PS5 Solution

Peter Ganong, Maggie Shi, Akbar Saputra, and Will Pennington

2024-11-03

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):
    - Partner 2 (name and cnet ID):
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: **___** **___**
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 warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
import requests
from bs4 import BeautifulSoup

# Read data
def get_soup(url):
  response = requests.get(url)
  return BeautifulSoup(response.text, 'lxml')

url = 'https://oig.hhs.gov/fraud/enforcement/'
soup = get_soup(url)

# Get titles and links
h2s = soup.find_all('h2')
title_data = []
link_data = []
for el in h2s:
  if (el.get('class') == ['usa-card__heading']):
    title_data.append(el.text.replace('\n', ''))
    link_data.append("https://oig.hhs.gov" + el.find_all('a',
 ↪  href=True)[0].get('href'))

# Get date data
spans = soup.find_all('span')
date_data = []
for el in spans:
  if (el.get('class') == ['text-base-dark', 'padding-right-105']):
```

```
    date_data.append(el.text)

# Get category data
uls = soup.find_all('ul')
cat_data = []
for el in uls:
  if (el.get('class') == ['display-inline', 'add-list-reset']):
    cat_text = el.find_all('li')[0].text
    cat_data.append(cat_text)

df = pd.DataFrame({
  'Title': title_data,
  'Date': date_data,
  'Category': cat_data,
  'Link': link_data
})

print(df.head(5))
```

```
                                               Title              Date  \
0  St. Louis County Woman Accused Of $3 Million H...  November 1, 2024
1  Lab Owner And Marketing Company Owner Both Fou...  November 1, 2024
2  Compound Ingredient Supplier Medisca Inc., To ...  November 1, 2024
3  Columbus Doctor, His Clinic Convicted of $1.5 ...   October 31, 2024
4  Quincy-Based Physician Group To Pay $650,000 T...   October 30, 2024

                     Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  State Enforcement Agencies
4  State Enforcement Agencies

                                                Link
0  https://oig.hhs.gov/fraud/enforcement/st-louis...
1  https://oig.hhs.gov/fraud/enforcement/lab-owne...
2  https://oig.hhs.gov/fraud/enforcement/compound...
3  https://oig.hhs.gov/fraud/enforcement/columbus...
4  https://oig.hhs.gov/fraud/enforcement/quincy-b...
```

2. Crawling (PARTNER 1)

```
def get_agency(action_links):
    soup = get_soup(action_links)
    uls = soup.find("ul", class_="usa-list")
    agency = uls.find_all("li")[1].text
    agency = agency.replace("Agency:", "")

    return agency

df['Agency'] = df['Link'].apply(get_agency)
print(df.head(5))
```

```
                                               Title              Date  \
0  St. Louis County Woman Accused Of $3 Million H...  November 1, 2024
1  Lab Owner And Marketing Company Owner Both Fou...  November 1, 2024
2  Compound Ingredient Supplier Medisca Inc., To ...  November 1, 2024
3  Columbus Doctor, His Clinic Convicted of $1.5 ...  October 31, 2024
4  Quincy-Based Physician Group To Pay $650,000 T...  October 30, 2024

                     Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  State Enforcement Agencies
4  State Enforcement Agencies

                                                Link  \
0  https://oig.hhs.gov/fraud/enforcement/st-louis...
1  https://oig.hhs.gov/fraud/enforcement/lab-owne...
2  https://oig.hhs.gov/fraud/enforcement/compound...
3  https://oig.hhs.gov/fraud/enforcement/columbus...
4  https://oig.hhs.gov/fraud/enforcement/quincy-b...

                                             Agency
0  U.S. Attorney's Office, Eastern District of Mi...
1  U.S. Attorney's Office, Middle District of Ten...
2                          U.S. Department of Justice
3                                               Ohio
4                             State of Massachusetts
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

The function takes starting year and starting month as an input.

