

**Executive Summary:**

In the 21st century, we can observe an increasing trend in the different types of online series produced and streamed. It would be crucial for any producer or director to check what kind of Online series do viewers like to watch. The variation in opinion subjects to genre, average run time, the star cast of the series, etc. As investors invest millions of dollars in the production of TV series, they need to identify and forecast the potential trend in the ratings thereby gaining proactive knowledge on the type of TV series that could potentially enhance the viewership and profit.

Amazon’s IMDB database is an online database that provides vivid information about a wide range of motion pictures and their contents such as Movies, TV Shows, etc. Scrapping the data for the top 250 TV series based on the number of ratings and identifying the potential trends existing amongst them.

For this study, we have identified some influential variables such as run time, genre, duration, etc., which might impact the ratings of any particular TV series. Checking out the trend between these variables and ratings for the top 250 TV series would help us identify what it would take for any type of TV Series to be among the highest-rated.

**Statement of Scope:**

The scope of this study/project lies in finding the decisive variables which impact/influence the rating of any particular TV series. Here, we have extracted different variables for the top 250 highest-rated TV series on IMDB.

**Project Objectives:**

• Scrap the data for the top 250 variables (TV Series) from the IMBD website along with rating and convert it in the required form for further analysis after performing data cleaning.

• Determine the cause and effect relationship among these variables to have a better understanding of the data.

• Identify the relationship between different variables such as runtime, genre etc. & the rating for any particular TV series. This would ultimately help us identify cause and effect relationships between different influencers on ratings for TV series which would, in turn, lead us to identify the trend between successful TV series

Here, Data about 250 TV series would serve as a sample of all successful TV series and generalize the trend successively.

**Variables:**

We have collected the data related to average run time, release year, end year, number of episodes, type of genre, number of awards, number of seasons, number of episodes, number of critic reviews provided, ratings etc. for each particular TV series from the sample data of top 250 series extracted from IMDB.

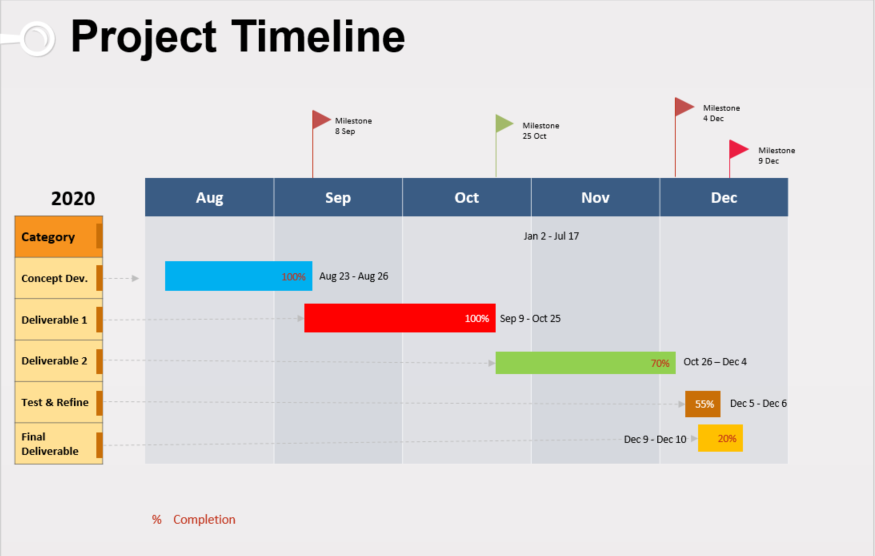
Here we are considering ratings as a target variable and trying to identify potential cause and effect relationship of all other variables on ratings to improve rating prediction in summary.

Target variable: Ratings

Independent variables: average run time, genre, number of seasons, number of episodes, number of reviews provided, number of critic reviews, release year, end year, number of awards.

**Project Schedule:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CATEGORY** | **NAME** | **TASK** | **START** | **END** | **COLOR** |
| **Concept Dev.** | **Kushal Shah, Loksagar Rudraraju Subramanyam** | **Idea Brainstorming** | 8/23/20 | 8/26/20 | Blue |
|  | **Kushal Shah, Loksagar Rudraraju Subramanyam** | **Project Shortlisting** | 8/26/20 | 8/28/20 | Blue |
|  | **Kaviya Venkatramanan, Khyati Nema** | **Project Proposal** | 9/3/20 | 9/7/20 | Blue |
|  |  |  | 8/29/20 | 9/8/20 | Blue |
| **Deliverable 1** | **Kushal Shah, Loksagar Rudraraju Subramanyam** | **Team Discussion 1 - Tool Selection** | 9/9/20 | 9/10/20 | Red |
|  | **Kaviya Venkatramanan, Khyati Nema** | **Team Discussion 2 - Data Preparation** | 9/12/20 | 9/14/20 | Red |
|  | **Kushal Shah, Loksagar Rudraraju Subramanyam** | **Team Discussion 3 - Web Scraping** | 9/1/20 | 9/11/20 | Red |
|  | **Kaviya Venkatramanan, Khyati Nema** | **Data Dictionary and Descriptive Statistics** | 10/20/20 | 10/22/20 | Red |
|  | **Loksagar Rudraraju Subramanyam** | **Project Deliverable 1 Submission** | 10/23/20 | 10/25/20 | Red |
|  |  |  |  |  |  |
| **Deliverable 2** | **Kushal Shah, Loksagar Rudraraju Subramanyam** | **Align Deliverable 1 Requirements** | 10/26/20 | 11/1/20 | Green |
|  | **Kaviya Venkatramanan, Khyati Nema** | **Data Analysis** | 11/4/20 | 11/5/20 | Green |
|  | **Kaviya Venkatramanan, Khyati Nema** | **Other Analysis** | 11/15/20 | 11/17/20 | Green |
|  | **Kushal Shah, Loksagar Rudraraju Subramanyam** | **Project Formatting** | 12/1/20 | 12/3/20 | Green |
|  | **Kaviya Venkatramanan, Khyati Nema** | **Project Deliverable 2 Submission** | 12/4/20 | 12/4/20 | Green |
|  |  |  |  |  |  |
| **Test & Refine** | **Kaviya Venkatramanan, Khyati Nema** | **Project Testing** | 12/5/20 | 12/6/20 | Brown |
|  | **Kushal Shah, Loksagar Rudraraju Subramanyam** | **Code QA** | 12/7/20 | 12/8/20 | Brown |
|  |  |  |  |  |  |
| **Final Deliverable** | **Final Deliverable** | **Presentation** | 12/9/20 | 12/10/20 | Orange |



