PS4 Solution

Peter Ganong, Maggie Shi, Richard Chen

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: **___**

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

1. Scraping

```
import requests
from bs4 import BeautifulSoup

# Read data
def get_soup(url):
  response = requests.get(url)
  return BeautifulSoup(response.text, 'lxml')

url = 'https://oig.hhs.gov/fraud/enforcement/'
soup = get_soup(url)

# Get titles and links
h2s = soup.find_all('h2')
title_data = []
link_data = []
for el in h2s:
  if (el.get('class') == ['usa-card__heading']):
    title_data.append(el.text.replace('\n', ''))
    link_data.append("https://oig.hhs.gov" + el.find_all('a',
 ↪  href=True)[0].get('href'))

# Get date data
spans = soup.find_all('span')
date_data = []
for el in spans:
  if (el.get('class') == ['text-base-dark', 'padding-right-105']):
```

```
    date_data.append(el.text)

# Get category data
uls = soup.find_all('ul')
cat_data = []
for el in uls:
  if (el.get('class') == ['display-inline', 'add-list-reset']):
    cat_text = el.find_all('li')[0].text
    cat_data.append(cat_text)

df = pd.DataFrame({
  'Title': title_data,
  'Date': date_data,
  'Category': cat_data,
  'Link': link_data
})

print(df.head(5))
```

```
                                               Title              Date  \
0  Yadkinville Woman Sentenced in Connection with...  January 29, 2026
1  Slidell Chiropractor Sentenced for Health Care...  January 28, 2026
2  Repeat Health Care Fraud Offender Sentenced fo...  January 28, 2026
3  Scranton Heart Institute Agrees To Pay $48,709...  January 28, 2026
4  Rheumatologist Agrees To Resolve False Claims ...  January 28, 2026

                    Category  \
0  Criminal and Civil Actions
1                    COVID-19
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                Link
0  https://oig.hhs.gov/fraud/enforcement/yadkinvi...
1  https://oig.hhs.gov/fraud/enforcement/slidell-...
2  https://oig.hhs.gov/fraud/enforcement/repeat-h...
3  https://oig.hhs.gov/fraud/enforcement/scranton...
4  https://oig.hhs.gov/fraud/enforcement/rheumato...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

* a. Pseudo-Code

The function takes starting year and starting month as an input.

- First, check if the starting year is valid (> 2013). If not, print an error message and exit the function.
- Initialize a `date` object representing the first day of the start month and year (`start_date`), and initialize an empty list to store the scrapped data (`enforcement_data`).
- Loop through pages 1 to 480 (the maximum number of pages currently available on the website).
- For each page:

  - Get the HTML content of the page.
  - Parse the content to find all enforcement actions on the page.
  - For each enforcement action found on the page:

    - Extract the title, action date, category, and link of the action.
    - Extract the agency information (we can use the function from previous step).

- Check date validity. If action date is earlier than `start_date`, stop the loop and return `enforcement_data` (ending the function). If not, continue looping and append the newly scrapped action into the `enforcement_data`.
- Pause for 1 second between each page request to prevent overloading the server.
- After looping through all pages (or stopping early if an action date is older than `start_date`), return `enforcement_data`.

As we can see from the pseudo-code above, we used `for` loop to loop through all pages. Other than that, we can also use `while` loop to do the same thing. We can set a flag to `True` to keep going to the next page, and when action date is earlier than `start_date`, set the flag to `False` that effectively ends the loop and return the result.

* b. Create Dynamic Scraper

::: {.cell execution_count=3}
``` {.python .cell-code}
from datetime import datetime
```

```
def scrape_actions(start_year, start_month, run_ind=False):
  if not run_ind:
    print("Run indicator is false")
    return
  if start_year < 2013:
    print("Please use a year >= 2013.")
    return []

  start_date = datetime(start_year, start_month, 1)
  enforcement_data = []

  for page in range(1, 481): # As per today, there are 480 pages in the url
    soup = get_soup(f"https://oig.hhs.gov/fraud/enforcement/?page={page}")

    actions = soup.find_all("header", class_="usa-card__header")
    for action in actions:
      title = action.find("h2").text.replace('\n', '')
      date_text = action.find("span", class_="text-base-dark").text
      category = action.find("li", class_="usa-tag").text
      full_link = "https://oig.hhs.gov" + action.find("a").get("href")
      agency = get_agency(full_link)

      action_date = datetime.strptime(date_text, "%B %d, %Y")

      if action_date < start_date:
        break
      else:
        enforcement_data.append({"Title": title,
                                 "Date": action_date,
                                 "Category": category,
                                 "Link": full_link,
                                 "Agency": agency})

    time.sleep(1)

  filename = f"enforcement_actions_{start_year}_{start_month}.csv"
  df = pd.DataFrame(enforcement_data)

  if not df.empty:
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")
  else:
    print("No data to save.")
```

