STAT 118: Notes P

Webscraping Text with `rvest`

Emily Malcolm-White

```
#LOAD PACKAGES
library(tidyverse)
library(rvest)
```

Webscraping Text

Let's look at the top 50 feature films in the first 7 months of 2023 listed on IMBD

```
URL <- read_html("https://www.imdb.com/search/title/?title_type=feature&year=2023-01-01,20
```

Notice that the data for all these films isn't housed inside a `<table>` element!

Titles

For example, check out the first few lines of html code for Oppenheimer:

```
<h3 class="lister-item-header">
        <span class="lister-item-index unbold text-primary">1.</span>
    <a href="/title/tt15398776/?ref_=adv_li_tt"
>Oppenheimer</a>
    <span class="lister-item-year text-muted unbold">(2023)</span>
</h3>
```

In this case, we want to look for the class `lister-item-header` AND then pull the text inside the `<a>` (link) tag.

`html_elements(".lister-item-header a")`

> **Tip**
>
> In this case, we want ALL titles so we used `html_elements()`. If we had only wanted the first title we would have used `html_element()`

Scrape IMBD for the titles of the 50 most popular feature films in the first 7 months of 2023.

```
title_data <- URL %>%
  html_elements(".lister-item-header a") %>%
  html_text()

title_data
```

```
 [1] "Oppenheimer"
 [2] "Barbie"
 [3] "No Hard Feelings"
 [4] "Guardians of the Galaxy Vol. 3"
 [5] "Meg 2: The Trench"
 [6] "Gran Turismo"
 [7] "Spider-Man: Across the Spider-Verse"
 [8] "The Pope's Exorcist"
 [9] "Teenage Mutant Ninja Turtles: Mutant Mayhem"
[10] "Dungeons & Dragons: Honor Among Thieves"
[11] "The Flash"
[12] "Asteroid City"
[13] "Mission: Impossible - Dead Reckoning Part One"
[14] "Elemental"
[15] "Bottoms"
[16] "Sound of Freedom"
[17] "The Super Mario Bros. Movie"
[18] "Past Lives"
[19] "Cobweb"
[20] "Cocaine Bear"
[21] "To Catch a Killer"
[22] "Transformers: Rise of the Beasts"
[23] "John Wick: Chapter 4"
[24] "The Little Mermaid"
[25] "The Covenant"
```

```
[26] "The River Wild"
[27] "Haunted Mansion"
[28] "Golda"
[29] "Fast X"
[30] "Indiana Jones and the Dial of Destiny"
[31] "Haunting of the Queen Mary"
[32] "Killers of the Flower Moon"
[33] "Cassandro"
[34] "Happiness for Beginners"
[35] "The Three Musketeers: D'Artagnan"
[36] "They Cloned Tyrone"
[37] "My Fault"
[38] "Paradise"
[39] "Insidious: The Red Door"
[40] "Joy Ride"
[41] "Knock at the Cabin"
[42] "Kandahar"
[43] "Satyaprem Ki Katha"
[44] "Evil Dead Rise"
[45] "Extraction II"
[46] "Hidden Strike"
[47] "Renfield"
[48] "The Out-Laws"
[49] "The Black Demon"
[50] "Beau Is Afraid"
```

Runtime

Scrape IMBD for the runtime of the 50 most popular feature films so far in 2023.

Check out the relevant HTML code for Oppenheimer:

```
    <p class="text-muted ">
            <span class="certificate">R</span>
                 <span class="ghost">|</span>
                 <span class="runtime">180 min</span>
                 <span class="ghost">|</span>
            <span class="genre">
Biography, Drama, History            </span>
    </p>
```

In this case, we need to reference the class `text-muted` AND the class `runtime`.

```
URL %>%
  html_nodes(".text-muted .runtime") %>%
  html_text()
```

```
 [1] "180 min" "114 min" "103 min" "150 min" "116 min" "134 min" "140 min"
 [8] "103 min" "99 min"  "134 min" "144 min" "105 min" "163 min" "101 min"
[15] "92 min"  "131 min" "92 min"  "105 min" "88 min"  "95 min"  "119 min"
[22] "127 min" "169 min" "135 min" "123 min" "91 min"  "123 min" "100 min"
[29] "141 min" "154 min" "114 min" "206 min" "107 min" "103 min" "121 min"
[36] "122 min" "117 min" "117 min" "107 min" "95 min"  "100 min" "119 min"
[43] "146 min" "96 min"  "122 min" "102 min" "93 min"  "95 min"  "100 min"
[50] "179 min"
```

Alternatively, we could have called class `text-muted` AND the 3rd span, but it's easier and likely more accurate to ask for the class `runtime` in case `runtime` is missing for some reason.

