Google Capstone Project - Movies Dataset Analysis

R Welsh

2024-10-16

link to the dataset used: https://www.kaggle.com/rounakbanik/the-movies-dataset

Inspiration for the Data Cleaning and Transformation part of this markdown: Movies Data Analysis by BarisBatuhan https://www.kaggle.com/code/barisbatuhan/movies-dataset-data-cleaning-and-analysis

Contents:

Business Objective Scenrio

I am a Junior Analyst who has been approached by a Film Investment Agency who wish to Invest a considerable amount into a film project in the near future, The Investment Agency has no bias towards a specific type of film or director they want to invest in they just want the best possible chance of return on their investment. The Investment Agency believes that analyzing previous film data could help them make a decision on their Investment. The insights I discover will then help guide the Investment strategy for the Agency. I will present my analysis along with my high-level recommendations back to the Investment Agency.

Importing Libraries I will need

- *Tidyverse* - for holding dataset and processing,list operations, graphics and data analysis.
- *Janitor* - for its Data Cleaning functions
- *Tinytex* - to make this markdown a PDF / Word document
- *Skimr* - for data manipulation
- *Reshape2* - for data visualisation purposes
- *Jsonlite* - if I need to clean up data from JSON strings
- *Corrplot* - in order to plot numerical correlations of data in graphs
- *Viridis* - to make visualisation clear and easy to read

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library(skimr)
library(tinytex)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:purrr':
##
##     flatten
```

```r
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```r
library(viridis)
```

```
## Loading required package: viridisLite
```

Importing the Data

```
# reads the csv metadata and prints the head

movies <- read.csv("movies_metadata.csv")

view(movies) # is print function but for purposes of markdown is view
```

Data Cleaning and Formatting

- Movies Metadata Dataset

```
head(movies)
```

```
##   adult
## 1 False
## 2 False
## 3 False
## 4 False
## 5 False
## 6 False
##
## 1           {'id': 10194, 'name': 'Toy Story Collection', 'poster_path': '/7G9915LfUQ2lVfwMEEhDsn3kT4
## 2
## 3     {'id': 119050, 'name': 'Grumpy Old Men Collection', 'poster_path': '/nLvUdqgPgm3F85NMCii9gVFUc
## 4
## 5 {'id': 96871, 'name': 'Father of the Bride Collection', 'poster_path': '/nts4iOmNnq7GNicycMJ9pSAn2
## 6
##     budget
## 1 30000000
## 2 65000000
## 3        0
## 4 16000000
## 5        0
## 6 60000000
##
## 1                          [{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 1075
## 2                         [{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}, {'id': 1075
## 3                                                  [{'id': 10749, 'name': 'Romance'}, {'id': 3
## 4                             [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}, {'id': 10749
## 5                                                                                          [{'id': 3
## 6 [{'id': 28, 'name': 'Action'}, {'id': 80, 'name': 'Crime'}, {'id': 18, 'name': 'Drama'}, {'id': 53
##                                homepage    id   imdb_id original_language
## 1 http://toystory.disney.com/toy-story   862 tt0114709                en
## 2                                       8844 tt0113497                en
## 3                                      15602 tt0113228                en
## 4                                      31357 tt0114885                en
## 5                                      11862 tt0113041                en
## 6                                        949 tt0113277                en
```

```
##                original_title
## 1                   Toy Story
## 2                     Jumanji
## 3            Grumpier Old Men
## 4           Waiting to Exhale
## 5 Father of the Bride Part II
## 6                        Heat
##
## 1                                                                                                                                  Led by
## 2 When siblings Judy and Peter discover an enchanted board game that opens the door to a magical wor
## 3                                                                                                  A family wedding reignites the
## 4
## 5                                                                                                               Just when George Bank
## 6                                                                                        Obsessive master thief, Neil McCauley
##   popularity                        poster_path
## 1  21.946943 /rhIRbceoE9lR4veEXuwCC2wARtG.jpg
## 2  17.015539 /vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg
## 3    11.7129 /6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg
## 4   3.859495 /16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg
## 5   8.387519 /e64sOI48hQXyru7naBFyssKFxVd.jpg
## 6  17.924927 /zMyfPUelumio3tiDKPffaUpsQTD.jpg
##
## 1                                                                                                                        [{'name': 'Pi
## 2 [{'name': 'TriStar Pictures', 'id': 559}, {'name': 'Teitler Film', 'id': 2550}, {'name': 'Interscop
## 3                                                                  [{'name': 'Warner Bros.', 'id': 6194}, {'name'
## 4                                                                                  [{'name': 'Twentieth Century Fo
## 5                                                [{'name': 'Sandollar Productions', 'id': 5842}, {'name': 'To
## 6                        [{'name': 'Regency Enterprises', 'id': 508}, {'name': 'Forward Pass', 'id': 675}, {'nam
##                                    production_countries release_date
## 1 [{'iso_3166_1': 'US', 'name': 'United States of America'}]   1995-10-30
## 2 [{'iso_3166_1': 'US', 'name': 'United States of America'}]   1995-12-15
## 3 [{'iso_3166_1': 'US', 'name': 'United States of America'}]   1995-12-22
## 4 [{'iso_3166_1': 'US', 'name': 'United States of America'}]   1995-12-22
## 5 [{'iso_3166_1': 'US', 'name': 'United States of America'}]   1995-02-10
## 6 [{'iso_3166_1': 'US', 'name': 'United States of America'}]   1995-12-15
##     revenue runtime
## 1 373554033      81
## 2 262797249     104
## 3         0     101
## 4  81452156     127
## 5  76578911     106
## 6 187436818     170
##                                                                                                spoken_languages
## 1                                                                                [{'iso_639_1': 'en', 'name': 'English'}]
## 2 [{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'fr', 'name': 'Français'}]
## 3                                                                                [{'iso_639_1': 'en', 'name': 'English'}]
## 4                                                                                [{'iso_639_1': 'en', 'name': 'English'}]
## 5                                                                                [{'iso_639_1': 'en', 'name': 'English'}]
## 6  [{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'es', 'name': 'Español'}]
##     status
## 1 Released
## 2 Released
## 3 Released
## 4 Released
```

```
## 5 Released
## 6 Released
##                                                                                 tagline
## 1
## 2                                        Roll the dice and unleash the excitement!
## 3                                Still Yelling. Still Fighting. Still Ready for Love.
## 4 Friends are the people who let you be yourself... and never let you forget it.
## 5 Just When His World Is Back To Normal... He's In For The Surprise Of His Life!
## 6                                                          A Los Angeles Crime Saga
##                          title video vote_average vote_count
## 1                    Toy Story False          7.7       5415
## 2                      Jumanji False          6.9       2413
## 3             Grumpier Old Men False          6.5         92
## 4            Waiting to Exhale False          6.1         34
## 5 Father of the Bride Part II False          5.7        173
## 6                         Heat False          7.7       1886
```

Columns to be Dropped

- *original_title*: since title column is also included and original_title column has non-ASCII characters, it can be dropped.
- *homepage*: there will be no analysis depending on the homepage of the movie, this column is uselesss for this specific analysis
- *imdb_id*: both ratings.csv and keywords.csv has id column to match with metadata dataset, thus no need for this column.
- *overview & tagline*: no text analysis will be made in this notebook. For retrieving the most important words, keywords.csv can be used
- *video & poster_path*: no image, video related processing will be made
- *spoken_languages*: original_language is included, no need for this column.

```
movies <- movies %>%
  select(-c(homepage, poster_path, video, imdb_id, overview, original_title, spoken_languages, tagline))

movies <- movies[!duplicated(movies), ]

movies <- movies[complete.cases(movies), ]
```

Time to get some info on this altered dataset

```
# To get the dimensions of your dataframe
dim(movies)
```

```
## [1] 45186    16
```

```
# To get the structure of your dataframe
str(movies)
```

```
## 'data.frame':    45186 obs. of  16 variables:
##  $ adult                : chr  "False" "False" "False" "False" ...
##  $ belongs_to_collection: chr  "{'id': 10194, 'name': 'Toy Story Collection', 'poster_path': '/7G9915
##  $ budget               : chr  "30000000" "65000000" "0" "16000000" ...
##  $ genres               : chr  "[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id
```

```
##  $ id                  : chr  "862" "8844" "15602" "31357" ...
##  $ original_language   : chr  "en" "en" "en" "en" ...
##  $ popularity          : chr  "21.946943" "17.015539" "11.7129" "3.859495" ...
##  $ production_companies : chr  "[{'name': 'Pixar Animation Studios', 'id': 3}]" "[{'name': 'TriStar
##  $ production_countries : chr  "[{'iso_3166_1': 'US', 'name': 'United States of America'}]" "[{'iso_
##  $ release_date        : chr  "1995-10-30" "1995-12-15" "1995-12-22" "1995-12-22" ...
##  $ revenue             : num  3.74e+08 2.63e+08 0.00 8.15e+07 7.66e+07 ...
##  $ runtime             : num  81 104 101 127 106 170 127 97 106 130 ...
##  $ status              : chr  "Released" "Released" "Released" "Released" ...
##  $ title               : chr  "Toy Story" "Jumanji" "Grumpier Old Men" "Waiting to Exhale" ...
##  $ vote_average        : num  7.7 6.9 6.5 6.1 5.7 7.7 6.2 5.4 5.5 6.6 ...
##  $ vote_count          : int  5415 2413 92 34 173 1886 141 45 174 1194 ...
```

I want to see if any rows have the film title missing

```r
# checking the title column for NA/Nulls
missing_titles <- movies[is.na(movies$title), ]

# Display the rows with missing titles
print(missing_titles)
```

```
##  [1] adult                 belongs_to_collection budget
##  [4] genres                id                    original_language
##  [7] popularity            production_companies  production_countries
## [10] release_date          revenue               runtime
## [13] status                title                 vote_average
## [16] vote_count
## <0 rows> (or 0-length row.names)
```

This comes back clear so I am good to go continuing my reformatting of the dataset, the string types of *id*, *popularity* and *budget* is *chr*, although they contain data that should be represented as numeric. This could cause errors with invalid parsing where results will be returned as Na. Also converting *release_date* to datetime instead of *chr* and extracting the year data will be helpful with future analysis.

```r
# Drop rows where 'title' is missing (already know this is 0 but run just as a precaution)
movies <- movies[!is.na(movies$title), ]

# Convert 'id' column to numeric
movies$id <- as.integer(movies$id)

# Convert 'popularity' column to numeric
movies$popularity <- as.numeric(movies$popularity)

# Convert 'budget' column to numeric
movies$budget <- as.numeric(movies$budget)

# Convert 'release_date' column to Date
movies$release_date <- as.Date(movies$release_date, format = "%Y-%m-%d")

# Extract the year from 'release_date' into 'release_year'
movies$release_year <- format(movies$release_date, "%Y")
```

As we can see from the dataset itself and the previous *str* function, *belongs_to_collection* column has too many null entries, therefore instead of giving the collection name, we can convert the data to 0 and 1, 0 for not belonging and 1 for belonging.

```
# Replace NA values in 'belongs_to_collection' with "None"
movies$belongs_to_collection[is.na(movies$belongs_to_collection)] <- "None"

# Convert 'belongs_to_collection' to binary: 1 if it's not "None", otherwise 0
movies$belongs_to_collection <- ifelse(movies$belongs_to_collection != "None", 1, 0)
```

From the previous *str* function and looking at the dataset their seems to be mostly false entered in the adult column. I want to find how many true values thier is to see if this column will be useful for further analysis

```
# Convert 'adult' column to logical so I can run the query (TRUE/FALSE)
movies$adult <- as.logical(movies$adult)

# Now, count the number of TRUE values in the 'adult' column
sum(movies$adult, na.rm = TRUE)
```

```
## [1] 9
```

I've found out that their is only 9 True values present in the adult column, this information will not give us anything significant, so that column is also dropped.

```
# Count the occurrences of each unique value in the 'adult' column
table(movies$adult)
```

```
##
## FALSE  TRUE
## 45177     9
```

```
# Removing the adult column
movies <- movies[, !names(movies) %in% "adult"]
```

```
# Viewing the dataset structure after this change
str(movies)
```

```
## 'data.frame':    45186 obs. of  16 variables:
##  $ belongs_to_collection: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ budget               : num  3.0e+07 6.5e+07 0.0 1.6e+07 0.0 6.0e+07 5.8e+07 0.0 3.5e+07 5.8e+07 .
##  $ genres               : chr  "[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id
##  $ id                   : int  862 8844 15602 31357 11862 949 11860 45325 9091 710 ...
##  $ original_language    : chr  "en" "en" "en" "en" ...
##  $ popularity           : num  21.95 17.02 11.71 3.86 8.39 ...
##  $ production_companies : chr  "[{'name': 'Pixar Animation Studios', 'id': 3}]" "[{'name': 'TriStar I
##  $ production_countries : chr  "[{'iso_3166_1': 'US', 'name': 'United States of America'}]" "[{'iso_3
##  $ release_date         : Date, format: "1995-10-30" "1995-12-15" ...
##  $ revenue              : num  3.74e+08 2.63e+08 0.00 8.15e+07 7.66e+07 ...
##  $ runtime              : num  81 104 101 127 106 170 127 97 106 130 ...
##  $ status               : chr  "Released" "Released" "Released" "Released" ...
##  $ title                : chr  "Toy Story" "Jumanji" "Grumpier Old Men" "Waiting to Exhale" ...
##  $ vote_average         : num  7.7 6.9 6.5 6.1 5.7 7.7 6.2 5.4 5.5 6.6 ...
##  $ vote_count           : int  5415 2413 92 34 173 1886 141 45 174 1194 ...
##  $ release_year         : chr  "1995" "1995" "1995" "1995" ...
```

I've noticed a few blank and null's in the *status* column, it may be a good idea to fill these with most common data. For *runtime*, again a similar case occurs and it can be handled by filling NaN values with the mean.

```
# Fill NA values in 'status' with the most common value
movies$status[is.na(movies$status)] <- names(sort(table(movies$status), decreasing = TRUE))[1]

# Replace 0 with NA in 'runtime'
movies$runtime[movies$runtime == 0] <- NA

# Fill NA values in 'runtime' with the mean
movies$runtime[is.na(movies$runtime)] <- mean(movies$runtime, na.rm = TRUE)
```

Now to check if their is any nulls in the *release_date* and *original_language* column's

```
# Find rows where 'release_date' or 'original_language' is missing
missing_data <- movies[is.na(movies$release_date) | is.na(movies$original_language), ]

view(missing_data)
```

This has returned 73 values, as I can only assume the original language and have no way of entering an exact correct release date I will remove these rows.

```
# Drop rows where 'release_date' is missing
movies <- movies[!is.na(movies$release_date), ]

# Drop rows where 'original_language' is missing
movies <- movies[!is.na(movies$original_language), ]
```

Some of my columns such as *genres*, *production_companies* and *production_countries* are *chr* strings. For easier processing and in order to perform any required list-based operations, these have to be converted into a list of inputs. The function below achieves this:

```
# Convert character strings in specified columns to lists of inputs
convert_to_list <- function(column) {
  lapply(column, function(x) {
    if (!is.na(x) && x != "") {
      strsplit(x, ", ")[[1]]
    } else {
      NA
    }
  })
}

# Apply the function to each column
movies$genres <- convert_to_list(movies$genres)
movies$production_companies <- convert_to_list(movies$production_companies)
movies$production_countries <- convert_to_list(movies$production_countries)

# Check the structure of the new columns to verify
str(movies$genres)
```

```
## List of 45113
```

```
##  $ : chr [1:6] "[{'id': 16" "'name': 'Animation'}" "{'id': 35" "'name': 'Comedy'}" ...
##  $ : chr [1:6] "[{'id': 12" "'name': 'Adventure'}" "{'id': 14" "'name': 'Fantasy'}" ...
##  $ : chr [1:4] "[{'id': 10749" "'name': 'Romance'}" "{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:2] "[{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:8] "[{'id': 28" "'name': 'Action'}" "{'id': 80" "'name': 'Crime'}" ...
##  $ : chr [1:4] "[{'id': 35" "'name': 'Comedy'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:8] "[{'id': 28" "'name': 'Action'}" "{'id': 12" "'name': 'Adventure'}" ...
##  $ : chr [1:6] "[{'id': 28" "'name': 'Action'}" "{'id': 12" "'name': 'Adventure'}" ...
##  $ : chr [1:6] "[{'id': 12" "'name': 'Adventure'}" "{'id': 28" "'name': 'Action'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:4] "[{'id': 35" "'name': 'Comedy'}" "{'id': 27" "'name': 'Horror'}]"
##  $ : chr [1:6] "[{'id': 10751" "'name': 'Family'}" "{'id': 16" "'name': 'Animation'}" ...
##  $ : chr [1:4] "[{'id': 36" "'name': 'History'}" "{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:4] "[{'id': 28" "'name': 'Action'}" "{'id': 12" "'name': 'Adventure'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 80" "'name': 'Crime'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:4] "[{'id': 80" "'name': 'Crime'}" "{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:6] "[{'id': 80" "'name': 'Crime'}" "{'id': 35" "'name': 'Comedy'}" ...
##  $ : chr [1:6] "[{'id': 28" "'name': 'Action'}" "{'id': 35" "'name': 'Comedy'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 53" "'name': 'Thriller'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 53" "'name': 'Thriller'}]"
##  $ : chr [1:8] "[{'id': 28" "'name': 'Action'}" "{'id': 12" "'name': 'Adventure'}" ...
##  $ : chr [1:8] "[{'id': 18" "'name': 'Drama'}" "{'id': 14" "'name': 'Fantasy'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:2] "[{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:6] "[{'id': 14" "'name': 'Fantasy'}" "{'id': 878" "'name': 'Science Fiction'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 80" "'name': 'Crime'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 80" "'name': 'Crime'}]"
##  $ : chr [1:6] "[{'id': 878" "'name': 'Science Fiction'}" "{'id': 53" "'name': 'Thriller'}" ...
##  $ : chr [1:4] "[{'id': 10749" "'name': 'Romance'}" "{'id': 12" "'name': 'Adventure'}]"
##  $ : chr [1:8] "[{'id': 14" "'name': 'Fantasy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:6] "[{'id': 36" "'name': 'History'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:2] "[{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:8] "[{'id': 12" "'name': 'Adventure'}" "{'id': 36" "'name': 'History'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 10751" "'name': 'Family'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:2] "[{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10752" "'name': 'War'}]"
##  $ : chr [1:8] "[{'id': 28" "'name': 'Action'}" "{'id': 80" "'name': 'Crime'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:4] "[{'id': 28" "'name': 'Action'}" "{'id': 14" "'name': 'Fantasy'}]"
##  $ : chr [1:8] "[{'id': 14" "'name': 'Fantasy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:6] "[{'id': 80" "'name': 'Crime'}" "{'id': 9648" "'name': 'Mystery'}" ...
##  $ : chr [1:8] "[{'id': 12" "'name': 'Adventure'}" "{'id': 16" "'name': 'Animation'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:6] "[{'id': 18" "'name': 'Drama'}" "{'id': 80" "'name': 'Crime'}" ...
##  $ : chr [1:6] "[{'id': 28" "'name': 'Action'}" "{'id': 53" "'name': 'Thriller'}" ...
##  $ : chr [1:4] "[{'id': 35" "'name': 'Comedy'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10769" "'name': 'Foreign'}]"
##  $ : chr [1:8] "[{'id': 28" "'name': 'Action'}" "{'id': 12" "'name': 'Adventure'}" ...
```

```
##  $ : chr [1:2] "[{'id': 18" "'name': 'Drama'}]"
##  $ : chr "[]"
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:6] "[{'id': 53" "'name': 'Thriller'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:6] "[{'id': 12" "'name': 'Adventure'}" "{'id': 10751" "'name': 'Family'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 53" "'name': 'Thriller'}]"
##  $ : chr [1:6] "[{'id': 10402" "'name': 'Music'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:2] "[{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:4] "[{'id': 35" "'name': 'Comedy'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:2] "[{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:4] "[{'id': 28" "'name': 'Action'}" "{'id': 878" "'name': 'Science Fiction'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10751" "'name': 'Family'}]"
##  $ : chr [1:4] "[{'id': 35" "'name': 'Comedy'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:2] "[{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:8] "[{'id': 27" "'name': 'Horror'}" "{'id': 28" "'name': 'Action'}" ...
##  $ : chr [1:6] "[{'id': 28" "'name': 'Action'}" "{'id': 53" "'name': 'Thriller'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 36" "'name': 'History'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:4] "[{'id': 35" "'name': 'Comedy'}" "{'id': 10751" "'name': 'Family'}]"
##  $ : chr [1:4] "[{'id': 27" "'name': 'Horror'}" "{'id': 878" "'name': 'Science Fiction'}]"
##  $ : chr [1:2] "[{'id': 99" "'name': 'Documentary'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 53" "'name': 'Thriller'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 53" "'name': 'Thriller'}]"
##  $ : chr [1:4] "[{'id': 10751" "'name': 'Family'}" "{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 80" "'name': 'Crime'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:4] "[{'id': 10749" "'name': 'Romance'}" "{'id': 18" "'name': 'Drama'}]"
##  $ : chr "[]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 10749" "'name': 'Romance'}]"
##  $ : chr [1:4] "[{'id': 28" "'name': 'Action'}" "{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:6] "[{'id': 12" "'name': 'Adventure'}" "{'id': 35" "'name': 'Comedy'}" ...
##  $ : chr [1:2] "[{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:6] "[{'id': 80" "'name': 'Crime'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:2] "[{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:8] "[{'id': 18" "'name': 'Drama'}" "{'id': 27" "'name': 'Horror'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 27" "'name': 'Horror'}" ...
##  $ : chr [1:6] "[{'id': 35" "'name': 'Comedy'}" "{'id': 18" "'name': 'Drama'}" ...
##  $ : chr [1:8] "[{'id': 28" "'name': 'Action'}" "{'id': 12" "'name': 'Adventure'}" ...
##  $ : chr [1:2] "[{'id': 35" "'name': 'Comedy'}]"
##  $ : chr [1:2] "[{'id': 18" "'name': 'Drama'}]"
##  $ : chr [1:10] "[{'id': 28" "'name': 'Action'}" "{'id': 12" "'name': 'Adventure'}" ...
##  $ : chr [1:2] "[{'id': 99" "'name': 'Documentary'}]"
##  $ : chr [1:4] "[{'id': 18" "'name': 'Drama'}" "{'id': 53" "'name': 'Thriller'}]"
##    [list output truncated]
```

```
str(movies$production_companies)
```

```
## List of 45113
##  $ : chr [1:2] "[{'name': 'Pixar Animation Studios'" "'id': 3}]"
##  $ : chr [1:6] "[{'name': 'TriStar Pictures'" "'id': 559}" "{'name': 'Teitler Film'" "'id': 2550}" .
##  $ : chr [1:4] "[{'name': 'Warner Bros.'" "'id': 6194}" "{'name': 'Lancaster Gate'" "'id': 19464}]"
##  $ : chr [1:2] "[{'name': 'Twentieth Century Fox Film Corporation'" "'id': 306}]"
```

```
##  $ : chr [1:4] "[{'name': 'Sandollar Productions'" "'id': 5842}" "{'name': 'Touchstone Pictures'" "'i
##  $ : chr [1:6] "[{'name': 'Regency Enterprises'" "'id': 508}" "{'name': 'Forward Pass'" "'id': 675}"
##  $ : chr [1:14] "[{'name': 'Paramount Pictures'" "'id': 4}" "{'name': 'Scott Rudin Productions'" "'id
##  $ : chr [1:2] "[{'name': 'Walt Disney Pictures'" "'id': 2}]"
##  $ : chr [1:6] "[{'name': 'Universal Pictures'" "'id': 33}" "{'name': 'Imperial Entertainment'" "'id
##  $ : chr [1:4] "[{'name': 'United Artists'" "'id': 60}" "{'name': 'Eon Productions'" "'id': 7576}]"
##  $ : chr [1:4] "[{'name': 'Columbia Pictures'" "'id': 5}" "{'name': 'Castle Rock Entertainment'" "'id
##  $ : chr [1:6] "[{'name': 'Columbia Pictures'" "'id': 5}" "{'name': 'Castle Rock Entertainment'" "'id
##  $ : chr [1:6] "[{'name': 'Universal Pictures'" "'id': 33}" "{'name': 'Amblin Entertainment'" "'id':
##  $ : chr [1:4] "[{'name': 'Hollywood Pictures'" "'id': 915}" "{'name': 'Cinergi Pictures Entertainmen
##  $ : chr [1:8] "[{'name': 'Le Studio Canal+'" "'id': 183}" "{'name': 'Laurence Mark Productions'" "'i
##  $ : chr [1:8] "[{'name': 'Universal Pictures'" "'id': 33}" "{'name': 'Légende Entreprises'" "'id': 3
##  $ : chr [1:4] "[{'name': 'Columbia Pictures Corporation'" "'id': 441}" "{'name': 'Mirage Enterprises
##  $ : chr [1:4] "[{'name': 'Miramax Films'" "'id': 14}" "{'name': 'A Band Apart'" "'id': 59}]"
##  $ : chr [1:6] "[{'name': 'O Entertainment'" "'id': 5682}" "{'name': 'Warner Bros.'" "'id': 6194}" .
##  $ : chr [1:2] "[{'name': 'Columbia Pictures'" "'id': 5}]"
##  $ : chr [1:4] "[{'name': 'Jersey Films'" "'id': 216}" "{'name': 'Metro-Goldwyn-Mayer (MGM)'" "'id':
##  $ : chr [1:4] "[{'name': 'Regency Enterprises'" "'id': 508}" "{'name': 'New Regency Pictures'" "'id
##  $ : chr [1:12] "[{'name': 'Silver Pictures'" "'id': 1885}" "{'name': 'Canal+'" "'id': 5358}" ...
##  $ : chr [1:4] "[{'name': 'Caravan Pictures'" "'id': 175}" "{'name': 'Hollywood Pictures'" "'id': 915
##  $ : chr [1:4] "[{'name': 'United Artists'" "'id': 60}" "{'name': 'Intial Productions'" "'id': 15547
##  $ : chr [1:4] "[{'name': 'Columbia Pictures'" "'id': 5}" "{'name': 'Castle Rock Entertainment'" "'id
##  $ : chr [1:2] "[{'name': 'New Line Cinema'" "'id': 12}]"
##  $ : chr [1:2] "[{'name': 'BBC Films'" "'id': 288}]"
##  $ : chr [1:42] "[{'name': 'Procirep'" "'id': 311}" "{'name': 'Constellation Productions'" "'id': 590
##  $ : chr [1:6] "[{'name': 'Ministère des Affaires Étrangères'" "'id': 2588}" "{'name': 'Alpha Films'
##  $ : chr [1:6] "[{'name': 'Hollywood Pictures'" "'id': 915}" "{'name': 'Via Rosa Productions'" "'id'
##  $ : chr [1:6] "[{'name': 'Universal Pictures'" "'id': 33}" "{'name': 'Atlas Entertainment'" "'id': 5
##  $ : chr [1:2] "[{'name': 'Iwerks Entertainment'" "'id': 70801}]"
##  $ : chr [1:4] "[{'name': 'Universal Pictures'" "'id': 33}" "{'name': 'Kennedy Miller Productions'" "
##  $ : chr [1:14] "[{'name': 'StudioCanal'" "'id': 694}" "{'name': 'Cinéa'" "'id': 874}" ...
##  $ : chr [1:6] "[{'name': 'Havoc'" "'id': 406}" "{'name': 'PolyGram Filmed Entertainment'" "'id': 138
##  $ : chr [1:2] "[{'name': 'Sony New Technologies'" "'id': 53390}]"
##  $ : chr [1:8] "[{'name': 'Dualstar Productions'" "'id': 6130}" "{'name': 'Warner Bros.'" "'id': 6194
##  $ : chr [1:2] "[{'name': 'Paramount Pictures'" "'id': 4}]"
##  $ : chr [1:6] "[{'name': 'Miramax'" "'id': 53009}" "{'name': 'Distant Horizon'" "'id': 85681}" ...
##  $ : chr [1:4] "[{'name': 'British Screen'" "'id': 871}" "{'name': 'Mayfair Entertainment Internation
##  $ : chr [1:6] "[{'name': 'Caravan Pictures'" "'id': 175}" "{'name': 'Underworld Entertainment'" "'id
##  $ : chr [1:2] "[{'name': 'Miramax Films'" "'id': 14}]"
##  $ : chr [1:4] "[{'name': 'New Line Cinema'" "'id': 12}" "{'name': 'Threshold Entertainment'" "'id':
##  $ : chr [1:4] "[{'name': 'The Rank Organisation'" "'id': 364}" "{'name': 'Columbia Pictures Corporat
##  $ : chr [1:4] "[{'name': 'Universal Pictures'" "'id': 33}" "{'name': 'Amblin Entertainment'" "'id':
##  $ : chr [1:6] "[{'name': 'New Line Cinema'" "'id': 12}" "{'name': 'Juno Pix'" "'id': 4286}" ...
##  $ : chr [1:4] "[{'name': 'Walt Disney Pictures'" "'id': 2}" "{'name': 'Walt Disney Feature Animation
##  $ : chr [1:2] "[{'name': 'Alliance Communications Corporation'" "'id': 2480}]"
##  $ : chr [1:4] "[{'name': 'Blue Parrot Productions'" "'id': 361}" "{'name': 'Bad Hat Harry Production
##  $ : chr "[]"
##  $ : chr [1:4] "[{'name': 'Miramax Films'" "'id': 14}" "{'name': 'Magnolia Pictures'" "'id': 1030}]"
##  $ : chr "[]"
##  $ : chr [1:4] "[{'name': 'Walt Disney Pictures'" "'id': 2}" "{'name': 'Caravan Pictures'" "'id': 175
##  $ : chr [1:4] "[{'name': 'Miramax Films'" "'id': 14}" "{'name': 'CiBy 2000'" "'id': 7832}]"
##  $ : chr [1:4] "[{'name': 'Telefilm Canada'" "'id': 7320}" "{'name': 'Melenny Productions'" "'id': 10
##  $ : chr [1:6] "[{'name': 'Paramount Pictures'" "'id': 4}" "{'name': 'Egg Pictures'" "'id': 802}" ..
##  $ : chr "[]"
```

