Pset5

Andrew White and Justine Silverstein

2024-11-10

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): Justine Silverstein silversteinj
    - Partner 2 (name and cnet ID): Andrew White awhite6
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: **___** **___** AW & JS
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: **___** 2 (1 from each participant) Late coins left after submission: **___** Justine: 1, Andrew: 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 warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")

import requests
from bs4 import BeautifulSoup
import urllib3
import time

import geopandas as gpd
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

1. Scraping: Go to the first page of the HHS OIG's "Enforcement Actions" page and scrape and collect the following into a dataset:

• Title of the enforcement action • Date • Category (e.g, "Criminal and Civil Actions") • Link associated with the enforcement action

Collect your output into a tidy dataframe and print its head.

```
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
```

```
enforcement_data = []
enforcement_cards = soup.find_all('li', class_='usa-card')

for card in enforcement_cards:
    title_tag = card.find('h2', class_='usa-card__heading')
    title = title_tag.find('a').get_text(strip=True) if title_tag else 'N/A'

    date_tag = card.find('span', class_='text-base-dark')
    date = date_tag.get_text(strip=True) if date_tag else 'N/A'
```

```
    category_tag = card.find('li', class_='usa-tag')
    category = category_tag.get_text(strip=True) if category_tag else 'N/A'

    link_tag = title_tag.find('a') if title_tag else None
    link = link_tag['href'] if link_tag else 'N/A'
    full_link = f"https://oig.hhs.gov{link}"

    enforcement_data.append({
        'Title of Enforcement Action': title,
        'Date': date,
        'Category': category,
        'Associated Link': full_link
    })
    time.sleep(0.2)

OIG_df = pd.DataFrame(enforcement_data)
print(OIG_df.head())
```

```
                      Title of Enforcement Action             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

                                    Associated 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)

2. Crawling: Then for each enforcement action, click the link and collect the name of the agency involved (e.g., for this link, it would be U.S. Attorney's Office, Eastern District of Washington). Update your dataframe with the name of the agency and print its head.

Hint: if you go to James A. Robinson's profile page at the Nobel Prize website here, right click anywhere along the line "Affiliation at the time of the award: University of Chicago, Chicago, IL, USA", and select Inspect, you'll see that this affiliation information is located at the third

tag out of five

tags under the

. Think about how you can select the third element of

out of five

elements so you're sure you scrape the affiliation information, not other. This way, you can scrape the name of agency to answer this question.

```
import requests
from lxml import html

def extract_agency_from_page(url):
    try:
        response = requests.get(url)
        if response.status_code != 200:
            return 'Failed to retrieve page'

        tree = html.fromstring(response.content)

        agency_text =
↪   tree.xpath('/html/body/main/div/div[2]/article/div/ul/li[2]/text()')

        if agency_text:
            return agency_text[0].strip()
        else:
            return 'No Agency Info'

    except Exception as e:
        print(f"Error processing {url}: {e}")
        return 'Error Fetching Agency'
```

```python
import time

def add_agency_column_to_OIG(OIG_df):
    agency_names = []

    for index, row in OIG_df.iterrows():
        link = row['Associated Link']
        if link:
            print(f"Processing link {index + 1}: {link}")
            agency_name = extract_agency_from_page(link)
        else:
            agency_name = 'No Link Available'
        agency_names.append(agency_name)
        time.sleep(1)

    OIG_df['Agency'] = agency_names
    return OIG_df

OIG_df = add_agency_column_to_OIG(OIG_df)
print(OIG_df.head())
```

```
Processing link 1:
https://oig.hhs.gov/fraud/enforcement/pharmacist-and-brother-convicted-of-15m-medicare-medica
Processing link 2:
https://oig.hhs.gov/fraud/enforcement/boise-nurse-practitioner-sentenced-to-48-months-for-cor
Processing link 3:
https://oig.hhs.gov/fraud/enforcement/former-traveling-nurse-pleads-guilty-to-tampering-with-
Processing link 4:
https://oig.hhs.gov/fraud/enforcement/former-arlington-resident-sentenced-to-prison-for-discl
Processing link 5:
https://oig.hhs.gov/fraud/enforcement/paroled-felon-sentenced-to-six-years-for-fraudulent-use
Processing link 6:
https://oig.hhs.gov/fraud/enforcement/former-licensed-counselor-sentenced-for-defrauding-medi
Processing link 7:
https://oig.hhs.gov/fraud/enforcement/macomb-county-doctor-and-pharmacist-agree-to-pay-700948
Processing link 8:
https://oig.hhs.gov/fraud/enforcement/rocky-hill-pharmacy-and-its-owners-indicted-for-conspir
Processing link 9:
https://oig.hhs.gov/fraud/enforcement/north-texas-medical-center-pays-142-million-to-resolve-
Processing link 10:
https://oig.hhs.gov/fraud/enforcement/new-england-doctor-pleads-guilty-to-drug-distribution-c
```

