Data Analysis with R

Webscraping and parsing data in R

2019-02-26

Requirement

- Intermediate knowledge in R,
- Need to install both R and the RStudio interface,
- Need to install the `tidyverse`,`rvest`

```r
# install.packages(c("tidyverse","rvest","ggrepel")
library(tidyverse)
library(rvest)
library(ggrepel)
```

What we want

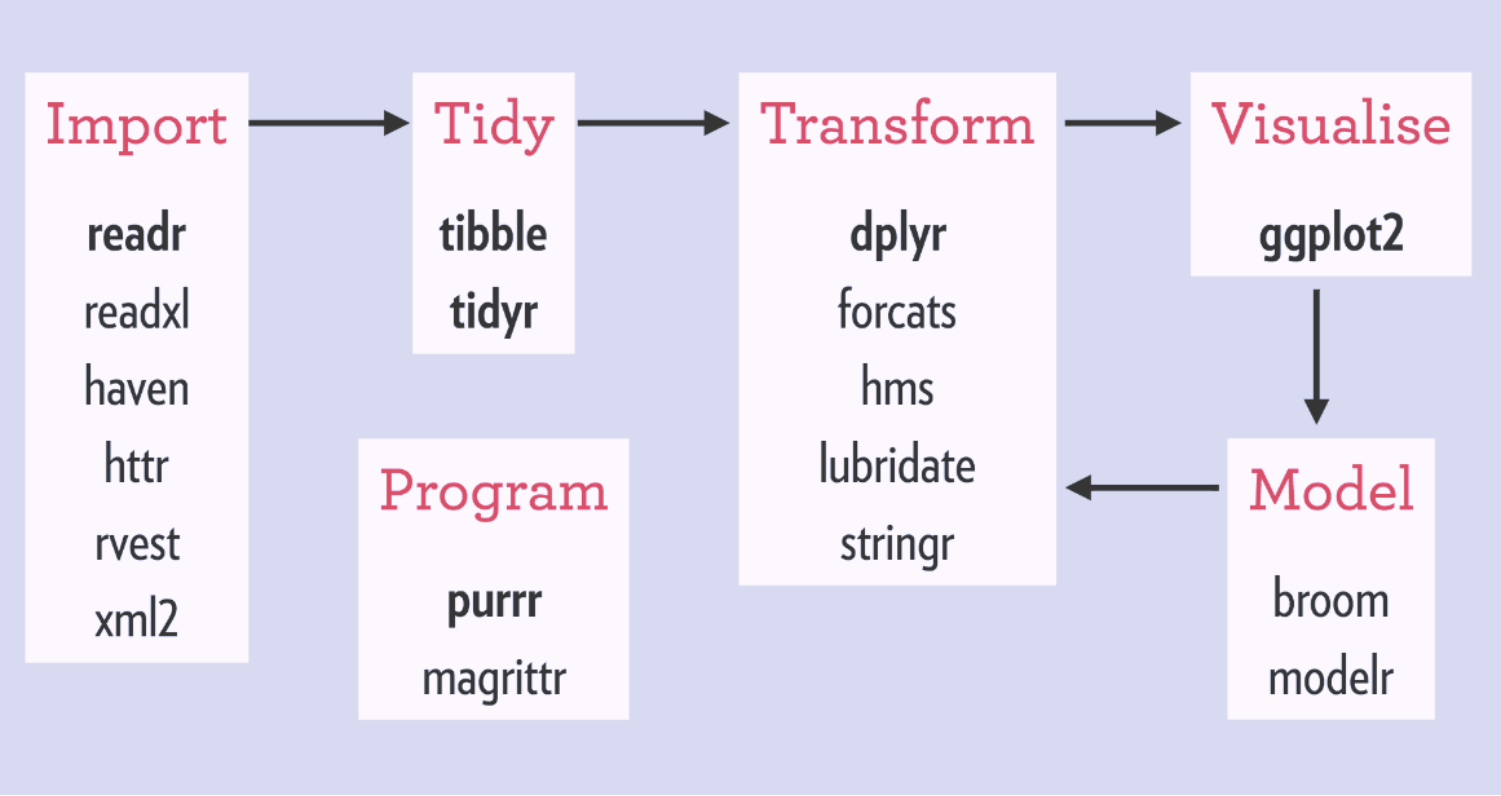
What we get

> *80 percent of a data scientist's valuable time is spent simply finding, cleansing, and organizing data, leaving only 20 percent to actually perform analysis...*
>
> IBM Data Analytics

What is tidyverse?

- a list of many R packages, including ggplot2, dplyr, tidyr, readr and stringr etc.
- a coherent system of packages for data manipulation, exploration and visualization
- In statistics, we typically assume data is "tidy"
  - data is in a tabular form
  - 1 row == 1 observation
  - 1 column == 1 variable
- Parsing HTML/XML/JSON is easy; but putting it into a tidy form is typically not easy

Tidyverse process



What is a pipe operator in Tidyverse?

- `f(x)` can be rewritten as `x %>% f`
- `f(x, y)` can be rewritten as `x %>% f(y)`
- suppose we want to compute the mean of the logarithm of x and round it to 1 decimal
- x is a vector of numbers

```
x <- c(0.109, 0.359, 0.63)
# have to nest a lot of parentheses together
round(mean(log(x)), 1)
```

```
## [1] -1.2
```

```
# pipe operator makes it cleaner!
x %>%
  log() %>%
  mean() %>%
  round(1)
```

```
## [1] -1.2
```

General procedure for web scraping

- Inspect element in your browser
- Tell R where to "look" on the page
- Get the data in R with a few trial and error attempts
- Manipulate the data into a tidy way if possible

`rvest` library

- A R library which allows us to scrape data from web pages easily
- Some basic functions in `rvest`:
  - `html_nodes()`: identifies HTML wrappers
  - `html_table()`: turns HTML tables into data frames
  - `html_text()`: strips the HTML tags and extracts only the text
  - `html_nodes(".class")`: calls node based on css class
  - `html_nodes("#id")`: calls node based on `<div>` id
  - `html_nodes(xpath="xpath")`: calls node based on xpath
  - `html_attrs()`: identifies attributes (useful for debugging)

