TMDB Box Office Prediction

Gabriel Lapointe

June 9, 2019

Contents

1 Business Overview

1.1 Context

In a world… where movies made an estimated $41.7 billion in 2018, the film industry is more popular than ever. For some movies, it's "You had me at 'Hello.' " For others, the trailer falls short of expectations and you think "What we have here is a failure to communicate."

1.2 Problem

With metadata on over 7,000 past films from The Movie Database, we have to predict their overall worldwide box office revenue.

1.3 Objective

The objective is to determine:

- What movies make the most money at the box office?
- How much does a director matter?
- How much does the budget matter?

We have to determine the movie revenue ($) in function of the budget ($) and in function of if the movie had a director as a member of the crew when they made it. Then, we have to identify let's say a top 10 in descending order of the predicted movie revenues.

2 Dataset Information

The train and test datasets come from the Kaggle competition TMDB Box Office Prediction. The objective is to know from the train dataset:

- The column names
- The number of columns
- The number of rows
- The number of values that are `NA` and empty

Knowing this information gives hints on how to analyse the data and how the features could be correlated. If there are any NA or empty values, we have to understand the meaning of NA and empty in their context. This holds also for budgets of 0$.

2.1 General Information

```
Number of rows:  3000
Number of columns:  22
```

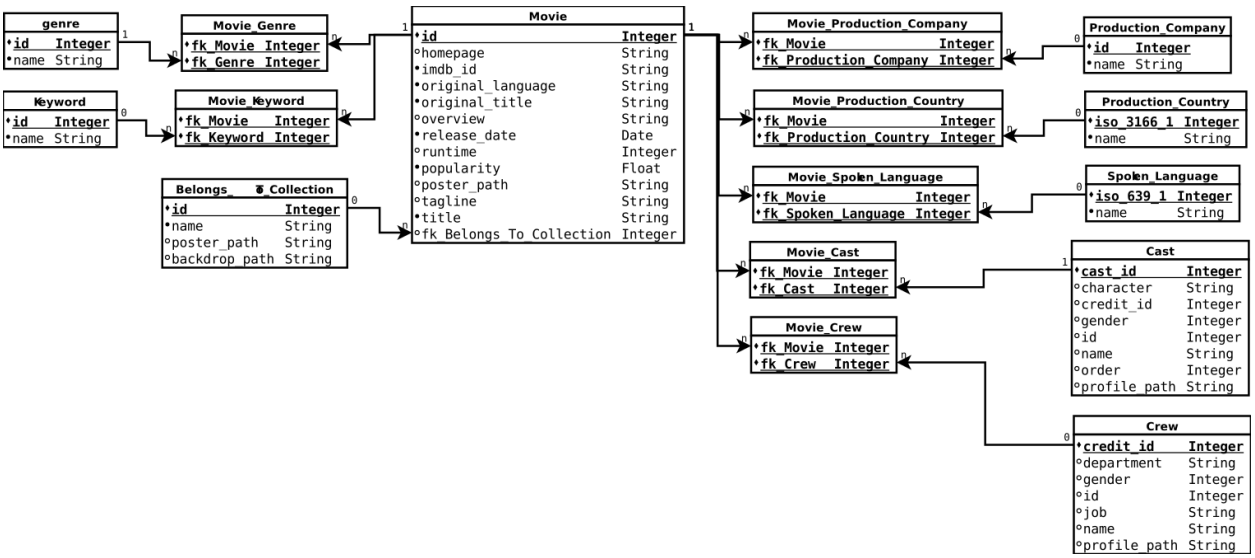
	Type	Number_of_NA	Number_of_empty
belongs_to_collection	character	0	2396
budget	integer	0	0
genres	character	0	7
homepage	character	0	2054
imdb_id	character	0	0
original_language	character	0	0
original_title	character	0	0

	Type	Number_of_NA	Number_of_empty
overview	character	0	8
popularity	numeric	0	0
poster_path	character	0	1
production_companies	character	0	156
production_countries	character	0	55
release_date	character	0	0
runtime	integer	2	NA
spoken_languages	character	0	20
status	character	0	0
tagline	character	0	597
title	character	0	0
Keywords	character	0	276
cast	character	0	0
crew	character	0	3
revenue	integer	0	0

```
Number of rows:  4398
Number of columns:  21
```

	Type	Number_of_NA	Number_of_empty
belongs_to_collection	character	0	3521
budget	integer	0	0
genres	character	0	16
homepage	character	0	2978
imdb_id	character	0	0
original_language	character	0	0
original_title	character	0	0
overview	character	0	14
popularity	numeric	0	0
poster_path	character	0	1
production_companies	character	0	258
production_countries	character	0	102
release_date	character	0	1
runtime	integer	4	NA
spoken_languages	character	0	42
status	character	0	2
tagline	character	0	863
title	character	0	3
Keywords	character	0	393
cast	character	0	0
crew	character	0	9

Here is the relational model of the dataset that is described in more details in sub-sections below.



2.2 Possible Algorithms

We know that the objective is to predict the revenue of every movie in the test dataset. We also know that:

- The output is the revenue of every movie.
- The output is a real number.
- The accuracy of the prediction is important over the speed (which should not be neglected)

Therefore, we need algorithms in the supervised learning and regression category. In this analysis, we consider the following models considering that the speed has to be taken in consideration:

- Linear regression
- Random Forest
- Extreme Gradient Boosting Trees

2.3 Code Book

2.3.1 Movie Collection Information

The feature `Belongs_to_collection` contains the following properties as a JSON array:

- id: The movie collection ID (Integer)
- name: The movie collection name (string)
- poster_path: The movie collection poster image path(string)
- backdrop_path: The movie collection backdrop path (string)

The value of this feature is empty if the movie is not in a collection (e.g.The movie collection Back to the future 1, 2, and 3). We deduce that:

- A movie is in its own collection if and only if the feature `belongs_to_collection` is empty.
- A movie cannot be in more than one collection.
- A collection may contain many movies.

2.3.2 Movie Genres Information

The feature `genres` contains the following properties as a JSON array:

- id: The genre ID (integer)

- name: The name of the genre

According to the dataset, we deduced that:

- A movie can have one or many genres.
- A genre can contain one or many movies.
- A movie can have no genre meaning that it is not classified.

2.3.3 Production Companies Information

The feature `production_companies` contains the following properties as a JSON array:

- iso_3166_1: The production company country abbreviation (e.g. US)
- name: The production company name

According to the dataset, we deduced that:

- A movie can be produced by one or many companies.
- A production company can have produced one or many movies.
- A movie that is not associated to at least one production company could be produced by amators or particular producer producing his own movies.

2.3.4 Production Countries Information

The feature `production_countries` contains the following properties as a JSON array:

- iso_3166_1: The production country abbreviation (e.g. US) where the movie has been produced
- name: The production country name

According to the dataset, we deduced that:

- A movie can be produced in one or many countries.
- A production country can contain one or many movies.
- A movie that is not associated to at least one production country could be ???

2.3.5 Spoken Languages Information

The feature `spoken_languages` contains the following properties as a JSON array:

- iso_639_1: The spoken language country abbreviation (e.g. US)
- name: The spoken language name (could contain non ascii characters)

According to the dataset, we deduced that:

- In a movie, one or many languages can be spoken.
- A spoken language can be used in one or many movies.
- A movie that is not associated to at least one spoken language could be silent movies like Charlie Chaplin or Mr. Bean.

2.3.6 Keywords Information

The feature `Keywords` contains the following properties as a JSON array:

- id: The keyword ID (Integer)
- name: The keyword (String)

According to the dataset, we deduced that:

- One or many keywords can be used for a movie.
- One or many movies can be associated to a keyword.

- A movie that is not associated to at least one keyword could be ???.

2.3.7 Cast Information

The feature `cast` contains the following properties as a JSON array:

- cast_id: The cast ID corresponding to one character playing in the movie.
- character: The character name in the movie (could be empty)
- credit_id: The credit ID as an hexadecimal string
- gender: The character is a female (1), a male (2) or other (0)
- id: The ID of the actor playing in the movie or of an event occuring in the movie
- name: The name of the actor playing in the movie or of the event occuring in the movie
- order: The order of displaying in the credit at the end of the movie
- profile_path: Avatar of the actor of the movie as an image path

Note that an actor can play in one or many movies and a cast can contain one or many actors.

2.3.8 Crew Information

The feature `crew` contains the following properties as a JSON array:

- credit_id: The credit ID as an hexadecimal string representing a member of the crew
- department: Department name in which the member of the crew was working (e.g. Directing, Writing, Sound, etc.)
- gender: The member of the crew is a female (1), a male (2) or other (0)
- id: The ID of the member of the crew
- job: The job name of the member of the crew (e.g. Director, Producer, Writer, Screenplay, etc.)
- name: Name of the member of the crew
- profile_path: Profile image path of the member of the crew

Note that a member of the crew can be hired to help making one or many movies and a movie can contain one or many members of the crew.

3 Data Preparation

The objective of preparing the data is to clean the dataset and make the dataset workable in order to visualize the data. Preparing the dataset is represented by the following steps:

1. Detect and fix values that seem to be wrong in their context.
2. Replace the NA or empty values in the dataset by meaningful values.
3. Determine and remove columns that will not help us on the visualisation of data.
4. Add new features from existing ones.

3.1 JSON Standard Validation

The objective is to detect features that contain invalid keys and/or values and fix them with valid values without modifying the context of the data.

If we take a closer look to the JSON strings, we notice that their keys and values of type string are all surrounded by single quotes. Here is an example taken from the first observation of the feature `belongs_to_collection`: `[{'id': 313576, 'name': 'Hot Tub Time Machine Collection', 'poster_path': '/iEhb00TGPucF0b4joM1ieyY026U.jpg', 'backdrop_path': '/noeTVcgpBiD48fDjFVic1Vz7ope.jpg'}]`. This is not respecting the JSON standard which requires double quotes for string values. The library `jsonlite` is validating this standard and we get a validation error. In order to fix that, we have to replace

all single quotes in all JSON strings of the dataset by double quotes. However, we have to assume that there may have single or double quotes in the string value itself.

Here is a list of syntax rules that we verify:

1. All keys have to be surrounded by double quotes `"` followed by a colon `:`, a space and its mapped value (e.g. `"id": 1`).
2. All string values have to be surrounded by double quotes `"` (e.g. `"string value"`).
3. Special characters like double quotes (”), backslashes `\` and squares `#` must not be used.
4. The value `None` (e.g. `"backdrop_path": None`), `[]`, `N/A` or empty must not be used as a value. The value `null` has to be used instead.
5. Each element of an array has to start with a bracket `{` followed by a double quote `"` (e.g. `{"id"`).
6. Each element of an array has to end with a bracket `}` (e.g. `"name": "Comedy"}`).
7. Each mapping (key, value) has to be separated by a comma `,` (e.g. `"id": 18, "name": "Drama"` or `"name": null, "id": 2` or `"name": "US", "id": 1`).

For example, there are some actors having a nickname in their character name in the cast. Those are written between double quotes (e.g. in the cast of Rocky Balboa: `'cast_id': 17, 'character': 'Adrianna "Adrian" Pennino'`). Because of such a case, we have to replace all double quotes by single quotes on first step.

3.2 Incorrect Values

An incorrect value is a value that is invalid in the feature context. For example, the feature `crew` having an observation that gives the keywords instead or the crew. It can be seen as a misplaced value. We also validate the coherence between observations of a same feature. It could occur that for the feature `genres` that all observations except one start with the `id` whereas one of them starts with `name` instead. This is an incoherence between this observation and the others.

Since there are too many error possibilities, we establish the following rules for every observation in every feature containing JSON strings:

- The JSON string in the feature `belongs_to_collection` has to start with the property `id` like `[{"id":` *or* be empty.
- The JSON string in the feature `genres` has to start with the property `id` like `[{"id":` *or* be empty.
- The JSON string in the feature `production_companies` has to start with the property `name` like `[{"name":` *or* be empty.
- The JSON string in the feature `production_countries` has to start with the property `iso_3166_1` like `[{"iso_3166_1":` *or* be empty.
- The JSON string in the feature `spoken_languages` has to start with the property `iso_639_1` like `[{"iso_639_1":` *or* be empty.
- The JSON string in the feature `keywords` has to start with the property `id` like `[{"id":` *or* be empty.
- The JSON string in the feature `cast` has to start with the property `cast_id` like `[{"cast_id":` *or* be empty.
- The JSON string in the feature `crew` has to start with the property `credit_id` like `[{"credit_id":` *or* be empty.

```
Number of detected invalid values in the feature 'belongs_to_collection': 0
Number of detected invalid values in the feature 'genres': 0
Number of detected invalid values in the feature 'production_companies': 0
Number of detected invalid values in the feature 'production_countries': 0
Number of detected invalid values in the feature 'spoken_languages': 0
Number of detected invalid values in the feature 'keywords': 0
Number of detected invalid values in the feature 'cast': 0
Number of detected invalid values in the feature 'crew': 0

Number of detected invalid values in the feature 'belongs_to_collection': 0
```

```
Number of detected invalid values in the feature 'genres': 0
Number of detected invalid values in the feature 'production_companies': 0
Number of detected invalid values in the feature 'production_countries': 0
Number of detected invalid values in the feature 'spoken_languages': 0
Number of detected invalid values in the feature 'keywords': 0
Number of detected invalid values in the feature 'cast': 0
Number of detected invalid values in the feature 'crew': 0
```

3.3 NA Values

The only feature containing `NA` in the train dataset is the movie runtime (6 movies among 7398). We consider the value `0` the same as the `NA` value because having a movie runtime of 0 minutes is impossible. We get the 2 movies from the train dataset where their runtime is `NA` and then we replace their value by a valid one taken from another source (e.g. IMDB).

```
knitr::kable(train[is.na(train$runtime), c("title", "release_date", "runtime")])
```

	title	release_date	runtime
1336		10/29/07	NA
2303	Happy Weekend	3/14/96	NA

```
## Source: https://www.imdb.com/title/tt1107828/
train[1336, "runtime"] <- 130

## Source: https://www.german-films.de/filmarchive/browse-archive/view/detail/film/happy-weekend/index.
train[2303, "runtime"] <- 94

knitr::kable(test[is.na(test$runtime), c("title", "release_date", "runtime")])
```

	title	release_date	runtime
3244	La caliente niña Julietta	3/20/81	NA
4490	Pancho, el perro millonario	6/6/14	NA
4633	Nunca en horas de clase	11/3/78	NA
6818	Miesten välisiä keskusteluja	1/4/13	NA

```
## Source: https://www.imdb.com/title/tt0082131/
test[244, "runtime"] <- 93

## Source: https://www.imdb.com/title/tt3132094/
test[1490, "runtime"] <- 91

## Source: https://www.filmaffinity.com/es/film267495.html
test[1633, "runtime"] <- 100

## Source: https://www.imdb.com/title/tt2192844/
test[3818, "runtime"] <- 90
```

##Empty Values Many of the features having at least one empty value are explained the following ways:

- `belongs_to_collection`: The movie does not belong to a collection of movies.
- `homepage`: The movie does not have a homapage.

- `poster_path`: The movie does not have a film poster. We assume that either the movie has a very low budget and get low revenue or they do not have this information or oversight the poster when inserting the movie in the database.
- `overview`: The movie does not have an overview. We assume that the movie is not enough popular to take the time to give an overview of the movie or it could be an oversight.
- `spoken_languages`: The movie could be a silent movie like Charlie Chaplin or Mr. Bean.
- `production_companies`: A movie self-made could be the reason why a movie does not use a production company.
- `tagline`: The does not have a tagline. We assume that a movie with a great tagline is a movie that we remember. We expect that the movie popularity will be higher. Movie without taglines are less popular and then the revenue could be lower.
- `Keywords`: No keywords describe the movie.
- `genres`: We assume that a movie not associated to genres is non classifed.
- `title`: The empty movie title could be an oversight because the original title is taken instead in some rare cases.

We assume that the empty values in the following features are oversights in the database or they just do not have the information on them:

- `production_countries`: It is impossible that a movie is produced nowhere. It could be possible that there is no information found about the production country.
- `crew`: It is impossible that a movie has been made by itself. It has to have at least one member of the crew like the producer.
- `status`: There are 7396 / 7398 movies that are relased. The 2 other movies do not have a status. In this case, we assume that they are released.
- `release_date`: There are 7397 / 7398 movies that have a release date.

We have to replace this empty date by a real date because we extract parts of the date in the data visualisation and transformation section. From the IMDB source the empty release date is replaced by the following one:

```
test$release_date[test$release_date == ""] <- "5/1/00"
```

3.4 Zero Values

Some movies budget are 0 in both datasets and need to be replaced by a significative value. In such a case, we use the median which gives better results for the train dataset than the mean. We are not using a linear regression model because the budget depends on many variables such as the popularity, the number of members in the crew, the number of characters in the casting, the production company and surely others that we do not have in our dataset.

```
There are 812 movies with 0 of budget.
```

```
The median of non-zero budgets in the train set is: 16000000
```

```
The median of non-zero budgets in the test set is: 17000000
```

3.5 Useless Features

The objective is to remove features that will not be useful for the prediction. It could be features where movies are all in the same category except few movies. It could also be unique values like IDs without `NA` or empty values.

The first one is the movie `status` because it seems that all movies in both dataset are released except only few of them that are not released.

status	number_of_movies
Released	2996

status	number_of_movies
Rumored	4

status	number_of_movies
	2
Post Production	5
Released	4389
Rumored	2

The `imdb_id` and `poster_path` are also useless because they have unique values. Then, we cannot classify them and get insights on the revenue based on them.

Since there are only 8 movies having no overview in the dataset, it is useless to see what is the revenue in function of if the movie has or not an overview. The same logic applies to the `original_title` and `title` features.

Since we want to know which production companies have made the best movies revenue, it becomes usaless to keep the production countries.

We keep the `title` only for the next section for informational purposes. It will be removed at the end of the next section.

4 Data Visualization and Transformation

The first objective is to verify if we reject or not our assumptions by visualizing the data using scatter plots or bar charts. This will help us to know which features are helpful or useless on the predictions.

The second objective is to transform the train dataset in order to obtain a matrix of real and integer values for the predictions purposes. Each time the conclusion according to their chart will show that the feature has a significant impact on the revenue, the string (including the ones containing JSON) feature is transformed to a numeric feature. Otherwise we remove the feature.

4.1 Movie Revenues

Let's get a general picture on the revenue values and verify if the values are normally distributed.

```
   Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
      1    2379808   16807068   66725852   68919204 1519557910
```

title	release_date	revenue
The Avengers	4/25/12	1519557910
Furious 7	4/1/15	1506249360
Avengers: Age of Ultron	4/22/15	1405403694
Beauty and the Beast	3/16/17	1262886337
Transformers: Dark of the Moon	6/28/11	1123746996
The Dark Knight Rises	7/16/12	1084939099
Pirates of the Caribbean: On Stranger Tides	5/14/11	1045713802
Finding Dory	6/16/16	1028570889
Alice in Wonderland	3/3/10	1025491110
Zootopia	2/11/16	1023784195

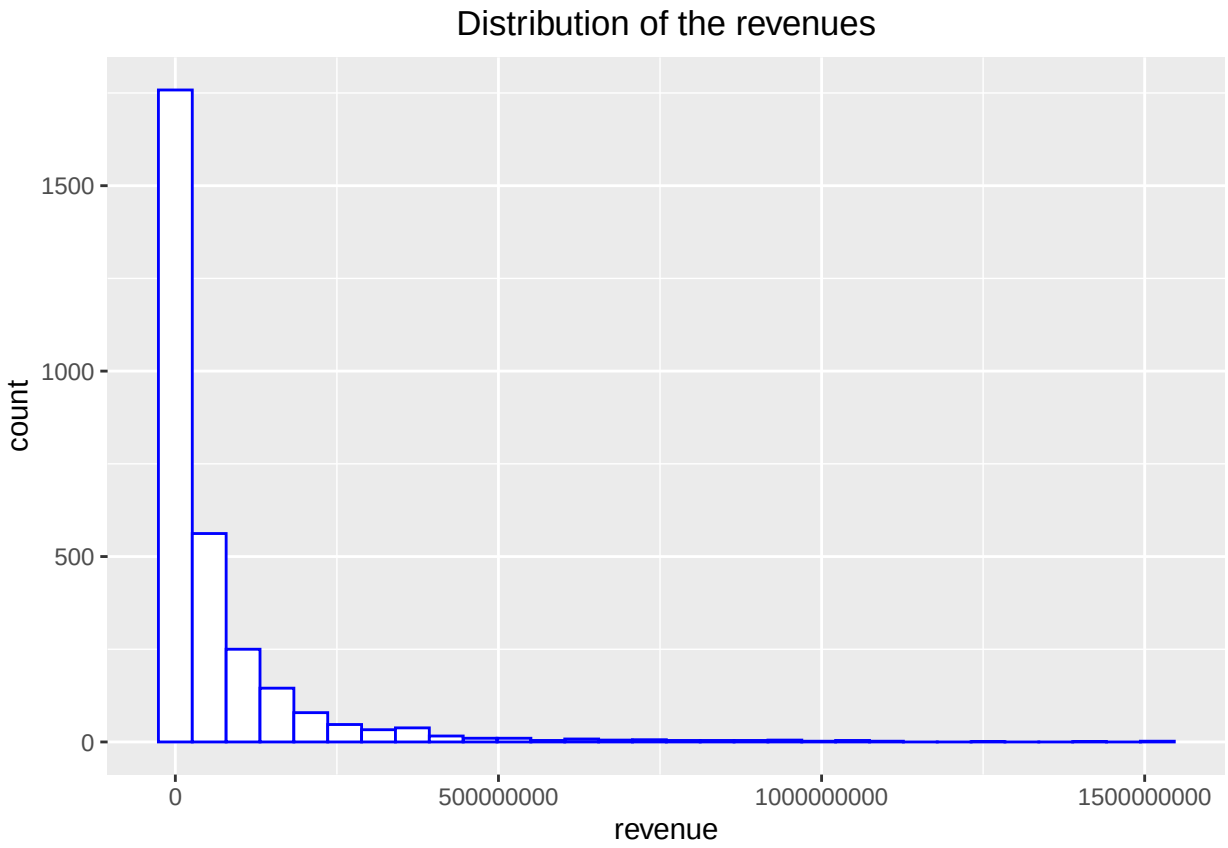
	title	release_date	revenue
2991	East of Eden	3/9/55	5
2992	American Adobo	9/29/01	4
2993	Saamy	5/5/03	3
2994	All at Once	6/5/14	3
2995	Borsalino	5/19/70	3
2996	Tere Naam	8/15/03	2
2997	The Wind in the Willows	10/16/96	1
2998	Mute Witness	9/28/95	1
2999	Missing	1/1/07	1
3000	The Merry Widow	8/26/25	1

There is a huge gap between the mean and the median implying that the revenue is skewed. We also note that the movie `All at Once` generated 3$ of revenue on the Box-Office. According to IMDB, the movie generated 3 514 780$ worldwide. Another example is the indian movie `Saamy` which generated 3$ of revenue. Again, according to IMDB, the movie generated 510 000 000 indian rupees gross (world) which correspond to 7 406 475 US dollars today (according to the exchange rate, the revenue seems to correspond to around 11 million dollars in 2003). The following questions would need to be answered:

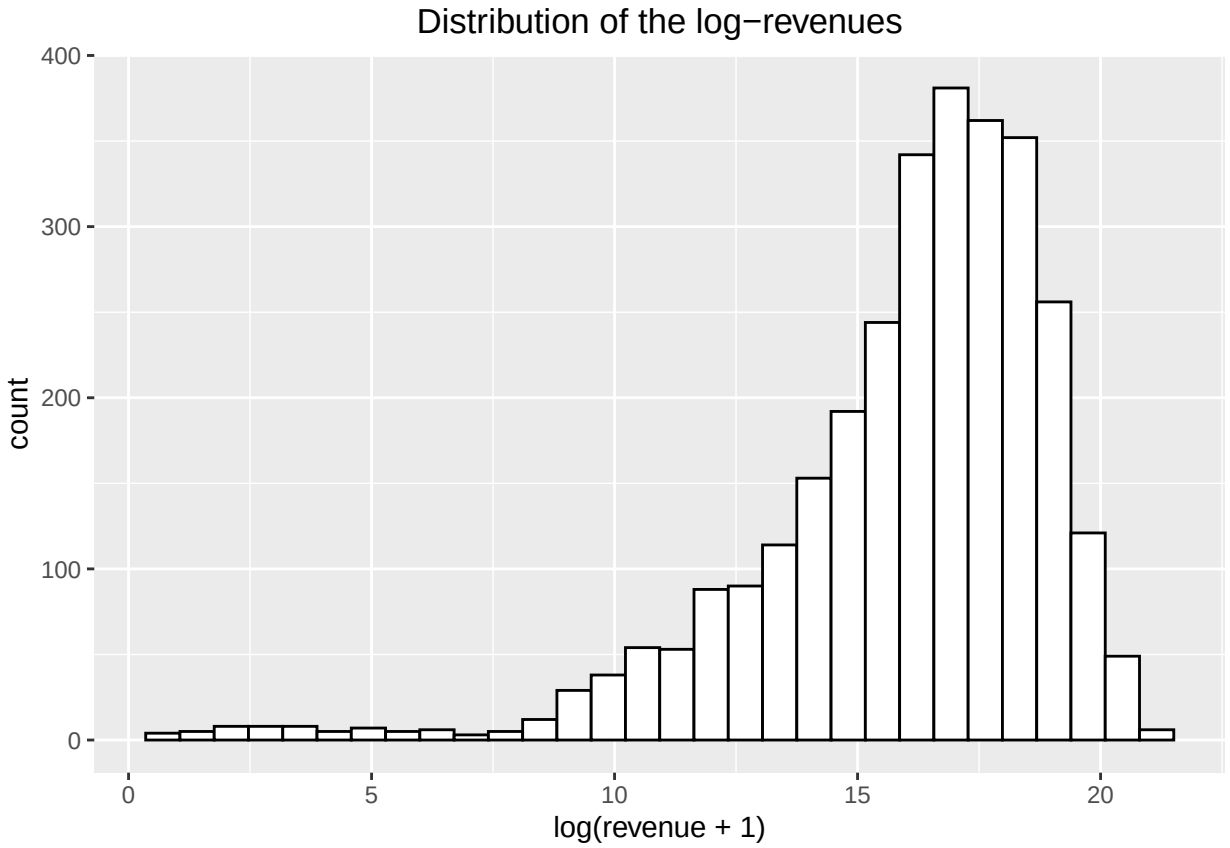
1. Are there incorrect revenues?
2. Are some revenues expressed in million of US dollars?
3. Are all revenues converted in US dollars?
4. Are all revenues converted in function of the money exchange rate when the dataset has been built or when the movie has been released?