```
##  $ : chr "[]"
##  $ : chr [1:6] "[{'name': 'Paramount Pictures'" "'id': 4}" "{'name': 'Columbia Pictures Corporation'
##  $ : chr [1:2] "[{'name': 'Paramount Pictures'" "'id': 4}]"
##  $ : chr [1:8] "[{'name': 'Hollywood Pictures'" "'id': 915}" "{'name': 'The Charlie Mopic Company'" "
##  $ : chr [1:6] "[{'name': 'Miramax Films'" "'id': 14}" "{'name': 'Island Pictures'" "'id': 3492}" ..
##  $ : chr [1:4] "[{'name': 'Warner Bros.'" "'id': 6194}" "{'name': 'Morgan Creek Productions'" "'id':
##  $ : chr [1:8] "[{'name': 'Metro-Goldwyn-Mayer (MGM)'" "'id': 8411}" "{'name': 'Motion Picture Corpo
##  $ : chr [1:2] "[{'name': 'New Line Cinema'" "'id': 12}]"
##  $ : chr [1:2] "[{'name': 'Miramax Films'" "'id': 14}]"
##  $ : chr [1:8] "[{'name': 'Renn Productions'" "'id': 82}" "{'name': 'Les Films Flam'" "'id': 1614}"
##  $ : chr [1:2] "[{'name': 'New Line Cinema'" "'id': 12}]"
##  $ : chr [1:8] "[{'name': 'A Band Apart'" "'id': 59}" "{'name': 'Dimension Films'" "'id': 7405}" ...
##  $ : chr [1:4] "[{'name': 'Silver Pictures'" "'id': 1885}" "{'name': 'Warner Bros.'" "'id': 6194}]"
##  $ : chr [1:4] "[{'name': 'Castleberg Productions'" "'id': 5841}" "{'name': 'Sandollar Productions'"
##  $ : chr [1:6] "[{'name': 'Les Films 13'" "'id': 1742}" "{'name': 'Canal+'" "'id': 5358}" ...
##  $ : chr [1:4] "[{'name': 'New Line Cinema'" "'id': 12}" "{'name': 'Juno Pix'" "'id': 4286}]"
##  $ : chr [1:4] "[{'name': 'Warner Bros.'" "'id': 6194}" "{'name': 'Morgan Creek Productions'" "'id':
##  $ : chr [1:12] "[{'name': 'The Movie Network (TMN)'" "'id': 3388}" "{'name': 'Allegro Films'" "'id'
##  $ : chr [1:6] "[{'name': 'Zweites Deutsches Fernsehen (ZDF)'" "'id': 4606}" "{'name': 'Ciak Filmpro
##  $ : chr [1:2] "[{'name': 'Miramax Films'" "'id': 14}]"
##  $ : chr [1:2] "[{'name': 'Columbia Pictures Corporation'" "'id': 441}]"
##  $ : chr [1:4] "[{'name': 'I.R.I.B. Channel 2'" "'id': 38953}" "{'name': 'Ferdos Films'" "'id': 3895
##  $ : chr [1:2] "[{'name': 'Miramax Films'" "'id': 14}]"
##  $ : chr [1:6] "[{'name': 'Bergen Film'" "'id': 575}" "{'name': 'Bard Entertainments'" "'id': 576}"
##  $ : chr [1:4] "[{'name': 'BET Pictures'" "'id': 38196}" "{'name': 'United Image Entertainme'" "'id'
##  $ : chr "[]"
##  $ : chr [1:4] "[{'name': 'Playhouse International Pictures'" "'id': 2492}" "{'name': 'The Samuel Go
##  $ : chr [1:6] "[{'name': 'Hollywood Pictures'" "'id': 915}" "{'name': 'Largo Entertainment'" "'id':
##  $ : chr [1:4] "[{'name': 'Twentieth Century Fox Film Corporation'" "'id': 306}" "{'name': 'Joe Wizar
##  $ : chr [1:2] "[{'name': 'Paramount Pictures'" "'id': 4}]"
##  $ : chr [1:2] "[{'name': 'Paramount Pictures'" "'id': 4}]"
##  $ : chr [1:4] "[{'name': 'Miramax Films'" "'id': 14}" "{'name': 'Addis Wechsler Pictures'" "'id': 4
##  $ : chr [1:4] "[{'name': 'TriStar Pictures'" "'id': 559}" "{'name': 'NFH Productions'" "'id': 18737
##  $ : chr [1:2] "[{'name': 'Paramount Pictures'" "'id': 4}]"
##  $ : chr [1:4] "[{'name': 'Miramax Films'" "'id': 14}" "{'name': 'Woods Entertainment'" "'id': 979}]
##  $ : chr [1:6] "[{'name': 'Twentieth Century Fox Film Corporation'" "'id': 306}" "{'name': 'WCG Ente
##  $ : chr [1:4] "[{'name': 'Castle Rock Entertainment'" "'id': 97}" "{'name': 'Midwinter Films'" "'id
##  $ : chr [1:16] "[{'name': 'Egg Pictures'" "'id': 802}" "{'name': 'PolyGram Filmed Entertainment'" "
##  $ : chr [1:2] "[{'name': 'Channel Four Films'" "'id': 181}]"
##  $ : chr [1:12] "[{'name': 'British Broadcasting Corporation (BBC)'" "'id': 3324}" "{'name': 'Westdeu
##  $ : chr [1:4] "[{'name': 'Columbia Pictures'" "'id': 5}" "{'name': 'Castle Rock Entertainment'" "'id
##   [list output truncated]
```

```
str(movies$production_countries)
```

```
## List of 45113
##  $ : chr [1:2] "[{'iso_3166_1': 'US'" "'name': 'United States of America'}]"
##  $ : chr [1:2] "[{'iso_3166_1': 'US'" "'name': 'United States of America'}]"
##  $ : chr [1:2] "[{'iso_3166_1': 'US'" "'name': 'United States of America'}]"
##  $ : chr [1:2] "[{'iso_3166_1': 'US'" "'name': 'United States of America'}]"
##  $ : chr [1:2] "[{'iso_3166_1': 'US'" "'name': 'United States of America'}]"
##  $ : chr [1:2] "[{'iso_3166_1': 'US'" "'name': 'United States of America'}]"
##  $ : chr [1:4] "[{'iso_3166_1': 'DE'" "'name': 'Germany'}" "{'iso_3166_1': 'US'" "'name': 'United Sta
##  $ : chr [1:2] "[{'iso_3166_1': 'US'" "'name': 'United States of America'}]"
```

```
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}" "{’iso_3166_1’: ’US’" "’name’: ’Uni
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’US’" "’name’: ’United Sta
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:8] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’DE’" "’name’: ’Germany’}"
##  $ : chr [1:4] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’US’" "’name’: ’United Sta
##  $ : chr [1:4] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}" "{’iso_3166_1’: ’US’" "’name’: ’Uni
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’US’" "’name’: ’United Sta
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’IT’" "’name’: ’Italy’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}" "{’iso_3166_1’: ’FR’" "’name’: ’Fra
##  $ : chr [1:6] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’DE’" "’name’: ’Germany’}"
##  $ : chr [1:4] "[{’iso_3166_1’: ’CN’" "’name’: ’China’}" "{’iso_3166_1’: ’FR’" "’name’: ’France’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’US’" "’name’: ’United Sta
##  $ : chr [1:4] "[{’iso_3166_1’: ’AU’" "’name’: ’Australia’}" "{’iso_3166_1’: ’US’" "’name’: ’United S
##  $ : chr [1:4] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’GB’" "’name’: ’United King
##  $ : chr [1:4] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}" "{’iso_3166_1’: ’GB’" "’
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’ZA’" "’name’: ’South Africa’}]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}" "{’iso_3166_1’: ’US’" "’name’: ’Uni
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’CA’" "’name’: ’Canada’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr "[]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:6] "[{’iso_3166_1’: ’IT’" "’name’: ’Italy’}" "{’iso_3166_1’: ’FR’" "’name’: ’France’}" .
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}" "{’iso_3166_1’: ’US’" "’name’: ’United Sta
##  $ : chr "[]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:6] "[{’iso_3166_1’: ’BE’" "’name’: ’Belgium’}" "{’iso_3166_1’: ’FR’" "’name’: ’France’}"
##  $ : chr [1:2] "[{’iso_3166_1’: ’CA’" "’name’: ’Canada’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
```

```
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:6] "[{’iso_3166_1’: ’CA’" "’name’: ’Canada’}" "{’iso_3166_1’: ’JP’" "’name’: ’Japan’}" .
##  $ : chr [1:4] "[{’iso_3166_1’: ’DE’" "’name’: ’Germany’}" "{’iso_3166_1’: ’US’" "’name’: ’United Sta
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’IR’" "’name’: ’Iran’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:6] "[{’iso_3166_1’: ’NL’" "’name’: ’Netherlands’}" "{’iso_3166_1’: ’GB’" "’name’: ’Unite
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr "[]"
##  $ : chr [1:4] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}" "{’iso_3166_1’: ’US’" "’name’: ’Un
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’FR’" "’name’: ’France’}]"
##  $ : chr [1:2] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}]"
##  $ : chr [1:8] "[{’iso_3166_1’: ’GB’" "’name’: ’United Kingdom’}" "{’iso_3166_1’: ’CA’" "’name’: ’Ca
##  $ : chr [1:2] "[{’iso_3166_1’: ’US’" "’name’: ’United States of America’}]"
##   [list output truncated]
```

Now I want to clean up the columns so only one parameter from the lists of each of these columns Is shown and referenced In order to make the analysis of data is clean and easy to read.

```r
# Function to parse and extract country names from JSON within a list
extract_countries <- function(list_element) {
  if (length(list_element) == 0 || all(is.na(list_element))) return(NA_character_)

  json_string <- paste0("[", paste(list_element, collapse = ", "), "]")
  json_string <- gsub("'", '"', json_string)

  parsed_data <- tryCatch({
    fromJSON(json_string)
  }, error = function(e) {
    return(NA_character_)
  })
```

```
  if (is.null(parsed_data)) return(NA_character_)

  country_names <- sapply(parsed_data, function(x) {
    if (is.list(x) && !is.null(x$name)) {
      return(x$name)
    } else {
      return(NA_character_)
    }
  })

  country_names <- country_names[!is.na(country_names)]
  return(paste(country_names, collapse = "; "))
}
```

```
# Apply the function to the production_countries column
movies$production_countries <- vapply(movies$production_countries, extract_countries, character(1))

# Verify the cleaned production_countries column
print(head(movies$production_countries))
```

```
## [1] "United States of America" "United States of America"
## [3] "United States of America" "United States of America"
## [5] "United States of America" "United States of America"
```

```
# Function to parse and extract Production Company names from JSON within a list
extract_companies <- function(list_element) {
  if (length(list_element) == 0 || all(is.na(list_element))) return(NA_character_)

  json_string <- paste0("[", paste(list_element, collapse = ", "), "]")
  json_string <- gsub("'", '"', json_string)

  parsed_data <- tryCatch({
    fromJSON(json_string)
  }, error = function(e) {
    return(NA_character_)
  })

  if (is.null(parsed_data)) return(NA_character_)

  company_names <- sapply(parsed_data, function(x) {
    if (is.list(x) && !is.null(x$name)) {
      return(x$name)
    } else {
      return(NA_character_)
    }
  })

  company_names <- company_names[!is.na(company_names)]
  return(paste(company_names, collapse = "; "))
}
```

```r
# Apply the function to the production_companies column
movies$production_companies <- vapply(movies$production_companies, extract_companies, character(1))

# Verify the cleaned production_countries column
print(head(movies$production_companies))
```

```
## [1] "Pixar Animation Studios"
## [2] "TriStar Pictures; Teitler Film; Interscope Communications"
## [3] "Warner Bros.; Lancaster Gate"
## [4] "Twentieth Century Fox Film Corporation"
## [5] "Sandollar Productions; Touchstone Pictures"
## [6] "Regency Enterprises; Forward Pass; Warner Bros."
```

```r
# Function to parse and extract Genre names from JSON within a list
extract_genres <- function(list_element) {
  if (length(list_element) == 0 || all(is.na(list_element))) return(NA_character_)

  json_string <- paste0("[", paste(list_element, collapse = ", "), "]")
  json_string <- gsub("'", '"', json_string)

  parsed_data <- tryCatch({
    fromJSON(json_string)
  }, error = function(e) {
    return(NA_character_)
  })

  if (is.null(parsed_data)) return(NA_character_)

  genre_names <- sapply(parsed_data, function(x) {
    if (is.list(x) && !is.null(x$name)) {
      return(x$name)
    } else {
      return(NA_character_)
    }
  })

  genre_names <- genre_names[!is.na(genre_names)]
  return(paste(genre_names, collapse = "; "))
}
```

```r
# Apply the function to the production_countries column
movies$genres <- vapply(movies$genres, extract_genres, character(1))

# Verify the cleaned production_countries column
print(head(movies$genres))
```

```
## [1] "Animation; Comedy; Family"      "Adventure; Fantasy; Family"
## [3] "Romance; Comedy"                "Comedy; Drama; Romance"
## [5] "Comedy"                         "Action; Crime; Drama; Thriller"
```

```r
# First lets Remove Scientific Notation from my data
options(scipen=999)
```

Now I can move onto the columns that have numeric values I want to clean up the budget and revenue columns so that the value of 0 is replaced by N/A as it is unfeasible that the amount of these values is 0

```r
# Replace 0 with NA in 'budget' and 'revenue'
movies$budget[movies$budget == 0] <- NA
movies$revenue[movies$revenue == 0] <- NA
```

Now time to categorise the budget values into groups

```r
# Number of rows with budget < 100
rows_less_than_100 <- nrow(movies[!is.na(movies$budget) & movies$budget < 100, ])

# Number of rows with budget > 100 and < 1000
rows_between_100_and_1000 <- nrow(movies[!is.na(movies$budget) & movies$budget > 100 & movies$budget < 

# Number of rows with budget > 1000 and < 10000
rows_between_1000_and_10000 <- nrow(movies[!is.na(movies$budget) & movies$budget > 1000 & movies$budget

# Print the results
cat("Number of rows with budget < 100: ", rows_less_than_100, "\n")
```

```
## Number of rows with budget < 100:  137
```

```r
cat("Number of rows with budget > 100 and < 1000: ", rows_between_100_and_1000, "\n")
```

```
## Number of rows with budget > 100 and < 1000:  100
```

```r
cat("Number of rows with budget > 1000 and < 10000: ", rows_between_1000_and_10000, "\n")
```

```
## Number of rows with budget > 1000 and < 10000:  52
```

There are some rows that have a budget and revenue value that are not correctly scaled. i.e: - id: 17402 - Title: Miami Rhapsody - Production Company: Hollywood Pictures
- Date: 1995-01-27
- Budget: 6 - Revenue: 5 (by looking IMDB, actual revenue can be seen as around 5 million)

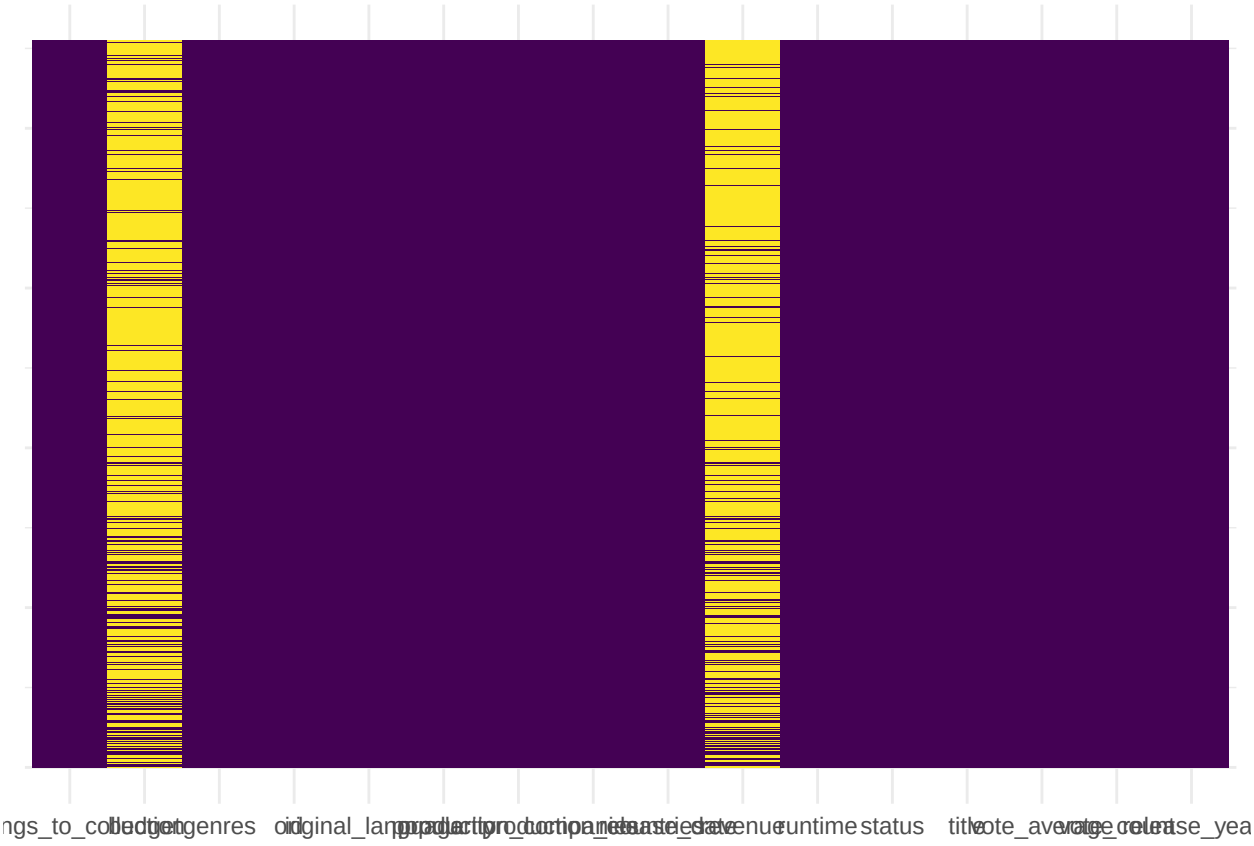
Because of this issue and after checking some of the notebooks shared from other Analysts, I have decided to adopt the scaling function they have For example, if the value is 1, then it scales to 1 million.

```r
# Creating the Function
scale_money <- function(num) {
  if (is.na(num)) {
    return(NA)
  } else if (num < 100) {
    return(num * 1000000)
```

```
  } else if (num >= 100 & num < 1000) {
    return(num * 10000)
  } else if (num >= 1000 & num < 10000) {
    return(num * 100)
  } else {
    return(num)
  }
}

# Applying the function to 'budget' and 'revenue' columns
movies$budget <- sapply(movies$budget, scale_money)
movies$revenue <- sapply(movies$revenue, scale_money)
```

After these steps, the columns can be osberved to see how many null or NaN entries there are. So, a heatmap of missing/Null data is below:

```
# Create a matrix of missing values
missing_vals <- is.na(movies) + 0  # Convert logical to numeric

# Melt the matrix to long format for ggplot
melted_missing_vals <- melt(missing_vals)

# Plot the heatmap
ggplot(data = melted_missing_vals, aes(x = Var2, y = Var1, fill = value)) +
  geom_tile() +
  scale_fill_viridis_c(option = "viridis") +
  theme_minimal() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.title = element_blank(),
        legend.position = "none")
```



Looking at the two columns we've been altering, *budget* and *revenue* the majority of values in these columns now seem to have usable data (Yellow) with the (Purple) NA/Nulls not being as prevalent in these two columns.

I should note not to take the Purple in the other columns as these having NA/Null data as the heatmap is only showing numerical data therefore other columns such as title which is charecther based will show as null despite having data.

to provide further clairty for this I will run counts for NA/Null values for specified columns

```
# Count NaN values in specified columns
nan_genres_count <- sum(is.na(movies$genres))
nan_revenue_count <- sum(is.na(movies$revenue))
nan_budget_count <- sum(is.na(movies$budget))
nan_production_company_count <- sum(is.na(movies$production_companies))
nan_production_country_count <- sum(is.na(movies$production_countries))

# Print the results
cat("NaN Genres Count: ", nan_genres_count, "\n")
```

```
## NaN Genres Count:  0
```

```
cat("NaN Revenue Count: ", nan_revenue_count, "\n")
```

```
## NaN Revenue Count:  37715
```

```r
cat("NaN Budget Count: ", nan_budget_count, "\n")
```

```
## NaN Budget Count:  36238
```

```r
cat("NaN Production Company Count: ", nan_production_company_count, "\n")
```

```
## NaN Production Company Count:  0
```

```r
cat("NaN Production Country Count: ", nan_production_country_count, "\n")
```

```
## NaN Production Country Count:  0
```

For *revenue*, *budget* and *production company* filling the values with the most appearing entry or mean is not so logical, since the number of null or NaN entries are huge (More than 20% of whole dataset). But for *genres* and *country* it can be done. The function below analyzes the most occuring values for columns in list formats.

```r
list_counter <- function(col, limiter = 9999, log = TRUE) {
  result <- list()

  for (cell in col) {
    if (is.na(cell[1])) {
      next
    }
    for (element in cell) {
      if (!is.null(result[[element]])) {
        result[[element]] <- result[[element]] + 1
      } else {
        result[[element]] <- 1
      }
    }
  }

  if (log) {
    cat("Size of words:", length(result), "\n")
  }

  result <- result[order(unlist(result), decreasing = TRUE)]

  if (log) {
    cat("Sorted result is:\n")
  }

  counter <- 1
  sum_selected <- 0
  total_selected <- 0
  returned <- list()
  for (i in names(result)) {
    if (counter > limiter) {
      total_selected <- total_selected + result[[i]]
    } else {
```

```
      counter <- counter + 1
      sum_selected <- sum_selected + result[[i]]
      total_selected <- total_selected + result[[i]]
      if (log) {
        cat(result[[i]], " - ", i, "\n")
      }
      returned <- c(returned, list(c(i, result[[i]])))
    }
  }

  if (log) {
    cat("Covered:", sum_selected, "out of", total_selected, "\n")
  }

  return(returned)
}
```

```
# Seperating the columns that contain combined genres
genres_separated <- movies %>%
  separate_rows(genres, sep = "; ")

# Verify the result
print(head(genres_separated))
```

Now to count the occurrences within each genre within the data.

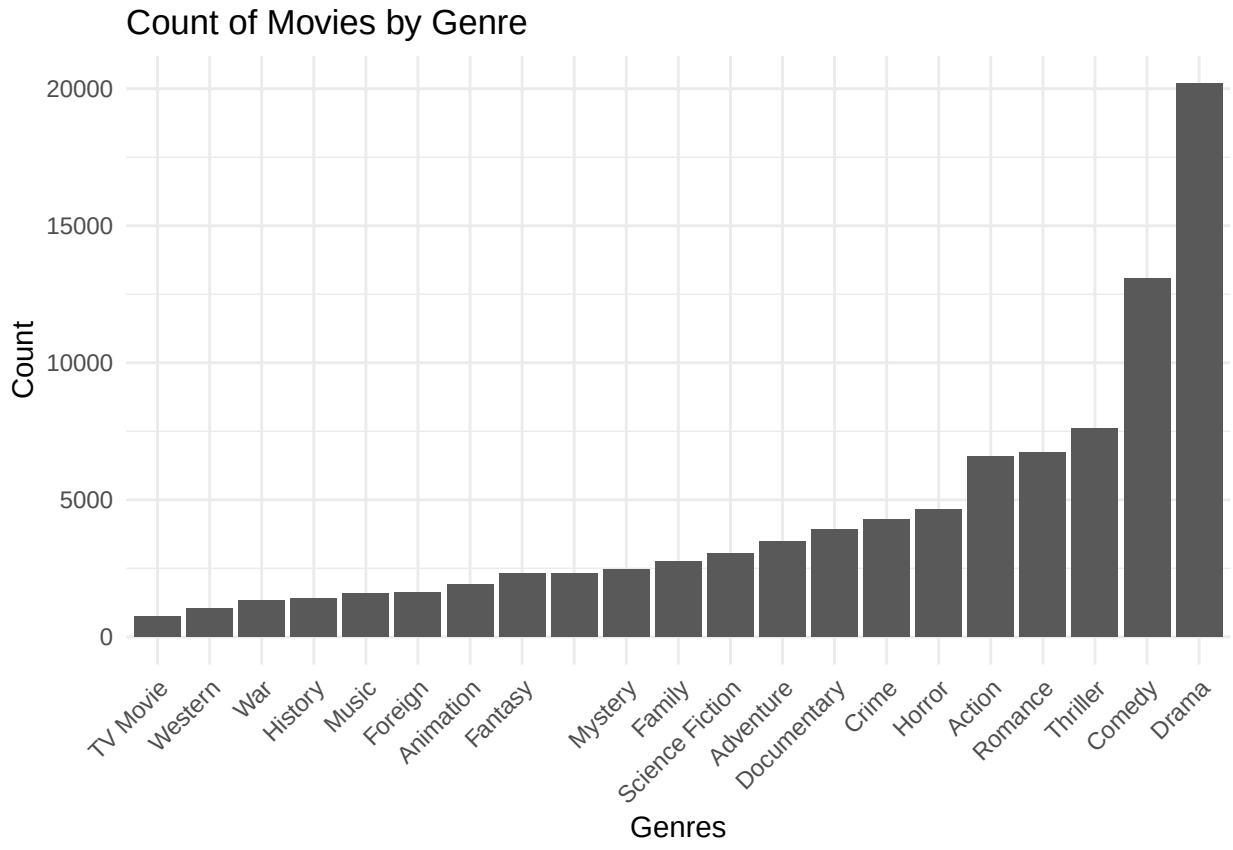
```
## # A tibble: 6 x 16
##   belongs_to_collection   budget genres       id original_language popularity
##                   <dbl>    <dbl> <chr>     <int> <chr>                   <dbl>
## 1                     1 30000000 Animation   862 en                       21.9
## 2                     1 30000000 Comedy      862 en                       21.9
## 3                     1 30000000 Family      862 en                       21.9
## 4                     1 65000000 Adventure  8844 en                       17.0
## 5                     1 65000000 Fantasy    8844 en                       17.0
## 6                     1 65000000 Family     8844 en                       17.0
## # i 10 more variables: production_companies <chr>, production_countries <chr>,
## #   release_date <date>, revenue <dbl>, runtime <dbl>, status <chr>,
## #   title <chr>, vote_average <dbl>, vote_count <int>, release_year <chr>
```

```
# Count the occurrences of each genre
genre_occur <- genres_separated %>%
  count(genres, sort = TRUE)

# Print the genre counts
print(genre_occur)
```

```
## # A tibble: 21 x 2
##    genres              n
##    <chr>           <int>
##  1 Drama           20190
```

```
##  2 Comedy          13090
##  3 Thriller         7606
##  4 Romance          6719
##  5 Action           6580
##  6 Horror           4667
##  7 Crime            4296
##  8 Documentary      3910
##  9 Adventure        3486
## 10 Science Fiction  3030
## # i 11 more rows
```

```r
# Plot the counts of each genre
ggplot(genre_occur, aes(x = reorder(genres, n), y = n)) +
  geom_bar(stat = "identity") +
  labs(title = "Count of Movies by Genre",
       x = "Genres",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Separating the columns that contain combined Countries
countries_separated <- movies %>%
  separate_rows(production_countries, sep = "; ")
```

```r
# Verify the result
print(head(countries_separated))
```

Now to see the top 20 Production Countries occurrences in the data.

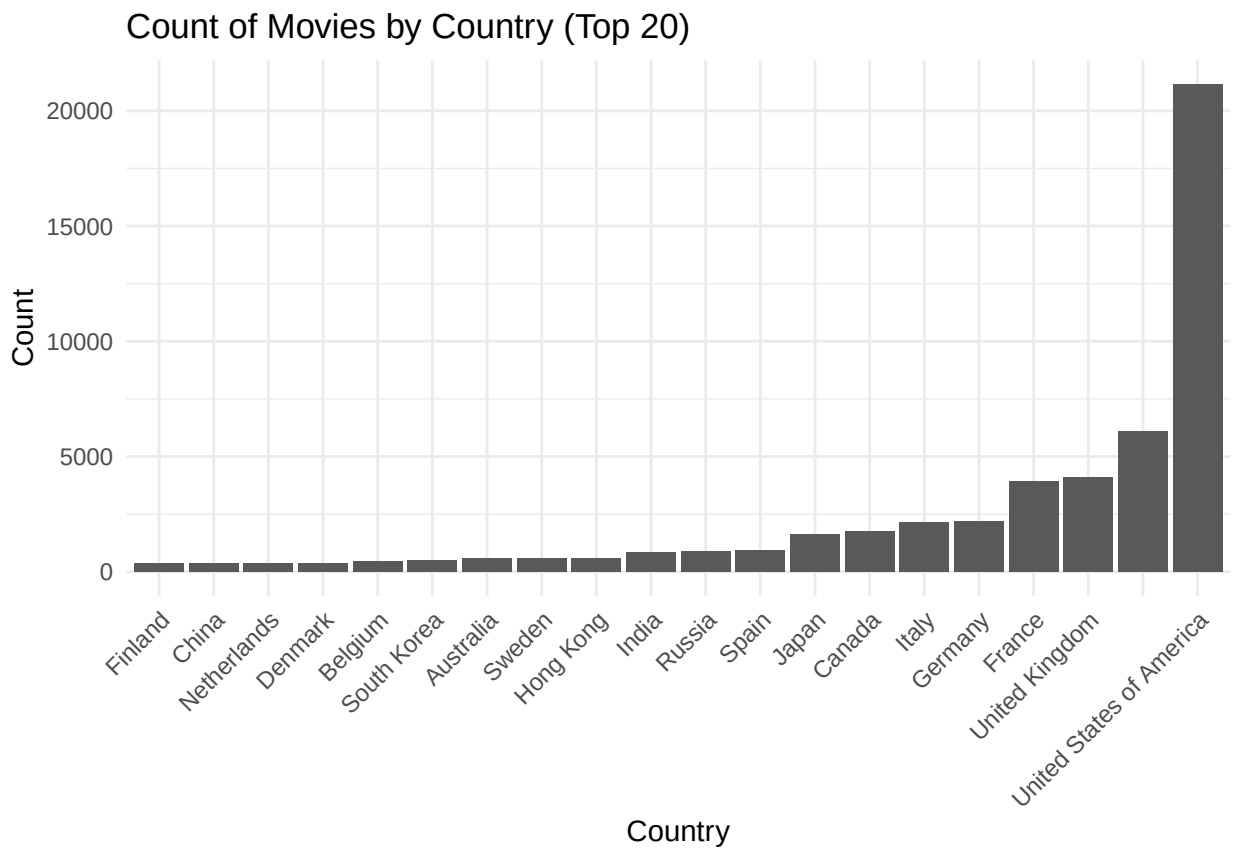
```
## # A tibble: 6 x 16
##   belongs_to_collection   budget genres          id original_language popularity
##                   <dbl>    <dbl> <chr>        <int> <chr>                   <dbl>
## 1                     1 30000000 Animation; ~   862 en                      21.9
## 2                     1 65000000 Adventure; ~  8844 en                      17.0
## 3                     1       NA Romance; Co~ 15602 en                      11.7
## 4                     1 16000000 Comedy; Dra~ 31357 en                       3.86
## 5                     1       NA Comedy       11862 en                       8.39
## 6                     1 60000000 Action; Cri~   949 en                      17.9
## # i 10 more variables: production_companies <chr>, production_countries <chr>,
## #   release_date <date>, revenue <dbl>, runtime <dbl>, status <chr>,
## #   title <chr>, vote_average <dbl>, vote_count <int>, release_year <chr>
```

