PS4 Submission

Paul Zee-Cheng

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: **_PAZ_**

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 additional scraping & parsing libraries
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
from datetime import datetime
```

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

headers = {
    "User-Agent": "Mozilla/5.0 (compatible; OIG-Enforcement-Scraper/1.0)"
}

response = requests.get(TARGET_URL, headers=headers)
response.raise_for_status()

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

results = []

cards = soup.select("ul.usa-card-group > li.usa-card")
for card in cards:
    title = None
    date = None
    category = None
    links = []

    header = card.select_one("header.usa-card__header")
```

```
    if not header:
        continue

    # ---- Title ----
    title_tag = header.select_one("h2.usa-card__heading a")
    if title_tag:
        title = title_tag.get_text(strip=True)
        links.append(urljoin(BASE_URL, title_tag["href"]))

    # ---- Date ----
    date_tag = header.select_one("div.font-body-sm")
    if date_tag:
        date = next(date_tag.stripped_strings, None)
    # ---- Category ----
    category_tag = header.select_one(
        "ul.display-inline.add-list-reset li"
    )
    if category_tag:
        category = category_tag.get_text(strip=True)

    # ---- Collect all links ----
    for a in card.select("a[href]"):
        full_url = urljoin(BASE_URL, a["href"])
        if full_url not in links:
            links.append(full_url)

    if title:
        results.append({
            "title": title,
            "date": date,
            "category": category,
            "links": links
        })

print(f"Scraped {len(results)} enforcement actions")
print(results[0])
```

```
Scraped 20 enforcement actions
{'title': 'Houston Transplant Doctor Indicted For Making False Statements In
Patients' Medical Records', 'date': 'February 5, 2026', 'category': 'Criminal
and Civil Actions', 'links':
['https://oig.hhs.gov/fraud/enforcement/houston-transplant-doctor-indicted-for-making-false-s
```

```
# Convert results to a pandas DataFrame
df = pd.DataFrame(results)
df.head()
print(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
10  Yadkinville Woman Sentenced in Connection with...  January 29, 2026
11  Attorney General Labrador Announces Sentencing...  January 29, 2026
12  Attorney General Hanaway Obtains Medicaid Frau...  January 29, 2026
13  Holmes Regional Medical Center Agreed to Pay $...  January 28, 2026
14  Slidell Chiropractor Sentenced for Health Care...  January 28, 2026
15  Repeat Health Care Fraud Offender Sentenced fo...  January 28, 2026
16  Scranton Heart Institute Agrees To Pay $48,709...  January 28, 2026
17  Rheumatologist Agrees To Resolve False Claims ...  January 28, 2026
18  Attorney General James Uthmeier Announces Arre...  January 28, 2026
19  Cordell Memorial Hospital Agreed to Pay $40,00...  January 27, 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
10      Criminal and Civil Actions
11      State Enforcement Agencies
12      State Enforcement Agencies
13  CMP and Affirmative Exclusions
14                        COVID-19
```

```
15      Criminal and Civil Actions
16      Criminal and Civil Actions
17      Criminal and Civil Actions
18      State Enforcement Agencies
19  CMP and Affirmative Exclusions

                                                 links
0   [https://oig.hhs.gov/fraud/enforcement/houston...
1   [https://oig.hhs.gov/fraud/enforcement/multica...
2   [https://oig.hhs.gov/fraud/enforcement/brookly...
3   [https://oig.hhs.gov/fraud/enforcement/delafie...
4   [https://oig.hhs.gov/fraud/enforcement/former-...
5   [https://oig.hhs.gov/fraud/enforcement/attorne...
6   [https://oig.hhs.gov/fraud/enforcement/ags-off...
7   [https://oig.hhs.gov/fraud/enforcement/florida...
8   [https://oig.hhs.gov/fraud/enforcement/forefro...
9   [https://oig.hhs.gov/fraud/enforcement/attorne...
10  [https://oig.hhs.gov/fraud/enforcement/yadkinv...
11  [https://oig.hhs.gov/fraud/enforcement/attorne...
12  [https://oig.hhs.gov/fraud/enforcement/attorne...
13  [https://oig.hhs.gov/fraud/enforcement/holmes-...
14  [https://oig.hhs.gov/fraud/enforcement/slidell...
15  [https://oig.hhs.gov/fraud/enforcement/repeat-...
16  [https://oig.hhs.gov/fraud/enforcement/scranto...
17  [https://oig.hhs.gov/fraud/enforcement/rheumat...
18  [https://oig.hhs.gov/fraud/enforcement/attorne...
19  [https://oig.hhs.gov/fraud/enforcement/cordell...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

1. Define a function scrape_enforcement_actions(start_year, start_month, run_scraper)
2. If run_scraper is False, print a message and exit the function
3. If start_year $<$ 2013, print a warning and exit the function
4. Initialize: A. empty list to store scraped records. B. Page counter = 0
5. While True: A. Construct page URL using ?page={page}. B. Request page and parse HTML. C. Extract all enforcement action cards and D. If no cards are found, break loop.
6. For each card in the page: A. Extract title, date, category, link. B. Convert date to datetime C. If date $<$ start_year/start_month, stop scraping entirely D. Otherwise, append record to list

7. When no more cards are found and loop break, THEN Increment page counter and sleep(1) before loading next page
8. Convert results list to DataFrame
9. Save DataFrame as enforcement_actions_{year}_{month}.csv
10. Return DataFrame

- b. Create Dynamic Scraper

