PSET5: Webscraping

Jakub Budz & Charles Huang

2024-11-09

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 (Jakub Budz jbudz1):
   - Partner 2 (Charles Huang chuang2):
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: **_JB_** **_CH_**
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: **_0_** Late coins left after submission: **_JB:2, CH: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 geopandas as gpd
import re
import altair as alt
import time
from bs4 import BeautifulSoup
import requests
import warnings
import lxml
from datetime import datetime
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

Title of the enforcement action Date Category (e.g, "Criminal and Civil Actions") Link associated with the enforcement action

```
# Import url into soup object
url = 'https://oig.hhs.gov/fraud/enforcement'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
```

```
# Create df
# Attribution: ChatGPT was used to get general understanding of classes & bs4
# functions. With, I was able to pull all of the enforcement actions into an
# object "actions," which made the scraping function much easier to apply.
data = []
actions = soup.find_all(
    'li', class_='usa-card card--list pep-card--minimal mobile:grid-col-12')
for action in actions:
    # Title
    title_tag = action.find('h2', class_='usa-card__heading')
    title = title_tag.get_text(strip=True) if title_tag else None
# Date
    date_tag = action.find('span', class_='text-base-dark')
    date = date_tag.get_text(strip=True) if date_tag else None
# Category
    category_tag = action.find('li',
                               class_='display-inline-block usa-tag text-no-lowercase text-ba
    category = category_tag.get_text(strip=True) if category_tag else None
# Link
    link = title_tag.find(
        'a')['href'] if title_tag and title_tag.find('a') else None
    if link and not link.startswith("http"):
        link = "https://oig.hhs.gov" + link
    data.append({
        'Title': title,
        'Date': date,
        'Category': category,
        'Link': link
    })
oig_data = pd.DataFrame(data)
print(oig_data.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. Crawling (PARTNER 1)

```python
# Define function that captures Agency names
# Attribution: ChatGPT assisted with the text stripping
# Attribution 2: The original function did not work or would time
# out for the larger dataframe, so I used ChatGPT to make it faster
# Also used StackOverflow for general guidance:
# https://stackoverflow.com/questions/68076739/
# creating-a-function-for-my-python-web-scraper-that-will-output-a-dictionary

session = requests.Session()


def retrieve_agency(link):
    try:
        # Set a timeout and use session for connection pooling
        response = session.get(link, timeout=5)  # Set timeout as needed
        soup = BeautifulSoup(response.text, 'lxml')
        li_tags = soup.find_all('li')

        for li in li_tags:
            span_tag = li.find('span', class_='padding-right-2 text-base')
            if span_tag and span_tag.text.strip() == "Agency:":
                agency_name = li.get_text(
                    strip=True).replace("Agency:", "").strip()
                return agency_name
    except requests.exceptions.RequestException as e:
        print(f"Error fetching {link}: {e}")
        return None


oig_data['Agency Name'] = oig_data['Link'].apply(lambda x: retrieve_agency(x))
print(oig_data.head())
```

```
                                               Title              Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  November 8, 2024
1  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
2  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
3  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
4  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024

                   Category  \
0  Criminal and Civil Actions
```

```
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                Link  \
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...

                                         Agency Name
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)

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.

• This function should first check that the year inputted >= 2013 before starting to scrape. If the year inputted < 2013, it should print a statement reminding the user to restrict to year >= 2013, since only enforcement actions after 2013 are listed. • It should save the dataframe output into a .csv file named as "enforcement_actions_year_month.csv" (do not commit this file to git) • If you're crawling multiple pages, always add 1 second wait before going to the next page to prevent potential server-side block. To implement this in Python, you may look up .sleep() function from time library.

