Analysing the content on Netflix

Project: STAT40620_Data Programming with R

Anisha Mittal

12/8/2020

ABSTRACT

From the last one year, the world is facing a pandemic due to which humanity has been forced to stay indoors. Everything in the world is done online now. Since staying indoors does not leave a lot of entertainment options, there has been a significant increase in watching movies and Tv shows. There is all type of content available from all over the world at one single platform: NETFLIX. This report analyses the contents available on Netflix US.

NOTE:

The dataset used in this project is taken from https://www.kaggle.com/shivamb/netflix-shows (https://www.kaggle.com/shivamb/netflix-shows) , and is updated monthly. The data contains information about the movies and Tv shows available on Netflix US. There is some missing data in this data set but for the purpose of this project, we will assume that this is accurate and perform our analysis.

OBJECTIVES

-> ETL: Extract Transform Load of the data to identify and select the useful data.

-> Study the content of the following basis:

1. Fitting an ARIMA model (time series) to the duration of movies over the year to find a pattern.
2. Year when the content was released and when it was added on Netflix.
3. Find a relation between the no. of seasons of Tv shows and their ratings.
4. Directors with maximum content.
5. Correlation between the categories of movies
6. Correlation between the categories of Tv shows.

ETL: EXTRACT TRANSFORM LOAD

In this section we will load the data from a csv file. We will clean and format the data to avoid any wrong analysis arising from missing data or non-standard values.

We could clearly see that there was a lot of missing value in the directors column, but we still wanted the column for our analysis. Therefore, before removing it as a column we created a vector of it.

We did not want to analyze the description of the content available on Netflix so, we will remove the whole column on description.

Next, we could see some rows with missing values with respect to ratings, casts, country and date added. So, we remove these rows.

show_id	type	title	cast	country	date_added	release_year	rating	duration	listed_in
Min. : 247747	Length:5256	Length:5256	Length:5256	Length:5256	Length:5256	Min. :1942	Length:5256	Length:5256	Length:5256
1st Qu.:80003185	Class :character	Class :character	Class :character	Class :character	Class :character	1st Qu.:2012	Class :character	Class :character	Class :character
Median :80152626	Mode :character	Mode :character	Mode :character	Mode :character	Mode :character	Median :2016	Mode :character	Mode :character	Mode :character
Mean :76141509						Mean :2013			
3rd Qu.:80238292						3rd Qu.:2018			
Max. :81235729						Max. :2020			

show_id	type	title	cast	country	date_added	release_year	rating	duration	listed_in
81145628	Movie	Norm of the North: King Sized Adventure	Alan Marriott, Andrew Toth, Brian Dobson, Cole Howard, Jennifer Cameron, Jonathan Holmes, Lee Tockar, Lisa Durupt, Maya Kay, Michael Dobson	United States, India, South Korea, China	September 9, 2019	2019	TV-PG	90 min	Children & Family Movies, Comedies
80117401	Movie	Jandino: Whatever it Takes	Jandino Asporaat	United Kingdom	September 9, 2016	2016	TV-MA	94 min	Stand-Up Comedy
70234439	TV Show	Transformers Prime	Peter Cullen, Sumalee Montano, Frank Welker, Jeffrey Combs, Kevin Michael Richardson, Tania Gunadi, Josh Keaton, Steve Blum, Andy Pessoa, Ernie Hudson, Daran Norris, Will Friedle	United States	September 8, 2018	2013	TV-Y7-FV	1 Season	Kids' TV
80058654	TV Show	Transformers: Robots in Disguise	Will Friedle, Darren Criss, Constance Zimmer, Khary Payton, Mitchell Whitfield, Stuart Allan, Ted McGinley, Peter Cullen	United States	September 8, 2018	2016	TV-Y7	1 Season	Kids' TV
80125979	Movie	#realityhigh	Nesta Cooper, Kate Walsh, John Michael Higgins, Keith Powers, Alicia Sanz, Jake Borelli, Kid Ink, Yousef Erakat, Rebekah Graf, Anne Winters, Peter Gilroy, Patrick Davis	United States	September 8, 2017	2017	TV-14	99 min	Comedies

DURATION OF MOVIES

Movies are released from all over the world available to watch for the whole world. Each movie industry from different countries has a set duration for the movies, but these movies are released all over the world. Not everyone is used to watching a 2hr hollywood movie. Does the creator want the whole world to be its target audience and does that decision change the duration of the movie? Is there any correlation between the duration of movies which are released for the whole world by any country.

We will use the forecast package available on R to study the plot of duration of movies as a time series, its sample auto correlation (ACF) sample partial auto correlation (PACF) plots to fit an ARIMA(p,d,q) model. We will use the tseries package available on R to verify our fitted model by using the Augmented Dickey Fuller test.

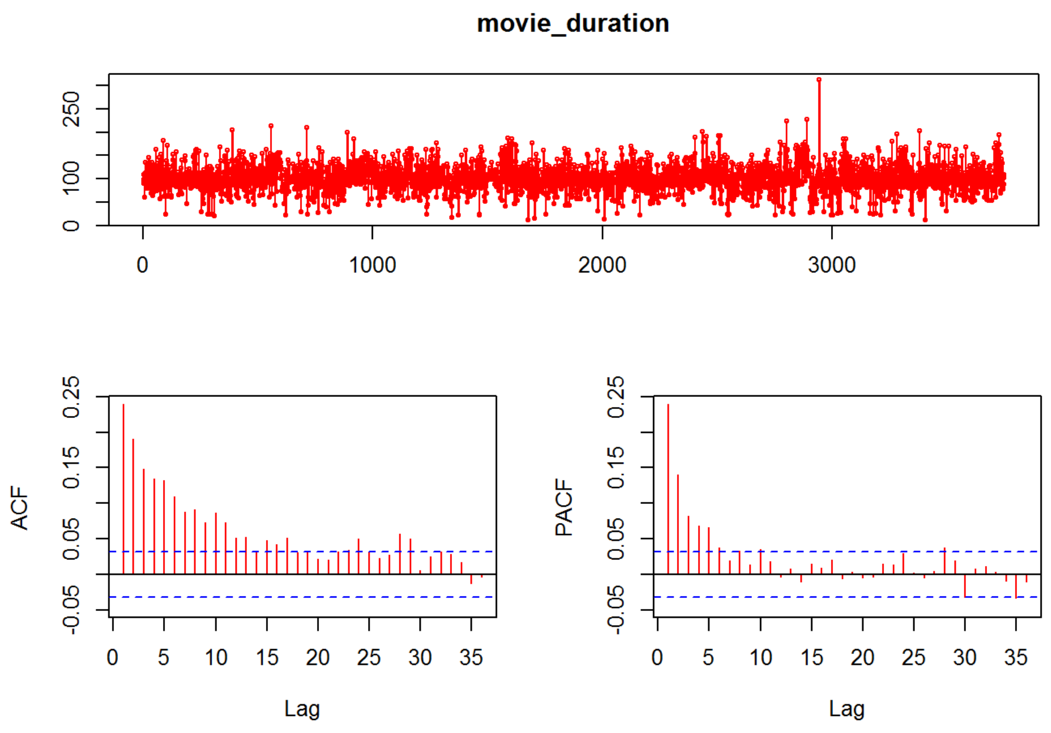
We begin by filtering the movies and the Tv shows from the data set. R considers the duration column as a character vector since after the values, the unit of duration (mins) is mentioned. So, we slice the vector and transform the vector to a numerical vector. This helps in easy plotting of the data.

