Problem Set #4

Zheng Chen

2026-02-05

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: ZC

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")

import requests
from bs4 import BeautifulSoup
```

Step 1: Develop initial scraper and crawler

```python
URL = 'https://oig.hhs.gov/fraud/enforcement/'

response = requests.get(URL)
soup = BeautifulSoup(response.text, 'lxml')

cards = soup.select("li.usa-card")
print(len(cards))
```

```
20
```

```python
### Check if the selcted tag works
first = cards[0]

a = first.select_one("h2 a")
title_exp = a.get_text(strip=True)
link_exp = "https://oig.hhs.gov" + a["href"]

date_exp = first.find("span", class_="text-base-dark
 ↪  padding-right-105").get_text(strip=True)

category_exp = first.find("li", class_="usa-tag").get_text(strip=True)

print(title_exp)
print(link_exp)
print(date_exp)
print(category_exp)
```

```
Houston Transplant Doctor Indicted For Making False Statements In Patients'
Medical Records
https://oig.hhs.gov/fraud/enforcement/houston-transplant-doctor-indicted-for-making-false-sta
February 5, 2026
Criminal and Civil Actions
```

```python
data = []

for card in cards:
    a = card.find("a")
    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)

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

    data.append({
        "title": title,
        "date": date,
        "category": category,
        "link": link
    })

df = pd.DataFrame(data)
df.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 web_scraper(start_year, start_month): BEGIN IF start_year < 2013: PRINT "Only enforcement actions after 2013 are listed" RETURN

SET page = 1 SET data = empty list SET continue_scraping = TRUE

WHILE continue_scraping == TRUE:

```
SET url = "https://oig.hhs.gov/fraud/enforcement/?page=" + page
REQUEST html from url
PARSE html with BeautifulSoup
FIND cards = all <li class="usa-card">

IF number of cards == 0:
    BREAK

FOR each card in cards:

    FIND <a> inside card
    SET title = text of <a>
    SET link = full URL from <a href>

    FIND <span class="text-base-dark padding-right-105">
    SET date = text of span
    EXTRACT year and month from date

    FIND <li class="usa-tag">
    SET category = text of li

    IF (year < start_year) OR
       (year == start_year AND month < start_month):
        SET continue_scraping = FALSE
        BREAK out of FOR loop

    APPEND {title, date, category, link} to data

WAIT 1 second
INCREMENT page by 1
```

CONVERT data to dataframe SAVE dataframe as "enforcement_actions_{start_year}_{start_month}.c

RETURN dataframe END ENDFUNCTION · · ·

- b. Create Dynamic Scraper

```python
def web_scraper(start_year, start_month):
    if start_year < 2013:
        print("Only enforcement actions after 2013 are listed")
        return None

    page = 1
    data = []
    continue_scraping = True

    while continue_scraping:
        url = f"https://oig.hhs.gov/fraud/enforcement/?page={page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, "lxml")

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

        if len(cards) == 0:
            break

        for card in cards:
            a = card.find("a")
            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)

            parts = date_text.split()
            year = int(parts[2])
            month_map = {
                "January": 1, "February": 2, "March": 3, "April": 4,
                "May": 5, "June": 6, "July": 7, "August": 8,
                "September": 9, "October": 10, "November": 11, "December":
                 ↪  12}
            month = month_map[parts[0]]

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

            if (year < start_year) or (year == start_year and month <
             ↪  start_month):
                continue_scraping = False
                break

            data.append({
                "title": title,
                "date": date_text,
                "category": category,
                "link": link
            })

        time.sleep(1)
        page += 1

    df = pd.DataFrame(data)
    df.to_csv(
        f"enforcement_actions_{start_year}_{start_month}.csv",
        index=False
    )

    return df
```

```
df_2024 = web_scraper(2024, 1)

print(f'The dataframe has {len(df_2024)} rows')
df_2024.head(5)
```

```
The dataframe has 1787 rows
```

	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

- c. Test Your Code

```python
df_2022 = web_scraper(2022, 1)
print(f'The new dataframe has {len(df_2022)} rows')
df_2022.head(5)
```