Answers to these questions would be very useful to know how to convert them and then get much better predictions. At least knowing money devise would have helped.

Let's see now how the revenue distribution behaves in order to know if the revenue is skewed or not. If so, we will adjust it with the logarithm.



We see that the revenue is highly skewed to the left. Let's see how the log distribution of the revenue behaves:



The log distribution reduced significantly the skewness of the revenue, hence we keep the log revenue instead.

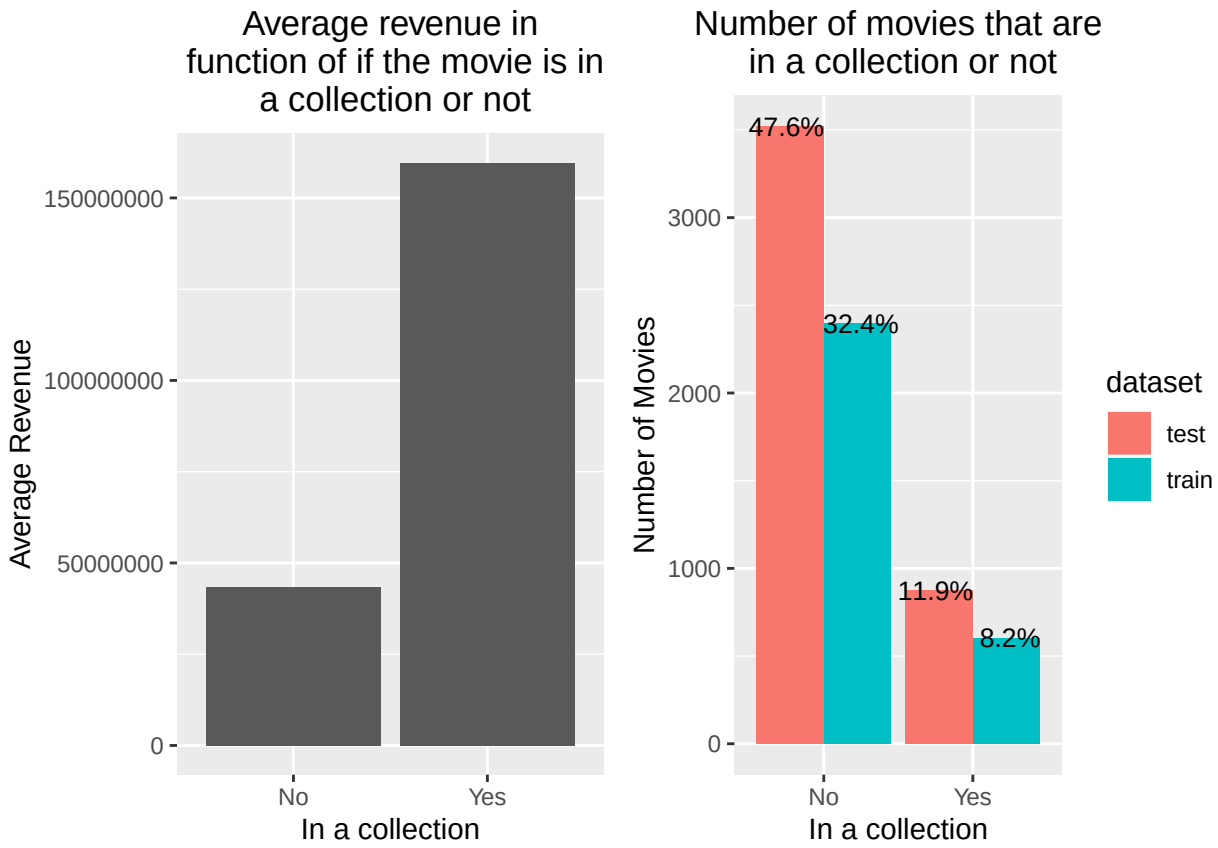
4.2 Movie Collection

The objective is to know if the income of a movie belonging in a collection is higher than a movie that is not in a collection. A movie that belongs in a collection could mean that:

- the movie was enough good to gain a large revenue that the producer(s) decided to make a second movie and so on as a series.
- the same or another producer decided to do a remake of the movie many years after the original one (e.g. Karate kid, Superman).

For both assumptions, we expect a greater revenue for movies in a collection.

We build a bar chart to represent the average log-revenue in function of if the movie is in a collection or not.

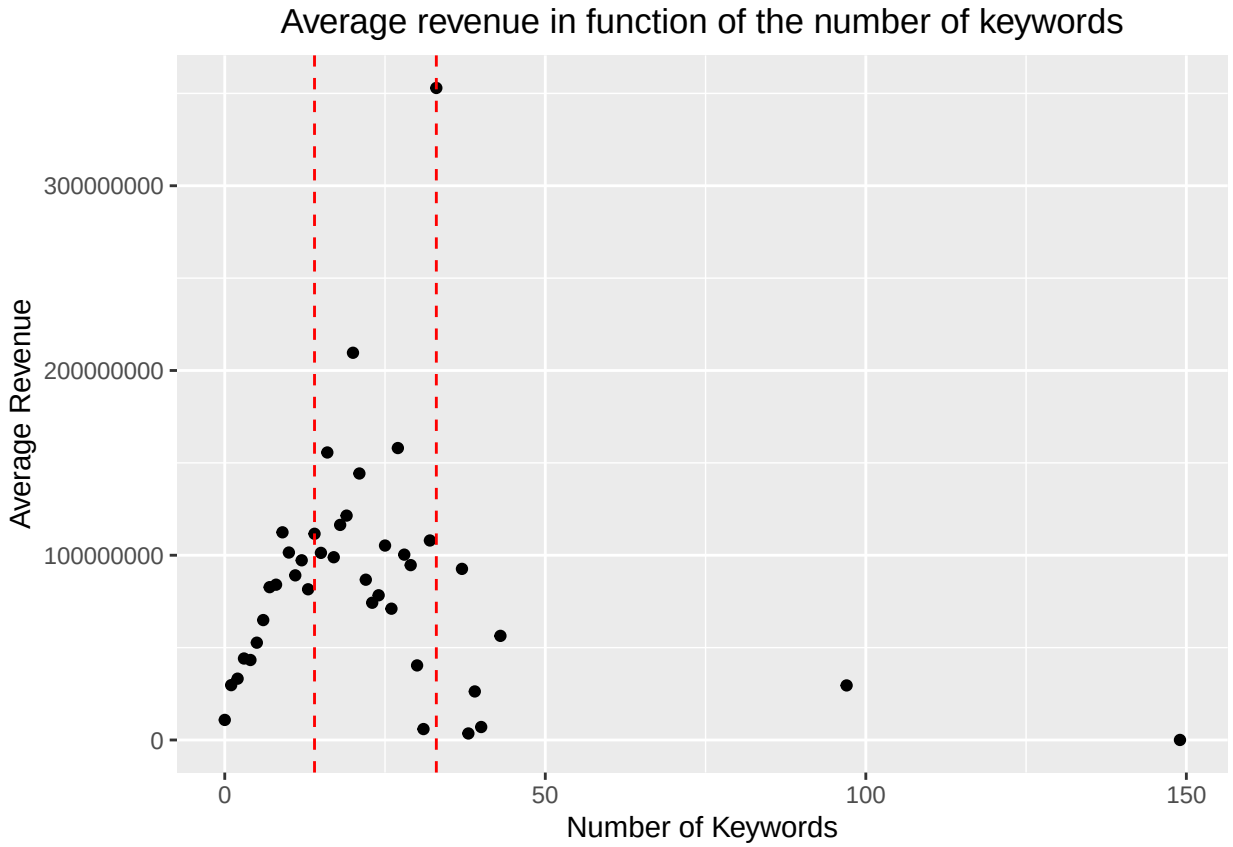


The percentage on the bar chart means the percentage of movies among the test and train sets combined. The difference between movies in a collection and the ones not in a collection is considerable. We see that 20% (which represent 1480 movies) of the movies are in a collection adding to the fact that the difference is considerable.

4.3 Movie Keywords

From the feature `keywords`, we only need to know the number of keywords used in order to facilitate the search of the movie. We assume that a movie associated to many keywords could help increasing its revenue because it is easier to search and find. However, if there are too many keywords, the precision of the search will be lower because the number of movies associated with all of these keywords will increase instead. It depends also on the precision of the keywords. For example, the movie `Casino Royale` keywords `James Bond`, `007`, `Digit` and `casino` are more precise than `bank`, `money` and `terrorist`.

We build a scatter plot to represent the average log-revenue in function of the number of keywords.



Having keywords helps on the movie revenue but having no keywords or too many of them have a negative impact. According to the scatter plot, movies with greatest revenues are located between 14 and 34 keywords. Outside of this range, the revenues are lower.

For example, if a model is based on a decision tree (as a weak learner), a random data point (x, y) would be in one of the following regions in $\mathbb{N} \times \mathbb{R}$ (where x is the number of keywords and y the revenue):

- $\{0, 1, \ldots, 13\} \times \mathbb{R}$
- $\{14, 15, \ldots, 33\} \times \mathbb{R}$
- $\{34, 35, \ldots\} \times \mathbb{R}$

where we expect, for a given random number of keywords x, that $\mathbb{P}(14 \leq x \leq 33) < \mathbb{P}(x \leq 13)$ and $\mathbb{P}(14 \leq x \leq 33) > \mathbb{P}(x \geq 34)$. In other terms, we expect that movies with less than 14 keywords are the most probable in the dataset. Let's calculate the probability for each region:

- The probability that $x \leq 13$ is 0.865.
- The probability that $14 \leq x \leq 33$ is 0.1323333.
- The probability that $x \geq 180$ is 0.0026667.

4.4 Members of the Crew

From the feature `crew`, we want to know if having a director as a member of the crew has significant impacts on the movie revenue. The director has the role to choose the cast and crew members (make generally the decisions), they have to be creative in order to ensure the movie is realized within the budget. However, the director depends on the budget to make the movie and also on his competences.

We build a bar chart in order to represent the average revenue in function of the number of directors.

number_of_directors	mean_log_revenue	mean_revenue	number_of_movies
0	17.296735	281448176.1	16
1	15.954078	64250338.1	2815
2	16.157325	89000934.1	146
3	16.811504	125950717.9	14
4	13.912836	3373166.0	3
5	9.813263	20872362.5	2
7	11.196022	132416.5	2
10	11.516379	100345.0	1
30	8.877940	7171.0	1

According to this bar chart, the difference between crew having at least a director and crew having no directors is not significant except for 5 and more directors. However, without knowing the number of movies per number of directors, we cannot know the reasons behind this.

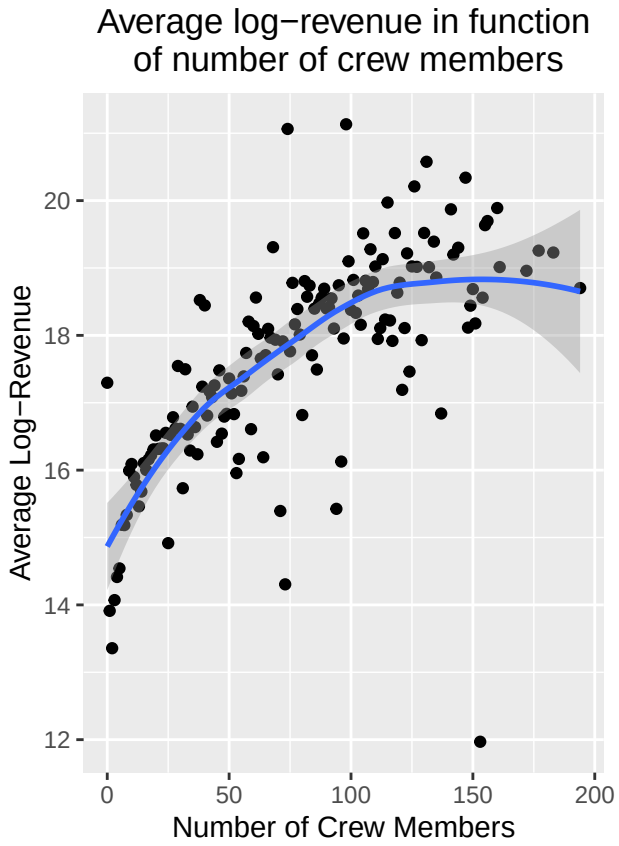
We see that only 16 movies with crews having no directors have been made compared to the others having at least a director. Is this an enough large sample to conclude that the revenue will be much better if the crew has no director, since it represents only 0.53% of the movies in the train dataset? On a mathematical point of view, the sample is not enough large to conclude but on a data driven point of view, those movies show that having or not directors in the crew does not matter. Furthermore, most of movies (93.83% of them) have been made with one director in the crew.

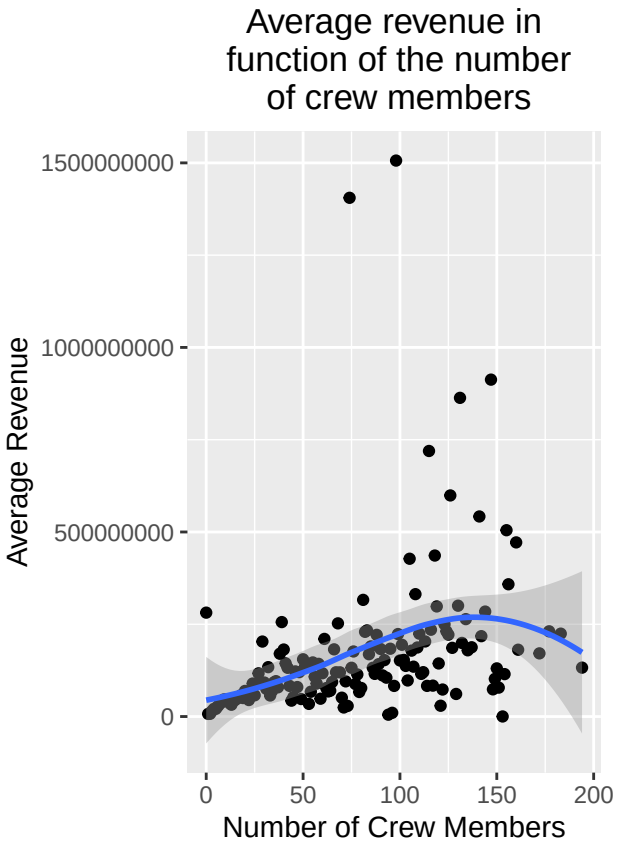
We also expect that a director will not be alone in the crew members. Some of the members have certainly written the movie script in order to be produced. The producer has the role to select the script and ensure the schedule and budget are respected. In order to improve the sequences of a movie and make the movie in its finished state, an editor is important. In summary, we assume that the number of members in the crew has an impact on the revenue. We expect that a small crew will make low revenue movies whereas a large crew will make bigger revenue movies.

```
directors <- NULL
train$number_of_crew_members <- unlist(lapply(train$crew, CountJSONArrayInFeature))
#train$crew <- NULL

test$number_of_crew_members <- unlist(lapply(test$crew, CountJSONArrayInFeature))
#test$crew <- NULL
```

We build a bar chart in order to represent the average revenue in function of the number of crew members.



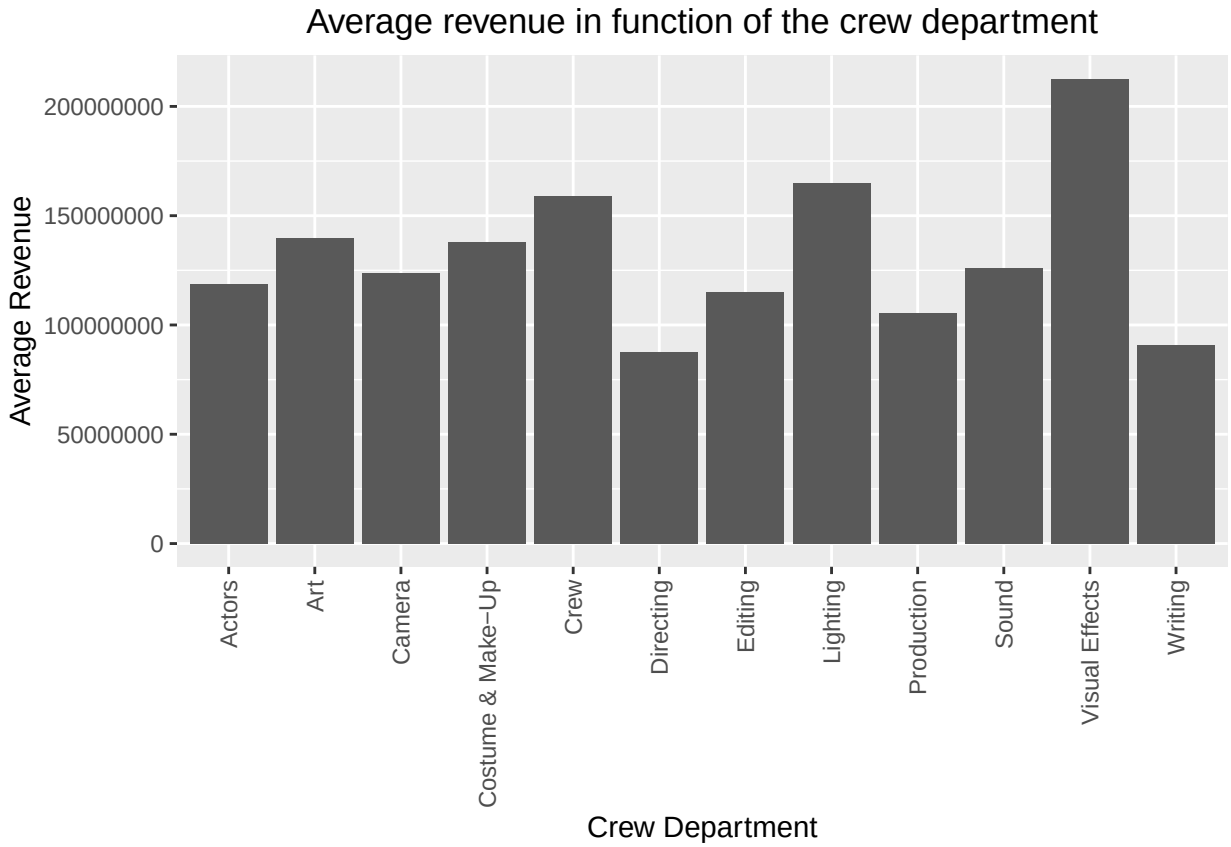


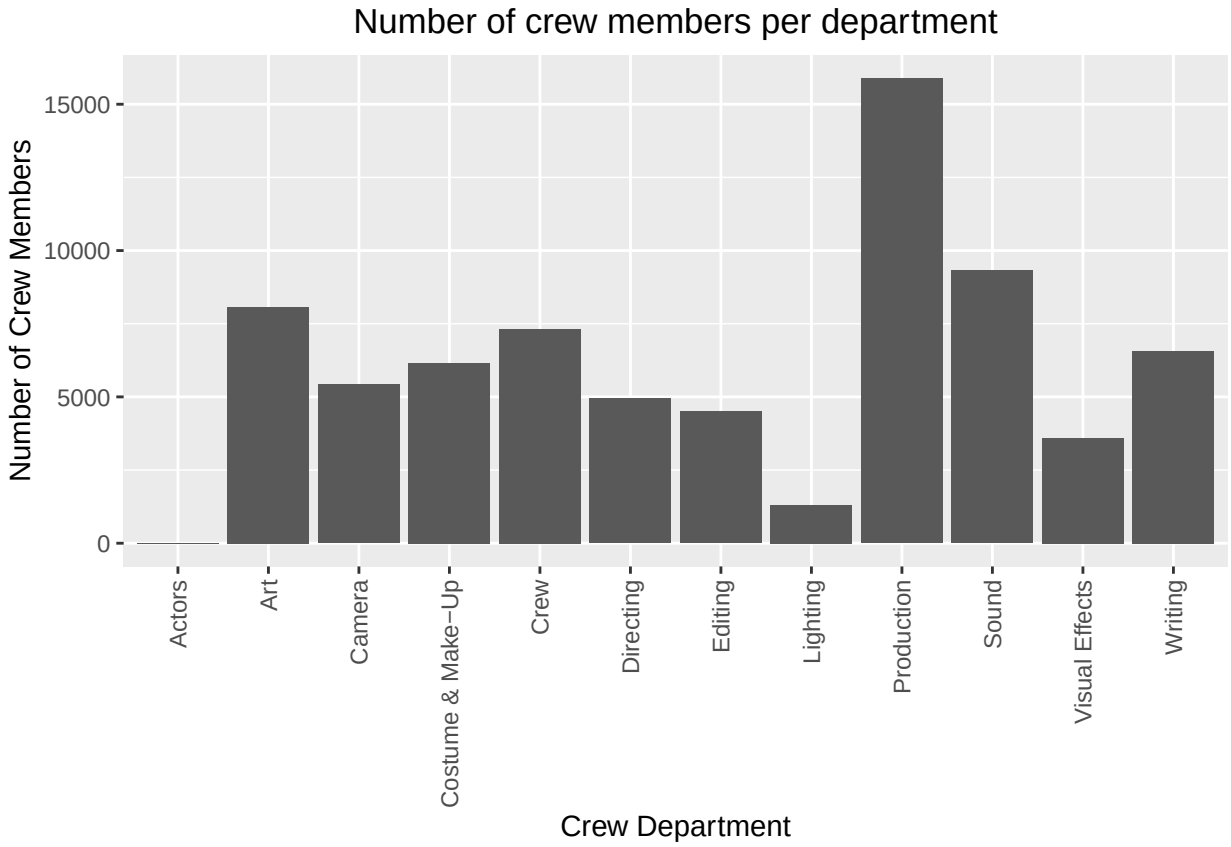
We show also the scatter plot of the average log-revenue in function of the number of members in the crew because it is more explicit that the revenue increases as the number of members increases until around 125 members.

4.5 Crew's Departments

A movie generally needs a large crew depending on the complexity of the special effects, the quality of the movie, costumes and so on. Thus, there are many teams where each of them works on their speciality. Examples of departments can be Production for producers, Art for set decorations and production designers, etc.

We want to know if departments have a correlation with the revenue and which ones are the most useful to generate greatest revenues.

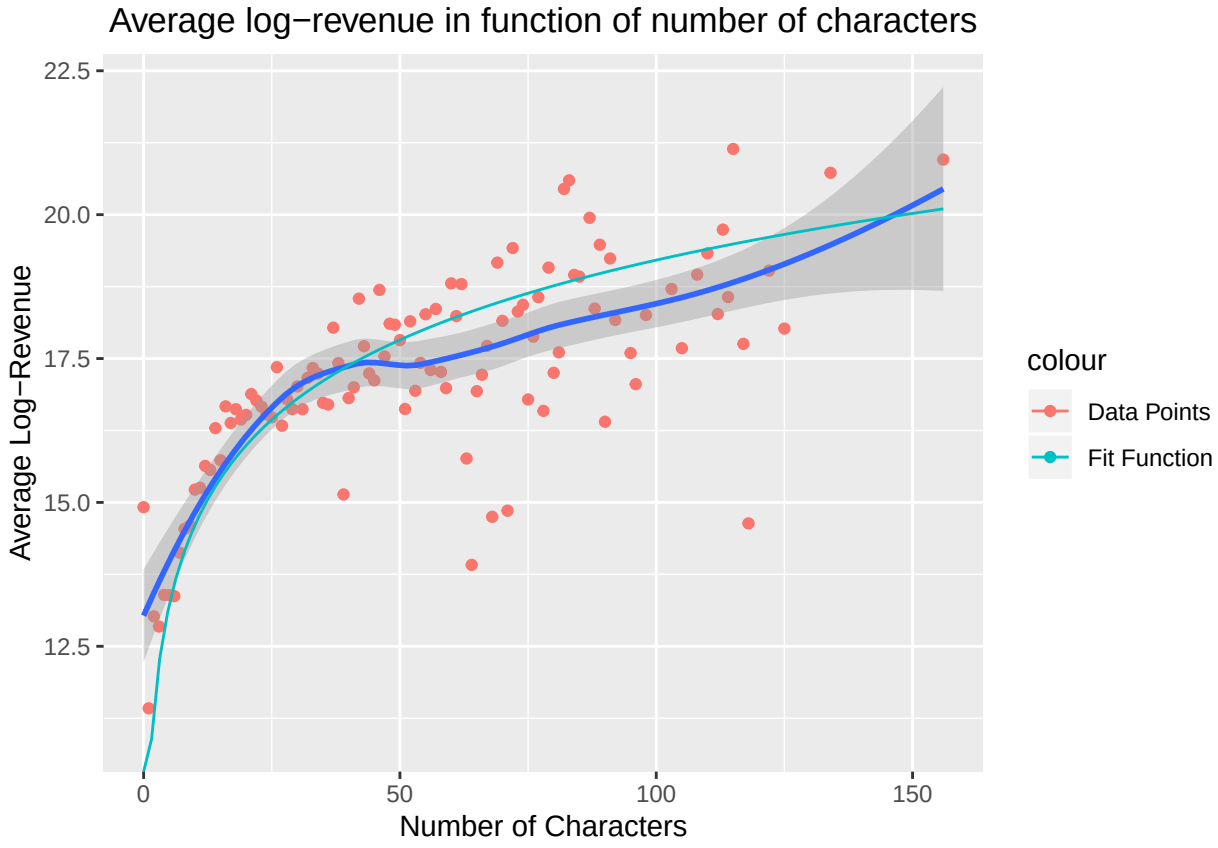

Average revenue in function of the crew department
Average Revenue
200000000
150000000
100000000
50000000
0
Actors
Art
Camera
Costume & Make−Up
Crew
Directing
Editing
Lighting
Production
Sound
Visual Effects
Writing
Crew Department

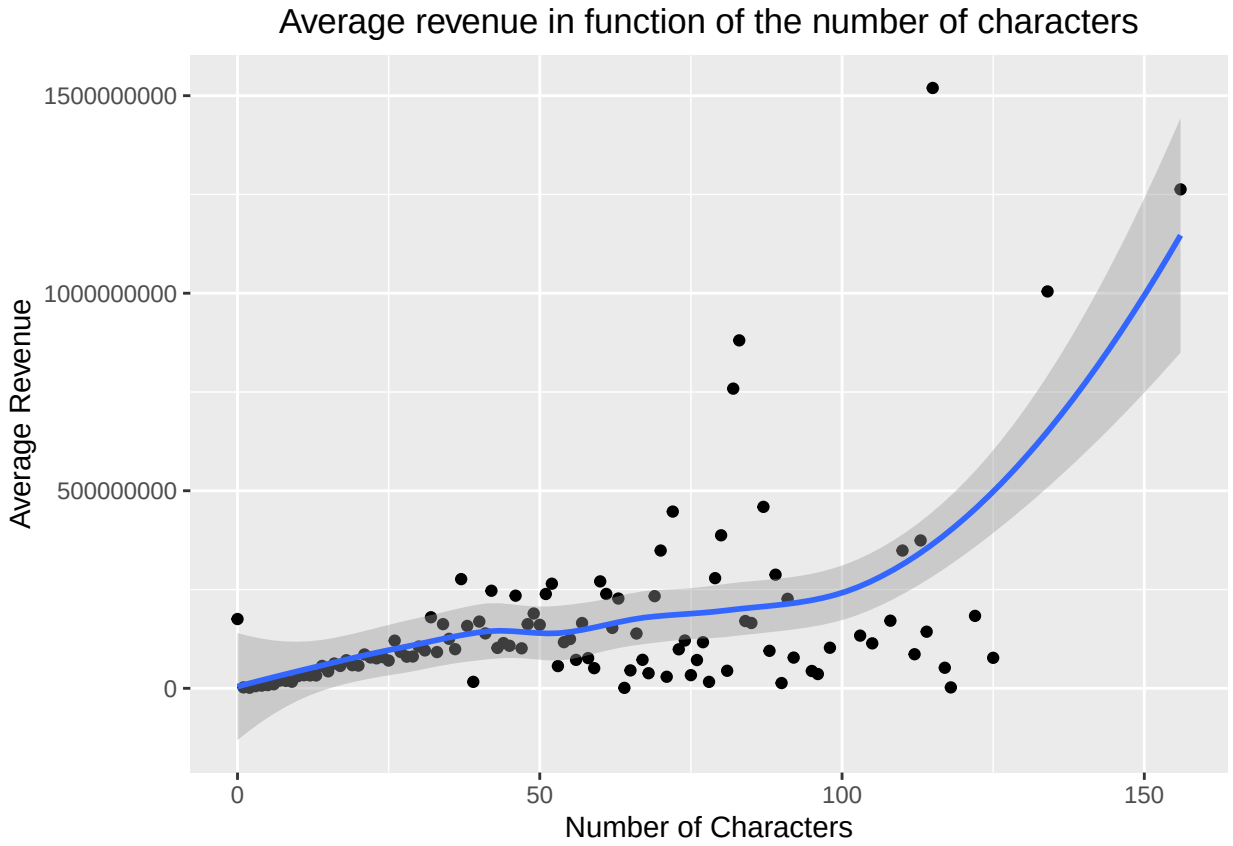

Number of crew members per department
Number of Crew Members
15000
10000
5000
0
Actors
Art
Camera
Costume & Make−Up
Crew
Directing
Editing
Lighting
Production
Sound
Visual Effects
Writing
Crew Department

4.6 Movie Casting

For the feature `cast`, we know that more characters are playing in a movie, more large must be the budget. This is also based on the popularity of the actors and how much money they ask to play in the movie. We assume that most of the time, the revenue is increasing as the number of actors increases.