**Data Preparation:**

**Data Access:**

we have scrapped URLs of all the Top 250 TV series for our analysis from Website: <https://www.imdb.com/chart/toptv/?sort=nv,desc&mode=simple&page=1>

This website contains the information about top 250 TV series based on the number of ratings which helped us gather the required data for our analysis. For further information regarding the respective TV series, we have scrapped the data from the corresponding TV series URL on IMDB.

For example, for the game of thrones series, we have scrapped the data from the URL below, to check different information such as release year, end year, number of episodes, number of seasons for that particular series.

URL: <https://www.imdb.com/title/tt0944947/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=12230b0e-0e00-43ed-9e59-8d5353703cce&pf_rd_r=RRPHAM1ZHNF1M2TF8VED&pf_rd_s=center-1&pf_rd_t=15506&pf_rd_i=toptv&ref_=chttvtp_tt_11>

**Data Consolidation:**

We scraped all the variables individually and used pandas Dataframe function in R to combine all the variables into a single Dataframe. We exported the consolidated Dataframe as a tab-delimited text file for further analysis.

**Data Cleaning:**

The data we scraped doesn’t require any cleaning. The end year has few null values because some TV series are still running. There are very few Null values for the variables number of reviews, number of critics’ reviews. The series which had null values for these two variables will be dropped.

**Data Transformation:**

The numerical data we scraped was originally in the text format, which was transformed into numeric data with the help of as.numeric function. The variable genre was transformed into +a categorical variable using as.factor.

**Data Reduction:**

The data reduction is not performed.

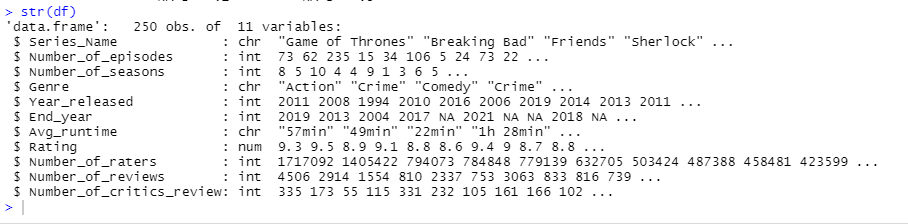
**Data Dictionary:**

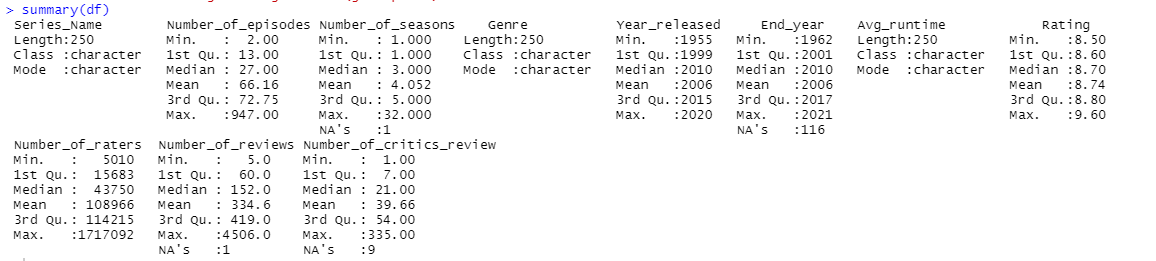
|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Source** |
| Series\_Name | Name of the series | Char | https://www.imdb.com/ |
| Number\_of\_Episodes | Number of Episodes in the TV Series | Numeric | https://www.imdb.com/ |
| Number\_of\_Seasons | Number of Seasons in the TV Series | Numeric | https://www.imdb.com/ |
| Genre | The Genre of the TV Series | Char | https://www.imdb.com/ |
| Years\_released | Year the Series was released | Integer | https://www.imdb.com/ |
| End\_year | Year the Series ended | Date | https://www.imdb.com/ |
| Avg\_runtime | Average runtime of TV Series | Numeric | https://www.imdb.com/ |
| Rating | TV Series Rating | Integer | https://www.imdb.com/ |
| Number\_of\_raters | Number of Raters | Numeric | https://www.imdb.com/ |
| Number\_of\_reviews | Number of Reviews | Numeric | https://www.imdb.com/ |
| Number\_of\_Critics\_reviews | Number of Critics Reviews | Numeric | https://www.imdb.com/ |