NBA example

- If we want to scrape the 2018-19 NBA schedule in October from basketball-reference.com (ignore the fact that you can download it directly)

2018-19 NBA Season | Standings | Schedule and Results | Leaders | Coaches | Player Stats ▾ | Other ▾

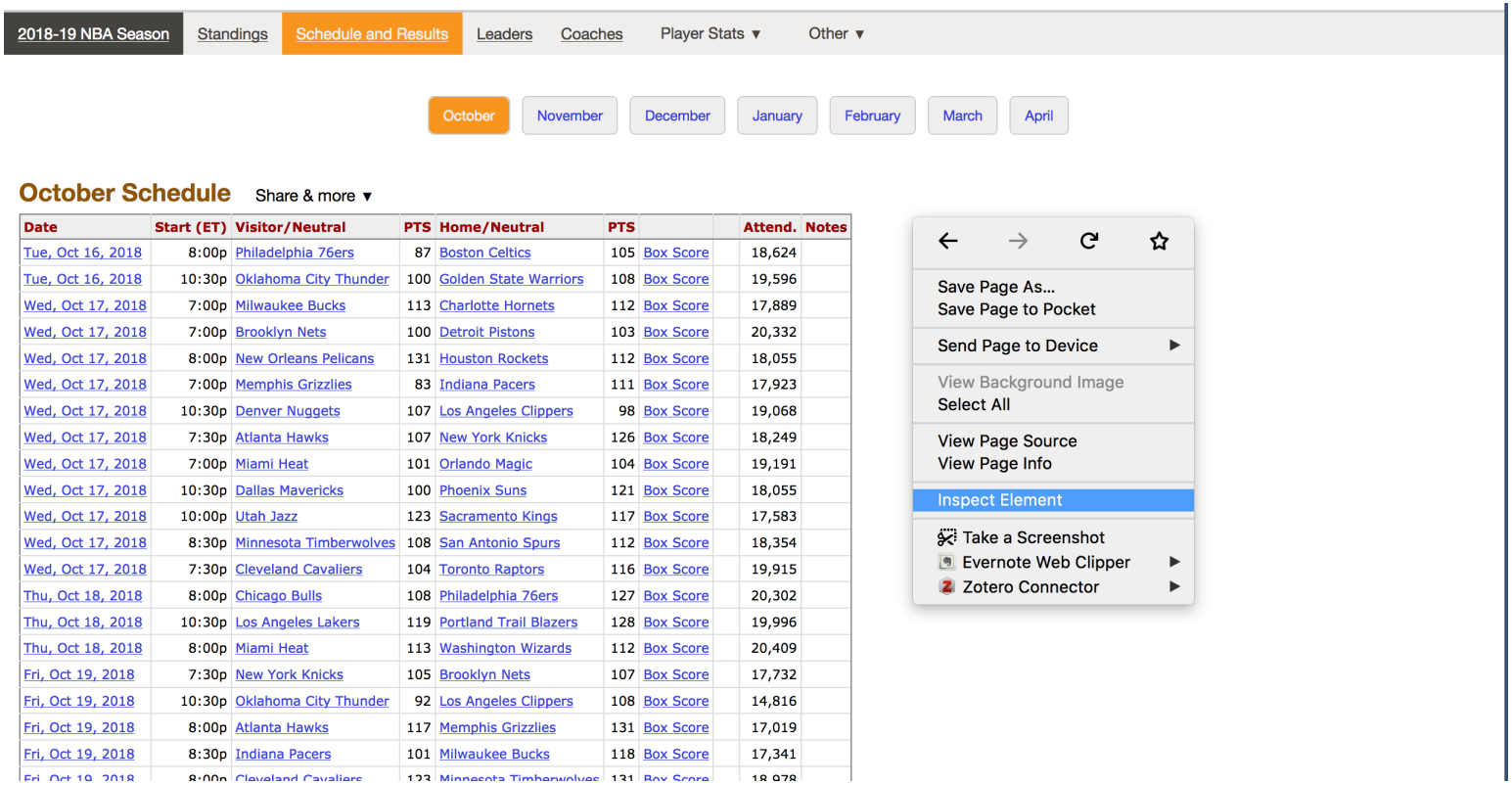
October Schedule

Share & more ▾

Date	Start (ET)	Visitor/Neutral	PTS	Home/Neutral	PTS			Attend.	Notes
Tue, Oct 16, 2018	8:00p	Philadelphia 76ers	87	Boston Celtics	105	Box Score		18,624	
Tue, Oct 16, 2018	10:30p	Oklahoma City Thunder	100	Golden State Warriors	108	Box Score		19,596	
Wed, Oct 17, 2018	7:00p	Milwaukee Bucks	113	Charlotte Hornets	112	Box Score		17,889	
Wed, Oct 17, 2018	7:00p	Brooklyn Nets	100	Detroit Pistons	103	Box Score		20,332	
Wed, Oct 17, 2018	8:00p	New Orleans Pelicans	131	Houston Rockets	112	Box Score		18,055	
Wed, Oct 17, 2018	7:00p	Memphis Grizzlies	83	Indiana Pacers	111	Box Score		17,923	
Wed, Oct 17, 2018	10:30p	Denver Nuggets	107	Los Angeles Clippers	98	Box Score		19,068	
Wed, Oct 17, 2018	7:30p	Atlanta Hawks	107	New York Knicks	126	Box Score		18,249	
Wed, Oct 17, 2018	7:00p	Miami Heat	101	Orlando Magic	104	Box Score		19,191	
Wed, Oct 17, 2018	10:30p	Dallas Mavericks	100	Phoenix Suns	121	Box Score		18,055	
Wed, Oct 17, 2018	10:00p	Utah Jazz	123	Sacramento Kings	117	Box Score		17,583	

