Problem Set 5

Jae Hu & Duoshu XU

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 (name and cnet ID): Duoshu Xu, duoshu
    - Partner 2 (name and cnet ID): Jae Hu, jaehu
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: DX JH
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: **___** Late coins left after submission: **___**
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 requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import geopandas as gpd
import json
from datetime import datetime

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

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
enforcement_data = []

for card in soup.find_all('li', class_='usa-card'):
    title_tag = card.find('h2', class_='usa-card__heading').find('a')
    title = title_tag.text.strip()
    link = "https://oig.hhs.gov" + title_tag.get('href')
    date_tag = card.find('span', class_='text-base-dark padding-right-105')
    date = date_tag.text.strip() if date_tag else "N/A"
    category_tag = card.find('ul', class_='display-inline add-list-reset')
    category = category_tag.find('li').text.strip() if category_tag else
↪   "N/A"
    enforcement_data.append({
        "Title": title,
        "Date": date,
        "Category": category,
        "Link": link
    })
```

```
df = pd.DataFrame(enforcement_data)
df.head()
```

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

2. Crawling (PARTNER 1)

```
def add_agency_info(df):
    enforcement_data = []

    for index, row in df.iterrows():
        response = requests.get(row["Link"])
        soup = BeautifulSoup(response.text, 'html.parser')

        agency_tag = soup.find(
            'span', class_='padding-right-2 text-base', text="Agency:")
        agency = agency_tag.find_next_sibling(
            text=True).strip() if agency_tag else "N/A"

        enforcement_data.append({
            "Title": row["Title"],
            "Date": row["Date"],
            "Category": row["Category"],
            "Link": row["Link"],
            "Agency": agency
        })

    return pd.DataFrame(enforcement_data)


df_with_agency = add_agency_info(df)
print(df_with_agency.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
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...
```

```python
base_url = "https://oig.hhs.gov/fraud/enforcement/?page="
all_enforcement_data = []

for page in range(1, 483):
    url = base_url + str(page)
    print(f"Scraping page {page}...")

    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    for card in soup.find_all('li', class_='usa-card'):
        title_tag = card.find('h2', class_='usa-card__heading').find('a')
        title = title_tag.text.strip()
        link = "https://oig.hhs.gov" + title_tag.get('href')
```

```
        date_tag = card.find('span', class_='text-base-dark
↪  padding-right-105')
        date = date_tag.text.strip() if date_tag else "N/A"

        category_tag = card.find('ul', class_='display-inline
↪  add-list-reset')
        category = category_tag.find(
            'li').text.strip() if category_tag else "N/A"

        all_enforcement_data.append({
            "Title": title,
            "Date": date,
            "Category": category,
            "Link": link
        })

    time.sleep(0.1)

df = pd.DataFrame(all_enforcement_data)
print(df)
df.to_csv("enforcement_actions_all.csv", index=False)
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

FUNCTION scrape_enforcement_actions(year, month) CHECK if year is less than 2013 DISPLAY "Please enter a year from 2013 onwards." RETURN

```
INITIALIZE enforcement_data as empty list
SET start_date to first day of the given month and year
SET current_date to today's date
SET url to construct initial URL using start_date

WHILE start_date is less than or equal to current_date
    FETCH page data using url
    PARSE the page using BeautifulSoup
```

```
    FOR each enforcement action in the parsed page
        READ title, date, category, and link
        SAVE data to enforcement_data list
    END FOR

    CHECK if there is a link to the next page
        SET url to next page's URL
        WAIT for 1 second
    IF no next page
        BREAK

    UPDATE start_date to the next month
END WHILE

CREATE DataFrame from enforcement_data
SAVE DataFrame to CSV file named "enforcement_actions_year_month.csv"
DISPLAY "Data scraping complete and saved to file."
```