```
# redefine variables for a dynamic scraper
START_URL = "https://oig.hhs.gov/fraud/enforcement/"

#Create code
def scrape_enforcement_actions(start_year, start_month, run_scraper=False):
    """
    Scrape HHS OIG enforcement actions starting from a given month/year.
    """

    # ---- Indicator to prevent re-running during knit ----
    if not run_scraper:
        print("run_scraper is False - skipping scraping.")
        return None

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

    start_date = datetime(start_year, start_month, 1)

    # ----- Initialize, create empty list ----
    records = []
    page = 0
    headers = {
        "User-Agent": "Mozilla/5.0 (PS4-Enforcement-Scraper)"
    }

    while True:
        # Psuedocode Step 5:
        # Construct page URL
        url = f"{START_URL}?page={page}"

        # request page and convert to soup
        response = requests.get(url, headers=headers)
```

```
        response.raise_for_status()
        soup = BeautifulSoup(response.text, "html.parser")

        # Extract enforcement action cards
        cards = soup.select("ul.usa-card-group > li.usa-card")

        # Break loop if no cards found
        if not cards:
            break

        stop_scraping = False

        # For each card, extract details
        for card in cards:
            header = card.select_one("header.usa-card__header")
            if not header:
                continue

            # ---- Title ----
            title_tag = header.select_one("h2.usa-card__heading a")
            if not title_tag:
                continue

            title = title_tag.get_text(strip=True)
            link = urljoin(BASE_URL, title_tag["href"])

            # ---- Date (first text node only) ----
            date_tag = header.select_one("div.font-body-sm")
            if not date_tag:
                continue

            # Convert date to datetime
            date_str = next(date_tag.stripped_strings, None)
            date_dt = datetime.strptime(date_str, "%B %d, %Y")

            # If date to early, stop scraping entirely
            if date_dt < start_date:
                stop_scraping = True
                break

            # ---- Category ----
            categories = [
                li.get_text(strip=True)
```

```
                for li in header.select(
                    "ul.display-inline.add-list-reset li"
                )
            ]

            # Append record to list
            records.append({
                "title": title,
                "date": date_dt.strftime("%Y-%m-%d"),
                "category": categories,
                "link": link
            })

        if stop_scraping:
            break

        # Increment page counter
        page += 1

        # Sleep before loading so we don't get locked out
        time.sleep(1)

    # convert to dataframe
    df = pd.DataFrame(records)

    #save dataframe
    filename = f"enforcement_actions_{start_year}_{start_month}.csv"
    df.to_csv(filename, index=False)
    print(f"Saved {len(df)} enforcement actions to {filename}")

    #return df
    return df
```

- c. Test Your Code

```
# While testing code, run_scraper=True. Else, set to false.
df_2024 = scrape_enforcement_actions(
    start_year=2024,
    start_month=1,
    run_scraper=False
)
```

```
run_scraper is False - skipping scraping.
```

```
# Check results
print(df_2024)
# HOLY FUCK IT WORKED
# (this section will appear blank while rendering in quarto, because test
 ↪  code is set to false)
```

