PS4 Answer

Andi Fang

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: **AF**

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")
```

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

Step 1: Develop initial scraper and crawler

```python
import requests
from bs4 import BeautifulSoup

url = "https://oig.hhs.gov/fraud/enforcement/"
resp = requests.get(url)

soup = BeautifulSoup(resp.text, "lxml")

rows = []
li_tags = soup.find_all("li", class_="usa-card")

for li in li_tags:
    h2 = li.find("h2", class_="usa-card__heading")
    if h2 is None:
        continue

    a = h2.find("a")
    if a is None:
        continue

    title = a.text
    link = "https://oig.hhs.gov" + a["href"]

    date_span = li.find("span", class_="text-base-dark")
    if date_span is None:
        continue

    date = date_span.text
    category_li = li.find("li", class_="usa-tag")
    if category_li is None:
        continue

    category = category_li.text

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

df = pd.DataFrame(rows)
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

FUNCTION scrape_enforcement_actions(year, month, run_scraper):

```
IF year < 2013:
    PRINT warning message
    RETURN None

IF run_scraper is False:
    PRINT message saying scraper is turned off
    RETURN None

Initialize empty list rows
Set page = 0
Set cutoff_date = first day of (year, month)

WHILE True:
    Construct URL with current page number
    Request page HTML
    Parse HTML with BeautifulSoup

    Find all <li class="usa-card"> on page
    IF no cards found:
        BREAK loop   (no more pages)

    FOR each card:
        Extract title, link, date, category
        Convert date string to datetime

        IF date < cutoff_date:
            STOP scraping entirely (break out of loops)

        Append record to rows

    Sleep for 1 second
    Increment page by 1

Convert rows to DataFrame
Save DataFrame to CSV file
RETURN DataFrame
```

- b. Create Dynamic Scraper

```python
from datetime import datetime

def scrape_enforcement_actions(year, month, run_scraper=True):
```

```python
if year < 2013:
    print("Please restrict to year >= 2013. Enforcement actions are only
     ↪  available after 2013.")
    return None

if not run_scraper:
    print("Scraper is turned off. No scraping performed.")
    return None

cutoff_date = datetime(year, month, 1)
rows = []
page = 0

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

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

    if not li_tags:
        break

    stop_scraping = False

    for li in li_tags:
        h2 = li.find("h2", class_="usa-card__heading")
        if h2 is None:
            continue

        a = h2.find("a")
        if a is None:
            continue

        title = a.text
        link = "https://oig.hhs.gov" + a["href"]

        date_span = li.find("span", class_="text-base-dark")
        if date_span is None:
            continue

        date_str = date_span.text
```

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

                if date < cutoff_date:
                    stop_scraping = True
                    break

                category_li = li.find("li", class_="usa-tag")
                if category_li is None:
                    continue

                category = category_li.text

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

            if stop_scraping:
                break

            page += 1
            time.sleep(1)

    df = pd.DataFrame(rows)

    filename = f"enforcement_actions_{year}_{month}.csv"
    df.to_csv(filename, index=False)

    return df

df_2024 = scrape_enforcement_actions(2024, 1, run_scraper=False)

# print("Total number of enforcement actions:", len(df_2024))

# earliest_row = df_2024.sort_values("date").head(1).iloc[0]
# print("\nEarliest enforcement action:")
# print("Date:", earliest_row["date"])
# print("Title:", earliest_row["title"])
# print("Category:", earliest_row["category"])
# print("Link:", earliest_row["link"])
```

```
Scraper is turned off. No scraping performed.
```

Total number of enforcement actions: 1807

Earliest enforcement action:

Date: 2024-01-03 00:00:00

Title: Former Nurse Aide Indicted In Death Of Clarksville Patient Arrested In Georgia

Category: State Enforcement Agencies

Link: https://oig.hhs.gov/fraud/enforcement/former-nurse-aide-indicted-in-death-of-clarksville-patient-arrested-in-georgia/

- c. Test Your Code

```python
df_2022 = scrape_enforcement_actions(2022, 1, run_scraper=False)

# print("Total number of enforcement actions:", len(df_2022))

# earliest_row = df_2022.sort_values("date").head(1).iloc[0]
# print("\nEarliest enforcement action:")
# print("Date:", earliest_row["date"])
# print("Title:", earliest_row["title"])
# print("Category:", earliest_row["category"])
# print("Link:", earliest_row["link"])
```