```r
# Count the occurrences of each Country
countries_occur <- countries_separated %>%
  count(production_countries, sort = TRUE)

# Print the Country counts
print(countries_occur)
```

```
## # A tibble: 159 x 2
##    production_countries           n
##    <chr>                      <int>
##  1 "United States of America" 21137
##  2 ""                          6111
##  3 "United Kingdom"            4086
##  4 "France"                    3916
##  5 "Germany"                   2212
##  6 "Italy"                     2134
##  7 "Canada"                    1762
##  8 "Japan"                     1647
##  9 "Spain"                      946
## 10 "Russia"                     907
## # i 149 more rows
```

```r
# filtering to the top 20
top_20_countries <- countries_occur %>%
  top_n(20, n) %>%
  arrange(desc(n))



# Plot the counts the Top 20 Countries
ggplot(top_20_countries, aes(x = reorder(production_countries, n), y = n)) +
  geom_bar(stat = "identity") +
  labs(title = "Count of Movies by Country (Top 20)",
       x = "Country",
```

```
      y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



In *genres Drama* is the most occurring with 20189 results and in *production_countries US* is the most frequent. These can be placed into NA cells of these columns:

```
fill_na_with_list <- function(cell, data) {
  if (is.na(cell)) {
    return(data)
  } else {
    return(cell)
  }
}
```

```
fill_na_with_list <- function(cell, data) {
  if (anyNA(cell)) {
    return(data)
  } else {
    return(cell)
  }
}

# 'genres_occur' and 'countries_occur' are already defined and processed through previous steps
```

```r
# Fill NA values in genres column
movies$genres <- lapply(movies$genres, function(row) {
  fill_na_with_list(row, list(genres_occur[[1]][[1]]))
})

# Fill NA values in production_countries column
movies$production_countries <- lapply(movies$production_countries, function(row) {
  fill_na_with_list(row, list(countries_occur[[1]][[1]]))
})

# Check the filled columns
head(movies$genres)
```

```
## [[1]]
## [1] "Animation; Comedy; Family"
##
## [[2]]
## [1] "Adventure; Fantasy; Family"
##
## [[3]]
## [1] "Romance; Comedy"
##
## [[4]]
## [1] "Comedy; Drama; Romance"
##
## [[5]]
## [1] "Comedy"
##
## [[6]]
## [1] "Action; Crime; Drama; Thriller"
```

```r
head(movies$production_countries)
```

```
## [[1]]
## [1] "United States of America"
##
## [[2]]
## [1] "United States of America"
##
## [[3]]
## [1] "United States of America"
##
## [[4]]
## [1] "United States of America"
##
## [[5]]
## [1] "United States of America"
##
## [[6]]
## [1] "United States of America"
```

Time to revist my dataset after all this alteration

```
# To get the dimensions of the dataframe
dim(movies)
```

```
## [1] 45113    16
```

```
# To get the structure of the dataframe
str(movies)
```

```
## 'data.frame':    45113 obs. of  16 variables:
##  $ belongs_to_collection: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ budget               : num  30000000 65000000 NA 16000000 NA 60000000 58000000 NA 35000000 580000
##  $ genres               :List of 45113
##   ..$ : chr "Animation; Comedy; Family"
##   ..$ : chr "Adventure; Fantasy; Family"
##   ..$ : chr "Romance; Comedy"
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Comedy"
##   ..$ : chr "Action; Crime; Drama; Thriller"
##   ..$ : chr "Comedy; Romance"
##   ..$ : chr "Action; Adventure; Drama; Family"
##   ..$ : chr "Action; Adventure; Thriller"
##   ..$ : chr "Adventure; Action; Thriller"
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Comedy; Horror"
##   ..$ : chr "Family; Animation; Adventure"
##   ..$ : chr "History; Drama"
##   ..$ : chr "Action; Adventure"
##   ..$ : chr "Drama; Crime"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Crime; Comedy"
##   ..$ : chr "Crime; Comedy; Adventure"
##   ..$ : chr "Action; Comedy; Crime"
##   ..$ : chr "Comedy; Thriller; Crime"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Action; Adventure; Crime; Thriller"
##   ..$ : chr "Drama; Fantasy; Science Fiction; Thriller"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Comedy; Drama; Family"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Fantasy; Science Fiction; Adventure"
##   ..$ : chr "Drama; Crime"
##   ..$ : chr "Drama; Crime"
##   ..$ : chr "Science Fiction; Thriller; Mystery"
##   ..$ : chr "Romance; Adventure"
##   ..$ : chr "Fantasy; Drama; Comedy; Family"
##   ..$ : chr "History; Drama; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Adventure; History; Drama; Family"
##   ..$ : chr "Comedy; Family; Romance"
```

```
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Drama; War"
##   ..$ : chr "Action; Crime; Drama; History"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Action; Fantasy"
##   ..$ : chr "Fantasy; Drama; Comedy; Thriller"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Crime; Mystery; Thriller"
##   ..$ : chr "Adventure; Animation; Drama; Family"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Drama; Crime; Thriller"
##   ..$ : chr "Action; Thriller; Drama"
##   ..$ : chr "Comedy; Romance"
##   ..$ : chr "Drama; Foreign"
##   ..$ : chr "Action; Adventure; Comedy; Family"
##   ..$ : chr "Drama"
##   ..$ : chr ""
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Thriller; Drama; Mystery"
##   ..$ : chr "Adventure; Family; Fantasy"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Music; Drama; Family"
##   ..$ : chr "Comedy"
##   ..$ : chr "Comedy; Romance"
##   ..$ : chr "Comedy"
##   ..$ : chr "Action; Science Fiction"
##   ..$ : chr "Drama; Family"
##   ..$ : chr "Comedy; Romance"
##   ..$ : chr "Comedy"
##   ..$ : chr "Horror; Action; Thriller; Crime"
##   ..$ : chr "Action; Thriller; Romance"
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Drama; History"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Comedy; Family"
##   ..$ : chr "Horror; Science Fiction"
##   ..$ : chr "Documentary"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Family; Drama"
##   ..$ : chr "Drama; Crime"
##   ..$ : chr "Drama; Comedy"
##   ..$ : chr "Romance; Drama"
##   ..$ : chr ""
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Action; Drama"
##   ..$ : chr "Adventure; Comedy; Family"
##   ..$ : chr "Comedy"
##   ..$ : chr "Crime; Drama; Thriller"
##   ..$ : chr "Drama"
##   ..$ : chr "Drama; Horror; Thriller; Romance"
##   ..$ : chr "Comedy; Horror; Romance"
```

```
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Action; Adventure; Drama; Thriller"
##   ..$ : chr "Comedy"
##   ..$ : chr "Drama"
##   ..$ : chr "Action; Adventure; Drama; Science Fiction; Thriller"
##   ..$ : chr "Documentary"
##   ..$ : chr "Drama; Thriller"
##   .. [list output truncated]
##  $ id                   : int  862 8844 15602 31357 11862 949 11860 45325 9091 710 ...
##  $ original_language    : chr  "en" "en" "en" "en" ...
##  $ popularity           : num  21.95 17.02 11.71 3.86 8.39 ...
##  $ production_companies : chr  "Pixar Animation Studios" "TriStar Pictures; Teitler Film; Interscope
##  $ production_countries :List of 45113
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Germany; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France; Germany; Italy; United States of America"
##   ..$ : chr "France; United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Italy"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; France"
##   ..$ : chr "France; Germany; Spain"
##   ..$ : chr "China; France"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France; United States of America"
##   ..$ : chr "Australia; United States of America"
##   ..$ : chr "France; United Kingdom"
##   ..$ : chr "United States of America; United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "South Africa"
##   ..$ : chr "United Kingdom; United States of America"
```

```
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Canada"
##   ..$ : chr "United States of America"
##   ..$ : chr ""
##   ..$ : chr "United States of America"
##   ..$ : chr "Italy; France; Switzerland"
##   ..$ : chr "United States of America"
##   ..$ : chr "France; United States of America"
##   ..$ : chr ""
##   ..$ : chr "United States of America"
##   ..$ : chr "Belgium; France; Italy"
##   ..$ : chr "Canada"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Canada; Japan; United States of America"
##   ..$ : chr "Germany; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Iran"
##   ..$ : chr "United States of America"
##   ..$ : chr "Netherlands; United Kingdom; Belgium"
##   ..$ : chr "United States of America"
##   ..$ : chr ""
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom"
```

```
##   ..$ : chr "France"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "United Kingdom; Canada; United States of America; Germany"
##   ..$ : chr "United States of America"
##   .. [list output truncated]
##  $ release_date         : Date, format: "1995-10-30" "1995-12-15" ...
##  $ revenue              : num  373554033 262797249 NA 81452156 76578911 ...
##  $ runtime              : num  81 104 101 127 106 170 127 97 106 130 ...
##  $ status               : chr  "Released" "Released" "Released" "Released" ...
##  $ title                : chr  "Toy Story" "Jumanji" "Grumpier Old Men" "Waiting to Exhale" ...
##  $ vote_average         : num  7.7 6.9 6.5 6.1 5.7 7.7 6.2 5.4 5.5 6.6 ...
##  $ vote_count           : int  5415 2413 92 34 173 1886 141 45 174 1194 ...
##  $ release_year         : chr  "1995" "1995" "1995" "1995" ...
```

I want to add a profit column to my dataset as a new value to aid analysis

```
# Create a new column 'profit'
movies$profit <- movies$revenue - movies$budget
```

```
# Get descriptive statistics for specific columns
summary(movies[, c("popularity", "revenue", "budget", "runtime", "vote_average", "profit", "release_yea
```

Now it's time to get an overall summary of our data

```
##    popularity          revenue              budget             runtime
##  Min.   :  0.0000   Min.   :     10000   Min.   :    10000   Min.   :   1.0
##  1st Qu.:  0.3975   1st Qu.:   2956494   1st Qu.:  2105625   1st Qu.:  87.0
##  Median :  1.1389   Median :  17314578   Median :  8806000   Median :  96.0
##  Mean   :  2.9412   Mean   :  69221938   Mean   : 21991610   Mean   :  97.5
##  3rd Qu.:  3.7271   3rd Qu.:  68079684   3rd Qu.: 25515000   3rd Qu.: 107.0
##  Max.   :547.4883   Max.   :2787965087   Max.   :380000000   Max.   :1256.0
##                     NA's   :37715        NA's   :36238
##   vote_average        profit           release_year
##  Min.   : 0.000   Min.   :-165710090   Length:45113
##  1st Qu.: 5.000   1st Qu.:  -1482475   Class :character
##  Median : 6.000   Median :  11261424   Mode  :character
##  Mean   : 5.635   Mean   :  59289246
##  3rd Qu.: 6.800   3rd Qu.:  62230837
##  Max.   :10.000   Max.   :2550965087
##                   NA's   :39733
```

Since the difference between min and max values for *budget*, *revenue* and *profit* is quite large, I will normalise these. In order to preserve the signs of the parameters, the formula will be applied as: - value / (max - min)

```
# Calculate the min and max values of the 'budget' column
min_val <- min(movies$budget, na.rm = TRUE)
max_val <- max(movies$budget, na.rm = TRUE)

# Normalize 'budget', 'revenue', and 'profit' columns
movies[, c("budget", "revenue", "profit")] <- lapply(movies[, c("budget", "revenue", "profit")], functi
```

```
  x / (max_val - min_val)
})

# Check the updated dataframe
head(movies[, c("budget", "revenue", "profit")])
```

```
##       budget   revenue    profit
## 1 0.07894945 0.9830628 0.9041134
## 2 0.17105713 0.6915899 0.5205328
## 3         NA        NA        NA
## 4 0.04210637 0.2143534 0.1722470
## 5         NA 0.2015288        NA
## 6 0.15789889 0.4932678 0.3353689
```

I now need to arrange *vote_counts* and *vote_averages* with a weighted manner. since there are lots of 0s in the dataset in both columns. The weighted process is implemented below and explained as: - Weighted Rating for a row (WR) = [(v + 1) / (v + m) * R] + [m / (m + v) * C] - v: number of votes for the movie - m: minimum votes required to be listed in the chart (quantile 0.75) - R: average rating of the movie - C: mean vote across the whole report

```
# Filter vote counts and vote averages
vote_counts <- as.integer(movies$vote_count[!is.na(movies$vote_count)])
vote_averages <- as.integer(movies$vote_average[!is.na(movies$vote_average)])

# Calculate C and m
C <- mean(vote_averages)
m <- quantile(vote_counts, 0.75)

# Define the weighted rating function
weighted_rating <- function(data) {
  v <- data$vote_count + 1  # added +1
  R <- data$vote_average
  return ((v / (v + m) * R) + (m / (m + v) * C))
}

# Apply the function to the dataframe
movies$weighted_rating <- apply(movies, 1, weighted_rating)

# Check the new column
head(movies$weighted_rating)
```

```
## [1] 7.684330 6.876554 6.160790 5.679730 5.626225 7.655557
```

- Keywords Dataset

First to import the keywords csv file and look at the columns of the data to see the format:

```
keywords <- read.csv("keywords.csv")

head(keywords)
```

```
##      id
## 1   862
## 2  8844
## 3 15602
## 4 31357
## 5 11862
## 6   949
##
## 1
## 2
## 3
## 4
## 5
## 6 [{'id': 642, 'name': 'robbery'}, {'id': 703, 'name': 'detective'}, {'id': 974, 'name': 'bank'}, {'
```

As with the *movies* dataset the *keywords* format includes a column of data in chr string which I will want to convert to simple list. I can do this using the previous function written in the movies dataset and any problematic values can be adjusted:

```
# Apply the convert_to_list function to the 'keywords' column
keywords$keywords <- convert_to_list(keywords$keywords)

# Check the structure of the new 'keywords' column
str(keywords$keywords)
```

```
## List of 46419
##  $ : chr [1:18] "[{'id': 931" "'name': 'jealousy'}" "{'id': 4290" "'name': 'toy'}" ...
##  $ : chr [1:12] "[{'id': 10090" "'name': 'board game'}" "{'id': 10941" "'name': 'disappearance'}" ..
##  $ : chr [1:8] "[{'id': 1495" "'name': 'fishing'}" "{'id': 12392" "'name': 'best friend'}" ...
##  $ : chr [1:10] "[{'id': 818" "'name': 'based on novel'}" "{'id': 10131" "'name': 'interracial relati
##  $ : chr [1:18] "[{'id': 1009" "'name': 'baby'}" "{'id': 1599" "'name': 'midlife crisis'}" ...
##  $ : chr [1:50] "[{'id': 642" "'name': 'robbery'}" "{'id': 703" "'name': 'detective'}" ...
##  $ : chr [1:12] "[{'id': 90" "'name': 'paris'}" "{'id': 380" "'name': 'brother brother relationship'
##  $ : chr "[]"
##  $ : chr [1:8] "[{'id': 949" "'name': 'terrorist'}" "{'id': 1562" "'name': 'hostage'}" ...
##  $ : chr [1:30] "[{'id': 701" "'name': 'cuba'}" "{'id': 769" "'name': 'falsely accused'}" ...
##  $ : chr [1:10] "[{'id': 833" "'name': 'white house'}" "{'id': 840" "'name': 'usa president'}" ...
##  $ : chr [1:4] "[{'id': 3633" "'name': 'dracula'}" "{'id': 11931" "'name': 'spoof'}]"
##  $ : chr [1:16] "[{'id': 1994" "'name': 'wolf'}" "{'id': 6411" "'name': 'dog-sledding race'}" ...
##  $ : chr [1:12] "[{'id': 840" "'name': 'usa president'}" "{'id': 2946" "'name': 'presidential electi
##  $ : chr [1:12] "[{'id': 911" "'name': 'exotic island'}" "{'id': 1454" "'name': 'treasure'}" ...
##  $ : chr [1:10] "[{'id': 383" "'name': 'poker'}" "{'id': 726" "'name': 'drug abuse'}" ...
##  $ : chr [1:22] "[{'id': 420" "'name': 'bowling'}" "{'id': 818" "'name': 'based on novel'}" ...
##  $ : chr [1:20] "[{'id': 612" "'name': 'hotel'}" "{'id': 613" "'name': \"new year's eve\"}" ...
##  $ : chr [1:8] "[{'id': 409" "'name': 'africa'}" "{'id': 1551" "'name': 'indigenous'}" ...
##  $ : chr [1:10] "[{'id': 380" "'name': 'brother brother relationship'}" "{'id': 1552" "'name': 'subwa
##  $ : chr [1:36] "[{'id': 395" "'name': 'gambling'}" "{'id': 416" "'name': 'miami'}" ...
##  $ : chr [1:14] "[{'id': 796" "'name': 'police brutality'}" "{'id': 2800" "'name': 'psychology'}" ..
##  $ : chr [1:38] "[{'id': 271" "'name': 'competition'}" "{'id': 441" "'name': 'assassination'}" ...
##  $ : chr [1:28] "[{'id': 7345" "'name': 'dead animal'}" "{'id': 11325" "'name': 'shot to death'}" ..
##  $ : chr [1:32] "[{'id': 30" "'name': 'individual'}" "{'id': 549" "'name': 'prostitute'}" ...
##  $ : chr [1:8] "[{'id': 497" "'name': 'shakespeare'}" "{'id': 931" "'name': 'jealousy'}" ...
##  $ : chr [1:14] "[{'id': 1228" "'name': '1970s'}" "{'id': 5248" "'name': 'female friendship'}" ...
```

```
##  $ : chr [1:8] "[{'id': 1316" "'name': 'captain'}" "{'id': 33384" "'name': 'napoleonic wars'}" ...
##  $ : chr [1:22] "[{'id': 402" "'name': 'clone'}" "{'id': 1566" "'name': 'dream'}" ...
##  $ : chr [1:14] "[{'id': 964" "'name': 'servant'}" "{'id': 2311" "'name': 'cabaret'}" ...
##  $ : chr [1:12] "[{'id': 897" "'name': 'rap music'}" "{'id': 3436" "'name': 'karate'}" ...
##  $ : chr [1:20] "[{'id': 222" "'name': 'schizophrenia'}" "{'id': 276" "'name': 'philadelphia'}" ...
##  $ : chr "[]"
##  $ : chr [1:20] "[{'id': 3020" "'name': 'sheep'}" "{'id': 4931" "'name': 'pig'}" ...
##  $ : chr [1:10] "[{'id': 392" "'name': 'england'}" "{'id': 437" "'name': 'painter'}" ...
##  $ : chr [1:50] "[{'id': 378" "'name': 'prison'}" "{'id': 570" "'name': 'rape'}" ...
##  $ : chr "[]"
##  $ : chr [1:4] "[{'id': 13014" "'name': 'orphan'}" "{'id': 204353" "'name': 'identical twin'}]"
##  $ : chr [1:14] "[{'id': 1525" "'name': 'puberty'}" "{'id': 6270" "'name': 'high school'}" ...
##  $ : chr [1:2] "[{'id': 3644" "'name': 'south africa'}]"
##  $ : chr [1:10] "[{'id': 392" "'name': 'england'}" "{'id': 497" "'name': 'shakespeare'}" ...
##  $ : chr [1:12] "[{'id': 10594" "'name': 'money'}" "{'id': 11254" "'name': 'loss of virginity'}" ...
##  $ : chr [1:14] "[{'id': 931" "'name': 'jealousy'}" "{'id': 1279" "'name': 'medicine'}" ...
##  $ : chr [1:18] "[{'id': 779" "'name': 'martial arts'}" "{'id': 1299" "'name': 'monster'}" ...
##  $ : chr [1:12] "[{'id': 596" "'name': 'adultery'}" "{'id': 1442" "'name': 'winter'}" ...
##  $ : chr [1:24] "[{'id': 1334" "'name': 'wedding vows'}" "{'id': 1459" "'name': 'marriage proposal'}"
##  $ : chr [1:28] "[{'id': 476" "'name': 'self-fulfilling prophecy'}" "{'id': 703" "'name': 'detective
##  $ : chr [1:54] "[{'id': 1463" "'name': 'culture clash'}" "{'id': 3591" "'name': 'settler'}" ...
##  $ : chr [1:6] "[{'id': 1992" "'name': 'professor'}" "{'id': 158718" "'name': 'lgbt'}" ...
##  $ : chr [1:14] "[{'id': 3703" "'name': 'law'}" "{'id': 5493" "'name': 'relatives'}" ...
##  $ : chr "[]"
##  $ : chr [1:18] "[{'id': 549" "'name': 'prostitute'}" "{'id': 2393" "'name': 'adoption'}" ...
##  $ : chr [1:2] "[{'id': 10183" "'name': 'independent film'}]"
##  $ : chr [1:8] "[{'id': 6075" "'name': 'sport'}" "{'id': 9963" "'name': 'kids and family'}" ...
##  $ : chr [1:10] "[{'id': 567" "'name': 'alcohol'}" "{'id': 2679" "'name': 'artist'}" ...
##  $ : chr "[]"
##  $ : chr [1:12] "[{'id': 2672" "'name': 'baltimore'}" "{'id': 4543" "'name': 'thanksgiving'}" ...
##  $ : chr [1:10] "[{'id': 2041" "'name': 'island'}" "{'id': 3344" "'name': 'letter'}" ...
##  $ : chr "[]"
##  $ : chr [1:10] "[{'id': 1990" "'name': 'cupboard'}" "{'id': 6186" "'name': 'games'}" ...
##  $ : chr [1:8] "[{'id': 570" "'name': 'rape'}" "{'id': 1419" "'name': 'gun'}" ...
##  $ : chr [1:16] "[{'id': 1233" "'name': 'composer'}" "{'id': 1310" "'name': 'mentor'}" ...
##  $ : chr [1:20] "[{'id': 248" "'name': 'date'}" "{'id': 398" "'name': 'slang'}" ...
##  $ : chr [1:8] "[{'id': 2041" "'name': 'island'}" "{'id': 5918" "'name': 'painting'}" ...
##  $ : chr [1:6] "[{'id': 4166" "'name': 'biotope'}" "{'id': 4459" "'name': 'vegetarian'}" ...
##  $ : chr [1:8] "[{'id': 2157" "'name': 'hacker'}" "{'id': 4563" "'name': 'virtual reality'}" ...
##  $ : chr "[]"
##  $ : chr [1:12] "[{'id': 128" "'name': 'love triangle'}" "{'id': 2335" "'name': 'southern france'}"
##  $ : chr [1:8] "[{'id': 897" "'name': 'rap music'}" "{'id': 970" "'name': 'parent child relationship
##  $ : chr [1:52] "[{'id': 246" "'name': 'dancing'}" "{'id': 380" "'name': 'brother brother relationsh
##  $ : chr [1:82] "[{'id': 258" "'name': 'bomb'}" "{'id': 416" "'name': 'miami'}" ...
##  $ : chr [1:2] "[{'id': 3616" "'name': 'college'}]"
##  $ : chr [1:10] "[{'id': 90" "'name': 'paris'}" "{'id': 745" "'name': 'nun'}" ...
##  $ : chr [1:4] "[{'id': 2778" "'name': 'florist'}" "{'id': 10235" "'name': 'family relationships'}]"
##  $ : chr [1:2] "[{'id': 10624" "'name': 'bully'}]"
##  $ : chr [1:6] "[{'id': 312" "'name': 'man vs machine'}" "{'id': 10158" "'name': 'alien planet'}" ..
##  $ : chr "[]"
##  $ : chr [1:12] "[{'id': 6381" "'name': 'loss of daughter'}" "{'id': 6440" "'name': 'hit-and-run dri
##  $ : chr [1:12] "[{'id': 1417" "'name': 'jurors'}" "{'id': 1936" "'name': 'blackmail'}" ...
##  $ : chr [1:8] "[{'id': 1357" "'name': 'fish'}" "{'id': 10594" "'name': 'money'}" ...
##  $ : chr [1:42] "[{'id': 494" "'name': 'father son relationship'}" "{'id': 801" "'name': 'bounty hun
```

```
##  $ : chr [1:48] "[{'id': 236" "'name': 'suicide'}" "{'id': 240" "'name': 'underdog'}" ...
##  $ : chr [1:8] "[{'id': 6984" "'name': 'racial segregation'}" "{'id': 10235" "'name': 'family relati
##  $ : chr "[]"
##  $ : chr [1:10] "[{'id': 4034" "'name': 'class society'}" "{'id': 7579" "'name': 'rich woman - poor m
##  $ : chr [1:30] "[{'id': 2082" "'name': 'sailing trip'}" "{'id': 2913" "'name': 'diary'}" ...
##  $ : chr [1:10] "[{'id': 5306" "'name': 'boss'}" "{'id': 5572" "'name': 'fistfight'}" ...
##  $ : chr [1:12] "[{'id': 169078" "'name': 'nitrous oxide'}" "{'id': 169079" "'name': 'candid camera'}
##  $ : chr [1:8] "[{'id': 441" "'name': 'assassination'}" "{'id': 477" "'name': 'train station'}" ...
##  $ : chr "[]"
##  $ : chr [1:12] "[{'id': 964" "'name': 'servant'}" "{'id': 1299" "'name': 'monster'}" ...
##  $ : chr [1:4] "[{'id': 3133" "'name': 'vampire'}" "{'id': 3630" "'name': 'half vampire'}]"
##  $ : chr [1:2] "[{'id': 155808" "'name': 'hometown'}]"
##  $ : chr [1:44] "[{'id': 720" "'name': 'helicopter'}" "{'id': 1261" "'name': 'river'}" ...
##  $ : chr [1:8] "[{'id': 14531" "'name': 'play'}" "{'id': 185011" "'name': 'hamlet'}" ...
##  $ : chr [1:36] "[{'id': 90" "'name': 'paris'}" "{'id': 376" "'name': 'neo-nazi'}" ...
##  $ : chr "[]"
##  $ : chr [1:6] "[{'id': 2398" "'name': 'narration'}" "{'id': 12396" "'name': 'hollywood'}" ...
##  $ : chr [1:20] "[{'id': 417" "'name': 'corruption'}" "{'id': 1568" "'name': 'undercover'}" ...
##   [list output truncated]
```

As I did with the *genres Production_Companies* and *Production_Countries* with the *Movies* Dataset I want to make the keywords column only reference the actual keyword(s) In order to make the analysis of data is clean and easy to read.

```r
# Function to parse and extract country names from JSON within a list
extract_keywords <- function(list_element) {
  if (length(list_element) == 0 || all(is.na(list_element))) return(NA_character_)

  json_string <- paste0("[", paste(list_element, collapse = ", "), "]")
  json_string <- gsub("'", '"', json_string)

  parsed_data <- tryCatch({
    fromJSON(json_string)
  }, error = function(e) {
    return(NA_character_)
  })

  if (is.null(parsed_data)) return(NA_character_)

  keyword_text <- sapply(parsed_data, function(x) {
    if (is.list(x) && !is.null(x$name)) {
      return(x$name)
    } else {
      return(NA_character_)
    }
  })

  keyword_text <- keyword_text[!is.na(keyword_text)]
  return(paste(keyword_text, collapse = "; "))
}
```

```r
# Apply the function to the production_countries column
keywords$keywords <- vapply(keywords$keywords, extract_keywords, character(1))
```

```r
# Verify the cleaned production_countries column
print(head(keywords$keywords))
```

```
## [1] "jealousy; toy; boy; friendship; friends; rivalry; boy next door; new toy; toy comes to life"
## [2] ""
## [3] "fishing; best friend; duringcreditsstinger; old men"
## [4] "based on novel; interracial relationship; single mother; divorce; chick flick"
## [5] "baby; midlife crisis; confidence; aging; daughter; mother daughter relationship; pregnancy; con
## [6] "robbery; detective; bank; obsession; chase; shooting; thief; honor; murder; suspense; heist; be
```

Now to remove any rows with NA values from the keywords dataframe.

```r
# Drop rows with NA values in the dataframe
keywords <- na.omit(keywords)

# Verify the changes
head(keywords)
```

```
##      id
## 1   862
## 2  8844
## 3 15602
## 4 31357
## 5 11862
## 6   949
##
## 1
## 2
## 3
## 4
## 5
## 6 robbery; detective; bank; obsession; chase; shooting; thief; honor; murder; suspense; heist; betra
```

