Problem Set 5

Zheng Cui & Xiaotian Tang

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): Zheng Cui (zhengcui)
    - Partner 2 (name and cnet ID): Xiaotian Tang (tangxiaotian)
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: **ZC** **XT**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **NaN**" (1 point)
6. Late coins used this pset: **\0** Late coins left after submission: **ZC:2, XT:3**
7. Knit your `ps5.qmd` to an PDF file to make `ps5.pdf`,
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
8. (Partner 1): push `ps5.qmd` and `ps5.pdf` to your github repo.
9. (Partner 1): submit `ps5.pdf` via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope

```
import pandas as pd
import altair as alt
import time
from datetime import datetime
import requests
from bs4 import BeautifulSoup
import geopandas as gpd
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

```
/Users/tang/Desktop/1Python
II/ProblemSet5/ve5/.venv/lib/python3.9/site-packages/urllib3/__init__.py:35:
NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the
'ssl' module is compiled with 'LibreSSL 2.8.3'. See:
https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
```

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
url = 'https://oig.hhs.gov/fraud/enforcement/'
# get the content using request
response = requests.get(url)
# parse the contents using Beautiful Soup.
soup = BeautifulSoup(response.text, "html.parser")
# the info that we want has a special style.
source = soup.find("ul", class_ = "usa-card-group padding-y-0" )
rows = source.find_all('div', class_='usa-card__container')
table_list = []
for row in rows:
  link = row.find("h2").a["href"]
  title = row.find("h2").a.get_text(strip=True)
  date = row.find("div").span.get_text(strip=True)
  category = row.find("div").ul.get_text(strip=True)
```

```
  row_list = [title,date,category,link]
  table_list.append(row_list)

tidy_df = pd.DataFrame(table_list, columns=["Title", "Date", "Category",
↪  "Link"])

print(tidy_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  /fraud/enforcement/pharmacist-and-brother-conv...
1  /fraud/enforcement/boise-nurse-practitioner-se...
2  /fraud/enforcement/former-traveling-nurse-plea...
3  /fraud/enforcement/former-arlington-resident-s...
4  /fraud/enforcement/paroled-felon-sentenced-to-...
```

2. Crawling (PARTNER 1)

```
agency_list =[]
for url in tidy_df["Link"]:
  response = requests.get("https://oig.hhs.gov"+ url)
  soup = BeautifulSoup(response.text, "html.parser")
  the_span_tag = soup.find("span", text=lambda x: x and x.strip() ==
↪  "Agency:")
  if the_span_tag:
    agency = the_span_tag.find_parent("li").get_text().replace("Agency:",
↪  "").strip()
```

```
  else:
    agency = ""
  agency_list.append(agency)

tidy_df["Agency"] = agency_list

print(tidy_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  /fraud/enforcement/pharmacist-and-brother-conv...
1  /fraud/enforcement/boise-nurse-practitioner-se...
2  /fraud/enforcement/former-traveling-nurse-plea...
3  /fraud/enforcement/former-arlington-resident-s...
4  /fraud/enforcement/paroled-felon-sentenced-to-...

                                          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)

```
def function(month, year):
# first check the year
  if year < 2013:
    print("Please restrict to year >= 2013")
    return
  else:
    continue

  while True:
    if page == 1:
      set url = "https://oig.hhs.gov/fraud/enforcement/"
    else:
      set url = "https://oig.hhs.gov/fraud/enforcement/?page=" + page

    scraping like what we do in step 1,
      if Date < input_time:
        break out of the while loop
      else:
        continue
  end while

  crawling like what we do in step 1 to obtain agency

  return tidy_df
```

- b. Create Dynamic Scraper (PARTNER 2)

```
def fancy_scraper(start_year, start_month):
  if start_year < 2013:
    print("Please note that only enforcement actions from 2013 onwards are
     ↪  available.")
    return

  stop_scraping = False
  table_list = []
  page = 1
  while stop_scraping == False:
# url change with page
    url = 'https://oig.hhs.gov/fraud/enforcement/' if page == 1 else
 ↪  f'https://oig.hhs.gov/fraud/enforcement/?page={page}'
```

