WEB Scraping

WEB Scraping HTML Documents

Data and information on the web is growing exponentially day by day. The amount of data available on the net is phenominal but there is only one thing that is limiting is the ability to access and get this data so that it can be constructively used for analytics purposes. Most of the data on the internet is not readily available as it is present in an unstructured format (HTML format) and is not downloadable. This project deals with a technique called web scraping which converts the data present in unstructured format (HTML tags) over the web to structured format which then can be easily accessed and used for analysis.

Benefits of WEB Scraping

Web scraping can be used for following purposes:-

- Scraping rating data from various sites to create recommendation engines
- Scraping text data for making NLP based system
- Scraping data from social media sites
- Scraping user reviews and feedbacks from e-commerce sites

Library

```
## Loading required package: xml2
```

DATA

```
# specify the url for the web site to be scrapped
url <- "http://www.imdb.com/search/title?count=100&release_date=2016,2016&title_type=feature"

# read the HTML code from the web site
webpage <- read_html(url)
```

DATA to be Scrapped

We will be scraping following data from the above mentioned web site:-

- RANK
- TITLE
- DESCRIPTION
- RUNTIME
- GENRE
- RATING
- METASCORE
- VOTES
- GROSS_Earning_In_Mil
- DIRECTOR
- ACTOR

WEB Scraping Steps

I am using Selector Gadget an open source web scraping software for HTML and CSS

1. RANK

Use the CSS selector to select rank and copy the corresponding CSS selector

```
# using CSS selectors to scrap rank section
RankDataHtml <- html_nodes(webpage, ".text-primary")

# Convert rank data to text
RankData <- html_text(RankDataHtml)

# Convert rank data from character to numeric
RankData <- as.numeric(RankData)
```

2. TITLE

Use the CSS selector to select title and copy the corresponding CSS selector

```
# using CSS selectors to scrap title selection
TitleDataHtml <- html_nodes(webpage, ".lister-item-header a")

# convert title data to text
TitleData <- html_text(TitleDataHtml)
```

3. DESCRIPTION

Use the CSS selector to select description and copy the corresponding CSS selector

```
# using CSS selectors to scrap description selection
DescriptionDataHtml <- html_nodes(webpage, ".ratings-bar+ .text-muted")

# convert description data to text
DescriptionData <- html_text(DescriptionDataHtml)

# removing \n from the description data
DescriptionData <- gsub("\n", "", DescriptionData)
```

4. RUNTIME

Use the CSS selector to select runtime and copy the corresponding CSS selector

```
# using CSS selectors to scrap runtime selection
RuntimeDataHtml <- html_nodes(webpage, ".text-muted .runtime")

# convert runtime data to text
RuntimeData <- html_text(RuntimeDataHtml)

# removing min from the runtime data
RuntimeData <- gsub(" min", "", RuntimeData)

# convert runtime data from charater to numeric
RuntimeData <- as.numeric(RuntimeData)
```

5. GENRE

Use the CSS selector to select genre and copy the corresponding CSS selector

```
# using CSS selectors to scrap genre selection
GenreDataHtml <- html_nodes(webpage, ".genre")

# convert genre data to text
GenreData <- html_text(GenreDataHtml)

# removing \n from the genre data
GenreData <- gsub("\n", "", GenreData)

# removing extra "" from the genre data
GenreData <- gsub(" ", "", GenreData)

# taking only first genre for each movie
GenreData <- gsub(",.*", "", GenreData)

# converting genre data from character to factor
GenreData <- as.factor(GenreData)
```

6. RATING

Use the CSS selector to select rating and copy the corresponding CSS selector

```
# using CSS selectors to scrap rating selection
RatingDataHtml <- html_nodes(webpage, ".ratings-imdb-rating strong")

# convert rating data to text
RatingData <- html_text(RatingDataHtml)

# convert rating data from character to numeric
RatingData <- as.numeric(RatingData)
```

7. VOTES

Use the CSS selector to select votes and copy the corresponding CSS selector

```
# using CSS selectors to scrap votes selection
VotesDataHtml <- html_nodes(webpage, ".sort-num_votes-visible span:nth-child(2)")

# convert votes data to text
VotesData <- html_text(VotesDataHtml)

# removing commas for the Votes data
VotesData <- gsub(",", "", VotesData)

# convert votes data from character to numeric
VotesData <- as.numeric(VotesData)
```

8. DIRECTOR

Use the CSS selector to select director and copy the corresponding CSS selector

```
# using CSS selectors to scrap director selection
DirectorDataHtml <- html_nodes(webpage, ".text-muted+ p a:nth-child(1)")

# convert director data to text
DirectorData <- html_text(DirectorDataHtml)
```

```r
# convert director data to factor
DirectorData <- as.factor(DirectorData)
```

9. ACTOR

Use the CSS selector to select actor and copy the corresponding CSS selector

```r
# using CSS selectors to scrap actor selection
ActorDataHtml <- html_nodes(webpage, ".lister-item-content .ghost+ a")

# convert actor data to text
ActorData <- html_text(ActorDataHtml)

# convert director data to factor
ActorData <- as.factor(ActorData)
```

10. METASCORE

Use the CSS selector to select metascore and copy the corresponding CSS selector

```r
# using CSS selectors to scrap metascore selection
MetascoreDataHtml <- html_nodes(webpage, ".metascore")

# convert metascore data to text
MetascoreData <- html_text(MetascoreDataHtml)

# removing extra spaces in metascore
MetascoreData <- gsub(" ","", MetascoreData)

# checking length of metascore
length(MetascoreData)
```

```
## [1] 95
```

```r
# convert metascore data to numeric
MetascoreData <- as.numeric(MetascoreData)

# The length of metascore data is 95 while we are scrappind data for 100 movies.
# The reason this has happened because 5 movies do not have corresponding metascore
# fields. The visual inspection of the web site shows that metascores are missing
# for movies 62, 71, 74, 79, 94

# find quantiles for metascore data
quantiles <- quantile(MetascoreData, probs = c(0.5), na.rm = TRUE)

# formulae for replacing missing metascore values with NAs
for (i in c(62, 71, 74, 79, 94)){

   aa <- MetascoreData[1:(i-1)]
   bb <- MetascoreData[i:length(MetascoreData)]
   MetascoreData <- append(aa,list("NA"))
   MetascoreData <- append(aa, quantiles)
   MetascoreData <- append(MetascoreData,bb)
}

# convert metascore data to numeric
MetascoreData <- as.numeric(MetascoreData)
```

```r
length(MetascoreData)
```

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

```r
summary(MetascoreData)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   23.00   49.00   63.00   61.23   72.50   99.00
```

11. GROSS EARNINGS

Use the CSS selector to select gross earnings and copy the corresponding CSS selector

```r
# using CSS selectors to scrap gross earnings selection
GrossEarningDataHtml <- html_nodes(webpage, ".ghost~ .text-muted+ span")

# convert gross eraning data to text
GrossEarningData <- html_text(GrossEarningDataHtml)

# removing $ in gross  earning
GrossEarningData <- gsub("M","", GrossEarningData)

# removing $ in gross earning
GrossEarningData<- substring(GrossEarningData,2,6)

length(GrossEarningData)
```

```
## [1] 89
```

```r
# Convert gross earnings data to numeric
GrossEarningData <- as.numeric(GrossEarningData)

# find quantiles for gross earnings data
quantiles1 <- quantile(GrossEarningData, probs = c(.5), na.rm = TRUE)

# The length of gross earnings data is 89 while we are scrappind data for 100 movies.
# The reason this has happened because 11 movies do not have corresponding gross earnings
# fields. The visual inspection of the web site shows that gross earnings are missing
# for movies 45, 48, 53, 62, 63, 68, 71, 72, 79, 94, 96

# formulae for replacing missing gross earnings values with NAs
for (i in c(45, 48, 53, 62, 63, 68, 71, 72, 79, 94, 96)){

   aaa <- GrossEarningData[1:(i-1)]
   bbb <- GrossEarningData[i:length(GrossEarningData)]
   GrossEarningData <- append(aaa, quantiles1)
   #GrossEarningData <- append(aaa,list("NA"))
   GrossEarningData <- append(GrossEarningData,bbb)
}

# convert gross earnings data to numeric
GrossEarningData <- as.numeric(GrossEarningData)
length(GrossEarningData)
```

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

```
summary(GrossEarningData)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.15   26.86   54.73   92.37  105.58  532.10
```

Create a Dataframe

We have now scraped 11 features for 100 movies which we will now convert into a single dataframe

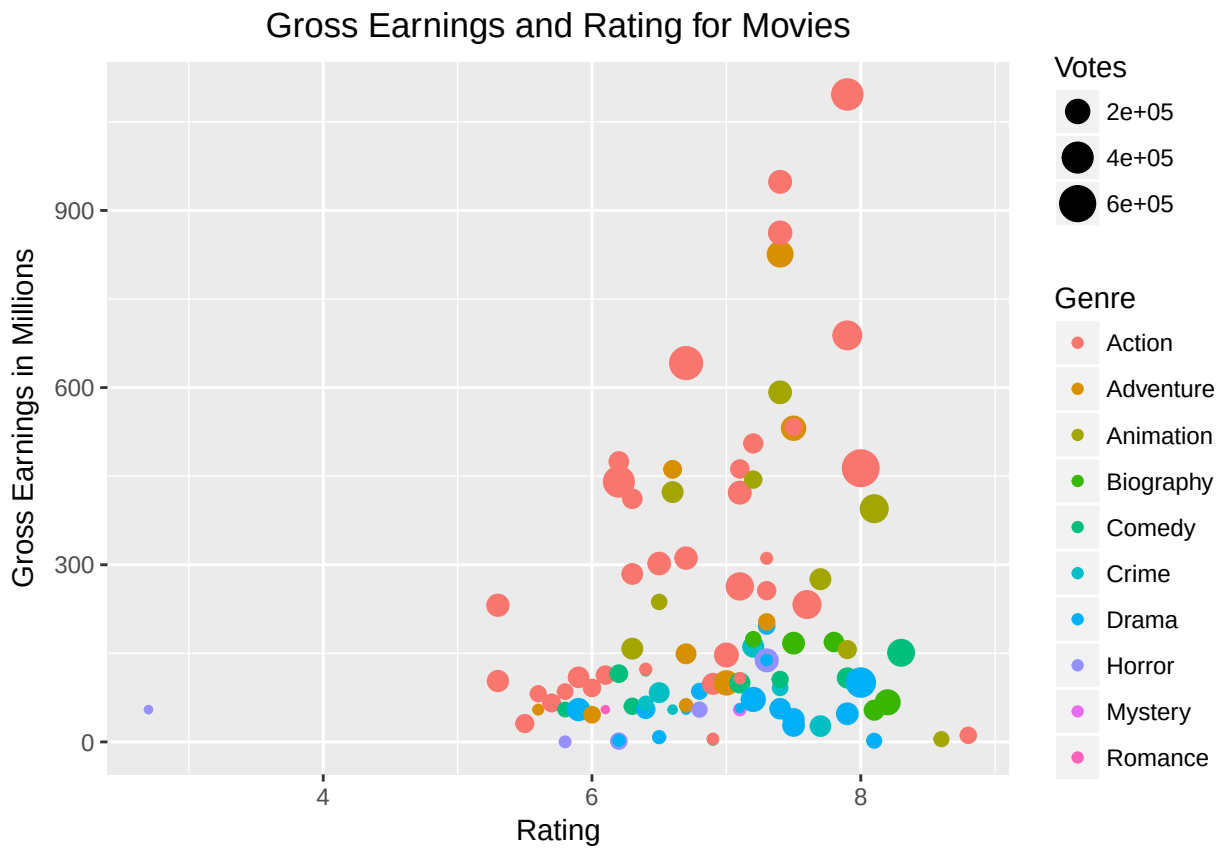
```
MovieDataframe <- data.frame(Rank = RankData, Title = TitleData,
                             Desc = DescriptionData, Runtime = RuntimeData,
                             Genre = GenreData, Rating = RatingData,
                             Metascore = MetascoreData,
                             Votes = VotesData,
                             GrossEarning = GrossEarningData,
                             Director = DirectorData, Actor = ActorData)

str(MovieDataframe)
```

```
## 'data.frame':    100 obs. of  11 variables:
##  $ Rank        : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Title       : Factor w/ 100 levels "10 Cloverfield Lane",..: 71 73 70 82 46 56 19 28 67 31 ...
##  $ Desc        : Factor w/ 100 levels "A boy seeks the help of a tree monster to cope with his single
##  $ Runtime     : num  117 123 108 103 128 107 109 133 133 90 ...
##  $ Genre       : Factor w/ 10 levels "Action","Adventure",..: 8 1 3 1 5 3 1 2 1 1 ...
##  $ Rating      : num  7.3 6.2 7.2 6 8.3 7.7 6.4 7.4 7.9 6.9 ...
##  $ Metascore   : num  62 40 59 42 93 81 70 66 65 63 ...
##  $ Votes       : num  165269 398041 63078 61171 264570 ...
##  $ GrossEarning: num  138.1 325 270.3 45.1 151 ...
##  $ Director    : Factor w/ 98 levels "Alex Proyas",..: 55 21 16 97 18 77 63 24 30 10 ...
##  $ Actor       : Factor w/ 88 levels "Aamir Khan","Aaron Poole",..: 41 85 59 58 71 8 7 26 35 77 ...
```

Analysis of Scraped Data

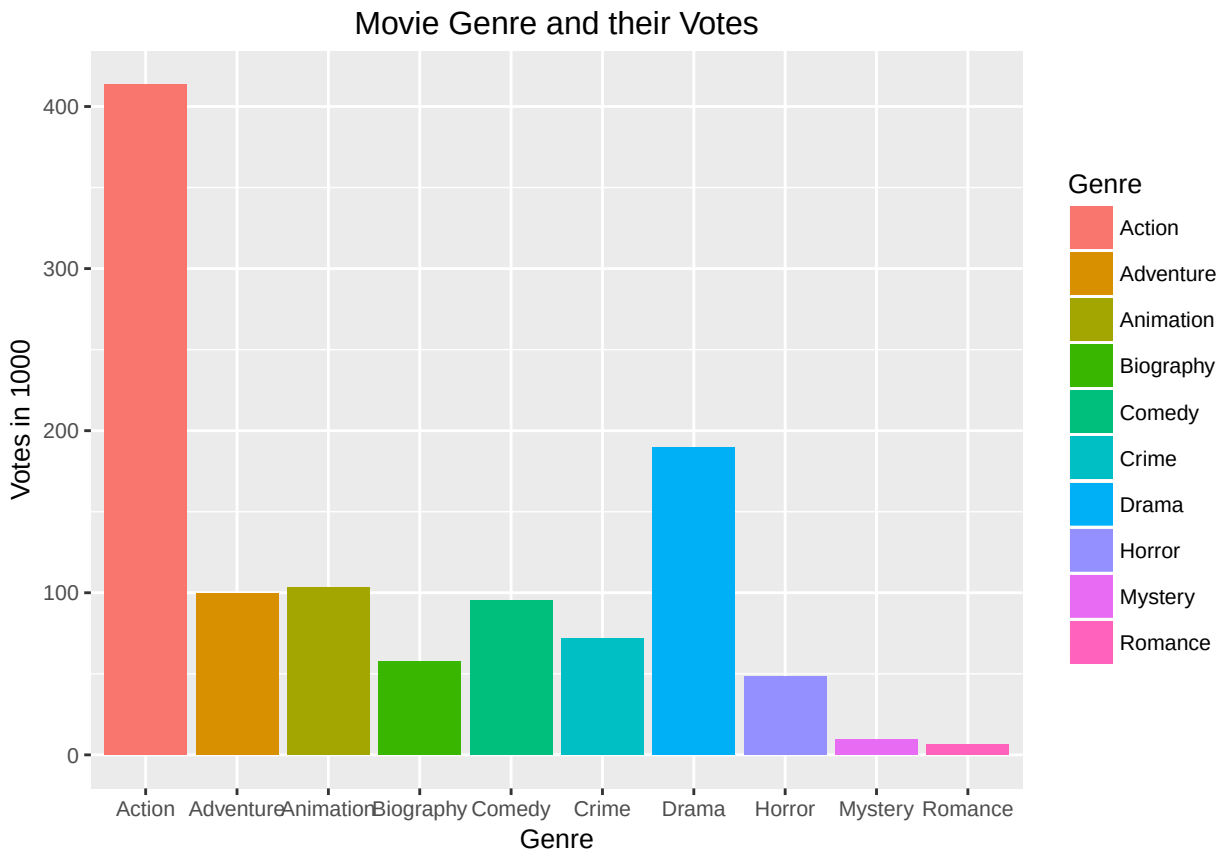
1. Movie Rating and their Gross Earnings

```
ggplot(data = MovieDataframe, aes(x = Rating, y = GrossEarning, fill = Genre,
                                  size= Votes, col = Genre)) +
                    geom_point(stat = "identity", position = "stack") +
                    ylab("Gross Earnings in Millions") +
                    ggtitle("Gross Earnings and Rating for Movies") +
                    theme(plot.title = element_text(hjust = 0.5))
```



