BUAN6356 – Group Project

TRENDING YOUTUBE VIDEO STATISTICS
| BUAN6356 | 28-NOV-2018

Aarya | AXX160530 |

Contents

1 Background

YouTube is one of the world's famous video sharing website. 300 hours of video are uploaded to YouTube every minute. Almost 5 billion videos are watched on YouTube every single day.

YouTube allows users to upload, view, rate, share, add to favorites, report, comment on videos, and subscribe to other users. It offers a wide variety of user-generated and corporate media videos. Available content includes video clips, TV show clips, music videos, short and documentary films, audio recordings, movie trailers, live streams, and other content such as video blogging, short original videos, and educational videos. Most of the content on YouTube is uploaded by individuals, but media corporations including CBS, the BBC, Vevo, and Hulu offer some of their material via YouTube as part of the YouTube partnership program.

As stated, since many videos are uploaded each minute, based on several factors, the videos are getting trended worldwide. There are a lot of factors considered for trending which not only include views, but also other interactions like shares, comments, likes, and dislikes. In this project, we have taken one such dataset from Kaggle, which contains the daily record of the top trending YouTube records. The aim of the project is to perform extensive data analysis using R, apply data mining techniques on various factors which are considered for making the video trending.

2 Objectives

The objectives of the project are to perform extensive exploratory data analysis on the dataset taken from Kaggle site and find some interesting relations and meaning from the data. Apply various data mining technique which we learned in BUAN6356 class which includes Linear Regression for understanding and predicting the relationship between the various numerical variables in the dataset, Logistic regression, KNN, CART models to predict the category id and compare the results of the technique and find the best model. Also, to perform sentiment analysis on the video description and title and draw various bar charts based on the sentiments and create a word cloud on words that are frequently used in title and descriptions for the trending videos. The analysis on the data is carried region wise viz. United States, Canada, Germany, France, and Great Britain. Also, analysis on combined data set is also carried out and the interesting relations and plots are drawn.

2.1 Dataset

The dataset is taken from Kaggle site "Trending YouTube Video Statistics". According to Variety magazine, "To determine the year's top-trending videos, YouTube uses a combination of factors including measuring users' interactions (number of views, shares, comments, and likes). Note that they're not the most-viewed videos overall for the calendar year". Top performers on the YouTube trending list are music videos (such as the famously virile "Gangnam Style"), celebrity and/or reality TV performances, and the random dude-with-a-camera viral videos that YouTube is well-known for.

This dataset includes several months (and counting) of data on daily trending YouTube videos. Data is included for the US, GB, DE, CA, and FR regions (USA, Great Britain, Germany, Canada, and France, respectively), with up to 200 listed trending videos per day.

Each region's data is in a separate file. Data includes the video title, channel title, publish time, tags, views, likes and dislikes, description, and comment count.

The data also includes a category_id field, which varies between regions. To retrieve the categories for a specific video, find it in the associated JSON. One such file is included for each of the five regions in the dataset.

The below table shows the description of the dataset which is considered for the project. The dataset can be availed in the below link from Kaggle.

https://www.kaggle.com/datasnaek/youtube-new

That data was collected for several countries: US (United States of America), GB (Great Britain), DE(Germany), CA(Canada), and FR(France). we have modified the dataset to analyze some hidden information. Such as, we removed duplicate video_id's and make use of them to retrieve some meaningful data. we removed some unrelated attributes. As per our requirements, we changed the type/class of few attributes too. We have derived some new attributes from the existing one.

All the modifications are done by R-Programming.

2.2 Techniques Used

We have used almost all the techniques which we have learned in the BUAN6356 class. Which includes correlation matrix to find the correlation matrix between the numerical variables in the dataset, extensive exploratory data analysis is carried out with various plots like bar plots, correlation plots, heat map, scatter plot etc. Various regression techniques like Linear regression to predict the numerical values like comment count, likes, view, and classification techniques like Logistic regression, KNN, CART to predict the category ID of the videos based on the independent variables in the dataset. Then a comparison is drawn based on the confusion matrix and accuracy of the model and the best model is then chosen. The features of the algorithms are discussed under each section where it is used, and their strength and weaknesses are discussed accordingly.

3 Exploratory Data Analysis:

3.1 Initial data loading and data manipulation:

The dataset is in .csv format and there are 5 csv file each for one region and 5 .json file which contains the category details of the dataset. These files are loaded into R and are manipulated, and few unwanted columns are removed from the data frame as they are not useful for further analysis. The following piece of code is used for the same

```
1.  library(data.table)
2.  library(syuzhet)
3.  library(rjson)
4.  library(jsonlite)
5.  library(plyr)
6.  library(stats)
7.  library(tidyverse)
8.  library(leaps)
9.  library(gains)
10. library(ggplot2)
11. library(lattice)
12. library(caret)
13. #date manipulation
14. library(lubridate)
15.
16. #data viz
17. library(corrplot)
18. library(gplots)
19. library(psych)
20. library(ggplot2)
21.
22. library(tm)
23. library(wordcloud)
24. library(RColorBrewer)
25. library(RCurl)
26.
27. #import data
28. us_videos.df <- read.csv("USvideos.csv")
29. gb_videos.df <- read.csv("GBvideos.csv")
30. ca_videos.df <- read.csv("CAvideos.csv")
31. de_videos.df <- read.csv("DEvideos.csv")
32. fr_videos.df <- read.csv("FRvideos.csv")
33.
34. #Data Cleaning:
35.
36. #since our analysis is on videos and comments based on videos, removing video
    s which has
37. #comments disabled and video has error or removed
38.
39. us_videos.df <- us_videos.df[!(us_videos.df$video_error_or_removed == "True"
    || us_videos.df$comments_disabled == TRUE),]
40. gb_videos.df <- gb_videos.df[!(gb_videos.df$video_error_or_removed == "True"
    || gb_videos.df$comments_disabled == TRUE),]
```

```r
ca_videos.df <- ca_videos.df[!(ca_videos.df$video_error_or_removed == "True" || ca_videos.df$comments_disabled == TRUE),]
de_videos.df <- de_videos.df[!(de_videos.df$video_error_or_removed == "True" || de_videos.df$comments_disabled == TRUE),]
fr_videos.df <- fr_videos.df[!(fr_videos.df$video_error_or_removed == "True" || fr_videos.df$comments_disabled == TRUE),]

#Dimension reduction: Removing fields which are not useful for analysis like

#comments disabled, ratings disabled as they are not going to give us any meaning.

us_videos.df <- us_videos.df[, !(colnames(us_videos.df) %in% c("comments_disabled","ratings_disabled","video_error_or_removed"))]
gb_videos.df <- gb_videos.df[, !(colnames(gb_videos.df) %in% c("comments_disabled","ratings_disabled","video_error_or_removed"))]
ca_videos.df <- ca_videos.df[, !(colnames(ca_videos.df) %in% c("comments_disabled","ratings_disabled","video_error_or_removed"))]
de_videos.df <- de_videos.df[, !(colnames(de_videos.df) %in% c("comments_disabled","ratings_disabled","video_error_or_removed"))]
fr_videos.df <- fr_videos.df[, !(colnames(fr_videos.df) %in% c("comments_disabled","ratings_disabled","video_error_or_removed"))]

#Adding location to datasets

us_videos.df$Location = "US"
gb_videos.df$Location = "GB"
ca_videos.df$Location = "CA"
de_videos.df$Location = "DE"
fr_videos.df$Location = "FR"

# reading category JSON file and importing to R
us_cat_json <- fromJSON("US_category_id.json")
gb_cat_json <- fromJSON("GB_category_id.json")
ca_cat_json <- fromJSON("CA_category_id.json")
de_cat_json <- fromJSON("DE_category_id.json")
fr_cat_json <- fromJSON("FR_category_id.json")

#converting the list to dataframe to add to data files

US_category <-  as.data.frame(cbind(us_cat_json[["items"]][["id"]], us_cat_json[["items"]][["snippet"]][["title"]]))
GB_category <-  as.data.frame(cbind(gb_cat_json[["items"]][["id"]], gb_cat_json[["items"]][["snippet"]][["title"]]))
CA_category <-  as.data.frame(cbind(ca_cat_json[["items"]][["id"]], ca_cat_json[["items"]][["snippet"]][["title"]]))
DE_category <-  as.data.frame(cbind(de_cat_json[["items"]][["id"]], de_cat_json[["items"]][["snippet"]][["title"]]))
FR_category <-  as.data.frame(cbind(fr_cat_json[["items"]][["id"]], fr_cat_json[["items"]][["snippet"]][["title"]]))

#adding names to fields
names(US_category) <- c("category_id","category_title")
names(GB_category) <- c("category_id","category_title")
names(CA_category) <- c("category_id","category_title")
names(DE_category) <- c("category_id","category_title")
names(FR_category) <- c("category_id","category_title")

#Merging the categories desc to country file based on category id for all country files
```

```
us_videos.df <- merge(x = us_videos.df, y = US_category, by = "category_id",
all = "TRUE")
gb_videos.df <- merge(x = gb_videos.df, y = GB_category, by = "category_id",
all = "TRUE")
ca_videos.df <- merge(x = ca_videos.df, y = CA_category, by = "category_id",
all = "TRUE")
de_videos.df <- merge(x = de_videos.df, y = DE_category, by = "category_id",
all = "TRUE")
fr_videos.df <- merge(x = fr_videos.df, y = FR_category, by = "category_id",
all = "TRUE")
#removing NA in case any after adding category title

us_videos.df <- na.omit(us_videos.df)
gb_videos.df <- na.omit(gb_videos.df)
ca_videos.df <- na.omit(ca_videos.df)
de_videos.df <- na.omit(de_videos.df)
fr_videos.df <- na.omit(fr_videos.df)

#converting category id from int to numeric
class(us_videos.df$category_id)
      us_videos.df$category_id <- as.numeric(us_videos.df$category_id)
      gb_videos.df$category_id <- as.numeric(gb_videos.df$category_id)
      ca_videos.df$category_id <- as.numeric(ca_videos.df$category_id)
      de_videos.df$category_id <- as.numeric(de_videos.df$category_id)
      fr_videos.df$category_id <- as.numeric(fr_videos.df$category_id)
      #combining videoes from all country
      Youtube_videos.df <- as.data.table(rbind(us_videos.df,fr_videos.df,ca_
videos.df,us_videos.df,de_videos.df)) nrow(Youtube_videos.df)
      Youtube_videos.df$category_id = as.numeric(Youtube_videos.df$category_
id)
```

Table 3.1 Table showing R Code for Initial Dataload

```
> str(us_videos.df)
'data.frame':   40949 obs. of  16 variables:
 $ category_id   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ video_id      : Factor w/ 6351 levels "-_jlqATo9eo",..: 2442 3507 4867 5596 873 5514 1769 2870 6212 3507 ...
 $ trending_date : Date, format: "2018-02-26" "2018-03-12" "2018-01-17" ...
 $ title         : Factor w/ 6455 levels "'Avengers: Infinity War' Cast Tours Los Angeles w/ James Corden",..: 2350 1782 704 566 839 3159 4662 2148 2481 1782 ...
 $ channel_title : Factor w/ 2207 levels "12 News","1MILLION Dance Studio",..: 1695 401 234 854 4 965 1643 769 1639 401 ...
 $ publish_time  : Date, format: "2018-02-20" "2018-02-27" "2018-01-11" ...
 $ tags          : Factor w/ 6055 levels "#guitar #musiciseverywhere #jammin #meme #funny #deeppurple #pinkfloyd",..: 4631 631 664 357 5559 12 4490 12 4470 631 ...
 $ views         : int  2633079 1080795 5186780 259340 12887949 153898 705015 228745 28013 945670 ...
 $ likes         : int  69999 23939 510 9059 338755 7551 37505 1848 2374 21927 ...
 $ dislikes      : int  2377 950 1774 177 6727 1028 1413 41 15 832 ...
 $ comment_count : int  10501 2282 665 934 19502 1904 16044 144 102 2131 ...
 $ thumbnail_link: Factor w/ 6352 levels "https://i.ytimg.com/vi/-_jlqATo9eo/default.jpg",..: 2442 3507 4868 5597 873 5515 1769 2870 6213 3507 ...
 $ description   : Factor w/ 6902 levels "","'A curious cat helps his owner with home improvements.'\\nWe're releasing a NEW BLACK & WHITE episode every wee"| __truncated__,..: 4970 812 4134 6406 6288 5632 4
730 3047 315 812 ...
 $ Location      : chr  "US" "US" "US" "US" ...
 $ trendingday   : 'difftime' num  6 13 6 6 ...
  ..- attr(*, "units")= chr "days"
 $ category_title: Factor w/ 31 levels "Action/Adventure",..: 11 11 11 11 11 11 11 11 11 11 ...
 - attr(*, "na.action")= 'omit' Named int  11912 13516 40895 40896 40897 40898 40899 40900 40901 40902 ...
  ..- attr(*, "names")= chr  "11912" "13516" "40895" "40896" ...
> str(Youtube_videos.df)
Classes 'data.table' and 'data.frame':  203899 obs. of  16 variables:
 $ category_id   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ video_id      : Factor w/ 77648 levels "-_jlqATo9eo",..: 2442 3507 4867 5596 873 5514 1769 2870 6212 3507 ...
 $ trending_date : Date, format: "2018-02-26" "2018-03-12" "2018-01-17" ...
 $ title         : Factor w/ 77906 levels "'Avengers: Infinity War' Cast Tours Los Angeles w/ James Corden",..: 2350 1782 704 566 839 3159 4662 2148 2481 1782 ...
 $ channel_title : Factor w/ 16015 levels "12 News","1MILLION Dance Studio",..: 1695 401 234 854 4 965 1643 769 1639 401 ...
 $ publish_time  : Date, format: "2018-02-20" "2018-02-27" "2018-01-11" ...
 $ tags          : Factor w/ 64569 levels "#guitar #musiciseverywhere #jammin #meme #funny #deeppurple #pinkfloyd",..: 4631 631 664 357 5559 12 4490 12 4470 631 ...
 $ views         : int  2633079 1080795 5186780 259340 12887949 153898 705015 228745 28013 945670 ...
 $ likes         : int  69999 23939 510 9059 338755 7551 37505 1848 2374 21927 ...
 $ dislikes      : int  2377 950 1774 177 6727 1028 1413 41 15 832 ...
 $ comment_count : int  10501 2282 665 934 19502 1904 16044 144 102 2131 ...
 $ thumbnail_link: Factor w/ 77629 levels "https://i.ytimg.com/vi/-_jlqATo9eo/default.jpg",..: 2442 3507 4868 5597 873 5515 1769 2870 6213 3507 ...
 $ description   : Factor w/ 67725 levels "","'A curious cat helps his owner with home improvements.'\\nWe're releasing a NEW BLACK & WHITE episode every wee"| __truncated__,..: 4970 812 4134 6406 6288 5632
4730 3047 315 812 ...
 $ Location      : chr  "US" "US" "US" "US" ...
 $ trendingday   : 'difftime' num  6 13 6 6 ...
  ..- attr(*, "units")= chr "days"
 $ category_title: Factor w/ 31 levels "Action/Adventure",..: 11 11 11 11 11 11 11 11 11 11 ...
 - attr(*, "na.action")= 'omit' Named int  11912 13516 40895 40896 40897 40898 40899 40900 40901 40902 ...
  ..- attr(*, "names")= chr  "11912" "13516" "40895" "40896" ...
 - attr(*, ".internal.selfref")=<externalptr>
```

Figure 3.1 Figure showing the structure of dataframe after importing/manipulating the dataset

3.2 Exploratory Data Analysis

3.2.1 Correlation between different numerical variables using correlation matrix – merged dataset for all regions together

The **Correlation is** the association between two or more variables. The correlation matrix is used to investigate the dependence between multiple variables at the same time. The result is a table containing the **correlation coefficients** between each variable and the others. There are three different methods of correlation "pearson", "kendall", "spearman". Here we have considered “pearson” method by default. (Kassambara, n.d.)

```
#Starting Exploratory Data Analysis:

#First starting with basic correlation matrix, find most correlated columns then plot the column which is most correlated.
#We are using the combined dataset for this
analysis assuming the trend will be same across countries.
#later extensive EDA will be performed on each country dataset and combined dataset.


#correlation matrix. Creating the correlation matrix for the numerical values and comparing the same
#using heat map and correlation plot.

cor.matrix = cor(Youtube_videos.df[,c("category_id","views","likes","dislikes","comment_count")])
cor.matrix


heatmap.2(cor.matrix, Rowv = FALSE, Colv = FALSE, dendrogram = "none",
          cellnote = round(cor.matrix,2),
          notecol = "black", key = FALSE, trace = 'none', margins = c(10,10), main = "Correlation Matrix")

corrplot.mixed(corr = cor.matrix, main = "Correlation Plot")
```

Table 3.2 Table showing correlation matrix and heat map for the correlation matrix

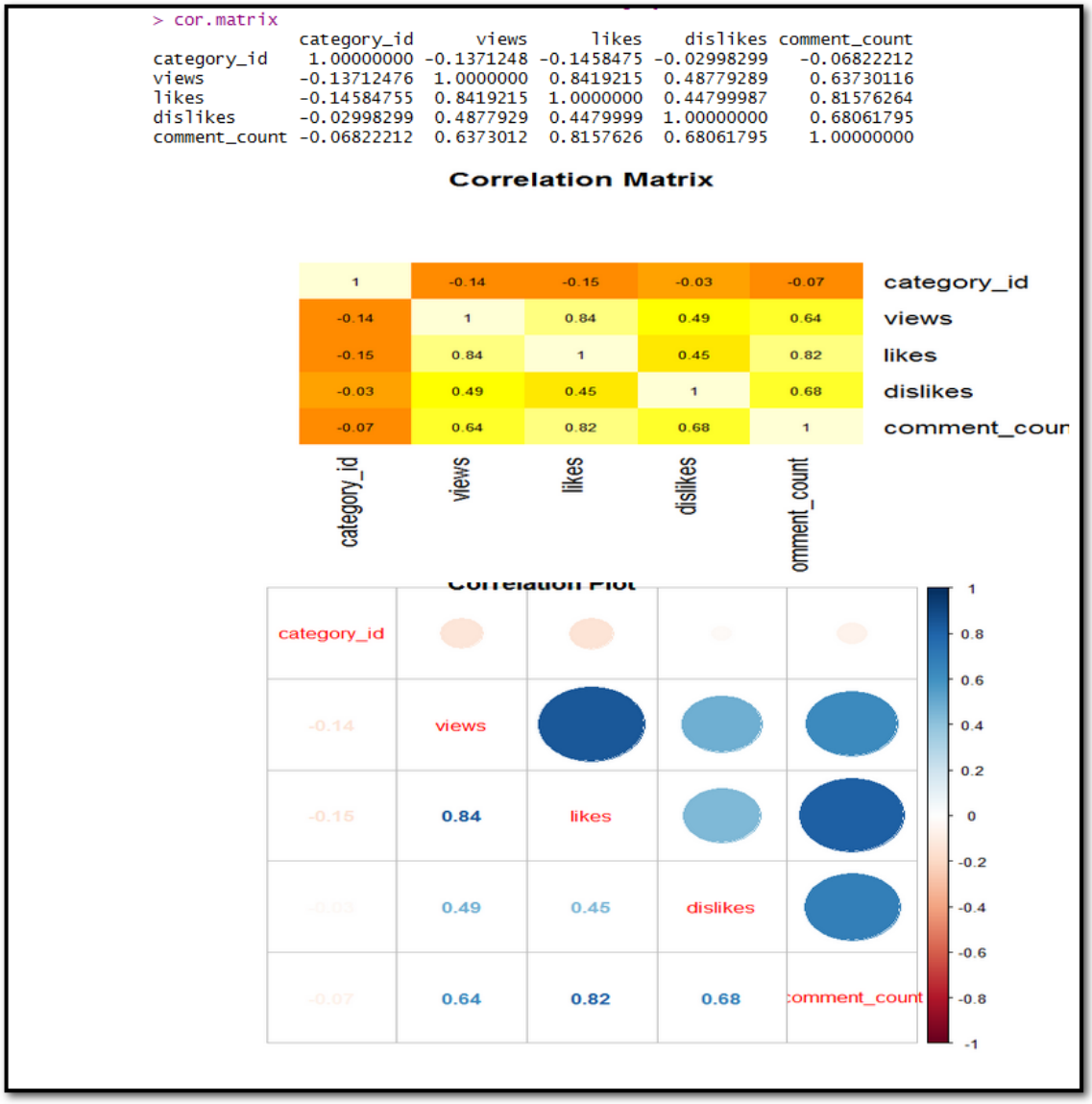


Figure 3.2 Figure showing Correlation matrix, heat map, and Correlation plot.

From the correlation matrix, heat map and the correlation plot, we can see that views and likes are highly correlated with each other. They have the highest correlation coefficient. We could infer that more the number of views, the more the chances of the video getting liked. The next high correlation is between the likes and comment count. Also, interestingly the correlation between dislikes and comment count is also larger.

Hence, we can conclude that people tend to comment on the video when they like or dislike the video rather viewing the video. This is the correlation matrix for the merged video.

3.2.1.1 *Results from Correlation of YouTube dataset*

The major result we could infer from this correlation test is the views count strongly correlates with likes, dislikes and comments count. We can conclude that as the number of comments increases or likes or dislikes increases, the views are also increasing. This is an interesting inference we deduced from this dataset. The below gives the summary of the coefficients of the correlation.

Linear Coefficient Correlation (method = pearson) between views & likes = 0.82

Linear Coefficient Correlation (method = pearson) between views & dislikes = 0.49

Linear Coefficient Correlation (method = pearson) between views & comment_count = 0.64

Point to see, all above correlations are positive & strong

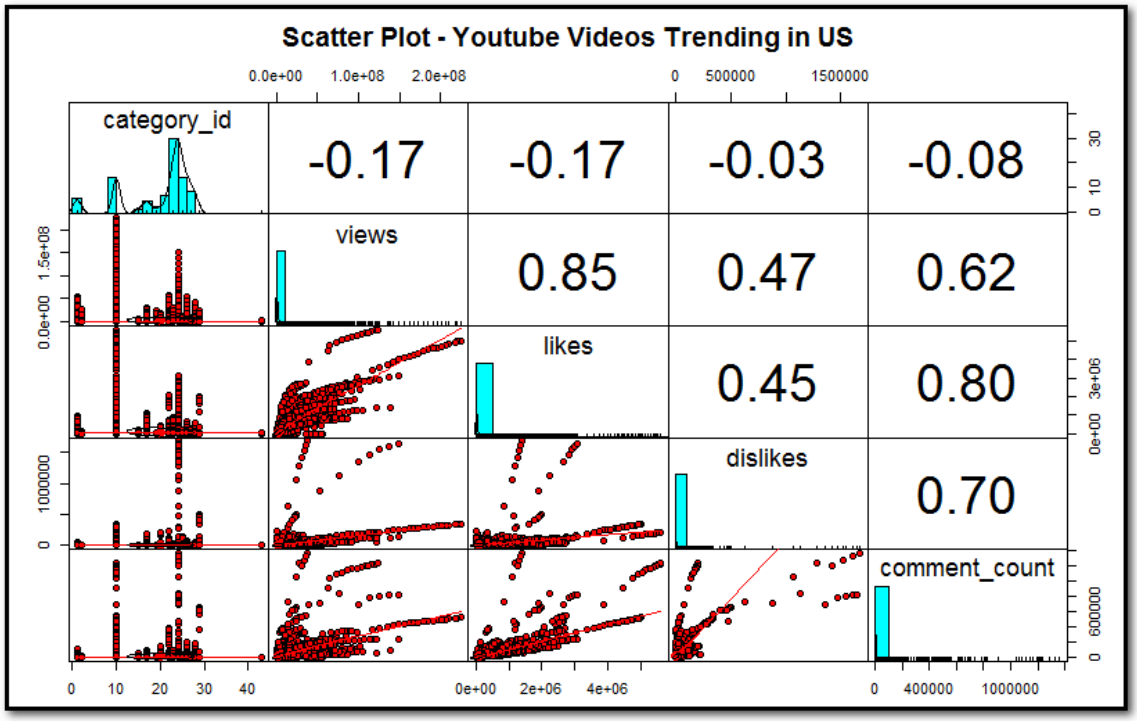
3.2.2 Panel plots for region wise data set

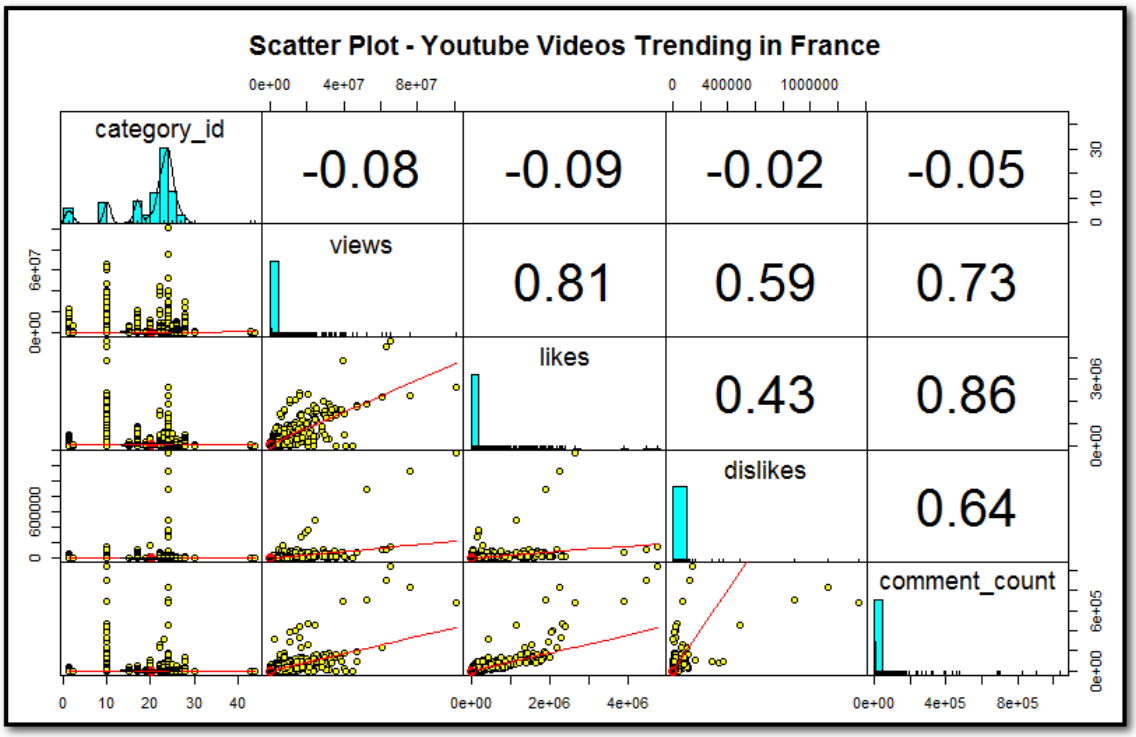
This section shows the panel plots for different region taken into consideration. The relation between the numerical fields are clearly explained graphically by these pair plots.

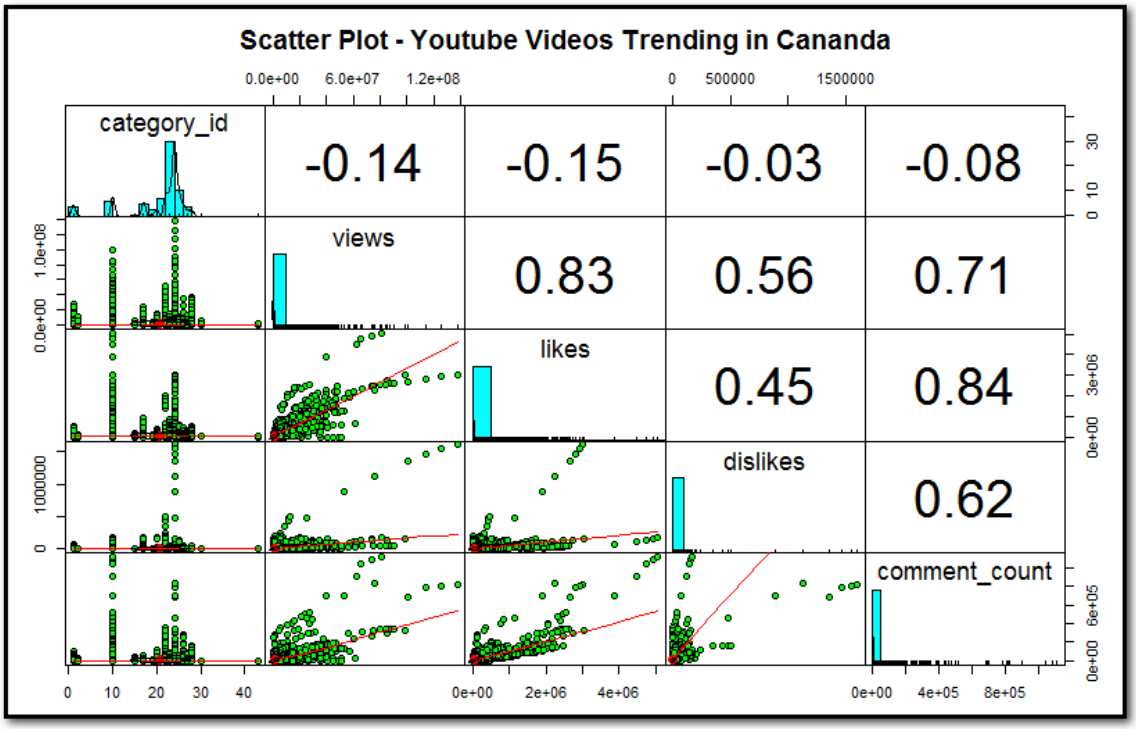
```
##Scatter Plots for correlation values

#pairs.panels(Youtube_videos.df[,c("category_id","views","likes","dislikes","comment_count")], )


pairs.panels(us_videos.df[,c("category_id","views","likes","dislikes","comment_count")],
             gap = 0,
             bg = c("red"),
             pch = 21,
             main = "Scatter Plot - Youtube Videos Trending in US")

pairs.panels(gb_videos.df[,c("category_id","views","likes","dislikes","comment_count")],
             gap = 0,
             bg = c("blue"),
             pch = 21,
             main = "Scatter Plot - Youtube Videos Trending in Great Britain"
)
```

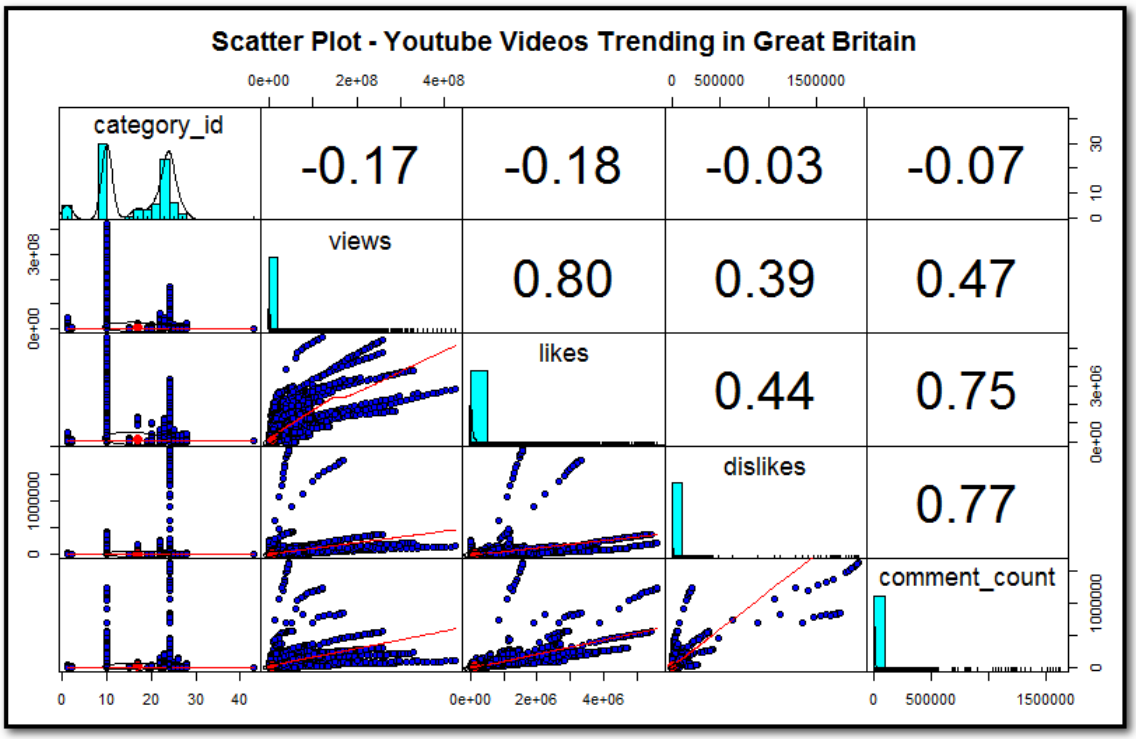
```
17.
18. pairs.panels(ca_videos.df[,c("category_id","views","likes","dislikes","commen
    t_count")],
19.              gap = 0,
20.              bg = c("green"),
21.              pch = 21,
22.              main = "Scatter Plot - Youtube Videos Trending in Cananda")
23.
24. pairs.panels(de_videos.df[,c("category_id","views","likes","dislikes","commen
    t_count")],
25.              gap = 0,
26.              bg = c("cyan"),
27.              pch = 21,
28.              main = "Scatter Plot - Youtube Videos Trending in Denmark")
29.
30. pairs.panels(fr_videos.df[,c("category_id","views","likes","dislikes","commen
    t_count")],
31.              gap = 0,
32.              bg = c("yellow"),
33.              pch = 21,
34.              main = "Scatter Plot - Youtube Videos Trending in France")
```

Table3.3 showing R code for Pair Panels for different Region




Scatter Plot - Youtube Videos Trending in France
category_id
views
likes
dislikes
comment_count
-0.08
-0.09
-0.02
-0.05
0.81
0.59
0.73
0.43
0.86
0.64


Scatter Plot - Youtube Videos Trending in Cananda
category_id
views
likes
dislikes
comment_count
-0.14
-0.15
-0.03
-0.08
0.83
0.56
0.71
0.45
0.84
0.62



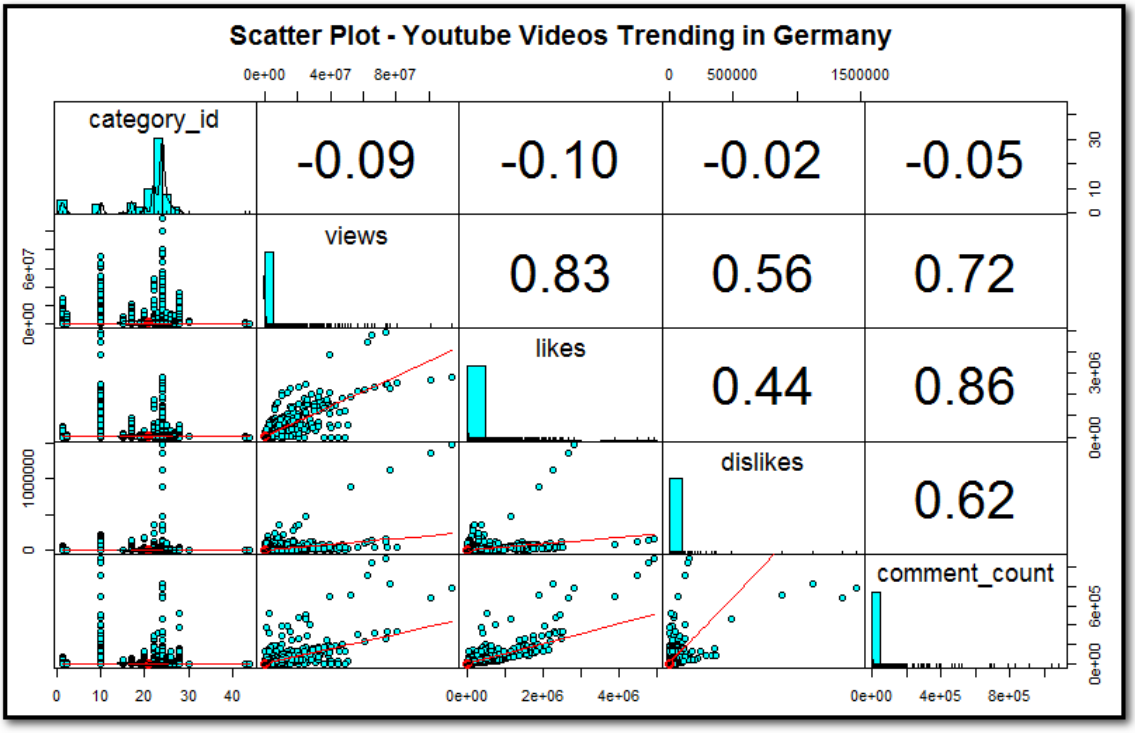


Figure3.3 Panel plots for different regions

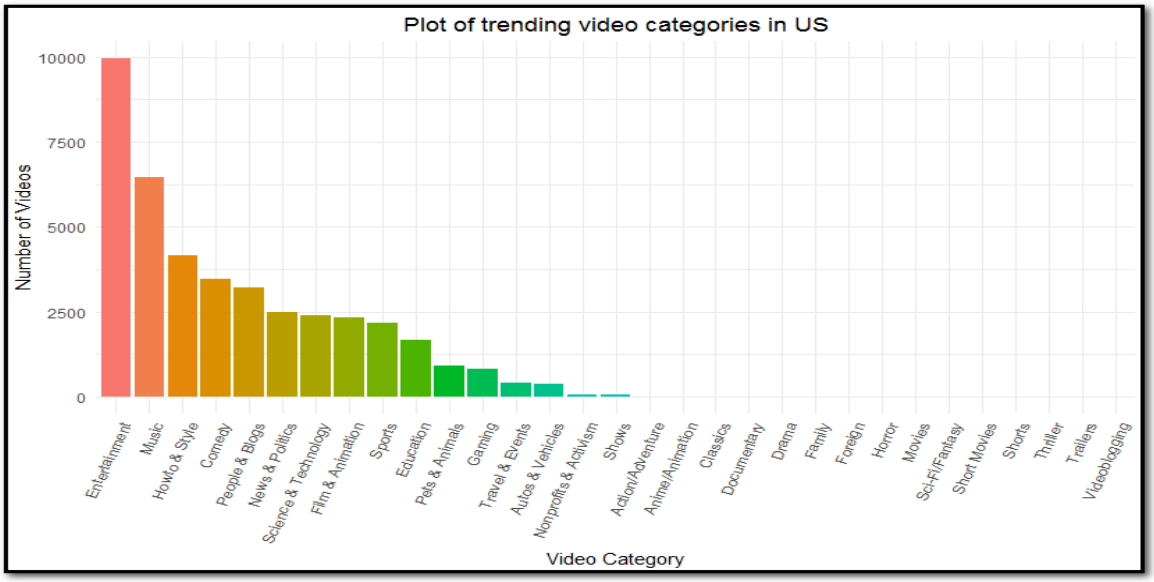
3.2.3 Trending videos in the different region based on Category

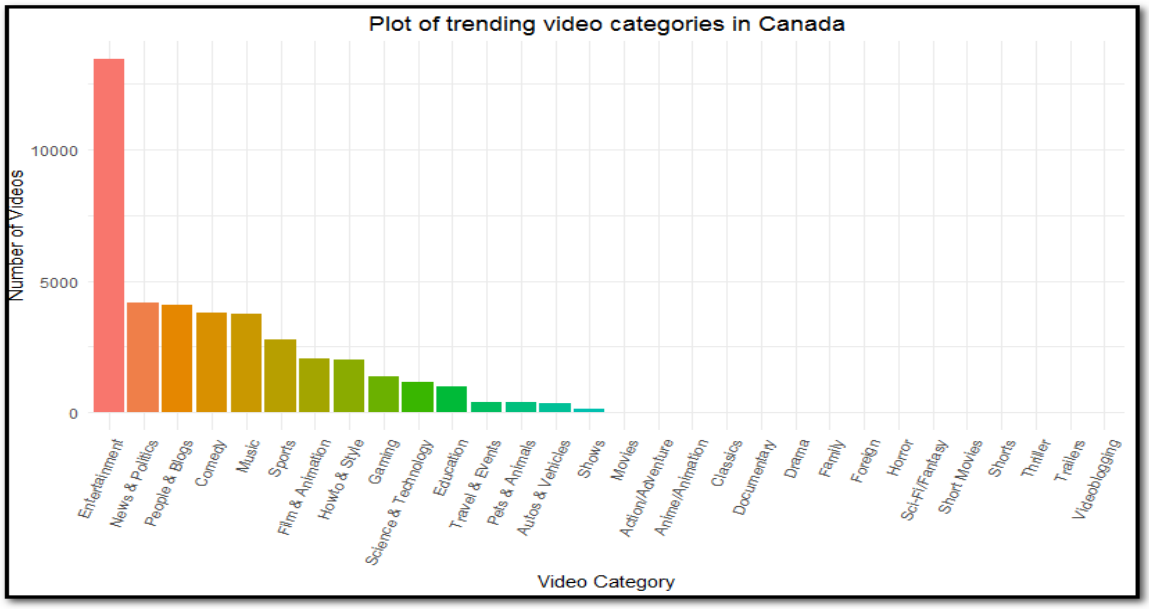
The section provides an in-depth data analysis of trending videos in different region based on the video categories. The bar chart self-explains the top trending video channels in different regions.