We build a scatter plot to represent the average log-revenue in function of the number of characters playing in the movie.





We have to consider the budget allowed because it is possible that many actors play in a low budget movie. These actors are paid with a lower salary or they are inexperienced which could justify their low salary. Another reason could be that many of these actors are playing a very short amount of time in the movie. This may explain why sometimes with many characters the revenue is still low. However, the curve

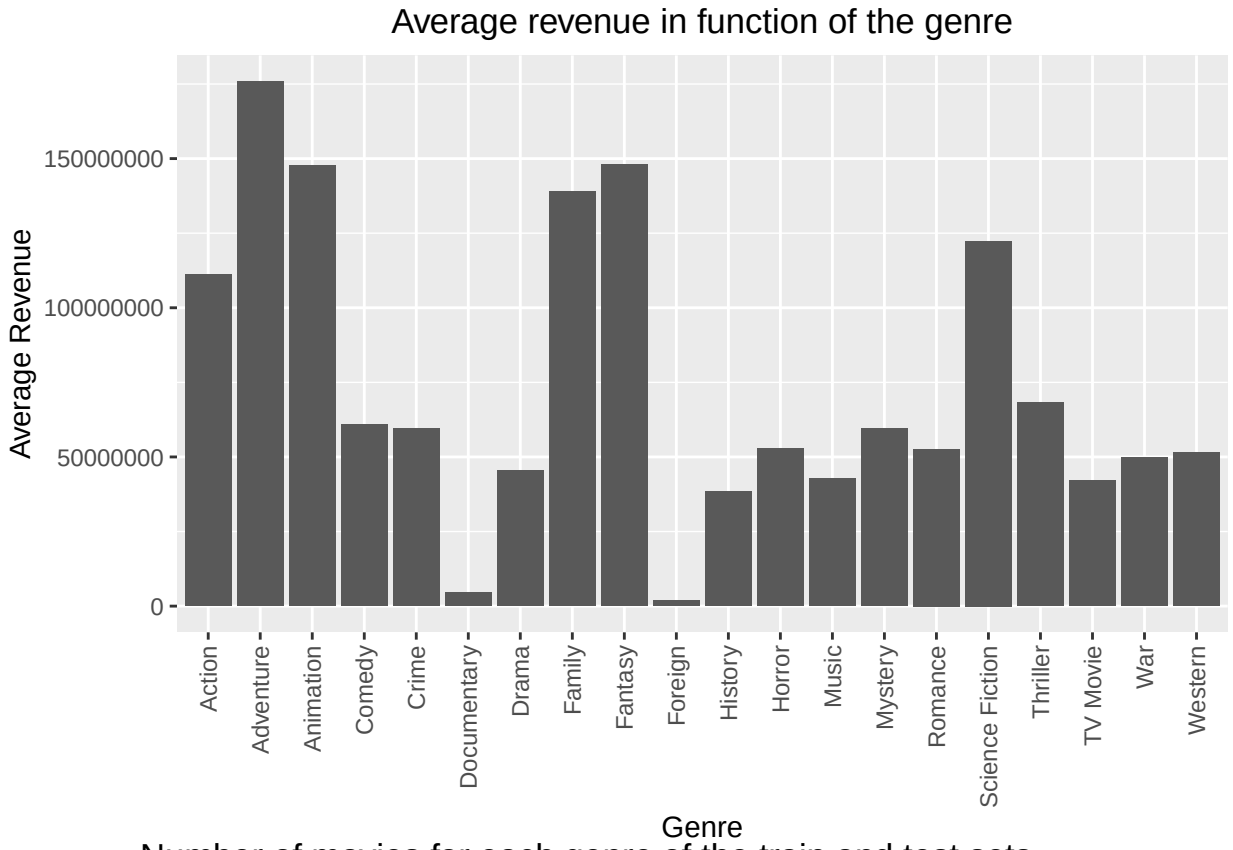
$$f(x) = 2\ln(x) + 10$$

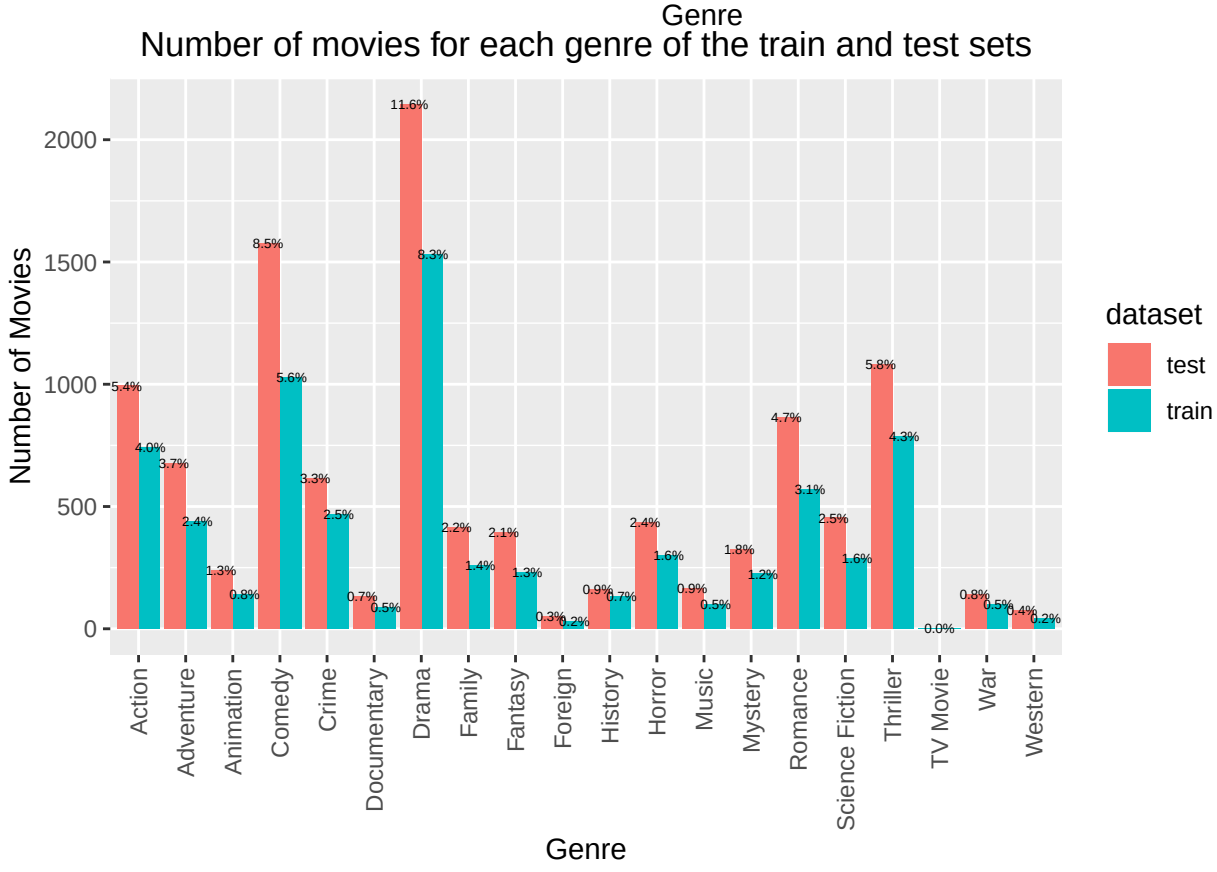
fits well with the data points and could be used to model them on the average log-revenue scatter plot. However, it does not mean that this model will fit well with the test data points.

4.7 Movie Genres

Nowadays, superheros movies (like Avengers, Superman, Iron man, etc.), action/adventure and animation movies are really popular and their incomes are generally big. These movies are classified as science-fiction, action, animation, adventure and fantasy. We assume that those genres of movies generate more revenues than the others. We know that a movie can be classified in one or many genres. However, we cannot assume that the genres in the train set will all be in the test set and vice-versa. Because of that, we cannot base our results on the average revenue, since the revenue does not exist in the test set. Therefore, we will have to estimate it. Knowing that, the first objective is to visualize how the revenue and the number of movies vary in function of the genre. This objective is split in 3 steps:

1. Calculate the average revenue for every genre in the train set.
2. Show a bar chart of the average revenue per genre.
3. Show the number of movies per genre for the train and test sets.


Average revenue in function of the genre
Average Revenue
150000000
100000000
50000000
0
Action
Adventure
Animation
Comedy
Crime
Documentary
Drama
Family
Fantasy
Foreign
History
Horror
Music
Mystery
Romance
Science Fiction
Thriller
TV Movie
War
Western
Genre


Number of movies for each genre of the train and test sets
Number of Movies
2000
1500
1000
500
0
dataset
test
train
Genre

According to the average revenue bar chart, it shows that the `Foreign` and `Documentary` genres generated the lowest average revenues with a significant difference compared to the others. However, the genre `TV Movie`seems to be a general genre (TV and film genres) maybe because they cannot classify the movies. According to the number of movies bar chart, we see that there are only few movies classified `TV Movie` making the average revenue not meaningful. Because of that, the number of movies has to be considered in order to get a better balance with the average revenue.

As we know, the revenue feature does not exist in the test set. In order to be consistent, we use an estimator of the revenue to know which genre has the best average revenue. The second objective is to keep the genre having the best estimated average revenue per movie. This objective is split in the following steps:

1. Define a scoring function estimating the weighted average revenue for each genre in the dataset.
2. Find features that have a strong correlation with the revenue.
3. Define a linear regression equation estimating the revenue.
4. Calculate the score for each genre for the train and test sets.
5. Show a bar plot of the genre in function of their score for the train and test sets.
6. Extract and set the genre with the best score for each movie.

We define the score function as the weighted estimated average revenue of a genre where the weight is the number of movies classified under a genre. In other terms, we have:

$$\begin{aligned} G_i(\hat{y}_i) &= \left(\frac{1}{k_i} \sum_{j=1}^{k_i} \hat{y}_{i,j} \right) \frac{k_i}{n_m} \\ &= \frac{1}{n_m} \sum_{j=1}^{k_i} \hat{y}_{i,j} \end{aligned}$$

where:

- k_i is the number of movies classified under the genre i
- $\hat{y}_{i,j}$ is the estimated revenue of the movie j in the genre i
- n_m is the number of movies in the dataset.

The estimation of the revenue is based on the budget because we know that the correlation between the revenue and the budget is strong (see Linear Regression Model section). We use the Pearson correlation test to find the correlation coefficient between the budget and the revenue. The correlation coefficient is defined by

$$r(x, y) = \frac{\sum_{i=1}^{n_m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n_m}(x_i - \bar{x})^2 \sum_{i=1}^{n_m}(y_i - \bar{y})^2}}$$

where:

- x is the budget vector of the train set
- y is the revenue vector
- $\bar{x}$ is the average budget
- $\bar{y}$ is the average revenue

```
Pearson Correlation Coefficient: 0.7470827


    Pearson's product-moment correlation

data:  train$budget and train$revenue
t = 61.537, df = 2998, p-value < 0.00000000000000022
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7308354 0.7624838
sample estimates:
      cor
```

```
0.7470827
```

The Pearson correlation test shows that our assumption holds, since the p-value is less than 0.05 and then rejecting the null hypothesis that $r(x, y) = 0$. Note that t is given by the formula

$$t = \frac{r\sqrt{n_m - 2}}{\sqrt{1 - r^2}}.$$

In order to get the p-value p, we have to calculate the probability $\mathbb{P}(T > t)$ using the Student's t-distribution where $T \sim t(n_m - 2)$. Using a confidence level of 95% with the t-distribution table and a degree of freedom of 2998, we obtain $T = 1.96$.

We estimate the revenue in function of the budget x with the linear regression equation defined as

$$\hat{y} = \beta_0 + \beta_1 x$$

where we have to find the coefficients $\beta_0, \beta_1 \in \mathbb{R}$.

```
Call:
lm(formula = train$revenue ~ train$budget)

Residuals:
       Min          1Q      Median          3Q         Max
-477294271  -34415553   -9755062   14440968  960853742

Coefficients:
                    Estimate      Std. Error t value             Pr(>|t|)
(Intercept)  -12091025.22992   2104131.29080  -5.746           0.00000001
train$budget         2.93414         0.04768  61.537 < 0.0000000000000002

(Intercept)  ***
train$budget ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 91440000 on 2998 degrees of freedom
Multiple R-squared:  0.5581,    Adjusted R-squared:  0.558
F-statistic:  3787 on 1 and 2998 DF,  p-value: < 0.00000000000000022
```

We found that β_0 is -12091025.2299178 and β_1 is 2.9341402. Substituting in $G_i(\hat{y}_i)$ we obtain the scoring function

$$\begin{aligned} G_i(x_i) &= \frac{1}{n_m} \sum_{j=1}^{k_i} (\beta_0 + \beta_1 x_{i,j}) \\ &= \frac{k_i \beta_0 + \beta_1 \sum_{j=1}^{k_i} x_{i,j}}{n_m}. \end{aligned}$$

name	score	id
Action	28309180.330	1
Adventure	22464879.679	2
Animation	5816394.702	3
Comedy	19703980.216	4
Crime	10449757.412	5
Documentary	772928.233	6
Drama	25868343.734	7

name	score	id
Family	10697366.957	8
Fantasy	10534587.098	9
Foreign	211834.512	10
History	2822244.962	11
Horror	4172061.321	12
Music	1368598.856	13
Mystery	4704819.970	14
Romance	8966102.835	15
Science Fiction	11911310.254	16
Thriller	19613628.305	17
TV Movie	859.892	18
War	2477443.965	19
Western	816228.680	20

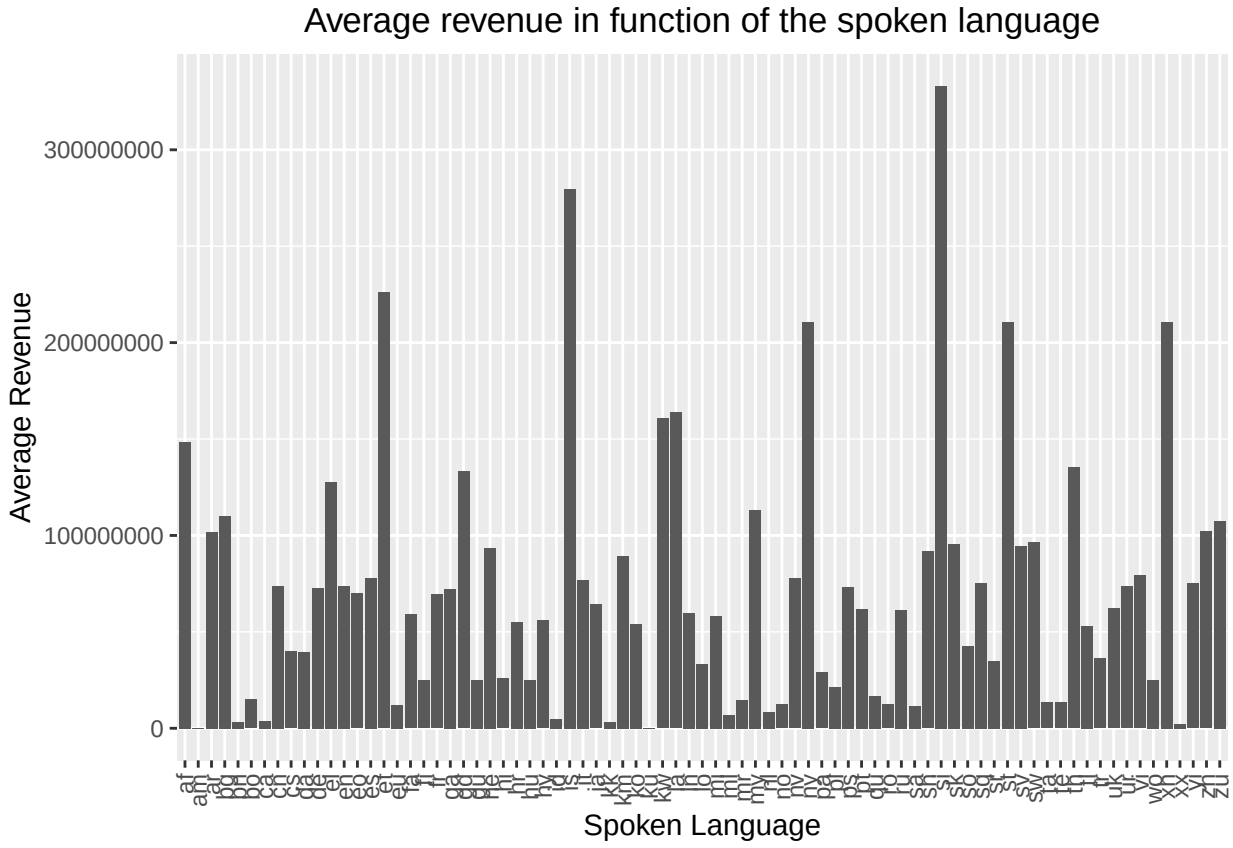
However, we extract the best average revenue per genre and keep it in our dataset because we got significative insights on the lowest and biggest average revenues.

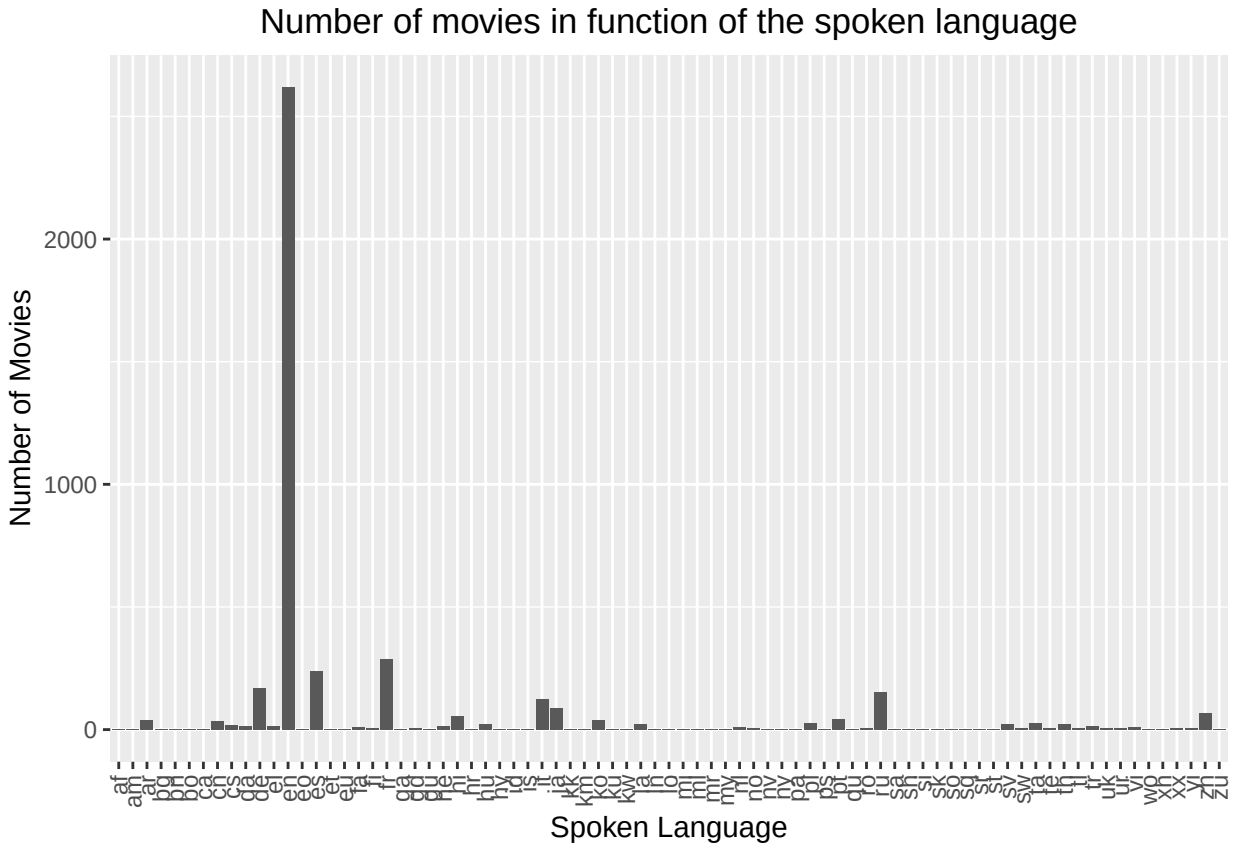
4.8 Spoken Languages

One can assume that movies in which actors speaks in English have the best revenue because it is a well-known language around the world. However, if a spoken language (e.g. Sinhalese) is used in only one movie and another spoken language is used in this movie (e.g. English) where the revenue is huge, the average revenue associated to the Sinhalese spoken language will be huge. In such case, the results could be biaised and mislead to predict the revenue.

The objective is to visualize the average revenue for every spoken language. Here are the steps to achieve the objective:

1. Extract the spoken language feature `iso_639_1` for every movie in the dataset with the revenue.
2. Group the data frame by spoken language where we keep the average revenue.
3. Show a bar chart of the average revenue in function of the spoken language.
4. Show a bar chart of the number of movies in function of the spoken language.





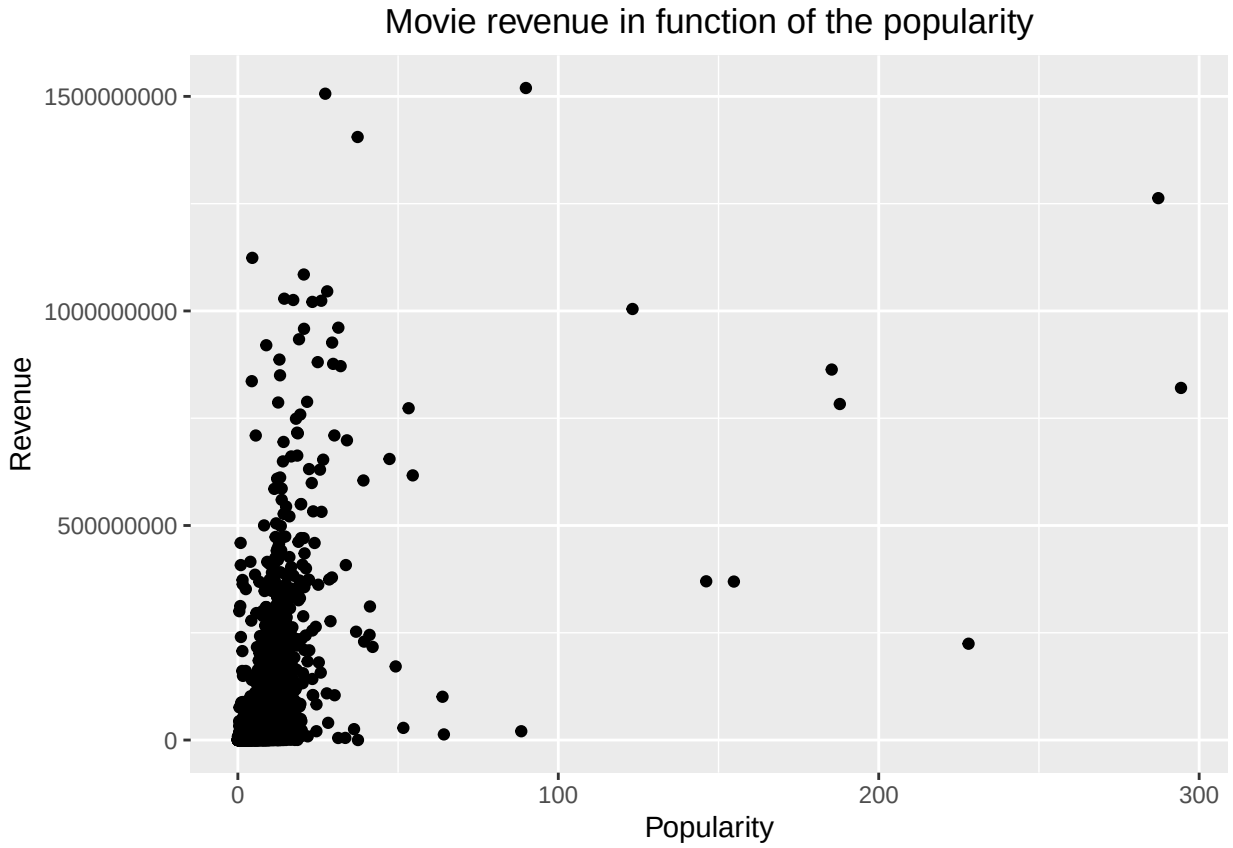

```
Percentage of the number of movies spoken in English: 87.27 %
```

We will not consider this feature to help calculating the predictions of the revenue. The reason is that a strong majority of movies are spoken in English which will not give useful insights.

