30538 Pset 4

Brook Jaffe

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: BGJ

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

```
import requests
with open(r'/Users/brookjaffe/Desktop/MPP/DAPII/ps4-bjaffe1/Enforcement
 ↪  Actions _ Office of Inspector General _ Government Oversight _ U.S.
 ↪  Department of Health and Human Services.html', 'r') as page:
  text = page.read()
from bs4 import BeautifulSoup
soup = BeautifulSoup(text, 'lxml')
date_html = soup.find_all('span', class_ = 'text-base-dark
 ↪  padding-right-105')
category_html = soup.find_all('li', class_ = 'display-inline-block usa-tag
 ↪  text-no-lowercase text-base-darkest bg-base-lightest margin-right-1')
title_html = soup.find_all('h2', class_ = 'usa-card__heading')

date_column = []
for date in date_html:
  date_only = date.text
  date_column.append(date_only)

cat_column = []
for category in category_html:
  cat = category.text
  cat_column.append(cat)

title_column = []
for title in title_html:
  title_text = title.text[1:-1]
  title_column.append(title_text)
```

```
updated_cats = []
for i in range(len(cat_column)):
  if cat_column[i] == 'COVID-19':
    new_cat = 'COVID-19, Criminal and Civil Actions'
  elif cat_column[i-1] == 'COVID-19':
    new_cat = None
  else:
    new_cat = cat_column[i]
  updated_cats.append(new_cat)
updated_cats = filter(None, updated_cats)
updated_cats = list(updated_cats)

link_column = []
for action in title_html:
  link = action.find('a').get('href')
  link_column.append(link)

scraped_df = pd.DataFrame({'title': title_column,
                           'date': date_column,
                           'enforcement_category': updated_cats,
                           'link': link_column})

print(scraped_df.head())
```

```
                                               title              date  \
0  Delafield Man Sentenced to 18 Months' Imprison...  February 3, 2026
1  AG's Office Secures Indictments Against Peabod...  February 2, 2026
2  Florida Man Pleads Guilty to Conspiracy to Vio...  January 30, 2026
3  Yadkinville Woman Sentenced in Connection with...  January 29, 2026
4  Slidell Chiropractor Sentenced for Health Care...  January 28, 2026

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

                                                link
0  https://oig.hhs.gov/fraud/enforcement/delafiel...
1  https://oig.hhs.gov/fraud/enforcement/ags-offi...
2  https://oig.hhs.gov/fraud/enforcement/florida-...
3  https://oig.hhs.gov/fraud/enforcement/yadkinvi...
```

```
4  https://oig.hhs.gov/fraud/enforcement/slidell-...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code Step 1: Check if the year is >= 2013

  Step 1a: If yes, proceed to step 2

  Step 1b: If no, remind the user to input a date that is in 2013 or later

Step 2: Inspect page html

Step 3: Check if any enforcement actions on the page have a date preceding the month and year requested

Step 3a: If yes, collect all enforcement actions above that one

Step 3b: If no, collect all enforcement actions on the page and precede to step 4

Step 4: Parse html and extract the url to go to the next page

Step 5: Follow that link

Step 6: Repeat steps 3-5 until the month+year prior to the one requested is reached

Step 7: Compile the collected information into a dataframe

- b. Create Dynamic Scraper

```
get_enforcement('January 2024')
```

```
jan_2024 =
 ↪  pd.read_csv('/Users/brookjaffe/Desktop/MPP/DAPII/ps4-bjaffe1/enforcement_actions_2024_1.
print(jan_2024.shape)
print(jan_2024.tail())
```

```
(1811, 5)
      Unnamed: 0                                              title  \
1806        1806  Athletico Management, PT Network, and Dynamic ...
1807        1807  Recover-Care Plaza West Care Center Agreed to ...
1808        1808  Maury County Caregiver Charged With Financial ...
1809        1809  Laredo Resident Admits To Impersonating Licens...
1810        1810  Former Nurse Aide Indicted In Death Of Clarksv...
```