NBA example

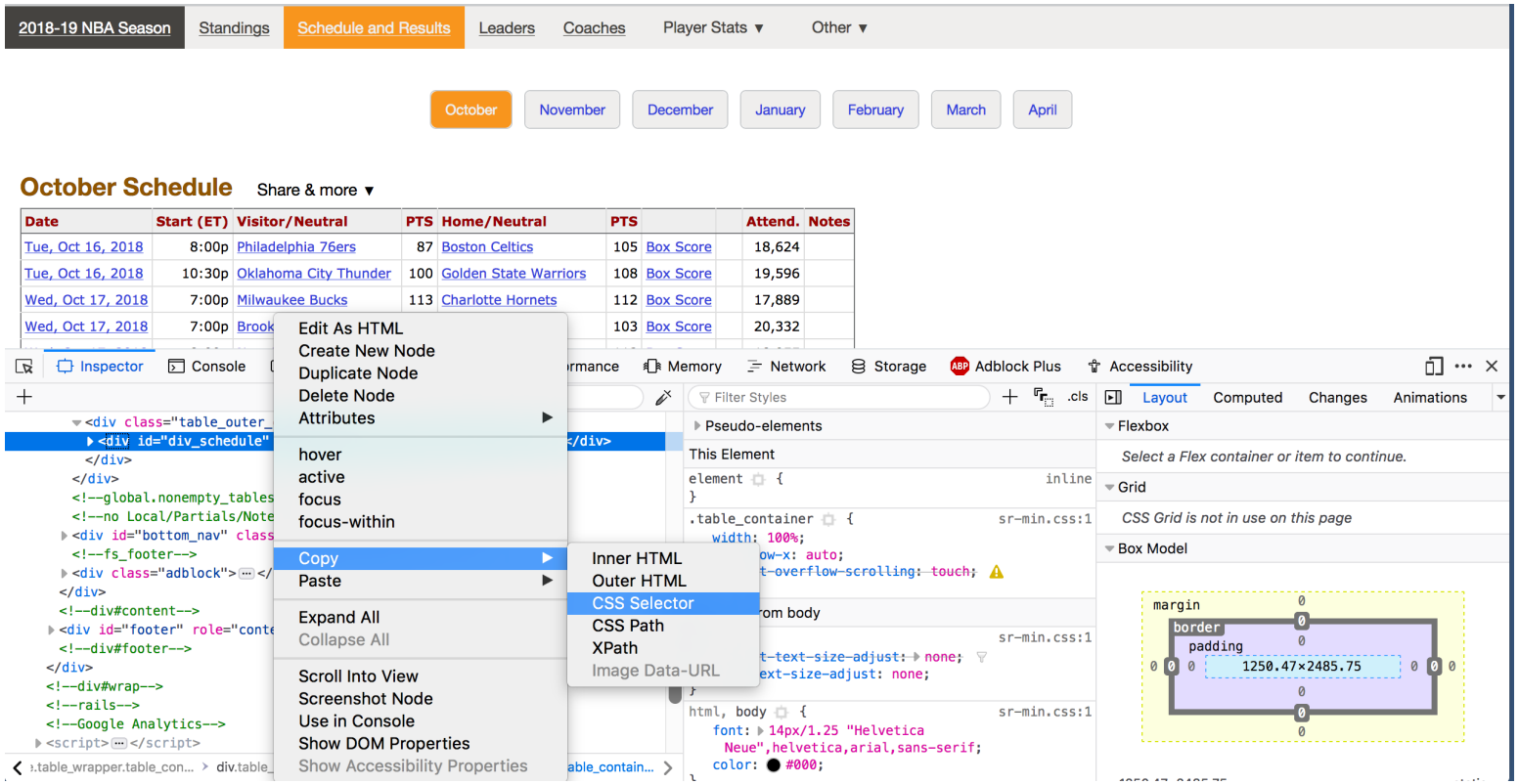
- Open the link in your browser
- Locate your table/content by clicking on “inspect element”



Date	Start (ET)	Visitor/Neutral	PTS	Home/Neutral	PTS		Attend.	Notes
Tue, Oct 16, 2018	8:00p	Philadelphia 76ers	87	Boston Celtics	105	Box Score	18,624	
Tue, Oct 16, 2018	10:30p	Oklahoma City Thunder	100	Golden State Warriors	108	Box Score	19,596	
Wed, Oct 17, 2018	7:00p	Milwaukee Bucks	113	Charlotte Hornets	112	Box Score	17,889	
Wed, Oct 17, 2018	7:00p	Brooklyn Nets	100	Detroit Pistons	103	Box Score	20,332	
Wed, Oct 17, 2018	8:00p	New Orleans Pelicans	131	Houston Rockets	112	Box Score	18,055	
Wed, Oct 17, 2018	7:00p	Memphis Grizzlies	83	Indiana Pacers	111	Box Score	17,923	
Wed, Oct 17, 2018	10:30p	Denver Nuggets	107	Los Angeles Clippers	98	Box Score	19,068	
Wed, Oct 17, 2018	7:30p	Atlanta Hawks	107	New York Knicks	126	Box Score	18,249	
Wed, Oct 17, 2018	7:00p	Miami Heat	101	Orlando Magic	104	Box Score	19,191	
Wed, Oct 17, 2018	10:30p	Dallas Mavericks	100	Phoenix Suns	121	Box Score	18,055	
Wed, Oct 17, 2018	10:00p	Utah Jazz	123	Sacramento Kings	117	Box Score	17,583	
Wed, Oct 17, 2018	8:30p	Minnesota Timberwolves	108	San Antonio Spurs	112	Box Score	18,354	
Wed, Oct 17, 2018	7:30p	Cleveland Cavaliers	104	Toronto Raptors	116	Box Score	19,915	
Thu, Oct 18, 2018	8:00p	Chicago Bulls	108	Philadelphia 76ers	127	Box Score	20,302	
Thu, Oct 18, 2018	10:30p	Los Angeles Lakers	119	Portland Trail Blazers	128	Box Score	19,996	
Thu, Oct 18, 2018	8:00p	Miami Heat	113	Washington Wizards	112	Box Score	20,409	
Fri, Oct 19, 2018	7:30p	New York Knicks	105	Brooklyn Nets	107	Box Score	17,732	
Fri, Oct 19, 2018	10:30p	Oklahoma City Thunder	92	Los Angeles Clippers	108	Box Score	14,816	
Fri, Oct 19, 2018	8:00p	Atlanta Hawks	117	Memphis Grizzlies	131	Box Score	17,019	
Fri, Oct 19, 2018	8:30p	Indiana Pacers	101	Milwaukee Bucks	118	Box Score	17,341	
Fri, Oct 19, 2018	8:00p	Cleveland Cavaliers	123	Minnesota Timberwolves	131	Box Score	18,978	