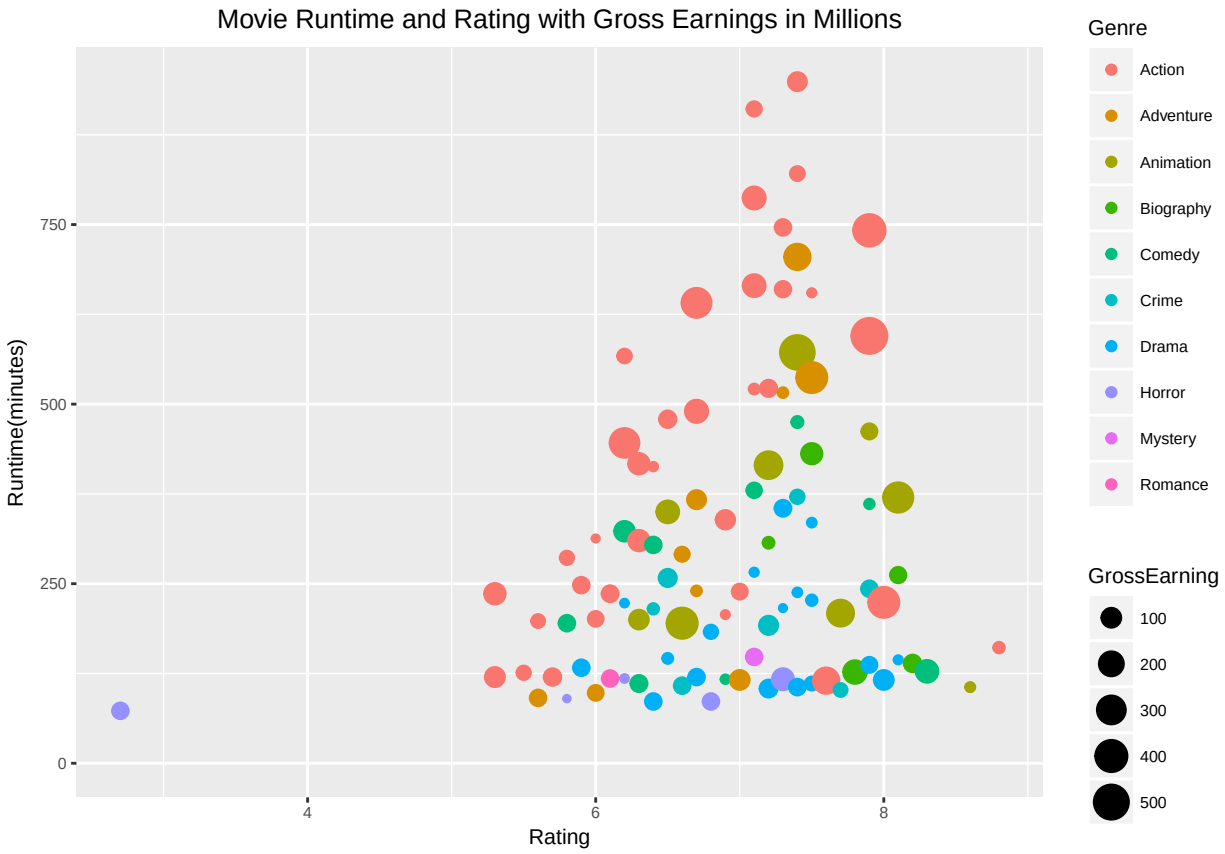
2. Movie Genre and their Votes

```
ggplot(data = MovieDataframe, aes(x = Genre, y = log(Votes), fill = Genre)) +
                    geom_bar(stat = "identity", position = "stack") +
                    ggtitle("Movie Genre and their Votes") +ylab("Votes in 1000") +
                    theme_grey(base_size = 10) +
                    theme(plot.title = element_text(hjust = 0.5))
```



3. Movie Runtime and Rating

```
ggplot(MovieDataframe, aes(x = Rating, y = Runtime, col = Genre, size = GrossEarning)) +
    geom_point(stat = "identity", position = "stack") +
  theme_grey(base_size = 8) + ylab("Runtime(minutes)") +
  ggtitle("Movie Runtime and Rating with Gross Earnings in Millions") +
  theme(plot.title = element_text(hjust = 0.5))
```


Movie Runtime and Rating with Gross Earnings in Millions
Runtime(minutes)
750
500
250
0
4
6
8
Rating
Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Horror
Mystery
Romance
GrossEarning
100
200
300
400
500