Problem Set 5

Zijing Zhao, Zac Shen

2024-11-05

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: Zijing Zhao, (cnetid: zijingz)
    - Partner 2: Zekai Shen, (cnetid: zekaishen)
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: Zijing Zhao, Zekai Shen
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: Zijing used 1, Zekai used 1 Late coins left after submission: Zijing left 1, Zekai left 0
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 csv
import re
import geopandas as gpd
import json
import us
import altair_saver as alt_saver

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

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
def scrape_page(page_number):
  base_url = "https://oig.hhs.gov/fraud/enforcement/?page="
  url = base_url + str(page_number)
  response = requests.get(url)
  soup = BeautifulSoup(response.content, 'html.parser')
  # Extract the required data
  data = []
  # every information required is stored in the cards under
   ↪  'usa-card_container'
  cards = soup.find_all('div', class_ = 'usa-card__container')
  for card in cards:
    title = 'NA'
    link = 'NA'
    date = 'NA'
    category = 'NA'
    # title and link are under the same tag
    title_tag = card.find('h2')
    if title_tag:
      a_tag = title_tag.find('a')
      if a_tag and 'href' in a_tag.attrs:
        title =a_tag.text.strip()
        # this website uses relevant path
        link = 'https://oig.hhs.gov'+a_tag['href']
    # find the date
    date_tag = card.find('span')
    if date_tag:
      date = date_tag.text.strip()
    # find the category
    ul_tag = card.find('ul')
    if ul_tag:
      li_tag = ul_tag.find('li')
      category = li_tag.text.strip()
    data.append([title,date,category,link])
  return data

# Loop through all pages and scrape data
all_data = []
```

```
for page in range(1, 481):
    page_data = scrape_page(page)
    all_data.extend(page_data)
    time.sleep(0.1)

print(all_data)

df = pd.DataFrame(all_data, columns = ['Title', 'Date', 'Category', 'Link'])
print(df.head())

# Save the DataFrame to a CSV file
df.to_csv('scraped_data.csv', index=False)
```

```
df = pd.read_csv('scraped_data.csv')
print(df.head())
```

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

                    Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

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

2. Crawling (PARTNER 1)

```
import logging
# Set up logging configuration
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
 ↪  %(message)s')

df = pd.read_csv('scraped_data.csv')

def scrape_agency(link):
  response = requests.get(link)
  soup = BeautifulSoup(response.content, 'html.parser')

  # Extract the second li tag under the first ul tag
  agency = 'NA'
  # the Agency name locates next to the "Agency" span object
  span_tag = soup.find('span', text = 'Agency:')
  if span_tag:
    agency = span_tag.find_next_sibling(text=True).strip()
  logging.info(f'Finished scraping {link}')
  return agency


# Iterate through the DataFrame and scrape the agency data
df['Agency'] = df['Link'].apply(scrape_agency)

# Save the updated DataFrame to a CSV file
df.to_csv('scraped_data_2.csv', index=False)

# Print the updated DataFrame
print(df.head())
```

```
df = pd.read_csv('scraped_data_2.csv')
print(df.head())
```

```
                                               Title              Date  \
0  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
1  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
2  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
3  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024
4  Former Licensed Counselor Sentenced For Defrau...  November 6, 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/boise-nu...
1  https://oig.hhs.gov/fraud/enforcement/former-t...
2  https://oig.hhs.gov/fraud/enforcement/former-a...
3  https://oig.hhs.gov/fraud/enforcement/paroled-...
4  https://oig.hhs.gov/fraud/enforcement/former-l...

                                               Agency
0  November 7, 2024; U.S. Attorney's Office, Dist...
1  U.S. Attorney's Office, District of Massachusetts
2  U.S. Attorney's Office, Eastern District of Vi...
3  U.S. Attorney's Office, Middle District of Flo...
4  U.S. Attorney's Office, Western District of Texas
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

def scrape_from(month,year): try year > 2013: for i in range(480): scrape_data() ——scraping codes for each page—— data.extend() if date < datetime(month+year): print('finish') break except: return ('please enter a date after Jan 1 2013')