```
# repeat what we do in step 1
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    source = soup.find("ul", class_ = "usa-card-group padding-y-0" )
    rows = source.find_all('div', class_='usa-card__container')

    for row in rows:
      date_text = row.find("div").span.get_text(strip=True)
      date = datetime.strptime(date_text, "%B %d, %Y") # format into datetime
 ↪  object
      # create a stop logic
      if date.year < start_year or (date.year == start_year and date.month <
       ↪  start_month):
        stop_scraping = True
        break
      link = row.find("h2").a["href"]
      title = row.find("h2").a.get_text(strip=True)
      category = row.find("div").ul.get_text(strip=True)
      row_list = [title,date,category,link]
      table_list.append(row_list)

    page += 1
    time.sleep(1)

  tidy_df = pd.DataFrame(table_list, columns=["Title", "Date", "Category",
 ↪  "Link"])

# # add up agency column
  agency_list =[]

  for url in tidy_df["Link"]:
    response = requests.get("https://oig.hhs.gov"+ url)
    soup = BeautifulSoup(response.text, "html.parser")
    the_span_tag = soup.find("span", text=lambda x: x and x.strip() ==
 ↪  "Agency:")
    if the_span_tag:
      agency = the_span_tag.find_parent("li").get_text().replace("Agency:",
 ↪  "").strip()
    else:
      agency = ""
    agency_list.append(agency)
```

```
  tidy_df["Agency"] = agency_list

#output to a csv file
  output_filename = f"enforcement_actions_{start_year}_{start_month}.csv"
  tidy_df.to_csv(output_filename, index=False, encoding='utf-8')

  return tidy_df

df_2301 = fancy_scraper(2023,1)


# obtain the number of enforcement actions
num_2301 = df_2301.shape[0]
print(f"There are {num_2301} enforcement actions since January 2023.")

# obtain the detail of the earliest enforcement
print(df_2301.tail(1))
```

```
There are 1534 enforcement actions since January 2023.
                                                  Title       Date  \
1533  Podiatrist Pays $90,000 To Settle False Billin... 2023-01-03

                        Category  \
1533  Criminal and Civil Actions

                                                   Link  \
1533  /fraud/enforcement/podiatrist-pays-90000-to-se...

                                                 Agency
1533  U.S. Attorney's Office, Southern District of T...
```

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

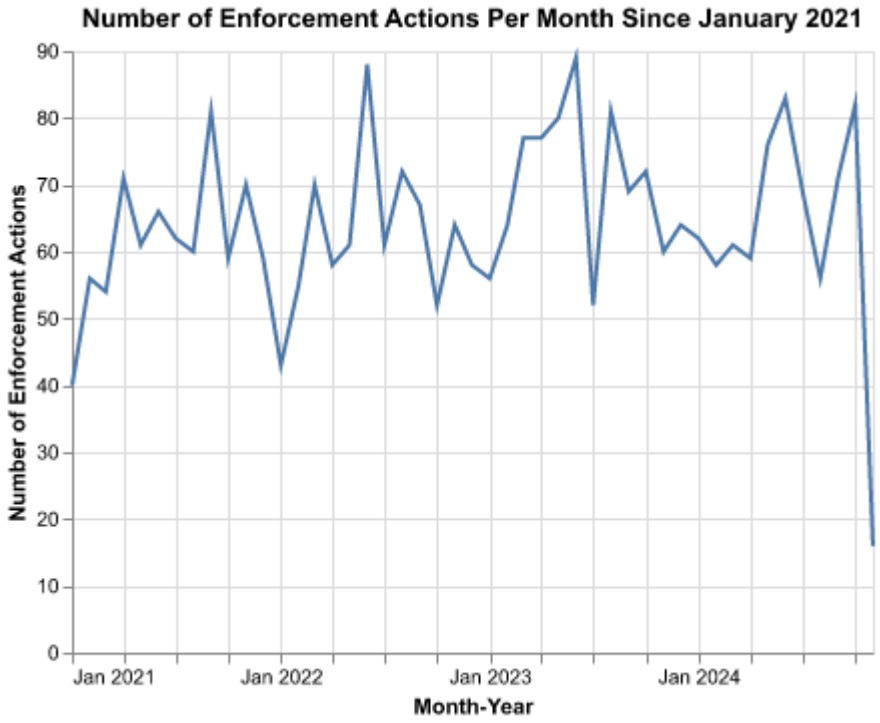
```
df = fancy_scraper(2021,1)

# obtain the number of enforcement actions
num = df.shape[0]
print(f"There are {num} enforcement actions since January 2021.")

