Problem Set 4

Shuyan Xin

2026-02-02

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: SX

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

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, "lxml")
cards = soup.find_all("div", class_="usa-card__container")

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

for card in cards:
    a_tag = card.find("a")
    titles.append(a_tag.text.strip())
    date_tag = card.find("span")
    dates.append(date_tag.text.strip())
    category_tag = card.find("li")
    categories.append(category_tag.text.strip())
    links.append("https://oig.hhs.gov" + a_tag.get('href'))

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

print(df.head())
```

```
                                                title                date  \
0  Houston Transplant Doctor Indicted For Making ...  February 5, 2026
1  MultiCare Health System to Pay Millions to Set...  February 4, 2026
2  Brooklyn Banker Pleads Guilty to Laundering Pr...  February 3, 2026
3  Delafield Man Sentenced to 18 Months' Imprison...  February 3, 2026
4  Former NFL Player Convicted for $197M Medicare...  February 3, 2026

                     category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2                   COVID-19
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                link
0  https://oig.hhs.gov/fraud/enforcement/houston-...
1  https://oig.hhs.gov/fraud/enforcement/multicar...
2  https://oig.hhs.gov/fraud/enforcement/brooklyn...
3  https://oig.hhs.gov/fraud/enforcement/delafiel...
4  https://oig.hhs.gov/fraud/enforcement/former-n...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

```
1. The function first checks whether the input year is earlier than 2013. If
 ↪  the year is less than 2013, it prints a warning message and stops the
 ↪  function.

2. The function creates a starting date using the input year and month, which
 ↪  will be used later to determine when to stop scraping.

3.
The page number is initialized to 0 so that the scraping process begins from
 ↪  the first page of enforcement actions.

4.
We create empty data structures to store the extracted information, including
 ↪  titles, dates, categories, and links.
```

```
5.
Then we need to use a while loop with a for loop embedded inside.
The while loop is used to repeatedly scrape pages until a stopping condition
 ↪ is met.
The for loop is used to iterate through and extract information from each
 ↪ enforcement action on a specific page, adding the data to the
 ↪ corresponding lists.
(1) In each iteration, the function constructs the URL using the current page
 ↪ number, retrieves the page, parses the HTML, and finds all enforcement
 ↪ action cards. If no cards are found, the loop stops.
(2) For each card, the function extracts the date and converts it to a
 ↪ datetime object. If the date is earlier than the starting date, the
 ↪ function stops scraping. If the date is in the correct range, the
 ↪ function extracts the title, category, and link, and stores all the
 ↪ information needed in the corresponding lists.
(3) After finishing one page, the page number is increased by one so that the
 ↪ function moves on to the next page.

6. Finally, we can run the function to return a DataFrame of all the scraped
 ↪ data according to the year and month we input.
```

- b. Create Dynamic Scraper

```python
def scrape(year, month):
    if year < 2013:
        print("Please restrict to year >= 2013.")
        return

    start_date = pd.to_datetime(f"{year}-{month}-01")
    page = 0
    titles = []
    dates = []
    categories = []
    links = []

    while True:
        url = f"https://oig.hhs.gov/fraud/enforcement/?page={page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, "lxml")
        cards = soup.find_all('div', class_='usa-card__container')

        if len(cards) == 0:
```

```
            break

        for card in cards:
            date_tag = card.find('span')
            date_text = date_tag.text.strip()
            date = pd.to_datetime(date_text)

            if date < start_date:
                return pd.DataFrame({
                    "title": titles,
                    "date": pd.to_datetime(dates),
                    "category": categories,
                    "link": links
                })

            a_tag = card.find('a')
            titles.append(a_tag.text.strip())
            dates.append(date_text)
            category_tag = card.find('li')
            categories.append(category_tag.text.strip())
            links.append("https://oig.hhs.gov" + a_tag.get('href'))

        page = page + 1

df_2024 = scrape(2024, 1)

print(len(df_2024))
print(f"We can get {len(df_2024)} enforcement actions from January 2024 till
 ↪  now.")

print(df_2024.sort_values("date").head(1))
```

```
1807
We can get 1807 enforcement actions from January 2024 till now.
                                                  title       date  \
1806  Former Nurse Aide Indicted In Death Of Clarksv... 2024-01-03

                        category  \
1806  State Enforcement Agencies

                                                  link
1806  https://oig.hhs.gov/fraud/enforcement/former-n...
```

