30538 Problem Set 5: Web Scraping

Boya Lin & Zidan Kong

2024-11-06

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): Boya Lin, boya1
    - Partner 2 (name and cnet ID): Zidan Kong, zidank
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: **BL** **ZK**
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

```
import pandas as pd
import altair as alt
import time
from datetime import datetime
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
import geopandas as gpd
import matplotlib.pyplot as plt

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

```
/Users/boyalin/Documents/GitHub/ppha30538_ps/DAP-PS5-ZK-BL/.venv/lib/python3.9/site-packages/
NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the
'ssl' module is compiled with 'LibreSSL 2.8.3'. See:
https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
```

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
# set the url
url = 'https://oig.hhs.gov/fraud/enforcement/'
# make a get request
response = requests.get(url)
# convert into a soup object
soup = BeautifulSoup(response.text, 'lxml')

# initialize list to store data
titles = []
dates = []
categories = []
links = []

for action in soup.find_all(
```

```
        'li', class_='usa-card card--list pep-card--minimal
         ↪ mobile:grid-col-12'):
    title_tag = action.find('h2', class_='usa-card__heading')
    # extract text from list elements and use 'N/A' if title not found
    title = title_tag.text.strip() if title_tag else 'N/A'
    titles.append(title)

    date_tag = action.find(
        'span', class_='text-base-dark padding-right-105')
    date = date_tag.text.strip() if date_tag else 'N/A'
    dates.append(date)

    category_tag = action.find(
        'li', class_='display-inline-block usa-tag text-no-lowercase
         ↪ text-base-darkest bg-base-lightest margin-right-1')
    category = category_tag.text.strip() if category_tag else 'N/A'
    categories.append(category)

    link_tag = action.find('a')
    if link_tag and link_tag.get('href'):
        link = 'https://oig.hhs.gov' + link_tag.get('href')
    else:
        link = 'N/A'
    links.append(link)

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

# display the head of the DataFrame
print(hhs_data.head())

# ChatGPT reference for extracting texts from list elements and
# aligning list length (by handeling NAs) to create DataFrame
```

```
                                               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
# initialize list to store agency
agencies = []

# iterate over each enforcement action link
for link in links:
    action_response = requests.get(link)
    action_soup = BeautifulSoup(action_response.text, 'lxml')

    # locate the 'Agency:' label and extract the agency name
    agency_tag = action_soup.find('span', text='Agency:')
    if agency_tag:
        agency_text = agency_tag.find_next_sibling(
            text=True).strip('" ').split(';')[-1].strip()
    else:
        agency_text = 'N/A'

    agencies.append(agency_text)
    time.sleep(1)

# add Agency column in DataFrame and display the head
hhs_data['Agency'] = agencies
```

```
print(hhs_data.head())

# ChatGPT reference for extracting texts from the next cell besides the
 ↪  "Agency:"
# text and removing any contents before ";" symbol
```

```
                                                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         U.S. Attorney's Office, District of Idaho
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)

I will use a while loop since i am going to iterate page by page, i will break the loop if it does not meet the date condition. I will define the while loop to run based on if the url still exists.

Define function enforcement_scraper(month, year):

```
IF year < 2013 THEN
    PRINT "Please restrict to year >= 2013."
    RETURN None

Create empty lists for later append:
    titles = empty list
    dates = empty list
    categories = empty list
    links = empty list
    agencies = empty list

base_url = "https://oig.hhs.gov/fraud/enforcement/"
target_date = datetime of input

session = requests.Session()

SET page_count = 0

WHILE url:
    page_count += 1
    PRINT "Fetching page {page_count}..." for checking scraping process

    response = session.GET(url)
    soup = BeautifulSoup(response.text, "lxml")

    actions = FIND all action items in soup
    IF actions is empty
        BREAK

    valid_date_page = False

    FOR EACH action find detail:
        title, date, category, full_link based on previous questions

        try:
            action_date = parse_action_date(date)
```

```
            IF action_date < target_date THEN
                CONTINUE
            ELSE
                valid_date_page = True

        APPEND title
        APPEND date
        APPEND category
        APPEND full_link OR "N/A"

        agency_text = fetch_agency_details(session, full_link)
        APPEND agency_text TO agencies

