Web Scraping

Stat 220

Bastola

February 11 2022

Web scraping

the process of downloading, parsing, and extracting data presented in an HTML file and then converting it into a structured format that allows us to analyze it.

Two different scenarios:

1. **Screen scraping:** extract data from source code of website, with html parser (easy) or regular expression matching (less easy).
2. **Web APIs (application programming interface):** website offers a set of structured http requests that return JSON or XML files.

`robotstxt` for permission

Use `robotstxt::paths_allowed()` to see if you have permission to scrape

You can scrape IMDB

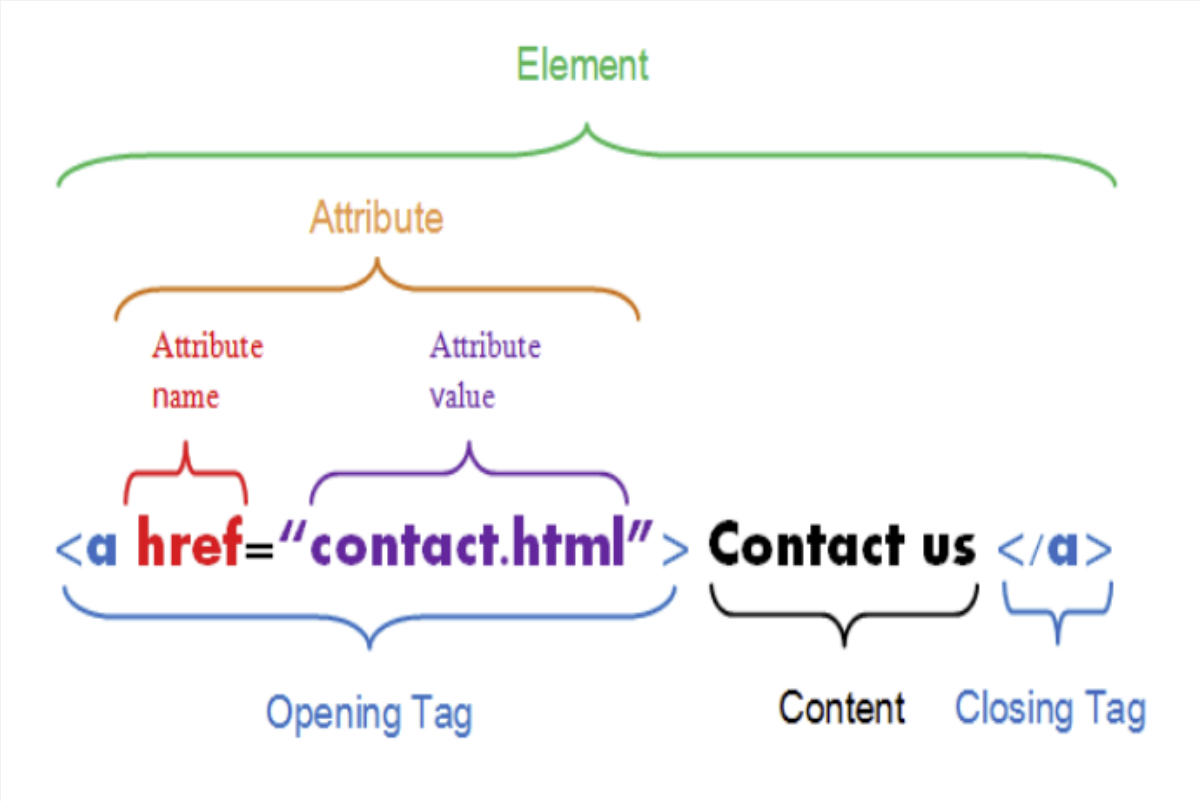
```
library(robotstxt)
paths_allowed("http://www.imdb.com")
[1] TRUE
```

But not Facebook!

```
paths_allowed("http://www.facebook.com")
[1] FALSE
```

HyperText Markup Language (HTML)

HTML page consists of series of elements which browsers use to interpret how to display the content



HyperText Markup Language (HTML)

While it is structured (hierarchical/tree based) it often is not available in a form useful for analysis (flat / tidy).

```
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p align="center">Hello world!</p>
  </body>
</html>
```

HTML tags

HTML uses `tags` to describe different aspects of document content

Tag	Example
heading	`<h1>My Title</h1>`
paragraph	`<p>A paragraph of content...</p>`
table	`<table> ... </table>`
anchor (with attribute)	`<a href="http://www.ratebeer.com">click here for link</a>`

Makes basic processing and manipulation of HTML data straight forward.

