorney Districts Sha

#Merge the count data
district_count_shp = district_shp.merge(df_district_count,
                                  left_on="judicial_d",
                                  right_on="Agency",
                                  how="left")

district_count_shp["Count"] = district_count_shp["Count"].fillna(0)
```

```
#make a map
chart_district_count = alt.Chart(district_count_shp).mark_geoshape().encode(
    alt.Color('Count:Q', scale=alt.Scale(scheme='Greens')),
    tooltip=['judicial_d:N', 'Count:Q']
).project(
    type='albersUsa'
).properties(
    width=800,
    height=500,
    title="Number of Actions by District"
)

chart_district_count
```



Extra Credit

1. Merge zip code shapefile with population

```
#Load the data
zip_shp = gpd.read_file("/Users/katherinetu/Desktop/PS5/gz_2010_us_860_00_500k/gz_2010

zip_pop = pd.read_csv("/Users/katherinetu/Desktop/PS5/DECENNIALDHC2020.P1-Data.csv")
```

```
#clean zip_pop
zip_pop["NAME"] = zip_pop["NAME"].str.replace("ZCTA5 ", "")
zip_pop.rename(columns={"NAME": "ZCTA5", "P1_001N": "Population"}, inplace=True)
```

```
#merge the files
zip_pop_merge = zip_shp.merge(zip_pop[["ZCTA5","Population"]],
                              left_on="ZCTA5",
                              right_on="ZCTA5",
                              how="left").drop(columns = ["ZCTA5"])
```

2. Conduct spatial join

```
#Conduct the join
zip_pop_merge = zip_pop_merge.to_crs(epsg=4326)
district_shp = district_shp.to_crs(epsg=4326)

zip_pop_district_join = gpd.sjoin(zip_pop_merge, district_count_shp,how="inner", predi
```

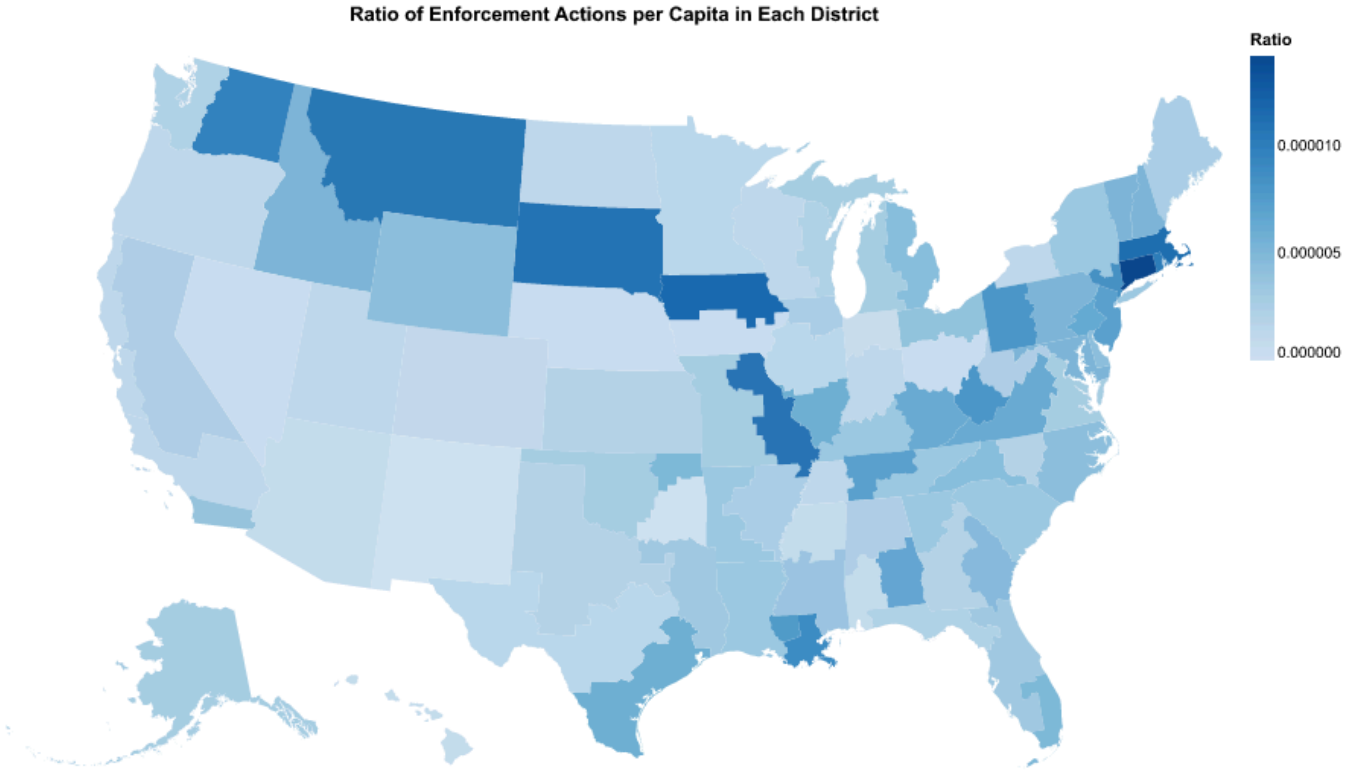
```
#Convert to numeric
zip_pop_district_join["Population"] = pd.to_numeric(zip_pop_district_join["Population"

#fill na with 0
zip_pop_district_join["Population"].fillna(0, inplace=True)
```

```
#Aggregate to get the population of each district
district_pop = zip_pop_district_join.groupby(
"judicial_d")["Population"].sum().reset_index(name = "District Pop")
```

3. Map the action ratio in each district

```
#merge the population to the shapefile
pop_district_shp = district_count_shp.merge(district_pop,
                                left_on="judicial_d",
                                right_on = "judicial_d",
                                how="left")

#calculate the ratio of enforcement actions
pop_district_shp["Ratio"] = pop_district_shp["Count"]/pop_district_shp["District Pop"]
```

```
#Map out the ratio
chart_ratio = alt.Chart(pop_district_shp).mark_geoshape().encode(
    alt.Color('Ratio:Q', scale=alt.Scale(scheme='Blues')),
    tooltip=['judicial_d:N', 'Ratio:Q']
).project(
    type='albersUsa'
).properties(
    width=800,
    height=500,
    title="Ratio of Enforcement Actions per Capita in Each District"
)

chart_ratio
```


Ratio of Enforcement Actions per Capita in Each District
Ratio
0.000010
0.000005
0.000000