```
  return None
```

:::

```
# Will take a long time to run
scrape_actions(2024, 1)
```

There are +/- 1756 actions, with the earliest being "Former Nurse Aide Indicted In Death Of Clarksville Patient Arrested In Georgia", published on 3 Jan 2024 under "State Enforcement Agencies", with Tennessee Attorney General as the agency.

- c. Test Your Code

```
# Will take a long time to run
scrape_actions(2022, 1)
```

There are +/- 3346 actions, with the earliest being "Integrated Pain Management Medical Group Agreed to Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded Individuals", published on 4 Jan 2022 under "Fraud Self-Disclosures".

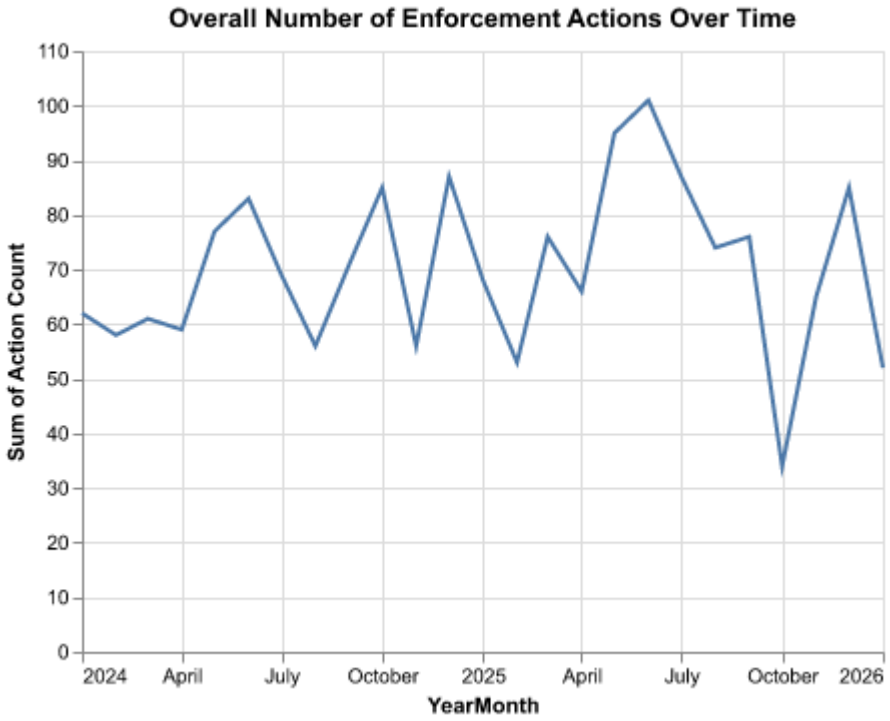
Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df = pd.read_csv("enforcement_actions_2024_1.csv")
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df['YearMonth'] = df['Date'].dt.to_period('M').dt.to_timestamp()
monthly_data = df.groupby(['YearMonth', 'Category']).agg({
                    'Title': 'count'}).reset_index().rename(columns={'Title':
                     ↪  'Action Count'})

overall_chart = alt.Chart(monthly_data).mark_line().encode(
  x='YearMonth:T',
  y='sum(Action Count):Q'
).properties(
  title="Overall Number of Enforcement Actions Over Time",
  width=400,
  height=300
)
```

```
overall_chart
```