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.

def scraper_year_check(month, year): if year < 2013: print an error statement return else: my_data = [] convert target month and year to a datetime object called target_date set current_page = 1

```
while True (keep going until loop)
    set URL to "https://oig.hhs.gov/fraud/enforcement/?page={current_page}"
    soup = fetch the page and make a soup object
    li_tags = use find_all on soup to collect all li tags
    data_found = boolean track if entry matching the date range found

    for each li in li_tags:
        find the date from the span tag and set it to date_text
        compare date_text to the target_date
        if date_text < target date:
            break the while loop (done collecting data)

        if loop not broken:

        find title
        find category
        find link
        append these to my_data as a dictionary
        time.sleep(1) #pauses for 1 second to prevent server block
        get the next page URL if available
convert my_data into a dataframe called oig_dataframe
oig_dataframe.to_csv(f"enforcement_actions_{year}_{month}.csv")
```

- b. Create Dynamic Scraper (PARTNER 2)

```python
def scraper_year_check(month, year):
    if year < 2013:
        print("Year must be 2013 or later. Please enter a valid year.")
        return

    my_data = []
    target_date = datetime(year, month, 1)
    current_page = 1

    while True:
        url = f"https://oig.hhs.gov/fraud/enforcement/?page={current_page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'lxml')

        li_tags = soup.find_all(
            'li', class_='usa-card card--list pep-card--minimal mobile:grid-col-12')

        data_found = False
        for li in li_tags:
            date_text = li.find('span', class_="text-base-dark padding-right-105"
                                ).get_text(
                strip=True) if li.find('span',
                                       class_='text-base-dark padding-right-105') else None

            if date_text:
                date_object = datetime.strptime(date_text, "%B %d, %Y")

                if date_object < target_date:
                    oig_data = pd.DataFrame(my_data)
                    filename = f"enforcement_actions_{year}_{month}.csv"
                    oig_data.to_csv(filename, index=False)
                    print(f"Data saved to {filename}")
                    return

                data_found = True

                title = li.find(
                    'h2', class_="usa-card__heading"
                ).get_text(strip=True) if li.find('h2') else None
                category = li.find('li', class_='display-inline-block'
                                   ).get_text(
                    strip=True) if li.find('li', class_='display-inline-block'
                                           ) else None
                link = li.find('a')['href'] if li.find('a') else None
                if link and not link.startswith("http"):
                    link = "https://oig.hhs.gov" + link

                my_data.append({
                    'Title': title,
                    'Date': date_text,
                    'Category': category,
                    'Link': link
                })
        # stop the loop if no data found
        if not data_found:
            break

        time.sleep(1)

        current_page += 1
```

```
scraper_year_check(1, 2023)

oig_2023 = pd.read_csv('enforcement_actions_2023_1.csv')
print(f'There are {len(oig_2023)} observations since 2021.')
oig_2023['Date'] = pd.to_datetime(oig_2023['Date'])
print(oig_2023.loc[oig_2023['Date'].idxmin()])
```

```
Data saved to enforcement_actions_2023_1.csv
There are 1534 observations since 2021.
Title       Podiatrist Pays $90,000 To Settle False Billin...
Date                                      2023-01-03 00:00:00
Category                            Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/podiatri...
Name: 1533, dtype: object
```

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

```
scraper_year_check(1, 2021)
oig_2021 = pd.read_csv('enforcement_actions_2021_1.csv')
oig_2021['Agency_Name'] = oig_2021['Link'].apply(lambda x: retrieve_agency(x))

print(f'There are {len(oig_2021)} observations since 2021.')
oig_2021['Date'] = pd.to_datetime(oig_2021['Date'])
print(oig_2021.loc[oig_2021['Date'].idxmin()])
```

```
Data saved to enforcement_actions_2021_1.csv
There are 3022 observations since 2021.
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_Name    U.S. Attorney's Office, Middle District of Ten...
Name: 3021, dtype: object
```