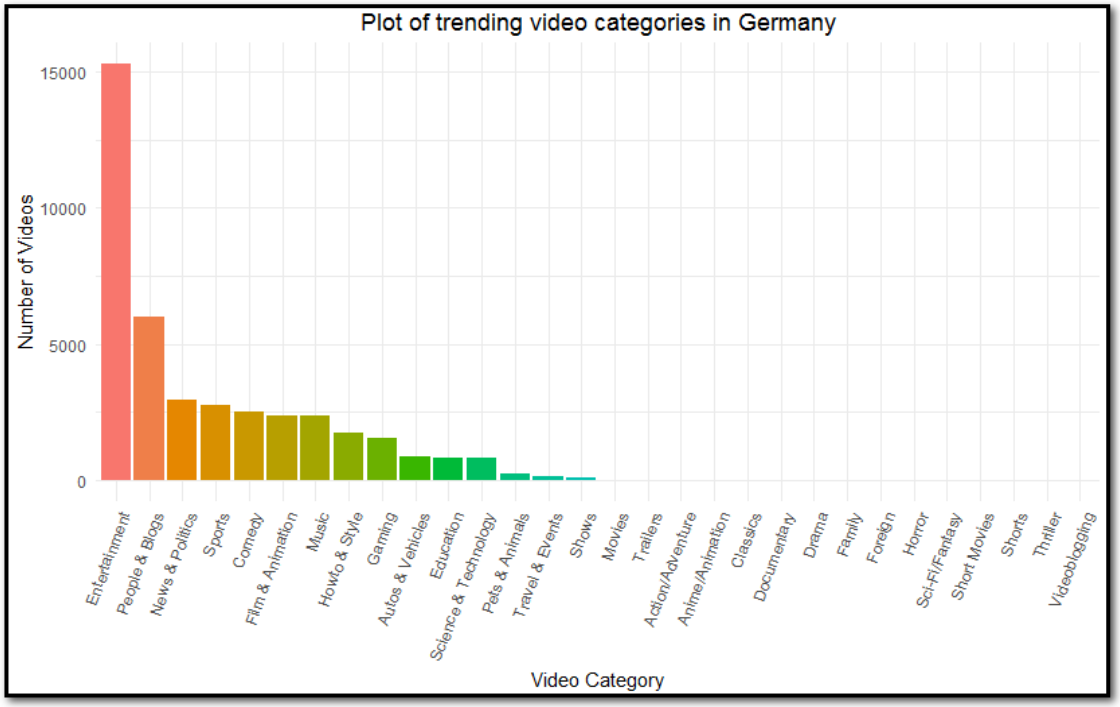
```
#Trending Videos Categories for each country <- done

#US
us_video_trend_count.df <- as.data.frame(sort(table(us_videos.df$category_title), decreasing = TRUE))

names(us_video_trend_count.df) <- c("category_title","count")


ggplot(us_video_trend_count.df, aes(x = category_title, y = count,fill = factor(category_title)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending video categories in US", x = "Video Category", y = "Number of Videos")

#Canada
ca_video_trend_count.df <- as.data.frame(sort(table(ca_videos.df$category_title), decreasing = TRUE))

names(ca_video_trend_count.df) <- c("category_title","count")


ggplot(ca_video_trend_count.df, aes(x = category_title, y = count,fill = factor(category_title)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending video categories in Canada", x = "Video Category", y = "Number of Videos")

#Germany
de_video_trend_count.df <- as.data.frame(sort(table(de_videos.df$category_title), decreasing = TRUE))

names(de_video_trend_count.df) <- c("category_title","count")


ggplot(de_video_trend_count.df, aes(x = category_title, y = count,fill = factor(category_title)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending video categories in Germany", x = "Video Category", y = "Number of Videos")

#France
fr_video_trend_count.df <- as.data.frame(sort(table(fr_videos.df$category_title), decreasing = TRUE))

names(fr_video_trend_count.df) <- c("category_title","count")


```

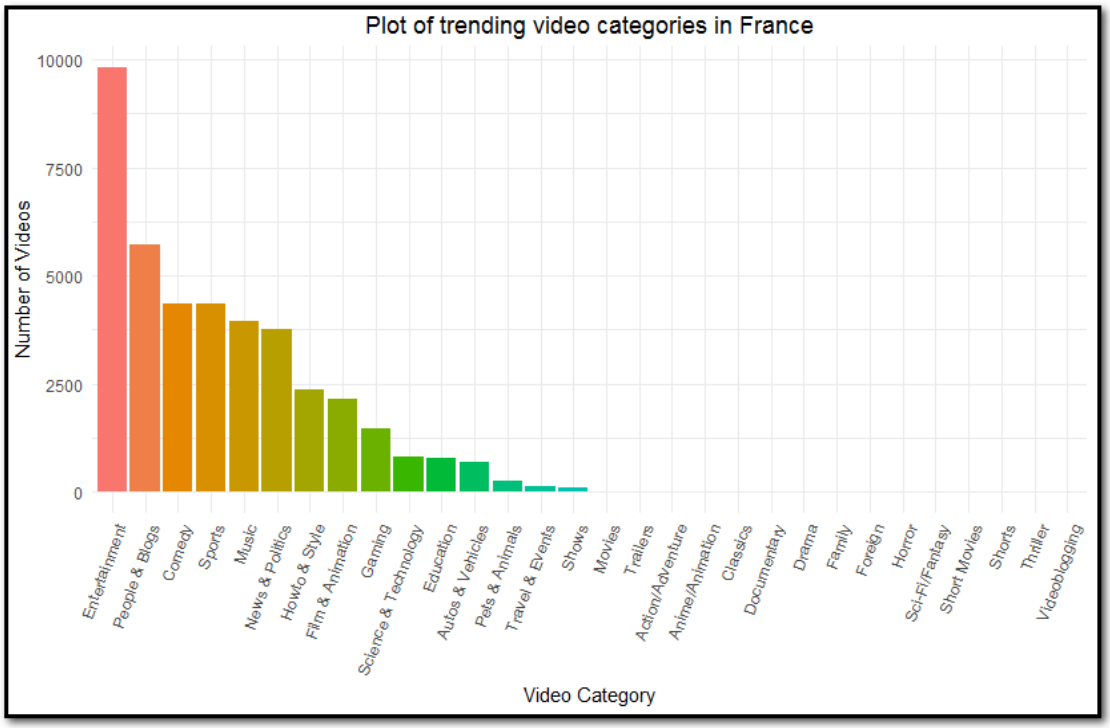
```r
ggplot(fr_video_trend_count.df, aes(x = category_title, y = count,fill = factor(category_title)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending video categories in France", x = "Video Category", y = "Number of Videos")

#United Kingdom
gb_video_trend_count.df <- as.data.frame(sort(table(gb_videos.df$category_title), decreasing = TRUE))

names(gb_video_trend_count.df) <- c("category_title","count")


ggplot(gb_video_trend_count.df, aes(x = category_title, y = count,fill = factor(category_title)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending video categories in UK", x = "Video Category", y = "Number of Videos")

#Overall
yt_video_trend_count.df <- as.data.frame(sort(table(Youtube_videos.df$category_title), decreasing = TRUE))

names(yt_video_trend_count.df) <- c("category_title","count")


ggplot(yt_video_trend_count.df, aes(x = category_title, y = count,fill = factor(category_title)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending video categories in 5 countries", x = "Video Category", y = "Number of Videos")
```

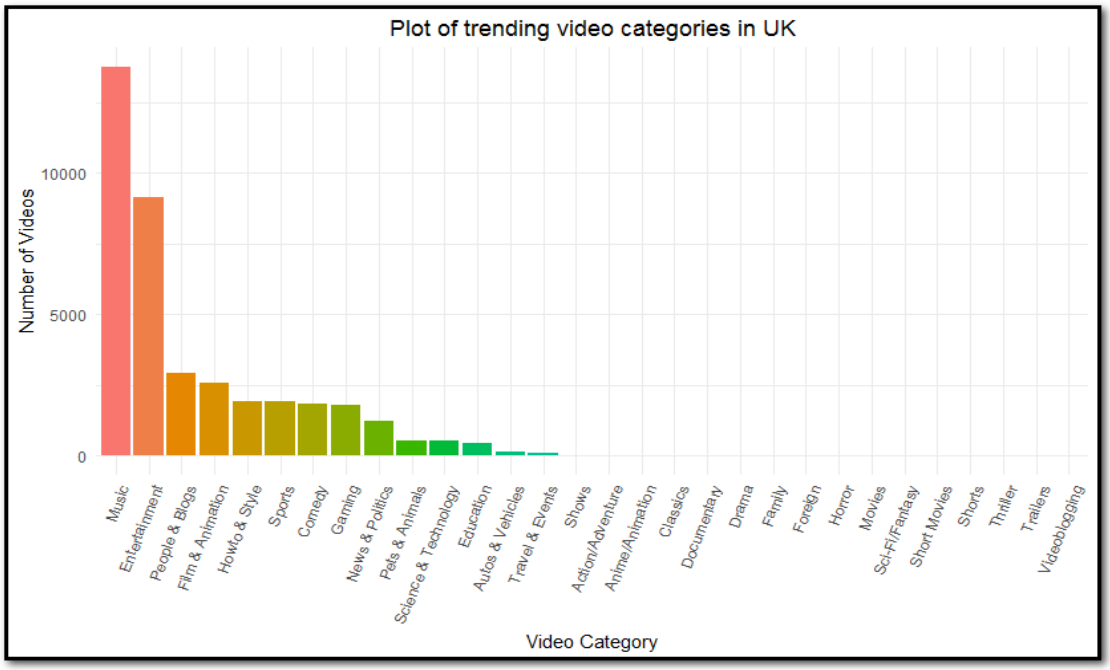
Table 3.4 R code for Trending video category in each region and merged dataset




Plot of trending video categories in Canada
Number of Videos
10000
5000
0
Entertainment
News & Politics
People & Blogs
Comedy
Music
Sports
Film & Animation
Howto & Style
Gaming
Science & Technology
Education
Travel & Events
Pets & Animals
Autos & Vehicles
Shows
Movies
Action/Adventure
Anime/Animation
Classics
Documentary
Drama
Family
Foreign
Horror
Sci-Fi/Fantasy
Short Movies
Shorts
Thriller
Trailers
Videoblogging
Video Category


Plot of trending video categories in Germany
Number of Videos
15000
10000
5000
0
Entertainment
People & Blogs
News & Politics
Sports
Comedy
Film & Animation
Music
Howto & Style
Gaming
Autos & Vehicles
Education
Science & Technology
Pets & Animals
Travel & Events
Shows
Movies
Trailers
Action/Adventure
Anime/Animation
Classics
Documentary
Drama
Family
Foreign
Horror
Sci-Fi/Fantasy
Short Movies
Shorts
Thriller
Videoblogging
Video Category


Plot of trending video categories in France
Number of Videos
10000
7500
5000
2500
0
Entertainment
People & Blogs
Comedy
Sports
Music
News & Politics
Howto & Style
Film & Animation
Gaming
Science & Technology
Education
Autos & Vehicles
Pets & Animals
Travel & Events
Shows
Movies
Trailers
Action/Adventure
Anime/Animation
Classics
Documentary
Drama
Family
Foreign
Horror
Sci-Fi/Fantasy
Short Movies
Shorts
Thriller
Videoblogging
Video Category


Plot of trending video categories in UK
Number of Videos
10000
5000
0
Music
Entertainment
People & Blogs
Film & Animation
Howto & Style
Sports
Comedy
Gaming
News & Politics
Pets & Animals
Science & Technology
Education
Autos & Vehicles
Travel & Events
Shows
Action/Adventure
Anime/Animation
Classics
Documentary
Drama
Family
Foreign
Horror
Movies
Sci-Fi/Fantasy
Short Movies
Shorts
Thriller
Trailers
Videoblogging
Video Category

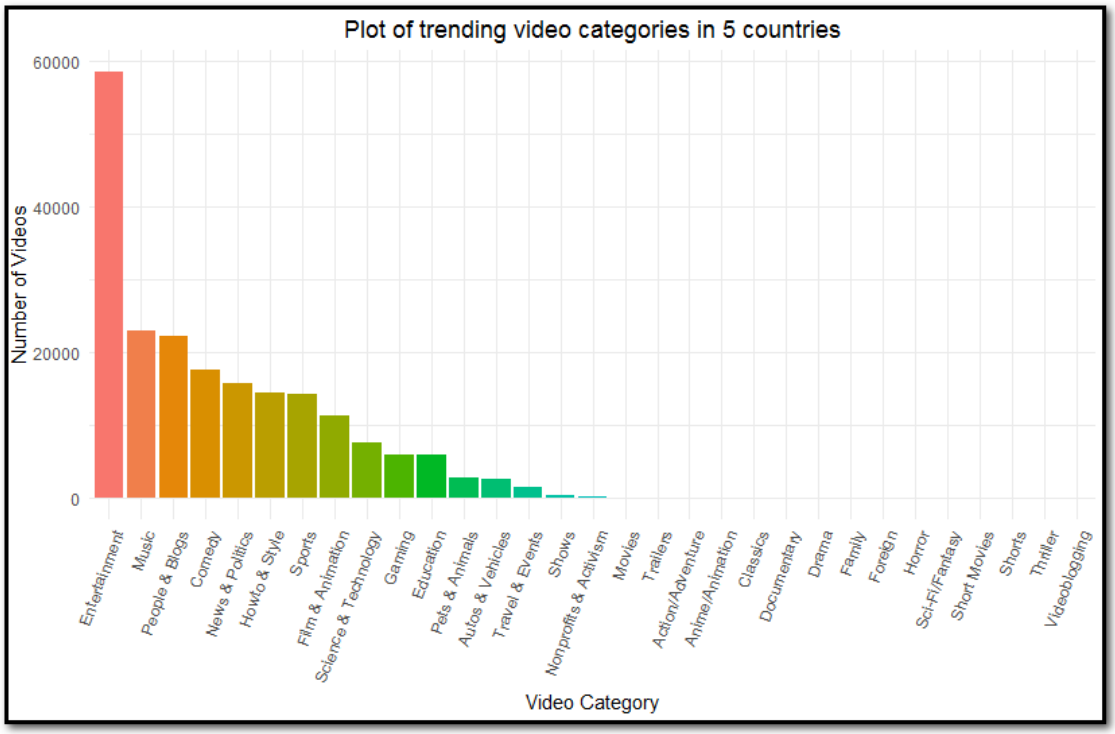


Figure3.4 Plot of trending videos for different region and merged dataset

The most frequent video category in 5countries is Entertainment. All of the countries have a large gap between Entertainment and second category except the United Kingdom. The UK is the only country that Entertainment took a second place which follows the music category. This result can show how much British have interested in Music. The second frequent video was music. Interest point is, music was only 2countries taking in 2nd place; UK(1st) and US(2nd). In the other countries, music took a place 5th(Canada, France) and 7th(Germany). Nevertheless, thanks to the amount of video in the US and UK, music could become the second most frequent video genre. 3rd place, people & blog took a place. People & blog category took in 3rd place in most countries (except the US), the overall number of videos were slightly behind from music.

3.2.4 Trending videos in the different region based on Category

The dataset is now explored on trending videos based on category. The below section shows region wise video trending bar plot of top 10 trending video channels in each region based on number of views

3.2.4.1 *Top trending Channels in the US*

```
#US
us_channel_trend.df <- as.data.frame(us_videos.df$channel_title[!duplicated(us_videos.df$title)])
names(us_channel_trend.df) <- c("Channel")
us_channel_trend.df <- as.data.frame(sort(table(us_channel_trend.df$Channel), decreasing = TRUE))
us_channel_trend.df <- us_channel_trend.df[1:10,]
names(us_channel_trend.df) <- (c("channel", "count"))

ggplot(us_channel_trend.df, aes(x = channel, y = count,fill = factor(channel)), ) +
  geom_bar(stat = "identity") +theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  + theme(axis.text.x = element_text(angle = 70,hjust =1),legend.position = "none") +
  labs(title = "Plot of trending Channel in US", x = "Channels", y = "Number of Views")
```

Table 3.5 R code for trending channels in the US

The below bar chart shows the top 10 trending video channels in the US based on a number of views. The ESPN channel has the highest number of views in the given period as seen in the bar chart. Also, the Ellen show the next most trending video channel, which is followed the tonight show starring Jimmy Fallon in 3rd place, Jimmy Kimmel live in a 4th place, Netflix at 6th and NBA and CNN in 7th and 8th place. It is very evident that American follow sports the most and ESPN is widely viewed across the US as it hosts most of the American Football matches. Also, Jimmy Kimmel standup is predominantly followed by NBA and CNN.

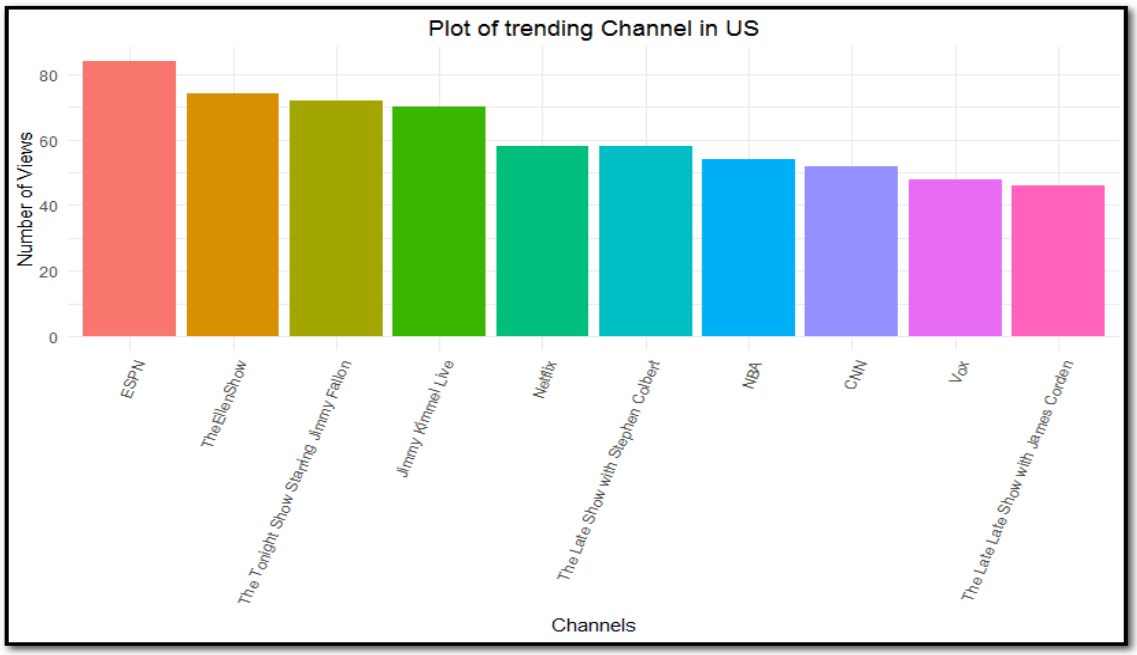


Figure 3.5 Top 10 trending channel in the US

3.2.4.2 Top trending Channels in Great Britain

```
#UK
gb_channel_trend.df <- as.data.frame(gb_videos.df$channel_title[!duplicated(gb_videos.df$title)])
names(gb_channel_trend.df) <- c("Channel")
gb_channel_trend.df <- as.data.frame(sort(table(gb_channel_trend.df$Channel), decreasing = TRUE))
gb_channel_trend.df <- gb_channel_trend.df[1:10,]
names(gb_channel_trend.df) <- (c("channel", "count"))

ggplot(gb_channel_trend.df, aes(x = channel, y = count,fill = factor(channel)), ) +
  geom_bar(stat = "identity") +theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending Channel in UK", x = "Channels", y = "Number of Views")
```

Table3.6 R code for trending channels in the UK

Interestingly the "The tonight show starring Jimmy Fallon" is the most trending video channel in the UK which was at third place in the US. Followed by The Elen Show and Jimmy Kimmel Live which has a very narrow gap in the number of views. The Saturday night show and the late-night show with James Corden is the next trending channels. Interestingly WWE stands at 7th place and TMZ sports at 10th place. We could see that

in the US, people are interested more towards sports in the given period and the people in the UK are interested in Live shows and news happenings rather sports.

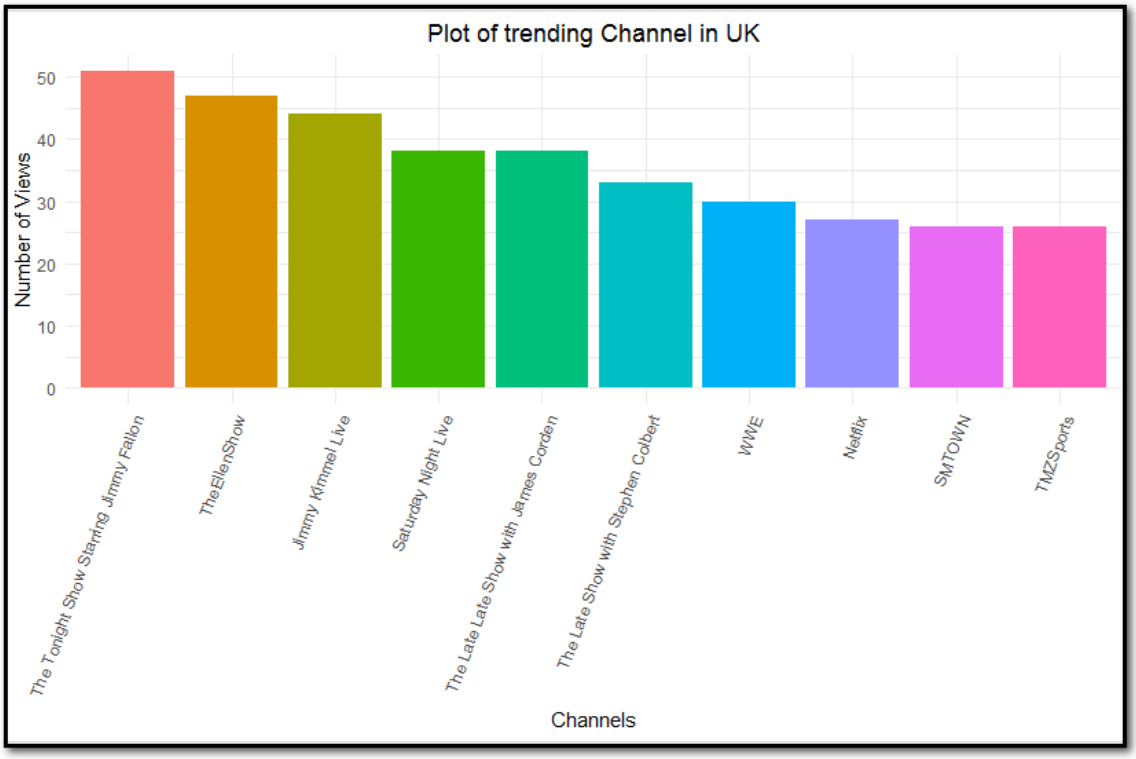


Figure 3.6 showing Top 10 trending channel in the UK

3.2.4.3 Top trending Channels in Canada

```
#Canada
ca_channel_trend.df <- as.data.frame(ca_videos.df$channel_title[!duplicated(ca_videos.df$title)])
names(ca_channel_trend.df) <- c("Channel")
ca_channel_trend.df <- as.data.frame(sort(table(ca_channel_trend.df$Channel), decreasing = TRUE))
ca_channel_trend.df <- ca_channel_trend.df[1:10,]
names(ca_channel_trend.df) <- (c("channel", "count"))

ggplot(ca_channel_trend.df, aes(x = channel, y = count,fill = factor(channel)), ) +
  geom_bar(stat = "identity") +theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending Channel in Canada", x = "Channels", y = "Number of Views")
```

Table3.7 R code for trending channels in Canada

The below-trend shows an interesting fact. The number of views is the highest among all the regions based on channels in Canada. And the top trending video channels are Indian channels in Canada. This might show that the number of Indians have moved into Canada recently and trying to catch up with their updates. The most trending video channel in Canada is a south Indian channel "Vikatan TV" famous for it magazines and news. The next trending channel is SET India an entertainment channel, followed by MS NBC which is the national broadcasting channel. There is a very narrow gap between the number of views across the top trending channels in Canada. The other prominent trending video channels are also Indian based which is SAB TV, Radaan Media. The CNN is in 8th place and ESPN is in 10th place.

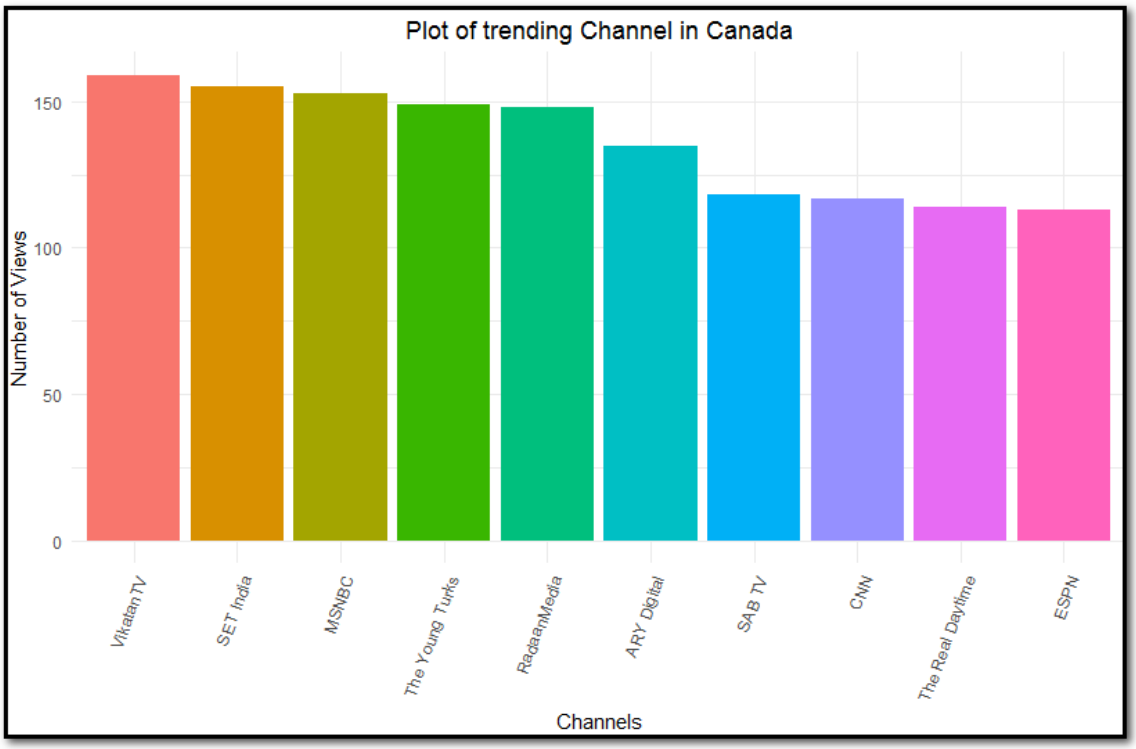


Figure 3.7 showing Top 10 trending channel in Canada

3.2.4.4 Top trending Channels in Germany

The below section shows the top 10 trending video channels in Germany. The number of hits for the top German based channel is very high which is 150 during the trending period. The below bar chart clearly shows the top 10 trending video channels in Germany

```r
#Germany
de_channel_trend.df <- as.data.frame(de_videos.df$channel_title[!duplicated(de_videos.df$title)])
names(de_channel_trend.df) <- c("Channel")
de_channel_trend.df <- as.data.frame(sort(table(de_channel_trend.df$Channel), decreasing = TRUE))
de_channel_trend.df <- de_channel_trend.df[1:10,]
names(de_channel_trend.df) <- (c("channel", "count"))

ggplot(de_channel_trend.df, aes(x = channel, y = count,fill = factor(channel)), ) +
  geom_bar(stat = "identity") + theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending Channel in Germany", x = "Channels", y = "Number of Views")
```

Table 3.8 R code for trending channels in Germany

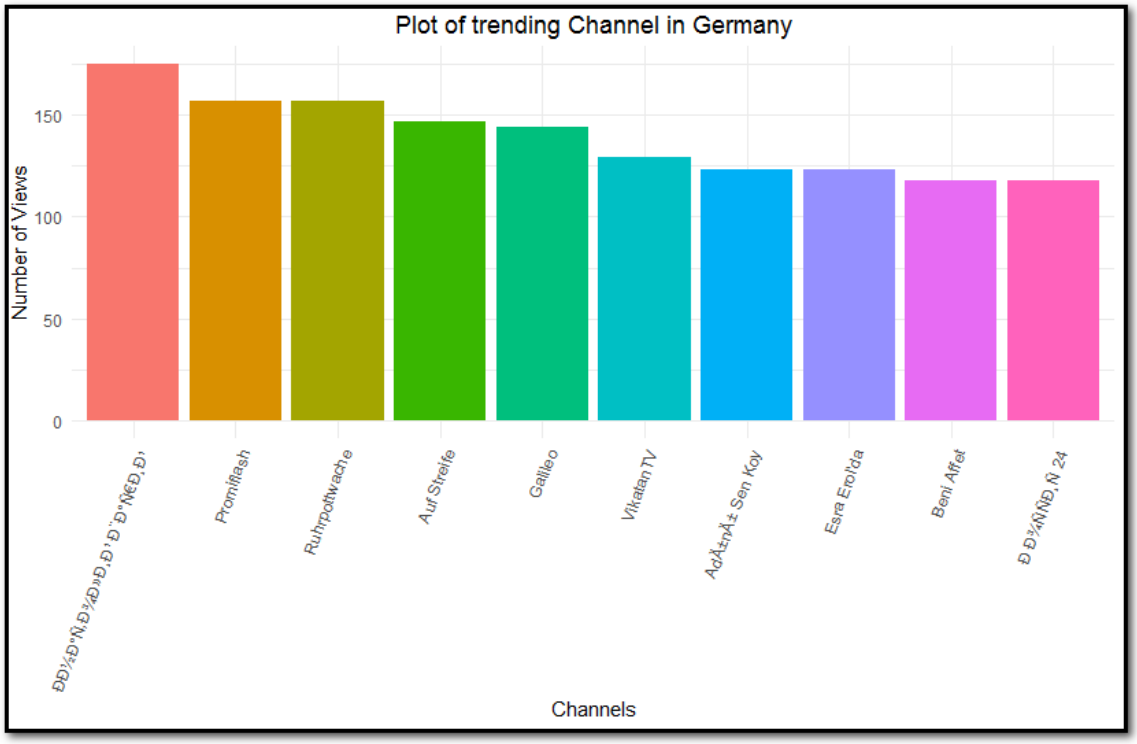


Figure 3.8 showing Top 10 trending channel in Germany

3.2.4.5 *Top trending Channels in France*

```r
#France
fr_channel_trend.df <- as.data.frame(fr_videos.df$channel_title[!duplicated(fr_videos.df$title)])
names(fr_channel_trend.df) <- c("Channel")
fr_channel_trend.df <- as.data.frame(sort(table(fr_channel_trend.df$Channel), decreasing = TRUE))
```

```r
fr_channel_trend.df <- fr_channel_trend.df[1:10,]
names(fr_channel_trend.df) <- (c("channel", "count"))

ggplot(fr_channel_trend.df, aes(x = channel, y = count,fill = factor(channel)), ) +
  geom_bar(stat = "identity") + theme_minimal() +  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending Channel in France", x = "Channels", y = "Number of Views")
```

Table 3.9 showing R code for trending channels in France

Again, the Vikatan TV is the top trending channel in France. Followed by Eihiwar Ettounsi, Radaan Media, Troom Troom FR and then Thiru TV. We could see that most of the channels trending are in Entertainment category which was evident from the previous analysis that most trending videos were from the entertainment category.

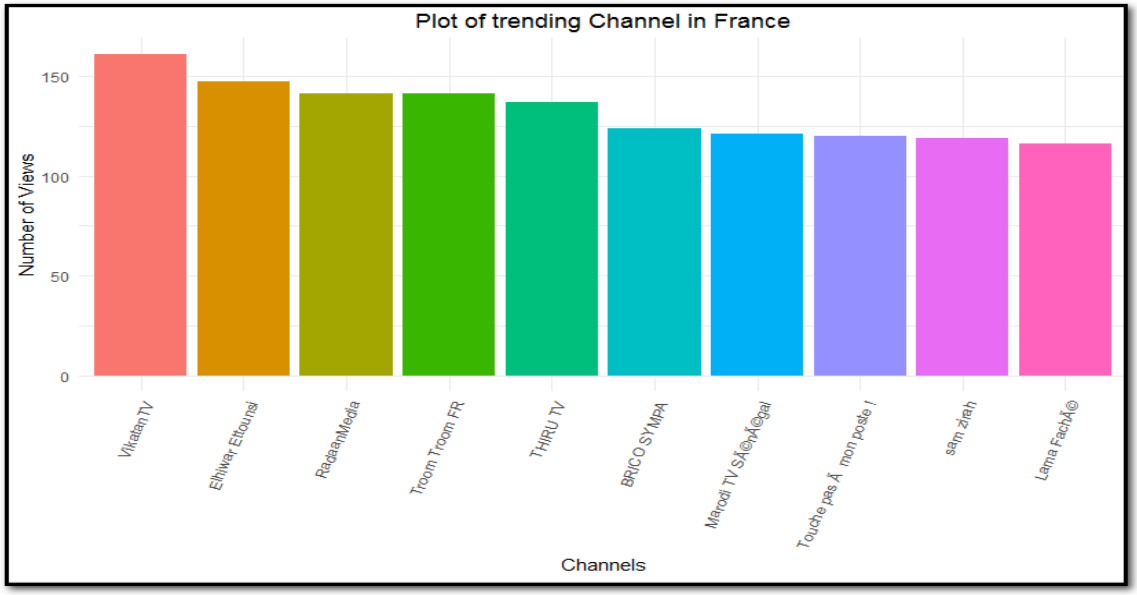


Figure 3.9 Top 10 trending channel in France

3.2.4.6 Top trending Channels in overall region

```r
#Overall
yt_channel_trend.df <- as.data.frame(Youtube_videos.df$channel_title[!duplicated(Youtube_videos.df$title)])
names(yt_channel_trend.df) <- c("Channel")
```

```r
yt_channel_trend.df <- as.data.frame(sort(table(yt_channel_trend.df$Channel), decreasing = TRUE))
yt_channel_trend.df <- yt_channel_trend.df[1:10,]
names(yt_channel_trend.df) <- (c("channel", "count"))

ggplot(yt_channel_trend.df, aes(x = channel, y = count,fill = factor(channel)), ) +
  geom_bar(stat = "identity") + theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of trending Channel in 5 countries", x = "Channels", y = "Number of Views")
```

Table 3.10 R code for trending channels in Overall Region.

We could see that the Eihiwar Ettoursi which is a French channel is the most trending channel in the selected time period across all the five regions and it is followed by the south Indian Channel Vikatan TV. Both these channels are almost having the same number of views in the selected period. They are followed by "The late show with Stephen Colbert" and MS NBC a news channel. Thus, it is very evident that the most trending channels are Entertainment category channels on YouTube in the selected period.

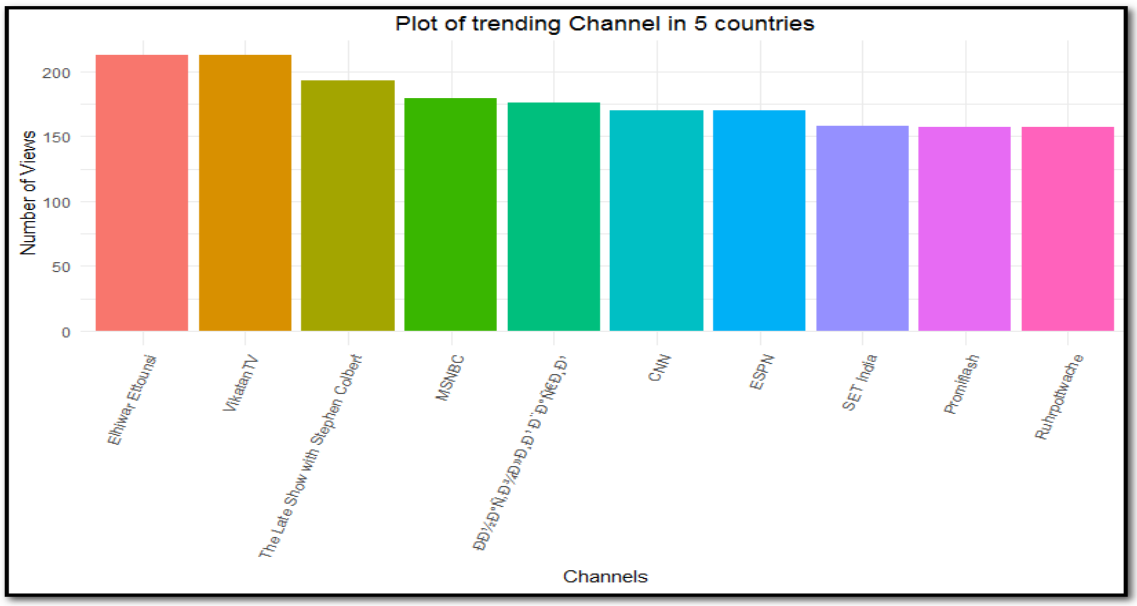


Figure 3.10 Top 10 trending channel in Overall Region

3.2.5 Analyzing the dataset of all countries with respect to Views, likes, dislikes, and comment counts.

In this section, we have done a detailed visualization on the dataset based on the various numerical fields like views, likes, dislikes and comments counts. The plots self-explain the analysis in these sections. In all the plots the numerical values are plotted against the video title.

3.2.5.1 Bivariate Comparison

Before we explore the numerical values of views, likes, dislikes, comment counts separately, we will analyze the variables between each other. Like Views Vs Likes, Views Vs comment counts Views vs Dislikes.

The below section shows the scatter plot between these variables and clearly explains us the relation between them. In all the cases, they are linear and mostly colinear as we overserved already in the heat map.

```
# Views Vs Likes
options(screen = 999)

ggplot(Youtube_videos.df[,.("views"=max(views),"likes"=max(likes)),by=title],
aes(views,likes,colour=likes,size=likes))+
  geom_jitter()+geom_smooth(method = 'lm')+guides(fill="none")+labs(title="Vi
ews Vs Likes",subtitle="In days")+
  theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "none")

#Views Vs Comment Counts
ggplot(Youtube_videos.df[,.("views"=max(views),"comments"=max(comment_count))
,by=title],aes(views,comments,colour=comments,size=comments))+
  geom_jitter()+geom_smooth(method = 'lm')+guides(fill="none")+labs(title="Vi
ews Vs Comments",subtitle="In days")+
  theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "none")

#Views Vs Comment Counts
ggplot(Youtube_videos.df[,.("views"=max(views),"dislikes"=max(dislikes)),by=t
itle],aes(views,dislikes,colour=dislikes,size=dislikes))+
  geom_jitter()+geom_smooth(method = 'lm')+guides(fill="none")+labs(title="Vi
ews Vs dislikes",subtitle="In days")+
  theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "none")
```

Table 3.11 R code for Scatter plot comparing Views ~ Likes and Views ~ comments and Views ~ dislikes

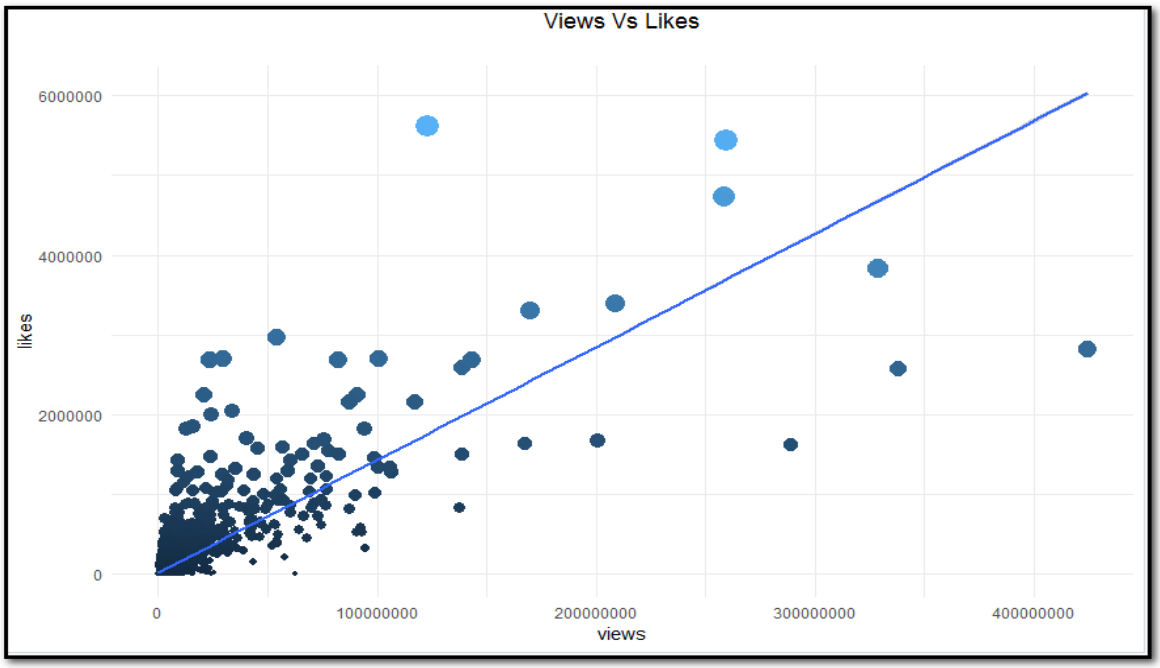


Figure 3.11 scatter plot between Views ~ Likes

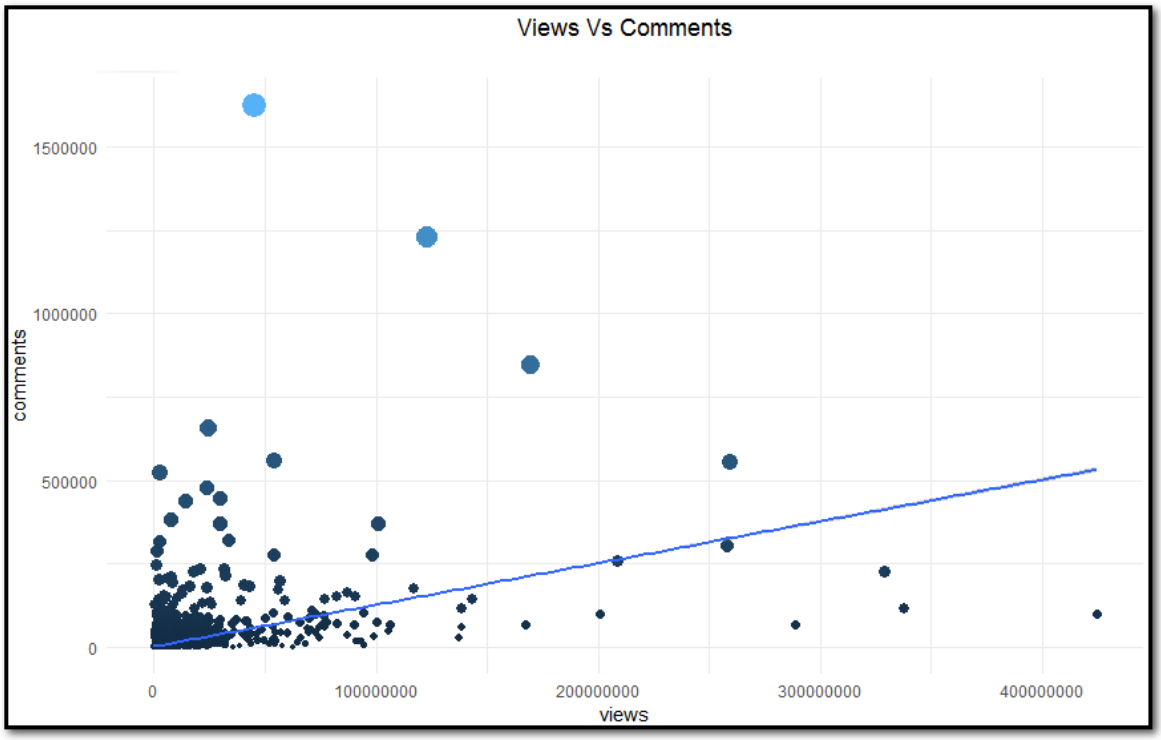


Figure 3.12 scatter plot between Views ~ Likes

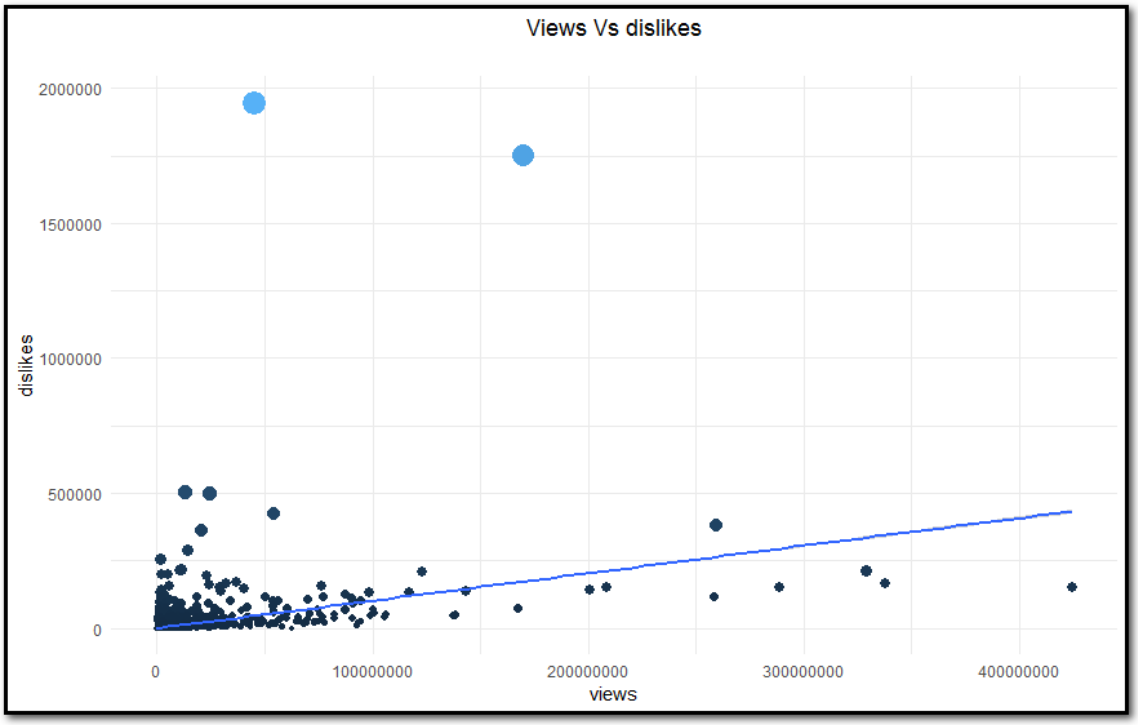


Figure 3.13 shows a scatter plot between Views ~ dislikes

From the above figures, it is again evident that there are a greater number of likes for the video which are viewed and getting trended across all the five regions. This result confirms our correlation plots and heat map.

3.2.5.2 *Analysing the dataset of all countries with respect to Views*

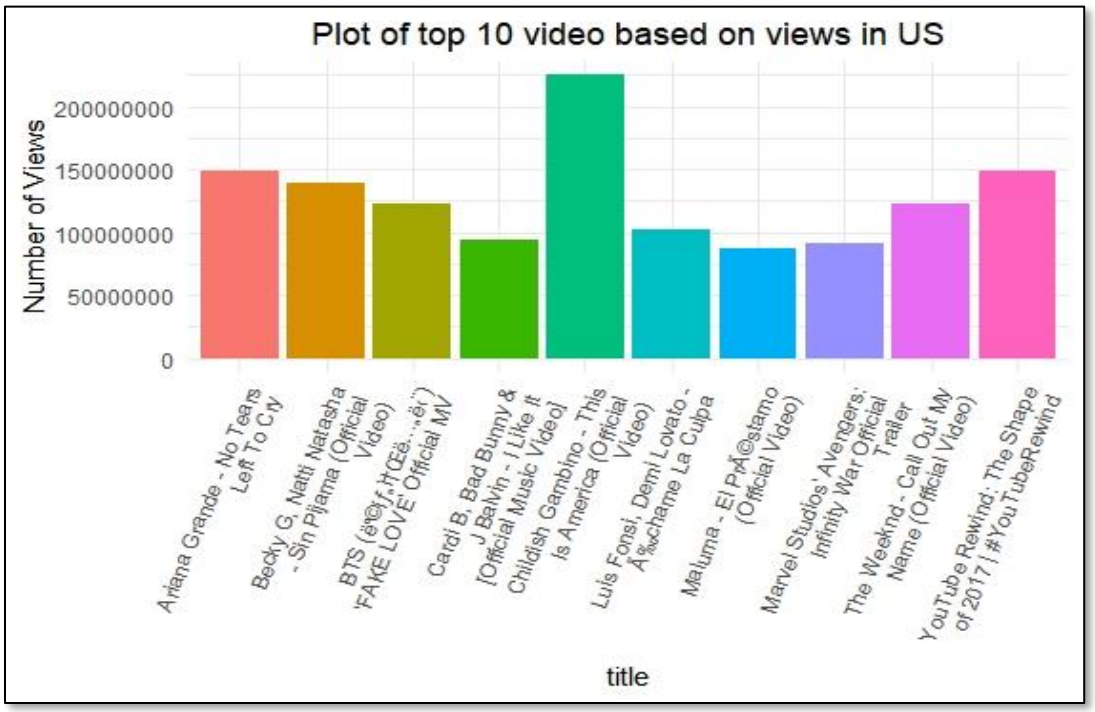
In this section, we have categorized the top 10 videos based on Views for each region and the overall dataset.

