title

author

Invalid Date

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: SH

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
from bs4 import BeautifulSoup
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

data = []
cards = soup.find_all('li', class_='usa-card')

for card in cards:
    heading_tag = card.find('h2', class_='usa-card__heading')
    if heading_tag and heading_tag.find('a'):
        link_tag = heading_tag.find('a')
        title = link_tag.get_text(strip=True)
        link = "https://oig.hhs.gov" + link_tag['href']

        info_div = card.find('div', class_='font-body-sm')
        if info_div:
            spans = info_div.find_all('span')
            date = spans[0].get_text(strip=True) if len(spans) > 0 else "N/A"
            category = spans[1].get_text(strip=True) if len(spans) > 1 else
↪   "N/A"
        else:
            date, category = "N/A", "N/A"

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

df = pd.DataFrame(data)

print(df.head())
```

```
                                               Title              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                                              Link
0      N/A  https://oig.hhs.gov/fraud/enforcement/houston-...
1      N/A  https://oig.hhs.gov/fraud/enforcement/multicar...
2      N/A  https://oig.hhs.gov/fraud/enforcement/brooklyn...
3      N/A  https://oig.hhs.gov/fraud/enforcement/delafiel...
4      N/A  https://oig.hhs.gov/fraud/enforcement/former-n...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code FUNCTION scrape_enforcement_actions(start_month, start_year, run_scraper):

```
1. INITIAL VALIDATION: IF start_year < 2013: PRINT "Reminder: Please restrict year to >= 2013." RETURN

2. EXECUTION CHECK (The Indicator): IF run_scraper is FALSE: PRINT "Scraper disabled to optimize compilation." RETURN

3. INITIALIZATION:
    – Set base_url to "https://oig.hhs.gov/fraud/enforcement/"
    – Create an empty list 'results_data'
    – Create 'target_date' object using (start_year, start_month)
    – Set 'current_page' to 1
    – Set 'continue_crawling' flag to TRUE
```

4. MAIN CRAWLING (The While Loop): WHILE continue_crawling is TRUE: - Construct the URL (handle page 1 vs. subsequent pages) - Request the webpage content and parse with BeautifulSoup - Identify all 'Enforcement Action' cards on the page

   ```
   FOR each card in the page:
       - Extract: Title, Link, Date, and Category
       - Convert the extracted Date string into a comparable object

       - CHECK DATE CONDITION:
         IF extracted_date < target_date:
             Set continue_crawling to FALSE
             BREAK (Stop processing cards)
         ELSE:
             Append card details to 'results_data'

   - IF continue_crawling is still TRUE:
       - Increment 'current_page'
       - WAIT for 1 second (time.sleep) to respect server limits
   - ELSE:
       - Exit the loop
   ```

5. DATA EXPORT:

   - Convert 'results_data' into a Tidy DataFrame
   - Save DataFrame to "enforcement_actions_year_month.csv"
   - RETURN the DataFrame