Motivating example

- Find the line of HTML which corresponds to the table
- Copy the CSS selectors and put it into `html_nodes()`



1st attempt with `html_text()`

- `html_text()` will extract the content, but it's super messy
- we would like to perserve the tabular format as seen on the page

```
url = "https://www.basketball-reference.com/leagues/NBA_2019_games.html"
url %>%
  read_html() %>%
  html_nodes("#schedule") %>%
  html_text()
```

```
## [1] "October Schedule Table\n   Date\n         Start (ET)\n         Visitor/
```

2nd attempt with `html_table()`

- `html_table()` is able to parse a table better than `html_text()`
- We often need to clean up some of the content after webscraping

```
bball = url %>%
  read_html() %>%
  html_nodes("#schedule") %>%
  html_table() %>%
  .[[1]]

names(bball) = c("date", "start_time", "visitor", "vistor_pts",
            "home", "home_pts", "col1", "col2", "attend", "Notes")
bball <- bball %>% dplyr::select(-col1, -col2, -Notes)
head(bball, 3)
```

```
##                date start_time               visitor vistor_pts
## 1 Tue, Oct 16, 2018      8:00p    Philadelphia 76ers         87
## 2 Tue, Oct 16, 2018     10:30p Oklahoma City Thunder        100
## 3 Wed, Oct 17, 2018      7:00p       Milwaukee Bucks        113
##                    home home_pts attend
## 1        Boston Celtics      105 18,624
## 2 Golden State Warriors      108 19,596
## 3     Charlotte Hornets      112 17,889
```

Analysis

- Once we have the data ready, we can start our analysis
- Eg. Which team has the highest home attendence in NBA?
- `dplyr` package provides some very useful functions for performing data analysis tasks
- Some basic functions in `dplyr`:
  - `select()`: select columns
  - `filter()`: filter rows
  - `arrange()`: re-order or arrange rows
  - `mutate()`: create new columns
  - `summarise()`: summarise values
  - `group_by()`: allows for group operations in the "split-apply-combine" concept

Analysis

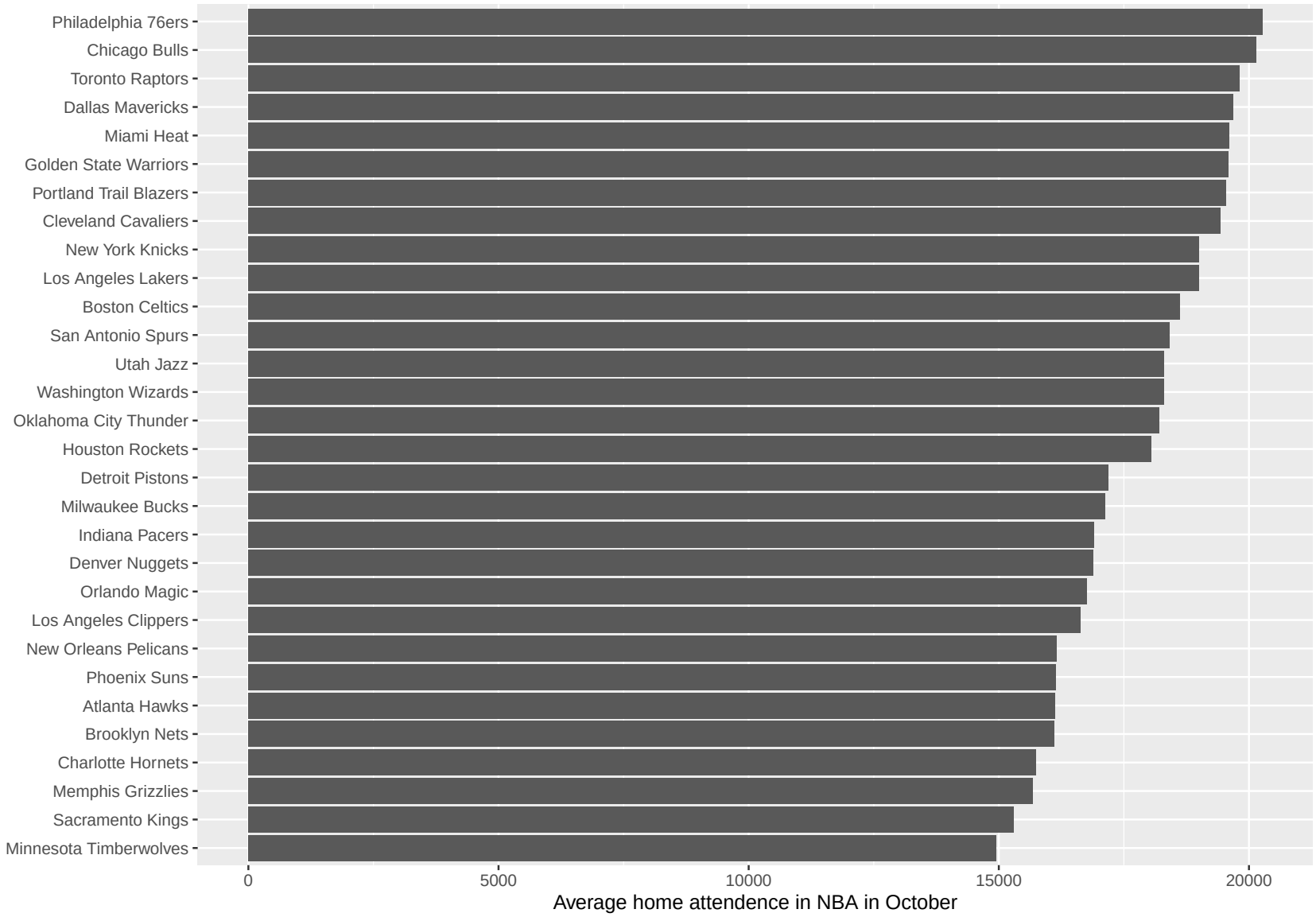
```r
attendAvg = bball %>%
  mutate(attend = as.numeric(str_replace_all(attend,",",""))) %>%
  group_by(home) %>%
  summarise(no_of_games = n(),
            avg_attend = mean(attend)) %>%
  arrange(desc(avg_attend))
attendAvg
```

```
## # A tibble: 30 x 3
##    home                    no_of_games avg_attend
##    <chr>                         <int>      <dbl>
##  1 Philadelphia 76ers                4     20268.
##  2 Chicago Bulls                     4     20140.
##  3 Toronto Raptors                   6     19819.
##  4 Dallas Mavericks                  3     19689
##  5 Miami Heat                        4     19600
##  6 Golden State Warriors             4     19596
##  7 Portland Trail Blazers            3     19548
##  8 Cleveland Cavaliers               4     19432
##  9 New York Knicks                   5     19001.
## 10 Los Angeles Lakers                4     18997
## # ... with 20 more rows
```

