PS4

Sophiko Skhirtladze

2026-02-07

"This submission is my work alone and complies with the 30538 integrity policy." **S.S**

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


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

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

Step 1: Develop initial scraper and crawler

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

resp = requests.get(BASE_URL, timeout=30, headers={"User-Agent":
 ↪  "Mozilla/5.0"})
resp.raise_for_status()

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

rows = []
for card in soup.select("div.usa-card__container"):
    a = card.select_one("h2.usa-card__heading a")
    if not a:
        continue

    title = a.get_text(strip=True)
    link = urljoin(BASE_URL, a.get("href", ""))

    date_span = card.select_one("div.font-body-sm span")
    date_str = date_span.get_text(strip=True) if date_span else None

    categories = [li.get_text(strip=True) for li in
 ↪  card.select("div.font-body-sm li.usa-tag")]
    category = "; ".join(categories) if categories else None
```

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

df = pd.DataFrame(rows)

df["date"] = pd.to_datetime(df["date"], errors="coerce")

print(df.head())
```

```
                                               title       date  \
0  Houston Transplant Doctor Indicted For Making ... 2026-02-05
1  MultiCare Health System to Pay Millions to Set... 2026-02-04
2  Brooklyn Banker Pleads Guilty to Laundering Pr... 2026-02-03
3  Delafield Man Sentenced to 18 Months' Imprison... 2026-02-03
4  Former NFL Player Convicted for $197M Medicare... 2026-02-03

                    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
- **Inputs:** `start_year`, `start_month`, `run_scraper` (True/False)

1. If `run_scraper` is `False`: do not scrape (return `None`) so Quarto knitting is fast.

2. If `start_year < 2013`: print reminder and return `None` (site only lists enforcement actions after 2013).
3. Define `start_date` = first day of (`start_year`, `start_month`).
4. Initialize `page = 1` and an empty list `rows = []`.
5. **While True:**
    a. Build URL: base page for `page = 1`, otherwise add `?page={page}`
    b. Request HTML, parse with BeautifulSoup
    c. Find all enforcement-action cards (`div.usa-card__container`)
        - If none found: break (no more pages)
    d. For each card:
        - Extract title + link from `h2.usa-card__heading a`
        - Extract date from `div.font-body-sm span`, parse to datetime
        - Extract category tags from `li.usa-tag` (join with `"; "` if multiple)
        - Append dict row to `rows`
    e. Early-stop rule: if the oldest date on this page is older than `start_date`, break.
    f. Set `page += 1`; `sleep(1)` to prevent potential server-side block.
6. Build DataFrame, drop duplicates, filter `date >= start_date`, sort, save CSV as: `enforcement_actions_{start_year}_{start_month:02d}.csv`
7. Return the DataFrame.

- b. Create Dynamic Scraper

```
def scrape_enforcement_actions_since(
    start_year: int,
    start_month: int,
    run_scraper: bool = False,
    sleep_seconds: float = 1.0,
    timeout: int = 30
) -> pd.DataFrame | None:

    if not run_scraper:
        print("run_scraper=False → scraper not running (prevents slow
        ↪ knitting).")
```

```
        return None

    if start_year < 2013:
        print("Please use start_year >= 2013 (only enforcement actions after
         ↪  2013 are listed).")
        return None

    if not (1 <= start_month <= 12):
        raise ValueError("start_month must be between 1 and 12.")

    start_date = datetime(start_year, start_month, 1)

    session = requests.Session()
    session.headers.update({"User-Agent": "Mozilla/5.0"})

    all_rows = []
    page = 1

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

        resp = session.get(url, timeout=timeout)
        resp.raise_for_status()

        soup = BeautifulSoup(resp.text, "html.parser")
        cards = soup.select("div.usa-card__container")


        if not cards:
            break

        page_dates = []

        for card in cards:
            a = card.select_one("h2.usa-card__heading a")
            if not a:
                continue

            title = a.get_text(strip=True)
            link = urljoin(BASE_URL, a.get("href", ""))

            date_span = card.select_one("div.font-body-sm span")
            date_str = date_span.get_text(strip=True) if date_span else None
```

```
            categories = [li.get_text(strip=True) for li in
↪   card.select("div.font-body-sm li.usa-tag")]
            category = "; ".join(categories) if categories else None

            date_dt = pd.to_datetime(date_str, errors="coerce")

