Problem Set - 2

Web Scraping

In this worksheet, we will learn how to scrape the web. We will need `R` package `rvest` for the scraping. Further, we will also learn some data handling with the combination of packages called `tidyverse`. (These packages have already been installed on your machines.)

```r
library(tidyverse)
library(rvest)
```

Our objective in the first task is to scrape the names of all the faculty in the Department of Mathematics and Statistics at IIT Kanpur.

Function `read_html` goes to a website and reads and saves the html code of that website.

```r
html <- read_html("https://www.iitk.ac.in/math/faculty")
```

The complete html code for the website is saved in the object `html`. You may go to the website, view source code and see what the html code looks like.

Since the goal is to extract all faculty's name, we go to the source code to find the first faculty's name: "ANAND A.":

```html
<tr>
<td colspan="2">
<h3 class="head3"><a href="/math/../new/akash-anand" target="_blank"
style="font-size: 15px !important; font-weight: bold;">ANAND A.</a>
( PhD, University of Minnesota)</h3>
</td>
</tr>
```

We see that it looks like all faculty's names are written in the `h3` tag with class being `head3`, and further in the hyperlink tag `a`. We can use function `html_elements()` to extract all instances of a particular kind of tag.

```r
# extracting all tags with class = head3. The
# "." indicates class.
name <- html_elements(html, ".head3")

# From all the head3 class, extracting all link tags
```

```r
name <- html_elements(name, "a")

# Extracting the text associated with the links
name <- html_text(name)

## A faster way
name <- html_elements(html, ".head3 a")
name <- html_text(name)
```

And we're done! Notice how we have to keep reassigning the value of `name` or find other variable names. This can be overcome by using "pipes" available in `tidyverse`. Using pipe `%>%` allows us to do a series of operations in one go. A pipe reads like "and then do.." The line below gives the same result as before!

```r
name <- html %>% html_elements(".head3 a") %>% html_text()
```

Having learned this, let's try a few tasks:

1. Write an R program to obtain the list of post doctoral fellows in the Department of Mathematics and Statistics at IIT Kanpur.

2. Write an R code to obtain the complete list of the top 250 movies on IMDb.

   ```r
   html <- read_html("https://www.imdb.com/chart/top/")
   ```

3. Write an R code to obtain the following information about the top 250 IMDb movies (and store the information in a clean data.frame)

   a. Movie name

   b. Movie year

   c. Movie rating

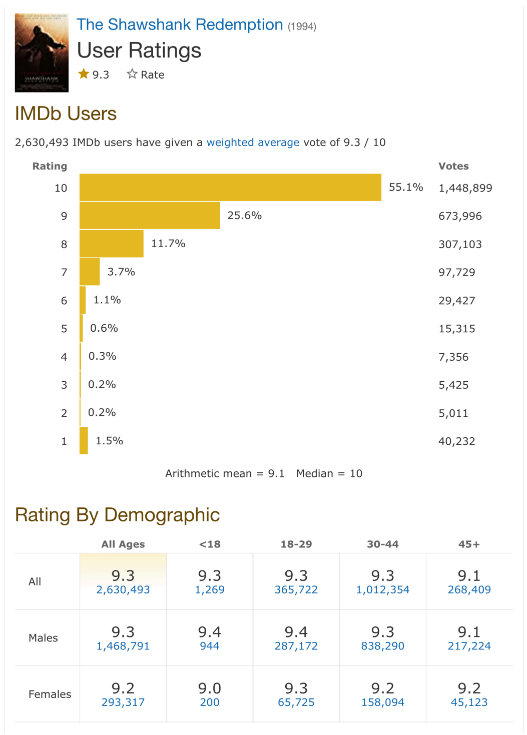
   d. Number of votes

   You may need to think about extracting parts of a string using functions `substring` or `gsub`. Also, you may need the function `html_attr`.

We will continue with our IMDb database example, and extract more detailed information. Any given movie on IMDb has a unique code attached to it. For example, the move "The Shawshank Redemption" has the code "tt0111161". This code is part of the URL of the movie. For example, the URL of "The Shawshank Redemption" is `https://www.imdb.com/title/tt0111161`

Similarly, any movie with code "xyz" has URL `https://www.imdb.com/title/xyz`. Further, IMDb also contains further details of the rating of every movie. This is available at the URL `https://www.imdb.com/title/xyz/ratings`

Go to the URL: `https://www.imdb.com/title/tt0111161/ratings` to see what the page looks like.


The Shawshank Redemption (1994)

User Ratings

★ 9.3 ☆ Rate

IMDb Users

2,630,493 IMDb users have given a weighted average vote of 9.3 / 10

Rating		Votes
10	55.1%	1,448,899
9	25.6%	673,996
8	11.7%	307,103
7	3.7%	97,729
6	1.1%	29,427
5	0.6%	15,315
4	0.3%	7,356
3	0.2%	5,425
2	0.2%	5,011
1	1.5%	40,232

Arithmetic mean = 9.1 Median = 10

Rating By Demographic

	All Ages	<18	18-29	30-44	45+
All	9.3 2,630,493	9.3 1,269	9.3 365,722	9.3 1,012,354	9.1 268,409
Males	9.3 1,468,791	9.4 944	9.4 287,172	9.3 838,290	9.1 217,224
Females	9.2 293,317	9.0 200	9.3 65,725	9.2 158,094	9.2 45,123

Our goal is to extract the following information from this movie:

- Number of votes for each type of vote: 1 - 10
- Number of votes by Men
- Number of votes by Women
- Overall rating by Men
- Overall rating by Women

In the source code for this page you will see that the information is in the form of a table. `html_table` can take an html input and scrape all tables in the html code.

```
html <- read_html("https://www.imdb.com/title/tt0111161/ratings")
all_ratings <- html %>% html_table()
```

The output of this is a `list` of all tables in the html document. There are three items in the list. Run the following code

```
length(all_ratings)
all_ratings[[1]] # First item in list
all_ratings[[2]] # Second item in list
all_ratings[[3]] # Third item in list
```

Now, the first item in list is a `data.frame` of the ratings breakdown of all votes. The second item is the gender distribution of the votes. The third is the votes from the top users of IMDb.

Using this, you may now think of extracting the information required and further complete the following tasks.

4. Repeat the above task for **all** movies in the IMDB top 250 list. The final output should be a `data.frame` with each row of the data.frame having the following information
   - Ranking of movie
   - Name of the movie
   - Year of the movie
   - Total number of votes
   - Final ratings
   - Number of votes for each type of rating: 1 - 10
   - Number of votes by Men
   - Number of votes by Women
   - Overall rating by Men
   - Overall rating by Women
5. Create a vector of length 250, having the URL for the poster for each movie in the top 250.
6. For each movie poster, calculate the proportion of pixels with a Euclidean distance of 0.2 units from black.

Introduction to dplyr

Now, we will go through some basics of the `dplyr` R package. The `dplyr` package provides a rich framework to do data manipulations and wrangling.

- Rows:
  - `filter()` chooses rows based on column values.
  - `slice()` chooses rows based on location.
  - `arrange()` changes the order of the rows.
- Columns:
  - `select()` changes whether or not a column is included.
  - `rename()` changes the name of columns.
  - `mutate()` changes the values of columns and creates new columns.
  - `relocate()` changes the order of the columns.
- Groups of rows:
  - `summarise()` collapses a group into a single row.

Please go through this website to understand the details of the `dplyr` package and these functions:

`https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html`

Please repeat the commands in this document on your own and test them out.