```
Processing link 11:
https://oig.hhs.gov/fraud/enforcement/attorney-general-alan-wilson-announces-upstate-woman-cl
Processing link 12:
https://oig.hhs.gov/fraud/enforcement/st-louis-county-woman-accused-of-3-million-home-health-
Processing link 13:
https://oig.hhs.gov/fraud/enforcement/lab-owner-and-marketing-company-owner-both-found-guilty
Processing link 14:
https://oig.hhs.gov/fraud/enforcement/compound-ingredient-supplier-medisca-inc-to-pay-2175m-t
Processing link 15:
https://oig.hhs.gov/fraud/enforcement/the-new-mexico-department-of-justice-charges-former-new
Processing link 16:
https://oig.hhs.gov/fraud/enforcement/nashville-woman-indicted-charged-in-tbi-medicaid-patien
Processing link 17:
https://oig.hhs.gov/fraud/enforcement/michael-depalma-md-and-virginia-i-spine-physicians-agre
Processing link 18:
https://oig.hhs.gov/fraud/enforcement/columbus-doctor-his-clinic-convicted-of-15-million-medi
Processing link 19:
https://oig.hhs.gov/fraud/enforcement/mercy-health-youngstown-agreed-to-pay-69000-for-alleged
Processing link 20:
https://oig.hhs.gov/fraud/enforcement/quincy-based-physician-group-to-pay-650000-to-resolve-a
                        Title of Enforcement Action              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

                                     Associated 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)

#There will be a while loop that will be conditioned on whether or not the webpage has a date earlier than the earliest date we want to scrape. I determined, using trial and error and ChatGPT, that I would need a while loop while coding the actual function, so I am updating the pseudo-code to be more reflective of the actual logic.

#receive input of web link and #a date object called year_month def year_scraper(site, year_month):

#receiving object enforcement_actions_year_month = []

#first loop, conditioned on the year_month #being >= 2013 if year_month >= 2013:

```
#run while loop to run data through
while date tag is >= year_month:
    #run function for reading in the html request
    read_data(site)

    #creating a dataframe out of certain
    #contents from the html request
    assembling dataframe containing:
    {Title of enforcement action, date, category, the link to the
    enforcement action, and the name of the associated agency.}

    #run a function within this function
    #that selects the html tag associated with
    #the next page link and then requests it,
    #in order to move between pages.
    #the tag is "a", and the class_="pagination-next"

    #append the contents of the dataframe,
    #from each iteration of the loop, to
    #the receiving object
    enforcement_actions_year_month.append(output)
```

#if year_month < 2013, print that it needs to be #later or equal to 2013 else: print("Turn back! We will only access data from 2013 and later!")

#finally, return the receiving object, saving it to csv: return(enforcement_actions_year_month.to_csv("address index = False))

b.

```
import datetime as datetime
from datetime import datetime
from datetime import date

def enforce_frame(soup):
  #container to be turned into df
  enforcement_data = []
  #capture all 'li' tag contents
  enforcement_cards = soup.find_all('li', class_='usa-card')
  #iterate through enforcement_cards for the 4 relevant tags
  for card in enforcement_cards:
      title_tag = card.find('h2', class_='usa-card__heading')
      title = title_tag.find('a').get_text(strip=True) if title_tag else
↪  'N/A'

      date_tag = card.find('span', class_='text-base-dark')
      date = date_tag.get_text(strip=True) if date_tag else 'N/A'

      category_tag = card.find('li', class_='usa-tag')
      category = category_tag.get_text(strip=True) if category_tag else 'N/A'

      link_tag = title_tag.find('a') if title_tag else None
      link = link_tag['href'] if link_tag else 'N/A'
      full_link = f"https://oig.hhs.gov{link}"

      enforcement_data.append({
        'Title of Enforcement Action': title,
        'Date': date,
        'Category': category,
        'Associated Link': full_link
    })
      time.sleep(0.2)

  OIG_df = pd.DataFrame(enforcement_data)
  return(OIG_df)
```