- b. Create Dynamic Scraper (PARTNER 2)

```
# Function to scrape data from a given month and year
from datetime import datetime

# Function to scrape data from a given month and year
def scrape_from(month, year):
    if year < 2013:
        print("Please restrict to year >= 2013, since only enforcement
        ↪  actions after 2013 are listed.")
        return
```

```
    data = []
    page_number = 1
    while True:
        scraped_data = scrape_page(page_number)
        if not scraped_data:
            break
        data.extend(scraped_data)

        # Check if the date in the scraped data is before the given month and
        ↪ year
        last_date_str = scraped_data[-1]['Date']
        try:
            last_date = datetime.strptime(last_date_str, '%B %d, %Y')
        except ValueError:
            continue
        if last_date < datetime(year, month, 1):
            break
        page_number += 1

    df = pd.DataFrame(data)
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
    filtered_df = df[df['Date'] >= datetime(year, month, 1)]

    # Scrape agency
    filtered_df['Agency'] = filtered_df['Link'].apply(scrape_agency)

    # Save the DataFrame to a CSV file
    file_name = f'enforcement_actions_{year}_{month}.csv'
    filtered_df.to_csv(file_name, index=False)

    return filtered_df

# Example usage
data_2023 = scrape_from(1, 2023)
print(len(data_2023))
print(data_2023.tail(1))
```

In this dataframe, 1534 enforcement were collected. The earliest enforcement acquired was Podiatrist Pays $90,000 To Settle False Billing Allegations on Jan 3, 2023.

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

```
data_2021 = scrape_from(1, 2021)
print(len(data_2021))
print(data_2021.tail(1))
```

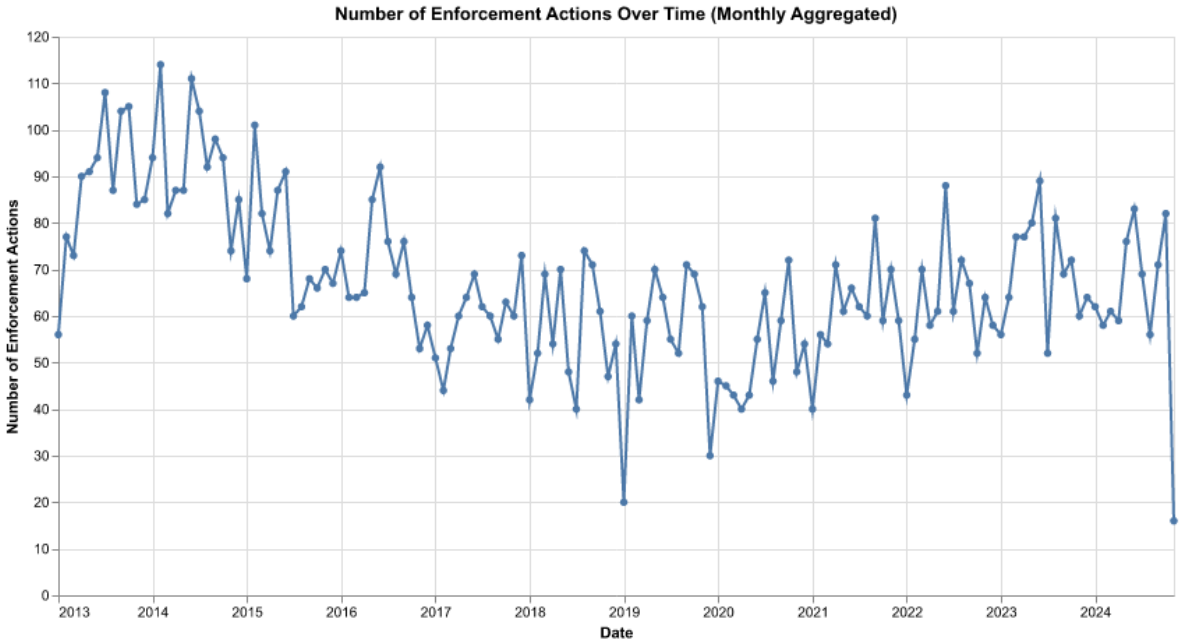
In this dataframe, 3022 enforcement were collected. The earliest enforcement acquired was The United States And Tennessee Resolve Claims With Three Providers For False Claims Act Liability Relating To 'P-Stim' Devices For A Total Of $1.72 Million on Jan 4, 2021.

