

Problem Set 4

Andrew Doherty Munro

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: ADM

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/hWhcHqH>
 - 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

```

import requests
from bs4 import BeautifulSoup

url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
html = response.text

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

```

Grab all enforcement action cards, which are denoted by “li”:

```

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

data = []
for act in enforce_acts:

    head = act.find('h2', class_='usa-card__heading')
    if head and head.find('a'):
        title = head.find('a').text.strip()
        link = head.find('a')['href']

        if link.startswith('/'):
            link = 'https://oig.hhs.gov' + link
        else:
            continue

    date_info = act.find('span', class_ = 'text-base-dark')

```

```

date = date_info.text.strip() if date_info else None

tag = act.find('li', class_='display-inline-block')
category = tag.text.strip() if tag else None

data.append({
    'Title': title,
    'Date': date,
    'Category': category,
    'Link': link
})

enforcement_data = pd.DataFrame(data)

enforcement_data.head()

```

	Title	Date	Category	Link
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
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

Function: `dynamic_scrape(month, year)`

Purpose: scrape all enforcement actions from a given date to the present

Input: Month and Year (in string format)

Output: Dataframe with Title, Category, Date, and link to enforcement action

Step 1: start with base URL and send GET request

Step 2: Parse HTML with BeautifulSoup as in part 1, get all enforcement actions

Step 3: Convert to dataframe format as above

Step 4: Check the last date pulled by selecting the last row of the dataframe. If this date is more recent than the date input, enter a while loop.

Step 5: The while loop will pull all enforcement actions from the next page using an f-string for the url. It will continue to do so for each page, append the dataframe, and update the

page counter by one until the condition `end_date > last_date` is no longer satisfied

Step 6: Once the last action pulled on a page is further back than our desired end date, exit the loop.

Step 7: Convert the dataframe's dates to datetime format, and filter out the dates that were pulled outside of our range.

Step 8: Return a tidy dataframe

- b. Create Dynamic Scraper

```
from datetime import datetime

def dynamic_scraper(date):

    end_date = datetime.strptime(date, '%B %Y')
    if end_date < datetime(2013, 1, 1):
        print("Please restrict date to 2013 or more recent; enforcement actions
        ↪ prior to 2013 not listed")
        return
    else:
        pass

    data = []
    url = 'https://oig.hhs.gov/fraud/enforcement/'
    response = requests.get(url)
    html = response.text

    soup = BeautifulSoup(html, "html.parser")
    enforce_acts = soup.find_all('li', class_ = 'usa-card')

    for act in enforce_acts:

        head = act.find('h2', class_='usa-card_heading')
        if head and head.find('a'):
            title = head.find('a').text.strip()
            link = head.find('a')['href']

            if link.startswith('/'):
                link = 'https://oig.hhs.gov' + link
            else:
                continue

        date_info = act.find('span', class_ = 'text-base-dark')
        date = date_info.text.strip() if date_info else None
```

```

tag = act.find('li', class_='display-inline-block')
category = tag.text.strip() if tag else None

data.append({
    'Title': title,
    'Date': date,
    'Category': category,
    'Link': link
})

enforcement_data = pd.DataFrame(data)

last_date = enforcement_data.iloc[-1]['Date']
last_date = datetime.strptime(last_date, '%B %d, %Y')

page = 2

while last_date > end_date:
    data = []
    url = f'https://oig.hhs.gov/fraud/enforcement/?page={page}'
    response = requests.get(url)
    html = response.text

    soup = BeautifulSoup(html, "html.parser")
    enforce_acts = soup.find_all('li', class_ = 'usa-card')

    for act in enforce_acts:

        head = act.find('h2', class_='usa-card__heading')
        if head and head.find('a'):
            title = head.find('a').text.strip()
            link = head.find('a')['href']

            if link.startswith('/'):
                link = 'https://oig.hhs.gov' + link
            else:
                continue

        date_info = act.find('span', class_ = 'text-base-dark')
        date = date_info.text.strip() if date_info else None

        tag = act.find('li', class_='display-inline-block')

```

```

        category = tag.text.strip() if tag else None

        data.append({
            'Title': title,
            'Date': date,
            'Category': category,
            'Link': link
        })

    data = pd.DataFrame(data)
    enforcement_data = pd.concat([enforcement_data, data], ignore_index=True)

    last_date = enforcement_data.iloc[-1]['Date']
    last_date = datetime.strptime(last_date, '%B %d, %Y')

    page += 1
    time.sleep(1)

    enforcement_data['Date'] = pd.to_datetime(enforcement_data['Date'],
    ↪ format='%B %d, %Y')
    enforcement_data = enforcement_data[enforcement_data['Date'] >= end_date]

    return enforcement_data

