Problem Set 4

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: SR

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 re
from urllib.parse import urljoin
import requests
from bs4 import BeautifulSoup
import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

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

```python
RUN_SCRAPER = False
```

Step 1: Develop initial scraper and crawler

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

HEADERS = {"User-Agent": "Mozilla/5.0"}

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

response = requests.get(START_URL, headers=HEADERS, timeout=30)
response.raise_for_status()

soup = BeautifulSoup(response.text, "lxml")

rows = []

action_links = soup.select("main h2 a[href^='/fraud/enforcement/']")

for a_tag in action_links:
    title = a_tag.get_text(strip=True)
    href = a_tag.get("href", "")

    if not title:
        continue
    if href.rstrip("/") == "/fraud/enforcement":
        continue

    full_link = urljoin(BASE_URL, href)

    card = a_tag.find_parent(["article", "li", "div"])
    card_text = " ".join(card.stripped_strings) if card else ""

    date_match = DATE_PATTERN.search(card_text)
    date = date_match.group(0) if date_match else None

    remainder = card_text.replace(title, "")
    if date:
        remainder = remainder.replace(date, "")
    category = " ".join(remainder.split()).strip() or None

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

df_actions = (
    pd.DataFrame(rows)
      .drop_duplicates(subset="link")
      .reset_index(drop=True)
)

df_actions.head()
```

	title	date	category	link
0	Houston Transplant Doctor Indicted F...	February 5, 2026	Criminal and Civil Actions	https://oig.hhs
1	MultiCare Health System to Pay Milli...	February 4, 2026	Criminal and Civil Actions	https://oig.hhs
2	Brooklyn Banker Pleads Guilty to Lau...	February 3, 2026	COVID-19	https://oig.hhs
3	Delafield Man Sentenced to 18 Months...	February 3, 2026	Criminal and Civil Actions	https://oig.hhs
4	Former NFL Player Convicted for $197...	February 3, 2026	Criminal and Civil Actions	https://oig.hhs

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

Collect all HHS OIG enforcement actions starting from a given month and year up to today. The actions are spread across multiple pages, so we need to keep moving forward page by page.

We do not use a simple for loop because we do not know how many pages exist ahead of time. Instead, we keep going until we reach actions that are older than our start date.

Pseudo-code

1. Start the function

   Take three inputs:

   - start_year
   - start_month
   - run_scraper (True or False)

2. Check if the year is valid

   If start_year is less than 2013:

   - Print a message telling the user to use a year greater than or equal to 2013
   - Stop the function

3. Create a starting date

   - Convert the start_year and start_month into a date
   - Use the first day of that month as the starting point

4. Set up variables before scraping

   - Set page = 1 (this represents the first page)
   - Create an empty list called all_rows to store the results
   - Create a variable keep_going = True to control the loop

5. Start scraping pages

   While keep_going is True:

   - If page equals 1:
   - Use the main enforcement actions URL

   Otherwise:

   - Add ?page=page to the URL
   - Download the webpage using requests
   - Parse the HTML using BeautifulSoup
   - Find all enforcement actions listed on that page
   - For each enforcement action:
   - Extract the title, date, category, and link
   - Convert the date into a date object
   - If the action's date is on or after the start date:
     - `Add the action to all_rows`
   - Otherwise:
   - Set keep_going = False
   - Stop checking more actions, because the rest will be even older
   - If keep_going is still True:
   - Move to the next page by increasing page by 1
   - Pause for 1 second to avoid sending too many requests

6. After scraping is finished

   - Convert all_rows into a pandas DataFrame
   - Remove any duplicate rows using the link
   - Save the DataFrame as:
   - enforcement_actions___.csv
   - Return the DataFrame

- b. Create Dynamic Scraper

```
def scrape_enforcement_actions_since(start_year, start_month,
↪   run_scraper=True):

    if run_scraper is False:
        print("Scraper is turned off. Skipping scraping.")
        return None

    if start_year < 2013:
        print("Only enforcement actions from 2013 onward are available.")
        print("Please choose a year greater than or equal to 2013.")
        return None

    start_date = pd.Timestamp(year=start_year, month=start_month, day=1)

    BASE_URL = "https://oig.hhs.gov"
    LIST_URL = "https://oig.hhs.gov/fraud/enforcement/"
    headers = {"User-Agent": "Mozilla/5.0"}

