Pset5

Yuan Qi, Aurora Zhang

2024-11-04

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): Yuan Qi, yuanqi
    - Partner 2 (name and cnet ID): AuroraZhang, zhanght
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: **_YQ_** **_AZ_**
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: **_2_**
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 geopandas as gpd
import matplotlib.pyplot as plt
import time
import requests
from bs4 import BeautifulSoup
import lxml
import re
import warnings
from selenium import webdriver
from datetime import datetime
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
# Get webpage content and parse it
url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Initialize lists to store each category, title, date, and link
categories = []
titles = []
dates = []
links = []

# Find all `li` tags that contain article information
actions = soup.find_all('li', class_='usa-card card--list pep-card--minimal
 ↪  mobile:grid-col-12')

# Loop through each `li` element to extract the title, date, category, and
 ↪  link
for i, action in enumerate(actions, start=1):
    # Find the title and link
```

```
    title_tag = action.find('h2', class_='usa-card__heading')
    title = title_tag.get_text(strip=True) if title_tag else None
    link = title_tag.find('a')['href'] if title_tag and title_tag.find('a')
↪   else None
    full_link = f"https://oig.hhs.gov{link}" if link else None

    # Find the date
    date_tag = action.find('span', class_='text-base-dark padding-right-105')
    date = date_tag.get_text(strip=True) if date_tag else None

    # Find the category
    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.get_text(strip=True) if category_tag else None

    # Add each category, title, date, and link to their respective lists
    categories.append(category)
    titles.append(title)
    dates.append(date)
    links.append(full_link)

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

# Print the result
print(actions.head())

# Save to local file
actions.to_csv(r'C:\Users\freya\Desktop\24 fall
↪   study\Python2\Pset5\scraped_data.csv', index=False, encoding='utf-8')
```

```
                    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
```

```
                                                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

                                                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
# Load the csv file from the last question
sd = pd.read_csv(r'C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\scraped_data.csv')

# Initialize the dataframe for agencies
agencies = []

# Loop through each link in the sd dataset
for link in sd['Link']:
    if pd.notna(link):
        response = requests.get(link)
        soup = BeautifulSoup(response.text, 'html.parser')

        # Search the information of agency
        agency_tag = soup.find('span', class_='padding-right-2 text-base',
 ↪  text="Agency:")
        agency = agency_tag.next_sibling.strip() if agency_tag else None

        # Append 'Agency' information to the dataframe
        agencies.append(agency)
    else:
        agencies.append(None)

# Add `Agency` column to sd dataset
```

```
sd['Agency'] = agencies

# Print and save the updated scraped data
print(sd.head())
sd.to_csv(r'C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\scraped_data_with_agency.csv', index=False,
 ↪  encoding='utf-8')
```

```
                    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

                                               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

                                                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...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2) Define function `scrape_enforcement_actions(month, year)`: # Step 1: Check if year is valid If `year < 2013`: Print warning message Return None

  # Step 2: Initialize lists for data, starting page, define month and year Initialize `titles`, `dates`, `categories`, `links` as empty lists Set `page = 1` Define the earliest date to scrape (specified by user)

  # Step 3: Construct the base URL

  # Step 4: Start pagination loop, use 'While' loop, there's 480 pages in total While `page <= 480`: # Construct the full URL for the current page Set `url = base_url + page`

  ```
  # Send request and parse HTML
  Send GET request to `url`
  Parse the HTML with BeautifulSoup

  # Step 5: Check if the date of the action is starting from that
  month+year to today and Extract qualifying data from each enforcement
  action on the page
  Use a for loop to iterate actions.
  For each action:
      Extract `title`, `date`, `category`, `link`
      Append extracted values to `titles`, `dates`, `categories`,
      `links`

  # Step 6: Increment page and wait before the next request
  Increment `page` by 1
  Wait for 1 second
  ```

  # Step 7: Create DataFrame and save data Create DataFrame with `titles`, `dates`, `categories`, `links` Save DataFrame as `enforcement_actions_year_month.csv`