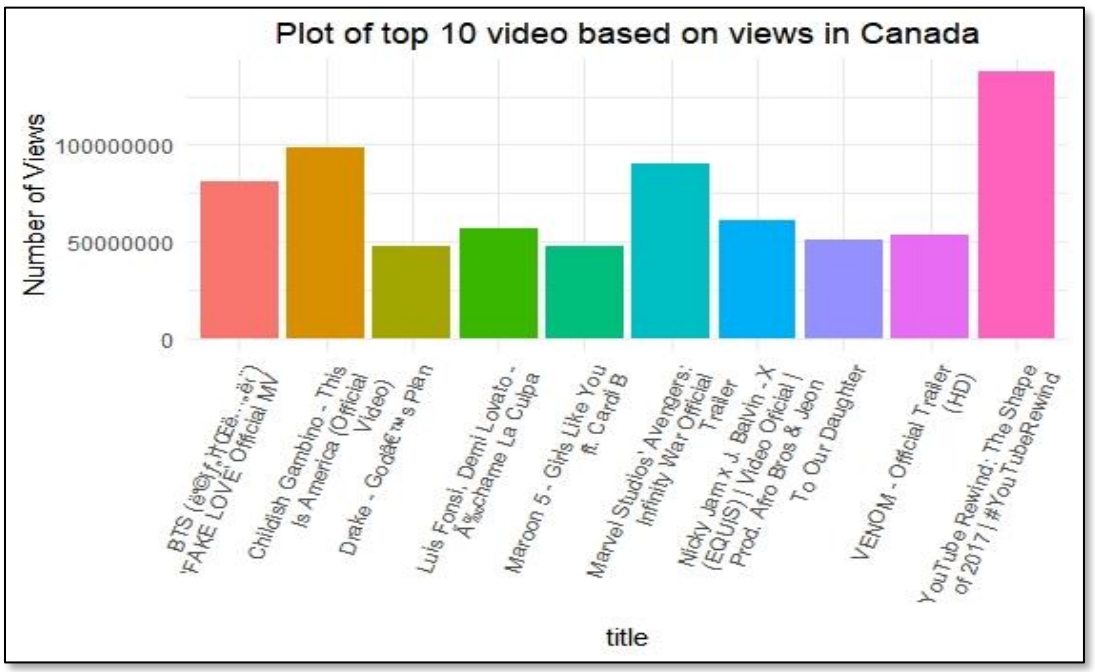
```
#1) US
us_video_view <- as.data.table(rbind(us_videos.df))
us_video_view <- us_video_view[,.("views"=round(max(views,na.rm = T), digits = 2)), by=.(title)][order(-views)]
us_video_view <- (us_video_view[1:10,])
ggplot(us_video_view, aes(x = stringr::str_wrap(title,25), y = views,fill = factor(title)), ) +
  geom_bar(stat = "identity") +theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on views in US", x = "title", y = "Number of Views")

#2) Canada
ca_video_view <- as.data.table(rbind(ca_videos.df))
```

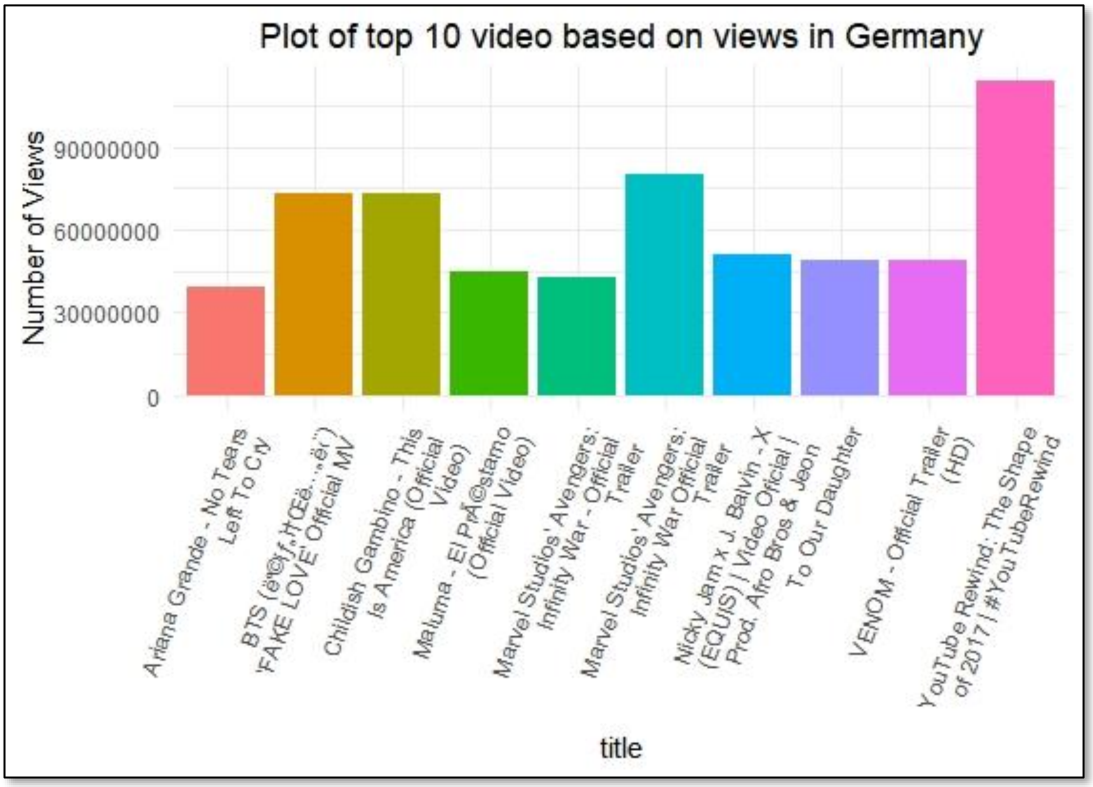
```
ca_video_view <- ca_video_view[,.("views"=round(max(views,na.rm = T), digits = 2)), by=.(title)][order(-views)]
ca_video_view <- (ca_video_view[1:10,])
ggplot(ca_video_view, aes(x = stringr::str_wrap(title, 25), y = views,fill = factor(title)), ) +
  geom_bar(stat = "identity") +theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on views in Canada", x = "title", y = "Number of Views")

#3) Germany
de_video_view <- as.data.table(rbind(de_videos.df))
de_video_view <- de_video_view[,.("views"=round(max(views,na.rm = T), digits = 2)), by=.(title)][order(-views)]
de_video_view <- (de_video_view[1:10,])
ggplot(de_video_view, aes(x = stringr::str_wrap(title, 25), y = views,fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on views in Germany", x = "title", y = "Number of Views")

#4) France
fr_video_view <- as.data.table(rbind(fr_videos.df))
fr_video_view <- fr_video_view[,.("views"=round(max(views,na.rm = T), digits = 2)), by=.(title)][order(-views)]
fr_video_view <- (fr_video_view[1:10,])
ggplot(fr_video_view, aes(x = stringr::str_wrap(title, 25), y = views,fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on views in France", x = "title", y = "Number of Views")

#5) UK
gb_video_view <- as.data.table(rbind(gb_videos.df))
gb_video_view <- gb_video_view[,.("views"=round(max(views,na.rm = T), digits = 2)), by=.(title)][order(-views)]
gb_video_view <- (gb_video_view[1:10,])
ggplot(gb_video_view, aes(stringr::str_wrap(title, 25), y = views,fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on views in UK", x = "title", y = "Number of Views")

#6) Overall
yt_video_view <- as.data.table(rbind(Youtube_videos.df))
yt_video_view <- yt_video_view[,.("views"=round(max(views,na.rm = T), digits = 2)), by=.(title)][order(-views)]
yt_video_view <- (yt_video_view[1:10,])
head(yt_video_view)
ggplot(yt_video_view, aes(x = stringr::str_wrap(title, 25), y = views,fill = factor(title)), ) +
```

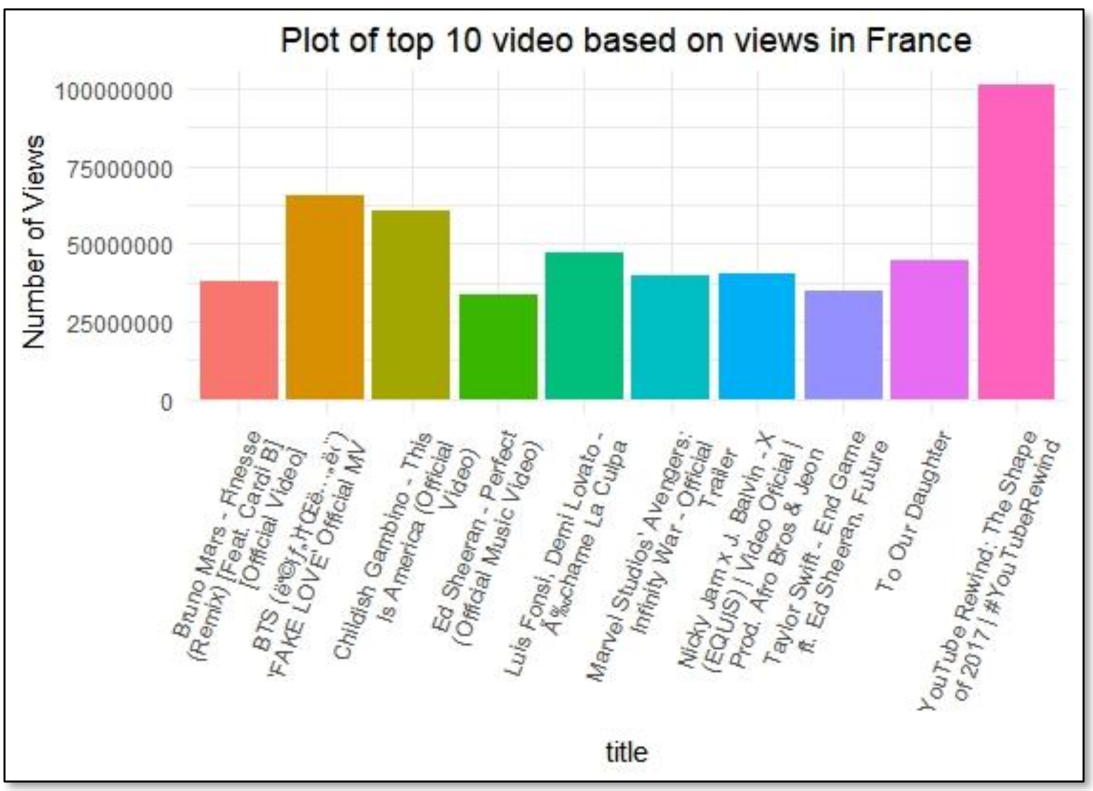
```
49.   geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
    t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
    ,legend.position = "none") +
50.   labs(title = "Plot of top 10 video based on views in 5 countries", x = "tit
    le", y = "Number of Views")
```

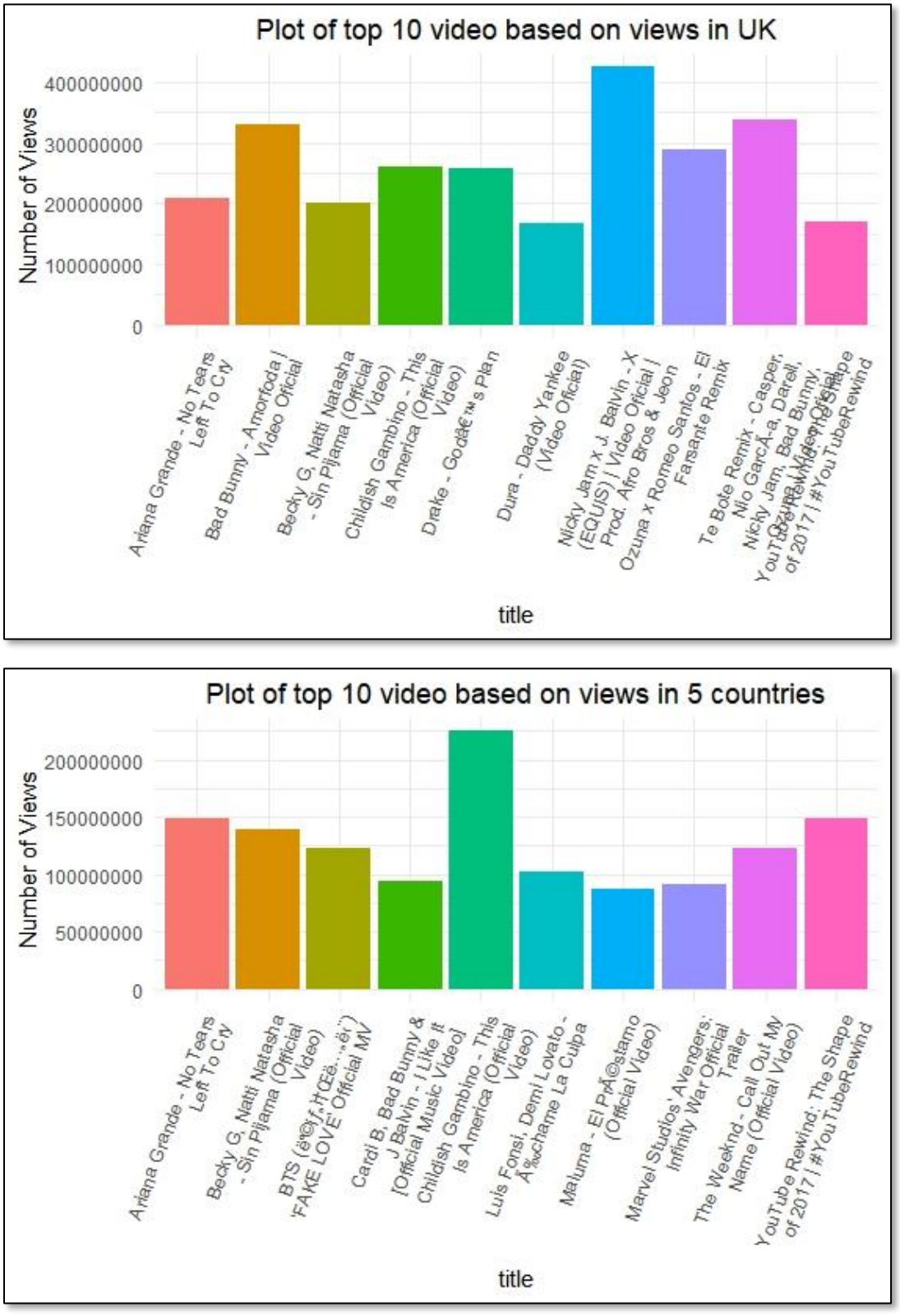
Table 3.12 R code for most views of a video for a different region.






Plot of top 10 video based on views in Germany
Number of Views
90000000
60000000
30000000
0
Ariana Grande - No Tears Left To Cry
Childish Gambino - This Is America (Official Video)
Marvel Studios' Avengers: Infinity War - Official Trailer
Marvel Studios' Avengers: Infinity War Official Trailer
Nicky Jam x J. Balvin - X (EQUIS) | Video Oficial | Prod. Afro Bros & Jeon
To Our Daughter
VENOM - Official Trailer (HD)
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
title


Plot of top 10 video based on views in France
Number of Views
100000000
75000000
50000000
25000000
0
Bruno Mars - Finesse (Remix) [Feat. Cardi B] [Official Video]
Childish Gambino - This Is America (Official Video)
Ed Sheeran - Perfect (Official Music Video)
Marvel Studios' Avengers: Infinity War - Official Trailer
Nicky Jam x J. Balvin - X (EQUIS) | Video Oficial | Prod. Afro Bros & Jeon
Taylor Swift - End Game ft. Ed Sheeran, Future
To Our Daughter
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
title



Figures 3.14 Top 10 video based on views in different regions and Overall Region

As like a bar chart, In May 2018, Childish Gambino released “This is America” music video, which debuted at number-one on the Hot 1001 in the US. Over the 200Million views recorded only in the US and took in the 2nd place in the other countries except for UK(4th place). As the result, maximum views in the 5 countries of this video recorded over the 225 Million views. 2nd place for “Youtube Rewind 2017”. 3rd for Ariana Grande “No Tear Left to Cry”. Overall, music video took 8 of 10 places in Top 10 of view videos. This trend shows that many Youtube users have using Youtube as a music player.

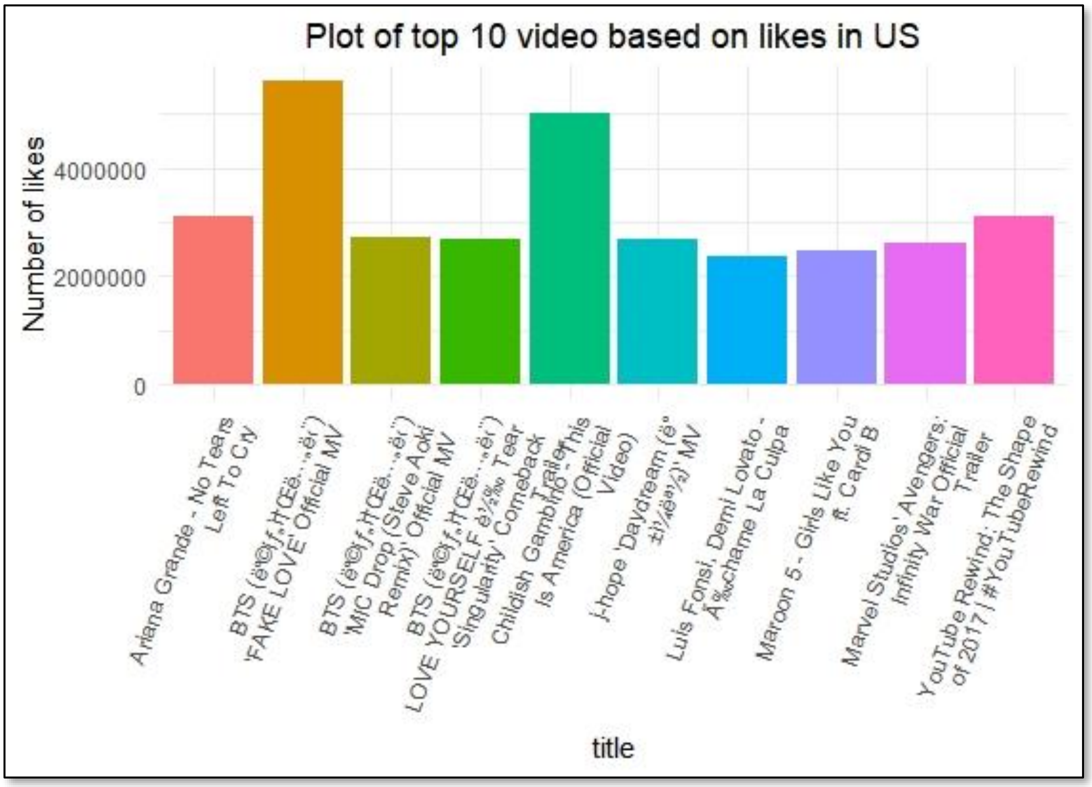
3.2.5.3 Analysing the dataset of all countries with respect to Likes

```
#Likes
#1.US
us_video_likes <- as.data.table(rbind(us_videos.df))
us_video_likes <- us_video_likes[,.("likes"=round(max(likes,na.rm = T), digits = 2)), by=.(title)][order(-likes)]
us_video_likes <- (us_video_likes[1:10,])

ggplot(us_video_likes, aes(x = stringr::str_wrap(title, 25), y = likes, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on likes in US", x = "title", y = "Number of likes")

#2) Canada
ca_video_likes <- as.data.table(rbind(ca_videos.df))
ca_video_likes <- ca_video_likes[,.("likes"=round(max(likes,na.rm = T), digits = 2)), by=.(title)][order(-likes)]
ca_video_likes <- (ca_video_likes[1:10,])

ggplot(ca_video_likes, aes(x = stringr::str_wrap(title, 25), y = likes, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +  theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on likes in Canada", x = "title", y = "Number of likes")

#3) Germany
de_video_likes <- as.data.table(rbind(de_videos.df))
de_video_likes <- de_video_likes[,.("likes"=round(max(likes,na.rm = T), digits = 2)), by=.(title)][order(-likes)]
de_video_likes <- (de_video_likes[1:10,])

ggplot(de_video_likes, aes(x = stringr::str_wrap(title, 25), y = likes, fill = factor(title)), ) +
```

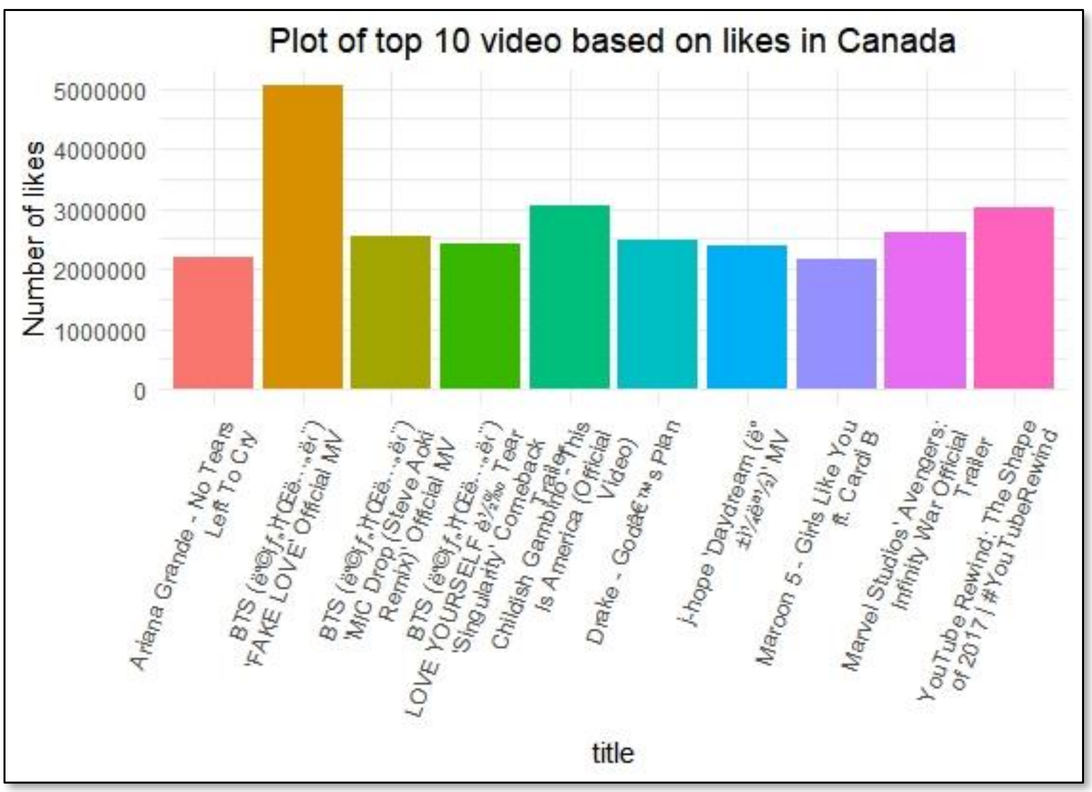
[1] https://www.billboard.com/articles/columns/chart-beat/8455823/childish-gambino-this-is-america-no-1-hot-100

```
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on likes in Germany", x = "title",
 y = "Number of likes")

#4) France
fr_video_likes <- as.data.table(rbind(fr_videos.df))
fr_video_likes <- fr_video_likes[,.("likes"=round(max(likes,na.rm = T), digit
s = 2)), by=.(title)][order(-likes)]
fr_video_likes <- (fr_video_likes[1:10,])

ggplot(fr_video_likes, aes(x = stringr::str_wrap(title, 25), y = likes, fill
= factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on likes in France", x = "title",
y = "Number of likes")

#5) UK
gb_video_likes <- as.data.table(rbind(gb_videos.df))
gb_video_likes <- gb_video_likes[,.("likes"=round(max(likes,na.rm = T), digit
s = 2)), by=.(title)][order(-likes)]
gb_video_likes <- (gb_video_likes[1:10,])

ggplot(gb_video_likes, aes(x = stringr::str_wrap(title, 25), y = likes, fill
= factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on likes in UK", x = "title", y =
"Number of likes")

#6) Overall
yt_video_likes <- as.data.table(rbind(Youtube_videos.df))
yt_video_likes <- yt_video_likes[,.("likes"=round(max(likes,na.rm = T), digit
s = 2)), by=.(title)][order(-likes)]
yt_video_likes <- (yt_video_likes[1:10,])
head(yt_video_likes)

ggplot(yt_video_likes, aes(x = stringr::str_wrap(title, 25), y = likes, fill
= factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on likes in 5 countries", x = "tit
le", y = "Number of likes")
```

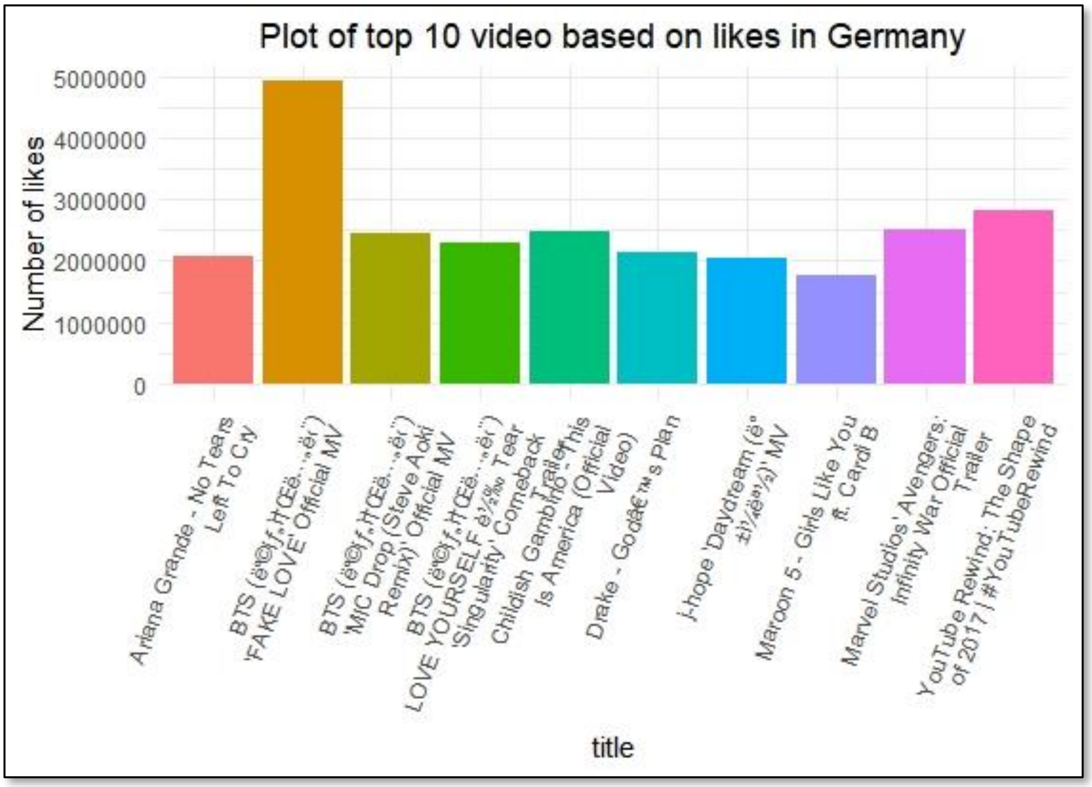
Table 3.14 R code for most Likes of a video for a different region.

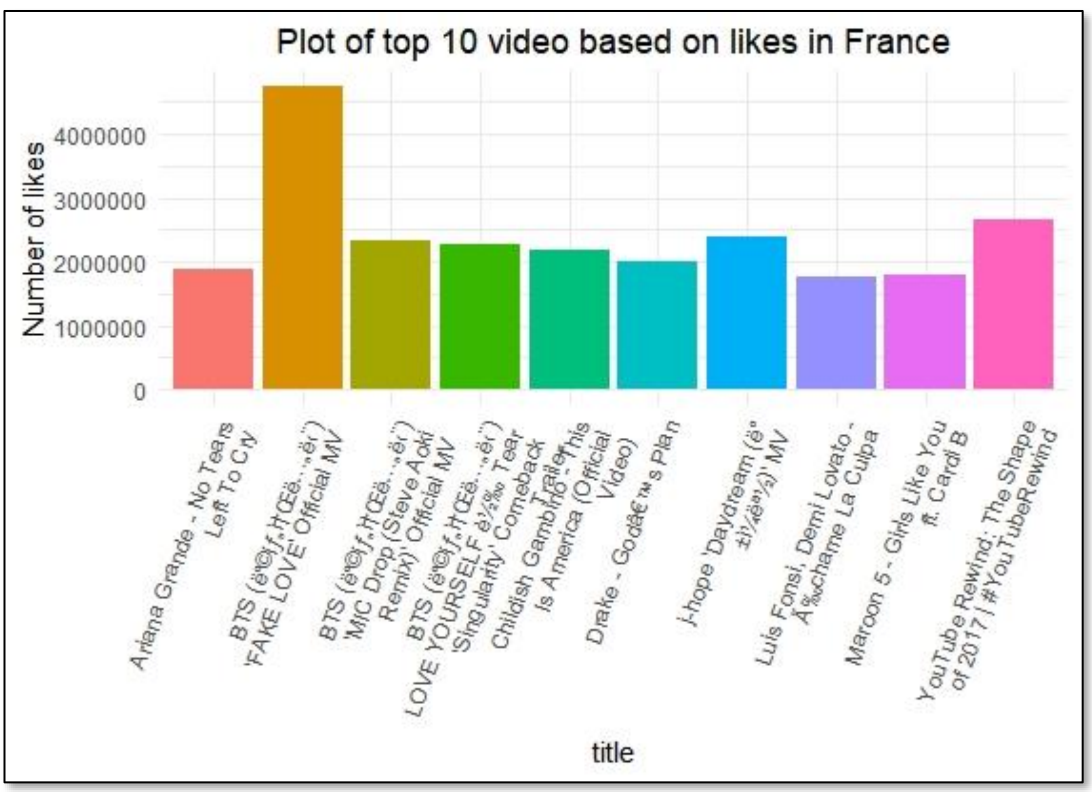


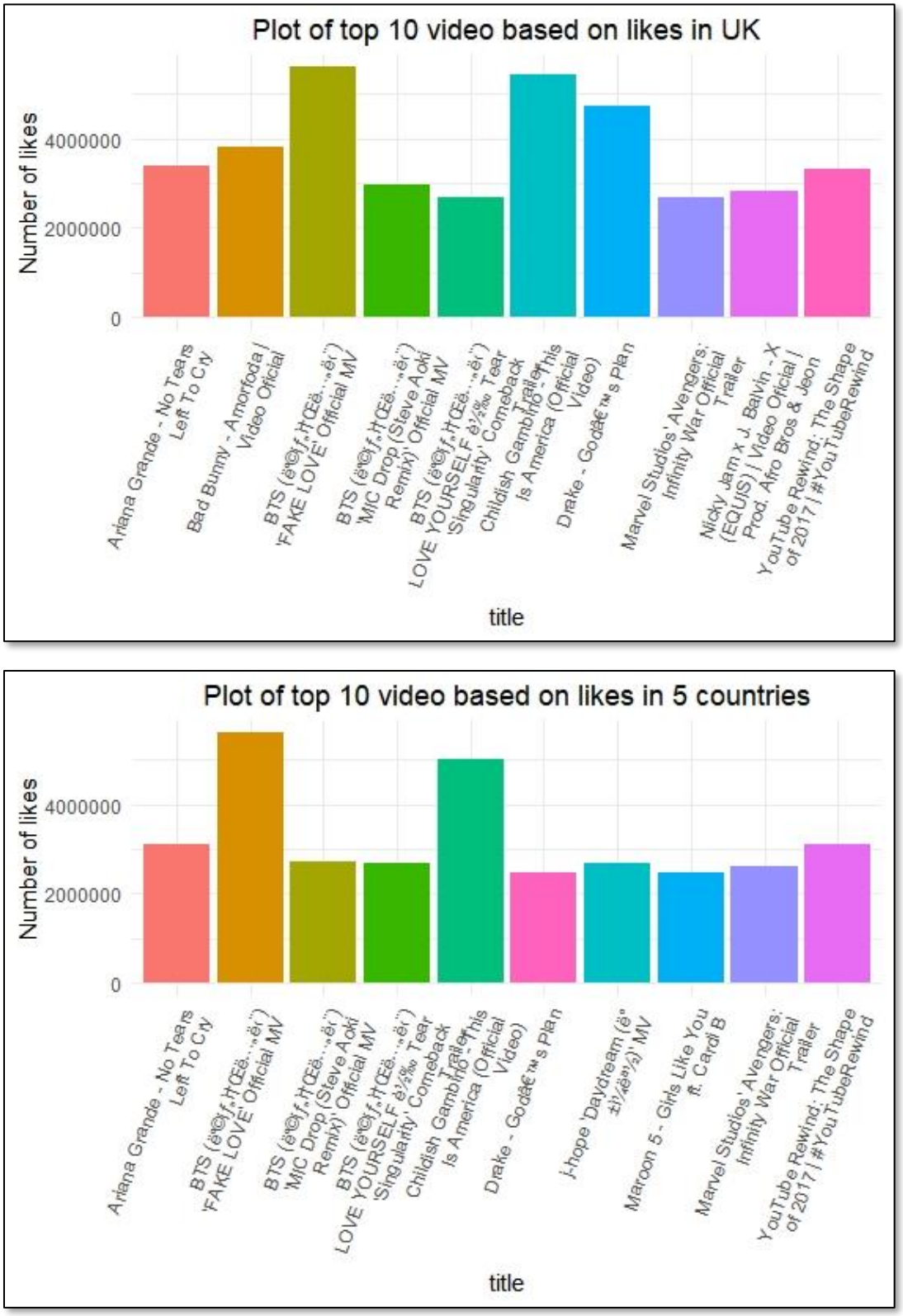
Plot of top 10 video based on likes in US
Number of likes
title



Plot of top 10 video based on likes in Canada
Number of likes
title


Plot of top 10 video based on likes in Germany
Number of likes
5000000
4000000
3000000
2000000
1000000
0
Ariana Grande - No Tears Left To Cry
BTS 'FAKE LOVE' Official MV
BTS 'MIC Drop (Steve Aoki Remix)' Official MV
BTS LOVE YOURSELF 'Singularity' Comeback Trailer
Childish Gambino - This Is America (Official Video)
Drake - God's Plan
j-hope 'Daydream' MV
Maroon 5 - Girls Like You ft. Cardi B
Marvel Studios' Avengers: Infinity War Official Trailer
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
title


Plot of top 10 video based on likes in France
Number of likes
4000000
3000000
2000000
1000000
0
Ariana Grande - No Tears Left To Cry
BTS 'FAKE LOVE' Official MV
BTS 'MIC Drop (Steve Aoki Remix)' Official MV
BTS LOVE YOURSELF 'Singularity' Comeback Trailer
Childish Gambino - This Is America (Official Video)
Drake - God's Plan
j-hope 'Daydream' MV
Luis Fonsi, Demi Lovato - La Culpa
Maroon 5 - Girls Like You ft. Cardi B
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
title



Figures 3.15 Top 10 video based on Likes in different regions and Overall Region

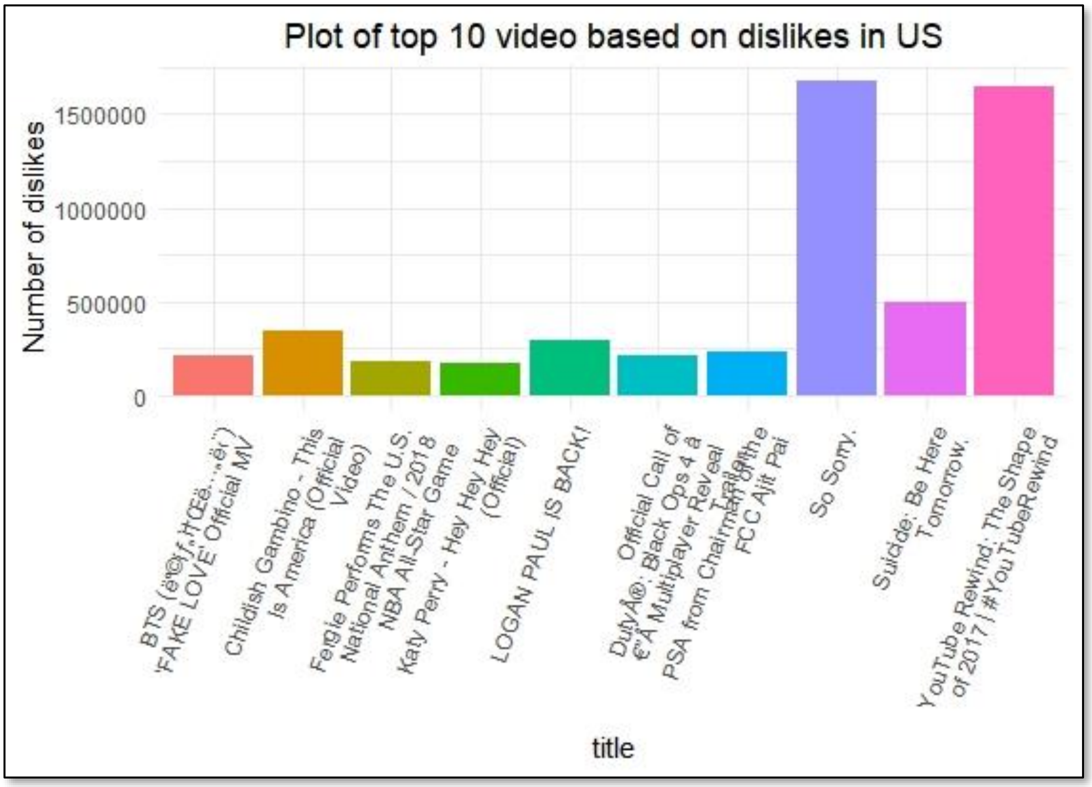
According to the bar chart, 8 of the top 1o likes video were also in the music category. The tendency of likes is following order views. The interesting point is the same singer take 1st and 5th, 6th place; BTS. They released a new song in May 2018 and made a syndrome of BTS both North America and Europe. This trend shows that this young South Korean boy group getting a positive rating at least from Youtube users.

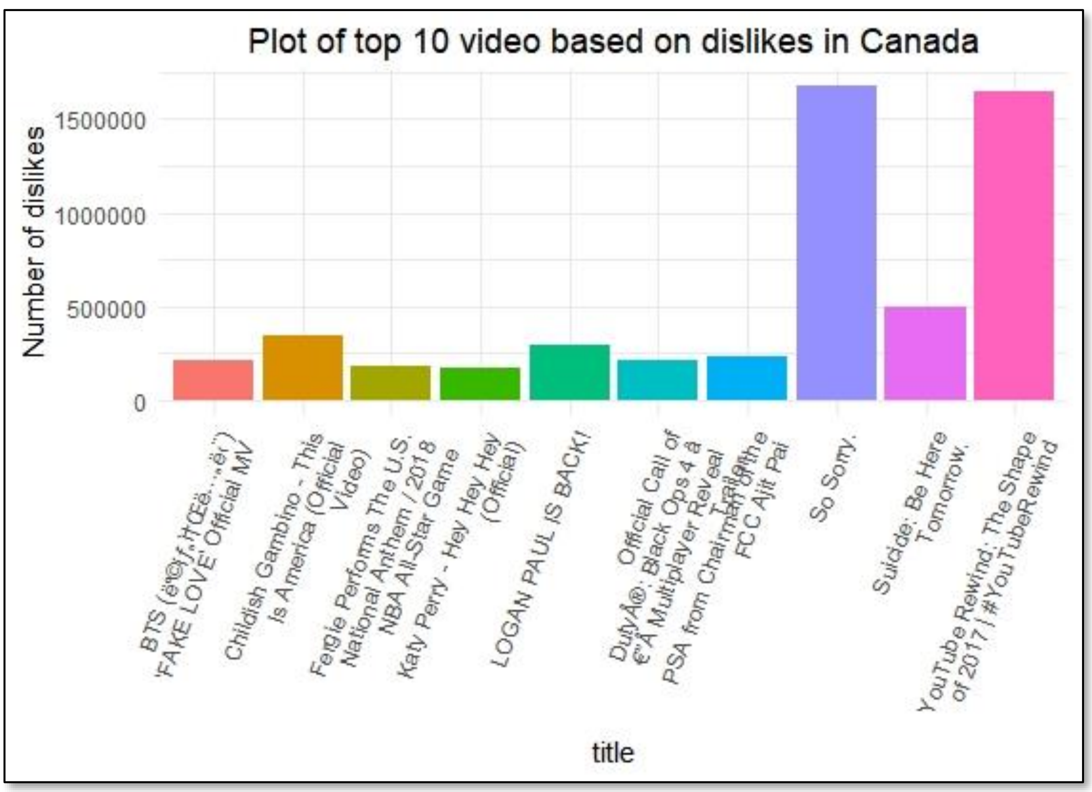
3.2.5.4 Analyzing the dataset of all countries with respect to dislikes

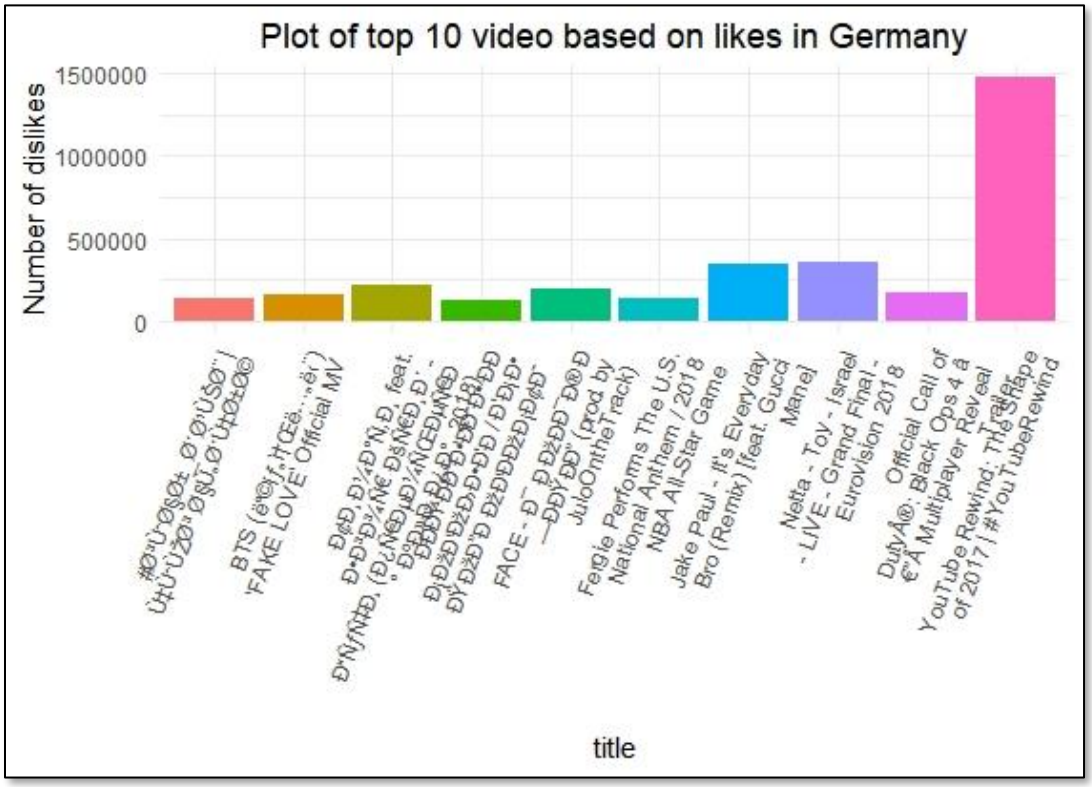
```
#3.Dislikes
#1) the US
us_video_dislikes <- aggregate(dislikes ~ title, data = us_videos.df, sum)

us_video_dislikes <- us_video_dislikes[order(us_video_dislikes$dislikes, decr
easing = TRUE),]

us_video_dislikes <- us_video_dislikes[1:10,]

ggplot(us_video_dislikes, aes(x = stringr::str_wrap(title, 25), y = dislikes,
 fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on dislikes in US", x = "title", y
 = "Number of dislikes")

#2) Canada
ca_video_dislikes <- aggregate(dislikes ~ title, data = ca_videos.df, sum)

ca_video_dislikes <- ca_video_dislikes[order(ca_video_dislikes$dislikes, decr
easing = TRUE),]

ca_video_dislikes <- ca_video_dislikes[1:10,]

ggplot(ca_video_dislikes, aes(x = stringr::str_wrap(title, 25), y = dislikes,
 fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on dislikes in Canada", x = "title
", y = "Number of dislikes")


#3) Germany
de_video_dislikes <- aggregate(dislikes ~ title, data = de_videos.df, sum)

de_video_dislikes <- de_video_dislikes[order(de_video_dislikes$dislikes, decr
easing = TRUE),]

de_video_dislikes <- de_video_dislikes[1:10,]

ggplot(de_video_dislikes, aes(x = stringr::str_wrap(title, 25), y = dislikes,
 fill = factor(title)), ) +
```

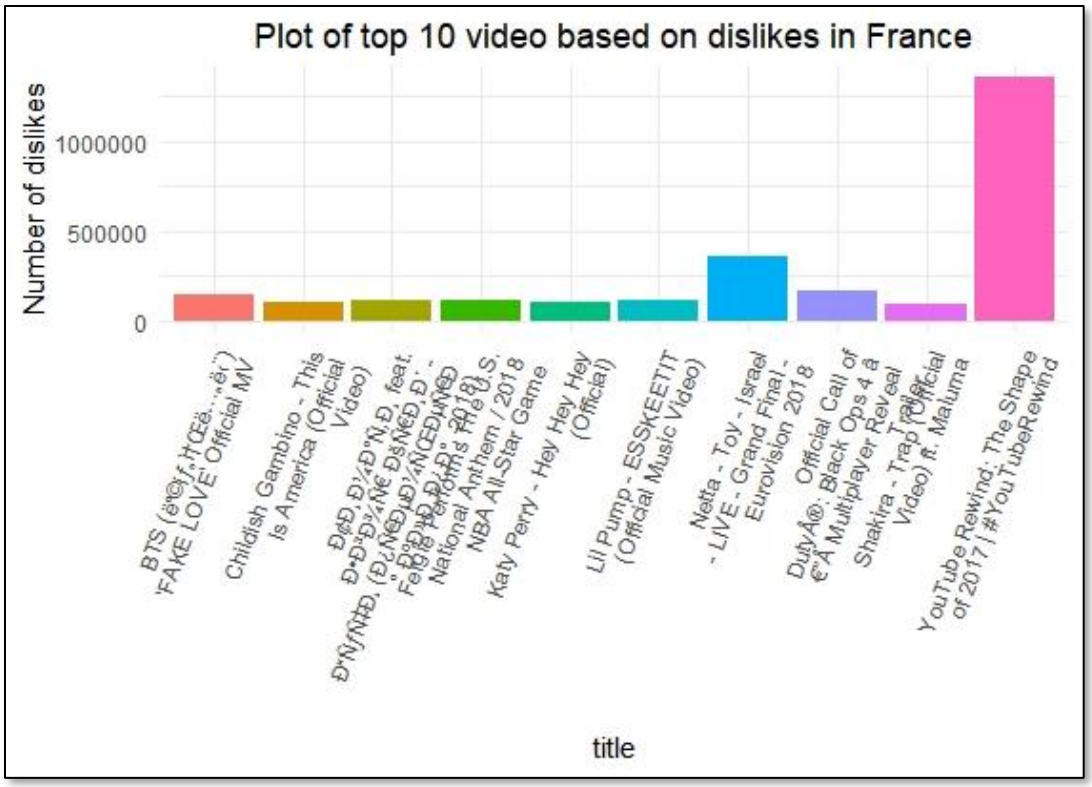
```
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 30,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on likes in Germany", x = "title",
 y = "Number of dislikes")

#4) France
fr_video_dislikes <- aggregate(dislikes ~ title, data = fr_videos.df, sum)

fr_video_dislikes <- fr_video_dislikes[order(fr_video_dislikes$dislikes, decr
easing = TRUE),]

fr_video_dislikes <- fr_video_dislikes[1:10,]

ggplot(fr_video_dislikes, aes(x = stringr::str_wrap(title, 25), y = dislikes,
 fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on dislikes in France", x = "title
", y = "Number of dislikes")

#5) UK
gb_video_dislikes <- aggregate(dislikes ~ title, data = gb_videos.df, sum)

gb_video_dislikes <- gb_video_dislikes[order(gb_video_dislikes$dislikes, decr
easing = TRUE),]

gb_video_dislikes <- gb_video_dislikes[1:10,]

ggplot(gb_video_dislikes, aes(x = stringr::str_wrap(title, 25), y = dislikes,
 fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on dislikes in UK", x = "title", y
 = "Number of dislikes")

#6) Overall
yt_video_dislikes <- aggregate(dislikes ~ title, data = Youtube_videos.df, su
m)

yt_video_dislikes <- yt_video_dislikes[order(yt_video_dislikes$dislikes, decr
easing = TRUE),]

yt_video_dislikes <- yt_video_dislikes[1:10,]

ggplot(yt_video_dislikes, aes(x = stringr::str_wrap(title, 25), y = dislikes,
 fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = elemen
t_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1)
,legend.position = "none") +
  labs(title = "Plot of top 10 video based on dislikes in 5 countries", x = "
title", y = "Number of dislikes")
```

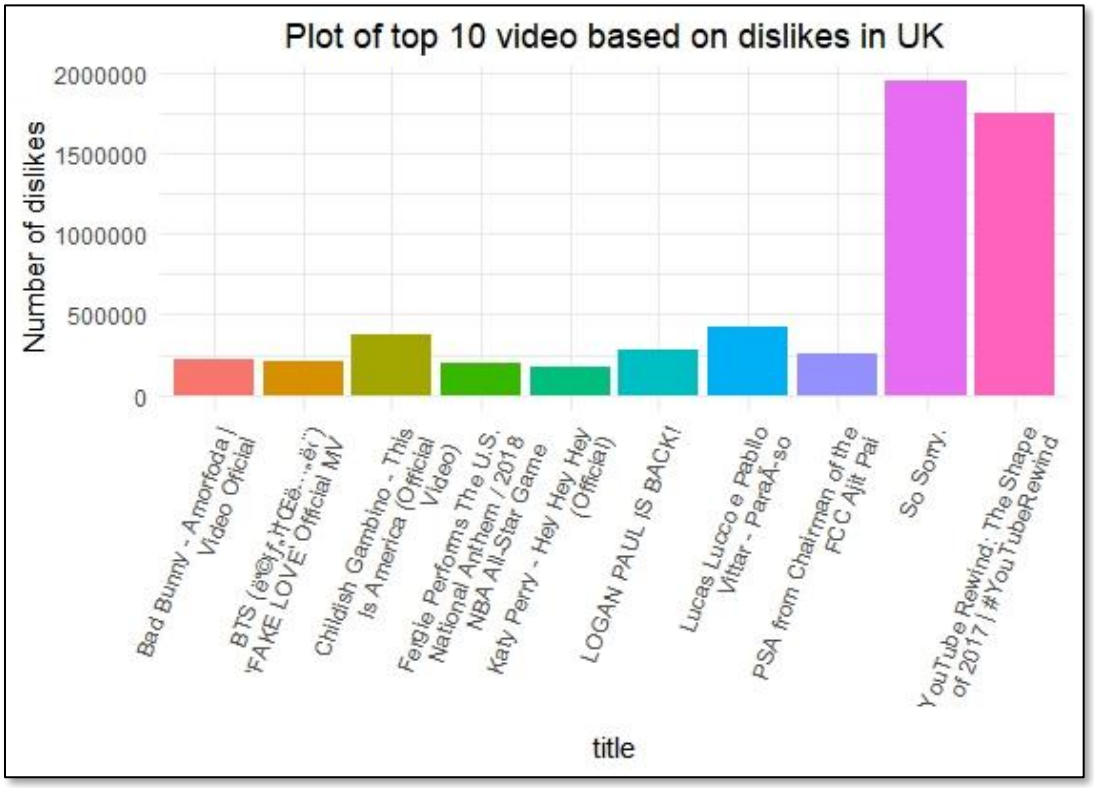
Table 3.15 showing R code for most dislikes of a video for a different region.