```
Scraper is turned off. No scraping performed.
```

Total number of enforcement actions: 3397

Earliest enforcement action:

Date: 2022-01-04 00:00:00

Title: Integrated Pain Management Medical Group Agreed to Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded Individuals

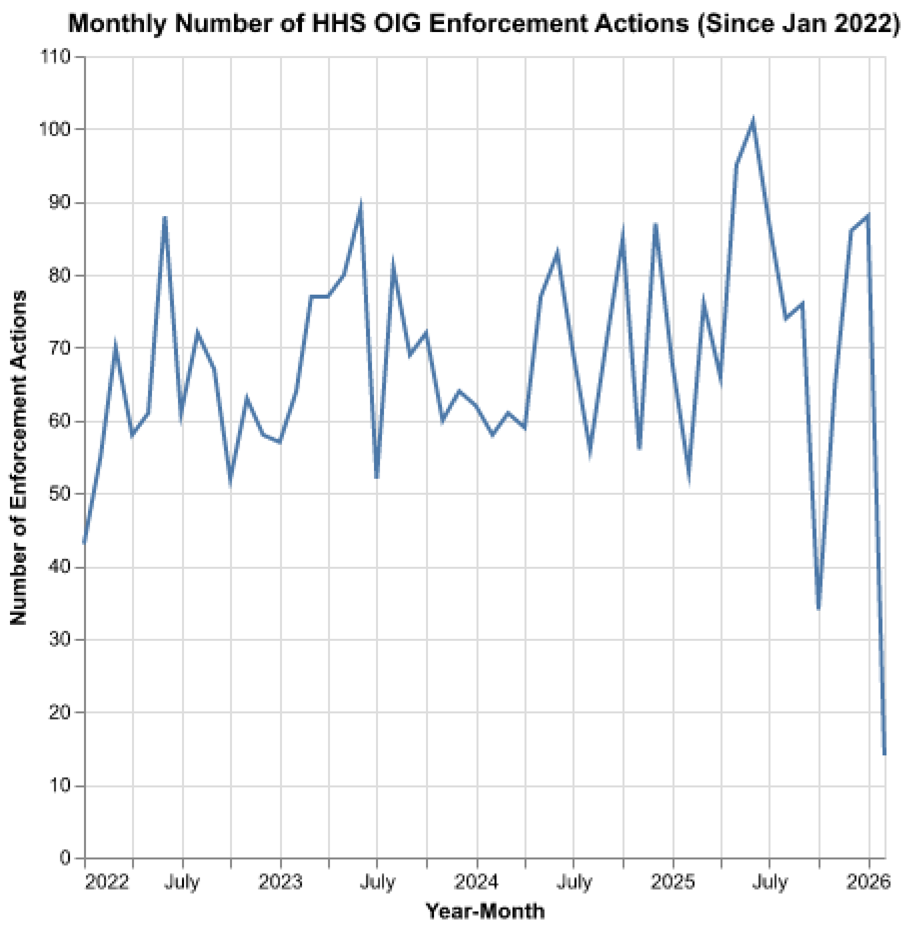
Category: Fraud Self-Disclosures

