PS4

JIUZHOU XIA

2026-02-04

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: **JX**

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 re
import time, random, requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse, parse_qs
from urllib.parse import urljoin

BASE_URL = "https://oig.hhs.gov/fraud/enforcement/"
HEADERS = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"}
def get_soup(url):
    r = requests.get(url, headers=HEADERS, timeout=30)
    r.raise_for_status()
    return BeautifulSoup(r.text, "html.parser")


def scrape_first_page(base_url=BASE_URL):
    soup = get_soup(base_url)
    rows = []
    for card in soup.select("li.usa-card"):
        heading = card.select_one("h2.usa-card__heading a")
        if not heading:
            continue
        date_span = card.select_one("div.font-body-sm span.text-base-dark")
        tags = card.select("li.usa-tag")
        rows.append({
            "title": heading.get_text(strip=True),
            "date": date_span.get_text(strip=True) if date_span else "",
            "category": tags[0].get_text(strip=True) if tags else "",
            "link": urljoin(base_url, heading.get("href", ""))
        })
    return pd.DataFrame(rows)


df_1 = scrape_first_page()
df_1.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 scrape_enforcement_actions(year, month):
BEGIN
    IF year < 2013 THEN
        print some notification
        RETURN empty result
    ENDIF
    target_date <- (year, month)
    rows <- empty list
    page <- 1
    WHILE TRUE DO
        url <- base_url + "?page=" + page
        soup <- fetch and parse url
        cards <- all card elements from current page
        IF cards is empty THEN
            BREAK
        ENDIF
        FOR each card IN cards DO
            title <- extract title from card
            date <- extract date from card
            category <- extract category from card
            link <- extract link from card
            IF parsed(date) < target_date THEN
                save rows to enforcement_actions_year_month.csv
                RETURN rows as dataframe
            ENDIF
            append (title, date, category, link) to rows
        ENDFOR
        page <- page + 1
    ENDWHILE
    save rows to enforcement_actions_year_month.csv
    RETURN rows as dataframe
END
ENDFUNCTION
```

Loop Choosing Using `while True` rather than `for page in range(1, N)` because we have too much pages and we need to stop it by date. So we can't predict how many pages we need to scrape.

- b. Create Dynamic Scraper

```python
from datetime import datetime

def parse_date(date_str):
    try:
        dt = datetime.strptime(date_str.strip(), "%B %d, %Y")
        return (dt.year, dt.month)
    except ValueError:
        return (9999, 99)


def scrape_enforcement_actions(year, month, base_url=BASE_URL):
    if year < 2013:
        print("early")
        return pd.DataFrame()
    target = (year, month)
    rows = []
    page = 1
    while True:
        url = base_url if page == 1 else f"{base_url}?page={page}"
        try:
            soup = get_soup(url)
        except Exception:
            break
        cards = soup.select("li.usa-card")
        if not cards:
            break
        for card in cards:
            heading = card.select_one("h2.usa-card__heading a")
            if not heading:
                continue
            date_span = card.select_one("div.font-body-sm
↪   span.text-base-dark")
            date_str = date_span.get_text(strip=True) if date_span else ""
            tags = card.select("li.usa-tag")
            category = tags[0].get_text(strip=True) if tags else ""
            if parse_date(date_str) < target:
                df = pd.DataFrame(rows)
                df.to_csv("enforcement_actions_year_month.csv", index=False,
↪   encoding="utf-8-sig")
                return df
            rows.append({
                "title": heading.get_text(strip=True),
                "date": date_str,
                "category": category,
                "link": urljoin(base_url, heading.get("href", ""))
            })
        time.sleep(1)
        page += 1
    df = pd.DataFrame(rows)
    df.to_csv("enforcement_actions_year_month.csv", index=False,
↪   encoding="utf-8-sig")
    return df

RUN_SCRAPER = True
if RUN_SCRAPER:
    scrape_enforcement_actions(2024, 1)
```

```
df2024 = pd.read_csv("enforcement_actions_year_month.csv",
↪   encoding="utf-8-sig")
print('Totally have', len(df2024), 'actions')
print(df2024.tail(1))
```

```
Totally have 1787 actions
                                                  title             date  \
1786  Former Nurse Aide Indicted In Death Of Clarksv...  January 3, 2024
```