Before fitting an ARIMA model to any time series we first have to de - mean the series.

We estimate the value of ‘p’ by looking at the PACF plot of the series. Note: If we difference the series, we look at the PACF plot of the differenced series. If the PACF plot decays exponentially then p = 0, otherwise p = no. of non-null lags That is; the number of lags poking out of the blue critical line which signifies the lags which are significantly different from zero.

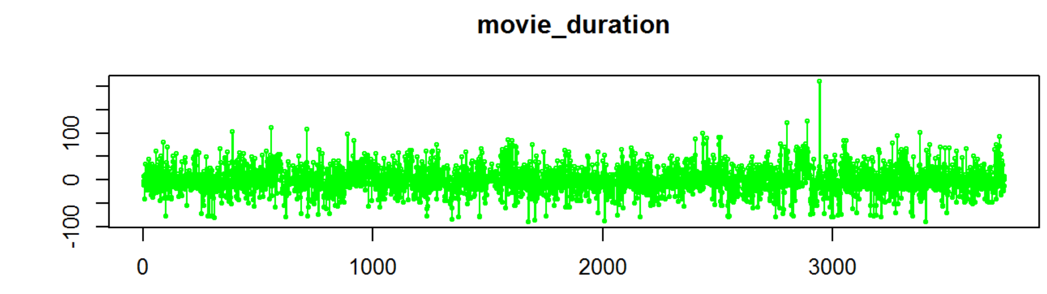
We estimate the value of ‘d’ and ‘q’ by looking at the ACF plot of the series. Note: If we difference the series, we look at the PACF plot of the differenced series to estimate ‘q’. If the ACF plot decays exponentially then q = 0, otherwise q = no. of non-null lags That is; the number of lags poking out of the blue critical line which signifies the lags which are significantly different from zero. If the ACF plot decays but at a very slow rate the d > 0. To determine the value of ‘d’ we difference the series and follow the same procedure till the time ACF plot decays exponentially. Then d = no. of times the series has been differenced.

