# 30538 Problem Set 4: Web Scraping

Benjamin Huynh

2026-02-07

**Due 02/07/2026 at 5:00PM Central**

"This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: BDH

## 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.

## (40%) Step 1: Develop initial scraper and crawler

**Scraping**: Go to the first page of the HHS OIG's "Enforcement Actions" page and scrape and collect the following into a dataset: * Title of the enforcement action * Date * Category (e.g, "Criminal and Civil Actions") * Link associated with the enforcement action

Collect your output into a tidy dataframe and print its `head`.

*Hint*: if you go to James A. Robinson's profile page at the Nobel Prize website here, right-click anywhere along the line "Affiliation at the time of the award: University of Chicago, Chicago, IL, USA", and select Inspect, you'll see that this affiliation information is located at the third `<p>` tag out of five `<p>` tags under the `<div class="content">`. Think about how you can select the third element of `<p>` out of five `<p>` elements so you're sure you scrape the affiliation information, not other. This way, you can scrape the name of agency to answer this question.

```
import requests
import os
import pandas as pd
import altair as alt
from bs4 import BeautifulSoup
from urllib.parse import urljoin
import re
import time
import datetime as dt

import vl_convert as vlc
from IPython.display import Image, display


BASE = "https://oig.hhs.gov"
URL = "https://oig.hhs.gov/fraud/enforcement/"

HEADERS = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 "
                  "(KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
}

DATE_RE = re.compile(
    r"\b(January|February|March|April|May|June|July|August|September|October|
November|December)"
    r"\s+\d{1,2},\s+\d{4}\b"
)
```

```
def extract_date(text):
    if not text:
        return None
    m = DATE_RE.search(text)
    return m.group(0) if m else None

def scrape_first_page(url=URL):
    r = requests.get(url, headers=HEADERS, timeout=30)
    r.raise_for_status()

    soup = BeautifulSoup(r.text, "html.parser")
    rows = []

    for h2 in soup.select("h2"):
        a = h2.find("a", href=True)
        if not a:
            continue

        title = a.get_text(strip=True)
        link = urljoin(BASE, a["href"])

        date_txt = None

        for sib in h2.next_siblings:
            if getattr(sib, "name", None) == "h2":
                break
            if getattr(sib, "get_text", None) is None:
                continue

            text = sib.get_text(" ", strip=True)
            candidate = extract_date(text)
            if candidate:
                date_txt = candidate
                break

        if date_txt is None:
            continue

        rows.append({
            "title": title,
            "date": pd.to_datetime(date_txt, format="%B %d, %Y"),
            "link": link,
        })

    df = pd.DataFrame(rows)
    df = df.sort_values("date", ascending=False).reset_index(drop=True)
    return df
```

```python
df1 = scrape_first_page()
print(df1.head())
```

```
                                               title       date  \
0  Brooklyn Banker Pleads Guilty to Laundering Pr... 2026-02-03
1  Former NFL Player Convicted for $197M Medicare... 2026-02-03
2  Delafield Man Sentenced to 18 Months' Imprison... 2026-02-03
3  AG's Office Secures Indictments Against Peabod... 2026-02-02
4  Florida Man Pleads Guilty to Conspiracy to Vio... 2026-01-30

                                                link
0  https://oig.hhs.gov/fraud/enforcement/brooklyn...
1  https://oig.hhs.gov/fraud/enforcement/former-n...
2  https://oig.hhs.gov/fraud/enforcement/delafiel...
3  https://oig.hhs.gov/fraud/enforcement/ags-offi...
4  https://oig.hhs.gov/fraud/enforcement/florida-...
```

## (40%) Step 2: Making the scraper dynamic

1. **Turning the scraper into a function**: You will write a function that takes as input a month and a year, and then pulls and formats the enforcement actions like in Step 1 starting from that month+year to today.