```

- c. Test Your Code

```

RUN_SCRAPER = False

if RUN_SCRAPER:

    df = dynamic_scraper('January 2024')

else:
    pass

```

This pulled 1787 enforcement actions. The last one is from 01-03-2024 and is titled “Former Nurse Aide Indicted In Death Of Clarksville Patient Arrested In Georgia”

Now, January 2022:

```

RUN_SCRAPER = False

if RUN_SCRAPER:

    data = dynamic_scraper('January 2022')

    data.to_csv('enforcement_actions_year_month.csv')

else:

    data = pd.read_csv('enforcement_actions_year_month.csv')

```

```

print(len(data))
print(data.iloc[-1]['Date'])

```

```

3377
2022-01-04

```

This pulled 3377 enforcement actions dating back to 1/4/22

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

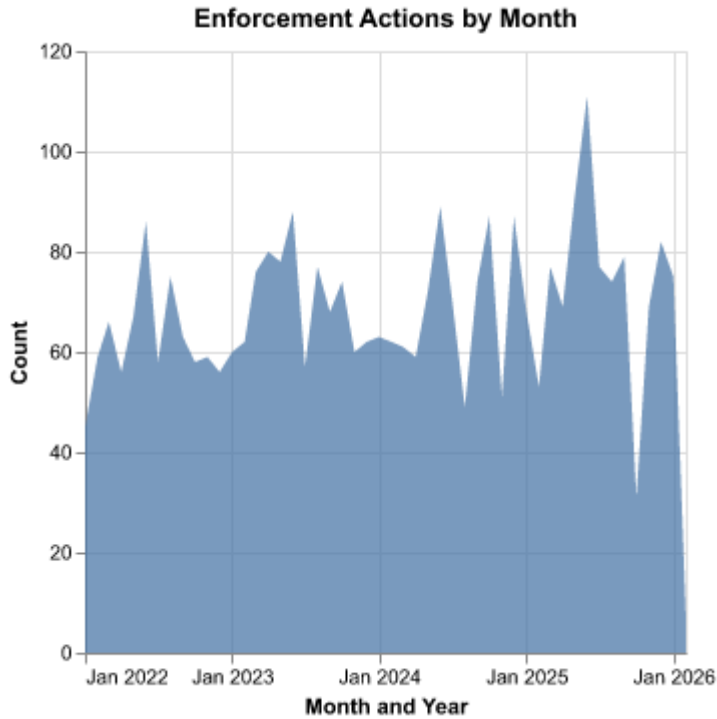
```

actions_by_month = alt.Chart(data, title='Enforcement Actions by
↪ Month').mark_area(opacity=0.75).encode(

    x = alt.X('yearmonth(Date):T', title = 'Month and Year'),
    y = alt.Y('count()', title = 'Count')
)

actions_by_month