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

    all_rows = []
    page = 1
    keep_going = True

    while keep_going:
        if page == 1:
            url = LIST_URL
        else:
            url = f"{LIST_URL}?page={page}"

        response = requests.get(url, headers=headers, timeout=30)
        response.raise_for_status()

        soup = BeautifulSoup(response.text, "lxml")

        title_links = soup.select("main h2 a[href^='/fraud/enforcement/']")

        if not title_links:
```

```python
            break

    for a in title_links:
        title = a.get_text(strip=True)
        href = a.get("href", "")

        if not title:
            continue
        if href.rstrip("/") == "/fraud/enforcement":
            continue

        link = urljoin(BASE_URL, href)

        card = a.find_parent(["article", "li", "div"])
        card_text = " ".join(card.stripped_strings) if card else ""

        date_match = date_pattern.search(card_text)
        if not date_match:
            continue

        date_str = date_match.group(0)
        action_date = pd.to_datetime(date_str)

        if action_date < start_date:
            keep_going = False
            break

        remainder = card_text.replace(title, "").replace(date_str, "")
        category = " ".join(remainder.split()).strip() or None

        all_rows.append(
            {
                "title": title,
                "date": action_date,
                "category": category,
                "link": link
            }
        )

    if keep_going:
        page += 1
        time.sleep(1)
df = (
    pd.DataFrame(all_rows)
    .drop_duplicates(subset="link")
    .reset_index(drop=True)
)

filename = f"enforcement_actions_{start_year}_{start_month:02d}.csv"
df.to_csv(filename, index=False)
print(f"Saved {len(df)} enforcement actions to {filename}")

return df
```

```
if RUN_SCRAPER:
    df_2024 = scrape_enforcement_actions_since(2024, 1, run_scraper=True)
else:
    df_2024 = pd.read_csv("enforcement_actions_2024_01.csv",
 ↪  parse_dates=["date"])

print("Number of enforcement actions:", len(df_2024))

earliest_action = df_2024.sort_values("date").iloc[0]
print("Earliest date:", earliest_action["date"].date())
print("Earliest title:", earliest_action["title"])
print("Earliest category:", earliest_action["category"])
print("Earliest link:", earliest_action["link"])
```

```
Number of enforcement actions: 1771
Earliest date: 2024-01-03
Earliest title: Former Nurse Aide Indicted In Death Of Clarksville Patient
Arrested In Georgia
Earliest category: State Enforcement Agencies
Earliest link:
https://oig.hhs.gov/fraud/enforcement/former-nurse-aide-indicted-in-death-of-clarksville-pat
```

- c. Test Your Code

```
if RUN_SCRAPER:
    df_2022 = scrape_enforcement_actions_since(2022, 1, run_scraper=True)
else:
    df_2022 = pd.read_csv("enforcement_actions_2022_01.csv",
 ↪  parse_dates=["date"])
```

```
num_actions = len(df_2022)
print("How many enforcement actions:", num_actions)

