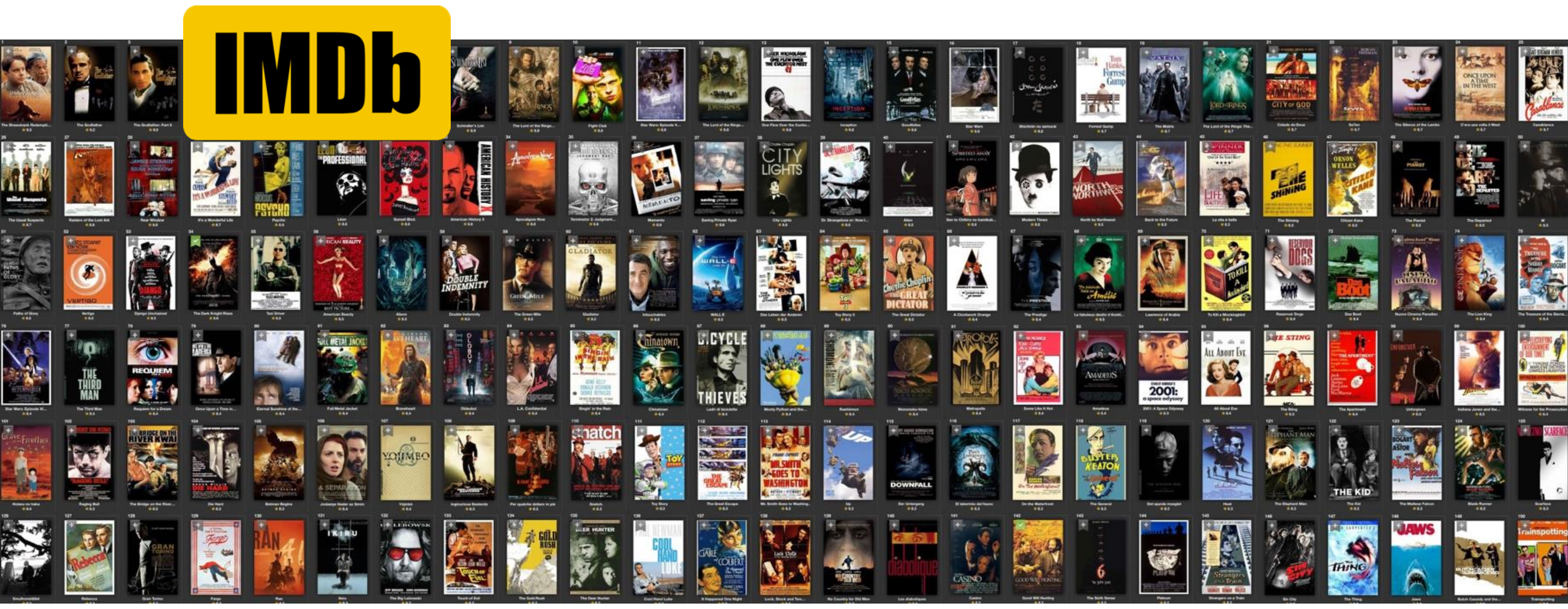
Bern University of Applied Sciences

Business Information Systems

DSWI

Animation and interaction

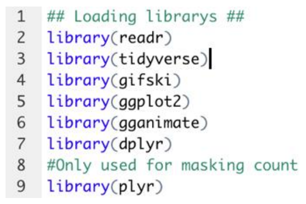
Midterm Group Work



Study programme	**BSc Business Information Systems**
Submitted by	**Birnbaumer Jamie-Lee, Djerrah Yacine, Zimmermann Yannik**
Subject area	**Data Science**
Expert	**Prof. Dr. Léonard Kwuida**
Datum of submission	**2. May 2021**

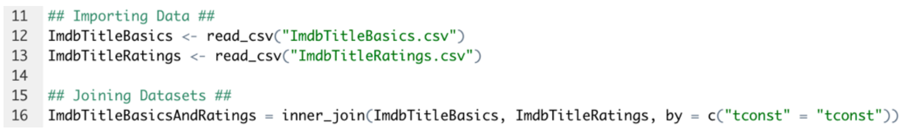
Content

1. Preparations



```r
## Loading librarys ##
library(readr)
library(tidyverse)
library(gifski)
library(ggplot2)
library(gganimate)
library(dplyr)
#Only used for masking count
library(plyr)
```