- Can we visulize it?

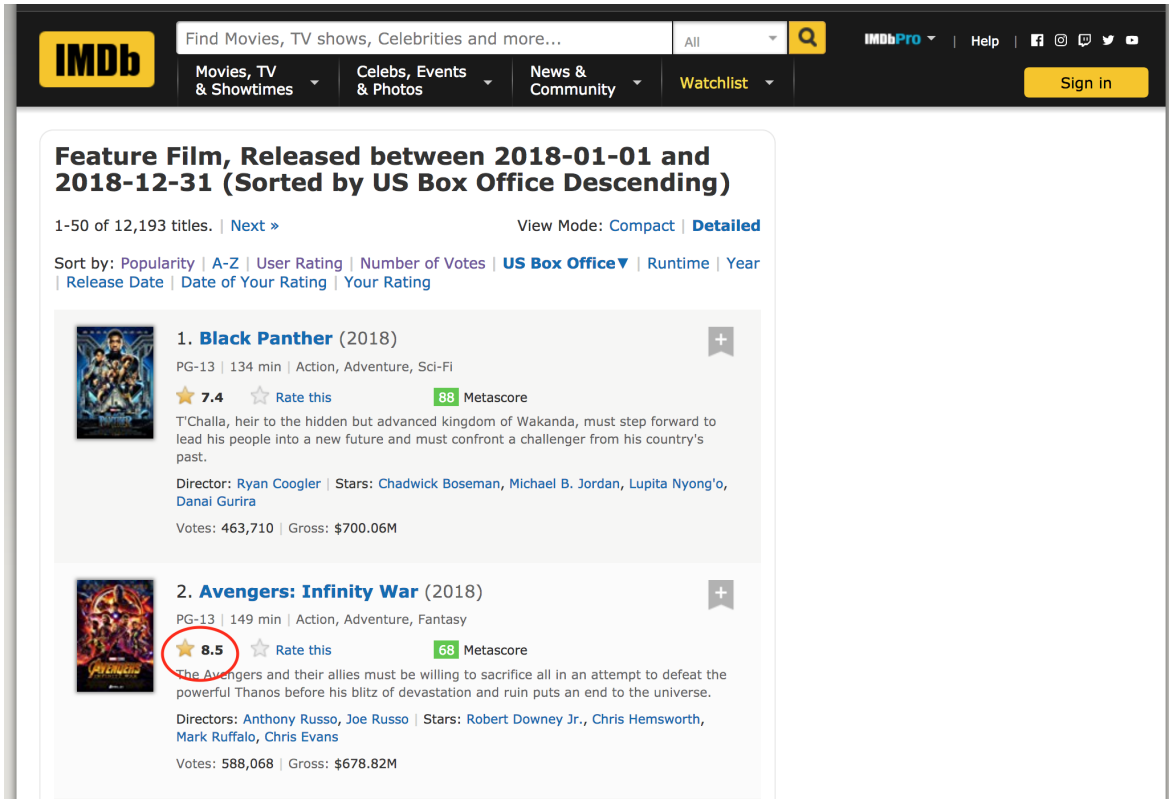
Visulization in ggplot2

```
attendAvg %>% ggplot(aes(x = reorder(home, avg_attend), y = avg_attend)) +
geom_col() + coord_flip() + labs(x = NULL, y = "Average home attendence in NBA i
```



Exercise

- This link shows top 50 movies sorted by US Box Office revenue on IMDb website
- Can you find the CSS selector of the rating of Avengers: Infinity War (2018)?
- Can you extract that rating in R?



Exercise - solution

```
url2 = "https://www.imdb.com/search/title?title_type=feature&release_date=2018-
url2 %>%
  read_html() %>%
  html_nodes("div.lister-item:nth-child(2) > div:nth-child(3)
             > div:nth-child(3) > div:nth-child(1)") %>%
  html_text() %>%
  as.numeric()
```

```
## [1] 8.5
```

IMDB example

- Question: Does a high rated movie correspond to a high revenue? (Note: this is not a random sampling)
- Let's say, we want to scrape a list of variables including title, rating, gross revenue and movie length of these 50 movies
- Sometimes, we cannot scrape all the info/variables at once

```
url2 %>%
  read_html() %>%
  html_nodes("div.lister-item:nth-child(1) > div:nth-child(3)") %>%
  html_text()
```