  Return DataFrame

- b. Create Dynamic Scraper (PARTNER 2)

```python
def scrape_enforcement_actions(month, year):
    # Step 1: Check if the year is valid (should be >= 2013)
    if year < 2013:
        print("Please provide a year >= 2013 as only enforcement actions
        ↪  after 2013 are available.")
```

```
        return None

    # Step 2: Initialize lists to store data and starting page number
    titles, dates, categories, links, agencies = [], [], [], [], []
    page = 1  # start from the first page
    start_date = datetime(year, month, 1)
    base_url = f'https://oig.hhs.gov/fraud/enforcement/?page='
    stop_scraping = False  # Flag to indicate when to stop scraping

    # Step 3: Get the total number of pages
    try:
        initial_response = requests.get(base_url + str(page), timeout=10)
        initial_response.raise_for_status()
        initial_soup = BeautifulSoup(initial_response.text, 'html.parser')

        # Extract total number of pages from pagination
        pagination = initial_soup.find('ul', class_='pagination')
        if pagination:
            last_page_link =
↪   pagination.find_all('a')[-2].get_text(strip=True)  # Get the
↪   second-to-last page number
            total_pages = int(last_page_link)
        else:
            total_pages = 1  # If pagination is not found, assume only one
↪   page
        print(f"Total pages found: {total_pages}")
    except Exception as e:
        print(f"Error fetching total pages: {e}")
        return None

    # Step 4: Start pagination loop
    while page <= total_pages and not stop_scraping:
        url = base_url + str(page)
        print(f"Scraping page {page} of {total_pages}: {url}")  # Debug:
         ↪  Print the URL being scraped

        try:
            # Send the GET request and parse the HTML content
            response = requests.get(url, timeout=10)
            response.raise_for_status()
            soup = BeautifulSoup(response.text, 'html.parser')
```

```
            # Find all enforcement action elements on the page
            actions = soup.find_all('li', class_='usa-card card--list
↪  pep-card--minimal mobile:grid-col-12')

            # If no actions are found, break the loop
            if not actions:
                print("No more actions found, stopping the scraping.")
                break

            # Step 5: Extract data from each enforcement action
            for action in actions:
                date_tag = action.find('span', class_='text-base-dark
↪  padding-right-105')
                date_str = date_tag.get_text(strip=True) if date_tag else
↪  None
                if date_str:
                    action_date = pd.to_datetime(date_str)
                    if action_date < start_date:
                        print(f"Reached data before {start_date}, stopping
                         ↪  the scraping.")
                        stop_scraping = True  # Set flag to stop scraping
                        break  # Exit the inner loop

                    dates.append(date_str)

                title_tag = action.find('h2', class_='usa-card__heading')
                title = title_tag.get_text(strip=True) if title_tag else None
                link = title_tag.find('a')['href'] if title_tag and
↪  title_tag.find('a') else None
                full_link = f"https://oig.hhs.gov{link}" if link else None

                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.get_text(strip=True) if category_tag
↪  else None

                titles.append(title)
                categories.append(category)
                links.append(full_link)

                if full_link:
```

```
                    try:
                        action_response = requests.get(full_link, timeout=10)
                        action_response.raise_for_status()
                        action_soup = BeautifulSoup(action_response.text,
↪  'html.parser')

                        agency_tag = action_soup.find('span',
↪  class_='padding-right-2 text-base', text="Agency:")
                        agency = agency_tag.next_sibling.strip() if
↪  agency_tag else None
                        agencies.append(agency)
                    except Exception as e:
                        print(f"Error fetching agency info from {full_link}:
                         ↪  {e}")
                        agencies.append(None)
                else:
                    agencies.append(None)

            # Continue to the next page only if we haven't set the
             ↪  stop_scraping flag
            if not stop_scraping:
                page += 1
                time.sleep(1)

        except Exception as e:
            print(f"Error fetching page {page}: {e}")
            break

    # Step 6: Create DataFrame and save as CSV
    df = pd.DataFrame({
        'Title': titles,
        'Date': dates,
        'Category': categories,
        'Link': links,
        'Agency': agencies
    })

    csv_filename = f"enforcement_actions_{year}_{month:02d}.csv"
    df.to_csv(csv_filename, index=False, encoding='utf-8')
    print(f"Data saved to {csv_filename}")

    return df
```

```
# Example usage of the function
df_2023 = scrape_enforcement_actions(1, 2023)
print(df_2023.head())
```

```
# Load the dataframe already scraped
df_2023 = pd.read_csv(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\enforcement_actions_2023_01.csv")

# The number of enforcement actions
print(f"The number of enforcement actions started from January 2023:")
print(len(df_2023))

# The eariest enforcement action
print("The shape of the eariest enforcement action:")
print(df_2023.iloc[-1])
```

```
The number of enforcement actions started from January 2023:
1534
The shape of the eariest enforcement action:
Title       Podiatrist Pays $90,000 To Settle False Billin...
Date                                            January 3, 2023
Category                            Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/podiatri...
Agency      U.S. Attorney's Office, Southern District of T...
Name: 1533, dtype: object
```

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

```
# Example usage of the function, scraping from January 2021
df_2021 = scrape_enforcement_actions(1, 2021)
print(df_2021.head())
```

```
# Load the dataframe already scraped
df_2021 = pd.read_csv(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\enforcement_actions_2021_01.csv")

# The number of enforcement actions
print(f"The number of enforcement actions started from January 2021:")
print(len(df_2021))
```

```
# The eariest enforcement action
print("The shape of the eariest enforcement action:")
print(df_2021.iloc[-1])
```

