PS4

Ruilin Yao

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: YRL

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 requests
from bs4 import BeautifulSoup as bs
import os

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

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

Step 1: Develop initial scraper and crawler

```python
data = []
url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
soup = bs(response.content, 'html.parser')
results = soup.find_all('li', class_='usa-card card--list pep-card--minimal
 ↪  mobile:grid-col-12')
for result in results:
    title = result.find('a').text.strip()
    date = result.find('span', class_='text-base-dark
 ↪  padding-right-105').text.strip()
    cat = result.find('li', class_='display-inline-block usa-tag
 ↪  text-no-lowercase text-base-darkest bg-base-lightest
 ↪  margin-right-1').text.strip()
    link = "https://oig.hhs.gov" + result.find('a')['href']
    data.append({'title': title, 'date': date, 'category': cat, 'link':
 ↪  link})
df = pd.DataFrame(data)
df.head()
```

	title	date	category	lin
0	Houston Transplant Doctor Indicted For Making ...	February 5, 2026	Criminal and Civil Actions	htt
1	MultiCare Health System to Pay Millions to Set...	February 4, 2026	Criminal and Civil Actions	htt
2	Brooklyn Banker Pleads Guilty to Laundering Pr...	February 3, 2026	COVID-19	htt
3	Delafield Man Sentenced to 18 Months' Imprison...	February 3, 2026	Criminal and Civil Actions	htt

	title	date	category	lin
4	Former NFL Player Convicted for $197M Medicare...	February 3, 2026	Criminal and Civil Actions	htt

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code if there is a file that already exists, return the file else, find the total page it have, for loop to go through each page, find all the enforcement actions, check the date of each enforcement action, if the date is after the input date, add it to the dataframe, else, break the loop and save the dataframe as a csv file.
- b. Create Dynamic Scraper

```python
def create_df(year, month):
    if year < 2013:
        print('Year must be greater than or equal to 2013, since only
         ↪  enforcement actions after 2013 are listed.')
        return None
    file_path = f'enforcement_actions_{year}_{month}.csv'
    if os.path.exists(file_path):
        df = pd.read_csv(file_path)
        return df
    data = []
    url = 'https://oig.hhs.gov/fraud/enforcement/'
    response = requests.get(url)
    soup = bs(response.content, 'html.parser')
    page_all = soup.find_all('a', class_='pagination__link')
    total_page = page_all[-1].text.strip()
    for i in range(1, int(total_page)+1):
        url = 'https://oig.hhs.gov/fraud/enforcement/?page=' + str(i)
        print(f'The {i} page URL: {url}')
        response = requests.get(url)
        soup = bs(response.content, 'html.parser')
        results = soup.find_all('li', class_='usa-card card--list
↪   pep-card--minimal mobile:grid-col-12')
        day_check = soup.find('span', class_='text-base-dark
↪   padding-right-105').text.strip()
        if pd.to_datetime(day_check) < pd.to_datetime(f'{year}-{month}-01'):
            break
        for result in results:
            title = result.find('a').text.strip()
            date = result.find('span', class_='text-base-dark
↪   padding-right-105').text.strip()
            if pd.to_datetime(date) < pd.to_datetime(f'{year}-{month}-01'):
                break
            cat = result.find('li', class_='display-inline-block usa-tag
↪   text-no-lowercase text-base-darkest bg-base-lightest
↪   margin-right-1').text.strip()
            link = "https://oig.hhs.gov" + result.find('a')['href']
            data.append({'title': title, 'date': date, 'category': cat,
↪   'link': link})
        time.sleep(1)
    df = pd.DataFrame(data)
    df.to_csv(f'enforcement_actions_{year}_{month}.csv', index=False,
↪   encoding='utf-8')
    return df
df_2024_01 = create_df(2024, 1)
print(f'There are {len(df_2024_01)} rows in the dataset.')
print('The detail of the ealiest data is:')
print(df_2024_01.tail(1).to_string())
```

```
There are 1787 rows in the dataset.
The detail of the ealiest data is:

                                                                                           title
                                                                                           date
                                                                                           category
                                                                                           link
1786  Former Nurse Aide Indicted In Death Of Clarksville Patient Arrested In
Georgia  January 3, 2024  State Enforcement Agencies
https://oig.hhs.gov/fraud/enforcement/former-nurse-aide-indicted-in-death-of-clarksville-pat
```

- c. Test Your Code

```
df_2022_01 = create_df(2022, 1)
print(f'There are {len(df_2022_01)} rows in the dataset.')
print('The detail of the ealiest data is:')
print(df_2022_01.tail(1).to_string())
```

