Web scraping

Data Science in a Box

datasciencebox.org

Modified by Tyler George

Scraping the web

Scraping the web: what? why?

- Increasing amount of data is available on the web

Scraping the web: what? why?

- Increasing amount of data is available on the web
- These data are provided in an unstructured format: you can always copy&paste, but it's time-consuming and prone to errors

Scraping the web: what? why?

- Increasing amount of data is available on the web
- These data are provided in an unstructured format: you can always copy&paste, but it's time-consuming and prone to errors
- Web scraping is the process of extracting this information automatically and transform it into a structured dataset

Scraping the web: what? why?

- Increasing amount of data is available on the web
- These data are provided in an unstructured format: you can always copy&paste, but it's time-consuming and prone to errors
- Web scraping is the process of extracting this information automatically and transform it into a structured dataset
- Two different scenarios:
  - Screen scraping: extract data from source code of website, with html parser (easy) or regular expression matching (less easy).
  - Web APIs (application programming interface): website offers a set of structured http requests that return JSON or XML files.

Web Scraping with rvest

Hypertext Markup Language

- Most of the data on the web is still largely available as HTML
- It is structured (hierarchical / tree based), but it''s often 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>
```

rvest

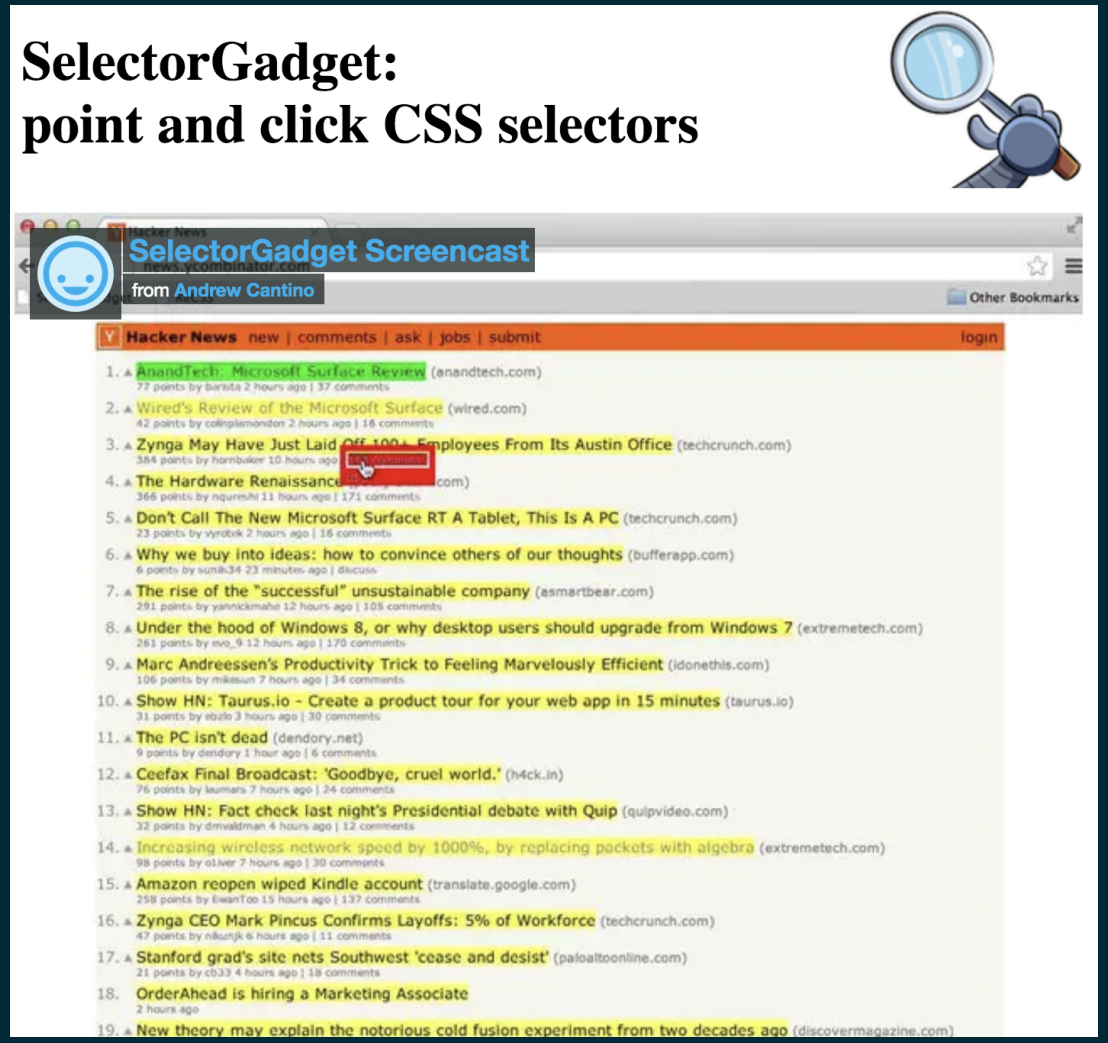
- The **rvest** package makes basic processing and manipulation of HTML data straight forward
- It's designed to work with pipelines built with %>%

Core rvest functions

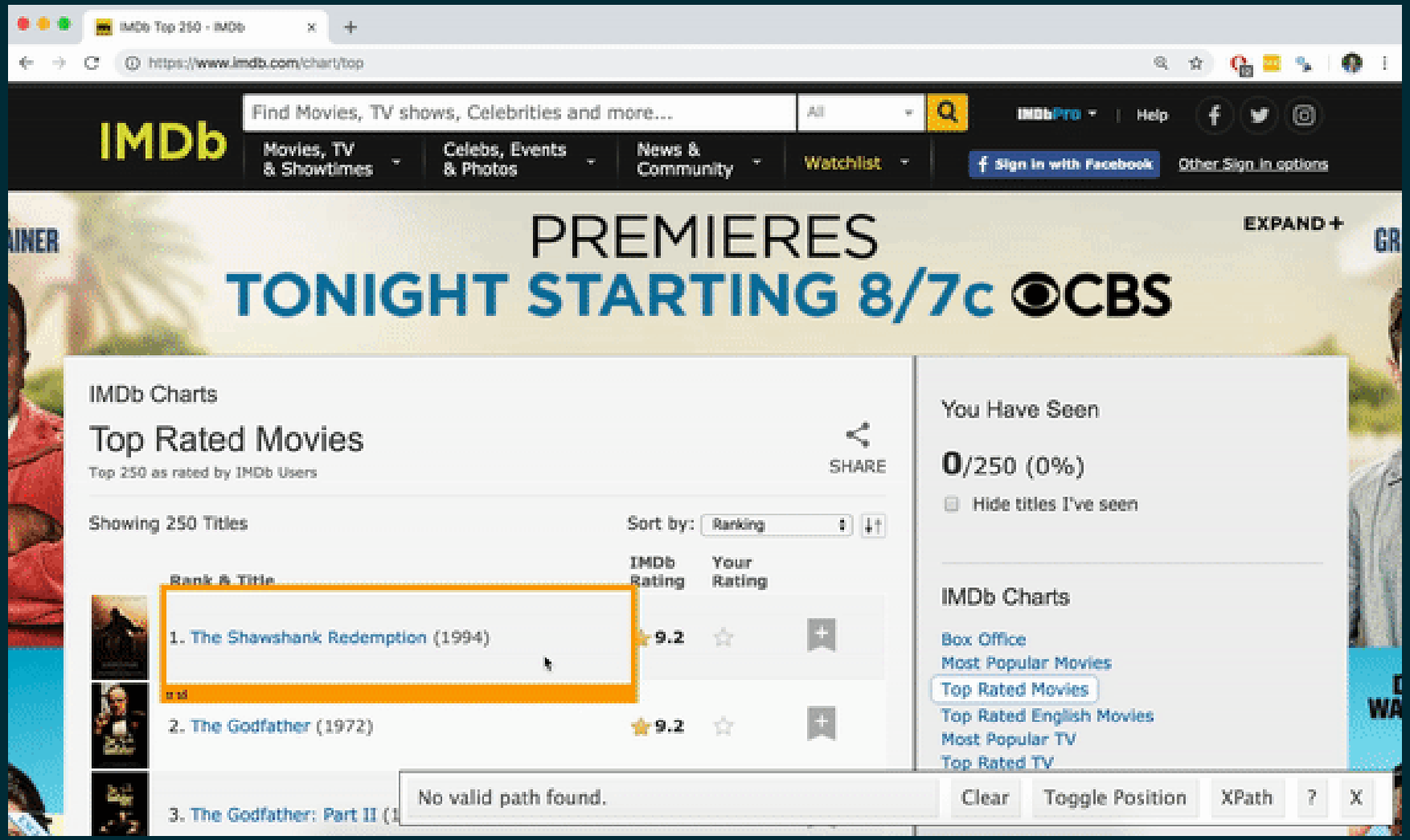
- `read_html` - Read HTML data from a url or character string
- `html_node` - Select a specified node from HTML document
- `html_nodes` - Select specified nodes from HTML document
- `html_table` - Parse an HTML table into a data frame
- `html_text` - Extract tag pairs' content
- `html_name` - Extract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

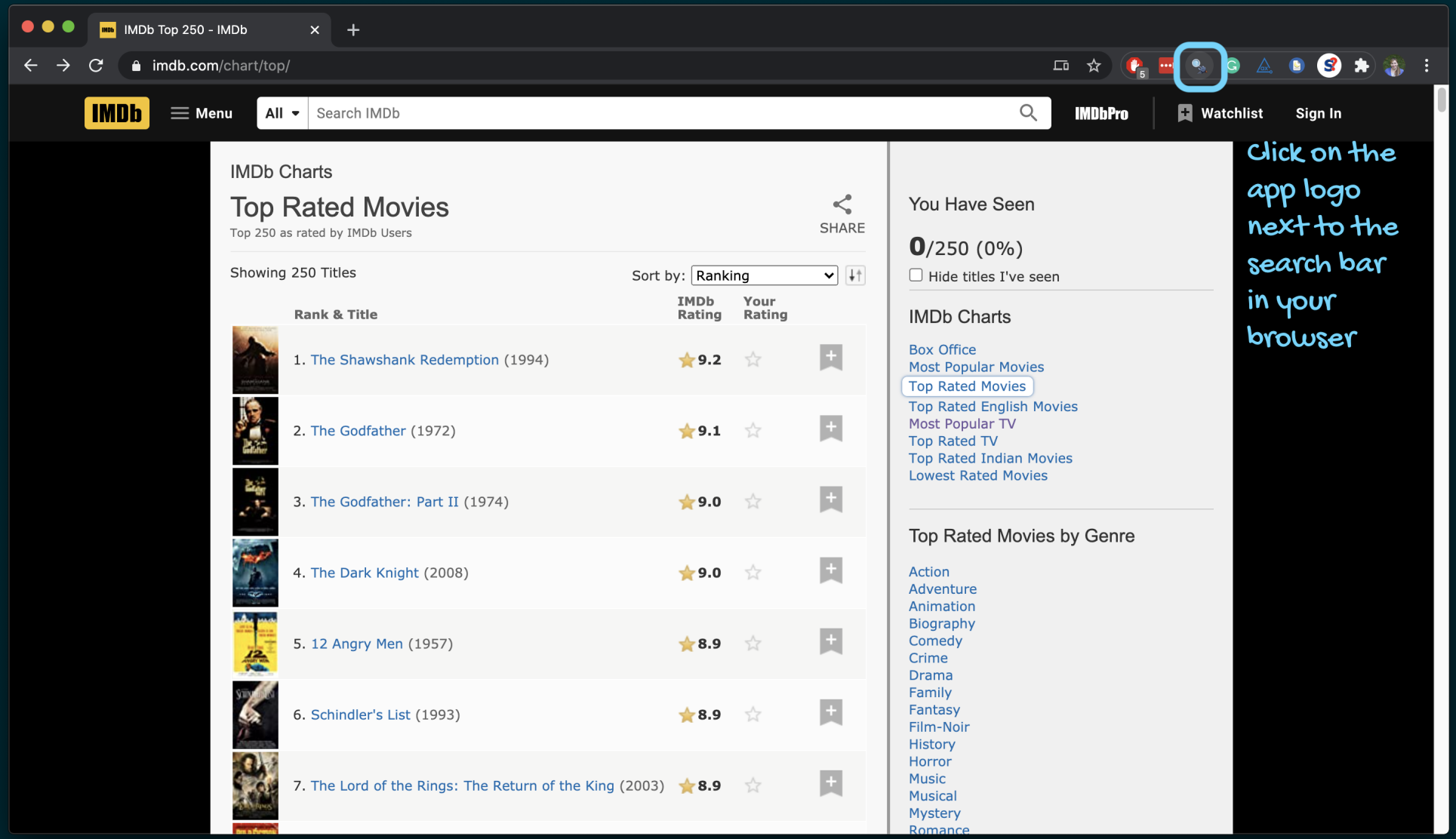
SelectorGadget

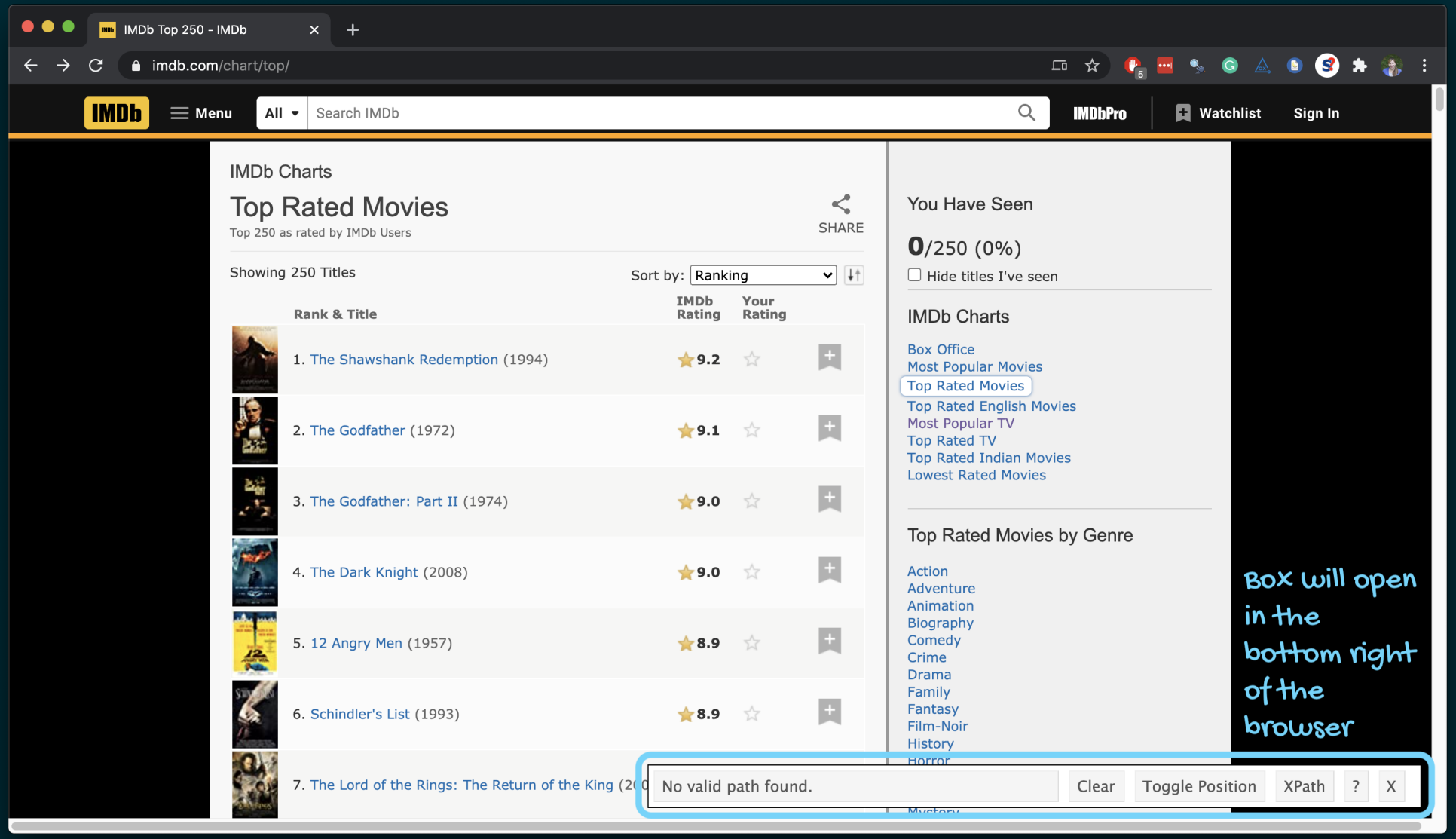
- Open source tool that eases CSS selector generation and discovery
- Easiest to use with the Chrome Extension
- Find out more on the SelectorGadget vignette

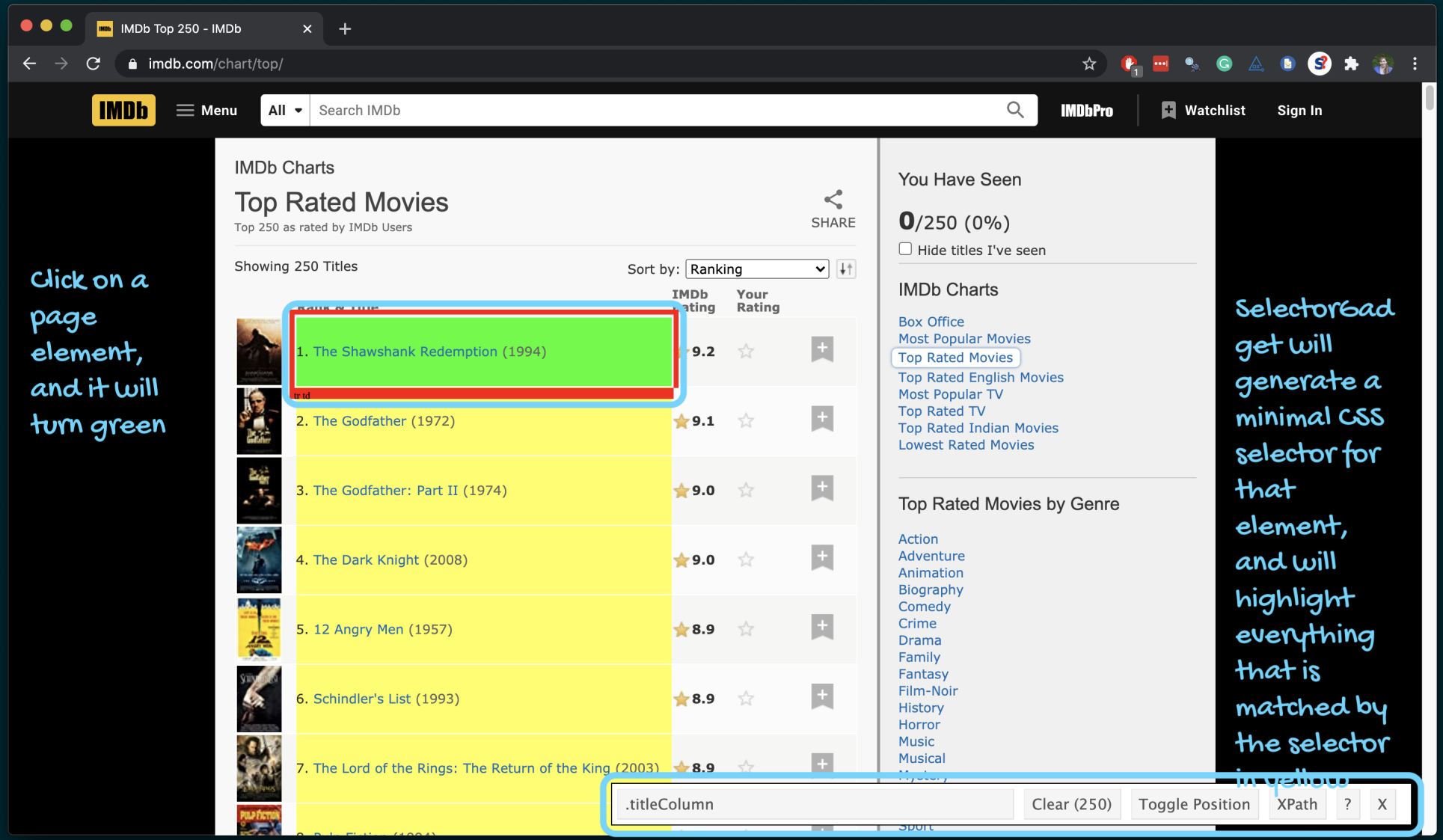


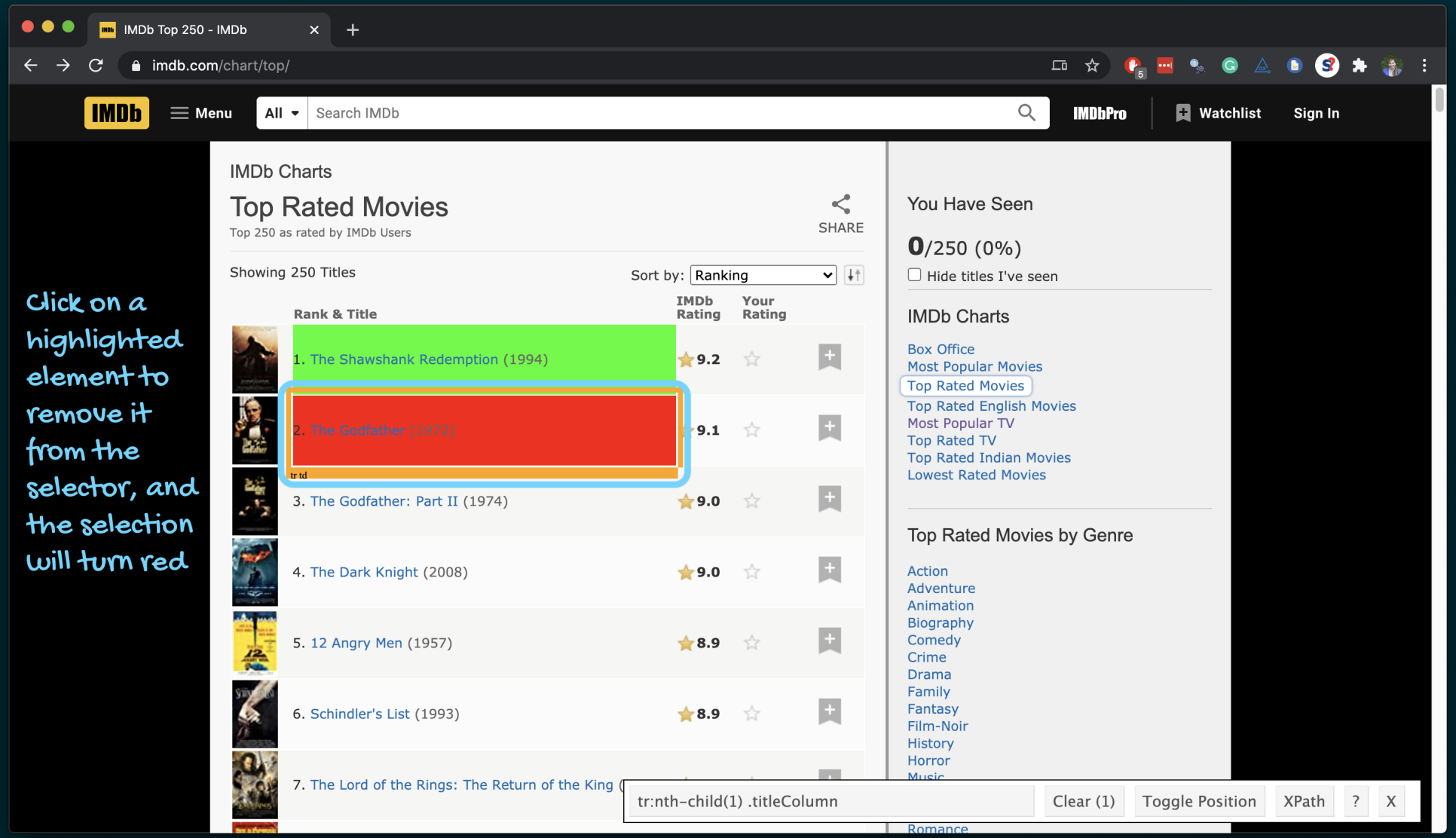
Using the SelectorGadget

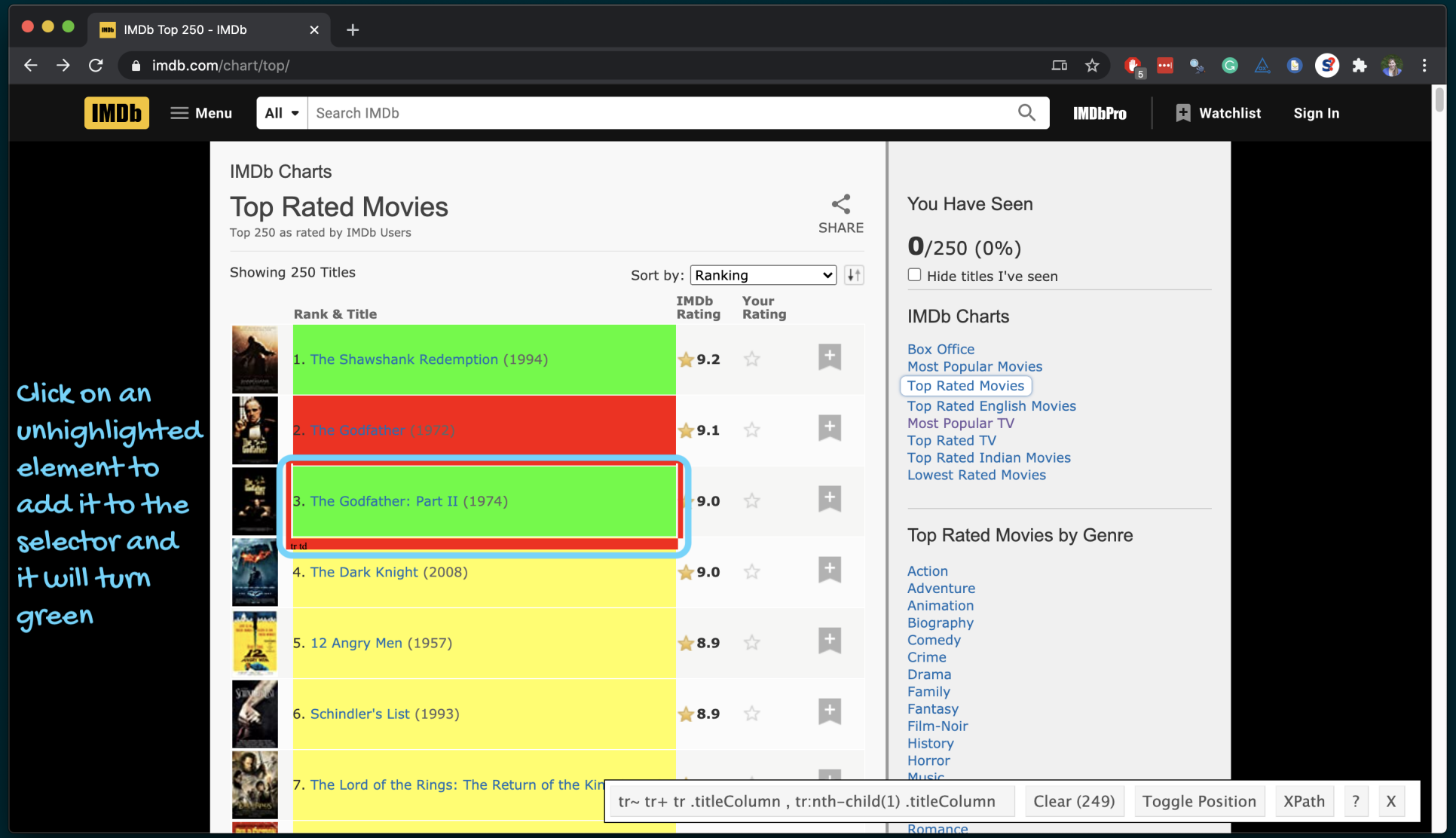



IMDb Top 250 - IMDb
imdb.com/chart/top/
IMDb
Menu
All
Search IMDb
IMDbPro
Watchlist
Sign In
IMDb Charts
Top Rated Movies
Top 250 as rated by IMDb Users
SHARE
Showing 250 Titles
Sort by: Ranking
Rank & Title
IMDb Rating
Your Rating
1. The Shawshank Redemption (1994) 9.2
2. The Godfather (1972) 9.1
3. 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
7. The Lord of the Rings: The Return of the King (2003) 8.9
You Have Seen
0/250 (0%)
Hide titles I've seen
IMDb Charts
Box Office
Most Popular Movies
Top Rated Movies
Top Rated English Movies
Most Popular TV
Top Rated TV
Top Rated Indian Movies
Lowest Rated Movies
Top Rated Movies by Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film-Noir
History
Horror
Music
Musical
Mystery
Romance
Click on the app logo next to the search bar in your browser


IMDb Top 250 - IMDb
imdb.com/chart/top/
IMDb
Menu
All
Search IMDb
IMDbPro
Watchlist
Sign In
IMDb Charts
Top Rated Movies
Top 250 as rated by IMDb Users
SHARE
Showing 250 Titles
