PS4

Yiduo Pan

2026-02-07

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: **YP**

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, re
import pandas as pd
from bs4 import BeautifulSoup
from urllib.parse import urljoin

base = "https://oig.hhs.gov"
url = base + "/fraud/enforcement/"

res = requests.get(url, headers={"User-Agent": "Mozilla/5.0"})
soup = BeautifulSoup(res.text, "html.parser")

all_a = soup.select('a[href^="/fraud/enforcement/"]')
all_a = [a for a in all_a if a.get_text(strip=True)]

exclude_prefixes = (
    "/fraud/enforcement/about/",
    "/fraud/enforcement/civil-monetary-penalty-authorities/",
)
action_a = []
for a in all_a:
    href = a.get("href", "")
    title = a.get_text(" ", strip=True)
    if href.startswith(exclude_prefixes):
        continue
    if len(title) < 15:
        continue
    if href == "/fraud/enforcement/":
        continue
    action_a.append(a)
```

```
seen = set()
uniq = []
for a in action_a:
    href = a.get("href")
    if href not in seen:
        seen.add(href)
        uniq.append(a)

rows = []
date_re = re.compile(

     ↪  r"(January|February|March|April|May|June|July|August|September|October|November|Decem
)

for a in uniq:
    title = a.get_text(" ", strip=True)
    link = urljoin(base, a.get("href"))

    block = a.find_parent(["li", "article", "div"])
    if block:
        txt = block.get_text(" ", strip=True)
        if len(txt) < 40:
            block2 = block.find_parent(["li", "article", "div"])
            if block2:
                block = block2
                txt = block.get_text(" ", strip=True)
    else:
        txt = ""

    # date
    m = date_re.search(txt)
    date = m.group(0) if m else None

    # category
    cat = None
    if block:
        tag = block.select_one('[class*="tag"], [class*="badge"],
 ↪  [class*="category"]')
        if tag:
            cat = tag.get_text(" ", strip=True)

    if cat is None:
        for c in [
            "Criminal and Civil Actions",
            "State Enforcement Agencies",
            "COVID-19",
            "Child Support",
            "CIA Reportable Events",
            "CMP and Affirmative Exclusions",
            "EMTALA/Patient Dumping",
            "Fraud Self-Disclosures",
            "Grant and Contractor Fraud Self-Disclosures",
            "Stipulated Penalties and Material Breaches",
        ]:
            if c in txt:
                cat = c
                break

    rows.append([title, date, cat, link])

df = pd.DataFrame(rows, columns=["title", "date", "category", "link"])
print(df.head(10))
print("n_rows:", len(df))
```

```
                                               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...
n_rows: 20
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code FUNCTION scrape_enforcement_actions(start_year, start_month, run_indicator):

  IF run_indicator is FALSE: LOAD "enforcement_actions_year_month.csv" RETURN dataframe

  IF start_year < 2013: PRINT "Please restrict to year >= 2013." RETURN empty dataframe

  INITIALIZE: page_number = 1 all_records = empty list start_cutoff_date = first day of (start_year, start_month)

  WHILE True:

```
  IF page_number == 1:
      url = base enforcement page
  ELSE:
      url = base page + "?page=" + page_number

  SEND HTTP request to url
  IF response not successful:
      BREAK loop

  PARSE HTML page:
      EXTRACT for each enforcement action:
```

```
                - title
                - date
                - category
                - link

        IF no actions found:
            BREAK loop

        ADD extracted records to all_records

        FIND earliest date on current page
        IF earliest date earlier than start_cutoff_date:
            BREAK loop

        page_number += 1
        WAIT 1 second (to avoid server blocking)
```

CONVERT all_records into dataframe

FILTER dataframe: KEEP only records with date >= start_cutoff_date

SAVE dataframe as: "enforcement_actions_year_month.csv" (do not commit this file to Git)