- First, check if the starting year is valid (> 2013). If not, print an error message and exit the function.
- Initialize a `date` object representing the first day of the start month and year (`start_date`), and initialize an empty list to store the scrapped data (`enforcement_data`).
- Loop through pages 1 to 480 (the maximum number of pages currently available on the website).
- For each page:
  - Get the HTML content of the page.
  - Parse the content to find all enforcement actions on the page.
  - For each enforcement action found on the page:
    - Extract the title, action date, category, and link of the action.
    - Extract the agency information (we can use the function from previous step).
- Check date validity. If action date is earlier than `start_date`, stop the loop and return `enforcement_data` (ending the function). If not, continue looping and append the newly scrapped action into the `enforcement_data`.
- Pause for 1 second between each page request to prevent overloading the server.
- After looping through all pages (or stopping early if an action date is older than `start_date`), return `enforcement_data`.

As we can see from the pseudo-code above, we used `for` loop to loop through all pages. Other than that, we can also use `while` loop to do the same thing. We can set a flag to `True` to keep going to the next page, and when action date is earlier than `start_date`, set the flag to `False` that effectively ends the loop and return the result.

- b. Create Dynamic Scraper (PARTNER 2)

```
from datetime import datetime

def scrape_actions(start_year, start_month):
  if start_year < 2013:
    print("Please use a year >= 2013.")
    return []

  start_date = datetime(start_year, start_month, 1)
  enforcement_data = []

  for page in range(1, 481): # As per today, there are 480 pages in the url
    soup = get_soup(f"https://oig.hhs.gov/fraud/enforcement/?page={page}")

    actions = soup.find_all("header", class_="usa-card__header")
    for action in actions:
      title = action.find("h2").text.replace('\n', '')
      date_text = action.find("span", class_="text-base-dark").text
      category = action.find("li", class_="usa-tag").text
      full_link = "https://oig.hhs.gov" + action.find("a").get("href")
      agency = get_agency(full_link)

      action_date = datetime.strptime(date_text, "%B %d, %Y")

      if action_date < start_date:
        break
      else:
        enforcement_data.append({"Title": title,
                                 "Date": action_date,
                                 "Category": category,
                                 "Link": full_link,
                                 "Agency": agency})

    time.sleep(1)

  filename = f"enforcement_actions_{start_year}_{start_month}.csv"
  df = pd.DataFrame(enforcement_data)

  if not df.empty:
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")
  else:
    print("No data to save.")
```

```
  return None
```

```
scrape_actions(2023, 1)
```

There are +/- 1503 actions, with the earliest being "Podiatrist Pays $90,000 To Settle False Billing Allegations", published on 3 Jan 2023 under "Criminal and Civil Actions", with U.S. Attorney's Office, Southern District of Texas as the agency.