Core `rvest` functions

Function	Description
`read_html`	Read HTML data from a url or character string
`html_node`	Select a specified node from HTML document
`html_nodes`	Select specified nodes from HTML document
`html_table`	Parse an HTML table into a data frame
`html_text`	Extract tag pairs' content
`html_name`	Extract tags' names
`html_attrs`	Extract all of each tag's attributes
`html_attr`	Extract tags' attribute value by name

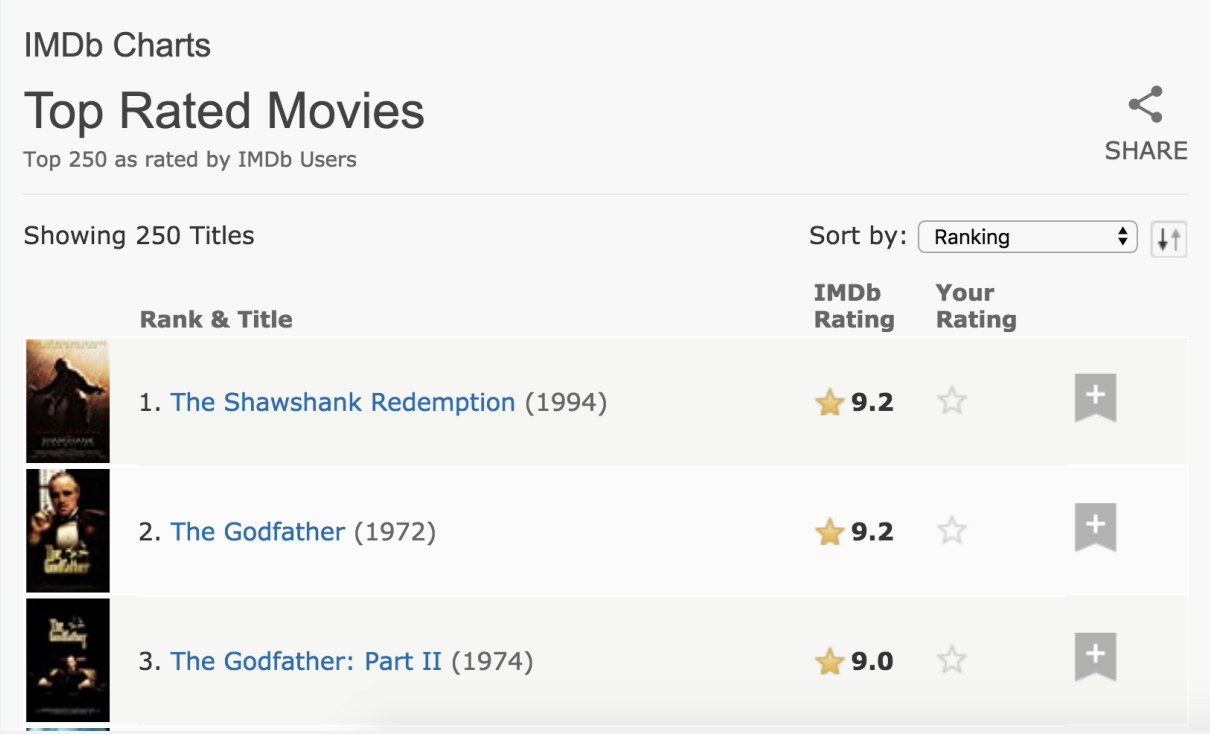
Top 250 movies on IMDB

http://www.imdb.com/chart/top

- Take a look at the web page **and** the html source code

  Chrome: right click -> View page source

- Look for the tag `<table>` tag



Read HTML into R

```
page <- read_html("http://www.imdb.com/chart/top")
```

```
page
{html_document}
<html xmlns:og="http://ogp.me/ns#" xmlns:fb="http://www.facebook.com/2008/fbml">
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
[2] <body id="styleguide-v2" class="fixed">\n            <img height="1" widt ...
```

```
str(page)
List of 2
 $ node:<externalptr>
 $ doc :<externalptr>
 - attr(*, "class")= chr [1:2] "xml_document" "xml_node"
```

