Problem Set 4

Alejandro Valdivia

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: **APE**

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
from bs4 import BeautifulSoup
import requests

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

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

Step 1: Develop initial scraper and crawler

```python
#Web page
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")
actions_list = [
    li for li in soup.find_all("li")
    if li.find("h2") and li.find("h2").find("a")
]

rows = []
for li in actions_list:
    h2 = li.find("h2")
    a = h2.find("a")
    title = a.get_text(strip=True)
    link = a.get("href", "")
    if link and not link.startswith("http"):
        link = "https://oig.hhs.gov" + (link if link.startswith("/") else "/"
 ↪  + link)

#Date
    date_str = ""
    for s in li.stripped_strings:
        if s == title:
            continue
        if "," in s and any(c.isdigit() for c in s) and len(s) < 30:
            date_str = s
            break
```

```
#Categories
    category_keywords = [
        "Criminal and Civil Actions", "State Enforcement Agencies",
        "CMP and Affirmative Exclusions", "COVID-19", "Child Support",
        "EMTALA/Patient Dumping", "Fraud Self-Disclosures",
        "Grant and Contractor Fraud Self-Disclosures",
        "CIA Reportable Events", "Stipulated Penalties and Material
         ↪  Breaches",
    ]
    categories = [s for s in li.stripped_strings if s in category_keywords]
    category_str = "; ".join(categories) if categories else ""

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

enforcement_df = pd.DataFrame(rows)
enforcement_df = enforcement_df.drop_duplicates(subset=["title", "link"],
 ↪  keep="first").reset_index(drop=True)
print(enforcement_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

Check whether to run the scraper

- If scraping is turned off, try to load the saved CSV file.
- If the file exists, return it and stop.
- Otherwise, continue to scraping.

Validate the year

- If the year is earlier than 2013, print a message saying enforcement actions are only available from 2013 onward and stop.

Set up

- Start from the first enforcement actions page.
- Begin at page 1.
- Create an empty list to store results.

Loop through pages

- Use a while loop since the total number of pages is unknown.

For each page:

- Download the page and extract all enforcement actions.
- If no actions are found, stop looping.

For each action:

- Read and interpret the action date.
- If the date is earlier than the requested start month and year, stop after this page.
- Otherwise, save the action's title, date, category, and link.
- Move to the next page and wait briefly before continuing.

Create output

- Convert the collected results into a dataframe.

Remove duplicates.

- Remove any actions that have the same title and link.
- Save the dataframe as a CSV file and return it.

b. Enforcements 2024

```python
from datetime import datetime

RUN_SCRAPER = False

CATEGORY_KEYWORDS = [
    "Criminal and Civil Actions", "State Enforcement Agencies",
    "CMP and Affirmative Exclusions", "COVID-19", "Child Support",
    "EMTALA/Patient Dumping", "Fraud Self-Disclosures",
    "Grant and Contractor Fraud Self-Disclosures",
    "CIA Reportable Events", "Stipulated Penalties and Material Breaches",
]

def parse_action_date(date_str):
    if not date_str or not date_str.strip():
        return None
    try:
        dt = datetime.strptime(date_str.strip(), "%B %d, %Y")
        return (dt.year, dt.month)
    except ValueError:
        return None

def scrape_enforcement_actions(month, year, run_scraper=RUN_SCRAPER):
    filename = f"enforcement_actions_{year}_{month:02d}.csv"

    if not run_scraper:
        try:
            return pd.read_csv(filename)
        except FileNotFoundError:
            pass

    if year < 2013:
        print("Only enforcement actions from 2013 onward are listed on the
         ↪  site. Please use year >= 2013.")
        return pd.DataFrame()