RETURN dataframe

- b. Create Dynamic Scraper

```python
import time
from datetime import datetime

def scrape_enforcement_actions(start_year, start_month, run_indicator=True):

    if not run_indicator:
        return pd.read_csv("enforcement_actions_year_month.csv")

    if start_year < 2013:
        print("Please restrict to year >= 2013.")
        return pd.DataFrame()

    base = "https://oig.hhs.gov"
    start_cutoff = datetime(start_year, start_month, 1)

    all_rows = []
    page = 1
```

```
date_re = re.compile(
    ↪ r"(January|February|March|April|May|June|July|August|September|October|November|I
)

while True:

    url = base + "/fraud/enforcement/" if page == 1 else base +
↪   f"/fraud/enforcement/?page={page}"
    res = requests.get(url, headers={"User-Agent":"Mozilla/5.0"})

    if res.status_code != 200:
        break

    soup = BeautifulSoup(res.text, "html.parser")

    links = soup.select('a[href^="/fraud/enforcement/"]')
    links = [a for a in links if len(a.get_text(strip=True)) > 15]

    if not links:
        break

    for a in links:
        title = a.get_text(strip=True)
        link = urljoin(base, a["href"])

        block = a.find_parent(["li","article","div"])
        txt = block.get_text(" ",strip=True) if block else ""

        m = date_re.search(txt)
        date = m.group(0) if m else None

        category = None
        for c in ["Criminal and Civil Actions","State Enforcement
        ↪ Agencies","COVID-19"]:
            if c in txt:
                category = c

        all_rows.append([title,date,category,link])

    earliest = pd.to_datetime(date,format="%B %d, %Y",errors="coerce")
    if pd.notna(earliest) and earliest < start_cutoff:
            break

        page += 1
        time.sleep(1)

    df = pd.DataFrame(all_rows,columns=["title","date","category","link"])
    df["date_dt"] = pd.to_datetime(df["date"],format="%B %d,
↪   %Y",errors="coerce")
    df = df[df["date_dt"]>=start_cutoff]

    df.to_csv("enforcement_actions_year_month.csv",index=False)
    return df

df_2024 = scrape_enforcement_actions(2024,1,True)
print(len(df_2024))
print(df_2024.sort_values("date_dt").iloc[0])
```

```
1787
title       Former Nurse Aide Indicted In Death Of Clarksv...
date                                        January 3, 2024
category                          State Enforcement Agencies
link        https://oig.hhs.gov/fraud/enforcement/former-n...
date_dt                                 2024-01-03 00:00:00
Name: 2056, dtype: object
```

The final dataframe contains 1787 enforcement actions.
The earliest enforcement action is as follows:
title Former Nurse Aide Indicted In Death Of Clarksv...
date January 3, 2024
category State Enforcement Agencies
link https://oig.hhs.gov/fraud/enforcement/former-n...

- c. Test Your Code

```
df_2022 = scrape_enforcement_actions(2022,1,True)
print(len(df_2022))
print(df_2022.sort_values("date_dt").iloc[0])
```

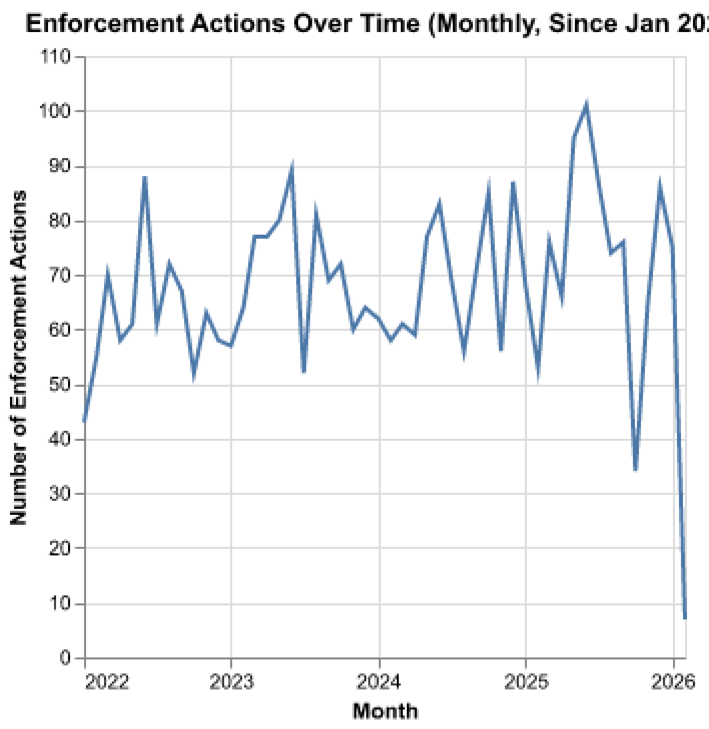
```
3377
title       Integrated Pain Management Medical Group Agree...
date                                        January 4, 2022
category                                               None
link        https://oig.hhs.gov/fraud/enforcement/integrat...
```

```
date_dt                                   2022-01-04 00:00:00
Name: 3883, dtype: object
```