Extract tables

Use `html_nodes()` to extract pieces out of HTML documents

```
tables <- page %>% html_nodes("table")
```

```
str(tables)
List of 1
 $ :List of 2
  ..$ node:<externalptr>
  ..$ doc :<externalptr>
  ..- attr(*, "class")= chr "xml_node"
 - attr(*, "class")= chr "xml_nodeset"
```

Not a data frame yet!

It points to the correct node ...

```
tables
{xml_nodeset (1)}
[1] <table class="chart full-width" data-caller-name="chart-top250movie">\n<c ...
```

but no data frame yet!

```
tables[[1]]
{html_node}
<table class="chart full-width" data-caller-name="chart-top250movie">
[1] <colgroup>\n<col class="chartTableColumnPoster">\n<col class="chartTableC ...
[2] <thead><tr>\n<th></th>\n          <th>Rank &amp; Title</th>\n          <t ...
[3] <tbody class="lister-list">\n<tr>\n<td class="posterColumn">\n\n    <span ...
```

Parse a table into a data frame

```
top250 <- html_table(tables[[1]])
glimpse(top250)
Rows: 250
Columns: 5
$ ``              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
$ `Rank & Title` <chr> "1.\n      The Shawshank Redemption\n        (1994)", "…
$ `IMDb Rating`  <dbl> 9.2, 9.1, 9.0, 9.0, 8.9, 8.9, 8.9, 8.8, 8.8, 8.8, 8.7, …
$ `Your Rating`  <chr> "12345678910\n        \n        \n            \n      …
$ ``              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
```

Parse tables into data frames

`html_table()` is vectorized so you can input a list of HTML tables and it will return a list of data frames

```
table_list <- html_table(tables)
str(table_list)
List of 1
 $ : tibble [250 × 5] (S3: tbl_df/tbl/data.frame)
  ..$              : logi [1:250] NA NA NA NA NA NA ...
  ..$ Rank & Title: chr [1:250] "1.\n      The Shawshank Redemption\n        (1994)"
  ..$ IMDb Rating : num [1:250] 9.2 9.1 9 9 8.9 8.9 8.9 8.8 8.8 8.8 ...
  ..$ Your Rating : chr [1:250] "12345678910\n        \n        \n            \n
  ..$              : logi [1:250] NA NA NA NA NA NA ...
```

Your Turn 1

05:00

Please clone the repository on web scraping to your local folder.

- Why isn't `top250` data frame we just scraped tidy? Use your data-wrangling toolkit to create a tidy data set with columns: `rank`, `title`, `year`, and `imdb.rating`

rank	title	year	imdb.rating
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

CSS

- CSS (Cascading Style Sheets) is a language that describes how HTML elements should be displayed.
- **CSS selectors:**
  - shortcuts for selecting HTML elements to style
  - can also be used to extract the content of these elements

IMDb Menu All Search IMDb IMDbPro

Best Picture-Winning (Sorted by Year Descending)

94 titles. View Mode: Compact | **Detailed**

Sort by: Popularity | A-Z | User Rating | Number of Votes | US Box Office | Runtime | **Year ▼** | Release Date | Date of Your Rating | Your Rating

1. **Nomadland** (2020)

R | 107 min | Drama

7.3 Rate this 93 Metascore

A woman in her sixties, after losing everything in the Great Recession, embarks on a journey through the American West, living as a van-dwelling modern-day nomad.

Director: Chloé Zhao | Stars: Frances McDormand, David Strathairn, Linda May, Gay DeForest

Votes: 144,633

2. **Parasite** (2019)

R | 132 min | Comedy, Drama, Thriller

8.6 Rate this 96 Metascore

Greed and class discrimination threaten the newly formed symbiotic relationship between the wealthy Park family and the destitute Kim clan.