Now to see the top 20 occurances in the Keywords Dataset.

```r
# Separating the columns that contain combined Keywords
keywords_separated <- keywords %>%
  separate_rows(keywords, sep = "; ")


keywords_separated <- keywords_separated %>%
  filter(keywords != "" & !is.na(keywords))

# Verify the result
print(head(keywords_separated))
```

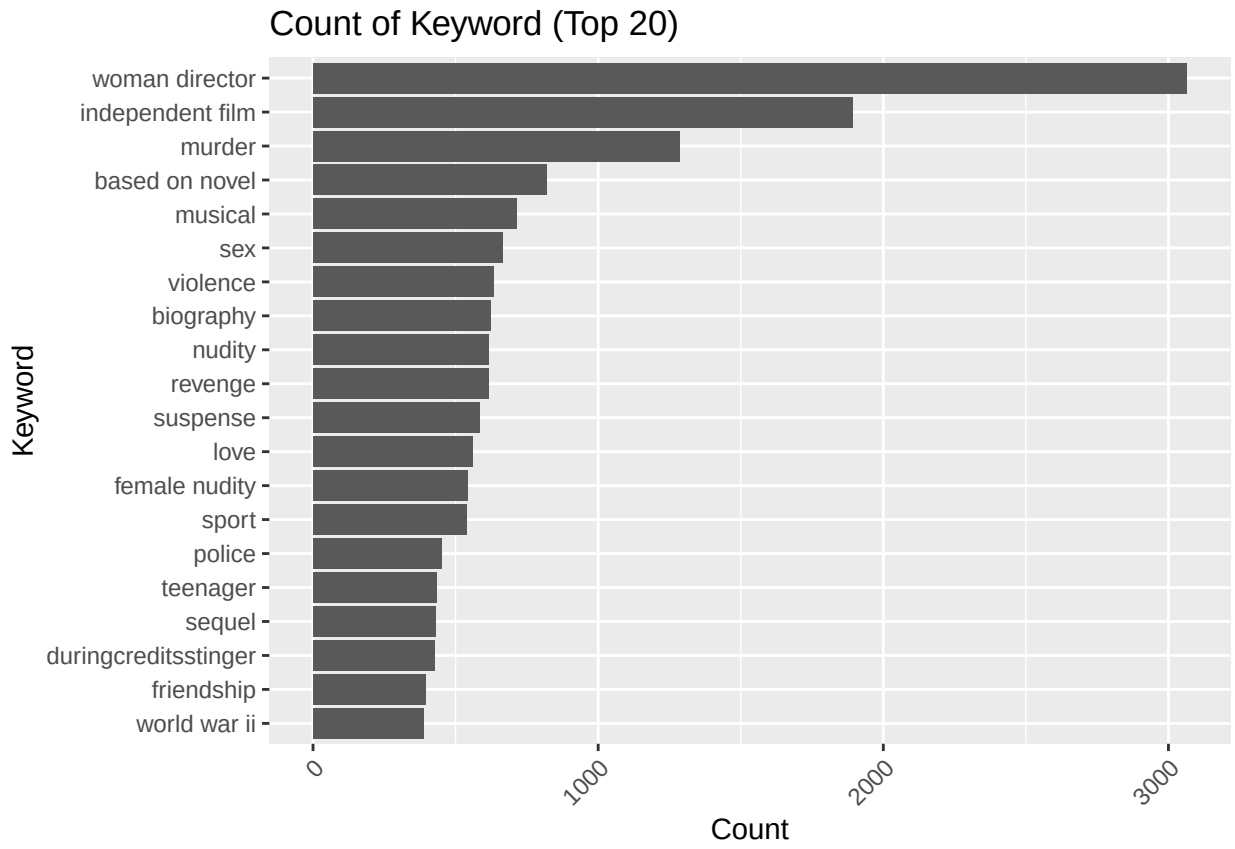
```
## # A tibble: 6 x 2
##      id keywords
##   <int> <chr>
## 1   862 jealousy
## 2   862 toy
```

```
## 3   862 boy
## 4   862 friendship
## 5   862 friends
## 6   862 rivalry
```

```r
# Count the occurrences of each Keyword
keywords_occur <- keywords_separated %>%
  count(keywords, sort = TRUE)

# Print the Keyword counts
print(keywords_occur)
```

```
## # A tibble: 19,422 x 2
##    keywords             n
##    <chr>            <int>
##  1 woman director    3064
##  2 independent film  1891
##  3 murder            1287
##  4 based on novel     818
##  5 musical            713
##  6 sex                663
##  7 violence           634
##  8 biography          622
##  9 nudity             617
## 10 revenge            614
## # i 19,412 more rows
```

```r
# filtering to the top 20
top_20_keywords <- keywords_occur %>%
  top_n(20, n) %>%
  arrange(desc(n))


# Plot the counts of the Top 20 Keywords
ggplot(top_20_keywords, aes(x = reorder(keywords, n), y = n)) +
  geom_bar(stat = "identity") +
  labs(title = "Count of Keyword (Top 20)",
       x = "Keyword",
       y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  coord_flip()
```



As you can see Women's Director is the Most Used Keyword among the data.

```
# Merge Movies and Keywords on the 'id' column using a left join
merged_df <- merge(movies, keywords, by = "id", all.x = TRUE)

# Check the structure and head of the merged dataframe
str(merged_df)
```

Since the *id* column is in in both the movies and keywords datasets and they represent the same movie, the datasets can be merged.

```
## 'data.frame':    46105 obs. of  19 variables:
##  $ id                    : int  2 3 5 6 11 12 13 14 15 16 ...
##  $ belongs_to_collection: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ budget                : num  NA NA 0.0105 NA 0.0289 ...
##  $ genres                :List of 46105
##   ..$ : chr "Drama; Crime"
##   ..$ : chr "Drama; Comedy"
##   ..$ : chr "Crime; Comedy"
##   ..$ : chr "Action; Thriller; Crime"
##   ..$ : chr "Adventure; Action; Science Fiction"
##   ..$ : chr "Animation; Family"
##   ..$ : chr "Comedy; Drama; Romance"
```

```
##   ..$ : chr "Drama"
##   ..$ : chr "Mystery; Drama"
##   ..$ : chr "Drama; Crime; Music"
##   ..$ : chr "Horror; Thriller; Mystery"
##   ..$ : chr "Adventure; Fantasy; Action; Thriller; Science Fiction"
##   ..$ : chr "Drama; Science Fiction"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Documentary"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Action; Crime"
##   ..$ : chr "Drama; War"
##   ..$ : chr "Drama"
##   ..$ : chr "Drama; Music; Romance"
##   ..$ : chr "Drama; War"
##   ..$ : chr "Animation; Science Fiction"
##   ..$ : chr "Western"
##   ..$ : chr "Animation; Comedy; Family"
##   ..$ : chr "Science Fiction; Drama; Romance"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Drama; Thriller; Crime"
##   ..$ : chr "Science Fiction; Mystery; Adventure"
##   ..$ : chr "Science Fiction; Thriller; Mystery"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Crime; Drama; Thriller"
##   ..$ : chr "Thriller; Crime; Drama"
##   ..$ : chr "Comedy; Science Fiction"
##   ..$ : chr "Drama; Music; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Drama; Comedy; Music"
##   ..$ : chr "Drama"
##   ..$ : chr "Adventure; Thriller; Science Fiction"
##   ..$ : chr "Comedy; Fantasy; Science Fiction"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Mystery; Thriller"
##   ..$ : chr "Science Fiction; Drama; Thriller"
##   ..$ : chr "Drama; Adventure; Action; History"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Adventure; Animation; Fantasy"
##   ..$ : chr "Action; Adventure; Crime; Thriller"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Adventure; Action"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Adventure; Action"
##   ..$ : chr "Drama; Music; Romance"
##   ..$ : chr "Adventure; Action"
##   ..$ : chr "Action; Comedy; Crime"
##   ..$ : chr "Documentary"
##   ..$ : chr "Documentary"
##   ..$ : chr "Crime; Drama; Mystery; Thriller"
##   ..$ : chr "Action; Thriller; Science Fiction; Adventure"
##   ..$ : chr "Action; Comedy; Crime"
##   ..$ : chr "Science Fiction; Action; Adventure"
```

```
##   ..$ : chr "Action; Drama; Adventure"
##   ..$ : chr "Comedy; Drama"
##   ..$ : chr "Comedy; Crime"
##   ..$ : chr "Thriller; Crime; Drama"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Crime; Drama"
##   ..$ : chr "Action; Drama; Thriller"
##   ..$ : chr "Adventure; Comedy; Science Fiction; Family"
##   ..$ : chr "Science Fiction; Action; Adventure; Thriller"
##   ..$ : chr "Thriller; Crime"
##   ..$ : chr "Drama; Music; Mystery"
##   ..$ : chr "Comedy; Drama; Mystery"
##   ..$ : chr "Drama; Mystery; Romance"
##   ..$ : chr "Action; Crime; Drama; Thriller"
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Romance; Comedy"
##   ..$ : chr "Comedy; Crime"
##   ..$ : chr "Drama; Thriller; Crime; Romance"
##   ..$ : chr "Crime; Drama; History; Thriller"
##   ..$ : chr "Adventure; Comedy; Family; Fantasy"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Fantasy; Drama; Animation; Adventure"
##   ..$ : chr "Drama; Fantasy"
##   ..$ : chr "Drama"
##   ..$ : chr "Adventure; Fantasy; Animation"
##   ..$ : chr "Fantasy; Adventure; Animation; Family"
##   ..$ : chr "Documentary; Music"
##   ..$ : chr "Documentary"
##   ..$ : chr "Action; Adventure; Comedy"
##   ..$ : chr "Documentary; Music"
##   ..$ : chr "Drama; Horror"
##   ..$ : chr "Romance; Fantasy; Drama; Comedy"
##   ..$ : chr "Horror"
##   ..$ : chr "Romance; Comedy; Drama"
##   ..$ : chr "Crime; Drama; Romance; Thriller"
##   .. [list output truncated]
##  $ original_language    : chr  "fi" "fi" "en" "en" ...
##  $ popularity           : num  3.86 2.29 9.03 5.54 42.15 ...
##  $ production_companies : chr  "Villealfa Filmproduction Oy; Finnish Film Foundation" "Villealfa Filn
##  $ production_countries :List of 46105
##   ..$ : chr "Finland"
##   ..$ : chr "Finland"
##   ..$ : chr "United States of America"
##   ..$ : chr "Japan; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Argentina; Denmark; Finland; France; Germany; Iceland; Italy; Netherlands; Norway; Swed
##   ..$ : chr "Germany; United Kingdom"
```

```
##   ..$ : chr "France"
##   ..$ : chr "Germany"
##   ..$ : chr "Canada; Spain"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Germany; United States of America"
##   ..$ : chr "Israel; Sweden"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "Japan"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Mexico"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America; Canada; Germany"
##   ..$ : chr "United States of America; United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "Spain"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Palestinian Territory; France; Germany; Israel; Netherlands"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "Germany; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France; United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Austria; Switzerland; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America; Hong Kong; United Kingdom"
##   ..$ : chr "China"
##   ..$ : chr "United States of America"
##   ..$ : chr "Japan"
##   ..$ : chr "Uruguay; United States of America; Germany; Paraguay"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Germany; Ireland"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Spain"
##   ..$ : chr "Austria"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "Spain; France"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "France; United States of America"
```

```
##   ..$ : chr "Denmark"
##   ..$ : chr "United States of America"
##   ..$ : chr "Germany"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "France; Poland"
##   ..$ : chr "France; Poland"
##   ..$ : chr "France; Poland; Switzerland"
##   ..$ : chr "United States of America"
##   ..$ : chr "Denmark; Sweden"
##   ..$ : chr "South Korea; Germany"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "Ireland; Luxembourg; Russia; United Kingdom; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Australia; United Kingdom; United States of America"
##   ..$ : chr "New Zealand; United States of America"
##   ..$ : chr "New Zealand; United States of America"
##   ..$ : chr "New Zealand; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Poland"
##   ..$ : chr "Poland"
##   ..$ : chr "Japan"
##   ..$ : chr "Japan"
##   ..$ : chr "United States of America"
##   ..$ : chr ""
##   ..$ : chr "France; United Kingdom; United States of America"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Denmark; Sweden"
##   ..$ : chr "Spain"
##   .. [list output truncated]
##  $ release_date         : Date, format: "1988-10-21" "1986-10-16" ...
##  $ revenue              : num  NA NA 0.0113 0.0319 2.0406 ...
##  $ runtime              : num  69 76 98 110 121 100 142 122 119 140 ...
##  $ status               : chr  "Released" "Released" "Released" "Released" ...
##  $ title                : chr  "Ariel" "Shadows in Paradise" "Four Rooms" "Judgment Night" ...
##  $ vote_average         : num  7.1 7.1 6.5 6.4 8.1 7.6 8.2 7.9 8 7.7 ...
##  $ vote_count           : int  44 35 539 79 6778 6292 8147 3438 1244 392 ...
##  $ release_year         : chr  "1988" "1986" "1995" "1993" ...
##  $ profit               : num  NA NA 0.000789 NA 2.011627 ...
##  $ weighted_rating      : num  6.29 6.19 6.42 6.05 8.09 ...
##  $ keywords             : chr  "underdog; prison; factory worker; prisoner; helsinki; independent fi
```

```
head(merged_df)
```

```
##   id belongs_to_collection     budget                          genres
## 1  2                     1         NA                     Drama; Crime
## 2  3                     1         NA                    Drama; Comedy
## 3  5                     1 0.01052659                    Crime; Comedy
## 4  6                     1         NA           Action; Thriller; Crime
```

```
## 5 11                    1 0.02894813 Adventure; Action; Science Fiction
## 6 12                    1 0.24737493                  Animation; Family
##   original_language popularity
## 1                fi   3.860491
## 2                fi   2.292110
## 3                en   9.026586
## 4                en   5.538671
## 5                en  42.149697
## 6                en  25.497794
##                                                production_companies
## 1                Villealfa Filmproduction Oy; Finnish Film Foundation
## 2                                     Villealfa Filmproduction Oy
## 3                                     Miramax Films; A Band Apart
## 4 Universal Pictures; Largo Entertainment; JVC Entertainment Networks
## 5                   Lucasfilm; Twentieth Century Fox Film Corporation
## 6                                          Pixar Animation Studios
##              production_countries release_date    revenue runtime   status
## 1                         Finland   1988-10-21         NA      69 Released
## 2                         Finland   1986-10-16         NA      76 Released
## 3        United States of America   1995-12-09 0.01131609      98 Released
## 4 Japan; United States of America   1993-10-15 0.03194015     110 Released
## 5        United States of America   1977-05-25 2.04057477     121 Released
## 6        United States of America   2003-05-30 2.47463232     100 Released
##                 title vote_average vote_count release_year       profit
## 1               Ariel          7.1         44         1988           NA
## 2 Shadows in Paradise          7.1         35         1986           NA
## 3          Four Rooms          6.5        539         1995 0.0007894945
## 4      Judgment Night          6.4         79         1993           NA
## 5           Star Wars          8.1       6778         1977 2.0116266402
## 6        Finding Nemo          7.6       6292         2003 2.2272573910
##   weighted_rating
## 1        6.294764
## 2        6.192691
## 3        6.424489
## 4        6.052879
## 5        8.085410
## 6        7.587055
##
## 1
## 2
## 3
## 4
## 5                                android; galaxy; hermit; death star; lightsaber; jedi; rescue missi
## 6 father son relationship; harbor; underwater; fish tank; great barrier reef; missing child; aftercr
```

- Credits Dataset

First to import the Credits csv file and look at the columns of the data to see the format:

```
credits <- read.csv("credits.csv")

head(credits)
```

```
##
## 1 [{'cast_id': 14, 'character': 'Woody (voice)', 'credit_id': '52fe4284c3a36847f8024fbd', 'gender': 1, 'id': 12903, 'name': 'Sarah Freema
## 2 [{'cast_id': 41, 'character': 'Baker', 'credit_id': '55935946c3a36869d1001b4d', 'gender
## 3 [{'cast_id': 2, 'character': 'Max Goldman', 'credit_id': '53e5fcc2c3a3684430000d65', 'gender': 2, 'id': 16523, 'name': 'Burgess Meredith', 'order': 5, 'pr
## 4 [{'cast_id': 1, 'character': 'Alan Parrish', 'credit_id': '56d1b15fc3a3681e4a008b6b', 'gender': 2, 'id': 1276777, 'name': 'Lamont Johnson', 'order': 8, 'proi
## 5 [{'cast_id': 1, 'character': "Savannah 'Vannah' Jackson", 'credit_id': '52fe44959251416c75039f0b', 'gender': 2, 'id': 26510, 'name': 'Eugene Levy', 'order': 10, 'pro
## 6 [{'cast_id': 25, 'character': 'Lt. Vincent Hanna', 'credit_id': '52fe4292c3a36847f80291f5', 'gender
##
## 1 [{'credit_id': '52fe4284c3a36847f8024f49', 'department': 'Directing', 'gender': 2, 'id': 7879, 'job
## 2
## 3
## 4
## 5
## 6
##      id
## 1   862
## 2  8844
## 3 15602
## 4 31357
## 5 11862
## 6   949
```

cast and *crew* like the two previous datasets has *chr* strings that I want to make into a list so I will use my function

```
# Apply the convert_to_list function to the 'credits' column
credits$cast <- convert_to_list(credits$cast)

# Check the structure of the new 'credits' column
str(credits$cast)
```

First lets start with the *cast*

```
## List of 45476
##  $ : chr [1:104] "[{'cast_id': 14" "'character': 'Woody (voice)'" "'credit_id': '52fe4284c3a36847f80
##  $ : chr [1:208] "[{'cast_id': 1" "'character': 'Alan Parrish'" "'credit_id': '52fe44bfc3a36847f80a7
##  $ : chr [1:56] "[{'cast_id': 2" "'character': 'Max Goldman'" "'credit_id': '52fe466a9251416c75077a8
##  $ : chr [1:81] "[{'cast_id': 1" "'character': \"Savannah 'Vannah' Jackson\"" "'credit_id': '52fe447
##  $ : chr [1:96] "[{'cast_id': 1" "'character': 'George Banks'" "'credit_id': '52fe44959251416c75039e
##  $ : chr [1:522] "[{'cast_id': 25" "'character': 'Lt. Vincent Hanna'" "'credit_id': '52fe4292c3a3684
##  $ : chr [1:456] "[{'cast_id': 1" "'character': 'Linus Larrabee'" "'credit_id': '52fe44959251416c750
##  $ : chr [1:56] "[{'cast_id': 2" "'character': 'Tom Sawyer'" "'credit_id': '52fe46bdc3a36847f810f771
##  $ : chr [1:48] "[{'cast_id': 1" "'character': 'Darren Francis Thomas McCord'" "'credit_id': '52fe44c
##  $ : chr [1:160] "[{'cast_id': 1" "'character': 'James Bond'" "'credit_id': '52fe426ec3a36847f801e10c
##  $ : chr [1:144] "[{'cast_id': 1" "'character': 'Andrew Shepherd'" "'credit_id': '52fe44dac3a36847f80
##  $ : chr [1:88] "[{'cast_id': 9" "'character': 'Count Dracula'" "'credit_id': '52fe44b79251416c7503e8
##  $ : chr [1:104] "[{'cast_id': 1" "'character': 'Balto (voice)'" "'credit_id': '52fe4409c3a368484e00
##  $ : chr [1:272] "[{'cast_id': 1" "'character': 'Richard Nixon'" "'credit_id': '52fe43c59251416c7501c
##  $ : chr [1:248] "[{'cast_id': 1" "'character': 'Morgan Adams'" "'credit_id': '52fe42f4c3a36847f802f0
##  $ : chr [1:360] "[{'cast_id': 4" "'character': \"Sam 'Ace' Rothstein\"" "'credit_id': '52fe424dc3a30
##  $ : chr [1:136] "[{'cast_id': 6" "'character': 'Marianne Dashwood'" "'credit_id': '52fe43cec3a36847f
```

```
##  $ : chr [1:192] "[{'cast_id': 42" "'character': 'Ted the Bellhop'" "'credit_id': '52fe420dc3a36847f8
##  $ : chr [1:120] "[{'cast_id': 1" "'character': 'Ace Ventura'" "'credit_id': '52fe44dfc3a36847f80af2
##  $ : chr [1:120] "[{'cast_id': 1" "'character': 'John'" "'credit_id': '52fe44509251416c7503059b'" "'g
##  $ : chr [1:128] "[{'cast_id': 1" "'character': 'Chili Palmer'" "'credit_id': '52fe448dc3a36847f809c
##  $ : chr [1:120] "[{'cast_id': 8" "'character': 'Helen Hudson'" "'credit_id': '52fe430ec3a36847f8037
##  $ : chr [1:137] "[{'cast_id': 1" "'character': 'Robert Rath'" "'credit_id': '52fe451cc3a36847f80bd1
##  $ : chr [1:40] "[{'cast_id': 1" "'character': 'Jessie Caldwell'" "'credit_id': '52fe45119251416c7504
##  $ : chr [1:112] "[{'cast_id': 1" "'character': 'Ben Sanderson'" "'credit_id': '52fe4245c3a36847f801
##  $ : chr [1:72] "[{'cast_id': 2" "'character': 'Othello'" "'credit_id': '52fe46d29251416c75084b0b'" "
##  $ : chr [1:130] "[{'cast_id': 9" "'character': 'Young Roberta Martin'" "'credit_id': '52fe44dec3a368
##  $ : chr [1:208] "[{'cast_id': 1" "'character': 'Anne Elliott'" "'credit_id': '52fe46ff9251416c7508a8
##  $ : chr [1:301] "[{'cast_id': 24" "'character': 'One'" "'credit_id': '52fe428ac3a36847f8026cdf'" "'g
##  $ : chr [1:53] "[{'cast_id': 2" "'character': 'Xiao Jingbao'" "'credit_id': '52fe46559251416c910511
##  $ : chr [1:120] "[{'cast_id': 13" "'character': 'Louanne Johnson'" "'credit_id': '52fe4546c3a36847f8
##  $ : chr [1:520] "[{'cast_id': 41" "'character': 'James Cole'" "'credit_id': '52fe4212c3a36847f8001b3
##  $ : chr [1:56] "[{'cast_id': 2" "'character': 'Henri Guillaumet'" "'credit_id': '52fe49b4c3a368484e
##  $ : chr [1:144] "[{'cast_id': 1" "'character': 'Babe the Gallant Pig (voice)'" "'credit_id': '52fe45
##  $ : chr [1:96] "[{'cast_id': 1" "'character': 'Dora Carrington'" "'credit_id': '52fe471bc3a36847f811
##  $ : chr [1:336] "[{'cast_id': 7" "'character': 'Sister Helen Prejean'" "'credit_id': '52fe426ac3a368
##  $ : chr [1:40] "[{'cast_id': 12" "'character': 'Tomas Minton'" "'credit_id': '5554bace9251416dd6002a
##  $ : chr [1:104] "[{'cast_id': 1" "'character': 'Diane Barrows'" "'credit_id': '52fe452f9251416c9102a
##  $ : chr [1:112] "[{'cast_id': 8" "'character': 'Cher Horowitz'" "'credit_id': '52fe4510c3a36847f80ba
##  $ : chr [1:80] "[{'cast_id': 4" "'character': 'Rev. Stephen Kumalo'" "'credit_id': '52fe456f9251416c
##  $ : chr [1:144] "[{'cast_id': 1" "'character': 'Richard III'" "'credit_id': '52fe44669251416c9100f59
##  $ : chr [1:104] "[{'cast_id': 1" "'character': 'Anthony Curtis'" "'credit_id': '52fe44409251416c7502
##  $ : chr [1:80] "[{'cast_id': 1" "'character': 'Robert Merivel'" "'credit_id': '52fe45999251416c91037
##  $ : chr [1:88] "[{'cast_id': 1" "'character': 'Lord Rayden'" "'credit_id': '52fe44e5c3a36847f80b086f
##  $ : chr [1:112] "[{'cast_id': 4" "'character': 'Suzanne Stone Maretto'" "'credit_id': '52fe4255c3a30
##  $ : chr [1:272] "[{'cast_id': 1" "'character': 'Finn Dodd'" "'credit_id': '52fe44959251416c75039e0b
##  $ : chr [1:393] "[{'cast_id': 17" "'character': 'Detective David Mills'" "'credit_id': '52fe4279c3a3
##  $ : chr [1:128] "[{'cast_id': 1" "'character': 'Pocahontas (voice)'" "'credit_id': '52fe43819251416c
##  $ : chr [1:32] "[{'cast_id': 1" "'character': 'Camille Baker'" "'credit_id': '52fe44a4c3a36847f80a1c
##  $ : chr [1:144] "[{'cast_id': 19" "'character': 'Michael McManus'" "'credit_id': '52fe4260c3a36847f8
##  $ : chr [1:56] "[{'cast_id': 1" "'character': 'Cynthia Mckay'" "'credit_id': '52fe4babc3a36847f820ea
##  $ : chr [1:96] "[{'cast_id': 1" "'character': 'Lenny'" "'credit_id': '52fe44419251416c7502e523'" "'g
##  $ : chr [1:40] "[{'cast_id': 2" "'character': 'Gino'" "'credit_id': '52fe4787c3a36847f813acfb'" "'ge
##  $ : chr [1:104] "[{'cast_id': 1" "'character': 'Sheriff Tom Palmer'" "'credit_id': '52fe4503c3a36848
##  $ : chr [1:64] "[{'cast_id': 2" "'character': 'Sadie Flood'" "'credit_id': '52fe49de9251416c750d60cb
##  $ : chr [1:40] "[{'cast_id': 1" "'character': 'Alex Cole'" "'credit_id': '52fe4aa2c3a368484e15ee3d'
##  $ : chr [1:80] "[{'cast_id': 10" "'character': 'Claudia Larson'" "'credit_id': '52fe44dbc3a36847f80a
##  $ : chr [1:112] "[{'cast_id': 11" "'character': 'Pablo Neruda'" "'credit_id': '52fe43e29251416c7502f
##  $ : chr [1:40] "[{'cast_id': 3" "'character': 'Pierre Lamontagne'" "'credit_id': '52fe4a259251416c75
##  $ : chr [1:80] "[{'cast_id': 1" "'character': 'Omri'" "'credit_id': '52fe44309251416c7502bdb5'" "'ge
##  $ : chr [1:136] "[{'cast_id': 1" "'character': 'Karen McCann'" "'credit_id': '52fe470d9251416c7508c4
##  $ : chr [1:128] "[{'cast_id': 10" "'character': 'Glenn Holland'" "'credit_id': '52fe4330c3a36847f804
##  $ : chr [1:112] "[{'cast_id': 18" "'character': 'Ashtray'" "'credit_id': '52fe43929251416c75015a3f'
##  $ : chr [1:96] "[{'cast_id': 1" "'character': 'Roz'" "'credit_id': '52fe47ef9251416c750aa5a7'" "'ger
##  $ : chr [1:96] "[{'cast_id': 14" "'character': 'Bud Macintosh'" "'credit_id': '52fe4506c3a36847f80b7
##  $ : chr [1:120] "[{'cast_id': 1" "'character': 'Dr. Benjamin Trace'" "'credit_id': '52fe44529251416c
##  $ : chr [1:40] "[{'cast_id': 2" "'character': 'Gennaro Spirito'" "'credit_id': '52fe459ac3a36847f80c
##  $ : chr [1:64] "[{'cast_id': 4" "'character': 'Loli'" "'credit_id': '52fe43c5c3a36847f806e5bf'" "'ge
##  $ : chr [1:144] "[{'cast_id': 1" "'character': 'Craig Jones'" "'credit_id': '52fe43999251416c75016ac
##  $ : chr [1:264] "[{'cast_id': 1" "'character': 'Seth Gecko'" "'credit_id': '52fe4271c3a36847f801f1e
##  $ : chr [1:144] "[{'cast_id': 1" "'character': 'Max Kirkpatrick'" "'credit_id': '52fe44949251416c750
```

```
##  $ : chr [1:168] "[{'cast_id': 1" "'character': 'Grover'" "'credit_id': '52fe4592c3a368484e062d21'" 
##  $ : chr [1:128] "[{'cast_id': 4" "'character': 'Henri Fortin / Jean Valjean'" "'credit_id': '52fe47
##  $ : chr [1:280] "[{'cast_id': 1" "'character': 'Lewis Farrell'" "'credit_id': '52fe4402c3a368484e00
##  $ : chr [1:56] "[{'cast_id': 1" "'character': 'David Leary'" "'credit_id': '52fe461a9251416c910493b
##  $ : chr [1:72] "[{'cast_id': 1" "'character': 'Colonel Hendricksson'" "'credit_id': '52fe44dcc3a368
##  $ : chr [1:56] "[{'cast_id': 14" "'character': 'Herself'" "'credit_id': '59d54df4c3a36845610206b5'" 
##  $ : chr [1:232] "[{'cast_id': 1" "'character': 'Freddy Gale'" "'credit_id': '52fe4550c3a368484e0531
##  $ : chr [1:120] "[{'cast_id': 1" "'character': 'Annie Laird'" "'credit_id': '52fe4512c3a36847f80bad
##  $ : chr [1:40] "[{'cast_id': 1001" "'character': 'Razieh'" "'credit_id': '52fe470cc3a36847f8120b0f'
##  $ : chr [1:112] "[{'cast_id': 3" "'character': \"Jimmy 'The Saint' Tosnia\"" "'credit_id': '52fe423
##  $ : chr [1:112] "[{'cast_id': 11" "'character': 'Antonia'" "'credit_id': '52fe4286c3a36847f8025d09'
##  $ : chr [1:89] "[{'cast_id': 2" "'character': 'Poppa'" "'credit_id': '52fe4b889251416c7510604b'" "'
##  $ : chr [1:128] "[{'cast_id': 2" "'character': 'Oona Hart'" "'credit_id': '52fe4d3c9251416c75134781
##  $ : chr [1:64] "[{'cast_id': 1" "'character': 'William Adamson'" "'credit_id': '52fe44aac3a36847f80a
##  $ : chr [1:121] "[{'cast_id': 1" "'character': \"Captain Christopher 'Skipper' Sheldon\"" "'credit_
##  $ : chr [1:128] "[{'cast_id': 2" "'character': 'Robert Grant'" "'credit_id': '52fe47239251416c7508f
##  $ : chr [1:136] "[{'cast_id': 1" "'character': 'Mike Donnelly'" "'credit_id': '52fe45c09251416c7506
##  $ : chr [1:96] "[{'cast_id': 11" "'character': 'Gene Watson'" "'credit_id': '52fe4334c3a36847f80421
##  $ : chr [1:48] "[{'cast_id': 3" "'character': 'August King'" "'credit_id': '52fe4657c3a368484e08b609
##  $ : chr [1:88] "[{'cast_id': 1" "'character': 'Mary Reilly'" "'credit_id': '52fe44dbc3a36847f80ae25
##  $ : chr [1:72] "[{'cast_id': 12" "'character': 'Maximillian | Pauly | Guido'" "'credit_id': '52fe44
##  $ : chr [1:314] "[{'cast_id': 10" "'character': \"Tommy 'Birdman' Rowland\"" "'credit_id': '52fe44e
##  $ : chr [1:121] "[{'cast_id': 1" "'character': \"Maj. Vic 'Deak' Deakins\"" "'credit_id': '52fe44dc
##  $ : chr [1:112] "[{'cast_id': 3" "'character': 'Henry'" "'credit_id': '52fe4583c3a36847f80cb93f'" "
##  $ : chr [1:160] "[{'cast_id': 2" "'character': 'Vinz'" "'credit_id': '52fe423fc3a36847f800f88f'" "'
##  $ : chr [1:168] "[{'cast_id': 1" "'character': 'Jo'" "'credit_id': '52fe46c8c3a36847f8111c0b'" "'ge
##  $ : chr [1:56] "[{'cast_id': 2" "'character': 'Narrator'" "'credit_id': '52fe469ac3a368484e099bcf'" 
##  $ : chr [1:56] "[{'cast_id': 1" "'character': 'Mayor John Pappas'" "'credit_id': '52fe43ee9251416c7
##   [list output truncated]
```

Now I want to get the director from the *crew*, I will do this in 3 steps.