Step 3: Plot data based on scraped data

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

Plot a line chart that shows: the number of enforcement actions over time (aggregated to each month+year) overall since January 2021

```
# Load the data to examine the structure
file_path =
 ↪  '/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/scraped_data.csv'
scraped_data = pd.read_csv(file_path)

# Convert the 'Date' column to datetime format
scraped_data['Date'] = pd.to_datetime(scraped_data['Date'], errors='coerce')

# Remove any rows where the date conversion failed (if any)
scraped_data = scraped_data.dropna(subset=['Date'])

# Extract year and month, then count enforcement actions per month
scraped_data['YearMonth'] = scraped_data['Date'].dt.to_period('M')
monthly_enforcement_counts =
 ↪  scraped_data.groupby('YearMonth').size().reset_index(name='EnforcementCount')

# Convert 'YearMonth' back to datetime format for plotting
monthly_enforcement_counts['YearMonth'] =
 ↪  monthly_enforcement_counts['YearMonth'].dt.to_timestamp()

# Plotting the line chart
chart = alt.Chart(monthly_enforcement_counts).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Date'),
    y=alt.Y('EnforcementCount', title='Number of Enforcement Actions')
).properties(
    title='Number of Enforcement Actions Over Time (Monthly Aggregated)',
    width=800,
    height=400
)

chart
```

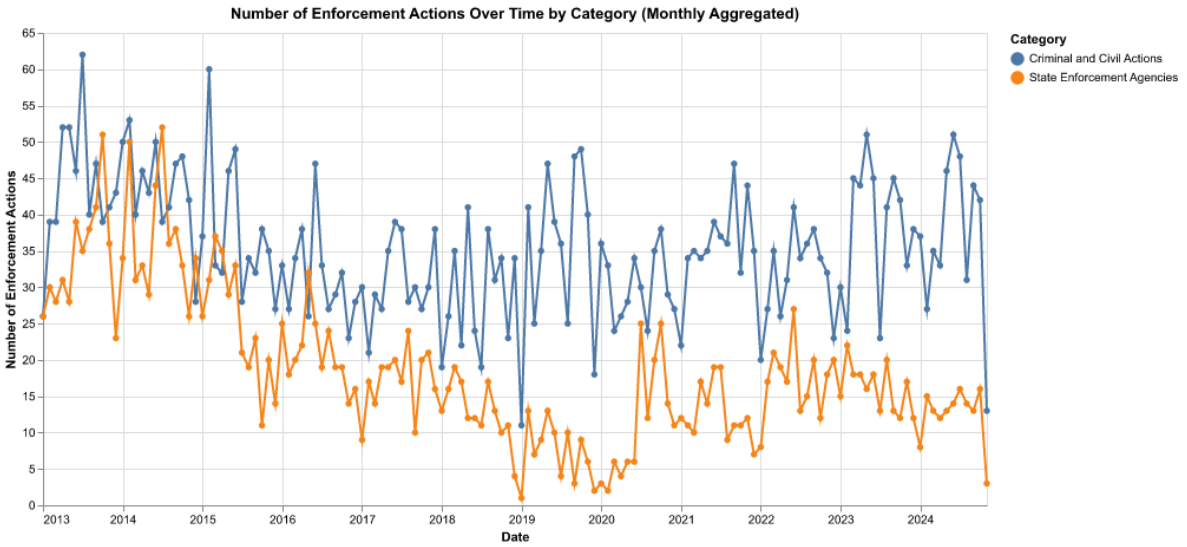


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

```
# Group by YearMonth and Category to count enforcement actions for each type
category_monthly_counts = scraped_data.groupby(['YearMonth',
 ↪  'Category']).size().reset_index(name='EnforcementCount')

# Convert 'YearMonth' back to datetime format for plotting
category_monthly_counts['YearMonth'] =
 ↪  category_monthly_counts['YearMonth'].dt.to_timestamp()

# Filter for relevant categories
filtered_categories =
 ↪  category_monthly_counts[category_monthly_counts['Category'].isin(['Criminal
 ↪  and Civil Actions', 'State Enforcement Agencies'])]

# Create Altair line chart
chart = alt.Chart(filtered_categories).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Date'),
    y=alt.Y('EnforcementCount', title='Number of Enforcement Actions'),
    color=alt.Color('Category', title='Category')
).properties(
```