END FUNCTION

- b. Create Dynamic Scraper (PARTNER 2)

```
base_url = "https://oig.hhs.gov/fraud/enforcement/?page="
enforcement_data = []
threshold_date = datetime(2023, 1, 1)
page = 1
stop_scraping = False

while not stop_scraping:
    url = base_url + str(page)
    print(f"Scraping page {page}...")

    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    for card in soup.find_all('li', class_='usa-card'):
        # Title and link
        title_tag = card.find('h2', class_='usa-card__heading').find('a')
        title = title_tag.text.strip()
        link = "https://oig.hhs.gov" + \
            title_tag.get('href')

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

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

        try:
            date = datetime.strptime(date_text, "%B %d, %Y")
        except ValueError:
            date = None

        if date and date < threshold_date:
            stop_scraping = True
            break

        category_tag = card.find('ul', class_='display-inline
↪  add-list-reset')
        category = category_tag.find(
            'li').text.strip() if category_tag else "N/A"

        enforcement_data.append({
            "Title": title,
            "Date": date_text,
            "Category": category,
            "Link": link
        })

    time.sleep(1)

    page += 1

df = pd.DataFrame(enforcement_data)

print(df)
df.to_csv("enforcement_actions_from_2023.csv", index=False)
print(
    f"\nTotal enforcement actions from 2023-01-01 to the most recent:
     ↪  {len(df)}")
print("\nEarliest Enforcement Action Scraped:")
print(df.iloc[-1])
```

```
def scrape_enforcement_actions(start_date):
    base_url = "https://oig.hhs.gov/fraud/enforcement/?page="
    enforcement_data = []
    page = 1
```

```
    stop_scraping = False

    while not stop_scraping:
        url = base_url + str(page)
        print(f"Scraping page {page}...")

        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        for card in soup.find_all('li', class_='usa-card'):
            title_tag = card.find('h2', class_='usa-card__heading').find('a')
            title = title_tag.text.strip()
            link = "https://oig.hhs.gov" + title_tag.get('href')

            date_tag = card.find('span', class_='text-base-dark
↪   padding-right-105')
            date_text = date_tag.text.strip() if date_tag else "N/A"

            try:
                date = datetime.strptime(date_text, "%B %d, %Y")
            except ValueError:
                date = None

            if date and date < start_date:
                stop_scraping = True
                break

            category_tag = card.find('ul', class_='display-inline
↪   add-list-reset')
            category = category_tag.find('li').text.strip() if category_tag
↪   else "N/A"

            enforcement_data.append({
                "Title": title,
                "Date": date_text,
                "Category": category,
                "Link": link
            })

        time.sleep(1)
        page += 1
```

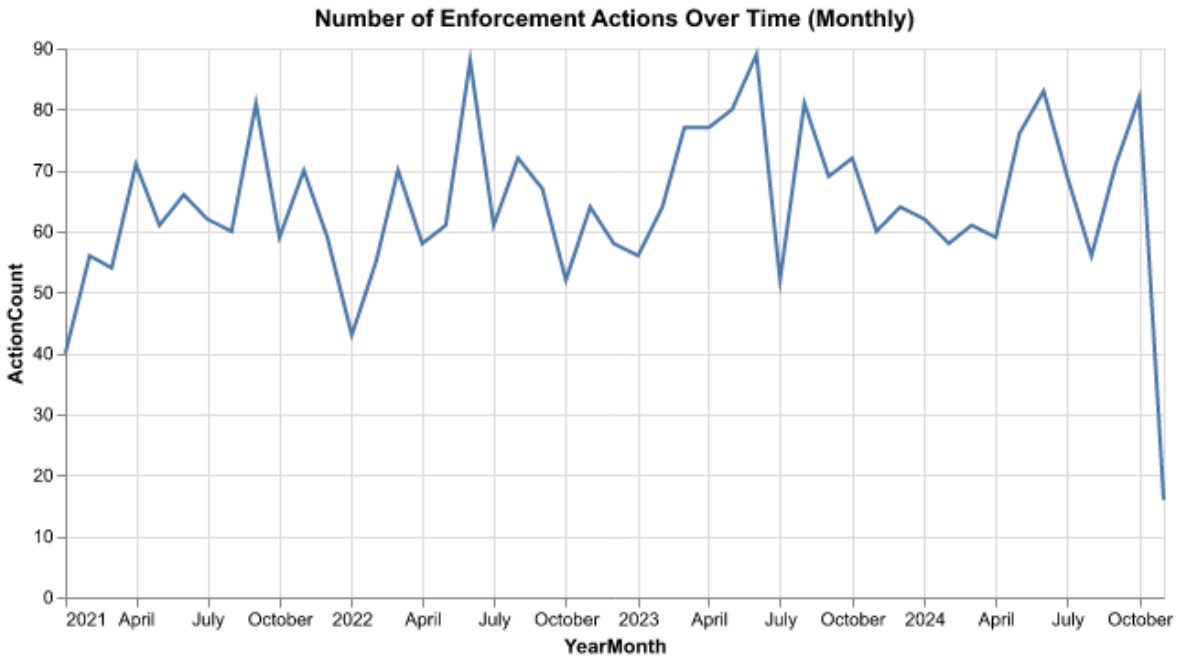
```
    df = pd.DataFrame(enforcement_data)
    return df

start_date = datetime(2021, 1, 1)
df_enforcement_actions_2021 = scrape_enforcement_actions(start_date)

print(df_enforcement_actions_2021)
df_enforcement_actions_2021.to_csv("enforcement_actions_from_2021.csv",
 ↪  index=False)
print(f"\nTotal enforcement actions from {start_date.strftime('%Y-%m-%d')} to
 ↪  the most recent: {len(df_enforcement_actions_2021)}")
print("\nEarliest Enforcement Action Scraped:")
print(df_enforcement_actions_2021.iloc[-1])
```