Sort by: Ranking
Rank & Title
IMDb Rating
Your 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
7. The Lord of the Rings: The Return of the King
You Have Seen
0/250 (0%)
Hide titles I've seen
IMDb Charts
Box Office
Most Popular Movies
Top Rated Movies
Top Rated English Movies
Most Popular TV
Top Rated TV
Top Rated Indian Movies
Lowest Rated Movies
Top Rated Movies by Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film-Noir
History
Horror
Box will open in the bottom right of the browser
No valid path found.
Clear
Toggle Position
XPath
?
X

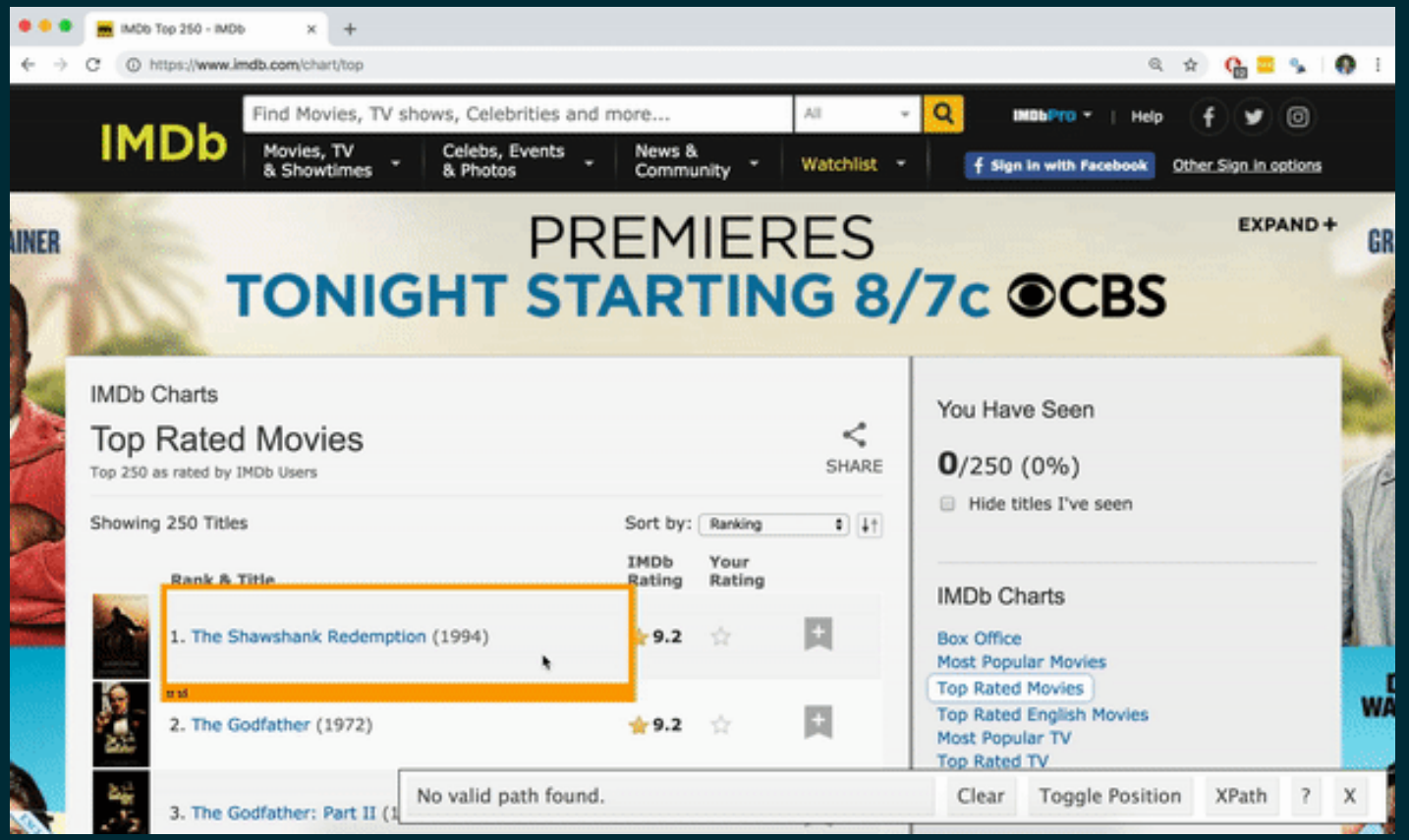

IMDb Top 250 - IMDb
imdb.com/chart/top/
IMDb
Menu
All
Search IMDb
IMDbPro
Watchlist
Sign In
IMDb Charts
Top Rated Movies
Top 250 as rated by IMDb Users
SHARE
Showing 250 Titles
Sort by: Ranking
IMDb Rating
Your Rating
1. The Shawshank Redemption (1994) 9.2
tr td
2. The Godfather (1972) 9.1
3. The Godfather: 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
You Have Seen
0/250 (0%)
Hide titles I've seen
IMDb Charts
Box Office
Most Popular Movies
Top Rated Movies
Top Rated English Movies
Most Popular TV
Top Rated TV
Top Rated Indian Movies
Lowest Rated Movies
Top Rated Movies by Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film-Noir
History
Horror
Music
Musical
.titleColumn
Clear (250)
Toggle Position
XPath
?
X
Click on a page element, and it will turn green
Selectorgad get will generate a minimal CSS selector for that element, and will highlight everything that is matched by the selector in yellow


IMDb Top 250 - IMDb
imdb.com/chart/top/
IMDb
Menu
All
Search IMDb
IMDbPro
Watchlist
Sign In
IMDb Charts
Top Rated Movies
Top 250 as rated by IMDb Users
SHARE
Showing 250 Titles
Sort by: Ranking
Rank & Title
IMDb Rating
Your Rating
1. The Shawshank Redemption (1994) 9.2
2. The Godfather (1972) 9.1
tr td
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
Click on a highlighted element to remove it from the selector, and the selection will turn red
You Have Seen
0/250 (0%)
Hide titles I've seen
IMDb Charts
Box Office
Most Popular Movies
Top Rated Movies
Top Rated English Movies
Most Popular TV
Top Rated TV
Top Rated Indian Movies
Lowest Rated Movies
Top Rated Movies by Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film-Noir
History
Horror
Romance
tr:nth-child(1) .titleColumn
Clear (1)
Toggle Position
XPath
?
X


IMDb Top 250 - IMDb
imdb.com/chart/top/
IMDb
Menu
All
Search IMDb
IMDbPro
Watchlist
Sign In
IMDb Charts
Top Rated Movies
Top 250 as rated by IMDb Users
SHARE
Showing 250 Titles
Sort by: Ranking
Rank & Title
IMDb Rating
Your Rating
1. The Shawshank Redemption (1994) 9.2
2. The Godfather (1972) 9.1
3. The Godfather: Part II (1974) 9.0
tr td
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 Kin
You Have Seen
0/250 (0%)
Hide titles I've seen
IMDb Charts
Box Office
Most Popular Movies
Top Rated Movies
Top Rated English Movies
Most Popular TV
Top Rated TV
Top Rated Indian Movies
Lowest Rated Movies
Top Rated Movies by Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film-Noir
History
Horror
Romance
Click on an unhighlighted element to add it to the selector and it will turn green
tr~ tr+ tr .titleColumn , tr:nth-child(1) .titleColumn
Clear (249)
Toggle Position
XPath
?
X

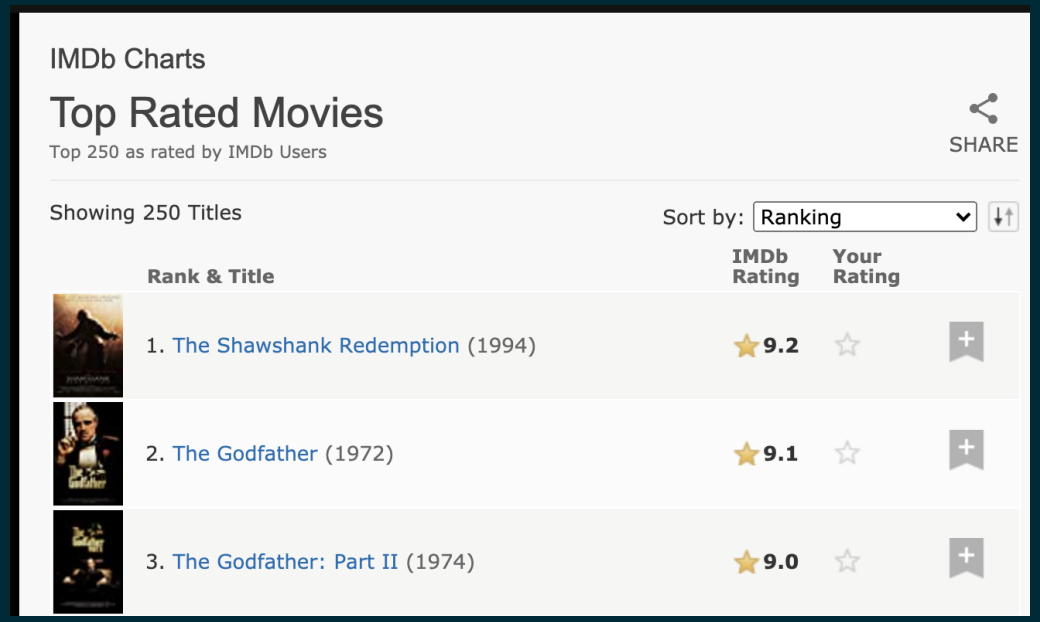
Using the SelectorGadget

Through this process of selection and rejection, SelectorGadget helps you come up with the appropriate CSS selector for your needs



Top 250 movies on IMDB

Take a look at the source code, look for the tag `table` tag (control +u in chrome): http://www.imdb.com/chart/top




