30538 Problem Set 5: Web Scraping

Neil Stein and Mitch Bobbin

2001-11-10

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 (Mitch Bobbin mbobbin):
    - Partner 2 (Neil Stein neilstein):
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: **\MB** **\NS**
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 a piece)_** Late coins left after submission: **\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")
```

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
import pandas as pd
from bs4 import BeautifulSoup
import requests

url = "https://oig.hhs.gov/fraud/enforcement/"

response = requests.get(url)

#parse the HTML content
soup = BeautifulSoup(response.text, "html.parser")

#find all h2 tags for enforcement action titles
tag_h = soup.find_all("h2", class_="usa-card__heading")

#find all span tags for dates
span_dates = soup.find_all("span", class_="text-base-dark padding-right-105")

#find all ul tags for categories
ul_categories = soup.find_all("ul", class_="display-inline add-list-reset")

#create empty lists to store our extracted data in
enforcement = []
dates = []
categories = []
hyperlinks = []
```

```
#loop over the objects we created from finding tags
for h2, span, ul in zip(tag_h, span_dates, ul_categories):
    #extract the hyperlink and text from h2
    a_tag = h2.find("a")
    if a_tag:
        enforcement.append(a_tag.text.strip())
        hyperlinks.append("https://oig.hhs.gov"+a_tag['href'])
        #extract the hyperlink

    #extract the date and strip whitespace
    dates.append(span.text.strip())

    #extract the category text
    category_text = [li.text.strip() for li in ul.find_all("li")]
    categories.append(", ".join(category_text))
    #join multiple categories if present

#create the df
df = pd.DataFrame({
    "Title of Enforcement Action": enforcement,
    "Date": dates,
    "Category": categories,
    "Hyperlink": hyperlinks
})

print(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

                                          Hyperlink
```

```
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 requests
from bs4 import BeautifulSoup

tag_h = soup.find_all("h2", class_="usa-card__heading")
#get the hyperlinks and store them in a list
my_links=[]
for h2 in tag_h:
  links=h2.find_all("a", href=True)
  for link in links:
    my_links.append(
      "https://oig.hhs.gov"+link.get("href"))

#create empty list to store all the hyperlinks into
agency = []
#loop over all the urls in the mylinks list to find the
#agency
for url in my_links:
    response = requests.get(url)
    page_soup = BeautifulSoup(response.text, "html.parser")
    li_tags = page_soup.find_all("li")
    found_agency=False
    #loop over the li tags that we found
    for li in li_tags:
        #find span labels
        label_span = li.find("span", class_="padding-right-2 text-base")
        #see if text has Agency in it
        if label_span and "Agency:" in label_span.text:
            agency_text = label_span.next_sibling
            if agency_text:
                agency_name = agency_text.strip() #remove #whitespace
                agency.append(agency_name)  #append to the list
            else:
                agency.append("NA")  #if agency na
```

```
            found_agency = True
            break  #break if we find the agency

    if not found_agency:
        #append "NA" if we don't find an agency
        agency.append("NA")


df["agency"]=agency

print(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

                                           Hyperlink  \
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)
  - writing a prompt using the input() method as the first step in function
  - that value is then converted to a datetime object and compared to the 2013 restriction listed. If/else statements should cover for this need
  - with that input datetime, we can use it as the key value in a for loop that covers every day between that date until today() as the limit. This loop will process (ascending) every full day (yyyy-mm-dd)
  - 1 second time delay (to avoid being blocked by the site) will be implemented using a 1 second delay with time.sleep()
- b. Create Dynamic Scraper (PARTNER 2)

```
import pandas as pd
import requests
import time
from bs4 import BeautifulSoup

def scraper_function():
    #input date step and confirm correct format
    date_input = input("Please enter a year & month (format should be
 ↪  [yyyy-mm]): ")
    try:
        date_obj = pd.to_datetime(date_input)
    except ValueError:
        print("Invalid date format. Please use YYYY-MM format.")
        return  #exit the function if the date format is invalid

    #checking the date is within the given range
    cutoff_date = pd.to_datetime("2013-01-01")
    if date_obj < cutoff_date:
        print("Please enter a date after the cutoff date:",
         ↪  cutoff_date.strftime("%Y-%m"))
        return  #exit if the date is before the cutoff