Link: https://oig.hhs.gov/fraud/enforcement/integrated-pain-management-medical-group-agreed-to-pay-10000-for-allegedly-violating-the-civil-monetary-penalties-law-by-employing-excluded-individuals/

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

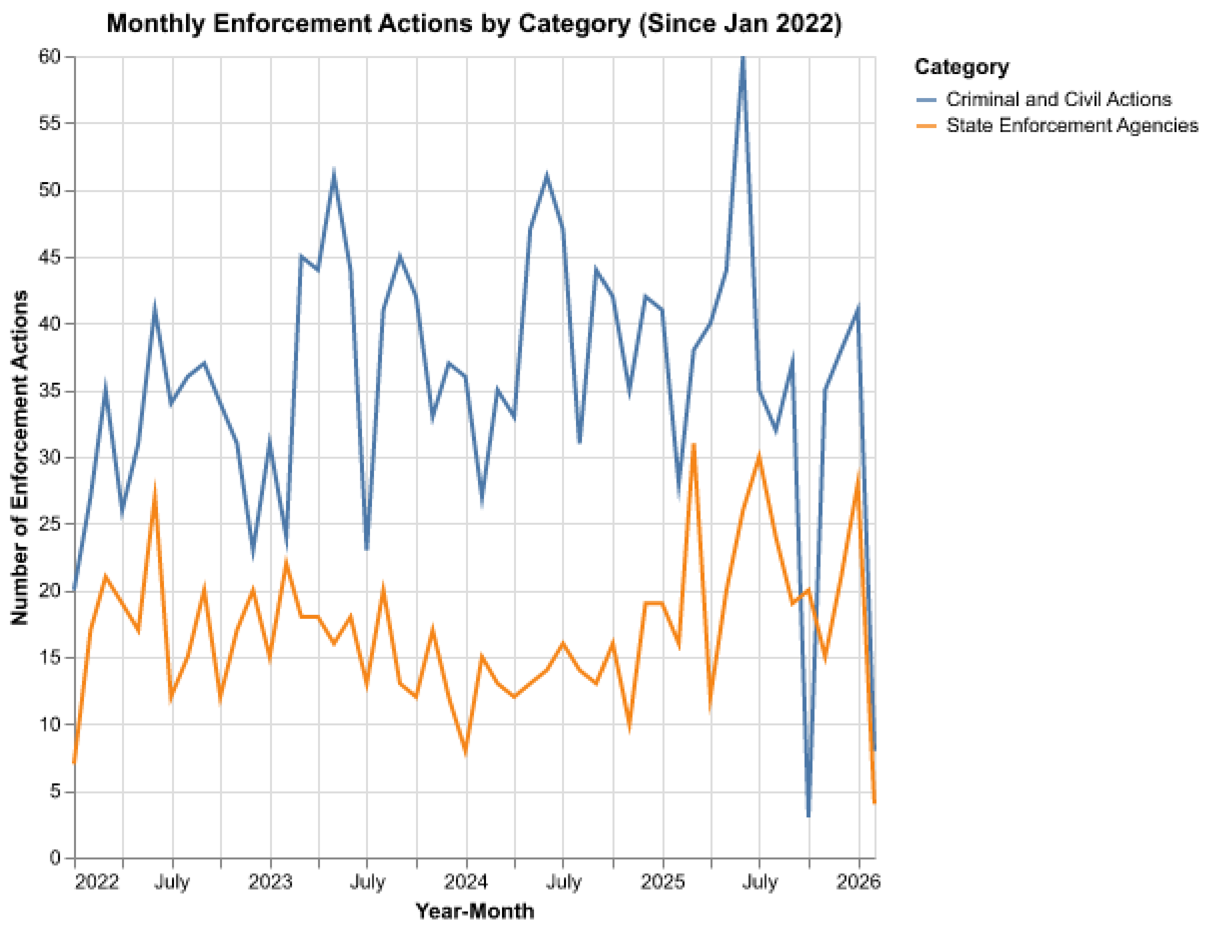
```
df = pd.read_csv("enforcement_actions_2022_1.csv")
df["date"] = pd.to_datetime(df["date"])
df["year_month"] = df["date"].dt.to_period("M").dt.to_timestamp()