Step 3: Plot data based on scraped data

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

```
df_enforcement = pd.read_csv("/Users/kevinxu/Desktop/PS5/df_2021.csv")
df_enforcement['Date'] = pd.to_datetime(
    df_enforcement['Date'], errors='coerce')

df_enforcement = df_enforcement.dropna(subset=[
                                                              'Date'])

df_enforcement['YearMonth'] = df_enforcement['Date'].dt.to_period(
    'M')

monthly_counts = df_enforcement.groupby(
    'YearMonth').size().reset_index(name='ActionCount')

monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()

line_chart = alt.Chart(monthly_counts).mark_line().encode(
    x='YearMonth:T',
    y='ActionCount:Q',
    tooltip=['YearMonth:T', 'ActionCount:Q']
).properties(
    title="Number of Enforcement Actions Over Time (Monthly)",
    width=600,
```

```
)

line_chart
```

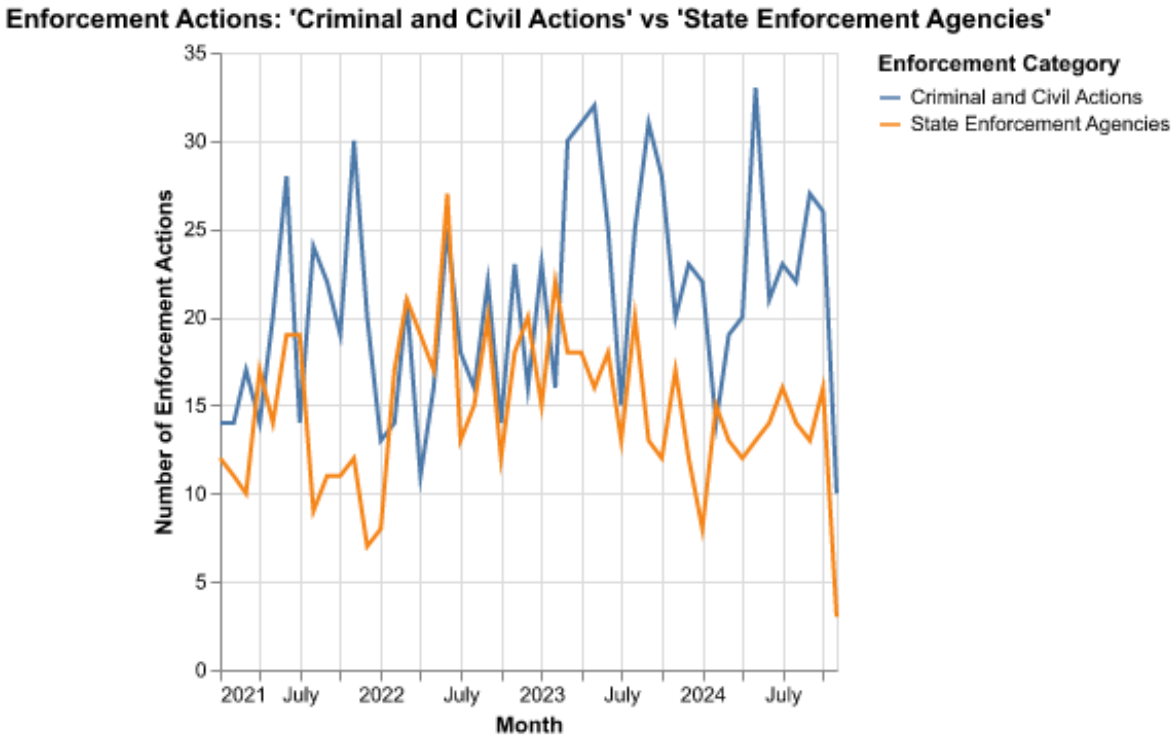


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

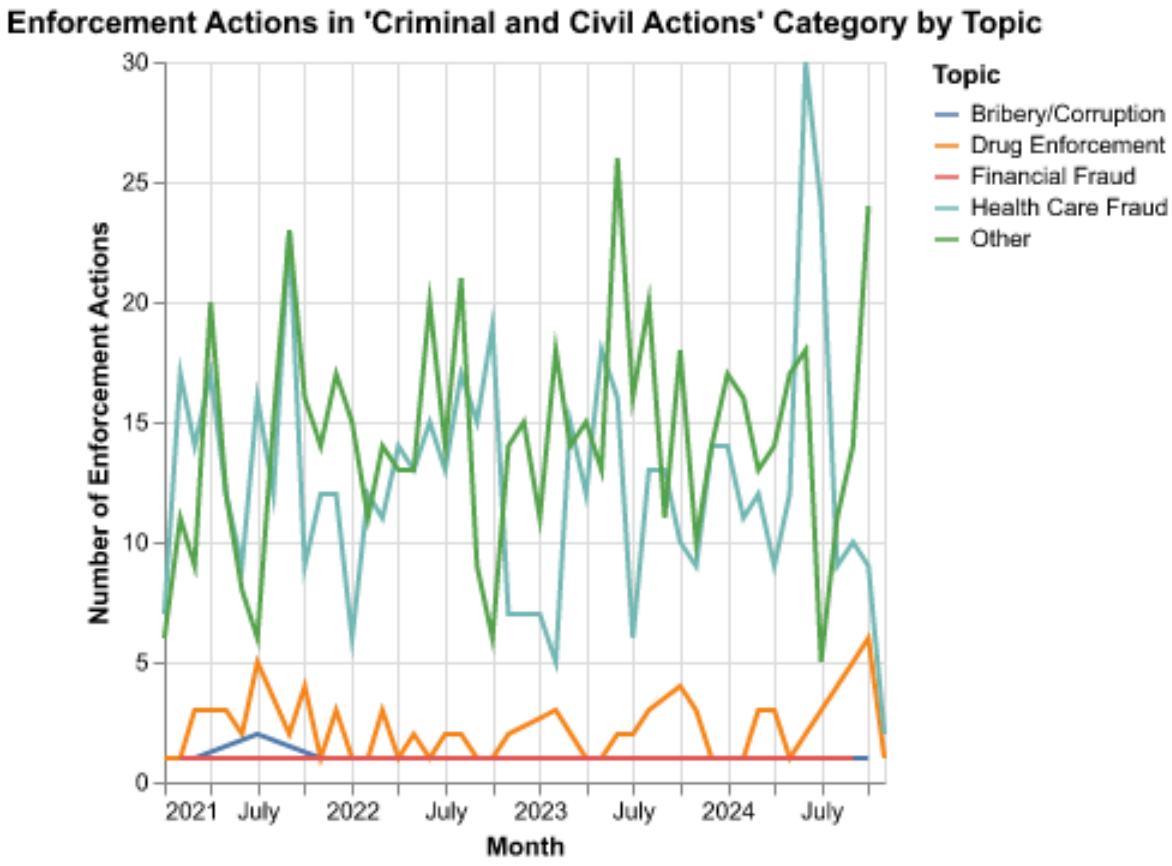
```
def categorize_action(row):
    if "criminal and civil actions" in row['Category'].lower():
        if "health" in row['Title'].lower():
            return "Health Care Fraud"
        elif "financial" in row['Title'].lower() or "bank" in
         ↪  row['Title'].lower():
            return "Financial Fraud"