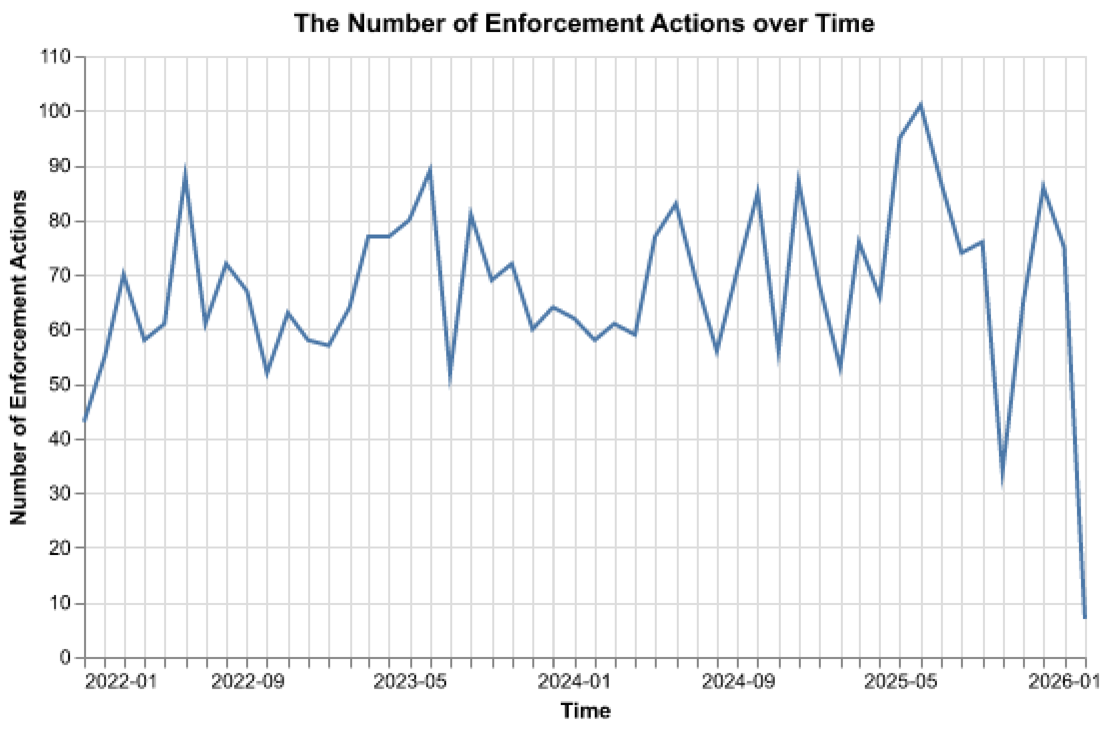
```
The new dataframe has 3377 rows
```

	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 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