1. I need to parse the crew data so that the JSON string for each row are now seperated by individual parameters

```r
# Function to parse the character string and convert to list
parse_crew <- function(cast_text) {
  text <- sub("\\£\\$\\£", "", gsub("\'|\"|\\[|\\]|\\{|\\}|,", "", gsub("\'credit_id\':|\'department\':
  data <- data.frame(matrix(trimws(unlist(strsplit(text, "\\£\\$\\£"))), ncol = 7, byrow = TRUE), string
  names(data) <- c("credit_id", "department", "gender", "id", "job", "name", "profile_path")
  return(data)
}

# Apply the parsing function to the 'crew' column and inspect parsed data
credits$parsed_crew <- lapply(credits$crew, parse_crew)

# Check if any parsed data is problematic
parsed_lengths <- sapply(credits$parsed_crew, nrow)
problematic_indices <- which(parsed_lengths == 0 | is.na(parsed_lengths))
print(problematic_indices)
```

```
##   [1]   190   615   636   662   712   864  1108  1273  1394  1406  1462  1490
##  [13]  1627  1629  1633  1635  1692  2372  2426  2968  3109  3461  3758  3764
```

```
##  [25]  3946  4188  4330  4334  4631  5187  5486  5620  5768  5880  6057  6116
##  [37]  6589  6610  6618  6791  6993  7144  7199  7274  8214  8234  8249  8293
##  [49]  9168  9299  9784  9831 10244 10866 11289 11392 11946 12254 12339 13052
##  [61] 13170 13171 13227 13236 13506 13593 13600 13624 13661 13757 13760 13914
##  [73] 13946 14068 14410 14455 14537 14546 14616 14692 14696 14752 14802 15036
##  [85] 15038 15083 15250 15264 15317 15377 15443 15530 15620 15773 15792 15858
##  [97] 15878 15971 16430 16606 16609 16618 16663 16837 16838 16842 16905 16916
## [109] 16954 17105 17203 17227 17326 17332 17383 17424 17428 17459 17463 17504
## [121] 17575 17659 17685 17686 17838 18037 18039 18075 18363 18441 18511 18573
## [133] 18581 18593 18617 18634 18673 18711 18748 18751 18806 18869 18941 19034
## [145] 19108 19119 19185 19194 19234 19314 19323 19352 19367 19369 19374 19414
## [157] 19416 19450 19473 19518 19542 19543 19546 19646 19693 19703 19714 19746
## [169] 19840 20159 20166 20176 20301 20315 20350 20433 20461 20484 20504 20589
## [181] 20627 20631 20711 20743 20778 20837 20893 21003 21030 21093 21167 21184
## [193] 21240 21250 21253 21391 21392 21423 21448 21484 21633 21636 21642 21678
## [205] 21708 21776 21946 22019 22198 22321 22342 22376 22377 22396 22401 22409
## [217] 22424 22491 22567 22685 22690 22692 22798 23005 23095 23128 23242 23323
## [229] 23366 23403 23404 23429 23829 23937 24018 24025 24157 24328 24330 24410
## [241] 24603 24722 25000 25036 25190 25195 25216 25348 25450 25524 25804 25876
## [253] 25997 26005 26258 26338 26734 26744 26809 26909 27077 27261 27320 27513
## [265] 27547 27564 27582 27603 27693 27696 27805 27811 27812 27900 27954 28019
## [277] 28146 28261 28295 28313 28690 28700 28735 28763 28765 28854 28978 28999
## [289] 29022 29055 29153 29214 29309 29323 29327 29434 29528 29585 29646 29697
## [301] 29717 29719 29746 29752 29769 29919 30023 30134 30137 30213 30241 30274
## [313] 30356 30376 30388 30407 30432 30523 30567 30590 30660 30695 30737 30757
## [325] 30768 30870 30914 30954 31020 31021 31034 31045 31069 31092 31137 31160
## [337] 31240 31256 31270 31280 31374 31376 31440 31450 31452 31475 31491 31556
## [349] 31617 31667 31687 31714 31733 31740 31744 31873 31960 31996 32016 32039
## [361] 32054 32150 32153 32210 32236 32238 32274 32278 32287 32295 32301 32312
## [373] 32336 32341 32353 32407 32481 32510 32691 32820 32826 33113 33431 33444
## [385] 33621 33671 33734 33766 33901 34015 34134 34283 34284 34285 34286 34288
## [397] 34292 34293 34295 34298 34299 34301 34302 34303 34327 34332 34340 34362
## [409] 34365 34375 34388 34390 34422 34431 34433 34434 34437 34439 34442 34445
## [421] 34453 34463 34641 34678 34688 34783 34787 34791 34842 34887 34970 35046
## [433] 35082 35089 35095 35128 35153 35200 35201 35301 35326 35327 35334 35339
## [445] 35357 35359 35408 35466 35491 35549 35550 35588 35642 35670 35672 35693
## [457] 35740 35785 35841 35886 35891 35894 36017 36053 36105 36113 36132 36205
## [469] 36207 36219 36224 36233 36301 36365 36379 36400 36416 36433 36447 36459
## [481] 36481 36484 36485 36491 36524 36535 36536 36537 36541 36542 36544 36574
## [493] 36579 36580 36581 36584 36585 36591 36593 36595 36624 36625 36630 36632
## [505] 36696 36717 36765 36794 36805 36840 36845 36872 36876 36890 36910 36960
## [517] 36967 36982 36986 36998 37054 37102 37117 37130 37134 37163 37226 37250
## [529] 37253 37280 37297 37301 37473 37504 37566 37627 37655 37669 37677 37752
## [541] 37766 37795 37809 37871 38043 38165 38173 38195 38254 38341 38350 38365
## [553] 38382 38389 38410 38448 38469 38501 38529 38545 38556 38572 38581 38614
## [565] 38675 38689 38708 38759 38841 38877 38884 38899 38905 38922 38939 39177
## [577] 39293 39332 39335 39351 39387 39440 39526 39552 39689 39703 39739 39743
## [589] 39744 39745 39748 39752 39753 39757 39759 39760 39761 39882 39912 39913
## [601] 39983 40077 40094 40136 40153 40155 40160 40193 40215 40216 40278 40365
## [613] 40406 40417 40424 40436 40466 40479 40485 40561 40588 40629 40630 40789
## [625] 40805 40895 40928 40992 41032 41033 41087 41169 41205 41218 41371 41382
## [637] 41411 41415 41496 41524 41541 41553 41564 41630 41631 41637 41638 41648
## [649] 41730 41731 41743 41772 41798 41804 41817 41836 41857 41858 41893 41992
## [661] 41998 42005 42157 42190 42202 42241 42270 42274 42319 42336 42375 42376
```

```
## [673] 42382 42402 42427 42430 42581 42585 42604 42680 42688 42694 42720 42746
## [685] 42767 42770 42779 42795 42810 42859 42866 42929 42952 42970 42971 43035
## [697] 43085 43094 43180 43247 43437 43438 43440 43441 43453 43493 43534 43549
## [709] 43552 43692 43733 43754 43760 43768 43780 43781 43800 43810 43827 43861
## [721] 43876 43912 43924 43932 43934 43952 43960 43973 43974 44041 44059 44060
## [733] 44102 44108 44109 44162 44163 44318 44342 44388 44424 44427 44434 44505
## [745] 44586 44603 44634 44703 44711 44712 44719 44734 44745 44764 44795 44872
## [757] 44982 44989 45077 45078 45081 45087 45097 45132 45159 45171 45237 45262
## [769] 45278 45349 45372
```

2. I need to make sure that the parsed data has transferred over into each parameter in a consistent manner, i.e name from the parse has went to name in the dataframe

```
# Function to ensure consistency in parsed data frames
ensure_consistency <- function(parsed_data) {
  if (is.data.frame(parsed_data) && nrow(parsed_data) > 0) {
    return(parsed_data)
  } else {
    return(data.frame(matrix(NA, ncol = 7, dimnames = list(NULL, c("credit_id", "department", "gender",
  }
}

# Apply consistency check
credits$parsed_crew <- lapply(credits$parsed_crew, ensure_consistency)
```

3. Now I need to create and implement a function that will extract the Director from the parsed data while also replacing the crew column in the credits dataframe with a new column only displaying the director of the film

```
# Function to extract the director's name
extract_director_name <- function(parsed_data) {
  if (is.null(parsed_data)) return(NA)
  for (row in 1:nrow(parsed_data)) {
    if (!is.null(parsed_data$job[row]) && !is.na(parsed_data$job[row]) && parsed_data$job[row] == "Dire
      return(parsed_data$name[row])
    }
  }
  return(NA)
}

# Apply the extraction function to the parsed data
credits$director <- sapply(credits$parsed_crew, extract_director_name)

# Drop the 'crew' and 'parsed_crew' columns
credits <- credits[, !names(credits) %in% c("crew", "parsed_crew")]

# Check the structure of the dataframe
str(credits)
```

```
## 'data.frame':    45476 obs. of  3 variables:
##  $ cast    :List of 45476
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Woody (voice)'" "'credit_id': '52fe4284c3a36847f8024f9
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Alan Parrish'" "'credit_id': '52fe44bfc3a36847f80a7c73'
```

```
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Max Goldman'" "'credit_id': '52fe466a9251416c75077a8d'"
##   ..$ : chr  "[{'cast_id': 1" "'character': \"Savannah 'Vannah' Jackson\"" "'credit_id': '52fe447792
##   ..$ : chr  "[{'cast_id': 1" "'character': 'George Banks'" "'credit_id': '52fe44959251416c75039eb9'
##   ..$ : chr  "[{'cast_id': 25" "'character': 'Lt. Vincent Hanna'" "'credit_id': '52fe4292c3a36847f80
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Linus Larrabee'" "'credit_id': '52fe44959251416c75039d9
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Tom Sawyer'" "'credit_id': '52fe46bdc3a36847f810f771'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Darren Francis Thomas McCord'" "'credit_id': '52fe44dbc
##   ..$ : chr  "[{'cast_id': 1" "'character': 'James Bond'" "'credit_id': '52fe426ec3a36847f801e10d'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Andrew Shepherd'" "'credit_id': '52fe44dac3a36847f80adf
##   ..$ : chr  "[{'cast_id': 9" "'character': 'Count Dracula'" "'credit_id': '52fe44b79251416c7503e811
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Balto (voice)'" "'credit_id': '52fe4409c3a368484e00bb65
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Richard Nixon'" "'credit_id': '52fe43c59251416c7501d6e1
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Morgan Adams'" "'credit_id': '52fe42f4c3a36847f802f65f'
##   ..$ : chr  "[{'cast_id': 4" "'character': \"Sam 'Ace' Rothstein\"" "'credit_id': '52fe424dc3a36847
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Marianne Dashwood'" "'credit_id': '52fe43cec3a36847f807
##   ..$ : chr  "[{'cast_id': 42" "'character': 'Ted the Bellhop'" "'credit_id': '52fe420dc3a36847f8000
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Ace Ventura'" "'credit_id': '52fe44dfc3a36847f80af279'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'John'" "'credit_id': '52fe44509251416c7503059b'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Chili Palmer'" "'credit_id': '52fe448dc3a36847f809c6eb'
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Helen Hudson'" "'credit_id': '52fe430ec3a36847f8037243'
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Robert Rath'" "'credit_id': '52fe451cc3a36847f80bd107'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Jessie Caldwell'" "'credit_id': '52fe45119251416c7504ab5
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Ben Sanderson'" "'credit_id': '52fe4245c3a36847f801117b
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Othello'" "'credit_id': '52fe46d29251416c75084b0b'" "'g
##   ..$ : chr  "[{'cast_id': 9" "'character': 'Young Roberta Martin'" "'credit_id': '52fe44dec3a36847f8
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Anne Elliott'" "'credit_id': '52fe46ff9251416c7508a8a9'
##   ..$ : chr  "[{'cast_id': 24" "'character': 'One'" "'credit_id': '52fe428ac3a36847f8026cdf'" "'gend
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Xiao Jingbao'" "'credit_id': '52fe46559251416c910511c3'
##   ..$ : chr  "[{'cast_id': 13" "'character': 'Louanne Johnson'" "'credit_id': '52fe4546c3a36847f80c4
##   ..$ : chr  "[{'cast_id': 41" "'character': 'James Cole'" "'credit_id': '52fe4212c3a36847f8001b31'"
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Henri Guillaumet'" "'credit_id': '52fe49b4c3a368484e13a
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Babe the Gallant Pig (voice)'" "'credit_id': '52fe450fc
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Dora Carrington'" "'credit_id': '52fe471bc3a36847f8123a
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Sister Helen Prejean'" "'credit_id': '52fe426ac3a36847f8
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Tomas Minton'" "'credit_id': '5554bace9251416dd6002a69
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Diane Barrows'" "'credit_id': '52fe452f9251416c9102a483
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Cher Horowitz'" "'credit_id': '52fe4510c3a36847f80ba283
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Rev. Stephen Kumalo'" "'credit_id': '52fe456f9251416c91
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Richard III'" "'credit_id': '52fe44669251416c9100f591'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Anthony Curtis'" "'credit_id': '52fe44409251416c7502e19
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Robert Merivel'" "'credit_id': '52fe45999251416c91037fc
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Lord Rayden'" "'credit_id': '52fe44e5c3a36847f80b086f'"
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Suzanne Stone Maretto'" "'credit_id': '52fe4255c3a36847
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Finn Dodd'" "'credit_id': '52fe44959251416c75039e0b'" "
##   ..$ : chr  "[{'cast_id': 17" "'character': 'Detective David Mills'" "'credit_id': '52fe4279c3a3684
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Pocahontas (voice)'" "'credit_id': '52fe43819251416c750
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Camille Baker'" "'credit_id': '52fe44a4c3a36847f80a1e49
##   ..$ : chr  "[{'cast_id': 19" "'character': 'Michael McManus'" "'credit_id': '52fe4260c3a36847f8019
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Cynthia Mckay'" "'credit_id': '52fe4babc3a36847f820ea2b
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Lenny'" "'credit_id': '52fe44419251416c7502e523'" "'gen
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Gino'" "'credit_id': '52fe4787c3a36847f813acfb'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Sheriff Tom Palmer'" "'credit_id': '52fe4503c3a368484e0
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Sadie Flood'" "'credit_id': '52fe49de9251416c750d60cb'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Alex Cole'" "'credit_id': '52fe4aa2c3a368484e15ee3d'" "
```

```
##   ..$ : chr  "[{'cast_id': 10" "'character': 'Claudia Larson'" "'credit_id': '52fe44dbc3a36847f80ae0
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Pablo Neruda'" "'credit_id': '52fe43e29251416c75021dd1
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Pierre Lamontagne'" "'credit_id': '52fe4a259251416c750d
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Omri'" "'credit_id': '52fe44309251416c7502bdb5'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Karen McCann'" "'credit_id': '52fe470d9251416c7508c47d'
##   ..$ : chr  "[{'cast_id': 10" "'character': 'Glenn Holland'" "'credit_id': '52fe4330c3a36847f804115
##   ..$ : chr  "[{'cast_id': 18" "'character': 'Ashtray'" "'credit_id': '52fe43929251416c75015a3f'" "'g
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Roz'" "'credit_id': '52fe47ef9251416c750aa5a7'" "'gender
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Bud Macintosh'" "'credit_id': '52fe4506c3a36847f80b7d1b
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Dr. Benjamin Trace'" "'credit_id': '52fe44529251416c7503
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Gennaro Spirito'" "'credit_id': '52fe459ac3a36847f80d075
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Loli'" "'credit_id': '52fe43c5c3a36847f806e5bf'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Craig Jones'" "'credit_id': '52fe43999251416c75016ac3'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Seth Gecko'" "'credit_id': '52fe4271c3a36847f801f1ef'" "
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Max Kirkpatrick'" "'credit_id': '52fe44949251416c75039d4
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Grover'" "'credit_id': '52fe4592c3a368484e062d21'" "'ger
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Henri Fortin / Jean Valjean'" "'credit_id': '52fe4773c3a
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Lewis Farrell'" "'credit_id': '52fe4402c3a368484e00a6b7
##   ..$ : chr  "[{'cast_id': 1" "'character': 'David Leary'" "'credit_id': '52fe461a9251416c910493b9'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Colonel Hendricksson'" "'credit_id': '52fe44dcc3a36847f8
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Herself'" "'credit_id': '59d54df4c3a36845610206b5'" "'g
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Freddy Gale'" "'credit_id': '52fe4550c3a368484e05314d'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Annie Laird'" "'credit_id': '52fe4512c3a36847f80badfd'"
##   ..$ : chr  "[{'cast_id': 1001" "'character': 'Razieh'" "'credit_id': '52fe470cc3a36847f8120b0f'" "
##   ..$ : chr  "[{'cast_id': 3" "'character': \"Jimmy 'The Saint' Tosnia\"" "'credit_id': '52fe423ec3a3
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Antonia'" "'credit_id': '52fe4286c3a36847f8025d09'" "'g
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Poppa'" "'credit_id': '52fe4b889251416c7510604b'" "'gen
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Oona Hart'" "'credit_id': '52fe4d3c9251416c75134781'" "
##   ..$ : chr  "[{'cast_id': 1" "'character': 'William Adamson'" "'credit_id': '52fe44aac3a36847f80a32
##   ..$ : chr  "[{'cast_id': 1" "'character': \"Captain Christopher 'Skipper' Sheldon\"" "'credit_id':
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Robert Grant'" "'credit_id': '52fe47239251416c7508f091'
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Mike Donnelly'" "'credit_id': '52fe45c09251416c7506146f
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Gene Watson'" "'credit_id': '52fe4334c3a36847f80421dd'
##   ..$ : chr  "[{'cast_id': 3" "'character': 'August King'" "'credit_id': '52fe4657c3a368484e08b609'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Mary Reilly'" "'credit_id': '52fe44dbc3a36847f80ae251'"
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Maximillian | Pauly | Guido'" "'credit_id': '52fe44c191
##   ..$ : chr  "[{'cast_id': 10" "'character': \"Tommy 'Birdman' Rowland\"" "'credit_id': '52fe44e1c3a3
##   ..$ : chr  "[{'cast_id': 1" "'character': \"Maj. Vic 'Deak' Deakins\"" "'credit_id': '52fe44dcc3a3
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Henry'" "'credit_id': '52fe4583c3a36847f80cb93f'" "'gen
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Vinz'" "'credit_id': '52fe423fc3a36847f800f88f'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Jo'" "'credit_id': '52fe46c8c3a36847f8111c0b'" "'gender
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Narrator'" "'credit_id': '52fe469ac3a368484e099bcf'" "'g
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Mayor John Pappas'" "'credit_id': '52fe43ee9251416c7502
##   .. [list output truncated]
##  $ id      : int  862 8844 15602 31357 11862 949 11860 45325 9091 710 ...
##  $ director: chr  "John Lasseter" "Joe Johnston" "Howard Deutch" "Forest Whitaker" ...
```

```
# Print the number of entries with no cast
print(paste("Entries with no cast:", sum(is.na(credits$cast))))
```

If there are cells both missing cast and director columns, they should be dropped:

```
## [1] "Entries with no cast: 0"
```

```
# Print the number of entries with no directors
print(paste("Entries with no directors:", sum(is.na(credits$director))))
```

```
## [1] "Entries with no directors: 887"
```

```
# Print the number of entries missing both cast and directors
print(paste("Entries missing both:", sum(is.na(credits$cast) & is.na(credits$director))))
```

```
## [1] "Entries missing both: 0"
```

```
# Drop rows with both cast and directors missing
credits <- credits[!(is.na(credits$cast) & is.na(credits$director)), ]

# Check the structure of the dataframe after dropping rows
str(credits)
```

```
## 'data.frame':    45476 obs. of  3 variables:
##  $ cast    :List of 45476
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Woody (voice)'" "'credit_id': '52fe4284c3a36847f8024f95
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Alan Parrish'" "'credit_id': '52fe44bfc3a36847f80a7c73'
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Max Goldman'" "'credit_id': '52fe466a9251416c75077a8d'"
##   ..$ : chr  "[{'cast_id': 1" "'character': \"Savannah 'Vannah' Jackson\"" "'credit_id': '52fe447925
##   ..$ : chr  "[{'cast_id': 1" "'character': 'George Banks'" "'credit_id': '52fe44959251416c75039eb9'
##   ..$ : chr  "[{'cast_id': 25" "'character': 'Lt. Vincent Hanna'" "'credit_id': '52fe4292c3a36847f80
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Linus Larrabee'" "'credit_id': '52fe44959251416c75039d97
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Tom Sawyer'" "'credit_id': '52fe46bdc3a36847f810f771'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Darren Francis Thomas McCord'" "'credit_id': '52fe44dbc3
##   ..$ : chr  "[{'cast_id': 1" "'character': 'James Bond'" "'credit_id': '52fe426ec3a36847f801e10d'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Andrew Shepherd'" "'credit_id': '52fe44dac3a36847f80adf
##   ..$ : chr  "[{'cast_id': 9" "'character': 'Count Dracula'" "'credit_id': '52fe44b79251416c7503e811
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Balto (voice)'" "'credit_id': '52fe4409c3a368484e00bb65
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Richard Nixon'" "'credit_id': '52fe43c59251416c7501d6e1
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Morgan Adams'" "'credit_id': '52fe42f4c3a36847f802f65f'
##   ..$ : chr  "[{'cast_id': 4" "'character': \"Sam 'Ace' Rothstein\"" "'credit_id': '52fe424dc3a36847
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Marianne Dashwood'" "'credit_id': '52fe43cec3a36847f807
##   ..$ : chr  "[{'cast_id': 42" "'character': 'Ted the Bellhop'" "'credit_id': '52fe420dc3a36847f8000
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Ace Ventura'" "'credit_id': '52fe44dfc3a36847f80af279'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'John'" "'credit_id': '52fe44509251416c7503059b'" "'gende
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Chili Palmer'" "'credit_id': '52fe448dc3a36847f809c6eb'
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Helen Hudson'" "'credit_id': '52fe430ec3a36847f8037243'
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Robert Rath'" "'credit_id': '52fe451cc3a36847f80bd107'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Jessie Caldwell'" "'credit_id': '52fe45119251416c7504ab5
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Ben Sanderson'" "'credit_id': '52fe4245c3a36847f801117b
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Othello'" "'credit_id': '52fe46d29251416c75084b0b'" "'ge
##   ..$ : chr  "[{'cast_id': 9" "'character': 'Young Roberta Martin'" "'credit_id': '52fe44dec3a36847f8
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Anne Elliott'" "'credit_id': '52fe46ff9251416c7508a8a9'
##   ..$ : chr  "[{'cast_id': 24" "'character': 'One'" "'credit_id': '52fe428ac3a36847f8026cdf'" "'gende
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Xiao Jingbao'" "'credit_id': '52fe46559251416c910511c3'
##   ..$ : chr  "[{'cast_id': 13" "'character': 'Louanne Johnson'" "'credit_id': '52fe4546c3a36847f80c4e
##   ..$ : chr  "[{'cast_id': 41" "'character': 'James Cole'" "'credit_id': '52fe4212c3a36847f8001b31'"
```

```
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Henri Guillaumet'" "'credit_id': '52fe49b4c3a368484e13a
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Babe the Gallant Pig (voice)'" "'credit_id': '52fe450fc
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Dora Carrington'" "'credit_id': '52fe471bc3a36847f8123a
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Sister Helen Prejean'" "'credit_id': '52fe426ac3a36847f
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Tomas Minton'" "'credit_id': '5554bace9251416dd6002a69
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Diane Barrows'" "'credit_id': '52fe452f9251416c9102a483
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Cher Horowitz'" "'credit_id': '52fe4510c3a36847f80ba283
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Rev. Stephen Kumalo'" "'credit_id': '52fe456f9251416c91
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Richard III'" "'credit_id': '52fe44669251416c9100f591'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Anthony Curtis'" "'credit_id': '52fe44409251416c7502e19
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Robert Merivel'" "'credit_id': '52fe45999251416c91037fc
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Lord Rayden'" "'credit_id': '52fe44e5c3a36847f80b086f'"
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Suzanne Stone Maretto'" "'credit_id': '52fe4255c3a36847
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Finn Dodd'" "'credit_id': '52fe44959251416c75039e0b'" "
##   ..$ : chr  "[{'cast_id': 17" "'character': 'Detective David Mills'" "'credit_id': '52fe4279c3a3684
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Pocahontas (voice)'" "'credit_id': '52fe43819251416c750
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Camille Baker'" "'credit_id': '52fe44a4c3a36847f80a1e49
##   ..$ : chr  "[{'cast_id': 19" "'character': 'Michael McManus'" "'credit_id': '52fe4260c3a36847f8019
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Cynthia Mckay'" "'credit_id': '52fe4babc3a36847f820ea2b
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Lenny'" "'credit_id': '52fe44419251416c7502e523'" "'gen
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Gino'" "'credit_id': '52fe4787c3a36847f813acfb'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Sheriff Tom Palmer'" "'credit_id': '52fe4503c3a368484e0
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Sadie Flood'" "'credit_id': '52fe49de9251416c750d60cb'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Alex Cole'" "'credit_id': '52fe4aa2c3a368484e15ee3d'" "
##   ..$ : chr  "[{'cast_id': 10" "'character': 'Claudia Larson'" "'credit_id': '52fe44dbc3a36847f80ae0
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Pablo Neruda'" "'credit_id': '52fe43e29251416c75021dd1
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Pierre Lamontagne'" "'credit_id': '52fe4a259251416c750d
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Omri'" "'credit_id': '52fe44309251416c7502bdb5'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Karen McCann'" "'credit_id': '52fe470d9251416c7508c47d'
##   ..$ : chr  "[{'cast_id': 10" "'character': 'Glenn Holland'" "'credit_id': '52fe4330c3a36847f804115
##   ..$ : chr  "[{'cast_id': 18" "'character': 'Ashtray'" "'credit_id': '52fe43929251416c75015a3f'" "'
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Roz'" "'credit_id': '52fe47ef9251416c750aa5a7'" "'gende
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Bud Macintosh'" "'credit_id': '52fe4506c3a36847f80b7d1
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Dr. Benjamin Trace'" "'credit_id': '52fe44529251416c7503
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Gennaro Spirito'" "'credit_id': '52fe459ac3a36847f80d075
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Loli'" "'credit_id': '52fe43c5c3a36847f806e5bf'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Craig Jones'" "'credit_id': '52fe43999251416c75016ac3'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Seth Gecko'" "'credit_id': '52fe4271c3a36847f801f1ef'" "
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Max Kirkpatrick'" "'credit_id': '52fe44949251416c75039d4
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Grover'" "'credit_id': '52fe4592c3a368484e062d21'" "'ge
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Henri Fortin / Jean Valjean'" "'credit_id': '52fe4773c3
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Lewis Farrell'" "'credit_id': '52fe4402c3a368484e00a6b7
##   ..$ : chr  "[{'cast_id': 1" "'character': 'David Leary'" "'credit_id': '52fe461a9251416c910493b9'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Colonel Hendricksson'" "'credit_id': '52fe44dcc3a36847f
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Herself'" "'credit_id': '59d54df4c3a36845610206b5'" "'
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Freddy Gale'" "'credit_id': '52fe4550c3a368484e05314d'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Annie Laird'" "'credit_id': '52fe4512c3a36847f80badfd'"
##   ..$ : chr  "[{'cast_id': 1001" "'character': 'Razieh'" "'credit_id': '52fe470cc3a36847f8120b0f'" "
##   ..$ : chr  "[{'cast_id': 3" "'character': \"Jimmy 'The Saint' Tosnia\"" "'credit_id': '52fe423ec3a3
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Antonia'" "'credit_id': '52fe4286c3a36847f8025d09'" "'
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Poppa'" "'credit_id': '52fe4b889251416c7510604b'" "'gen
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Oona Hart'" "'credit_id': '52fe4d3c9251416c75134781'" "
##   ..$ : chr  "[{'cast_id': 1" "'character': 'William Adamson'" "'credit_id': '52fe44aac3a36847f80a32
##   ..$ : chr  "[{'cast_id': 1" "'character': \"Captain Christopher 'Skipper' Sheldon\"" "'credit_id':
```

```
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Robert Grant'" "'credit_id': '52fe47239251416c7508f091'
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Mike Donnelly'" "'credit_id': '52fe45c09251416c7506146f
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Gene Watson'" "'credit_id': '52fe4334c3a36847f80421dd'
##   ..$ : chr  "[{'cast_id': 3" "'character': 'August King'" "'credit_id': '52fe4657c3a368484e08b609'"
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Mary Reilly'" "'credit_id': '52fe44dbc3a36847f80ae251'"
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Maximillian | Pauly | Guido'" "'credit_id': '52fe44c19
##   ..$ : chr  "[{'cast_id': 10" "'character': \"Tommy 'Birdman' Rowland\"" "'credit_id': '52fe44e1c3a3
##   ..$ : chr  "[{'cast_id': 1" "'character': \"Maj. Vic 'Deak' Deakins\"" "'credit_id': '52fe44dcc3a3
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Henry'" "'credit_id': '52fe4583c3a36847f80cb93f'" "'gen
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Vinz'" "'credit_id': '52fe423fc3a36847f800f88f'" "'gend
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Jo'" "'credit_id': '52fe46c8c3a36847f8111c0b'" "'gender
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Narrator'" "'credit_id': '52fe469ac3a368484e099bcf'" "'
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Mayor John Pappas'" "'credit_id': '52fe43ee9251416c75023
##   .. [list output truncated]
##  $ id      : int  862 8844 15602 31357 11862 949 11860 45325 9091 710 ...
##  $ director: chr  "John Lasseter" "Joe Johnston" "Howard Deutch" "Forest Whitaker" ...
```

In this instance there is 0 rows that needed to be dropped now I am ready to merge this dataset into my main dataset.