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

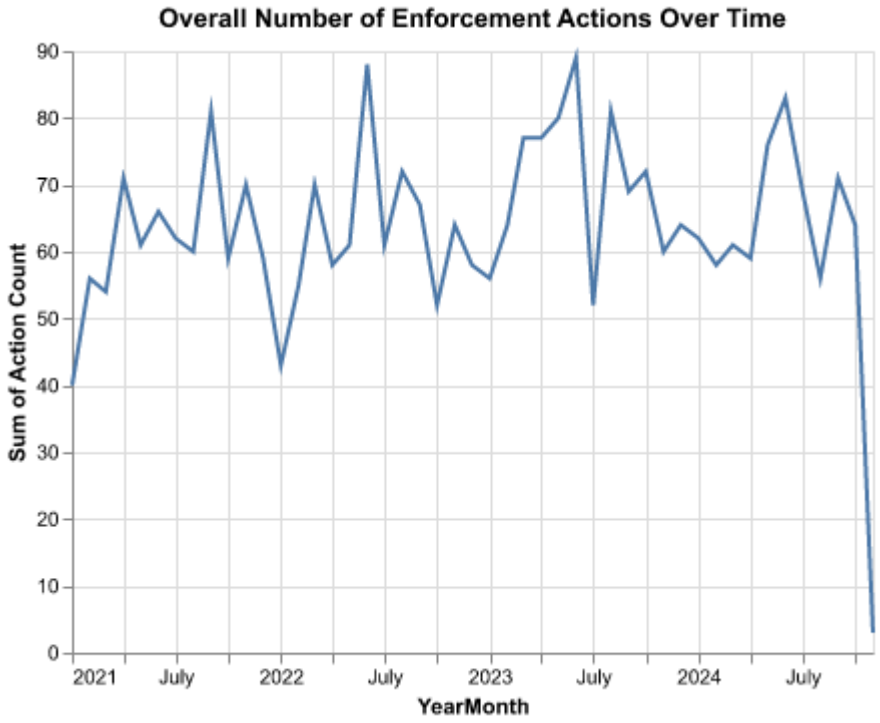
```
scrape_actions(2021, 1)
```

There are +/- 2991 actions, with the earliest being "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", published on 4 Jan 2021 under "Criminal and Civil Actions", with U.S. Attorney's Office, Middle District of Tennessee as the agency.

Step 3: Plot data based on scraped data

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

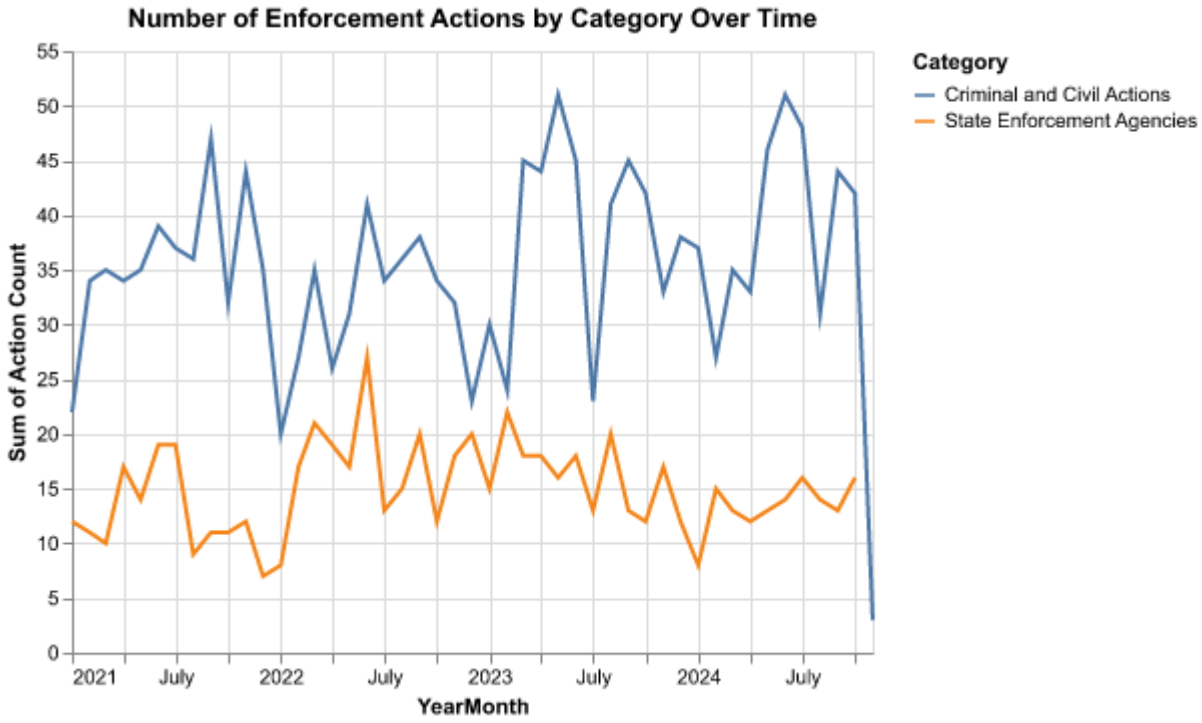
```
df = pd.read_csv("enforcement_actions_2021_1.csv")
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df['YearMonth'] = df['Date'].dt.to_period('M').dt.to_timestamp()
monthly_data = df.groupby(['YearMonth', 'Category']).agg({
                    'Title': 'count'}).reset_index().rename(columns={'Title':
                     ↪  'Action Count'})

overall_chart = alt.Chart(monthly_data).mark_line().encode(
  x='YearMonth:T',
  y='sum(Action Count):Q'
).properties(
  title="Overall Number of Enforcement Actions Over Time",
  width=400,
  height=300
)

overall_chart
```



