PS4

Clara Leo

2026-02-03

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: CL

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

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

Step 1: Develop initial scraper and crawler

```python
import requests

# I asked Chat to give me the code to complete the following
# step for a URL instead of a file since the lecture only
# showed us how to do it for a file.
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
html = response.text

from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
```

```python
title_links = []

for h2 in soup.find_all("h2"):
    a = h2.find("a", href=True)
    if a and a["href"].startswith("/fraud/enforcement/"):
        title_links.append(a)

rows = []

for a in title_links:
    card = a.find_parent("li")
    if not card:
        continue

    title = a.get_text(strip=True)
    link = "https://oig.hhs.gov" + a["href"]

    date = card.find(
        "span",
        class_="text-base-dark padding-right-105"
    ).get_text(strip=True)

    categories = [
        li.get_text(strip=True)
        for li in card.find_all(
            "li",
            class_="display-inline-block usa-tag text-no-lowercase
↪   text-base-darkest bg-base-lightest margin-right-1"
        )
    ]

    rows.append({
        "title": title,
        "date": date,
        "categories": categories,
        "link": link
    })

df = pd.DataFrame(rows)
df
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

FUNCTION violations_df(url, year, month) If year<2013, output "No data before 2013" data = [empty set] Looping infinitely: title = [title from soup] date = [date from soup] category = [category from soup] url = [url from soup] append to data pause for 1 second Break when page is empty END FUNCTION

- b. Create Dynamic Scraper

```
from datetime import datetime

def violations_df(base_url, year, month):
    # --- guard clause ---
    if year < 2013:
        print("No data before 2013")
        return None

    cutoff = datetime(year, month, 1)

    rows = []
    page = 0

    while True:
        url = f"{base_url}?page={page}"
        print(f"Scraping page {page}")

        resp = requests.get(url)
        soup = BeautifulSoup(resp.text, "html.parser")

        title_links = []
        for h2 in soup.find_all("h2"):
            a = h2.find("a", href=True)
            if a and a["href"].startswith("/fraud/enforcement/"):
                title_links.append(a)

        # --- stop if page has no enforcement actions ---
        if not title_links:
            print("Stopping: no enforcement actions found on page")
            break

        rows_before = len(rows)

        for a in title_links:
            card = a.find_parent("li")
            if not card:
                continue

            title = a.get_text(strip=True)
            link = "https://oig.hhs.gov" + a["href"]

            date_text = card.find(
                "span",
                class_="text-base-dark padding-right-105"
            ).get_text(strip=True)

            date = datetime.strptime(date_text, "%B %d, %Y")

            # --- stop once we pass cutoff ---
            if date < cutoff:
                print("Stopping: reached date cutoff")
                return pd.DataFrame(rows)

            categories = [
                li.get_text(strip=True)
                for li in card.find_all(
                    "li",
                    class_="display-inline-block usa-tag text-no-lowercase
↪   text-base-darkest bg-base-lightest margin-right-1"
                )
            ]

            rows.append({
                "title": title,
                "date": date,
                "categories": categories,
                "link": link
            })

        # --- stop if page added nothing new ---
        if len(rows) == rows_before:
            print("Stopping: no new rows added")
            break

        page += 1
        time.sleep(1)