#select the enforcement links and iterate through each

```
# Base URL for the site
# this will be required for the function
# define this before running the function
base_url = "https://oig.hhs.gov"

def add_agency_column_to_OIG(OIG_df):
    agency_names = []

    for index, row in OIG_df.iterrows():
        link = row['Associated Link']
        if link:
            print(f"Processing link {index + 1}: {link}")
            agency_name = extract_agency_from_page(link)
        else:
            agency_name = 'No Link Available'
        agency_names.append(agency_name)
        time.sleep(1)
    OIG_df['Agency'] = agency_names
    return OIG_df
OIG_df = add_agency_column_to_OIG(OIG_df)
print(OIG_df.head())
```

```
Processing link 1:
https://oig.hhs.gov/fraud/enforcement/pharmacist-and-brother-convicted-of-15m-medicare-medica
Processing link 2:
https://oig.hhs.gov/fraud/enforcement/boise-nurse-practitioner-sentenced-to-48-months-for-cor
Processing link 3:
https://oig.hhs.gov/fraud/enforcement/former-traveling-nurse-pleads-guilty-to-tampering-with-
Processing link 4:
https://oig.hhs.gov/fraud/enforcement/former-arlington-resident-sentenced-to-prison-for-discl
Processing link 5:
https://oig.hhs.gov/fraud/enforcement/paroled-felon-sentenced-to-six-years-for-fraudulent-use
Processing link 6:
https://oig.hhs.gov/fraud/enforcement/former-licensed-counselor-sentenced-for-defrauding-medi
Processing link 7:
https://oig.hhs.gov/fraud/enforcement/macomb-county-doctor-and-pharmacist-agree-to-pay-700948
Processing link 8:
https://oig.hhs.gov/fraud/enforcement/rocky-hill-pharmacy-and-its-owners-indicted-for-conspir
Processing link 9:
https://oig.hhs.gov/fraud/enforcement/north-texas-medical-center-pays-142-million-to-resolve-
```

```
Processing link 10:
https://oig.hhs.gov/fraud/enforcement/new-england-doctor-pleads-guilty-to-drug-distribution-c
Processing link 11:
https://oig.hhs.gov/fraud/enforcement/attorney-general-alan-wilson-announces-upstate-woman-ch
Processing link 12:
https://oig.hhs.gov/fraud/enforcement/st-louis-county-woman-accused-of-3-million-home-health-
Processing link 13:
https://oig.hhs.gov/fraud/enforcement/lab-owner-and-marketing-company-owner-both-found-guilty
Processing link 14:
https://oig.hhs.gov/fraud/enforcement/compound-ingredient-supplier-medisca-inc-to-pay-2175m-t
Processing link 15:
https://oig.hhs.gov/fraud/enforcement/the-new-mexico-department-of-justice-charges-former-new
Processing link 16:
https://oig.hhs.gov/fraud/enforcement/nashville-woman-indicted-charged-in-tbi-medicaid-patien
Processing link 17:
https://oig.hhs.gov/fraud/enforcement/michael-depalma-md-and-virginia-i-spine-physicians-agre
Processing link 18:
https://oig.hhs.gov/fraud/enforcement/columbus-doctor-his-clinic-convicted-of-15-million-medi
Processing link 19:
https://oig.hhs.gov/fraud/enforcement/mercy-health-youngstown-agreed-to-pay-69000-for-allegec
Processing link 20:
https://oig.hhs.gov/fraud/enforcement/quincy-based-physician-group-to-pay-650000-to-resolve-a
                        Title of Enforcement Action              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

                                       Associated 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...
```

#create a loop to loop through the "Date" category

```
import datetime as datetime
from datetime import datetime
from datetime import date

#set year_month as the first month of 2013
target_date = date(2013, 1, 1)

#site will be the website we've been using
site = "https://oig.hhs.gov/fraud/enforcement/"

# Base URL for the site
# (will be required for get_links() function)
base_url = "https://oig.hhs.gov"

#adding code from Step 1
#make it a function for easier manipulation
def reading_data(url): #url will be the appropriate link
  response = requests.get(url)
  soup = BeautifulSoup(response.content, "html.parser")
  return(soup)

#code for selecting specifically date tags and saving them
def find_dates(soup):
  date_list = []
  enforcement_cards = soup.find_all('li', class_='usa-card')
  for card in enforcement_cards:
    date_tag = card.find('span', class_='text-base-dark')
    the_date = date_tag.get_text(strip=True) if date_tag else 'N/A'
    my_date = datetime.strptime(the_date, "%B %d, %Y")
    date_list.append(my_date)
  return(date_list)

#code for moving between pages based on date tags
def site_moving(site):
```

```
    #define the dataframe that will receive
    #all info
    full_data = pd.DataFrame(columns = ['Title of Enforcement Action',
↪   'Date', 'Category', 'Associated Link', 'Agency'])
    current_url = site
    while current_url:
        #read in data
        soup = reading_data(current_url)
        #create list of dates
        date_list = find_dates(soup)
        #fill dataframe
        df = enforce_frame(soup)
        #add "Agents" to dataframe
        OIG_df = add_agency_column_to_OIG(df)
        #concat data
        full_data = pd.concat([full_data, OIG_df], ignore_index=True)

        # Check if any date on this page is before the target date
        if any(d.date() < target_date for d in date_list):
            print("Stopping, found a date before target:", target_date)
            break

        # Find next page link
        next_link_tag = soup.find("a", class_="pagination-next")
        if next_link_tag:
            next_link = next_link_tag.get("href")
            current_url = f"{site}{next_link}"

        else:
            print("No more pages to navigate.")
            break
    return full_data

#each time you move to a new page, make a date_list
#if no object in that date list is less than 2023-1
#find the next page tab and request that link,
#put that link in reading_data()
```

Attribution: Asked ChatGPT how to create datetime object that would capture both year and month at once. Asked ChatGPT how to convert an object into a datetime object. Also asked ChatGPT how to create a function that would loop in order to move forward through pages. The main change was switching to a while loop.

