PS4

Cecilia Zhong

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: **_C.Z_**

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')
alt.renderers.enable("png")
```

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

Step 1: Develop initial scraper and crawler

```
import requests
from bs4 import BeautifulSoup
BASE_URL = "https://oig.hhs.gov"
START_URL = "https://oig.hhs.gov/fraud/enforcement/"

# Define user agent header to identify scraper
myheader = {
    "User-Agent": "DAP30538CourseBot/1.0 (m.shi@uchicago.edu)"
}

# Request HTML content from enforcement page
response = requests.get(START_URL, headers=myheader)
soup = BeautifulSoup(response.content, "lxml")

rows = []
seen = []

# List of month names used to detect date strings
months = ["January","February","March","April","May","June","July",
          "August","September","October","November","December"]

# Find all hyperlink tags and filter potential enforcement action links
all_a = soup.find_all("a")

for a in all_a:
    href = a.get("href")

    # Keep only enforcement-related links
    if (href is None) or (not href.startswith("/fraud/enforcement/")):
```

```
        continue
    if href == "/fraud/enforcement/":
        continue

    # Extract title text
    title = a.get_text(strip=True)
    if title == "":
        continue

    # Construct full link URL
    full_link = BASE_URL + href

    # Avoid duplicate entries
    if full_link in seen:
        continue
    seen.append(full_link)

    # --------------------------------
    # Step 1: Locate date string by scanning forward in HTML
    # --------------------------------
    date = None
    element = a

    for _ in range(60):
        element = element.next_element
        if element is None:
            break

        if isinstance(element, str):
            txt = element.strip()

            if txt == "":
                continue

            # Identify date using month name + comma + numeric check
            if (txt.split(" ")[0] in months) and ("," in txt) and
             ↪  any(ch.isdigit() for ch in txt):
                date = txt
                break

    # Skip non-action entries that do not contain valid dates
    if date is None:
        continue
```

```
    # ------------------------------
    # Step 2: Locate category text following date
    # ------------------------------
    category = None

    for _ in range(60):
        element = element.next_element
        if element is None:
            break

        if isinstance(element, str):
            txt = element.strip()

            if txt == "":
                continue

            # Remove decorative separators or symbols
            if txt in ["·", "|"]:
                continue

            # Skip if encountering another date-like string
            if (txt.split(" ")[0] in months) and ("," in txt) and
             ↪  any(ch.isdigit() for ch in txt):
                continue

            # Assign first valid text as category label
            category = txt
            break

    # Store scraped record
    rows.append({
        "title": title,
        "date": date,
        "category": category,
        "link": full_link
    })

    # Limit output to first 20 enforcement actions for Step 1
    if len(rows) >= 20:
        break

# Convert results into DataFrame
```

```
df = pd.DataFrame(rows)

print(df.head(10))
print(df.shape)
```

```
                                               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
5  Attorney General Hanaway Obtains Medicaid Frau...  February 3, 2026
6  AG's Office Secures Indictments Against Peabod...  February 2, 2026
7  Florida Man Pleads Guilty to Conspiracy to Vio...  January 30, 2026
8  Forefront Living Hospice Agreed to Pay $1.9 Mi...  January 30, 2026
9  Attorney General Jeff Jackson Announces Health...  January 30, 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
5  State Enforcement Agencies
6  State Enforcement Agencies
7  Criminal and Civil Actions
8      Fraud Self-Disclosures
9  State Enforcement Agencies

                                                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...
5  https://oig.hhs.gov/fraud/enforcement/attorney...
6  https://oig.hhs.gov/fraud/enforcement/ags-offi...
7  https://oig.hhs.gov/fraud/enforcement/florida-...
8  https://oig.hhs.gov/fraud/enforcement/forefron...
9  https://oig.hhs.gov/fraud/enforcement/attorney...
(20, 4)
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

1. Check indicator
   - If run_scraper is False, print a message and stop.
2. Validate input
   - If start_year < 2013, print reminder and stop.
3. Initialize variables
   - Create empty list to store results
   - Create list/set to track seen links
   - Convert start year-month into datetime threshold
   - Set page number to 1
4. Crawl pages using a while loop
   - A simple for loop is not sufficient because the number of pages is unknown.
   - Continue crawling until stopping conditions are met.

