\RecustomVerbatimEnvironment{verbatim}{Verbatim}{ showspaces = false, showtabs = false, breaksymbolleft={}, breaklines }

Problem Set 5

AUTHOR
Daiel & Nasser

PUBLISHED
November 9, 2024

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 (Daniel Avila, davila2020): • Partner 2 (Nasser Alshaya, alshaya):
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: **DA NA**
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 time
import requests
from bs4 import BeautifulSoup
import warnings
import geopandas as gpd
import matplotlib.pyplot as plt

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

url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
soup.text[0:50]
```

```
'\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\nEnforcement Actions | Office '
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
tag = soup.find_all("ul")
ul_children = soup.find_all(lambda t: t.name == 'ul' and t.find_all("div"))

# title of enforcement action
ul_children_titles = soup.find_all(lambda t: t.name == 'ul' and t.find_all("a"))

titles_list = []
for title in ul_children_titles:
    for a in title.find_all("a"):
        text = a.get_text()
        titles_list.append(text)

     ## selecting only the titles, since the previous version of titles_list has extra text
titles_list = titles_list[136:156]

# date
ul_children_date = soup.find_all(lambda t: t.name == 'div' and t.find_all("span"))

dates_list = []
for date in ul_children_date:
    for span in date.find_all("span"):
        date = span.get_text()
        dates_list.append(date)

dates_list = dates_list[33:53]


# category
ul_children_category = soup.find_all(lambda t: t.name == 'ul' and t.find_all("span") and t.find_a

category_list = []
for category in ul_children_category:
    for li in category.find_all("li"):
        category = li.get_text()
        category_list.append(category)

for i in range(len(category_list)):
    category_list[i] = category_list[i].replace("\n","")

category_list = category_list[74:135]

string_check = ["Criminal and Civil Actions", "State Enforcement Agencies", "CMP and Affirmative

category_list_new = []
for i in range(len(category_list)):
```

```
    for j in range(len(string_check)):
        if category_list[i] == string_check[j]:
            category_list_new.append(category_list[i])
        else:
            pass

#links
links_list = []
for item in ul_children:
    for a in item.find_all('a'):
        full_link = "https://oig.hhs.gov/" + a["href"]
        links_list.append(full_link)

df = pd.DataFrame({
    "ea_titles": titles_list,
    "dates": dates_list,
    "categories": category_list_new,
    "links": links_list})

df.head()
```

	ea_titles	dates	categories	links
0	Pharmacist and Brother Convicted of $15M Medic...	November 8, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/pharmac...
1	Boise Nurse Practitioner Sentenced To 48 Month...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/boise-n...
2	Former Traveling Nurse Pleads Guilty To Tamper...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/former-...
3	Former Arlington Resident Sentenced To Prison ...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/former-...
4	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/paroled...

2. Crawling (PARTNER 1)

```
df["links"] = df["links"].astype(str)
li_tags = []

# i want to iterate through the links columns
for row in range(len(df["links"])):
    url = df.loc[row, "links"]
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'lxml')

    # parse the html and find the agency
```

```
    li_tag = soup.find_all("li")
    agency_found = False
    for li in li_tag:
        li = li.get_text()

    #searching just for the agency
        if li.startswith("Agency"):
            li_tags.append(li)
            agency_found = True
            break

    if not agency_found:
        li_tags.append("NaN")

# cleaning the "agency:" out
for agency in range(len(li_tags
)):
    li_tags[agency] = li_tags[agency].replace("Agency:", "")

# append the list as a new column into the dataframe
df["agency"] = li_tags

df.head()
```

	ea_titles	dates	categories	links	agency
0	Pharmacist and Brother Convicted of $15M Medic...	November 8, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/pharmac...	U.S. Department of Justice
1	Boise Nurse Practitioner Sentenced To 48 Month...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/boise-n...	November 7, 2024; U.S. Attorney's Office, Dist...
2	Former Traveling Nurse Pleads Guilty To Tamper...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/former-...	U.S. Attorney's Office, District of Massachusetts
3	Former Arlington Resident Sentenced To Prison ...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/former-...	U.S. Attorney's Office, Eastern District of Vi...
4	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024	Criminal and Civil Actions	https://oig.hhs.gov//fraud/enforcement/paroled...	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)

Step 0: visually inspect the page to determine what need scrapping. Step 1: define constant variables needed: source_link, start_date, Step 2: set condition to assure input year larger than 2013 Step 3: using a for loop, iterate over the range of pages based on f-string for the link of next page. Step 4: the loop will scrape every page and extract the data needed to build the dataset basec an their tags and classes and add them to an empty dictionary which will the results of ecah iteration will be appended to an empty list to collect all data in one object; an if condition to break the loop based on strating date is included as well. Step 5: an inner for loop will visit all extracted links and scrape the page of each enforcement to extract the agency name, an if condition will check whether an agency exsists or not and store the text value accoringly. Step 6: the function stores the resulted dataset and then converts it to a csv file.

- b. Create Dynamic Scraper (PARTNER 2)

I ended up with `1553` enforcement actions and the details of the earliest one are as the following: Title: Podiatrist Pays $90,000 To Settle False Billin.. Date: January 3, 2023 Category: Criminal and Civil Actions Agency: U.S. Attorney's Office, Southern District of Texas

```
import datetime
def scrape_hhs_oig(year, month):
    """
    Scrapes the HHS OIG's Enforcement Actions page for a given year and month.