4.9 Movie Popularity

The popularity is a value that gets updated daily and takes a number of things into account like views, number of user ratings/watchlist/favourite additions and release date (Source: https://www.themoviedb.org/talk/56e614a2c3a3685aa4008121).

One can assume that more a movie is popular, more its revenue is greater. However, a movie could be viewed by a large number of people where most of them do not like the movie enough to pay to see it. People could add the movie in the watchlist and never watch it. If we go further, a movie could be very popular because of its great trailer or ads. What if the trailer contains the best parts of the movie just to attract as many people as possible? When people will watch it, they will be dissapointed because they were expecting much more of the movie. In conclusion, we assume that the popularity should be useful on the prediction of the movie revenue. Let's see a scatter plot to show if our assumption holds or not.



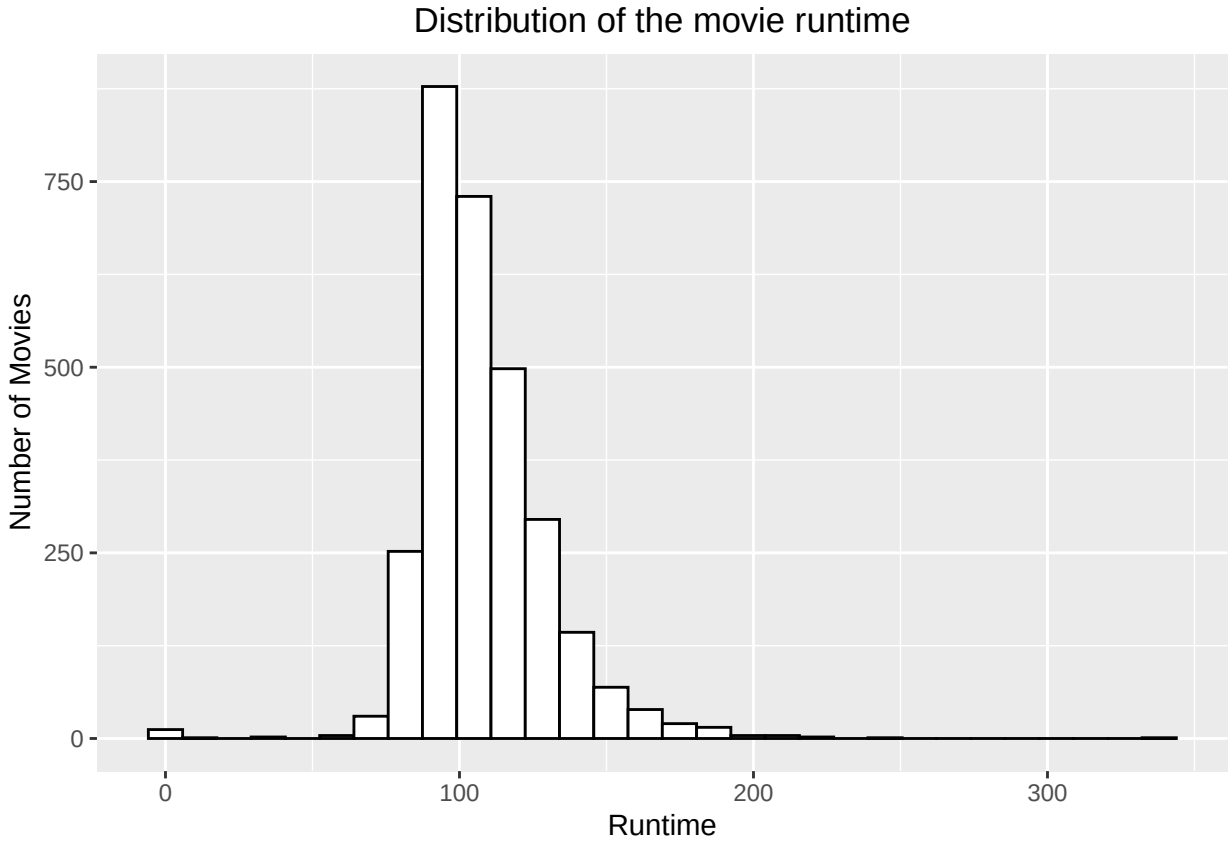
According to this scatter plot, we see that most of the movies have better revenues when the popularity increases. Let's see a top 10 of movies with the greatest revenue and their popularity.

	title	popularity	revenue
1127	The Avengers	89.887648	1519557910
1762	Furious 7	27.275687	1506249360
2771	Avengers: Age of Ultron	37.379420	1405403694
685	Beauty and the Beast	287.253654	1262886337
2323	Transformers: Dark of the Moon	4.503505	1123746996
907	The Dark Knight Rises	20.582580	1084939099
2136	Pirates of the Caribbean: On Stranger Tides	27.887720	1045713802
2563	Finding Dory	14.477677	1028570889
882	Alice in Wonderland	17.285093	1025491110
735	Zootopia	26.024868	1023784195

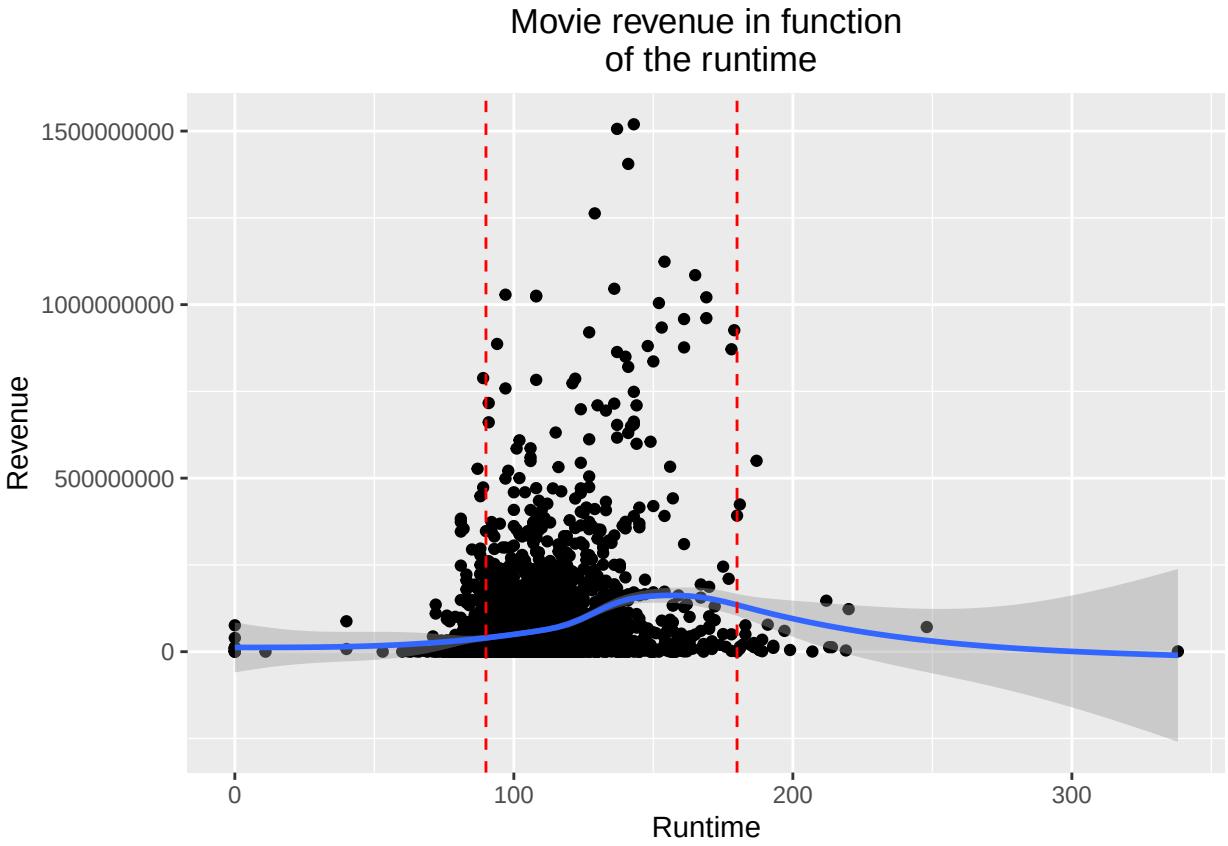
Our assumption seems to hold, but the popularity could be misleading at few points on the prediction of the revenue because there are outliers. A good example of this is the movie *Beauty and the Beast* with a revenue of 1262886337 (ranked 4th among the 3000 movies) and a great popularity of 287.253654 whereas the movie *Transformers: Dark of the Moon* generated a revenue of 1123746996 (ranked 5th among the 3000 movies) and has a low popularity of 4.503505. Without knowing precisely on what variables the popularity is based from our dataset, we cannot explain why there are outliers.

4.10 Movie Runtime

Generally, a movie should not be too long or too short. A too long movie may become boring or make people watching the movie in 2 parts or more because it takes too much time in a day to watch the full movie. For too short movies, the story could be shorten where there is no conclusion, skip some part of the story or not long enough because the movie is so good. For both cases, we expect that the imapcts on the revenue is negative. Before showing the revenue in function of the runtime, we have to verify if the runtime is skewed or not.



The runtime distribution is acceptable as it is. Let's see now if our assumptions hold.



According to this scatter plot, movies with a runtime less than 90 minutes or greater than 180 minutes obtains lower revenues. For example, if a model is based on a decision tree (as a weak learner), a random data point (x, y) would be in one of the following regions in $\mathbb{R}^2$ (where x is the runtime and y the revenue):

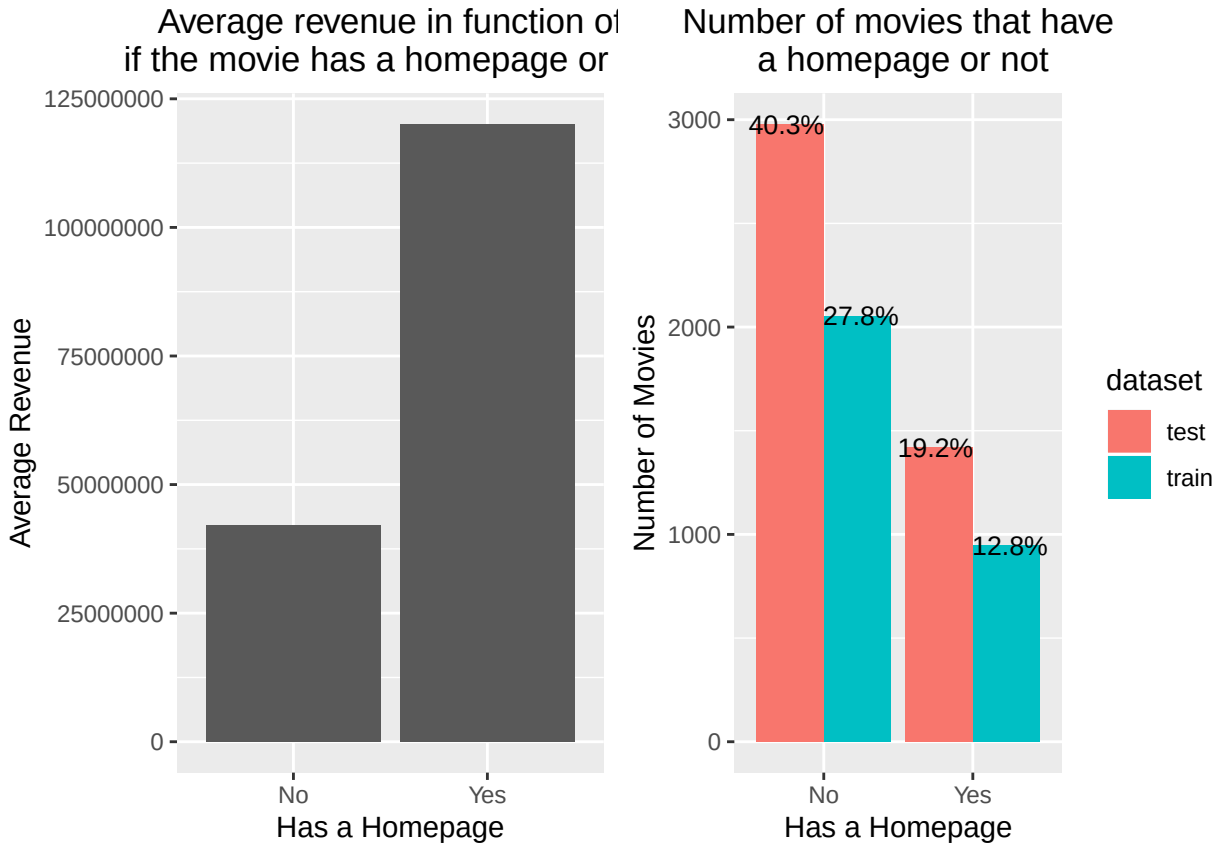
- $[0, 90[\times \mathbb{R}$
- $[90, 180[\times \mathbb{R}$
- $[180, \infty[\times \mathbb{R}$

where we expect, for a given random runtime x, that $\mathbb{P}(90 \leq x < 180) > \mathbb{P}(x < 90)$ and $\mathbb{P}(90 \leq x < 180) > \mathbb{P}(x > 180)$. In other terms, we expect that movies with runtime between 90 and 180 minutes are the most probable in the dataset. Let's calculate the probability for each region:

- The probability that $x < 90$ is 0.1376667.
- The probability that $90 \leq x < 180$ is 0.8526667.
- The probability that $x \geq 180$ is 0.0096667.

4.11 Movie Homepage

Homepage for movies makes them more visible and accessible to people because they can get more details (news, new movies coming up, critics, trailers) on movies and get an overview of the list of movies (e.g. Marvel's movies). This is a good way to attract people and make them watch the movies. For example, if someone wants to see what Disney's movies are coming up soon, he search for Disney's movies and will quickly find the Disney's website.

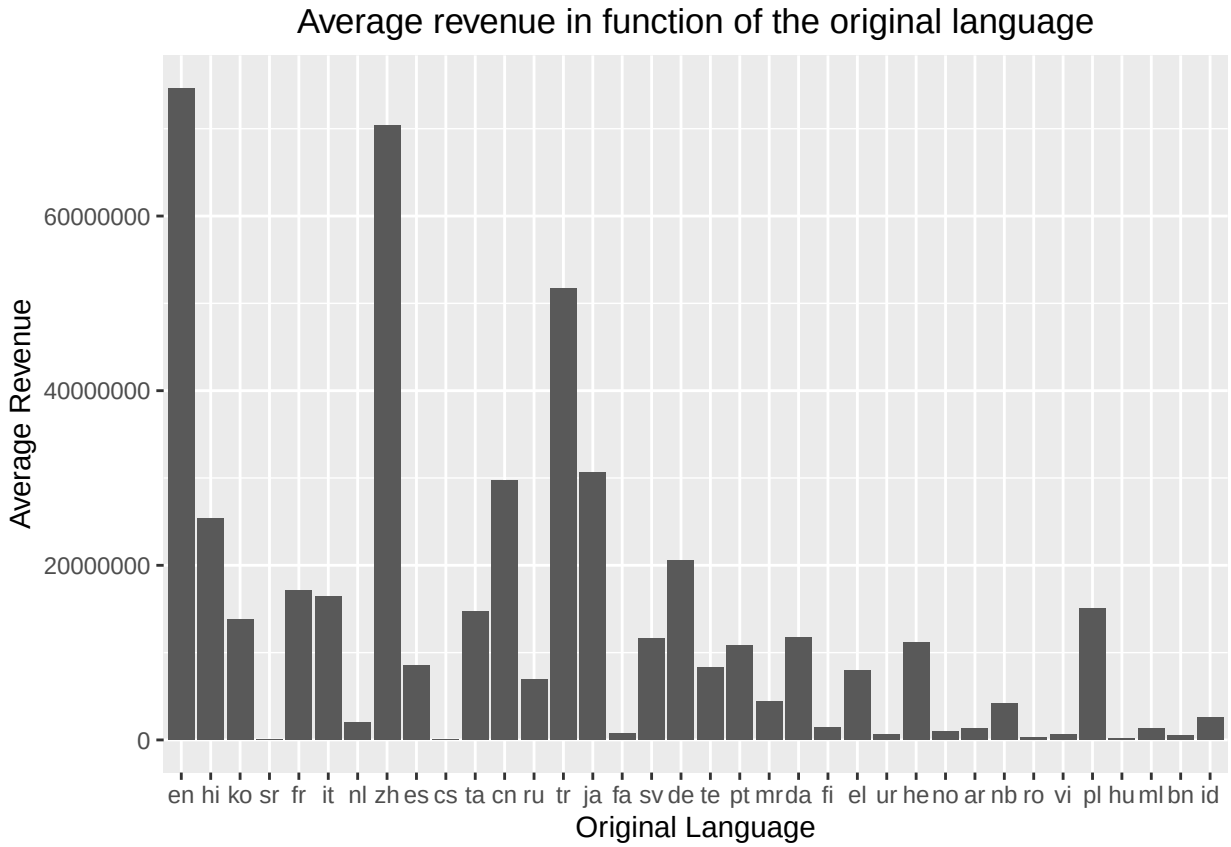


According to this bar chart, the average revenue is bigger for movies having a homepage than the ones that do not. This feature should be helpful on the model prediction.

4.12 Movie Original Language

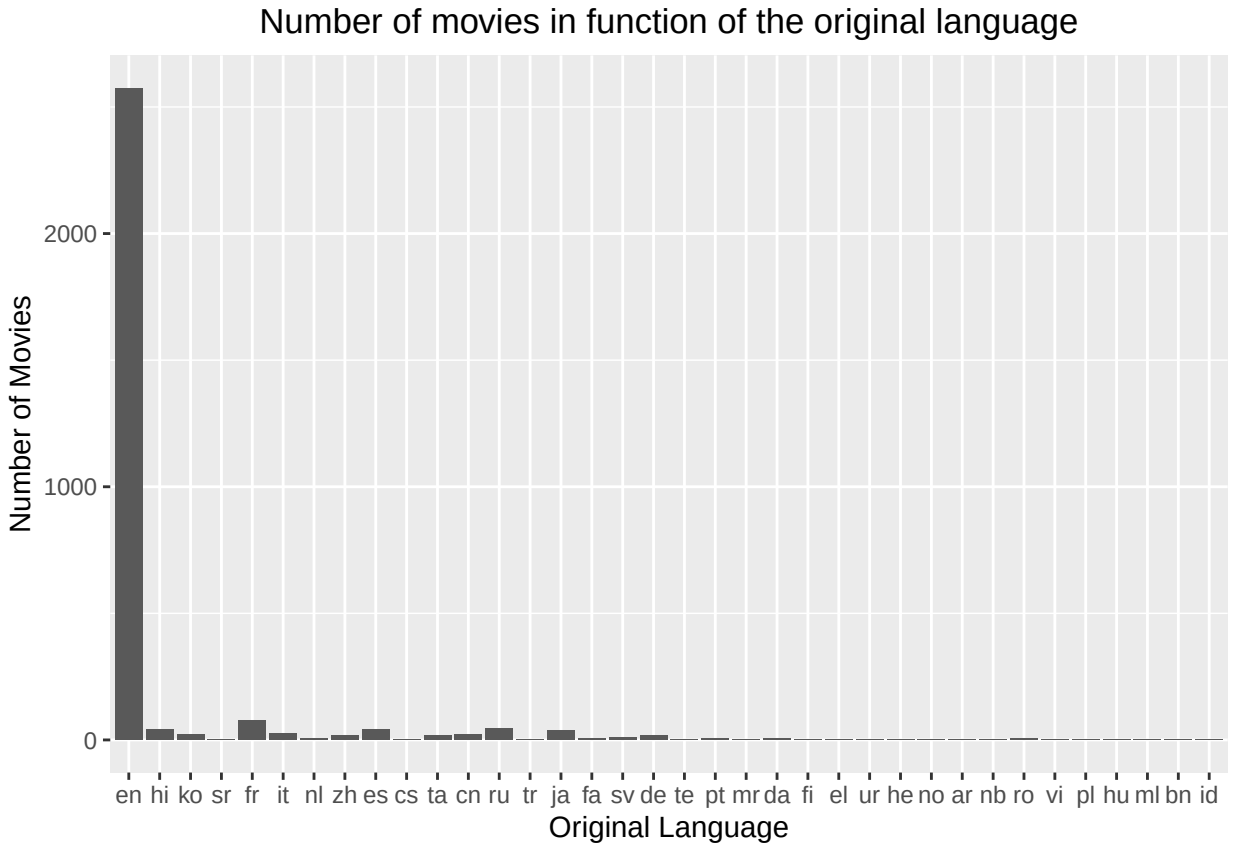
We know that some languages are more spoken around the world than others. One can assume that English language spoken in movies is the most popular and has the best revenue average because it is a well-know language around the world. Movies where the actors speaks foreign languages that are spoken in only a country or in a part of a country will need to be translated and may not be known. We expect that this has a negative impact on the revenue.

However, we saw many good Chinese movies (specially with martial arts) with great Chinese actors (e.g. Jackie Chan, Donnie Yen, Jet Li) that got known around the world. Knowing that, we assume that Chinese movies have generated among the best revenue.



original_language	mean_log_revenue	mean_revenue	number_of_movies
1	16.20347	74665909	2575
8	15.24687	70376369	19
14	15.38403	51663408	3
15	15.81980	30651799	37
12	15.72050	29772885	20
2	15.38370	25346369	42
18	14.58356	20530901	18
5	13.72194	17132566	78
6	14.61032	16415129	24
32	16.52410	15010834	2

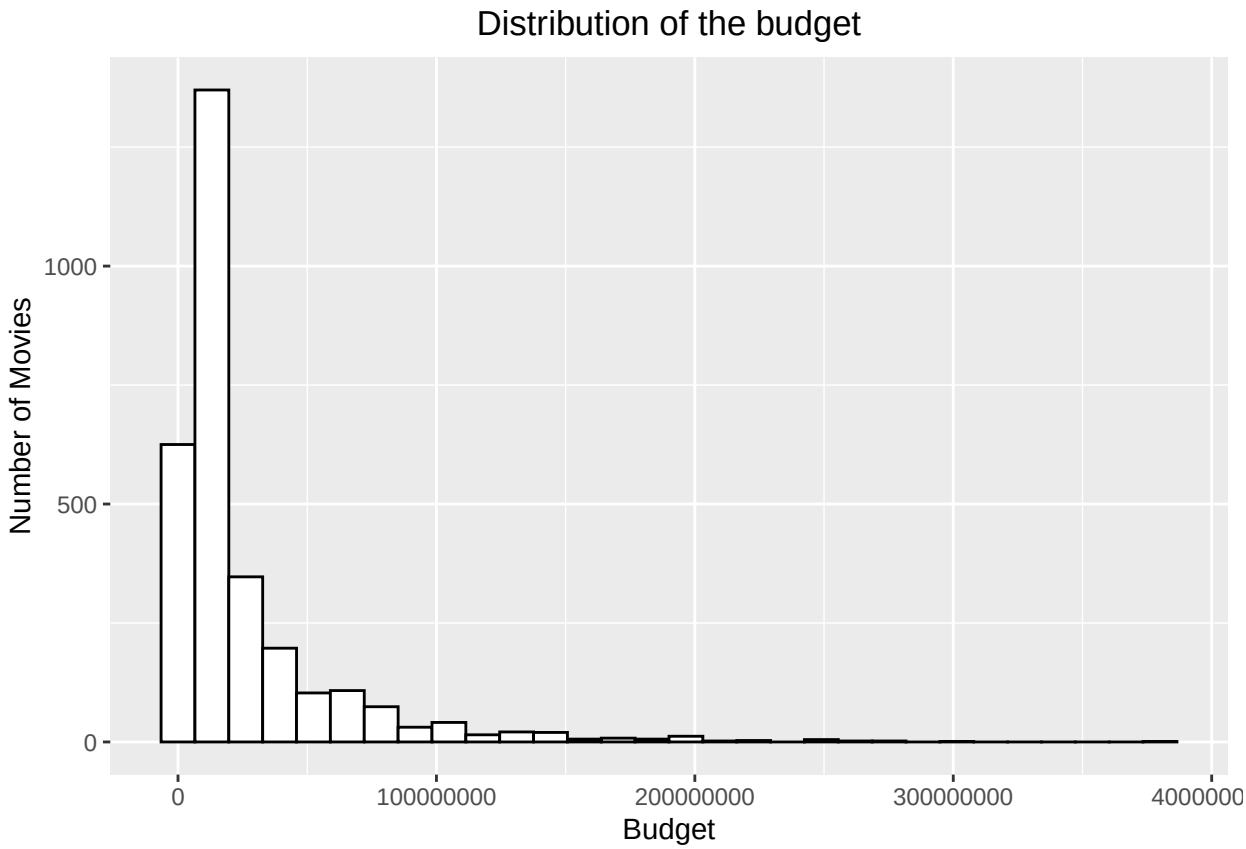
Let's see the number of movies by original language.