Step 3: Plot data based on scraped data

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

```python
import numpy as np

oig_2021['Date'] = pd.to_datetime(oig_2021['Date'])

oig_2021_summary = oig_2021.groupby(oig_2021['Date'].dt.to_period('M')).size(

).reset_index(name='Count')

# Convert the 'Date' column back to datetime format
oig_2021_summary['Date'] = oig_2021_summary['Date'].dt.to_timestamp()

enf_actions = alt.Chart(oig_2021_summary).mark_line().encode(
    x=alt.X('Date:T', axis=alt.Axis(format='%Y-%m', title='Date',
                                    labelAngle=-45)),
    y=alt.Y('Count:Q', title='Count of Enforcement Actions')
).properties(title="Monthly HHS Enforcement Actions from 2021 Onwards")

enf_actions.save('enf_actions.png')
```

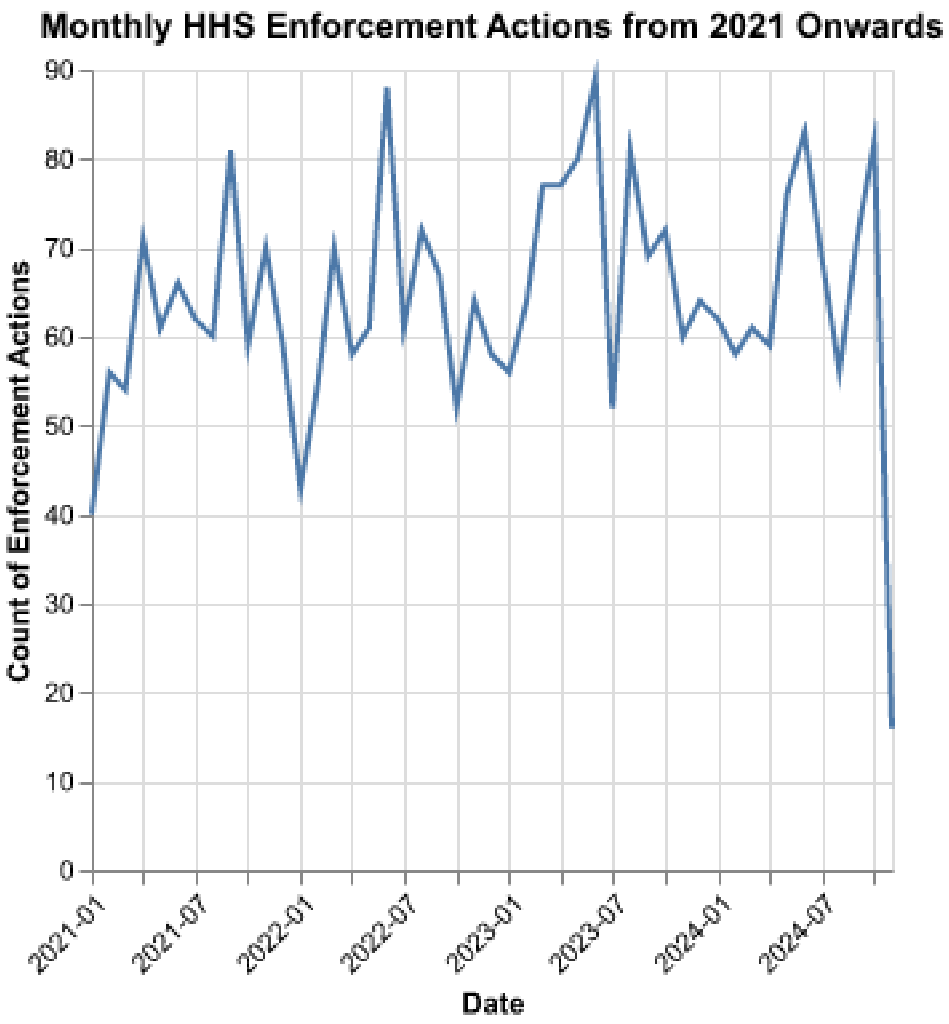

Figure 1: Enforcement Actions

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

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

```
# Filter df
crim_civ = oig_2021[oig_2021["Category"] == "Criminal and Civil Actions"]
# Assign keywords
health_keywords = ["health", "medical", "doctor", "medicaid", "healthcare",
                   "nurse", "pharmacy", "hospital", "provider", 'care',
                   'psychiatrist']
```

```python
financial_keywords = ["finance", "bank", "money laundering", "tax",
                      "accounting",
                      "fraud", "false claims", "falsely claimed",
                      "embezzlement", 'billing']
drug_keywords = ["drug", "narcotics", "opioid",
                 "pharmaceutical", "DEA", "substance", "physician"]
bribery_keywords = ["bribery", "corruption", "kickback", "fraudulent"]
other_keywords = []