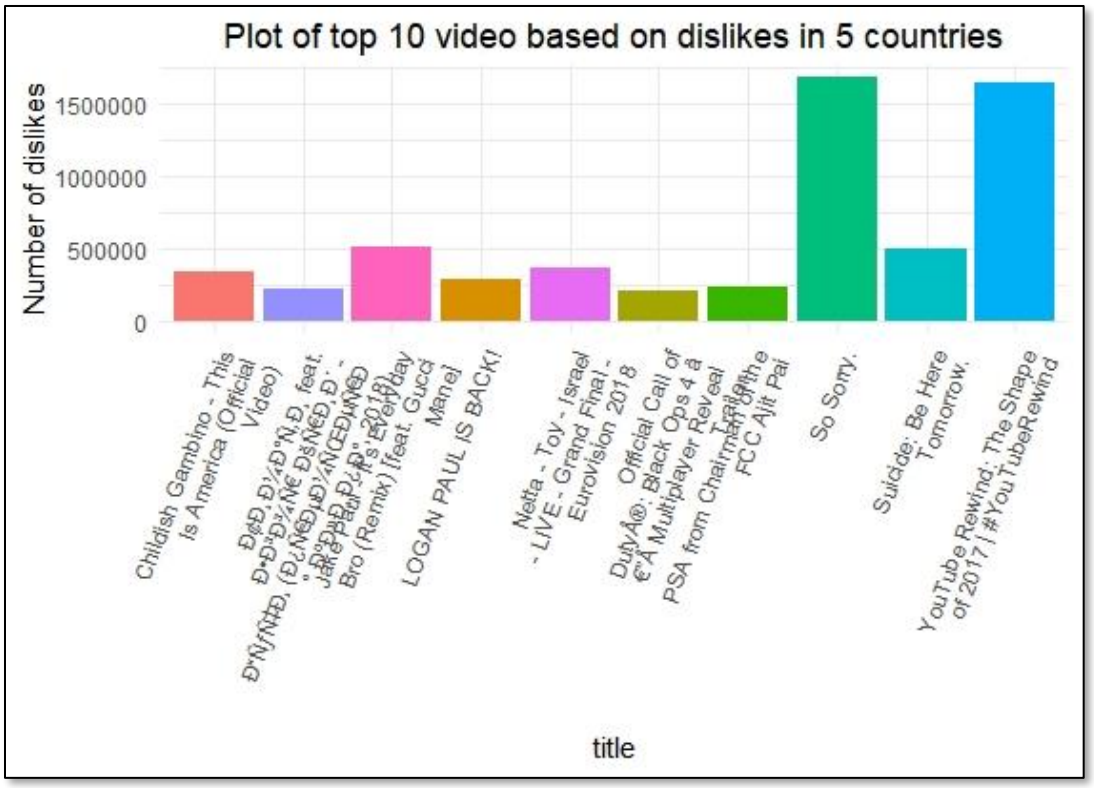

Plot of top 10 video based on dislikes in US
Number of dislikes
1500000
1000000
500000
0
LOGAN PAUL IS BACK!
So Sorry.
Suicide: Be Here Tomorrow.
title


Plot of top 10 video based on dislikes in Canada
Number of dislikes
1500000
1000000
500000
0
LOGAN PAUL IS BACK!
So Sorry.
Suicide: Be Here Tomorrow.
title


Plot of top 10 video based on likes in Germany
Number of dislikes
1500000
1000000
500000
0
title


Plot of top 10 video based on dislikes in France
Number of dislikes
1000000
500000
0
BTS (ë°©íƒ„ì†Œë…„ë‹¨) 'FAKE LOVE' Official MV
Childish Gambino - This Is America (Official Video)
Netta - Toy - Israel - LIVE - Grand Final - Eurovision 2018
Official Call of Duty®: Black Ops 4 – Multiplayer Reveal Trailer
Shaktira - Trap (Official Video) ft. Maluma
YouTube Rewind: The Shape of 2017 | #YouTubeRewind
title





Figures 3.16 Top 10 video based on dislikes in different regions and Overall Region

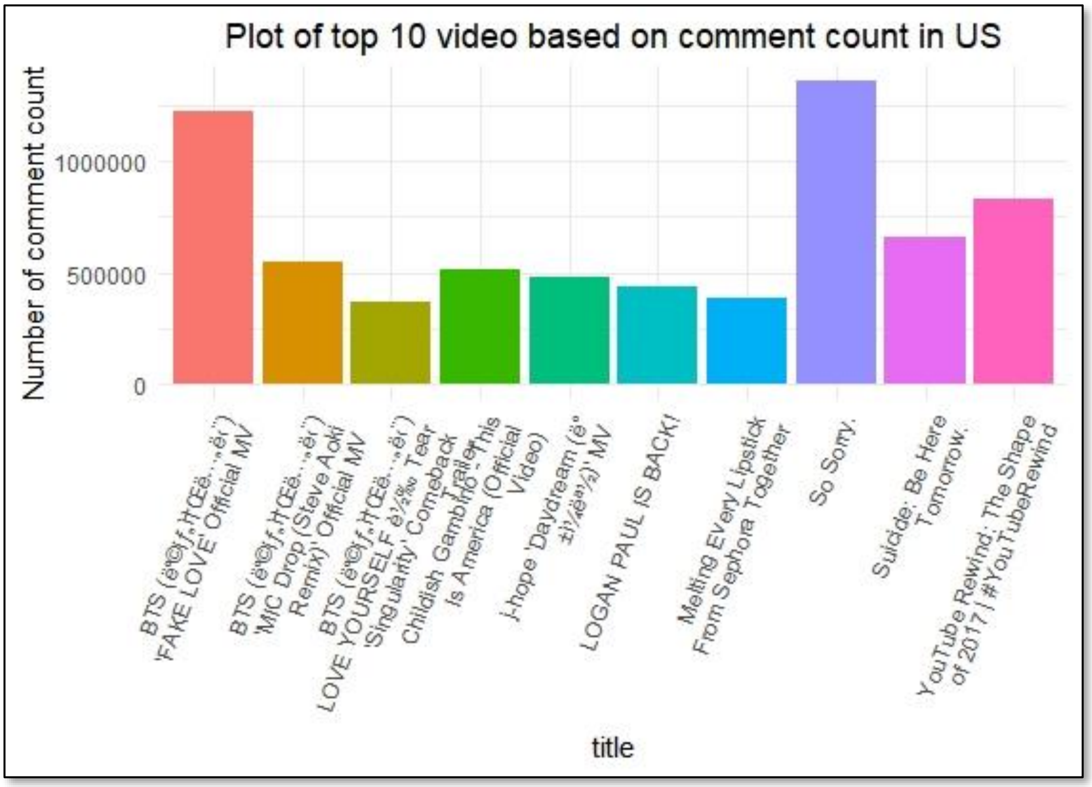
The most get dislike videos in 5 countries was Logan Paul's "so sorry." This video uploaded for apologize after he uploaded unfiltered suicide body during his trip to Japan. This video got a large number of dislikes only in the US, Canada, and the UK, the other countries not in the ranking. As the result, we can assume that Logan Paul had a reputation in the US, Canada, and the UK before that fault. 2nd place dislike video was "Youtube rewind: the shape of 2017". This video is the collection of the videos, people, music, and memes that made in 2017. The interesting point is this video also took place the 4th place of likes video in 5 countries as well.

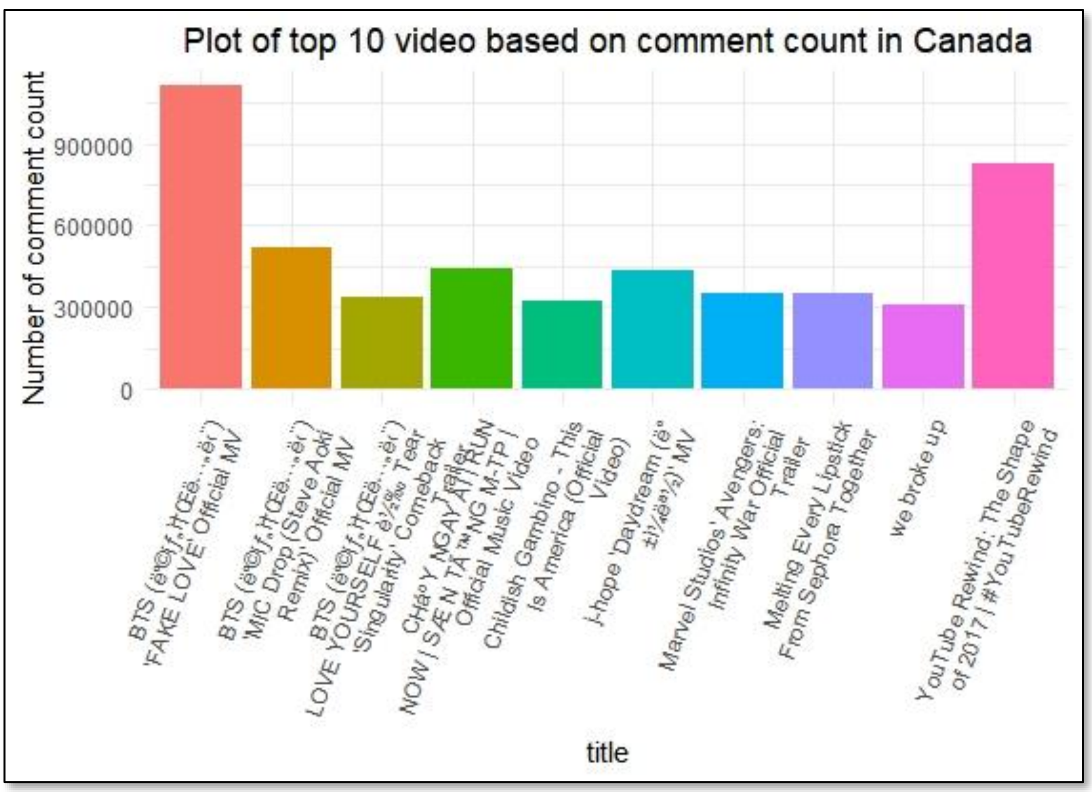
3.2.5.5 *Analysing the dataset of all countries with respect to comment counts*

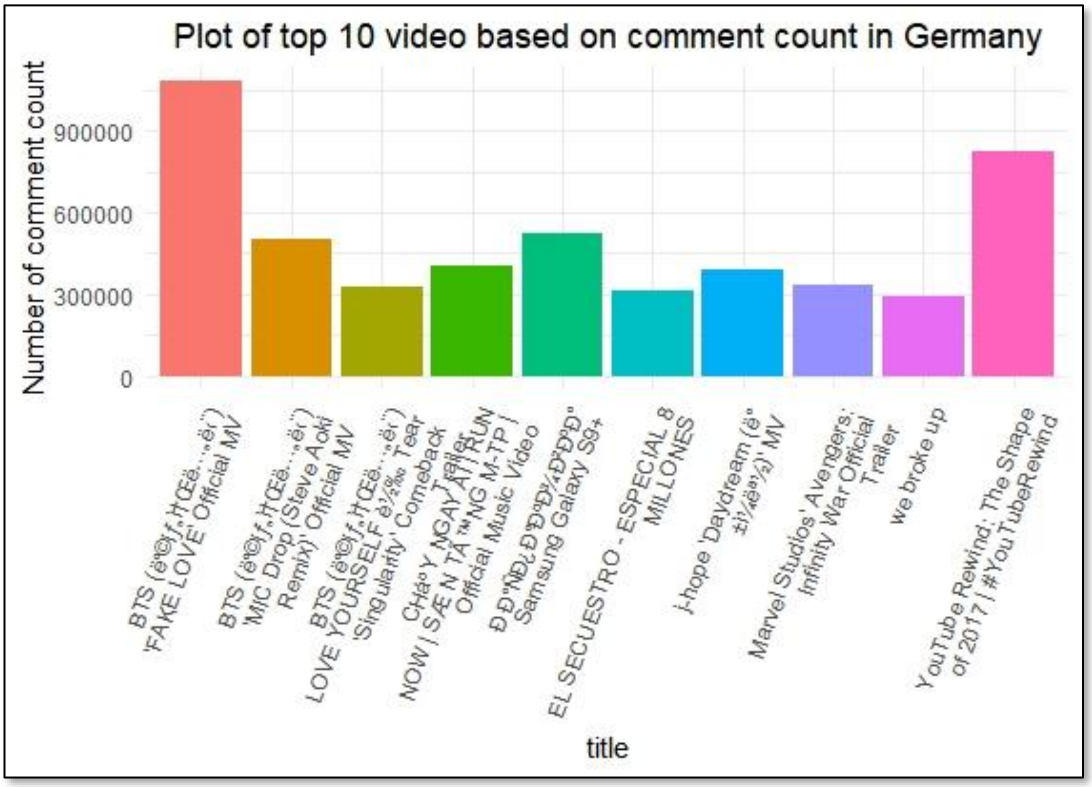
```
#4. comment counts
#1) US
us_video_comment_count <- as.data.table(rbind(us_videos.df))
us_video_comment_count <- us_video_comment_count[,.("comment_count"=round(max(comment_count,na.rm = T), digits = 2)), by=.(title)][order(-comment_count)]
us_video_comment_count <- (us_video_comment_count[1:10,])

ggplot(us_video_comment_count, aes(x = stringr::str_wrap(title, 25), y = comment_count, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on comment count in US", x = "title", y = "Number of comment count")

#2) Canada
ca_video_comment_count <- as.data.table(rbind(ca_videos.df))
ca_video_comment_count <- ca_video_comment_count[,.("comment_count"=round(max(comment_count,na.rm = T), digits = 2)), by=.(title)][order(-comment_count)]
ca_video_comment_count <- (ca_video_comment_count[1:10,])

ggplot(ca_video_comment_count, aes(x = stringr::str_wrap(title, 25), y = comment_count, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on comment count in Canada", x = "title", y = "Number of comment count")


#3) Germany
de_video_comment_count <- as.data.table(rbind(de_videos.df))
de_video_comment_count <- de_video_comment_count[,.("comment_count"=round(max(comment_count,na.rm = T), digits = 2)), by=.(title)][order(-comment_count)]
de_video_comment_count <- (de_video_comment_count[1:10,])

```

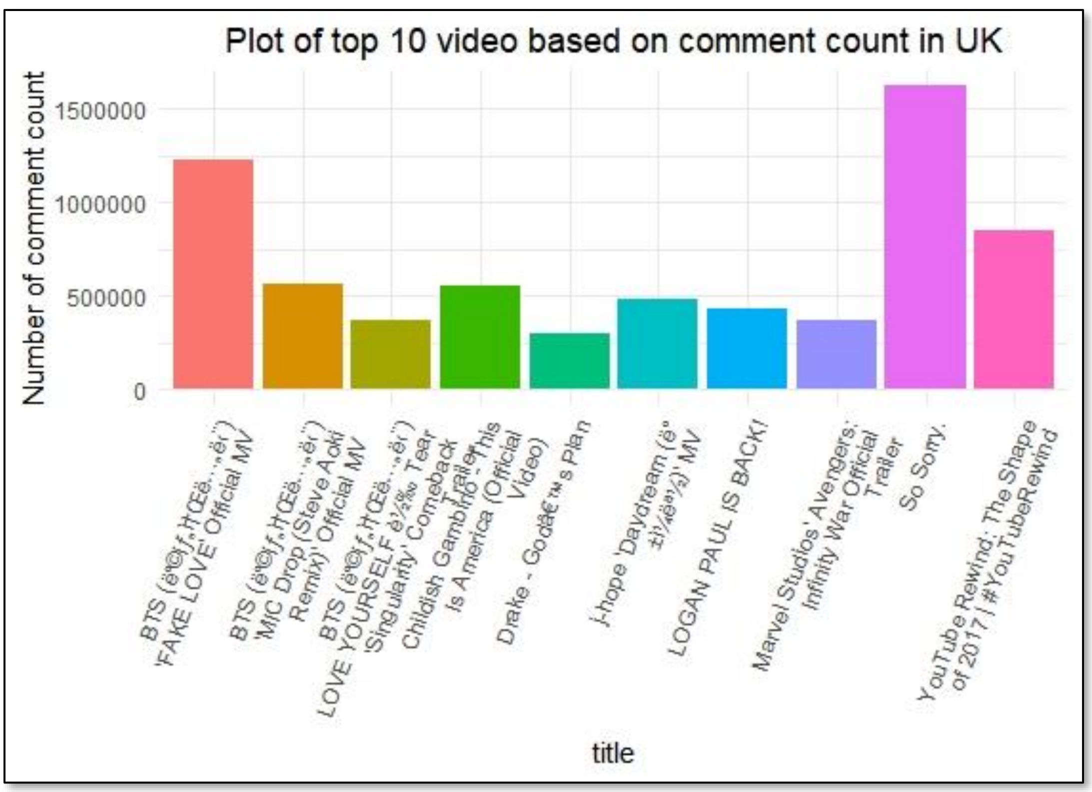
```
ggplot(de_video_comment_count, aes(x = stringr::str_wrap(title, 25), y = comment_count, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on comment count in Germany", x = "title", y = "Number of comment count")

#4) France
fr_video_comment_count <- as.data.table(rbind(fr_videos.df))
fr_video_comment_count <- fr_video_comment_count[,.("comment_count"=round(max(comment_count,na.rm = T), digits = 2)), by=.(title)][order(-comment_count)]
fr_video_comment_count <- (fr_video_comment_count[1:10,])

ggplot(fr_video_comment_count, aes(x = stringr::str_wrap(title, 25), y = comment_count, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on comment count in France", x = "title", y = "Number of comment count")

#5) the UK

gb_video_comment_count <- as.data.table(rbind(gb_videos.df))
gb_video_comment_count <- gb_video_comment_count[,.("comment_count"=round(max(comment_count,na.rm = T), digits = 2)), by=.(title)][order(-comment_count)]
gb_video_comment_count <- (gb_video_comment_count[1:10,])

ggplot(gb_video_comment_count, aes(x = stringr::str_wrap(title, 25), y = comment_count, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on comment count in UK", x = "title", y = "Number of comment count")

#6) Overall
yt_video_comment_count <- as.data.table(rbind(Youtube_videos.df))
yt_video_comment_count <- yt_video_comment_count[,.("comment_count"=round(max(comment_count,na.rm = T), digits = 2)), by=.(title)][order(-comment_count)]
yt_video_comment_count <- (yt_video_comment_count[1:10,])
head(yt_video_comment_count)

ggplot(yt_video_comment_count, aes(x = stringr::str_wrap(title, 25), y = comment_count, fill = factor(title)), ) +
  geom_bar(stat = "identity") + theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) + theme(axis.text.x = element_text(angle = 70,hjust = 1),legend.position = "none") +
  labs(title = "Plot of top 10 video based on comment count in 5 countries", x = "title", y = "Number of comment count")

```

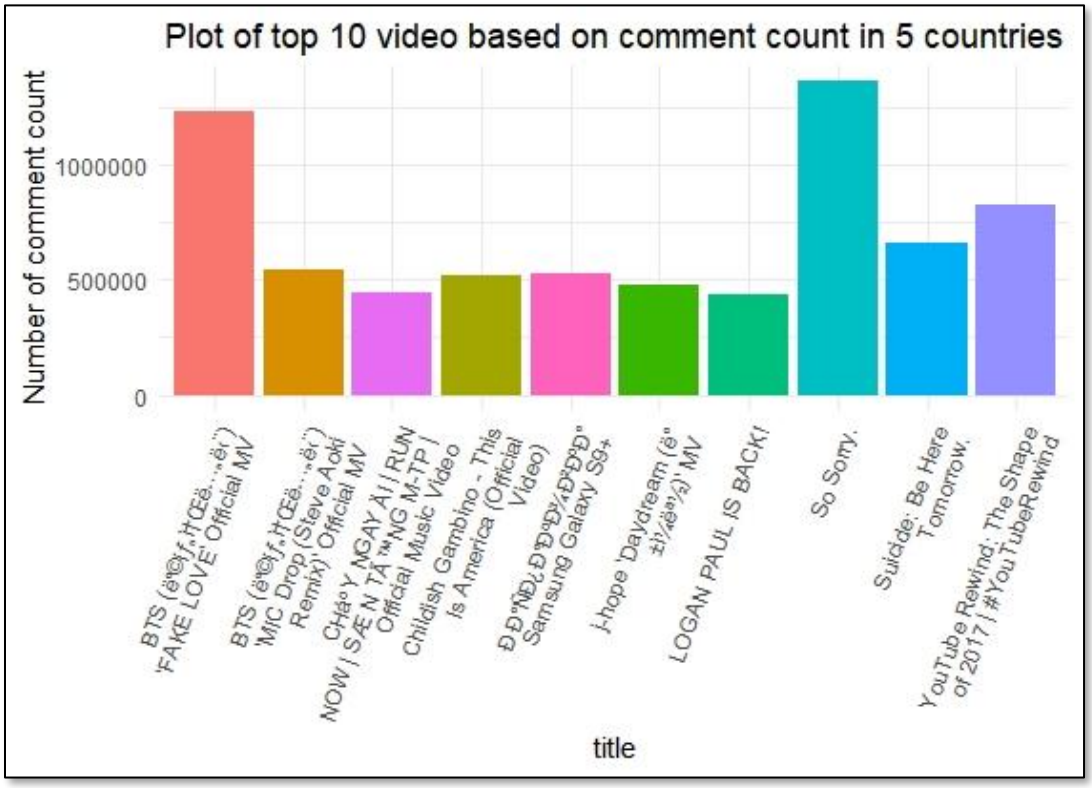
Table 3.16 R code for most comment count of a video for a different region.


Plot of top 10 video based on comment count in US
Number of comment count
0
500000
1000000
title
BTS 'FAKE LOVE' Official MV
BTS 'MIC Drop (Steve Aoki Remix)' Official MV
BTS LOVE YOURSELF Tear 'Singularity' Comeback Trailer
Childish Gambino - This Is America (Official Video)
j-hope 'Daydream' MV
LOGAN PAUL IS BACK!
Melting EVery Lipstick From Sephora Together
So Sorry.
Suicide: Be Here Tomorrow.
YouTube Rewind: The Shape of 2017 | #YouTubeRewind


Plot of top 10 video based on comment count in Canada
Number of comment count
0
300000
600000
900000
title
BTS 'FAKE LOVE' Official MV
BTS 'MIC Drop (Steve Aoki Remix)' Official MV
BTS LOVE YOURSELF Tear 'Singularity' Comeback Trailer
Official Music Video
Childish Gambino - This Is America (Official Video)
j-hope 'Daydream' MV
Marvel Studios' Avengers: Infinity War Official Trailer
Melting EVery Lipstick From Sephora Together
we broke up
YouTube Rewind: The Shape of 2017 | #YouTubeRewind


Plot of top 10 video based on comment count in Germany
Number of comment count
0
300000
600000
900000
title
'FAKE LOVE' Official MV
'MIC Drop (Steve Aoki Remix)' Official MV
LOVE YOURSELF 'Singularity' Comeback Trailer
Official Music Video
Samsung Galaxy S9+
EL SECUESTRO - ESPECIAL 8 MILLONES
j-hope 'Daydream' MV
Marvel Studios' Avengers: Infinity War Official Trailer
we broke up
YouTube Rewind: The Shape of 2017 | #YouTubeRewind


Plot of top 10 video based on comment count in UK
Number of comment count
0
500000
1000000
1500000
title
'FAKE LOVE' Official MV
'MIC Drop (Steve Aoki Remix)' Official MV
LOVE YOURSELF 'Singularity' Comeback Trailer
Childish Gambino - This Is America (Official Video)
j-hope 'Daydream' MV
LOGAN PAUL IS BACK!
Marvel Studios' Avengers: Infinity War Official Trailer
So Sorry.
YouTube Rewind: The Shape of 2017 | #YouTubeRewind



Figures 3.17 Top 10 video based on comment count in 5 countries

The most get comment count video in 5 countries was Logan Paul's "so sorry.". As we see before, this video is the top of dislikes video. On the other hand, 2nd place took "BTS-Fake love". BTS took 3rd place for views, 1stplace for likes. both likes and dislikes from Youtube users; fans and haters are related to the number of comment count of videos. We can assume BTS is one of the hottest singers in these countries. 2nd place was the most get like video, "This is America" and 3rd place was the most get dislike video, "Youtube Rewind 2017".

3.2.6 The ratio of likes in the total number of likes and dislikes by categories

In order to find out the Trading video, we eliminated the missing value for likes, dislikes and comment counts of the dataset. After that, we analyze which categories get high like rate compare with the total like and dislike.

```r
#5. Like percentage by video categories

# removing the thunmbnail link column
Youtube_videos1.df <- Youtube_videos.df[,-12]

# setting all 0 values for likes, dislikes and comments as NA
Youtube_videos1.df[, 8:11][Youtube_videos1.df[, 8:11] == 0] <- NA

# counting the missing values
sapply(Youtube_videos1.df, function(x) sum(is.na(x)))
# eliminate the missing values and Confirm the missing values elimination
Youtube_videos1.df <- Youtube_videos1.df %>%
  filter(!is.na(likes) & !is.na(dislikes) & !is.na(comment_count))
sapply(Youtube_videos1.df, function(x) sum(is.na(x)))

# a new derived variable that gives the ratio of likes and dislikes for a vid
eo
Youtube_videos1.df$like_percentage <- Youtube_videos1.df$likes/(Youtube_video
s1.df$dislikes+Youtube_videos1.df$likes)*100

like_count.df <- as.data.frame(sort(tapply(Youtube_videos1.df$like_percentage
,Youtube_videos1.df$category_title,mean), decreasing = TRUE))
like_count.df <- cbind(rownames(like_count.df), data.frame(like_count.df, row
.names=NULL))
colnames(like_count.df) <-c('type','like_percentage')

ggplot(like_count.df, aes(x = reorder(type,-
like_percentage), y = like_percentage,fill = factor(type)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_t
ext(angle = 60,hjust = 1),legend.position = "none") +
  labs(title = "Like Percentage by Video Categories", x = "Video Category", y
 = "Like Percentage (%)") +
  geom_text(aes(label=round(like_percentage,2)), vjust=-0.3, size=2.5)
```

Table 3.17 showing R code for like percentage [Likes / (Likes + Dislikes)]

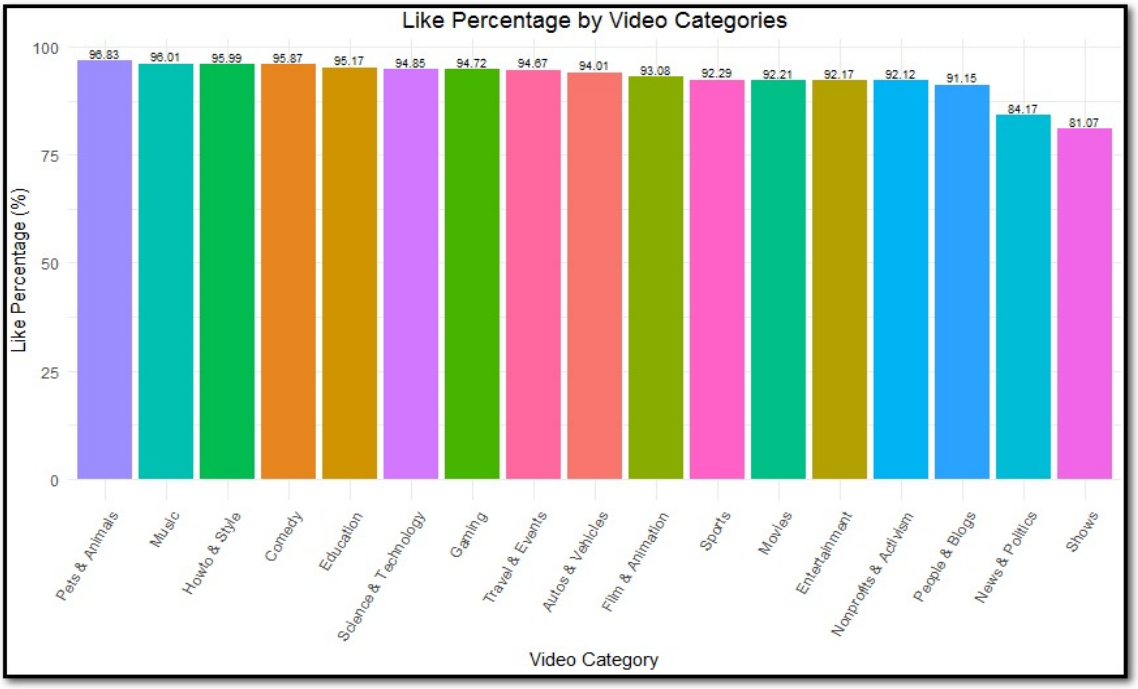


Figure 3.18 Top 10 video based on Like percentage by video categories

As above bar chart, Pets & Animal, Music, How to & Style, Comedy, and Education categories got over 95% of like percentage in the total number of likes and dislikes. On the other hand, News & Politics and Shows got less than 90% of like percentage in the total number of likes and dislikes. A politics category can be explained that it included controversial issues. However, the result of shows is surprising. Most of these videos were reviews of irritating contents. Moreover, most of the episode started with the climax of the story to users get bias to the characters.

3.2.7 Gap analysis being a trending video after published

Here, we analyzed the time the video has taken to trend after being published.

```
#Time between trending and published.
ggplot(Youtube_videos.df[trendingday<30],aes(as.factor(trendingday),fill=as.factor(trendingday)))+geom_bar()+guides(fill="none")+
  labs(title=" Time between published and trending by countries",subtitle="In days")+
  xlab(NULL)+ylab(NULL) + facet_grid(rows = vars(Location)) +
  theme_minimal() +   theme(plot.title = element_text(hjust = 0.5)) +
  coord_cartesian(xlim = c(0, 40))
```

Table 3.18 R code for Time between trending and publishing

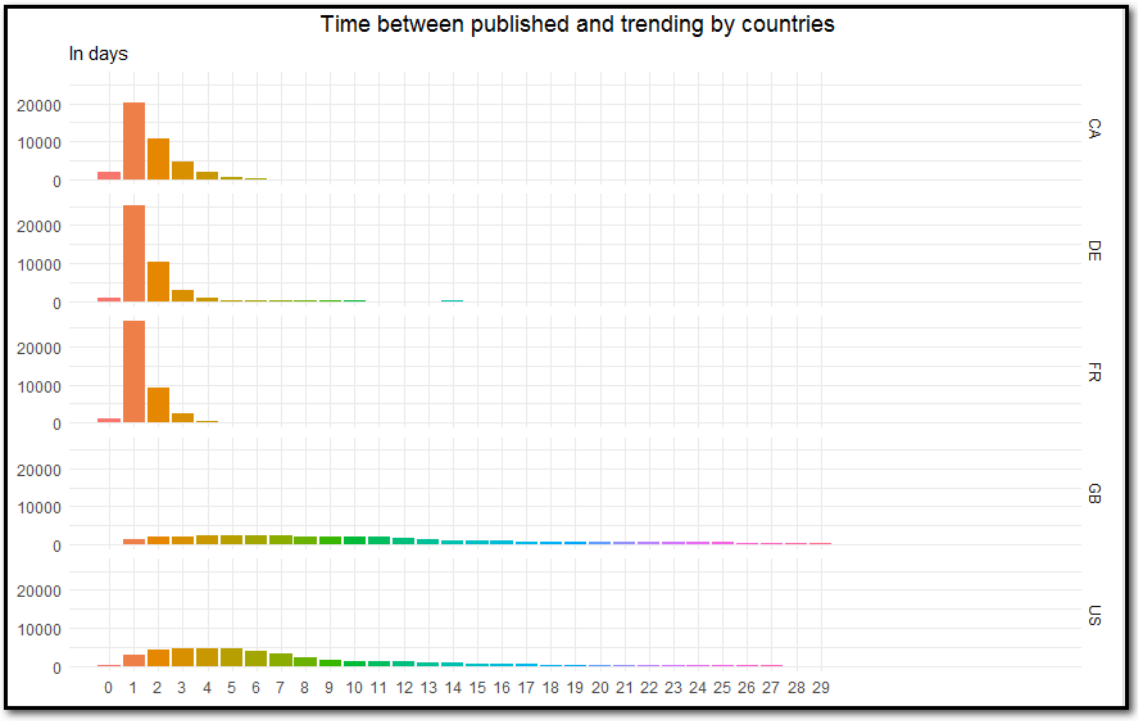


Figure 3.19 Time between published and trending for each region

In Germany and France, it takes a little time to reach the trending page, 1 or 2 days usually. Similar case to Canada, but here is higher the 2-day time difference, it takes a little longer to reach the trending page.

In the UK and the US, it takes a longer time for the video to get trended. The number of hits in the initial days is very low when compared to Canada, France or Germany.

Now, analyzing the trend to find how many days it took for the video to get trending after publishing for the overall regions.

In order to analysis how many days take being trending video after published video, we used data pre-processing. We organized the table frequent ordered.

```
1. #Chaning the date format for easy date manipulations.
2. us_videos.df$trending_date <- ydm(us_videos.df$trending_date)
3. gb_videos.df$trending_date <- ydm(gb_videos.df$trending_date)
4. ca_videos.df$trending_date <- ydm(ca_videos.df$trending_date)
5. de_videos.df$trending_date <- ydm(de_videos.df$trending_date)
6. fr_videos.df$trending_date <- ydm(fr_videos.df$trending_date)
7. us_videos.df$publish_time <- ymd(substr(us_videos.df$publish_time,start = 1,s
   top = 10))
8. gb_videos.df$publish_time <- ymd(substr(gb_videos.df$publish_time,start = 1,s
   top = 10))
```

```
ca_videos.df$publish_time <- ymd(substr(ca_videos.df$publish_time,start = 1,stop = 10))
de_videos.df$publish_time <- ymd(substr(de_videos.df$publish_time,start = 1,stop = 10))
fr_videos.df$publish_time <- ymd(substr(fr_videos.df$publish_time,start = 1,stop = 10))

#calculating date difference between published time and trending date to find the day it became trending
us_videos.df$trendingday = us_videos.df$trending_date - us_videos.df$publish_time
gb_videos.df$trendingday = gb_videos.df$trending_date - gb_videos.df$publish_time
ca_videos.df$trendingday = ca_videos.df$trending_date - ca_videos.df$publish_time
de_videos.df$trendingday = de_videos.df$trending_date - de_videos.df$publish_time
fr_videos.df$trendingday = fr_videos.df$trending_date - fr_videos.df$publish_time

# deffierence between publising and trending
t.df <- table(Youtube_videos1.df$trendingday)
t.df <- as.data.frame(t.df)
t.order <- t.df[order(-t.df$Freq),]
head(t.order,10)
```

Table 3.19 R code for like percentage analysis

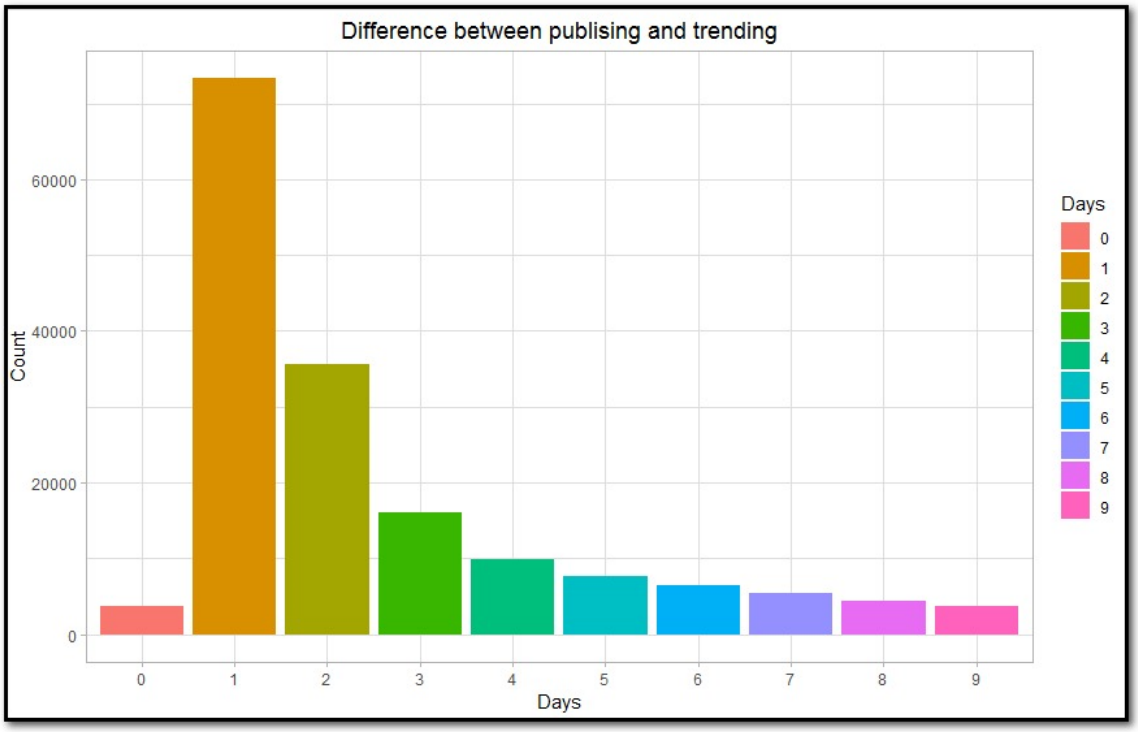


Figure 3.20 difference between publishing and trending in Overall Region

As the result, the video will most likely start trending within 2 days of being published. The meaning of this is most of the trending videos spread within 2days. This result can show how much videos spread fast through this video streamer service.

3.2.8 Top 10 Trend videos in the 5 countries

To estimate how much trendy of all videos, we build a trend score for each video. Firstly, we selected data over the 1 million views for considering popularity. Secondly, after normalization of each number of views, likes, comment count, and trending day. Then we set a trend score for estimating tendency of the trend. The formula of trend score is **abs (views + likes + comment count – trending day[2])**

```
# The Most trending video
# videos over the one million views
high_youtube <- Youtube_videos1.df %>%
  filter(views >= 1000000)
high_youtube_norm$trendscore <- abs(high_youtube_norm$views + high_youtube_norm$likes + high_youtube_norm$comment_count - high_youtube_norm$trendingday)
high_youtube_trend <- as.data.table(rbind(high_youtube_norm))
high_youtube_trend <- high_youtube_trend[,.("trendscore"=round(max(trendscore,na.rm = T), digits = 2)), by=.(title)][order(-trendscore)]
high_youtube_trend <- (high_youtube_trend[1:10,])
head(high_youtube_trend,10)

ggplot(subset(high_youtube_trend, trendscore > 7.5), aes(x = stringr::str_wrap(title, 25), y = trendscore,fill = factor(trendscore)), ) +
  geom_bar(stat = "identity") + theme_light() + theme(axis.text.x = element_text(angle = 65,hjust = 1),legend.position = "none") +
  labs(title = "Top 10 Trend Videos in the World", x = "", y = "Trend Score") +
  geom_text(aes(label=round(trendscore,2)), vjust=-0.3, size=2.5)
```

Table 3.19 R code for Trending Score analysis

[2] Trending day = Trend date - publishing time

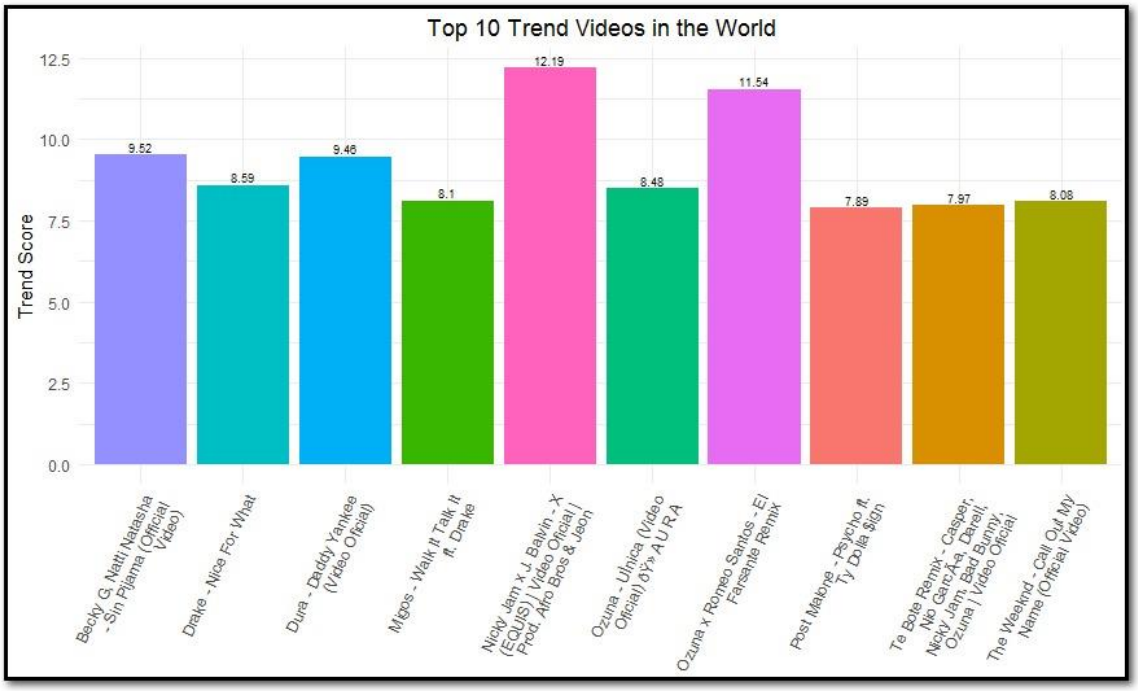


Figure 3.20 Top 10 video based on Trending Score analysis in 5 countries

As a result, the top 10 of Trend video in 5 countries are all music categories. The interesting point is 60% of music video released from South America. Top 1~4 trend videos are all Spanish music video from South America, Hip-Hop & Dance music. These videos views count over 1Billion (11/28/2018). We did not add any South America countries, nevertheless, Latin America's music videos got high scores in trend aspect.