    #coop over each month from the input date to the current date
    all_data = []  #collect data for all months and pages
    for month_date in pd.date_range(start=date_obj,
     ↪  end=pd.to_datetime("today"), freq='MS'):
```

```
        formatted_date = month_date.strftime("%Y-%m")
        page_number = 1
        print(f"Scraping data for {formatted_date}...")

        while True:
            #define the URL with page number
            scrape_url =
↪   f"https://oig.hhs.gov/fraud/enforcement/?page={page_number}"
            scrape_response = requests.get(scrape_url)

            if scrape_response.status_code != 200:
                print(f"Error fetching page {page_number} for
                 ↪  {formatted_date}. Skipping...")
                break  #exit the loop if the page is not found #or request
                 ↪  fails

            #parse the HTML content with BeautifulSoup
            scrape_soup = BeautifulSoup(scrape_response.text, "html.parser")

            #find all required elements
            scrape_tag_h = scrape_soup.find_all("h2",
↪   class_="usa-card__heading")
            scrape_span_dates = scrape_soup.find_all("span",
↪   class_="text-base-dark padding-right-105")
            scrape_ul_categories = scrape_soup.find_all("ul",
↪   class_="display-inline add-list-reset")

            #break if no data is found on the page
            if not scrape_tag_h:
                break

            #extract text and hyperlinks, ensuring each list #has a matching
             ↪  entry
            for h2, span, ul in zip(scrape_tag_h, scrape_span_dates,
             ↪  scrape_ul_categories):
                #extract the hyperlink and text from h2
                title = h2.find("a").text.strip() if h2.find("a") else "N/A"
                hyperlink = "https://oig.hhs.gov" + h2.find("a")['href'] if
↪   h2.find("a") else "N/A"

                #extract the date
                date_text = span.text.strip() if span else "N/A"
```

```
                latest_date = pd.to_datetime(date_text, errors='coerce')

                #check if the latest date is within range and stop if it's
                 ↪ before the target date
                if latest_date and latest_date < date_obj:
                    print("Reached date outside range.")
                    scrape_df = pd.DataFrame(all_data, columns=["Title of
↪  Enforcement Action", "Date", "Category", "Hyperlink"])
                    scrape_df.to_csv("enforcement_actions_year_month.csv.",
↪  index=False)
                    print("Data has been saved to 'scraped_data.csv'.")
                    return  #stop the function if date is
                    #outside range

                # Extract the category text
                category_text = [li.text.strip() for li in ul.find_all("li")]
↪  if ul else ["N/A"]
                categories = ", ".join(category_text)

                #append to the all_data list as a tuple
                all_data.append((title, date_text, categories, hyperlink))

            #move to the next page and add a delay
            page_number += 1
            time.sleep(0.5)

    #save the data at the end of all months
    scrape_df = pd.DataFrame(all_data, columns=["Title of Enforcement
↪  Action", "Date", "Category", "Hyperlink"])
    scrape_df.to_csv("enforcement_actions_year_month.csv", index=False)
    print("Data has been saved to enforcement_actions_year_month.csv.")

scraper_function()
```

```
scrape_2023=pd.read_csv("enforcement_actions_year_month.csv")
print(scrape_2023.iloc[-1])
print(scrape_2023.shape)
```

```
Title of Enforcement Action    The United States And Tennessee Resolve
Claims...
Date                                                              January 4,
2021
```

```
Category                                                Criminal and Civil
Actions
Hyperlink
https://oig.hhs.gov/fraud/enforcement/the-unit...
Name: 3021, dtype: object
(3022, 4)
```

The earliest enforcement action is on January 3rd, 2023. Its enforcement title is Podiatrist Pays $90,000 To Settle False Billing Allegations, and its url is https://oig.hhs.gov/fraud/enforcement/podiatrist-pays-90000-to-settle-false-billing-allegations/. It was brought by the US Attorney's Office, Southern District of Texas. There are 1,534 actions we collected into the scraped dataframe.
* c. Test Partner's Code (PARTNER 1)

```
scraper_function()
```

```
scraped_data_df=pd.read_csv("enforcement_actions_year_month.csv")
print(scraped_data_df.iloc[-1])
print(scraped_data_df.shape)
```