   - It is *very important* to make sure that you include an indicator whether or not to actually run the function. If you do not include an indicator, then each time you try to knit the qmd file, the scraper will run, and it will take a very long time to compile your pdf. Instead, run the function once to create a file called enforcement_actions_year_month.csv, which you can use in subsequent parts of the pset. Then turn the indicator off so that knitting your qmd file goes smoothly.
   - This function should first check that the year inputted >= 2013 before starting to scrape. If the year inputted < 2013, it should print a statement reminding the user to restrict to year >= 2013, since only enforcement actions after 2013 are listed.
   - It should save the dataframe output into a .csv file named as "enforcement_actions_*year_month*.csv" (do not commit this file to git)
   - If you're crawling multiple pages, always add 1 second wait before going to the next page to prevent potential server-side block. To implement this in Python, you may look up `.sleep()` function from `time` library.

a. Before writing out your function, write down pseudo-code of the steps that your function will go through. If you use a loop, discuss what kind of loop you will use and how you will define it. *Hint: Note that a simple `for` loop may not be sufficient for what this crawler requires. Use online resources to look into different types of loops or different ways of using `for` loops to see if there is something that is more appropriate for this task.*

b. Now code up your dynamic scraper and run it to start collecting the enforcement actions since January 2024. How many enforcement actions do you get in your final dataframe? What is the date and details of the earliest enforcement action it scraped?
c. Now, let's go a little further back. Test your code by collecting the actions since January 2022. *Note that this can take a while.* How many enforcement actions do you get in your final dataframe? What is the date and details of the earliest enforcement action it scraped? Use the dataframe from this process for every question after this.

*Hint*:

```
If you go to the next page in this HHS OIG's "Enforcement Actions" page,
you'll notice a pattern:

  * Second page URL: https://oig.hhs.gov/fraud/enforcement/?page=2
  * Third page URL: https://oig.hhs.gov/fraud/enforcement/?page=3
  * and so on ...
```

2A) First, I define the function that helps me scrape through the page for the different types of enforcement actions scrape_enforcement_actions(month, year, run_scraper=False)

If the function to run the scrape (run_scraper) turns out to be false, I will load the CSV containing the data (if applicable) and return the results (…print(df.head()))

I will then set the start date (start_date = date(year, month, 1) and initialize with page =1 and collect all rows (all_rows = [])

If it is true: I will provide the base url and the intended page. Then, scrape through the page with until the function breaks (only when the page is empty).

I will append rows only if date is later than or at the start date (date >= start_date), and will only break if a date that pops up is older than the start date.

'Page += 1' and then sleep(1) before the next page.

Save to the csv file, but do not commit the changes, and then return to the dataframe.

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

def scrape_enforcement_actions(month: int, year: int, run_scraper: bool =
False):
    out_csv = f"enforcement_actions_{year}_{month:02d}.csv"

    # If run_scraper is off, load the cached CSV if it exists
    if not run_scraper:
        if os.path.exists(out_csv):
            return pd.read_csv(out_csv, parse_dates=["date"])
        print(f"run_scraper=False and {out_csv} not found. Run once with
run_scraper=True.")
        return None
```

```
    # Guardrail required by assignment
    if year < 2013:
        print("Please restrict to year >= 2013 (only enforcement actions after
2013 are listed).")
        return None

    start_date = dt.date(year, month, 1)

    all_pages = []
    page = 1

    while True:
        url = BASE_URL if page == 1 else f"{BASE_URL}?page={page}"

        # Step 1 logic: scrape a single page into a DataFrame
        df_page = scrape_first_page(url)

        if df_page.empty:
            break

        # Filter to keep only relevant dates
        df_page["date_only"] = df_page["date"].dt.date
        df_keep = df_page[df_page["date_only"] >=
start_date].drop(columns=["date_only"])
        all_pages.append(df_keep)

        # Stop once we've gone earlier than requested start date
        oldest_on_page = df_page["date"].min().date()
        if oldest_on_page < start_date:
            break

        page += 1
        time.sleep(1)  # required pause

    out = pd.concat(all_pages, ignore_index=True) if all_pages else
pd.DataFrame(
        columns=["title", "date", "link"]
    )

    out = out.sort_values("date", ascending=False).reset_index(drop=True)
    out.to_csv(out_csv, index=False)

    return out
```

I used a while loop because the total number of pages I need to scrape through is unknown/not known in advance, so it will continue until a data-dependent stopping condition is reached.

2B)