```
599           <div class="desc">Showing <span>250</span> Titles</div>
600         </div>
601       </div>
602       <br class="clear">
603       <table class="chart full-width" data-caller-name="chart-top250movie">
604         <colgroup>
605           <col class="chartTableColumnPoster"/>
606           <col class="chartTableColumnTitle"/>
607           <col class="chartTableColumnIMDbRating"/>
608           <col class="chartTableColumnYourRating"/>
609           <col class="chartTableColumnWatchlistRibbon"/>
610         </colgroup>
611         <thead>
612         <tr>
613           <th></th>
614           <th>Rank &amp;amp; Title</th>
615           <th>IMDb Rating</th>
616           <th>Your Rating</th>
617           <th></th>
618         </tr>
619         </thead>
620         <tbody class="lister-list">
621
622   <tr>
623     <td class="posterColumn">
624
625     <span name="rk" data-value="1"></span>
626     <span name="ir" data-value="9.222796866017044"></span>
627     <span name="us" data-value="7.791552E11"></span>
628     <span name="nv" data-value="2297666"></span>
629     <span name="ur" data-value="-1.7772031339829564"></span>
630 <a href="/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-8962-
    327b42fe94b1&pf_rd_r=RP41R6C3PS7J108DDRNN&pf_rd_s=center-
    1&pf_rd_t=15506&pf_rd_i=top&ref_=chttp_tt_1"
631 > <img src="https://m.media-
    amazon.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGde
    QXVyMTMxODk2OTU@._V1_UY67_CR0,0,45,67_AL_.jpg" width="45" height="67" alt="The
    Shawshank Redemption"/>
632 </a>     </td>
```

First check if you're allowed!