- b. Create Dynamic Scraper

```python
from datetime import datetime

def scrape_enforcement_actions(start_month, start_year, run_scraper=False):
    if start_year < 2013:
        print("Please restrict to year >= 2013, since only enforcement
         ↪  actions after 2013 are listed.")
        return None

    if not run_scraper:
        print("Scraper is currently disabled (run_scraper=False).")
        return None

    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    all_data = []
```

```
    current_page = 1
    continue_crawling = True
    threshold_date = datetime(start_year, start_month, 1)

    while continue_crawling:
        url = base_url if current_page == 1 else
↪   f"{base_url}?page={current_page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'html.parser')

        cards = soup.find_all('li', class_='usa-card')
        if not cards:
            break

        for card in cards:
            title_tag = card.find('h2', class_='usa-card__heading').find('a')
            title = title_tag.get_text(strip=True)
            link = "https://oig.hhs.gov" + title_tag['href']

            info_div = card.find('div', class_='font-body-sm')

            date_tag = info_div.find('span') if info_div else None
            date_str = date_tag.get_text(strip=True) if date_tag else None

            category_tag = card.find('li', class_='usa-tag')
            category = category_tag.get_text(strip=True) if category_tag else
↪   "N/A"

            if date_str:
                current_item_date = datetime.strptime(date_str, "%B %d, %Y")

                if current_item_date < threshold_date:
                    continue_crawling = False
                    break

                all_data.append({
                    "Title": title,
                    "Date": date_str,
                    "Category": category,
                    "Link": link
                })

        if continue_crawling:
            current_page += 1
            time.sleep(1)

    return pd.DataFrame(all_data)

df_2024 = scrape_enforcement_actions(start_month=1, start_year=2024,
↪   run_scraper=True)

if df_2024 is not None and not df_2024.empty:
    print(f"Number of enforcement actions in final dataframe:
     ↪   {len(df_2024)}")

    earliest_action = df_2024.iloc[-1]
    print(f"Date of earliest enforcement action: {earliest_action['Date']}")
    print(f"Details of earliest enforcement action:
     ↪   {earliest_action['Title']}")
```

```
Number of enforcement actions in final dataframe: 1787
Date of earliest enforcement action: January 3, 2024
Details of earliest enforcement action: Former Nurse Aide Indicted In Death
Of Clarksville Patient Arrested In Georgia
```

- c. Test Your Code

```
enforcement_actions_year_month = scrape_enforcement_actions(start_month=1,
↪   start_year=2022, run_scraper=True)

if enforcement_actions_year_month is not None:
    print(f"Total enforcement actions since Jan 2022:
     ↪   {len(enforcement_actions_year_month)}")
    earliest_entry = enforcement_actions_year_month.iloc[-1]

    print("\n--- Earliest Enforcement Action Details ---")
    print(f"Date: {earliest_entry['Date']}")
    print(f"Title: {earliest_entry['Title']}")
    print(f"Category: {earliest_entry['Category']}")
    enforcement_actions_year_month.to_csv("enforcement_actions_2022_1.csv",
↪   index=False)
    print("\nFile saved successfully: enforcement_actions_2022_1.csv")
```