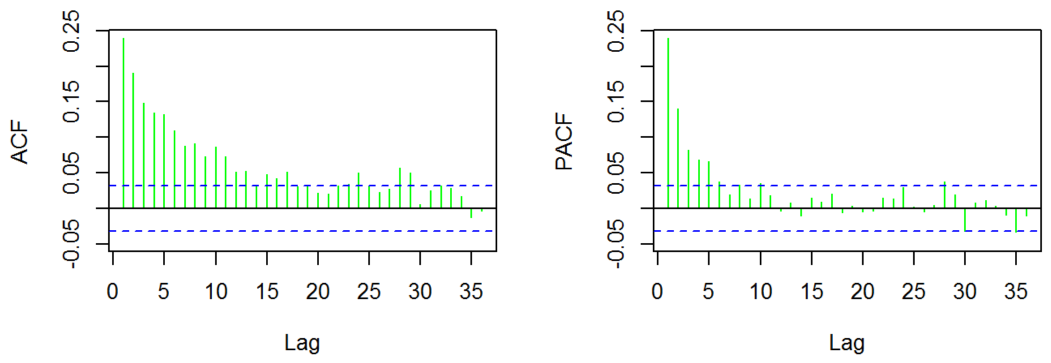
The Augmented dickey fuller test is used to test weather the series has a unit root (null hypothesis) or is stationary (alternative hypothesis). We reject the null hypothesis if the p value obtained from the test is less than statistically significant level. We will consider 5% significance level.



```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    12.0    88.0    99.0   101.9   117.0   312.0
```

```
## [1] 27.05714
```





```
## Warning in adf.test(diff(movie_duration), k = 0): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(movie_duration)
## Dickey-Fuller = -101.62, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

OBSERVATIONS

-> The plots in red color shows the general plot of the duration of movies as a time series. Where the plot of duration is around its mean value. The plots in green shows the de-mean time series. Now we will consider these plots to fit an ARIMA model. We can see the ACF and PACF plots are decaying exponentially. Therefore, We do not plot the differenced time series.

-> The p value obtained from the Augmented Dickey Fuller test is 0.01 < 0.05. That is, It is less than 5% significance level. Therefore we can reject the null hypothesis and accept the alternative hypothesis.

From the above observations we can say that, p=d=q=0. Duration of movies is stationary time series. The augmented Dickey Fuller test verifies this.

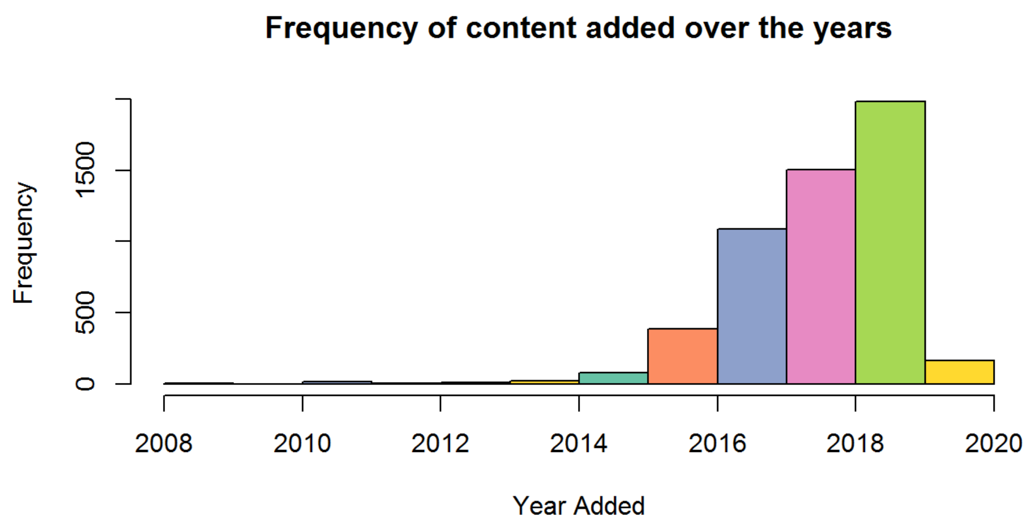
Duration of movies is a white noise with constant mean(μ) and variance(σ^2) over time. This means that each observation is uncorrelated with other observation. We can not find a pattern on the duration of movies which are released over the world. This means that a 2hr movie might be normal in the US but might not be watched in one go by people in other countries. That is, The movie creator might not consider reducing or increasing the duration of the movie in order to trget the whole world. The average duration of a movie is approximately 102 mins with a deviation of 27 mins. So we can say that the duration of most movies lies in the interval (75,129) mins.

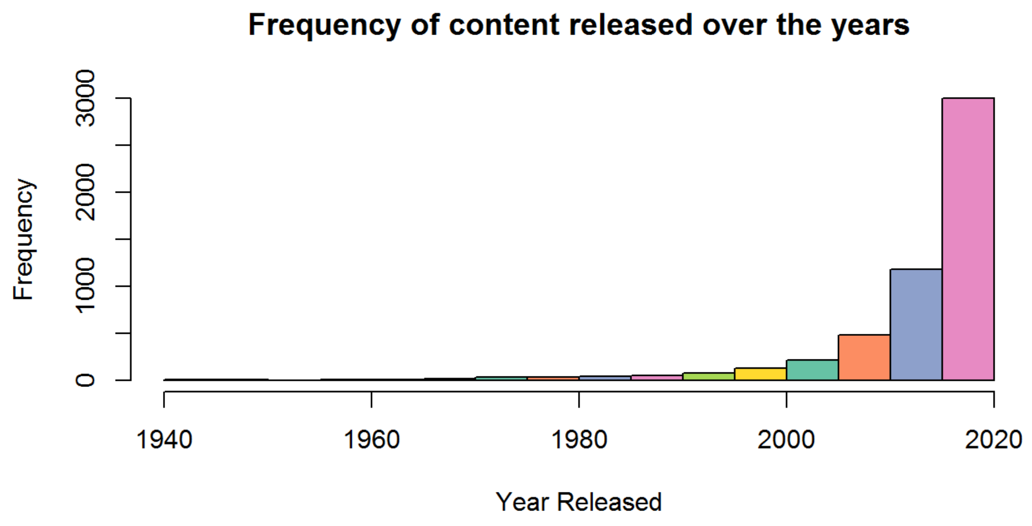
YEAR ADDED AND YEAR RELEASED

Netflix was launched in the US in the year 1998 and was then launched to 190 countries in 2016. This part of the analysis visualizes the frequency of content added and released over the years.

In the raw data the format of the dates was given as month name dd, yyyy. We first change the format to YYYY-MM-DD using the anytime package available on R. Then we add a new numeric vector to the data set with only the years of when the content was added to Netflix. The data is then plotted in two different histograms: 1. Frequency of content added over the years 2. Frequency of content released over the years

The color of the histogram has been selected by using the RColorBrewer package available on R for clear distinction of the years. A table is created to see the oldest content available on Netflix.





	type	year_released	title	year_added
2014	Movie	1942	The Battle of Midway	2017
2022	Movie	1944	Tunisian Victory	2017
