Data collection of doctors recommended by the US immigration offices using web scraping techniques in R.

Rutendo Madziwo , Maggie Szlosek , Rachel LaFlamme

Abstract

This project aims to scrape data from an interactive government website that has no endpoint urls using R and Docker. Our client, ProPublica wants to find out how many doctors recommended by the US government to green card applicants have outstanding malpractice suits against them. However, they do not know who these doctors are and where they are located. Our focus is, therefore, to collect information of all the doctors on the US immigration website by scraping the government website.

Introduction

We worked under the direct instruction of ProPublica for their article on medical malpractice among doctors recommended to green card applicants. The question they were trying to answer with this research was *How many doctors recommended by the US government to green card applicants have outstanding malpractice suits against them?* This is the main question our research is attempting to help answer by supplying them data to support their assumptions.

More specifically within this project, we are trying to answer *Who are the doctors being recommended to green card applicants, and how many are there?*. As there is no reliable and searchable database released by the government or any other entity that can be compared with a list of doctors with malpractice suits against them, it is our job to create this database for the use of ProPublica in their article.

Background

ProPublica is a news website that is focused on investigative journalism that "expose[s] abuses of power and betrayals of the public trust by government, business, and other institutions, using the moral force of investigative journalism to spur reform through the sustained spotlighting of wrongdoing." [1] It was established in 2008 in New York, NY, and continues to uphold this mission statement in a variety of disciplines, including but not limited to politics, civil rights, and education.

ProPublica plans on writing an upcoming article on doctors who have malpractice claims against them, but are still being recommended by the government to green card applicants. In order for an applicant to receive a green card, they must have an appointment and full examination with a state-sanctioned doctor. All of these recommended doctors can be found on the government's immigration website, my.uscis.gov/findadoctor.

The concern is that, because this examination is necessary for green card applicants and if something goes wrong, they are concerned that they may not receive their green card if they report any cases of malpractice by these doctors. ProPublica has collected testimonies of green card applicants that have experienced instances that would constitute malpractice, but did not report it to either the government or any other entity in fear of not being able to immigrate, which means that these doctors are still practicing and still being recommended by the government for these examinations.

While these testimonies appear to be the main part of the upcoming article, ProPublica would also like to include a data element by comparing doctors who are known to have malpractice suits filed against them in the past with doctors who are recommended to green card applicants via the government website. While ProPublica currently has a list of the doctors with malpractice violations, there is no reliable list of doctors approved by the goverment to give these examinations.

Our job was to collect the names and information of all of the doctors found on my.uscis.gov/findadoctor and compile them into an easily understood file that could then be compared with the preexisting list of doctors with known malpractice suits. In order to do this, we had to scrape the data found on this website.

We feel that this research is significant, because, if these violations are taking place, this is a clear violation of the safety of immigrants to this country. These people are in a vulnerable position because of their status as immigrants, and thus could be taken advantage of. If this is happening, it is important to expose it, and using tools such as data analysis, which is what ProPublica will be doing with this research, can help to accomplish this.

Our aim was to create a dataset based on the entries listed on the USCIS Find a Doctor website. Each entry on the website contains the name of the medical facility, name of the government-recommended doctor or doctors, and the address and phone number of the facility. However, the search bar on the website can only be used to search for doctors using US zipcode or city name. This creates an issue when initially thinking about how to build an algorithm because there are approximately 42,000 zipcodes in the U.S. and no reliable dataset that contains all the current zipcodes. The website is also structured in such a way that there are only 10 entries per webpage; however, when the user clicks the next button, they can see the doctors that are from a further radius than the initially typed zipcode. They can click the next button to span an approximately 500 mile radius before the website no longer provides a 'next' button.

For other websites, it is common to see that the search entry appears at the end of the url which would allow each url to have a single corresponding html page. However, with the USCIS site, there are thousands of html pages housed under one url. The website has a static url of https://my.uscis.gov/findadoctor regardless of the search entry and this eliminates a plethora of web-scraping methods such as get() and post().

Because of this, we looked for other options for R to interact with the webpage in order to bypass the non-changing url but still scrape the unique html pages accurately and effectively.

Methods

We collected the doctors' data using the package rvest for web-scraping, RSelenium for web navigation and an external platform Docker for virtual interaction with the web browser.

Docker is a platform to develop, deploy, and run applications inside containers[2]. We were able to virtually interact with the USCIS website by connecting to a port in Docker and opening Chrome and this enabled us to control and see what was happening on the website at a given time. Initially, Docker was installed before installing and loading the necessary R Packages. The command `docker run -d -p 4445:4444 selenium/standalone-chrome` was then run in the R Terminal after we had installed our packages. This command sets up the virtual Chrome container to enable interaction with the Chrome web browser. In order to check if Docker is running, one can type in `docker ps`. We eventually open the browser using RSelenium commands before scraping our data.

RSelenium is a package in R which helps one connect to a Selenium server [3]. This server in turn connects to the Chrome web browser and hence allowed us to automate our webscraping experience. RSelenium is responsible not only for opening and closing the browser, but it allowed us to virtually navigate the web page and automatically control the scraping. This was especially useful as our website had no endpoint urls and hence could not rely on more traditional web scraping methods. In addition, it made the process of scraping the data faster as one can simply allow the code to run and scrape multiple pages without needing to manually click the specific website.

