Exploration of Netflix Dataset in R

Yigit Erol

18 04 2020

Overview

This dataset consists of tv shows and movies available on Netflix as of 2019. The dataset which can be found here: https://www.kaggle.com/shivamb/netflix-shows (https://www.kaggle.com/shivamb/netflix-shows) is collected from Flixable which is a third-party Netflix search engine.

Exploration and Modification of Dataset

In this part we will check the obervations, variables and values of our data.

This section created by 3 parts; data reading, data cleaning and data visualisation

3 different libraries (ggplot2, ggpubr, plotly) are used to visualise data.

Data Reading

Lets read the data and rename it as “netds” to get more useful and easy coding in functions. In the below we have to write na.string=c(""“,”NA") because, some values of our data are empty or taking place as NA. If we do not specify them at the begining in reading function, we can not reach the missing values in future steps.

- Why does not stringsAsFactors default as FALSE ?

The argument ‘stringsAsFactors’ is an argument to the ‘data. frame()’ function in R. It is a logical that indicates whether strings in a data frame should be treated as factor variables or as just plain strings.

```
netds <- read.csv("netflix_titles.csv", na.strings = c("", "NA"), stringsAsFactors =FALSE)
```

In the dataset there are 6234 observations of 12 following variables describing the tv shows and movies:

```
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
values_table1 <- rbind(c('show_id', 'type', 'title', 'director', 'cast', 'country', 'date_add
ed', 'release_year', 'rating' , 'duration', 'listed_in', 'description'), c("Unique ID for eve
ry Movie / TV Show",
     "Identifier - A Movie or TV Show",
     "Title of the Movie or TV Show",
     "Director of the Movie /TV Show",
    "Actors involved in the Movie / TV Show",
    "Country where the movie / show was produced",
    "Added date on Netflix",
    "Actual release year of the Movie / TV Show",
    "Rating type of the Movie or TV Show",
    "Total Duration - in minutes or number of seasons",
    "Genere",
    "The summary description"))

fig_table1 <- plot_ly(
  type = 'table',
  columnorder = c(1,2),
  columnwidth = c(12,12),
  header = list(
    values = c('<b>VARIABLES</b><br>', '<b>DESCRIPTION</b>'),
    line = list(color = '#506784'),
    fill = list(color = '#119DFF'),
    align = c('left','center'),
    font = list(color = 'white', size = 12),
    height = 40
  ),
  cells = list(
    values = values_table1,
    line = list(color = '#506784'),
    fill = list(color = c('#25FEFD', 'white')),
    align = c('left', 'left'),
    font = list(color = c('#506784'), size = 12),
    height = 30
    ))

fig_table1
```

VARIABLES	DESCRIPTION
show_id	Unique ID for every Movie / TV Show
type	Identifier - A Movie or TV Show
title	Title of the Movie or TV Show
director	Director of the Movie /TV Show
cast	Actors involved in the Movie / TV Show
country	Country where the movie / show was produced
date_added	Added date on Netflix

release_year	Actual release year of the Movie / TV Show
rating	Rating type of the Movie or TV Show
duration	Total Duration - in minutes or number of seasons
listed_in	Genere
description	The summary description

Data Cleaning

As a first step of the cleaning part, we can remove unnecessary variables and parts of the data such as show_id variable. Also description variable will not be used for the analysis or visualisation but it can be useful for the further analysis or interpretation.

```
netds$show_id <- NULL
```

Rating is categorical variable so we will change the type of it.

```
netds$rating <- as.factor(netds$rating)
```

We also can change the date format of date_added variable.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
netds$date_added <- mdy(netds$date_added)
```

“type” and “Listed_in” should be categorical variable

```
netds$listed_in <- as.factor(netds$listed_in)

netds$type <- as.factor(netds$type)
```

Missing values can be problem for the next steps. Therefore, we have to check them before the analyse and then we can fill the missing values of some variables if it is necessary.

```
# printing the missing values by creating a new data frame

data.frame("Variable"=c(colnames(netds)), "Missing Values"=sapply(netds, function(x) sum(is.na(x))), row.names=NULL)
```

```
##        Variable Missing.Values
## 1          type              0
## 2         title              0
## 3      director           1969
## 4          cast            570
## 5       country            476
## 6    date_added             11
## 7  release_year              0
## 8        rating             10
## 9      duration              0
## 10    listed_in              0
## 11  description              0
```

We can clearly see that missing values take place in director, cast, country, data_added and rating variables. Since rating is the categorical variable with 14 levels we can fill in (approximate) the missing values for rating with a mode.

```
#function to find a mode

mode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

netds$rating[is.na(netds$rating)] <- mode(netds$rating)
```

Check again the filling

```
data.frame("Variable"=c(colnames(netds)), "Missing Values"=sapply(netds, function(x) sum(is.n
a(x))), row.names=NULL)
```

```
##        Variable Missing.Values
## 1          type              0
## 2         title              0
## 3      director           1969
## 4          cast            570
## 5       country            476
## 6    date_added             11
## 7  release_year              0
## 8        rating              0
## 9      duration              0
## 10    listed_in              0
## 11  description              0
```

Now, we are going to drop the missing values, at point where it will be necessary. We also drop duplicated rows in the dataset based on the “title”, “country”, “type”," release_year" variables.