    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    start_ym = (year, month)
    rows = []
    page_num = 1
    reached_before_start = False

    while True:
        url = f"{base_url}?page={page_num}" if page_num > 1 else base_url
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, "html.parser")

        actions_list = [
            li for li in soup.find_all("li")
            if li.find("h2") and li.find("h2").find("a")
        ]
        if not actions_list:
            break

        for li in actions_list:
            h2 = li.find("h2")
            a = h2.find("a")
            title = a.get_text(strip=True)
            link = a.get("href", "")
            if link and not link.startswith("http"):
                link = "https://oig.hhs.gov" + (link if link.startswith("/")
↪   else "/" + link)

            date_str = ""
            for s in li.stripped_strings:
                if s == title:
                    continue
                if "," in s and any(c.isdigit() for c in s) and len(s) < 30:
                    date_str = s
                    break

            action_ym = parse_action_date(date_str)
            if action_ym is not None and action_ym < start_ym:
                reached_before_start = True
                continue
            if action_ym is None and date_str:
                continue

            categories = [s for s in li.stripped_strings if s in
↪   CATEGORY_KEYWORDS]
```

```
            category_str = "; ".join(categories) if categories else ""

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

        if reached_before_start:
            break
        page_num += 1
        time.sleep(1)

    df = pd.DataFrame(rows)
    df = df.drop_duplicates(subset=["title", "link"],
↪   keep="first").reset_index(drop=True)
    df.to_csv(filename, index=False)
    return df
```

```
df_2024 = scrape_enforcement_actions(1, 2024, run_scraper=RUN_SCRAPER)
print("January 2024 to today:")
print("Number of enforcement actions:", len(df_2024))
if len(df_2024) > 0:
    df_2024["date_parsed"] = pd.to_datetime(df_2024["date"], format="%B %d,
↪   %Y", errors="coerce")
    df_2024_sorted =
↪   df_2024.dropna(subset=["date_parsed"]).sort_values("date_parsed")
    earliest = df_2024_sorted.iloc[0]
    print("Earliest action date:", earliest["date"])
    print("Earliest action details:", earliest["title"])
```

```
January 2024 to today:
Number of enforcement actions: 1787
Earliest action date: January 3, 2024
Earliest action details: Former Nurse Aide Indicted In Death Of Clarksville
Patient Arrested In Georgia
```

c. Enforcements 2022

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

df_2022["date_parsed"] = pd.to_datetime(df_2022["date"], format="%B %d, %Y",
↪   errors="coerce")
print("January 2022 to today:")
print("Number of enforcement actions:", len(df_2022))
```

```
if len(df_2022) > 0:
    df_2022_sorted =
    ↪ df_2022.dropna(subset=["date_parsed"]).sort_values("date_parsed")
    earliest = df_2022_sorted.iloc[0]
    print("Earliest action date:", earliest["date"])
    print("Earliest action details:", earliest["title"])
    print("Earliest category:", earliest["category"])
```

