Section A01
CDS 101 *Introduction to Computational and Data Sciences*

Class 12: Web Scraping

June 6, 2018

General

Announcements

- Homework 2 due tonight @ 11:59pm: http://summer18.cds101.com/assignments/homework-2/
- Complete Reading 10 on web scraping and submit questions by 9:00am on Friday, June 8th
- Homework 3 on web scraping to be posted soon, will be due by 11:59pm on Tuesday, June 12th
- Be prepared to share and discuss your proposed questions for the Midterm Project on Friday, June 8th

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:

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).

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.

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.
- Why R? It includes all tools necessary to do web scraping, familiarity, direct analysis of data... But python, perl, java, and javascript 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).

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>
  <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.

rvest

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

Core functions:

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.

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.

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.

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.

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.

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.

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 element
`element>element`	`div > p`	Select all <p> elements with 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

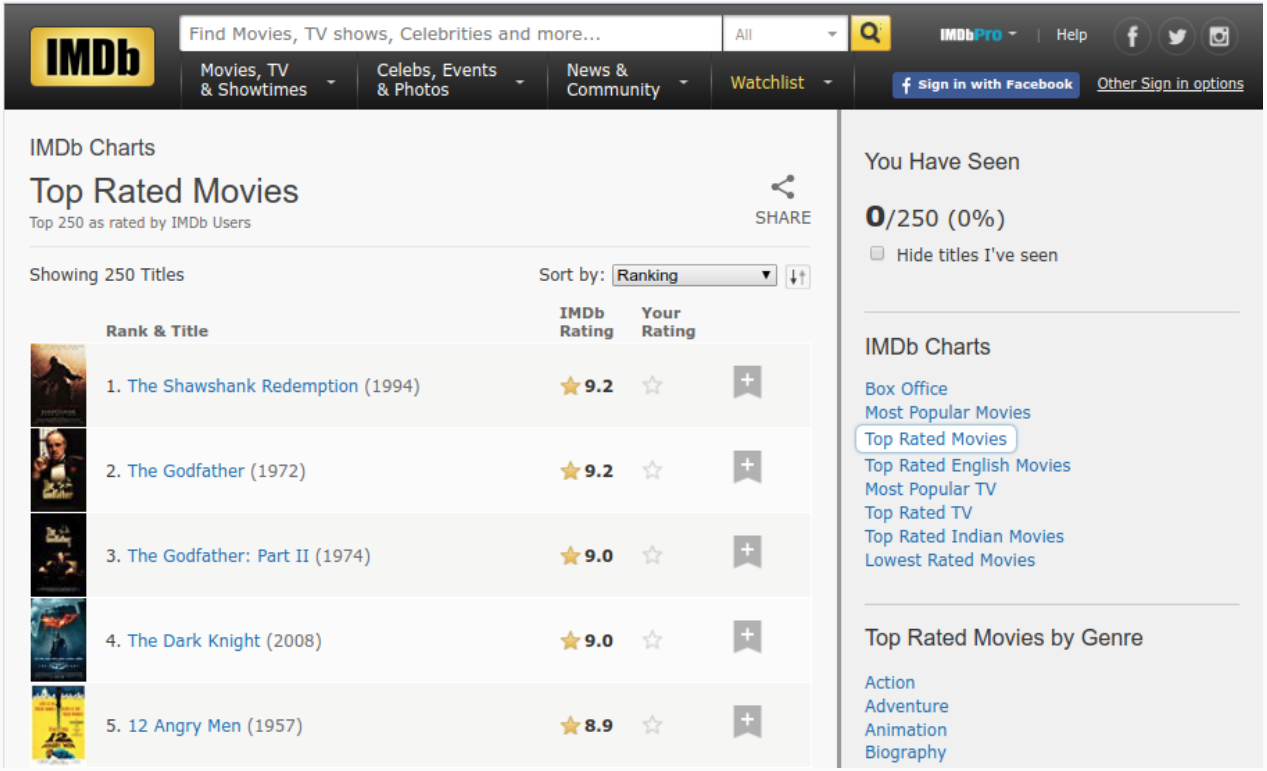
- SelectorGadget: Open source tool that eases CSS selector generation and discovery
- Install the Chrome Extension
- A box will open in the bottom right of the website. Click on a page element that you would like your selector to match (it will turn green). SelectorGadget will then generate a minimal 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.

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



First check to make sure you're allowed!

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

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

Fetch HTML page and save to disk

```
read_html("http://www.imdb.com/chart/top") %>%
  write_html("imdb_top_250.html")
```

- It's recommended that you fetch the HTML for the Top 250 Movies page once and then save it for offline use.
- Two reasons you would want to do this:
    - Being a good internet citizen: you want to avoid "asking" for the same HTML page over and over again, as this places stress on the webserver and in the most extreme cases it can make it crash
    - Reproducibility: web pages are frequently updating and changing, so by taking a snapshot you ensure that you can reproduce your results

Select and format pieces

```
page <- read_html("imdb_top_250.html")  # Load and parse saved HTML file

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

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

scores <- page %>%
  html_nodes("#main strong") %>%        # ".article strong" also works
  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

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:

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)
```

```
## Observations: 250
## Variables: 3
## $ title <chr> "The Shawshank Redemption", "The Godfather", "The Godfat
## $ year  <dbl> 1994, 1972, 1974, 2008, 1957, 1993, 2003, 1994, 1966, 19
## $ score <dbl> 9.2, 9.2, 9.0, 9.0, 8.9, 8.9, 8.9, 8.9, 8.8, 8.8, 8.8, 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)
```

```
## Observations: 250
## Variables: 3
## $ title <chr> "The Shawshank Redemption", "The Godfather", "The Godfat
## $ year  <dbl> 1994, 1972, 1974, 2008, 1957, 1993, 2003, 1994, 1966, 19
## $ score <dbl> 9.2, 9.2, 9.0, 9.0, 8.9, 8.9, 8.9, 8.9, 8.8, 8.8, 8.8, 8
```