```
library(robotstxt)
paths_allowed("http://www.imdb.com")
```

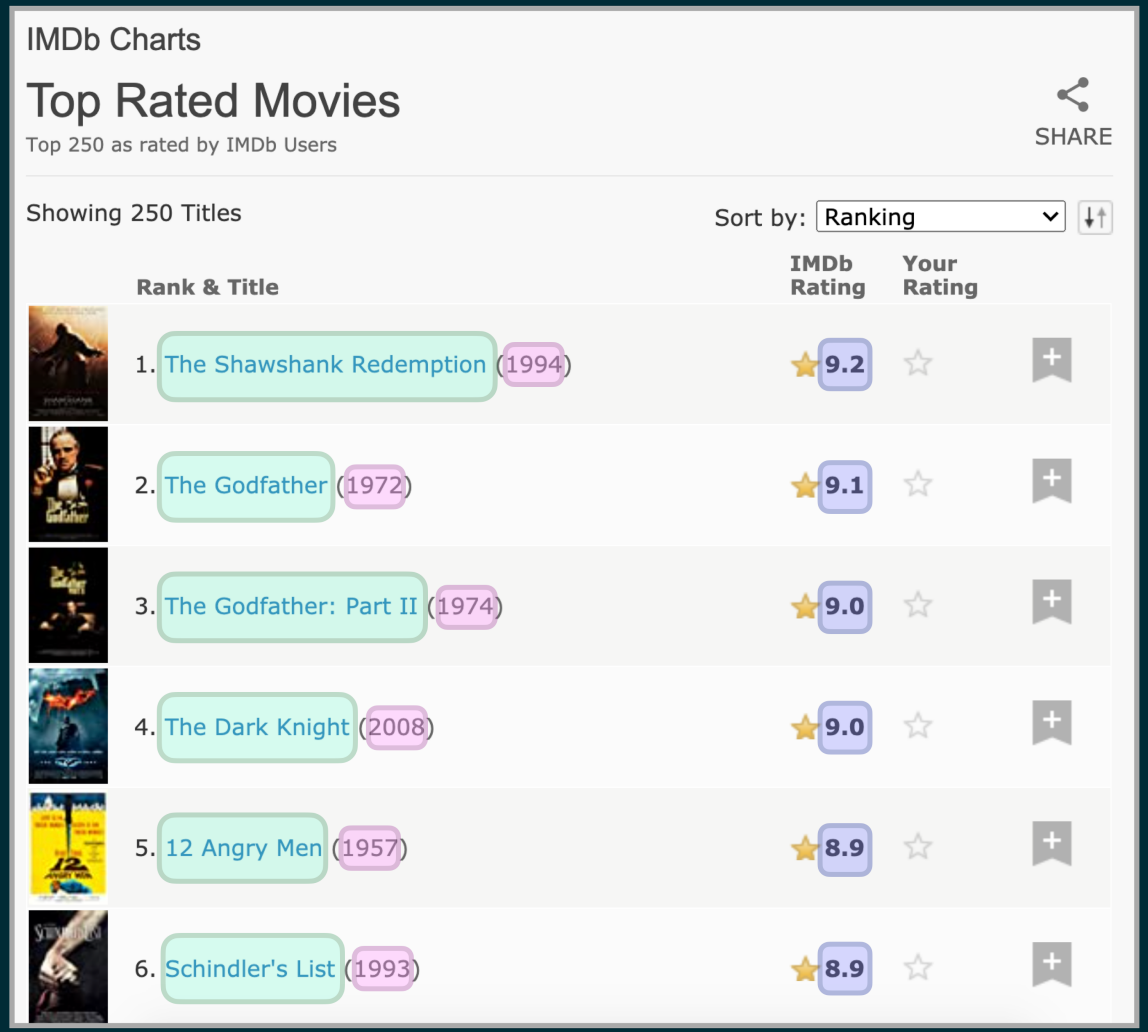
```
## [1] TRUE
```

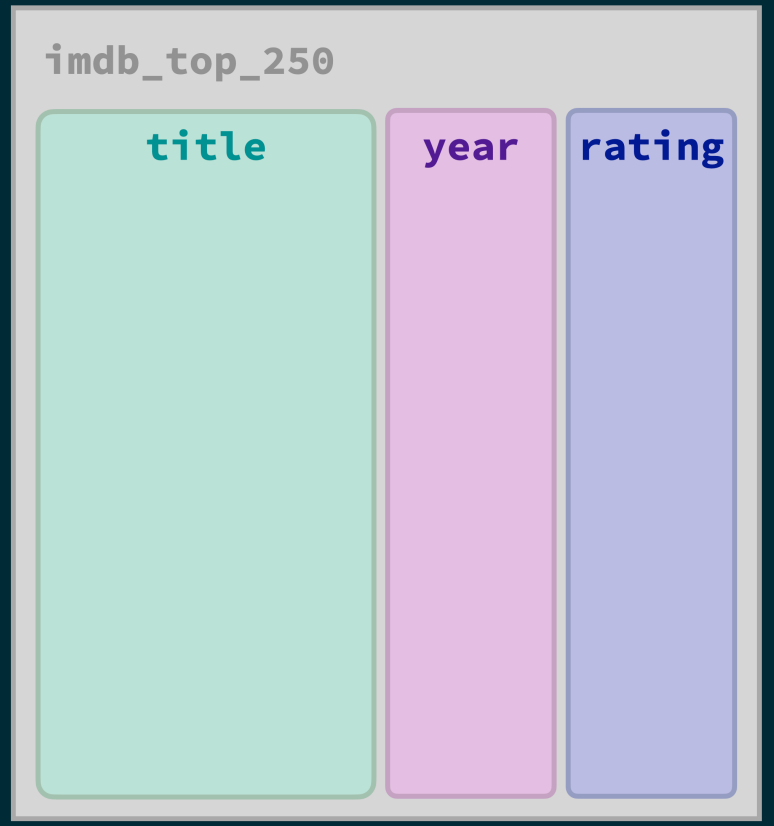
vs. e.g.

```
paths_allowed("http://www.facebook.com")
```

```
## [1] FALSE
```

Plan


IMDb Charts
Top Rated Movies
Top 250 as rated by IMDb Users
SHARE
Showing 250 Titles
Sort by: Ranking
Rank & Title
IMDb Rating
Your 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


imdb_top_250
title
year
rating

Plan

1. Read the whole page

2. Scrape movie titles and save as `titles`

3. Scrape years movies were made in and save as `years`

4. Scrape IMDB ratings and save as `ratings`

5. Create a data frame called `imdb_top_250` with variables `title`, `year`, and `rating`

Step 1. Read the whole page

Read the whole page

```
page <- read_html("https://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 ...
## [2] <body id="styleguide-v2" class="fixed">\n            <img  ...
```

A webpage in R

- Result is a list with 2 elements

```
typeof(page)
```

```
## [1] "list"
```

A webpage in R

- Result is a list with 2 elements

```
typeof(page)
```

```
## [1] "list"
```

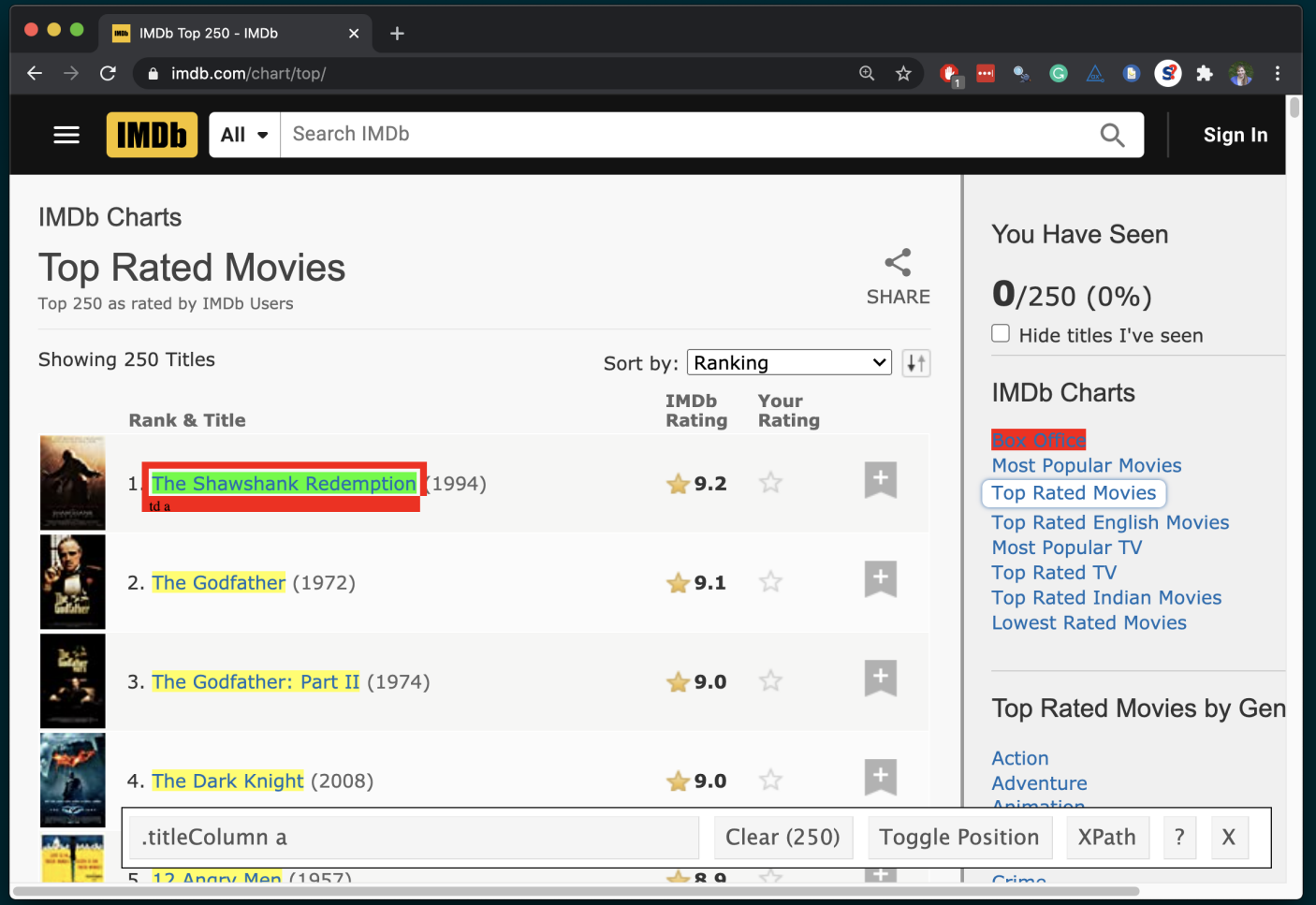
- that we need to convert to something more familiar, like a data frame....

```
class(page)
```

```
## [1] "xml_document" "xml_node"
```

Step 2. Scrape movie titles and save as `titles`

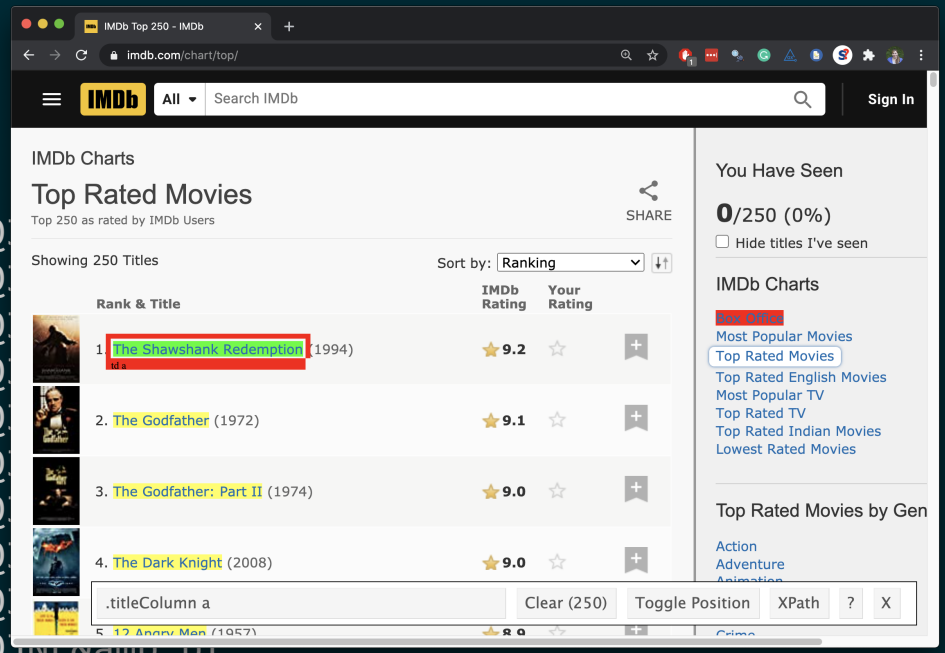
Scrape movie titles



Scrape the nodes

```
page %>%
  html_nodes(".titleColumn a")
```

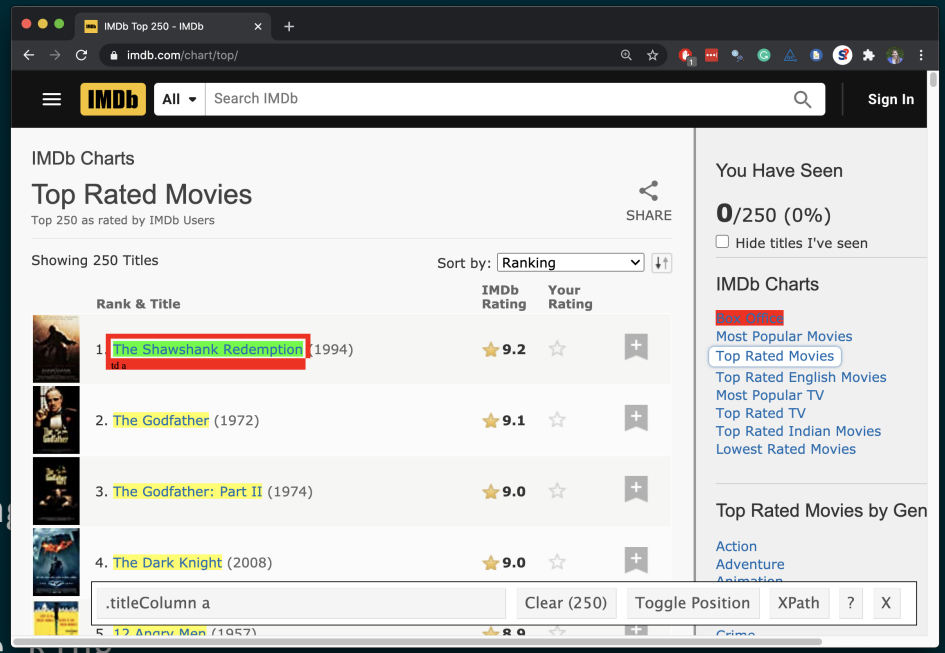
```
## {xml_nodeset (250)}
##  [1] <a href="/title/tt0111161/?pf_rd_m=A2FGELUUNOQ
##  [2] <a href="/title/tt0068646/?pf_rd_m=A2FGELUUNOQ
##  [3] <a href="/title/tt0071562/?pf_rd_m=A2FGELUUNOQ
##  [4] <a href="/title/tt0468569/?pf_rd_m=A2FGELUUNOQ
##  [5] <a href="/title/tt0050083/?pf_rd_m=A2FGELUUNOQ
##  [6] <a href="/title/tt0108052/?pf_rd_m=A2FGELUUNOQ
##  [7] <a href="/title/tt0167260/?pf_rd_m=A2FGELUUNOQ
##  [8] <a href="/title/tt0110912/?pf_rd_m=A2FGELUUNOQ
##  [9] <a href="/title/tt0060196/?pf_rd_m=A2FGELUUNOQ
## [10] <a href="/title/tt0120737/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_ ...
## [11] <a href="/title/tt0137523/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_ ...
## [12] <a href="/title/tt0109830/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_ ...
## [13] <a href="/title/tt1375666/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_ ...
## [14] <a href="/title/tt0167261/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_ ...
## [15] <a href="/title/tt0080684/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_ ...
## [16] <a href="/title/tt0133093/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_ ...
...
```



Extract the text from the nodes