# obtain the detail of the earliest enforcement
print(df.tail(1))
```

```
There are 3022 enforcement actions since January 2021.
                                                    Title       Date  \
3021  The United States And Tennessee Resolve Claims... 2021-01-04

                         Category  \
3021  Criminal and Civil Actions

                                                     Link  \
3021  /fraud/enforcement/the-united-states-and-tenne...

                                                   Agency
3021  U.S. Attorney's Office, Middle District of Ten...
```

Step 3: Plot data based on scraped data

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

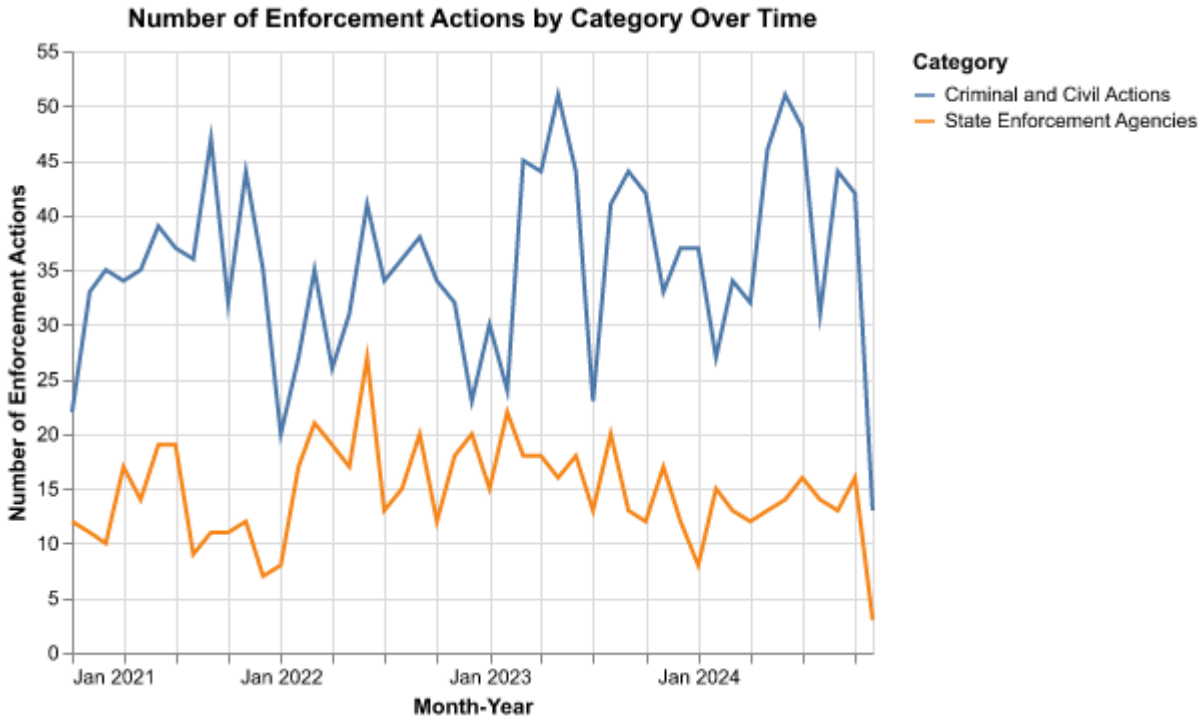
```
chart = alt.Chart(df).mark_line().encode(
    x=alt.X('yearmonth(Date):T', title='Month-Year'),
    y=alt.Y('count():Q', title='Number of Enforcement Actions')
).properties(width=400, height=300,
    title="Number of Enforcement Actions Per Month Since January 2021"
)

chart.show()
```



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

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