The *id* of the merged dataset which consists of the movies and keywords and *id* of credits columns point to the same movies, this means I can merge the credits dataset into the main dataset with this information.

```
# I will do this merge with a left join making the credits joining the merged datset I made earlier to
final_movies_data <- merge(merged_df, credits, by = "id", all.x = TRUE)
```

```
dim(final_movies_data)
```

A brief overview of the main dataframe is below

```
## [1] 46208    21
```

```
str(final_movies_data)
```

```
## 'data.frame':    46208 obs. of  21 variables:
##  $ id                    : int  2 3 5 6 11 12 13 14 15 16 ...
##  $ belongs_to_collection: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ budget                : num  NA NA 0.0105 NA 0.0289 ...
##  $ genres                :List of 46208
##   ..$ : chr "Drama; Crime"
##   ..$ : chr "Drama; Comedy"
##   ..$ : chr "Crime; Comedy"
##   ..$ : chr "Action; Thriller; Crime"
##   ..$ : chr "Adventure; Action; Science Fiction"
##   ..$ : chr "Animation; Family"
##   ..$ : chr "Comedy; Drama; Romance"
```

```
##   ..$ : chr "Drama"
##   ..$ : chr "Mystery; Drama"
##   ..$ : chr "Drama; Crime; Music"
##   ..$ : chr "Horror; Thriller; Mystery"
##   ..$ : chr "Adventure; Fantasy; Action; Thriller; Science Fiction"
##   ..$ : chr "Drama; Science Fiction"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Documentary"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Action; Crime"
##   ..$ : chr "Drama; War"
##   ..$ : chr "Drama"
##   ..$ : chr "Drama; Music; Romance"
##   ..$ : chr "Drama; War"
##   ..$ : chr "Animation; Science Fiction"
##   ..$ : chr "Western"
##   ..$ : chr "Animation; Comedy; Family"
##   ..$ : chr "Science Fiction; Drama; Romance"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Drama; Thriller; Crime"
##   ..$ : chr "Science Fiction; Mystery; Adventure"
##   ..$ : chr "Science Fiction; Thriller; Mystery"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Crime; Drama; Thriller"
##   ..$ : chr "Thriller; Crime; Drama"
##   ..$ : chr "Comedy; Science Fiction"
##   ..$ : chr "Drama; Music; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Drama; Comedy; Music"
##   ..$ : chr "Drama"
##   ..$ : chr "Adventure; Thriller; Science Fiction"
##   ..$ : chr "Comedy; Fantasy; Science Fiction"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Mystery; Thriller"
##   ..$ : chr "Science Fiction; Drama; Thriller"
##   ..$ : chr "Drama; Adventure; Action; History"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Adventure; Animation; Fantasy"
##   ..$ : chr "Action; Adventure; Crime; Thriller"
##   ..$ : chr "Drama; Thriller"
##   ..$ : chr "Adventure; Action"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Adventure; Action"
##   ..$ : chr "Drama; Music; Romance"
##   ..$ : chr "Adventure; Action"
##   ..$ : chr "Action; Comedy; Crime"
##   ..$ : chr "Documentary"
##   ..$ : chr "Documentary"
##   ..$ : chr "Crime; Drama; Mystery; Thriller"
##   ..$ : chr "Action; Thriller; Science Fiction; Adventure"
##   ..$ : chr "Action; Comedy; Crime"
##   ..$ : chr "Science Fiction; Action; Adventure"
```

```
##   ..$ : chr "Action; Drama; Adventure"
##   ..$ : chr "Comedy; Drama"
##   ..$ : chr "Comedy; Crime"
##   ..$ : chr "Thriller; Crime; Drama"
##   ..$ : chr "Drama; Romance"
##   ..$ : chr "Crime; Drama"
##   ..$ : chr "Action; Drama; Thriller"
##   ..$ : chr "Adventure; Comedy; Science Fiction; Family"
##   ..$ : chr "Science Fiction; Action; Adventure; Thriller"
##   ..$ : chr "Thriller; Crime"
##   ..$ : chr "Drama; Music; Mystery"
##   ..$ : chr "Comedy; Drama; Mystery"
##   ..$ : chr "Drama; Mystery; Romance"
##   ..$ : chr "Action; Crime; Drama; Thriller"
##   ..$ : chr "Comedy; Drama; Romance"
##   ..$ : chr "Drama"
##   ..$ : chr "Romance; Comedy"
##   ..$ : chr "Comedy; Crime"
##   ..$ : chr "Drama; Thriller; Crime; Romance"
##   ..$ : chr "Crime; Drama; History; Thriller"
##   ..$ : chr "Adventure; Comedy; Family; Fantasy"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Adventure; Fantasy; Action"
##   ..$ : chr "Fantasy; Drama; Animation; Adventure"
##   ..$ : chr "Drama; Fantasy"
##   ..$ : chr "Drama"
##   ..$ : chr "Adventure; Fantasy; Animation"
##   ..$ : chr "Fantasy; Adventure; Animation; Family"
##   ..$ : chr "Documentary; Music"
##   ..$ : chr "Documentary"
##   ..$ : chr "Action; Adventure; Comedy"
##   ..$ : chr "Documentary; Music"
##   ..$ : chr "Drama; Horror"
##   ..$ : chr "Romance; Fantasy; Drama; Comedy"
##   ..$ : chr "Horror"
##   ..$ : chr "Romance; Comedy; Drama"
##   ..$ : chr "Crime; Drama; Romance; Thriller"
##   .. [list output truncated]
##  $ original_language    : chr  "fi" "fi" "en" "en" ...
##  $ popularity           : num  3.86 2.29 9.03 5.54 42.15 ...
##  $ production_companies : chr  "Villealfa Filmproduction Oy; Finnish Film Foundation" "Villealfa Filn
##  $ production_countries :List of 46208
##   ..$ : chr "Finland"
##   ..$ : chr "Finland"
##   ..$ : chr "United States of America"
##   ..$ : chr "Japan; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Argentina; Denmark; Finland; France; Germany; Iceland; Italy; Netherlands; Norway; Swed
##   ..$ : chr "Germany; United Kingdom"
```

```
##   ..$ : chr "France"
##   ..$ : chr "Germany"
##   ..$ : chr "Canada; Spain"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Germany; United States of America"
##   ..$ : chr "Israel; Sweden"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "Japan"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Mexico"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America; Canada; Germany"
##   ..$ : chr "United States of America; United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "Spain"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Palestinian Territory; France; Germany; Israel; Netherlands"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "Germany; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "France; United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Austria; Switzerland; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America; Hong Kong; United Kingdom"
##   ..$ : chr "China"
##   ..$ : chr "United States of America"
##   ..$ : chr "Japan"
##   ..$ : chr "Uruguay; United States of America; Germany; Paraguay"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Germany; Ireland"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Spain"
##   ..$ : chr "Austria"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "Spain; France"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "France; United States of America"
```

```
##   ..$ : chr "Denmark"
##   ..$ : chr "United States of America"
##   ..$ : chr "Germany"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "France; Poland"
##   ..$ : chr "France; Poland"
##   ..$ : chr "France; Poland; Switzerland"
##   ..$ : chr "United States of America"
##   ..$ : chr "Denmark; Sweden"
##   ..$ : chr "South Korea; Germany"
##   ..$ : chr "United States of America"
##   ..$ : chr "United Kingdom; United States of America"
##   ..$ : chr "Ireland; Luxembourg; Russia; United Kingdom; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Australia; United Kingdom; United States of America"
##   ..$ : chr "New Zealand; United States of America"
##   ..$ : chr "New Zealand; United States of America"
##   ..$ : chr "New Zealand; United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Poland"
##   ..$ : chr "Poland"
##   ..$ : chr "Japan"
##   ..$ : chr "Japan"
##   ..$ : chr "United States of America"
##   ..$ : chr ""
##   ..$ : chr "France; United Kingdom; United States of America"
##   ..$ : chr "United Kingdom"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "United States of America"
##   ..$ : chr "Denmark; Sweden"
##   ..$ : chr "Spain"
##   .. [list output truncated]
##  $ release_date         : Date, format: "1988-10-21" "1986-10-16" ...
##  $ revenue              : num  NA NA 0.0113 0.0319 2.0406 ...
##  $ runtime              : num  69 76 98 110 121 100 142 122 119 140 ...
##  $ status               : chr  "Released" "Released" "Released" "Released" ...
##  $ title                : chr  "Ariel" "Shadows in Paradise" "Four Rooms" "Judgment Night" ...
##  $ vote_average         : num  7.1 7.1 6.5 6.4 8.1 7.6 8.2 7.9 8 7.7 ...
##  $ vote_count           : int  44 35 539 79 6778 6292 8147 3438 1244 392 ...
##  $ release_year         : chr  "1988" "1986" "1995" "1993" ...
##  $ profit               : num  NA NA 0.000789 NA 2.011627 ...
##  $ weighted_rating      : num  6.29 6.19 6.42 6.05 8.09 ...
##  $ keywords             : chr  "underdog; prison; factory worker; prisoner; helsinki; independent fil
##  $ cast                 :List of 46208
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Taisto Olavi Kasurinen'" "'credit_id': '52fe420dc3a3684
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Nikander'" "'credit_id': '52fe420dc3a36847f8000087'" "'
##   ..$ : chr  "[{'cast_id': 42" "'character': 'Ted the Bellhop'" "'credit_id': '52fe420dc3a36847f8000
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Frank Wyatt'" "'credit_id': '52fe420dc3a36847f8000259'"
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Luke Skywalker'" "'credit_id': '52fe420dc3a36847f800044
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Marlin (voice)'" "'credit_id': '52fe420ec3a36847f800067
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Forrest Gump'" "'credit_id': '52fe420ec3a36847f800074f'
```

```
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Lester Burnham'" "'credit_id': '52fe420ec3a36847f80007d
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Charles Foster Kane'" "'credit_id': '52fe420ec3a36847f8
##   ..$ : chr  "[{'cast_id': 33" "'character': 'Selma Jezkova'" "'credit_id': '52fe420fc3a36847f8000a1
##   ..$ : chr  "[{'cast_id': 4" "'character': 'James'" "'credit_id': '52fe420fc3a36847f8000a8f'" "'gen
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Korben Dallas'" "'credit_id': '52fe420fc3a36847f8000b53
##   ..$ : chr  "[{'cast_id': 10" "'character': 'Maria'" "'credit_id': '52fe420fc3a36847f8000c87'" "'ge
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Ann'" "'credit_id': '52fe420fc3a36847f8000d33'" "'gende
##   ..$ : chr  "[{'cast_id': 13" "'character': 'Himself'" "'credit_id': '55a58678c3a36812f1000041'" "'
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Captain Jack Sparrow'" "'credit_id': '52fe420fc3a36847
##   ..$ : chr  "[{'cast_id': 3" "'character': \"Beatrix 'The Bride' Kiddo\"" "'credit_id': '52fe4210c3
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Staff Sgt. Sykes'" "'credit_id': '52fe4210c3a36847f8001
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Eyal'" "'credit_id': '52fe4210c3a36847f80011cd'" "'gend
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Matt'" "'credit_id': '52fe4210c3a36847f800122d'" "'gend
##   ..$ : chr  "[{'cast_id': 29" "'character': 'Captain Benjamin L. Willard'" "'credit_id': '52fe4210c3
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Heinz'" "'credit_id': '52fe4211c3a36847f80013c7'" "'gen
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Bill Munny'" "'credit_id': '52fe4211c3a36847f8001447'" 
##   ..$ : chr  "[{'cast_id': 2" "'character': \"Homer / Itchy / Barney / Grampa / Stage Manager / Krus
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Joel Barish'" "'credit_id': '52fe4211c3a36847f800167f'" 
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Octavio'" "'credit_id': '52fe4211c3a36847f8001741'" "'g
##   ..$ : chr  "[{'cast_id': 37" "'character': 'Captain Jack Sparrow'" "'credit_id': '52fe4212c3a36847
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Tom Stall'" "'credit_id': '52fe4212c3a36847f8001927'" "
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Dr. Dave Bowman'" "'credit_id': '52fe4212c3a36847f8001a
##   ..$ : chr  "[{'cast_id': 41" "'character': 'James Cole'" "'credit_id': '52fe4212c3a36847f8001b31'" 
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Alicia'" "'credit_id': '52fe4212c3a36847f8001ba9'" "'ge
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Jimmy B. \"Rabbit\" Smith'" "'credit_id': '52fe4212c3a3
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Luther Whitney'" "'credit_id': '52fe4212c3a36847f8001ce
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Khaled'" "'credit_id': '52fe4213c3a36847f8001dbd'" "'ge
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Sam Lowry'" "'credit_id': '52fe4213c3a36847f8001e8f'" "
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Johnny Cash'" "'credit_id': '52fe4213c3a36847f8001f43'" 
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Frankie Dunn'" "'credit_id': '52fe4213c3a36847f8002015'
##   ..$ : chr  "[{'cast_id': 16" "'character': 'Billy Elliot'" "'credit_id': '52fe4213c3a36847f800211d
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Derek Vinyard'" "'credit_id': '52fe4213c3a36847f8002187
##   ..$ : chr  "[{'cast_id': 13" "'character': 'Ray Ferrier'" "'credit_id': '52fe4213c3a36847f800226b'
##   ..$ : chr  "[{'cast_id': 6" "'character': 'President James Dale / Art Land'" "'credit_id': '52fe42
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Jesse'" "'credit_id': '52fe4214c3a36847f80023e5'" "'gen
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Leonard'" "'credit_id': '52fe4214c3a36847f80024db'" "'g
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Rick Deckard'" "'credit_id': '52fe4214c3a36847f800259f'
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Nameless'" "'credit_id': '52fe4214c3a36847f8002697'" "'
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Jesse'" "'credit_id': '52fe4214c3a36847f800276b'" "'gen
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Nausicaä (voice)'" "'credit_id': '52fe4215c3a36847f8002
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Detective Ricardo \"Rico\" Tubbs'" "'credit_id': '52fe4
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Susan Watkins'" "'credit_id': '52fe4215c3a36847f800299b
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Indy'" "'credit_id': '52fe4215c3a36847f8002a05'" "'gend
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Michael Djerzinski'" "'credit_id': '52fe4215c3a36847f80
##   ..$ : chr  "[{'cast_id': 4" "'character': 'Indiana Jones'" "'credit_id': '52fe4215c3a36847f8002c09
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Baby (Frances) Houseman'" "'credit_id': '52fe4215c3a368
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Indiana Jones'" "'credit_id': '52fe4216c3a36847f8002e1d
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Det. Axel Foley'" "'credit_id': '52fe4216c3a36847f8002f
##   ..$ : chr  "[{'cast_id': 8" "'character': '(voice)'" "'credit_id': '52fe4216c3a36847f8002fb9'" "'g
##   ..$ : chr  "[{'cast_id': 6" "'character': ''" "'credit_id': '52fe4216c3a36847f8002ff9'" "'gender':
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Paul Biegler'" "'credit_id': '52fe4216c3a36847f8003055'
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Harry S. Stamper'" "'credit_id': '52fe4216c3a36847f8003
##   ..$ : chr  "[{'cast_id': 17" "'character': 'Det. Axel Foley'" "'credit_id': '52fe4217c3a36847f8003
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Kevin Flynn/Clu'" "'credit_id': '52fe4217c3a36847f80033
```

```
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Maximus'" "'credit_id': '52fe4217c3a36847f8003435'" "'g
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Manuela'" "'credit_id': '52fe4217c3a36847f80034ff'" "'g
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Tom'" "'credit_id': '52fe4217c3a36847f80035fb'" "'gend
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Léon Montana'" "'credit_id': '52fe4217c3a36847f80036b3
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Cecilie'" "'credit_id': '52fe4217c3a36847f8003717'" "'g
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Travis Bickle'" "'credit_id': '52fe4218c3a36847f80037d7
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Lola'" "'credit_id': '52fe4218c3a36847f80038df'" "'gen
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Marty McFly'" "'credit_id': '52fe4218c3a36847f8003a13'
##   ..$ : chr  "[{'cast_id': 13" "'character': \"Major Alan 'Dutch' Schaeffer\"" "'credit_id': '52fe42
##   ..$ : chr  "[{'cast_id': 19" "'character': 'Franky Four Fingers'" "'credit_id': '52fe4219c3a36847f8
##   ..$ : chr  "[{'cast_id': 10" "'character': 'Julie Vignon'" "'credit_id': '52fe4219c3a36847f8003c99
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Karol Karol'" "'credit_id': '52fe4219c3a36847f8003d17'"
##   ..$ : chr  "[{'cast_id': 9" "'character': 'Valentine Dussaut'" "'credit_id': '52fe4219c3a36847f8003
##   ..$ : chr  "[{'cast_id': 9" "'character': 'Tony Montana'" "'credit_id': '52fe4219c3a36847f8003e47'
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Jørgen Mortensen'" "'credit_id': '52fe4219c3a36847f8003
##   ..$ : chr  "[{'cast_id': 9" "'character': 'Adult Monk'" "'credit_id': '52fe4219c3a36847f8003fbd'" "
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Vivian Ward'" "'credit_id': '52fe4219c3a36847f8004051'"
##   ..$ : chr  "[{'cast_id': 12" "'character': 'The Dude'" "'credit_id': '52fe421ac3a36847f8004131'" "
##   ..$ : chr  "[{'cast_id': 16" "'character': 'Chris Wilton'" "'credit_id': '530633d692514134912c121e
##   ..$ : chr  "[{'cast_id': 14" "'character': 'Eliot Ness'" "'credit_id': '52fe421ac3a36847f800427d'"
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Willy Wonka'" "'credit_id': '52fe421ac3a36847f80042ff'"
##   ..$ : chr  "[{'cast_id': 28" "'character': 'Frodo Baggins'" "'credit_id': '52fe421ac3a36847f800448
##   ..$ : chr  "[{'cast_id': 13" "'character': 'Frodo Baggins'" "'credit_id': '52fe421ac3a36847f8004589
##   ..$ : chr  "[{'cast_id': 12" "'character': 'Frodo Baggins'" "'credit_id': '52fe421bc3a36847f80046f
##   ..$ : chr  "[{'cast_id': 1" "'character': 'Frodo (voice)'" "'credit_id': '52fe421bc3a36847f80047df
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Ursula Zyro'" "'credit_id': '52fe421bc3a36847f8004887'"
##   ..$ : chr  "[{'cast_id': 8" "'character': 'Adam'" "'credit_id': '52fe421bc3a36847f800492b'" "'gend
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Ashitaka (voice)'" "'credit_id': '52fe421bc3a36847f8004
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Chihiro Ogino (voice)'" "'credit_id': '52fe421bc3a36847
##   ..$ : chr  "[{'cast_id': 15" "'character': 'Himself (as The Rolling Stones)'" "'credit_id': '52fe4
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Himself'" "'credit_id': '52fe421cc3a36847f8004c65'" "'g
##   ..$ : chr  "[{'cast_id': 3" "'character': 'Ulysses Everett McGill'" "'credit_id': '52fe421cc3a3684
##   ..$ : chr  "[{'cast_id': 2" "'character': 'Himself'" "'credit_id': '52fe421cc3a36847f8004d8d'" "'g
##   ..$ : chr  "[{'cast_id': 25" "'character': 'Hans'" "'credit_id': '52fe421cc3a36847f8004e73'" "'gen
##   ..$ : chr  "[{'cast_id': 11" "'character': 'Phil Connors'" "'credit_id': '52fe421dc3a36847f8004f01
##   ..$ : chr  "[{'cast_id': 5" "'character': 'Dracula'" "'credit_id': '52fe421dc3a36847f8004fcd'" "'g
##   ..$ : chr  "[{'cast_id': 6" "'character': 'Kresten'" "'credit_id': '52fe421dc3a36847f800507b'" "'g
##   ..$ : chr  "[{'cast_id': 7" "'character': 'Ángel/Juan/Zahara'" "'credit_id': '52fe421dc3a36847f8005
##   .. [list output truncated]
##  $ director             : chr  "Aki Kaurismäki" "Aki Kaurismäki" "Allison Anders" "Stephen Hopkins"
```

```
head(final_movies_data)
```

```
##   id belongs_to_collection     budget                            genres
## 1  2                     1         NA                      Drama; Crime
## 2  3                     1         NA                     Drama; Comedy
## 3  5                     1 0.01052659                     Crime; Comedy
## 4  6                     1         NA           Action; Thriller; Crime
## 5 11                     1 0.02894813 Adventure; Action; Science Fiction
## 6 12                     1 0.24737493                  Animation; Family
##   original_language popularity
## 1                fi   3.860491
## 2                fi   2.292110
## 3                en   9.026586
```

```
## 4                en   5.538671
## 5                en  42.149697
## 6                en  25.497794
##                                                production_companies
## 1                Villealfa Filmproduction Oy; Finnish Film Foundation
## 2                                      Villealfa Filmproduction Oy
## 3                                      Miramax Films; A Band Apart
## 4 Universal Pictures; Largo Entertainment; JVC Entertainment Networks
## 5                   Lucasfilm; Twentieth Century Fox Film Corporation
## 6                                          Pixar Animation Studios
##              production_countries release_date    revenue runtime   status
## 1                         Finland   1988-10-21         NA      69 Released
## 2                         Finland   1986-10-16         NA      76 Released
## 3        United States of America   1995-12-09 0.01131609      98 Released
## 4 Japan; United States of America   1993-10-15 0.03194015     110 Released
## 5        United States of America   1977-05-25 2.04057477     121 Released
## 6        United States of America   2003-05-30 2.47463232     100 Released
##                 title vote_average vote_count release_year       profit
## 1               Ariel          7.1         44         1988           NA
## 2 Shadows in Paradise          7.1         35         1986           NA
## 3          Four Rooms          6.5        539         1995 0.0007894945
## 4      Judgment Night          6.4         79         1993           NA
## 5           Star Wars          8.1       6778         1977 2.0116266402
## 6        Finding Nemo          7.6       6292         2003 2.2272573910
##   weighted_rating
## 1        6.294764
## 2        6.192691
## 3        6.424489
## 4        6.052879
## 5        8.085410
## 6        7.587055
##
## 1
## 2
## 3
## 4
## 5                                android; galaxy; hermit; death star; lightsaber; jedi; rescue missi
## 6 father son relationship; harbor; underwater; fish tank; great barrier reef; missing child; aftercr
##
## 1
## 2
## 3
## 4
## 5 [{'cast_id': 3, 'character': 'Luke Skywalker', 'credit_id': '52fe420dc3a36847f8000441', 'gender': 1
## 6
##          director
## 1  Aki Kaurismäki
## 2  Aki Kaurismäki
## 3  Allison Anders
## 4 Stephen Hopkins
## 5    George Lucas
## 6  Andrew Stanton
```

Intial Data Anaylsis on Dataset

First of all, let's list top 10 movies regarding *weighted_rating*, *popularity* and *profit*

```
# Creating new table named top_rated_movies that will show the top 10 based on weighted rating in Desce
top_rated_movies <- final_movies_data %>%
  arrange(desc(weighted_rating)) %>%
  select(title, director, genres, profit, popularity, weighted_rating) %>%
  head(10)

print(top_rated_movies)
```

```
##                          title             director
## 1  Dilwale Dulhania Le Jayenge        Aditya Chopra
## 2     The Shawshank Redemption       Frank Darabont
## 3                The Godfather Francis Ford Coppola
## 4                    Your Name.       Makoto Shinkai
## 5              The Dark Knight    Christopher Nolan
## 6                   Fight Club        David Fincher
## 7                 Pulp Fiction    Quentin Tarantino
## 8             Schindler's List     Steven Spielberg
## 9                    Whiplash      Damien Chazelle
## 10               Spirited Away       Hayao Miyazaki
##                                  genres      profit popularity weighted_rating
## 1                 Comedy; Drama; Romance 0.228427064   34.45702        8.907146
## 2                           Drama; Crime 0.008793571   51.64540        8.486488
## 3                           Drama; Crime 0.629138690   41.10926        8.481284
## 4              Romance; Animation; Drama          NA   34.46125        8.393603
## 5         Drama; Action; Crime; Thriller 2.156789505  123.16726        8.291352
## 6                                  Drama 0.099617761   63.86960        8.289045
## 7                       Thriller; Crime 0.541932056  140.95024        8.287776
## 8                   Drama; History; War 0.787824856   41.72512        8.276203
## 9                                  Drama 0.025769099   64.29999        8.275880
## 10 Fantasy; Adventure; Animation; Family 0.684031409   41.04887        8.273422
```

```
# Creating new table named top_popular_movies that will show the top 10 based on popularity in Descendi
top_popular_movies <- final_movies_data %>%
  arrange(desc(popularity)) %>%
  select(title, director, genres, profit, popularity, weighted_rating) %>%
  head(10)

print(top_popular_movies)
```

```
##                           title       director
## 1                       Minions     Kyle Balda
## 2                  Wonder Woman  Patty Jenkins
## 3           Beauty and the Beast    Bill Condon
## 4                   Baby Driver   Edgar Wright
## 5                   Big Hero 6 Chris Williams
## 6                      Deadpool     Tim Miller
```

```
## 7  Guardians of the Galaxy Vol. 2     James Gunn
## 8                            Avatar  James Cameron
## 9                        John Wick Chad Stahelski
## 10                       Gone Girl  David Fincher
##                                          genres    profit popularity
## 1          Family; Animation; Adventure; Comedy 2.8493670   547.4883
## 2                     Action; Adventure; Fantasy 1.7673635   294.3370
## 3                       Family; Fantasy; Romance 2.9024088   287.2537
## 4                                  Action; Crime 0.5013588   228.0327
## 5  Adventure; Family; Animation; Action; Comedy 1.2818902   213.8499
## 6                     Action; Adventure; Comedy 1.9082423   187.8605
## 7    Action; Adventure; Comedy; Science Fiction 1.7458779   185.3310
## 8   Action; Adventure; Fantasy; Science Fiction 6.7132427   185.0709
## 9                               Action; Thriller 0.1809565   183.8704
## 10                      Mystery; Thriller; Drama 0.8114170   154.8010
##    weighted_rating
## 1         6.391622
## 2         7.186580
## 3         6.790313
## 4         7.167948
## 5         7.785942
## 6         7.393474
## 7         7.583261
## 8         7.194410
## 9         6.988994
## 10        7.884747
```

```r
# Creating new table named top_profitable_movies that will show the top 10 based on profit in Descendin


top_profitable_movies <- final_movies_data %>%
  arrange(desc(profit)) %>%
  select(title, director, genres, profit, popularity, weighted_rating) %>%
  head(10)

print(top_profitable_movies)
```

```
##                                         title        director
## 1                                      Avatar   James Cameron
## 2                  Star Wars: The Force Awakens     J.J. Abrams
## 3                                     Titanic   James Cameron
## 4                                Jurassic World Colin Trevorrow
## 5                                    Furious 7       James Wan
## 6                                  The Avengers     Joss Whedon
## 7  Harry Potter and the Deathly Hallows: Part 2     David Yates
## 8                       Avengers: Age of Ultron     Joss Whedon
## 9                                        Frozen      Chris Buck
## 10                         Beauty and the Beast     Bill Condon
##                                          genres   profit popularity
## 1   Action; Adventure; Fantasy; Science Fiction 6.713243  185.07089
## 2   Action; Adventure; Science Fiction; Fantasy 4.798083   31.62601
## 3                       Drama; Romance; Thriller 4.329151   26.88907
## 4  Action; Adventure; Science Fiction; Thriller 3.588328   32.79048
## 5                                        Action 3.463905   27.27569
```

```
## 6            Science Fiction; Action; Adventure 3.419979   89.88765
## 7                    Family; Fantasy; Adventure 3.202716   24.99074
## 8            Action; Adventure; Science Fiction 2.961667   37.37942
## 9                  Animation; Adventure; Family 2.958549   24.24824
## 10                    Family; Fantasy; Romance 2.902409  287.25365
##    weighted_rating
## 1         7.194410
## 2         7.490233
## 3         7.489954
## 4         6.495109
## 5         7.283348
## 6         7.393775
## 7         7.885038
## 8         7.289715
## 9         7.286958
## 10        6.790313
```

