Web Scrpping

02-25-2020

```
library(tidyverse)
```

```
## -- Attaching packages ----------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts -------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rvest)
```

```
## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

HTML and XML

Here is an example of a simple HTML page:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Of course, there are a lot of different tags. Go to https://www.w3schools.com/html/ to see some html basics. Nevertheless, all these tags are predefined so your browser knows what each tag means.

(We only need to only a bit HTML in order to do web scrapping)

Look at web page source code

It is important to identify the thing that you want to scrape in the webpage. The best way to do it is to use the inspect function in the Chrome browser.

imdb example

Suppose we want to get the list of most top rated movies from https://www.imdb.com/chart/top/?ref_=nv_mv_250

We see that all movies names are under the `<td>/<a>` nodes. The `<td>` node has a class name `titleColumn`.

```r
html <- read_html("https://www.imdb.com/chart/top/?ref_=nv_mv_250")
# it finds all the <td> nodes, but we only need the node with class `titleColumn`
td_nodes <- html %>% html_nodes("td")
# it finds all the <td> nodes with class titleColumn
title_columns <- html %>% html_nodes("td.titleColumn")
# it finds all the <a> nodes within <td> nodes with class titleColumn
a_nodes <- title_columns %>% html_nodes("a")
# html_text to get the values inside the <a> </a> tag
movie_names <- a_nodes %>% html_text()
head(movie_names)
```

```
## [1] "The Shawshank Redemption" "The Godfather"
## [3] "The Godfather: Part II"   "The Dark Knight"
## [5] "12 Angry Men"             "Schindler's List"
```

Now, we also want to capture the ratings.

```r
imdb_ratings <-  html %>%
  html_nodes("td.ratingColumn.imdbRating") %>%
  html_nodes("strong") %>%
  html_text()
```

```r
tibble(title = movie_names, rating = imdb_ratings)
```

```
## # A tibble: 250 x 2
##    title                                             rating
##    <chr>                                             <chr>
##  1 The Shawshank Redemption                          9.2
##  2 The Godfather                                     9.1
##  3 The Godfather: Part II                            9.0
##  4 The Dark Knight                                   9.0
##  5 12 Angry Men                                      8.9
##  6 Schindler's List                                  8.9
##  7 The Lord of the Rings: The Return of the King     8.9
##  8 Pulp Fiction                                      8.9
##  9 The Good, the Bad and the Ugly                    8.8
## 10 The Lord of the Rings: The Fellowship of the Ring 8.8
## # ... with 240 more rows
```

How if you also want to get the years?

```
years <- html %>%
  html_nodes("td.titleColumn") %>%
  html_nodes("span.secondaryInfo") %>%
  html_text() %>%
  str_extract("\\d+")
```

```
tibble(title = movie_names, year = years, rating = imdb_ratings)
```

```
## # A tibble: 250 x 3
##    title                                             year  rating
##    <chr>                                             <chr> <chr>
##  1 The Shawshank Redemption                          1994  9.2
##  2 The Godfather                                     1972  9.1
##  3 The Godfather: Part II                            1974  9.0
##  4 The Dark Knight                                   2008  9.0
##  5 12 Angry Men                                      1957  8.9
##  6 Schindler's List                                  1993  8.9
##  7 The Lord of the Rings: The Return of the King     2003  8.9
##  8 Pulp Fiction                                      1994  8.9
##  9 The Good, the Bad and the Ugly                    1966  8.8
## 10 The Lord of the Rings: The Fellowship of the Ring 2001  8.8
## # ... with 240 more rows
```

There is also a cute function `html_table`.

```
html %>% html_node("table.chart.full-width") %>%
  html_table() %>%
  as_tibble(.name_repair = "unique") %>%
  select(rank_and_title = `Rank & Title`, rating = `IMDb Rating`) %>%
  separate(rank_and_title, c("rank", "title", "year"), sep = "\n")
```

```
## New names:
## * `` -> ...1
## * `` -> ...5
```

```
## # A tibble: 250 x 4
##    rank  title                                           year          rating
##    <chr> <chr>                                           <chr>          <dbl>
##  1 1.    "      The Shawshank Redemption"                "         (199~    9.2
##  2 2.    "      The Godfather"                           "         (197~    9.1
##  3 3.    "      The Godfather: Part II"                  "         (197~    9
##  4 4.    "      The Dark Knight"                         "         (200~    9
##  5 5.    "      12 Angry Men"                            "         (195~    8.9
##  6 6.    "      Schindler's List"                        "         (199~    8.9
##  7 7.    "      The Lord of the Rings: The Return of the ~ "         (200~    8.9
##  8 8.    "      Pulp Fiction"                          "        (199~    8.9
##  9 9.    "      The Good, the Bad and the Ugly"        "        (196~    8.8
## 10 10.   "      The Lord of the Rings: The Fellowship of ~ "     (200~    8.8
## # ... with 240 more rows
```

Now, we want to url link to the movie “The Shawshank Redemption”.

```
shawshank_url <- html %>%
  html_nodes("td.titleColumn") %>%
  html_nodes("a") %>%
  keep(html_text(.) == "The Shawshank Redemption") %>%
  html_attr("href")
```

But it is not the complete url, we need to base url.

```
shawshank_full_url <- str_c("https://www.imdb.com/", shawshank_url)
```

Then we could futher scrape things from `shawshank_full_url`.

Treat HTML as (kind of) XML

What is XML?

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

See https://www.w3schools.com/xml/ for more XML examples.

In the XML technology, there is a powerful tool called `XPath`. See https://www.w3schools.com/xml/xpath_syntax.asp for details.

The XPath syntax is quite powerful, we could get the url for Shawshank redeption in almost one line.