        elif "drug" in row['Title'].lower():
            return "Drug Enforcement"
        elif "bribery" in row['Title'].lower() or "corruption" in
         ↪  row['Title'].lower():
            return "Bribery/Corruption"
        else:
```

```
            return "Criminal and Civil Actions"
    elif "state enforcement agencies" in row['Category'].lower():
        return "State Enforcement Agencies"
    return "Other"


df_enforcement['Topic'] = df_enforcement.apply(
    categorize_action, axis=1)
df_enforcement['Date'] = pd.to_datetime(
    df_enforcement['Date'], format='%B %d, %Y')
df_enforcement['Month'] = df_enforcement['Date'].dt.to_period(
    'M')

monthly_counts = df_enforcement.groupby(
    ['Month', 'Topic']).size().reset_index(name='Count')
monthly_counts['Month'] = monthly_counts['Month'].dt.to_timestamp()
main_chart_data = monthly_counts[monthly_counts['Topic'].isin(
    ["Criminal and Civil Actions", "State Enforcement Agencies"])]

main_chart = alt.Chart(main_chart_data).mark_line().encode(
    x=alt.X('Month:T', title='Month'),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions'),
    color=alt.Color('Topic:N', title='Enforcement Category'),
    detail='Topic:N'
).properties(
    title="Enforcement Actions: 'Criminal and Civil Actions' vs 'State
 ↪  Enforcement Agencies'"
)

main_chart
```