```
                date        enforcement_category  \
1806  January 4, 2024      Fraud Self-Disclosures
1807  January 4, 2024      Fraud Self-Disclosures
1808  January 4, 2024  State Enforcement Agencies
1809  January 3, 2024  Criminal and Civil Actions
1810  January 3, 2024  State Enforcement Agencies

                                                link
1806  /fraud/enforcement/athletico-management-pt-net...
1807  /fraud/enforcement/recover-care-plaza-west-car...
1808  /fraud/enforcement/maury-county-caregiver-char...
1809  /fraud/enforcement/laredo-resident-admits-to-i...
1810  /fraud/enforcement/former-nurse-aide-indicted-...
```

There are 1,811 enforcement actions in the final dataframe, with the earliest being the indictment of a former nurse aide from January 3, 2024 in the State Enforcement Agencies category.

- c. Test Your Code

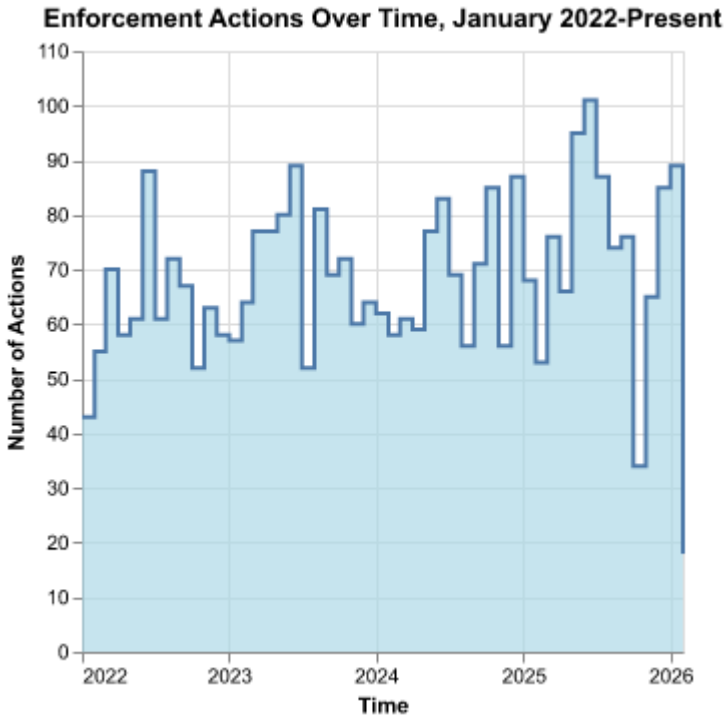
```
get_enforcement('January 2022')
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df_enforcement =
 ↪  pd.read_csv('/Users/brookjaffe/Desktop/MPP/DAPII/ps4-bjaffe1/enforcement_actions_2022_1.c
df_enforcement['date'] = pd.to_datetime(df_enforcement['date'])

df_enforcement['yearmonth'] =
 ↪  df_enforcement['date'].dt.to_period('M').astype(str)

df_enforcement['yearmonth'] = pd.to_datetime(df_enforcement['yearmonth'])

df_enforcement_over_time =
 ↪  df_enforcement.groupby('yearmonth').count().reset_index()

enforcement_over_time = alt.Chart(df_enforcement_over_time, title =
 ↪  'Enforcement Actions Over Time, January 2022-Present').mark_area(
  color='lightblue',
  interpolate='step-after',
```

