PSet 4

Taha Rashid

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: TBR

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
alt.renderers.enable('html')


from datetime import datetime
import time

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


import requests
from bs4 import BeautifulSoup
```

Step 1: Develop initial scraper and crawler

```
# Set your ID (best practice)
my_headers = {
    "User-Agent": "MPP30538_Bot (taharashid@uchicago.edu)"
}

# URL we want to scrape
url = "https://oig.hhs.gov/fraud/enforcement/"

# Fetch the page from the internet
response = requests.get(url, headers=my_headers)

# Create a soup (searchabel object)
soup = BeautifulSoup(response.text, 'lxml')
```

```
# Extracting the right information

cards = soup.find_all('li', class_='usa-card')

# Grab the first card from our list
first_card = cards[0]

# 1. Find the <a> tag inside this specific card
```

```
link_tag = first_card.find('a')

# Extract Title (the text)
title = link_tag.text.strip()

# Extract the Link (the 'href' attribute)
link = link_tag.get('href')
```

```
# Extract date
date_text = first_card.find('span').text

# Test the extraction
date_text # got 'February 5, 2026' --> Exctraction successful no errors so
 ↪  far

# Extract category
category = first_card.find('li', class_='usa-tag').text.strip()

category # 'Criminal and Civil Actions' --> Extraction successful
```

```
'Criminal and Civil Actions'
```

```
# Consolidate Data

all_actions = []

for card in cards:
    # Get the Title and Link from the <h2>
    header = card.find('h2', class_='usa-card__heading')
    link_tag = header.find('a')

    title = link_tag.text.strip()
    # Glue the base URL to the relative path [cite: 2124, 2136]
    link = "https://oig.hhs.gov" + link_tag['href']

    # 2. Get the Date from the <span>
    date = card.find('span').text.strip()

    # 3. Get the Category from the <li> with class 'usa-tag'
    category = card.find('li', class_='usa-tag').text.strip()

    # 4. Save this observation as a dictionary [cite: 1612]
```

```
    all_actions.append({
        "Title": title,
        "Date": date,
        "Category": category,
        "Link": link
    })
```

```
df = pd.DataFrame(all_actions)

display(df) # Everything seems in order
```

	Title	Date	Category
0	Houston Transplant Doctor Indicted For Making ...	February 5, 2026	Criminal and Civil Actions
1	MultiCare Health System to Pay Millions to Set...	February 4, 2026	Criminal and Civil Actions
2	Brooklyn Banker Pleads Guilty to Laundering Pr...	February 3, 2026	COVID-19
3	Delafield Man Sentenced to 18 Months' Imprison...	February 3, 2026	Criminal and Civil Actions
4	Former NFL Player Convicted for $197M Medicare...	February 3, 2026	Criminal and Civil Actions
5	Attorney General Hanaway Obtains Medicaid Frau...	February 3, 2026	State Enforcement Agencies
6	AG's Office Secures Indictments Against Peabod...	February 2, 2026	State Enforcement Agencies
7	Florida Man Pleads Guilty to Conspiracy to Vio...	January 30, 2026	Criminal and Civil Actions
8	Forefront Living Hospice Agreed to Pay $1.9 Mi...	January 30, 2026	Fraud Self-Disclosures
9	Attorney General Jeff Jackson Announces Health...	January 30, 2026	State Enforcement Agencies
10	Yadkinville Woman Sentenced in Connection with...	January 29, 2026	Criminal and Civil Actions
11	Attorney General Labrador Announces Sentencing...	January 29, 2026	State Enforcement Agencies
12	Attorney General Hanaway Obtains Medicaid Frau...	January 29, 2026	State Enforcement Agencies
13	Holmes Regional Medical Center Agreed to Pay $...	January 28, 2026	CMP and Affirmative Exclusio
14	Slidell Chiropractor Sentenced for Health Care...	January 28, 2026	COVID-19
15	Repeat Health Care Fraud Offender Sentenced fo...	January 28, 2026	Criminal and Civil Actions
16	Scranton Heart Institute Agrees To Pay $48,709...	January 28, 2026	Criminal and Civil Actions
17	Rheumatologist Agrees To Resolve False Claims ...	January 28, 2026	Criminal and Civil Actions
18	Attorney General James Uthmeier Announces Arre...	January 28, 2026	State Enforcement Agencies
19	Cordell Memorial Hospital Agreed to Pay $40,00...	January 27, 2026	CMP and Affirmative Exclusio

