Python and R LAB
University LUISS Guido Carli
Master of Science in Data Science and Management

Task 2: R

Group members: Cantelmo Carlotta - 746131, Imperatore Claudia- 753571, Lona Masriera Marian-746121

In this part, we carried out an analysis of the data through a series of graphical representations that allowed us to highlight the relationships between some of the variables.

```
install.packages (tidyverse)

packages = c("magrittr", 'lubridate', 'tidyverse')

installed_packages = packages %in% rownames(installed.packages())

if (any(installed_packages == FALSE)) {

  install.packages(packages[!installed_packages])

}
```

First, we have to clean the data set that we have chosen selecting the variables, cleaning the data, data manipulation and data analysis with ggplot.
Now we can import our dataset:

```
df <- read.csv('DF_1999movies.csv')

library(readxl)

df <- read_excel("film1999.xlsx")

View(df)
```

Then, we can have a general understanding of the data set, through the following functions:

```
glimpse(df)

head(df)
```

Now, we deselect the variables that are not useful for our analysis:

```
df <- df %>%

  select(-Response, -Error, -Poster, -Ratings, -imdbID, -Website, -X)
```

We can also continue to clean the data set checking for duplicates:

```
df <- df %>%

  distinct()
```

Checking for missing cells:
1. Calculating the product of dimensions of data frame

```
totalcells = prod(dim(df))
```

2. Calculating the number of cells with na

```
missingcells = sum(is.na(df))
```

3. Calculating the percentage of missing values

```
percentage = (missingcells * 100 )/(totalcells)

print("Percentage of missing values' cells")

print (percentage)
```

Now we ca continue with data manipulation.
1. Transforming 'Release' and DVD values into date:

```
library(lubridate)

df$Released <- dmy(df$Released)

NotR <- sum(is.na(df$Released))

length(df$Released)
```

1875/4890*100 → checking for percentage of missing values

```
df$DVD <- dmy(df$DVD)

notdvd <- sum(is.na(df$DVD))
```

length(df$DVD)

4061/4890 * 100 #83% of missing value

2. Transforming “Runtime” into integer:

name <- df$Runtime

df$Runtime <- as.integer(str_sub(name,1,nchar(name)-4))

length(df$Runtime)

narun <- sum(is.na(df$Runtime))

1468/4890*100 → 30% of missing values

3. Transforming “Boxoffice” in numeric

df$BoxOffice <- as.integer(gsub('[$,]', '', df$BoxOffice))

nabox <- sum(is.na(df$BoxOffice))

4492/4890 * 100 → 90% of missing value

4. Transforming “Imbdvotes” in numeric

df$imdbVotes <- as.integer(gsub('[,]', '', df$imdbVotes))

nabox <- sum(is.na(df$))

Continuing with the variables cleaning

1. Genre

str(df$Genre)

df$Genre <- as.factor(df$Genre)

table(df$Genre)

2. Year

range(df$Year)

3. Rated

length(df$Rated)

```
df$Rated <- as.factor(df$Rated) #risultano 4000 NAN

summary(df)

Rated.final1 <- na.omit(df$Rated)

sum(is.na(Rated.final1))

Rated.final2 <- subset(df$Awards, df$Awards!= "Not Rated" & df$Awards!=
"Unrated")

Rated.final <- c(Rated.final1, Rated.final2)
```

4. Awards

```
length(df$Awards)

Awards.final <- na.omit(df$Awards)

df$Awards <- as.factor((df$Awards))

length(Awards.final)

df$Awards <- fct_collapse(df$Awards, USA = c('USA', 'United States'))

USA = c('USA', 'United States')

fct_match(df$Genre, 'Horror')

genre1 <- as.character(df$Genre)

genre2<- str_extract(genre1, 'horror')

Genre <- fct_collapse(df$Genre, USA = c('USA', 'United States'))
```

5. Title

```
length(df$Title)
```

Are there empty cells?

```
is.na(df$Title)

str(df$Title)
```

6. Ratings

```
summary(df)
```

7. Country

```
library(forcats)

df$Country <- as.factor(df$Country)

df$Country <- fct_collapse(df$Country, USA = c('USA', 'United States'))
```

8. Language

```
length(df$Language)

is.na(df$Language)

Language.1 <- subset(df$Language, df$Language != "English")

Language.2 <- subset(df$Language, df$Language == "English")

Languages <- c(Language.1, Language.2)

str(Languages)
```