```
#title, country, type and release_year

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
netds=distinct(netds, title, country, type, release_year, .keep_all = TRUE)
```

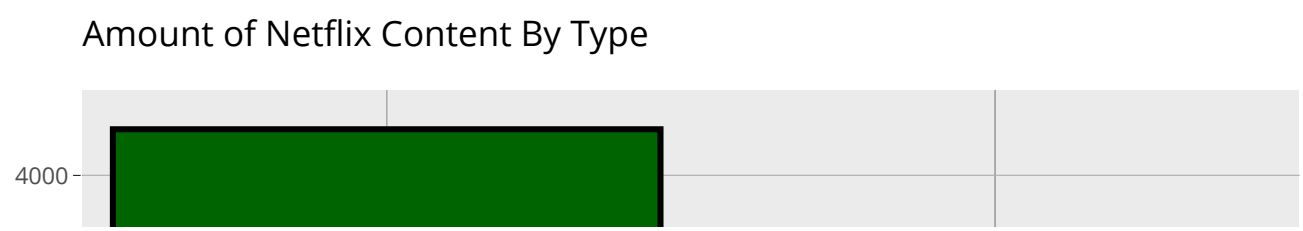
Data cleaning process is done. Now we can start to visualisation.

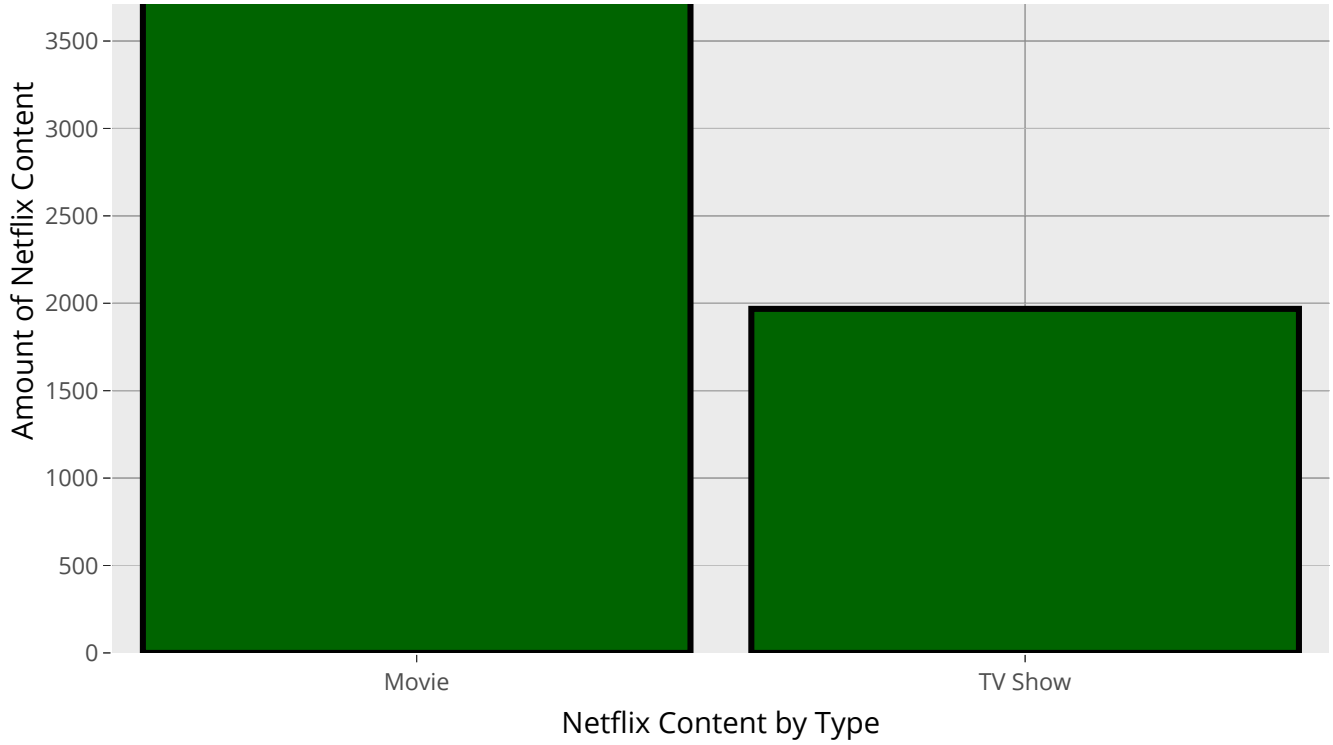
Data Visualisation

Amount of Netflix Content by Type

In the first graphy, ggplot2 library is used and data visualised with basic bar graph. In the code part, some arguments of functions will be described.

