PSET04

Fatima Hussain

2026-02-06

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: **FH**

`I used AI (Claude) to help me understand the HTML structure of the HHS OIG enforcement actions page - specifically identifying the correct CSS classes to target when writing my BeautifulSoup selectors (e.g. usa-card, usa-card__heading, text-base-dark). I also used it to help brainstorm keyword lists for classifying the five topics in Step 3, since manually going through thousands of titles to find patterns would have been tedious.`

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.

```
import pandas as pd
import altair as alt
import time
import requests
from bs4 import BeautifulSoup
from datetime import datetime

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

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

Step 1: Develop initial scraper and crawler

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

cards = soup.find_all('li', class_='usa-card')

titles = []
dates = []
categories = []
links = []

for card in cards:
    heading = card.find('h2', class_='usa-card__heading')
    if heading:
        a_tag = heading.find('a')
        title = a_tag.get_text(strip=True)
        link = "https://oig.hhs.gov" + a_tag['href']

        date_span = card.find('span', class_='text-base-dark')
        date_str = date_span.get_text(strip=True) if date_span else None

        tag_li = card.find('li', class_='display-inline-block')
        category = tag_li.get_text(strip=True) if tag_li else None

        titles.append(title)
        dates.append(date_str)
```

```
        categories.append(category)
        links.append(link)

enforcement_actions = pd.DataFrame({
  'title': titles,
  'date': dates,
  'category': categories,
  'link': links
})

enforcement_actions.head()
```

	title	date	category	lin
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

```
FUNCTION scrape_enforcement_actions(start_month, start_year):
    1. INPUT VALIDATION: Check if start_year >= 2013
       - If not, print restrict to year 2013

    2. CREATE start_date from start_month and start_year

    3. INITIALIZE empty lists for titles, dates, categories, links

    4. SET page = 1 and keep_going = True

    5. WHILE keep_going is True:
       a. BUILD URL: "https://oig.hhs.gov/fraud/enforcement/?page={page}"
       b. FETCH the page HTML using requests.get()
       c. PARSE the HTML with BeautifulSoup
       d. FIND all <li> elements with class 'usa-card'
```

```
    e. IF no cards found, BREAK (we've gone past the last page)
    f. FOR each card:
       - EXTRACT title from <h2 class="usa-card__heading"> > <a> tag
       - EXTRACT link from the <a> tag's href attribute
       - EXTRACT date from <span class="text-base-dark">
       - EXTRACT category from <li class="display-inline-block">
       - PARSE the date string into a datetime object
       - IF the parsed date < start_date:
           SET keep_going = False and BREAK out of the for loop
         ELSE:
           APPEND title, date, category, link to respective lists
    g. INCREMENT page by 1
    h. SLEEP for 1 second

6. CREATE a pandas DataFrame from the collected lists
7. SAVE to CSV file named "enforcement_actions_year_month.csv"
8. RETURN the DataFrame
```

I use a *while loop* instead of for loop because we do not know how many pages to scrape. The page listing is sorted by date, so we keep incrementing the page number until we encounter a date earlier than our start date. A while loop with a boolean flag (`keep_going`) is ideal because the stopping condition depends on the content of the data, not a predetermined count.

