PS4

Kunsh

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: KK

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 requests
from bs4 import BeautifulSoup

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

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

Step 1: Develop initial scraper and crawler

```
def scraper():
  results = []
  url = "https://oig.hhs.gov/fraud/enforcement/"

  respone = requests.get(url)
  soup = BeautifulSoup(respone.text,'lxml')

  entries = soup.find_all('li',class_="usa-card card--list pep-card--minimal
 ↪  mobile:grid-col-12")

  for entry in entries:
    enforcement_action = entry.find("a").get_text()
    link = entry.find("a")
    link = "https://oig.hhs.gov/" + link["href"]
    date = entry.find("span").get_text()
    category = entry.find("li").get_text()
    results.append({
              'Enforcement_action': enforcement_action,
              'Date': date,
              'Category': category,
              'Link': link
          })

  df = pd.DataFrame(results)
  return(df)

df_enforcement = scraper()
```

```
print(df_enforcement.head())
```

```
                                Enforcement_action              Date  \
0  Houston Transplant Doctor Indicted For Making ...  February 5, 2026
1  MultiCare Health System to Pay Millions to Set...  February 4, 2026
2  Brooklyn Banker Pleads Guilty to Laundering Pr...  February 3, 2026
3  Delafield Man Sentenced to 18 Months' Imprison...  February 3, 2026
4  Former NFL Player Convicted for $197M Medicare...  February 3, 2026

                    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/multica...
2  https://oig.hhs.gov//fraud/enforcement/brookly...
3  https://oig.hhs.gov//fraud/enforcement/delafie...
4  https://oig.hhs.gov//fraud/enforcement/former-...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code

I will use a While loop that runs forever so that I can break the loop inside the function based on the date of the data

FUNCTION: webcrawler(month,year) Create string turning year and month into one date Turn string into pandas datetime variable

check if year $<$ 2013 if so print data only available till 2013

RESULT: Make empty lift for storing results URL: 1st page of data link

```
While loop that runs forever
    download html of URL
    parse html
```

```
    find all entries with li tag and class associated with entries

    loop over entries with li tag
    extract the enfocrement action
    find the tag of the link
    extract the link
    extract the date and convert to a datetime object
    extract the category
    append new row of data to the dataframe
    convert new entry of date column to a datetime object
    check if new entry is before specified month and year
        if yes convert Result list to a dataframe
        if yes save dataframe as csv
        if yes exit function
    else
        append dictionary of extracted values to RESULT list

    1 second break

    find tag which contains next page link
    access nested tag with link
    assign link to URL
```

- b. Create Dynamic Scraper

```
def crawler(year,month):
  min_date = month+" 1,"+year
  min_date = pd.to_datetime(min_date)
  if min_date < pd.to_datetime("January 1,2013"):
    print("Data only available till 2013")

  results = []
  url = "https://oig.hhs.gov/fraud/enforcement/"

  while True:
    respone = requests.get(url)
    soup = BeautifulSoup(respone.text,'lxml')

    entries = soup.find_all('li',class_="usa-card card--list
↪   pep-card--minimal mobile:grid-col-12")

    for entry in entries:
      enforcement_action = entry.find("a").get_text()
```

```
      link = entry.find("a")
      link = "https://oig.hhs.gov/" + link["href"]
      date = pd.to_datetime(entry.find("span").get_text())
      category = entry.find("li").get_text()
      if date < min_date:
        df = pd.DataFrame(results)
        df.to_csv("enforcement_data_"+year+"_"+month+".csv")
        return(df)
      else:
        results.append({
                'Enforcement_action': enforcement_action,
                'Date': date,
                'Category': category,
                'Link': link
            })

    time.sleep(1)

    next_page = soup.find("li",class_ = "next-page")
    next_page = next_page.find("a")
    url = "https://oig.hhs.gov/fraud/enforcement/" + next_page["href"]

run_scraper = False

if run_scraper:
  crawler("2024", "January")
  df_enforcement = pd.read_csv("enforcement_data_2024_January.csv")
else:
  df_enforcement = pd.read_csv("enforcement_data_2024_January.csv")

print(df_enforcement.shape)
df_enforcement.sort_values(by="Date",ascending=True,inplace=True)
print(f"The oldest violation was {df_enforcement.iloc[0,1]}")
print(f"The date was {df_enforcement.iloc[0,2]}")
print(f"The category was {df_enforcement.iloc[0,3]}")
print(f"The link is {df_enforcement.iloc[0,4]}")
```