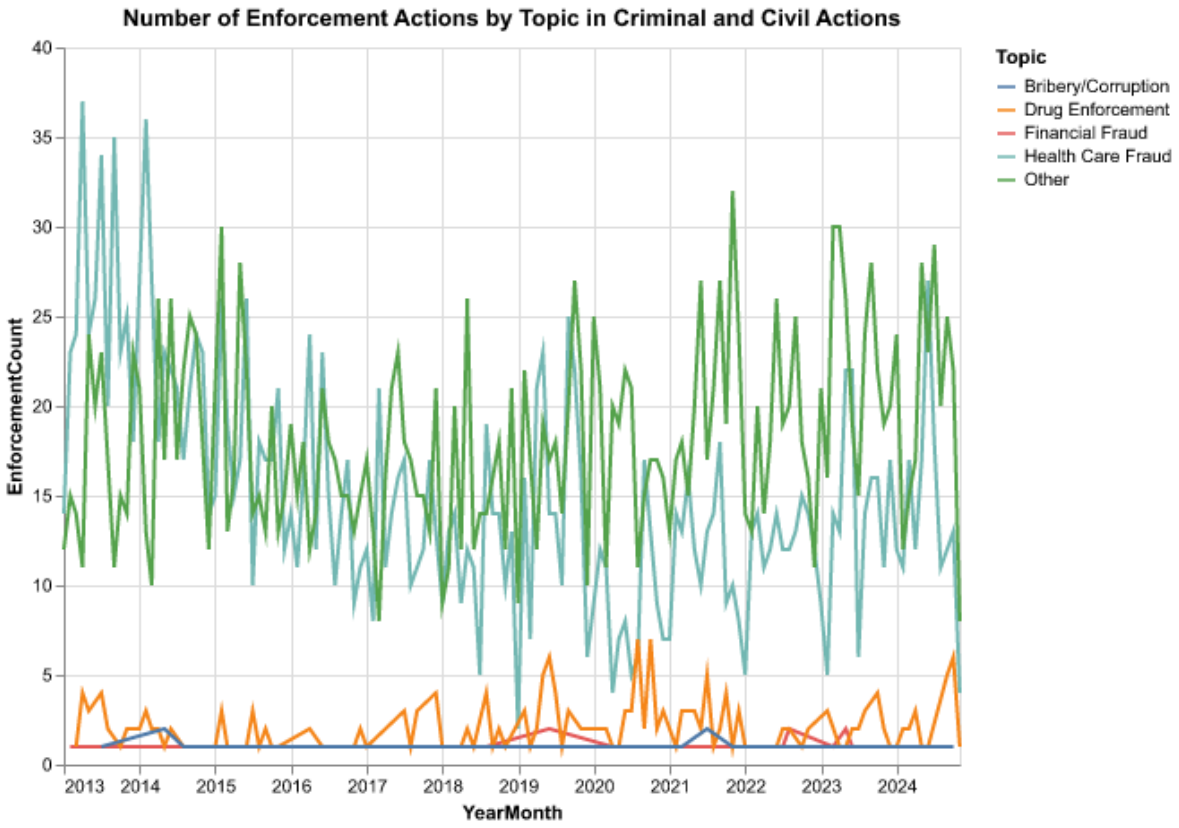
```
    title='Number of Enforcement Actions Over Time by Category (Monthly
 ↪  Aggregated)',
    width=800,
    height=400
)

chart
```