    Args:
        year (int): The year to filter by.
        month (int): The month to filter by.

    Returns:
        list: A dataframe containing the scraped data.
    """
    source_link = "https://oig.hhs.gov"
    start_date = datetime.datetime(year, month, 1)
    if year < 2013:
        print("Please input a year greater than or equal to 2013.")
        return
    enforcements =[]
    # Crawling:
    for i in range(480):
        link = f'https://oig.hhs.gov/fraud/enforcement/?page={i}'
        response = requests.get(link)
        soup = BeautifulSoup(response.text, 'lxml')

        for li in soup.find_all('li', class_='usa-card'):
            item = {}
        # Extract title and link
            h2 = li.find('h2', class_='usa-card__heading')
            a_tag = h2.find('a', href=True)
```

```
            item['title'] = a_tag.get_text(strip=True)
            item['link'] = source_link + a_tag['href']
            # Extract date and category
            date_span = li.find('span', class_='text-base-dark')
            action_date_str = date_span.get_text(strip=True)
            action_date = datetime.datetime.strptime(action_date_str, '%B %d, %Y')
            if action_date < start_date:
                df_enforcements = pd.DataFrame(enforcements)
                df_enforcements.to_csv('enforcement_actions_year_month.csv',index = False)
                return df_enforcements
            else:
                item['date'] = action_date_str
            # Extract categories:
            category_li = li.find('ul',
             class_ = 'display-inline add-list-reset')
            item['category'] = category_li.get_text(strip=True)
            enforcements.append(item)
            # Extract Agency:
            agency_response = requests.get(item['link'])
            agency_soup = BeautifulSoup(agency_response.text, 'lxml')

            for li in agency_soup.find_all('article', class_='grid-col desktop:grid-col-10 margin
                agency_li = li.find_all('li')
                agency_name = next(
                    (item.text.replace('Agency:', '').strip() for
                     item in agency_li if item.find('span') and 'Agency:'
                     in item.find('span').text),None)
                if agency_name:
                    item["Agency"] = agency_name
                    break
                else:
                    item["Agency"] = None
                    break
        # Add a 1-second delay between requests
        time.sleep(1)
    # Creating dataframe of enforcements:
    df_enforcements = pd.DataFrame(enforcements)
    df_enforcements.to_csv('enforcement_actions_year_month.csv',
    index = False)
    return df_enforcements
```

```
# Details of earliest enforcement action scrapped,
# commented out to speed knitting:

# scrape_hhs_oig(2024, 10).tail(1)
```

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

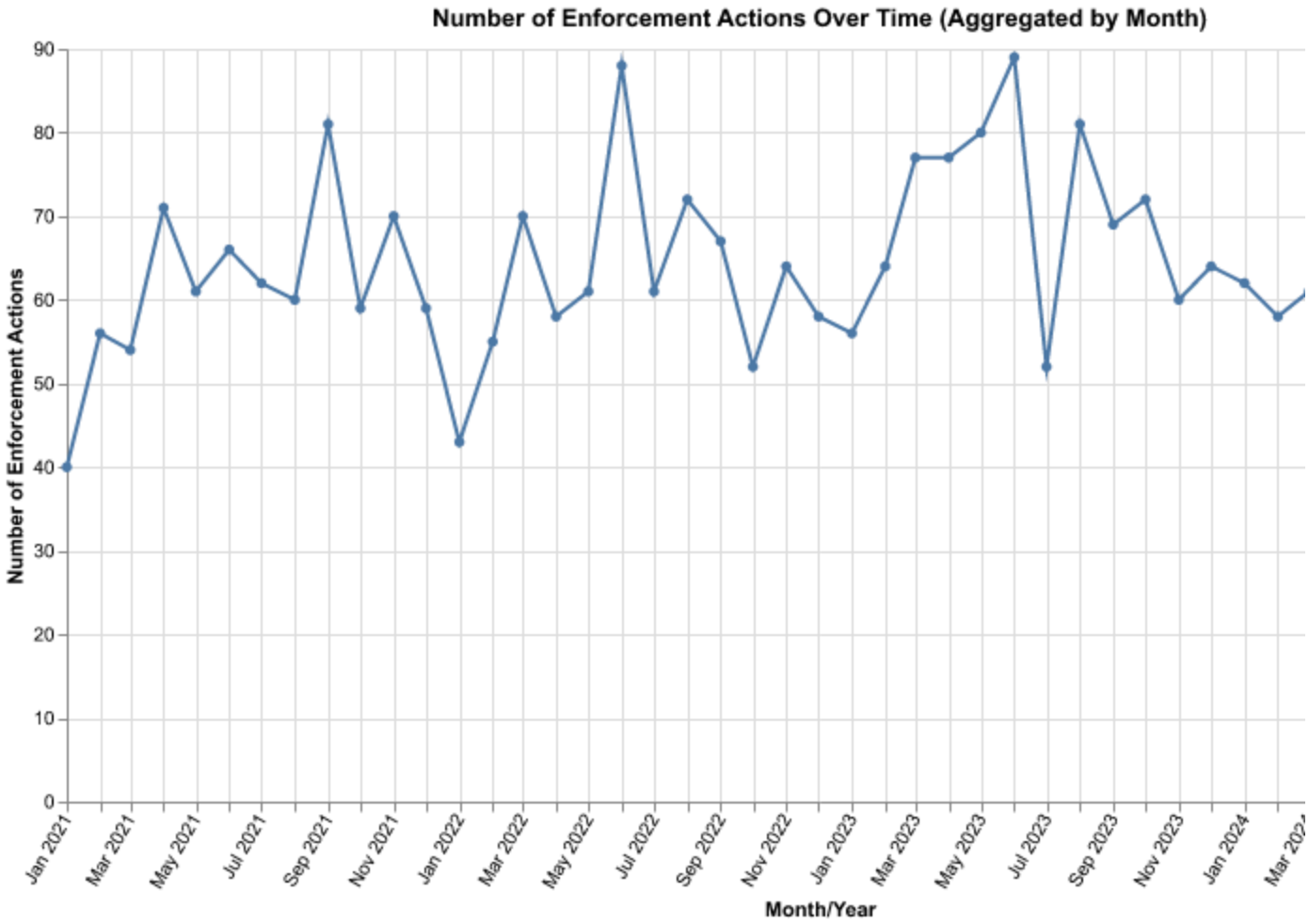
```
# Commenetd out to speed knitting:
# df_2021 = scrape_hhs_oig(2021,1)
# df_2021.tail(1)
```

Step 3: Plot data based on scraped data

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

Upon visual inspection of the plot, the number of enforcemnt actions fluctuates between 40 and 90, with a minimum count of 30 in November of this year after a rapid decrease in comparison to the previous month.

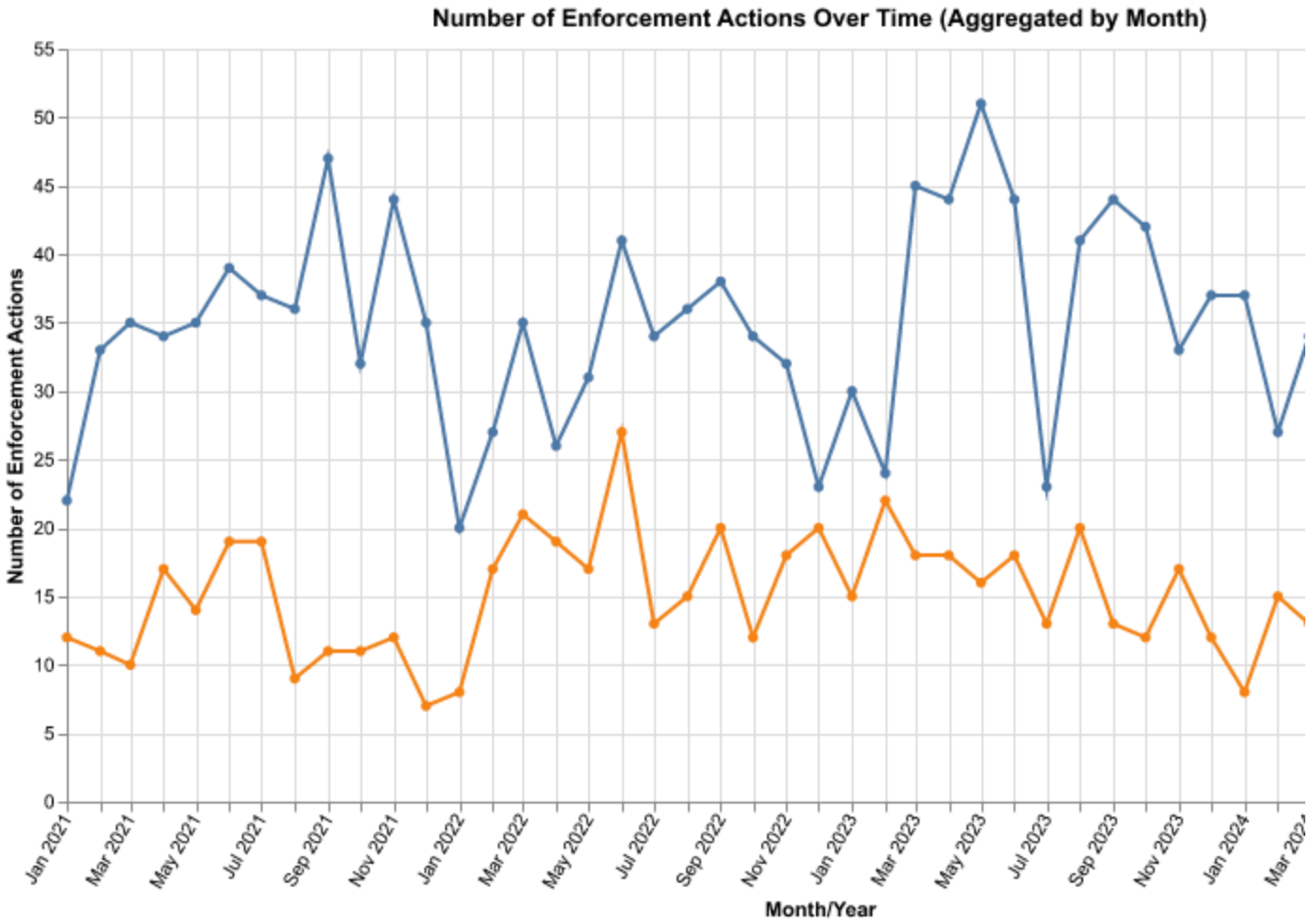
```
import os
base_path = r'c:/Users/danie/Documents/GitHub/Nasser_Daniel_pset5'
path_data = os.path.join(base_path,'enforcement_actions_year_month.csv')