#Put everything together

```
import datetime as datetime
from datetime import datetime
from datetime import date

#set year_month as the first month of 2013
target_date = date(2023, 1, 1)
#site will be the website we've been using
site = "https://oig.hhs.gov/fraud/enforcement/"
# Base URL for the site
# (will be required for get_links() function)
base_url = "https://oig.hhs.gov"

def the_crawler(site, target_date, base_url):
  #read in the data
  soup = reading_data(site)
  #check whether your year is greater than
  #or equal to 2013
  if target_date <= date(2013, 1, 1):
    print("Turn back! Subset for 2013 and later")
  else:
    #insert the function for scraping
    all_the_data = site_moving(site)
  return all_the_data.to_csv("enforcement_actions_year_month.csv", index =
  ↪  False)
```

Run the_crawler

```
#set year_month as the first month of 2013
target_date = date(2023, 1, 1)
#site will be the website we've been using
site = "https://oig.hhs.gov/fraud/enforcement/"
# Base URL for the site
# (will be required for get_links() function)
base_url = "https://oig.hhs.gov"

the_crawler(site, target_date, base_url)
```

Load the dataframe from the csv

```
#enforcement_actions_2023 =
 ↪  pd.read_csv("C:/Users/andre/Documents/GitHub/Andrew_and_Justine_Pset5/enforcement_actions

enforcement_actions_2023 =
 ↪  pd.read_csv("/Users/justinesilverstein/Desktop/Andrew_and_Justine_Pset5/enforcement_actio

#drop the the columns that slipped through the cracks
enforcement_actions_2023 = enforcement_actions_2023.drop([1534, 1535, 1536,
 ↪  1537, 1538, 1539])
```

Step 2 Part B Answer: Originally, the output of the crawler was 1,540 entries, meaning 1,540 enforcement actions. However, it became clear that, due to the nature of the loop conditioning the scraper, a few entries past 2023 had been captured. We located them, there were 6, and removed them, bringing the correct total down to 1,534 enforcement actions. The earliest entry we grabbed is from January 3rd, 2023. The action was a podiatrist settling false billing allegations by paying $90,000.00. It has been filed under Criminal and Civil Actions, and the associated agency is the U.S. Attorney's Office in the Southern District of Texas.

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

```
import datetime as datetime
from datetime import datetime
from datetime import date

#set year_month as the first month of 2013
target_date = date(2021, 1, 1)
#site will be the website we've been using
site = "https://oig.hhs.gov/fraud/enforcement/"
# Base URL for the site
# (will be required for get_links() function)
base_url = "https://oig.hhs.gov"

def the_crawler(site, target_date, base_url):
  #read in the data
  soup = reading_data(site)
  #check whether your year is greater than
  #or equal to 2013
  if target_date <= date(2013, 1, 1):
    print("Turn back! Subset for 2013 and later")
  else:
    #insert the function for scraping
```

```
    all_the_data = site_moving(site)
  return all_the_data.to_csv("enforcement_actions_year_month.csv", index =
   ↪  False)

the_crawler(site, target_date, base_url)
```

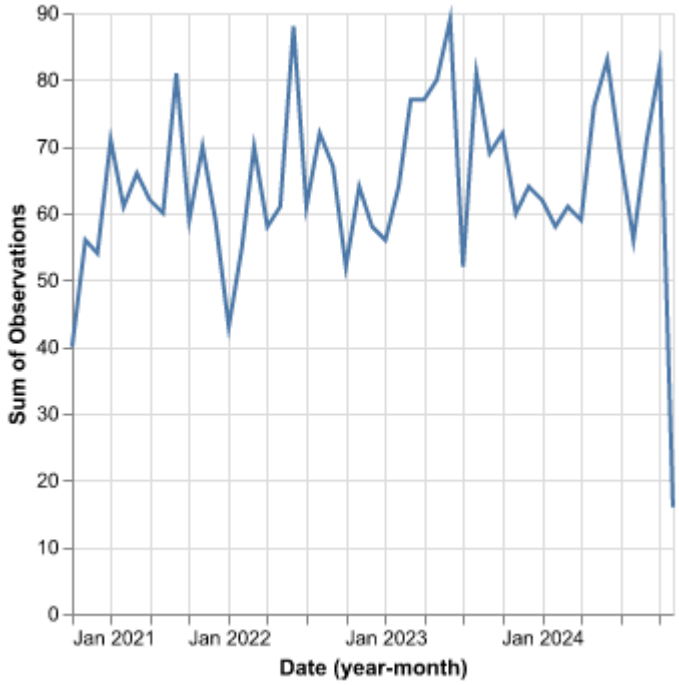
```
enforcement_actions_2021 =
 ↪  pd.read_csv("/Users/justinesilverstein/Desktop/Andrew_and_Justine_Pset5/enforcement_acti

#enforcement_actions_2021 =
 ↪  pd.read_csv("C:/Users/andre/Documents/GitHub/Andrew_and_Justine_Pset5/enforcement_actions

#trim the actions that slipped through the cracks
enforcement_actions_2021 = enforcement_actions_2021.drop([3022, 3023, 3024,
 ↪  3025, 3026, 3027, 3028, 3029, 3030, 3031, 3032, 3033, 3034, 3035, 3036,
 ↪  3037, 3038, 3039])

#enforcement_actions_2021 =
 ↪  pd.read_csv("C:/Users/andre/Documents/GitHub/Andrew_and_Justine_Pset5/enforcement_actions
```

