30538 Problem Set 4: Web Scraping

Peter Ganong, Maggie Shi, Richard Chen

2026-02-01

Due 02/07/2026 at 5:00PM Central

"This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: Ziyun Wu

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.

(40%) Step 1: Develop initial scraper and crawler

Scraping: Go to the first page of the HHS OIG's "Enforcement Actions" page and scrape and collect the following into a dataset: * Title of the enforcement action * Date * Category (e.g, "Criminal and Civil Actions") * Link associated with the enforcement action

Collect your output into a tidy dataframe and print its `head`.

	title	date	category
0	Delafield Man Sentenced to 18 Months' Imprison...	February 3, 2026	Criminal and Civil Actions
1	Former NFL Player Convicted for $197M Medicare...	February 3, 2026	Criminal and Civil Actions
2	Florida Man Pleads Guilty to Conspiracy to Vio...	January 30, 2026	Criminal and Civil Actions
3	Yadkinville Woman Sentenced in Connection with...	January 29, 2026	Criminal and Civil Actions
4	Slidell Chiropractor Sentenced for Health Care...	January 28, 2026	COVID-19 | Criminal and Civil

Hint: if you go to James A. Robinson's profile page at the Nobel Prize website here, right-click anywhere along the line "Affiliation at the time of the award: University of Chicago, Chicago, IL, USA", and select Inspect, you'll see that this affiliation information is located at the third `<p>` tag out of five `<p>` tags under the `<div class="content">`. Think about how you can select the third element of `<p>` out of five `<p>` elements so you're sure you scrape the affiliation information, not other. This way, you can scrape the name of agency to answer this question.

(40%) Step 2: Making the scraper dynamic

1. **Turning the scraper into a function**: You will write a function that takes as input a month and a year, and then pulls and formats the enforcement actions like in Step 1 starting from that month+year to today.

    - It is *very important* to make sure that you include an indicator whether or not to actually run the function. If you do not include an indicator, then each time you try to knit the qmd file, the scraper will run, and it will take a very long time to compile your pdf. Instead, run the function once to create a file called enforcement_actions_year_month.csv, which you can use in subsequent parts of the pset. Then turn the indicator off so that knitting your qmd file goes smoothly.
    - This function should first check that the year inputted >= 2013 before starting to scrape. If the year inputted < 2013, it should print a statement reminding the user to restrict to year >= 2013, since only enforcement actions after 2013 are listed.
    - It should save the dataframe output into a .csv file named as "enforcement_actions_*year_month*.csv" (do not commit this file to git)

- If you're crawling multiple pages, always add 1 second wait before going to the next page to prevent potential server-side block. To implement this in Python, you may look up `.sleep()` function from `time` library.

a. Before writing out your function, write down pseudo-code of the steps that your function will go through. If you use a loop, discuss what kind of loop you will use and how you will define it. *Hint: Note that a simple `for` loop may not be sufficient for what this crawler requires. Use online resources to look into different types of loops or different ways of using `for` loops to see if there is something that is more appropriate for this task.*

1.Define a function scrape_enforcement_actions(start_year, start_month, run_scraper, out_dir, sleep_seconds)

2.If start_year < 2013, print a reminder and return None.

3.Create start_date as the first day of start_year-start_month

4.Define output CSV name enforcement_actions_{start_year}_{start_month:02d}.csv

5.If run_scraper is False: If the CSV exists, load it and return it Otherwise, print a message that the file does not exist and return None.

6.Initialize: page = 0 rows = []

7.While True Build URL for the current page: If page == 0, use the base URL(no page parameter) Else, use ?page = {page} Download HTML and parse with BeautifulSoup. Find all title links in the results list (e.g., h2 a).

```
For each title link:

    Find the parent container element for this entry.
    Extract: title text, href link, date string, category tags.Convert date string to a date
    If date >= start_date, append the row to rows.
    Else, stop processing further entries and stop crawling (because pages are sorted by dat
If we reached older-than-start_date OR the page has no entries, break the loop.
Sleep 1 second, then increment page += 1.
```

8.Convert rows to a DataFrame.

9.Save DataFrame to CSV in out_dir (do not commit).