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

            if pd.notna(date_dt):
                page_dates.append(date_dt.to_pydatetime())


        if page_dates and min(page_dates) < start_date:
            break

        page += 1
        time.sleep(sleep_seconds)

    df = pd.DataFrame(all_rows)

    df = (
        df.drop_duplicates(subset=["title", "link"])
          .loc[df["date"].notna()]
          .loc[df["date"] >= start_date]
          .sort_values("date", ascending=False)
          .reset_index(drop=True)
    )

    out_csv = f"enforcement_actions_{start_year}_{start_month:02d}.csv"
    df.to_csv(out_csv, index=False)
    print(f"Saved: {out_csv} | Rows: {len(df)}")

    return df
```

- c. Test Your Code

```
RUN_SCRAPER = False
```

```python
df_2024 = scrape_enforcement_actions_since(
    2024, 1,
    run_scraper=RUN_SCRAPER
)

if df_2024 is None:
    df_2024 = pd.read_csv(
        "enforcement_actions_2024_01.csv",
        parse_dates=["date"]
    )

n_2024 = len(df_2024)
print("Number of actions since Jan 2024:", n_2024)

earliest_2024 = df_2024.sort_values("date").iloc[0]
print("\nEarliest scraped action since Jan 2024:")
print(earliest_2024)
```

```
run_scraper=False → scraper not running (prevents slow knitting).
Number of actions since Jan 2024: 1787

Earliest scraped action since Jan 2024:
title       Former Nurse Aide Indicted In Death Of Clarksv...
date                                      2024-01-03 00:00:00
category                             State Enforcement Agencies
link        https://oig.hhs.gov/fraud/enforcement/former-n...
Name: 1786, dtype: object
```

```python
df_2022 = scrape_enforcement_actions_since(2022, 1, run_scraper=RUN_SCRAPER)

if df_2022 is None:
    df_2022 = pd.read_csv("enforcement_actions_2022_01.csv",
 ↪  parse_dates=["date"])

print("\nNumber of actions since Jan 2022:", len(df_2022))
earliest_2022 = df_2022.sort_values("date").iloc[0]
print("\nEarliest scraped action since Jan 2022:")
print(earliest_2022)


print("\nHead of df_2022:")
print(df_2022.head())
```

```
run_scraper=False → scraper not running (prevents slow knitting).

Number of actions since Jan 2022: 3377

Earliest scraped action since Jan 2022:
title       Integrated Pain Management Medical Group Agree...
date                                    2022-01-04 00:00:00
category                              Fraud Self-Disclosures
link        https://oig.hhs.gov/fraud/enforcement/integrat...
Name: 3376, dtype: object

Head of df_2022:
                                                 title       date  \
0  Houston Transplant Doctor Indicted For Making ... 2026-02-05
1  MultiCare Health System to Pay Millions to Set... 2026-02-04
2  Brooklyn Banker Pleads Guilty to Laundering Pr... 2026-02-03
3  Former NFL Player Convicted for $197M Medicare... 2026-02-03
4  Attorney General Hanaway Obtains Medicaid Frau... 2026-02-03

                    category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2                   COVID-19
3  Criminal and Civil Actions
4  State Enforcement Agencies