4 Data Mining Techniques

4.1 Data Mining Methods Used:

Data mining refers to business analytics methods that go beyond counts, descriptive techniques, reporting, and methods based on business rules. While we do introduce data visualization, which is commonly the first step into more advanced analytics, the book focuses mostly on the more advanced data analytics tools. Specifically, it includes statistical and machine-learning methods that inform decision-making, often in an automated fashion. Prediction is

typically an important component, often at the individual level. (Data Mining for Business Analytics, 2018, p. 5)

We have used the below data mining methodologies in our project.

- Linear Regression
- KNN
- CART
- Linear Discriminant Analysis

The below section discusses the advantages of these models, their features and weakness of these algorithms.

4.1.1 Linear Regression

What is linear regression? Its types like backward, stepwise – need to write a gist on that (probably a small paragraph from the text) explaining the below points.

Linear Regression is a statistical model which is used to fit a relationship between a numerical outcome variable Y (also called target or dependent variable) and a set of predictors X1, X2, …, Xp (also referred to as independent variables).

The following function approximates the relationship between the predictors and outcome variable:

$$Y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta pXp + \varepsilon$$

Where $\beta 0 \ldots \beta p$ are coefficients and ε is noise or unexplained part. Data are then used to estimate coefficients and to quantify the noise. (Data Mining for Business Analytics, 2018)

Linear regression is great when the relationship between covariates and the response variable is known to be linear. This is good as it shifts focus from statistical modeling and to data analysis and preprocessing. It is great for learning to play with data without worrying about the intricate details of the model.

A clear disadvantage is that it oversimplifies many problems as in it leads to overfitting of data. Often, covariates and response variables don't exhibit a linear relationship, hence linear regression cannot be used.

In this section, we have taken the numerical features of the YouTube videos data, that is, a number of views, likes, dislikes and comment count and based on these, we have predicted what category a video belongs to, after splitting the data into train and test sets. This will be done using various machine learning models. The goal here is to compare various classification models and to decide which model fits the best for our data set.

4.1.2 Linear Discriminant Analysis

Logistic regression is a simple and powerful linear classification algorithm as it is based upon the concept of searching for a linear combination of variables (predictors) that best separates two classes (targets). LDA makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made.

We have used the MASS library to execute the LDA model. Prior probabilities of category id classes are too low but the proportion of trace of coefficients of Linear Discriminant Analysis is giving good classification rates. DA makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made.

4.1.3 CART

Prediction Trees are used to predict a response or class YY from input X1, X2,…,XnX1, X2,…, Xn. If it is a continuous response it's called a regression tree, if it is categorical, it's called a classification tree. Tree-based models split the data according to certain cutoff values in the features multiple times. Splitting means that different subsets of the dataset are created, where each instance belongs to one subset. The final subsets are called terminal or leaf nodes and the intermediate subsets are called internal nodes or split nodes.

At each node of the tree, we check the value of one the input XiXi and depending on the (binary) answer we continue to the left or to the right subbranch. When we reach a leaf we will find the prediction (usually it is a simple statistic of the dataset the leaf represents, as the most common value from the available classes). It helps us explore the structure of a set of data while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome.

We have used part and Caret library for classification trees modeling. We have pruned the tree to avoid overfitting the data. We wanted to select a tree size that minimizes the cross-validated error, the xerror column printed by printcp(). We have pruned the tree to reduce the size using prune (fit, cp=). We have printed out the cp table of cross-validation errors. The R-squared for a regression tree is 1 minus rel error. xerror (or relative cross-validation error where "x" stands for "cross") is a scaled version of an overall average of the 5 out-of-sample errors across the 5 folds.

The major disadvantage of using trees is that it is quite unstable, so a few changes in the training dataset might create a completely different tree. That's because each split depends on the parent split. And if a different feature gets selected as the first split feature, the whole tree structure will change. It does not generate confidence in the model if the structure flips so easily.

4.1.4 KNN

We have executed the kNN algorithm as it is one of the simplest classification algorithm but one of the most used learning algorithms. Not compared to better-supervised learning models but generally, the k-NN model gives high accuracy in general because the model structure is determined from the data. And in our case also, kNN model gives the highest accuracy among all the models. When KNN is used for classification, the output can be calculated as the class with the highest frequency from the K-most similar instances. Class probabilities can be calculated as the normalized frequency of samples that belong to each class in the set of K most similar instances for a new data instance. For example, in a binary classification problem (class is 0 or 1):

p(class=0) = count(class=0) / (count(class=0) + count(class=1))

And because the algorithm stores almost all the training data, it is computationally expensive and takes more time. It requires large memory for storing the entire training dataset for prediction. For our dataset also, N is very high and so the prediction stage is bit slow. Also, KNN requires scaling of data because KNN uses the Euclidean distance between two data points to find nearest neighbors. Euclidean distance is sensitive to magnitudes. The features with high magnitudes will weigh more than features with low magnitudes. So, we performed normalization on the data and then performed the kNN model on normalized data for better classification accuracy.

4.2 Linear Regression

We have used Linear Regression technique as explained before in our dataset. The Linear Regression technique is applied to each region dataset where we have tried to predict the number of views based on the other different independent variables viz. likes, dislikes, comments count and category id.

We have also tried to implement exhaustive search and stepwise regression to find the predictor accuracy in the model.

Then we have discussed on the model performance using Lift and Decile charts. These charts are drawn for one linear model of each region and the performance is explained.

The datasets are divided into training and validation dataset in the ratio 60:40 and the model is developed on the training dataset and used to predict the validation dataset.

We have considered View as the dependent variable and likes, dislikes, comment count and category ID as an independent variable to create the model.

4.2.1 Linear Regression in US Dataset

We have considered View as the dependent variable and likes, dislikes, comment count and category ID as an independent variable to create the model.

4.2.1.1 *Data Partition:*

```
### Partitioning Data
set.seed(1)  # set seed for reproducing the partition

#View(Youtube_videos.df)
train.index <- createDataPartition(us_videos.df$views, p=0.6 , list=FALSE)
train.df <- us_videos.df[train.index,]
nrow(train.df)
valid.df <- us_videos.df[-train.index, ]
nrow(valid.df)
```

Table 4.1 showing R code for Data partition of US Videos

```
> ### Partitioning Data
> set.seed(1)  # set seed for reproducing the partition
>
> #View(Youtube_videos.df)
> train.index <- createDataPartition(us_videos.df$views, p=0.6 , list=FALSE)
> train.df <- us_videos.df[train.index,]
> nrow(train.df)
[1] 24572
> valid.df <- us_videos.df[-train.index, ]
> nrow(valid.df)
[1] 16377
>
```

Figure 4.1 showing number of Training and Validation records considered for the model.

4.2.1.2 *Running Regression and Prediction*

```
### Run regression
us_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = train.df)
us_videos.lm
options(scipen = 999)
summary(us_videos.lm)

### Generate predictions
us_videos.lm.pred <- predict(us_videos.lm, valid.df)
some.residuals <- valid.df$views[1:20000] - us_videos.lm.pred[1:20000]
#correlation
cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
residuals_List<-
data.frame("Predicted" = us_videos.lm.pred[1:20000], "Actual" = valid.df$views[1:20000],
                           "Residual" = some.residuals)
residuals_List
```

Table 4.2 showing R code for Linear Regression and prediction on US Video training and Validation dataset.

From the result of the linear regression below, we could see that the adjusted R-Square value is 77.02%. The model is moderate in predicting the number of views.

```
> ### Run regression
> us_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = train.df)
> us_videos.lm

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Coefficients:
  (Intercept)          likes       dislikes  comment_count    category_id
    411773.84          34.28          82.34          -94.38       -5633.22

> options(scipen = 999)
> summary(us_videos.lm)

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-36894034   -432716   -230501    134529  68745014

Coefficients:
                Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   411773.8441  63002.6339   6.536      0.0000000000645 ***
likes             34.2797      0.1664 205.967 < 0.0000000000000002 ***
dislikes          82.3400      1.1688  70.447 < 0.0000000000000002 ***
comment_count    -94.3800      1.2578 -75.035 < 0.0000000000000002 ***
category_id    -5633.2196   2876.5313  -1.958                0.0502 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3338000 on 24567 degrees of freedom
Multiple R-squared:  0.7703,    Adjusted R-squared:  0.7702
F-statistic: 2.059e+04 on 4 and 24567 DF,  p-value: < 0.00000000000000022

>
> ### Generate predictions
> us_videos.lm.pred <- predict(us_videos.lm, valid.df)
> some.residuals <- valid.df$views[1:20000] - us_videos.lm.pred[1:20000]
> #correlation
> cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
                views      likes   dislikes comment_count category_id
views          1.0000000  0.8430785  0.4309130     0.59703821 -0.17334235
likes          0.8430785  1.0000000  0.4286766     0.79060450 -0.17428186
dislikes       0.4309130  0.4286766  1.0000000     0.72229220 -0.03211910
comment_count  0.5970382  0.7906045  0.7222922     1.00000000 -0.07165537
category_id   -0.1733423 -0.1742819 -0.0321191    -0.07165537  1.00000000
> residuals_List<-data.frame("Predicted" = us_videos.lm.pred[1:20000], "Actual" = valid.df$views[1:20000],
+                            "Residual" = some.residuals)
> residuals_List
          Predicted    Actual       Residual
1       1089610.3398   1080795    -8815.3398
2      10731864.4637  12887949   2156084.5363
3        569932.7058    153898   -416034.7058
4        479128.9835     28013   -451115.9835
```

Figure 4.2 Showing the Regression and Prediction result of US Video

4.2.1.3 Accuracy Measure:

A key component of most measures is the difference between actual y and predicted y (error)

Some Measures of Error

MAE or MAD: Mean absolute error (deviation) Gives an idea of the magnitude of errors

Average error: Gives an idea of systematic over- or under-prediction

MAPE: Mean absolute percentage error

RMSE (root-mean-squared-error): Square the errors, find their average, take the square root

Total SSE: Total sum of squared errors (Data Mining for Business Analytics, 2018)

```
### Accuracy measures

accuracy(us_videos.lm.pred, valid.df$views)
```

Table 4.3 showing R code for Accuracy Measure

```
> accuracy(us_videos.lm.pred, valid.df$views)
                ME     RMSE      MAE        MPE      MAPE
Test set 18232.2 3609929 1211838 -312.1014 337.9702
>
```

Figure 4.3 showing the accuracy measure result

The root means the square error is 3609929 as shown above for this model. We will compare the model effectiveness based on the RMSE value.

4.2.1.4 Exhaustive Search Method

The algorithm searches the whole attribute subset space in breadth-first order. Since the number of subsets for even moderate values of p is very large, after the algorithm creates the subsets and runs all the models, we need some way to examine the most promising subsets and to select from them. The challenge is to select a model that is not too simplistic in terms of excluding important parameters (the model is under-fit), nor overly complex thereby modeling random noise (the model is over-fit).

A very popular criterion to choose the model is based on adjusted R square, which is the proportion of explained variability in the model. Second popular criteria are AIC and BIC which measure the goodness of fit of a model but also include a penalty that is a function of the number of parameters in the model. (Data Mining for Business Analytics, 2018)

```
### Exhaustive Search

search <- regsubsets(views ~ likes+dislikes+comment_count+category_id, data =
 train.df, nbest = 1, nvmax = dim(train.df)[2],
                     method = "exhaustive")
sum <- summary(search)

# show models
sum$which
# show metrics
sum$rsq
sum$adjr2
sum$cp
```

```
13.
14. #dropping two predictors
15.
16. us_videos.lm_new <- lm(views ~ likes+dislikes, data = train.df)
17. us_videos.lm_new
18. options(scipen = 999)
19. summary(us_videos.lm_new)
20.
21. us_videos.lm.new.pred <- predict(us_videos.lm_new, valid.df, type = "response
    ")
22.
23. accuracy(us_videos.lm.new.pred, valid.df$views)
```

Table 4.4 showing R code for Exhaustive Search Linear Regression

In order to find the number of predictors to be included or excluded in the model, we have used the exhaustive search method. In the adjR2 there is no decrease. It keeps on increasing and hence there is no need to do any dimension reduction or predictor removal. We will have to use all the four predictors as all of them are statistically significant which is again proved by exhaustive search. The same is again verified by the backward stepwise method.

```
> # show models
> sum$which
  (Intercept) likes dislikes comment_count category_id
1        TRUE  TRUE    FALSE         FALSE       FALSE
2        TRUE  TRUE    FALSE          TRUE       FALSE
3        TRUE  TRUE     TRUE          TRUE       FALSE
4        TRUE  TRUE     TRUE          TRUE        TRUE
> # show metrics
> sum$rsq
[1] 0.7107814 0.7236652 0.7702443 0.7702802
> sum$adjr2
[1] 0.7107696 0.7236427 0.7702163 0.7702428
> sum$cp
[1] 6362.002109 4986.159917    6.835087    5.000000
>
> #dropping two predictors
>
> us_videos.lm_new <- lm(views ~ likes+dislikes, data = train.df)
> us_videos.lm_new

Call:
lm(formula = views ~ likes + dislikes, data = train.df)

Coefficients:
(Intercept)        likes     dislikes
  424631.71        24.83        21.18

> options(scipen = 999)
> summary(us_videos.lm_new)

Call:
lm(formula = views ~ likes + dislikes, data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-58969567   -495383   -356067     20949  87687459

Coefficients:
             Estimate  Std. Error t value            Pr(>|t|)
(Intercept) 424631.7083  24884.7949   17.06 <0.0000000000000002 ***
likes           24.8332      0.1157  214.60 <0.0000000000000002 ***
dislikes        21.1832      0.9350   22.66 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3707000 on 24569 degrees of freedom
Multiple R-squared:  0.7167,    Adjusted R-squared:  0.7167
F-statistic: 3.108e+04 on 2 and 24569 DF,  p-value: < 0.00000000000000022

>
> us_videos.lm.new.pred <- predict(us_videos.lm_new, valid.df, type = "response"
)
>
> accuracy(us_videos.lm.new.pred, valid.df$views)
               ME     RMSE      MAE       MPE     MAPE
Test set 33133.94 4051954 1257304 -460.8055 477.7766
>
```

Figure 4.4 Showing result of the exhaustive search and removing predictor after analyzing the search result

4.2.1.5 Stepwise Regression

Stepwise Regression is a combination of forwarding and backward selections. You start with no predictors, then sequentially add the most contributive predictors (like forward selection). After adding each new variable, remove any variables that no longer provide an improvement in the model fit (like backward selection). (Kassambara, Statistical tools for high-throughput data analysis) (Kassambara, Statistical tools for high-throughput data analysis)

```
### Stepwise Regression
us_videos.lm.stepwise <- step(us_videos.lm, direction = "backward")
summary(us_videos.lm.stepwise)  # Which variables were dropped/added?
us_videos.lm.stepwise.pred <- predict(us_videos.lm.stepwise, valid.df)
accuracy(us_videos.lm.stepwise.pred, valid.df$views)
```

Table 4.5 showing R code for Stepwise Regression

```
> ### Stepwise Regression
> us_videos.lm.stepwise <- step(us_videos.lm, direction = "backward")
Start:  AIC=738196.8
views ~ likes + dislikes + comment_count + category_id

                Df          Sum of Sq                 RSS    AIC
<none>                                 273793861907321664 738197
- category_id    1     42741204078336 273836603111400000 738199
- dislikes       1   553094097755002 88 329103271682821952 742716
- comment_count  1   62748708311398144 336542570218719808 743265
- likes          1 472787907110734528 746581769018056192 762844
> summary(us_videos.lm.stepwise)  # Which variables were dropped/added?

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-36894034   -432716   -230501    134529  68745014

Coefficients:
              Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   411773.8441  63002.6339   6.536      0.0000000000645 ***
likes            34.2797      0.1664 205.967 < 0.0000000000000002 ***
dislikes         82.3400      1.1688  70.447 < 0.0000000000000002 ***
comment_count   -94.3800      1.2578 -75.035 < 0.0000000000000002 ***
category_id   -5633.2196   2876.5313  -1.958                0.0502 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3338000 on 24567 degrees of freedom
Multiple R-squared:  0.7703,    Adjusted R-squared:  0.7702
F-statistic: 2.059e+04 on 4 and 24567 DF,  p-value: < 0.00000000000000022

> us_videos.lm.stepwise.pred <- predict(us_videos.lm.stepwise, valid.df)
> accuracy(us_videos.lm.stepwise.pred, valid.df$views)
                ME     RMSE      MAE       MPE     MAPE
Test set 18232.2 3609929 1211838 -312.1014 337.9702
```

Figure 4.5 showing Stepwise Regression result

From the results of the backward stepwise regression, we can see that there are no predictors should be removed and the adjR2 value is 77.06% which is the same as the initial linear model. Hence, we could decide that the initial model is good in prediction and when compared to the exhaustive regression model as the RMSE value is low and also the adjusted R square value is also higher.

4.2.1.6 Lift and Decile Chart

Lift Charts:

Lift charts also called lift curves, gains curves or gains charts, helps determine how effectively we can "skim the cream" by selecting a relatively small number of records and getting a relatively large portion of the responders. The input required to construct a lift curve is the validation dataset that has been "scored" by appending to each record the propensity that it will belong to a given class.

Interpreting Lift Chart:

Lift chart consists of lift curve and a baseline. The greater the area between the lift curve and the baseline, the better the model.

Restrictions on Lift Charts:

Lift charts require that the predictable attribute be a discrete value. In other words, you cannot use lift charts to measure the accuracy of models that predict continuous numeric values. The prediction accuracy for all discrete values of the predictable attribute is shown in a single line. If you want to see prediction accuracy lines for any individual value of the predictable attribute, you must create a separate lift chart for each targeted value.

You can add multiple models to a lift chart if the models all have the same predictable attribute.

Decile Charts:

The information from the lift charts can be portrayed as a decile chart. The decile chart aggregates all the information into 10 buckets.

How the Decile Analysis is Calculated

1. The hold-out or validation sample is scored according to the model being tested. 2. The records are sorted by their predicted scores in descending order and divided into ten equal-sized bins or deciles. The top decile contains 10% of the population most likely to respond and the bottom decile contains 10% of the population least likely to respond, based on the model scores.
3. The deciles and their actual response rates are graphed on the x and y-axes, respectively.

After the decile analysis is built, you'll want to look at the height of the bars in relation to one another. Deciding whether a model is worth moving forward with depends on the pattern you see when viewing the decile analysis.
Ideal Situation: The Staircase Effect
When you're looking at a decile analysis, you want to see a staircase effect; that is, you'll want the bars to descend in order from left to right. (Cumulative Gains and Lift Charts, The University of Regina, Pg 1)

```
### Generate Lift Chats
gain <- gains(us_videos.lm.pred, valid.df$views)
gain
### cumulative lift chart
options(scipen=999)
### Plot graph
plot(c(0,gain$cume.pct.of.total*sum(valid.df$views))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative Views", main="Lift Chart",
     col = "blue1", type="l")
### baseline
lines(c(0,sum(valid.df$views))~c(0,dim(valid.df)[1]), col="brown2", lty=2)


### Decile-wise lift chart
barplot(gain$mean.resp/mean(valid.df$views), names.arg = gain$depth,
        xlab = "Percentile", ylab = "Mean Response", main = "Decile-
wise lift chart",
```

```
17.         col = "coral1")
```

Table 4.6 showing R Code for Lift and Decile chart

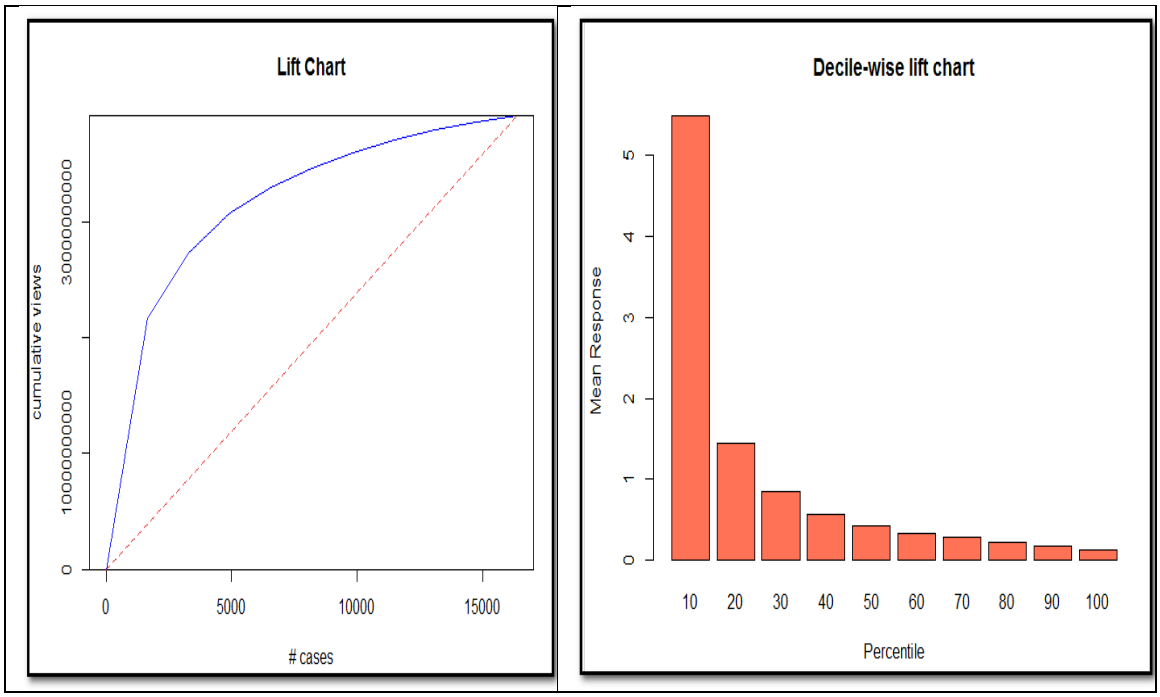


Figure 4.6 showing Lift chart and Decile chart for US Videos

As Explained above, the lift chart and decile charts help us to predict the best values according to our needs. As can be seen from the lift chart above for US dataset, the regression model works much better than random prediction since the area between the blue line (response after predictive model) and the red line (response before predictive model) is more.

To select the videos with top views, the decile chart comes handy. In this case, we could see that the top 10% of the videos have views which are 5 times more than randomly selecting 10% of videos from the dataset. So, for selecting videos with higher views, decile and lift chart comes handy.

4.2.2 Linear Regression in UK Dataset

We have considered View as the dependent variable and likes, dislikes, comment count and category ID as an independent variable to create the model.

4.2.2.1 *Data Partition:*

```
#linear regression for gb_videos

### Partitioning Data
set.seed(2)  # set seed for reproducing the partition
train.index <- createDataPartition(gb_videos.df$views, p=0.6 , list=FALSE)
train.df <- gb_videos.df[train.index,]
nrow(train.df)
valid.df <- gb_videos.df[-train.index, ]
nrow(valid.df)
```

Table 4.7 showing R code for Data partition of UK Videos

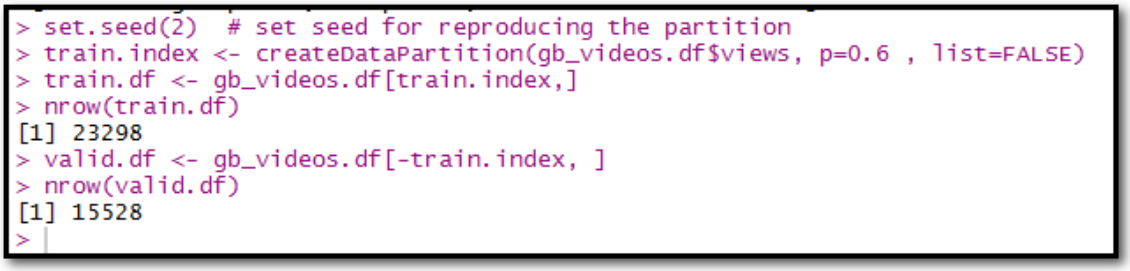


Figure 4.7 showing number of Training and Validation records considered for the model.

4.2.2.2 *Running Regression and Prediction*

```
gb_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = train.df)
gb_videos.lm
options(scipen = 999)
summary(gb_videos.lm)

### Generate predictions
gb_videos.lm.pred <- predict(gb_videos.lm, valid.df)
some.residuals <- valid.df$views[1:20000] - gb_videos.lm.pred[1:20000]
#correlation
cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
residuals_List<-
data.frame("Predicted" = gb_videos.lm.pred[1:20000], "Actual" = valid.df$views[1:20000],
                           "Residual" = some.residuals)
residuals_List
```

Table 4.8 showing R code for Linear Regression and prediction on GB Video training and Validation dataset.

```
> gb_videos.lm

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Coefficients:
  (Intercept)          likes       dislikes  comment_count    category_id
   -336618.73          62.77         172.11        -284.34       13700.41

> options(scipen = 999)
> summary(gb_videos.lm)

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-99759891   -962780     37289    494328 250381010

Coefficients:
                 Estimate   Std. Error t value            Pr(>|t|)
(Intercept)   -336618.7289  159675.6266  -2.108              0.0350 *
likes             62.7660       0.2996 209.470 <0.0000000000000002 ***
dislikes         172.1121       2.1529  79.943 <0.0000000000000002 ***
comment_count    -284.3379       2.9356 -96.858 <0.0000000000000002 ***
category_id     13700.4096    8323.9441   1.646              0.0998 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9599000 on 23293 degrees of freedom
Multiple R-squared:  0.7435,    Adjusted R-squared:  0.7435
F-statistic: 1.688e+04 on 4 and 23293 DF,  p-value: < 0.00000000000000022

>
> ### Generate predictions
> gb_videos.lm.pred <- predict(gb_videos.lm, valid.df)
> some.residuals <- valid.df$views[1:20000] - gb_videos.lm.pred[1:20000]
> #correlation
> cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
                  views      likes   dislikes comment_count category_id
views          1.0000000  0.7978517  0.4097205    0.48631582 -0.17062553
likes          0.7978517  1.0000000  0.4525944    0.76419579 -0.18508282
dislikes       0.4097205  0.4525944  1.0000000    0.74968578 -0.03231380
comment_count  0.4863158  0.7641958  0.7496858    1.00000000 -0.07053606
category_id   -0.1706255 -0.1850828 -0.0323138   -0.07053606  1.00000000
> residuals_List<-data.frame("Predicted" = gb_videos.lm.pred[1:20000], "Actual"
= valid.df$views[1:20000],
+                            "Residual" = some.residuals)
> residuals_List
           Predicted    Actual         Residual
1       2408797.22988   2155048    -253749.22988
2       1236307.83660   8411639    7175331.16340
3      18676385.70274  21136674    2460288.29726
4        863797.21830   1383357     519559.78170
```

Figure 4.8 Showing the Regression and Prediction result of GB Videos

From the regression result, we can see that the adjusted R-square value is 74.35%. The model is able to explain 74.35% of the relation between the dependent and independent variables here, which is views with rest of the independent variable.

4.2.2.3 Accuracy Measure:

```
### Accuracy measures

accuracy(gb_videos.lm.pred, valid.df$views)
```

Table 4.9 showing R code for Accuracy Measure

```
> accuracy(gb_videos.lm.pred, valid.df$views)
                 ME    RMSE     MAE      MPE     MAPE
Test set -38702.74 9926423 3173982 22.78496 133.5705
```

Figure 4.9 showing Result for Accuracy Measure

4.2.2.4 Exhaustive Search Method

To find the best predictors, we have deployed the exhaustive search method.

```
### Exhaustive Search
search <- regsubsets(views ~ likes+dislikes+comment_count+category_id, data =
 train.df, nbest = 1, nvmax = dim(train.df)[2],
                      method = "exhaustive")
sum <- summary(search)

# show models
sum$which
# show metrics
sum$rsq
sum$adjr2
sum$cp
```

Table 4.10 showing R Code for exhaustive search

From the output below, we can see that the all the predictors are required. Hence we are not removing any predictors from the regression.

```
> sum$which
  (Intercept) likes dislikes comment_count category_id
1        TRUE  TRUE    FALSE         FALSE       FALSE
2        TRUE  TRUE    FALSE          TRUE       FALSE
3        TRUE  TRUE     TRUE          TRUE       FALSE
4        TRUE  TRUE     TRUE          TRUE        TRUE
> # show metrics
> sum$rsq
[1] 0.6365673 0.6731711 0.7435095 0.7435393
> sum$adjr2
[1] 0.6365517 0.6731430 0.7434764 0.7434952
> sum$cp
[1] 9714.707694 6392.178025    5.708999    5.000000
>
> #dropping two predictors
>
> gb_videos.lm_new <- lm(views ~ likes+dislikes, data = train.df)
> gb_videos.lm_new

Call:
lm(formula = views ~ likes + dislikes, data = train.df)

Coefficients:
(Intercept)        likes     dislikes
   76914.87        42.14        25.18

> options(scipen = 999)
> summary(gb_videos.lm_new)

Call:
lm(formula = views ~ likes + dislikes, data = train.df)

Residuals:
       Min         1Q     Median         3Q        Max
-138039408    -930403    -135463     161612  301921575

Coefficients:
             Estimate Std. Error t value            Pr(>|t|)
(Intercept) 76914.8735 79942.6574   0.962               0.336
likes          42.1395     0.2414 174.591 <0.0000000000000002 ***
dislikes       25.1796     1.8167  13.860 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11380000 on 23295 degrees of freedom
Multiple R-squared:  0.6395,    Adjusted R-squared:  0.6395
F-statistic: 2.067e+04 on 2 and 23295 DF,  p-value: < 0.00000000000000022

>
> gb_videos.lm.new.pred <- predict(gb_videos.lm_new, valid.df, type = "response"
)
>
> accuracy(gb_videos.lm.new.pred, valid.df$views)
                 ME     RMSE      MAE       MPE     MAPE
Test set -10369.49 11520050 3240230 -89.20698 113.6514
```

Figure 4.10 Showing result of the exhaustive search and removing predictor after analyzing the search result

4.2.2.5 *Stepwise Regression*

```
### Stepwise Regression
gb_videos.lm.stepwise <- step(gb_videos.lm, direction = "both")
summary(gb_videos.lm.stepwise)  # Which variables were dropped/added?
gb_videos.lm.stepwise.pred <- predict(gb_videos.lm.stepwise, valid.df)
accuracy(gb_videos.lm.stepwise.pred, valid.df$views)
```

Table 4.11 showing R code for Stepwise Regression

```
> gb_videos.lm.stepwise <- step(gb_videos.lm, direction = "both")
Start:  AIC=749134.8
views ~ likes + dislikes + comment_count + category_id

                Df           Sum of Sq                  RSS    AIC
<none>                                  2146055412538300928 749135
- category_id    1     249588344263680 2146305000882564608 749136
- dislikes       1  588806290392636416 2734861702930937344 754781
- comment_count  1  864339567777831936 3010394980316132864 757018
- likes          1 4042597282626928128 6188652695165229056 773807
> summary(gb_videos.lm.stepwise)  # Which variables were dropped/added?

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-99759891   -962780     37289    494328 250381010

Coefficients:
                Estimate   Std. Error t value            Pr(>|t|)
(Intercept)   -336618.7289  159675.6266  -2.108              0.0350 *
likes              62.7660       0.2996 209.470 <0.0000000000000002 ***
dislikes          172.1121       2.1529  79.943 <0.0000000000000002 ***
comment_count    -284.3379       2.9356 -96.858 <0.0000000000000002 ***
category_id     13700.4096    8323.9441   1.646              0.0998 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9599000 on 23293 degrees of freedom
Multiple R-squared:  0.7435,    Adjusted R-squared:  0.7435
F-statistic: 1.688e+04 on 4 and 23293 DF,  p-value: < 0.00000000000000022

> gb_videos.lm.stepwise.pred <- predict(gb_videos.lm.stepwise, valid.df)
> accuracy(gb_videos.lm.stepwise.pred, valid.df$views)
                ME     RMSE      MAE      MPE     MAPE
Test set -38702.74 9926423 3173982 22.78496 133.5705
```

Figure 4.11 showing Stepwise Regression result

The adjusted R square value is same as of the initial model and exhaustive search method which is 74.35%. Thus, the model will be able to explain for almost 75% of the relation between the views and likes, dislikes, comment count and category id.

4.2.2.6 *Lift and Decile Chart*

```
### Generate Lift Chats
gain <- gains(gb_videos.lm.pred, valid.df$views)
gain
### cumulative lift chart
options(scipen=999)
### Plot graph
plot(c(0,gain$cume.pct.of.total*sum(valid.df$views))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative Views", main="Lift Chart",
     col = "blue1", type="l")
### baseline
lines(c(0,sum(valid.df$views))~c(0,dim(valid.df)[1]), col="brown2", lty=2)


### Decile-wise lift chart
barplot(gain$mean.resp/mean(valid.df$views), names.arg = gain$depth,
        xlab = "Percentile", ylab = "Mean Response", main = "Decile-
wise lift chart",
        col = "coral1")
```

Table 4.12 showing R Code for Lift and Decile chart

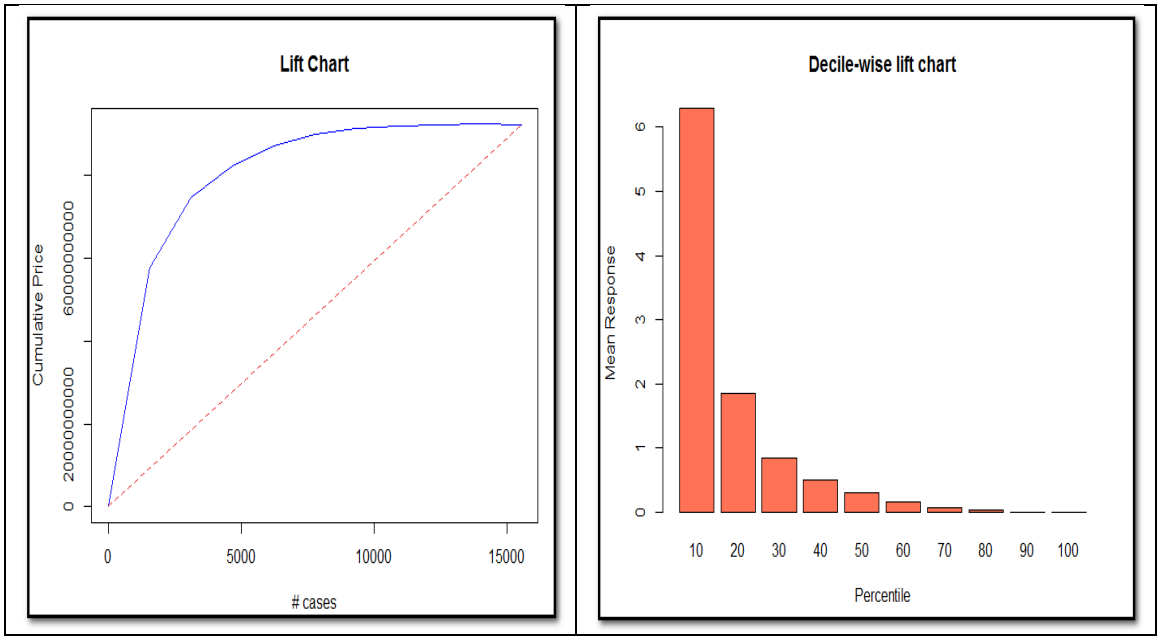


Figure 4.12 showing Lift chart and Decile chart for GB Videos

As can be seen from the lift chart above for UK dataset, the regression model works much better than random prediction since the area between the blue line (response after predictive model) and the red line (response before predictive model) is more.

And the decile chart shows that the predictive model is 6 times better than the random prediction by using just 10% of data.

4.2.3 Linear Regression in Canada Dataset

We have considered View as the dependent variable and likes, dislikes, comment count and category ID as an independent variable to create the model.

4.2.3.1 Data Partition:

```
#linear regression for ca_videos

### Partitioning Data
set.seed(2)  # set seed for reproducing the partition
train.index <- createDataPartition(ca_videos.df$views, p=0.6 , list=FALSE)
train.df <- ca_videos.df[train.index,]
nrow(train.df)
valid.df <- ca_videos.df[-train.index, ]
nrow(valid.df)
```

Table 4.13 showing R code for Data partition of Canada Videos

```
> train.df <- ca_videos.df[train.index,]
> nrow(train.df)
[1] 24487
> valid.df <- ca_videos.df[-train.index, ]
> nrow(valid.df)
[1] 16320
```

Figure 4.13 showing number of Training and Validation records considered for the model.

4.2.3.2 Running Regression and Prediction

```
### Run regression
ca_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = train.df)
ca_videos.lm
options(scipen = 999)
summary(ca_videos.lm)

### Generate predictions
ca_videos.lm.pred <- predict(ca_videos.lm, valid.df)
some.residuals <- valid.df$views[1:20000] - ca_videos.lm.pred[1:20000]
#correlation
cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
```

```
12. residuals_List<-
    data.frame("Predicted" = ca_videos.lm.pred[1:20000], "Actual" = valid.df$view
    s[1:20000],
13.                              "Residual" = some.residuals)
```

Table 4.14 showing R code for Linear Regression and prediction on Canada Video training and Validation dataset.

```
> ca_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = tr
ain.df)
> ca_videos.lm

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Coefficients:
  (Intercept)         likes      dislikes  comment_count    category_id
    462646.88          24.28          58.13         -46.59       -6889.97

> options(scipen = 999)
> summary(ca_videos.lm)

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-27232270   -310335   -194147     53591  50932081

Coefficients:
                Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   462646.8773  38276.5280  12.087 < 0.0000000000000002 ***
likes             24.2828      0.1698 143.006 < 0.0000000000000002 ***
dislikes          58.1322      0.8090  71.856 < 0.0000000000000002 ***
comment_count    -46.5937      1.1944 -39.009 < 0.0000000000000002 ***
category_id    -6889.9678   1725.9322  -3.992            0.0000657 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1804000 on 24482 degrees of freedom
Multiple R-squared:  0.741,     Adjusted R-squared:  0.741
F-statistic: 1.751e+04 on 4 and 24482 DF,  p-value: < 0.00000000000000022

>
> ### Generate predictions
> ca_videos.lm.pred <- predict(ca_videos.lm, valid.df)
> some.residuals <- valid.df$views[1:20000] - ca_videos.lm.pred[1:20000]
> #correlation
> cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
                  views      likes    dislikes comment_count category_id
views          1.0000000  0.8282504  0.54703659    0.70555679 -0.13698243
likes          0.8282504  1.0000000  0.44933392    0.85828919 -0.14425744
dislikes       0.5470366  0.4493339  1.00000000    0.62009005 -0.03459931
comment_count  0.7055568  0.8582892  0.62009005    1.00000000 -0.07605769
category_id   -0.1369824 -0.1442574 -0.03459931   -0.07605769  1.00000000
> residuals_List<-data.frame("Predicted" = ca_videos.lm.pred[1:20000], "Actual"
= valid.df$views[1:20000],
+                            "Residual" = some.residuals)
```

Figure 4.14 Showing the Regression and Prediction result of Canada Videos

The adjusted R square value is 74.1% from the regression output. So, the model will explain 74 % of the relation between the dependent variables and independent variables.

4.2.3.3 *Accuracy Measure:*

```
### Accuracy measures

accuracy(ca_videos.lm.pred, valid.df$views)
```

Table 4.15 showing R code for Accuracy Measure

```
> ### Accuracy measures
> accuracy(ca_videos.lm.pred, valid.df$views)
                 ME     RMSE      MAE       MPE     MAPE
Test set -29988.56 1561610 618270.5 -279.2691 301.2615
>
```

Figure 4.15 showing result for Accuracy Measure

4.2.3.4 *Exhaustive Search Method*

```
### Exhaustive Search
search <- regsubsets(views ~ likes+dislikes+comment_count+category_id, data =
 train.df, nbest = 1, nvmax = dim(train.df)[2],
                     method = "exhaustive")
sum <- summary(search)

# show models
sum$which
# show metrics
sum$rsq
sum$adjr2
sum$cp
```

Table 4.16 showing R Code for exhaustive search

```
> # show models
> sum$which
  (Intercept) likes dislikes comment_count category_id
1        TRUE  TRUE    FALSE         FALSE       FALSE
2        TRUE  TRUE     TRUE         FALSE       FALSE
3        TRUE  TRUE     TRUE          TRUE       FALSE
4        TRUE  TRUE     TRUE          TRUE        TRUE
> # show metrics
> sum$rsq
[1] 0.6859987 0.7243166 0.7408410 0.7410096
> sum$adjr2
[1] 0.6859858 0.7242940 0.7408093 0.7409673
> sum$cp
[1] 5199.10827 1578.97095   18.93628    5.00000
>
```

Figure 4.16 Showing result of the exhaustive search and removing predictor after analyzing the search result

From the results of the exhaustive search method, we are not removing any predictors from the model.

4.2.3.5 Stepwise Regression

```
1. ### Stepwise Regression
2. ca_videos.lm.stepwise <- step(ca_videos.lm, direction = "both")
3. summary(ca_videos.lm.stepwise)  # Which variables were dropped/added?
4. ca_videos.lm.stepwise.pred <- predict(ca_videos.lm.stepwise, valid.df)
5. accuracy(ca_videos.lm.stepwise.pred, valid.df$views)
```

Table 4.17 showing R code for Stepwise Regression

```
> ### Stepwise Regression
> ca_videos.lm.stepwise <- step(ca_videos.lm, direction = "both")
Start:  AIC=705496.7
views ~ likes + dislikes + comment_count + category_id

               Df         Sum of Sq                 RSS    AIC
<none>                                 79661178823706976 705497
- category_id    1    51854537820048  79713033361527024 705511
- comment_count  1  4951502850019264  84612681673726240 706971
- dislikes       1 16800725226714080  96461904050421056 710181
- likes          1 66543942208329472 146205121032036448 720364
> summary(ca_videos.lm.stepwise)  # Which variables were dropped/added?

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-27232270   -310335   -194147     53591  50932081

Coefficients:
                Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   462646.8773  38276.5280  12.087 < 0.0000000000000002 ***
likes             24.2828      0.1698 143.006 < 0.0000000000000002 ***
dislikes          58.1322      0.8090  71.856 < 0.0000000000000002 ***
comment_count    -46.5937      1.1944 -39.009 < 0.0000000000000002 ***
category_id    -6889.9678   1725.9322  -3.992            0.0000657 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1804000 on 24482 degrees of freedom
Multiple R-squared:  0.741,     Adjusted R-squared:  0.741
F-statistic: 1.751e+04 on 4 and 24482 DF,  p-value: < 0.00000000000000022

> ca_videos.lm.stepwise.pred <- predict(ca_videos.lm.stepwise, valid.df)
> accuracy(ca_videos.lm.stepwise.pred, valid.df$views)
                 ME    RMSE      MAE       MPE     MAPE
Test set -29988.56 1561610 618270.5 -279.2691 301.2615
```

Figure 4.17 showing Stepwise Regression result

The adjusted R2 value is as same as the base model as there is no change in the number of predictors. Hence, it is evident that the model explains 74.1% of the changes with respect to the dependent variables.

4.2.3.6 Lift and Decile Chart

```
1. ### Generate Lift Chats
2. gain <- gains(ca_videos.lm.pred, valid.df$views)
3. gain
4. ### cumulative lift chart
```

```
options(scipen=999)
### Plot graph
plot(c(0,gain$cume.pct.of.total*sum(valid.df$views))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative Views", main="Lift Chart",
     col = "blue1", type="l")
### baseline
lines(c(0,sum(valid.df$views))~c(0,dim(valid.df)[1]), col="brown2", lty=2)


### Decile-wise lift chart
barplot(gain$mean.resp/mean(valid.df$views), names.arg = gain$depth,
        xlab = "Percentile", ylab = "Mean Response", main = "Decile-
wise lift chart",
        col = "coral1")
```

Table 4.18 showing R Code for Lift and Decile chart

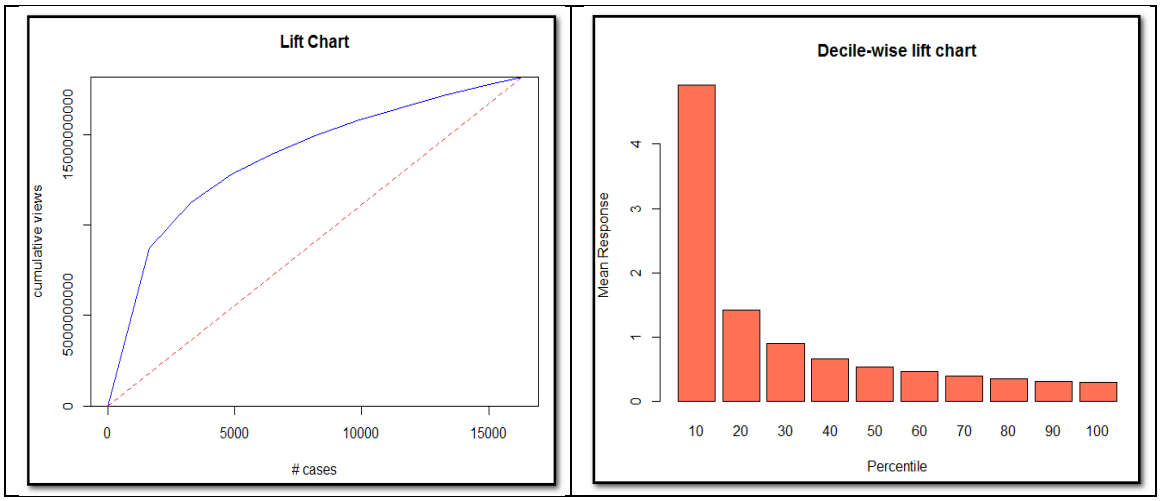


Figure 4.18 showing Lift chart and Decile chart for Canada Videos

As can be seen from the lift chart above for Canada dataset, the regression model works much better than random prediction since the area between the blue line(response after predictive model) and the red line(response before predictive model) is more.

And the decile chart shows that the predictive model is 5 times better than the random prediction by using just 10% of data.

4.2.4 Linear Regression in France Dataset

We have considered View as the dependent variable and likes, dislikes, comment count and category ID as an independent variable to create the model.

4.2.4.1 *Data Partition:*

```
#linear regression for fr_videos

### Partitioning Data
set.seed(2)  # set seed for reproducing the partition
train.infrx <- createDataPartition(fr_videos.df$views, p=0.6 , list=FALSE)
train.df <- fr_videos.df[train.infrx,]
nrow(train.df)
valid.df <- fr_videos.df[-train.infrx, ]
nrow(valid.df)
```

Table 4.19 showing R code for Data partition of France Videos

```
> set.seed(2)  # set seed for reproducing the partition
> train.infrx <- createDataPartition(fr_videos.df$views, p=0.6 , list=FALSE)
> train.df <- fr_videos.df[train.infrx,]
> nrow(train.df)
[1] 24368
> valid.df <- fr_videos.df[-train.infrx, ]
> nrow(valid.df)
[1] 16242
>
```

Figure 4.19 showing number of Training and Validation records considered for the model.

4.2.4.2 *Running Regression and Prediction*

```
### Run regression
fr_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = train.df)
fr_videos.lm
options(scipen = 999)
summary(fr_videos.lm)

### Generate predictions
fr_videos.lm.pred <- predict(fr_videos.lm, valid.df)
some.residuals <- valid.df$views[1:20000] - fr_videos.lm.pred[1:20000]
#correlation
cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
residuals_List<-
data.frame("Predicted" = fr_videos.lm.pred[1:20000], "Actual" = valid.df$views[1:20000],
                           "Residual" = some.residuals)
```

Table 4.20 showing R code for Linear Regression and prediction on France Video training and Validation dataset.

```
> ### Run regression
> fr_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = tr
ain.df)
> fr_videos.lm

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Coefficients:
  (Intercept)          likes       dislikes  comment_count    category_id
    172249.62          17.55          60.63         -32.86       -2324.50

> options(scipen = 999)
> summary(fr_videos.lm)

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-16253551   -131479   -108238    -32889  25915554

Coefficients:
                Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   172249.6157  16314.5680  10.558 < 0.0000000000000002 ***
likes             17.5524      0.1268 138.403 < 0.0000000000000002 ***
dislikes          60.6284      0.7094  85.464 < 0.0000000000000002 ***
comment_count    -32.8634      0.8952 -36.710 < 0.0000000000000002 ***
category_id    -2324.5003    759.9097  -3.059              0.00222 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 826100 on 24363 degrees of freedom
Multiple R-squared:  0.7659,    Adjusted R-squared:  0.7658
F-statistic: 1.992e+04 on 4 and 24363 DF,  p-value: < 0.00000000000000022

>
> ### Generate predictions
> fr_videos.lm.pred <- predict(fr_videos.lm, valid.df)
> some.residuals <- valid.df$views[1:20000] - fr_videos.lm.pred[1:20000]
> #correlation
> cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
                  views       likes    dislikes comment_count category_id
views          1.00000000  0.83246022  0.58146056    0.74885808 -0.08583404
likes          0.83246022  1.00000000  0.43416499    0.87008369 -0.08933223
dislikes       0.58146056  0.43416499  1.00000000    0.63732217 -0.02039429
comment_count  0.74885808  0.87008369  0.63732217    1.00000000 -0.04731126
category_id   -0.08583404 -0.08933223 -0.02039429   -0.04731126  1.00000000
> residuals_List<-data.frame("Predicted" = fr_videos.lm.pred[1:20000], "Actual"
= valid.df$views[1:20000],
+                            "Residual" = some.residuals)
```

Figure 4.20 Showing the Regression and Prediction result of France Videos

The adjusted R square value is 76.58 % and the model is one of the best when compared with the other sites in predicted the number of views. The accuracy of the model is captured in terms of the RMSE value and it is calculated in the below section.

4.2.4.3 Accuracy Measure:

```
10. ### Accuracy measures
11.
```

```
12. accuracy(fr_videos.lm.pred, valid.df$views)
```

Table 4.21 showing R code for Accuracy Measure

```
> accuracy(fr_videos.lm.pred, valid.df$views)
                ME      RMSE       MAE       MPE      MAPE
Test set 4510.093 988403.8 269423.3 -704.863 723.5691
```

Figure 4.21 showing result for Accuracy Measure

4.2.4.4 Exhaustive Search Method

```
12. ### Exhaustive Search
13. search <- regsubsets(views ~ likes+dislikes+comment_count+category_id, data =
     train.df, nbest = 1, nvmax = dim(train.df)[2],
14.                      method = "exhaustive")
15. sum <- summary(search)
16.
17. # show models
18. sum$which
19. # show metrics
20. sum$rsq
21. sum$adjr2
22. sum$cp
```

Table 4.22 showing R Code for exhaustive search

From the result of the exhaustive search method, we are not removing any predictors in this model as the adjusted R2 value stays increasing and did not decrease.

```
> # show mofrls
> sum$which
  (Intercept) likes dislikes comment_count category_id
1        TRUE  TRUE    FALSE         FALSE       FALSE
2        TRUE  TRUE     TRUE         FALSE       FALSE
3        TRUE  TRUE     TRUE          TRUE       FALSE
4        TRUE  TRUE     TRUE          TRUE        TRUE
> # show metrics
> sum$rsq
[1] 0.6929900 0.7526518 0.7657856 0.7658755
> sum$adjr2
[1] 0.6929774 0.7526315 0.7657568 0.7658371
> sum$cp
[1] 7583.46794 1377.05755   12.35697    5.00000
```

Figure 4.22 Showing result of exhaustive search and removing predictor after analysing the search result

4.2.4.5 *Stepwise Regression*

```
### Stepwise Regression
fr_videos.lm.stepwise <- step(fr_videos.lm, direction = "both")
summary(fr_videos.lm.stepwise)  # Which variables were dropped/adfrd?
fr_videos.lm.stepwise.pred <- predict(fr_videos.lm.stepwise, valid.df)
accuracy(fr_videos.lm.stepwise.pred, valid.df$views)
```

Table 4.23 showing R code for Stepwise Regression

```
> ### Stepwise Regression
> fr_videos.lm.stepwise <- step(fr_videos.lm, direction = "both")
Start:  AIC=664007.1
views ~ likes + dislikes + comment_count + category_id

               Df          Sum of Sq                 RSS    AIC
<none>                                 16626202886380902 664007
- category_id    1     6385538514006 16632588424894908 664014
- comment_count  1   919680701022828 17545883587403730 665317
- dislikes       1  4984584116956846 21610787003337748 670395
- likes          1 13072409642087306 29698612528468208 678141
> summary(fr_videos.lm.stepwise)  # Which variables were dropped/adfrd?

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-16253551   -131479   -108238    -32889  25915554

Coefficients:
                Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   172249.6157  16314.5680  10.558 < 0.0000000000000002 ***
likes             17.5524      0.1268 138.403 < 0.0000000000000002 ***
dislikes          60.6284      0.7094  85.464 < 0.0000000000000002 ***
comment_count    -32.8634      0.8952 -36.710 < 0.0000000000000002 ***
category_id    -2324.5003    759.9097  -3.059              0.00222 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 826100 on 24363 degrees of freedom
Multiple R-squared:  0.7659,    Adjusted R-squared:  0.7658
F-statistic: 1.992e+04 on 4 and 24363 DF,  p-value: < 0.00000000000000022

> fr_videos.lm.stepwise.pred <- predict(fr_videos.lm.stepwise, valid.df)
> accuracy(fr_videos.lm.stepwise.pred, valid.df$views)
                ME     RMSE      MAE       MPE     MAPE
Test set 4510.093 988403.8 269423.3 -704.863 723.5691
```

Figure 4.23 showing Stepwise Regression result

4.2.4.6 Lift and Decile Chart

```
### Generate Lift Chats
gain <- gains(fr_videos.lm.pred, valid.df$views)
gain
### cumulative lift chart
options(scipen=999)
### Plot graph
plot(c(0,gain$cume.pct.of.total*sum(valid.df$views))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative Views", main="Lift Chart",
     col = "blue1", type="l")
### baseline
lines(c(0,sum(valid.df$views))~c(0,dim(valid.df)[1]), col="brown2", lty=2)


### France-wise lift chart
barplot(gain$mean.resp/mean(valid.df$views), names.arg = gain$frpth,
        xlab = "Percentile", ylab = "Mean Response", main = "France-
wise lift chart",
        col = "coral1")
```

Table 4.24 showing R Code for Lift and Decile chart

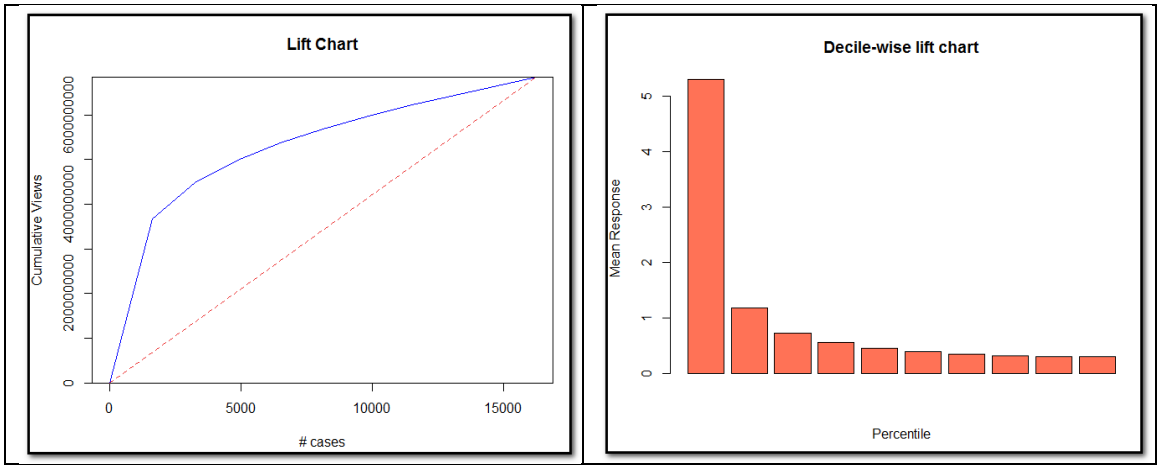


Figure 4.24 showing Lift chart and Decile chart for France Videos

As can be seen from the lift chart above for France dataset, the regression model works much better than random prediction since the area between the blue line(response after predictive model) and the red line(response before predictive model) is more.

And the decile chart shows that the predictive model is 5 times better than the random prediction by using just 10% of data.

4.2.5 Linear Regression in Germany Dataset

We have considered View as the dependent variable and likes, dislikes, comment count and category ID as an independent variable to create the model.

4.2.5.1 Data Partition:

```
#linear regression for de_videos

### Partitioning Data
set.seed(2)  # set seed for reproducing the partition
train.index <- createDataPartition(de_videos.df$views, p=0.6 , list=FALSE)
train.df <- de_videos.df[train.index,]
nrow(train.df)
valid.df <- de_videos.df[-train.index, ]
nrow(valid.df)
```

Table 4.25 showing R code for Data partition of Germany Videos

```
> set.seed(2)  # set seed for reproducing the partition
> train.index <- createDataPartition(de_videos.df$views, p=0.6 , list=FALSE)
> train.df <- de_videos.df[train.index,]
> nrow(train.df)
[1] 24352
> valid.df <- de_videos.df[-train.index, ]
> nrow(valid.df)
[1] 16232
```

Figure 4.25 showing number of Training and Validation records considered for the model.

4.2.5.2 Running Regression and Prediction

```
### Run regression
de_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = train.df)
de_videos.lm
options(scipen = 999)
summary(de_videos.lm)
### Generate predictions
de_videos.lm.pred <- predict(de_videos.lm, valid.df)
some.residuals <- valid.df$views[1:20000] - de_videos.lm.pred[1:20000]
#correlation
cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
residuals_List<-
data.frame("Predicted" = de_videos.lm.pred[1:20000], "Actual" = valid.df$views[1:20000],
                           "Residual" = some.residuals)
```

Table 4.26 showing R code for Linear Regression and prediction on Germany Video training and Validation dataset.

```
> ### Run regression
> de_videos.lm <- lm(views ~ likes+dislikes+comment_count+category_id, data = tr
ain.df)
> de_videos.lm

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Coefficients:
  (Intercept)          likes       dislikes  comment_count    category_id
    194414.41          21.51          55.29         -37.30       -1122.16

> options(scipen = 999)
> summary(de_videos.lm)

Call:
lm(formula = views ~ likes + dislikes + comment_count + category_id,
    data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-22500355   -174538   -149280     -1372  48466005

Coefficients:
                Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   194414.4125  25176.7000   7.722   0.0000000000000119 ***
likes             21.5134      0.1606 133.934 < 0.0000000000000002 ***
dislikes          55.2890      0.6692  82.618 < 0.0000000000000002 ***
comment_count    -37.3046      1.1480 -32.496 < 0.0000000000000002 ***
category_id    -1122.1649   1145.7829  -0.979                0.327
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1239000 on 24347 degrees of freedom
Multiple R-squared:  0.7689,    Adjusted R-squared:  0.7689
F-statistic: 2.025e+04 on 4 and 24347 DF,  p-value: < 0.00000000000000022

>
> ### Generate predictions
> de_videos.lm.pred <- predict(de_videos.lm, valid.df)
> some.residuals <- valid.df$views[1:20000] - de_videos.lm.pred[1:20000]
> #correlation
> cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
                   views      likes    dislikes comment_count category_id
views          1.00000000  0.8374303  0.58354418    0.76033016 -0.08680761
likes          0.83743031  1.0000000  0.43963746    0.87765626 -0.10197275
dislikes       0.58354418  0.4396375  1.00000000    0.62955995 -0.01509476
comment_count  0.76033016  0.8776563  0.62955995    1.00000000 -0.05552353
category_id   -0.08680761 -0.1019727 -0.01509476   -0.05552353  1.00000000
> residuals_List<-data.frame("Predicted" = de_videos.lm.pred[1:20000], "Actual"
= valid.df$views[1:20000],
+                           "Residual" = some.residuals)
```

Figure 4.26 Showing the Regression and Prediction result of Germany Videos

From the summary of the linear regression, the adjusted R square value turns out to be around 76.89%. This means that the model is good and can explain up to 77% of the

variation of the dependent variable with respect to the other independent variable. The accuracy of the model is measured in terms of RMSE which is calculated in the below section.

4.2.5.3 Accuracy Measure:

```
13. ### Accuracy measures
14.
15. accuracy(de_videos.lm.pred, valid.df$views)
```

Table 4.27 showing the R Code for accuracy measure

```
> accuracy(de_videos.lm.pred, valid.df$views)
                   ME     RMSE      MAE       MPE     MAPE
Test set -12512.51 1135854 372570.7 -683.805 705.0507
```

Figure 4.27 Showing the accuracy measure result

4.2.5.4 Exhaustive Search Method

```
23. ### Exhaustive Search
24. search <- regsubsets(views ~ likes+dislikes+comment_count+category_id, data =
     train.df, nbest = 1, nvmax = dim(train.df)[2],
25.                      method = "exhaustive")
26. sum <- summary(search)
27.
28. # show models
29. sum$which
30. # show metrics
31. sum$rsq
32. sum$adjr2
33. sum$cp
```

Table 4.28 showing R Code for exhaustive search

From the result of the exhaustive search, we are not removing any predictors. There is a very small reduction in the adjusted R square and hence we decided not to drop any predictors.

```
> # show models
> sum$which
  (Intercept) likes dislikes comment_count category_id
1        TRUE  TRUE    FALSE          FALSE       FALSE
2        TRUE  TRUE     TRUE          FALSE       FALSE
3        TRUE  TRUE     TRUE           TRUE       FALSE
4        TRUE  TRUE     TRUE           TRUE        TRUE
> # show metrics
> sum$rsq
[1] 0.7012895 0.7587914 0.7688943 0.7689034
> sum$adjr2
[1] 0.7012773 0.7587716 0.7688658 0.7688654
> sum$cp
[1] 7122.403780 1066.336374    3.959199    5.000000
>
```

Figure 4.28 Showing result of exhaustive search and removing predictor after analysing the search result

4.2.5.5 Stepwise Regression

```
11. ### Stepwise Regression
12. de_videos.lm.stepwise <- step(de_videos.lm, direction = "both")
13. summary(de_videos.lm.stepwise)  # Which variables were dropped/added?
14. de_videos.lm.stepwise.pred <- predict(de_videos.lm.stepwise, valid.df)
15. accuracy(de_vdeos.lm.stepwise.pred, valid.df$views)
```

Table 4.29 showing R code for Stepwise Regression

```
> ### Stepwise Regression
> de_videos.lm.stepwise <- step(de_videos.lm, direction = "both")
Start:  AIC=683310.4
views ~ likes + dislikes + comment_count + category_id

                Df          Sum of Sq                 RSS     AIC
- category_id    1     1472321611216 37372884933342760 683309
<none>                               37371412611731544 683310
- comment_count  1  1620930884951376 38992343496682920 684342
- dislikes       1 10477156217333008 47848568829064552 689327
- likes          1 27534275254171272 64905687865902816 696751

Step:  AIC=683309.3
views ~ likes + dislikes + comment_count

                Df          Sum of Sq                 RSS     AIC
<none>                               37372884933342752 683309
+ category_id    1     1472321611208 37371412611731544 683310
- comment_count  1  1633765088329440 39006650021672192 684349
- dislikes       1 10480060789799840 47852945723142592 689327
- likes          1 27907438121949024 65280323055291776 696890
> summary(de_videos.lm.stepwise)  # Which variables were dropped/added?

Call:
lm(formula = views ~ likes + dislikes + comment_count, data = train.df)

Residuals:
     Min       1Q   Median       3Q      Max
-22505441   -173474   -150229      -468  48464659

Coefficients:
              Estimate  Std. Error t value             Pr(>|t|)
(Intercept)  171072.5184   8114.5672   21.08 <0.0000000000000002 ***
likes            21.5305      0.1597  134.84 <0.0000000000000002 ***
dislikes         55.2946      0.6692   82.63 <0.0000000000000002 ***
comment_count   -37.3762      1.1456  -32.62 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1239000 on 24348 degrees of freedom
Multiple R-squared:  0.7689,    Adjusted R-squared:  0.7689
F-statistic: 2.7e+04 on 3 and 24348 DF,  p-value: < 0.00000000000000022

> de_videos.lm.stepwise.pred <- predict(de_videos.lm.stepwise, valid.df)
> accuracy(de_videos.lm.stepwise.pred, valid.df$views)
               ME     RMSE      MAE       MPE     MAPE
Test set -12558.79 1135948 372811.6 -686.2414 707.4922
>
```

Figure 4.29 showing Stepwise Regression result

The Backward stepwise method confirms us that there is no need in dropping the predictors and all the predictors are significant. Hence the adjusted R square remains the same at 76.89%.

4.2.5.6 Lift and Decile Chart

```
35. ### Generate Lift Chats
36. gain <- gains(de_videos.lm.pred, valid.df$views)
37. gain
38. ### cumulative lift chart
39. options(scipen=999)
40. ### Plot graph
41. plot(c(0,gain$cume.pct.of.total*sum(valid.df$views))~c(0,gain$cume.obs),
42.      xlab="# deses", ylab="Cumulative Views", main="Lift Chart",
43.      col = "blue1", type="l")
44. ### baseline
45. lines(c(0,sum(valid.df$views))~c(0,dim(valid.df)[1]), col="brown2", lty=2)
46.
47.
48. ### Decile-wise lift chart
49. barplot(gain$mean.resp/mean(valid.df$views), names.arg = gain$depth,
50.         xlab = "Percentile", ylab = "Mean Response", main = "Decile-
    wise lift chart",
51.         col = "coral1")
```

Table 4.30 showing R Code for Lift and Decile chart

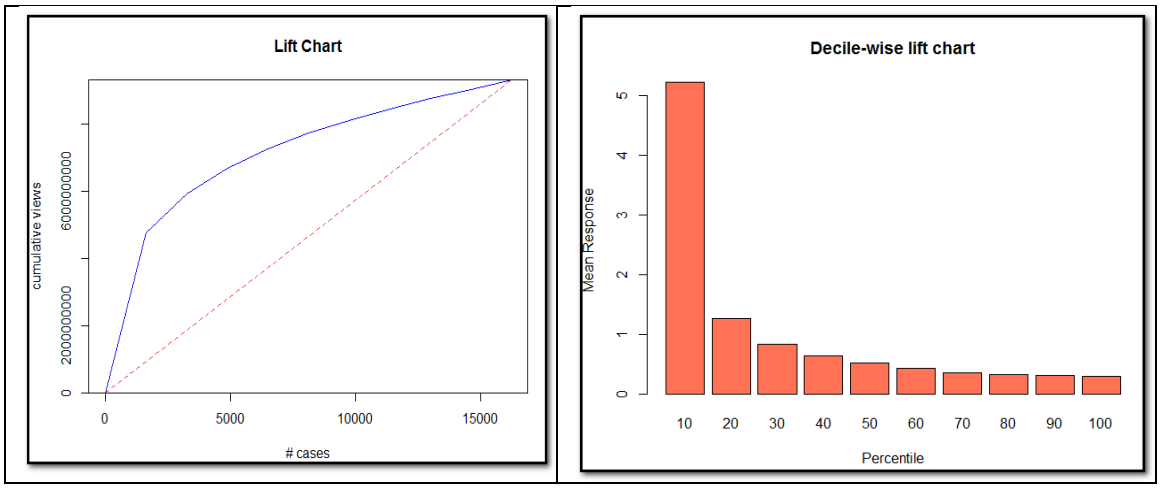


Figure 4.30 showing Lift chart and Decile chart for Germany Videos

As can be seen from the lift chart above for Germany dataset, the regression model works much better than random prediction since the area between the blue line(response after predictive model) and the red line(response before predictive model) is more.

And the decile chart shows that the predictive model is 5 times better than the random prediction by using just 10% of data.

4.2.6 Linear Regression in Complete Dataset

We have considered View as the dependent variable and likes, dislikes, comment count and category ID as an independent variable to create the model. Here we have performed the Linear regression for the overall dataset. This regression model will give us the insight on how views depend on the other parameters considered in the dataset.

4.2.6.1 Data Partition:

```
1. ### Linear Regression for the combined video dataset
2. ### Partitioning Data
3. set.seed(121)  # set seed for reproducing the partition
4. train.index <- createDataPartition(Youtube_videos.df$views, p=0.6 , list=FALS
   E)
5. train.df <- Youtube_videos.df[train.index,]
6. valid.df <- Youtube_videos.df[-train.index, ]
```

Table 4.31 showing R code for Data partition of YouTube Videos

```
> nrow(train.df)
[1] 121068
> nrow(valid.df)
[1] 80708
>
```

Figure 4.31 showing number of Training and Validation records considered for the model.

4.2.6.2 Running Regression and Prediction

```
1. Youtube_videos.lm <- lm(views ~ likes+dislikes+comment_count+Location+categor
   y_id, data = train.df)
2. Youtube_videos.lm
3. options(scipen = 999)
4. summary(Youtube_videos.lm)
5.
6. ### Generate predictions
7. Youtube_videos.lm.pred <- predict(Youtube_videos.lm, valid.df)
8. some.residuals <- valid.df$views[1:20000] - Youtube_videos.lm.pred[1:20000]
9. #correlation
10. cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
11. residuals_List<-
    data.frame("Predicted" = Youtube_videos.lm.pred[1:20000], "Actual" = valid.df
    $views[1:20000],
12.                            "Residual" = some.residuals)
```

```
13. residuals_List
```

Table 4.32 showing R code for Linear Regression and prediction on YouTube Video training and Validation dataset.

```
> summary(Youtube_videos.lm)

Call:
lm(formula = views ~ likes + dislikes + comment_count + Location +
    category_id, data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-68460323   -395020    141850    352367 275304751

Coefficients:
                Estimate   Std. Error  t value            Pr(>|t|)
(Intercept)   -380274.8757   54052.2760   -7.035       0.000000000002 ***
likes              49.8708       0.1178  423.309 < 0.0000000000000002 ***
dislikes          120.5799       0.7389  163.186 < 0.0000000000000002 ***
comment_count    -184.8772       0.9825 -188.164 < 0.0000000000000002 ***
LocationDE      13791.3088   46048.3340    0.299                0.765
LocationFR     -67295.9191   46213.4898   -1.456                0.145
LocationGB     884178.4227   47635.0202   18.562 < 0.0000000000000002 ***
LocationUS     -48099.8782   46017.3692   -1.045                0.296
category_id     11476.3457    2055.4547    5.583       0.000000023642 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5089000 on 121059 degrees of freedom
Multiple R-squared:  0.7114,    Adjusted R-squared:  0.7114
F-statistic: 3.73e+04 on 8 and 121059 DF,  p-value: < 0.00000000000000022

>
> ### Generate predictions
> Youtube_videos.lm.pred <- predict(Youtube_videos.lm, valid.df)
> some.residuals <- valid.df$views[1:20000] - Youtube_videos.lm.pred[1:20000]
> #correlation
> cor(train.df[,c("views","likes","dislikes","comment_count","category_id")])
                views      likes    dislikes comment_count category_id
views          1.0000000  0.7866922  0.40289865     0.5001444 -0.15148576
likes          0.7866922  1.0000000  0.43385300     0.7685132 -0.17094313
dislikes       0.4028987  0.4338530  1.00000000     0.7351842 -0.03432573
comment_count  0.5001444  0.7685132  0.73518417     1.0000000 -0.07595820
category_id   -0.1514858 -0.1709431 -0.03432573    -0.0759582  1.00000000
> residuals_List<-data.frame("Predicted" = Youtube_videos.lm.pred[1:20000], "Actual"
.df$views[1:20000],
+                           "Residual" = some.residuals)
```

Figure 4.32 Showing the Regression and Prediction result of YouTube Videos

The adjusted R square value turns out to be 71.14% which is a moderate value for the model. The model will be able to explain 71% of the variation and the remain 29% will not be able to explain by this model due to the standard errors. The Accuracy of the model is calculated in the below section.

4.2.6.3 Accuracy Measure:

```
1. ### Accuracy measures
```

```
2.  accuracy(Youtube_videos.lm.pred, valid.df$views)
```

Table 4.33 showing R code for Accuracy Measure

```
> accuracy(Youtube_videos.lm.pred, valid.df$views)
                    ME     RMSE      MAE       MPE      MAPE
Test set -2285.418 5035303 1327407 346.6919 550.0384
```

Figure 4.33 showing result for Accuracy Measure

4.2.6.4 Exhaustive Search Method

```
1.  ### Exhaustive Search
2.  search <- regsubsets(views ~ likes+dislikes+comment_count+category_id, data =
     train.df, nbest = 1, nvmax = dim(train.df)[2],
3.                       method = "exhaustive")
4.  sum <- summary(search)
5.  # show models
6.  sum$which
7.  # show metrics
8.  sum$rsq
9.  sum$adjr2
10. sum$cp
```

Table 4.34 showing R Code for exhaustive search

```
> ### Exhaustive Search
> search <- regsubsets(views ~ likes+dislikes+comment_count+category_id, data = train.df, nb
est = 1, nvmax = dim(train.df)[2],
+                      method = "exhaustive")
> sum <- summary(search)
> # show models
> sum$which
  (Intercept) likes dislikes comment_count category_id
1        TRUE  TRUE    FALSE         FALSE       FALSE
2        TRUE  TRUE    FALSE          TRUE       FALSE
3        TRUE  TRUE     TRUE          TRUE       FALSE
4        TRUE  TRUE     TRUE          TRUE        TRUE
> # show metrics
> sum$rsq
[1] 0.6188846 0.6455281 0.7100165 0.7100249
> sum$adjr2
[1] 0.6188815 0.6455222 0.7100093 0.7100154
> sum$cp
[1] 38049.587923 26928.093363     6.513192     5.000000
```

Figure 4.34 Showing result of an exhaustive search

4.2.6.5 Stepwise Regression

```
### Stepwise Regression
Youtube_videos.lm.stepwise <- step(Youtube_videos.lm, direction = "both")
summary(Youtube_videos.lm.stepwise)  # Which variables were dropped/added?
Youtube_videos.lm.stepwise.pred <- predict(Youtube_videos.lm.stepwise, valid.df)
accuracy(Youtube_videos.lm.stepwise.pred, valid.df$views)
```

Table 4.35 showing R code for Stepwise Regression

```
> ### Stepwise Regression
> Youtube_videos.lm.stepwise <- step(Youtube_videos.lm, direction = "both")
Start:  AIC=3739223
views ~ likes + dislikes + comment_count + Location + category_id

               Df           Sum of Sq                  RSS     AIC
<none>                                  3135354036934485504 3739223
- category_id   1     807385492151296 3136161422426636800 3739252
- Location      4   14750970303243776 3150105007237729280 3739784
- dislikes      1  689695376707871744 3825049413642357248 3763293
- comment_count 1  916985459959009280 4052339496893494784 3770282
- likes         1 4640923701808728064 7776277738743214080 3849192
> summary(Youtube_videos.lm.stepwise)  # which variables were dropped/added?

Call:
lm(formula = views ~ likes + dislikes + comment_count + Location +
    category_id, data = train.df)

Residuals:
      Min        1Q    Median        3Q       Max
-68460323   -395020    141850    352367 275304751

Coefficients:
                Estimate   Std. Error  t value            Pr(>|t|)
(Intercept)   -380274.8757   54052.2760   -7.035       0.000000000002 ***
likes              49.8708       0.1178  423.309 < 0.0000000000000002 ***
dislikes          120.5799       0.7389  163.186 < 0.0000000000000002 ***
comment_count    -184.8772       0.9825 -188.164 < 0.0000000000000002 ***
LocationDE      13791.3088   46048.3340    0.299                0.765
LocationFR     -67295.9191   46213.4898   -1.456                0.145
LocationGB     884178.4227   47635.0202   18.562 < 0.0000000000000002 ***
LocationUS     -48099.8782   46017.3692   -1.045                0.296
category_id     11476.3457    2055.4547    5.583       0.000000023642 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5089000 on 121059 degrees of freedom
Multiple R-squared:  0.7114,    Adjusted R-squared:  0.7114
F-statistic: 3.73e+04 on 8 and 121059 DF,  p-value: < 0.00000000000000022

> Youtube_videos.lm.stepwise.pred <- predict(Youtube_videos.lm.stepwise, valid.df)
> accuracy(Youtube_videos.lm.stepwise.pred, valid.df$views)
                ME     RMSE      MAE       MPE     MAPE
Test set -2285.418 5035303 1327407 346.6919 550.0384
```

Figure 4.35 showing Stepwise Regression result

4.2.6.6 Lift and Decile Chart

```
### Generate Lift Chats
gain <- gains(Youtube_videos.lm.pred, valid.df$views)
### cumulative lift chart
options(scipen=999)
### Plot graph
plot(c(0,gain$cume.pct.of.total*sum(valid.df$views))~c(0,gain$cume.obs),
     xlab="# cases", ylab="cumulative views", main="Lift Chart",
     col = "blue1", type="l")
### baseline
lines(c(0,sum(valid.df$views))~c(0,dim(valid.df)[1]), col="brown2", lty=2)

### Decile-wise lift chart
barplot(gain$mean.resp/mean(valid.df$views), names.arg = gain$depth,
        xlab = "Percentile", ylab = "Mean Response", main = "Decile-
wise lift chart",
        col = "coral1")
```

Table 4.36 showing R Code for Lift and Decile chart

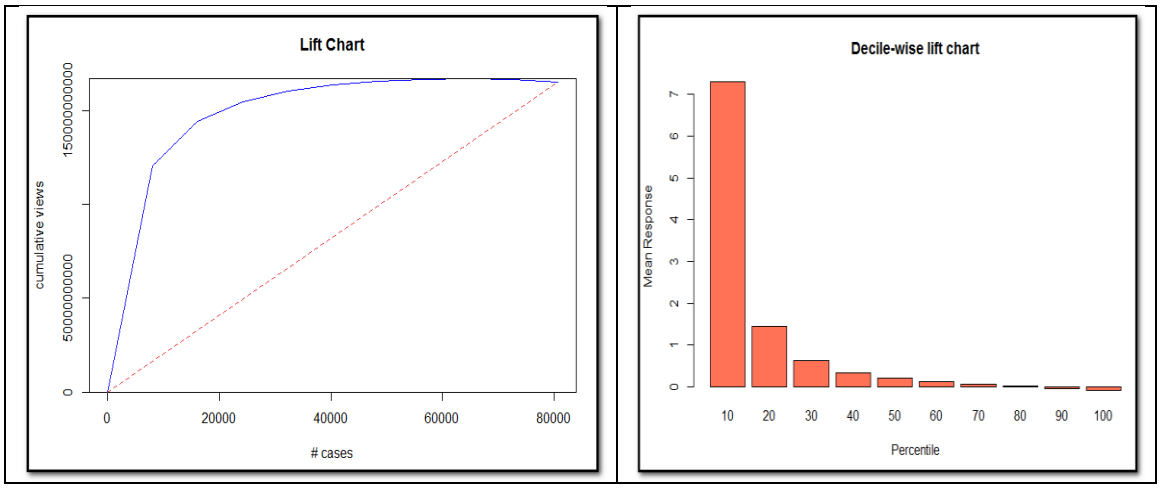


Figure 4.36 showing Lift chart and Decile chart for YouTube Videos

A lift chart and decile lift chart based on fitting a linear a regression model to the all video dataset. The charts are based on the validation data. The model's predictive performance in terms of lift is better than the baseline model since its lift curve is higher than that of the baseline model. The lift and decile charts in the above figure would be useful in the following scenario: choosing the top 10% of the videos that gave the highest predicted views, for example, we would gain 7 times the amount of views, compared to choosing 10% of the videos at random. This can be seen from the decile chart also as shown in the figure. (Data Mining for Business Analytics, 2018)

4.2.7 Conclusions from Linear Regression

The linear regression models are thus created for the different datasets and they are being compared based on their results and adjusted R square value. The stepwise and exhaustive regression is run to find the necessary predictors. Here since the number of variables is very less and all the predictors are statistically significant, we have not done any dimension reduction or predictor removal. The model performance is also evaluated using the Lift and Decile chart and the ranking to get the most number of views are explained successfully.

4.3 Classification Techniques

We have applied three different techniques for creating a classification model to predict the category ID based on comments counts, views, likes and dislikes fields. We have used the confusion matrix to compare the accuracy of the model and have concluded which models give us better accuracy. In the following sections, we have discussed the models used and their accuracy and comparison in detail. In all the cases, we have considered the merged dataset which includes dataset from all the five regions together.

In this section, we have taken the numerical features of the YouTube videos data, that is, a number of views, likes, dislikes and comment count and based on these, we have predicted what category a video belongs to, after splitting the data into train and test sets. This will be done using various machine learning models. The goal here is to compare various classification models and to decide which model fits the best for our data set.

For developing and predict the model, we have divided the dataset into training and validation dataset in the ratio 80:20. We have also normalized data for simplicity and then we have used the normalized dataset for the further model development and prediction.

```
### Partitioning the data into training and validation data

set.seed(123)

training.index <- createDataPartition(Youtube_videos.df$category_id, p = 0.8,
 list = FALSE)

data.train <- Youtube_videos.df[training.index, ]
data.valid <- Youtube_videos.df[-training.index, ]

data.train.x = data.train[,c("views","likes","dislikes","comment_count")]
```

```
data.train.y = data.train[,c("category_id")]

data.valid.x = data.valid[,c("views","likes","dislikes","comment_count")]
data.valid.y = data.valid[,c("category_id")]

norm.values  <- preProcess(data.train.x, method = c("center", "scale"))

data.train.norm <- predict(norm.values, data.train.x)
data.valid.norm <- predict(norm.values, data.valid.x)


data.train.norm = cbind(data.train.norm,data.train.y)
data.valid.norm = cbind(data.valid.norm,data.valid.y)
```

Table 4.37 showing R Code for data partition and normalization for Classification modeling.

4.3.1 Linear Discriminant Analysis

```
### Partitioning the data into training and validation data

set.seed(123)

training.index <- createDataPartition(Youtube_videos.df$category_id, p = 0.8,
 list = FALSE)

data.train <- Youtube_videos.df[training.index, ]
data.valid <- Youtube_videos.df[-training.index, ]

data.train.x = data.train[,c("views","likes","dislikes","comment_count")]
data.train.y = data.train[,c("category_id")]

data.valid.x = data.valid[,c("views","likes","dislikes","comment_count")]
data.valid.y = data.valid[,c("category_id")]

norm.values  <- preProcess(data.train.x, method = c("center", "scale"))

data.train.norm <- predict(norm.values, data.train.x)
data.valid.norm <- predict(norm.values, data.valid.x)


data.train.norm = cbind(data.train.norm,category_id = data.train.y)
data.valid.norm = cbind(data.valid.norm,category_id = data.valid.y)
```

Table 4.38 showing R code for LDA on YouTube dataset

The prior probabilities of the dataset are shown below. Since there are 17 categories, the % of data for each category in the training dataset is as shown below. The group means is calculated for each category id and each variable used in LDA. This also as shown below.

```
> lda1
Call:
lda(category_id ~ views + likes + dislikes + comment_count, data = data.train.norm)

Prior probabilities of groups:
           1            2           10           15           17           19           20
5.503890e-02 1.313136e-02 1.128365e-01 1.338270e-02 6.980708e-02 7.148068e-03 2.940149e-02
          22           23           24           25           26           27           28
1.092257e-01 8.626112e-02 2.865725e-01 7.770919e-02 7.061016e-02 2.903979e-02 3.695416e-02
          29           30           43           44
5.272160e-04 9.808670e-05 2.243733e-03 1.226084e-05

Group means:
         views       likes     dislikes comment_count
1   0.07984225 -0.01742272 -0.031188196   -0.01856361
2  -0.16325970 -0.21920669 -0.088791298   -0.14309067
10  0.60910700  0.75419712  0.186961894    0.37904863
15 -0.13198837 -0.15322643 -0.082019777   -0.08625058
17 -0.05432614 -0.12860778 -0.049019172   -0.09272916
19 -0.15419115 -0.20723909 -0.078606753   -0.12674119
20 -0.05873072 -0.03541716  0.072873302    0.08173633
22 -0.12167685 -0.11759313 -0.028802480   -0.06033583
23 -0.07428774  0.01139239 -0.038049927   -0.02225728
24 -0.04585152 -0.08314772  0.009596655   -0.03290805
25 -0.19400993 -0.23295438 -0.062964251   -0.13008679
26 -0.12227702 -0.10060740 -0.060394571   -0.04418461
27 -0.16271582 -0.13227695 -0.074230418   -0.09602658
28 -0.02917528 -0.08049091 -0.031413338   -0.01610380
29  0.19348527  0.98191199  1.962264364    2.25727101
30 -0.07930507 -0.16661559 -0.070321419   -0.15798948
43 -0.14775982 -0.22236349 -0.066501236   -0.15983443
44 -0.26468234 -0.26663627 -0.104448459   -0.19133185

Coefficients of linear discriminants:
                    LD1         LD2         LD3        LD4
views          0.4807469  1.88104615 -0.71947584 -0.1524822
likes         -2.1629761 -1.40651281  1.31330093  0.4978844
dislikes      -0.3319342 -0.52976675  0.02881026  1.4354952
comment_count  1.2364100  0.07873086 -1.49536551 -1.4492716

Proportion of trace:
   LD1    LD2    LD3    LD4
0.8993 0.0588 0.0364 0.0054
>
> pred <- predict(lda1, data.valid.norm)
> names(pred)
[1] "class"     "posterior" "x"
```

Figure 4.37 showing LDA output of the dataset for classification

There are four coefficients of the linear discriminants in the output. Here the LD is created for each variable considered in the model. The proportion of the trace gives the percentage of separations achieved by each Linear discriminant coefficient. The percentage separation achieved by the first discriminant is about 89.93% percentage and the second by 5.88% and rest two by about 4%. The prediction results in three attributes class, posterior or probabilities, and x. We concentrate on Class to get the category ID prediction.