- c. Test Your Code

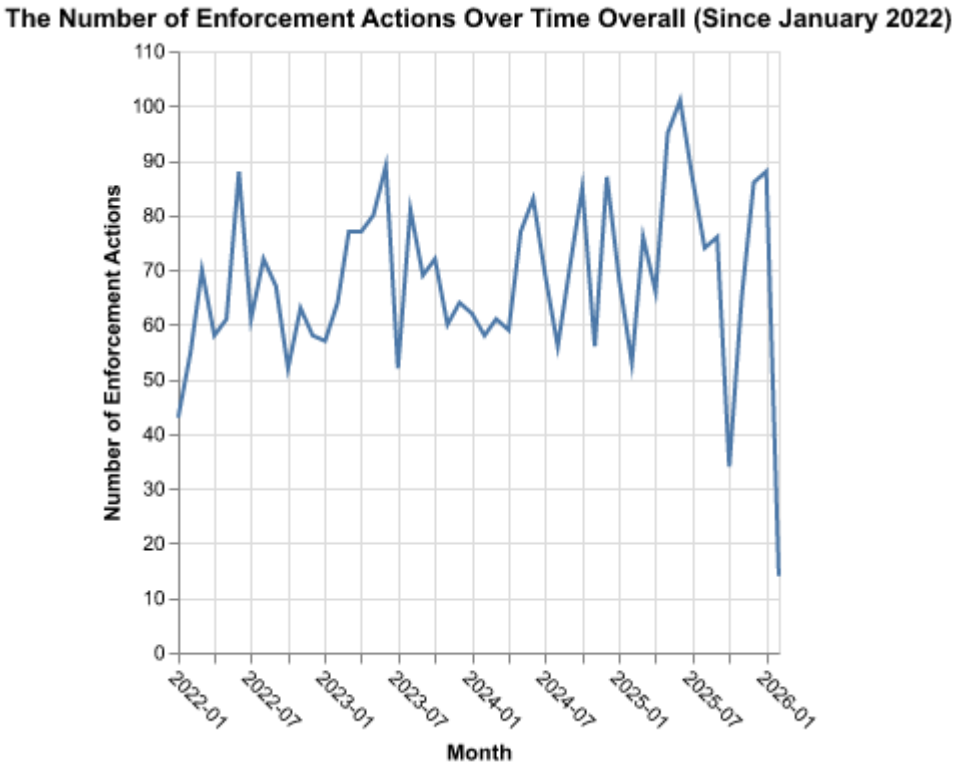
```
df_2022 = scrape(2022, 1)

print(len(df_2022))
print(f"We can get {len(df_2022)} enforcement actions from January 2022 till
 ↪  now.")

print(df_2022.sort_values("date").head(1))

df_2024.to_csv("enforcement_actions_2024_01.csv", index=False)
df_2022.to_csv("enforcement_actions_2022_01.csv", index=False)
df_2022.to_csv("enforcement_actions_year_month.csv", index=False)
```

```
3397
We can get 3397 enforcement actions from January 2022 till now.
                                                      title       date  \
3396  Integrated Pain Management Medical Group Agree... 2022-01-04

                    category  \
3396  Fraud Self-Disclosures

                                                      link
3396  https://oig.hhs.gov/fraud/enforcement/integrat...
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

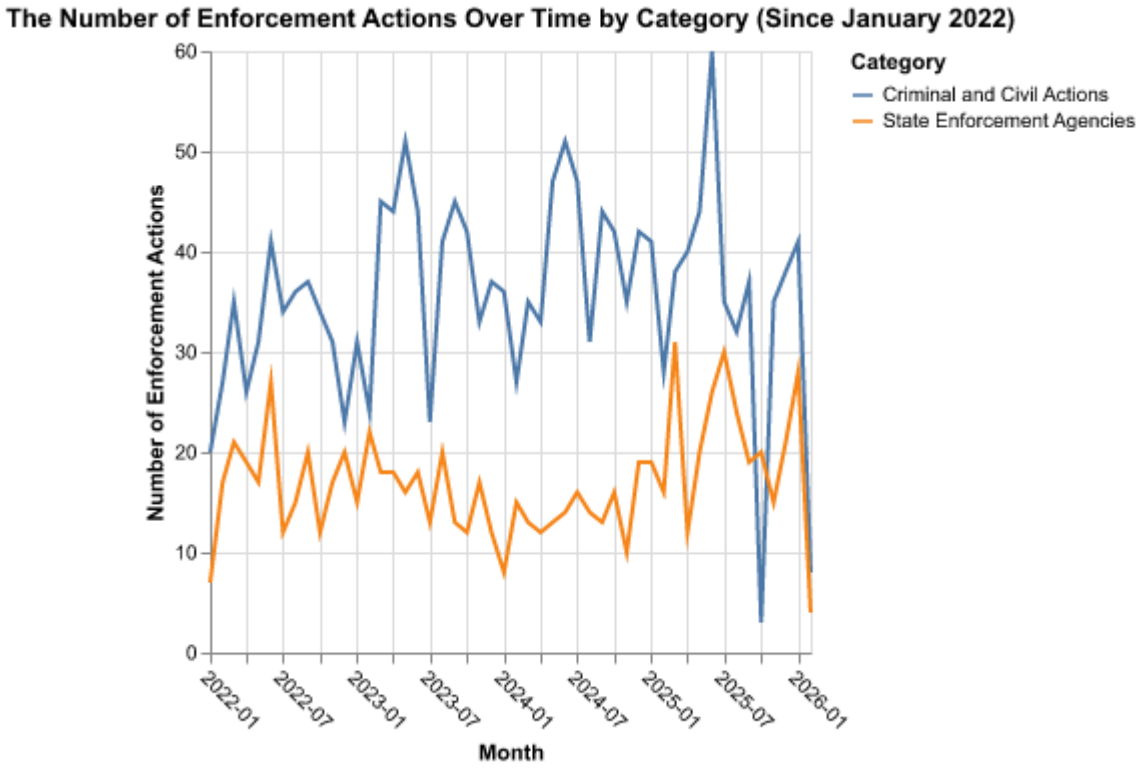
```
enforcement_actions_over_time = alt.Chart(df_2022).mark_line().encode(
    x = alt.X("yearmonth(date):T", title="Month",
 ↪  axis=alt.Axis(format="%Y-%m", tickCount=12, labelAngle=45)),
    y = alt.Y("count(title):Q", title="Number of Enforcement Actions")
). properties(title ="The Number of Enforcement Actions Over Time Overall
 ↪  (Since January 2022)")