- based on five topics

```
criminal_civil_data = scraped_data[scraped_data['Category'] == 'Criminal and
 ↪  Civil Actions']

# Function to classify topics based on title
def classify_topic(title):
    title_lower = title.lower()
    if 'health care' in title_lower or 'medicare' in title_lower or
     ↪  'insurance' in title_lower:
        return 'Health Care Fraud'
    elif 'bank' in title_lower or 'financial' in title_lower or 'stock' in
     ↪  title_lower:
        return 'Financial Fraud'
    elif 'drug' in title_lower:
        return 'Drug Enforcement'
    elif 'bribery' in title_lower or 'corruption' in title_lower:
```

```
        return 'Bribery/Corruption'
    else:
        return 'Other'

# Apply the classification function
criminal_civil_data['Topic'] =
 ↪  criminal_civil_data['Title'].apply(classify_topic)

# Filter for the specified topics
topics = ["Health Care Fraud", "Financial Fraud", "Drug Enforcement",
 ↪  "Bribery/Corruption", "Other"]
filtered_criminal_civil_data =
 ↪  criminal_civil_data[criminal_civil_data['Topic'].isin(topics)]

# Group by YearMonth and Topic to count enforcement actions for each type
count_filtered_criminal_civil_data =
 ↪  filtered_criminal_civil_data.groupby(['YearMonth',
 ↪  'Topic']).size().reset_index(name='EnforcementCount')

# Convert YearMonth to string format
count_filtered_criminal_civil_data['YearMonth'] =
 ↪  count_filtered_criminal_civil_data['YearMonth'].dt.strftime('%Y-%m')

# Create the Altair chart
chart = alt.Chart(count_filtered_criminal_civil_data).mark_line().encode(
    x='YearMonth:T',
    y='EnforcementCount:Q',
    color='Topic:N'
).properties(
    title='Number of Enforcement Actions by Topic in Criminal and Civil
 ↪  Actions',
    width=500,
    height=400
)

chart
```



Step 4: Create maps of enforcement activity

For these questions, use this US Attorney District shapefile (link) and a Census state shapefile (link)

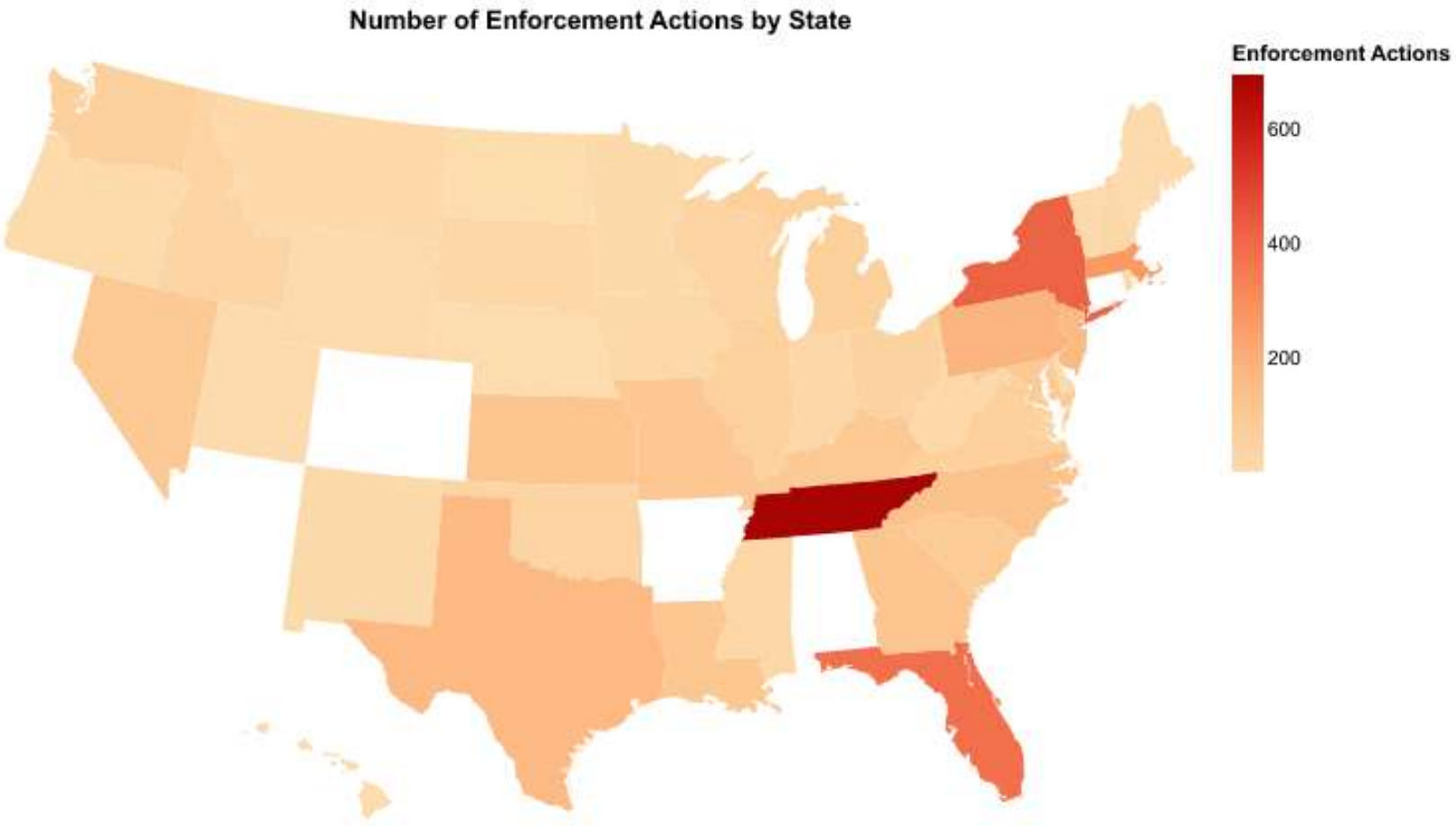
1. Map by State (PARTNER 1)