```
                        category  \
1786  State Enforcement Agencies

                                                 link
1786  https://oig.hhs.gov/fraud/enforcement/former-n...
```

- c. Test Your Code

```
RUN_SCRAPER = True
if RUN_SCRAPER:
    scrape_enforcement_actions(2022, 1)
df2022 = pd.read_csv("enforcement_actions_year_month.csv",
 ↪  encoding="utf-8-sig")
print('Totally have', len(df2022), 'actions')
print(df2022.tail(1))
```

```
Totally have 3377 actions
                                                title             date  \
3376  Integrated Pain Management Medical Group Agree...  January 4, 2022

                    category  \
3376  Fraud Self-Disclosures

                                                 link
3376  https://oig.hhs.gov/fraud/enforcement/integrat...
```

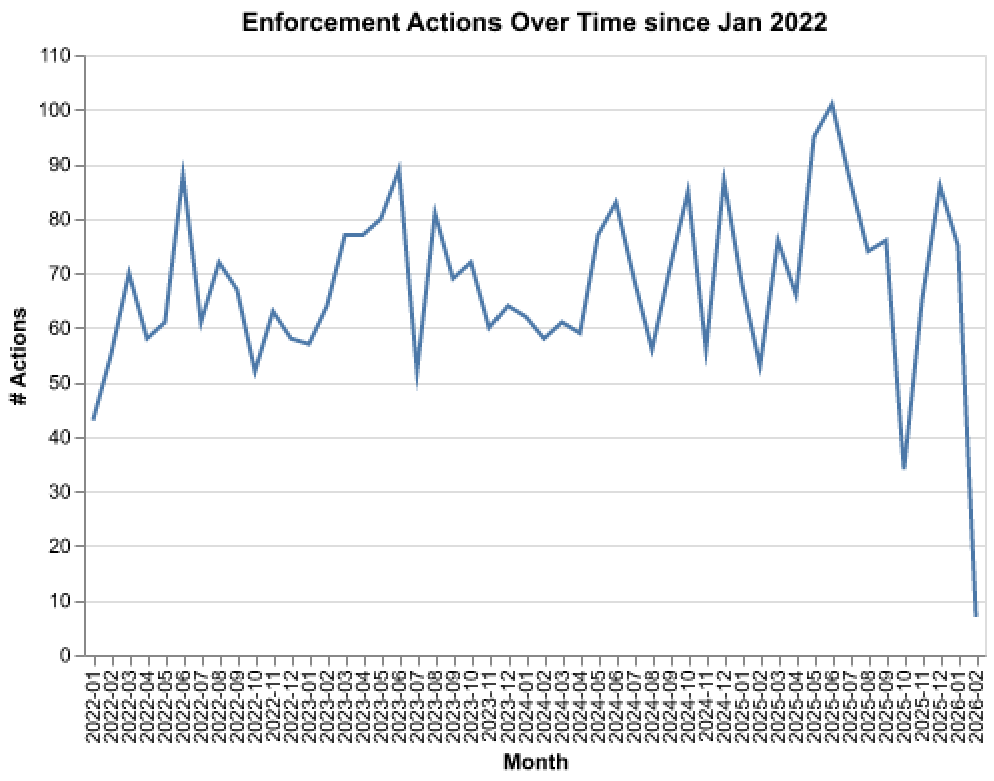
Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df = df2022.copy()
df['parsed_date'] = pd.to_datetime(df['date'], format='%B %d, %Y',
 ↪  errors='coerce')
df = df.dropna(subset=['parsed_date'])
df = df[df['parsed_date'] >= '2022-01-01']

monthly =
 ↪  df.groupby(df['parsed_date'].dt.to_period('M')).size().reset_index(name='count')
monthly['month'] = monthly['parsed_date'].astype(str)
monthly = monthly[['month', 'count']]
```