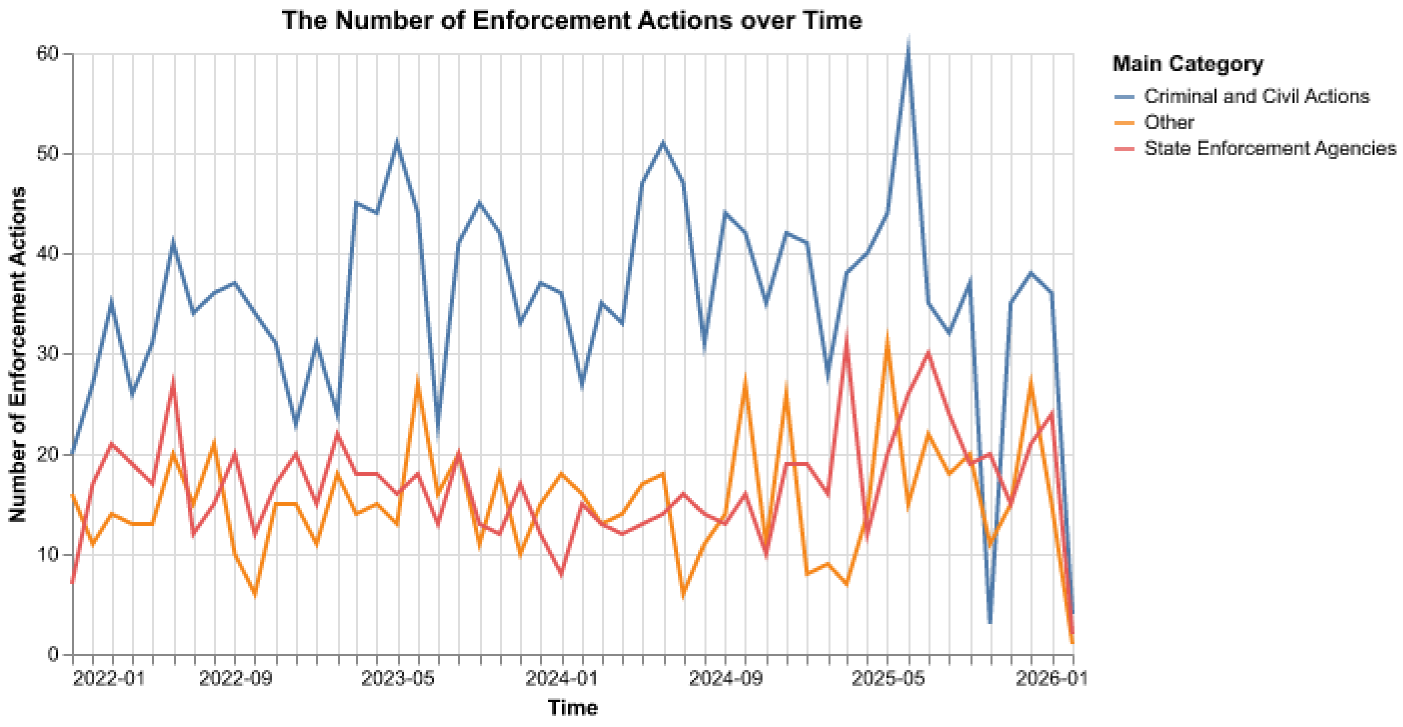
```python
df_2022["date"] = pd.to_datetime(df_2022["date"])

chart_line = alt.Chart(df_2022).transform_timeunit(
    ym='yearmonth(date)'
).transform_aggregate(
    Count='count()',
    groupby=['ym']
).mark_line().encode(
    alt.X('ym:T',title='Time',
    axis=alt.Axis(format='%Y-%m',tickCount=50)),
    alt.Y('Count:Q', title='Number of Enforcement Actions')
).properties(
    title='The Number of Enforcement Actions over Time',
    width=500,height=300
)

chart_line
```