Director: Bong Joon Ho | Stars: Kang-ho Song, Sun-kyun Lee, Yeo-jeong Cho, Woo-sik Choi

Votes: 716,013 | Gross: $53.37M

3. **Green Book** (2018)

PG-13 | 130 min | Biography, Comedy, Drama

8.2 Rate this 69 Metascore

A working-class Italian-American bouncer becomes the driver of an African-American classical pianist on a tour of venues through the 1960s American South.

Director: Peter Farrelly | Stars: Viggo Mortensen, Mahershala Ali, Linda Cardellini, Sebastian Maniscalco

Votes: 449,769 | Gross: $85.08M

4. **The Shape of Water** (2017)

R | 123 min | Drama, Fantasy, Romance

7.3 Rate this 87 Metascore

At a top secret research facility in the 1960s, a lonely janitor forms a unique relationship with an amphibious creature that is being held in captivity.

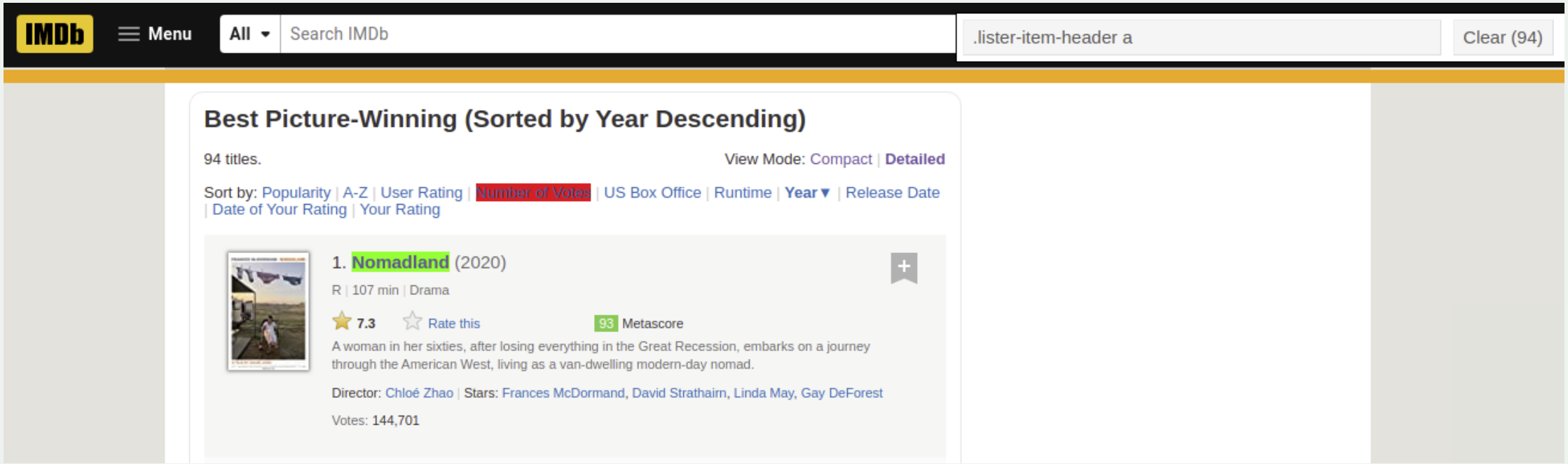
Director: Guillermo del Toro | Stars: Sally Hawkins, Octavia Spencer, Michael Shannon, Doug Jones