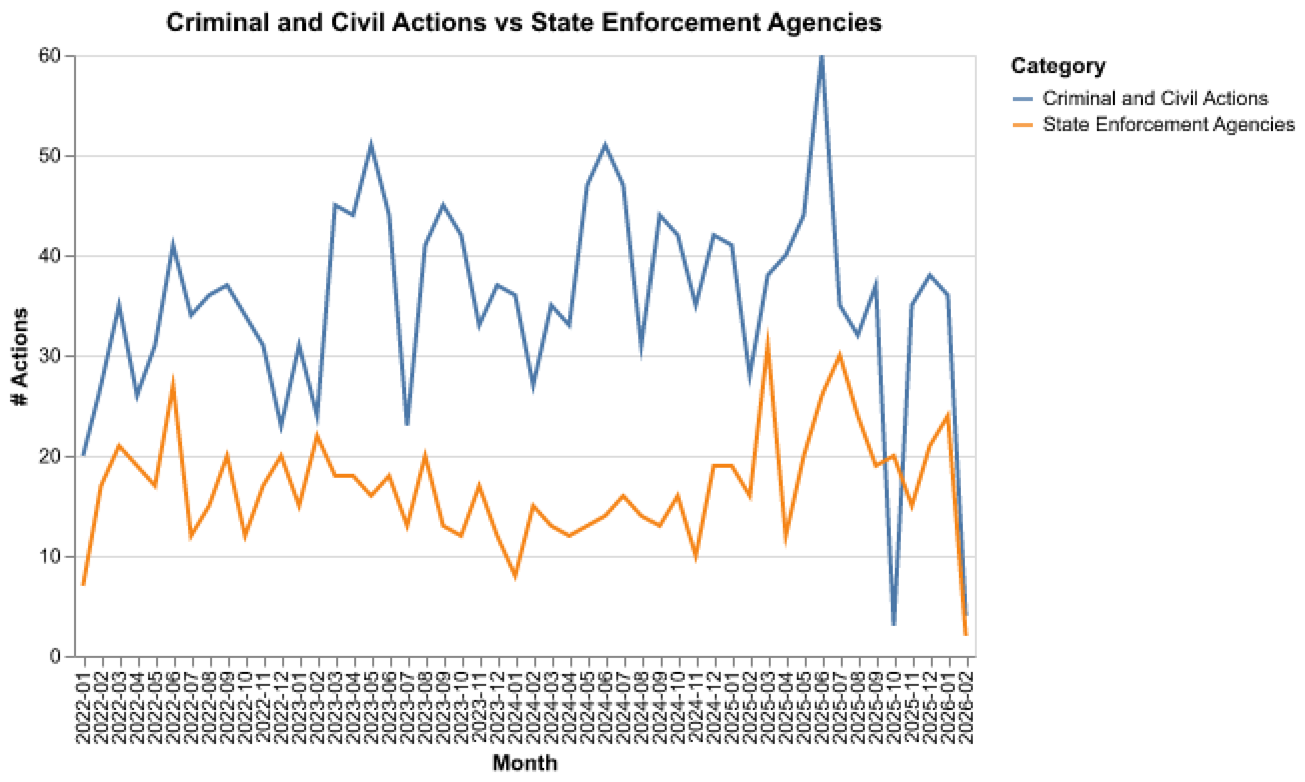
```
alt.Chart(monthly).mark_line().encode(
    x=alt.X('month:N', title='Month'),
    y=alt.Y('count:Q', title='# Actions')
).properties(title='Enforcement Actions Over Time since Jan 2022',width=450)
```



2. Plot the number of enforcement actions categorized:

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

```
sub = df[df['category'].isin(['Criminal and Civil Actions', 'State
↪  Enforcement Agencies'])]
sub = sub[sub['parsed_date'] >= '2022-01-01']
monthly_cat = sub.groupby([sub['parsed_date'].dt.to_period('M'),
↪  'category']).size().reset_index(name='count')
monthly_cat['month'] = monthly_cat['parsed_date'].astype(str)
monthly_cat = monthly_cat[['month', 'category', 'count']]
alt.Chart(monthly_cat).mark_line().encode(
    x=alt.X('month:N', title='Month'),
    y=alt.Y('count:Q', title='# Actions'),
    color=alt.Color('category:N', legend=alt.Legend(title='Category'))
).properties(title='Criminal and Civil Actions vs State Enforcement
↪  Agencies', width=450)
```