Maybe we want to keep the `min` on the end, but it forces it into being a stringr rather than a number which makes it difficult to sort or filter.

```
library(readr)
# need this package for parse_number()
```

Figure 1: Artwork by @allisonhorst

```
runtime_data <- URL %>%
  html_nodes(".text-muted .runtime") %>%
  html_text() %>%
  parse_number() %>% #this picks out only the numbers (and drops characters, in this case,
  as.numeric()

runtime_data
```

```
 [1] 180 114 103 150 116 134 140 103  99 134 144 105 163 101  92 131  92 105  88
[20]  95 119 127 169 135 123  91 123 100 141 154 114 206 107 103 121 122 117 117
[39] 107  95 100 119 146  96 122 102  93  95 100 179
```

Ratings

Scrape IMBD for the ratings of the 50 most popular feature films in the first 7 months of 2023.

Check out the relevant HTML code for Oppenheimer:

```
<div class="inline-block ratings-imdb-rating" name="ir" data-value="8.6">
    <span class="global-sprite rating-star imdb-rating"></span>
    <strong>8.6</strong>
</div>
```

Let's scrape it!

```
rating_data <- URL %>%
  html_elements(".ratings-imdb-rating strong") %>%
  html_text() %>%
  as.numeric()

rating_data
```

```
 [1] 8.6 7.4 6.5 8.0 5.3 7.4 8.8 6.1 7.5 7.3 6.8 6.7 8.0 7.0 6.9 7.8 7.1 8.2 5.9
[20] 5.9 6.6 6.1 7.8 7.2 7.5 5.5 6.2 6.4 5.8 6.8 4.4 5.8 6.0 6.8 6.7 6.2 6.3 5.6
[39] 6.5 6.1 6.0 7.3 6.6 7.0 5.3 6.4 5.4 3.7 6.8
```

> **Warning**
>
> Notice that there are only 49 ratings listed, not 50! There is no way to figure out which one is missing besides doing it by hand...
> Which one is it?
> Once we figure out which one is it is, we should should add a blank element for the rating for that movie using the `append` function.
> `rating_data <- append(rating_data, values=FALSE, after=11)`

It's Killers of the Flower Moon (#32)!

```
rating_data <- append(rating_data, values=NA, after=31)
```

Notice how it is the correct length (50) now!

Number of Votes

Scrape IMBD for the number of votes of the 50 most popular feature films in the first 7 months of 2023.

Relevant code for Oppenheimer:

```
        <p class="sort-num_votes-visible">
                <span class="text-muted">Votes:</span>
                <span name="nv" data-value="391689">391,689</span>
        </p>
```

Let's scrape it!

```
votes_data <- URL %>%
  html_elements(".sort-num_votes-visible span:nth-child(2)") %>%
  html_text() %>%
  parse_number() %>%
  as.numeric()

votes_data
```

```
 [1] 391689 261950  54644 299228  27214  13944 241099  56813  20320 172351
[11] 151313  66120 132397  43568    898  50049 185103  12376  10015  87113
[21]  21892  75715 266711 112420 110526   3190  11231    789  90216  95616
[31]    674    160  11383  10443  26390  16486  12223  31097  12634  92680
[41]  19621  22297 108958 121482  13285  66278  28541   5268  38593
```

> **Warning**
>
> Same issue as before! We were supposed to have 50 but only got 49. It's Killers of the Flower Moon (#32), again!
>
> ```
> votes_data <- append(votes_data, values=NA, after=31)
> ```

Metascore

Scrape IMBD for the number of votes of the 50 most popular feature films in the first 7 months of 2023.

Relevant code for Oppenheimer:

```
            <div class="inline-block ratings-metascore">
<span class="metascore  favorable">88        </span>
        Metascore
            </div>
```

Let's scrape it!

```
metascore_data <- URL %>%
  html_elements(".metascore") %>%
  html_text() %>%
  parse_number() %>%
  as.numeric()

metascore_data
```

```
 [1] 88 80 59 64 40 46 86 45 74 72 55 74 81 58 78 43 46 94 50 54 43 42 78 59 63
[26] 47 49 56 58 81 48 74 45 74 63 52 69 57 53 36 63
```

> **Warning**
>
> Yikes! Now we only have 41 when we should have 50.
> We *could* manually go through and figure out which 9 are missing or we could reassess how important the metascore data is to us...

Combining it all together into a data frame!