# Function to apply keywords


def assign_topic(agency_name):
    if agency_name is None:
        return "Other"
    agency_name = agency_name.lower().strip()
    if any(keyword in agency_name for keyword in health_keywords):
        return "Health Care Fraud"
    elif any(keyword in agency_name for keyword in drug_keywords):
        return "Drug Enforcement"
    elif any(keyword in agency_name for keyword in bribery_keywords):
        return "Bribery/Corruption"
    if any(keyword in agency_name for keyword in financial_keywords):
        return "Financial Fraud"
    else:
        return "Other"


# Apply function to df
crim_civ['Topic'] = crim_civ['Title'].apply(assign_topic)
```

```python
cca_agency = crim_civ['Agency_Name'].value_counts().reset_index()

cca_agency_chart = alt.Chart(cca_agency).mark_line(point=True).encode(
    x=alt.X('Agency_Name', sort='-y', title='Agency'),
    y=alt.Y('count', title='Number of Criminal and Civil Actions')
).configure_axis(
    labelFontSize=8,
    labelAngle=-45
)
cca_agency_chart.save('agency_chart.png')
```

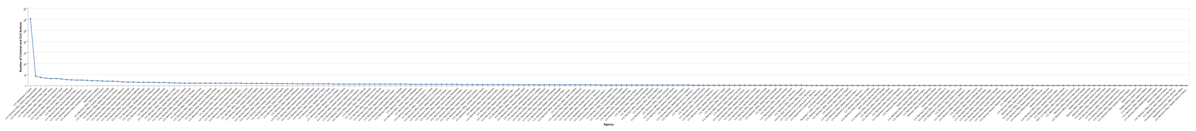

Figure 2: Criminal and Civil Actions

- based on five topics

```
cca_split = crim_civ['Topic'].value_counts().reset_index()

cca_split_chart = alt.Chart(cca_split).mark_line(point=True).encode(
    x=alt.X('Topic', sort='-y', title='Topic'),
    y=alt.Y('count', title='Number of Actions')
).configure_axis(
    labelFontSize=8,
    labelAngle=-45
).properties(
    title='Number of Actions per Topic',
    width=500,
    height=400
).configure_axis(
    labelAngle=-45,
    labelFontSize=12  # Set axis label font size for readability
)

cca_split_chart.save('split_chart.png')
```