```
html %>%
  html_nodes(xpath = "
     //td[@class = 'titleColumn']
     /a[text() = 'The Shawshank Redemption']
     /@href") %>%
  html_text()
```

```
## [1] "/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-8962-327b42fe94b1&pf_rd_r=N
```

We could also select all movies which has rating > 8.5.

```r
html %>%
  html_nodes(xpath = "
      //td[@class = 'ratingColumn imdbRating']
      /strong[text() > 8.5]
      /../../td[@class = 'titleColumn']/a/text()") %>%
  html_text()
```

```
##  [1] "The Shawshank Redemption"
##  [2] "The Godfather"
##  [3] "The Godfather: Part II"
##  [4] "The Dark Knight"
##  [5] "12 Angry Men"
##  [6] "Schindler's List"
##  [7] "The Lord of the Rings: The Return of the King"
##  [8] "Pulp Fiction"
##  [9] "The Good, the Bad and the Ugly"
## [10] "The Lord of the Rings: The Fellowship of the Ring"
## [11] "Fight Club"
## [12] "Forrest Gump"
## [13] "Inception"
## [14] "Star Wars: Episode V - The Empire Strikes Back"
## [15] "The Lord of the Rings: The Two Towers"
## [16] "The Matrix"
## [17] "Goodfellas"
## [18] "One Flew Over the Cuckoo's Nest"
## [19] "Seven Samurai"
## [20] "Se7en"
## [21] "City of God"
## [22] "Parasite"
## [23] "Life Is Beautiful"
## [24] "The Silence of the Lambs"
## [25] "It's a Wonderful Life"
## [26] "Star Wars: Episode IV - A New Hope"
```

Scrapping dynamic webpages

`rvest` is only able to scrape static web pages. If you want to scrape dynamic web pages, we need to start a headless browser.

https://stats.nba.com/ is one of such websites that are not static.

PS: actually nba.com has a undocumented API, see https://github.com/seemethere/nba_py/wiki/stats.nba.com-Endpoint-Documentation

```r
library(RSelenium)
```

```r
# Your could use different browser
server <- wdman::phantomjs(port = 4567L, verbose = FALSE)
# server <- wdman::chrome(port = 4567L, version = "80.0.3987.106", verbose = FALSE)
rd <- remoteDriver(browserName = "phantomjs", port = 4567L)
```

```r
rd$open(silent = TRUE)
rd$navigate("https://stats.nba.com/leaders/?SeasonType=Regular%20Season")
```

Have no idea what's happening?

```r
rd$screenshot(display = TRUE)
```

```r
rd$getPageSource() %>%
  str_flatten() %>%
  read_html() %>%
  html_node("div.nba-stat-table__overflow table") %>%
  html_table()
```

```
##   #                Player GP  MIN  PTS  FGM  FGA  FG% 3PM  3PA  3P%  FTM  FTA
## 1 1          James Harden 55 36.7 35.2 10.2 23.0 44.3 4.6 12.7 36.3 10.2 11.8
## 2 2          Bradley Beal 50 35.8 30.1 10.3 22.5 45.7 2.7  8.1 33.3  6.9  8.1
## 3 3            Trae Young 54 35.5 30.1  9.3 20.9 44.3 3.5  9.5 36.9  8.0  9.3
## 4 4 Giannis Antetokounmpo 52 30.9 29.7 10.9 19.8 55.1 1.5  4.7 31.4  6.4 10.3
##    FT% OREB DREB  REB AST STL BLK TOV  EFF
## 1 86.3  1.1  5.3  6.4 7.3 1.7 0.9 4.4 32.7
## 2 84.7  1.0  3.4  4.4 6.0 1.1 0.4 3.4 25.0
## 3 85.9  0.5  3.9  4.4 9.2 1.1 0.1 4.8 27.1
## 4 62.1  2.3 11.3 13.7 5.8 1.1 1.1 3.7 34.8
##  [ reached 'max' / getOption("max.print") -- omitted 46 rows ]
```

loop over the table by clicking the next button

```r
leader <- NULL
for (i in 1:6) {
  leader <- rd$getPageSource() %>%
    str_flatten() %>%
    read_html() %>%
    html_node("div.nba-stat-table__overflow table") %>%
    html_table() %>%
    bind_rows(leader, .)
  nextbutton <- rd$findElement("css", "a.stats-table-pagination__next")
  nextbutton$clickElement()
}
```

```r
leader
```

```
##   #                Player GP  MIN  PTS  FGM  FGA  FG% 3PM  3PA  3P%  FTM  FTA
## 1 1          James Harden 55 36.7 35.2 10.2 23.0 44.3 4.6 12.7 36.3 10.2 11.8
## 2 2          Bradley Beal 50 35.8 30.1 10.3 22.5 45.7 2.7  8.1 33.3  6.9  8.1
## 3 3            Trae Young 54 35.5 30.1  9.3 20.9 44.3 3.5  9.5 36.9  8.0  9.3
## 4 4 Giannis Antetokounmpo 52 30.9 29.7 10.9 19.8 55.1 1.5  4.7 31.4  6.4 10.3
##    FT% OREB DREB  REB AST STL BLK TOV  EFF
## 1 86.3  1.1  5.3  6.4 7.3 1.7 0.9 4.4 32.7
## 2 84.7  1.0  3.4  4.4 6.0 1.1 0.4 3.4 25.0
## 3 85.9  0.5  3.9  4.4 9.2 1.1 0.1 4.8 27.1
## 4 62.1  2.3 11.3 13.7 5.8 1.1 1.1 3.7 34.8
##  [ reached 'max' / getOption("max.print") -- omitted 264 rows ]
```

```
# close the browser finally.
server$stop()
```

```
## [1] TRUE
```