Votes: 403,301 | Gross: $63.86M

Click here to take a look at the webpage

SelectorGadget

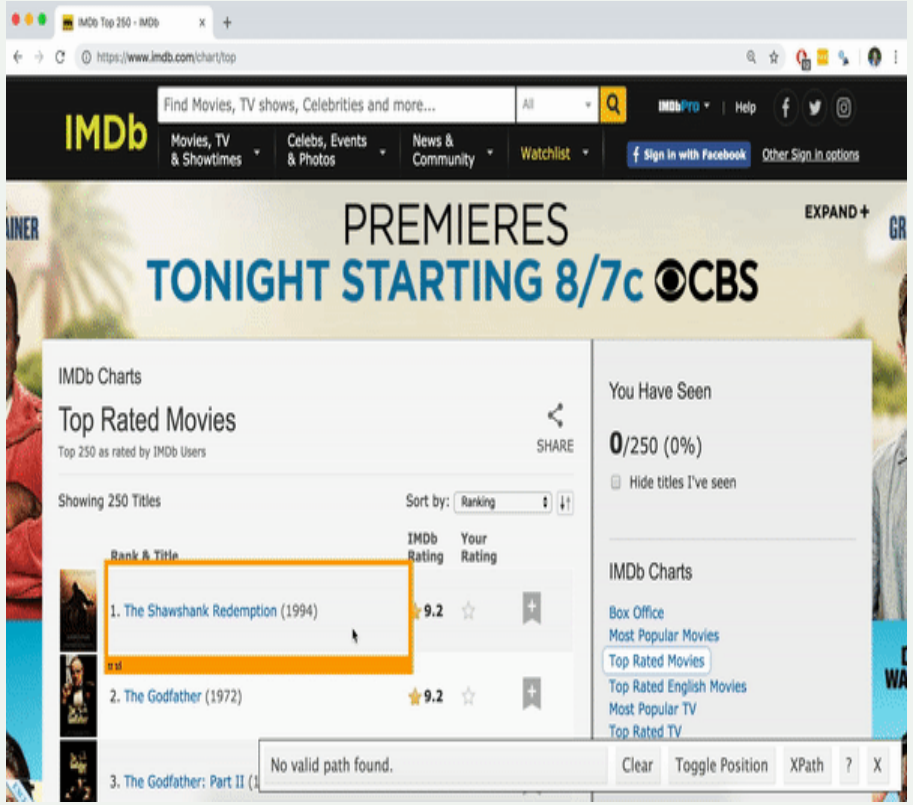
- SelectorGadget is a point-and-click CSS selector, specifically for **Chrome**
- Comes as a Chrome Extension (Click to install!)



Click here for a list of selectors

SelectorGadget

- Select all elements that are related to that object. Next, select anything in yellow you do not want



Source: Adam Loy

Read HTML into R

```
webpage <- read_html('https://www.imdb.com/search/title/?groups=best_picture_
```

```
webpage
{html_document}
<html xmlns:og="http://ogp.me/ns#" xmlns:fb="http://www.facebook.com/2008/fbml">
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
[2] <body id="styleguide-v2" class="fixed">\n            <img height="1" widt ...
```

```
str(webpage)
List of 2
 $ node:<externalptr>
 $ doc :<externalptr>
 - attr(*, "class")= chr [1:2] "xml_document" "xml_node"
```

Extract titles

Use `html_nodes()` to extract pieces out of HTML documents

```
title_data <- webpage %>% html_nodes(".lister-item-header a") %>% html_text()
```

```
title_data
 [1] "Nomadland"
 [2] "Parasite"
 [3] "Green Book"
 [4] "The Shape of Water"
 [5] "Moonlight"
 [6] "Spotlight"
 [7] "Birdman or (The Unexpected Virtue of Ignorance)"
```

Your Turn 2

03:00

- Use the selector gadget tool to find the CSS for extracting **year** the movie came out.
- Tidy the data
  - Using `parse_number()`
  - Using regex

Final Result:

```
 [1] 2020 2019 2018 2017 2016 2015 2014 2013 2012 2011 2010 2008 20
[16] 2004 2004 2003 2002 2001 2000 1999 1998 1997 1996 1995 1994 19
[31] 1990 1989 1988 1987 1986 1985 1984 1983 1982 1981 1980 1979 19
[46] 1975 1974 1973 1972 1971 1970 1969 1968 1967 1966 1965 1964 19
[61] 1960 1959 1958 1957 1956 1955 1954 1953 1952 1951 1950 1949 19
[76] 1945 1944 1942 1942 1941 1940 1939 1938 1937 1936 1935 1934 19
[91] 1930 1929 1927 1927
```

Your Turn 3

```
[1] "A woman in her sixties, after losing everything in the Great F
[2] "Greed and class discrimination threaten the newly formed symbi
[3] "A working-class Italian-American bouncer becomes the driver of
[4] "At a top secret research facility in the 1960s, a lonely janit
[5] "A young African-American man grapples with his identity and se
[6] "The true story of how the Boston Globe uncovered the massive s
```

These are the **descriptions** of the movies in the IMDb webpage.