10.Return the DataFrame.

b. Now code up your dynamic scraper and run it to start collecting the enforcement actions since January 2024. How many enforcement actions do you get in your final dataframe? What is the date and details of the earliest enforcement action it scraped?

c. Now, let's go a little further back. Test your code by collecting the actions since January 2022. *Note that this can take a while.* How many enforcement actions do you get in your final dataframe? What is the date and details of the earliest enforcement action it scraped? Use the dataframe from this process for every question after this.

Hint:

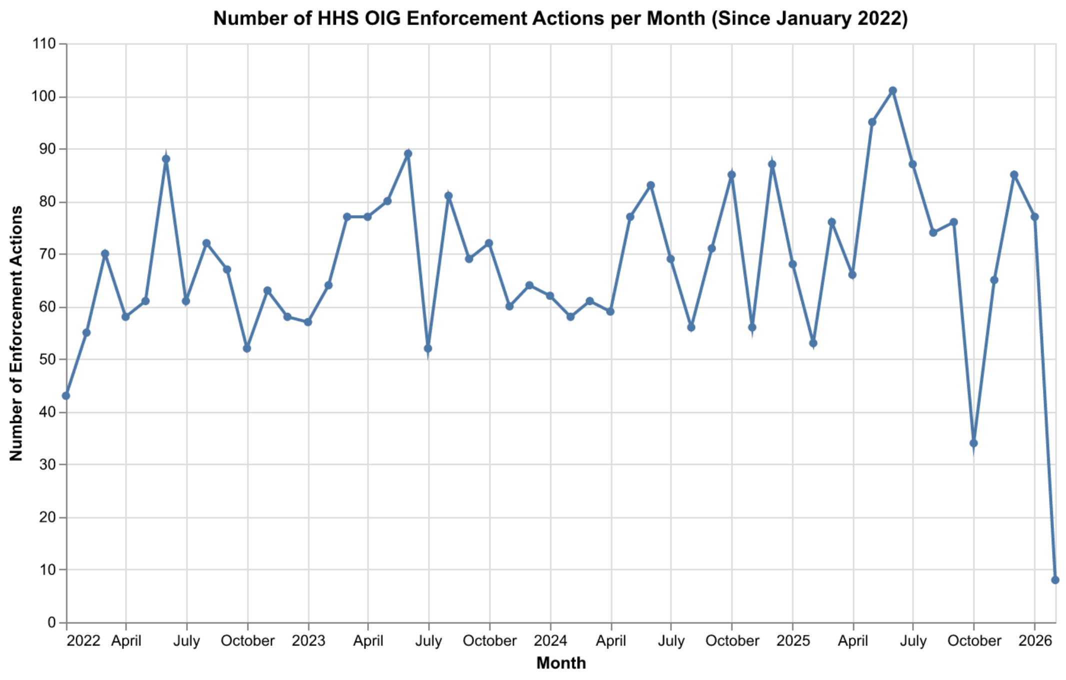
```
If you go to the next page in this HHS OIG's "Enforcement Actions" page, you'll notice a pat

  * Second page URL: https://oig.hhs.gov/fraud/enforcement/?page=2
  * Third page URL: https://oig.hhs.gov/fraud/enforcement/?page=3
  * and so on ...
```

```
Number of enforcement actions since Jan 2022: 3379
Earliest enforcement action scraped (since Jan 2022):
Date: January 4, 2022
Title: Integrated Pain Management Medical Group Agreed to Pay $10,000 for Allegedly Violatin
Category: Fraud Self-Disclosures
Link: https://oig.hhs.gov/fraud/enforcement/integrated-pain-management-medical-group-agreed-
Saved CSV: enforcement_actions_2022_01.csv
```