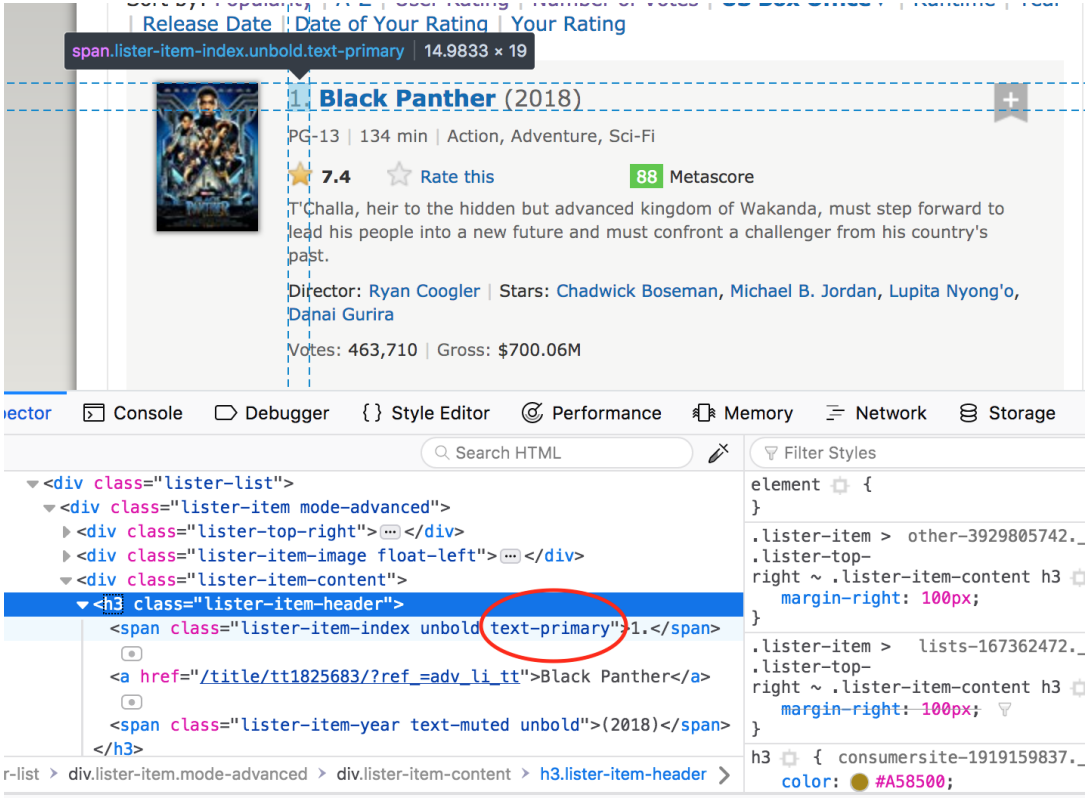
```
## [1] "\n\n        1.\n    \n    Black Panther\n    (2018)\n\n    \n
```

```
url2 %>%
  read_html() %>%
  html_nodes("div.lister-item:nth-child(1) > div:nth-child(3)") %>%
  html_table()
```

```
## Error in html_table.xml_node(X[[i]], ...): html_name(x) == "table" is not TR
```

IMDB example

- If our target is not formatted in a HTML table form nicely, we have to scrape each part separately
- Eg. let's scrape the ranking, rating and title of the movie first
- Ranking of these 50 movies are defined in the same class



IMDB example

- `.text-primary` will return the ranking of all 50 movies
- Similarly, we can get rating and title by locating their class respectively

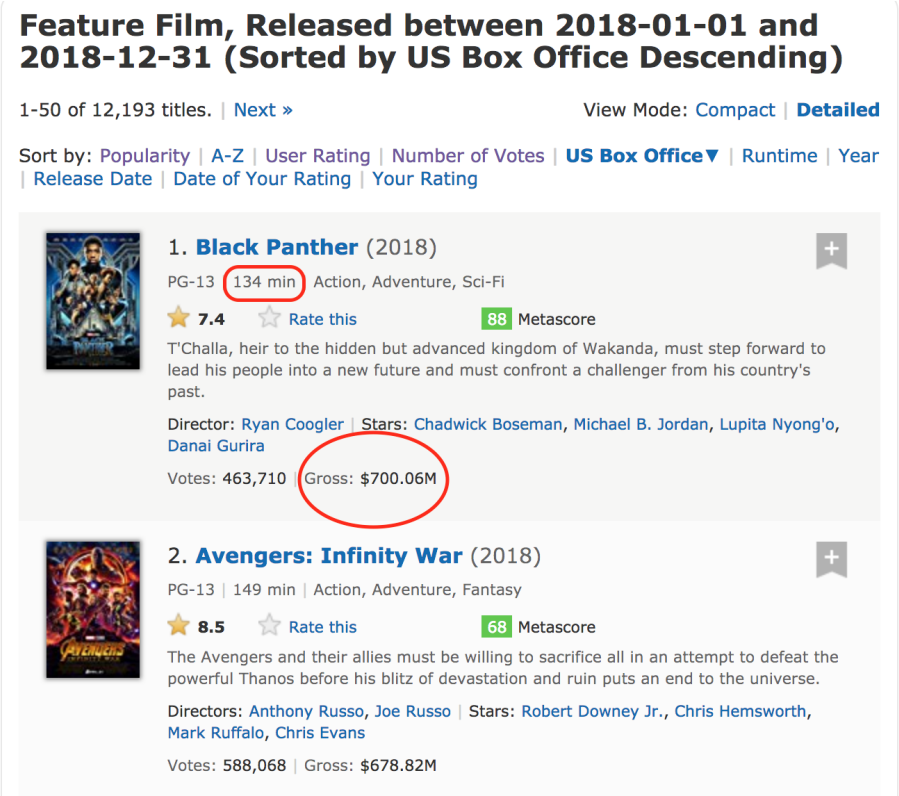
```
ranking = url2 %>% read_html() %>% html_nodes(".text-primary") %>%
  html_text()

rating = url2 %>% read_html() %>% html_nodes('.ratings-imdb-rating strong') %>%
  html_text()

title = url2 %>% read_html() %>% html_nodes('.lister-item-header a') %>%
  html_text()
```

Exercise

- Scrape revenue and movie length from the page



Feature Film, Released between 2018-01-01 and 2018-12-31 (Sorted by US Box Office Descending)

1-50 of 12,193 titles. | Next » 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. **Black Panther** (2018)

PG-13 | 134 min | Action, Adventure, Sci-Fi

7.4 Rate this 88 Metascore

T'Challa, heir to the hidden but advanced kingdom of Wakanda, must step forward to lead his people into a new future and must confront a challenger from his country's past.