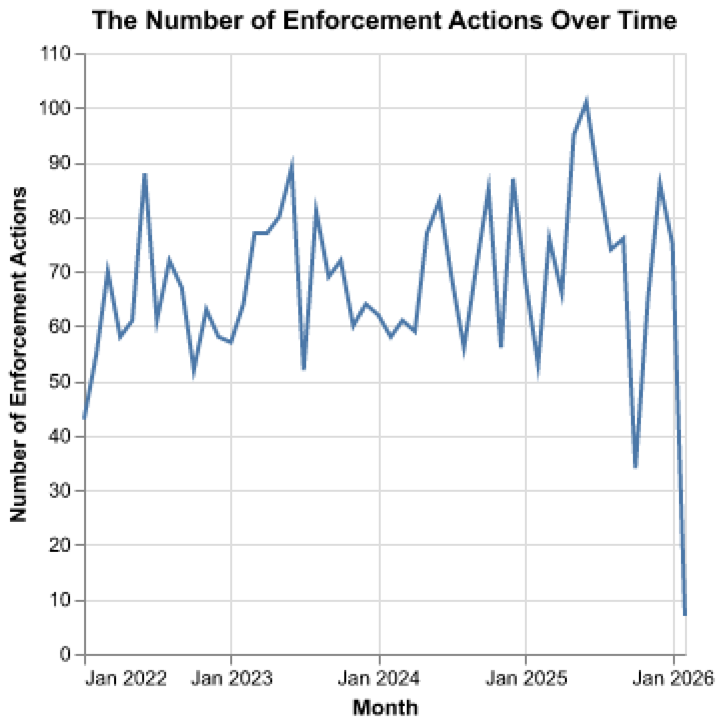
```
There are 3377 rows in the dataset.
The detail of the ealiest data is:
```

```
3376  Integrated Pain Management Medical Group Agreed to Pay $10,000 for
Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded
Individuals  January 4, 2022  Fraud Self-Disclosures
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

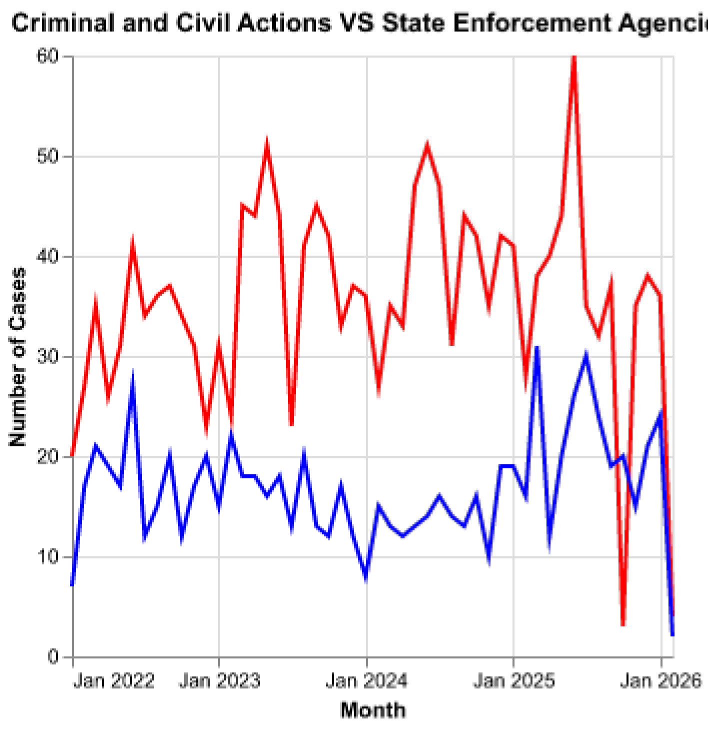
```
df_2022_01['date'] = pd.to_datetime(df_2022_01['date'])
alt.Chart(df_2022_01, title="The Number of Enforcement Actions Over
↪   Time").mark_line().encode(
    x = alt.X('yearmonth(date):T', title='Month'),
    y = alt.Y('count(title):Q', title='Number of Enforcement Actions'),
)
```



2. Plot the number of enforcement actions categorized:

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