        SLEEP for 1 seconds

    IF valid_date_page IS False THEN
        PRINT "No relevant actions found on this page. Ending scraping."
        BREAK

    url = find_next_page_url(soup, base_url)
    IF url IS NOT None THEN
        Update url to next page
        PRINT "Moving to next page..."
    ELSE
        PRINT "No next page found. Ending scraping."
        BREAK out of the loop


data = pd.DataFrame(titles, dates, categories, links, agencies)
filename = "enforcement_actions_{year}_{month}.csv"
SAVE data TO CSV file named filename WITHOUT index
PRINT "Data saved to {filename}"

RETURN data
```

Referencing for understanding Pseudo-Code,https://builtin.com/data-science/pseudocode

- b. Create Dynamic Scraper (PARTNER 2)

```python
def enforcement_scraper(month, year):
    # check if the input year is valid
    if year < 2013:
        print('Please restrict to year >= 2013.')
```

```
        return None

    # initialize lists to store data
    titles = []
    dates = []
    categories = []
    links = []
    agencies = []

    # define the base URL
    url = 'https://oig.hhs.gov/fraud/enforcement/'
    target_date = datetime(year, month, 1)

    # set up a requests session for efficiency
    session = requests.Session()

    # loop through pages until no more pages are found
    page_count = 0
    while url:
        page_count += 1

        response = session.get(url)
        soup = BeautifulSoup(response.text, 'lxml')

        # find all actions on the page
        actions = soup.find_all(
            'li', class_='usa-card card--list pep-card--minimal
            ↪ mobile:grid-col-12')
        if not actions:
            break

        # flag to check if all actions on the page are before the target date
        page_has_relevant_actions = False

        for action in actions:
            # extract title, date, category, and link for each action
            title_tag = action.find(
                'h2', class_='usa-card__heading')
            title = title_tag.text.strip() if title_tag else 'N/A'

            date_tag = action.find(
                'span', class_='text-base-dark padding-right-105')
```

```
            date_text = date_tag.text.strip() if date_tag else 'N/A'

            category_tag = action.find(
                'li', class_='display-inline-block usa-tag text-no-lowercase
                 ↪  text-base-darkest bg-base-lightest margin-right-1')
            category = category_tag.text.strip() if category_tag else 'N/A'

            link_tag = action.find('a')
            link = link_tag['href'] if link_tag else None
            full_link = f'https://oig.hhs.gov{link}' if link and
↪  link.startswith(
                '/') else link

            # parse the action date and filter based on target_date
            try:
                action_date = datetime.strptime(date_text, '%B %d, %Y')
                if action_date < target_date:
                    continue  # skip actions before the target date
                else:
                    page_has_relevant_actions = True  # found at least one
↪  relevant action on this page
            except ValueError:
                continue  # skip actions with invalid date format

            # append data for actions that meet the date criteria
            titles.append(title)
            dates.append(date_text)
            categories.append(category)
            links.append(full_link or 'N/A')

            # if agency details are critical, make an additional request here
            if full_link:
                action_response = session.get(full_link)
                action_soup = BeautifulSoup(action_response.text, 'lxml')
                agency_tag = action_soup.find('span', string='Agency:')
                agency_text = (
                    agency_tag.find_next_sibling(string=True).strip(
                        '" ').split(':')[-1].strip()
                    if agency_tag else 'N/A'
                )
                agencies.append(agency_text)
```

```
            # pause to avoid hitting the server too frequently
            time.sleep(0.2)

        # check if this page had no relevant actions after the target date
        if not page_has_relevant_actions:
            break

        # find the link to the next page
        next_page = soup.find('a', class_='pagination-next')
        if next_page:
            next_link = next_page['href']
            url = urljoin(url, next_link)
        else:
            break

    # create a DataFrame with the collected data
    data = pd.DataFrame({
        'Title': titles,
        'Date': dates,
        'Category': categories,
        'Link': links,
        'Agency': agencies
    })

    # save the DataFrame to a CSV file
    filename = f'enforcement_actions_{year}_{month}.csv'
    data.to_csv(filename, index=False)
    print(f'Data saved to {filename}')

    return data

# ChatGPT reference for setting target date and looping through different
 ↪  pages
```

```
# collect the enforcement actions since January 2023
data_2023 = enforcement_scraper(1, 2023)

num_actions_2023 = len(data_2023)
print(f'Total number of enforcement actions from January 2023:
 ↪  {num_actions_2023}')
earliest_action_2023 = data_2023.sort_values(by='Date').iloc[0]
```