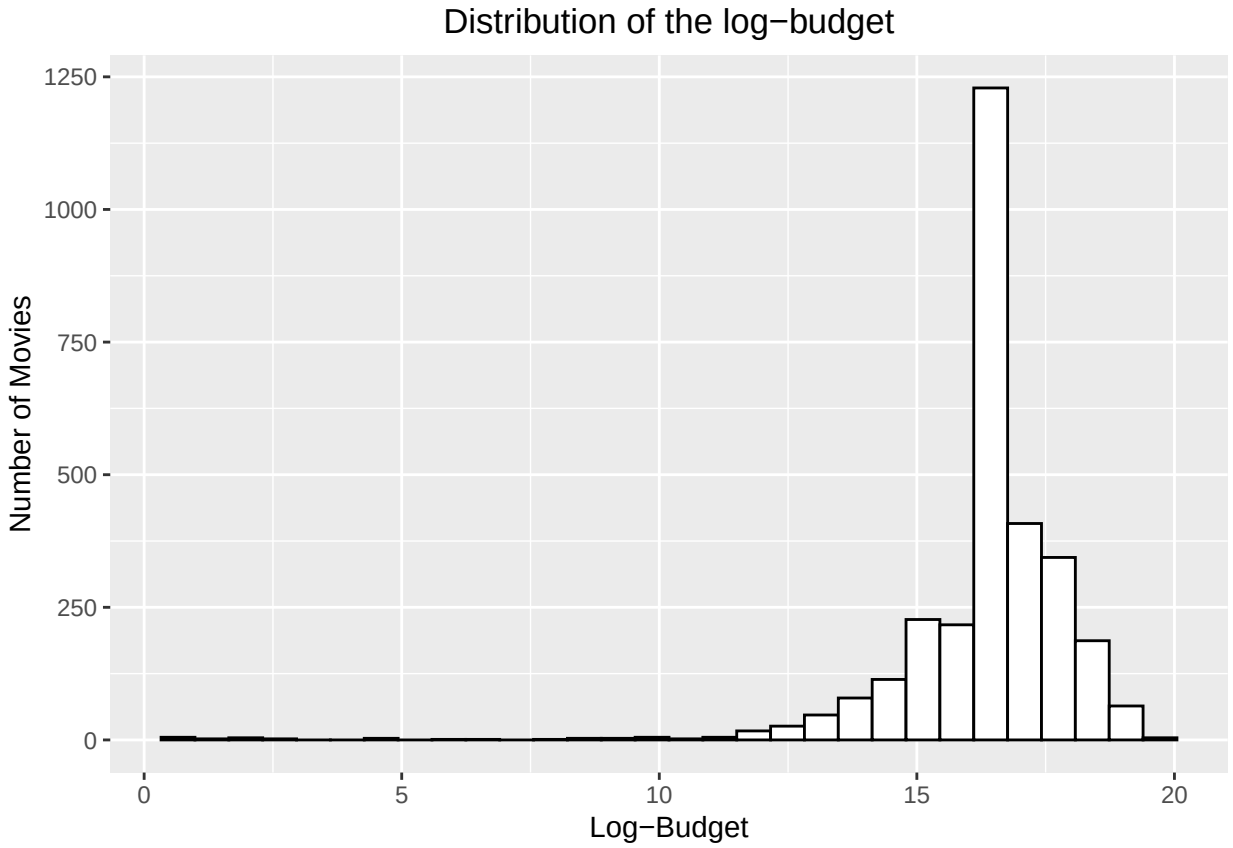
As we expected, English original language in movies dominates and have generated the best average revenue. However, other languages than English are not significative to get strong insights on the revenue. Hence, we will not consider it in our model for predictions.

4.13 Movie Budget

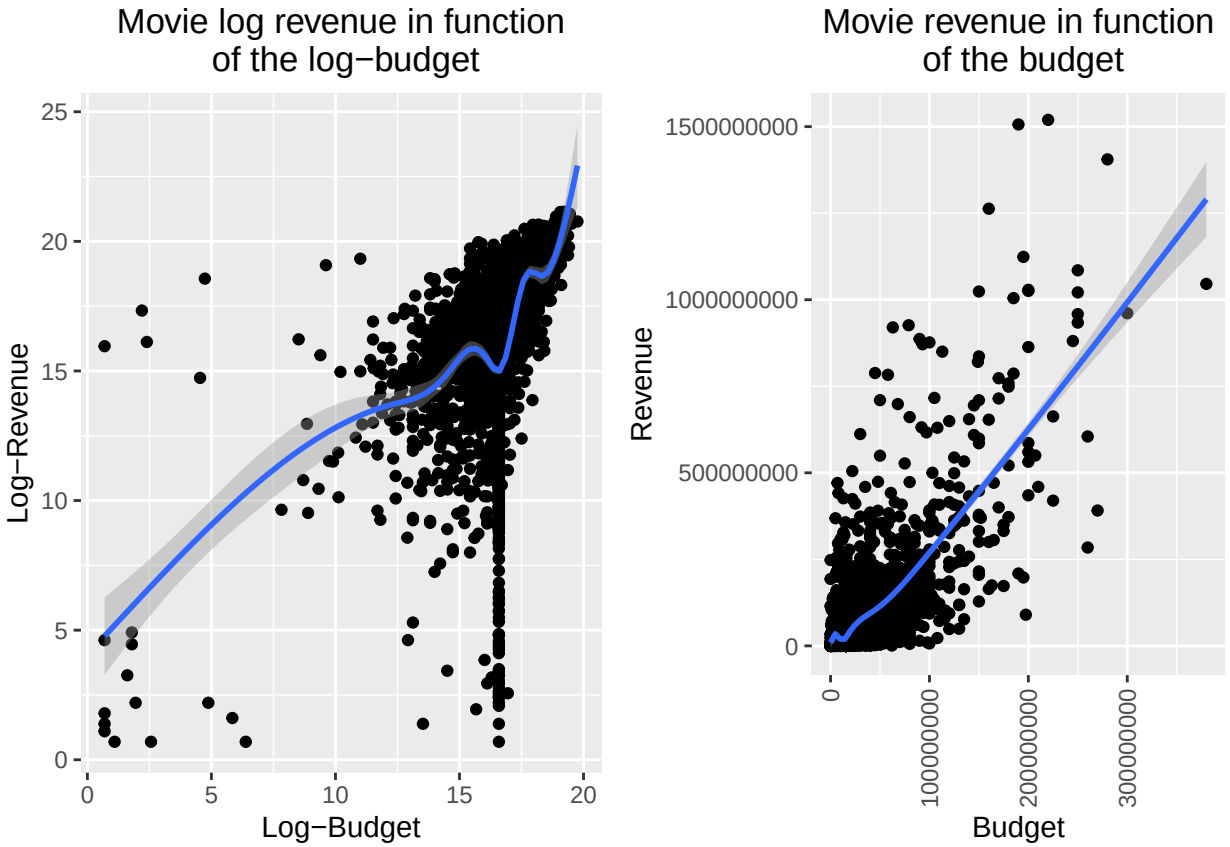
The budget is always an important constraint to consider before making a movie. If the budget allowed is low, we expect that the revenue will be also low but most of the time, it is much greater than the budget. Let's see the distribution of the budget:



The distribution of the budget has a high skewness to the left and need to be adjusted using a log distribution instead.



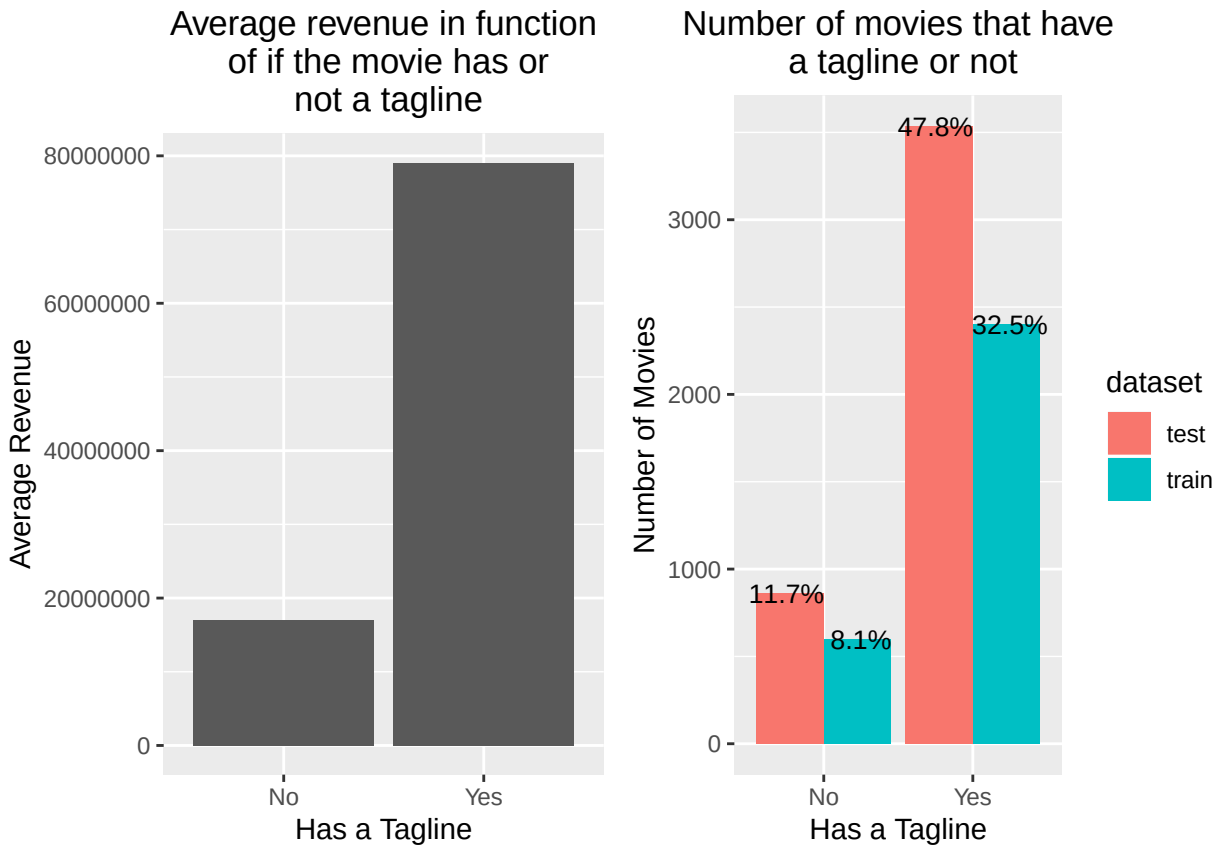
Now that the budget is better adjusted to remove the high skewness, let's see how is the log-revenue in function of the log-budget.



Our assumptions holds most of the time and the scatter plot describes a linear increasing of the revenue in function of the budget. Therefore, the budget matters and has a significant impact on the revenues. Note that the line of points that we can see in the log-budget scatter plot represents the median of the log-budget. Initially, the budget was zero for them and we changed them to the median instead.

4.14 Movie Tagline

A movie having a tagline can help the people remembering that movie because they are catchy and help selling the movie on a marketing point of view. Thus, we assume that movies having no tagline have their average revenue lower than the ones having a tagline.



According to these bar charts, our assumption holds but the difference is not very significative. This feature is then not useful enough to be kept for our model.

4.15 Release Date

We know that making million or billion dollars was harder as we get back many years ago because mainly of the inflation and life cost. Therefore, it makes sense that the revenue was considerably lower in the '80s and older than nowadays. We expect that the revenue will be much lower in the '50s, '60's until the '90s than the 2000s and 2010s.

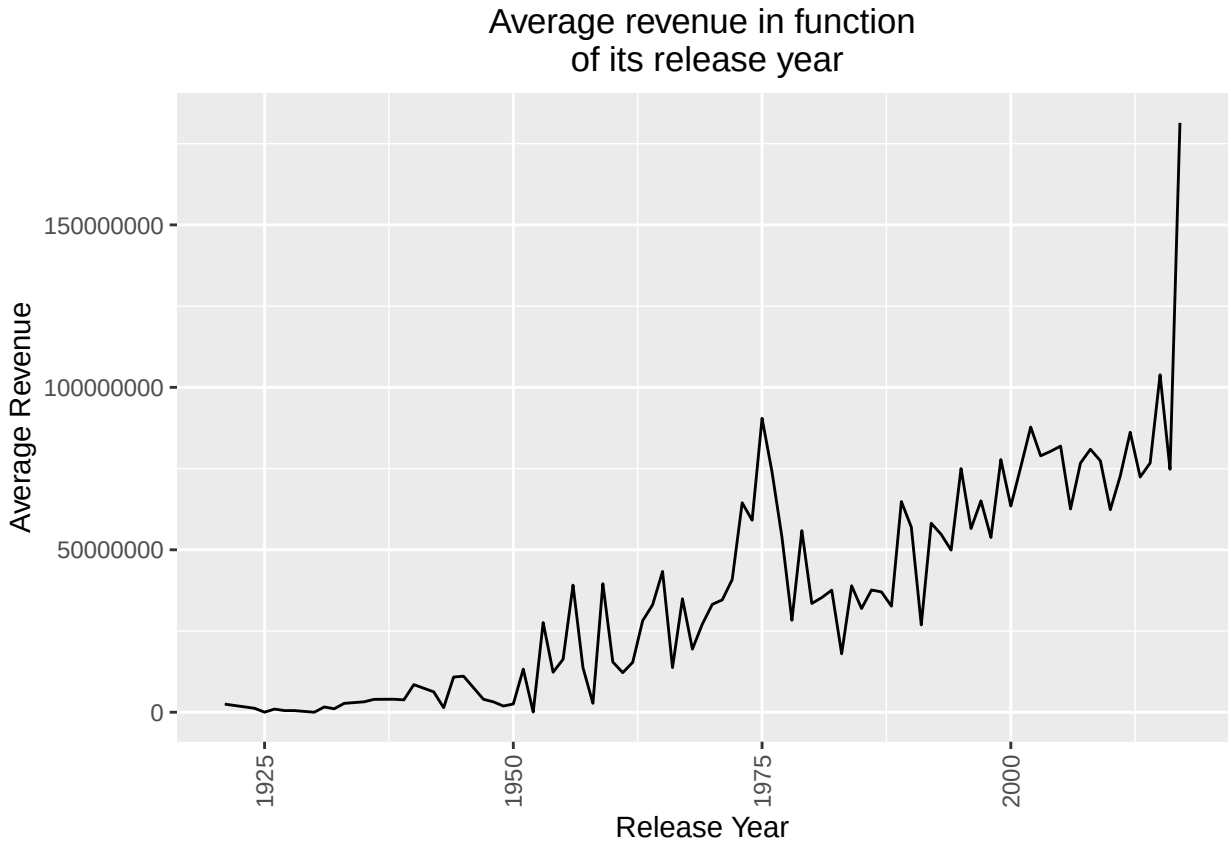
We assume also that a movie released on the weekend (mainly Friday or Saturday) generates more revenue because people are generally not working the next day. They will see the movie in the evening but we cannot check that assumption because the time is not provided in this dataset.

Another assumption is that releasing a movie on Summer generates more revenue because young students going to high school or primary school can see the movie anytime during the daylight. Indeed, the school normally ends at the middle/end of June and starts on end of August or beginning of September.

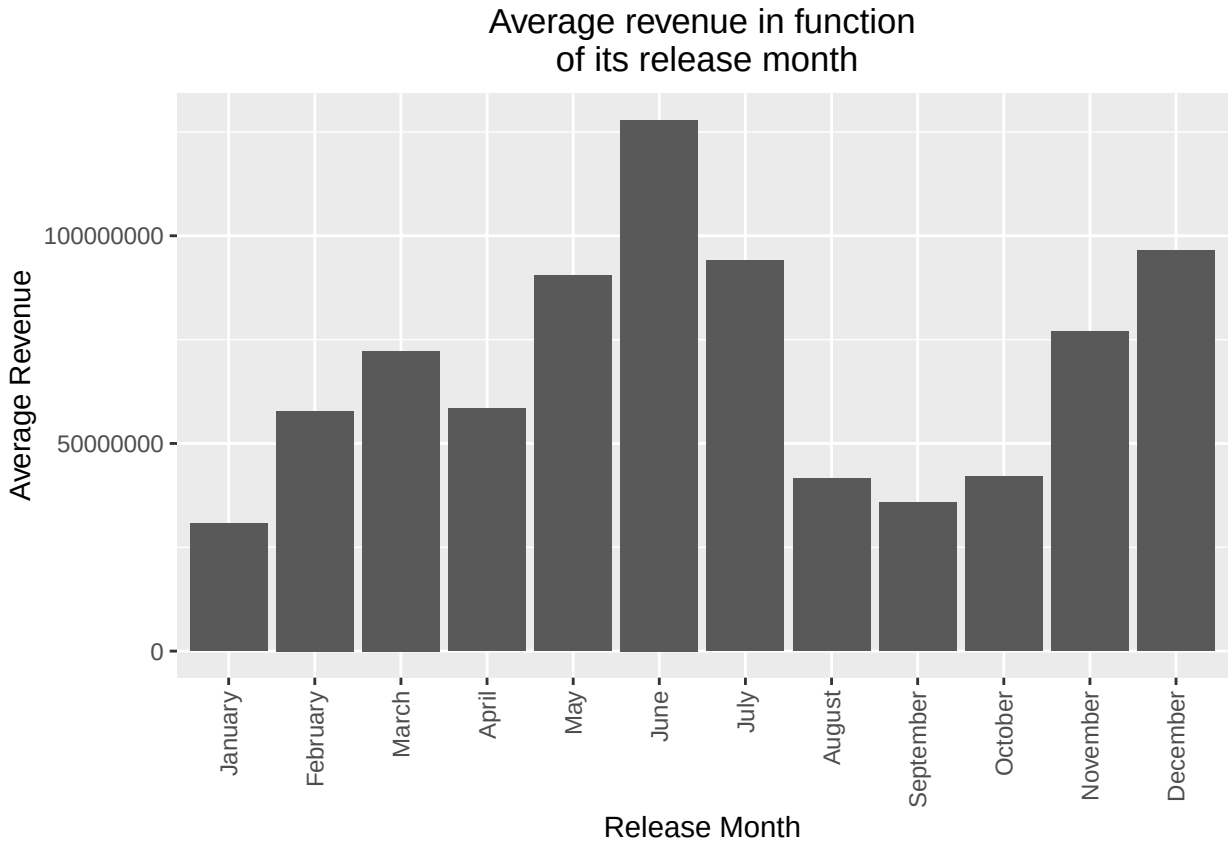
The objective is to determine if the year, month or/and weekday the movie was released have a considerable impact on the revenue. Here are the steps to achieve this objective:

1. Convert the release date from string to date with a more lisible format `YYYY-MM-DD`.
2. Extract the year, month and weekday in a data frame for every movie release date.
3. Show bar charts of the revenue in function of the release year, release month and release week-day.
4. Show line charts of the number of movies in function of the release year, release month and release weekday.

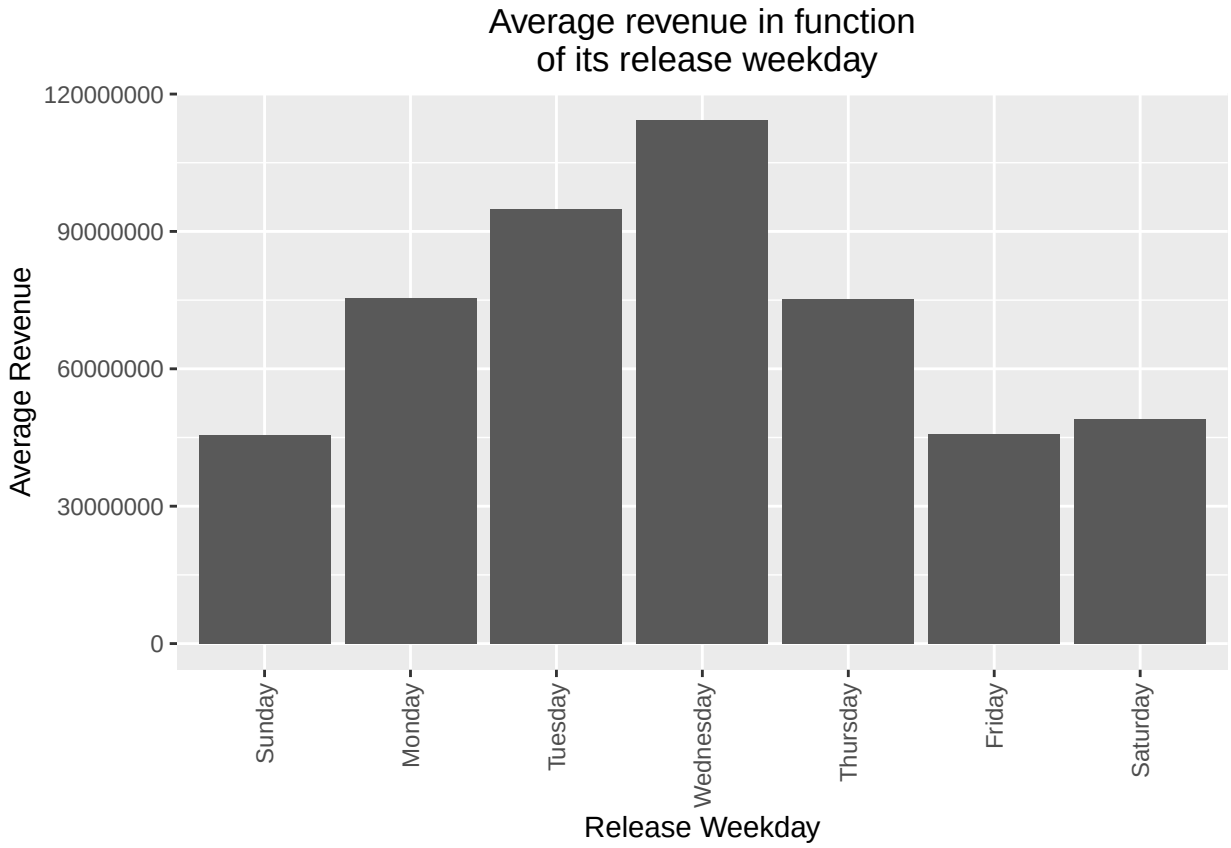
Let's see how is the average revenue per year.



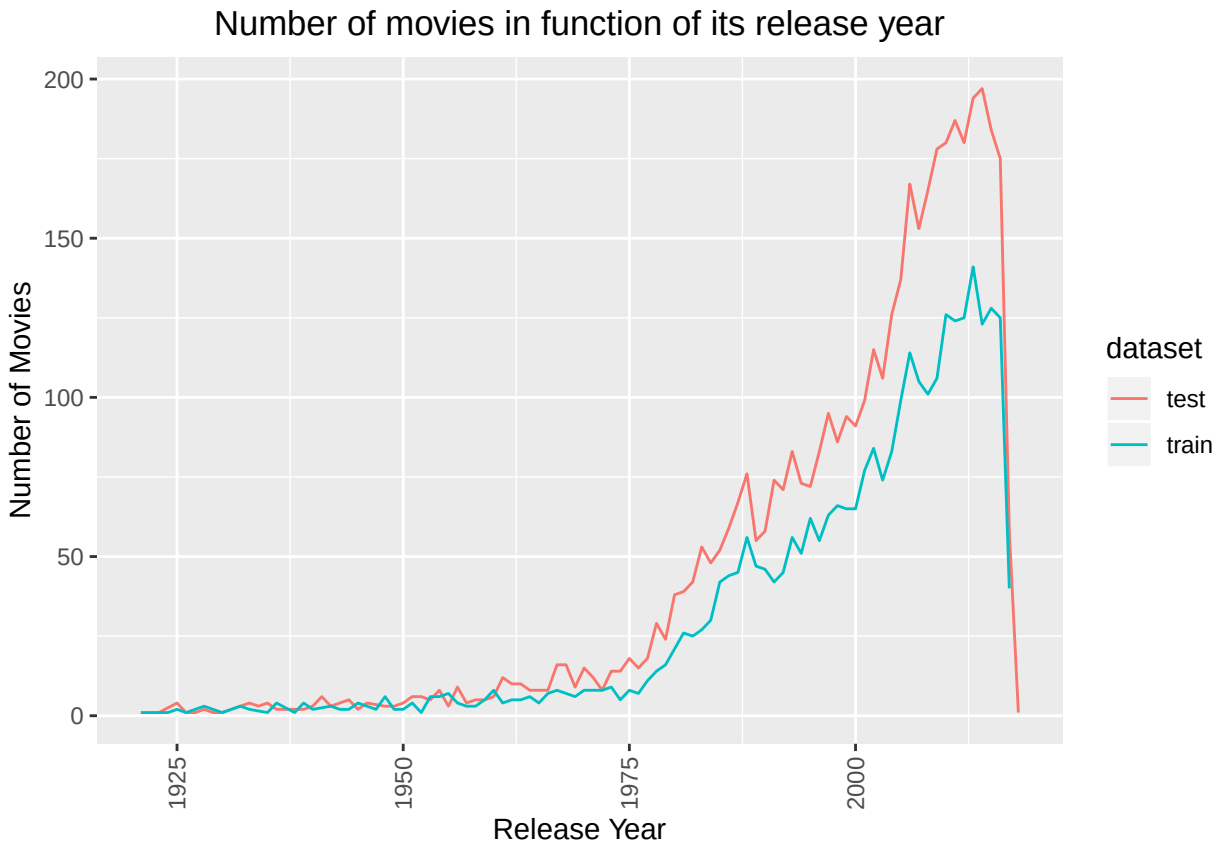
Generally, more recent are the movies, better are the revenue. However, we see that around the year 1975, the average revenues were as good as today which is suprising. Let's see how is the average revenue per month.



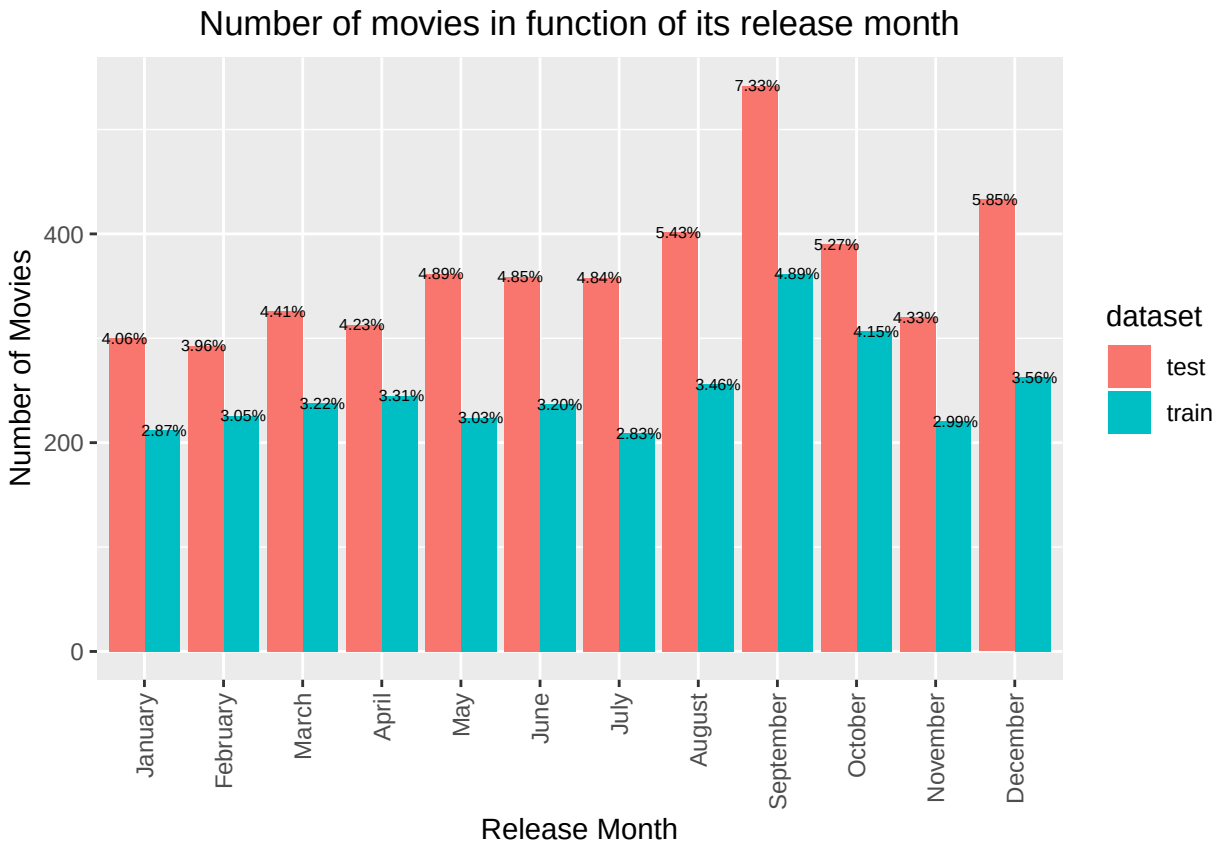
Months of May, June and July got better revenue (specially June) which may be caused by the summer time as our assumption states. In December, the revenue is also great probably because of Christmas where children are on holiday vacations the last part of the month.