df_2021 = pd.read_csv(path_data)
line_chart = alt.Chart(df_2021).mark_line(point=True).encode(
    x=alt.X('yearmonth(date):T', title = 'Month/Year',
     axis =alt.Axis(tickCount = 48, labelAngle = -60)),
    y=alt.Y('count():Q', title='Number of Enforcement Actions'),
    tooltip=[alt.Tooltip('yearmonth(date):T', title='Month/Year'),
             alt.Tooltip('count():Q', title='Enforcement Count')]
).properties(
    title='Number of Enforcement Actions Over Time (Aggregated by Month)',
    width=800,
    height=400
)
line_chart
```



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

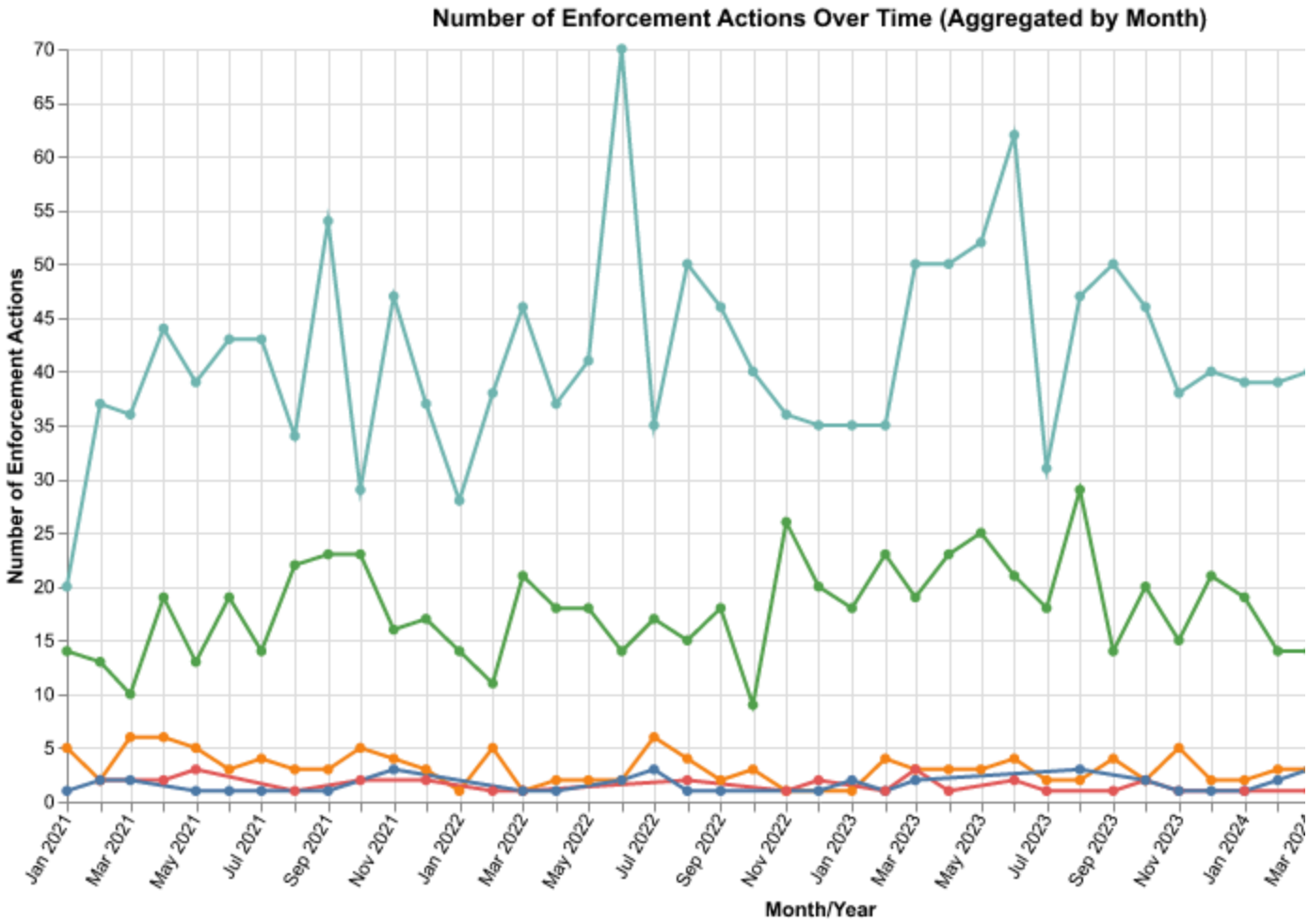
- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"