4.3.1.1 Classification Matrix and Accuracy Measure

```
1. #Checking the model accuracy
2. category_id <- as.factor(data.valid.norm$category_id)
3. class(category_id)
4.
5. print("Confusion Matrix")
6. # Predicted v/s Actual accuracy
```

```
#print(table(pred$class, category_id))  # pred v actual
confusionMatrix(pred$class, category_id)


# Percent accuracy of model
print("Accuracy")
print(mean(pred$class == data.valid.norm$category_id))  # percent accurate
```

Table 4.39 showing the R Code for Classification Matrix and Accuracy Measure

The confusion matrix is created by comparing the actual and the predicted value. The result is as shown below. From the result, we could see the accuracy for this model prediction is only 30.17% with 0.5 propensity (default). We are not changing the propensity for simplicity purpose as we are going to compare multiple models.

```
          Reference
Prediction     1     2    10    15    17    19    20    22    23    24    25    26    27    28    29    30    43    44
        1      1     0     3     0     0     0     0     4     0     0     0     2     0     7     0     0     0     0
        2      0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        10    92     0   853     0    94     0    14    82   212   275     4    23    15     0     0     0     0     0
        15     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        17     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        19     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        20     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        22     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        23     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        24  2211   525  3711   514  2748   290  1181  4313  3281 11443  3140  2850  1164  1529    22     3    78     1
        25     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        26     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        27     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        28     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        29     1     0    20     0     0     0    11    16     0    26     0    12     0     1     6     0     0     0
        30     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        43     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        44     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

Overall Statistics

               Accuracy : 0.3017
                 95% CI : (0.2973, 0.3062)
    No Information Rate : 0.288
    P-Value [Acc > NIR] : 5.995e-10

                  Kappa : 0.03
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 10 Class: 15 Class: 17 Class: 19 Class: 20 Class: 22 Class: 23 Class: 24 Class: 25 Class: 26
Sensitivity          4.338e-04  0.00000   0.18596   0.0000   0.00000  0.000000  0.00000   0.0000   0.00000   0.97437   0.0000   0.0000
Specificity          9.996e-01  1.00000   0.97759   1.0000   1.00000  1.000000  1.00000   1.0000   1.00000   0.05073   1.0000   1.0000
Pos Pred Value       5.882e-02      NaN   0.51262      NaN       NaN       NaN       NaN      NaN       NaN   0.29338      NaN      NaN
Neg Pred Value       9.435e-01  0.98713   0.90454   0.9874   0.93031  0.992888  0.97043   0.8917   0.91434   0.83033   0.9229   0.9292
Prevalence           5.653e-02  0.01287   0.11249   0.0126   0.06969  0.007112  0.02957   0.1083   0.08566   0.28800   0.0771   0.0708
Detection Rate       2.452e-05  0.00000   0.02092   0.0000   0.00000  0.000000  0.00000   0.0000   0.00000   0.28062   0.0000   0.0000
Detection Prevalence 4.169e-04  0.00000   0.04081   0.0000   0.00000  0.000000  0.00000   0.0000   0.00000   0.95650   0.0000   0.0000
Balanced Accuracy    5.000e-01  0.50000   0.58178   0.5000   0.50000  0.500000  0.50000   0.5000   0.50000   0.51255   0.5000   0.5000
                     Class: 27 Class: 28 Class: 29 Class: 30 Class: 43 Class: 44
Sensitivity            0.00000   0.00000 0.2142857 0.000e+00  0.000000 0.000e+00
Specificity            1.00000   1.00000 0.9978650 1.000e+00  1.000000 1.000e+00
Pos Pred Value             NaN       NaN 0.0645161       NaN       NaN      NaN
Neg Pred Value         0.97109   0.96231 0.9994593 9.999e-01  0.998087 1.000e+00
Prevalence             0.02891   0.03769 0.0006866 7.357e-05  0.001913 2.452e-05
Detection Rate         0.00000   0.00000 0.0001471 0.000e+00  0.000000 0.000e+00
Detection Prevalence   0.00000   0.00000 0.0022806 0.000e+00  0.000000 0.000e+00
Balanced Accuracy      0.50000   0.50000 0.6060754 5.000e-01  0.500000 5.000e-01
>
>
> # Percent accuracy of model
> print("Accuracy")
[1] "Accuracy"
> print(mean(pred$class == data.valid.norm$category_id))  # percent accurate
[1] 0.3017068
>
```

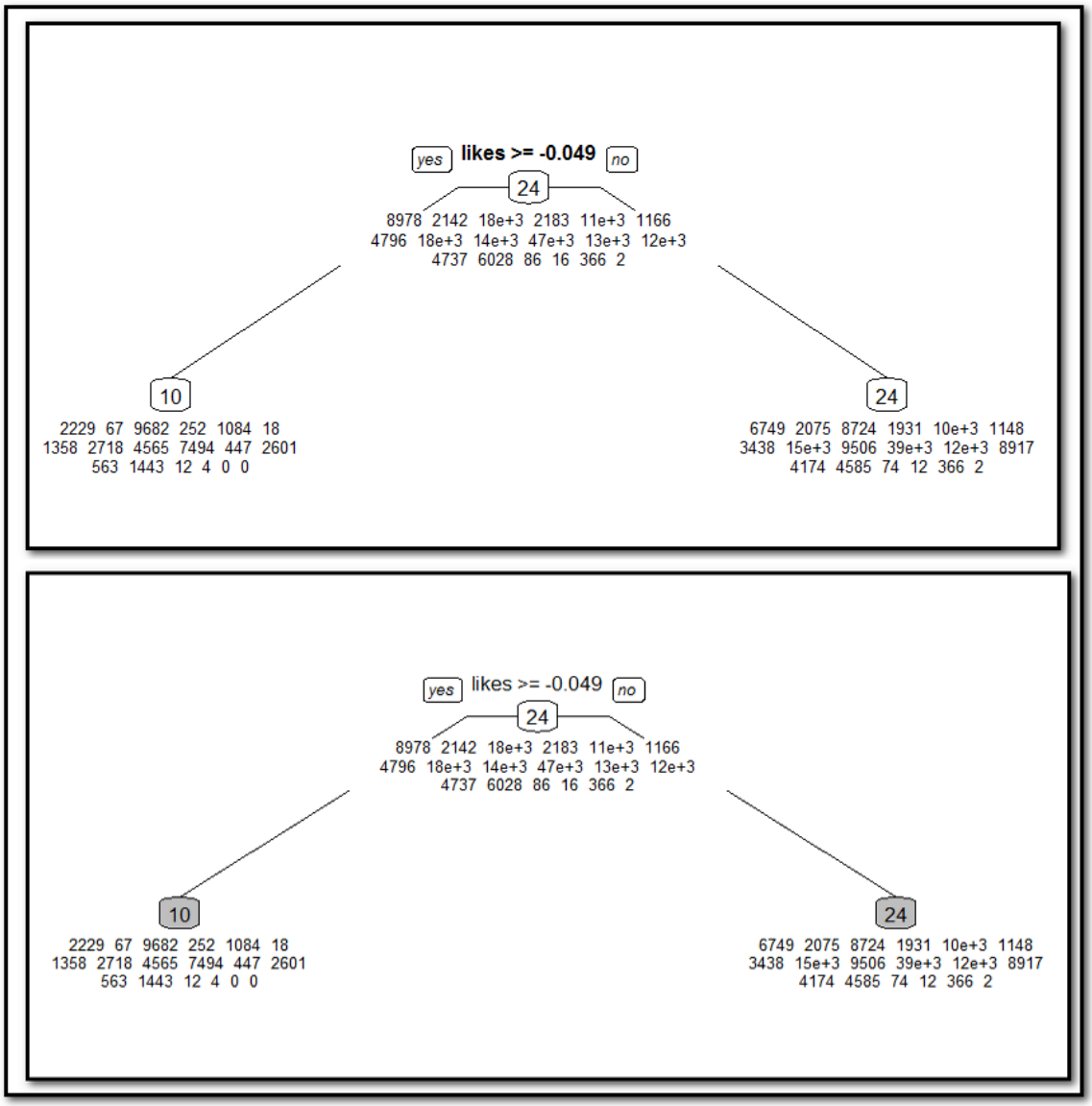
Figure 4.38 showing the classification matrix and Accuracy Measure

4.3.2 CART Model

```
##Predicting the accuracy with the CLASSIFICATION TREES model
options(screen = 999)
default.ct <- rpart(category_id ~ views+likes+dislikes+comment_count, data =
data.train.norm, method = "class")
prp(default.ct, type = 1, extra = 1, under = TRUE, split.font = 2, varlen = -
10)

### Complexity Parameters
### xval:  Number of folds to use cross-validation procedure
### CP: sets the smallest value for the complexity paraeter
```

```
cv.ct <- rpart(category_id ~ views+likes+dislikes+comment_count, data = data.train.norm, method = "class",
                cp = 0.00001, minsplit = 5, xval = 5)

### Obtaining the Best-Pruned Tree
set.seed(1)

# minsplit is the minimum number of observations in a node for a split to be attempted. xval is number K of folds in a K-fold cross-validation.

cv.ct <- rpart(category_id ~ views+likes+dislikes+comment_count, data = data.train.norm, method = "class", cp = 0.00001, minsplit = 1, xval = 5)

# Print out the cp table of cross-validation errors.
#The R-squared for a regression tree is 1 minus rel error. xerror (or relative cross-validation error where "x" stands for "cross") is a scaled version of an overall average of the 5 out-of-sample errors across the 5 folds.

pruned.ct <- prune(cv.ct, cp = 0.0154639)
prp(pruned.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
    box.col=ifelse(pruned.ct$frame$var == "<leaf>", 'gray', 'white'))
```

Table 4.40 showing R code for CART Model on YouTube dataset



Figures 4.39 showing Tree and best-pruned tree for the classification model built

4.3.2.1 Classification Matrix and Accuracy Measure

```
### Confusion Matrices

### for Training set
pruned.ct.point.pred.train <- predict(pruned.ct,
                                      data = data.train.norm,
                                      type = "class")
confusionMatrix(pruned.ct.point.pred.train, as.factor(data.train.norm$categor
y_id))

```

```
### for Validation set
pruned.ct.point.pred.valid <- predict(pruned.ct,
                                      newdata = data.valid.norm,
                                      type = "class")
confusionMatrix(pruned.ct.point.pred.valid, as.factor(data.valid.norm$categor
y_id))
```

Table 4.41 showing the R Code for Classification Matrix and Accuracy Measure

The confusion matrix is created by comparing the actual and the predicted value. The result is as shown below. From the result, we could see the accuracy for this model prediction is only 29.92% with 0.5 propensity (default). We are not changing the propensity for simplicity purpose as we are going to compare multiple models. This model performs poor in accuracy when compared to the LDA model.

```
> confusionMatrix(pruned.ct.point.pred.valid, as.factor(data.valid.norm$category_id))
Confusion Matrix and Statistics

          Reference
Prediction    1    2   10   15   17   19   20   22   23   24   25   26   27   28   29   30   43   44
        1     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        2     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        10  592    8 2384   53  282    5  377  698 1127 1927  111  635  137  363    6    1    0    0
        15    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        17    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        19    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        20    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        22    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        23    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        24 1713  517 2203  461 2560  285  829 3717 2366 9817 3033 2252 1042 1174   22    2   78    1
        25    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        26    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        27    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        28    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        29    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        30    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        43    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
        44    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0

Overall Statistics

               Accuracy : 0.2992
                 95% CI : (0.2948, 0.3037)
    No Information Rate : 0.288
    P-Value [Acc > NIR] : 0.0000003277

                  Kappa : 0.0649
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 10 Class: 15 Class: 17 Class: 19 Class: 20 Class: 22
Sensitivity           0.00000  0.00000   0.51973    0.0000   0.00000  0.000000   0.00000     0.0000
Specificity           1.00000  1.00000   0.82532    1.0000   1.00000  1.000000   1.00000     1.0000
Pos Pred Value            NaN      NaN   0.27383       NaN       NaN       NaN       NaN        NaN
Neg Pred Value        0.94347  0.98713   0.93131    0.9874   0.93031  0.992888   0.97043     0.8917
Prevalence            0.05653  0.01287   0.11249    0.0126   0.06969  0.007112   0.02957     0.1083
Detection Rate        0.00000  0.00000   0.05846    0.0000   0.00000  0.000000   0.00000     0.0000
Detection Prevalence  0.00000  0.00000   0.21350    0.0000   0.00000  0.000000   0.00000     0.0000
Balanced Accuracy     0.50000  0.50000   0.67252    0.5000   0.50000  0.500000   0.50000     0.5000
                     Class: 23 Class: 24 Class: 25 Class: 26 Class: 27 Class: 28 Class: 29  Class: 30
Sensitivity            0.00000    0.8359    0.0000    0.0000   0.00000   0.00000 0.0000000 0.00000000
Specificity            1.00000    0.2335    1.0000    1.0000   1.00000   1.00000 1.0000000 1.00000000
Pos Pred Value             NaN    0.3061       NaN       NaN       NaN       NaN       NaN        NaN
Neg Pred Value         0.91434    0.7787    0.9229    0.9292   0.97109   0.96231 0.9993134 0.99992643
Prevalence             0.08566    0.2880    0.0771    0.0708   0.02891   0.03769 0.0006866 0.00007357
Detection Rate         0.00000    0.2407    0.0000    0.0000   0.00000   0.00000 0.0000000 0.00000000
Detection Prevalence   0.00000    0.7865    0.0000    0.0000   0.00000   0.00000 0.0000000 0.00000000
Balanced Accuracy      0.50000    0.5347    0.5000    0.5000   0.50000   0.50000 0.5000000 0.50000000
                     Class: 43  Class: 44
Sensitivity           0.000000 0.00000000
Specificity           1.000000 1.00000000
Pos Pred Value             NaN        NaN
```

Figure 4.40 showing the classification matrix and Accuracy Measure

4.3.3 KNN Model

In this section, the category id is predicted using the KNN model. The code and the results are as follows

```
1.  ##Predicting Accuracy with the KNN MODEL
2.
3.  nn <- knn(train = data.train.norm[,
    c("views","likes","dislikes","comment_count")],
4.            test =
    data.valid.norm[,c("views","likes","dislikes","comment_count")],
5.            cl = data.train.norm[["category_id"]], k = 3)
6.
7.
8.  data.valid.norm$category_id = as.factor(data.valid.norm$category_id)
9.  confusionMatrix(nn, data.valid.norm$category_id)
```

Table 4.42 showing R code for KNN on YouTube dataset

```
> summary(nn)
   1    2   10   15   17   19   20   22   23   24   25   26   27   28
2192  403 6014  461 2698  195 1356 4309 3160 11967 2748 2666  963 1117
  29   30   43   44
   9    3   93    1
```

Figures 4.41 showing KNN Model output with category groping

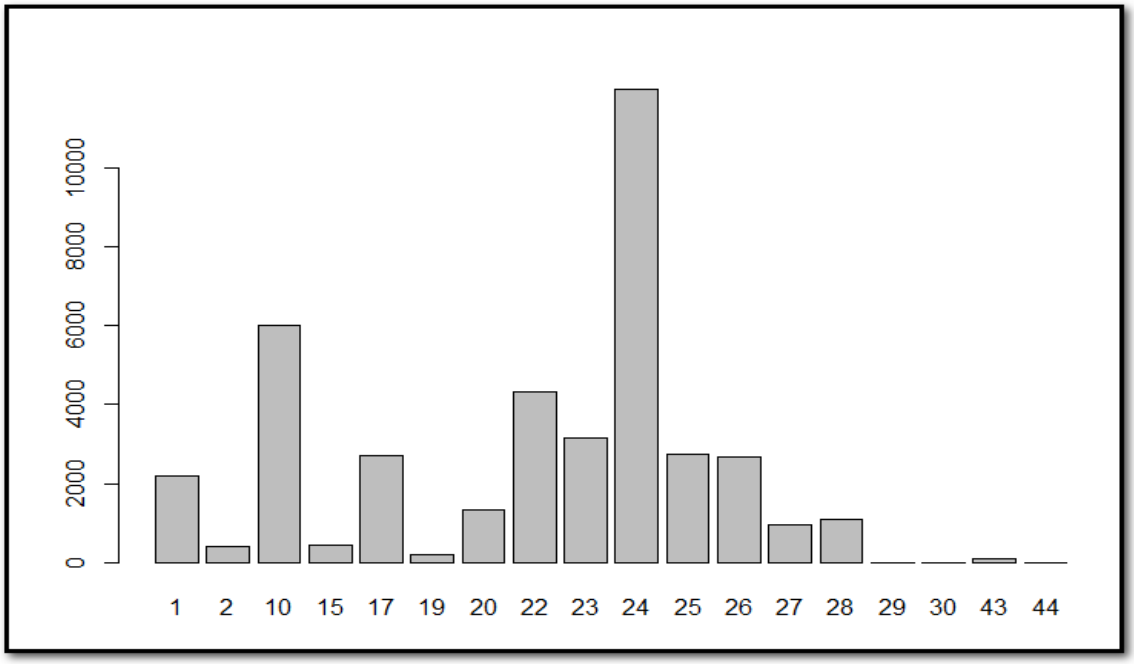


Figure 4.42 showing KNN category Result

The above figure shows the category prediction using the KNN model. The bar chart shows the frequency of the categories with respect to a number of videos in each category.

4.3.3.1 Classification Matrix and Accuracy Measure

```
1.
2.  #Get the confusion matrix to see accuracy value and other parameter values
3.
4.  data.valid.norm$category_id = as.factor(data.valid.norm$category_id)
5.  KNN_conf <- confusionMatrix(nn, data.valid.norm$category_id)
6.  KNN_conf
7.
8.  knn_auc <- multiclass.roc(data.valid.norm$category_id, as.numeric(nn))
9.  auc(knn_auc)
```

Table 4.43 showing the R Code for Classification Matrix and Accuracy Measure

```
> KNN_conf
Confusion Matrix and Statistics

          Reference
Prediction    1    2   10   15   17   19   20   22   23   24   25   26   27   28   29
        1  1049   10  113   19   95   13   48  155   89  375   98   63   29   36    0
        2    11   84   24    1   16    7   13   48   20   92   40   21   12   12    0
        10  109   30 3913   18  139    9   71  270  347  706   87  195   46   69    3
        15   15    5   31  179   23    2   17   37   24   71   10   25    7   15    0
        17  103   39  128   19 1125   11   43  237  113  566  172   75   26   40    1
        19   10    4   10    1   10   57    4   21   12   38   10   11    3    4    0
        20   41   13   79   12   77    5  574   89   67  219   65   59   37   19    0
        22  183   66  260   35  228   22   92 1648  184  935  324  174   84   70    0
        23   82   22  310   22  123   13   90  181 1451  575   50  126   51   64    0
        24  459  114  726   78  619   67  255 1031  580 6586  678  394  166  194    1
        25   74   34   75   13  223   10   50  324   56  566 1175   56   42   44    1
        26   84   33  238   33   94   15   73  192  145  411   76 1143   64   63    2
        27   30   21   68   12   25    7   20   71   53  161   45   65  364   20    0
        28   24   10   74    7   39    2   31   70   60  194   35   45   20  506    0
        29    0    0    1    0    0    0    0    0    0    1    0    2    0    0    5
        30    0    0    1    0    0    0    0    0    1    0    0    0    0    0    0
        43    0    0    3    0    0    0    0    4    0   33    2    0    1    2    0
        44    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0
          Reference
Prediction   30   43   44
        1     0    0    0
        2     0    2    0
        10    1    1    0
        15    0    0    0
        17    0    0    0
        19    0    0    0
        20    0    0    0
        22    0    4    0
        23    0    0    0
        24    0   19    0
        25    0    4    1
        26    0    0    0
        27    0    1    0
        28    0    0    0
        29    0    0    0
        30    1    0    0
        43    0   48    0
        44    0    0    0

Overall Statistics

               Accuracy : 0.4933
                 95% CI : (0.4884, 0.4982)
    No Information Rate : 0.2857
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.4079
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 10 Class: 15 Class: 17 Class: 19 Class: 20
```

Figure 4.43 showing the Confusion Matrix output for KNN model

The confusion matrix is created by comparing the actual and the predicted value. The result is as shown above. From the result, we could see the accuracy for this model prediction is 49.33% with 0.5 propensity (default). This is the highest out of all the three models developed and hence this model is considered the best in predicting the category ID to a better extent.

4.3.4 Summary of Classification Models

The summary the for the classification models are as follows.

```
#Model Summarization

# creating a data frame to summarize the results


Model <- c('LDA', 'CART-Tuned', 'KNN')

Accuracy <- c(lda_conf$overall[[1]], CART_Conf$overall[[1]],
              KNN_conf$overall[[1]])

AUC <- c(auc(LDA_auc), auc(CART_auc), auc(knn_auc))

Results <- data.frame(Model, Accuracy, AUC)
Results
```

Table 4.44 showing the R code for Category Model Summary

```
> Results
       Model  Accuracy       AUC
1        LDA 0.3116590 0.5206872
2 CART-Tuned 0.3279643 0.5294712
3        KNN 0.4933218 0.6830059
```

Figure 4.44 showing the Accuracy and AUC for all the three models developed

From the above figure, we can conclude that the KNN has the highest accuracy of 49.33% in predicting the category ids accurately. The area under the curve for all the three models are also calculated and KNN stands out in that also. Hence, we can conclude that KNN is the best fit model out of these three in classifying the category id from the features considered.

5 Sentiment Analysis

In this section, the sentiment analysis on the YouTube video title and description has been carried out and the inference on the sentiment of the videos are analyzed. The bar chart with different sentiment count and a word cloud showing the frequency of the words used in the dataset. The goal of performing sentiment analysis here is to identify the keywords associated with a video to be able to use them to categorize them. For example, a video with the title "Official music video" will belong to the music category and so on. Similarly, it can be performed on description and tags of the video to create a set of keywords associated with a video.

5.1 Sentiment Analysis on YouTube video title

In this section, we have done the sentiment analysis on the YouTube video title of each region dataset and merged dataset. We have used "calculate_Sentiment" from Rsentiment package to calculate the actual sentiments of the words used in the title of each video. Also, we have used word cloud from word cloud package to create a word cloud that repeats more than a certain limit on the dataset. Creating a word cloud was one of the interesting things in this project and it clearly shows the relation between the words used in most trending videos. An interesting fact which we had learned while doing this analysis was few video titles deliberately include words that went viral in the past though there were not related to videos which had increased the views to that video and they have become trending. This was a comment given by a famous Indian YouTuber while searching for a few interesting facts on "how to make a YouTube video trending".

5.1.1 Sentiment Analysis on YouTube video title – United States

```
# United States

set.seed(1000)
corpus <- Corpus(VectorSource(list(us$title)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
```

```
10.
11. dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
12. freq <- colSums(as.matrix(dtm))
13. sentiments_cal <- calculate_sentiment(names(freq))
14. sentiments <- cbind(sentiments_cal, as.data.frame(freq))
15. sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L
16.
17. wordcloud(sentiments$text,sentiments$freq, min.freq = 5,max.words = 300,color
    s = brewer.pal(6,"Dark2"),random.order = F)
18.
19. sentiments <- as.data.table(sentiments_cal)
20. sentiments1 <- sentiments[,.N,by=.(sentiment)]
21. sentiments1[,"Total":=sum(N)]
22. sentiments1 <- sentiments1[,.("Percentage"=100*N/Total),by=.(sentiment)]
23.
24. ggplot(sentiments1,aes(x = sentiment,y = Percentage ,fill=sentiment )) + geom
    _bar(stat = "identity") + ggtitle("Sentiment analysis on title - United State
    s") +
25.   xlab("Sentiment type")+ylab(" Percentage") + theme(axis.text.x = element_te
    xt(angle = 45, size=8,hjust = 1)) +
26.   theme_minimal() +  theme(plot.title = element_text(hjust = 0.5))
```

Table 5.1 R Code for sentiment analysis and Word cloud for YouTube Video title

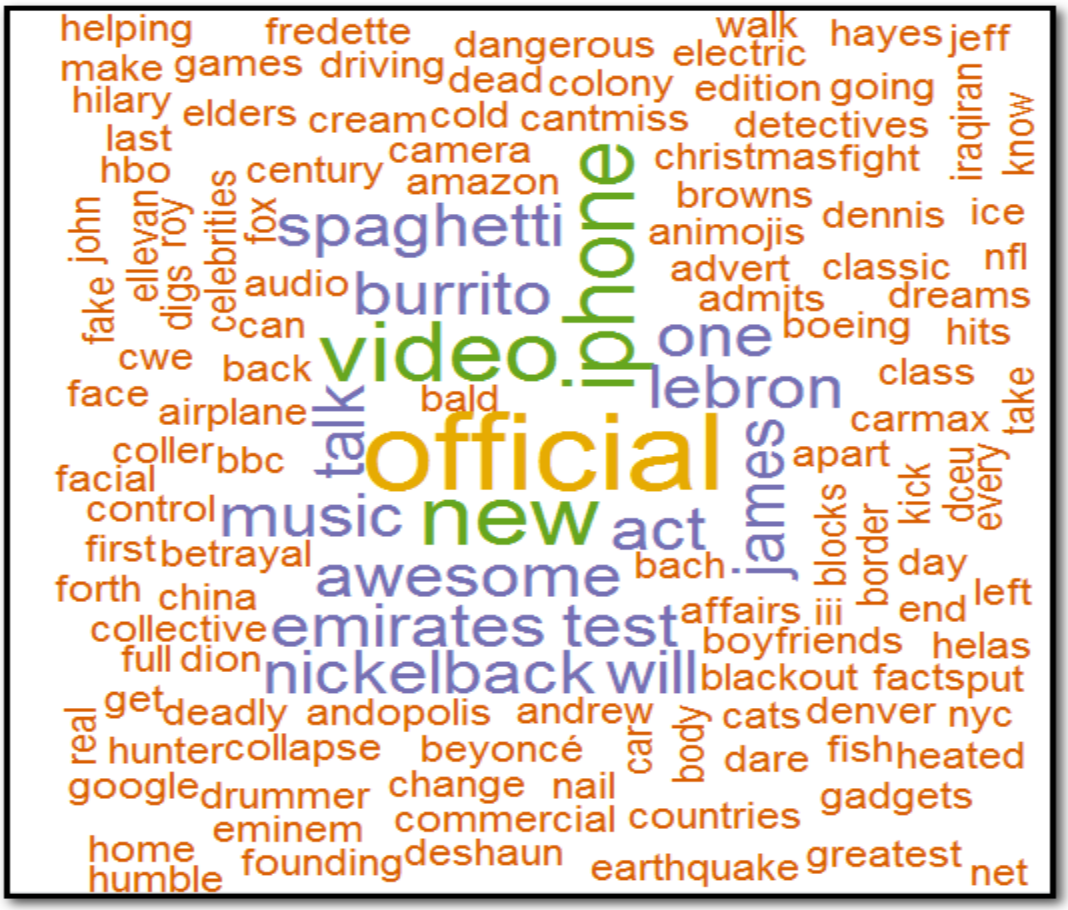


Figure 5.1 Word Cloud for the most used words in the video title

Interestingly, the number of words which had repeated more than 5 times in the title is very high in the United States. From the above word cloud, it is evident that Official, Video, New, James, Nickelback, Emirates, awesome, spaghetti, iPhone are the words that used a greater number of times in the video that went trending in the United States. We could conclude from this the videos which got trended are titled with catchy phrases like New, awesome, and cookery videos are also popular in the United States. Also, we could see there is a huge craze for iPhone and a lot of videos have posted and getting trended on iPhone.

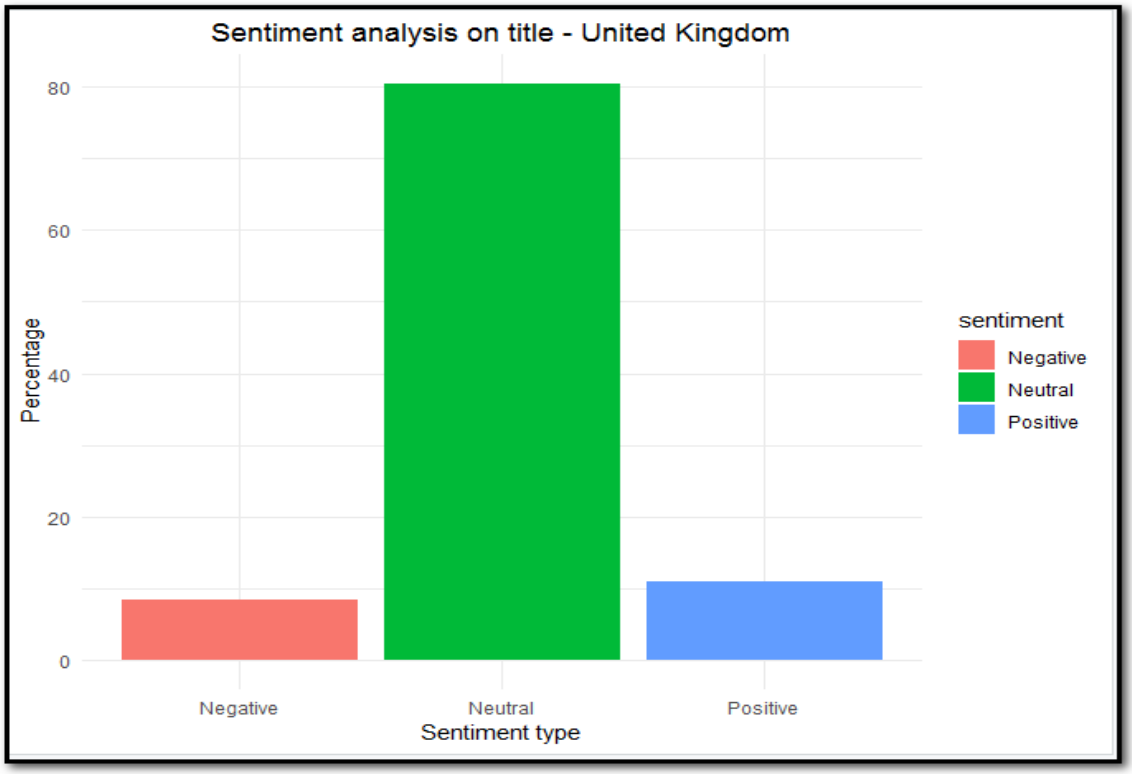


Figure 5.2 Sentiment bar graph based on the sentiment percentage

The above bar chart gives us the percentage of the sentiment on video titles. We could see 80% of the words in the title are neutral words whereas almost 12 % of the words are positive and 8% of the words are negative in nature. This clearly gives the trend in the United States.

5.1.2 Sentiment Analysis on YouTube video title – United Kingdom

```
#Performing sentiment analysis on video title and description.

#Sentiment Analysis on Video Title:

gb <- fread("GBvideos.csv",encoding = "UTF-8")
fr <- fread("FRvideos.csv",encoding = "UTF-8")
ca <- fread("CAvideos.csv",encoding = "UTF-8")
us <- fread("USvideos.csv",encoding = "UTF-8")
de <- fread("DEvideos.csv",encoding = "UTF-8")


# Sentiment Analysis:

#United Kingdom

set.seed(1000)
corpus <- Corpus(VectorSource(list(gb$title)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))

dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L

wordcloud(sentiments$text,sentiments$freq, min.freq = 2,max.words = 300,colors = brewer.pal(6,"Dark2"),random.order = F)

sentiments <- as.data.table(sentiments_cal)
sentiments1 <- sentiments[,.N,by=.(sentiment)]
sentiments1[,"Total":=sum(N)]
sentiments1 <- sentiments1[,.("Percentage"=100*N/Total),by=.(sentiment)]

ggplot(sentiments1,aes(x = sentiment,y = Percentage ,fill=sentiment )) + geom_bar(stat = "identity") + ggtitle("Sentiment analysis on title - United Kingdom") +
  xlab("Sentiment type")+ylab(" Percentage") + theme(axis.text.x = element_text(angle = 45, size=8,hjust = 1)) +
  theme_minimal() +   theme(plot.title = element_text(hjust = 0.5))
```

Table 5.2 R Code for sentiment analysis and Word cloud for YouTube Video title

Figure 5.3 Word Cloud for the most used words in the video title

From the above word cloud, it is evident that Official, Video, Music, and Audio are the words that used a greater number of times in the video that went trending in the United Kingdom. We could also see Taylor, Christmas, autumn etc in the word cloud.

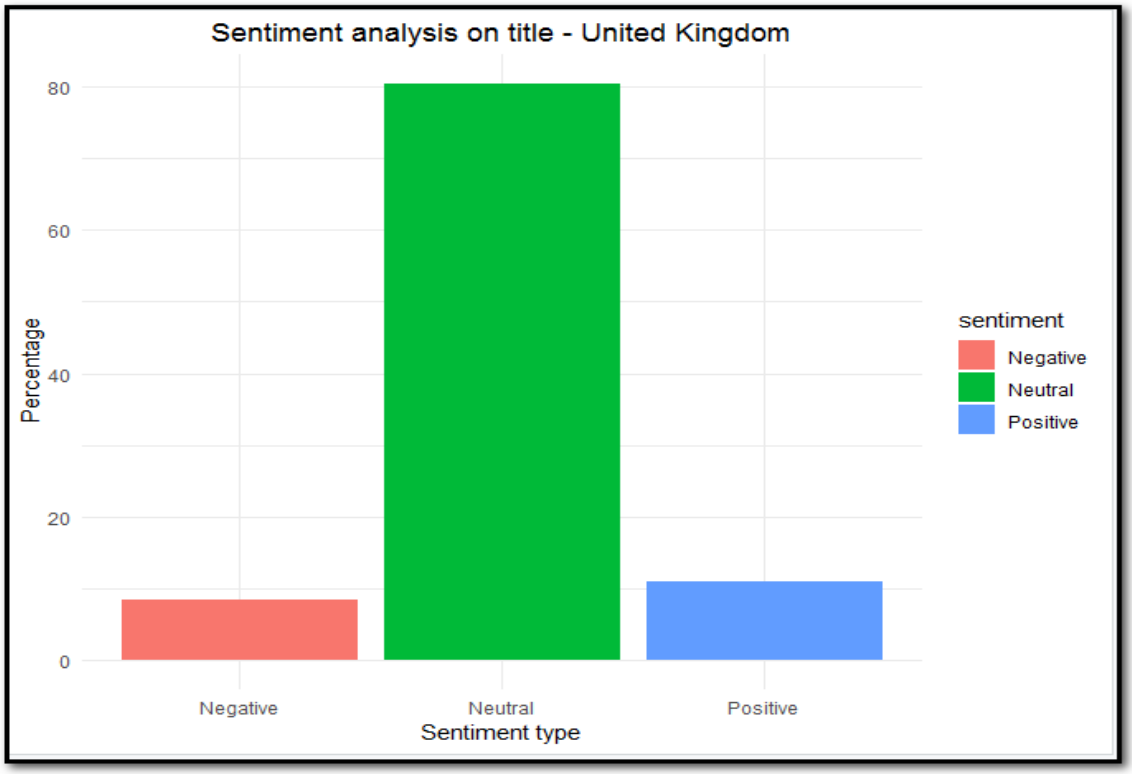


Figure 5.4 Sentiment bar graph based on the sentiment percentage

The sentiment type plot remains almost the same as that of the United States. We could see 80% of the words in the title are neutral words whereas almost 12 % of the words are positive and 8% of the words are negative in nature. This clearly gives the trend in the United Kingdom.

5.1.3 Sentiment Analysis on YouTube video title – Canada

```
1.    # Canada
2.
3.  set.seed(1000)
4.  corpus <- Corpus(VectorSource(list(ca$title)))
5.  corpus <- tm_map(corpus, removePunctuation)
6.  corpus <- tm_map(corpus, content_transformer(tolower))
7.  corpus <- tm_map(corpus, removeNumbers)
8.  corpus <- tm_map(corpus, stripWhitespace)
9.  corpus <- tm_map(corpus, removeWords, stopwords("english"))
10.
11. dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
12. freq <- colSums(as.matrix(dtm))
13. sentiments_cal <- calculate_sentiment(names(freq))
14. sentiments <- cbind(sentiments_cal, as.data.frame(freq))
15. sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L
16.
17. wordcloud(sentiments$text,sentiments$freq, min.freq = 2,max.words = 300,color
    s = brewer.pal(6,"Dark2"),random.order = F)
18.
19. sentiments <- as.data.table(sentiments_cal)
20. sentiments1 <- sentiments[,.N,by=.(sentiment)]
21. sentiments1[,"Total":=sum(N)]
22. sentiments1 <- sentiments1[,.("Percentage"=100*N/Total),by=.(sentiment)]
23.
24. ggplot(sentiments1,aes(x = sentiment,y = Percentage ,fill=sentiment )) + geom
    _bar(stat = "identity") + ggtitle("Sentiment analysis on title - Canada") +

25.   xlab("Sentiment type")+ylab(" Percentage") + theme(axis.text.x = element_te
    xt(angle = 45, size=8,hjust = 1)) +
26.   theme_minimal() + theme(plot.title = element_text(hjust = 0.5))


40.
```

Table 5.3 R Code for sentiment analysis and Word cloud for YouTube Video title

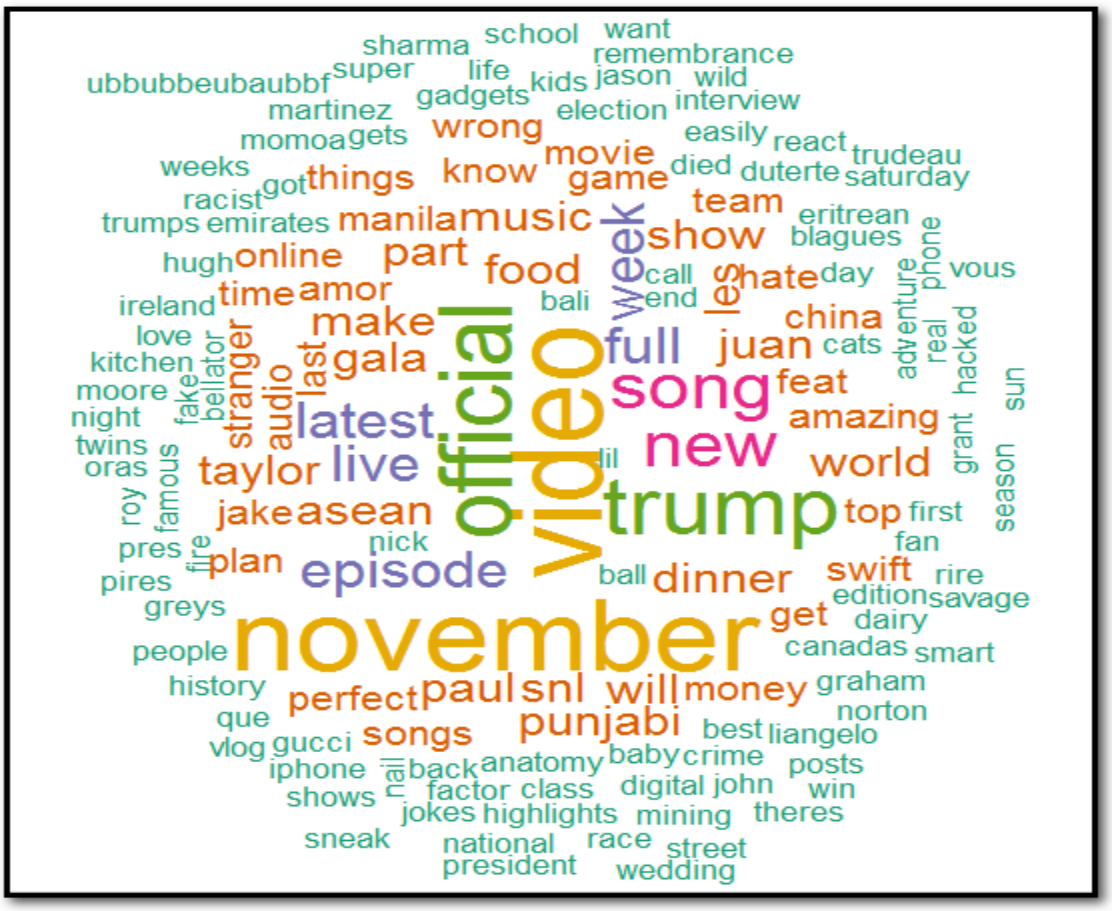


Figure 5.5 Word Cloud for the most used words in the video title

From the word cloud, it is evident that the words “Official”, “Video”, “latest”, “November” have been used most number of times. Interestingly, videos with words “Trump” has trended in Canada rather in the United States. Also, we could Punjabi is another word repeated multiple times which could infer us there is a larger portion of people watching Punjabi videos. Other predominant words that trended as like other regions are music, Taylor, dinner, swift, iPhone.

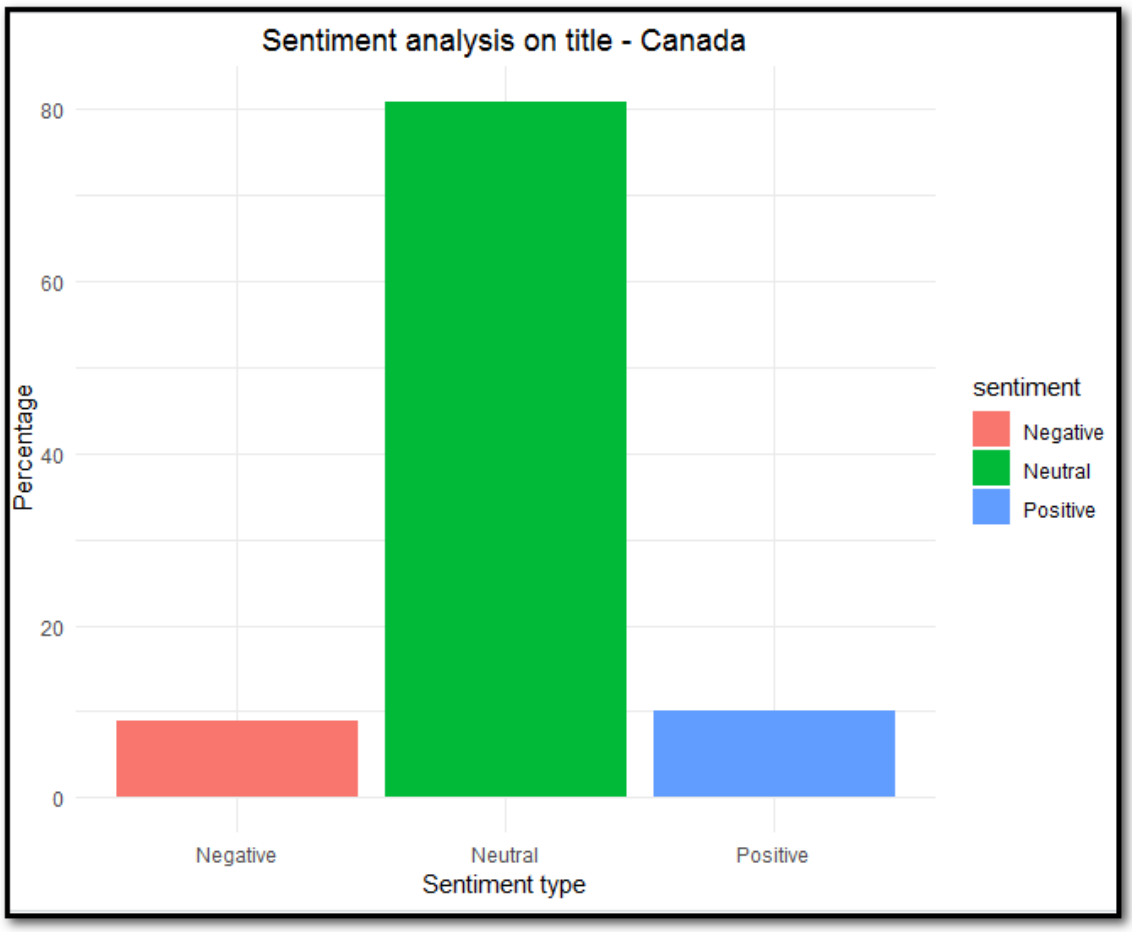


Figure 5.6 Sentiment bar graph based on the sentiment percentage

The sentiment type plot remains almost same for Canada also. Here, we could see there is a slight increase in neutral words to 81% and positive to 10% and negative words are of 9% percentage. We can conclude that most words are in a neutral zone in all the regions analyzed till now.

5.1.4 Sentiment Analysis on YouTube video title – Germany

```
# Germany

set.seed(1000)
corpus <- Corpus(VectorSource(list(de$title)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))

```

```
dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L

wordcloud(sentiments$text,sentiments$freq, min.freq = 2,max.words = 300,colors = brewer.pal(6,"Dark2"),random.order = F)

sentiments <- as.data.table(sentiments_cal)
sentiments1 <- sentiments[,.N,by=.(sentiment)]
sentiments1[,"Total":=sum(N)]
sentiments1 <- sentiments1[,.("Percentage"=100*N/Total),by=.(sentiment)]

ggplot(sentiments1,aes(x = sentiment,y = Percentage ,fill=sentiment )) + geom_bar(stat = "identity") + ggtitle("Sentiment analysis on title - Germany") +

  xlab("Sentiment type")+ylab(" Percentage") + theme(axis.text.x = element_text(angle = 45, size=8,hjust = 1)) +
  theme_minimal() + + theme(plot.title = element_text(hjust = 0.5))
```

Table 5.4 R Code for sentiment analysis and Word cloud for YouTube Video title

Figure 5.7 Showing Word Cloud for the most used words in the video title

From the word cloud, we could see predominantly more German words are used as a title in Germany which are viral or trending in Germany. The word which was used a greater number of times is below and das. Also, other words with high frequency are died, auf, der, mit, freundin. Few English words like water, video, unboxing, the official is also been used many videos that have trended in Germany.

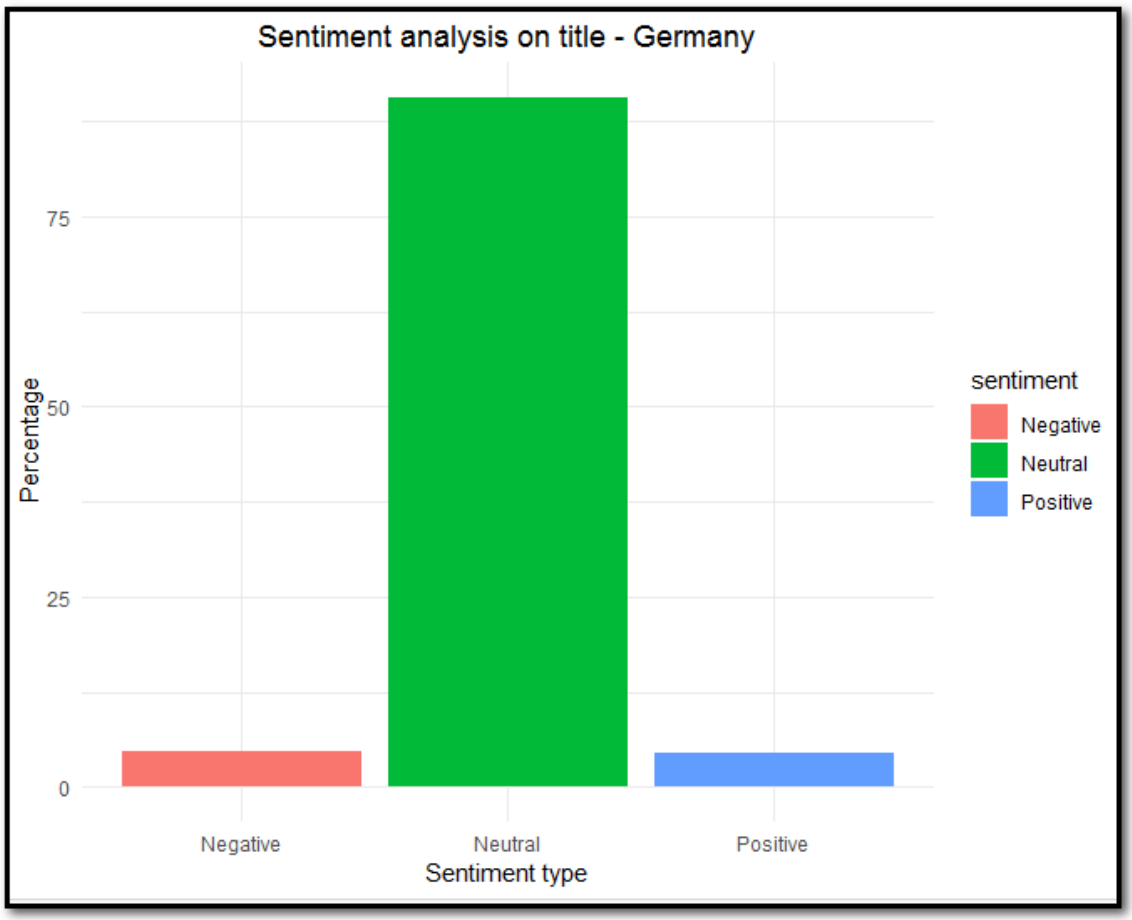


Figure 5.8 Sentiment bar graph based on the sentiment percentage

The number of neutral words has increased in Germany which is almost 90% of total words. The Negative and positive sentiments are occupying almost 4.5% each. This can be due to the use of a greater number of German words and R could not have handled it properly. This might be a shortcoming and needed more analysis and this might not result in the actual outcome of the title.

5.1.5 Sentiment Analysis on YouTube video title – France

```
# France
set.seed(1000)
corpus <- Corpus(VectorSource(list(fr$title)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
```

```
9.
10. dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
11. freq <- colSums(as.matrix(dtm))
12. sentiments_cal <- calculate_sentiment(names(freq))
13. sentiments <- cbind(sentiments_cal, as.data.frame(freq))
14. sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L
15.
16. wordcloud(sentiments$text,sentiments$freq, min.freq = 2,max.words = 300,color
    s = brewer.pal(6,"Dark2"),random.order = F)
17.
18. sentiments <- as.data.table(sentiments_cal)
19. sentiments1 <- sentiments[,.N,by=.(sentiment)]
20. sentiments1[,"Total":=sum(N)]
21. sentiments1 <- sentiments1[,.("Percentage"=100*N/Total),by=.(sentiment)]
22.
23. ggplot(sentiments1,aes(x = sentiment,y = Percentage ,fill=sentiment )) + geom
    _bar(stat = "identity") + ggtitle("Sentiment analysis on title - France") +

24.   xlab("Sentiment type")+ylab(" Percentage") + theme(axis.text.x = element_te
    xt(angle = 45, size=8,hjust = 1)) +
25.   theme_minimal() +  theme(plot.title = element_text(hjust = 0.5))
```

Table 5.5 R Code for sentiment analysis and Word cloud for YouTube Video title

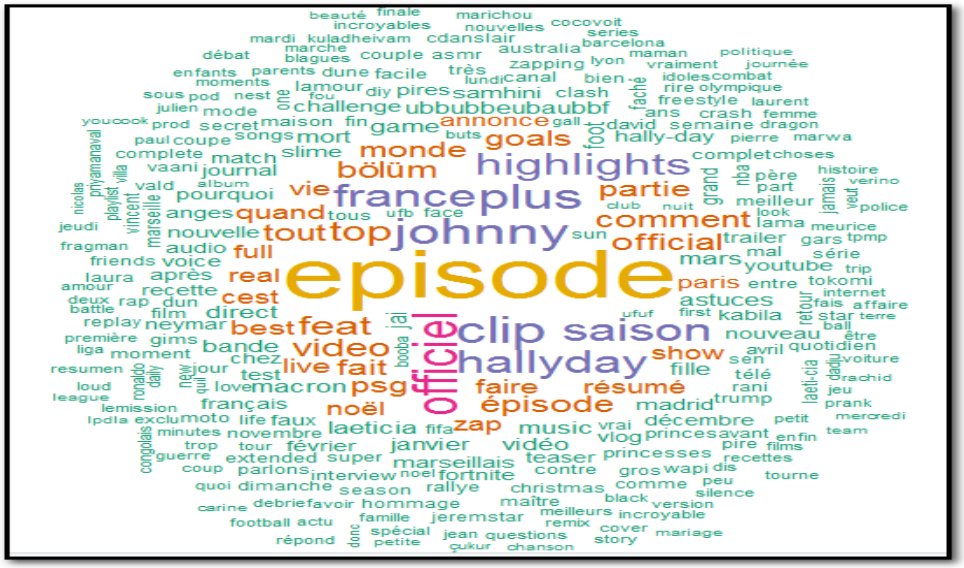


Figure 5.9 Word Cloud for the most used words in the video title

From the word cloud, a lot of words have been repeated more than 5 times in the video title. There is a huge number of English and French words mixed up in the title which have trended in France. The predominant word is “episode” probably a series or a serial was a hit and trended during the period of the data set. Other words include Johnny, France, plus,

highlights, comment, official, holiday, video etc. The highlights might even refer to French open Tennis highlights which would have concluded in the time period in the dataset.