```
print(f'Details of earliest enforcement action scraped:
 ↪  {earliest_action_2023}')
```

```
Data saved to enforcement_actions_2023_1.csv
Total number of enforcement actions from January 2023: 1534
Details of earliest enforcement action scraped: Title       Shelby County Man
Indicted for Offenses Relati...
Date                                                  April 1, 2024
Category                                Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/shelby-c...
Agency      U.S. Attorney's Office, Eastern District of Ke...
Name: 507, dtype: object
```

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

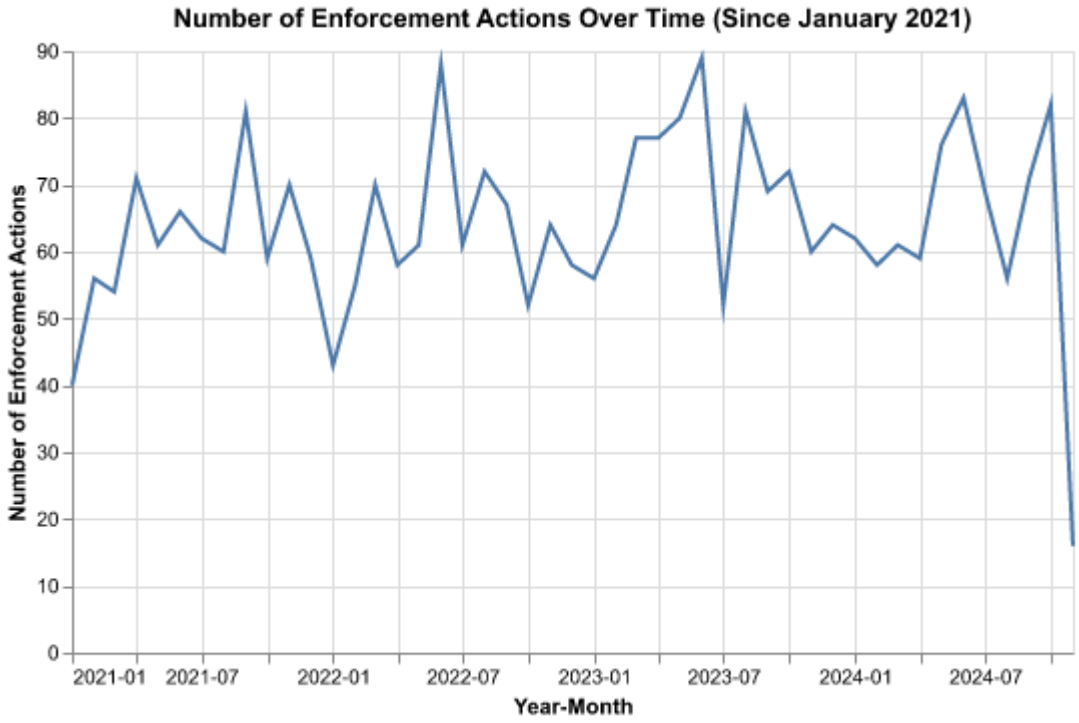
```
# collect the enforcement actions since January 2021
data_2021 = enforcement_scraper(1, 2021)

num_actions_2021 = len(data_2021)
print(f'Total number of enforcement actions from January 2021:
 ↪  {num_actions_2021}')
earliest_action_2021 = data_2021.sort_values(by='Date').iloc[0]
print(f'Details of earliest enforcement action scraped:
 ↪  {earliest_action_2021}')
```

```
Data saved to enforcement_actions_2021_1.csv
Total number of enforcement actions from January 2021: 3022
Details of earliest enforcement action scraped: Title       Red Rocks
Radiation and Oncology, Alliance Hea...
Date                                                  April 1, 2021
Category                                  Fraud Self-Disclosures
Link        https://oig.hhs.gov/fraud/enforcement/red-rock...
Agency                                                       N/A
Name: 2870, dtype: object
```

Step 3: Plot data based on scraped data

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

```
enforcement_actions_2021 = pd.read_csv('enforcement_actions_2021_1.csv')

