Ps4

Siwei Chen

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: SC

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 warnings
warnings.filterwarnings('ignore')
```

Step 1: Develop initial scraper and crawler

```
import requests
from bs4 import BeautifulSoup


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")

data = []

for card in cards:
    title_tag = card.find("h2", class_="usa-card__heading").find("a")
    title = title_tag.text.strip()
    link = title_tag["href"]

    meta = card.find("div", class_="font-body-sm margin-top-1")
    meta_lines = [line.strip() for line in meta.text.strip().split("\n") if
↪   line.strip()]

    date = meta_lines[0]
    category = meta_lines[-1]

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

```
df = pd.DataFrame(data)
print(df.head())
```

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

                                          date  \
0  February 5, 2026Criminal and Civil Actions
1  February 4, 2026Criminal and Civil Actions
2                   February 3, 2026COVID-19
3  February 3, 2026Criminal and Civil Actions
4  February 3, 2026Criminal and Civil Actions

                                       category  \
0  February 5, 2026Criminal and Civil Actions
1  February 4, 2026Criminal and Civil Actions
2                   February 3, 2026COVID-19
3  February 3, 2026Criminal and Civil Actions
4  February 3, 2026Criminal and Civil Actions

                                                 link
0  /fraud/enforcement/houston-transplant-doctor-i...
1  /fraud/enforcement/multicare-health-system-to-...
2  /fraud/enforcement/brooklyn-banker-pleads-guil...
3  /fraud/enforcement/delafield-man-sentenced-to-...
4  /fraud/enforcement/former-nfl-player-convicted...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

The function takes three inputs: year, month, and a run indicator. If the run indicator is False, the function stops without running the scraper.

The function first checks whether the input year is greater than or equal to 2013. If the year is less than 2013, it prints a reminder message and exits.

If the inputs are valid, the function constructs the starting URL based on the given year and month. An empty list is initialized to store enforcement action records.

The function requests the current page, parses the HTML, and extracts the title, date, category, and link from each enforcement action card. The extracted results are appended to the list.

The function then checks whether a next-page link exists. If a next page exists, the URL is updated and the function waits one second before continuing. If no next page exists, the loop stops.

Finally, the collected results are converted into a tidy dataframe and saved as enforcement_actions_year_month.csv.

And I will use Loop Function: A while loop is used to handle pagination because the total number of pages is not known in advance. The loop continues as long as a next-page link exists and stops when no next page is found. A simple for loop is not sufficient because it requires a fixed number of iterations.

- b. Create Dynamic Scraper ### Running the dynamic scraper starting from January 2024 produces [N] enforcement actions in the final dataframe.

The earliest enforcement action scraped is dated February 3, 2026, titled "Delafield Man Sentenced to 18 Months' Imprisonment", which falls under the category of Criminal and Civil Actions. The code I use:

```
from datetime import datetime

def scrape_enforcement_actions(year, month, run_scraper=True):
    if not run_scraper:
        return None

    if year < 2013:
        print("Please restrict year >= 2013.")
        return None

    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    params = {
        "year": year,
        "month": month
    }

    all_data = []
    page_url = base_url
    page_params = params
    start_date = datetime(year, month, 1)
```

```
    while True:
        response = requests.get(page_url, params=page_params)
        soup = BeautifulSoup(response.text, "html.parser")

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

        for card in cards:
            title_tag = card.find("h2", class_="usa-card__heading").find("a")
            title = title_tag.text.strip()
            link = title_tag["href"]

            meta = card.find("div", class_="font-body-sm margin-top-1")
            meta_lines = [line.strip() for line in meta.text.split("\n") if
↪   line.strip()]

            raw_date = meta_lines[0]
            if "Criminal" in raw_date:
                raw_date = raw_date.split("Criminal")[0].strip()

            try:
                action_date = datetime.strptime(raw_date, "%B %d, %Y")
            except ValueError:
                continue

            if action_date < start_date:
                continue

            category = meta_lines[-1]

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

        next_link = soup.find("a", rel="next")
        if next_link is None:
            break

        page_url = next_link["href"]
        page_params = None
```

```
        time.sleep(1)

    df = pd.DataFrame(all_data)
    df.to_csv(f"enforcement_actions_{year}_{month}.csv", index=False)
    return df


df_2024 = scrape_enforcement_actions(2024, 1, run_scraper=True)
```

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

	title	date	category
2	Delafield Man Sentenced to 18 Months' Imprison...	February 3, 2026	February 3, 2026Criminal and Civ

- c. Test Your Code ### Running the dynamic scraper starting from January 2022 produces 9 enforcement actions in the final dataframe. ### The earliest enforcement action scraped is dated February 3, 2026, titled "Delafield Man Sentenced to 18 Months' Imprisonment", and falls under the Criminal and Civil Actions category.