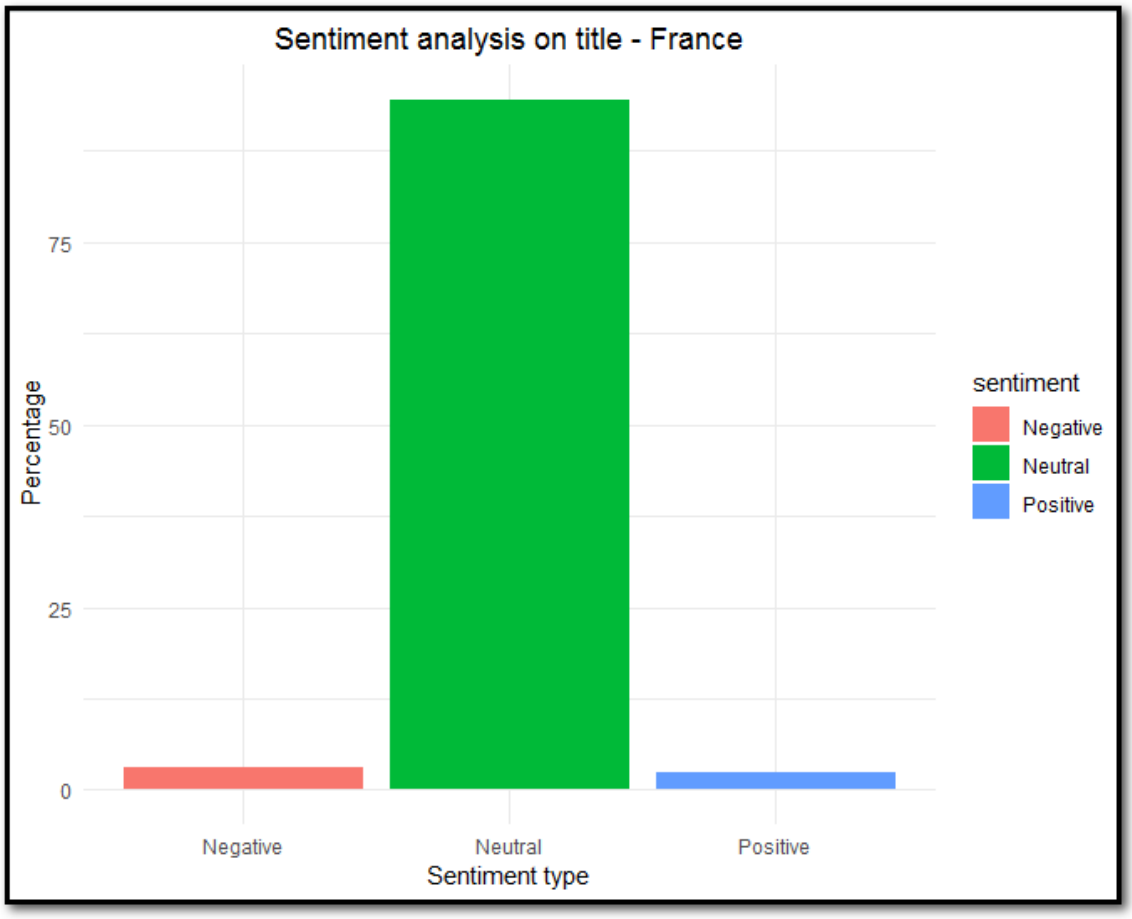


Figure 5.10 Sentiment bar graph based on the sentiment percentage

The sentiments are still moving towards neutral which is about 95% and positive and negative sentiment occupy only 5 % together. This can be because of French words and many words are used in the trending videos. Despite removing stop words from French, there might have been words with no meanings in the title and hence it has skewed the neutral sentiments.

5.1.6 Sentiment Analysis on YouTube video title – Overall Dataset

```
#Overall
set.seed(1000)
corpus <- Corpus(VectorSource(list(videos$title)))
corpus <- tm_map(corpus, removePunctuation)
```

```
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, removeWords, stopwords("french"))
corpus <- tm_map(corpus, removeWords, stopwords("german"))

dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L

wordcloud(sentiments$text,sentiments$freq, min.freq = 5,max.words = 300,colors = brewer.pal(6,"Dark2"),random.order = F)

sentiments <- as.data.table(sentiments_cal)
sentiments1 <- sentiments[,.N,by=.(sentiment)]
sentiments1[,"Total":=sum(N)]
sentiments1 <- sentiments1[,.("Percentage"=100*N/Total),by=.(sentiment)]

ggplot(sentiments1,aes(x = sentiment,y = Percentage ,fill=sentiment )) + geom_bar(stat = "identity") + ggtitle("Sentiment analysis on title - All Countries") +
  xlab("Sentiment type")+ylab(" Percentage") + theme(axis.text.x = element_text(angle = 45, size=8,hjust = 1)) +
  theme_minimal() +  theme(plot.title = element_text(hjust = 0.5)
```

Table 5.6 R Code for sentiment analysis and Word cloud for YouTube Video title

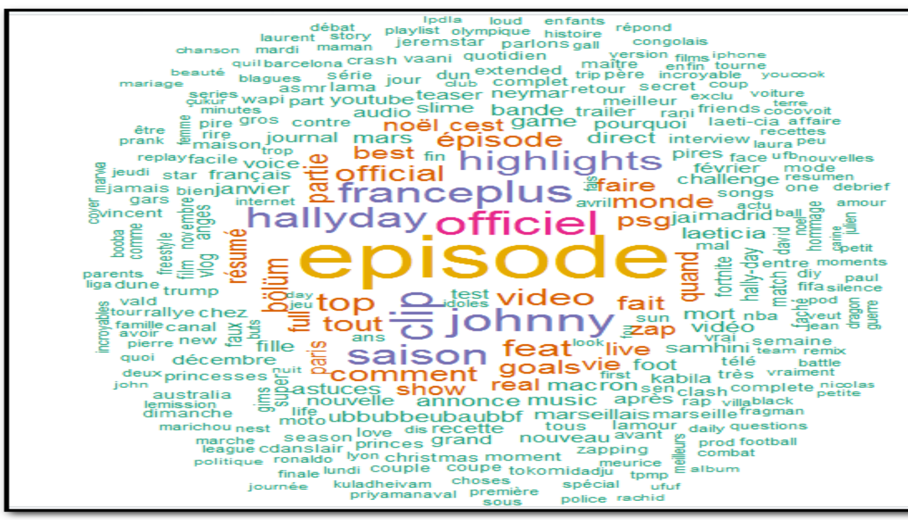


Figure 5.11 Word Cloud for the most used words in the video title

The word cloud of the overall dataset showcases that the word “episode” has been used the most number of times in title across all regions. It is then followed by Officiel, Hallyday,

franceplus, highlights, official, comment etc. The word cloud clearly shows the trend in all five countries together.

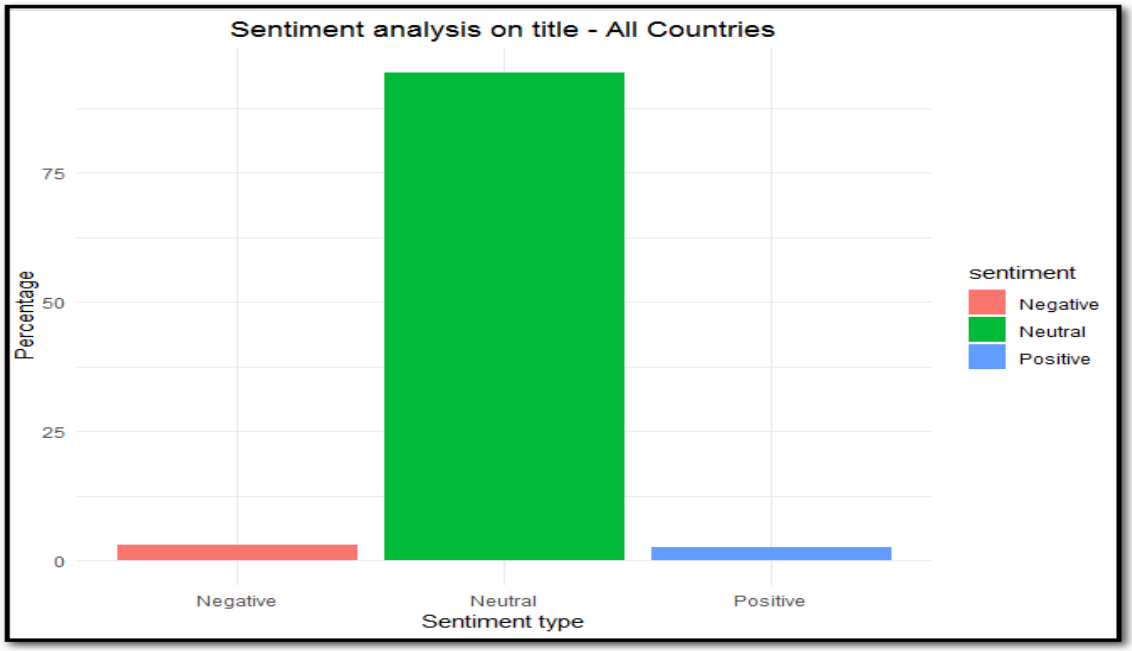


Figure 5.12 Sentiment bar graph based on the sentiment percentage

The sentiments type as expected is having a more of neutral which is occupying 95% and positive and negative sentiments together take up around 5 %. This is clearly seen from the trends that have been in the individual regions and could be a skewed value as stated before. Thus, the sentiment analysis of video title has done successfully, and the results are interpreted as expected.

5.2 Sentiment Analysis on YouTube video description

This section, we have run the sentiment analysis for the Video description. We have analyzed the sentiment of the words in the description and categorized them based on sentiments like Positive, anger, Joy, fear, sadness, negative etc. These are then plotted into bar plot and are compared for each region and overall region. In the below section, the analysis has been carried out in detail for each region and for the overall dataset.

5.2.1 Sentiment Analysis on YouTube video description – United States

```
#United States

us_desc_sentiment <- get_nrc_sentiment(us$description)
us_desc_sentiment <- data.frame(feeling = names(colSums(us_desc_sentiment)), total = colSums(us_desc_sentiment), row.names = NULL)
set.seed(1000)
corpus <- Corpus(VectorSource(list(us$description)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))

dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
#sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L

wordcloud(sentiments$text,sentiments$freq, min.freq = 5,max.words = 300,colors = brewer.pal(6,"Dark2"),random.order = F)

sentiments <- as.data.table(sentiments_cal)
sentiments1 <- sentiments[,.N,by=.(sentiment)]
sentiments1[,"Total":=sum(N)]
sentiments1 <- sentiments1[,.("Percentage"=100*N/Total),by=.(sentiment)]


ggplot(data = us_desc_sentiment, aes(x = reorder(feeling, -total, na.rm=TRUE), y = total)) +
  geom_bar(aes(fill = feeling), stat = "identity") +
  theme_minimal()+
  theme(legend.position = "none", axis.text.x = element_text(angle=45)) +
  xlab("Sentiment") + ylab("Total Count") + ggtitle("Total Description Sentiment Score - UNITED STATES") +
  theme(plot.title = element_text(hjust = 0.5))  # add to all plots
```

Table 5.7 showing R Code for sentiment analysis and Word cloud for YouTube Video description

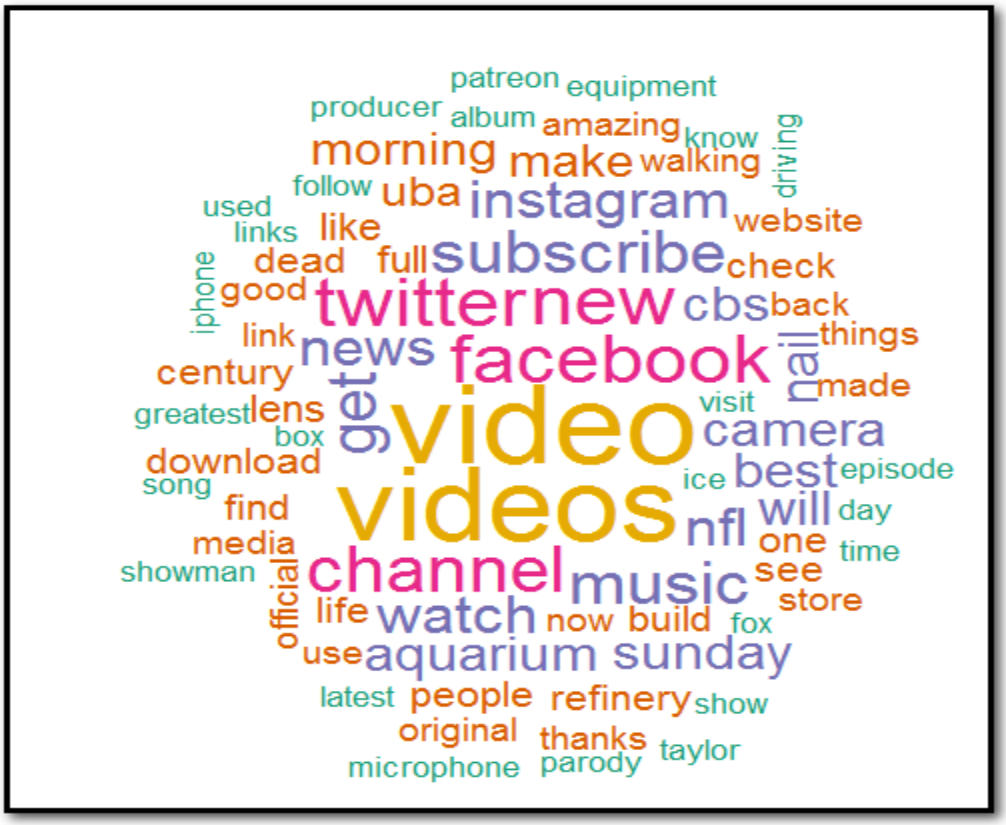


Figure 5.23 Showing Word Cloud for the most used words in the video description

From the word cloud, it is evident that Video, Videos, Facebook, Twitter, subscribe, Instagram, Sunday, channel, music, morning, camera are the most repetitive words used in the description. We can understand that most of the video description has been positive and the publisher has tried to give more catch words like so that more subscribers watch their videos.

The below bar chart showcases the words used in description grouped into different sentiments like anger, joy, happiness, sad etc. From the plot, we could see that there are a greater number of positive sentiments used in the video description which would make a lot of people to watch the views. The next high sentiment is anticipation, which clearly says why the videos are getting trending as they create a huge anticipation among the viewers by these catchy words. The other sentiments like trust, negative, joy, fear, sadness, surprise, anger and disgust follow them in order. As a conclusion, there are a greater number of positive words used in the description of the videos which got trended in the United States in the selected period.

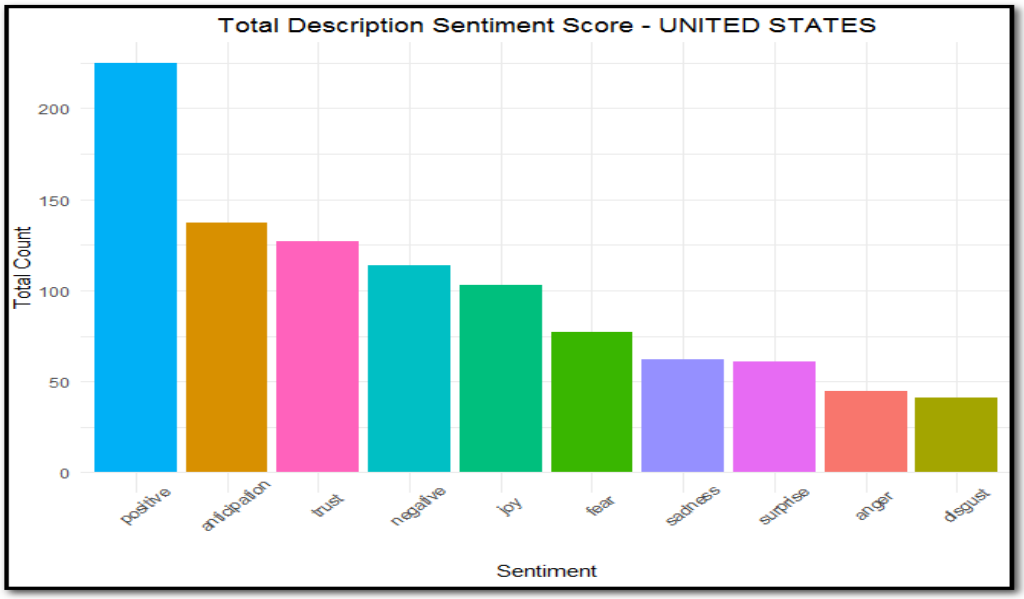


Figure 5.24 showing Sentiment bar graph based on the sentiment percentage

5.2.2 Sentiment Analysis on YouTube video description – United Kingdom

```
# UNITED KINGDOM
gb_desc_sentiment <- get_nrc_sentiment(gb$description)
gb_desc_sentiment <- data.frame(feeling = names(colSums(gb_desc_sentiment)), total = colSums(gb_desc_sentiment), row.names = NULL)
set.seed(1000)
corpus <- Corpus(VectorSource(list(gb$description)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
#sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L
wordcloud(sentiments$text,sentiments$freq, min.freq = 5,max.words = 300,colors = brewer.pal(6,"Dark2"),random.order = F)
ggplot(data = gb_desc_sentiment, aes(x = reorder(feeling, -total, na.rm=TRUE), y = total)) +
  geom_bar(aes(fill = feeling), stat = "identity") +
  theme_minimal()+
  theme(legend.position = "none", axis.text.x = element_text(angle=45)) +
  xlab("Sentiment") + ylab("Total Count") + ggtitle("Total Description Sentiment Score - UNITED KINGDOM") +
  theme(plot.title = element_text(hjust = 0.5))
```

Table 5.8 showing R Code for sentiment analysis and Word cloud for YouTube Video description

Figure 5.25 Showing Word Cloud for the most used words in the video description

Channel, music, video are words that are repeated the greatest number of time in the video description. The word cloud has a huge number of words repeated when compared to the United States. The other trending words are uba videos, new, official, capture, exclusive and social networks like Twitter, Facebook, Instagram etc.

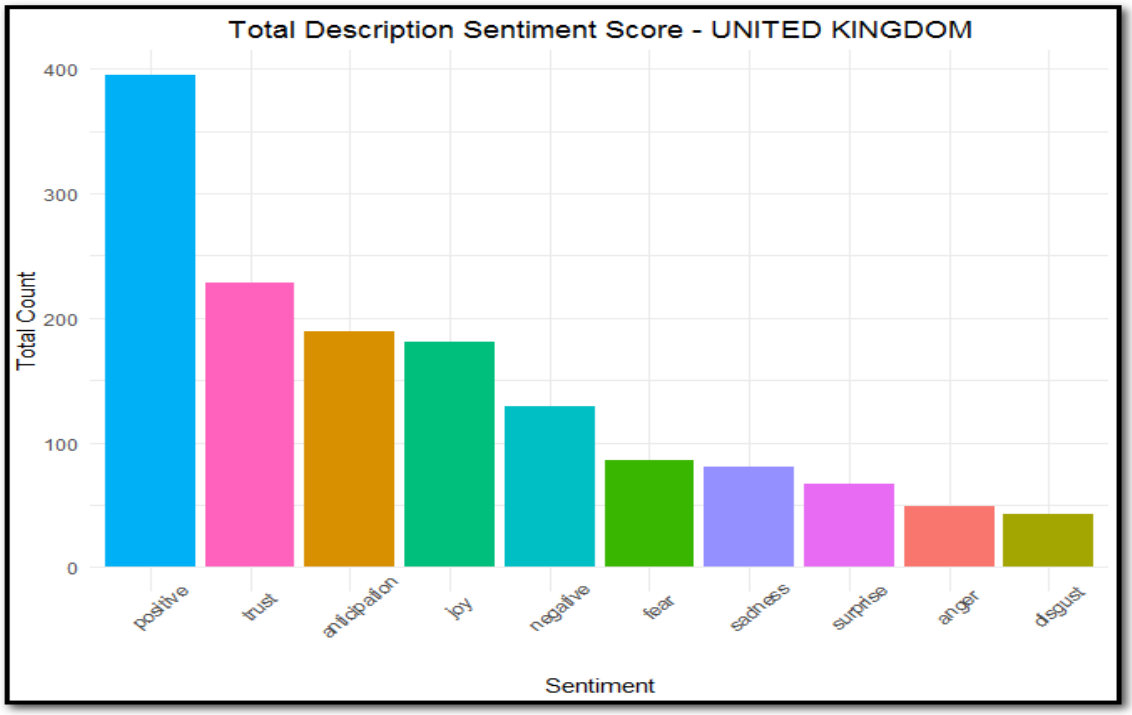


Figure 5.26 showing Sentiment bar graph based on the sentiment percentage

The above bar chart showcases the sentiment of the words used in description grouped into different anger, joy, happiness, sad etc. From the plot, we could see that there are a greater number of positive sentiments used in the video description. It is followed by trust, anticipation, and Joy. The negative sentiments and fear and sadness, anger cover the least percentage of words in the description. Hence, we could say the videos with a lot of positive sentiments have been trending in the United Kingdom when compared to the United States.

5.2.3 Sentiment Analysis on YouTube video description – Canada

```
# CANADA

ca_desc_sentiment <- get_nrc_sentiment(ca$description)
ca_desc_sentiment <- data.frame(feeling = names(colSums(ca_desc_sentiment)),
total = colSums(ca_desc_sentiment), row.names = NULL)

set.seed(1000)
corpus <- Corpus(VectorSource(list(ca$description)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))

```

```
dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
#sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L

wordcloud(sentiments$text,sentiments$freq, min.freq = 10,max.words = 300,colors = brewer.pal(6,"Dark2"),random.order = F)

ggplot(data = ca_desc_sentiment, aes(x = reorder(feeling, -total, na.rm=TRUE), y = total)) +
  geom_bar(aes(fill = feeling), stat = "identity") +
  theme_minimal()+
  theme(legend.position = "none", axis.text.x = element_text(angle=45)) +
  xlab("Sentiment") + ylab("Total Count") + ggtitle("Total Description Sentiment Score - CANADA") +
  theme(plot.title = element_text(hjust = 0.5))  # add to all plots

```

Table 5.9 showing R Code for sentiment analysis and Word cloud for YouTube Video description

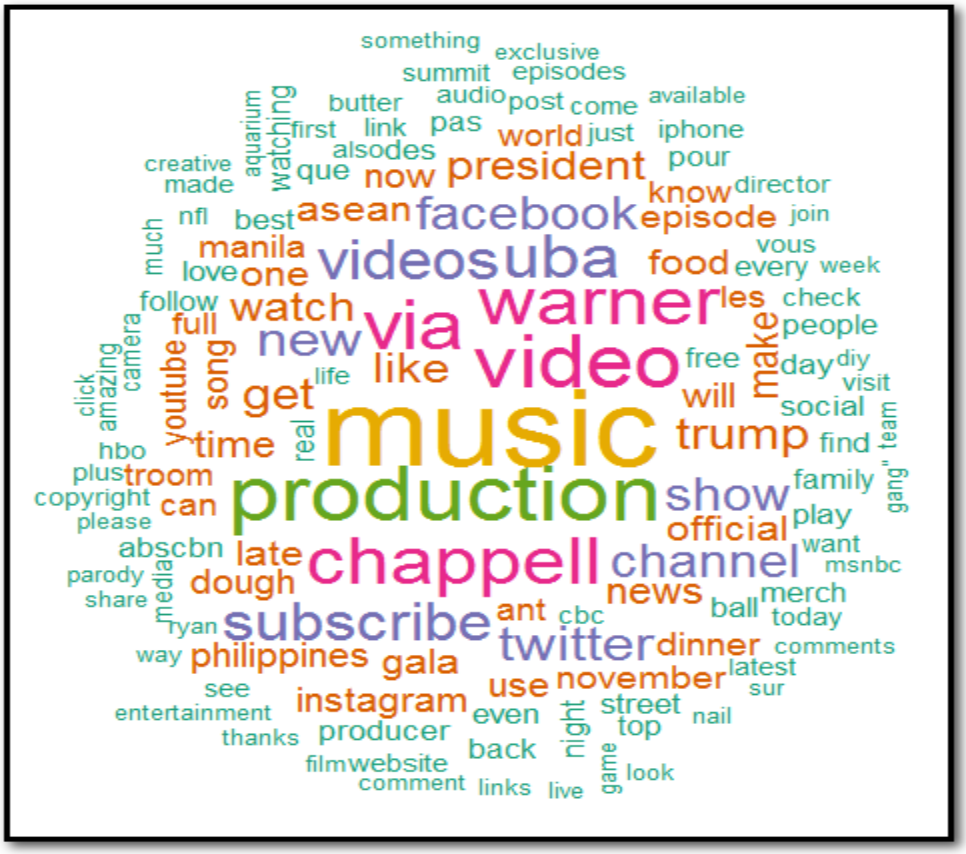


Figure 5.27 Showing Word Cloud for the most used words in the video description

Production, music, video, Chappell, Warner, videos, subscribe are words that are repeated the greatest number of times in the video description. Here since many words are repeated, the word cloud frequency has been increased to 10. The cloud has other prominent descriptions like twitter, facebook, Instagram, channel etc. Also, Trump, dinner, official are the other trending words used in the video description.

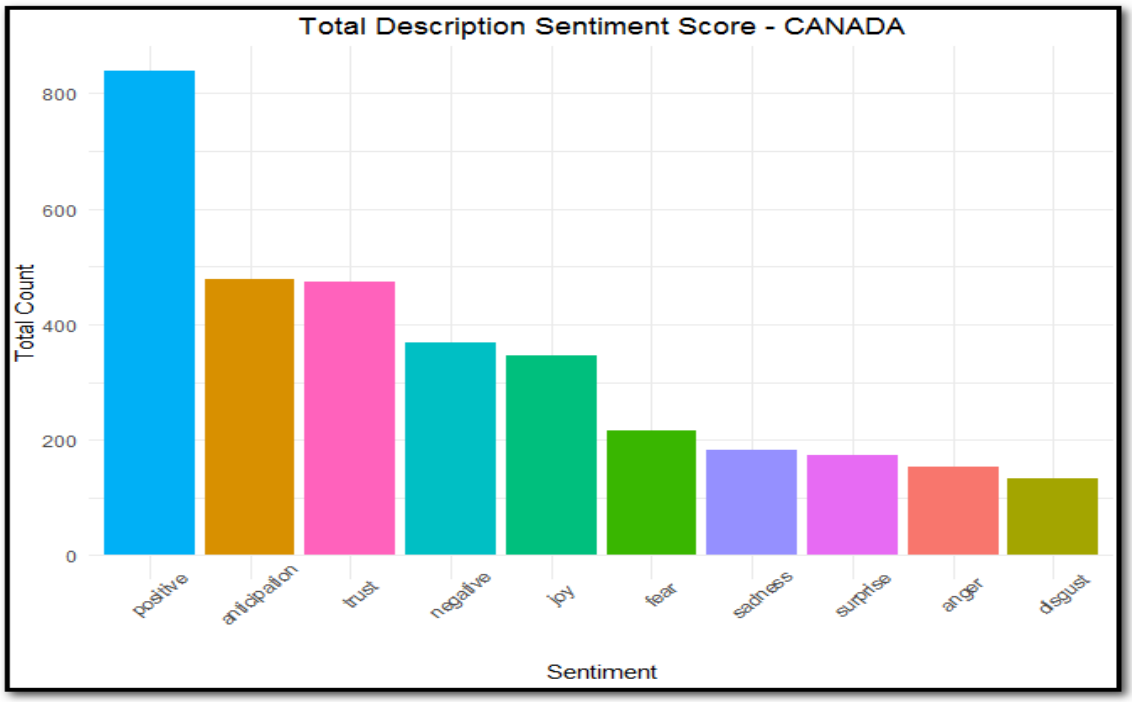


Figure 5.28 showing Sentiment bar graph based on the sentiment percentage

The video description in Canada has a huge number of positive words used. The next sentiment is anticipation and trust are sharing almost equal percentage of words which is around 480 words each, then comes the negative and joy around 375 words. The anger and disgust sentiment stand the last in Canada video description. Overall Canada video descriptions are more positive rather neutral or negative.

5.2.4 Sentiment Analysis on YouTube video description – Germany

```
# Germany

de_desc_sentiment <- get_nrc_sentiment(de$description)
de_desc_sentiment <- data.frame(feeling = names(colSums(de_desc_sentiment)),
total = colSums(de_desc_sentiment), row.names = NULL)

```

```
set.seed(1000)
corpus <- Corpus(VectorSource(list(de$description)))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))

dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
#sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L

wordcloud(sentiments$text,sentiments$freq, min.freq = 5,max.words = 300,color
s = brewer.pal(6,"Dark2"),random.order = F)


ggplot(data = de_desc_sentiment, aes(x = reorder(feeling, -
total, na.rm=TRUE), y = total)) +
  geom_bar(aes(fill = feeling), stat = "identity") +
  theme_minimal()+
  theme(legend.position = "none", axis.text.x = element_text(angle=45)) +
  xlab("Sentiment") + ylab("Total Count") + ggtitle("Total Description Sentim
ent Score - Germany") +
  theme(plot.title = element_text(hjust = 0.5))  # add to all plots
```

Table 5.10 showing R Code for sentiment analysis and Word cloud for YouTube Video description

Figure 5.29 Showing Word Cloud for the most used words in the video description

Und, auf, der, von, mit, uba, die are the words that used a greater number of times in the video description in Germany. Most of the words used in the description are German and English is very rare. This shows Germans prefer German over English in all aspects.

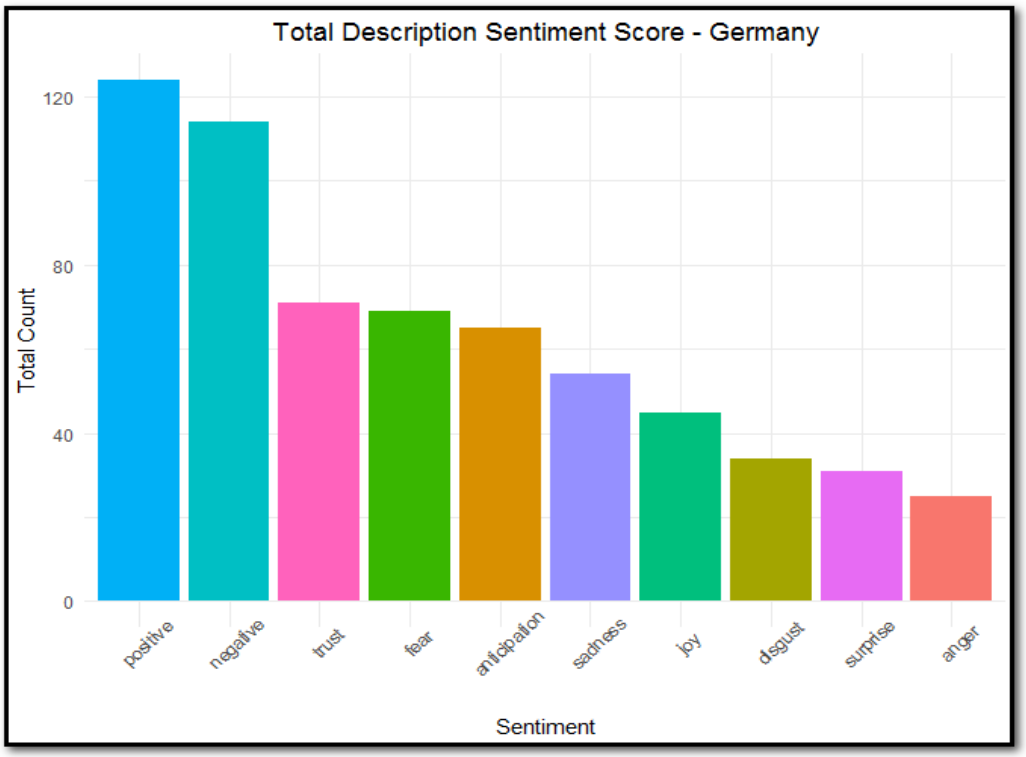


Figure 5.30 showing Sentiment bar graph based on the sentiment percentage

The video description in Germany, a huge number of positive and negative sentiments. This trend is new as most of the other regions have more of the positive sentiments. This is followed by trues, fear, anticipation. Surprise and anger the least used sentiment in the description.

5.2.5 Sentiment Analysis on YouTube video description – France

```
# FRANCE


set.seed(1000)
corpus <- Corpus(VectorSource(list(sample(fr$description,size = 10000))))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
```

```
dtm <- DocumentTermMatrix(VCorpus(VectorSource(corpus[[1]]$content)))
freq <- colSums(as.matrix(dtm))
sentiments_cal <- calculate_sentiment(names(freq))
sentiments <- cbind(sentiments_cal, as.data.frame(freq))
#sentiments[contains(match = "uu",vars = sentiments$text),"freq"] <- 0L

wordcloud(sentiments$text,sentiments$freq, min.freq = 5,max.words = 300,colors = brewer.pal(6,"Dark2"),random.order = F)
fr_desc_Sentiment <- fr$description
fr_desc_sentiment <- gsub("\\\\n", " ", fr_desc_sentiment)
fr_desc_sentiment <- gsub("http[^[:blank:]]+", "", fr_desc_sentiment)
fr_desc_sentiment <- gsub("www[^[:blank:]]+", "", fr_desc_sentiment)
fr_desc_sentiment <- gsub('[[:digit:]]+', "", fr_desc_sentiment)
fr_desc_sentiment <- gsub("[[:punct:]]+", "", fr_desc_sentiment)
fr_desc_sentiment <- gsub("\\s+"," ", fr_desc_sentiment)

fr_desc_sentiment <- get_nrc_sentiment(fr$description)
fr_desc_sentiment <- data.frame(feeling = names(colSums(fr_desc_sentiment)), total = colSums(fr_desc_sentiment), row.names = NULL)


ggplot(data = fr_desc_sentiment, aes(x = reorder(feeling, -total, na.rm=TRUE), y = total)) +
  geom_bar(aes(fill = feeling), stat = "identity") +
  theme_minimal()+
  theme(legend.position = "none", axis.text.x = element_text(angle=45)) +
  xlab("Sentiment") + ylab("Total Count") + ggtitle("Total Description Sentiment Score - FRANCE") +
  theme(plot.title = element_text(hjust = 0.5))  # add to all plots
```

Table 5.11 showing R Code for sentiment analysis and Word cloud for YouTube Video description

Figure 5.31 Showing Word Cloud for the most used words in the video description

Les, sur, pour, des, vous are the words are most predominantly used in France. The other common words are Twitter, Facebook, videos etc.

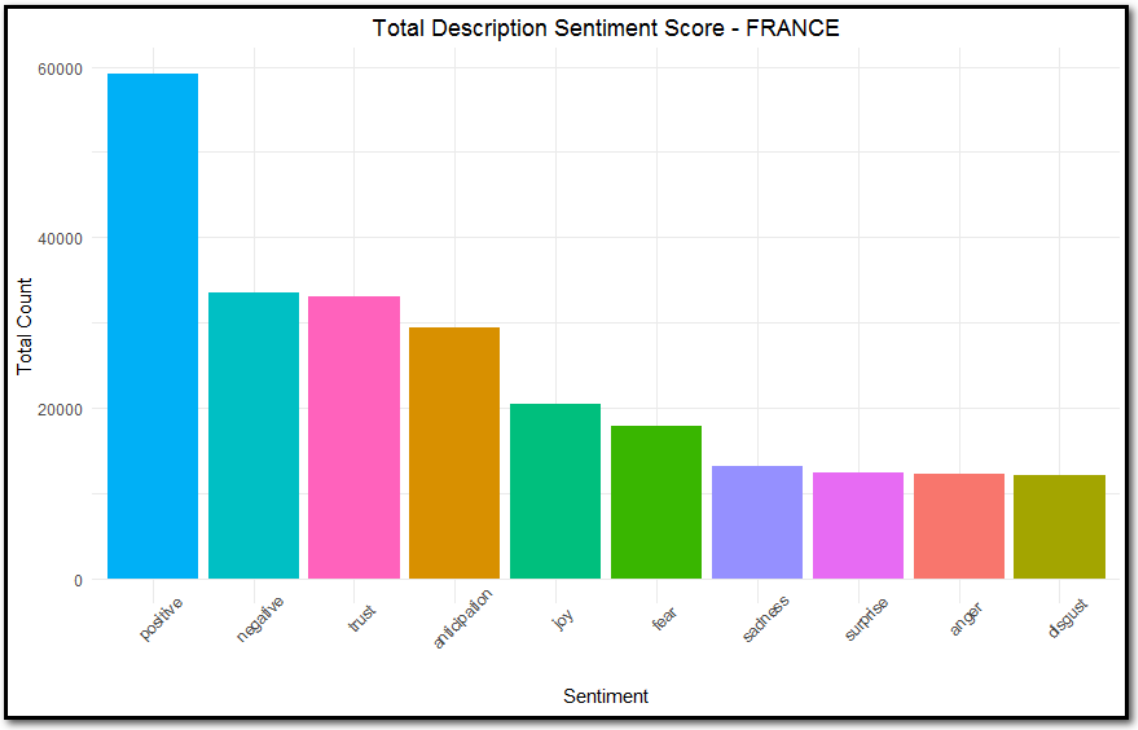


Figure 5.32 showing Sentiment bar graph based on the sentiment percentage

The video description in Germany, a huge number of positive and negative sentiments. This trend is new as most of the other regions have more of the positive sentiments. This is followed by trues, fear, anticipation. Surprise and anger the least used sentiment in the description.

5.3 Conclusions from Sentiment Analysis

From the sentiment analysis, we could see that most of the title and descriptions either uses a positive sentiment or a neutral sentiment word which have made the video to get trending. The word cloud showed us the words that were frequently used in the trending videos in most of the region. The word which predominantly used in all the videos is Subscribe, video, music, twitter, Facebook, music, Instagram etc. Thus, the sentiment analysis has been successfully completed and we have driven it through a graphically. There could be a few other analyses done like analysis of words near to subscription like social media words etc. This is not considered under the current scope of the project.

The conclusion from sentiment analysis is, the more the positive the title and description the more the chance the video is getting trended and getting a lot of views and comments. This is irrespective of region and has been proved with sentiment analysis.

6 Results and conclusion

The YouTube Trending dataset merged version contains almost 200,000 records with 13 variables and it is recorded for total 105 days. We have created few variables like trending days extra.

We started by understanding each variable in the individual dataset and then merged them to find various outcomes. We did an intense exploratory data analysis, applied correlation, and other visualization techniques and inferred a lot of interesting facts. The views count strongly correlated with comments count, likes and dislikes.

We then used the linear regression model on the variables of each individual dataset and deduced the accuracy for each model. We also used the backward regression technique and exhaustive search technique to find different model accuracy and compared the best models.

We also did sentiment analysis on the title and description of the YouTube videos each individually across dataset and in the combined dataset. The sentiment analysis on these datasets took a lot of time as the number of records were huge and the sentiment function took a lot of time to categorize the words. We had to compromise the number of records in the case of description by taking sampling in few cases as it was taking too much memory and time to execute. This is one of the drawbacks of this dataset.

Another drawback on the dataset is we did not have the actual comments of the dataset. Only the comment counts were present, and we could not perform the sentiment analysis or text mining on the actual comments. We tried to access these data from YouTube with the video title and video id but took a long time to get a result for even one video. Also, the dataset had become too bulky and we were not able to handle the load on R.

6.1 Future Work:

There could be more analysis done with this dataset like the analysis on the trending days vs published days, analysis on the tag trends, bringing images in the report using the thumbnail data provided which would have made the report more colorful. (Kaggle kernel, Ghosh, Soumya, 2018)

In the process of comparing various models for classification, we realized that predicting video category based on tags and video description given, in addition to the numerical values of data can enhance our model and its accuracy to a significant level. So, we started applying Natural Language Processing by using n-grams model with TF-IDF and Latent Semantic Analytics (SVD) to create vectors from text features like Video Description & Tags. But because of the time constraint, we couldn't complete it.

7 References

- http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software#what-is-correlation-matrix
- https://www.youtube.com/watch?v=y21yWgMWMc8
- https://www.youtube.com/watch?v=JNtcslLZYsY
- https://www.youtube.com/watch?v=3gJngOCyrZg
- https://www.kaggle.com/sgonkaggle/exploring-youtube-trending-videos-insights-eda
- https://stat.ethz.ch/R-manual/R-devel/library/base/html/strsplit.html
- https://stackoverflow.com/questions/10294284/remove-all-special-characters-from-a-string-in-r
- https://rstudio-pubs-static.s3.amazonaws.com/28038_1bcb9aa80ca84f27ace07d612872861a.html
- http://www.sthda.com/english/wiki/reordering-data-frame-columns-in-r
- https://stackoverflow.com/questions/5831794/opposite-of-in
- https://stackoverflow.com/questions/7531868/how-to-rename-a-single-column-in-a-data-frame
- http://www.r-tutor.com/r-introduction/data-frame
- https://stackoverflow.com/questions/2261079/how-to-trim-leading-and-trailing-whitespace-in-r
- https://www.datamentor.io/r-programming/operator
- http://www.sthda.com/english/wiki/scatter-plot-matrices-r-base-graphs
- https://stats.stackexchange.com/questions/57746/what-is-residual-standard-error
- https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
- https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient
- https://www.chegg.com/homework-help/definitions/slope-of-regression-line-31
- https://stackoverflow.com/questions/1330989/rotating-and-spacing-axis-labels-in-ggplot2
- https://www.kaggle.com/bansmohit/us-youtube-eda-category-predictions
- https://www.kaggle.com/dmanmeds/eda-supervised-learning-text-analysis

8 My Experience with the Project

Working on this project was a great learning experience both theoretically as well as from the practical implementation perspective. While working on the project, we applied many concepts which were taught in the class and also learnt some new concepts and algorithms along the way, which led to discover some useful and interesting insights from the datasets.

While deciding the topic and dataset for our project we found two datasets from **Kaggle**. The first one was a **Black Friday Sales** dataset and the other one was a **YouTube video statistic**. We first decided to work on Black Friday Sale dataset but in that dataset the product categories were not mentioned so the scope for classification was less and the dataset had only in-store data but today people mostly shop online. Also, lots of work has already been done on the in-store sale dataset. So, we finalized the YouTube video statistics dataset. As with the YouTube Dataset, we found more possibilities of exploration and prediction. This dataset includes several months (and counting) of data on daily trending YouTube videos. Data is included for the **US, GB, DE, CA, and FR region**s (USA, Great Britain, Germany, Canada, and France, respectively), with up to 200 listed trending videos per day. Each region's data is in a separate file. Data includes the **video title, channel title, publish time, tags, views, likes and dislikes, description, and comment count.**

Then after analysing the dataset more, we decided to work on the following areas:

1. Exploratory Data Analysis on this data set to understand the data well, which included finding correlation between different numerical variables, panel plots and predicting trending videos in various regions.
   (This part was mainly done by **Varadharajan** and **Jeongsu Han**)

2. Predicting the number of views based on other parameters such as likes, dislikes, comment count etc. by linear regression and then plotting lift chart and decile chart for more exploration and understanding for complete dataset. And we did the same for different regions of the world to get a more targeted response.
   (This was completed by **Divyashree**, **Aarya** and **Varadharajan**)

3. Applying various classification models including LDA, kNN and Classification Trees on the dataset to get insight on video category id. The goal was to predict the category of the video (music, sports, etc) using our features and then identifying the best model. I computed the confusion

matrix for each of the model and calculated the accuracy using that and I also calculated area under the curve identify the best suited model. (**I (Aarya)** completed this part of our project)

4. Performing Sentiment Analysis to identify the keywords associated with a video to be able to use them to categorize them for various regions. (**Megha** did this part of the project)

I was able to successfully execute various models and was able to interpret the statistical results. For predicting the number of views, I executed the linear regression model with the parameters likes, dislikes, comment count and category id and then plotted the lift chart and decile chart as explained in the class. Then it was continued further for different regions of the world by Divyashree.

For applying classification models and comparison, I mainly applied the concepts learnt from the class. I divided the dataset first into training dataset and validation data set. I executed all the models to predict **Video Category** based on **Likes, Dislikes, View,** and **Comment counts**. Then firstly I applied LDA model using **MASS** library. **Prior probabilities** of category id classes were too low but **proportion of trace of coefficients** of Linear Discriminant Analysis were giving good classification rates. Then secondly I applied decision (classification) tree model using libraries **rpart** and **rpart.plot** and plotted the **classification tree** and then reduced it to **best pruned tree** to remove overfitting of the data. I wanted to select a tree size that minimizes the cross-validated error, the x-error. I pruned the tree to reduce the size using **prune (fit, cp=)**. I printed out the **cp table** of cross-validation errors. The R-squared for a regression tree is 1 minus relative error. X-error (or relative cross-validation error where "x" stands for "cross") is a scaled version of overall average of the 5 out-of-sample errors across the 5 folds. Then I executed the **kNN** model with the library **ISLR** and **FNN**. Since, kNN uses Euclidean distance between two data points to find nearest neighbours, so we I normalised the data and then ran the model. Also, because our dataset was very large, the execution of kNN was bit slow but it gave the best accuracy. Finally, I made a comparison table for all the models with their accuracies and with their area under curve.

I also tried to plot **ROC curve** with **CaTools** library and **colAU**C function but because of the large number of output categories, it was not very interpretable. So, we didn't add it to our report.

The accuracies can be further improved by using the Tags and video description parameter from our dataset. So, I started learning about Natural Language Processing with R to execute it on our data set. And I learned that we can do it by

using **n-grams** model with **TF-IDF** and **Latent Semantic Analytics(SVD)** to create vectors from text features like Video Description & Tags. I used R libraries like **quanteda** for **text pre-processing** (stemming, stop-words) & creating **n-grams** and **irlba** for **SVD**. And I worked on the code but was not able to remove some of the errors and all the test cases were not executing successfully and so because of the time constraint I couldn't complete it. But this was a very important learning part of this project for me as I learnt a completely new and interesting concept and discovered some new R libraries.

Finally, we all contributed to various interpretation of the results to prepare the final report and Varadhrajan formatted it to a beautiful document. It was a great learning experience and wonderful team coordinated work. I am really fortunate to get Varadharajan, Megha , Divyashree and Jeongsu Han as my project mates. I learnt a lot from them as well.