```
library(tibble)
library(dplyr)
library(ggplot2)


# Here we created a new table by the name of "amount_by_type" and applied some filter by usin
g dplyr library. Primarly, group_by() function is used to select variable and then used summa
rise() function with n() to count number of TV Shows and Movies.

amount_by_type <- netds %>% group_by(type) %>% summarise(
  count = n())

# In ggplot2 library, the code is created by two parts. First one is ggplot(), here we have t
o specify our arguments such as data, x and y axis and fill type. then continue with + and ty
pe of the graph will be added by using geom_graphytype.


figure00  <- ggplot(data = amount_by_type, aes(x= type, y= count, fill= type))+
  geom_bar(colour ="black", size= 0.8, fill = "dark green" ,  stat = "identity")+
  guides(fill= FALSE)+
  xlab("Netflix Content by Type") + ylab("Amount of Netflix Content")+
  ggtitle("Amount of Netflix Content By Type")

ggplotly(figure00, dynamicTicks = T)
```





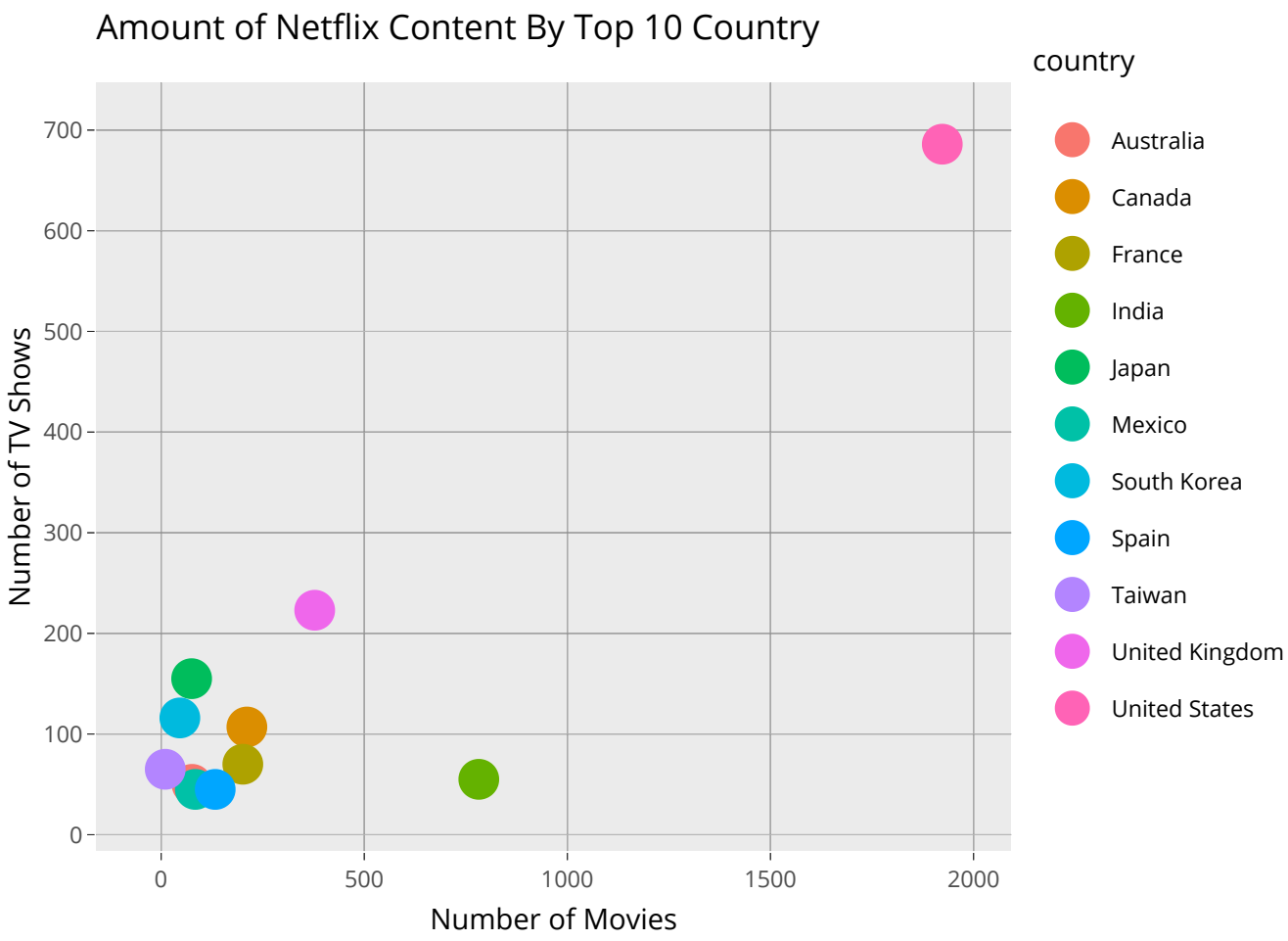
As we see from above there are more than 2 times more Movies than TV Shows on Netflix.

Amount of Netflix Content By Top 10 Country

```
# 1: split the countries (ex: "United States, India, South Korea, China" form to 'United Stat
es' 'India' 'South Korea' 'China') in the country column by using strsplit() function and the
n assign this operation to "k" for future use.

k <- strsplit(netds$country, split = ", ")