```
chart = alt.Chart(df).mark_line().transform_filter(
    alt.FieldOneOfPredicate(field='Category', oneOf=['Criminal and Civil
 ↪  Actions', 'State Enforcement Agencies'])
).encode(
    x=alt.X('yearmonth(Date):T', title='Month-Year'),
    y=alt.Y('count():Q', title='Number of Enforcement Actions'),
    color=alt.Color('Category:N', title='Category')
).properties(width=400, height=300,
    title="Number of Enforcement Actions by Category Over Time"
)

chart
```

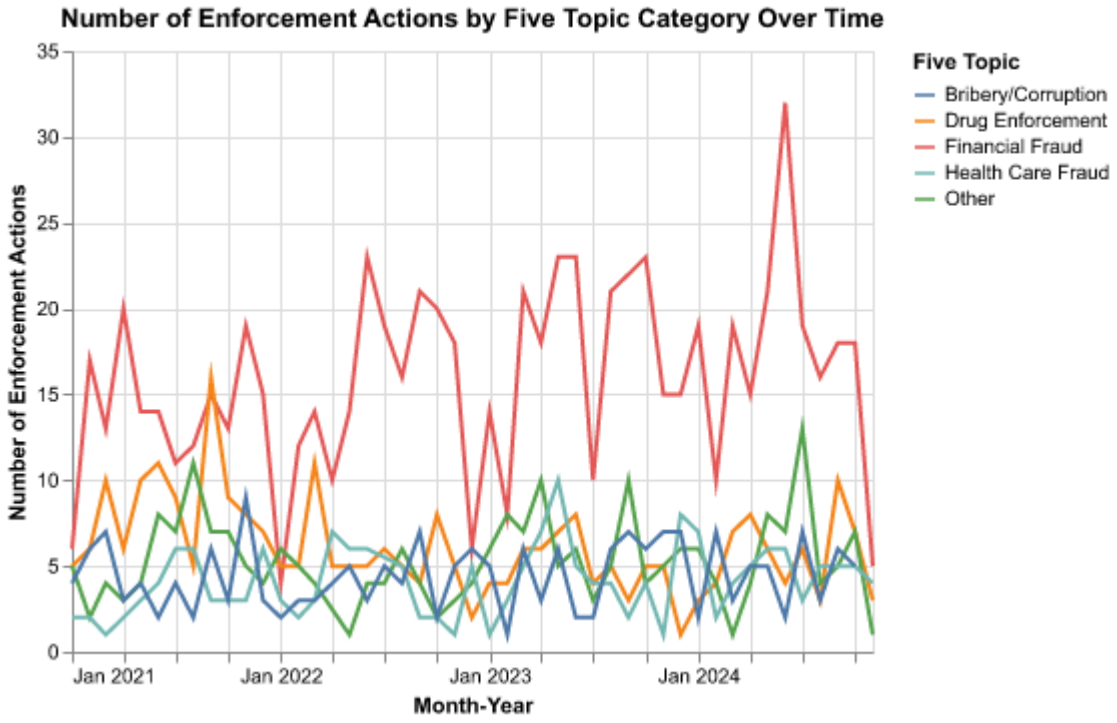


- based on five topics

```
# filter for "Criminal and Civil Actions"
df_cca = df[df["Category"]=="Criminal and Civil Actions"]

# Define the categorization function based on the Title column
def categorize_action(title):
    title_lower = title.lower()
    if any(word in title_lower for word in ['drug', 'narcotic', 'opioid',
     ↪  'substance', 'distribution', 'prescription']):
        return 'Drug Enforcement'
    elif any(word in title_lower for word in ['bribery', 'corruption',
     ↪  'bribe', 'kickback', 'graft']):
        return 'Bribery/Corruption'
    elif any(word in title_lower for word in ['financial', 'bank', 'money',
     ↪  'investment', 'fund', 'fraud', 'billing', 'scheme']):
        return 'Financial Fraud'
    elif any(word in title_lower for word in ['health', 'care', 'hospital',
     ↪  'medicare', 'medicaid', 'doctor', 'pharmacy']):
        return 'Health Care Fraud'
    else:
```

```
        return 'Other'

# Apply the categorization function to the 'Title' column in your DataFrame
df_cca['five_topic'] = df_cca['Title'].apply(categorize_action)
```