Step 3: Plot data based on scraped data

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

```
#make a column returning 1 for each observation
def  one_dummy(C):
  observation = []
  i = 0
  #produces a 1 for every entry
  while 1 > 0 and i < C:
    observation.append(1)
    i += 1
  return(observation)

#add observation column to df
enforcement_actions_2021["Observations"] =
 ↪  one_dummy(len(enforcement_actions_2021))

#make a line chart that is
```

```
#"Observation" by yearmonth("Date")
#Observation represents 1 enforcement
alt.Chart(enforcement_actions_2021).mark_line().encode(
  alt.X("yearmonth(Date)"),
  alt.Y("sum(Observations)")
)
```



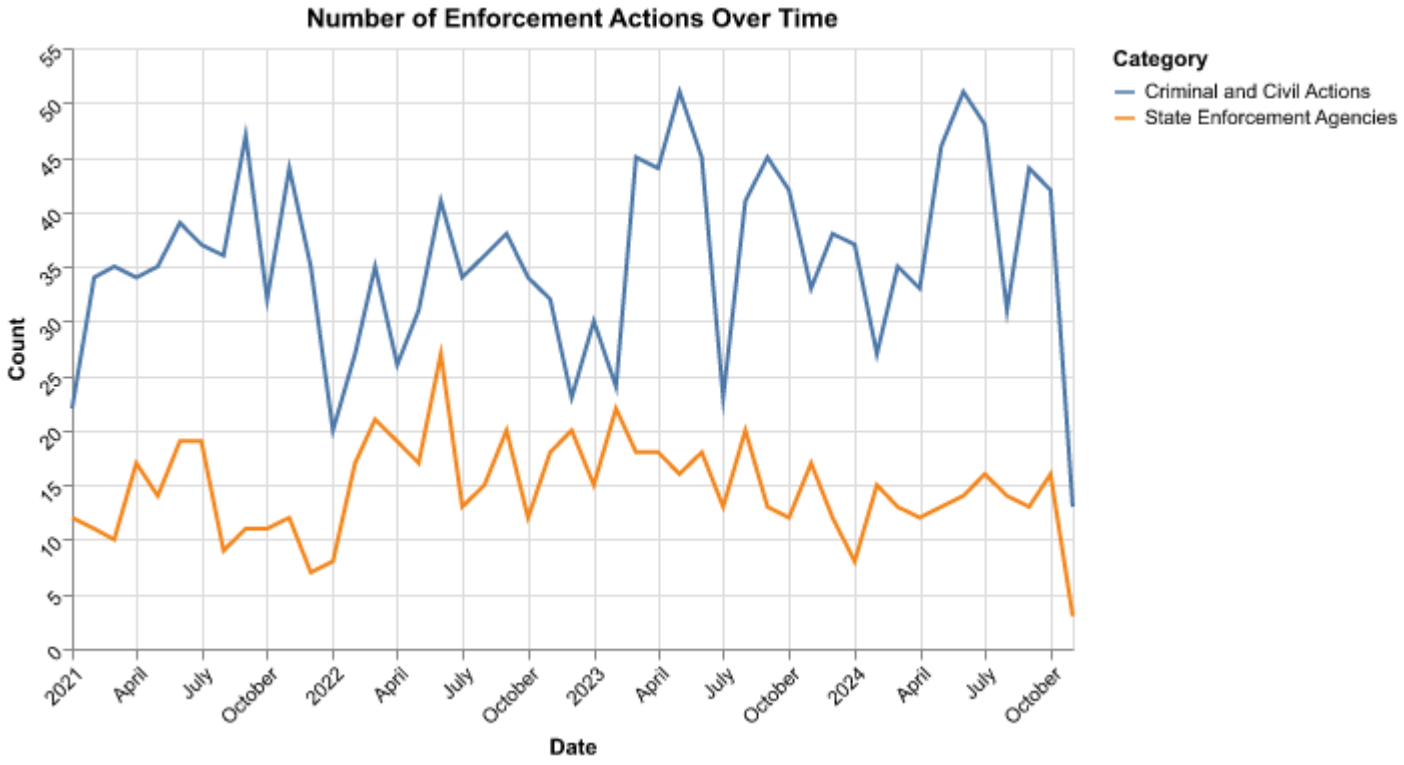
2. Plot the number of enforcement actions categorized: (PARTNER 1)

Plot a line chart that shows: the number of enforcement actions split out by:

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

```
enforcement_actions_2021['Date'] =
 ↪  pd.to_datetime(enforcement_actions_2021['Date'], errors='coerce')

category_filter =
 ↪  enforcement_actions_2021[enforcement_actions_2021['Category'].isin(['Criminal
 ↪  and Civil Actions', 'State Enforcement Agencies'])]

category_counts =
 ↪  category_filter.groupby([category_filter['Date'].dt.to_period('M'),
 ↪  'Category']).size().reset_index(name='Count')
```

```
category_counts['Date'] = category_counts['Date'].dt.to_timestamp()

line_chart = alt.Chart(category_counts).mark_line().encode(
    x='Date:T',
    y='Count:Q',
    color='Category:N',
    tooltip=['Date:T', 'Category:N', 'Count:Q']
).properties(
    title='Number of Enforcement Actions Over Time',
    width=500,
    height=300
).configure_axis(
    labelAngle=-45
)

line_chart.show()
```

