MovieLens Project Report

BenoitA

2022-12-09

1. Introduction

We are provided with a dataset of historical movie rating (called "edx") and we have the task to predict movie ratings and calculate RMSE. To do so, we are advised to build a model, that we will check against a test dataset (also provided and called "final_holdout_test"). To get the full mark, our objective is to build a model that will result in a RMSE below 0.86490.

To achieve this, the project will go through different steps that we will describe in the Methodology section. We will then describe the Results, and finally provide a Conclusion to this report.

To help the reader, we have kept clear definitions of each variables used in the project:

1.1 Data dictionary

1.1.1 Dataframes

Data frame	In the original dataset	Description
edx	Yes	Train data set
final_holdout_test	Yes	Test data set
user_table	No	User focused (grouped) data set
users_rating_summ	No	User data summary
movie_table	No	Movie focused (grouped) data set
movies_rating_summ	No	Movie data summary
genres_table	No	Genre focused (grouped) data set
year_rating_table	No	Rating year focused (grouped) data set
age_rating_table	No	Rating year focused (grouped) data set

There are a few additional dataframes created for the purpose of the models whioch are not listed here.

1.1.2 Data in original dataframes

Data frame	Variable Name	In the original dataset	Description
edx AND final_holdout_test	userId	Yes	Id of the user that rated the movie
edx AND final_holdout_test	movieId	Yes	Id of the rated movie
edx AND final_holdout_test	rating	Yes	Rating of the movie given by the user
edx AND final_holdout_test	timestamp	Yes	Date and time of the rating
edx AND final_holdout_test	title	Yes	Name and year of release of the rated movie

Data frame	Variable Name	In the original dataset	Description
edx AND final_holdout_test	genres	Yes	Category of the rated movie by genre
edx AND final_holdout_test	year_of_rating	No	Year of the rating was given
edx AND final_holdout_test	movie_year	No	Year that the movie was released
edx AND final_holdout_test	rating_age	No	age (in years) of the movie at the time of the rating

1.1.3 Data in additional dataframes

Those table have often the following variables:

- an ID (movie, user...) or a type (e.g. a genre)
- a count of the ratings (rows in the original edx table), often labeled as 'n_'
- an average rating (average by the focused filter/group), often labeled as 'avg_rating'
- a standard deviation of the rating, often labeled as 'sd'
- a difference with the overall mean for the purpose of the models, often labeled as 'b_'

2. Methodology & Modeling Process

The methodology applied is the following:

1. Exploration of the data sets in order to have an overview of the available data and variables
2. Data wrangling to prepare the data sets
3. Analysis of the data sets to understand possible correlations between variables that can be later used in the model
4. Modelling, starting from the most simple model and sophisticating bit by bit, checking the RMSE after each attempts to select the model with the lowest RMSE

2.1 Datasets overview

2.1.1 edx dataset

Here is an overview of the structure of the dataset:

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
```

The variables have the following features:

```
##           vars       n         mean           sd         min        max
## userId       1 9000055 3.586982e+04     20585.25         1.0      71567
## movieId      2 9000055 4.121700e+03      8942.11         1.0      65133
## rating       3 9000055 3.510000e+00         1.06         0.5          5
## timestamp    4 9000055 1.032616e+09 115966785.22 789652009.0 1231131736
## title        5 9000055          NaN           NA         Inf       -Inf
```

```
## genres       6 9000055          NaN           NA         Inf       -Inf
##                  range       se
## userId        71566.0     6.86
## movieId       65132.0     2.98
## rating            4.5     0.00
## timestamp 441479727.0 38655.48
## title            -Inf       NA
## genres           -Inf       NA
```

We can observe the followings:

1. The dataset is made up of a huge number of rows but a relatively small number of columns (predictors).
2. It is not a complete 'matrix' where all users would have rated all movies. We have only 1.21 % of coverage.
3. Years are including at the end of the film name and could potentially be a variable that impacts rating.
4. Timestamp is not an easy format to handle.
5. Genres are aggregated and is not easy for analysis.

2.1.2 final_holdout_test dataset

Here is an overview of the structure of the dataset:

```
## 'data.frame':    999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : int  231 480 586 151 858 1544 590 4995 34 432 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)"
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Roman
```

The variables have the following features:

```
##           vars      n         mean           sd         min        max
## userId       1 999999 3.587021e+04     20586.16         1.0      71567
## movieId      2 999999 4.107590e+03      8904.97         1.0      65133
## rating       3 999999 3.510000e+00         1.06         0.5          5
## timestamp    4 999999 1.032520e+09 115938574.35 789652009.0 1231131028
## title        5 999999          NaN           NA         Inf       -Inf
## genres       6 999999          NaN           NA         Inf       -Inf
##                  range        se
## userId        71566.0     20.59
## movieId       65132.0      8.90
## rating            4.5      0.00
## timestamp 441479019.0 115938.63
## title            -Inf        NA
## genres           -Inf        NA
```

We can observe the followings:

1. The dataset structure is the same as the edx dataset
2. But the number of rows is much smaller

2.2 Datasets preparation

Based on the observations, we decide to add a few columns to the edx dataset to help us in the analysis and modelling:

- Timestamp is not easy to handle (it is not readable, nor categorised). We can convert it to year to make it a more useful predictor and call it 'year_of_rating'. We can also delete the timestamp column which we will not use.
- Extract the year of the movie, which might be a useful predictor. We will call it 'movie_year'.
- Compute the age of the movie at the time the rating was done, which might also be a predictor. We will call it 'rating_age'.

We replicate the preparation to the final_holdout_test dataset so that we can use the sets for calculating the predictions and RSME.