```
(1787, 5)
The oldest violation was Former Nurse Aide Indicted In Death Of Clarksville
Patient Arrested In Georgia
The date was 2024-01-03
The category was State Enforcement Agencies
```

```
The link is
https://oig.hhs.gov//fraud/enforcement/former-nurse-aide-indicted-in-death-of-clarksville-pat
```

There are 1787 actions in the data and the details of the oldest violation is printed above.

- c. Test Your Code

```
run_scraper = False

if run_scraper:
  crawler("2022", "January")
  df_enforcement = pd.read_csv("enforcement_data_2022_January.csv")
else:
  df_enforcement = pd.read_csv("enforcement_data_2022_January.csv")

print(df_enforcement.shape)
df_enforcement.sort_values(by="Date",ascending=True,inplace=True)
print(f"The oldest violation was {df_enforcement.iloc[0,1]}")
print(f"The date was {df_enforcement.iloc[0,2]}")
print(f"The category was {df_enforcement.iloc[0,3]}")
print(f"The link is {df_enforcement.iloc[0,4]}")
```

```
(3377, 5)
The oldest violation was Integrated Pain Management Medical Group Agreed to
Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by
Employing Excluded Individuals
The date was 2022-01-04
The category was Fraud Self-Disclosures
The link is
https://oig.hhs.gov//fraud/enforcement/integrated-pain-management-medical-group-agreed-to-pay
```

There are 3377 actions in the data and the details of the oldest violation is printed above.

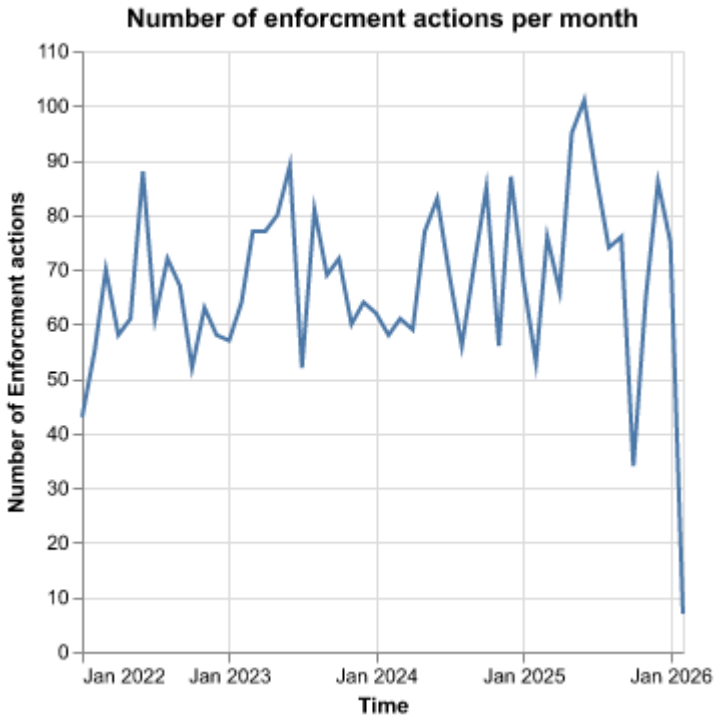
Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df_enforcement["Date"] = pd.to_datetime(df_enforcement["Date"])
df_enforcement["year"]= df_enforcement["Date"].dt.year
df_enforcement["month"]= df_enforcement["Date"].dt.month
df_enforcement_agg =
 ↪  df_enforcement.groupby(["year","month"],as_index=False).size()
```

```
df_enforcement_agg["date"] = pd.to_datetime(df_enforcement_agg[["year",
 ↪  "month"]].assign(DAY=1))

alt.Chart(df_enforcement_agg).mark_line().encode(
  x = alt.X("date:T",title="Time",timeUnit="utcyearmonth"),
  y = alt.Y("size:Q", title="Number of Enforcment actions")
).properties(
  title = "Number of enforcment actions per month"
)
```