```
df['Date'].dtype # currently a string object

# Convert the 'Date' column to actual datetime objects
df['Date'] = pd.to_datetime(df['Date'])

df.dtypes # now date time object
```

```
display(df)
```

	Title	Date	Category	L
0	Houston Transplant Doctor Indicted For Making ...	2026-02-05	Criminal and Civil Actions	h
1	MultiCare Health System to Pay Millions to Set...	2026-02-04	Criminal and Civil Actions	h
2	Brooklyn Banker Pleads Guilty to Laundering Pr...	2026-02-03	COVID-19	h
3	Delafield Man Sentenced to 18 Months' Imprison...	2026-02-03	Criminal and Civil Actions	h
4	Former NFL Player Convicted for $197M Medicare...	2026-02-03	Criminal and Civil Actions	h
5	Attorney General Hanaway Obtains Medicaid Frau...	2026-02-03	State Enforcement Agencies	h
6	AG's Office Secures Indictments Against Peabod...	2026-02-02	State Enforcement Agencies	h
7	Florida Man Pleads Guilty to Conspiracy to Vio...	2026-01-30	Criminal and Civil Actions	h
8	Forefront Living Hospice Agreed to Pay $1.9 Mi...	2026-01-30	Fraud Self-Disclosures	h
9	Attorney General Jeff Jackson Announces Health...	2026-01-30	State Enforcement Agencies	h
10	Yadkinville Woman Sentenced in Connection with...	2026-01-29	Criminal and Civil Actions	h
11	Attorney General Labrador Announces Sentencing...	2026-01-29	State Enforcement Agencies	h
12	Attorney General Hanaway Obtains Medicaid Frau...	2026-01-29	State Enforcement Agencies	h
13	Holmes Regional Medical Center Agreed to Pay $...	2026-01-28	CMP and Affirmative Exclusions	h
14	Slidell Chiropractor Sentenced for Health Care...	2026-01-28	COVID-19	h
15	Repeat Health Care Fraud Offender Sentenced fo...	2026-01-28	Criminal and Civil Actions	h
16	Scranton Heart Institute Agrees To Pay $48,709...	2026-01-28	Criminal and Civil Actions	h
17	Rheumatologist Agrees To Resolve False Claims ...	2026-01-28	Criminal and Civil Actions	h
18	Attorney General James Uthmeier Announces Arre...	2026-01-28	State Enforcement Agencies	h
19	Cordell Memorial Hospital Agreed to Pay $40,00...	2026-01-27	CMP and Affirmative Exclusions	h

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

We will run a while loop until we reach our desired break point, i.e. the year < 2013. We want all the articles posted after that so first we construct a while loop that takes in the url and finds the information as we did before. We will have an increment +1 to add the page count that we're looping through, since that is how we can move from page 1 url to page 2. On each page, we check the dates of the actions. As soon as we find a date that is older than the month and year the user requested, we stop the loop entirely.

We always add a 1-second pause before moving to a new page to prevent server-side block. Once we are done, we save all the information we collected into a CSV file so we don't have to scrape it again

- b. Create Dynamic Scraper

```
# def scrape_hhs_data(start_year, start_month, run_scraper=False):
#     if start_year < 2013:
#         print("Year must be >= 2013.")
#         return None
#     if not run_scraper:
#         return None

#     start_date = datetime(start_year, start_month, 1)
#     results = []
#     page_num = 1
#     keep_scraping = True

#     while keep_scraping:
#         url = f"https://oig.hhs.gov/fraud/enforcement/?page={page_num}"
#         response = requests.get(url)
#         soup = BeautifulSoup(response.content, 'html.parser')

#         cards = soup.find_all('div', class_='usa-card__container')
#         if not cards:
#             break

#         for card in cards:
#             date_container = card.find('div', class_='font-body-sm')
#             if not date_container:
#                 continue