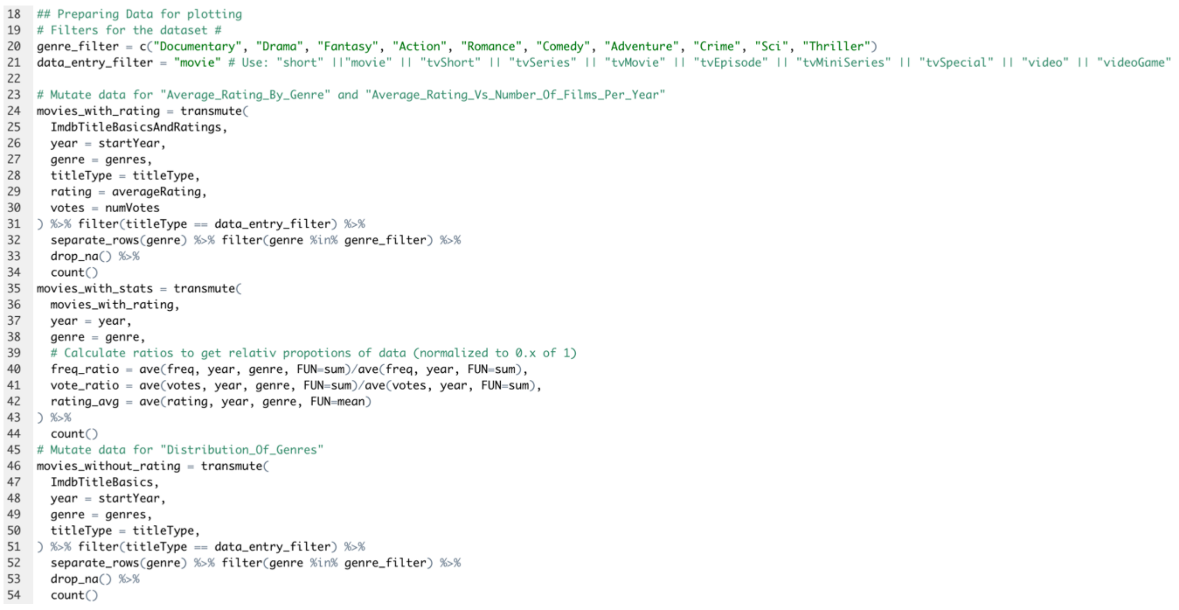
First, the necessary libraries must be initialised. Missing libraries can be installed via the Package Installer. To reproduce our tutorial, you will need the libraries listed on the left. Afterwards we will import our datasets. The source of the data is listed in the source directory. Generally, we work with a dataset from the movie database IMDb. More precisely, we use the data collections with all movie titles over more than 100 years as well as the corresponding movie ratings. We will be Inner-Joining the ImdbTitleRatings into ImdbTitleBasics Chose Inner-Join to make sure you will only insert datasets with an actual ratings value and skip the empty entries.



```r
## Importing Data ##
ImdbTitleBasics <- read_csv("ImdbTitleBasics.csv")
ImdbTitleRatings <- read_csv("ImdbTitleRatings.csv")

## Joining Datasets ##
ImdbTitleBasicsAndRatings = inner_join(ImdbTitleBasics, ImdbTitleRatings, by = c("tconst" = "tconst"))
```

The following code excerpt shows how the data is prepared to be displayed in the plots. To maintain a significance, the data is reduced to the 10 most common genres & types.

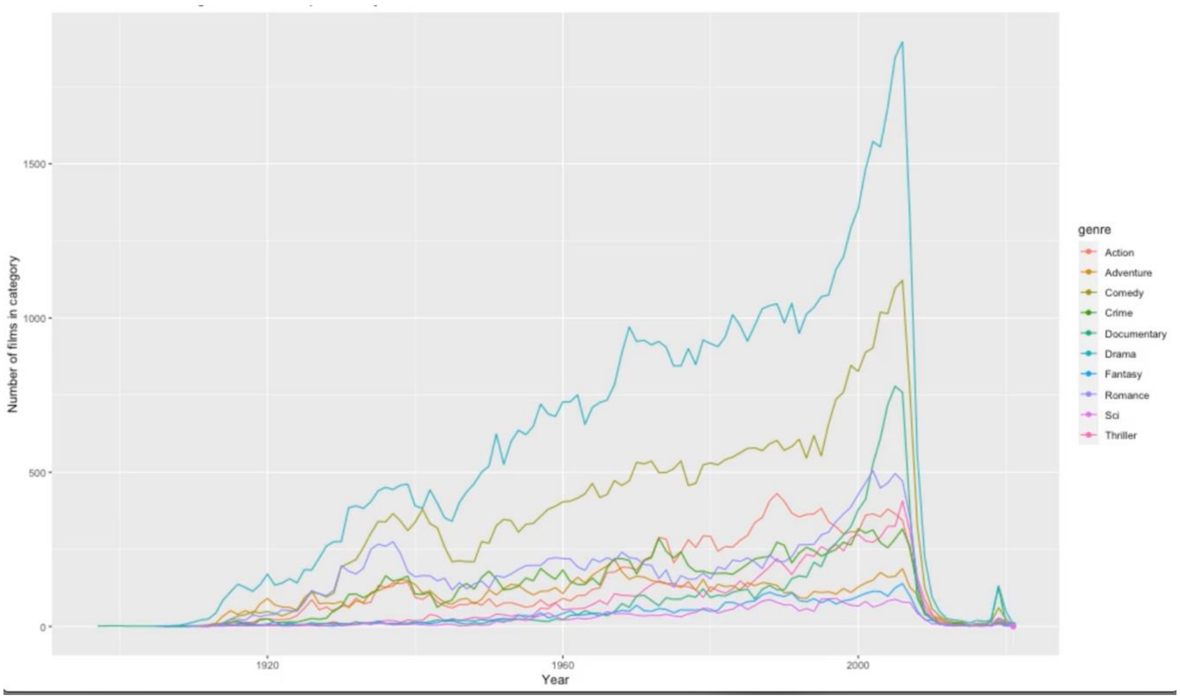


```r
## Preparing Data for plotting
# Filters for the dataset #
genre_filter = c("Documentary", "Drama", "Fantasy", "Action", "Romance", "Comedy", "Adventure", "Crime", "Sci", "Thriller")
data_entry_filter = "movie" # Use: "short" ||"movie" || "tvShort" || "tvSeries" || "tvMovie" || "tvEpisode" || "tvMiniSeries" || "tvSpecial" || "video" || "videoGame"

# Mutate data for "Average_Rating_By_Genre" and "Average_Rating_Vs_Number_Of_Films_Per_Year"
movies_with_rating = transmute(
  ImdbTitleBasicsAndRatings,
  year = startYear,
  genre = genres,
  titleType = titleType,
  rating = averageRating,
  votes = numVotes
) %>% filter(titleType == data_entry_filter) %>%
  separate_rows(genre) %>% filter(genre %in% genre_filter) %>%
  drop_na() %>%
  count()
movies_with_stats = transmute(
  movies_with_rating,
  year = year,
  genre = genre,
  # Calculate ratios to get relativ propotions of data (normalized to 0.x of 1)
  freq_ratio = ave(freq, year, genre, FUN=sum)/ave(freq, year, FUN=sum),
  vote_ratio = ave(votes, year, genre, FUN=sum)/ave(votes, year, FUN=sum),
  rating_avg = ave(rating, year, genre, FUN=mean)
) %>%
  count()
# Mutate data for "Distribution_Of_Genres"
movies_without_rating = transmute(
  ImdbTitleBasics,
  year = startYear,
  genre = genres,
  titleType = titleType,
) %>% filter(titleType == data_entry_filter) %>%
  separate_rows(genre) %>% filter(genre %in% genre_filter) %>%
  drop_na() %>%
  count()
```

2. Distribution of Movie Genres

In the first plot that we generate and animate, the animation runs along the X-axis and shows the progression of the number of movies in the respective genres over the years. From 2005 onwards, almost no data seems to have been collected by the originator of the data set. To make sure that our assumption is correct, we have made a request to the data originator, so far without a reply.



From line 72 to 78, we generate the line-plot using ggplot with the years on the X-axis (x=year) and the frequency on the Y-axis (y=freq). Further on the following lines, we add titles to each of the axes and the plot.

```
72  ## Distribution_Of_Genres -- Plot ##
73  Distribution_Of_Genres = ggplot(movies_without_rating, aes(x=year, y=freq, group=genre, color=genre)) +
74    geom_line() +
75    geom_point() +
76    ggtitle("Distribution of movie genres over the past 100 years") +
77    ylab("Number of films in category") +
78    xlab("Year")
```

Then we go on with the transitions. First, we define the animation progression in relation to the year. Then we add the required parameters such as duration, frames per second, width and height and save the animation as a GIF in the project directory.

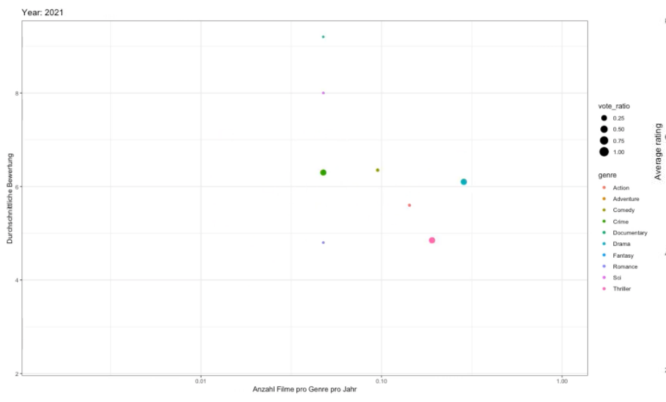
```
91  Animation_Distribution_Of_Genres = Distribution_Of_Genres +
92    transition_reveal(year) +
93    enter_fade() +
94    exit_shrink()
```

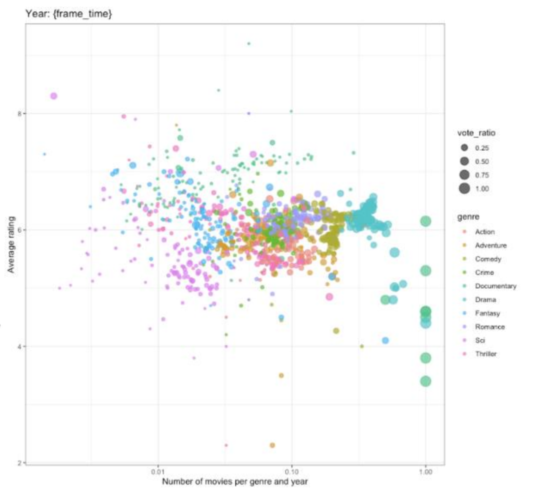
```
101 animate(Animation_Distribution_Of_Genres, duration = 10, fps = 20, width = 1000, height = 600, renderer = gifski_renderer())
102 anim_save("Distribution_Of_Genres.gif")
```

3. Average movie rating in Genres compared to the average number of movies produced within one year

The second plot, again from the ggplot library, is of the type geom_point, which results in the following plot. The animation represents a time progression and the points, which represent the genres each color one, vary in size depending on the proportion of votes.

Since the animation only shows the data per year (left picture), we attached the picture on the right, which shows the data of all the years, for a better understanding.





This scatterplot is generated with ggplot with a somewhat more complex axis description. We define here that the frequency of the films should be represented and that the average rating should vary in size. In addition, the genres should be displayed in different colors.

```r
## Average_Rating_Vs_Number_Of_Films_Per_Year -- Plot ##
Average_Rating_Vs_Number_Of_Films_Per_Year = ggplot(movies_with_stats, aes(freq_ratio, rating_avg, size = vote_ratio, color = genre)) +
  #geom_point() +
  geom_point(aes(size = vote_ratio), alpha = 0.7) +
  scale_x_log10() +
  theme_bw() +
  # gganimate specific bits
  labs(title = 'Year: {frame_time}', x = 'Number of movies per genre and year', y = 'Average rating')
```

In the following snippet we define the animation where the transitions run according to the year and we enter the data with *enter_fade()* and end by exiting the data with *exit_shrink()*.