We can combine all this data into one data frame:

```
movies <- data.frame(Title = title_data,
Runtime = runtime_data,
Rating = rating_data,
Votes = votes_data
)

movies
```

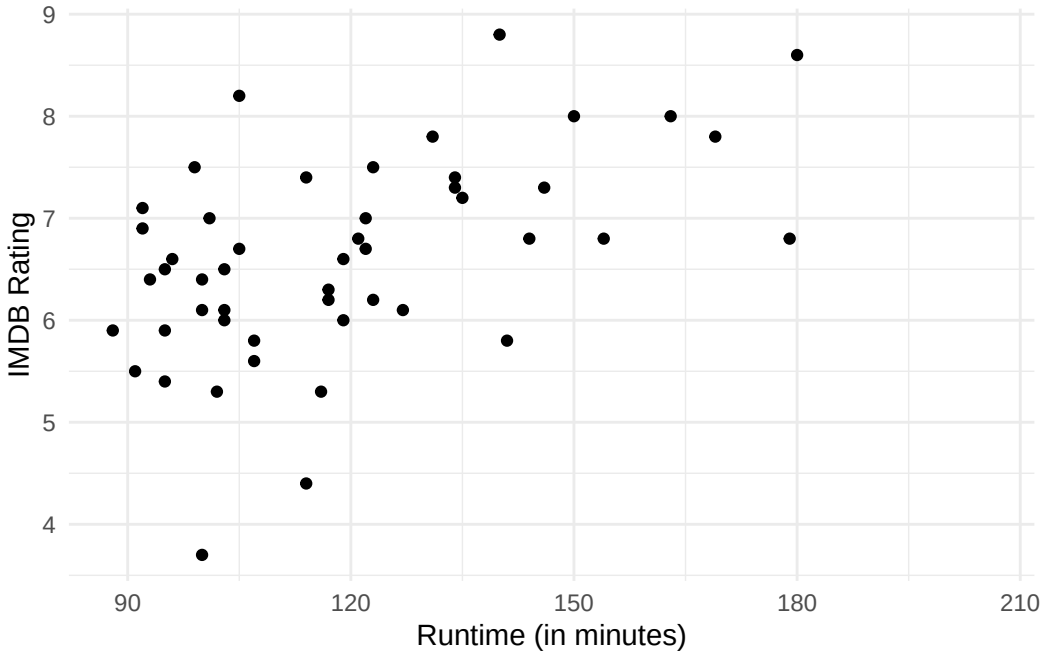
	Title	Runtime	Rating	Votes
1	Oppenheimer	180	8.6	391689
2	Barbie	114	7.4	261950
3	No Hard Feelings	103	6.5	54644
4	Guardians of the Galaxy Vol. 3	150	8.0	299228
5	Meg 2: The Trench	116	5.3	27214
6	Gran Turismo	134	7.4	13944
7	Spider-Man: Across the Spider-Verse	140	8.8	241099
8	The Pope's Exorcist	103	6.1	56813
9	Teenage Mutant Ninja Turtles: Mutant Mayhem	99	7.5	20320
10	Dungeons & Dragons: Honor Among Thieves	134	7.3	172351
11	The Flash	144	6.8	151313
12	Asteroid City	105	6.7	66120
13	Mission: Impossible - Dead Reckoning Part One	163	8.0	132397
14	Elemental	101	7.0	43568
15	Bottoms	92	6.9	898
16	Sound of Freedom	131	7.8	50049
17	The Super Mario Bros. Movie	92	7.1	185103
18	Past Lives	105	8.2	12376
19	Cobweb	88	5.9	10015
20	Cocaine Bear	95	5.9	87113
21	To Catch a Killer	119	6.6	21892
22	Transformers: Rise of the Beasts	127	6.1	75715
23	John Wick: Chapter 4	169	7.8	266711
24	The Little Mermaid	135	7.2	112420
25	The Covenant	123	7.5	110526
26	The River Wild	91	5.5	3190
27	Haunted Mansion	123	6.2	11231
28	Golda	100	6.4	789
29	Fast X	141	5.8	90216

30	Indiana Jones and the Dial of Destiny	154	6.8	95616
31	Haunting of the Queen Mary	114	4.4	674
32	Killers of the Flower Moon	206	NA	NA
33	Cassandro	107	5.8	160
34	Happiness for Beginners	103	6.0	11383
35	The Three Musketeers: D'Artagnan	121	6.8	10443
36	They Cloned Tyrone	122	6.7	26390
37	My Fault	117	6.2	16486
38	Paradise	117	6.3	12223
39	Insidious: The Red Door	107	5.6	31097
40	Joy Ride	95	6.5	12634
41	Knock at the Cabin	100	6.1	92680
42	Kandahar	119	6.0	19621
43	Satyaprem Ki Katha	146	7.3	22297
44	Evil Dead Rise	96	6.6	108958
45	Extraction II	122	7.0	121482
46	Hidden Strike	102	5.3	13285
47	Renfield	93	6.4	66278
48	The Out-Laws	95	5.4	28541
49	The Black Demon	100	3.7	5268
50	Beau Is Afraid	179	6.8	38593

Make a list OR Make a plot!

```
ggplot(movies, aes(x=runtime_data, y=rating_data)) +
  geom_point() +
  theme_minimal() +
  xlab("Runtime (in minutes)") +
  ylab("IMDB Rating")
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
```


IMDB Rating
9
8
7
6
5
4
90
120
150
180
210
Runtime (in minutes)