```
#Running for Jan 2024
df_2024 = scrape_enforcement_actions(1, 2024, run_scraper=True)
df_2024
```

| | title | date | link |
|---|---|---|---|
| 0 | Brooklyn Banker Pleads Guilty to Laundering Pr… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/brooklyn… |
| 1 | Delafield Man Sentenced to 18 Months' Imprison… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/delafiel… |
| 2 | Former NFL Player Convicted for $197M Medicare… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/former-n… |
| 3 | AG's Office Secures Indictments Against Peabod… | 2026-02-02 | https://oig.hhs.gov/fraud/enforcement/ags-offi… |
| 4 | Florida Man Pleads Guilty to Conspiracy to Vio… | 2026-01-30 | https://oig.hhs.gov/fraud/enforcement/florida-… |
| … | … | … | … |
| 1764 | Athletico Management, PT Network, and Dynamic … | 2024-01-04 | https://oig.hhs.gov/fraud/enforcement/athletic… |
| 1765 | Recover-Care Plaza West Care Center Agreed to … | 2024-01-04 | https://oig.hhs.gov/fraud/enforcement/recover-… |
| 1766 | Maury County Caregiver Charged With Financial … | 2024-01-04 | https://oig.hhs.gov/fraud/enforcement/maury-co… |
| 1767 | Laredo Resident Admits To Impersonating Licens… | 2024-01-03 | https://oig.hhs.gov/fraud/enforcement/laredo-r… |
| 1768 | Former Nurse Aide Indicted In Death Of Clarksv… | 2024-01-03 | https://oig.hhs.gov/fraud/enforcement/former-n… |

```
#earliest 2024 instance
n_actions_2024 = len(df_2024)
earliest_row_2024 = df_2024.tail(1)
print("Number of enforcement actions collected:", n_actions_2024)
print("Earliest enforcement action row:")
print(earliest_row_2024)
```

```
Number of enforcement actions collected: 1769
Earliest enforcement action row:
                                                  title       date  \
1768  Former Nurse Aide Indicted In Death Of Clarksv... 2024-01-03

                                                   link
1768  https://oig.hhs.gov/fraud/enforcement/former-n...
```

```
                                                   link
1768  https://oig.hhs.gov/fraud/enforcement/former-n...
```

2C)

```
#Running for Jan 2022
df_2022 = scrape_enforcement_actions(1, 2022, run_scraper=True)
df_2022
```

| | title | date | link |
|---|---|---|---|
| 0 | Brooklyn Banker Pleads Guilty to Laundering Pr… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/brooklyn… |
| 1 | Delafield Man Sentenced to 18 Months' Imprison… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/delafiel… |
| 2 | Former NFL Player Convicted for $197M Medicare… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/former-n… |
| 3 | AG's Office Secures Indictments Against Peabod… | 2026-02-02 | https://oig.hhs.gov/fraud/enforcement/ags-offi… |
| 4 | Florida Man Pleads Guilty to Conspiracy to Vio… | 2026-01-30 | https://oig.hhs.gov/fraud/enforcement/florida-… |
| … | … | … | … |
| 3354 | North Carolina Physician Indicted for Adultera… | 2022-01-05 | https://oig.hhs.gov/fraud/enforcement/north-ca… |
| 3355 | Attorney General Alan Wilson announces arrest … | 2022-01-05 | https://oig.hhs.gov/fraud/enforcement/attorney… |
| 3356 | Central Medical Systems, LLC, Alan Trent Harle… | 2022-01-04 | https://oig.hhs.gov/fraud/enforcement/central-… |
| 3357 | Ohio home healthcare provider agrees to pay $5… | 2022-01-04 | https://oig.hhs.gov/fraud/enforcement/ohio-hom… |
| 3358 | Integrated Pain Management Medical Group Agree… | 2022-01-04 | https://oig.hhs.gov/fraud/enforcement/integrat… |

```
#earliest 2022 instance
n_actions_2022 = len(df_2022)
earliest_row_2022 = df_2022.tail(1)
print("Number of enforcement actions collected:", n_actions_2022)
```

```
print("Earliest enforcement action row:")
print(earliest_row_2022)
```

```
Number of enforcement actions collected: 3359
Earliest enforcement action row:
                                                  title       date  \
3358  Integrated Pain Management Medical Group Agree... 2022-01-04

