ps4

Dominick Sanakiewicz

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:DS **___**

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 os
from bs4 import BeautifulSoup
import requests
import warnings
import lxml

warnings.filterwarnings('ignore')
```

Step 1: Develop initial scraper and crawler

```
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url, headers={"User-Agent": "Mozilla/5.0"})
soup = BeautifulSoup(response.text, "html.parser")

data = []

titles = soup.select("h2 a")

for a in titles:
    title = a.get_text(strip=True)
    link = "https://oig.hhs.gov" + a["href"]

    parent = a.parent.parent
    text = list(parent.stripped_strings)

    date = text[1] if len(text) > 1 else None
    category = text[2] if len(text) > 2 else None

    data.append({
        "title": title,
        "date": date,
        "category": category,
        "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
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 Check run indicator If run_scrape is False: Print a message Return an empty dataframe

  If year < 2013:

  Print a warning message Return an empty dataframe

Parameters Set start_date to the first day of the input month and year Set base_url to the HHS OIG Enforcement Actions page Create an empty list to store scraped rows Set page = 1

Begin crawling loop (while loop) Construct the URL: Use the base URL if page == 1 Otherwise append ?page={page} Parse the HTML Select all enforcement action title links on the page

Check stopping condition: If no titles are found: Exit the loop

Extract data from page For each title link: Extract the title text Construct the full link Locate the surrounding HTML block Extract the date and category text Append the extracted values to the data list

Check date threshold Convert collected dates to datetime If the earliest scraped date is earlier than start_date: Exit the loop (have scraped far enough back)

Pause and advance Wait one second to avoid server overload Increment the page counter

Post-processing Convert the collected list into a df Parse dates and drop invalid rows Filter to enforcement actions on or after start_date Sort by date

Save the df as enforcement_actions_year_month.csv

Return the final clean df

- b. Create Dynamic Scraper

```
def scrape_actions(month, year, run_scrape=True):
    if not run_scrape:
        print("not scraping, turn it on bro")
        return pd.DataFrame()

    if year < 2013:
        print("too old bro!!")
        return pd.DataFrame()

    start_date = pd.Timestamp(year, month, 1)

    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    data = []
    page = 1

    while True:
        url = base_url if page == 1 else base_url + f"?page={page}"
        response = requests.get(url, headers={"User-Agent": "Mozilla/5.0"})
        soup = BeautifulSoup(response.text, "html.parser")

        titles = soup.select("h2 a")
        if len(titles) == 0:
            break

        for a in titles:
            title = a.get_text(strip=True)
            link = "https://oig.hhs.gov" + a["href"]

            parent = a.parent.parent
            text = list(parent.stripped_strings)

            date = text[1] if len(text) > 1 else None
            category = text[2] if len(text) > 2 else None

            data.append({"title": title, "date": date, "category": category,
 ↪  "link": link})

        df_tmp = pd.DataFrame(data)
        df_tmp["date"] = pd.to_datetime(df_tmp["date"], errors="coerce")
        if df_tmp["date"].min() < start_date:
            break

        time.sleep(1)
```

```
        page += 1

    df = pd.DataFrame(data)
    df["date"] = pd.to_datetime(df["date"], errors="coerce")
    df = df.dropna(subset=["date"])
    df = df[df["date"] >= start_date].sort_values("date")

    fname = f"enforcement_actions_{year}_{month:02d}.csv"
    df.to_csv(fname, index=False)

    return df
```

```
run_scrape = True

df_2024 = scrape_actions(1, 2024, run_scrape=run_scrape)

len(df_2024)
```

1787

```
df_2024.head(1)
```

	title	date	category	link
1786	Former Nurse Aide Indicted In Death Of Clarksv...	2024-01-03	State Enforcement Agencies	https:/

```
run_scrape = False
```

```
run_scrape = True

df_2022 = scrape_actions(1, 2022, run_scrape=run_scrape)

len(df_2022)
```

