Problem Set 4

Brandon Texeira

Invalid Date

Due 02/07 at 5:00PM Central.

"This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: BHT

Github Classroom Assignment Setup and Submission Instructions

1. **Accepting and Setting up the PS4 Assignment Repository**
    - Each student must individually accept the repository for the problem set from Github Classroom ("ps4") – https://classroom.github.com/a/hWhtcHqH
        - You will be prompted to select your cnetid from the list in order to link your Github account to your cnetid.
        - If you can't find your cnetid in the link above, click "continue to next step" and accept the assignment, then add your name, cnetid, and Github account to this Google Sheet and we will manually link it: https://rb.gy/9u7fb6
    - If you authenticated and linked your Github account to your device, you should be able to clone your PS4 assignment repository locally.
    - Contents of PS4 assignment repository:
        - `ps4_template.qmd`: this is the Quarto file with the template for the problem set. You will write your answers to the problem set here.
2. **Submission Process**:
    - Knit your completed solution `ps4.qmd` as a pdf `ps4.pdf`.
        - Your submission does not need runnable code. Instead, you will tell us either what code you ran or what output you got.
    - To submit, push `ps4.qmd` and `ps4.pdf` to your PS4 assignment repository. Confirm on Github.com that your work was successfully pushed.

Grading

- You will be graded on what was last pushed to your PS4 assignment repository before the assignment deadline
- Problem sets will be graded for completion as: {missing (0%); - (incomplete, 50%); + (excellent, 100%)}
  - The percent values assigned to each problem denote how long we estimate the problem will take as a share of total time spent on the problem set, not the points they are associated with.
- In order for your submission to be considered complete, you need to push both your `ps4.qmd` and `ps4.pdf` to your repository. Submissions that do not include both files will automatically receive 50% credit.

```python
import pandas as pd
import altair as alt
import time
from datetime import datetime
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin

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

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

Step 1: Develop initial scraper and crawler

```python
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
soup.text[0:50]
test = soup.find("main")
len(test)
test2 = test.find("section", id="results")
test3 = test2.find_all("li")
test3[0]
items = soup.find_all(
    "li",
    class_="usa-card card--list pep-card--minimal mobile:grid-col-12"
)
# h2 is the Title of Enforcement Action
# span is date under div class=font-body-sm...
# li class=display-inline... is category
# a href under h2 is link

base_url = "https://oig.hhs.gov"

data = []
for item in items:
    # Title + link
    a_tag = item.find("h2", class_="usa-card__heading").find("a")
    title = a_tag.get_text(strip=True)
    link = urljoin(base_url, a_tag["href"])

    # Date
    date = item.find("span", class_="text-base-dark").get_text(strip=True)

    # Category
    category = item.find("li", class_="usa-tag").get_text(strip=True)

    data.append({
        "Title of Enforcement Action": title,
        "Date": date,
        "Category": category,
        "Link": link
    })
```

```
df_enforcement = pd.DataFrame(data)
df_enforcement.head()
```

```
   Title of Enforcement Action                        Date              Category                     Li
0  Houston Transplant Doctor Indicted For Making ...  February 5, 2026  Criminal and Civil Actions    htt
1  MultiCare Health System to Pay Millions to Set...  February 4, 2026  Criminal and Civil Actions    htt
2  Brooklyn Banker Pleads Guilty to Laundering Pr...  February 3, 2026  COVID-19                     htt
3  Delafield Man Sentenced to 18 Months' Imprison...  February 3, 2026  Criminal and Civil Actions    htt
4  Former NFL Player Convicted for $197M Medicare...  February 3, 2026  Criminal and Civil Actions    htt
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code I want the scraper to go through the html code on the enforcement actions page to pull the name of the enforcement action, the date, the category of the enforcement action, and the hyperlink to that enforcement action. I want to be able to set the start time and include an on/off switch. The following is what I expect the code to be able to do.