                                                   link
3358  https://oig.hhs.gov/fraud/enforcement/integrat...
```

```
df_2022_B = scrape_enforcement_actions(1, 2022, run_scraper=False)
df_2022_B
```

| | title | date | link |
|---|---|---|---|
| 0 | Brooklyn Banker Pleads Guilty to Laundering Pr… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/brooklyn… |
| 1 | Delafield Man Sentenced to 18 Months' Imprison… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/delafiel… |
| 2 | Former NFL Player Convicted for $197M Medicare… | 2026-02-03 | https://oig.hhs.gov/fraud/enforcement/former-n… |
| 3 | AG's Office Secures Indictments Against Peabod… | 2026-02-02 | https://oig.hhs.gov/fraud/enforcement/ags-offi… |
| 4 | Florida Man Pleads Guilty to Conspiracy to Vio… | 2026-01-30 | https://oig.hhs.gov/fraud/enforcement/florida-… |
| … | … | … | … |
| 3354 | North Carolina Physician Indicted for Adultera… | 2022-01-05 | https://oig.hhs.gov/fraud/enforcement/north-ca… |
| 3355 | Attorney General Alan Wilson announces arrest … | 2022-01-05 | https://oig.hhs.gov/fraud/enforcement/attorney… |
| 3356 | Central Medical Systems, LLC, Alan Trent Harle… | 2022-01-04 | https://oig.hhs.gov/fraud/enforcement/central-… |
| 3357 | Ohio home healthcare provider agrees to pay $5… | 2022-01-04 | https://oig.hhs.gov/fraud/enforcement/ohio-hom… |
| 3358 | Integrated Pain Management Medical Group Agree… | 2022-01-04 | https://oig.hhs.gov/fraud/enforcement/integrat… |

## (20%) Step 3: Plot data based on scraped data

*Note*: To complete this part of the pset, reference the csv file you created in step 2, enforcement_actions_year_month.csv.

1. Plot a line chart with altair that shows: **the number of enforcement actions** over time (aggregated to each month+year) overall since January 2022.

First, we set up the required functions.

```
#Load dataset
df = pd.read_csv("enforcement_actions_2022_01.csv", parse_dates=["date"])

df["month"] = df["date"].dt.to_period("M").dt.to_timestamp()

def infer_category_from_title(title: str) -> str:
    t = str(title).lower()

    # State enforcement items often contain state-specific language
    if any(k in t for k in ["attorney general", "state of", "state
settlement", "state enforcement"]):
        return "State Enforcement Agencies"

    return "Criminal and Civil Actions"

df["category"] = df["title"].apply(infer_category_from_title)

def assign_topic(title: str) -> str:
    t = str(title).lower()

    # Financial Fraud
    if any(k in t for k in ["bank", "financial", "money laundering", "wire
fraud", "securities",
                            "mortgage", "credit", "loan"]):
        return "Financial Fraud"

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

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

    # Health Care Fraud
    if any(k in t for k in ["medicare", "medicaid", "health care",
```

```
"healthcare", "hospital",
                              "clinic", "home health", "hospice", "physician",
"nursing"]):
        return "Health Care Fraud"

    return "Other"

df["topic"] = df["title"].apply(assign_topic)
```

```
monthly_overall = (
    df.groupby("month", as_index=False)
      .size()
      .rename(columns={"size": "n_actions"})
)
```

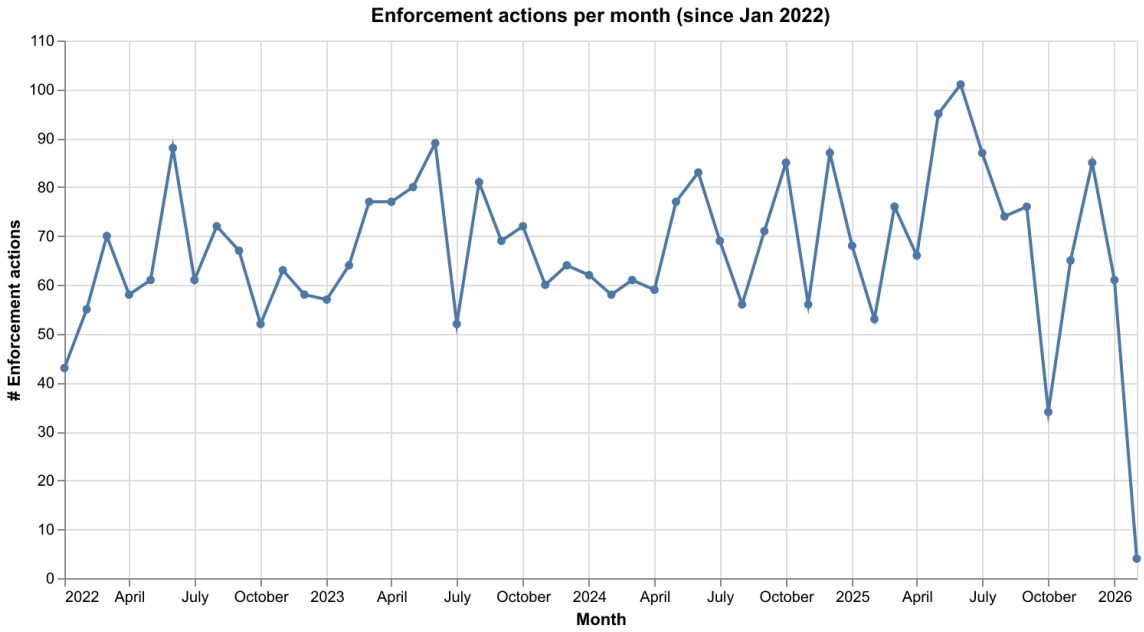
```
monthly_by_category = (
    df.groupby(["month", "category"], as_index=False)
      .size()
      .rename(columns={"size": "n_actions"})
)
```

