Notes Webscraping and Ethics
DATA 110
Saidi

- Ethics of Mining and Web Scraping
- Webscraping tutorial
- Homework

Ethics of Mining and Web Scraping

We will use two tutorials to learn a bit about mining for data and scraping the web for data. In both cases, the overall goal is to create our own dataset, based on information we can glean directly from sources. The downside is that, if used improperly, we could violate ethics by revealing private information.

Example of Questionable Use of Data Scraping

(from https://www.vox.com/2016/5/12/11666116/70000-okcupid-users-data-release)

In May 2016, the online OpenPsych Forum published a paper by Kirkegaard and Bjerrekær (2016) titled "The OkCupid data set: A very large public data set of dating site users." The resulting data set contained nearly 70,000 users with 2,620 variables scraped from OkCupid dating website. Variables included usernames, gender, dating preferences, religiosity, political preferences, etc. The purpose of the data dump was to provide an open public data set to fellow researchers, and it could possibly be used to answer questions such as this one suggested in the abstract of the paper: whether the zodiac sign of each user was associated with any of the other variables (spoiler alert: it wasn't).

The data scraping did not involve any illicit technology such as breaking passwords.

"But the data set reveals deeply personal information about many of the users. OkCupid uses a series of personal questions — on topics such as sexual habits, politics, fidelity, feelings on homosexuality, etc. — to help match people on the site. 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." (Vox)

Nonetheless, the author received many comments on the OpenPsych Forum challenging the work as an ethical breach and accusing him of doxing people by releasing personal data.

Web Scraping Tutorial

R for Web Scraping

We will use Web Scraping with R with a Chrome extension called SelectorGadget. See below.

Web Scraping can be extremely useful (and fun) to extract information from a web page that uses html and css coding to extract that page's information and turn it in to a dataset.

1. Click on this link to follow how to scrape the web for html information [Beginner's Guide on Web Scraping in R (using rvest) with hands-on example]

   - You will install "rvest" on R
   - Download the Chrome extension from SelectorGadget from here. You need to get the Selector Gadget Chrome Extension from the tutorial under the "Pre-Requisites", under the "install.packages(rvest) bullet.

Be sure that once you have loaded the Chrome extension, you see this symbol on the top right of your Chrome page

Watch this video on how to use SelectorGadget.

By the way, *HTML*, HyperText Markup Language, gives content structure and meaning by defining that content as, for example, headings, paragraphs, or images. *CSS*, or Cascading Style Sheets, is a presentation language created to style the appearance of content—using, for example, fonts or colors.

The two languages—HTML and CSS—are independent of one another. As a rule, HTML will always represent content, and CSS will always represent the appearance of that content.

2. Follow step 4 for Web Scraping the movie rankings
   (Believe me – this is fun, AND it works!!!!)

4. Scraping a webpage using R

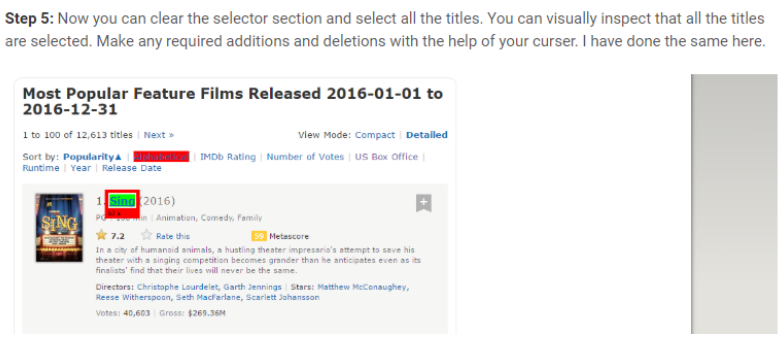
Now, let's get started with scraping the IMDb website for the 100 most popular feature films released in 2016. You can access them here.

```
#Loading the rvest package
library('rvest')

#Specifying the url for desired website to be scraped
url <- 'http://www.imdb.com/search/title?count=100&release_date=2016,2016&title_type=feature'

#Reading the HTML code from the website
webpage <- read_html(url)
```

3. Follow steps 5 and 6 for web scraping for movie titles.



Have fun following steps 7-11 for Description, Runtime, Genre, Rating, Metascore, Votes, Gross_Earning_in_Mil , Director and Actor data. Try to complete all sections so that you can create the data frame to then analyze the data and create data visualizations.

Important **

1. In steps 7-10, you will collect information that will have missing data. IT WILL BE DIFFERENT VALUES FROM THOSE THAT THE TUTORIAL PROVIDES. This is because the website is updated daily.

   See code below for a fix for the **metascores** *error and the* **gross earnings.**

2. Feel free to create the plots using ggplot instead of qplot. Also feel free to make the ggplots interactive using plotly. You will have to answer the 3 questions at the end using some filtering techniques.

   You will need to create additional code to filter to get the **EXACT ANSWERS** to the 3 questions. Simply using the plots will **NOT GIVE YOU ALL THE INFORMATION YOU NEED TO ANSWER THE 3 QUESTIONS.**

Dealing with cleaning character strings using stringr

The code below will give you a bit of understanding of how to deal with cleaning character strings.

```
## Find metascore data with missing values and replace with NAs (this is an automated method)
```

```
{r}

ratings_bar_data <- html_nodes(webpage,'.ratings-bar') %>%
# scrape the ratings bar and convert to text

  html_text2()

head(ratings_bar_data)                                  # look at the ratings bar
```

```
metascore_data <- str_match(ratings_bar_data, "\\d{2} Metascore") %>%
# extract Metascore
  str_match("\\d{2}") %>%
  as.numeric()                                          # convert to number
length(metascore_data)
metascore_data
summary(metascore_data)
```

```
## Find the missing gross earnings (automated) Earnings are part of the votes bar in the html, scrape the votes bar and extract earnings with a regular expression to get the NAs in context.
```

```
{r}
# scrape the votess bar and convert to text
votes_bar_data <- html_nodes(webpage,'.sort-num_votes-visible') %>%
  html_text2()
head(votes_bar_data)                                    # look at the votes bar data
gross_data <- str_match(votes_bar_data, "\\$.+$")        # extract the gross earnings
gross_data <- gsub("M","",gross_data)                    # clean data: remove 'M' sign
gross_data <- substring(gross_data,2,6) %>%              # clean data: remove '$' sign
  as.numeric()
length(gross_data)
```

Week 9 Homework Assignment

(Worth up to 15 points) Click on Beginner's Guide on Web Scraping in R (using rvest) with hands-on example

1. Follow all steps on this site, copying and pasting the code into your own markdown file.
2. Create the 3 graphs from the tutorial, THEN filter for additional details to answer the 3 questions at the end of the tutorial.
3. Once you have tested your code and it works, knit and publish your work.

When completing the assignment, be aware of the pitfalls I mention in the notes and how to deal with them.

Submit the completed tutorial by **11:59 pm on Tuesday, ___ .**