These can now be used for further visulisations later on in my analysis either in R or with a BI tool

Numerical Analysis on the data

Time to view and draw potential correlations within columns from my data

```
# Dropping the 'id' column to make sure this doesn't skew the result of the correlation
movies_no_id <- final_movies_data %>% select(-id)

# create a table that contains only numeric columns
final_movies_numeric <- movies_no_id %>% select(where(is.numeric))

# Calculate the correlation matrix
correlation_matrix <- cor(final_movies_numeric, use = "complete.obs")
```

Creating a correlation matrix

```
## Warning in cor(final_movies_numeric, use = "complete.obs"): the standard
## deviation is zero
```

```
# View the correlation matrix
print(correlation_matrix)
```

```
##                       belongs_to_collection       budget popularity   revenue
## belongs_to_collection                     1           NA         NA        NA
## budget                                   NA  1.000000000 0.30866098 0.7294058
## popularity                               NA  0.308660977 1.00000000 0.4408875
## revenue                                  NA  0.729405773 0.44088754 1.0000000
## runtime                                  NA  0.189965186 0.08670614 0.1872102
## vote_average                             NA -0.007420028 0.16107316 0.1678527
## vote_count                               NA  0.586031980 0.48042108 0.7706898
## profit                                   NA  0.580727036 0.43600154 0.9804951
```

```
## weighted_rating                         NA  0.090115204 0.24672421 0.2751546
##                          runtime vote_average vote_count    profit
## belongs_to_collection         NA           NA         NA        NA
## budget                0.18996519 -0.007420028  0.5860320 0.5807270
## popularity            0.08670614  0.161073157  0.4804211 0.4360015
## revenue               0.18721025  0.167852695  0.7706898 0.9804951
## runtime               1.00000000  0.315635996  0.1991221 0.1682126
## vote_average          0.31563600  1.000000000  0.3182842 0.2018868
## vote_count            0.19912209  0.318284160  1.0000000 0.7487947
## profit                0.16821265  0.201886764  0.7487947 1.0000000
## weighted_rating       0.31759743  0.894231921  0.4660172 0.3015597
##                       weighted_rating
## belongs_to_collection              NA
## budget                       0.0901152
## popularity                   0.2467242
## revenue                      0.2751546
## runtime                      0.3175974
## vote_average                 0.8942319
## vote_count                   0.4660172
## profit                       0.3015597
## weighted_rating              1.0000000
```

From viewing the correlation matrix I have noticed that the NA/Null values have also transferred over this will cause complications as these values won't be recognised as a numerical value so I will have to change this.

```
# Replace NA/NaN values with 0 in the correlation matrix
correlation_matrix[is.na(correlation_matrix)] <- 0
correlation_matrix[is.nan(correlation_matrix)] <- 0

# Confirm the updated correlation matrix
print(correlation_matrix)
```

```
##                       belongs_to_collection       budget popularity   revenue
## belongs_to_collection                     1  0.000000000 0.00000000 0.0000000
## budget                                    0  1.000000000 0.30866098 0.7294058
## popularity                                0  0.308660977 1.00000000 0.4408875
## revenue                                   0  0.729405773 0.44088754 1.0000000
## runtime                                   0  0.189965186 0.08670614 0.1872102
## vote_average                              0 -0.007420028 0.16107316 0.1678527
## vote_count                                0  0.586031980 0.48042108 0.7706898
## profit                                    0  0.580727036 0.43600154 0.9804951
## weighted_rating                           0  0.090115204 0.24672421 0.2751546
##                          runtime vote_average vote_count    profit
## belongs_to_collection 0.00000000  0.000000000  0.0000000 0.0000000
## budget                0.18996519 -0.007420028  0.5860320 0.5807270
## popularity            0.08670614  0.161073157  0.4804211 0.4360015
## revenue               0.18721025  0.167852695  0.7706898 0.9804951
## runtime               1.00000000  0.315635996  0.1991221 0.1682126
## vote_average          0.31563600  1.000000000  0.3182842 0.2018868
## vote_count            0.19912209  0.318284160  1.0000000 0.7487947
## profit                0.16821265  0.201886764  0.7487947 1.0000000
## weighted_rating       0.31759743  0.894231921  0.4660172 0.3015597
```

```
##                         weighted_rating
## belongs_to_collection         0.0000000
## budget                        0.0901152
## popularity                    0.2467242
## revenue                       0.2751546
## runtime                       0.3175974
## vote_average                  0.8942319
## vote_count                    0.4660172
## profit                        0.3015597
## weighted_rating               1.0000000
```

```r
# Visualize using corrplot and viridis
corrplot(correlation_matrix, method = "color", col = viridis(200),
         type = "upper", order = "hclust",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col = "black", tl.srt = 45, # Text label color and rotation
         diag = FALSE # Hide the diagonal
)
```

Drawing visualisation from the matrix

Conclusions from the graph As we can see, there is a strong correlation (value > 0.7) between these parameters: - 0.73, budget and revenue - 0.78, vote_count and revenue - 0.75, profit and vote_count - 0.98, profit and revenue

What can we draw from this?

The more *revenue* a movie has, the more *profit* the movie will have and this result is expected. However, other conclusions can be reached, too: - If a movie has a higher *budget*, it is excpected to also have higher *revenue*. - The more *number of votes* a movie has, the more *revenue* and therefore *profit* the movie has.

This seems logical, because the amount of votes also indicates the *popularity* of a movie and popular ones tend to have more *revenue*. However, the relationship between *popularity* and *vote_count* or *profit / revenue* is not so strong. This result is a surprise. However, we can still say that there is a moderate correlation between: - 0.46, popularity and revenue - 0.56, popularity and vote_count - 0.44, popularity and profit - 0.61, budget and vote_count

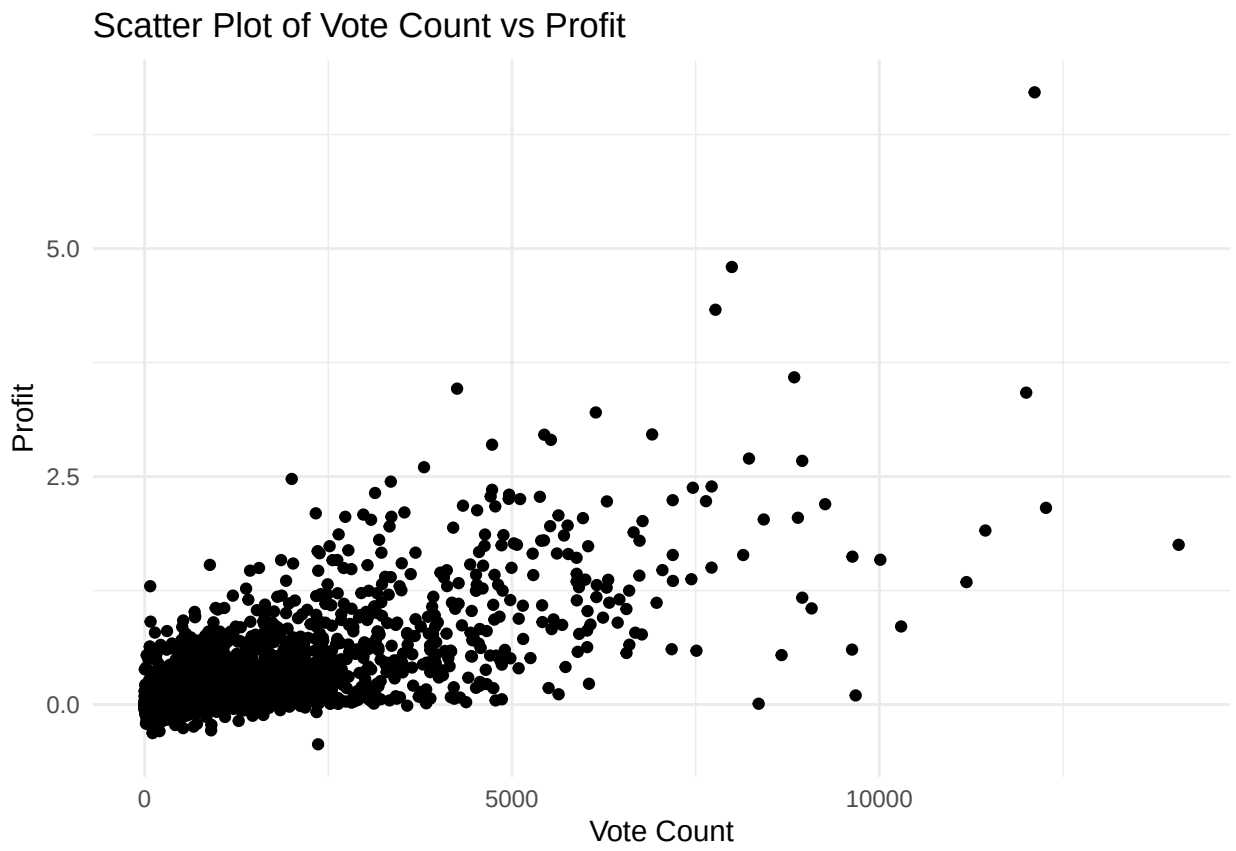
The most surprising result was having almost no correlation between *vote_average* and any other parameter except *weighted_rating*. Because it seems logical that higher voted movies tends to have more popularity and revenue, but this is not the case. On the other hand, after some processing of *vote_average* in order to create *weighted_rating*, some moderate correlations between *weighted_rating* and other parameters are seen: - 0.41, popularity and weighted_rating - 0.42, vote_count and weighted_rating - 0.30, profit and weighted_rating

It's time to draw comparisons from some of these corralated parameters from my main dataset

```
# Creating a Scatter Graph of vote count and profit

g <- ggplot(final_movies_data, aes(x = vote_count, y = profit)) +
  geom_point() +
  labs(title = "Scatter Plot of Vote Count vs Profit",
       x = "Vote Count",
       y = "Profit") +
  theme_minimal()

# Print the plot
print(g)
```

Vote Count vs Profit

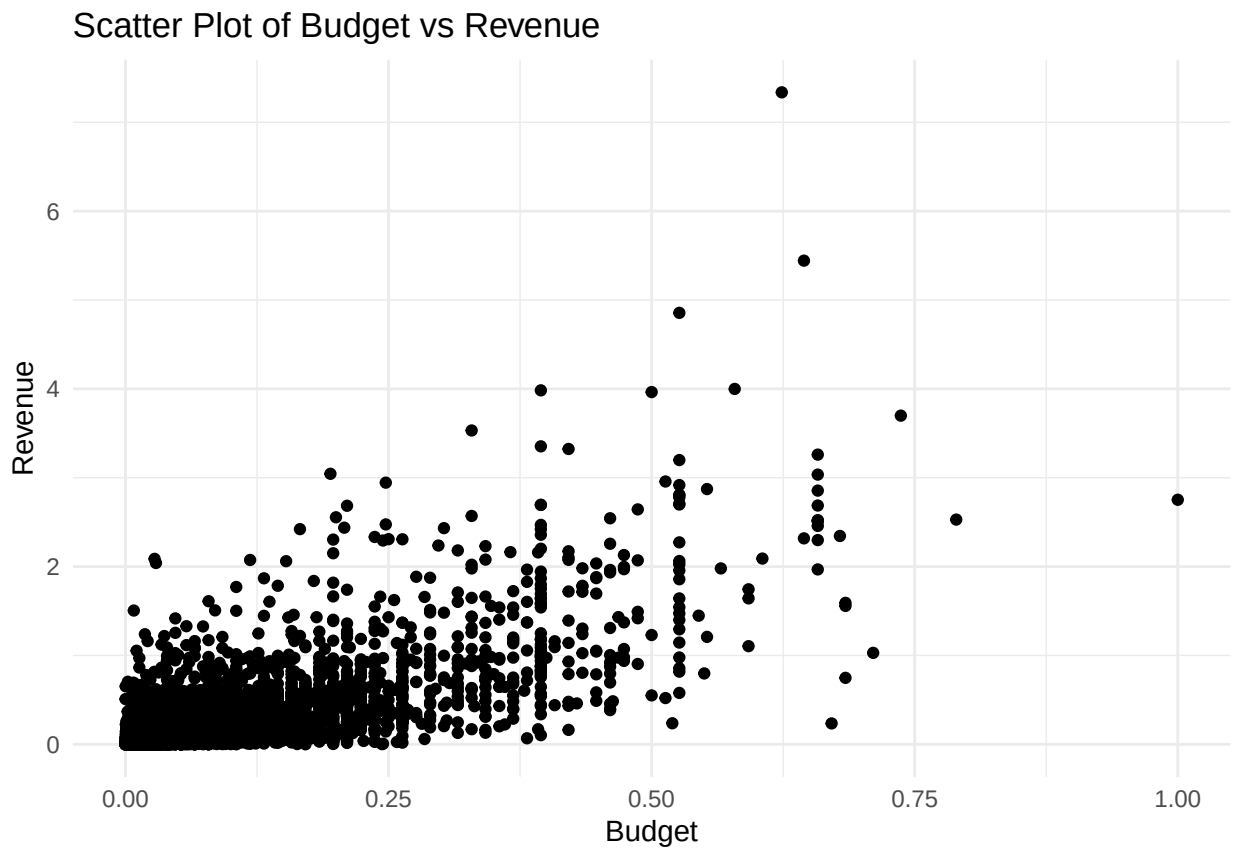
```
## Warning: Removed 40751 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



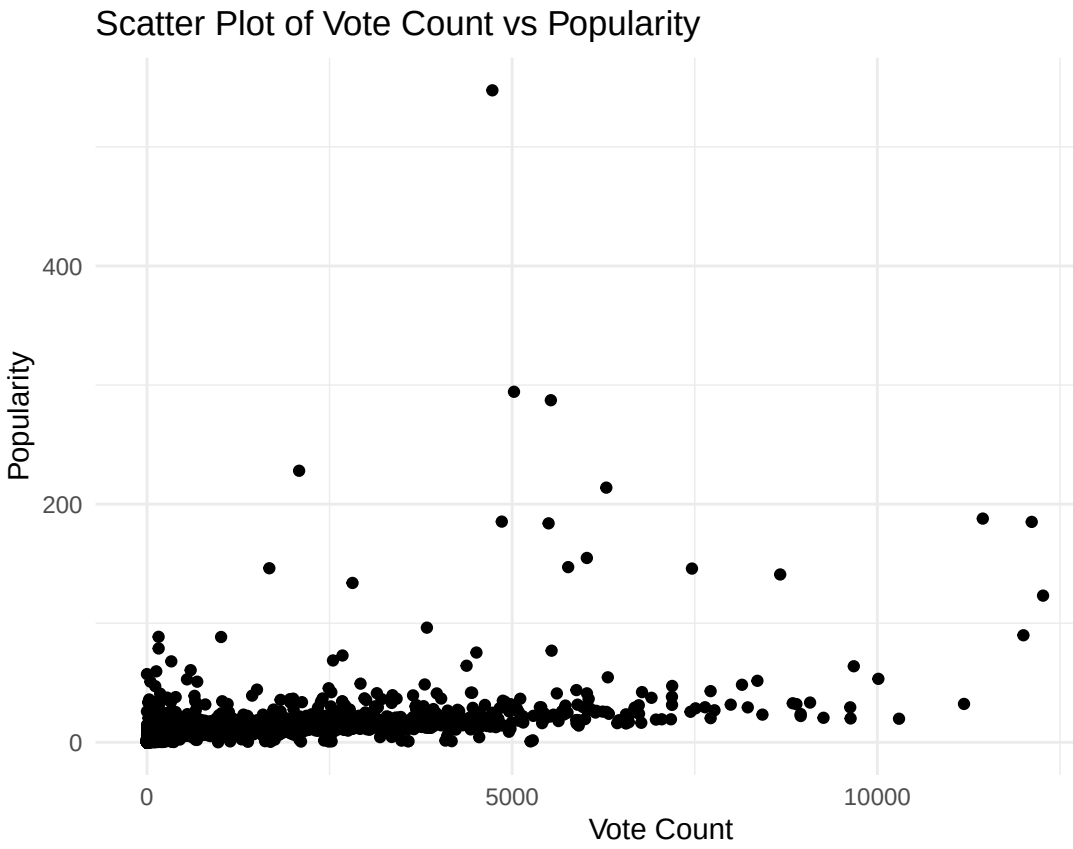

```
# Creating the scatter plot for budget vs. revenue
g <- ggplot(final_movies_data, aes(x = budget, y = revenue)) +
  geom_point() +
  labs(title = "Scatter Plot of Budget vs Revenue",
       x = "Budget",
       y = "Revenue") +
  theme_minimal()

# Print the plot
print(g)
```

Budget vs Revenue

```
## Warning: Removed 40751 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

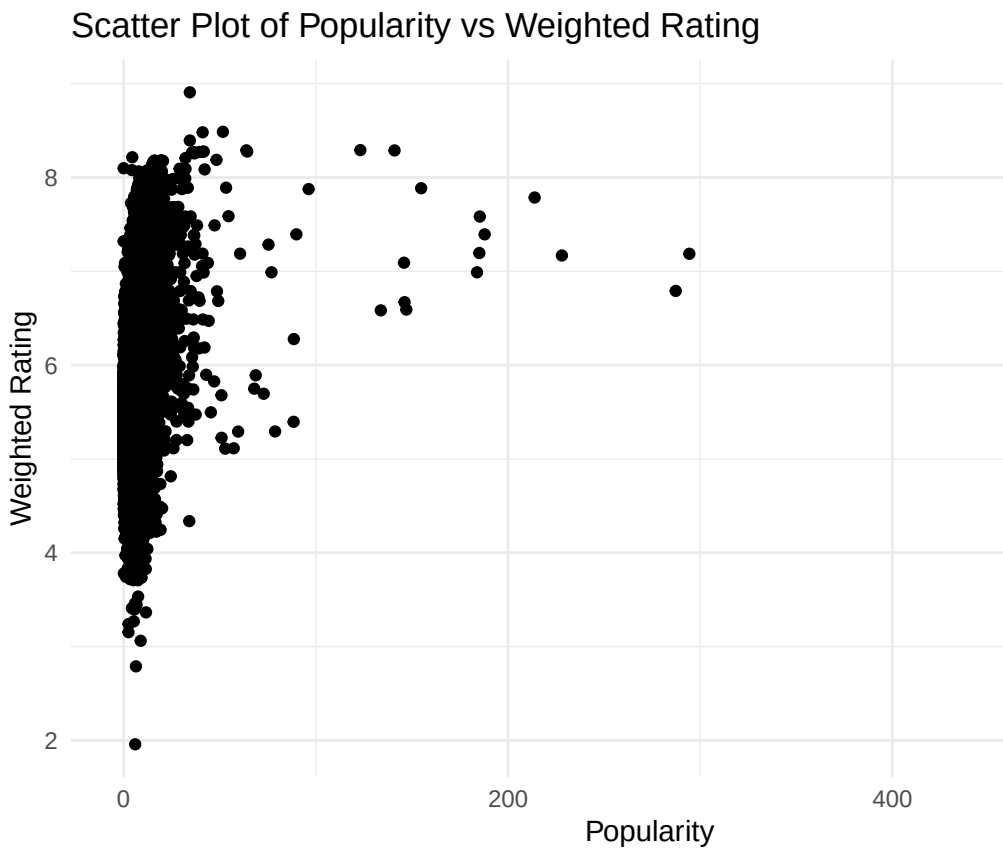



```r
# Creating the scatter plot for vote count vs. popularity

g <- ggplot(final_movies_data, aes(x = vote_count, y = popularity)) +
  geom_point() +
  labs(title = "Scatter Plot of Vote Count vs Popularity",
       x = "Vote Count",
       y = "Popularity") +
  theme_minimal()

# Print the plot
print(g)
```



Vote Count vs Popularity

```r
g <- ggplot(final_movies_data, aes(x = popularity, y = weighted_rating)) +
  geom_point() +
  labs(title = "Scatter Plot of Popularity vs Weighted Rating",
       x = "Popularity",
       y = "Weighted Rating") +
  theme_minimal()

# Print the plot
print(g)
```



Popularity vs Weighted Rating

After the analysis of the numerical data,It's now time to look at the specific data to answer my Business Objective

Performance of the Last 5 years

I've decided to only look at data from the 2016-2020 period as this is the most recent in the dataset, this is in order to make my analysis more relevent to the Business Objective of making a Future Investment. A concern I have is that more recent data has probably been provided since this dataset which may skew my findings.

```
# Ensure 'release_year' is numeric
final_movies_data$release_year <- as.numeric(final_movies_data$release_year)

# Define the range for the last 5 years
last_5_years <- (max(final_movies_data$release_year) - 4):max(final_movies_data$release_year)

# Filter the dataset for the last 5 years
movies_last_5_years <- final_movies_data %>%
  filter(release_year %in% last_5_years)

# Verify the filtered dataset
print(head(movies_last_5_years))
```

```
##      id belongs_to_collection     budget                                genres
```

```
## 1 14564                    1 0.06579121                            Horror
## 2 38700                    1         NA            Thriller; Action; Crime
## 3 43074                    1 0.37895734            Action; Fantasy; Comedy
## 4 47933                    1 0.43422195 Action; Adventure; Science Fiction
## 5 47971                    1 0.22369010          Action; Adventure; Crime
## 6 58431                    1 0.07368615  Romance; Horror; Comedy; Thriller
##   original_language popularity
## 1                en  24.535733
## 2                en   2.178546
## 3                en  17.162744
## 4                en  16.993841
## 5                en  17.918269
## 6                en   7.954623
##
## 1                                Paramount Pictures; Vertigo Entertainment; Macari/Edelstein; Parkes
## 2                                                                  Columbia Pictures; Sony Pictu
## 3 Columbia Pictures; Village Roadshow Pictures; Montecito Picture Company, The; LStar Capital; Feigc
## 4                                                Twentieth Century Fox Film Corporation; Centropolis
## 5                                               Paramount Pictures; Revolution Studios; One Race Fil
## 6                                  Lionsgate; Handsomecharlie Films; Cross Creek Pictures; Darko Ente
##      production_countries release_date    revenue   runtime   status
## 1 United States of America   2017-02-01 0.21863967 102.00000 Released
## 2 United States of America   2018-11-07         NA  97.48758  Planned
## 3 United States of America   2016-07-14 0.60303563 116.00000 Released
## 4 United States of America   2016-06-22 1.02550576 120.00000 Released
## 5 United States of America   2017-01-13 0.91093886 107.00000 Released
## 6 United States of America   2016-02-04 0.04309147 108.00000 Released
##                            title vote_average vote_count release_year
## 1                          Rings          4.8       1075         2017
## 2              Bad Boys for Life          0.0         12         2018
## 3                   Ghostbusters          5.3       2203         2016
## 4   Independence Day: Resurgence          4.9       2550         2016
## 5     xXx: Return of Xander Cage          5.5       1497         2017
## 6 Pride and Prejudice and Zombies          5.5        662         2016
##       profit weighted_rating
## 1  0.15284847        4.814474
## 2         NA        3.835023
## 3  0.22407829        5.299366
## 4  0.59128381        4.904865
## 5  0.68724876        5.494508
## 6 -0.03059468        5.487939
##
## 1 hallucination; investigation; drama; sequel; suspense; curse; vhs; death; maggot; supernatural hor
## 2                                                                                          adventu
## 3                                                                        female friendship; gl
## 4                                                                                          alterna
## 5         tattoo; spy; airplane; secret agent; parachute; swimming pool; sequel; extreme sports; ba
## 6                          based on novel; dystopia; shaolin; prejudice; zombie; pride; golddigger; reg
##
Morgan', 'ord## 1 4, 'profile_path': '/v98s8QgimO6P3aRBd6lRTAdUcXS.jpg'}, {'cast_id': 15, 'character': 'Burke', 'cred
## 2
## 3 [{'cast_id': 14, 'character': 'Abby Yates', 'credit_id': '54e47470c3a36823d4002695', 'gender': 1,
## 4
cast_id': 139,## 5haracter': "Ainsley's Girls", 'credit_id': '590cbf66c3a3686519001764', 'gender': 1, 'id': 1811045, 'n
```

```
vLU595BdBMcLjo##w6jpg'}, {'cast_id': 20, 'character': 'Mr. Bennet', 'credit_id': '542318660e0a263b84001d56', 'gender'
##              director
## 1 F. Javier Gutiérrez
## 2        Joe Carnahan
## 3           Paul Feig
## 4     Roland Emmerich
## 5         D.J. Caruso
## 6         Burr Steers
```

Most Films Per Year by Director

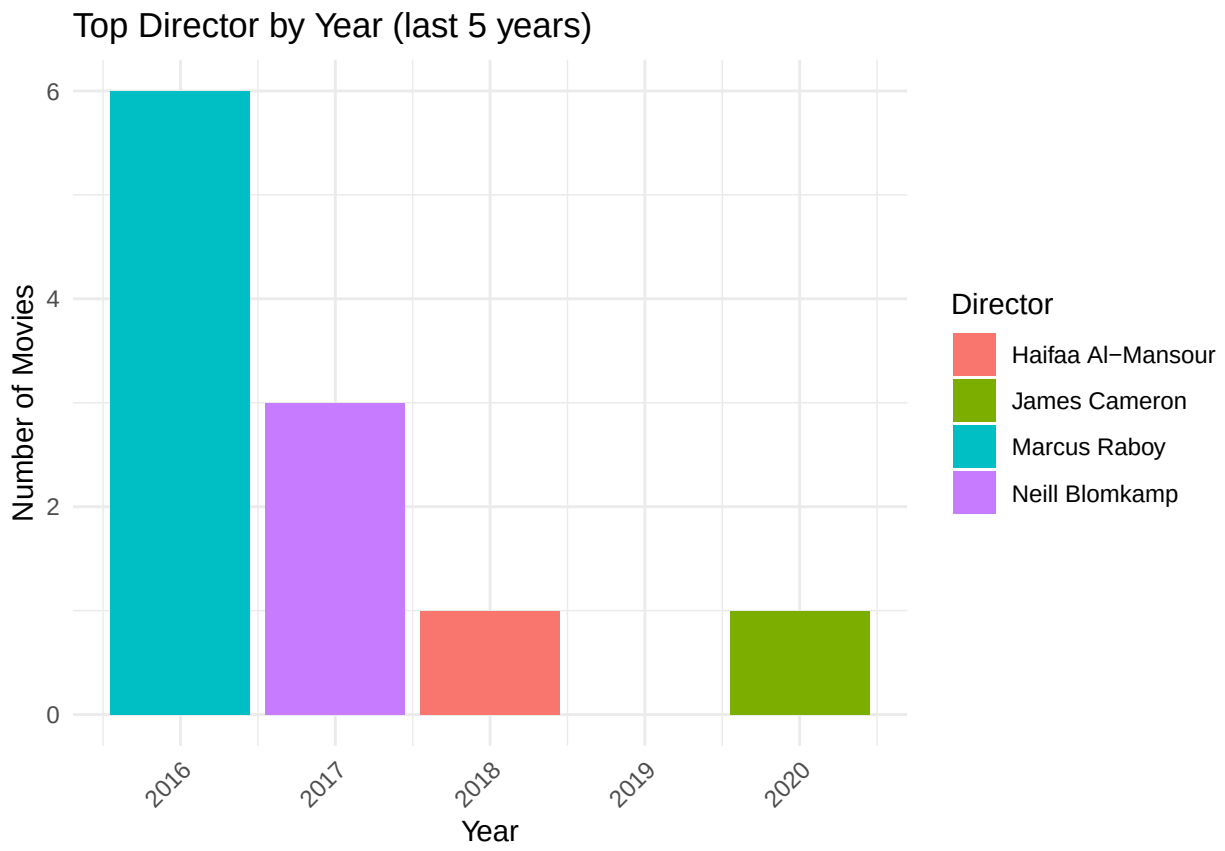
```
# Creating a new data table named top directors by each year
most_films_directors <- movies_last_5_years %>%
  filter(!is.na(release_year) & !is.na(director)) %>%
  group_by(release_year, director) %>%
  summarise(count = n()) %>%
  arrange(release_year, desc(count)) %>%
  slice(1) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'release_year'. You can override using the
## `.groups` argument.
```

```
# Verify the top directors by year
print(most_films_directors)
```

```
## # A tibble: 4 x 3
##   release_year director          count
##          <dbl> <chr>             <int>
## 1         2016 Marcus Raboy          6
## 2         2017 Neill Blomkamp        3
## 3         2018 Haifaa Al-Mansour     1
## 4         2020 James Cameron         1
```

```
# Plot the top directors by year
ggplot(most_films_directors, aes(x = release_year, y = count, fill = director)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Top Director by Year (last 5 years)",
       x = "Year",
       y = "Number of Movies",
       fill = "Director") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



This Shows the most active directors over the last 5 years, this helps us see what directors take on the most work, this could help us derterime a directors potential availability and willingness to explore a new project.

Most Profitable Director's in the last 5 years

```
# Group by year and director, then summarize profit
top_profitable_directors <- movies_last_5_years %>%
  filter(!is.na(release_year) & !is.na(profit)) %>%
  group_by(release_year, director) %>%
  summarise(total_profit = sum(profit, na.rm = TRUE)) %>%
  arrange(release_year, desc(total_profit)) %>%
  slice(1) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'release_year'. You can override using the
## `.groups` argument.
```