```
  line=True
).encode(
  x = alt.X('yearmonth:T').title('Time'),
  y = alt.Y('title').title('Number of Actions')
)

enforcement_over_time.show()
```

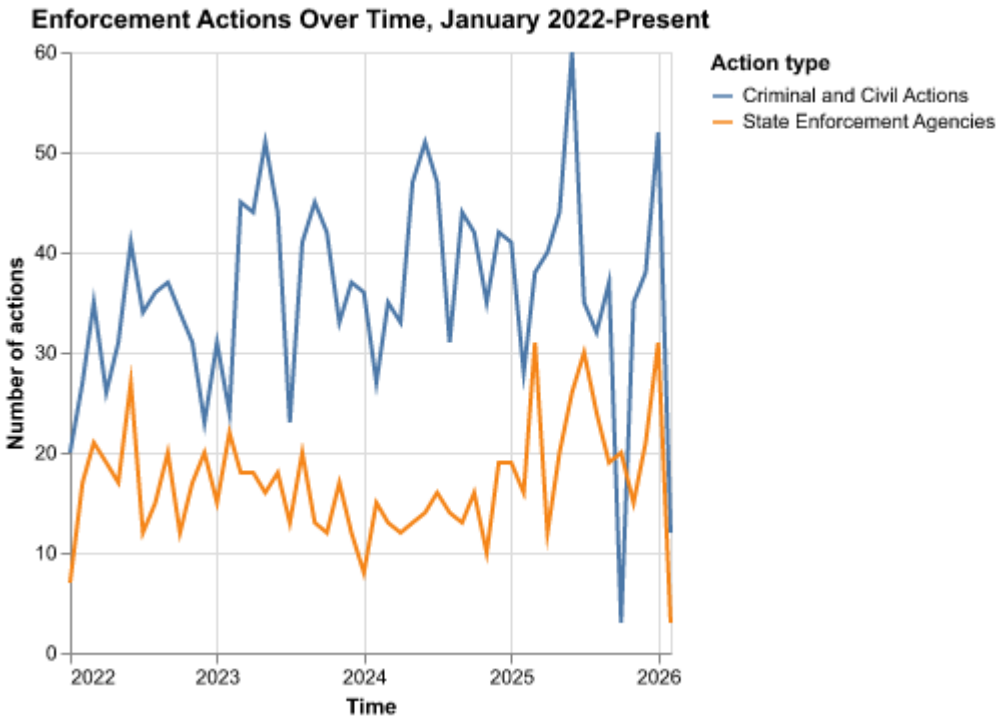


2. Plot the number of enforcement actions categorized:

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

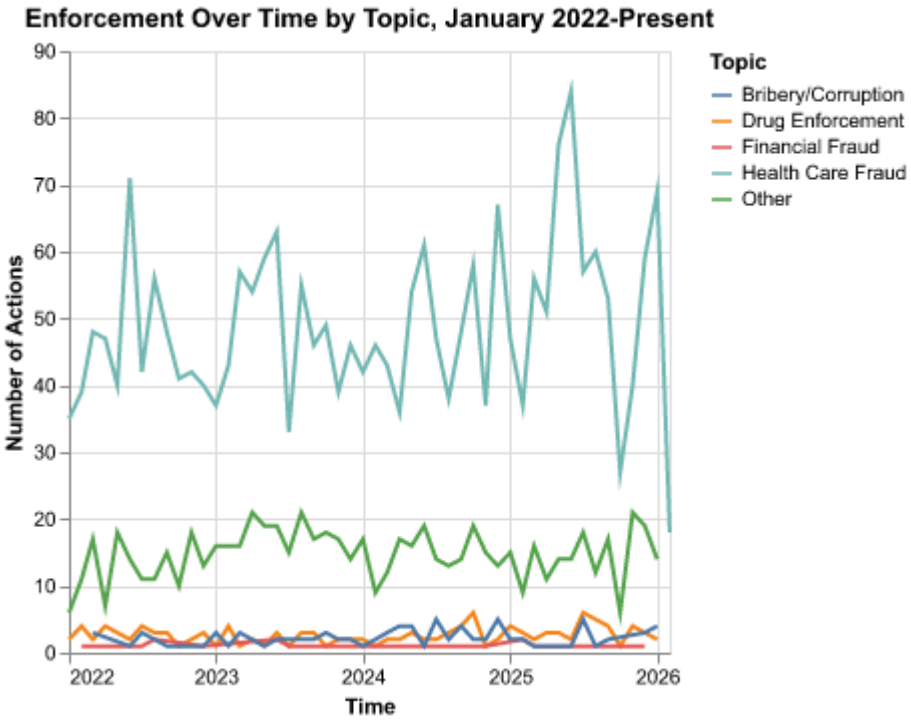
```
df_enforcement['classification'] = None
for i in df_enforcement.index:
  if df_enforcement.loc[i, 'enforcement_category'] == 'Criminal and Civil
   ↪ Actions':
    df_enforcement.loc[i, 'classification'] = 'Criminal and Civil Actions'
  elif df_enforcement.loc[i, 'enforcement_category'] == 'State Enforcement
   ↪ Agencies':
```

```
    df_enforcement.loc[i, 'classification'] = 'State Enforcement Agencies'

df_enforcement_two_categories = df_enforcement.groupby(by = ['yearmonth',
 ↪  'classification']).count().reset_index()

crim_vs_state = alt.Chart(df_enforcement_two_categories).mark_line().encode(
  x = alt.X('yearmonth:T').title('Time'),
  y = alt.Y('title:Q').title('Number of actions'),
  color = alt.Color('classification:N').title('Action type')
).properties(
  title={
    'text': 'Enforcement Actions Over Time, January 2022-Present'
  }
)

crim_vs_state.show()
```



- based on five topics

```
df_enforcement['topic'] = 'Other'

for i in df_enforcement.index:
  if (('Health' in df_enforcement.loc[i, 'title']) |
  ('Medicaid' in df_enforcement.loc[i, 'title']) |
  ('Medicare' in df_enforcement.loc[i, 'title']) |
  ('Doctor' in df_enforcement.loc[i, 'title']) |
  ('Medical' in df_enforcement.loc[i, 'title']) |
  ('Unlicensed' in df_enforcement.loc[i, 'title']) |
  ('Psychotherapy' in df_enforcement.loc[i, 'title']) |
  ('Civil Monetary Penalties Law' in df_enforcement.loc[i, 'title']) |
  ('Hosptial' in df_enforcement.loc[i, 'title']) |
  ('Caregiver' in df_enforcement.loc[i, 'title']) |
  ('False Claims Act' in df_enforcement.loc[i, 'title'])):
    df_enforcement.loc[i, 'topic'] = 'Health Care Fraud'
  elif (('Bank' in df_enforcement.loc[i, 'title']) |
  ('Financial' in df_enforcement.loc[i, 'title'])):
    df_enforcement.loc[i, 'topic'] = 'Financial Fraud'
  elif (('Drug' in df_enforcement.loc[i, 'title']) |
  ('Substance' in df_enforcement.loc[i, 'title']) |
  ('Pharmaceutical' in df_enforcement.loc[i, 'title']) |
  ('Opioid' in df_enforcement.loc[i, 'title'])):
    df_enforcement.loc[i, 'topic'] = 'Drug Enforcement'
  elif (('Kickback' in df_enforcement.loc[i, 'title']) |
  ('Bribe' in df_enforcement.loc[i, 'title']) |
  ('Embezzle' in df_enforcement.loc[i, 'title'])):
    df_enforcement.loc[i, 'topic'] = 'Bribery/Corruption'

df_enforcement_topics = df_enforcement.groupby(by = ['yearmonth',
 ↪  'topic']).count().reset_index()

enforcement_by_topic = alt.Chart(df_enforcement_topics).mark_line().encode(
  x = alt.X('yearmonth:T').title('Time'),
  y = alt.Y('title:Q').title('Number of Actions'),
  color = alt.Color('topic:N').title('Topic')
).properties(
  title = {
    'text': 'Enforcement Over Time by Topic, January 2022-Present'
  }
)

enforcement_by_topic.show()
```


Enforcement Over Time by Topic, January 2022-Present
Number of Actions
90
80
70
60
50
40
30
20
10
0
2022
2023
2024
2025
2026
Time
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other