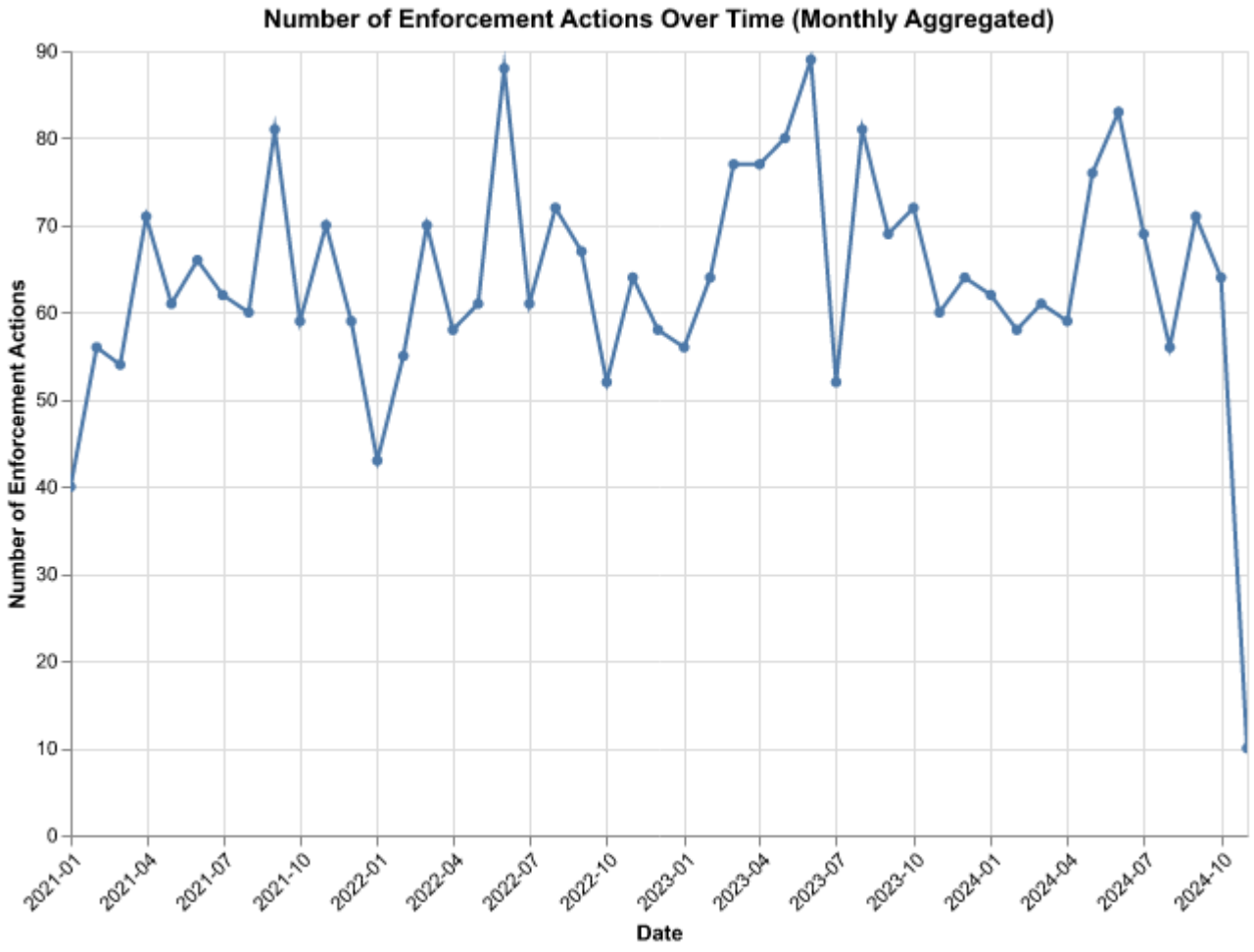
```
The number of enforcement actions started from January 2021:
2998
The shape of the eariest enforcement action:
Title       The United States And Tennessee Resolve Claims...
Date                                            January 4, 2021
Category                            Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/the-unit...
Agency      U.S. Attorney's Office, Middle District of Ten...
Name: 2997, dtype: object
```

Step 3: Plot data based on scraped data

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

```
# Load the data from the CSV file
df_2021 = pd.read_csv(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\enforcement_actions_2021_01.csv")

# 1. Convert 'Date' column from text to datetime format
df_2021['Date'] = pd.to_datetime(df_2021['Date'], errors='coerce')

# 2. Aggregate data by month and year
# Convert each date to a "YearMonth" period
df_2021['YearMonth'] = df_2021['Date'].dt.to_period('M')

# 3. Group data by "YearMonth" and count occurrences to get the number of
 ↪  actions per month
monthly_counts =
 ↪  df_2021.groupby('YearMonth').size().reset_index(name='Enforcement
 ↪  Actions')

# 4. Convert "YearMonth" back to timestamp format so Altair can interpret it
 ↪  as a date
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()

# 5. Create the Altair line chart
```

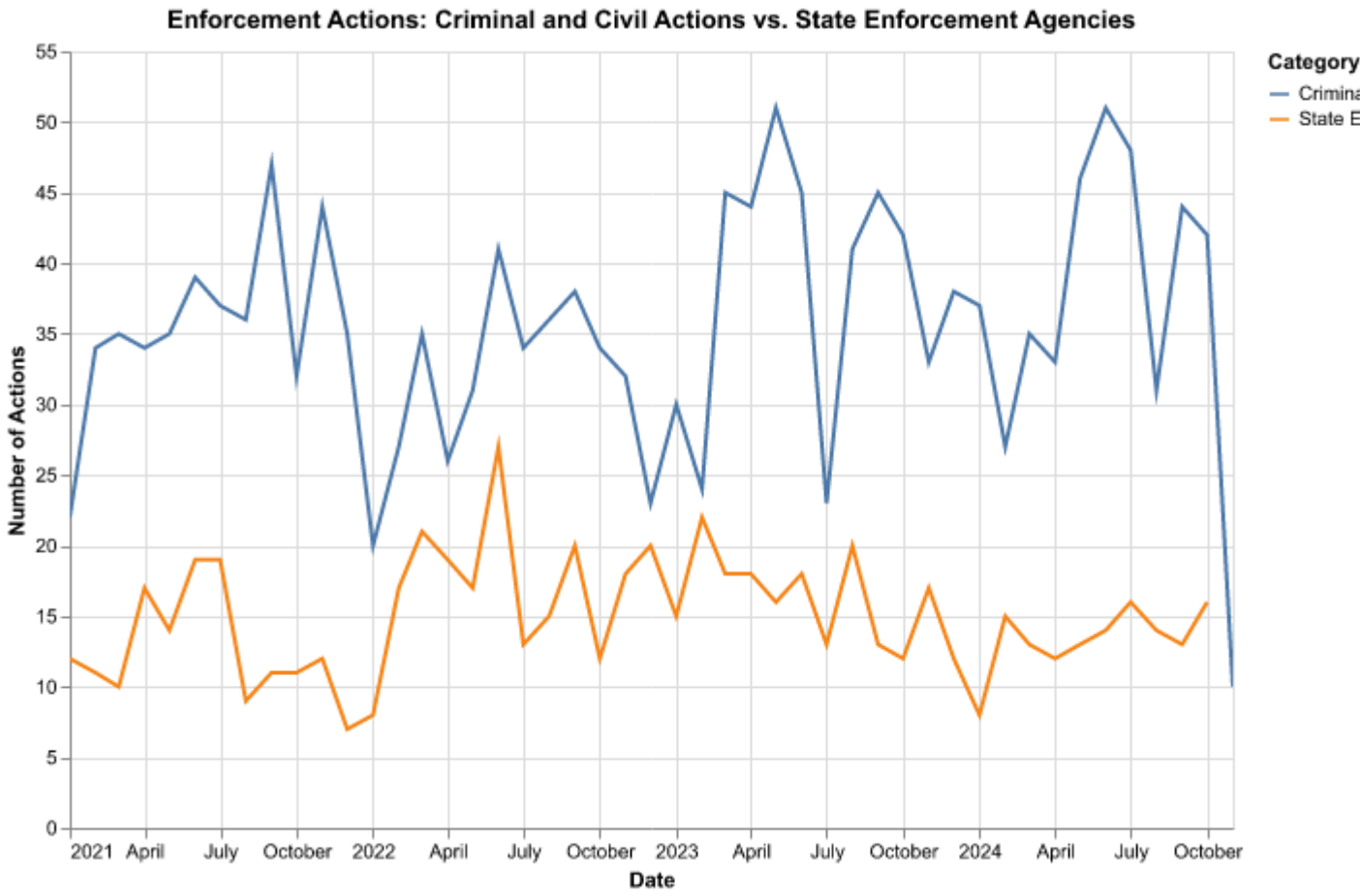
```
line_chart = alt.Chart(monthly_counts).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Date', axis=alt.Axis(format='%Y-%m',
    ↪  labelAngle=-45)),
    y=alt.Y('Enforcement Actions:Q', title='Number of Enforcement Actions'),
    tooltip=['YearMonth:T', 'Enforcement Actions:Q']  # Display month and
    ↪  count on hover
).properties(
    title='Number of Enforcement Actions Over Time (Monthly Aggregated)',
    width=600,
    height=400
).interactive()  # Enable interactive zooming and panning

# Display the chart
line_chart.show()
```