   For each page:

   a. Construct page URL

   b. Download and parse HTML

   c. Find candidate enforcement links

   d. For each link:
      - Extract title
      - Scan forward to extract date
      - Skip entries without valid date
      - Convert date string to datetime
      - Extract category text
      - If date >= threshold, store row
      - If date < threshold, stop crawling

e. Stop crawling if:

- No valid rows found
- Old date threshold reached

f. Move to next page and wait 1 second

5. Convert results into DataFrame
6. Save CSV file enforcement_actions_year_month.csv

- b. Create Dynamic Scraper

```
from datetime import datetime

BASE_URL = "https://oig.hhs.gov"
START_URL = "https://oig.hhs.gov/fraud/enforcement/"

myheader = {
    "User-Agent": "DAP30538CourseBot/1.0 (your_email@uchicago.edu)"
}

months = ["January","February","March","April","May","June","July",
          "August","September","October","November","December"]


def scrape_enforcement_actions(start_year, start_month, run_scraper=False):

    # -------------------------------
    # Indicator (required by pset)
    # -------------------------------
    if run_scraper == False:
        print("Scraper NOT run. Turn indicator on to create CSV.")
        return None

    # -------------------------------
    # Check year restriction
    # -------------------------------
    if start_year < 2013:
        print("Please restrict to year >= 2013.")
        return None

    # Convert start date threshold
    start_dt = datetime(start_year, start_month, 1)
```

```
rows = []
seen_links = []

page = 1
stop_crawling = False

# ------------------------------
# While loop for unknown pages
# ------------------------------
while stop_crawling == False:

    # Build page URL
    if page == 1:
        url = START_URL
    else:
        url = START_URL + f"?page={page}"

    response = requests.get(url, headers=myheader)
    soup = BeautifulSoup(response.content, "lxml")

    all_a = soup.find_all("a")
    page_rows = 0

    for a in all_a:

        href = a.get("href")

        # Keep enforcement links only
        if href is None:
            continue
        if not href.startswith("/fraud/enforcement/"):
            continue
        if href == "/fraud/enforcement/":
            continue

        title = a.get_text(strip=True)
        if title == "":
            continue

        full_link = BASE_URL + href

        if full_link in seen_links:
```

```
            continue
        seen_links.append(full_link)

        # ------------------------------
        # Find date
        # ------------------------------
        date_str = None
        element = a

        for _ in range(60):
            element = element.next_element
            if element is None:
                break

            if isinstance(element, str):
                txt = element.strip()
                if txt == "":
                    continue

                if (txt.split(" ")[0] in months) and ("," in txt) and
                 ↪  any(ch.isdigit() for ch in txt):
                    date_str = txt
                    break

        if date_str is None:
            continue

        # Convert to datetime
        date_dt = datetime.strptime(date_str, "%B %d, %Y")

        # ------------------------------
        # Find category
        # ------------------------------
        category = None
        for _ in range(60):
            element = element.next_element
            if element is None:
                break

            if isinstance(element, str):
                txt = element.strip()
                if txt == "":
                    continue
```

```
                    if (txt.split(" ")[0] in months) and ("," in txt) and
                     ↪  any(ch.isdigit() for ch in txt):
                        continue

                    category = txt
                    break

            # ------------------------------
            # Stop condition
            # ------------------------------
            if date_dt < start_dt:
                stop_crawling = True
                continue

            rows.append({
                "title": title,
                "date": date_str,
                "category": category,
                "link": full_link
            })

            page_rows += 1

        # If no rows found, stop
        if page_rows == 0:
            break

        # Move to next page
        page += 1
        time.sleep(1)

    df = pd.DataFrame(rows)

    # Save CSV
    filename = f"enforcement_actions_{start_year}_{start_month}.csv"

    df.to_csv(filename, index=False)

    print(f"CSV saved: {filename}")
    return df

RUN_SCRAPER = False
```

```
#df_2024 = scrape_enforcement_actions(2024, 1, RUN_SCRAPER)

df_2024 = pd.read_csv('enforcement_actions_2024_1.csv')

# 1) How many enforcement actions?
n_actions = df_2024.shape[0]

# 2) Earliest enforcement action (by date)
df_2024["date_dt"] = pd.to_datetime(df_2024["date"])
earliest_row = df_2024.loc[df_2024["date_dt"].idxmin()]

earliest_date = earliest_row["date"]
earliest_title = earliest_row["title"]
earliest_category = earliest_row["category"]
earliest_link = earliest_row["link"]

print(f"How many enforcement actions in final dataframe? {n_actions}")
print("Earliest enforcement action scraped:")
print(f"  Date: {earliest_date}")
print(f"  Title: {earliest_title}")
print(f"  Category: {earliest_category}")
print(f"  Link: {earliest_link}")
```

```
How many enforcement actions in final dataframe? 1787
Earliest enforcement action scraped:
  Date: January 3, 2024
  Title: Laredo Resident Admits To Impersonating Licensed Nurse
  Category: Criminal and Civil Actions
  Link:
  https://oig.hhs.gov/fraud/enforcement/laredo-resident-admits-to-impersonating-licensed-nurs
```

I scraped 1,787 enforcement actions starting from January 2024.