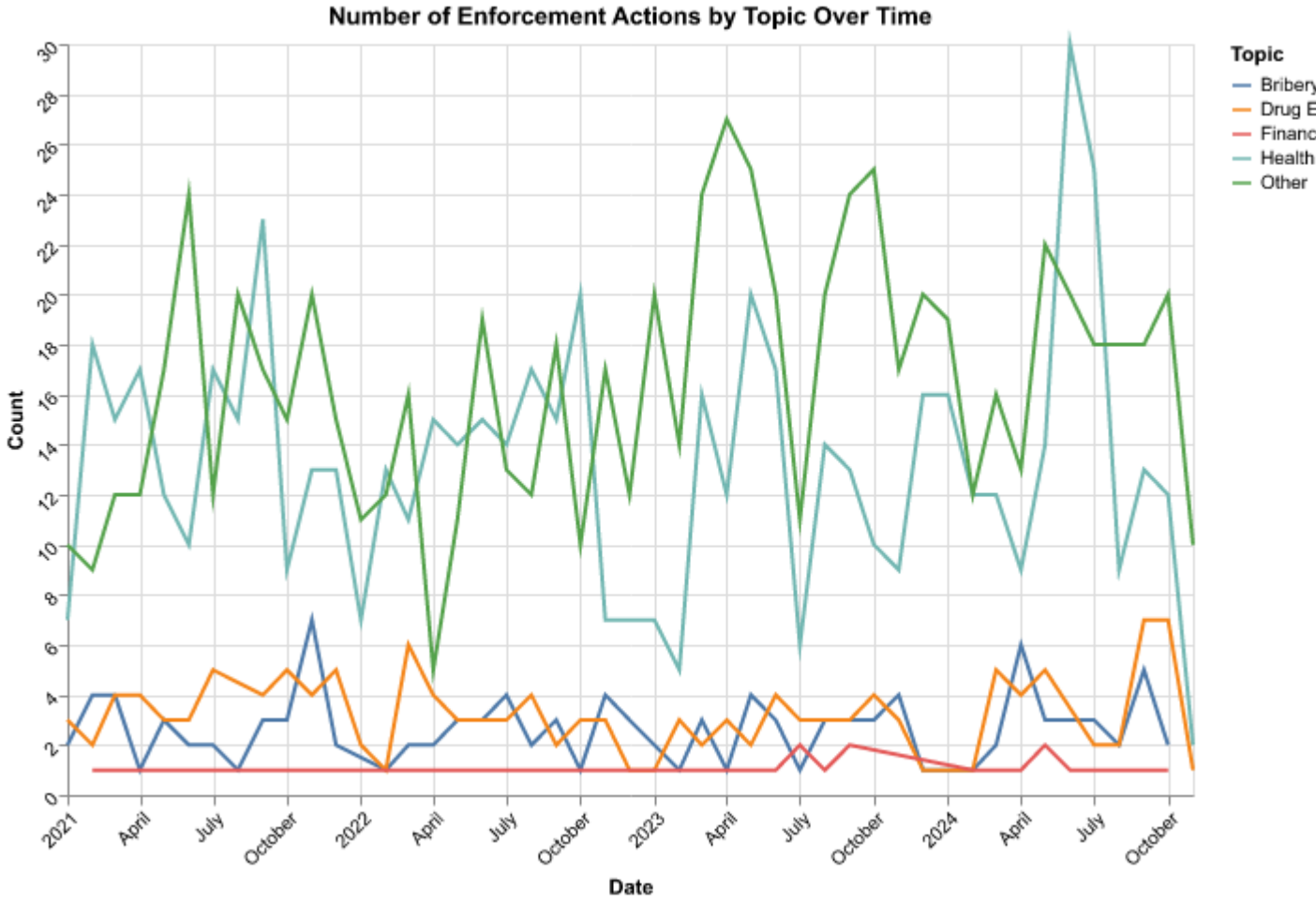


- based on five topics

Hint: You will need to divide the five topics manually by looking at the title and assigning the relevant topic. For example, if you find the word "bank" or "financial" in the title of an action, then that action should probably belong to "Financial Fraud" topic.

```
enforcement_actions_2021['Date'] =
 ↪  pd.to_datetime(enforcement_actions_2021['Date'], errors='coerce')

criminal_and_civil_df =
 ↪  enforcement_actions_2021[enforcement_actions_2021['Category'] ==
 ↪  'Criminal and Civil Actions']

topics_keywords = {
    'Health Care Fraud': ['health', 'hospital'],
    'Financial Fraud': ['bank', 'financial', 'money laundering',
     ↪  'investment', 'tax evasion'],
    'Drug Enforcement': ['drug', 'trafficking', 'narcotics', 'opioid'],
    'Bribery/Corruption': ['bribery', 'corruption', 'kickback'],
    'Other': []
}

def assign_topic(title):
    title = title.lower()
    for topic, keywords in topics_keywords.items():
        if any(keyword in title for keyword in keywords):
            return topic
    return 'Other'

criminal_and_civil_df['Topic'] = criminal_and_civil_df['Title of Enforcement
 ↪  Action'].apply(assign_topic)

category_counts =
 ↪  criminal_and_civil_df.groupby([criminal_and_civil_df['Date'].dt.to_period('M'),
 ↪  'Topic']).size().reset_index(name='Count')

category_counts['Date'] = category_counts['Date'].dt.to_timestamp()

line_chart = alt.Chart(category_counts).mark_line().encode(
    x='Date:T',
    y='Count:Q',
    color='Topic:N',
    tooltip=['Date:T', 'Topic:N', 'Count:Q']
).properties(
```

```
    title='Number of Enforcement Actions by Topic Over Time',
    width=600,
    height=400
).configure_axis(
    labelAngle=-45
)

line_chart.show()
```