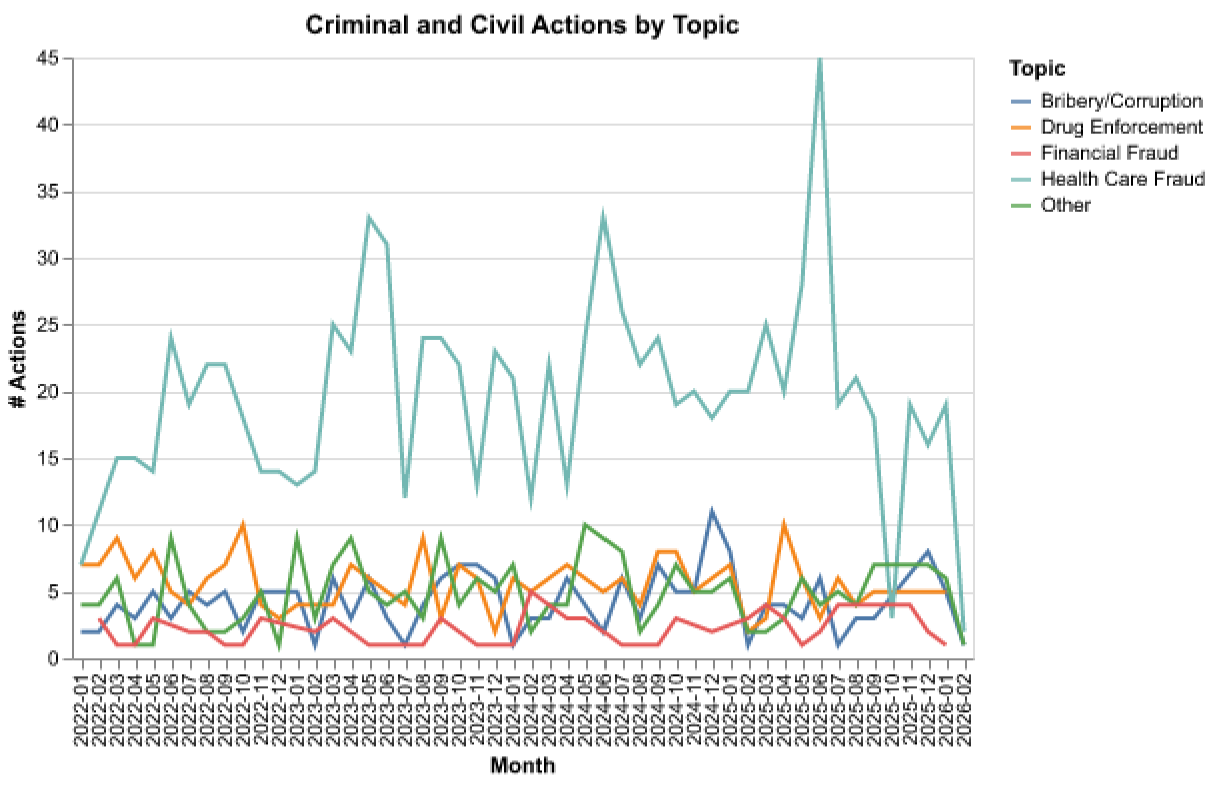


- based on five topics

```
def topicselec(title):
    t = (title or "").lower()
    if any(k in t for k in ['bank', 'financial', 'launder', 'wire fraud',
     ↪  'securities']):
        return 'Financial Fraud'
    if any(k in t for k in ['drug', 'pill mill', 'opioid', 'controlled
     ↪  substance', 'pharmacy']):
        return 'Drug Enforcement'
    if any(k in t for k in ['bribery', 'kickback', 'corruption', 'bribe']):
        return 'Bribery/Corruption'
    if any(k in t for k in ['medicare', 'medicaid', 'health care',
     ↪  'healthcare', 'fraud', 'false claim']):
        return 'Health Care Fraud'
    return 'Other'


cca = df[df['category'] == 'Criminal and Civil Actions'].copy()
cca = cca[cca['parsed_date'] >= '2022-01-01']
cca['topic'] = cca['title'].apply(topicselec)

monthly_topic = cca.groupby([cca['parsed_date'].dt.to_period('M'),
 ↪  'topic']).size().reset_index(name='count')
monthly_topic['month'] = monthly_topic['parsed_date'].astype(str)
monthly_topic = monthly_topic[['month', 'topic', 'count']]
alt.Chart(monthly_topic).mark_line().encode(
    x=alt.X('month:N', title='Month'),
    y=alt.Y('count:Q', title='# Actions'),
    color=alt.Color('topic:N', legend=alt.Legend(title='Topic'))
).properties(title='Criminal and Civil Actions by Topic', width=450)
```


Criminal and Civil Actions by Topic
# Actions
Month
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other
2022-01 … 2026-02