- b. Create Dynamic Scraper

```python
# to run the function
RUN_SCRAPER = False

def scrape_enforcement_actions(start_month, start_year):
    if start_year < 2013:
        print("Restrict to year >= 2013, since only enforcement actions after
         ↪  2013 are listed.")
        return None

    start_date = datetime(start_year, start_month, 1)

    all_titles = []
    all_dates = []
    all_categories = []
    all_links = []

    page = 1
    keep_going = True
```

```
while keep_going:
    url = f"https://oig.hhs.gov/fraud/enforcement/?page={page}"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    cards = soup.find_all('li', class_='usa-card')

    if len(cards) == 0:
        break

    for card in cards:
        heading = card.find('h2', class_='usa-card__heading')
        if heading:
            a_tag = heading.find('a')
            title = a_tag.get_text(strip=True)
            link = "https://oig.hhs.gov" + a_tag['href']

            date_span = card.find('span', class_='text-base-dark')
            date_str = (date_span.get_text(strip=True)
                        if date_span else None)

            tag_li = card.find('li', class_='display-inline-block')
            category = (tag_li.get_text(strip=True)
                        if tag_li else None)

            if date_str:
                action_date = datetime.strptime(date_str, "%B %d, %Y")
                if action_date < start_date:
                    keep_going = False
                    break

            all_titles.append(title)
            all_dates.append(date_str)
            all_categories.append(category)
            all_links.append(link)

    page += 1
    time.sleep(1)

enforcement_data = pd.DataFrame({
  'title': all_titles,
  'date': all_dates,
```

```
      'category': all_categories,
      'link': all_links
    })

    filename = f"enforcement_actions_{start_year}_{start_month}.csv"
    enforcement_data.to_csv(filename, index=False)
    print(f"Saved {len(enforcement_data)} actions to {filename}")

    return enforcement_data

# January 2024
if RUN_SCRAPER:
    data2024 = scrape_enforcement_actions(1, 2024)
    print(f"\nTotal enforcement actions since Jan 2024: {len(data2024)}")
    print(f"\nEarliest action scraped:")
    print(data2024.tail(1))
```

- c. Test Your Code

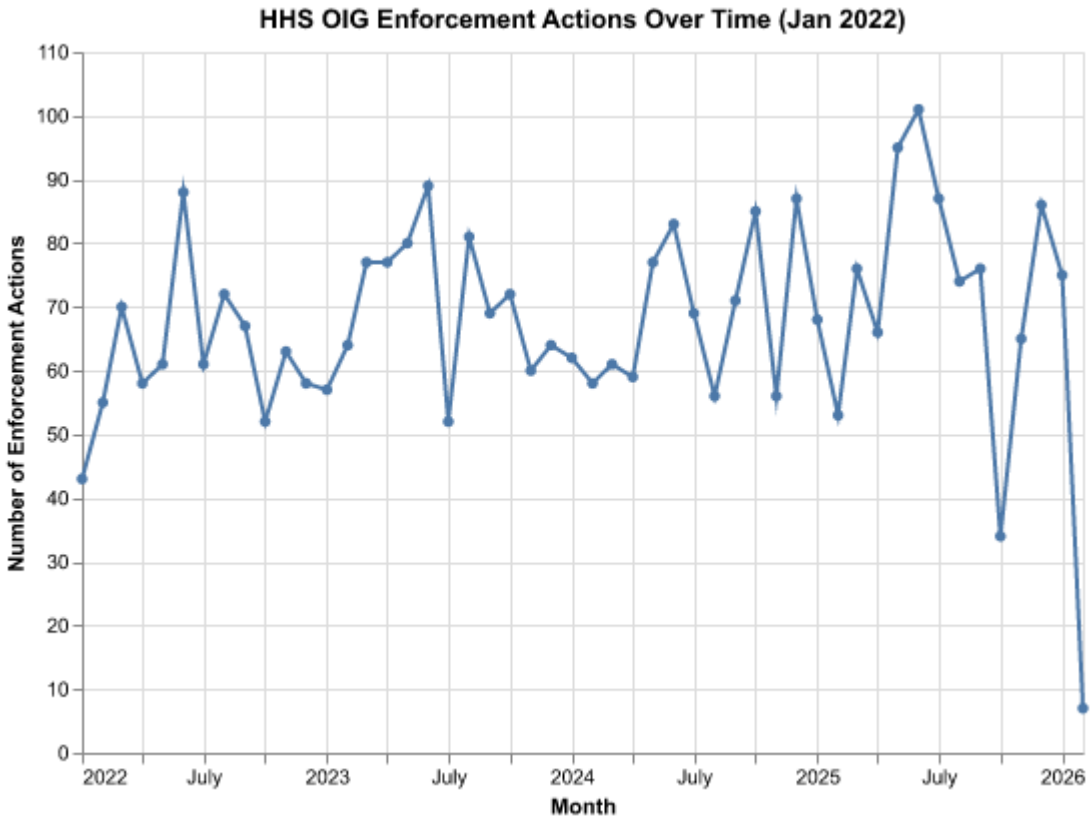
```
# January 2022
if RUN_SCRAPER:
    data2022 = scrape_enforcement_actions(1, 2022)
    print(f"\nTotal enforcement actions since Jan 2022: {len(data2022)}")
    print(f"\nEarliest action scraped:")
    print(data2022.tail(1))
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
jan_2022 = pd.read_csv("enforcement_actions_2022_1.csv")
jan_2022['date'] = pd.to_datetime(jan_2022['date'], format='%B %d, %Y')
jan_2022['year_month'] = jan_2022['date'].dt.to_period('M').dt.to_timestamp()

monthly_counts = (jan_2022.groupby('year_month')
                  .size()
                  .reset_index(name='count'))

chart1 = (alt.Chart(monthly_counts).mark_line(point=True).encode(
  x=alt.X('year_month:T', title='Month'),
```

```
  y=alt.Y('count:Q', title='Number of Enforcement Actions')
  ).properties(
    title='HHS OIG Enforcement Actions Over Time (Jan 2022)',
    width=500, height=350
  )
)

chart1
```