                                                 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/former-n...
4  https://oig.hhs.gov/fraud/enforcement/attorney...
```

Step 3: Plot data based on scraped data

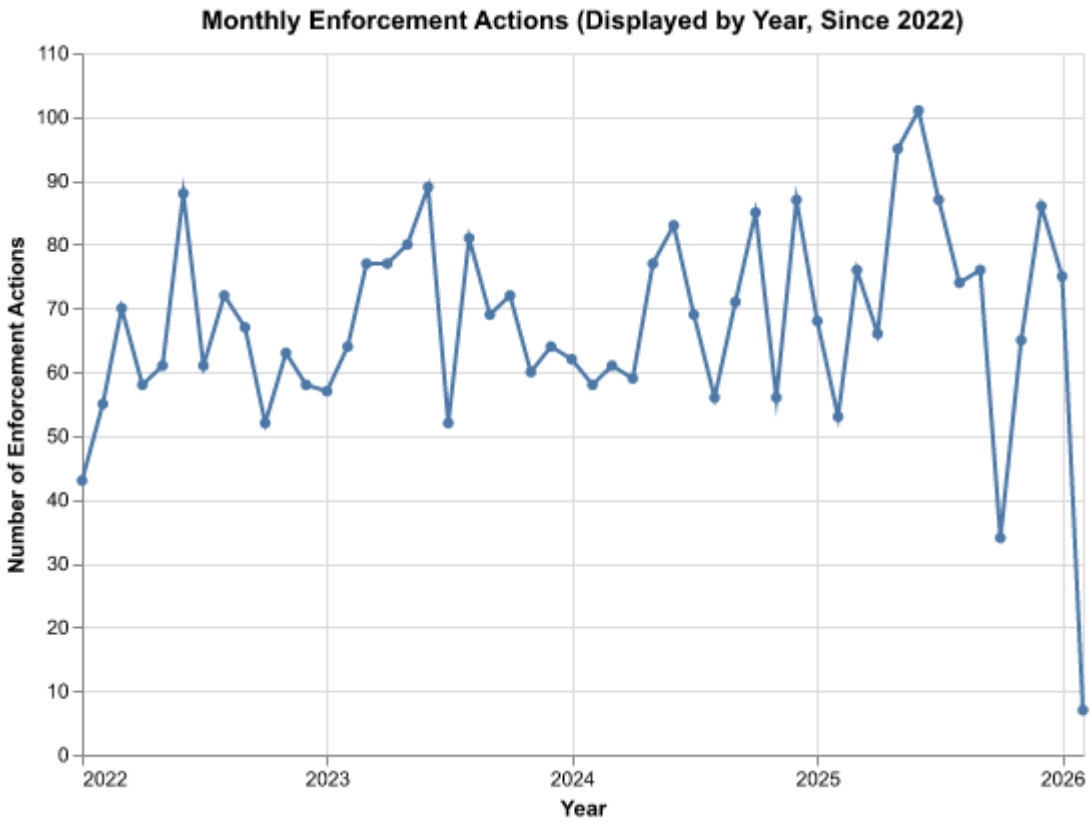
1. Plot the number of enforcement actions over time

```python
df_2022["year_month"] = df_2022["date"].dt.to_period("M").dt.to_timestamp()
monthly_counts = (
    df_2022
    .groupby("year_month")
```

```
    .size()
    .reset_index(name="n_actions")
)
```

```
line_chart = (
    alt.Chart(monthly_counts)
    .mark_line(point=True)
    .encode(
        alt.X(
            "year_month:T",
            title="Year",
            axis=alt.Axis(format="%Y", tickCount="year")
        ),
        alt.Y(
            "n_actions:Q",
            title="Number of Enforcement Actions"
        )
    )
    .properties(
        width=500,
        height=350,
        title="Monthly Enforcement Actions (Displayed by Year, Since 2022)"
    )
)