```
page %>%
  html_nodes(".titleColumn a") %>%
  html_text()
```

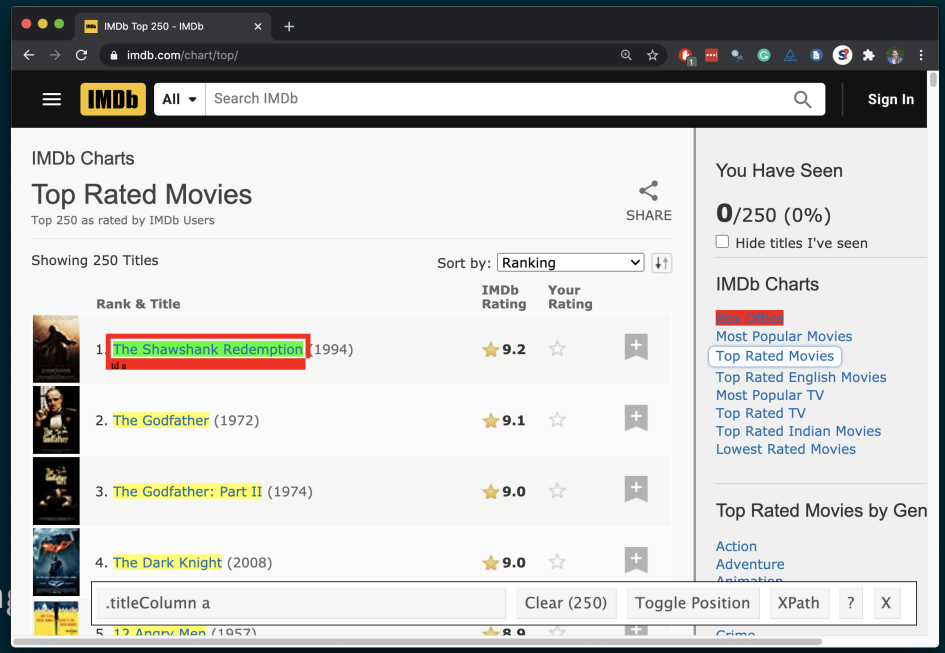
```
##   [1] "The Shawshank Redemption"
##   [2] "The Godfather"
##   [3] "The Godfather: Part II"
##   [4] "The Dark Knight"
##   [5] "12 Angry Men"
##   [6] "Schindler's List"
##   [7] "The Lord of the Rings: The Return of the Kin
##   [8] "Pulp Fiction"
##   [9] "The Good, the Bad and the Ugly"
##  [10] "The Lord of the Rings: The Fellowship of the Ring
##  [11] "Fight Club"
##  [12] "Forrest Gump"
##  [13] "Inception"
##  [14] "The Lord of the Rings: The Two Towers"
##  [15] "Star Wars: Episode V - The Empire Strikes Back"
##  [16] "The Matrix"
...
```



Save as `titles`

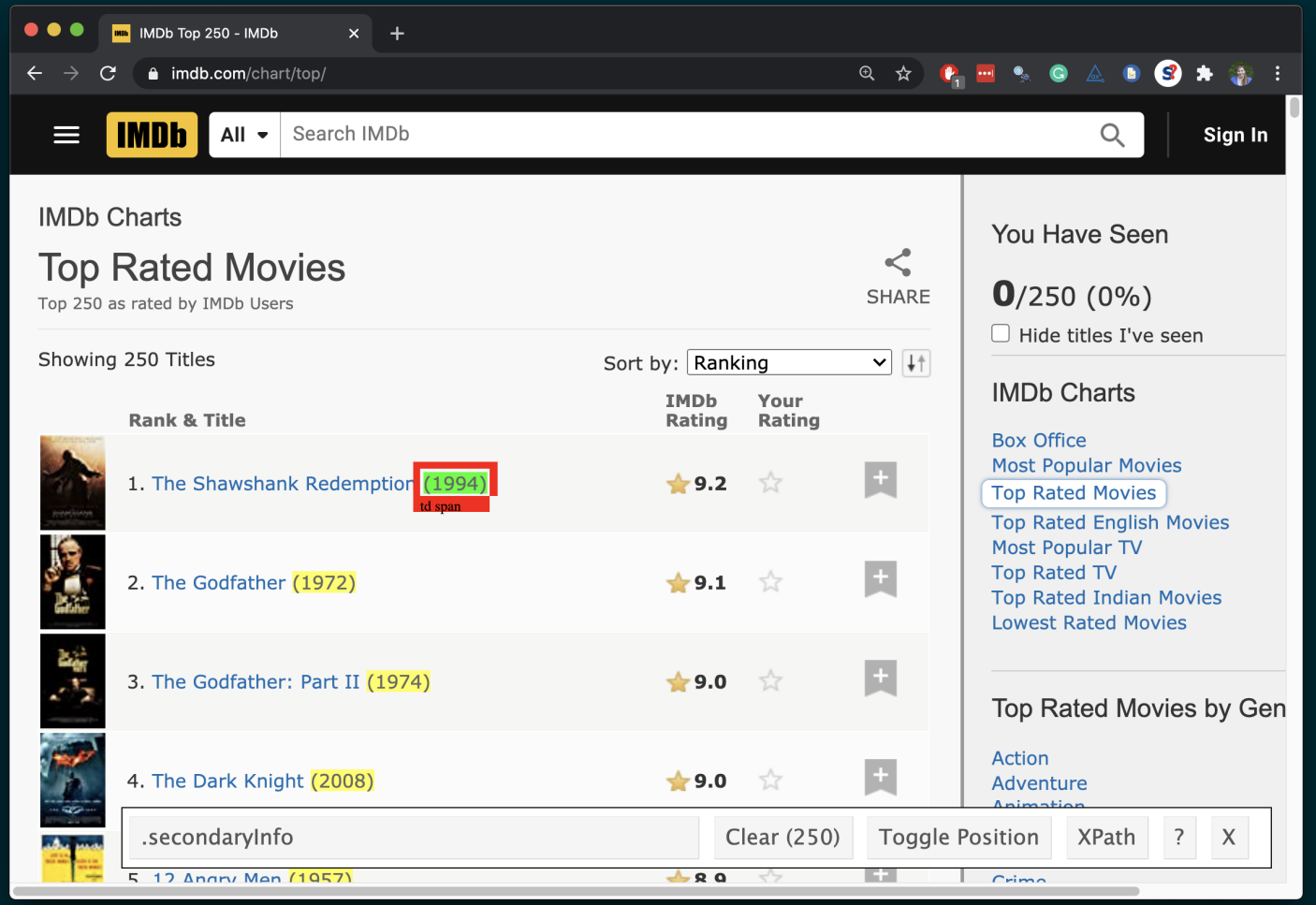
```
titles <- page %>%
  html_nodes(".titleColumn a") %>%
  html_text()

titles
```

```
##   [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 Kin
##   [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] "The Lord of the Rings: The Two Towers"
...
```



Step 3. Scrape year movies were made and save as `years`

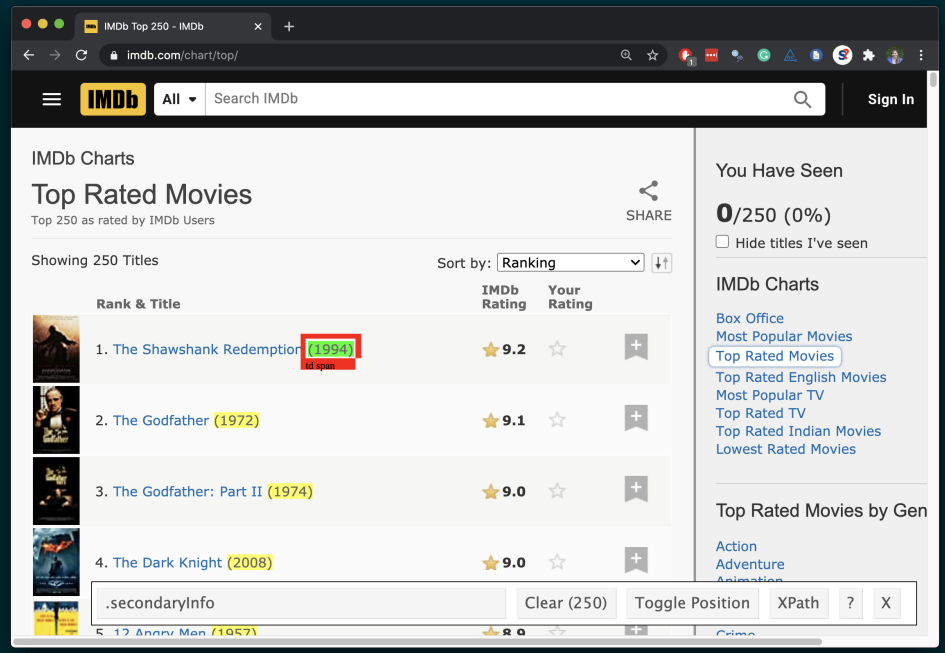
Scrape years movies were made in



Scrape the nodes

```
page %>%
  html_nodes(".secondaryInfo")
```

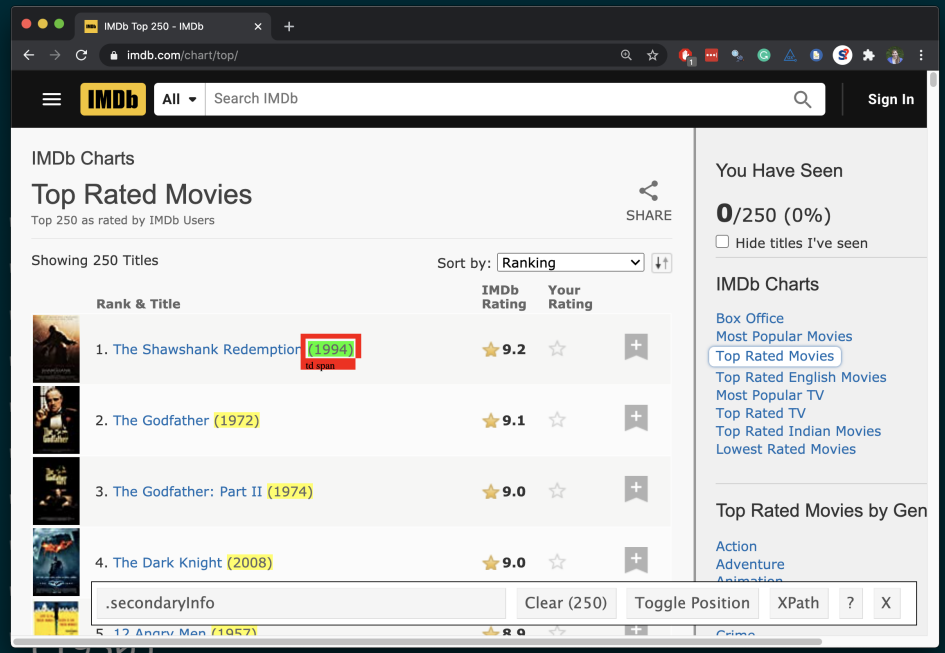
```
## {xml_nodeset (250)}
##  [1] <span class="secondaryInfo">(1994)</span>
##  [2] <span class="secondaryInfo">(1972)</span>
##  [3] <span class="secondaryInfo">(1974)</span>
##  [4] <span class="secondaryInfo">(2008)</span>
##  [5] <span class="secondaryInfo">(1957)</span>
##  [6] <span class="secondaryInfo">(1993)</span>
##  [7] <span class="secondaryInfo">(2003)</span>
##  [8] <span class="secondaryInfo">(1994)</span>
##  [9] <span class="secondaryInfo">(1966)</span>
## [10] <span class="secondaryInfo">(2001)</span>
## [11] <span class="secondaryInfo">(1999)</span>
## [12] <span class="secondaryInfo">(1994)</span>
## [13] <span class="secondaryInfo">(2010)</span>
## [14] <span class="secondaryInfo">(2002)</span>
## [15] <span class="secondaryInfo">(1980)</span>
## [16] <span class="secondaryInfo">(1999)</span>
## ...
```



Extract the text from the nodes

```
page %>%
  html_nodes(".secondaryInfo") %>%
  html_text()
```

```
##   [1] "(1994)" "(1972)" "(1974)" "(2008)" "(1957)"
##   [7] "(2003)" "(1994)" "(1966)" "(2001)" "(1999)"
##  [13] "(2010)" "(2002)" "(1980)" "(1999)" "(1990)"
##  [19] "(1954)" "(1995)" "(1991)" "(2002)" "(1997)"
##  [25] "(1977)" "(1998)" "(2014)" "(2001)" "(1999)"
##  [31] "(1994)" "(1962)" "(2002)" "(1991)" "(1995)"
##  [37] "(1960)" "(1994)" "(1936)" "(1998)" "(1988)"
##  [43] "(2014)" "(2000)" "(2006)" "(2011)" "(2006)"
##  [49] "(1968)" "(1954)" "(1988)" "(1979)" "(1979)"
##  [55] "(1981)" "(1940)" "(2006)" "(2012)" "(1957)"
##  [61] "(2008)" "(2018)" "(1957)" "(1980)" "(2018)" "(1964)"
##  [67] "(1997)" "(2003)" "(2019)" "(2016)" "(2017)" "(2012)"
##  [73] "(1986)" "(1984)" "(2018)" "(2019)" "(1981)" "(1963)"
##  [79] "(2009)" "(1999)" "(1995)" "(1984)" "(1995)" "(2009)"
##  [85] "(2020)" "(1997)" "(1983)" "(1968)" "(1992)" "(1931)"
##  [91] "(1958)" "(2007)" "(1985)" "(1941)" "(2012)" "(2000)"
## ...
```



Clean up the text

We need to go from `"(1994)"` to `1994`:

- Remove `(` and `)`: string manipulation
- Convert to numeric: `as.numeric()`

stringr

- **stringr** provides a cohesive set of functions designed to make working with strings as easy as possible
- Functions in stringr start with `str_*()`, e.g.
  - `str_remove()` to remove a pattern from a string

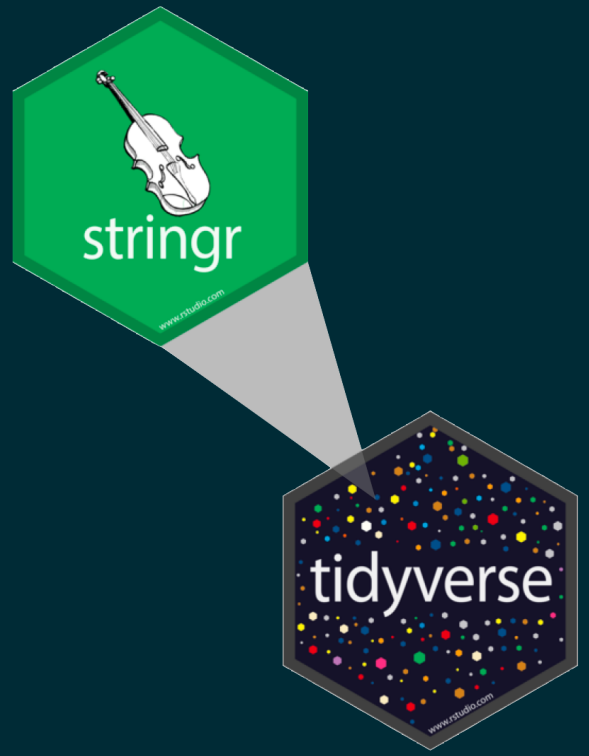
```
str_remove(string = "jello", pattern = "el")
```

