Week 5 - Web Scraping

Data 201 - 422

2024-08-12

Overview

Today we will be introducing techniques to scrape data from the internet. Web scraping is a very useful tool as a data scientist. It allows you to harvest data from sources like the internet and API's. In this example, we will walk you through how to extract new data from websites via web scraping and then, you will be tasked to replicate what we have just walked through. As always, our good industry practices will be the first thing we set up in this project. Please use Firefox or Google Chrome for this exercise as we need to use the "Inspect elements" functionality that these browsers incorperate (see screenshot in GitHub repo). If you need more information about HTML structure, visit: https://rvest.tidyverse.org/articles/rvest.html this will explain the basic tree-like structure of HTML

Deliverables

We want you to build up a data frame of web scraped data from https://www.trustpilot.com/review/ubereats.com. This is a website of reviews pertaining to Uber Eats. In this example you should successfully scrape the reviews, the reviewer, date of review, number of stars for the corresponding review, and the title of the review. We will supply you with some helper functions to deal with messy strings. After you have successfully scraped these elements, you need to automate the process and scrape five pages. This will require you to use a *for-loop*, the string pasting method *paste0*.

This lab will be spread over two weeks. We understand web scraping isn't an easy topic. But if it was easy, everyone would be doing it!

Resources

As we'll be working with lists in RStudio, here's a link to the differences between lists and data frames in R: https://rstudio.github.io/r-manuals/r-intro/Lists-and-data-frames.html

Understanding for-loops: https://www.geeksforgeeks.org/for-loop-in-r/

Understanding inspector tool: https://zapier.com/blog/inspect-element-tutorial/

Example

We will use http://books.toscrape.com/ as our example website for web scraping, it is a mock website specifically designed to practice web scraping.

First, we will load our associated libraries. Use *install.packages("package_you_want_to_install")* in the console if you get an error pertaining to any of these libraries.

```
library(rvest)
library(tidyselect)
library(xml2)
library(tidyverse)
library(purrr)
library(readr)
```

Using the *read_html* function from rvest, we can parse the HTML of the webpage. if you call the variable page, you should see the HTML structure we have when you call *page*.

```
link <- "http://books.toscrape.com/"
page <- read_html(link)
# call page to see the HTML structure.
```

Once you have downloaded the HTML of the page you will be able to obtain the desired data. To do this you will have to select the desired elements or nodes with the html_element or html_elements functions to scrape the first or all occurrences. We recommend using the developer tools in Firefox or Chrome to find the associated attribute - You will have to do this yourself soon so it's a good idea to see if you can find the elements we have supplied and understand why they are being used. Ask a tutor if you need some help.

```
titles <- page %>%
  html_elements("h3")
```

We can see that the title is embedded into the HTML structure under the "h3" tag, to parse this data and convert our titles to text we can use html_text2().

```
titles <- page %>%
  html_elements("h3") %>% html_text2()
```

rvest provide two functions named html_element and html_elements which returns a single attribute or all attributes of a node. In the previous examples we got the titles of the books, but the titles were truncated. Inspecting the HTML of the site you can see that each h3 has a link with an attribute named title that contains the full name of each book, so to get them you can do the following

```
titles <- page %>%
  html_elements("h3") %>%
  html_elements("a") %>%
  html_attr("title")
```

Now we have a list of titles for all the books in the first page. Next we can try and obtain their associated prices.

```
book_price <- page %>%
  html_elements(".price_color") %>% html_text2()
```

```
title_tibble <- tibble(titles)
price_tibble <- tibble(book_price)

book_df <- bind_cols(title_tibble, price_tibble)
```

Your Turn

Now we have walked through an example around web scraping. We want you to build up a data frame of web scraped data from https://www.trustpilot.com/review/ubereats.com. This is a website of reviews pertaining to Uber Eats. In this example you should successfully scrape the reviews, the reviewer, date of review, number of stars for the corresponding review, and the title of the review. We will supply you with some helper functions to deal with messy strings. After you have successfully scraped these elements, you need to automate the process and scrape five pages. This will require you to use a *for-loop*, the string pasting method *paste0*.

To get to the data, you will need some functions of the rvest package, like we used earlier. To convert a website into an XML object, you use the read_html() function. To extract the relevant nodes from the XML object you use html_nodes(), whose argument is the class descriptor, prepended by a . to signify that it is a class. The output will be a list of all the nodes found in that way. To extract the tagged data, you need to apply html_text() to the nodes you want. For the cases where you need to extract the attributes instead, you apply html_attrs(). This will return a list of the attributes, which you can subset to get to the attribute you want to extract. Look at the documentation about the associated methods if you are struggling to understand. Or ask a tutor!

```
#The link you will need to start.

page <- read_html("https://www.trustpilot.com/review/ubereats.com")

################################################################################

#Function to help convert date strings to a real date format.

extract_dates <- function(review_date) {

  dates <- sub("Date of experience: ", "", review_date)
  dates <- mdy(dates)

  return(dates)
}

################################################################################

#Helpers to help parse string ratings, you will need to employ something
#else to grab the first number which corresponds to the star rating.
#There is another step that will void having to use this func if you can figure it out.

alt_texts <- sapply(ratings, function(x) x["alt"], simplify = TRUE)
alt_texts <- unname(alt_texts)
print(alt_texts)

################################################################################

#Helper to isolate the rating x out of 5 stars. we are taking x.
#How can i make this number an int after?

extract_rating <- function(rating_text) {
  sub("Rated (\\d) out of 5 stars", "\\1", rating_text)
}
```

Extra for experts

you have now successfully scraped five pages worth of data, now use lapply to vectorize this process. The resulting data frame should be the same as your data frame that you just scraped. Good luck!