2.3 Data Analysis

We will now analyze more thoroughly some of the (initial or created) variables of the edx dataset trying to find insights that can then be helpful to build the prediction model.

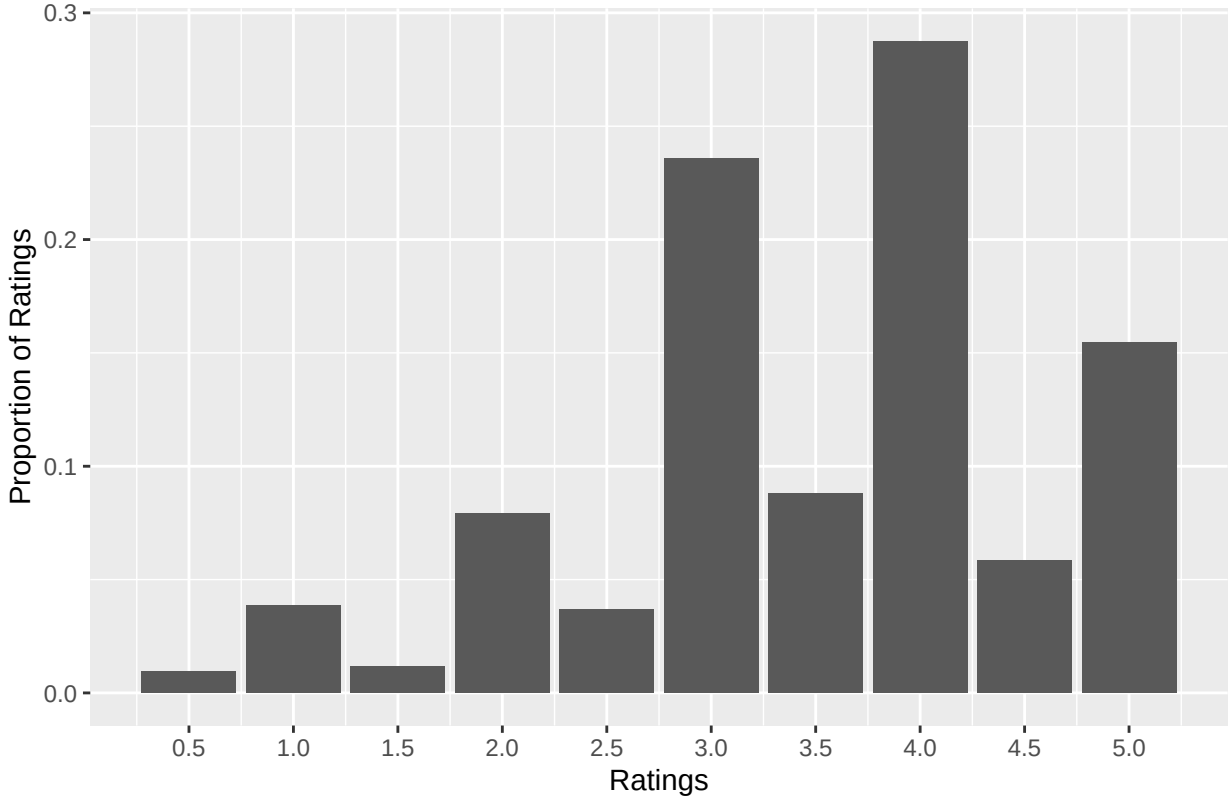
2.3.1 Rating insights

The rating is the variable that we aim to predict.

The ratings are distributed from 0.5 to 5 stars with half star increments. The average is 3.51 and the standard deviation is 1.06.

The ratings are distributed as follows:

Figure 1: Proportion of ratings per rating



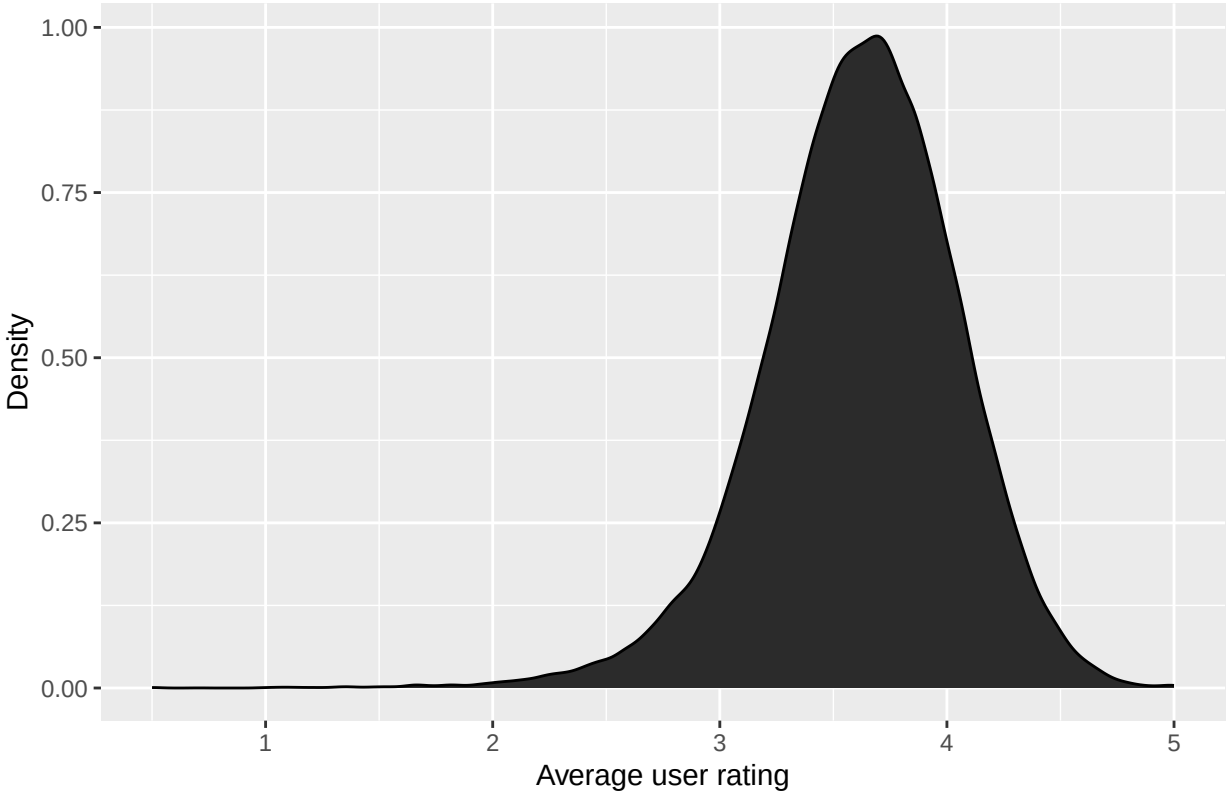
Insight(s):

(1) Half stars ratings are less common than whole star ratings

2.3.2 Users insights

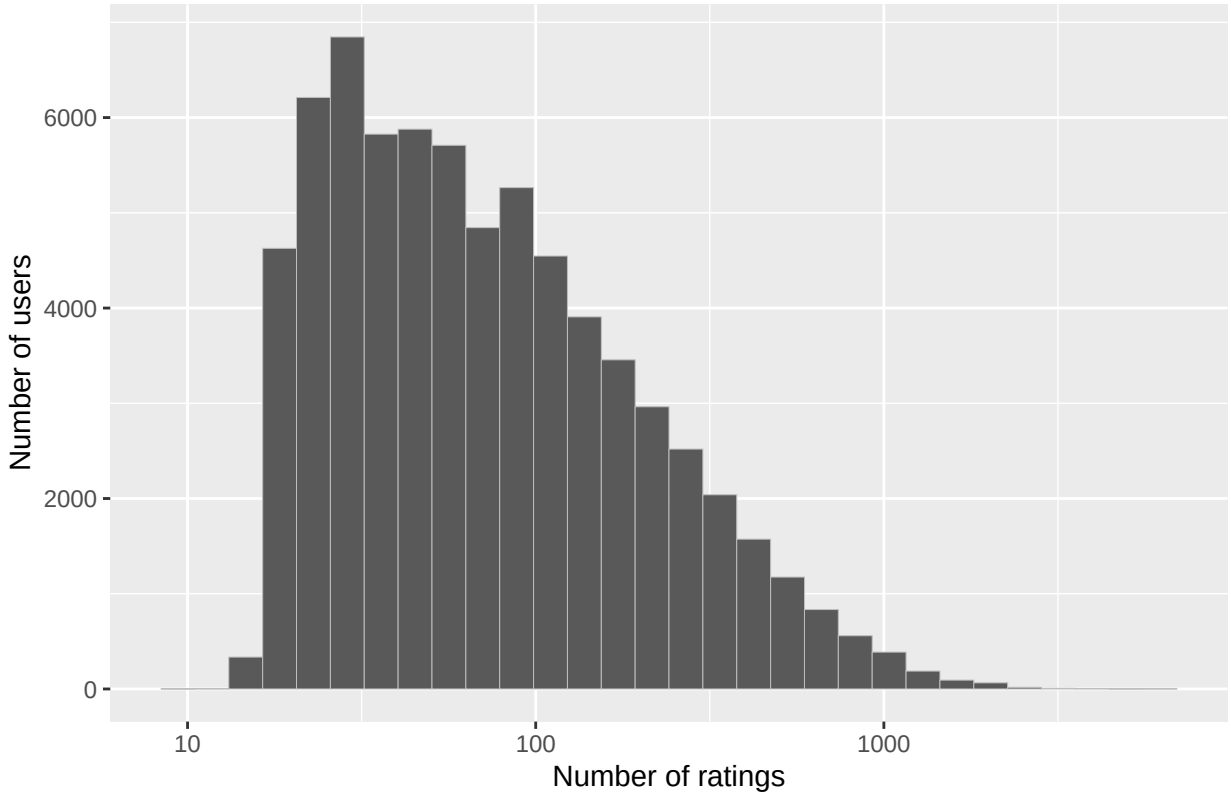
There are 69878 users that rated movies. Users have given a minimum average rating of 0.5 star and a maximum of 5 stars. The average rating of users is 3.61 stars with a standard deviation of 0.5. The distribution of ratings per user is the following:

Figure 2: Distribution of average ratings given by users



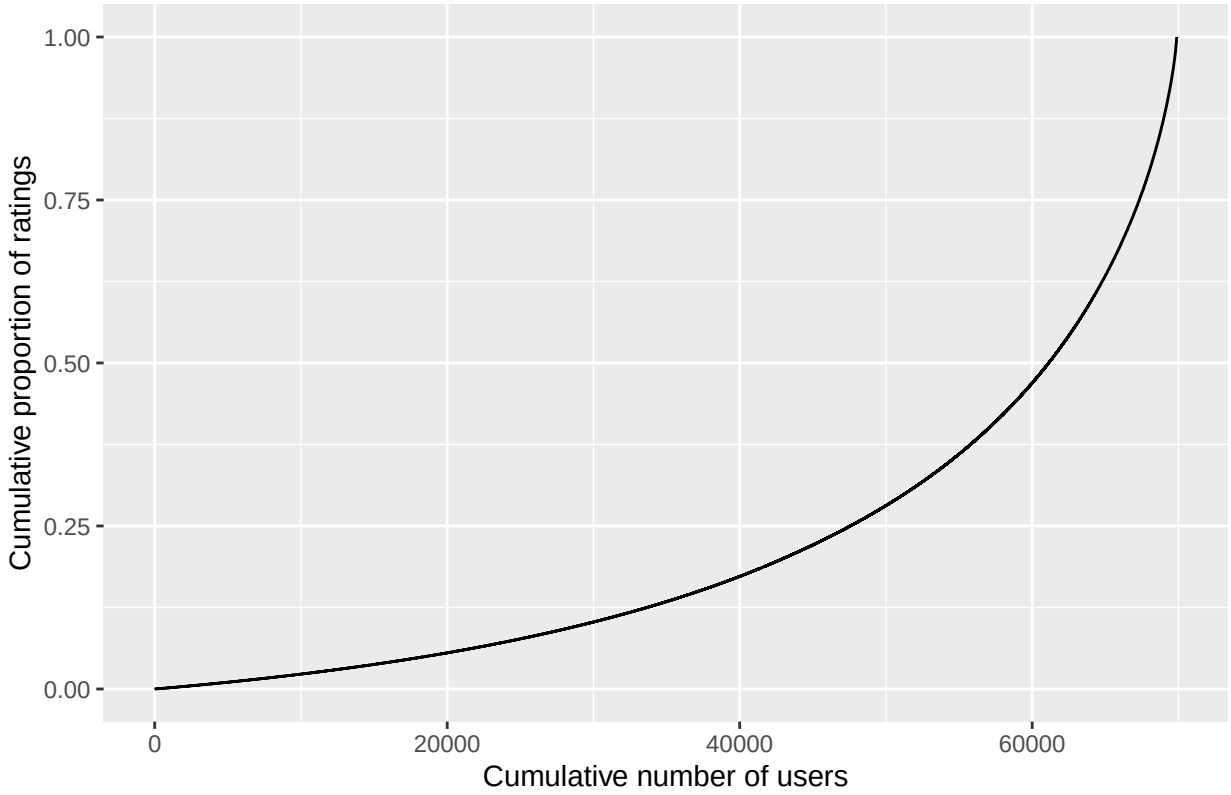
Users have rated a minimum of 10 movies and a maximum of 6616 movies. On average they have rated 128.8 movies with a standard deviation of 195.06. The median number of movies rated per user is 62. The distribution of the number of movies rated per user is the following:

Figure 3: Number of ratings per user



And we can observe that the ratings are highly concentrated of a small number of users:

Figure 4: Cumulative proportion of ratings