#             # Extract only the date portion
#             full_text = date_container.get_text(strip=True, separator="|")
#             date_str = full_text.split("|")[0].strip()

#             try:
#                 article_date = datetime.strptime(date_str, "%B %d, %Y")
#             except ValueError:
#                 continue

#             if article_date < start_date:
#                 keep_scraping = False
#                 break

#             title_tag = card.find('h2').find('a')
#             title = title_tag.get_text(strip=True)
```

```
#             link = "https://oig.hhs.gov" + title_tag['href']
#             category = full_text.split("|")[-1].strip() if "|" in full_text
 ↪  else "N/A"

#             results.append({'Title': title, 'Date': article_date,
 ↪  'Category': category, 'Link': link})

#         print(f"Scraped page {page_num}...")
#         page_num += 1
#         time.sleep(1)

#     return pd.DataFrame(results)

# # Run the final check
# df_final = scrape_hhs_data(2024, 1, run_scraper=True)

# if df_final is not None and not df_final.empty:
#     print(f"Total actions: {len(df_final)}")
#     print(f"Earliest: {df_final.iloc[-1]['Date'].strftime('%Y-%m-%d')} -
 ↪  {df_final.iloc[-1]['Title']}")
```

- c. Test Your Code

```
# # Run the scraper starting from January 2022
# df_2022 = scrape_hhs_data(2022, 1, run_scraper=True)

# df_2022.to_csv("hhs_enforcement_2022_2026.csv", index=False)
```

```
# # Final 2013 Data Set


# def scrape_hhs_data(start_year, start_month, run_scraper=False):
#     # Requirement: Check if year >= 2013
#     if start_year < 2013:
#         print("Please restrict to year >= 2013, since only enforcement
 ↪  actions after 2013 are listed.")
#         return None


#     if not run_scraper:
#         print("Scraper is disabled. Function returning None.")
#         return None
```

```
#     start_date = datetime(start_year, start_month, 1)
#     results = []
#     page_num = 1
#     keep_scraping = True

#     while keep_scraping:
#         url = f"https://oig.hhs.gov/fraud/enforcement/?page={page_num}"
#         response = requests.get(url)
#         soup = BeautifulSoup(response.content, 'html.parser')

#         cards = soup.find_all('div', class_='usa-card__container')
#         if not cards: break

#         for card in cards:
#             date_container = card.find('div', class_='font-body-sm')
#             if not date_container: continue

#             full_text = date_container.get_text(strip=True, separator="|")
#             date_str = full_text.split("|")[0].strip()

#             try:
#                 article_date = datetime.strptime(date_str, "%B %d, %Y")
#             except ValueError: continue

#             if article_date < start_date:
#                 keep_scraping = False
#                 break

#             title_tag = card.find('h2').find('a')
#             results.append({
#                 'Title': title_tag.get_text(strip=True),
#                 'Date': article_date,
#                 'Category': full_text.split("|")[-1].strip(),
#                 'Link': "https://oig.hhs.gov" + title_tag['href']
#             })

#         page_num += 1
#         time.sleep(1)

#     df = pd.DataFrame(results)

#     # Save to CSV format
```

```
#     filename = "enforcement_actions_year_month.csv"
#     df.to_csv(filename, index=False)
#     print(f"File saved as {filename}")

#     return df
```

```
# df_2013 = scrape_hhs_data(2013, 1, run_scraper=True)
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

