IE 5374 Project - Sec3 - Group16 2

Fengbo Ma Ankita Yadav Zeeshan Ali Shaikh

10/12/2021

Contents

IE 5374 Foundations of Data Analytics

Project 2

Project proposal

The group discovered the data sets of YouTube Trending Video Data Analysis for Major English-spoken YouTube Countries (USA, Canada, Great Britain) which was collected using YouTube API.

The source of this dataset is kaggle. The group selected this data set as it maintains user interaction factors namely, likes, dislikes, views, comments, Trending videos and which category it falls in.

While there are numerous datasets available online, YouTube dataset was chosen as it has all the required attributes which satisfies project requirements. It has likes, dislikes, and ..... to perform probability mass function, cumulative distribution functions. Tags can be used to predict the emotions using Text mining.The data set has dates and time which can be taken into account for generating the trends using Time series analysis.

The final data set chosen Major English-spoken YouTube Countries (USA, Canada, Great Britain) only because the data and language varies from region to region which makes it complex to analyse as well as the group is not familiar with the languange to verify the results. The data set will be more reliable, organized, and convenient to use which not only includes information about the most liked, disliked, viewed, commented, and the top performers in a specific category/genre video but answers the business questions through visualizations.

Data Acquisition

Data acquisition is the process of collecting and acquiring data from disparate source systems that capture the real-world physical phenomena and converting them into a digital form that can be manipulated by a computer and software.

Data Wrangling

```
# Import package used in the entire project
library(tidyr)
library(dplyr)
library(magrittr)
library(stringr)
library(lubridate)
library(lemon)
library(knitr)
library(ggplot2)
library(gridExtra)
library(tidyverse)
library(gapminder)
library(highcharter)
library(cluster)
library(factoextra)
library(tidytext)
library(wordcloud)
library(reshape2)
```

```
library(prob)
library(RColorBrewer)
library(Rtsne)
library(nonlinearTseries)
```

Discovering

In this step, we explore the dataset, find important trends or missing values, and conceptualize in order to use it.

```
# Import data for Canada
dfCA = read_csv("CA_youtube_trending_data.csv")

# Import data for Great Britain
dfGB = read_csv("GB_youtube_trending_data.csv")

# Import data for USA
dfUS = read_csv("US_youtube_trending_data.csv")

# Import data for YouTube API breaking catgory ID
dfCID = read_csv("Category_list.csv")
```

Import data

Summary Statisics In this section, we check all the column names in the data frames for Canada(dfCA), Great Britain(dfGB), and USA(dfUS) from which we extract the unique carrier id of flights and check whether the data frames consist of NA values, and displaying the number of NA values in each column.

```
# Display column name for all the data frame
colnames(dfCA)
```

```
##  [1] "video_id"          "title"             "publishedAt"
##  [4] "channelId"         "channelTitle"      "categoryId"
##  [7] "trending_date"     "tags"              "view_count"
## [10] "likes"             "dislikes"          "comment_count"
## [13] "thumbnail_link"    "comments_disabled" "ratings_disabled"
## [16] "description"
```

```
colnames(dfGB)
```

```
##  [1] "video_id"          "title"             "publishedAt"
##  [4] "channelId"         "channelTitle"      "categoryId"
##  [7] "trending_date"     "tags"              "view_count"
## [10] "likes"             "dislikes"          "comment_count"
## [13] "thumbnail_link"    "comments_disabled" "ratings_disabled"
## [16] "description"
```

```
colnames(dfUS)
```

```
##  [1] "video_id"          "title"             "publishedAt"
##  [4] "channelId"         "channelTitle"      "categoryId"
##  [7] "trending_date"     "tags"              "view_count"
## [10] "likes"             "dislikes"          "comment_count"
## [13] "thumbnail_link"    "comments_disabled" "ratings_disabled"
## [16] "description"
```

```
# First time checking for NA variable
kable(
  dfCA %>%
    summarise_all(funs(sum(is.na(.)))) %>%
    t()
,caption="Check for NA Data Frame CA")
```

Table 1: Check for NA Data Frame CA

video_id	0
title	0
publishedAt	0
channelId	0
channelTitle	0
categoryId	0
trending_date	0
tags	0
view_count	0
likes	0
dislikes	0
comment_count	0
thumbnail_link	0
comments_disabled	0
ratings_disabled	0
description	1731

```
kable(
  dfGB %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
  ,caption="Check for NA Data Frame GB")
```

Table 2: Check for NA Data Frame GB

video_id	0
title	0
publishedAt	0
channelId	0
channelTitle	0

categoryId	0
trending_date	0
tags	0
view_count	0
likes	0
dislikes	0
comment_count	0
thumbnail_link	0
comments_disabled	0
ratings_disabled	0
description	1680

```
kable(
  dfUS %>%
    summarise_all(funs(sum(is.na(.))))%>%
    t()
,caption="Check for NA Data Frame US")
```

Table 3: Check for NA Data Frame US

video_id	0
title	0
publishedAt	0
channelId	0
channelTitle	0
categoryId	0
trending_date	0
tags	0
view_count	0
likes	0
dislikes	0
comment_count	0
thumbnail_link	0
comments_disabled	0
ratings_disabled	0
description	1538

The result could clearly indicate that in the above tables there are a couple of columns which do not have NA values except for description. This is neglected as we do not have any analysis to be performed on this column.

Discovering Discrete Data In this step, we determine the total number of records of data which is available for in each csv file of all three datasets using the TOTAL_RECORDS attribute.

```
# Count for max number row in the data set using summarize
# function rather than dimension function
# Summarize function provide only one out out that we need
kable(
```

```
  dfCA %>%
    summarize(TOTAL_RECORDS=n()),
caption="Number of rows in CA Dataset")
```

Table 4: Number of rows in CA Dataset

TOTAL_RECORDS
95944

```
kable(
  dfGB %>%
    summarize(TOTAL_RECORDS=n()),
caption="Number of rows in GB Dataset")
```

Table 5: Number of rows in GB Dataset

TOTAL_RECORDS
95995

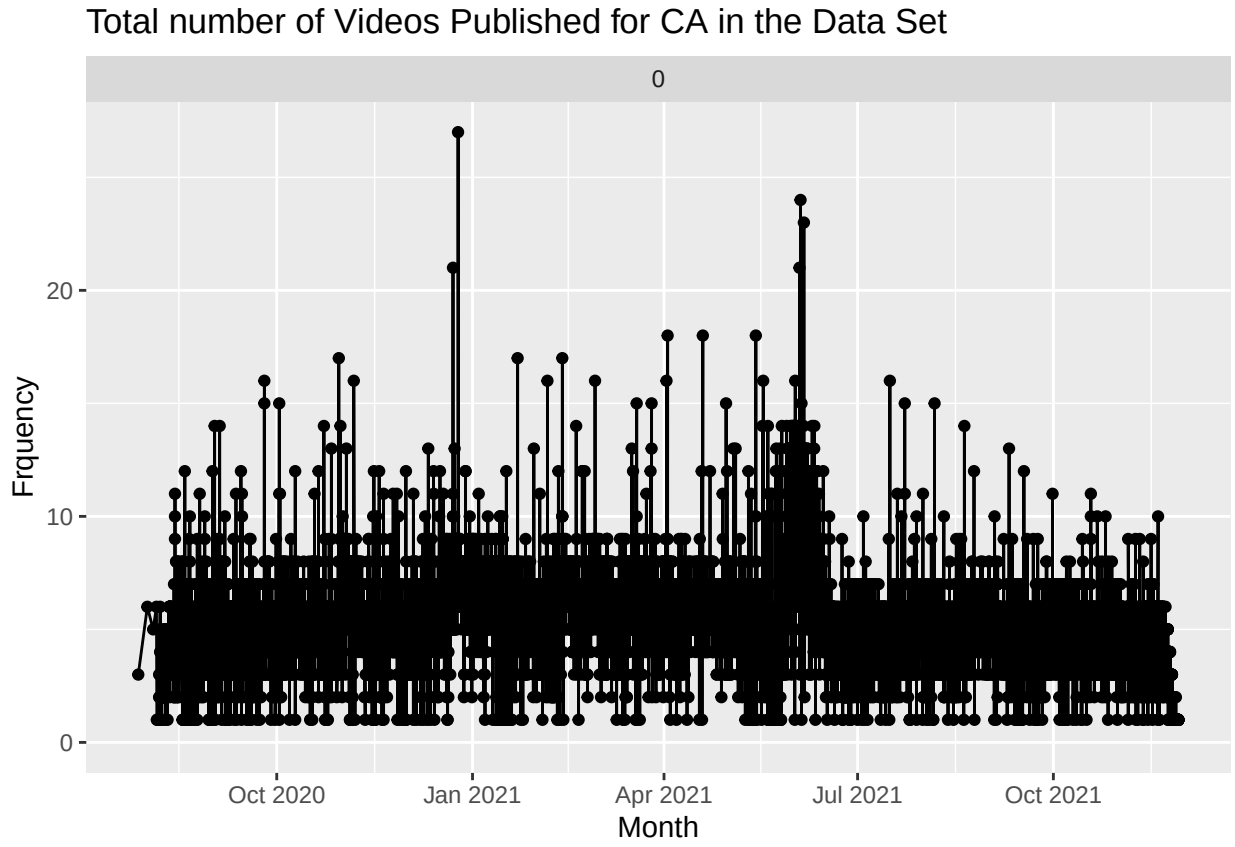
```
kable(
  dfUS %>%
    summarize(TOTAL_RECORDS=n()),
caption="Number of rows in US Dataset")
```