```



2. Plot the number of enforcement actions categorized:

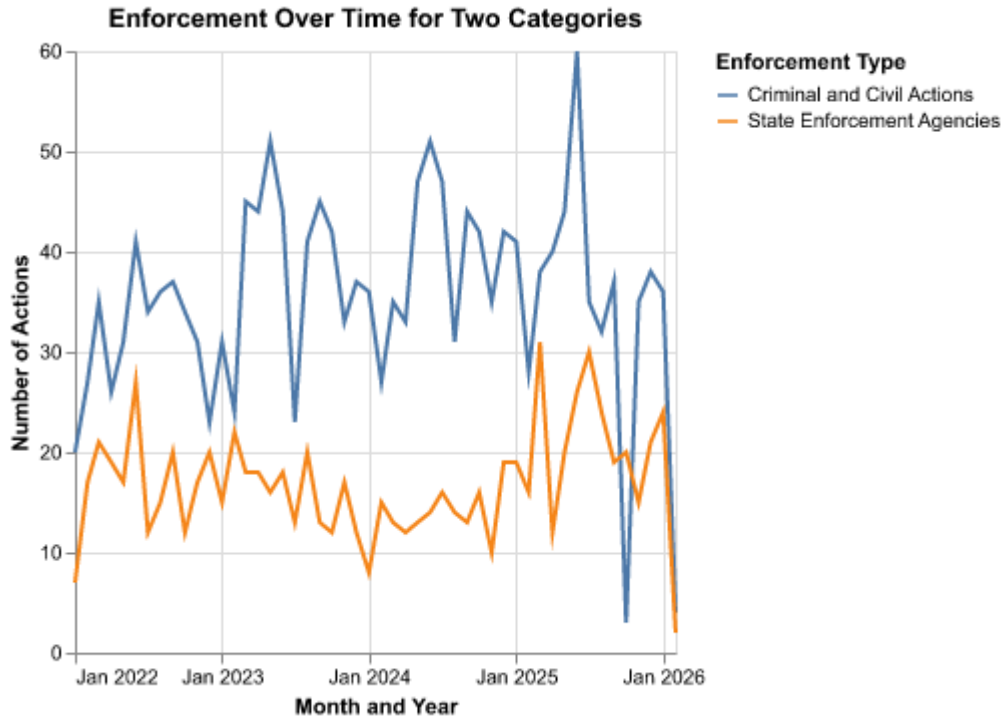
- based on “Criminal and Civil Actions” vs. “State Enforcement Agencies”

```
cca_sea = data[data['Category'].isin(['Criminal and Civil Actions', 'State
↪ Enforcement Agencies'])]

cca_sea['Date'] = pd.to_datetime(cca_sea['Date'])

cca_sea_plot = alt.Chart(cca_sea, title="Enforcement Over Time for Two
↪ Categories").mark_line().encode(
    x = alt.X('yearmonth(Date):T', title = 'Month and Year'),
    y = alt.Y('count():Q', title = 'Number of Actions'),
    color=alt.Color('Category:N',
        legend=alt.Legend(title='Enforcement Type'))
)

cca_sea_plot
```



- based on five topics

I have access to Claude code, so first I'm going to filter the data to only 'Criminal and Civil Actions' and write it to a csv so Claude can read it.

```

crim_only = data[data['Category'] == 'Criminal and Civil Actions']

crim_only.to_csv('cca_only_data.csv')

```

I then gave Claude the following instructions:

“I want your help classifying all of the actions in cca_data_only.csv based on their titles. My instructions for this are to classify these into the following five categories: “Health Care Fraud”, “Financial Fraud”, “Drug Enforcement”, “Bribery/Corruption”, and “Other”. Hint: You will need to divide the five topics manually by looking at the title and assigning the relevant topic. For example, if you find the word “bank” or “financial” in the title of an action, then that action should probably belong to “Financial Fraud” topic.”

It read the file and wrote a script which is largely just the patterns it identified. I have reviewed and adapted it and applied it to my specific task, which is applying it to “crim_only”:

```

def classify_enforcement_action(title):
    """
    Classify enforcement action based on title keywords.
    Priority order: Drug Enforcement > Bribery/Corruption > Financial Fraud >
    ↪ Health Care Fraud > Other
    """
    title_lower = title.lower()

    # Drug Enforcement - Check first (most specific)
    drug_keywords = [
        'opioid', 'oxycodone', 'controlled substance', 'fentanyl',
        ↪ 'hydrocodone',
        'illegal distribution', 'drug diversion', 'prescription fraud',
        ↪ 'narcotics',
        'distribute and dispense', 'illegally distribut', 'adulterated',
        ↪ 'misbranded',
        'drug', 'pharmaceutical', 'pharmacy fraud', 'pill mill', 'morphine',
        'codeine', 'methadone', 'suboxone', 'buprenorphine', 'amphetamine',
        'distributing controlled', 'dispensing controlled', 'unlawful
        ↪ distribution'
    ]

    # Bribery/Corruption - Check second
    bribery_keywords = [
        'kickback', 'bribe', 'bribery', 'corruption', 'illegal remuneration',
        'paying kickback', 'receiving kickback', 'anti-kickback', 'kickback
        ↪ scheme',
        'kickback conspiracy', 'corrupt', 'pay to play', 'quid pro quo',
        'inducement', 'referral fee', 'illegal payment'
    ]

    # Financial Fraud - Check third
    financial_keywords = [
        'money laundering', 'bank fraud', 'wire fraud', 'mail fraud',
        'securities fraud', 'investment fraud', 'ponzi', 'embezzle',
        'tax fraud', 'tax evasion', 'financial fraud', 'credit card fraud',
        'identity theft', 'aggravated identity', 'stolen identity',
        'financial institution', 'banker', 'banking', 'securities',
        'insider trading', 'accounting fraud', 'financial statement'
    ]

    # Health Care Fraud - Most common, check fourth