While RSelenium was responsible for most of the web manouvering, the package we used for scraping the data from each of the pages was `rvest` [4]. This package makes harvesting data from a website easy as it can find specific html nodes, and their children. It also allows one to use both XPaths and CSS selectors so though we eventually stuck to using basic elements, we were not limited to one option. As a side note, we chose to use CSS selectors for web navigation with the RSelenium package.

We created a function to scrape this data and took advantage of the purrr package in R to map all our scraped elements together. This function was able to scrape all the exception datapoints that contained different or missing nodes from our standard entries with one doctor per medical facility such as multiple approvied doctors working in one facility and entries with no phone numbers listed. To clean up our data, `dplyr` and `tidyverse` were used for text-processing the such that zipcodes, states, and cities, were in separate columns from the rest of the address in the resulting doctors' dataset [5][6]. At the moment, this function is running in a for loop but will be converted to a while loop in order to allow for different state scenarios.

The final code written to collect the doctors' information allows a user to input one zipcode at a time in order to scrape data. Once that zipcode is entered, the doctors and facilities on that web page are harvested using `rvest` before moving on to the next page. Clicking to the next page has been automated using `RSelenium` and a for loop was implemented in our code such that for a certain number of times, the website's `Next` button is clicked, moves on to the next page, scrapes that page and so on. The website itself has been written in such a way that an actual user can keep clicking to find the nearest doctors within a 500 miles radius. As such, we have also manually entered different zipcodes in different parts of the USA so as to capture all the doctors in the country and create different datasets.

Results

At this time, we have been able to work out a code that accurately scrapes the website of all doctors within a 500-mile radius of a starting zip code, at which time the website no longer has a 'next' button. Then the code must run separately from a different origin point in order to find more doctors recommended by the website. At this point, we have run the code from three different locations including Northampton, Miami, and a remote location in Arizona and are still scraping the other locations. Propublica asked for a dataset that is organized by the following variables: name, address, city, state, zipcode, and phone number which is representative of all the government-approved doctors in the entirety of the United States at the time the website was scraped and have created this dataset. Since our dataset contains all categorical and string entries, minimal statistical work can be done aside from some summary statistics of the number of doctors scraped per zipcode, state, etc.(Results to follow)

Discussion

This data is useful because, based on information we received from our client, the undertaking of scraping this data from this website had not been reliably undertaken until now. This could be an important resource not just to understanding how prevalent medical malpractice is among this group of doctors immigrants are required to see, but also to find other trends possible within this group. In addition, we know that this will potentially help immigrants who are being taken advantage of by the current system, in which no protections are put in place against malpractice violations.

Limitations

There were a number of limitations we have with the current research and our product. Because of the method in which the website is set up, in order to get a full list of all of the doctors in the country, the code must be run multiple times from multiple origin points. If one needs just one region, this code will work much more quickly, but will not grant the user a full look at all of the doctors that are recommended by uscis.gov. This means that, even though we determined that this was the best example of code we could use, because of the format of the website, it still takes a long time to run and must run multiple times. While we hope to give a complete list to ProPublica of all doctors recommended, if they wish to check or if someone else wishes to replicate our work, the process for gathering data is relatively tedious. In addition, this code was written specifically for this particular website, meaning that it likely could not scrape data from a similar or even similar formatted website, as it is largely based on the individual HTML coding of uscis.gov. This makes the overall impact of this code on its own relatively small.

In addition, this data was gathered specifically for the use of ProPublica, meaning that we may not be able to spread this data without their express permission. This puts limits on the usefulness of our code, as the data could be used for other projects and transformed if other outlets see fit, particularly because this data has never been gathered before. However, because of ProPublica's role, we may not be able to share this information.

Another potential problem is that we do not know exactly how ProPublica plans to use this data set. While we understand the underlying goal, we have not seen the list of doctors that our dataset will be compared to and so we do not know whether or not the data will be used responsibly and ethically. While we understand ProPublica's mission and stand by it, and trust that our data will help rather than harm people, it is still a concern that is currently on our minds.

This dataset has the potential to help with many things, in addition to its intended purpose of contributing to this article on green card applicants and medical malpractice, authored by ProPublica. In addition, it could be used to compare doctors being recommended by the government to, for example, general sexual assault charges or other crimes, or just having a greater understanding of the types of medical doctors who are being recommended to immigrants. In addition to the useful information the data gathered provides, the code itself could potentially be manipulated to scrape other websites with more complex HTML formats, or at least help users gain a better understanding of how to potentially scrape these websites. In doing so, this could help to unlock other large databases that are, at this point, hidden to us and difficult to access.

References

1. ProPublica the mission. https://www.propublica.org/about/;
2. Docker docker home page. https://www.docker.com;
3. RSelenium r bindings for selenium 2.0 remote webdriver. http://ropensci.github.io/RSelenium/;
4. rvest easily harvest. http://rvest.tidyverse.org;
5. dplyr a grammar of data manipulation. https://dplyr.tidyverse.org;
6. tidyverse tidyverse. https://www.tidyverse.org;