line_chart
```



2. Plot the number of enforcement actions categorized:

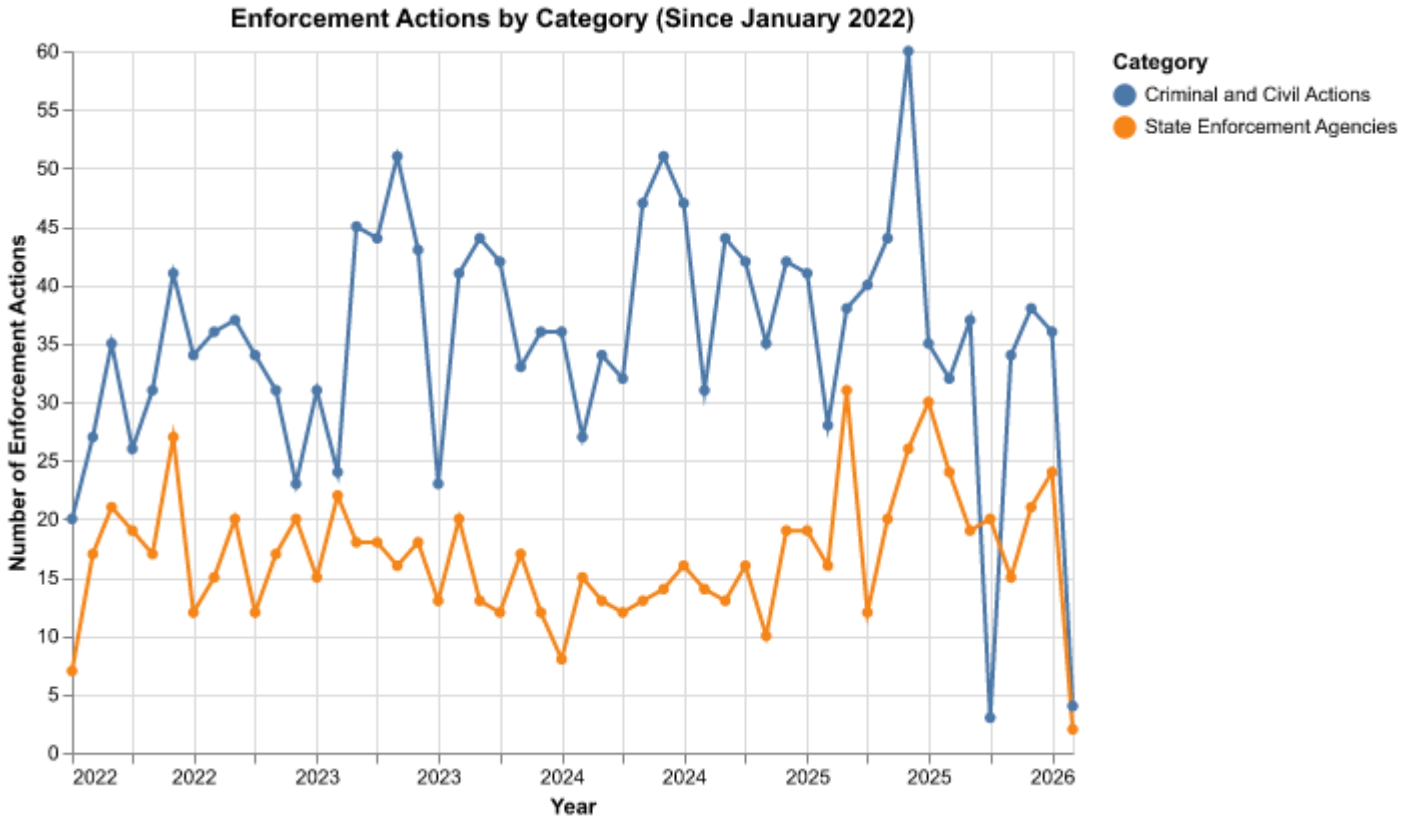
- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"

```
category_subset = df_2022[
    df_2022["category"].isin([
        "Criminal and Civil Actions",
        "State Enforcement Agencies"
    ])
]

category_monthly = (
    category_subset
    .groupby(["year_month", "category"])
    .size()
    .reset_index(name="n_actions")
)
```

```
category_chart = (
    alt.Chart(category_monthly)
    .mark_line(point=True)
    .encode(
        alt.X(
            "year_month:T",
            title="Year",
            axis=alt.Axis(format="%Y")
        ),
        alt.Y(
            "n_actions:Q",
            title="Number of Enforcement Actions"
        ),
        alt.Color(
            "category:N",
            title="Category"
        )
    )
    .properties(
        width=500,
        height=350,
        title="Enforcement Actions by Category (Since January 2022)"
    )
)

category_chart
```



- based on five topics

```
criminal_df = df_2022[
    df_2022["category"] == "Criminal and Civil Actions"
].copy()
```

```
def classify_topic_v2(title: str) -> str:
    t = title.lower()

    # 1. HEALTH CARE FRAUD
    if any(k in t for k in [
        "medicare", "medicaid", "health care", "healthcare",
        "hospital", "physician", "doctor", "nurse",
        "clinic", "patient", "medical", "health",
        "home health", "hospice", "laboratory", "lab", "nursing",
        "testing", "therapy", "treatment",
        "durable medical", "dme",
        "billing", "claims", "reimbursement",
```

```
        "telemedicine", "telehealth",
        "covid", "covid-19", "pandemic",
        "kickback"
    ]):
        return "Health Care Fraud"

    # 2. DRUG ENFORCEMENT
    if any(k in t for k in [
        "opioid", "controlled substance", "drug trafficking",
        "distribution", "pharmacy", "prescription", "drug",
        "pill mill"
    ]):
        return "Drug Enforcement"

    # 3. FINANCIAL FRAUD
    if any(k in t for k in [
        "bank", "financial", "wire fraud", "money laundering",
        "loan", "securities", "investment", "mortgage", "tax evasion"
    ]):
        return "Financial Fraud"

    # 4. BRIBERY / CORRUPTION
    if any(k in t for k in [
        "bribe", "bribery", "corruption"
    ]):
        return "Bribery/Corruption"