9. Director

```
length(df$Director)

is.na(df$Director)

str(df$Director)

summary(df)
```

Now we can start with some graphic representations:

Figure 1:

This chart is a bar plot that represent the "Runtime" frequency value. It is clear that we have a higher frequency in the first Runtime's values and this means that most movies last about 100 minutes. We use a simple bar plot function.

```
barplot(table(Runtime),  # frequency distribution of the variable Runtime

        col= c(4,5),

        main="Barplot of the Runtime",

        xlab = "runtime")
```

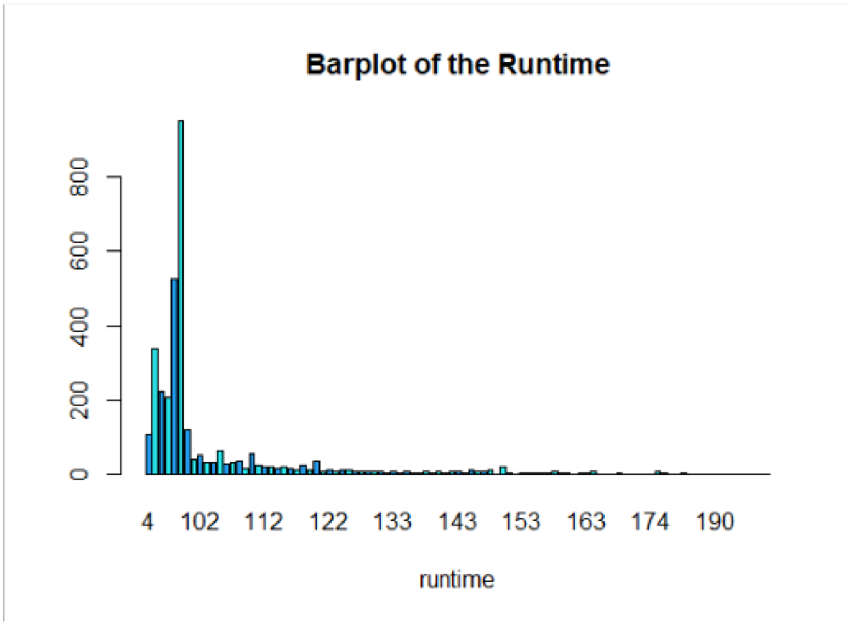


Figure 2:

This histogram represents the density about the first 50 values about the period in which the movie has been released.

```
Votes <- df$imdbVotes

str(Votes)

Votes_f <- as.numeric(Votes)

str(Votes_f)

votes <- Votes_f[1:50]

hist(votes,  # put the variable

     main="time of release",

     xlab= "released",

     freq = F,   # To have the density on the Y-axis (F stands for FALSE...)

     col="lightgreen")
```

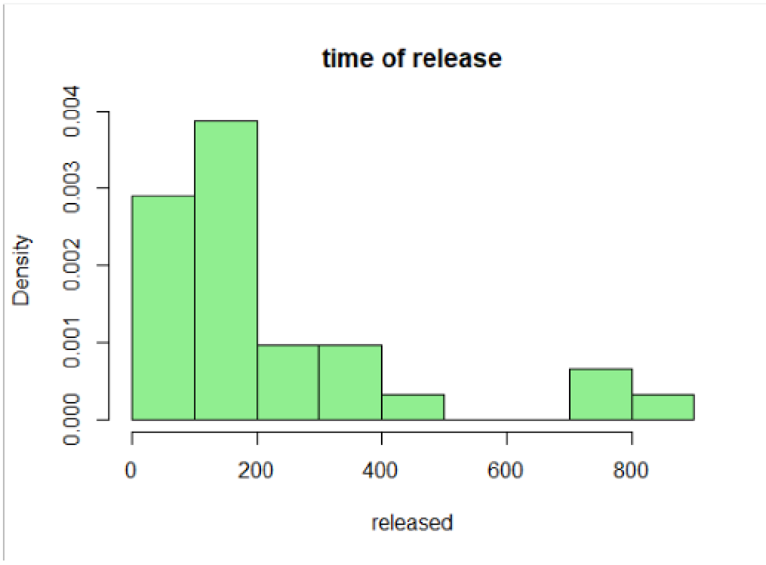


Figure 3:

In this chart we compute the frequency distribution of the first 10 “Genre” values from the data frame. The genre Drama is the one with the highest frequency distribution.

```
Genre10 %>%

  mutate(n = n/sum(n)) %>%

  ggplot(aes(x=reorder(Genre, -n),

             y= n,

             fill= Genre)) +

  geom_col()+

  labs(x='',y= 'Relative frequencies',subtitle= 'Relative frequencies of the top
10 genres', title='Top 10 genres')+

  theme_gray()
```