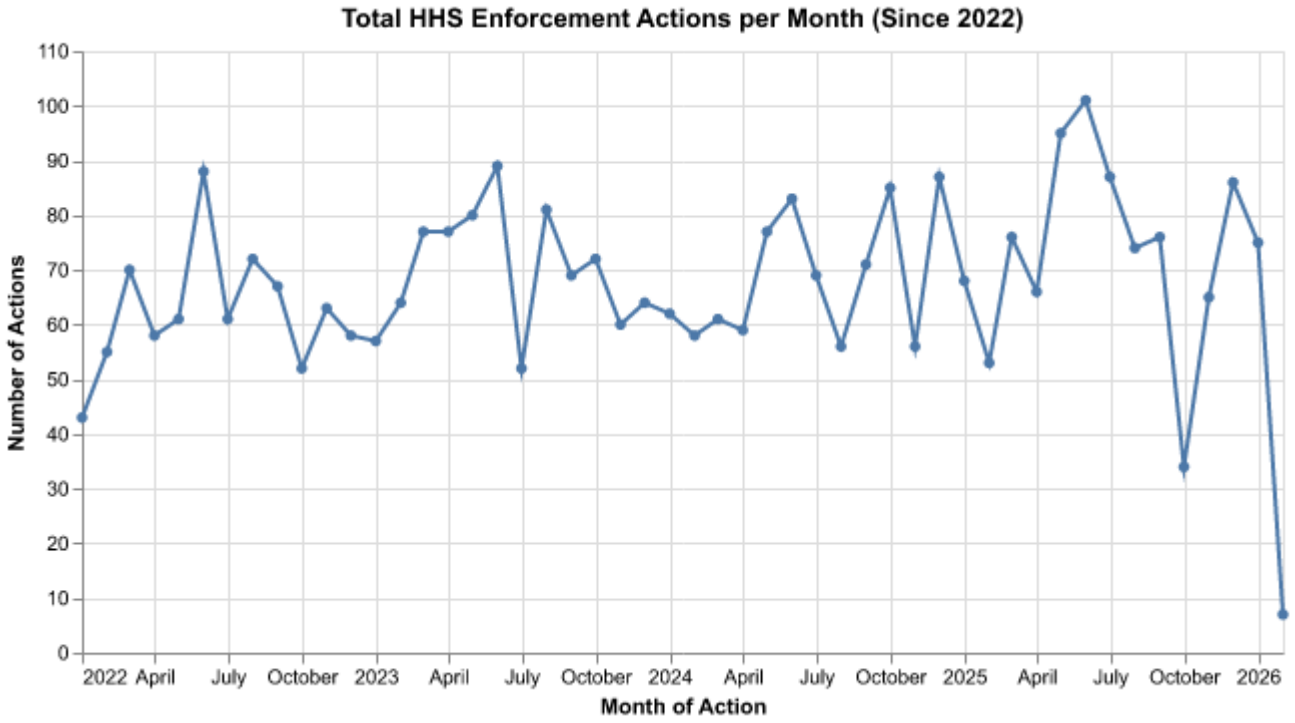
```
# Load the full dataset
df = pd.read_csv("enforcement_actions_year_month.csv")
df['Date'] = pd.to_datetime(df['Date'])

# Filter for Step 3 (Since Jan 2022)
df_plot = df[df['Date'] >= '2022-01-01'].copy()

# Create a Month-Year column for aggregation
df_plot['MonthYear'] = df_plot['Date'].dt.to_period('M').dt.to_timestamp()
```