2006	Movie	1945	Know Your Enemy - Japan	2017
2007	Movie	1946	Let There Be Light	2017
2931	Movie	1946	The Stranger	2018
2021	Movie	1947	Thunderbolt	2017

OBSERVATIONS

We can see from the histogram that there is a significant increase in the frequency of content added after 2015. It means it was only after 2016, the year of worldwide launch of Netflix, did the frequency of content added was tripled. This Shows that people are consuming the content added at warp speed making Netflix increase their supply and being the top OTT platform today.

We can see from the histogram that content released in the 1900’s are still relevant in today’s world and being added on Netflix in the late 2010’s. The oldest movie was made available in 2017 is titled: THE BATTLE OF MIDWAY which was released in 1942. The oldest Tv Show was made available in 2017 is titled: THE TWILIGHT ZONE which was released in 1963.

There are 355 movies and Tv shows from the 1900’s that are still significant in the world today and is easily available for generation Z to enjoy.

DURATION Vs. RATINGS (TV SHOWS)

There are a lot of Tv shows available on Netflix with n number of seasons and different ratings. Let us see the frequency of Tv shows with most seasons and their ratings.

The following table gives the meaning of each rating.

Ratings	Fullform
NR	Not Restricted
R	Restricted
TV-14	May not be suitable for ages 14 and under,parental guidance suggested
TV-G	General Audience
TV-MA	May not be suitable for ages 17 and under, Mature Audience
TV-PG	Parental Guidance suggested
TV-Y	Suitable for Children of all ages
TV-Y7	Suitable for ages 7+
TV-Y7-FV	Suitable for ages 7+, Fantasy Violence

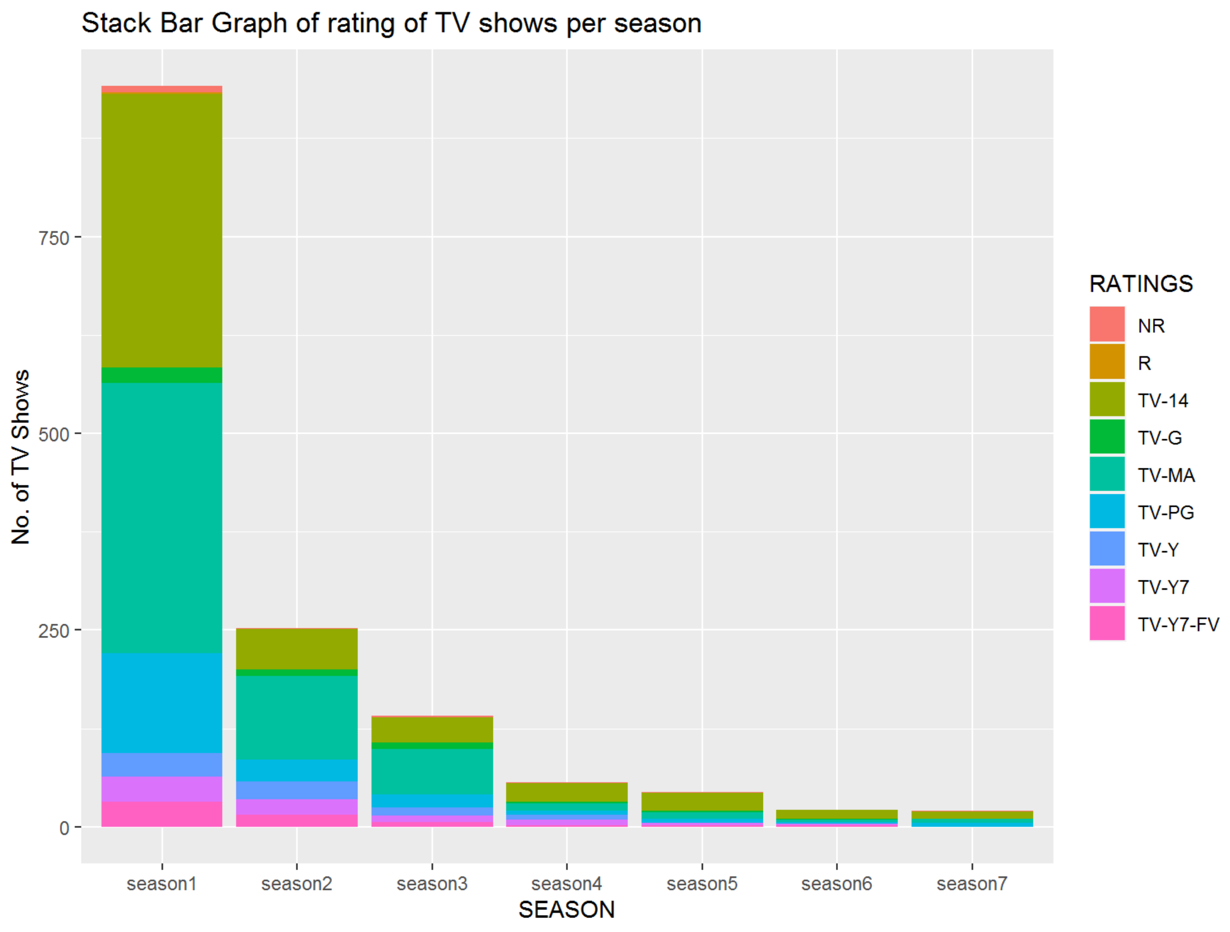
Before cleaning the data there were some shows which continued till the 11th season. But after cleaning the data it was found the maximum number of season available for Tv shows are 7.

The Function first filters the Tv shows with different number of seasons to concatenate to one vector with the ratings and their duration. The the function creates a 7*9 matrix. The matrix tells the number of Tv shows with their duration and ratings. These values were put in using a for loop. There were some ratings and duration with no Tv shows, so these values were replaced by 0. The function takes the input as the whole data set and in return we get the matrix created.

The ggplot2 package in R is used to further plot the above created matrix as stacked bar plot.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# WELCOME TO HELP FILE OF find_ratings FUNCTION
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

-> To access this file type the following command find_ratings_help()

-> The following function takes an argument in the form of the Data Frame.
-> The data frame is divided in to 7 Seasons and 9 Ratings of TV - Shows.

=> To use the function call: find_ratings(data_frame_name)

-> In general the function has only one input and it returns a data frame in
   the output.
-> The output is ratings_df data frame, which contains the count of TV Shows
   with respect to their ratings and duration.
```