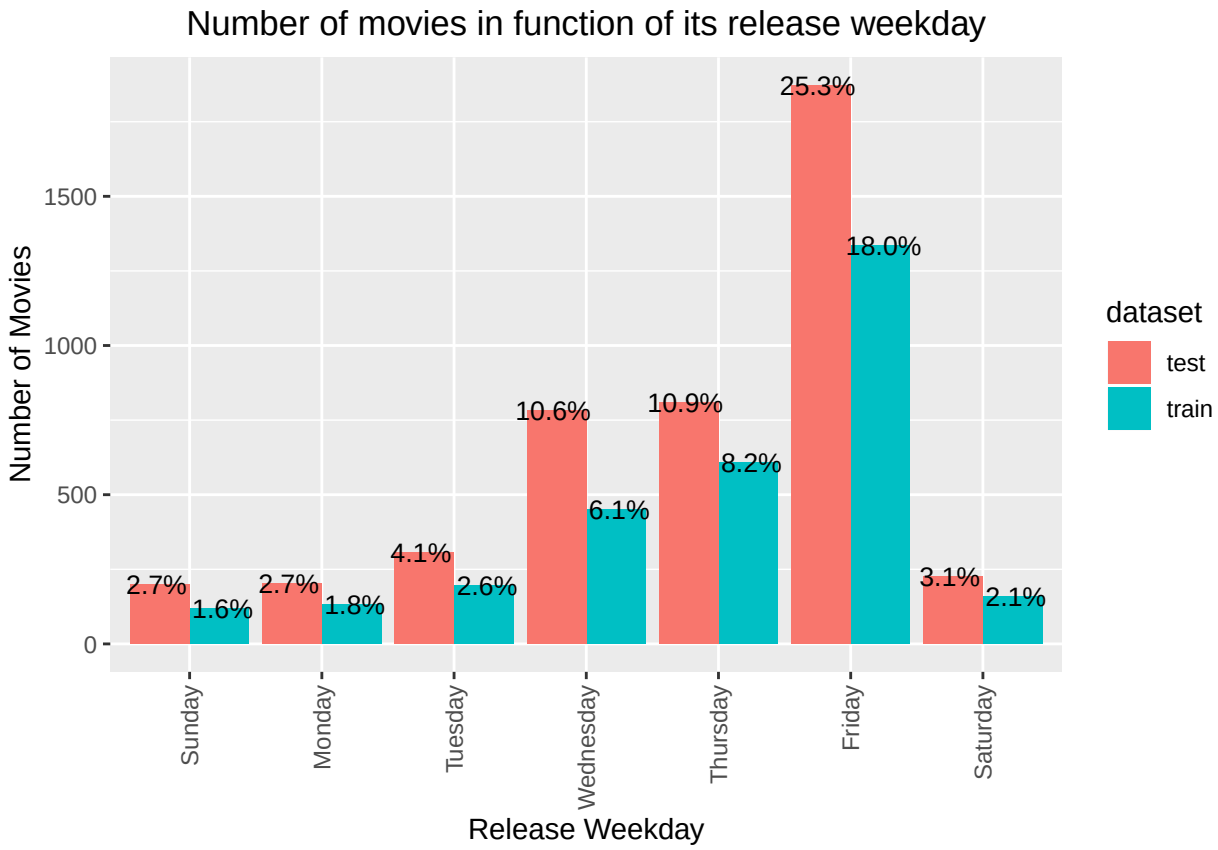
Movies relased on Wednesday got better revenue followed by Thuesday. This is suprising because it is in the middle of the week. The number of movies per year, month and weekdays could be a factor to consider because it could explain the suprising insights we got.



There are a huge difference on the number of movies made betwween 1975 and older, and 2000 and 2013. The good average revenue we saw around 1975 is explained by few (around 10 and less) movies that got big revenues whereas more than 100 have been made from the early 2000. In these 100 movies, some of them got enough low revenues to low down the average considerably. This is why the average revenues seem to be similar since 1950.



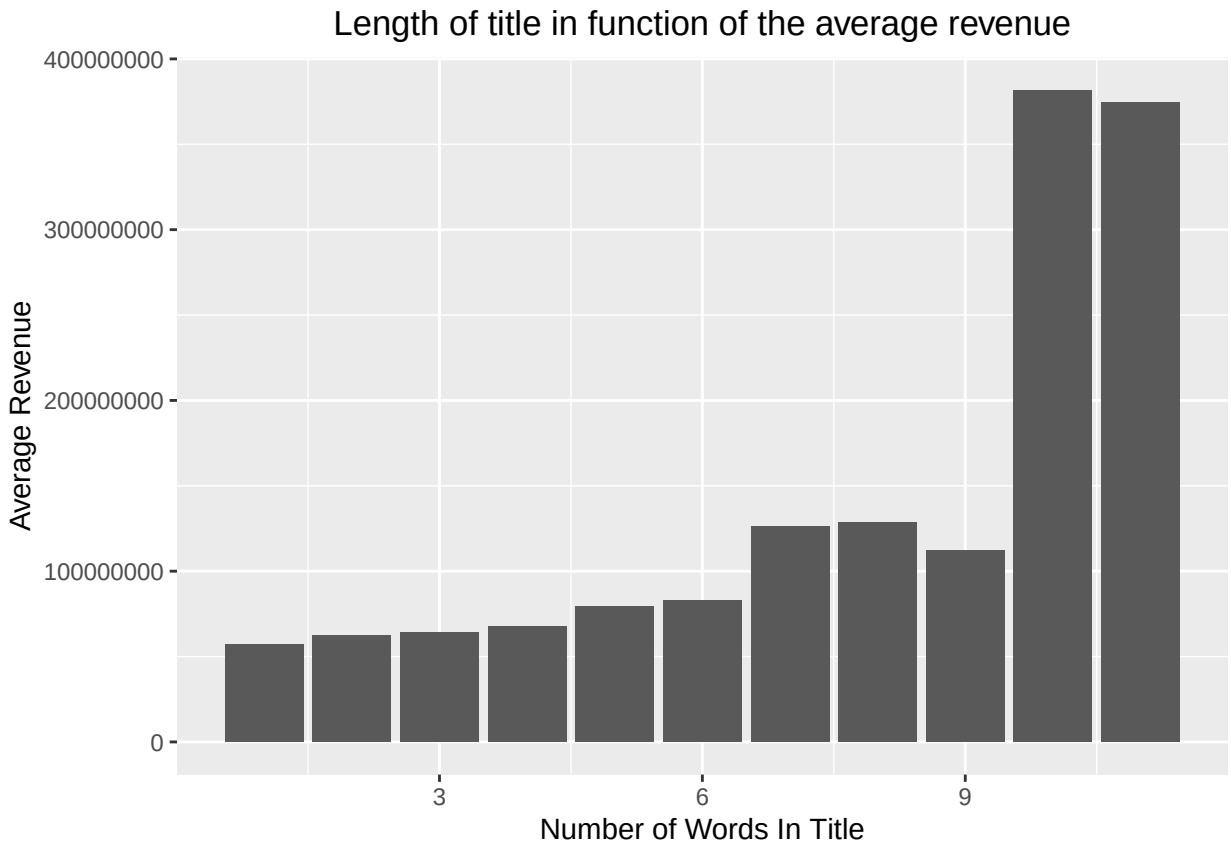
In order, most of movies are released on September, October and December. This is explained by the back-to-school period, Halloween and Christmas. However, more than 200 movies have been released every month and September get among the lowest average revenues.

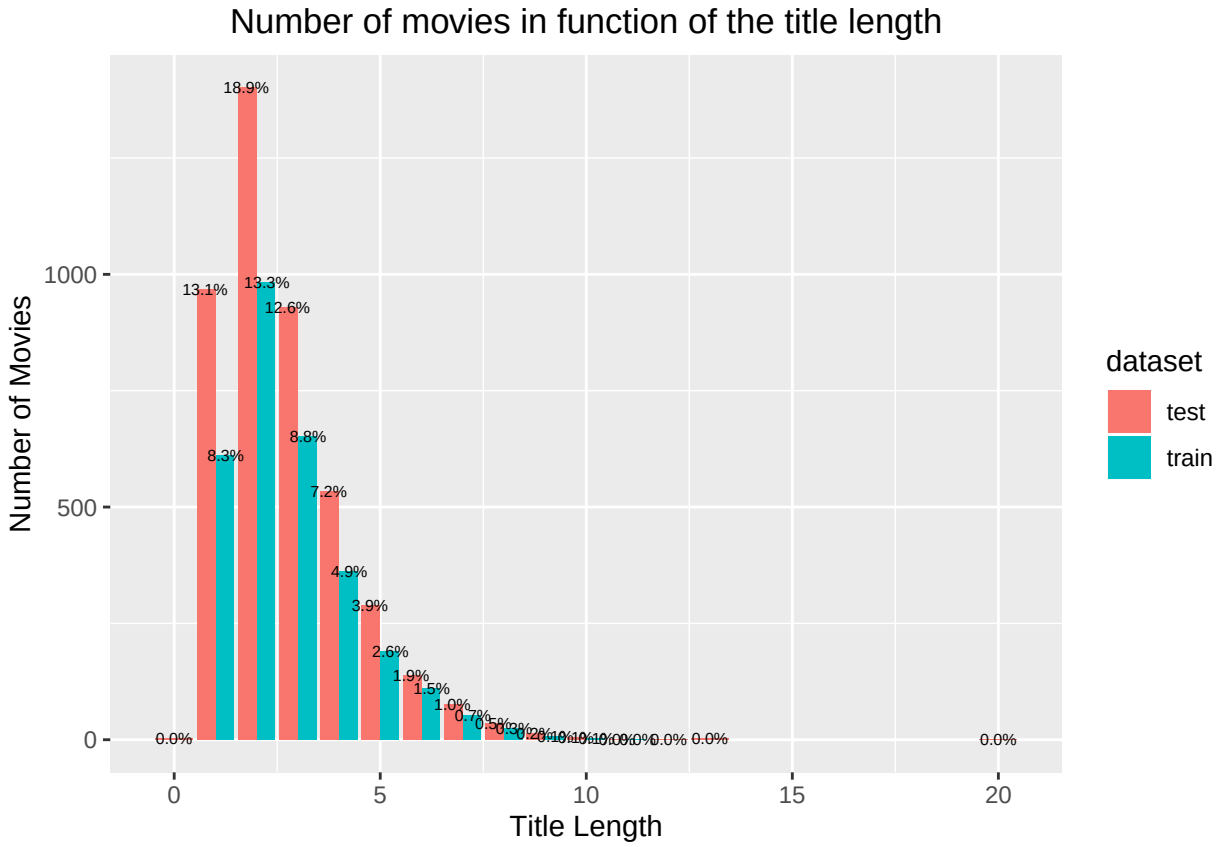


We see clearly that Friday dominates on the number of movies released. Having a much larger sample of movies on Friday increases the chances that some of them generated among the biggest and the lowest revenues.

4.16 Movie Title

Generally, the title is something we want to remember easily, then it should be short and attractive. Hence, we assume that shorter titles with attractive words help on the revenues.


Length of title in function of the average revenue
Average Revenue
400000000
300000000
200000000
100000000
0
3
6
9
Number of Words In Title


Number of movies in function of the title length
Number of Movies
1000
500
0
18.9%
13.1%
13.3%
12.6%
8.3%
8.8%
7.2%
4.9%
3.9%
2.6%
1.9%
1.5%
1.0%
0.7%
0.0%
0
5
10
15
20
Title Length
dataset
test
train

4.17 Production Companies

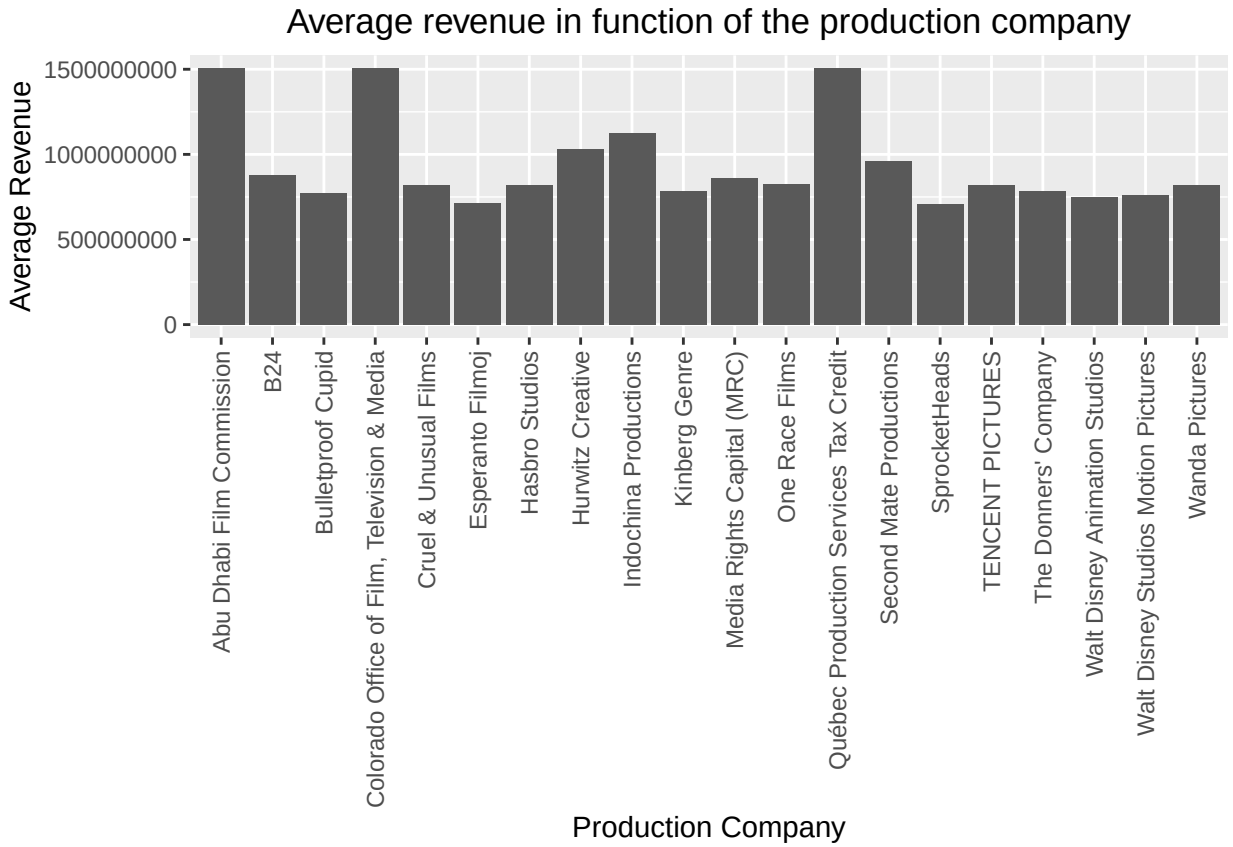
A production company is generally the one that sets the budget of a movie and makes decisions on hiring directors, actors and writters (crew and casting). Production companies can team up together to work on bigger movies involving many great actors. Thus, we expect that making a movie without a production company will generate a very low revenue. For companies teaming up with others contribute on the budget. Some well-known production companies are known to produce great movies. For example, *Walt Disney Pictures* is known for animated and family movies where The beauty and the beast got a great revenue. Others like *Marvel Studios*, *Warner Bros*, *Paramount Pictures* and *Twentieth Century Fox Film Corporation* are also well-known and generated great movies as well.

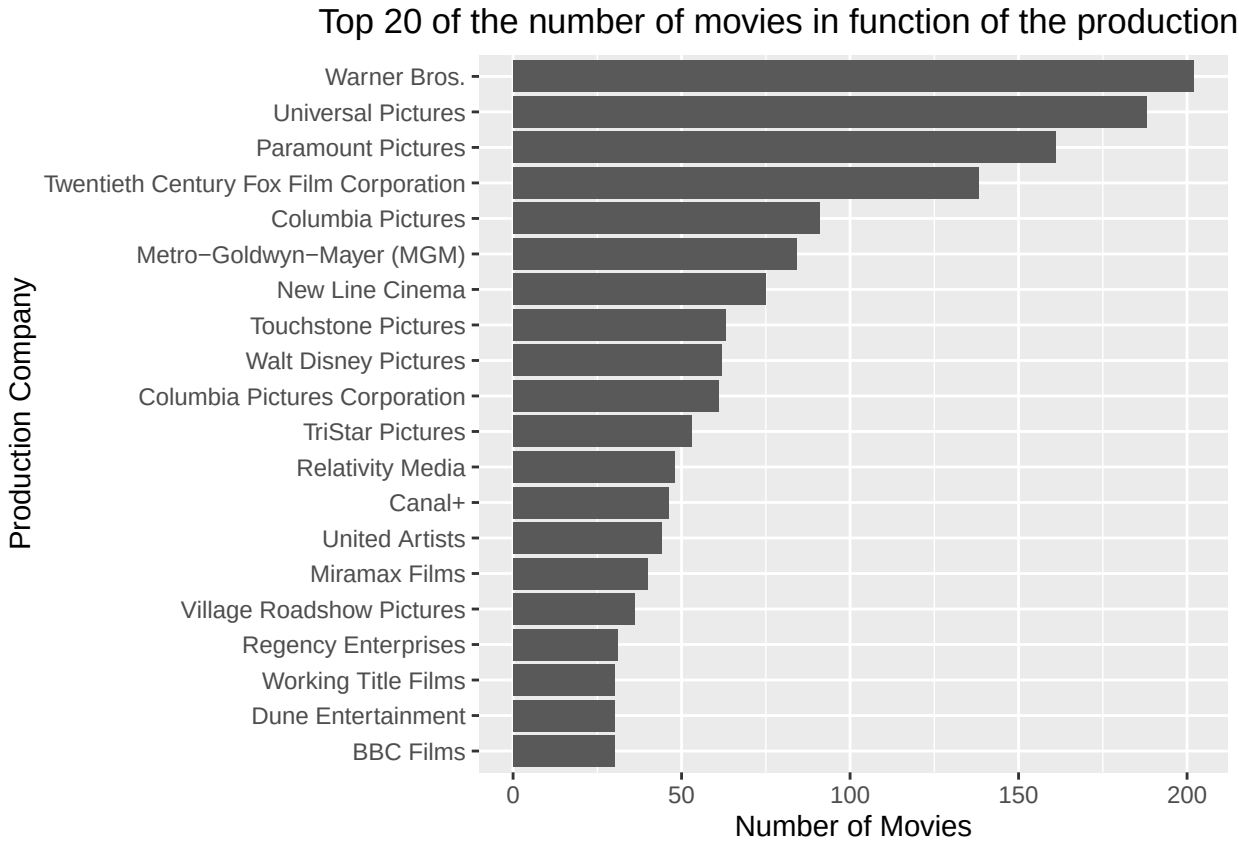
In this dataset, we see that a movie can have been made by many production companies. In this case, each of them could have contributed on the budget. However, the contribution for each of them is not known. We only know the budget allowed to make the movie. We could divide the budget by the number of production companies involved but we assume that this will not be accurate.

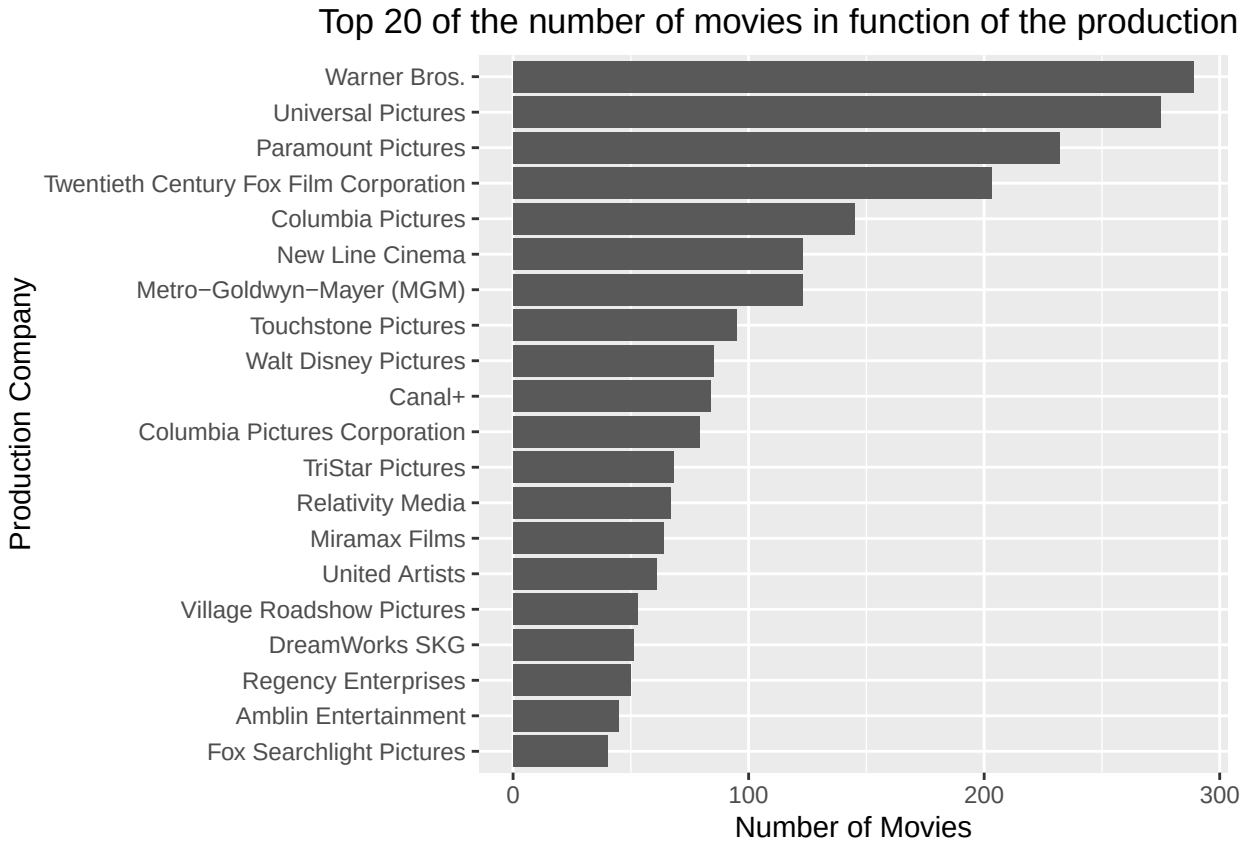
We have to be careful about the budget because some companies like *Film It Suda* has a budget of $ but a revenue of $ which is completly absurd. Also, we could consider only the number of movies made by a production company to sort the companies from the most important to the less important. Here again, what if a company is new (made its first movie), invested a high percentage of the budget and this movie generated among the best revenue? In such case, considering only the number of movies is not sufficient.

We know that one or many production companies can produce a movie. However, we cannot assume that the production companies in the train set will all be in the test set and vice-versa. Because of that, we cannot base our results on the average revenue, since the revenue does not exist in the test set. Therefore, we will have to estimate it. Knowing that, the first objective is to visualize how the revenue and the number of movies vary in function of the production company. This objective is split in 3 steps:

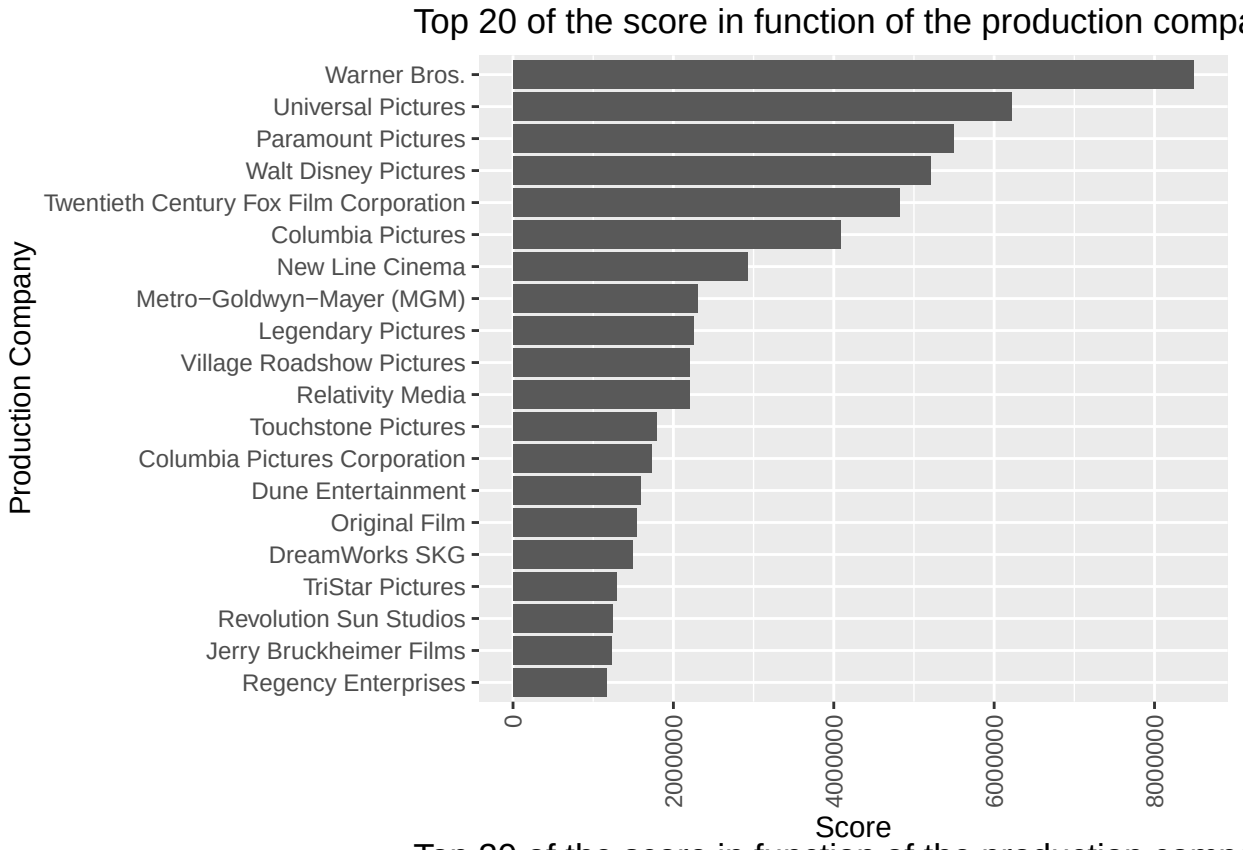
1. Calculate the average revenue for every production company in the train set.
2. Show a bar chart of the top 20 of the best average revenue per production company.
3. Show a top 20 table of the number of movies per production company for the train and test sets.