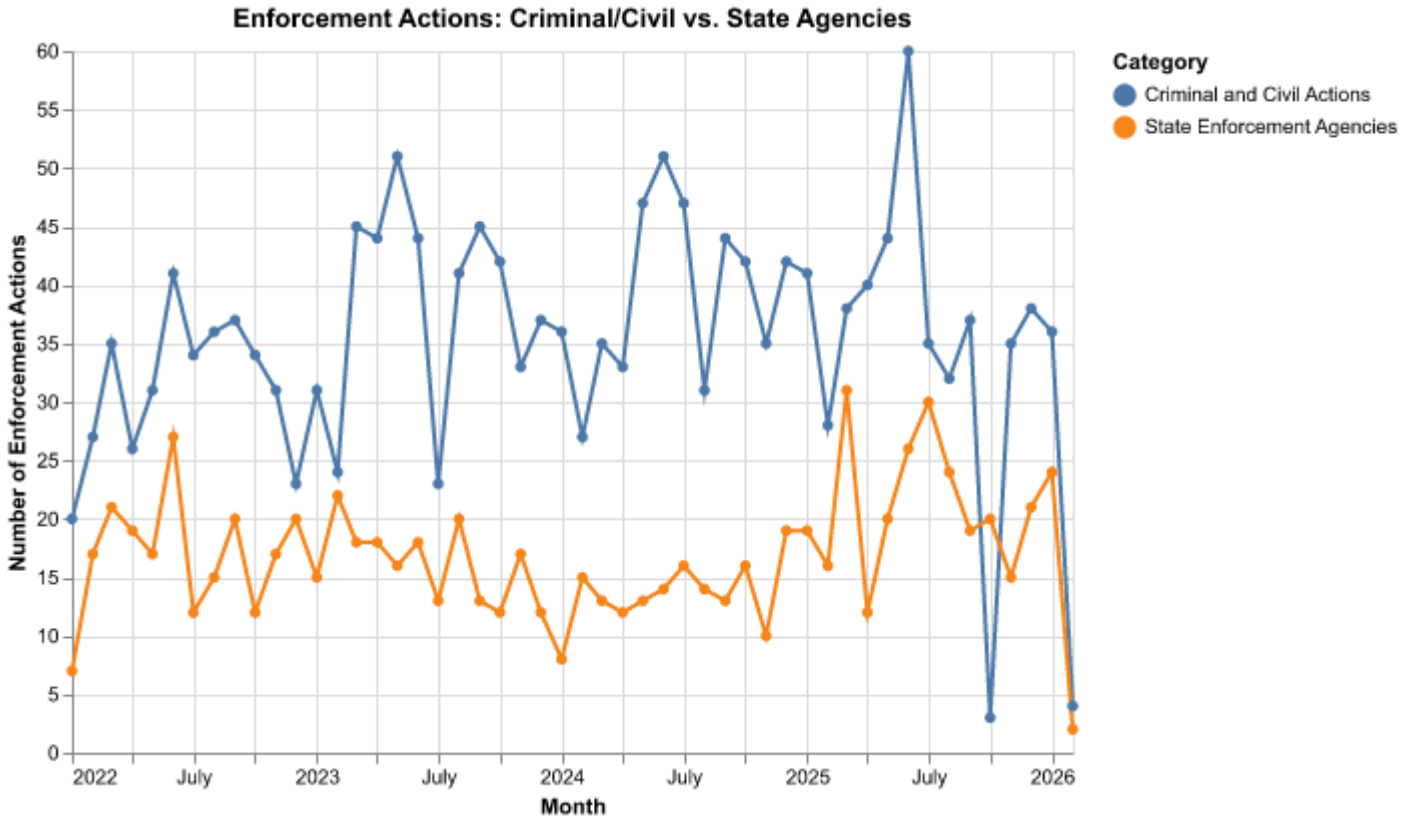


2. Plot the number of enforcement actions categorized:

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

```
# Filter to just the two main categories
df_two_cat = jan_2022[jan_2022['category'].isin(
  ['Criminal and Civil Actions', 'State Enforcement Agencies']
)]
```

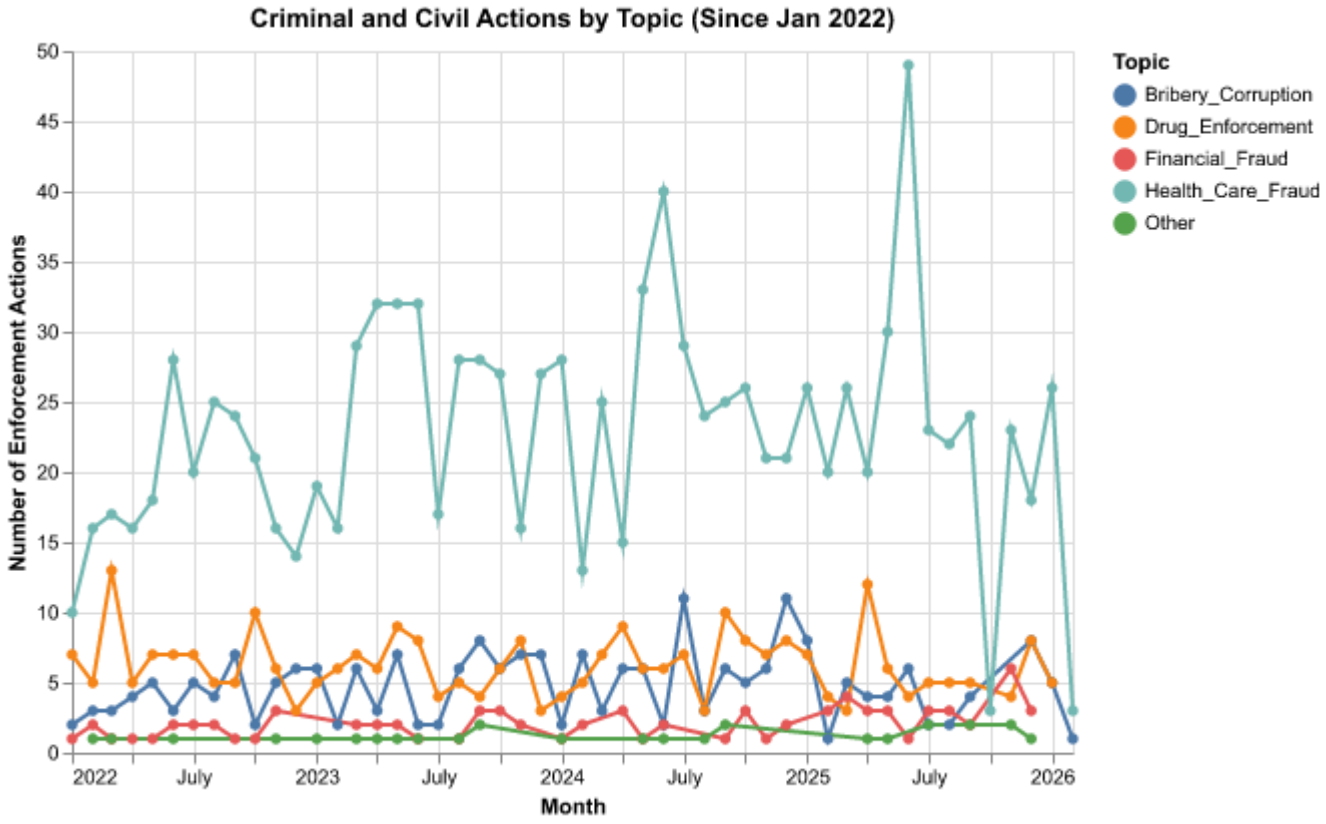
```
monthly_by_2_cat = (df_two_cat.groupby(['year_month', 'category'])
                   .size()
                   .reset_index(name='count'))

chart2 = (alt.Chart(monthly_by_2_cat).mark_line(point=True).encode(
  x=alt.X('year_month:T', title='Month'),
  y=alt.Y('count:Q', title='Number of Enforcement Actions'),
  color=alt.Color('category:N', title='Category')
  ).properties(
    title='Enforcement Actions: Criminal/Civil vs. State Agencies',
    width=500, height=350
    )
)

chart2
```