    # find the earliest action (smallest date)
df_sorted = df_2022.sort_values("date")
earliest_row = df_sorted.iloc[0]

print("Earliest date:", earliest_row["date"].date())
print("Earliest title:", earliest_row["title"])
print("Earliest category:", earliest_row["category"])
print("Earliest link:", earliest_row["link"])
```

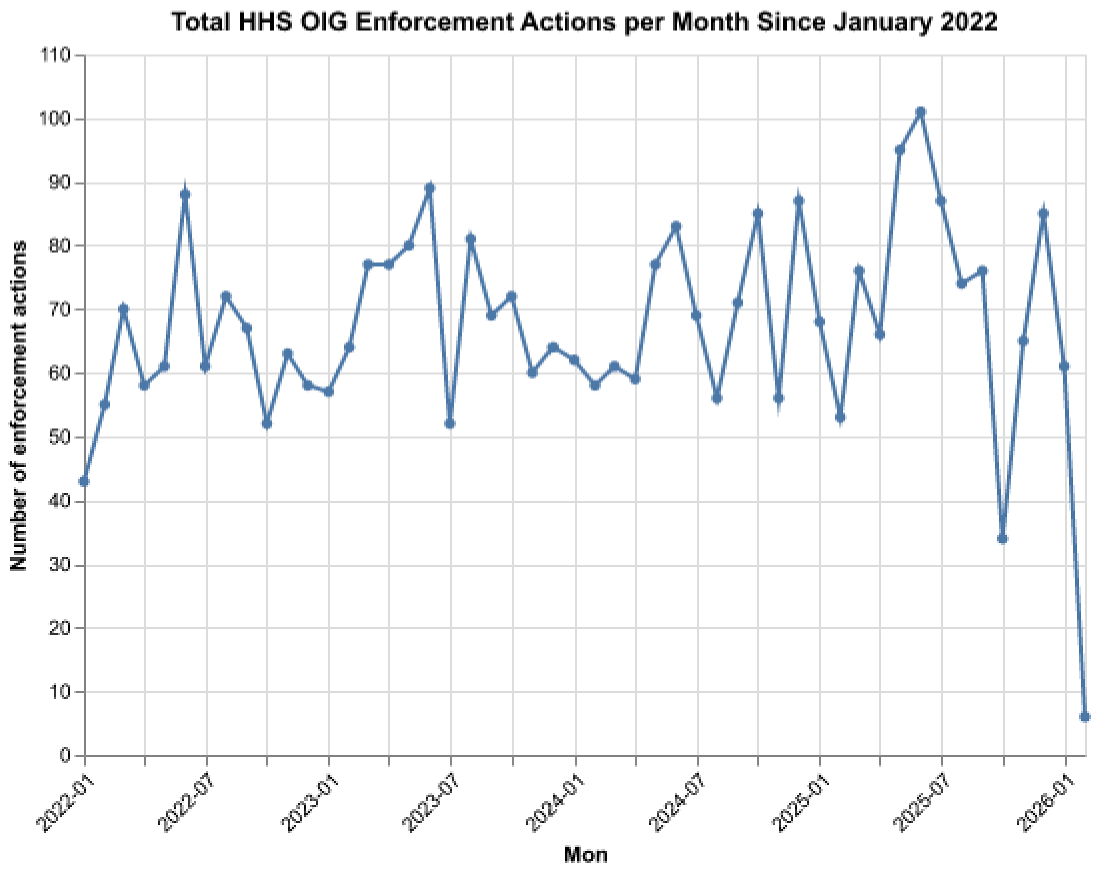
```
How many enforcement actions: 3361
Earliest date: 2022-01-04
Earliest title: Integrated Pain Management Medical Group Agreed to Pay
$10,000 for Allegedly Violating the Civil Monetary Penalties Law by Employing
Excluded Individuals
Earliest category: Fraud Self-Disclosures
Earliest link:
https://oig.hhs.gov/fraud/enforcement/integrated-pain-management-medical-group-agreed-to-pay-
