**Codebook for Subsidy Webscraping**

This is the “codebook” or walkthrough for the webscraping algorithm I used to collect, wrangle, clean subsidy data from Subsidy Tracker. Credit for the wonderful data goes to Subsidy Tracker, who makes their data available for free so long as you avoid bottle-necking their sever with requests. The web scraping files are:

* subsidy\_scrape\_basic\_data.R, which gathers subsidy-level data
* subsidy\_scrape\_detailed\_data.R, which gathers firm-level data

They should be on my Github page. If they are not, please contact me for the files.

*The Idea*

[Subsidy Tracker](https://subsidytracker.goodjobsfirst.org/) is a database you can query for cases of subsidies issued to US firms by local, state, and federal governments. For a given case, Subsidy tracker tries to code as much relevant information as possible. For the firm, it codes its legal name, parent company (if any), NAICS industry, headquarters location, etc. For the subsidy, it codes the amount given, the type of subsidy (e.g., grant, loan, tax freeze, tax credit, and +20 others), government plan through which the subsidy was given, and more. Sometimes the fields are present, other times not.

The idea is to write a webscraping algorithm that, through a series of steps, gathers this detailed information and puts it into an accessible data frame format. If there are subsidy cases and dimensions on which a “subsidy” can be measured, then we want an dataset. Each row will be a single subsidy observation, and each column will be a dimension.

What steps are needed to do this, and how many are there? The answer to this question depends on whether Subsidy Tracker’s website is dynamic. It is. If you visit Subsidy Tracker’s website, you’ll notice you can select (1) a US state and (2) a type of subsidy from a drop-down menu. These drop-down menus make the webpage user-friendly, but they also make the webpage dynamic. This means that if you want to *actively* change the state or the subsidy type, you have to manually click the drop-down menu and make a selection. If you do this, you’ll notice another dynamic element – a data table appears with lots of pages, and in order to cycle the pages you need to click an arrow button.

The dynamic structure is bothersome but easy to deal with. The web scraper needs to complete the following things: (1) visit the main Subsidy Tracker webpage; (2) select a state; (3) select a subsidy type; (4) retrieve, organize, and clean the data table that pops up; (5) click the next arrow; and (6) repeat until each data table page has been scraped. Then go back to the main page and repeat the process for the rest of the state-subsidy type combinations. This is just a for-loop over a state index and a subsidy type index, plus some meta loop things to make sure an error doesn’t stop the process.

One of the key parts of this web scraper is a data field – the webpage for the US firm receiving a subsidy. Every subsidy case has a firm URL that contains the information about where the firm is located, what industry it’s in, etc. By default, this feature gets scraped in the algorithm. However, I wanted to draw attention to it because it is so important. Later on, when the algorithm has scraped all the subsidy information, we’ll need to extract the URLs from our dataset and perform *another* loop that (1) visits the URL and (2) scrapes the firm-level information. This part is straightforward and does not involve a remote approach, though it does take a significantly long time. Since there are more than 200,000 subsidy cases, you need to make 200,000 webpage visits and scrapes.

**The Data**

At the date of construction, the final data frame has observations and dimensions. If you follow the .R cleaning file, you will notice that some of the scrapes had issues (e.g., for some reason, the “Maine” observations should be North Carolina); like all programs scraping the web, there are probably others issues that users will hopefully identify and let me know about.

Since there are *many* issues with how counties are named and coded, as well as with how much information is generally available for a subsidy, I decided to make two (main) datasets available. The first is the subsidy dataset *before* any cleaning was done (except for joining the web scraping batches). This is the dataset. The twenty-seven columns are:

1. address (character): the address of where the US firm is located
2. awarding\_agency (character): what government body gave the subsidy award; it can be named body or, in most cases, “multiple” unnamed bodies.
3. Capital\_investment (numeric):
4. Case (character): unique case number
5. City (character): city where receiving firm is located
6. Company (character): name of firm receiving subsidy
7. County (character): county where firm is located
8. Loan\_value\_notes (character): notes about the loan
9. Location (character): state where firm is located
10. Major\_industry\_of\_parent (character): broad industry of firm (not associated with a known coding system)
11. Naics\_industry\_code (character): 6-digit NAICS code of the firm
12. Notes (character): background information about the subsidy
13. Number\_of\_jobs\_or\_training (numeric): number of new jobs promised by firm
14. Parent\_company (character): parent company of firm
15. Program\_name (character): name of government program sponsoring the subsidy
16. Project\_description (character): short description of what the subsidy was given for
17. Source\_notes (character): information about where the subsidy information was found
18. Source\_of\_data (character): source information about the subsidy
19. Specific\_industry\_of\_parent (character): narrow description of firm’s industry (also not associated with a known coding system)
20. Subsidy\_source (character): government level that gave subsidy (local, state, federal, or multiple)
21. Subsidy\_value (numeric): dollar value of the subsidy
22. Type\_of\_subsidy (character): category of subsidy given
23. Value\_of\_loan\_guarantee\_or\_bailout\_assistance (character): NA
24. Wage\_data (numeric): wage promises for the jobs created from the subsidy
25. Wage\_data\_type (character): unit of measurement for wages (e.g., average wage per hour, average annual salary)
26. Year (character): year in which subsidy was given
27. Zip (character): zip code of firm

Most of these fields are completely useless, and most have far more missing observations than complete ones. For example, there is probably noting interesting about the “loan\_value\_notes” variable or the “wage\_data\_type” variable. Someone may find it interesting, and I hope they do; but it likely has very little practical use. I included it just to be complete.

The second main dataset is the dataset I cleaned. This is the dataset I wanted to use for political economy research, particularly on how subnational US governments distribute subsidies across industries and whether those industries are efficient and technological or inefficient and dying. Because I had a political economy mindset, I mostly paid attention to the following things: county names, subsidy values, and NAICS industry. Subsidy Tracker’s system for naming counties is, honestly, quite atrocious. When a firm was in more than one county, Subsidy Tracker either coded it as “multiple counties” or something similar, or listed each of the counties with an inconsistent separator (e.g., a comma, a vertical bar, or a semicolon). Other times, it would include the word “county” and take it away for others. Then there were simply a large number of cases for which the county variable was missing (which is understandable). All told, after trying to clean as best as possible, I ended up with a dataset. There is still *plenty* of missing data in this dataset, particularly in the “major\_industry\_of\_parent,” “specific\_industry\_of\_parent,” and “naics” dimensions. However, because the “company” field is almost always listed, there is still the possibility that a highly dedicated and resilient person could scour the web for the firms, find their NAICS identifier, and add it to the dataset. I wanted to make it possible for someone to do this, so I included the cases with missing industry data.