SEASON	NR	R	TV-14	TV-G	TV-MA	TV-PG	TV-Y	TV-Y7	TV-Y7-FV
season1	8	2	347	20	344	126	30	32	32
season2	1	0	51	8	107	27	23	20	15
season3	2	0	32	8	58	16	11	8	6
season4	1	0	24	2	10	5	6	7	2
season5	1	0	23	2	8	5	0	3	2
season6	0	0	11	1	3	2	0	2	2
season7	1	0	9	0	5	5	0	0	0



OBSERVATONS

With help of the above function we can make the following observations:

-> Almost 70% of the TV shows are not available or not continued after the 1st season.

-> There are only 20 TV shows with 7 seasons available on Netflix.

-> There is a 85% decrease from season 1 to season 2 of Tv shows with TV-14 rating. There is a 70% decrease from season 1 to season 2 of Tv shows with TV-MA rating. There is a 78% decrease from season 1 to season 2 of Tv shows with TV-PG rating.

-> Only the shows with rating TV-14, TV-MA, and TV-PG are available on Netflix that go on till season 7. This means that Tv shows on Netflix are for more mature and adult audience or require parental guidance.

-> There are hardly any shows with restricted ratings.

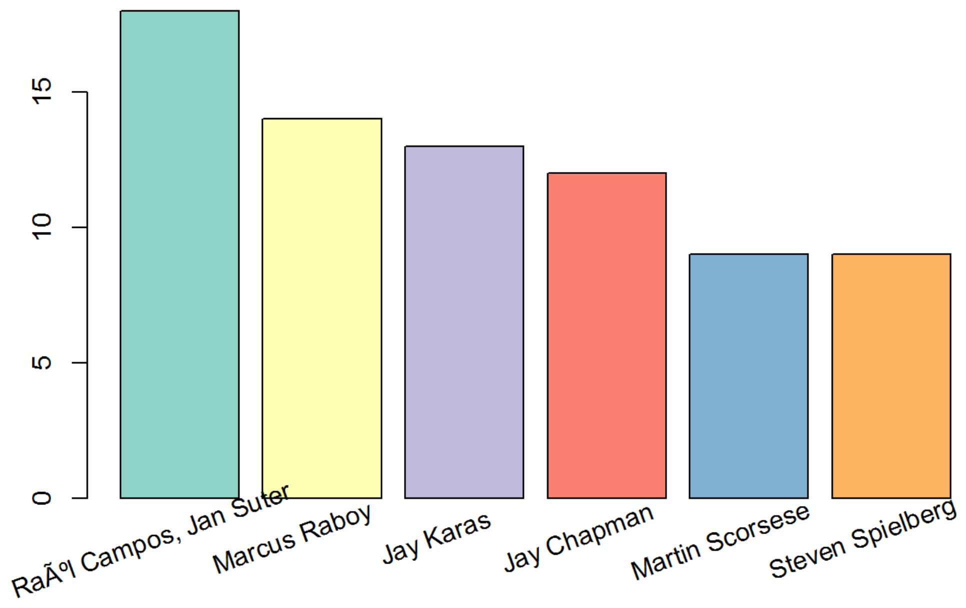
DIRECTORS WITH MAXIMUM CONTENT

Using the director vector created in the beginning we will see the top 6 directors with maximum content available on Netflix.

We will first remove the missing values from the vector. On checking it was seen that out of the top 6, number 2 was given to missing values. So it was important to clean the data.

Then we sorted the directors in decreasing order and selected the top 6 to create a data frame.

The data frame was visualized as a bar plot for analysis. The color of the bar plot has been selected by using the RColorBrewer package available on R for clear distinction of the years.



OBSERVATIONS

We can clearly see the top 6 directors with maximum content are:

Director	Freq
RaÃºl Campos, Jan Suter	18
Marcus Raboy	14
Jay Karas	13
Jay Chapman	12
Martin Scorsese	9
Steven Spielberg	9

CORRELATION OF CATEGORIES

Looking at how the varied the content available on Netflix is. The content has been listed in multiple categories. We will find correlations between two categories of the content available on Netflix.

In order to do that we will create another function. The function first removes missing values, renames and filter the categories of the whole data to create a new vector. The function then removes the white noise from the data. Since the content is listed under multiple categories, the function groups and counts the content by their type and categories uniquely. Then the function creates different data frames for movies and Tv shows. Both these data frames contains categories as rows and columns and counts the number of contents listed under multiple categories. Then we bind these two data frames to calculate the correlations easily and not repeating our work.The we use ggplot2 to plot the correlation between any two categories. The function takes the input as the whole data set and in return we get the correlation plots.

The range of correlation for movies is [0, 1000] st

-> count around 0 suggest No correlation between two categories

-> count around 500 suggests a significant amount of correlation between two categories

-> count around 1000 suggests a very strong/overlapping correlation between two categories.

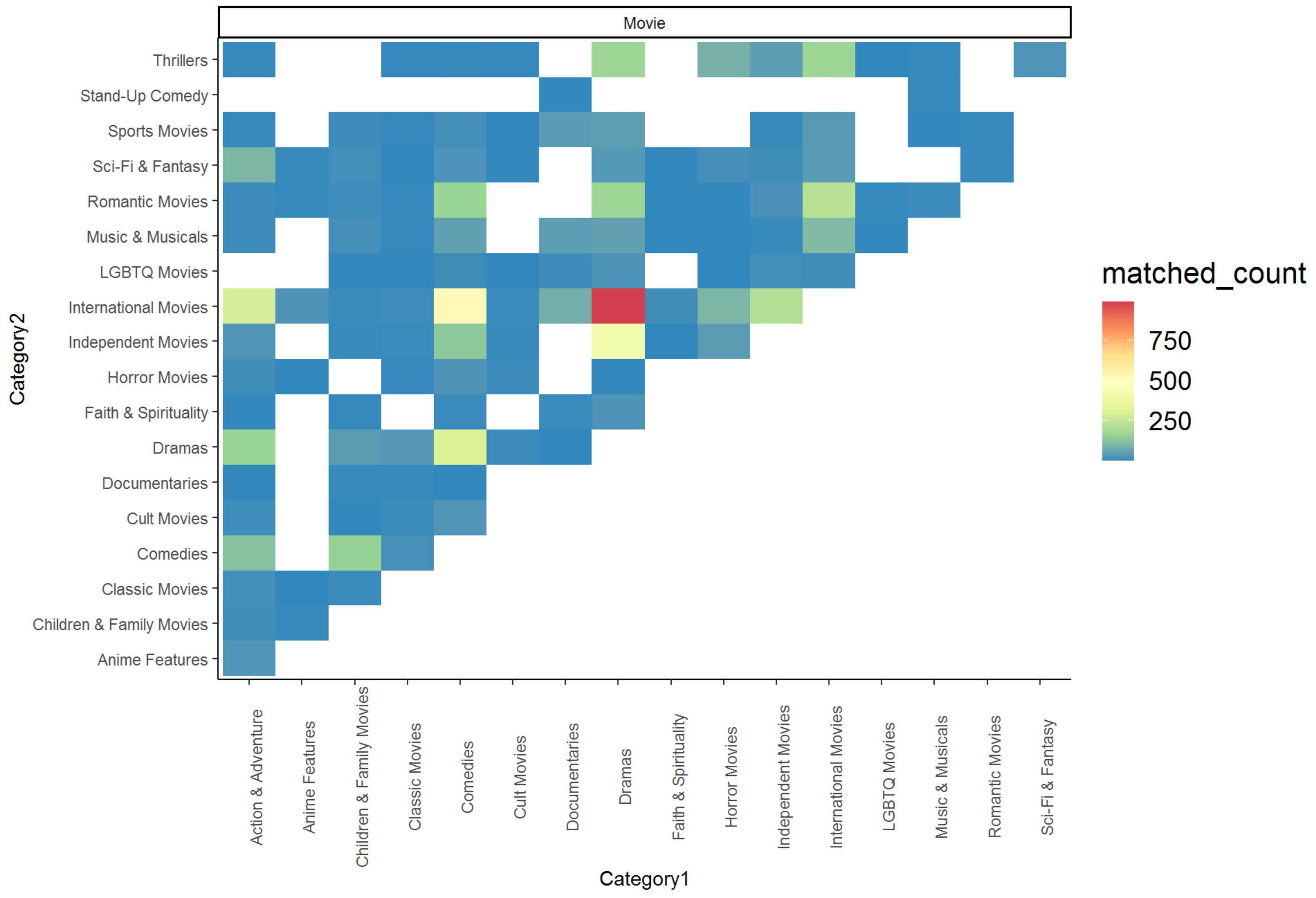
The range of correlation for Tv Shows is [0, 300] st