```
##          0%         25%         50%         75%        100%
## "0.0000011" "0.0460328" "0.1336052" "0.3164254" "1.0000000"
```

Insight(s):

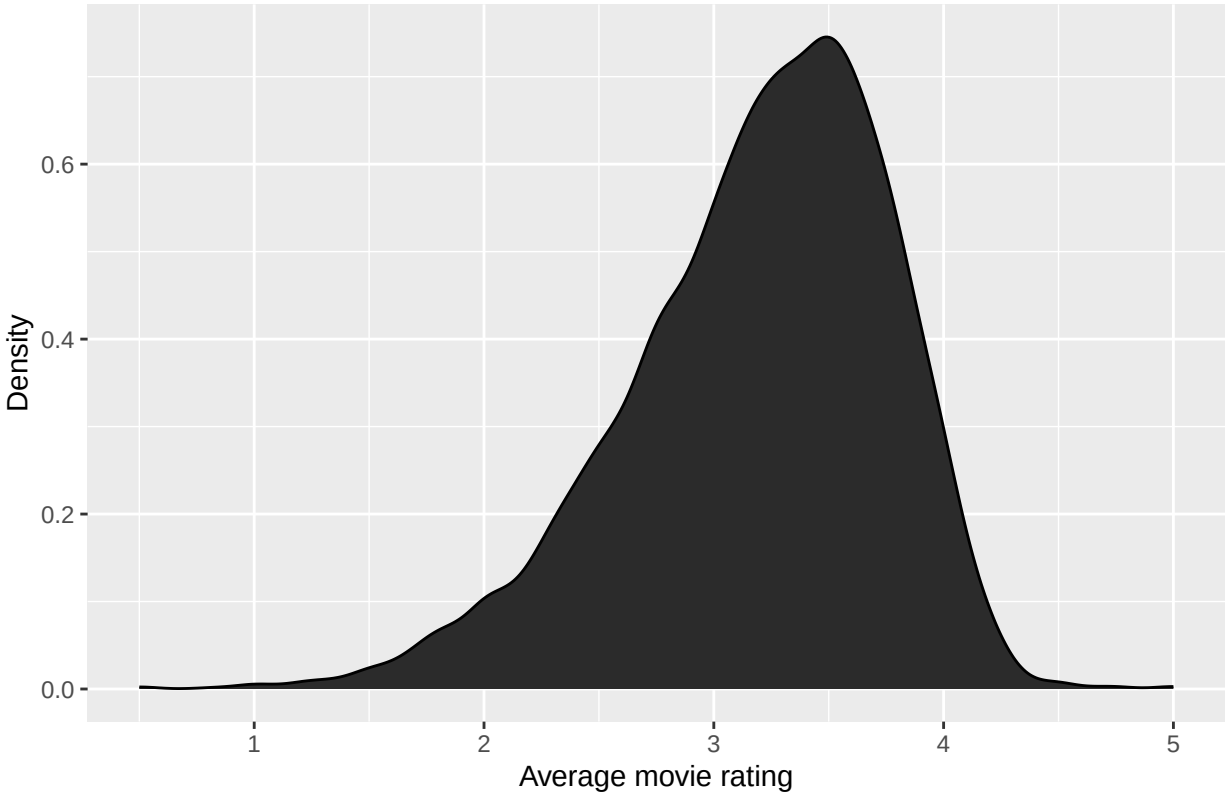
(2) Significant variability of the rating depending on the user rating the movies

(3) The 25% more prolific users in terms of ratings produce almost 70% of the total number of ratings

2.3.3 Movies insights

There are 10677 movies that have been rated. The average rating per movie varies from 0.5 star to 5 stars. Its average is 3.19 stars with a standard deviation of 0.57. The distribution is the following:

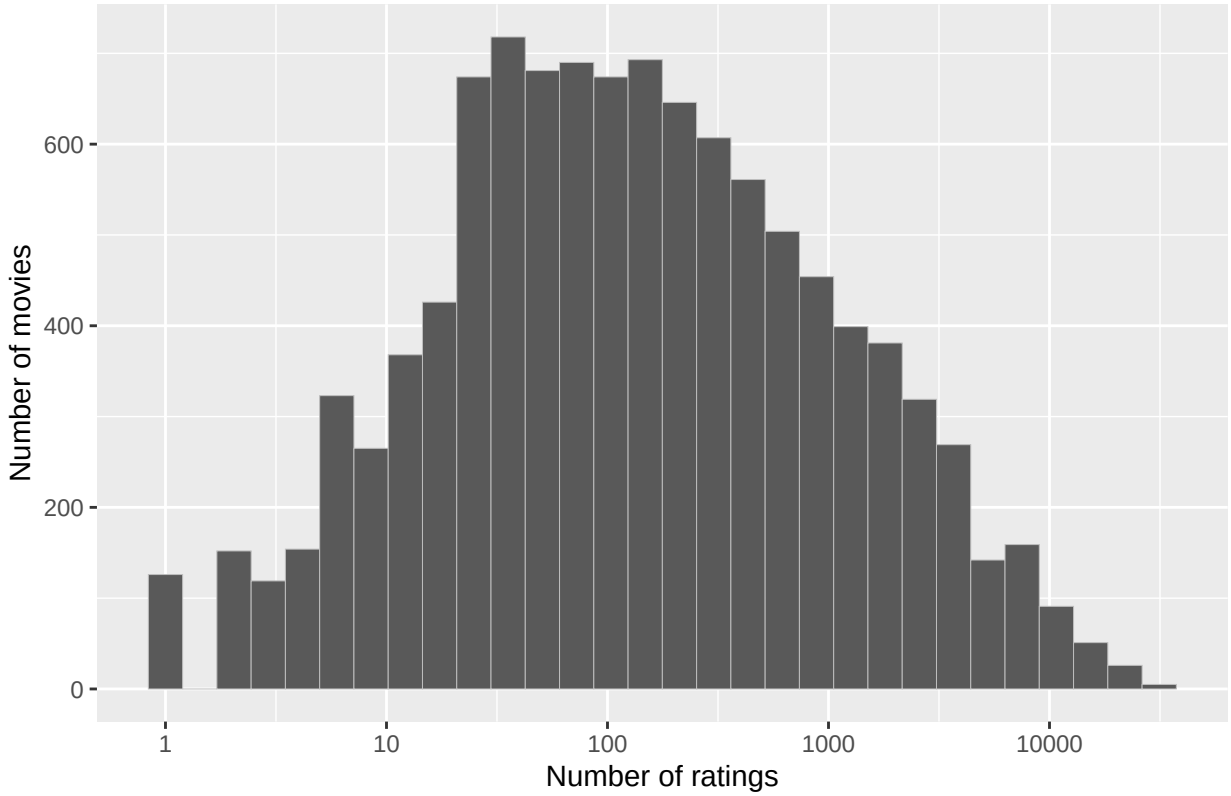
Figure 5: Distribution of average movie ratings



Movies have been rated by a minimum of 1 user and a maximum of 31362 users. On average they have benn rated by 842.94 users with a standard deviation of 2238.48. The median number of ratings per movie is 122.

The distribution is the following:

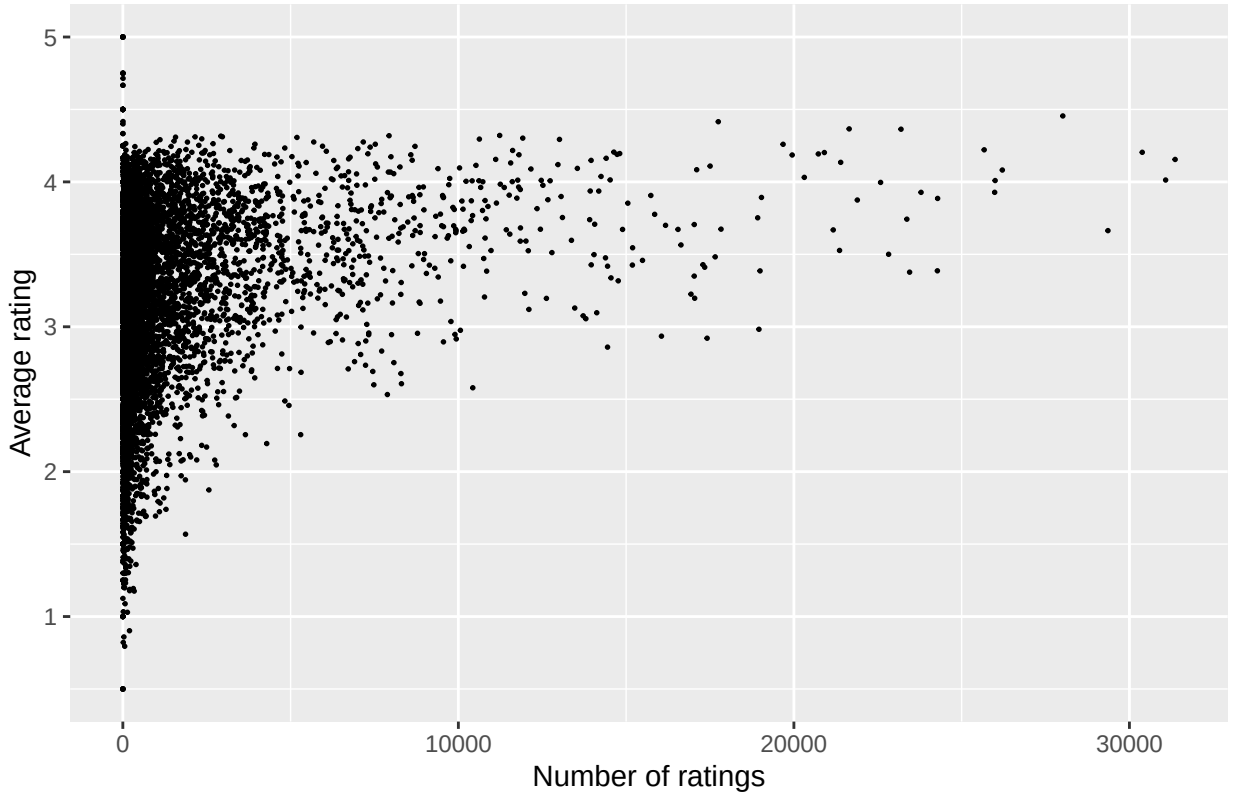
Figure 6: Number of ratings per movie



So we observe a significant variability per movie and we can wonder if the number of ratings per movie affects the rating.

The average rating depending on the number of ratings is the following:

Figure 7: Average rating per number of ratings