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

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

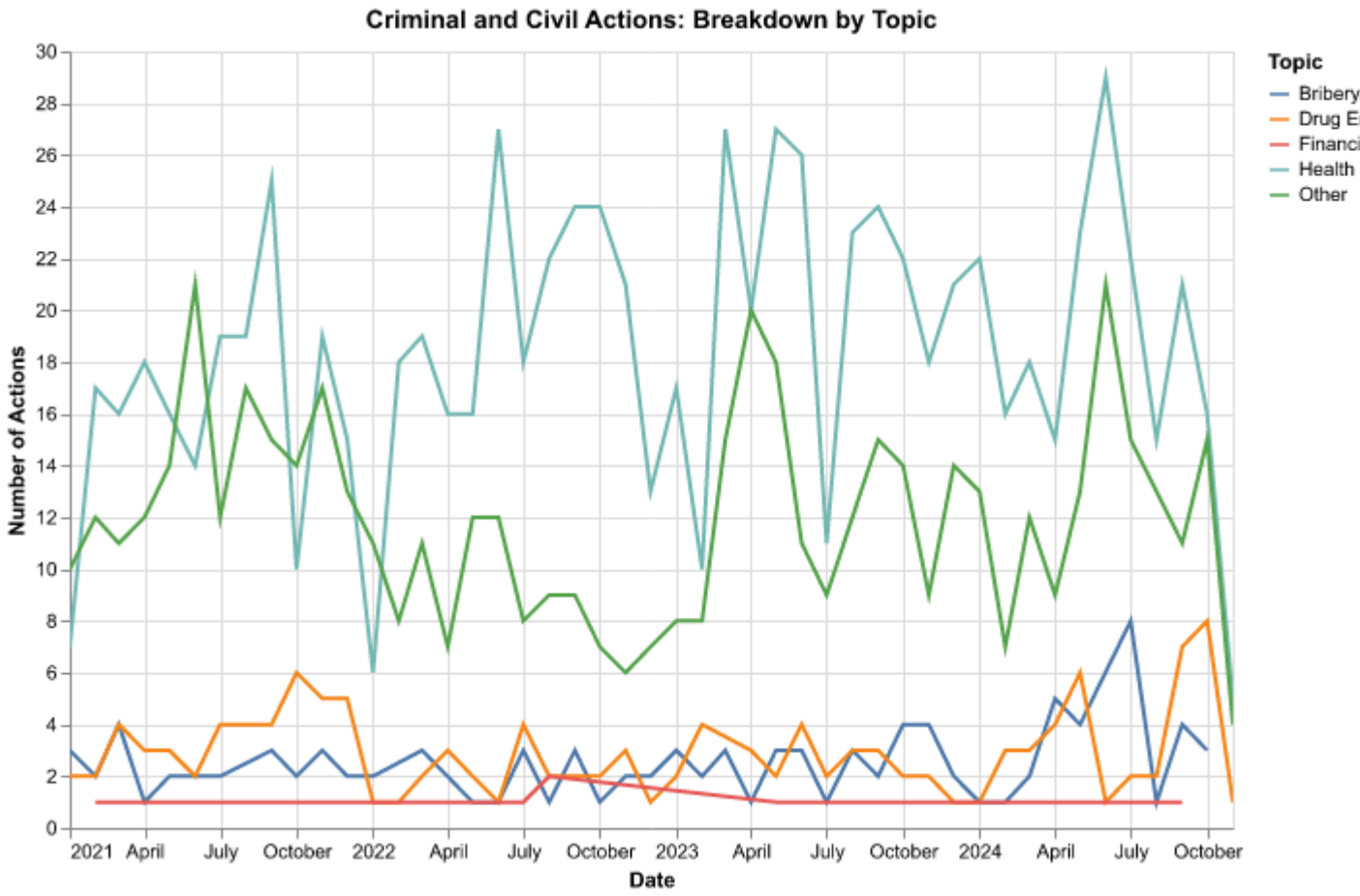
```
# Load the data from the CSV file
df_2021 = pd.read_csv(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\enforcement_actions_2021_01.csv")

# Filter and create a subset with only the desired categories: "Criminal and
 ↪  Civil Actions" and "State Enforcement Agencies"
df_subset = df_2021[df_2021["Category"].isin(["Criminal and Civil Actions",
 ↪  "State Enforcement Agencies"])]

# Convert 'Date' to datetime
df_subset["Date"] = pd.to_datetime(df_subset["Date"])

### Chart 1: "Criminal and Civil Actions" vs. "State Enforcement Agencies"
# Group by 'Date' and 'Category' to get counts
df_grouped_category = df_subset.groupby([df_subset["Date"].dt.to_period("M"),
 ↪  "Category"]).size().reset_index(name="Count")
df_grouped_category["Date"] = df_grouped_category["Date"].dt.to_timestamp()
 ↪  # Convert to timestamp for Altair

# Plot the first chart
chart1 = alt.Chart(df_grouped_category).mark_line().encode(
    x=alt.X("Date:T", title="Date"),
    y=alt.Y("Count:Q", title="Number of Actions"),
    color=alt.Color("Category:N", title="Category")
).properties(
    title="Enforcement Actions: Criminal and Civil Actions vs. State
 ↪  Enforcement Agencies",
    width=600,
    height=400
).interactive()
chart1.show()
```