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

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

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

monthly_total = (
    df
    .groupby("year_month")
    .size()
    .reset_index(name="number_actions")
)
```

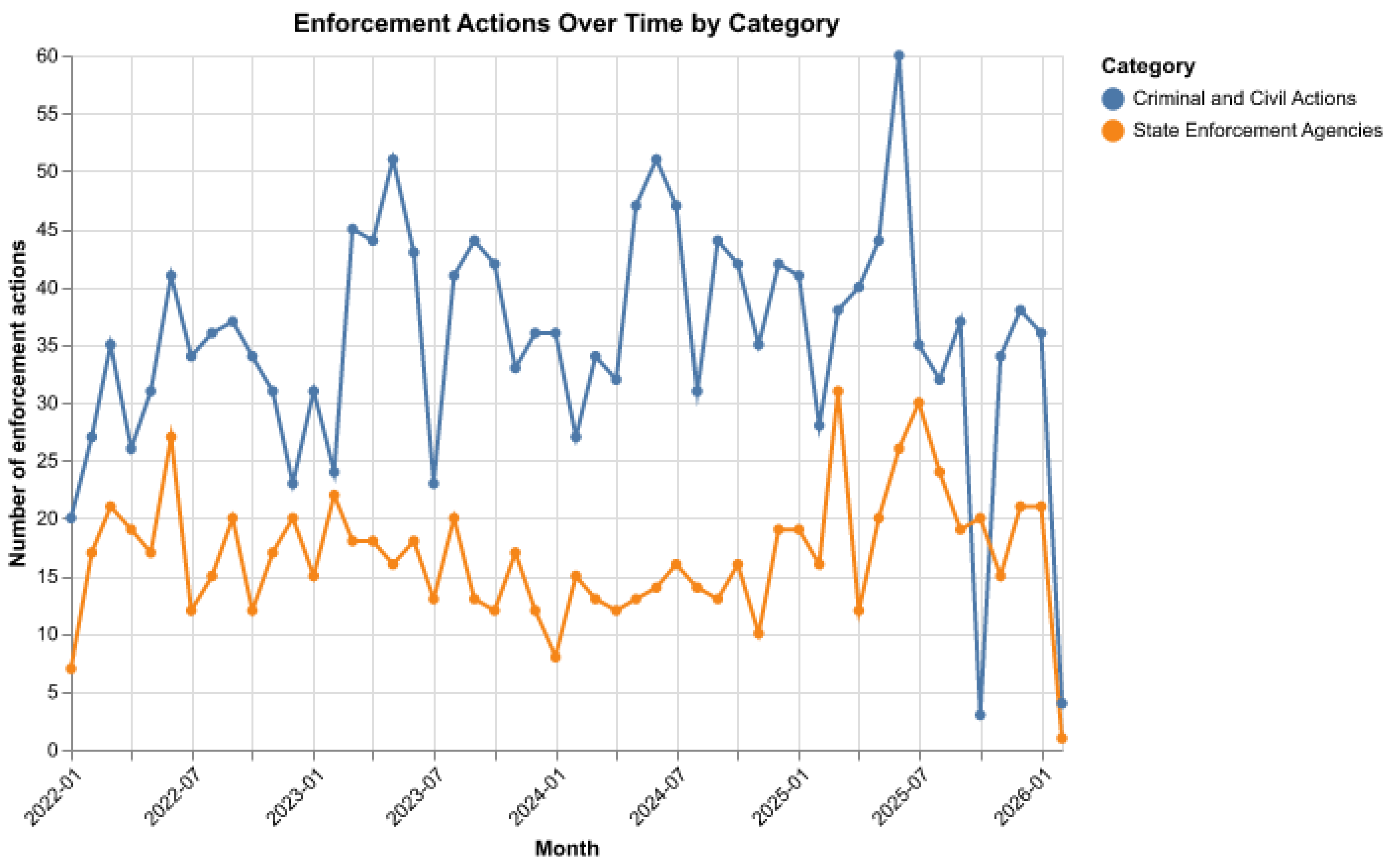
```
alt.Chart(monthly_total).mark_line(point=True).encode(
    alt.X("year_month:T", title="Mon",  axis=alt.Axis(format="%Y-%m",
 ↪  labelAngle=-45)),
    alt.Y("number_actions:Q", title="Number of enforcement actions"),
).properties(
    title="Total HHS OIG Enforcement Actions per Month Since January 2022",
    width=500,
    height=350
)
```



2. **Plot the number of enforcement actions categorized:**

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