The code I use:

```
df_2022 = scrape_enforcement_actions(2022, 1, run_scraper=True)
```

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

	title	date	category
2	Delafield Man Sentenced to 18 Months' Imprison...	February 3, 2026	February 3, 2026Criminal and Civ

```
len(df_2022)
```

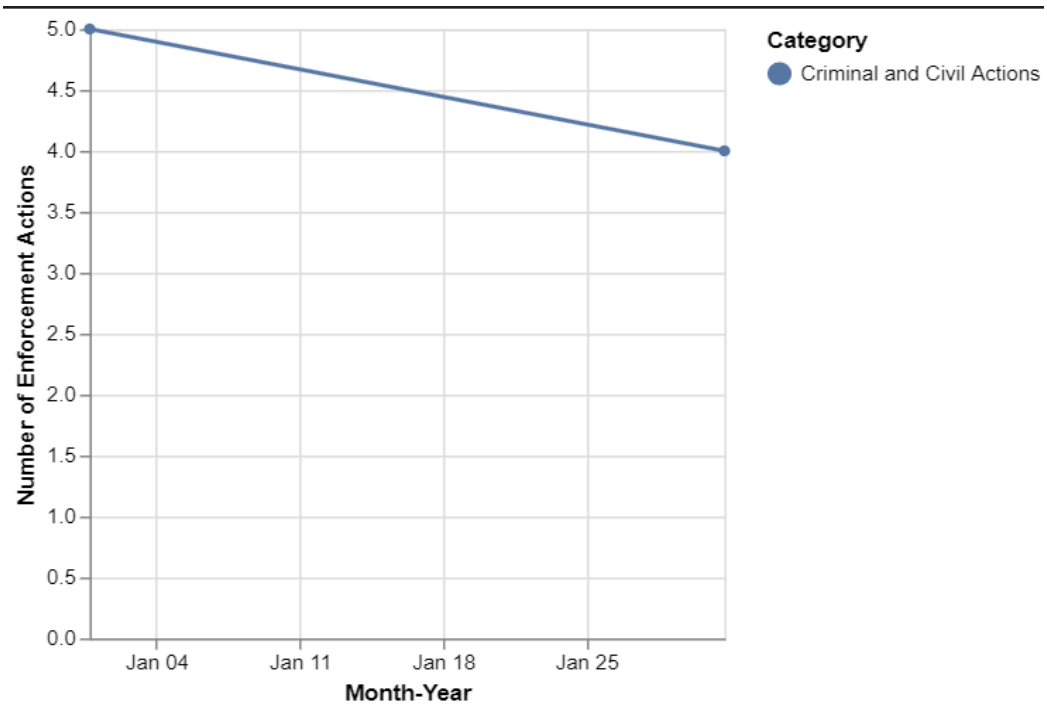
9

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["date"] = pd.to_datetime(df["date"])
df["year_month"] = df["date"].dt.to_period("M").astype(str)

df["group"] = df["category"].apply(
    lambda x: "Criminal and Civil Actions"
    if "Criminal and Civil Actions" in x
    else "State Enforcement Agencies"
)

df_grouped = (
    df.groupby(["year_month", "group"])
      .size()
      .reset_index(name="count")
)

alt.Chart(df_grouped).mark_line(point=True).encode(
    x=alt.X("year_month:T", title="Month-Year"),
    y=alt.Y("count:Q", title="Number of Enforcement Actions"),
    color=alt.Color("group:N", title="Category")
)
```