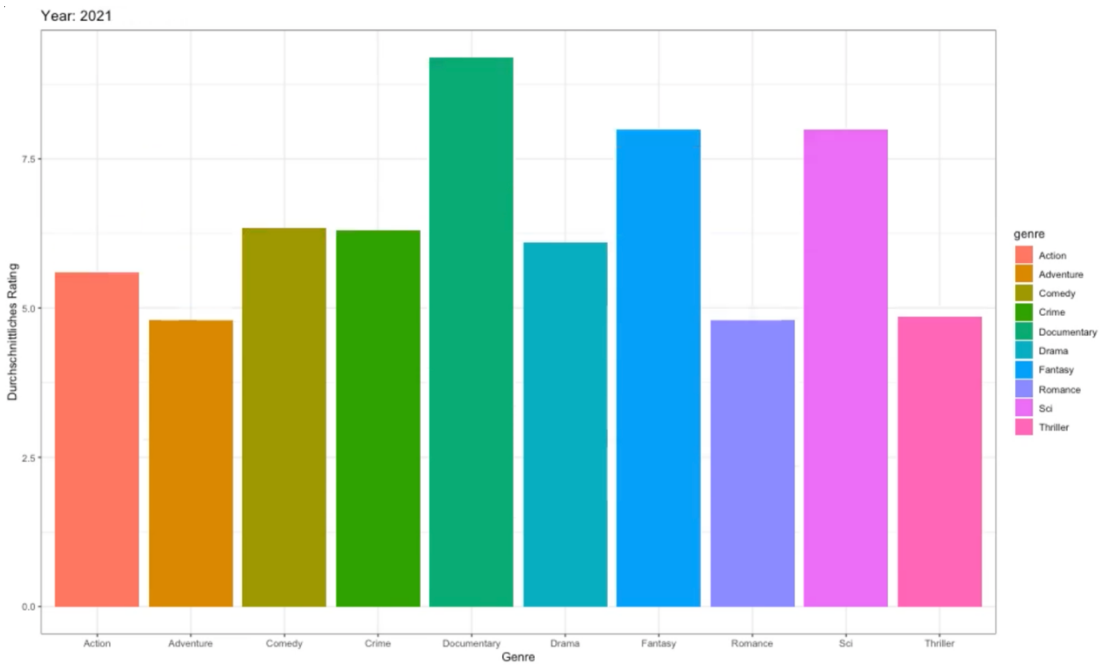
```r
Animation_Average_Rating_Vs_Number_Of_Films_Per_Year = Average_Rating_Vs_Number_Of_Films_Per_Year +
  transition_time(year) +
  ease_aes('linear') +
  enter_fade() +
  exit_shrink()
```

Here we generate and render the gif with gifski, give a name to the output file which will be saved in the project root folder.

```r
animate(Animation_Average_Rating_Vs_Number_Of_Films_Per_Year, duration = 30, fps = 20, width = 1000, height = 600, renderer = gifski_renderer())
anim_save("Average_Rating_Vs_Number_Of_Films_Per_Year.gif")
```

4. Average movie rating by Genres from 1897 to 2021

The third plot is a bar chart. Here the animation runs within the bars and represents a progression over more than 100 years. Shown are the average ratings (Y-axis) for the respective genres (X-axis) from 1897 to 2021.



Again, we generate the bar-plot using ggplot with the genre on the X-axis (x=genre) and the average rating on the Y-axis (y=rating_avg). We use the classic ggplot2 theme *theme_bw* and on line 61 we name the titles of each axis as well as the plot title, which will be the variable frame time in the animation, so the year.

```r
56  ## Average_Rating_By_Genre -- Plot ##
57  Average_Rating_By_Genre = ggplot(movies_with_stats, aes(x=genre, y=rating_avg, fill=genre)) +
58    geom_bar(stat='identity') +
59    theme_bw() +
60    # gganimate specific bits
61    labs(title = 'Year: {frame_time}', x = 'Genre', y = 'Average rating')
```

From line 81 to 85, we define and create the animation. The values within the animation change according to the year. And again, we enter and exit the data with the known commands.

```r
81  Animation_Average_Rating_By_Genre = Average_Rating_By_Genre +
82    transition_time(year) +
83    ease_aes('sine-in-out') +
84    enter_fade() +
85    exit_shrink()
```

And just like the previous plots, for the third plot, we generate an animated gif with the same animation attributes rendered with gifski and we save it with the mentioned file name in line 98 into the source folder.

```r
97  animate(Animation_Average_Rating_By_Genre, duration = 20, fps = 20, width = 1000, height = 600, renderer = gifski_renderer())
98  anim_save("Average_Rating_By_Genre.gif")
```

5. Sources

5.1. Dataset

«IMDb Dataset – From 1888 to 2023» (7th of April, 2021)

Opened 12th of April 2021,

From https://www.kaggle.com/komalkhetlani/imdb-dataset

5.2. Chart Animations and R Code

«the R Graph Gallery - Animation» (2018)

Opened 12th of April 2021,

From https://www.r-graph-gallery.com/animation.html

«gganimate: How to Create Plots with Beautiful Animation in R» (2018)

Opened 12th of April 2021,

From https://www.datanovia.com/en/blog/gganimate-how-to-create-plots-with-beautiful-animation-in-r/

6. Declaration

We confirm that we have written this work independently. All passages in the text that do not originate from us are marked as quotations and provided with an exact reference to their origin.
The sources used (also applies to illustrations, graphics, etc.) are listed in the bibliography or list of sources.

Birnbaumer Jamie-Lee	Djerrah Yacine	Zimmermann Yannik
[illegible]	[illegible]	[illegible]