```
df_criminal_civil = df_2021
df_criminal_civil = df_2021[(df_2021["category"] == "Criminal and Civil Actions") | (df_2021["cat
line_criminal_state = alt.Chart(df_criminal_civil).mark_line(point = True).encode(
    x = alt.X("yearmonth(date):T", title = "Month/Year",
    axis = alt.Axis(tickCount = 48, labelAngle = -60)),
    y = alt.Y("count():Q", title = "Number of Enforcement Actions"),
    color = "category:N",
    tooltip = [alt.Tooltip('yearmonth(date):T', title='Month/Year'),
               alt.Tooltip('count():Q', title='Criminal Actions/State Enforcement Counts')]
).properties(
    title='Number of Enforcement Actions Over Time (Aggregated by Month)',
    width=800,
    height=400
)
line_criminal_state
```




```
five_df = df_2021

#health
health_list = ["Doctor", "Medicare", "Medicaid", "Health", "Physician", "Nurse", "Medi"]
pattern = "|".join(health_list)
five_df["Health Care Fraud"] = (five_df["title"].str.contains(pattern, case = False)).astype(int)

#financial
financial_list = ["Financial", "Bank", "Insurance", "Market", "Credit", "Bankruptcy"]
pattern = "|".join(financial_list)
five_df["Financial Fraud"] = (five_df["title"].str.contains(pattern, case = False)).astype(int)

#drug
drug_list = ["Drug", "Pharma", "Ingredient", "Narcot", "Prescription"]
pattern = "|".join(drug_list)
five_df["Drug Enforcement"] = (five_df["title"].str.contains(pattern, case = False)).astype(int)

#bribery
bribery_list = ["Bribe", "Corruption", "Extortion", "kickback"]
pattern = "|".join(bribery_list)
five_df["Bribery/Corruption"] = (five_df["title"].str.contains(pattern, case = False)).astype(int
```

```
#other
five_df["Other"] = (
    (five_df["Health Care Fraud"] == 0) &
    (five_df["Financial Fraud"] == 0) &
    (five_df["Drug Enforcement"] == 0) &
    (five_df["Bribery/Corruption"] == 0)).astype(int)


# using this to combine s.t. we have one column of categorical variables
def combine_column(row):
    if row["Health Care Fraud"] == 1:
        return "Health Care Fraud"
    elif row["Financial Fraud"] == 1:
        return "Financial Fraud"
    elif row["Drug Enforcement"] == 1:
        return "Drug Enforcement"
    elif row["Bribery/Corruption"] == 1:
        return "Bribery/Corruption"
    else:
        return "Other"

#combining
five_df["combined"] = five_df.apply(lambda x: combine_column(x), axis = 1)