Director: Ryan Coogler | Stars: Chadwick Boseman, Michael B. Jordan, Lupita Nyong'o, Danai Gurira

Votes: 463,710 | Gross: $700.06M

2. **Avengers: Infinity War** (2018)

PG-13 | 149 min | Action, Adventure, Fantasy

8.5 Rate this 68 Metascore

The Avengers and their allies must be willing to sacrifice all in an attempt to defeat the powerful Thanos before his blitz of devastation and ruin puts an end to the universe.

Directors: Anthony Russo, Joe Russo | Stars: Robert Downey Jr., Chris Hemsworth, Mark Ruffalo, Chris Evans

Votes: 588,068 | Gross: $678.82M

Exercise - solution

```r
revenue = url2 %>% read_html() %>% html_nodes('.ghost~ .text-muted+ span') %>%
  html_text()

# movie length
runtime = url2 %>% read_html() %>% html_nodes('.text-muted .runtime') %>%
  html_text()
```

IMDB example

- Putting all of them together into a dataframe
- Some numeric variables have special characters in it, such as ".","min","$","M".
- We can manipulate strings using `stringr` package
  - `str_c`: combine two strings
  - `str_sub`: extract parts of a string
  - `str_to_lower`: change the text to lower case
  - `str_to_upper`: change the text to upper case
  - `str_detect`: check if a character vector matches a pattern
  - `str_replace`: replace matches with new strings
  - `str_split`: split a string up into pieces.
  - `str_remove`: remove a certain pattern of strings

IMDB example - Cleaning Data

```
movie_df = data.frame(ranking = ranking, title = title, rating = rating,
                      runtime = runtime, revenue = revenue, stringsAsFactors =
movie_df = movie_df %>% mutate(
                    ranking = str_remove_all(ranking,"\\.") %>% as.numeric(),
                    rating = as.numeric(rating),
                    runtime = str_remove_all(runtime," min") %>% as.numeric(),
                    revenue = str_remove_all(revenue, "M"),
                    revenue = str_replace_all(revenue, "\\$",""),
                    revenue = as.numeric(revenue))
```