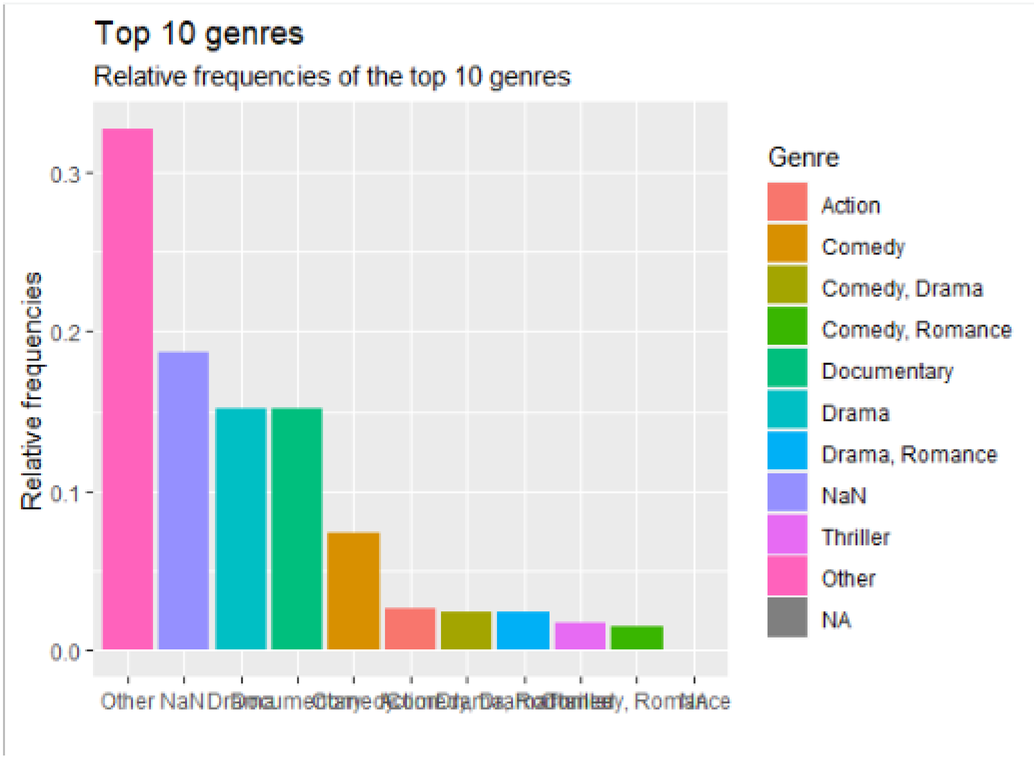


Figure 4:

In this scatterplot we try to represent the relationship between the Ratings and the Votes by IMDB. We use the first 10 values for both our variables because the observations were a lot and it was difficult to represent all of them.

```
Rating <- df$imdbRating

str(Rating)

Rating10 <- df %>%

  mutate(Rating = fct_lump(Rating, n = 9)) %>%

  count(Rating, sort = TRUE) %>%

  print(n = Inf)

Votes <- df$imdbVotes

str(Votes)

Votes10 <- df %>%

  mutate(Votes = fct_lump(Votes, n = 9)) %>%

  count(Votes, sort = TRUE) %>%
```

```
  print(n = Inf)
ggplot(df, aes(x= Votes, y=Rating )) +
  geom_point() +
  labs(x="Runtime",
       y="Rating",
       title = "Relationship between runtime and ratings") +
  theme_bw()
```