```
Title of Enforcement Action    The United States And Tennessee Resolve
Claims...
Date                                                          January 4,
2021
Category                                                Criminal and Civil
Actions
Hyperlink
https://oig.hhs.gov/fraud/enforcement/the-unit...
Name: 3021, dtype: object
(3022, 4)
```

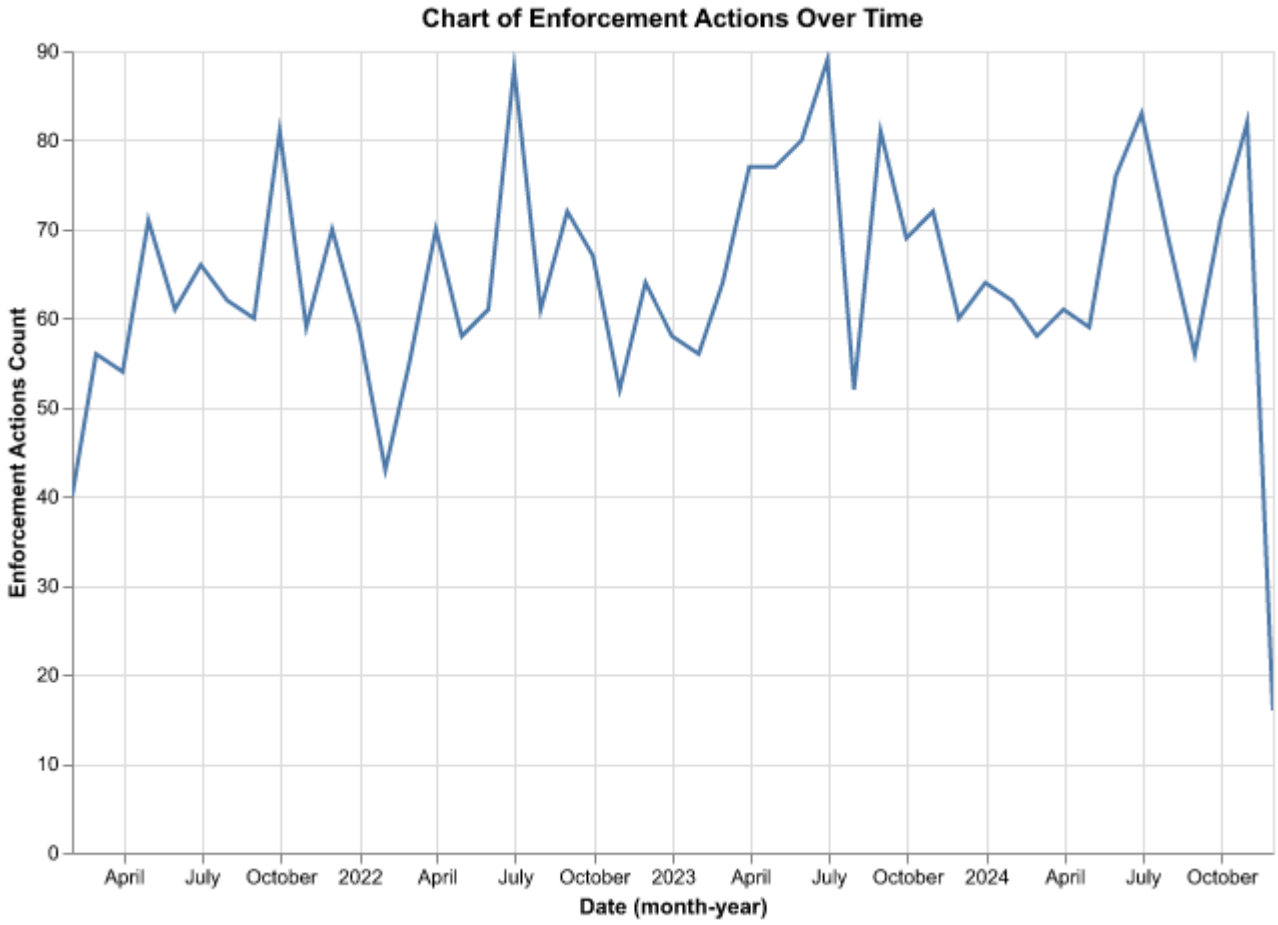
We have 3,022 actions in our final df. Note that this can change by the time we submit the assignment, as the site is updated regularly and I am writing this answer on 11/8/24.

Date of earliest scraped enforcement action: Jan 4th, 2021. The title of the action 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. It was in the Criminal and Civil Actions Category. The URL is https://oig.hhs.gov/fraud/enforcement/the-united-states-and-tennessee-resolve-claims-with-three-providers-for-false-claims-act-liability-relating-to-p-stim-devices-for-a-total-of-172-million/

Step 3: Plot data based on scraped data

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