Table 6: Number of rows in US Dataset

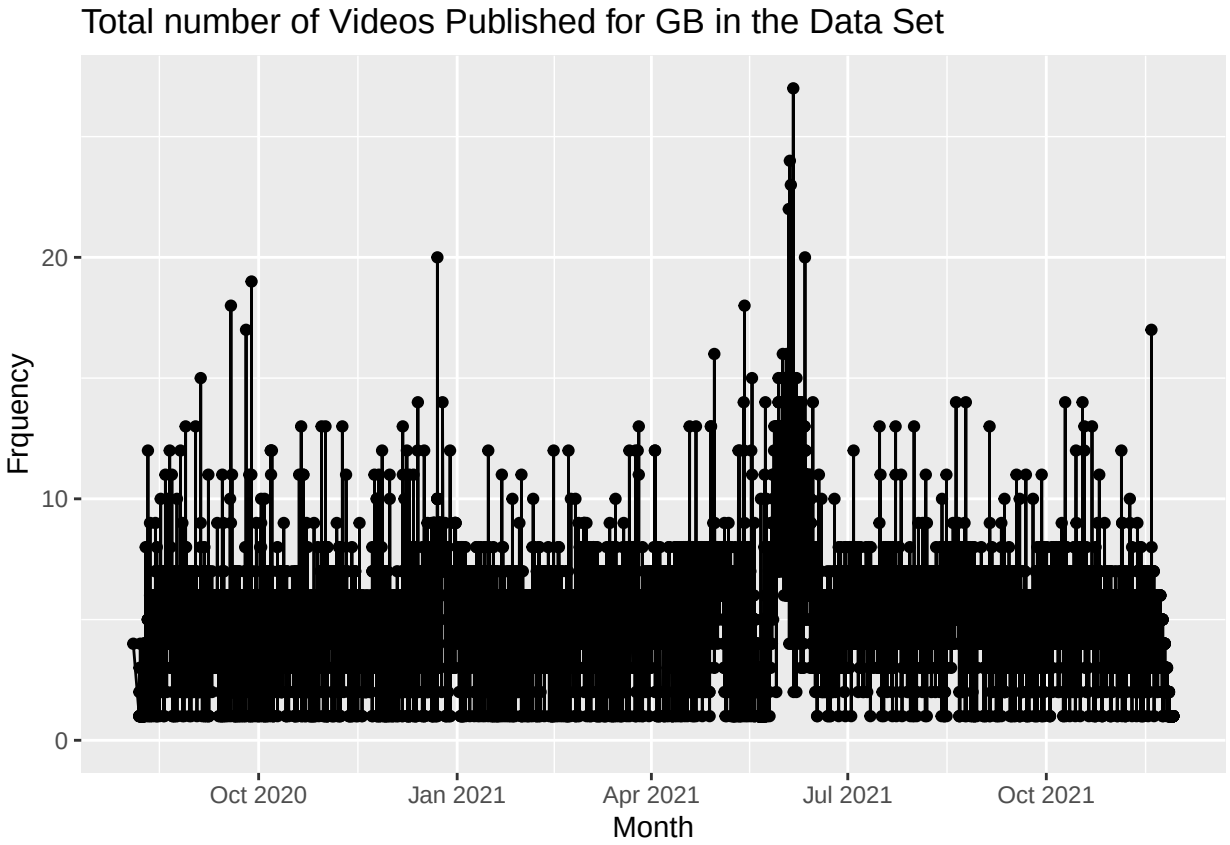
TOTAL_RECORDS
95991

Data Discovering using Visualizations Displaying the total number of videos published on the Youtube using line chart for all three countries.

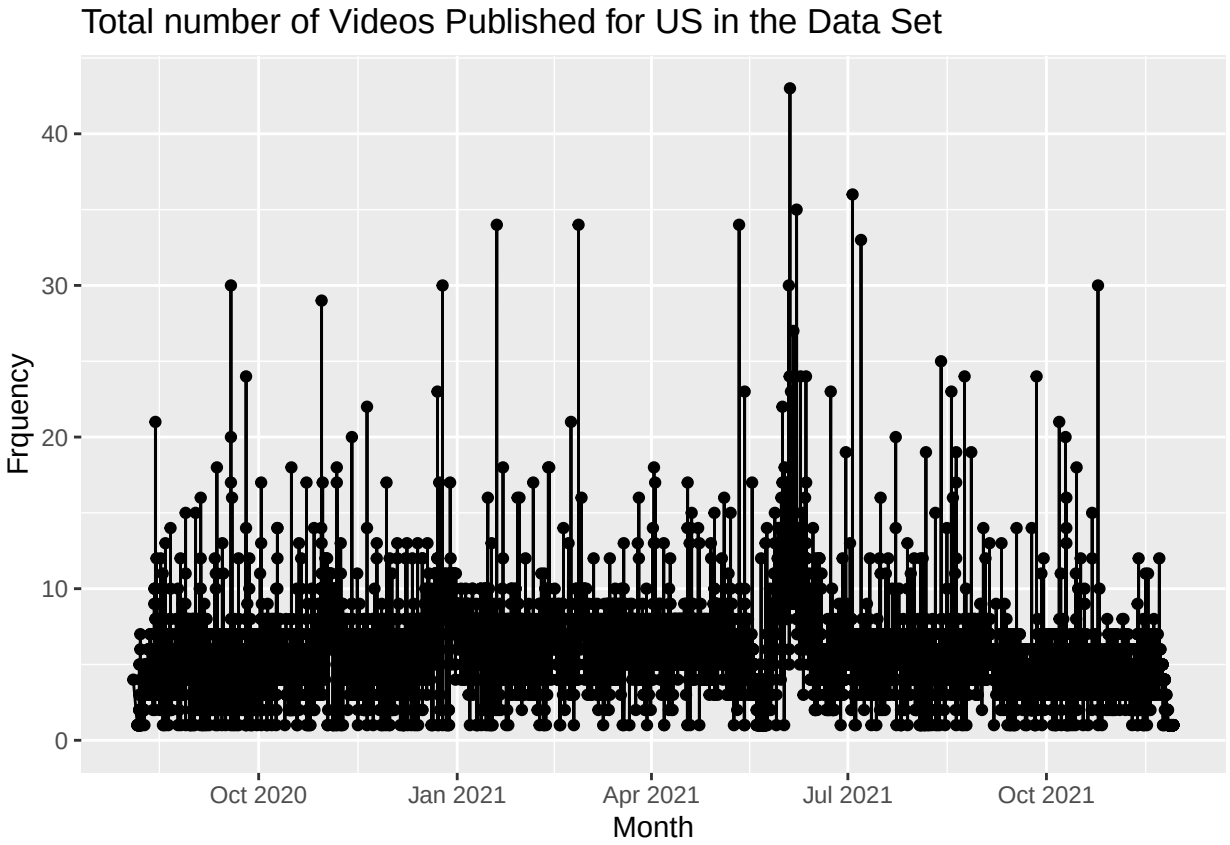
```
# Data discovering using visualizations
# Display the videos throughout the year.

# plot line chart for CA
dfCA %>%
  count(publishedAt) %>%
  ggplot(aes(x=publishedAt, y=n)) +
  geom_line(stat = "identity") +
  ggtitle("Total number of Videos Published for CA in the Data Set")+
  geom_point()+
  expand_limits(y=0)+
  facet_wrap(~y,scales=("free_x"))+
  labs(x="Month", y="Frquency")
```




```
# plot line chart for GB
dfGB %>%
  count(publishedAt) %>%
  ggplot(aes(x=publishedAt, y=n)) +
  geom_line(stat = "identity") +
  ggtitle("Total number of Videos Published for GB in the Data Set")+
  geom_point()+
  expand_limits(y=0)+
  labs(x="Month", y="Frquency")
```




```
# plot line chart for US
dfUS %>%
  count(publishedAt) %>%
  ggplot(aes(x=publishedAt, y=n)) +
  geom_line(stat = "identity") +
  ggtitle("Total number of Videos Published for US in the Data Set")+
  geom_point()+
  expand_limits(y=0)+
  labs(x="Month", y="Frquency")
```




```
#Creating Column based on the Country name using mutate function.
dfCA = dfCA %>%
  mutate(
    country  = "CA"
  )

dfGB = dfGB %>%
  mutate(
    country  = "GB"
  )

dfUS = dfUS %>%
  mutate(
    country  = "US"
  )

#Binding all the four dataset into one large dataset.
df = rbind(dfCA, dfGB,dfUS)

rm(dfCA)
rm(dfGB)
rm(dfUS)
```

Convert String to Dates (Date Change)/Time Format Change Earlier we had a single column for date and time called publishedAt. Hence, the group sepearted it into two other columns named Pub-

lished.date and Published.time respectively using mutate function. The Trending.Date and Published.date are converted into "YYYY-MM-DD" format whereas the Published.time is displayed in "HH:MM:SS" format.

```
#Using the mutate function to create a new columns for date time and months.
#as.date for formatting the to convert them into desired format.

df = df %>%
  mutate(df,Published.date = as.Date(df$publishedAt,"%Y/%M/%D"))%>%
  mutate(df,Published.time = format(df$publishedAt,"%H:%M:%S"))%>%
  mutate(df,Trending.Date = as.Date(df$trending_date, "%Y/%M/%D"))

#Converting for Trending date.
df= df %>%
  mutate(Trending.Date = ymd(trending_date)) %>%
  mutate_at(vars(Trending.Date), funs(year, month, day))

#Converting for Trending date.
df = df %>%
  mutate(date = ymd(Published.date)) %>%
  mutate_at(vars(Published.date), funs(year, month, day))
```

Cleaning

This step includes deleting empty rows or cells, fixing NA values, examining outliers.

Remove NA (Fix missing data) In our data sets, we had the last column filled with NA's, so we removed them and checked again whether there is any NA's left or not, in the other half of the code, we replaced all the NA's by '0' so that we don't lose our data from the data set. At the end of this section, we have displayed the result of values with no NA's in the data set.

```
#Removing all possible NA values from the dfCID table using na.omit function.

dfCID = dfCID %>%
  select(categoryId, categoryname) %>%
  na.omit()

dfCID
```

```
## # A tibble: 17 x 2
##    categoryId categoryname
##         <dbl> <chr>
##  1          1 Film & Animation
##  2          2 Cars & Vehicles
##  3         10 Music
##  4         15 Pets & Animals
##  5         17 Sport
##  6         18 Short Movies
##  7         19 Travel & Events
##  8         20 Gaming
##  9         21 Videoblogging
## 10         22 People & Blogs
## 11         23 Comedy
```

```
## 12         24 Entertainment
## 13         25 News & Politics
## 14         26 How-to & Style
## 15         27 Education
## 16         28 Science & Technology
## 17         29 Non-profits & Activism
```

```
#---------------------------------------------
sort(unique(df$categoryId))
```

```
##  [1]  1  2 10 15 17 19 20 22 23 24 25 26 27 28 29
```

```
length(sort(unique(df$categoryId)))
```

```
## [1] 15
```

Enriching

This involves incorporating data from other datasets and removing the irrelevant ones for easy data manipulation.

Add Category Add extra column for identify the real meaning of categoryId and perform analysis based on it later in the project.

```
#Adding the dfCID dataset to the df dataset using the join function.
df = left_join(df,dfCID)
```

Validating

In this step, we verify whether the data is of high quality, dependable, and logical.

Checking NA value After joining the datasets there is a possibility that our data set might again have redundancies, so we again check for NA values using the summarize function.

```
# Re-run NA checks

kable(
  df %>%
    summarise_all(funs(sum(is.na(.)))) %>%
    t()
,caption="NA check for the data set")
```

Table 7: NA check for the data set

video_id	0
title	0
publishedAt	0

channelId	0
channelTitle	0
categoryId	0
trending_date	0
tags	0
view_count	0
likes	0
dislikes	0
comment_count	0
thumbnail_link	0
comments_disabled	0
ratings_disabled	0
description	4949
country	0
Published.date	0
Published.time	0
Trending.Date	0
year	0
month	0
day	0
date	0
categoryname	0

As we do not have any analysis to be performed on the description column, we neglect it. Hence, all the other columns except for description do not contain any NA values.