# convert to datetime format
enforcement_actions_2021['Date'] = pd.to_datetime(
    enforcement_actions_2021['Date'], format='%B %d, %Y')

# extract 'Year-Month' for grouping
enforcement_actions_2021['Year-Month'] = enforcement_actions_2021[
    'Date'].dt.to_period('M')

# convert 'Year-Month' to string for Altair compatibility
enforcement_actions_2021['Year-Month'] = enforcement_actions_2021[
    'Year-Month'].astype(str)

# aggregate by Year-Month and count
enforcement_counts = enforcement_actions_2021.groupby(
    'Year-Month').size().reset_index(name='Enforcement Actions')

# plot the data using Altair
chart = alt.Chart(enforcement_counts).mark_line().encode(
    x=alt.X('Year-Month:T', axis=alt.Axis(format='%Y-%m',
 ↪  title='Year-Month')),
    y=alt.Y('Enforcement Actions:Q', title='Number of Enforcement Actions')
).properties(
    title='Number of Enforcement Actions Over Time (Since January 2021)',
    width=500,
    height=300
)

chart.show()

# ChatGPT reference for changing date format for "Novembe, 2024"
```

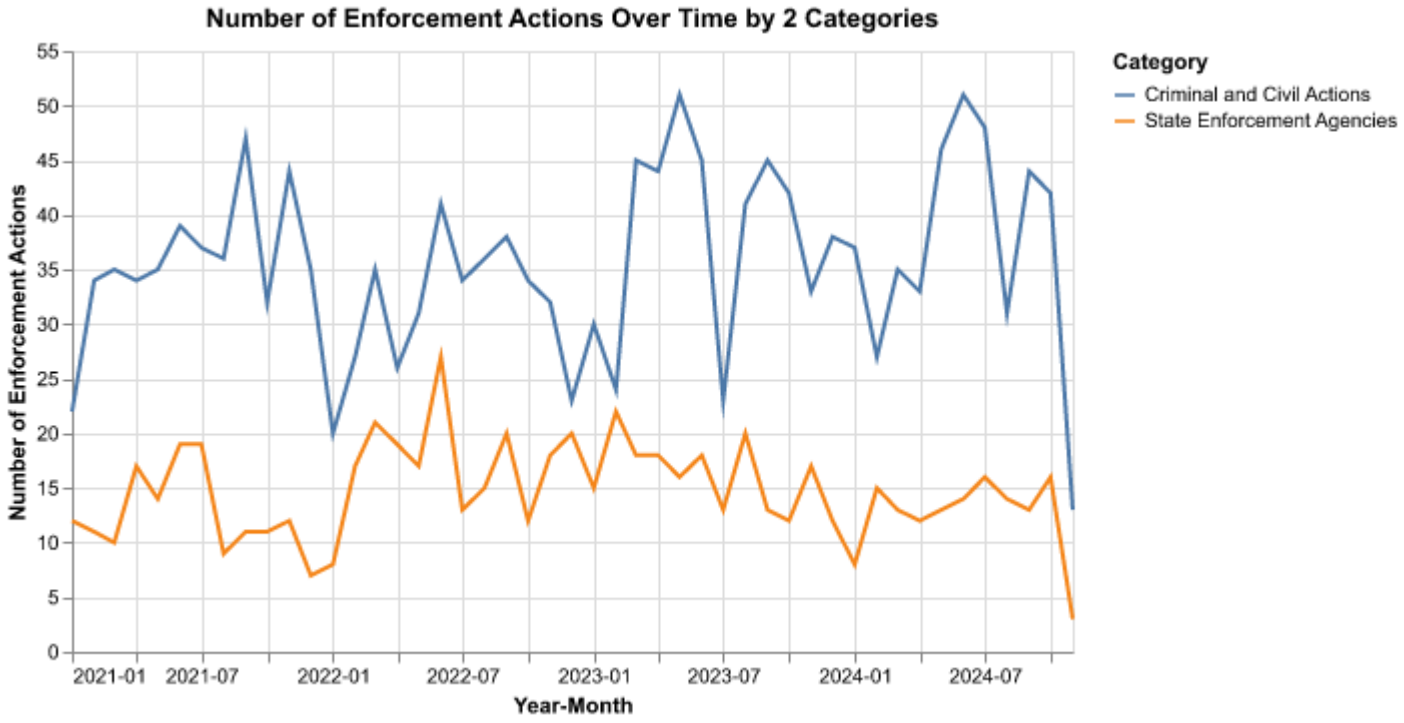


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