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

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

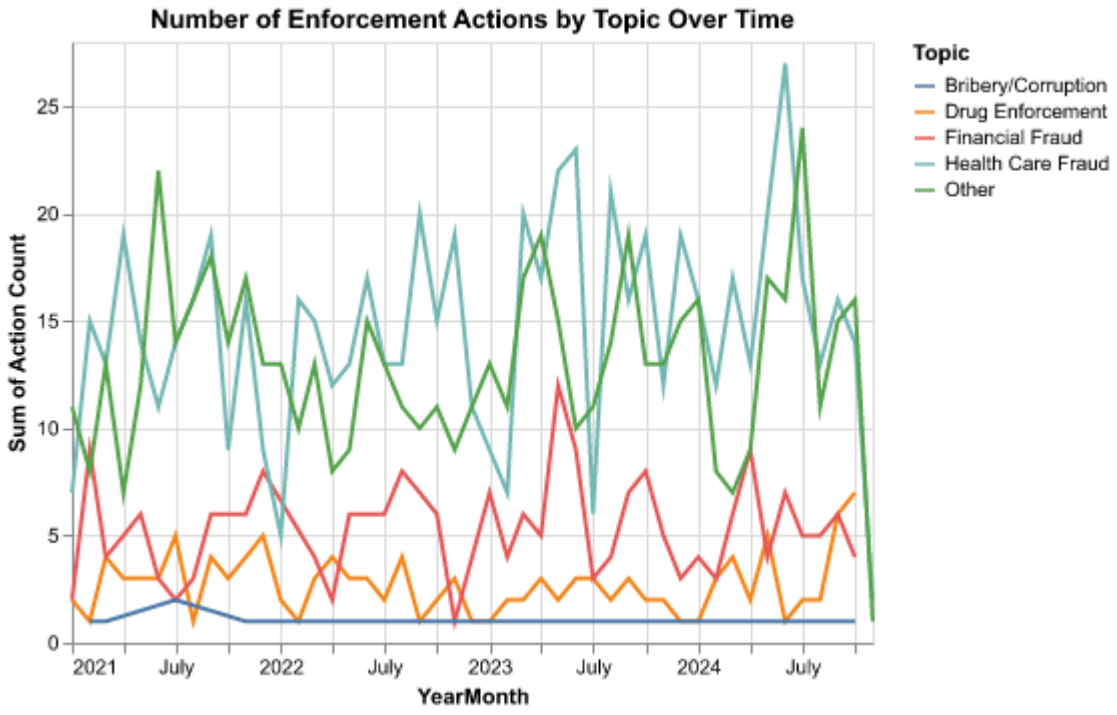
```
filtered_data = monthly_data[monthly_data['Category'].isin(
              ['Criminal and Civil Actions', 'State Enforcement Agencies'])]
category_chart = alt.Chart(filtered_data).mark_line().encode(
  x='YearMonth:T',
  y='sum(Action Count):Q',
  color='Category:N'
).properties(
  title="Number of Enforcement Actions by Category Over Time",
  width=400,
  height=300
)

category_chart
```



- based on five topics

```
def categorize_topic(title):
  title = title.lower()

  # Check for Health Care Fraud
  if "health care" in title or "medicare" in title or "medicaid" in title:
    return "Health Care Fraud"
  # Check for Financial Fraud
  elif "financial" in title or "wire" in title or "bank" in title or "fraud"
   ↪  in title:
    return "Financial Fraud"
  # Check for Drug Enforcement
  elif "drug" in title or "opioid" in title or "narcotics" in title:
    return "Drug Enforcement"
  # Check for Bribery/Corruption
  elif "bribery" in title or "corruption" in title:
    return "Bribery/Corruption"
  else:
    return "Other"
```