    # 5. OTHER (residual)
    return "Other"
```

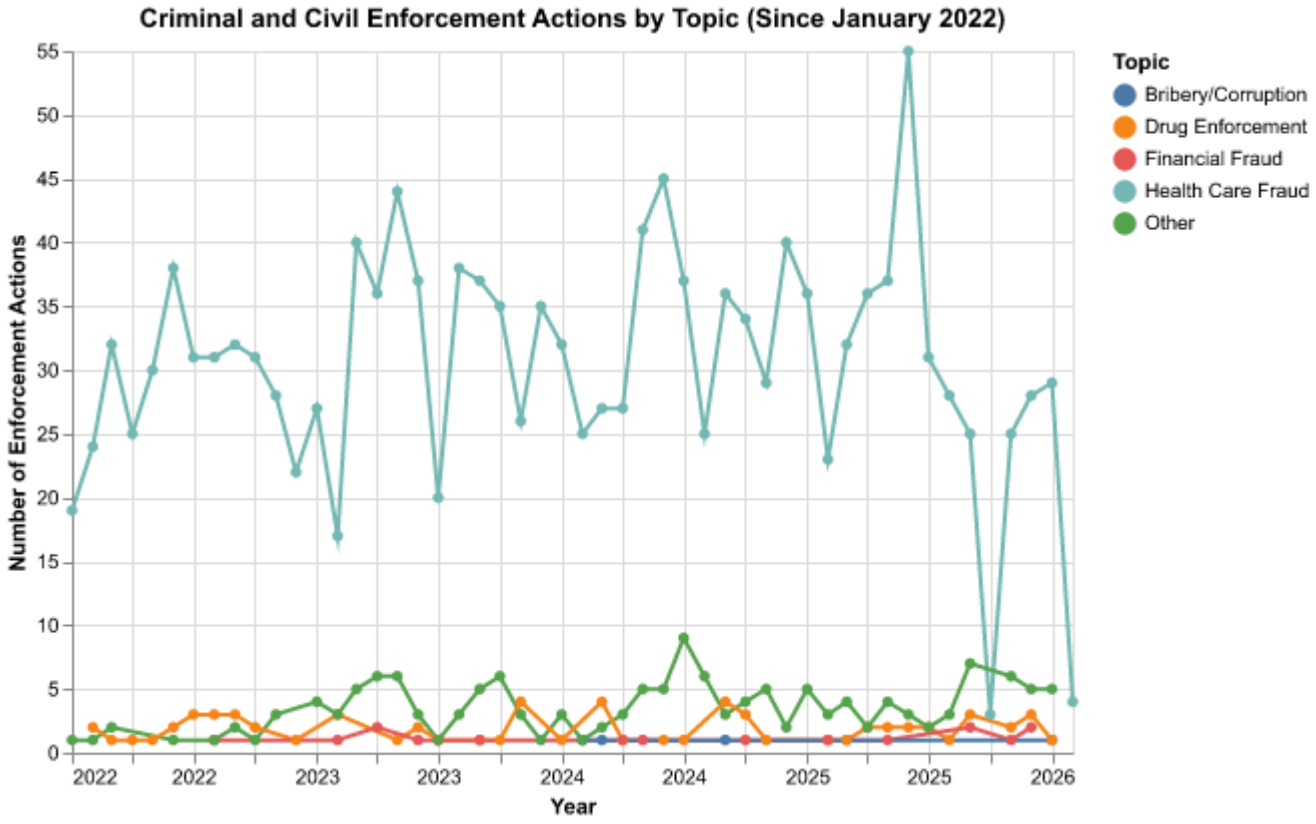
```
criminal_df["topic"] = criminal_df["title"].apply(classify_topic_v2)
```

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

```
topic_chart = (
    alt.Chart(topic_monthly)
```

```
    .mark_line(point=True)
    .encode(
        alt.X(
            "year_month:T",
            title="Year",
            axis=alt.Axis(format="%Y")
        ),
        alt.Y(
            "n_actions:Q",
            title="Number of Enforcement Actions"
        ),
        alt.Color(
            "topic:N",
            title="Topic"
        )
    )
    .properties(
        width=500,
        height=350,
        title="Criminal and Civil Enforcement Actions by Topic (Since January
 ↪  2022)"
    )
)

topic_chart
```


Criminal and Civil Enforcement Actions by Topic (Since January 2022)
Number of Enforcement Actions
55
50
45
40
35
30
25
20
15
10
5
0
2022
2022
2023
2023
2024
2024
2025
2025
2026
Year
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other