This does not suggest a strong correlation between the average rating of a movie and the number of ratings. This is confirmed by the coefficient of correlation 0.21 between the two variables.

Insight(s):

(4) Significant variability of the rating depending on the movie

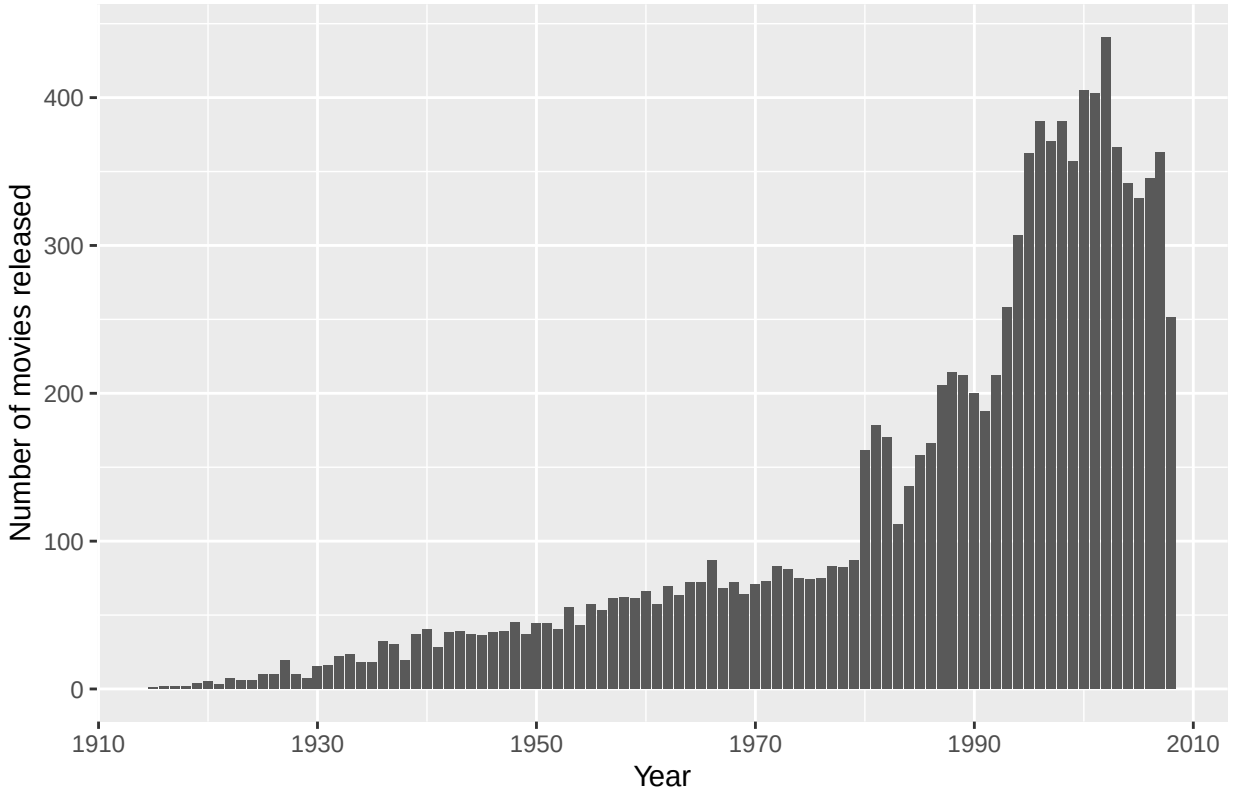
(5) High concentration of movies with 50 to 500 ratings

(6) No clear correlation between the average rating of a movie and the number of ratings

2.3.4 Movie year insights

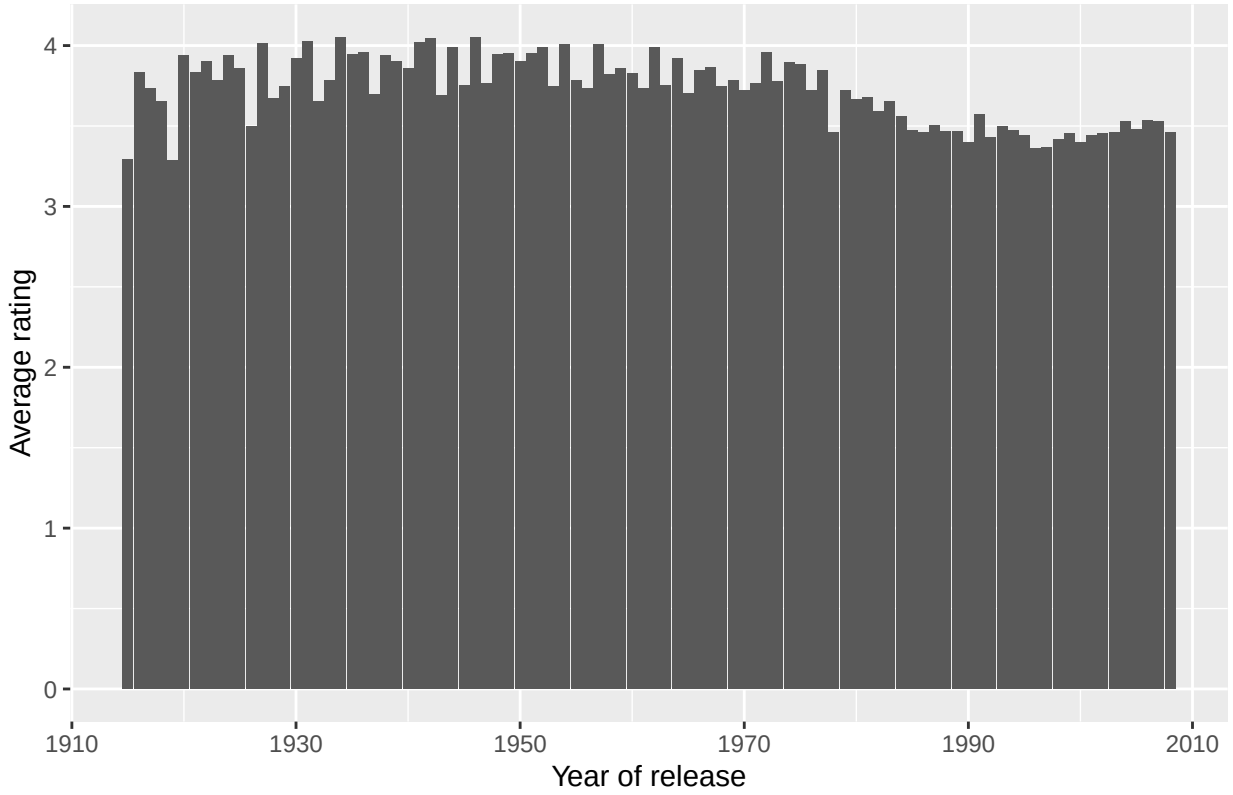
The number of movies released per year is the following:

Figure 8: Number of movies released per year



So we can observe that recent movies are much more prevalent than old movies. We now try to understand if there is a correlation between year of release and rating:

Figure 9: Average rating per release year



This chart suggest that old movies have better rating than recent ones. The correlation suggested from the chart is confirmed by the coefficient of correlation between the year of release of the movie and the average rating -0.58.

Insight(s):

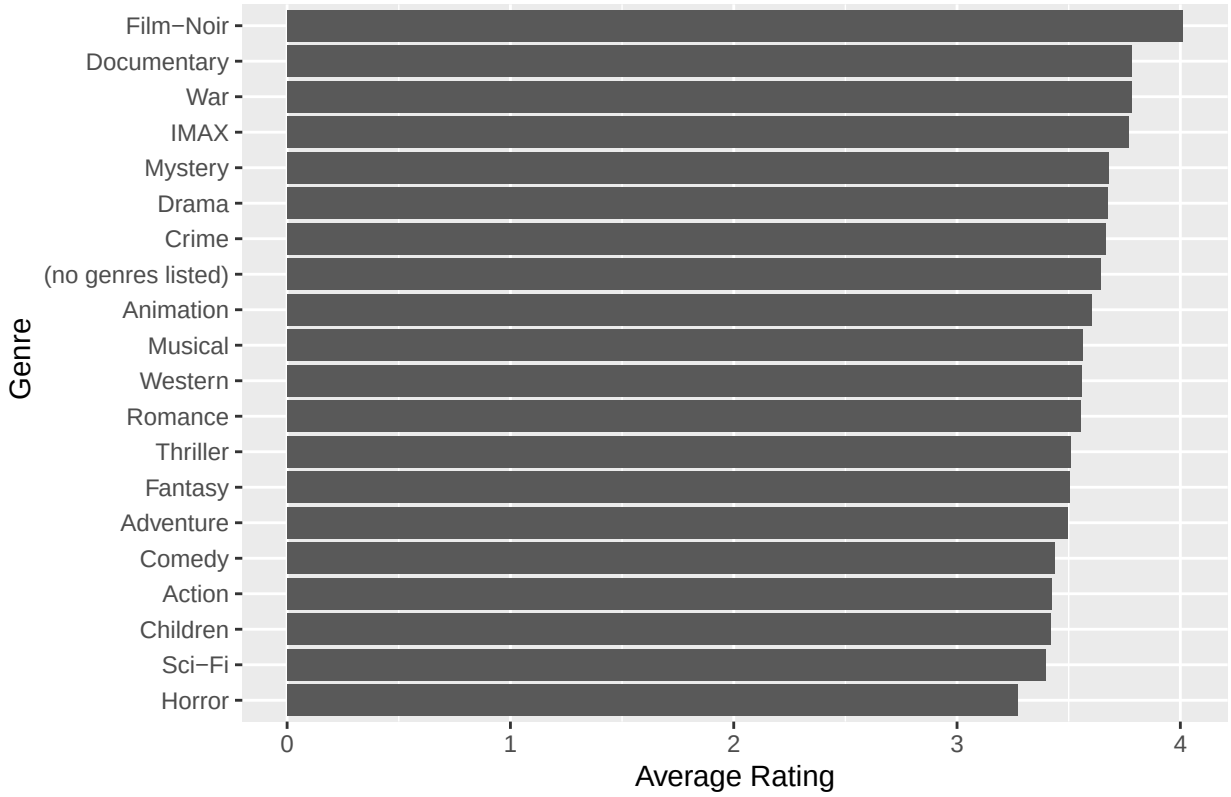
(7) Recent movie have a slightly lower average rating (hovering around 3.5) than older movie (between 3.5 and 4)

(8) High prevalence of recent movies