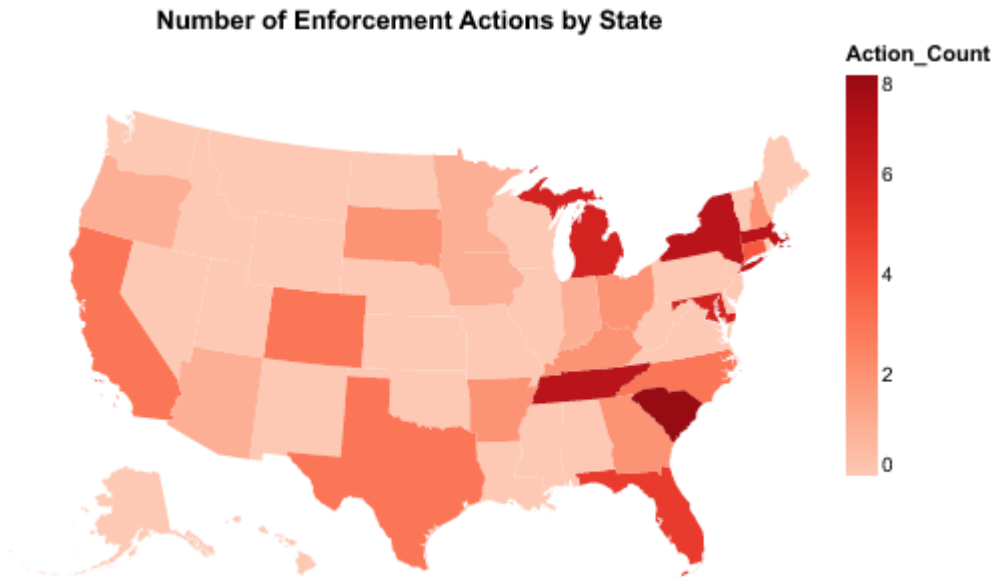
```
df['Topic'] = df['Title'].apply(categorize_topic)
monthly_data = df.groupby(['YearMonth', 'Category', 'Topic']).agg({
                    'Title': 'count'}).reset_index().rename(columns={'Title':
                     ↪  'Action Count'})
filtered_data = monthly_data[monthly_data['Category'].isin(
              ['Criminal and Civil Actions'])]
topic_chart = alt.Chart(filtered_data).mark_line().encode(
  x='YearMonth:T',
  y='sum(Action Count):Q',
  color='Topic:N'
).properties(
  title="Number of Enforcement Actions by Topic Over Time",
  width=400,
  height=300
)

topic_chart
```



Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

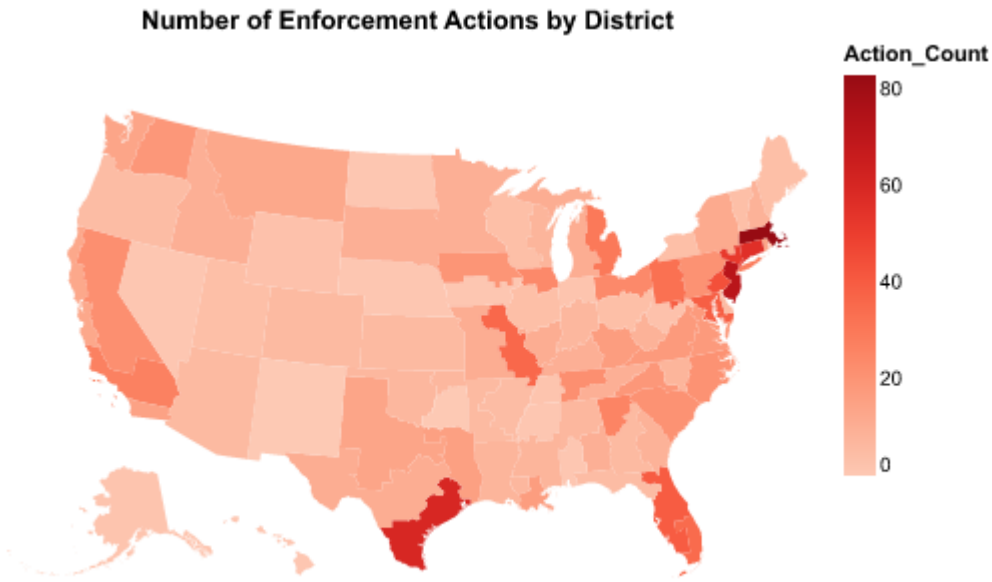
```
import geopandas as gpd

state_actions = df[df['Agency'].str.contains("State of ", na=False)]

def extract_state(agency):
  if "State of" in agency:
    start_index = agency.find("State of") + len("State of")
    state_name = agency[start_index:].strip()
    return state_name
  else:
    return None

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

state_summary = state_actions.groupby('State').agg(
  Action_Count=('Title', 'count')
).reset_index()

state_gdf = gpd.read_file("cb_2018_us_state_500k.shp")
state_gdf = state_gdf.merge(state_summary, left_on='NAME', right_on='State',
 ↪  how='left').fillna(0)
state_df = state_gdf[['geometry', 'NAME', 'Action_Count']]

state_map = alt.Chart(state_df).mark_geoshape().encode(
  color=alt.Color('Action_Count:Q', scale=alt.Scale(scheme='reds'))
).properties(
  title="Number of Enforcement Actions by State",
  width=400,
  height=300
).project('albersUsa')

state_map
```