```

```

healthcare_keywords = [
    'medicare', 'medicaid', 'false claim', 'false billing', 'health care
    ↪ fraud',
    'healthcare fraud', 'medical fraud', 'billing fraud', 'upcoding',
    'unbundling', 'phantom billing', 'unnecessary service', 'unnecessary
    ↪ procedure',
    'hospice fraud', 'home health', 'durable medical equipment', 'dme
    ↪ fraud',
    'patient dumping', 'stark law', 'self-referral', 'false statement',
    'fraudulent claim', 'fraudulent billing', 'genetic test',
    ↪ 'telemedicine fraud',
    'covid-19 relief', 'covid relief', 'cares act', 'ppp fraud', 'eidl
    ↪ fraud',
    'laboratory fraud', 'diagnostic test', 'compound cream', 'pain
    ↪ cream',
    'ambulance fraud', 'dialysis fraud', 'nursing home', 'skilled
    ↪ nursing',
    'physical therapy fraud', 'occupational therapy', 'speech therapy
    ↪ fraud',
    'fraudulently bill', 'submit false', 'submitting false'
]

# Check in priority order

# 1. Drug Enforcement
for keyword in drug_keywords:
    if keyword in title_lower:
        # Special case: if it's about medical devices or equipment, might
        ↪ be healthcare
        if 'medical device' in title_lower and 'adulterat' in
        ↪ title_lower:
            return 'Drug Enforcement' # Adulterated medical devices
        elif 'distribut' in title_lower and any(drug in title_lower for
        ↪ drug in ['opioid', 'oxycodone', 'controlled', 'fentanyl']):
            return 'Drug Enforcement'
        elif any(drug in title_lower for drug in ['opioid', 'oxycodone',
        ↪ 'controlled substance', 'fentanyl', 'hydrocodone']):
            return 'Drug Enforcement'

# 2. Bribery/Corruption
for keyword in bribery_keywords:
    if keyword in title_lower:

```

```

        return 'Bribery/Corruption'

# 3. Financial Fraud
for keyword in financial_keywords:
    if keyword in title_lower:
        # Check if it's primarily healthcare-related despite having
        ↪ financial elements
        if any(hc in title_lower for hc in ['medicare', 'medicaid',
        ↪ 'health care', 'healthcare', 'medical']):
            return 'Health Care Fraud'
        return 'Financial Fraud'

# 4. Health Care Fraud
for keyword in healthcare_keywords:
    if keyword in title_lower:
        return 'Health Care Fraud'

# Additional checks for Health Care Fraud (common patterns)
if any(pattern in title_lower for pattern in [
    'physician', 'doctor', 'nurse', 'hospital', 'clinic', 'medical',
    'health', 'patient', 'treatment', 'therapy', 'diagnostic',
    'laboratory', 'pharmacy', 'prescription', 'ambulance'
]):
    # Check if it's not about drugs or kickbacks
    if not any(drug in title_lower for drug in ['opioid', 'controlled
    ↪ substance', 'distribut']):
        if 'kickback' not in title_lower and 'bribe' not in title_lower:
            return 'Health Care Fraud'

# 5. Other - Everything else
# This includes: assault, abuse, obstruction of justice, general theft,
↪ etc.
return 'Other'

```

```

crim_only['Topic'] = crim_only['Title'].apply(classify_enforcement_action)

crim_only[['Title', 'Topic']].head()

```

	Title	Topic
0	Houston Transplant Doctor Indicted For Making ...	Health Care Fraud

	Title	Topic
1	MultiCare Health System to Pay Millions to Set...	Health Care Fraud
3	Delafield Man Sentenced to 18 Months' Imprison...	Bribery/Corruption
4	Former NFL Player Convicted for \$197M Medicare...	Health Care Fraud
7	Florida Man Pleads Guilty to Conspiracy to Vio...	Bribery/Corruption

Now, plot as instructed:

```
cca_plot = alt.Chart(crim_only).mark_line().encode(
    x=alt.X('yearmonth(Date):T', title='Month and Year'),
    y=alt.Y('count():Q', title='Number of Occurrences'),
    color=alt.Color('Topic', legend=alt.Legend(title='Type of Action'))
).properties(
    title='Frequency of 5 Criminal and Civil Enforcement Actions over Time'
)

cca_plot
```