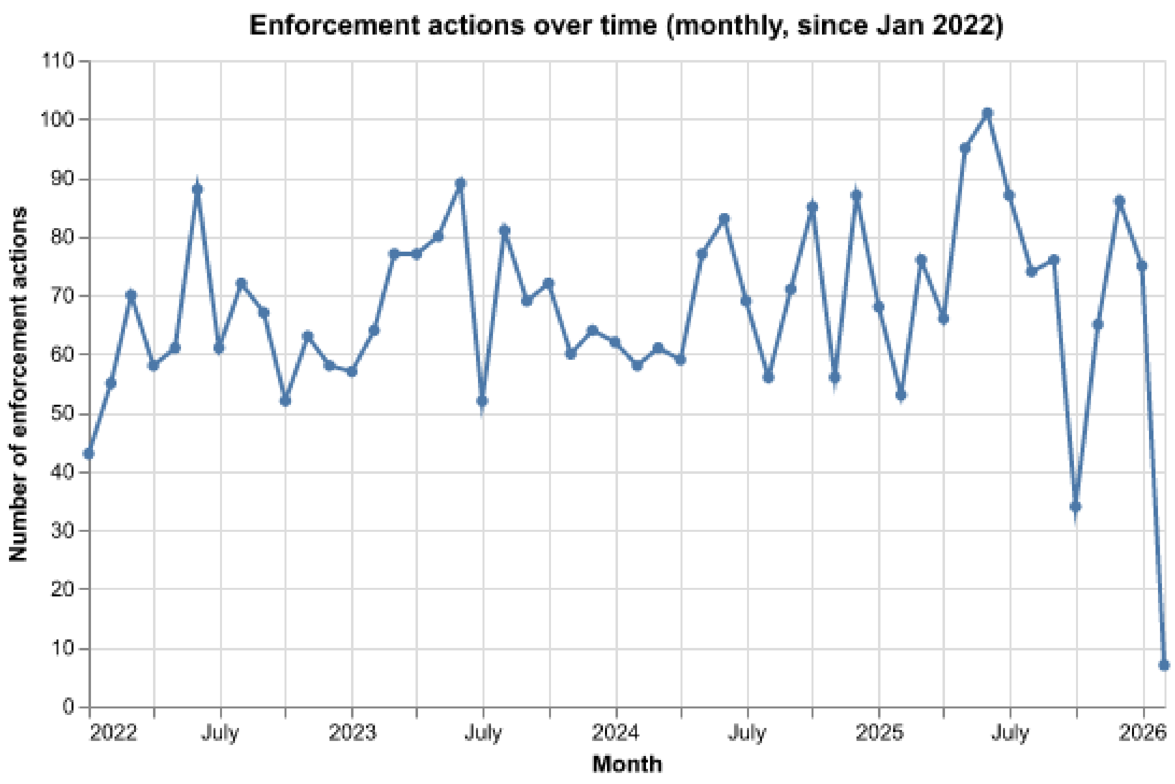
```
January 2022 to today:
Number of enforcement actions: 3377
Earliest action date: January 4, 2022
Earliest action details: Integrated Pain Management Medical Group Agreed to
Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by
Employing Excluded Individuals
Earliest category: Fraud Self-Disclosures
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df_2022_valid = df_2022.dropna(subset=["date_parsed"]).copy()
df_2022_valid["year_month"] =
↪ pd.to_datetime(df_2022_valid["date_parsed"].dt.strftime("%Y-%m-01"))
monthly = df_2022_valid.groupby("year_month",
↪ as_index=False).size().rename(columns={"size": "count"})

line_chart = (
    alt.Chart(monthly)
    .mark_line(point=True)
    .encode(
        x=alt.X("year_month:T", title="Month"),
        y=alt.Y("count:Q", title="Number of enforcement actions"),
    )
    .properties(title="Enforcement actions over time (monthly, since Jan
↪ 2022)", width=500, height=300)
)
line_chart
```