- Parse the webpage to produce a vector of the descriptions.
- Tidy the description by removing unwanted regexes.

03:00

Runtime

```
runtime_data <- html_nodes(webpage,'.text-muted .runtime') %>%
  html_text() %>%
  str_replace_all(" min", "") %>%
  as.numeric()
```

```
runtime_data
 [1] 107 132 130 123 111 129 119 134 120 100 118 120 131 122 151 1
[20] 135 155 122 123 194 162 178 142 195 130 118 181  99 133 163 1
[39] 191 125 124 105 183  93 120 133 202 129 175 104 172 113 153 1
[58] 129 228 153 125 212 115 161 175  90 108 118 152 114 138 110 1
[77] 126 134 102 118 130 238 126 116 176 132 105 112 112 123 152 1
```

Ratings

```
rating_data <- html_nodes(webpage,'.ratings-imdb-rating strong') %>
  html_text() %>%
  as.numeric()
```

```
rating_data
 [1] 7.3 8.6 8.2 7.3 7.4 8.1 7.7 8.1 7.7 7.9 8.0 8.0 7.5 8.1 8.5 8
[20] 8.2 8.5 8.3 7.1 7.8 7.4 8.3 8.8 8.9 8.2 8.6 8.0 7.4 8.0 7.7 8
[39] 8.0 7.2 7.7 7.8 8.1 8.0 8.1 8.7 9.0 8.3 9.2 7.7 7.9 7.8 7.4 7
[58] 6.5 8.3 7.5 8.3 8.1 6.7 8.1 6.8 7.7 8.1 7.6 6.6 7.2 8.2 7.5 7
[77] 7.1 7.6 8.5 7.7 8.1 8.1 7.9 7.2 6.7 7.7 8.1 5.9 7.4 5.9 8.1 5
```

Number of votes

```
votes_data <- html_nodes(webpage,'.sort-num_votes-visible span:nth-
  html_text() %>%
  str_replace_all(",", "") %>%
  as.numeric()
```

```
votes_data
 [1]  144744  716484  449985  403404  295812  453354  611825  6778
[10]  238235  665574  827775  440902  926869 1269327  668572  4308
[19]  225610  899522 1434403 1121935  221537 1120220  186670 10064
[28] 1298447  399478 1366121  255426  107138  498095  100384  4021
[37]  389403   58984  225679   58609   50650  141635  330291  2608
[46]  972513 1213573  254801 1749842  118854  100105  108357   374
[55]   33858  221924   93202   12739  284917  108530  175965  2326
```

Combine all

Show 10 entries Search:

	Year	Title	Description	Runtime	Rating	Votes
1	2020	Nomadland	A woman in her sixties, after losing everything in the Great Recession, embarks on a journey through the American West, living as a van-dwelling modern-day nomad.	107	7.3	144744
2	2019	Parasite	Greed and class discrimination threaten the newly formed symbiotic relationship between the wealthy Park family and the destitute Kim clan.	132	8.6	716484
3	2018	Green Book	A working-class Italian-American bouncer becomes the driver of an African-American classical pianist on a tour of venues through the 1960s American South.	130	8.2	449985

Original webpage

Your Turn 4

- Scrape the names, scores, and years of most popular TV shows on IMDB: www.imdb.com/chart/tvmeter
- Create a data frame called `tvshows` with four variables: `rank`, `name`, `score`, `year`

rank	name	score	year
1	Pam & Tommy	7.6	2022
2	The Book of Boba Fett	7.7	2021
3	The Woman in the House	6.4	2022
4	Euphoria	8.4	2019
5	Ozark	8.5	2017
6	All of Us Are Dead	7.6	2022
7	Attack on Titan	9.0	2013
8	Reacher	8.5	2022

05:00