3377

```
run_scrape = False
df_2022 = pd.read_csv("enforcement_actions_2022_01.csv",
 ↪  parse_dates=["date"])
```

```
len(df_2022)
df_2022.head(1)
```

	title	date	category	link
0	Integrated Pain Management Medical Group Agree...	2022-01-04	Fraud Self-Disclosures	https://oig.hh

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df = pd.read_csv(
    "enforcement_actions_2022_01.csv",
    parse_dates=["date"]
)

df["year_month"] = df["date"].dt.to_period("M").dt.to_timestamp()

monthly_counts = (
    df.groupby("year_month")
      .size()
      .reset_index(name="n_actions")
)

monthly_counts.head()
```

	year_month	n_actions
0	2022-01-01	43
1	2022-02-01	55
2	2022-03-01	70
3	2022-04-01	58
4	2022-05-01	61

```
alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X("year_month:T", title="Month"),
    y=alt.Y("n_actions:Q", title="Number of enforcement actions")
```

```
).properties(
    width=600,
    height=300,
    title="Enforcement Actions per Month since 2022"
)
```

```
alt.Chart(...)
```

2. Plot the number of enforcement actions categorized:

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

```
df["year_month"] = df["date"].dt.to_period("M").dt.to_timestamp()

df["category"] = df["category"].fillna("")
```

```
df["group"] = "Other"
df.loc[df["category"].str.contains("Criminal and Civil Actions", case=False,
 ↪  na=False), "group"] = "Criminal and Civil Actions"
df.loc[df["category"].str.contains("State Enforcement Agencies", case=False,
 ↪  na=False), "group"] = "State Enforcement Agencies"

monthly_two = (
    df[df["group"].isin(["Criminal and Civil Actions", "State Enforcement
 ↪  Agencies"])]
    .groupby(["year_month", "group"])
    .size()
    .reset_index(name="n_actions")
)

alt.Chart(monthly_two).mark_line().encode(
    x=alt.X("year_month:T", title="Month"),
    y=alt.Y("n_actions:Q", title="Number of enforcement actions"),
    color=alt.Color("group:N", title="Category")
).properties(
    width=600,
    height=300,
    title="Enforcement Actions per Month: Criminal & Civil vs State
 ↪  Enforcement"
)
```

```
alt.Chart(...)
```

```
cc = df[df["group"] == "Criminal and Civil Actions"].copy()

def topic_from_title(t):
    t = (t or "").lower()

    if any(k in t for k in ["medicare", "medicaid", "hospital", "clinic",
     ↪  "physician", "hospice", "health care", "home health", "nursing"]):
        return "Health Care Fraud"
    if any(k in t for k in ["bank", "financial", "loan", "wire fraud", "money
     ↪  laundering", "launder", "securities", "credit", "mortgage"]):
        return "Financial Fraud"
    if any(k in t for k in ["opioid", "fentanyl", "drug", "controlled
     ↪  substance", "pharmacy", "pill", "prescription", "narcotic"]):
        return "Drug Enforcement"
    if any(k in t for k in ["brib", "kickback", "corrupt", "extortion", "quid
     ↪  pro quo"]):
        return "Bribery/Corruption"
    return "Other"

cc["topic"] = cc["title"].apply(topic_from_title)

monthly_topics = (
    cc.groupby(["year_month", "topic"])
      .size()
      .reset_index(name="n_actions")
)

alt.Chart(monthly_topics).mark_line().encode(
    x=alt.X("year_month:T", title="Month"),
    y=alt.Y("n_actions:Q", title="Number of enforcement actions"),
    color=alt.Color("topic:N", title="Topic")
).properties(
    width=600,
    height=300,
    title="Criminal & Civil Actions per Month by 5 Categories"
)
```

```
alt.Chart(...)
```

outside sources:

https://www.youtube.com/watch?v=DcI_AZqfZVc - vidoe on web scrapping

Chat GPT queries: How do I scrape nested tags using Beautiful Soup in python? , Data has a multiple pages in html how to scrape using Beautfil Soup in python with line by line short example, embedding time delay in web scraping; finding the 5 categories in the data and helping understand the intuiton behind having multiple