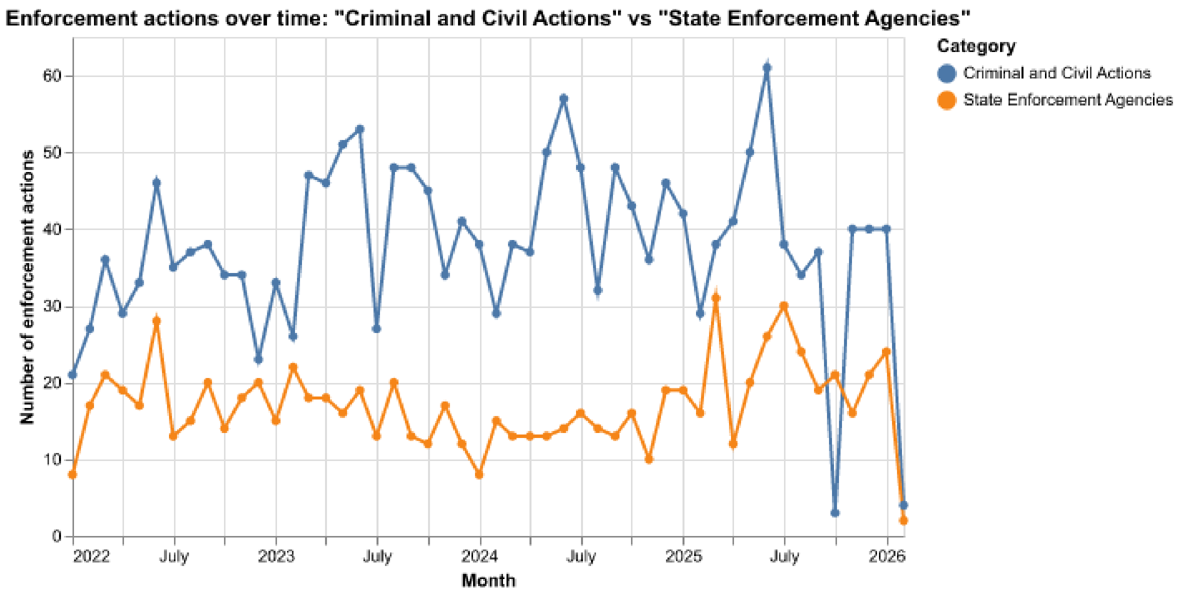


2. Plot the number of enforcement actions categorized:

Criminal and Civil Actions vs. State Enforcement Agencies:

```
df_cat = df_2022.dropna(subset=["date_parsed"]).copy()
df_cat["year_month"] =
↪   pd.to_datetime(df_cat["date_parsed"].dt.strftime("%Y-%m-01"))
targets = ["Criminal and Civil Actions", "State Enforcement Agencies"]
rows = []
for _, r in df_cat.iterrows():
    cats = [c.strip() for c in str(r["category"]).split(";") if c.strip() in
↪   targets]
    for c in cats:
        rows.append({"year_month": r["year_month"], "category": c})
cat_long = pd.DataFrame(rows)
monthly_by_cat = cat_long.groupby(["year_month", "category"],
↪   as_index=False).size().rename(columns={"size": "count"})

line_by_cat = (
    alt.Chart(monthly_by_cat)
    .mark_line(point=True)
    .encode(
        x=alt.X("year_month:T", title="Month"),
        y=alt.Y("count:Q", title="Number of enforcement actions"),
        color=alt.Color("category:N", title="Category"),
    )
    .properties(
        title='Enforcement actions over time: "Criminal and Civil Actions" vs
↪   "State Enforcement Agencies"',
        width=500,
        height=300,
    )
)
line_by_cat
```