#plotting
line_five = alt.Chart(five_df).mark_line(point = True).encode(
    x = alt.X("yearmonth(date):T", title = "Month/Year",
    axis = alt.Axis(tickCount = 48, labelAngle = -60)),
    y = alt.Y("count():Q", title = "Number of Enforcement Actions"),
    color = "combined:N",
    tooltip = [alt.Tooltip('yearmonth(date):T', title='Month/Year'),
               alt.Tooltip('count():Q', title='Criminal Actions/State Enforcement Counts')]
).properties(
    title='Number of Enforcement Actions Over Time (Aggregated by Month)',
    width=800,
    height=400
)
line_five
```



Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

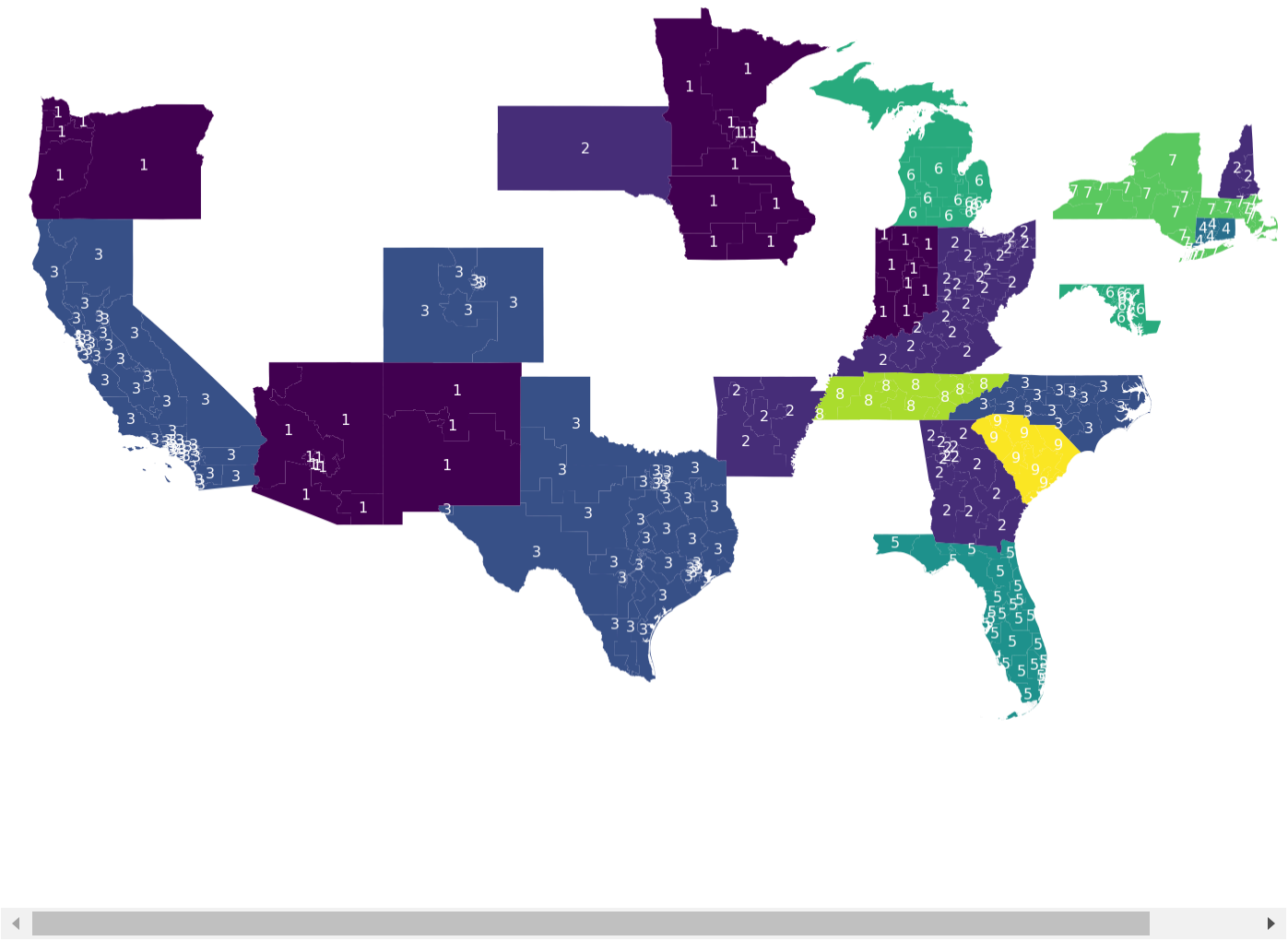
```
filepath = 'c:/Users/danie/Documents/GitHub/Nasser_Daniel_pset5/'
path_districts = os.path.join(
    filepath,'geo_export_3d06657f-30d6-461b-af00-153ad6e863c9.shp')
path_census = os.path.join(filepath,'cb_2018_us_cd116_500k.shp')
df_census = gpd.read_file(path_census)
df_districts = gpd.read_file(path_districts)

#cleaning state names
s2fp = pd.read_csv("fips2county.tsv", sep='\t', header='infer', dtype=str, encoding='latin-1')
s2fp = s2fp[["StateFIPS", "StateName"]]
s2fp.columns = ["STATEFIP", "State"]
s2fp = s2fp.value_counts().reset_index().drop("count", axis = 1)
s2fp = s2fp.set_index("STATEFIP")["State"]

# mapping values from s2fp to df_census
# this gets me state names in census data
```

```
df_census["State"] = df_census["STATEFP"].map(s2fp)