Step 4: Create maps of enforcement activity

1. Map by State (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!

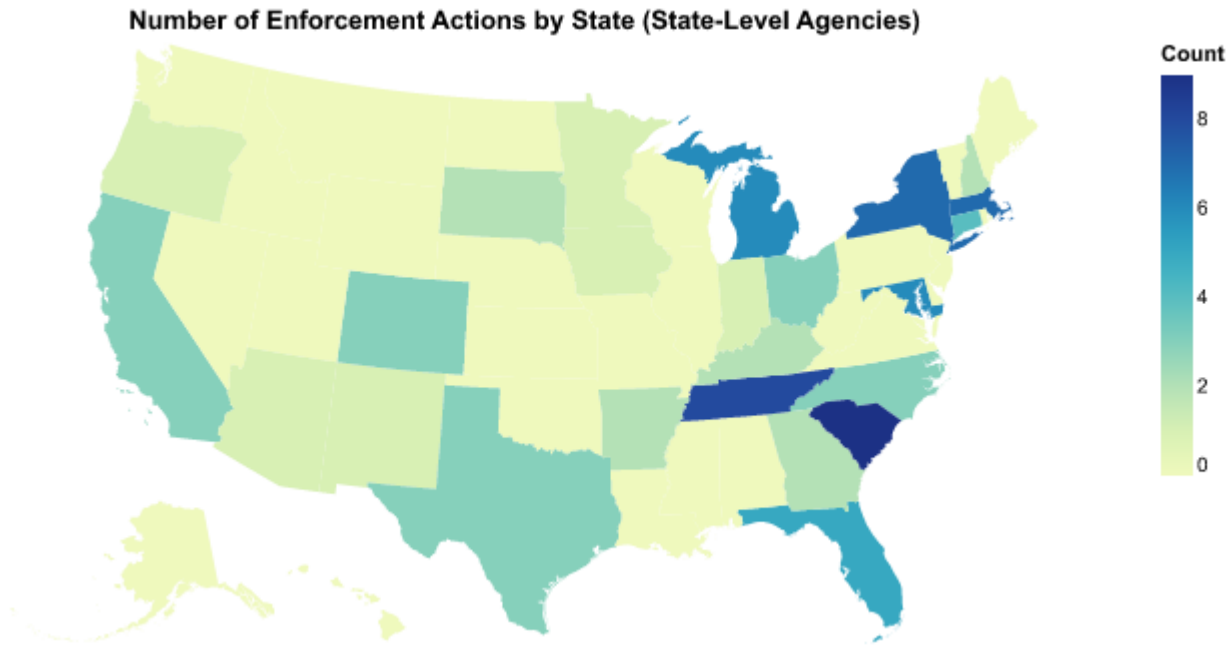
Below is the choropleth map for state level agencies. However I noticed labeling issues from the beginning as we had a lot of NAs or agencies were not labeled consistently.

```
import geopandas as gpd
import re

#shapefile_path =
 ↪  "C:/Users/andre/Documents/GitHub/Andrew_and_Justine_Pset5/cb_2018_us_state_500k.shp"

shapefile_path =
 ↪  "/Users/justinesilverstein/Desktop/Andrew_and_Justine_Pset5/cb_2018_us_state_500k.shp"
us_states = gpd.read_file(shapefile_path)

state_actions_df =
 ↪  enforcement_actions_2021[enforcement_actions_2021['Category'] == 'State
 ↪  Enforcement Agencies']

state_actions_df = state_actions_df[state_actions_df['Agency'].notna()]

def extract_state_from_agency(agency_name):
    if isinstance(agency_name, str):
        match = re.search(r"State of\s+([A-Za-z\s]+)", agency_name)
        if match:
            return match.group(1).strip()
        match = re.search(r"Attorney General of\s+([A-Za-z\s]+)",
 ↪  agency_name)
        if match:
            return match.group(1).strip()
        match = re.search(r"([A-Za-z\s]+)", agency_name)
        if match:
            return match.group(1).strip()
    return None

state_actions_df['State'] =
 ↪  state_actions_df['Agency'].apply(extract_state_from_agency)
```

```
print(state_actions_df[['Agency', 'State']].head())
```

```
                    Agency           State
10  State of South Carolina  South Carolina
14      State of New Mexico      New Mexico
15       State of Tennessee       Tennessee
17                     Ohio            Ohio
19   State of Massachusetts   Massachusetts
```

```
state_counts = state_actions_df['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Count']

state_counts['State'] = state_counts['State'].str.strip()

state_counts = state_counts.rename(columns={'State': 'State_Name'})

merged = us_states.merge(state_counts, left_on='NAME', right_on='State_Name',
 ↪  how='left')

merged['Count'] = merged['Count'].fillna(0)

choropleth = alt.Chart(merged).mark_geoshape().encode(
    color='Count:Q',
    tooltip=['NAME:N', 'Count:Q']
).project('albersUsa').properties(
    title="Number of Enforcement Actions by State (State-Level Agencies)",
    width=600,
    height=300
).configure_view(
    stroke=None
)

choropleth.show()
```



