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

Class 13: Introduction to Web Scraping II

June 7, 2018

General

Annoucements

- Reading 10 on web scraping posted, submit questions by 9:00am on Friday, June 8th
- Homework 3 on web scraping posted, 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

Web Scraping Activity

Web scraping activity

Navigate to http://www.imdb.com/chart/tvmeter and scrape the list of the most popular TV shows. The result should be a tibble with 100 rows and 4 columns: rank, tv show name, year, and rating. The variables should be in this order.

Web scraping activity

Navigate to http://www.imdb.com/chart/tvmeter and scrape the list of the most popular TV shows. The result should be a tibble with 100 rows and 4 columns: rank, tv show name, year, and rating. The variables should be in this order.

- The code blocks from the Top 250 Movies example will work for some, but not all of this exercise.

Web scraping activity

Navigate to http://www.imdb.com/chart/tvmeter and scrape the list of the most popular TV shows. The result should be a tibble with 100 rows and 4 columns: rank, tv show name, year, and rating. The variables should be in this order.

- The code blocks from the Top 250 Movies example will work for some, but not all of this exercise.
- Primary objective is to use the SelectorGadget tool to modify the HTML nodes you need to grab

Web scraping activity

Navigate to http://www.imdb.com/chart/tvmeter and scrape the list of the most popular TV shows. The result should be a tibble with 100 rows and 4 columns: rank, tv show name, year, and rating. The variables should be in this order.

- The code blocks from the Top 250 Movies example will work for some, but not all of this exercise.
- Primary objective is to use the SelectorGadget tool to modify the HTML nodes you need to grab
- How do you take the example code and modify it to work for this activity?

Scraping code: IMDB Top 250 Movies

```r
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_remove("\\(") %>% # remove (
  str_remove("\\)") %>% # remove )
  as.numeric()

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

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

Scraping code: IMDB Top 250 Movies

```r
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_remove("\\(") %>% # remove (
  str_remove("\\)") %>% # remove )
  as.numeric()

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

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

Scraping code: IMDB Top 250 Movies

```r
page <- read_html("http://www.imdb.com/chart/tvmeter")

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") %>%
  html_text() %>%
  as.numeric()

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

Scraping code: IMDB Top 250 Movies

```r
page <- read_html("http://www.imdb.com/chart/tvmeter")

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") %>%
  html_text() %>%
  as.numeric()

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

TV show titles

Let's check to see if it's actually necessary to change the `titles` code:

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

TV show titles

Let's check to see if it's actually necessary to change the `titles` code:

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

The length of the `titles` vector is:

```r
length(titles)
```

```
## [1] 100
```

TV show titles

Let's check to see if it's actually necessary to change the `titles` code:

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

The length of the `titles` vector is:

```
length(titles)
```

```
## [1] 100
```

And the first 10 elements in `titles` are:

```
##  [1] "13 Reasons Why"      "Westworld"           "The Handmaid's Tale"
##  [4] "Game of Thrones"     "Grey's Anatomy"      "Lucifer"
##  [7] "Riverdale"           "Brooklyn Nine-Nine"  "Suits"
## [10] "Supernatural"
```

TV show titles

Let's check to see if it's actually necessary to change the `titles` code:

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

The length of the `titles` vector is:

```r
length(titles)
```

```
## [1] 100
```

And the first 10 elements in `titles` are:

```
##  [1] "13 Reasons Why"      "Westworld"           "The Handmaid's Tale"
##  [4] "Game of Thrones"     "Grey's Anatomy"      "Lucifer"
##  [7] "Riverdale"           "Brooklyn Nine-Nine"  "Suits"
## [10] "Supernatural"
```

So far, so good!