(20%) Step 3: Plot data based on scraped data

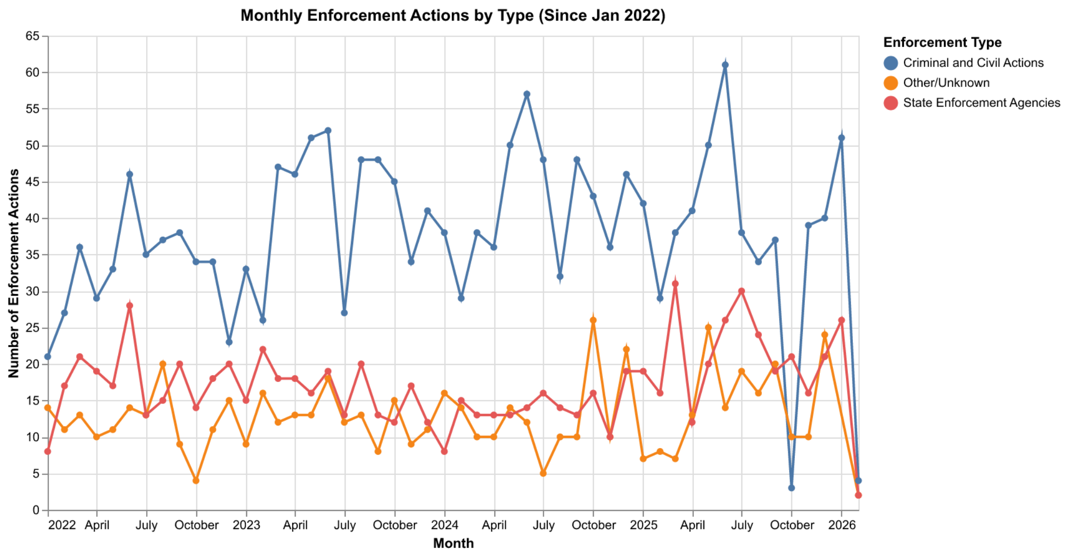
Note: To complete this part of the pset, reference the csv file you created in step 2, enforcement_actions_year_month.csv.

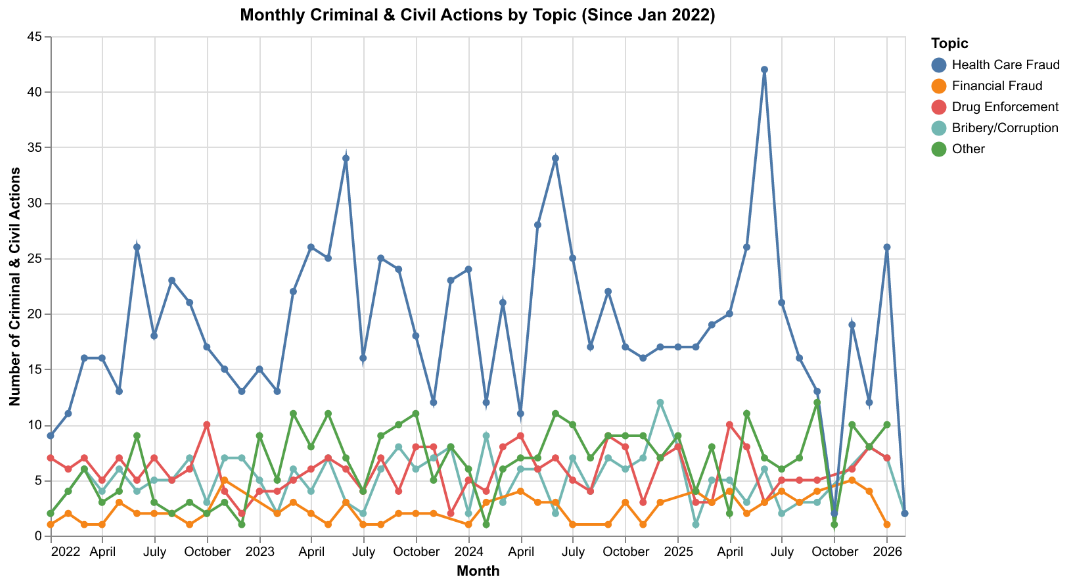
1. Plot a line chart with altair that shows: **the number of enforcement actions** over time (aggregated to each month+year) overall since January 2022,



2. Plot a line chart with altair that shows: **the number of enforcement actions** split out by:

   - "Criminal and Civil Actions" vs. "State Enforcement Agencies"
   - Five topics in the "Criminal and Civil Actions" category: "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. We suggest using AI to identify patterns in your scraped data and suggest ways of classifying based on the titles.*


Monthly Enforcement Actions by Type (Since Jan 2022)
Number of Enforcement Actions
Month
Enforcement Type
Criminal and Civil Actions
Other/Unknown
State Enforcement Agencies


Monthly Criminal & Civil Actions by Topic (Since Jan 2022)
Number of Criminal & Civil Actions
Month
Topic
Health Care Fraud
Financial Fraud
Drug Enforcement
Bribery/Corruption
Other