```
chart = alt.Chart(df_cca).mark_line().encode(
    x=alt.X('yearmonth(Date):T', title='Month-Year'),
    y=alt.Y('count():Q', title='Number of Enforcement Actions'),
    color=alt.Color('five_topic:N', title='Five Topic')
).properties( width=400, height=300,
    title='Number of Enforcement Actions by Five Topic Category Over Time'
)

chart
```

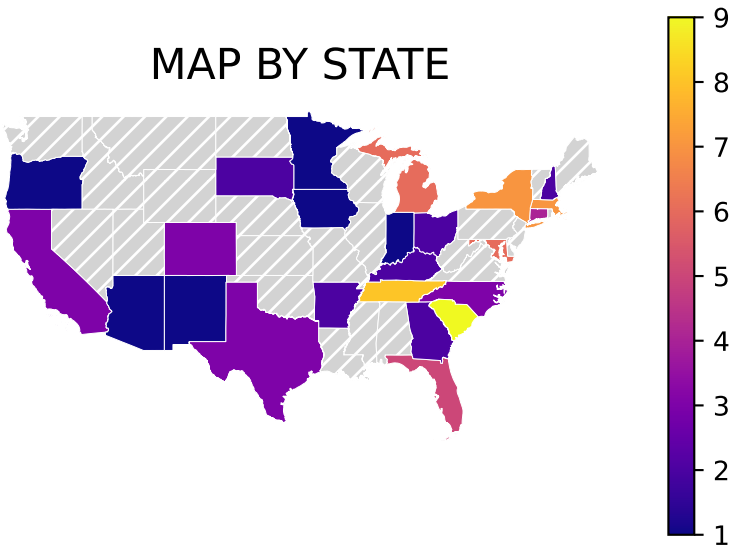


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
# select rows with "Agency" column contains "State of"
df_state = df[df['Agency'].str.contains('State of', na = False)]

# clean the state name
df_state['NAME'] = df_state['Agency'].str.replace('State of ','')

# aggregate
df_state_agg = df_state.groupby('NAME').size().reset_index(name = 'Count')

# import geodata
file_path = '/Users/tang/Desktop/1Python
 ↪  II/ProblemSet5/ve5/cb_2018_us_state_500k/cb_2018_us_state_500k.shp'

geodata = gpd.read_file(file_path)

# merge
merge_state = geodata.merge(df_state_agg, on = 'NAME', how = 'left')
```

```
# merge_state only contains state in the contiguougs United States
merge_state = merge_state.cx[-125:-66, 24:50]


merge_state.plot(column="Count",
                 edgecolor="white",
                 linewidth=0.2,
                 legend=True,
                 cmap="plasma",
                 missing_kwds={
                     "color": "lightgray",
                     "edgecolor": "white",
                     "hatch": "///"
                     })
plt.axis("off")
plt.title("MAP BY STATE", fontsize=16)
plt.show()
```