(Partner 1) Map by state: Among actions taken by state-level agencies, clean the state names you collected and plot a choropleth of the number of enforcement actions for each state. Hint: look for "State of" in the agency info!

```
# Load your data
crawled_data =
 ↪  pd.read_csv('/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/scraped_data_2.csv

# Extract state names based on the pattern "State of [State Name]"
#crawled_data['State'] = crawled_data['Agency'].str.extract(r'State of
 ↪  (\w+)')
```

```
# Extract state names based on the pattern "State of [State Name]"
#crawled_data['State'] = crawled_data['Agency'].str.extract(r'State of
 ↪  (\w+)')
state_mapping_complete = {
    'Alabama': 'Alabama', 'Alaska': 'Alaska', 'Arizona': 'Arizona',
     ↪  'Arkansas': 'Arkansas',
    'California': 'California', 'Colorado': 'Colorado', 'Connecticut':
     ↪  'Connecticut',
    'Delaware': 'Delaware', 'Florida': 'Florida', 'Georgia': 'Georgia',
     ↪  'Hawaii': 'Hawaii',
    'Idaho': 'Idaho', 'Illinois': 'Illinois', 'Indiana': 'Indiana', 'Iowa':
     ↪  'Iowa',
    'Kansas': 'Kansas', 'Kentucky': 'Kentucky', 'Louisiana': 'Louisiana',
     ↪  'Maine': 'Maine',
    'Maryland': 'Maryland', 'Massachusetts': 'Massachusetts', 'Michigan':
     ↪  'Michigan',
    'Minnesota': 'Minnesota', 'Mississippi': 'Mississippi', 'Missouri':
     ↪  'Missouri',
    'Montana': 'Montana', 'Nebraska': 'Nebraska', 'Nevada': 'Nevada', 'New
     ↪  Hampshire': 'New Hampshire',
    'New Jersey': 'New Jersey', 'New Mexico': 'New Mexico', 'New York': 'New
     ↪  York',
    'North Carolina': 'North Carolina', 'North Dakota': 'North Dakota',
     ↪  'Ohio': 'Ohio',
    'Oklahoma': 'Oklahoma', 'Oregon': 'Oregon', 'Pennsylvania':
     ↪  'Pennsylvania', 'Rhode Island': 'Rhode Island',
    'South Carolina': 'South Carolina', 'South Dakota': 'South Dakota',
     ↪  'Tennessee': 'Tennessee',
    'Texas': 'Texas', 'Utah': 'Utah', 'Vermont': 'Vermont', 'Virginia':
     ↪  'Virginia',
    'Washington': 'Washington', 'West Virginia': 'West Virginia',
     ↪  'Wisconsin': 'Wisconsin', 'Wyoming': 'Wyoming', 'Puerto Rico':
     ↪  'Puerto Rico', 'Virgin Islands': 'Virgin Islands'
}

# Update the 'State' column based on the presence of each state name in the
 ↪  'Agency' column
for keyword, state_name in state_mapping_complete.items():
    crawled_data.loc[crawled_data['Agency'].str.contains(keyword, case=False,
 ↪  na=False), 'State'] = state_name
```

```
# Drop rows with missing state information after extraction
state_counts = crawled_data['State'].dropna().value_counts().reset_index()
state_counts.columns = ['state_name', 'enforcement_count']

# Add FIPS codes for each state
state_counts['fips'] = state_counts['state_name'].apply(lambda x:
 ↪  us.states.lookup(x).fips.zfill(2) if us.states.lookup(x) else None)
state_counts['fips'] = state_counts['fips'].astype(str)

# Filter out rows without FIPS codes (if any)
state_counts = state_counts.dropna(subset=['fips'])

# Load US states topology in Altair (preloaded in Altair as a GeoJSON URL)
us_states_url = 'https://vega.github.io/vega-datasets/data/us-10m.json'

# Create Altair chart using the albersUsa projection
states_chart = alt.Chart(alt.topo_feature(us_states_url,
 ↪  'states')).mark_geoshape().encode(
    color=alt.Color('enforcement_count:Q',
 ↪  scale=alt.Scale(scheme='orangered'), title='Enforcement Actions'),
    tooltip=[alt.Tooltip('state_name:N', title='State'),
 ↪  alt.Tooltip('enforcement_count:Q', title='Count')]
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(state_counts, 'fips', ['enforcement_count'])
).project('albersUsa').properties(
    width=600,
    height=400,
    title='Number of Enforcement Actions by State'
)

states_chart.save('/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/state_chart.png

# alt_saver.save(states_chart,
 ↪  '/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/state_chart.png')

# ![Number of Enforcement Actions Over Time (Monthly
 ↪  Aggregated)]('/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/state_chart.png'
```