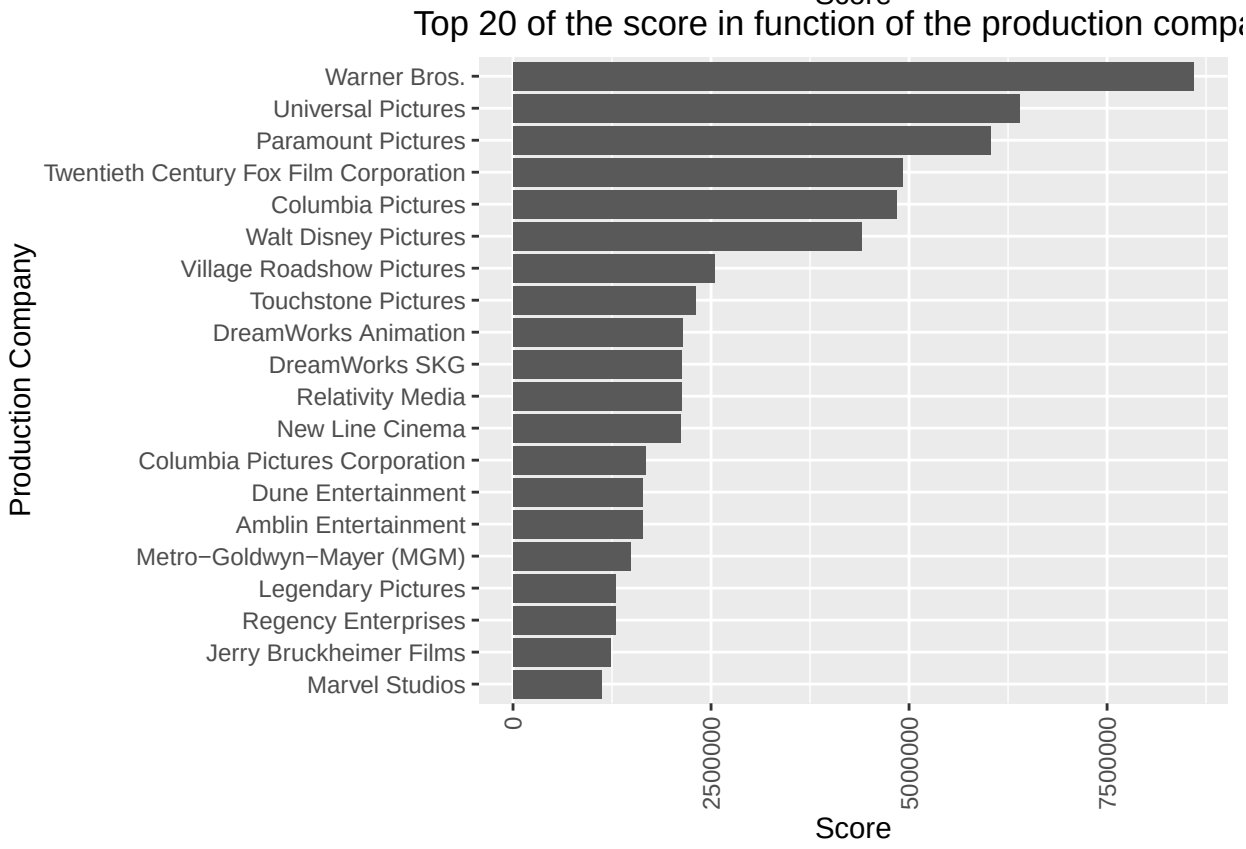

Average revenue in function of the production company
Average Revenue
1500000000
1000000000
500000000
0
Abu Dhabi Film Commission
B24
Bulletproof Cupid
Colorado Office of Film, Television & Media
Cruel & Unusual Films
Esperanto Filmoj
Hasbro Studios
Hurwitz Creative
Indochina Productions
Kinberg Genre
Media Rights Capital (MRC)
One Race Films
Québec Production Services Tax Credit
Second Mate Productions
SprocketHeads
TENCENT PICTURES
The Donners' Company
Walt Disney Animation Studios
Walt Disney Studios Motion Pictures
Wanda Pictures
Production Company


Top 20 of the number of movies in function of the production
Production Company
Warner Bros.
Universal Pictures
Paramount Pictures
Twentieth Century Fox Film Corporation
Columbia Pictures
Metro−Goldwyn−Mayer (MGM)
New Line Cinema
Touchstone Pictures
Walt Disney Pictures
Columbia Pictures Corporation
TriStar Pictures
Relativity Media
Canal+
United Artists
Miramax Films
Village Roadshow Pictures
Regency Enterprises
Working Title Films
Dune Entertainment
BBC Films
0
50
100
150
200
Number of Movies



According to these bar charts, the production companies having the best average revenue are not in the top 20 of the production companies having produced the largest number of movies. Some of them has only one movie produced with one of the best average revenues contrary to the ones having the largest number of movies. In such a case, we use the same method as we used to calculate the score for the genre.





According to these bar charts, it makes sense for most of the production companies that are in the top

20. However, we thought that `Marvel Studios` would have been ranked higher because of the well-known `Avengers`, `Spiderman`, `Ironman` movies for examples. Compared to `Warner Bros` on the number of movies produced, `Marvel Studios` is disadvantaged because they produced less movies.

5 Models

Now that we have observed how the movie revenue behave in function of our features, we start the prediction phase. This phase consists to create a model based on our train dataset in order to train our model and then apply it on the test dataset to obtain the predicted revenue. We know that regression algorithms are suitable for the problem we have to solve. Thus, we start with a linear regression model and then we compared with the Extreme gradient boosting trees and Random forest models.

Before starting to build a model from our train and test datasets, we need to define variables in order to prepare for the regression algorithms that will follow:

- $n_c \in \mathbb{N}_*$ be the number of features (columns) in the dataset.
- $n_r \in \mathbb{N}_*$ be the number of movies (rows) in the dataset.
- $X \in M_{n_r,n_c}(\mathbb{R}^+)$ be the matrix representing a dataset.
- $x_i(x_{i,1}, x_{i,2}, \ldots, x_{i,n_c})$ be the movie i in the dataset X where $x_i \in \mathbb{R}^{n_c}$.
- $y = (y_1, y_2, \ldots, y_{n_r})$ be the movie revenue to predict in the test dataset where every $y_i \in \mathbb{R}$.

In the train dataset, the number of features is $n_c = 15$ and the number of movies is $n_r = 3000$. The movie revenue known in the train set only is $y = (y_1, y_2, \ldots, y_{3000}) = (12314651, 95149435, 13092000, \ldots, 82087155)$.

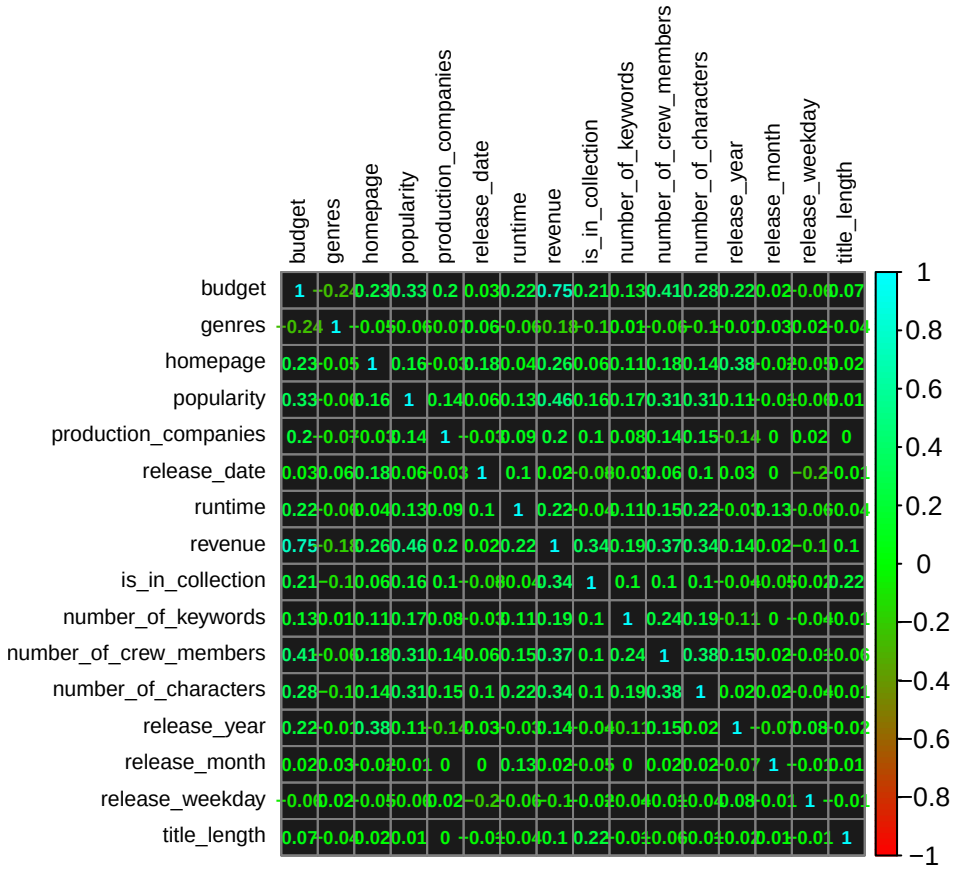
In order to obtain the movie revenue Y in the test dataset, we need to apply on each movie x_i a function f such that $y_i = f(x_i) + \epsilon(x_i)$ or equivalently

$$y_i - f(x_i) = \epsilon(x_i)$$

where $\epsilon(x_i)$ is the residuals function for $i = 1, 2, \ldots, 4398$. Note that $f(x_i)$ is our predictor function for the test dataset X where $n_r = 4398$ and $n_c = 15$.

5.1 Linear Regression Model

The objective is to build a linear regression model knowing how the features correlate together and which ones have the strongest correlation with the revenue.



We observe that the strongest correlation is between the budget (x_1) and the revenue (y) with $r(x_1, y) = 0.71$ which is good comparing to the other features.

Let's fit a log linear regression equation $\log(f(x_{i,1}) + 1) = \beta_0 + \beta_1 \log(x_{i,1} + 1)$ to the points $(x_{1,1}, y_1), (x_{2,1}, y_2), \ldots, (x_{3000,1}, y_{3000})$ and get the equation coefficients $\beta_0, \beta_1 \in \mathbb{R}$.

```
0.006 sec elapsed


Call:
lm(formula = log(train$revenue + 1) ~ log(train$budget + 1))

Residuals:
     Min       1Q   Median       3Q      Max
-15.4448  -0.8803   0.7271   1.6901  12.7212

Coefficients:
                      Estimate Std. Error t value             Pr(>|t|)
(Intercept)            2.84788    0.47202   6.033         0.0000000018 ***
log(train$budget + 1)  0.80118    0.02868  27.939 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.727 on 2998 degrees of freedom
Multiple R-squared:  0.2066,    Adjusted R-squared:  0.2063
F-statistic: 780.6 on 1 and 2998 DF,  p-value: < 0.00000000000000022
```

Using the estimated intercept value and the estimated budget coefficient, we found that the regression

equation is

$$\log(f(x_{i,1}) + 1) = 2.84788 + 0.80118 \log(x_{i,1} + 1).$$

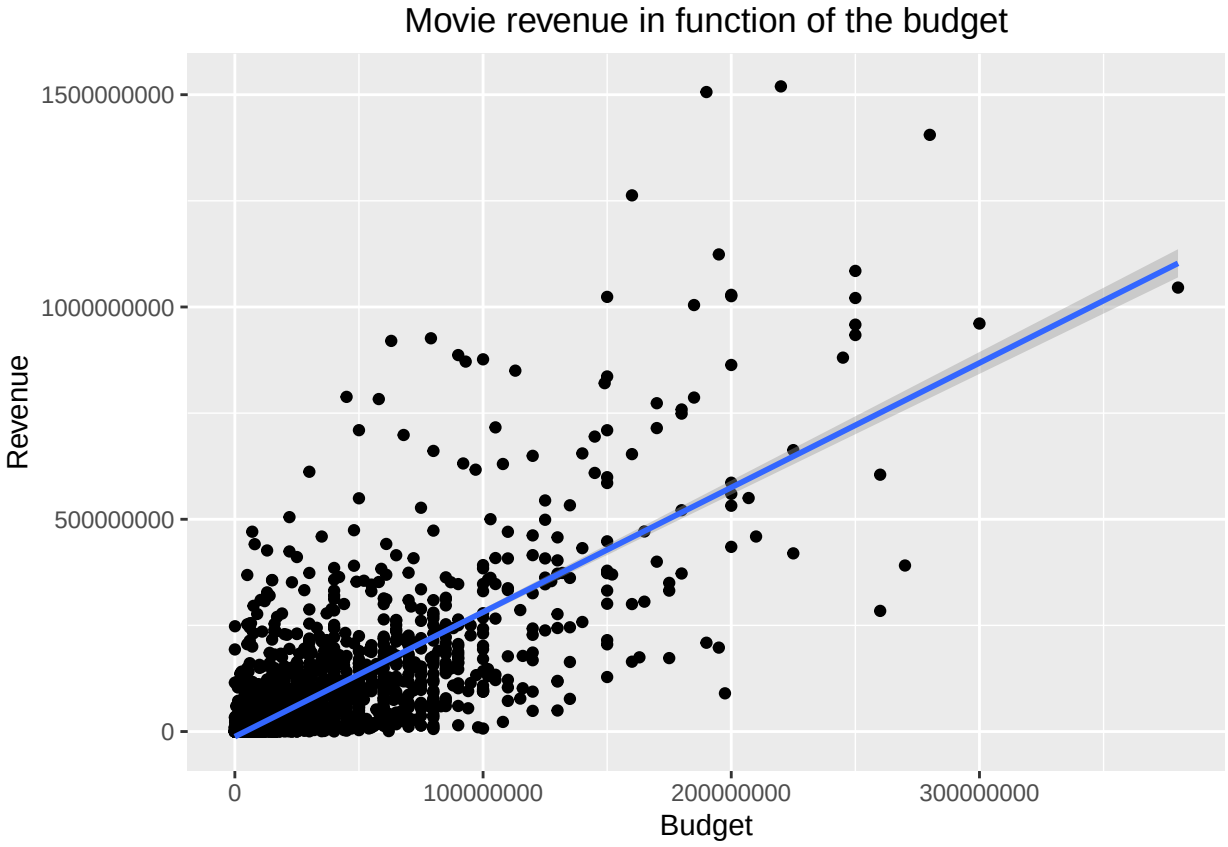
To calculate the residuals, we use the RMSLE (states for Root Mean Squared Log Error) function

$$RMSLE(y, f(x_j)) = RMSE(\log(y+1), \log(f(x_j+1))) = \sqrt{\frac{1}{n_r} \sum_{i=1}^{n_r} (\log(f(x_{i,j}) + 1) - \log(y_i + 1))^2}.$$

Then, we apply the RMSLE on our predicted revenues $f(X)$ and the revenues y given in the train dataset. We obtain:

```
RMSLE = 2.72638
```

This regression model penalizes outliers (points that are far from the linear model). Movies with low budget and generating big revenues are not following this linear model and then are penalized. The same applies with big budget allowed to movies that generated low revenues.



5.2 Extreme gradient Boosting Trees Model

The extreme gradient boosting trees model is part of an ensemble set of algorithms using the gradient descent to minimize the loss function and adjusting the residuals with weak learners (Boosting phase). For mathematical details about how the algorithm works, refer to this paper.

The weak learners could be seen as humans pulling a huge structure. Having more humans to pull the heavy structure will increase the total strength and then the structure will move quicker. Of course some of these humans are stronger than others. This is the same for weak learners. In our case, the algorithm uses regression trees as weak learners. Some of these trees could be better depending on how they are built.

5.2.1 Model Preparation Phase

The objective is to initialize the values of the parameters of the `xgboost` model and then find the optimal combination of them in order to minimize the loss function Root Mean Squared Log Error (RMSLE). Here is the list of parameters we know:

- `booster`: We will use the regression trees, hence `gbtree`.
- `objective`: The learning objective function representing the loss function to minimize. Since the suared log error loss objective function is not available yet for R, we use the linear regression objective function `reg:linear`.
- `eval_metric`: The score function used. In our case it is RMSLE but we can use `rmse` where we take the logarithm of the true and predicted revenues instead.

We have to determine the optimal value of the following parameters:

- `eta`: Step size shrinkage of new weights on regulation in order to help preventing overfitting. Corresponds to the eta of this paper.
- `gamma`: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Corresponds to the gamma of this paper.
- `lambda`: L2 Regulation term on weights in order to reduce the overfitting. Corresponds to the lambda of this paper.
- `subsample`: For each tree, a subsample (a fraction in the dataset p_r where $p_r \in]0, 1]$) of the movies is taken randomly.
- `colsample_bytree`: For each tree, a subsample (a fraction in the dataset p_c where $p_c \in]0, 1]$) of the features is taken randomly.
- `min_child_weight`: Defines the minimum sum of weights of all observations required in a child.
- `nrounds`: The number of regression trees
- `max_depth`: The maximum depth of the regression trees

This gives a total of 8 parameters to fine-tune which is to many to apply a grid search algorithm on all of them. Indeed, if we pick n candidates for every parameter, this gives n^8 possible combinations. Therefore, we will use as the initial state the default parameters given by the cross-validation function `xgb.cv`. Then, we apply (in order) the following steps where each of them has to be tested with the RMSLE score:

1. Find the best `eta` and number of trees by reducing the stepsize `eta` and increasing `nrounds` if not enough.
2. Find the best `gamma`.
3. Find the best L2 regulation `lambda`.
4. Find the best L1 regulation `alpha`.
5. Find the best sub-sample ratio of movies per tree with `subsample`.
6. Find the best sub-sample ratio of features per tree with `colsample_bytree`.
7. Find the best minimum sum of weights with `min_child_weight`.
8. Find the best maximum depth of regression trees with `max_depth`.

```
## Default parameters in xgb.cv.
# parameters <- list(booster          = "gbtree",
#                    eta              = 0.3,
#                    gamma            = 0,
#                    lambda           = 1,
#                    alpha            = 0,
#                    subsample        = 1,
#                    colsample_bytree = 1,
#                    min_child_weight = 1,
#                    max_depth        = 6)

## Optimized parameters.
```

```
parameters <- list(booster          = "gbtree",
                   objective        = "reg:linear",
                   eval_metrics     = "rmse",
                   eta              = 0.075,
                   gamma            = 0,
                   lambda           = 4,
                   alpha            = 0,
                   subsample        = 0.95,
                   colsample_bytree = 1,
                   min_child_weight = 1,
                   max_depth        = 4)

cv.number_of_folds = 10
```

5.2.2 Cross Validation

The objective is to train our model on the train set using our optimal parameters and analyse where it predicts well versus where it does not predict correctly. According to that analysis, we will adjust the parameters and if necessary, add features that could help or remove features that are noise.

We proceed to a 10-fold cross-validation to get the optimal number of trees and rmlse score. We use random subsamples representing 80% of the training set. Since we don't have too many movies (3000) and features (16), we can use 10 folds instead of 5 folds. Thus, the training set will be split in 10 test samples where each test sample has 300 movies.

Since the RMSLE evaluation metrics and the loss function associated with the RMSLE evaluation metrics function are not yet supported in the `xgboost` package of R, we have to define them manually. In the vector representation, the objective function is defined as

$$F(\hat{y}, y) = \frac{1}{2}(\ln(\hat{y}+1) - \ln(y+1))^2.$$

This is the RMSE (Root Mean Squared Error) loss function where $f(x_i) = \ln(\hat{y}+1)$ and $y = \ln(y+1)$. We keep only the predicted revenues part $\frac{\partial F(\hat{y},y)}{\partial \hat{y}}$ of the gradient $\nabla F(\hat{y}, y) = \left(\frac{\partial F(\hat{y},y)}{\partial \hat{y}}, \frac{\partial F(\hat{y},y)}{\partial y}\right)$:

$$\begin{aligned}\frac{\partial F(\hat{y}, y)}{\partial \hat{y}} &= \frac{1}{2}\frac{\partial(\ln(\hat{y}+1) - \ln(y+1))^2}{\partial \hat{y}} \\ &= \frac{\ln(\hat{y}+1) - \ln(y+1)}{\hat{y}+1}.\end{aligned}$$

We deduce the second partial derivative over predicted revenues from the first one:

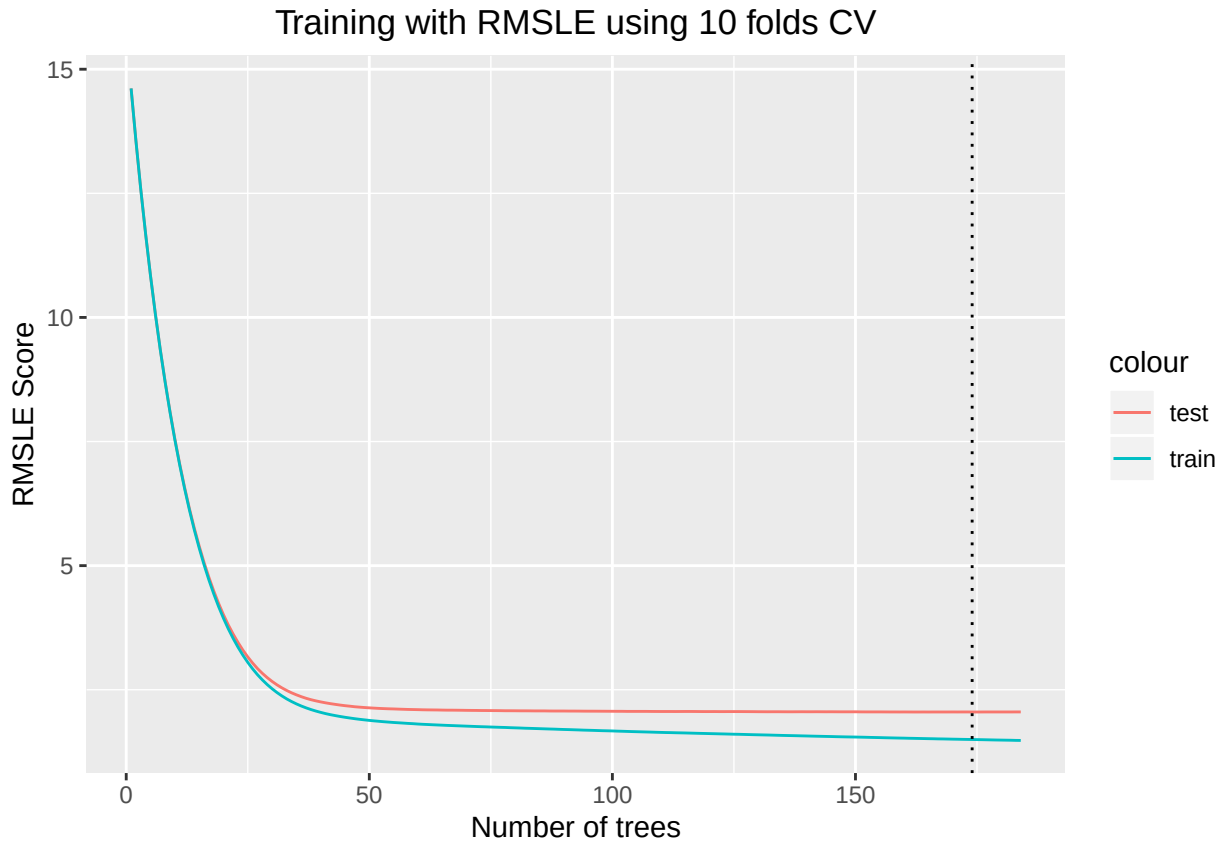
$$\begin{aligned}\frac{\partial^2 F(\hat{y}, y)}{\partial \hat{y}^2} &= \frac{\partial \frac{\ln(\hat{y}+1) - \ln(y+1)}{\hat{y}+1}}{\partial \hat{y}} \\ &= \frac{1 - \ln(\hat{y}+1) + \ln(y+1)}{(\hat{y}+1)^2}.\end{aligned}$$

We can now create our customized objective function returning the gradient and the hessian of F.