- based on five topics

```
### Chart 2: Breakdown of "Criminal and Civil Actions" by Five Topics
# Define a function to classify topics based on 'Title'
def classify_topic(title):
    if "Health Care Fraud" in title or "Healthcare" in title or "Health Care"
     ↪  in title or "Medicaid" in title or "Medicare" in title or "Medical"
     ↪  in title:
        return "Health Care Fraud"
    elif "Financial" in title or "Bank" in title:
        return "Financial Fraud"
    elif "Drug" in title or "Opioid" in title or "Pharmaceutical" in title:
        return "Drug Enforcement"
    elif "Bribe" in title or "Corruption" in title or "Kickback" in title or
     ↪  "Embezzlement" in title:
        return "Bribery/Corruption"
```

```
    else:
        return "Other"

# Apply the function to create a new 'Topic' column for "Criminal and Civil
 ↪  Actions"
df_criminal_civil = df_subset[df_subset["Category"] == "Criminal and Civil
 ↪  Actions"].copy()
df_criminal_civil["Topic"] = df_criminal_civil["Title"].apply(classify_topic)

# Group by 'Date' and 'Topic' to get counts
df_grouped_topic =
 ↪  df_criminal_civil.groupby([df_criminal_civil["Date"].dt.to_period("M"),
 ↪  "Topic"]).size().reset_index(name="Count")
df_grouped_topic["Date"] = df_grouped_topic["Date"].dt.to_timestamp()  #
 ↪  Convert to timestamp for Altair

# Plot the second chart
chart2 = alt.Chart(df_grouped_topic).mark_line().encode(
    x=alt.X("Date:T", title="Date"),
    y=alt.Y("Count:Q", title="Number of Actions"),
    color=alt.Color("Topic:N", title="Topic")
).properties(
    title="Criminal and Civil Actions: Breakdown by Topic",
    width=600,
    height=400
).interactive()

chart2.show()
```