```
df_crim = df[df["category"] == "Criminal and Civil Actions"].copy()

monthly_by_topic = (
    df_crim.groupby(["month", "topic"], as_index=False)
           .size()
           .rename(columns={"size": "n_actions"})
)
```

Now, I can plot the first chart for this problem, charting overall enforcement actions during 2022.

```
chart_overall = (
    alt.Chart(monthly_overall)
    .mark_line(point=True)
    .encode(
        x=alt.X("month:T", title="Month"),
        y=alt.Y("n_actions:Q", title="# Enforcement actions"),
        tooltip=[alt.Tooltip("month:T", title="Month"),
                 alt.Tooltip("n_actions:Q", title="# Actions")]
    )
    .properties(title="Enforcement actions per month (since Jan 2022)",
width=700, height=350)
)
chart_overall
chart_overall.save("chart_overall.png", scale_factor=2)
```

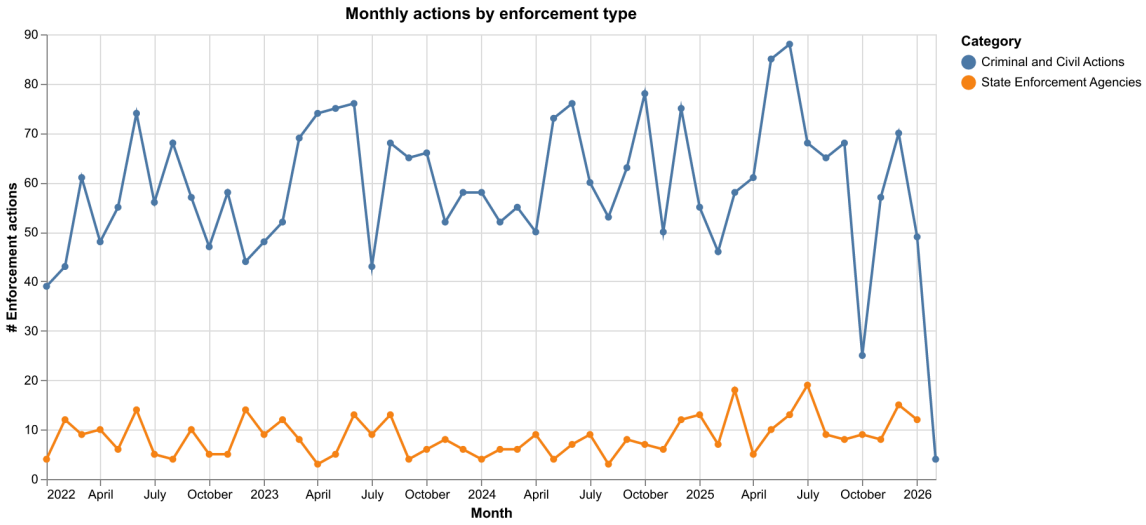



2. Plot a line chart with altair that shows: **the number of enforcement actions** split out by:

   - "Criminal and Civil Actions" vs. "State Enforcement Agencies"
   - Five topics in the "Criminal and Civil Actions" category: "Health Care Fraud", "Financial Fraud", "Drug Enforcement", "Bribery/Corruption", and "Other". *Hint: You will need to divide the five topics manually by looking at the title and assigning the relevant topic. For example, if you find the word "bank" or "financial" in the title of an action, then that action should probably belong to "Financial Fraud" topic. We suggest using AI to identify patterns in your scraped data and suggest ways of classifying based on the titles.*

Line graph depicting Criminal/Civil vs State Enforcement Agencies