```
None
```

Per the csv generated file, the total number is 1,807. The earliest date is 1/3/2024, about a former nurse aid.

Now to test code from Jan 2022:

```
# PULL DATASET TO GRAPH

# While testing code, run_scraper=True. Else, set to false.
df_2022 = scrape_enforcement_actions(
    start_year=2022,
    start_month=1,
    run_scraper=False
)
```

```
run_scraper is False - skipping scraping.
```

Per the csv generated, this made 3,397 enforcement actions. The date and details of the earliest action are: 1/4/2022; fraud self disclosures; about the integrated pain management group..

Step 3: Plot data based on scraped data

```
# Import additional graphing and stats libraries
import re
import numpy as np
import vl_convert as vlc
from altair_saver import save
import selenium

# WHY DOESNT PNG WORK TO ENABLE GRAPHS
alt.renderers.enable('default')
```

```
RendererRegistry.enable('default')
```

```
# read in newly created csv
df = pd.read_csv('enforcement_actions_2022_1.csv')

# Preprocess data

# Convert date to datetime
df['date'] = pd.to_datetime(df['date'])

# Create month-year column for aggregation
df['month_year'] = df['date'].dt.to_period('M').astype(str)

# Extract year and month separately for better sorting
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month

# Create a proper datetime for the start of each month for plotting
df['month_date'] = df['date'].dt.to_period('M').dt.to_timestamp()

df.head()
```

	title	date	category	link
0	Houston Transplant Doctor Indicted For Making ...	2026-02-05	['Criminal and Civil Actions']	https:/
1	MultiCare Health System to Pay Millions to Set...	2026-02-04	['Criminal and Civil Actions']	https:/
2	Brooklyn Banker Pleads Guilty to Laundering Pr...	2026-02-03	['COVID-19']	https:/
3	Delafield Man Sentenced to 18 Months' Imprison...	2026-02-03	['Criminal and Civil Actions']	https:/
4	Former NFL Player Convicted for $197M Medicare...	2026-02-03	['Criminal and Civil Actions']	https:/

1. Plot the number of enforcement actions over time

```
# Aggregate overall counts by month
overall_counts = df.groupby(['month_date', 'year',
 ↪  'month']).size().reset_index(name='count')
overall_counts = overall_counts.sort_values(['year', 'month'])
print(overall_counts)
```

```
   month_date  year  month  count
0  2022-01-01  2022      1     43
```

1	2022-02-01	2022	2	55
2	2022-03-01	2022	3	70
3	2022-04-01	2022	4	58
4	2022-05-01	2022	5	61
5	2022-06-01	2022	6	88
6	2022-07-01	2022	7	61
7	2022-08-01	2022	8	72
8	2022-09-01	2022	9	67
9	2022-10-01	2022	10	52
10	2022-11-01	2022	11	63
11	2022-12-01	2022	12	58
12	2023-01-01	2023	1	57
13	2023-02-01	2023	2	64
14	2023-03-01	2023	3	77
15	2023-04-01	2023	4	77
16	2023-05-01	2023	5	80
17	2023-06-01	2023	6	89
18	2023-07-01	2023	7	52
19	2023-08-01	2023	8	81
20	2023-09-01	2023	9	69
21	2023-10-01	2023	10	72
22	2023-11-01	2023	11	60
23	2023-12-01	2023	12	64
24	2024-01-01	2024	1	62
25	2024-02-01	2024	2	58
26	2024-03-01	2024	3	61
27	2024-04-01	2024	4	59
28	2024-05-01	2024	5	77
29	2024-06-01	2024	6	83
30	2024-07-01	2024	7	69
31	2024-08-01	2024	8	56
32	2024-09-01	2024	9	71
33	2024-10-01	2024	10	85
34	2024-11-01	2024	11	56
35	2024-12-01	2024	12	87
36	2025-01-01	2025	1	68
37	2025-02-01	2025	2	53
38	2025-03-01	2025	3	76
39	2025-04-01	2025	4	66
40	2025-05-01	2025	5	95
41	2025-06-01	2025	6	101
42	2025-07-01	2025	7	87
43	2025-08-01	2025	8	74