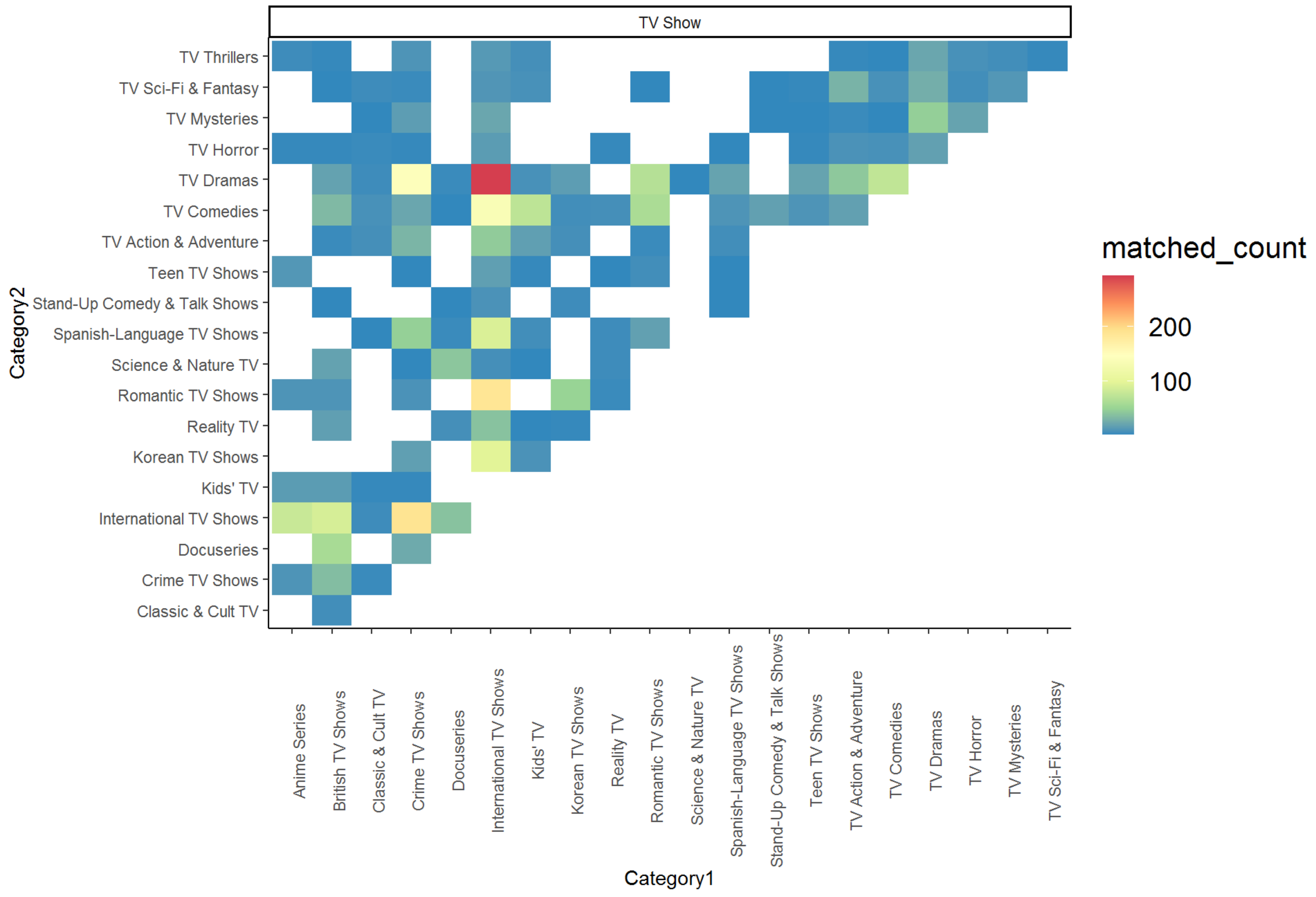
-> count around 0 suggest No correlation between two categories

-> count around 150 suggests a significant amount of correlation between two categories

-> count around 300 suggests a very strong/overlapping correlation between two categories.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# WELCOME TO HELP FILE OF find_corr FUNCTION
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

-> To access this file type the following command find_corr_help()

-> The following function takes an argument in the form of the Data Frame.
-> The data frame is divided in to 2 categories and the count of the content

=> To use the function call: find_corr(data_frame_name)

-> In general the function has only one input and it returns two plots in
   the output.
-> The outputs are correlation plots of the categories of movies and Tv shows.
```


Movie
Thrillers
Stand-Up Comedy
Sports Movies
Sci-Fi & Fantasy
Romantic Movies
Music & Musicals
LGBTQ Movies
International Movies
Independent Movies
Horror Movies
Faith & Spirituality
Dramas
Documentaries
Cult Movies
Comedies
Classic Movies
Children & Family Movies
Anime Features
Category2
Action & Adventure
Anime Features
Children & Family Movies
Classic Movies
Comedies
Cult Movies
Documentaries
Dramas
Faith & Spirituality
Horror Movies
Independent Movies
International Movies
LGBTQ Movies
Music & Musicals
Romantic Movies
Sci-Fi & Fantasy
Category1
matched_count
750
500
250


TV Show
TV Thrillers
TV Sci-Fi & Fantasy
TV Mysteries
TV Horror
TV Dramas
TV Comedies
TV Action & Adventure
Teen TV Shows
Stand-Up Comedy & Talk Shows
Spanish-Language TV Shows
Science & Nature TV
Romantic TV Shows
Reality TV
Korean TV Shows
Kids' TV
International TV Shows
Docuseries
Crime TV Shows
Classic & Cult TV
Category2
Anime Series
British TV Shows
Classic & Cult TV
Crime TV Shows
Docuseries
International TV Shows
Kids' TV
Korean TV Shows
Reality TV
Romantic TV Shows
Science & Nature TV
Spanish-Language TV Shows
Stand-Up Comedy & Talk Shows
Teen TV Shows
TV Action & Adventure
TV Comedies
TV Dramas
TV Horror
TV Mysteries
TV Sci-Fi & Fantasy
Category1
matched_count
200
100

OBSERVATIONS

MOVIES

-> There is a strong correlation between dramas and international movies. There is a significant amount of correlation between dramas and comedies, and comedies and international movies. This suggests many of the international movies are listed under dramas. Other of the international movies are listed under comedies.

-> There is a significant amount of correlation between dramas and independent movies.

-> There are some interesting missing overlaps in categories, such as Faith and Spirituality doesn’t seem to overlap with LGBTQ movies.

TV SHOWS

-> There is a strong correlation between Tv dramas and international Tv shows. There is a significant amount of correlation between crime Tv shows and Tv dramas, romantic Tv shows and international, and crime Tv shows and international Tv shows. This suggests many of the international Tv shows are listed under Tv dramas. other of the international Tv shows are listed under crime Tv shows and romantic Tv shows.

CONCLUSION

We successfully cleaned the data and learned the following:

-> The duration of movies is a white noise time series and there is no correlation between any two observations.

-> Movies and Tv shows released in the mind 1900’s are still significant and available online.

-> Maximum number of seasons are available for Tv-14,Tv-MA, and TV-PG type shows.

-> RaÃºl Campos Jan Suter is the director with maximum content.

-> Most of the content is listed in as dramas and international.

APPENDIX