```
## [1] "jlo"
```

  - `str_replace()` to replace a pattern with another

```
str_replace(string = "jello", pattern = "j", replacement = "h")
```

```
## [1] "hello"
```



Clean up the text

```
page %>%
  html_nodes(".secondaryInfo") %>%
  html_text() %>%
  str_remove("\\(") # remove (
```

```
##   [1] "1994)" "1972)" "1974)" "2008)" "1957)" "1993)" "2003)"
##   [8] "1994)" "1966)" "2001)" "1999)" "1994)" "2010)" "2002)"
##  [15] "1980)" "1999)" "1990)" "1975)" "1954)" "1995)" "1991)"
##  [22] "2002)" "1997)" "1946)" "1977)" "1998)" "2014)" "2001)"
##  [29] "1999)" "2019)" "1994)" "1962)" "2002)" "1991)" "1995)"
##  [36] "1985)" "1960)" "1994)" "1936)" "1998)" "1988)" "1931)"
##  [43] "2014)" "2000)" "2006)" "2011)" "2006)" "1942)" "1968)"
##  [50] "1954)" "1988)" "1979)" "1979)" "2000)" "1981)" "1940)"
##  [57] "2006)" "2012)" "1957)" "1950)" "2008)" "2018)" "1957)"
##  [64] "1980)" "2018)" "1964)" "1997)" "2003)" "2019)" "2016)"
##  [71] "2017)" "2012)" "1986)" "1984)" "2018)" "2019)" "1981)"
##  [78] "1963)" "2009)" "1999)" "1995)" "1984)" "1995)" "2009)"
##  [85] "2020)" "1997)" "1983)" "1968)" "1992)" "1931)" "1958)"
##  [92] "2007)" "1985)" "1941)" "2012)" "2000)" "1952)" "1959)"
##  [99] "2004)" "1948)" "1952)" "1962)" "1921)" "1987)" "2016)"
...
```

Clean up the text

```
page %>%
  html_nodes(".secondaryInfo") %>%
  html_text() %>%
  str_remove("\\(") %>% # remove (
  str_remove("\\)") # remove )
```

```
##   [1] "1994" "1972" "1974" "2008" "1957" "1993" "2003" "1994"
##   [9] "1966" "2001" "1999" "1994" "2010" "2002" "1980" "1999"
##  [17] "1990" "1975" "1954" "1995" "1991" "2002" "1997" "1946"
##  [25] "1977" "1998" "2014" "2001" "1999" "2019" "1994" "1962"
##  [33] "2002" "1991" "1995" "1985" "1960" "1994" "1936" "1998"
##  [41] "1988" "1931" "2014" "2000" "2006" "2011" "2006" "1942"
##  [49] "1968" "1954" "1988" "1979" "1979" "2000" "1981" "1940"
##  [57] "2006" "2012" "1957" "1950" "2008" "2018" "1957" "1980"
##  [65] "2018" "1964" "1997" "2003" "2019" "2016" "2017" "2012"
##  [73] "1986" "1984" "2018" "2019" "1981" "1963" "2009" "1999"
##  [81] "1995" "1984" "1995" "2009" "2020" "1997" "1983" "1968"
##  [89] "1992" "1931" "1958" "2007" "1985" "1941" "2012" "2000"
##  [97] "1952" "1959" "2004" "1948" "1952" "1962" "1921" "1987"
## [105] "2016" "1960" "2010" "2020" "1944" "1971" "1927" "1976"
...
```

Convert to numeric

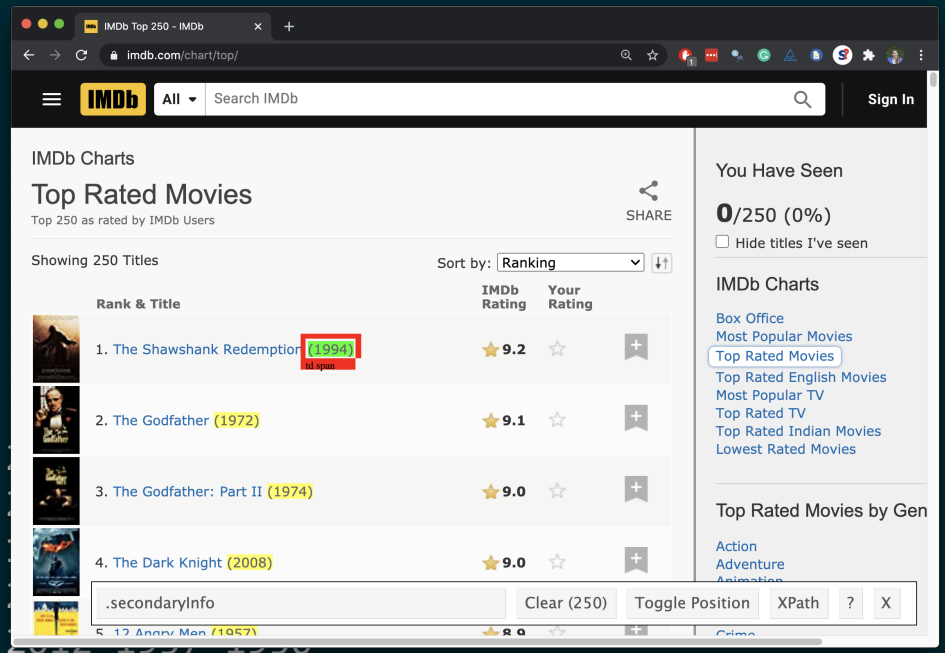
```
page %>%
  html_nodes(".secondaryInfo") %>%
  html_text() %>%
  str_remove("\\(") %>% # remove (
  str_remove("\\)") %>% # remove )
  as.numeric()
```

```
##   [1] 1994 1972 1974 2008 1957 1993 2003 1994 1966 2001 1999 1994
##  [13] 2010 2002 1980 1999 1990 1975 1954 1995 1991 2002 1997 1946
##  [25] 1977 1998 2014 2001 1999 2019 1994 1962 2002 1991 1995 1985
##  [37] 1960 1994 1936 1998 1988 1931 2014 2000 2006 2011 2006 1942
##  [49] 1968 1954 1988 1979 1979 2000 1981 1940 2006 2012 1957 1950
##  [61] 2008 2018 1957 1980 2018 1964 1997 2003 2019 2016 2017 2012
##  [73] 1986 1984 2018 2019 1981 1963 2009 1999 1995 1984 1995 2009
##  [85] 2020 1997 1983 1968 1992 1931 1958 2007 1985 1941 2012 2000
##  [97] 1952 1959 2004 1948 1952 1962 1921 1987 2016 1960 2010 2020
## [109] 1944 1971 1927 1976 2011 1955 1973 1983 2000 2019 2001 1962
## [121] 2010 1965 2009 1989 1995 1997 1961 1985 1988 1950 2018 2004
## [133] 1975 2021 1950 1959 2005 1992 1997 2004 2013 1961 1963 2007
## [145] 1995 1948 2006 2001 2009 1980 1975 1974 1988 1998 2010 1925
...
```

Save as years

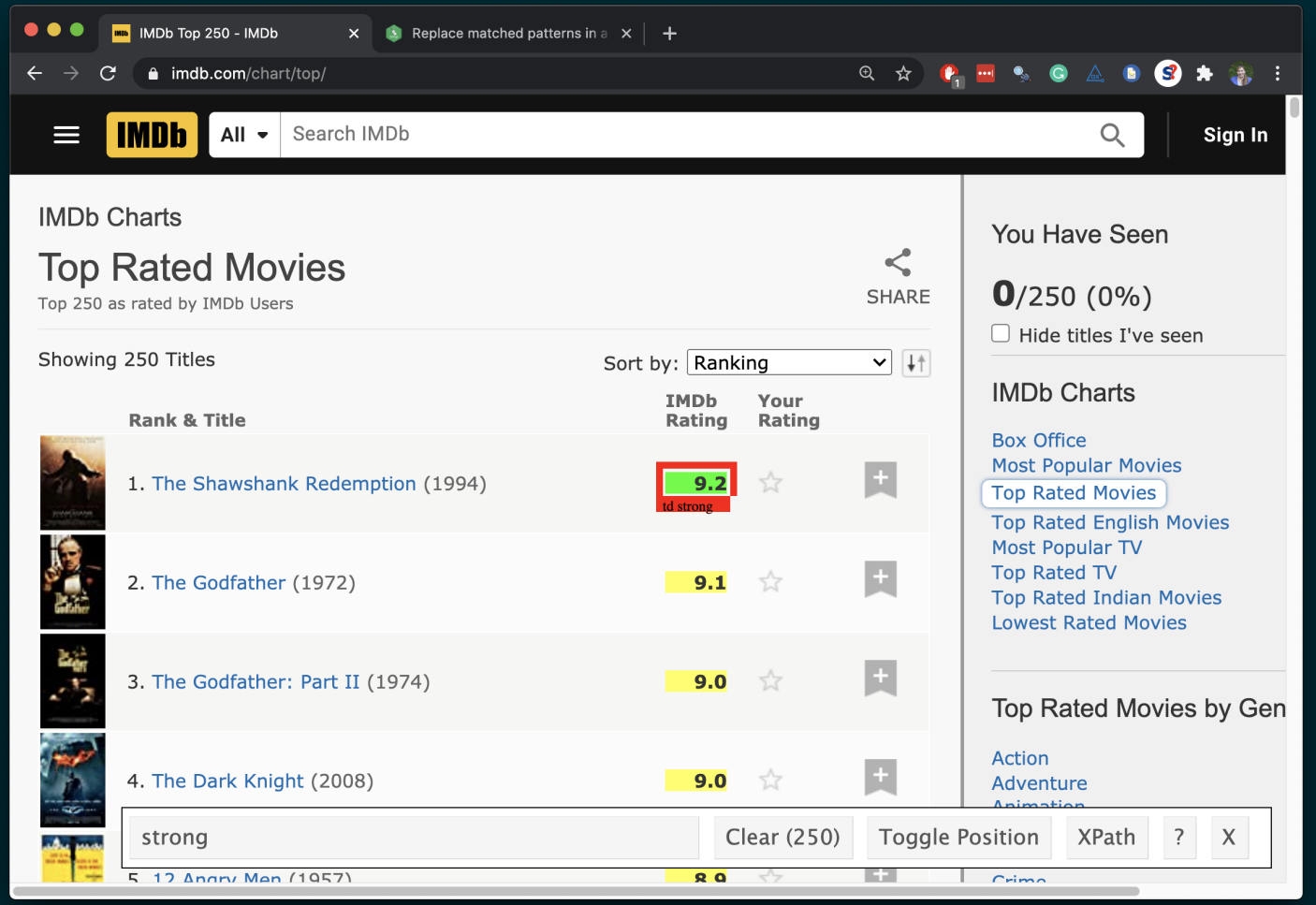
```
years <- page %>%
  html_nodes(".secondaryInfo") %>%
  html_text() %>%
  str_remove("\\(") %>% # remove (
  str_remove("\\)") %>% # remove )
  as.numeric()

years
```

```
##   [1] 1994 1972 1974 2008 1957 1993 2003 1994 1966
##  [13] 2010 2002 1980 1999 1990 1975 1954 1995 1991
##  [25] 1977 1998 2014 2001 1999 2019 1994 1962 2002
##  [37] 1960 1994 1936 1998 1988 1931 2014 2000 2006
##  [49] 1968 1954 1988 1979 1979 2000 1981 1940 2006
##  [61] 2008 2018 1957 1980 2018 1964 1997 2003 2019 2016 2017 2012
##  [73] 1986 1984 2018 2019 1981 1963 2009 1999 1995 1984 1995 2009
##  [85] 2020 1997 1983 1968 1992 1931 1958 2007 1985 1941 2012 2000
##  [97] 1952 1959 2004 1948 1952 1962 1921 1987 2016 1960 2010 2020
## [109] 1944 1971 1927 1976 2011 1955 1973 1983 2000 2019 2001 1962
## [121] 2010 1965 2009 1989 1995 1997 1961 1985 1988 1950 2018 2004
...
```



Step 4. Scrape IMDB ratings and save as `ratings`

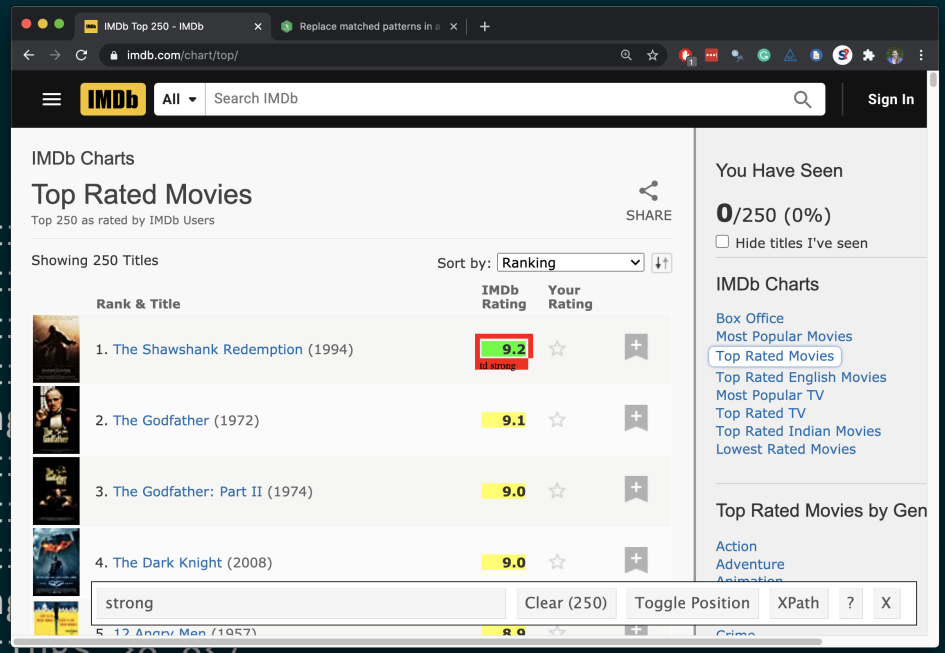
Scrape IMDB ratings



Scrape the nodes