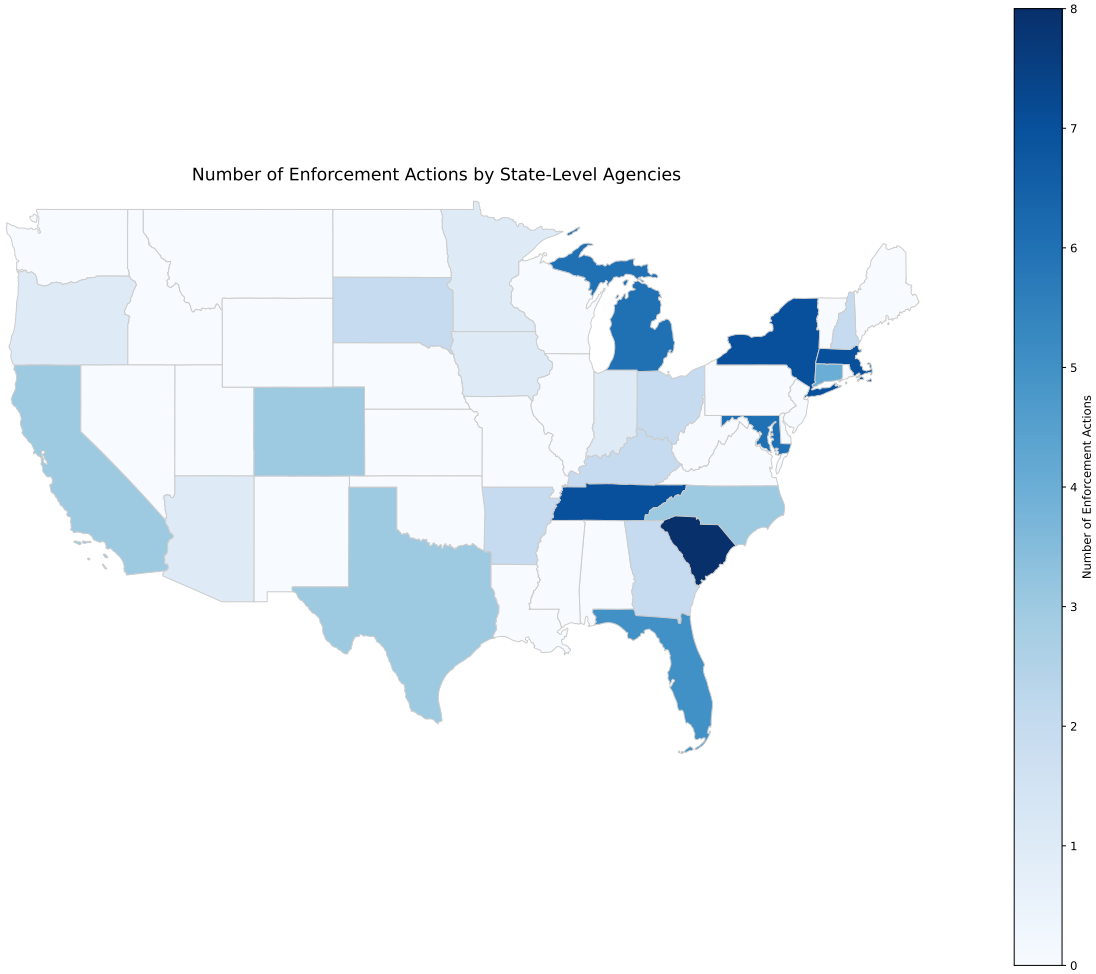


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
# Load the data from the CSV file
df_2021 = pd.read_csv(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\enforcement_actions_2021_01.csv")

# Filter for agencies containing "State of"
df_state_agencies = df_2021[df_2021["Agency"].str.contains("State of",
 ↪  na=False)]

# Use a function to extract the state name following "State of"
def extract_state(agency):
```

```
    if "State of" in agency:
        return agency.split("State of")[-1].strip()
    return None

# Apply the function to create a new 'State' column
df_state_agencies["State"] = df_state_agencies["Agency"].apply(extract_state)

# Group by 'State' to get the count of enforcement actions for each state
state_counts = df_state_agencies["State"].value_counts().reset_index()
state_counts.columns = ["State", "Enforcement_Action"]

#Load the shapefile of U.S. states using geopandas
state_shp = gpd.read_file(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\cb_2018_us_state_20m.shp")

# Merge the state counts with the shapefile data
merged_state_shp = state_shp.merge(state_counts, left_on="NAME",
 ↪  right_on="State", how="left")
merged_state_shp["Enforcement_Action"] =
 ↪  merged_state_shp["Enforcement_Action"].fillna(0)  # Fill NaN values with
 ↪  0

# Plot the choropleth map
fig, ax = plt.subplots(1, 1, figsize=(20, 15))
merged_state_shp.plot(column='Enforcement_Action',
                      cmap='Blues',
                      linewidth=0.8,
                      ax=ax,
                      edgecolor='0.8',
                      legend=True,
                      legend_kwds={'label': "Number of Enforcement Actions",
                                   'orientation': "vertical"})
ax.set_title('Number of Enforcement Actions by State-Level Agencies',
             fontdict={'fontsize': '15', 'fontweight': '3'})
ax.set_xlim([-130, -65])
ax.set_ylim([23, 50])
ax.axis('off')
plt.show()
```

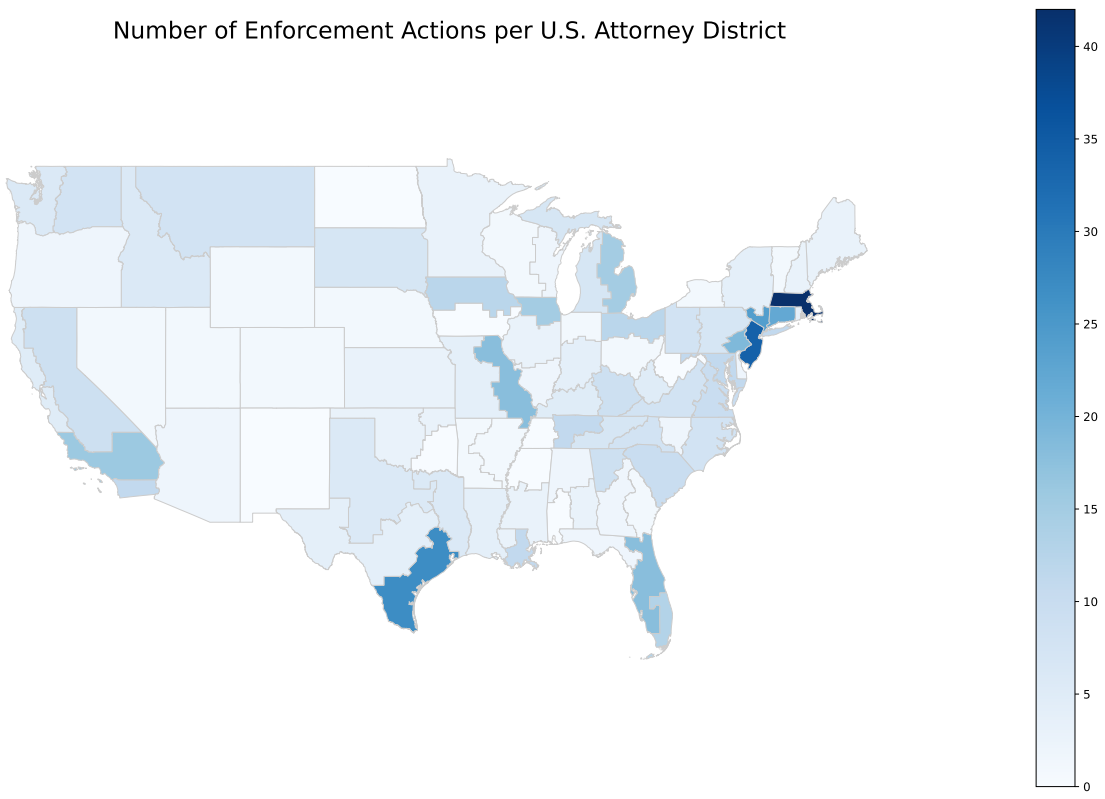


2. Map by District (PARTNER 2)

```
districts = gpd.read_file(r'C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\US Attorney Districts Shapefile
 ↪  simplified_20241108\geo_export_4cfb3345-bfd9-4002-81b4-dea50f88d0c5.shp')