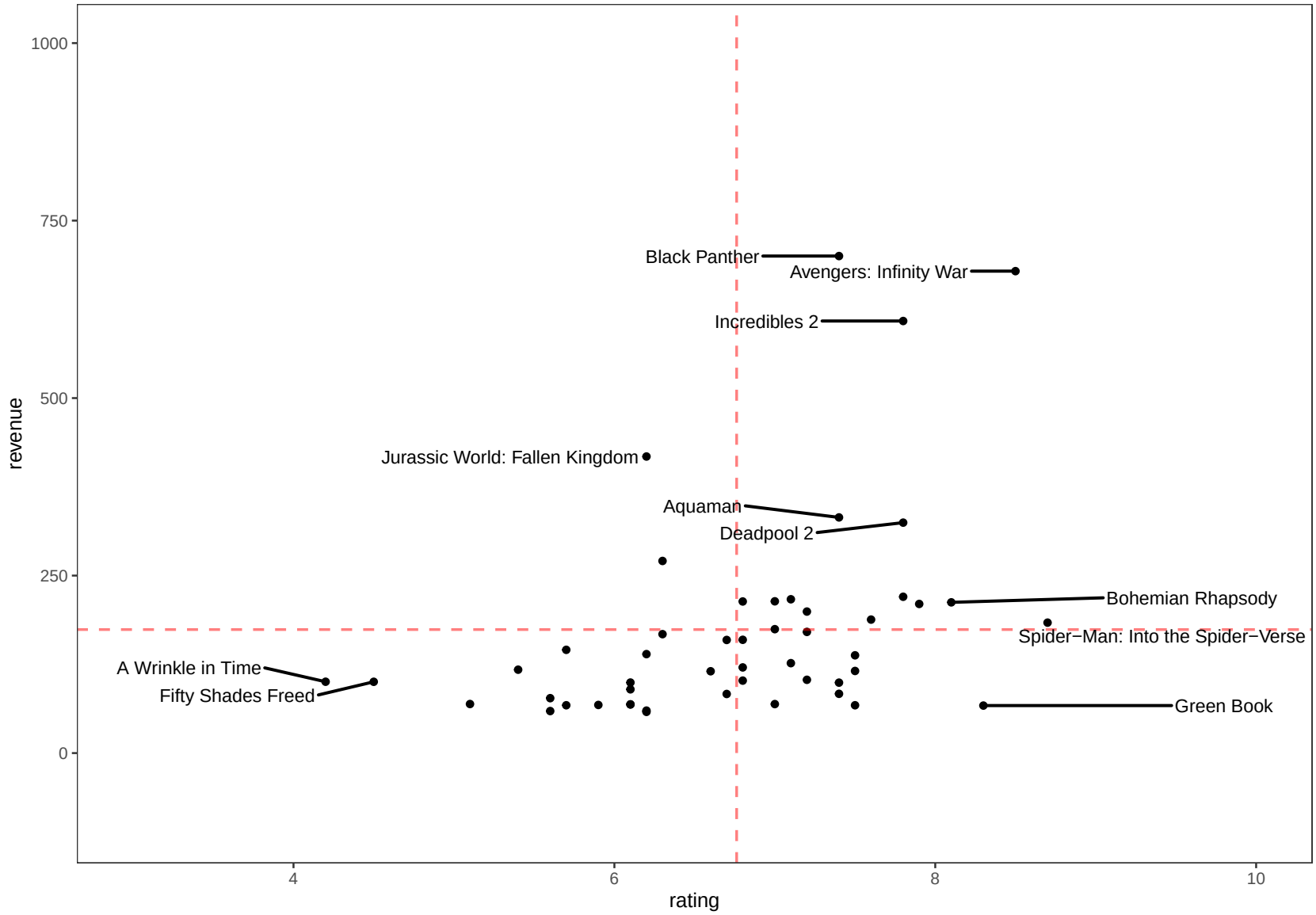
IMDB example - Visualization

- Dashed red line indicates the average rating and revenue respectively

```
p = movie_df %>% ggplot( aes(x=rating, y=revenue)) +
  geom_point() + theme_bw() + xlim(c(3,10)) + ylim(c(-100,1000)) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
  geom_hline(yintercept = mean(movie_df$revenue),
             size = 0.7, color = "red", alpha=0.5, linetype="dashed") +
  geom_vline(xintercept = mean(movie_df$rating),
             size = 0.7, color = "red", alpha=0.5, linetype="dashed") +
 geom_text_repel(movie_df %>% filter(revenue < 320, rating > 8),
                 mapping = aes(x=rating, y=revenue, label = title),
                 size = 3.5,
                 nudge_x = 1.5,
                 direction    = "y",
                 segment.size = 0.8) +
 geom_text_repel(movie_df %>% filter(rating < 5 | revenue > 320),
                 mapping = aes(x=rating, y=revenue, label = title),
                 size = 3.5,
                 nudge_x = -0.85,
                 direction    = "y",
                 segment.size = 0.8)
```

IMDB example

```
p
```



IMDB example - Modelling

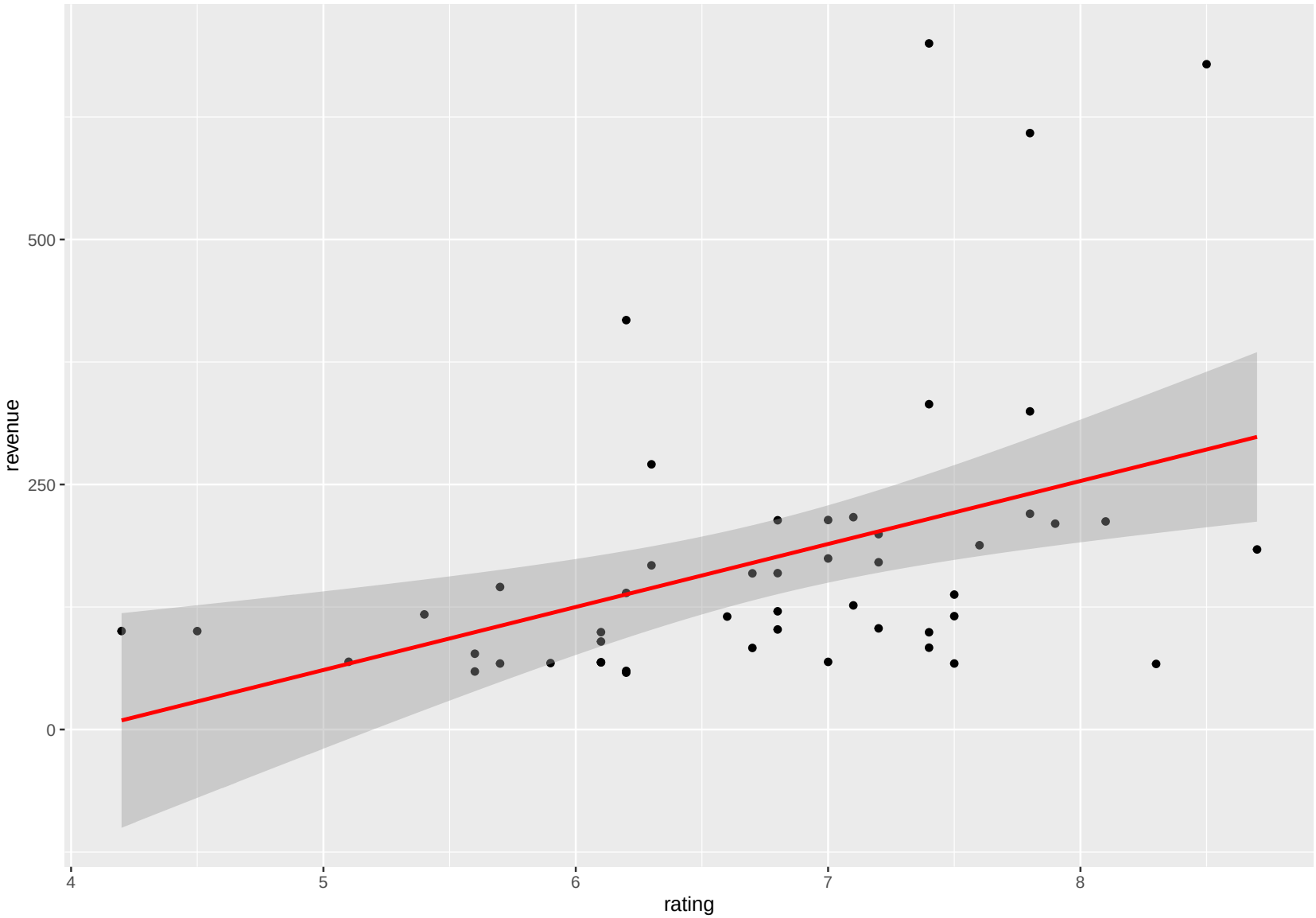
- We can build a simple linear regression model (Be aware that this is not a random sampling)

```r
summary(lm(revenue ~ rating, data = movie_df))
```

```
##
## Call:
## lm(formula = revenue ~ rating, data = movie_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -205.94  -73.04  -34.60   24.16  485.05
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -260.69     135.86  -1.919  0.06096 .
## rating         64.28      19.89   3.232  0.00223 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 134.7 on 48 degrees of freedom
## Multiple R-squared:  0.1787, Adjusted R-squared:  0.1616
## F-statistic: 10.44 on 1 and 48 DF,  p-value: 0.002227
```

IMDB example - Visualize the linear fit

```
ggplot(movie_df, aes(x = rating, y = revenue)) +
  geom_point() + stat_smooth(method = "lm", col = "red")
```



Conclusion

- Covered a few libraries, `rvest`, `dplyr`, `stringr`, `ggplot2`
- Identify the appropriate CSS selector is the key
- Data science workflow: