Section 001 CDS 101 *Introduction to Computational and Data Sciences*

Class 16: Introduction to Web Scraping

March 22, 2018

General

Annoucements

- Midterm Project Extension: due Tuesday, March 27, 2018 by 1:30pm
- Complete draft from each team must be uploaded to Github by Friday, March 27th at noon
    - All required sections: Data Tidying section, Exploratory Data Analysis section with 3 or 4 questions and answers
    - I will review these drafts and provide comments on Github, which you should use as advice for improving the report
    - **Mandatory:** Not having a complete draft uploaded in time can result in up to a 5% grade penalty on the midterm
    - Penalties will be proportional to how much of the draft is missing
- Midterm presentations on Tuesday, March 27th and Thursday, March 29th

Scraping 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'stime-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.
- Why R? It includes all tools necessary to do web scraping, familiarity, direct analysis of data... But python, perl, java are also efficient tools.

Web Scraping with rvest

Hypertext Markup Language

Most of the data on the web is still largely available as HTML - while it is structured (hierarchical / tree based) it often is not available in a form useful for analysis (flat / tidy).

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

rvest

`rvest` is a package from Hadley Wickham that makes basic processing and manipulation of HTML data straight forward.

Core functions:

- `read_html` - read HTML data from a url or character string.
- `html_nodes` - select specified nodes from the HTML document using CSS selectors.
- `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.

CSS selectors

We will be using a tool called selector gadget to help up identify the html elements of interest - it does this by constructing a css selector which can be used to subset the html document.

Selector	Example	Description
`element`	`p`	Select all <p> elements
`element element`	`div p`	Select all <p> elements inside a <div> element
`element>element`	`div > p`	Select all <p> elements with <div> as a parent
`.class`	`.title`	Select all elements with class="title"
`\#id`	`.name`	Select all elements with id="name"
`[attribute]`	`[class]`	Select all elements with a class attribute
`[attribute=value]`	`[class=title]`	Select all elements with class="title"

SelectorGadget

- SelectorGadget: Open source tool that eases CSS selector generation and discovery
- Install the Chrome Extension
- A box will open in the bottom right of the website. Click on a page element that you would like your selector to match (it will turn green). SelectorGadget will then generate a minimal CSS selector for that element, and will highlight (yellow) everything that is matched by the selector.
- Now click on a highlighted element to remove it from the selector (red), or click on an unhighlighted element to add it to the selector. Through this process of selection and rejection, SelectorGadget helps you come up with the appropriate CSS selector for your needs.

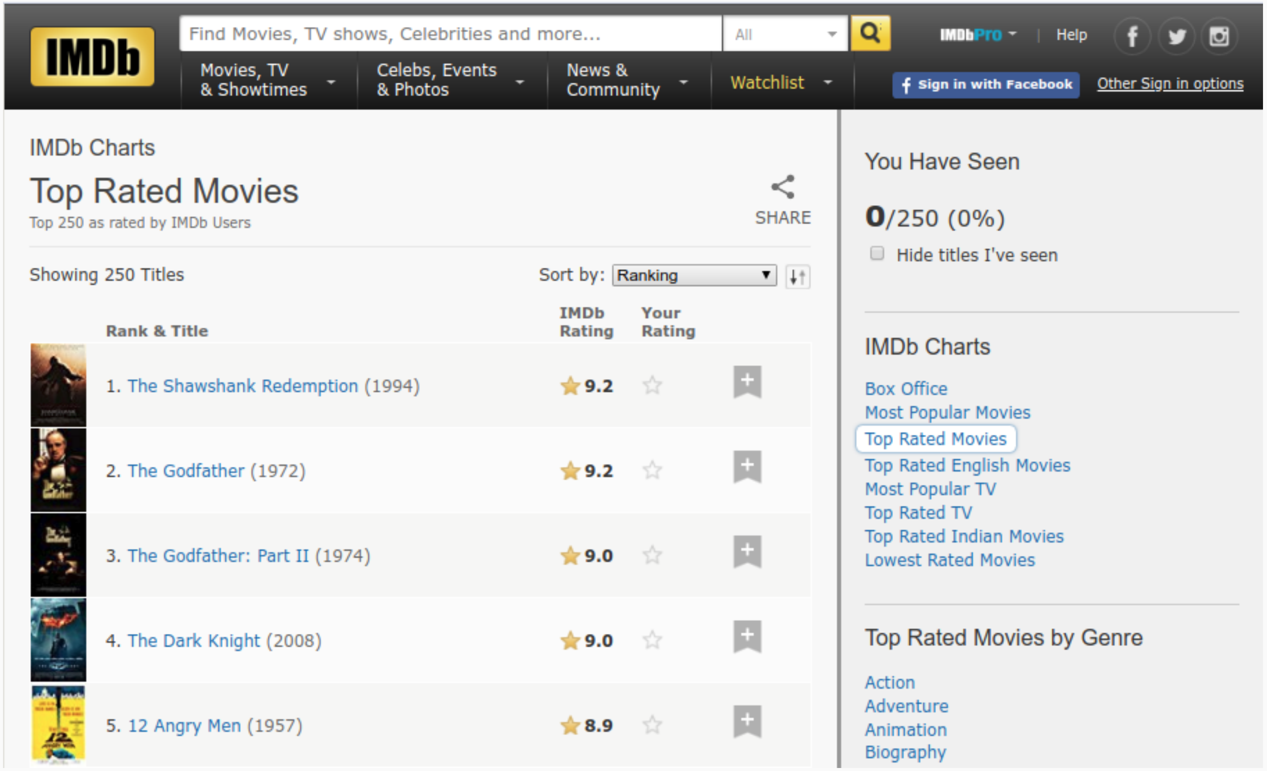
```r
vignette("selectorgadget")
```

Top 250 movies on IMDB

Top 250 movies on IMDB

Take a look at the source code, look for the tag `table` tag:

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



IMDb Charts

Top Rated Movies

Top 250 as rated by IMDb Users

Showing 250 Titles   Sort by: Ranking

	Rank & Title	IMDb Rating	Your Rating
	1. The Shawshank Redemption (1994)	9.2	
	2. The Godfather (1972)	9.2	
	3. The Godfather: Part II (1974)	9.0	
	4. The Dark Knight (2008)	9.0	
	5. 12 Angry Men (1957)	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

First check to make sure you're allowed!

```r
# install.packages("robotstxt")
library(robotstxt)
paths_allowed("http://www.imdb.com")
```

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

Select and format pieces

```r
library(rvest)

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

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

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

scores <- page %>%
  html_nodes(".article strong") %>%
  html_text() %>%
  as.numeric()

imdb_top_250 <- data_frame(
  title = titles, year = years, score = scores)
```

IMDB Scraped Table

title	year	score
The Shawshank Redemption	1994	9.2
The Godfather	1972	9.2
The Godfather: Part II	1974	9
The Dark Knight	2008	9
12 Angry Men	1957	8.9
Schindler's List	1993	8.9
The Lord of the Rings: The Return of the King	2003	8.9
Pulp Fiction	1994	8.9
The Good, the Bad and the Ugly	1966	8.8
Fight Club	1999	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:

```
str(imdb_top_250)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    250 obs. of  3 variables:
##  $ title: chr  "The Shawshank Redemption" "The Godfather" "The Godfath
##  $ year : num  1994 1972 1974 2008 1957 ...
##  $ score: num  9.2 9.2 9 9 8.9 8.9 8.9 8.9 8.8 8.8 ...
```

- Add a variable for rank

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

IMDB Scraped Table (Updated)

title	year	score	rank
The Shawshank Redemption	1994	9.2	1
The Godfather	1972	9.2	2
The Godfather: Part II	1974	9	3
The Dark Knight	2008	9	4
12 Angry Men	1957	8.9	5
Schindler's List	1993	8.9	6
The Lord of the Rings: The Return of the King	2003	8.9	7
Pulp Fiction	1994	8.9	8
The Good, the Bad and the Ugly	1966	8.8	9
Fight Club	1999	8.8	10
…	…	…	…

Analyze

How would you go about answering this question: Which 1995 movies made the list?

Analyze

How would you go about answering this question: Which 1995 movies made the list?

```
imdb_top_250 %>%
  filter(year == 1995)
```

```
## # A tibble: 9 x 4
##   title                year score  rank
##   <chr>               <dbl> <dbl> <int>
## 1 Se7en                1995  8.60    22
## 2 The Usual Suspects   1995  8.60    26
## 3 Braveheart           1995  8.30    74
## 4 Toy Story            1995  8.30    93
## 5 Heat                 1995  8.20   123
## 6 Casino               1995  8.20   145
## 7 Before Sunrise       1995  8.10   207
## 8 La Haine             1995  8.00   230
## 9 Twelve Monkeys       1995  8.00   242
```

Analyze

How would you go about answering this question: Which years have the most movies on the list?

Analyze

How would you go about answering this question: Which years have the most movies on the list?

```
imdb_top_250 %>%
  count(total = year) %>%
  arrange(desc(total)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   total     n
##   <dbl> <int>
## 1  2017     4
## 2  2016     4
## 3  2015     5
## 4  2014     6
## 5  2013     4
```

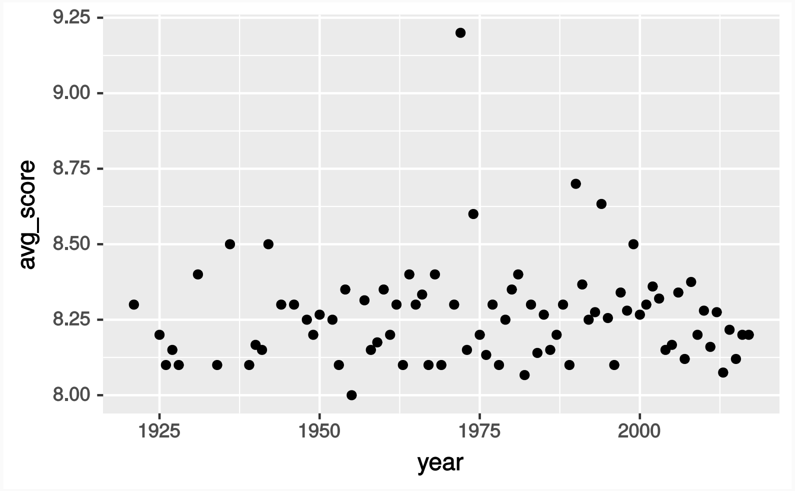
Visualize

How would you go about creating this visualization: Visualize the average yearly score for movies that made it on the top 250 list over time.

Visualize

How would you go about creating this visualization: Visualize the average yearly score for movies that made it on the top 250 list over time.

```
imdb_top_250 %>%
  group_by(year) %>%
  summarize(avg_score = mean(score)) %>%
  ggplot() +
    geom_point(mapping = aes(x = year, y = avg_score))
```



Potential challenges

- Unreliable formatting at the source
- Data broken into many pages
- Too many tables/structures that vary

Credits