# 2: Created a new dataframe by using data.frame() function. First column should be type = se
cond one country=. Created type column by using rep() function. The function replicates the v
alues in netds$type depends on the length of each element of k. we used sapply()) function. N
ow k is our new data in sapply(). it means that calculate the lenght of each element of the k
list so that we create type column. In the country column, we used just unlist() function. It
simply converts the list to vector with all the atomic components are being preserved.

netds_countries<- data.frame(type = rep(netds$type, sapply(k, length)), country = unlist(k))

# 3: Changed the elements of country column as character by using as.charachter() function.

netds_countries$country <- as.character(netds_countries$country)

# 4: we created new grouped data frame by the name of amount_by_country. NA.omit() function d
eletes the NA values on the country column/variable. Then we groupped countries and types by
 using group_by() function (in the "dplyr" library). After that used summarise() function to
 summarise the counted number of observations on the new "count" column by using n() functio
n.

amount_by_country <- na.omit(netds_countries) %>%
  group_by(country, type) %>%
  summarise(count = n())

# 5: Actually we can use the "amount_by_country" dataframe to observe number of TV Show or Mo
vie in countries. However, this list is too big to be visualized. Thus, we will create a new
 dataframe as table to see just top 10 countries by the name of "u".

# reshape() fonction will be used to create a reshaped grouped data. amount_by_country is use
d as data in the function. In this function, we will describe id variable, names of the valu
e, time variable, and direction. Direction is character string, partially matched to either
 "wide" to reshape to wide format, or "long" to reshape to long format. Then we applied arran
ge() funtion to the reshaped grouped data. The dplyr function arrange() can be used to reorde
r (or sort) rows by one or more variables. In this part we sort count.movie column as descend
ing.

# To check to arguments and detailed descriptions of functions please use to help menu or goo
gle.com

# After the arrange funtion, top_n() function is used to list the specified number of rows.

u <- reshape(data=data.frame(amount_by_country),idvar="country",
                          v.names = "count",
                          timevar = "type",
                          direction="wide") %>% arrange(desc(count.Movie)) %>%
                          top_n(10)
```

```
## Selecting by count.TV Show
```

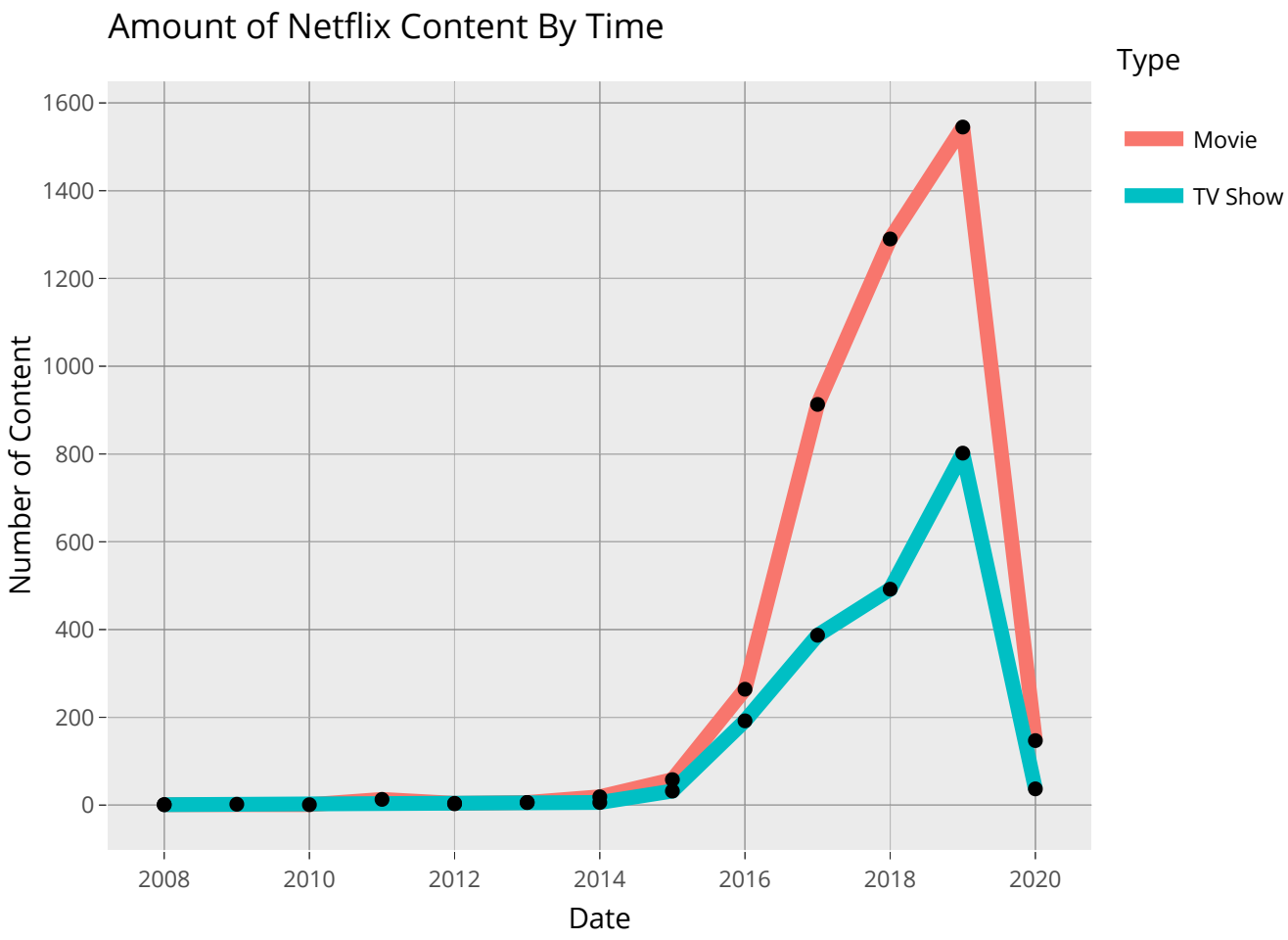
```
# 6: names of the second and third columns are changed by using names() function as seen belo
w.