```
import pandas as pd
import altair as alt

# filtering towards the prompt's specified date range
clipping_start_date = pd.to_datetime("2021-01-01")
clipping_end_date = pd.to_datetime("today")
scraped_data_df["Date"] = pd.to_datetime(scraped_data_df["Date"])

plotting_enforcement_df = scraped_data_df[(scraped_data_df["Date"] >=
 ↪  clipping_start_date) & (scraped_data_df["Date"] <= clipping_end_date)]

# aggregating over month & year
grouped_enforcement = plotting_enforcement_df.groupby(pd.Grouper(key= "Date",
 ↪  freq= "M")).size().reset_index(name= "Count")

# line chart plotting
enforc_chart = alt.Chart(grouped_enforcement).mark_line().encode(
    x= alt.X("Date:T", title= "Date (month-year)"),
    y= alt.Y("Count", title= "Enforcement Actions Count"),
).properties(
    title= "Chart of Enforcement Actions Over Time",
    width= 600,
    height= 400,
).interactive()

enforc_chart.show()
```

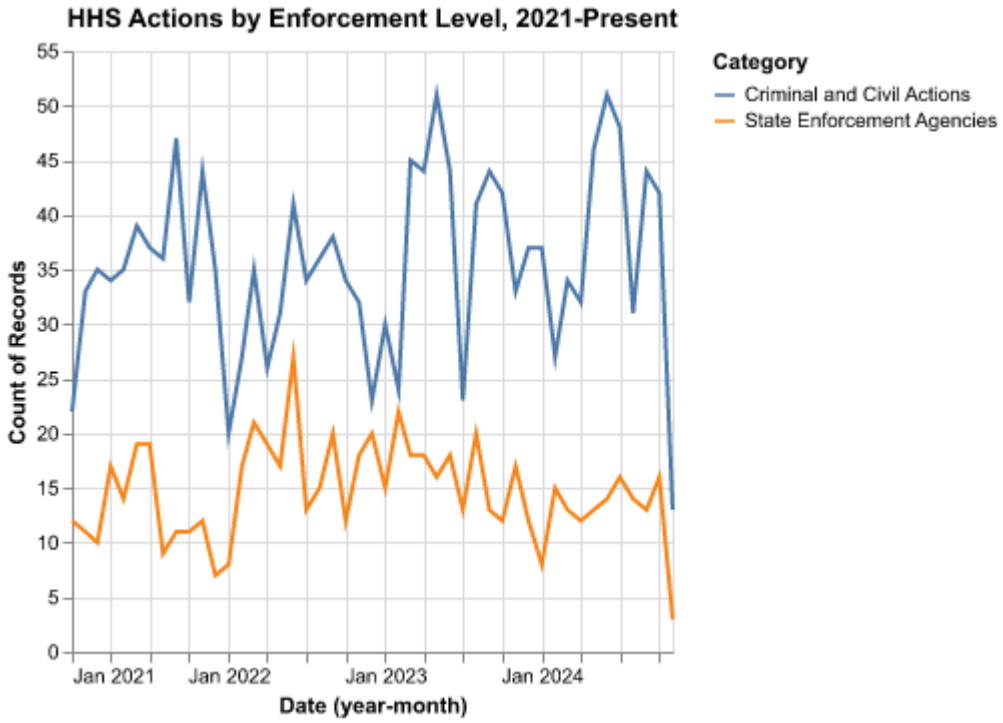


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