Checking Date Type Double checking data type for dates. Select everything is not in date format.

```
# Checking for Published date.
kable(
  df %>%
    dplyr::filter(is.Date(Published.date)==FALSE) %>%
    select(Published.date)
  ,caption="Checking Date format for Published date"
)
```

Table 8: Checking Date format for Published date

Published.date

```
# Checking for Trending.Date

kable(
  df %>%
    dplyr::filter(is.Date(Trending.Date)==FALSE) %>%
    select(Trending.Date)
,caption="Checking Date format for Trending date"
)
```

Table 9: Checking Date format for Trending date

Trending.Date

The empty values means that the dates are in the correct format.

Check Duplication There's a possibility that a YouTube video with the same video id and title is available, hence, in this section, we are trying to check for duplication and list them down for the same. Check duplication by group by every possible categories.

```
# Use group by + count to find if everything is duplicate
# Check n for count

x = duplicated(df)
y = TRUE
y %in% x
```

```
## [1] TRUE
```

```
kable(
  y %in% x
   , caption = "Any TRUE in Duplication Check? ")
```

Table 10: Any TRUE in Duplication Check?

x
TRUE

```
df = distinct(df)

A = df %>%
  group_by_all() %>%
  count(sort = TRUE) %>%
  ungroup() %>%
  select(video_id, title, n) %>%
  head(10)

kable(
  A
  ,caption = "Check for Duplications" )
```

Table 11: Check for Duplications

video_id	title	n
–0bCF-iK2E	Jadon Sancho • Magical Skills & Goals	1
–0bCF-iK2E	Jadon Sancho • Magical Skills & Goals	1
–0bCF-iK2E	Jadon Sancho • Magical Skills & Goals	1

video_id	title	n
–0bCF-iK2E	Jadon Sancho • Magical Skills & Goals	1
–0bCF-iK2E	Jadon Sancho • Magical Skills & Goals	1
–14w5SOEUs	Migos - Avalanche	1
–14w5SOEUs	Migos - Avalanche	1
–14w5SOEUs	Migos - Avalanche (Official Video)	1
–14w5SOEUs	Migos - Avalanche (Official Video)	1
–14w5SOEUs	Migos - Avalanche (Official Video)	1

```
rm(A, x, y)
```

Probability

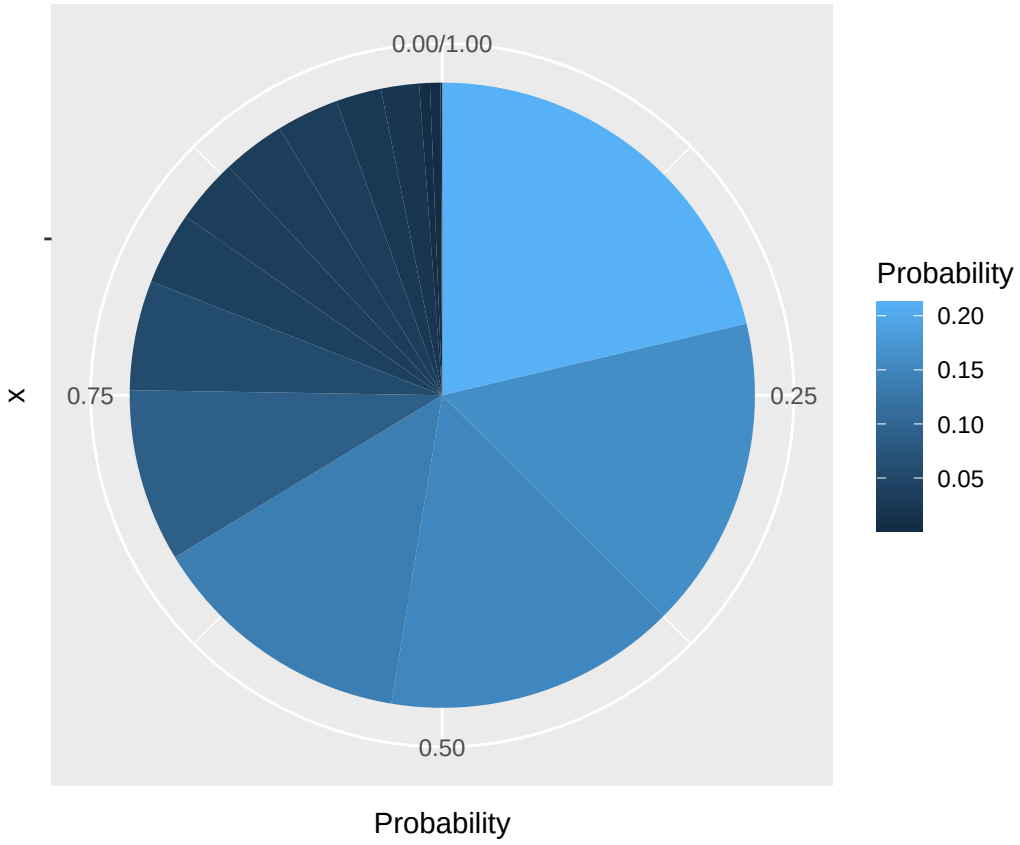
1. What are the most popular trending categories? PMF/ CDF

```
# Analysis trending categories
# Counting published categories

A = df %>%
  select (title, categoryId) %>%
  count(categoryId, sort = TRUE)

#Left join of dfCID on A

B = left_join(A,dfCID)

# Mutate a new column of Probability and calculating CFD of it

B = B %>%
  mutate(
    Probability = n / (sum(n)))%>%
  mutate(
    cdf = cumsum(Probability))
B
```

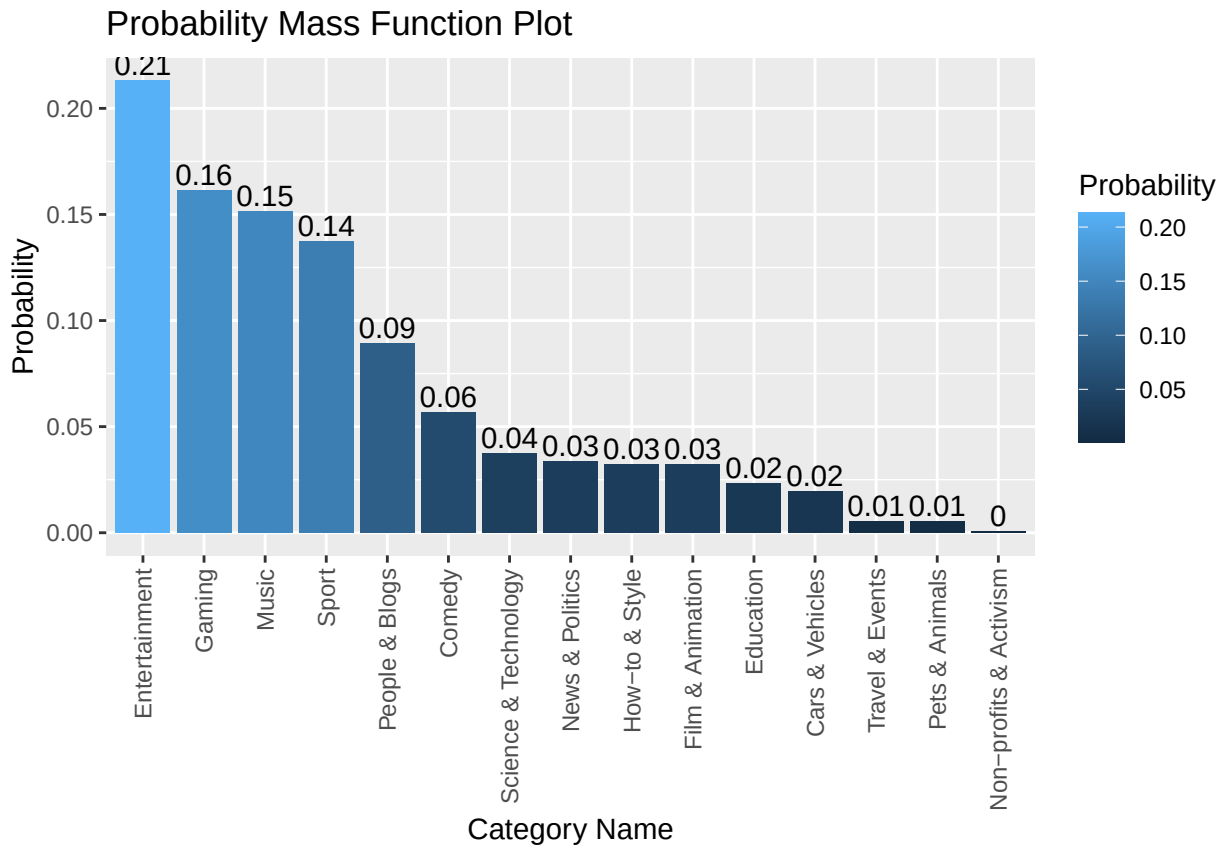
```
## # A tibble: 15 x 5
##    categoryId     n categoryname           Probability   cdf
##         <dbl> <int> <chr>                        <dbl> <dbl>
##  1         24 61321 Entertainment              0.213    0.213
##  2         20 46458 Gaming                     0.162    0.375
##  3         10 43526 Music                      0.151    0.526
##  4         17 39465 Sport                      0.137    0.663
##  5         22 25683 People & Blogs             0.0893   0.753
##  6         23 16305 Comedy                     0.0567   0.809
##  7         28 10796 Science & Technology       0.0375   0.847
##  8         25  9627 News & Politics            0.0335   0.880
##  9         26  9332 How-to & Style             0.0324   0.913
## 10          1  9299 Film & Animation           0.0323   0.945
## 11         27  6726 Education                  0.0234   0.968
## 12          2  5661 Cars & Vehicles            0.0197   0.988
```

```
## 13          19  1586 Travel & Events            0.00551  0.994
## 14          15  1568 Pets & Animals             0.00545  0.999
## 15          29   261 Non-profits & Activism     0.000907 1
```