The earliest enforcement action in the dataframe is dated January 3, 2024, titled "Laredo Resident Admits To Impersonating Licensed Nurse." This action is categorized under Criminal and Civil Actions, and the details can be found at: https://oig.hhs.gov/fraud/enforcement/laredo-resident-admits-to-impersonating-licensed-nurse/

- c. Test Your Code

```
RUN_SCRAPER = False
```

```python
# df_2022 = scrape_enforcement_actions(2022, 1, RUN_SCRAPER)

df_2022 = pd.read_csv('enforcement_actions_2022_1.csv')

# 1) How many enforcement actions?
n_actions = df_2022.shape[0]

# 2) Earliest enforcement action (by date)
df_2022["date_dt"] = pd.to_datetime(df_2022["date"])
earliest_row = df_2022.loc[df_2022["date_dt"].idxmin()]

earliest_date = earliest_row["date"]
earliest_title = earliest_row["title"]
earliest_category = earliest_row["category"]
earliest_link = earliest_row["link"]

print(f"How many enforcement actions in final dataframe? {n_actions}")
print("Earliest enforcement action scraped:")
print(f"  Date: {earliest_date}")
print(f"  Title: {earliest_title}")
print(f"  Category: {earliest_category}")
print(f"  Link: {earliest_link}")
```

```
How many enforcement actions in final dataframe? 3377
Earliest enforcement action scraped:
  Date: January 4, 2022
  Title: Central Medical Systems, LLC, Alan Trent Harley And Joan Harley
  Agree To Pay $600K To Settle False Claims Act Liability
  Category: Criminal and Civil Actions
  Link:
  https://oig.hhs.gov/fraud/enforcement/central-medical-systems-llc-alan-trent-harley-and-joa
```

I scraped 3,377 enforcement actions starting from January 2022.

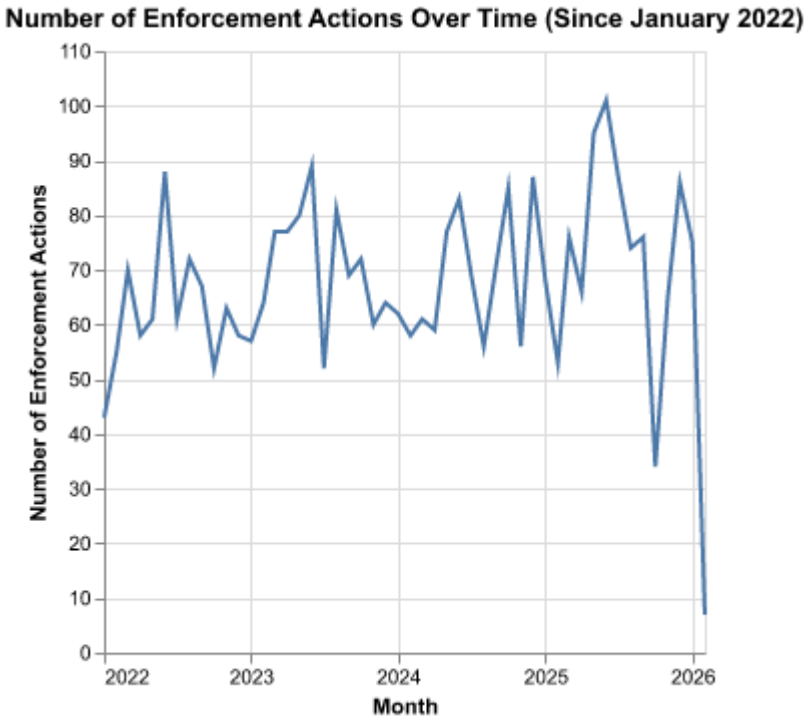
The earliest enforcement action in the dataframe is dated January 4, 2022, titled "Central Medical Systems, LLC, Alan Trent Harley And Joan Harley Agree To Pay $600K To Settle False Claims Act Liability." This action is categorized under Criminal and Civil Actions, and the details can be found at: https://oig.hhs.gov/fraud/enforcement/central-medical-systems-llc-alan-trent-harley-and-joan-harley-agree-to-pay-600k-to-settle-false-claims-act-liability/

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
# Convert date to datetime
df_2022["date"] = pd.to_datetime(df_2022["date"])

# Create month-year column
df_2022["year_month"] = df_2022["date"].dt.to_period("M").astype(str)

# Count enforcement actions per month
monthly_counts = (
    df_2022.groupby("year_month")
      .size()
      .reset_index(name="n_actions")
)