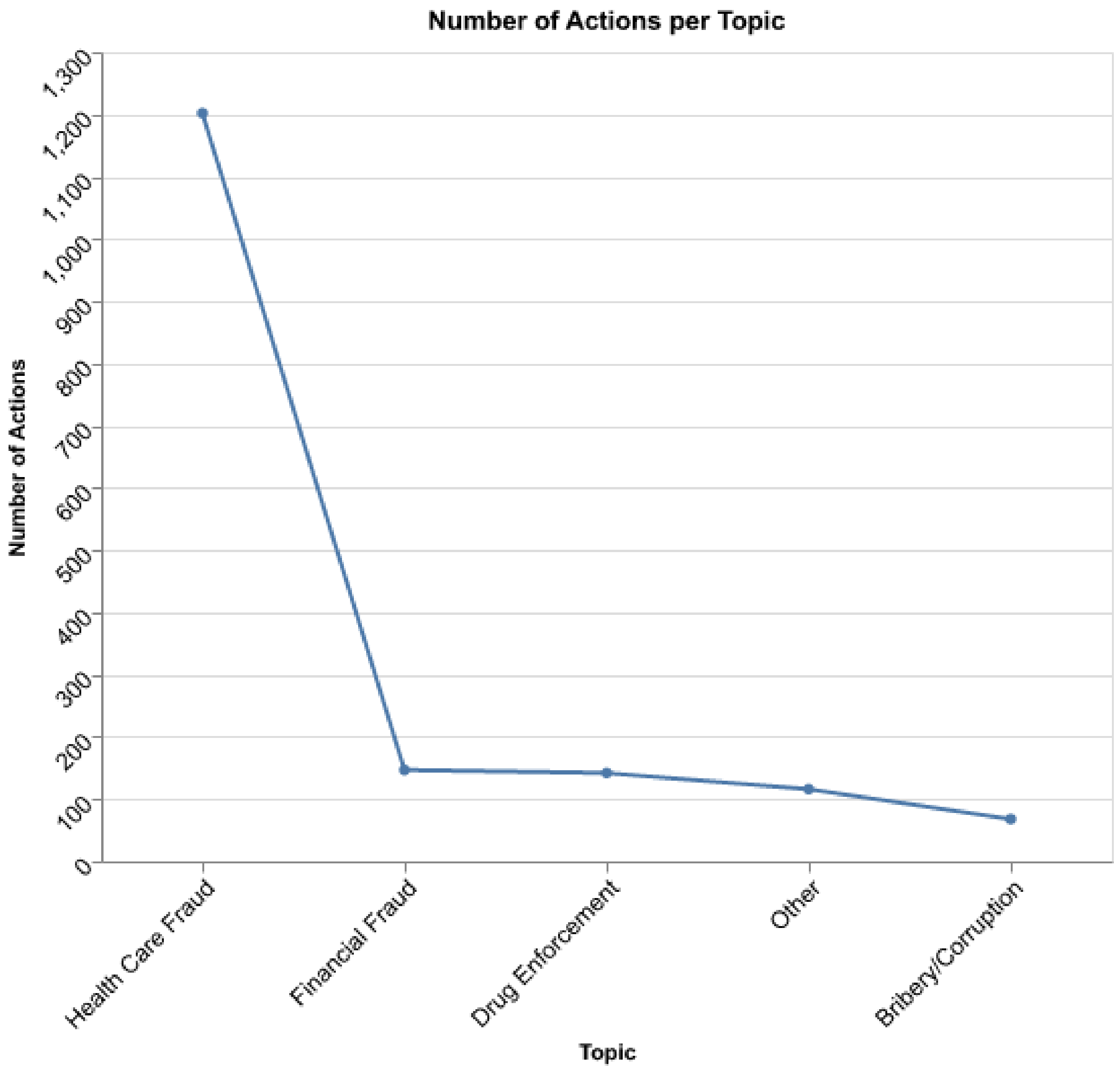

Figure 3: Criminal and Civil Actions Topics

Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
state_level = oig_2021[oig_2021['Agency_Name'].str.contains(r'\bstate of\b',
                                                         case=False, na=False)]
```