```
# filter data
enforcement_2category = enforcement_actions_2021[enforcement_actions_2021[
    'Category'].isin(['Criminal and Civil Actions', 'State Enforcement
     ↪  Agencies'])]

# count the occurrences of each Category by Year-Month
enforcement_2category = enforcement_2category.groupby(
    ['Year-Month', 'Category']).size().reset_index(name='Count')

# plot the data using Altair
chart_2 = alt.Chart(enforcement_2category).mark_line().encode(
    x=alt.X('Year-Month:T', axis=alt.Axis(format='%Y-%m',
 ↪  title='Year-Month')),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions'),
    color='Category:N'
).properties(
```

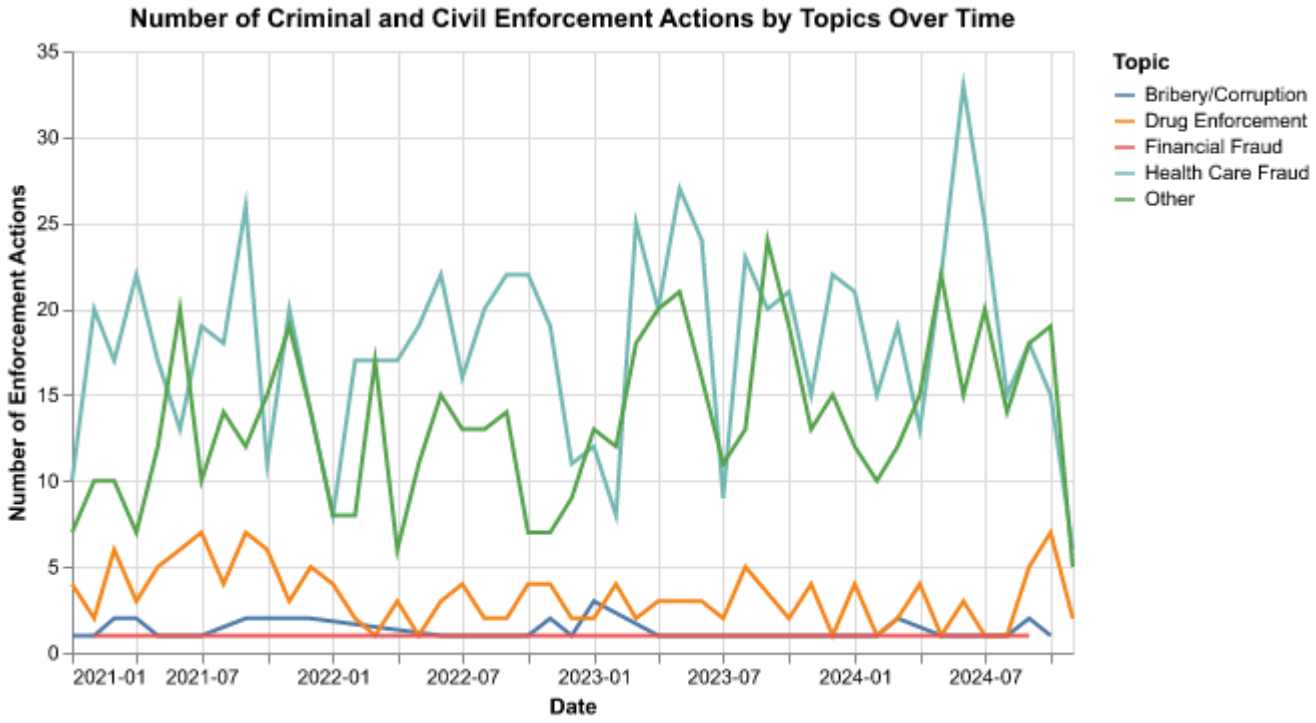
```
    title='Number of Enforcement Actions Over Time by 2 Categories',
    width=500,
    height=300
)

chart_2.show()
```



- based on five topics

```
# filter to "Criminal and Civil Actions" category
criminal_civil_actions = enforcement_actions_2021[enforcement_actions_2021[
    'Category'] == 'Criminal and Civil Actions'].copy()