Based on five topics

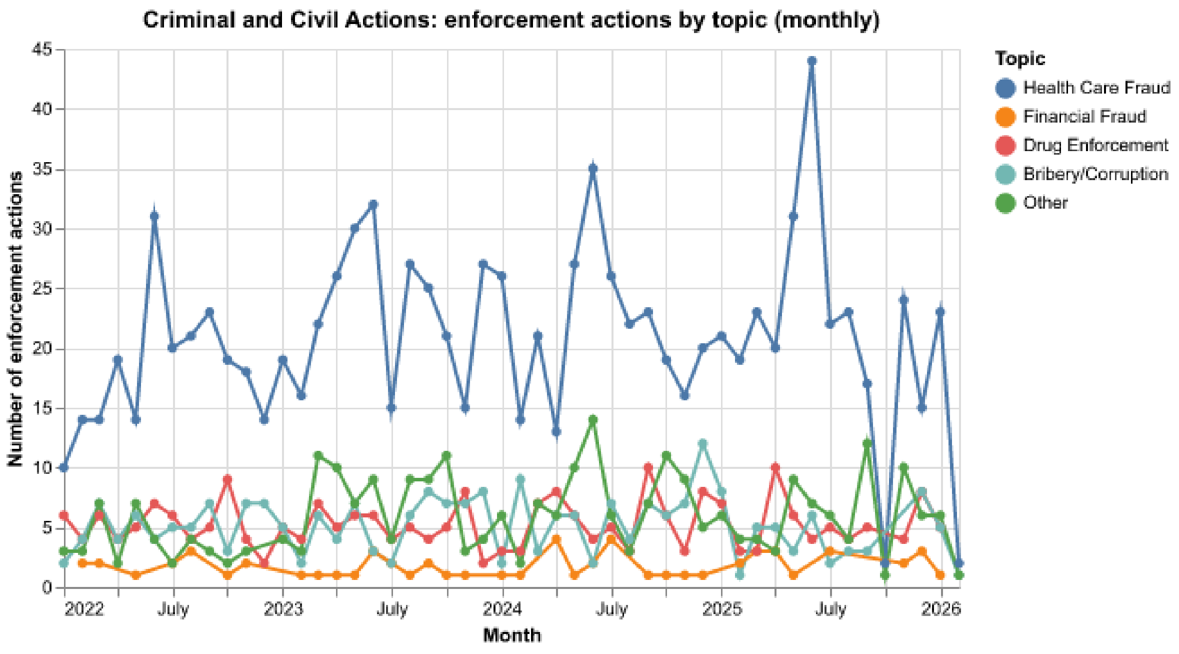
```
df_cca = df_2022.dropna(subset=["date_parsed"]).copy()
df_cca = df_cca[df_cca["category"].str.contains("Criminal and Civil Actions",
↪   na=False)]
df_cca["year_month"] =
↪   pd.to_datetime(df_cca["date_parsed"].dt.strftime("%Y-%m-01"))
```

```python
def assign_topic(title):
    if not isinstance(title, str):
        return "Other"
    t = title.lower()
    if any(k in t for k in ["bribery", "bribe", "corruption", "kickback"]):
        return "Bribery/Corruption"
    if any(k in t for k in ["drug", "opioid", "controlled substance",
     ↪  "prescription", "diversion", "pill mill", "distribut"]):
        return "Drug Enforcement"
    if any(k in t for k in ["bank", "financial", "embezzlement", "loan",
     ↪  "mortgage", "securities", "tax fraud", "money laundering"]):
        return "Financial Fraud"
    if any(k in t for k in ["medicare", "medicaid", "health care",
     ↪  "healthcare", "medical", "billing", "false claims", "nursing",
     ↪  "hospital", "clinic", "therapy", "chiropractor", "beneficiary",
     ↪  "defraud", "medicaid fraud", "medicare fraud"]):
        return "Health Care Fraud"
    return "Other"

df_cca["topic"] = df_cca["title"].apply(assign_topic)
monthly_by_topic = df_cca.groupby(["year_month", "topic"],
 ↪  as_index=False).size().rename(columns={"size": "count"})


topic_order = ["Health Care Fraud", "Financial Fraud", "Drug Enforcement",
 ↪  "Bribery/Corruption", "Other"]
monthly_by_topic["topic"] = pd.Categorical(monthly_by_topic["topic"],
 ↪  categories=topic_order, ordered=True)
monthly_by_topic = monthly_by_topic.sort_values(["year_month", "topic"])

line_by_topic = (
    alt.Chart(monthly_by_topic)
    .mark_line(point=True)
    .encode(
        x=alt.X("year_month:T", title="Month"),
        y=alt.Y("count:Q", title="Number of enforcement actions"),
        color=alt.Color("topic:N", title="Topic", sort=topic_order),
    )
    .properties(
        title='Criminal and Civil Actions: enforcement actions by topic
 ↪  (monthly)',
        width=500,
        height=300,
    )
)
line_by_topic
```


Criminal and Civil Actions: enforcement actions by topic (monthly)
Number of enforcement actions
July
July
July
July
Month
Topic
Health Care Fraud
Financial Fraud
Drug Enforcement
Bribery/Corruption
Other