Start of function: scrape_enforcement_actions(start_year, start_month, run_scraper)

```
If run_scraper is set to false
    PRINT "Scraper is turned off"
    Stop running
```

```
If start_year < 2013
    Print "Year must be 2013 or later"
    Stop running
Set start_date = first day of the start year and start month
Create an empty list to store data
Set page = 0

If more pages exist
    Add beginning of url to scraped url
    Download page HTML
    Parse HTML
    Find all enforcement action items on page
    If no items found
        Stop loop

    For each enforcement action item
        Scrape title, link, date, and category
        Convert date to date format for pandas
        If date is invalid
            Skip item # my first attempt ran for a very long time due to an
            improper date format. I had to use ChatGPT to debug this in
            addition to converting some trickier parts of this pseudocode to
            real code
        If date is earlier than start_date
            Save data to CSV file
            Return data and stop loop
        Add enforcement action data to data list

    Wait 1 second
    Continue to next page
```

End function

- b. Create Dynamic Scraper

```python
def scrape_enforcement_actions(start_year, start_month, run_scraper=False):
    if not run_scraper:
        print("Scraper is turned off. Set run_scraper=True to run.")
        return None
    if start_year < 2013:
        print("Please restrict scraping to year >= 2013. No data available
         ↪  before 2013.")
        return None
    base_url = "https://oig.hhs.gov"
    headers = {"User-Agent": "Mozilla/5.0"}
    start_date = datetime(start_year, start_month, 1)
    today = datetime.today()

    all_data = []
    page = 0
    while True:
        url = f"https://oig.hhs.gov/fraud/enforcement?page={page}"
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, "html.parser")
        items = soup.find_all(
            "li",
            class_="usa-card card--list pep-card--minimal mobile:grid-col-12"
        )

        if not items:
            break
        for item in items:
            a_tag = item.select_one("h2.usa-card__heading a")
            date_tag = item.select_one("span.text-base-dark")
            category_tag = item.select_one("li.usa-tag")
            if not (a_tag and date_tag):
                continue
            title = a_tag.get_text(strip=True)
            link = urljoin(base_url, a_tag["href"])
            date_str = date_tag.get_text(strip=True)
            category = category_tag.get_text(strip=True) if category_tag else
↪   None

            date = pd.to_datetime(date_str, errors="coerce")

            if pd.isna(date):
                continue

            if date < start_date:
                df = pd.DataFrame(all_data)
                filename =
↪   f"enforcement_actions_{start_year}_{start_month}.csv"
                df.to_csv(filename, index=False)
                print(f"Saved {len(df)} records to {filename}")
                return df
            all_data.append({
                "Date": date,
                "Title of Enforcement Action": title,
                "Category": category,
                "Link": link
            })

        page += 1
        time.sleep(1)
```

```
RUN_SCRAPER = False #changed to false after successfully scraping data

scrape_enforcement_actions(
    start_year=2024,
    start_month=1,
    run_scraper=RUN_SCRAPER
)
```

```
Scraper is turned off. Set run_scraper=True to run.
```

The earliest enforcement action is from January 3rd, 2024. The enforcement action is Former Nurse Aide Indicted In Death Of Clarksville Patient Arrested In Georgia, and the category is State Enforcement Agencies. There are 1807 Enforcement Actions scraped.

- c. Test Your Code

```
RUN_SCRAPER = False #changed to false after successfully scraping data

scrape_enforcement_actions(
    start_year=2022,
    start_month=1,
    run_scraper=RUN_SCRAPER
)
```