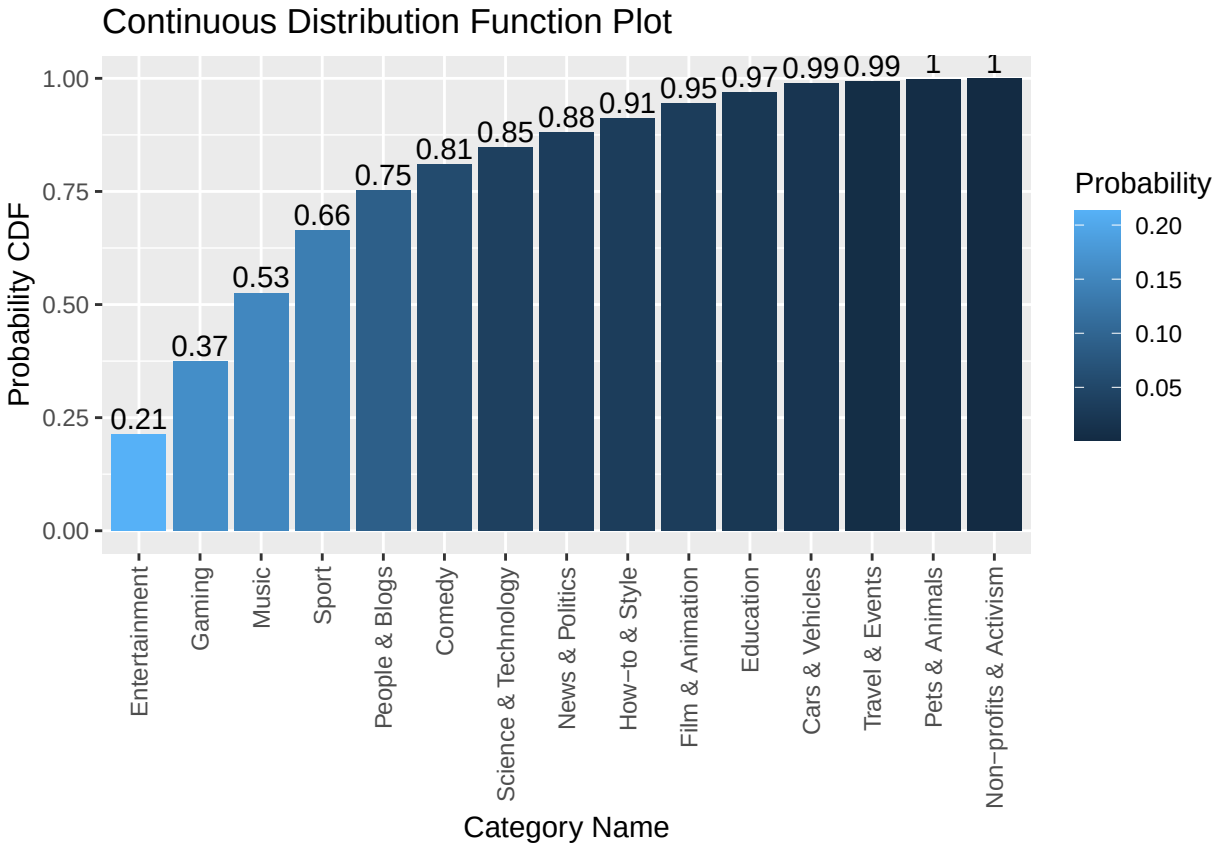
```
#Plotting a graph

B %>%
  ggplot(aes(x="", y=`Probability`, fill = Probability))+
  geom_bar(width = 1, stat = "identity")+
  coord_polar("y", start=0)
```




```
##Plotting a graph

B %>%
  ggplot(aes(x=reorder(`categoryname`,-`Probability`), y=`Probability`,
             fill= `Probability`)) +
    geom_bar(stat="identity")+
    ggtitle("Probability Mass Function Plot ")+
    labs(x= "Category Name", y= "Probability")+
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
    geom_text(aes(label=round(Probability,2)), position=position_dodge(width=0.9), vjust=-0.25)
```




```
##Plotting a graph

B %>%
  ggplot(aes(x=reorder(`categoryname`,`cdf`), y=`cdf`,
             fill= `Probability`)) +
    geom_bar(stat="identity")+
    ggtitle("Continuous Distribution Function Plot ")+
    labs(x= "Category Name", y= "Probability CDF")+
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
    geom_text(aes(label=round(cdf,2)),
              position=position_dodge(width=0.9), vjust=-0.25)
```




```
#-----------------------------------------------
#Removing the variable A and B from environment
rm(A, B)
```

The group created another dataset which had the corresponding category names and merged it with the category id which was available in the original dataset. Using these category names, the PMF and CDF plot was generated using the ggplot function.

From the first bar graph, it can be clearly seen that PMF for Entertainment category was the highest whereas the Non-profits & Activism videos had almost negligible PMF. From the second bar graph, it can be inferred that Pets & Animals as well as Non-profits & Activism have the highest CDF whereas the Entertainment Category has the lowest.

2. What is probability of getting a higher like (>10000) if getting comment disabled? Will disabling comment function brings the video more likes?

First, we filter out all the videos which has likes more than 10000, calculate its joint probability and count the number of videos with comment enabled and comment disabled.

```
#Counting the Total number of records
a=df %>%
  select(likes,categoryname) %>%
   summarise(records_og = n())

#Counting the values of number of records greater than 10000.
```

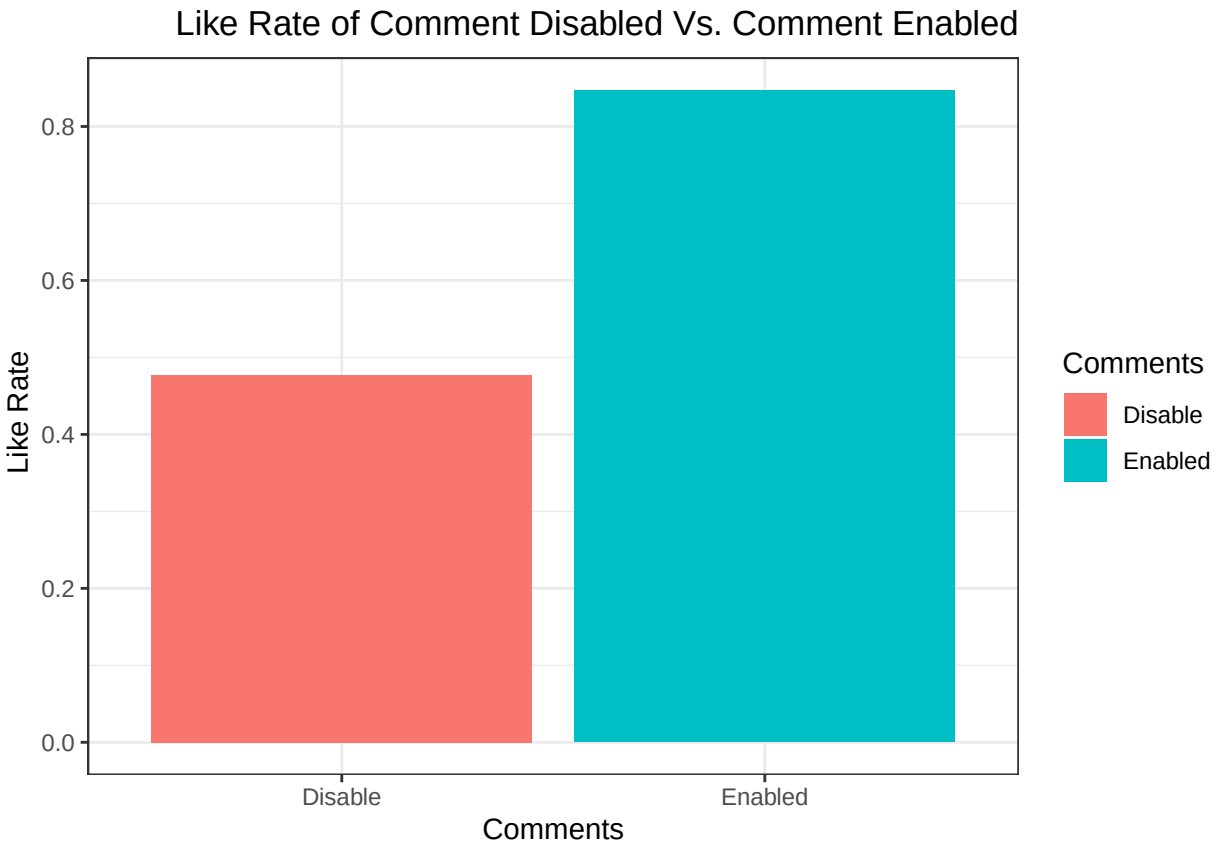
```
df1 <- df %>%
  # dplyr::filter(comments_disabled == FALSE) %>%
  select(likes,categoryname) %>%
  dplyr::filter(likes > 10000 ) %>%
  summarise(records = n())

#Finding the probablity
prob= df1/a

#Counting the Total number of records
a1=df %>%
  select(likes,categoryname) %>%
   summarise(records_og = n())

#Counting the number of records which have value greater than
#10000 and their comment disabled.
df2 <- df %>%
  dplyr::filter(comments_disabled == TRUE) %>%
  select(likes,categoryname) %>%
  dplyr::filter(likes > 10000 ) %>%
  summarise(records = n())

prob1= df2/a1

#Joint probability.
a3=prob*prob1

#For plotting we now find the values greater than 10000 of comments
#disabled and count them upon the total number of comments disabled.
df3 <- df %>%
  dplyr::filter(comments_disabled == TRUE) %>%
  select(likes,categoryname) %>%
  dplyr::filter(likes > 10000 ) %>%
  summarise(records = n())
df4 <- df %>%
  dplyr::filter(comments_disabled == TRUE) %>%
  summarise(records = n())

df5 <- df3/df4

#Comments Enabled, filtering out likes which are greater than
#10000 and summarising the records.

df6 <- df %>%
  dplyr::filter(comments_disabled == FALSE) %>%
  select(likes,categoryname) %>%
  dplyr::filter(likes > 10000 ) %>%
  summarise(records = n())
df7 <- df %>%
  dplyr::filter(comments_disabled == FALSE) %>%
  summarise(records = n())

df8 <- df6/df7
```

```
#Creating a data frame for creating a plot
zz <- as.data.frame(c(df5,df8))
colnames(zz) <- c("Disable","Enabled")
rownames(zz) <- c("Prob")
zz <- gather(zz,Prob, factor_key=TRUE)
colnames(zz) <- c('Comments','Prob')

kable(zz,
      caption = "Like Rate (Comment Disabled vs. Enabled")
```

Table 12: Like Rate (Comment Disabled vs. Enabled

Comments	Prob
Disable	0.4771777
Enabled	0.8469564

```
#Plotting the graph
zz%>%
  ggplot(aes(x=Comments, y=Prob,
             fill= Comments))+
  geom_bar(stat = "identity")+
  theme_bw() +
  labs( y = ' Like Rate') +
  theme(plot.title = element_text(hjust = 1.0)) +
  ggtitle("Like Rate of Comment Disabled Vs. Comment Enabled")
```



From the bar graphs, the probability of getting a higher like (>10000) for videos with Comments Enabled is almost 50% more than the videos with Comments Disabled.