- based on five topics

```
sub_chart = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('Month:T', title='Month'),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions'),
    color='Topic:N',
    detail='Topic:N'
).transform_filter(
    alt.FieldOneOfPredicate(field='Topic', oneOf=[
                            "Health Care Fraud", "Financial Fraud", "Drug
                             ↪  Enforcement", "Bribery/Corruption", "Other"])
).properties(
    title="Enforcement Actions in 'Criminal and Civil Actions' Category by
 ↪  Topic"
)

sub_chart
```

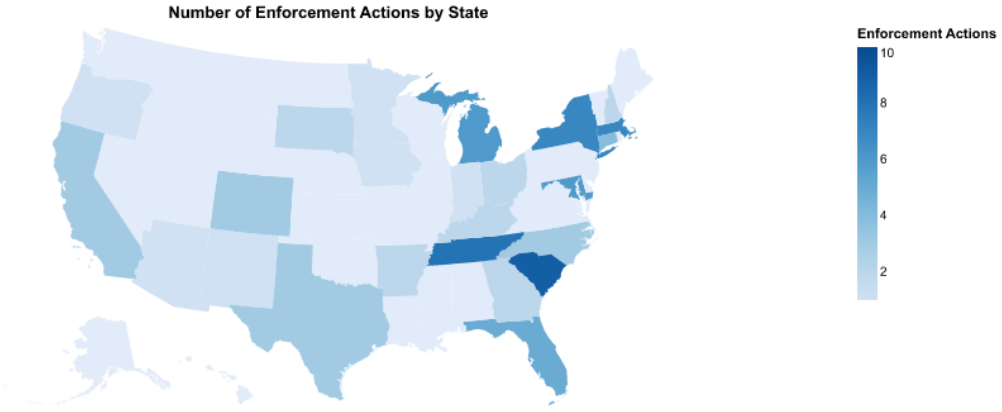


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
df_enforcement = pd.read_csv("/Users/kevinxu/Desktop/PS5/df_2021.csv")

state_agency_actions = df_enforcement[df_enforcement['Agency'].str.contains(
    "State of", na=False)]
state_agency_actions['State_Name'] =
 ↪  state_agency_actions['Agency'].str.extract(
    r"State of\s+(.+)")

state_action_counts = state_agency_actions.groupby(
    'State_Name').size().reset_index().rename(columns={0: 'Action_Count'})
```

```
state_shape_gdf =
 ↪  gpd.read_file("/Users/kevinxu/Desktop/cb_2018_us_state_500k/cb_2018_us_state_500k.shp").m
    state_action_counts, left_on="NAME", right_on="State_Name",
 ↪  how="left").fillna({'Action_Count': 0})

state_choropleth_geojson = state_shape_gdf.__geo_interface__

alt_map_chart =
 ↪  alt.Chart(alt.Data(values=state_choropleth_geojson['features'])).mark_geoshape().encode(
    color=alt.Color('properties.Action_Count:Q',
                    title="Enforcement Actions",
                    scale=alt.Scale(scheme="blues", domain=[1, 10])),
    tooltip=[
        alt.Tooltip('properties.NAME:N', title='State'),
        alt.Tooltip('properties.Action_Count:Q', title='Action Count')
    ]
).project(
    type='albersUsa'
).properties(
    width=800,
    title="Number of Enforcement Actions by State"
)