2. Map by District (PARTNER 2)

```
district_actions = df[df['Agency'].str.contains("District", na=False)]

def extract_district(agency):
  parts = agency.split(',')

  for part in parts:
    part = part.strip()
    if "District" in part:
      return part

  return None

district_actions['District'] =
 ↪  district_actions['Agency'].apply(extract_district)

district_summary = district_actions.groupby('District').agg(
  Action_Count=('Title', 'count')
).reset_index()
```

```
district_gdf =
 ↪  gpd.read_file("geo_export_d961e3f8-9a3b-4044-96c5-7d417502905e.shp")
district_gdf = district_gdf.merge(district_summary, left_on='judicial_d',
 ↪  right_on='District', how='left').fillna(0)
district_df = district_gdf[['geometry', 'judicial_d', 'Action_Count']]
district_df = district_df.rename(columns={'judicial_d': 'Judicial District'})

district_map = alt.Chart(district_df).mark_geoshape().encode(
  color=alt.Color('Action_Count:Q', scale=alt.Scale(scheme='reds'))
).properties(
  title="Number of Enforcement Actions by District",
  width=400,
  height=300
).project('albersUsa')

district_map
```



Extra Credit

1. Merge zip code shapefile with population

```
zip_data = pd.read_csv('DECENNIALDHC2020.P1-Data.csv')
zip_gdf = gpd.read_file("gz_2010_us_860_00_500k.shp")

zip_data = zip_data[zip_data['P1_001N'].str.isnumeric()]
zip_data['ZIP'] = zip_data['NAME'].str.replace("ZCTA5 ", "", regex=False)
zip_data['Population'] = zip_data['P1_001N'].astype(int)

zip_data = zip_data[['ZIP', 'Population']]
zip_gdf = zip_gdf.merge(zip_data, how='left', left_on='ZCTA5',
 ↪  right_on='ZIP')
```

2. Conduct spatial join

```
joined_gdf = gpd.sjoin(zip_gdf[['ZIP', 'Population', 'geometry']],
 ↪  district_gdf[['judicial_d', 'geometry']], how='inner',
 ↪  predicate='intersects')

population_by_district =
 ↪  joined_gdf.groupby('judicial_d')['Population'].sum().reset_index()
district_pop_gdf = district_gdf.merge(population_by_district,
 ↪  on='judicial_d', how='left')
```

3. Map the action ratio in each district

```
district_pop_gdf['Action_Per_Capita'] = district_pop_gdf['Action_Count'] /
 ↪  district_pop_gdf['Population']
district_pop_gdf = district_pop_gdf[['geometry', 'judicial_d',
 ↪  'Action_Count', 'Action_Per_Capita']]
district_pop_gdf = district_pop_gdf.rename(columns={'judicial_d': 'Judicial
 ↪  District'})

district_pop_map = alt.Chart(district_pop_gdf).mark_geoshape().encode(
    color=alt.Color('Action_Per_Capita:Q', scale=alt.Scale(scheme='reds'),
 ↪  title='Actions per Capita')
```

```
).properties(
    title="Ratio of Enforcement Actions per Capita by District",
    width=400,
    height=300
).project('albersUsa')

district_pop_map
```