3. What kinds of videos are having more chances of being liked/disliked/commented?

```
#Joining the df and dfCID
df = left_join(df,dfCID)

#Grouping by likes and categoryname, counting them in descending order
#and showing the 1st outcome
kable(
  df %>%
    group_by(likes,categoryname) %>%
    count(sort = TRUE) %>%
    head(1)
  ,caption="Checking for Likes Video according to Categories")
```

Table 13: Checking for Likes Video according to Categories

likes	categoryname	n
0	People & Blogs	329

```
#Grouping by likes and categoryname, counting them in descending order
#and showing the 1st outcome
kable(
  df %>%
    group_by(dislikes,categoryname) %>%
    count(sort = TRUE) %>%
    head(1)
  ,caption="Checking for Disliked Video according to Categories")
```

Table 14: Checking for Disliked Video according to Categories

dislikes	categoryname	n
0	People & Blogs	329

```
#Grouping by likes and categoryname, counting them in descending order
#and showing the 1st outcome, where the comments is enabled
kable(
  df %>%
    dplyr::filter(comments_disabled == FALSE) %>%
    group_by(categoryname) %>%
    summarise(records = n()) %>%
    arrange(desc(records)) %>%
    head(1)
  ,caption="Checking for Disliked Video according to Categories")
```

Table 15: Checking for Disliked Video according to Categories

categoryname	records
Entertainment	59561

```
#Grouping by likes and categoryname, counting them in descending
#order and showing the 1st outcome, where the comments is disabled
kable(
  df %>%
    dplyr::filter(comments_disabled == TRUE) %>%
    group_by(categoryname) %>%
    summarise(records = n()) %>%
    arrange(desc(records)) %>%
    head(1)
  ,caption="Checking for Disliked Video according to Categories")
```

Table 16: Checking for Disliked Video according to Categories

categoryname	records
Entertainment	1760

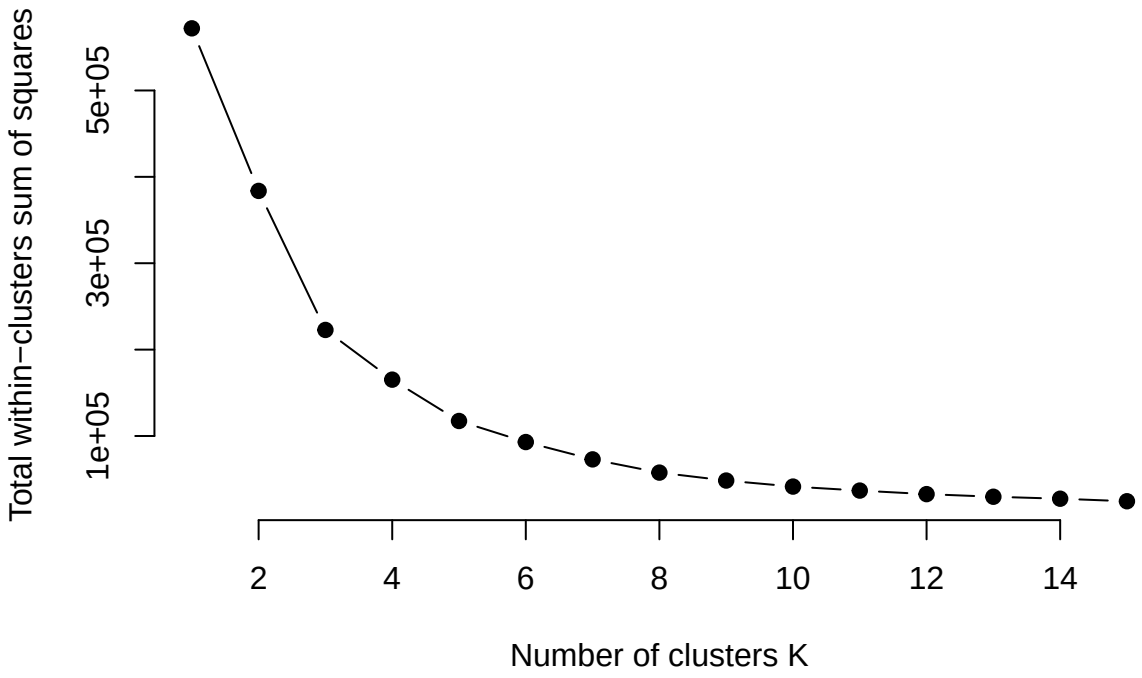
Clustering

4. Analysis relationsips between number of views and like rate.

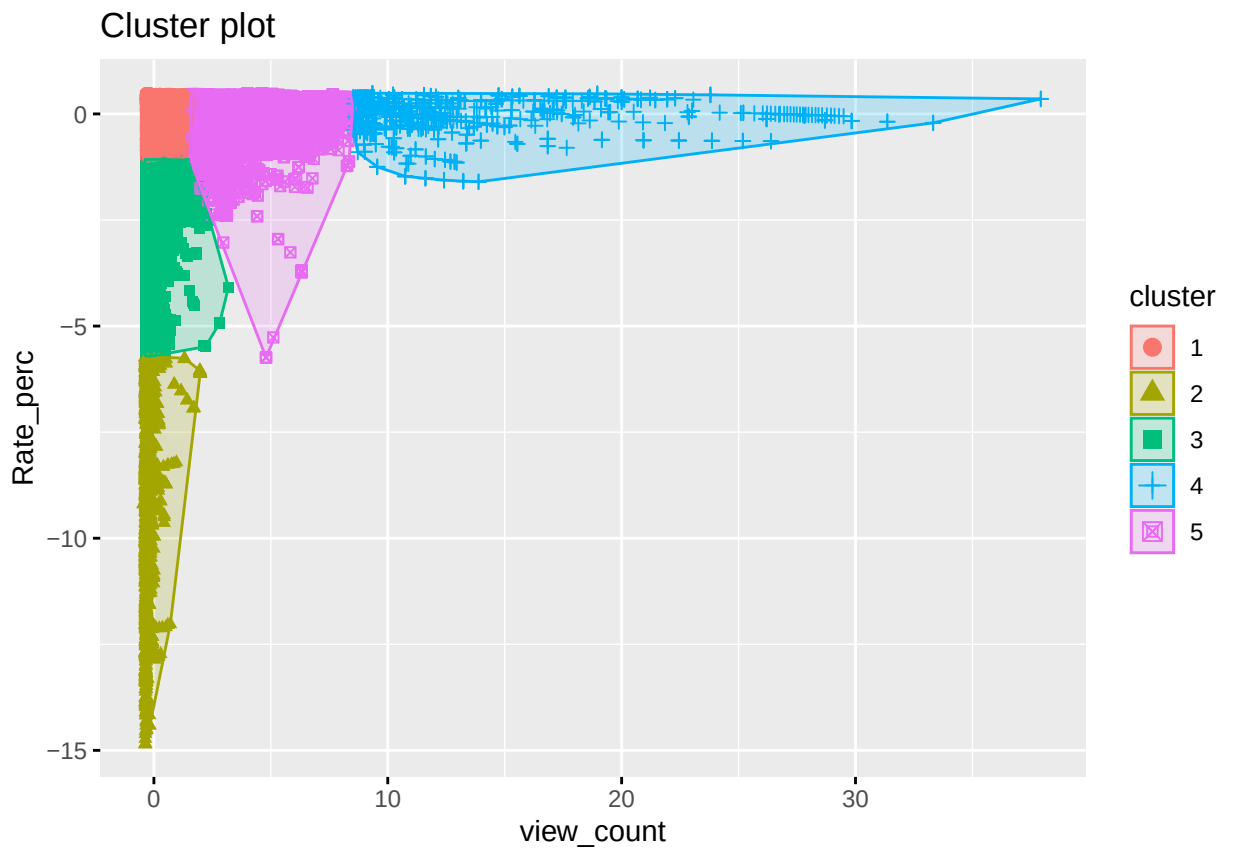
Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups.

In this section, we used k-means clustering in order to calculate the relation between the view counts and the Like-Dislike rate.

```
#Creating clusters from the data set using the select, mutate,
#and mathematical function.
dfclustertest = df [df$ratings_disabled != TRUE,]
dfclustertest = dfclustertest %>%
  select(view_count,likes, dislikes) %>%
  mutate (
    Like_rate = likes/(likes+dislikes),
    Dislike_rate = dislikes/(likes+dislikes)
  ) %>%
  replace_na(list(Like_rate = 0)) %>%
  replace_na(list(Disike_rate = 0)) %>%
  mutate(
    Rate_perc = Like_rate-Dislike_rate
  ) %>%
  replace_na(list(Rate_perc = 0))

dfclustertest1 = dfclustertest %>%
  select(view_count, Rate_perc)

dfclustertest2 = dfclustertest1 %>%
  scale()

#-------------------------------------------------
#Setting the seed to 123 so that output remains the same every time we run the
set.seed(123)

#Function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(dfclustertest2, k, nstart = 10 )$tot.withinss
}

#Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

#Extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)

plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```




```
#---------------------------------------------
#Creating a clustering using the kmeans function
#Making a string of the custer to plot the graph

clus <- kmeans(dfclustertest2, centers = 5, nstart = 25)

# str(clus)
#
#  clus

Cluster_Plot = fviz_cluster(clus, data = dfclustertest2, geom="point")