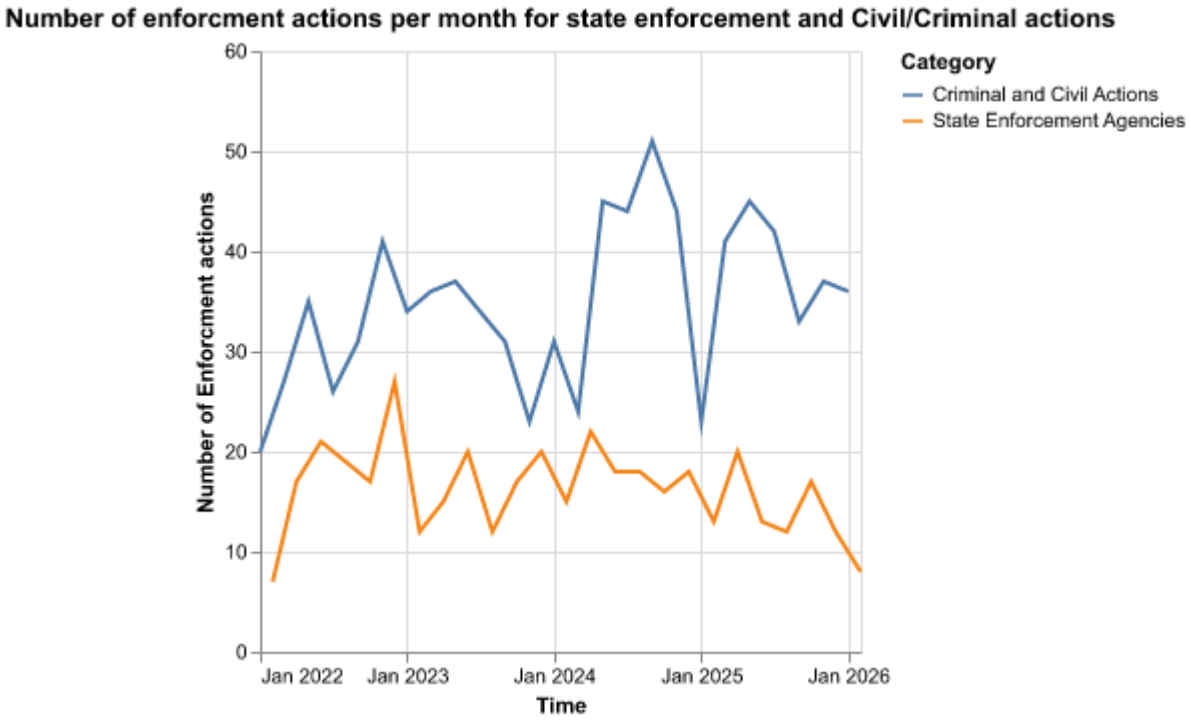
2. Plot the number of enforcement actions categorized:

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

```
df_enforcement_CCA_SEA = df_enforcement[(df_enforcement["Category"] ==
 ↪  "Criminal and Civil Actions") | (df_enforcement["Category"] ==  "State
 ↪  Enforcement Agencies")]
```

```
df_enforcement_agg_2 =
 ↪  df_enforcement_CCA_SEA.groupby(["year","month","Category"],as_index=False).size()
df_enforcement_agg_2["date"] = pd.to_datetime(df_enforcement_agg[["year",
 ↪  "month"]].assign(DAY=1))

alt.Chart(df_enforcement_agg_2).mark_line().encode(
  x = alt.X("date:T",title="Time",timeUnit="utcyearmonth"),
  y = alt.Y("size:Q", title="Number of Enforcment actions"),
  color= alt.Color("Category:N")
).properties(
  title = "Number of enforcment actions per month for state enforcement and
 ↪  Civil/Criminal actions"
)
```