- based on five topics

```
criminal_civil = jan_2022[jan_2022['category'] == 'Criminal and Civil
↪  Actions'].copy()

def classify_topic(title):
    title_lower = title.lower()

    # Drug Enforcement
    drug_keywords = ['drug', 'opioid', 'fentanyl', 'narcotic', 'controlled
↪  substance',
                     'prescription', 'pill mill', 'distributing medication',
                     'illegal distribution', 'dea', 'methamphetamine',
                      ↪  'cocaine',
                     'heroin', 'morphine', 'adderall', 'oxycodone',
                      ↪  'hydrocodone',
                     'pharmaceutical diversion', 'drug diversion',
                      ↪  'substance']

    # Bribery/Corruption
    bribery_keywords = ['brib', 'corrupt', 'embezzl', 'kickback', 'extort',
                        'gratuity', 'influence', 'public official', 'gift',
                        'anti-kickback']

    # Financial Fraud
    financial_keywords = ['bank', 'financial', 'money launder', 'tax
↪  evasion',
                          'tax fraud', 'wire fraud', 'wire transfer',
                          'embezzlement', 'identity theft', 'identity fraud',
                          'passport fraud', 'stolen identity', 'forgery',
                          'counterfeit', 'theft of government', 'theft
                           ↪  related',
                          'government funds']

    # Health Care Fraud
    health_keywords = ['health care', 'medicare', 'medicaid', 'tricare',
                       'false claims', 'false billing', 'billing scheme',
                       'upcoding', 'phantom billing', 'unnecessary services',
                       'medically unnecessary', 'durable medical equipment',
                       'home health', 'hospice', 'nursing', 'hospital',
                       'physician', 'doctor', 'clinic', 'telemedicine',
                       'telehealth', 'laboratory', 'ambulance', 'pharmacy',
                       'patient', 'medical', 'health', 'hhs', 'cms',
                       'insurance fraud', 'claims', 'restitution',
```

```
                      'settle', 'resolve', 'plea', 'sentenced', 'convicted',
                      'indicted', 'charged', 'guilty']

    if any(kw in title_lower for kw in drug_keywords):
        return 'Drug_Enforcement'
    elif any(kw in title_lower for kw in bribery_keywords):
        return 'Bribery_Corruption'
    elif any(kw in title_lower for kw in financial_keywords):
        return 'Financial_Fraud'
    elif any(kw in title_lower for kw in health_keywords):
        return 'Health_Care_Fraud'
    else:
        return 'Other'

criminal_civil['topic'] = criminal_civil['title'].apply(classify_topic)

monthly_by_topic = (criminal_civil.groupby(['year_month', 'topic'])
                    .size()
                    .reset_index(name='count'))

chart3 = (alt.Chart(monthly_by_topic).mark_line(point=True).encode(
  x=alt.X('year_month:T', title='Month'),
  y=alt.Y('count:Q', title='Number of Enforcement Actions'),
  color=alt.Color('topic:N', title='Topic')
  ).properties(
    title='Criminal and Civil Actions by Topic (Since Jan 2022)',
    width=500, height=350
  )
)

chart3
```


Criminal and Civil Actions by Topic (Since Jan 2022)
Number of Enforcement Actions
50
45
40
35
30
25
20
15
10
5
0
2022
July
2023
July
2024
July
2025
July
2026
Month
Topic
Bribery_Corruption
Drug_Enforcement
Financial_Fraud
Health_Care_Fraud
Other