# now need to extract state name in df_2021 and groupby state
df_2021_state = df_2021.dropna(subset = "Agency").drop_duplicates(keep = "first")
df_2021_state = df_2021_state[df_2021_state['Agency'].apply(lambda x: x.startswith("State of"))].
df_2021_state["State"] = df_2021_state["Agency"].apply(lambda x: x.replace("State of", ""))

#this is giving me a count of enforcement actions per state done by state level agencies
df_2021_state_groupby = df_2021_state.groupby("State").size().reset_index(name = "count")

#merging the census geodata with the enforcement activity
#needed to get rid of values in the census that were not in the enforcement activity
df_census = df_census.dropna(subset = "State")
df_census["State"] = df_census["State"].astype(str)

#this is solving an unbelievable whitespace issue
enforcement_states = df_2021_state_groupby["State"].str.strip().to_list()
df_census["State"] = df_census["State"].str.strip()
df_census = df_census[df_census["State"].isin(enforcement_states)]
df_2021_state_groupby["State"] = df_2021_state_groupby["State"].str.strip()

#merge on the two
df_census = df_census.merge(df_2021_state_groupby, how = "left", on = "State")
df_census.head()
```

	STATEFP	CD116FP	AFFGEOID	GEOID	LSAD	CDSESSN	ALAND	AWATER	geometry	State	co
0	47	06	5001600US4706	4706	C2	116	16770155959	324676580	POLYGON ((-87.15023 36.5677, -87.14962 36.5685...	Tennessee	8
1	48	06	5001600US4806	4806	C2	116	5564805243	255530191	POLYGON ((-97.3886 32.61731, -97.38856 32.6189...	Texas	3
2	48	07	5001600US4807	4807	C2	116	419784487	3069802	POLYGON ((-95.77383 29.87516, -95.76962 29.875...	Texas	3
3	48	26	5001600US4826	4826	C2	116	2349987793	191353567	POLYGON ((-97.39826 32.99996, -97.39792 33.013...	Texas	3

	STATEFP	CD116FP	AFFGEOID	GEOID	LSAD	CDSESSN	ALAND	AWATER	geometry	State	co
4	04	08	5001600US0408	0408	C2	116	1398129833	4067789	POLYGON ((-112.62578 33.6533, -112.60846 33.65...	Arizona	1

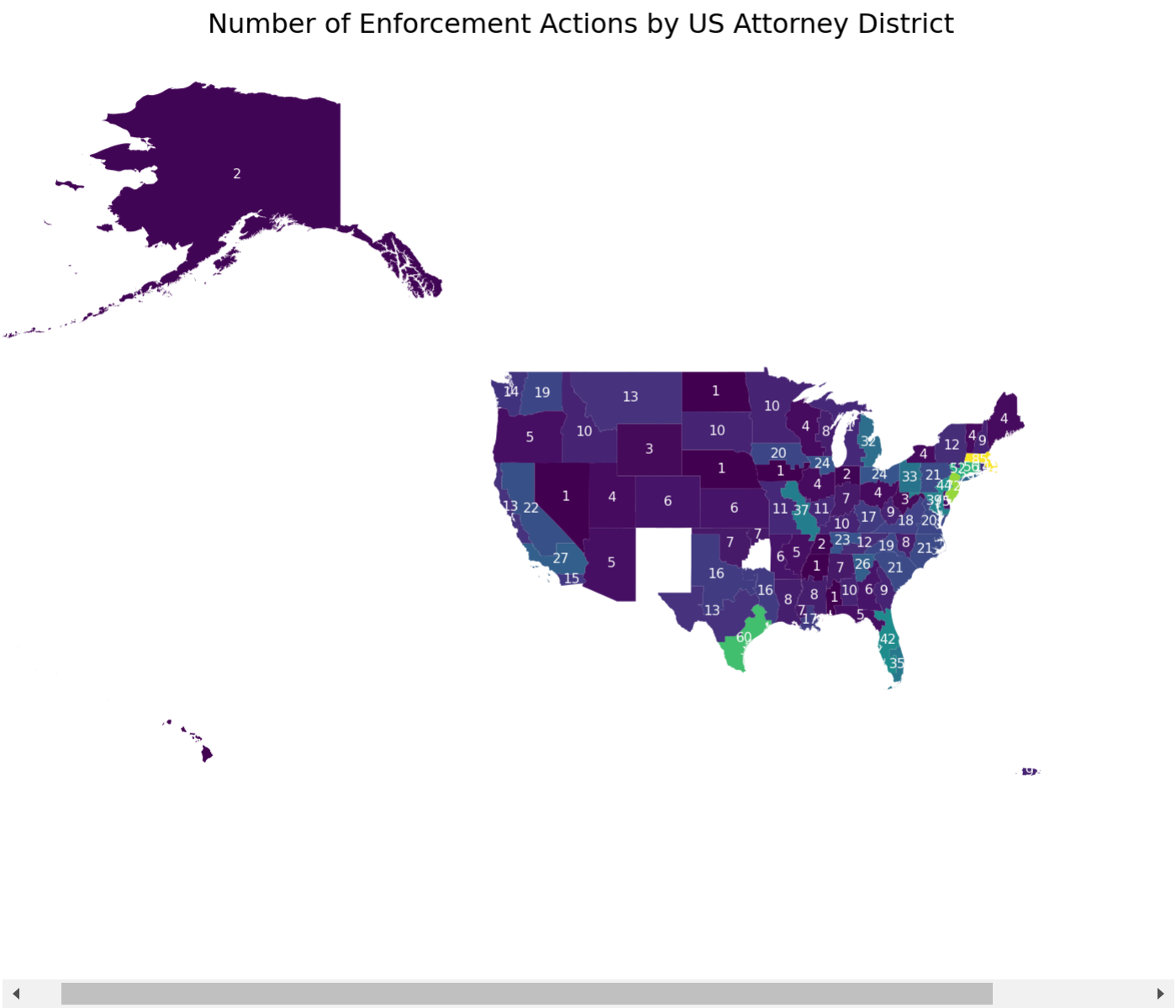
```
'''daniels future plot'''