alt_map_chart
```

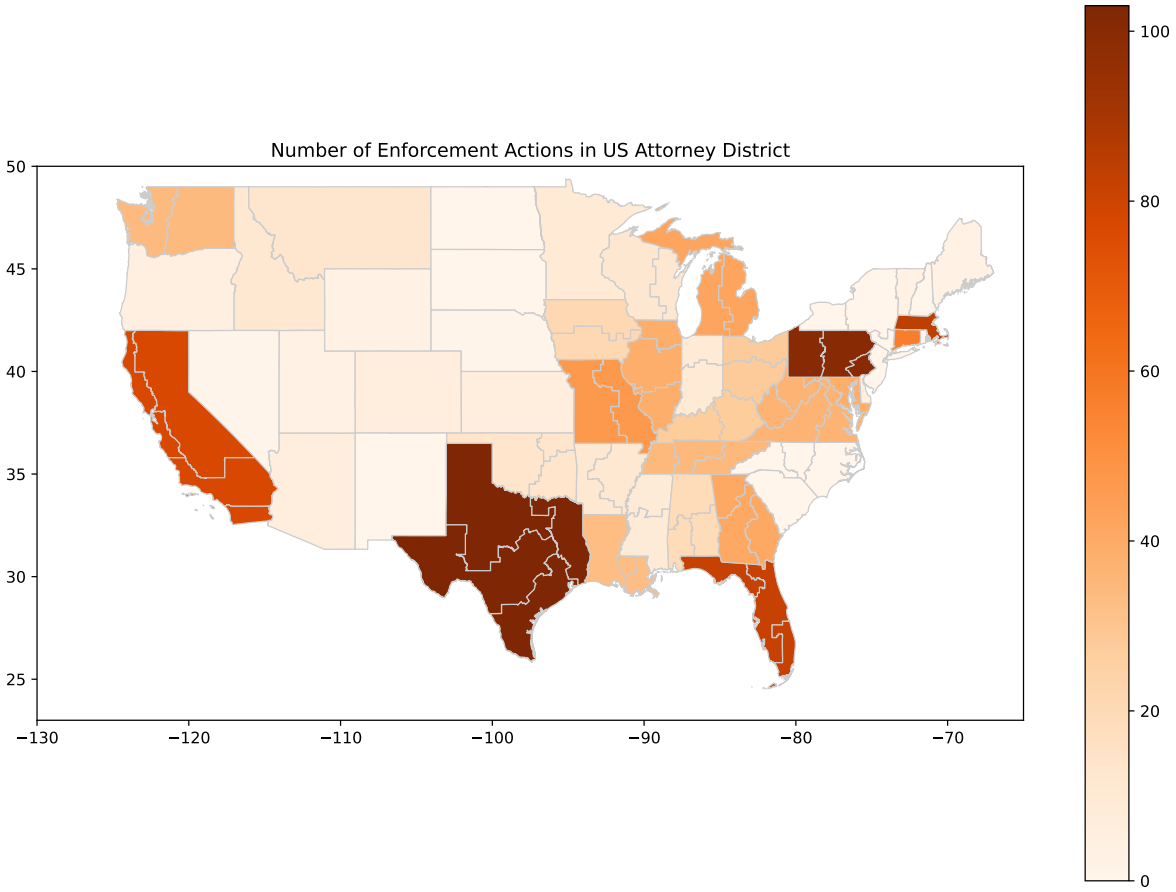


2. Map by District (PARTNER 2)

```
df_enforcement = pd.read_csv("/Users/kevinxu/Desktop/PS5/df_2021.csv")
district_shape_gdf = gpd.read_file(
    "/Users/kevinxu/Desktop/US Attorney Districts Shapefile
     ↪  simplified_20241108/geo_export_06d90bff-6de5-4159-8ecf-202d73783d42.shp")

district_shape_gdf['District_Name'] =
 ↪  district_shape_gdf['judicial_d'].str.split(
).str[-1]
print(district_shape_gdf[['judicial_d', 'District_Name']].head())
district_level_actions =
 ↪  df_enforcement[df_enforcement['Agency'].str.contains(
    "District", na=False)]
district_level_actions['District_Name'] =
 ↪  district_level_actions['Agency'].str.extract(
    r"District of\s+(.+)")

district_action_counts =
 ↪  district_level_actions['District_Name'].value_counts(
).reset_index()
district_action_counts.columns = ['District_Name', 'Action_Count']

district_action_counts['District_Name'] =
 ↪  district_action_counts['District_Name'].str.strip()

district_choropleth_gdf = district_shape_gdf.merge(
    district_action_counts, left_on="District_Name",
 ↪  right_on="District_Name", how="left")
district_choropleth_gdf['Action_Count'] =
 ↪  district_choropleth_gdf['Action_Count'].fillna(
    0)

fig, ax = plt.subplots(1, 1, figsize=(14, 10))
district_choropleth_gdf.plot(
    column='Action_Count',
    cmap='Oranges',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.8',
    legend=True
```