# define function to categorize actions based on the 'Title' column
def categorize_topic(title):
    """ function to categorize topics within Criminal and Civial Actions
    ↪  category """
    title_lower = title.lower()
    if 'healthcare' in title_lower or 'health' in title_lower or 'medicare'
    ↪  in title_lower or 'medicaid' in title_lower:
```

```
        return 'Health Care Fraud'
    elif 'bank' in title_lower or 'financial' in title_lower:
        return 'Financial Fraud'
    elif 'drug' in title_lower or 'drugs' in title_lower or 'controlled
     ↪ substances' in title_lower or 'oxycodone' in title_lower:
        return 'Drug Enforcement'
    elif 'conspiracy' in title_lower or 'bribery' in title_lower or
     ↪ 'corruption' in title_lower:
        return "Bribery/Corruption"
    else:
        return "Other"

# apply the categorization function to each row of the 'Title' column
criminal_civil_actions['Topic'] = criminal_civil_actions['Title'].apply(
    categorize_topic)

# aggregate by Year-Month and Topic
enforcement_counts = criminal_civil_actions.groupby(
    ['Year-Month', 'Topic']).size().reset_index(name='Count')

# plot the data using Altair
chart_3 = alt.Chart(enforcement_counts).mark_line().encode(
    x=alt.X('Year-Month:T', axis=alt.Axis(format='%Y-%m',
            title='Date')),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions'),
    color='Topic:N',
    tooltip=['Year-Month:T', 'Topic:N', 'Count:Q']
).properties(
    title='Number of Criminal and Civil Enforcement Actions by Topics Over
 ↪ Time',
    width=500,
    height=300
)

chart_3.show()
```

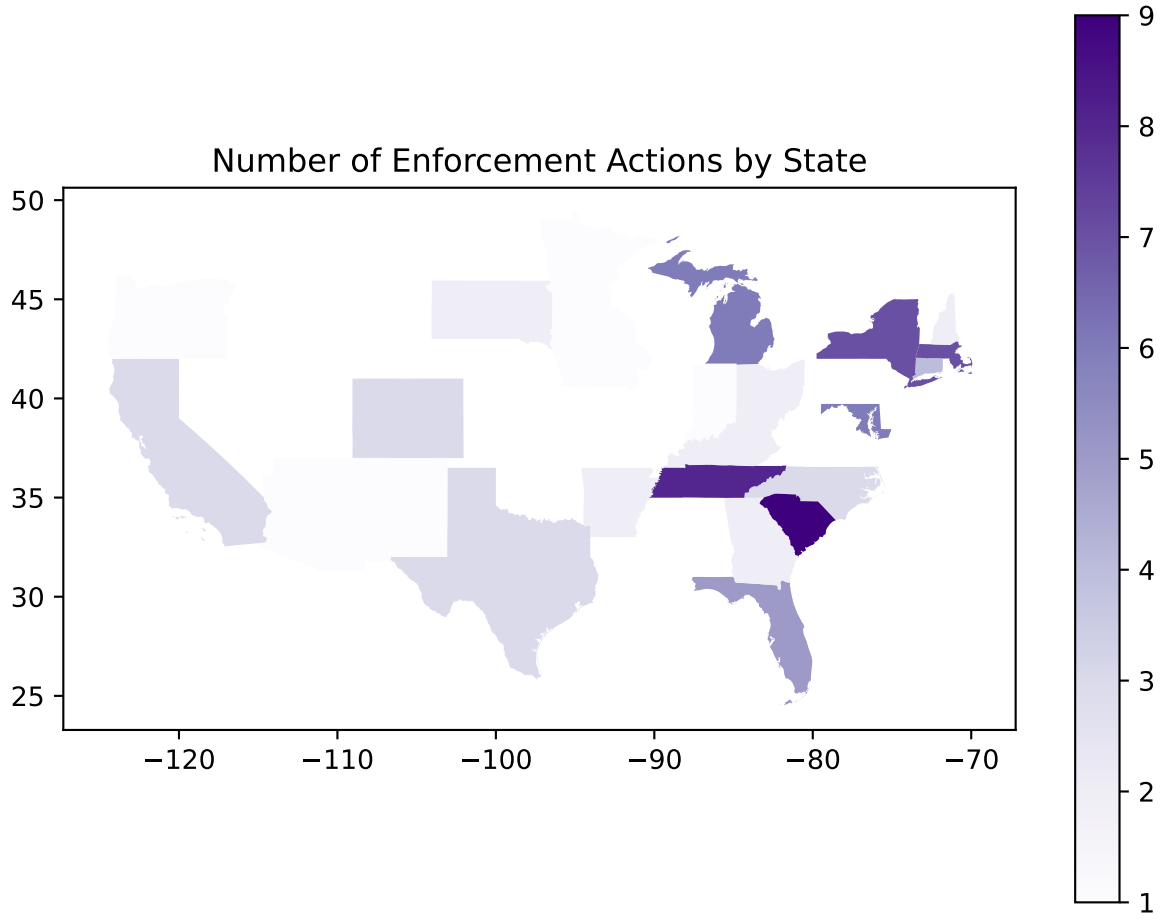


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
us_state = gpd.read_file(

    ↪  '/Users/boyalin/Documents/GitHub/ppha30538_ps/DAP-PS5-ZK-BL/cb_2018_us_state_500k/cb
us_attorney = gpd.read_file(
    '/Users/boyalin/Documents/GitHub/ppha30538_ps/DAP-PS5-ZK-BL/US Attorney
    ↪  Districts Shapefile
    ↪  simplified_20241109/geo_export_73b461cc-c808-4101-b111-3755d4c22267.shp')

# handeling NAs in the 'Agency' column
enforcement_actions_2021 = enforcement_actions_2021[enforcement_actions_2021[
    'Agency'].notna()]

# filter for State-level actions and extract state names
state_actions = enforcement_actions_2021[enforcement_actions_2021[
    'Agency'].str.contains('State of')]
```