# Convert back to datetime for plotting
monthly_counts["year_month"] = pd.to_datetime(monthly_counts["year_month"])
```

```
chart1 = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X("year_month:T", title="Month"),
    y=alt.Y("n_actions:Q", title="Number of Enforcement Actions")
).properties(
    title="Number of Enforcement Actions Over Time (Since January 2022)"
)

chart1
```



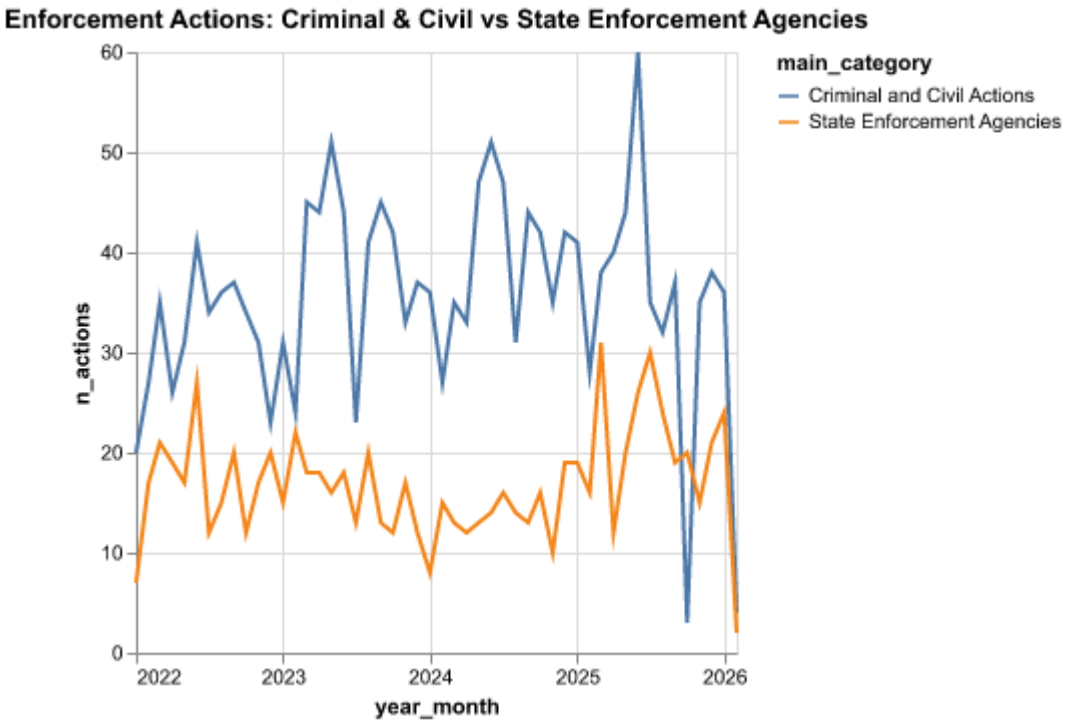
2. Plot the number of enforcement actions categorized:

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

```
df_2022["main_category"] = df_2022["category"].apply(
    lambda x: x if x in ["Criminal and Civil Actions", "State Enforcement
    ↪  Agencies"] else None
)

monthly_cat = (
    df_2022.groupby(["year_month", "main_category"])
      .size()
      .reset_index(name="n_actions")
)

monthly_cat["year_month"] = pd.to_datetime(monthly_cat["year_month"])
```

```
chart2 = alt.Chart(monthly_cat).mark_line().encode(
    x="year_month:T",
    y="n_actions:Q",
```

```
    color="main_category:N"
).properties(
    title="Enforcement Actions: Criminal & Civil vs State Enforcement
 ↪  Agencies"
)

chart2
```



- based on five topics

```
def classify_topic(title: str) -> str:
    t = (title or "").lower()

    # Drug Enforcement
    drug = [
        "opioid", "fentanyl", "oxycodone", "hydrocodone",
        "controlled substance", "drug", "pill"
    ]
    if any(k in t for k in drug):
        return "Drug Enforcement"
```

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

    # Financial Fraud
    financial = [
        "wire fraud", "bank", "loan", "credit", "mortgage",
        "securities", "investment", "crypto", "bitcoin"
    ]
    if any(k in t for k in financial):
        return "Financial Fraud"

    # Health Care Fraud
    health = [
        "medicare", "medicaid", "hospital", "clinic",
        "physician", "doctor", "nurse", "patient",
        "billing", "claims", "health care", "healthcare"
    ]
    if any(k in t for k in health):
        return "Health Care Fraud"

    return "Other"
```

```
df_criminal = df_2022[df_2022["category"] == "Criminal and Civil
 ↪  Actions"].copy()

df_criminal["topic"] = df_criminal["title"].apply(classify_topic)

monthly_topic = (
    df_criminal.groupby(["year_month", "topic"])
    .size()
    .reset_index(name="n_actions")
)

monthly_topic["year_month"] = pd.to_datetime(monthly_topic["year_month"])
```

```
chart3 = alt.Chart(monthly_topic).mark_line().encode(
    x="year_month:T",
    y="n_actions:Q",
    color="topic:N"
).properties(
    title="Criminal and Civil Actions by Topic"
)

chart3
```