```
page %>%
  html_nodes("strong")
```

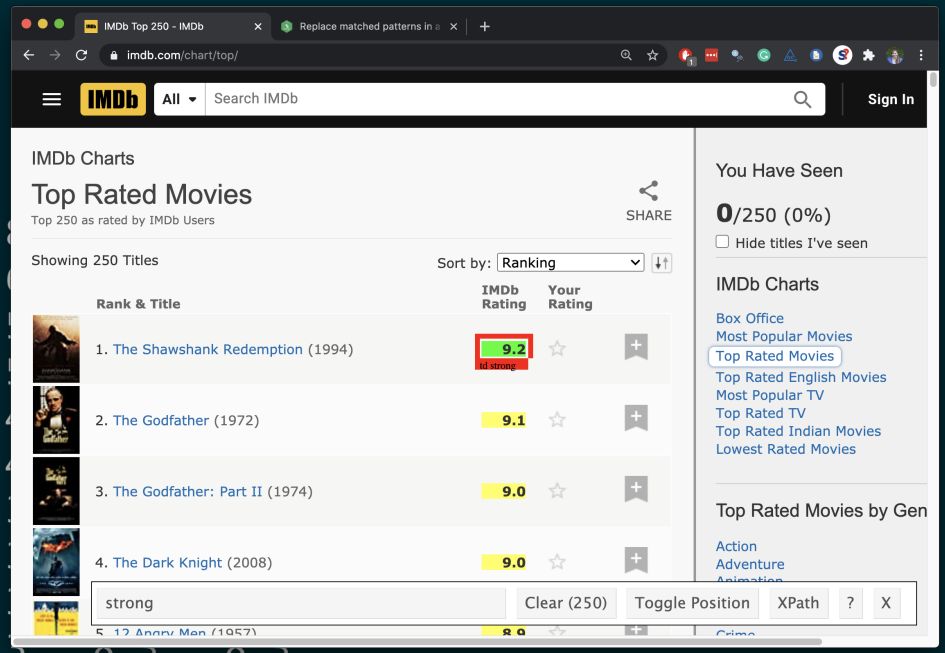
```
## {xml_nodeset (250)}
##  [1] <strong title="9.2 based on 2,498,614 user rat
##  [2] <strong title="9.1 based on 1,722,841 user rat
##  [3] <strong title="9.0 based on 1,195,759 user rat
##  [4] <strong title="9.0 based on 2,448,983 user rat
##  [5] <strong title="8.9 based on 738,054 user ratin
##  [6] <strong title="8.9 based on 1,279,061 user rat
##  [7] <strong title="8.9 based on 1,726,847 user rat
##  [8] <strong title="8.8 based on 1,928,854 user rat
##  [9] <strong title="8.8 based on 724,318 user ratin
## [10] <strong title="8.8 based on 1,748,127 user rating
## [11] <strong title="8.8 based on 1,965,915 user ratings">8.8</ ...
## [12] <strong title="8.7 based on 1,928,857 user ratings">8.7</ ...
## [13] <strong title="8.7 based on 2,197,508 user ratings">8.7</ ...
## [14] <strong title="8.7 based on 1,561,025 user ratings">8.7</ ...
## [15] <strong title="8.7 based on 1,216,407 user ratings">8.7</ ...
## [16] <strong title="8.6 based on 1,782,159 user ratings">8.6</ ...
...
```



Extract the text from the nodes

```
page %>%
  html_nodes("strong") %>%
  html_text()
```

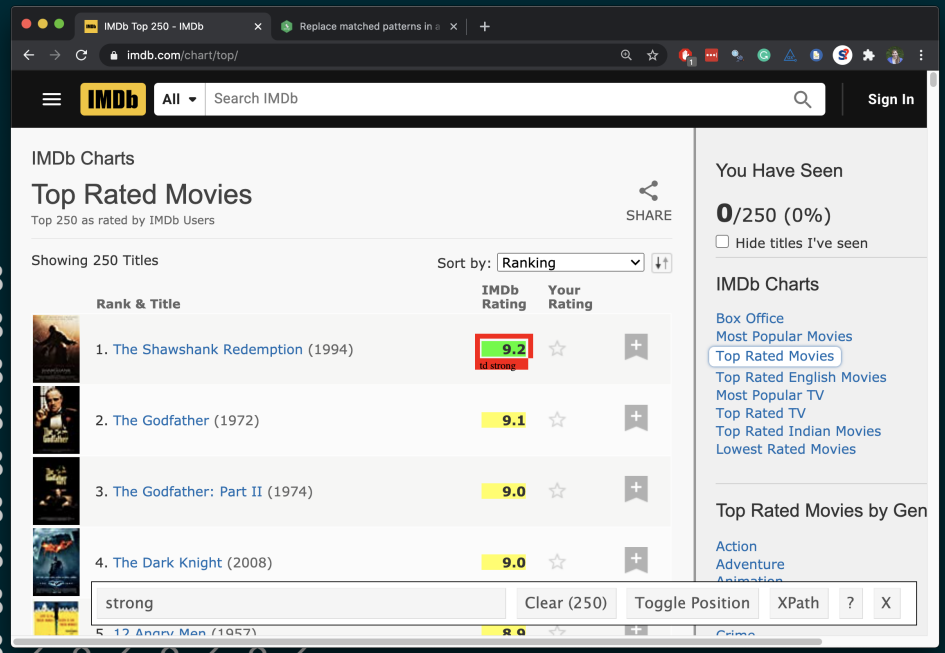
```
##   [1] "9.2" "9.1" "9.0" "9.0" "8.9" "8.9" "8.9" "8.
##  [11] "8.8" "8.7" "8.7" "8.7" "8.7" "8.6" "8.6" "8.
##  [21] "8.6" "8.6" "8.6" "8.6" "8.6" "8.6" "8.5" "8.
##  [31] "8.5" "8.5" "8.5" "8.5" "8.5" "8.5" "8.5" "8.
##  [41] "8.5" "8.5" "8.5" "8.5" "8.5" "8.5" "8.5" "8.
##  [51] "8.4" "8.4" "8.4" "8.4" "8.4" "8.4" "8.4" "8.
##  [61] "8.4" "8.4" "8.4" "8.4" "8.4" "8.4" "8.3" "8.
##  [71] "8.3" "8.3" "8.3" "8.3" "8.3" "8.3" "8.3" "8.
##  [81] "8.3" "8.3" "8.3" "8.3" "8.3" "8.3" "8.3" "8.
##  [91] "8.3" "8.3" "8.3" "8.3" "8.3" "8.3" "8.3" "8.3  8.3  8.3
## [101] "8.3" "8.3" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2"
## [111] "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2"
## [121] "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2"
## [131] "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.2"
## [141] "8.2" "8.2" "8.2" "8.2" "8.2" "8.2" "8.1" "8.1" "8.1" "8.1"
## [151] "8.1" "8.1" "8.1" "8.1" "8.1" "8.1" "8.1" "8.1" "8.1" "8.1"
...
```



Convert to numeric

```
page %>%
  html_nodes("strong") %>%
  html_text() %>%
  as.numeric()
```

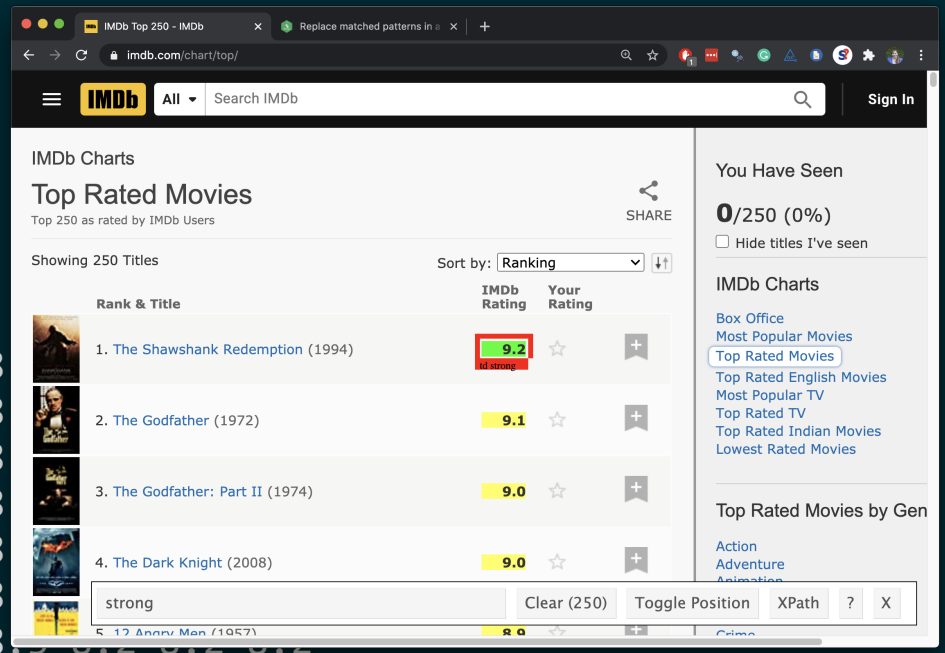
```
##   [1] 9.2 9.1 9.0 9.0 8.9 8.9 8.9 8.8 8.8 8.8 8.8 8
##  [16] 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8
##  [31] 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8
##  [46] 8.5 8.5 8.4 8.4 8.4 8.4 8.4 8.4 8.4 8.4 8.4 8
##  [61] 8.4 8.4 8.4 8.4 8.4 8.4 8.3 8.3 8.3 8.3 8.3 8
##  [76] 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8
##  [91] 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8
## [106] 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8
## [121] 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2
## [136] 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.1 8.1 8.1 8.1
## [151] 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1
## [166] 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1
## [181] 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1
## [196] 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1
## [211] 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.0 8.0 8.0 8.0
...
```



Save as `ratings`

```
ratings <- page %>%
  html_nodes("strong") %>%
  html_text() %>%
  as.numeric()

ratings
```

```
##   [1] 9.2 9.1 9.0 9.0 8.9 8.9 8.9 8.8 8.8 8.8 8.8 8
##  [16] 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8.6 8
##  [31] 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8
##  [46] 8.5 8.5 8.4 8.4 8.4 8.4 8.4 8.4 8.4 8.4 8.4 8
##  [61] 8.4 8.4 8.4 8.4 8.4 8.4 8.3 8.3 8.3 8.3 8.3 8
##  [76] 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8
##  [91] 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8.3 8
## [106] 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2
## [121] 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2
## [136] 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.2 8.1 8.1 8.1 8.1
## [151] 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1
## [166] 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1 8.1
...
```



Step 5. Create a data frame called `imdb_top_250`

Create a data frame: imdb_top_250

```
imdb_top_250 <- tibble(
  title = titles,
  year = years,
  rating = ratings
  )

imdb_top_250
```

```
## # A tibble: 250 x 3
##   title                     year rating
##   <chr>                    <dbl>  <dbl>
## 1 The Shawshank Redemption  1994    9.2
## 2 The Godfather             1972    9.1
## 3 The Godfather: Part II    1974    9
## 4 The Dark Knight           2008    9
## 5 12 Angry Men              1957    8.9
## 6 Schindler's List          1993    8.9
## # ... with 244 more rows
```

Previous 1 2 3 4 5 … 32 Next

	title	year	rating
1	The Shawshank Redemption	1994	9.2
2	The Godfather	1972	9.1
3	The Godfather: Part II	1974	9
4	The Dark Knight	2008	9
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	8.8

Clean up / enhance

May or may not be a lot of work depending on how messy the data are

- See if you like what you got:

```
glimpse(imdb_top_250)
```

```
## Rows: 250
## Columns: 3
## $ title  <chr> "The Shawshank Redemption", "The Godfather", "Th~
## $ year   <dbl> 1994, 1972, 1974, 2008, 1957, 1993, 2003, 1994, ~
## $ rating <dbl> 9.2, 9.1, 9.0, 9.0, 8.9, 8.9, 8.9, 8.8, 8.8, 8.8~
```

- Add a variable for rank

```
imdb_top_250 <- imdb_top_250 %>%
  mutate(rank = 1:nrow(imdb_top_250)) %>%
  relocate(rank)
```

```
## # A tibble: 250 x 4
##     rank title                                    year rating
##    <int> <chr>                                   <dbl>  <dbl>
##  1     1 The Shawshank Redemption                 1994    9.2
##  2     2 The Godfather                            1972    9.1
##  3     3 The Godfather: Part II                   1974    9
##  4     4 The Dark Knight                          2008    9
##  5     5 12 Angry Men                             1957    8.9
##  6     6 Schindler's List                         1993    8.9
##  7     7 The Lord of the Rings: The Return of the K~  2003    8.9
##  8     8 Pulp Fiction                             1994    8.8
##  9     9 The Good, the Bad and the Ugly           1966    8.8
## 10    10 The Lord of the Rings: The Fellowship of t~  2001    8.8
## 11    11 Fight Club                              1999    8.8
## 12    12 Forrest Gump                            1994    8.7
## 13    13 Inception                               2010    8.7
## 14    14 The Lord of the Rings: The Two Towers   2002    8.7
## 15    15 Star Wars: Episode V - The Empire Strikes ~  1980    8.7
## 16    16 The Matrix                              1999    8.6
## 17    17 Goodfellas                              1990    8.6
## 18    18 One Flew Over the Cuckoo's Nest         1975    8.6
## 19    19 Seven Samurai                           1954    8.6
## 20    20 Se7en                                   1995    8.6
## # ... with 230 more rows
```

What next?

Which years have the most movies on the list?

Which years have the most movies on the list?