```
44 2025-09-01  2025      9     76
45 2025-10-01  2025     10     34
46 2025-11-01  2025     11     65
47 2025-12-01  2025     12     86
48 2026-01-01  2026      1     88
49 2026-02-01  2026      2     14
```

```
# Graph actions over time
chart1 = alt.Chart(overall_counts).mark_line(point=True).encode(
    x=alt.X('month_date:T', title='Month'),
    y=alt.Y('count:Q', title='Number of Enforcement Actions'),
    tooltip=['month_date:T', 'count:Q']
).properties(
    title='Overall Enforcement Actions Over Time (Monthly)',
    width=700,
    height=400
).configure_axis(
    labelAngle=45
)

save(chart1, 'EAOT.html')
```

Figure 1 shows the actions over time.

2. Plot the number of enforcement actions categorized:

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

```
# First, split up categories
def categorize_action(category_str):
    if isinstance(category_str, str):
        if "'Criminal and Civil Actions'" in category_str or '"Criminal and
        ↪ Civil Actions"' in category_str:
            return 'Criminal and Civil Actions'
        elif "'State Enforcement Agencies'" in category_str or '"State
        ↪ Enforcement Agencies"' in category_str:
            return 'State Enforcement Agencies'
    return 'Other Categories'

# Add these categories to the dataframe
df['main_category'] = df['category'].apply(categorize_action)
```

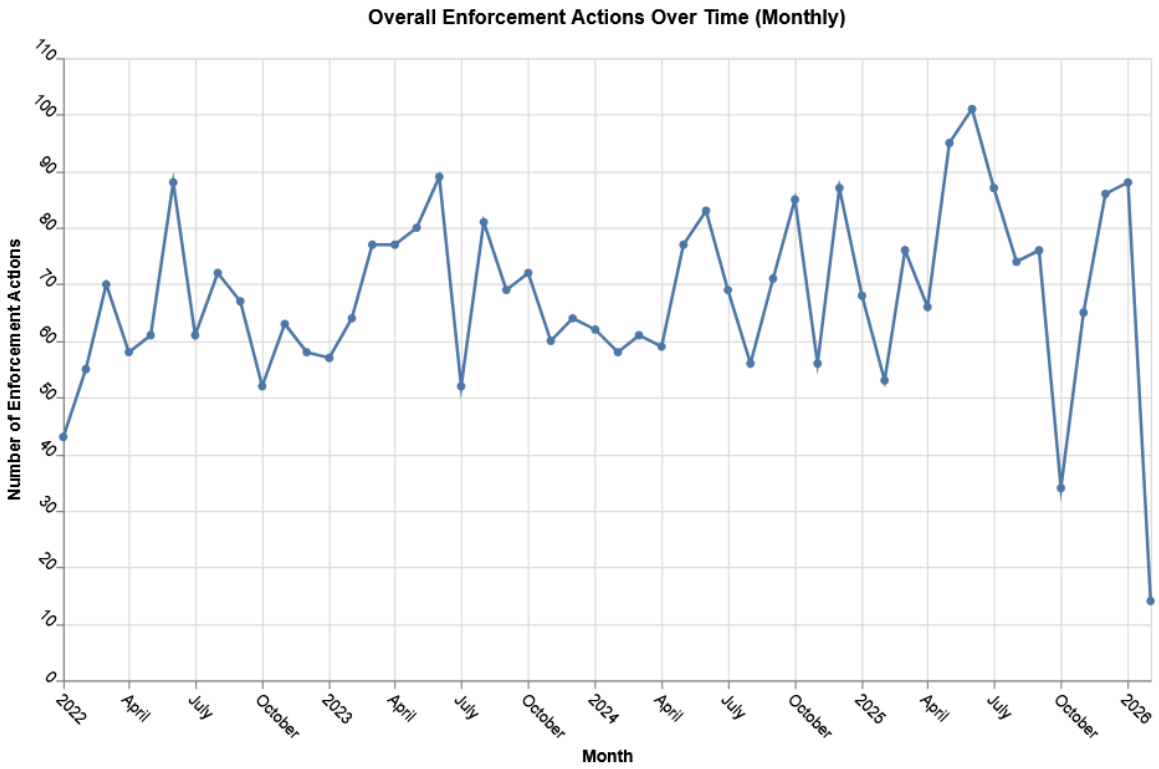

Figure 1: Enforcement Actions Over Time.

```
# Create group dataset to chart
category_counts = df.groupby(['month_date',
 ↪  'main_category']).size().reset_index(name='count')
category_counts = category_counts.sort_values(['month_date',
 ↪  'main_category'])
```

```
# Create graph of criminal/civil v state

chart2 = alt.Chart(category_counts).mark_line(point=True).encode(
    x=alt.X('month_date:T', title='Month'),
    y=alt.Y('count:Q', title='Number of Enforcement Actions'),
    color=alt.Color('main_category:N',
                    legend=alt.Legend(title='Category')),
    tooltip=['month_date:T', 'main_category:N', 'count:Q']
).properties(
```

```
    title='Enforcement Actions by Main Category (Monthly)',
    width=600,
    height=300
).configure_axis(
    labelAngle=45
)


save(chart2, 'EABC.html')
```

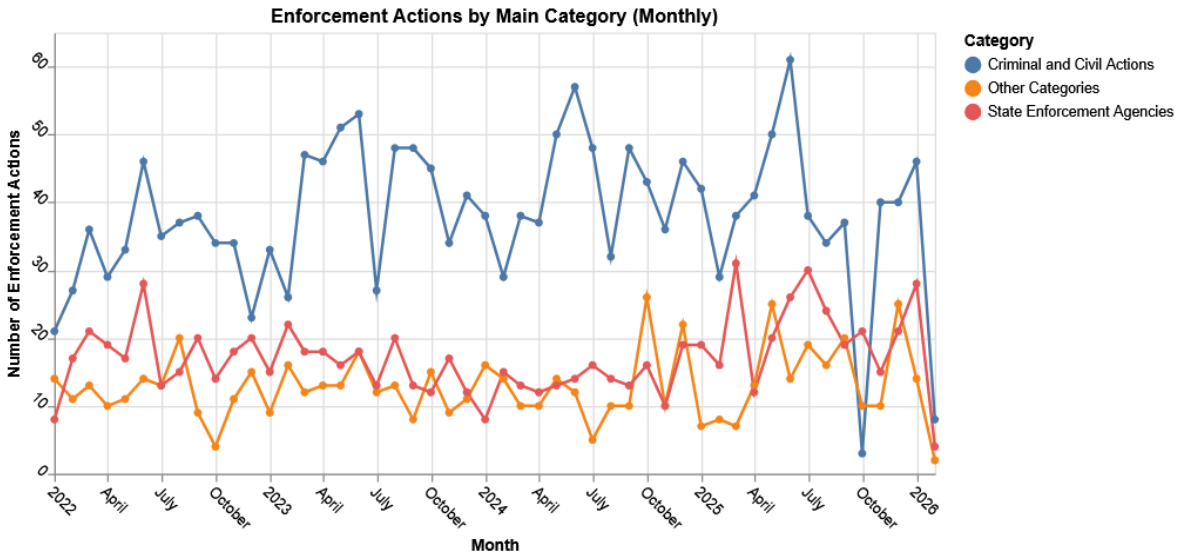

Figure 2: Enforcement Actions by Category.

Figure 2 shows the different categories of actions over time.

- based on five topics

```
# Create a new column for the five topics based on keyword matching
criminal_df = df[df['main_category'] == 'Criminal and Civil Actions'].copy()

# Create a series of criminal categoreis:
def classify_criminal_action(title):
    title_lower = str(title).lower()

    # Expanded keyword matching
```

```
    if any(word in title_lower for word in ['opioid', 'oxycodone',
     ↪  'controlled substance', 'drug',
                                            'prescription', 'fentanyl',
                                             ↪  'morphine', 'adderall']):
        return 'Drug Enforcement'
    elif any(word in title_lower for word in ['money laundering', 'wire
     ↪  fraud', 'financial',
                                              'tax fraud', 'identity theft',
                                               ↪  'embezzl', 'ponzi']):
        return 'Financial Fraud'
    elif any(word in title_lower for word in ['bribery', 'kickback',
     ↪  'corruption', 'remuneration']):
        return 'Bribery/Corruption'
    elif any(word in title_lower for word in ['health care fraud', 'medicare
     ↪  fraud', 'medicaid fraud',
                                              'false claims', 'defraud
                                               ↪  medicare', 'defraud
                                               ↪  medicaid']):
        return 'Health Care Fraud'
    else:
        return 'Other'

# Apply classification system
criminal_df['criminal_subcategory'] =
 ↪  criminal_df['title'].apply(classify_criminal_action)

# Create a graphable subdivided dataframe
criminal_counts = criminal_df.groupby(['month_date',
 ↪  'criminal_subcategory']).size().reset_index(name='count')
criminal_counts = criminal_counts.sort_values(['month_date',
 ↪  'criminal_subcategory'])

print(criminal_counts)
```

```
    month_date criminal_subcategory  count
0   2022-01-01   Bribery/Corruption      2
1   2022-01-01     Drug Enforcement      6
2   2022-01-01      Financial Fraud      1
3   2022-01-01    Health Care Fraud      8
4   2022-01-01                Other      4
..         ...                  ...    ...
234 2026-01-01    Health Care Fraud     19
```

```
235 2026-01-01                Other     15
236 2026-02-01   Bribery/Corruption      2
237 2026-02-01    Health Care Fraud      2
238 2026-02-01                Other      4

[239 rows x 3 columns]
```

```
# Create graph of criminal categories over time
chart3 = alt.Chart(criminal_counts).mark_line(point=True).encode(
    x=alt.X('month_date:T', title='Month'),
    y=alt.Y('count:Q', title='Number of Enforcement Actions'),
    color=alt.Color('criminal_subcategory:N',
                    legend=alt.Legend(title='Criminal Subcategory')),
    tooltip=['month_date:T', 'criminal_subcategory:N', 'count:Q']
).properties(
    title='Criminal and Civil Actions: Breakdown by Subcategory (Monthly)',
    width=600,
    height=300
).configure_axis(
    labelAngle=45
)

save(chart3, 'EA5T.html')
```

Figure 3 shows the criminal subcategories of enforcement actions over time.

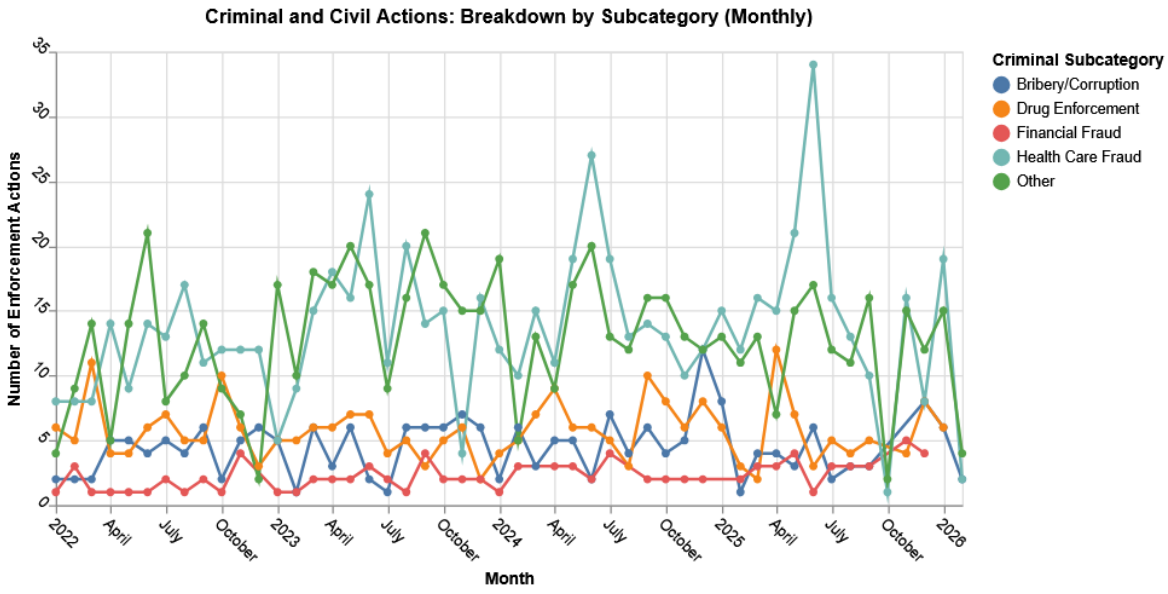

Figure 3: Enforcement Actions divided by Criminal Subcategory.