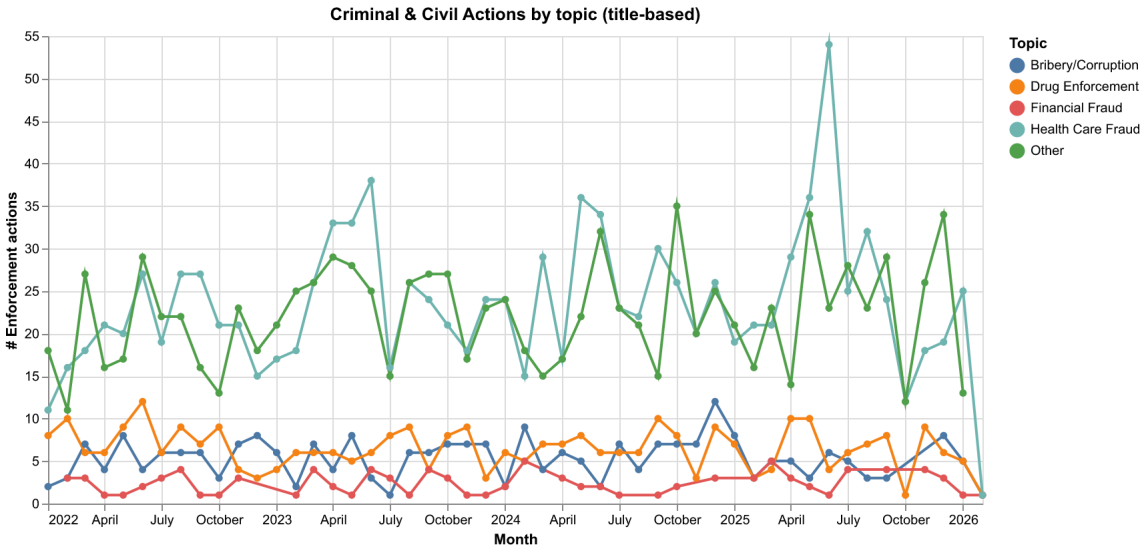
```
chart_two_categories = (
    alt.Chart(monthly_by_category)
    .mark_line(point=True)
    .encode(
        x=alt.X("month:T", title="Month"),
        y=alt.Y("n_actions:Q", title="# Enforcement actions"),
        color=alt.Color("category:N", title="Category"),
        tooltip=[alt.Tooltip("month:T", title="Month"),
                 alt.Tooltip("category:N", title="Category"),
                 alt.Tooltip("n_actions:Q", title="# Actions")]
    )
    .properties(title="Monthly actions by enforcement type", width=700,
height=350)
)
chart_two_categories
chart_two_categories.save("chart_two_categories.png", scale_factor=2)
```




Another line chart depicting topics within Criminal and Civil Actions

```
chart_topics = (
    alt.Chart(monthly_by_topic)
    .mark_line(point=True)
    .encode(
        x=alt.X("month:T", title="Month"),
        y=alt.Y("n_actions:Q", title="# Enforcement actions"),
        color=alt.Color("topic:N", title="Topic"),
        tooltip=[alt.Tooltip("month:T", title="Month"),
                 alt.Tooltip("topic:N", title="Topic"),
                 alt.Tooltip("n_actions:Q", title="# Actions")]
    )
    .properties(title="Criminal & Civil Actions by topic (title-based)",
width=700, height=350)
)

chart_topics
chart_topics.save("chart_topics.png", scale_factor=2)
```


Criminal & Civil Actions by topic (title-based)
# Enforcement actions
55
50
45
40
35
30
25
20
15
10
5
0
2022
April
July
October
2023
April
July
October
2024
April
July
October
2025
April
July
October
2026
Month
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other