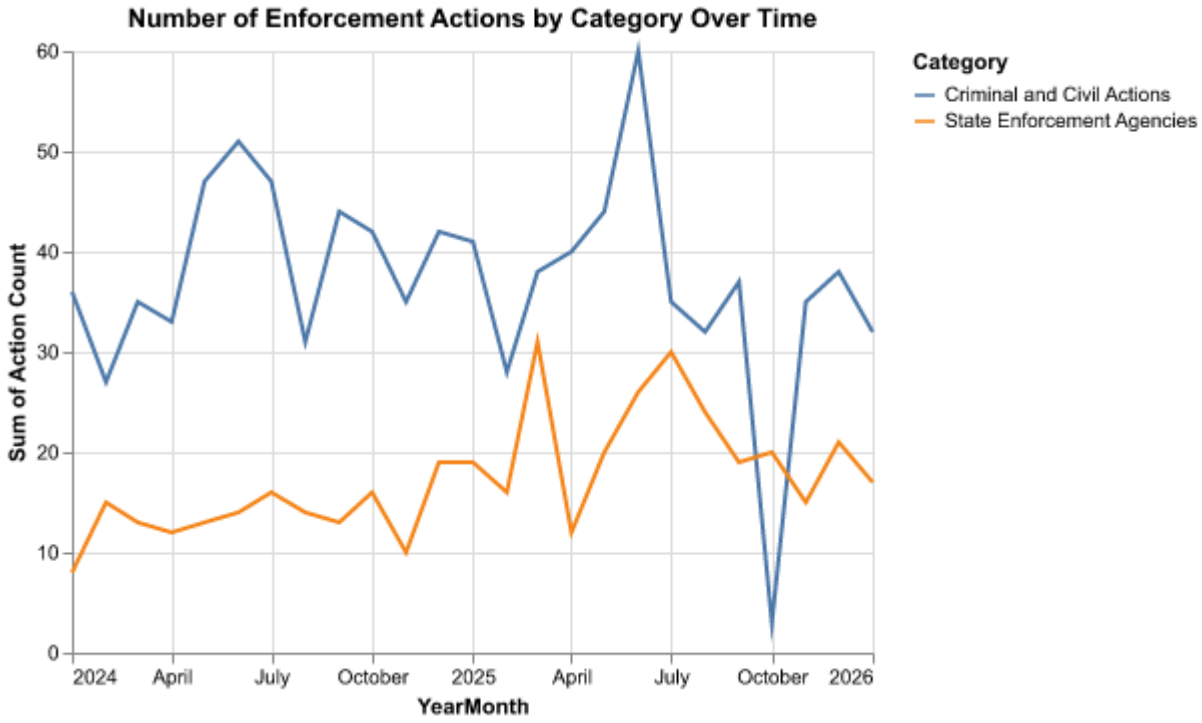


2. Plot the number of enforcement actions categorized:

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

```
filtered_data = monthly_data[monthly_data['Category'].isin(
              ['Criminal and Civil Actions', 'State Enforcement Agencies'])]
category_chart = alt.Chart(filtered_data).mark_line().encode(
  x='YearMonth:T',
  y='sum(Action Count):Q',
  color='Category:N'
).properties(
  title="Number of Enforcement Actions by Category Over Time",
  width=400,
  height=300
)

category_chart
```



- based on five topics

```
def categorize_topic(title):
  title = title.lower()

  # Check for Health Care Fraud
  if "health care" in title or "medicare" in title or "medicaid" in title:
    return "Health Care Fraud"
  # Check for Financial Fraud
  elif "financial" in title or "wire" in title or "bank" in title or "fraud"
   ↪  in title:
    return "Financial Fraud"
  # Check for Drug Enforcement
  elif "drug" in title or "opioid" in title or "narcotics" in title:
    return "Drug Enforcement"
  # Check for Bribery/Corruption
  elif "bribery" in title or "corruption" in title:
    return "Bribery/Corruption"
  else:
    return "Other"

df['Topic'] = df['Title'].apply(categorize_topic)
```

```
monthly_data = df.groupby(['YearMonth', 'Category', 'Topic']).agg({
                    'Title': 'count'}).reset_index().rename(columns={'Title':
                     ↪  'Action Count'})
filtered_data = monthly_data[monthly_data['Category'].isin(
                ['Criminal and Civil Actions'])]
topic_chart = alt.Chart(filtered_data).mark_line().encode(
  x='YearMonth:T',
  y='sum(Action Count):Q',
  color='Topic:N'
).properties(
  title="Number of Enforcement Actions by Topic Over Time",
  width=400,
  height=300
)

topic_chart
```