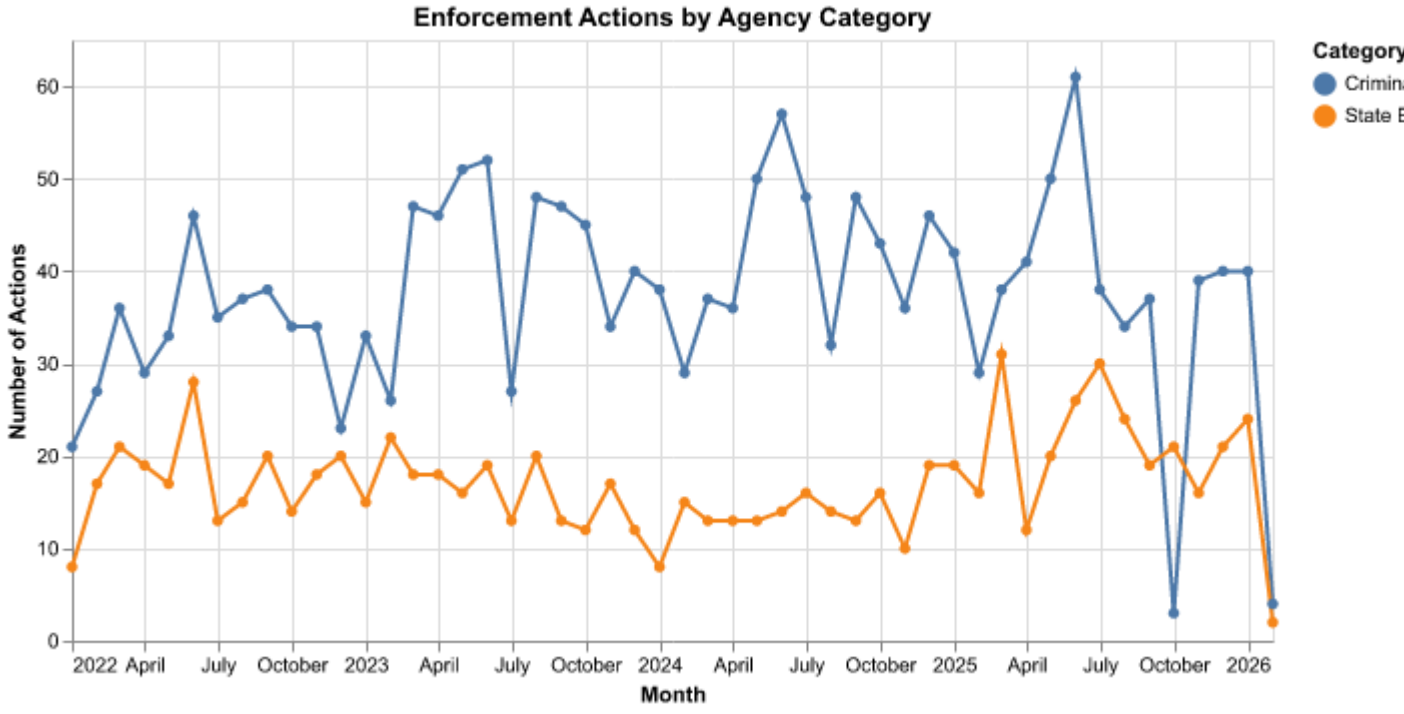
2. Plot the number of enforcement actions categorized:

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

```
# Aggregate count per month
overall_counts =
 ↪  df_plot.groupby('MonthYear').size().reset_index(name='n_actions')

chart1 = alt.Chart(overall_counts).mark_line(point=True).encode(
    x=alt.X('MonthYear:T', title='Month of Action'),
    y=alt.Y('n_actions:Q', title='Number of Actions'),
    tooltip=['MonthYear', 'n_actions']
).properties(
    title='Total HHS Enforcement Actions per Month (Since 2022)',
    width=600
)
```