TV show years

```r
page <- read_html("http://www.imdb.com/chart/tvmeter")

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") %>%
  html_text() %>%
  as.numeric()

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

TV show years

Next, let's check if the `years` code works for us:

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

TV show years

Next, let's check if the `years` code works for us:

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

And the first few elements in `years` are:

```
## [1] "2017"   "2016"   "2017"   "\n\n1" "2011"   "\n\n2" "2005"   "2015"
```

TV show years

Next, let's check if the `years` code works for us:

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

And the first few elements in `years` are:

```
## [1] "2017"   "2016"   "2017"   "\n\n1" "2011"   "\n\n2" "2005"   "2015"
```

Not so lucky this time.

TV show years

Next, let's check if the `years` code works for us:

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

And the first few elements in `years` are:

```
## [1] "2017"   "2016"   "2017"   "\n\n1" "2011"   "\n\n2" "2005"   "2015"
```

Not so lucky this time. Let's see how we can fix this.

SelectorGadget `years` demo

Follow along in Google Chrome

TV show years (revised)

Here's our revised `years` code based on our SelectorGadget work:

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

TV show years (revised)

Here's our revised `years` code based on our SelectorGadget work:

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

The first 10 elements in our revised `years` are:

```
##  [1] "2017" "2016" "2017" "2011" "2005" "2015" "2016" "2013" "2011" "2
```

TV show years (revised)

Here's our revised `years` code based on our SelectorGadget work:

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

The first 10 elements in our revised `years` are:

```
##  [1] "2017" "2016" "2017" "2011" "2005" "2015" "2016" "2013" "2011" "2
```

Much better!

Note: We should append `%>% as.numeric()` to our `years` definition so that the years are interpreted by R as integers, not text.

TV show user scores

```r
page <- read_html("http://www.imdb.com/chart/tvmeter")

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

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

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

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

TV show user scores

Will the `scores` code work?

```r
scores <- page %>%
  html_nodes("#main strong") %>%
  html_text() %>%
  as.numeric()
```

TV show user scores

Will the `scores` code work?

```r
scores <- page %>%
  html_nodes("#main strong") %>%
  html_text() %>%
  as.numeric()
```

The first 10 elements in `scores` are:

```
##  [1] 8.2 8.9 8.6 9.5 7.6 8.2 7.7 8.3 8.6 8.5
```

TV show user scores

Will the `scores` code work?

```
scores <- page %>%
  html_nodes("#main strong") %>%
  html_text() %>%
  as.numeric()
```

The first 10 elements in `scores` are:

```
##  [1] 8.2 8.9 8.6 9.5 7.6 8.2 7.7 8.3 8.6 8.5
```

As of right now, this is working as expected if we check the number of elements in `scores`:

```
length(scores)
```

```
## [1] 100
```

TV show user scores

Will the `scores` code work?

```r
scores <- page %>%
  html_nodes("#main strong") %>%
  html_text() %>%
  as.numeric()
```

The first 10 elements in `scores` are:

```
##  [1] 8.2 8.9 8.6 9.5 7.6 8.2 7.7 8.3 8.6 8.5
```

As of right now, this is working as expected if we check the number of elements in `scores`:

```r
length(scores)
```

```
## [1] 100
```

However...

TV show user scores

Will the `scores` code work?

```
scores <- page %>%
  html_nodes("#main strong") %>%
  html_text() %>%
  as.numeric()
```

The first 10 elements in `scores` are:

```
##  [1] 8.2 8.9 8.6 9.5 7.6 8.2 7.7 8.3 8.6 8.5
```

As of right now, this is working as expected if we check the number of elements in `scores`:

```
length(scores)
```

```
## [1] 100
```

However... just a couple of months ago, I got `99` instead of `100` when running this code.

TV show user scores

Will the `scores` code work?

```
scores <- page %>%
  html_nodes("#main strong") %>%
  html_text() %>%
  as.numeric()
```

The first 10 elements in `scores` are:

```
##  [1] 8.2 8.9 8.6 9.5 7.6 8.2 7.7 8.3 8.6 8.5
```