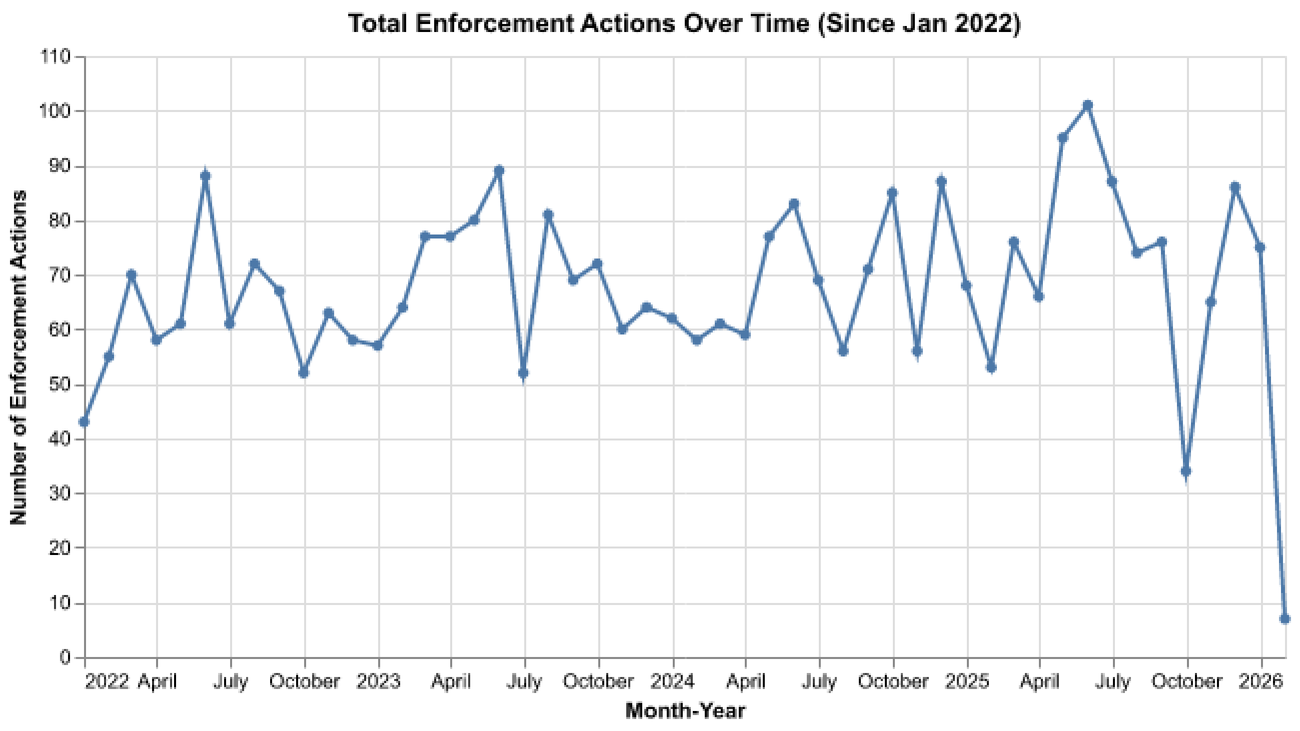
```
Total enforcement actions since Jan 2022: 3377
```

```
--- Earliest Enforcement Action Details ---
Date: January 4, 2022
Title: Integrated Pain Management Medical Group Agreed to Pay $10,000 for
Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded
Individuals
Category: Fraud Self-Disclosures

File saved successfully: enforcement_actions_2022_1.csv
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```python
file_name = "enforcement_actions_2022_1.csv"
df_from_csv = pd.read_csv(file_name)

df_from_csv['Date'] = pd.to_datetime(df_from_csv['Date'])
df_from_csv['YearMonth'] =
 ↪  df_from_csv['Date'].dt.to_period('M').dt.to_timestamp()

monthly_trends =
 ↪  df_from_csv.groupby('YearMonth').size().reset_index(name='action_count')

line_chart = alt.Chart(monthly_trends).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Month-Year'),
    y=alt.Y('action_count:Q', title='Number of Enforcement Actions'),
    tooltip=['YearMonth:T', 'action_count:Q']
).properties(
    title='Total Enforcement Actions Over Time (Since Jan 2022)',
    width=600,
    height=300
)

line_chart
```