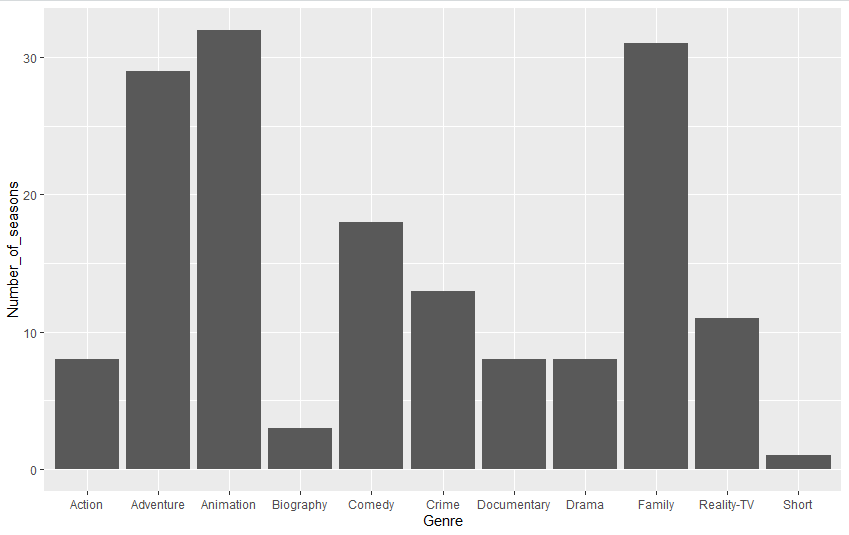
The source is <https://www.imdb.com/chart/toptv/?sort=nv,desc&mode=simple&page=1> for all the variables.

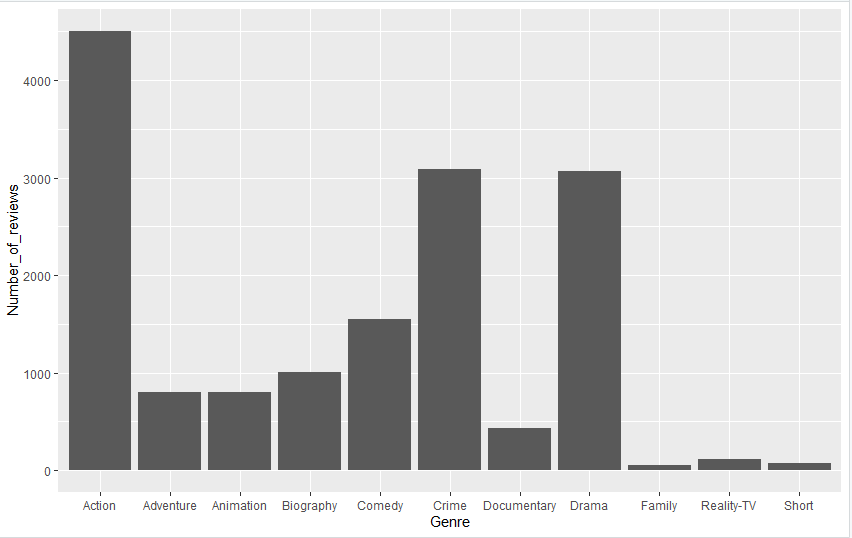
**Visualizations**

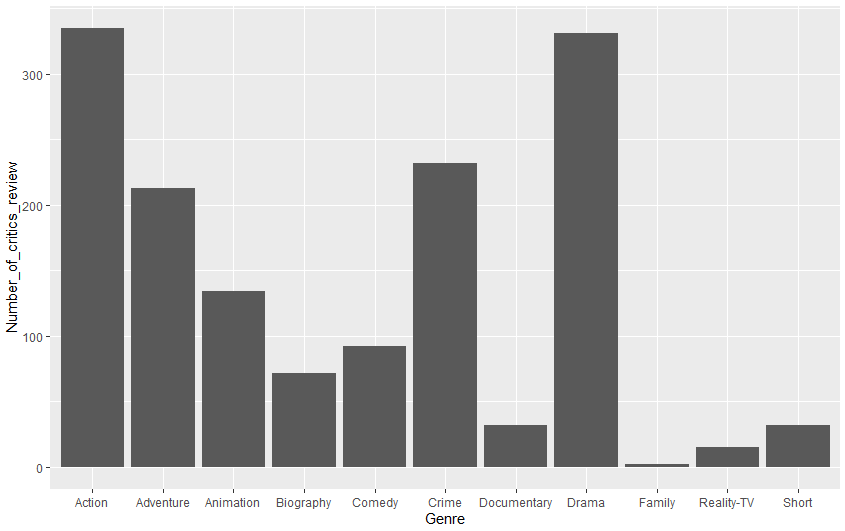
Data Visualizations aid us to gain insights into the data. We have employed ggplots and other multivariate plots to interpret the data effectively. We analyzed our data using R and Tableau.