```
imdb_top_250 %>%
  count(year, sort = TRUE)
```

```
## # A tibble: 86 x 2
##    year     n
##   <dbl> <int>
## 1  1995     8
## 2  1957     7
## 3  1994     6
## 4  1997     6
## 5  2000     6
## 6  2004     6
## # ... with 80 more rows
```

Which 1995 movies made the list?

Which 1995 movies made the list?

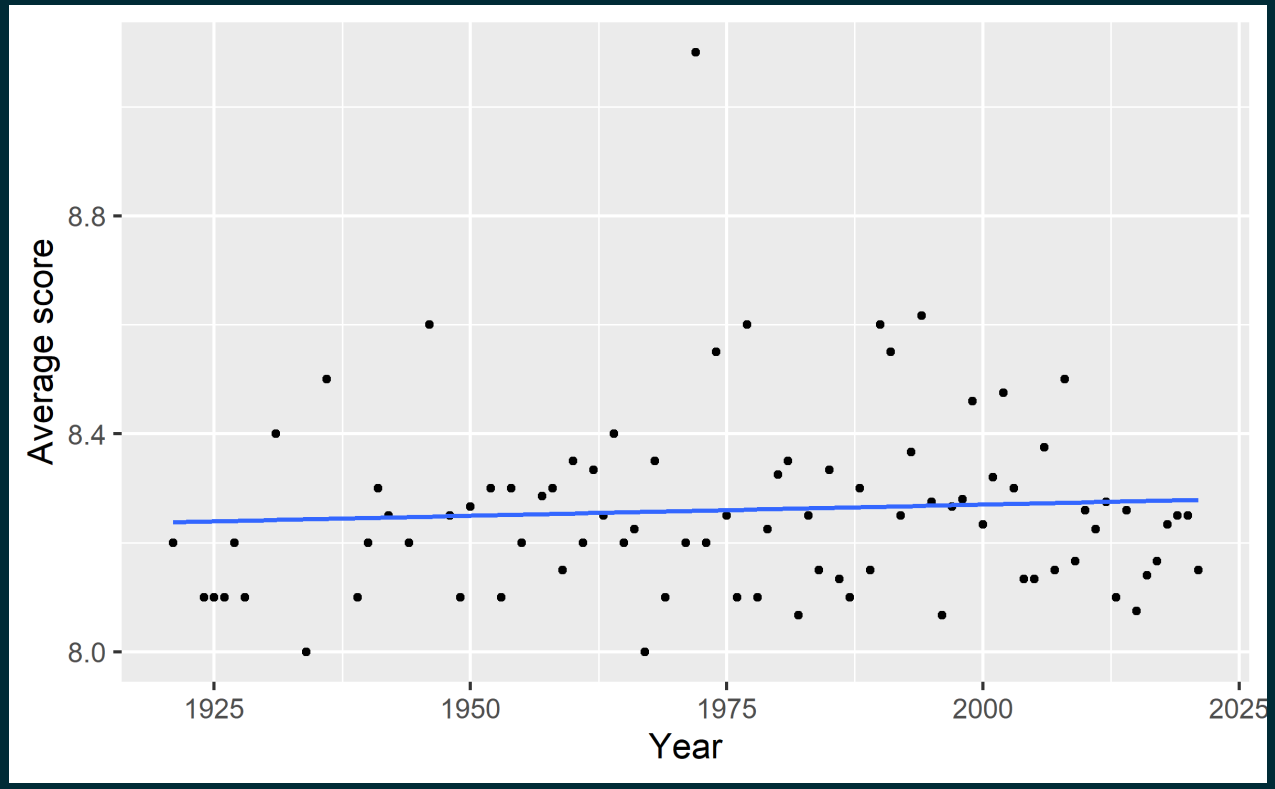
```
imdb_top_250 %>%
  filter(year == 1995) %>%
  print(n = 8)
```

```
## # A tibble: 8 x 4
##    rank title                year rating
##   <int> <chr>               <dbl>  <dbl>
## 1    20 Se7en                1995    8.6
## 2    35 The Usual Suspects   1995    8.5
## 3    81 Toy Story            1995    8.3
## 4    83 Braveheart           1995    8.3
## 5   125 Heat                 1995    8.2
## 6   145 Casino               1995    8.2
## 7   194 Before Sunrise       1995    8.1
## 8   224 La Haine             1995    8
```

Visualize the average yearly rating for movies that made it on the top 250 list over time.

Visualize the average yearly rating for movies that made it on the top 250 list over time.

Plot Code



Visualize the average yearly rating for movies that made it on the top 250 list over time.

Plot | Code

```
imdb_top_250 %>%
  group_by(year) %>%
  summarise(avg_score = mean(rating)) %>%
  ggplot(aes(y = avg_score, x = year)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Year", y = "Average score")
```

"Can you?" vs "Should you?"

Researchers just released profile data on 70,000 OkCupid users without permission

By Brian Resnick | @B_resnick | brian@vox.com | May 12, 2016, 6:00pm EDT

A group of researchers has released a data set on nearly 70,000 users of the online dating site OkCupid. The data dump breaks the cardinal rule of social science research ethics: It took identifiable personal data without permission.

The information — while publicly available to OkCupid users — was collected by Danish researchers who never contacted OkCupid or its clientele about using it.

The data, collected from November 2014 to March 2015, includes user names, ages, gender, religion, and personality traits, as well as answers to the personal questions the site asks to help match potential mates. The users hail from a few dozen countries around the world.

The data dump did not reveal anyone's real name. But it's entirely possible to use clues from a user's location, demographics, and OkCupid user name to determine their identity.

> If your OkC username is one you've used anywhere else, I now know your sexual preferences & kinks, your answers to thousands of questions.
>
> — Scott B. Weingart (@scott_bot) May 11, 2016

Source: Brian Resnick, Researchers just released profile data on 70,000 OkCupid users without permission, Vox.

"Can you?" vs "Should you?"

Emil OW Kirkegaard @KirkegaardEmil · May 8
The OKCupid paper has now been submitted. This means that the dataset is now public! Enjoy! :) openpsych.net/forum/showthre...

26 38

Ethan Jewett @esjewett · May 11
@KirkegaardEmil This data set is highly re-identifiable. Even includes usernames? Was any work at all done to anonymize it?

3 9

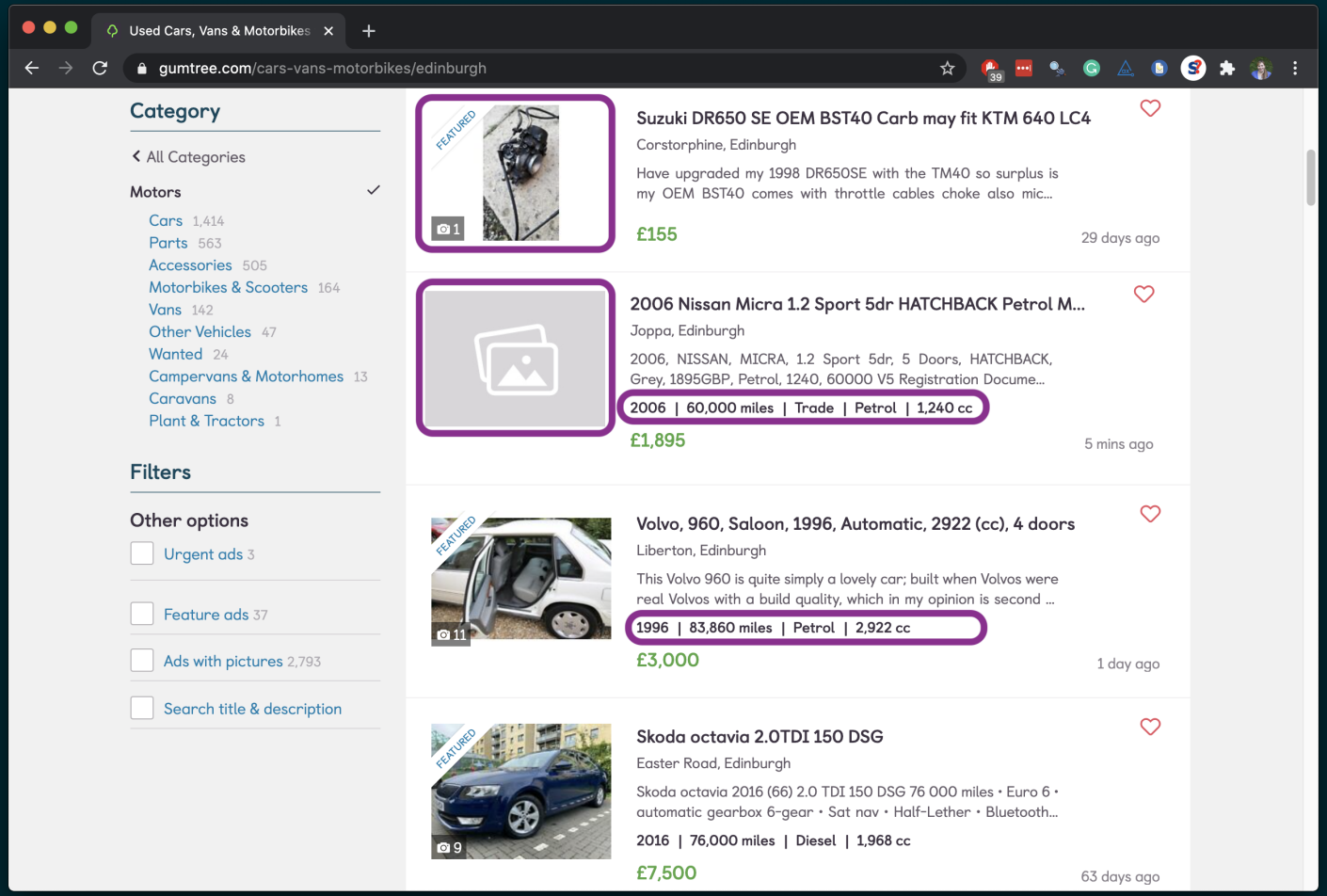
Emil OW Kirkegaard
@KirkegaardEmil

Follow

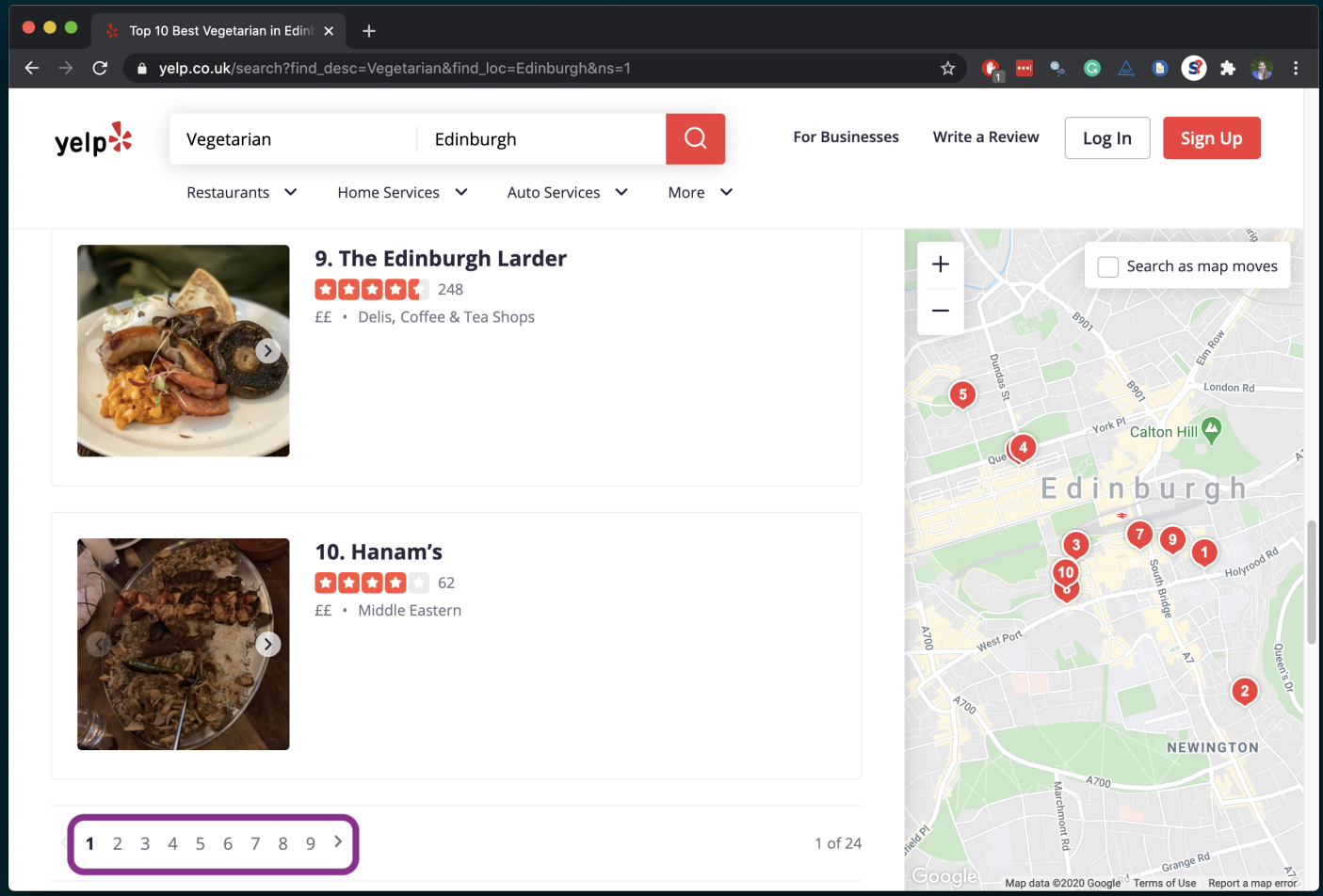
@esjewett No. Data is already public.

Challenges

Unreliable formatting at the source



Data broken into many pages



Workflow

Screen scraping vs. APIs

Two different scenarios for web scraping:

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

A new R workflow

- When working in an R Markdown document, your analysis is re-run each time you knit
- If web scraping in an R Markdown document, you'd be re-scraping the data each time you knit, which is undesirable (and not *nice*)!
- An alternative workflow:
  - Use an R script to save your code
  - Saving interim data scraped using the code in the script as CSV or RDS files
  - Use the saved data in your analysis in your R Markdown document