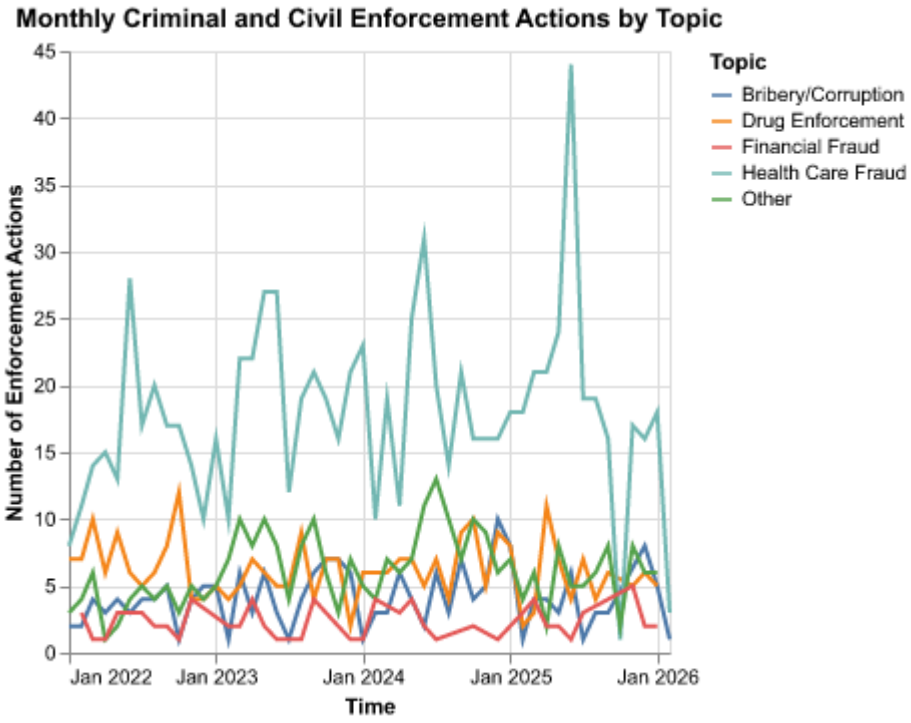
- based on five topics

```
def categorize(text):
  text = text.lower()
  if any(word in text for word in ['drug', 'opioid', 'narcotic', 'controlled
   ↪  substance', 'pill', 'pharmacy', 'pharmaceutical']):
```

```
    return "Drug Enforcement"
  elif any(word in text for word in ['bank', 'financial', 'laundering',
   ↪  'securities', 'money', "wire","funds"]):
    return "Financial Fraud"
  elif any(word in text for word in ['kickback', 'bribe', 'bribery',
   ↪  'corruption', 'anti-kickback']):
    return "Bribery/Corruption"
  elif any(word in text for word in ['medicare', 'medicaid', 'health',
   ↪  'hospital', 'medical', 'doctor', 'patient', 'nurse', 'care']):
    return "Health Care Fraud"
  else:
    return "Other"

df_cca = df_enforcement[df_enforcement["Category"] == "Criminal and Civil
 ↪  Actions"].copy()
df_cca["subcategory"] = df_cca["Enforcement_action"].apply(categorize)
df_agg = df_cca.groupby(["year", "month", "subcategory"],
 ↪  as_index=False).size()
df_agg["date"] = pd.to_datetime(df_agg[["year", "month"]].assign(day=1))

alt.Chart(df_agg).mark_line().encode(
    x = alt.X("date:T", title="Time", timeUnit="utcyearmonth"),
    y = alt.Y("size:Q", title="Number of Enforcement Actions"),
    color = alt.Color("subcategory:N", title = "Topic")
).properties(
    title = "Monthly Criminal and Civil Enforcement Actions by Topic"
)
```


Monthly Criminal and Civil Enforcement Actions by Topic
Number of Enforcement Actions
45
40
35
30
25
20
15
10
5
0
Jan 2022
Jan 2023
Jan 2024
Jan 2025
Jan 2026
Time
Topic
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other