# Clean the Agency column to retain only district information
df_2021['judicial_d'] = df_2021['Agency'].str.replace("U.S. Attorney's
 ↪  Office,", '', regex=True).str.strip()

# Count enforcement actions per district
district_counts = df_2021['judicial_d'].value_counts().reset_index()
```

```
district_counts.columns = ['judicial_d', 'Enforcement_Actions_Counts']

# Merge with district counts
merged_district_shp = districts.merge(district_counts, on='judicial_d',
 ↪  how='left').fillna(0)
```

```
fig, ax = plt.subplots(figsize=(20, 15))
merged_district_shp .plot(column='Enforcement_Actions_Counts', cmap='Blues',
 ↪  linewidth=0.8, edgecolor= '0.8', ax=ax, legend= True,
 ↪  legend_kwds={'orientation': 'vertical', 'shrink': 0.8})
ax.set_title('Number of Enforcement Actions per U.S. Attorney District',
 ↪  fontsize=20)

ax.set_axis_off()
ax.set_xlim(-130, -60)
ax.set_ylim(20, 55)
plt.show()
```



Extra Credit

1. Merge zip code shapefile with population

```
# Load the ZIP code shapefile
zip_shp = gpd.read_file(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset4\gz_2010_us_860_00_500k.shp")

# Load the population data
pop_data = pd.read_csv( r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\DECENNIALDHC2020.P1_2024-11-08T160209\DECENNIALDHC2020.P1-Data.csv")
print(pop_data.head())

# Extract ZIP code numbers from the "NAME" column
pop_data["ZCTA5"] = pop_data["NAME"].str.split(" ").str[-1]

# Inspect the data to confirm the ZIP codes were extracted correctly
print(pop_data.head())

# Merge the shapefile with the population data
merged_pop_zip = pop_data.merge(zip_shp, on="ZCTA5", how="left")
```

```
           GEO_ID                  NAME   P1_001N  Unnamed: 3
0       Geography  Geographic Area Name   !!Total         NaN
1  860Z200US00601            ZCTA5 00601     17242         NaN
2  860Z200US00602            ZCTA5 00602     37548         NaN
3  860Z200US00603            ZCTA5 00603     49804         NaN
4  860Z200US00606            ZCTA5 00606      5009         NaN
           GEO_ID                  NAME   P1_001N  Unnamed: 3  ZCTA5
0       Geography  Geographic Area Name   !!Total         NaN   Name
1  860Z200US00601            ZCTA5 00601     17242         NaN  00601
2  860Z200US00602            ZCTA5 00602     37548         NaN  00602
3  860Z200US00603            ZCTA5 00603     49804         NaN  00603
4  860Z200US00606            ZCTA5 00606      5009         NaN  00606
```

2. Conduct spatial join

```
# Load the district code shapefile
districts_shp = gpd.read_file(r'C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\US Attorney Districts Shapefile
 ↪  simplified_20241108\geo_export_4cfb3345-bfd9-4002-81b4-dea50f88d0c5.shp')
```

```
# Load the ZIP code shapefile
zip_shp = gpd.read_file(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset4\gz_2010_us_860_00_500k.shp")

# Conduct a spatial joint between Zipcode shapefile and District shapefile
zip_district_join = gpd.sjoin(zip_shp, districts_shp, how="inner",
 ↪  predicate="within")
print(zip_district_join.head())

# Merge the population data with the `zip_district_join` result
zip_district_pop = zip_district_join.merge(pop_data, on="ZCTA5", how="left")
print(zip_district_pop[['judicial_d','ZCTA5', 'P1_001N']].head(10))

# Check and convert the 'P1_001N' column to numeric format
zip_district_pop["P1_001N"] = pd.to_numeric(zip_district_pop["P1_001N"],
 ↪  errors="coerce")

# Aggregate: Group by 'judicial_d' and calculate the total population
district_pop_agg = zip_district_pop.groupby("judicial_d").agg({"P1_001N":
 ↪  "sum"}).reset_index()

district_pop_agg.to_csv(r"C:\Users\freya\Desktop\24 fall
 ↪  study\Python2\Pset5\district_pop_agg.csv", index=False)
```

```
           GEO_ID  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
5  8600000US01060  01060  01060  ZCTA5      10.918

                                            geometry  index_right statefp  \
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...           50      25
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...           50      25
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...           50      25
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...           50      25
5  POLYGON ((-72.65724 42.36369, -72.65709 42.363...           50      25

                 judicial_d         aland  ...          state     chief_judg
                 \
0  District of Massachusetts  2.020513e+10  ...  Massachusetts  Dennis Saylor
```

```
1  District of Massachusetts  2.020513e+10  ...  Massachusetts  Dennis Saylor
2  District of Massachusetts  2.020513e+10  ...  Massachusetts  Dennis Saylor
3  District of Massachusetts  2.020513e+10  ...  Massachusetts  Dennis Saylor
5  District of Massachusetts  2.020513e+10  ...  Massachusetts  Dennis Saylor

           nominating term_as_ch  shape_leng  shape_area  abbr district_n  \
0  George W. Bush (R)     2019.0   17.629736    2.318549    MA          1
1  George W. Bush (R)     2019.0   17.629736    2.318549    MA          1
2  George W. Bush (R)     2019.0   17.629736    2.318549    MA          1
3  George W. Bush (R)     2019.0   17.629736    2.318549    MA          1
5  George W. Bush (R)     2019.0   17.629736    2.318549    MA          1

     shape__are    shape__len
0  3.881867e+10  2.199726e+06
1  3.881867e+10  2.199726e+06
2  3.881867e+10  2.199726e+06
3  3.881867e+10  2.199726e+06
5  3.881867e+10  2.199726e+06