- Add a variable for rank

```
imdb_top_250 <- imdb_top_250 %>%
  mutate(rank = row_number())
```

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)
```

title	year	score	rank
Se7en	1995	8.6	23
The Usual Suspects	1995	8.6	27
Braveheart	1995	8.3	75
Toy Story	1995	8.3	93
Heat	1995	8.2	122
Casino	1995	8.2	145
Before Sunrise	1995	8.1	208
La Haine	1995	8.0	230
Twelve Monkeys	1995	8.0	250

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(year) %>%
  arrange(desc(n)) %>%
  head(5)
```

year	n
1995	9
1957	7
2014	7
2000	6
2001	6

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.

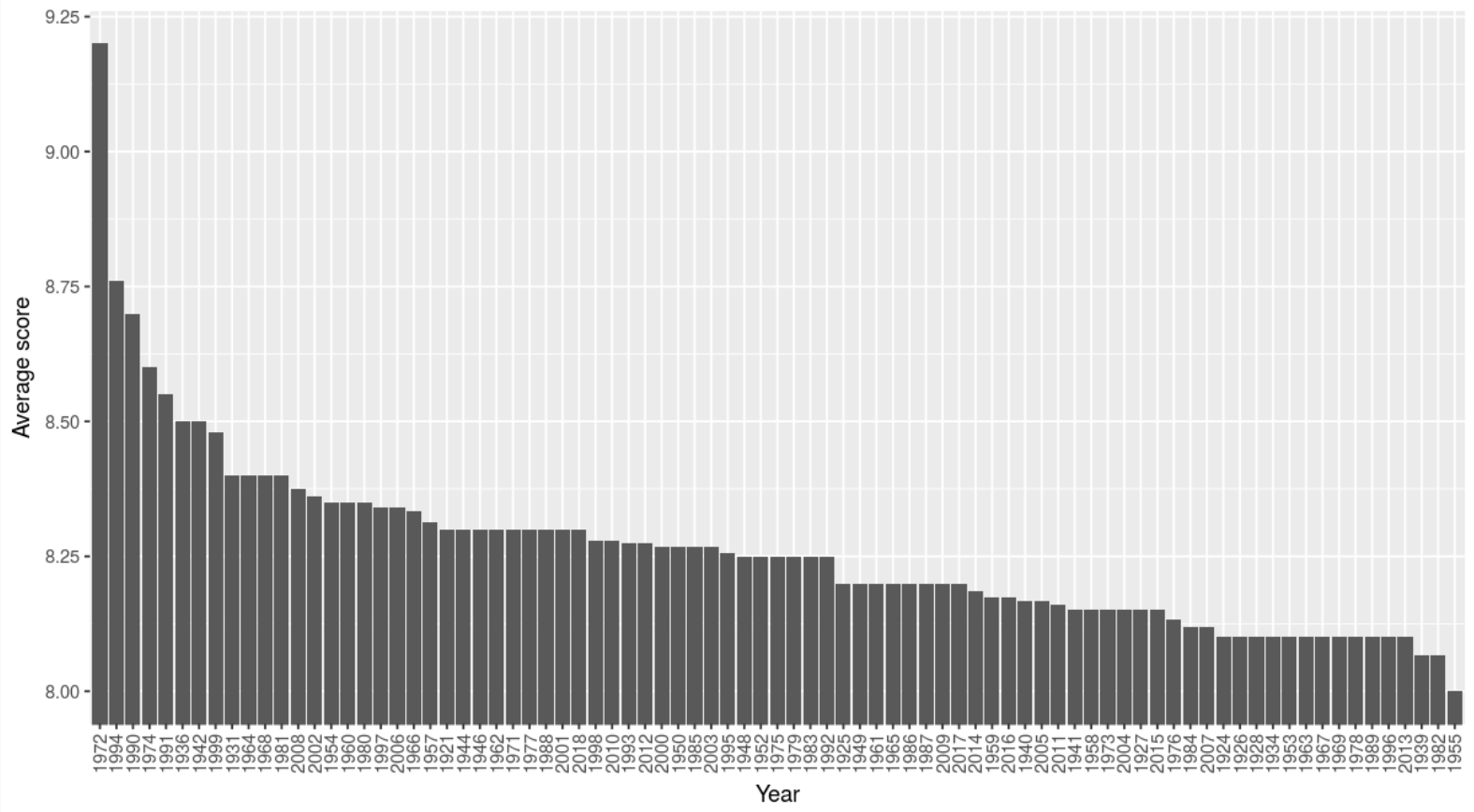
Visualize

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 %>%
  mutate_at(vars(year), as.character) %>%      # Convert the year column to the
  group_by(year) %>%                           #   character data type
  summarize(avg_score = mean(score)) %>%
  ggplot() +
  geom_col(
    mapping = aes(
      x = fct_reorder(year, desc(avg_score)),  #  Sort year using avg_score
      y = avg_score
    )
  ) +
  labs(
    x = "Year",
    y = "Average score"
  ) +
  coord_cartesian(ylim = combine(8.0, 9.2)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

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.



Potential challenges

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

Credits

These slides were adapted from the Web Scraping slides developed by Mine Çetinkaya-Rundel and made available under the CC BY license.