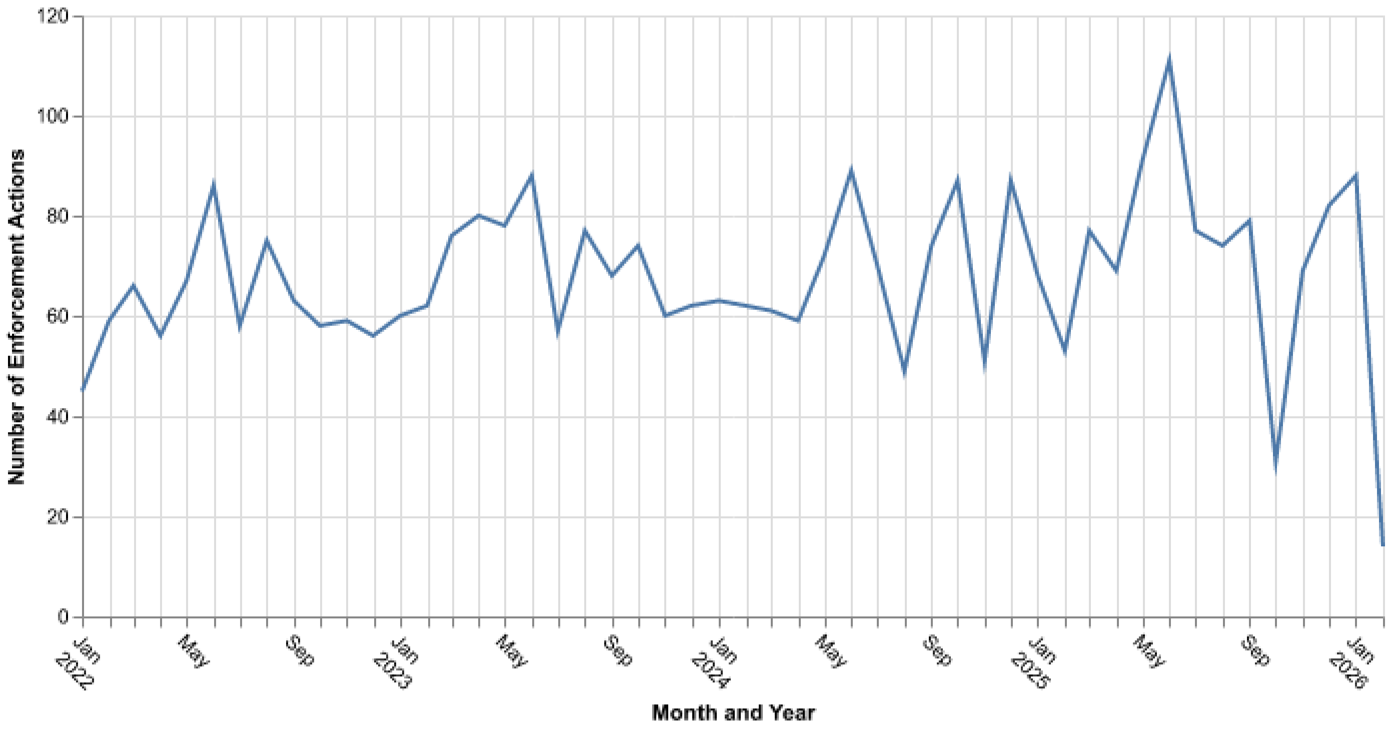
```
Scraper is turned off. Set run_scraper=True to run.
```

There are 3397 enforcement actions pulled. The earliest is from Janaury 4th, 2022 and is Integrated Pain Management Medical Group Agreed to Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded Individuals.