names(u)[2] <- "Number_of_Movies"
names(u)[3] <- "Number_of_TV_Shows"

# 7: In the arrange() function we sorted our count.movie columns as descending but, now, we w
ant to change this sort depends on the total values of "number of Movies" and "number of TV S
hows". To sort a data frame in R, use the order() function. By default, sorting is ASCENDING.
Therefore, we have to specify as descending. + is used to specify total operation.

u <- u[order(desc(u$Number_of_Movies +u$Number_of_TV_Shows)),]

# 8: Now we can create our graph by using ggplot2 library. First argument of the ggplot funct
ion is our data.frame, then we specified our variables in the aes() function. coloured the gr
aphy depends on the countries. Then typeof the graph is writed as geom_point and dot size spe
cified as 5. After that we named x and y axis. Title of the graph is wroted by using ggtitle
() function.

library(ggplot2)



figure000 <- ggplot(u, aes(Number_of_Movies, Number_of_TV_Shows, colour=country))+
  geom_point(size=5)+
  xlab("Number of Movies") + ylab("Number of TV Shows")+
  ggtitle("Amount of Netflix Content By Top 10 Country")

ggplotly(figure000, dynamicTicks = T)
```



We see that the United States is a clear leader in the amount of content on Netflix.

Amount of Netflix Content By Time

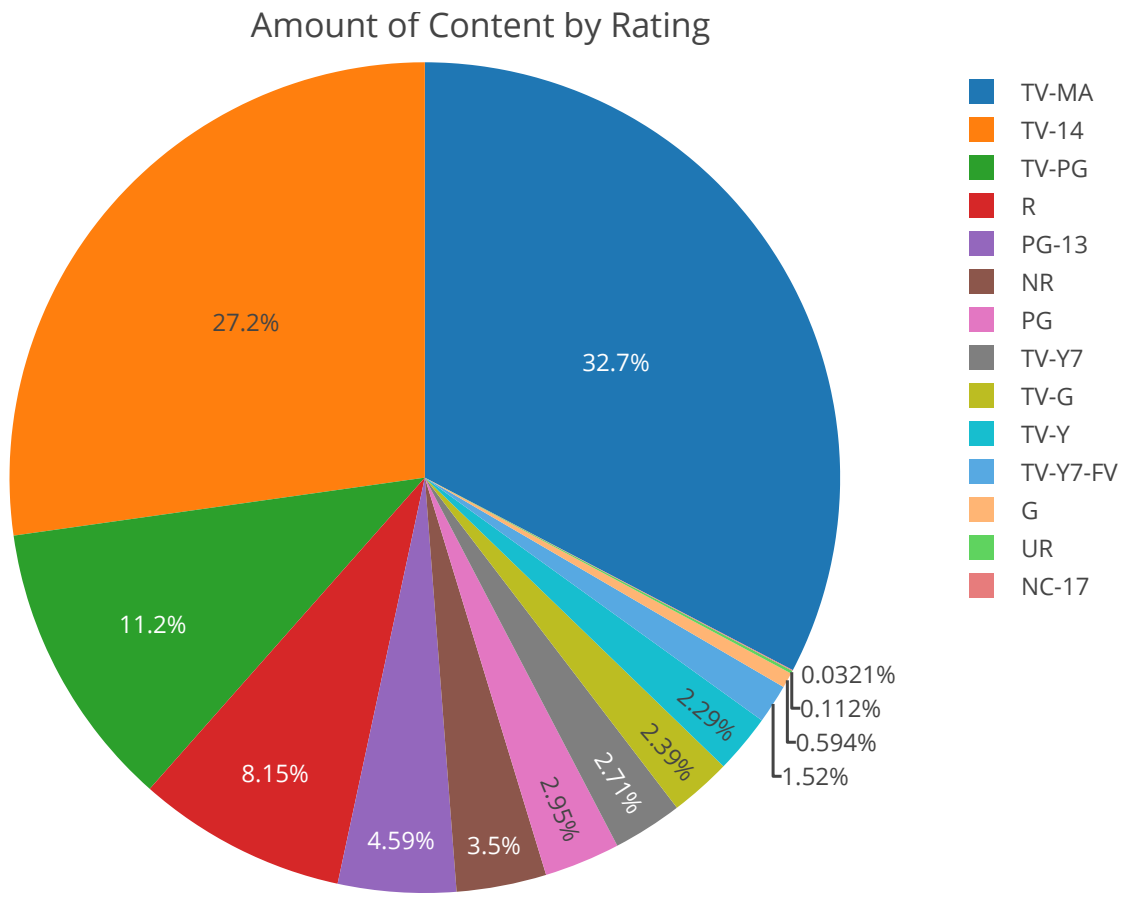
```
# 0: To see number contents by time we have to create a new data.frame. This process is a lit
tle tiring. Maybe there is a short way but I couldn't find it. Lets start!