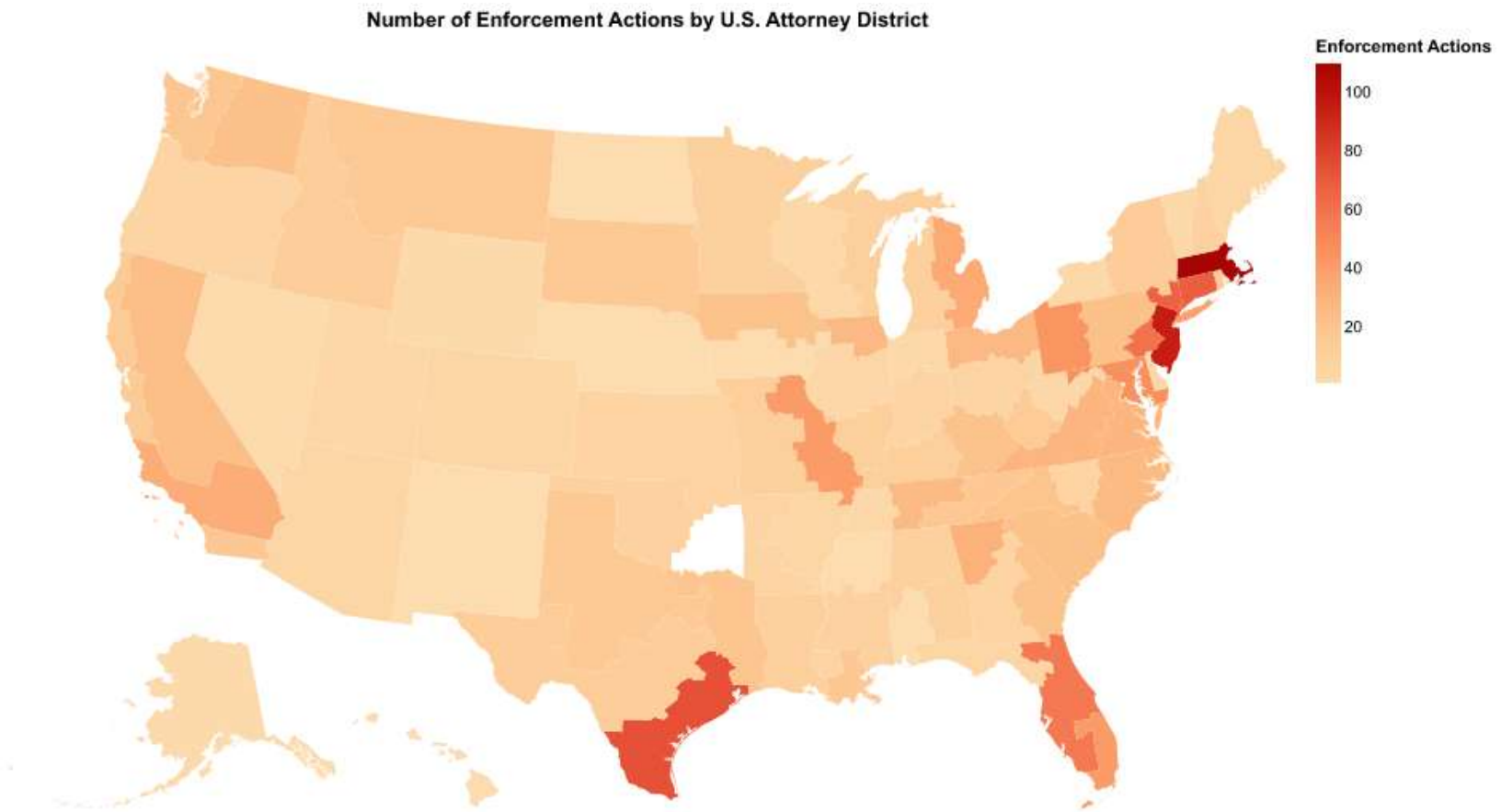

Number of Enforcement Actions by State
Enforcement Actions
600
400
200

2. Map by District (PARTNER 2)

Map by district: Among actions taken by US Attorney District-level agencies, clean the district names so that you can merge them with the shapefile, and then plot a choropleth of the number of enforcement actions in each US Attorney District. Hint: look for "District" in the agency info.

```
crawled_data =
 ↪  pd.read_csv('/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/scraped_data_2.csv

def format_district_name(agency_text):
    match = re.search(r'((\w+ )?District of (.+))', agency_text)
    if match:
        return match.group(1).strip()  # Capture the entire match (including
         ↪  "Eastern District of New York" etc.)
    return None

crawled_data['district'] = crawled_data['Agency'].apply(lambda x:
 ↪  format_district_name(x) if pd.notnull(x) else None)

# Count the number of enforcement actions by district
district_counts = crawled_data['district'].value_counts().reset_index()
district_counts.columns = ['district_name', 'enforcement_count']

# Load the district shapefile
district =
 ↪  gpd.read_file('/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/US
 ↪  Attorney Districts Shapefile