```
state_level['State'] = state_level['Agency_Name'].str.replace(r'\bstate of\b ',
                                                              '', case=False, regex=True)
state_counts = state_level.groupby('State').size().reset_index(name='Count')
state_data = gpd.read_file(
    "/Users/jakub/Downloads/cb_2018_us_state_500k/cb_2018_us_state_500k.shp")
merged_state = state_counts.merge(
    state_data, left_on="State", right_on="NAME", how='left')
merged_state = gpd.GeoDataFrame(merged_state, geometry="geometry")
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
merged_state.plot(
    column="Count",
    cmap="RdBu",
    linewidth=0.8,
    ax=ax,
    legend=True,
    legend_kwds={"shrink": 0.5, "orientation": "vertical"},
    missing_kwds={'color': 'lightgrey'}
)
ax.set_xlim([-130, -60])
ax.set_ylim([20, 55])
ax.set_aspect('equal')
ax.set_axis_off()
ax.set_title("Number of Actions per State", fontsize=16, pad=20)
plt.savefig('state_actions.png', format='png')
plt.close()
```

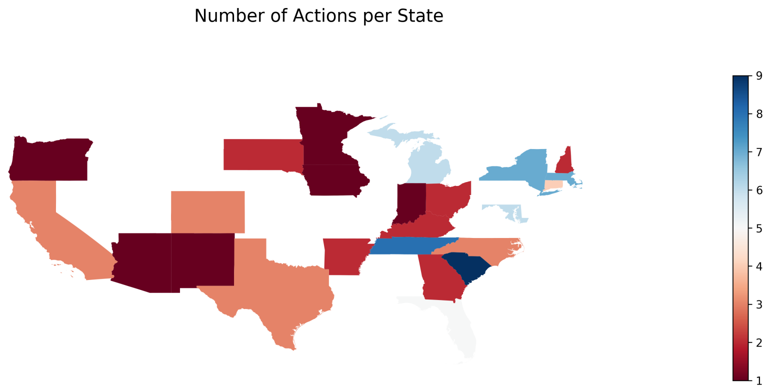

Figure 4: Actions per State

2. Map by District (PARTNER 2)

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.

(We need to take the oig_2021 data and subset it to actions taken by US Attorney District level agencies only. Then clean the names and plot them with the shapefile data.)

```
import geopandas as gpd
import re

oig_2021_district = oig_2021[oig_2021['Agency_Name'].str.contains(
    'District', na=False)]
district_data = gpd.read_file(
    "/Users/jakub/Downloads/US Attorney Districts Shapefile simplified_20241109/geo_export_7
district_data.head()
```

```python
# Attribution: Used ChatGPT for help to clean the data. Searched for "how to
# clean text string so that it removes all text before a specified symbol such
# as a comma or colon". Used the split() function to do this


def clean_agency(agency_name):
    parts = re.split(r'[,:;]', agency_name)
    return parts[-1].strip() if parts else agency_name


oig_2021_district['Agency_Name'] = oig_2021_district['Agency_Name'].apply(
    clean_agency)

# Need to group the data by district:

district_counts = oig_2021_district.groupby(
    'Agency_Name').size().reset_index(name="Action Count")

# Now we merge the district data with the shapefile:

merged_districts = district_counts.merge(
    district_data, left_on="Agency_Name", right_on="judicial_d", how='left')
merged_districts = gpd.GeoDataFrame(merged_districts, geometry="geometry")
```

Finally we can plot the choropleth:

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1, figsize=(15, 10))

merged_districts.plot(
    column="Action Count", cmap="Blues", linewidth=0.8, ax=ax, legend=True,
    legend_kwds={
        "shrink": 0.5,
        "orientation": "vertical"
    }
)

# Set xlim and ylim to focus on the contiguous U.S. (continental 48 states)
# to avoid map drift
ax.set_xlim([-130, -60])
ax.set_ylim([20, 55])
ax.set_aspect('equal')

ax.set_axis_off()
ax.set_title("Number of HHS Actions per Judicial District",
             fontsize=16, pad=20)
plt.savefig('district_actions.png', format='png')
plt.close()
```

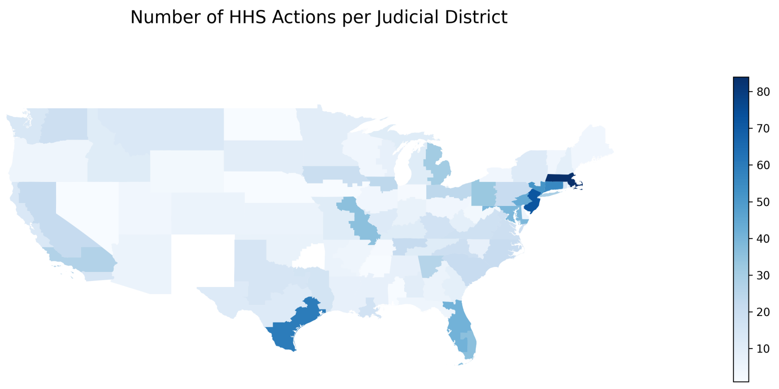

Figure 5: Actions per District