# 1: Title column take place in our dataframe as character therefore I have to convert it to
 tbl_df format to apply the function below. If this column remains in character format and I
 want to implement the function, R returns an error: " Error in UseMethod("group_by_") : no a
pplicable method for 'group_by_' applied to an object of class "character""     Therefore, fi
rst I assign it title column to f then convert the format as tibble and then assign it again
 to title column.

f <- netds$title
f <-tibble(f)
netds$title <- f

# 2: new_date variable created by selecting just years. In this way, we can analyze and visua
lise the data more easy

library(lubridate)

netds$new_date <- year(netds$date_added)




# 2: df_by_date crated as a new grouped data frame. Titles are grouped depending the new_date
(year) and then na.omit function applied to date column to remove NA values. Finally, number
 of added contents in a day calculated by using summarise() and n() functions.

df_by_date <- netds$title %>%
  group_by(netds$new_date, netds$type) %>%
  na.omit(netds$new_date) %>%
  summarise(added_content_num = n())


# 3: now we will visualize our new grouped data frame.

library(ggplot2)


Type<- df_by_date$`netds$type`

Date <- df_by_date$`netds$new_date`

Content_Number <- df_by_date$added_content_num

g1<- ggplot(df_by_date, aes(Date, Content_Number))+
  geom_line(aes(colour = Type), size = 2)+
  geom_point() +
  xlab("Date") +
  ylab("Number of Content")+
  ggtitle("Amount of Netflix Content By Time")

ggplotly(g1, dynamicTicks = T)
```



From above we see that starting from the year 2016 the total amount of content was growing exponentially. We also notice how fast the amount of movies on Netflix overcame the amount of TV Shows. The reason for the decline in 2020 is that the data we have is ending begining of the 2020. At the begining of 2020, the number of ingredients produced is small. Before to say something about 2020 we have to see year-end data.

Amount of Content by Rating

```
# Here plotly library used to visualise data. To see the graph in chunk output or console you
have to assign it to somewhere such as "fig"

library(plotly)

data <-netds$title %>%
  group_by(netds$rating) %>%
  summarise(content_num = n())

names(data) [1] <- "rating"
names(data) [2] <- "content"

# From the above, we created our new table to use in graph


figure2 <- plot_ly(data, labels = ~rating, values = ~content, type = 'pie')

# In the first part of visualisation, again, we have to specify our data labels, values,  x a
d y axis and type of graph.

# In second part, adding title and other arguments of graph.

figure2 <- figure2 %>% layout(title = 'Amount of Content by Rating',
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))

figure2
```

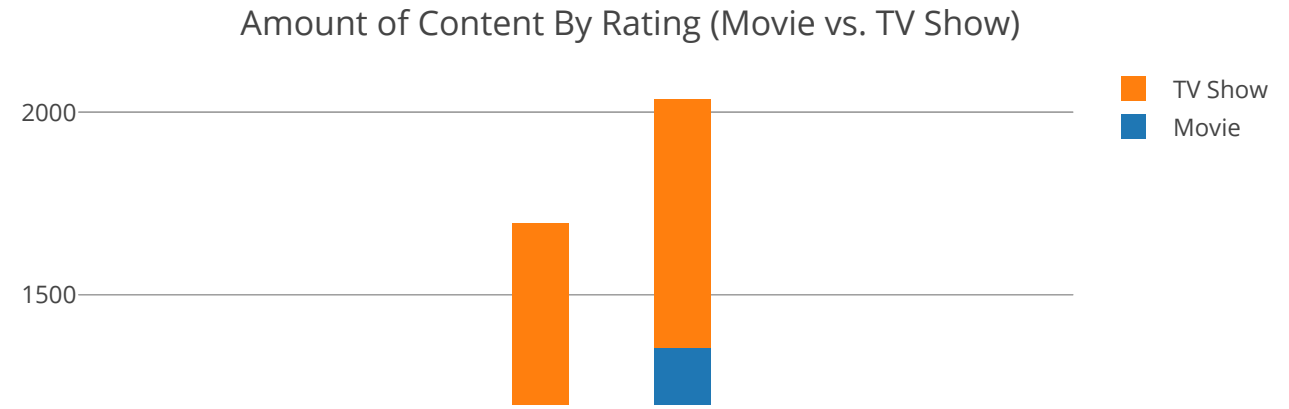


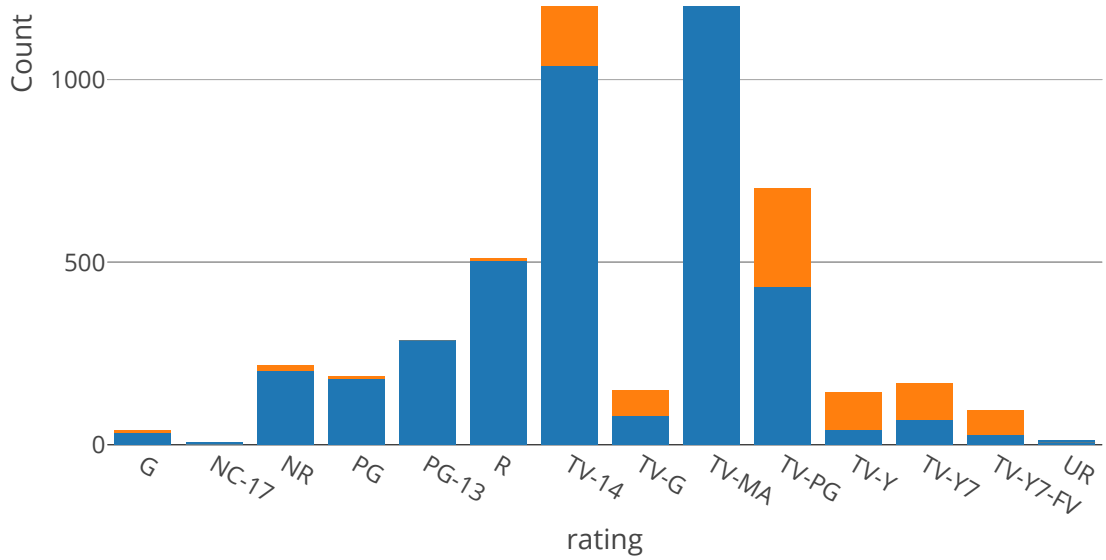
Amount of content by Rating (Movie vs. TV Show)