Cluster_Plot
```




```
# rm(clus, dfclustertest,dfclustertest2, k.values, wss_values, wss)
```

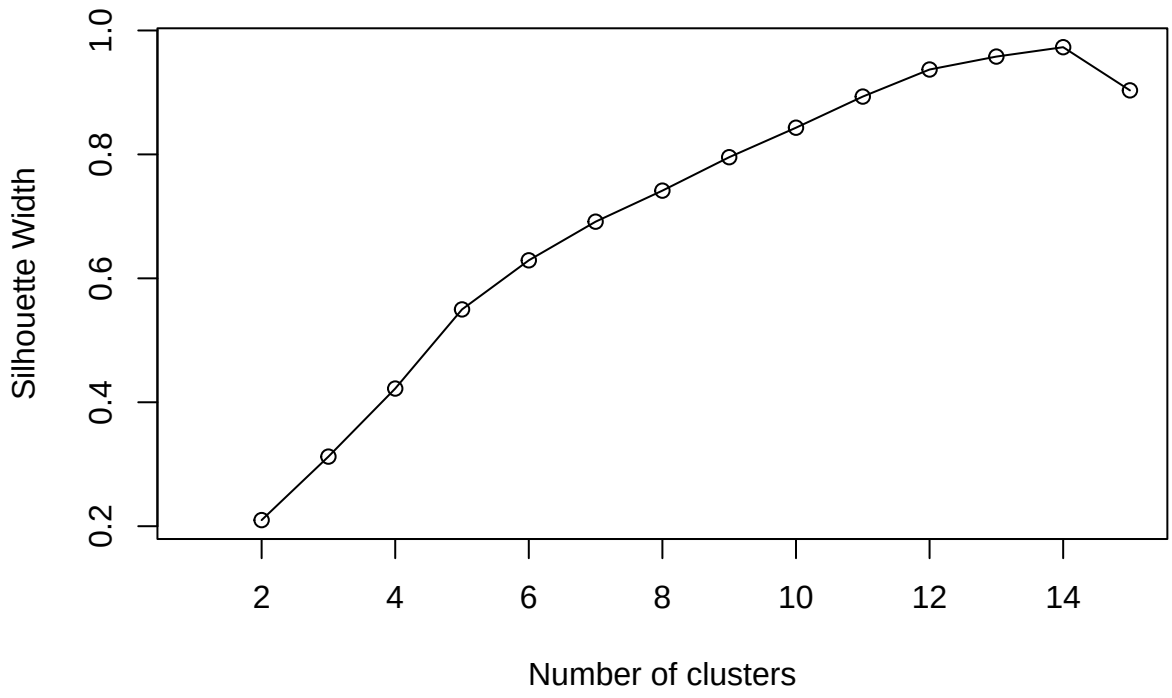
From the first graph, it was decided that k=5 is the one that maximizes the average within cluster sum of squares over a range of values of k from 2 to 14. With number of clusters equal to 5, it can be observed that Cluster 2 and 3 have lower likes and view count whereas Cluster 1 and 5 had the moderate likes and view comments but the Cluster 4, on the other hand, had the most likes and view counts.

5. Which category are likely obtain more views and likes

Selecting likes and categoryid and displaying the first 2000 items and then we compute all the dissimilarities (Distance) between observations in the data set, convert into matrix and then find the values which has similarities with the minimum and maximum distance for clustering. This data is then partitioned into 5 clusters using pam() function which is then plotted.

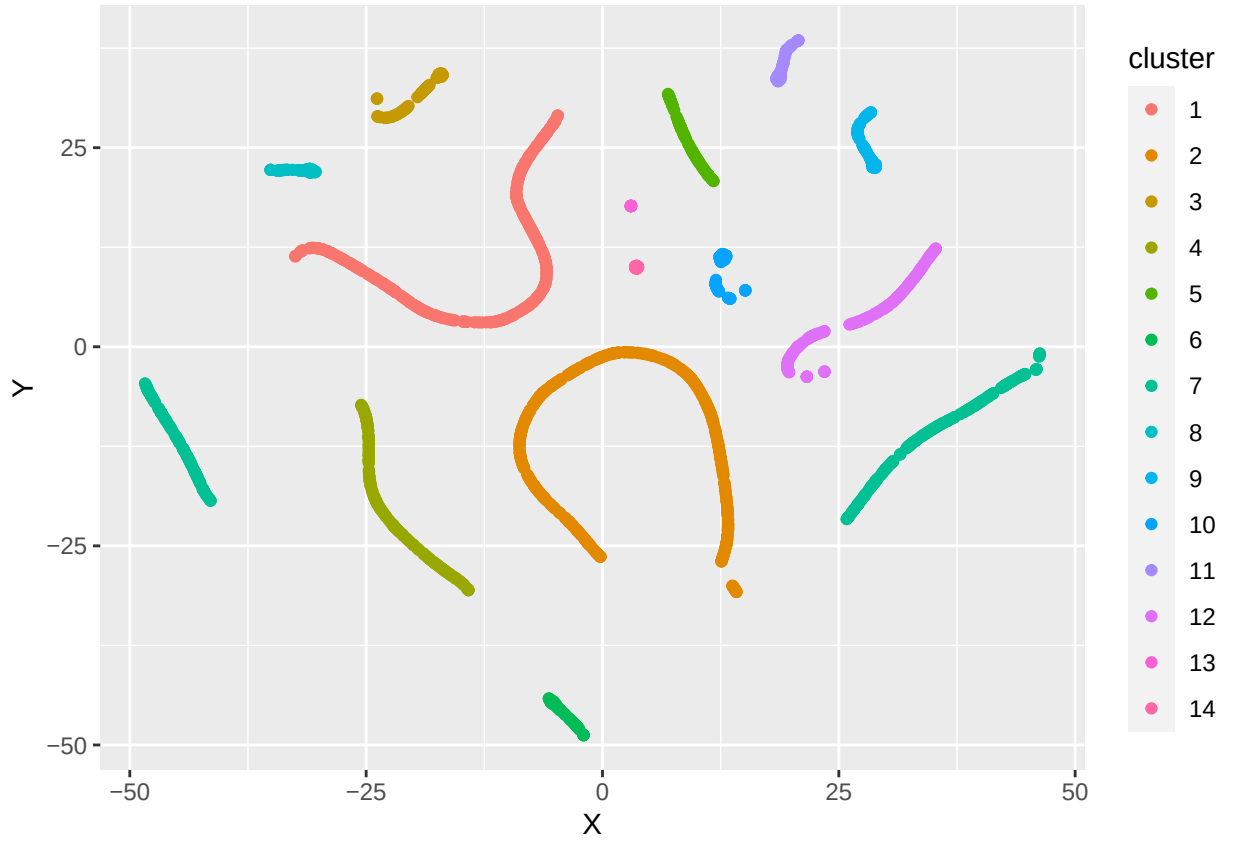
```
# Compute Gower distance

# Compute Gower distance
set.seed(1234)
A = df %>%
  select(likes, categoryId) %>%
  head(2000)
A$categoryId = as.factor(A$categoryId)

gower_dist <- daisy(A, metric = "gower")
gower_mat <- as.matrix(gower_dist)
```

```r
# Print most similar clients
df[which(gower_mat ==
           min(gower_mat[gower_mat != min(gower_mat)]), arr.ind = TRUE)[1, ], ]

# Print most dissimilar clients
df[which(gower_mat ==
           max(gower_mat[gower_mat != max(gower_mat)]), arr.ind = TRUE)[1, ], ]

sil_width <- c(NA)
for(i in 2:15){
  pam_fit <- pam(gower_dist, diss = TRUE, k = i)
  sil_width[i] <- pam_fit$silinfo$avg.width
}
plot(1:15, sil_width,
     xlab = "Number of clusters",
     ylab = "Silhouette Width")
lines(1:15, sil_width)
```




```r
k <- 14
pam_fit <- pam(gower_dist, diss = TRUE, k)
pam_results <- A %>%
  mutate(cluster = pam_fit$clustering) %>%
  group_by(cluster) %>%
  do(the_summary = summary(.))
pam_results$the_summary
```

```
tsne_obj <- Rtsne(gower_dist, is_distance = TRUE)
tsne_data <- tsne_obj$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(pam_fit$clustering))

ggplot(aes(x = X, y = Y), data = tsne_data) +
  geom_point(aes(color = cluster))
```



Using average silhouette method, the first graph shows the number of clusters which had the most width was 14. From the output it can be seen that 14 different clusters are created, grouped, factored according to the number of likes and their respective categories, stored the result as into two categories (X,Y) and plotted as various colors displayed on graph.

As the conclusion, category performed the best is category 10 (Music). Contradictly, category 25 (News & Politics) performed worth. A youtuber should make more music videos rather than politics videos to gain more likes.

Text Minning

6. Given that a statement video games brings violence to people and unconsciously influence audiances as they play more games. Is the statement real? Analysis the game records based on data given.

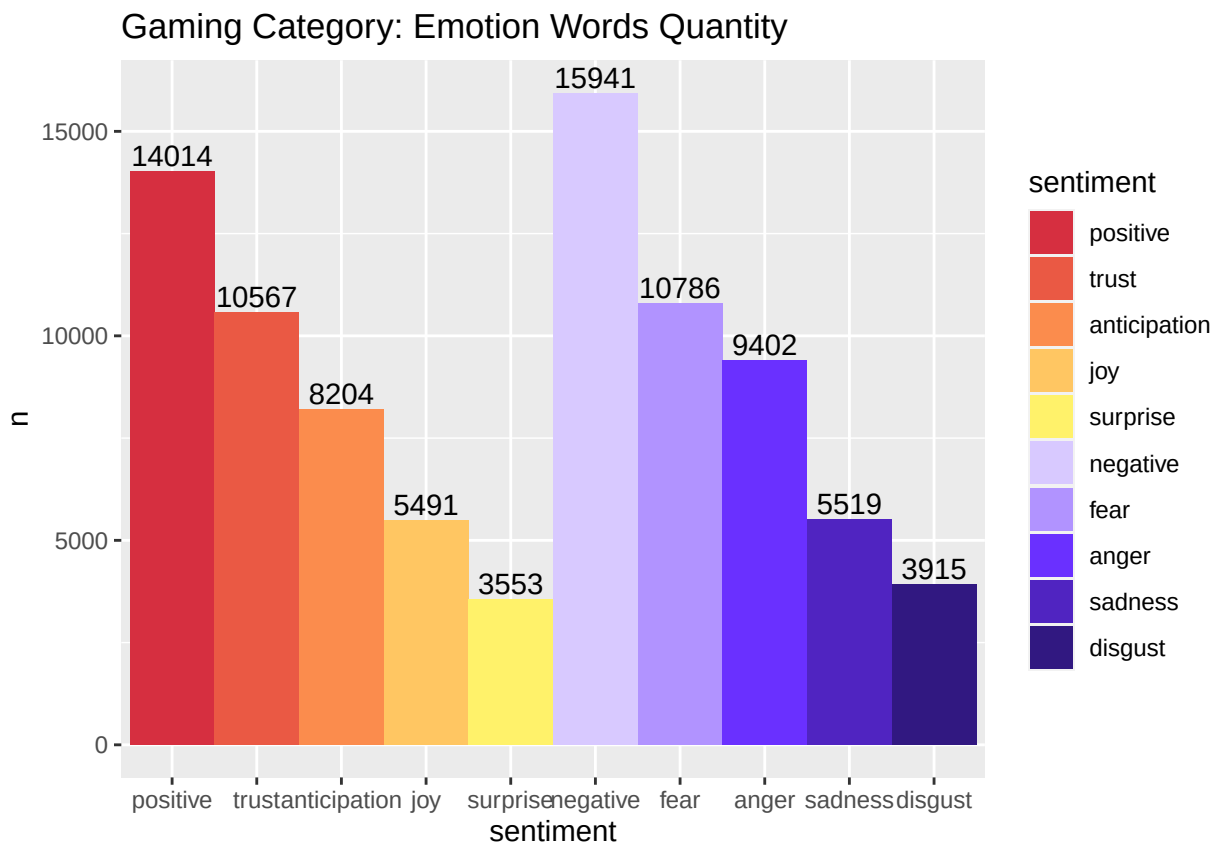
Text mining is the process of transforming unstructured text into a structured format to identify meaningful patterns and new insights.

The data set is first separated into two categories Gaming Videos (category id = 20) and Non-gaming Videos. Text analysis is performed on both these categories in order to detect various sentiments mentioned in the tags and then display it

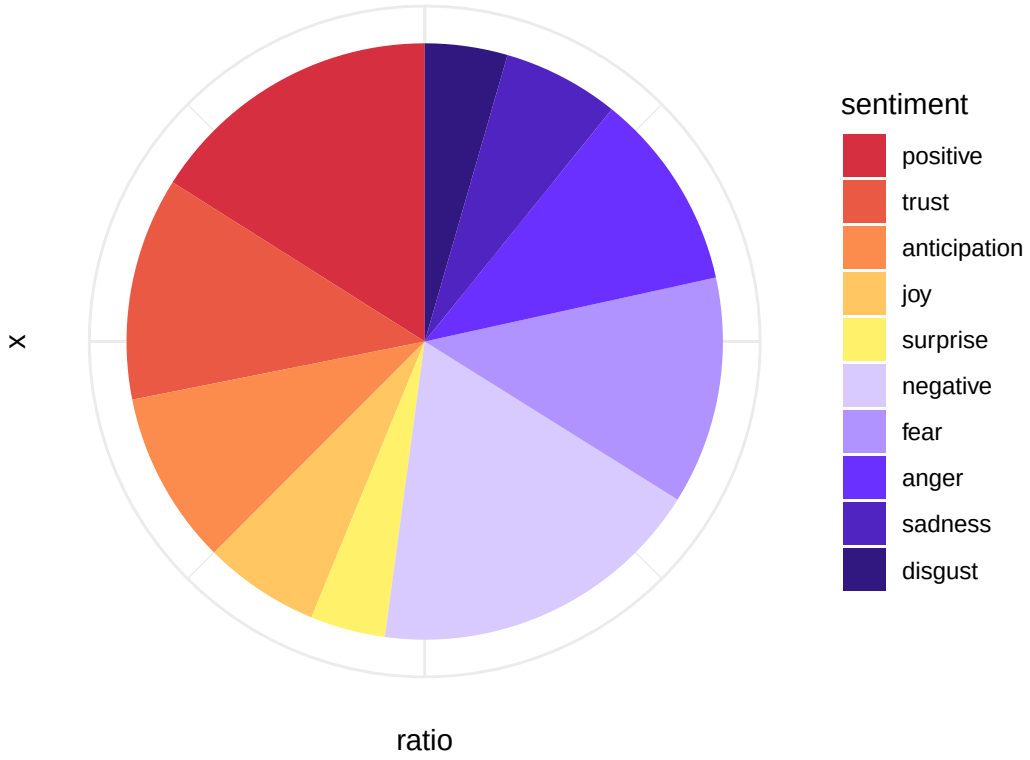
The group found out that there were several rows with "[None]" in the tag column, submitted them into new dataset, used get_sentiments() function to determine the emotions provided in the tags and categorize each of them and display them accordingly.