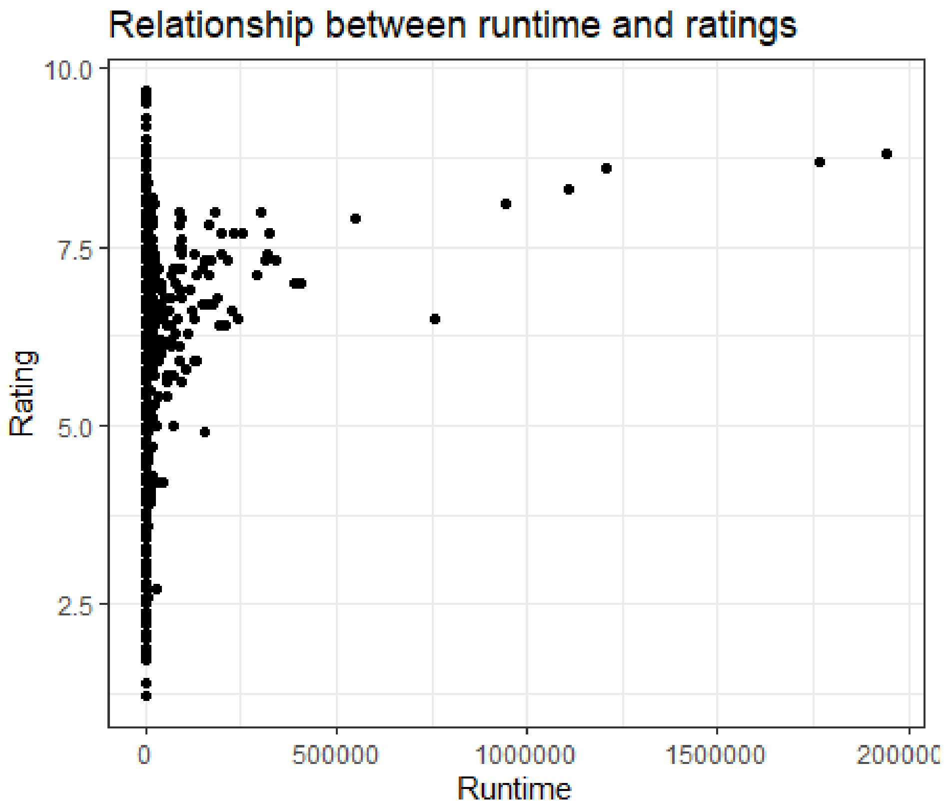
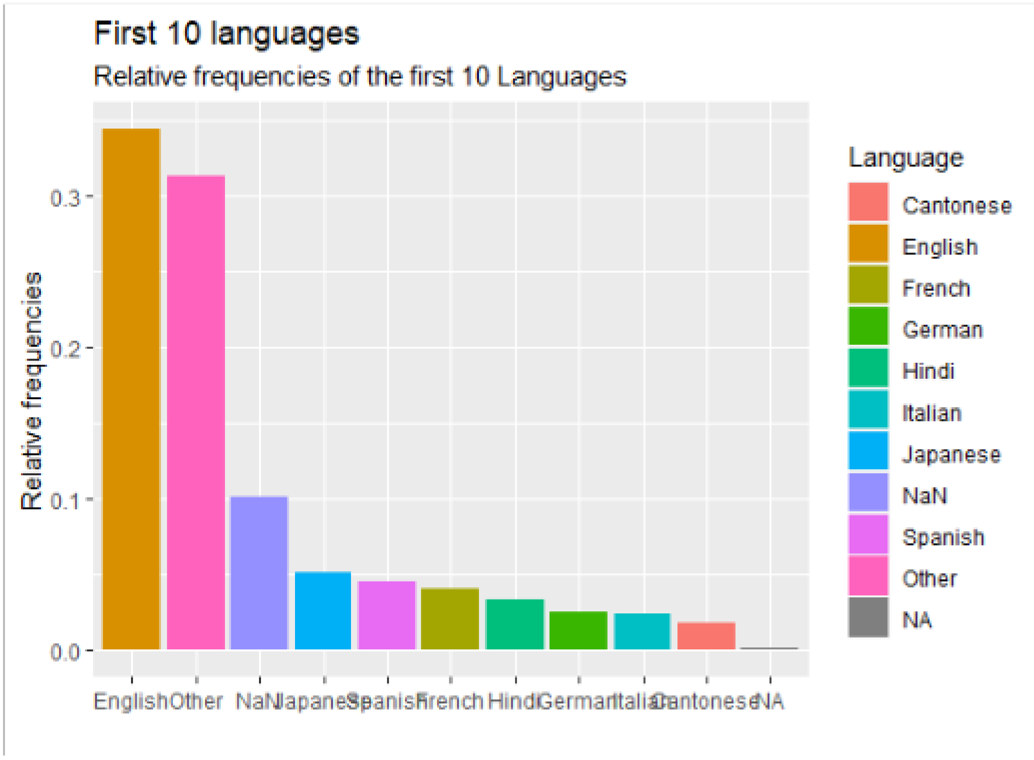


Figure 5:



In this bar plot, we use the first 10 values about the Language variable, and we compute the relative frequencies of these values. Clearly, the language with higher frequency is English.

```
Language <- df$Language

Language10 <- df %>%

  mutate(Language = fct_lump(Language, n = 9)) %>%

  count(Language, sort = TRUE) %>%

  print(n = Inf)

Language10 %>%

  mutate(n = n/sum(n)) %>%

  ggplot(aes(x=reorder(Language, -n),

             y= n,

             fill= Language)) +

  geom_col()+

  labs(x='', y= 'Relative frequencies', subtitle= 'Relative frequencies of the
first 10 Languages', title='First 10 languages')+
```

```
  theme_gray()
```

Figure 6: 'Top 10 Countries'

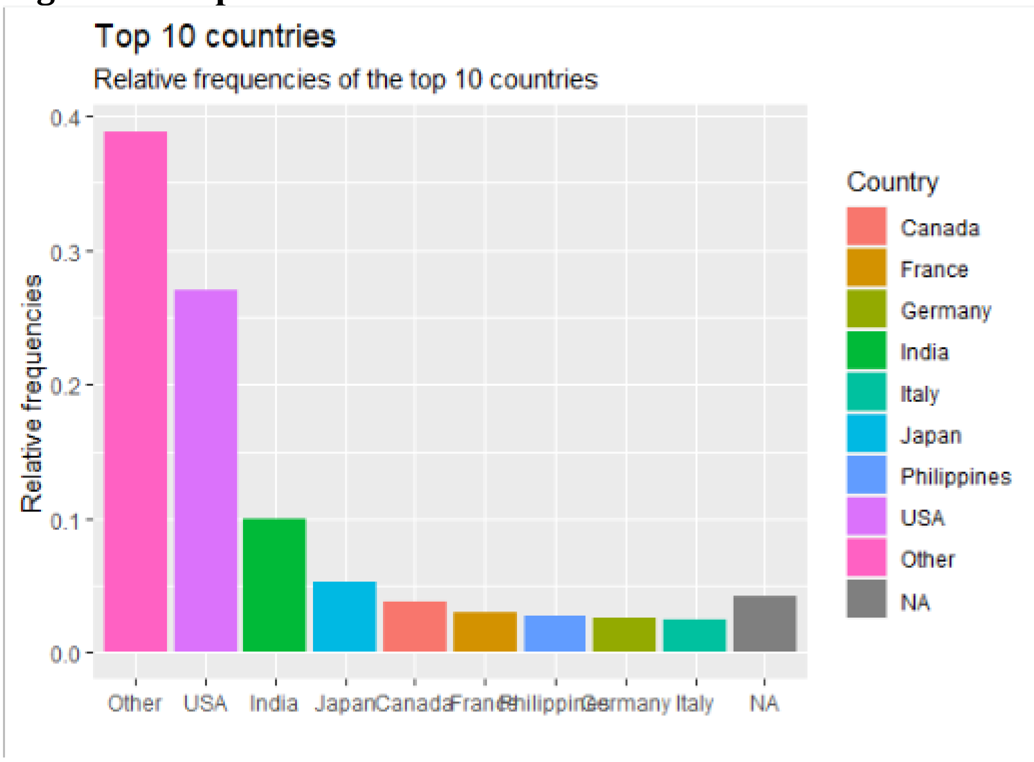


Fig 6 represents the top ten most frequent countries for the movie dataset.

The plot was obtained by selecting the 10 most frequent levels (8+'Other'+Na) in the column Countries through fct_lump() from the forcats library and by replacing the string 'NaN' with 'not available' data value.

The barplot was build thank to ggplot() and the geom_col() function was used. In order to display the relative frequencies, the number of occurrences were divided by the total sum (n). For the descending order the function reorder () was used. In order to fill the bars with different color the the function 'fill=Country' was adoperated.

For a clearer code, pipes and functions from the 'dplyr' package were used.

```
df$Country <- na_if(df$Country, 'NaN')

Country10 <- df %>%

  mutate(Country = fct_lump(Country, n = 8)) %>%

  count(Country, sort = TRUE) %>%

  print(n = Inf)
```

```
Country10 %>%
  mutate(n = n/sum(n)) %>%
  ggplot(aes(x=reorder(Country, -n),
             y= n,
             fill= Country)) +
  geom_col()+
  labs(x='',y= 'Relative frequencies',subtitle= 'Relative frequencies of the top 10 countries', title='Top 10 countries')+
  theme_gray()
```

Figure 7:

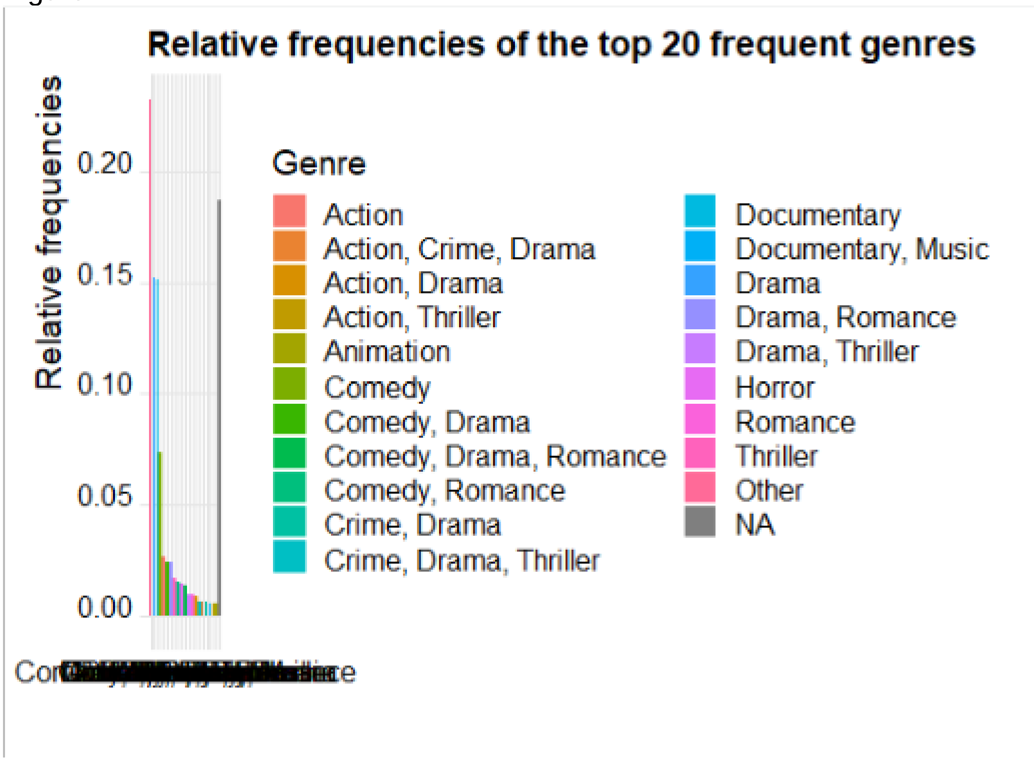


Fig 7 represents the top twenty most frequent movie genres for the movie dataset. The passages for figure 6 were used also for building this plot.

```
df$Genre <- na_if(df$Genre, 'NaN')

Genre20 <- df %>%
```

```
  mutate(Genre = fct_lump(Genre, n = 18)) %>%

  count(Genre, sort = TRUE) %>%

  print(n = Inf)

Genre20 %>%

  mutate(n= n/sum(n)) %>%

  ggplot(aes(x=reorder(Genre, -n), y= n, fill=Genre)) +

  geom_col() +

  labs(x='',y= 'Relative frequencies',title= 'Relative frequencies of the top 20
frequent genres')
```

The plot shows some levels that might be redundant, although they represent different genres, one might consider grouping the levels for a better visualization. The grouping choice was made according to the first genre displayed: if a level contained multiple genres, the first one was considered for the grouping. This passage was manually executed through the fct_collapse() from the forcats library. Again, pipes and mutate() were used for clearer code.

```
Genre20 %>%

  mutate(Genre = fct_collapse(Genre,

                          other = ('Other'),

                          Action = c('Action','Action, Crime, Drama', 'Action, Drama',
'Action, Thriller'),

                          Animation = ('Animation'),

                          Comedy = c('Comedy', 'Comedy, Drama', 'Comedy, Drama, Romance',
'Comedy, Romance'),

                          Crime = c("Crime", "Crime, Drama, Thriller", 'Crime, Drama'),

                          Documentary = c('Documentary','Documentary, Music'),

                          Drama= c('Drama', 'Drama, Romance', 'Drama, Thriller'),

                          Horror = ('Horror'),

                           Romance= ('Romance'),
```

```
                              Thriller = ('Thriller'))) %>%

  ggplot(aes(x=reorder(Genre, -n), y= n, fill=Genre)) +

  geom_col() +

  labs(x='',y= 'Relative frequencies',title= 'Relative frequencies of the top 20
frequent genres')
```