```
df["main_category"] = df["category"].where(
    df["category"].isin(
        ["Criminal and Civil Actions", "State Enforcement Agencies"]
    ),
    other="Other"
)

df[df["main_category"] != "Other"]

category_monthly = (
    df[df["main_category"] != "Other"]
    .groupby(["year_month", "main_category"])
    .size()
    .reset_index(name="number_actions")
)

alt.Chart(category_monthly).mark_line(point=True).encode(
    alt.X("year_month:T", title="Month",  axis=alt.Axis(format="%Y-%m",
 ↪  labelAngle=-45)),
    alt.Y("number_actions:Q", title="Number of enforcement actions"),
    color=alt.Color("main_category:N", title="Category"),
).properties(
    title="Enforcement Actions Over Time by Category",
    width=500,
    height=350
)
```



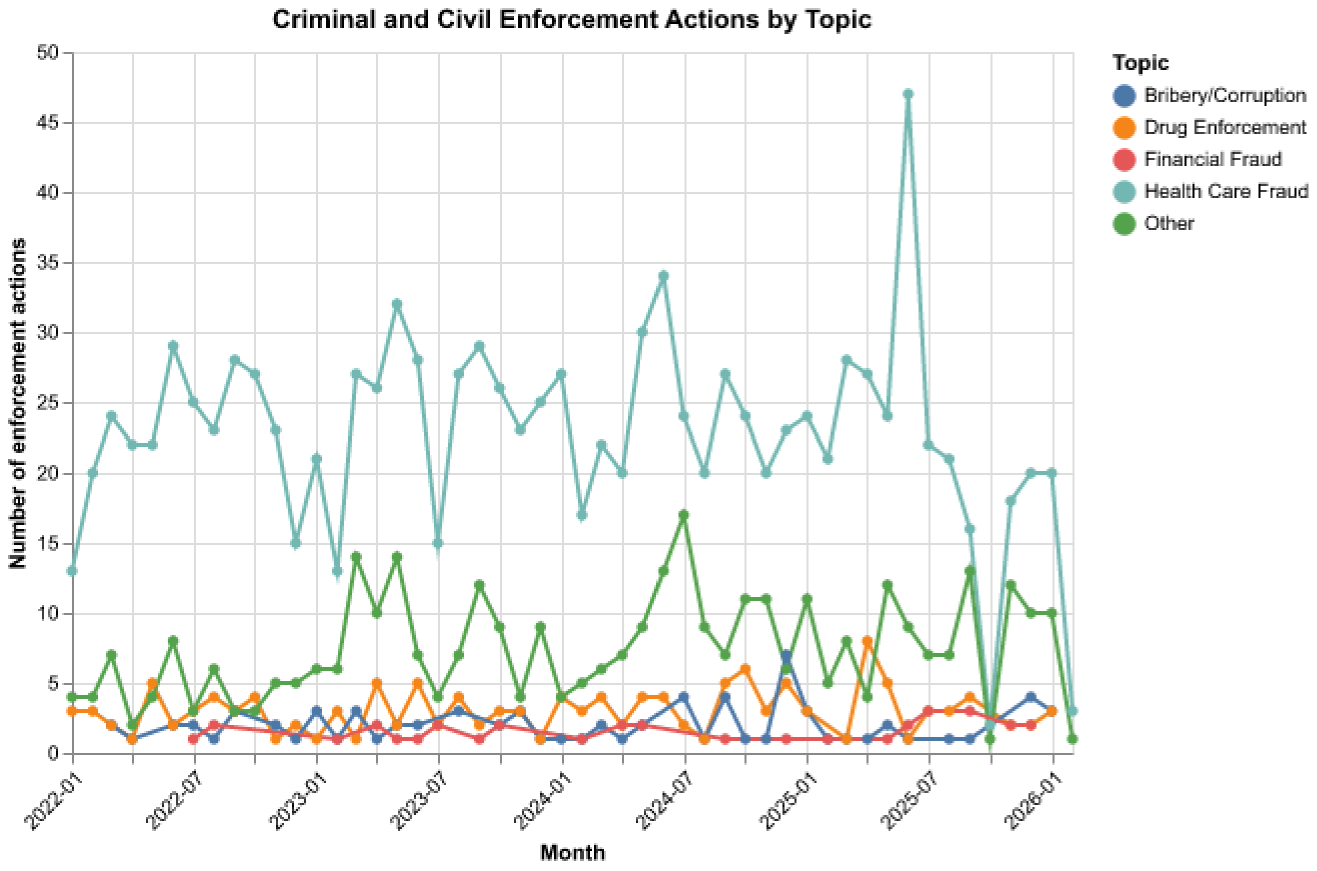
- based on five topics

```python
df_cc = df[df["category"] == "Criminal and Civil Actions"].copy()
df_cc["year_month"] = df_cc["date"].dt.to_period("M").dt.to_timestamp()

def classify_topic(title):
    t = title.lower()

    # Health Care Fraud keywords
    health_words = ["medicare", "medicaid", "hospital", "clinic",
↪   "physician", "medical", "health care", "healthcare", "billing"]

    # Financial Fraud keywords
    financial_words = ["bank", "banker", "financial", "wire fraud", "loan",
↪   "lending", "money laundering", "laundering"]

    # Drug Enforcement keywords
    drug_words = ["drug", "opioid", "fentanyl", "controlled substance",
 ↪  "pharmacy", "pill"]

    # Bribery/Corruption keywords
    corruption_words = ["bribe", "bribery", "corruption", "kickback",
 ↪  "kickbacks"]

    if any(w in t for w in health_words):
        return "Health Care Fraud"
    elif any(w in t for w in financial_words):
        return "Financial Fraud"
    elif any(w in t for w in drug_words):
        return "Drug Enforcement"
    elif any(w in t for w in corruption_words):
        return "Bribery/Corruption"
    else:
        return "Other"

df_cc["topic"] = df_cc["title"].apply(classify_topic)
df_cc["topic"].value_counts()
```

```
topic
Health Care Fraud     1144
Other                  371
Drug Enforcement       143
Bribery/Corruption      78
Financial Fraud         34
Name: count, dtype: int64
```

```
topic_monthly = (
    df_cc
    .groupby(["year_month", "topic"])
    .size()
    .reset_index(name="number_actions")
)
```

```
alt.Chart(topic_monthly).mark_line(point=True).encode(
    alt.X("year_month:T", title="Month", axis=alt.Axis(format="%Y-%m",
 ↪  labelAngle=-45)),
    alt.Y("number_actions:Q", title="Number of enforcement actions"),
    color=alt.Color("topic:N", title="Topic"),
).properties(
    title="Criminal and Civil Enforcement Actions by Topic",
    width=500,
    height=350
)
```


Criminal and Civil Enforcement Actions by Topic
Number of enforcement actions
Month
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other