```
)

ax.set_xlim(-130, -65)
ax.set_ylim(23, 50)
ax.set_title("Number of Enforcement Actions in US Attorney District")
plt.show()
```

```
                    judicial_d District_Name
0  Western District of Kentucky      Kentucky
1  Eastern District of Kentucky      Kentucky
2  Southern District of Indiana       Indiana
3    Middle District of Alabama       Alabama
4  Southern District of Alabama       Alabama
```



Extra Credit

1. Merge zip code shapefile with population

```
import geopandas as gpd
import pandas as pd

shapefile_path =
 ↪  "/Users/kevinxu/Desktop/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp"
zip_shapefile = gpd.read_file(shapefile_path)

population_data_path =
 ↪  "/Users/kevinxu/Desktop/DECENNIALDHC2020.P1_2024-11-10T150929/DECENNIALDHC2020.P1-Data.cs
population_data = pd.read_csv(population_data_path)

population_data = population_data[population_data['NAME']
                                  != "Geographic Area Name"]

population_data = population_data.rename(columns={'NAME': 'ZCTA5'})

population_data['ZCTA5'] = population_data['ZCTA5'].str.extract(r'(\d{5})')

zip_shapefile['ZCTA5'] = zip_shapefile['ZCTA5'].astype(str).str.zfill(5)
population_data['ZCTA5'] = population_data['ZCTA5'].astype(str).str.zfill(5)
common_zips = set(zip_shapefile['ZCTA5']).intersection(
    set(population_data['ZCTA5']))
print(f"Number of common ZIP codes after transformation: {len(common_zips)}")

merged_data = zip_shapefile.merge(population_data, on="ZCTA5", how="left")
merged_data = merged_data.drop(columns=['Unnamed: 3', 'GEO_ID_y'])
print(merged_data.head())
merged_data.to_file("merged_zip_population.shp")
```

```
Number of common ZIP codes after transformation: 32923
         GEO_ID_x  ZCTA5   NAME   LSAD  CENSUSAREA  \
0  8600000US01040  01040  01040  ZCTA5      21.281
1  8600000US01050  01050  01050  ZCTA5      38.329
2  8600000US01053  01053  01053  ZCTA5       5.131
3  8600000US01056  01056  01056  ZCTA5      27.205
4  8600000US01057  01057  01057  ZCTA5      44.907
```

```
                                              geometry P1_001N
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...   38238
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...    2467
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...    2031
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...   21002
4  MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...    8152
```

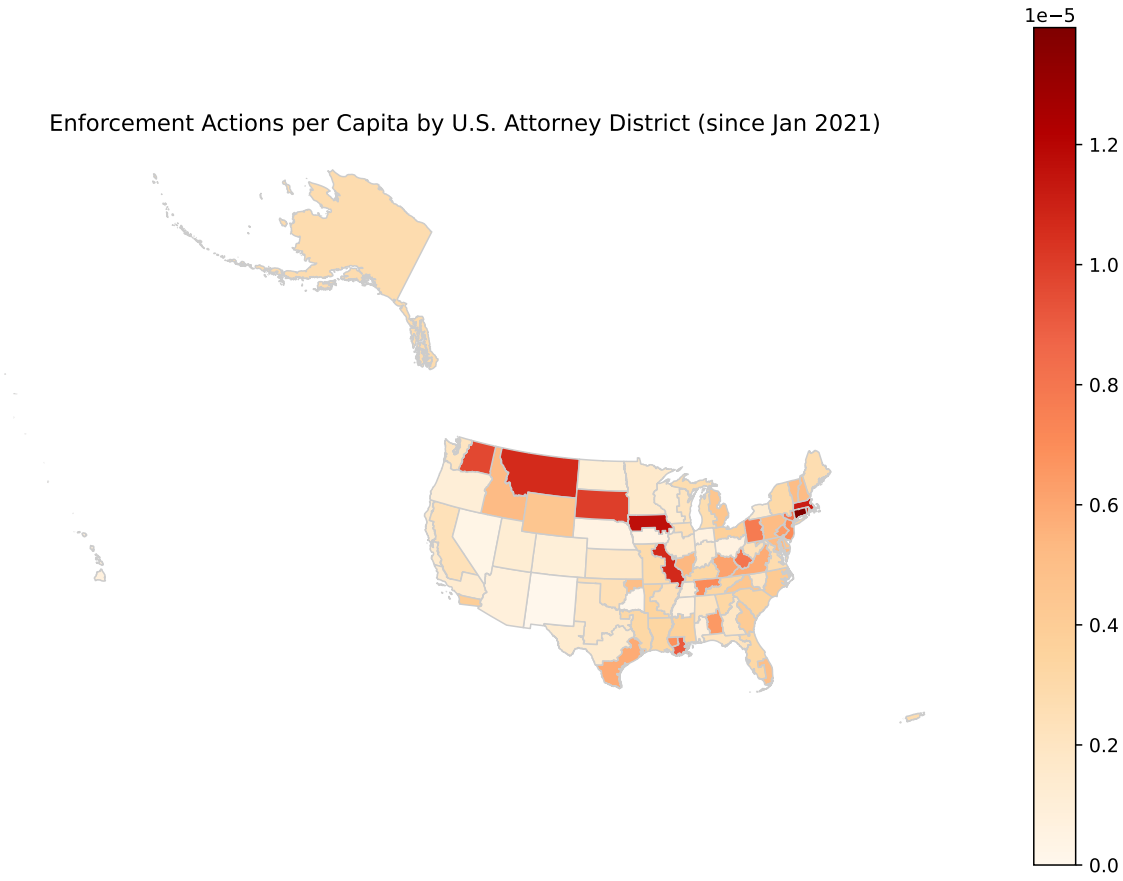
2. Conduct spatial join