```r
knitr::opts_chunk$set(echo = TRUE)
library("tidyr")
library("ggplot2")
library('tidyverse')
library('data.table')
library('anytime')      # Change date format
library('dplyr')
library("forecast")     # Plot of time series
library("tseries")      # Time series significant test
library('RColorBrewer') # color change of plots
library("lemon")        # Pretty printing of data frames
kint_print.data.frame <- lemon_print

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# EXTRACT TRANSFORM LOAD
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

# Load the data set
data_set = read.csv("./netflix_titles.csv",header=TRUE)

# Creating a vector from the data set
Director = data_set$director

# Removing unwanted columns
data_set <- subset(data_set,select=-c(description,director))


# Removing empty rows
data_set = data_set[!(data_set$rating==""),]
data_set = data_set[!(data_set$cast==""),]
data_set = data_set[!(data_set$country==""),]
data_set = data_set[!(data_set$date_added==""),]


# data count after cleaning the data
summary(data_set)
head(data_set,5)


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# DURATION OF MOVIES
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

# Filtering different types of content available on Netflix
movie = data_set %>% filter(type=="Movie")
tv_show = data_set %>% filter(type=="TV Show")

# Slicing of character vector to remove 'min' from duration and converting
# to numeric vector.
movie_duration = as.numeric(substr(movie$duration,0,3))

# Plotting the duration of movies as a Time Series
tsdisplay(movie_duration, col = "red")  #(package forecast)

# summary
summary(movie_duration)
sd(movie_duration)

# De-meaning the above plot and re-plotting
movie_duration = (movie_duration-mean(movie_duration))
tsdisplay(movie_duration, col = "green")


# testing stationarity (package tseries)
adf.test(diff(movie_duration), k=0)
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# YEAR ADDED AND YEAR RELEASED
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

# Changing the dates format (package anytime)
date = data_set$date_added = as.Date(anytime(data_set$date_added))

# change full date added to just year added
year_added = as.numeric(format(date, format="%Y"))

# adding a new column of year added
data_set$year_added = year_added

# histogram of year added (package RColorBrewer)
hist(year_added, col = brewer.pal(6,'Set2'), xlab = "Year Added",
     main = "Frequency of content added over the years")

# histogram of year released (package RColorBrewer)
hist(data_set$release_year, col = brewer.pal(6,'Set2') , xlab = "Year Released",
     main = "Frequency of content released over the years")

# Creating a table of the oldest content available on Netflix

year_released = head(sort(data_set$release_year, decreasing = F),356)
top_oldcontent = head(data_set[order(data_set$release_year),][3],356)
year_Added = head(data_set[order(data_set$release_year),][11],356)
Typ = head(data_set[order(data_set$release_year),][2],356)
old = cbind(Typ, year_released,top_oldcontent,year_Added)

head(old)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# DATA FRAME EXPLAININFG THE MEANING OF RATINGS
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

# Creating the vectors
Ratings = c("NR","R","TV-14","TV-G","TV-MA","TV-PG","TV-Y","TV-Y7","TV-Y7-FV")

Fullform = c("Not Restricted","Restricted",
             "May not be suitable for ages 14 and under,parental guidance suggested",
             "General Audience",
             "May not be suitable for ages 17 and under, Mature Audience",
             "Parental Guidance suggested","Suitable for Children of all ages",
             "Suitable for ages 7+","Suitable for ages 7+, Fantasy Violence")

# Creating a data frame
rating_key = data.frame(Ratings,Fullform)

rating_key
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# find_ratings HELP FILE
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

find_ratings_help <- function(){

  cat("

  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
  # WELCOME TO HELP FILE OF find_ratings FUNCTION
  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

  -> To access this file type the following command find_ratings_help()

  -> The following function takes an argument in the form of the Data Frame.
  -> The data frame is divided in to 7 Seasons and 9 Ratings of TV - Shows.

  => To use the function call: find_ratings(data_frame_name)
```

```
  -> In general the function has only one input and it returns a data frame in
     the output.
  -> The output is ratings_df data frame, which contains the count of TV Shows
     with respect to their ratings and duration.

  ")

}

find_ratings_help()

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# FUNCTION CREATION: FIND RATING TO GENERATE SEASON WISE RATINGS
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

find_ratings <- function(data_set){

  # Filtering the Tv shows season on the basis of no. of seasons

        season1 = data_set %>% filter(duration=="1 Season")
        season2 = data_set %>% filter(duration=="2 Seasons")
        season3 = data_set %>% filter(duration=="3 Seasons")
        season4 = data_set %>% filter(duration=="4 Seasons")
        season5 = data_set %>% filter(duration=="5 Seasons")
        season6 = data_set %>% filter(duration=="6 Seasons")
        season7 = data_set %>% filter(duration=="7 Seasons")

  # Concatenating the ratings season wise

        ratings = c(season1$rating,season2$rating,season3$rating,season4$rating,
                    season5$rating,season6$rating,season7$rating)