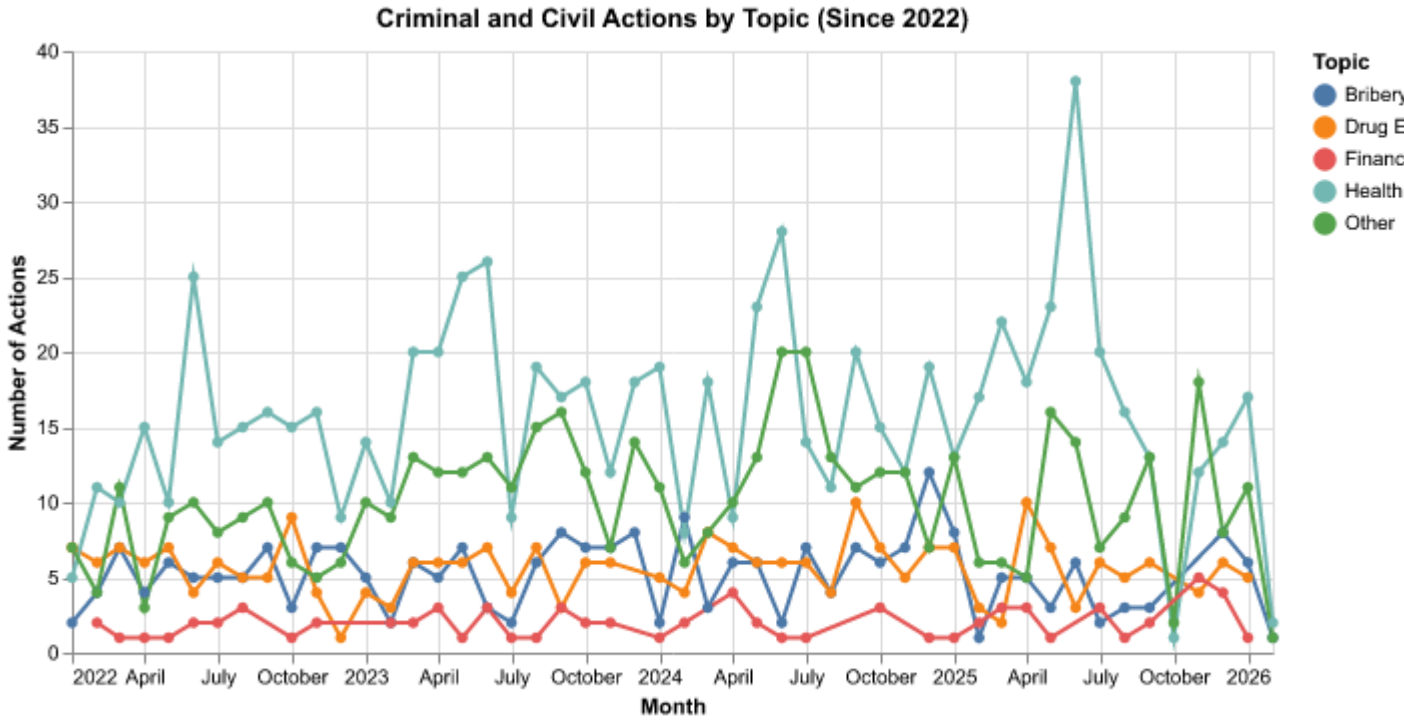
```
chart1.display()


# Filter for specific categories
cat_df = df_plot[df_plot['Category'].isin(['Criminal and Civil Actions',
 ↪  'State Enforcement Agencies'])]

# Group by Month and Category
split_counts = cat_df.groupby(['MonthYear',
 ↪  'Category']).size().reset_index(name='n_actions')

chart2 = alt.Chart(split_counts).mark_line(point=True).encode(
    x=alt.X('MonthYear:T', title='Month'),
    y=alt.Y('n_actions:Q', title='Number of Actions'),
    color='Category:N',
    tooltip=['MonthYear', 'Category', 'n_actions']
).properties(
    title='Enforcement Actions by Agency Category',
    width=600
)

chart2.display()
```





- based on five topics

```
def classify_topic(title):
    t = title.lower()
    if any(word in t for word in ['kickback', 'bribe', 'corruption',
     ↪  'referral']):
        return 'Bribery/Corruption'
    if any(word in t for word in ['drug', 'opioid', 'substance', 'pharmacy',
     ↪  'prescription']):
        return 'Drug Enforcement'
    if any(word in t for word in ['money laundering', 'securities', 'wire
     ↪  fraud', 'tax', 'bank', 'financial']):
        return 'Financial Fraud'
    if any(word in t for word in ['medicare', 'medicaid', 'health care',
     ↪  'medical', 'hospital', 'doctor']):
        return 'Health Care Fraud'
    return 'Other'

# Apply classification only to the Criminal/Civil category
df_criminal = df_plot[df_plot['Category'] == 'Criminal and Civil
 ↪  Actions'].copy()
df_criminal['Topic'] = df_criminal['Title'].apply(classify_topic)

# Aggregate for plotting
topic_counts = df_criminal.groupby(['MonthYear',
 ↪  'Topic']).size().reset_index(name='n_actions')

chart3 = alt.Chart(topic_counts).mark_line(point=True).encode(
    x=alt.X('MonthYear:T', title='Month'),
    y=alt.Y('n_actions:Q', title='Number of Actions'),
    color='Topic:N',
    tooltip=['MonthYear', 'Topic', 'n_actions']
).properties(
    title='Criminal and Civil Actions by Topic (Since 2022)',
    width=600
)

chart3.display()
```


Criminal and Civil Actions by Topic (Since 2022)
Number of Actions
0
5
10
15
20
25
30
35
40
2022
April
July
October
2023
April
July
October
2024
April
July
October
2025
April
July
October
2026
Month
Topic
Bribery
Drug E
Financ
Health
Other