enforcement_actions_over_time
```



2. Plot the number of enforcement actions categorized:

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

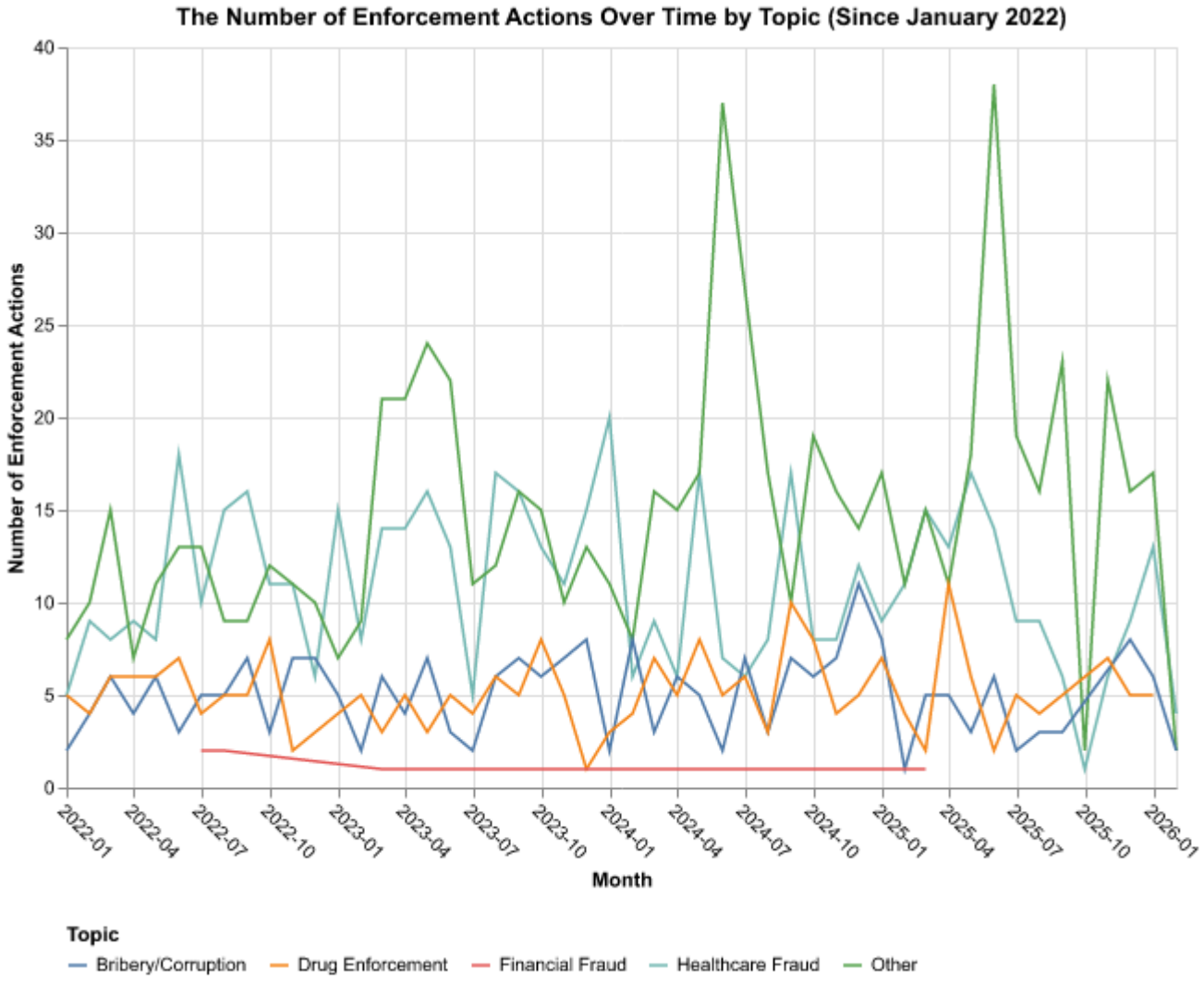
```
df_2022_sub = df_2022[df_2022["category"].isin(["Criminal and Civil Actions",
 ↪  "State Enforcement Agencies"])]