```
df_2022_01_fillter = df_2022_01[df_2022_01['category'].isin(['Criminal and
↪  Civil Actions', 'State Enforcement Agencies'])]
base = alt.Chart(df_2022_01_fillter, title="Criminal and Civil Actions VS
↪  State Enforcement Agencies").encode(
    x = alt.X('yearmonth(date):T', title='Month'),
)
line1 = base.transform_filter(
    alt.datum.category == 'Criminal and Civil Actions'
).mark_line(color='red').encode(
    y=alt.Y('count():Q', title='Number of Cases')
)
line2 = base.transform_filter(
    alt.datum.category == 'State Enforcement Agencies'
).mark_line(color='blue').encode(
    y=alt.Y('count():Q', title='Number of Cases')
)
line1 + line2
```



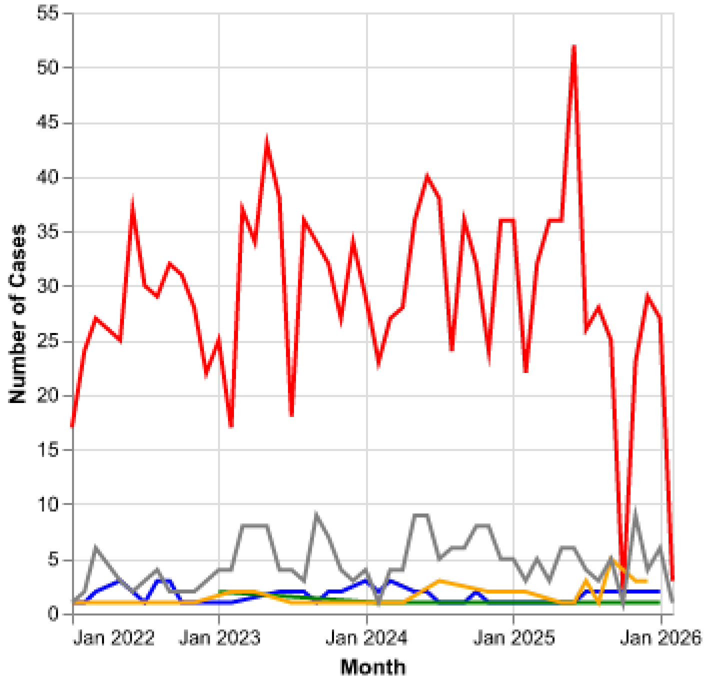
- based on five topics

```python
category_keywords = {
    "Health Care Fraud": [
        "medicare", "medicaid", "healthcare", "health care",
         ↪  "pharmaceutical",
        "prescription", "kickback", "upcoding", "billing fraud", "false
         ↪  claims",
        "medical", "hospital", "clinic", "physician", "doctor", "nursing
         ↪  home",
        "dme", "durable medical equipment", "lab", "laboratory", "therapy"
    ],
    "Financial Fraud": [
        "securities", "insider trading", "ponzi", "investment fraud", "bank
         ↪  fraud",
        "mortgage fraud", "tax fraud", "irs", "money laundering", "wire
         ↪  fraud",
        "mail fraud", "credit card", "identity theft", "forex",
         ↪  "commodities",
        "embezzlement", "accounting fraud", "financial statement", "loan
         ↪  fraud"
    ],
    "Drug Enforcement": [
        "drug trafficking", "narcotics", "controlled substance", "cocaine",
        "heroin", "methamphetamine", "fentanyl", "opioid", "distribution",
        "possession with intent", "conspiracy to distribute", "cartel",
        "pill mill", "prescription drug", "pharmacy", "drug conspiracy"
    ],
    "Bribery/Corruption": [
        "bribery", "corruption", "kickback", "extortion", "racketeering",
        "rico", "public corruption", "official misconduct", "pay to play",
        "foreign corrupt practices", "fcpa", "bid rigging", "contract fraud",
        "influence peddling", "gratuity", "illegal gratuity"
    ]
}

def classify_title(title):
    title_lower = title.lower()
    for category, keywords in category_keywords.items():
        for keyword in keywords:
            if keyword in title_lower:
                return category
    return "Other"
```

```python
df_2022_01_cc =  df_2022_01[df_2022_01['category'] == 'Criminal and Civil
↪  Actions']
df_2022_01_cc['category'] = df_2022_01_cc['title'].apply(classify_title)
base = alt.Chart(df_2022_01_cc, title="Criminal and Civil Actions VS State
↪  Enforcement Agencies").encode(
    x = alt.X('yearmonth(date):T', title='Month'),
)
line1 = base.transform_filter(
    alt.datum.category == 'Health Care Fraud'
).mark_line(color='red').encode(
    y=alt.Y('count():Q', title='Number of Cases')
)
line2 = base.transform_filter(
    alt.datum.category == 'Drug Enforcement'
).mark_line(color='blue').encode(
    y=alt.Y('count():Q', title='Number of Cases')
)
line3 = base.transform_filter(
    alt.datum.category == 'Bribery/Corruption'
).mark_line(color='green').encode(
    y=alt.Y('count():Q', title='Number of Cases')
)
line4 = base.transform_filter(
    alt.datum.category == 'Financial Fraud'
).mark_line(color='orange').encode(
    y=alt.Y('count():Q', title='Number of Cases')
)
line5 = base.transform_filter(
    alt.datum.category == 'Other'
).mark_line(color='gray').encode(
    y=alt.Y('count():Q', title='Number of Cases')
)
line1 + line2 + line3 + line4 + line5
```

Criminal and Civil Actions VS State Enforcement Agencies


Number of Cases
Jan 2022
Jan 2023
Jan 2024
Jan 2025
Jan 2026
Month