 ↪  simplified_20241108/geo_export_4d52aed3-d07e-4fbd-b5ca-8d2f7d5527bd.shp')

merged_gdf = district.merge(district_counts, left_on='judicial_d',
 ↪  right_on='district_name', how='left')
merged_gdf = merged_gdf.to_crs("EPSG:4326")
geojson_data = json.loads(merged_gdf.to_json())

# Create Altair chart with `albersUsa` projection
alt_chart =
 ↪  alt.Chart(alt.Data(values=geojson_data['features'])).mark_geoshape().encode(
    color=alt.Color('properties.enforcement_count:Q',
 ↪  scale=alt.Scale(scheme='orangered'), title='Enforcement Actions'),
    tooltip=[alt.Tooltip('properties.district_name:N', title='District'),
 ↪  alt.Tooltip('properties.enforcement_count:Q', title='Actions')]
```

```
).project(
    'albersUsa'
).properties(
    width=800,
    height=500,
    title='Number of Enforcement Actions by U.S. Attorney District'
)

alt_chart.save('/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/district_chart.png

# alt_saver.save(alt_chart,
 ↪  '/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/district_chart.png')


#![Number of Enforcement Actions by U.S. Attorney
 ↪  District](/Users/zhaozijing/Documents/GitHub/problem-set-5-zac-tina/district_chart.png)
```


Number of Enforcement Actions by U.S. Attorney District
Enforcement Actions
100
80
60
40
20