[5 rows x 21 columns]
                 judicial_d  ZCTA5 P1_001N
0  District of Massachusetts  01040   38238
1  District of Massachusetts  01050    2467
2  District of Massachusetts  01053    2031
3  District of Massachusetts  01056   21002
4  District of Massachusetts  01060   14461
5  District of Massachusetts  01062   10910
6  District of Massachusetts  01066     215
7  District of Massachusetts  01069    8428
8  District of Massachusetts  01070     577
9  District of Massachusetts  01072    1366
```

3. Map the action ratio in each district

```
# Merge the merged_district_shp and district_pop_agg
merged_district_enforcement = merged_district_shp.merge(district_pop_agg,
 ↪  on="judicial_d", how="left")

# Calculate the ratio of enforcement on Population
merged_district_enforcement['Action_Pop_Ratio'] =
 ↪  merged_district_enforcement['Enforcement_Actions_Counts'] /
 ↪  merged_district_enforcement['P1_001N']
```

```
print(merged_district_enforcement)

# Plot the mapping result
fig, ax = plt.subplots(figsize=(25, 20))
merged_district_enforcement.plot(
    column='Action_Pop_Ratio',
    cmap='Blues',
    linewidth=1,
    edgecolor='0.8',
    ax=ax,
    legend=True,
    legend_kwds={'orientation': 'vertical', 'shrink': 0.8}
)
ax.set_title('Ratio of Enforcement Actions per Population in U.S. Attorney
 ↪  Districts', fontsize=20)
ax.set_axis_off()
ax.set_xlim(-130, -60)
ax.set_ylim(20, 55)
plt.show()
```

```
   statefp                            judicial_d         aland        awater
   \
0       21           Western District of Kentucky  4.970555e+10  1.651516e+09
1       21           Eastern District of Kentucky  5.257394e+10  7.238213e+08
2       18           Southern District of Indiana  5.824517e+10  5.941176e+08
3       01             Middle District of Alabama  3.412673e+10  5.472423e+08
4       01           Southern District of Alabama  6.235882e+10  3.052681e+09
..     ...                                    ...           ...           ...
89      69  District of Northern Marianas Islands  4.722925e+08  4.644252e+09
90      12           Southern District of Florida  2.448809e+10  1.159277e+10
91      40          Northern District of Oklahoma  2.231989e+10  7.189768e+08
92      50                    District of Vermont  2.387418e+10  1.030417e+09
93      10                   District of Delaware  5.045926e+09  1.399986e+09

                      state           chief_judg            nominating  \
0                  Kentucky      Greg N. Stivers      Barack Obama (D)
1                  Kentucky         Danny Reeves    George W. Bush (R)
2                   Indiana  Jane Magnus-Stinson      Barack Obama (D)
3                   Alabama    Emily Coody Marks      Donald Trump (R)
4                   Alabama        Kristi DuBose    George W. Bush (R)
..                      ...                  ...                   ...
```

```
89  Northern Marianas Islands   Ramona V. Manglona      Barack Obama (D)
90                    Florida     K. Michael Moore  George H.W. Bush (R)
91                   Oklahoma         John Dowdell      Barack Obama (D)
92                    Vermont    Geoffrey Crawford      Barack Obama (D)
93                   Delaware        Leonard Stark      Barack Obama (D)

    term_as_ch  shape_leng  shape_area abbr district_n    shape__are  \
0       2018.0   16.200585    5.216899  KYW          6  8.123902e+10
1       2019.0   13.514251    5.451047  KYE          6  8.547129e+10
2       2016.0   14.956126    6.137433  INS          7  9.818187e+10
3       2019.0   10.235799    3.858442  ALM         11  5.645450e+10
4       2017.0   12.976906    3.278871  ALS         11  4.772733e+10
..         ...         ...         ...  ...        ...           ...
89      2011.0    3.252892    0.040335   MP          9  5.200276e+08
90      2014.0   17.941594    2.503158  FLS         11  3.466156e+10
91      2019.0    8.154257    2.316496  OKN         10  3.569729e+10
92      2017.0    9.571257    2.797955   VT          2  4.826994e+10
93      2014.0    4.285079    0.541590   DE          3  8.634830e+09

      shape__len                                           geometry  \
0   1.964255e+06  MULTIPOLYGON (((-89.48248 36.50214, -89.48543 ...
1   1.654681e+06  POLYGON ((-84.62012 39.07346, -84.60793 39.073...
2   1.887626e+06  POLYGON ((-85.86281 40.46476, -85.86212 40.406...
3   1.236201e+06  POLYGON ((-85.33828 33.49471, -85.33396 33.492...
4   1.567095e+06  MULTIPOLYGON (((-88.08682 30.25987, -88.07676 ...
..           ...                                                ...
89  3.702108e+05  MULTIPOLYGON (((145.28433 14.17537, 145.28404 ...
90  2.117424e+06  MULTIPOLYGON (((-81.9667 24.52376, -81.97837 2...
91  9.857548e+05  POLYGON ((-95.04951 36.99959, -95.03786 36.999...
92  1.248962e+06  POLYGON ((-72.67477 45.01547, -72.58988 45.013...
93  5.622177e+05  MULTIPOLYGON (((-75.56246 39.51266, -75.56745 ...

    Enforcement_Actions_Counts    P1_001N  Action_Pop_Ratio
0                          5.0  1551786.0          0.000003
1                          9.0  1642124.0          0.000005
2                          4.0  3467235.0          0.000001
3                          3.0   943609.0          0.000003
4                          0.0   385301.0          0.000000
..                         ...        ...               ...
89                         0.0        NaN               NaN
90                        13.0  5702674.0          0.000002
91                         3.0   767664.0          0.000004
92                         1.0   490131.0          0.000002
```

```
93                              2.0    320940.0           0.000006

[94 rows x 18 columns]
```