enforcement_actions_by_category = alt.Chart(df_2022_sub).mark_line().encode(
    x = alt.X("yearmonth(date):T", title="Month",
 ↪  axis=alt.Axis(format="%Y-%m", tickCount=12, labelAngle=45)),
    y = alt.Y("count(title):Q", title="Number of Enforcement Actions"),
    color = alt.Color("category:N", title="Category")
). properties(
    title ="The Number of Enforcement Actions Over Time by Category (Since
 ↪  January 2022)"
    )

enforcement_actions_by_category
```



- based on five topics

```
df_2022_CCA = df_2022[df_2022["category"] == "Criminal and Civil Actions"]
df_2022_CCA.loc[

 ↪  df_2022_CCA["title"].str.contains("Medicare|Medicaid|Healthcare|Medical|Hospital|Clinic"
 ↪  case=False, na=False),
    "topic"
] = "Healthcare Fraud"
df_2022_CCA.loc[
    df_2022_CCA["title"].str.contains("Bank|Finance|Financial|Loan|Mortgage",
 ↪  case=False, na=False),
    "topic"
] = "Financial Fraud"
df_2022_CCA.loc[
    df_2022_CCA["title"].str.contains("Opioid|Drug|Pharma|Medication",
 ↪  case=False, na=False),
    "topic"
] = "Drug Enforcement"
df_2022_CCA.loc[
```

```
    df_2022_CCA["title"].str.contains("Bribery|Corruption|Kickback",
    case=False, na=False),
    "topic"
] = "Bribery/Corruption"
df_2022_CCA["topic"] = df_2022_CCA["topic"].fillna("Other")

enforcement_actions_by_topic =
    alt.Chart(df_2022_CCA).mark_line(strokeWidth=1.5).encode(
    x = alt.X("yearmonth(date):T", title="Month",
    axis=alt.Axis(format="%Y-%m", tickCount=12, labelAngle=45)),
    y = alt.Y("count(title):Q", title="Number of Enforcement Actions"),
    color = alt.Color(
        "topic:N",
        title="Topic",
        legend=alt.Legend(orient="bottom", labelFontSize=10)
    )
).properties(
    title="The Number of Enforcement Actions Over Time by Topic (Since
    January 2022)",
    height=400,
    width=600
    )

enforcement_actions_by_topic
```


The Number of Enforcement Actions Over Time by Topic (Since January 2022)
Number of Enforcement Actions
0
5
10
15
20
25
30
35
40
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
2024-04
2024-07
2024-10
2025-01
2025-04
2025-07
2025-10
2026-01
Month
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Healthcare Fraud
Other