We can cross-validate with our train dataset by using the RMSE evaluation metrics function where we use $\ln(y+1)$ instead of y. This being said, the budget and production companies score features should be transformed the same way to make them on the same scale as the revenue.

```
##### xgb.cv 10-folds
    iter train_rmse_mean train_rmse_std test_rmse_mean test_rmse_std
       1       14.612629    0.009269596      14.613499    0.09053551
```

```
      2       13.551432    0.008635995       13.552966    0.08950538
      3       12.571300    0.007936109       12.572899    0.08917685
      4       11.666760    0.007158131       11.669136    0.08859532
      5       10.832460    0.006361763       10.835752    0.08738129
---
    180        1.485591    0.012896206        2.052519    0.11811698
    181        1.483975    0.012681009        2.052667    0.11819405
    182        1.482320    0.013013837        2.052321    0.11841180
    183        1.480079    0.013088287        2.052878    0.11866629
    184        1.478521    0.012916050        2.053148    0.11929996
Best iteration:
 iter train_rmse_mean train_rmse_std test_rmse_mean test_rmse_std
  174        1.497405     0.01389263       2.051478     0.1182795
```

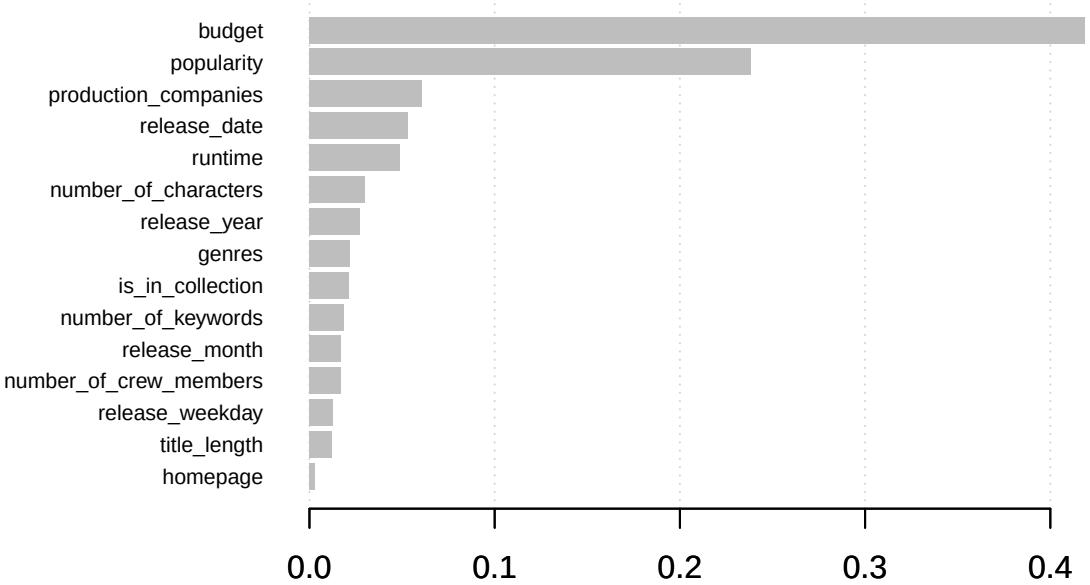


We want to see which features generated gains when used by the regression trees models.

```
0.736 sec elapsed
```

```
                 Feature        Gain      Cover  Frequency
 1:               budget 0.418796166 0.17092710 0.16157205
 2:           popularity 0.238153217 0.12788771 0.14628821
 3: production_companies 0.060477573 0.15426167 0.11397380
 4:         release_date 0.053256814 0.13078109 0.11179039
 5:              runtime 0.048660129 0.07645582 0.09825328
 6: number_of_characters 0.029716025 0.04512862 0.05109170
 7:         release_year 0.027069868 0.04367652 0.04672489
 8:               genres 0.021641134 0.04108366 0.04934498
 9:    is_in_collection 0.021264140 0.02426977 0.01310044
10:  number_of_keywords 0.018704333 0.02683995 0.04279476
```

```
11:          release_month 0.017189327 0.05952646 0.04803493
12: number_of_crew_members 0.016865800 0.03938773 0.04672489
13:        release_weekday 0.012821006 0.02319294 0.02969432
14:           title_length 0.012326826 0.02516000 0.03056769
15:               homepage 0.003057643 0.01142095 0.01004367
```



According to the heatmap of the correlation coefficients between features, we saw that the budget and popularity were the most correlated features. This could explain why these 2 features have the most importance in the XGBoost model.

5.2.3 Predictions

The objective is to apply our optimal model to our test dataset in order to calculate and obtain the predictions on the movies revenue in a CSV file. Since we will obtain our predictions as $\hat{Y} = \ln(\hat{y} + 1)$, we have to transform them to $\hat{y} = \exp(\hat{Y}) + 1$ in order to get the real predicted movie revenues.

id	revenue
3001	4314513
3002	159023
3003	9878781
3004	12701747
3005	409458
3006	10615440
3007	7593620
3008	11027305
3009	21462192
3010	449399074
3011	581626
3012	3698061
3013	39355780
3014	158928
3015	11655022
3016	724561
3017	148602826
3018	119359274

id	revenue
3019	6415849
3020	296886251
3021	47790742
3022	24820953
3023	1303501
3024	5979812
3025	4207540

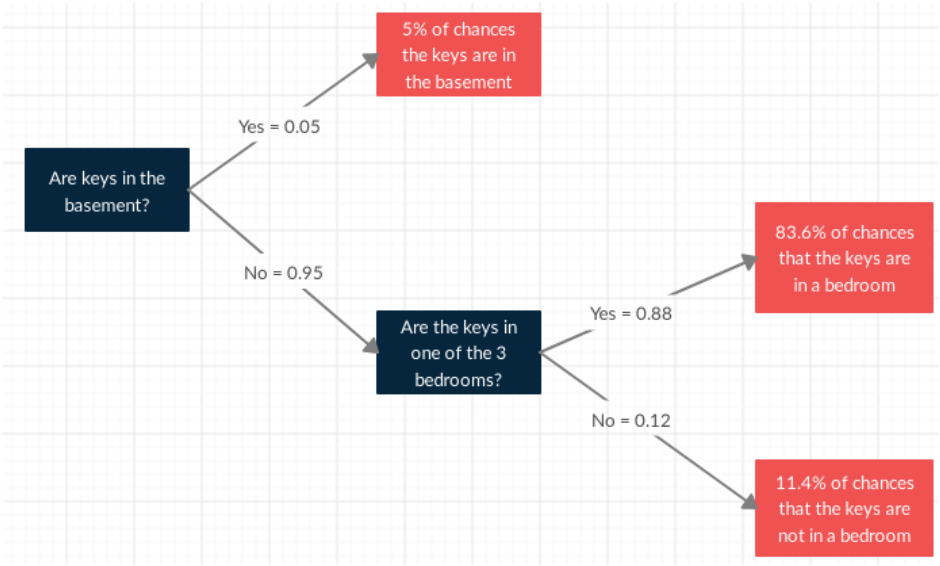
5.3 Random Forest Model

The Random forest model is part of the ensemble set of models like the gradient boosting trees one. This model uses decision trees as weak learners (see example here) in order to build its forest of n decision trees. Each decision tree in the forest uses a random subset of features on each question. Only a random subset of the training data points is used to answer a question. The purpose is to not use the same source of data but to increase the diversity in order to get more robust overall predictions. Then, the average of all decision tree estimates in the forest is taken as the prediction. The Random forest model is explained in details here.

To make decision trees intuitive for everyone, we will describe what it represents in our everyday life. Let's say that you are searching for your keys. You do not know yet but your keys are on a small desk close to your bed. Since you are pretty sure that they are in the house, then you start your researches in the house. However, they can at any place in the house making the model not enough accurate. You need the help of weak learners (decision trees) in order to facilitate your search. Right now, this can be seen as a model without any trees.

Suppose that your house has a basement and is a one-story house. The question you are asking yourself is: Are my keys in the basement? You are pretty sure that they are not in the basement. Thus, you just limit the range of your researches and then gain accuracy. However, it is still not accurate because there are too many places to look for. So you are asking yourself a second question: Are my keys in one of the 3 bedrooms? According to your past experiences, most of the time your let your keys in one of the 3 bedrooms. This limits the researches range to 3 rooms in the house. The weak learner is now a 2-level decision tree. Going further will give a very accurate result but if your habits change in the future, it could lead to the wrong room at the third level. This is the equivalent way to say that your model overfits.

Here is the decision tree representing our example where we quantified the probabilities for each decision:



5.3.1 Preparing Models

The objective is to initialize the values of the parameters of the `randomForest` model and then find the optimal combination of them in order to minimize the loss function Root Mean Squared Log Error (RMSLE). Here is the list of parameters to determine:

- `ntree`: The number of decision trees to grow.
- `mtry`: Number of variables randomly sampled as candidates at each split. Default: mtry = number of features / 3.
- `nodesize`: Minimum size of terminal nodes. Default: 5
- `maxnodes`: Maximum number of terminal nodes trees in the forest can have.
- `nPerm`: Number of times the Out-Of-Bag (OOB) data are permuted per tree for assessing variable importance.

This gives a total of 4 parameters to fine-tune which may be slow to apply a grid search algorithm on all of them. Indeed, if we pick n candidates for every parameter, this gives n^4 possible combinations. Therefore, we will use as the initial state the default parameters.

5.3.2 Cross Validation

The objective is to train our model on the train set using our optimal parameters and analyse where it predicts well versus where it does not predict well. According to that analysis, we will adjust the parameters and if necessary, add features that could help or remove features that are noise. Note that the Random forest model applies the Mean Squared Error (MSE) loss function:

$$MSE(y, \hat{y}) = \frac{1}{n_r} \sum_{i=1}^{n_r} (y_i - \hat{y}_i)^2.$$

Using $\ln(y+1)$ and $\ln(\hat{y}+1)$ instead of y and $\hat{y}$ gives the MSLE loss function instead. It follows that applying the square root on the resulting MSLE gives the RMSLE.

```
0.971 sec elapsed
```

	number_of_trees	RMSLE
1	1	2.451381
2	2	2.498880
3	3	2.361645
4	4	2.381348
5	5	2.385414
6	6	2.389985
7	7	2.349636
8	8	2.338400
9	9	2.333927
10	10	2.322673
11	11	2.314264
12	12	2.312216
13	13	2.310553
14	14	2.310130
15	15	2.300217
16	16	2.292763
17	17	2.298380
18	18	2.286415
19	19	2.277159
20	20	2.272324
21	21	2.269051
22	22	2.267539

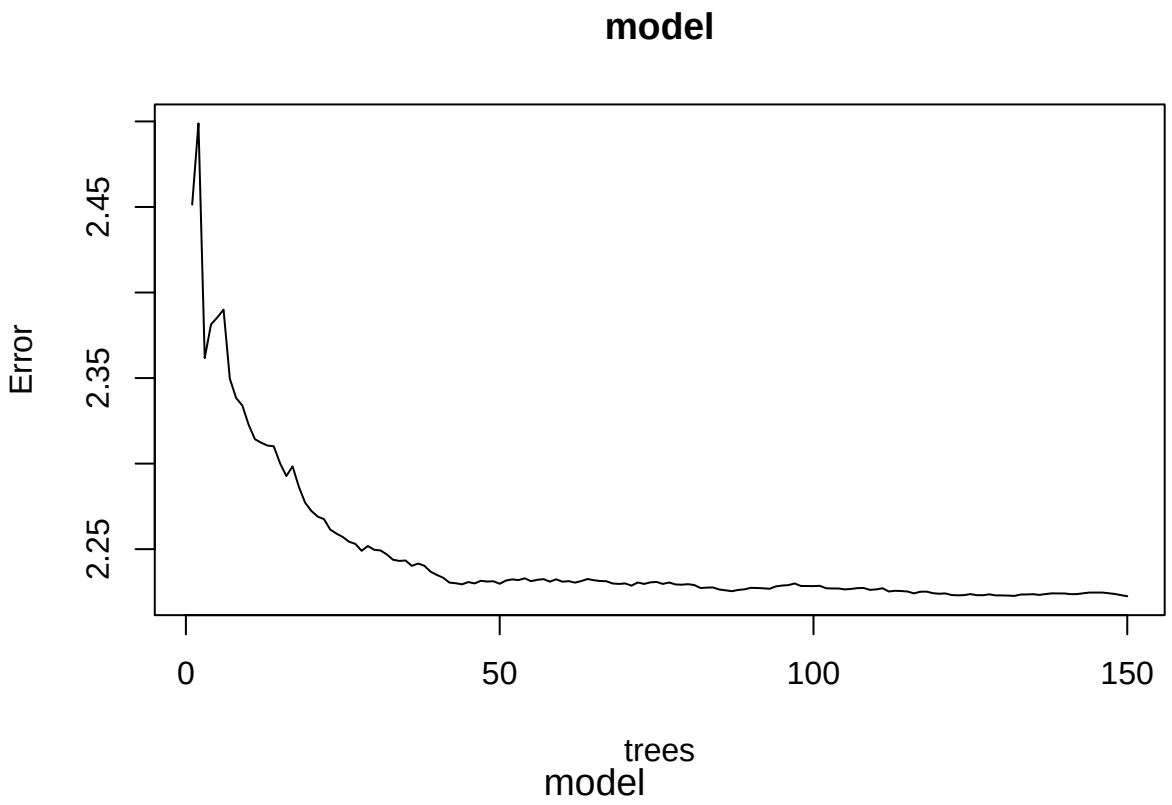
```
23              23 2.261461
24              24 2.259072
25              25 2.257130
26              26 2.254330
27              27 2.253091
28              28 2.249051
29              29 2.251846
30              30 2.249629
31              31 2.249257
32              32 2.246959
33              33 2.243814
34              34 2.243128
35              35 2.243384
36              36 2.240245
37              37 2.241590
38              38 2.240290
39              39 2.236809
40              40 2.234928
41              41 2.233281
42              42 2.230451
43              43 2.230084
44              44 2.229443
45              45 2.230777
46              46 2.229980
47              47 2.231498
48              48 2.231087
49              49 2.231245
50              50 2.229776
51              51 2.231668
52              52 2.232354
53              53 2.231925
54              54 2.232948
55              55 2.231275
56              56 2.232113
57              57 2.232465
58              58 2.231004
59              59 2.232374
60              60 2.230971
61              61 2.231316
62              62 2.230440
63              63 2.231364
64              64 2.232563
65              65 2.231851
66              66 2.231399
67              67 2.231259
68              68 2.229948
69              69 2.229678
70              70 2.229967
71              71 2.228665
72              72 2.230545
73              73 2.229688
74              74 2.230605
75              75 2.230845
76              76 2.229683
```

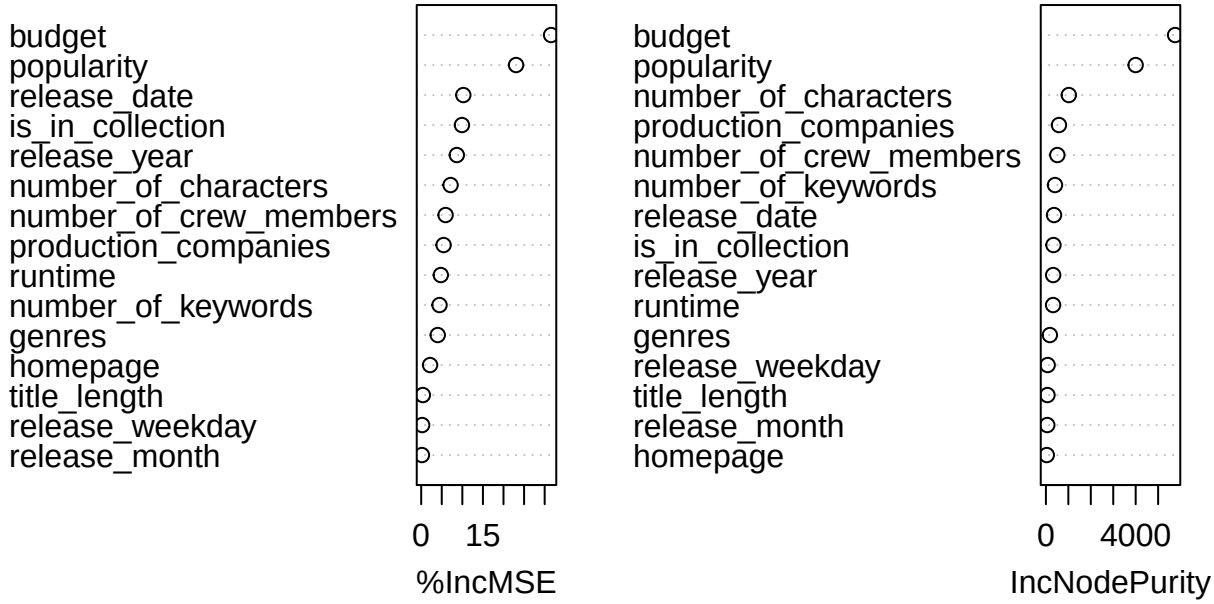
77	77	2.230525
78	78	2.229380
79	79	2.229237
80	80	2.229545
81	81	2.229039
82	82	2.227293
83	83	2.227529
84	84	2.227604
85	85	2.226411
86	86	2.225948
87	87	2.225425
88	88	2.226174
89	89	2.226517
90	90	2.227385
91	91	2.227306
92	92	2.227120
93	93	2.226842
94	94	2.228247
95	95	2.228737
96	96	2.228985
97	97	2.229945
98	98	2.228484
99	99	2.228474
100	100	2.228410
101	101	2.228581
102	102	2.227170
103	103	2.227017
104	104	2.227013
105	105	2.226487
106	106	2.226771
107	107	2.227234
108	108	2.227301
109	109	2.226210
110	110	2.226494
111	111	2.227156
112	112	2.225272
113	113	2.225671
114	114	2.225514
115	115	2.225244
116	116	2.224165
117	117	2.225104
118	118	2.225202
119	119	2.224312
120	120	2.223940
121	121	2.224127
122	122	2.223194
123	123	2.223021
124	124	2.223136
125	125	2.223736
126	126	2.223122
127	127	2.223080
128	128	2.223610
129	129	2.222974
130	130	2.222981

```
131             131 2.222873
132             132 2.222683
133             133 2.223471
134             134 2.223543
135             135 2.223675
136             136 2.223278
137             137 2.223782
138             138 2.224186
139             139 2.224120
140             140 2.224107
141             141 2.223690
142             142 2.223759
143             143 2.224231
144             144 2.224653
145             145 2.224635
146             146 2.224643
147             147 2.224264
148             148 2.223829
149             149 2.223143
150             150 2.222486
```

```
The number of trees minimizing the RMSLE is: 150
```

```
The minimum RMSLE is: 2.222486
```





	%IncMSE	IncNodePurity
budget	31.5646564	5756.08440
genres	4.0194215	175.70373
homepage	2.1623018	38.50311
popularity	23.0567482	3998.32885

```
production_companies    5.4432039     581.31007
release_date           10.2338177     361.64389
runtime                 4.7678996     320.16624
is_in_collection        9.9054878     336.03305
number_of_keywords      4.4518129     404.76698
number_of_crew_members  5.9282017     508.88321
number_of_characters    7.1368130    1020.90305
release_year            8.6532571     322.97839
release_month           0.1763142      60.58258
release_weekday         0.2347733      77.25628
title_length            0.4057597      68.95388
```

For the same reason given as for the XGBoost model, the budget and popularity are strongly correlated with the revenue which explains how much they are important to the model.

5.3.3 Predictions

The objective is to apply our optimal model to our test dataset in order to calculate and obtain the predictions on the movies revenue in a CSV file. Since we will obtain our predictions as $\hat{Y} = \ln(\hat{y} + 1)$, we have to transform them to $\hat{y} = \exp(\hat{Y}) + 1$ in order to get the real predicted movie revenues.

id	revenue
3001	4276768
3002	760654
3003	20307379
3004	8911920
3005	635989
3006	15480556
3007	6901509
3008	37648273
3009	22284831
3010	217249288
3011	805837
3012	6171755
3013	13017530
3014	314831
3015	37836242
3016	2259311
3017	87696962
3018	103952642
3019	14227692
3020	183804855
3021	50728817
3022	40257583
3023	4648216
3024	21789500
3025	6965802

6 Conclusion

We conclude on the following points:

1. A comparison table of the models with their RMSLE score and runtime.
2. The top 10 of movies generating the biggest revenue.
3. From our best scores for predicting the revenues, we want to know why the model did not predict well for some movies?

6.1 Models Comparison Table

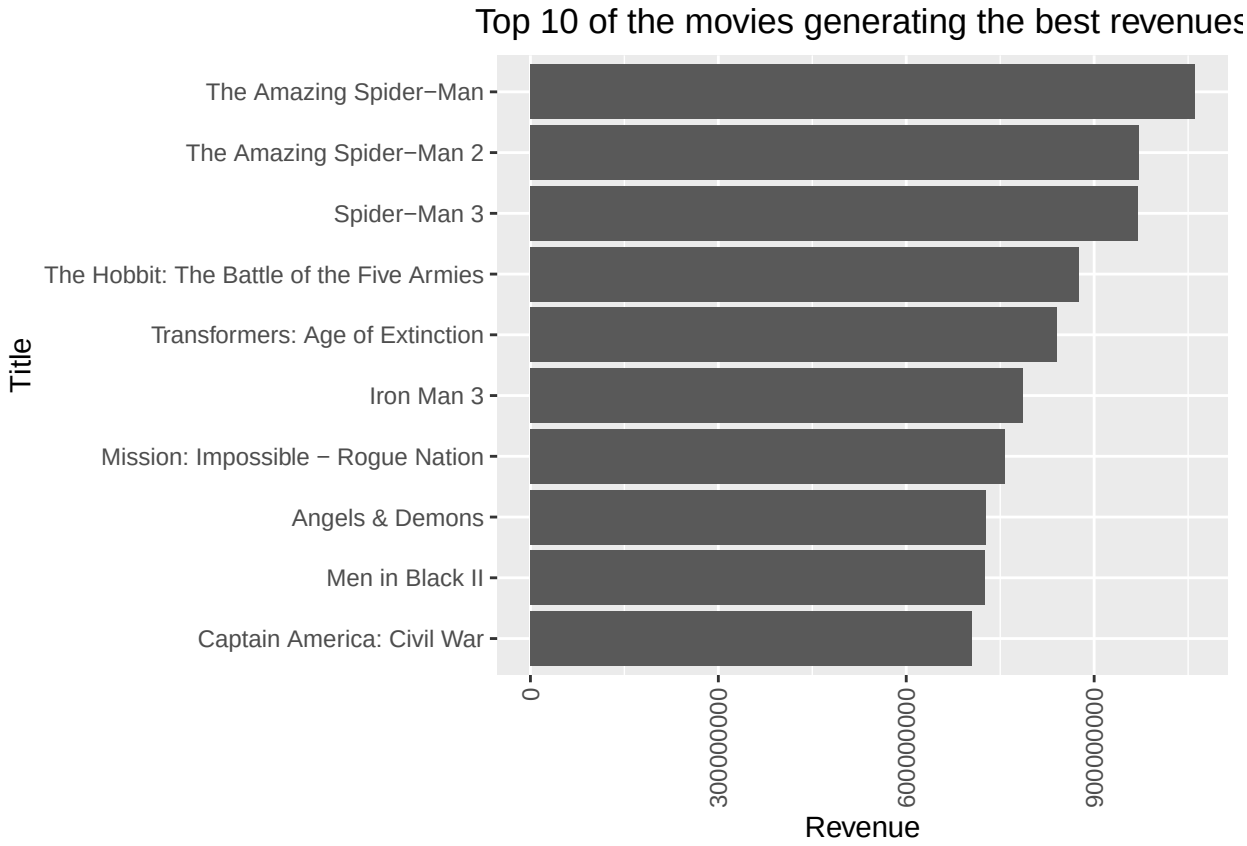
model	rmsle	running_time
Linear Regression	2.726380	0.006
Extreme Gradient Boosting Trees	2.051478	0.736
Random Forest	2.222486	0.971

All of these models ran under one second, hence the best model is the extreme gradient boosting trees because of its lowest prediction errors.

6.2 Top Movie Revenues Predicted

We present the top 10 movies that generated the biggest predicted revenues among the test set:

id	revenue	title
3986	1060699185	The Amazing Spider-Man
3389	971304842	The Amazing Spider-Man 2
3097	970064381	Spider-Man 3
6868	876261463	The Hobbit: The Battle of the Five Armies
3964	841197337	Transformers: Age of Extinction
7239	786385618	Iron Man 3
4051	757131400	Mission: Impossible - Rogue Nation
5291	727860941	Angels & Demons
3879	725543427	Men in Black II
4590	705472396	Captain America: Civil War



6.3 Retrospective on Predictions

Our best model to predict the revenue of movies in the test set has performed well for most of the movies but also did not predict well for some of the movies. Let's see the top 20 of the worst predeiction our model calculated among the 3000 movies in the train set:

title	revenue	predicted_revenue	error
Transformers: Dark of the Moon	1123746996	176107415	947639580
Beauty and the Beast	1262886337	327481828	935404508
The Avengers	1519557910	688390529	831167380
Furious 7	1506249360	707665058	798584301
Avengers: Age of Ultron	1405403694	629655283	775748411
Jurassic Park	920100000	224995773	695104227
Zootopia	1023784195	344037575	679746619
Gravity	716392705	119481589	596911115
Batman Begins	374218673	951225433	577006761
Transformers: Revenge of the Fallen	836297228	261752450	574544777
Ice Age: Dawn of the Dinosaurs	886686817	316628072	570058745
Transformers	709709780	151639576	558070204
The Lion King	788241776	238725948	549515827
Star Wars: Episode III - Revenge of the Sith	850000000	306050555	543949444
The Passion of the Christ	611899420	96855340	515044080
Maleficent	758539785	268291635	490248149
Finding Dory	1028570889	552024344	476546545
Deadpool	783112979	322222723	460890256
Ghost	505000000	46820283	458179717

title	revenue	predicted_revenue	error
The Twilight Saga: New Moon	709827462	263309240	446518222

The first issue identified is about the revenue where we saw that it could be a gross or worldwide revenue. The amount will not be the same. Moreover, we saw in our analysis of the revenues that some movies have revenues near zero (e.g. the movie `Sammy` with a revenue of 3$). Some use the Worldwide US while others use gross revenues and we cannot say if they are all in US dollars or if some of them are converted or not. Knowing that, it is obvious that we cannot have a very good accuracy. Here are some examples of abnormal low revenues:

	title	revenue	predicted_revenue	error	budget
2981	The Cookout	12	208026	208014.42986	16000000
2982	Chasing Liberty	12	18192747	18192735.90576	23000000
2983	The Day He Arrives	11	74640	74629.59931	16000000
2984	Bats	10	669697	669687.07301	16000000
2985	Elektra Luxx	10	1000418	1000408.89668	16000000
2986	Pollock	8	1397246	1397238.62494	6
2987	Bodyguard	8	7902	7894.79743	130
2988	Kops	8	1000169	1000161.91416	16000000
2989	He-Man and She-Ra: The Secret of the Sword	7	367266	367259.45964	16000000
2990	Never Talk to Strangers	6	5133134	5133128.23871	6400000
2991	East of Eden	5	33112	33107.19715	1
2992	American Adobo	4	176	172.87801	344
2993	Saamy	3	88	85.04917	1
2994	All at Once	3	112431	112428.66487	750000
2995	Borsalino	3	6941078	6941075.26772	16000000
2996	Tere Naam	2	122	120.84895	1
2997	The Wind in the Willows	1	9564	9563.25557	12
2998	Mute Witness	1	142	141.13163	2
2999	Missing	1	95527	95526.07704	16000000
3000	The Merry Widow	1	22473	22472.93364	592

We strongly recommend to unify the units of the revenues. This means to convert them in one unit (e.g. Gross revenues in US dollars) for all movies with correct conversion. If it is hard to integrate, adding the necessary conversions, deductions as features in the dataset would at least be very useful.