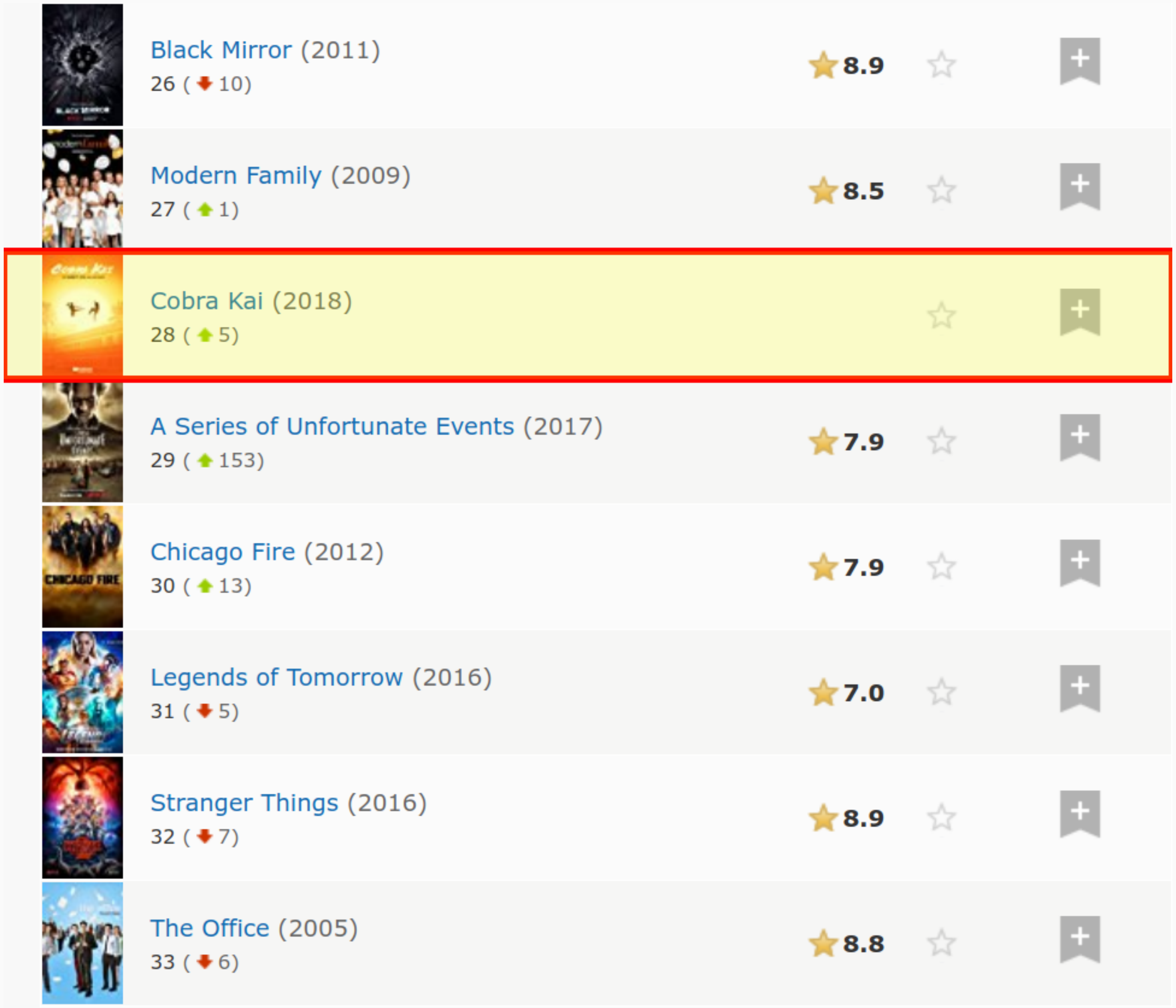
As of right now, this is working as expected if we check the number of elements in `scores`:

```
length(scores)
```

```
## [1] 100
```

However... just a couple of months ago, I got `99` instead of `100` when running this code. Why would that happen?