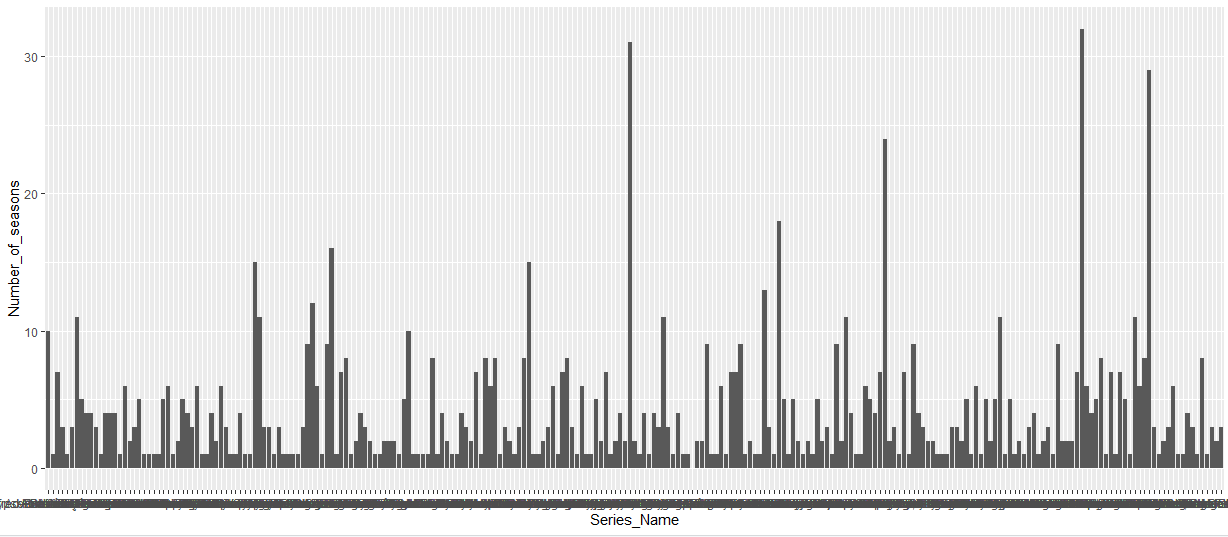




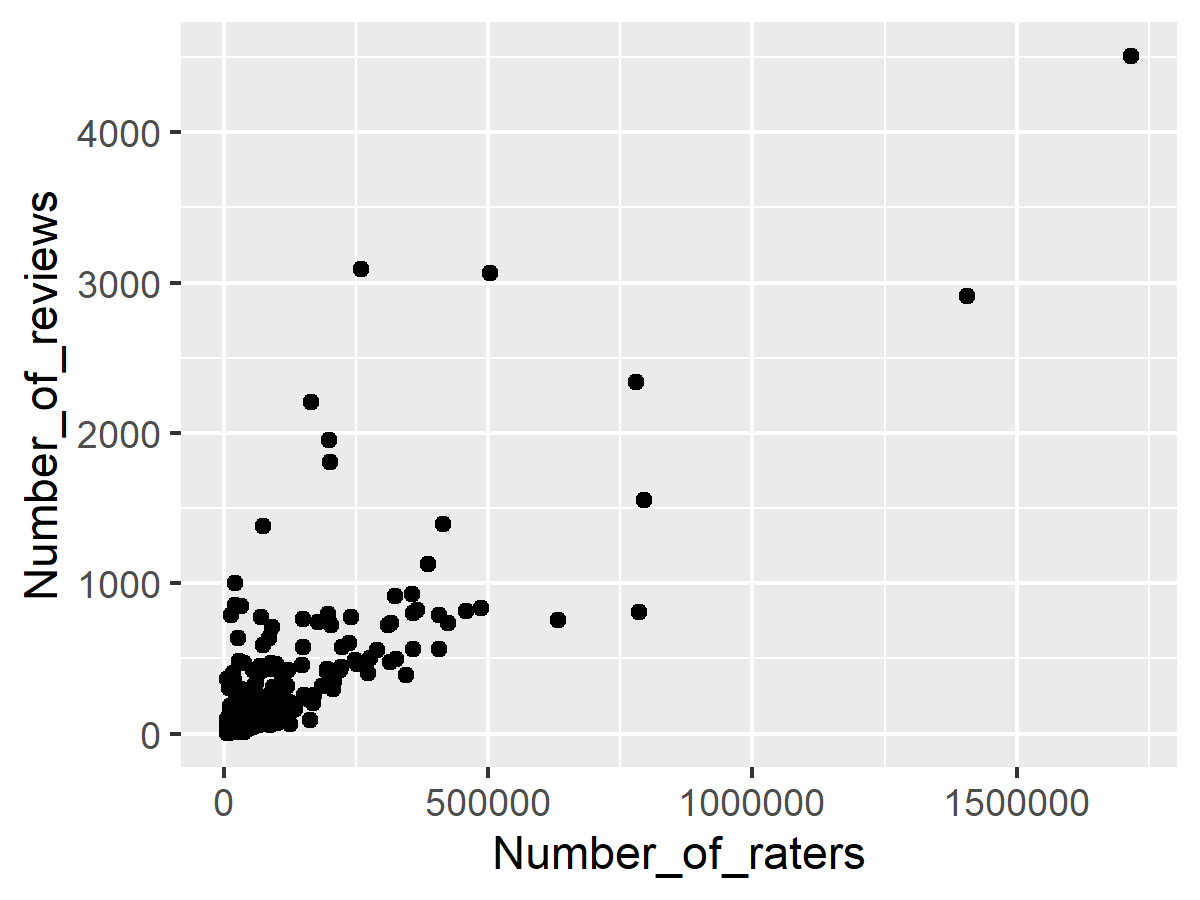


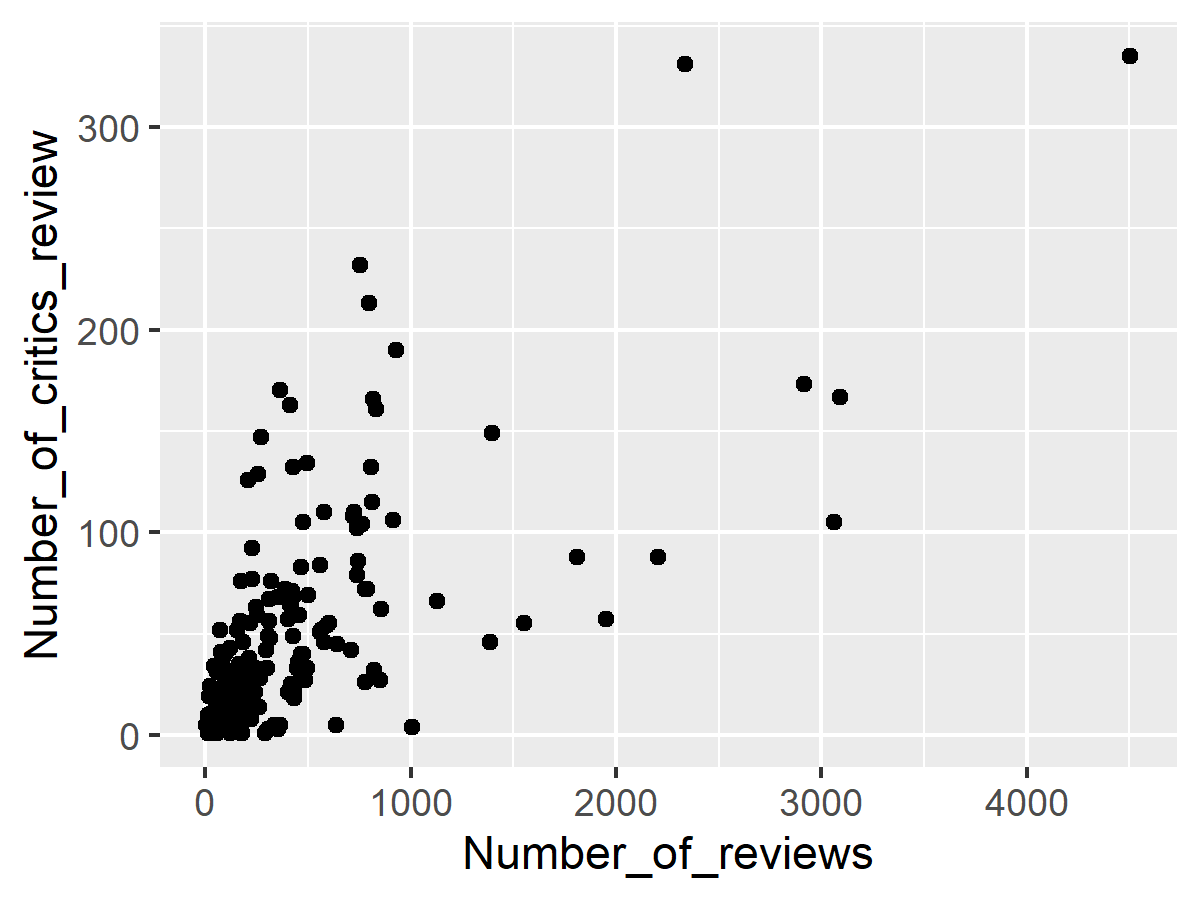




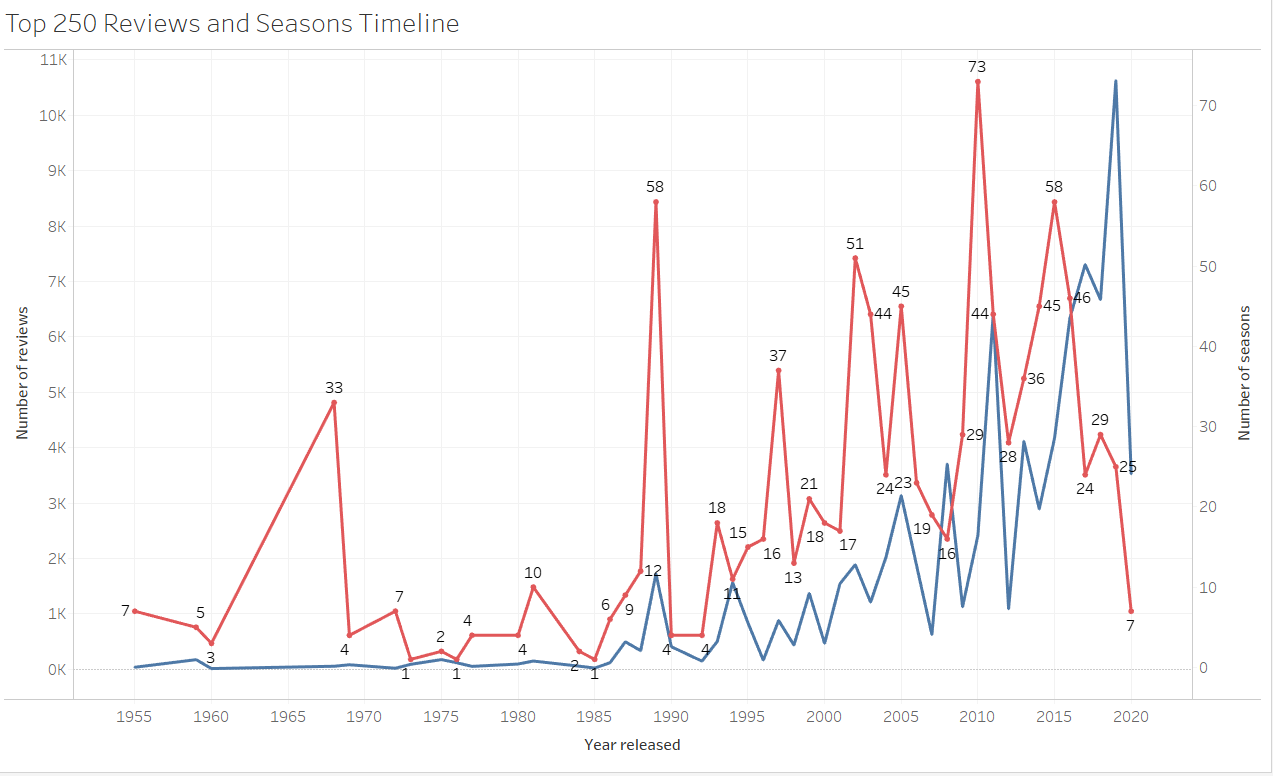


The graph below shows the trends between the number of raters and the number of reviews. We can see a strong linear relationship between the two variables. There appears to few outliers in the data. The second scatterplot shows if there is any relationship between the number of reviews and the number of critics reviews and again we can see a linear relationship between the two variables.

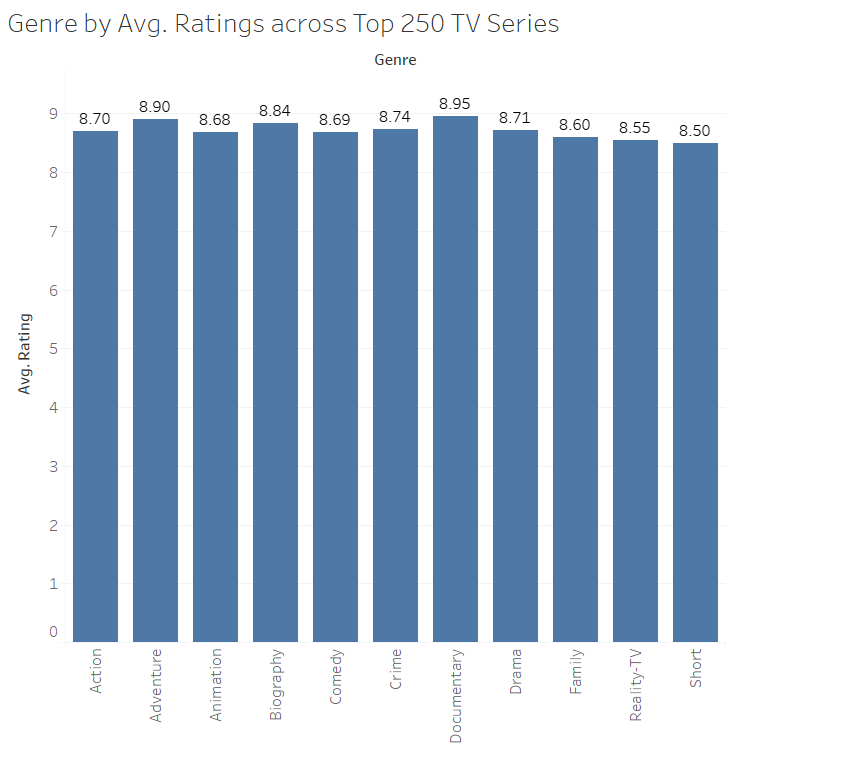


****

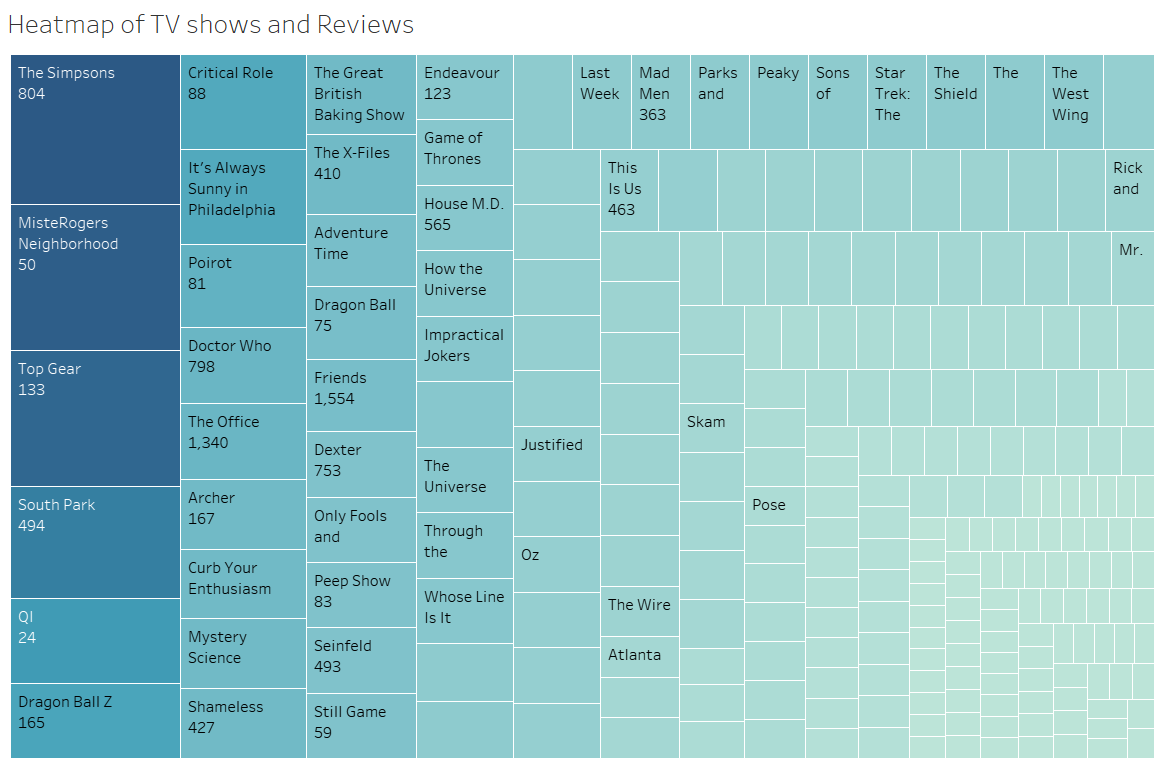
This visualization is showing the trending of the number of reviews and the number of seasons in a kind of Dual Axis Chart. We can see that year 2010 contributed most number of seasons to the Top 250 TV shows and the number of reviews appear to peak in the subsequent year of a peaked season year. We can see an upward trend in the number of reviews as more people are contributing in the year 2000s and onward.

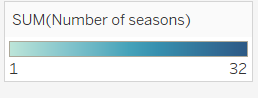


This was an interesting visual as we can see the average ratings across all the Genres hovered around 8.70 with the maximum rating going to Documentary (8.95). There could be several reasons attributed to it.



In the below heat map, interestingly, we can see that The Simpsons with the highest number of seasons (32 seasons) had 804 reviews, while MisteRogers Neighborhood with the second-highest number of seasons (31) had only 50 reviews.





**Appendix:**

#loading the libraries

library(rvest)

library(stringr)

# this url contains the details of Top 250 TV series based on the number of ratings

url = 'https://www.imdb.com/chart/toptv/?sort=nv,desc&mode=simple&page=1'

imdb = read\_html(url)

#scraping urls of top 250 TV shows

name = html\_nodes(imdb,'.lister-list > tr > td:nth-child(2) > a')

# urls we scraped above doesn't have "https://www.imdb.com" included in the start of the urls, we are adding that by pasting it in front of each urls.

links = paste("https://www.imdb.com", html\_attr(name, "href"))

#the code below is used for scraping the name of the series

# we are getting the names of the series by using html\_text function and adding that into

name\_of\_series = html\_text(name)

name\_of\_series=c(name\_of\_series)

name\_of\_series

## number of episode

# we created the empty vector to add all the values we scrap.

# we used for loop to iterate each url in the links and scrap the respective values from the links

## we used as.numeric function to transform the values as numeric.

number\_of\_episodes=c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

numb = html\_node(tv\_link,'a.bp\_item > div:nth-child(1) > div:nth-child(1) > span:nth-child(2)')

numb2 = html\_text(numb)

num3 = as.numeric(substr(numb2,1,regexpr("e",numb2)-2))

number\_of\_episodes[i]=num3

}

number\_of\_episodes

#number of seasons

# we created the empty vector to add all the values we scrap.

# we used for loop to iterate each url in the links and scrap the respective values from the links

## we used as.numeric function to transform the values as numeric.

# there was a problem with the 103rd value, so we have to run the loop again from 104th link to the end

number\_of\_seasons = c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

seasons =html\_nodes(tv\_link,'.seasons-and-year-nav > div:nth-child(4) > a:nth-child(1)')

number\_of\_seasons[i] = as.numeric(html\_text(seasons))

}

for (i in 104:length(links)) {

tv\_link = read\_html(links[i])

seasons =html\_nodes(tv\_link,'.seasons-and-year-nav > div:nth-child(4) > a:nth-child(1)')

number\_of\_seasons[i] = as.numeric(html\_text(seasons))

}

number\_of\_seasons

#genre

# we created the empty vector to add all the values we scrap.

# we used for loop to iterate each url in the links and scrap the respective values from the links

genre = c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

genre\_raw = html\_nodes(tv\_link,'.subtext > a[href^="/search"]')

genre[i] = html\_text(genre\_raw)

}

genre

#end year and start year

# we created empty vectors for both end year and start year.

# used for to loop through each url from the links

#used html\_text to get the text part from the html\_nodes

# we used substr,regexpr function to get the start year and end year from the string that had both the values together

end\_year=c()

release\_year = c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

release = html\_node(tv\_link,'.subtext > a:last-child')

rele = html\_text(release)

release\_year[i] = as.numeric(substr(rele,regexpr('s',rele)+3,regexpr('s',rele)+6))

end\_years = as.numeric(substr(rele,regexpr("s",rele)+8,regexpr("s",rele)+11))

end\_year[i]=end\_years

}

end\_year

release\_year

# average runtime of episodes

# we created empty vectors for both end year and start year.

# used for to loop through each url from the links

# used html\_text to get the text part from the html\_nodes

# we used trimws function to remove all the whitespace from the texts.

runtime=c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

r = html\_node(tv\_link,'.subtext > time:nth-child(2)')

run1 = html\_text(r)

runtime[i] = trimws(str\_remove\_all(run1,"\n"), which = c("both", "left", "right"), whitespace = "[ \t\r\n]")