2. Map by District (PARTNER 2)

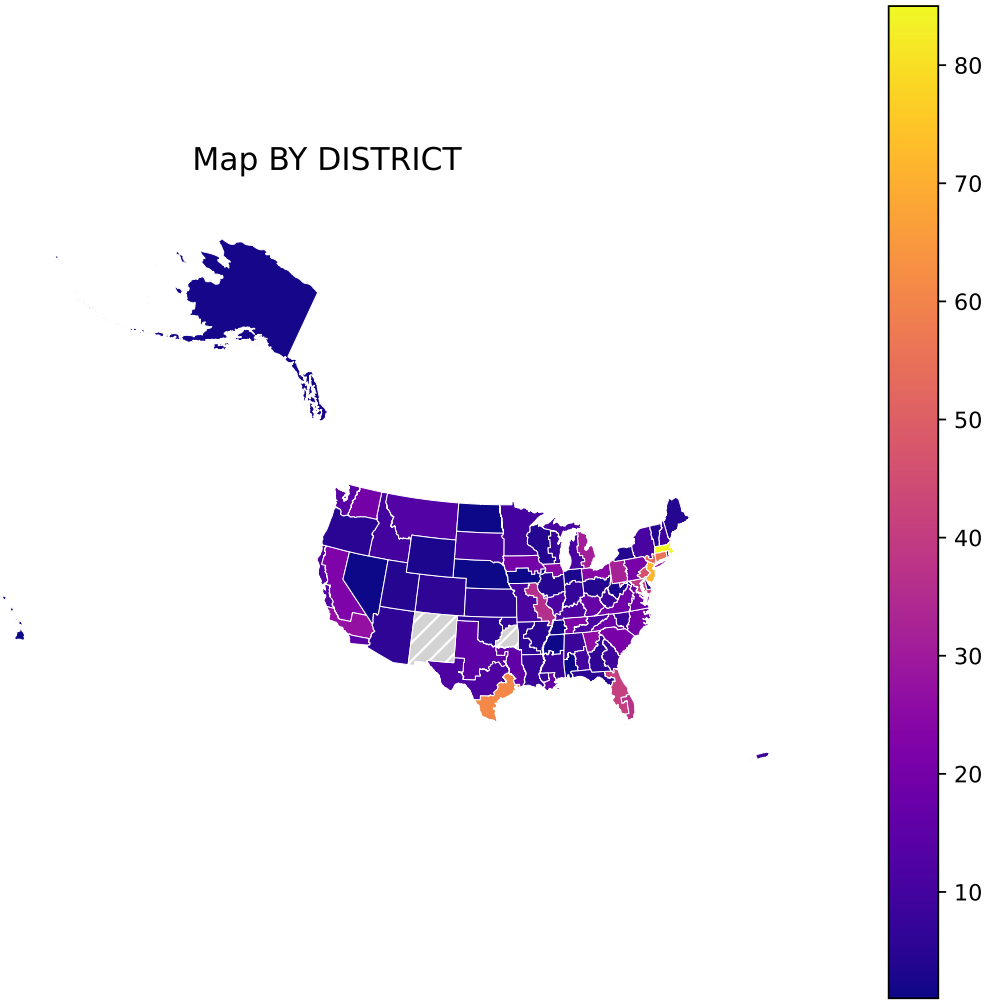
```
file_path_2 = '/Users/tang/Desktop/1Python II/ProblemSet5/ve5/US Attorney
 ↪  Districts Shapefile
 ↪  simplified_20241107/geo_export_406c52ee-4f0e-4671-9999-86ab3e0daa12.shp'

attorney = gpd.read_file(file_path_2)
# attorney.head()

# select rows with "Agency" column contains "U.S. Attorney's Office, "
df_dist = df[df['Agency'].str.contains("Attorney", na=False) &
 ↪  df['Agency'].str.contains("District", na=False)]

# clean the state name
df_dist['judicial_d'] = df_dist['Agency'].str.replace("U.S. Attorney's
 ↪  Office, ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("U.S. Attorney’s
 ↪  Office, ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("U.S. Attorney’s
 ↪  Office; ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("June 28, 2024:
 ↪  ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("†",'')
```

```
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("U.S. Department of
 ↪  Justice and ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("2021; ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("Connecticut
 ↪  Attorney General and ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("U.S. Attorney
 ↪  General, ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("U.S. AttorneyÍs
 ↪  Office, ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("U.S. Attorney’s
 ↪  Office ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("Attorney's Office,
 ↪  ",'')
df_dist['judicial_d'] = df_dist['judicial_d'].str.replace("Attorney’s Office,
 ↪  ",'')

# some judicial_d columns contain 2 unit, devide into two rows
df_dist['judicial_d'] = df_dist['judicial_d'].apply(lambda x: x.split(' and
 ↪  ') if ' and ' in x else [x])
df_dist = df_dist.explode('judicial_d')

# strip
df_dist['judicial_d'] = df_dist['judicial_d'].str.strip()

# deal with detail
df_dist.loc[df_dist['judicial_d'] == "Eastern District of Pennsylvani",
 ↪  'judicial_d'] = "Eastern District of Pennsylvania"

# aggregate
df_dist_agg = df_dist.groupby('judicial_d').size().reset_index(name =
 ↪  'Count')

# merge
merge_dist = attorney.merge(df_dist_agg, on = 'judicial_d', how = 'left')
merge_dist.head()
```