```
import altair as alt

#filter out anything other than state or crim and civil from
#our category variable. Then color based upon the two
#dif categories:
state_v_crim_plot=alt.Chart(scraped_data_df).transform_filter(
        (
            alt.datum.Category == "State Enforcement Agencies") |
        (
            alt.datum.Category == "Criminal and Civil Actions")
            ).mark_line().encode(
```

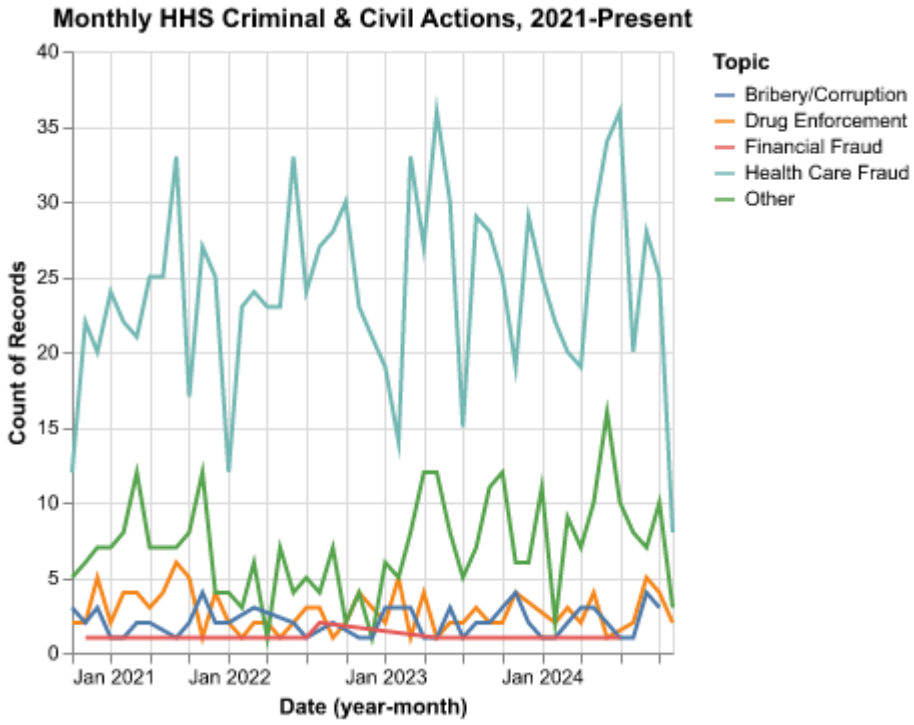
```
  alt.X("yearmonth(Date):T"),
  alt.Y("count()"),
  color=("Category")
).properties(
    title="HHS Actions by Enforcement Level, 2021-Present")
state_v_crim_plot.show()
```



- based on five topics

```
import numpy as np
#filter
crim_civil_df=scraped_data_df[scraped_data_df["Category"]=="Criminal and
 ↪  Civil Actions"]

#Define a function to determine the topic based on keywords
def assign_topic(title):
    title = title.lower()
    if "healthcare" in title or "health care" in title or "medic" in title or
     ↪  "claims" in title:
        return "Health Care Fraud"
```

```
    elif "bank" in title or "financial" in title:
        return "Financial Fraud"
    elif "drug" in title or "distribut" in title:
        return "Drug Enforcement"
    elif "bribe" in title or "corruption" in title or "kickback" in title:
      return "Bribery/Corruption"
    else:
        return "Other"

#apply the function to the df's enfrocement action, and assign
#the result to a new column
crim_civil_df["Topic"] = crim_civil_df["Title of Enforcement
 ↪  Action"].apply(assign_topic)

#plot with the topic being the color of the line
topic_chart=alt.Chart(crim_civil_df).mark_line().encode(
  alt.X("yearmonth(Date):T"),
  alt.Y("count()"),
  color="Topic"
).properties(title="Monthly HHS Criminal & Civil Actions, 2021-Present")
topic_chart.show()
```



Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
#apply the for loop to extract each rows agency:
import requests
from bs4 import BeautifulSoup
import time

agency=[]


for url in scraped_data_df["Hyperlink"]:
    try:
        #get start time. ultimately will use for sanity check
        #while it is running.
        request_start_time = time.time()

        #get request for each url that is present in hyperlink
```