  # Creating a data frame of seasons and ratings

        rating_df = data.frame(matrix(ncol = 9, nrow = 7))

        colnames(rating_df) <- sort(unique(ratings))


        season1_length = length(unique(season1$rating))
        season1_cols = sort(unique(season1$rating))
        season1_rating = table(season1$rating)

        season2_length = length(unique(season2$rating))
        season2_cols = sort(unique(season2$rating))
        season2_rating = table(season2$rating)

        season3_length = length(unique(season3$rating))
        season3_cols = sort(unique(season3$rating))
        season3_rating = table(season3$rating)

        season4_length = length(unique(season4$rating))
        season4_cols = sort(unique(season4$rating))
        season4_rating = table(season4$rating)

        season5_length = length(unique(season5$rating))
        season5_cols = sort(unique(season5$rating))
        season5_rating = table(season5$rating)

        season6_length = length(unique(season6$rating))
        season6_cols = sort(unique(season6$rating))
        season6_rating = table(season6$rating)

        season7_length = length(unique(season7$rating))
        season7_cols = sort(unique(season7$rating))
        season7_rating = table(season7$rating)
```

```
  # Loop on column index

        for (i in 1:season1_length){

                rating_df[1, season1_cols[i]] <- season1_rating[[i]]
        }
        for (i in 1:season2_length){

                rating_df[2, season2_cols[i]] <- season2_rating[[i]]
        }
        for (i in 1:season3_length){

                rating_df[3, season3_cols[i]] <- season3_rating[[i]]
        }
        for (i in 1:season4_length){

                rating_df[4, season4_cols[i]] <- season4_rating[[i]]
        }
        for (i in 1:season5_length){

                rating_df[5, season5_cols[i]] <- season5_rating[[i]]
        }
        for (i in 1:season6_length){

                rating_df[6, season6_cols[i]] <- season6_rating[[i]]
        }
        for (i in 1:season7_length){

                rating_df[7, season7_cols[i]] <- season7_rating[[i]]
        }

        rating_df[is.na(rating_df)] <- 0

        rating_df <- cbind(
          SEASON=c("season1","season2","season3","season4","season5","season6","season7"),
          rating_df)

        return(rating_df)
}

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# FUNCTION CALL
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

rating_df <- find_ratings(data_set)
rating_df

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# GGPLOT SEASON VS RATINGS
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

cols_to_select = colnames(rating_df)
cols_to_select <- cols_to_select[-1]

ggdata <- gather(data=rating_df,key=RATINGS,value=count,c(cols_to_select))
ggdata <- subset(ggdata,select=c(RATINGS,count,SEASON))

# Stacked Barplot

ggplot(ggdata, aes(fill=RATINGS,y = count , x=SEASON, order = RATINGS)) +
  geom_bar(position="stack", stat="identity") +
  ggtitle("Stack Bar Graph of rating of TV shows per season") +
  ylab("No. of TV Shows")


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
```

```
# DIRECTORS WITH MAXIMUM CONTENT
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

# Removing null values
Director = Director[Director!=""]

# Selecting the top 6 directors
top_director = head(sort(table(Director),decreasing = T))

# Creating a data frame for the plot
director_df = data.frame(top_director)

# Barplot of top 6 directors
x <- barplot(top_director, xaxt="n", col = brewer.pal(6,'Set3')) #(package RColorBrewer)
text(cex = 1,x=x-0.25,y=-1.5,director_df$Director, xpd = T, srt = 20)


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# CALLING THE DATA FRAME: director_df
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

director_df
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# find_ratings HELP FILE
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
find_corr_help <- function(){

  cat("

  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
  # WELCOME TO HELP FILE OF find_corr FUNCTION
  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

  -> To access this file type the following command find_corr_help()

  -> The following function takes an argument in the form of the Data Frame.
  -> The data frame is divided in to 2 categories and the count of the content

  => To use the function call: find_corr(data_frame_name)

  -> In general the function has only one input and it returns two plots in
     the output.
  -> The outputs are correlation plots of the categories of movies and Tv shows.

  ")

}

find_corr_help()


# CORRELATION OF CATEGORIES
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#


find_corr <- function(data_set){

  # Filtering,removing missing values, and renaming the categories from the data
  show_categories <- data_set %>%
                     select(c('show_id','type','listed_in')) %>%
                     separate_rows(listed_in, sep = ',') %>%
                     rename(Category = listed_in)

  # Remove white space
  show_categories$Category <- trimws(show_categories$Category)

  # Group and count type of content and their unique categories
```

```
  unique_category <- show_categories %>% group_by(type,Category) %>%  summarise()

  # Creating a data frames
  category_corr_movies<-data.frame(expand_grid(type = 'Movie',
                       Category1 = subset(unique_category, type == 'Movie')$Category,
                       Category2 = subset(unique_category, type == 'Movie')$Category))

  category_corr_TV <-data.frame(expand_grid(type = 'TV Show',
                    Category1 = subset(unique_category, type == 'TV Show')$Category,
                    Category2 = subset(unique_category, type == 'TV Show')$Category))

  # Binding the data frames created
  category_corr <- rbind(category_corr_movies,category_corr_TV)

  category_corr$matched_count <- apply(category_corr, MARGIN = 1,
      FUN = function(x) {length(intersect(subset(show_categories,
                         type == x['type'] & Category == x['Category1'])$show_id,
    subset(show_categories, type == x['type'] & Category == x['Category2'])$show_id))})

  category_corr <- subset(category_corr,
    (as.character(Category1) < as.character(Category2)) & (matched_count > 0))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# GGPLOT CORRELATION OF CATEGORIES
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

  # Correlation plot of movies
  print(ggplot(subset(category_corr, type == 'Movie'),
    aes(x = Category1, y = Category2, fill = matched_count)) +
    geom_tile() + facet_wrap( ~type, scales = 'free') + theme_classic() +
    scale_fill_distiller(palette = "Spectral") +
    theme(axis.text.x = element_text(angle = 90) ,
    legend.text = element_text(size = 14), legend.title = element_text(size = 16)))

  # Correlation plot of Tv shows
  print(ggplot(subset(category_corr, type == 'TV Show'),
    aes(x = Category1, y = Category2, fill = matched_count)) +
    geom_tile() + facet_wrap( ~type, scales = 'free') + theme_classic() +
    scale_fill_distiller(palette = "Spectral") +
    theme(axis.text.x = element_text(angle = 90),
    legend.text = element_text(size = 14), legend.title = element_text(size = 16))
  )

}
corr_df<- find_corr(data_set)
```