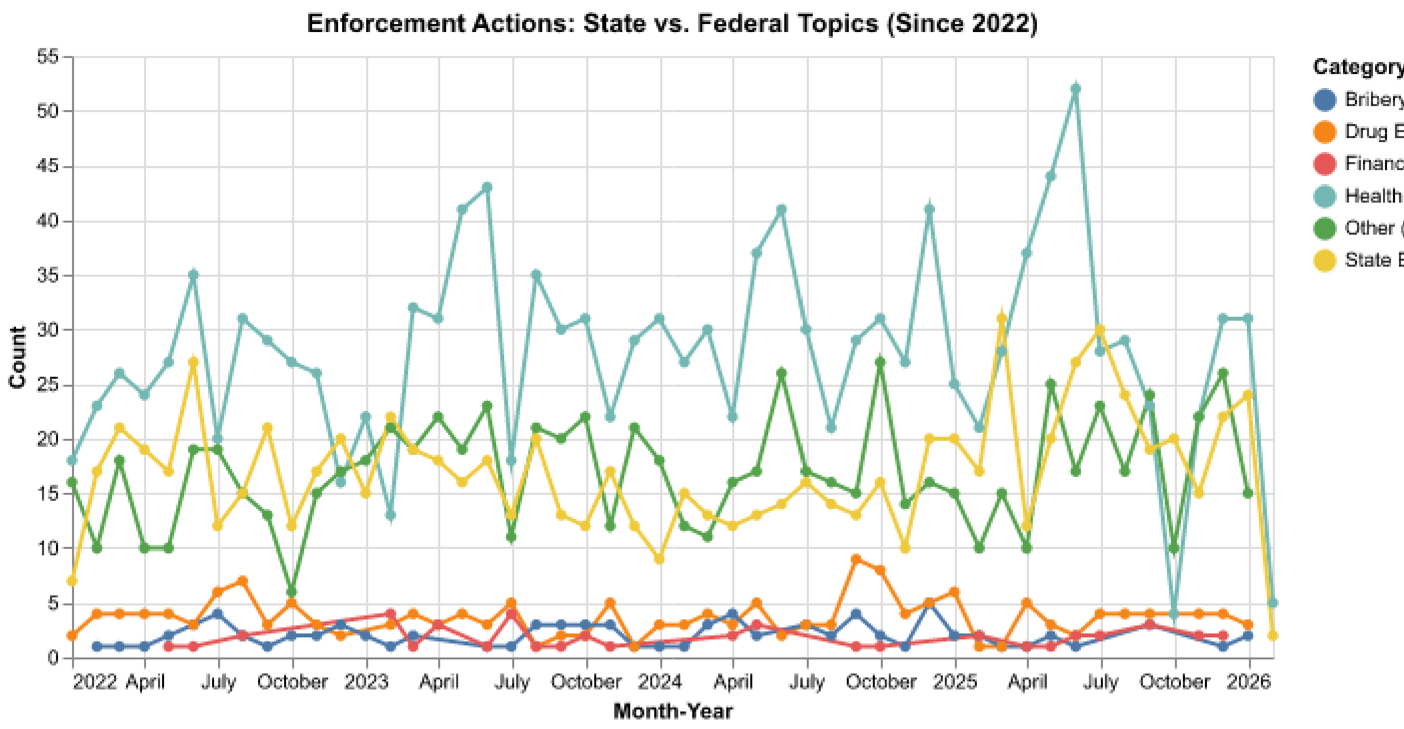


2. Plot the number of enforcement actions categorized:

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

```
def assign_final_segments_v3(row):
    cat = str(row['Category']).lower() if pd.notna(row['Category']) else ""
    title = str(row['Title']).lower()


    if 'state' in cat or 'attorney general' in cat or 'state of' in title:
        return "State Enforcement Agencies"

    if any(w in title for w in ['medicare', 'medicaid', 'health', 'patient',
     ↪  'hospital', 'doctor', 'nurse']):
        return "Health Care Fraud"
    elif any(w in title for w in ['bank', 'financial', 'wire', 'money',
     ↪  'laundering', 'tax', 'investment', 'loan']):
        return "Financial Fraud"
    elif any(w in title for w in ['drug', 'opioid', 'pharmacy',
     ↪  'prescription', 'controlled']):
        return "Drug Enforcement"
    elif any(w in title for w in ['bribe', 'kickback', 'corruption',
     ↪  'official']):
        return "Bribery/Corruption"
    else:
        return "Other (Crim/Civ)"

df_from_csv['Final_Group'] = df_from_csv.apply(assign_final_segments_v3,
 ↪  axis=1)

print(df_from_csv['Final_Group'].value_counts())


segmented_data = df_from_csv.groupby(['YearMonth',
 ↪  'Final_Group']).size().reset_index(name='n_actions')


line_chart_debug = alt.Chart(segmented_data).mark_line(point=True).encode(
    x=alt.X('YearMonth:T', title='Month-Year'),
    y=alt.Y('n_actions:Q', title='Count'),
    color=alt.Color('Final_Group:N', title='Category Split'),
    tooltip=['YearMonth:T', 'Final_Group:N', 'n_actions:Q']
).properties(
    title='Enforcement Actions: State vs. Federal Topics (Since 2022)',
    width=600,
    height=300
)

line_chart_debug.display()
```

```
Final_Group
Health Care Fraud            1396
State Enforcement Agencies    848
Other (Crim/Civ)              831
Drug Enforcement              173
Bribery/Corruption             85
Financial Fraud                44
Name: count, dtype: int64
```


Enforcement Actions: State vs. Federal Topics (Since 2022)
Count
Month-Year
0
5
10
15
20
25
30
35
40
45
50
55
2022
April
July
October
2023
April
July
October
2024
April
July
October
2025
April
July
October
2026
Category
Bribery
Drug E
Financ
Health
Other (
State E