2.3.5 Movie genres insights

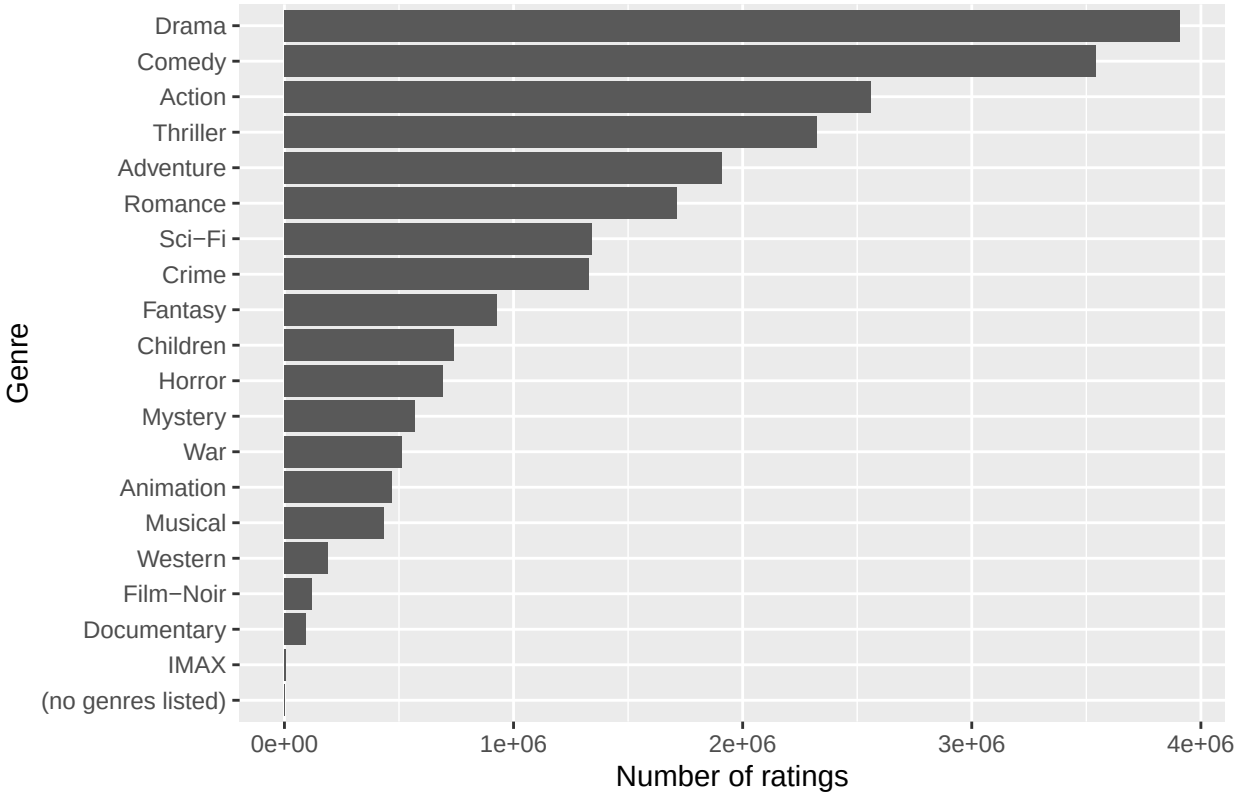
The average rating per genre is the following:

Figure 10: Average rating per genre



We can observe that genres have substantial different average rating. The minimum average rating is 3.27 while the maximum rating is 4.01. The average rating is 3.59 and the standard deviation is 0.17. We now examine prevalence of genres and look at the number of rating per genre:

Figure 11: Number of ratings per genre



Insight(s):

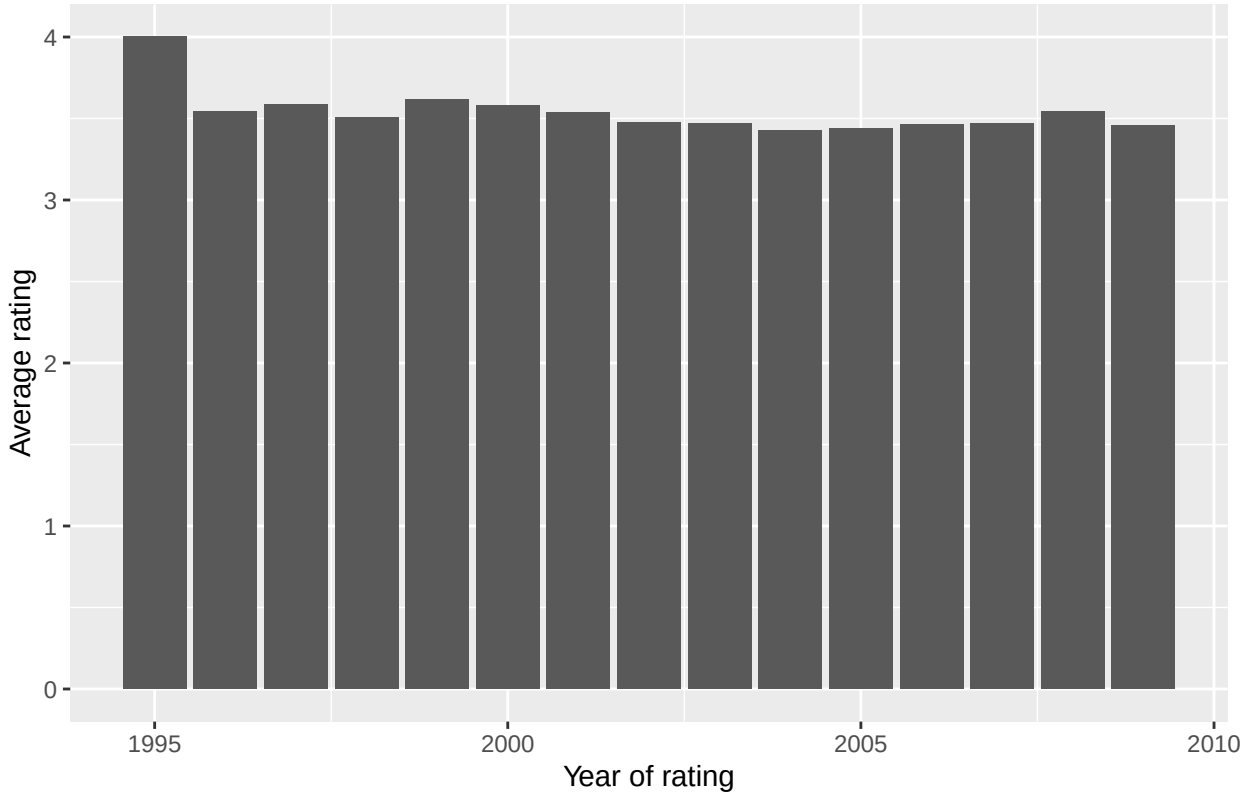
(9) Genre affects rating significantly. The worst rated genre (Horror) has a significant different average rating from the best rated genre (Film-Noir)

(10) High prevalence some genres (Drama, Comedy, Action, Thriller)

2.3.6 Rating year insights

The average rating per year of rating is the following:

Figure 12: Average rating per year of rating



Note that 1995 is not significant of this relates to only 2 ratings.

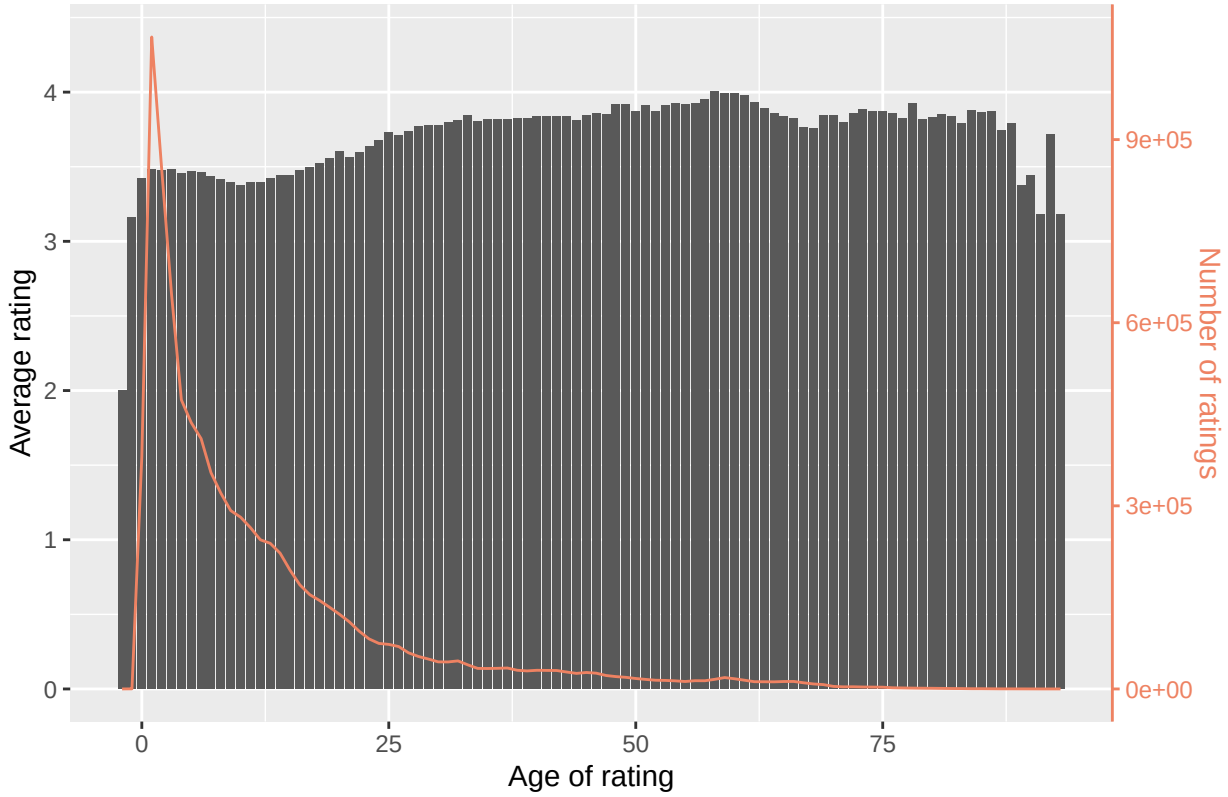
Insight(s):

(11) Rating seems to be stable over years of which movies where rated

2.3.7 Rating age insights

The average rating per age of rating is the following:

Figure 13: Average rating per age of rating



We can observe that rating is influenced by age (coefficient of correlation of 0.49), but the number of observations varies significantly, with a low number of ratings for aged ones.

Insight(s):

(12) Possible correlation between age and rating

2.4 Modelling

We now move on to the modeling. Our objective is to minimize the RSME, hence we will start by calculating the RSME function.

We will then start modelling. The first attempt will be the simplest model we can think off (using the average rating). We will then add possible explanatory variables to the model to improve its performance, meaning trying to reduce the RSME.

2.4.1 Compute RSME function

If we define the following notation:

- $y_{u,i}$ as the rating value of $movie_i$ and $user_u$
- μ as the average rating
- $\varepsilon_{u,i}$ as the independent error
- $b_{\_}$.. as the effect of an additional variable

Variable will have a 'hat' when they represent a prediction (e.g. $\hat{y}$) and will not have when they represent true values.

The RSME is:

$$RSME = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

Which translates into R as:

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

2.4.2 Simplest Model (Model 1)

The simplest model we can build is using the average rating cross the board.

$$Y_{u,i} = \mu + \varepsilon_{u,i}$$

The code is the following:

```
predicted_rating_model1 <- mean(edx$rating)
predicted_rating_model1
```

```
## [1] 3.512465
```

This model give the following RSME:

```
## [1] 1.061202
```

The RSME is quite high (above one star).

2.4.3 Adding the movie effect (Model 2)

Based on insight (4), we can try to improve the model by adding the movie effect. We define a new model by adding a movie specific effect:

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$$

With b_i, the movie specific effect, computed as the average of $Y_{u,i}$ minus the overall mean for each movie i.

The code is the following:

```
mu <- mean(edx$rating)
movie_table <- movie_table %>% mutate (b_i = avg_rating - mu)
predicted_ratings_model2 <- mu + final_holdout_test %>% left_join(movie_table,
by='movieId') %>% .$b_i
```

This model give the following RSME:

```
## [1] 0.9439087
```

This model 2 is indeed an improvement compared to model 1 as the RMSE is lower.

2.4.4 Adding the user effect (Model 3)

Based on insight (2) and (3), we anticipate that the user rating the movie has a large influence on the rating. We can therefore try to improve the model by adding the user effect. We define a new model by adding a user specific effect:

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$$

With b_u, the user specific effect, computed as the average of $Y_{u,i}$ minus the overall mean for each user u.

The code is the following:

```
user_table <- user_table %>% mutate (b_u = avg_rating - mu)
predicted_ratings_model3 <- final_holdout_test %>% left_join(movie_table, by='movieId')
%>% left_join(user_table, by='userId') %>% mutate(pred = mu + b_i + b_u) %>% .$pred
```

This model give the following RSME:

```
## [1] 0.8850398
```

This model 3 is indeed an improvement compared to model 2 as the RMSE is lower, but this is still higher than the objective.

2.4.5 Adding the movie year effect (Model 4)

Based on insight (7), we can try to improve the model by adding the year of rating effect. We define a new model by adding a movie year specific effect:

$$Y_{u,i} = \mu + b_i + b_u + b_t + \varepsilon_{u,i}$$

With b_t, the movie year specific effect, computed as the average of $Y_{u,i}$ minus the overall mean for each year of release of the movie t.

The code is the following:

```
movie_table_summary <- movie_table_summary %>% mutate (b_t = avg_rating_per_year - mu)

movie_table <- movie_table %>% left_join(select(movie_table_summary,movie_year,b_t),
by='movie_year')

predicted_ratings_model4 <- final_holdout_test %>% left_join(movie_table, by='movieId')
%>% left_join(user_table, by='userId') %>% mutate(pred = mu + b_i + b_u + b_t) %>% .$pred
```

This model give the following RSME:

```
## [1] 0.9015546
```

This model 4 is not an improvement compared to model 3, so we do not keep this additional variable and return to model 3.

2.4.6 Adding the genre effect (Model 5)

Based on insight (9), we can try to improve the model by adding the genre effect. We define a new model by adding a genre specific effect:

$$Y_{u,i} = \mu + b_i + b_u + b_g + \varepsilon_{u,i}$$

With b_g, the genre specific effect, computed as the average of $Y_{u,i}$ minus the overall mean for each genre g.

The code is the following:

```
genres_table <- edx %>% mutate(b_ge = rating - mu) %>% group_by(genres) %>% summarize(b_g
= mean(b_ge))

predicted_ratings_model5 <- final_holdout_test %>% left_join(movie_table, by='movieId')
%>% left_join(user_table, by='userId') %>% left_join(genres_table, by='genres')  %>%
mutate(pred = mu + b_i + b_u + b_g) %>% .$pred
```

This model give the following RSME:

```
## [1] 0.9453827
```

This model 5 is not an improvement compared to model 3, so we do not keep this additional variable and return to model 3.

2.4.7 Adding the age effect (Model 6)

Based on insight (9), we can try to improve the model by adding the genre effect. We define a new model by adding a age specific effect:

$$Y_{u,i} = \mu + b_i + b_u + b_a + \varepsilon_{u,i}$$

With b_a, the age specific effet, computed as the average of $Y_{u,i}$ minus the overall mean for each age of rating a.

The code is the following:

```
age_rating_table <- age_rating_table %>% mutate (b_a = avg_rating - mu)

predicted_ratings_model6 <- final_holdout_test %>% left_join(movie_table, by='movieId')
%>% left_join(user_table, by='userId') %>% left_join(age_rating_table, by='rating_age')
%>% mutate(pred = mu + b_i + b_u + b_a) %>% .$pred
```

This model give the following RSME:

```
## [1] 0.8975873
```

This model 6 is not an improvement compared to model 3, so we do not keep this additional variable and return to model 3.

2.4.8 Using regularisation to improve the performance (Model 7)

First we try regularisation on the model 2, to make sure we can improve, before applying to both movie and user effect and optimise the lambda. The code to test the improvement is the following:

```
lambda <- 3
movie_table_regularized <- edx %>% group_by(movieId) %>% summarize(b_i = sum(rating -
mu)/(n()+lambda), n = n())
predicted_ratings_model7test <- final_holdout_test %>% left_join(movie_table_regularized,
by='movieId') %>% mutate(pred = mu + b_i) %>% .$pred
RMSE(final_holdout_test$rating, as.vector(predicted_ratings_model7test))
```

```
## [1] 0.9438538
```

We can see that we get a slight improvement. Therefore, l'ets try to apply to both user and movie effect and optimize lambda. The code is the following:

```
lambdas <- seq(0, 10, 0.25)

rmse_model7 <- sapply(lambdas, function(l){
     mu <- mean(edx$rating)
     movie_table_regularized <- edx %>%
          group_by(movieId) %>%
          summarize(b_i = sum(rating - mu)/(n()+l))
     user_table_regularised <- edx %>%
          left_join(movie_table_regularized, by="movieId") %>%
          group_by(userId) %>%
          summarize(b_u = sum(rating - b_i - mu)/(n()+l))
     predicted_ratings_model7 <-
          final_holdout_test %>%
          left_join(movie_table_regularized, by = "movieId") %>%
          left_join(user_table_regularised, by = "userId") %>%
          mutate(pred = mu + b_i + b_u) %>%
          .$pred
     return(RMSE(final_holdout_test$rating, predicted_ratings_model7))
```

```
})

lambda <- lambdas[which.min(rmse_model7)]

movie_table_regularized <- edx %>% group_by(movieId) %>% summarize(b_i = sum(rating -
mu)/(n()+lambda))
user_table_regularised <- edx %>% left_join(movie_table_regularized, by="movieId") %>%
group_by(userId) %>% summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

predicted_ratings_model7 <-
          final_holdout_test %>%
          left_join(movie_table_regularized, by = "movieId") %>%
          left_join(user_table_regularised, by = "userId") %>%
          mutate(pred = mu + b_i + b_u) %>%
          .$pred
```

This model give the following RSME:

```
## [1] 0.864817
```

This model is a great improvement from model 3 and allows us to reach the objective of RMSE below 0.86490. This is therefore our final model.

3. Results

The modelling results are the followings:

method	RMSE
Simplest model - Model 1	1.0612018
Model with movie effect - Model 2	0.9439087
Model 2 with user effect - Model 3	0.8850398
Model 3 with movie year effect - Model 4	0.9015546
Model 3 with genre effect - Model 5	0.9453827
Model 3 with rating age effect - Model 6	0.8975873
Model 3 regularised - Model 7	0.8648170

We can see the followings:

- Movie and user have a great influence on the rating and including those effects helped to improve the model to a great extent
- Movie year, rating age and genre did not help to improve the model
- The regularization of the model with movie and user effect had a great impact on improving the performance of the model

We select the model 7, which has the best performance and allows to reach the performance expected with a RMSE below 0.86490.

4. Conclusion

The exercise was super interesting. I was especially happy to learn that variables that seem to have correlation with rating (such as movie release year, genre...) are not improving the model when added to the model.

The final model allowed us to reached the expected performance in terms of RMSE.

However, I see two ways to improve further the model:

1. Genre: Based on the data analysis, genre should have a great impact on rating. I believe that the attempt to take it into account in model 5 was not successful because the genres were stacked in one column. I believe that breaking down the column and using adjustment for each genre would probably improve the model.
2. Matrix factorization: This technique would probably improve the model significantly. Indeed the model 7 leaves out the fact that groups of movies and groups of users have similar rating patterns and matrix factorisation would therefore help to improve the performance of the model.