```
# data preparation
data2 <-netds$title %>%
  group_by(netds$rating, netds$type)%>%
  summarise(content_num = n())

names(data2) [1] <- "rating"
names(data2) [2] <- "type"
names(data2) [3] <- "content"

newdata2 <- reshape(data=data.frame(data2),idvar="rating",
                          v.names = "content",
                          timevar = "type",
                          direction="wide")



names(newdata2)[2] <- "Movie"
names(newdata2)[3] <- "TV Show"

#


newdata2$`TV Show`[is.na(newdata2$`TV Show`)] <- print(0)
```

```
## [1] 0
```

```
# visualisation

library(plotly)

rating <- newdata2$rating
Movie <- newdata2$Movie
Tv_Show <- newdata2$`TV Show`


figure3 <- plot_ly(newdata2, x = ~rating, y = ~Movie, type = 'bar', name = 'Movie')

figure3 <- figure3 %>% add_trace(y = ~Tv_Show, name = 'TV Show')

figure3 <- figure3 %>% layout(yaxis = list(title = 'Count'),
                         barmode = 'stack',
                         title="Amount of Content By Rating (Movie vs. TV Show)")

figure3
```





Top 20 Genres on NETFLIX

```
# data preparation

library(crayon)
```

```
##
## Attaching package: 'crayon'
```

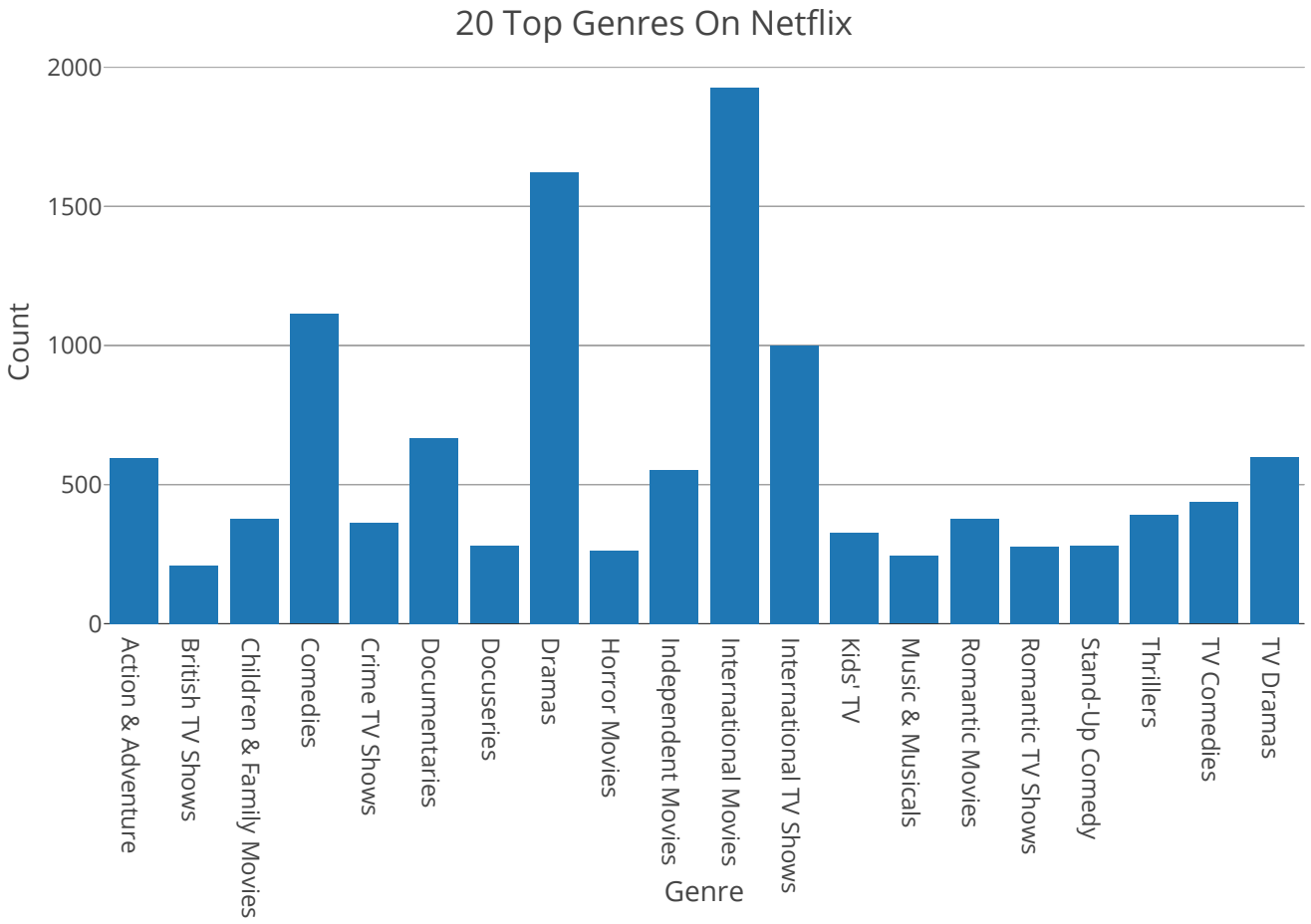
```
## The following object is masked from 'package:plotly':
##
##     style
```

```
## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```
# before apply to strsplit function, we have to make sure that type of the variable is charac
ter.

netds$listed_in<- as.character(netds$listed_in)

t20 <- strsplit(netds$listed_in, split = ", ")


count_listed_in<- data.frame(type = rep(netds$type,
                                        sapply(t20, length)),
                             listed_in = unlist(t20))


count_listed_in$listed_in <- as.character(gsub(",","",count_listed_in$listed_in))


df_count_listed_in <- count_listed_in %>%
                              group_by(listed_in) %>%
                              summarise(count = n()) %>%
                              top_n(20)
```

```
## Selecting by count
```

```
# visualisation

figure4 <- plot_ly(df_count_listed_in, x= ~listed_in, y= ~df_count_listed_in$count, type = "b
ar" )