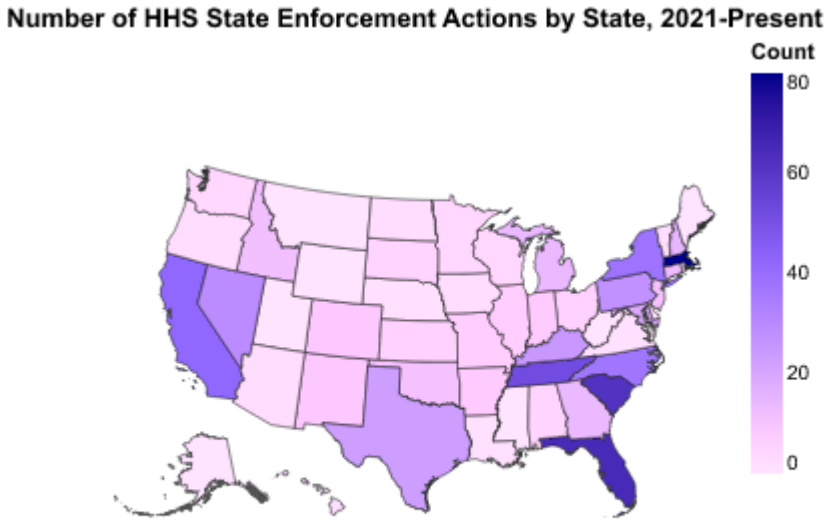
```
        #column.
        response = requests.get(url)
        #end time to see how long it takes to get the url
        request_end_time = time.time()
        print(f"Time taken for request to {url}: {request_end_time -
         ↪  request_start_time} seconds")

        #parse the HTML content
        page_soup = BeautifulSoup(response.text, "html.parser")

        #find ul tags that are relevant
        ul_tags = page_soup.find_all("ul", class_="usa-list
↪  usa-list--unstyled margin-y-2")
        #set a variabl that can tell us if we found an agency
        #the default will be false.
        found_agency = False
        for ul in ul_tags:
            #loop over all the li tags in the ul tags
            li_tags = ul.find_all("li")

            for li in li_tags:
                #measure how long its taking to find the li tags
                li_start_time = time.time()

                #find the span with the Agency: label
                label_span = li.find("span", class_="padding-right-2
↪  text-base")

                if label_span and "Agency:" in label_span.text:
                    agency_text = label_span.next_sibling
                    if agency_text:
                        agency_name = agency_text.strip()  #remove whitespace
                        agency.append(agency_name)
                        #append to the list
                    else:
                        agency.append("NA")
                        #append "NA" if agency name is missing.
                        #change the found agency value to true
                        #to exit the loop
                    found_agency = True
                    break  #stop checking other li tags if agency is found
```

```
                li_end_time = time.time()
                print(f"Time taken to check an `li` tag: {li_end_time -
                 ↪  li_start_time} seconds")

            if found_agency:
                break
            #exit the outer loop once the agency is found

        if not found_agency:
            #append "NA" if we dont find an agency
            agency.append("NA")

        time.sleep(.5)

    except requests.RequestException as e:
        #handle request-related errors
        print(f"Error fetching {url}: {e}")
        agency.append("NA")

#add the agency information to the df
scraped_data_df["agency"] = agency
```

```
import geopandas as gpd
import numpy as np
import pandas as pd
import altair as alt
scraped_data_df=pd.read_csv(
    r"C:\\Users\\Mitch\Documents\\enforcement_actions_year_month.csv")
#I had adjustments I had to make in our earlier code. For timely #submission
 ↪  I'm pulling this saved csv so I dont have to run
#a 50 minute loop to get our df as it is needed again.
state_geo_data= gpd.read_file("cb_2018_us_state_500k.shp")

state_actions_df=scraped_data_df[scraped_data_df["Category"]=="State
 ↪  Enforcement Agencies"]

#clean the agency column so we can extract the state. remove
#any of the strings that aren't a state:
state_actions_df["state"] = np.where(
    state_actions_df["agency"].notnull(),
    state_actions_df["agency"]
```

```
    .str.replace(r"(State of |State
↪   |General|Genera|Inspector|U\.S\.|Attorney|Northern|Eastern|Western|Southern|Office,|Depa
↪   "", regex=True)
    .str.strip(),  #remove any leading or trailing whitespace
    np.nan  #assign NaN if the agency column is empty
)

#change a few names since there's some cleaning still to be done
#and the format should match the geo df for the merge
state_actions_df["state"] = np.where(
    state_actions_df["state"].str.contains("California", case=False,
↪   na=False),
    "California",
    state_actions_df["state"]
)

state_actions_df["state"] = np.where(
    state_actions_df["state"].str.contains("Hawai'i", case=False, na=False),
    "Hawaii",
    state_actions_df["state"]
)

state_actions_df["state"] = np.where(
    state_actions_df["state"].str.contains("Virgin Islands", case=False,
↪   na=False),
    "United States Virgin Islands",
    state_actions_df["state"]
)

#now groupby and give us a count for each state in our data
state_actions_totals_df=state_actions_df.groupby("state").size().reset_index(name="Count")
#rename the column in geo df to match our state actions df
state_geo_data = state_geo_data.rename(columns={"NAME": "state"})
#merge df with gdf
state_geo_data_totals = state_geo_data.merge(
    state_actions_totals_df, on="state", how="left")

#fill any states who had no obs in our original df with 0s
state_geo_data_totals["Count"] = state_geo_data_totals["Count"].fillna(0)

#plot map, making sure to provide borders to each state,
#and changing the color scheme so its more clear which states
```

```
#have high values and which have low.
alt.Chart(state_geo_data_totals).mark_geoshape(
    stroke="black", strokeWidth=.5).encode(
    alt.X("geometry"),
    color=alt.Color("Count",scale=alt.Scale(range=["white","darkblue"]))
).project("albersUsa"
).properties(title="Number of HHS State Enforcement Actions by State,
 ↪  2021-Present")
```