The final dataframe contains 3377 enforcement actions.
The earliest enforcement action is as follows:
title Integrated Pain Management Medical Group Agree… date January 4, 2022 category None link https://oig.hhs.gov/fraud/enforcement/integrat… ## Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

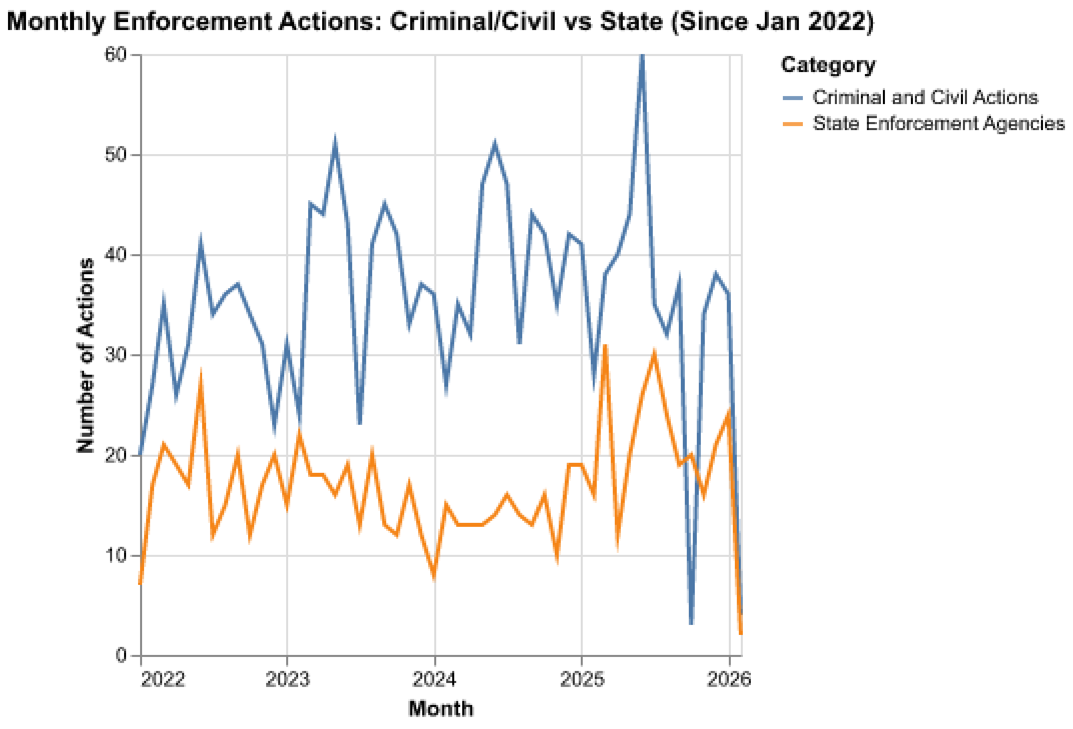
```python
df = pd.read_csv("enforcement_actions_year_month.csv")

df["date_dt"] = pd.to_datetime(df["date"], errors="coerce")

df = df[df["date_dt"] >= "2022-01-01"].copy()

df["year_month"] = df["date_dt"].dt.to_period("M").dt.to_timestamp()

monthly_all = (
    df.groupby("year_month")
      .size()
      .reset_index(name="n_actions")
)

chart1 = (
    alt.Chart(monthly_all)
    .mark_line()
    .encode(
        x=alt.X("year_month:T", title="Month"),
        y=alt.Y("n_actions:Q", title="Number of Enforcement Actions"),
        tooltip=["year_month:T", "n_actions:Q"]
    )
    .properties(title="Enforcement Actions Over Time (Monthly, Since Jan
 ↪  2022)")
)

chart1
```