Blank TV show scores



Black Mirror (2011)	26 (▼10)	8.9
Modern Family (2009)	27 (▲1)	8.5
Cobra Kai (2018)	28 (▲5)	
A Series of Unfortunate Events (2017)	29 (▲153)	7.9
Chicago Fire (2012)	30 (▲13)	7.9
Legends of Tomorrow (2016)	31 (▼5)	7.0
Stranger Things (2016)	32 (▼7)	8.9
The Office (2005)	33 (▼6)	8.8

SelectorGadget `scores` demo

Follow along in Google Chrome

TV show user scores (revised)

Here's our revised `scores` code based on our SelectorGadget work that will take into account shows that may have a missing score:

```r
scores <- page %>%
  html_nodes(".imdbRating") %>%
  html_text() %>%
  as.numeric()
```

TV show user scores (revised)

Here's our revised `scores` code based on our SelectorGadget work that will take into account shows that may have a missing score:

```r
scores <- page %>%
  html_nodes(".imdbRating") %>%
  html_text() %>%
  as.numeric()
```

The first 10 elements in our revised `scores` are:

```
##  [1] 8.2 8.9 8.6 9.5 7.6 8.2 7.7 8.3 8.6 8.5
```

TV show user scores (revised)

Here's our revised `scores` code based on our SelectorGadget work that will take into account shows that may have a missing score:

```
scores <- page %>%
  html_nodes(".imdbRating") %>%
  html_text() %>%
  as.numeric()
```

The first 10 elements in our revised `scores` are:

```
##  [1] 8.2 8.9 8.6 9.5 7.6 8.2 7.7 8.3 8.6 8.5
```

That hasn't changed, and the number of elements in `scores` is:

```
length(scores)
```

```
## [1] 100
```

Creating the data tibble

```r
page <- read_html("http://www.imdb.com/chart/tvmeter")

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

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

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

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

TV show rank

The shows on the page are already sorted by rank.

TV show rank

The shows on the page are already sorted by rank.

So we can just use the row numbers to create the rank column:

```
imdb_top_tv <- data_frame(
  title = titles,
  year = years,
  score = scores
) %>%
  mutate(rank = row_number())
```

TV show tibble

We have everything we need, so let's take the original code for making the tibble:

TV show tibble

We have everything we need, so let's take the original code for making the tibble:

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

TV show tibble

We have everything we need, so let's take the original code for making the tibble:

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

and change the variable name to `imdb_top_tv`, put the columns in the correct order, and add in the ranks column:

TV show tibble

We have everything we need, so let's take the original code for making the tibble:

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

and change the variable name to `imdb_top_tv`, put the columns in the correct order, and add in the ranks column:

```
imdb_top_tv <- data_frame(
  title = titles, year = years, score = scores) %>%
  mutate(rank = row_number()) %>%
  select(rank, title, year, score)
```

Create a RDS file

Finally, let's save our work so that we don't need to always reconnect to the website:

Create a RDS file

Finally, let's save our work so that we don't need to always reconnect to the website:

```
imdb_top_tv %>%
  write_rds("2018-06-08T2035EST_imdb_tv.rds", compress = "gz")
```

Create a RDS file

Finally, let's save our work so that we don't need to always reconnect to the website:

```
imdb_top_tv %>%
  write_rds("2018-06-08T2035EST_imdb_tv.rds", compress = "gz")
```

Notice that the date and time that the data was scraped is part of the filename.

Create a RDS file

Finally, let's save our work so that we don't need to always reconnect to the website:

```
imdb_top_tv %>%
  write_rds("2018-06-08T2035EST_imdb_tv.rds", compress = "gz")
```

Notice that the date and time that the data was scraped is part of the filename.

The list on this webpage changes frequently, so this documents when the scraping occured!

Complete scraping code

```r
page <- read_html("http://www.imdb.com/chart/tvmeter")

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

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

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

imdb_top_tv <- data_frame(
  title = titles, year = years, score = scores) %>%
  mutate(rank = row_number()) %>%
  select(rank, title, year, score)
```

IMDB TV Table

rank	title	year	score
1	13 Reasons Why	2017	8.2
2	Westworld	2016	8.9
3	The Handmaid's Tale	2017	8.6
4	Game of Thrones	2011	9.5
5	Grey's Anatomy	2005	7.6
6	Lucifer	2015	8.2
7	Riverdale	2016	7.7
8	Brooklyn Nine-Nine	2013	8.3
9	Suits	2011	8.6
10	Supernatural	2005	8.5
...	...	...	...

Credits

These slides were adapted from the following sources:

- The Web Scraping slides and Mini HW 12 - Web Scraping assignment developed by Mine Çetinkaya-Rundel and made available under the CC BY 4.0 license.