}

runtime

## ratings

# we created the empty vector to add all the values we scrap.

# we used for loop to iterate each url in the links and scrap the respective values from the links

## we used as.numeric function to transform the values as numeric.

ratings=c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

rate = html\_node(tv\_link,'.ratingValue > strong:nth-child(1) > span:nth-child(1)')

rate1 = as.numeric(html\_text(rate))

ratings[i]=rate1

}

ratings

#number of raters

# we created the empty vector to add all the values we scrap.

# we used for loop to iterate each url in the links and scrap the respective values from the links

# we removed the comma in between the numbers using gsub function.

number\_of\_raters = c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

raters = html\_nodes(tv\_link,'span.small')

number\_of\_raters[i] = as.numeric(gsub(",", "", html\_text(raters)))

}

number\_of\_raters

# number of reviews #problem

# we created empty vectors named number\_of\_reviews to store the values we scraped

# used for to loop through each url from the links

# used html\_text to get the text part from the html\_nodes

# we used as.numeric, substr,regexpr function to get the required number\_of\_reviews

number\_of\_reviews = c()

for (i in 1:172) {

tv\_link = read\_html(links[i])

reviews = html\_nodes(tv\_link,'div.titleReviewBarItem:nth-child(1) > div:nth-child(2) > span:nth-child(1) > a:nth-child(1)')

number\_of\_reviews\_text = gsub(",", "", html\_text(reviews))

number\_of\_reviews[i] = as.numeric(substr(number\_of\_reviews\_text,1,regexpr('u',number\_of\_reviews\_text)-1))

}

for (i in 173:length(links)) {

tv\_link = read\_html(links[i])

reviews = html\_nodes(tv\_link,'div.titleReviewBarItem:nth-child(1) > div:nth-child(2) > span:nth-child(1) > a:nth-child(1)')

number\_of\_reviews\_text = gsub(",", "", html\_text(reviews))

number\_of\_reviews[i] = as.numeric(substr(number\_of\_reviews\_text,1,regexpr('u',number\_of\_reviews\_text)-1))

}

number\_of\_reviews

# number of critic reviews

# we created empty vectors named number\_of\_critics\_reviews to store the values we scraped

# used for to loop through each url from the links

# used html\_text to get the text part from the html\_nodes

# we used as.numeric, substr,regexpr function to get the required number\_of\_critics\_reviews

number\_of\_critics\_reviews = c()

for (i in 1:length(links)) {

tv\_link = read\_html(links[i])

critics = html\_nodes(tv\_link,'div.titleReviewBar> div.titleReviewbarItemBorder > div:nth-child(2) > span:nth-child(1)')

critics\_text = html\_text(critics)

number\_of\_critics\_reviews[i] = as.numeric(substr(critics\_text,regexpr('c',critics\_text)-4,regexpr('c',critics\_text)-1))

}

number\_of\_critics\_reviews

# creating dataframe

# we created data frame using pandas function and gave name for each column in the dataframe

imdb\_df = data.frame(name\_of\_series, number\_of\_episodes, number\_of\_seasons, genre, release\_year, end\_year, runtime, ratings, number\_of\_raters, number\_of\_reviews, number\_of\_critics\_reviews)

names(imdb\_df)=c("Series\_Name","Number\_of\_episodes","Number\_of\_seasons","Genre","Year\_released","End\_year","Avg\_runtime","Rating","Number\_of\_raters","Number\_of\_reviews","Number\_of\_critics\_review")

imdb\_df

# we have extracted the dataframe as tab delimited file

write.table(imdb\_df,file = "ImdbTvSeries.txt", sep="\t")

**Descriptive Statistics**

install.packages("tidyverse")

install.packages("lubridate")

install.packages("ggplot2")

install.packages("ggplot")

install.packages("psych")

library(ggplot2)

library(ggplot)

library(psych)

library(lubridate)

library(tidyverse)

install.packages("devtools")

library(devtools)

df <- read.table("ImdbTvSeries.txt")

tail(df)

summary(df)

str(df)

df$Avg\_runtime

hist(df$Number\_of\_critics\_review,df$Genre)

hist(df$Number\_of\_reviews)

geom\_histogram(df$Number\_of\_seasons)

plot2 <- ggplot(df, aes(x = Number\_of\_reviews, y = Number\_of\_critics\_review)) +

geom\_point()

ggsave("imdb-1.png", plot2, width=4, height=3)

plot <- ggplot(df, aes(x = Number\_of\_raters, y = Number\_of\_reviews)) +

geom\_point()

ggsave("imdb-0.png", plot, width=4, height=3)

ggplot2.barplot(data=df, xName="Genre", yName='Number\_of\_seasons',width = 1)

ggplot(data = df, aes(x = Genre, y = Number\_of\_reviews)) +

geom\_col(position = position\_dodge())

ggplot(data = df, aes(x = Genre, y = Number\_of\_critics\_review)) +

geom\_col(position = position\_dodge())

ggplot(data = df, aes(x = Genre, y = Number\_of\_seasons)) +

geom\_col(position = position\_dodge())

average\_Rating <- average(df$Rating)

ggplot(data = df, aes(x = Genre, y = average(Rating)) +

geom\_col(position = position\_dodge())

ggplot(data = df, aes(x = Genre, y = Number\_of\_seasons)) +

geom\_col(position = position\_dodge())