```
import geopandas as gpd
import pandas as pd

zip_shapefile_path = "/Users/kevinxu/Desktop/merged_zip_population.shp"
zip_data = gpd.read_file(zip_shapefile_path)
district_shapefile_path = "/Users/kevinxu/Desktop/US Attorney Districts
 ↪  Shapefile
 ↪  simplified_20241108/geo_export_06d90bff-6de5-4159-8ecf-202d73783d42.shp"
district_data = gpd.read_file(district_shapefile_path)

if zip_data.crs != district_data.crs:
    zip_data = zip_data.to_crs(district_data.crs)
zip_district_join = gpd.sjoin(
    zip_data, district_data, how="inner", predicate="intersects")
zip_district_join['population_total'] = pd.to_numeric(
    zip_district_join['P1_001N'], errors='coerce')
district_population = zip_district_join.groupby(
    'judicial_d')['population_total'].sum().reset_index()
district_population_data = district_data.merge(
    district_population, on="judicial_d", how="left")

district_population_data.to_file("district_population_data.shp")
```

3. Map the action ratio in each district

```
district_population_path =
 ↪  "/Users/kevinxu/Desktop/district_population_data.shp"
enforcement_actions_path = "/Users/kevinxu/Desktop/PS5/df_2021.csv"
district_population_data = gpd.read_file(district_population_path)
```

```
enforcement_data = pd.read_csv(enforcement_actions_path)

enforcement_data['Date'] = pd.to_datetime(
    enforcement_data['Date'], errors='coerce')
enforcement_data = enforcement_data[enforcement_data['Date'] >= "2021-01-01"]
enforcement_data['district_name'] = enforcement_data['Agency'].str.extract(
    r'Office, (.*)')
enforcement_count = enforcement_data.groupby(
    'district_name').size().reset_index(name='enforcement_count')

district_population_data = district_population_data.merge(
    enforcement_count, left_on='judicial_d', right_on='district_name',
 ↪  how='left')
district_population_data['enforcement_count'] =
 ↪  district_population_data['enforcement_count'].fillna(
    0)
district_population_data['enforcement_ratio'] =
 ↪  district_population_data['enforcement_count'] / \
    district_population_data['population']
district_population_data = district_population_data.to_crs("EPSG:5070")

fig, ax = plt.subplots(1, 1, figsize=(12, 8))
district_population_data.plot(column='enforcement_ratio', cmap='OrRd',
                              linewidth=0.8, ax=ax, edgecolor='0.8',
 ↪  legend=True)
ax.set_title(
    'Enforcement Actions per Capita by U.S. Attorney District (since Jan
     ↪  2021)')
ax.set_axis_off()
plt.show()
```


Enforcement Actions per Capita by U.S. Attorney District (since Jan 2021)
1e−5
1.2
1.0
0.8
0.6
0.4
0.2
0.0