2. Plot the number of enforcement actions categorized:

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

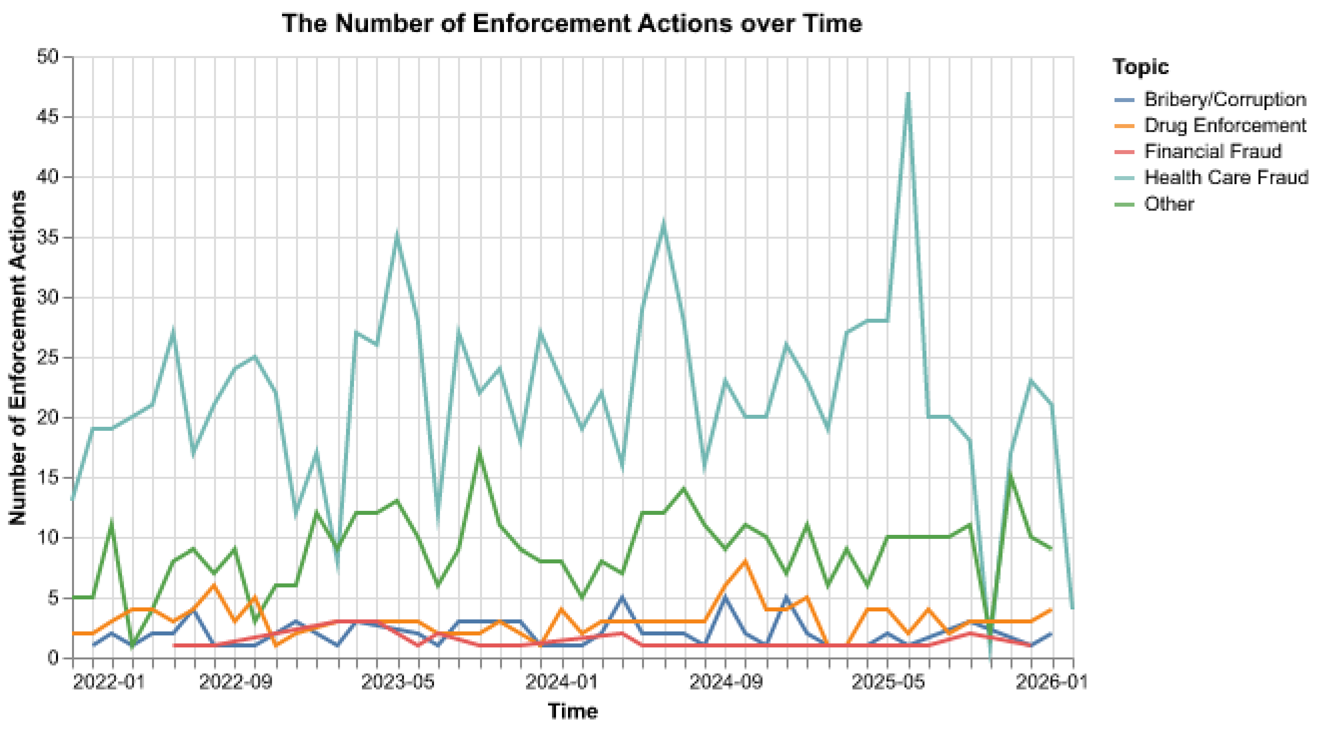
```
df_2022["main_category"] = df_2022["category"].where(
    df_2022["category"].isin([
        "Criminal and Civil Actions",
        "State Enforcement Agencies"
    ]),
    other="Other"
)

chart_line_cat = alt.Chart(df_2022).transform_timeunit(
    ym='yearmonth(date)'
).transform_aggregate(
    Count='count()',
    groupby=['ym', 'main_category']
).mark_line().encode(
    alt.X('ym:T',title='Time',
    axis=alt.Axis(format='%Y-%m',tickCount=50)),
    alt.Y('Count:Q', title='Number of Enforcement Actions'),
    alt.Color('main_category:N',legend=alt.Legend(title='Main Category'))
).properties(
    title='The Number of Enforcement Actions over Time',
    width=500,height=300
)

chart_line_cat
```



- based on five topics

```
def classify_topic(title):
    t = title.lower()

    if any(k in t for k in ["medicare", "medicaid", "health", "hospital",
     ↪  "doctor", "patient"]):
        return "Health Care Fraud"

    if any(k in t for k in ["bank", "financial", "loan", "wire",
     ↪  "securities", "tax"]):
        return "Financial Fraud"
```

```python
    if any(k in t for k in ["drug", "opioid", "pharmacy", "fentanyl",
     ↪  "controlled substance"]):
        return "Drug Enforcement"

    if any(k in t for k in ["bribe", "bribery", "corruption", "kickback"]):
        return "Bribery/Corruption"

    return "Other"

df_2022["topic"] = None

mask = df_2022["main_category"] == "Criminal and Civil Actions"
df_2022.loc[mask, "topic"] = df_2022.loc[mask, "title"].apply(classify_topic)

chart_line_topic = alt.Chart(df_2022).transform_filter(
    alt.datum.topic != None
).transform_timeunit(
    ym='yearmonth(date)'
).transform_aggregate(
    Count='count()',
    groupby=['ym', 'topic']
).mark_line().encode(
    alt.X('ym:T',title='Time',
    axis=alt.Axis(format='%Y-%m',tickCount=50)),
    alt.Y('Count:Q', title='Number of Enforcement Actions'),
    alt.Color('topic:N',legend=alt.Legend(title='Topic'))
).properties(
    title='The Number of Enforcement Actions over Time',
    width=500,height=300
)

chart_line_topic
```


The Number of Enforcement Actions over Time
Number of Enforcement Actions
Time
0
5
10
15
20
25
30
35
40
45
50
2022-01
2022-09
2023-05
2024-01
2024-09
2025-05
2026-01
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other