The plot after the transformation is way clearer and more intuitive:

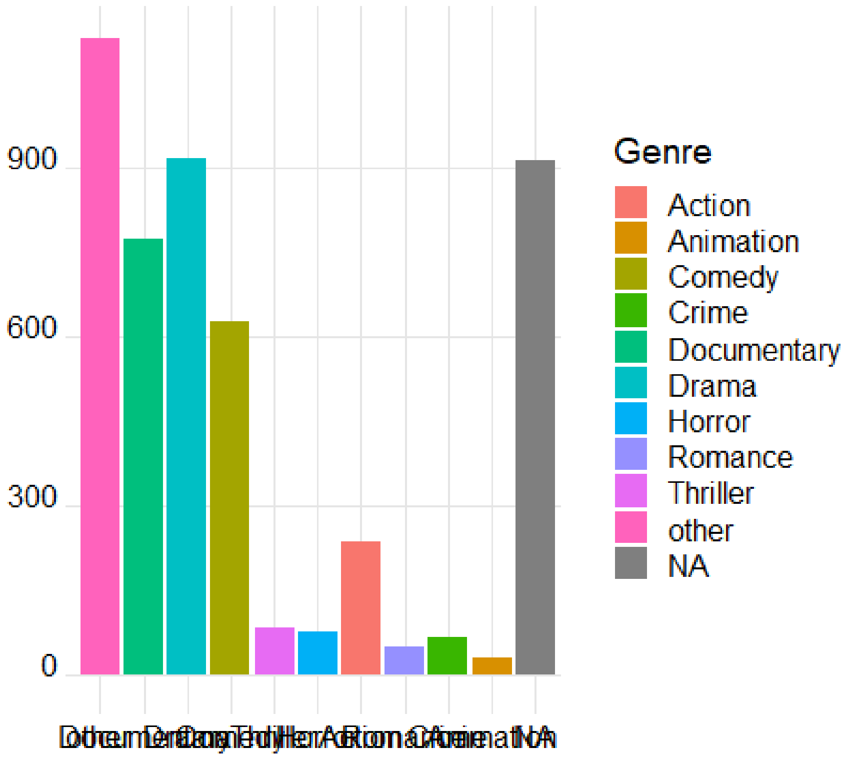


Figure 9: ‘Runtime per genres’

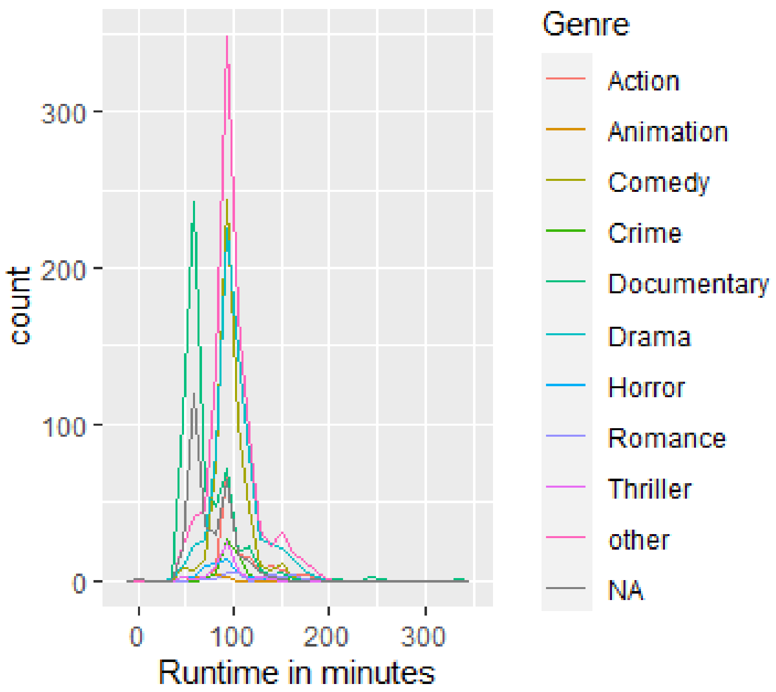
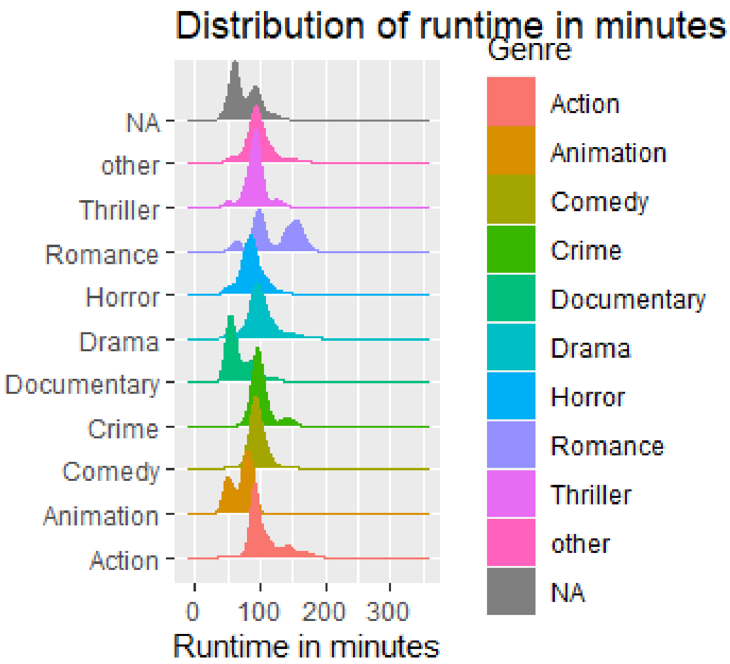