figure4 <- figure4 %>% layout(xaxis=list(categoryorder = "array",
                                         categoryarray = df_count_listed_in$listed_in,
                                         title="Genre"), yaxis = list(title = 'Count'),
                                         title="20 Top Genres On Netflix")


figure4
```



Top 20 Directors By The Amount of Content on Netflix

```
# data preparation

dir20 <- strsplit(netds$director, split = ", ")

titles_director <-  data.frame(type= rep(netds$type, sapply(dir20, length)), director = unlis
t(dir20))

titles_director$director <- as.character(gsub(","," ", titles_director$director))

titles_director <- na.omit(titles_director) %>%
                          group_by(director) %>%
                          summarise(count = n()) %>%
                          arrange(desc(count)) %>%
                          top_n(20)
```

```
## Selecting by count
```

```
titles_director <- as.data.frame(titles_director)

library(tibble)

titles_director<- titles_director %>%
                  remove_rownames %>%
                  column_to_rownames(var = "director")




# visualisation as table

fig_table2 <- plot_ly(
  type = 'table',
  header = list(
    values = c( '<b>Director<b>', names(titles_director)),
  align = c('left', rep('center', ncol(titles_director))),
  line = list(width = 1, color = 'black'),
  fill = list(color = '#506784'),
  font = list(family = "Arial", size = 14, color = "white")
  ),
  cells = list(
    values = rbind(
      rownames(titles_director),
      t(as.matrix(unname(titles_director)))
    ),
    align = c('left', rep('center', ncol(titles_director))),
    line = list(color = "black", width = 1),
    fill = list(color = c('white')),
    font = list(family = "Arial", size = 12, color = c("black"))
  ))

fig_table2
```

Director	count
Jan Suter	21

Jan Suter	21
RaÃºl Campos	19
Jay Karas	14
Marcus Raboy	14
Jay Chapman	12
Martin Scorsese	9
Steven Spielberg	9
David Dhawan	8
Johnnie To	8
Lance Bangs	8
Shannon Hartman	8
Umesh Mehra	8
Cathy Garcia-Molina	7
Dibakar Banerjee	7
Hakan AlgÃ¼l	7
Noah Baumbach	7
Quentin Tarantino	7
Robert Rodriguez	7
Ryan Polito	7