fig, ax = plt.subplots(1, 1, figsize=(10, 8))
df_census.plot(column='count', ax=ax)
ax.set_title('Number of Enforcement Actions by US State', fontsize = 15, va = "center")
# To zoom in for a larger map

ax.set_xlim(-125, -65)
ax.set_ylim(20, 50)

for idx, row in df_census.iterrows():
    centroid = row['geometry'].centroid
    ax.text(centroid.x, centroid.y, str(row['count']),
            fontsize=6, ha='center', va='center', color='white')
ax.axis('off')
plt.show()
```



2. Map by District (PARTNER 2)

```
df_2021["Agency"] = df_2021["Agency"].apply(
    lambda x: x.split(',', 1)[1].strip() if isinstance(x, str) and ',' in x
    else x)

df_grouped = df_2021.groupby('Agency').size().reset_index(name='count')

df_merged = df_districts.merge(df_grouped, left_on = 'name' ,right_on='Agency',how = 'inner')

fig, ax = plt.subplots(1, 1, figsize=(12, 8))
df_merged.plot(column='count', ax=ax, legend=True)
ax.set_title('Number of Enforcement Actions by US Attorney District')
# To zoom in for a larger map
ax.set_xlim(-180 ,-50)
for idx, row in df_merged.iterrows():
    centroid = row['geometry'].centroid
    ax.text(centroid.x, centroid.y, str(row['count']),
            fontsize=6, ha='center', va='center', color='white')
```

```
ax.axis('off')
plt.show()
```



Extra Credit

1. Merge zip code shapefile with population

```
# Loading data:
filepath = "c:/Users/danie/Documents/GitHub/Nasser_Daniel_pset5"
path_shp = os.path.join(filepath,
 "gz_2010_us_860_00_500k.shp")
df_zip = gpd.read_file(path_shp)

pop_data = os.path.join(filepath,'DECENNIALDHC2020.P1-Data.csv')
```

```
df_zip_pop = pd.read_csv(pop_data)

# Clean zip population columns:
df_zip_pop  = df_zip_pop.iloc[1:]
df_zip_pop['NAME'] = df_zip_pop['NAME'].apply(
    lambda x: x.replace('ZCTA5 ', ''))

# Merging datasets for zip codes:
df_merged_zips = df_zip.merge(df_zip_pop,
 left_on = 'ZCTA5', right_on = 'NAME', how = 'inner')
```

2. Conduct spatial join

```
spatial_joint = gpd.sjoin(
    df_merged_zips,df_districts,how = 'inner', predicate = 'intersects')

district_pop = spatial_joint.groupby('name').agg(
    population = ('P1_001N', 'count'),
    zip = ('ZCTA5', 'first'),
    geometry = ('geometry', 'first')
).reset_index()
```

3. Map the action ratio in each district

```
district_ratio = district_pop.merge(df_merged, on ='name',how = 'inner')

district_ratio['ratio'] = round(
    district_ratio['count']/district_ratio['population'],3)*100

agg_ratio = district_ratio.groupby('name').agg(
    geometry = ('geometry_y','first'),
    ratio = ('ratio','first'))
```

```
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
spatial_joint.plot(column='name', ax=ax,)
ax.set_title('Ratio of Enforcement Actions per population in % since 2021')
# To zoom in for a larger map
ax.set_xlim(-180 ,-50)
for idx, row in agg_ratio.iterrows():
    centroid = row['geometry'].centroid
    ax.text(centroid.x, centroid.y, str(row['ratio']),
            fontsize=6, ha='center', va='center', color='black')
ax.axis('off')
plt.show()
```

Ratio of Enforcement Actions per population in % since 2021