	statefp	judicial_d	aland	awater	state	chief_judg
0	21	Western District of Kentucky	4.970555e+10	1.651516e+09	Kentucky	Greg N. Stivers
1	21	Eastern District of Kentucky	5.257394e+10	7.238213e+08	Kentucky	Danny Reeves
2	18	Southern District of Indiana	5.824517e+10	5.941176e+08	Indiana	Jane Magnus-Stinson
3	01	Middle District of Alabama	3.412673e+10	5.472423e+08	Alabama	Emily Coody Marks

	statefp	judicial_d	aland	awater	state	chief_judg
4	01	Southern District of Alabama	6.235882e+10	3.052681e+09	Alabama	Kristi DuBose

```
# set central longitude and latitude for the Albers USA projection
albers_usa_projection = ccrs.AlbersEqualArea(central_longitude=-96,
 ↪  central_latitude=37)

# create canvas
fig, ax = plt.subplots(figsize=(10, 8), subplot_kw={'projection':
 ↪  albers_usa_projection})

# Convert the data to the Albers USA projection
map_dist_albers = merge_dist.to_crs(albers_usa_projection.proj4_init)

# plot
map_dist_albers.plot(
    column="Count",
    legend=True,
    ax=ax,
    edgecolor="white",
    linewidth=0.2,
    cmap="plasma",
    missing_kwds={"color": "lightgray", "edgecolor": "white", "hatch": "///"}
)

ax.set_extent([-180, -60, 15, 72], crs=ccrs.PlateCarree())
ax.axis("off")
ax.set_title("Map BY DISTRICT", fontsize=14)
plt.show()
```



Extra Credit

1. Merge zip code shapefile with population

```
# import data
popu = pd.read_csv('/Users/tang/Desktop/1Python
 ↪  II/ProblemSet5/ve5/DECENNIALDHC2020.P1_2024-11-08T205535/DECENNIALDHC2020.P1-Data.csv')

zipgeo = gpd.read_file('/Users/tang/Desktop/1Python
 ↪  II/ProblemSet5/ve5/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp')
zipgeo.head()
```

```
# clean popu
popudata = popu.iloc[1:,1:-1]
popudata = popudata.rename(columns={"P1_001N":"Population"})
popudata['NAME'] = popudata['NAME'].str.replace("ZCTA5 ",'')

# merge
merge_popu = zipgeo.merge(popudata, on = 'NAME', how = 'left')
merge_popu.head()
```

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry
0	8600000US01040	01040	01040	ZCTA5	21.281	POLYGON ((-72.62734 42.16203, -72.62
1	8600000US01050	01050	01050	ZCTA5	38.329	POLYGON ((-72.95393 42.34379, -72.95
2	8600000US01053	01053	01053	ZCTA5	5.131	POLYGON ((-72.68286 42.37002, -72.68
3	8600000US01056	01056	01056	ZCTA5	27.205	POLYGON ((-72.39529 42.18476, -72.39
4	8600000US01057	01057	01057	ZCTA5	44.907	MULTIPOLYGON (((-72.39191 42.0806

2. Conduct spatial join

```
dist_popu = gpd.sjoin(merge_popu, attorney, how="inner", predicate="within")
# dist_popu.head()

# change population to numeric format
dist_popu["Population"] = pd.to_numeric(dist_popu["Population"])

# aggregate
dist_popu_agg =
 ↪  dist_popu.groupby("judicial_d")["Population"].sum().reset_index()

# display
dist_popu_agg.head()
```