    return pd.DataFrame(rows)
```

- c. Test Your Code

```
page = 'https://oig.hhs.gov/fraud/enforcement/'
month = 1
year = 2022

#df_since2022 = violations_df(page, year, month)
```

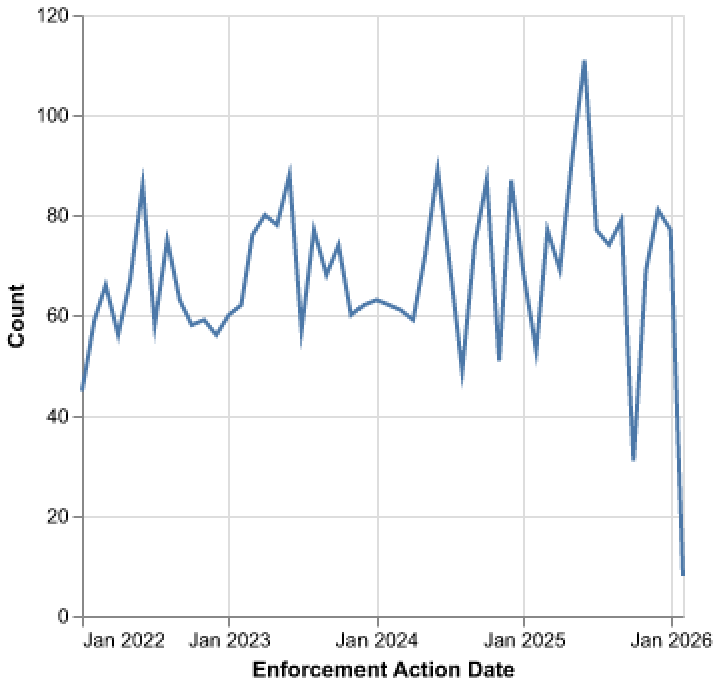
```
# df_since2022.to_csv(
#     "enforcement_actions_year_month.csv",
#     index=False
# )
```

```python
import os as os

PATH = '/Users/Carl/Documents/gitfiles'
new_df_path = os.path.join(PATH, 'enforcement_actions_year_month.csv')
new_df = pd.read_csv(new_df_path)

new_df
```

Step 3: Plot data based on scraped data

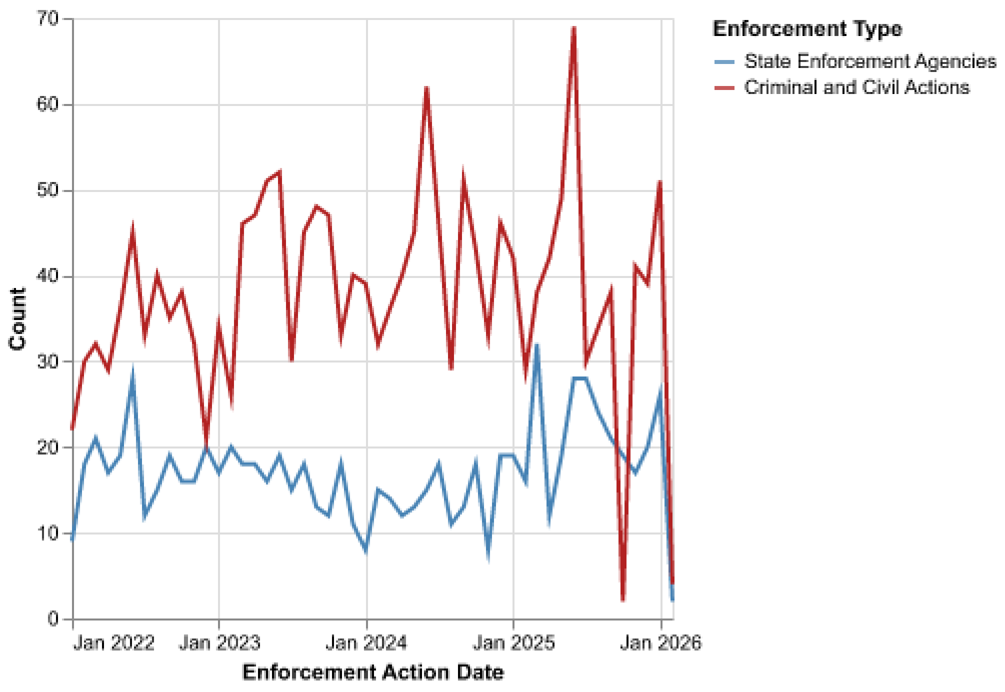
There are 3,379 observations in the dataframe. The earliest scraped enforcement action is from January 4, 2022. The title is "Integrated Pain Management Medical Group Adreed to Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded Individuals." The category is is Fraud Self-Disclosures.

1. Plot the number of enforcement actions over time



2. Plot the number of enforcement actions categorized:

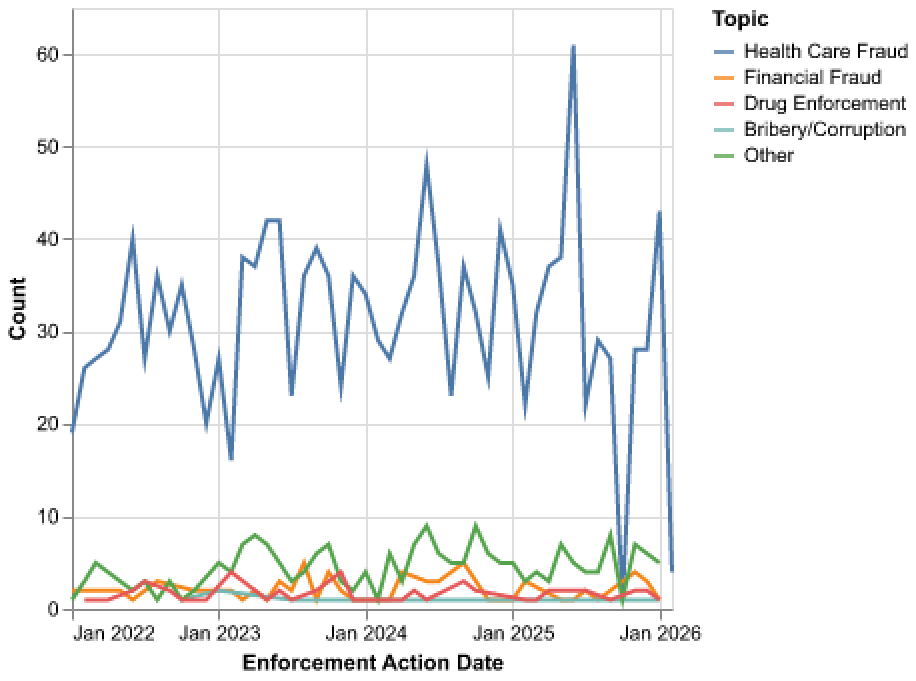
- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"



- based on five topics