2. Plot the number of enforcement actions categorized:

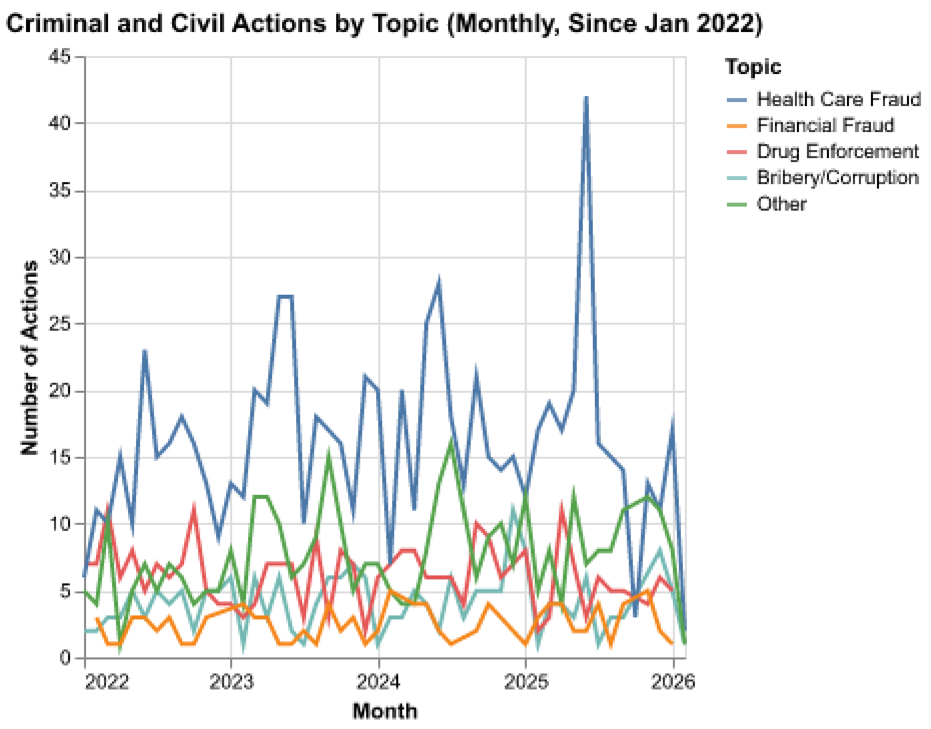
- based on “Criminal and Civil Actions” vs. “State Enforcement Agencies”

```
two_cats = ["Criminal and Civil Actions", "State Enforcement Agencies"]

monthly_two = (
    df[df["category"].isin(two_cats)]
    .groupby(["year_month", "category"])
    .size()
    .reset_index(name="n_actions")
)

chart2a = (
    alt.Chart(monthly_two)
    .mark_line()
    .encode(
        x=alt.X("year_month:T", title="Month"),
        y=alt.Y("n_actions:Q", title="Number of Actions"),
        color=alt.Color("category:N", title="Category"),
        tooltip=["year_month:T", "category:N", "n_actions:Q"]
    )
    .properties(title="Monthly Enforcement Actions: Criminal/Civil vs State
↪   (Since Jan 2022)")
)

chart2a
```



- based on five topics

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

    # Financial Fraud
    if any(k in t for k in ["bank", "financial", "launder", "wire fraud",
    ↪   "money laundering", "tax", "securities", "investor"]):
        return "Financial Fraud"

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

    # Bribery/Corruption
    if any(k in t for k in ["brib", "kickback", "corrupt", "extortion",
     ↪  "racketeer", "racket"]):
        return "Bribery/Corruption"

    # Health Care Fraud
    if any(k in t for k in ["medicare", "medicaid", "health care",
     ↪  "healthcare", "physician", "doctor", "hospital", "clinic", "hospice",
     ↪  "nursing", "home health"]):
        return "Health Care Fraud"

    return "Other"


df_cc = df[df["category"] == "Criminal and Civil Actions"].copy()
df_cc["topic"] = df_cc["title"].apply(classify_topic)

monthly_topics = (
    df_cc.groupby(["year_month", "topic"])
         .size()
         .reset_index(name="n_actions")
)

topic_order = ["Health Care Fraud", "Financial Fraud", "Drug Enforcement",
 ↪  "Bribery/Corruption", "Other"]

chart2b = (
    alt.Chart(monthly_topics)
    .mark_line()
    .encode(
        x=alt.X("year_month:T", title="Month"),
        y=alt.Y("n_actions:Q", title="Number of Actions"),
        color=alt.Color("topic:N", sort=topic_order, title="Topic"),
        tooltip=["year_month:T", "topic:N", "n_actions:Q"]
    )
    .properties(title="Criminal and Civil Actions by Topic (Monthly, Since
 ↪  Jan 2022)")
)

chart2b
```


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