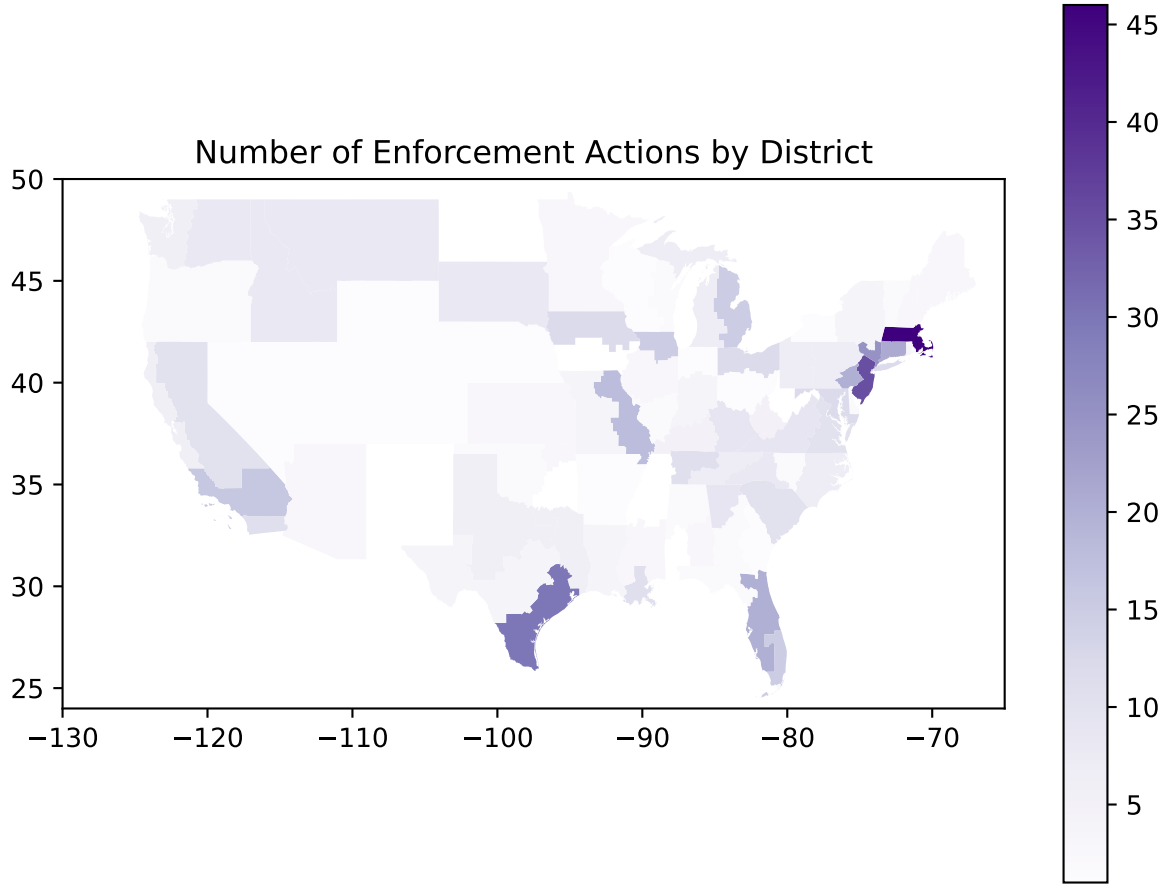
```
state_actions['State'] = state_actions['Agency'].str.split(
    'State of').str[-1].str.strip()

# aggregate by state
state_enforcement_counts = state_actions.groupby(
    'State').size().reset_index(name='Enforcement Actions')

# select only the 'NAME' and 'geometry' columns from us_state
us_state_filtered = us_state[['NAME', 'geometry']]

# merge the enforcement counts with the state geometries
state_actions_merged = state_enforcement_counts.merge(
    us_state_filtered, left_on='State', right_on='NAME', how='left')
state_actions_merged = gpd.GeoDataFrame(
    state_actions_merged, geometry='geometry')

# plot the choropleth
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
state_actions_merged.plot(column='Enforcement Actions',
                          ax=ax, legend=True, cmap='Purples')
ax.set_title('Number of Enforcement Actions by State',
             fontdict={'fontsize': '12'})
plt.show()

# ChatGPT reference for ValueError: Cannot mask with non-boolean array
# containing NA / NaN values
# ChatGPT reference for extracting contents after the key word 'State of'
```