monthly_counts = (
    df.groupby("year_month")
      .size()
      .reset_index(name="count")
)

line_overall = (
    alt.Chart(monthly_counts)
    .mark_line()
    .encode(
        alt.X("year_month:T", title="Year-Month"),
        alt.Y("count:Q", title="Number of Enforcement Actions"),
        tooltip=["year_month:T", "count:Q"]
    )
    .properties(
        title="Monthly Number of HHS OIG Enforcement Actions (Since Jan
 ↪  2022)",
        width=400,
        height=400
    )
)

line_overall
```



2. Plot the number of enforcement actions categorized:

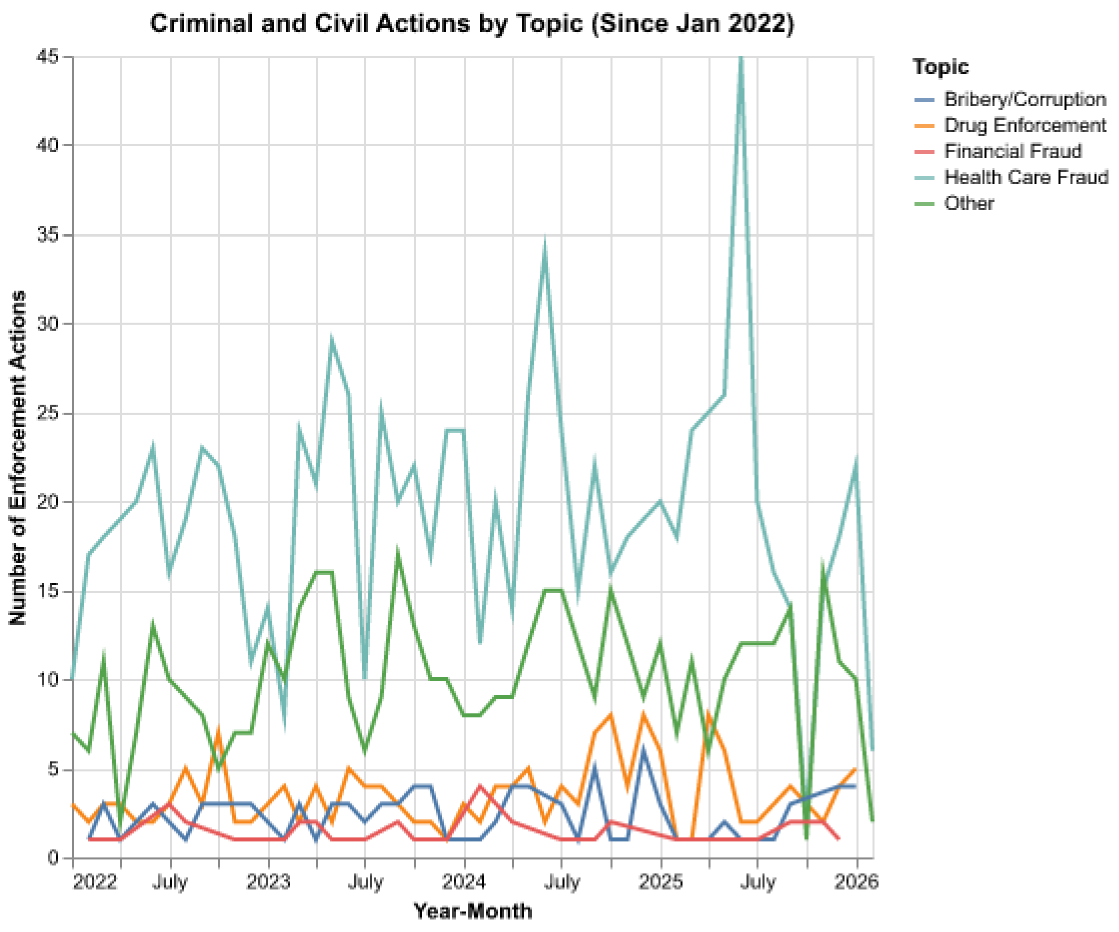
- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"

```
category_counts = (
    df[df["category"].isin([
        "Criminal and Civil Actions",
        "State Enforcement Agencies"
    ])]
    .groupby(["year_month", "category"])
    .size()
    .reset_index(name="count")
)

line_category = (
    alt.Chart(category_counts)
    .mark_line()
    .encode(
        alt.X("year_month:T", title="Year-Month"),
        alt.Y("count:Q", title="Number of Enforcement Actions"),
        color=alt.Color("category:N", title="Category"),
        tooltip=["year_month:T", "category:N", "count:Q"]
    )
    .properties(
        title="Monthly Enforcement Actions by Category (Since Jan 2022)",
        width=400,
        height=400
    )
)

line_category
```



- based on five topics

```python
cca = df[df["category"] == "Criminal and Civil Actions"]

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

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

topic_counts = (
    cca.groupby(["year_month", "topic"])
       .size()
       .reset_index(name="count")
)

line_topics = (
    alt.Chart(topic_counts)
    .mark_line()
    .encode(
        alt.X("year_month:T", title="Year-Month"),
        alt.Y("count:Q", title="Number of Enforcement Actions"),
        color=alt.Color("topic:N", title="Topic"),
        tooltip=["year_month:T", "topic:N", "count:Q"]
    )
    .properties(
        title="Criminal and Civil Actions by Topic (Since Jan 2022)",
        width=400,
        height=400
    )
)

line_topics
```


Criminal and Civil Actions by Topic (Since Jan 2022)
Number of Enforcement Actions
45
40
35
30
25
20
15
10
5
0
2022
July
2023
July
2024
July
2025
July
2026
Year-Month
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other