```
#Filter the category id as 20 and then selecting the title, tags, and cateogyId.
A = df %>%
  dplyr::filter (df$categoryId == "20") %>%
  select(title, tags, categoryId)

# Creating a subset from Tags where the rows contains data as [None]
A = subset(A,tags!="[None]")

#Making one-token-per-row.
B = A %>%
    unnest_tokens(word, title)

#Get specific sentiment lexicons in a tidy format, with one row per word
NRC = get_sentiments("nrc")
data(stop_words)


B = B %>%
  inner_join(NRC) %>%
  anti_join(stop_words)
#Counting and Displaying Top 5 words for Gaming Videos
kable(B %>%
  count(word, sort = TRUE) %>%
  head(5)
  , caption = "Top 5 Words Appear for Gaming Videos")

#Counting and Displaying the ratio
B = B %>%
  count(sentiment, sort = TRUE) %>%
  mutate(ratio = n/sum(n))

#Making new column Index and arrange them accordingly
B_1 = B %>%
  mutate( index = c(2,1,2,1,2,1,2,1,2,1)) %>%
  arrange(index)

#Factoring them
B_1$sentiment = factor(B_1$sentiment, levels = B_1$sentiment)

#Plotting the graph for sentiments
B_1 %>%
  ggplot (aes(x = sentiment, y = n , fill = sentiment))+
  geom_bar(width = 1, stat = "identity")+
  scale_fill_manual(values = c("positive" = "#d62f40","trust" = "#ea5944",
                               "anticipation" = "#fb8c4d","joy"="#ffc662",
                               "surprise"="#fff26a","negative"="#d8c9ff",
                               "fear"="#b193ff","anger"="#6a30ff",
```

```
                                "sadness"="#5024c1","disgust"="#311881"))+
  geom_text(aes(label=n), position=position_dodge(width=0.9), vjust=-0.25)+
  ggtitle("Gaming Category: Emotion Words Quantity")
```

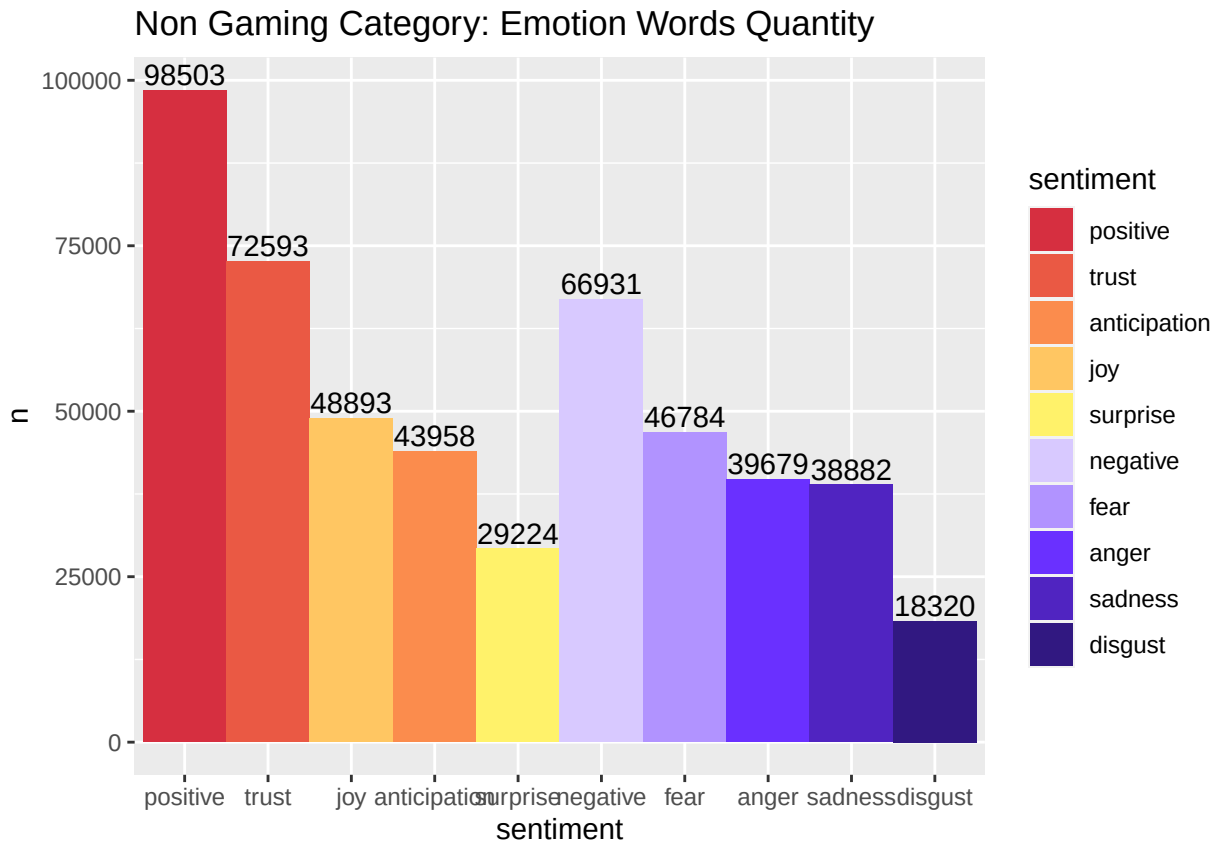



```
B_1 %>%
  ggplot(aes(x="", y=ratio, fill = sentiment))+
  geom_bar(width = 1, stat = "identity")+
  coord_polar("y", start=0)+
  scale_fill_manual(values = c("positive" = "#d62f40","trust" = "#ea5944",
                               "anticipation" = "#fb8c4d","joy"="#ffc662",
                               "surprise"="#fff26a","negative"="#d8c9ff",
                               "fear"="#b193ff","anger"="#6a30ff",
                               "sadness"="#5024c1","disgust"="#311881"))+
  theme_minimal()+
  theme(axis.text.x=element_blank())+
  ggtitle("Gaming Category: Emotion Words Percentage")
```



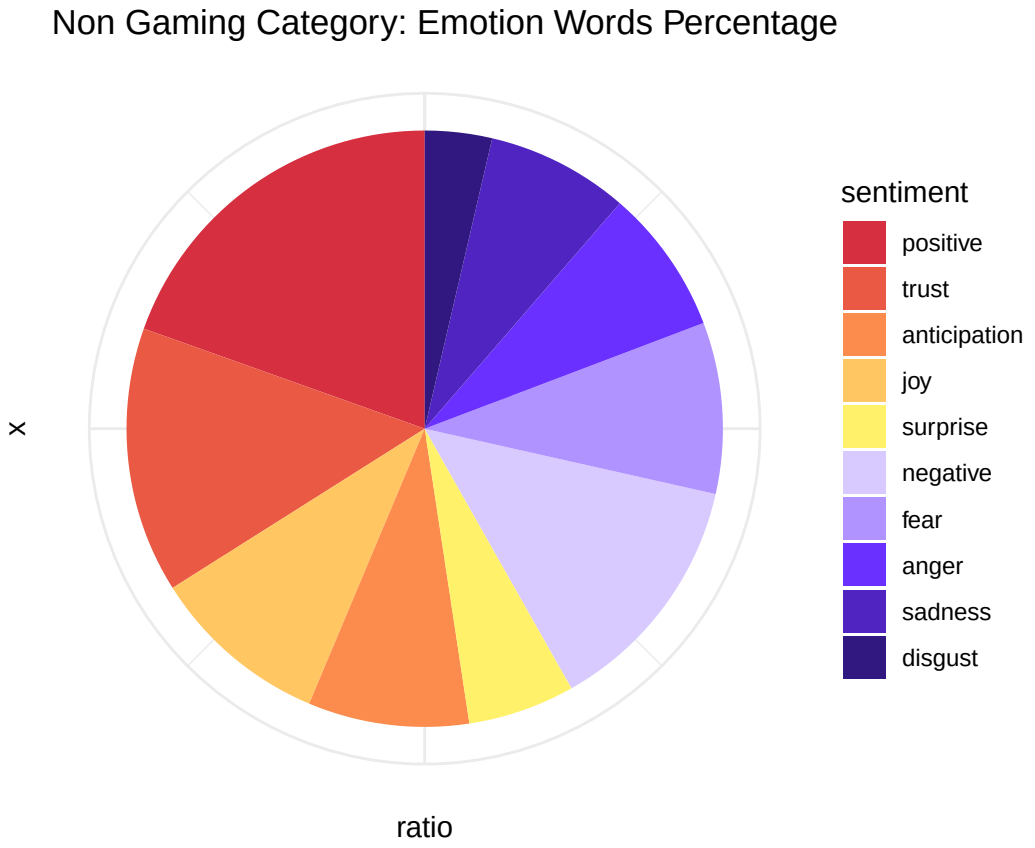

```
#-----------------------------------------------
#Filter the category id as 20 and then selecting the title, tags, and cateogyId.
A1 = df %>%
  dplyr::filter (df$categoryId != "20") %>%
  select(title, tags, categoryId)

# Creating a subset from Tags where the rows contains data as [None]
A1 = subset(A1,tags!="[None]")

#Making one-token-per-row.
B1 = A1 %>%
    unnest_tokens(word, title)