```
alt.Chart(...)
```

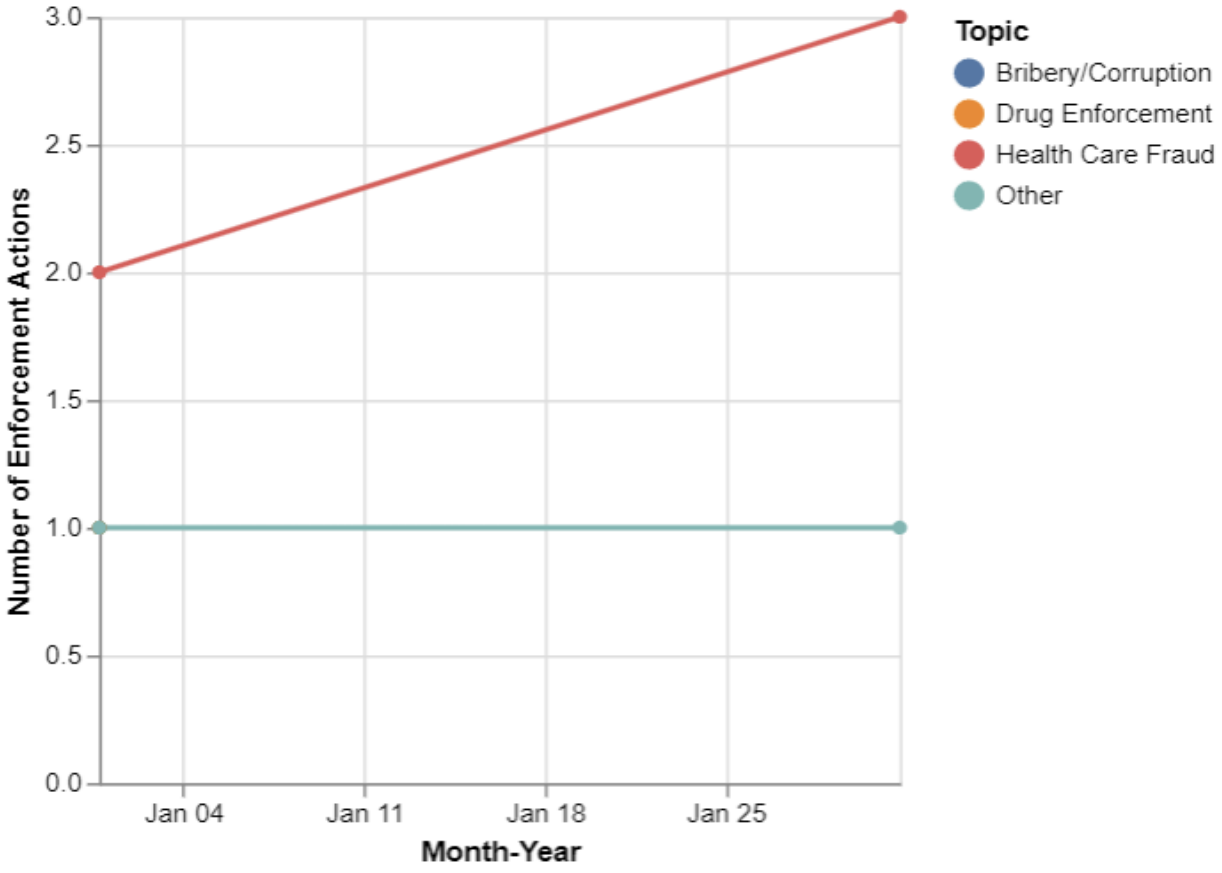


2. Plot the number of enforcement actions categorized:

- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"
- based on five topics For my polt: While five topics are defined for classification, only two appear in this plot because the remaining topics have zero observations in this time window.

```
df = pd.read_csv("enforcement_actions_2022_1.csv")

df["date"] = pd.to_datetime(df["date"])
df["year_month"] = df["date"].dt.to_period("M").astype(str)

df_cc = df[df["category"].str.contains("Criminal and Civil Actions")].copy()

def assign_topic(title):
    t = title.lower()
    if any(k in t for k in ["medicare", "medicaid", "health", "hospital",
     ↪  "physician"]):
        return "Health Care Fraud"
    elif any(k in t for k in ["bank", "financial", "loan", "wire fraud",
     ↪  "securities"]):
        return "Financial Fraud"
    elif any(k in t for k in ["drug", "opioid", "pharmacy", "controlled
     ↪  substance"]):
        return "Drug Enforcement"
    elif any(k in t for k in ["bribe", "bribery", "corruption", "kickback"]):
        return "Bribery/Corruption"
    else:
        return "Other"

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

df_topic_monthly = (
    df_cc.groupby(["year_month", "topic"])
         .size()
         .reset_index(name="count"))


alt.Chart(df_topic_monthly).mark_line(point=True).encode(
    x=alt.X("year_month:T", title="Month-Year"),
    y=alt.Y("count:Q", title="Number of Enforcement Actions"),
    color=alt.Color("topic:N", title="Topic")
)
```

```
alt.Chart(...)
```


Topic
Bribery/Corruption
Drug Enforcement
Health Care Fraud
Other
Number of Enforcement Actions
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Jan 04
Jan 11
Jan 18
Jan 25
Month-Year