```
# Establish the patterns

import re

health_care_keywords = [
    "medicare", "medicaid", "health care", "healthcare", "medical",
    "physician", "doctor", "nurse", "clinic", "hospital",
    "home health", "laboratory", "pharmacy", "telehealth",
    "false claims", "kickback"
]

financial_keywords = [
    "bank", "banker", "wire fraud", "money laundering",
    "identity theft", "embezzlement", "financial",
    "covid", "relief", "ppp"
]
```

```python
drug_keywords = [
    "opioid", "controlled substance", "oxycodone",
    "drug", "distribution", "pill mill",
    "adulterated", "misbranded"
]

bribery_keywords = [
    "bribery", "corruption", "kickback", "conspiracy",
    "illegal remuneration", "self-referral"
]

def assign_topic(title):
    title = title.lower()

    if any(k in title for k in health_care_keywords):
        return "Health Care Fraud"
    elif any(k in title for k in drug_keywords):
        return "Drug Enforcement"
    elif any(k in title for k in financial_keywords):
        return "Financial Fraud"
    elif any(k in title for k in bribery_keywords):
        return "Bribery/Corruption"
    else:
        return "Other"

new_df["topic"] = new_df["title"].apply(assign_topic)

new_df['topic'].unique
```


Topic
Health Care Fraud
Financial Fraud
Drug Enforcement
Bribery/Corruption
Other
Count
60
50
40
30
20
10
0
Jan 2022
Jan 2023
Jan 2024
Jan 2025
Jan 2026
Enforcement Action Date