2. Map by District (PARTNER 2)

```
# extract attorney data for groupby and merge
us_attorney_actions = enforcement_actions_2021[enforcement_actions_2021[
    'Agency'].str.contains("U.S. Attorney's Office")]
us_attorney_actions['State'] = us_attorney_actions['Agency'].str.split(
    "U.S. Attorney's Office,").str[-1].str.strip()

# clean and structure cells
us_attorney_actions['State'] = us_attorney_actions[
    'State'].str.replace(' †††', '')
us_attorney_actions['State'] = us_attorney_actions[
    'State'].str.replace(' ††', '')
```

```
us_attorney_actions['State'] = us_attorney_actions[
    'State'].str.replace('District of Idaho Boise', 'District of Idaho')

# clean and structure cells for merge
us_attorney['judicial_d'] = us_attorney['judicial_d'].str.replace(
    'District of District of Columbia', 'District of Columbia', regex=False
)
# group by to get action counts
us_attorney_actions = us_attorney_actions.groupby(
    'State').size().reset_index(name='Enforcement Actions')

# only need 'judicial_d' and 'geometry' columns from us_attorney
us_attorney_filtered = us_attorney[['judicial_d', 'geometry']]

# merge data
us_attorney_actions_merged = us_attorney_actions.merge(
    us_attorney_filtered, left_on='State', right_on='judicial_d', how='left')

# turn to geodataframe
us_attorney_actions_merged = gpd.GeoDataFrame(
    us_attorney_actions_merged, geometry='geometry')

# plot
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
us_attorney_actions_merged.plot(column='Enforcement Actions',
                                ax=ax, legend=True, cmap='Purples')
ax.set_title('Number of Enforcement Actions by District',
             fontdict={'fontsize': '12'})
ax.set_xlim(-130, -65)  # longitude limits for the contiguous U.S.
ax.set_ylim(24, 50)
plt.show()

# ChatGPT reference for Error message: Cannot mask with non-boolean array
 ↪  containing
# NA/NaN values, and for removing special symbols like ' †††' and setting
 ↪  limits
# when plotting.
```



Extra Credit

1. **Merge zip code shapefile with population**

2. **Conduct spatial join**

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