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

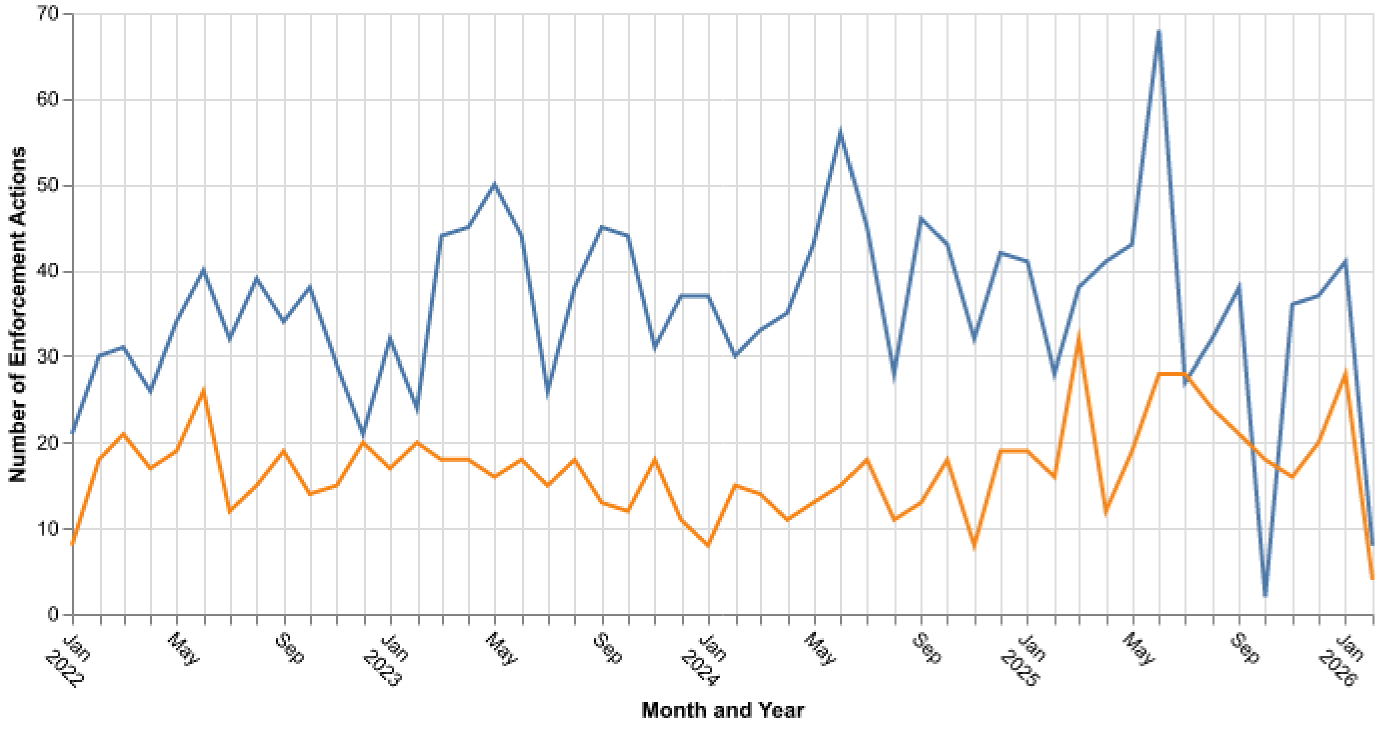
```
df = pd.read_csv("enforcement_actions_2022_1.csv")
df_enforce = pd.DataFrame(df)
chart1 = alt.Chart(df_enforce).mark_line().encode(
    x = alt.X("yearmonth(Date):T", title="Month and
 ↪  Year").axis(tickCount="month", labelAngle=45,
 ↪  labelExpr="[timeFormat(datum.value, '%b'), timeFormat(datum.value, '%m')
 ↪  == '01' ? timeFormat(datum.value, '%Y') : '']"),
    y = alt.Y("count():Q", title = "Number of Enforcement Actions")
).properties(width = 650)
chart1.display()
```



2. Plot the number of enforcement actions categorized:

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

```
df_enforce_subset = df_enforce[df_enforce["Category"].isin(["Criminal and
↪  Civil Actions", "State Enforcement Agencies"])]
chart2 = alt.Chart(df_enforce_subset).mark_line().encode(
    x = alt.X("yearmonth(Date):T", title="Month and
↪   Year").axis(tickCount="month", labelAngle=45,
↪   labelExpr="[timeFormat(datum.value, '%b'), timeFormat(datum.value, '%m')
↪   == '01' ? timeFormat(datum.value, '%Y') : '']"),
    y = alt.Y("count():Q", title = "Number of Enforcement Actions"),
↪   color=alt.Color("Category:N", title="Category")
).properties(width = 650)
chart2.display()
```



- based on five topics