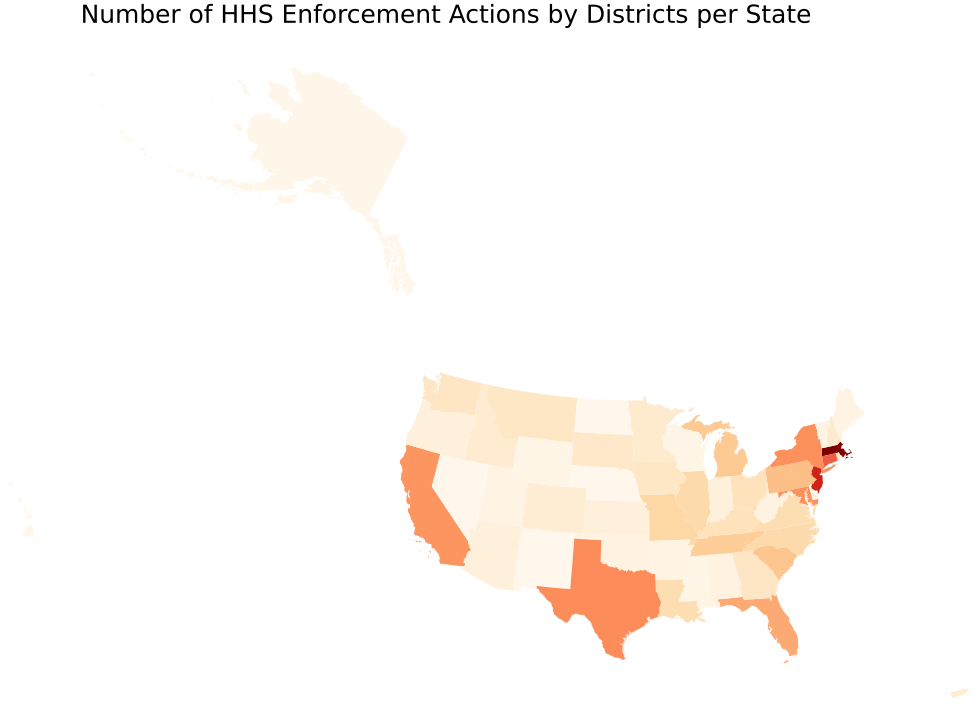
2. Map by District (PARTNER 2)

```
import pandas as pd
import re
import geopandas as gpd
import matplotlib.pyplot as plt

# importing our state shapefile
us_states = gpd.read_file("cb_2018_us_state_500k.shp")

# processing our agency column into districts (and states) using a Regular
 ↪  Expressions function
```

```
# got the idea from stack overflow -
 ↪  https://stackoverflow.com/questions/55194224/extract-names-from-string-with-python-regex

district_df = scraped_data_df.copy()
district_df["agency"] = district_df["agency"].astype(str)


# empty columns to be filled
district_df["State"]= ""
district_df["District"] = ""

# regular expression function
def state_district_extractor(string):
  pattern = r"U\.S\. Attorney's Office(?:,
 ↪  (Eastern|Western|Northern|Middle|Central|Southern|District of) )?(.*)"

  match = re.match(pattern, string)
  # distict name case
  if match:
    district = match.group(1) + " " + match.group(2) if match.group(1) else
 ↪  match.group(2)
    state = match.group(2)
    return state, district

  # "state of" name case
  elif "State of" in string or not string.split():
    state = string.split("State of ")[1].strip()
    return state, None

  # just state name case
  elif string.strip():
    state = string.strip()
    return state, None

  # multi-agency cases
  elif "U.S. Department of Justice" in string:
    state = string.split("District of ")[-1].strip()
    return state, "N/A"

  elif "U.S. Attorney's Office" in string:
      remaining_part = string.split("U.S. Attorney's Office, ")[1].strip()
      parts = remaining_part.split("District of ")
```

```
      state = parts[-1].strip()
      district = parts[0].strip()
      return state, district

  # empty cases
  return "N/A", "N/A"

district_df[["State","District"]] =
 ↪  district_df["agency"].apply(state_district_extractor).apply(pd.Series)

# Correcting for Washington, DC and other exceptions
district_df.loc[district_df["State"] == "Columbia", "State"] = "Washington,
 ↪  D.C."
district_df.loc[district_df["State"] == "Columbia", "State"] = "Washington,
 ↪  D.C."