#Using inner_join function of Rrc words
#Using Anto_join to stop the words that are not needed
#Counting the sentiment words, and making tne ratio of it.
B1 = B1 %>%
  inner_join(NRC) %>%
  anti_join(stop_words) %>%
  count(sentiment, sort = TRUE) %>%
  mutate(ratio = n/sum(n))

#Making new column Index and arrange them accordingly
B_11 = B1 %>%
  mutate( index = c(1,1,2,1,2,1,2,2,1,2)) %>%
  arrange(index)
```

```
#Factoring them
B_11$sentiment = factor(B_11$sentiment, levels = B_11$sentiment)

#Plotting the graph for sentiments
B_11 %>%
  ggplot (aes(x = sentiment, y = n , fill = sentiment))+
  geom_bar(width = 1, stat = "identity")+
  scale_fill_manual(values = c("positive" = "#d62f40","trust" = "#ea5944",
                               "anticipation" = "#fb8c4d","joy"="#ffc662",
                               "surprise"="#fff26a","negative"="#d8c9ff",
                               "fear"="#b193ff","anger"="#6a30ff",
                               "sadness"="#5024c1","disgust"="#311881"))+
  geom_text(aes(label=n), position=position_dodge(width=0.9), vjust=-0.25)+
  ggtitle("Non Gaming Category: Emotion Words Quantity")
```




```
B_11 %>%
  ggplot(aes(x="", y=ratio, fill = sentiment))+
  geom_bar(width = 1, stat = "identity")+
  coord_polar("y", start=0)+
  scale_fill_manual(values = c("positive" = "#d62f40","trust" = "#ea5944",
                               "anticipation" = "#fb8c4d","joy"="#ffc662",
                               "surprise"="#fff26a","negative"="#d8c9ff",
                               "fear"="#b193ff","anger"="#6a30ff",
                               "sadness"="#5024c1","disgust"="#311881"))+
  theme_minimal()+
```

```
  theme(axis.text.x=element_blank())+
  ggtitle("Non Gaming Category: Emotion Words Percentage")
```




```
#Removing the Variable from the environments,
rm(A,A1,B,B_1,B_11,B1,C,NRC,stop_words)
```

The bar and pie charts for gaming and non-gaming category illustrates that the people were feeling all kinds of negative emotions (including fear,sadness, disgust, anger) while watching a video belonginb to the gaming category whereas people watching videos belonging to the non-gaming categories had negative impact too but lesser as compared the former category. The opposite is true for the positive sentiments for both gaming and non-gaming category.

7. Tag present the topic that people interested though out the year. Tags appeared the most for three country.

In this secction, the group used the tag and category column to perform wordcloud.

```
#Selecting category, tags and removing na values from them
A = df %>%
  select(categoryId,tags) %>%
  na.omit()

# Creating a subset from Tags where the rows contains data as [None]
A = subset(A,tags!="[None]")
```

```
#Replacing the not required symbol to required symbol.
A$tags = gsub("\\|", ",", A$tags)
# A$tags = unlist(strsplit(A$tags, ","))

#Making one-token-per-row.
B = A %>%
  unnest_tokens(CleanTag, tags,token = 'regex', pattern=",")

#Counting and displaying the first 2000 tags,
C = B %>%
  count(CleanTag, sort = TRUE) %>%
  head(2000)

#Setting the seed and using wordcloud to plot the tags.
set.seed(1234)
C %>%
  with(wordcloud(CleanTag, n, max.words = 2000,random.order=FALSE))
```

Here, the words used in the tags are counted and displayed. The size of the word is proportional to the number of times those words are repeated. The output states that "funny" is the most used word followed by minecraft,comedy, and football.

Time series

8. What are the parten and relationship of view count?

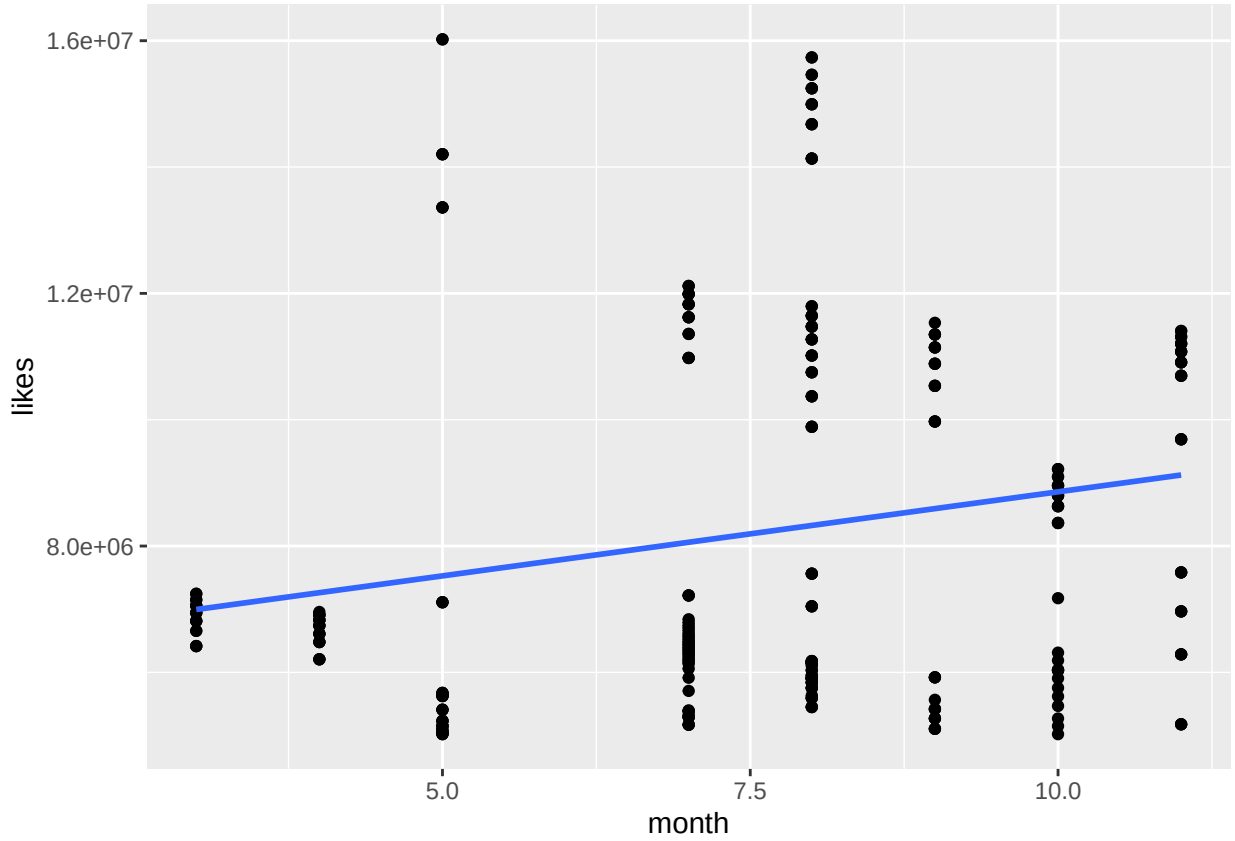
It is a time series Recurrence Quantification Analysis (RQA, Heat map) when over in the lecture. It need to go though a process of pre-treading the raw data. From a Time series to A Emphases reconstruction; then finally a RQA to dig the relationship between time and the repeating. Time series analysis have a particular algorithm for such task, and lucky we direct use a package pre-builded.

```
# Time series Recurrence Quantification Analysis (RQA, Heat map)
#Display the first 200 items of select section

ts = df %>%
  count(publishedAt) %>%
  select(n) %>%
  head(200)

#Selecting the range from 1 to 200.
ts = ts$n[1:200]

# Recurrence plot of acc signals for walking

rqa.analysis=rqa(time.series = ts, embedding.dim=2, time.lag=3,
                 radius=2,lmin=2,do.plot=FALSE,distanceToBorder=2)
```

9. Is youtube community having more and more high quality videos (likes over 5M) being published? Prediction of high-quality videos.

We filter the likes that are more than 5000000, so that we can have the proper regression graph, use lm() function, which is used to create a linear model to carry out regression on using likes on y axis and month on x axis. Same operation is performed for dislikes on y axis and month on x axis. Finally a regression graph is plotted.

```
#Filtering likes greater
a=df %>%
  dplyr::filter(likes >5000000)

# Performing regression on using likes on y axis and month on x axis.
fit1=lm( likes ~ month,data=a)
  summary(fit1)

#Plotting the output
ggplot(fit1,aes(y=likes,x=month))+
  geom_point()+
  stat_smooth(method="lm",se=FALSE)
```



From the regression plot, the likes are segregated according to the months and the regression line indicates that the number of likes keeps on increasing. The second graph depicts the relationship between view count and dislikes along with the regression line which has a positive slope indicating that as the view count increases, the number of likes increases too.

Conclusion

Initially, the group researched various websites and open-source data sets sharing platforms and finalized the YouTube Data set. This data set was chosen because it includes likes dislikes as numerical values on which clustering can be performed. The dataset also has dates and time to perform Time series, Tags which includes words depicting different sentiments which is supportive to perform Text mining.

While calculating Probability, the data set had Category id but had missing Category Names which was required for visualization of the ideas and their correlation analysis. Hence, the group researched that and added the attribute to our original data set using the left join function. Probability for each was calculated using the formula and Cumulative Distribution Function (CDF) was calculated using the cumsum () function and the same was visualized using ggplot ()

The group determined the number of clusters using two methods. We used the daisy () function which establishes the similarities and dissimilarities between two data points. A more robust function of the k-means was discovered called pam () which partitions the data around the medoids into k clusters which were incorporated in the second problem of Clustering.

In-Text mining, using the tags column the different sentiments of people were picked up using the get_sentiments () function using the dplyr library. This function gets a specific sentiment lexicon in a tidy format, with one row per word, in a form that can be combined with a one-word-per-row. Furthermore, the group encountered the issue of the symbols like "\|" which were causing a problem to perform wordcloud

which was overcome by using the gsub () function which detects all those patterns and replaces them with what you want.

As one of the two critical methods was being covered during the semester, A recurrence Quantification Analysis was being performed during the project. In the first part of the Time-series Analysis, by manually testing multiple combinations of lag and threshold distance, a reasonable heat map was generated with the tolerable number of noises. In the second part of the time series analysis, by looking at the top rending videos (View over 5M), the fit line has been observed that an increase every month of high-quality video being made an entire YouTube community.

As the project requirement stated, the final submission should be in pdf format by using the knit to pdf in-build function within RStudio. With computation restrictions such as short in memory, the team spent more than the necessary amount of time to knit the pdf file. Some minor tips were discovered during attempts, such as adding cache = TRUE for R code block to avoid calculating the same factors repeatedly. Since a large data set takes an overwhelming amount of memory, knitting would still cause the device to not respond or even crush.

In a conclusion, the team successfully encountered all the constraints and met the initial objectives. Proper procedures of data wrangling and data-driven business question were mastered after the complement of the project.