2. Map by District (PARTNER 2)

```
#read in the shapefile
#att_district =
 ↪  gpd.read_file("C:/Users/andre/Documents/GitHub/Andrew_and_Justine_Pset5/geo_export_994e6

att_district =
 ↪  gpd.read_file("/Users/justinesilverstein/Desktop/Andrew_and_Justine_Pset5/geo_export_d66

#for handling strange projection sizes
att_district = att_district.to_crs("EPSG:5070")

#clean enforcement_actions_2021 such that only actions
#referencing a U.S. Attorney's Office are included
attorney_office = "U.S. Attorney's Office"

def only_attorney(df):
  #receiving box
  box = []
  for index, row in df.iterrows():
      if isinstance(row["Agency"], str) and attorney_office in row["Agency"]:
```

```
        box.append(row)
      else:
        pass
  return pd.DataFrame(box)

#run function
attorney_data = only_attorney(enforcement_actions_2021)

#create new variable called District, by removing attorney_office
#str objects from each Agency entry in attorney_data

def strip_away(df):
  df["District"] = df["Agency"].str.replace("U.S. Attorney's Office,", "",
 ↪  regex = False)
  #make into a string
  df["District"] = df["District"].astype(str)
  #remove leading spaces
  df["District"] = df["District"].str.lstrip()
  return df

strip_data = strip_away(attorney_data)

strip_data

#ensure that att_district["judicial_d"] is a string
att_district["judicial_d"] = att_district["judicial_d"].astype(str)
```

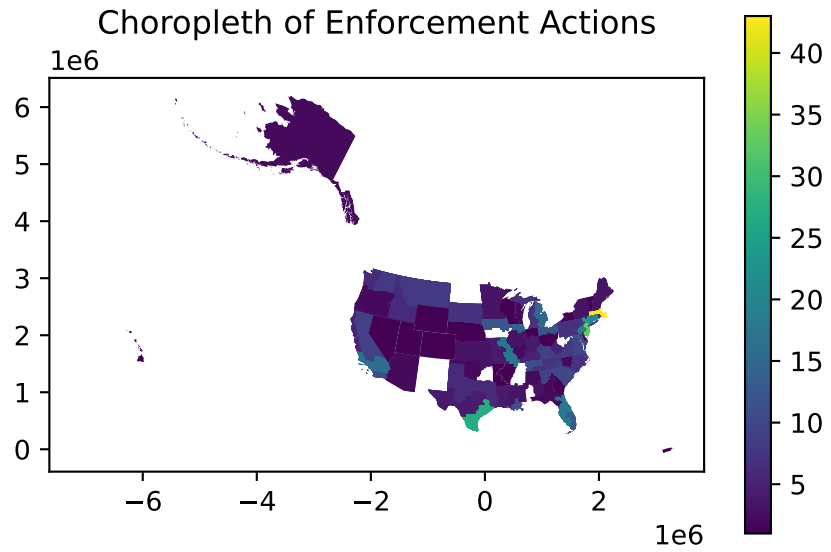
Attribution: asked ChatGPT how to refine my code to make it more readable and able to skip over non-strong Agency values when subsetting. Also asked how to remove elements of a string object. Also used: https://www.digitalocean.com/community/tutorials/python-trim-string-rstrip-lstrip-strip

Merge shapefile with attorney_data

```
#for dislaying the plot
import matplotlib.pyplot as plt

attorney_shape = pd.merge(att_district, strip_data, left_on = "judicial_d",
 ↪  right_on = "District", how = "left")

#groupby "District" and get whole thing
grouped_shape =
 ↪  attorney_shape.groupby("District")["Observations"].count().reset_index()
```

```
#merge data to add this column to shapefile
second_shape = attorney_shape.merge(grouped_shape, on = "District")

#create the choropleth

choropleth = second_shape.plot(column = "Observations_y", legend = True)

choropleth.set_title("Choropleth of Enforcement Actions")
```

```
Text(0.5, 1.0, 'Choropleth of Enforcement Actions')
```



Attribution: Asked ChatGPT how to use a Groupby object in a choropleth, also used: https://saturncloud.io/blog/how-to-convert-dataframegroupby-object-to-dataframe-in-pandas/ Also asked Chat GPT how to use EPSG:5070 to handle projection issues, and used it for ideas for adding elements to my plot, namely the title

Extra Credit

1. **Merge zip code shapefile with population**

2. **Conduct spatial join**

3. **Map the action ratio in each district**