```
# Verify the top profitable directors by year
print(top_profitable_directors)
```

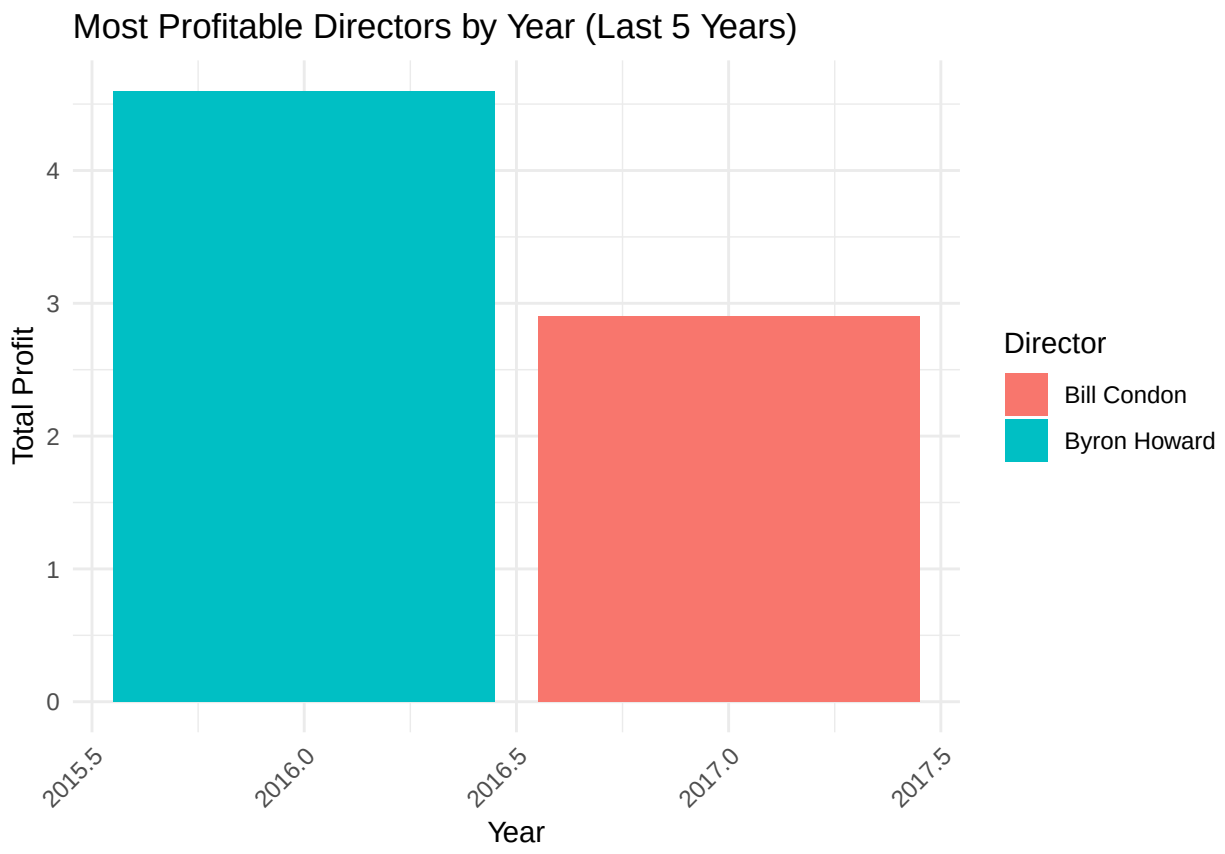
```
## # A tibble: 2 x 3
##   release_year director     total_profit
##          <dbl> <chr>               <dbl>
```

```
## 1          2016 Byron Howard          4.60
## 2          2017 Bill Condon           2.90
```

```r
# Verify the top profitable directors by year
print(top_profitable_directors)
```

```
## # A tibble: 2 x 3
##   release_year director     total_profit
##          <dbl> <chr>               <dbl>
## 1         2016 Byron Howard         4.60
## 2         2017 Bill Condon          2.90
```

```r
# Plot the most profitable directors for the last 5 years
ggplot(top_profitable_directors, aes(x = release_year, y = total_profit, fill = director)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Most Profitable Directors by Year (Last 5 Years)",
       x = "Year",
       y = "Total Profit",
       fill = "Director") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



As you can see data is only available for the first 2 years 2016-2017 therefore not much of a conclusion can be derived from this

Highest Rating by Director

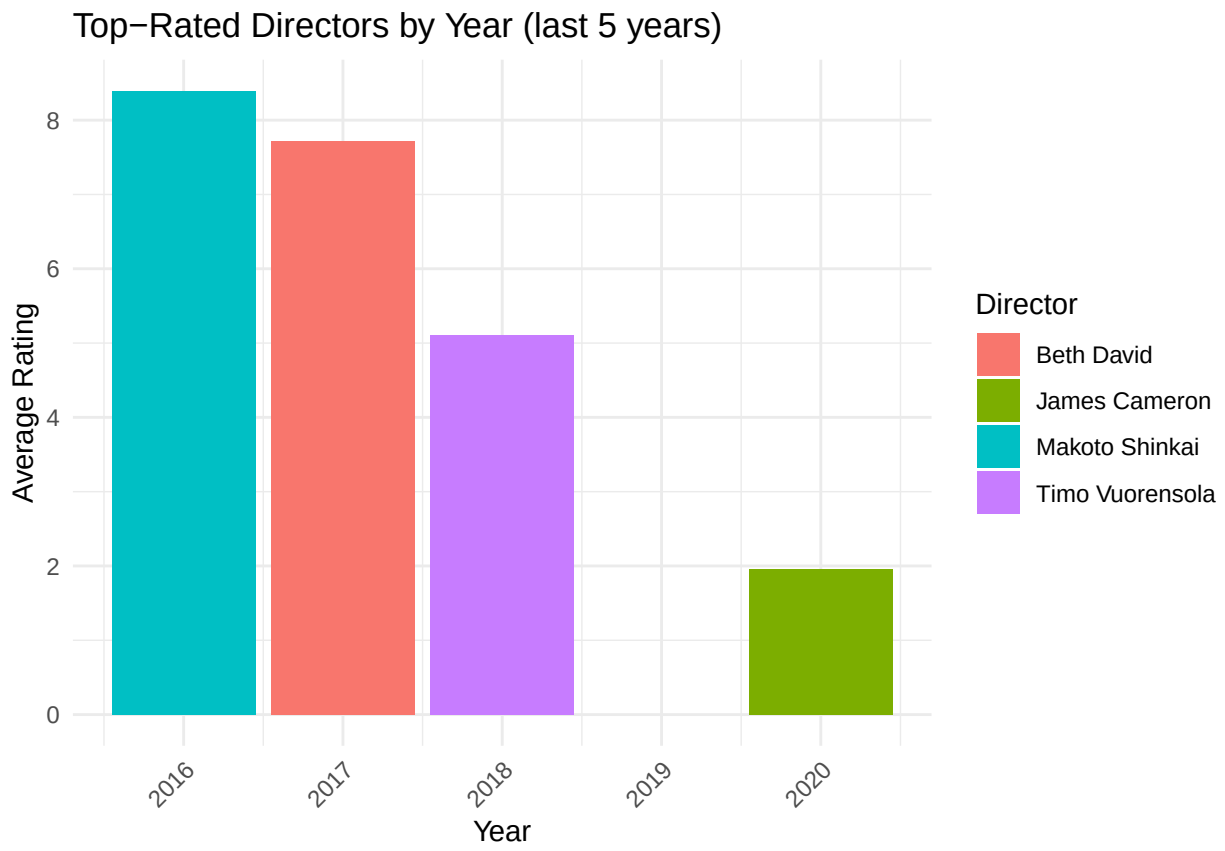
```
# Group by year and director, then summarize rating
top_rated_directors <- movies_last_5_years %>%
  filter(!is.na(release_year) & !is.na(weighted_rating)) %>%
  group_by(release_year, director) %>%
  summarise(average_rating = mean(weighted_rating, na.rm = TRUE)) %>%
  arrange(release_year, desc(average_rating)) %>%
  slice(1) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'release_year'. You can override using the
## `.groups` argument.
```

```
# Verify the top-rated directors by year
print(top_rated_directors)
```

```
## # A tibble: 4 x 3
##   release_year director        average_rating
##          <dbl> <chr>                    <dbl>
## 1         2016 Makoto Shinkai            8.39
## 2         2017 Beth David                7.72
## 3         2018 Timo Vuorensola           5.11
## 4         2020 James Cameron             1.96
```

```
# Plot the top-rated directors by year
ggplot(top_rated_directors, aes(x = release_year, y = average_rating, fill = director)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Top-Rated Directors by Year (last 5 years)",
       x = "Year",
       y = "Average Rating",
       fill = "Director") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



This Shows the best rated directors, each year over the last 5 years, this gives us an indication of what director we could hire that would be able to create a film that would be well recieved by the audience.

Films by Genre over the Last 5 years

```
# Separate combined genres
genre_separated_last_5 <- movies_last_5_years %>%
  separate_rows(genres, sep = "; ")

# Verify the separated genres
print(head(genre_separated_last_5))
```

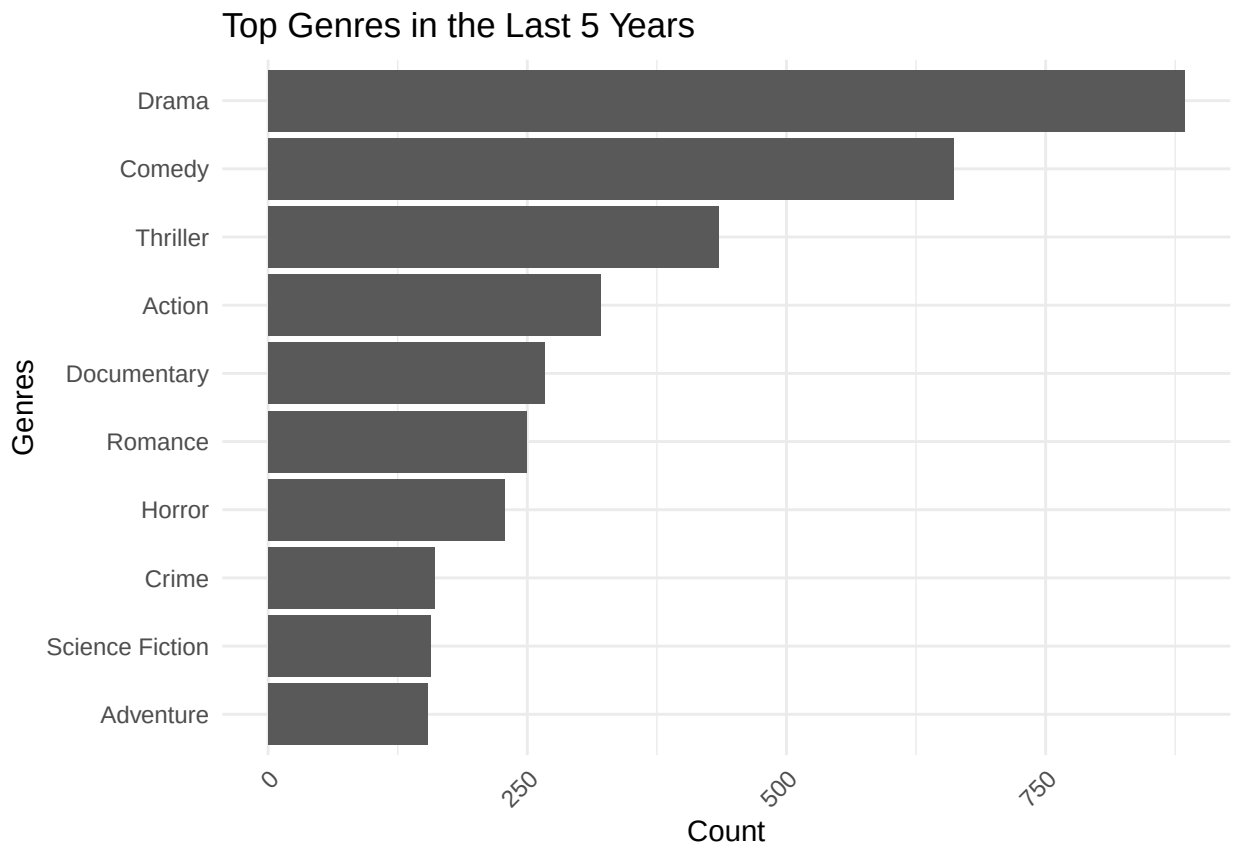
```
## # A tibble: 6 x 21
##      id belongs_to_collection  budget genres   original_language popularity
##   <int>                 <dbl>   <dbl> <chr>    <chr>                  <dbl>
## 1 14564                     1  0.0658 Horror   en                      24.5
## 2 38700                     1 NA      Thriller en                       2.18
## 3 38700                     1 NA      Action   en                       2.18
## 4 38700                     1 NA      Crime    en                       2.18
## 5 43074                     1  0.379  Action   en                      17.2
## 6 43074                     1  0.379  Fantasy  en                      17.2
## # i 15 more variables: production_companies <chr>, production_countries <list>,
## #   release_date <date>, revenue <dbl>, runtime <dbl>, status <chr>,
## #   title <chr>, vote_average <dbl>, vote_count <int>, release_year <dbl>,
```

```
## #   profit <dbl>, weighted_rating <dbl>, keywords <chr>, cast <list>,
## #   director <chr>
```

```
# Count the occurrences of each genre
genre_counts_last_5 <- genre_separated_last_5 %>%
  count(genres, sort = TRUE)

# Verify the genre counts
print(genre_counts_last_5)
```

```
## # A tibble: 20 x 2
##    genres               n
##    <chr>            <int>
##  1 "Drama"            884
##  2 "Comedy"           661
##  3 "Thriller"         435
##  4 "Action"           321
##  5 "Documentary"      267
##  6 "Romance"          249
##  7 "Horror"           228
##  8 "Crime"            161
##  9 "Science Fiction"  157
## 10 "Adventure"        154
## 11 "Family"           108
## 12 "Animation"        104
## 13 "Mystery"          101
## 14 "Fantasy"           96
## 15 "TV Movie"          69
## 16 "History"           67
## 17 "Music"             63
## 18 ""                  54
## 19 "War"               51
## 20 "Western"           16
```

```
# Filter to the top genres
top_genres_last_5 <- genre_counts_last_5 %>%
  top_n(10, n) %>%
  arrange(desc(n))

# Plot the top genres
ggplot(top_genres_last_5, aes(x = reorder(genres, n), y = n)) +
  geom_bar(stat = "identity") +
  labs(title = "Top Genres in the Last 5 Years",
       x = "Genres",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  coord_flip()
```



As you can see Drama and Comedy Top the chart meaning these movies are the most prevalent over the last 5 years

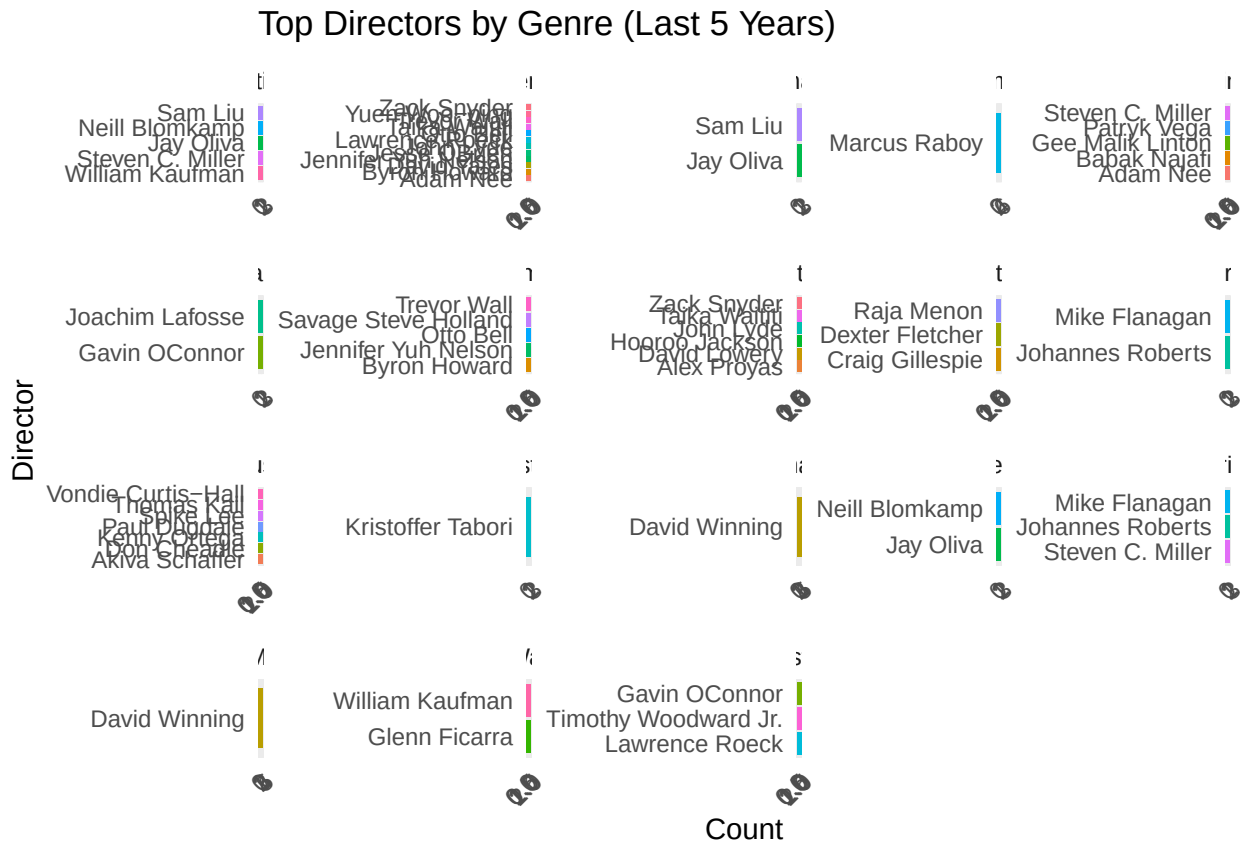
Genres by Director over the Last 5 years

```
# Group by genres and director, then count
genre_director_counts_last_5 <- genre_separated_last_5 %>%
  group_by(genres, director) %>%
  summarise(count = n(), .groups = 'drop') %>%
  arrange(genres, desc(count)) %>%
  group_by(genres) %>%
  slice_max(count, n = 1) %>%
  ungroup()

genre_director_counts_last_5 <- genre_director_counts_last_5 %>%
  filter(!is.na(director) & !is.na(genres))

# Verify the genre-director counts
print(head(genre_director_counts_last_5))
```

```
## # A tibble: 6 x 3
##   genres    director         count
##   <chr>     <chr>            <int>
## 1 Action    Jay Oliva            3
```

```
## 2 Action    Neill Blomkamp       3
## 3 Action    Sam Liu              3
## 4 Action    Steven C. Miller     3
## 5 Action    William Kaufman      3
## 6 Adventure Adam Nee             2
```

```
# Plot the top directors for each genre
ggplot(genre_director_counts_last_5, aes(x = reorder(director, count), y = count, fill = director)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ genres, scales = "free") +
  labs(title = "Top Directors by Genre (Last 5 Years)",
       x = "Director",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none") +
  coord_flip()
```



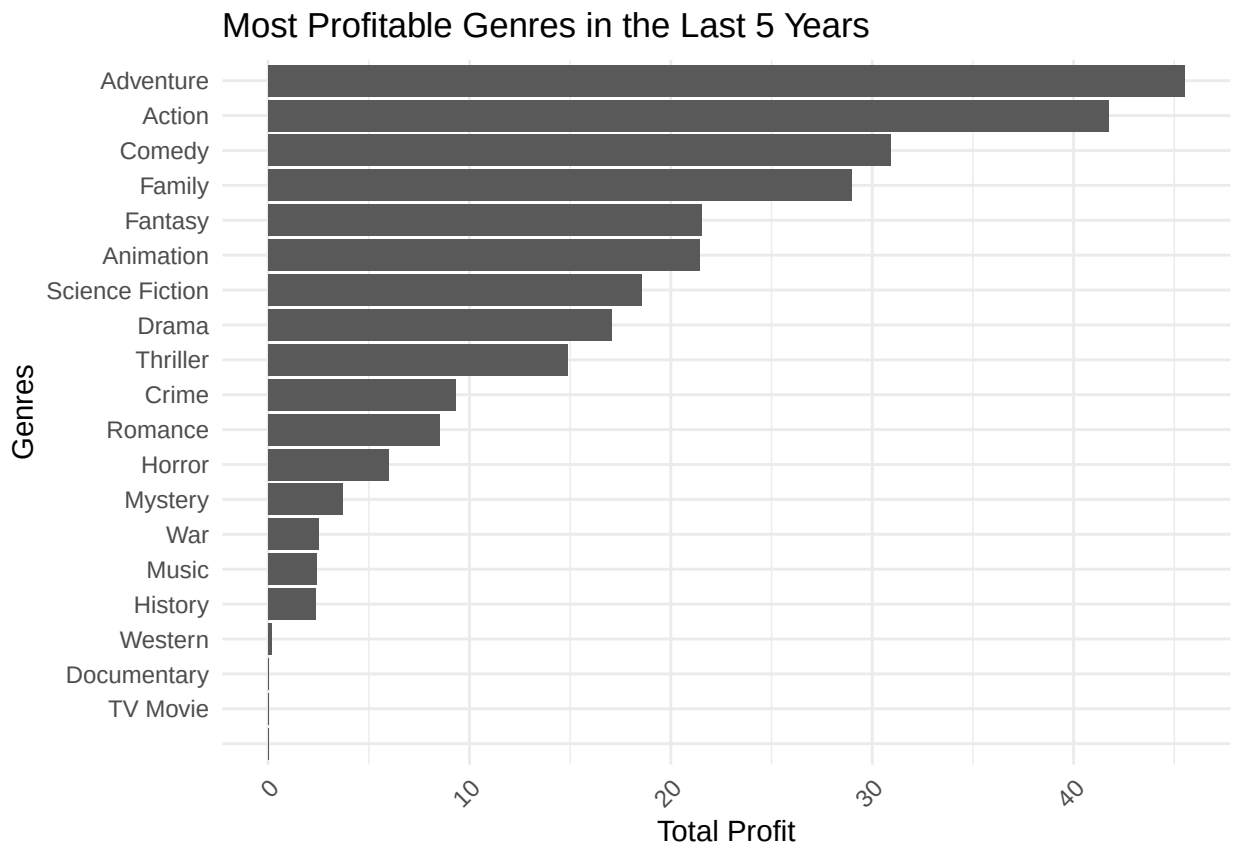
This gives us an indication of who we should hire to Direct our movie based on genre as certain Directors will be more specialised towards a certain Genre

Genre by Profit over the last 5 years

```r
# Group by genres and summarize the total profit
genre_profit_last_5 <- genre_separated_last_5 %>%
  group_by(genres) %>%
  summarise(total_profit = sum(profit, na.rm = TRUE)) %>%
  arrange(desc(total_profit))

genre_profit_last_5 <- genre_profit_last_5 %>%
  filter(!is.na(genres) & !is.na(total_profit))

# Verify the genre profit
print(genre_profit_last_5)
```

```
## # A tibble: 20 x 2
##    genres            total_profit
##    <chr>                    <dbl>
##  1 "Adventure"             45.5
##  2 "Action"                41.7
##  3 "Comedy"                30.9
##  4 "Family"                29.0
##  5 "Fantasy"               21.5
##  6 "Animation"             21.4
##  7 "Science Fiction"       18.5
##  8 "Drama"                 17.1
##  9 "Thriller"              14.9
## 10 "Crime"                  9.33
## 11 "Romance"                8.49
## 12 "Horror"                 5.98
## 13 "Mystery"                3.68
## 14 "War"                    2.49
## 15 "Music"                  2.40
## 16 "History"                2.38
## 17 "Western"                0.177
## 18 "Documentary"            0.0211
## 19 ""                       0
## 20 "TV Movie"               0
```

```r
# Plot the most profitable genres
ggplot(genre_profit_last_5, aes(x = reorder(genres, total_profit), y = total_profit)) +
  geom_bar(stat = "identity") +
  labs(title = "Most Profitable Genres in the Last 5 Years",
       x = "Genres",
       y = "Total Profit") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  coord_flip()
```



Adventure tops the chart with Action and Comedy films also showing they make a good amount of profit.

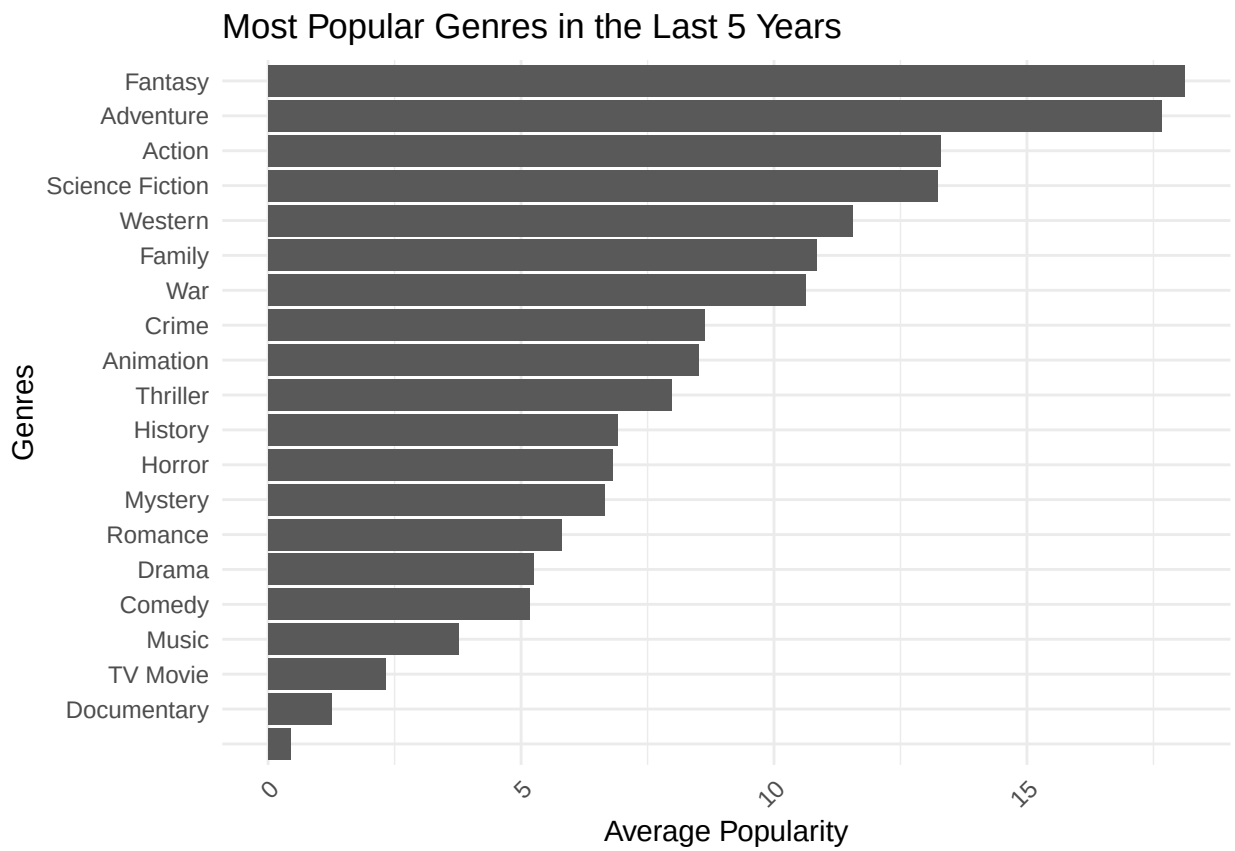
Genre Popularity over the last 5 years

```
# Group by genres and summarize the average popularity
genre_popularity_last_5 <- genre_separated_last_5 %>%
  group_by(genres) %>%
  summarise(average_popularity = mean(popularity, na.rm = TRUE)) %>%
  arrange(desc(average_popularity))

# Verify the genre popularity
print(genre_popularity_last_5)
```

```
## # A tibble: 20 x 2
##    genres            average_popularity
##    <chr>                          <dbl>
##  1 "Fantasy"                       18.1
##  2 "Adventure"                     17.7
##  3 "Action"                        13.3
##  4 "Science Fiction"               13.2
##  5 "Western"                       11.6
##  6 "Family"                        10.8
##  7 "War"                           10.6
##  8 "Crime"                          8.64
```

```
##  9 "Animation"                 8.51
## 10 "Thriller"                  7.98
## 11 "History"                   6.91
## 12 "Horror"                    6.81
## 13 "Mystery"                   6.64
## 14 "Romance"                   5.80
## 15 "Drama"                     5.25
## 16 "Comedy"                    5.17
## 17 "Music"                     3.77
## 18 "TV Movie"                  2.32
## 19 "Documentary"               1.25
## 20 ""                          0.444
```

```r
# Plot the most popular genres
ggplot(genre_popularity_last_5, aes(x = reorder(genres, average_popularity), y = average_popularity)) +
  geom_bar(stat = "identity") +
  labs(title = "Most Popular Genres in the Last 5 Years",
       x = "Genres",
       y = "Average Popularity") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  coord_flip()
```



Adventure and Action are high which would be expected based on the last chart, however it is surprising to see Comedy quite low down based on popularity.

Conclusion

Adventure and *Action* films seem to be well received by the Public and draw a substantial amount of profit.

Comedy films draw a high amount of profit but the quantity of films made in this genre is high which could be a driving factor in this, as the *Popularity* of *Comedy* films from the public over the last 5 years is actually low compared to other high profit genres.

Science Fiction and *Adventure* films despite being the lowest in terms of quantity being made seem to be the most profitable, these films seem to be niche films with a high fandom as both these films seem to have high Popularity among the other genres.

Recommendations

I suggest the Investment Agency start a Film Project with the Emphasis being on *Action/Science Fiction* with the director being either *Neill Blomkamp* or *Jay Oliva* as these two directors are familar with this type of genre.

Next Steps

I will be Creating Visualisations and a Dashboard of the 3 Datasets I created earlier from the full dataset.

- Top Profitable Movies
- Top Rated Movies
- Top Popular Movies