How to build a webscraper in R?

Youssef El Bouhassani, y.elbouhassani@gmail.com

2017-11-22T13:09:13-06:00

Context

Webscraping is a handy way to get information from webpages. Websites like https://www.thefork.nl/ contain information on indivual restaurants. In this tutorial a simple webscraper will be built that gets the names and ratings of individual restaurants.

Approach

When searching for restaurants, say in Amsterdam, https://www.thefork.nl/ returns multiple pages. Each page contains an excerpt information and detailed information if you click on individual restaurants. The pipeline for the scraper looks as follows

1. From the webpage get urls linking to the individual pages
2. From each page url get the urls linking to the individual restaurants
3. From the restarant urls get name and rating (and other information if necessary)

Building the scraper

Step 0: getting things ready

SelectorGadget

Websites are built using `html` and `css` code. The `css` code contains information on the tags of the individual elements. To extract information form the website we have to indicate which tages we are looking for. This can be done using the inspect function on the browser. A more handy tool is the SelectorGadget. This is an add on that allows you to easily find the `css` tags you are looking for. A quick tutorial van be found on https://vimeo.com/23269072.

R Libraries

To create the scraper we need two important `libraries`: `rvest` for webscraping and `dplyr` for data manipulation. Install and activate libraries

The `rvest` package contains a number of function that we will use:

1. `read_html()`. Gets all `html` code given a url
2. `html_nodes()`. Gets the information associated with the given tag.
3. `html_text()`. Converst `html` code to readable text
4. `html_attr()`. Gets the attributes associated with the `html` code If the underslying attribute is a hyperlink we use `html_attr(href)` to get the hyperlink.

These functions can be chaned using the `dplyr` pipe operator as follows:

```
url %>% read_html() %>% html_nodes(TagName) %>% html_text()
```

With `TagName` being the name of the relevant `css` tag.

Step 1: Get the website URL

We are looking for information on restaurants in Amsterdam. When you search for Amsterdam in https://www.thefork.nl/ you get the following URL:

Step 2: Get the links to individual restaurants

The URL shows that there are multiple pages which can be navigated using the next and previous button on the bottom of the page. The scraper should recognize the number of pages available. We use the SelectorGadget to find the tag associated with the last page. In this case this tag is `#pagination_results li:nth-child(21) a`. Based on the the last page index we can create a list of urls for all pages. The structure of a page is as follows:

`https://www.thefork.nl/restaurant+amsterdam?page=1`

By appending the number of the page to this url, we can create a list of urls for all pages.

From `urllist` we can get the urls of the individual restaurants.

We don't want to do that page by page, so an `lapply` function will do the trick. Keep in mind that it might take some time to complete running this function.

Step 3. Get information on individual restaurants

Now we have a list containing all urls for individual restaurants, we can get the relevant information from these pages. For now we focus on getting the name and the rating of the restaurants.

The data will be put in a data frame. So first we create an empty data frame with two columns. One for names of restaurants and one for the ratings. Then we loop through all the individual restaurant urls, get `html` code and the text associated with the `css` tags for name and rating.

When selecting the tags keep in mind that we are interested in the average rating mentioned next to the restaurant name. When using the SelectorGadget to select the ratings, the ratings by individuals reviewers are selected as well. So you have to click these away in orther to get the tag of the average rating.

It is important to keep in mind that information on restaurants might be missing. In this case we have to explicitly state that an `NA` should be used for missing information using a simple `ifelse()`. If this is not done, an error message will be given when using the `rbind` function stating that the data frames have different number of rows.

Possible issues

One of the issues you might encounter is a time out warning. This might have multiple reasons. For instance if the internal signal is down for a brief moment. If this happens after the loop in step 3 is been running for a while, you don't want to start all over again. You look at the index `i` to see where the loop stopped and change the starting index in the for loop. So instead of using `for(i in 1:length(res_urls))` you can use `for(i in lastindex + 1:length(res_urls))` with `lastindex` being the value of `i` at the time when the loop stopped. Make sure you just run the loop without initiating the data frame using `df <- data.frame(naam = character(0),rating = numeric(0))`. If you do this, the data frame will be cleared and you will have to start all over again.

Complete code

```r
######################### STEP 0: GETTING THINGS READY

# Clean the workspace
rm(list = ls())

# Activate libraries
library(rvest)
library(dplyr)

######################### STEP 1: GET THE WEBSITE URL

# Define website URL
url <- "https://www.thefork.nl/restaurant+amsterdam"

######################### STEP 2: GET LINKS TO INDIVIDUAL RESTAURANTS
# Get the number of the last page
lastpage <- url %>%
            read_html() %>%
            html_nodes("#pagination_results a") %>%
            html_text() %>%
            as.numeric() %>%
            tail(.,2) %>%
            head(.,1)

# Create a list of all urls starting from 1 untill the last page
urllist <- paste("https://www.thefork.nl/restaurant+amsterdam?page=",1:lastpage,sep="")

# Read html from urllist to get the urls of idividual restaurants
res_urls <- lapply(urllist,function(x){read_html(x) %>%
            html_nodes(".resultItem-name a") %>%
            html_attr("href") %>%
            paste("https://www.thefork.nl",.,sep="")}) %>%
            unlist()

######################### STEP 3: GET INFROMATION ON INDIVIDUAL RESTAURANTS

# From the individuals restaurant urls, read the name and the rating
# Create an empty data frame with the right column names
df <- data.frame(naam = character(0),rating = numeric(0))

for(i in 1:length(res_urls)) {
  # Per restraurant read the html code
  res_html <- res_urls[i] %>% read_html()
  # Get name
  naam <- res_html %>%
          html_nodes(".restaurantSummary-name") %>%
          html_text() %>%
          as.character() %>%
          ifelse(is.null(.),NA,.)

  # Get rating
  rating <- res_html %>%
            html_nodes("#restaurantAvgRating .rating-ratingValue") %>%
```

```
            html_text() %>% gsub(",",".",.) %>%
            as.numeric() %>%
            ifelse(is.null(.),NA,.)

  # Add the new information to the old dataframe
  df <- rbind(df,data.frame(naam,rating))
}


#write to csv
write.csv(df,"theforkrating.csv",row.names = F)
```