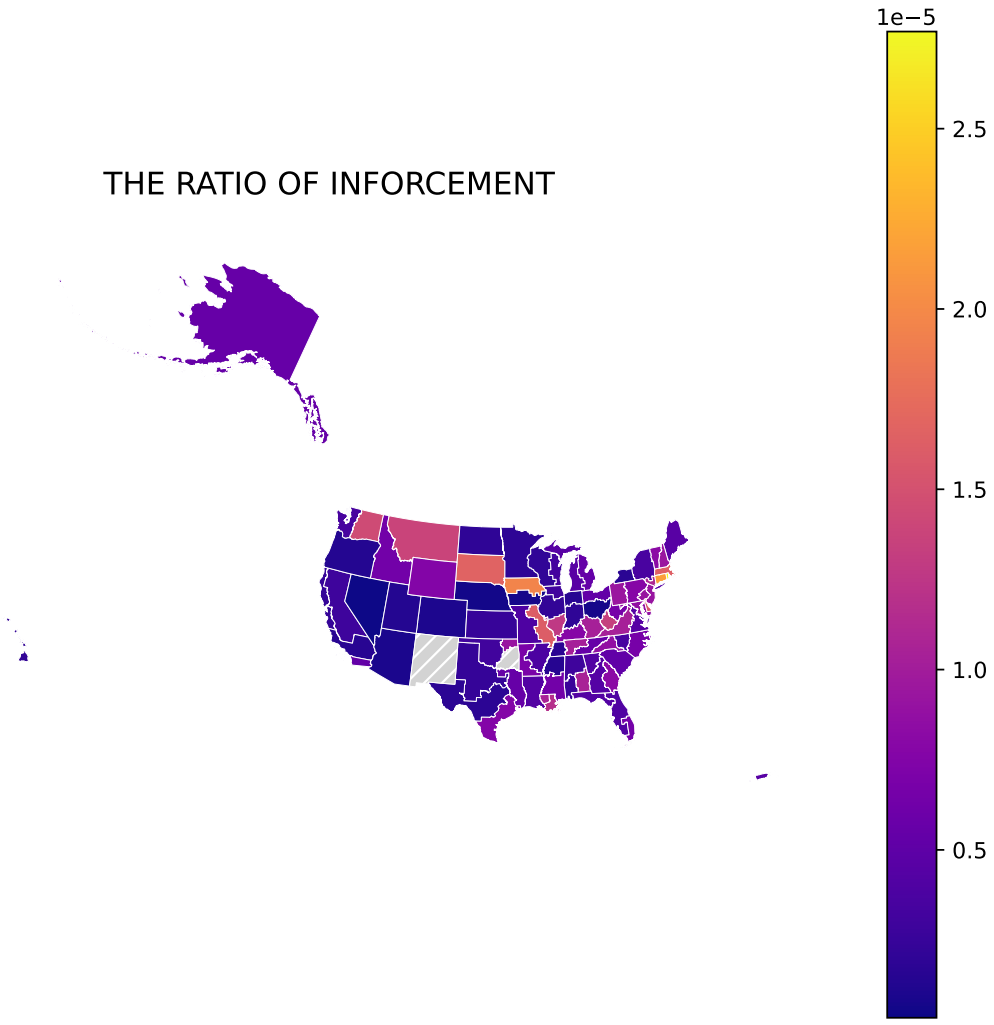
	judicial_d	Population
0	Central District of California	17541476.0
1	Central District of Illinois	1836364.0
2	District of Alaska	358718.0
3	District of Arizona	6648991.0

	judicial_d	Population
4	District of Colorado	5636000.0

3. Map the action ratio in each district

```
# merge
merge_ratio = dist_popu_agg.merge(df_dist_agg, on = 'judicial_d', how =
 ↪  'inner')

# calculate ratio
merge_ratio['ratio'] = merge_ratio['Count']/merge_ratio['Population']

# merge for mapping
map_ratio = attorney.merge(merge_ratio, on = 'judicial_d', how = 'left')
```

```
# create canvas
fig, ax = plt.subplots(figsize=(10, 8), subplot_kw={'projection':
 ↪  albers_usa_projection})

# Convert the data to the Albers USA projection
map_ratio_albers = map_ratio.to_crs(albers_usa_projection.proj4_init)

# plot
map_ratio_albers.plot(
    column="ratio",
    legend=True,
    ax=ax,
    edgecolor="white",
    linewidth=0.2,
    cmap="plasma",
    missing_kwds={"color": "lightgray", "edgecolor": "white", "hatch": "///"}
)

ax.set_extent([-180, -60, 15, 72], crs=ccrs.PlateCarree())
ax.axis("off")
ax.set_title("THE RATIO OF INFORCEMENT", fontsize=14)
plt.show()
```


THE RATIO OF INFORCEMENT
1e−5
2.5
2.0
1.5
1.0
0.5