district_df.loc[district_df["District"].isnull() &
 ↪  district_df["agency"].str.contains("U.S. Attorney's Office"), "District"]
 ↪  = district_df["agency"].str.replace("U.S. Attorney's Office, ", '',
 ↪  regex=False)
# cleanup in case the terminology makes it through
district_df["State"] = district_df["State"].str.replace("District of ", "",
 ↪  regex= False)
district_df["State"] = district_df["State"].str.replace("U.S. Attorney's
 ↪  Office, ", "", regex= False)
district_df["State"] = district_df["State"].str.replace("Attorney General",
 ↪  "", regex= False)
district_df["State"] = district_df["State"].str.replace("U.S. Attorney
 ↪  General", "", regex= False)
district_df["State"] = district_df["State"].str.replace("U.S. Department of
 ↪  Justice and U.S. Attorney's Office, ", "", regex= False)
# persistant cases of the direction -- specific exceptions for states with
 ↪  directional names
district_df["State"] = district_df["State"].str.replace("Northern", "",
 ↪  regex= False)
district_df["State"] = district_df["State"].str.replace("Eastern", "", regex=
 ↪  False)
district_df["State"] = district_df["State"].str.replace("Southern", "",
 ↪  regex= False)
district_df["State"] = district_df["State"].str.replace("Western", "", regex=
 ↪  False)
district_df["State"] = district_df["State"].str.replace("Middle", "", regex=
 ↪  False)
```

```
district_df["State"] = district_df["State"].str.replace("Central", "", regex=
 ↪  False)

# choropleth mapping by district
district_grouping = district_df.groupby("State").size().reset_index(name=
 ↪  "District_Count")
merged_df = us_states.merge(district_grouping, left_on= "NAME", right_on=
 ↪  "State", how= "left")
# per the recommendation in Ed, I will use the Albers USA projection
merged_df = merged_df.to_crs("ESRI:102003")

# plotting
merged_df.plot(column= "District_Count", cmap= "OrRd", figsize=(10, 8))
plt.title("Number of HHS Enforcement Actions by Districts per State")
plt.axis("off")
plt.show()
```



Extra Credit

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

2. **Conduct spatial join**

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