Figure 8 shows the runtime of the movies per genres. Again, pipe and mutate were used for a clearer code. Building upon the previous code, top 20 genres were filtered and then manually adjusted through fct_lump() and fct_collapse. In order to plot the frequency of the continuous variable ‘Runtime’, the *freqpoly plot* was used from the ggplot2 library. Runtime values were plotted on the x ax, runtimes’counts on the y and the genres differentiation was obtained through ‘color’, therefore coloring each line according to the genres.

```
df %>%

  mutate(Genre = fct_lump(Genre, n = 18)) %>%

  mutate(Genre = fct_collapse(Genre,

                         other = ('Other'),

                         Action = c('Action','Action, Crime, Drama', 'Action, Drama',
'Action, Thriller'),

                         Animation = ('Animation'),

                         Comedy = c('Comedy', 'Comedy, Drama', 'Comedy, Drama,
Romance', 'Comedy, Romance'),

                         Crime = c("Crime", "Crime, Drama, Thriller", 'Crime, Drama'),
```

```
                         Documentary = c('Documentary','Documentary, Music'),

                         Drama= c('Drama', 'Drama, Romance', 'Drama, Thriller'),

                         Horror = ('Horror'),

                         Romance= ('Romance'),

                         Thriller = ('Thriller'))) %>%

  ggplot(aes(Runtime, color=Genre))+

  geom_freqpoly(binwidth=5)+

  theme_gray()+

  labs(x='Runtime in minutes',y= '',title= 'Distribution of runtime in minutes of the
movies according to the genre')
```

Figure 9: 'Density of Runtime per genres'




```
install.packages('ggridges')

library(ggridges)

theme_set(theme_ridges())
```

Figure 8 could also be displayed through ‘geom_density_ridges’, a plot that is built in the ggridges library for ggplot2. The plot returns a density plot for each of the values in the color aesthetic. The default kernel is a Gaussian. It is clear that figure 9 provides a better visualization for multiple continuous values and it makes easy the comparisons.

For instance, thanks to the default Gaussian kernel it is possible to state that, for the movies in the dataset, on average the ‘Documentary’ have a shorter running time than the Crime one.

```
df %>%

  mutate(Genre = fct_lump(Genre, n = 18)) %>%

  mutate(Genre = fct_collapse(Genre,

                              other = ('Other'),

                              Action = c('Action','Action, Crime, Drama', 'Action, 
Drama', 'Action, Thriller'),

                              Animation = ('Animation'),

                              Comedy = c('Comedy', 'Comedy, Drama', 'Comedy, Drama, 
Romance', 'Comedy, Romance'),

                              Crime = c("Crime", "Crime, Drama, Thriller", 'Crime, 
Drama'),

                              Documentary = c('Documentary','Documentary, Music'),

                              Drama= c('Drama', 'Drama, Romance', 'Drama, Thriller'),

                              Horror = ('Horror'),

                              Romance= ('Romance'),

                              Thriller = ('Thriller'))) %>%

  ggplot(aes(y=Genre, x=Runtime, color=Genre))+

  geom_density_ridges(aes(fill=Genre))+

  theme_gray()+

  labs(x='Runtime in minutes',y= '',title= 'Distribution of runtime in minutes of the 
movies according to the genre')
```